Feature-Driven Data Exploration for Volumetric Rendering

Insoo Woo, *Member*, *IEEE*, Ross Maciejewski, *Member*, *IEEE*, Kelly P. Gaither, *Member*, *IEEE*, and David S. Ebert, *Fellow*, *IEEE*

Abstract—We have developed an intuitive method to semiautomatically explore volumetric data in a focus-region-guided or value-driven way using a user-defined ray through the 3D volume and contour lines in the region of interest. After selecting a point of interest from a 2D perspective, which defines a ray through the 3D volume, our method provides analytical tools to assist in narrowing the region of interest to a desired set of features. Feature layers are identified in a 1D scalar value profile with the ray and are used to define default rendering parameters, such as color and opacity mappings, and locate the center of the region of interest. Contour lines are generated based on the feature layer level sets within interactively selected slices of the focus region. Finally, we utilize feature-preserving filters and demonstrate the applicability of our scheme to noisy data.

Index Terms—Direct volume rendering, transfer function, focus+context visualization.

1 Introduction

The emergence of web-based scientific simulation portals [1], [2] has created an abundance of volumetric data, and an effective means of exploring volumetric data is through direct volume rendering. One of the most common methods for direct volume rendering is to employ the use of interactive transfer function widgets which users interactively use to define a mapping of the volumetric data values to optical properties (color and opacity). Unfortunately, creating an appropriate transfer function often involves tedious adjustment and fine tuning of parameters, resulting in a trial-and-error type approach by the user. This problem is further compounded in scientific portals (e.g., the NanoHub [3]), as the scientists who need to analyze the data are often novices in volumetric rendering. Furthermore, relevant sparse features within the volumetric data can easily be occluded by semitransparent surrounding values, making it difficult to find the specific value range within the occluded regions that should be enhanced for better visualization and analysis. This volumetric exploration problem is illustrated in Fig. 1.

In Fig. 1, a scientist has created an Indium-Arsenic (InAs) quantum dot simulation [4], and the scientist wants to explore the electron wave function data encapsulated in the resultant volumetric output. Here, the important structure to the end user is the vertical stack consisting of three InAs quantum dots. However, this structure is not easily found using 1D and 2D histogram widgets. The transfer function widget in Fig. 1 (middle) uses a 2D histogram widget [5] plotting the value versus value gradient magnitude of the volumetric data. The user has drawn a series of rectangular boxes within the widget. The structures found in region A of the histogram are of little to no importance to the scientist, and, while the most important structures of the data are located in the histogram region of Fig. 1 marked B, the data in region B are so sparse that there are no visual cues to guide the user to select those regions in the histogram space. Note that both a linear and logarithmic mapping of the opacity was applied in the 2D histogram with similar results.

Thus, novice end users of direct volume rendering tools need methods where rendering parameters can be suggested in a way that incorporates the end-user's expert knowledge of interest within the data set and the properties of the volumetric data. In this work, we propose a novel interaction approach to semiautomatically generate appropriate rendering parameters to help users efficiently explore and analyze their volumetric data sets. Key contributions include the following:

- A hardware-accelerated graphics pipeline (Fig. 2) that enables interactive visualization through a user-specified region of interest. Features within the region of interest are analyzed and automatically mapped to a set of rendering parameters based on the features. Moreover, our pipeline includes feature-preserving denoising methods (e.g., median filters) to support noisy volumetric data analysis.
- A novel selection scheme for color mapping and data enhancement based on data types (interval and ratio), utilizing measurement theory. Features of interest can be highlighted based on regions and focus values.

- *I. Woo is with the Purdue Visual Analytics Center, Purdue University, PO Box 519, 465 Northwestern Ave., West Lafayette, IN 47907. E-mail: iwoo@purdue.edu.*
- *R. Maciejewski is with the Arizona State University, PO Box 878809, Tempe, AZ 85287. E-mail: rmacieje@asu.edu.*
- *K.P. Gaither is with the Texas Advanced Computing Center, University of Texas, Research Office Complex 1.101, J.J. Pickle Research Campus, Building 196, 10100 Burnet Road, Austin R8700, TX 78758-4497. E-mail: kelly@tacc.utexas.edu.*
- *D.S. Ebert is with Purdue Visual Analytics Center, Purdue University, 465 Northwestern Ave., West Lafayette, IN 47907. E-mail: ebertd@purdue.edu.*

Manuscript received 2 July 2010 ; revised 15 Nov. 2010; accepted 23 Dec. 2011; published online 26 Jan. 2012.
Recommended for acceptance by T. Möller.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-07-0132.
Digital Object Identifier no. 10.1109/TVCG.2012.24.

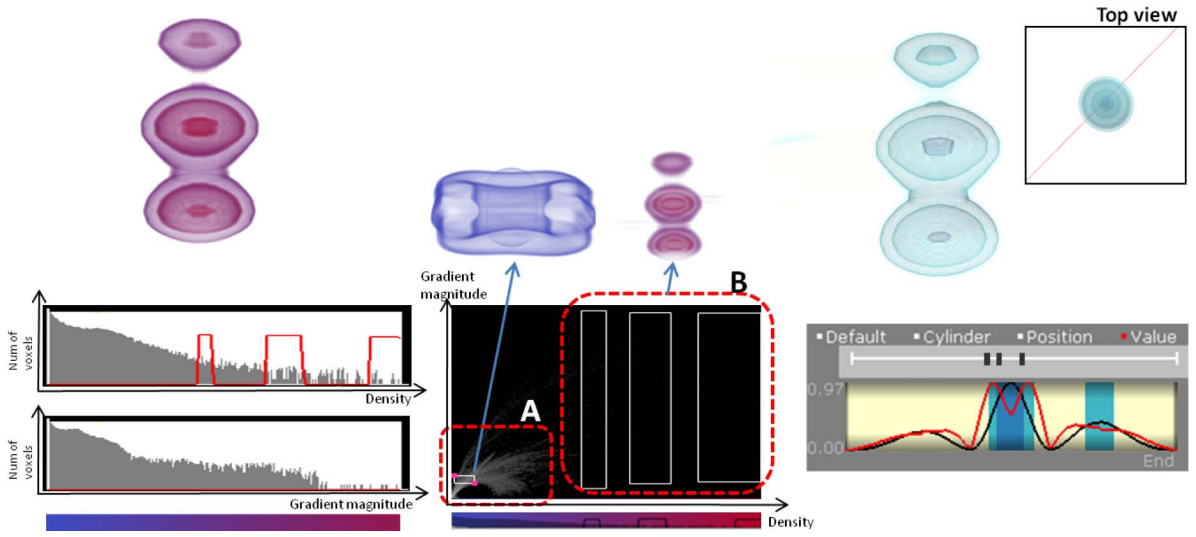


Fig. 1. Comparison of transfer function design between a 1D histogram widget (left), a 2D histogram widget (center), and our semiautomated transfer function widget (right). In the 2D histogram, the region A includes the majority of the volumetric data; however, the most important information is found in the sparse areas (B) in the 2D histogram. In our method, the user is able to highlight meaningful boundaries using the value profile for the region of interest. The "top view" (rightmost figure) shows the ray (in red) passing through the volume and the line graph displays the scalar values and gradient magnitudes along the ray (in red).

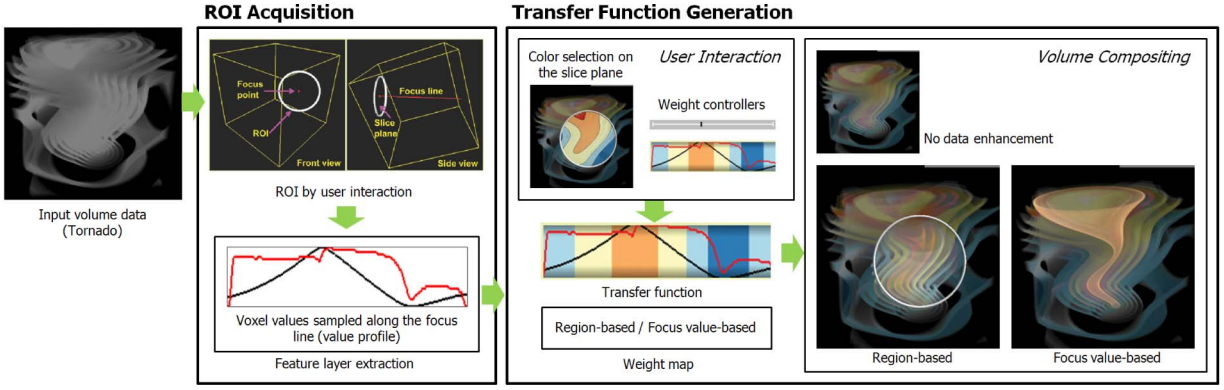


Fig. 2. Schematic diagram of our semiautomated data exploration method. The sequence consists of user's region of interest acquisition, feature extraction, color selection, and data enhancement.

- An intuitive user interface for transfer function design (Fig. 3), modification, and interaction utilizing line charts and contour lines within a slice view for enhancing local data features. Level sets are extracted from a slice view using a CUDA [6] kernel, contour lines are displayed on the view, and then colors can be changed on the slice view based on the level set.

