This is to certify that the
dissertation entitled

Analysis and Visualization of Volumetric Data Sets

presented by

Paul Benjamin Albee

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science and Engineering

Major Professor's Signature

19 Aug 2004

Date

DATE DUE	DATE DUE	DATE DUE

Analysis and Visualization of Volumetric Data Sets

By

Paul Benjamin Albee

A Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Department of Computer Science and Engineering

2004

ABSTRACT

ANALYSIS AND VISUALIZATION OF VOLUMETRIC DATA SETS

By

Paul Benjamin Albee

The ability to generate volumetric data sets of materials, particularly at sub-millimeter resolutions, has increased in recent years. The tools to facilitate analysis of non-medical volumes are still primitive and require substantial user input. This thesis addresses three tasks of volumetric analysis, producing efficient algorithms for segmentation, interest detection, and view path generation.

The first major contribution is a clustering based segmentation algorithm that was designed to operate on noisy volumetric data. The initial data is a reconstruction from tomography or MRI and is inherently noisy. The segmentation algorithm operates by detecting statistically similar, although possibly non-contiguous, regions and then relabeling the volume monotonically using a Bayesian classifier based on the detected clusters in the volume.

The second major contribution is the development of a trainable interest detector. A Radial Mass Transform (RMT) is defined to characterize the local structure at each location in the volume. This transform is demonstrated to provide a rich characterization of local structure. The RMT is used as the base input to a Support Vector Machine (SVM) classifier to generate an application specific interest detector. The user selects a set of interesting and uninteresting regions in the volume, the SVM is trained using the labeled data, and the entire volume is classified.

The third major contribution is a view path, or tour, generating algorithm. Once volumes have been processed, a visual representation of the volume is helpful for users to better understand the structure of the data. An algorithm is developed to automatically

generate view paths that maximize the amount of interest shown in a series of cross sections. The cross sections can be viewed as an animation, or virtual tour, providing the user with additional information about the structure of object(s) in the volume.

The algorithms are demonstrated on hundreds of synthetic volumes and dozens of real volumes, including many microvolumes scanned using the Argonne National Laboratory APS. Two common threads running through this work are that all techniques were designed to work on large volumes and to operate in parallel when possible. A typical data set is on the order of 400–600 megabytes and may require several hours, if not days, of processing time on a desktop computer. The techniques we have developed are amenable to parallel processing, reducing processing time from days to hours or minutes.

To Laura

ACKNOWLEDGMENTS

This dissertation marks the conclusion of several years of effort. I would like to thank my advisor, Professor George C. Stockman, for his encouragement and support. I would also like to thank my committee members, Professor Juyang Weng, Professor William Punch, Professor Alvin Smucker, and Dr. Mark Rivers, for their input and suggestions.

A significant portion of this work was made possible with support from Argonne National Laboratory. The staff at GeoSoilEnviroCARS and SRI-CAT at the Advanced Photon Source were generous both in terms of time and the provision of data. I would especially like to thank Ian McNulty and Francesco DeCarlo at SRI-CAT.

At Michigan State University I would like to thank the staff of the Department of Computer Science and Engineering. Thanks to Linda Moore for providing encouragement. Thanks to Adam Pitcher for his help making compute resources available for the parallel experiments.

Members of the PRIP lab have provided valuable input and comments over the years. I would like to thank Arun Ross and Anoop Namboodiri in particular for their input on the interest detection and view path problems.

Thanks to Jim Siebert from the Department of Radiology at Michigan State for providing the spinal data and a detailed explanation of how MRI imaging works.

The work done in this thesis was supported in part by the U.S. Department of Energy under contract no. W-31-109-Eng-38. The data used for this work was collected at GSE-CARS, Sector 13 at the Advanced Photon Source and at SRI-CAT, Sector 2 at the Advanced

Photon Source.

Bee stinger data was provided by Dr. Andrew Peele at the University of Melbourne, Australia. The scutigera data was provided by Dr. Martin Fanenbruck at Ruhr University Bochum, Germany.

The coffee shops in East Lansing provided an excellent work environment and a steady supply of coffee.

Finally I'd like to thank my wife Laura, my parents Paul and Bonita Albee, and friends David and Stacy Hammond for their support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Images in this dissertation are presented in color.

Chapter 1

Introduction to the problem

Advances in volumetric data acquisition have allowed researchers to easily collect data on objects and materials that were previously unable to be evaluated. One large class of data is non-medical data that is currently being acquired via x-ray tomography at facilities such as the Advanced Photon Source (APS) at Argonne National Laboratory (ANL). The APS is a high energy x-ray source that can be used to image materials that in the past were too dense or too small for conventional x-ray tomography. Many of the volumes being collected at the APS and elsewhere have unknown internal structures reducing the usefulness of those analysis techniques that require extensive *a priori* knowledge. My work is to develop automatic and semi-automatic tools to assist the scientists in evaluating this new data. The tools being developed will enable the user to semi-automatically evaluate a tomographic volume. The advantage of using such tools is that the user's attention can then be guided to points of interest within the volume that they may either miss, or simply not have time to find manually. The work presented in this dissertation was originally motivated by micro-tomographic soil analysis. As the work progressed, other imaging modalities, e.g. MRI and optical slices, were considered, leading to the development of algorithms aimed at general volumetric data.

1.1 Medical versus Non-medical Volumetric Imaging

The discipline of volumetric data analysis can be broken into two broad areas, medical and non-medical volumetric imaging. The fundamental difference between these areas is the existence of a priori knowledge of object features. In medical analysis there is a large amount of a priori knowledge with respect to what features are expected. An example of this would be examining brain images. The shape of the structures in the brain vary slightly from person to person, but in general the structures are similar. It is this property that allows analysis tools to be developed, e.g. Duta [28], Cootes and Taylor [18], Bookstein [8], that learn the shape of the structures of interest. Once the tools have learned the shape, they can be used to locate and extract instances of the shape from other images.

In non-medical volumetric imaging, depending on the area of interest, there is a wide range in the amount of available prior knowledge. The amount of prior knowledge can range from complete knowledge of what should be present to no knowledge about the object. Objects about which there is complete knowledge include microchips and micro-machined objects. In both of these cases, CAD models may be available that can be used for analyzing the objects. Partial knowledge may be available for biological samples, like fruit flies and ants. Like the brain example in medical imaging the insects are similar within species with only small variations. However, no knowledge about structure is available for objects such as soil aggregates and soil percolation tests. The lack of knowledge about these internal structures is due to the processes that form the materials. Objects without any available knowledge of their internal structure are interesting since they are often formed through poorly characterized natural processes. It is hoped that studying these natural structures will help scientists to better understand the processes that form the structures.

1.2 Segmentation

In order to to analyze structures within an object, they must first be located and extracted from the object. This process of *segmentation* is one of the first steps in most volumetric analyses. Segmentation is particularly troublesome when there is no knowledge of the internal structure of the object to be segmented. The lack of prior knowledge of the internal structure requires that several different segmentation techniques be tried.

1.3 Interest Detection

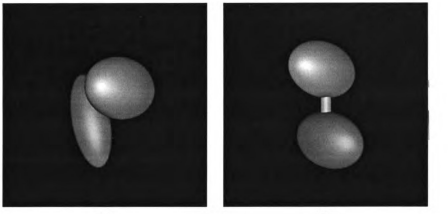
Once regions have been segmented from an object, features need to be extracted from these regions. *Feature extraction* can refer to something as simple as getting the location of each discrete component or subregion in the segmented region. Feature extraction can also refer to collecting several high level metrics such as bounding region, moments, statistical properties, or related measures, that characterize the segmented region. Feature extraction coupled with good segmentation is vital to performing any meaningful analysis, although the features themselves may be sufficient for analysis.

Some feature extraction can also be done on non-segmented objects, or prior to segmentation. Among the types of features that can be extracted are population statistics for the object, determination of the presence of structure within the object, and data quality assessment. Rather than develop a set of specific features, a technique for identifying domain specific interest is developed in Chapter 5.

1.4 Viewpoint Selection

One of the major problems with visually evaluating volumetric data is locating good viewpoints. When viewing volumetric data, there can be an excessive amount of visual information presented to the user. When looking at volumetric data, it is important to look at

it from the right viewpoint. This is illustrated in Figure 1.1(a) and Figure 1.1(b). There appear to be two different objects, but a connecting tube is occluded in Figure 1.1(a). Once the viewpoint has been changed, the connection is visible as shown in Figure 1.1(b).



(a) (b)

Figure 1.1: (a) Sample objects with apparently two separate objects. (b) Sample objects showing connection, only one object.

1.5 Thesis Contributions

There are three major contributions in this dissertation: n-ary volume segmentation, interest detection, and view path determination. The overall goal is to develop a set of analysis techniques appropriate to volumetric data and to implement the techniques as tools that can then be used for analyzing volumetric data.

The first contribution is a clustering based algorithm for performing n-ary segmentation on volumetric data sets. The segmentation algorithm is designed to operate on large noisy volumetric data sets. The algorithm generates an n-ary segmented volume with monotonic labels: two voxels labeled $j < k$ implies the voxel with label j was less dense in the original volume than the voxel labeled k. The segmented results are suitable for noise reduction, contrast enhancement, illustration generation, and as input to later processing algorithms.

The second contribution is the development of a user trainable interest detection framework. The first component of the interest is based on the Radial Mass Transform, a novel

feature analogous to both the Radon and Hough transforms. The RMT provides a rich description of the volume, and provides a rotation- and translation- invariant characterization of features within a volume. The second component is an application of a Support Vector Machine classifier providing a convenient method for the user to train interest point detection.

The third contribution is a view path generation algorithm for generating smooth tours through a volume that maximize the presentation of interesting regions within a volume.

1.6 Thesis Outline

The organization of this dissertation is as follows. Chapter 2 presents a literature review of data collection, preprocessing, segmentation, volume representation, feature extraction, registration, and visualization techniques. Initial techniques for segmenting volumes and extracting features from the volumes are presented in Chapters 4 and 5. Chapter 6 presents an algorithm for automatically generating tours of a 3D volume. Chapter 7 summarizes the contributions of this dissertation and outlines future work.

Chapter 2

Literature Review

This chapter describes some of the issues and techniques related to the analysis of tomographic volumes. Tomographic analysis can be broken into the following steps: data collection, preprocessing, volumetric reconstruction, segmentation, volume representation encoding, feature extraction, registration, and visualization. Section 2.1 discusses the data collection process for tomographic imaging. Basic issues related to projection preprocessing are given in Section 2.2. Once the volume has been reconstructed, the objects of interest need to be segmented from the volume. Several different techniques for segmentation are discussed in Section 2.3. The segmented volume can be represented in a variety of ways; some of the common techniques are described in Section 2.4. After the volume has been reconstructed, segmented, and stored, features of interest need to be extracted from the volume. Some feature types and extraction techniques are in Section 2.5. In the case of the same sample being used in difference acquisition cycles, features in the volume need to be registered to detect changes from data set to data set. Finally, visualizing the data helps the user to see the results of analysis that has been done on the volume and to draw insights from the data. Several visualization techniques are described in Section 2.7.

2.1 Data Collection

Data collection is the very first step in doing any sort of volumetric analysis. There are currently four generations of x-ray tomography data collection devices used in the world, but they all operate in essentially the same manner.

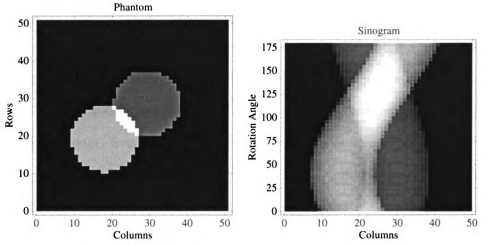
Ultimately, data collection can be viewed as generating a set of projections $\mathcal{P} = \{p_0, p_1, \ldots, p_{n-1}, p_n\}$ of an object $\mathcal{O}$. In Figure 2.1(a) a two dimensional phantom is shown. Figure 2.1(b) shows what is known as a sinogram. The sinogram is constructed by taking the set of projections $\mathcal{P}$ as a image. The projection process can be modeled using the Radon transform [56].

Given a function, the Radon transform can be used to construct a projection of $f(x, y)$ from some angle θ. The Radon transform is constructed by taking a series of line integrals along a rotated axis, and combining them to produce a one dimensional projection of the function $f(x, y)$. The equation for the line integral is given in Equation 2.1 where δ is the Dirac Delta function.

$$\mathrm{P}_\theta(t_1) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\,\delta(x\cos\theta + y\sin\theta - t_1)\,dxdy \tag{2.1}$$

The line integral along $\mathrm{P}_\theta(t_1)$ is computed for all values of t_1. The rotation in the Dirac function is used to rotate the line integral while the value t_1 is used to select the distance of the Dirac function from the origin.

The mechanical setup for microtomographic imaging is shown in Figure 2.2. The sample is placed on a pin mounted on a rotation stage. The sample is rotated through 180 degrees in the x-ray beam. Images are taken of the scintillator screen at regular intervals measuring the x-ray attenuation through the sample.



(a) Cross Section (b) Sinogram

Figure 2.1: Synthetic slice phantom for an intersecting bar phantom and the corresponding **sinogram**.

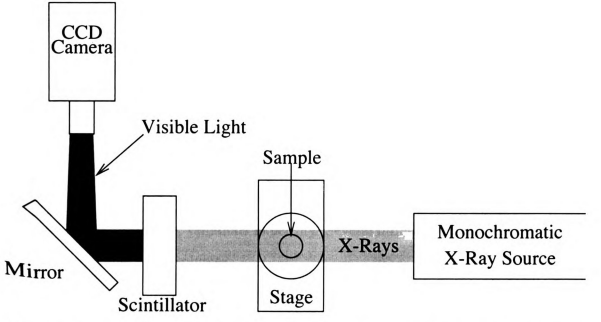


Figure 2.2: Micro-tomographic data acquisition setup. This is a schematic of the imaging setup at GSECARS, Sector 13, at the APS at ANL.

2.2 Preprocessing

Preprocessing is used to remove noise from the data prior to reconstruction. There are several types of standard preprocessing that can be done. Among these are Shepp-Logan [99] filtering, zinger suppression, bar-suppression, white field and black field removal, and projection alignment [90]. Shepp-Logan filtering is done by reducing the contribution of low frequency elements from reconstructions. Zinger suppression is done to remove pixels from the projection where an x-ray, or other high-energy photon, struck and saturated a sensing element. Bar removal is the process of removing straight line bars from the sinogram [83]. The bars in the sinogram correspond to sensor in-homogeneities. Black field and white field normalization are done to remove the sensor noise floor and x-ray beam structure respectively.

2.3 Segmentation

Segmentation is the process of extracting objects or regions of interest from a image, or, more generally, a function of N dimensions. Simple thresholding techniques that rely on a single metric do not work well in most applications. In tomography and MRI they do not work for several reasons. Chief among the reasons is the problem of noise in the data. The reconstruction process is an inherently noisy operation due to a combination of sensor noise, imprecise alignment of acquired data, partial voxel coverage, and inexact floating point arithmetic during reconstruction. Additionally, there are problems associated with smooth, or nearly smooth, variations in density. Several different techniques have tried to handle segmenting noise volumes.

2.3.1 Snakes

Snakes, also referred to as active contours, are energy driven boundary detectors. Snakes allow a contour to be placed near a boundary, then forces are applied to the snake to make

it more closely follow the boundary. Forces that affect the evolution of a snake are smoothness constraints, boundary length constraints, and image forces. The basic form of a snake, given by Kass et al. in [57], is an energy minimizing contour. The contour is defined as a spline that tends to minimizes internal bending energy, either minimizes or maximizes length, and maximizes image energy. The original formulation for the snake energy E_{snake} for a contour $\mathbf{v}$ from [57] is given in Equation 2.2.

$$E_{snake}(\mathbf{v}(s)) = \int_{s=0}^{1} \alpha(s) \left| \frac{\partial \mathbf{v}}{\partial s} \right|^2 + \beta(s) |\frac{\partial^2 \mathbf{v}}{\partial s^2}|^2 - \gamma |\nabla I(\mathbf{v})| ds \tag{2.2}$$

The first two terms in Equation 2.2 relate to the continuity and smoothness of the snake. The third term deals with the energy of the image along the snake, in this formulation image energy is the gradient of the image under the snake. $\alpha, \beta,$ andγ are weighting parameters. The snake is initially positioned in the image outside of the region to be segmented. Some sort of gradient descent algorithm is then applied to move the snake to the boundary of the region that is being segmented.

Additional work related to snakes has been related to devising better energy functions by Xu et al. in [112], [113], dual active contours by Gunn et al. in [43], adding statistical shape information by Leventon et al. in [61], and dynamic programming to support hard constraints by Amini et al [1]. McInerney and Terzopoulos develop an active surface the can change topology targeted at medical volumes [72].

2.3.2 Clustering

Clustering defines a set of exemplar patterns or feature sets that represent a group of samples in feature space. A set of features based on the data to be clustered are generated and then used in the clustering algorithm. Clusters are generated by minimizing intra-cluster distances and maximizing inter-cluster distances: additionally, there may be a target number of clusters.

For clustering-based segmentation, image features such as intensity, intensity variance, and texture statistics can be used. Once the clusters have been identified and classified, the image elements can be classified into different categories. Examples of clustering algorithms are given in [54]. Indicator Kriging [77, 108] attempts to binarize a volume by learning two distributions, and then applying a classifier that incorporates spatial distance from a known classified voxel end estimated class prior probabilities.

2.3.3 Connected Components Analysis

Connected components analysis (CCA) is a technique for identifying components in a labeled image. A connected component is a collection of image elements that have the same label and are adjacent to each other. A labeled point that is not part of a component in the region to be labeled is selected and assigned a component label. This point is then used as a seed point, or starting point, for propagating the new component label. The neighbors of the seed point are then examined. Neighbors with the same label that are not part of any component are assigned the component label and added to the set of seed points. After all of the seed point's neighbors have been examined, it is removed from the set of seed points. The propagation continues until there are no more seed points. An example of propagation is shown in Figure 2.3.

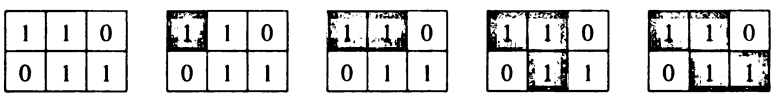


Figure 2.3: Example of how labels propagate during connected components analysis.

CCA is used when the image has already been labeled. In the case of tomographic data, the volume may have already been labeled; the next step is to extract the components and generate per component statistics. Some of the simple statistics that can be generated are the centroids of each component, minimum enclosing rectangular prisms, major and minor axes of the components, sizes of the components, and component size distribution.

2.3.4 Region Growing

Region Growing (RG) is closely related to connected components analysis. Rather than working on a labeled image to grow components, RG operates on unlabeled images and generates regions based on some similarity measure. The simplest similarity measure is requiring that adjacent elements have exactly the same value, indicating that connected components analysis can be considered a special case of region growing.

Two of the major factors affecting region growing are the choice of similarity measure and the choice of neighborhoods. Similarity can be specified as a distance between two elements. Depending on the application, a threshold is used to determine how far apart two elements are before they are no longer considered similar. In the case of a binary image, the distance threshold is set to zero. For images with scalar values at each element, the threshold may often be some value greater than zero. For the case of vectors at each element, some sort of distance metric can be computed, then a scalar threshold can be applied. The similarity threshold does not necessarily need to be held constant. The threshold may vary based on component statistics or some correction factor to allow for global image variations, e.g. non-uniform sensor response. The choice of neighborhood affects both the shape of the components and the speed at which points are aggregated. In two dimensions, edge neighbors and vertex neighbors are defined. Edge neighbors are those neighbors where there is a common edge between the elements, while vertex neighbors share only one vertex. In three dimensions there are also face neighbors, which share a face. In higher dimensions there are additional types of neighborhoods.

2.3.5 Iso-Surface Extraction

An iso-surface is a surface where the image values of all the points on the surface are the same. A variety of techniques exist for generating iso-surfaces. One of the larger classes is the set of marching methods. Among these are marching cubes [66], marching triangles [48], and level sets and fast marching methods [96]. The marching cubes and

0	0	0	0	0	0	0	0	0
1	1	1	2	2	2	2	2	1
1	2	2	2	2	3	2	1	0
0	2	1	2	2	3	2	1	0
0	1	1	1	2	2	1	1	0
0	2	1	2	2	1	1	0	0
0	0	0	0	0	0	0	0	0

Figure 2.4: Iso-surface for value 2.

marching tetrahedron algorithms work by locating where the desired iso-values are located in a volume. An example of an iso-boundary is given in Figure 2.4. Viola et al. [106] treat volumetric data sets as texture, and then apply thresholds to filtered versions of the data.

2.4 Representation

There are several broad categories of representation for storing reconstructions: raw volume reconstructions, lossless compressed reconstructions, lossy compressions and tree encodings. The data may also be stored as a set of iso-surfaces, generalized cylinders, or other processed representations that store the data as a set of features. Finally the data may be stored in some transform space, such as a Fourier or wavelet space, a medial axis representation, or a union of R where R is some primitive such as a sphere or ellipsoid. The nature of the representation will affect the fidelity of the volume to the original data, how the data must be accessed for efficient operations, and the types of analysis that can be performed on the volume.

The discrete space representation of volumetric data is described as simple, yet verbose and structureless in [101]. A technique for converting the discrete data into a partitioning tree is presented. The advantage of using a partitioning tree representation is that it allows for relatively simple axis aligned data removal. That is, rectangular regions aligned with the data axes are easily removed to reveal previously obscured details.The partitioning

tree approach is closely related to octree encoding where the volume is encoded using a hierarchical occupancy description. See [89] for a discussion of octrees.

2.5 Feature Extraction and Analysis

One of the main goals of segmentation is to identify features within the data. Additionally, many features cannot be computed without the data being segmented. Among these features are volume, surface area, medial axes, fractal dimension, and tortuosity. Westerman and Ertl combine segmentation and feature detection into a wavelet based rendering technique [109]. Rohr provides an overview of feature extraction [86] and landmark based analysis techniques [85].

2.5.1 Skeletons in the Volume

The skeleton of an object is a set of points that remain after a modified form of erosion. When the object is eroded, the removal of points is constrained such that a point is not removed if its removal would cause the object containing it to be broken into two or more objects. The process of identifying the skeleton can be further constrained by specifying the type of neighborhood connectivity required. Closely related to the image skeleton is the medial axis (MA) [100]. The MA is defined in terms of continuous images where a set of points is selected that are equidistant from the edges of the containing object. For a binary object, the MA representation can be used to completely reconstruct the object. In the discrete representation, the MA is not precisely defined, but can be approximated reasonably well. Frequently the MA and the skeleton are used interchangeably. While they are not identical, they are typically close to each other and can be used interchangeably in practice.

There are several categories of techniques for computing the skeleton or MA of a volume. The most direct method is constrained binary erosion [87]. While constrained binary

erosion is conceptually simple, it is expensive since it is an iterative process and becomes more expensive as the perimeter of the objects increase. Another class of techniques described by Russ in [87] is based on thresholding distance maps of the volume being segmented with constraints on connectivity and local maximas. The constraints on connectivity ensure that the extracted skeleton is connected. Additionally, the local maximas in the distance map actually correspond to the axis or skeleton, and must be preserved during the thresholding operation.

A technique that utilizes two different distance maps for computing the skeleton of a volume is presented by Zhou et al. in [115]. A distance map based on a modified Manhattan distance map, called the ssField, is computed with respect to a point likely to be on the skeleton. A second map, the bsField, that approximates a Euclidean distance map is computed for the same volume. The ssField is used to generate a set of clusterings of voxels where each cluster contains one point in the skeleton. The values in bsField that correspond to a cluster in the ssField are used to select the medial point of each cluster. In addition to generating the skeleton of the object, the graph representation used for skeleton construction encodes the presence of holes within the volume.

Once the skeleton has been extracted from the volume there are several techniques for deriving further information about the volume for the MA representation.

2.5.2 Extended Gaussian Image

The Gaussian image (GI) is a technique for encoding directional information from a function in a position independent manner. Conceptually the GI can be thought of as taking the gradient at some position in a function, and then projecting it onto the surface of a sphere. An example EGI for a soil aggregate is shown in Figure 2.5. The Extended Gaussian Image (EGI) is defined and discussed in [51]. The EGI has several interesting properties; among these are ease of computation and uniqueness for convex objects.

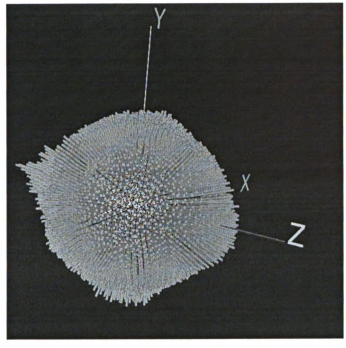


Figure 2.5: Gaussian Image for a soil aggregate.

2.6 Registration

When change detection is done on volumes, registration between different acquisition cycles must be done. In the case of volumes generated from materials such as soil aggregates, there are no explicit fiducial points. The reason for the lack of fiducial points is that the act of placing fiducial points in the samples would act to damage or destroy the samples. Other materials may not be amenable to fiducial placement because of issues related to beam hardening, closeness to x-ray absorption edges for either the sample or the fiducial marker, or the fiducial markers can not remain in place during the treatment of the samples between acquisition cycles.

Interest operators have been defined for two dimensional data. Typically the interest operators work by identifying high variance regions in the image, then computing a transform that will align corresponding points in multiple images. See Chapter 9 of [98] for more details on interest operators and their applicability to image registration. Rohr provides a discussion of a wide range of feature detectors and interest operators in [86].

2.7 Visualization

The results from reconstruction, feature extraction, and analysis need to be presented in a form that is meaningful to a person. This need leads to the problem of visualization. There are three main parts to the visualization problem — display technique, efficiency, and interaction.

A variety of techniques exist for presenting volumetric data to a user. At a high level there are projective techniques and (quasi-)immersive techniques. Projective techniques are those that present the user with a two dimensional projection of the data being visualized [32]. Immersive and quasi-immersive techniques are techniques that present the user with a three-dimensional display of the data. Immersive displays can range from a desktop system with a pair of LCD glasses to a fully immersive environment like a CAVE[20]. These display devices, coupled with adequate compute resources, allow the user to "move" through the displayed data. The motion may be some predetermined path, or may be driven by interactive navigation [11].

For more detailed information on visualization, refer to [94]

Chapter 3

Proposed Research

3.1 Introduction

There are three major contributions proposed for this thesis: volume segmentation, interest detection, and view path generation. The overall goal is to develop a set of analysis techniques appropriate to volumetric data and to implement the techniques as tools that can then be used for analyzing volumetric data.

3.2 Volume Segmentation

The first subproblem to address is segmenting volumes. One set of samples used in this thesis are tomographic volumes of soil aggregates. With a soil aggregate there is a large amount of internal structure. The pores, or air and water channels, within the aggregate are the structures of interest to soil scientists. These structures need to be extracted from the aggregate and quantified in some meaningful fashion. The extraction of the pores requires identifying not only the pore boundary, but also the aggregate boundary and the pore endpoints where the pore breaks the surface of the aggregate. An accurately quantified pore network can be used to support simulation of transport phenomena.

The primary problem associated with soil aggregate segmentation is the lack of *a priori*

structure knowledge. The only thing known about the structure of a soil aggregate prior to evaluation is that it has some sort of structure. Techniques that are able to segment the different materials in a soil aggregate are needed as one of the first steps in analysis.

Tools that are developed to evaluate soil structures should also be applicable for analyzing other types of volumetric data. Some of the classes of data where these techniques might be applicable are as follows. Some classes of medical data could be evaulated where there is no *a priori* structure knowledge, specifically bone and fine tissue structure. Failed microchips could be analyzed; while a working chip is highly structured, the flaws that lead to chip failure may not have a characteristic structure. Composite materials are becoming common in industrial and commerical applications, and are sensitive to flaws in their microstructure. Catastrophic failures would begin at these microstructure flaws and propagate through the material, causing an integrity failure.

3.3 Interest Detection

The second subproblem to address is detecting interesting regions within a volume. One of the major problems associated with volumetric data, or any large data set, is finding the areas of interest within the data set. In a large volume there may be features of interest that are not visible from most viewpoints, may be small and therefore easy to miss, or may simply be hard to identify visually. Interest detectors that can detect salient features within the volume can assist users in making more efficient use of their time. Techniques to detect interest should be adaptable to different applications, and flexible enough to allow the use of a wide range of features.

3.4 View Path Generation

The third subproblem is generating a view path through the volume that covers the interesting regions within the volume. When viewing a volume, some viewpoints are better than others. If the appropriate volume cross sections and viewpoints are selected, features of interest can then be presented to the user. If poor viewpoints are selected, then important features may be missed. Since rendering a large volume takes a significant amount of time, help in selecting optimal, or nearly optimal, viewpoints will help to minimize the amount of time spent looking for good viewpoints.

In a large volume it is likely that there will be more than one viewpoint that satisfies some optimality or interest criteria. Ideally the user will be able to select which interest and optimality criteria are going to be used. The software will then extract a set of viewpoints that meet the user's requirements. Once the viewpoints have been selected, they need to be presented to the user. The simple approach is to simply show each view to the user. Simply stepping through viewpoints does not necessarily show any inter-viewpoint information and in an immersive display system will likely disorient the user. To alleviate these problems, a good viewing path between different viewpoints can be constructed. The viewing path would have to satisfy smoothness, continuity, and length constraints in order to be considered as a good path. The advantage of a viewing path that satisfies some smoothness constraints is that the constraints can be constructed so that a user in an immersive system does not become disoriented or nauseated.

Chapter 4

Segmentation

In this chapter two segmentation techniques are explored. The first technique is based on the Extended Gaussian Image (EGI) [51], while the second is based on single-link clustering. A set of criteria for segmenting volumetric data sets is developed. The EGI based algorithm is developed, evaluated, and found to not satisfy the criteria. The single-link clustering algorithm is then developed, evaluated, and found to satisfy the algorithm criteria. The single-link algorithm is then used to segment a collection of test volumes.

4.1 Problem Description

Segmentation is the problem of separating one or more objects in an image from the background. The goal of developing techniques to facilitate volumetric data analysis leads to the following set of criteria for the segmentation algorithm:

1. Simplicity of computation.

2. Robust in the presence of noise.

3. Allows for n-ary segmentation, in addition to binary segmentation.

4. Generates intermediate state information that enables a scientist to drive the segmentation towards a desired goal.

These criteria are discussed in the following sections.

4.1.1 Simplicity of computation

A typical reconstructed volume acquired at the APS is on the order of 512 megabytes in size. Future data sets may grow to be as large as 4 gigabytes for a full resolution volume. With these data sizes in mind, simplicity of computation encompasses two factors. The first factor is that any algorithm should be computationally affordable. An algorithm that requires more than polynomial effort is unacceptable. The second factor is that any algorithm should minimize the number of iterations through the data. An additional desirable, but not necessary, property of an algorithm is for it to be parallizable.

4.1.2 Robustness

The volumes being segmented are noisy due to both the original data collection environment and also the reconstruction process itself. During the original image acquisition several noise processes are present. These processes include scattered radiation from the x-ray source, electrical noise in the sensor, defects in the sensor, variations in the scintillator crystal, beam hardening and crystalline and positioning errors in the sample holder. During the reconstruction process, noise can also be introduced through missing samples in the data set, imprecise floating point arithmetic, poor filter selection, inability to correct for imaging errors, centering correction error and inappropriate scaling. Given the wide range of possible noise sources, it is necessary to construct algorithms that behave well in the presence of noise, and in particular should take globally derived information into account rather than strictly local information.

4.1.3 N-ary segmentation

While simply extracting object(s) from a volume via binary segmentation is useful, it is desirable to be able to perform some sort of hierarchical segmentation. Taking into account the size of data mentioned previously, being able to perform n-ary segmentation, where $n \geq 2$ for multiple values of n, from a single segmentation cycle allows us to maximize the amount of information we can generate. In the soil samples used to develop and validate these techniques, more than one type of material is present. A segmentation algorithm that provides only binary segmentation does not allow us to readily extract the different materials from the volume, an example of this is shown in Figre 4.11.

4.1.4 Intermediate state information

The segmentation algorithm should allow multiple segmentation cycles from a single initial processing stage. This ability would allow a scientist to explicitly identify the classes of materials detected during the initial segmentation stage as either object or background as well as perform segmentation analogous to band-pass filtering where background features may be discontinuous. The practical application is not only excluding the medium the sample is in, but also excluding materials in the volume that may not currently be of interest.

4.2 Segmentation Techniques

4.2.1 Extended Gaussian Image/Polyhedral Histogram

The Extended Gaussian Image (EGI) is one where the observed gradients of a signal or function are computed, and then mapped to quantized spherical coordinates [51, 63, 67]. A straightforward extension is projecting quantized gradients onto the facets a polyhedron, thus producing a type of histogram using the facet indices as bin indices. The resulting map can then be used to detect symmetries (see [102]) by detecting ring-like protrusions

for rotational symmetries or paired protrusions at the antipodes of the polyhedra. Sun [102] uses a tessellated icosahedron with the triangular facets merged to produce a set of hexagons and pentagons for the surface mesh. This approach helps to reduce the amount of computation when generating the EGI at the cost of having projection regions with different areas. Rather than using the EGI to detect symmetries, we are using it to detect the presence of features within a volume.

There are three main motivations for using the EGI or polyhedral histogram (PolyHist). The first is a compact encoding of approximate gradient information. PolyHist utilizes two scalar fields. The first field is a vector magnitude field, denoted as **VMF**, computed from gradients in the input volume. The second field is the facet encoding field, denoted by **FEF**. The values in the **FEF** correspond to the facet indices of the polyhedra being used to generate the PolyHist. The **FEF** provides an approximation for the gradient vector at each sample point, as well smoothing out much of the noise inherent in gradient approximations of sampled functions.

The second motivating factor is the ability of the EGI to be used in a multi-resolution mode. With each subdivision of a triangular facet, the resolution of the histogram is increased by a factor of four. With careful bookkeeping, lower resolution EGIs can be computed from higher resolution EGIs. Conversely, the lower resolution tessellations can be used to speed up the initial generation of the EGI by using a hierarchical search of the facets. Additionally, some sort of fractal analysis of the gradient direction may be possible by taking advantage of the different tessellation levels.

The third motivating factor is that EGIs are simple to compute. The value in the **FEF** is set to the index of the facet whose normal would produce the largest value when dotted with the gradient vector. After the **FEF** has been generated, the EGI of any sub-volume can be computed by examining the **FEF** and the **VMF**. The dual-field representation also allows for thresholds based on vector magnitude, integer histograms, and continuous histograms, all from the same representation. Additionally, the major cost of computing

the EGI is paid once, while several different analyses can then be performed.

