Visibility-Driven Transfer Functions

Carlos D. Correa [*] Kwan-Liu Ma [†]

University of California, Davis

Abstract

Direct volume rendering is an important tool for visualizing complex data sets. However, in the process of generating 2D images from 3D data, information is lost in the form of attenuation and occlusion. The lack of a feedback mechanism to quantify the loss of information in the rendering process makes the design of good transfer functions a difficult and time consuming task. In this paper, we present the notion of visibility-driven transfer functions, which are transfer functions that provide a good visibility of features of interest from a given viewpoint. To achieve this, we introduce visibility histograms. These histograms provide graphical cues that intuitively inform the user about the contribution of particular scalar values to the final image. By carefully manipulating the parameters of the opacity transfer function, users can now maximize the visibility of the intervals of interest in a volume data set. Based on this observation, we also propose a semi-automated method for generating transfer functions, which progressively improves a transfer function defined by the user, according to a certain importance metric. Now the user does not have to deal with the tedious task of making small changes to the transfer function parameters, but now he/she can rely on the system to perform these searches automatically. Our methodology can be easily deployed in most visualization systems and can be used together with traditional 1D opacity transfer functions based on scalar values, as well as with multidimensional transfer functions and other more sophisticated rendering algorithms.

Keywords: Transfer functions, volume rendering

Index Terms: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism

1 Introduction

Despite the proliferation of volume rendering software, the design of effective transfer functions is still a challenge. The growing popularity of GPU-based volume renderers has advocated the use of a more exploratory approach, where users can arrive at good transfer functions via trial-and-error modification of opacity and color values. However, effective transfer functions are often the product of time-consuming tweaking of opacity parameters until meeting a desired quality metric, often subjective. One possible explanation for this ad hoc methodology is the lack of an objective measure to quantify the quality of transfer functions. In this paper, we propose the use of a visibility metric, which attempts to measure the impact of individual samples on the image generated by a volumetric object. Visibility has been studied in the past, either to measure the quality of a given viewpoint [2], or to enhance the rendering process with ghost and cutaway views [24].

In this paper, we present a more fundamental approach, where visibility is used as a primitive to quantify the quality of transfer functions and ease their design towards more meaningful and efficient visualization. The transfer functions generated with this approach are then known collectively as *visibility-driven transfer functions*. We present several mechanisms for obtaining these. On one hand, quantifying visibility helps the user to better create and refine transfer functions from an initial specification to one that matches the relative visibility amount of the different samples. On the other hand, visibility guides the semi-automatic generation of transfer functions, by replacing the tedious exploration of transfer function space by an automatic approach, which attempts to maximize the visibility of features of interest.

Our contributions are two-fold. On one hand, we present a notion of visibility histogram, which represents the visibility of the sample values from a given viewpoint. These visibility histograms provide a feedback mechanism for both manual and automatic transfer function design. Visibility histograms, however, are view and opacity dependent. Therefore, we present and compare two GPU-based algorithms for computing these histograms at interactive rates. On the other hand, we show how visibility can be used to formulate an objective function that is minimized whenever the distribution of visible samples matches a desired transfer function roughly defined by the user. We present a mathematical formulation of this problem that can be solved with a variety of optimization algorithms, such as steepest descent and nonlinear conjugate gradient methods. We show that this semi-automatic approach provides tools for the user to better manipulate the parameter space of the transfer functions. User can initiate linear searches of specific parameters that converge to optimal solutions with respect to visibility, without requiring to manually tweak them. This direct manipulation of transfer function parameters is often a tedious task that involves making small changes to a set of parameters, sometimes impossible to specify by hand. Our approach can explore these subtle variations more efficiently to provide the best solution to the user and becomes an important aid for volume exploration. Through a series of examples we show that our approach can be deployed in a variety of visualization applications and can be customized quite easily for different application requirements.

2 Related Work

Transfer function design is an essential part of volume visualization. Approaches to this problem are often classified as either data- or image-centric [17]. Data-centric approaches analyze the scalar field and its properties to guide the design of transfer functions. The most commonly used is a 1D transfer function based on scalar data value. Researchers have proposed higher-dimensional transfer functions based on first and second derivatives of the volume, i.e., gradient information [13, 10] and curvature [9, 11]. To aid in the process of finding transfer functions these approaches often make use of histograms, which represent graphically the distribution of values along the different dimensions. For 2D transfer functions, for example, surfaces of interest appear as arcs. Kniss et al. [12] exploit this behavior to derive a set of manipulation widgets as a user interface. As more dimensions are added, the N-dimensional histogram becomes increasingly difficult to understand and manipulate. Lum and Ma use a variant of the 2D histogram with gradient-aligned samples instead of first derivatives. This led to a different graphical representation of the distribution of samples where regions of different degree of homogeneity can be associated with

