Local WYSIWYG Volume Visualization

Hanqi Guo Xiaoru Yuan *

Key Laboratory of Machine Perception (Ministry of Education), and School of EECS, Peking University
Center for Computational Science and Engineering, Peking University

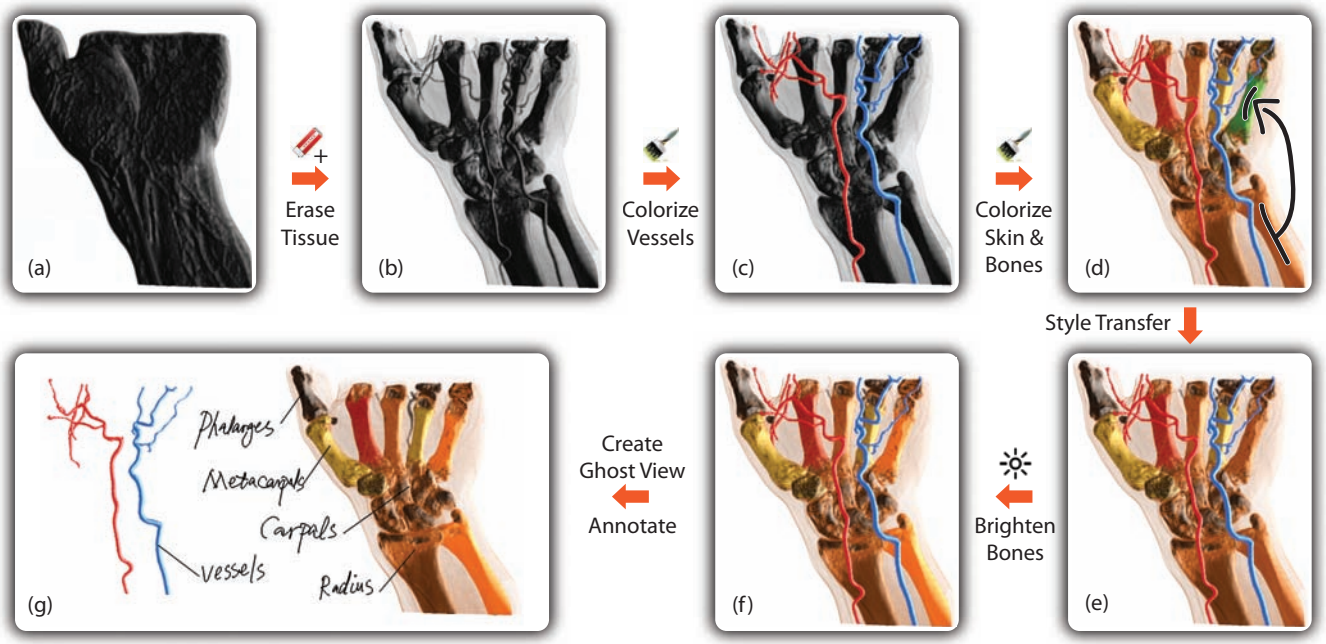


Figure 1: A CT scan of a hand explored and visualized using our Local WYSIWYG Volume Visualization System. (a) The initial rendering of the raw data; (b) the dense outer part of the volume removed with eraser tool; (c) the vessels are painted with different colors; (d) the bones and the skin are painted with colors; (e) the rendering style of a metacarpal is transferred from another bone by sketching; (f) the brightness of the bones in the previous operations are adjusted by painting; (g) the vessels are selected and moved to create a ghost view with annotations.

Abstract

In this paper, we propose a novel volume visualization system enabling local transfer function specification through direct painting or sketching on the rendered image, in a WYSIWYG style. Localized transfer functions are defined on scalar topology regions specified by the user. Intelligent and fast feature inference algorithms have been developed to convert user's input to the region specification and to achieve desirable feature styles with the local transfer functions. In our system, users can not only manipulate the color appearance of the object volume, but also apply style transfer and generate various illustration styles with a unified input gesture. Without manual transfer function editing and without parameter specification, our system is capable of generating informative illustrations that intuitively highlight user specified local features.
Keywords: Volume visualization, Local transfer functions, WYSIWYG, Scalar field topology, Sketch-based user interface.

1 Introduction

For understanding and gaining insight into sophisticated volumetric scalar fields, volume visualization has been widely applied in many areas, ranging from medicine and engineering to physics and biology. Effective interactive visual exploration of volumetric scalar fields is essential for successful volume visualization. As graphics hardware is advancing and new rendering algorithms have being developed, modern volume rendering software is more and more able to produce high quality images at sufficiently high frame rates for moderately sized data sets. Furthermore, over the last decade volume rendering techniques are been developed to a high degree of sophistication. Despite all this progression, there still remains a need for well designed visualization systems that fully support intuitive and flexible volume data exploration.

Transfer functions map voxel values to visual properties such as colors and opacities. Essentially, they give a classification of the features in the volume data. Their design is crucial to obtain effective volume visualization results and is one of the aspects that researchers have been devoted to support. However, assigning different voxels to corresponding colors and opacities, according to their structure and characteristics in the data, is not trivial. A common practice of transfer function design is to create a color/alpha value lookup table from a density curve. Real-time feedback from the rendered volume images gives visual hints for further adjusting this transfer function. However, such straightforward classification fails to distinguish features, whose distributions overlap with other components in the density range. More recently, derivatives, such as gradient magnitude [20], curvatures [18], and ambient occlusion [7], have been exploited to improve classification in transfer function design. Nevertheless, it is still challenging to build one-to-one mappings from all actual components to the feature space, even with state-of-the-art transfer function design technologies. An improvement for classification in transfer function design is to con-

*e-mail: {hanqi.guo,xiaoru.yuan}@pku.edu.cn

sider a voxel's 3D spatial position. However, it is commonly considered to be a difficult task to interact with complex objects in 3D.

