Title

Topology-controlled volume rendering

Journal

IEEE Transactions on Visualization and Computer Graphics, 13(2)

ISSN

1077-2626

Authors

Weber, Gunther H
Dillard, Scott E
Carr, Hamish
et al.

Publication Date

2007-03-01

DOI

10.1109/TVCG.2007.47

Peer reviewed

Topology-Controlled Volume Rendering

Gunther H. Weber, *Member*, *IEEE Computer Society*, Scott E. Dillard, Hamish Carr, *Member*, *IEEE*, Valerio Pascucci, *Member*, *IEEE*, and Bernd Hamann, *Member*, *IEEE*

Abstract—Topology provides a foundation for the development of mathematically sound tools for processing and exploration of scalar fields. Existing topology-based methods can be used to identify interesting features in volumetric data sets, to find seed sets for accelerated isosurface extraction, or to treat individual connected components as distinct entities for isosurfacing or interval volume rendering. We describe a framework for direct volume rendering based on segmenting a volume into regions of equivalent contour topology and applying separate transfer functions to each region. Each region corresponds to a branch of a hierarchical contour tree decomposition, and a separate transfer function can be defined for it. The novel contributions of our work are 1) a volume rendering framework and interface where a unique transfer function can be assigned to each subvolume corresponding to a branch of the contour tree, 2) a runtime method for adjusting data values to reflect contour tree simplifications, 3) an efficient way of mapping a spatial location into the contour tree to determine the applicable transfer function, and 4) an algorithm for hardware-accelerated direct volume rendering that visualizes the contour tree-based segmentation at interactive frame rates using graphics processing units (GPUs) that support loops and conditional branches in fragment programs.

Index Terms—Direct volume rendering, transfer function design, topology, contour tree, simplification.

1 Introduction

Volume rendering is a standard technique used in scientific visualization. It is based on defining "optical" properties for points in the three-dimensional (3D) domain of a scalar field and computing resulting pixel intensities in an image plane [1]. Optical properties at a given point are normally based wholly or partially on the function value (and sometimes gradient magnitude). This approach assumes that scalar values, such as density values in a computed-tomography (CT) scan, map directly to physical properties such as tissue type. Even when volume rendering is applied to nonmedical data this model is often used [2], [3]. While this approach provides a structural overview of an entire data set, it is unable to distinguish between distinct features that share the same scalar value. If, for example, a region of interest is enclosed by an "uninteresting" region and both regions overlap in value range, occlusion generally prevents effective visualization of the interesting region.

Improved methods for defining transfer functions utilize additional derived quantities such as gradient magnitude [4], but they still apply the same transfer function uniformly throughout the domain. As a result, these methods, while effective at enhancing the perception of distinct material/tissue type interfaces, still do not allow a viewer to differentiate fully between separate features. Other methods use segmentation information to apply different transfer functions to classified regions [5], [6] or even to apply different rendering modalities [7], [8]. These techniques require segmentation information for each volume and rely either on manual segmentation or domain-specific segmentation algorithms.

The underlying weakness of all of these methods is that the determination of opacity properties either fails to distinguish between different features at similar isovalues, or is based on expensive and/or domain-specific segmentation methods. It is therefore desirable to have a general model of spatially local transfer function definition by exploiting topological definitions of what constitutes a feature. By incorporating topological information into direct volume rendering in a systematic fashion, we generalize existing topological inputs to transfer function design and provide methods that are practical for direct user manipulation or, eventually, automatic design of locally defined transfer functions.

Isosurface topology provides insight into the fundamental structure of a data set, independently of the application domain. Topology considers connected components in a relatively intuitive fashion that can be used for a wide variety of applications. The *contour tree* [9] provides expert users with additional information about their data and facilitates data exploration in the absence of strong domain-specific techniques. Topological analysis of scalar data sets can be used to identify interesting behavior of a scalar field and to aid in the data exploration process. Section 2 provides an overview of related work.

While specialized segmentations are applicable to particular problem domains, the power of our topology-based segmentation lies in its generality. Our approach is related to the work of Takeshima et al. [10], who used a topological abstraction called *volume skeleton tree* to define topological attributes as additional input for multidimensional transfer

- *G.H. Weber and S.E. Dillard are with the Visualization and Computer Graphics Research Group, Institute for Data Analysis and Visualization (IDAV), University of California, Davis, One Shields Avenue, Davis, CA 95616-8562. E-mail: {ghweber, sedillard}@ucdavis.edu.*
- *H. Carr is with the School of Computer Science and Informatics, University College Dublin, Belfield, Dublin 4, Ireland. E-mail: hamish.carr@ucd.ie.*
- *V. Pascucci is with the Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, Box 808, L-560, Livermore, CA, 94551. E-mail: pascucci@acm.org.*
- *B. Hamann is with the Visualization and Computer Graphics Research Group, Institute for Data Analysis and Visualization (IDAV) and the Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA 95616-8562. E-mail: hamann@cs.ucdavis.edu.*

Manuscript received 2 Feb. 2006; revised 26 May 2006; accepted 7 July 2006; published online 10 Jan. 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0008-0206.

functions. Our method extends their ideas and is more general. Instead of setting transfer functions according to fixed topological indices, such as depth of topological nesting, we allow a user to assign independent transfer functions to topologically distinct features which may or may not share the same indices. In contrast to the work of Takeshima et al., we define the opacity at a sample point based on its topological characterization rather than interpolating between the opacities at corners of the associated grid cell. In essence, our approach performs postclassification in volume rendering while Takeshima et al.'s approach uses preclassification. While the use of postclassification for medical data is controversial, it significantly reduces artifacts in rendered images [11] and is particularly suitable for simulated data. Furthermore, instead of using linear interpolation applied to a tetrahedron we consider trilinear interpolation for a hexaheadral cell, as hexahedral meshes are most commonly used for direct volume rendering of data given as samples on a regular, rectilinear mesh.

We have implemented topological data simplification for trivariate scalar data sets. Topological data simplification techniques simplify a data set by making its topology correspond to hierarchical simplifications of the contour tree, thus representing various levels of detail and showing significant features at a glance while still allowing fine details to be revealed on demand. Existing simplification methods result in geometric and visual artifacts such as fine thread-like structures and flat regions showing up as large objects abruptly becoming visible. We demonstrate how to remove topological features in a way that reduces these problems (Section 4). We also describe data structures (Section 3) and rendering algorithms for topological volume rendering (Section 5) that can be implemented efficiently using current graphics hardware. Our framework supports interactive volumetric visualization of noisy or topologically complex data.

2 Related Work

2.1 Contour Trees

In addition to its importance for isosurface extraction [12], topology has become valuable for more general exploration of scalar fields. Topology, up to now, has been used mainly in the context of isosurface extraction, where the topological properties of interest are usually the number of connected components and the genus of an isosurface, i.e., the number of independent tunnels or "holes" in an isosurface. Morse theory [13] shows that topological changes in scalar fields defined on manifolds occur at distinct isolated points called *critical points*. A Reeb graph [14] expresses the evolution of individual contours as a graph that is defined by these critical points and their relationships. For simply connected domains, the Reeb graph is always a tree structure, called a contour tree [9], which is more easily computed than the general Reeb graph.

Before discussing the contour tree in detail, we start with some basic definitions. The term *isosurface* has been used in the literature to mean the inverse image of an isovalue, an individual connected component of such an inverse image, or a triangular approximation of either. In mathematics, the term *level set* is usually used, instead of isosurface, but this term has other connotations in graphics and visualization. To avoid confusion, we use the terms isosurface and *contour* as follows: Given a field $f : \mathbb{R}^3 \Rightarrow \mathbb{R}$, the *isosurface* of f for an *isovalue* h is the inverse image $f^{-1}(h)$ of the isovalue. Since this isosurface may consist of multiple connected components, we use *contour* to refer to an individual connected component of an isosurface.

The contour tree is a structure that captures the topological evolution of an isosurface as the isovalue varies. Its nodes are critical points of the data set where the number of contours changes. Nodes of degree one (leaves of the tree) are minima and maxima where contours are created or destroyed. Interior nodes of degree three or higher are saddles where two or more contours merge or a single contour separates into multiple disconnected contours. Arcs of the contour tree represent contours between critical points, i.e., contours which do not change topology (with the exception of genus changes) as the isovalue varies between critical values. Algorithms for computing the contour tree [15], [16], [17], [18] were first defined for tetrahedral meshes using linear interpolation and later for hexahedral meshes using trilinear interpolation [19].