Example EGIs

EGIs encode information about the direction of gradients and their magnitude. In general EGIs are computed for surfaces, but we have instead computed them for any gradient above some threshold within a volume. The first EGI example is for a cube phantom, shown in Figure 4.1. Note the dominant gradient energies are aligned with the coordinate axes. The

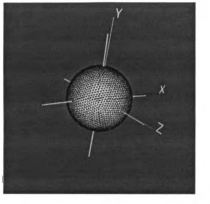


Figure 4.1: EGI for cube phantom. All gradient energy in the EGI is approximately aligned to the coordinate axes.

second example EGI is from a cylinder phantom, shown in Figure 4.2. The phantom is constructed in such a way that the cylinder passes through the volume so there are no ends in the volume. The radial symmetry of the cylinder produces an EGI where all the energy is on a narrow band that circles the polyhedron. The third example EGI is from a sphere phantom, shown in Figure 4.3. The energy for this phantom is distributed approximately uniformly. The variation in the energy distribution is due to the use of a gradient estimator, and the fact that a significant number of the phantom voxels are partially occupied, contributing to gradient estimation error. The fourth example EGI is from a soil aggregate, and is shown in Figure 4.4. The aggregate is asymmetric, and this asymmetry is reflected in the asymmetrical distribution of energy in the EGI.

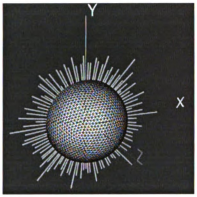


Figure 4.2: EGI for cylinder phantom.

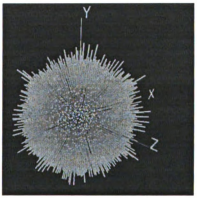


Figure 4.3: EGI for sphere phantom.

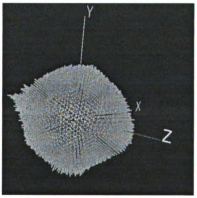


Figure 4.4: EGI for soil aggregate.

PolyHist Field Generation

As noted previously, two fields, **VMF** and **FEF**, need to be generated. The gradient is estimated using the Zucker-Hummel gradient estimator given in [5]. Equation 4.1 defines the neighborhood where the estimator will be applied. Equation 4.2 shows the convolution that computes the partial derivative for x of the neighborhood.

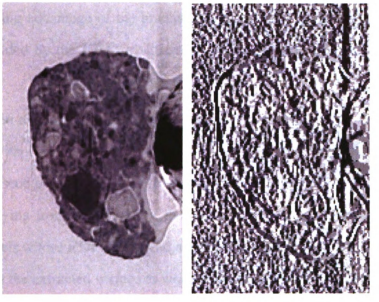
$$\mathrm{F}(x,y,z) = 3 \times 3 \times 3 \text{ Neighborhood centered at } (x,y,z) \tag{4.1}$$

$$\frac{\partial \widehat{\mathrm{F}}(x,y,z)}{\partial x} = \mathrm{F}(x,y,z) \otimes \left[-\begin{bmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{2}}{2} & 1 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \end{bmatrix} \quad [0] \quad \begin{bmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{2}}{2} & 1 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \end{bmatrix} \right] \tag{4.2}$$

The estimators for $\frac{\partial \widehat{\mathrm{F}}(x,y,z)}{\partial y}$ and $\frac{\partial \widehat{\mathrm{F}}(x,y,z)}{\partial z}$ are produced by rotating the convolution matrix in Equation 4.2 appropriately. Once the gradient has been estimated, the magnitude is computed and used to normalize the gradient vector. The magnitude is then stored in the **VMF**, while the normalized gradient vector is intersected with the polyhedron being used for the EGI. The resulting facet index is stored in the **FEF**. Figure 4.5(a) shows a sample aggregate slice whose **FEF** and **VMF** are shown in Figure 4.5(b) and Figure 4.5(c), respectively. The EGI for the sample aggregate is shown in Figure 4.4.

PolyHist Generation

After the fields have been produced, the histograms must be generated. The simplest histogram is one where each index value in the **FEF** is counted. This procedure will give a histogram that indicates if there is any dominant direction in the volume being considered.



(a) Input Slice (b) FEF

(c) VMF (d) Components

Figure 4.5: Soil Aggregate

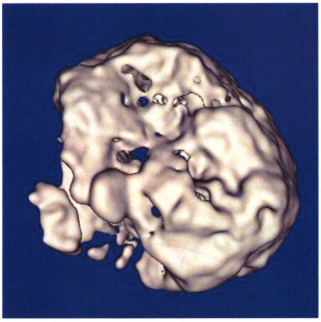
Component Extraction

Once the encoding fields have been generated, it is possible to extract boundaries from the volume. By taking advantage of the gradient magnitudes in the **VMF** and connectivity constraints provided by the **FEF**, a variant of region growing can be performed on the encoding fields.

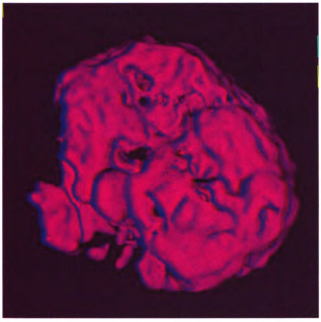
The extraction algorithm operates by searching the encoding fields until an unlabeled voxel is found with a magnitude exceeding some user specified threshold. Using the unlabeled voxel as a seed point, a region is grown that tracks the boundary. In order to keep the region from straying from the boundary, the region is allowed to grow only by including neighboring voxels whose **FEF** value is adjacent to the **FEF** value for the seed voxel. This technique allows the extracted surface to wrap around objects, but prevents the merging of voxels whose gradients point in radically different directions. Figure 4.5(d) shows a cross section of the soil aggregate with the extracted components.

Extracted Interfaces

The following sets of figures are surfaces extracted from a soil sample volume using the EGI based technique described in this section. Figure 4.6(a) and Figure 4.7(a) show the largest component extracted from the volume. This component corresponds to the outer surface of the aggregate and is similar to the view from an optical microscope. Figure 4.8(a) and Figure 4.9(a) show the internal pores that were detected. With the pores separated from the aggregate, they are now available for use in generating population statistics related to pore size and distribution. The (b) subfigure for Figures 4.6 – 4.9 are anaglyphs that are used for 3D viewing. An anaglyph is a stereo-pair rendered by combining red and blue images. A pair of glasses with red and blue filters are used to provide a separate to each eye.

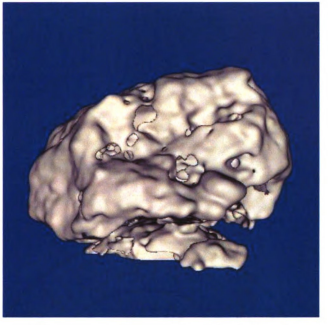


(a) Outer surface of aggregate

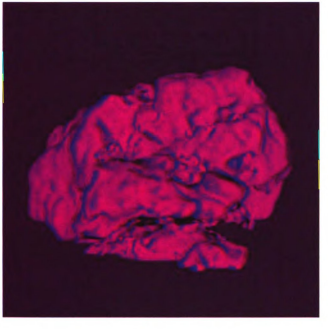


(b) Anaglyph of outer surface of aggregate

Figure 4.6: View of external boundary extracted from soil sample volume.

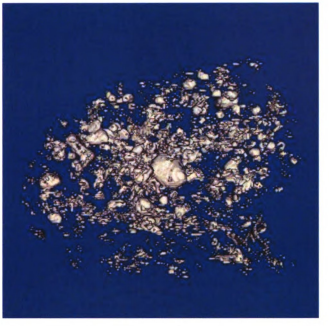


(a) Outer surface of aggregate

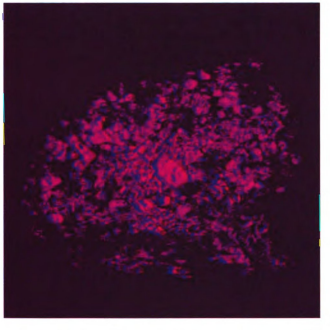


(b) Anaglyph of outer surface of aggregate

Figure 4.7: Alternate view of external boundary extracted from soil sample volume.

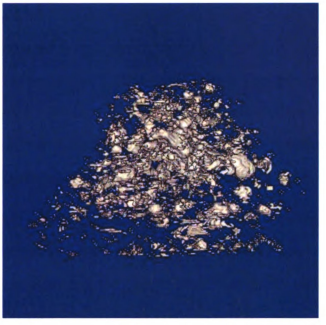


(a) Internal pore boundaries

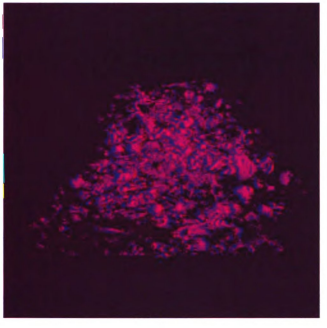


(b) Anaglyph of internal pore boundaries

Figure 4.8: View of internal pores extracted from soil sample volume.



(a) Alternate view of internal pore boundaries



(b) Anaglyph of internal pore boundaries

Figure 4.9: Alternate view of internal pores extracted from soil sample volume.

Suitability of PolyHist

The PolyHist approach works reasonably well in extracting strong interfaces from a volume. However, while interface extraction is valuable, the algorithm suffers from being extremely expensive. During the construction of the encoding fields, two more full-resolution volumes are generated to encode the magnitude and directions of the EGI. After these fields are computed, the region-growing algorithm traverses the data in a depth-first manner. This traversal technique results in potentially inefficient memory accesses.

In terms of satisfying the criteria established in Section 4.1, this algorithm fails to meet criteria 2 and 3. The robustness criteria is not met since the computation of the EGI is susceptible to noise, and the n-ary segmentation criteria is not met since interfaces are detected rather than regions.

4.2.2 Thresholds

In order to segment an aggregate from the surrounding medium, a threshold is used. A global threshold is used to clip the entire aggregate from the volume. A histogram of the intensities of the voxels is generated from the input volume. Treating the histogram as a continuous function, a first derivative is computed and used to identify maxima and minima in the histogram. The two largest local maxima are identified. The threshold is then set to the intensity with the minima point between the maxima. If the actual data is well-approximated by a bi-modal Gaussian distribution, then the threshold will be the smallest minima between the maxima. Figure 4.10 shows an example of a placing the threshold in a histogram of a Kenyan soil aggregate. The first derivative is shown, along with the locations of the two assumed distribution means, and the minima between them.

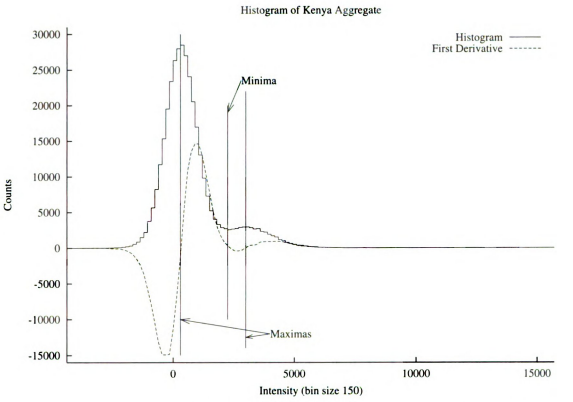
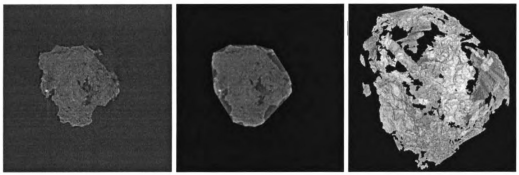


Figure 4.10: Histogram of Kenyan aggregate with minima and maxima marked.

Extract Aggregate From Background

The aggregate is extracted from the background in order to define a boundary for surface breaking pores. The boundary of the aggregate is determined using a convex hull on a per slice basis. Areas outside of the hull are indicated using a sentinel value. The initial separation of the soil aggregate from the background is done by thresholding a median-filtered version of the slice. This threshold produces a binary image of the aggregate and some noise in the background around the aggregate. Erosion and dilation operations are then applied to the binary image to remove the noise in the background region. A *Roberts* edge detector is then applied to the cleaned image. Strong edge points identified by the edge detector are then used to construct a convex hull. Once the hull has been constructed, all points outside of the hull are labeled with a sentinel value to indicate that they are not part of the soil aggregate.



(a) Input (b) Segmented (c) Extracted Pores

Figure 4.11: Aggregate Data

An input image is shown in Figure 4.11a, and the segmented image is shown in Figure 4.11b. Note that there are background regions around the edges of the aggregate in the segmented image. These spaces may correspond to either surface breaking pores or concavities in the surface of the soil aggregate. Once all the slices are processed, they are recombined to generate the segmented soil aggregate volume, which is shown in Figure 4.11c.

Suitability of Simple Threshold

A simple threshold is not suitable since it fails to meet criteria 3 in Section 4.1. While a simple threshold may work to binarize the data, it does not readily allow for n-ary segmentation. The assumption that the data falls into two classes also causes this algorithm to fail the n-ary segmentation criteria.

4.3 Proposed Modified Single-Link Algorithm

We propose a segmentation algorithm that develops a set of Gaussian distributions describing the volume using a non-spatial clustering algorithm.

4.3.1 Description of Algorithm

The segmentation algorithm is essentially a single-link clustering algorithm, similar to algorithms discussed in [54], coupled with a Bayesian classifier. This algorithm operates by extracting a set of non-spatially-related clusters from the input volume. The features being clustered are the mean and variance of all non-overlapping $M \times M \times M$ neighborhoods. The variance and mean, $\sigma^2_{neigh}, \mu_{neigh}$, are computed for each neighborhood. A neighborhood is merged with an existing cluster if it satisfies the condition

$$\frac{|\mu_{neigh} - \mu_{cluster}|}{\max\left(\sigma^2_{neigh}, \sigma^2_{cluster}\right)} < \alpha \tag{4.3}$$

where α is a similarity parameter set by the user. The merging process involves updating $\mu_{cluster}$ and $\sigma^2_{cluster}$ for the cluster that satisfies the merge criteria. If the condition is not satisfied, a new cluster is created using σ^2_{neigh} and μ_{neigh}. This process continues until all voxels have been clustered. Additionally, a forcing step may be performed. Forcing merges clusters by iteratively taking the two closest clusters, by $\mu_{cluster}$, from the set of all clusters and merging them until a user-indicated target number of clusters has been reached. Details are given in Algorithm 1.

Once the data has been clustered, the original volume is classified using a Bayesian classifier, described in [27]. Currently the cluster with the lowest mean is set as the background; all other clusters are set as foreground. The user of the software can iteratively segment subvolumes to determine the appropriate assignment of clusters as foreground or background. After the cluster classes have been assigned each voxel in the original volume is classified.

Algorithm Validation

Since the techniques developed in later chapters depend on good segmentation, the proposed segmentation algorithm was validated using both synthetic data and real data.

Algorithm 1: Clustering Algorithm
Input: The volume $Volume$ to be clustered, distance threshold α, clustering window width N, forcing flag $ForceFlag$, desired number of clusters $NumClusts$
Output: Ordered table containing clusters

CLUSTER($Volume, \alpha, N, ForceFlag, NumClusts$)
(1) $rows$ = ROWS($Volume$)
(2) $cols$ = COLS($Volume$)
(3) $slices$ = SLICES($Volume$)
(4) $ClustTab = \emptyset$
(5) **foreach** $r \in \{0, \ldots, rows | (r - \lfloor N/2 \rfloor \bmod N) = 0\}$
(6) **foreach** $c \in \{0, \ldots, cols | (c - \lfloor N/2 \rfloor \bmod N) = 0\}$
(7) **foreach** $s \in \{0, \ldots, slices | (s - \lfloor N/2 \rfloor \bmod N) = 0\}$
(8) $CandClust$ = GENCANDCLUST($Volume, r, c, s, N$)
(9) $ClustRef$ =
(10) FINDCLOSEST($ClustTab, CandClust, \alpha$)
(11) **if** $ClustRef \neq$ **NULL**
(12) $ClustRef$ = UPDATECLUST($ClustRef, CandClust$)
(13) **else**
(14) $ClustTab = ClustTab \cup CandClust$
(15) **repeat**
(16) $anymerged$ = **false**
(17) **foreach** $CurrClust \in ClustTab$
(18) **repeat**
(19) $merged$ = **false**
(20) **foreach** $TestClust \in ClustTab$
(21) **if** $CurrClust \neq TestClust$
(22) **if** WITHINTHRESH($CurrClust, TestClust, \alpha$)
(23) $anymerged$ = **true**
(24) $merged$ = **true**
(25) $CurrClust$ =
(26) UPDATECLUST($CurrClust, TestClust$)
(27) $ClustTab = ClustTab \setminus TestClust$
(28) **until** $merged$ == **false**
(29) **until** $anymerged$ == **false**
(30) **if** ($ForceFlag$ == **true**) **and** ($|ClustTab| > NumClusts$)
(31) **while** $|ClustTab| > NumClusts$
(32) $ClustPairs = \{\{a, b\} | \forall a, b \in ClustTab$ **and** $a \neq b\}$
(33) $(ClustA, ClustB)$ = CLOSESTPAIR($ClustPairs$)
(34) $ClustA$ = UPDATECLUST($ClustA, ClustB$)
(35) $ClustTab = ClustTab \setminus ClustB$
(36) $ClustTab$ = SORTBYCLUSTMEAN($ClustTab$)
(37) **return** $ClustTab$

Simulated Data

The segmentation algorithm was initially validated using a suite of synthetic phantoms. Phantoms were used in order to provide absolute ground truth. The basic validation procedure was to generate a synthetic volume, and add Gaussian noise to the volume in increasing levels. After the synthetic volumes were prepared, the clustering algorithm was applied, followed by the segmentation algorithm. Confusion matrices were then generated for the resulting labeled and segmented volumes.

Each phantom simulated a mass of homogeneous rods. The generation algorithm drew a random number of randomly oriented straight lines through the volume. Once a set of lines were drawn, a Euclidean distance map was used to compute the distance of each voxel from the lines through the volume. The distance map was then thresholded such that all voxels with a distance greater than an arbitrary threshold were set to zero, while the remaining voxels were set to one. A sample volume is shown in Figure 4.12.

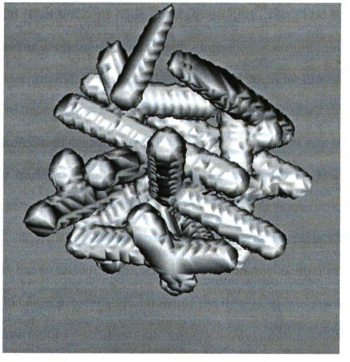


Figure 4.12: Homogenous rod phantom, rendering of volume iso-surface.

Phantom Testing Methodology

The phantom was used to provide insight into the performance of the segmentation algorithm in the presence of noise. To exercise the segmentation algorithm, the synthetic data set along with sets of target Gaussian distributions was used to generate noisy data sets to segment. After the noised phantoms were generated, the clustering algorithm was applied for a range of α values. The noised volumes were then labeled and confusion matrices were generated.

Performance on Phantom

The results from the experimental runs described above for the rod phantom are given in Tables 4.1 – 4.8. The timing information given in Tables 4.1, 4.3, 4.5, and 4.7 is the time in seconds per voxel in the volume. The values in the Tables 4.2, 4.4, 4.6, and 4.8 are the number of clusters detected in the volumes. The timing information and the cluster counts are the average of 20 repetitions for each set of parameters. The same set of seeds was used for noise generation for each set of parameters.

In this set of experiments, there were three parameters. The first parameter was the α parameter, while the second and third parameters were the standard deviations, σ_1 and σ_2 respectively, for the noise generators. Holding σ_1 and σ_2 constant, we observe that the time per voxel decreases as the α parameter increases. For the same σ_1 and σ_2 we also see that the number of clusters detected decreases. This behavior is consistent with the clustering algorithm allowing larger inter-cluster distances for merging clusters as the α parameter increases. The decrease in execution time is a consequence of a smaller average cluster set, reducing the amount of time needed to determine the disposition of each new neighborhood as the volume is processed.

The effect of changing the α parameter can be seen in Figures 4.13 – 4.16. The original data had two clusters with the distributions $\mu_1 = 0, \sigma_1 = 0.2$ and $\mu_2 = 1, \sigma_2 = 0.2$ respectively. The corresponding confusion matrices are given in Tables 4.9 – 4.12. Shown

Table 4.1: Average time to detect clusters with $\alpha = 0.25$ over 20 trials. Time is given in seconds per voxel.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.147	0.145	0.142	0.141	0.139	0.133	0.132	0.129	0.127	0.124
0.2	0.134	0.145	0.143	0.133	0.136	0.135	0.131	0.125	0.127	0.124
0.3	0.142	0.140	0.139	0.135	0.132	0.133	0.130	0.127	0.126	0.123
0.4	0.138	0.137	0.135	0.132	0.130	0.127	0.129	0.126	0.124	0.122
0.5	0.137	0.135	0.133	0.130	0.127	0.127	0.124	0.124	0.124	0.119
0.6	0.131	0.131	0.128	0.128	0.124	0.124	0.124	0.122	0.121	0.121
0.7	0.131	0.129	0.128	0.128	0.124	0.124	0.124	0.121	0.120	0.119
0.8	0.125	0.126	0.127	0.123	0.123	0.121	0.121	0.122	0.120	0.119
0.9	0.120	0.123	0.123	0.121	0.120	0.119	0.119	0.120	0.118	0.117
1.0	0.120	0.121	0.121	0.120	0.119	0.119	0.119	0.116	0.118	0.118

Table 4.2: Average number of clusters detected with $\alpha = 0.25$ over 20 trials.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	9.4	8.7	8.6	8.2	8.0	7.5	7.0	7.2	6.7	6.7
0.2	8.8	8.6	8.2	8.0	7.8	7.5	7.1	6.8	6.5	6.3
0.3	8.3	8.2	8.0	7.8	7.6	7.3	6.8	6.5	6.6	6.3
0.4	8.2	8.0	7.8	7.7	7.3	7.0	6.7	6.5	6.6	6.4
0.5	7.6	7.5	7.4	7.1	7.0	6.6	6.5	6.5	6.3	6.1
0.6	7.2	7.3	7.0	6.8	6.7	6.5	6.5	6.3	6.0	5.8
0.7	7.2	6.9	6.7	6.6	6.7	6.3	6.3	6.2	6.0	5.7
0.8	6.2	6.5	6.5	6.5	6.3	6.2	6.0	5.9	5.8	5.7
0.9	5.8	6.5	6.3	6.2	6.2	5.8	5.8	5.5	5.5	5.5
1.0	5.8	6.2	6.1	6.0	5.9	5.8	5.7	5.7	5.5	5.5

Table 4.3: Average time to detect clusters with $\alpha = 0.50$ over 20 trials. Time is given in seconds per voxel.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.099	0.102	0.103	0.102	0.101	0.099	0.098	0.096	0.096	0.095
0.2	0.094	0.105	0.104	0.095	0.103	0.101	0.099	0.099	0.099	0.095
0.3	0.099	0.105	0.102	0.103	0.101	0.100	0.101	0.098	0.097	0.094
0.4	0.099	0.104	0.103	0.101	0.100	0.099	0.097	0.095	0.095	0.096
0.5	0.099	0.101	0.100	0.100	0.098	0.099	0.097	0.095	0.097	0.096
0.6	0.098	0.102	0.100	0.099	0.097	0.098	0.096	0.095	0.096	0.094
0.7	0.096	0.098	0.095	0.098	0.096	0.096	0.096	0.096	0.098	0.094
0.8	0.091	0.098	0.097	0.097	0.098	0.097	0.095	0.093	0.095	0.093
0.9	0.090	0.092	0.094	0.095	0.097	0.096	0.094	0.095	0.097	0.094
1.0	0.089	0.091	0.094	0.094	0.097	0.095	0.095	0.095	0.095	0.094

Table 4.4: Average number of clusters detected with $\alpha = 0.50$ over 20 trials.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	3.2	4.0	4.0	4.1	4.0	3.6	3.5	3.2	3.1	3.0
0.2	3.2	4.0	4.2	4.1	4.0	3.6	3.5	3.4	3.2	3.1
0.3	3.2	4.0	4.0	4.0	3.9	3.5	3.5	3.4	3.3	3.1
0.4	3.2	4.0	4.0	4.0	3.5	3.5	3.4	3.4	3.2	3.2
0.5	3.3	4.0	4.0	3.7	3.6	3.4	3.2	3.2	3.2	3.1
0.6	3.6	3.9	3.7	3.5	3.5	3.5	3.3	3.3	3.1	3.0
0.7	2.8	3.3	3.3	3.4	3.4	3.3	3.3	3.1	3.0	3.0
0.8	2.4	3.2	3.3	3.3	3.2	3.2	3.0	3.0	3.0	3.0
0.9	2.3	2.9	3.1	3.1	3.1	3.1	3.0	3.0	3.0	3.0
1.0	2.2	2.5	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0

Table 4.5: Average time to detect clusters with $\alpha = 0.75$ over 20 trials. Time is given in seconds per voxel.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.090	0.090	0.091	0.092	0.090	0.089	0.091	0.088	0.090	0.088
0.2	0.090	0.091	0.092	0.083	0.092	0.091	0.085	0.090	0.090	0.080
0.3	0.090	0.089	0.091	0.091	0.091	0.088	0.089	0.089	0.087	0.088
0.4	0.089	0.088	0.088	0.089	0.091	0.090	0.088	0.090	0.088	0.086
0.5	0.089	0.090	0.089	0.090	0.090	0.089	0.089	0.088	0.090	0.086
0.6	0.087	0.088	0.089	0.090	0.090	0.089	0.089	0.089	0.089	0.088
0.7	0.088	0.086	0.090	0.088	0.090	0.088	0.088	0.088	0.090	0.086
0.8	0.089	0.089	0.088	0.088	0.089	0.089	0.087	0.089	0.088	0.089
0.9	0.089	0.088	0.089	0.089	0.088	0.089	0.087	0.089	0.088	0.087
1.0	0.087	0.088	0.086	0.089	0.089	0.088	0.087	0.087	0.088	0.089

Table 4.6: Average number of clusters detected with $\alpha = 0.75$ over 20 trials.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	2.0	2.2	2.5	2.6	2.5	2.5	2.3	2.2	2.1	2.1
0.2	2.0	2.1	2.5	2.6	2.5	2.5	2.4	2.2	2.1	2.1
0.3	2.0	2.1	2.4	2.5	2.5	2.4	2.4	2.2	2.1	2.1
0.4	2.0	2.1	2.2	2.5	2.4	2.3	2.3	2.2	2.1	2.1
0.5	2.0	2.0	2.1	2.3	2.3	2.3	2.3	2.2	2.1	2.1
0.6	2.0	2.0	2.0	2.2	2.3	2.3	2.2	2.2	2.1	2.1
0.7	2.0	2.0	2.0	2.1	2.2	2.2	2.2	2.2	2.1	2.1
0.8	2.0	2.0	2.0	2.0	2.1	2.1	2.1	2.1	2.1	2.0
0.9	2.0	2.0	2.0	2.0	2.0	2.0	2.1	2.0	2.0	2.0
1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0

Table 4.7: Average time to detect clusters with $\alpha = 1.0$ over 20 trials. Time is given in seconds per voxel.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.087	0.086	0.088	0.085	0.087	0.087	0.088	0.086	0.089	0.087
0.2	0.091	0.091	0.087	0.088	0.090	0.089	0.081	0.088	0.090	0.082
0.3	0.089	0.088	0.088	0.088	0.088	0.088	0.086	0.086	0.086	0.087
0.4	0.088	0.088	0.088	0.089	0.088	0.089	0.088	0.087	0.086	0.086
0.5	0.087	0.089	0.088	0.086	0.087	0.087	0.088	0.089	0.086	0.085
0.6	0.088	0.088	0.088	0.088	0.089	0.087	0.087	0.088	0.089	0.087
0.7	0.088	0.088	0.088	0.087	0.087	0.088	0.087	0.087	0.085	0.087
0.8	0.088	0.088	0.088	0.086	0.088	0.086	0.087	0.088	0.088	0.088
0.9	0.081	0.083	0.086	0.087	0.090	0.086	0.086	0.084	0.088	0.089
1.0	0.077	0.080	0.081	0.083	0.084	0.088	0.086	0.085	0.088	0.087

Table 4.8: Average number of clusters detected with $\alpha = 1.0$ over 20 trials.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.3	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.4	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.5	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.6	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.7	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.8	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
0.9	1.0	1.1	1.6	2.0	2.0	2.0	2.0	2.0	2.0	2.0
1.0	1.0	1.0	1.1	1.4	1.9	1.9	2.0	2.0	2.0	2.0

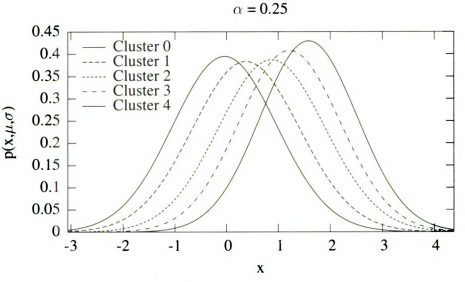


Figure 4.13: Detected clusters for wire phantom with $\mu_1 = 0, \mu_2 = 1, \sigma_1 = \sigma_2 = 0.2$ and $\alpha = 0.25$. Cluster 0: ($\mu = -0.0335, \sigma = 1.01$), Cluster 1: ($\mu = 0.378, \sigma = 1.04$), Cluster 2: ($\mu = 0.877, \sigma = 1.03$), Cluster 3: ($\mu = 1.22, \sigma = 0.978$), Cluster 4: ($\mu = 1.59, \sigma = 0.929$).

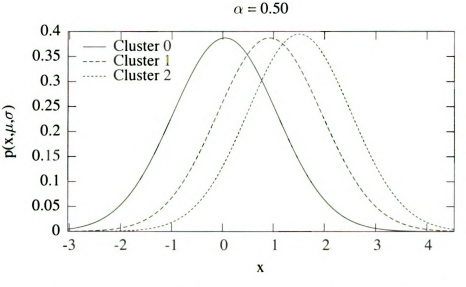


Figure 4.14: Detected clusters for wire phantom with $\mu_1 = 0, \mu_2 = 1, \sigma_1 = \sigma_2 = 0.2$ and $\alpha = 0.5$. Cluster 0: ($\mu = 0.0498, \sigma = 1.03$), Cluster 1: ($\mu = 0.926, \sigma = 1.03$), Cluster 2: ($\mu = 1.51, \sigma = 1.01$).

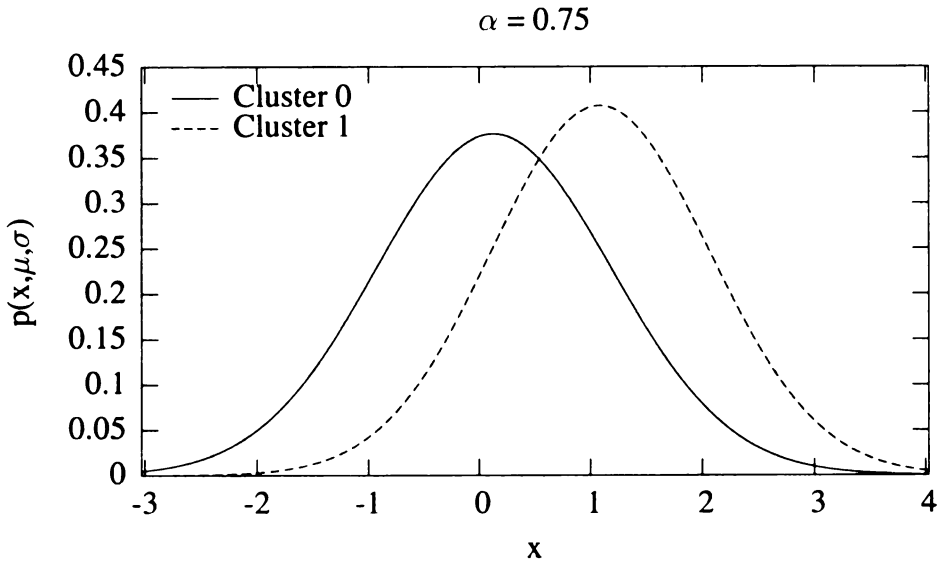


Figure 4.15: Detected clusters for wire phantom with $\mu_1 = 0, \mu_2 = 1, \sigma_1 = \sigma_2 = 0.2$ and $\alpha = 0.75$. Cluster 0: $(\mu = 0.132, \sigma = 1.06)$, Cluster 1: $(\mu = 1.09, \sigma = 0.980)$.

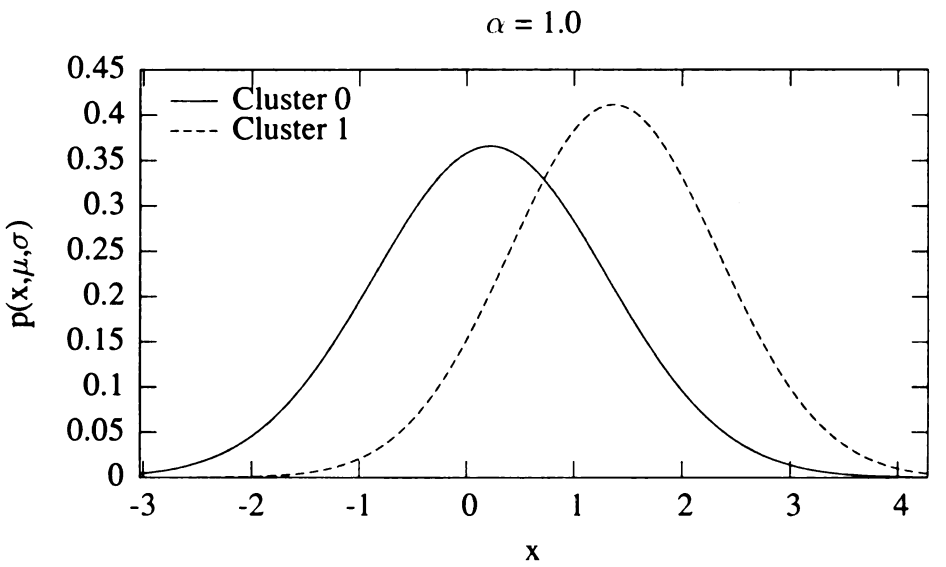


Figure 4.16: Detected clusters for wire phantom with $\mu_1 = 0, \mu_2 = 1, \sigma_1 = \sigma_2 = 0.2$ and $\alpha = 1.0$. Cluster 0: $(\mu = 0.224, \sigma = 1.09)$, Cluster 1: $(\mu = 1.37, \sigma = 0.969)$.

