Intuitive Transfer Function Editing Using Relative Visibility Histograms

Shengzhou Luo
Graphics Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland
luos@tcd.ie

Subhrajyoti Maji
Graphics Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland
majis@tcd.ie

John Dingliana
Graphics Vision and Visualisation Group (GV2)
Trinity College Dublin
Ireland
John.Dingliana@tcd.ie

ABSTRACT

In this paper, we present an interactive approach for intuitively editing colors and opacity values in transfer functions for volume visualization. We introduce the concept of a relative visibility histogram, which represents the difference between the global visibility distribution across the full volume and the local visibility distribution within a user-selected region in the viewport. From this measure, we can infer what subset of the 3D volume the user intends to select when they click on a region in the 2D rendered image of the data set, and use this to modify relevant parts of the transfer function. We use this selection mechanism for two alternative purposes. The first is to allow output-driven editing of the transfer function, whereby a user can change the opacity values and colors of features without directly having to manipulate the transfer function itself. The second is to extract visually dominant features in any user-selected region of interest, so that the user may individually edit their appearance and then merge these to create new transfer functions. Our approach is lightweight compared to similar techniques and performs in real-time.

Keywords

Volume rendering, transfer function, visibility, visibility histograms

1 INTRODUCTION

A recurring challenge in volume visualization is defining effective transfer functions (TF), which assign color and opacity (alpha value) to specific data ranges for visualization. Due to the non-linear relationship between the transfer function and the resultant rendering, the process of editing transfer functions is often counter-intuitive, typically necessitating a trial-and-error process. This may be addressed using an output sensitive approach where the user can more directly control the appearance of the visualization, without explicit knowledge of the transfer function.

In this paper, we propose a technique which enables us to infer a user's intended changes to the visualization when they click or select a region in the rendered image of a 3D volume data set. 3D selection is a non-trivial process in volume visualization due to the presence of many overlapped layers of transparent data. This is achieved in our solution by weighting the data in the selected region based on the proportion of materials visible to the user within that region. We introduce the concept of a *relative visibility histogram*, derived from the relationship between the global visibility and the local visibility of data in the user-selected region. Based on this weighting, the user can directly modify colors and opacity values in the rendered image of the volume data, in a manner analogous to painting a 3D scene. In addition, we introduce an automated technique for creating transfer function components from relative visibility histograms to represent features of interest in the selected regions. This technique allows users to edit transfer function on a feature level by manipulating the colors and opacity values of the components and merging them to create new transfer functions. Compared to other similar techniques, our approach is relatively lightweight, requiring only intermediate information about the visibility of data samples. It is thus simple to implement and performs in real-time.

2 RELATED WORK

Transfer function specification is an essential part of the volume visualization pipeline. The specification is often achieved by a trial-and-error process, which in-

volves a significant amount of tweaking of colors and opacity values, and the resultant visualization largely depends on how well the transfer function captures features of interest [KKH02] [KWTM03].One of the challenges for such an approach is highlighted by Mindek et al. [MMGB17] who argue that often small changes in input parameters (i.e. the transfer function variables) can lead to disproportionately large changes to the visualization. While Mindek et al. propose a data-sensitive solution that addresses this disproportionality, others take the route of output-driven transfer function editing, that is, manipulation of what is rendered, without the user being explicitly exposed to the underlying changes to the transfer function.

For instance, Guo et al. [GMY11] proposed a sketch-based approach that allows direct manipulation of transfer functions by brushing strokes on top of volume rendered images, which is similar to the operations in painting applications for 2D images. Later, Guo and Yuan [GY13] extended the sketch-based technique for specifying local transfer functions for topology regions using contour trees. Wu and Qu [WQ07] presented an approach that allows users to select sample images rendered using predefined transfer functions and generates new transfer functions by fusing multiple features in distinct volume renderings. Bruckner and Gröller [BG07] presented style transfer functions, which allow the user to specify styles extracted from actual illustrations in the transfer function. Ropinski et al. [RPSH08] proposed a stroke-based approach for specifying transfer functions by drawing strokes near silhouettes on a monochromatic view of the volume and generating transfer function components [CKLE98] that later can be modified and combined to explore the volume.

Many of the aforementioned approaches require, among other things, a model of what is visible to the user from a particular view direction, in other words, the visibility of features in volume data. The visibility of a sample refers to the alpha contribution of a sample to the final image, taking into account the degree to which it is occluded by other samples. This can be computed during ray-casting as the difference between the accumulated alpha of a sample and the accumulated alpha of the previous sample along a ray in the view direction [Ems08]. Correa and Ma presented the general notion of visibility histograms [CM11] which represent the distribution of visibility over intensity ranges in a volume rendering image. Wang et al. [WZC+11] extended visibility histograms to feature visibility histograms for measuring the influence of features on the resultant volume rendered images. Wiebel et al. [WVFH12] found that the user usually perceives features at a screen position with the highest visibility along a ray and exploited this information for volume picking.