The contour tree may be augmented by additional nodes representing changes in topological genus of contours [19], nodes representing noncritical vertices of the mesh, or nodes representing any arbitrary points in the volume. We use the term *fully augmented contour tree* to refer to the contour tree augmented by all vertices of a mesh. Each grid vertex corresponds to a node in the fully augmented contour tree and maps to an arc (when a regular point) or a node (being a critical point) in the contour tree.

Since each contour in an isosurface corresponds one-to-one with a point on an arc in the contour tree, it is possible to use the contour tree as an index for a volume data set and identify all contours for a given isovalue. This relationship has also been used to find seed cells [17], [20] for the *continuation* method [21] of level set extraction.

2.2 Utilizing Topology in Scientific Visualization

Even if the topology is known, a user interface for effective data exploration must still be developed. Bajaj et al. [22] introduced an interface called the *contour spectrum*, where properties such as isosurface area, enclosed volume, and the contour tree were plotted alongside isosurfaces to provide users with additional cues to interesting isovalues. Weber et al. [23], [24] devised tools for exploring scalar fields based on detecting critical points and critical regions. Cox et al. [25] defined a digital Morse theory and also used critical points and regions to explore medical data. Fujishiro et al. [26], [27] used the contour tree to detect significant isovalues automatically for transfer function design.

Carr and Snoeyink [20] extended the idea of seeded isosurface extraction based on the contour tree [17] 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

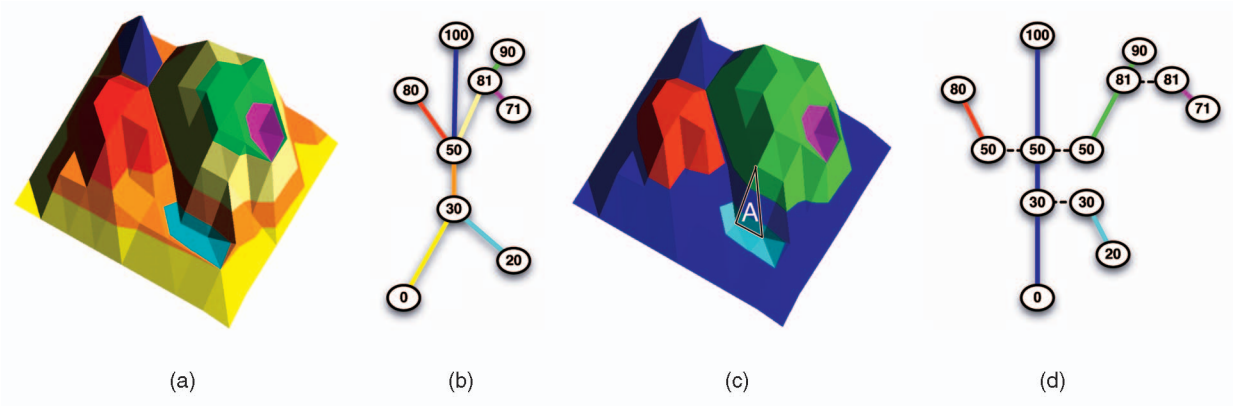


Fig. 1. Example of a segmentation defined by a contour tree. (a) Terrain data set showing topological zone segmentation. (b) Contour tree of the terrain, with edges color-coded based on the corresponding topological zones. (c) Terrain data set showing topological zone segmentation for branch decomposition. (d) Branch decomposition of contour tree of the terrain, with edges color-coded based on the corresponding topological zones.

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.

Takahashi et al. [28] extended this idea and used interval volumes to display volumetric regions of uniform topology, "peeling" away layers of a volume to examine the internal structure. Recently, Takeshima et al. [10] used topological information such as isosurface inclusion level in multi-dimensional transfer functions, to support operations such as "peeling" off layers in a volumetric data set.

2.3 Topology Simplification

Noise in a data set creates a large number of irrelevant critical points that can distract from truly relevant topological features. Topology simplification suppresses insignificant features by removing, or *cancelling*, pairs of critical points that are viewed unimportant according to a specified measure. Besides the contour tree, two topological structures are used widely for scalar topology simplification: the *volume skeleton tree* that corresponds to a contour tree augmented with nodes corresponding to genus changes [29], [30] and the *Morse-Smale complex* [31], [32], [33].

Our work is based on topology simplification methods introduced by Takahashi et al. [29], Carr et al. [34], and Pascucci et al. [35]. Carr et al. [34] simplified the contour tree with two basic operations: leaf pruning and node collapses. Leaf pruning removes a leaf and the arc incident to the leaf from the contour tree. Since contours are extracted from seeds stored in a contour tree, deleting an arc from the contour tree has the side effect of discarding the corresponding contours from further consideration. Carr et al. further show 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, i.e., all points corresponding to the pruned arc get assigned the value of the saddle. Node collapses remove degree-two vertices and do not affect the contours that can be extracted or values in the data set. Pruning and collapsing are performed in an order that minimizes the error based on a local geometric measure, such as *hypervolume* (an integral of the scalar field over the enclosed volume), *volume* or *persistence*, with node collapses having priority over leaf pruning.

Pascucci et al. [35] described *branch decomposition*, an efficient way for storing a hierarchy of contour tree simplifications. A branch decomposition of a tree is a hierarchical decomposition if 1) exactly one branch connects two leaves (called the root branch) and 2) every other branch connects one leaf to an interior node of another branch. Fig. 1b shows a contour tree, and Fig. 1d shows its branch decomposition. Since a branch (with the exception of the root branch) connects a leaf to an interior node of another branch, it 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. Discarding a branch from the branch decomposition is equivalent to performing a cancellation, which is equivalent to a vertex prune operation in the framework of Carr et al. [34].

2.4 Classification in Volume Rendering

Original approaches for volume rendering consider only the scalar value to classify samples [1], [3]. Even though Levoy's method [1] also considers gradient magnitude it is only used to simulate "surfaces of constant thickness" and does not affect classification. Kniss et al. [4] improved visualizations by adding gradient magnitude and additional higher-order derivatives as parameters for classification, while still applying the same transfer function uniformly throughout the domain.

If a segmentation of the data set is available, which is often true for medical data sets, it is possible to improve volume-rendered results by considering segmentation information during classification and rendering individual segmented regions differently. For example, it is possible to use different transfer functions for different tissue types, or to emphasize particular parts of the anatomy while hiding others [5], [6]. By combining multiple rendering modalities, such as *maximum intensity projection* and *direct volume rendering* with incorporation of different transfer functions [7], [8], it is possible to highlight volume portions of interest while still providing a context to the user. However, these techniques require segmentation information which must be obtained by manual segmentation or by domain-specific segmentation algorithms.

More recently, Takeshima et al. [10] used information from the contour tree to design multidimensional transfer functions which use derived topological quantities as input. In addition to genus, the authors designed transfer functions which express the "inclusion level" of a contour, defined as the number of equal-valued contours which surround and occlude that contour. Lower inclusion levels, which represent the outermost features, are given lower opacities so that higher inclusion levels, which represent the innermost features can be seen clearly. Furthermore, Takahashi et al. [36] used the volume skeleton tree to find optimal viewpoints that maximize the number of visible features for volume rendering.

Recently, statistical learning-based methods have been introduced for interactive volume data segmentation [37]. The idea underlying such methods is to have a user specify interactively what regions in a data set constitute a "feature." By pointing out such regions, it is then possible to characterize them by scalar field behavior in a local neighborhood, and to use the resulting characterization for segmentation. However, learning-based methods require manual training to define features.

Kniss et al. [38] improved on the classical classification by observing that a unique classification of samples is not always possible. In particular, in the proximity of boundaries it can improve visualization results by considering an uncertainty if classification and use statistical methods to generate "fuzzy" classifications.

In summary, a significant body of work exists concerned with the use of contour trees for interactive isosurface exploration, on their use in automatic transfer function design, and in designing globally uniform transfer functions based on parameters which may include some topological information. However, interactive exploration of scalar fields using topologically defined transfer functions in their most general sense has not previously been considered, and our paper discusses a solution to this problem.

3 Contour-Tree-Defined Segmentation

Since individual contours map to points in the contour tree, each arc of the tree represents the union of all contours which map to it. 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 we refer to as *topological zones*, following Cox et al. [25]. For example, in Fig. 1a, the sample terrain has been divided along each contour that passes through a critical point, and the topological zones are shown in different colors.

We also follow Takeshima et al. [10] by using topological zones for segmentation in volume rendering. However, their work has several limitations: their topological indices treat all zones with the same nesting depth as equivalent, ruling out differentiation between two topologically similar but distinct objects. Moreover, their methods are based on the topology of tetrahedral meshes, which can introduce geometric and topological artifacts in the output images [39]. Furthermore, they computed opacities for the volume rendering integral by interpolating over tetrahedra, instead of looking up opacities directly in the relevant transfer function, which can lead to further artifacts in cells spanning multiple topological zones.