Table 4.9: Confusion matrix for wire5 phantom using $\sigma_1 = \sigma_2 = 0.2$ and $\alpha = 0.25$.

	0	1	2	3	4
0	170887	0	12219	4255	17433
1	28014	0	6381	2670	20285

Table 4.10: Confusion matrix for wire5 phantom using $\sigma_1 = \sigma_2 = 0.2$ and $\alpha = 0.50$.

	0	1	2
0	181207	11037	12550
1	33293	7467	16590

in Tables 4.13 – 4.16 are the classification error rates for the wire phantom. For volumes where more clusters were detected than were actually present in the ground truth, a detected cluster is identified as corresponding to the true class that had a majority of the detected voxels. For example, in Table 4.9 detected clusters $(0, 2, 3)$ are assigned to true class 0, while detected cluster (4) is assigned to to true class 1.

Examining the misclassification rates for the wire phantom indicates average misclassification rates up to approximately 22%. The highest error rates occurred when $\sigma_1 \geq 0.9$. The error rate is attributable to several factors. The first factor is in the test cases where there is significant overlap between the two classes. When $\sigma_1 = 1.0$, its standard deviation is the same as the distance to the second class. The error rate also tends to increase as σ_2 approaches 1.0. This leads to the second factor: the classifier is biased towards classes with lower means, and classes with more members.

On synthetic data, the segmentation algorithm performs well, provided the data are reasonably separated. In the case of the synthetic data, the two classes fell into 81.9% background, and 18.1% foreground. In a data set where the classes are more evenly distributed, the error rate should decrease.

Table 4.11: Confusion matrix for wire5 phantom using $\sigma_1 = \sigma_2 = 0.2$ and $\alpha = 0.75$.

	0	1
0	189949	14845
1	38945	18405

Table 4.12: Confusion matrix for wire5 phantom using $\sigma_1 = \sigma_2 = 0.2$ and $\alpha = 1.0$.

	0	1
0	195770	9024
1	43812	13538

Table 4.13: Misclassification rates for $\alpha = 0.25$ over 20 trials. The percentage of the voxels that were misclassified is given.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.0	0.1	0.4	1.1	2.1	3.1	3.9	3.4	3.3	3.4
0.2	0.1	0.5	1.6	3.0	4.4	5.3	5.8	6.1	6.5	7.0
0.3	1.0	2.4	3.9	5.4	6.4	7.3	7.9	8.5	8.8	8.8
0.4	2.9	5.3	7.4	8.9	9.4	10.3	10.9	10.8	10.8	11.0
0.5	5.6	8.6	10.6	11.7	12.2	13.2	13.8	13.6	12.9	12.8
0.6	6.8	10.7	13.0	14.1	14.7	15.2	15.3	15.3	15.0	14.4
0.7	7.7	12.5	15.4	16.7	17.1	17.1	16.9	16.7	16.4	16.1
0.8	9.4	14.5	17.6	19.1	19.3	19.0	18.6	18.2	17.9	17.5
0.9	10.6	16.1	19.4	20.8	21.2	20.9	20.4	19.9	19.4	19.0
1.0	11.1	15.9	19.9	21.6	21.8	21.9	21.9	21.5	21.0	20.5

Table 4.14: Misclassification rates for $\alpha = 0.50$ over 20 trials. The percentage of the voxels that were misclassified is given.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.0	0.1	0.3	1.1	2.1	3.1	4.0	4.8	5.6	6.3
0.2	0.1	0.8	2.1	3.0	4.0	4.8	5.5	6.0	6.5	7.1
0.3	1.1	2.1	3.8	5.3	6.8	7.5	8.0	8.3	8.5	8.7
0.4	3.2	4.9	6.9	8.2	9.6	10.4	10.7	10.8	10.8	10.8
0.5	5.5	8.1	10.1	11.2	12.1	13.1	13.3	13.2	13.1	12.9
0.6	6.8	10.5	12.8	14.1	14.6	14.9	15.2	15.3	15.2	14.9
0.7	13.6	12.6	15.2	16.6	16.9	16.9	16.8	16.6	16.4	16.3
0.8	18.0	14.3	16.9	18.6	19.1	19.0	18.6	18.2	17.8	17.5
0.9	19.8	17.2	18.2	20.2	21.1	21.0	20.5	19.9	19.4	18.9
1.0	20.1	19.5	19.3	21.5	21.9	21.9	21.9	21.7	21.1	20.5

Table 4.15: Misclassification rates for $\alpha = 0.75$ over 20 trials. The percentage of the voxels that were misclassified is given.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.0	0.0	0.6	1.8	3.2	4.6	5.9	7.1	8.5	9.9
0.2	0.8	0.8	1.5	2.8	4.1	5.4	6.5	7.7	8.9	10.2
0.3	3.3	3.3	4.2	5.3	6.4	7.3	8.0	8.8	9.7	10.7
0.4	6.5	6.5	7.4	8.5	9.3	9.9	10.2	10.6	11.0	11.6
0.5	9.8	9.8	10.8	11.8	12.4	12.6	12.7	12.7	12.7	12.9
0.6	12.9	13.0	14.0	14.9	15.3	15.3	15.1	14.9	14.6	14.4
0.7	15.3	15.6	16.8	17.7	17.9	17.8	17.4	17.0	16.4	16.0
0.8	16.0	16.9	18.6	20.0	20.3	20.1	19.5	18.9	18.2	17.6
0.9	15.3	16.8	19.3	21.3	21.9	21.8	21.5	20.7	19.9	19.1
1.0	13.8	16.1	19.3	21.8	21.9	21.9	21.9	21.9	21.3	20.5

Table 4.16: Misclassification rates for $\alpha = 1.0$ over 20 trials. The percentage of the voxels that were misclassified is given.

σ_1	σ_2									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.0	0.1	1.0	2.6	4.4	6.0	7.8	9.6	11.2	12.3
0.2	0.4	0.5	1.7	3.3	5.0	6.5	8.1	9.9	11.4	12.5
0.3	2.3	2.6	3.8	5.2	6.5	7.7	8.9	10.4	11.8	12.8
0.4	5.3	5.6	6.9	8.2	9.1	9.8	10.4	11.3	12.3	13.2
0.5	8.6	9.0	10.3	11.4	12.0	12.3	12.5	12.8	13.3	13.9
0.6	11.1	11.9	13.4	14.5	14.9	14.9	14.7	14.6	14.7	14.9
0.7	11.4	13.5	15.8	17.2	17.5	17.3	16.9	16.5	16.3	16.1
0.8	9.6	13.8	17.1	18.9	19.5	19.2	18.7	18.2	17.8	17.5
0.9	21.9	21.6	19.6	20.0	20.8	20.9	20.4	19.8	19.3	18.8
1.0	21.9	21.9	21.7	21.8	21.9	21.9	21.9	21.3	20.6	20.1

4.3.2 Real Data

Synthetic data is useful for validating the algorithm's performance, but it is necessary to evaluate the performance on real data. To this end, 10 data sets have been prepared, as described in Table 4.17. Since the ground truth for the real data sets is not known, evaluation of the segmentation is done in two steps. The first step is to apply the clustering algorithm to each data set for four α values, and compare the summed distributions to the real histograms of density/absorption for each data set. The second step will be to apply the Bayes classifier to the data and examine the clustered and segmented volumes.

Table 4.17: Real data sets used for testing the segmentation algorithm.

Data Set	Rows	Columns	Slices	Description
5atah1_a	658	658	311	Podzol soil aggregate.
5atah2_a	658	658	290	Podzol soil aggregate.
5atah3_a	658	658	250	Podzol soil aggregate.
5ath1_a	658	658	386	Podzol soil aggregate.
5ath2_a	658	658	403	Podzol soil aggregate.
5ath4_a	658	658	258	Soil aggregate.
5btah2_a	658	658	293	Soil aggregate.
5btah4_a	658	658	316	Soil aggregate.
Scutigera	512	512	315	Small crustacean. Data set has some reconstruction errors that cause ghosting of the scutigera in the air region of the volume.
impregnated_bullion_a	341	341	381	Peat in acrylic resin, machined into a cylinder.

Prior to running the tests, each sample volume was first cropped to remove most of the air surrounding the objects in the volume. The cropping step was done to reduce the size of the data and the compute time. After each volume was cropped, it was median filtered using a kernel of width 3 to eliminate some of the noise in the volume. The primary effect was to remove outliers from the data, while still preserving the majority of the structure in the volume.

The first round of tests were run using $\alpha \in [0.25, 0.5, 0.75, 1.0]$. The number of detected clusters and processing time as a function of the α parameter are shown in Tables 4.18–4.27.

Table 4.18: Data set 5atah1_a α parameter performance. Dimensions = ($658^2 \times 311$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	58	325	2.414e-06	5.603e+00
0.5	23	166.77	1.239e-06	7.251e+00
0.8	17	109.35	8.121e-07	6.432e+00
1.0	13	84.46	6.272e-07	6.497e+00

Table 4.19: Data set 5atah2_a α parameter performance. Dimensions = ($658^2 \times 290$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	75	364.51	2.903e-06	4.860e+00
0.5	35	174.26	1.388e-06	4.979e+00
0.8	23	120.36	9.586e-07	5.233e+00
1.0	15	88.38	7.039e-07	5.892e+00

Table 4.20: Data set 5atah3_a α parameter performance. Dimensions = ($658^2 \times 250$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	59	246.13	2.274e-06	4.172e+00
0.5	27	130.85	1.209e-06	4.846e+00
0.8	17	82.97	7.665e-07	4.881e+00
1.0	11	69.18	6.391e-07	6.289e+00

.

Figure 4.17 shows cluster labeled slices from the Scutigera data set. Each sub-image corresponds to a different α parameter, showing how the different clusters merge as the α parameter increases.

The number of clusters and execution time decreased as α increased in the test cases. The decrease in the number of clusters is due to the relaxing of the similarity and the time decrease is due to the reduced number of clusters being searched in each iteration of the clustering algorithm.

Table 4.21: Data set 5ath1_a α parameter performance. Dimensions = ($658^2 \times 386$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	57	436.05	2.609e-06	7.650e+00
0.5	25	218.95	1.310e-06	8.758e+00
0.8	17	142.15	8.506e-07	8.362e+00
1.0	12	112.25	6.717e-07	9.354e+00

Table 4.22: Data set 5ath2_a α parameter performance. Dimensions = ($658^2 \times 403$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	55	406.67	2.331e-06	7.394e+00
0.5	23	204.77	1.174e-06	8.903e+00
0.8	15	135.15	7.746e-07	9.010e+00
1.0	10	112.17	6.429e-07	1.122e+01

Table 4.23: Data set 5ath4_a α parameter performance. Dimensions = ($658^2 \times 258$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	46	255.88	2.291e-06	5.563e+00
0.5	22	128.47	1.150e-06	5.840e+00
0.8	14	82.9	7.421e-07	5.921e+00
1.0	8	66.03	5.911e-07	8.254e+00

Table 4.24: Data set 5btah2_a α parameter performance. Dimensions = ($658^2 \times 293$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	83	445.71	3.513e-06	5.370e+00
0.5	35	209.31	1.650e-06	5.980e+00
0.8	22	133.2	1.050e-06	6.055e+00
1.0	16	95.77	7.549e-07	5.986e+00

Table 4.25: Data set 5btah4_a α parameter performance. Dimensions = ($658^2 \times 315$)

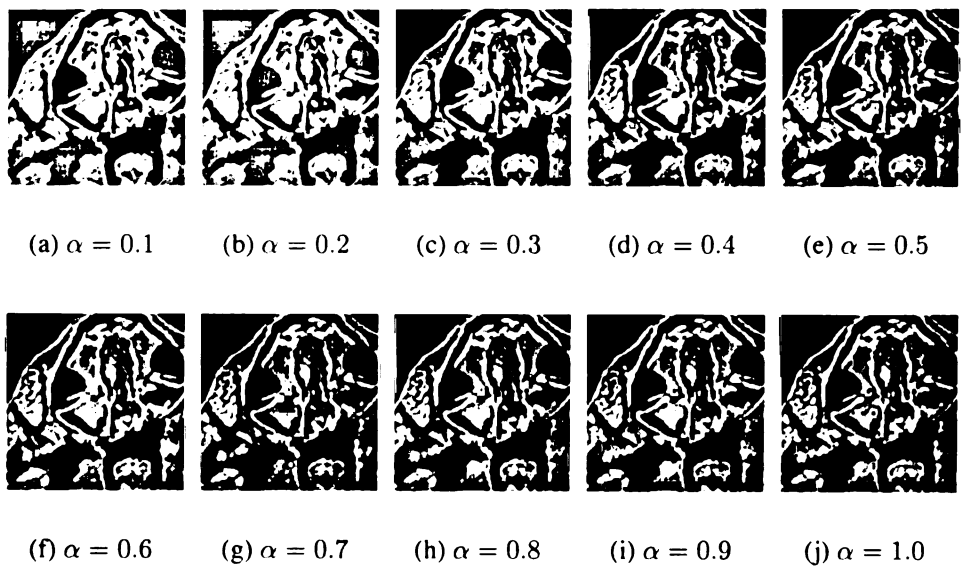
α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	77	423.14	3.103e-06	5.495e+00
0.5	35	211.8	1.553e-06	6.051e+00
0.8	22	133.58	9.794e-07	6.072e+00
1.0	16	98.67	7.235e-07	6.167e+00

Table 4.26: Data set Scutigera α parameter performance. Dimensions = ($512^2 \times 316$)

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	40	170.46	2.058e-06	4.261e+00
0.5	17	87.76	1.059e-06	5.162e+00
0.8	8	60.24	7.272e-07	7.530e+00
1.0	8	49.78	6.009e-07	6.223e+00

Table 4.27: Data set impregnated_bullion_a α parameter performance. Dimensions = $(341^2 \times 381)$

α	Clusters	Total Seconds	Seconds/Voxel	Seconds/Cluster
0.2	108	257.94	5.822e-06	2.388e+00
0.5	49	117.49	2.652e-06	2.398e+00
0.8	32	69.46	1.568e-06	2.171e+00
1.0	17	50.1	1.131e-06	2.947e+00



(a) $\alpha = 0.1$ (b) $\alpha = 0.2$ (c) $\alpha = 0.3$ (d) $\alpha = 0.4$ (e) $\alpha = 0.5$

(f) $\alpha = 0.6$ (g) $\alpha = 0.7$ (h) $\alpha = 0.8$ (i) $\alpha = 0.9$ (j) $\alpha = 1.0$

Figure 4.17: Scutigera showing different cluster labels for varying values of α. The number of clusters is $[51, 25, 17, 11, 10, 9, 8, 6, 6, 6]$.

The Bhattacharrya distance [26] is used to compare the distance between the actual histogram, and the weighted sum of the clusters. In order to perform the comparisons, the histograms and the weighted sums were all scaled such that they summed to 1. A discrete form of the Bhattacharrya distance, shown in Equation 4.4, is used to compute the distances [97]. The scaled histogram and the scaled weighted sum of Gaussians are K_1 and K_2 respectively.

$$d(K_1, K_2) = \left(1 - \sum_{n=1}^{N} \sqrt{K_1(n;T_1) * K_2(n;T_2)}\right)^{\frac{1}{2}} \tag{4.4}$$

The distances are shown in Table 4.28, with smaller values indicating a closer match between the histogram and the weighted sum. With the exception of the impregnated_bullion_a sample, the distance between the histogram and the weighted sum of Gaussians decreases as the α parameter decreases. As the α parameter decreases, more clusters are generated allowing for a better fit. In the case of the impregnated_bullion_a sample, the clustering algorithm over-fit the histogram. While the maximum error approaches 40% for $\alpha = 1.0$, the maximum error for $\alpha = 0.25$ is 13.22%. The two main sources of error are not accurately fitting the main peak in the histogram, and that the actual absorption values are not drawn from a set of Gaussian distributions.

To qualitatively evaluate the detected clusters requires generating a sum of Gaussian distributions from the cluster table, where each Gaussian contribution is weighted by the number of voxels it covers. The scale on the vertical axis for the histogram and the sum of Gaussians differs, but the shapes of the distributions should be similar. The histograms for each of the test volumes, along with their associated weighted sum of Gaussians, are shown in Figures 4.18–4.27.

Examination of the sum of Gaussian distributions and the histograms indicates the clustering algorithm detects a set of Gaussians that closely matches the shape of the underlying histogram in the real data. Close examination reveals some of the fine features of the his-

Table 4.28: Bhattacharyya distance between histogram and sum of weighted Gaussians from clustering algorithm.

Data Set	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1.0$
5atah1_a	0.0609	0.1409	0.2504	0.2691
5atah2_a	0.0392	0.1198	0.2291	0.2773
5atah3_a	0.0436	0.1016	0.2133	0.2332
5ath1_a	0.0616	0.1438	0.2462	0.2837
5ath2_a	0.1275	0.2220	0.3377	0.3573
5ath4_a	0.1045	0.2376	0.3186	0.3328
5btah2_a	0.0361	0.0457	0.1598	0.1809
5btah4_a	0.0471	0.0928	0.1876	0.2071
Scutigera	0.1322	0.3090	0.3296	0.3651
impregnated_bullion_a	0.0434	0.0154	0.0154	0.0286

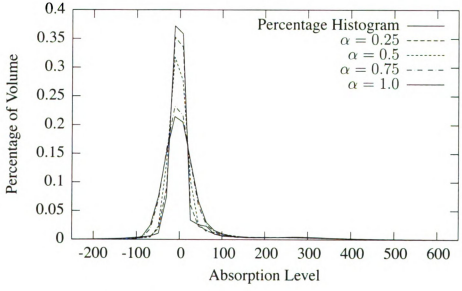


Figure 4.18: Comparison of 5atah1_a histogram with sum of Gaussians from clustering algorithm

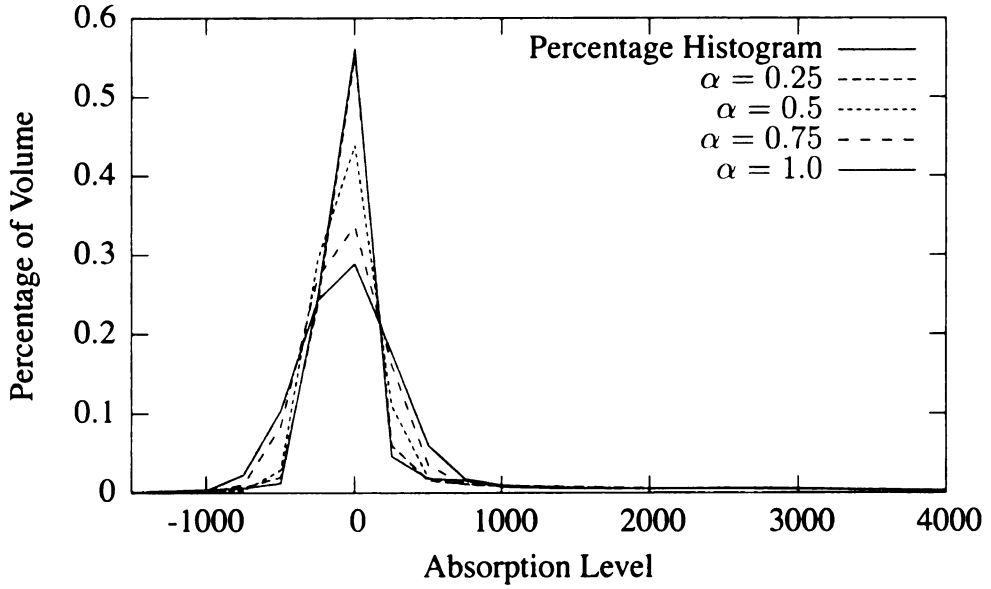


Figure 4.19: Comparison of 5atah2_a histogram with sum of Gaussians from clustering algorithm.

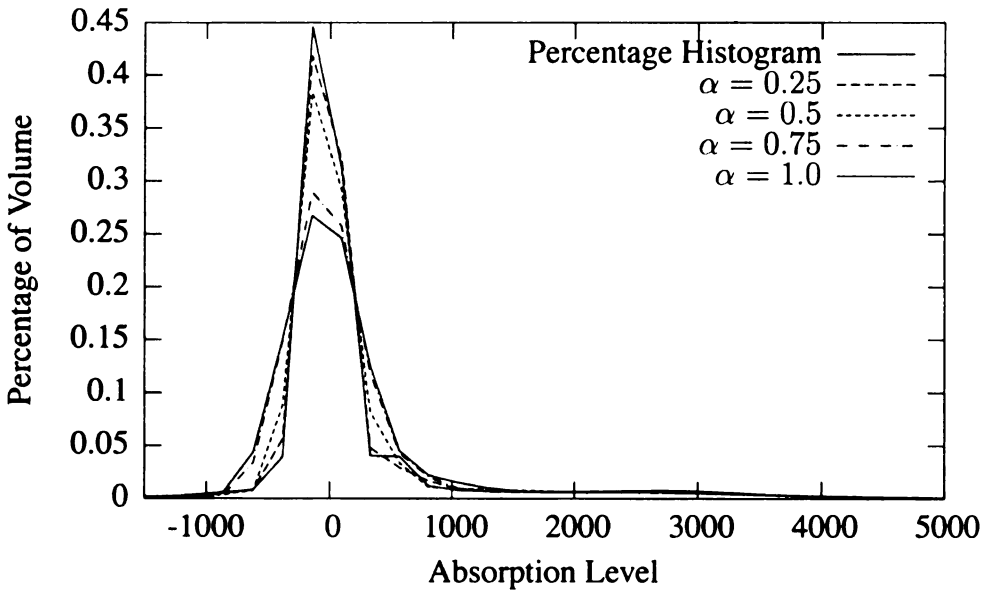


Figure 4.20: Actual histogram of 5atah3_a data versus the detected distributions.

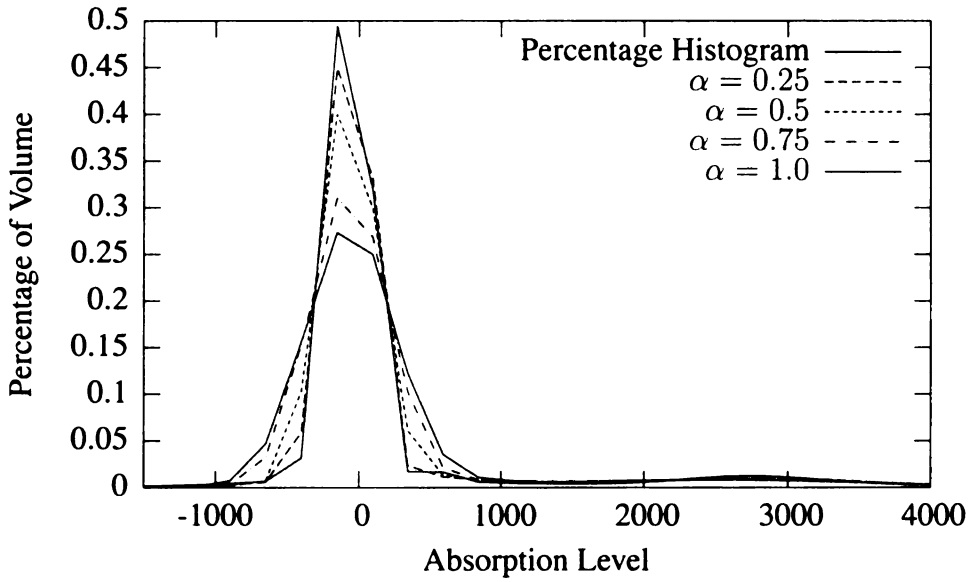


Figure 4.21: Comparison of 5ath1_a histogram with sum of Gaussians from clustering algorithm.

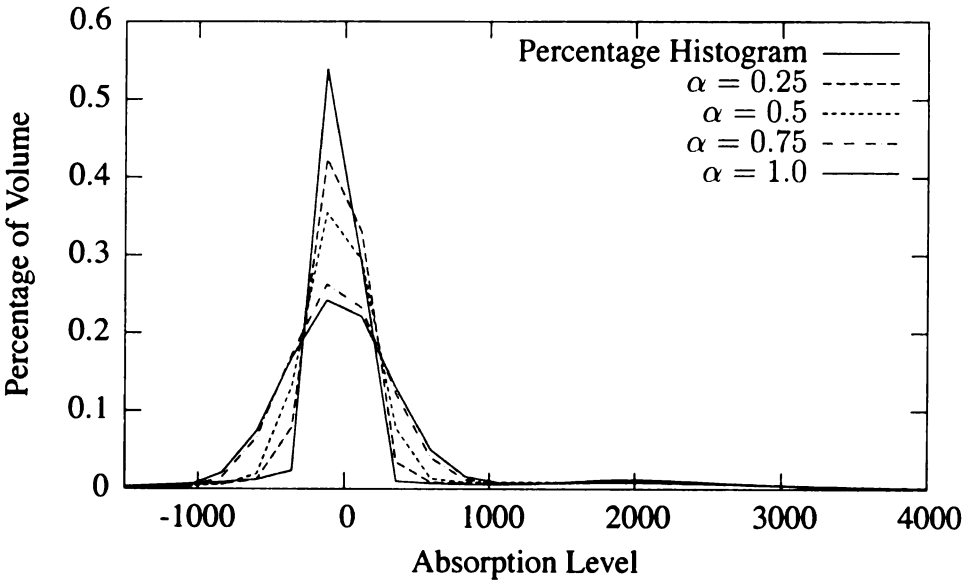


Figure 4.22: Comparison of 5ath2_a histogram with sum of Gaussians from clustering algorithm.

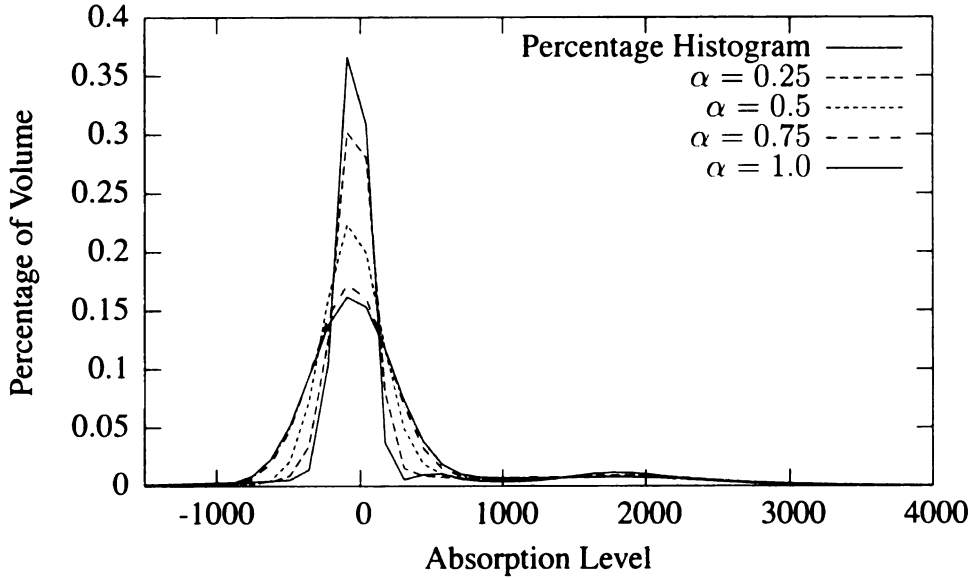


Figure 4.23: Actual histogram of 5ath4_a data versus the detected distributions.

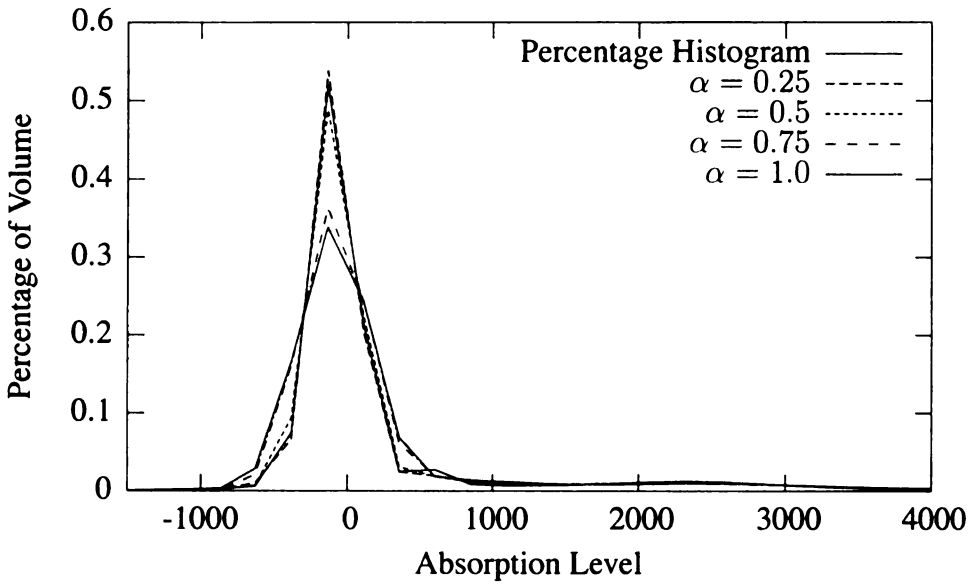


Figure 4.24: Comparison of 5btah2_a histogram with sum of Gaussians from clustering algorithm.

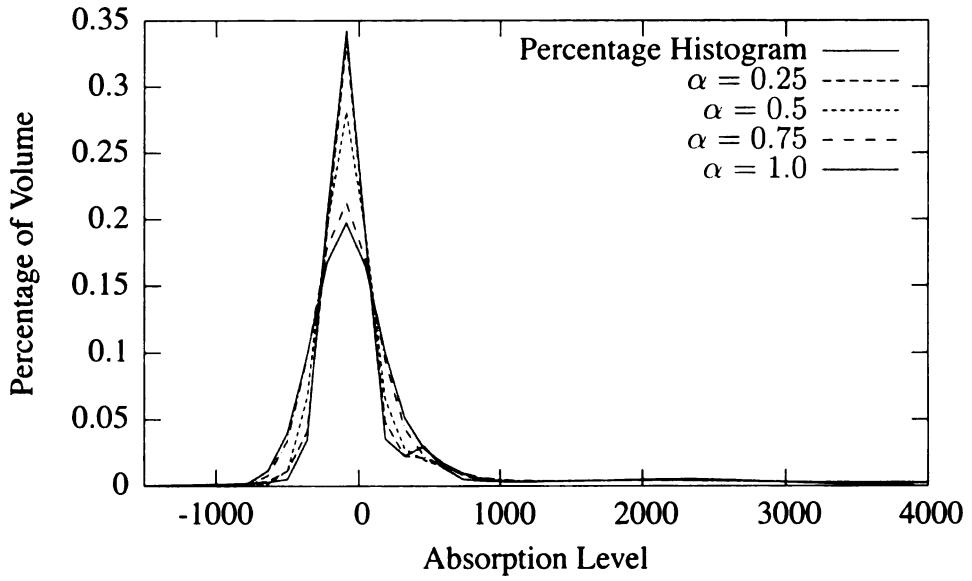


Figure 4.25: Comparison of 5btah4_a histogram with sum of Gaussians from clustering algorithm.

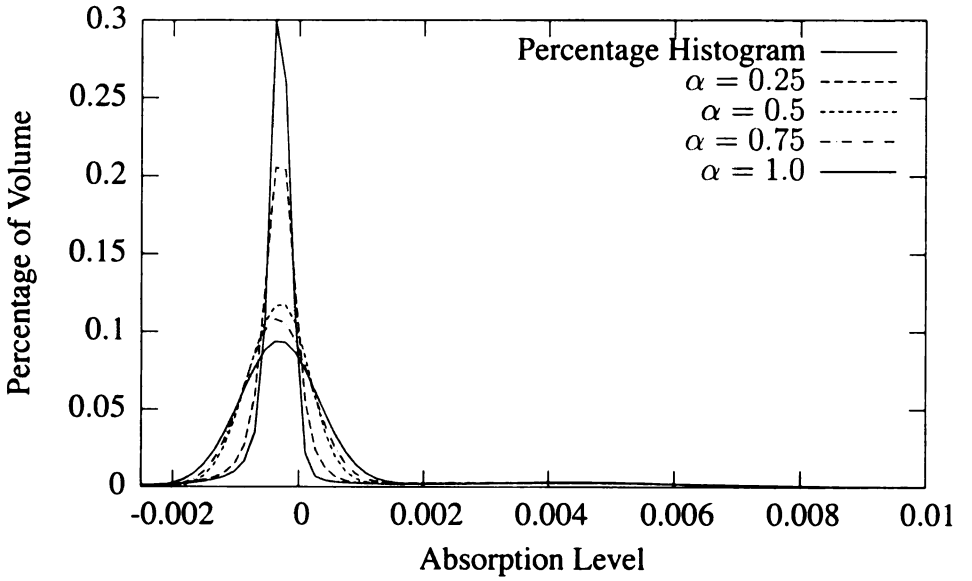


Figure 4.26: Comparison of Scutigera histogram with sum of Gaussians from clustering algorithm.

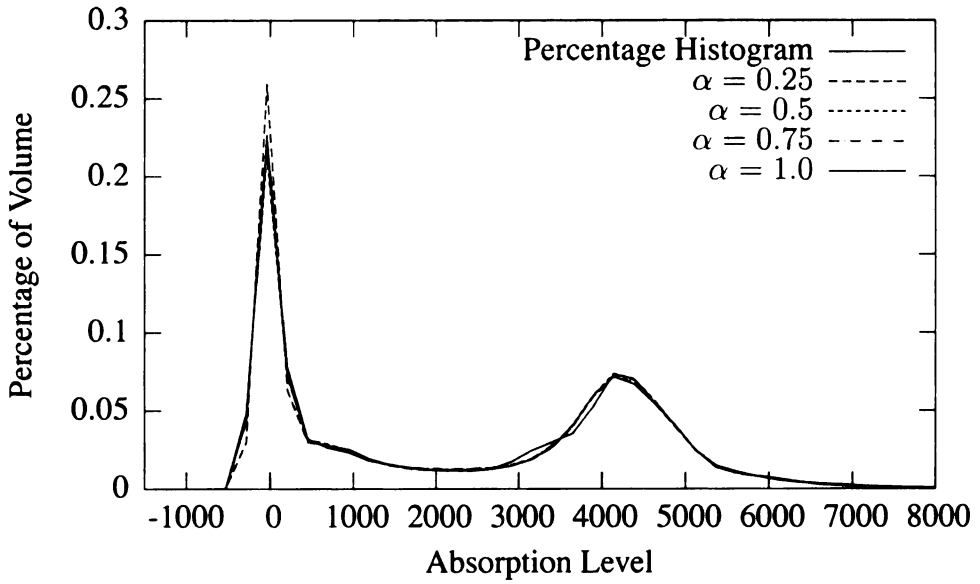


Figure 4.27: Comparison of impregnated_bullion_a histogram with sum of Gaussians from clustering algorithm.

togram are lost as α increases, but the gross distribution shape is still preserved.