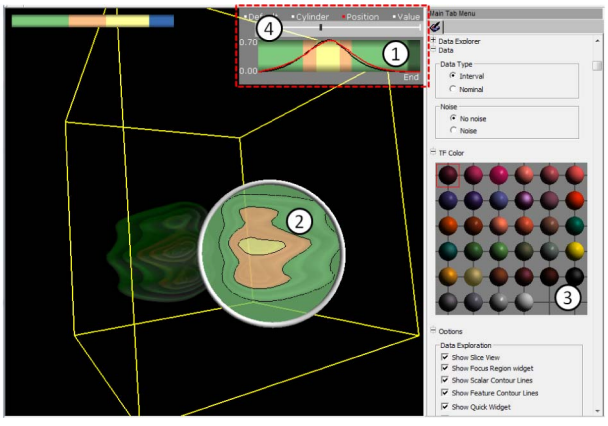


Fig. 3. The user-interface controls of our interactive feature-driven exploration tool. 1: The value profile widget. 2: The contour line widget. 3: The color palette selection widget. 4: The focus region widget.

The use of the feature selection tool enables users to generate appropriate transfer functions more effectively. Furthermore, the addition of the value profile widget provides simulation scientists with another view in which to analyze their data. Finally, our semiautomated tools make use of line graphs and heightmaps that are generated by sampling scalar values along a specific line segment, or on a specific slice surface within the volumetric data. By applying both line graph and contour line metaphors within a volume visualization application, users can further investigate and analyze data features by specifying the *spatial region* of interest within the volumetric data or by varying the *scalar value range* of interest to highlight features.

In this paper, we will discuss the advantages of our pipeline and interactive widgets and their ability to enhance the overall visualization and analysis process for the end user. In the next section, we summarize the related work in transfer function design, color selection, and image denoising. In Section 3, we describe our pipeline and algorithms

for interactive transfer function design. In Section 4, we explain the implementation details of the algorithm. Finally, we discuss our results and present feedback from end users in Section 5 and propose future work in Section 6.

2 Related Work

Interactive transfer function design has been addressed through many different approaches. Direct manipulation widgets help users to explore volumetric data sets by using data probes, clipping planes, classification, shading, and color picking widgets [5], [7]. Some approaches have utilized design galleries [8] or parallel coordinate style interfaces [9] for transfer function design, while other recent approaches have focused on user-oriented design schemes. For example, Rezk-Salama et al. [10] provided a high-level transfer function model using widgets that map directly to volume features (such as controls for bone color and opacity). Wu and Qu [11] designed a framework to manipulate rendered images for multiple volumetric objects and merge the images into a single rendering. Bruckner and Gröller [12] employed a stylized transfer function widget that enables users to intuitively select a nonphotorealistic shading style from the style transfer function. Bajaj et al. [13] introduced the contour spectrum interface that presents users with a collection of data characteristics to select significant isovalues for isosurfacing. Lundström et al. [14] introduced the α-histogram which incorporates spatial coherence as a means of automated transfer function design. This work divides the data into subblocks (spatial sub regions), computes local histograms, amplifies peaks by raising histogram values to the power of α, and calculates an α histogram by summing up all the amplified histograms. The work by Lundström et al. provides transfer functions on a global scale as opposed to our work, which focuses on enhancing local data features. Correa and Ma [15] proposed a new volume exploration technique by introducing the concept of a scale field that allows users to assign colors and opacities based on size. However, the concept of a scale field works best when the data sets have a well-defined underlying shape, which is often not the case in scientific simulations.

Besides ill-defined volumetric structures, another issue is that the size and complexity of scientific simulations has also been drastically increasing. As such, recent work has focused on the use of focus+context visualization [16]. In order to enable users to select a region of interest with ease, research has focused on two primary tasks: defining a focus and context area, and effectively emphasizing the focused area. Lu et al. [17] inferred a user's regions of interest by capturing eye gaze positions as the user watched a rotating volume. Li et al. [18] proposed a system for automatically generating a variety of cut-away illustrations. Viola et al. [19] introduced cut-away views for volume rendering based on importance by suppressing less important information. However, their method requires data segmentation and user-assigned importance values to each data segment. Ropinski et al. [20] introduced a stroke-based transfer function design scheme computing a difference histogram for all pairs of the control points along inner and outer strokes [20]. For better peak detection, this work used weights based on the voxels' visibility. It works properly when the data have distinctive volume object boundaries since the user can easily draw stroke lines along the intensity boundary. However, many simulation data sets lack well-defined edges and boundaries.

Along with focus+context methods, raycasting-based data analysis methods have also been employed to aid users when exploring their volumetric data. These methods create automated or semiautomated rendering parameter settings or provide an interface supporting volumetric layer exploration. Opacity peeling [21] generates an image of an opacity layer (image buffer) whenever an accumulated alpha value exceeds a threshold and allows the user to explore the opacity layers. Malik et al. [22] extended the concept by analyzing values sampled along rays through the volumetric data and extracting feature layers. Instead of accumulating opacity, Malik's work searches for transition points from the sampled values to extract feature layers with prominent peaks. Correa et al. [23] introduced the visibility histogram where, for a given viewpoint, visibility is weighted using opacity, which is considered as importance, and used to maximize the visibility of the value range of interest. Kohlmann et al. [24] used ray profiles to facilitate the process of finding and highlighting interest points in 2D slice views starting from the users' mouse click location in the 3D view. However, their main focus is on the construction of a ray profile library that stores ray profile samples for different CT data sets and on matching profiles to find close similarities.

Another factor to consider in scientific simulation data is that the data are often temporal in nature. As such, previous work has also focused on defining a transfer function for time-varying volumetric data. Recent work by Akiba and Ma [25], Akiba et al. [26] utilized parallel coordinate plots to create a volume rendering interface for exploring multivariate time-varying data sets. By means of a prediction-correction process, Muelder and Ma [27] proposed a method to predict the feature regions in the previous frame, making the feature tracking coherent and easy to extract the actual features of interest. Maciejewski et al. [28] used density estimation for creating a temporally coherent set of transfer functions of a group of feature space histograms.

While these previous methods have attempted to provide user friendly interfaces for transfer function manipulation, there are still barriers to the adoption of the methods by scientists and general users. In this paper, we focus on a new data exploration and transfer function design scheme. We provide the end user with familiar ray profile plots and an interactive means of defining regions of interests, as shown in Fig. 3. From these tools, we are able to semiautomatically generate rendering parameters and reduce some of the transfer function design burden for the end user.

3 Interactive Feature-Driven Data Exploration

Since analysis methods are affected by the structure and nature of data [29], we define our visual mapping parameter choices based on the underlying data type. In measurement theory, the measurement scale is categorized into four

TABLE 1
Color and Opacity Selections Based on Their Data Type

Data Type	Noise	Filtering	Color Sel.	Opacity Enhancement
Interval	with	O	Sequential	Local peak value per layer
	without	X		
Ratio	with	O	Diverging	Local maximum gradient magnitude per layer
	without	X		

categories (nominal, ordinal, interval, and ratio) [30]. Nominal data have no order, ordinal data have order, interval data have meaningful intervals, and ratio data have the height level of measurement. These different types of data have features that need to be highlighted in different ways [29]. In the case of volumetric data, we consider two categories of data (Table 1), interval and ratio data, and generalize a set of guidelines and rules for automated or semiautomated methods for our visualization techniques. Nominal and ordinal data types are left for future consideration.

The color selection schemes, emphasized areas, and filtering parameters are semiautomatically determined by the user-defined data type. These schemes are applied automatically in our transfer function generation pipeline depending on the data type. Fig. 2 illustrates the sequence of operations in our approach. First, we provide the user with an initial rendering of their data. During our first pass, transfer function parameters are generated using a sine function that maps the scalar values to an opacity, thus creating high and low contours [31] as seen in Fig. 2 (input volume data). Next, using the initial volume rendering, the user interacts with the volume. Mouse clicks indicate the user's region of interest (Fig. 2, ROI acquisition), and a focus point and a focus region are created resulting in a focus line that is a viewing ray passing through the focus point. Then, our system extracts feature layers to compute the number of colors and the data enhancement points and displays the results in a value profile widget. A color transfer function and a weight map are automatically defined based on the extracted feature layers (Fig. 2, transfer function and weight map). The weight map is utilized to modulate the opacity. In this way, users can explore their data in focus-region-based or focus-value-driven ways (Fig. 2, volume composition). In addition, we apply nonlinear filters to the value profiles and slice view images to support our transfer function generation approach for noisy volumetric data sets.

3.1 Region of Interest (ROI) Acquisition

The region of interest acquisition is the first step in our data exploration method. Earlier methods for ROI selection included the use of the mouse cursor along with several prespecified context layers from the volume data set [32]. However, this method entails the tedious process of manually creating several context layers that capture important features within the data set. Our work chooses to utilize a simpler interface design in which mouse movements are used to obtain the ROI within the volume data from the 2D screen. First, the user mouse clicks on the region of interest and drags the mouse in order to define the size of their region of interest. The mouse click defines the initial center of the ROI.