[*] e-mail: correac@cs.ucdavis.edu

[†] e-mail:ma@cs.ucdavis.edu

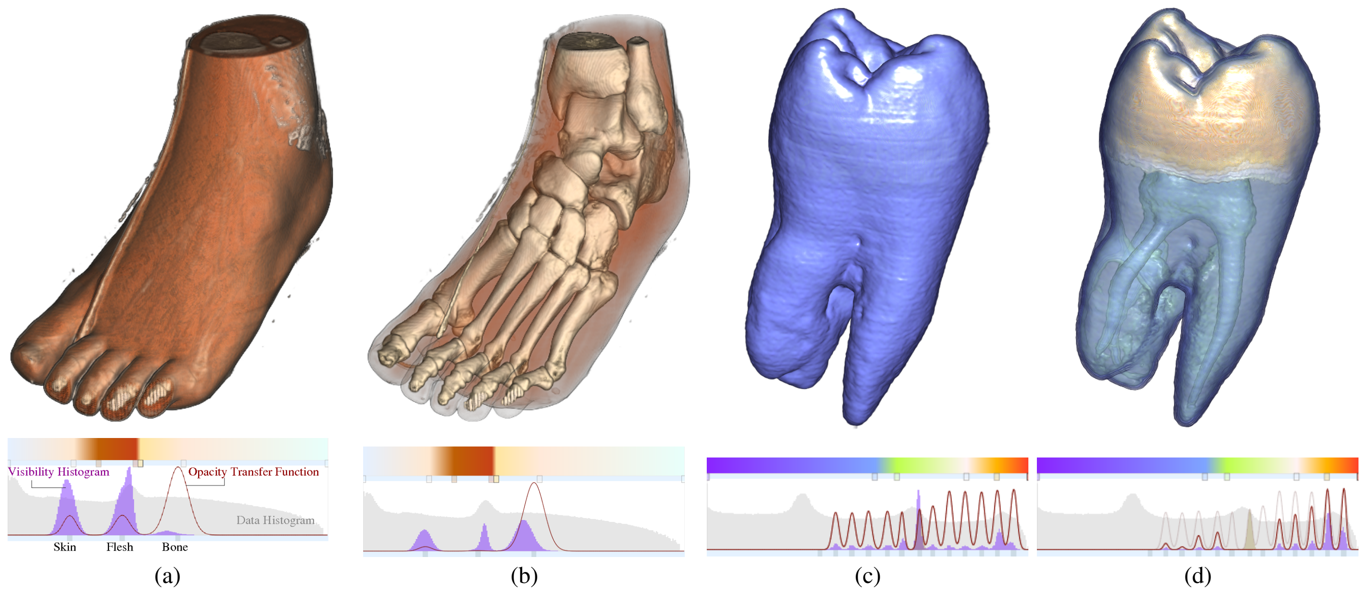


Figure 1: Visibility histogram for two datasets. (a) The visibility histogram (purple plot) applied to a CT foot dataset reveals that the opacity transfer function defined by the user (red line strip) is not effective at highlighting the interval of interest (bone). This is due to flesh tissue occluding underneath layers. (b) The user then manipulates the opacity of the skin and flesh intervals until the bone tissue has enough visibility. The VH provides immediate feedback that helps the user converge to a solution. (c) For a tooth dataset, rendering of different isosurfaces becomes a tedious approach. Similar to the case before, certain values occlude most of the important intervals (see the peak in the visibility histogram). Rather than letting the user manipulate each interval, we run an algorithm that automatically searches along the parameter space until the visibility of the surfaces of interest is maximized. The result is seen in (d). Notice how the most occluding intervals are made transparent.

different opacity and lighting parameters [14]. These approaches, despite their popularity and ease of implementation, cannot capture spatial information that may provide a better visibility of features of interest. Roettger et al. propose a solution by grouping spatially connected regions in the 2D histograms used for classification [20]. Lundström et al. suggest the use of local histograms [15] to represent the spatial distribution of scalar samples. As an alternative to histograms, Bajaj et al. propose the Contour Spectrum [1], which depicts a set of data attributes as a series of 1D plots for fast isosurfacing. As an alternative to these manual methods, Fujishiro et al. propose a topology analysis to derive good transfer functions automatically [7].

Unlike data-centric approaches, image-based methods operate on the rendered images. He et al. use a stochastic approach to search good transfer functions given a set of rendered images [8]. Marks et al. present design galleries, which organize a broad selection of volume rendered images as the product of a series of transfer functions. The user can explore the image-based space in the search for satisfactory transfer functions [16]. Fang et al. describe another image-based approach where a transfer function is defined as a sequence of image operations whose parameters can be explored by the user to achieve a desired classification [5]. Wu and Qu proposed a system that uses editing operations and stochastic search of the transfer function parameters to maximize the similarity between volume-rendered images given by the user [25].

In all of the above, the issue of visibility is more a consequence of transfer function design than a design parameter. In this paper, we propose to use visibility to guide transfer function design, for both manual and automatic searches. Image-based approaches often recur to optimization approaches and stochastic searches to find good transfer functions [8, 16, 25]. Our approach is a data-centric approach with similar goal-oriented searches, where an objective function, in our case in terms of visibility, is minimized. The notion of visibility has been used to find optimal viewpoints for volume rendering, as described by Bordoloi and Shen [2]. In their paper, visibility is used to construct an entropy function that guides the selection of optimal viewpoints. A similar approach is proposed by Takahashi et al. [23], although a volume is now separated into feature components. In our paper, we use visibility in a rather complementary way, which finds a transfer function with maximum visibility from any given viewpoint.

The need for maximizing visibility of feature of importance has led to a number of techniques that operate in the rendering space. Non-photorealistic rendering (NPR) operators modulate the opacity of samples in a view-dependent manner, increasing the visibility of otherwise occluded features [19]. Interactive cutaways achieve visibility by removing occluding surfaces [4]. Importance-driven volume rendering achieves a similar effect by mapping the importance of features to levels of sparseness, which control the visibility of features [24]. Context-preserving volume rendering combines NPR operations with the accumulated opacity, or visibility, to reduce the opacity of unimportant objects [3]. By keeping track of the accumulated opacity into a layered data structure, Rezk-Salama and Kolb propose opacity peeling, which helps reveal occluded features of interest via a multi-layer metaphor [18]. In our paper, we follow a more fundamental approach, where visibility is incorporated as part of the transfer function design process.