We extend this work by allowing assignment of independent transfer functions to individual topological zones; by applying the trilinear interpolant over hexahedral cells; by showing how to find the correct opacity in cells spanning multiple topological zones; and by improving the treatment of topological simplification in the data.

We compute the contour tree of the underlying trilinear interpolant [12] with a variation of the method from Pascucci and Cole-McLaughlin [19], which applied the underlying graph-theoretic algorithm [18] to the union of cellwise join and split trees instead of the edges of a tetrahedral mesh. In fact, any graph which accurately represents the topology of the trilinear interpolant may be substituted instead. To simplify processing, we construct a widget in each cube in which the mesh edges are augmented by edges between body saddles, face saddles, and vertices [40]. Nothing, however, depends on this choice, and the method described in [19] can be substituted instead.

Once we have computed the contour tree, we simplify it to remove symbolic perturbation [41], then assign each topological zone a transfer function. Consequently, every topological detail, including noise, is assigned its own transfer function. For medium to large data sets, especially when physically sampled, the topological complexity makes it necessary to represent the contour tree hierarchically. We store the hierarchy using the *branch decomposition* described by Pascucci et al. [35].

In addition to providing a hierarchical representation of the contour tree, branch decomposition also avoids an "oversegmentation" of the volume. 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. Branch decomposition naturally concatenates a sequence of arcs into a single unit, see Fig. 1d, unifying their topological zones into a single zone, see Fig. 1c. Furthermore, branch decomposition permits user interactions on a coarse, simplified tree and propagating transfer functions down to the full resolution tree.

We construct the branch decomposition as follows: At each step, a saddle-extremum pair is pruned from the contour tree, and the remaining node is collapsed, leaving an arc. This arc is the parent to which the saddle-extremum pair is linked as a child. The order in which the pairs are pruned is determined by a priority measure. Pascucci et al. [35] used *persistence* as a priority measure, which is the absolute difference in function values between the saddle and extremum. Geometric measures, such as volume and hypervolume, are described by Carr et al. [34]. Pairs with lower priority are pruned first, and they become the children of pairs with higher priority. The end result is a rooted tree of saddle-extremum pairs, or branches. The most important branches, as indicated by the priority measure, are at the top of this tree.

Currently, we do not consider saddles corresponding to genus changes. Our zones of "topologically equivalent" contours are only topologically equivalent up to the genus. However, the same argument for using a branch decomposition to avoid an oversegmentation also holds here. Saddles which correspond to a genus change of a contour do not affect the segmentation—contours of different genus can be distinguished based solely on function value alone.

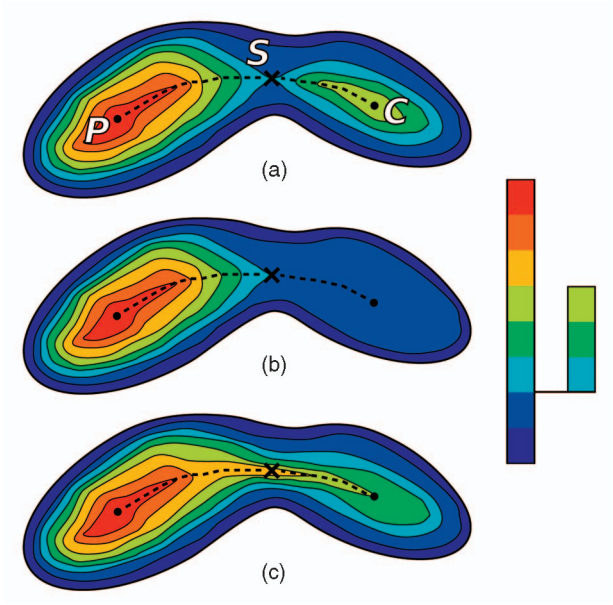


Fig. 2. (a) A function with two maxima; branch decomposition of contour tree shown on right side. (b) The right branch has been cancelled by flattening the right topological zone. (c) The right branch has been cancelled by reversing the gradient along the dotted line.

This is a design decision, however, and nodes for genus change could be represented if the additional information were desired. As genus changes do not contribute to the complexity of the contour tree-based segmentation, we do not consider them in our simplification scheme.

4 Improved Topological Simplification

One straightforward way to change isosurface topology is to "chop off" peaks, and correspondingly "fill in" valleys [34]. (Figs. 2a and 2b show an example, where the peak on the right-hand side (marked C) is chopped off.) This goal can be achieved by setting the transfer function for the canceled branch to a solid color, the color of the parent branch's transfer function at the value of the saddle which connects the child to the parent. While this approach displays an image with topology consistent with the pruned contour tree, the topological changes in the function may not be ideal from a user's perspective. Sweeping an isosurface down the simplified contour tree will show the pruned components "popping" abruptly into existence, as can be seen in the movie accompanying this paper (which can be found at http://www.computer.org/tvcg/archives.htm). In volume-rendered images, a region of uniform opacity properties can also become visible. For a smoother evolution of simplified isosurfaces, or for a closer approximation of the underlying scalar field, other methods must be used.

We follow the strategy of Pascucci et al. [35], which proves that one can always modify a scalar field to correspond to a simplified contour tree. The proof relies on reversing the gradient flow along a path from the saddle to the extremum of the branch to be pruned. In fact, the authors proved a stronger result, which states that the region in which the gradient flow is reversed can be made arbitrarily small. This condition is achieved by subdividing the mesh, resulting in arbitrarily thin thread-like structures connecting the former extremum to the remainder of the function. This approach, too, generates undesirable visual artifacts, and we instead reverse the gradient flow in a *tube*—a region inflated from the thread to a size defined by the following heuristic: sweep a sphere out from the saddle (S in Fig. 2). The base radius of the tube is the distance at which the sphere first intersects a topological zone other than the parent's or child's. This radius shrinks as the tube extends from the saddle toward the extremum.

Consider a branch that ends at a maximum, such as the branch SC in Fig. 2. The region of gradient reversal is a tube, the *child tube*, that starts at the saddle S and extends "upward" toward the maximum C. This child tube crosses each contour of SC's topological zone once, and at each crossing we define a new function value. The original values of the contours ascend from the saddle to the maximum. To reverse the gradient, we define new values that descend, as illustrated in Fig. 2c. Since the starting point of this tube, S, is lower than all values in the topological zone of C, the path cannot descend without creating a new minimum. To avoid creating this new minimum, we increase the value at S, by way of a *parent tube* which extends from the saddle into its parent's topological zone, PS in Fig. 2. The gradient along this parent tube is not reversed. Instead, the values are increased in order to elevate the saddle's function value. If the value at P is greater than the value at C (which is usually true for the hierarchical branch decomposition [35]) then it is possible to construct a monotonically decreasing path from P, through S, ending at C, thus effecting the simplification.

The tubes are created by a "best effort" process, which starts at the saddle and follows the steepest path in both directions. It stops if a maximum or minimum is reached, or if the path is obstructed by another topological zone. If the end of the parent tube does not have a value greater than the value at the end of the child tube, then a monotonically descending path cannot be constructed along the tube axis. In such a case, where the value at the highest point on the parent tube reaches a value v, we uniformly scale all values in SC's topological zone so that they are less than v.

If the child SC has children, then the child tube may also become obstructed. It becomes necessary to flatten all points in C's topological zone with value greater than at the end of the child tube. The tip of the child tube thus becomes the new maximum. In extreme cases where no significant tubes can be constructed, the method degenerates into the "chopping-off" method mentioned at the beginning of this section.

5 Volume Rendering

Volume rendering is based on integration of optical properties along a ray passing through the image plane. Samples are taken at regular intervals along this ray, and color and opacity are determined using a transfer function. Results are then composited. Various methods have been developed to accelerate this process, but all share the same conceptual model.

Instead of a global transfer function, we use a (possibly) distinct transfer function for each topological zone. For any given point p in the domain, we must therefore determine the topological zone to which p belongs, and then apply the appropriate transfer function. This task is complicated by the fact that the boundaries between topological zones are formed by isosurfaces at critical values, which rarely

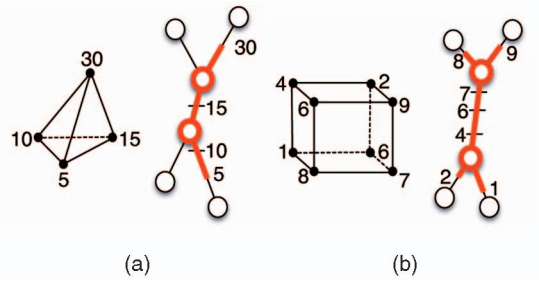


Fig. 3. Portion of contour tree corresponding to a cell. Vertices are labeled with their function values. These values are also highlighted in the contour tree to show the correspondence between vertices and nodes and arcs in the contour tree. (a) Tetrahedral cell, linear interpolation. (b) Hexahedral cell, trilinear interpolation.

coincide neatly with the mesh. Fig. 1a shows that the triangle labeled A intersects three topological zones, the middle of which contains no vertex of A. In general, cells can intersect arbitrarily many topological zones.

