SEMI-AUTOMATIC TRANSFER FUNCTION GENERATION FOR VOLUMETRIC DATA VISUALIZATION USING CONTOUR TREE ANALYSES

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in the School of Information Technologies at The University of Sydney

JIANLONG ZHOU

January 2011

THE UNIVERSITY OF SYDNEY

Abstract

Volume visualization is now an essential tool for volumetric data analysis. Topology based techniques have begun to emerge as a general framework as scientific data continue to increase in size and complexity. They are used to capture significant features of the data at an abstract level, enabling and facilitating data understanding in visualization. The contour tree is one of data structures used to store topological abstractions of data sets. This thesis focuses on investigating effective uses of the contour tree in improving efficiency of volumetric data analysis. It achieves the goals by increasing the presentation of topology information in order to explore various relationships of geometrical structures present in a data set. It also utilizes the topology to automate transfer function generations in volume rendering. To make this work practical, this thesis also deals with topology simplification.

Contour tree, one of topological abstractions of the scalar field, is a data structure representing nesting relationships of connected components of contours. Real-world data sets produce unmanageably large contour trees caused by noise. Contour tree simplification removes small-scale topological features while maintaining essential structure of data. The thesis proposes a contour tree simplification approach which uses multiple measures of importance. The proposed approach introduces the concepts of attribute space, importance triangle and importance space into the simplification pipeline. It maximizes the advantages of each measure of importance in the contour tree simplification. This approach allows for the simplification of contour trees by considering the overall attribute space, instead of limited attribute space used by conventional approaches. It also gives a better evaluation of the importance of branches.

Relations between structures in a volume are also of interest to viewers besides shapes. The thesis presents an approach on depicting structural relationships between objects in volume rendering through a concept of structural relationship preserved mapping. The contour tree controlled structural relationship depiction allows users to perceive structural relationships in a more direct way. The advantage of this approach is that it allows analysis of volumetric data to focus on revealing high-level topological relations instead of low-level rendering parameter modulations, and thus improves understanding of volumetric data.

Despite the proliferation of volume rendering and manipulation techniques, the key to comprehensible volume rendering still lies in the design of effective transfer functions. This thesis presents a novel approach for automating transfer function generations in volume rendering. The new approach utilizes topological attributes derived from the contour tree to automate the transfer function generation process. In the new approach, a residue flow model based on Darcy's Law is utilized to control distributions of opacity between branches in the contour tree. Topological attributes are also used to control color selection in a perceptual color space and create harmonic color transfer functions. The transfer functions reveal structural relationships (e.g. inclusion relationship) automatically. They are optimized to maximize opacity and color differences between structures.

Acknowledgements

First of all, I would like to express my deep appreciation to my supervisor, Associate Professor Masahiro Takatsuka. I could not have gone through this degree without his incredible guidance, patience, intelligence and thoughtfulness. Thanks to his rich knowledge and intelligence I always get either a related hint or background for my research challenges from him.

I would like to thank Professor Peter Eades for his guidance over the years.

I would like to thank Professor Zhiyan Wang for his guidance in SCUT, as well as kind support and encouragement during the past years.

I would also like to thank my previous and current colleagues at the ViSLAB: Kelvin Cheng, John Stavrakakis, Choon Jin Ng (CJ), Ren Shao, Balint Seeber, Yi Liu (Eva), Moonyati Yatid, Daniel Barry, Alex Gervesh, Michael Bui, Julien Fruteau, Jacob Munkberg, Charlotte Bradshaw, Matthew Chen and many others. Many thanks are given to them for providing comments on my research ideas at ViSLAB weekly meetings and other various times, or proof-reading my research papers. Thanks to John for his very mathematical brain for always trying to describe a problem with mathematical equations and graphs. Thanks to CJ for giving laughter and entertainment, creating a relaxed and enjoyable working atmosphere at the ViSLAB.

Thanks to Dr. Tony Huang for his valuable discussions and help on my research.

Thanks to Dr. Robert Kong for proof-reading the thesis.

Thanks to the various people at workshop, dp, the administrative and academic staff at the School of Information Technologies (SIT) for helping me throughout the years in various areas during my time at the SIT.

I would like to thank The University of Sydney, National ICT Australia and China Scholarship Council for their financial support for the research.

Lastly, great thanks go to my son and wife, as well as my whole family for their love and constant encouragement throughout.

Contents

Abstract **iii**

Acknowledgements **v**

List of Figures **xiv**

List of Tables **xv**

1 Introduction **1**

1.1 Motivation . . . 3

1.1.1 Topology in Visualization . . . 3

1.1.2 Generations of Rendering Parameters . . . 4

1.1.3 Geometric Relationships in Volume Data . . . 5

1.1.4 Complexities of Topology . . . 6

1.2 Challenges . . . 6

1.2.1 Topology Simplification . . . 6

1.2.2 Topological Relationship Depiction . . . 7

1.2.3 Automation of Transfer Function Generations . . . 8

1.3 Objectives of This Work . . . 10

1.4 Approach and Methodology . . . 11

1.5 Thesis Outline . . . 12

2 Volume Visualization **15**

2.1 Overview . . . 15

2.2 Volume Data . . . 16

2.3 Optical Model for Volume Rendering . . . 17

2.4 Ray-Casting . . . 19

2.5 Transfer Functions . . . 21
2.6 Approaches of Transfer Function Design . . . 23
2.6.1 Early Transfer Function Approaches . . . 24
2.6.2 Multi-Dimensional Transfer Functions . . . 25
2.6.3 Halo Transfer Function and Tone Mapping . . . 27
2.6.4 Semantic Transfer Function Specifications . . . 27
2.6.5 Topology Controlled Transfer Function Specifications . . . 28
2.6.6 Color Transfer Functions . . . 29
2.6.7 Summary of Previous Work . . . 29
2.7 Relationship Depiction . . . 30
2.8 Summary . . . 33

3 Contour Trees **35**
3.1 Preliminaries . . . 35
3.1.1 Sampling and Reconstruction . . . 35
3.1.2 Manifolds . . . 36
3.1.3 Level Sets and Contours . . . 36
3.2 Topological Abstractions . . . 37
3.2.1 Morse-Smale Complexes . . . 37
3.2.2 Reeb Graphs . . . 39
3.2.3 Contour Trees . . . 40
3.3 Setup of Contour Trees . . . 41
3.3.1 Definition of the Contour Tree . . . 41
3.3.2 Previous Work . . . 44
3.3.3 Extraction of Critical Points . . . 46
3.3.4 Computing Contour Trees . . . 46
3.3.5 Multi-Resolution Contour Trees . . . 50
3.4 Existing Contour Tree Simplification Solutions . . . 51
3.5 Contour Tree-Defined Volume Segmentation . . . 53
3.6 Contour Trees in Visualization . . . 55
3.6.1 Topological Representation of Scalar Field . . . 55
3.6.2 Isosurface Extraction and Simplification . . . 55
3.6.3 Depiction of Inclusion Relationship . . . 57
3.6.4 Transfer Function Design in Volume Rendering . . . 59

3.7 Summary 60

4 Multiple Measures of Importance for Contour Tree Simplification 61

4.1 Introduction 62

4.2 A Glance of Topological Simplification 63

4.3 Basics of Contour Tree Simplification 64

4.3.1 Contour Tree Structure 65

4.3.2 Basic Simplification Operations 65

4.4 Concept of Importance 66

4.4.1 Definition of Importance 66

4.4.2 Evaluation of Importance 66

4.5 Importance-Driven Contour Tree Simplification 69

4.5.1 Concept of Attribute Space 70

4.5.2 Concept of Importance Triangle and Importance Space 71

4.5.3 Comparison of *ITri* and Conventional Approaches 74

4.6 Experimental Results and Discussions 77

4.7 Summary 81

5 Structural Relationship Preservation and Depiction in Volume Rendering 83

5.1 Introduction 84

5.2 From Perception to Structural Relationship Preservation 85

5.2.1 Structural Relationship Preserved Mapping 87

5.2.2 Topological Relationships in Data Set 87

5.3 Relationship Depiction in Volume Rendering 89

5.3.1 Depiction of Inclusion Relationship 89

5.3.2 Depiction of Neighboring Relationship 93

5.4 Experimental Results 96

5.5 Discussions 101

5.6 Summary 101

6 Contour Tree Controlled Automatic Transfer Function Generations 103

6.1 Introduction 104

6.2 Framework Overview 106

6.3 Residue Flow Model 106

6.3.1 Darcy's Law 107

6.3.2 Residue Flow Model 108
6.4 Opacity Transfer Function Generations 110
6.4.1 Representation of Inclusion Relationship 110
6.4.2 Residue Flow and Opacity Transfer Function Design 111
6.4.3 Residue Flow from Parent to Children 112
6.4.4 Residue Flow Among Siblings 114
6.5 Color Transfer Function Generations 116
6.5.1 Color Spaces 116
6.5.2 Automation of Color Transfer Functions 118
6.6 Experimental Results and Discussions 120
6.7 Summary 125

7 Qualitative Evaluation **127**
7.1 Methodology 127
7.2 Experimental Results 129
7.2.1 CT Knee Data Set 129
7.2.2 CT Foot Data Set 129
7.2.3 MR Brain Tumor Data Set 132
7.3 Qualitative Comparison 133
7.3.1 Depiction of Structural Relationships 133
7.3.2 Automation of Transfer Function Generations 135
7.3.3 Involvement From Users 136
7.4 Applications 138
7.5 Limitations 139
7.6 Summary 140

8 Conclusions **141**
8.1 Summarized Contributions 141
8.2 Future Research Directions 143

A Priority Line Window **147**

Abbreviations **149**

Bibliography **151**

List of Figures

1.1 Volume rendering of human torso data [7]. 1

1.2 Flow visualization of the intricate topology of a vortex breakdown bubble in a delta wing simulation. High-quality stream surfaces with adaptive resolution are shown in red and green, corresponding to the separation surfaces of two saddle points that determine the boundary of the bubble. This is demonstrated by an additional closed stream surface surrounding the vortex core and integrated from the tip of the wing that wraps the whole structure (Image courtesy of the Scientific Computing and Imaging Institute, University of Utah, USA) [1]. 2

1.3 Add topology into the data space to provide knowledge of structural information of a volume data set. 3

1.4 An example of a 2D mesh and its contour tree. 4

1.5 Relations in a data set defined in [27]. 5

1.6 Leaf pruning and vertex reduction during the contour tree simplification [25]. 7

1.7 Visualizing simulated implosion in laser fusion: (a) The corresponding VST, (b) with topologically-accentuated 1D opacity transfer function, (c) with 2D opacity transfer function depending also on the inclusion level, and (d) with 2D opacity transfer function that visually extracts inner structures. [108, 54]. 9

2.1 Voxels constituting a volumetric object after it has been discretized. . . 17

2.2 An amount of radiant energy emitted at s_0 and $\tilde{s}$ is partially absorbed along the ray [40]. 18

2.3 An example of rendering result using ray-casting [91]. 19

2.4 A ray casts into voxels of a 3D volume data [40]. 20

2.5 A ray is discretized to compute intensity analytically [40]. 20
2.6 Relations between transfer functions and data elements. 22
2.7 A user interface of transfer function specifications [2]. 23
2.8 Relation representation in a graph [27]. 32

3.1 The boundary of a crystal in the Morse-Smale complex consists of lower-dimensional cells: quads, arcs and nodes [49]. 39
3.2 Reeb graph of the height function on the triple torus. The three tunnels of the model are mapped to three loops in the graph [85]. 40
3.3 An example of a 2D mesh and its contour tree. Different level sets are drawn upon the mesh. 41
3.4 Sample data set and its Level sets of as the value of function increases [24]. 42
3.5 Contour tree for Figure 3.4. 43
3.6 Hierarchical decomposition of a contour tree. 51
3.7 Example of a segmentation defined by a contour tree [122]. 54
3.8 Classification of isosurface transitions at saddles depending on embedding in 3D space (C_1 and C_2 are saddle types) [107]. 57
3.9 A new inclusion relationship appears at a saddle shown in (a), and an existing inclusion relationship dissolves at a saddle shown in (b). 58

4.1 Multi-scale analysis of a univariate function. (a) Visualization of the function. (b) Critical points of the function partition the domain into monotonic regions. Pairs of critical points identify features, whose sizes are equal to the difference in function value of the critical points. (c) Small-sized monotonic regions are explicitly identified and removed, leaving behind the "significant" features [48]. 64
4.2 The principle of the contour tree simplification. Beginning from the aim, the CTS process first gets properties of branches to be removed by interpretation. Based on properties of branches, a solution is proposed. The contour tree is then simplified using the presented solution. 69
4.3 The pipeline of importance evaluation in the importance-driven contour tree simplification. 70
4.4 Importance triangle. 72
4.5 The comparison of *ITri* and persistence in the CTS. 75

4.6 A set of threshold curves of I_t with different I_t in a two-dimensional case. 76
4.7 The comparison of priority lines based on different measures of importance for the "fuel" data set. 77
4.8 Rendering and topology of "fuel" data set. The object on the lower left side in each image is the head part view of the object, and the corresponding contour tree is on the right hand side. 78
4.9 Rendering and topology of "TumorHead" data set. The corresponding contour tree of the data set is on the right hand side in each image. . . . 80

5.1 Combine perceptual structure into the volume rendering based data analysis pipeline. 86
5.2 An example of a 2D mesh and its contour tree. The 2D mesh is segmented into subregions and indexed with the contour tree. 90
5.3 Inclusion relationship analysis in the contour tree. 91
5.4 Neighboring relationship analysis in the contour tree. 94
5.5 Harmonic hue wheel templates. A collection of colors that fall into the gray areas is considered to be harmonic. The templates can be rotated by an arbitrary angle. See [31] for the exact definitions. 95
5.6 The neighboring relation color wheel: different color ranges are used to encode neighboring relationship of branches in the contour tree. 96
5.7 Inclusion relationship analysis based on the contour tree: 1) Contour tree, 2) Scalar transfer function, 3) Inclusion transfer function. 97
5.8 Depiction of inclusion relationship of "neghip" data set. 98
5.9 Depiction of neighboring relationship of "fuel" data set. 99
5.10 Depiction of neighboring relationship of "fuel" data set. 100

6.1 Flow chart showing the pipeline for automatic transfer function generations. 107
6.2 Setup of Darcy's experiment in one dimensional case. 107
6.3 Water is poured at the root of the tree and flows throughout the tree. Part of water is absorbed by each branch and residue is delivered to the next depth level. 109
6.4 Various transfer function shapes used in a branch. 115
6.5 RGB color space [4]. 116
6.6 HSV color space [4]. 117

6.7 CIELAB color space [5]. 117
6.8 Volume rendered nucleon data set using harmonic hue T-Type and linear transfer function shape. 121
6.9 Transfer function set generated in Figure 6.8. 122
6.10 Volume rendered fuel data set using full hue range and linear transfer function shape. 123

7.1 Volume rendered knee data set using full hue range and hat-like transfer function shape. 130
7.2 Volume rended foot data set using harmonic hue T-Type and hat-like transfer function shape. 131
7.3 Volume rendered tumor head data set using full hue range and hat-like transfer function shape. 132
7.4 Comparison of volume rendered CT knee data set with: (a) our approach, and (b) VolView 3.2. 133
7.5 Comparison of volume rendered CT foot data set with: (a) our approach, and (b) VolView 3.2. 134
7.6 Comparison of volume rendered MR tumor head data set with: (a) our approach, and (b) VolView 3.2. 134

A.1 The priority line and threshold line. 147

List of Tables

4.1 Comparison of various measures of importance 68

Chapter 1

Introduction

Over the past decades, great advances have been made across all scientific disciplines, including GeoScience, Medical Imaging and Fluid Dynamics. These developments have brought large quantities of volume data into the hands of radiologists and scientists. As humans continue to amass and link an unprecedented wealth of data in widely disparate fields, it is essential to make sense of these complex data through visual inspection.

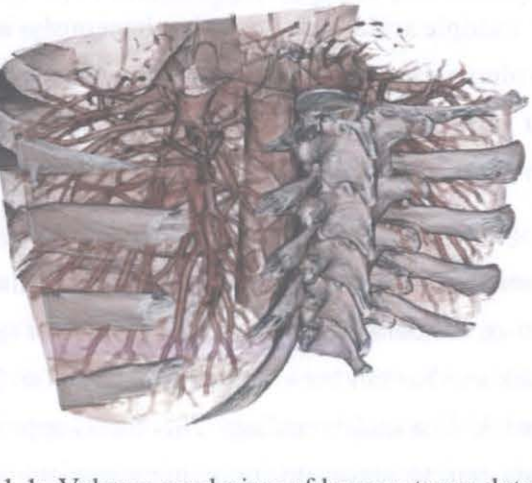


Figure 1.1: Volume rendering of human torso data [7].

Visualization is one of the most important communication and analysis tools available. It enables users to gain insights and process large amounts of data rapidly. Figure 1.1 and Figure 1.2 show two visualization examples. Of all visualization techniques, volume rendering is regarded as a powerful tool to interpret 3D volume data. A volume data generally represents the spatial distribution of a scalar or vector property over some

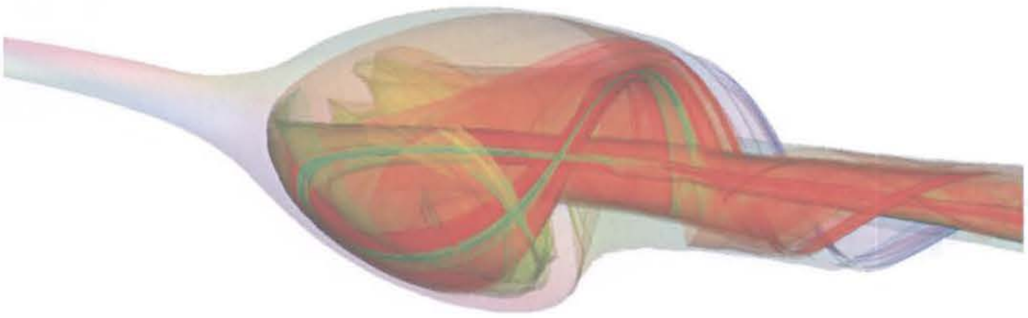


Figure 1.2: Flow visualization of the intricate topology of a vortex breakdown bubble in a delta wing simulation. High-quality stream surfaces with adaptive resolution are shown in red and green, corresponding to the separation surfaces of two saddle points that determine the boundary of the bubble. This is demonstrated by an additional closed stream surface surrounding the vortex core and integrated from the tip of the wing that wraps the whole structure (Image courtesy of the Scientific Computing and Imaging Institute, University of Utah, USA) [1].

d-dimensional space. This thesis does not address the question of visualizing vector data or of visualizing multiple scalar functions simultaneously, it is instead restricted to single scalar-valued volume data. In the context of this thesis, the term of *volume rendering*, such as *direct volume rendering*, refers to the visualization of static 3D scalar fields such as those in 3D medical images.

A variety of different approaches have been developed in the past decades for volume rendering. However, the definition of complex parameter space in volume rendering is a challenging task and has not been well addressed so far. This limits its practical applications in volumetric data understanding. This thesis approaches the issue by utilizing topology of data sets to automate rendering parameter generations in volume rendering. Specifically, the *contour tree*, one of the data structures of topological abstraction, is used in this thesis to store topological information of data sets. This thesis also utilizes the contour tree to explore various relationships of structures in a data set in volume rendering. To make this work practical, it is necessary to deal with contour tree simplifications. The core of the thesis is the use of the contour tree in volume rendering to explore volumetric data efficiently.

1.1 Motivation

A volume data set is a collection of 3D objects in various fields. 3D objects occupy different spatial spaces. A space consists of a set of points (see Figure 1.3). It is not helpful for users to understand a data set if it lacks structural information. A topology is a system of sets that describe the connectivity of the set [134]. The addition of topology into the data space provides knowledge of the structural information in a volume data set (see Figure 1.3). This data space is called topological space. A topological space is a set of points which know who their neighbors are [134].

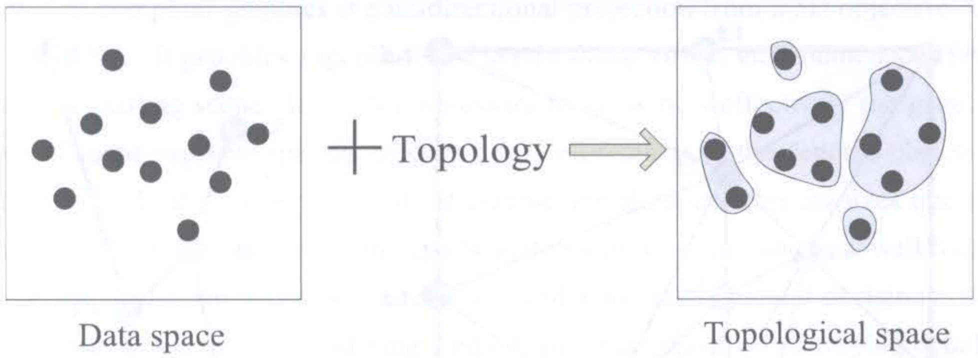


Figure 1.3: Add topology into the data space to provide knowledge of structural information of a volume data set.

1.1.1 Topology in Visualization

In recent years, topology based techniques have begun to emerge as a general framework to extract and analyze features of data sets in visualization. In the recent IEEE VisWeek 2009 [6], a special tutorial session of "Scalar Topology in Visual Data Analysis" was introduced by presenting a systematic overview of current topological methods for the benefit of experienced researchers in visualization [14], in order to widely disseminate topology based techniques in visualization researches. As the tutorial demonstrated, topological techniques can be used to express a wide variety of features in scalar fields based on either gradient- or threshold-based segmentations. Furthermore, topology-based feature definitions are inherently hierarchical providing a framework for noise removal as well as multi-scale analysis. Recent advances in algorithms have produced efficient, easy to implement algorithms to compute and process topological features.

1.1.2 Generations of Rendering Parameters

Volume rendering is becoming an essential tool for volumetric data analysis. Despite the proliferation of volume rendering and manipulation techniques, the key to comprehensible volume rendering still lies in the automation of various rendering parameters, e.g. transfer functions. These parameters are crucial in the understanding of the overall volumetric data and individual features contained within the volume space. The central question is *how to automate rendering parameter generations so that users can understand the underlying data easily and correctly.*

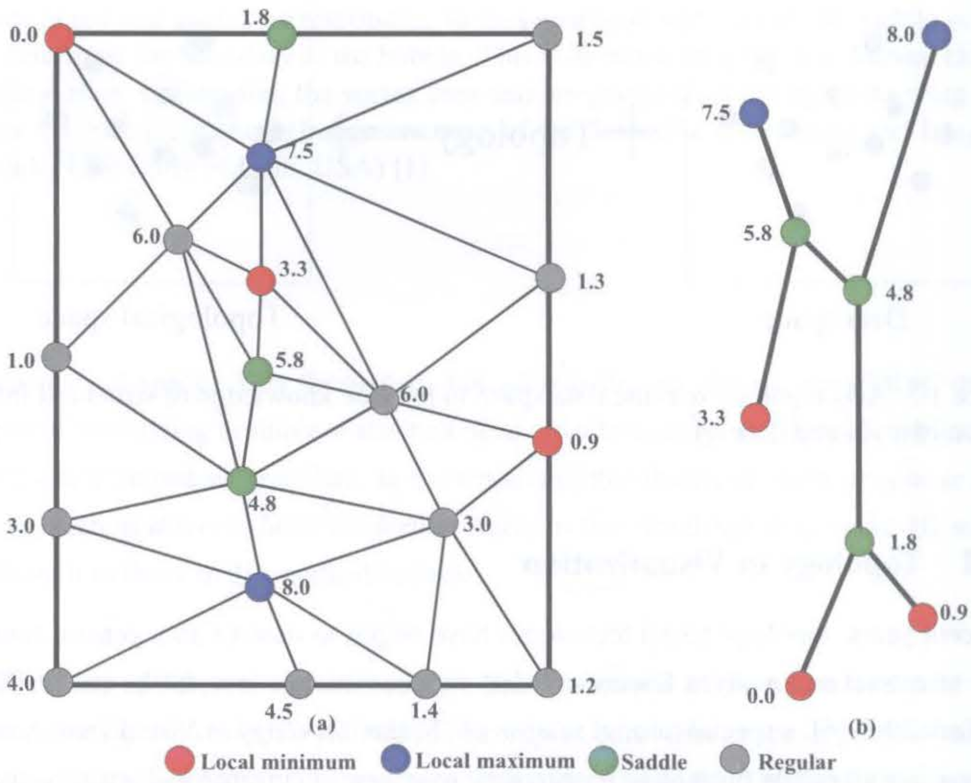


Figure 1.4: An example of a 2D mesh and its contour tree.

The contour tree is a topological abstraction of data sets [22]. It is a graph that tracks contours of level sets as they appear, join, split and disappear based on critical points in data sets. See Chapter 3 for details of definition and setup of the contour tree. Figure 1.4 is an example of a 2D mesh and its corresponding contour tree. As a data structure to record topological events, the contour tree has been used to find important isovalues for transfer function generations [106, 107] and to apply individual transfer functions for various subregions in volume rendering [122]. Unfortunately, the underlying weakness of previous approaches is that *rendering parameters such as transfer functions are still specified manually through user interactions.* More efforts are still necessary for solving how to use the topology to automate rendering parameters. The principal technical challenges for these problems include how to automate rendering parameters such as transfer functions that are used to get a wide spread of dissimilar output renderings.

1.1.3 Geometric Relationships in Volume Data

Traditional computer graphics is a unidirectional projection from a 3D objective scene to a 2D image. It provides capabilities of synthesizing virtual environments or regenerating an existing scene. It is often necessary to know how effectively the generated graphical scene supports specific tasks or how well features of the denoted objects can be discerned. In the context of medical volume visualization, this involves questions of how effectively the visualization depicts anatomical structures and how well features of anatomical structures can be discerned in order to make medical decisions. Most of the conventional volume rendering methods place emphasis on conveying details of the desired features or structures by exposing them clearly to viewers in the results. However, spatial relations between structures in a volume are also of interest to viewers [27]. There exists various kinds of relations between objects in a volume data set. Figure 1.5 shows examples of relations defined in [27]. In medical applications, for example, radiologists are interested in not only shape of structures, but also their neighboring information (e.g., how close they are). Such relation information of structures is crucial for visual analysis and understanding of volumetric data in various applications [27]. These geometrical spatial relationships can be expressed and studied using topology at the abstract level. Hence, it is important to investigate how topology of structural relationships can be computed and utilized in order to improve understanding volumetric data.

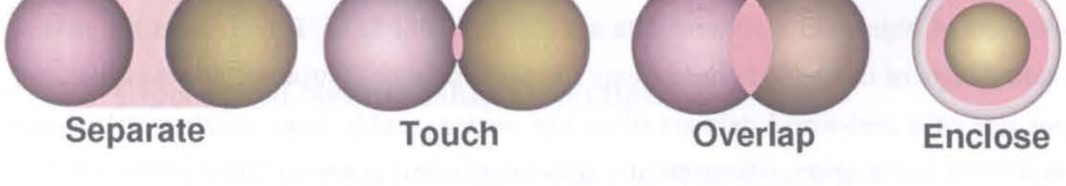


Figure 1.5: Relations in a data set defined in [27].

1.1.4 Complexities of Topology

Because of noise and artifacts in a real-world data set, an experimentally acquired data set often yields irrelevant topological structures. They often distort the true topological features in the contour tree. This makes the contour tree impractical in data analysis and visualization. To make the topology-driven techniques practical in volume rendering, it is also necessary to deal with contour tree simplifications. Despite various approaches of the contour tree simplification being proposed [25, 106, 84], they all did not make full use of information derived from the contour tree. For example, single measure of importance instead of multiple measures of importance is often used to evaluate the importance of branches during the contour tree simplification. Improvements are necessary to produce a better topology simplification.

As such, the topology-driven techniques are ideal approaches to improve understanding of volumetric data sets. These ideal approaches are very ambitious and in the short to medium term, some theories and developments are still necessary. This thesis aims to improve volume rendering and volumetric data analysis by doing further research on these ideal approaches.

1.2 Challenges

Volume rendering is not about "pretty pictures"; its ultimate purpose is comprehension. As scientific data continue to increase in size and complexity, topological tools have been developed to capture significant features of the data at an abstract level enabling and facilitating data understanding by researchers. This section shows challenges of utilizing topology in volume rendering that this thesis focuses on.

1.2.1 Topology Simplification

The contour tree is one of the effective approaches used for depicting topological relationships of objects in a volume data set [24, 22, 108, 122]. It represents the nesting relationships of connected components of isosurfaces or contours. However, the contour tree of a real-world data set often has unmanageably large number of branches because of noise and artifacts from a data acquisition process. This makes the contour tree impractical in data analysis and visualization. A meaningful simplification is necessary for the contour tree. Topology simplification suppresses insignificant features by removing, or canceling, pairs of critical points that are viewed unimportant according to a specified measure. Different approaches are proposed to simplify the contour tree for practical uses in volume data analysis. For example, Carr et al. [25] simplified the contour tree with two basic operations: leaf pruning and node reduction (see Figure 1.6). The scheme involves computation of several metrics that are used for ranking the "importance" of an arc before pruning. The geometric measures used in the contour tree simplification include persistence, volume, and hypervolume. Pascucci et al. [84] used a data structure named branch decomposition for multi-resolution hierarchical representation of the contour tree. The measure of importance used in the branch decomposition is persistence.

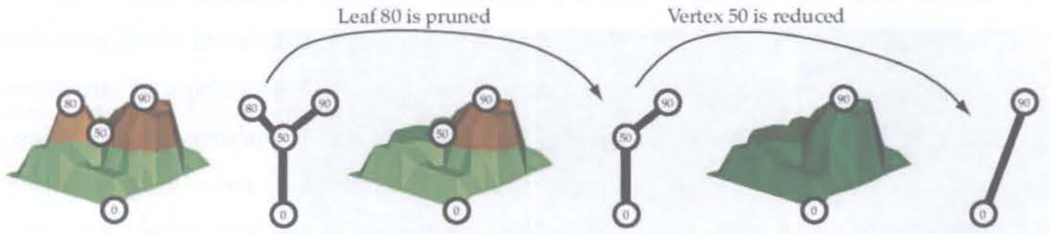


Figure 1.6: Leaf pruning and vertex reduction during the contour tree simplification [25].

However, a single measure of importance cannot depict the importance of branches of the contour tree sufficiently. For example, the measure of persistence is effective to remove random noise based on its small scalar value range, but some large objects of interest having similar scalar value ranges are also removed at the same time in this case. This is also true for measures of volume and hypervolume. A *challenge* for the contour tree simplification is *how to effectively evaluate the importance of branches and make full use of advantages of different importance measures simultaneously.* We strongly believe that *appropriate integration of each measure's advantages will produce a better topology simplification.*

1.2.2 Topological Relationship Depiction

As mentioned, spatial relations between various structures in a volume are also of interest to viewers besides shapes [27]. A volume data set contains a large number of disjointed objects at different target scalar field values. Its field structure is often characterized by spatial configurations of a finite number of feature isosurfaces that segment a volume data set into several components. There are various kinds of relations existing between objects in a volume data set. Such relation information of objects is crucial for visual analysis and understanding of volumetric data in various applications [27]. Chan et al. [27] presented a relation-aware volume exploration pipeline. Typical solutions for inclusion relationship depiction were also explored, such as depth peeling [12, 19], focus+context [121, 69] or focal region based approach [127].

All these approaches try to represent different relationships in a data set based on pure rendering techniques, but they do not touch the basic theory behind the problem — the topology. To meet this end, Takahashi et al. [107, 108] introduced the concept of inclusion level into volume rendering based on Volume Skeleton Tree (VST), to explore the inclusion relationship of structures. Although Takahashi et al.'s method uses a contour tree like data structure to analyze inclusion relationships of structures, *they still consider scalar values of structures globally while applying inclusion level based transfer function globally. Other structural relationships are also necessary to be represented in volume rendering by utilizing topology to improve understanding of data sets.*

1.2.3 Automation of Transfer Function Generations

Despite the proliferation of rendering and manipulation techniques, volume rendering still requires time-consuming interactions for tweaking visualization parameters to obtain comprehensible rendering results. Of all rendering parameters, transfer functions play the key role in comprehensible volume rendering. Transfer functions map scalar values to specific colors and opacities. They assume that scalar values map directly to physical properties such as tissue types. Thus, they are crucial in the understanding of the overall volumetric data and individual features contained within the volume space. Various approaches have been developed to automate or ease specification of transfer functions.

Conventional approaches often use boundary information to automate transfer function specifications [60, 61]. This process is time-consuming and comes short in repeatable results. In addition, volumetric data often contain nested inner structures, i.e., the inclusion relationship — one of topological relationships as mentioned above. This commonly seen inclusion relationship heavily affects the understanding of volumetric data. Conventional transfer functions cannot automatically fully clarify such inner structures. Thus, a systematic and automatic scheme to specify transfer functions while revealing inclusion relationships is highly desirable to greatly facilitate the volume rendering process. To meet this purpose, a typical solution proposed by Takahashi et al. [107, 108] introduces the concept of inclusion level into volume rendering based on VST as shown in Figure 1.7. The inclusion level is also used in transfer function definitions. As mentioned above, they still consider the scalar values of objects globally while applying inclusion level based transfer functions. Because the scalar value of an object is locally but not globally meaningful, it is more reasonable to apply scalar transfer function for each object locally while considering the topology globally.

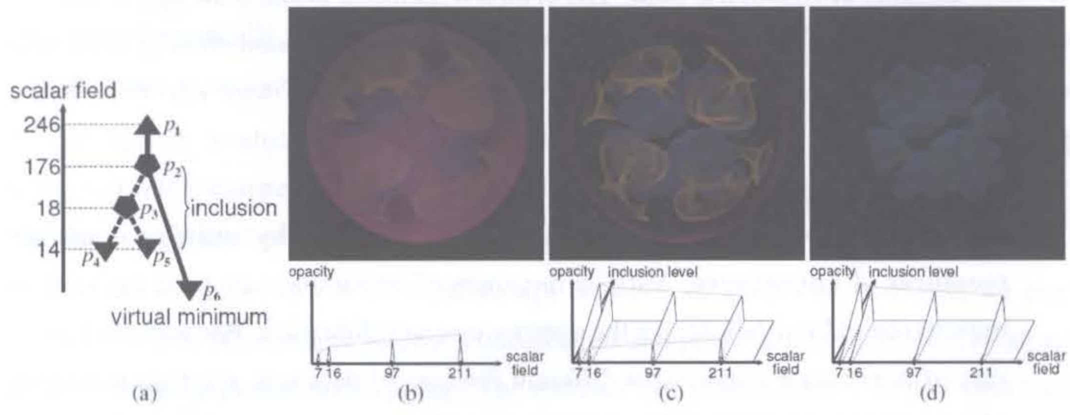


Figure 1.7: Visualizing simulated implosion in laser fusion: (a) The corresponding VST, (b) with topologically-accentuated 1D opacity transfer function, (c) with 2D opacity transfer function depending also on the inclusion level, and (d) with 2D opacity transfer function that visually extracts inner structures. [108, 54].

Since each contour in a data set corresponds to a point one-to-one on an arc in the contour tree and an arc corresponds to a region in the data set, it is possible to use the contour tree as a region index for a volume data set to identify different regions. This is useful to specify various transfer functions for different regions locally in volume rendering. However, as there are often many arcs in the contour tree, it is impractical and time-consuming to specify transfer functions for each arc (region) one-by-one manually. Weber et al. [122] used the contour tree to index different regions of a data set and specify transfer functions locally for individual regions. However, they neither consider inclusion relationships of regions nor automate the transfer function generations based on the contour tree. The principle technical challenges for these problems include *how to automate generations of transfer functions that are used to get a wide spread of dissimilar output renderings while revealing inclusion relationship between structures, and what measures are used to control transfer function differences between subregions based on the contour tree.*

1.3 Objectives of This Work

The ultimate goal of this thesis is to *develop new theories on effective contour tree simplifications, as well as new theories on utilizing topology to improve and automate the understanding of volumetric data.* The central *hypothesis* is that *if the approaches to be used for volumetric data analysis are based on topology, the automation of rendering parameter generations and understanding of data sets will be more effective.* More specifically, there are three main objectives as follows:

- **Develop new contour tree simplification approaches by utilizing multiple measures of importance.** Various measures of importance are used to evaluate importance of branches during the contour tree simplification. Because each measure of importance emphasizes different features of data sets, multiple measures of importance should be utilized simultaneously to more adequately evaluate importance of branches and improve the efficiency of contour tree simplification.

- **Develop new methods on depicting structural relationships in volume rendering by utilizing topology.** Spatial relations between various structures in a volume are also of interest to viewers besides shapes. Because the basic theory behind the depiction of structural relationships in volumetric data lies in the topology, the new theories should use topological properties derived from the contour tree to represent and depict relations between structures in volume rendering. The new theories will bridge the structural relationships and understanding of volumetric data by utilizing topology.

- **Develop new approaches for automating transfer function generations by utilizing topology.** In the new approach, the contour tree acts as a visual index to access various regions of a volume, and captures associated global topological attributes involved in volumetric data in the pipeline. The transfer function generations will include no or few involvements from users. The opacity transfer functions should depict inclusion relationship between structures and maximize differences between them, while the color transfer functions should convey meaningful relations among structures instead of arbitrarily defined colors based on users' preferences.

1.4 Approach and Methodology

The main approach to achieve the aims of this thesis is the utilization of topology of data sets. Specifically, this thesis uses the contour tree as the topological abstraction data structure to analyze data sets. The research methodology is detailed as follows:

We begin by evaluating approaches to the contour tree simplification through comparing various measures of importance. The result from this comparison is then used to design new approaches for the contour tree simplification.

In order to depict structural relationships between structures in a volume, we analyze the roles of relations in volume visualization and propose a new concept of structural relationship preserved mapping. The contour tree is then used to depict structural relationships based on the concept of structural relationship preserved mapping.

Based on the theories of structural relationship depiction, we analyze how fluid flows in a porous medium. This analysis helps us to design a model to control the distribution of opacity between branches in the contour tree. The color harmonization is one of popular designs in computer graphics for getting visual aesthetic appeal of rendering images. Topological attributes derived from the contour tree are also used to create harmonic colors in color transfer functions. The contour tree is used to control and automate the process of transfer function generations.

A series of experimental evaluation is then conducted to show the effectiveness of the proposed approaches in automatic generation of volumetric rendering. The experiments show how the proposed approaches are used to automate transfer function generations, while depicting inclusion relationships between structures. With these results, our approach can help users to analyze volumetric data efficiently and easily.

Chapter 2

Volume Visualization

The interpretation of volume data is considerably difficult because of their intrinsic complexity. In order to evaluate the abilities of volume visualization for volumetric data analysis, it is necessary to understand the principles of volume visualization algorithms. This chapter draws an overall picture of significant concepts of volume rendering, while focusing in particular on the optical model for volume rendering, typical rendering algorithms, e.g. ray-casting, as well as transfer functions.

2.1 Overview

As a subfield of scientific visualization, Kaufman et al. [58] defined *volume visualization* as:

> "[. . . . . .] a method of extracting meaningful information from volumetric data sets through the use of interactive graphics and imaging. It addresses the representation, manipulation and rendering of volumetric data sets, providing mechanisms for peering into structures and understanding their complexity and dynamics. Typically, the data set is represented as a 3D regular grid of volume elements (voxels) and stored in a volume buffer (also called cubic framebuffer), which is a large 3D array of voxels. However, data is often defined at scattered or irregular locations that require using alternative representations and rendering algorithms."

This definition encompasses all the important aspects of modern theories and practice in the field [47]. Volume visualization researchers continuously strive for improving the representation of data structures over the years. Volume visualization has proved itself as an effective technique for the exploration of large and complex data sets, even if its main application domain is still medical visualization.

Visualization unifies the largely independent but convergent fields of:

- Computer graphics
- Image processing
- Computer vision
- Computer-aided design
- Signal processing
- User interface studies

Direct Volume Rendering (DVR) (often called volume rendering) is one of the most flexible volume visualization approaches. Not only can it show simple shapes of surfaces, but it also allows more detailed 3D images of the internal structures of the volume to be presented. This thesis focuses on volume rendering of 3D scalar volume data.

2.2 Volume Data

Traditionally, computer graphics represented a model as a set of vectors which were displayed on vector graphic displays. With the introduction of raster displays, polygons became the basic rendering primitive, where the polygons of a model were rasterized into pixels, which represent the compounds of the frame buffer. Compared to surface data which solely determines the outer shell of an object, volumetric data is used to describe the internal structures of a solid object. A volume data is usually considered to represent a continuous function in a three-dimensional space. Thus, each point in space corresponds to a function value, formulated mathematically as:

$$f : \mathbb{R}^3 \rightarrow \mathbb{R} \tag{2.1}$$

Usually, datasets acquired from measurements do not have continuous values, they are limited to the points in space where measurements have been collected. A very common case is that the data points constitute a uniform regular grid. Such data is so called discrete volume data. The discrete volume data (also called volume data) can be thought of as a simple three-dimensional array of cubic elements (*voxels*), each representing a unit of space (see Figure 2.1). The three-dimensional array can be seen as a stack of two-dimensional arrays of data values and each of these two-dimensional arrays as an image (or slice), where each of the data values represents a pixel (see Figure 2.1). This alternative view is motivated by the slice oriented traditional way that radiologists look at a volumetric data set. A volumtric data is denoted by a matrix $V = \Gamma^{X \times Y \times Z}$ with X rows, Y columns and Z slices, which represents a discrete grid of volume elements (or voxels) $v \in \{1,...,X\} \times \{1,...,Y\} \times \{1,...,Z\}$. For each voxel, its scalar value is denoted by $I(v) : N^3 \to \Gamma$. This scalar value, for example, reflects the X-ray intensity in CT volumetric data. The voxel value can also be a vector to represent the object properties in some specific fields (e.g. computational fluid dynamics). Each voxel is characterized by its position in the 3D grid. Medical volume data obtained from MRI and CT-scanners are typically anisotropic with an equal sampling density in x and y direction but a coarser density along the z direction. The size is typically about 100 slices or more with 512×512 voxels each. The volume data is the basis for our assessment of volume rendering algorithms.

