WYSIWYG (What You See is What You Get) Volume Visualization

Hanqi Guo, Ningyu Mao, and Xiaoru Yuan, *Member, IEEE*

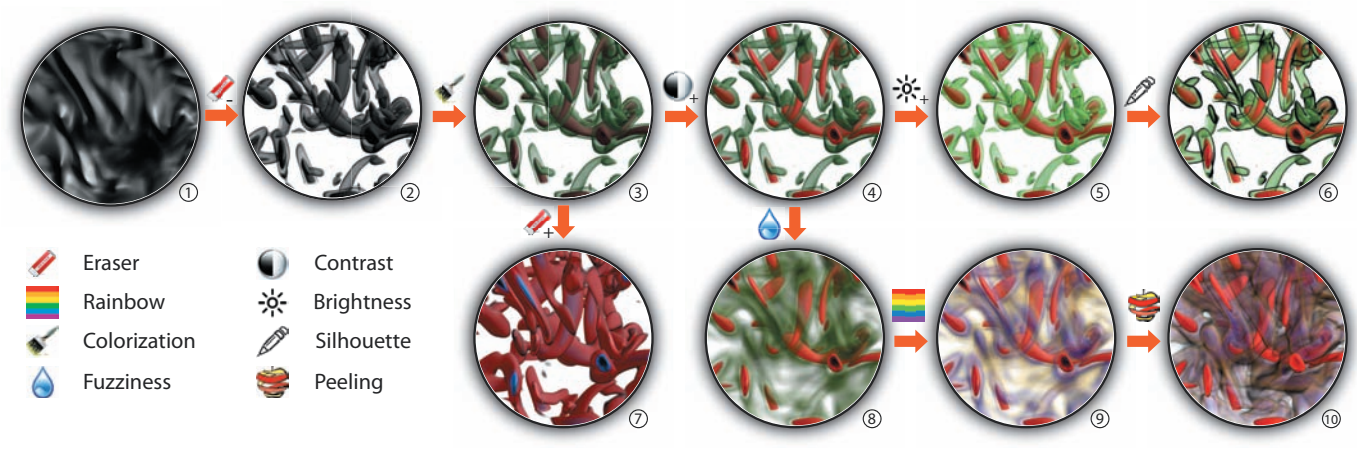

Fig. 1. One volumetric flow data set interacts with our WYSIWYG volume visualization system. Starting from an initial rendering of the raw volume on the left, the user selects corresponding tools and applies strokes on top of the rendered image: ①→②, remove the outer and intermediate parts of the volume with the volume eraser tool; ②→③, paint the volume with the colorization tool; ③→④, enhance the contrast of the rendered volume; ④→⑤, increase the brightness of the top part of the volume; ⑤→⑥, add silhouettes to the rendered volume; ③→⑦, increase the opacity of the outer part of the volume; ④→⑧, increase the fuzziness of the outer features; ⑧→⑨, colorize fuzzy matter with the rainbow tool; ⑨→⑩, recover the previously removed volume materials with the peeling tool.

Abstract—In this paper, we propose a volume visualization system that accepts direct manipulation through a sketch-based What You See Is What You Get (WYSIWYG) approach. Similar to the operations in painting applications for 2D images, in our system, a full set of tools have been developed to enable direct volume rendering manipulation of color, transparency, contrast, brightness, and other optical properties by brushing a few strokes on top of the rendered volume image. To be able to smartly identify the targeted features of the volume, our system matches the sparse sketching input with the clustered features both in image space and volume space. To achieve interactivity, both special algorithms to accelerate the input identification and feature matching have been developed and implemented in our system. Without resorting to tuning transfer function parameters, our proposed system accepts sparse stroke inputs and provides users with intuitive, flexible and effective interaction during volume data exploration and visualization.

Index Terms—Volume rendering, Sketching input, Human-computer interaction, Transfer functions, Feature space.

1 Introduction

Direct volume rendering has been widely applied in various domains, from medical diagnoses to atmospheric simulations, as one of the most effective ways to visualize volumetric data. However, the difficulties of setting proper mapping functions to convert original volumetric data to renderable color and opacity values limited the application of volume rendering. In the process of volume rendering, transfer functions [23] are designed to map the raw volume data values to colors and transparencies, before composite images are constructed. Obtaining appropriate transfer functions is critical to comprehend and analyze the complicated structures of volume data by reducing occlusions on undesired features and highlighting important objects in the volume.

The main obstacle related to transfer function design comes from the non-intuitive interface in the current design practice. The most common transfer function design interface is to directly assign opacity and color value to each volume density value by adjusting curves, which define the mapping. However, It is non-trivial for non-expert users to generate informative and clear visualization results with such a design paradigm. Users have to understand how different density values are mapped to the rendered image features, and establish these connections in their brain during the transfer function design. For example, in medical CT data visualization, if users want to hide the skins and reveal the structure of the bones, they have to first recognize that skin has lower intensity values, and then reduce the opacities of the corresponding values in the feature space. Sometimes, small changes in the transfer function may result in dramatic differences in rendering appearance. Due to the perspective projection and compositing effects, there is no one-to-one mapping between the color effects in the transfer function curves and the final rendered appearance. This steep learning curve makes volume rendering not easily accessible to layman users. Even with the latest development in volume visualization research, with which users can modify the occlusion or other color properties of certain features, the detachment between the image space (the target image of volume rendering) and parameter space (the parameters that users can operate and modify) makes transfer function specification still cumbersome.

In this paper, we propose a WYSIWYG (What You See Is What You Get) volume visualization system that accepts direct sketch and

- *Hanqi Guo, Ningyu Mao, and Xiaoru Yuan are with Key Laboratory of Machine Perception (Ministry of Education), and School of EECS, Peking University, E-mail: {hanqi.guo,ningyu.mao,xiaoru.yuan}@pku.edu.cn.*
- *Hanqi Guo and Xiaoru Yuan are also with Center for Computational Science and Engineering, Peking University.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

paint manipulations from the user and enables goal-oriented operation on volume exploration and visualization. With the proposed system, users can directly change the appearances of the volume rendered images with desirable visual effects through sketching and painting. Painting different colors, changing opacities, erasing volume structures, modifying contrast and brightness, adding silhouettes, peeling volume feature layers and other operations can be directly applied to the volume rendered images by different painting tools. Real-time animated visual feedback enables the user to adjust the input interactively to achieve satisfying results. Without the need for switching back and forth between the parameter space (transfer functions) and the image space, an intuitive, goal oriented volume exploration and transfer function design paradigm has been established in our system. The system lets the users focus on the spatial and finer structures of the volume and make desirable modifications without interrupting the observation process or compromising the effectiveness of volume comprehension.