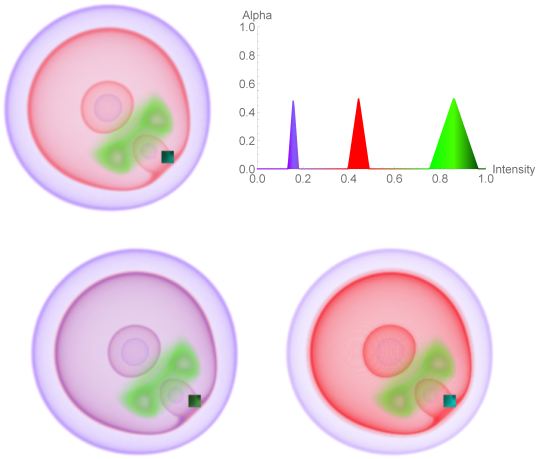


Figure 1: Sample operations using our technique. Top row: a nucleon with a selected region and its TF. Bottom left: Selected material colored in blue; Bottom right: Opacity of selected material enhanced

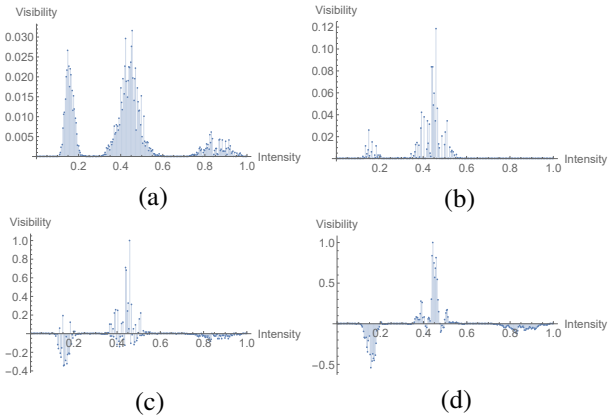


Figure 2: (a) Global Visibility Histogram of data set shown in Figure 1; (b) Local Visibility Histogram of selected region; (c) Relative Visibility Histogram; (d) Relative visibility histogram after smoothing

3 RELATIVE VISIBILITY HISTOGRAMS

In this section, we define a novel concept called the *Relative Visibility Histogram*, which is used as a mechanism for allowing users to select a subset of a 3D volume data set based on a selection in a 2D rendered view of the data. This is a key component that facilitates the two techniques, presented in subsequent sections, for intuitively manipulating transfer functions in volume rendering.

A visibility histogram [CM11] represents the visibility distribution of all voxels in the viewport when rendered from a given view, in other words, how visible any voxel is, given its opacity and the degree to which it is occluded by other voxels in the view direction. We will use the more specific term *Global visibility histogram*, H, to describe such a distribution and *Local visibility*

histogram, H_L, to describe the histogram representing the local visibility distribution for the voxels that contribute to a region of interest (ROI) in the rendered image, e.g., for a rectangular ROI on screen, this would be all the voxels that lie in the frustum extended by the rectangle. Furthermore, we introduce the concept of a Relative Visibility Histogram, derived from the former which is defined as the difference between H_L and H divided by the maximum of the absolute value in the difference, i.e.,

$$H_R = H_r / max(abs(H_r))$$

where $H_r = H_L - H$. The relative visibility histogram is scaled to the range $[-1,1]$ by dividing by the maximum absolute value in the histogram.

The purpose of the relative visibility histogram is as follows: firstly the local H_L component captures the dominant intensities in the ROI. Secondly, the subtraction and normalization against the global context capture a representation of which intensities are particularly densely distributed within the ROI and not elsewhere in the view. Essentially, we assume that when a user selects any particular ROI to select a subset of the volume, there is a strong likelihood that they will choose an area that contains a high number of voxels of the intensity ranges that they are interested in and that stands out as clearly dominated by that intensity range compared to the rest of the view.

An example is illustrated in Figure 1 and 2. Figure 1 shows a nucleon data set, its associated transfer function and sample modifications using our technique. The global visibility histogram is shown in Figure 2(a) and the local visibility histogram for the region of interest (the rectangle in inverted color) is shown in Figure 2(b). The relative visibility histogram is shown in Figure 2(c).

In order to smooth the histogram, we apply a Gaussian kernel to H_r and then scale it to the range $[-1,1]$. So the smoothed relative visibility histogram is

$$H_G = H_g / max(abs(H_g))$$

where $H_g = Gaussian(H_r, n, \sigma)$, n is the size and σ is the standard deviation of the Gaussian kernel (see Figure 2(d)).

Henceforth, this smoothed histogram H_G will be referred to as the relative visibility histogram, and $H_G(i)$, which is the value of the relative visibility histogram H_G at intensity i, will be referred to as the relative visibility of intensity i.

4 OUTPUT-DRIVEN COLOR AND ALPHA EDITING

The first application of the relative-visibility histogram is in allowing users to manipulate the visualization with no explicit knowledge of the transfer function. In this use-case, the user simply needs to select regions of interest on the rendered image that they wish to emphasize or color. Then, a single pass of volume ray casting is done to calculate the global visibility histogram for the whole volume and the local visibility histogram for the selected region in the viewport. From this we calculate the relative visibility histogram, which provides a measure of visible materials within the selected region. This is used to infer features that the user intends to edit in the visualization. More precisely, H_G is used as a weighting function to blend the colors or opacity values in the original transfer function with a user-selected target color or alpha value.

The user-selected target color is blended with the original transfer function for intensity ranges that have positive values in the relative visibility histogram as below:

$$C_i = \begin{cases} C_i + H_G(i)(C_s - C_i) & \text{if } H_G(i) > 0 \\ C_i & \text{otherwise} \end{cases}$$

where $H_G(i)$ denotes the relative visibility at intensity i in H_G, C_s is the user-selected target color and C_i the color of intensity i in the original transfer function.

Similarly, the alpha (A_i) of the transfer function is increased in intensity ranges that have positive relative visibility values, and decreased for ranges with negative relative visibility values, as follows:

$$A_i = \begin{cases} A_i + H_G(i)(1 - A_i) & \text{if } H_G(i) > 0 \\ A_i - H_G(i)(0 - A_i) & \text{otherwise} \end{cases}$$

Note that the color blending and alpha blending operations can be applied separately. Figure 1(c) displays the result of only applying the color blending to the volume rendering of the nucleon data set in Figure 1(a), and Figure 1(d) displays the result of only applying the alpha blending to the original.