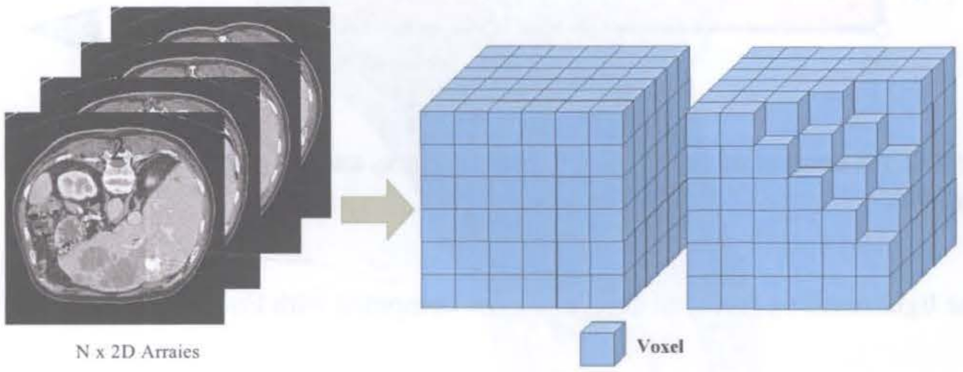


Figure 2.1: Voxels constituting a volumetric object after it has been discretized.

2.3 Optical Model for Volume Rendering

The basic goal of volume rendering is to find a good approximation of the low albedo optical model that expresses the relationship between the volume intensity and opacity function, and the intensity in the image plane. In this section, we describe a typical physical model on which volume rendering algorithms are based. Physical optical models are afforded by the radiative transport theory which attempts to view a volume as a cloud populated with particles. The transport of light is studied by considering the various phenomena at work. Light from a source can either be scattered or absorbed by particles. There might be a net increase when particles emit light themselves. The most important optical models for direct volume rendering are described in a survey paper by Nelson Max [78]. Models which take into account all the phenomena tend to be very complicated. Much simpler local models are used in practice. The optical model accounting for emission and absorption results in the *volume rendering integral* (VRI), that computes the light reaching the camera (eye).

This optical model (absorption plus emission) is the most common one in direct volume rendering. Particles emit light, and occlude, i.e., absorb, incoming light. However, there is no scattering or indirect illumination. Figure 2.2 illustrates the idea of the absorption and emission of radiant energy along a viewing ray [40].

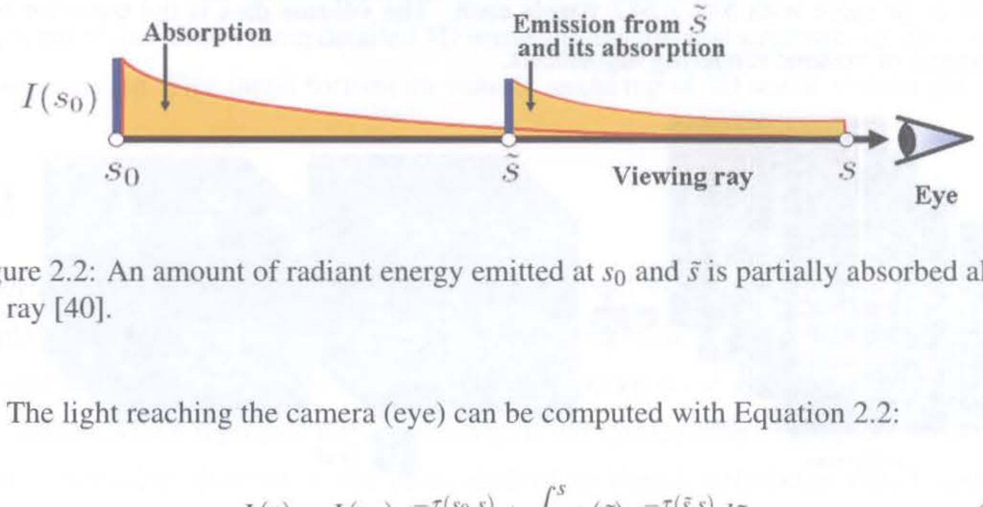


Figure 2.2: An amount of radiant energy emitted at s_0 and $\tilde{s}$ is partially absorbed along the ray [40].

The light reaching the camera (eye) can be computed with Equation 2.2:

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)} d\tilde{s}. \tag{2.2}$$

The light intensity is given by I, with $I(s)$ being the value at the exit point, i.e., the image pixel value, and $I(s_0)$ being the initial intensity at the start point s_0 of the ray, i.e., the light entering from the background. The initial intensity is absorbed along the viewing ray based on extinction τ. The function $q(s)$ specifies the emission at a point along the ray. In summary, the first term in Equation 2.2 accounts for the absorption of light as the ray passes through the volume and the second term captures the emission and color contribution from within the volume, which is also affected by absorption.

The extinction τ is also called the optical depth and computed with Equation 2.3:

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)ds, \tag{2.3}$$

where κ is the absorption coefficient. In practice, the volume rendering integral is evaluated numerically through either back-to-front or front-to-back compositing (i.e., alpha blending) of samples along the ray, which is most easily illustrated in the method of ray-casting [41].

2.4 Ray-Casting

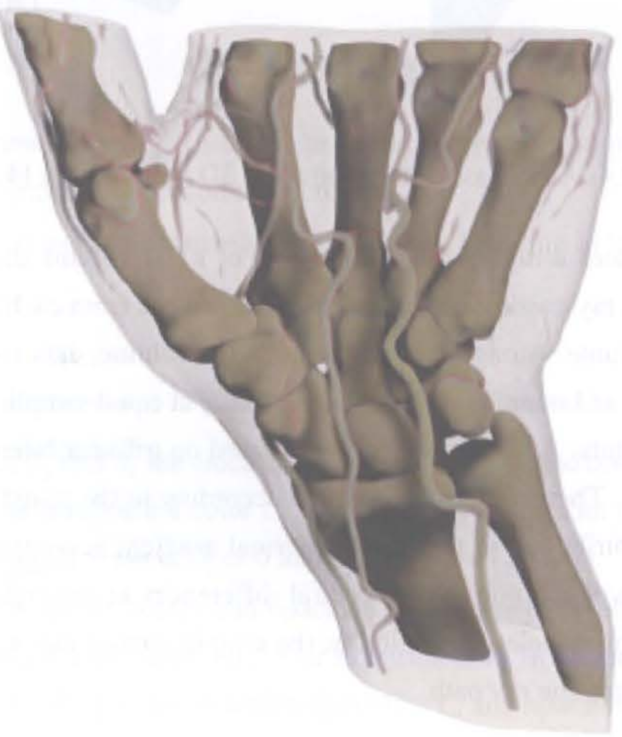


Figure 2.3: An example of rendering result using ray-casting [91].

Ray-casting is one of the typical image-order volume rendering algorithms [71, 72, 125]. Of all volume rendering algorithms, ray-casting has the largest body of publications over the years. The basic goal of ray-casting is to allow the best use of 3D data set and not attempt to impose any geometric structure on it. It solves one of the most important limitations of surface extraction techniques, namely the way in which they display a projection of a thin shell in the acquisition space. Surface extraction techniques fail to take that into account, particularly in medical imaging, data may originate from fluid and other materials which may be partially transparent and should be modeled as such. Ray-casting does not suffer from this limitation. Figure 2.3 is an example of ray-casting based rendering result [91].

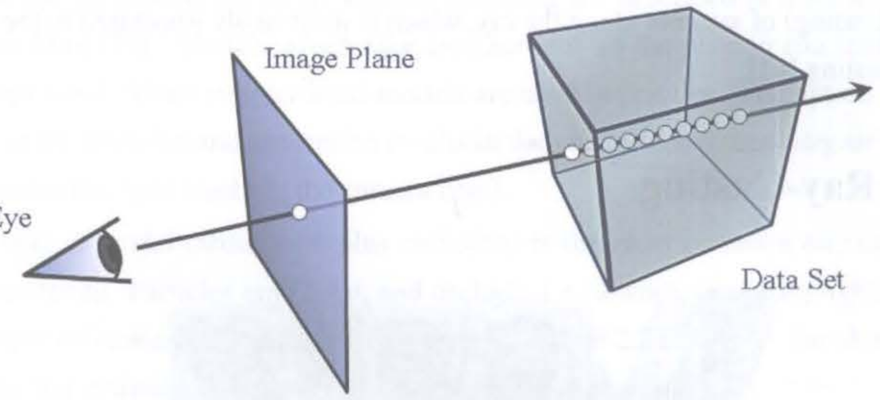


Figure 2.4: A ray casts into voxels of a 3D volume data [40].

Figure 2.4 illustrates a ray casting into voxels of a 3D volume data [40]. In the image-space oriented ray-casting approaches, rays are casted from each pixel in the image plane into the volume. Along their way through the volume, data are defined at the corners of each voxel and samples are calculated usually at equal sampling distances between two sample points. A sample is computed based on trilinear interpolation within a cell of eight voxels. Thereafter, it is classified according to the transfer functions. If that sample has a contribution to the ray, the normal gradient is computed based on a trilinear interpolation of the normalized central differences at the eight voxels of the cell which contains the sample point. Finally, the sample composites with the previous samples of the ray along the ray path.

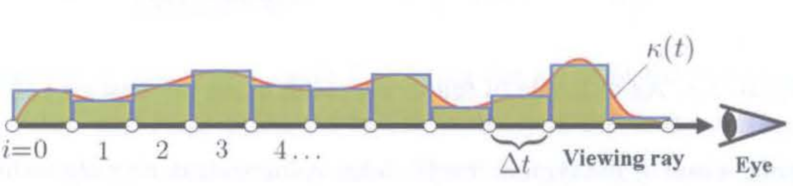


Figure 2.5: A ray is discretized to compute intensity analytically [40].

In the general case, Equation 2.2 can not be computed analytically. Numerical methods are used to compute the volume rendering integral in practice. The ray is divided into equal small segments, for which the optical properties are assumed to be approximately constant (see Figure 2.5 [40]).

Through approximating both the emissions and absorptions along a ray, the approximate evaluation of the volume rendering integral can be denoted as [41]:

$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i e^{-\tilde{\tau}(0,t)} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \prod_{j=0}^{i-1} (1-\alpha_i), \tag{2.4}$$

where $\tilde{C}$ is the final color, C_i is the color at the location i, T is the length of the ray, and Δt denotes the distance between successive resampling locations. Equation 2.4 can be evaluated iteratively by alpha blending in either back-to-front, or front-to-back order. Equation 2.5 is used to iteratively compute alpha blending in back-to-front order:

$$C_i^{'} = C_i + (1-\alpha_i) C_{i-1}^{'}, \tag{2.5}$$

where $C_i^{'}$ is the new color, $C_{i-1}^{'}$ is the color of the previous location, C_i and α_i are color and opacity of the current location respectively.

Equation 2.6 is used to iteratively compute alpha blending in front-to-back order:

$$\begin{aligned} C_i^{'} &= C_{i+1}^{'} + \left(1-\alpha_{i+1}^{'}\right) C_i, \\ \alpha_i^{'} &= \alpha_{i+1}^{'} + \left(1-\alpha_{i+1}^{'}\right) \alpha_i, \end{aligned} \tag{2.6}$$

where new values $C_i^{'}$ and $\alpha_i^{'}$ are calculated from the color C_i and opacity α_i at the current location i, and the composited color $C_{i+1}^{'}$ and opacity $\alpha_{i+1}^{'}$ from the previous location $i+1$. The starting condition is $C_n^{'} = 0$ and $\alpha_n^{'} = 0$ ($n = \lfloor T/\Delta t \rfloor$).

All algorithms obtain colors and opacities in discrete intervals along a linear path and composite them with Equation 2.5 or Equation 2.6. However, the algorithms can be distinguished by the process in which the colors C_i and opacities α_i are calculated in each interval i, and how wide the interval width Δt is chosen [79].

2.5 Transfer Functions

C and α in Equation 2.5 and Equation 2.6 are usually set by means of transfer functions, commonly implemented as lookup-tables. The lookup-table maps scalar value and other data features to various colors and opacities. This mapping is also called classification, which classifies data into various objects based on different opacities. Figure 2.6 shows relations between transfer functions and various data elements. In the most simple form, transfer functions use just scalar values for this mapping, but it is also possible to use the local derivatives or other information for this purpose. The specification and generation, in automatic or semiautomatic fashion, of these functions are part of unsolved problems in direct volume rendering. Figure 2.7 shows an example of user interface of transfer function specifications used in *Kitware Inc.*'s *VolView 3.2* [2]. Curves/polylines are often utilized in a conventional user interface to define the mapping in transfer function specifications. In manual exploration of an unknown data set, these user interface controls are moved and reshaped in order to best match the feature to be classified. The more distinct features to visualize, the more tuning of visual appearance is needed so that the joint rendering of the entire volume becomes informative. Thus, the user's interaction with the transfer function constitutes a central part of the exploratory process [74].

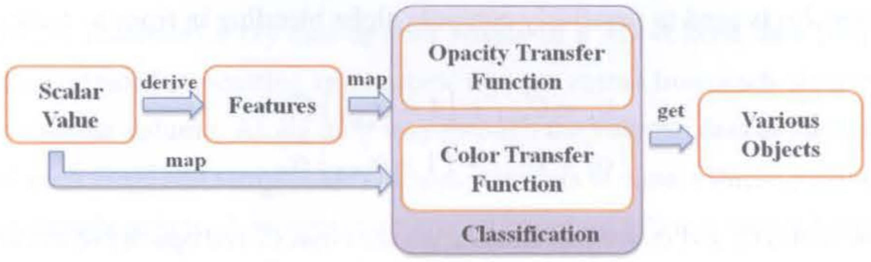
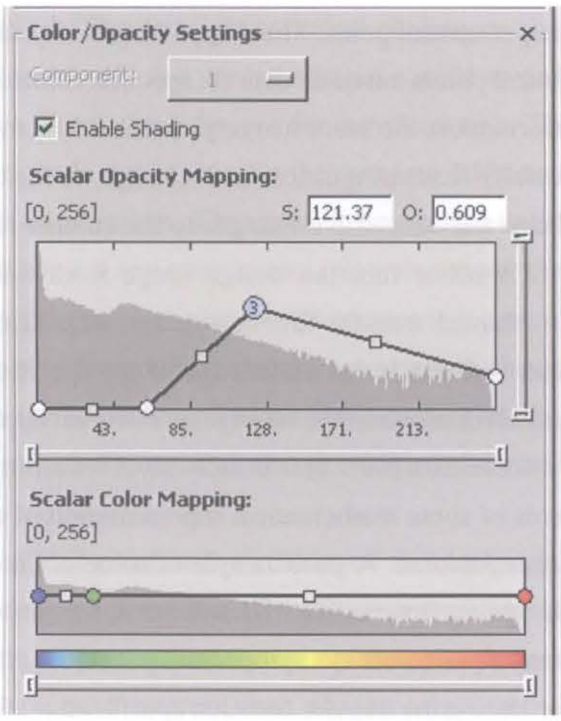


Figure 2.6: Relations between transfer functions and data elements.

Since the scalar field is not available in a continuous representation, but only in a discrete form, it is necessary to interpolate the available data at the sampled points during composition. In the presence of a transfer function, it is necessary to decide whether this interpolation step is done before or after the application of the transfer function. The former case is called *pre-classification* (i.e classification before interpolation) and the later *post-classification* (i.e classification after interpolation). Usually, the post-classification is "correct" and creates higher quality of renderings in the sense that values in the underlying continuous scalar field are being interpolated and then classified [47, 41].


Color/Opacity Settings

Component: 1

[✓] Enable Shading

Scalar Opacity Mapping:

[0, 256] S: 121.37 O: 0.609

Scalar Color Mapping:

[0, 256]

Figure 2.7: A user interface of transfer function specifications [2].

2.6 Approaches of Transfer Function Design

The transfer function specification is arguably one of the most important tasks in volume visualization. Research on transfer functions is focused on making the parameter space of transfer functions easier to explore. Many methods have been proposed to that end ranging from 1D transfer function, multi-dimensional transfer function, sematic transfer function, to incorporate other data features in transfer functions. While the transfer function's role is simply to assign optical properties such as opacity and color to the data being visualized, the value of the resulting visualization will be largely dependent on how well these optical properties capture features of interest. Specifying a good transfer function can be a difficult and tedious task for several reasons. First, it is difficult to uniquely identify features of interest in the transfer function domain. Even though a feature of interest may be easily identifiable in the spatial domain, the range of data values characterizing the feature may be difficult to isolate in the transfer function domain. This is due to the fact that other uninteresting regions may contain the same range of data values. Second, transfer functions can have an enormous number of degrees of freedom. Even simple 1D transfer functions using linear ramps require two degrees of freedom per control point. Third, typical user interfaces do not guide the user in setting these control points based on data set specific information such as shapes. Without this type of information, the user must rely on trial and error. The trial and error interaction can be especially frustrating since small changes to the transfer function can result in surprisingly large and unintuitive changes to the volume rendering [64].

Existing schemes of transfer function design range from fully manual to semi-automatic techniques. Manual transfer function design is primarily based on experience, allowing the user to bring in his knowledge of the specific data as well as his personal taste. Trial-and-error is one of the widely used manual approaches to find good transfer functions for volume data [66]. This usually involves arbitrarily and repeatedly manipulating coefficients of some mathematical representation of the transfer function to adjust the visualization outcome. A good sample of some of the existing approaches for transfer function designing (interactive trial-and-error, metric-based, contour graph and design galleries) were squared off in a symposium panel [87]. Recent researches try to use other information during the transfer function specification, for example, the contour tree and illustrative rendering. Typical opacity transfer function and color transfer function approaches are reviewed in this section.

2.6.1 Early Transfer Function Approaches

Early volume rendering approaches often use 1D transfer functions to explore volume data [71]. One-dimensional transfer functions usually refer to transfer functions which only use one parameter (e.g. scalar value) to generate optical properties. This can be defined through directly mapping scalar values to optical properties, or using different interfaces.

Fang et al. [42] presented an image-based transfer function model that integrates 3D image processing tools (e.g. image enhancement and boundary detection) into the volume visualization pipeline. The model defines a transfer function as a sequence of 3D image processing procedures, and allows users to adjust a set of qualitative and descriptive parameters to achieve their subjective visualization goals.

Design galleries [76] approach is another viable alternative to facilitate transfer function selection. It generates all possible transfer functions simultaneously based on automatic analysis, each representing a different configuration of the transfer function. The satisfactory transfer function is then selected from these representatives and then implicitly optimized. The principle technical challenges are to generate different transfer functions automatically that are used to get a wide spread of dissimilar output renderings and arrange the resulting designs for easy browsing. The color transfer function is parameterized by values that segment the data into different subranges, which are arbitrarily assigned different colors. Thus color is being used only to identify subranges of the data, and not to convey any quantitative relations among the data.

Contour spectrum [9] approach consists of metrics that are computed over a scalar field. This more data-centric approach visually summarizes the space of isosurfaces in terms of metrics like surface area and mean gradient magnitude, thereby guiding the choice of isovalue for isosurfacing, and also providing information useful for transfer function generation.

König and Gröller [66] combined elements of the design galleries approach and trial-and-error techniques with the use of real-time raycasting hardware. In this approach, the transfer function specification is simplified as a three steps process: first the user indicates scalar ranges of interest, then assigns colors to these interesting ranges and finally assigns opacities to these ranges. Numerous feedback renderings are performed during this process in order to get a good transfer function. The method presumes that the user already knows the scalar range of interest.

These approaches do not provide any mechanism to automate transfer function generation and still apply transfer functions globally instead of locally to structures.

2.6.2 Multi-Dimensional Transfer Functions

2D transfer functions were introduced by Levoy in 1988 [71]. Levoy introduced two styles of transfer function, both two-dimensional, and both using gradient magnitude for the second dimension besides using the scalar value as the first dimension. Furthermore, other data information (e.g. second derivatives) are introduced to form the multi-dimensional transfer function.

Kindlmann and Durkin [60] proposed a semi-automatic method which is a highly-regarded technique for generating transfer functions from volume data. This method makes the assumption that features of interest in the data are the boundary regions between different materials. It uses a data structure named histogram volume to capture the relationship between data values and boundary representations (first and second directional derivatives).

Kniss et al. [64, 65] noted that previous approaches assumes that a given isovalue has uniform meaning throughout the data set. They observed that this assumption causes problems, and designed an interface to construct multi-dimensional transfer functions interactively (based on scalar value, gradient magnitude, and a second directional derivative). This interface adds derivatives information as parameters to the transfer function, but continued to apply the same transfer function everywhere in the data. A color picker widget is provided based on the hue-lightness-saturation (HLS) color space. The user specifies a color for an object simply by clicking on that object, then moving the mouse horizontally and vertically until the desired hue and lightness are visible. There are no theoretical relations between the selected color and the object. In order to automate the feature selection in multi-dimensional parameter space for multi-dimensional transfer functions, Maciejewski et al. [75] applied a clustering method to the 2D feature space and cluster spaces into subregions with similar features. However, users are still required to do complex manual interactions. Because it does not consider inclusion relationship of structures inside data, automation of transfer function generations is still not solved.

Kindlmann et al. [61] added the curvature of the function as a parameter into multi-dimensional transfer functions. The curvature of a surface is defined by the relationship between small positional changes on the surface, and the resulting changes in the surface normal. Curvature information can be used to emphasize ridge and valley surface creases. This approach still lacks any spatial locality.

Huang and Ma [53] presented a technique that can suggest a 2D transfer function by using the results of partial region growing from a point selected in volume space. Tzeng et al. [116] described an interface for specifying transfer functions that takes into account additional data properties such as texture and position besides the scalar value and its derivatives. The texture and position information are used as the seed information to classify volume data. This method can be regarded as an extension of region growing based approach for transfer function specification.

Lum and Ma [73] used the concept that transfer functions can be used for specifying of any of a number of optical properties including surface illumination, and presented a multi-dimensional lighting transfer function. The lighting transfer function is used to control how lighting is used for illustrating different material boundaries in a volume. The input of lighting transfer functions consists of samples read along the gradient direction.

Correa and Ma [35] introduced *occlusion spectrum*, which is a 2D distribution of intensity values and occlusion, into 2D transfer functions in order to depict occlusion patterns in rendering. Users need to interact occlusion parameters in the occlusion transfer function editor to visualize volume data, which introduces further parameters and decisions into the complex parameter space of multi-dimensional transfer functions.

In a word, automation of transfer function specifications are still not solved, but more parameters to be modulated by users are introduced in multi-dimensional transfer functions.

2.6.3 Halo Transfer Function and Tone Mapping

In order to enhance depth perception in volume rendering scene, Bruckner and Gröller [19] presented the halo transfer function to classify structures of interest based on data value, direction and position. The halo transfer function defines a basic seed intensity at a sample position. This value is then combined with gradient magnitude and dot product between view vector and the normalized gradient vector to form the final halo seed intensity. Then a field of halo intensity values is generated from the seeds by applying a filtering process. Finally, the halo intensities are mapped to the actual color and opacity contributions of the halo and combined with the regular volume rendering.

Visualizing volumetric data with both high spatial and intensity resolutions on a limited resolution display device is challenging. Yuan et al. [126] presented a transfer function specification interface with nonlinear magnification of the density range and logarithmic scaling of the color/opacity range. A dynamic tone mapping is used to preserve high resolution details on regular display devices.

These transfer functions still need complex interactions.

2.6.4 Semantic Transfer Function Specifications

The specification of transfer functions is a complex task and requires visualization expert knowledge about the underlying rendering technique. In the case of multiple volumetric attributes (e.g. density, gradient magnitude, etc.) and multiple visual styles based on different optical properties, the specification of the multi-dimensional transfer function becomes more challenging and non-intuitive. Salama et al. [92] introduced an additional level of abstraction for parametric models of transfer functions — semantic level, in order to enable a non-visualization expert to specify transfer functions easily. The proposed sematic model is defined based on a set of reference data, a list of the relevant structures contained in the data, and transfer functions template for the reference data. Semantics such as "sharpness" and "visibility" can be created and modified by using primitive editors.

Rautek et al. [89] presented a semantic layer mapping from several volumetric attributes to multiple illustrative visual styles in volume rendering. Semantic layers allow a domain expert to specify the mapping in the natural language of the domain. Volumetric attributes and visual styles are represented as fuzzy sets. The mapping is specified by rules that are evaluated with fuzzy logic arithmetics. The user specifies the fuzzy sets and the rules without special knowledge about the underlying rendering technique.

Semantic layers allow for a linguistic specification of the mapping from attributes to visual styles replacing the traditional transfer function specification. The basic low level of semantics is still the traditional different transfer function specification methods as reviewed above.

2.6.5 Topology Controlled Transfer Function Specifications

Section 3.6.4 reviewed this topic in details. Topology controlled transfer function specification is discussed here in order to show the integrity of the review of transfer function approaches. Introducing topological attributes into the transfer function specification allows volume rendering to capture global characteristics of the data set while locating regions of particular interest. Fujishiro et al. [46] introduced topological attributes in transfer function generations by using Reeb graphs. Takahashi et al. [106, 107] and Takeshima et al. [108] used topological attributes derived from the contour tree to define transfer functions. However, they applied transfer functions to a data set according to fixed topological indices (e.g. depth) and did not explore transfer function automation deeply. Conventional transfer function approaches cannot distinguish between distinct features that share the same scalar value. To overcome this problem, Weber et al. [122] used the contour tree to index various subregions of a volume and specify transfer functions locally for individual subregions. However, as there are often many branches in a contour tree, it is impractical and time-consuming to define transfer functions for each subregion corresponding to a branch one-by-one manually. The inclusion relationship is also not considered in [122].

2.6.6 Color Transfer Functions

Color transfer functions map scalar densities and other features to colors in order to label different objects and create aesthetic appeal in volume rendering. Color transfer functions have in most cases been guided by personal preferences or even just by random assignments [120]. Furthermore, color harmony is one popular design aspect in terms of aesthetics. Cohen-Or et al. [31] introduced a framework of automated image color harmonization. Wang et al. [119, 120] extended this color harmonization to be used in volume visualization. They presented how color harmonization is used to semi-automate color definitions in volume rendering. In their pipeline, users need to specify the hue component of a HSV color and the system automatically optimize other color components in order to create a harmonic color. Our work extends the color selection algorithm in [119] by incorporating topological features to automate color transfer function generations in volume rendering.

In addition, recent work on transfer functions introduced other features of volume data into the transfer function pipeline. For example, Caban and Rheingans [20] used local textural properties of voxels to differentiate structures which have similar intensity values. Selver and Güzelis [95] used a volume histogram stack to do semi-automatic transfer function initialization. Neural networks are used in this process. Correa and Ma [34] mapped size of features to color and opacity. In this chapter, we consider the effects not only of size (volume) of structures, but also of hypervolume and persistence of structures on transfer functions. Persistence, volume and hypervolume are parameters derived from the contour tree and are more meaningful in representing the importance of structures in transfer functions.

2.6.7 Summary of Previous Work

The transfer function specification methods investigated, except topology controlled transfer function specification approaches [106, 107, 108, 122], use global properties to detect isovalues that are significant for transfer function specifications. Implicitly, this determines that the isovalue is significant everywhere in the data. Disadvantages of these approaches are obvious: they cannot depict different objects which have similar scalar value ranges effectively; they also cannot consider global structure of the volume into the pipeline. Topology controlled transfer function specification methods are good starting points for specifying transfer functions for local individual regions while emphasizing topology of the data. This chapter aims to extend the topology controlled transfer function approaches and to create transfer functions for different regions indexed by the contour tree automatically.

Furthermore, although the investigated approaches could find good transfer functions for structures of interest in some degree, there is still not much research in the field of color transfer function based on perceptive color space. Most of the previous work on transfer function generations focus on the opacity transfer function specifications, the color transfer function is usually selected randomly based on the user's individual taste or commonly accepted preferences (e.g. vessels are usually red). However, colors could be used to reveal interrelationships within objects inside a data set. For example, in the perceptual color space [17, 16], the difference between objects (e.g. distance, scalar value) can be depicted based on the perceptual dimensions of colors, in order to improve perception between different structures. We believe that specifying color transfer functions for structures by utilizing topology based on perceptual color space could improve the understanding efficiency of volume data.

This thesis analyzes problems of existing transfer function design approaches, and proposes an automatic transfer function generation method by utilizing topological attributes derived from the contour tree. Details of the automation of transfer function generation proposed in this thesis will be covered in Chapter 6.

2.7 Relationship Depiction

As mentioned in Section 1.2.2, spatial relations between various structures in a volume are also of interest to viewers besides shapes. Relationship and its uses have been studied in several research fields. In information visualization, the relationships between different objects are often represented using tables, graphs, etc. [36]. Hao et al. [51] developed an algorithm to reflect hierarchy and importance-based relationships in time series data. It allows users to quickly perceive relative importance and hierarchy relations within sets using importance-driven layouts. Collins and Carpendale [32] presented a method named VisLink to depict relationships between 2D visualizations in 3D space. Relationships are revealed using connections between two visualizations. The VisLink environment allows the viewer to query a given visualization in terms of connections.

In computer graphics, scene graph [101] is a commonly used tree data structure for representing the hierarchical relationship of geometric objects. The major purpose of scene graph is to efficiently model and organize objects in a scene.

In 3D visualization, Cook et al. [33] used an A-Buffer data structure to order relationships of polyhedra cells of a mesh in order to depict the visibility relationship. The A-buffer (anti-aliased, area-averaged, accumulation buffer) is a general hidden surface mechanism. It resolves visibility among an arbitrary collection of opaque, transparent, and intersecting objects [21]. Kriz et al. [68] embedded 3D nano-structures into their associated macro-properties of wave-velocity surface topology to depict the structure-property relationships. The enhanced interpretation helps users to improve the understanding of the data set. Viola et al. [118] used a focus+context approach to depict importance relationship of objects. Occluded objects are represented through importance-driven volume rendering. Occluding objects and occluded objects are applied with different importance factors. Other similar approaches [127, 121, 69, 18] also use the concept of focus+context to explore inner structures in volume rendering.

Chan et al. [27] presented a relation-aware volume exploration pipeline. In [27], various spatial relations are defined, represented, and used in volume rendering. However, all of these operations highly depend on a segmented data set. If the segmentation information is not available, the relation definition lacks object boundary information and is difficult to set up. This unavoidably affects the following relation representation and effectiveness of uses in volume rendering. If the segmentation creates wrong information, the following relation based exploration does not make sense at all. Image segmentation is acknowledged as a very difficult and complex task. It is often desirable to be avoided or at least postponed until after a first inspection of 3D data with volume rendering. Furthermore, the graph used to represent relations in a data set is a general graph and it is difficult for users to understand the connection between relations and the data set (see Figure 2.8). This work motivates us to introduce topological relationship to volume rendering to analyze volume data sets. Elmqvist and Tsigas [39] defined a taxonomy of the design space of occlusion management techniques to formalize a common terminology and theoretical framework for occlusion interactions. They derived a set of five orthogonal design patterns for effective reduction of 3D occlusion.

All these investigated approaches try to represent different relationships in a data set based on pure rendering techniques, but they do not touch the basic theory behind the problem — the topology.

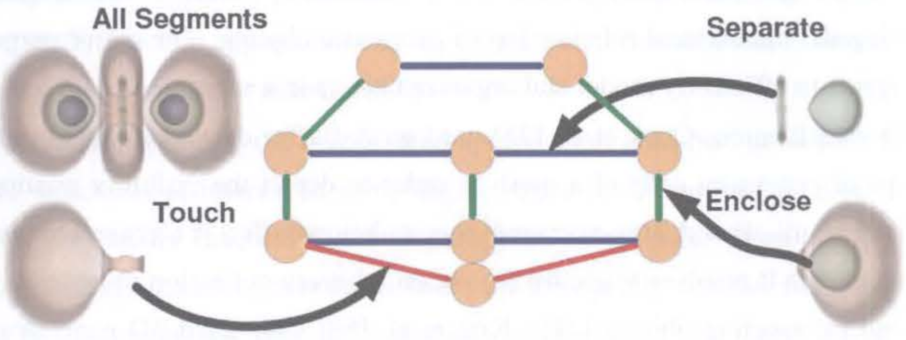


Figure 2.8: Relation representation in a graph [27].

Representing structures with different relationships belongs to the problem of revealing topological information between them. Kniss et al. [63] proposed a framework to encode topology information of a data set in raster-based representation, and used it to enforce constraints or fix classification errors in visualization applications. One of its typical application examples is to render an MRI head data set with correct position relationship of various objects by fixing classification errors. The topological attributes in a volume were also considered in a framework to define transfer functions in volume rendering [108]. The contour tree has been used to explore the relation between iso-surfaces and their evolution [23]. Shinagawa et al. [96] introduced the analysis of parent-child and sibling relationships between contours. Takahashi et al. [107] used the contour tree to depict inclusion relationships in volume rendering. The contour tree was also used to represent topology of volume data and index subregions in transfer function specifications [122].

In contrast, our work in this thesis focuses on relationship preservation in volume rendering. Thus, more relations other than hierarchy as used in computer graphics are considered. Because object relations in a data set are related to the topology of the data set, our work also focuses on using the contour tree to reveal various relationships in volumetric data. Instead of depending on segmented data set as used in [27] to define spatial relations, our approach creates the contour tree directly from the data set and then use it to represent topological relations. More details are covered in Chapter 5.

2.8 Summary

This chapter firstly provided an overview of volume visualization, and then reviewed the basics of volume rendering. It gave the definition of volume data and introduced one of the widely used optical models for volume rendering. The details of ray-casting algorithm was provided to show how volume rendering is used to generate 3D rendering results. Because the key to comprehensible volume rendering still lies in the design of effective transfer functions, this chapter defined transfer functions and presented problems of transfer function generations for current approaches. Relationship depiction in several research fields was also reviewed in this chapter. They are used to show how relationship depiction can be used in volume rendering to help users to better understand volumetric data.

Chapter 3

Contour Trees

This thesis explores effective uses of contour trees in volume rendering. This chapter aims to cover foundations that are required to understand subsequent chapters without going into the details of specific papers. The chapter reports the state of the art of research on contour trees, particularly drawing upon preliminaries, definitions and computations of contour trees. As one of the abstraction techniques to reduce data by computing a topological description of the data, contour trees have wide applications in visualization. This chapter also covers typical applications of contour trees in visualization.

3.1 Preliminaries

Since the contour tree depends on contours and isosurfaces, which are inherently geometric in nature, the contour tree and most other topological techniques are more closely related to geometric techniques than signal processing. Before the details of the contour tree is given, this section introduces preliminary concepts used to construct contour trees.

3.1.1 Sampling and Reconstruction

Volume data are usually known as discrete point samples in 3D space. Geometric and topological techniques, however, generally require continuous functions in order to find critical points and to determine the relationship between them. As a result, scientific visualization techniques that exploit geometry and topology commonly reconstruct the reconstruction function using geometric *meshes* [22].

A *mesh* is a subdivision of the region of interest into geometric primitives called *cells*. In general, the vertices that define the cells are given by the *grid* — i.e. the mesh is constructed using the point samples as vertices. Common example cells in 3D space include tetrahedral and hexahedral cells. An *interpolant* is used on the mesh to compute interpolated data values during the contour tree computation (e.g. finding critical points). A trilinear interpolant is a typical interpolant in 3D space.

This thesis considers trilinear interpolation for a hexahedral cell because hexahedral meshes are most commonly used for direct volume rendering of data given as samples on a regular, rectilinear mesh.

3.1.2 Manifolds

Given a continuous scalar field $\mathscr{F}$ defined on a domain $\mathscr{M}$ ($\mathscr{M} \subset \mathbb{R}^d$), a scalar function $f : \mathscr{M} \to \mathbb{R}$ is a real-valued function. It is often useful to think of f as a *manifold*: a mathematical generalization of a surface. Formally, a *topological d-dimensional manifold*, d-manifold for short, is a topological space that is everywhere locally homeomorphic to $\mathbb{R}^d$ [22]. A *homeomorphism* is a function f which is a bijection (so it has an inverse f^{-1}) with both f and f^{-1} being continuous [70].

A d-manifold is a topological space that locally looks like $\mathbb{R}^d$ (the Euclidean space). In other words, each point admits a coordinate system, consisting of coordinate functions on the points of the neighborhood, determining the topology of the neighborhood [134]. For three-dimensional scalar data, f can be written as the set of points of the form $\{(x,y,z,f(x,y,z)) : (x,y,z) \in \mathbb{R}^3\}$. This set of points forms a 3-manifold embedded in a four-dimensional space. f measures a property such as light intensity or absorption of radiation in CT in medical imaging. f may also be treated as an intensity map, emphasizing spatial dimensions and the function value.

3.1.3 Level Sets and Contours

An *isoline* is a line of points with a common value of the function f. This common value is called the *isovalue* of the isoline. In general, considering a continuous scalar field $\mathscr{F}$ defined on a domain $\mathscr{M}$ ($\mathscr{M} \subset \mathbb{R}^d$), $f : \mathscr{M} \to \mathbb{R}$. $\mathscr{M}$ is assumed to be a simplicial complex[1]. Any 3D volume can be decomposed into a simplicial complex. The function f within $\mathscr{M}$ is completely determined by the values on the n simplex vertices of the domain $\mathscr{M}$. For a point inside a simplex, its function value is a linear interpolation of the values on the vertices. The *functional range* of the field $\mathscr{F}$ is the interval between the minimum and maximum values of the function f , $[f_{min}, f_{max}]$. For a scalar value $h \in [f_{min}, f_{max}]$, the *level set* of the field $\mathscr{F}$ at the value h is the subset of points $L(h) \subset \mathscr{M}$ such that $f(x) = h$ for any $x \in L(h)$. This is expressed as $L(h) = \{(x) \,|\, f(x) = h\}$, h is the *isovalue*. Topologically, a level set may consist of 0, 1, or more connected components [24]. In a simplicial complex, these connected components will be of dimension $\leq (d-1)$.

For two-dimensional data, each level set is a linear feature, called an *isoline*. The most familiar use of isolines is on topographic maps, where the function f represents land elevation. For three-dimensional data, each level set is a set of surfaces in three dimensions: each isocontour is called an *isosurface*. *Contour* is often used as a general term for a connected component of a level set in a space of arbitrary dimension.

3.2 Topological Abstractions

Abstraction is one of the principle methods to reduce data information in order to be analyzed practically by a human. Usually, abstraction reduces data by computing a topological description of the function. Principle topological abstractions include Morse-Smale Complexes, Reeb graphs and contour trees.

3.2.1 Morse-Smale Complexes

A real-valued smooth scalar function $f : \mathscr{M} \to \mathbb{R}$ defined on a smooth d-manifold $\mathscr{M}$ is a *Morse function* if none of its critical points are degenerate (i.e., the Hessian matrix at all critical points is non-singular), and no two critical points have the same function value. Morse theory studies the relationship between critical points of a Morse function and the topological structure of its domain space. Complex natural phenomena, both

[1]In general, we are unable to represent surfaces precisely in a computer system, because it has finite storage. Consequently, surfaces are sampled and represented with triangulations. A triangulation is a *simplicial complex*, a combinatorial space that can represent a space. With simplicial complexes, the topology of a space is separated from its geometry, much like the separation of syntax and semantics in logic [134].

sampled and simulated, are often modeled as Morse functions. MRI scans generate Morse functions that are used in medical imaging to reconstruct human tissues. Electron density distributions computed by high-resolution molecular simulations are Morse functions whose topology express bonds among the atoms in molecular structures [84]. The structure of geometric models used in computer graphics and CAD applications can be effectively represented in terms of the topology of a Morse function [52, 15]. The *Morse Lemma* [77] states that in the neighborhood of a critical point p of f , the function can be rewritten as a quadric (taking 3-manifold as an example) [48]:

$$f(x,y,z) = f(p) \pm x^2 \pm y^2 \pm z^2. \tag{3.1}$$

The *index* of p is equal to the number of negative signs in the above expression. Critical points of index 0, 1, 2 and 3 are called *minimum, 1-saddle, 2-saddle* and *maximum*, respectively.

The *Morse-Smale complex* (MS complex) is a topological data structure that provides an abstract representation of the gradient flow behavior of a scalar field [98, 15, 49, 50]. It decomposes the domain $\mathscr{M}$ into monotonic regions and represents the topological structure of f. This is done through gradient lines (or integral lines).

Assuming that the function f is everywhere differentiable, the gradient at each point is calculated. An integral line of f is a maximal path in $\mathscr{M}$ whose tangent vectors agree with the gradient of f at every point of the path. Each integral line has a natural origin and destination at critical points of f where the gradient becomes zero. *Ascending* and *descending* manifolds are obtained as clusters of integral lines having common origin and destination respectively. The MS complex partitions $\mathscr{M}$ into regions by clustering integral lines that share common origin and destination. It is simply the set of these regions, usually shown by drawing the boundaries. In MS functions, the integral lines connect critical points of different indices. For example, the three-dimensional cells of the MS complex cluster integral lines that originate at a given minimum and terminate at an associated maximum. The cells of different dimensions are called *crystals*, *quads*, *arcs* and *nodes*. Note that the MS complex is an overlay of ascending and descending manifolds, which individually partition $\mathscr{M}$ as well. The arcs form a pairing of critical points that we call the combinatorial structure of the MS complex [50].

Figure 3.1 illustrates a three-dimensional cell of the MS Complex [49]. Every crystal has a unique origin and destination node, the minimum and maximum, respectively,

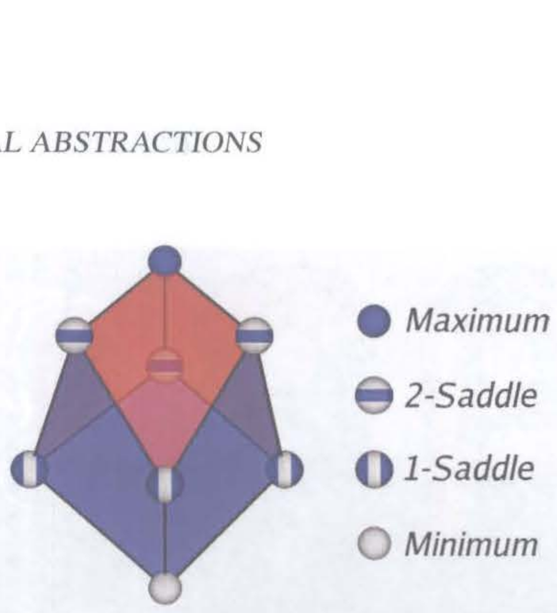


Figure 3.1: The boundary of a crystal in the Morse-Smale complex consists of lower-dimensional cells: quads, arcs and nodes [49].

which are end points of integral lines lying within. Note that the glyphs used for critical points indicate their local neighborhood.