Furthermore, in volume visualization, there is a need to distinguish different functional structures with similar material properties. One example is formed by vessels in medical volumetric data. Veins and arteries take different roles in the blood transportation. Consequently, they are labeled with different colors in most medical textbook illustrations. One way to solve both aforementioned problems in transfer function design is to replace a globally defined transfer function with several localized ones. However, associating the location information in transfer function classification is not trivial and often requires intensive manual intervention from domain users. Before a good visualization result can be manually obtained, the users tune the transfer function by trial and error, while mentally building the mapping from the feature space to the visual components. This traditional way of designing a transfer function is unintuitive and increases the design complexity even more.

In our recent paper, we proposed WYSIWYG volume visualization [13], which allows intuitive direct operations on volume rendered image instead of transfer function tuning. The effects of the brushing operations are applied to similar features in the global scope of the whole volume. In this work, we further develop a new transfer function design system which takes advantage of the spatial information for classification. In our system, the spatial information awareness is achieved by allowing the users to operate directly on the rendered images of the volume data with sketching and brushing, and in a WYSIWYG style [13]. In addition, the color, transparency, as well as the brightness of the local features can be interactively changed by painting on the image by different tools. In the paradigm of local transfer function, features with indistinguishable density distribution, can be classified differently due to their spatial location and connectivity. The contour topology [2] and its simplification [23] are general-purposed and mathematically sound in scalar field analysis and are used to subdivide the volumetric data in a preprocessing stage, and later on to assist in feature location specification. The main contributions of this work can be summarized as follows:

- We propose a novel WYSIWYG local transfer function design which allowing users to apply effects on topological *local* features;
- Flexible painting, sketching, combined with gesture user interactions are designed to help users explore and operate on local features effectively.

Compared to the previous global WYSIWYG volume visualization [13], the fundamental difference is the introduction of locality. On one hand, introducing locality can not only distinguish more meaningful features, but also reduce the confusion in user interaction. The local brushed region can exclusively associate the local features in the volume data, without affecting all similar features in the global scope. On the other hand, designing local transfer functions also needs intuitive WYSIWYG approach without resorting to the complex parameter space.

In the remainder of the paper, Section 2 summarizes the related work in several aspects. After an overview in Section 3, Section 4 introduces the user interaction tools the system provides. The data pre-processing, which mainly involves scalar field topology extraction, is described in Section 5, and then detailed run-time algorithms are presented in Section 6. Conclusions are drawn in Section 8 before results and discussion in Section 7.

2 Related Work

In this section, we survey the recent developments in transfer function design, and briefly review the closely-related literature in the field of sketch-based user interface, and illustrative volume visualization.

Transfer Function Design has become one of the central topics in volume visualization research. The main effort in transfer function design research is to make better correspondences between the visible feature space, on which the user can directly operate, and the real volume characteristics. The two major categories in transfer function design are formed by image-based and data-centric methods [24].

Data-centric methods use the scalar value as well as the derivatives to classify the voxels and generate informative visualization results. The most straightforward mapping of a transfer function is a one dimensional lookup table to convert scalar values into colors and opacities [1]. In such design, all volume regions with the same density value will be colored the same and are visually indistinguishable. Adding more dimensions to the transfer function feature space can often improve the classification capability. For example, if the gradient magnitude of a scalar field is considered in 2D transfer functions [20], boundaries of the materials are highlighted. Higher order gradients are also used for multi-dimensional transfer function design [17, 19]. In addition, curvature [18], feature size [6], and ambient occlusion of samples [7] are proposed to enhance transfer function design. LH histograms [29] that define L and H values as the local extremes derived from every voxel, are proposed to highlight certain boundaries by specifying transfer functions in the LH histogram space. Instead of directly using the scalar value and the derivatives, topology information of the volume data can also guide transfer function design [11, 31]. The transfer function design can also be localized to each topology region [33]. Zhou and Takatsuka [36] further proposed a method to automatically specify localized transfer functions based on scalar field topology. In this work, both numerical properties and topology information are utilized in local transfer function specification, so that spatially disconnected features with similar properties can be distinguished.

On the other hand, image-based transfer function design is goal-oriented. A few image-based methods were developed early on in volume visualization. A user-guided visual optimization approach is proposed by He et al. [15], based on the rendering results of stochastically generated transfer functions. Design galleries [22] layout all available volume rendered images with randomly generated transfer functions for user selection and configuration. Transfer function design can also be defined as a sequence of 3D image processing routines, which aim at achieving a visualization goal, e.g. boundary enhancement, set by the users [10]. However, it is still difficult and unintuitive for users to control the results with image-based method interactively.

In order to make transfer function design more intuitive and practical for users, the community has contributed various techniques to overcome the problems. Semantic-based transfer functions [27] were proposed to bridge the gap between the parameter space and the semantic components. Semantic models are abstracted and defined based on the reference data. Illustrative visual styles can also be defined by semantics [25]. Besides by the explicit definition of semantics, transfer functions can also be specified by examples [21]. Another approach to simplify transfer function design exploits machine learning techniques. Tzeng et al. proposed a framework that converts stroke input into multi-dimensional transfer functions using a neural network [32]. More recently, several methods have been proposed to make transfer function design more straightforward by introducing direct operations on volume rendered images. Wu and Qu [34] proposed a technique to fuse features from multiple volume rendered results. Users can add or remove features by drawing strokes on the image. Semantics can be defined by users by brushing interaction on the rendered image, in order to assign parameters for visual mappings [12]. Stroke-based transfer function design [28] also allows users to extract and define layers in volume data by specifying foreground and background on

rendered images. In WYSIWYG volume visualization [13], users can directly manipulate the volume rendered images by painting on the results, and the system gives real-time image feedback to the users. In this work, we further enable users to operate on local features.

Sketch-based User Interface is intuitive and user-friendly, and it has been widely used in different tasks and applications. Designers can also stylize non-photorealistic effects on three-dimensional models by directly annotating strokes [16]. In volume graphics, sketch-based user interfaces are also helpful. Volume data can be interactively edited by interactive sketching [4]. In volume segmentation, the segmentation results can be edited and optimized by a sketch-based user interface on rendered images [35].