Once the ROI is determined, a ray profile is obtained and analyzed. The initial center of the ROI is then repositioned from the original location to a new locally optimal location based on analysis of the ray profile in the area selected and the user selected datatype. Based on the datatype, the ROI is centered at either the maximum gradient magnitude or at the maximum scalar value in the 3D neighborhood. Then, the user can shift the default position of the ROI along the viewing ray, allowing him or her to acquire a different ROI along this focus line.

The repositioning and analysis of the ray profile is the main step in our semiautomated data exploration. Since features can be identified with local peaks and maximum gradient magnitudes obtained by sampling along the ray at regular intervals, we employ the concept of a feature layer to generate the visual mapping based on these local peaks and maximum gradient magnitudes. Each layer corresponds to the minimal pairwise spatial distance calculated between peaks along the ray profile. A feature layer stores the minimum and maximum of the local scalar values and the minimum and maximum of the local gradient magnitudes to determine a value range for the color and data enhancement within each layer. Depending on the type of volumetric data (interval or ratio as defined by the user), the maxima and minima of these values are used to determine the transition points (as shown in the value profile widget of Fig. 3 and detailed in Fig. 4) that divide the profile into multiple layers.

We first extract feature layers based on the local extrema along the ray profile and then use these feature layers for data exploration. Malik et al. [22] employed such a ray profile for feature peeling. Multiple rays are cast to extract scalar values along the rays. This approach allows the users to explore volumetric data sets layer-by-layer. We also use the variation of scalar values along the rays to extract feature layers. In addition, we use the local peaks, both from scalar values and gradient magnitude, of each layer to provide color assignment and data enhancement for features of interest in layers according to the data type of the input volume data set. Thus, each layer is found

Fig. 4. The larger color bar on the top displays the scalar (black) and gradient magnitude (red) value profiles. Three vertical black lines indicate the positions of transition points that create four feature layers. The vertical bar on the right is divided into four regions bounded by the scalar value ranges of the corresponding colors in the color map shown in the color bar below.

(a) (b)

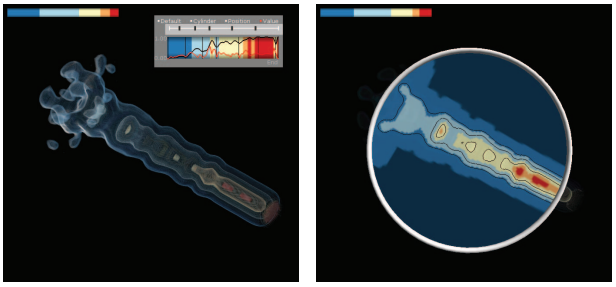
Fig. 5. Value profile of a noisy MRI data. (a) Original value profile of the scalar values and (b) same profile filtered with a median low pass filter.

between two extrema of either the scalar or gradient values along the ray profile. The ordering of the layers by scalar value is done so that the correct color scheme will be mapped to each layer. As shown in Fig. 4, the layers are perceptually ordered by color; however, the ordering of the layers on the ray profile would not match the perceptual color ordering without this step.

In the color selection stage (Section 3.2.1), we assign a unique color to each identified layer. In our work, we utilize the Color Brewer [33] color schemes and match each scheme to the appropriate data type (nominal maps to qualitative, interval to sequential, ratio to divergent, and ordinal to sequential). However, in some cases, we may not be able to extract the necessary number of layers either due to the difficulty in determining feature layers (for example, in the case of gradually increasing scalar values) or due to the presence of excessive layers. To overcome this, we add a user-defined parameter to specify the number of feature layers, i.e., the number of colors, to use in these cases. When the number of extracted layers does not exceed this user-defined parameter, we iteratively sort the layers based on their value range. Then, we select the largest layer, and split it into two until the number of layers is equal to the user-specified parameter value. When the number of extracted layers exceeds this user-defined parameter, we automatically merge layers with similar local peaks until the number of layers is equal to the specified parameter value. The layers are sorted in increasing order of the maximum peak values, the peaks of two consecutive layers are compared and the two layers are merged into one if the distance between the two peaks was the smallest distance between all other peaks.

In the presence of noisy volumetric data, the problem of extracting layers from volume profiles is further compounded. In our data exploration and transfer function design scheme, we use contour lines to represent significant value ranges in the region of interest. Unfortunately, contour lines for noisy volumetric data sets contain many artifacts which can confuse the end user. Our work follows the contour conventions proposed by Viola et al. [34] while providing a more general interface for generating contours while removing noise. Our system uses nonlinear filters, parallelized using CUDA [6], and we compare the quality of the resulting data sets in Section 4. Fig. 5a shows a noisy value profile, here one can imagine that the number of transitions found would result in a large number of nonessential layers. As such, we utilize nonlinear filters such as the median filter that are used to smooth such value variations [22] as shown in Fig. 5b. This smoothes the ray profile and allows for easier feature layer extraction.

The denoising is only done on the ray profile and the slice view. The original data are never denoised, so the output of the pipeline is only slightly different in the noisy versus nonnoisy data. This is due to the fact that a user specifies a maximum number of layers, and layers found during the computational analysis that are only at a small distance from each other will be merged. In the case of noisy local peaks, the local peaks typically become merged. The denoising is only needed for the contour views.



(a) (b)

Fig. 6. Initial rendering of a *fuel simulation* data set with the default parameter settings. (a) Shows the changes of scalar values (in black on the top right) and gradient magnitudes (in red) and the regions enhanced based on this profile. (b) Shows the contour lines generated automatically to convey the shapes within the region of interest.

3.2 Transfer Function Generation

In our approach, the value profile and contour lines are used to provide an interactive environment for data exploration. Fig. 6 illustrates the initial data visualization based on our value profile and contour lines using default color and opacity selection. The line graph on the top right in Fig. 6a shows the scalar and gradient value variations from the value profile. Fig. 6b shows the shapes of the volumetric objects with contour lines in the region of interest. Based on the feature layers, the number of colors and emphasized areas are computed. The color bar on the top left of the figure shows the color transfer function. A cool color is selected as an outer color and a warm color is selected as an inner object using the Color Brewer's diverging color scheme (Red-Yellow-Blue). The user can change the colors directly on the slice view. The ticks above the line graph indicate important features (maximum intensities in each local area) to be enhanced. The user can add or remove ticks for data enhancement.

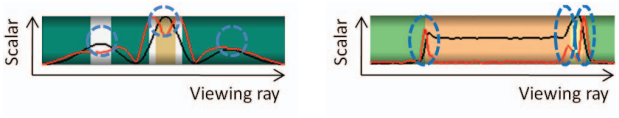
3.2.1 Color Selection

We assign a unique color to every layer identified during the ROI acquisition phase. Depending on the data type, we use Color Brewer's color schemes (sequential, diverging, and qualitative) by default as listed in Table 1. Fig. 4 shows the assignment of a color map with four colors to a value profile with four layers. The scalar range of each color in the color map is obtained by computing the mean value of each set of adjacent peak values as marked on the vertical line on the right in this figure.

Ideally, the number of colors to be used can be determined from the number of layers. The process of merging similar layers and specifying the number of desired layers in the feature extraction step prevents the assignment of different colors to similar layers and generates an appropriate color transfer function.

3.2.2 Data Enhancement

Our method utilizes color to define a feature; however, feature enhancement is also an important setting for transfer function design. There are several approaches to



(a) (b)

Fig. 7. Enhancement regions identified from the value profile of a region of interest based on the data type. (a) Local peaks of each layer are enhanced for interval data. (b) Local maximum gradients of each layer are enhanced for ratio data.

data feature enhancement. We modulate the opacity values to enhance the data features based on the extracted critical values. As a default opacity setting, we apply a sine function to map the scalar data value to opacity, which creates alternating low and high opacity contours [31], shown on the left side in Fig. 2.

However, different data types have different characteristics that require the emphasis of appropriate regions in the value profile as shown in Fig. 7 and Table 1. For example, ratio data are characterized by the difference of values rather than the sequence of values. Therefore, the boundaries of each layer are more important and need to be emphasized instead of the peaks of each layer. Fig. 8 illustrates the difference between enhancing local peaks and local maximum gradients for a ratio data set (the *tooth* data set).

For the data enhancement, the ranges of local peaks and maximum gradient magnitudes obtained from the value profile are maintained in a weight map and the weights are multiplied by initial opacities to obtain final opacities. We employ a 1D weight map to enhance the features of interest (e.g., local maximum value and local maximum gradient magnitude). Local maximum gradient magnitudes are usually used for the boundary enhancement [35]. However, simulation data sets (e.g., the probability distribution of atoms) tend to have smooth data variations. Therefore, we use the local maximum value for simulation data sets as the default setting, as shown in Fig. 7a. In Kindlmanns approach, the boundaries of objects are defined by sharp curves in the histogram. By utilizing the ray profile, the changes in density along the profile allow us to determine transitions between features that are not necessarily visible in the histogram (as shown in Fig. 1).