The WYSIWYG concept, which is very intuitive and effective for various tasks, has been widely applied in much 2D image editing software like Adobe Photoshop. Instant and real-time results are fed back to users during the painting process in our tool, just like the tools in Photoshop. In addition to image editing tools, there have been a few attempts at direct operation on volume data. For example, the volume catcher [27] and volume cutout [38] enable efficient volume segmentation by applying strokes to the volume rendered images. Multiple features in distinct rendered images can be fused into a comprehensive one by adding or deleting features with strokes [37]. More recently, Ropinski et al. [33] proposed the stroke-based transfer function. Features are extracted and selected based on the user defined strokes indicating foreground and background, and then classified as layers of transfer functions, which can be further composed to new results by tuning opacities of each layer. In our work, we make volume visualization and exploration more flexible by introducing several simple sketching and painting primitives on the rendered image space. Unlike the operations in Photoshop, our tool takes effects on the global visualization results based on transfer functions, instead of localized operations. Compared to the prior works on volume visualization, our method only relies on the current rendered result and the user input, without explicit specifications of foreground and background, feature layers, or pre-prepared images with specific features. The real-time feedback also allows users to suspend operations when the rendered image is satisfying.

The contribution of this paper is two-fold. First, we present a novel WYSIWYG volume visualization system, which accepts user sketch input with a full function set and gives real-time feedback. Second, a series of algorithms, including fast user intention inference and semantic realization are proposed to facilitate the interactive data exploration.

The remainder of this paper is organized as follows. We summarize the related work in Section 2, and then give an overview of the system in Section 3. The user interactions, algorithms and implementation details are described in Section 4 and Section 5. We briefly present the implementation and performance in Section 6. Results are shown in Section 7 and the limitation of the method are discussed in Section 8, before the conclusions are drawn in Section 9.

2 Related Work

Transfer function design is the central topic in direct volume rendering. Generally speaking, transfer functions can be categorized into two major types, namely image-based and data-centric methods [29]. Image-based transfer methods, which are goal-oriented, utilize images as the input. For example, transfer functions can be generated using genetic algorithms, which are guided and optimized by user selection on the rendered images in the thumbnails [16]. The rendered images with different transfer functions can also be organized and presented in Design Galleries for user selection [26]. Transfer functions can also be defined as a sequence of 3D image processing procedures, which allows users to achieve optimal visualization results [12]. Our method also utilizes the rendered image space for feature inference, in order to give image feedback to users with new results interactively.

Unlike image-based methods, data-centric transfer function design focuses more on the volume data itself and its derivative properties. For example, 1D transfer functions, map the scalar sample values into optical properties, thus classifying the volume data for direct volume rendering [2]. Better classification results can be obtained with multi-dimensional transfer functions by utilizing derivative properties of the volume data, e.g. first and second order gradient magnitudes [23, 19], which aid in displaying surface information. Usually, features that are not distinguishable in 1D feature space can be better shown by leveraging proper multi-dimensional transfer functions. In order to obtain better feature separation results with multi-dimensional transfer functions, many other derivative properties have been taken into consideration. For example, Kindlmann et al. [20] take advantage of curvatures to assist the transfer function design. LH histogram [34] makes it easier to find boundaries, by accounting for local extremas that are derived from every voxels. Local statistical properties are also used to find sophisticated features [28]. The approximated relative size [5] of the objects in volume data helps in finding features with certain sizes. The occlusion spectrum [6] reveals the occlusion relationships among spatial patterns. Visibility-driven transfer functions [7] provide a feedback mechanism that highlights the visibility of features from a given viewpoint. In our work, we take advantage of derivative properties, e.g. the visibility [7] for data analysis.

Although the community has contributed a variety of methods for transfer function design, it is still a cumbersome and non-trivial work for untrained non-expert domain users to design transfer functions for volume visualization, because transfer function design is very unintuitive. Several methods have been proposed to make transfer functions more intuitive. For example, users can change the appearance of the visualization results by semantics based transfer functions [32, 30].

It is noticeable that sketch-based user interfaces are widely used in various applications, from image editing to 3D modeling, and from NPR effect design to volume segmentation, as one of the most methods for the tasks, because it is very intuitive for users to achieve desired results. For example, 2D gray images can be colorized by painting a few colored strokes on the image [22]. A sketch-based user interface also help users during 3D modeling. The models can be interactively designed by drawing silhouettes of objects on the 2D canvas [17]. With WYSIWYG NPR [18], designers can directly annotate strokes on a 3D model with strokes to stylize non-photorealistic effects for objects. In addition to image and model design, it is also quite effective to utilize a sketch-based user interface for volumetric data manipulation. Direct volume editing [4] allows users to edit the volume data by interactive sketching. Users can also perform volume segmentation by sketching on the image [38, 27].

A few frameworks have been proposed for facilitating transfer function design by leveraging sketch-based interfaces. For example, users can sketch the volume slices to assign color and transparency [36, 35], then artificial neural network based algorithms define the high-dimensional transfer function for rendering. Later, Wu and Qu [37] proposed a method for users to sketch on the rendered images to fuse multiple features in distinct rendering results into a comprehensive one, by adding or deleting features with strokes. Guo et al. [14] utilized sketch-based input to facilitate multi-dimensional transfer function design. Sketch queries are accepted by the system and then the corresponding component of the transfer function is highlighted in the design interface. Ropinski et al. [33] proposed a stroke-based user interface for 1D transfer function design. Layers are extracted according to the inputted strokes, which defines foreground and background, then the transfer functions are further refined by adjusting the colors and transparencies of each layer. In our work, real-time feedback is given to users, which is flexible enough for users to stop at any intermediate result when satisfied. Compared to the work by Wu and Qu [37], our work does not rely on pre-defined images with different features, but only depend on the current rendered image and user input. Compared to the work by Ropinski et al. [33], users do not need to explicitly define either foreground and background, nor the layers of volume data.