3 Visibility Histograms

The visibility of a sample refers to the contribution of a sample to the final image, in terms of opacity. This visibility is also known as the accumulated opacity of a sample s:

$$\alpha(s) = 1 - e^{\int_s^D \tau(t)dt} \tag{1}$$

where $\tau(t)$ is the attenuation coefficient of a sample, usually represented as an opacity transfer function O. Therefore, the visibility

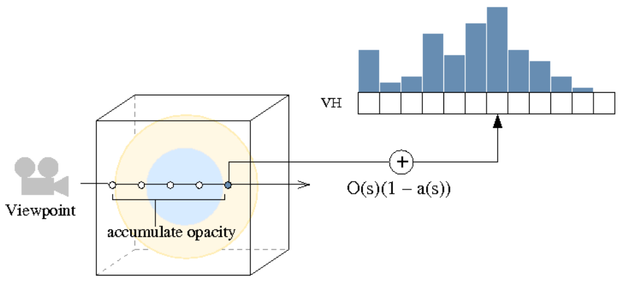


Figure 2: Computation of visibility histograms. Given a viewpoint, the total opacity of a given sample, computed as the product of the original opacity and the transfer function and the accumulated opacity, is added to the corresponding bin in the histogram.

of a sample depends on the opacity of the sample, $O(s)$, usually defined by the user, and the viewpoint, which affects the accumulated opacity in front of the sample.

A visibility histogram (VH) represents graphically the distribution of this visibility function in relation to the domain values of the volume. Traditional data histograms weight each sample value uniformly. For a visibility histogram, samples are weighted by visibility and added into bins that partition the range of values in the scalar field. For all sample values x in a volume V:

$$VH(x) = O(x) \int_{s\in\Omega} \delta(s,x)(1-\alpha(s))ds \tag{2}$$

where $\delta(s,x)$ is a function:

$$\delta(s,x) = \begin{cases} 1 & V(s) = x \\ 0 & otherwise \end{cases} \tag{3}$$

In practice, the histogram is computed at discrete bins, and the accumulated opacity is discretized with front-to-back compositing at discrete intervals. For a sample s:

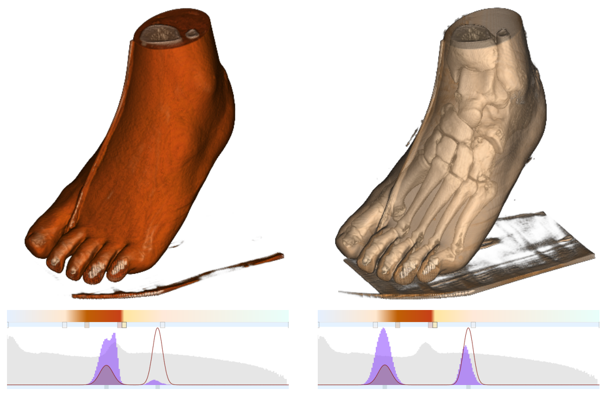
$$\begin{aligned} VH[x] &= VH[x] + (1-\alpha(s))O(x) \\ \alpha(s+\Delta s) &= (1-\alpha(s))O(x) + \alpha(s) \end{aligned} \tag{4}$$

Fig. 2 illustrates this process. Bordoloi et al. use a similar aggregation of visibilities to weight the data histogram and compute the entropy of a volume rendered image from a given viewpoint [2]. However, since only the final entropy is required, they do not need to explicitly compute the visibility-weighted histogram. In our case, the histogram itself is important as a visual aid.

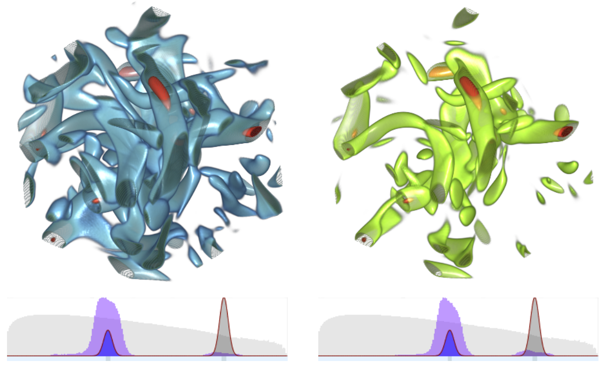
Visibility histograms help discover occlusion patterns on the data. For example, *strong occluders*, common in CT scans of anatomical structures. tend to dominate the visibility distribution. If the occluder has a large enough opacity, it prevents the occluded samples from being visible, making the histogram heavily skewed towards the unoccluded values. Conversely, if the occluder has a sufficiently small opacity, the VH now shifts towards the newly visible samples. An example is shown in Fig. 3(a). In contrast, some datasets exhibit a rather uniform distribution and no particular interval dominates in terms of visibility. These are common in certain simulation data, where scalar values vary evenly in the domain. An example is shown in Fig. 3(b). In this case, changing the opacity of the different isosurfaces does not have a dramatic effect on the distribution of the visibility, which reveals that no strong occluder is present in the data.

3.1 GPU-assisted Computation

As described above, it becomes important to compute visibility histograms at interactive rates. We have experimented with two GPU-assisted implementations, based on gather and scatter operations



(a) Occlusion signature on a CT foot dataset



(b) Occlusion signature on a flow simulation

Figure 3: Occlusion Signatures of two different datasets. (a) The signature on the histogram reveals the presence of a strong occluder. A little opacity in the interval of the occluder has a dramatic effect in the visibility of the hidden features. (b) The signature of this histogram reveals that no interval in the scalar field is a strong occluder.

respectively. In both approaches, there is a first pass which renders the volume from a given viewpoint using view-aligned slices. Each slice is used to compute the visibility values of its samples and compute the accumulated opacity, which is in turn required for the next slice.