3.2.2 Reeb Graphs

The MS complex is a fundamental topological structure that partitions the domain of a real-valued function into regions having uniform gradient flow behavior. The *Reeb graph* [90, 96, 97] is also a fundamental data structure that encodes the topology of a manifold. However, it expresses the evolution of level sets (contours) in cross-sections of a manifold, and describes the connectivity of its level sets [37]. The Reeb graph of f is obtained by contracting the level sets of f to points. An example of this construction is shown in Figure 3.2 for the triple torus model. In this figure, the function f is equal to the z coordinate of its vertices [85].

Points on the Reeb graph that correspond to contours passing through critical points of f (maxima, minima and saddles) are called *nodes*. The rest of the Reeb graph consists of *arcs* connecting the nodes. The Morse theory guarantees that the contour topology changes only in correspondence to critical points. Moreover, under the assumption that the height function is Morse, the structure of the Reeb graph is rather simple and represents the topological skeleton of the object. The triple torus in Figure 3.2 can be characterized topologically with eight singular points. These singular points are represented as nodes and are connected to each other by an edge (arc) representing a set of homotopically equivalent connected contours on consecutive cross-sectional planes. As clearly seen from the figure, a Reeb graph can provide a topological skeleton of a

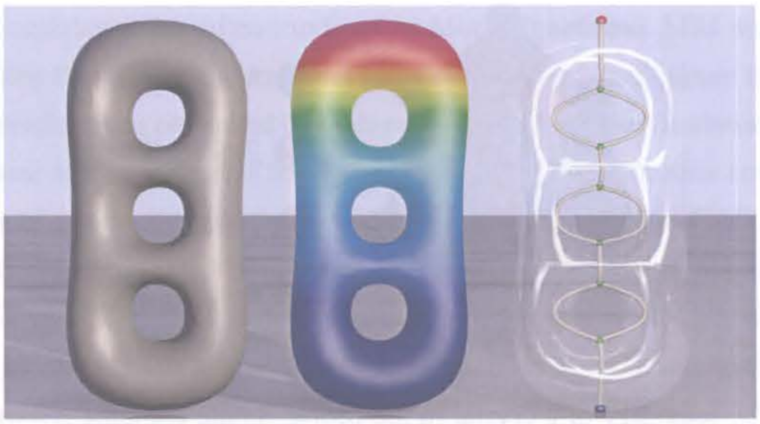


Figure 3.2: Reeb graph of the height function on the triple torus. The three tunnels of the model are mapped to three loops in the graph [85].

3D surface [45].

3.2.3 Contour Trees

Both Reeb graphs and contour trees track changes to a contour as a single parameter is varied. However, the Reeb graph is more general than the contour tree, and is computed for manifolds more complex than a simple surface defined by a function over $\mathscr{M}$ ($\mathscr{M} \subset \mathbb{R}^d$). A Reeb graph is defined to be a *contour tree* when it is free of loops (cycles), and effectively captures the transitions of level sets when f is single-valued. Tierny et al. [112] introduced a procedure named *loop surgery* to reduce a Reeb graph to be loop free, which is similar to a contour tree.

Figure 3.3 shows an example of a 2D mesh and its corresponding contour tree. Three level sets with isovalue of 5 are drawn upon the mesh. Different level sets and their corresponding points in the contour tree are encoded with colors. Since the contour tree is at the core of this thesis, section 3.3 will give a more formal definition and computation of the contour tree.

In summary, the differences among MS complexes, Reeb graphs and contour trees can be depicted as follows: Given a scalar function $f : \mathscr{M} \to \mathbb{R}$, defined on a manifold surface $\mathscr{M}$, the topology of $\mathscr{M}$ can be effectively studied by contracting the connected components of the level sets of f to single points (i.e., the Reeb graph) or joining its critical points with flow lines of f (i.e., the MS complex). The Reeb graph and the MS complex provide an abstract representation of the surface $\mathscr{M}$. The former is based on

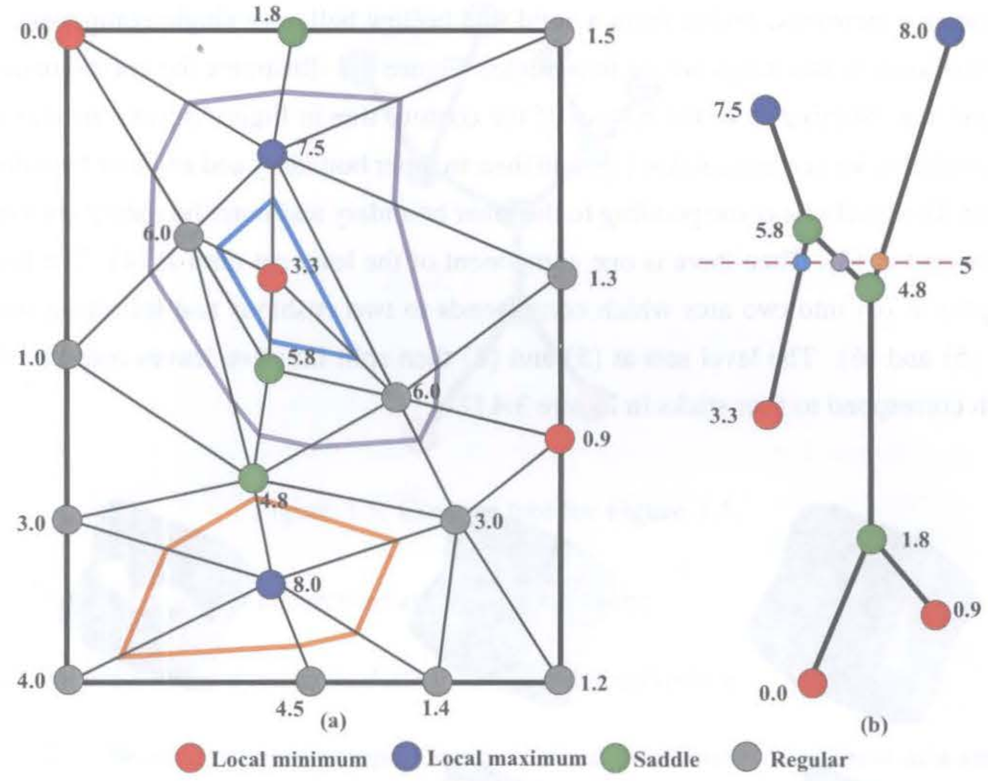


Figure 3.3: An example of a 2D mesh and its contour tree. Different level sets are drawn upon the mesh.

the topological changes of the level sets of f, the latter mainly uses the connectivity of the critical points with respect to the behavior of the gradient field of f. A Reeb graph is called a contour tree when it has no loops [86].

3.3 Setup of Contour Trees

Recall that in Section 3.2, we give a short introduction of the contour tree as one of topological abstraction techniques to reduce data information. This section gives a formal definition of the contour tree and details on setting up contour trees.

3.3.1 Definition of the Contour Tree

The *contour tree* is a graph that tracks contours of the level set as they appear, join, split and disappear. Figure 3.4 [24] illustrates level sets of a function $f(x)$ that, as the parameter x increases, evolve from a solid to a hollow ball, to a single component, to two cushions, to two rings and to four sticks. Figure 3.5 illustrates the corresponding contour tree. Starting from the bottom of the contour tree in Figure 3.5 and increasing the parameter, we see one solid at (1), and then an inner boundary and an outer boundary appear. The level sets corresponding to the inner boundary and outer boundary encloses and merges at (3). Then there is one component of the level set until at (4). The level set splits at (4) into two arcs which corresponds to two cushions and following rings until (5) and (6). The level sets at (5) and (6) then split into two leaves respectively, which correspond to four sticks in Figure 3.4 [24].

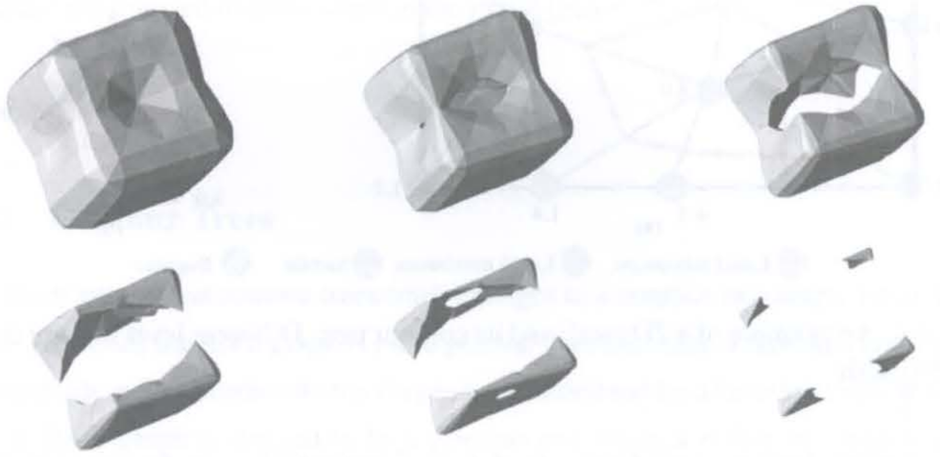


Figure 3.4: Sample data set and its Level sets of as the value of function increases [24].

From the evolution of the level sets described above, the contour tree can be described as recording what happens to components of the level set in response to certain *events* that correspond to the critical points, if we continue to think of the parameter values as time [24]. A component is created either by appearing, separated from all existing components, or by an existing component splitting to become two or more new components. Similarly, a component is destroyed either by collapsing down to a single point and disappearing, or by joining with another component to make a new, combined component.

Because one certain event occurs only at critical points, Carr et al. [24] limited events of level sets to limited cases when the isovalue h changes and equal to the value of a critical point:

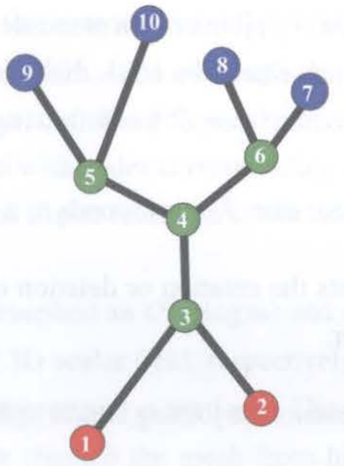


Figure 3.5: Contour tree for Figure 3.4.

1) A new component is created at a local minimum;

2) An existing component is destroyed at a local maximum;

3) Two or more existing components are joined into a new component at a saddle point;

4) The topological genus of an existing component is changed at a saddle point [82];

5) An existing component is split into two or more new components at a saddle point;

6) Any combination of 3) – 5). Both splits and joins can occur at a highly degenerate multi-saddle.

If types of events can be identified, then the contour tree can be constructed by a sweep through changing isovalue h. Each component of the level set is created at a critical point of type 1), 3), or 5), and is destroyed at a critical point of type 2), 3), or 5). Such a critical point is called a *node*. For each component, the node where it is created and the one where it is deleted are connected by an edge called an *arc*. The components then have a one-to-one relationship with the arcs. Through sweeping the isovalue h from f_{min} to f_{max} following previous steps, the contour tree is then constructed.

In summary, the contour tree is a data structure that captures the topological characteristics of a scalar field [117]. As h increases in the level set of $L(h) = \{(x) \mid f(x) = h\}$, contours appear at local minima of f, join or split at saddles, and disappear at local maxima of f. If each contour is represented as a node, the evolution of the level set forms a tree called contour tree. The contour tree $\mathscr{T}$ has following characteristics:

- Each node v in the contour tree $\mathscr{T}$ corresponds to a critical point.
- Each leaf node represents the creation or deletion of a component at a local extremum of the parameter.
- Each interior node represents the joining and/or splitting of two or more components at saddle points.
- Each arc represents a component in the level sets for all values of the parameter between the values of the data points at each end of the arc.
- Every cut on the arc $(v_i, v_j) \in \mathscr{T}$ by an isovalue $v_i \leq h \leq v_j$ represents a connect component (contour) of the level set $L(h)$. The number of cuts of $\mathscr{T}$ at the value h is equal to the number of connected components for the level set $L(h)$.

3.3.2 Previous Work

The contour tree was introduced by Boyell and Ruston [13], as a summary of the evolution of contours on a map (i.e. in 2D), and was used by Freeman and Morse [44] to find terrain profiles in a contour map. van Kreveld et al. [117] used the contour tree structure to compute isolines on terrain maps in geographic information systems. With terrain maps, a surface model is computed from elevation values at sample points in the plane. Isolines, often called contours, are the curves consisting of points at a given height that can be seen on any topographic map. Contours can be traced from a surface model relatively easily, given a starting point, or seed on each. van Kreveld et al. used the contour tree to generate "seed sets" for any query height value. Prior to 1995, the contour tree was typically constructed from previously extracted polygonal contour lines.

Takeshima et al. [108] and Takahashi et al. [106, 107] gave the first algorithm for computing the contour tree for a triangulation in two dimensions. This algorithm traces ascending and descending paths from saddles in the mesh to form a surface network, connecting all saddles and local extrema in the mesh. Local extrema in the mesh are identified, and transferred to the contour tree. This algorithm was extended to three dimensions by Takahashi et al. [105, 106], although the contour tree is referred to in this case as the *volume skeleton tree*. The volume skeleton tree is similar to the contour tree except that it is augmented with nodes corresponding to genus changes. It was used to design transfer functions and depict inclusion relationship in volume rendering [106, 107].

van Kreveld et al. [117] described an $O(m \log m)$ and $O(m^2)$ algorithm to construct a contour tree from a 2D and 3D scalar field, respectively. The function is defined on a simplicial mesh with m elements and n vertices. The contour tree is computed by sweeping a polygonal contour through the mesh from high to low, then from low to high. They also noted that the contour tree could be used to extract isosurfaces from volume data. Multiple saddles and boundary cases required special handling during computing.

Tarasov and Vyalyi [109] improved the time complexity of van Kreveld et al.'s work to $O(m \log m)$ in the 3D case, by subdividing simplices into as many as 576 cells each. Pascucci [81, 82] further extended this algorithm to track topological genus, as well as connectivity.

Carr et al. [24] simplified Tarasov and Vyalyi's algorithm to construct the contour tree in all dimensions. The join tree and split tree are constructed and merged to build the contour tree in $o(m + n \log n)$. Pascucci and Cole-McLaughlin [82] computed Betti numbers of contours to distinguish different topology of contours within an edge of the contour tree. The divide-and-conquer approach [82] allows output-sensitive construction of contour trees and easy extension to parallel implementation. They also adapted the algorithm to handle cubic cells with a trilinear interpolation function [82, 83].

Carr and Snoeyink [23] used the contour tree as an interface to display topological structures of isosurfaces and segment individual contours in a scalar field. They computed path seeds for each edge, which generate a seed cell necessary for rapid extraction of a selected contour in runtime. The contour tree evolves as the function changes over time. Edelsbrunner et al. [38] combined the evolving sequence of contour trees defined from continuous space-time data into a single data structure. However, utilization of this data structure in visualization applications is not addressed [100].

More recently, Takahashi et al. [102] introduced manifold learning into the contour tree construction. It approximates contour trees from a set of scattered samples embedded in the high-dimensional space. The contour tree is extracted as a projection of point clouds into 3D space. The constructed contour tree is then used in volume rendering.

The contour tree used in this thesis is constructed based on Carr et al. [24] and Pascucci et al. [83, 84]'s methods.

3.3.3 Extraction of Critical Points

Critical points are crucial to create the contour tree of data sets. Roughly speaking, a critical point is defined to be a point that activates a topological change in contour evolution such as creation, merging, splitting and deletion of contours. More formally, taken a 3-manifold as an example, a critical point p of a function $f(x,y,z)$ is defined to be a point that satisfies:

$$\frac{\partial f}{\partial x}(p) = \frac{\partial f}{\partial y}(p) = \frac{\partial f}{\partial z}(p) = 0. \tag{3.2}$$

In order to extract critical points, different interpolation methods are used, for example, linear interpolation through tetrahedralization and trilinear interpolation through hexhedralization. The method of linear interpolation of a volume data set through tetrahedralization can uniquely determine the isosurface evolution as the scalar field value decreases. In the tetrahedralization, each voxel has its neighboring voxels that constitute a triangulated sphere surrounding the target voxel itself. By comparing scalar field values of neighboring voxels on the sphere with that of the target, the algorithm partitions the surrounding sphere into two types of connected regions: plus regions that include points having larger scalar field values than the target, and minus regions having smaller scalar field values. By identifying the configuration of these plus and minus regions on the sphere, we can extract critical points [107]. Hexahedral with trilinear interpolation [83] is also used to identify critical points. We use trilinear interpolation for a hexahedral cell in our pipeline, because hexahedral meshes are most commonly used for direct volume rendering of data given as samples on regular, rectilinear meshes.

3.3.4 Computing Contour Trees

As mentioned in Subsection 3.3.2, algorithms for computing contour tree have been widely investigated [104, 55, 117, 24, 83] for tetrahedral meshes using linear interpolation and hexahedral meshes using trilinear interpolation. Specially, Carr et al. [24]

presented an efficient scheme to compute a contour tree in $O(m+n\log n)$ time complexity in any number of dimensions, where m is the number of simplices, and n is the number of vertices. This algorithm sets up the Contour Tree (CTr) through constructing two graphs individually, which are the *Join Tree* (JT) that represents the appearance and merging of isosurface components, and *Split Tree* (ST) that represents the disappearance and splitting of isosurface components. Then the JT and ST are merged to form the contour tree. Pascucci and Cole-Mclaughlin [82, 83] adapted the algorithm to handle cubic cells with a trilinear interpolation function. This algorithm is divided into three stages [82, 83]:

- Sorting of the vertices in the field;
- Computing the Join Tree JT and Split Tree ST;
- Merging the JT with the ST to build the CTr.

Each of the stage is covered in following subsections based on [82, 83].

Sorting Vertices

The vertices of the mesh are ordered by increasing function value in $O(n\log n)$ time using any standard sorting technique. It is important to note that the remainder of the algorithm relies on the assumption that there are no two vertices with the same function value. Typical input fields do not satisfy this assumption, therefore Pascucci and Cole-Mclaughlin [82] imposed a symbolic perturbation of the function values by replacing the test $f(v_i) \overset{?}{<} f(v_j)$ with the test $i \overset{?}{<} j$. After sorting, this integer comparison solves consistently the ties when $f(v_i) = f(v_j)$. In the following the symbol i is also used for the node of CTr, JT or ST that corresponds to v_i.

Computing the JT and the ST

The computation of the JT and ST is performed in two sweeps through the data in forward and reverse vertex order [82]. The sweep to compute the JT is conceptually simple: a join occurs when a vertex v has two higher-isovalued neighbors that belong to different connected components of $x : f(x) > f(v)$. Algorithm 1 shows the computation of the JT. The JT is built incrementally with a tree data-structure supporting the obvious functions `NewTree()`, `AddNode`(XT,i) and `AddArc`(XT,i,j). Implicitly the JT

tracks the history of the Union operations of a UnionFind [110] data structure over the set of vertices in the mesh with respectively increasing and decreasing function value. NewSet(UF, i) creates the new set $\{i\}$, with reference node i. If k belongs to the set i then Find(UF, k) returns i in constant time. Union(UF, i, j) redirects the pointers of all the elements in j to pointer to i, if i has larger cardinality than j (vice versa if $|i| < |j|$).The Boolean function IsMin$(\mathscr{F}, v_i)$ returns true if v_i is a local minimum in $\mathscr{F}$ [82]. Each vertex v_i is associated with two lists UpAdj, of incident edges (v_i, v_j) with $j > i$, and DownAdj of incident edges (v_i, v_j) with $j < i$. In this way, IsMin$(\mathscr{F}, v_i)$ can test in constant time if i is a minimum (DownAdj is empty) and the loop on line 7 directly scans the elements of DownAdj.

Algorithm 1: Computing the JoinTree().

```
input : vertices, edges
output: $JT$
1  $JT$ = NewTree ();
2  $UF$ = NewUF ();
3  for $i = 0$ to $n-1$ do
4      AddNode ($JT$, $i$);
5      if IsMin ($\mathscr{F}, v_i$) then
6          NewSet ($UF$, $i$);
7      for each edge $v_i v_j$ with $j < i$ do
8          $i' \leftarrow$ Find ($UF$, $i$);
9          $j' \leftarrow$ Find ($UF$, $j$);
10         if $i' \neq j'$ then
11             AddArc($JT$, $i'$, $j'$);
12         Union ($UF$, $i'$, $j'$);
13 return $JT$;
```

The routine SplitTree has the same structure as JoinTree. The only differences are as follows: (a) the main loop (line 3) in Algorithm 1 would scan the vertices in reverse order, (b) the **if** statement in line 5 would test IsMax instead of IsMin, and (c) the inner loop (line 7) would consider the edges (v_i, v_j) with $j > i$ [82].

Merging the JT with the ST

The third stage of the contour tree algorithm merges the join tree JT and split tree ST to obtain the contour tree CTr. Algorithm 2 illustrates the process of merging the

JT with ST to form the CTr. During this process, the upper leaves of the JT and the lower leaves of the ST are successively removed from both trees and added to the CTr. Consequently the data structure representing the JT and the ST supports the additional operations `DelNode`(XT,i) and `Leaf`(XT,i). `DelNode`(XT,i) removes the node i from XT while maintaining the consistency of XT by removing any arc ij and replacing any pair of arcs ij, ik with the arc jk. The Boolean function `Leaf`(XT,i) tests whether the node i is a leaf of XT. More specifically `Leaf`(JT,i) is true if the JT has no arc ij with $j < i$, and `Leaf`(ST,i) is true if the ST has no arc ij with $j > i$. `GetAdj`(XT,i) returns a vertex j if XT contains the arc ij. A queue data structure is used to store pairs $[NodeName, TreeName]$ and is managed with the functions `NewQ()` (to create a queue), `Get`(Q) (to get a pair from the queue Q) and `Put`$(Q, [i,XT])$ (to add a pair to Q) [82].

Algorithm 2: Computing the ContourTree().

```
input : JT, ST
output: $CTr$
1  $Q \leftarrow$ NewQ ();
2  $CTr \leftarrow$ NewTree ();
3  for $i = 0$ to $n-1$ do
4      AddNode ($CTr$, $i$);
5      if Leaf ($JT,i$) then
6          Put ($Q$, $[i,JT]$);
7      if Leaf ($ST,i$) then
8          Put ($Q$, $[i,ST]$);
9      while $[i,XT] \leftarrow$ Get ($Q$) do
10         $j \leftarrow$ GetAdj ($XT$, $i$);
11         DelNode($ST,i$);
12         DelNode($JT,i$);
13         AddArc ($CTr, ij$);
14         if Leaf ($XT,j$) then
15             Put($Q,[j,XT]$);
16 return $CTr$;
```

The size of the CTr can be minimized by deleting any node that has exactly degree two with `DelNode`. This reduction to a minimal CTr can be done directly during the construction of the JT and ST. This makes the algorithm slightly more complicated but has the advantage of reducing the size of the intermediate storage [82].

3.3.5 Multi-Resolution Contour Trees

As mentioned above, the contour tree of a scalar field is a graph obtained by contracting all the connected components of the level sets of the field into points. It has proven effective as a data structure for topological representation of data sets and as a user interface component guiding interactive data exploration sessions. In practice, these uses have been very limited due to the problem of presenting a graph that may be overwhelming in size. Topological simplification techniques help in relieving this problem by reducing the size of the graph. Topological simplification will be covered in later chapters. This subsection introduces a different data structure that is used to present contour trees: Pascucci et al. [84] introduced a multi-resolution data structure for representing contour trees and an algorithm for its construction. The hierarchical layout of contour trees allows coarse-to-fine rendering of the tree.

Hierarchical Tree Representation

Typically contour trees are represented as a list of nodes and a list of arcs, where each arc is defined as a node pair. Pascucci et al. [84] used an alternative — *branch decomposition*, where a *branch* is defined as a monotone path in the graph traversing a sequence of nodes with non-decreasing (or non-increasing) value of f. The first and last nodes in the sequence are called the endpoints of the branch. All other nodes are said to be interior to the branch. A set of branches is called a branch decomposition of a graph if every arc the graph appears in exactly one branch of the set. The standard representation of a graph satisfies this definition, where every branch is a single arc.

A hierarchical decomposition of a contour tree is constructed with branches: the endpoints of each branch (except the root) represent a saddle-extremum pair that form an atomic component. Figure 3.6 illustrates an example of hierarchical decomposition of a contour tree. In this figure, the root branch B_0 connects the two global extrema. The branches B_2 and B_3 pair two maxima with split saddles and can be canceled independently. B_1 pairs a minimum with a join saddle and cannot be canceled before B_3 because of their parent-child relation. This simplification process defines a hierarchy of cancelations where a branch B_1 is said to be the parent of branch B_3 if one endpoint of B_3 is interior to B_1. The root branch has no parent and cannot be simplified. Removal of a parent before one of its children disconnects the tree.

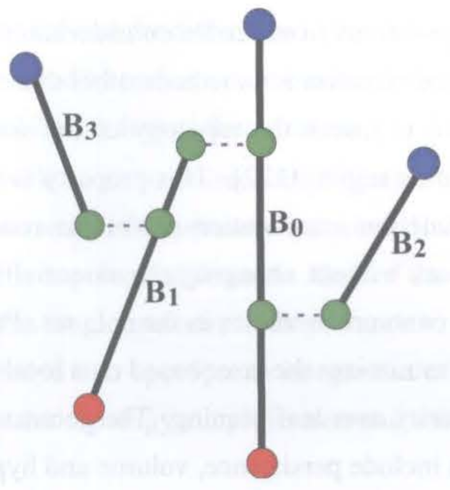


Figure 3.6: Hierarchical decomposition of a contour tree.

Hierarchical Contour Trees

Single resolution algorithm for computing a contour tree is described in the previous sections [24, 82], which builds the contour tree through computing the join tree and split tree. The multi-resolution contour tree is built using a similar approach but building directly a hierarchical decomposition of the contour tree, and based on branches. The resulting trees of the join tree and split tree are stored as branch decompositions. The leaves of the *JT* and *ST* are stored in a priority queue, which always provides access to the leaf branch with the lowest priority. The priority used in [84] is the length of a branch, it could also be volume, hypervolume [25], or their combinations [128]. The tree can be simplified by removing a branch that does not disconnect the tree. Through changing the threshold of the priority of branches, the contour tree can be presented in multi-resolutions.

3.4 Existing Contour Tree Simplification Solutions

Topology simplification suppresses insignificant features by removing, or canceling, pairs of critical points that are considered unimportant according to a specified measure. Carr et al. [25, 26] simplified the contour tree with two basic operations: leaf pruning and node reduction. The scheme involves computation of several metrics that are used for ranking the "importance" of an arc before pruning. Leaf pruning removes a leaf and the arc incident to the leaf from the contour tree. Removing an arc from the contour tree discards the corresponding contours from further consideration in the rendering process. Carr et al. showed that when visualization methods other than isosurface extraction are used, data can be modified to match the topology of the simplified contour tree by "flattening" the corresponding region [122]. This property is useful when the contour tree is used for transfer function specification in volume rendering. Node reduction removes degree-two vertices without changing the essential structure of the contour tree. It does not affect the contours or values in the data set. Pruning and reduction are performed in an order that minimizes the error based on a local geometric measure with node reduction having priority over leaf pruning. The geometric measures used in the contour tree simplification include persistence, volume and hypervolume.

Pascucci et al. [84] presented a multi-resolution data structure for representing contour trees and a method for its construction (also see Section 3.3.5). The multi-resolution contour tree is computed directly from join and split trees and this guarantees that atomic simplification steps of the tree correspond to atomic reduction of proper pairs of critical points. The multi-resolution data structure uses branch decomposition, an efficient way for storing a hierarchy of contour tree simplifications. A priority queue is used to store the leaf branches of the join tree and split tree during construction of the multi-resolution contour tree. The priority for each branch in the scheme is the persistence (length) of the branch. The priority of a branch shows importance of the branch. A branch is defined by a pair of critical points: a saddle and an extremum that are connected by a monotone path. Each saddle-extremum pair corresponds to a topological simplification, or cancellation, of critical points. Pascucci et al. [85] also used persistence based simplification to eliminate insignificant saddle-extremum pairs from the Reeb graph.

Takahashi et al. [106] simplified the contour tree through computing the difference in scalar field between two nodes of different patterns, and then selecting one pattern to be discarded by finding the pair of nodes having the smallest difference. The number of simplification steps is controlled by a threshold that limits the acceptable difference. This simplification process is done one by one until no pattern can satisfy the given threshold. This process actually replaces three edges at a saddle point with a single new edge, based on the height of the edge [25]. Takahashi et al. [105] simplified the contour tree by pruning leaves to determine the "most important" isovalues for transfer functions. In choosing which leaves to prune, the authors define a new weight value. The new weight value is the product of the volume swept by the isosurface component on the subtree that is discarded and the difference in the scalar field between critical end points of the subtree. It is used as an importance measure to simplify the contour tree. Saddles are processed until only a few of them remain.

In a word, previous work on contour tree simplification is based on importance of arcs/branches. Various measures of importance are used. Each measure has its advantages in identifying specific topological features. One of the common points of previous work is that they utilize single measure of importance in a topology simplification pipeline. We strongly believe that appropriate integration of various measures of importance will produce a better topology simplification and will be covered in Chapter 4.

3.5 Contour Tree-Defined Volume Segmentation

Since the concept of flexible isosurface [24, 25] supports independent manipulation of single contours, Takahashi et al. [103] introduced contour tree-based volume segmentation in volume rendering. An individual contour is mapped to a point in the contour tree. Similarly, each arc of the contour tree represents the union of all contours which are mapped to points on the arc. This union can be thought of as the volume being swept out by the contour as its isovalue is varied, starting at the critical value which creates the contour, and ending at the value which destroys it. This sweep defines a partition of the space into topologically distinct regions, which are referred to as topological zones. Weber et al. [122] extended this idea in volume rendering to index individual subregions by branches of the contour tree and apply transfer functions to subregions independently.

Figure 3.7 [122] is an example of a segmentation defined by a contour tree, where (a) is a terrain data set showing topological zone segmentation, (b) is the contour tree of the terrain with color-encoded edges corresponding to topological zones in (a), (c) is the terrain data set showing topological zone segmentation for branch decomposition, and (d) is branch decomposition of contour tree of the terrain with color-encoded edges corresponding to topological zones in (c).

From this example, we see that the branch decomposition segments and indexes each subregion of the data set topologically. Weber et al. [122] used this idea in volume rendering and proposed a volume rendering framework which classifies volume data based on the contour tree and assigns unique transfer function to each subregion

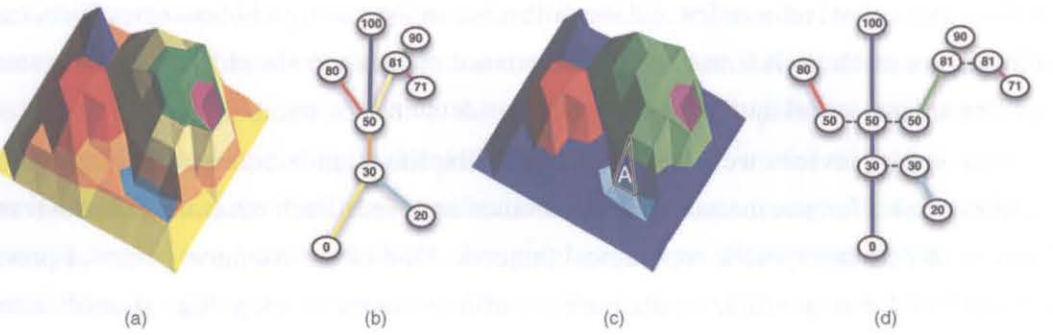


Figure 3.7: Example of a segmentation defined by a contour tree [122].

corresponding to a branch of the contour tree.

Compared with the general contour tree, the advantages of the branch decomposition include:

- Provide a hierarchical representation of the contour tree. A hierarchical multi-resolution representation of the contour tree allows linear time access simplified representations of the topology.

- Avoid an "over-segmentation" of the volume [122]. While having a topological zone per contour family with equivalent topology is useful, it does not take into account that a segmentation is only necessary if regions overlap in value range. The branch decomposition naturally concatenates a sequence of arcs into a single unit, see Figure 3.7(d), unifying their topological zones into a single zone, see Figure 3.7(c). Furthermore, branch decomposition permits user interactions on a coarse, simplified tree and propagating transfer functions down to the full resolution tree.

- There are less branches in the branch decomposition than arcs in the general contour tree.

This thesis uses branch decomposition to index subregions of volume data set and automatically generate transfer functions for various subregions, which is covered in Chapter 6.

3.6 Contour Trees in Visualization

As a data structure to record topological events, the contour tree has been used to fast extract isosurfaces [117, 24], to guide mesh simplification [29], to find important isovalues for transfer function generations [106, 107], to compute topological parameters of isosurfaces [59], as an abstract representation of scalar fields [9], and to manipulate individual contours [24]. This section mainly reviews applications of the contour tree in visualization, especially in providing topological information of a scalar field, isosurface extraction and simplification, depiction of inclusion relationships, and transfer function design in volume rendering.

3.6.1 Topological Representation of Scalar Field

The purpose of the visualization is to aid the user in understanding the structure of the data [115]. Common methods for visualizing scalar fields detect structures and present a display to users which communicates these structures. This makes users infer the global scalar structure from what is frequently an insufficient display of information. Bajaj et al. [9] introduced an interface called the *contour spectrum*, where properties such as isosurface area, enclosed volume and the contour tree were plotted alongside isosurfaces in order to provide users with additional cues. This kind of presentation enforces the topological information of a scalar field. Bajaj and Pascucci [10] also presented isocontours of a scalar field along with the critical points and topology graph. Contours are used to display the exact shape of an object in a scalar field, while the topology graph attempts to show the relations among all such objects in the field. It does not give the details of shapes of particular objects. The topological information serves to both provide information which is not available in commonly used scalar visualization techniques, and enhance the information provided by common visualization techniques.

3.6.2 Isosurface Extraction and Simplification

van Kreveld et al. [117] used the contour tree to obtain seed sets for isosurface traversal. The size of the seed sets is provably small. This kind of seed sets is efficient for the isosurface traversal. Carr and Snoeyink [24] extended the minimal seed sets of van Kreveld et al. [117] with path seeds. The path seeds generate paths based on the contour tree by associating seed cells with individual contours, using the contour tree as a visual index to the contours. This association underpins the *flexible isosurface* interface in which individual contours are treated as distinct entities. Since this concept supports independent manipulation of single contours, it imports a notion of spatial locality to the task of extracting surfaces from a scalar field. Individual contours can be deleted (allowing a user to view otherwise obscured portions of an isosurface), rendered in different colors, or evolved to new isovalues without affecting other contours. As a motivating example, the authors showed a CT data set of a head, and used the contour tree to display a contour representing the brain without displaying the contour representing the skull at the same isovalue [122]. Carr et al. [25] used local geometric measures to simplify flexible isosurfaces based on the contour tree. Local geometric measures such as surface area and contained volume are defined for individual contours. These local geometric measures are then used to simplify the contour trees, suppressing minor topological features of the data. A flexible isosurface interface is then combined with these measures to explore individual contours of a data set.

As mentioned in Section 3.3.5, Pascucci et al. [84] presented a multi-resolution data structure, branch decomposition, to represent contour trees and an algorithm for its construction. This data structure provides a hierarchical layout that allows coarse-to-fine rendering of the tree in a progressive user interface. The multi-resolution contour tree is computed directly from join and split trees and this guarantees that atomic simplification steps of the tree correspond to atomic reduction of proper pairs of critical points. A priority queue is used to store the leaf branches of the join tree and split tree during construction of the multi-resolution contour tree. A branch is defined by a pair of critical points: a saddle and an extremum that are connected by a monotone path. Each saddle-extremum pair corresponds to a topological simplification, or cancelation, of critical points. Pruning a branch from the branch decomposition is equivalent to performing a vertex pruning operation in the scheme of Carr et al. [25, 122].

More details of the contour tree simplification is covered in Chapter 4. This thesis sets up the contour tree simplification pipeline based on the branch decomposition. However, we extend measures of importance and propose a combined approach to simplify the contour tree in order to make full use of the advantages of different measures of importance at the same time.

3.6.3 Depiction of Inclusion Relationship

Boyell and Ruston [13] created the contour tree to represent the nesting relationship of a set of polygonal contours manually converted from a topographic map. An "outside" region is designated that encloses everything, and nodes representing contours are added to the tree one at a time. Because there is a well-defined outside region, each contour has a distinct inside and outside, and the node for each contour is connected to the node for the contour immediately outside it. But no details were provided on the mechanics of the construction, which may have been manual [22].

Takahashi et al. [107] presented an algorithm to extract view-independent nested structures, i.e., isosurface inclusion relationships, without multidirectional ray intersection tests. The method distinguishes the specific type of topological transition in isosurfaces that yields inclusion relations between their connected components. They classified isosurface transitions at saddles into four types as shown in Figure 3.8. In these isosurface transition types, only the type (b) in Figure 3.8 introduces isosurface inclusion relationships. Here, a new inclusion relationship appears when the saddle is C_1 (from right to left), and an existing inclusion relationship dissolves when the saddle is C_2 (from left to right) when decreasing the corresponding scalar field value.

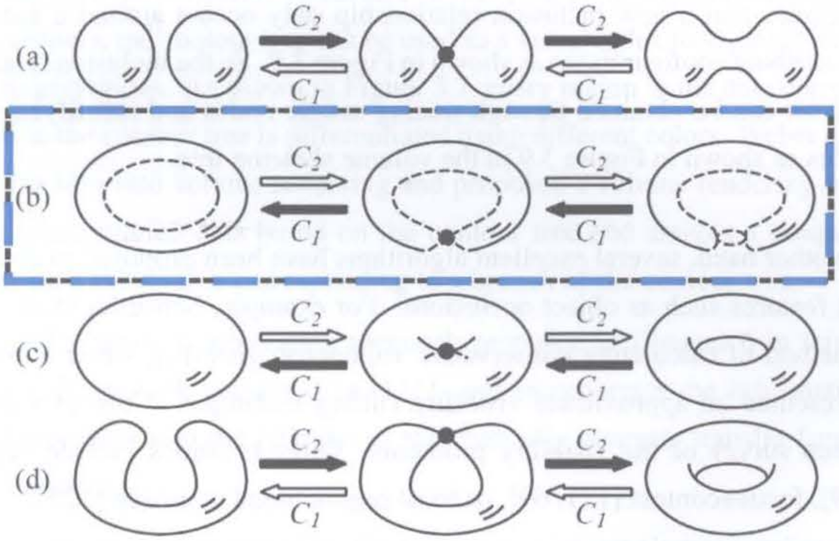


Figure 3.8: Classification of isosurface transitions at saddles depending on embedding in 3D space (C_1 and C_2 are saddle types) [107].

Takahashi et al. classified evolving isosurfaces into two categories: solid isosurfaces

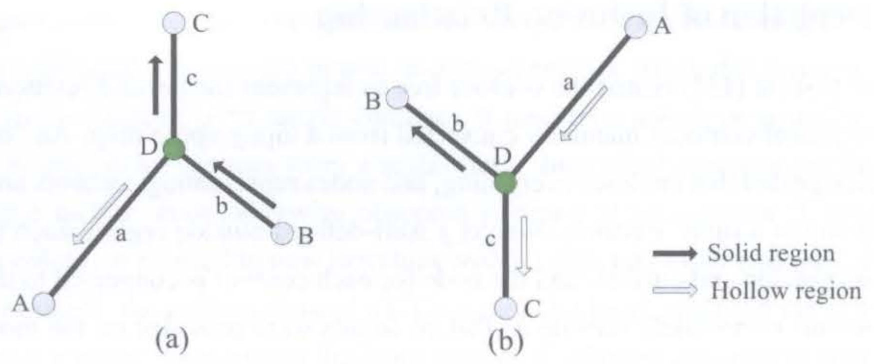


Figure 3.9: A new inclusion relationship appears at a saddle shown in (a), and an existing inclusion relationship dissolves at a saddle shown in (b).

where their interior samples are larger in the scalar field than those on the corresponding isosurfaces, and hollow isosurfaces where the interior is smaller. This implies that solid isosurfaces always expand as the scalar field value decreases while hollow isosurfaces always shrink. In order to extract the inclusion relationships, each link incident to a saddle node in volume skeleton tree in [107] is then identified as solid or hollow region. [107] concludes that a new inclusion relationship only occurs around a saddle node which has attribute configurations as shown in Figure 3.9. So the inclusion relationships in the data set can be obtained through tracing saddle nodes and identifying types of saddle nodes as shown in Figure 3.9 in the volume skeleton tree.

On the other hand, several excellent algorithms have been proposed to detect view-dependent features such as object occlusions. For example, Schaufler et al. [94] proposed a method of calculating conservative volumetric visibility, while Klosowski et al. [62] presented an approximate visibility culling technique. Cohen-Or et al. [30] made a deep survey of the visibility problems. Other solutions include depth peeling [12, 19], focus+context [121, 69], or focal region based approach [127].

The inclusion relationship analysis in this thesis uses the results from Takahashi et al. [107] and overcomes disadvantages of inclusion relationship analysis based on the general contour tree. This thesis also analyzes other topological relationships, e.g. neighboring relationship, based on the contour tree.

3.6.4 Transfer Function Design in Volume Rendering

A transfer function is used to classify scalar samples in a data set. It is one of the important techniques in volume rendering. Original approaches for volume rendering consider the scalar value and its derivatives as parameters to classify samples [71, 64], while still applying the same transfer function uniformly throughout the domain.

Takeshima et al. [108] and Takahashi et al. [106, 107] described how to automate transfer function design by using the volume skeleton tree to detect isovalues where major changes in isosurface topology occurred, then emphasizing those isovalues. Takeshima et al. [108] also derived isosurface-trajectory distance, which describes closeness between any two of the isosurface components, from the volume skeleton tree. The isosurface-trajectory distance is then used as the second dimension of opacity transfer function besides the scalar value as the first dimension. This transfer function is used to trail symmetric isosurface trajectories. Takeshima et al. [108] and Takahashi et al. [106, 107] employed the inclusion information encoded in the volume skeleton tree and add another second dimension of "inclusion level" into the opacity transfer function to depict inclusion relationship in volume rendering.