The weight map assigns a scalar value scaled from zero to one for each voxel as the voxel's weight. By default, the initial values for all the density bins are set to 0.2. The empirical value 0.2 deemphasizes the high opacities from the default opacities. Typically, the maximum scalar value of a layer and the maximum gradient magnitude should be emphasized in the case of sequential data sets (e.g., simulation data) and ratio data sets (e.g., medical data), respectively. We compute a value range $[\alpha_i, \beta_i]$ to be highlighted for each peak by obtaining the minimum and maximum of scalar values of the N closest neighboring sampling points of the ith peak along the ray. We set the value of N to 7 in this paper.

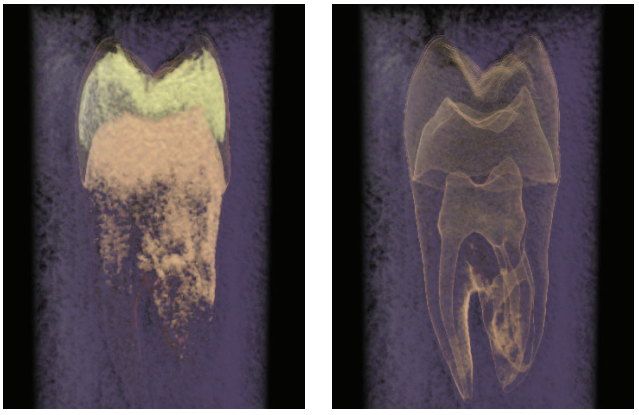
The weight map is updated using

$$w_j = \begin{cases} 1, & \alpha_i \le v_j \le \beta_i, j \in [0, \text{num of bins}), \\ 0.2, & \text{otherwise}, \end{cases} \quad (1)$$

where w_j is the jth weight in the ray profile histogram and v_j is the voxel scalar value. However, even weighted opacities cannot be used to display the values with zero opacity, as shown by Zhou et al. [36]. Thus, we use (2) to compute the opacity for the range of focus values

$$\alpha_{vd} = \max\{\alpha_o, \alpha_w\} \times w_j, \quad (2)$$

where, α_{vd} is the enhanced opacity, α_o is original opacity, and α_w is the opacity of the windowing function as defined in [36]. Values obtained for the weight mapping and nearest neighbor sampling were obtained through visual analysis of repeated trials working with end users. Future work will focus on determining automatic optimal parameter settings.



(a) (b)

Fig. 8. Images showing the differences arising from enhancing different regions of the *tooth* data set. (a) Image obtained by enhancing local peaks of each layer. (b) Image obtained by enhancing local maximum gradients of each layer.

Additionally, we apply focus-region-based data enhancement along a ray cast through the volume data. For this enhancement, we assume that the region is cylindrical or spherical. For spherical regions, the center point (C) of the focus region and the radius (r) of the ROI are used to compute the weighted opacities, (5). Data enhancement is performed based on the distance between the position of a voxel and the center point of the focus region.

For cylindrical regions that emphasize the information around the focus line, the two end points (x_1, x_2) of a focus line, a radius (r) of a focus region, and the distance (d) between a sample position (v_p) and the line (x_1, x_2) are used for the computation of weighted opacities. The opacity values are computed according to their shape. For a cylinder, the opacity is computed as

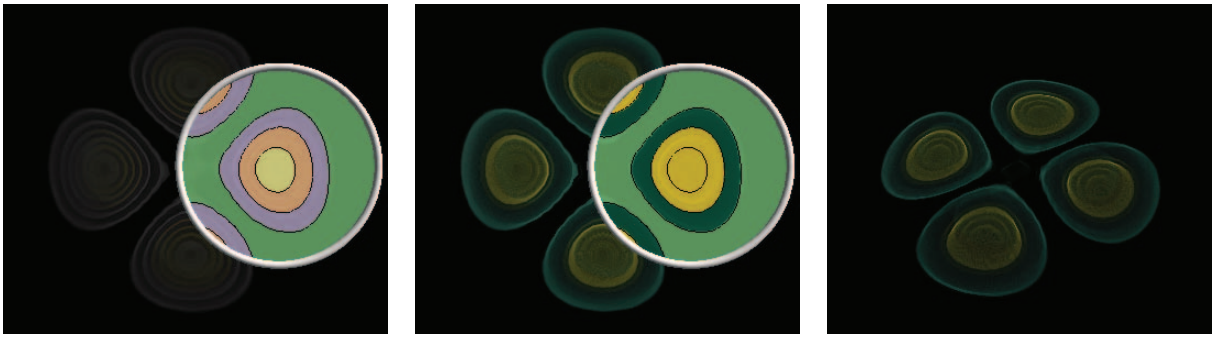
$$\alpha_{rd} = \alpha_{vd} \times \left(k + \left(2 - \frac{d}{r}\right) \times I_c(v_p)\right), \quad (3)$$

$$I_c(v_p) = \begin{cases} 1, & d \le r, \\ 0, & \text{otherwise}. \end{cases} \quad (4)$$

For a sphere

$$\alpha_{rd} = \alpha_{vd} \times \left(k + \left(2 - \frac{|v_p - C|}{r}\right) \times I_s(v_p)\right), \quad (5)$$

$$I_s(v_p) = \begin{cases} 1, & |v_p - C| \le r, \\ 0, & \text{otherwise}, \end{cases} \quad (6)$$



(a) (b) (c)

Fig. 9. User-defined color selection on a slice view displaying contour lines. (a) Shows a slice view. (b) Shows the resulting visualization after manually selecting colors and enhancing features of interest. (c) Shows the final resulting image.

where α_{vd} is the enhanced opacity from (2), α_{rd} is the resultant opacity, k is a constant (in this paper, we use $k = 0.5$ for all images for focus-region-based data enhancement), v_p is a sample position, and $I_c(v_p)$ and $I_s(v_p)$ are indicator functions that identify whether a voxel is within the region of interest or not.

3.2.3 Contour Line Extraction

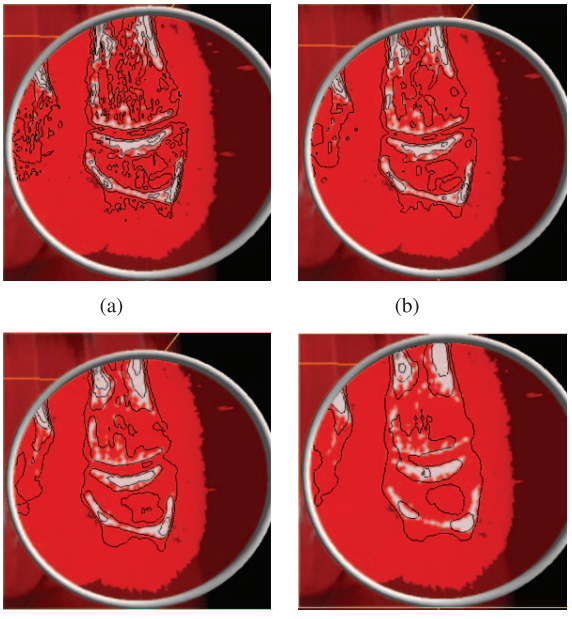
Using a transfer function design interface to highlight a region on the slice view or change the color of the region is often unintuitive as there is no direct link between the slice view and the transfer function interface. In order to better guide users, our system employs the use of contour lines in the slice view to provide intuitive cues for ROI selection. Contour lines have been shown to be effective in non-photorealistic rendering to convey shape information [37] and can be used to illustrate value regions or shapes of simulation data. Moreover, contour lines on a slice view in the region of interest provide intuitive clues to select and enhance the region of interest. As such, we extract and visualize contour lines on the slice view to allow users to change visual properties of the region of interest as well as to convey shape in the selected regions of interest. Users can control the number of level sets by adjusting the sampling space of the contour lines as well as the number of the extracted feature layers. Whenever a user changes the position of the ROI, the view on the slice plane is updated and contour lines are regenerated. We extract contour lines from the image in the slice view to illustrate the shape of the volumetric objects based on the critical boundaries of the data in the regions of interest. Contour lines are extracted using the Marching Square algorithm [38] instead of image-based edge detection algorithms since 3D line primitives obtained from the former can be regenerated and displayed clearly regardless of the magnification and camera position unlike the image-based methods. A 2D slice of the user-specified size is placed and moved along the ray and the slice captures the values and stores them to a render target. The color buffer is then used to extract contour lines.

The number of the level sets of the contour lines is computed and isovalues ranging from 0 to 1 (where the value is internally scaled) are determined for each level to generate contour lines based on the extracted layers from the value profile. For volumetric data with no noise, the contour lines can be directly generated to show critical boundaries as shown in Fig. 6b. Moreover, we also allow users to select and assign colors to various regions formed by the contour lines on the slice view plane. Fig. 9 shows the user-defined color selection using a slice view.

Unfortunately, noisy volumetric data produce many contour lines, as shown in Fig. 10, that clutter the display, reducing their utility in conveying shape information. Therefore, we apply median and bilateral filters to the image of a slice view to remove the noise before generating contours. We describe the implementation details of contour line generation in the next Section.