Furthermore, as the blending process is fast, any changes can be applied by the user iteratively, analogous to a paintbrush-like tool for a large number successive of operations.

5 TRANSFER FUNCTION COMPONENTS

In addition to low-level output-driven editing of the transfer function, the relative visibility histogram allows us to support editing the transfer function at a feature level, which is a desirable use case in many volume visualization applications. For instance, Ropinski et al. [RPSH08] reported that physicians had high-level requests such as emphasizing, adding or removing specific features in collaboratively adapting visualizations.

As previously discussed, relative visibility histograms reveal what intensity ranges are concentrated in the

view frustum behind a selected 2D viewport region. The visually dominant features in the selected region can be represented by *transfer function components* proportional to the positive parts of relative visibility histograms.

A discrete set of such component transfer functions can be composed into a new transfer function for the full data set, and then the colors and opacity values of individual components can be separately modified.

5.1 Feature Specification Using Transfer Function Components

In this approach, features are automatically generated as transfer function components based on the relative visibility histograms created from user-selected regions. Let $F(i)$ denote a transfer function component derived from the relative visibility histogram H_G.

$$F(i) = \begin{cases} H_G(i) & \text{if } H_G(i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $H_G(i)$ is the value of the relative visibility histogram H_G at intensity i.

Figure 3 displays two examples of the transfer function components created by a rectangular selection within the rendered volume image respectively. Figure 3 (a) and (b) show the two selected regions in the volume rendering. Figure 3 (c) and (d) show the two transfer function components which represent the relative visibility distributions of features in the two user-selected regions respectively. The regions are rectangles centered around a single point in 2D screen space, and the region sizes can be modified according to the user's need.

5.2 Image-Space Clustering for region of interest selection

Heretofore, we have assumed a rectangular region of interest centered on a point such as the location of the mouse cursor when a user clicks on the screen. While this is a reasonable representation of the user's interest for the purposes of low-level iterative editing proposed in the previous sections, a more robust representation of the user's intended selection may be obtained by a more generalized representation of this region. We propose that one alternative of the region of interest can be obtained by segmenting the rendered image into contiguous visual objects in 2D (we avoid using the term "feature" here to avoid confusion with the transfer function features discussed previously).

In order to achieve visual object selection, image segmentation is performed on volume rendered images and the resultant segments are stored as individual masks

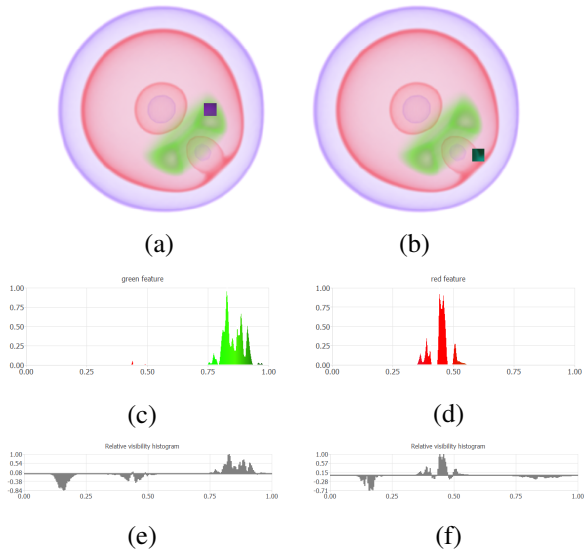


Figure 3: Examples of rectangular region selections. Left: A TF component (c) created from a green rectangular region (a) (highlighted in inverted colors) and its relative visibility histogram (e); Right: A TF component (d) created from a red region (b) and its relative visibility histogram (f).

for object selection. More specifically, a GPU accelerated k-means clustering implementation is used to accomplish interactive segmentation of volume rendered images. The distance metric used for the segmentation is the Euclidean distance in the RGB color space.

Now, when the user clicks on a position of the volume rendered image, a selected region is formed by all pixels that belong to the same segment as the pixel at the mouse position. Figure 4 displays the regions selected by clicking on the same screen positions as in Figure 3.

Compared to the rectangular regions in Figure 3 (a) and (b), the selected regions in Figure 4 (a) and (b) are heterogeneously-shaped segments with colors similar to the pixels at the respective mouse positions. These screen regions are larger in size, resulting in larger view frustums with more voxels being selected, and the colors across the selection tends to be more homogeneous due to the clustering. As a result of this, we note that the resultant transfer function components in Figure 4 (c) and (d) are more continuous compared to those in Figure 3 (c) and (d) for the rectangular selections. The relative visibility histograms in Figure 4 (e) and (f) are also smoother than those in Figure 3 (e) and (f). To disambiguate the two select techniques, we refer to them henceforth as *rectangular selection* and *generalized selection*.

5.3 Merging Transfer Function Components

A new transfer function can be created by merging several transfer function components.