Since the concept of flexible isosurface [24, 25] supports independent manipulation of single contours, the contour tree can be used as a visual index to segment data set into different zones/regions. As shown in Figure 3.7, every region in the data corresponding to a branch in the contour tree is differentiated using different colors. Weber et al. [122] extended this idea into volume rendering and proposed a volume rendering framework which classifies volume data based on the contour tree and assigns a unique transfer function to each subvolume corresponding to a branch of the contour tree.

The transfer function generation approach proposed in Chapter 6 in this thesis is based on the framework presented in [122], and incorporates the information of inclusion relationships into the pipeline to automatically generate transfer functions for various subregions.

In summary, the contour tree plays the following roles in volume visualization. First, it provides topological attributes of a scalar field, which are not directly obtained from conventional rendering techniques. Second, the contour tree generates a minimal seed set for efficient isosurface extraction and simplification. Third, it encodes inclusion relationships in a data set and guides users to depict inclusion relationships in volume rendering. Forth, it provides a user interface to segment and render each individual connected component during data analysis. This property is significant in transfer function generations in volume rendering.

3.7 Summary

This chapter provided fundamentals of the contour tree, giving preliminaries used to define the contour tree. Three typical topological abstraction techniques were investigated, and particularly focused on the data structure of the contour tree. The state of the art of research on contour trees were reported and algorithms for computing contour trees were reviewed. Typical applications of contour trees in visualization were also presented in order to demonstrate potentials of contour trees in volume rendering.

Chapter 4

Multiple Measures of Importance for Contour Tree Simplification

As presented in Chapter 3, contour trees have wide applications in visualization from representing topology of scalar fields to defining transfer functions in volume rendering. However, real-world data sets produce unmanageably large contour trees because of noise or artifacts during the acquisition process. This makes contour trees be unpractical and limits their uses in data analysis. Contour Tree Simplification (CTS) is often used as a necessary step to remove branches with small importance values in contour trees, and maintain essential structure of data.

This chapter compares previous contour tree simplification approaches, and proposes an importance-driven CTS approach. The proposed approach combines multiple measures of importance through the introduction of various concepts to maximize the advantages of each measure of importance. In the attribute space, various attributes (measures) of a branch are organized in a single space. The concept of the importance triangle is proposed to evaluate the importance of a branch by size of the importance triangle in the attribute space. It considers the whole attribute space and gives a better evaluation of the importance of branch, instead of limited space used by conventional approaches during the CTS. Finally, importance of branches is compared in the importance space during the CTS. The contributions of this chapter are as follows:

- A concept of attribute space is proposed to organize various attributes of a branch in a single space. As a result, the importance of various branches can be compared in an importance space during the CTS.

- A concept of importance triangle is proposed to evaluate the importance of branches, which can make full use of advantages of multiple measures simultaneously.
- A single simplification threshold considers multiple measures of importance simultaneously and allow users to manipulate thresholds more meaningfully and efficiently during the CTS process.

4.1 Introduction

Topology has been an important tool for analyzing scalar data and flow fields in visualization. As one of the topological abstractions of a scalar field, the contour tree represents the nesting relationships of connected components of isosurfaces. It allows a user to highlight the direct correlation of object parts (e.g. anatomic parts in medical data) with single branches in the contour tree. It has wide applications as reviewed in Section 3.6 in visualization and other fields.

However, the contour tree is vulnerable to noise and artifacts in the input data. Many real-world data sets produce unmanageably large contour trees because of noise and artifacts from a data acquisition process. Small details, in particular noise, cause the contour tree size to increase. As a result, the contour tree for a large data set can have millions of edges. This makes it difficult to recognize edges that correspond to objects of interest. The edges of interest may be suppressed. This results in the contour tree being impractical as an abstraction of acquired data, either for automatic processing, or for direct human-guided visualization. The CTS would remove branches that are unimportant, while making the size of tree small enough for user interaction and maintain the essential structure of the data. The CTS performs two operations [25, 26]: leaf pruning and node reduction. Leaf pruning removes topology (removes "structure" resulting in changing its topology) from the field (and the tree) by selecting a leaf node of low importance and removing it. While node reduction eliminates connectivity between regular points from the contour tree, leaving the topology unchanged.

Existing solutions of the CTS are investigated in Section 3.4. The CTS often uses a measure of importance to evaluate various branches in the contour tree and then simplify it. There are various measures and types of importance for a given branch in the contour tree. The selection and use of appropriate different measures of importance is critical for the effective contour tree simplification. Despite multiple measures of importance currently existing [84, 25, 26], conventional CTS approaches still have the following problems:

- They often use single measure of importance to evaluate importance of branches. Because various measures of importance emphasize different features of data sets, it is obvious that single measure of importance is not enough in evaluating importance of branches;
- To determine a region that preserved branches located in the attribute space, a user has to estimate multiple linear discriminating functions.

In this chapter, we propose an importance-driven approach for the CTS. The proposed approach uses multiple measures of importance through introducing concepts of attribute space, importance triangle and importance space into the CTS pipeline. It maximizes advantages of each measure of importance. Interfaces are provided to allow users to interact with branches more meaningfully and efficiently. The objective of this chapter is to deliver an improved approach to the use of importance measures, in order to make full use of advantages of multiple measures of importance simultaneously and improve the CTS efficiency. The importance-driven CTS has advantages: it considers the whole attribute space, instead of the limited attribute space used by conventional approaches. This gives a better evaluation of the importance of a branch. The proposed approach can be generalized to process branches with more than three measures.

4.2 A Glance of Topological Simplification

Topological features of a field are characterized by its critical points. However, a typical scalar field has noise or artifacts from acquisition process. Critical point analysis on such a function relies on topological simplification, i.e., the ability to identify which critical points represent actual features, and selectively removing those that do not. To simplify, the topological simplification is applied to a univariate function f as shown in Figure 4.1 [48]. Critical points (maxima and minima) of f partition the domain into monotonic regions. This partition is stored as a graph whose nodes are the critical points of f and edges represent the monotonic curves. Pairs of critical points identify topological features of the function. The size of each feature is defined as the absolute difference in function value between the two critical points and is called the *persistence* of the critical pair. The smaller-sized features are not significant, probably due to noise in the data, and can be removed explicitly to obtain a global view of the function. Removal of critical pairs can be implemented in a purely combinatorial fashion by updating the graph representation of the partition [48].

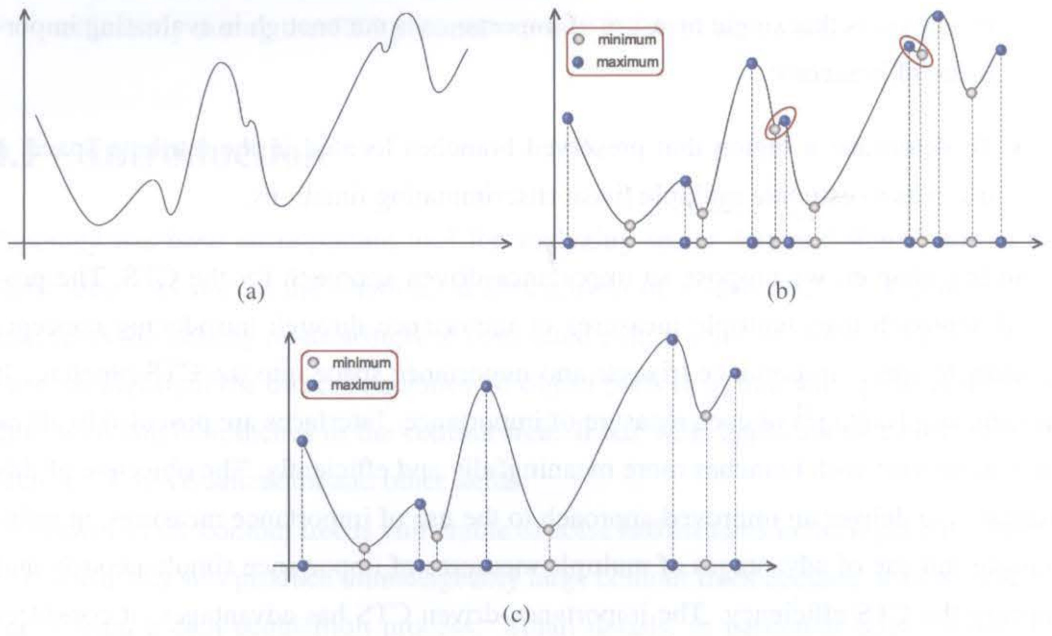


Figure 4.1: Multi-scale analysis of a univariate function. (a) Visualization of the function. (b) Critical points of the function partition the domain into monotonic regions. Pairs of critical points identify features, whose sizes are equal to the difference in function value of the critical points. (c) Small-sized monotonic regions are explicitly identified and removed, leaving behind the "significant" features [48].

Meaningful and important features of a given function are not always captured by the notion of persistence. For example, extrema with function values within a given range may correspond to relevant features, and in this case simplification should leave these extrema unaffected. Therefore, analysis of the critical point pairs and arcs of the topology can lead to better understanding of actual locations of features, and where to apply topological simplification [48].

4.3 Basics of Contour Tree Simplification

To help understand the details of the CTS, this section introduces the data structure of the contour tree and basic simplification operations on the CTS.

4.3.1 Contour Tree Structure

Typically, the contour tree is represented as a list of *nodes* and a list of *arcs*, where each arc is defined as a node pair. The nodes of the contour tree correspond to critical points of a scalar field and are therefore associated with their critical values. Furthermore, nodes that correspond to extrema are leaf nodes, and nodes that correspond to saddle points are related to genus changes. Pascucci et al. [84] used an alternative *branch decomposition* where a *branch* is defined as a monotone path in the graph traversing a sequence of nodes with non-decreasing (or nonincreasing) value of the scalar field. A set of branches is called a branch decomposition of a graph if every arc the graph appears is exactly one branch of the set. Given a scalar field and its contour tree, a graph simplification algorithm is applied to the contour tree. Then this simplification is carried back to simplify the input data. Alternatively, the simplified contour tree and the simplified data can be used to design transfer functions or extract objects.

In this thesis, we assume that each arc or branch of the contour tree has a simplification measure (weight) that is used to indicate the arc/branch's importance. Lower importance arcs/branches are good candidates to be removed during simplification. Based on this idea, we store the following values in the contour tree structure:

- Scalar value of each critical point.
- Arc/branch pair information.
- Arc/branch's basic priority (importance) value, e.g. persistence, volume and hypervolume.

4.3.2 Basic Simplification Operations

The basic simplification operations used in this thesis are based on topology simplification methods introduced by Takahashi et al. [105, 106], Pascucci et al. [84], and Carr et al. [25]. We do similar basic simplification operations as in [25, 84]: leaf pruning and node reduction. Similar to the approach used in [84], we use a priority queue to keep track of the arcs/branches of the tree with their associated priority. The priority of each arc/branch is equal to its importance value. We also use the data structure of the branch decomposition in the contour tree simplification pipeline. The difference between our operations and the operations in [25, 84] is that our operations prune leaves based on combinations of various measures of importance. This is covered in the following sections.

4.4 Concept of Importance

In the contour tree, each arc/branch corresponds to a region in the data domain. The importance of each region can be depicted using different measures. These measures are then used to drive the contour tree simplification process. This section covers detailed definitions of measures of importance.

4.4.1 Definition of Importance

In the Merriam-Webster dictionary [3], importance is defined as follows: Importance means a quality or aspect having great worth or significance. It implies a value judgment of the superior worth or influence of something or someone. It describes the quality (positive or negative) that renders something desirable or valuable, and worthy of note.

In this chapter, we define importance as follows: Importance suggests an evaluation or judgment of significance of an object in a data set. It describes the quality that renders an object desirable or valuable, and worthy of note in visualization. This quality is represented by some measures that evaluate the degree of an object which draws attention to viewers in visualization. In the CTS, importance is a simplification value that indicates the branch's significance. Branches with lower importance are candidates to be removed during the CTS. This chapter focuses on effective uses of measures that are used to evaluate importance of branches.

4.4.2 Evaluation of Importance

Object importance describes priority of each object within the volume data. It is a positive scalar value, which is constant for the whole object. The importance of one object is related to different features of the scalar field depending on applications, such as scalar value, size, position and their combinations. In the contour tree, each arc/branch represents a region or object within the volume data. Arc/branch's importance is related to these factors: volume, hypervolume, position, persistence. In our approach, we focus on following measures based on the data properties:

- Persistence
- Volume
- Hypervolume

The persistence of a pair of critical points that are simplified is equal to the absolute difference in their function values. Volume is the voxel count of the region enclosed by the isosurface, and hypervolume is the integral of the scalar field f over the enclosed region R. Persistence, volume and hypervolume are derived from the dataset itself. Carr et al. [25] and Gyulassy et al. [49, 48] consider that these measures are linear to their relative importance values. We use p, v and hv to represent persistence, volume and hypervolume respectively, and are computed as follows:

$$p = |f(x_{extreme}) - f(x_{saddle})|, x \in R, \tag{4.1}$$

$$v = \int_{x \in R} dx, \tag{4.2}$$

$$hv = \int_{x \in R} f(x)\, dx. \tag{4.3}$$

where f is the scalar field to be analyzed, $x_{extreme}$ and x_{saddle} are extreme and saddle values of the scalar field respectively, R is the region that the current edge represents.

When we think of the scalar value of each voxel as the mass of that voxel, the importance described by hypervolume is based on the mass of the region corresponding to a branch, i.e. what the weight of a branch is. While persistence describes importance based on the number of steps of the sweep for which a feature retains its topological uniqueness, volume describes the importance based on the size of the region corresponding to a branch. Obviously, various measures describe the importance of a branch from different physical aspects.

From the importance's point of view, these measures are different descriptors of importance for a branch. For example, given two branches b_1 and b_2 with same p and v, $f_{max} = 100$, $f_{min} = 2$, $v_1 = v_2 = 50$, $p_1 = p_2 = 100 - 2 = 98$. There are 49 voxels whose scalar value is 100 and 1 voxel whose scalar value is 2 in b_1, while there are 49 voxels whose scalar value is 2 and 1 voxel whose scalar value is 100 in b_2. In this case, we get $hv_1 = 4902$, $hv_2 = 198$. From this example, we see that persistence and volume cannot decide hypervolume uniquely. Hypervolume is an independent measure of importance from the importance's point of view.

Meaningful and important features of a 3D scalar field are not always captured by the notion of persistence. This is also true for volume and hypervolume. For example, an object with small scalar values may have large spatial extent. In this case, if the measures of persistence and hypervolume are used, the branch corresponding to this object may be removed during the contour tree simplification. But when the measure of volume is used, the branch may be preserved. Table 4.1 compares three measures of importance. Through the comparison, we see that each measure of importance cannot uniquely and effectively depict a contour or region. So the contour tree simplification based on a single importance measure is not enough for an effective simplification. Different measures of importance need to be combined together to more accurately depict the importance of objects.

Table 4.1: Comparison of various measures of importance

Measure type	**Persistence**	**Volume**	**Hypervolume**
How to compute	absolute difference of scalar values	voxel count	integral of the scalar field over a region
Typical structure of interest with high values	large difference in intensity relative to surroundings, e.g.skull,vessels	objects with large spatial extent	objects with high intensity and large spatial extent
Unwanted objects with high values	noise, artifacts	objects with large spatial extent and low intensity values	objects with high intensity and small spatial extent, e.g. noise, artifacts
Cases of small values	objects with low intensity values	objects with small spatial extent, e.g. noise, artifacts	objects with small spatial extent or low intensity values
Advantages	highlight high-contrast objects	highlight objects with large spatial extent	preserve small persistence features, eliminate apparent noise and artifacts
Disadvantages	easily to suppress large objects with limited ranges of voxel intensity	easily to suppress small objects with large intensity values	high-intensity noise or artifacts may be preserved as objects

The contributions of different measures of importance for the final importance value are different. The final importance value I is expressed as:

$$I = g(p, v, hv). \tag{4.4}$$

where g is the function used to combine different measures of importance.

4.5 Importance-Driven Contour Tree Simplification

As mentioned, there are various measures of importance for a given branch in the contour tree. The selection and use of appropriate measures of importance is significant for the effective CTS. This section presents an importance-driven approach, which combines multiple measures of importance to make full use of their advantages in the CTS pipeline.

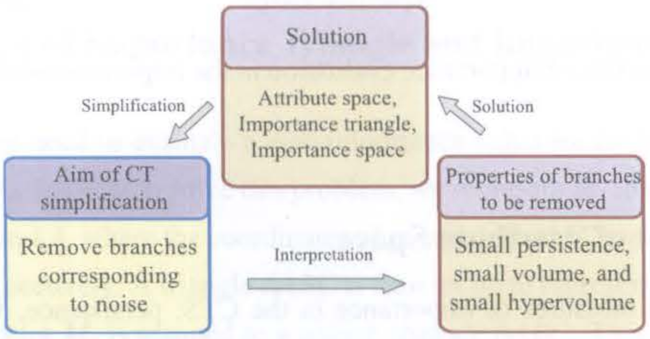

Solution
Attribute space, Importance triangle, Importance space
Simplification
Solution
Aim of CT simplification
Remove branches corresponding to noise
Interpretation
Properties of branches to be removed
Small persistence, small volume, and small hypervolume

Figure 4.2: The principle of the contour tree simplification. Beginning from the aim, the CTS process first gets properties of branches to be removed by interpretation. Based on properties of branches, a solution is proposed. The contour tree is then simplified using the presented solution.

Usually, noise is characterized by its small persistence, small volume and small hypervolume in data sets. Therefore, we aim to remove branches with small persistence, small volume and small hypervolume in the CTS process. The principle of the CTS is shown in Figure 4.2. The concepts of attribute space, importance triangle and importance space are introduced into the pipeline. The concept of attribute space organizes all attributes (measures) of branches in one space, the concept of importance triangle is used to evaluate the importance value of each branch based on multiple measures of importance, and the concept of importance space compares importance of all branches in one space. Figure 4.3 shows the process of importance evaluation in the importance-driven contour tree simplification. Various measures of importance are firstly represented in the attribute space. Then the importance triangle is used to map multiple measures of importance onto one value. The new importance values of branches are compared in the importance space to make simplification decisions. The details of each part will be covered in the later subsections.

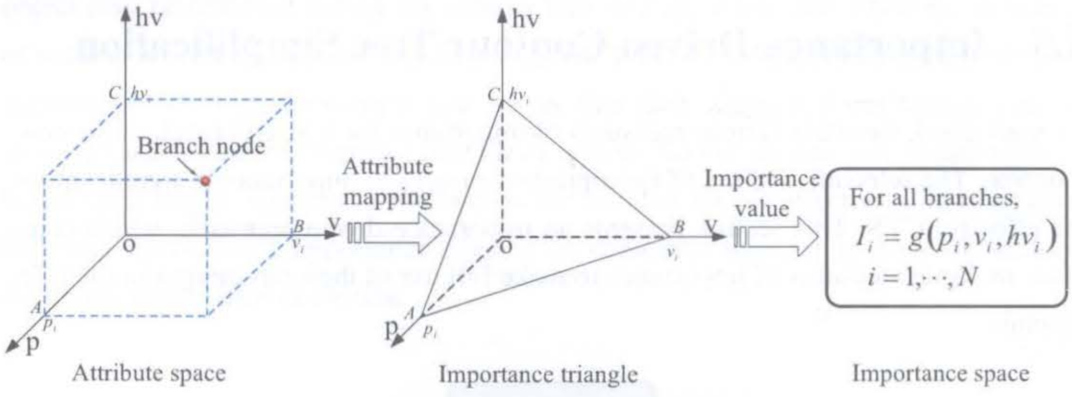


Figure 4.3: The pipeline of importance evaluation in the importance-driven contour tree simplification.

4.5.1 Concept of Attribute Space

We consider three measures of importance in the CTS: persistence, volume and hypervolume. The goal of our approach is to combine three measures of importance to evaluate the importance of a branch, trying to keep advantages and minimize disadvantages of each measure during the CTS. In other words, we need to find a solution to the function g stated in Section 4.4. So this problem can be expressed as follows: suppose that we have N branches in the contour tree, each branch has a property field which stores a 3-dimensional vector representing three importance measures of persistence, volume and hypervolume. These N vectors are represented as $\boldsymbol{M}_i$, $i = 1, ..., N$. We need to find a mapping which maps a 3-dimensional vector $\boldsymbol{M}_i$ of importance measures onto a scalar value. The importance measure vector is represented as:

$$\boldsymbol{M}_i = \begin{bmatrix} p_i \\ v_i \\ hv_i \end{bmatrix} \tag{4.5}$$

In order to allow the use of measures of importance with various units together in a single pipeline, this chapter introduces an abstract space — attribute space — into the CTS pipeline. In the attribute space, each measure of importance is represented with an axis in the 3D Cartesian coordinate system as shown in the left figure of Figure 4.3. Importance values on each axis are represented with a single unit — relative importance. The relative importance is used to show the importance of one branch relative to the branch with the peak importance value in the contour tree. In this way, despite each measure of importance having different unit, the abstract level of relative importance evaluates different measures of importance using a single unit. So the relative importance can be used to evaluate the combination of various measures of importance in one space. In the attribute space, a node is used to represent a branch with specific persistence, volume and hypervolume as shown in Figure 4.3.

4.5.2 Concept of Importance Triangle and Importance Space

As mentioned, we need to evaluate a new importance value for each branch based on components of $\boldsymbol{M}_i$. In order to solve this problem, we represent $\boldsymbol{M}_i$ in the attribute space as shown in Figure 4.4, where the coordinates of points A, B and C are $(p_i, 0, 0)$, $(0, v_i, 0)$ and $(0, 0, hv_i)$ respectively. A triangle ΔABC is then set up to represent the vector $\boldsymbol{M}_i$. In this way, each vector $\boldsymbol{M}_i$ is mapped to a unique triangle ΔABC. This triangle is named *Importance Triangle* (*ITri*).

In the *ITri*, because the volume of the tetrahedra $OABC$ (see Figure 4.4) is proportional to the product of p_i, v_i and hv_i, the final volume would be zero if any one of three measures is zero. If it is used to evaluate the importance of a branch based on multiple measures, the final value of importance of a branch would be zero if any one of its measures is zero. This property does not meet practical requirements on importance evaluation of branches, for example, some branches with two of measures of importance being non-zero may be preserved (i.e., the final importance value should be non-zero). So the concept of importance area is used to evaluate the importance of a branch in the CTS. The importance area is the area of the triangles formed by any two measures of importance as shown in Figure 4.4. Because the area of triangles in the tetrahedron $OABC$ has the relation as shown in Equation 4.6 [43, 88], we use the area

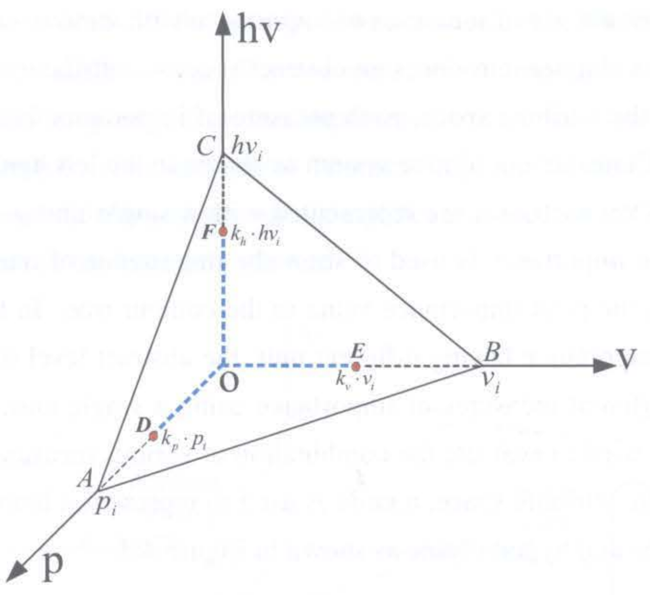


Figure 4.4: Importance triangle.

of the importance triangle ΔABC as the final value of relative importance in the CTS:

$$S^2_{\Delta ABC} = S^2_{\Delta OAB} + S^2_{\Delta OBC} + S^2_{\Delta OAC}, \tag{4.6}$$

where S is the area of various triangles. The size (area) of *ITri* is computed with Equation 4.7:

$$S_i = \frac{1}{2}\left\|\overrightarrow{AB} \times \overrightarrow{AC}\right\| = \frac{1}{2}\sqrt{(hv_i \cdot p_i)^2 + (v_i \cdot p_i)^2 + (hv_i \cdot v_i)^2}, \tag{4.7}$$

$$I_i = S_i, \tag{4.8}$$

where S_i is the area of *ITri* of the ith branch in the contour tree. The final importance value of the ith branch I_i is defined to be equal to S_i as shown in Equation 4.8. Because I_i is based on the size of *ITri*, we call this measure of importance *ITri*. As mentioned, branches with small p_i, v_i and hv_i correspond to noise physically in the data space, where S_i is also small according to Equation 4.7. Therefore, *ITri* provides a method to evaluate importance of branches based on multiple measures of importance. Physically, it focuses on applying small importance values to branches with small p_i, v_i and hv_i, which correspond to noise in data sets and will be removed during the CTS.

From Equation 4.7, we get following conclusions:

- If one of the values of p_i, v_i and hv_i is $\rightarrow 0$ (where "$\rightarrow 0$" means "close to zero"), the final importance value I_i depends on values of other two measures. This implies that even if one measure of importance is $\rightarrow 0$, the branch will still be preserved when the other two measures are large enough during the simplification. However, if a single measure is used in the pipeline, the branch will be removed when the measure is $\rightarrow 0$.

- If two values of p_i, v_i and hv_i are $\rightarrow 0$, the value of I_i is $\rightarrow 0$. There are three special cases for this situation: 1) $p_i = 0$, $v_i = 0$ and $hv_i \neq 0$; 2)$p_i = 0$, $v_i \neq 0$ and $hv_i = 0$; 3) $p_i \neq 0$, $v_i = 0$ and $hv_i = 0$. The first and third case are physically impossible because a region with zero-volume does not exist. In the second case, scalar value of voxels corresponding to the current branch should be zero to meet these conditions. This zero-value region is meaningless physically in most cases in understanding the data set. So we do not need to consider this situation.

- In practical applications, users often have such requirement: the persistence, volume and hypervolume of preserved branches need to be greater than a specific threshold respectively in order to get a meaningful simplification. Equation 4.7 consider three measures simultaneously, this means that the proposed approach directly meets the user requirements. But if a single measure is used in the simplification pipeline, it is not possible to consider multiple measures at the same time.

These properties exhibit advantages compared with conventional CTS approaches using a single measure.

In general, if the ith branch has n measures m_{ij} $(j = 1, ..., n)$ of importance in the contour tree, the importance measure vector becomes an n-dimensional vector. In this case, the area of *ITri* used in the case of 3-dimensional measure vector is extended to the concept of the area of the *hypotenuse face* [88] in the case of an n-dimensional vector. The area of the hypotenuse face is used to evaluate contribution of all measure variables m_{ij} $(j = 1, ..., n)$ of the ith branch to the final importance value of this branch and computed with Equation 4.9:

$$I_i = \frac{1}{n-1}\sqrt{\sum_{k=1}^{n}\prod_{j=1, j\neq k}^{n} m_{ij}^2}. \tag{4.9}$$

After getting the importance value I_i of each branch, importance values of all branches based on the concept of importance area are compared in one space — the importance space. The concept of importance space organizes all branches in one space, and allows to compare the importance of branches based on multiple measures of importance. In the importance space, a threshold I_t is used to control the simplification level: a branch that has larger importance value of I_i than the specified threshold I_t is removed during the CTS as represented with Inequation 4.10:

$$I_i > I_t. \tag{4.10}$$

4.5.3 Comparison of *ITri* and Conventional Approaches

ITri shows advantages in the CTS compared with conventional approaches. This subsection compares *ITri* with conventional approaches in three aspects: importance value, regions of preserved branches, and threshold in the importance space.

In the tetrahedron formed from various measures of importance as shown in Figure 4.4, if methods such as weighted summation of various measures of importance (as shown in Equation 4.11) are used to get the importance value in the CTS, the final importance value corresponds to the summation of the length of three lines *OD*, *OE* and *OF* in the importance space as shown in Figure 4.4. If a single measure of importance is used, the importance value corresponds to the length of the line of *OA*, *OB*, or *OC* in the importance space.

$$I_i = k_p \cdot p_i + k_v \cdot v_i + k_h \cdot hv_i, \tag{4.11}$$

where k_p, k_v and k_h are weighting coefficients. It is obvious that these conventional methods only consider part of the attribute space in the CTS pipeline. Compared with conventional methods, the final importance value of our approach corresponds to area of triangles instead of the length of lines. It considers the whole attribute space and gives a better evaluation on the importance of branch during the CTS.

Figure 4.5 is a 2D diagram to show comparison of *ITri* and persistence as measures of importance in the CTS. In this figure, the red curve represents the threshold I_t where *ITri* is used as the measure of importance in the CTS, and the vertical blue line represents the threshold p_t where persistence p is used as measure of importance in the CTS. When *ITri* is used in the CTS, branch nodes positioned on the top side of the red

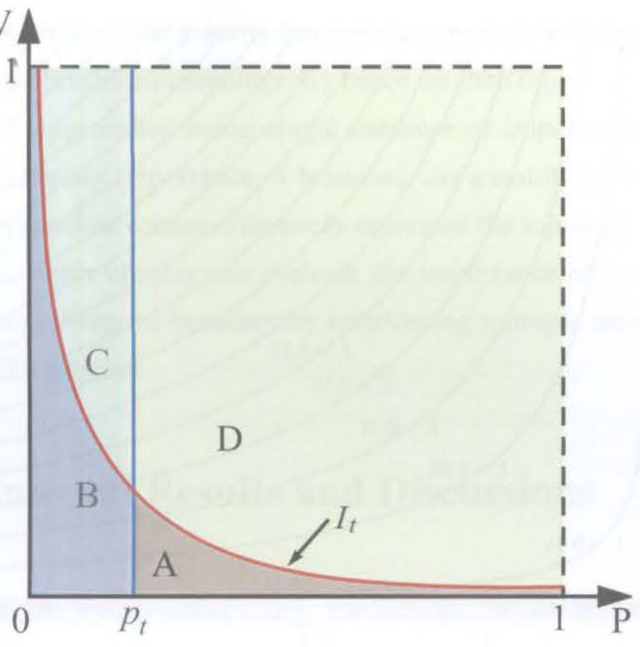


Figure 4.5: The comparison of *ITri* and persistence in the CTS.

curve (i.e., the region C and D) are preserved while ones below the red curve (i.e., the region A and B) are removed. By contrast, if persistence is used as a single measure of importance in the CTS, branch nodes positioned on the right side of the blue line (i.e., the region A and D) are preserved while ones on the left side of the blue line (i.e., the region B and C) are removed during the simplification. The difference between *ITri* and persistence is obvious: persistence removes branches in the region C which are considered as branches of interest by *ITri*, while it preserves branches in the region A which are considered as noise by *ITri*. Despite persistence of branches in the region C being smaller than p_t, they are possibly branches of interest considering their corresponding volume and hypervolume. Similarly, despite persistence of branches in the region A being larger than p_t, they are possibly noise considering their corresponding volume and hypervolume. *ITri* balances these considerations and simplify the contour tree more effectively. The difference between *ITri* and volume or hypervolume in the CTS is similar with the difference between *ITri* and persistence.

Figure 4.6 shows a set of threshold curves with different I_t in the importance space in a 2D diagram. I_t is increased from bottom left to top right in this figure. From these curves we see that persistence, volume and hypervolume of preserved branches are increased accordingly with the increase of I_t (we only show persistence and volume

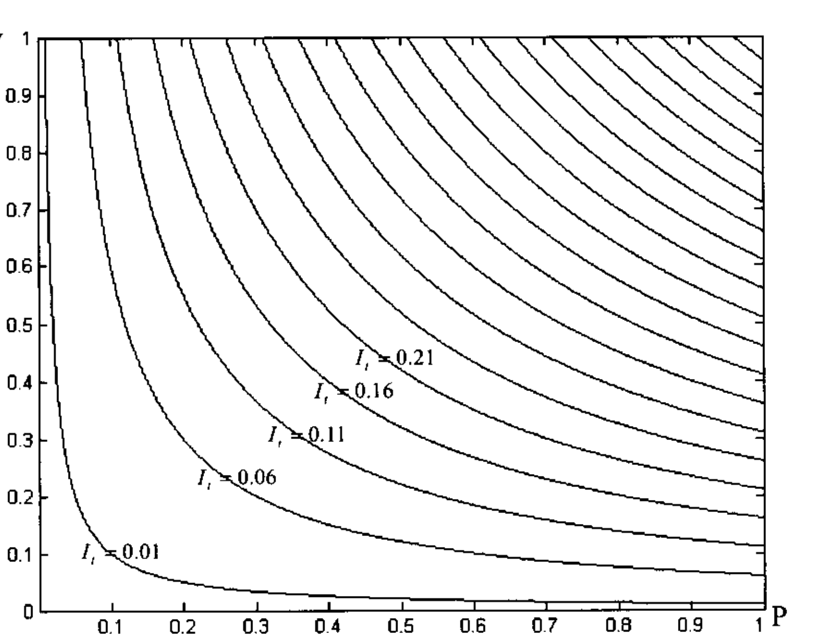


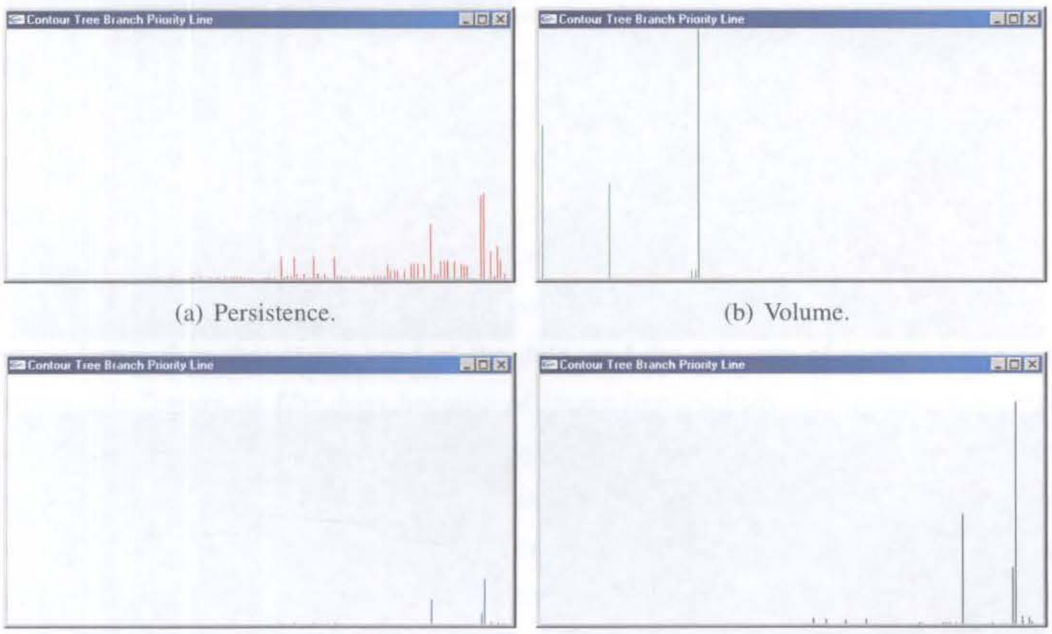
Figure 4.6: A set of threshold curves of I_t with different I_t in a two-dimensional case.

in Figure 4.6 in this 2D example). By contrast, if a single measure of persistence is used, only persistence of preserved branches is increased with the increase of p_t. Even if other measures of importance (e.g. volume) are zero, branches are still preserved as long as their persistence is larger than p_t. This is also true for volume and hypervolume when they are used as a single measure of importance in the CTS. CTS approaches using single measure of importance obviously have shortcomings in evaluating importance of branches: they cannot evaluate importance of branches effectively. As a result, the CTS results based on conventional approaches cannot effectively represent topology of data sets.

From Figure 4.5, we see that if branches in the region C and D need to be preserved based on conventional approaches, users have to determine multiple discriminating functions (i.e. multiple thresholds). As shown in the 2D example in Figure 4.5, one threshold of volume needs to be specified besides the threshold of p_t to approximately determine the region C and D. The determination of multiple thresholds for a given task is often difficult. It is similar to find a point in a multi-dimensional space. This determination process lacks guiding information. On the other hand, *ITri* only needs one threshold to determine the region C and D during the CTS process. It balances contributions of multiple measures of importance for the final importance value of a branch. With the help of the priority line window presented in Appendix, users can effectively modulate thresholds meaningfully based on the *ITri*.

In a word, CTS approaches using single measure of importance obviously have shortcomings in evaluating importance of branches. As a result, the CTS results based on conventional approaches cannot effectively represent the topology of data sets. The proposed approach in this chapter can evaluate the importance of branches better and allow users to preserve/remove branches by considering multiple measures simultaneously during the CTS process.

4.6 Experimental Results and Discussions


Contour Tree Branch Priority Line
Contour Tree Branch Priority Line
Contour Tree Branch Priority Line
Contour Tree Branch Priority Line

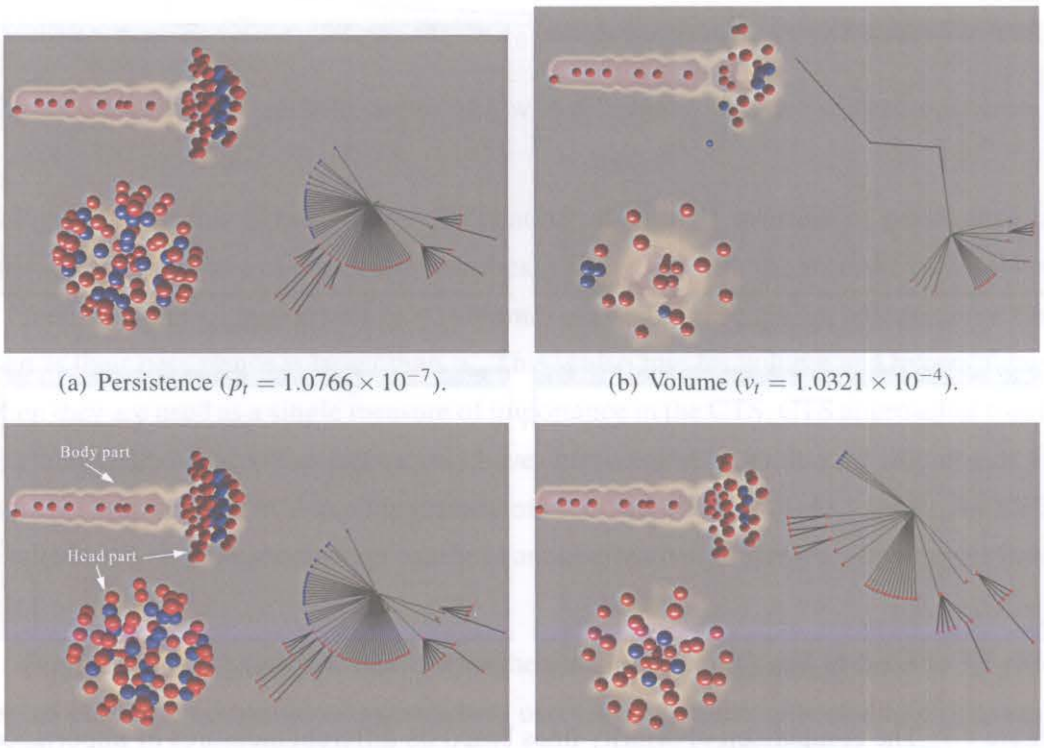
(a) Persistence. (b) Volume.

(c) Hypervolume. (d) *ITri*.

Figure 4.7: The comparison of priority lines based on different measures of importance for the "fuel" data set.

We conducted experiments on various data sets to demonstrate the effectiveness and usefulness of the proposed approach. Our system was run on a Windows XP platform on a Dell machine (Intel Core2Duo CPU E4400, 3GiB RAM) equipped with an NVIDIA GeForce 8300GS graphics card.

Figure 4.7 shows the comparison of priority lines based on different measures of importance for the "fuel" data set (see `http://www.volvis.org/`). From the comparison we see that priority lines of different measures show obvious patterns. We can find some interesting similarities and differences between them: The priority values of the left part in Figure 4.7(a) and Figure 4.7(c) are $\rightarrow 0$. Even if the corresponding priority values in Figure 4.7(b) have value peaks, the priority values of the left part in Figure 4.7(d) are still $\rightarrow 0$. On the contrary, the priority values of the right part in Figure 4.7(a) and Figure 4.7(c) have value peaks. Even if the corresponding priority values in Figure 4.7(b) are $\rightarrow 0$, the priority values of the right part in Figure 4.7(d) have value peaks. These properties exactly match conclusions we discussed in Section 4.5.2. The priority line of *ITri* shows that it can capture advantages and minimize disadvantages of various measures of importance.