4 Implementation

We implemented the data exploration pipeline on a Windows XP PC with an Intel Pentium 4 3.40 GHz CPU, 2 GB RAM, and a GeForce 8800 GTX graphics card. The median filtering with a 3×3 window and contour line



(a) (b) (c) (d)

Fig. 10. Images of the *foot* data set rendered by a median filter with various sizes. (a) Shows contour lines when no filters are applied. (b), (c), and (d) Show the contour lines extracted from the image filtered by a 3×3, 5×5, and 9×9 size median filter, respectively.

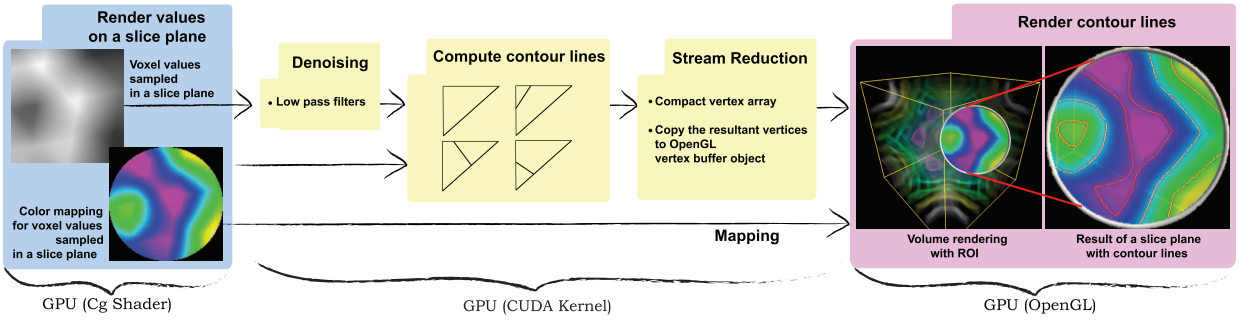


Fig. 11. Pipeline for contour line generation using the CUDA kernel. The first step, implemented in the Cg shader, generates a texture containing scalar values sampled from the volumetric data, which is then mapped to colors on the slice plane. The second step filters noisy data sets to remove noise. The third step computes contour lines and performs stream reduction on the CUDA kernel and the final step renders the computed contour lines.

generation takes 5.7-7.8 and 9.9-10.8 milliseconds for 128×128 and 256×256 grid textures on the GPU, respectively.

This section describes the implementation details of the key aspects in our data exploration pipeline namely value profiling, slice view generation, contour line generation, and image denoising.

Value profiling. To obtain a value profile from the user's mouse click in the ROI, we compute two points, one each on the near and far planes of the view frustum using gluUnproject. These two points, determined using the ray and bounding box intersection test, define the line segment for the value profile. To sample scalar and gradient magnitude values along this ray, sampling points are generated from the starting point to the ending point of the line segment and are stored in a 1D RGB component texture. A rectangle is rendered with this texture to pass the sampling points to the graphics pipeline. In a pixel shader, each texel is used as a sampling position to obtain the scalar value and the gradient magnitude at the corresponding position in the 3D volume texture. The sampled scalar value and the 3D gradient magnitude are stored in the render target. For data sets with noise, scalar, and gradient magnitude values are filtered with a median filter using the host CPU due to the low computational cost. The sampled values are copied to the host memory in order to compute the initial center position of the ROI as well as the feature layers.

Slice view generation. The pipeline for the slice view produces two color buffers in the render target. One stores the scalar and gradient magnitude values for the contour line generation and the other stores the color (RGB) of each scalar value and the transparency (Alpha) to display in the slice view. Images are generated from the slice view by treating it as a circle, and an orthographic camera facing this plane is used to sample values within the circle of the focus region. Regions outside of the circle are rendered as translucent by adjusting the alpha component as shown in Fig. 11.

Contour line generation. Level sets of the contour lines are determined based on the feature layers obtained from the value profile. Fig. 11 illustrates the contour line generation pipeline. The image (2D texture) is treated as a grid and contour lines are computed using the Marching Squares algorithm [38]. However, for better memory efficiency, triangle primitives are used instead of squares because each test using triangles produces zero or one contour line, while the Marching Square test produces zero to three contour lines. It reduces the memory consumption during the stream reduction operation [39], [40] while the number of iterations over the CUDA kernel becomes twice as many as the level sets. This doubling is due to the fact that the contour lines are extracted from triangles (two triangles per a grid cell). This tradeoff in using triangle primitives over squares is justified because efficient memory consumption is of greater concern than computation time in most web-based portals, such as NanoHub [3], that have to support multiple simultaneous users.

Each triangle primitive is assigned to a thread, and each thread generates zero or one line, as shown in the third module (Contour lines generation) in Fig. 11, and the number of lines is stored in the *flag* in Fig. 12. The fourth step produces a scattered array of line vertices, *lines (scattered)* in Fig. 12. Unwanted elements (no line primitives) from the scattered array are removed by applying stream reduction (using the CUDA Data Parallel Primitive (CUDPP) library [40]) and the resultant array becomes compact like *lines (compact)* in Fig. 12. An OpenGL buffer object is created to store all the line primitives since the CUDA kernel and the OpenGL pipeline can both access the OpenGL buffer. The number of contour lines computed in each iteration is obtained from the stream reduction index array and is used as the offset into the memory block to obtain the memory starting point of the next iteration. Using the CUDA architecture, contour lines of five level sets are generated within 3-4 milliseconds for a 128×128 grid texture on the GPU.

Image denoising. Noisy volumetric data sets result in messy contour lines as shown in Fig. 10. To avoid this, the

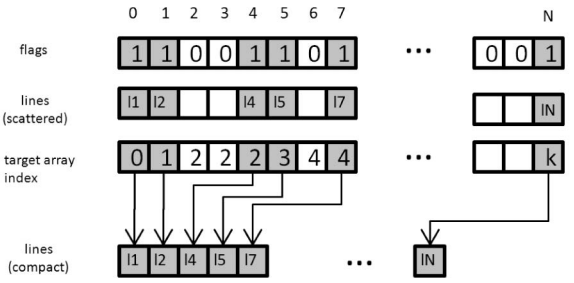


Fig. 12. A schematic diagram of the reduction for contour line generation. *flags* stores the flags that indicate each triangle has a line. *target array index* stores indices for lines to be stored in the compact array of lines. Finally, *line*, the mapped OpenGL VBO, stores compact lines.

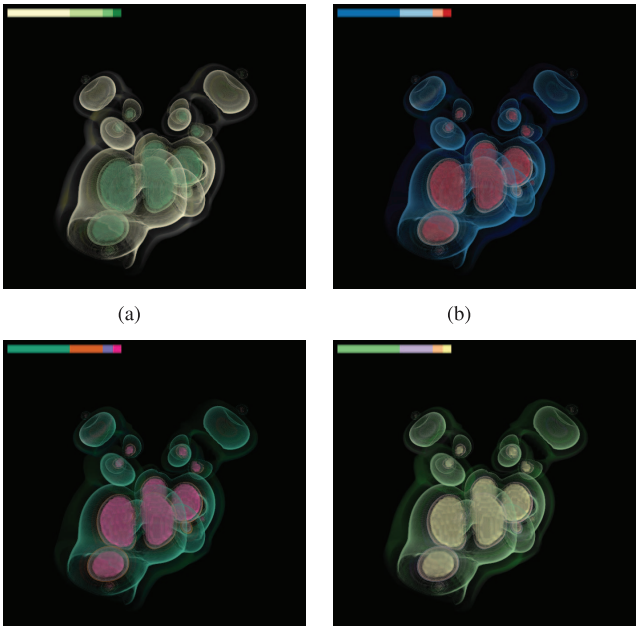


Fig. 13. Images from the *neghip* data set rendered using various color schemes provided by the Color Brewer [33]. (a) Uses the Yellow-Green sequential color scheme. (b) Uses the Red-Yellow-Blue diverging color scheme. (c) and (d) Use the Dark2 and the Accent qualitative color schemes, respectively.

image obtained from the slice view is filtered using a median filter or a bilateral filter. The median filter has been implemented based on the HDR Image Processing Library [41] and designed for the CUDA kernel. In our implementation, a 3×3 median filter is applied to the slice view image by default. However, users are given the option to change both the filter and its parameters.

5 Results and Discussion

During the design and evaluation process of our work, we have collaborated with computational nanotechnology and flow simulation researchers. In fact, the need for a simplified interface for volume rendering on nanohub.org was the impetus for this work. Based on our initial success with computational nanotechnology data, we have applied the pipeline to explore a variety of data sets including ratio and interval data sets (both with and without noise). In this section, we present a gallery of results highlighting various aspects of our data exploration pipeline and its ability to extract important features in various data sets as well as feedback from our various user constituents.