The difference of the two implementations is in the process of gathering the visibility information of each slice. Our first approach is based on the approach by Fluck et al. [6]. This divides the screen area into tiles of 8×8 tiles. Since each pixel in the tile contains up to 4 components, the tile is used to store a 256-bin histogram. Each component of this tile contains the contribution of visibility to the sample value indicated by its position. For instance, the RGBA components of the top left pixel in the tile contains the visibility of all samples with value $0,1,2$ and 3, the next pixel those samples with values $4,5,6$ and 7, and so on. Hardware-supported blending adds the histograms of all the view-aligned slices. At the end, the histogram is distributed along the different tiles. A hierarchical gathering approach adds up the local histograms, and the result is read back to the CPU.

After considering this approach, we turned to scatter operations on the GPU, which ease the implementation. Our implementation is based on the approach by Scheuermann and Hensley [21]. Each slice generated during the first pass is sent to a vertex texture or vertex buffer, which is then used to render point primitives in the screen position corresponding to the bin in the histogram. After adding all the contributions of the different slices, the histogram can be transferred to the CPU by reading the screen area of a view port of size $1\times N$, where N is the number of bins.

Fig. 4(a) shows a comparison of these two methods in terms

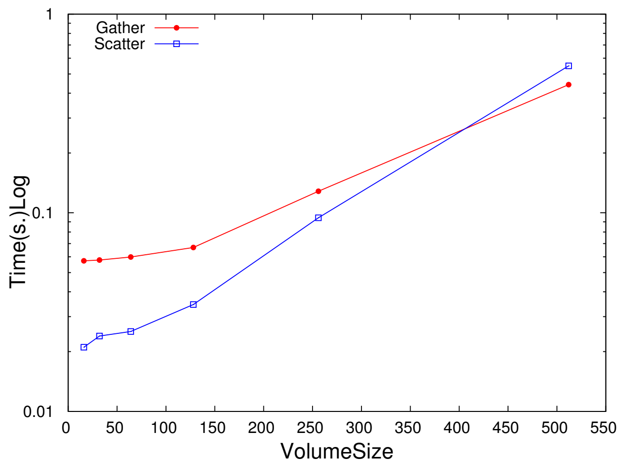


Figure 4: Evaluation of GPU-based histogram algorithms. Timing comparison between gather (red) and scatter algorithm (green) in seconds (log scale) vs. size of view-dependent volume. Note that scatter algorithm is faster for sampling rates less than $1/512$.

of speed. We computed the histogram at varying resolutions. Each resolution indicates a window size where the histogram is computed as well as a sampling resolution along the view direction. We noticed that our algorithm based on scatter operations was faster than the one based on gather for sizes up to 256^3. This is consistent with the results obtained by Scheuermann and Hensley [21] for 2D images. However, the cost grows faster and, at a size of 512^3, the gather algorithm outperforms the scatter algorithm. This can be explained due to the differences in performance between vertex and pixel processing capabilities in contemporary GPUs. The gather operation works exclusively in pixel shaders, while the scatter operation uses point primitives and vertex shaders. For large sizes, the vertex shader overhead overcomes the pixel shader overhead and we see a difference in performance. For this reason, we used a resolution of 256^3 to find the VH. The effect on the resulting VH is not dramatic, as can be seen in Fig.4(b). For smaller resolutions, though, the resulting VH is biased and scaled in a way that misrepresents the true visibility of the sampled data values.

4 Visibility-Guided Transfer Function Design

Visibility histograms provide the basis for generating visibility-driven transfer functions. A first approach consists of a manual transfer function design with immediate feedback. As the user manipulates the opacity transfer functions, the VH shows the user their impact on the visibility distribution. An example is shown in Figs. 1(a-b). Initially, the opacity transfer function defined by the user seems to indicate more importance to bone tissue, but it ends up with little visibility as it is occluded by skin and tissue. With a VH, the user can now manipulate the parameters of the transfer function, in this case the size of a Gaussian bell, until the visibility of the bone is high enough. The result in Fig.1(b) shows a better depiction of the three tissues with similar visibilities, as seen in the shape of the VH.

Figs. 1(c-d) shows a tooth dataset. In this case, the design of an effective transfer function is complicated as the user is required to manipulate more parameters, corresponding to a number of isosurfaces of interest. We notice a similar behavior as in the previous case, where one group of tissues dominates visibility and the inner tissues (such as the layers below the enamel and the nerves) are not visible at all. Reducing the opacity of the occluding tissue and the outer isosurfaces results in a much better depiction of the inner layers. Although effective, manual control of the transfer function parameters becomes a tedious task as the number of parameter increases, and as the resolution of the manipulation decreases. For example, notice how the sizes of the Gaussian bells are so small that direct manipulation via mouse interaction becomes increasingly difficult. Due to the multiplicative nature of attenuation in volume rendering, these little opacities may have a noticeable effect in visibility. For this reason, we present a semi-automatic approach that attempts to automate the search of transfer functions that maximize the visibility of important values.

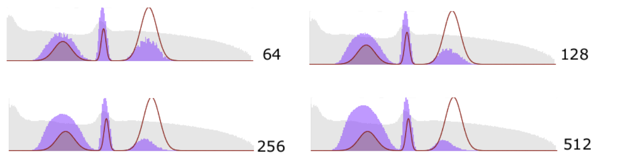


Figure 5: Effect of sampling resolution. As the resolution of the view-dependent slices decreases, the frequencies of the interval on the right are scaled and biased, misrepresenting the actual visibility of the sample values, measured as the visibility that would be obtained at the Nyquist frequency ($1/512$ in this example).