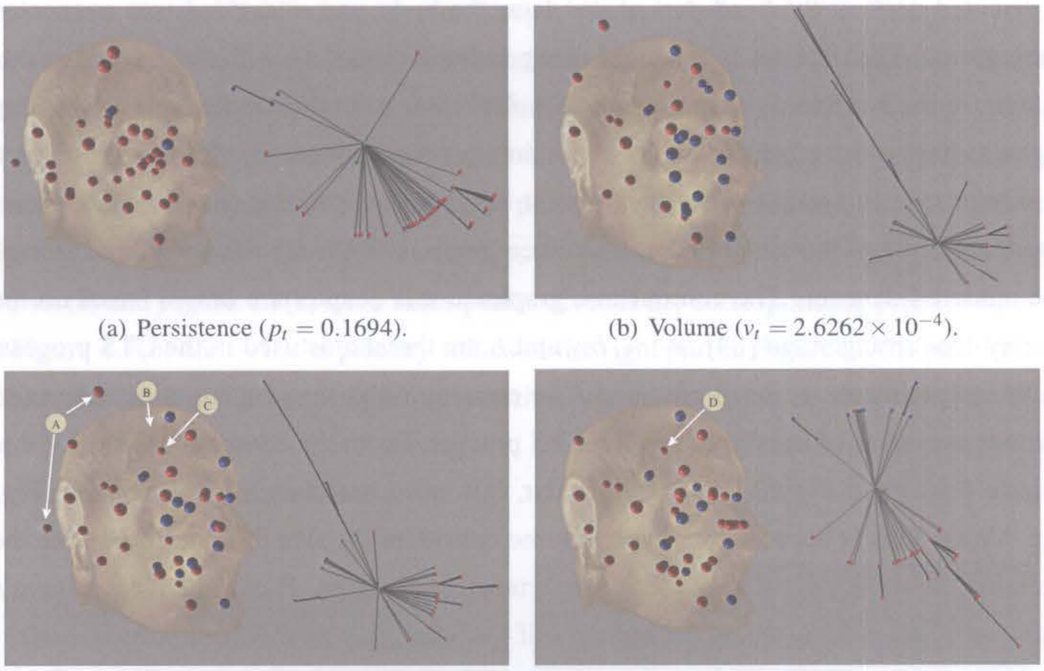
(a) Persistence ($p_t = 1.0766 \times 10^{-7}$). (b) Volume ($v_t = 1.0321 \times 10^{-4}$).

(c) Hypervolume ($hv_t = 1.0766 \times 10^{-7}$). (d) *ITri* ($I_t = 1.0766 \times 10^{-7}$).

Figure 4.8: Rendering and topology of "fuel" data set. The object on the lower left side in each image is the head part view of the object, and the corresponding contour tree is on the right hand side.

Figure 4.8 shows the rendering and topology of the "fuel" data set. Features of the "fuel" data set include: it is a long rod shaped object; there are evenly circularly distributed parts at the head part of the data; the body part is divided into connected subregions. The data set is rendered using volume rendering with the critical points drawn on their original positions, which helps users to understand the data topology visually (red nodes represent the local maximum points, blue nodes represent the local minimum points, and green nodes represent the root points in this chapter). The corresponding level of the simplified contour tree graph is drawn on the right hand side of the rendered data set. The contour tree graphs in this chapter are drawn based on the Orrery-like arrangement [84]. p_t, v_t, hv_t and I_t are thresholds used in the CTS process. In this experiment, we set p_t, hv_t and I_t be equal in order to compare effectiveness of various measures of importance in the CTS process. From the comparison, we see that Figure 4.8(a) and Figure 4.8(c) are similar, but more branches are preserved in Figure 4.8(a). This is because that hypervolume considers the size of features besides the scalar value during the evaluation of importance of features. From the hypervolume's point of view, a branch will be removed if its hypervolume is small even if its persistence is large. In Figure 4.8(b), although volume can preserve some critical points successfully, it cannot remove noise (e.g. the blue point outside the structure) effectively even if the threshold is increased and larger than that of other measures. Through comparison of Figure 4.8(c) and Figure 4.8(d), we see that the simplification results of the two methods are same at the body part because topological subregions are relatively larger than that at the head of the data, and the measure of hypervolume can capture this feature as *ITri* does because of larger hypervolume. However, it is clear that the simplification results are completely different at the head part of the data. Hypervolume cannot capture features at the head part of the object, while *ITri* can capture these evenly circularly distributed topological subregions successfully. This is because that the topological subregions at the head part are smaller and the hypervolume cannot distinguish between them with the same hv_t as used at the body part. However, *ITri* considers three measures of importance at the same time, and captures features at the head part successfully.

For a domain specific case, for example, in a medical data set with tumors inside, tumors often have low ranges of scalar values relative to surrounding objects. It is difficult to detect them because of this reason. Furthermore, there are also different size of tumors in the data set. In order to simplify the contour tree of this kind of data set, users need to preserve branches with small persistence while considering volume and hypervolume of branches at the same time. Figure 4.9 presents the rendering and



(a) Persistence ($p_t = 0.1694$). (b) Volume ($v_t = 2.6262 \times 10^{-4}$).

(c) Hypervolume ($hv_t = 0.3964$). (d) *ITri* ($I_t = 0.0602$).

Figure 4.9: Rendering and topology of "TumorHead" data set. The corresponding contour tree of the data set is on the right hand side in each image.

topology of the "TumorHead" data set (data courtesy of B Terwey, Bremen), where there is a brain tumor pointed out by the arrow B in Figure 4.9(c). p_t, v_t, hv_t and I_t are thresholds used in the CTS process. Given the number of preserved branches being the same for various measures of importance after the simplification, the experiment aims to compare differences of preserved branches based on various measures of importance. In this experiment, the goal of the CTS is to preserve the branch corresponding to the large tumor of B as shown in Figure 4.9(c) while removing other noise branches as much as possible. Because the number of preserved branches is the same, we see that threshold values are different for various measures of importance. This implies that the evaluation of importance of a branch is different when using various measures of importance. In Figure 4.9(a), because the persistence of the tumor is small, the node and branch corresponding to the tumor are removed when persistence is used as the measure of importance during the CTS. From the comparison of Figure 4.9(b), Figure 4.9(c) and Figure 4.9(d), we see that all of the images preserve the tumor node (e.g. the nodes that the arrow C and D point to in Figure 4.9(c) and Figure 4.9(d) respectively).

However, differences are obvious: volume and hypervolume cannot remove some nodes effectively (e.g. the nodes that the arrow A points to in Figure 4.9(c)) because of lacking of consideration of other importance measures.

From the experiments, we can see that the proposed approach can simplify the contour tree meaningfully and effectively. The utility of our approach is two-fold: as a comprehensive solution for combining multiple measures of importance in the CTS, and as a general effective interface with a priority line window to manipulate thresholds in the CTS. Our approach provided a complete pipeline for users to organize, display and simplify the contour tree. It showed advantages in the CTS process. As represented in Equation 4.7, *ITri* takes three measures of importance into consideration. The features captured by *ITri* are more meaningful compared with features captured by a single measure. Another advantage of our approach is that it can be easily extended as a general scheme to simplify the contour tree with more than three measures of importance. This property encourages users to develop more measures of importance which finely depict features of data sets. This powerful scheme allows users to perform the CTS with several mouse clicks, which improves the CTS efficiency greatly.

In addition to approximations of geometric measures as used in this chapter, statistical properties of the sample values inside region can also be used to measure importance of branches [26], for example, the mean value and the standard deviation. Besides, the width-to-height ratio of the region can also be used as a measure to evaluate importance of branches. For example, given two regions with the same volume, persistence and/or hypervolume, the importance of the region with a long thin shape and the region with a compact round shape is different and can help the user to evaluate the importance of branches. For example, if the user is more interested in round and compact shape, the region with round shape is more important than the region with the long thin shape. These measures of importance can be used in the approach proposed in this chapter to improve the evaluation of the importance of branches.

4.7 Summary

This chapter introduced an importance-driven approach for the CTS. The proposed approach combined multiple measures of importance into a CTS pipeline through the concepts of attribute space, importance triangle and importance space. Interaction interfaces made the CTS a meaningful and user-directed process.

The theoretical analysis and experimental results demonstrated that the presented approach has advantages in the use of measures of importance compared with conventional approaches: It allows for the simplification of contour trees by considering the whole attribute space and giving a better evaluation of the importance of a branch, instead of limited attribute space used by conventional approaches. We presented three measures of importance in this chapter. However, the proposed approach can be easily extended as a general scheme to simplify the contour tree with more than three measures of importance. This property allows users to define various measures of importance based on specific data sets and applications flexibly in order to improve the simplification efficiency. This powerful scheme allows users to perform the CTS with several mouse clicks, and improve efficiency.

Chapter 5

Structural Relationship Preservation and Depiction in Volume Rendering

From previous chapters, the contour tree is defined as a topological abstraction of a scalar field, and a graph that tracks changes to a contour. Topological relations are recorded as the evolution of a contour in a scalar field. Volume rendering provides a powerful scheme for users to understand relations between structures as more structures can now be revealed in a single volume rendered image. Unlike the relations between opaque objects in typical 3D computer graphics, spatial relation analysis for volume rendered images is more complicated due to the property of semitransparent structures.

This chapter presents structural relationship preservation as a technique for improving understanding of volume rendering based 3D data analysis. Because basic theory behind the depiction of relationship in a volume data set lies in the topology, this chapter utilizes topological properties derived from the contour tree to represent and depict relations between structures. Structural relationship preservation in volume rendering provides more intuitive and physically meaningful renderings that depict physical structures more clearly. It enhances information provided by common rendering techniques. The contributions of this chapter are as follows:

- A concept of structural relationship preserved mapping between data and rendering space is proposed. The concept allows for every structural relationship of interest in the data space to be mapped onto objects in the rendering space by utilizing various rendering methods.

- Two typical relationships of inclusion and neighboring are defined, extracted, represented, and revealed in volume rendering in the pipeline.

- Contour tree controlled structural relationship representation allows users to perceive structural relationship, and control how and what structural relationships are revealed in the pipeline.

5.1 Introduction

Traditional computer graphics is a unidirectional projection from a 3D objective scene to a 2D image. It provides capabilities of synthesizing virtual environments or regenerating an existing scene. It is usual to think in rational that how effectively the generated scene supports specific tasks or how well features of the denoted objects can be discerned. In the context of medical volume visualization, this involves questions of how effectively the visualization depicts anatomical structures and how well features of anatomical structures can be discerned in order to make decisions. Most of the conventional volume rendering methods place emphasis on conveying details of the desired features or structures, by exposing them clearly to viewers in the results. However, spatial relations between structures in a volume are also of interest to viewers [27]. Relationship and its uses have been studied in several research fields as investigated in Section 2.7. There are various kinds of relations existing between objects in a volume data set. For example, in surgical planning, radiologists are interested in not only shape of structures of interest, but also their neighboring information (e.g., how close they are). Such relation information of structures is crucial for the visual analysis and understanding of volumetric data in various applications [27].

Volume visualization creates 3D renderings which show perceptual attributes such as color. As Bertin [11] pointed out, information is about relationships between things, and similarly a picture is about relationships between graphical marks. It is intuitively obvious that a good visualization exposes *structural relationship* of the information that it represents. We expect a good visualization to be a structural relationship preserved mapping between an information domain and a perceptual domain, such that mental models and understanding of the data set are improved during data analysis.

Conventional visualization approaches do not take into account the structure of the underlying data. The explicit knowledge of the structural characteristics is not integrated into the rendering pipeline. Therefore, the structural relationship analysis part is often taken as a post-processing step in image space [124], and depends on manual inspection to reveal various relations [27]. In volume rendering, the perceivable structural relationships are either represented by visual elements such as shaded areas, transparent surfaces, or implicitly represented by placing visual elements (e.g. elements which help the user to better understand topological relations) in renderings. Structural relationships of objects are assumed to be induced on a set of visual elements by means of multiple structural features. Explicitly depicting structural relationships can enhance visual representation and understanding of structures in the data set.

This chapter presents an approach on depicting structural relationships between objects in volume rendering through a concept of structural relationship preserved mapping. The concept is set up through analyzing the roles of perception in volume visualization. Two typical relationships of inclusion and neighboring are defined and depicted in volume rendering respectively. The contour tree controlled structural relationship depiction allows users to perceive structural relationship in a more direct way. The objective of this chapter is to deliver a new visualization approach for better understanding, presentation, and visualization of volumetric data based on structural relationships. The structural relationship preservation approach has advantages: it allows analyses of volumetric data to focus on revealing high-level topological relations instead of low-level rendering parameter modulation. The approach improves understanding of volumetric data. The experimental results show that structural relationship preservation provides information which is not available in commonly used volume rendering techniques, as well as it enhances information provided by common rendering techniques.

5.2 From Perception to Structural Relationship Preservation

Volume visualization aims at making the mapping from data to renderings as effective as possible. It seeks to maximize the amount of information that a human viewer can perceive out of the presentation. Jankun-Kelly et al. [56] presented a general model of the visualization exploration process for a volume data set. The effectiveness of a visualization can be measured in terms of ease and directness of acquiring its intended interpretation [36]. In this thesis, we assume that volume data can be classified using transfer functions or other methods. The purpose of volume visualization is to visually present these classified structures in order to extract data information for making decisions. In the context of volume visualization, the structural correspondence between volume data and perceptual visual elements in the rendering should be set up in order to get the effective volume visualization. Therefore, analysis of the roles of perception for volume visualization must consider the interpretation of rendered images. The volume visualization process generates rendering images and the interpretation process uses the rendered image from the visualization process to get properties of the original data. Because the interpretation of a rendered image depends on human visual perception, Dastani [36] pointed out that a visualization is effective if the intended structure of the data and the perceptual structure of the visualization coincide. The effectiveness of volume visualization refers to effectively present structured data information and effectively find structures of interest based on visual presentations. Based on this view, we combine perceptual structure into the volume rendering based data analysis pipeline as shown in Figure 5.1. The figure shows how a volume data set is rendered and explained during a volume rendering based data analysis pipeline.

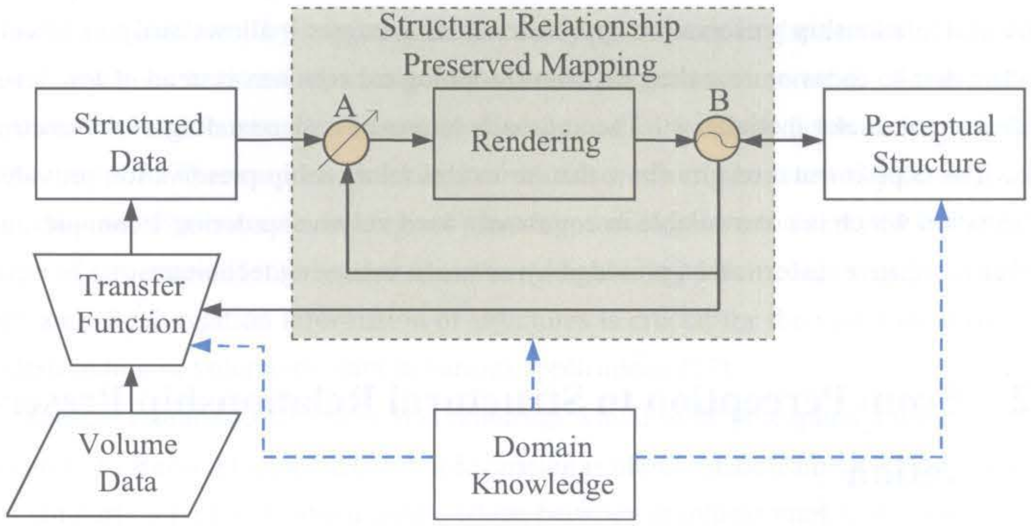


Figure 5.1: Combine perceptual structure into the volume rendering based data analysis pipeline.

In Figure 5.1, the volume rendering process begins with an input volume data set. The first step in this process is to determine the structure of the data. This step is a part of classification in volume rendering. It transforms the input data to structured data. Transfer functions and other classification methods are used in this step. The second step is to render the structured data. Because it is not possible to display all objects classified in the previous step at the same time, it is necessary to determine which structures will be displayed and how. This is determined at the point A in Figure 5.1. Then the structured data set is rendered followed by its interpretation indicated at the point B in Figure 5.1. In reverse, through interpretation, the user compares perceptual structure and rendered image in order to see whether the perceptual structure from the domain knowledge and the rendering display coincide. If this is not true, the user needs to return to the point A to modify rendering options. The user may also return to the transfer function module to classify the data and get new structured data. During this pipeline, the domain knowledge is used in transfer function modulation, rendering interpretation, and perceptual structure understanding.

5.2.1 Structural Relationship Preserved Mapping

Based on the analysis pipeline as shown in Figure 5.1, volume visualization is then defined as a mapping between structured data elements and visual perceptual elements. In order to explain this mapping, we introduce a relational system into the volume rendering based data analysis pipeline. We assume that certain relations exist between structured data elements, e.g. topological relations and size relations. We also assume that there are similar relations between visual perceptual elements. An effective volume visualization should define a mapping between data and renderings such that every relationship between objects in the data space is mapped onto visual variables in the rendering space. The structural relationships between structured data are preserved during this mapping. The shaded region in Figure 5.1 shows how structural relationships are mapped and preserved during the data analysis pipeline.

5.2.2 Topological Relationships in Data Set

Relational features provide information about adjacency, repetitive patterns and geometrical relationships among objects. In theory, a relationship is a mathematical object that in general can have a very complex type. It is a specific connection between objects, entities or concepts. There are different relationships between objects in the data space: inclusion, neighboring, etc.. Topological relationship refers to the global topology of structures in a volume data set. Topological analysis of scalar data sets helps users to understand global structure of the data during the data exploration process. This chapter uses the contour tree to represent the topological structure of a volume data set. We also confine our interest on relationships that are used to enhance the understanding and perception of structures in the rendering. To provide a formal basis, we give definitions of two typical relationships of inclusion and neighboring as follows.

Inclusion Relationship

In 3D space, structures of interest are often included by other structures because of their spatial positions. The inclusion relationship refers to the situation of objects included by others in 3D space, which creates inner objects and outer objects respectively. Takahashi et al. [107] called this as isosurface embedding.

This problem is prevalent. For example, in a 3D medical data set, objects are often concave, and are at least partly occluded by others. In order to understand the inclusion relationship more clearly, we give following definitions: The viewpoint vp is some point in the 3D space representing the viewer or camera position. The *inclusion relationship* is defined as: Let v_1 and v_2 be two distinct voxels of a data set and $near(v_1)$ and $near(v_2)$ be the interpolated regions of v_1 and v_2 with their neighboring voxels. Relative to the viewpoint vp, v_2 includes/occludes v_1 if there is a half-line hl starting at vp and points p_1 in $hl \cap near(v_1)$ and p_2 in $hl \cap near(v_2)$ so that p_2 lies between vp and p_1 on hl, where "$\cap$" represents the intersect operation between line and region.

The inclusion relationship is commonly seen in a volume data set and heavily affects understanding of it. For example, in a human head data set, a user cannot get complete tumor information inside the brain because of inclusions by outside surfaces (for example, see Figure 7.6(b)). In order to interpret objects of interest effectively, the inclusion relationship between objects needs to be specifically depicted. Consequently, providing clear insight into inner structures involved in a volume data set has been a challenging task in the field of volume rendering.

Neighboring Relationship

The position of different volume element within the overall volume, is challenging. It plays critical roles in information understanding. Usually, a volume data set can be divided into various subregions which represent different objects. Neighboring relationship refers to the situation that objects are extremely connected or there is a gap interval which can be transparent or opaque between them. The neighboring information has the role of controlling the perceptual information of structures, for example, it is used to modulate optical properties of structures to improve perception of difference between structures. It can also be used in surgical planning to reveal closeness of structures in order to make decisions.

5.3 Relationship Depiction in Volume Rendering

As shown in Figure 5.1, volume rendering helps to map relationships from structured data to perceptual structure. The relationships are preserved during this mapping. So it is necessary to develop appropriate volume rendering methods in order to reveal various relationships. We believe that appropriate depiction of structural relationships will produce better understanding of a 3D data set.

The depiction of topological relationships in 3D rendering space can enhance understanding of structures in the data set. Usually, not all relationships between objects can be revealed completely in a 2D image view during data analysis. In this section, we present approaches to depict inclusion and neighboring relationships respectively based on the contour tree.

5.3.1 Depiction of Inclusion Relationship

Despite direct volume rendering being one of the main techniques for visualizing entire volumetric structures at once, it still requires time-consuming interactions for tweaking visualization parameters in order to obtain comprehensible rendering results. Furthermore, scalar value is locally but not globally meaningful in a data set. Traditional volume rendering cannot effectively depict inclusion relationships because of the overlapping of scalar value ranges. Although Takahashi et al.'s approach [107] can analyze inclusion relationship based on the topology, it still applies transfer functions to a data set globally by the concept of "inclusion level". Weber et al. [122] defined transfer functions for different subregions based on the contour tree, but they did not consider the inclusion relationship in the pipeline.

Figure 5.2 shows an example of a 2D mesh and its corresponding contour tree. As mentioned, the contour tree acts as a visual index to segment data into various subregions. Figure 5.2 presents the segmented 2D mesh with different subregions. Arcs

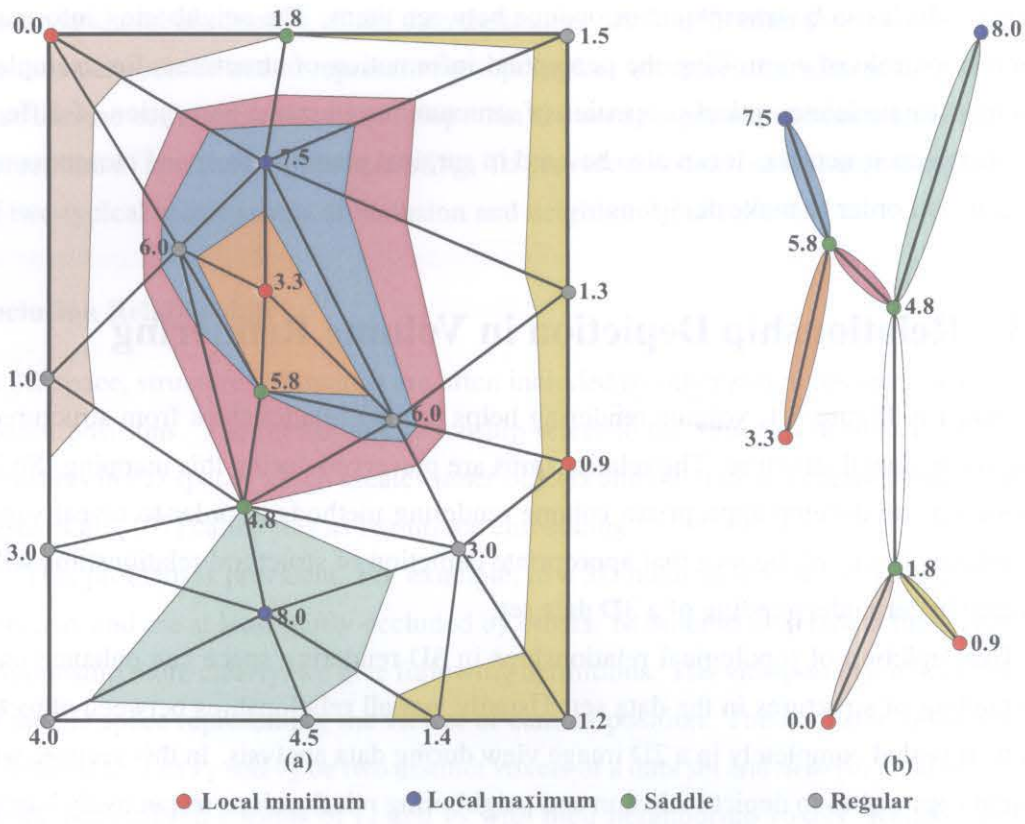


Figure 5.2: An example of a 2D mesh and its contour tree. The 2D mesh is segmented into subregions and indexed with the contour tree.

of the contour tree and their corresponding subregions are encoded with colors. This example is used to analyze relationships implied in the contour tree.

Similar to approaches in [103], this chapter uses graph based approach to analyze the inclusion relationship represented in the contour tree. In this chapter, the contour tree of a data set is represented as a graph (see Figure 5.3(b)). As shown in Figure 5.3(b), given a contour c as the "outside" isosurface — i.e. a point in the tree, and choose two points p and q in the tree. c, p and q can be critical points or regular points. In order to determine whether p is "inside" q, we find the path P (dashed line in blue in Figure 5.3(b)) from c to p in the contour tree. There is only one such path, because it is a tree. Similarly, we find the path Q (dashed line in red in Figure 5.3) from c to q. p is "inside" q if Q is a prefix of P (i.e. we can only get to p by passing through q first). The points c, p and q belong to various arcs. The subregions that these arcs correspond to are displayed in Figure 5.3(a). From this figure, we see that the inclusion relationship of

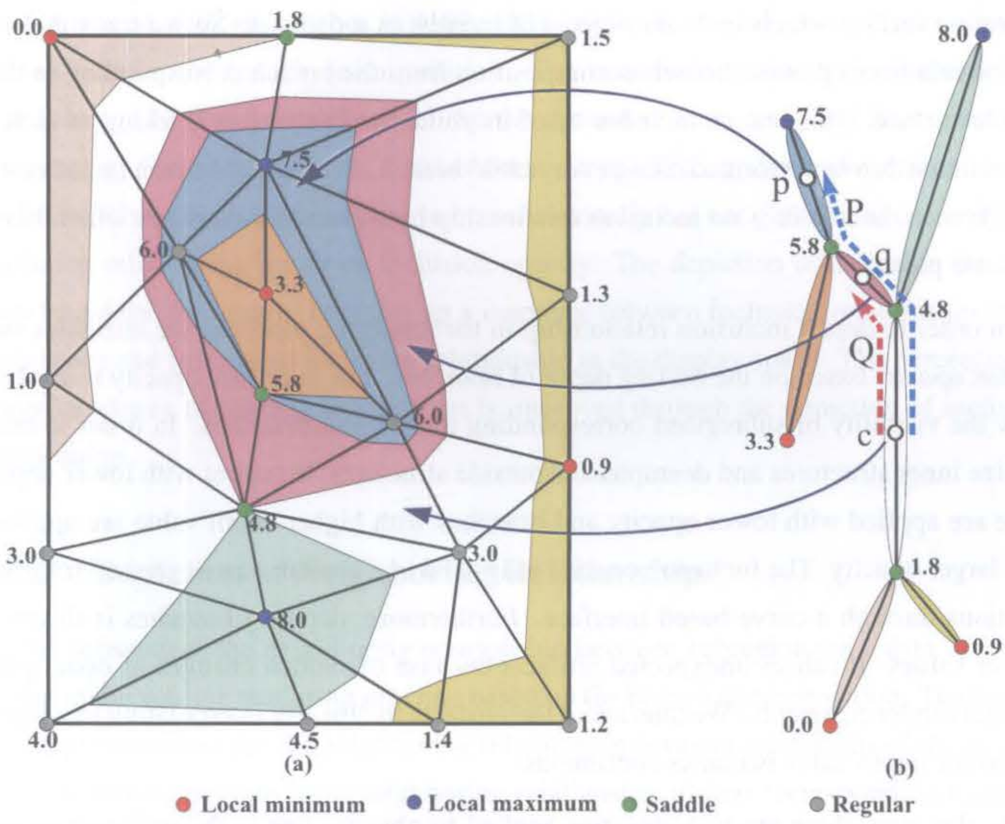


Figure 5.3: Inclusion relationship analysis in the contour tree.

c, p and q represented in the contour tree graph is clearly revealed between subregions in the data set.

Usually, if there is no overlap in scalar value ranges of objects, inclusion relationship can be depicted based on general transfer functions by rendering outside objects transparently/semi-transparently. If regions overlap in their value ranges, it is not possible to depict inclusion relationships using general transfer functions. In this case, a segmentation is necessary. We use the branch decomposition to analyze the inclusion relationship in this chapter. The branch decomposition provides a direct way to track inclusion relationship based on branches. Because a branch is usually a concatenation of a sequence of arcs in the contour tree, the subregion corresponding to a branch is a combination of subregions corresponding to arcs. We do not consider inclusion relationships of subregions corresponding to individual arcs, but analyze inclusion relationships of subregions corresponding to branches. We assume that there exists an outermost surface which includes objects of interest in a data set. So we track the inclusion relationship in the branch decomposition from the branch corresponding to the outside surface. This branch includes all of its children. The further tracking of inclusion relationship is performed along every child branch. The parent branch includes its child branch, and there is no inclusion relationship between child branches which have the same parent.

In order to depict inclusion relationship in the rendering pipeline, we introduce *inclusion opacity* based on the nesting depth of branches. The inclusion opacity is used to show the visibility of subregions corresponding to different branches. In order to emphasize inner structures and deemphasize outside structures, branches with lower depth value are applied with lower opacity and branches with higher depth value are applied with larger opacity. The inclusion opacity is applied in a similar way as general transfer functions through a curve based interface. Furthermore, depth of branches is discrete integer values. It causes unexpected artifacts because of sudden changes of opacity in the final rendering results. We thus avoid the artifacts by linearly interpolating the depth so that the depth value becomes continuous.

At this step, there are two opacities applied to objects: one is the inclusion opacity α_d, and another is the opacity α_t specified by the original transfer function. Two opacities need to be combined in order to get the final opacity α applied to objects as represented in Equation 5.1:

$$\alpha = f(\alpha_d, \alpha_t), \tag{5.1}$$

where f is the function used to combine two opacities. A typical combination is computed in Equation 5.2:

$$\alpha = \alpha_d \cdot \alpha_t. \tag{5.2}$$

Given two branches a and b, we assume branch a is included by branch b and represented as $a \in b$. The final opacity for a and b is α_a and α_b respectively. α_a and α_b should meet inequality 5.3 in order to preserve inclusion relationship in the rendering:

$$\alpha_a > \alpha_b. \tag{5.3}$$

Compared with approaches in [107, 103], our method analyze inclusion relationships based on the branch decomposition. The branch decomposition overcomes the disadvantage of "over-segmentation" by the general contour tree, and thus avoids the unnecessary inclusion analysis. Furthermore, our method applies transfer functions of different subregions corresponding to branches locally but not globally when analyzing inclusion relationship based on inclusion opacity. The depiction of inclusion relationship based on the contour tree sets up a mapping between inclusion relationship in the data space and perceptual inclusion relationship in the display space. The perception of inner structures of interest in a data set is improved through the depiction of inclusion relationship.

5.3.2 Depiction of Neighboring Relationship

In this subsection, the neighboring relationship between subregions in a data set is depicted in the volume rendering pipeline based on the branch decomposition. The branch decomposition encodes the neighboring relationship between subregions of the data set directly: two subregions have neighboring relationship if their corresponding branches are connected; otherwise, the neighboring relationship does not exist. In order to reveal this relationship based on the contour tree, we check branch connection information and use this information to emphasize structures in volume rendering pipeline.

In the contour tree, neighbors of a given branch include its parent, children and sibling branches. For example, as shown in Figure 5.4(b), d's neighbors include its parent b and sibling e. f is not connected with d, so it is not the neighbor of d. Figure 5.4(a) shows these corresponding neighboring relationships in the data domain. From this figure, we see that the contour tree can represent neighboring relationship of objects faithfully.

In order to reveal neighboring relationship in volume rendered images, a color range based approach is proposed to encode neighboring relationship presented in the contour tree. We use different colors to render different regions in order to reveal neighboring relationship in this chapter. More specifically, the color transfer function is used to depict neighboring relationship in volume rendering and creates aesthetic appeal in the rendering. Color transfer functions map scalar densities to colors in order to label different objects and create aesthetic appeal in volume rendering, and to improve data analysis efficiency. Conventional color transfer functions are parameterized by values that

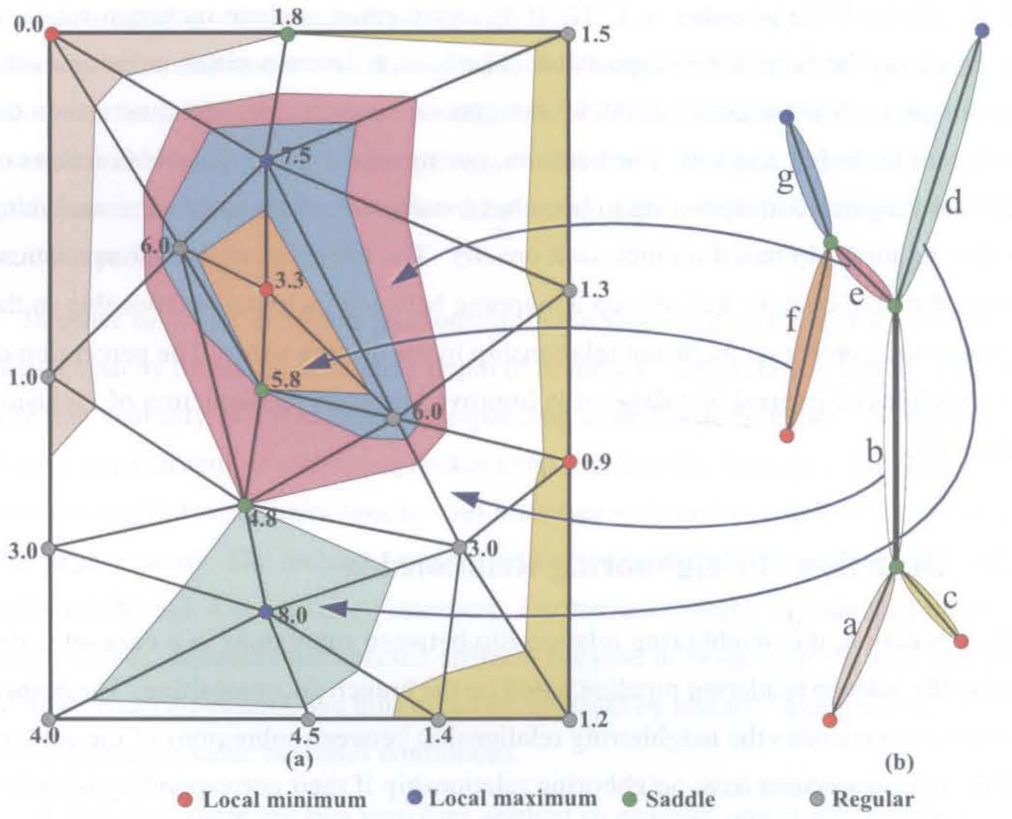


Figure 5.4: Neighboring relationship analysis in the contour tree.

segment the data into different subranges, which are arbitrarily assigned with different colors based on experiences and personal preferences [65, 64]. Thus color is only used to identify subranges of the data, and not to convey any quantitative relations among the data. Furthermore, color harmony is one popular design aspect in terms of aesthetics. Cohen-Or et al. [31] introduced a framework of automated image color harmonization. Wang et al. [119, 120] extended this color harmonization to be used in volume visualization. They presented how color harmonization is used to semi-automate color definitions in volume rendering. The scheme uses manually specified parameters such as importance factors to control color components (hue, vividness, lightness) selection for segmented data rendering.

Our work extends the color selection algorithm in [119] by incorporating topological features, and encodes neighboring relationship in color transfer function generations in volume rendering. Figure 5.5 shows typical harmonic color templates. Any harmonic template could be used to represent different regions to reveal neighboring relationships.

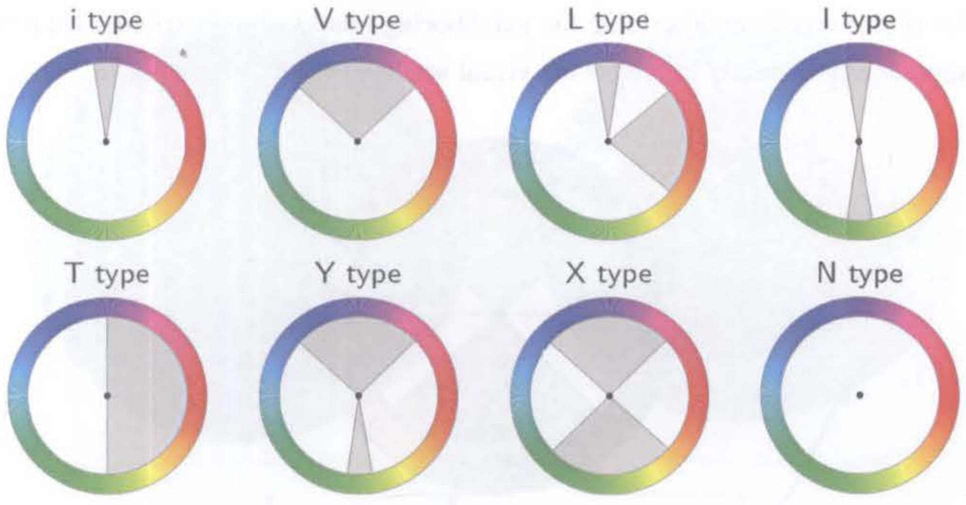


Figure 5.5: Harmonic hue wheel templates. A collection of colors that fall into the gray areas is considered to be harmonic. The templates can be rotated by an arbitrary angle. See [31] for the exact definitions.

In our approach, the hue wheel in the HSV color space is divided into four ranges to represent neighboring regions in the rendering as shown in Figure 5.6: current (one point), children, parent (one point) and siblings. Because the current branch has multiple children and sibling branches, larger hue ranges need to be allocated to children and sibling branches in order to more easily differentiate multiple objects based on colors. In this chapter, we use the harmonic color template X-Type [31] as the neighboring relationship preservation template. The harmonic color template X-Type has two distinguished properties: 1) it has two large harmonic hue ranges with same sizes (grey regions in Figure 5.6); 2) the two harmonic hue ranges are also uniformly distributed on the hue wheel. The first property allows users to more easily differentiate multiple objects based on colors. The second property helps to more easily differentiate two groups of objects in the rendering: children and sibling. In Figure 5.6, the size of gray regions for children and sibling branches is 93.6° respectively, and the position of the current branch and parent branch is at the center of each white region. Same sizes of the hue range are allocated to children and sibling branches. This also helps to balance color representations of objects in the rendering. Based on this approach, neighboring relationships between branches in the contour tree are preserved in the rendering with various colors. Users may perceive neighboring relationships of various subregions through viewing the relation color wheel and rendering at the same time. The color wheel acts as a legend in describing the neighboring relationship in volume rendering. This approach also greatly improves the visual aesthetic appeal of rendering.

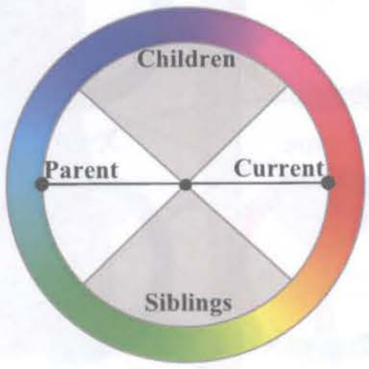


Figure 5.6: The neighboring relation color wheel: different color ranges are used to encode neighboring relationship of branches in the contour tree.

The depiction of neighboring relationship has wide applications, for example, in medical applications, radiologists often need to know which structures are affected by tumors or other structures of interest in order to do surgery planning. The depiction of neighboring relationship can meet this purpose directly: affected structures have neighboring relationship with structure of interest and can be depicted based on the neighboring relationship. The perception of neighboring relationship of structures is enhanced by the direct depiction of neighboring relationship in the rendering display.

5.4 Experimental Results

We conducted experiments on various datasets to demonstrate the effectiveness and usefulness of our approach. Our system was run on a Ubuntu platform on a Dell machine (Intel Core2Duo CPU E4400, 3GiB RAM) equipped with an NVIDIA GeForce 8300GS graphics card.

Figure 5.7 shows the rendering of "nucleon" (see `http://www.volvis.org/`) data set to depict inclusion relationship. In this data set, small inner structures are included by outside surfaces. Users need to depict this inclusion relationship.in order to reveal inner structures. The top-left image in Figure 5.7 is rendered using general direct volume rendering. It shows that the general direct volume rendering cannot depict the inclusion relationship effectively because of overlap of scalar value ranges of different

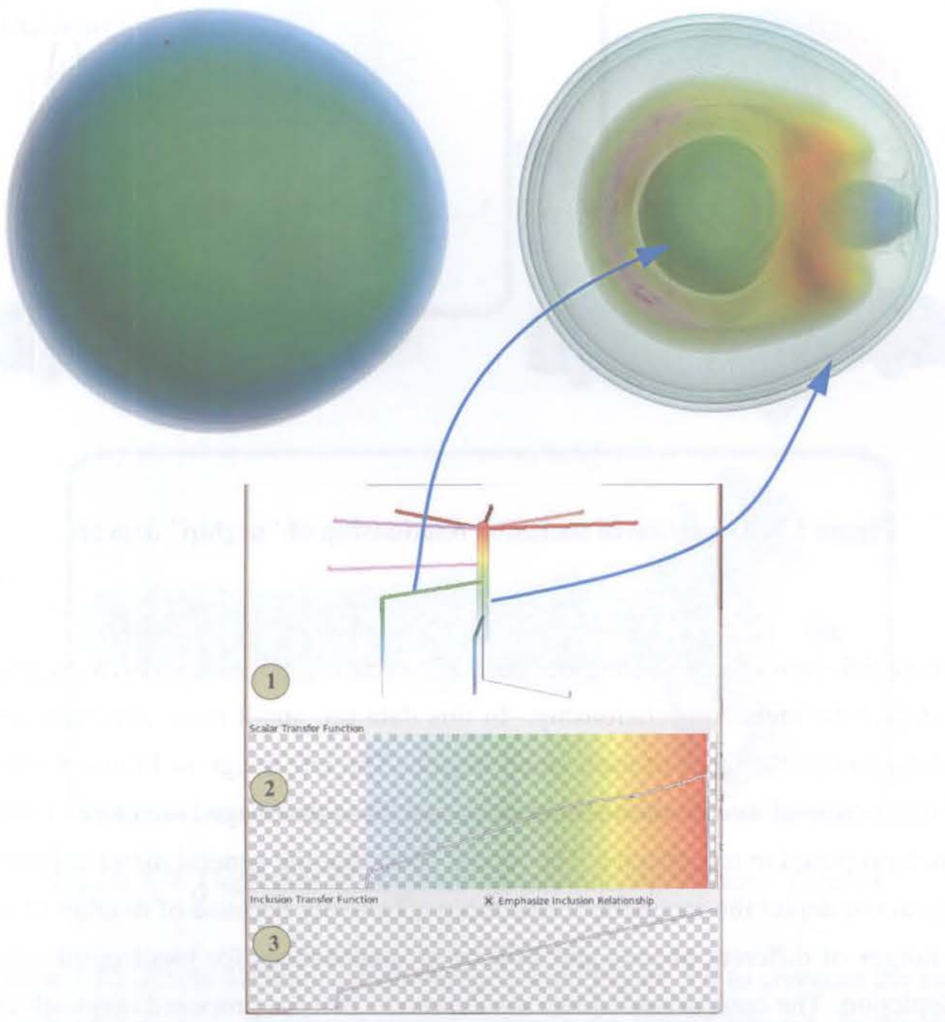

1
Scalar Transfer Function
2
Inclusion Transfer Function
✕ Emphasize Inclusion Relationship
3

Figure 5.7: Inclusion relationship analysis based on the contour tree: 1) Contour tree, 2) Scalar transfer function, 3) Inclusion transfer function.

objects. The top-right image is rendered using the inclusion relationship depiction approach proposed in this chapter. Its corresponding contour tree is shown in the middle of this figure. Through the comparison of the two results, we see that the proposed approach can depict inner structures clearly and it provides richer information for the user to analyze the data set. This depiction allows the user to easily understand topological relationships of structures inside the data set. From this experiment, we see that the proposed approach has the following advantages: the inclusion relationship is effectively depicted, and it also improves the perception of structures through presenting different layers of structures at the same time.