5.1 Nonnoisy Volumetric Data

Fig. 6a shows the results of applying the value profiling and data enhancement aspects of our pipeline to a *fuel simulation* data set. The value profile shows that important regions within the data are occluded by outer layers from a specific viewpoint. Our pipeline automatically extracts four layers based on the value profile and the peak value of each layer is enhanced. This enhancement, combined with the default diverging Color Brewer scheme, allows a user to easily identify important parts previously occluded within the data set. Based on the value profile, four layers are extracted and the peak values of each layer are enhanced. The number of colors is equal to the number of feature layers. Various Color Brewer schemes can be applied to a data set depending on its type. Fig. 13 shows different color schemes applied to the *neghip* data set ranging from a sequential color scheme Fig. 13a to a diverging scheme in Fig. 13b and qualitative color settings in Figs. 13c and 13d.

Automatic parameter specification in our pipeline is especially useful for scientists who are unfamiliar with volume rendering and visualization techniques. However, our pipeline also supports advanced visualization users by allowing them to tweak rendering parameters such as the color, opacity, and the number of colors.

Fig. 14 shows different rendering results using traditional 1D and 2D transfer function widgets and our widget. We worked with a nanoscientist who had previously been working with volumetric rendering tools for analyzing quantum dot simulations. This scientist was familiar with 1D and 2D transfer functions, and using these traditional tools, the user was able to visualize and highlight structures within the data set. In the 1D transfer function space, Fig. 14a, the scientist was able to explore the scalar data values and create a suitable transfer function; however, he reported that the automatic parameter settings provided a quicker and better rendering in terms of his

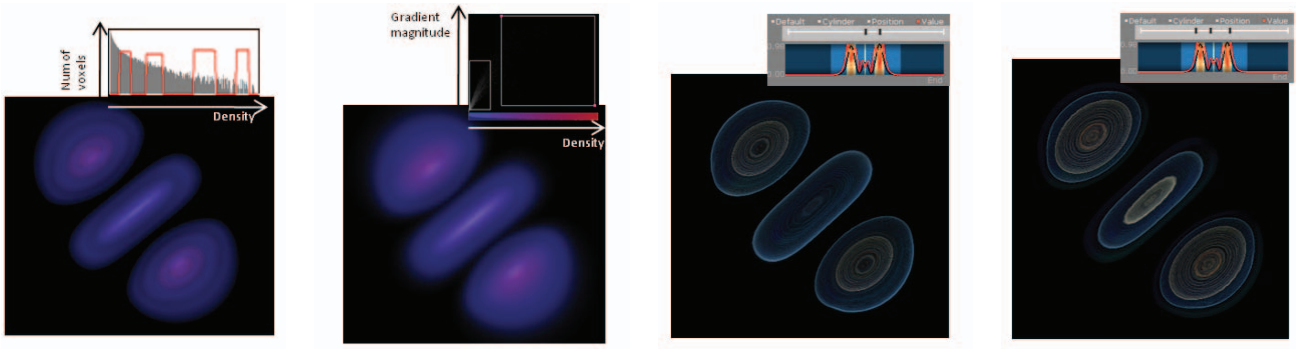


Fig. 14. Plots of a wave function using a 1D histogram widget, a 2D histogram widget, and our data exploration widget. (a) The result using the 1D histogram. (b) The result using the 2D histogram. Most of the values are binned within the left small box, but the area does not have any inner structure within it. (c) The result with the local maximum highlighted. (d) The result with the local maximum gradient magnitude highlighted.

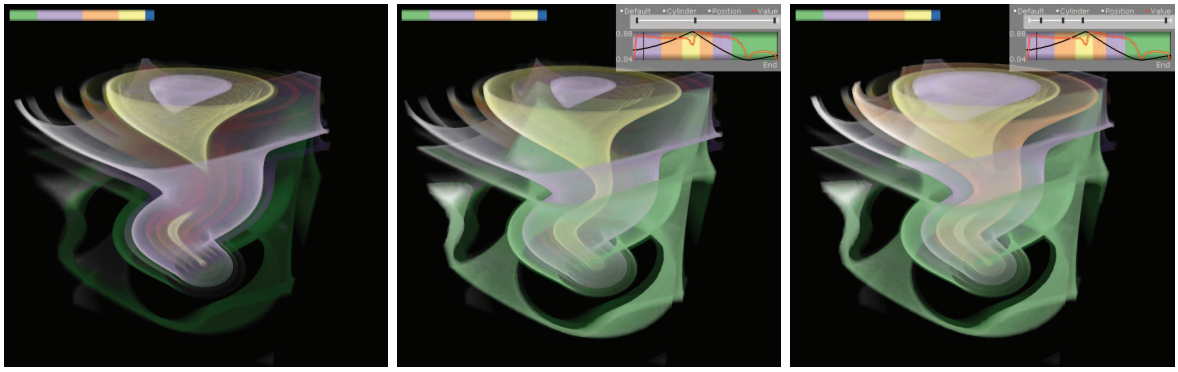


Fig. 15. Results of a *tornado* data set rendered by our data exploration method. The left image is a result of enhancing local peaks in the region of interest, the middle image shows a result from enhancing local maximum gradient magnitudes, and the right shows a resulting image from user-defined data enhancements.

analysis. In the 2D transfer function case, Fig. 14b, since the important structure within the simulation data is sparse, the upper right portion of the 2D transfer function histogram appears to contain no data; however, this region needs to be selected to create an appropriate visualization. In this case, the user had difficulty in determining the correct place to search for data and utilized a hunt and peck approach until finally settling on an image with inner structures of the data blurred out. By using our method, Figs. 14c and 14d, the scientist reported that he was able to reduce his search time and gain insights about the changes in wave functions within his data. According to the user, Fig. 14d is better than both Figs. 14a and 14b as the user was able to better highlight the inner structure of the data and gain an understanding of the electron potential fields within the data.

Figs. 14c and 14d show the difference between emphasizing local scalar maxima and local gradient magnitude maxima, respectively. In comparing Fig. 14c to 14d, the user found both images to be useful. Fig. 14c provided the scientist with a better understanding of the electron potential clouds, while Fig. 14d provided more details about the inner structures of the data.

Informal feedback from computational nanotechnology researchers and computational flow researchers has been very positive. These scientists find this system interface more effective and intuitive for exploring, analyzing, and understanding the features in their simulation data than existing interfaces. In terms of the usefulness, this tool helped them to better understand the distribution of the wave functions within the quantum dot area. From the line plot (value profile) that cuts through the center of their simulation data (e.g., quantum dot), they could see their simulation result (e.g., the wave function data) along critical directions in real space and directly highlight features of interest. In addition, they told us that this helped them in performing quantitative analysis which was absent in previously available visualization tools that only provide qualitative analysis. Previously, the scientist had developed scripting programs (e.g., MATLAB programs) to perform quantitative analysis by generating line plots in critical directions. Finally, our system also helped them calculate optical matrix elements in various directions for their simulation. While obtaining this informal feedback, we also provided the end user with commonly used viewing directions (e.g., top, front, and $+-45$ rotation views) to help better extract the value profile. We found that the addition of these viewing angles were also a very popular feature and were able to further reduce the amount of time needed to analyze and visualize a data set.

The usefulness (to the scientist) was in the reduced amount of exploration time needed to generate the image (note that without the semiautomatic approach the transfer function needs 4 peaks (Fig. 14a). The addition of the ray profile tool provided them with a means to perform quantitative analysis that was previously ported to other software tools. Thus, the addition of the tools for the semiautomatic generation and quantitative analysis are able to both reduce the burden of transfer function creation on the user while enhancing their overall analytic capability.

As such, our data exploration method more closely couples the physics governing the data and techniques for mapping this data using a judiciously chosen transfer function. Allowing the tighter integration of the data with the transfer function provides a more intuitive interface for manipulating parameters present in the data itself and is one of the key strengths of the method. Because users are allowed to interactively manipulate the data and data gradients and see an illustration on the screen, they get a much more intuitive notion of how to best communicate and understand the physics under consideration.

5.2 Time-Varying Data

We further applied our methodology to the investigation of fluid flow. Fig. 15 shows a *tornado* data set and compares results obtained using the default and user-defined parameter settings. Fig. 16 shows the resultant images obtained by setting rendering parameters to extract interesting regions and track them in the *convection* data set. These images show a time advection of the flow in this data set. Informal feedback from a computational fluid dynamics expert also yielded positive results. This researcher indicated that by using our visualization techniques, the convection layers can clearly be identified. This allows the user to better understand the time-varying behavior of this complex data, allowing for better identification of salient structures. With respect to the convection in a box, Fig. 16, the user is able to clearly see the nested structure of the flow