Our solution to this problem is based on the fact that monotone paths in any scalar field always map to monotone paths in the contour tree and vice versa. Given a monotone path P in the scalar field that passes through p and terminates at vertices corresponding to nodes in the contour tree, we trace the corresponding monotone path Q in the contour tree. Due to monotonicity, there will only be one point q on this path with the same isovalue as p; thus, we apply the transfer function for the branch on which q lies.

To determine a suitable monotone path, we observe that, for any point p, there always exists a monotone path P through p that starts at a local maximum and ends at a local minimum. Moreover, linear interpolation on tetrahedra and trilinear interpolation on hexahedra both guarantee that local extrema in the cell occur at cell vertices.

If linear interpolation is applied to a tetrahedral mesh, finding a monotone path is simple. In each tetrahedral cell, all points including the vertices belong to a single monotone path in the contour tree, as shown in Fig. 3a. This fact simplifies our task, as we do not need to find a monotone path explicitly. Instead, we use the path from highest to lowest-valued vertices of the tetrahedron.

Trilinear interpolation on hexahedra does not have this property, since saddles may occur inside or on faces of hexahedral cells, causing monotone paths in the cell to map to more than one possible monotone path in the contour tree, as shown in Fig. 3b. We instead base our construction on the linearity of the interpolant along lines parallel to the coordinate system, and construct a monotone path from axis-parallel segments. Fig. 4 shows an example for the bilinear case.

A monotone path through a trilinear cell is constructed according to Algorithm 1. At any point X, we choose an axis-parallel line and follow it to the boundaries of the cell. Since f is linear for every axis-parallel line, it must be either constant or monotone. If it is monotone, we repeat the process on the boundary of the cell until we reach a vertex, extending a monotone path as we proceed. For constant lines, the endpoints of the line belong to the same contour and the same topological zone as X, and we substitute either endpoint of the line for X and continue.

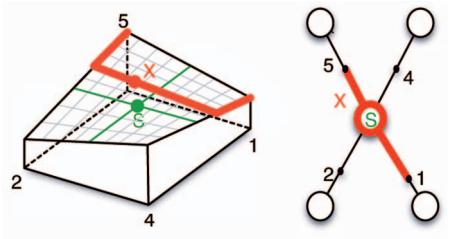


Fig. 4. Finding the arc corresponding to a point within a cell by following a monotone path. The cell contains a saddle (indicated by a green S) and, thus, a fork in the corresponding subgraph of the contour tree. By finding a monotone path in the cell, it is possible to find the arc in the contour tree that corresponds to the region containing the sample.

Input $t : [0,1]^3 \Rightarrow \mathbb{R}$ trilinear interpolant inside cell;
Sample position $\mathbf{p} \in [0,1]^3$ inside trilinear cell
Output: $\mathbf{p}_{\text{lo}}$ and $\mathbf{p}_{\text{hi}}$ positions of mesh vertices (end points of the monotone path through $\mathbf{p}$)

Create original monotone path to boundary faces:
$\mathbf{p}_{\text{lo}} = \text{roundDown}(x, \mathbf{p})$
$\mathbf{p}_{\text{hi}} = \text{roundUp}(x, \mathbf{p})$
if $\text{t}(\mathbf{p}_{\text{lo}}) > t(\mathbf{p}_{\text{hi}})$ **then** $\text{swap}(\mathbf{p}_{\text{lo}}, \mathbf{p}_{\text{hi}})$

Complete descending path to mesh vertex:
$\mathbf{temp} = \text{roundUp}(y, \mathbf{p}_{\text{lo}})$
$\mathbf{p}_{\text{lo}} = \text{roundDown}(\text{y}, \mathbf{p}_{\text{lo}})$
if $t(\mathbf{p}_{\text{lo}}) > t(\mathbf{temp})$ **then** $\text{swap}(\mathbf{p}_{\text{lo}}, \mathbf{temp})$
$\mathbf{temp} = \text{roundUp}(z, \mathbf{p}_{\text{lo}})$
$\mathbf{p}_{\text{lo}} = \text{roundDown}(\text{z}, \mathbf{p}_{\text{lo}})$
if $t(\mathbf{p}_{\text{lo}}) > t(\mathbf{temp})$ **then** $\text{swap}(\mathbf{p}_{\text{lo}}, \mathbf{temp})$

Complete ascending path to mesh vertex:
$\mathbf{temp} = \text{roundUp}(y, \mathbf{p}_{\text{hi}})$
$\mathbf{p}_{\text{hi}} = \text{roundDown}(y, \mathbf{p}_{\text{hi}})$
if $t(\mathbf{p}_{\text{hi}}) < t(\mathbf{temp})$ **then** $\text{swap}(\mathbf{p}_{\text{hi}}, \mathbf{temp})$
$\mathbf{temp} = \text{roundUp}(z, \mathbf{p}_{\text{hi}})$
$\mathbf{p}_{\text{hi}} = \text{roundDown}(z, \mathbf{p}_{\text{hi}})$
if $t(\mathbf{p}_{\text{hi}}) < f(\mathbf{temp})$ **then** $\text{swap}(\mathbf{p}_{\text{hi}}, \mathbf{temp})$

Algorithm 1: Algorithm for finding vertices of a cell where the monotone path through a sample starts and ends. The function $roundDown(axis, \mathbf{p})$ returns the position of $\mathbf{p}$ moved to the "lower" boundary along the specified axis, i.e., it replaces the component corresponding to the axis with zero. Analogously, $roundUp(axis, \mathbf{p})$ returns the position of $\mathbf{p}$ moved to the "upper" boundary along the specified axis, i.e., it replaces the component corresponding to the axis with one.

Once the high and low vertices are determined, we look up the high and low arcs. We then walk from the high arc towards the low arc, stopping at the arc which contains the sampled value of f. Contour trees are "free trees" without a root. It is not immediately obvious which direction to walk if we are trying to move from one arc to another. Instead of trying to navigate the contour tree directly, we walk in the branch decomposition, which is a rooted tree. Suppose we are trying to find the path between branches b and c. Since the tree is rooted, b and c must share a common ancestor a.

The path from b to c goes from b to a and then back down to c. This path is easily retrieved using the parent links of the branch decomposition. Further, we do not need to walk the entire path from b to a to c in order to find the branch we are looking for, the one which contains the sample point. Instead, we follow the paths from b to a and from c to a simultaneously. At each step, we walk up the parent link of the deeper of the two branches b and c. To avoid overshooting when walking in the direction of increasing function value, we stop when the step we are about to take goes past a critical point (a saddle) with a higher value than the sampled point. Similarly, when walking in the direction of decreasing function value, if the next step goes past a saddle with a lower value than the sample, we stop. If neither of these conditions is ever met, the walk will terminate when both paths reach the common ancestor, i.e., the desired branch.

6 User Interface for Transfer Function Design

To define transfer functions and apply them to topological zones, we have extended the *flexible isosurface* interface [20], which allows for direct user selection and manipulation of individual contours using the standard metaphor of "select-and-operate." Conceptually, we would like the user to select a topological zone directly in a rendered image by clicking the mouse in a zone of interest. To achieve this, we project a ray through the data originating from the corresponding pixel, and select the first "visible" topological zone. Visibility is determined by evaluating the volume rendering integral for a single ray passing through the volume and finding the first sample with an opacity above a certain threshold. For volume-rendered data, this approach is not always straightforward, especially when dealing with multiple layers of topological zones. Thus, we provide a slice view in which the user can select the topological zone's intersection with that slice. In either case, once we have determined a sample (along the ray) or a pixel (in a slice), we use the fully augmented contour tree to determine the topological zone to which the sample (or pixel) belongs.

We also provide the ability to directly select arcs in the contour tree, as in the flexible isosurface interface [20]. For this purpose, we display the contour tree in one of two representations. A TOPORRERY view in Fig. 11c shows the branch decomposition contour tree in a three-dimensional radial layout [35] where height corresponds to function value. We also provide a more traditional two-dimensional view of the contour tree as shown in Fig. 11b. In this view, the contour tree is drawn based on the convention that the y-coordinate corresponds to function value. We have also found it useful to include a vertical transfer function editor (as opposed to the more traditional horizontal layout) next to the contour tree interface. By linking the y-coordinates of the contour tree interface and transfer function editor, contextual topological information is availible to the user as they edit the transfer function.