Examination of the histograms indicates a set of Gaussian distributions can be found to approximate the underling distribution of the data. Using this result allows for re-coding of the data using the cluster labels. The most immediate advantage is simple thresholding can be used to perform binarization and n-ary segmentation. While the absolute value of a classified voxel is lost, the cluster-labeled voxels retain their relative order. Two side-effects of labeling the voxels with the cluster labels are intensity range normalization, and contrast enhancement. These side-effects may simplify some processing operations, but the user must be aware that smooth gradients will be eliminated.

Volume 5atah1_a, shown in Figure 4.28, has a piece of soil material hanging from the side. In Sub-Figures 4.28(a)-4.28(c) a cross section of the protruding material is visible as a small region to the left of the image. In Sub-Figure 4.28(d) the protruding material is visible, again to the left of the image. The segmentation algorithm was able to preserve the structure that connects the main body of the aggregate to the small piece hanging to the side. The disk shaped structure at the bottom of the image is the head of the acrylic pin on which the sample was mounted. In Figure 4.29(a) an elliptical area can be seen surrounding

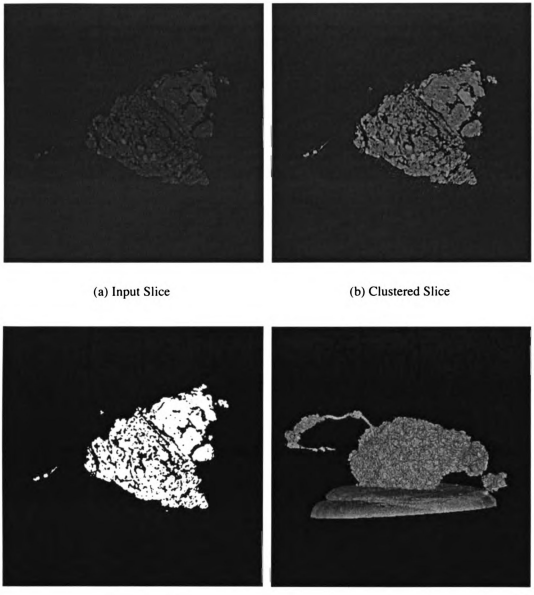
a brighter region on the right side of the aggregate. In the corresponding clustered slice, Figure 4.29(b), the same feature is present indicating the clustering algorithm does preserve structured regions. Figure 4.30(a) shows a cross section of a scutigera crustacean. By using the clustered volume a three dimensional rendering of the crustacean can be generated similar to Figure 4.30(d). Figure 4.31(a) shows a cross section of an acrylic impregnated peat sample. The clustering algorithm is able to separate the acrylic resin from the peat. During the rendering process, the acrylic resin was removed from the volume revealing the structure of the soil, shown in Figure 4.31(d). For comparison, the original sample is shown in Figure 4.32. Additional segmented volumes may be found in Appendix A.

Suitability of Algorithm

Four criteria the segmentation algorithm must meet were identified at the beginning of this chapter.

1. Simplicity of computation.
2. Robust in the presence of noise.
3. Allows for n-ary segmentation, in addition to binary segmentation.
4. Generates intermediate state information allowing a scientist to drive the segmentation towards a desired goal.

The computation performed by the clustering algorithm is simple. Each voxel in the volume is visited once during the clustering phase, once during the labeling phase and once during each classification cycle. The clustering algorithm processes a 483.9 megabyte volume in 445.71 seconds. For each n^3 neighborhood, the algorithm performs $\Theta(n^3)$ additions, $\Theta(n^3)$ multiplications, 1 subtraction, and 1 square root, giving the neighborhood computation a computational complexity of $\Theta(n^3)$. Merging two clusters is a constant time operation. The order of the elements in the cluster table is maintained using an $m * \log(m)$



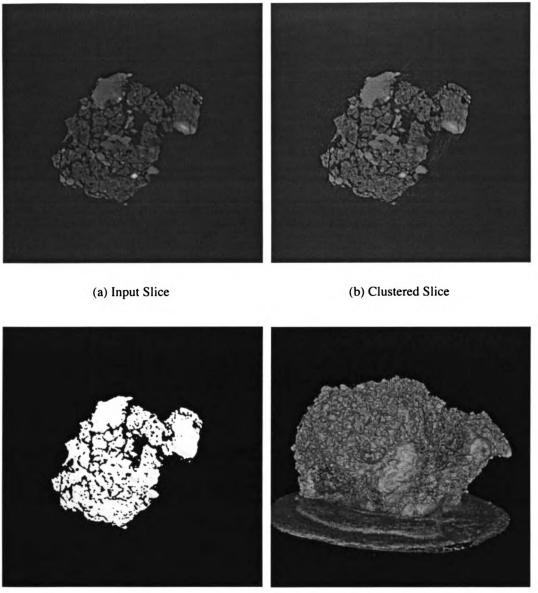
(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure 4.28: 5atah1_a segmentation results



(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure 4.29: 5atah2_a segmentation results

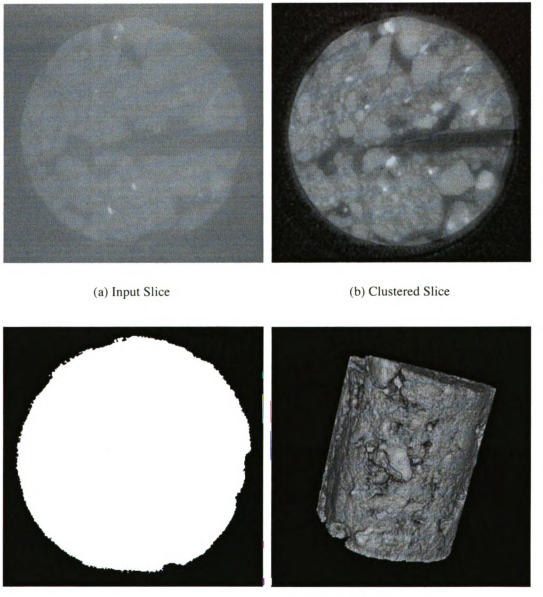
(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure 4.30: Scutigera segmentation results



(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure 4.31: impregnated_bullion_a segmentation results

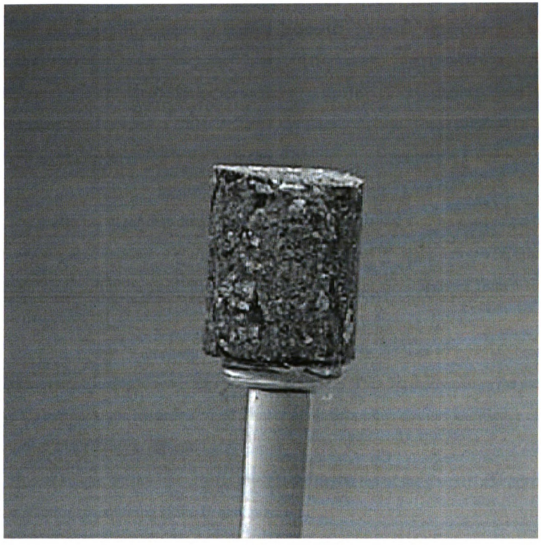


Figure 4.32: Photograph of impregnated_bullion_a original sample.

sorting algorithm, where m is number of clusters in the table. In the worst case merging clusters is an $O(m^2)$ operation if a merge operation causes all clusters to merge into one. Since the cluster table order is maintained, merging clusters is on average an $O(m)$ operation. The most expensive phase of the clustering algorithm is typically maintaining the table order, but since the table size tends to be small on average the cost is not significant.

Since the algorithm operates by building large non-contiguous regions for its cluster statistics, it is robust in the presence of noise until there is almost no interclass separation. Furthermore, the classification stage uses a Bayesian decision process causing clusters with little support to have a low contribution to the classification phase. The α parameter allows the user to control how closely the clustering algorithm matches the data. For noisier data sets, α can be increased to generate larger clusters whose statistics are more stable.

The clustering algorithm discovers a number of classes that can be used to perform either binary segmentation or n-ary segmentation. The user of the algorithm can experiment with different partitionings of the clusters to drive different classifications from a single clustering cycle.

The cluster table is amenable to later merging to allow the user to generate broader clusters without having to rerun the clustering algorithm. This flexibility allows the user to drive the segmentation towards a target number of clusters when *a priori* knowledge of the number of true classes is present.

This technique satisfies the criteria listed at the beginning of the chapter, and provides a fast solution to exploring arbitrary data sets.

4.4 Conclusion

The clustering algorithm proposed in this chapter satisfies the four criteria established at the beginning of the chapter. In addition to satisfying the basic criteria, the algorithm is amenable to the application of other similarity criteria. Finally, the algorithm is domain

independent in the sense that the only *a priori* knowledge required for segmentation is that separable structures are present in the volume being operated on. The clustering algorithm can generate a set of weighted Gaussians that closely match the shape of histogram on real data. This was demonstrated on 10 real data sets.

Chapter 5

Interest Detection

Interest detection is the process of finding interesting or potentially interesting points, and presenting them to the user or to algorithms downstream. While exactly what points in a volume are interesting is application-specific, it is possible to provide a framework that allows for a variety of types of interest to be identified. This chapter develops a general method for detecting user-identified interest points using a new interest operator and support vector machines (SVM) [10].

5.1 Existing Interest Detectors

There are structure detectors such as the Harris-Stephens corner edge detector [44], the Kitchen-Rosenfeld gray-level corner detector [58], the Canny edge detector [14], the Hough transform [52] and its derivatives. There are detectors that measure the amount of symmetry in an image, gradient discontinuities, uniformity in a region, energy or variance, or other low-level features [117]. Salience measures [95] attempt to find important points in the image, often in the context of perceptual importance or as it relates to image compression. There are operators that detect points that are also candidates for matching across multiple images based on correspondences between points at the intersection of lines that form approximately right angles [116]. The discrete mass transform and discrete sym-

metry transform act to detect points that are symmetric near an edge [39]. Corner detectors can be applied to detect 3D corners in a volume [84]. Structure tensor methods examine the gradient structure of images [68, 69, 4, 34]. Wavelet coefficients are used to detect features that are present across multiple scales [95].

5.2 Interest Detection Algorithm Criteria

The goal of developing techniques to facilitate interest detection leads to the following set of criteria:

1. Simplicity of computation.

2. Adaptability to different types of data.

3. Generates and preserves intermediate state information that enables a scientist to drive the interest detection towards a desired goal.

4. Amenable to incremental refinement.

5. The framework should be extendible as new interest detection techniques are developed.

These criteria are discussed in the following sections.

5.2.1 Simplicity of Computation

A typical reconstructed volume acquired at the Advanced Photon Source is on the order of 512 megabytes in size. Future data sets may grow to be as large as 4 gigabytes for a full resolution volume. With these data sizes in mind, simplicity of computation encompasses two factors. The first factor is that any algorithm should be computationally inexpensive.

An algorithm that requires more than polynomial effort is unacceptable. The second factor is that any algorithm should minimize the number of iterations through the data. An additional desirable, but not necessary, property of an algorithm is for it to be parallizable.

5.2.2 Adaptability to Different Types of Data

Volumetric data sets can be generated from a wide range of materials, and may have almost no inter-sample similarities. Techniques for detecting interest should be adaptable to disparate data sets, particularly those for which there is a minimum of *a priori* knowledge regarding the structure of the material.

5.2.3 State Preservation

Since the data sets are large, any intermediate data should be storable in such a way that it can be reused later. This reuse is important both for facilitating further data exploration and for amortizing the cost of any expensive processing steps.

5.2.4 Incremental Refinement

Since there may be no *a priori* knowledge about the structure of the data, users may need to explore the data. The techniques used for interest detection should allow users to refine their interest detection by building on previous attempts.

5.2.5 Extendible Framework

As mentioned earlier, there is a wide array of existing interest detectors. It is probable that more interest detectors will be developed in the future. Therefore, the interest detection framework must be flexible enough to accommodate new interest detectors and features as they are developed. This extensibility is essential not only for future developments, but also to allow the user to select appropriate features for the current work.

5.3 Proposed Interest Detection Framework

Rather than attempting to develop a single interest detector that is appropriate across all possible inputs, a flexible framework is proposed that combines the results from a variety of different interest operators or feature detectors. The goal is to combine the outputs of a diverse set of operators in such a way that users of the system can teach the interest detector the appropriate features to consider for the specific class of volumes being examined.

The proposed framework has three major stages. The first stage is the training stage, where the user provides sets of interesting and non-interesting points. In the second stage a classifier for labeling interesting and non-interesting classes is constructed. The third stage is the classification of all points in the volume into an interest class. A high-level diagram of the framework is given in Figure 5.1.

The first stage of the interest detection process is to have the user provide two sets of training examples, i.e. interesting and uninteresting points. Since interest is in general ill-defined, this approach will provide users from different domains the opportunity to specify those points that are relevant to their analysis needs.

The second state of interest detection is training the Support Vector Machine (SVM) to classify the points identified by the user as interesting and uninteresting. After each training and partial classification pass, the user can either accept the training results or can retrain the SVM with additional training points.

The third stage is the classification of the entire volume. The model developed from training the SVM in stage two is applied to all voxels in the volume being classified. The final classification can generate either a binary result or a range of confidence values.

5.4 Support Vector Machines

At the core of the interest detection framework is the classifier used to identify the interesting points in the volume. The classifier selected for this application is the Support Vector

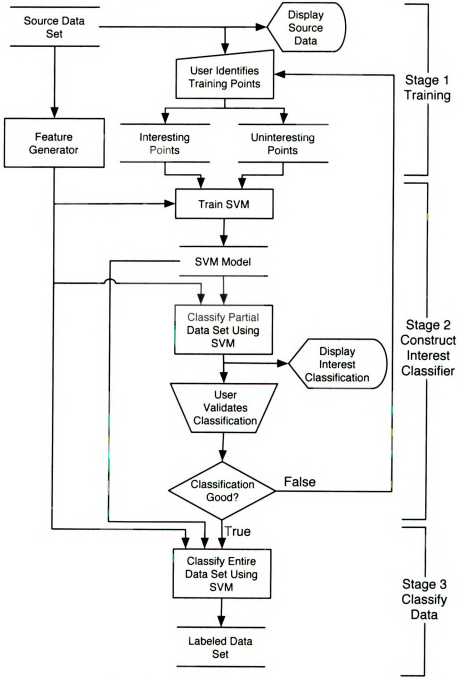


Figure 5.1: Interest detection framework overview.

Machine (SVM) [42]. SVMs have several properties that make them desirable for detecting user-specified features. Among these properties are fast training, non-linear classisification, and robust behavior in the presence of noise.

SVMs provide a robust mechanism for classifying data where the inter-class boundaries are non-linear. The classification problem being addressed in this work is locating areas in a volume that are perceptually similar, as well as identifying multiple types of interest points, e.g. edges and corners. The SVM has been demonstrated in experiments to produce reliable results with relativly small training sets of postive and negative examples [10, 7, 42]. The Torch3 SVM implementation is used in this work [17].

5.5 Proposed Radial Mass Transform Interest Detector

The detectors mentioned in Section 5.1 are based on a variety of different computation techniques; some are and some are not suitable for application to three dimensional data.

The *radial mass transform* (RMT) is designed to provide a basis for extracting rotation- and translation-invariant features from a volume. The radial mass transform integrates the mass (density, intensity, etc.) at distance r from some arbitrary fixed point p_0 in space. The area of integration for a 2D image is shown in Figure 5.2. Let V denote either a 3D volume or a 2D slice or image. The spherical Radon transform (see Tanimoto [103]) integrates all mass in f at radius ρ from p_0 and is shown in Equation 5.1. The continuous radial mass transform integrates mass centered at p_o creating a 1D function of radius r as defined in Equation 5.5 using an abuse of notation given for its intuitive value only. Each point $p_o \in V$ is mapped to a 1D signature (function) that gives the mass in V at distance ρ from p_o. $f(p)$ is the mass (density, intensity, etc.) at point p. In continuous 3D space, we are integrating the mass in a spherical shell, and in a 2D slice, we are integrating the mass over a circumference. The RMT captures a great deal of structure/texture of the neighborhood of a point N_{p_o} and is therefore an efficient representation for segmentation, clustering or

determination of interest points.

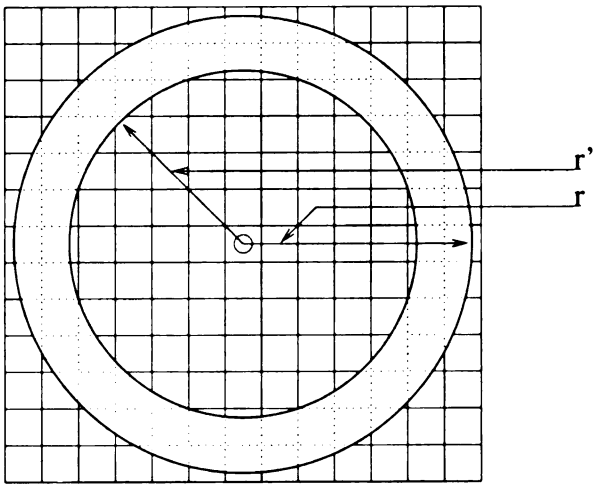


Figure 5.2: Cross section of spherical shell describing region of integration for RMT.

$$H(p_0, \rho) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(p)\, \delta(\rho, p, p_0)\, dx\, dy\, dz \tag{5.1}$$

$$\delta(\rho, p, p_0) = \begin{cases} \infty & \text{if } \rho = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

$$p_0 = (x_0, y_0, z_0) \tag{5.3}$$

$$p = (x, y, z) \tag{5.4}$$

$$m(p_0, \rho) \approx \int_{p \in V} [||p - p_o|| = \rho]\, f(p) dV \tag{5.5}$$

The formulation of the RMT in Equation 5.5 will result in increasing values as ρ increases. As a consequence of this property, the RMT as initially formulated is inappropriate for later use in classification since the values for $m(p_0, \rho)$ when ρ is large will have greater variance for homogeneous regions than the absolute values of $m(p_0, \rho)$ when ρ is small. To rectify this problem, the RMT is normalized by the surface area of the region involved

in the integral. The reformulated RMT with the normalizing factor is shown in Equation 5.6. Unless otherwise noted, all further discussion of the RMT will refer to this normalized version.

$$m\left(p_0, \rho\right) \approx \frac{\int_{p \in V} \left[||p - p_o|| = \rho\right] f(p) dV}{4\pi\rho^2} \tag{5.6}$$

5.6 Discrete Radial Mass Transform Computation

The discrete radial mass transform (RMT) operates by integrating the mass surrounding a discrete point at a discrete set of radii: the computation is outlined in Algorithm 2. The benefits of this transform are two-fold. The first benefit is that we are able to characterize the structure of a region of $(2r+1)^3$ voxels centered at a point with a vector of length $r+1$. The second benefit is that this characterization is, in the continuous space, rotationally and translationally invariant. The two benefits should allow for more robust analysis of data and provide a degree of data dimension reduction.

The data dimension reduction comes at a cost of significant computational overhead. A naive implementation will have each voxel in the volume contributing to approximately $\frac{4}{3}\pi r^3$ mass accumulation computations. In other words, to compute the RMT for a single voxel requires a computational effort proportional to the volume of a sphere of radius r. This effort will be required at each voxel at which the RMT is desired. The overall running time of this transform, assuming a stride of 1 during computation, is $\mathrm{O}(r^3 \times rows \times cols \times slices)$.

Algorithm 2: Discrete radial mass transform.
Input: Volume V, Origin for transform p_0, Maximum radius $r_{\max}$
Output: Discrete radial mass transform RMT_{p_0}
RMT($V, p_o, r_{\max}$)
(1) **for** $r = 1$ **to** $r_{\max}$
(2) $\quad \mathrm{RMT}_{p_0,r} = 0$
(3) $\quad$ **foreach** offset $\in RMTlookup_r$
(4) $\quad\quad \mathrm{RMT}_{p_0,r} = \mathrm{RMT}_{p_0,r} + V_{p_0+\mathrm{offset}}$
(5) **return** RMT_{p_0}

The computational requirements are such that this transform is best suited to a parallel implementation. Additionally, it is beneficial to compute the transform only in regions where structure is likely to be present. After computation, the RMT can be viewed as a vector field that encodes neighborhood information for each voxel.

5.6.1 Theoretical RMT Error

One of the sources of error in the discrete RMT is the quantization of the spherical shells that are used to integrate the mass. In Table 5.1 the volume of the discrete RMT shells is compared to the volume of continuous RMT shells. The volume of a continuous RMT shell of width 1 is:

$$\mathrm{RMTVolume}_{\mathrm{cont}}(r) = \frac{4\pi\left((r+0.5)^{3.} + (r-0.5)^{3.}\right)}{3} \tag{5.7}$$

The volume of the discrete RMT shell $\mathrm{RMTVolume}_{\mathrm{disc}}(r)$ is derived from the size of the $RMTlookup_r$ table used for computing the transform. The relative error for radius r is defined as:

$$\mathrm{VolError}(r) = \frac{\mathrm{RMTVolume}_{\mathrm{disc}}(r) - \mathrm{RMTVolume}_{\mathrm{cont}}(r)}{\mathrm{RMTVolume}_{\mathrm{cont}}(r)} \tag{5.8}$$

The largest relative error is at radius 1, with an overall average relative error of 0.103. This error should serve as an upper bound on the error for the discrete RMT versus the

Table 5.1: Volumes of continuous and discrete RMT shells

Radius	Continuous	Discrete	Discrete Error
1	13.6136	18	0.322
2	51.3127	62	0.209
3	114.145	98	-0.141
4	202.109	210	0.039
5	315.206	350	0.110
6	453.437	450	-0.008
7	616.799	602	-0.024
8	805.295	762	-0.054
9	1018.92	1142	0.121
10	1257.68	1250	-0.006

continuous RMT. The source of error in the discrete RMT is that voxels cover more than one radii, but in the discrete computation contribute only to one mass integration. When the discrete RMT is computed in uniform areas, there is no error. Errors arise when the RMT includes non-uniform regions, since partially covered voxels contribute more or less than they should to the integral. While the discretization process introduces some error, it does not appear to be a serious problem. The effects of the discretization errors can been seen in rotational invariance tests in Section 5.7.1.

The partial occupancy error may be resolvable by introducing a contribution factor for each voxel at a given radius, shown in Algorithm 3. The addition of this factor would, as a minimum, double the time required to compute the transform.

Algorithm 3: Discrete radial mass transform with partial occupancy.
Input: Volume V, Origin for transform p_0, Maximum radius $r_{\max}$
Output: Discrete radial mass transform RMT_{p_0}
RMT PARTIAL($V, p_o, r_{\max}$)
(1) **for** $r = 1$ **to** $r_{\max}$
(2) $\quad \mathrm{RMT}_{p_0,r} = 0$
(3) $\quad$ **foreach** $\{\mathrm{offset}, \mathrm{scale}\} \in RMTlookup_r$
(4) $\quad\quad \mathrm{RMT}_{p_0,r} = \mathrm{RMT}_{p_0,r} + (V_{p_0+\mathrm{offset}} \times \mathrm{scale})$
(5) **return** RMT_{p_0}

5.6.2 Discrete RMT Features

Features that characterize the underlying data can be computed from the RMT itself. These features, and others, are implicitly used when the RMT is used as a feature for the SVM classification. The features that have been generated for explicit use are jaggedness, jaggedness2, sum of squares of rmt, sum of squares of inter-radii differences, and the scaled sum of squares of inter-radii differences. The jaggedness feature measures the smoothness of the vector by constructing a set of vectors from the RMT for a point p_0, and then computing the cosine of the angles between the vectors. The resulting cosine is then shifted and scaled to give a response that increases with the angle. The exact formulation of the jaggedness measure is given in Equations 5.9 – 5.11.

$$\text{jaggedness}\left(\text{RMT}p_0\right) = \sum_{i=1}^{\rho_{max}-2} \frac{1-\Theta\left(\text{rmt}_{p_0,i}, \text{rmt}_{p_0,i+1}, \text{rmt}_{p_0,i+2}\right)}{2} \tag{5.9}$$

$$\text{RMT}p_0 = \langle \text{rmt}_{p_0,1}, \text{rmt}_{p_0,2}, \ldots, \text{rmt}_{p_0,\rho_{max}} \rangle \tag{5.10}$$

$$\Theta\left(a,b,c\right) = \frac{(a-b)\times(b-c)+1}{\sqrt{((a-b)^2+1)}\times\sqrt{((b-c)^2+1)}} \tag{5.11}$$

The jaggedness2 measure is similar to jaggedness, but instead sums the absolute cosines of the angles between vectors constructed from the RMT. The formulation for jaggedness2 is given in Equation 5.12.

$$\text{jaggedness2}\left(\text{RMT}p_0\right) = \sum_{i=1}^{\rho_{max}-2} \left|\Theta\left(\text{rmt}_{p_0,i}, \text{rmt}_{p_0,i+1}, \text{rmt}_{p_0,i+2}\right)\right| \tag{5.12}$$

The sum of squares of rmt (SSR) feature computes the L_2 norm, or vector length of the RMT. The sum of squares of inter-radii differences (SSIRD) feature is used to measure the amount of change in the RMT by looking at the length of the vector produced by taking the difference of the adjacent elements in the RMT. The scaled sum of squares of inter-radii differences (SSSIRD) is the ratio of the SSIRD to the SSR. The formal definitions of these

features are given in Equations 5.13–5.15.

$$\mathrm{SSR}\left(\mathrm{RMT}p_0\right) = \sqrt{\left(\sum_{i=1}^{\rho_{max}} \mathrm{rmt}_{p_0,i}^2\right)} \tag{5.13}$$

$$\mathrm{SSIRD}\left(\mathrm{RMT}p_0\right) = \sqrt{\left(\sum_{i=1}^{\rho_{max}-1} \mathrm{rmt}_{p_0,i}^2 - \mathrm{rmt}_{p_0,i+1}^2\right)} \tag{5.14}$$

$$\mathrm{SSSIRD}\left(\mathrm{RMT}p_0\right) = \frac{\mathrm{SSIRD}\left(\mathrm{RMT}p_0\right)}{\mathrm{SSR}\left(\mathrm{RMT}p_0\right)} \tag{5.15}$$

Examples of each of these features is given Section 5.7.3.

5.7 Experiments

5.7.1 Validation of Discrete RMT Rotational Invariance

The RMT property of rotational invariance is essential for detecting similar features in different regions in an image and for detecting the same features in different data sets of the same object. To test how close the discrete RMT is to being rotationally invariant, in the first test, a set of 421 different phantoms were constructed. The phantoms were cylinders with a radius of 5 units inside a cubic region 21 units on a side. The cylinders were specified via a line segment and radius. One end of the line segment was placed at the center of the volume, while the other end was placed at 421 regularly spaced locations on one face of the cube. The second test for rotational invariance was done using two sets of synthetic volumes that were generated from a real spinal MRI and a single vertebra MRI. The procedure for generating each phantom was to select a sub-region of the real volume and generate a series of rotated versions of the volume. All rotations were done first about the y-axis, and then about the z-axis. The rotated volume was then sampled to construct a new volume for computing the RMT. The rotation and sampling procedure was conducted for rotation angles from $-\pi$ to π using $\frac{2\pi}{91}$ steps. For each generated volume,

the RMT was computed and stored. For both sets of synthetic volumes, the error in the RMT was then computed by computing the Euclidian distance between the RMTs for each orientation and the RMT for the unrotated volume. The errors for the RMT computation in these experiments are shown in Figures 5.3–5.5.

In Figure 5.3 the error is symmetric, indicating that for the cylindrical phantom, the RMT is rotationally invariant within the constraints imposed by discretizing the transform. The error plots for the MRI derived phantoms, shown in Figures 5.4 and 5.5, while not ex-

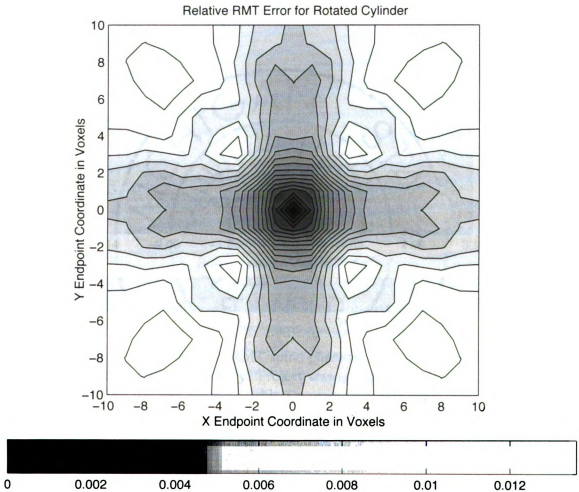


Figure 5.3: Relative error plot for RMT cylinder phantom

hibiting the symmetry of the cylinder phantom, are periodic. The difference in the errors is due in part to the interpolation algorithms used to generate the phantoms. The error results in Table 5.2 indicate that the largest error was 2.82%, for the phantom generated from a

single vertebra. As previously noted, one of the sources of error in the RMT computation is the problem of partial voxel occupancy.

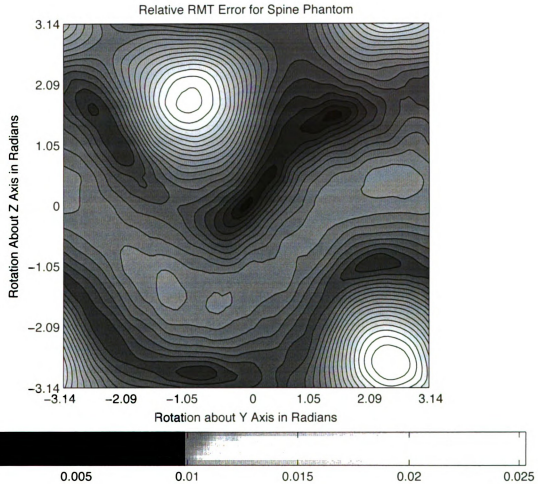


Figure 5.4: Relative error plot for RMT spine phantom. MR experimental data provided by James E Siebert, MSU Radiology. Project was approved by the MSU Committee for Research Involving Human Subjects; subject written informed consent was obtained.

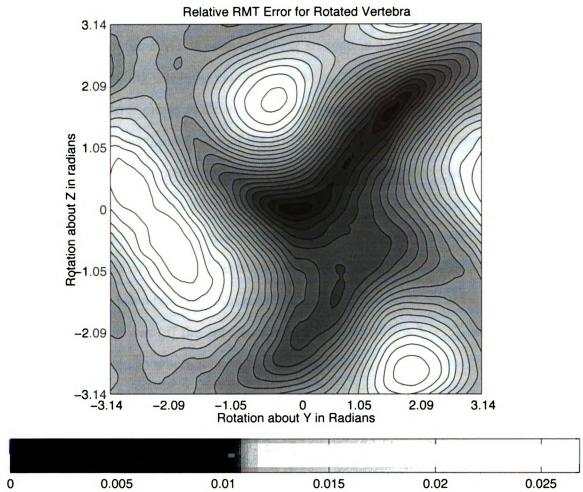


Figure 5.5: Relative error plot for RMT vertebra phantom. MR experimental data provided by James E Siebert, MSU Radiology. Project was approved by the MSU Committee for Research Involving Human Subjects; subject written informed consent was obtained.

Table 5.2: Results from RMT rotational invariance tests.

	Number of	Relative Error				
Data Set	Orientations	Mean	Std Dev	Median	Min	Max
Cylinder Phantom	441	0.0117	0.0021	0.0122	0.0000	0.0143
Spine	8281	0.0115	0.0049	0.0110	0.0000	0.0266
Vertebra	8281	0.0164	0.0065	0.0169	0.0000	0.0282

5.7.2 RMT Response

As the RMT is computed, mass is integrated in a spherical shell around the center of the transform. When the RMT is computed near an edge in an otherwise uniform volume, it will begin responding when the center is within ρ_{max} voxels of the edge. As the center of the transform moves closer to the edge, the response moves towards ρ_0 in the RMT.

Figure 5.6 shows the response of the RMT as it is applied to a phantom that contains a sphere of radius 1 at the geometric center of an otherwise uniform volume. When the RMT is computed at slices < 31 there is no response, that is the RMT is zero, since the sphere has not been encountered. At slice 31, the sphere is encountered for the RMT component ρ_{10}, while the rest of the RMT has a zero response. As the slice increases, the portion of the RMT that shows a response moves towards ρ_1 until the RMT is centered on the sphere. Once the RMT moves off the sphere, the response moves towards ρ_{10} until the center of the RMT is at slice 53 where it has a zero response again. Figures 5.7—5.9 show the same computation for phantoms with spheres of radii 4, 8, and 10. In all cases the response of the RMT is symmetric. For the sphere of radius 10, the response will not contain zeros at ρ_{10} since the RMT is smaller than the sphere causing the response.