Illustrative Volume Visualization can further convey the details in volume data in an informative way [9]. Such effects can also be configured with transfer functions [26]. Two-level volume rendering [14] was provided to render different parts of a volume with corresponding different techniques. Svaknine et al. [30] proposed a goal-oriented user interface to create illustrations with different motifs. In VolumeShop [3], cutaway and ghosting illustrations can be interactively generated with a straightforward user interface. In our system, such illustration effects can be naturally achieved, using location specification by sketching strokes.

3 Overview

The main design goal of the system is to allow users to specify local transfer functions by flexible user interactions in rendered image space. In order to achieve this goal, there are two major problems to solve, including the design of local transfer function space and user interaction. One of the practical ways to add location information into transfer function is to apply pre-segmentation, which often relies on domain-specific algorithms and intensive human intervention. We notice that users often want to discriminate disconnected structures, which are often with similar properties. Thus, we can use contour topology analysis, which is a general and powerful tool to abstract the features and define transfer functions. To handle local transfer function design, in our WYSIWYG user interaction design, local features in the rendered images can be colorized, enhanced, removed, or brightened using the provided tool set by direct painting and sketching. Grouping and locking functions are available for convenient editing. Users can select to work on a local, group or global scope to control the locality of operations.

The pipeline of the system is shown in Figure 2. In a preprocessing stage, the abstract topology information of the raw volume data is extracted, and the volume data is partitioned into different topological regions. During run-time, volume rendered images are generated from the pre-processed data with localized transfer functions. The results of volume rendering can be interactively changed by a user through either painting with different tools or drawing strokes. The system incrementally converts the user input into local transfer functions by feature inference. Volume rendering results are updated instantly for feedback to the users.

4 User Interaction Design

In this section we describe the user interaction design in our system, including sketching and painting, operation scopes, and other interactive tools for changing visual effects. Operations, such as grouping and locking, are triggered by traditional hot keys, buttons or menus. Also, standard 3D visualization operations, such as rotating, panning, and zooming are supported.

4.1 Painting and Sketching

In the volume rendering view, users can switch between two interaction modes, painting and sketching. In these modes users can apply several visual effects and operations.

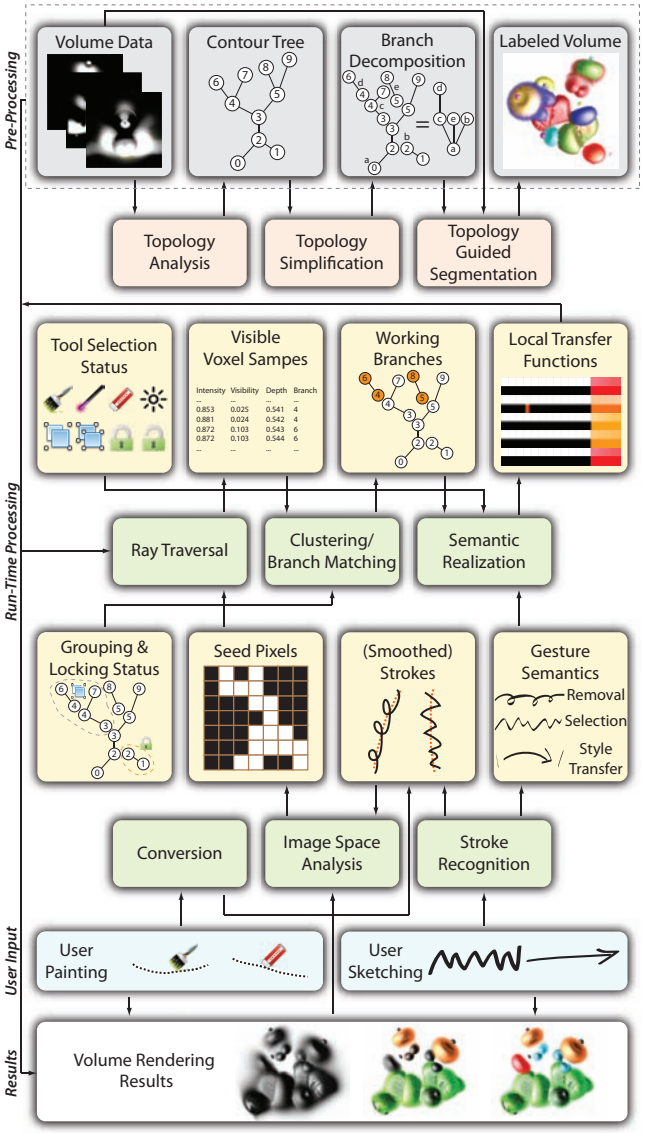


Figure 2: The system pipeline with both data pre-processing and run-time algorithms.

Painting updates the rendered results in real-time as a user moves a brush over the image. Users can stop painting at any time when the visual effects are satisfying. Figure 3 (a)-(c) shows an initially black blob being painted orange.

Sketching allows users to define single or multiple operations by gestures, a combination of strokes with semantics. After the last stroke completes, the system automatically recognizes the input gesture and applies the corresponding effects. For example, in Figure 3 (d)-(f), after placing an orange stroke on the black blob, the target volume is instantly rendered orange. With such gestures, both simple actions (e.g. colorization) and compound operations (e.g. style transfer) can be defined and applied.