7 Implementation

7.1 Hardware-Accelerated Volume Rendering

Volume rendering based on topological zone look-up can be implemented in graphics processing unit (GPU) fragment programs that support "for-loops," such as those of NV40-class NVIDIA graphics cards. The algorithms outlined in Section 5, the monotone path search and branch look-up, are straightforward to implement in a high-level shader language like Cg. The image data is stored as a 3D texture to which trilinear interpolation is applied, and the vertex-to-branch map is stored as a 3D texture to which nearest-neighbor interpolation is applied. The output of this texture are references (texture coordinates) to an array of branches. These branch records are stored logically in a 1D array, but since the size of any one texture dimension is limited, it becomes necessary to wrap this array as a 2D texture. The fields for each branch record are: a reference to its parent, the value at its saddle, its depth in the rooted branch decomposition, and a reference to its transfer function. The transfer functions are stored in another logically 1D, physically 2D, array. Transfer function references are texture coordinates, to which a function value is added to offset the texture coordinate and select the proper color.

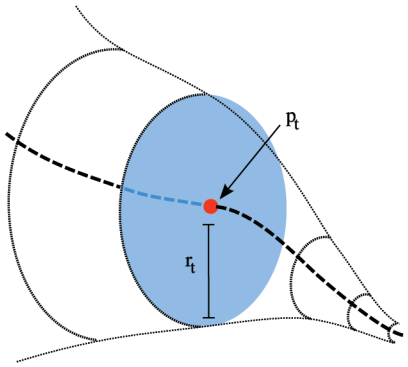


Fig. 5. Tubes are defined by a ball swept along the steepest path (dotted line) from a saddle to an extremum. At each isovalue along the path, the ball intersects the isocontour at value v, and this disk forms the cross-section of the tube. Near the center of the disk, $p_t(v)$, fragments assume the modified function value which reverses the gradient, $f_t(v)$. Outside the disk, fragments are left unchanged.

7.2 Visualization of Simplified Topology

Visualization of simplified topology is also implemented in graphics hardware. We render tubes between saddles and extrema dynamically, rather than modifying the scalar field beforehand. Each tube extends from a saddle to an extremum and is defined by a list of "tube points." To achieve gradient reversal, we define the new function values to be increasing along the tube, where they were previously decreasing (see Section 4). In addition to the new function value, we also define the radius for each tube point, which varies the thickness of the tube.

The path for a tube is defined in a preprocessing step. Since the tubes are monotone paths, each tube point falls on a unique contour and has a unique function value. Consequently, during rendering, we can look up tube points from a table, indexed by function value, similar to the way transfer functions are used. For each function value v, the associated position $p_t(v)$, tube radius $r_t(v)$, and "replacement function value" $f_t(v)$, i.e., the value that replaces the original value at that location, are stored, see Fig. 5.

We use a blending function b around the axis of the tube,

$$b = \frac{\sigma}{r_t(v)}\Big(r_t(v) - \|p_s - p_t(v)\|\Big),$$

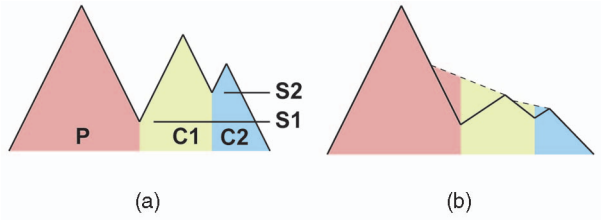


Fig. 6. Topological zone P is the parent of C1, which is the parent of C2. The contour tree branch of C2 is pruned, and the function values of C2 are scaled about the saddle value S2. Later, when the branch of C1 is pruned, the function values of C2 are scaled again about a different saddle value S1. There could be arbitrarily many allowable scalings, so all scaling transformations for a topological zone are composed into a scale-bias pair. (a) Unsimplified. (b) Simplified.

where p_s is the location of the current sample (i.e., the fragment currently considered by the fragment program) and v is the function value at that point. The values p_t and r_t are read from the look-up table (texture) that defines the current tube. The blending function specifies to what degree the replacement function value is used instead of the original function value at the current location. If it is zero, the replacement value has no effect on the current location. If it is one, the replacement value is used instead of the current function value. Between a blending function value of zero and one, the final function value f_{out} (i.e., the result of the fragment program) is obtained by linear interpolation between the original function value v_{in} at the current location and the corresponding replacement function value $f_t(v_{\text{in}})$, i.e.,

$$v_{out} = \text{clamp}(1-b)v_{in} + \text{clamp}(b)f_t(v_{in}), \text{where}$$

$$\text{clamp(x)} = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } 0 < x < 1, \\ 1 & \text{if } 1 \leq x. \end{cases}$$

We restrict the computed value of b to the interval [0, 1]. The parameter σ defines the width of a tube by controlling how quickly b falls off as one moves away from the tube's central axis. It is important that σ is larger than one; otherwise, b attains its maximum value only along the axis. A larger σ value, such as two, ensures that the interior volume of the tube with a b value of one is large enough to be visible during rendering. For each voxel, the closest tube in its topological zone is used to compute a modified function value. This closest tube is determined in a preprocessing step by checking the parent tube and all child tubes of that zone and identifying the tube which is closest according to the blending function b. A reference to the appropriate tube is stored in a 3D texture and accessed during rendering.

In addition to these modifications, we may also need to scale the function values of an entire topological zone (as mentioned at the end of Section 4.) Recall that if the parent tube cannot be extended to a point with greater (less) function value than the child maximum (minimum), then the function values of the child zone are scaled down (up) to permit a monotonic path from the parent extremum to the child. This scaling is a uniform 1D scaling transformation about the saddle value. We store the parameters of this transformation as a scale-and-bias pair, one for each branch, in the branch textures. The scaling is performed on the interpolated fragments during rendering. This is done before the fragment is modified by the tube blending function, so the tube replacement function value should reflect the scale and bias transformation.

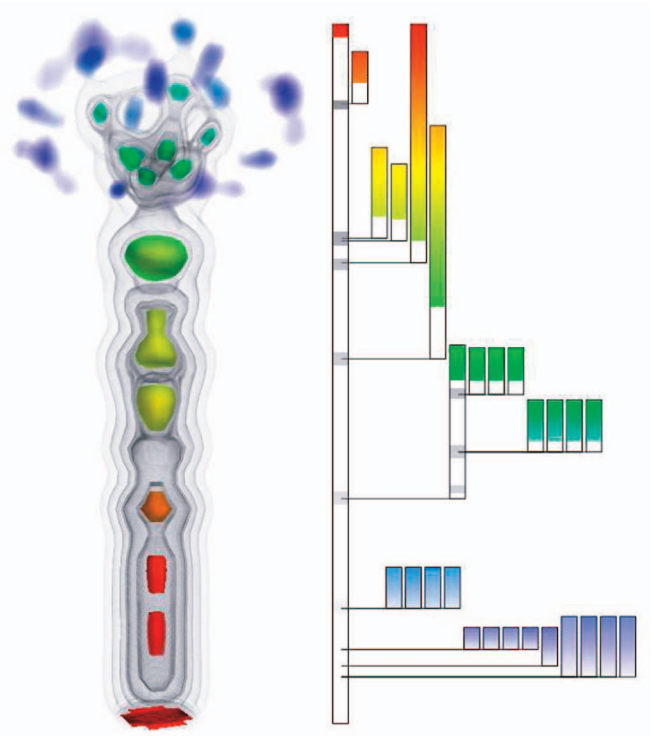


Fig. 7. Rendering of fuel data set consisting of $64 \times 64 \times 64$ voxels. This image was generated using a high sampling rate to capture high-frequency details in the transfer function. The tree was simplified by hypervolume.

In more complex scenarios, where a simplified branch contains children of its own, these children may need to be scaled again to effect their own simplifications. In Fig. 6, the topological zone labeled C2 first is scaled about the saddle value S2 when it is simplified. It is scaled again about saddle value S1 when C1 (the parent of C2) is simplified. Since a topological zone can have many parents, we compose all of the scaling transformations into a single scale-and-bias pair. When a user prunes a branch, the resulting scaling transformation s is applied to all of the branch's children by multiplying each childs scale-bias pair by s. Similarly, when the pruning operation is reversed, the inverse transformation s^{-1} is applied to all of b's children.