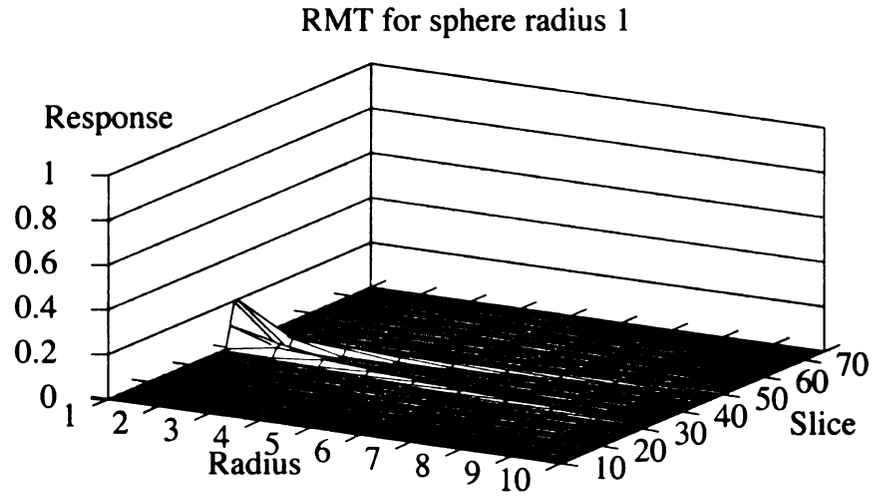


Figure 5.6: RMT response for sphere phantom with radius 1

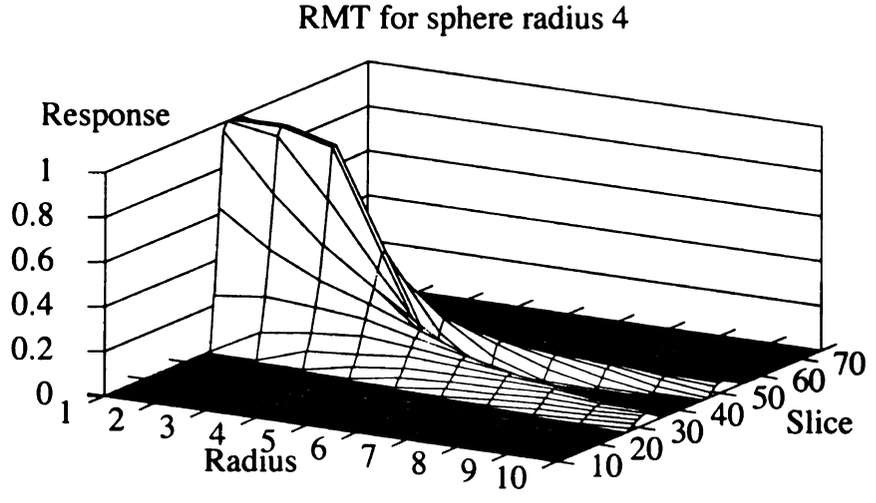


Figure 5.7: RMT response for sphere phantom with radius 4

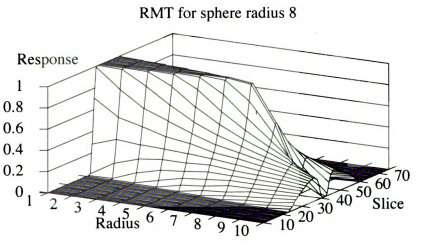


Figure 5.8: RMT response for sphere phantom with radius 8

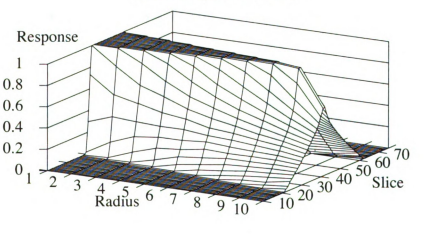


Figure 5.9: RMT response for sphere phantom with radius 10

5.7.3 RMT Feature Tests

Ideally the RMT will generate a non-smooth transform in the presence of texture but a relatively smooth transform when structure is not present. To test this behavior, several different phantoms were generated. All phantom volumes were 128^3. After the volumes were generated, the RMT was computed for each volume with $\rho \in [1, 10]$. From the RMT a set of features was computed. Several of the phantoms are discussed below along with their RMTs and derived features. Additional phantoms are given in Appendix B.

1. **Uniform volume**. A volume with a single value is generated. The RMT should give a flat response to this phantom, since there is no variation dependent on radius in the volume.

   In the experiments, the RMT for the uniform volume phantom, Figure 5.10, had a completely flat response. The lack of response in a uniform area is appropriate if we apply the heuristic that the interior of uniform regions is uninteresting. The square regions in the figure correspond to the mass for radius r with the transform centered at that voxel. A schematic of this layout is shown in Figure 5.11.

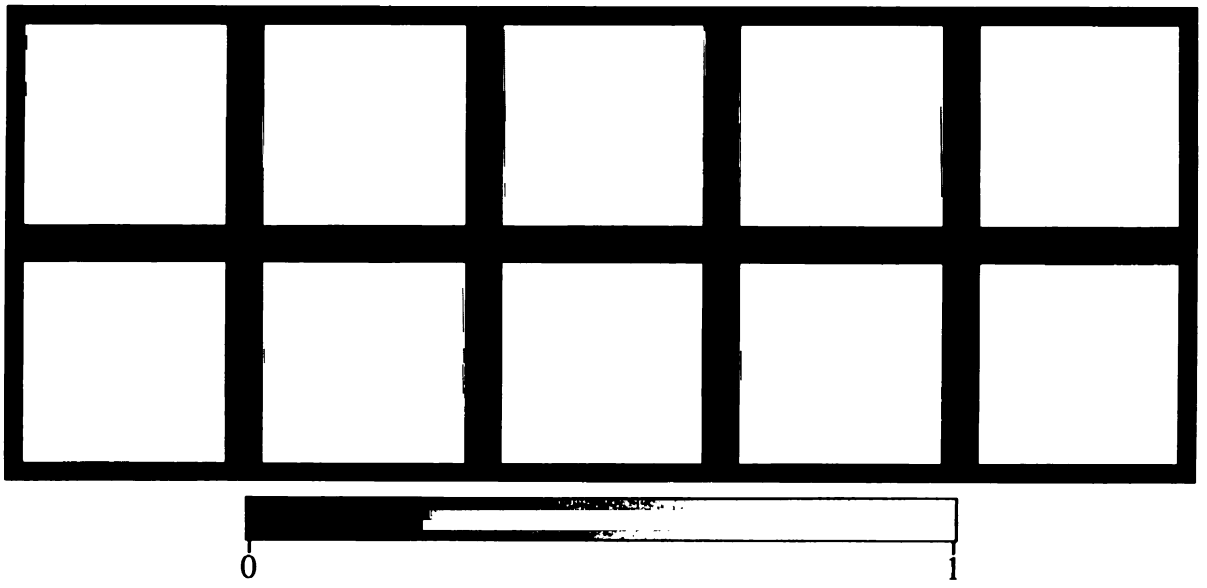


Figure 5.10: Discrete RMT of length 10 for uniformVolume phantom slice 64.

✚ (X,Y) Radius 1	✚ (X,Y) Radius 2	✚ (X,Y) Radius 3	✚ (X,Y) Radius 4	✚ (X,Y) Radius 5
✚ (X,Y) Radius 6	✚ (X,Y) Radius 7	✚ (X,Y) Radius 8	✚ (X,Y) Radius 9	✚ (X,Y) Radius 10

Figure 5.11: Schematic for graphical RMT display.

2. **Single ramp**. A gradient is generated that goes from 0 to 1 across the volume as in Figure 5.12.

Figure 5.12: singleRamp phantom slice 64.

Figure 5.13 shows a graphical representation of the RMT. For this phantom, the RMT was computed using a maximum radius of 10. The figure displays the normalized integrated mass at each of the radii of the RMT for a particular slice.

The RMT returns a smooth response to a smooth gradient, with the jaggedness measure for the RMT uniform when the RMT is computed for smooth regions. The jaggedness response is shown in Figure 5.14. The jaggedness2 response is the same

and is not shown.

The SSR feature, Figure 5.15, shows a smooth gradient response similar to the original input.

The SSIRD and SSSIRD features, Figures 5.16 and 5.17, do not show a smooth response. The non-smooth response is a function of looking at the squared amount of change as a function of the radius in the RMT. The advantage of this type of response is that we are able to generate a response in regions with a uniform gradient, but non-uniform intensity.

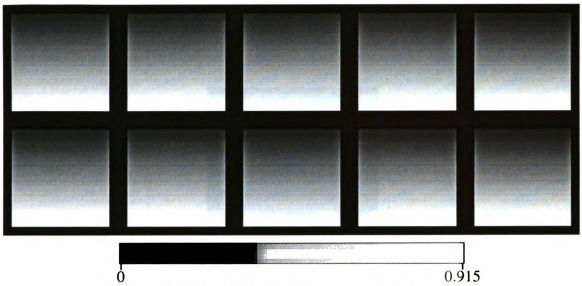


Figure 5.13: RMT for single ramp gradient phantom.

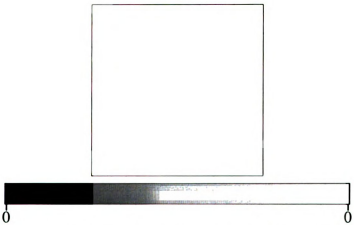


Figure 5.14: Jaggedness feature singleRamp phantom slice 64.

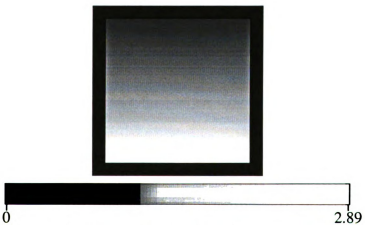


Figure 5.15: SSR feature singleRamp phantom slice 64.

Figure 5.16: SSIRD feature singleRamp phantom slice 64.

Figure 5.17: SSSIRD feature singleRamp phantom slice 64.

3. **Single radial gradient**. A single line is run between two corners of the volume and the Euclidian Distance Map (EDM) is generated. Figure 5.18 shows a slice of the phantom, while Figure 5.19 shows the RMT for the same slice.

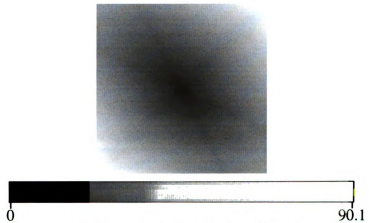


Figure 5.18: Slice of single radial gradient phantom.

In the RMT image, a blurring effect can be seen where the image energy is spread more as the radius from the center of the transform increases.

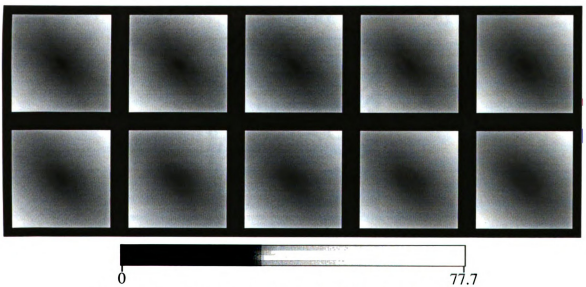


Figure 5.19: RMT for slice of single radial gradient phantom.

The jaggedness feature for the single radial gradient phantom, Figure 5.20, gives a strong response close to the origin of the radial gradient. There is a weaker response

along a set of lines within the slice that correspond to artifacts in the phantom from the EDM algorithm. The jaggedness2 feature had a flat response and is not shown.

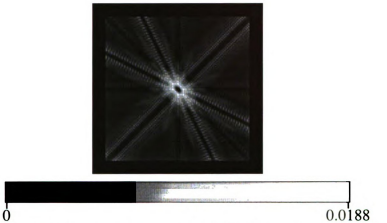


Figure 5.20: Jaggedness feature singleRadialGradient phantom slice 64.

The SSR feature, Figure 5.21, can be considered as the magnitude of the total energy of the RMT. Even in a situation where the RMT has a smooth response, the SSR feature will have a stronger response as the amount of integrated energy increases. This response is used later in the SSSIRD feature as a scaling factor.

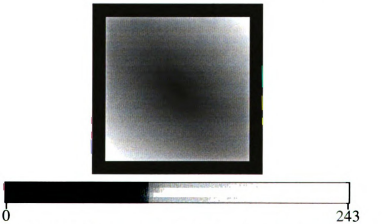


Figure 5.21: SSR feature singleRadialGradient phantom slice 64.

The SSIRD feature, Figure 5.22, has a high response when the RMT is jagged, and when the RMT has a steep slope. Here the response is strong near the center of the radial gradient, and along the regions where there are EDM artifacts. In the artifact

regions it is strong where the jaggedness feature had a strong response, and where the jaggedness feature had very little response. The SSSIRD feature, Figure 5.23, is the SSIRD feature scaled by the SSR feature. The scaling causes the SSSIRD feature to have a stronger response where the RMT magnitude is low.

Figure 5.22: SSIRD feature singleRadialGradient phantom slice 64.

Figure 5.23: SSSIRD feature singleRadialGradient phantom slice 64.

4. **Multiple radial gradients**. The phantom, Figure 5.24, is generated by running a set of lines through the volume, and then generating an EDM from the lines.

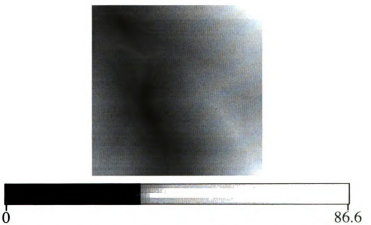


Figure 5.24: multipleRadialGradients phantom slice 64.

The RMT, Figure 5.25, shows the blurring effect where sharp features become less pronounced as the radius of the transform increases.

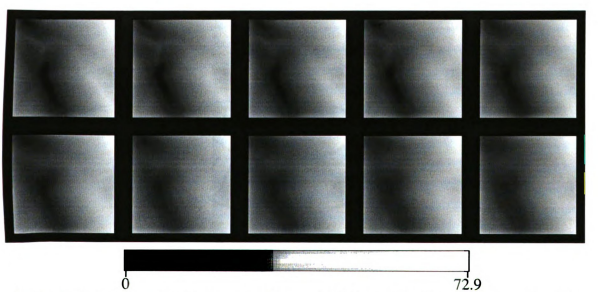


Figure 5.25: Discrete RMT of length 10 for multipleRadialGradients phantom slice 64.

The bright ridge in the phantom shows up as a ridge in the jaggedness feature, Figure 5.26. The dark ridges also show up in the jaggedness feature. In general, local maxima and minima will produce a non-zero response in the jaggedness feature.

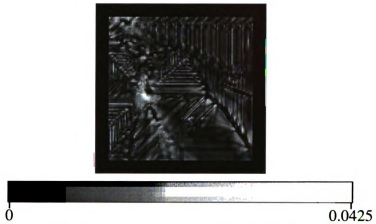


Figure 5.26: Jaggedness feature multipleRadialGradients phantom slice 64.

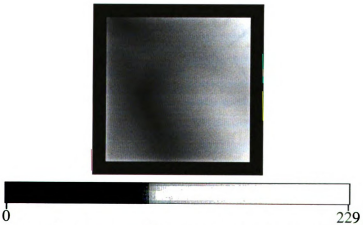


Figure 5.27: SSR feature multipleRadialGradients phantom slice 64.

The SSIRD, Figure 5.28, shows a high response in the regions of the phantom with local maxima and minima. The response falls off as the transform moves away from the maxima and minima. The SSSIRD, Figure 5.29, shows a similar response, but is weighted towards regions in the phantom with low energy.

Figure 5.28: SSIRD feature multipleRadialGradients phantom slice 64.

Figure 5.29: SSSIRD feature multipleRadialGradients phantom slice 64.

5. **Multiple tubes**. A multiple radial gradient phantom is generated, and then thresholded to generate a binary volume, Figure 5.30.

   Again, the response of the RMT in uniform regions should be flat, while near the edges of the cylinders it will be non-smooth. The junctions between cylinders should also give a non-smooth response. In Figure 5.31, a response in the region between two cylinders can be seen in the bottom row.

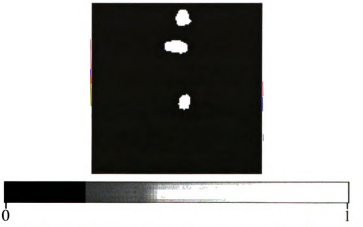


Figure 5.30: multipleTubes phantom slice 64.

Figure 5.31: Discrete RMT of length 10 for multipleTubes phantom slice 64.

The SSR feature, Figure 5.32, shows a strong response on the tubes in the phantom,

and then falls of quickly as the center of the transform moves away from the tubes. The SSSIRD feature, Figure 5.33, has a stronger response at the center of the region between the two tubes. This response is caused by a nearby junction between the tubes.

Figure 5.32: SSR feature multipleTubes phantom slice 64.

Figure 5.33: SSSIRD feature multipleTubes phantom slice 64.

6. **Subvolume of scutigera**. A portion of a scutigera head is used as a phantom (Figure 5.34). The RMT, Figure 5.35, shows a blurring effect where the transform is averaging mass at progressively greater radii. The RMT for the Scutigera has negative values where the average value of the voxels in a spherical shell was negative.

Figure 5.34: ScutigeraSub phantom slice 64.

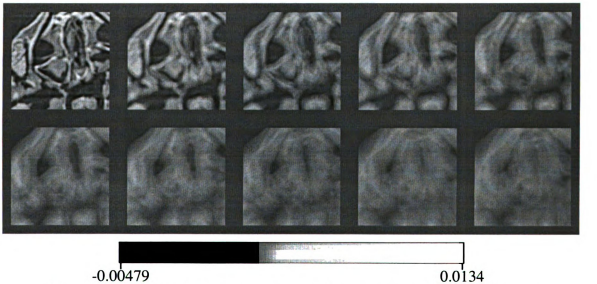


Figure 5.35: Discrete RMT of length 10 for ScutigeraSub phantom slice 64.

The jaggedness feature, Figure 5.36, shows the strongest responses near corners in the volume. The SSR feature, Figure 5.37, shows the strongest responses in bright regions of the volume. The SSIRD and SSSIRD features, Figures 5.38 and 5.39,

have their responses concentrated along linear structures in the volume. These linear structures correspond to the exo- and endo-skeleton of the scutigera.

Figure 5.36: Jaggedness feature ScutigeraSub phantom slice 64.

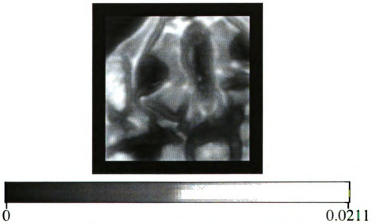


Figure 5.37: SSR feature ScutigeraSub phantom slice 64.

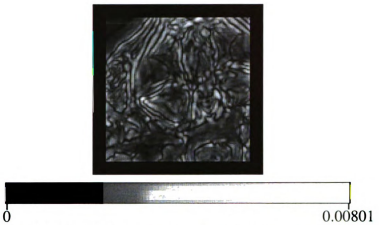


Figure 5.38: SSIRD feature ScutigeraSub phantom slice 64.

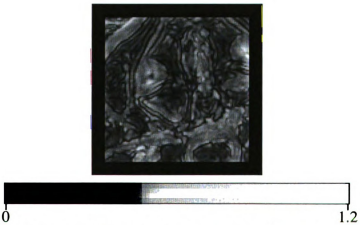


Figure 5.39: SSSIRD feature ScutigeraSub phantom slice 64.

7. **Jack**. A model of a children's jack was constructed as a phantom with a variety of regular features.

   The RMT of the jack is shown in Figure 5.41.

Figure 5.40: Jack phantom slice 64.

Figure 5.41: Discrete RMT of length 10 for jack phantom slice 64.

The jaggedness feature for the jack phantom in Figure 5.42.

Figure 5.42: Jaggedness feature for jack phantom slice 64.

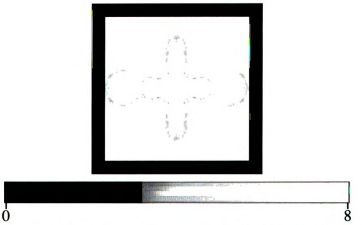


Figure 5.43: Jaggedness2 feature for jack phantom slice 64.

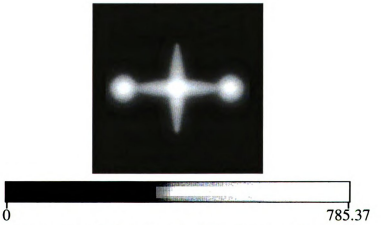


Figure 5.44: SSR feature for jack phantom slice 64.

Figure 5.45: SSIRD feature for jack phantom slice 64.

Figure 5.46: SSSIRD feature for jack phantom slice 64.

5.7.4 Performance

The computational cost of the RMT algorithm is high. For a realistic volume, the total amount of CPU time consumed is measured in hours, if not days. One approach to minimizing the wall-clock time is to compute the RMT in parallel and store it for later use. Table 5.3 summarizes the size and number of transforms for several real data sets. Tables 5.4 – 5.9 show the results from a set of performance experiments on these data sets.

Table 5.3: Sizes of real data sets and number of transforms in millions.

Data Set	Rows	Columns	Slices	Number of Transforms		
				$\rho_{max} = 2$	$\rho_{max} = 5$	$\rho_{max} = 10$
5atah1_a	658	658	311	131.3	126.4	118.4
5atah2_a	658	658	290	122.3	117.6	109.9
5atah3_a	658	658	250	105.2	100.8	93.6
5ath1_a	658	658	386	163.4	157.9	149.0
5ath2_a	658	658	403	170.7	165.0	155.9
5ath4_a	658	658	258	108.6	104.1	96.9
5btah2_a	658	658	293	123.6	118.8	111.1
5btah4_a	658	658	315	133.0	128.1	120.1
Scutigera	512	512	316	80.5	77.1	71.7
imp._bullion_a	341	341	381	42.8	40.6	37.2
spine	512	512	68	16.5	14.6	11.6
all phantoms	128	128	128	1.9	1.6	1.3

All of the phantoms have the same size, shown in the last row of Table 5.3. Tables 5.4 – 5.6 show that the computational effort for the RMT is constant for a given ρ_{max} regardless of the underlying contents of the data. Variations in the wall-clock times are attributable to network congestion and waiting for commands from the job server.

The real data sets were run using the same experimental setup, with the results shown in Tables 5.7 – 5.9. For data sets with the same slice dimension, the per-slice execution times were within 1 second. With the exception of the Scutigera data set, the execution times increased as the number of voxels increased. The Scutigera data set had a slice dimension of 512 by 512. This data size may have resulted in poor performance with the CPU data-

Table 5.4: RMT Performance on phantom data using 18 hosts, $\rho_{max} = 2$. Times are given in Hours:Minutes:Seconds.

Data Set	CPU Time					Wall Time Total
	Volume				Slice Total	
	Load	Process	Store	Total		
ScutigeraSub	0:00:10	0:00:42	0:00:27	0:01:19	0:00:01	0:00:11
jack	0:00:10	0:00:42	0:00:27	0:01:18	0:00:01	0:00:11
multipleRadialGradients	0:00:10	0:00:41	0:00:27	0:01:18	0:00:01	0:00:18
multipleRadialGradients_100	0:00:10	0:00:41	0:00:27	0:01:18	0:00:01	0:00:11
multipleTubes	0:00:10	0:00:42	0:00:27	0:01:18	0:00:01	0:00:11
pointGradients	0:00:10	0:00:42	0:00:27	0:01:19	0:00:01	0:00:12
singlePointGradient	0:00:10	0:00:42	0:00:27	0:01:18	0:00:01	0:00:12
singleRadialGradient	0:00:10	0:00:42	0:00:27	0:01:19	0:00:01	0:00:22
singleRamp	0:00:10	0:00:42	0:00:27	0:01:18	0:00:01	0:00:11
singleSphere	0:00:10	0:00:42	0:00:27	0:01:19	0:00:01	0:00:12
singleTube	0:00:10	0:00:42	0:00:27	0:01:19	0:00:01	0:00:11
spheres	0:00:10	0:00:41	0:00:27	0:01:18	0:00:01	0:00:12
spheres_100	0:00:09	0:00:41	0:00:27	0:01:18	0:00:01	0:00:11
uniformVolume	0:00:10	0:00:42	0:00:27	0:01:19	0:00:01	0:00:12

Table 5.5: RMT Performance on phantom data using 18 hosts, $\rho_{max} = 5$. Times are given in Hours:Minutes:Seconds.

Data Set	CPU Time					Wall Time Total
	Volume				Slice	
	Load	Process	Store	Total	Total	
ScutigeraSub	0:00:19	0:04:39	0:00:39	0:05:37	0:00:03	0:00:28
jack	0:00:19	0:04:38	0:00:39	0:05:36	0:00:03	0:00:29
multipleRadialGradients	0:00:18	0:04:38	0:00:39	0:05:35	0:00:03	0:01:10
multipleRadialGradients_100	0:00:19	0:04:38	0:00:39	0:05:37	0:00:03	0:00:28
multipleTubes	0:00:19	0:04:38	0:00:39	0:05:36	0:00:03	0:00:29
pointGradients	0:00:19	0:04:38	0:00:39	0:05:36	0:00:03	0:00:27
singlePointGradient	0:00:19	0:04:38	0:00:39	0:05:37	0:00:03	0:00:27
singleRadialGradient	0:00:20	0:04:38	0:00:39	0:05:36	0:00:03	0:00:28
singleRamp	0:00:20	0:04:38	0:00:39	0:05:37	0:00:03	0:00:28
singleSphere	0:00:20	0:04:38	0:00:39	0:05:37	0:00:03	0:00:30
singleTube	0:00:20	0:04:38	0:00:39	0:05:36	0:00:03	0:00:31
spheres	0:00:20	0:04:38	0:00:39	0:05:37	0:00:03	0:00:29
spheres_100	0:00:19	0:04:38	0:00:39	0:05:37	0:00:03	0:00:27
uniformVolume	0:00:19	0:04:38	0:00:39	0:05:37	0:00:03	0:00:27

Table 5.6: RMT Performance on phantom data using 18 hosts, $\rho_{max} = 10$. Times are given in Hours:Minutes:Seconds.

Data Set	CPU Time					Wall Time Total
	Volume				Slice Total	
	Load	Process	Store	Total		
ScutigeraSub	0:00:32	0:26:58	0:00:47	0:28:17	0:00:16	0:01:54
jack	0:00:31	0:27:00	0:00:47	0:28:19	0:00:16	0:01:53
multipleRadialGradients	0:00:32	0:26:57	0:00:47	0:28:16	0:00:16	0:01:56
multipleRadialGradients_100	0:00:32	0:26:58	0:00:47	0:28:17	0:00:16	0:01:51
multipleTubes	0:00:32	0:26:57	0:00:47	0:28:16	0:00:16	0:01:52
pointGradients	0:00:32	0:26:59	0:00:47	0:28:17	0:00:16	0:01:52
singlePointGradient	0:00:32	0:26:59	0:00:47	0:28:18	0:00:16	0:01:51
singleRadialGradient	0:00:33	0:26:58	0:00:47	0:28:18	0:00:16	0:01:46
singleRamp	0:00:32	0:26:58	0:00:47	0:28:17	0:00:16	0:01:44
singleSphere	0:00:32	0:26:58	0:00:47	0:28:18	0:00:16	0:01:49
singleTube	0:00:32	0:26:57	0:00:47	0:28:16	0:00:16	0:01:54
spheres	0:00:31	0:26:59	0:00:47	0:28:17	0:00:16	0:02:00
spheres_100	0:00:32	0:26:57	0:00:47	0:28:15	0:00:16	0:02:01
uniformVolume	0:00:32	0:26:59	0:00:47	0:28:17	0:00:16	0:02:00

cache.

Table 5.7: RMT Performance on real data using 18 hosts, $\rho_{max} = 2$. Times are given in Hours:Minutes:Seconds.

	CPU Time					Wall
	Volume				Slice	Time
Data Set	Load	Process	Store	Total	Total	Total
5atah1_a	0:05:45	0:42:02	0:36:24	1:24:11	0:00:16	0:09:42
5atah2_a	0:05:21	0:39:08	0:33:39	1:18:08	0:00:16	0:10:14
5atah3_a	0:04:35	0:33:38	0:29:15	1:07:29	0:00:16	0:09:27
5ath1_a	0:07:09	0:51:58	0:45:39	1:44:46	0:00:16	0:13:46
5ath2_a	0:07:28	0:54:18	0:47:47	1:49:32	0:00:16	0:13:51
5ath4_a	0:04:47	0:34:34	0:30:39	1:09:59	0:00:17	0:08:50
5btah2_a	0:05:24	0:39:32	0:34:06	1:19:01	0:00:16	0:09:57
5btah4_a	0:05:48	0:42:34	0:36:39	1:25:01	0:00:16	0:10:59
Scutigera	0:03:37	0:29:53	0:21:15	0:54:45	0:00:11	0:06:59
imp._bullion_a	0:02:05	0:13:50	0:10:35	0:26:31	0:00:04	0:03:46
spine	0:00:44	0:06:05	0:04:20	0:11:09	0:00:10	0:01:50

Table 5.8: RMT Performance on real data using 18 hosts, $\rho_{max} = 5$. Times are given in Hours:Minutes:Seconds.

	CPU Time					Wall
	Volume				Slice	Time
Data Set	Load	Process	Store	Total	Total	Total
5atah1_a	0:12:18	5:38:02	0:59:53	6:50:12	0:01:22	0:27:20
5atah2_a	0:11:27	5:14:38	0:55:34	6:21:40	0:01:22	0:25:35
5atah3_a	0:09:49	4:29:41	0:47:43	5:27:13	0:01:22	0:21:54
5ath1_a	0:15:22	7:02:36	1:15:03	8:33:01	0:01:22	0:33:51
5ath2_a	0:16:05	7:21:44	1:18:11	8:56:00	0:01:22	0:35:02
5ath4_a	0:10:08	4:38:44	0:49:13	5:38:04	0:01:22	0:22:39
5btah2_a	0:11:34	5:18:00	0:56:18	6:25:53	0:01:22	0:26:02
5btah4_a	0:12:27	5:42:43	1:00:44	6:55:54	0:01:22	0:28:14
Scutigera	0:07:50	4:39:48	0:34:28	5:22:05	0:01:03	0:20:30
imp._bullion_a	0:04:27	1:49:02	0:17:17	2:10:46	0:00:21	0:09:15
spine	0:01:28	0:53:05	0:06:26	1:00:59	0:01:03	0:04:42

Table 5.9: RMT Performance on real data using 18 hosts, $\rho_{max} = 10$. Times are given in Hours:Minutes:Seconds.

	CPU Time					Wall
	Volume				Slice	Time
Data Set	Load	Process	Store	Total	Total	Total
5atah1_a	0:22:49	39:03:25	1:41:59	41:08:14	0:08:29	2:34:34
5atah2_a	0:21:13	36:14:02	1:34:50	38:10:05	0:08:29	2:24:33
5atah3_a	0:18:04	30:51:50	1:20:38	32:30:31	0:08:29	2:07:28
5ath1_a	0:28:45	49:06:55	2:08:12	51:43:52	0:08:29	3:13:20
5ath2_a	0:30:08	51:23:47	2:14:18	54:08:12	0:08:29	3:24:23
5ath4_a	0:18:48	31:56:16	1:23:36	33:38:40	0:08:29	2:08:51
5btah2_a	0:21:34	36:38:44	1:35:50	38:36:09	0:08:29	2:28:25
5btah4_a	0:23:18	39:36:06	1:43:34	41:42:58	0:08:29	2:36:03
Scutigera	0:14:24	33:46:41	0:51:18	34:52:24	0:07:04	2:07:33
imp._bullion_a	0:08:11	12:12:53	0:24:55	12:45:58	0:02:07	0:49:07
spine	0:02:21	5:28:50	0:08:23	5:39:35	0:07:04	0:23:10

Table 5.10 highlights the performance gains from parallelizing the execution of the RMT and interest detection computations. While the total CPU time for the parallel execution exceeded 1 day, the wall-clock times were relatively short. The best wall-clock performance comes from running the interest detection on a volume without using a cached RMT. Computing the RMT alone takes more time than the combined execution due to the overhead of generating the cache files, but provides the opportunity to reuse the stored results in later processing. The interest classification alone took the longest wall-clock time but was actually the fastest in terms of CPU time. This apparent discrepancy was caused by bottle necks in network data transfer. To minimize multiple interest classifications, the best approach would be to use cached RMTs with a parallel version of the interest classifier.

Table 5.10: RMT Performance

		CPU Time					Wall
		Volume				Slice	Time
Activity	Hosts	Load	Process	Store	Total	Total	Total
Interest/RMT	36	0:00:01	38:13:37	0:00:50	38:14:28	0:08:24	1:10:07
RMT	34	0:21:05	41:40:01	1:14:05	43:15:11	0:09:30	2:07:32
Interest	1	0:17:13	0:52:17	0:00:42	1:10:12	0:00:15	2:47:08

Figure 5.47 displays the timing results from each of the experimental runs, and a function fitted to the timing data for $\rho_{max} \in \{2, 5, 10\}$. The spurious data points for $\rho_{max} = 10$ and $\rho_{max} = 5$ are from the Scutigera sub-volume. One possible explanation for the excess execution time is that the Scutigera data slices were sized in such a way that data-cache misses were induced at a higher than normal rate.

Experiment Environment

The performance experiments were run using eighteen Sun Blade 100s, one Sun Ultra 10, and one Sun Fire V420. Figure 5.48 provides a schematic of the experiment environment. For the experiments, a job server is run on the Sun Ultra 10 that generates commands for running jobs on partial volumes. Each of the Sun Blades runs a client that receives

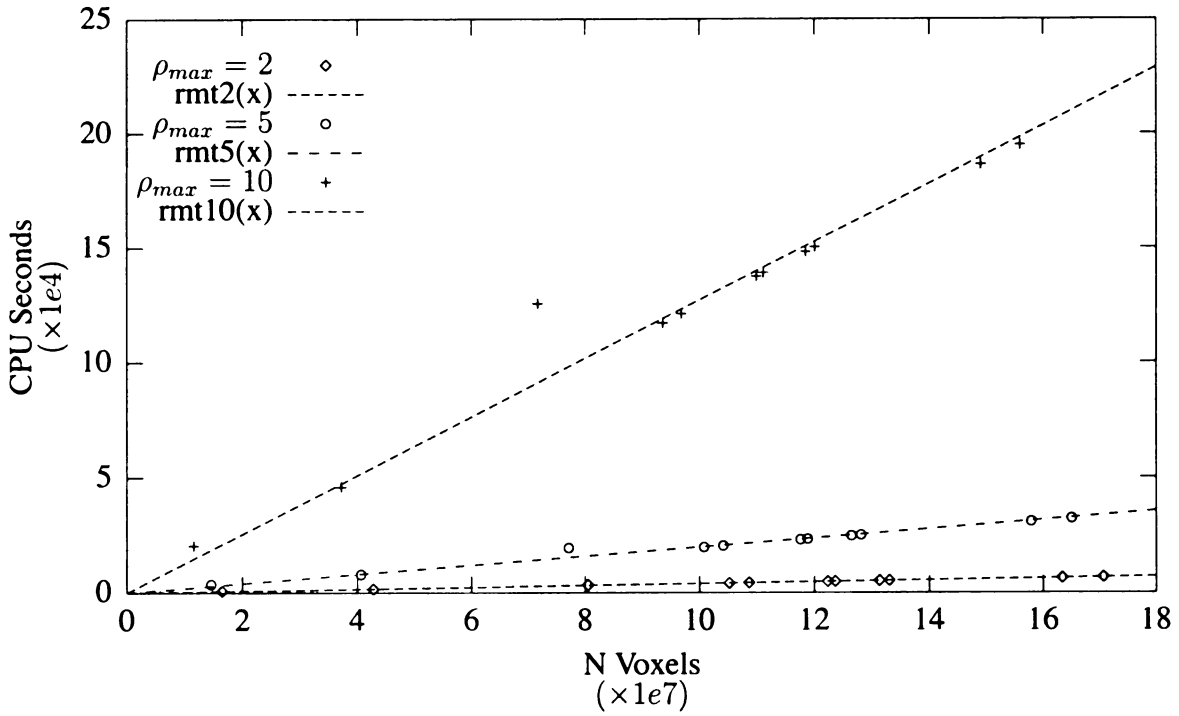


Figure 5.47: Execution time of RMT as a function of the number of voxels processed.

the commands from the server, and then executes the commands. All files that comprise a volume are stored on a RAID served by the Sun Fire V420. The client computers are connected via a 100 megabit-per-second connection to a gigabit switch. The gigabit switch in turn connects to a gigabit network that also hosts the file server. The job server is connected via a 100 megabit network link to the gigabit network hosting the file server. All timing information, with the exception of the wall-clock times, was collected from system timing information.