4.2 Operation Scopes

Desired effects can be applied to different operation scopes. As shown in Figure 4, there are three operation scopes currently supported in the system: local, group and global. By default, if no groups are defined, user operations only take effect on the local scope, that is, other topological regions will not be affected. Oppositely, the appearance of all similar features will be modified if

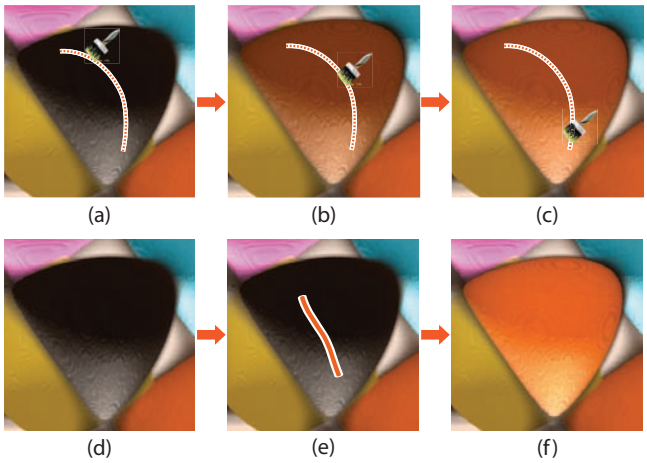


Figure 3: Interaction modes: (a)-(c) painting mode; (e)-(f) sketching mode.

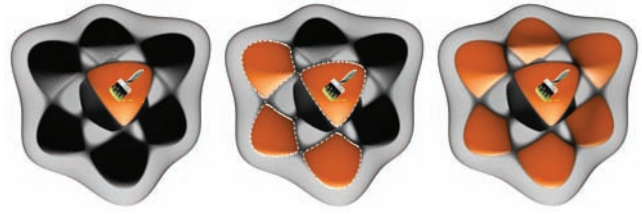


Figure 4: Three different operation scopes: painting scope is (a) local; (b) group; and (c) global.

global scope is selected, similarly as in WYSIWYG volume visualization [13]. In the group scope, the features in the defined group will be simultaneously operated on. The group scope is often handy to use when topological regions are fragmented.

Selecting, grouping and locking tools are provided to apply operations on several features simultaneously, users can select and group features. By default, the current selection will be cleared when new selections or empty selection are made. Users can also add or remove from the selection by pressing keyboard modifiers, similar to the operations in image editing software. When features are selected, the boundaries are highlighted with Photoshop-like dashed lines in the image. A locking function is also provided to prevent accidental modifications on already fine tuned features.

The system also provides drilling down and rolling up operations to enable effective volume data exploration. In the drilling down operation, the volume features that are not in the selection are hidden and locked. Users will only view and operate on the selected volume features and be able to focus on a local region (Figure 5). Drilling down and its inverse rolling up are triggered by sketching. These functions can be used to create focus+context illustrations.

4.3 Visual Effects and Interactive Tools

A full set of interactive tools is provided to modify the appearance of the visualization results, eventually enabling the users to discover new insights in the data set through interactive exploration. The provided interactive tools are categorized in different modes and listed in Table 1. The tools can also be categorized by their visual effects.

Appearance Changing The color, transparency, and silhouette parameters can be either directly or indirectly changed. Specifically, the color of the features can be edited by either the colorization painting tool or colorization strokes in sketching mode (Figure 1(b) and (c)). The transparency can be increased or decreased by applying the eraser painting tool. The pigtail stroke

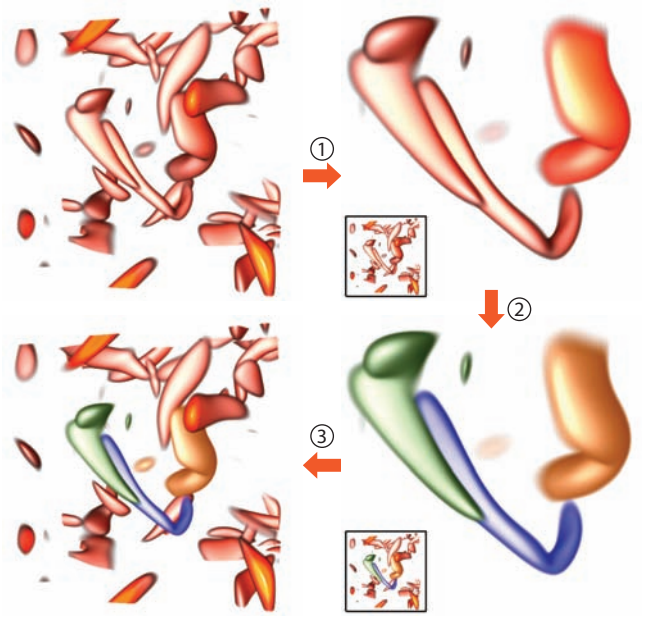


Figure 5: Drilling down and rolling up. In ①, a selected region is extracted for drilling down; in ②, the focused region is painted with a different color; in ③, the focused region is restored to the original view by rolling up.

	Tool/Gesture	Description
Painting	Eraser	Decrease/increase visibility
	Colorization	Change color
	Brightness	Increase/decrease brightness
	Silhouette	Add/remove silhouette
	Magic Wand	Create/extend/remove selection
Triggering	Group	Group the current selection
	Ungroup	Ungroup the current selection
	Lock	Lock the current selection
	Unlock	Unlock the current selection
Sketching	Colored Lines	Colorize the corresponding features with the line color
	Pigtail	Remove the corresponding features
	Drilling Down	Hide the regions that are not selected Focuse on the selection
	Rolling Up	Undo drilling down
	Arrow	Create ghost view
	Style Transfer	Transfer style from one region (group) to another
	Annotation	Write annotation

Changing Appearance *Selection/Grouping* *Utilities*

Table 1: The user interaction tool set of the proposed Local WYSIWYG Volume Visualization system.

((Figure 6(d)), a stroke which self-intersects several times, can fully remove corresponding features by making them fully transparent. The Brightness painting tool can make the features brighter

or darker. Users can also add silhouettes by painting along the corresponding silhouette boundary locations of the surfaces in the volume rendered images.

Style Transfer The style (appearance) of a certain region can be transferred to another region (or group) by the style transfer tool. Style transfer is triggered by an arrow gesture (Figure 1(d)). Two additional strokes are required to specify the source and the destination of the transfer. It is more straightforward to assign visual properties from existing examples than starting fresh with colorization or eraser tools.