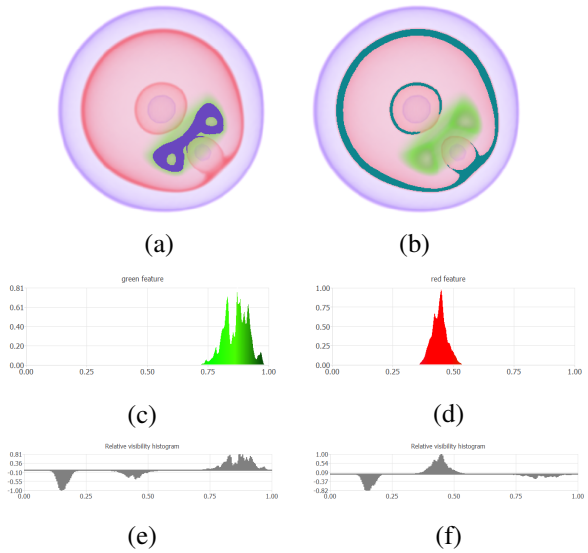


Figure 4: Examples of generalized segment selections. Left: A TF component (c) created from a green generalized selection segment (a) (highlighted in inverted colors) and its relative visibility histogram (e); Right: A TF component (d) created from a red segment (b) and its relative visibility histogram (f).

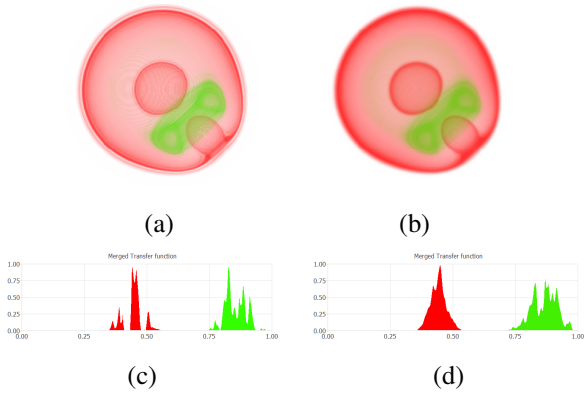


Figure 5: (a) and (c): Volume rendering and its TF obtained from merging the TF components in Figure 3 using *rectangular selection*; (b) and (d): Volume rendering and TF based on *generalized selection* in Figure 4

The opacity function A_i is defined by a weighted sum of transfer function components clipped to the range $[0,1]$.

$$A_i = \begin{cases} a(i) & \text{if } a(i) \in [0,1] \\ 0 & \text{if } a(i) < 0 \\ 1 & \text{if } a(i) > 1 \end{cases}$$

where $a(i)$ is the weighted sum of transfer function components, i.e.

$$a(i) = \sum_{j=1}^{n} w_j F_j(i)$$

where w_j ($w_j \geq 0$) is the weight of transfer function component F_j, $F_j(i)$ is the value of F_j at intensity i, and n is the number of transfer function components.

Figure 5 shows the results of merging the transfer function components in Figure 3 and Figure 4 respectively.

Note that there are "wood grain" artifacts in the volume rendered image in Figure 5 (a), especially in the red feature. The artifacts are due to the gaps in the transfer function components, as shown in Figure 5 (c). In contrast, the volume rendered image in Figure 5 (b) does not have noticeable artifacts, because the transfer function components in Figure 5 (d) are smoother.

There may be overlaps between transfer function components. Two methods for deciding the color of the overlaps in the merged transfer functions are described below.

5.3.1 *Blending colors of transfer function components*

The first method is blending the colors of the transfer function components based on the weights and the values of the transfer function components.

In order to keep the blended colors in a valid range, the weights for blending the colors of transfer function components are normalized using weighted averages of the weights and the values of the transfer function components. The normalized weight of transfer function component F_j is defined by

$$W_j = \frac{w_j F_j(i)}{\sum_{k=1}^{n} w_k F_k(i)}$$

where w_j ($w_j \geq 0$) is the weight of transfer function component F_j, $F_k(i)$ is the value of the transfer function component F_k at intensity i, and n is the number of transfer function components.

Hence, the color function C_i is defined by

$$C_i = \sum_{j=1}^{n} W_j c_j$$

where W_j is the normalized weight of transfer function component F_j and c_j is the color of F_j, and n is the number of transfer function components.

Using this method, new colors that do not exist in the settings of transfer function components may be introduced due to the blending of colors of overlapping transfer function components. Moreover, a feature, represented by a transfer function component, may have various colors in the final volume rendering.

5.3.2 *Using colors of the dominant components*

In some cases, using a single color per feature is preferable for better distinction of features in the volume rendering.

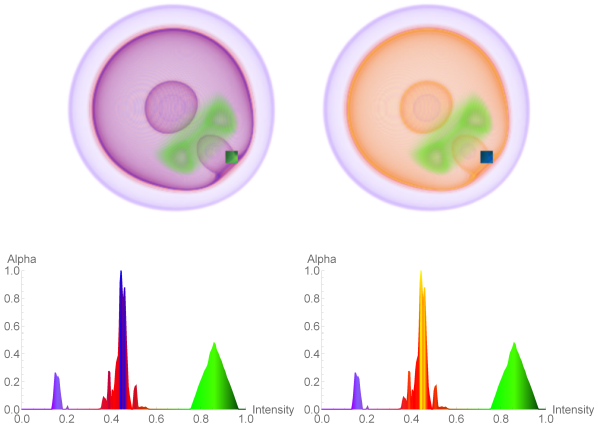


Figure 6: Combining operations. Left: Blue applied to selected region of TF in Figure 1 and opacity enhanced; Right: Yellow applied and opacity enhanced. The modified TF is shown for each case below the rendering