It is a difficult problem to map image space features to the objects in

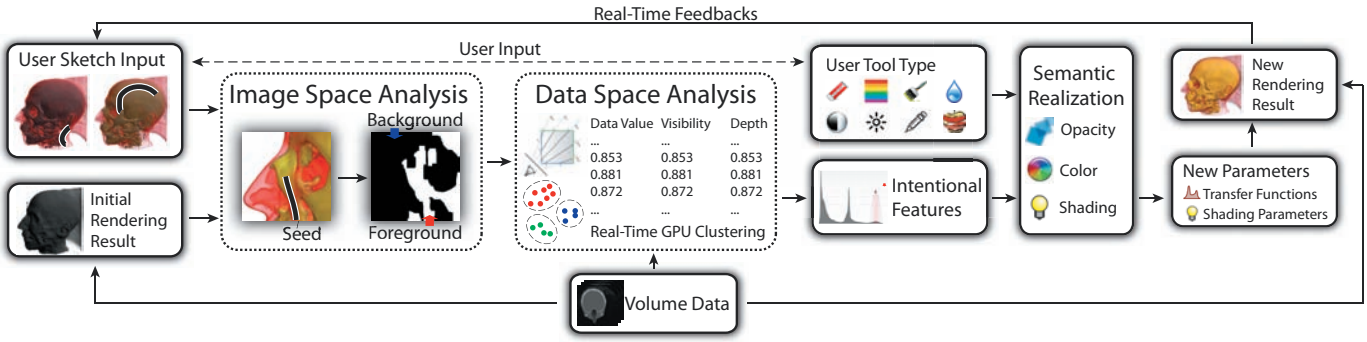

Fig. 2. The pipeline of the system. From user sketch input, the intentional features are derived by image space analysis and data space analysis. The new rendering parameters are determined according to the user tool type, before the new rendering result is fed back to users in real time.

volume data, which is the key for any sketch-based volume exploration tool. In general, sketch-based user interaction systems infer user intentions according to sketch inputs, and then take effect on the results. Kohlmann et al. [21] proposed a solution, which analyzes the ray profile and finds the best match from the contextual meta information in the knowledge database. In Volume Cutout [38], the objects of interest are located by the optimization with stroke and over-segmented volume segmentation. Malik et al. [25] proposed a framework of ray profile analysis, which locates the transition points by slope thresholds. Ropinski et al. [33] identify desired features by comparing the ray profiles of both foreground and background stroke specifications, and then decide the intentional depths by histogram analysis.

In addition to transfer function design, other techniques such as illustrative visualization methods [10] can further enhance effects and convey more details informatively by non-photorealistic rendering techniques. For example, contours and silhouettes can visualize the boundaries in volumes effectively [9]. Transfer functions for rendering styles can also be utilized in illustrative volume rendering [31]. In VolumeShop [3], a dynamic volume illustration environment is introduced for interactive visualization. In this work, we also show examples of how to define styles by sketching in volume rendering.

3 System Overview

The main objective of our system is to supply the user with intuitive and goal-oriented interaction on volume data through a WYSIWYG user interface. Users can select various semantics to change the visual effects of the visualization result, by indicating the desired features with direct sketch. The pipeline of the system (Fig. 2) starts from the initial rendering result and the user sketch input, and gives real-time image feedback, which allow users to iteratively tune the rendered image using various tools. Several intermediate steps are key to the whole pipeline. User inputs are first converted to foreground and background patches by image space analysis, and then the corresponding feature clusters (intentional features) in the volume data are derived according to image analysis results via data space analysis. The new transfer functions are generated according to the intentional features and the current tool type, before the rendered image is updated.

Various tools are developed based on the WYSIWYG concept, which allows the user to directly change visual appearances and explore the features in the volume data by sketching, without accounting for the parameter space. Specifically, color, transparency, as well as complicated effects like fuzziness, silhouette, and contrast can be adjusted and refined by direct sketch with the corresponding tool. Data navigation with the peeling tool further allows the user to explore the features in the volume rendered images without explicit layer definition.

4 User Interaction

Similar to operations in popular image editing software, we provide a full set of tools for direct manipulation on the direct volume rendered image. Real-time results are fed back to users while brushing

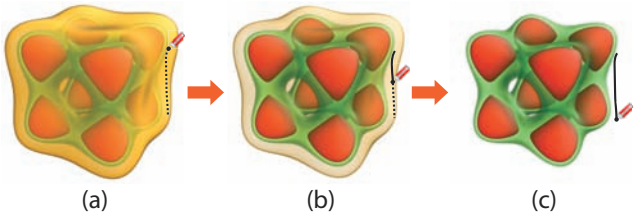

Fig. 3. The animation sequence of the eraser operation. The mouse is pressed in (a), and then it is moved along the dashed line in (b) and (c). The skin gradually disappears as the mouse moves.

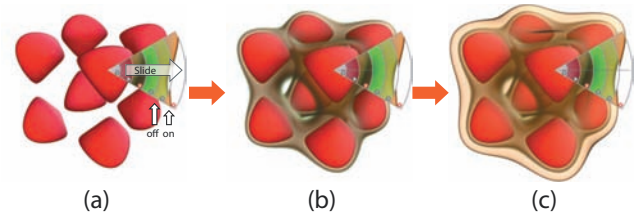

Fig. 4. The peeling slider for layer exploration. After clicking on a seed point on the image, users can explore the automatically extracted outer features by sliding on the widget. The appearance of different feature layers are shown in each slice of the pie.

on the image with various tools. For example, the visual objects will be gradually erased when brushing on the image with the eraser tool (Fig. 3). There are eight types of visual effects for the direct changing of appearances in our tool set, which are introduced here:

Colorization This tool gradually changes the color of the selected feature.

Eraser Users can increase or decrease the visibility of desired objects in the rendered image.

Boundary Fuzziness This operation allows users to make the desired boundary fuzzy.

Silhouette Boundaries are enhanced with a silhouette effect. Silhouette provides exaggerated boundary information using NPR (non-photorealistic rendering) styles.

Brightness The luminance of features can be directly changed without explicitly changing colors.

Structure Contrast The contrast of the structures that are sketched upon is automatically enhanced by indirectly changing the color saturation and lighting configurations.

Rainbow This tool forces the maximum color differences for the features that are sketched on. The features are colorized as gradient colors, which are either preset or user-defined.

Layer Exploration In addition to appearance changing, a peeling slider (Fig. 4), which is a pie-like widget, contains several sectors rep-

resenting the appearance of different automatically extracted layers. Layers can be toggled by clicking a small red button. During the interaction, the corresponding layers will be shown in the main image.

5 Algorithms

The algorithms of the proposed system consist of two major procedures, the feature inference and the semantic realization. The feature inference process converts the image space input into the corresponding distribution functions that are defined on the parameter space, and then the semantic realization process takes effect on the rendering result by changing the rendering parameters according to feature inference results in real-time.

5.1 Fundamentals

In this section, we briefly define and formularize several important concepts in volume visualization for convenience. Without loss of generality, we assume that the appearance of the volume rendered image is only related to transfer functions, giving identical camera configurations. The shading effects of different features can also be defined by similar visual mappings.

Feature space is generally defined as a m-dimensional parameter space, whose dimensionality m is determined by the number of properties of the volume data. For example, a 1D feature space can be defined in the range of the intensity value of the CT scan. The feature space can also be multidimensional when accounting for derivative dimensions e.g. gradient magnitudes.