8 Results

Fig. 7 shows 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 a contour tree-based segmentation, it is possible to reveal internal structures (shown in color) while rendering surrounding structures using a low opacity in gray scale to provide context. Before interaction, the branch decomposition of the contour tree was simplified to 26 branches based on hypervolume. It is shown next to the rendered image to clarify the assignment of transfer functions to branches. Oversampling was used to obtain high-quality approximations of the gray-level isosurfaces that provide context.

Fig. 8a shows 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 a ^{16}O nucleus, see http://www.volvis.org/. In addition to simplifying the branch decomposition to five branches for interaction, an extraneous branch was manually pruned. Here, we

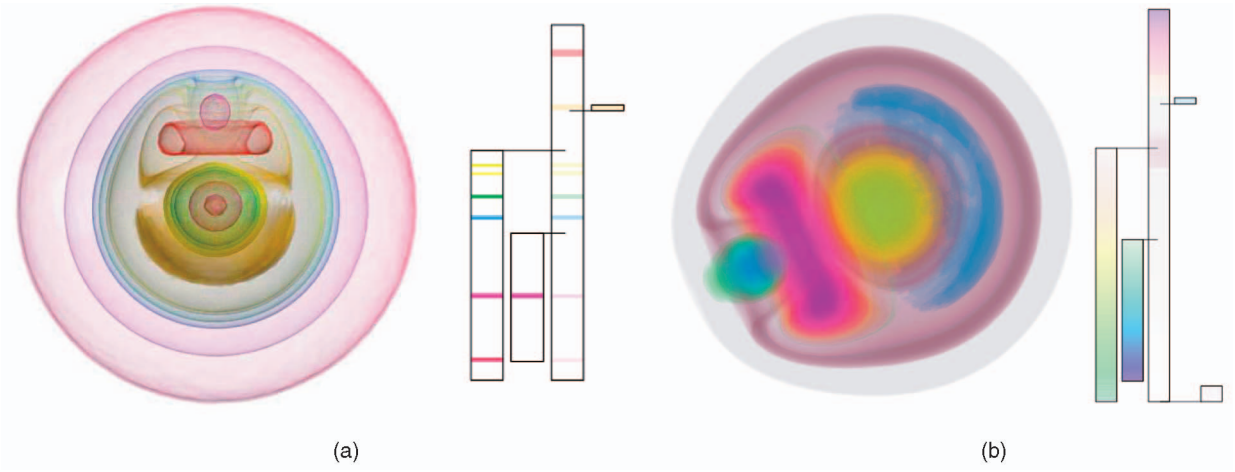


Fig. 8. Rendering of nucleon data set consisting of $41 \times 41 \times 41$ voxels. This image was generated using a high sampling rate to capture high-frequency details in the transfer function. The tree was simplified by hypervolume, and one extraneous branch was manually pruned.

used the topologically-based segmentation to simulate one of the topological attributes in Takeshima et al.'s work. We modified the opacity manually based on inclusion level to improve visibility of internal structures. Fig. 8b shows the same data set with a transfer function that emphasizes isosurfaces less and shows more of the internal structure of the nucleon. We emphasize regions around two minima in the "interior" while deemphasizing the "exterior" by choosing a much higher transparency and less saturated colors. Without the use of a global transfer function, it would not be possible to highlight these internal structures without having them occluded by external regions characterized by the same value range.

Figs. 9 and 10 show the results of applying our method of topology simplification to the fuel data set. In Fig. 9, we see an outside, isosurface-like view showing that the two separate components, corresponding to the "crown" and "shaft," have been merged. The appearance of the crown is changed somewhat—that topological zone has been scaled down to allow for a monotone path to be created between the two components.

In Fig. 10, the same data set is shown by means of a single slice through the center of the data, illustrating the evolution of the interior values of the data set. In this figure, the unsimplified data is shown at the top, followed by versions simplified with flattening [20], gradient reversal [35], and our new method. As we can see, the effect of flattening in Fig. 10b is to replace a region of varying isovalues with a single isovalue. In contrast, gradient reversal in Fig. 10c bridges the saddle, allowing some of the higher isovalues to remain. However, it is apparent both in the connection of crown to shaft at the right hand end, and the interior development at the left-hand end, that this

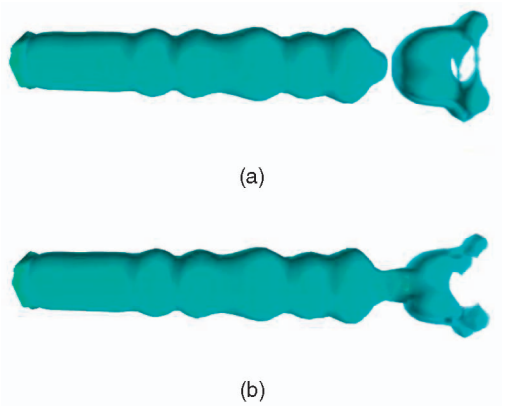


Fig. 9. Isosurface-like volume rendering of the fuel data set, showing (a) the original data and (b) the simplification with a single tube between the "crown" and "shaft." The change in appearance of the crown results from scaling down the entire topological zone, which is equivalent to computing an isosurface for a lower isovalue.

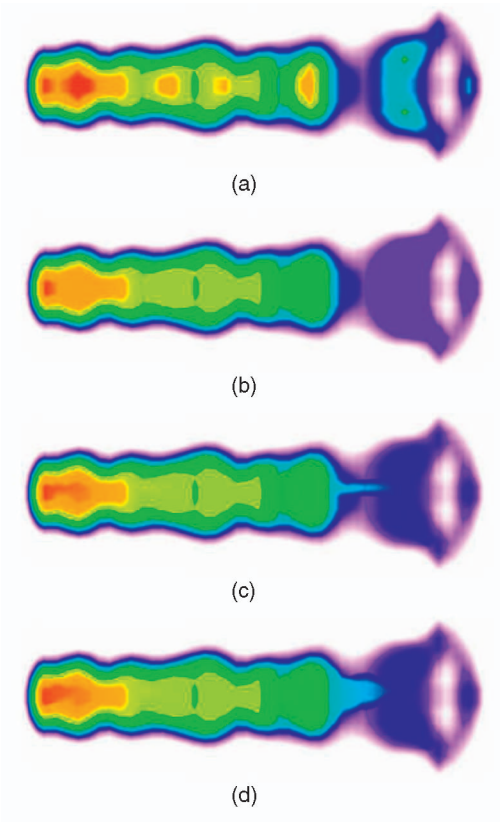


Fig. 10. Slice rendering of the fuel data set, showing (a) unsimplified and (b) simplified versions obtained by flattening, (c) gradient reversal, and (d) wide tubes.