Each of the client computers was a sun Blade 100 with a 502 MHz CPU, and 128 MB of RAM. Each computer also had approximately 8 GB of available temporary storage for intermediate processing results. During the timing runs, the computers were used solely for computing the RMT.

The file server was connected to a RAID that provided access to the files that made up each volume. The file server was also responsible for storing the RMT results and the

performance logs.

The performance of the distributed execution can be improved without optimizing the implementations by adding more client computers to the compute pool, or by using faster computers with more RAM. The nature of the RMT is such that adding additional computers will improve performance until there is one computer per slice, or the file servers and network become the bottleneck.

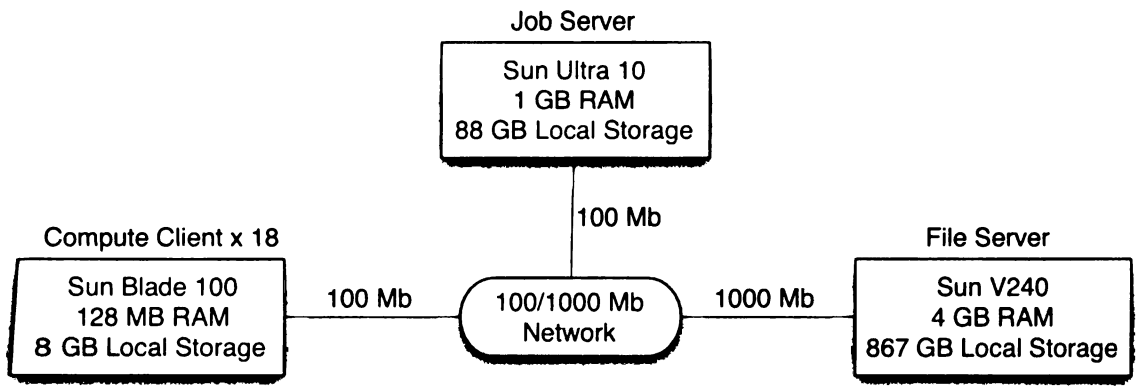


Figure 5.48: Schematic of performance experiment setup.

5.7.5 Future RMT Directions

In addition to using the RMT as a feature for interest detection, there may be some additional applications of the RMT. The RMT may facilitate registration by providing a mechanism for selecting discriminating regions within a volume. Discriminating regions may be identifiable by selecting regions where the RMT has a jagged response. By selecting the RMTs with the highest jaggedness measure, a constellation of points can be identified for use in other registration algorithms. The RMT may be useful for performing sphere-fitting operations on a n-ary segmented volume.

5.8 Interest Detection

One of the main purposes of studying the RMT is to use it as a feature for interest detection. The interest detection framework, outlined in Figure 5.1 on page 73, uses an SVM and the RMT.

The first stage of the interest classification process is having the user identify a set of interesting points P_i and a set of uninteresting points P_u such that $P_i \cap P_u = \emptyset$. The two sets of points provide the basis for generating the data model for the SVM. The RMT for each of the selected points is generated, and then used as labeled input to the SVM. In Figure 5.50, cross sections of a volume are presented, along with labeled interest points. The interesting points, as identified by the user, are shown as red cubes, while the uninteresting points are shown as blue diamonds.

5.8.1 Test Cases

The first test case is the jack phantom from Section 5.7.3. The phantom is shown in Figure 5.49. The training points for the jack were automatically selected during the phantom construction. The interesting points were points that were at the junction of two or more primitives at the surface of the phantom during construction. The points of interest fell in

the creases at the center of the jack, and at the intersections of the limbs of the jack where they connected to the spherical ends.

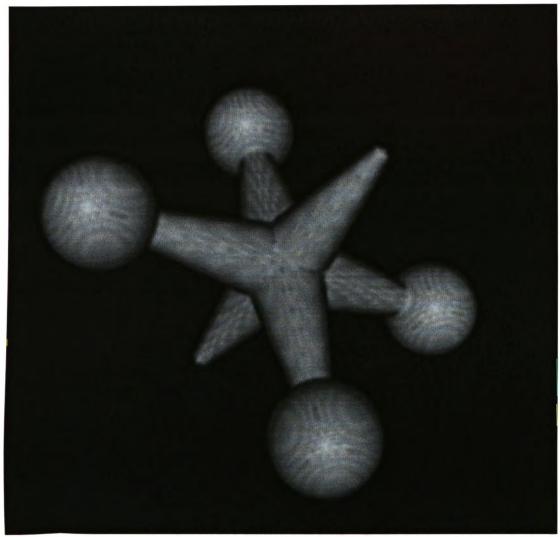


Figure 5.49: Rendering of jack phantom. The surface mottling is from the anti-aliasing used during phantom construction

The locations of the jack interest points for training the SVM are shown in Figure 5.50. There were 62 interesting and 59 uninteresting points used for training with all points located in one octant of the volume.

The result of the interest classification is shown in Figure 5.51. The central cube-like region corresponds to the ridges at the core of the jack. The four annular structures

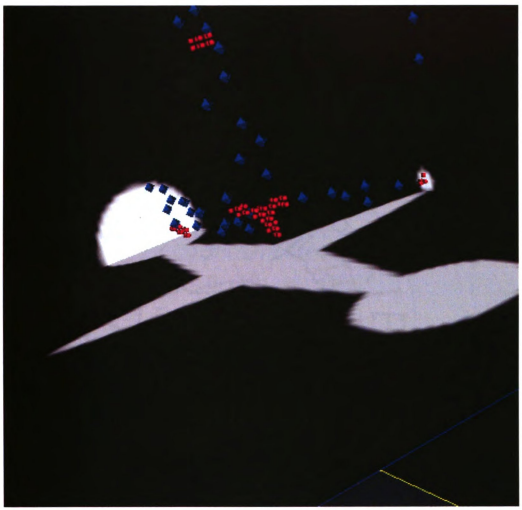


Figure 5.50: Training interest points for jack phantom. Cubes are interesting while diamonds are uninteresting.

correspond to the junctions of the limbs of the jack with the large spheres. The two dot-like regions correspond to the junction of the limbs of the jack with small spherical caps. Also faintly visible near the annular structures are interesting points where the surfaces of the large spheres caused a relatively weak interest response due to aliasing effects when the phantom was constructed.

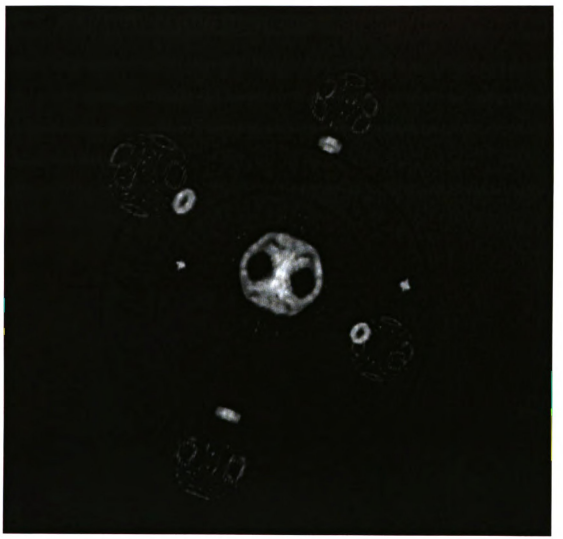


Figure 5.51: Interesting points for jack phantom.

For the soil sample, interesting points are at the junction of regions of dissimilar material. The interesting and uninteresting points were selected by hand, then used to train a SVM on the initial volume. The sample points are shown in Figure 5.52. After the model

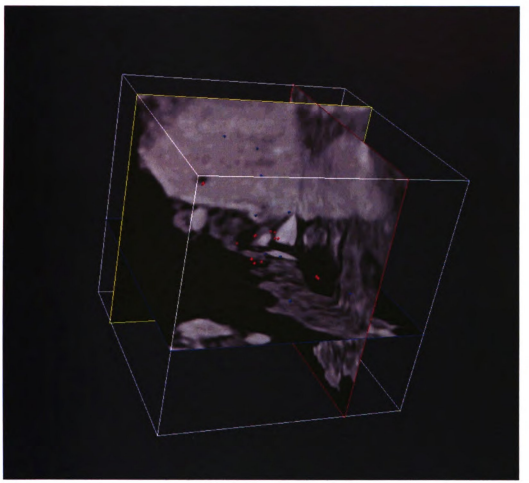


Figure 5.52: Interest points in soil aggregate.

was trained, the volume that contained the slice shown in Figure 5.53(a) was classified producing the interesting points shown in Figure 5.53(b). The original slice and the interest slice were then combined to highlight the interesting points in the original slice, shown in Figure 5.53(c). The model that was used for the volume in Figure 5.53 was also used on the volume shown in Figure 5.54. Since the texture of the volume was very different, few points in the second volume were classified as interesting. This phenomenon highlights

(a) Input Slice (b) Interest (c) Interest Overlay

Figure 5.53: Overlay of detected interest points on soil sample used for training.

the fact that the interest detection framework relies on having been trained on a representative sample of interesting and uninteresting points in order to effectively perform interest classification. Ultimately this behavior is caused by the SVM, since it generates the model used for classification directly from the training samples.

(a) Input Slice (b) Interest (c) Interest Overlay

Figure 5.54: Overlay of detected interest points on soil sample not used for training.

The last of the samples is the base of the bee stinger. In this sample, the interesting regions were the boundaries of the stinger with the air. The training points from the second round of interest selection are shown in Figure 5.55. Owing to the reconstruction artifacts

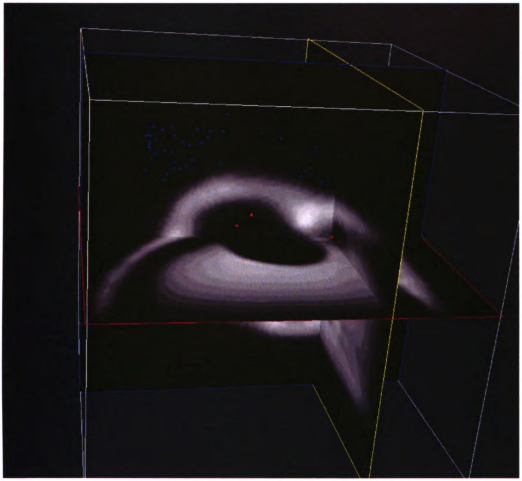
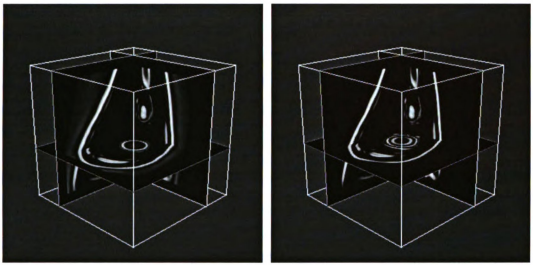


Figure 5.55: Training interest points for stinger; cubes are interesting while diamonds are uninteresting.

in the volume due to phase contrast tomography, there are large regions with smoothly varying intensity. Initial attempts to train and classify the volume were not successful, and are shown in Figure 5.56. Switching to the clustered version of the stinger volume allowed the interest classification to pick out the boundaries within the stinger, shown in Figure 5.57. The interest detection system was able to highlight the outer boundaries of the



(a) First Training Iteration (b) Second Training Iteration

Figure 5.56: Unsuccessful interest classifications of stinger. In the first iteration misclassified points form a cloud around the outside of the stinger. In the second iteration misclassified points form a circle on the inside of the stinger.

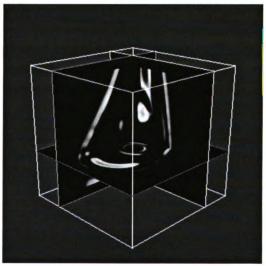


Figure 5.57: Successful interest classifications of stinger.

stinger, as well as the boundaries of the tube located inside the stinger. A 3D rendering of the detected stinger boundaries is shown in Figure 5.58.

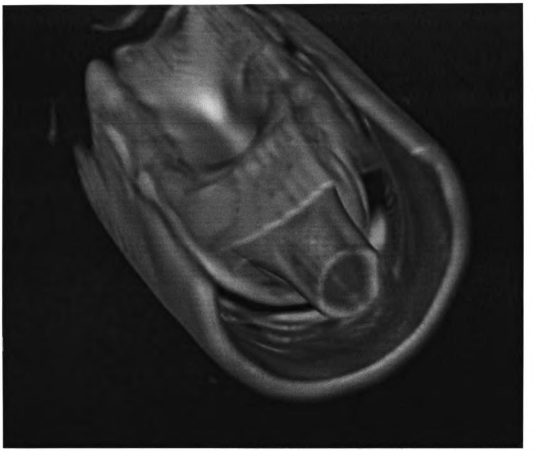


Figure 5.58: Rendering of interesting regions of stinger.

The SVM provides a robust mechanism for classification of a volume into interesting and uninteresting voxels. The classification can be either a binary classification or a continuous classification can be generated to provide information on how interesting a particular voxel is. The computational costs for the SVM are relatively low when compared to the cost of computing the RMT. When using the SVM for classification, the user must be sure that the classification of the original training points is correct. Future work in this area will include developing techniques to tune the SVM for interest classification.

5.9 Satisfaction of Interest Detection Criteria

Five criteria the interest detection algorithm must meet were identified at the beginning of this chapter.

1. Simplicity of computation.
2. Adaptability to different types of data.
3. Generates and preserves intermediate state information that enables a scientist to drive the interest detection towards a desired goal.
4. Amenable to incremental refinement.
5. The framework should be extendible as new interest detection techniques are developed.

While the RMT is computationally expensive, it is readily amenable to parallel computation, thus reducing wall-clock times to reasonable levels. The classification using the SVM is fast for single samples and is also amenable to parallel computation. The actual computations that are performed are simple; the cost in computation comes from processing large data sets. The individually inexpensive computations performed in parallel allow the interest detection algorithm to satisfy the criteria requiring computational simplicity.

The RMT can be computed for any type of data that is represented as a scalar field, independent of the origin of the data. The RMT can be extended to support vector fields at a cost of increased compute time. The SVM as used in the interest detection framework requires a vector as input, again independent of the source of the data. The simple input requirements of the RMT and SVM allow the interest detection algorithm to be applied to a wide range of data sources, satisfying the criteria that the algorithm be adaptable to different data types.

In order to amortize the cost of computing the RMT, it is computed once for a volume and then is reused in later processing. Training data is stored and can be used as a basis for

further training. Training the SVM takes at most several seconds. The ability to quickly retrain the SVM allows the user to explore different training sets easily. Since intermediate data is stored, it is easy to pursue different interest classifications from a common starting point. These factors allow the the criteria requiring storage of intermediate state and incremental refinement to be satisfied.

Finally, the SVM component of the feature detection framework is able to work with any form of data that can be given as a vector of scalar data. The SVM will operate best if all inputs are in similar ranges, requiring only that features be normalized. The simplest extension of the interest detection framework would be to augment the RMT vectors with second-order features computed from the RMT and other non-RMT based scalar features. The ability of the SVM to operate on longer input vectors that satisfy a simple range requirement allows the framework to satisfy the extendibility criteria.

5.10 Conclusions

The RMT is a novel feature that encodes information about the structure of the volume in a rotation- and translation-invariant manner. While the the RMT is expensive to compute sequentially for an entire volume, it is trivial to compute in parallel. Additionally the cost of computing the RMT can be amortized over several different classification cycles. The information encoded in the RMT is sufficient to allow for the generation of meaningful second-order features about the volumes. These features can, in future work, be used in addition to the raw RMT for performing classification. Additionally, the RMT shows some promise for performing volumetric registration.

Using the SVM along with the RMT allows for user specified interest detection without explicit encoding of *a priori* knowledge of interesting features. The framework presented in this chapter is amenable to the addition of other features to allow for a more robust interest classification system. While this framework is not suitable for fully automatic analysis, it

should provide assistance to the human expert in evaluating volumetric data.

Chapter 6

View Path Generation

The human visual system is capable of extracting 3D shape information from a series of 2D images. The focus of this chapter is developing a technique for generating a video stream that presents a series of cross sections of a volume to a user. The purpose of the producing a video stream is to help the user understand the structure of objects in the volume and to aid in the presentation of portions of the volume with high interest.

6.1 Problem Statement

View path generation addresses the problem of identifying a set of good viewing positions and the transitions between those viewing positions. This chapter focuses on identifying sets of cutting planes that maximize the amount of interest covered in a volume. After the planes are identified, a path satisfying a smoothness constraint is generated to move the view from cutting plane to cutting plane.

The goal of developing techniques to facilitate volumetric data analysis leads to the following criteria for the view path selection algorithm:

1. The path generation algorithm should be automatic.

2. The path should be smooth.

3. The path produced should maximize interest.

4. Multiple alternative paths should be generated during a path generation cycle.

5. The path generating system should allow user interaction, if not intervention.

6. The computational complexity should be low.

These criteria are discussed in the following sections.

6.1.1 Automatic Path Generation

Given a volume with n voxels there are potentially ${}_nP_m$ possible paths of length $1 \leq m \leq n$ if a path is restricted to visiting each voxel only once. If the restriction of visiting a voxel only once is relaxed, there are potentially an infinite number of paths through a volume. Given the size of the path space, the generation algorithm must be able to produce acceptable paths without user intervention. Manual intervention is problematic since re-slicing a volume is a computationally expensive operation, and many of the possible paths will not cover any interesting portions of the volume.

6.1.2 Smooth Paths

The paths that are generated should be smooth. If the path is non-smooth, the presentation of the path to the user may be disorienting. Jumps in the path through the volume will tend to hinder the user developing a model of structure present in the volume. In extreme cases, non-smooth or rapidly changing views may induce nausea in some users [53]. Non-smooth, or oscillating paths, may also cause a failure in 3D shape recovery for a user [65].

6.1.3 Interest Maximization

As noted above, the path space is extremely large. View paths selected by the path generation algorithm should maximize the amount of interest displayed along the path. By

requiring that the paths cover interesting portions of the volume, the computational effort of generating the rendering can be better justified.

6.1.4 Multiple Paths

For a volume with non-trivial structure there may be more than one path satisfying the smooth path and interest maximization criteria. The path generating algorithm should produce multiple paths, allowing for better coverage of the volume. The generation of multiple paths simultaneously allows for amortizing the cost of path generation.

6.1.5 User Interaction

While users may not be able to drive the path generation directly, they should be able to select from the available paths produced by the algorithm. The user may also be able to specify locations in the volume that must or must not be visited during a tour.

6.1.6 Computational Complexity

For any number of interest points n the number of potential paths that visit each point once is $O(n!)$. The consequence of this fact is that an exhaustive search of the path space for 10 interest points leads to over 3 million possible paths. A typical volume may have a minimum of 1000 interesting points, making exhaustive searches infeasible.

6.2 General Algorithm

The general algorithm for path generation takes as its input a set of n interesting points. As its output it produces a set of paths specified in terms of a set of cutting planes; each plane is specified in terms of a plane origin and normal. The components of the general algorithm are shown in Figure 6.1.

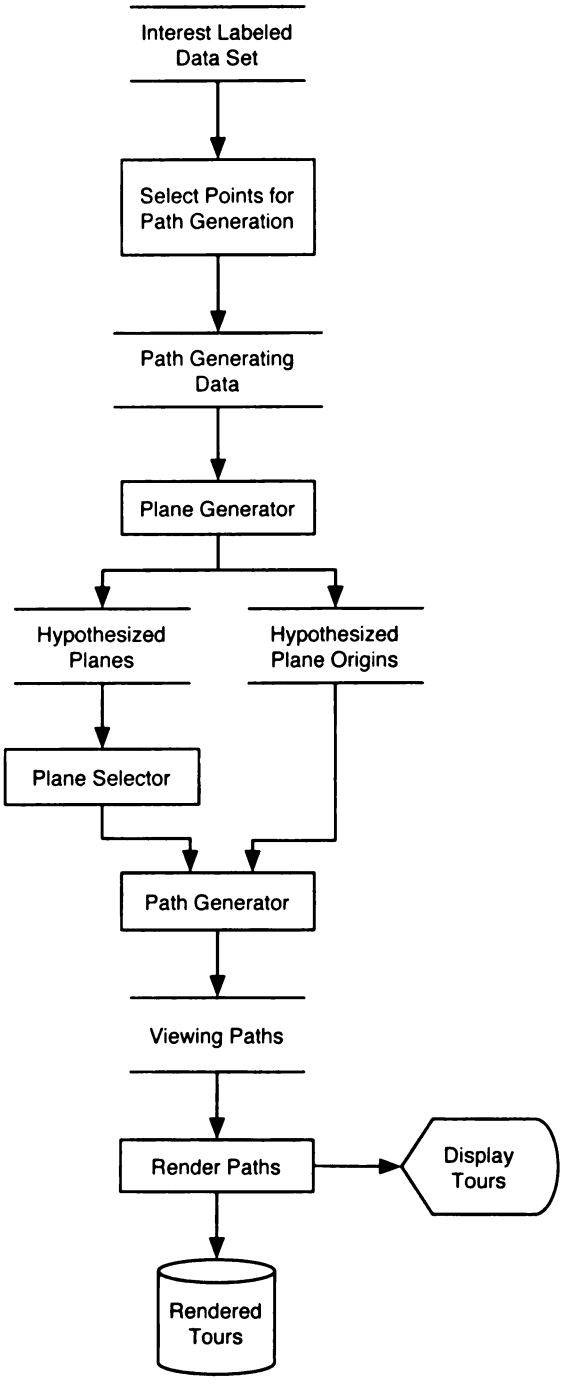


Figure 6.1: General algorithm for view path generation

The view path generation is related to identifying optimal projections. Hypothesized planes that maximize the amount interest that they display can be considered optimal. The remainder of this chapter discusses view path generation, but the techniques described for generating cutting planes are appropriate for for single projection selection.

6.3 Possible Approaches

There is a range of possible approaches to determining a path through the volume. The simplest approach is to simply scan along the coordinate axes in the volume. Simple scanning algorithms fail to meet the interest maximization and multiple path criteria identified in Section 6.1. The interest maximization criteria is not met since view planes maximizing interest may not be axis aligned. Simple axis scanning algorithms will fail when the structure in a data set is not axis aligned, see Figure 6.2 for an example. At the other extreme is examining all possible paths through the volume. Any exhaustive search algorithm will violate the computational complexity criteria. Graph theoretic approaches construct a graph using the interest points in a volume, and then compute the path via some traversal algorithm. The primary problems with this approach are determining where to place the graph vertices, the level of vertex connectivity, and the appropriate weighting for the edges. Computational geometry approaches attempt to take advantage of the geometric relations between the vertices. One computational geometry approach is using the Delaunay triangulation [23] and it's dual the Voronoi diagram to construct graphs suitable for some graph theoretic approach. Another computational geometry approach is to hypothesize cutting planes in the volume that satisfy some coverage property with respect to the interest points in the volume. The planes can be hypothesized via techniques such as the Delaunay triangulation and the Hough transform [52]. Once the planes have been hypothesized, a sorting algorithm can then be used to order the planes for display.

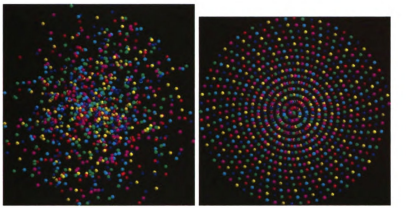


Figure 6.2: (a) Set of 1024 points viewed using an arbitrary projection. (b) Same set of points as (a) using a projection that highlights the structure of the points.

6.4 Proposed Algorithm

An approach to generating a set of viewing paths through a volume is given below.

1. Select a subset of high interest points from an interest classified volume.

2. Use a Hough transform parameterized as plane specifications in polar form to hypothesize cutting planes.

3. Perform non-maximal suppression to cull the candidate planes to those at local maxima in the plane parameter space. The non-maximal suppression serves to select planes for the path generation.

4. Construct a Delaunay triangulation of the culled plane parameterizations to construct a graph encoding smooth plane to plane transforms. (Use the QHull algorithm[6])

5. Construct a set of cutting planes from the plane specification graph using Dijkstra's single source shortest path algorithm [19].

6. Generate a view path using an interpolating spline to move the cutting plane location and normal through the volume.

6.4.1 Satisfaction of algorithm criteria

The algorithm proposed above satisfies all the criteria given in Section 6.1. Once the threshold has been selected for the interest points, none of the steps outlined in the algorithm require user intervention. Using a parameterization of the plane space in spherical coordinates enforces that spatially close points in the accumulator will have similar normals. The use of spline interpolation will ensure that the path itself will satisfy reasonable smoothness criteria. The Hough transform will accumulate interest along planes and the non-maximal suppression will select a subset of planes that have maximum interest in the accumulator space. Dijkstra's algorithm produces a tree structure. The current algorithm does not allow users to specify where in the volume the path will visit, but they may select among the paths that are generated. The traversal from the tree root to a leaf specifies a path. All of the sub-algorithms are computationally inexpensive, leading to an overall computationally inexpensive algorithm. A more detailed discussion of the computational complexity is given in the next section.

6.4.2 Computational complexity of algorithm

Each of the stages of the algorithm is performed once, allowing the summation of the computational costs. The cost of thresholding a volume is $\mathrm{O}(n)$ for a volume with n voxels, since thresholding visits each voxel in the volume once. The cost of the Hough transform is dictated by the number and resolution of the free parameters in the transform space and the number of values to be transformed. For this algorithm the Hough transform has 2 free parameters θ, ϕ and 1 dependent parameter ρ. Assuming that the free parameters are quantized the same into m levels, the cost of the Hough transform has a computational cost of $\mathrm{O}(m^2)$ for each value transformed. The total cost of the Hough transform is $\mathrm{O}(n \times m^2)$. The non-maximal suppression operation examines the neighborhood of each voxel once, leading to a computational cost of $\mathrm{O}(m^3)$ where the m is the number of levels in the quantizations of the parameters in the accumulator. The Delaunay triangulation has incremental

$O\left(n^{\lceil \frac{d}{2} \rceil}\right)$ algorithms where there are n input points in d dimensions [40]. Dijkstra's algorithm with a trivial implementation is $O(V^2)$ where V is the number of vertices in the graph. A simple modification of the algorithm allows for $O(E \lg V)$ running time where E is the number of edges in the graph. Since the path generation algorithm operates by constructing a Delaunay triangulation of the candidate points the number of edges E in the candidate path graph will be relatively small. Spline generation is an $O(n)$ algorithm where n is the number of points being interpolated.

6.5 Experimental Results

To test the view path generation algorithm, two data sets were tested. The first data set was the jack phantom, while the second was the base of the bee stinger. Each data set is a cube 128 units on a side with a real-valued interest level at each voxel. The first processing step was to compute the Hough transform for interest points in the volumes. For the stinger data set, voxels whose interest level was greater than or equal to 4 were used. For the jack data set, voxels whose interest was greater than 0 were used. For both data sets, each parameter in the Hough transform space was quantized into 128 levels.

The accumulator volumes were then passed through a non-maximal suppression filter to select local maxima in the accumulator array. By removing non-maximal points from the accumulator the candidate planes with the most local interest are preserved. After the suppression step, the accumulator indices for the non-zero planes were extracted. For both data sets approximately ten thousand cutting planes were extracted from the accumulator.

A Delaunay triangulation was then constructed for the candidate planes, generating a connected graph. The vertices in the graph represent the parameters for the plane, while the edges indicate acceptable plane to plane transitions. Dijkstra's single source shortest path algorithm was then applied to the graph, starting at the first candidate plane extracted from the accumulator. All paths that covered at least 20 candidate planes were then extracted.

Figure 6.3 shows all of the extracted paths for the stinger volume. The stinger volume produced 776 different candidate paths, with the longest path covering 33 cutting planes. Figure 6.4 shows the extracted paths from the jack volume. The jack volume produced 2490 different candidate paths, with the longest path covering 87 cutting planes. After the path tree was extracted, the candidate plane normals were decoded and plane centers were generated from the set of voxels contributing to the plane.

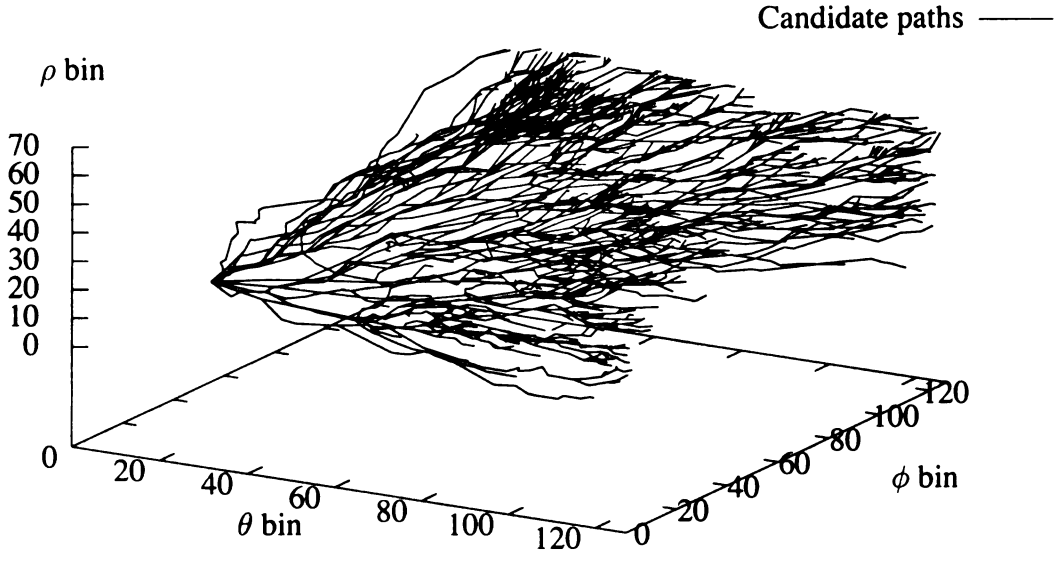


Figure 6.3: Candidate paths for stinger data set. All paths are generated from a single point.

Figure 6.5 is subset of 40 of the rendered cross-sections of the interest labeled version of the stinger data set. The path that specifies the cross-sections was automatically generated. The rendering starts by cutting the stinger length-wise, then rotates the cutting plane through the part of the stinger that connects to the bee. After the base of the stinger has been covered, the cutting plane is then reoriented such that the plane normal is parallel to the long axis of the stinger. The plane then moves along the length of the stinger. Figure 6.6 follows the same path shown in Figure 6.5, but the rendering is done on the original

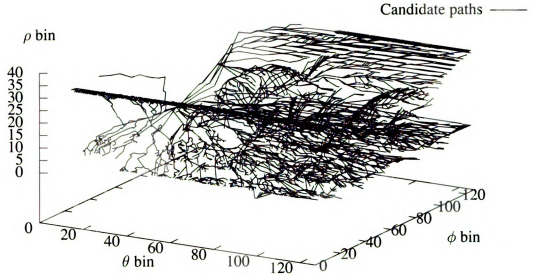


Figure 6.4: Candidate paths for jack data set. All paths are generated from a single point.

reconstruction data rather than the interest classified data.

Figures 6.7 and 6.9 are renderings of the interest labeled jack phantom. Two paths were chosen arbitrarily from the set of generated paths. In Figure 6.7 the cutting plane is initially centered on the high interest structure at the center of the phantom. In the third and fourth rows, the cutting plane has rotated to cover the small spherical caps on the short limbs of the jack. The remaining rows have the cutting plane rotating around the center region of the jack. The corresponding raw volume is rendered in Figure 6.8.

Figure 6.9 again begins by rotating about the center of the jack. The rotation about the center continues until the fifth row where the cutting plane then rotates to cover the interesting regions on the ends of two limbs. The last two rows have the cutting plane rotating and aligning itself with one of the volume axes passing through one of the annular interest regions on a jack limb ending with a large sphere. The corresponding raw volume is rendering in Figure 6.10.

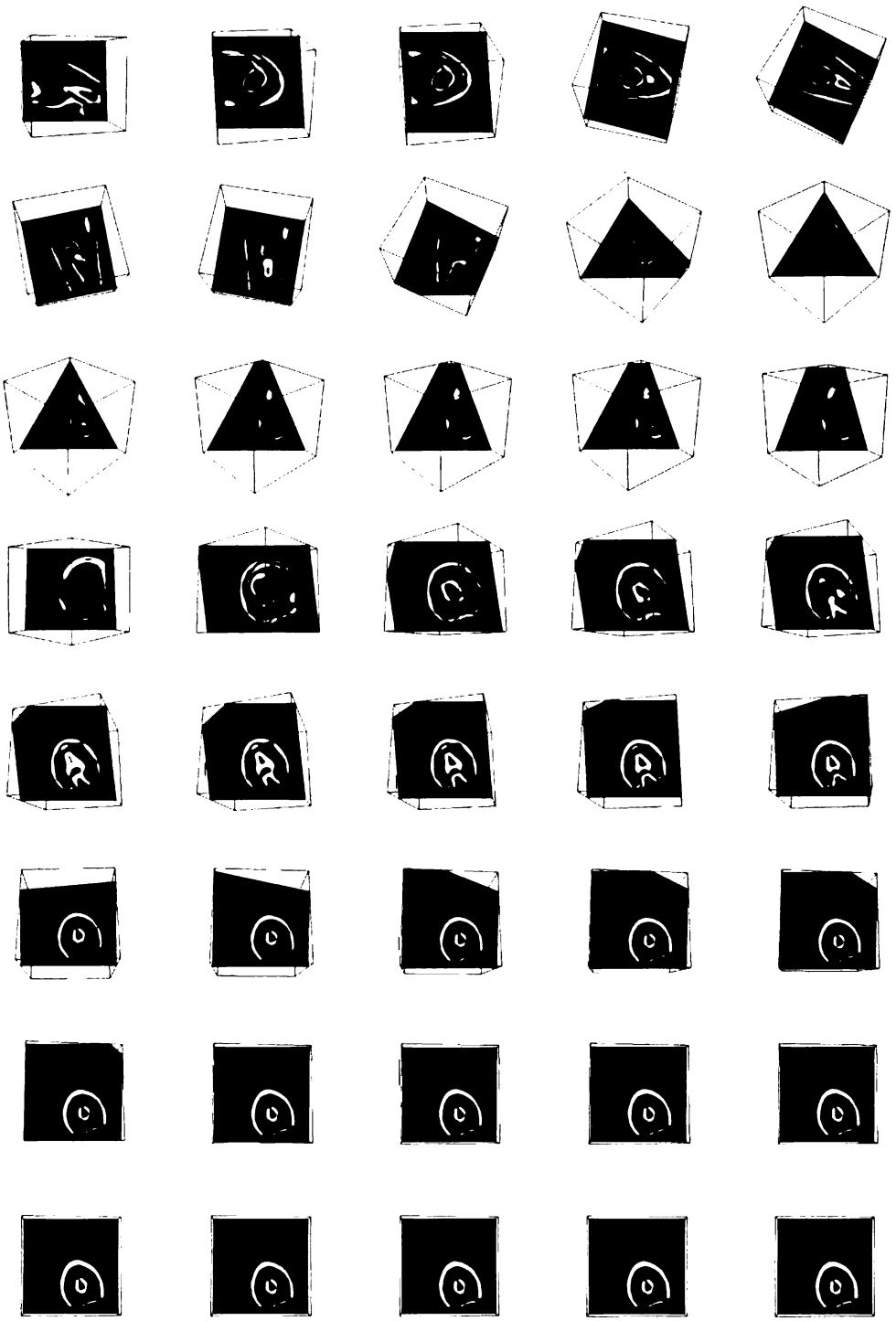


Figure 6.5: Rendering of interest classified stinger volume along an automatically generated path. The frames go from left to right, and top to bottom. Every 20th frame from the animation is shown. The full video is 58 seconds and should be available on the WWW at `http://www.cse.msu.edu/~albeepau/`.