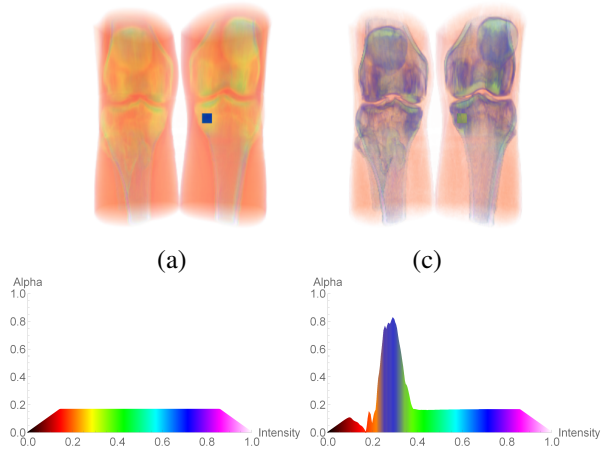


Figure 7: Left: CT-knee data set and basic TF; Right: Volume rendering and TF after blue applied and opacity enhanced for the selected region

Thus an alternative to blending colors is to use the color of the visually dominant transfer function component as the merged color.

The color function C_i is defined by

$$C_i = c_j, \quad j = \underset{k \in \{1,\ldots,n\}}{\operatorname{argmax}} w_k F_k(i)$$

where c_j is the color and w_j ($w_j \geq 0$) is the weight of transfer function component F_j with $w_j F_j(i)$ at intensity i that is maximum among the n transfer function components.

With this method, different features would have distinct colors, so that they are distinguishable by colors in the volume rendering.

6 RESULTS

Our solution comprises a ray-cast volume renderer and a visibility computation module, both implemented on the GPU using CUDA. The implementation is lightweight and achieves real-time performance at 30 to 40 frames per second on a computer equipped with an Intel Xeon E3-1246 v3 CPU and an NVIDIA Quadro K4200 graphics card.

We present some results to demonstrate the effectiveness of our approach on the nucleon (voxel dimensions: $41 \times 41 \times 41$), CT-knee ($379 \times 229 \times 305$) data sets, one time-step of a simulated turbulent vortex flow ($128 \times 128 \times 128$) and one time-step of a simulated supernova ($432 \times 432 \times 432$). Our implementation was able to handle all the data sets at interactive rates.

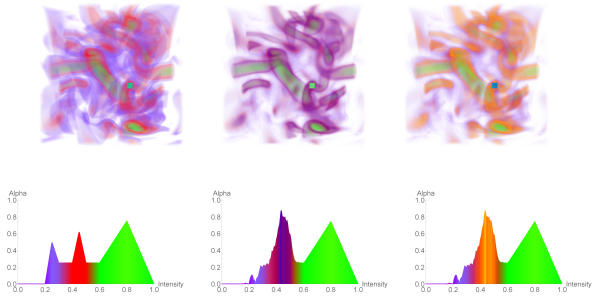
6.1 Output-driven Transfer Function Editing

Figure 6(left) displays the result of both applying color blue and adjusting alpha of the TF in Figure 1. Note that the intensity ranges with initial red color in the middle of the transfer function have been blended with blue and have become purple. Similarly, Figure 6(right) shows the result of applying yellow and adjusting alpha. Here, the intensity ranges in the middle have become orange after blending with yellow. In both cases, the alpha of the relevant parts of the transfer function is increased and the alpha of the less relevant parts is decreased in order to emphasize the materials of interest.

Figure 7(left) shows a rendered image of a CT-knee data set with a selected region over the bone and the initial transfer function. Figure 7(right) shows the image and the transfer function after applying a blue color and alpha adjustment. The bone material becomes mostly blue and is emphasized due to increased opacity, while the materials around the bone with lower relative intensity ranges are de-emphasized.

Figure 8 shows results of enhancing one time-step of a turbulent vortex data set. Figure 8(a) shows the rendered image and original transfer function. Figure 8(b) shows a clear visualization, and respective TF, of the materials of interest blended with blue and emphasized with higher alpha. Similarly, Figure 8(c) shows the materials of interest blended with yellow and emphasized with higher alpha.

The examples show that the technique can be applied effectively to a range of different data sets and transfer functions. Although we only show single step examples due to space constraints, it should be noted that changes can be applied by the user iteratively, analogous to a paintbrush-like tool for a large number successive of operations.



(a) (b) (c)

Figure 8: (a)Turbulent vortex data set and initial TF; (b)Blue applied to selected material and opacity enhanced; (c) Yellow applied and opacity enhanced.

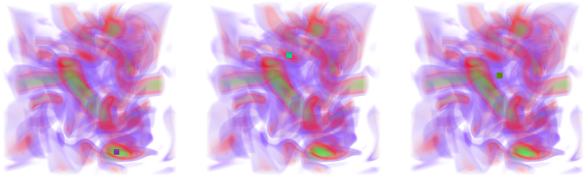
6.2 TF-components editing

Figure 9, Figure 10 and Figure 11 show results of creating and merging transfer function components on the time-step of the turbulent vortex data set.

Figure 9 displays three transfer function components created from a green region, a red region and a purple region, which are rectangular regions highlighted in inverted colors, in the volume rendering respectively. In contrast, Figure 10 displays three transfer function components created from three generalized segments based on image segmentation results. The three segments are visual objects with similar colors and are selected by clicking on the same positions as in the rectangular regions in Figure 9.