Referring to the work by Rezk-Salama et al. [32], transfer functions can be generalized as a set of parameters based on the primitives in the feature space, regardless of the representation of transfer functions. For example, the simplest primitive for a 1D transfer function is a 1D lookup table. Denote the primitives of the feature space as $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n\}^T$, then the transfer functions can be defined upon the primitives:

$$TF = \sum_{i=1}^{n} \mathbf{c}_i \mathbf{f}_i = \mathbf{C}^T \mathbf{F}, \tag{1}$$

where $\mathbf{c}_i$ is the tuple of the color and opacity values, and where $\mathbf{C}^T$ is a $n \times 4$ matrix, which consists of $\mathbf{c}_i$. We utilize the CIE $L^*a^*b^*$ as the color space plus alpha channel for convenience, although the actual storage of color is in RGBA space. Except as otherwise noted, transfer functions in following sections are defined in 1D, and they are commonly stored as a lookup table in memory. Approaches for multidimensional and more complicated transfer functions can also be developed by reasonable extensions to the above definitions.

5.2 Feature Inference

Feature inference, which aims to determine the features to act on according to the user input, is the key of the system (Fig. 5). This problem is very different from picking up opaque objects in common scene rendering, where picking positions can be determined by referencing the depth buffer. There are multiple problems that present themselves when attempting to map from the image space objects to the feature space distribution in volume rendering. First, volume rendered images often contain visual features that are fuzzy, semi-transparent and multi-layered. In such an environment, it is hard to determine the depths and positions of the corresponding objects. Secondly, even if the intended positions in the 3D space can be determined, mapping such information to the feature space is still a problem. The third problem is ambiguity. We need to exclude the obvious interferences although users tend to avoid ambiguity when operating on the images.

To solve these problems, we propose a method for inferring the user intended features according to the sketches on the volume rendered image. The algorithm consists of two steps (Fig. 2): first, finding out the most likely region in the image space; second, determining the distribution of the feature in the data space and the feature space. One of the major challenges with the system is performance. Since real-time feedback is required during the user interaction, several data reduction and acceleration techniques are utilized.

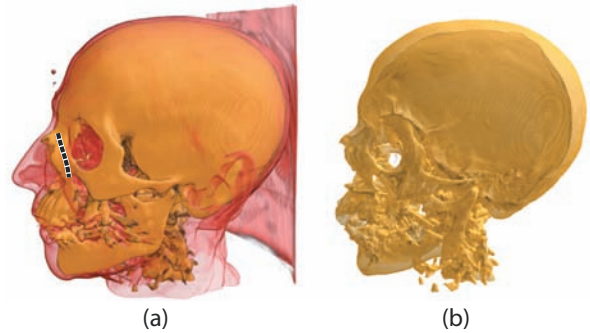

Fig. 5. The result of the feature inference by a user stroke.

5.2.1 Image Space Analysis

The first step of feature inference is to find the most significant visual objects in the 2D image space, which can be achieved by segmenting the image into foreground and background patches according to the user input. The appearance of the foreground patch should be closer to the pixels that are highlighted by user strokes. After image segmentation, the corresponding data are collected by ray traversal from the pixels for further feature inference. To accelerate the whole pipeline, we only consider the image in the local region. In practice, the window size of the local image is set to 30×30. If the window size is too small to cover enough features, further data space analysis may not exclude ambiguity as desired. On the other hand, the time and storage complexity of image segmentation and further data space analysis will prevent the pipeline working in real time.

We utilize the Graph Cut algorithm [13], which combines high performance with good segmentation quality for extracting the foreground and the background of the local image. Since we only have foreground constraints for the image segmentation, we further employ a two-pass method [38] to generate the background seeds before the second pass of the standard Graph Cut process.

Suppose the image is a graph $\mathscr{G}\langle\mathscr{V},\mathscr{E}\rangle$, where nodes $\mathscr{V}$ are the pixels in the image, and edges $\mathscr{E}$ are the relationships between the neighborhoods. Each node i in $\mathscr{V}$ will be labeled as $l_i = 0$ or 1 (background or foreground), by optimizing the energy function:

$$E(\mathbf{l}) = \mu \sum_i R(l_i) + \sum_{i,j} B(l_i, l_j), \tag{2}$$

where R is the likelihood function, and B presents the boundary properties. In the general Graph Cut algorithm, it requires the initial seed set of the foreground G_f and G_b, and then the likelihood function can be written as:

$$\begin{cases} R(l_i=1)=0 & R(l_i=0)=\infty & i \in G_f, \\ R(l_i=1)=\infty & R(l_i=0)=0 & i \in G_b, \\ R(l_i=1)=-ln(\frac{d_i^F}{d_i^B+d_i^F}) & R(l_i=0)=-ln(\frac{d_i^B}{d_i^B+d_i^F}) & otherwise. \end{cases} \tag{3}$$

The distance d_i^F and d_i^B are evaluated as the minimal distance between the pixel and any clusters in G_f or G_b. The clustering is performed by K-means algorithm. However, in our problem, we only have the initial foreground G_f available. The two-pass Graph Cut algorithm is used to decide G_b in the first pass with a modified R, and then run in the second pass with G_f and G_b using Eq. 3. The modified R in the first pass is:

$$\begin{cases} R(l_i=1)=0 & R(l_i=0)=\infty & i \in G_f, \\ R(l_i=1)=-ln(\frac{d_i^F}{K+d_i^F}) & R(l_i=0)=-ln(\frac{K}{K+d_i^F}) & otherwise. \end{cases} \tag{4}$$

where K is a constant, which is the estimation of the distance d_i^B in Eq. 3. The second term of the energy function 2 is the boundary property:

$$B(C_i, C_j) = \alpha exp(-\frac{||C_i - C_j||^2}{2\sigma^2}), \tag{5}$$

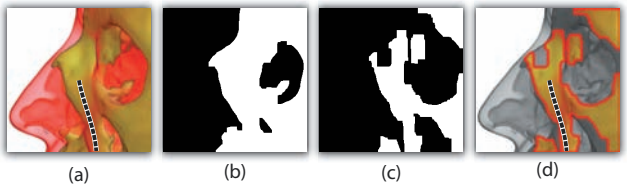

Fig. 6. The two pass Graph Cut: (a) the input seed stroke as G_f in the first pass; (b) the G_b generated by the first pass (black portion); (c) the final result of the segmentation; (d) the segmentation result, where the foreground is in color and bounded by red lines, the background in gray. The final foreground mask is the approximation of the bone structure, and the background mask contains irrelevant and mixed features.

where the distance $||C_i - C_j||$ is the difference of pixels in the color space. α and σ are pre-defined parameters. The result of the two-pass Graph Cut algorithm on the local image is illustrated in Fig. 6. The final foreground patch presents the approximation of the bone structure, where it is less occluded by outer skin. The classification results on image space are the seed for further data space analysis, which extracts the corresponding distribution on the feature space.