Ghost View Users can directly generate a ghost view from the current selection by arrow gestures. The new ghost view can be rotated, panned, and zoomed independently. The current ghost view can also be recovered by the rolling up gesture. Figure 1(g) is an example of a ghost view, which clearly illustrates the detailed structures of ghosting components in a focus+context manner.

Annotation Annotations through sketching can be naturally added into a rendered image to make the illustrations more informative and insightful. There are two components in each annotation, namely a position and a string.

5 Data Pre-Processing

In the pre-processing process, contour topology information is used to measure and describe local features and to achieve local appearance specification. As shown in Figure 2, there are three major steps in the pre-processing phase. First, a contour tree is computed from the raw volume data. Second, a (simplified) branch decomposition is computed based on the contour tree. Third, the branch decomposition is used to segment the data in a labeled volume.

A *contour* of a continuous scalar field $f : \mathbb{M}^n \to \mathbb{R}$ on a continuous n-dimensional manifold $\mathbb{M}^n$ is defined as a connected component of a level set $L(h) = \{x | f(x) = h\}$, where h is an *iso-value*. While varying h from the global minimum f_{min} to the global maximum f_{max}, the contours join, split, or disappear at *critical points*, thus the topology of the isosurfaces changes as h varies. Since there are no holes in the simplex $\mathbb{M}^n$, the change of the contours can be tracked as a *contour tree* [2]. A contour tree is capable of representing the topological structures for scalar fields defined on a simplex. Due to the sensitivity to noise, contour trees are often too large to manage. Consequently, simplification methods are needed to make describing features more practical [5]. Branch decomposition [23] is an alternative way to present a contour tree. Each *branch* in branch decomposition represents a monotone path traversing a series of nodes in the contour tree.

In our implementation, we use branch decomposition instead of original contour trees for reasons. On one hand, branch decomposition provides a native multi-resolution representation of the contour tree, which is consequently straightforward to simplify. On the other hand, based on branch decomposition data structures, high-quality topology-controlled volume rendered image can be obtained [33].

In the pre-processing pipeline, a labeled volume is obtained by topology-guided segmentation [33]. The labels are the indices of the branches in the branch decomposition tree. Currently, the filtering threshold and the max number of branches for the tree simplification is fixed in run-time, due to the performance and storage limitation on the graphics card. The raw volume, together with the branch decomposition tree and the labeled volume, form the input for the run-time processing.

6 Run-Time Algorithm Details

In this section, we introduce the run-time algorithms of the proposed framework in detail. There are two major problems to be solved in the pipeline. First, the sketching and painting inputs need to be converted into the feature descriptions desired by the users.

Figure 6: Stroke types: (a) painting strokes (invisible); (b) ordinary strokes; (c) functional strokes; (d) hybrid strokes. In (d), the smoothed strokes (in dashed lines) are derived from hybrid strokes.

Second, the rendering parameters need to be changed accordingly, before the updated rendering results are fed back to the users.

Based on the pre-processed data, the local regions in the system are defined as the branches of the branch decomposition tree. As shown in Figure 2, the user input strokes are first converted into *(smoothed) strokes*, then the *seed pixels* for ray traversal are derived from the (smoothed) strokes. After ray traversal and data clustering, the *working branch set* **B** which defines the operation region can be obtained. There are also other branch sets in the system. The *lock set* **L** contains the branches whose appearance cannot be edited. The current selection is defined by the *selection set* **S**. In addition, groups $\mathbf{G}_i$ can be defined to synchronize the selection behavior and visual appearance changes.

6.1 Stroke Processing and Image Space Analysis

In this step, the local image patches are derived from the input strokes. In painting mode, each input point is also treated as a sample point on an invisible stroke. In sketching mode, there are three stroke types: ordinary, functional, and hybrid (Figure 6). Ordinary strokes indicate the desired positions and can be used directly as the seed for image space analysis. Functional strokes are recognized as gestures for the desired operation. For example, the arrow gestures indicate the style transfer operation. Hybrid strokes simultaneously define gesture and positional information. For positional information, the hybrid strokes are converted to ordinary strokes by straightening the lines before any further processing.

In the next step, the seed pixels in the image space are computed by propagating from the pixels covered by the smoothed strokes. The assumption here is that the seed pixels are similar to the pixels that are covered by the strokes [13]. Foreground and background patches are labeled by a two-pass Graph Cut algorithm. The foreground pixels, which are close to the pixels hit by the strokes, are the seeds for ray traversal and further analysis.

6.2 Ray Traversal and Feature Inference

In order to interactively change appearance and apply other operations in runtime, the system needs to know which branches to work on, namely the working branch set **B**. Besides, the numerical distribution of the features, indicated by the user strokes, should also be derived for further local transfer function modification. We collect multi-variate samples along the rays that start from the seed pixels in the image space. Several properties are collected at each sample point x during ray traversal, including scalar value $s(x)$, transparency, visibility $V(x)$, depth, and branch index b. The visibility $V(x)$ represents the visual significance at sample point x [8]. The working branch is determined by choosing the branch with the highest score, which is defined by the average visibility of a branch. If a branch belongs to the locked branch set **L**, it will not be selected. The working branch set **B** is further supplemented by adding other branches which are in the same group **G**, or all branches in the data set when in global scope.

The collected samples whose branch indices belong to the working branch set are further clustered to evaluate the numerical distributions. In order to give users real-time feedback, especially in

painting mode, light-weight and hardware accelerated clustering algorithms are used [13].

6.3 Semantic Realization

Semantic realization process takes effect on the rendered image according to the feature inference result. After determination of working branches and evaluation of numerical distributions, different operations can be applied by modifying the local transfer functions, updating the locking and grouping status, transferring styles, and creating and updating ghost views. Without loss of generality, a local transfer function $\mathbf{T}_b$ for branch b is defined as

$$\mathbf{T}_b = \sum_{s=1}^{n} \mathbf{c}_{b,s}\mathbf{f}_s = \mathbf{C}_b^T\mathbf{F}, \tag{1}$$

where the tuples $\mathbf{c}_{b,s}$ contain color and opacity values, and $\mathbf{f}_s$ is a component in feature space $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$. Compared to the global transfer function form [13], each branch in our system has a specific transfer function. In this framework, each component corresponds to a scalar value, and $\mathbf{c}_{b,s}$ is specified to the scalar value s. Thus, $\mathbf{C}_b^T$ is a 1D lookup table data structure.