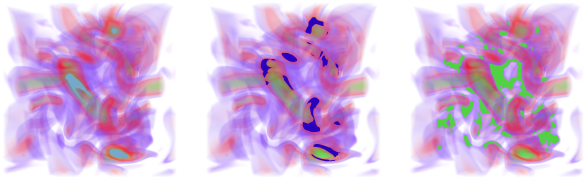
Figure 11 (a) and (d) show the volume rendered image and the transfer function created from merging the three transfer function components in Figure 9. The user interface for editing and merging transfer function components is displayed in Figure 11 (d), which shows the individual transfer function components with their weights and colors on the left, and the resultant transfer function at the bottom. Similarly, Figure 11 (b) and (e) display the volume rendered image and the transfer function created from merging the three transfer function components in Figure 10 with color blending, and Figure 11 (c) and (f) display the results of merging the three transfer function components in Figure 10 using colors of the dominant components. In Figure 11, the three transfer function components are merged with weights {2,1,0.2}, so that the purple feature is de-emphasized, the green feature is emphasized, and the red feature remains the same level of opacity.

Figure 13 displays the results of merging two transfer function components, i.e. the red feature and the green feature, with weights {1, 1}. The transfer function components are created from the user-selected rectangular regions in Figure 12 (a) and (b), and the user-selected generalized segments in Figure 12 (c) and (d) respectively. Figure 12 (e) shows the initial transfer function used for volume rendering.



(a) (b) (c)

Figure 9: User-selected regions highlighted in inverted colors. (a): a rectangular region in the green material; (b): a rectangular region in the red material; (c): a rectangular region in the purple material



(a) (b) (c)

Figure 10: Generalized segments selected at the same positions as in Figure 9, highlighted in inverted colors. (a): a segment in the green material; (b): a segment in the red material; (c): a segment in the purple material

From Figure 11 and Figure 13, we observe that the transfer function components created from segments are smoother and have wider intensity ranges than those created from rectangular regions.

7 CONCLUSION

In this paper, we introduce relative visibility histograms for inferring user intentions and present interactive techniques for editing colors and opacity values in transfer functions for volume visualization. We describe an output-driven color and alpha editing technique as well as a higher level technique that involves creating and merging transfer function components which represent features in the volume rendering.

Our color and alpha editing approach described in section 4 has a similar interaction paradigm to that proposed by Guo et al. [GMY11] in terms of emphasizing features and applying colors to features. However, the feature definition in Guo et al.'s approach relies on clustering of four attributes, i.e. depth, visibility, alpha and intensity. The clustering of attributes of voxels may be computationally heavy particularly for large volume data sets. In contrast, our approach identifies relevant intensity ranges of the transfer function based purely on visibility information, thus requiring a much more lightweight approach.

Our transfer function components approach discussed in section 5 is similar to the work by Ropinski et al. [RPSH08] in how the transfer function components are modified and merged to create new transfer functions. However, the two approaches differ in how features are