5.2.2 Data Space Analysis

Data space analysis aims to infer feature clusters from the data space based on the result of the image segmentation results. The data values, as well as their visual properties are collected by ray traversal, and then they are classified into clusters. A synthesized score is calculated for each cluster, as the result of data space analysis. In further processes in the pipeline, most of the algorithms only need to consider the cluster with the maximum score, while some of the algorithms account for multiple clusters (e.g. contrast enhancement, etc.)

In the ray traversal process, there are four properties collected at each sample point, including the depth, visibility, opacity, as well as the data value. The depth values can be directly calculated according to ray configurations. The data value and opacity value are looked up from both volume data and the transfer function. In addition, the visibility value, which is the contribution of a sample to the final image [7, 8], is formalized as:

$$V(x) = (1 - O(x))\alpha(s(x)), \tag{6}$$

where $O(x)$ is the current attenuation value of the ray, and $s(x)$ is the sample value at position x.

The input of the clustering algorithm is the multivariate data samples collected from the ray traversal, which are visualized in Fig. 7(a). Typically, the amount of the input samples can reach 10^6. To handle the multivariate samples with such size, K-means, which is a light weight clustering tool is exploited before further processing. The performance of the K-means is further boosted by accelerated GPU implementation for real-time interaction. The distance metric of the multivariate samples is weighted Euclidean distance. As we observed in the experiments, the change of the weights only slightly affects the clustering result, if reasonable weights are given to the data value and visibility properties. The choice of the centroid number K is based on the estimation of how many different features (both visual and numerical) are behind the local image region. A higher centroid number will significantly increase both time and storage complexities. Typically, K is set to 10, and this number can be modified depending on the complexity of the dataset.

The above ray traversal and clustering process are applied on both foreground and background image patches that are generated from image space analysis. Thus, we obtain the foreground clusters $\{F_f^i\}$ and the background clusters $\{F_b^i\}$. The background clusters help in the exclusion of ambiguity, since features that appear in both the foreground and the background are not likely to be the desired ones.

The score for each cluster needs to consider several aspects, including the visual significance (visibility), the depths, as well as the ambiguity. Among the above factors, the principle property is the visibility. Hence we can define the score of the i-th cluster as

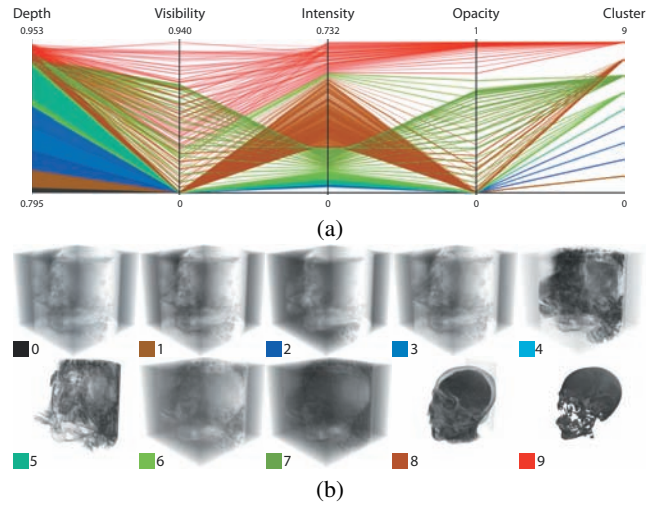

Fig. 7. The clustering result of data space analysis: (a) visualization of the multivariate samples with parallel coordinates, presenting four numerical properties and the cluster membership information; (b) the corresponding features of the clusters.

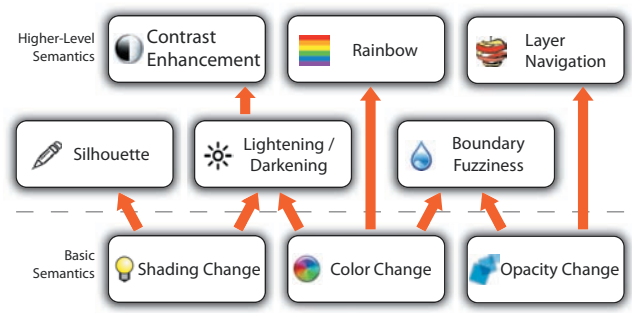

Fig. 8. Dependency relationships of the semantic realization algorithm. The realization algorithm of higher level semantics is built upon the basic ones.

$$S(i) = V_i, \tag{7}$$

where V_i is the average visibility of the i-th cluster. More factors should be considered, when the visibility of the clusters is similar, or serious ambiguity exists. So we amend the score function by adding two auxiliary items:

$$S(i) = V_i + \beta_1 D_i + \beta_2 d_i^{F_b}, \tag{8}$$

where D_i is the centroid value of depth of the i-th cluster, and $d_i^{F_b}$ is the minimum distance between the centroid and all centroids in the background clusters F_b, which measures the ambiguity. The coefficients β_1 and β_2 are the weights, which makes the amended items smaller than V_i. Based on our experiments and observation, typical values for β_1 and β_2 are around 10^{-4} and 10^{-3}. Greater values of β_1 and β_2 may hide the principle property V_i, but ignoring the influence of the two factors may make the selection undesirable. For faster user interaction, β_2 can be set to 0, since only the data behind the foreground image patch needs to be calculated.

5.3 Semantic Realization

Semantic realization converts the feature inference result into new transfer functions, thus updating the volume rendered image gradually during the interaction. In our tool set, color, transparency and

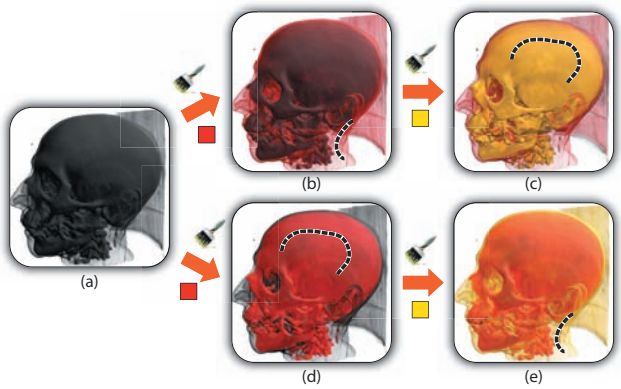

Fig. 9. The colorization operation: (a) the initial status; (b) the skin painted with red color; (c) the bone painted with yellow color; (d) the bone painted with red color; (e) the skin colorized with yellow. The black curves illustrate how the results can be achieved by painting on the image.