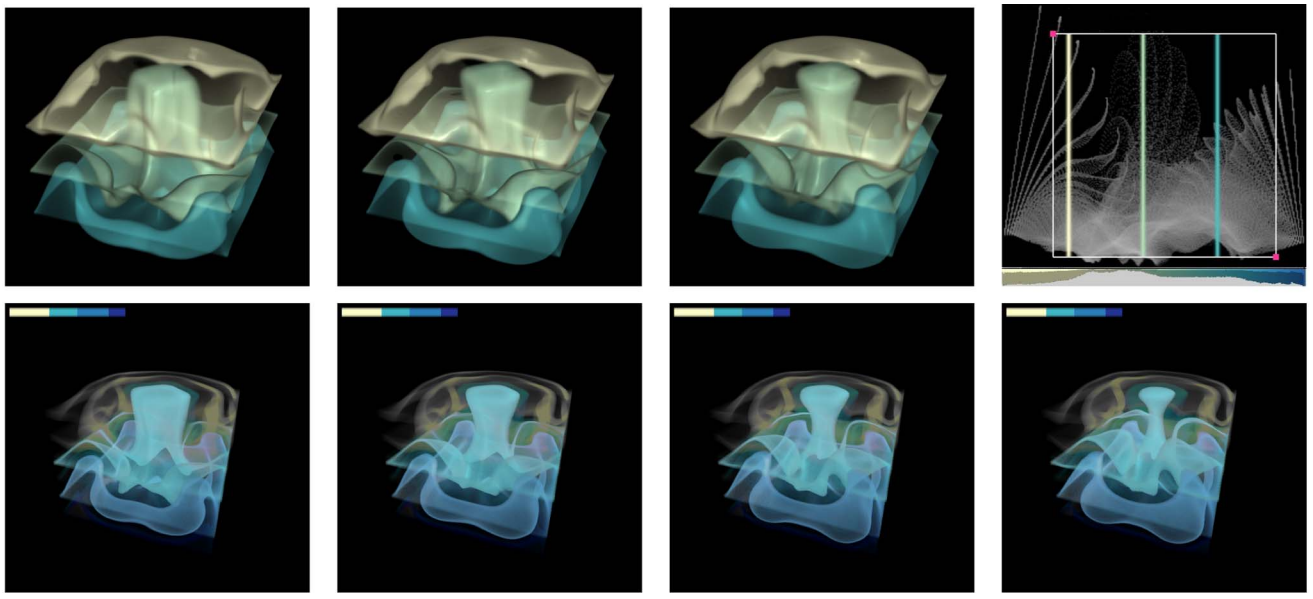


Fig. 16. Results of our data enhancement technique applied to visualize temporal advection in a convection simulation. The series of images (Top and Bottom) show every second sequential time step starting from 220. (Top) Using the sinusoidal transfer function method from Svakhine et al. [31]. (Bottom) Our semiautomatic transfer function generation procedure. Note that the layers in the bottom image visualize several different structures of the flow, particularly near the top portions of the image.

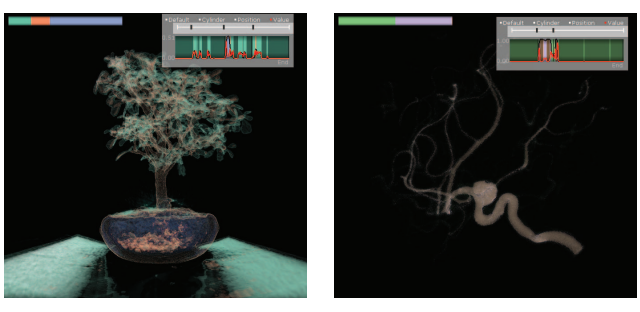
gradients. By allowing users to interactively investigate and pinpoint boundaries of interest, we can create visualizations that show global structures without occluding regions of interest that are limited to a more local region of interest.

We also applied our methods to the tornado data. Fig. 15 shows the nested structure of the flow and enables users to pinpoint the structure of the gradients. Visualizing flow in this manner is a powerful means for viewing these nested structures and how they change over time. Note that for all temporal data sets, a single time step was used to calculate the visual parameters. As such, a static transfer function is used for all time step renderings in order to keep coherency between the mapping of color and opacity to a particular scalar or gradient value. Depending on the time step chosen to generate the transfer function, occlusion of structures in future time steps may occur. By keeping the color mapping coherent (i.e., for all time steps, the same color maps to the same scalar value), as the data change, some structures may be lost. As such, the user could recolor the data based on the ray profile at any time step. Future work will focus on solutions to this issue of the tradeoff between color coherency and the occlusion of structures.

5.3 Noisy Data

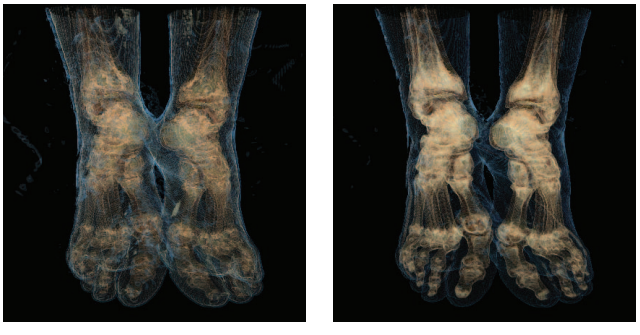
While our previous examples showed smooth data sets, noisy data sets require further user intervention to generate useful results. Other parameters, such as the median window size, can be modified by the user to interactively select an optimal size that generates the best results. This is demonstrated in Fig. 10 where a noisy data set is rendered without any filtering in Fig. 10a. Application of the 3×3, 5×5, and 9×9 size median filter yields better results as shown in Figs. 10b to 10d. However, as we increase the window size, the rendered image starts losing important feature lines. Therefore, a default size of 3×3 is adopted in our implementation.

Fig. 17 shows another example with feature boundaries enhanced by our default rendering parameters. While these initial results are good at showing overall feature boundaries, in some cases, the default rendering parameters will not produce a clear visualization. This is illustrated in Fig. 18a. Here we show a rendering example using noisy data where the default rendering parameters are suboptimal. In this case,



(a) (b)

Fig. 17. Images of (a) *bonsai* data set. (b) *aneurism* data set rendered with default settings showing enhanced boundary areas.



(a) (b)

Fig. 18. Images of the *feet* data set rendered (a) with the default system-generated settings, and (b) after the user adjusts the emphasis points.

the user may edit the transfer function to obtain better results, as seen in Fig. 18b.

We explored both Median and Gaussian filters using guided exploration with expert users. Users expressed that they could tell no difference between the applied filters and were likely to simply use the default values in the cases where filtering is needed.

6 Conclusion and Future Work

In this paper, we have presented a data exploration method for volumetric rendering using value profiles and contour lines. We devised methods to automatically provide default rendering parameter settings based on the features in the region of interest. This technique is applicable to interval and ratio data types with or without noise to support a variety of volumetric data sets, such as medical data and scientific simulation data. Based on the data types, we extract feature layers from the data based on a value profile and provide contour lines from the obtained feature layers. The extracted contour lines help to illustrate the shape of the data set in the region of interest.

To support data sets with noise, we denoise the value profile and the image captured on a slice view by filtering them with a feature preserving nonlinear filter (median or bilateral filter). We also modulate the opacities based on the feature layers to emphasize boundary features and maximum intensities in the region of interest. Moreover, we parameterized a weight map, the radius of an ROI, and a focus line.

We plan to include the other two data types (nominal and ordinal) [42] and add further automation to the system by providing users with an optimal viewpoint that can show the important data features by default [43]. Further, we also plan to extend our data exploration approach to support multidimensional transfer functions. Since a slice view has more sampled voxels, 2D ROI specification based on slices will be useful to generate a local feature-driven 2D transfer function using nonparametric clustering algorithms (e.g., kernel density estimation) [28]. Illustrative visualization techniques have been used to convey important features of data sets by abstracting away unnecessary details [18], [37]. We would like to incorporate some of these techniques to improve users' perception [44] of the extracted feature layers. Also, since scientific simulations typically return multivariate data sets, we plan to extend our technique to look at the ray profile across several variables at once. This will generate a variety of layers, and exploration will be done on ways to merge layers for enhanced visualization.

Acknowledgments

This work has been supported by the US Department of Homeland Security's VACCINE Center under Award Number 2009-ST-061-CI0001 and the US National Science Foundation (NSF) under Grants 0328984, 0121288, and 0906379.

References

[1] C. Catlett, W.E. Allcock, P. Andrews, and R. Aydt, "TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications," *Proc. HPC and Grids in Action,* 2007.

[2] The Open Science Grid Consortium, http://www.opensciencegrid.org/, 22 Apr., 2011.

[3] nanoHUB.org, http://www.nanohub.org/ 22 Apr., 2011.

[4] G. Klimeck, S.S. Ahmed, N. Kharche, M. Korkusinski, M. Usman, M. Prada, and T. Boykin, "Atomistic Simulation of Realistically Sized Nanodevices Using Nemo 3D Part II: Applications," http://nanohub.org/resources/3825, 22 Apr., 2011, Jan. 2008.

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

[6] NVIDIA Corporation, "NVIDIA CUDA Compute Unified Device Architecture," http://www.nvidia.com/object/cuda_home_new.html, 22 Apr., 2011, Nov. 2007.

[7] J. Kniss, G. Kindlmann, and C. Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets," *Proc. Conf. Visualization,* pp. 255-562, 2001.

[8] J. Marks, B. Andalman, P.A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, "Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation," *Computer Graphics,* vol. 31, pp. 389-400, 1997.

[9] M. Tory, S. Potts, and T. Möller, "A Parallel Coordinates Style Interface for Exploratory Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 1, pp. 71-80, Jan./Feb. 2005.