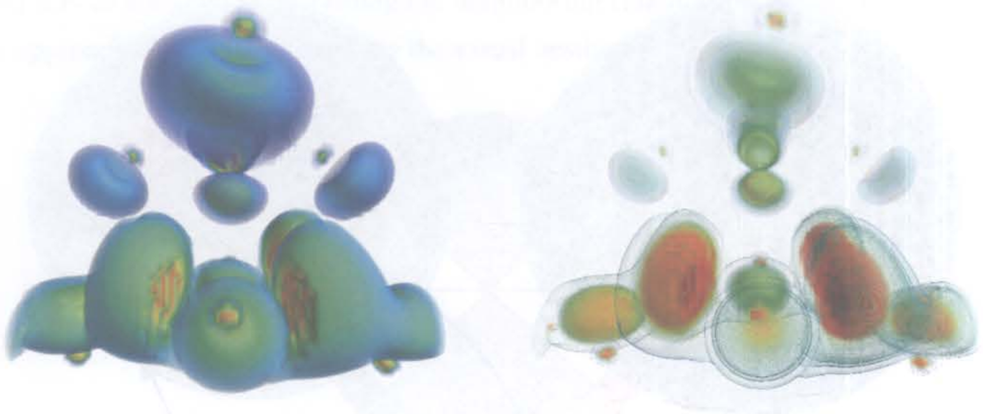


Figure 5.8: Depiction of inclusion relationship of "neghip" data set.

Figure 5.8 shows the rendering of "neghip" (see `http://www.volvis.org/`) data set to depict the inclusion relationship. In this data set, small inner structures are included by outside surfaces of each small object. The left image in Figure 5.8 is rendered using general direct volume rendering, and the right image is rendered using the approach proposed in this chapter. The results show that the general direct volume rendering cannot depict the inclusion relationship effectively because of overlap of scalar value ranges of different objects and lacking of mechanisms for topological relationship depiction. The comparison of two results shows that the proposed approach in this chapter can depict the inclusion relationship effectively and provides more insight into the data set than conventional approaches.

Figure 5.9 presents the rendering result of depicting neighboring relationship in the "fuel" data set. The contour tree and relation color wheel are presented at the same time. In the top-left image, the current branch and its neighboring branches are pointed out in order to show their neighboring relationships. In the bottom image, the subregion corresponding to the user selected current branch and its neighboring structures are emphasized and rendered in various colors based on the relation color wheel presented at the top-right of the figure. In this way, users can perceive neighboring structures of a selected subregion clearly. It helps users to understand which structures are affected during interaction with a specific subregion and improves the perception of neighboring relationship.

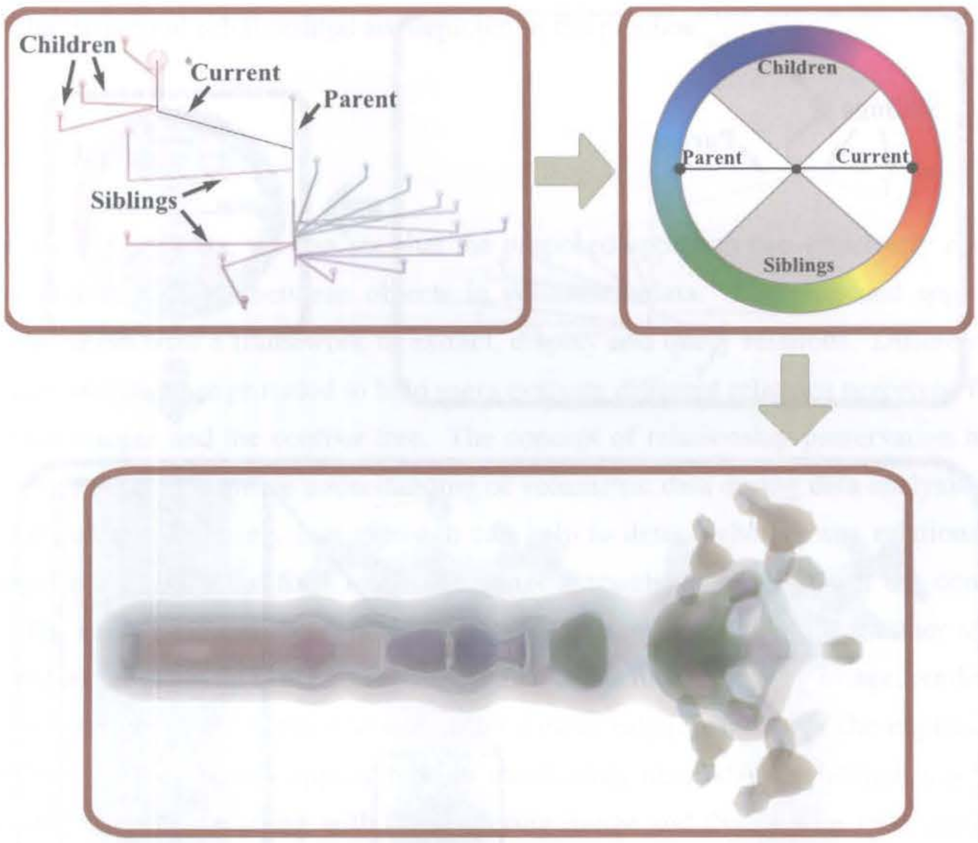


Figure 5.9: Depiction of neighboring relationship of "fuel" data set.

Figure 5.10 selects different current branch from Figure 5.9 to compare the rendering results of depicting neighboring relationship in the "fuel" data set. In this figure, Figure 5.10(a) is the original rendering without depiction of neighboring relationship. Figure 5.10(b) presents the rendering with neighboring relation depiction and the local enlarged view of the body part of the data. The user selected current branch and its neighboring branches are pointed out as shown in the contour tree at the top-left part of the figure. From the contour tree, we see that the structure corresponding to the user selected current branch has three neighboring structures topologically: one parent branch and two sibling branches. In Figure 5.10(b), the structure corresponding to the user selected current branch is rendered in red color and its neighboring structures are rendered in different colors in the enlarged view. This structural relationship is clearly represented visually in the rendering result. From the comparison, we see that the depiction of neighboring relationship in the rendering enhances the perception of differences and topology of various regions of the data, and thus improves the overall

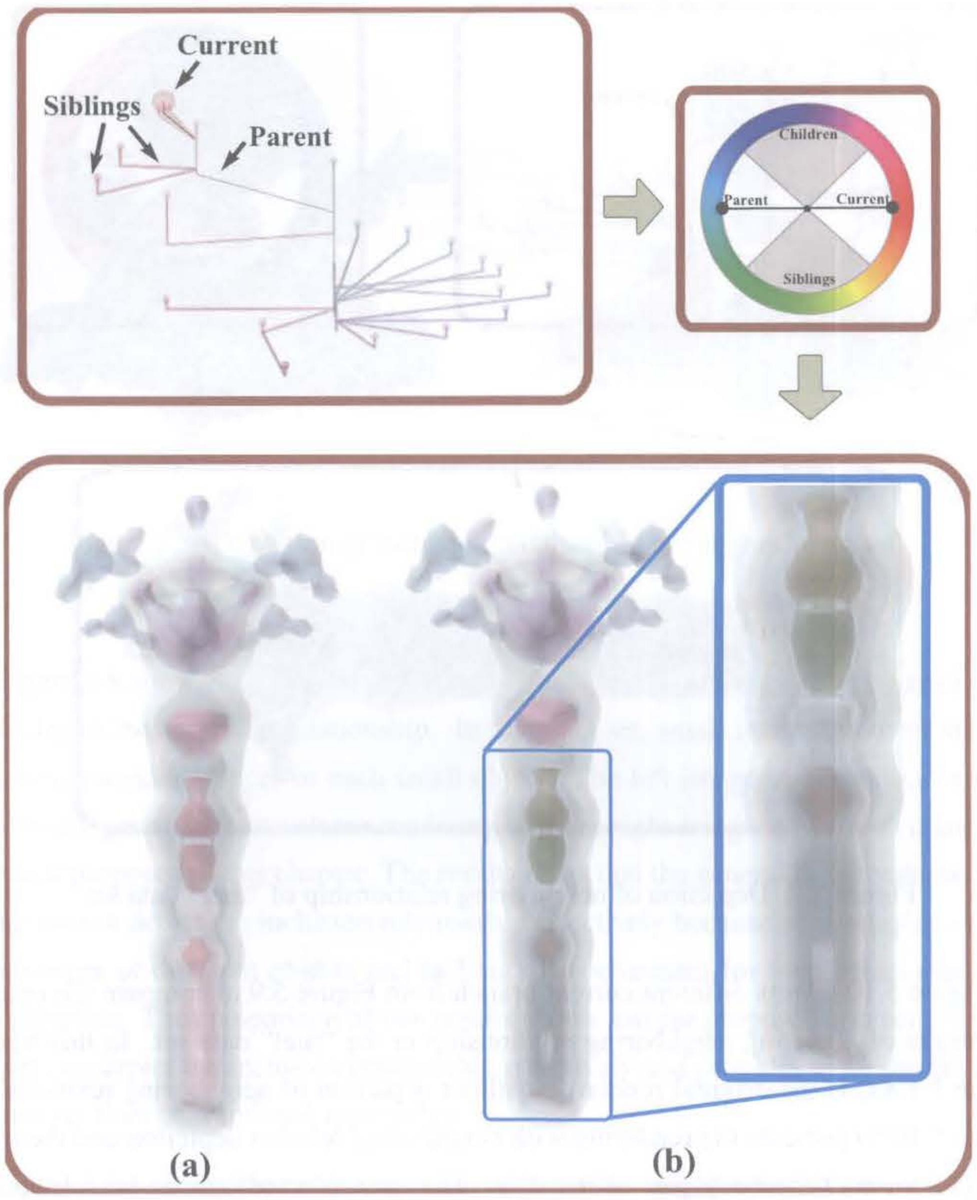


Figure 5.10: Depiction of neighboring relationship of "fuel" data set.

understanding of the data.

The experimental results show that structural relationships in a 3D data set are critical features used to help the user to understand data sets. Various structural relationships are depicted and preserved in renderings perceptually using different graphics representations. The contour tree controlled structural relationship depiction allows the user to perceive structural relationships in a more direct way. The user can also control how and what structural relationships are depicted in the pipeline.

5.5 Discussions

From the experiments, we can see that the proposed approach can effectively extract and visualize relations between objects in volumetric data. The proposed approach provided users with a framework to extract, display and query relations. Different interaction facilities are provided to help users evaluate different relations perceived from rendered images and the contour tree. The concept of relationship preservation mapping helps users to improve understanding of volumetric data during data analysis. For general volume rendering, our approach can help to detect whether any relations are revealed correctly in the final rendering image through interacting with the contour tree. For example, from the contour tree, one structure is included by another structure. If this relationship is not correctly revealed in the final rendering image, rendering parameters should be changed to reveal the correct relationship. For the exploration of unknown data sets, our approach helps users easily understand structures and their relations through interacting with the rendering image and the contour tree, and thus improve understanding of volumetric data.

Compared with the work in [27], we analyzed two types of relations: neighboring and inclusion. In fact, the neighboring relation proposed in our approach includes both separate and touch relations in [27]. Furthermore, our approach is not viewpoint dependent as used in [27]. Also our approach does not consider overlap relation as shown in [27]. The contour tree is a data structure to depict topological relations more directly than the relation graph presented in [27]. These properties allow the proposed approach to reveal and analyze relations in volumetric data more effectively.

5.6 Summary

This chapter utilized topological properties derived from the contour tree to represent and depict various relationships inside a data set. We investigated structural relationship preserved mapping as a technique to enhance and improve volume rendering based 3D data set analysis. Usually, not all relationships between objects can be depicted completely in a 2D image view during data analysis. We defined and depicted two typical relationships of inclusion and neighboring in volume rendering respectively. The experimental results showed that structural relationships in a 3D data set are critical features used to help the user to understand data sets. They were depicted and preserved in renderings perceptually using various graphics representations. The contour tree controlled structural relationship depiction allows the user to perceive structural relationship in a more direct way. The user can also control how and what structural relationships are depicted in the pipeline. The presented approach enhances the information provided by conventional rendering techniques. The presented approach extended applications of the contour tree in volume rendering.

Chapter 6

Contour Tree Controlled Automatic Transfer Function Generations

As presented in previous chapters, the contour tree can be simplified to an appropriate size to represent the topology of a data set. It can also be used to depict topological relationships inside a data set in volume rendering, which enhances the information provided by common rendering techniques.

However, in volume rendering, the transfer function generation is still a challenging task for comprehensive renderings. This chapter introduces topological properties derived from the contour tree in transfer function design process. It presents a model, named as *a residue flow model*, to automate opacity transfer function generations. The generated opacity transfer functions can automatically depict inclusion relationships of structures. Furthermore, an automatic color transfer function generation scheme is also developed by utilizing topological properties derived from the contour tree to automatically select harmonic colors. The generated color transfer functions convey meaningful relations among structures instead of arbitrarily defined colors based on users' preferences. The contributions of this chapter are as follows:

- A residue flow model based on Darcy's Law is proposed to control opacity residue flow between branches in the contour tree.

- Based on the residue flow model, we design a comprehensive framework dedicated to automate transfer function generations controlled by topological attributes derived from the contour tree.

- Contour tree controlled harmonic color transfer function is proposed to get visual aesthetic appeal of rendering in order to improve data analysis efficiency.

6.1 Introduction

The ultimate goal of volume visualization is to provide useful insights into volumetric data. Direct volume rendering is one of the effective and flexible visualization methods for three-dimensional volumetric data. Despite the proliferation of volume rendering and manipulation techniques, the key to comprehensible volume rendering still lies in the design of effective transfer functions, which map scalar values to specific colors and opacities. Transfer functions assume that scalar values map directly to physical properties such as tissue types. Thus, they are crucial in the understanding of the overall volumetric data and individual features contained within the volume space. Various approaches have been developed to automate or ease specification of transfer functions. Details of various approaches of transfer function design are investigated in Section 2.6.

Conventional approaches often use boundary information to automate transfer function specifications [60, 61]. Despite various data features (e.g. derivatives, curvatures, texture, size) being used to define transfer functions, most of the current volume rendering methods highly depend on manual specifications (at least in some degree). This process is time-consuming and comes short in repeatable results. In addition, volumetric data often contain nested inner structures. This commonly seen inclusion relationship heavily affects understanding of volumetric data. Conventional transfer functions cannot automatically fully clarify such inner structures. Thus, a systematic and automatic scheme to specify transfer functions while revealing inclusion relationships is highly desirable to greatly facilitate the volume rendering process.

In addition to providing topological features of a volume, the contour tree also acts as a visual index to segment a volume [122]. Takahashi et al. [106, 107] and Takeshima et al. [108] used topological attributes derived from the contour tree to define transfer functions. However, they still applied transfer functions to a data set globally according to fixed topological indices (e.g. depth of topological nesting). Conventional transfer function approaches cannot distinguish between distinct features that share the same scalar value. To overcome this problem, Weber et al. [122] used the contour tree to index various subregions of a volume and specify transfer functions locally for individual subregions. However, as there are often many branches in a contour tree, it is impractical and time-consuming to define transfer functions for each subregion corresponding to a branch of the contour tree one-by-one manually. The underlying weakness of these methods is that transfer functions are still specified manually through user interactions. They do not consider how to use the topology to automate transfer function generations. The principle technical challenges for these problems include how to generate transfer functions that are used to get a wide spread of dissimilar output renderings and to reveal inclusion relationship between subregions. Users also need to decide what measures are used to control transfer function differences between subregions based on the contour tree.

Moreover, any of different subregions in volumetric data may bear interesting intensity variations which need to be visually represented in a faithful manner. Color transfer functions allow aesthetically appealing rendering in order to improve data analysis efficiency. Conventional color transfer functions are parameterized by values that segment the data into different subranges, which are arbitrarily assigned with different colors based on experiences and personal preferences [65, 64]. Thus color is used to only identify subranges of the data, and not to convey any quantitative relations among the data. One popular design aspect of color in graphics and image is color harmony [120]. Wang et al. [119] presents a harmonic color design scheme for illustrative visualization. The scheme uses manually specified parameters such as importance factors to control color components (hue, vividness, lightness) selection for segmented data rendering. The principal weakness of this scheme is that the manually specified parameters have no direct relations with features derived from the data set itself. Vividness and lightness are still selected arbitrarily based on users' preferences in some degree. Thus, an automatic scheme to specify color transfer functions by incorporating data features (e.g. topological features) is highly desirable, and expected to greatly improve the visual aesthetic appeal of rendering.

This chapter presents an approach for automating transfer function generations by utilizing topological attributes derived from the contour tree. The contour tree acts as a visual index to access segmented volume, and captures associated global topological attributes involved in volumetric data. The objective of this chapter is to deliver a new transfer function generation paradigm based on the contour tree. The generated transfer functions can depict inclusion relationship between structures and maximize rendering differences between them. The proposed approach has advantages. It allows more efficient automation of transfer function generations. Exploration on the data is carried out through controlling of residue flow rate, instead of complicated low-level transfer function parameter adjustments used in conventional approaches.

6.2 Framework Overview

The proposed pipeline in this chapter for automatic transfer function generations consists of several processes. An overview of the framework is shown in Figure 6.1. Given a volume data set, the contour tree is created and simplified. The contour tree is used as a visual index of various subregions/structures of the data set, and different topological attributes are derived from the contour tree. These topological attributes are then used to control transfer function generations. Based on the contour tree, an opacity residue flow model is set up. The opacity residue flow model treats opacity transfer function generation as a kind of liquid (e.g. water) flow process between branches of the contour tree: opacity is treated as water, and topological attributes are used to control opacity flow between branches. Interfaces are provided to allow users to control the opacity flow rate and the shape of local transfer function in a branch, and thus decide the final transfer function. Meanwhile, topological attributes are also used to generate harmonic colors for color transfer functions. The generated opacity and color transfer functions are then used to render the data set.

In this framework, users can change the opacity flow rate and local transfer function shape to refine rendering results. The opacity flow rate and local transfer function shape are used in the pipeline to keep topological relationships in the rendering, which are represented in the contour tree. This chapter focuses on processes of how the contour tree controls transfer function generations in the pipeline. The details of each process will be covered in the later sections.

6.3 Residue Flow Model

In this section, we model the distribution of opacity between branches in the contour tree as a multi-stream water flow process. When the water passes through a branch, some of it is re-directed by the branch and residues are delivered to child branches at the next depth level. This section presents Darcy's Law as a basic theory to get residues when

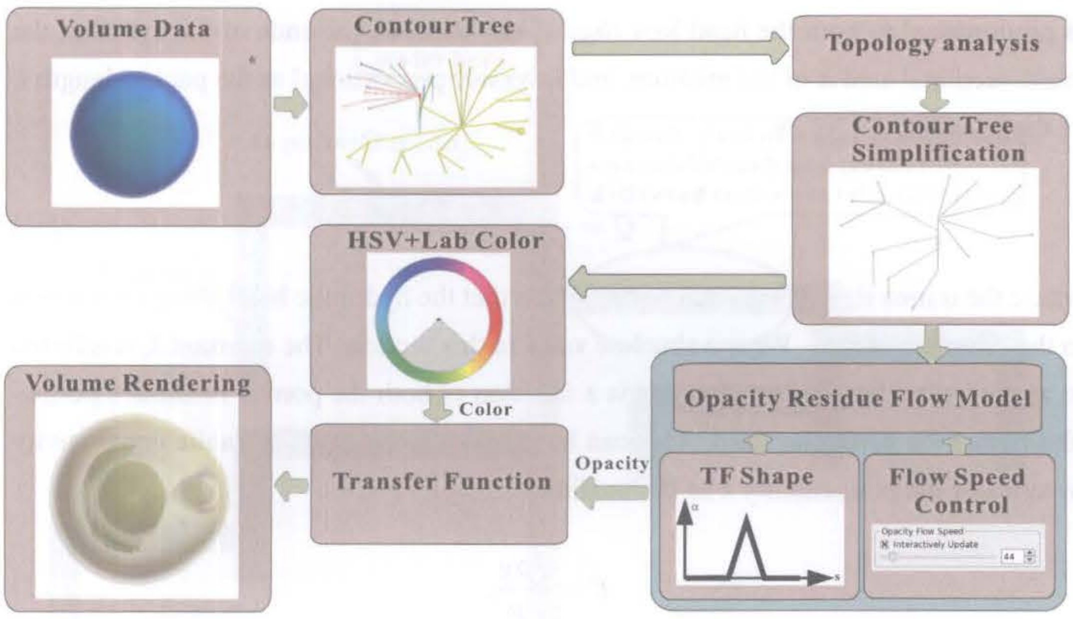


Figure 6.1: Flow chart showing the pipeline for automatic transfer function generations.

the water passes through a branch, and gives an overview of residue flowing between branches.

6.3.1 Darcy's Law

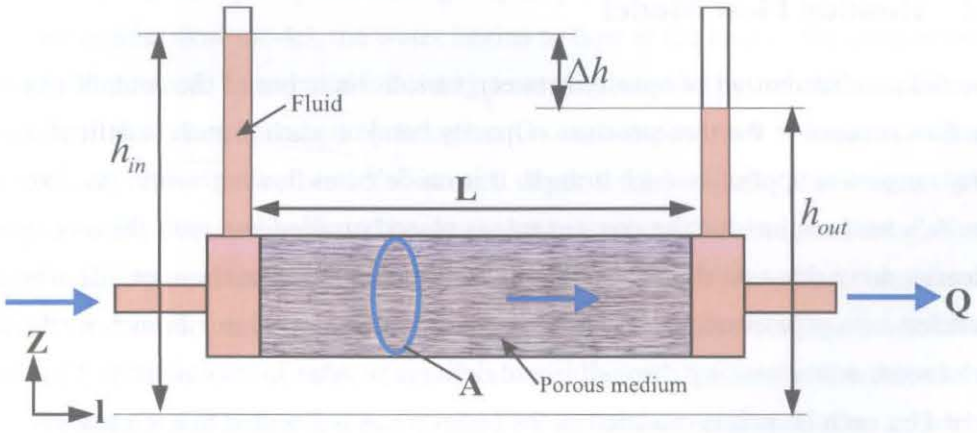


Figure 6.2: Setup of Darcy's experiment in one dimensional case.

Darcy's Law [8, 67] is a generalized relationship for fluid flow in porous medium. Figure 6.2 shows the setup of Darcy's experiment in one dimensional case. Darcy's Law demonstrates that the volumetric flow rate Q of the fluid, through the porous medium, is proportional to both the head loss $(h_{out} - h_{in})$ between the ends of the cylinder, the cross-sectional area A of the medium, and inversely proportional to the packed length L of the cylinder:

$$Q = -KA\frac{\Delta h}{L}, \tag{6.1}$$

where the minus sign on the right hand reflects that the hydraulic head always decreases in the direction of flow. We use absolute value in this section. The constant K is referred to as the hydraulic conductivity, and is a function of both the porous medium's permeability and the particular fluid. This can be seen by writing the hydraulic conductivity in terms of the permeability k of the medium:

$$K = k\frac{\rho g}{\mu}, \tag{6.2}$$

where ρ and μ are the fluid's density and viscosity, and g is the acceleration due to gravity. [57] summarizes average hydraulic conductivity K for general materials. This chapter uses sand as the porous medium, and the value of K for water flowing through sand is 300 [57]. The choice of materials does not affect rendering results except for the use of different flow rates in the rendering pipeline.

6.3.2 Residue Flow Model

We model the distribution of opacity between various branches of the contour tree as a water flow process in the tree structure. Opacity band on each branch is defined as the opacity range size applied to each branch. It is modeled as flowing water. As shown in Figure 6.3, we imagine that the contour tree is placed upsidedown with the root upside and leaves downside. At the same time, the water is poured at the root. The branch is modeled as a pipe structure with porous medium inside. Every branch in the tree absorbs some water passing through it and delivers residue to its children. The water absorbed by each branch is modeled as the opacity band allocated to each branch.

The Darcy's Law is adapted to control the opacity distribution between branches in the contour tree: the packed length L of the cylinder in the Darcy's Law is modeled as the persistence of the branch; the cross-sectional area A of the medium in the Darcy's Law is modeled as the number of child branches; the volumetric flow rate Q of the fluid is modeled as the opacity flow rate in the branch; and Δh is modeled as the opacity

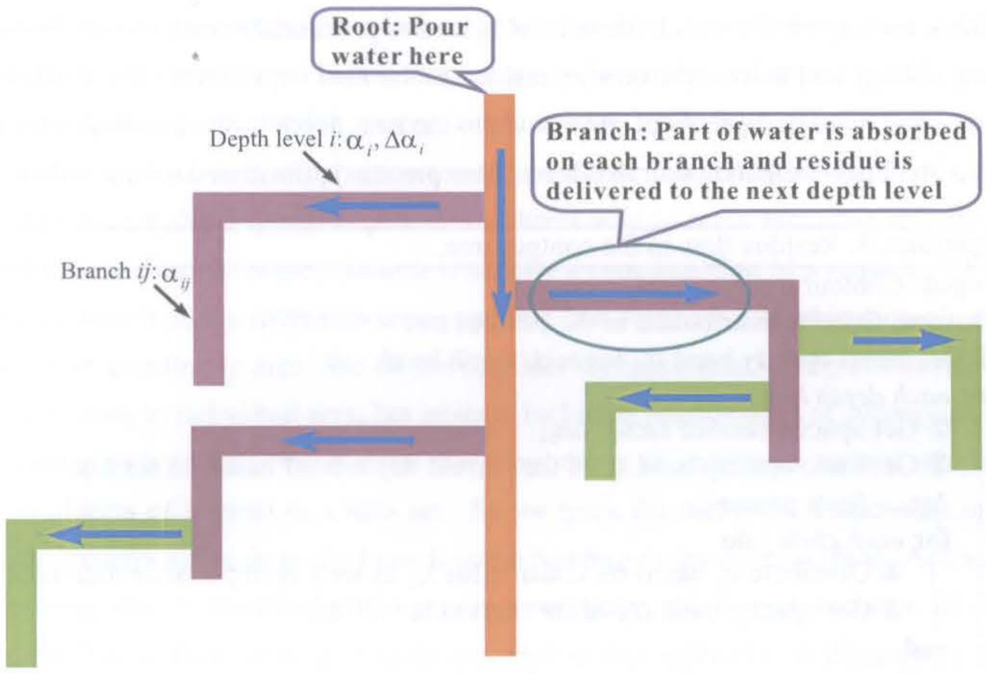

Root: Pour water here
Depth level i: α_i, $\Delta\alpha_i$
Branch: Part of water is absorbed on each branch and residue is delivered to the next depth level
Branch ij: α_{ij}

Figure 6.3: Water is poured at the root of the tree and flows throughout the tree. Part of water is absorbed by each branch and residue is delivered to the next depth level.

residue factor in the branch. So given the opacity flow rate Q, the opacity residue factor Δh of each branch can be obtained from Equation 6.1. The opacity residue factor is used to modulate the opacity band and get opacity residue for the next depth level.

In the residue flow model, the water begins to flow at the root of the contour tree as shown in Figure 6.3. Some of the flowing water is absorbed by branches, and there is the residue that is not absorbed by branches on each depth level. We assume that the water initially absorbed by branches at each depth level is equal. This means that the water is initially evenly distributed between depth levels. After this, the Darcy's Law is applied to each parent branch sequently from the root branch in order to get residues and then deliver them to child branches at the next depth level. The residue flows down to the next depth level, and combined together with the initially evenly distributed water. This combined water is then partly absorbed by the branch and the residue is delivered to the next depth level further. This process is continued until reaching leave branches.

Based on this model, the opacity band on each branch is allocated using following procedure: Initially, the opacity range used to render the overall data set is divided into equal-size bands based on the maximum depth value. This allows the opacity band to be evenly distributed between each depth level. Then from the root of the contour tree, the initial opacity band on each depth level is further processed in order to distribute it among siblings and deliver the opacity residue to the next depth level. The residue of opacity band on each depth level sinks down to the next depth level and contributes for the opacity band computation of that level. This process is illustrated in Algorithm 3.

Algorithm 3: Residue flow in the contour tree.

```
Input: Contour tree
Output: Opacity distribution in the contour tree
① Get initial opacity band $\alpha_d$ for each depth level;
for each depth level i do
    ② Get opacity residue factor $\Delta h_i$;
    ③ Get basic opacity band $\alpha_i$ of the current depth level based on residue
    $\Delta\alpha_{i-1}$ from parents;
    for each child j do
        ④ Distribute $\alpha_i$ based on scalar value $s_{ij}$ as well as importance measures;
        ⑤ Get opacity band $\alpha_{ij}$ of the current branch $b_{ij}$;
    end
    ⑥ Get residue $\Delta\alpha_i$ of the current depth level;
end
```

Algorithm 3 shows that the residue is generated on the parent depth level and flows to the next depth level. The residue is also distributed between siblings on each depth level. The details of each step in Algorithm 3 are presented in later sections.

6.4 Opacity Transfer Function Generations

Objectives of Opacity Transfer Function (OTF) in this chapter include: maximize rendering differences between various objects inside volume data, and represent corresponding inclusion relationship between objects at the same time. The branch decomposition is used to represent the topology of volume data in this chapter. This section first presents how the contour tree is used to represent inclusion relationship of objects inside volume data. Then the residue flow model is used to control the flow of opacity band in the contour tree in order to specify OTF of volume data.

6.4.1 Representation of Inclusion Relationship

Chapter 5 defined and analyzed various relationships (e.g. inclusion relationship) in a data set by utilizing topological properties derived from the contour tree. Specifically, it was shown that the inclusion relationship of structures depends on nesting depth of branches in the contour tree. In this chapter, the inclusion relationship is used to aid the automation of transfer function generations in volume rendering.

We use the branch decomposition to analyze the inclusion relationship in this chapter. The branch decomposition provides a direct way to track inclusion relationship based on branches. Because a branch is usually a concatenation of a sequence of arcs in the contour tree, the subregion corresponding to a branch is a combination of subregions corresponding to arcs. We do not consider inclusion relationships of subregions corresponding to individual arcs, but analyze inclusion relationships of subregions corresponding to branches. We assume that there exists an outermost surface which includes objects of interest in a data set. So we track the inclusion relationship in the branch decomposition from the branch corresponding to the outer surface. According to the basic theory described in the previous paragraph, this branch includes all of its children. The further tracking of inclusion relationship is performed along every child branch. The parent branch includes its child branch, and there is no inclusion relationship between child branches which have the same parent. In order to emphasize inner structures and deemphasize outer structures, branches with lower depth value are applied with lower opacity and branches with higher depth value are applied with larger opacity. This forms the basis of the opacity residue flow presented in later subsections.

6.4.2 Residue Flow and Opacity Transfer Function Design

Typical approaches used to render inner structures and depict inclusion relationships in volume rendering include: volume clipping [123] and transparent/semitransparent surfaces [111, 113]. Comparatively, transparent/semitransparent surfaces are more often used in volume rendering because they can display inner structures and outside structures simultaneously. Users can perceive spatial information of different layers while emphasizing inner structures by using transparent/semitransparent surfaces. This kind of presentation thus improves users' understanding on volume data. This chapter uses the approach of transparent/semitransparent surfaces to depict inclusion relationships between structures by designing opacity transfer functions. In volume rendering, structures with smaller opacities are displayed more transparently than structures with larger opacities. In order to display inner structures and outside structures at the same time, larger opacities need to be applied to inner structures while smaller opacities are applied to outside structures. Based on these fundamentals, the goals of the opacity transfer function design based on the contour tree include: to maximize differences between branches depending on topological attributes, to depict inclusion relationships between branches as clear as possible, as well as to allocate higher and larger range of opacities to inner structures to emphasize inner structure. In order to meet these goals, we physically model the distribution of opacity in the contour tree as a water flow process in an upsidedown tree structure in the residue flow model.

Based on the residue flow model, when the water flows from root to leaves in the upsidedown tree, some of water are absorbed by branches, residues continuously flow to branches on the next depth level. This results in that the root and branches close to the root of the tree get smaller amount of the water and branches far from the root of the tree get larger amount of the water. Because the contour tree can be used to represent inclusion relationships between structures as presented in Section 5.3 and Section 6.4.1, inclusion relationships between structures can be depicted by designing the opacity distribution between branches in the contour tree. If the water is regarded as the opacity allocated to each branch of the contour tree, the allocation of water in an upsidedown tree can ideally meet goals of the opacity transfer function design based on the contour tree. The water absorbed by each branch is regarded as the opacity allocated to that branch, while residues continuously flow to branches on the next depth level. The flowing of residues is used to maximize differences of the water allocation between branches, and thus control the difference of opacities between branches. That means branches on the higher depth level get more water than those on the lower depth level. The result is that inner structures can be rendered more opaquely than outside structures and thus are emphasized. According to these principles, the following sections present how the residue flow model is used to distribute opacities between branches in the contour tree during the opacity transfer function generation process.

6.4.3 Residue Flow from Parent to Children

In order to distribute opacity between branches and thus generate OTF of volumetric data, we first divide the opacity range applied to volume data into equal segments based on the maximum depth of the contour tree. The initial value of each opacity band is computed using Equation 6.3:

$$\alpha_d = \frac{\alpha_r}{d_{\max}}, \tag{6.3}$$

where α_d is the initial opacity band used to render a subregion corresponding to a branch on each depth level, α_r is the opacity range used to render the overall volume data set, $d_{\max}$ is the maximum depth value of the contour tree.

As mentioned in the residue flow model in the previous section, the opacity band applied to each branch is modeled as water and the branch is modeled as a pipe with a porous medium inside. According to Darcy's Law, the absolute head loss value Δh_i between the ends of the current branch in the contour tree is computed by Equation 6.4:

$$\Delta h_i = \frac{1}{K} \cdot \frac{Q \cdot p_i}{n_i^c}. \tag{6.4}$$

where $\Delta h_i \in [0.0, 1.0]$, p_i is the persistence of the current branch, n_i^c is the number of children of the current branch.

In this chapter, Δh_i is used as a factor to evaluate opacity band absorbed by the branch and residue delivered to the next depth level. The opacity band and residue on the ith depth level are computed by Equation 6.5 and Equation 6.6 respectively:

$$\alpha_i = (\alpha_d + \Delta\alpha_{i-1}) \cdot (1 - \Delta h_i), \tag{6.5}$$

$$\Delta\alpha_i = (\alpha_d + \Delta\alpha_{i-1}) \cdot \Delta h_i, \tag{6.6}$$

where $i \geq 1$, $\alpha_0 = \alpha_d \cdot (1 - \Delta h_0)$, and $\Delta\alpha_0 = \alpha_d - \alpha_0$. α_i is the opacity band applied to branches on the ith depth level. $\Delta\alpha_i$ is the residue generated on the ith depth level.

The opacity residue $\Delta\alpha_i$ flows from lower depth level to higher depth level in the contour tree. As mentioned, branches on the lower depth level correspond to outer structures and branches on the higher depth level correspond to inner structures. The flow of the opacity residue in the contour tree allows users to apply larger opacity band to inner structures than outer structures, and thus emphasize inner structures in volume rendering.

6.4.4 Residue Flow Among Siblings

The opacity band and residue flowing to branches on each depth level need to be distributed further between sibling branches on that depth level. One of objectives of the further distribution of opacity band and residue is to maximize differences between sibling branches. Various sibling branches have different extreme and saddle values. The saddle value of each branch decides its exact location on the depth level it resides. Importance values of persistence, volume and hypervolume of each branch are also contributed to enhance differences between sibling branches. Considering these factors, the final opacity band applied to each branch is computed by Equation 6.7:

$$\alpha_{ij} = \alpha_i \cdot g_{sb}\left(p_{ij}, v_{ij}, hv_{ij}\right) \cdot g_{sd}\left(s_{ij}\right), \quad (j = 1, \ldots, n_i^s), \tag{6.7}$$

where α_{ij} is the opacity band applied to the jth sibling branch on the ith depth level. α_i is the basic opacity band on the ith depth level and computed by Equation 6.5. p_{ij}, v_{ij} and hv_{ij} are persistence, volume and hypervolume respectively of the jth sibling branch on the ith depth level. g_{sb} is the function used to control distribution of opacity band between siblings based on importance values p_{ij}, v_{ij} and hv_{ij}. n_i^s is number of siblings on the ith depth level. g_{sd} is the function used to control distribution of opacity band between siblings based on saddle value s_{ij} of each branch.

In order to evaluate influences of various importance values of a branch on the opacity band and residue flow, we use a concept of importance triangle [128] to combine various importance values together in order to modulate opacity band. The size (area) of the importance triangle is used as the solution to the function g_{sb} in Equation 6.7 and computed by Equation 6.8:

$$g_{sb}\left(p_{ij}, v_{ij}, hv_{ij}\right) = I_{ij} = \frac{1}{2}\sqrt{\left(hv_{ij} \cdot p_{ij}\right)^2 + \left(v_{ij} \cdot p_{ij}\right)^2 + \left(hv_{ij} \cdot v_{ij}\right)^2}, \tag{6.8}$$

where p_{ij}, v_{ij} and hv_{ij} are firstly normalized by corresponding local maximum within sibling branches respectively on the ith depth level.

On the ith depth level, sibling branches have different saddle values. Branches with lower saddle values are close to the start of the root of the contour tree, and branches with larger saddle values are far away from the start of the root of the contour tree. Because a branch is a concatenation of a sequence of arcs in the general contour tree, we apply a smaller opacity band to branches with lower saddle values and a larger opacity band to branches with larger saddle values, in order to emphasize branches with larger saddle values and represent the inclusion relationship. Based on this observation, the solution to the function g_{sd} in Equation 6.7 is computed by Equation 6.9:

$$g_{sd}\left(s_{ij}\right) = \frac{s_{ij} - s_i^{\min}}{\Delta s_i}, \tag{6.9}$$

where $\Delta s_i = s_i^{\max} - s_i^{\min}$, $s_i^{\max} = \max(s_{i0}, s_{i1}, \ldots, s_{i,n_i^s})$, $s_i^{\min} = \min(s_{i0}, s_{i1}, \ldots, s_{i,n_i^s})$.

Finally, the actual opacity range applied to the jth branch on the ith depth level is:

$$\alpha_{ij}^l = \alpha_{\min} + \sum_{m=0}^{i-1} \alpha_m, \tag{6.10}$$

$$\alpha_{ij}^h = \alpha_{\min} + \sum_{m=0}^{i-1} \alpha_m + \alpha_{ij}, \tag{6.11}$$

where α_{ij}^l is the lower opacity value applied to the current branch. α_{ij}^h is the upper opacity value applied to the current branch. $\alpha_{\min}$ is the minimum opacity value applied to the overall data set.

After getting the opacity range of each branch, one of the transfer function shapes is selected to generate actual opacity values of each branch. In this chapter, we provide five typical transfer function shapes as shown in Figure 6.4. Each transfer function shape has its own features: the hat-like shape aims to reveal isosurface-like shapes in the data set; the linear shape can show other structures except isosurfaces. Other transfer function shapes have similar features with the linear shape and hat-like shape. The choice of the transfer function shape depends on the complexity of volumetric data. If a data set contains complex structures inside, the hat-like shape is more effective to reveal isosurfaces of structures to differentiate them. Otherwise, the linear shape and other shapes can also reveal inner structures effectively.

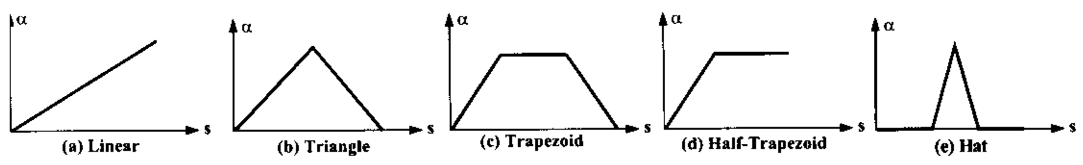


Figure 6.4: Various transfer function shapes used in a branch.

6.5 Color Transfer Function Generations

A Color Transfer Function (CTF) is used to help users to differentiate structures with colors. Automation of CTF generations is still a challenging task, which requires users to generate colors for various structures meaningfully. This section firstly presents various color spaces, and then introduces topological features derived from the contour tree into the pipeline of CTF generations. With the proposed approach, the CTF generation becomes a meaningful process determined by topology instead of arbitrarily user's preferences dependent process.

6.5.1 Color Spaces

Different color spaces are used to describe colors. The RGB (Red, Green, Blue) color space (see Figure 6.5) describes a color completely, but it does not carry direct semantic information about the color, a person cannot visualize a color given its RGB value; they only tell how red, green, or blue it is. Additionally, in the RGB space, equal geometric distances within the space do not, in general, correspond to equal perceptual changes in color. Due to this perceptual non-linearity, the RGB space is not useful for direct color comparison based on geometric separation within the RGB cube.

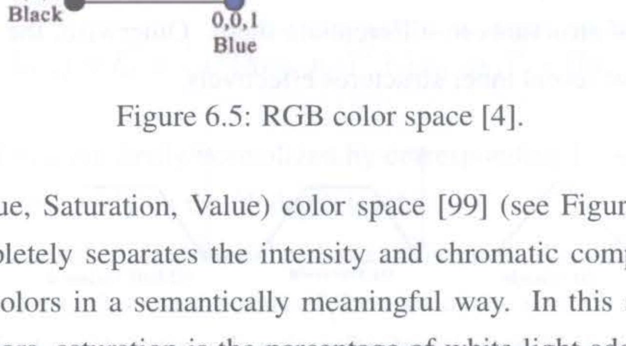


Figure 6.5: RGB color space [4].