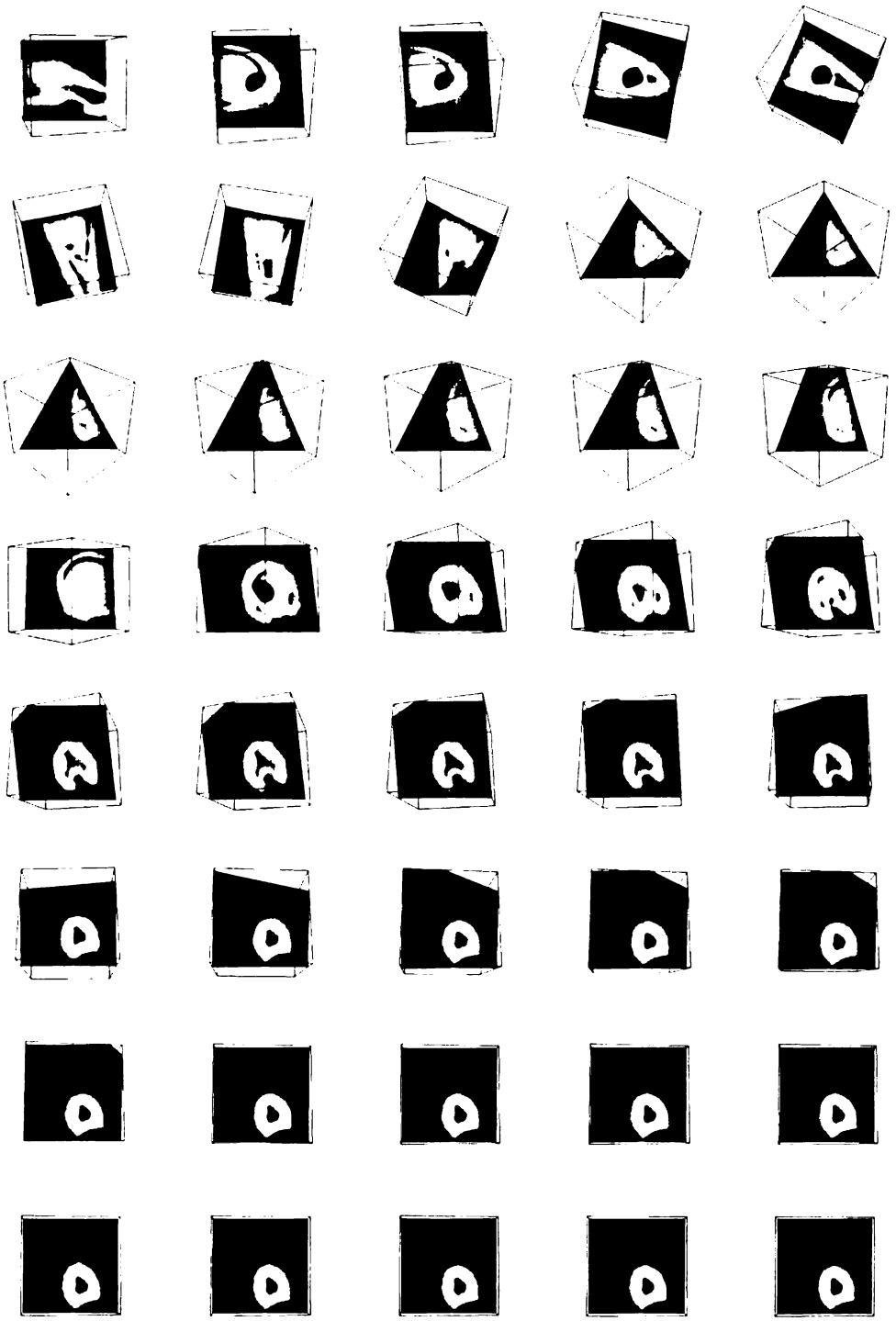


Figure 6.6: Rendering of original stinger volume along an automatically generated path. The frames go from left to right, and top to bottom. Every 20th frame from the animation is shown. The full video is 58 seconds and should be available on the WWW at `http://www.cse.msu.edu/~albeepau/`.

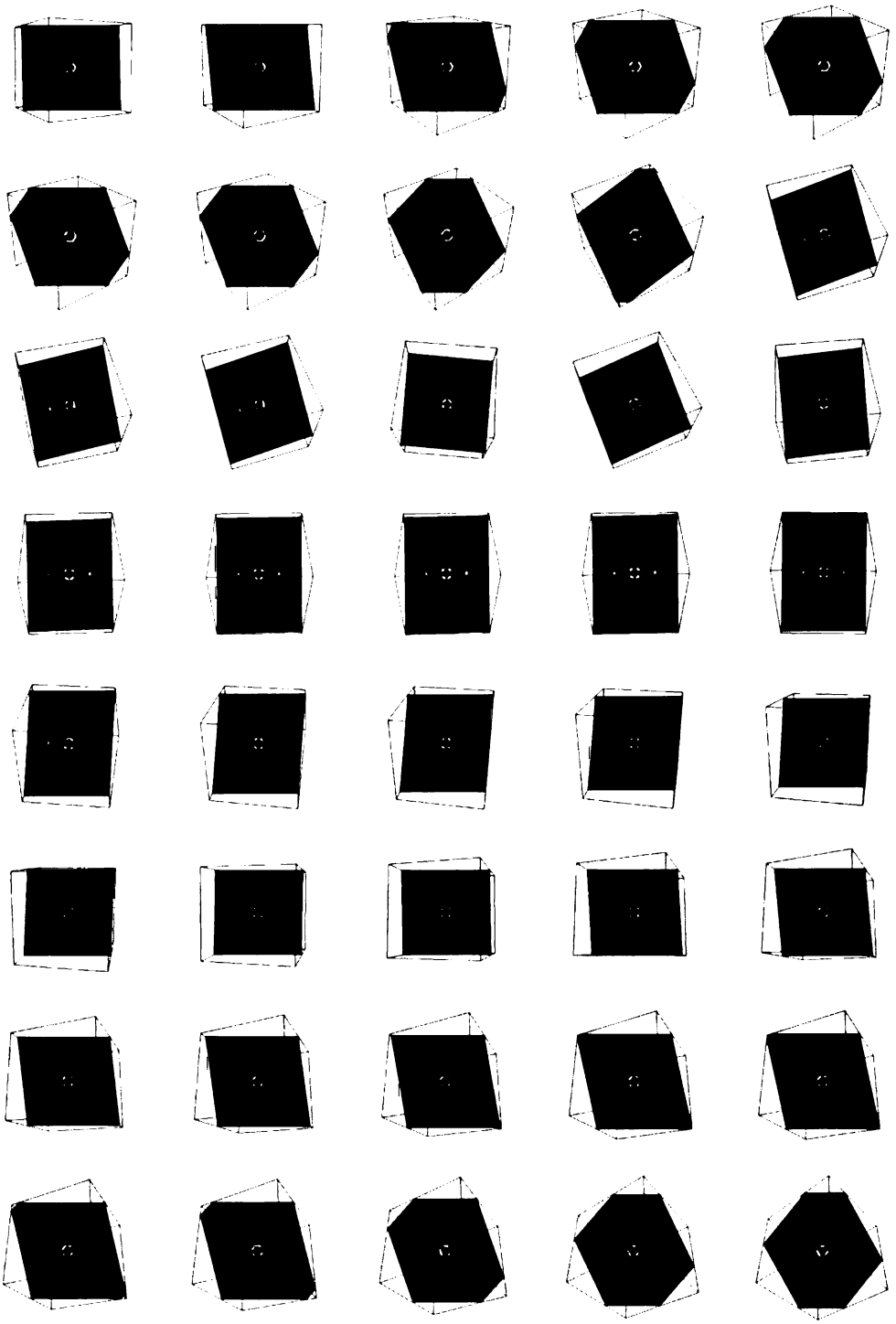


Figure 6.7: Rendering of interest classified jack volume along an automatically generated path. The frames go from left to right, and top to bottom. Every 20th frame from the animation is shown. The full video is 58 seconds and should be available on the WWW at `http://www.cse.msu.edu/~albeepau/`.

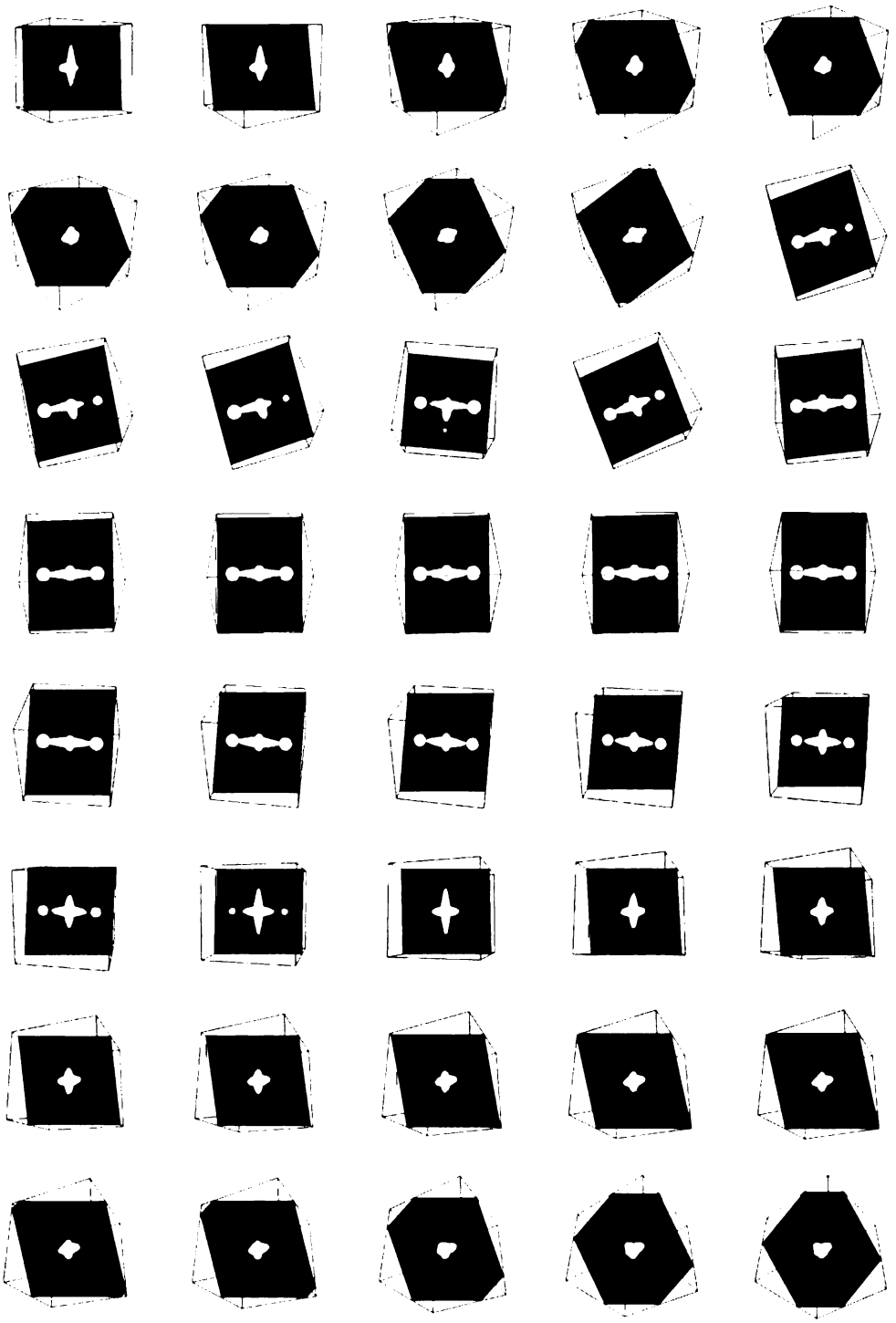


Figure 6.8: Rendering of raw jack volume along an automatically generated path. The frames go from left to right, and top to bottom. Every 20th frame from the animation is shown. The full video is 58 seconds and should be available on the WWW at `http://www.cse.msu.edu/~albeepau/`.

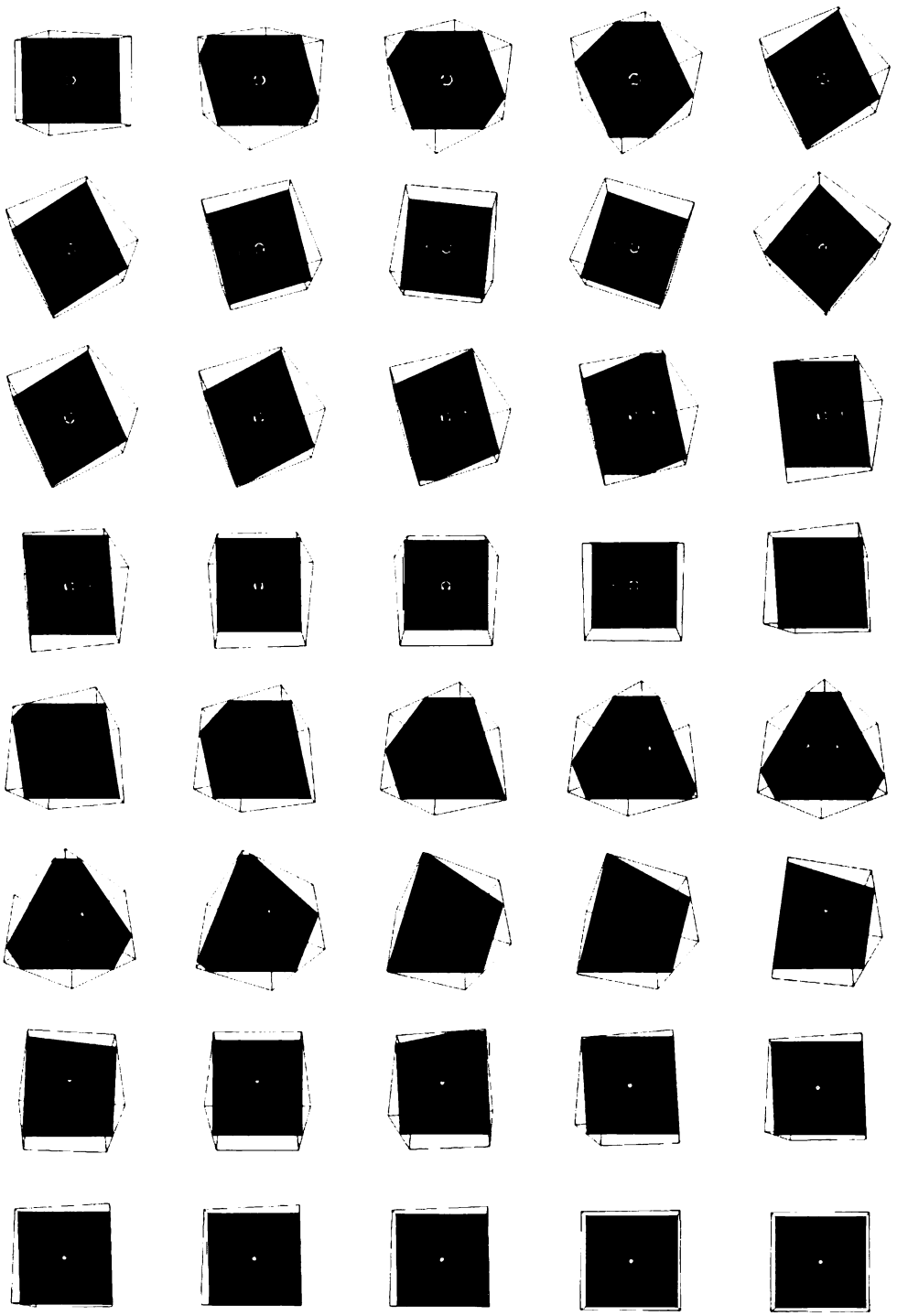


Figure 6.9: Rendering of interest classified jack volume along an automatically generated path. The frames go from left to right, and top to bottom. Every 20th frame from the animation is shown. The full video is 58 seconds and should be available on the WWW at `http://www.cse.msu.edu/~albeepau/`.

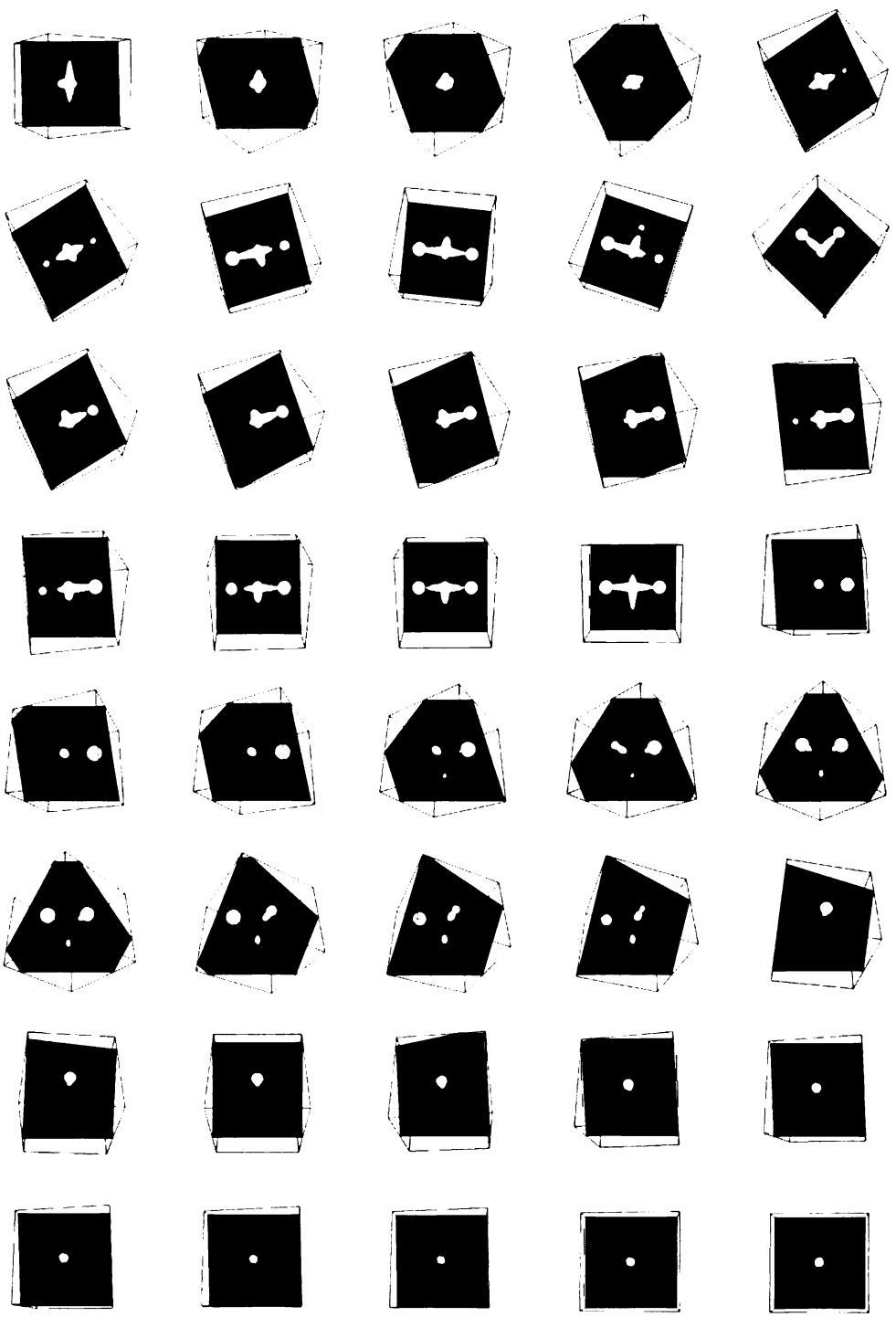


Figure 6.10: Rendering of raw jack volume along an automatically generated path. The frames go from left to right, and top to bottom. Every 20th frame from the animation is shown. The full video is 58 seconds and should be available on the WWW at `http://www.cse.msu.edu/~albeepau/`.

The two paths generated for the jack phantom cover the center of the volume, and then diverge to cover different portions of the volume. The difference in coverage is apparent in Figures 6.8 and 6.10.

6.5.1 Discussion

The animations produced from the automatically generated paths are reasonable. In the case of the stinger animation (Figure 6.5) the view path covers the interest points in the volume very well. The path is particularly satisfying since it visits the large pore at the bee end of the stinger and then flies through the tube along the length of the stinger. The animations for the jack, Figures 6.7–6.10, provide good coverage of the jack but are not as satisfying. The stinger is densely covered with interesting points while the jack phantom has the interesting points located in a few relatively small regions.

The automatic path generation algorithm generates a wide range of paths very quickly. The path collection generation time for the test data sets is approximately two minutes. User input would be appropriate to select the path(s) to be rendered, since rendering the test volume takes 23 seconds per frame. Giving the user the ability to suggest that certain planes be considered or avoided could be addressed by adjusting the Hough accumulator.

The quality of the paths is still to be determined, since the algorithm enforces smoothness constraints on the path graph generation in terms of plane specification, but does not enforce any smoothness constraints on the path center determination. These experiments point to a need to develop a technique to measure the quality of a path through the volume in terms the smoothness of the path location. An additional measure of path quality would be determining how well the path covers the volume, and the level of redundancy in the coverage.

The graph generation algorithm relies on computing the Delaunay triangulation. While the Delaunay triangulation helps to enforce plane parameter smoothness constraints, it may introduce an unacceptable level of cost when considering a large number of potential inter-

est points. One solution to enforcing overall smoothness constraints would be to add three more parameters to the Hough transform computing the centroid of the points contributing to the hypothesized plane. Delaunay triangulation becomes prohibitively expensive for data whose dimensionality exceeds 3.

6.6 Conclusions

The simple path generation algorithm presented in this chapter was successful in generating acceptable view paths through a volume. The overall algorithm is relatively simple and computationally reasonable for the parameterization used in the Hough transform. The primary bottle neck in constructing a collection of rendered view paths is the time required to re-slice and render a volume. With the application of a parallel renderer, the time to generate a collection of rendered paths should be reasonable.

Study of the overall algorithm reveals several areas where additional work would be fruitful. Among these are improved plane hypothesizing via a more complex plane parameterization or some other plane fitter. Construction of the view path graph can be enhanced to facilitate enforcing final path smoothness constraints. The view path graph may be eliminated completely through the use of a different plane selection system. Human studies to assess the quality of the 3D structure presentation and path quality would provide guidance on what constraints are relevant for path generation. Finally, user interaction and intervention should be facilitated. Techniques using active contours may be appropriate for tracking paths through the accumulator that maximize overall path interest.

Chapter 7

Conclusions and Future Work

The overall goal of this thesis was to develop a suite of new algorithms and tools to support analysis and visualization of volumetric data sets. Early research was strongly oriented towards microtomographic volumes of soil. As the the research progressed, the emphasis shifted to volumes in general. Volumetric data sets from medicine, soil science, geology, and material science share some attributes making general analysis algorithms desirable. The data sets are generally large, requiring a significant amount of effort for manual analysis. The data acquisition processes are subject to a wide range of noise processes. The structure of the data sets is often unknown, reducing the utility of analysis techniques that require *a priori* knowledge of the structures. In order to address these commonalities, three major subproblems were pursued in this dissertation. The first subproblem was the n-ary segmentation problem for volumetric data sets. The second subproblem was detecting user-specified interest regions in volumetric data sets. The third subproblem was generating virtual tours of a volume that provide users with insight into the structure of the volume.

The n-ary segmentation problem was approached by developing a clustering based algorithm that is efficient and provides n-ary segmented volumes with the degree of segmentation controlled by one parameter. The interest detection problem was addressed through developing a novel transform that encodes structural information in a rotation- and

translation- invariant manner. The results of the transform were then used in a supervised learning system to develop an interest classifier that is both domain specific and extendible. View path generation was approached through a combination of transform space encoding using the Hough transform [52], computational geometry, and graph theory. A more detailed review of each of the three contributions follows.

7.1 Segmentation

The clustering-based segmentation algorithm proposed in Chapter 4 satisfies the four criteria established for an acceptable algorithm, which were that the algorithm be computationally simple, be robust in the presence of noise, provide n-ary segmentation, and support user-driven segmentation. In addition to satisfying the basic criteria, the algorithm is amenable to the application of other cluster similarity criteria. Finally, the algorithm is domain independent in the sense that the only *a priori* knowledge required for segmentation is that separable density distributions are present in the volume. The clustering algorithm can generate a set of weighted Gaussians that closely matches the shape of the histogram of real data, with the ability to trade-off the fidelity of the histogram match with execution speed. The algorithm was validated on both synthetic and real data sets using both quantitative measures and qualitative comparisons. The algorithm has the advantages of being fast to compute and producing monotonic segmentation results.

The segmentation algorithm's execution time is dependent on the size of the volume, the α parameter controlling cluster merging, and the actual distribution of the elements in the volume. The time to process a volume in excess of 637 megabytes was less than 9 minutes on a Sun Ultra 10. The worst case performance for the algorithm is $\mathrm{O}\left(nm^2\right)$, where n is the number of voxels being processed, and m is the number of clusters. The expected order is $\mathrm{O}\left(nm\right)$ since the m^2 term only arises when all detected clusters are merged into one cluster. In the expected case, the algorithm is efficient and provides good performance for

large data sets.

The segmentation results produced by the algorithm is an integer relabeling of the volume. The relabeling is monotonic with respect to the original data. This monotonicity property allows the data to be interpreted in a manner similar to that of the original data. The labeling has the added advantage of reducing noise within the volume and enhancing boundaries between regions of dissimilar material. The various displays and downstream use of the segmentation data lend support to its value.

Future directions for the segmentation algorithm include operating on multi-channel data, automatic setting of the α parameter, improving the cluster merging criteria, and parallelizing the algorithm. Currently the algorithm operates only on scalar data. Some imaging modalities (optical thin-sections, confocal microscopy, and MRI) produce a vector at each image point. Extending the clustering algorithm to handle multi-channel data would take advantage of the richer statistics available from such data. The α parameter is currently set by the user at the beginning of clustering. The Bhattacharrya distance [26], or some other goodness-of-fit measure, could be used to automatically set the α parameter. Automating parameter setting would allow the algorithm to be used more readily by non-experts. Closely related to the α parameter is the merging criterion. Distances such as the Mahalanobis distance [54] may be more appropriate for setting the α parameter driving cluster merging, particularly if the segmentation algorithm is extended to multi-channel data sets. The segmentation algorithm is currently a serial algorithm. Improvements in performance may be possible if the algorithm is parallelized.

7.2 Interest Detection

Interest detection is the process of locating points or regions within the volume that are salient with respect to the user's needs. Deriving a single criterion for interest is ineffective since interest is a domain and application specific concept. The approach taken for inter-

est detection was two-fold. The first part was developing the normalized discrete Radial Mass Transform (RMT) and the second was to perform supervised learning using a Support Vector Machine (SVM) to classify data transformed using the RMT as interesting or uninteresting.

The RMT is a novel feature that encodes information about the structure of a volume in a rotation- and translation-invariant manner. The RMT was first defined as a continuous function that integrated the mass on the surface of a sphere centered at a point. The RMT was then normalized by the surface area of the sphere defining the area of integration. Finally, the RMT was discretized to support computation using a uniformly sampled volume. The continuous RMT exhibits rotational and translational invariance. The discretized RMT is not rotationally invariant but is close to being so. The discretization effects that cause some rotational variance can be eliminated with a modified RMT at the cost of significant compute time using a weighted computation. The discrete RMT is translation invariant for discrete steps.

The richness of the RMT was also shown via a set of second order features computed from the RMT. The second order features demonstrated that the RMT can be used to detect a variety of different structures within a volume. The second order features were not studied further, but will be of interest in future work.

The computation of the RMT requires $\mathrm{O}\left(r^3 n\right)$ effort, where r is the maximum radius of the RMT and n is the number of positions where the RMT is computed. While the RMT is expensive to compute sequentially for an entire volume, it is trivial to compute in parallel. Additionally, the cost of computing the RMT can be amortized over several different classification cycles. The information encoded in the RMT is sufficient to allow for the generation of meaningful second-order features about the volumes. These features can, in future work, be used in addition to the raw RMT for performing classification. Additionally, the RMT shows promise for performing volumetric registration.

Using the SVM along with the RMT allows for user-specified interest detection without

explicit encoding of *a priori* knowledge of interesting features. The framework presented in Chapter 5 is amenable to the addition of other features to allow for a more robust interest classification system. While this framework is not suitable for fully automatic analysis, it should provide assistance to the human expert in evaluating volumetric data.

The RMT is useful since it generates a a concise description of the regions for which it is computed. This description has been demonstrated to provide a rich basis for further feature generation, as well as serving as a good feature for interest classification. The RMT coupled with the SVM satisfies the criteria identified in Chapter 5. The RMT is simple to compute and when computed in parallel allows for a speedup proportional to the number of compute nodes available. The RMT does not require the data to fall in any particular ranges for computation, making it adaptable to different data types. The RMT is easily stored, allowing it to be used for multiple applications. The SVM, using the RMT as a feature, can be iteratively trained to support incremental refinement of interest classifications. Finally, the interest framework is able to accommodate other features in addition to the RMT for interest classification.

The RMT may facilitate registration by providing a mechanism for selecting discriminating regions within a volume. Discriminating regions may be identifiable by selecting regions where the RMT has a jagged response. By selecting the RMTs with the highest jaggedness measure, a constellation of points can be identified for use in other registration algorithms [85, 86]. The RMT may also be extended to facilitate multi-channel data analysis, allowing for better characterization of data sets.

7.3 View Path Generation

The path generation algorithm presented in Chapter 6 was successful in generating acceptable view paths through a volume that highlighted the structures in regions of dense interest. The overall algorithm is conceptually simple and computationally reasonable for

the parameterization used in the Hough transform. The computational effort for path generation depends on the number of interest points used to build the accumulator n_{int}, the quantization of the accumulator m, the number of points extracted from the accumulator n_{acc} and the number of edges n_{edges} generated during the path construction. The total computational effort is $O(n_{int}m^3 + n_{acc}^2 + n_{edges} \lg n_{acc})$. In general, n_{acc} will be 1 to 3 orders of magnitude less than n_{int}, leading to an expected computational cost of $O(n_{int}m^3)$. The primary bottleneck in constructing a collection of rendered view paths is the time required to re-slice and render a volume. Approximately 20 seconds are required to render each slice in a 128^3 unit volume. With the application of a parallel renderer, the time to generate a collection of rendered paths should be reasonable.

The view path generation algorithm satisfies the criteria identified in Chapter 6. It automatically generates a set of paths derived from a set of interest maximizing cutting planes. The user can select one or more of the paths for rendering, with future work including developing techniques to support direct user intervention in path generation.

The overall algorithm admits several opportunities where additional work would be fruitful. Among these are improved plane hypothesizing via a more complex plane parameterization, or some other plane fitter. Construction of the view path graph can be enhanced to facilitate enforcing final path smoothness constraints. The view path graph may be eliminated completely through the use of a different plane selection system. Finally, user interaction and intervention should be facilitated. Techniques using active contours may be appropriate for tracking paths through the accumulator that maximize overall path interest.

7.4 Parallel Computing

A common thread through each of the subproblems in the dissertation was the application of parallel computing techniques. Typical data sets are on the order of 400 – 600 megabytes, leading to serial execution times measured in hours or days. The parallel versions of the

presented algorithms reduced the execution times to minutes or hours. There is further work to be done in parallelizing the entire suite of analysis tools.

7.5 Overall Applicability

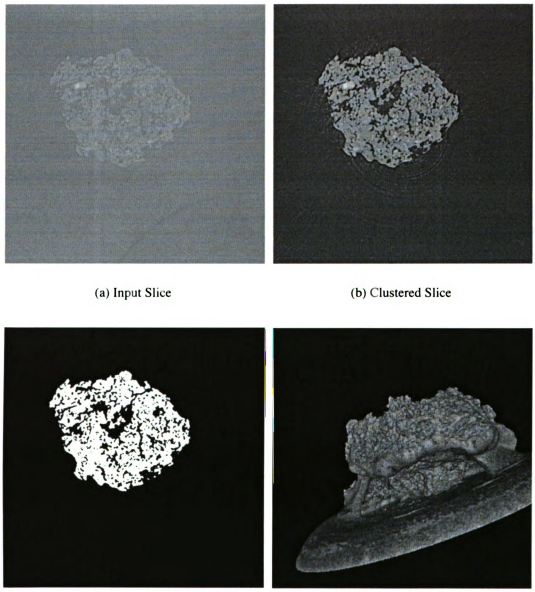
We envision different types of usage of the new tools. First, is direct support for the scientist sampling the material. The scientist would need substantial computing resources to analyze and visualize material in one work day. Current computational resources can be utilized in parallel to support exploratory data analysis using the techniques that have been developed. The second type of user is the scientist on the Internet viewing the products of the algorithms in this thesis. A scientist may use these tools to provide representative samples to colleagues who lack access to the materials, or the means to directly work with materials being studied. The algorithms in this thesis may also provide means for developing presentations to explain the material being studied to students, funding agencies, and the general public.