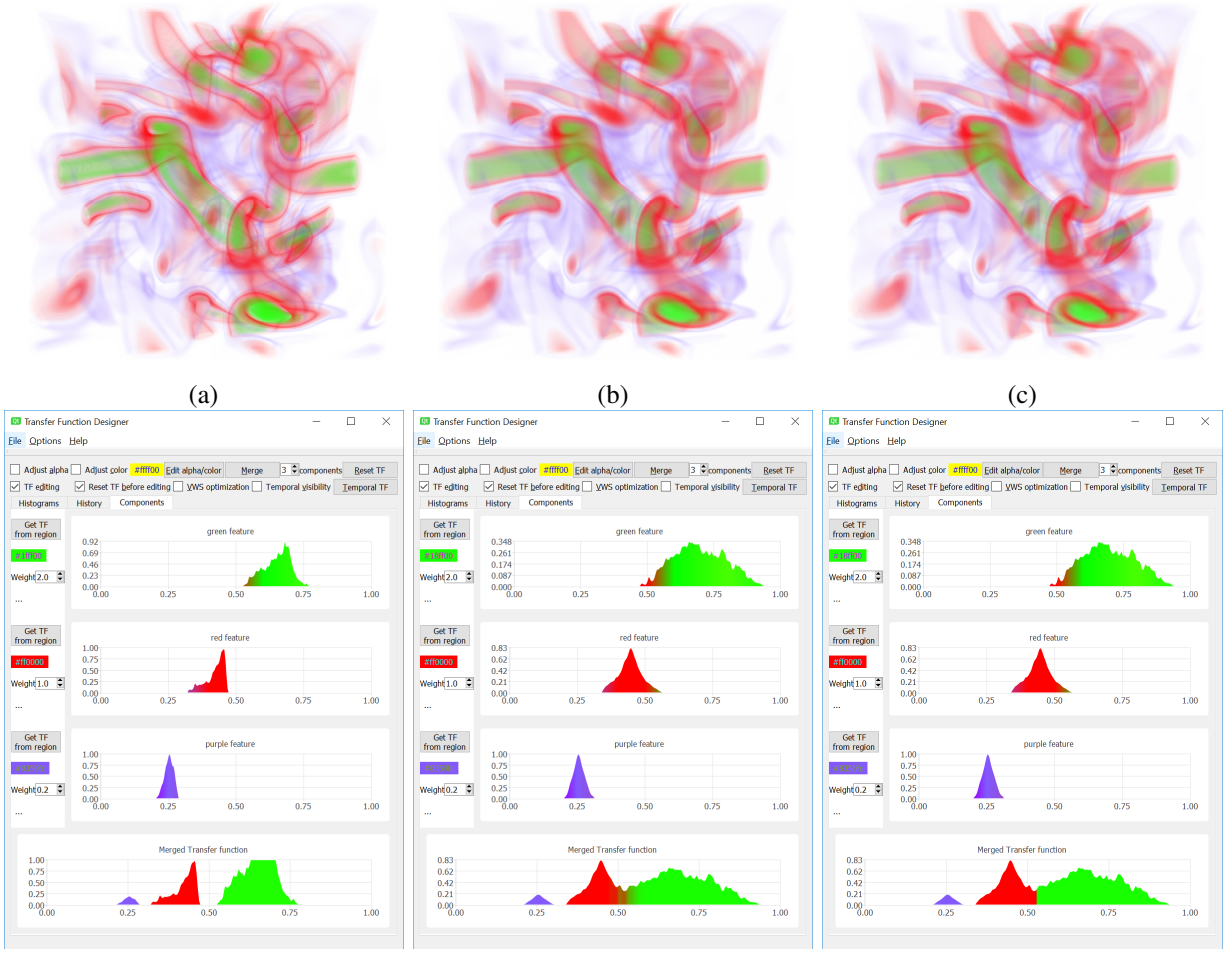


Figure 11: Merging 3 features with weights {2, 1, 0.2} and colors from peaks of TF components; (a) and (d): Volume rendering and TF from merging the TF components in Figure 9; (b) and (e): Volume rendering and TF from merging the TF components in Figure 10 with color blending; (c) and (f): Volume rendering and TF from merging the TF components in Figure 10 using colors of the dominant components

identified and transfer function components are generated. The approach by Ropinski et al. generates two further strokes which are both parallel to the user-drawn stroke along the silhouette and positioned in the same distance on its opposite sides. They hypothesize that the inner stroke covers the feature of interest in image space while the outer stroke does not cover it. In some cases, such as a complex flow visualization, it may be difficult to draw a stroke along the silhouette and determine a distance so that the two further generated strokes would be one inside the feature of interest and the other outside of it.

In our transfer function components approach, only a region inside the feature of interest is needed for creating a transfer function component to represent the feature. Moreover, apart from selecting a rectangular region around the mouse position, the user can also select segments generated from k-means clustering in image space. The segments often cover more pixels and thus lead to smoother transfer function components.

Both the visibility-driven transfer functions by Correa and Ma [CM11] and the feature visibility technique by Wang et al. [WZC+11] utilize iterative optimizations to refine the transfer functions. Correa and Ma derived the target visibility distribution from the user-defined transfer function, while Wang et al. allow the user to specify the target feature visibility. The transfer functions are then refined by minimizing the difference between the visibility distribution of the current volume rendering and the target visibility distribution. In contrast, our approaches do not involve an iterative optimization process. The relative visibility histogram is only computed once in both of the uses cases we presented.

However, the proposed techniques are subject to the limitations of 1D transfer functions, e.g. it is not possible to separate features of interest that overlap in the 1D transfer function domain. Both the color and alpha edit-

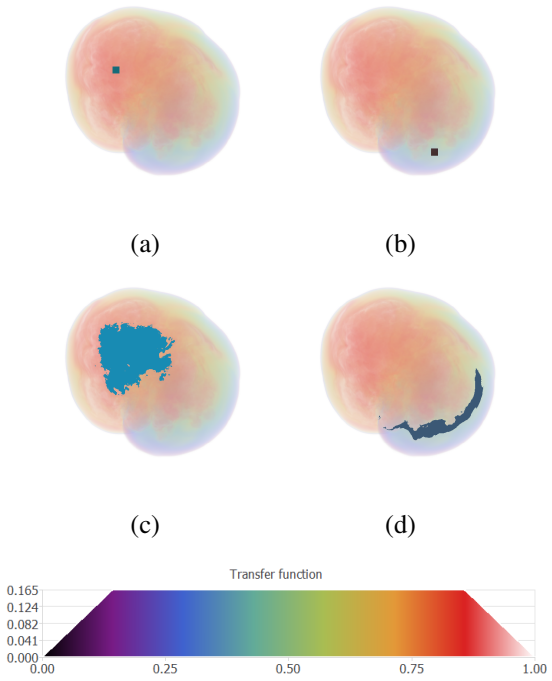


Figure 12: User selections on the rendering of a supernova data set; (a) & (b): User-selected rectangular regions highlighted in inverted colors; (c) & (d): User-selected generalized segments highlighted in inverted colors; (e): The initial transfer function

ing and transfer function components techniques are based on relative visibility histograms, which indicate the difference between the global visibility distribution across the volume and the local visibility distribution within the user-selected region. Therefore, only materials that are visible in the initial transfer function can be captured and edited by the proposed techniques.

In future, we would like to conduct user studies to evaluate the effectiveness of the proposed techniques, in particular with expert users in specific domains that use volume data. We believe our approach is particularly suited for tasks where there is no clear a priori search target, which might be the case in many complex fluid visualizations. We used some standard approaches to some component mechanisms of our solution and further study may be warranted to determine if the use of alternative segmentation techniques or a perceptually-based color space such CIE-LAB may improve the quality of the overall solution.

8 ACKNOWLEDGEMENTS

The Supernova data set is made available by Dr. John Blondin at the North Carolina State University through US Department of Energy's SciDAC Institute for Ultrascale Visualization. Other data sets were obtained from the Volume Library courtesy of Stefan Roettger, the Stanford Volume Data archive and the Time-varying data repository at UCDavis. The nucleon data set was obtained from the free distribution of the Voreen engine. We would like to thank the respective owners for making these data sets available.

This research has been conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1895.

9 REFERENCES

[BG07] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007.

[CKLE98] Silvia Castro, Andreas König, Helwig Löffelmann, and Eduard Gröller. Transfer Function Specification for the Visualization of Medical Data. Technical Report, Universität Wien, 1998.

[CM11] Carlos D. Correa and Kwan-Liu Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):192–204, 2011.