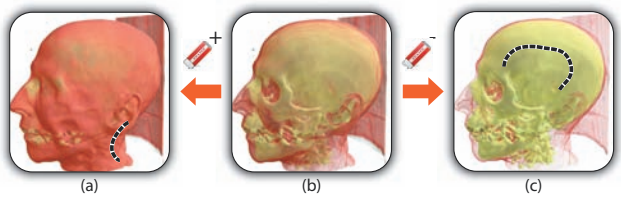

Fig. 10. The eraser operation: (a) the opacity of the skin portion is increased; (b) the initial status; (c) the opacity of the skin portion is decreased.

shading changes are the most basic semantics, and higher level semantics are built upon the basic semantics by decompositing the semantics into lower level ones. For example, changing the brightness indirectly changes the color, and contrast enhancement indirectly changes the brightness. The semantic realization can be denoted as an Incremental Transfer Function (ITF), which is the incremental modification of the current TF, and can in turn be generated by compositing a new color and transparency matrix $\mathbf{C}'$:

$$\Delta TF = \mathbf{C}'^{\mathbf{T}}\mathbf{F}. \tag{9}$$

In most cases, such as for color and transparency change, only the cluster with the maximum score is considered. For multi-feature operations, e.g. contrast enhancement and layer navigation, multiple features are considered.

In order to control the speed of feedback during the interaction (Fig. 3), we define the time rate λ to control the speed of incremental changes. The time rate λ can be either manually assigned by users or be automatically adjusted according to the difference of consecutive images. By default, λ equals to the reciprocal of the current frame rate. On each time step t, the new TF_t is the sum of the TF on the previous time step and the ITF weighted by the time rate λ:

$$TF_t := TF_{t-1} \oplus \lambda \Delta TF_t, \tag{10}$$

where operation $\oplus$ is the alpha blending of two colors.

5.3.1 Color and Transparency Changes

Color and transparency are among the most important factors affecting the visualization results. Higher level semantics like fuzziness, brightness and contrast are also based upon color and transparency changes. All the row vectors in the color matrix $\mathbf{C}'$ of the ITF are set to be the destination color. The visual effect of color and transparency changes is shown in Fig. 9 and Fig. 11.

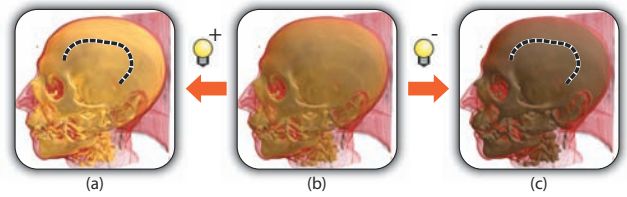

Fig. 11. The brightness operation: (a) the brightness of the bone is increased; (b) the initial status; (c) the brightness of the bone is decreased.

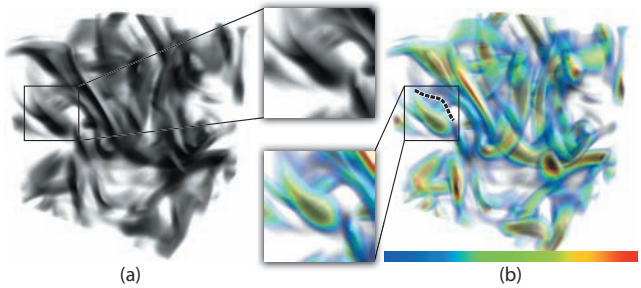

Fig. 12. Rainbow operation: (a) before operation; (b) after operation. The color strip in (b) is the user selected rainbow for the effect.

Fig. 13. Boundary fuzziness: (a) before operation; (b) after operation.

5.3.2 Brightness Change

The lightening and darkening semantics aim to adjust the luminance of the features. They can be generalized as changing the luminance component of the colors in the CIE $L^*a^*b^*$ color space. The object color C' can be transformed from the original color in the transfer function:

$$\mathbf{C}' = diag(1 \pm \lambda, 1, 1, 1)\mathbf{C}, \tag{11}$$

where the diagonal matrix only transforms the luminance component of the original color. The color can be further converted into RGB space for rendering.

5.3.3 Rainbow

As we described in Section 4, the rainbow effect encodes the desired features into different colors, according to the data distribution. We perform a histogram equalization for the desired feature, before mapping the colors to the transfer functions.

5.3.4 Boundary Fuzziness

The fuzziness semantic makes the boundary of the visual feature more fuzzy by making the local TF more smooth. This process is done by a feature space anisotropic diffusion equation, whose conductivity is inversely proportional to the data histogram $H(\mathbf{F})$:

$$\frac{\partial \mathbf{C}}{\partial t} - \mathbf{A}\frac{1}{H(\mathbf{F})}\Delta \mathbf{C} = 0, \tag{12}$$

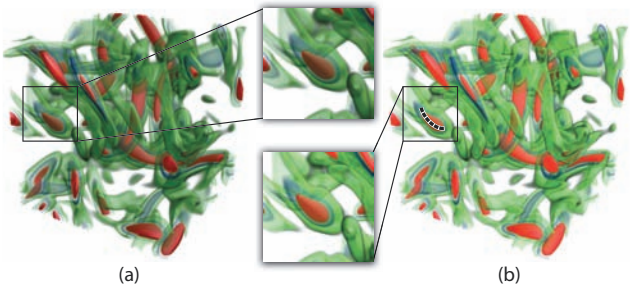

Fig. 14. Contrast enhancement: (a) before operation; (b) after operation.

where $\Delta\mathbf{C}$ is calculated by differencing the neighborhood opacity values, and each scalar a_i in the conductivity vector $\mathbf{A}$ equals to 0 or 1, depending on whether the feature primitive $\mathbf{f}_i$ is in the selected feature cluster. After a few steps of iterations, the modified color matrix $\mathbf{C}$ can be obtained for the transparency transform. During the diffusion, the color is also blended with neighborhood features.

5.3.5 Contrast Enhancement

The contrast enhancement is based upon color changes. In image processing, contrast enhancement can be done by histogram equalization. Since transfer functions indirectly take effect on the image contrast in volume rendering, we need an alternative way to achieve the goal. In our work, the image contrast is enhanced by increasing the contrast of the features in the volume. By utilizing the feature inference result F_f, a discrete histogram equalization is conducted.

Denote $\mathbf{f}_k$ as the clusters in F_f, and l_k as the luminance contribution of feature $\mathbf{f}_k$, which is the sum of the luminance of the samples weighted by the visibility values. Firstly, the normalized accumulated histogram of the features is calculated after sorting:

$$c_i = \sum_{j}^{i} l_i. \tag{13}$$

We would like to create a transform to make the accumulated histogram c_i linear, i.e. $c_i = iM$, for a constant M. The final objective luminance of each feature l'_k is

$$l'_k = c_i(l_{max} - l_{min}) + l_{min}. \tag{14}$$

After the histogram equalization, the contrast enhancement is decomposed into several brightness operations.