APPENDICES

Appendix A

Segmented Volumes

This appendix contains additional segmentation output produced by the method in Chapter 4.



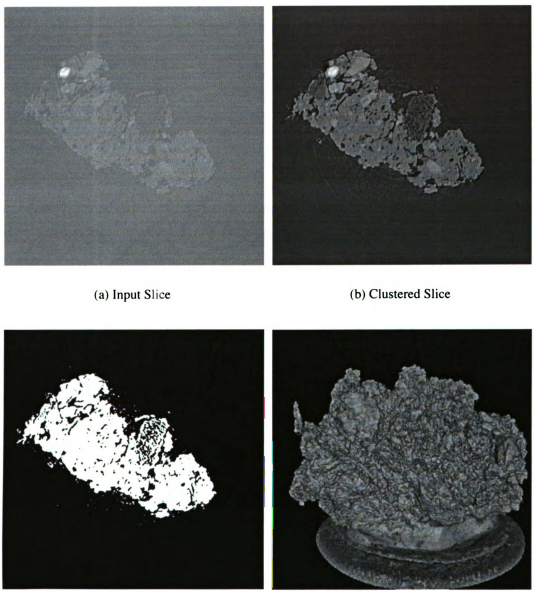
(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure A.1: 5atah3_a segmentation results



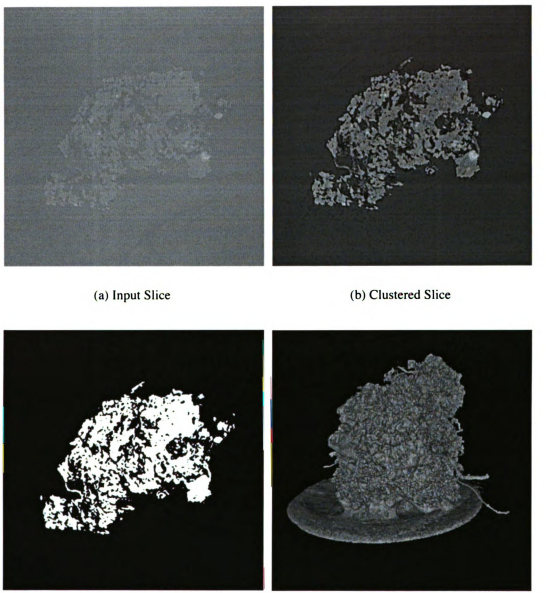
(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure A.2: 5ath1_a segmentation results



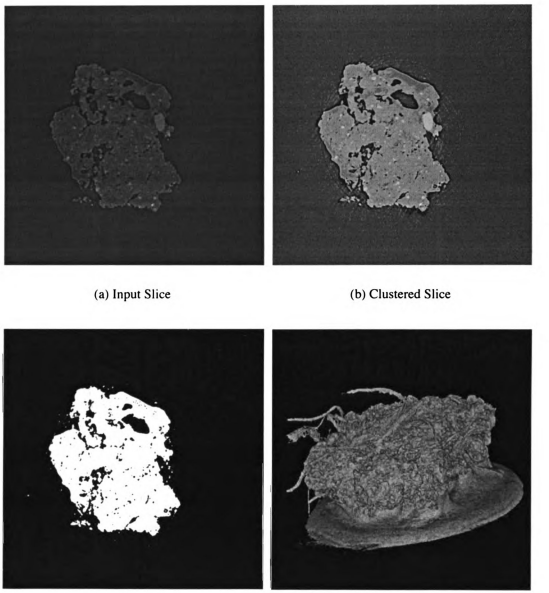
(a) Input Slice

(b) Clustered Slice

(c) Segmented Slice

(d) Volume Rendering

Figure A.3: 5ath2_a segmentation results



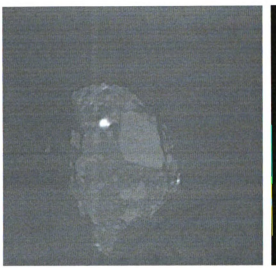
(a) Input Slice

(b) Clustered Slice

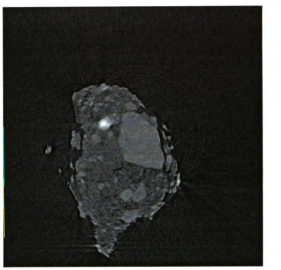
(c) Segmented Slice

(d) Volume Rendering

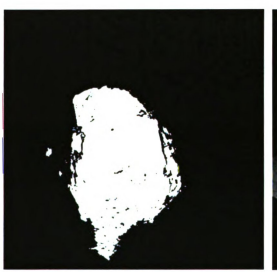
Figure A.4: 5ath4_a segmentation results



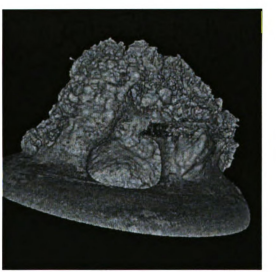
(a) Input Slice



(b) Clustered Slice

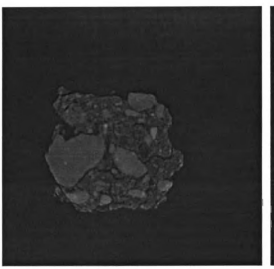


(c) Segmented Slice

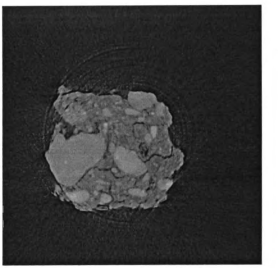


(d) Volume Rendering

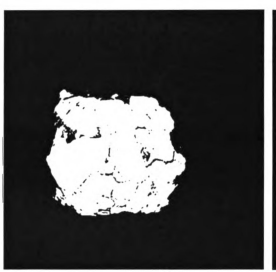
Figure A.5: 5btah2_a segmentation results



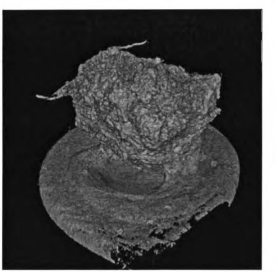
(a) Input Slice



(b) Clustered Slice



(c) Segmented Slice



(d) Volume Rendering

Figure A.6: 5btah4_a segmentation results

Appendix B

RMT Figures

This appendix contains additional results from the methods of Chapter 5.

- **100 radial gradients**. 100 lines are placed in the volume, and the Euclidian Distance Map (EDM) is generated.

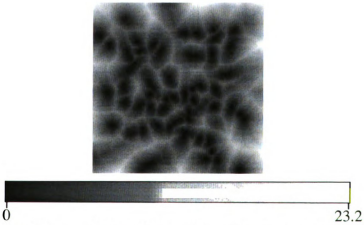


Figure B.1: multipleRadialGradients 100 phantom slice 64.

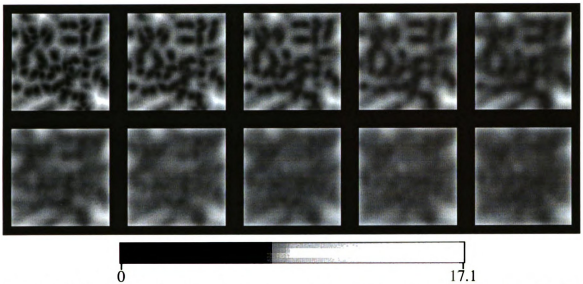


Figure B.2: Discrete RMT of length 10 for multipleRadialGradients 100 phantom slice 64.

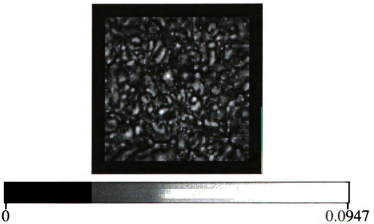


Figure B.3: Jaggedness feature multipleRadialGradients 100 phantom slice 64.

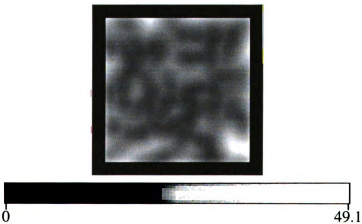


Figure B.4: SSR feature multipleRadialGradients 100 phantom slice 64.

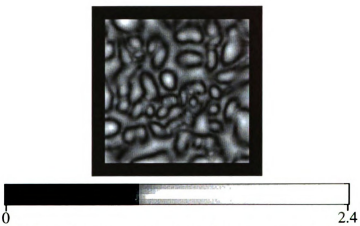


Figure B.5: SSIRD feature multipleRadialGradients 100 phantom slice 64.

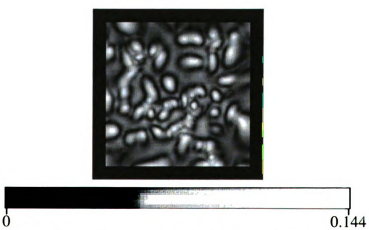


Figure B.6: SSSIRD feature multipleRadialGradients 100 phantom slice 64.

- **Single point gradient**. A single point is placed in the center of the volume, and the EDM is generated.

Figure B.7: singlePointGradient phantom slice 64.

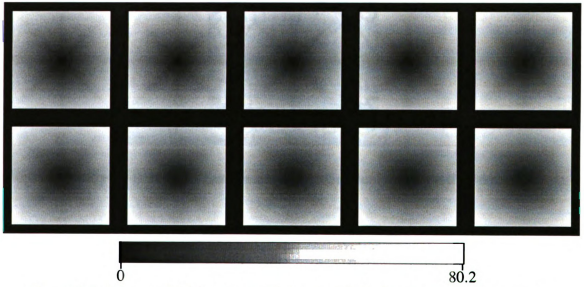


Figure B.8: Discrete RMT of length 10 for singlePointGradient phantom slice 64.

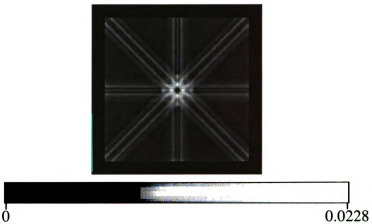


Figure B.9: Jaggedness feature singlePointGradient phantom slice 64.

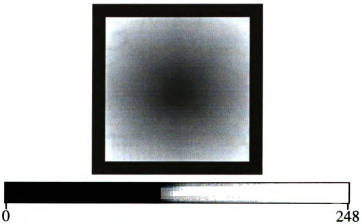


Figure B.10: SSR feature singlePointGradient phantom slice 64.

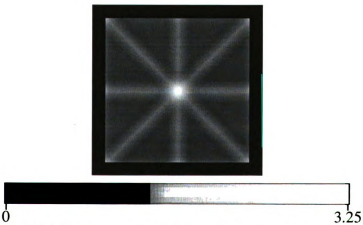


Figure B.11: SSIRD feature singlePointGradient phantom slice 64.

Figure B.12: SSSIRD feature singlePointGradient phantom slice 64.

- **Point gradients**. The point gradients are generated by randomly placing points in the volume, and generating a EDM.

  When the RMT is centered on a point gradient, it will have a ramp structure, and will be non-smooth as it moves away from the centers.

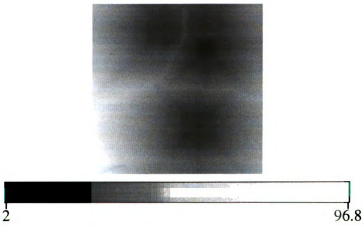


Figure B.13: pointGradients phantom slice 64.

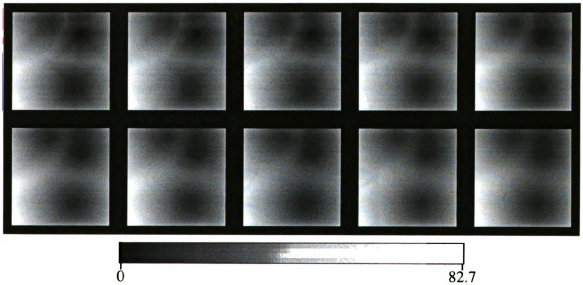


Figure B.14: Discrete RMT of length 10 for pointGradients phantom slice 64.

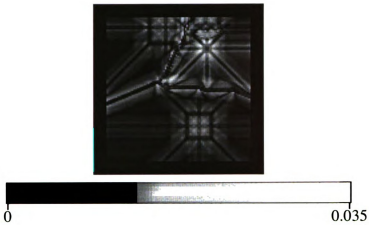


Figure B.15: Jaggedness feature pointGradients phantom slice 64.

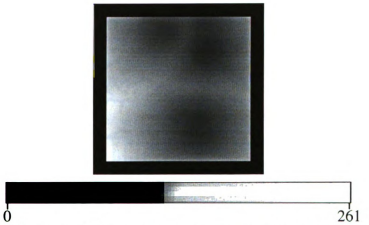


Figure B.16: SSR feature pointGradients phantom slice 64.

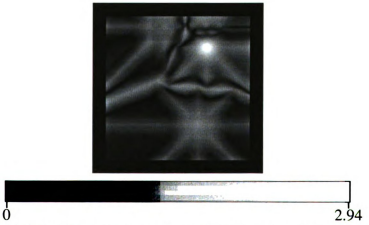


Figure B.17: SSIRD feature pointGradients phantom slice 64.

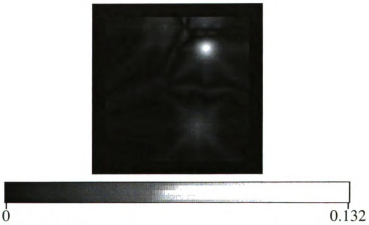


Figure B.18: SSSIRD feature pointGradients phantom slice 64.

- **Single sphere**. A single point gradient is thresholded.

Figure B.19: singleSphere phantom slice 64.

Figure B.20: Discrete RMT of length 10 for singleSphere phantom slice 64.

Figure B.21: Jaggedness feature singleSphere phantom slice 64.

Figure B.22: SSR feature singleSphere phantom slice 64.

Figure B.23: SSIRD feature singleSphere phantom slice 64.

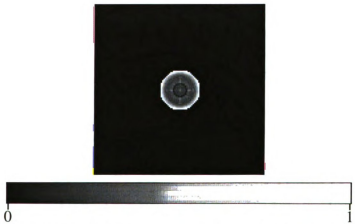


Figure B.24: SSSIRD feature singleSphere phantom slice 64.

- **Single tube**. A single radial gradient is thresholded.

Figure B.25: singleTube phantom slice 64.

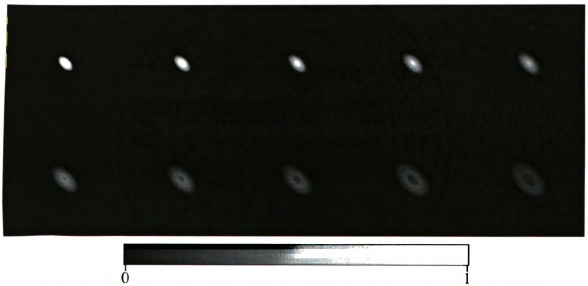


Figure B.26: Discrete RMT of length 10 for singleTube phantom slice 64.

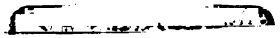


Figure B.27: Jaggedness feature singleTube phantom slice 64.

Figure B.28: SSR feature singleTube phantom slice 64.

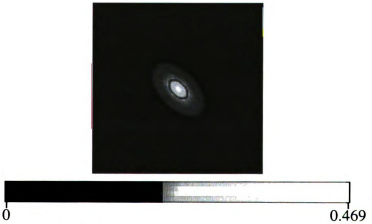


Figure B.29: SSIRD feature singleTube phantom slice 64.

Figure B.30: SSSIRD feature singleTube phantom slice 64.

- **Spheres**. A multiple point gradient is thresholded. As the RMT is computed with it's center with ρ units of the sphere, it will produce a response proportional to the degree of overlap between the sphere and the shells that define the RMT.

Figure B.31: spheres phantom slice 64.

Figure B.32: Discrete RMT of length 10 for spheres phantom slice 64.

Figure B.33: Jaggedness feature spheres phantom slice 64.

Figure B.34: SSR feature spheres phantom slice 64.

Figure B.35: SSIRD feature spheres phantom slice 64.

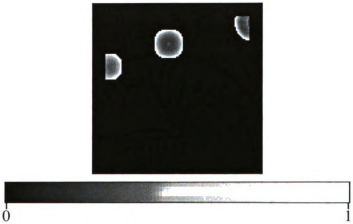


Figure B.36: SSSIRD feature spheres phantom slice 64.

Figure B.37: spheres 100 phantom slice 64.

Figure B.38: Discrete RMT of length 10 for spheres 100 phantom slice 64.

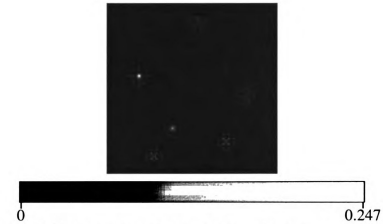


Figure B.39: Jaggedness feature spheres 100 phantom slice 64.

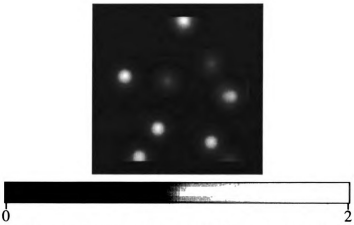


Figure B.40: SSR feature spheres 100 phantom slice 64.

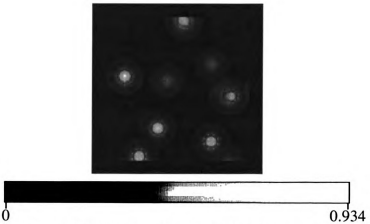


Figure B.41: SSIRD feature spheres 100 phantom slice 64.

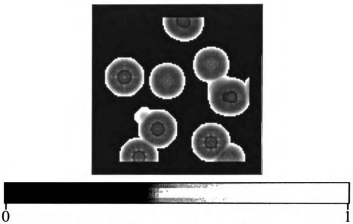


Figure B.42: SSSIRD feature spheres 100 phantom slice 64.

BIBLIOGRAPHY

Bibliography

[1] Amir A Amini, Saeid Tehrani, and Terry E Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *ICCV88*, pages 95–99, 1988.

[2] Shigeru Ando. Consistant gradient operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 252–265, 2000.

[3] Chandrajit Bajaj, Insung Ihm, and Joe Warren. Higher-order interpolation and least-squares approximation using implicit algebraic surfaces. *ACM Transactions on Graphics*, 12(4):327–347, 1993.

[4] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[5] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1982.

[6] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[7] Peter Bartlett and John Shawe-Taylor. *Advances in Kernel Methods : Support Vector Learning*, chapter 4, pages 43–54. MIT Press, Cambridge, Mass., 1999.

[8] F. Bookstein. *Morphometric Tools for Landmark Data.* Cambridge University Press, 1991.

[9] Paul Borrel and Ari Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, 1994.

[10] Berhnard E. Boser, Isabelle M. Guyon, and Vladimr N. Vapnik. A training algorithm for optimal margin classfiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[11] Martin L. Brady, Kenneth K. Jung, H. T. Nguyen, and Thinh PQ Nguyen. Interactive volume navigation. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):243–256, 1998. ISSN 1077-2626.

[12] Peter I. Brooker. *Geostatistical Primer.* World Scientific Publishing Co. Pte. Ltd., 1991.

[13] Sue Brown, Jeff Campbell, Sherri Griffin, Dick James, and Ray Haythornwaite. Failure mechanisms detected in memory chips during routine construction analysis. In *1999 IEEE International Workshop on Memory Technology, Design, and Testing*, 1999.

[14] J. Canny. Finding edges and lines in images. In *MIT AI-TR*, 1983.

[15] Laurent Cohen, Eric Bardinet, and Nicholas Ayache. Surface reconstruction using active contour models. Technical Report 1824, INRIA, 1993.

[16] Chuck Collins and Kenneth Stephenson. A circle packing algorithm. *Computational Geometry : Theory and Applications*, 25:233–256, 2003.

[17] R. Collobert, S. Benigo, and J. Mariéthoz. Torch: A modular machine learning software library. Technical Report IDIAP-RR 02-46, Dalle Molle Institute for Perceptual Artificial Intelligence, 2003.

[18] T. F. Cootes and C. J. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester, Wolfson Image Analysis Unit, Imaging Science and Biomedical Engineering, Manchester M13 9PT, United Kingdom, 1999.

[19] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[20] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, 1993.

[21] G R Davis. Analytical algorithm for the generation of polygonal projection data for tomographic reconstruction. *Nuclear Instruments and Methods in Physics Research Section A*, 382:548–552, 1996.

[22] G R Davis and J C Elliott. X-ray microtomography scanner using time-delay integration for elimination of ring artefacts in the reconstructed image. *Nuclear Instruments and Methods in Physics Research Section A*, 394:157–162, 1997.

[23] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*, pages 145–162. Springer-Verlag, Berlin, 1997.

[24] Gert Van de Wouwer, Paul Scheunders, Stefan Livens, and Dirk Van Dyck. Wavelet correlation signatures for color texture classification. *Pattern Recognition*, 32(3):443–451, 1999.

[25] Etienne Decencière, Chantal de Fouquet, and Fernand Meyer. Applications of kriging to image sequence coding. *Signal Processing: Image Communication*, 13(3):227–249, 1998.

[26] A. Djouadi, Ö. Snorrason, and F. D. Garber. The quality of training sample estimates of the bhattacharyya coefficient. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):92–97, 1990.

[27] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley-Interscience, 2 edition, 2000.

[28] Nicolae Duta. *Learning-Based Detection, Segmentation and Matching of Objects.* PhD thesis, Michigan State University, East Lansing, MI 48824, 2000.

[29] Richard E.Williamson and Hale F. Trotter. *Multivariate Mathematics.* Prentice Hall, 1996.

[30] J. Fayolle, C. Ducottet, L. Riou, and S. Coudert. A wavelet based multiscale detection scheme of feature points. In *ICPR00*, pages Vol III: 425–428, 2000.

[31] J. Fayolle, L. Riou, and C. Ducottet. Robustness of a multiscale scheme of feature points detection. *PR*, 33(9):1437–1453, September 2000.

[32] James D. Foley, Andries Van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 2nd edition, 1990.

[33] David Forsey and Richard H. Bartels. Surface fitting with hierarchical splines. *ACM Transactions on Graphics*, 14(2):134–161, 1995.

[34] Wolfgang Förstner. A feature based correspondence algorithm for image matching. In *ISPRS86*, pages III: 150–166. International Society for Photogrammetry and Remote Sensing, 1986.

[35] Carl Kesselman Ian Foster and Steven Tuecke Gene Tsudick. A security architecture for computational grids. In *Proc. of the 5th ACM Conference on Computer and Communication Security.* ACM Press, 1998.

[36] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[37] Werner Frei and Chung-Ching Chen. Fast boundry detection: A generalization and a new algorithm. *IEEE Transactions on Computers*, C-26(10):988–998, 1977.

[38] Ajeetkumar Gaddipatti, Raghu Machiraju, and Roni Yagel. Steering image generation with wavelet based perceptual metric. In *Eurographics '97*, 1997.

[39] Vito Di Gesú, Cesare Valenti, and Laurent Strinati. Local operators to detect regions of interest. *Pattern Recognition Letters*, 18(11–13):1077–1081, November 1997.

[40] Jacob E. Goodman and Joseph O'Rourke. *Handbook of discrete and computational geometry*. CRC Press, Inc., 1997.

[41] Amara Graps. An introduction to wavelets. *IEEE Computational Science and Engineering, Summer 1995*, 2(2), 1995.

[42] Steve R. Gunn. Support vector machines for classification and regression. Technical report, University of Southampton Faculty of Engineering and Applied Science Department of Electronics and Computer Science, 1998.

[43] Steve R Gunn and Mark S Nixon. Robust snake implementation: A dual active contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):63–68, 1997.

[44] C. Harris and M.J. Stephens. A combined corner and edge detector. In *Alvey88*, pages 147–152, 1988.

[45] Hans-Christian Hege, Tobias Höllerer, and Detlev Stalling. Volume rendering: Mathematical models and algorithmic aspects. Technical Report TR-93-07, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 1994.

[46] Gabor T. Herman. *Image Reconstruction From Projections*. Computer Science and Applied Mathematics. Academic Press, 1980.

[47] A Hilton, A Stoddart, J Illingworth, and T Windeatt. Building 3D graphical models of complex objects. In H Jones, R Raby, and D Vicars, editors, *Proceedings of 14th Annual Conference of Eurographics UK Chapter*, volume 1, pages 193–204, Imperial College, 1996.

[48] A Hilton, A Stoddart, J Illingworth, and T Windeatt. Marching triangles: Range image fusion for complex object modelling. In *Proc of IEEE International Conference on Image Processing*, volume 2, pages 381–384, Laussane, 1996.

[49] Thomas Hofmann, Jan Puzicha, and Joachim Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[50] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 295–302. ACM SIGGRAPH, ACM Press, 1994. ISBN 0-89791-667-0.

[51] Berthold K. P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671 – 1686, December 1984.

[52] P. Hough. Method and means for recognizing complex patterns. US Patent 3,069,654, 1962.

[53] Peter A Howarth and P J Costello. Studies into the visual effects of immersion in virtual environments, 1997. `http://www.lboro.ac.uk/departments/hu/groups/viserg/`.

[54] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[55] John W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[56] A. C. Kak and B. A. Roberts. Reconstruction from projections: Applications in computerized tomography. In Tzay Y. Young and King-Sun Fu, editors, *Handbook of Pattern Recognition and Image Processing.* Academic Press, 1986.

[57] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[58] Les Kitchen and Azriel Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, 1(2):95–102, December 1982.

[59] Ulrich Köthe. Edge and junction detection with an improved structure tensor. In *DAMG03*, pages 25–32, 2003.

[60] Yongbum Lee, Takeshi Hara, Hiroshi Fujita, Shigeki Itoh, and Takeo Ishigaki. Nodule detection on chest helical CT scans by using a genetic algorithm. In *Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems (IIS '97) Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems (IIS '97)*, 1997.

[61] M.E. Leventon, W.E.L. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *CVPR00*, pages I:316–323,, 2000.

[62] Lars Lippert. *Wavelet-based Volume Rendering.* PhD thesis, Swiss Federal Institute of Technology Zürich, 1998.

[63] James J. Little. Extended gaussian images, mixed volumes, shape reconstruction. In *Proceedings of the first annual symposium on Computational Geometry*, pages 15–23. ACM Press, 1985.

[64] Jiming Liu and Yuan Y. Tang. Adaptive image segmentation with distributed behavior-based agents. *PAMI*, 21(6):544–551, 1999.

[65] J.M. Loomis and D.W. Eby. Perceiving structure from motion: Failure of shape constancy. *ICCV*, pages 383–391, 1988.

[66] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, 1987.

[67] Bradley Lowekamp, Penny Rheingans, and Terry S. Yoo. Exploring surface characteristics with interactive gaussian images: a case study. In *Proceedings of the conference on Visualization '02*, pages 553–556. IEEE Computer Society, 2002.

[68] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[69] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA81*, pages 121–130, 1981.

[70] Alan P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEETOVCG*, 5(4):308–321, 1999. ISSN 1077-2626.

[71] D.J. Marchette, J.L. Solka, R. Guidry, and J. Green. The advanced distributed region of interest tool. *PR*, 31(12):2103–2118, December 1998.

[72] T. McInemey and D. Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Transactions on Medical Imaging*, 18(10):840–850, October 1999.

[73] Vasileios Megalooikonomou, Haimonti Dutta, and Despina Kontos. Fast and effective characterization of 3D region data. In *ICIP 2002*, pages I: 421–424, 2002.

[74] David Meyers, Shelley Skinner, and Kenneth Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(2):228–258, 1992.

[75] Norman B. Nill and Brian H. Bouzas. Objective image quality measure derived from digital image power spectra. *Optical Engineering*, 31(4):813–825, 1992.

[76] Norman B. Nill and Brian H. Bouzas. Poster: Objective image quality measure derived from digital image power spectra, 1992. `http://www.mitre.org/tech/mtf/poster.pdf`.

[77] Wonho Oh and W. Brent Lindquist. Image thresholding by indicator kriging. *PAMI*, 21(7):590–602, 1999.

[78] C.M. Privitera, M. Azzariti, and L.W. Stark. Locating regions-of-interest for the mars rover expedition. *JRS*, 21(17):3327–3347, November 2000.

[79] C.M. Privitera and L.W. Stark. Evaluating image processing algorithms that predict regions of interest. *PRL*, 19(11):1037–1043, September 1998.

[80] C.M. Privitera and L.W. Stark. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *PAMI*, 22(9):970–982, September 2000.

[81] Ari Rappoport, Yaacov Hel-Or, and Michael Werman. Interactive design of smooth objects with probabilistic point constraints. *ACM Transactions on Graphics*, 13(2):156–176, 1994.

[82] D. Reisfeld, H. Wolfson, and Y. Yeshurun. Context-free attentional operators: The generalized symmetry transform. *IJCV*, 14(2):119–130, March 1995.

[83] Mark Rivers. GSECARS tomography processing software. `http://cars9.uchicago.edu/software/idl/tomography.html`.

[84] Karl Rohr. On 3d differential operators for detecting point landmarks. *Image and Vision Computing*, 3(15):219–233, 1997.

[85] Karl Rohr. *Landmark-Based Image Analysis*, volume 21 of *Computational Imaging and Vision*. Kluwer Academic Publishers, Dordrecht, February 2001.

[86] Karl Rohr. CVPR tutorial slides: Feature extraction, 2004. http://www.i-u.de/schools/rohr/slides/feature_extraction_CVPR04_rohr.pdf.

[87] John C Russ. *The Image Processing Handbook*. CRC Press and IEEE Press, third edition, 1999.

[88] John A. Saghri, Patrick S. Cheatham, and Ali Habibi. Image quality measure based on a human visual system model. *Optical Engineering*, 28(7):813–818, 1989.

[89] Hanan Samat. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, 1990.

[90] S. Schaller, J.E. Wildberger, R. Raupach, M. Niethammer, K. Klingenbeck-Regn, and T. Flohr. Spatial domain filtering for fast modification of the tradeoff between image sharpness and pixel noise in computed tomography. *MedImg*, 22(7):846–853, July 2003.

[91] Paul Scheunders, Stefan Livens, Gert Van de Wouwer, P. Vautrot, and Dirk Van Dyck. Wavelet-based texture analysis. *International Journal on Computer Science and Information Management*, 1(2):22–34, 1998.

[92] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. In *ICCV98*, pages 230–235, 1998.

[93] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *IJCV*, 37(2):151–172, June 2000.

[94] Will Schroeder, Ken Martin, and Bill Lorenson. *The Visualization Toolkit*. Prentice Hall, 2 edition, 1992.

[95] Nicu Sebe and Michael S. Lew. Comparing salient point detectors. *Pattern Recognition Letters*, 24:89–96, 2003.

[96] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2 edition, 1999.

[97] Ali Shahrokni. CVonline: On-line compendium of computer vision; Bhattacharyya distance. http://homepages.inf.ed.ac.uk/rbf/CVonline/.

[98] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[99] L. Shepp and B. Logan. The Fourier reconstruction of a head section,. *IEEE Transaction on Nuclear Science*, 21(1):21–43, 1974.

[100] A. Sirjani and G. R. Cross. On representation of a shape's skeleton. *Pattern Recognition Letters*, 12:149–154, 1991.

[101] Kalpathi R. Subramanian and Bruce F. Naylor. Converting discrete images to patitioning treees. *IEEETOVCG*, 3(3):273–288, 1997.

[102] Changming Sun and Jamie Sherrah. 3d symmetry detection using the extended Gaussian image. *PAMI*, 19(2):164–169, 1997.

[103] Steven L. Tanimoto. *The Elements of Artificial Intelligence Using Common Lisp*. Computer Science Press, 1995.

[104] Allen van Gelder and Jane Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.

[105] P. Vautrot, Gert Van de Wouwer, Paul Scheunders, Stefan Livens, Dirk Van Dyck, and Noël Bonnet. Rotation-invariant texture segmentation using continuous wavelets. In *Proceedings 2nd IEEE Symposium on applications of time-frequency and time-scale methods*, Coventry, UK, 1997.

[106] I. Viola, A. Kanitsar, and M. E. Gröller. Hardware-based nonlinear filtering and segmentation using high-level shading languages. In *Proceedings of IEEE Visualization '03*, pages 309–316, 2003.

[107] Gregor von Laszewski, Mei-Hui Su, Joseph A. Insley, Ian Foster, John Bresnahan, Carl Kesselman, Marcus Thiebaux, Mark L Rivers amd Steve Wang amd Brian Tieman, and Ian McNulty. Real-time analysis, visualization, and steering of microtomography experiments at photon sources. In *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[108] Hans Wackernagel. *Multivariate Geostatistics*. Springer-Verlag Berlin Heidelberg New York, 1995. isbn:3-540-60127-9.

[109] Rüdiger Westermann and Thomas Ertl. A multiscale approach to integrated volume segmentation and rendering. *Computer Graphics Forum*, 16(3):C117–C127, 1997.

[110] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.

[111] Andrew Witkin, Kurt Fleischer, and Alan H Barr. Energy constraints on parameterized models. In *Proceedings of SIGGRAPH 1987*, pages 225–232. ACM SIGGRAPH, 1987.

[112] Chenyang Xu and Jerry L Prince. Gradient vector flow: A new external force for snakes. In *CVPR97*, pages 66–71, 1997.

[113] Chenyang Xu and Jerry L Prince. Gradient vector flow: A new external force for snakes. *IEEE Transactions on Image Processing*, pages 359–369, 1998.

[114] Zhanjun Yue, Ardeshir Goshtasby, and Laurens V Ackerman. Automatic detection of rib borders in chest radiographs. *IEEE Trans. Medical Imaging*, 14(3):525–536, 1995.

[115] Yong Zhou and Arthur W Toga. Efficient skeletonization of volumetric objects. *PAMI*, 5(3):196–209, 1999.

[116] Barbara Zitová, Jaroslav Kautsky, Gabriele Peters, and Jan Flusser. Robust detection of significant points in multiframe images. *PRL*, 20(2):199–206, February 1999.

[117] Oscar A. Zuniga and Robert M. Haralick. Gradient threshold selection using the facet model. *Pattern Recognition*, 21(5):493–503, 1988.