4.1 Semi-automatic Transfer Function Design

As described above, small changes in the opacity of samples values may change dramatically the visibility of occluded samples. In many cases, the user is required to perform minuscule changes that are almost impossible to make using mouse interaction. To automate this process, we formulate the design of transfer functions as an energy minimization problem.

First, we derive an energy function to represent the desired properties of an effective transfer function. For this, we assume that there is an opacity function O, defined by the user, which represents the overall desired importance of a given data value. This can also be regarded as the initial transfer function. We can also obtain this opacity function using alternative methods, for instance, by introducing pre-defined transfer functions or by analyzing the data and/or its topology [7]. We also assume that the opacity function can be obtained as a function of a parameter vector Θ. Examples of parametric primitives are Gaussian, rectangular and triangular functions, commonly supported in contemporary visualization systems.

We define the following energy components, designed to highlight certain desired aspect of the transfer function:

User-satisfaction. To ensure user-satisfaction, we minimize the mismatch between the computed opacity function and the original one defined by the user. The simplest way to represent this mismatch is via the square difference between the opacity functions:

$$E_S(x,\Theta) = (A(x,\Theta) - O(x))^2 \tag{5}$$

where $O(x)$ is the opacity function defined by the user, and $A(x,\Theta)$ is the opacity function derived by the system in terms of the opacity parameters Θ.

Visibility. The second component is used to maximize the visibility of a given sample. Because not all the samples are equally important, as defined in the opacity function O, we can weight the visibility of a sample by its opacity:

$$E_V(x,\Theta) = -O(x)(1 - \alpha(x,\Theta)) \tag{6}$$

where α is the visibility of a sample, as defined in Eq.1. Note that the sign is negative, which is minimized as the visibility of important values increases.

Constraints. Finally, we introduce constraints on the parameter space Θ. These constraints are used to control the minimum and maximum values opacities of value intervals in the final opacity function. This is particularly important for providing context in the resulting image. Without constraints, it may be the case that simply making all unimportant values transparent reveals the important ones. However, it provides little context. For each parameter in $\theta_i \in \Theta$, we define an interval of desired values $[\theta^i_{min}, \theta^i_{max}]$. The energy component is:

$$E_C(\Theta) = \sum_i \left[\theta^i_{min} - \theta_i\right]^2_+ + \left[\theta_i - \theta^i_{max}\right]^2_+ \qquad (7)$$

where $[x]_+$ is a clamping operation, such that $[x]_+ = x$ if $x > 0$ or 0, otherwise.

4.2 Optimization Algorithm

Once these components have been derived, we can define the process of transfer function design as a minimization problem,

$$\underset{\Theta}{\operatorname{argmin}} \sum \beta_1 E_S(\Theta) + \beta_2 E_V(\Theta) + \beta_3 E_C(\Theta) \qquad (8)$$

where β_1, β_2 and β_3 are weighting parameters that allows the user to give more importance to one component than the other.

To test and validate this formulation, we first define the parameter space as a mixture of Gaussians. That is, the opacity function A is defined as:

$$A(x) = \sum_i \alpha_i G_{\mu_i,\sigma_i}(x) \qquad (9)$$

where $G_{\mu,\sigma}(x)$ is a Gaussian function of mean μ and standard deviation σ. The parameter space corresponds to $\Theta = \{\mu_i, \sigma_i\}$. This model has the advantage that good transfer functions can be obtained with a small number of parameters, and the Gaussian falloff, given by the standard deviation, prevents the appearance of aliasing artifacts.

To solve this minimization problem, we follow a greedy approach. Since the energy function cannot be easily derived as an analytic function in general, finding a global optimum might require an exhaustive search of the parameter space, which is prohibitive. Instead, we use progressive search of the optimal solution by exploring the parameter space in directions that gradually decrease the energy function, $\Theta_{k+1} = \Theta_k + \gamma\Lambda_k$, where Λ_k is a search direction. A steepest direction approach goes in the direction opposite to the gradient of the energy, $\Lambda_k = -\nabla E$. Unfortunately, this method may result in zig-zagging effects around the optimal solution which converge slowly. Alternatively, one can use nonlinear conjugate gradient methods, which searches in directions conjugate to the ones previously explored, in an attempt to converge more rapidly to the optimal solution. To avoid reaching a local minimum, we may introduce a resetting mechanism to the conjugate gradient method that allows it to move in the steepest direction when little improvement is achieved in a given conjugate direction. We compared the results of both steepest descent and conjugate gradient, as summarized in Fig.6 We see that conjugate gradients converge more rapidly to the optimal solution that steepest descent, making it more attractive for interactive systems.

4.3 Exploration of Transfer Function Space

Because there may be many different local minima in the parameter space, these iterative algorithms cannot guarantee to converge to the optimal solution. Rather than performing an exhaustive search, we offer the user a mechanism to explore the parameter space semi-automatically. The user is able to specify a given parameter, say the amplitude or deviation of a Gaussian bell, and search for the best solution in a given direction. The system will search in that direction until a local minima is found. Since only a single parameter is changed at a time, this is usually fast compared to re-optimization. Furthermore, the user can explore small variations of the opacity transfer function with little effort. This mechanism has been used to increase and decrease the opacity of specific isosurfaces without compromising much the visibility of important regions. See the accompanying video for examples.