5.3.6 Layer Navigation

The layer navigation aims to help users find hidden features along the ray. Similar to contrast enhancement, it is also a multi-layer operation, which is composited as the sum of transparency changes. The weight of each layer is defined by users with the peeling slider (Fig. 4). When the ratio of each layer is changed, a new transfer function is generated.

6 Implementation and Performance

The system was implemented in C++ and GLSL. A standard ray-casting volume rendering engine with a pre-integrated transfer function [11] is utilized. Currently the feature space is 1D. The silhouette rendering is also based on a pre-integration technique. Various GPU acceleration techniques for user interactions are exploited, e.g. K-means clustering, etc. The feature inference and the volume rendering engine are executed in separate threads asynchronously, in order to hide the lags during the interaction. During the user interaction, the feature inference is invoked when any new stroke point is inputted. Meanwhile, the rendering thread keeps on updating the image result. When the feature inference is faster than the rendering, it runs multiple cycles before the next frame is completed, thus reducing the rendering overhead.

Data Set	Size	T_r	T_i	T_d	T_s	T
Head	256x256x225	82.4	0.8	17.6	4.6	23.0
Visible Human (Part)	512x512x512	219.3	1.1	49.7	4.2	55.0
Vorticity	128x128x128	30.9	0.7	6.6	4.6	11.9
Engine Block	256x256x110	62.4	0.8	30.7	4.2	35.8
Isabel	500x500x100	117.4	0.8	22.7	4.6	28.2

Table 1. The average timings of the system. T_r, T_i, T_d, T_s are the timings (in milliseconds) of rendering, image space analysis, data space analysis, semantic realization; T is the total time of feature inference.

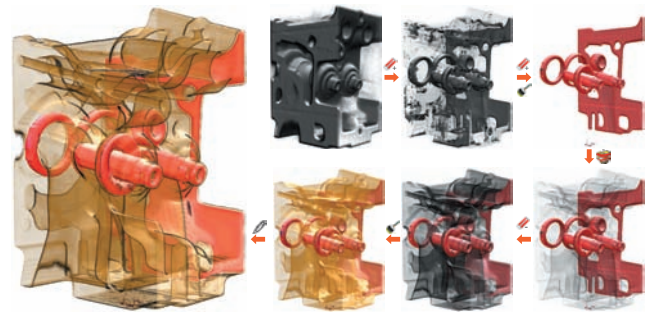

Fig. 15. The final result and intermediate steps of exploring engine data

The performance of the proposed system was tested on a Dell T3400 workstation, with a 2.66GHz CPU, 4GB memory, and an NVidia GTX 470 graphics card with 1280MB video memory. The timings of the test data are listed in Table 1. The steps size is 0.5 voxel, and the rendered image resolution is 800x800. Full shading with silhouette style, which is controlled by shading transfer function, is performed during the rendering. In addition to the step size and image size, other factors also influence the performance, e.g. the window size for local image analysis, which indirectly decides the amount of the samples in data space analysis.

In addition to the standard PC environment, touchable devices, e.g. iPad are also supported for users. The proposed WYSIWYG system provides a greater advantage to portable devices, since it is extremely inconvenient and inaccurate to design transfer functions with fingers on limited screen spaces with traditional tools. It also shows convenience for volume data navigation on the iPad with multi-touch support. Gestures can be mapped to take the place of auxiliary key operations. Multiple users can not only share but also collaborate on the same volume data at the same time. Due to the limited computational resources on the iPad, a remote system is implemented. In our system, an iPad acts as the client, and a workstation acts as the server. The touch signals are transferred to the server, and then the server transfers the updated volume rendered images back to the iPad. The average bandwidth of the compressed transmission is about 200kb/s in a WLAN environment. The image quality can be dynamically changed according to the frame rate.

7 Results

In this section, we demonstrate several cases of volume data exploration with our WYSIWYG data exploration tool.

In Fig. 15, we demonstrate how to explore an engine block CT scan with our system in a few steps. In the very beginning, the image is rendered with the linear ramp transfer function. After removing the outer materials, the core part of the block is revealed. This part can be further colorized as orange. The boundary of the outer part can be recovered by utilizing the peeling slider. Then a fine visualization result can be achieved.

For medical data, which contains many implicit layers with rich semantics, it is more straightforward to explore the features of the data during 3D navigation. Fig. 16 presents intermediate and final rendering results of the Visible Male data set. Domain knowledge can be better integrated into the whole exploration process. Users can quickly

Fig. 16. The final result and intermediate steps of exploring partial visible male data.

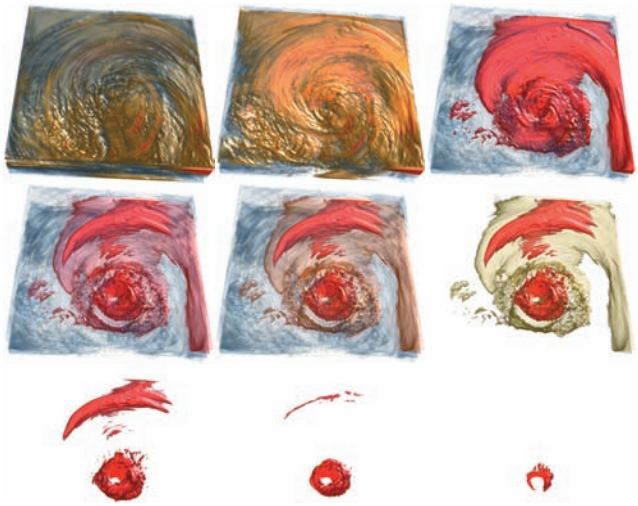

Fig. 17. The possible process in the exploration of Hurricane Isabel data

identify the meaningful features and layers, and then make them highlighted in different appearances.

Our system is also capable of navigating atmospheric simulation data (Fig. 17). We choose the wind speed channel of the Hurricane Isabel data set for demonstration. By eraser and rainbow tools, several major structures are revealed in the result. Users can interactively explore and understand the data with our tool. The visual effect can be further improved by contrast enhancement, etc. Compared to traditional transfer function design tools, our tool is more intuitive and natural for scientific data exploration.

Based on user feedback from several users, especially from users new to volume rendering, the proposed method is more intuitive than traditional transfer function design. It is natural and direct to operate in image space, without the trial-and-error process necessary with feature space. The goal-oriented data exploration process also makes direct volume rendering more usable and accessible for users. In the future, we plan to perform a formal user study to thoroughly investigate the system.