6.3.1 Style Transfer

Style transfer copies the style of a source branch to a destination branch. Styles can cover many visual parameters, including colors, shadings, etc. The transfer of color is tricky, so we focus on transfer of color. Our implementation is based on the method proposed by Lu and Ebert [21], which maps the mean and standard deviation to each color channel. A statistical analysis is used to impose the color style of one branch to another branch or a group of branches. We define the color transfer as the color distribution transfer from the source branch to the destination branch. Assuming that the feature distribution and color distribution of the branches are simple, a Gaussian function is fit onto each opacity-weighted color channel.

6.3.2 Direct Visual Appearance Editing

Three basic visual parameters can be modified with the user interface, namely color, opacity, and silhouette intensity. Changing brightness is done by changing the color. In the painting mode, an incremental local transfer function $\Delta\mathbf{T}_b$ updates the current transfer function at each new stroke position and is defined for a branch b by

$$\Delta\mathbf{T}_b = \lambda\Delta\mathbf{C}_b^T\mathbf{F}. \tag{2}$$

where $\Delta\mathbf{C}_b^T$ is a color and opacity lookup table for the incremental change of the local transfer function for branch b, and λ is the time rate constant that controls the speed of change. At each new stroke sample, the new local transfer function $\mathbf{T}_b^{'}$ is a combination of the previous local transfer function $\mathbf{T}_b$ and the incremental local transfer function:

$$\mathbf{T}_b^{'} = \mathbf{T}_b \oplus \Delta\mathbf{T}_b, \tag{3}$$

where operator $\oplus$ blends two colors in CIE $L^*a^*b^*$ color space. In the sketching mode, an operation is applied only at the end of a stroke. We regard such operation as a style transfer from a solid color to the working branch set $\mathbf{B}$.

6.3.3 Selection Making

Selection making operation makes the current working branch set $\mathbf{B}$ equal to the selection set $\mathbf{S}$. The working branch set is decided by ray traversal and feature inference when users paint on the image. During the working branch selection, branches in the lock set $\mathbf{L}$ are also allowed to be selected. Groups can be further made by copying the current selection set $\mathbf{S}$ to a new group $\mathbf{G}'$. Branches can also be locked in order to prevent further appearance changes.

Dataset	Size	N_b	T_{pre}	T_r	T_i
Hand	$244 \times 124 \times 257$	128	16.8	0.134	0.307
Chest	$384^2 \times 240$	128	64.9	0.148	0.170
Vorticity	128^3	128	5.8	0.094	0.117
Combustion	$480 \times 720 \times 120$	160	112.9	0.271	0.309

Table 2: The timings of the system. N_b is the number of branches; T_{pre}, T_r, and T_i are the average timings of pre-processing, rendering, and feature inference in seconds.

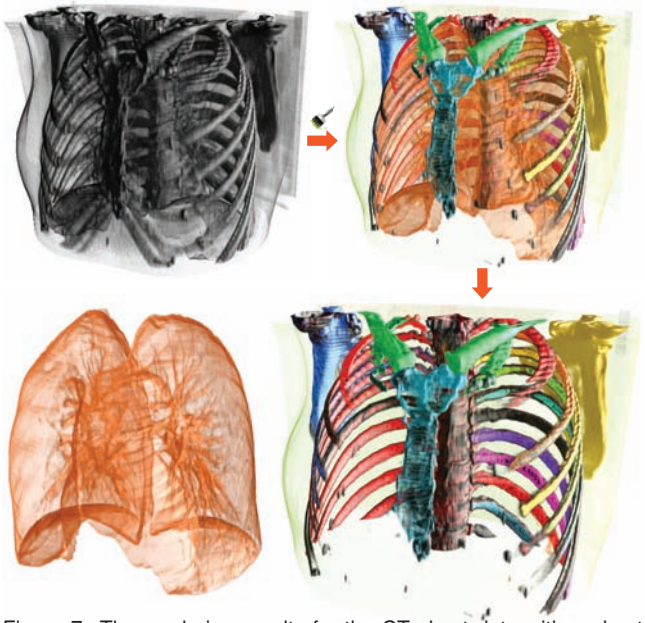


Figure 7: The rendering results for the CT chest data with a ghost view of the lungs. A global eraser operation is applied first to remove the outer part. Then the colors of different structures in the volume are changed with a colorization tool. To reveal detailed features of the lungs, a ghost view is created in the last stage.

7 Results and Discussion

We have tested the proposed system on several data sets of different types, and invited domain experts to use the system and give feedback.

7.1 Implementation and Performance

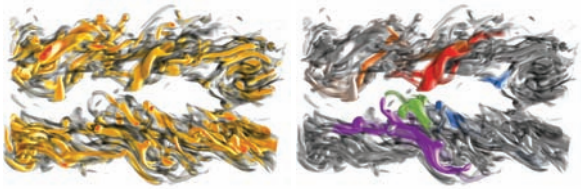
The system is implemented in C++ and uses OpenGL and CUDA. Volume data with topology information and the transfer functions are transferred to the video memory, and the resulting images are returned and rendered on an OpenGL buffer. The local transfer functions are stored as a 2D lookup table in the video memory. When transfer functions for several branches are modified, the corresponding rows of the lookup table are updated respectively.

We performed a timing test (Table 2) on a Dell T3500 workstation with an Intel Xeon W3503 2.40GHz CPU, 8GB RAM, and an NVidia GTX560 graphics card with 1GB video memory. The rendered image resolution is fixed at 800×600, and the step size of raycasting is 0.5 voxel. Full shading is applied during the rendering. From the table, we can see that the system is interactive during the run-time interaction, although the pre-processing takes relatively a long time. The rendering performance is also related to different rendering states and viewpoints.