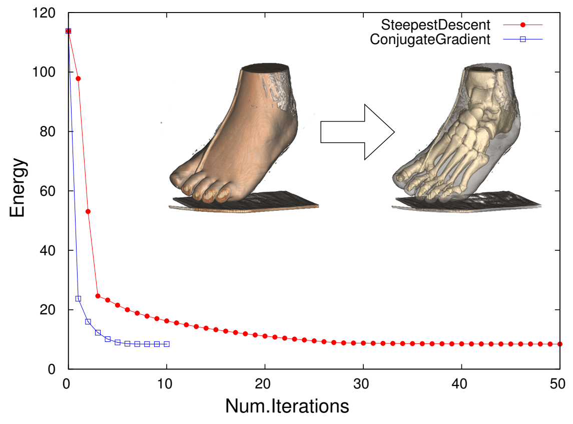


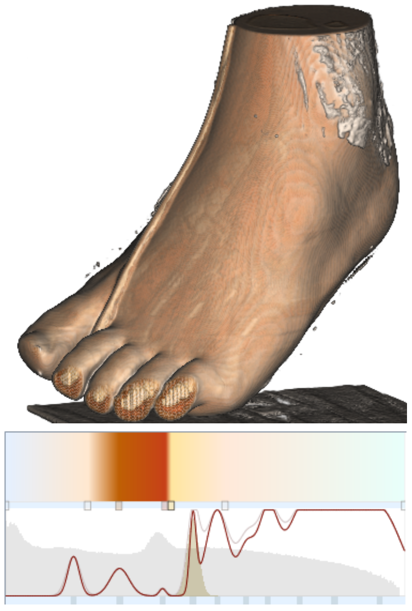
Figure 6: Convergence of different algorithms for transfer function optimization. We plot energy vs. number of iterations. Conjugate gradient reaches a good solution in less than 10 iterations while steepest descent requires up to 5 times the number of iterations.

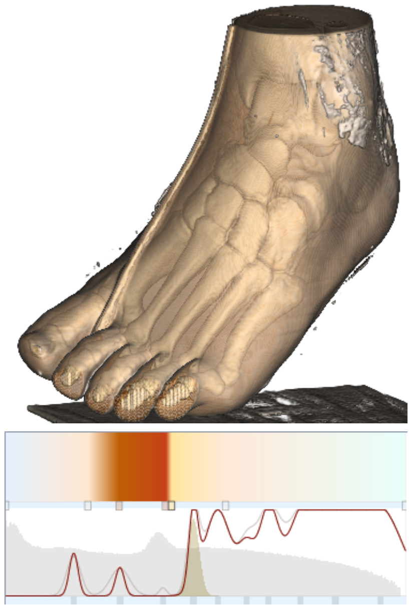
5 Results

We applied our approach on a number of datasets, including anatomical and flow simulation data. Fig. 7 shows the result of optimization for a foot dataset. From left to right, we show the classification when shifting the weight on Eq. 8 from user satisfaction to visibility. In Fig.7(a), the system performs small changes on the opacity function to match the user specification. However, bone tissues have little visibility. As the weights in the objective function become uniform (Fig.7(b)), the resulting opacity function now provides visibility of the bone tissue, while satisfying the user opacity function. Notice how the skin and some tissue are still represented, while we get a clear view of the bone. Finally, Fig.7(c) shows the case where the weights favor visibility over user satisfaction. Now the opacity of the skin is reduced considerably while flesh tissue is completely transparent. However, we obtain a high visibility of the bone tissue.

Fig. 8 shows the classification of a vortex dataset [22]. Initially, the user sets the opacity function using a pre-defined collection of Gaussian bells distributed along the data domain, and modifies the desired opacity of entire intervals. We notice that important isosurfaces (green to orange area) become occluded by the isosurfaces coded in blue. Fig.8(b) shows the result of automatic classification. The generated transfer function exhibits a falloff in opacity for the unimportant isosurfaces so that visibility is attained for the important intervals. The resulting image clearly shows the inner features while providing the outer shape of the features as a context. Fig.8(c) shows the effect of shifting the importance to the left (outer layers of features). Note how the outer layers in blue become more transparent, while the isosurfaces in green become visible.

Fig. 9 shows the classification of a supernova dataset. Initially, the user sets the opacity as a number of Gaussian bells modulated in a linear ramp, to highlight isosurfaces of increasing scalar value, in this case entropy (Fig.9a). After optimization, the resulting transfer function provides better visibility to the inner layers (Fig.9). Notice how the resulting opacity approaches an exponential curve, to counteract the multiplicative effect of attenuation. Fig. 9c shows the result after modifying the importance of certain intervals. After a few iterations, the resulting transfer function gives maximum visibility to the inner layer (red). Since visibility-driven transfer functions are viewpoint dependent, the user tries to adapt the new opacity mapping to a different viewpoint (Fig.9d). The system is able to increase the opacity of the isosurfaces in the blue-green-white region, since they do not affect much the visibility of the red feature that now appears unoccluded from this vantage point.


(a) $\beta_1 = 0.05, \beta_2 = 0.95$


(b) $\beta_1 = \beta_2 = 0.5$

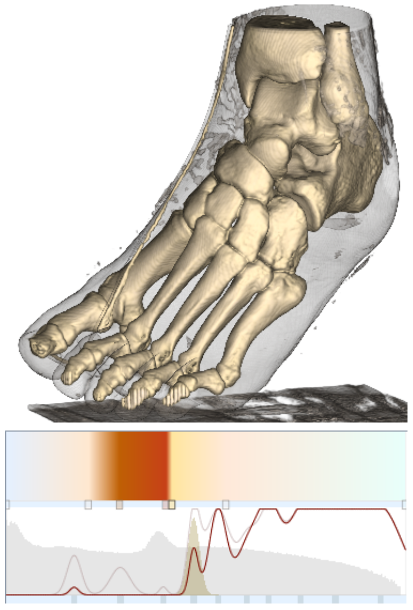

(c) $\beta_1 = 0.95, \beta_2 = 0.05$

Figure 7: Three results of automatic classification of a CT foot dataset, depending on the optimization weights. (a) Giving more weight to user satisfaction does not deviate much from the original user specification, but results in little visibility of important tissue (bone). (b) Uniform weighting achieves a balance between user satisfaction and visibility. Now bone tissue is visible, while skin and flesh tissue are still represented, although with little opacity. (c) Finally, giving more weight to visibility allows the system to decrease the opacity of occluding tissue.