The HSV (Hue, Saturation, Value) color space [99] (see Figure 6.6) is based on a cone and completely separates the intensity and chromatic components. In doing so, it describes colors in a semantically meaningful way. In this space, hue is used to distinguish colors, saturation is the percentage of white light added to a pure color and value refers to the perceived light intensity. The advantage of the HSV color space is that it is closer to human conceptual understanding of colors and has the ability to separate chromatic and achromatic components.

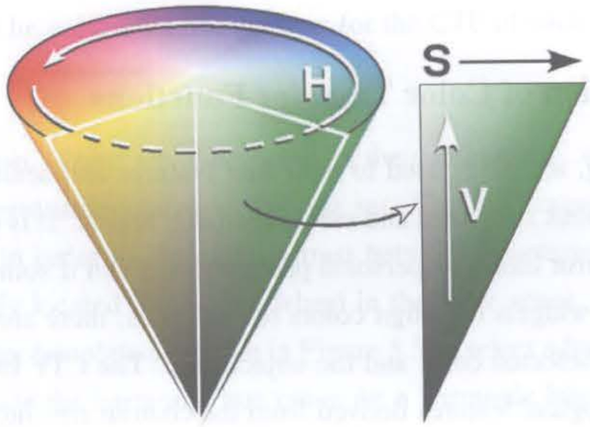


Figure 6.6: HSV color space [4].

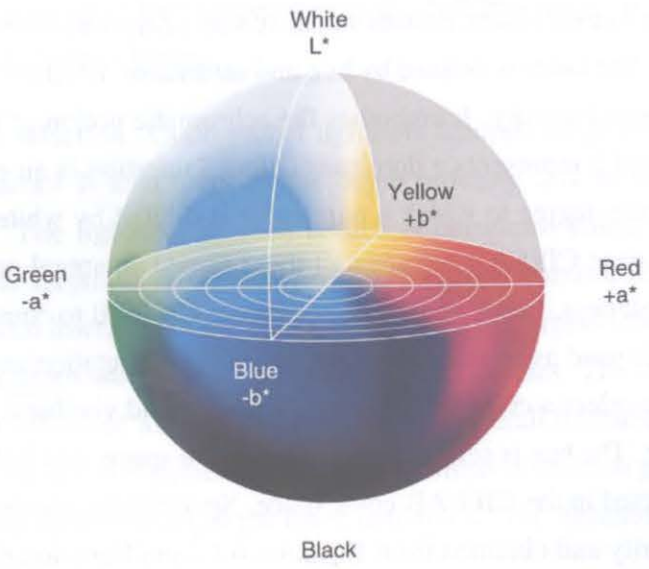


Figure 6.7: CIELAB color space [5].

The CIE L*a*b*, also called CIELAB or, for short, Lab color space, (see Figure 6.7), is designed to approximate human vision and defines colors more closely to the human color perception than the HSV color space. It is an approximately uniform color space, In a uniform color space, the differences between points plotted in the color space correspond to visual differences between the colors plotted.

Because color is one of the important factors which influence perception of objects in volume rendering, the perceptive color spaces of CIELAB and HSV are considered in the proposed transfer function specification pipeline.

6.5.2 Automation of Color Transfer Functions

In volume rendering, a CTF is used to map data features and scalar values to colors in order to label various structures and create aesthetic appeal. It is often defined arbitrarily by trial and error based on personal preferences. Even if some transfer function approaches provide widgets to assign colors to structures, there are no theoretical relations between the selected color and the object [64]. The CTF in this chapter aims to incorporate topological features derived from the contour tree into the CTF specification, and focus attention to inner structures. It also aims to label different structures with various colors, and automate CTF generation to create harmonic renderings. Thus the CTF specification becomes a meaningful but not arbitrary process.

When viewing a color object, human visual system characterizes it by its brightness and chromaticity. The latter is defined by hue and saturation. Brightness is a subjective measure of luminous intensity. It embodies the achromatic notion of intensity. Hue is a color attribute and it represents a dominant color. Saturation is an expression of the relative purity or the degree to which a pure color is diluted by white light [28]. The perceptual color space CIELAB includes all these three perceptual components: hue, saturation and brightness. In our approach, we use the CIELAB to compute colors. The HSV color space is used as an interface between color computation and the CTF specification. Thus we select a color triple of hue, lightness and vividness for each branch of the contour tree. The hue is selected in the HSV color space, and lightness as well as vividness are selected in the CIELAB color space. Specifically, vividness is defined as color's relative purity and obtained from Equation 6.12 and Equation 6.13 [119]:

$$chrom(h,s,v) = \sqrt{a^2+b^2}, \tag{6.12}$$

$$vivid(h,s,v) = chrom(h,s,v)/chrom(h,1,1), \tag{6.13}$$

where a and b are color components in the CIELAB color space.

This chapter extends Wang et al.'s approach [119] in the following ways: various hues are selected for each branch, and topological attributes (e.g. importance value) derived from the contour tree instead of specified manually as in [119] are used in lightness and vividness selection. The contour tree is used to automate the overall color selection process. The color selection process for the CTF of each branch is presented as follows:

Step 1 Hue selection. There are N branches in the contour tree, which correspond to various subregions/structures in the data set. This step creates various hues for each branch in order to increase contrast between structures. Different N hues may be evenly located on the hue wheel in the HSV space. We may also use a harmonic color template as shown in Figure 5.5 to select a hue type and limit hue positions inside the harmonic hue range on a harmonic hue wheel. We start to specify colors to each branch from the root of the contour tree. The choice of the color hue type depends on users' preferences and complexity of volumetric data. If a volume data contains large number of objects inside, users may select a hue type with large hue range or even the full hue range in order to increase differences between objects.

Step 2 Lightness selection. Good global lightness contrast helps to discriminate different features. It also helps in the depth ordering task to discriminate inner structures. The lightness of each branch is computed based on its depth and subtree size. Depth of the branch helps to highlight contrast of inner structures. Branches with lower depth are applied with higher lightness values, and branches with higher depth are applied with lower lightness values. The subtree size allows to consider affects of number of objects on the overall lightness. Branches with larger subtree size are applied with higher lightness in order to get higher contrast between structures and highlight inner structures. The lightness is computed by Equation 6.14:

$$L_{ij} = L_{\max} - \frac{d_{ij}}{d_{\max}} \cdot \left(1 - \frac{n^s_{ij}}{n^s_{\max}}\right) \cdot (L_{\max} - L_{\min}), \qquad (6.14)$$

where L_{ij} is the lightness applied to the current branch, n^s_{ij} is the size of subtree beginning from the current branch, $n^s_{\max}$ is the whole contour tree size.

Step 3 Vividness selection. Vividness can be a strong depth cue because decreasing vividness gives the effect of seeing in a fog [93]. It guides the observer to the most important features of the data. The vividness of each branch is computed based on its depth and importance value. For the former, we apply lower vividness values to branches with lower depth in order to focus users' attention to inner structures. The importance value of each branch is also considered and more important branches are colored with a higher vividness. So this specification can be represented with Equation 6.15:

$$V_{ij} = \left(V_{\min} + \frac{d_{ij}}{d_{\max}} \cdot (V_{\max} - V_{\min})\right) \cdot I_{ij}, \tag{6.15}$$

where V_{ij} is the vividness applied to the current branch, d_{ij} and $d_{\max}$ are the depth of the current branch and maximum depth of the current tree, I_{ij} is the importance value of the current branch computed by Equation 6.8, $V_{\min}$ and $V_{\max}$ are the minimum and maximum vividness specified by the user which determine the overall vividness of the visualization.

After the hue, lightness and vividness have been selected, we convert these (h, L, V) triples to their (h, s, v) equivalents and finally to RGB for the transfer function. To obtain (h, s, v) from a given (h, L, V), we use the binary search in a given hue slice in the HSV space [119].

6.6 Experimental Results and Discussions

We conducted experiments on various data sets to demonstrate the effectiveness and utility of the proposed approach in volume rendering. Our system was run on a Ubuntu platform on a Dell machine (Intel Core2Duo CPU E4400, 3GiB RAM) equipped with an NVIDIA GeForce 8300GS graphics card.

We first used the proposed approach to automate the transfer function generation for the "nucleon" data set, a $41 \times 41 \times 41$ voxel data set resulting from a simulation of the probability distribution of a nucleon in the atomic nucleus ^{16}O, see `http://www.volvis.org/`. Figure 6.8 shows the result of the experiment. In this figure, the harmonic color hue T-Type and linear transfer function shape are used. Figure 6.9 shows the transfer function set generated in this experiment. By observing the result, the nested inner structures and outer layers are clearly revealed. This figure clearly

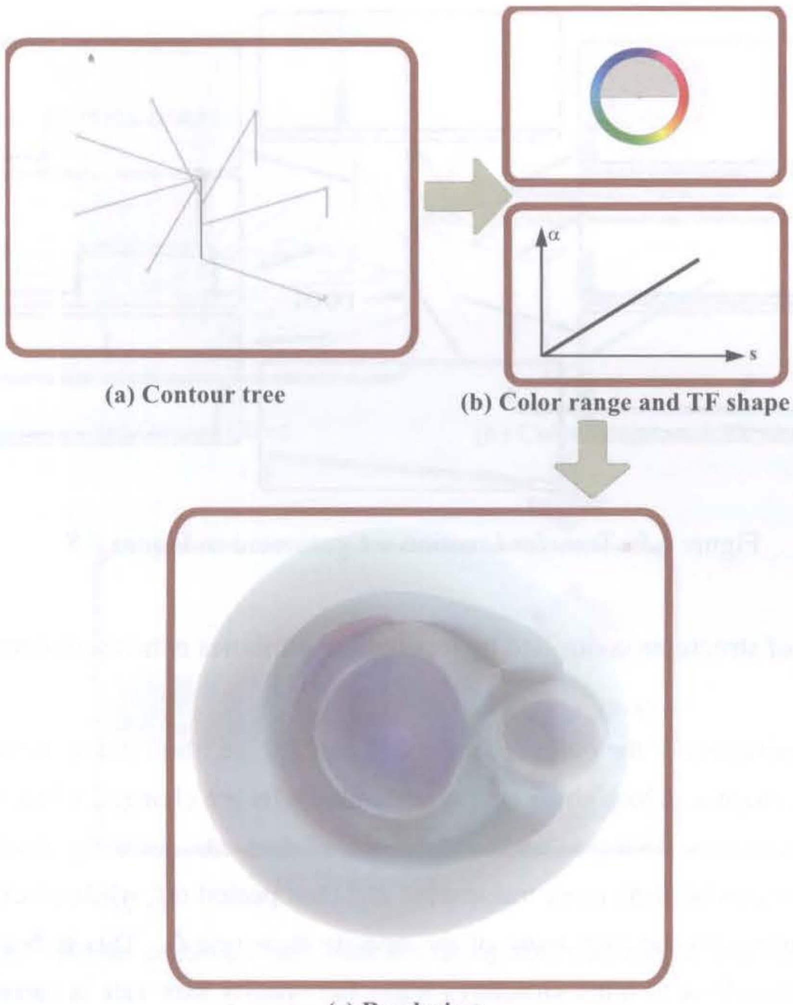


Figure 6.8: Volume rendered nucleon data set using harmonic hue T-Type and linear transfer function shape.

shows various layers from outside to inner side with increased opacity, and increased vividness and decreased lightness. Harmonic colors give users aesthetic appeal and enhance understanding of structures.

Figure 6.10 shows the rendering result of the "fuel" data set, a $64 \times 64 \times 64$ voxel data set resulting from a simulation of fuel injected into a combustion chamber, see `http://www.volvis.org/`. Using the contour tree controlled residue flow model, it is possible to automatically reveal inner structures while rendering surrounding layers using a low opacity. Colors of various structures are differentiated, and inner structures are emphasized and applied with colors with high vividness and low lightness.

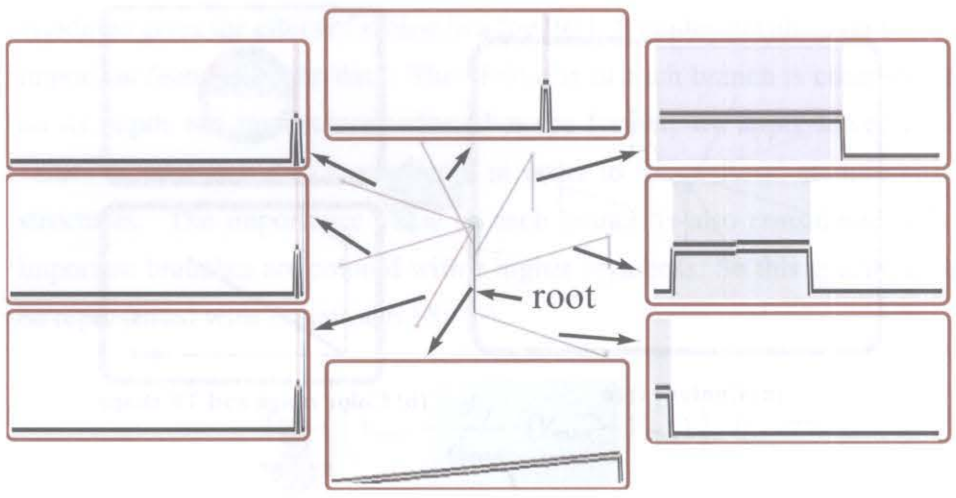


Figure 6.9: Transfer function set generated in Figure 6.8.

Thus colors of structures controlled by topological attributes enhance understanding of structures.

The effectiveness of the proposed approach can also be observed in movies accompanying this chapter, which show how rendering results are changed when the opacity flow rate Q is increased for the "fuel" and "nucleon" data sets. From the movies, we see that outer surfaces become more transparent and then peeled off, while inner structures become clearer with the increasing of the opacity flow rate Q. This is because more opacity residues flow to inner structures when the opacity flow rate is increased. The movies give users a better understanding of the effectiveness of the residue flow model in transfer function generations.

The volume rendering pipeline in this chapter is implemented based on graphics processing unit (GPU) fragment programs. The performance of the system allows real-time exploration of the volumetric data. The processing time of automatically generating transfer functions for a data set depends on the number of branches in the contour tree and size of the data set. For small data sets, such as the "fuel" data set, the number of branches is usually within a manageable size (e.g. less than 90 before simplification and around or less than 20 after simplification). The transfer function can be generated interactively for such data sets. We created movies for "fuel" and "nucleon" data sets to show how transfer functions are generated interactively when changing the opacity flow rate in this chapter. The frame rate (the size of the view port is 400×374 in this chapter) for the "fuel" data set is 36.5fps (20 branches, and original branch number is 86),

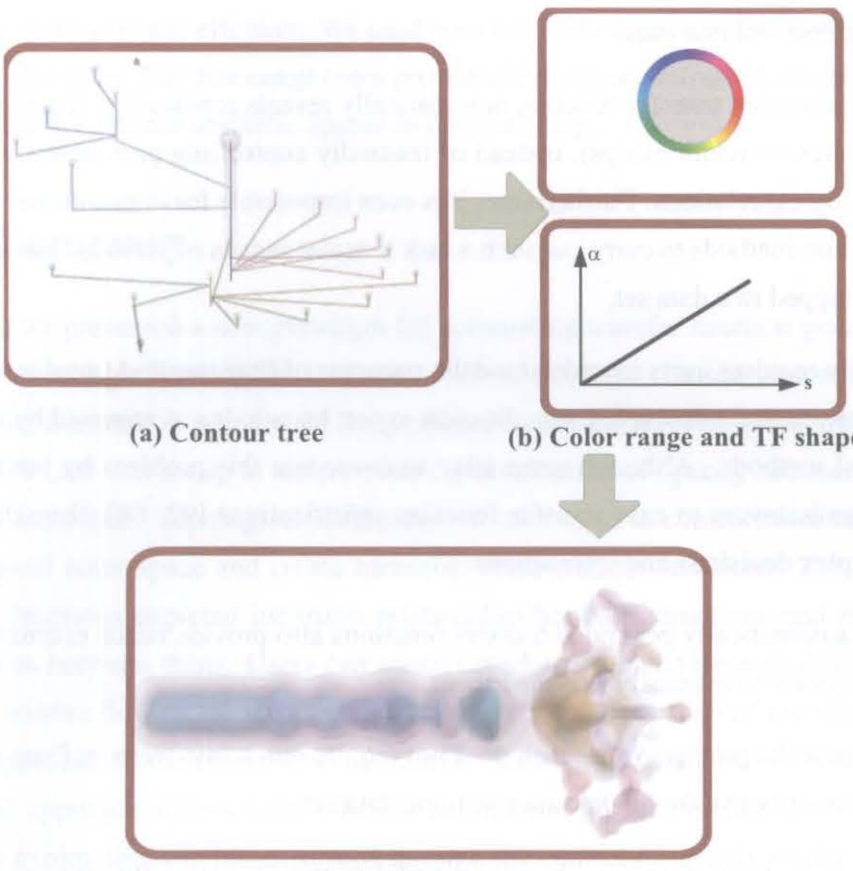


Figure 6.10: Volume rendered fuel data set using full hue range and linear transfer function shape.

and the frame rate for the "nucleon" data set is 55.8fps (8 branches, and original branch number is 40). However, frame rates for larger data sets are still sufficiently high for automatically transfer function generations. For example, the processing time of generating transfer functions for the CT knee data set is 1.27s (10 branches, and original branch number is 954968). After generating transfer functions, users can interactively explore the data sets.

From the experiments, we can see that the proposed approach can effectively generate transfer functions automatically. Compared with conventional transfer function methods, our approach has the following advantages:

- It only requires users to simply control the opacity residue flow rate, instead of time-consuming interactions for tweaking complex transfer function parameters to explore volume data.

- The generated transfer function automatically reveals topological relations (such as inclusion relationships), instead of manually controlling parameters to depict topological relations. Furthermore, it is even impossible for conventional transfer function methods to carry out such a task if scalar ranges of various structures are overlapped in a data set.

- It only requires users to understand the concepts of flow rate and transfer function shape, instead of complex visualization expert knowledge as required by conventional methods. Although some tried to overcome this problem by introducing semantic layers to ease transfer function specifications [92, 89], they still need complex decisions and interactions.

- The automatically generated transfer functions also provide initial estimates even if users are to manually fine tune them.

- Because the proposed approach does not require much involvement from users, it is even effective for an unknown volume data set.

- The advantage of the residue flow model is that it allows the opacity residue to sink down to leaves inside an upsidedown contour tree, and results in larger opacities of branches on the higher depth level. This means that inner structures get larger opacity than outer structures, and thus emphasize inner structures.

In summary, our new approach greatly improved the transfer function generation process, and it would be significantly beneficial to users who are not visualization expert.

Various opacity transfer function shapes are provided in the framework. Opacity transfer functions with a hat-like shape try to capture isosurfaces in the data set. This results in the possibility that users may see through outer surfaces to perceive inner structures in volume rendering. However, other transfer function shapes are also powerful in revealing structures in volumetric data. For example, the linear shape used in Figure 6.8 and Figure 6.10 is effective in depicting inner structures of data sets. Compared with Wang et al.'s method [119] which uses parameters such as importance levels specified manually by users to control color generations, our approach controls color generations based on topological attributes directly derived from the contour tree. This is more meaningful and effective. We used both full hue range color and harmonic color in the experiments. Full hue range color provides high color contrast of structures, while harmonic color creates aesthetic appeal in the rendering.

6.7 Summary

This chapter presented a new paradigm for automating transfer function generations in volume rendering. Topological attributes derived from the contour tree were used to control the automation process. In the proposed approach, a residue flow model based on Darcy's Law was set up to differentiate the distribution of opacity between branches of the contour tree. Topological attributes were also used to control color selection in a perceptual color space and create harmonic color transfer functions. The generated transfer functions depicted inclusion relationship between structures and maximized differences between them. Users can control rendering results through differentiating opacity residue flow rates. Experiments on various data sets showed the effectiveness of our approach in the automation of transfer function generations. In summary, the proposed approach allows more efficient automation of transfer function generation, and data exploration can be performed through the control of opacity residue flow rate rather than complicated low-level transfer function parameters.

Chapter 7

Qualitative Evaluation in Volumetric Medical Image Analysis

In previous chapters, we presented a novel approach to depict structural relationships and automate transfer function generations in volume rendering. It is important to evaluate the extent to which the proposed approach meets the original objectives, and to determine how it compares to existing solutions. It is also important to understand the impact the approach will have on those who will make use of it. We performed some experiments in Chapter 4, Chapter 5 and Chapter 6 to show the effectiveness of the proposed approach in volumetric data analysis. This chapter specifically focuses on evaluating the effectiveness of the proposed approach in more complicated volumetric medical data analysis. It is dedicated to assessing the approach in a qualitative sense by comparing the proposed approach with conventional approaches in volumetric data analysis.

7.1 Methodology

This chapter performs a qualitative analysis of the proposed approach as compared to conventional volume rendering approaches with manual transfer function specifications. More specifically, we used the *Kitware Inc.*'s *VolView 3.2* [2] as a conventional volume rendering tool in our experiments in order to compare the effectiveness of our proposed approach. VolView 3.2 is a widely used interactive system for volume visualization that allows researchers to explore and analyze complex 3D medical or scientific data. It does not provide any automatic schemes for transfer function generations. So transfer functions are generated manually by trial-and-error during the experiment. Other conventional volume rendering approaches could be used in the experiments. The main difference between other approaches and VolView is that other volume rendering tools incorporate more features of data sets and provide various widgets to ease transfer function definitions. However, they share the common disadvantages in volumetric data analysis: they define transfer functions manually through trial-and-error, and have no specific mechanisms to depict structural relationship in transfer function generations (see the investigation in Section 2.6). Because VolView is more widely used by researchers and visualization users, this chapter utilizes VolView to render various data sets in order to show the effectiveness of our proposed approach.

Our evaluation consists of three main components as follows:

- *The ability to depict structural relationships.* As mentioned, structural relationships are significant factors which affect understanding of volume data. This chapter compares the ability to depict the structural relationships of our approach with conventional volume rendering approaches, in order to show the effectiveness of our approach in volumetric data analysis.

- *The ability to automate transfer function generations.* Automation of transfer function generations is highly demanded in volume rendering based data analysis. This ability decides the effectiveness of the approach in volumetric data analysis. In comparing this ability of our approach with conventional approaches, we present the transfer function generation process and the requirements for creating a comprehensive rendering with different approaches.

- *The degree of involvement from users.* In order to generate transfer functions and create a comprehensive rendering in volume visualization, users often need to do interactions on various parameters. The degree of this involvement affects the ease and effectiveness of the approach in volumetric data analysis.

7.2 Experimental Results

To demonstrate how the proposed approach performs on volumetric data analysis, we used various complex medical data sets, such as CT knee data set, CT foot data set, and MR brain tumor data set. In each experiment, the properties of the data set are firstly presented. Difficulties to visualize the data set are also discussed. The experimental results are then presented to show how the proposed approach renders the data set effectively. Our system was run on a Ubuntu platform on a Dell machine (Intel Core2Duo CPU E4400, 3G RAM) equipped with an NVIDIA GeForce 8300GS graphics card.

7.2.1 CT Knee Data Set

We first conducted an experiment on a moderately sized $379 \times 229 \times 305$ voxel CT knee data set (see `http://www9.informatik.uni-erlangen.de/External/vollib/`). The CT knee data set is composed of structures of left and right knees. These mainly include patellas, femurs, tibias and fibulas, as well as the skin. The purpose of this experiment is to render the various bones and the skin at the same time, in order to provide a comprehensive rendering. The primary difficulty to render this data set lies in the similarity of various bones and thus being difficult to differentiate them. The proposed approach was used in the experiment to render the CT knee data set. The experimental result is shown in Figure 7.1. The full hue range and hat-like transfer function shape are used in this figure during the transfer function generation process. Various structures are represented with different branches of the contour tree. As a result, structures are differentiated with various opacities and colors based on the residue flow model and harmonic colors. As shown in Figure 7.1, the various bones and the skin are clearly rendered and differentiated with attractively aesthetic colors. This experiment shows that our approach can effectively generate transfer functions for various structures automatically while optimized to depict inclusion relationships at the same time.

7.2.2 CT Foot Data Set

The CT foot data set (see `http://www.volvis.org/`) with moderately sized $256 \times 256 \times 256$ voxels contains various small bones of toes. It is time consuming for users to generate transfer functions to visualize bones and outer surface layers simultaneously

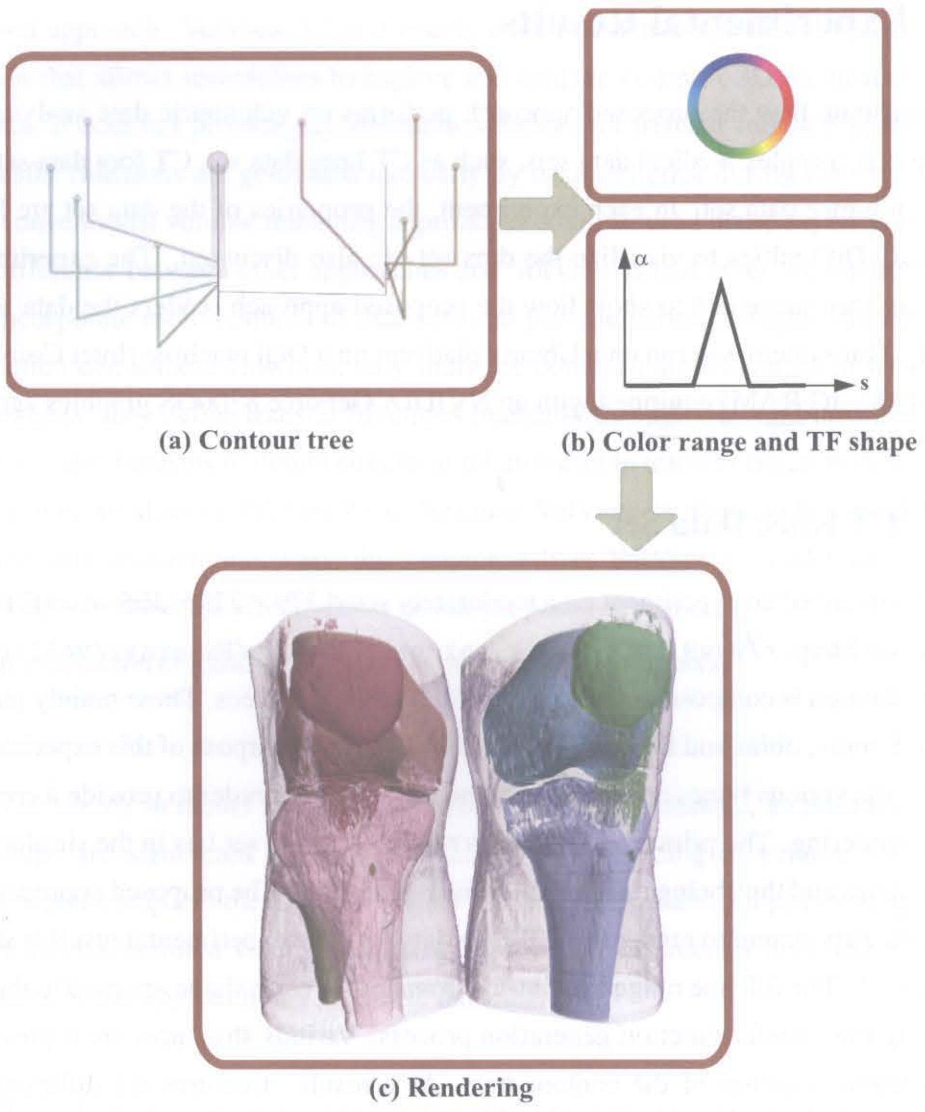


Figure 7.1: Volume rendered knee data set using full hue range and hat-like transfer function shape.

with conventional approaches. The purpose of this experiment is to automatically generate transfer functions for each structure in order to differentiate them visually while displaying their outer surface layers at the same time. The proposed approach was used in the experiment. The experimental result is shown in Figure 7.2. In this figure, the harmonic color hue T-Type and hat-like transfer function shape are used. The inner bone structures and outer surface layers are clearly revealed with different colors. Inner structures are rendered using high opacity, high vividness and low lightness, while outer surfaces are rendered using low opacity, low vividness and high lightness controlled by

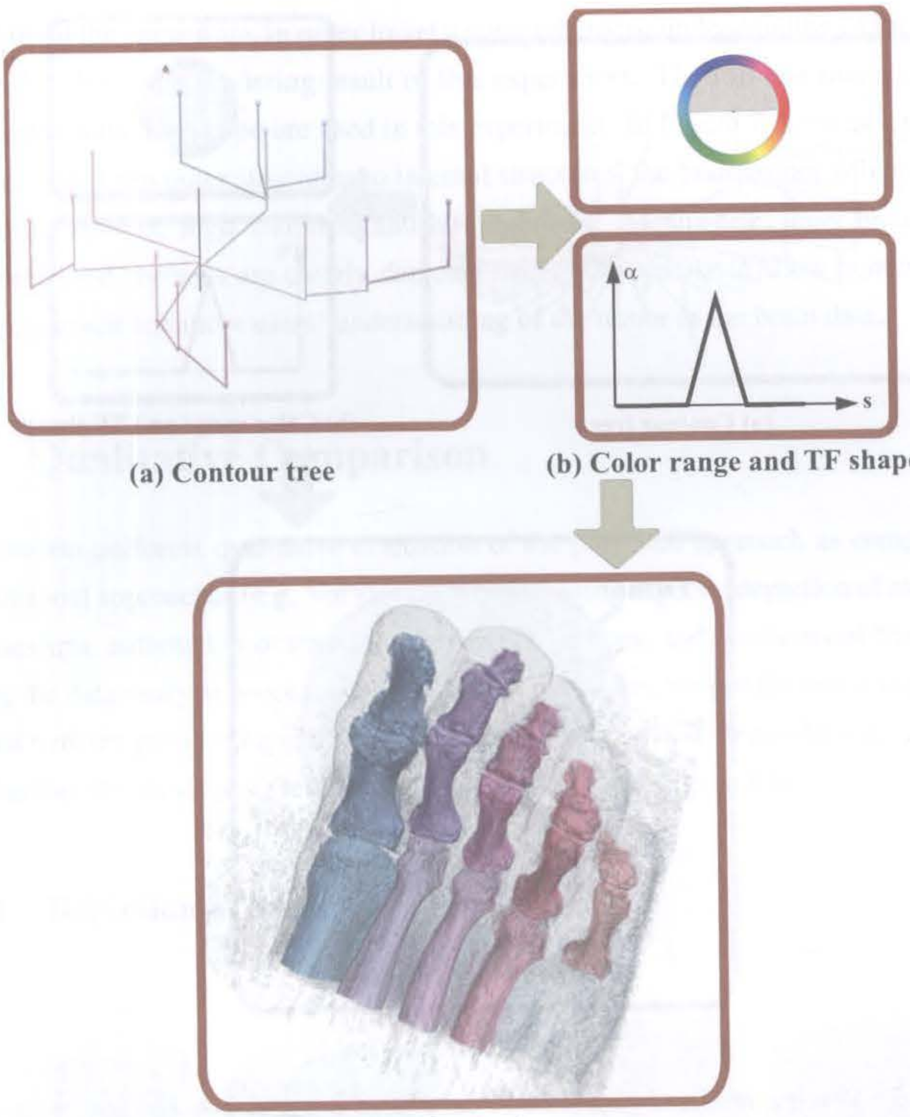


Figure 7.2: Volume rended foot data set using harmonic hue T-Type and hat-like transfer function shape.

the contour tree. The transfer function for each structure is automatically and locally defined, in order to highlight internal structures while revealing outer layers.

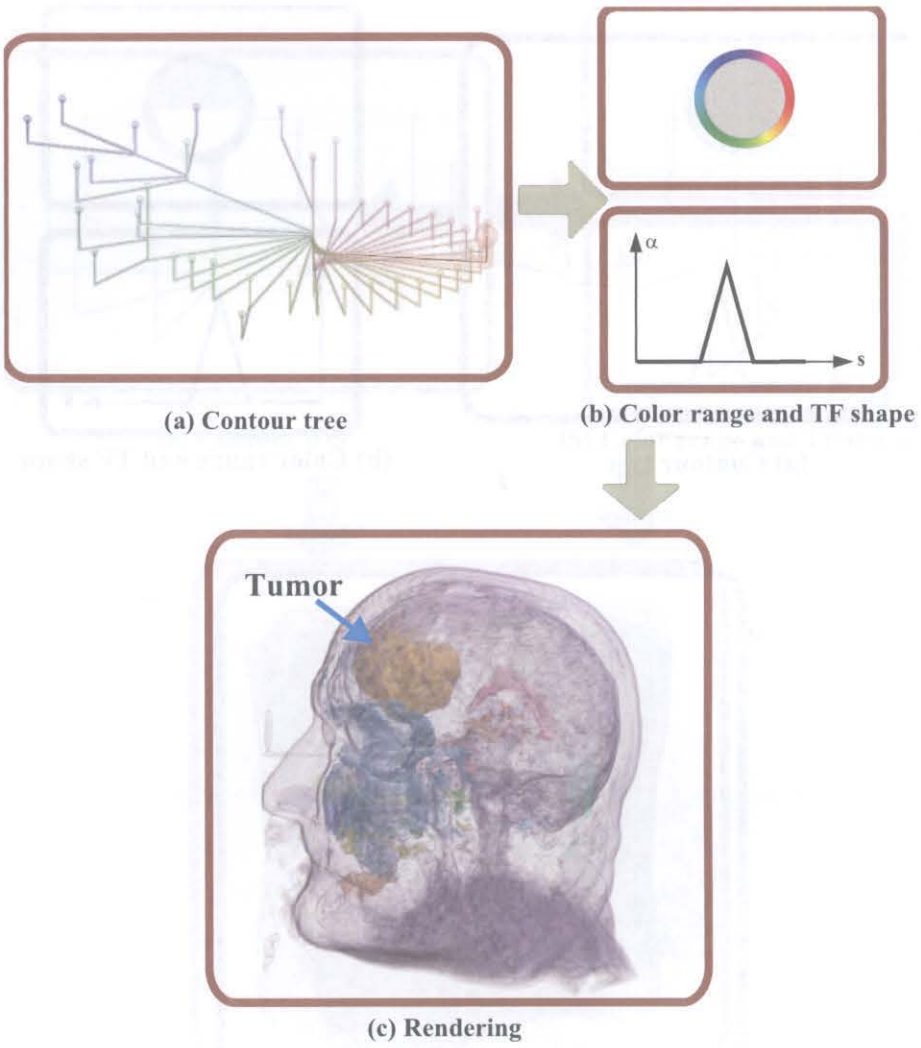


Figure 7.3: Volume rendered tumor head data set using full hue range and hat-like transfer function shape.

7.2.3 MR Brain Tumor Data Set

The proposed approach was also applied to a more complicated data set, an MR head data set with brain tumors inside (data courtesy of B Terwey, Bremen). Because of similar scalar values between tumors and surrounding structures, conventional volume rendering approaches cannot differentiate the brain tumor and other complex brain structures effectively in the MR head data set. As a result, the tumor is often included by other outer complicated brain structures. This severely affects understanding of the tumor. The purpose of this experiment is to visualize the tumor in the brain and other outer structures at the same time, in order to get a comprehensive understanding of the tumor. Figure 7.3 shows the rendering result of this experiment. The full hue range and hat-like transfer function shape are used in this experiment. In Figure 7.3, because opacity residues flow from outer structures to internal structures, the brain tumor is highlighted using high opacity, high vividness and low lightness. Meanwhile, inner brain shape and outer head surfaces are clearly depicted using low opacity to show context. The rendering result enhances users' understanding of the tumor in the brain data.

7.3 Qualitative Comparison

This section performs qualitative evaluation of the proposed approach as compared to conventional approaches (e.g. VolView 3.2) in following aspects: depiction of structural relationships, automation of transfer function generations, and involvement from users during the data analysis process. For comparison purposes, we put the rendering results created with the proposed approach in the previous section and the results with VolView 3.2 together in this section (see Figure 7.4, Figure 7.5 and Figure 7.6).

7.3.1 Depiction of Structural Relationships

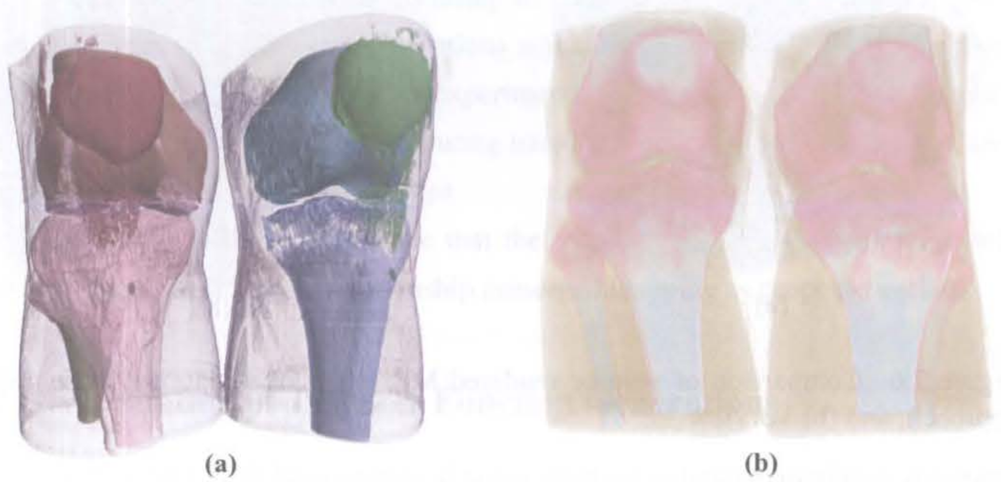


Figure 7.4: Comparison of volume rendered CT knee data set with: (a) our approach, and (b) VolView 3.2.

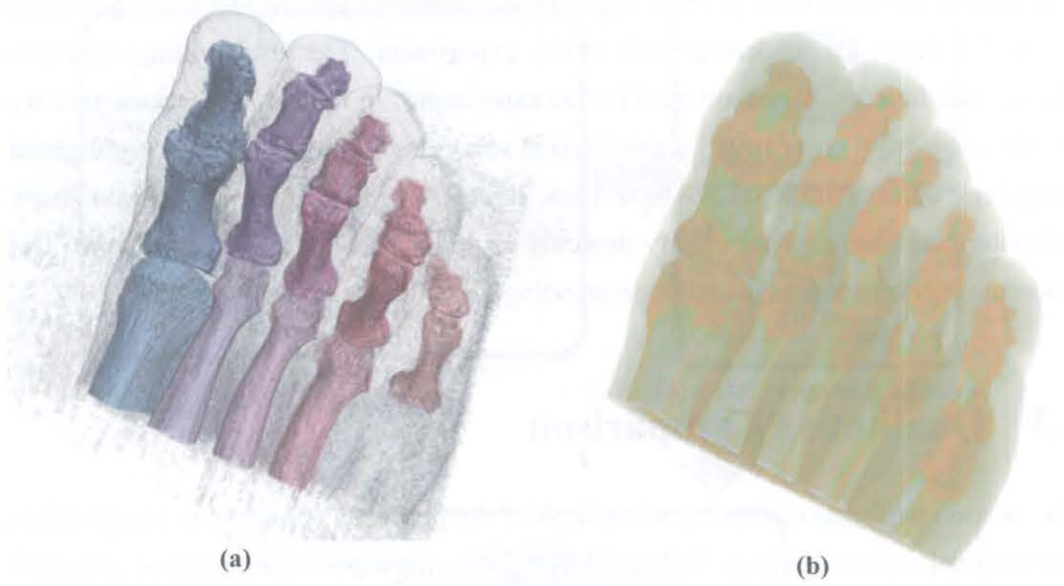


Figure 7.5: Comparison of volume rendered CT foot data set with: (a) our approach, and (b) VolView 3.2.

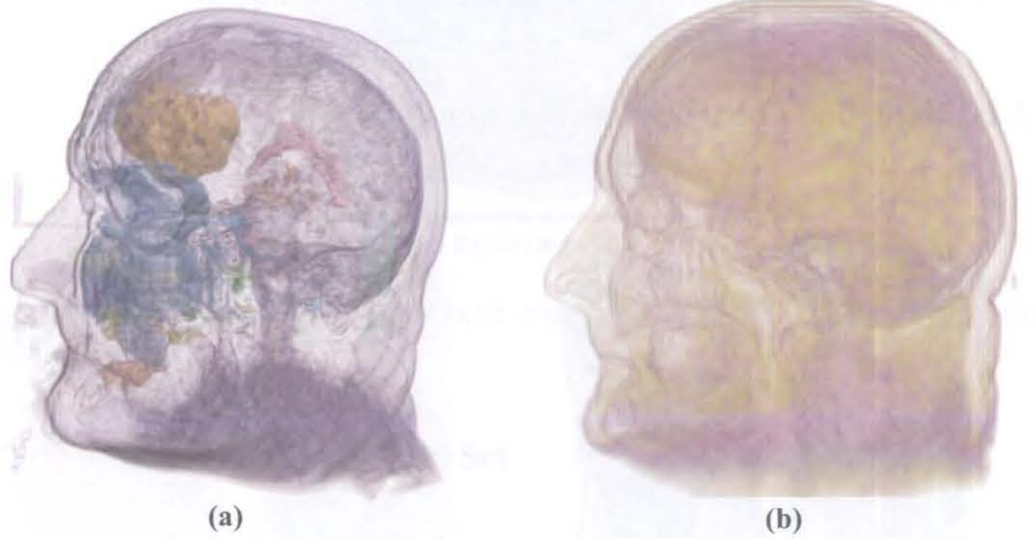


Figure 7.6: Comparison of volume rendered MR tumor head data set with: (a) our approach, and (b) VolView 3.2.

As mentioned in previous chapters, structural relationships play significant roles in understanding volumetric data sets. Depiction of structural relationships provides more intuitive and physically meaningful renderings. It enhances information provided by common rendering techniques. This subsection compares the ability of depicting structural relationships of the proposed approach with the conventional approach as used in VolView 3.2.

As shown in Figure 7.4(a), Figure 7.5(a) and Figure 7.6(a), we see that inner structures (e.g. bones and the tumor) are rendered with higher opacities and different colors, while outer surface layers are rendered with lower opacities. On the contrary, the rendering results as shown in Figure 7.4(b), Figure 7.5(b) and Figure 7.6(b) do not reveal inner structures and outer surface layers clearly. Especially in Figure 7.6(b), the inner tumor is not revealed at all.