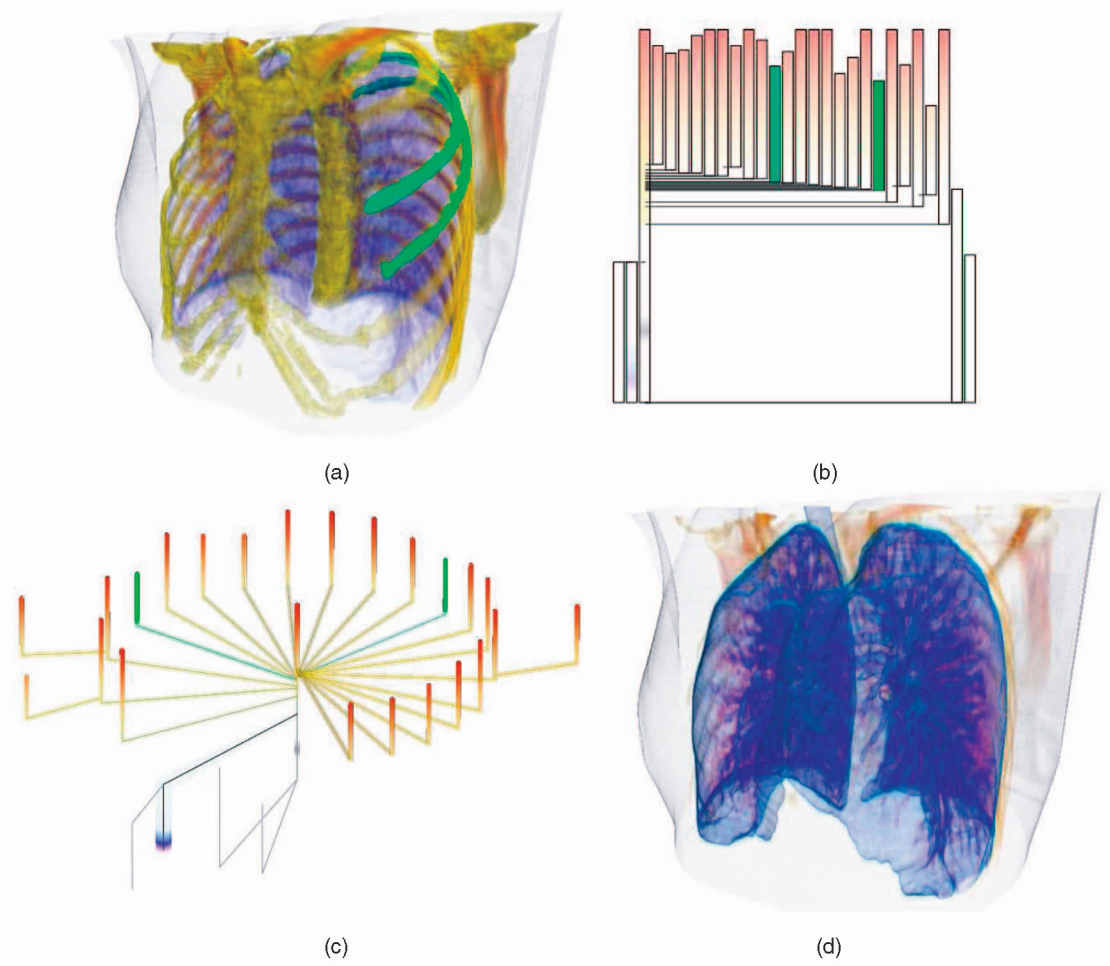


Fig. 11. Normal chest CT data set consisting of $384 \times 384 \times 240$ voxels. In (a), two ribs are highlighted in green; the lung is shown in violet. The tree has been simplified to 28 branches based on hypervolume. (b) shows the flat tree layout, and (c) shows the orrery layout. In (d), the ribs have been removed and the lung is emphasized.

Data set	Voxels	Pre-processing	Branches	Transfer function texture	Branch texture	Framerate
Fuel	$64 \times 64 \times 64$	2 sec.	86	$256 \times 8 \times 32$ bits	$256 \times 1 \times 80$ bits	21 fps
Aneurism	$256 \times 256 \times 256$	2 min., 6 sec.	31,850	$1024 \times 1024 \times 32$ bits	$256 \times 128 \times 80$ bits	2.5 fps
Chest CT	$384 \times 384 \times 240$	42 min., 45 sec.	99,406	$1024 \times 512 \times 32$ bits	$512 \times 256 \times 80$ bits	1.3 fps

Fig. 12. Frame rates and texture usage for Fuel, Aneurism, and Normal Chest CT data sets. Measurements were taken with an NVIDIA 7800 GPU. Frame rates are for 512×512 images, using approximately one sample per voxel per ray. Texture usage reflects the transfer function and branch textures. We used 32 bits per voxel for the voxel-to-branch map, and eight bits per voxel for the data itself.

narrow connection does a poor job of approximating the overall field. Finally, Fig. 10d shows the effect of the swept tubes, which broaden the connections to produce a smoother connection between components.

The value of this simplification method can also be observed in the movie accompanying this paper, which shows the evolution of the fuel data set using volume renderings for which a thin ramp function is varied over the isovalue range. In this movie, the data is shown in three forms: unsimplified, simplified by gradient reversal, and simplified with tube sweeping. As we see, the tube sweeping is a better approximation of the scalar field than the gradient reversal, exactly as predicted. Furthermore, simplified components gradually grow instead of popping up immediately, leading to smoother transitions and giving a user a better understanding of connectivity between components.

Fig. 11a shows a normal chest CT scan, available at http://radiology.uiowa.edu/downloads. It consists of $384 \times 384 \times 240$ voxels and shows how our segmentation performs on a moderately sized real-world data set. The tree, shown in Fig. 11b, has been simplified using hypervolume as underlying priority measure. Two ribs are highlighted in green. Note that the segmentation does not capture the entire rib; it shows only the regions of high density which are isolated from the spine and sternum. In Fig. 11d, the ribs have been made transparent in order to emphasize the lungs. The low-density regions near the lung-air interface, shown in blue, are easily separated from those near the skin-air interface, show in gray.

Fig. 12 summarizes memory utilization and frame rates for a hardware implementation of our method. The "aneurism" data set, see http://www.volvis.org/, is included as a medium-size example. For small data sets, such as the "fuel" data set, interactive frame rates are possible. However, frame rates for larger data sets are still sufficiently high for interactive transfer function

specification. Rendering measurements are based on a 1.8 gigahertz AMD Athlon 64 system with one gigabyte of RAM and an NVIDIA 7800 graphics board with 256 megabytes of graphics RAM. Rendered images consist of 512×512 pixels. Due to memory requirements of slightly more than one gigabyte of RAM for the "chest CT" data set, preprocessing measurements were performed on a three gigahertz Xeon system with four gigabytes of RAM.

9 Conclusions and Future Work

We have shown how to extend topological segmentation of the domain of trivariate scalar fields to direct volume rendering; how to define and edit spatially local transfer functions based on the topological structure encapsulated in the contour tree; and how to implement ray casting for topological zones. We believe that this method of topologically defining a segmentation greatly increases the flexibility of transfer function design and significantly extends the utility of direct volume rendering.

We plan to assess the impact of different contour tree simplification measures on resulting visualizations. Future work will also be directed at interface design. We plan to consider different visual representations of the contour tree, such as representations that show nesting properties of contours [42], [43] and also to incorporate genus changes [19] in the user interface. While adding these saddles to the augmented contour tree does not change the segmentation as no branching occurs, they can still provide additional valuable information.

In addition to assigning independent transfer functions to topological zones, we would use different material properties, or even entirely different rendering modalities as discussed in [7], [8]. It would be interesting to combine our method with some of the user interface elements described by Takahashi et al. [28] and Kniss et al. [4]. In particular, the ability to drag topological zones to different positions and generate an exposed view would be a powerful user interface component.

Acknowledgments

This work was supported by the US National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI) and a large Information Technology Research (ITR) grant; the National Institutes of Health under contract P20 MH60975-06A2, funded by the National Institute of Mental Health and the US National Science Foundation; the National Institutes of Health under contract 1R01 GM70444-01A1, funded by the National Institute of General Medical Science; by the Director, Office of Science, US Department of Energy under contract DE-AC03-76SF00098; and the Lawrence Berkeley National Laboratory (LBNL). The authors thank Oliver Kreylos for his slice generation code used in the volume renderer, the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis, and the members of the School of Computer Science and Informatics at University College Dublin for their support.

References

[1] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications,* vol. 8, no. 3, pp. 29-37, May 1988.

[2] N.L. Max, "Optical Models for Volume Rendering," *IEEE Trans. Computer Graphics,* vol. 1, no. 2, pp. 99-108, 1995.

[3] P. Sabella, "A Rendering Algorithm for Visualizing 3D Scalar Fields," *Computer Graphics (Proc. ACM SIGGRAPH '88),* vol. 22, no. 4, pp. 51-58, 1988.

[4] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional Transfer Functions for Interactive Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 8, no. 3, pp. 270-285, 2002.

[5] U. Tiede, T. Schiemann, and K.H. Höhne, "High Quality Rendering of Attributed Volume Data," *Proc. IEEE Visualization Conf. '98,* pp. 255-262, 1998.

[6] B. Preim, D. Selle, W. Spindler, K.J. Oldhafer, and H.-O. Peitgen, "Interaction Techniques and Vessel Analysis for Preoperative Planning in Liver Surgery," *Proc. Third Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI),* pp. 608-617, 2000.

[7] H. Hauser, L. Mroz, G.-I. Bischi, and M.E. Gröller, "Two-Level Volume Rendering—Fusing MIP and DVR," *Proc. IEEE Visualization Conf. '00,* pp. 211-218, 2000.

[8] M. Hadwiger, C. Berger, and H. Hauser, "High-Quality Two-Level Volume Rendering of Segmented Data Sets on Consumer Graphics Hardware," *Proc. IEEE Visualization Conf. '03,* pp. 301-308, 2003.

[9] R.L. Boyell and H. Ruston, "Hybrid Techniques for Real-Time Radar Simulation," *Proc. IEEE 1963 Fall Joint Computer Conf.,* pp. 445-458, 1963.

[10] Y. Takeshima, S. Takahashi, I. Fujishiro, and G.M. Nielson, "Introducing Topological Attributes for Objective-Based Visualization of Simulated Datasets," *Volume Graphics,* A.E. Kaufman, K. Mueller, E. Gröller, D.W. Fellner, T. Möller, and S.N. Spencer, eds., pp. 137-145, Eurographics Assoc., 2005.