7.2 Case Studies

We apply the proposed tool on several datasets as listed in Table 2. The hand data and the chest data are from CT scans, and the combustion data is from a simulation.

Figure 1 demonstrates the interactive operation sequence on the CT scan of a human hand. If the data set is visualized with tradi-



(a) (b)

Figure 8: The rendering results for the combustion dataset at timestep 66 with a global transfer function (a) and local transfer functions (b). The structures in the data field are clearly highlighted with local transfer functions.

tional global transfer functions, it is difficult to specify visual parameters at such granularity. For example, the density distribution of the vessels overlaps with that of the bone. In previous work, by using a segmented volume [14], the classification becomes clear, but their data segmentation often requires intensive manual intervention. In our framework, the raw data set is partitioned with contour topology information, which is lightweight and often reasonable. The visual appearance of each topological zone can be modified in multiple flexible ways.

In Figure 7, a CT chest data is shown. Compared to the vertical transfer function editor for unique topological zones [33], the manual mapping between the branches and the corresponding regions is avoided. For example, when users would like to change the color of a specific rib, they would have to find out which branch corresponds to this rib first, before the local transfer function could be modified. In our framework, users do not need to access these visualization parameters directly, but can directly start editing.

To present how the proposed method performs on simulation data, we apply the system on timestep 54 of the vorticity data (Figure 5) and the combustion data (Figure 8). Traditionally, global transfer functions are favored for simulation data, because the spatial numerical distribution is important. However, global transfer functions may lead to severe visual interference between nearby features. Our method provides a solution to highlight the structural insights of the data set, and also may conveniently generate volume visualization in illustration style.

We invited experts from medical and fluid dynamic backgrounds to try our system and give feedback. The user from medical research rated the system as excellent after trial use. Without any prior experiences in volume visualization, he can use the system to explore medical volume data without difficulty after a very brief introduction on the system. It is straightforward to identify important features, and generate an illustration in high quality. Compared with tuning parameter lines in the tradition of transfer function design methods he also tried, he gave strong preference on our way. The user from fluid dynamic research also gave positive comments. With localized transfer function and intuitive tools, interesting structures can be revealed and illustrated effectively. From user perspectives, they can set different appearances for spatially disconnected components, so they do not even have to know contour topology and the branches in the data explicitly.

7.3 Discussions

From the above cases, we can see that the proposed method can be effective to generate insightful volume visualizations by emphasizing local features. Since contour topology is general for volume data analysis, our tool is applicable to a variety of data, not limited to CT scan and turbulence data. Traditional transfer functions, however, which are defined in the global domain, are not likely to be able to adjust local appearances in most practical data sets. On the other hand, even if localized transfer functions would be used, the manual specification of the huge parameter space remains hard for users to handle.

	Interactivity	TF Design	Localized	Editing	Selection	Illustration	Painting	Sketching
Our Work	+	+	+	(+)	+	+	+	+
WYSIWYG [13]	+	+	-	-	-	-	+	-
VolumeShop [3]	+	-	+	-	+	+	+	-
Direct Vol. Editing [4]	+	-	+	+	-	+	+	-
Stroke-Based TF [28]	+	+	-	-	+	-	-	+
DVRI Editing [34]	-	+	-	-	-	-	-	+

Table 3: Comparison to previous work which supports 2D interaction on volume rendered images. The plus means that the method is capable of applying a certain kind of functions, while minus means that it does not apply. Plus in brackets indicates that the corresponding technique has the potential to support the certain function.

We compare our tool to existing work that supports 2D interaction on rendered images (Table 3). Our tool is the first to combine the transfer function design and localized operations in a single framework. Illustration effects are readily incorporated. Potentially, the volume editing functions could be added into the system for finer granularity tuning.

Compared with the previous WYSIWYG volume visualization [13] work, there are multiple benefits introduced by local transfer function design. From the classification aspect, local transfer function is much better than global mapping. Disconnected features in with similar numerical properties can now distinguished by their locations, so the global transfer function limitation in the previous work is eliminated by introducing the local transfer functions defined on topology regions. Take the hand data for example, the vessels are in the same region in the 1D feature space, so it is not possible to change the appearance of specific vessels, without influencing other ones. From user perspectives, our tool provides an implicit and intuitive interface to sophisticated local transfer functions. Without extensive practice, users can handle the tool and create insightful results. In WYSIWYG design, the localized operations are also important. The user painting region can exclusively take effect on the local features, instead of all similar features in the global scope. Thus, local operations make user interaction less confusing, and more flexible. In addition, we also introduce sketch- and gesture-based user interaction, which makes the system full-functioned and self-consistent, especially for local operations. Illustrative visualization can also be created with components like ghost views and annotations.

There are several limitations in our work, and some factors may influence the user experience and the exploration process. Since the locality is based on contour topology, restrictions apply depending on the circumstances. First, contour trees are quite vulnerable to noises, so data sets with lower noise may perform better. For example, the contour tree of the fMRI data is very hard to describe clear features due to noise, so our method may not be as good as expected. Second, the number of branches, which controls the number of local topological regions, may affect the granularity of the operations. Currently, we only support static number of branches in run-time due to storage and performance issues. If the number of branches is too low, local features may be submerged into other components. Otherwise, users need to combine small features into meaningful components during the interaction if the number is too large (the theoretically complexity is $O(n)$). Dynamic branching may be supported in future implementation with rational algorithms and user interaction design. Third, the initial parameters e.g. transfer function and camera configuration may also influence the exploration process.

8 Conclusions and Future Work

In this work, we present a volume visualization system called Local WYSIWYG Volume Visualization, which is capable to assign

local transfer functions through direct sketching or painting on the volume rendered image. Our user interaction design serves two purposes. It not only provides the user with an intuitive WYSIWYG style interaction to specify target volume objects for assigning colors and opacities, but also provides spatial locality to achieve local specification of transfer functions. Such design lowers the threshold for using transfer function in volume data exploration.