5.1 Limitations

Our approach proves to be a powerful mechanism for exploring the transfer function space and avoid time-consuming trial and error iterations. However, the resulting transfer function is still bound by the intrinsic limitations of the transfer function space. For instance, 1D transfer functions may not help separate features of interest that share the same data values. Our approach will obtain a good transfer function given a viewpoint and an initial transfer function but cannot help separate these any further, since the result is still a 1D transfer function. Fortunately, our approach is not restricted to 1D transfer functions. For a 2D transfer function based on scalar values and gradient magnitude, we can derive a parameter space Θ that matches this space, for example, by including location and standard deviations in the y direction.

An aspect of our approach is the introduction of interpolated values in the VH. View-aligned rays sample the volume at positions other than voxel centers, introducing interpolated values in the histogram. Although these do not represent that actual values in the dataset, they describe the distribution of samples used for post-classification, similar to obtaining a histogram on a smoothed volume. Alternatives include using a shear-warp variant as in [2], or using fractional values when computing the histogram. Another limitation is the reliance on iterative algorithms to find the optimal transfer function. Since the problem space can be concave, these methods may converge to local minima. Finding the global minima may prove to be difficult and time-consuming, since it requires to know the energy function to all possible combinations of the parameter values. Higher-order algorithms, such as Newton-Raphson may also be explored.

6 Conclusions

We have presented the concept of visibility-driven transfer functions. We provided two methods, one manual and one automatic. The first makes use of visibility histograms, which encode the distribution of visibility values from a given viewpoint. The second method finds the best parameters of an opacity transfer function that maximizes visibility of important features. The use of visibility histograms alone prove to be an important element towards the understanding of complex datasets. On one hand, it provides feedback to the user regarding the effectiveness of an opacity transfer function. On the other hand, it gives cues about the structure of the dataset and the distribution of visibility helps discover intervals that result in more occlusion than others. The use of visibility histograms can also improve the understanding of other algorithms such as view selection and importance-driven/cutaway volume rendering. In both cases, the notion of visibility is essentially the same. Our GPU-based technique to compute visibility histograms will provide a better feedback of the inner workings of such techniques. The optimization approach was tested only on 1D transfer functions, but they can be extended to higher-dimensional transfer functions. Furthermore, one can incorporate lighting and view-dependent parameters to guide the creation of better illustrative visualizations. We believe that this approach can help us obtain cutaway and ghosted views of volumes semi-automatically, where the opacity of occluding surfaces is modified in a view-dependent manner to overcome the limitations of 1D transfer functions. Because our approach computes the visibility histogram on a view-oriented manner, much in the way it is done for volume rendering, we believe that this can be implemented and deployed in contemporary visualization systems with little effort.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants CNS- 0716691, OCI-0325934, OCI-0749217, CNS-0551727, CCF-0811422, CCF-0808896, OCI-0749227 and OCI-049321, and the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FC02-06ER25777 and DE-FG02-08ER54956.

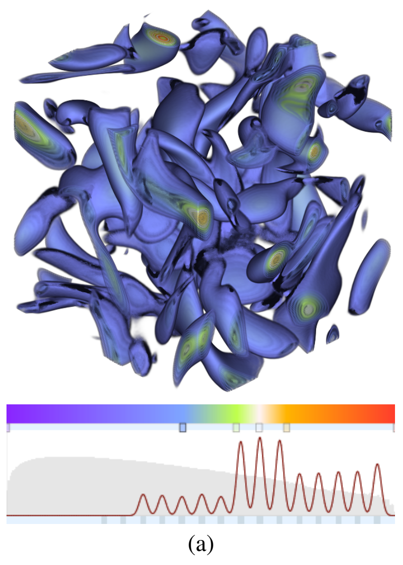
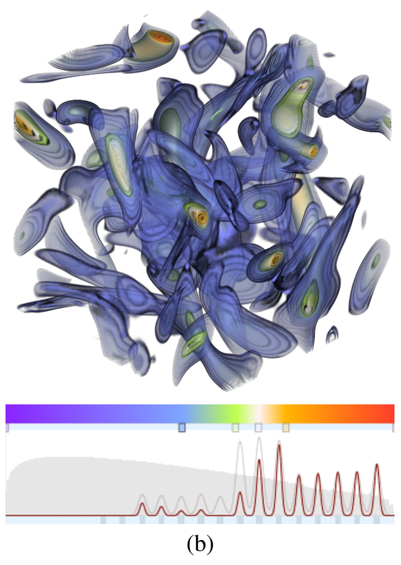
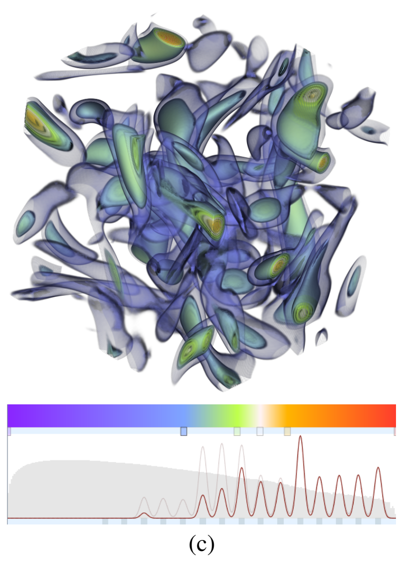




Figure 8: Classification of vortex data. (a) The user sets the opacity function using a pre-defined collection of Gaussian bells equally distributed along the data domain. We notice that important isosurfaces (green to orange area) become occluded by the isosurfaces coded in blue. (b) The automatically generated transfer function exhibits a falloff in opacity for the unimportant isosurfaces so that visibility is attained for the important intervals. The resulting image shows the inner features while still providing context. (c) The user now shifts the importance to the interval between blue and green isosurfaces. The outer layers in blue become more transparent, while the isosurfaces in green become visible.