[11] K. Engel, M. Hadwiger, J.M. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-Time Volume Graphics,* chapter 4. A.K. Peters, Ltd., pp. 89-92, 2006.

[12] G.M. Nielson, "On Marching Cubes," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 3, pp. 341-351, July-Sept. 2003.

[13] J.W. Milnor, *Morse Theory.* Princeton Univ. Press, May 1963.

[14] G. Reeb, "Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique," *Comptes Rendus de l'Acadèmie des Sciences de Paris,* vol. 222, pp. 847-849, 1946.

[15] 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,* vol. 14, no. 3, pp. 181-192, 1995.

[16] T. Itoh and K. Koyamada, "Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 4, pp. 319-327, 1995.

[17] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D.R. Schikore, "Contour Trees and Small Seed Sets for Isosurface Traversal," *Proc. 13th Ann. ACM Symp. Computational Geometry (SoCG),* pp. 212-220, 1997.

[18] H. Carr, J. Snoeyink, and U. Axen, "Computing Contour Trees in All Dimensions," *Computational Geometry—Theory and Applications,* vol. 24, no. 2, pp. 75-94, Feb. 2003.

[19] V. Pascucci and K. Cole-McLaughlin, "Parallel Computation of the Topology of Level Sets," *Algorithmica,* vol. 38, no. 2, pp. 249-268, Oct. 2003.

[20] H. Carr and J. Snoeyink, "Path Seeds and Flexible Isosurfaces Using Topology for Exploratory Visualization," *Data Visualization 2003 (Proc. VisSym '03),* pp. 49-58, 2003.

[21] G. Wyvill, C. McPheeters, and B. Wyvill, "Data Structure for Soft Objects," *The Visual Computer,* vol. 2, pp. 227-234, 1986.

[22] C.L. Bajaj, V. Pascucci, and D.R. Schikore, "The Contour Spectrum," *Proc. IEEE Visualization Conf. '97,* pp. 167-173, Oct. 1997.

[23] G.H. Weber, G. Scheuermann, H. Hagen, and B. Hamann, "Exploring Scalar Fields Using Critical Isovalues," *Proc. IEEE Visualization Conf. '02,* pp. 171-178, 2002.

[24] G.H. Weber, G. Scheuermann, and B. Hamann, "Detecting Critical Regions in Scalar Fields," *Data Visualization Conf. (Proc. VisSym '03),* pp. 85-94, 2003.

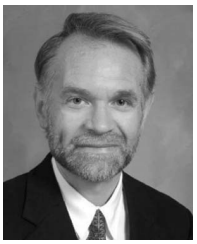
[25] J. Cox, D.B. Karron, and N. Ferdous, "Topological Zone Segmentation of Scalar Volume Data," *J. Math. Imaging and Vision,* vol. 18, pp. 95-117, 2003.
[26] I. Fujishiro, Y. Takeshima, T. Azuma, and S. Takahashi, "Volume Data Mining Using 3D Field Topology Analysis," *IEEE Computer Graphics and Applications,* vol. 20, no. 5, pp. 46-51, Sept./Oct. 2000.
[27] I. Fujishiro, T. Azuma, and Y. Takeshima, "Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topology Analysis," *Proc. IEEE Visualization Conf. '99,* pp. 467-470, Oct. 1999.
[28] S. Takahashi, I. Fujishiro, and Y. Takeshima, "Interval Volume Decomposer: A Topological Approach to Volume Traversal," *Visualization and Data Analysis Conf. '05 (Proc. SPIE),* 2005.
[29] S. Takahashi, Y. Takeshima, and I. Fujishiro, "Topological Volume Skeletonization and Its Application to Transfer Function Design," *Graphical Models,* vol. 66, no. 1, pp. 24-49, Jan. 2004.
[30] S. Takahashi, G.M. Nielson, Y. Takeshima, and I. Fujishiro, "Topological Volume Skeletonization Using Adaptive Tetrahedrization," *Proc. Geometric Modeling and Processing Conf. '04,* pp. 227-236, 2004.
[31] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Morse-Smale Complexes for Piecewise Linear 3-Manifolds," *Proc. 19th ACM Symp. Computational Geometry,* pp. 361-370, 2003.
[32] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, "A Topological Hierarchy for Functions on Triangulated Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 4, pp. 385-396, 2004.
[33] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann, "Topology-Based Simplification for Feature Extraction from 3D Scalar Fields," *Proc. IEEE Visualization Conf. '05,* pp. 535-542, Oct. 2005.
[34] H. Carr, J. Snoeyink, and M. van de Panne, "Simplifying Flexible Isosurfaces Using Local Geometric Measures," *Proc. IEEE Visualization Conf. '04,* pp. 497-504, Oct. 2004.
[35] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli, "Multi-Resolution Computation and Presentation of Contour Trees," Lawrence Livermore Nat'l Laboratory, Technical Report UCRL-PROC-208680, preliminary version appeared in the *Proc. IASTED Conf. Visualization, Imaging, and Image Processing (VIIP '04),* pp. 452-290, 2005.
[36] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita, "A Feature-Driven Approach to Locationg Optimal Viewpoints for Volume Visualization," *Proc. IEEE Visualization Conf. '05,* pp. 495-502, Oct. 2005.
[37] F.-Y. Tzeng, E. Lum, and K.-L. Ma, "An Intelligent System Approach to Higher-Dimensional Classification of Volume Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 3, pp. 273-284, May/June 2005.
[38] J. Kniss, R.V. Uitert, A. Stephens, G.-S. Li, T. Tasdizen, and C. Hansen, "Statistically Quantitative Volume Rendering," *Proc. IEEE Visualization Conf. '05,* pp. 287-294, Oct. 2005.
[39] H. Carr, T. Möller, and J. Snoeyink, "Simplicial Subdivisions and Sampling Artifacts," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 2, pp. 231-242, 2006.
[40] H. Carr, "Topological Manipulation of Isosurfaces," PhD dissertation, Univ. of British Columbia, Apr. 2004.
[41] H. Edelsbrunner and E.P. Mücke, "Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms," *ACM Trans. Graphics,* vol. 9, pp. 66-104, 1990.
[42] Y. Shinagawa, T.L. Kunii, and Y.L. Kergosien, "Surface Coding Based on Morse Theory," *IEEE Computer Graphics and Applications,* vol. 11, no. 5, pp. 66-78, 1991.
[43] S. Takahashi, Y. Shinagawa, and T.L. Kunii, "A Feature-Based Approach for Smooth Surfaces," *Proc. Fourth ACM Symp. Solid Modeling and Applications (SMA '97),* pp. 97-110, 1997.

Gunther H. Weber received the PhD degree in computer science from the University of Kaiserslautern in 2003. His research on visualization of adaptive mesh refinement data and topology-based methods for the exploration of volume data was performed in close collaboration with the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis, and the Lawrence Berkeley National Laboratory (LBNL). Currently, he is a postdoctoral researcher at IDAV and the Berkeley Drosophila Transcription Network Project at LBNL. He is a member of the IEEE Computer Society.

Scott E. Dillard received the BS degree in computer science from the University of California, Davis (UC Davis), in 2004. He is currently a PhD student at UC Davis, working with the Institute for Data Analysis and Visualization (IDAV). His research interests include graphics, visualization, image processing, and computational geometry.

Hamish Carr completed the PhD degree at the University of British Columbia in May 2004 and was appointed as a lecturer at the University College Dublin, effective September 2004. His research interests center around the application of topological analysis to computer graphics and to scientific and medical visualization, but more broadly include computer graphics, computational geometry, and geometric applications. He is a member of the IEEE and the IEEE Computer Society.

Valerio Pascucci received the PhD degree in computer science from Purdue University in May 2000 and the EE Laurea (master's), from the University "La Sapienza" in Rome in December 1993, as a member of the Geometric Computing Group. He has been a computer scientist and project leader at the Lawrence Livermore National Laboratory, Center for Applied Scientific Computing (CASC) since May 2000. Prior to his CASC tenure, he was a senior research associate at the University of Texas at Austin, Center for Computational Visualization, CS and TICAM Departments. He is a member of the IEEE and the IEEE Computer Society.



Bernd Hamann received the PhD degree in computer science from Arizona State University in 1991. He serves as an Associate Vice Chancellor for Research and is a professor of computer science at the University of California, Davis. His main interests are visualization, geometric modeling, and computer graphics. He was awarded a 1992 US National Science Foundation (NSF) Research Initiation Award, a 1996 NSF CAREER Award, and a 2006 University of California Presidential Chair in Undergraduate Eduction. He is a member of the IEEE and the IEEE Computer Society.