[Ems08] Gerlinde Emsenhuber. *Visibility Histograms in Direct Volume Rendering*. Master's Thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, November 2008.

[GMY11] Hanqi Guo, Ningyu Mao, and Xiaoru Yuan. WYSIWYG (What You See is What You Get) Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106–2114, 2011.

[GY13] Hanqi Guo and Xiaoru Yuan. Local WYSIWYG volume visualization. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 65–72, February 2013.

[KKH02] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, July 2002.

[KWTM03] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 513–

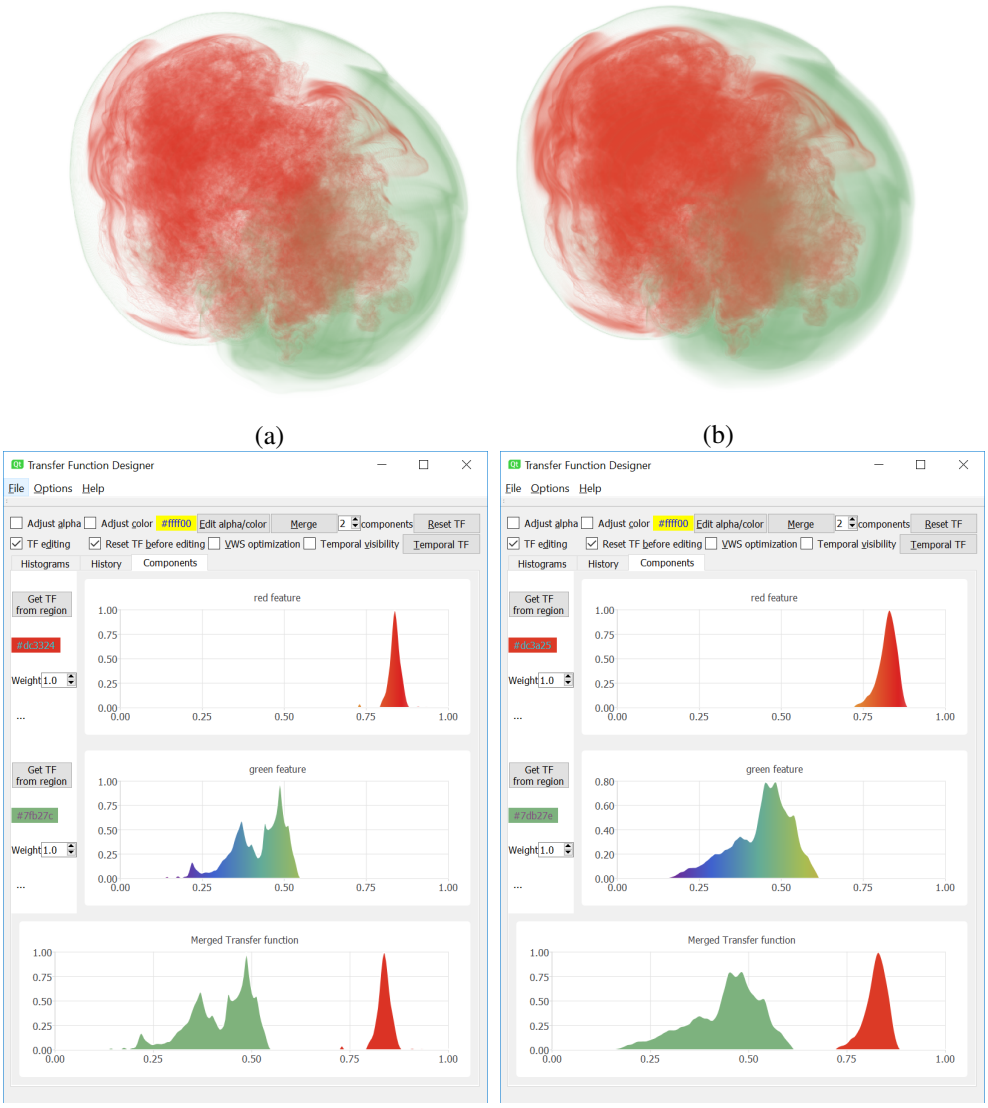


Figure 13: Merging 2 features with weights {1, 1} and colors from peaks of TF components; (a) & (c): Volume rendering and TF from merging the TF components in Figure 12 (a) and (b); (b) & (d): Volume rendering and TF from merging the TF components in Figure 12 (c) and (d)

520, Washington, DC, USA, 2003. IEEE Computer Society.

[MMGB17] Peter Mindek, Gabriel Mistelbauer, Eduard Gröller, and Stefan Bruckner. Data-sensitive visual navigation. *Computers & Graphics*, 67:77–85, October 2017.

[RPSH08] Timo Ropinski, Jörg-Stefan Praßni, Frank Steinicke, and Klaus H. Hinrichs. Stroke-Based Transfer Function Design. In *IEEE/EG International Symposium on Volume and Point-Based Graphics*, pages 41–48. IEEE, 2008.

[WQ07] Yingcai Wu and Huamin Qu. Interactive transfer function design based on editing direct volume rendered images. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1027–1040, 2007.

[WVFH12] A. Wiebel, F.M. Vos, D. Foerster, and H.-C. Hege. WYSIWYP: What You See Is What You Pick. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2236–2244, 2012.

[WZC+11] Yunhai Wang, Jian Zhang, Wei Chen, Huai Zhang, and Xuebin Chi. Efficient opacity specification based on feature visibilities in direct volume rendering. *Computer Graphics Forum*, 30(7):2117–2126, 2011.