Conventional approaches usually differentiate structures based on large contrast with their surrounding structures, i.e. gradient information. This is effective in some degree as shown in Figure 7.4(b) and Figure 7.5(b). However, if structures of interest have similar scalar values with their surroundings, users cannot differentiate them, and thus cannot depict structural relationships as shown in Figure 7.6(b).

From the comparisons, we see that the proposed approach can depict structural relationships (e.g. inclusion relationship) effectively in complicated volumetric medical data sets. The underlying reason is that the proposed approach represents topological relationships of structures explicitly and utilizes them during the transfer function generation process. Conventional approaches do not exploit topological attributes between structures explicitly during transfer function generations. Although some conventional approaches can depict inclusion relationship to some degree as investigated in Section 2.7, they need excessive user interactions which reduce efficiency and come short in repeatable results. Specifically in this experiment, VolView 3.2 does not have mechanisms to depict structural relationships during transfer function generation process, and thus fails to depict structural relationships.

From the comparison, we conclude that the results using the proposed approach support the theory of structural relationship preserved mapping as proposed earlier.

7.3.2 Automation of Transfer Function Generations

One of the problems which limits practical applications of volume rendering in volumetric data analysis is the definition of complex parameter space, especially the definition of transfer functions.

With the approach in this thesis, the transfer function generation process requires users to select a color hue wheel and a transfer function shape. The transfer functions are then generated automatically. A useful "by-product" of our approach is that users can interactively explore structures (especially inner structures) in the data set through interacting with the residue flow rate with a simple slidebar.

Comparatively, when conventional approaches for transfer function generations are used (e.g. the approaches used in VolView 3.2), users have to conduct complex low-level transfer function parameter adjustments by trial-and-error manually. Four parameters of opacity, red, green and blue need to be modulated manually in this process. The help information used in this manual process often includes a histogram of the data set, experiences and domain knowledge (e.g. the scalar value of bones is usually higher than that of soft tissues). Users have to modulate four parameters of opacity, red, green and blue manually by trial-and-error respectively. This process is time-consuming and comes short in repeatable results.

From the comparisons as mentioned above, we concluded that our approach can automate transfer function generations more effectively than conventional approaches. The results support the theory of automatic transfer function generations as presented in the previous chapters.

7.3.3 Involvement From Users

The involvement from users primarily refers to what interactions are used to create comprehensive rendering and how complex the interactions are. This section compares the complexity of interactions and efficiency of transfer function generations between our approach and conventional approaches.

Complexity of Interactions

Our approach only requires few involvements from users to create a comprehensive rendering. These involvements primarily include the selection of a color hue wheel, the selection of a transfer function shape, and possible interactions of the residue flow rate. These involvements are simply performed with several mouse clicks. Users do not need to make complex decisions. They even do not need any visualization expert knowledge to create renderings.

In conventional approaches, users often use various polylines/curves (e.g. the user interface used in VolView 3.2 as shown in Figure 2.7) to define opacity transfer functions. It is quite difficult to explore an ideal curve in a 2D space. To simplify, conventional approaches often use a number of 2D control points to define a polyline to approximate the curve of a transfer function. Every 2D point has 2 degrees of freedom (DOF). With the help of a histogram displayed as the background of the transfer function widget, it still takes much time for users to define an ideal transfer function for the comprehensive rendering by trial-and-error. Even if semantic transfer function approaches as investigated in Section 2.6 are used, users still need complex decisions and interactions in transfer function generations. Moreover, the definition of the color transfer function is also a time consuming task for conventional approaches. There are no specific rules between physical meaning and colors. Users often define colors for various features of data sets randomly and/or based on their preferences. It is more difficult for users to define attractive aesthetic colors for a comprehensive rendering manually. This process requires much visualization expert knowledge to do interactions and decisions.

From the comparisons, we observed that the proposed approach in this thesis simplified the opacity transfer function generation process, from the complicated trial-and-error process for the curve definition to a simple mouse clicking for the selection of a transfer function shape. Also the color transfer function generation process is simplified, from the modulation of three unrelated parameters (red, green and blue) to the selection of a color hue wheel with a simple mouse clicking without complex decisions.

Efficiency of Transfer Function Generations

Because the contour tree is precomputed, the time used to generate transfer functions with our approach include: 1) selection time for the color hue wheel and the transfer function shape, 2) processing time for automatically generating transfer functions. The first item usually takes seconds with several mouse clicks. The second item can be done in real time for small data sets (e.g. the nucleon and fuel data sets as shown in section 6.6). Frame rates for larger data sets are also sufficiently high for automatic transfer function generations. For example, the processing time for generating transfer functions for the CT knee data set is 1.27s (10 branches, and original branch number is 954968). Altogether, our approach allows users to generate transfer functions quickly and easily.

Comparatively, conventional approaches take much more time to generate transfer functions for volume data. This is because that users need to perform complex interactions and decisions for each parameter by trial-and-error as mentioned above. Even for an experienced user, it is also time consuming to get an ideal transfer function for a volume data. For example, we spent more than 5 minutes to create the transfer function by trial-and-error used in Figure 7.6(b). Even if we spent much longer time than the time used in our proposed approach, the transfer function in Figure 7.6(b) is still not good enough to depict structures of interest (e.g. the tumor in the brain). The color transfer function is also not aesthetically attractive as in our approach. This is same for Figure 7.4 and Figure 7.5.

This comparison demonstrated that the proposed approach can efficiently generate transfer functions, and therefore meets the hypothesis and objectives of this thesis as presented in Chapter 1.

7.4 Applications

The direct results of this thesis include important theories for volume rendering and volumetric data analysis, therefore guiding rendering design and enabling new volume rendering software. Specifically, the highly demanding theories on automation of rendering parameter generations will benefit researchers and practitioners in this area. Researchers and practitioners can apply the theories developed in this thesis in visualization software. Firstly, the proposed approach is used to automate transfer function generations, which are optimized for depicting structural relationships such as the inclusion relationship. Besides, it can be used to generate initial transfer functions for a complex data set. If users are not satisfied with the generated transfer functions, the transfer functions generation using the proposed approach can be further optimized to get comprehensive rendering. This significantly reduces the time of transfer function generations as compared with conventional approaches. More specifically, the results of this thesis could potentially benefit healthcare and other industrial areas, for example, non-destructive testing and computational fluid dynamics.

- The direct application of this research is to visualize and analyze volumetric medical images (e.g. CT, MRI) which are heavily used in hospitals. Radiologists could potentionally benefit from this work to improve the efficiency of routine medical image analysis.

- Non-destructive testing (NDT): NDT is a wide group of analysis techniques used in science and industry to evaluate the properties of a material, component or system without causing damage. It is a highly valuable technique that can save both money and time in product evaluation, troubleshooting and research. Industrial CT is one of widely used NDT techniques. The results of this thesis can be used in NDT to improve the efficiency of data analysis.

- Computational fluid dynamics (CFD): In computational fluid dynamics, engineers use numerical calculations to accurately simulate many engineering problems that once required the use of physical experiments involving wind and water tunnels. CFD has come to serve as an instrument in the design of many familiar engineering processes. Volume rendering is a vital part of analyzing CFD simulation results. The novel theories developed in this thesis promote a wider use of volume rendering in CFD analysis.

7.5 Limitations

The effectiveness of our approach depends on the quality of the contour tree, which is decided by signal-to-noise ratio (SNR) and contrast of volumetric data. Data sets with higher SNR and higher contrast create higher quality contour trees, which can more accurately represent topology of data sets. Fortunately, with the improvement of scanning techniques, scientists may get volumetric data with higher quality, and thus our method becomes more effective in volumetric data analysis. Currently, we may improve the quality of the contour tree through preprocessing of volumetric data (e.g. filtering). More algorithms of the contour tree can also be developed to create high quality contour trees.

The proposed approach requires excessive memory to store contour trees in the pipeline. The required size of memory depends on the size of contour trees. As mentioned above, the SNR and contrast of volumetric data affect the contour tree quality, and thus also the size of contour trees. However, with the advancement of hardware nowadays, memory is becoming cheaper. This problem is not highly critical and a desktop computer with more than 10GB memory is not an unusal configuration nowadays.

Typical preset transfer function shapes are usually available for users in most widely used visualization software, in order to provide quick and easy exploration of data sets. These transfer function shapes are often used to set global transfer functions of volumetric data. In the proposed approach, users need to select the transfer function shape manually for a local transfer function for each branch. It is often based on the data properties, for example, the hat-like shape is more effective to reveal iso-surfaces in complex regions, the linear shape is effective for homogenous regions. The manual selection of the transfer function shape affects the overall effectiveness of the proposed approach. So automation of selection of transfer function shape would further improve the efficiency of automatic transfer function generations.

7.6 Summary

In this chapter, we described experiments conducted with various data sets to demonstrate the effectiveness of the proposed approach in this thesis. Qualitative comparisons were performed in three aspects to show advantages of the proposed approach. Limitations of the proposed approach were also discussed for the future improvement. From the comparison, we concluded that with the theories proposed in this thesis, volume rendering researchers and practitioners are able to visualize and analyze volumetric data simply with the control of a few easily understood parameters. It only requires practitioners to learn easily understood concepts such as residue flow rate, instead of complex visualization expert knowledge as required by conventional methods. Because the proposed approach aims to automate transfer functions and does not require much involvement from users, it is even effective for an unknown volumetric data set. The novel approach greatly improves the efficiency of transfer function generations, and it is significantly beneficial to users who are not visualization expert. So it greatly improves practical applications of volume rendering in volumetric data analysis, especially in medical image analysis. As a result, the proposed approach significantly improves the efficiency of volumetric data analysis.

Chapter 8

Conclusions

Volume rendering is becoming an essential tool for volumetric data analysis. Its ultimate purpose is comprehension. As scientific data continue to increase in size and complexity, topology based techniques have begun to emerge as a general framework to capture significant features of the data at an abstract level, enabling and facilitating data understanding in visualization. The contour tree is one of the topological abstractions of data sets. It has been used to define transfer functions and other rendering parameters in volumetric data analysis.

In the context of this work, we focused on investigating how the contour tree could be used to improve the efficiency of volumetric data analysis. This thesis approached the issue by utilizing the topology of data sets to explore various relationships of structures in a data set in volume rendering. It also utilized the topology to automate rendering parameter generation in volume rendering. To make this work practical, this thesis dealt with topology simplifications.

In this chapter, a summary of contributions of the thesis is presented. Ideas for future research directions are also discussed.

8.1 Summarized Contributions

This thesis was concerned with analyzing effective uses of the contour tree in volume rendering. The thesis contributed to extract data information and understand data sets through depicting topological relationships and automating analysis of volumetric data. In particular, the contributions of this thesis are outlined as follows:

Multiple Measures of Importance for Contour Tree Simplification

We proposed a *multiple measures of importance approach for the contour tree simplification*. The proposed approach used multiple measures of importance simultaneously in the contour tree simplification process by introducing concepts of attribute space, importance triangle and importance space. It maximized advantages of each measure of importance in the contour tree simplification process. Interfaces were provided to allow users to interact with branches more meaningfully and efficiently. The importance-driven approach allows for the simplification of contour trees by considering the overall attribute space and giving better evaluation of importance of a branch, instead of the limited attribute space used by conventional approaches. The priority line window allows the specification of an importance threshold more efficiently and meaningfully, instead of unguided low-level threshold adjustment of single measure of importance [25]. The proposed approach can be generalized to process branches with more than three measures.

Structural Relationship Preserved Mapping

We presented an approach on depicting structural relationships between objects in volume rendering through *a concept of structural relationship preserved mapping*. The concept was set up through analyzing the roles of perception in volume visualization. Two typical relationships of inclusion and neighboring were defined and depicted in volume rendering respectively. The contour tree controlled structural relationship depiction allows users to perceive structural relationship in a more direct way. The advantage of the structural relationship preservation approach is that it allows analysis of volumetric data to focus on revealing high-level topological relations instead of low-level rendering parameter modulations, and thus improves understanding of volumetric data. The structural relationship preservation provides information which is not available in commonly used volume rendering techniques, and enhances information provided by common rendering techniques.

Automatic Transfer Function Generations

This thesis presented a novel approach for automating transfer function generations. The new approach utilized topological attributes derived from the contour tree to automate transfer function generations. In the new approach, a *residue flow model* based on *Darcy's Law* was employed to control distributions of opacity between branches in the contour tree. Topological attributes were also used to control color selection in a perceptual color space and create *harmonic color transfer functions*. The transfer functions reveal structural relationships (e.g. inclusion relationship) automatically. They were optimized to maximize opacity and color differences between structures.

Some of the theories and results presented in this thesis have been published in IEEE TVCG [129], other journals [130], proceedings [132] and book chapters [131]. As well, some results have been published as a technical report [128] or are ready to be submitted [133].

8.2 Future Research Directions

Within this thesis, we have shown the successful roles of topology, especially the contour tree, in improving the effectiveness of volume rendering in volumetric data analysis. However, there are potentials for further research and development based on this research. This section presents a number of interesting areas for further development.

Structural Relationship Depiction

The concept of the structural relationship preserved mapping proposed in this thesis opens opportunities for possible research areas. The open issues include: 1) Definition of new structural relationships based on topology and other various features of volumetric data. Various features of a 3D data set can be used to define new structural relationships. For example, two objects may belong to the same category in a data set (e.g. two tumors or toe bones). We may name this as member relationship. Various features of the data set need to be detected to define new structural relationships. 2)New methods of representation of structural relationship. With the new structural relationships, we need to develop novel methods to reveal them in volume rendering based on topology.

Automation of Selection of Local Transfer Function Shapes

In the proposed approach, users are required to select a local transfer function shape manually during the transfer function generation process. The manual selection of the transfer function shape may affect the overall effectiveness of the proposed approach. So automation of selection of transfer function shapes would further improve the efficiency of automatic transfer function generations. Various topological attributes and other data features may be used in this process.

Different transfer function shapes may be applied to different branches in the contour tree in order to further improve the effectiveness of transfer function generations for volumetric data. Furthermore, more choices of local transfer functions for each branch can be developed. For example, multi-dimensional transfer functions and other approaches reviewed in Section 2.6 can be adapted for each branch to automate and improve the overall effectiveness of transfer function generations. An extensive evaluation study on the proposed approaches would also be one of our future directions.

Automation of Other Rendering Parameters

Volume rendering has a complex parameter space that limits its practical applications in data analysis. The parameter space refers to a set of parameters used to produce a rendering image. The parameters that are available in visualization software can be divided into two major categories [114]: 1) View specific parameters: view position, orientation, zoom, light position (point light), light direction (distant light), shading coefficients (ambient, diffuse, specular); 2) Data specific parameters: color transfer function (mapping between data value and color), opacity transfer function (mapping between data value and transparency level), slicing plane position and orientation.

As mentioned above, besides transfer functions, lighting and shading are critical parameters which affect the rendering quality especially homogeneous regions in volume rendering. These settings are often defined randomly or based on experiences. In addition, after getting a rendering display, users often perform interactions, such as rotate and zoom in/zoom out, to modulate camera positions in order to perceive as much information as possible. This process lacks theoretical guidance and is time consuming. Topology of data sets is one of the factors that affect these parameter settings. For example, camera information needs to be different for a simple object and a complex data set with multiple objects inside, in order to improve the efficiency of data understanding.

One of the future research directions may aim to utilize topology of data sets to automate lighting and camera parameter generations, in order to maximize the display of structural information of objects while minimizing user interactions. The challenges include: what are the measurements that are used to evaluate the displaying of structural information in the rendering display window based on topology, as well as how to set up and minimize the cost function that is used to control these measurements.

Volume Rendering of the Future

The ultimate ideal of volume rendering would be to allow users to analyze volumetric data without or with much fewer involvements and decisions. In addition, because end users are usually not visualization experts, the involvements and decisions should not require much visualization expert knowledge from users. Volume rendering ideally would provide visualization for tasks that require human judgment, and other tasks would be automated where possible. But finding a productive balance between automation and visualization is a challenge and is one of the goals of visual analytic methods [80]. The automation of rendering parameter generations is a critical part of this ideal. The work in this thesis is an important step toward to minimizing the gap to the ideal.

Appendix A

Priority Line Window

Given persistence, volume, hypervolume and *ITri* of a set of branches, we need an intuitive way to present and manipulate them. This appendix introduces an interface named priority line window to facilitate the display, and manipulate their corresponding importance meaningfully.

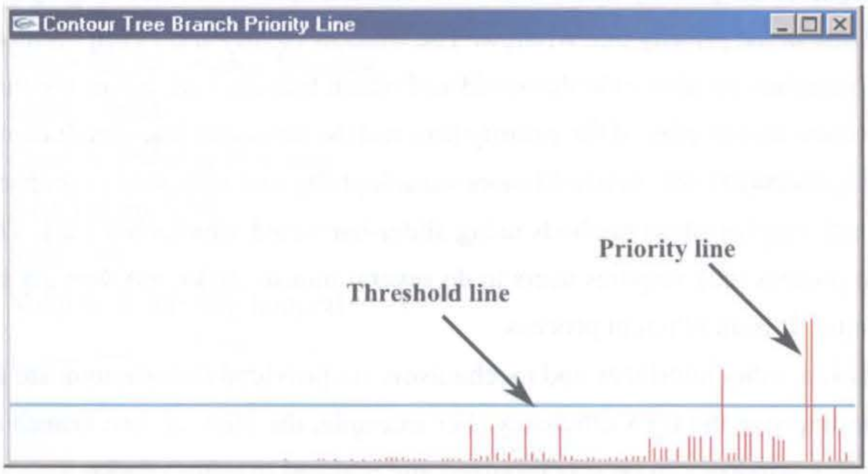


Figure A.1: The priority line and threshold line.

The global description of priority values of a given measure of importance allows users to perceive and manipulate branches in a different way. Similar to the histogram of an image, the priority values of branches of a data set can also be represented as a group of lines and used in the CTS. The priority line is set up as follows: the horizontal axis represents different branches, and the vertical axis represents importance values. The index number of branches is based on the data structure used in our implementation and kept constant during the simplification. Each branch is represented with a vertical line, and its height is the corresponding importance value. We also call importance values priority values because they are used as priority values of priority queues in the CTS pipeline.

The priority line window compares the different priority values of branches of a given measure, allowing users to find patterns of the priority values (e.g. peaks of values). The order of branches represented in priority lines does not affect users' comparison as long as lines of various measures of importance use the same order. The patterns help users to manipulate different measures of importance more meaningfully. For example, from the priority line window, we see that there are several obvious peaks in the priority lines of volume and hypervolume. These peaks may correspond to regions of interest in the data set. During the CTS, a priority threshold is often necessary to control the simplification process. Similar to using a histogram to set meaningful thresholds in image understanding, the priority line window can help users to set the threshold based on patterns of priority lines in order to get an effective CTS. Figure A.1 is an example of the priority line window. In this example, users can directly move the threshold line in the priority line window. The window clearly shows importance values of which branches are above the threshold and which branches are below the threshold. The interaction and display of the priority line and the threshold line simultaneously allow users to modulate the threshold more meaningfully and effectively compared with conventional simplification methods using slider-bar based approaches [25]. This simplification process only requires users to do several mouse clicks and then get simplification results. It is an efficient process.

In addition, other interfaces and mechanisms are provided to communicate and collaborate to improve the CTS efficiency. For example, the click on one branch or node in any display window can be synchronized and updated in other windows.

Abbreviations

CT	Computerized Tomography
CTF	Color Transfer Function
CTr	Contour Tree
CTS	Contour Tree Simplification
DVR	Direct Volume Rendering
HLS	Hue, Lightness, Saturation
HSV	Hue, Saturation, Value
ITri	Importance Triangle
JT	Join Tree
MRI	Magnetic Resonance Imaging
MS	Morse-Smale
OTF	Opacity Transfer Function
RGB	Red, Green, Blue
ST	Split Tree
VRI	Volume Rendering Integral

Bibliography

[1] Siam: Computational science and engineering. http://www.siam.org/books/series/csecover.php, 15 April 2010. (document), 1.2

[2] Kitware Inc. http://www.kitware.com/products/volview.html, 2 November 2009. (document), 2.5, 2.7, 7.1

[3] Merriam-webster online. http://www.merriam-webster.com/dictionary/importance, 23 March 2010. 4.4.1

[4] Color space – wikipedia, the freee encyclopedia. http://en.wikipedia.org/wiki/Color_space, 25 November 2009. (document), 6.5, 6.6

[5] Linocolor.com. http://www.linocolor.com/, 25 November 2009. (document), 6.7

[6] VisWeek 2009. http://vis.computer.org/VisWeek2009/vis/, 26 October 2009. 1.1.1

[7] Kitware inc. http://kitware.com/products/vvimagegallery.php, 29 March 2010. (document), 1.1

[8] C. A. Aumann and E. D. Ford. Modeling tree water flow as an unsaturated flow through a porous medium. *Journal of Theoretical Biology*, 219(4):415–429, 2002. 6.3.1

[9] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *IEEE Visualization '97*, pages 167–173, 1997. 2.6.1, 3.6, 3.6.1

[10] C. L. Bajaj, V. Pascucci, and D. Schikore. Visualization of scalar topology for structural enhancement. In *Proceedings of IEEE Visualization '98*, pages 51–58, 1998. 3.6.1

[11] J. Bertin. *Graphics and graphic information processing.* Walter de Gruter, Berlin, 1981. 5.1

[12] D. Borland, J. P. Clarke, J. R. Fielding, and R. M. Taylor II. Volumetric depth peeling for medical image display. In R. F. Erbacher, J. C. Roberts, M. T. Gröhn, and K. Börner, editors, *Proceedings of SPIE Visualization and Data Analysis*, volume 6060, San Jose, USA, 2006. 1.2.2, 3.6.3

[13] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proceedings of IEEE 1963 Fall Joint Computer Conf.*, pages 445–458, 1963. 3.3.2, 3.6.3

[14] P.-T. Bremer, H. Carr, A. Gyulassy, and G. H. Weber. Scalar topology in visual data analysis. Tutorial in IEEE VisWeek 2009, October 2009. 1.1.1

[15] Peer-Timo Bremer, Herbert Edelsbrunner, Bernd Hamann, and Valerio Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004. 3.2.1, 3.2.1

[16] C. A. Brewer. Color use guidelines for mapping and visualization. In A. M. MacEachren and D. R. F. Taylor, editors, *Visualization in Modern Cartography*, chapter 7, pages 123–147. Elsevier Science, Tarrytown, NY, 1994. 2.6.7

[17] C. A. Brewer. Guidelines for use of the perceptual dimensions of color for mapping and visualization. In J. Bares, editor, *Proceedings of SPIE Color Hard Copy and Graphic Arts III*, volume 2171, pages 54–63, Bellingham, WA, USA, May 1994. 2.6.7

[18] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006. 2.7

[19] S. Bruckner and M. E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344–1351, 2007. 1.2.2, 2.6.3, 3.6.3

[20] J. J. Caban and P. Rheingans. Texture-based transfer functions for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1364–1371, 2008. 2.6.6

[21] Loren Carpenter. The A-buffer, an antialiased hidden surface method. *Computer Graphics*, 18-3:103–108, 1984. 2.7

[22] H. Carr. *Topological Manipulation of Isosurfaces*. PhD thesis, University of British Columbia, Canada, 2004. 1.1.2, 1.2.1, 3.1.1, 3.1.2, 3.6.3

[23] H. Carr and J. Snoeyink. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *Proceedings of the Symposium on Data visualisation 2003*, pages 49–58, 2003. 2.7, 3.3.2

[24] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003. (document), 1.2.1, 3.1.3, 3.3.1, 3.4, 3.3.1, 3.3.2, 3.3.4, 3.3.5, 3.5, 3.6, 3.6.2, 3.6.4

[25] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proceedings of IEEE conference on Visualization '04*, pages 497–504, 2004. (document), 1.1.4, 1.2.1, 1.6, 3.3.5, 3.4, 3.5, 3.6.2, 3.6.4, 4.1, 4.3.2, 4.4.2, 8.1, A

[26] H. Carr, J. Snoeyink, and M. van de Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry*, 43(1):42–58, January 2010. 3.4, 4.1, 4.6

[27] M.-Y. Chan, H. Qu, K.-K. Chung, W.-H. Mak, and Y. Wu. Relation-aware volume exploration pipeline. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1683–1690, Nov.-Dec. 2008. (document), 1.1.3, 1.5, 1.2.2, 2.7, 2.8, 2.7, 5.1, 5.5

[28] W. Chen, Y. Q. Shi, and G. Xuan. Identifying computer graphics using hsv color model and statistical moments of characteristic functions. In *IEEE International Conference on Multimedia and Expo 2007*, pages 1123–1126, 2007. 6.5.2

[29] Y.-J. Chiang and X. Lu. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. *Computer Graphics Forum*, 22(3):493–504, 2003. 3.6

[30] D. Cohen-Or, Y. Chrysanthou, C. Silva, and G. Drettakis. Visibility, problems, techniques and applications. In *ACM SIGGRAPH'00 Course Notes*, Louisiana, USA, 2000. ACM. 3.6.3

[31] D. Cohen-Or, O. Sorkine, T. Leyvand R. Gal, and Y.-Q. Xu. Color harmonization. *ACM Transactions on Graphics*, 25(3):624–630, 2006. (document), 2.6.6, 5.3.2, 5.5, 5.3.2

[32] C. Collins and S. Carpendale. VisLink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007. 2.7

[33] R. Cook, N. Max, C. T. Silva, and P. L. Williams. Image-space visibility ordering for cell projection volume rendering of unstructured data. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):695–707, November/December 2004. 2.7

[34] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380–1387, 2008. 2.6.6

[35] C. Correa and K.-L. Ma. The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1465–1472, 2009. 2.6.2

[36] M. Dastani. The role of visual perception in data visualization. *Journal of Visual Languages and Computing*, 13:601–622, 2002. 2.7, 5.2

[37] H. Doraiswamy and V. Natarajan. Efficient algorithms for computing reeb graphs. *Computational Geometry: Theory and Applications*, 42(6-7):606–616, 2009. 3.2.2

[38] H. Edelsbrunner, J. Harer, A. Mascarenhas, and V. Pascucci. Time-varying reeb graphs for continuous space-time data. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 366–372, New York, NY, USA, 2004. ACM. 3.3.2

[39] Niklas Elmqvist and Philippas Tsigas. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095–1109, 2008. 2.7

[40] K. Engel, , M. Hadwiger, J. Kniss, and C. Rezk-Salama. Tutorial on real-time volume graphics. In *Eurographics 2006*, 2006. (document), 2.3, 2.2, 2.4, 2.5, 2.4

[41] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama, and D. Weiskopf. Real-time volume graphics. Course Notes 28 In SIGGRAPH 2004, 2004. 2.3, 2.4, 2.5

[42] S. Fang, T. Biddlecome, and M. Tuceryan. Image-based transfer function design for data exploration in volume visualization. In *IEEE Visualization '98*, pages 319–326, 1998. 2.6.1

[43] M. Fitting. Pythagoras' theorem for areas — revisited. Technical report, Department of Mathematics and Computer Science, City University of New York, 2001. 4.5.2

[44] H. Freeman and S. Morse. On searching a contour map for a given terrain elevation profile. *Journal of the Franklin Institute*, 284(1):1–25, 1967. 3.3.2

[45] I. Fujishiro, T. Azuma, and Y. Takeshima. Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis (case study). In *Proceedings of IEEE Visualization Conference '99*, pages 467–470, 1999. 3.2.2

[46] I. Fujishiro, Y. Takeshima, T. Azuma, and S. Takahashi. Volume data mining using 3d field topology analysis. *IEEE Computer Graphics Applications*, 20(5):46–51, 2000. 2.6.5

[47] M. E. M. Gherardelli. *Hardware Accelerated Volume Visualization on PC Clusters*. PhD thesis, Universität Stuttgart, 2004. 2.1, 2.5

[48] A. Gyulassy, M. A. Duchaineau, V. Natarajan, V. Pascucci, E. M. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1432–1439, 2007. (document), 3.2.1, 4.2, 4.1, 4.2, 4.4.2

[49] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. A topological approach to simplification of three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):474–484, 2006. (document), 3.2.1, 3.2.1, 3.1, 4.4.2

[50] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, 2007. 3.2.1

[51] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Importance-driven visualization layouts for large time series data. In *Proceedings of IEEE Symposium on Information Visualization 2005*, pages 203–210, Minneapolis, MN, USA, October 2005. 2.7

[52] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM. 3.2.1

[53] R. Huang and K.-L. Ma. Rgvis: Region growing based techniques for volume visualization. In *Proceedings of Pacific Graphics Conference 2003*, pages 355–363, Washington, DC, USA, 2003. 2.6.2

[54] S. Takahashi I. Fujishiro and Y. Takeshima. Collaborative visualization: Topological approaches to parameter tweaking for informative volume rendering. In *Proceedings of Systems Modeling and Simulation: Theory and Applications, Asia Simulation Conference 2006*, pages 1–5, Tokyo, Japan, October 2006. (document), 1.7

[55] T. Itoh and K. Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):319–327, 1995. 3.3.4

[56] T. J. Jankun-Kelly, Kwan Liu Ma, and Michael Gertz. A model for the visualization exploration process. In *Proceedings of the conference on Visualization '02*, pages 323–330, 2002. 5.2

[57] M. Kasenow. *Determination of Hydraulic Conductivity from Grain Size Analysis.* Water Resources Publication, 2002. 6.3.1

[58] A. Kaufman, K. H. Höhne, W. Krügrer, and P. Schröder. Research issues in volume visualization. *Computer Graphics and Applications*, 14(2):63–67, March 1994. 2.1

[59] L. Kettner, J. Rossignac, and J. Snoeyink. The safari interface for visualizing time-dependent volume data using iso-surfaces and contour spectra. *Computational Geometry*, 25(1-2):97–116, 2003. 3.6

[60] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization*, pages 79–86, 1998. 1.2.3, 2.6.2, 6.1

[61] G. L. Kindlmann, R. T. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization*, pages 513–520, 2003. 1.2.3, 2.6.2, 6.1

[62] J. T. Klosowski and C. T. Silva. The prioritized-layered projection algorithm for visible set estimation. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):108–123, 2000. 3.6.3

[63] J. Kniss, W. Hunt, K. Potter, and P. Sen. Istar: A raster representation for scalable image and volume data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1424–1431, 2007. 2.7

[64] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002. 2.6, 2.6.2, 3.6.4, 5.3.2, 6.1, 6.5.2

[65] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In Thomas Ertl, Ken Joy, and Amitabh Varshney, editors, *Proceedings of IEEE Visualization 2001*, pages 255–262, 2001. 2.6.2, 5.3.2, 6.1

[66] A. H. König and E. M. Gröller. Mastering transfer function specification by using volumepro technology. In *Proceedings of the 17th Spring Conference on Computer Graphics (SCCG)*, 2001. 2.6, 2.6.1

[67] N. Krešić. *Hydrogeology and Groundwater Modeling*. CRC Press, 2007. 6.3.1

[68] R. D. Kriz, D. Farkas, A. A. Ray, and J. T. Kelso. Visualization of structure-property relationships: Spanning the length scales from nano to macro. In *Proceedings of the International Conference on Computational and Experimental Engineering Science*, Corfu, Greece, July 2003. Invited Paper. 2.7

[69] J. Kruger, J. Schneider, and R. Westermann. Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):941–948, 2006. 1.2.2, 2.7, 3.6.3

[70] T. Lawson. *Topology: A Geometric Approach*. Oxford University Press, 2003. 3.1.2

[71] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(5):29–37, May 1988. 2.4, 2.6.1, 2.6.2, 3.6.4

[72] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions Computer Graphics and Applications*, 9(3):245–261, 1990. 2.4

[73] E. B. Lum and K.-L. Ma. Lighting transfer functions using gradient aligned sampling. In *Proceedings of IEEE Visualization '04*, pages 289–296, Austin, Texas, USA, October 2004. 2.6.2

[74] C. Lundström. *Efficient Medical Volume Visualization — An Approach Based on Domain Knowledge*. PhD thesis, Linköping Uiversity, Sweden, 2007. 2.5

[75] R. Maciejewski, I. Woo, W. Chen, and D. Ebert. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1473–1480, 2009. 2.6.2

[76] J. Marks and et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *ACM SIGGRAPH Conference on Computer Graphics '97*, pages 389–400, 1997. 2.6.1

[77] Y. Matsumoto. *An Introduction to Morse Theory*, volume 208. The American Mathematical Society, Providence, Rhode Island, USA, 2002. 3.2.1

[78] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995. 2.3

[79] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *IEEE/ACM Symposium on Volume Visualization*, Salt Lake City, Utah, October 2000. 2.4

[80] S. I. O'Donoghue, A.-C. Gavin, N. Gehlenborg, D. S. Goodsell, J.-K. Hériché, C. B. Nielsen, C. North, A. J. Olson, J. B. Procter, D. W. Shattuck, T. Walter, and B. Wong. Visualizing biological data – now and in the future. *Nature Methods Supplement*, 7(3):2–4, 2010. 8.2

[81] V. Pascucci. On the topology of the level sets of a scalar field. In *Abstracts of the 13th Canadian Conference on Computational Geometry*, pages 141–144, 2001. 3.3.2

[82] V. Pascucci and K. Cole-McLaughlin. Efficient computation of the topology of level sets. In *Proceedings of IEEE Visualization 2002*, pages 187–194, 2002. 3.3.1, 3.3.2, 3.3.4, 3.3.4, 3.3.4, 3.3.4, 3.3.4, 3.3.4, 3.3.5

[83] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(1):249–268, 2003. 3.3.2, 3.3.3, 3.3.4

[84] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proceedings of the IASTED conference on Visualization, Imaging, and Image Processing*, pages 452–290, 2004. 1.1.4, 1.2.1, 3.2.1, 3.3.2, 3.3.5, 3.3.5, 3.3.5, 3.4, 3.6.2, 4.1, 4.3.1, 4.3.2, 4.6

[85] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Transactions on Graphics*, 26(3):58, 2007. (document), 3.2.2, 3.2, 3.4

[86] G. Patanè, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the reeb graph. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):583–595, 2009. 3.2.3

[87] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Avila, K. Raghu, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21:16–22, May-June 2001. 2.6

[88] J.-P. Quadrat, J. B. Lassere, and J.-B. Hiriart-Urruty. Pythagoras' theorem for areas. *American Mathematical Monthly*, 108(6):549–551, 2001. 4.5.2, 4.5.2

[89] P. Rautek, S. Bruckner, and M. E. Gröller. Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1336–1343, 2007. 2.6.4, 6.6

[90] G. Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique. *Comptes Rendus Acad, Science Paris*, 222:847–849, 1946. 3.2.2

[91] C. Rezk-Salama, M. Hadwiger, T. Ropinski, and P. Ljung. Advanced illumination techniques for gpu volume raycasting. In *ACM SIGGRAPH Courses Program*. ACM, 2009. (document), 2.3, 2.4

[92] C. Rezk-Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, 2006. 2.6.4, 6.6

[93] P. Sabella. A rendering algorithm for visualizing 3d scalar fields. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH'88)*, pages 51–58, 1988. 6.5.2

[94] G. Schaufler, J. Dorsey, X. Decoret, and F. X. Sillion. Conservative volumetric visibility with occluder fusion. In *SIGGRAPH'00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 229–238, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 3.6.3

[95] M. A. Selver and C. Güzelis. Semi-automatic transfer function initialization for abdominal visualization using self generating hierarchical radial basis function networks. *IEEE Transactions on Visualization and Computer Graphics*, 15(3), 2009. 2.6.6

[96] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 1991. 2.7, 3.2.2

[97] Yoshihisa Shinagawa and Tosiyasu L. Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, 11(6):44–51, November 1991. 3.2.2

[98] S. Smale. On gradient dynamical systems. *Ann. of Math.*, 74:199–206, 1961. 3.2.1

[99] A. R. Smith. Color gamut transform pairs. In *SIGGRAPH'78: Proceedings of the 5th annual conference on computer graphics and interactive techniques*, pages 12–19, 1978. 6.5.1

[100] B.-S. Sohn and C. Bajaj. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2006. 3.3.2

[101] H. Sowizral. Scene graphs in the new millennium. *IEEE Computer Graphics Applications*, 20(1):56–57, 2000. 2.7

[102] S. Takahashi, I. Fujishiro, and M. Okada. Applying manifold learning to plotting approximate contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1185–1192, 2009. 3.3.2

[103] S. Takahashi, I. Fujishiro, and Y. Takeshima. Interval volume decomposer: A topological approach to volume traversal. In *Proceedings of SPIE – Visualization and Data Analysis 2005*, 2005. 3.5, 5.3.1, 5.3.1

[104] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14(3):181–192, 1995. 3.3.4

[105] S. Takahashi, G. M. Nielson, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization using adaptive tetrahedralization. In *Proceedings of the Geometric Modeling and Processing 2004*, pages 227–236, 2004. 3.3.2, 3.4, 4.3.2

[106] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004. 1.1.2, 1.1.4, 2.6.5, 2.6.7, 3.3.2, 3.4, 3.6, 3.6.4, 4.3.2, 6.1

[107] S. Takahashi, Y. Takeshima, I. Fujishiro, and G. M. Nielson. *Scientific Visualization: The Visual Extraction of Knowledge from Data*, chapter Emphasizing Isosurface Embeddings in Direct Volume Rendering, pages 185–206. Springer-Verlag, 2005. (document), 1.1.2, 1.2.2, 1.2.3, 2.6.5, 2.6.7, 2.7, 3.3.2, 3.3.3, 3.6, 3.6.3, 3.8, 3.6.3, 3.6.4, 5.2.2, 5.3.1, 5.3.1, 6.1

[108] Y. Takeshima, S. Takahashi, I. Fujishiro, and G. M. Nielson. Introducing topological attributes for objective-based visualization of simulated datasets. In *Volume Graphics*, pages 137–145, 2005. (document), 1.2.1, 1.2.2, 1.2.3, 1.7, 2.6.5, 2.6.7, 2.7, 3.3.2, 3.6.4, 6.1

[109] S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3d in o(n log n) steps. In *Proceedings of the fourteenth ACM symposium on Computational geometry*, pages 68–75, 1998. 3.3.2

[110] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975. 3.3.4

[111] T. Theußl. Issues on displaying 3d data for scientific visualization. In *Proceedings of CESCG'98*, 1998. 6.4.2

[112] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1177–1184, 2009. 3.2.3

[113] Christian Tietjen, Tobias Isenberg, and Bernhard Preim. Combining silhouettes, surface, and volume rendering for surgery education and planning. In *IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 303–310, 2005. 6.4.2

[114] M. Tory, S. Potts, and T. Möller. A parallel coordinates style interface for exploratory volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):71–80, 2005. 8.2

[115] E. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, 1983. 3.6.1

[116] F. Tzeng, E. Lum, and K. Ma. A novel interface for higher-dimensional classification of volume data. In *IEEE Visualization 2003*, 2003. 2.6.2

[117] M. van Kreveld, R. Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the 13th ACM symposium on Computational geometry*, pages 212–220, 1997. 3.3.1, 3.3.2, 3.3.4, 3.6, 3.6.2

[118] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, July-August 2005. 2.7

[119] L. Wang, J. Giesen, K. T. McDonnell, P. Zolliker, and K. Mueller. Color design for illustrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1739–1754, 2008. 2.6.6, 5.3.2, 5.3.2, 6.1, 6.5.2, 6.5.2, 6.5.2, 6.6

[120] L. Wang and K. Mueller. Harmonic colormaps for volume visualization. In *Proceedings of IEEE/EG Symposium on Volume and Point-Based Graphics*, pages 322–325, Los Angeles, USA, 2008. 2.6.6, 5.3.2, 6.1

[121] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proceedings of IEEE Visualization 2005*, pages 367–374, Minneapolis, MN, USA, 2005. 1.2.2, 2.7, 3.6.3

[122] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):330–341, 2007. (document), 1.1.2, 1.2.1, 1.2.3, 2.6.5, 2.6.7, 2.7, 3.4, 3.5, 3.5, 3.7, 3.6.2, 3.6.4, 5.3.1, 6.1

[123] D. Weiskopf, K. Engel, and T. Ertl. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):298–312, 2003. 6.4.2

[124] R. Westermann and T. Ertl. A multiscale approach to integrated volume segmentation and rendering. *Computer Graphics Forum (Proceedings of Eurographics '97)*, 16(3):117–129, 1997. 5.1

[125] C. Wittenbrink, T. Malzbender, and M. Goss. Opacity-weighted color interpolation for volume sampling. In *Symposium on Volume Visualization'98*, pages 135–142, 1998. 2.4

[126] X. Yuan, M. X. Nguyen, B. Chen, and D. H. Porter. Hdr volvis: High dynamic range volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):433–445, 2006. 2.6.3

[127] J. Zhou, Z. Wang, and K. D. Tönnies. Focal region-based volume rendering. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(5):665–677, 2006. 1.2.2, 2.7, 3.6.3

[128] Jianlong Zhou and Masahiro Takatsuka. Contour tree simplification based on a combined approach. Technical Report TR 624, School of Information Engineering, The University of Sydney, Australia, 2008. ISBN 9781742100593. 3.3.5, 6.4.4, 8.1

[129] Jianlong Zhou and Masahiro Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1481–1488, November-December 2009. 8.1

[130] Jianlong Zhou and Masahiro Takatsuka. Structural relationship preservation and depiction in volume rendering. *International Journal of Computer and Information Science*, 10(2):23–32, 2009. 8.1

[131] Jianlong Zhou and Masahiro Takatsuka. Structural relationship preservation in volume rendering. In R. Lee, G. Hu, and H. Miao, editors, *Computer and Information Science 2009*, volume 208 of *Studies in Computational Intelligence*, chapter 21, pages 229–238. Springer, 2009. 8.1

[132] Jianlong Zhou and Masahiro Takatsuka. Importance driven contour tree simplification. In *Proceedings of 3rd International Conference on Computational Intelligence and Industrial Application 2010*, 2010. 8.1

[133] Jianlong Zhou and Masahiro Takatsuka. On multiple measures of importance for contour tree simplification. *The Visual Computer (International Journal of Computer Graphics)*, 2011. To be submitted. 8.1

[134] A. J. Zomorodian. *Topology of Computing*. Cambridge University Press, Cambridge, UK, 2005. 1.1, 3.1.2, 1