In the future, several improvements may enhance the system. Without too much additional effort, our system can support volume editing. A formal user study will be conducted to further validate the effectiveness of the system. We are also going to apply our method to datasets from different domains. The proposed method can also be generalized to time-varying volume data, by extracting the topological structures over time evolution of the data. Currently the system only works on scalar field data. We would like to extend our interaction metaphor to support vector fields or multi-variate volume data.

ACKNOWLEDGEMENTS

This work is supported by 863 Program Project (No. 2010AA012402), National Natural Science Foundation of China Project (No. 60903062 and 61170204). This work is also partially supported by National NSFC Key Project No. 61232012 and the Strategic Priority Research Program - Climate Change: Carbon Budget and Relevant Issues of the Chinese Academy of Sciences Grant No. XDA05040205.

REFERENCES

[1] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *Proc. IEEE Visualization 1997*, pages 167–174, 1997.

[2] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proc. IEEE 1963 Fall Joint Computer Conference*, pages 445–458, 1963.

[3] S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *Proc. IEEE Visualization 2005*, pages 671–678, 2005.

[4] K. Bürger, J. Krüger, and R. Westermann. Direct volume editing. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1388–1395, 2008.

[5] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. IEEE Visualization 2004*, pages 497–504, 2004.

[6] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1380–1387, 2008.

[7] C. Correa and K.-L. Ma. The occlusion spectrum for volume classification and visualization. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1465–1472, 2009.

[8] C. Correa and K.-L. Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Trans. Vis. Comput. Graph.*, 17(2):192–204, 2011.

[9] D. Ebert and P. Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *Proc. IEEE Visualization 2000*, pages 195–202, 2000.

[10] S. Fang, T. Biddlecome, and M. Tuceryan. Image-based transfer function design for data exploration in volume visualization. In *Proc. IEEE Visualization 1998*, pages 319–326, 1998.

[11] I. Fujishiro, T. Azuma, and Y. Takeshimat. Automating transfer function design for comprehensible volume rendering based on 3D field topology analysis. In *Proc. IEEE Visualization 1999*, pages 467–470, 1999.

[12] M. Gerl, P. Rautek, T. Isenberg, and E. Gröller. Semantics by analogy for illustrative volume visualization. *Computers & Graphics*, 36(3):201–213, 2012.

[13] H. Guo, N. Mao, and X. Yuan. WYSIWYG (What You See Is What You Get) volume visualization. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2106–2114, 2011.

[14] H. Hauser, L. Mroz, G. I. Bischi, and E. Gröller. Two-level volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 7(3):242–252, 2001.

[15] T. He, L. Hong, A. E. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *Proc. IEEE Visualization 1996*, pages 227–234, 1996.

[16] R. D. Kalnins, L. Markosian, B. J. Meier, M. A. Kowalski, J. C. Lee, P. L. Davidson, M. Webb, J. F. Hughes, and A. Finkelstein. WYSIWYG NPR: drawing strokes directly on 3D models. In *Proc. ACM SIGGRAPH 2002*, pages 755–762, 2002.

[17] G. L. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *VVS '98: Proc. IEEE Symposium on Volume visualization*, pages 79–86, 1998.

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

[19] J. Kniss, G. L. Kindlmann, and C. D. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 8(3):270–285, 2002.

[20] M. Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.

[21] A. Lu and D. S. Ebert. Example-based volume illustrations. In *Proc. IEEE Visualization 2005*, pages 655–662, 2005.

[22] J. Marks, B. Andalman, P. A. Beardsley, W. T. Freeman, S. Gibson, J. K. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. M. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proc. ACM SIGGRAPH 1997*, pages 389–400, 1997.

[23] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *VIIP 2004: Proc. IASTED conference on Visualization, Imaging, and Image Processing*, pages 452–290, 2004.

[24] H. Pfister, W. Lorensen, C. Bajaj, G. L. Kindlmann, W. J. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Comput. Graph. Appl.*, 21(3):16–22, 2001.

[25] P. Rautek, S. Bruckner, and E. Gröller. Semantic layers for illustrative volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1336–1343, 2007.

[26] P. Rautek, S. Bruckner, and E. Gröller. Interaction-dependent semantics for illustrative volume rendering. *Comput. Graph. Forum*, 27(3):847–854, 2008.

[27] C. Rezk-Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Trans. Vis. Comput. Graph.*, 12(5):1021–1028, 2006.

[28] T. Ropinski, J.-S. Praßni, F. Steinicke, and K. H. Hinrichs. Stroke-based transfer function design. In *Proc. of IEEE/EG International Symposium on Volume and Point-Based Graphics*, pages 41–48, 2008.

[29] P. Sereda, A. V. Bartrolí, I. Serlie, and F. A. Gerritsen. Visualization of boundaries in volumetric data sets using LH histograms. *IEEE Trans. Vis. Comput. Graph.*, 12(2):208–218, 2006.

[30] N. A. Svakhine, D. S. Ebert, and D. Stredney. Illustration motifs for effective medical volume illustration. *IEEE Comput. Graph. Appl.*, 25(3):31–39, 2005.

[31] Y. Takeshima, S. Takahashi, I. Fujishiro, and G. M. Nielson. Introducing topological attributes for objective-based visualization of simulated datasets. In *VG05: Proc. IEEE/EG International Symposium on Volume Graphics*, pages 137–145, 2005.

[32] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Trans. Vis. Comput. Graph.*, 11(3):273–284, 2005.

[33] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 13(2):330–341, 2007.

[34] Y. Wu and H. Qu. Interactive transfer function design based on editing direct volume rendered images. *IEEE Trans. Vis. Comput. Graph.*, 13(5):1027–1040, 2007.

[35] X. Yuan, N. Zhang, M. X. Nguyen, and B. Chen. Volume cutout. *The Visual Computer*, 21(8-10):745–754, 2005.

[36] J. Zhou and M. Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1481–1488, 2009.