[10] C. Rezk-Salama, M. Keller, and P. Kohlmann, "High-level User Interfaces for Transfer Function Design with Semantics," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1021-1028, Sept./Oct. 2006.

[11] Y. Wu and H. Qu, "Interactive Transfer Function Design Based on Editing Direct Volume Rendered Images," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 5, pp. 1027-1040, Sept./Oct. 2007.

[12] S. Bruckner and M.E. Gröller, "Style Transfer Functions for Illustrative Volume Rendering," *Computer Graphics Forum,* vol. 26, no. 3, pp. 715-724, Sept. 2007.

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

[14] C. Lundström, A. Ynnerman, P. Ljung, A. Persson, and H. Knutsson, "The Alpha-Histogram: Using Spatial Coherence to Enhance Histograms and Transfer Function Design," *Proc. Eurographics/IEEE-VGTC Symp. Visualization,* pp. 227-234, 2006.

[15] C. Correa and K.-L. Ma, "Size-Based Transfer Functions: A New Volume Exploration Technique," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1380-1387, Nov./Dec. 2008.

[16] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman, "The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering," *Proc. IEEE Conf. Visualization,* pp. 367-374, 2005.

[17] A. Lu, R. Maciejewski, and D.S. Ebert, "Volume Composition Using Eye Tracking Data," *Proc. Eurographics/IEEE-VGTC Symp. Visualization,* pp. 115-122, 2006.

[18] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin, "Interactive Cutaway Illustrations of Complex 3D Models," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 31-40, 2007.

[19] I. Viola, A. Kanitsar, and M.E. Gröller, "Importance-Driven Feature Enhancement in Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 4, pp. 408-418, July-Aug. 2005.

[20] T. Ropinski, J.-S. Praßni, F. Steinicke, and K.H. Hinrichs, "Stroke-Based Transfer Function Design," *Proc. IEEE/EG Int'l Symp. Volume and Point-Based Graphics,* pp. 41-48, 2008.

[21] C. Rezk-Salama and A. Kolb, "Opacity Peeling for Direct Volume Rendering," *Computer Graphics Forum,* vol. 25, no. 3, pp. 597-606, 2006.

[22] M.M. Malik, T. Möller, and M.E. Gröller, "Feature Peeling," *Proc. Graphics Interface,* pp. 273-280, 2007.

[23] C.D. Correa and K.-L. Ma, "Visibility-Driven Transfer Functions," *Proc. IEEE Pacific Visualization Symp.,* pp. 177-184, 2009.

[24] P. Kohlmann, S. Bruckner, A. Kanitsar, and M.E. Gröller, "Contextual Picking of Volumetric Structures," *Proc. IEEE Pacific Visualization Symp.,* pp. 185-192, 2009.

[25] H. Akiba and K.-L. Ma, "A Tri-Space Visualization Interface for Analyzing Time-Varying Multivariate Volume Data," *Proc. Eurographics/IEEE-VGTC Symp. Visualization,* pp. 115-122, 2007.

[26] H. Akiba, K.-L. Ma, J.H. Chen, and E.R. Hawkes, "Visualizing Multivariate Volume Data from Turbulent Combustion Simulations," *Computing in Science and Eng.*, vol. 9, no. 2, pp. 76-83, Mar.-Apr. 2007.
[27] C. Muelder and K.-L. Ma, "Interactive Feature Extraction and Tracking by Utilizing Region Coherency," *Proc. IEEE-VGTC Pacific Visualization Symp.*, pp. 17-24, Apr. 2009.
[28] R. Maciejewski, I. Woo, W. Chen, and D. Ebert, "Structuring Feature Space: A Non-Parametric Method for Volumetric Transfer Function Generation," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1473-1480, Nov./Dec. 2009.
[29] C. Ware, *Information Visualization: Perception for Design.* Morgan Kaufmann, 2004.
[30] S.S. Stevens, "On the Theory of Scales of Measurement," *Science*, vol. 103, no. 2684, pp. 677-680, 1946.
[31] N. Svakhine, Y. Jang, D. Ebert, and K. Gaither, "Illustration and Photography Inspired Visualization of Flows and Volumes," *Proc. IEEE Conf. Visualization*, pp. 687-694, 2005.
[32] J. Krüger, J. Schneider, and R. Westermann, "ClearView: An Interactive Context Preserving Hotspot Visualization Technique," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, pp. 941-948, Sep./Oct. 2006.
[33] M. Harrower and C.A. Brewer, "Colorbrewer.org: An Online Tool for Selecting Colour Schemes for Maps," *Cartographic J.*, vol. 40, no. 1, pp. 27-37, June 2003.
[34] I. Viola, A. Kanitsar, and M.E. Gröller, "Hardware-based Nonlinear Filtering and Segmentation Using High-level Shading Languages," *Proc. IEEE Conf. Visualization*, pp. 309-316, 2003.
[35] G. Kindlmann and J.W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. IEEE Symp. Volume Visualization (VVS '98)*, pp. 79-86, 1998.
[36] J. Zhou, M. Hinz, and K.D. Tönnies, "Focal Region-guided Feature-Based Volume Rendering," *Proc. First Int'l Symp. 3D Data Processing, Visualization, and Transmission*, pp. 87-90, 2002.
[37] D. Ebert and P. Rheingans, "Volume Illustration: Nonphotorealistic Rendering of Volume Models," *Proc. IEEE Conf. Visualization*, pp. 195-202, 2000.
[38] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. Conf. Computer Graphics and Interactive Techniques*, pp. 163-169, 1987.
[39] D. Horn, "Stream Reduction Operations for GPGPU Applications," *Proc. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, pp. 573-583, 2007.
[40] M. Harris, S. Sengupta, and J.D. Owens, "Parallel Prefix Sum (scan) with CUDA," *GPU Gems 3*, H. Nguyen, ed. Addison Wesley, pp. 851-876, 2007.
[41] HDR Image Processing Library, http://courses.ece.uiuc.edu/ece498, 22 Apr., 2011.
[42] L.D. Bergman, B.E. Rogowitz, and L.A. Treinish, "A Rule-Based Tool for Assisting Colormap Selection," *Proc. Conf. Visualization*, pp. 118-125, 1995.
[43] S. Takahashi and Y. Takeshima, "A Feature-Driven Approach to Locating Optimal Viewpoints for Volume Visualization," *Proc. IEEE Conf. Visualization*, pp. 495-502, 2005.
[44] M.-Y. Chan, Y. Wu, W.-H. Mak, W. Chen, and H. Qu, "Perception-Based Transparency Optimization for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1283-1290, Nov./Dec. 2009.

Insoo Woo received the BS degree in computer engineering in 1998 from Dong-A University in South Korea. He is currently working toward the PhD degree in the School of Electrical and Computer Engineering at Purdue University and a research assistant in the Purdue University Rendering and Perception Lab. He worked as a software engineer from 1997 to 2006. His research interests include GPU-aided Techniques for Computer Graphics and Visualization. He is a member of the IEEE.

Ross Maciejewski received the PhD degree in electrical and computer engineering from Purdue University in December, 2009. He is currently an assistant professor at Arizona State University in the School of Computing, Informatics & Decision Systems Engineering. Prior to this, he served as a visiting assistant professor at Purdue University and worked at the Department of Homeland Security Center of Excellence for Command Control and Interoperability in the Visual Analytics for Command, Control, and Interoperability Environments (VACCINE) group. His research interests include geovisualization, visual analytics, and nonphotorealistic rendering. He is a member of the IEEE and the IEEE Computer Society.

Kelly P. Gaither received the masters and bachelors degree in computer science from Texas A&M University in 1992 and 1988, respectively, and the doctoral degree in computational engineering from Mississippi State University in May, 2000. While working toward the PhD degree, she worked full time at the Simulation and Design Center in the National Science Foundation Engineering Research Center as the leader of the visualization group. She is the director of Data & Information Analysis at the Texas Advanced Computing Center (TACC), is leading the scientific visualization, data management & collections, and data mining & statistics programs at TACC while conducting research in scientific visualization and data analysis. She is a research scientist, also serves as the area director for visualization in the National Science Foundation funded TeraGrid project. She has a number of refereed publications in fields ranging from Computational Mechanics to Supercomputing Applications to Scientific Visualization. She has given a number of invited talks. Over the past 10 years, she has actively participated in the IEEE Visualization conference and served as the IEEE Visualization conference general chair in 2004. She is currently serving on the IEEE Visualization and Graphics Technical Committee. She is a member of the IEEE.

David S. Ebert received the PhD degree in computer science from Ohio State University. He is the silicon valley professor in the School of Electrical and Computer Engineering at Purdue University, a University Faculty Scholar, the director of the Purdue University Rendering and Perceptualization Lab, and the director of the Visual Analytics for Command, Control and Interoperability Environments Center of Excellence. His research interests include novel visualization techniques, visual analytics, volume rendering, information visualization, perceptually based visualization, illustrative visualization, and procedural abstraction of complex, massive data. He is a fellow of the IEEE and the IEEE Comptuer Society, and a member of the IEEE Computer Society's Board of Governors.