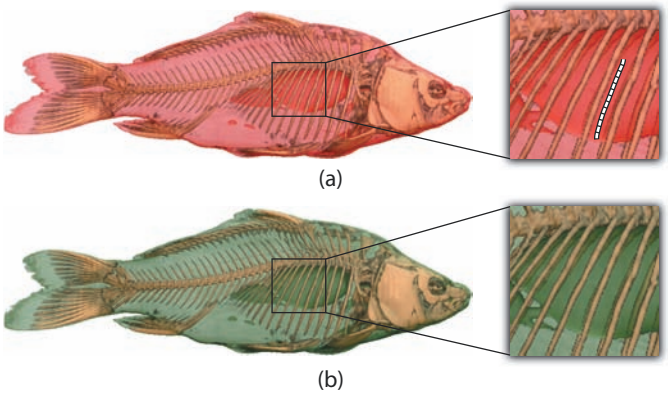

Fig. 18. Illustration of the global operation limitation: (a) user sketch on the air bladder of the CT carp data with colorization tool; (b) both the air bladder and the skin, which are with the same properties, are turned to green.

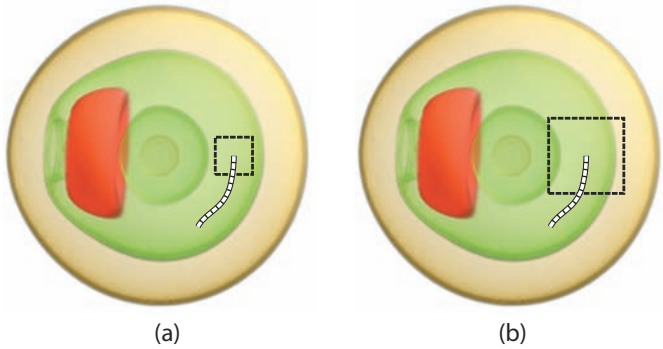

Fig. 19. Relationships between the quality of ambiguity exclusion and the window size of local image analysis: (a) the window size is too small to distinguish the semantic foreground (green materials) and the background (yellow materials), which leads to high ambiguity; (b) the window size is enough to ensure the green materials are the intended feature.

8 Limitations and Known Issues

There are a few limitations and known issues of the proposed system. First, all the operations on the rendered image are global, instead of local operations. During the interaction, effects are taken on all features with the same properties as the user sketches on the image. In Fig. 18, we demonstrate how the user failed to change the color of the surface of the air bladder, while leaving the other parts unchanged. However, since the surface of the air bladder and the skin of the carp are within the same range of intensity values, they are not able to be distinguished by the classification of 1D transfer functions.

Second, the exclusion of ambiguity may be unsatisfactory when the visibility and depths of overlapped transparent materials are too close. Meanwhile, the footprint of these materials on the image are much larger than the window size of local image analysis (c.f. Section 5.2.1). In Fig. 19, the visibility of both yellow and green materials are very close, leading to high ambiguity. This problem could be fixed by enlarging the window size (Fig. 19(b)), but this would be at the cost of sacrificing more computation time.

Third, the initial parameter configurations may influence the exploration process, e.g. initial transfer functions, the viewpoints, etc. In addition, the frame rate of the direct volume rendering also has a big impact on the visual effect. If the frame rate is too low, the interactivity of the system is significantly reduced.

9 Conclusions and Future Works

In this work, we present a novel WYSIWYG volume visualization system that accepts direct sketch on the rendered images. Users can ex-

plore and change the appearance of the direct volume rendered images by sketching on the volume rendered image with various tools, and the system gives real-time feedback for users during the interaction. To achieve this goal, several algorithms are utilized for the interactive operations, including image and data space analysis, and the semantic realization process. The proposed system is flexible and intuitive for volume data exploration and visualization.

A few extensions and applications for this work can be developed in the future. More NPR effects other than silhouette can be added into the current system to make it available for sketch-based illustrative visualization design. For the intention inference algorithm, we would like to further develop specific strategies for interval volume rendering [1], which provide a wider spectrum of visual classification. We would also like to extend the current system to facilitate multidimensional and multivariate transfer function design for more applications. Furthermore, it is noticeable that local transfer functions [24] and two-level volume rendering [15] with segmentation information provides better classification for local structures of the data, instead of global mapping, but the parameter space is even more complicated.

Acknowledgments

The authors wish to thank the anonymous reviewers for their comments. This work is supported by National Natural Science Foundation of China Project No. 60903062, Beijing Natural Science Foundation Project No. 4092021, 863 Program Project 2010AA012400, Chinese Ministry of Education Key Project No. 109001.

References

[1] M. Ament, D. Weiskopf, and H. Carr. Direct interval volume visualization. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1505–1514, 2010.

[2] C. L. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *Proceedings of IEEE Visualization 1997*, pages 167–174, 1997.

[3] S. Bruckner and M. E. Gröller. Volumeshop: An interactive system for direct volume illustration. In *Proceedings of 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] 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.

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

[7] C. Correa and K.-L. Ma. Visibility driven transfer functions. In *Proceedings of IEEE Pacific Visualization 2009*, pages 177–184, 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] B. Csébfalvi, L. Mroz, H. Hauser, A. König, and E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Comput. Graph. Forum*, 20(3):452–460, 2001.

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

[11] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, 2001.

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

[13] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721 –741, 1984.

[14] H. Guo, H. Xiao, and X. Yuan. Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates. In *Proceedings of IEEE Pacific Visualization 2011*, pages 19–26, 2011.

[15] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of IEEE Visualization 2003*, pages 301–308, 2003.

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

[17] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH 1999*, pages 409–416, 1999.

[18] 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 *Proceedings of ACM SIGGRAPH 2002*, pages 755–762, 2002.

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

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

[21] P. Kohlmann, S. Bruckner, A. Kanitsar, and M. Groller. Contextual picking of volumetric structures. In *Proceedings of IEEE Pacific Visualization Symposium 2009*, pages 185–192, 2009.

[22] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, 2004.

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

[24] C. Lundström, P. Ljung, and A. Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 12(6):1570–1579, 2006.

[25] M. M. Malik, T. Möller, and M. E. Gröller. Feature peeling. In *Proceedings of Graphics Interface 2007*, pages 273–280, 2007.

[26] 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 *Proceedings of ACM SIGGRAPH 1997*, pages 389–400, 1997.

[27] S. Owada, F. Nielsen, and T. Igarashi. Volume catcher. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pages 111–116, 2005.

[28] D. Patel, M. Haidacher, J.-P. Balabanian, and E. Groller. Moment curves. In *Proceedings of IEEE Pacific Visualization Symposium 2009*, pages 201–208, 2009.

[29] H. Pfister, W. E. Lorensen, C. L. 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.

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

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

[32] 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.

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

[34] 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.

[35] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization 2003*, pages 505–512, 2003.

[36] 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.

[37] 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.

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