References

[1] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *Proc. IEEE Visualization 1997*, pages 167–173, 1997.

[2] U. Bordoloi and H.-W. Shen. View selection for volume rendering. In *Proc. IEEE Visualization 2005*, pages 487–494, Oct. 2005.

[3] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Groller. Illustrative context-preserving exploration of volume data. *IEEE Trans. on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

[4] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive Cutaway Illustrations. *Computer Graphics Forum*, 22(3):523–532, 2003.

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

[6] O. Fluck, S. Aharon, D. Cremers, and M. Rousson. GPU histogram computation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research posters*, page 53, 2006.

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

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

[9] J. Hladůvka, A. König, and E. Gröller. Curvature-based transfer functions for direct volume rendering. In *Spring Conference on Computer Graphics 2000 (SCCG 2000)*, volume 16, pages 58–65, 2000.

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

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

[12] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proc. IEEE Visualization 2001*, pages 255–262, 2001.

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

[14] E. B. Lum and K.-L. Ma. Lighting transfer functions using gradient aligned sampling. In *Proc. IEEE Visualization 2004*, pages 289–296, 2004.

[15] C. Lundstrom, P. Ljung, and A. Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 12(6):1570–1579, 2006.

[16] 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. In *SIGGRAPH '97: Proc. Annual conference on Computer graphics and interactive techniques*, pages 389–400, 1997.

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

[18] C. Rezk-Salama and A. Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum*, 25:596–606, 2006.

[19] P. Rheingans and D. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Trans. on Vis. and Computer Graphics*, 7(3):253–264, 2001.

[20] S. Roettger, M. Bauer, and M. Stamminger. Spatialized transfer functions. In *Eurographics - IEEE VGTC Symposium on Visualization*, pages 271–278, 2005.

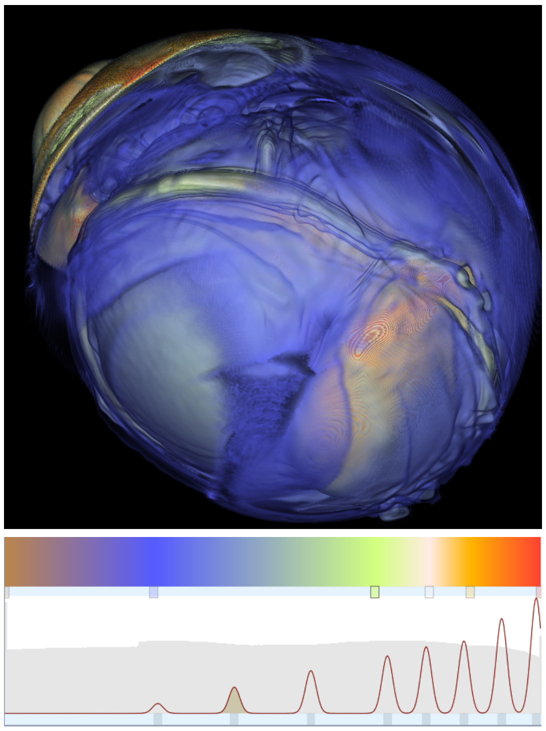
[21] T. Scheuermann and J. Hensley. Efficient histogram generation using scattering on GPUs. In *I3D '07: Proc. Symposium on Interactive 3D graphics and games*, pages 33–37, 2007.

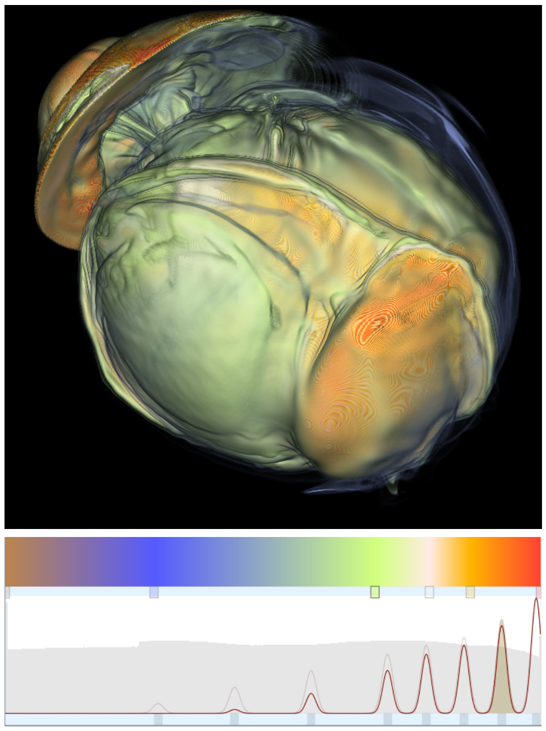
[22] D. Silver and X. Wang. Tracking and visualizing turbulent 3d features. *IEEE Trans. on Vis. and Computer Graphics*, 3(2):129–141, 1997.

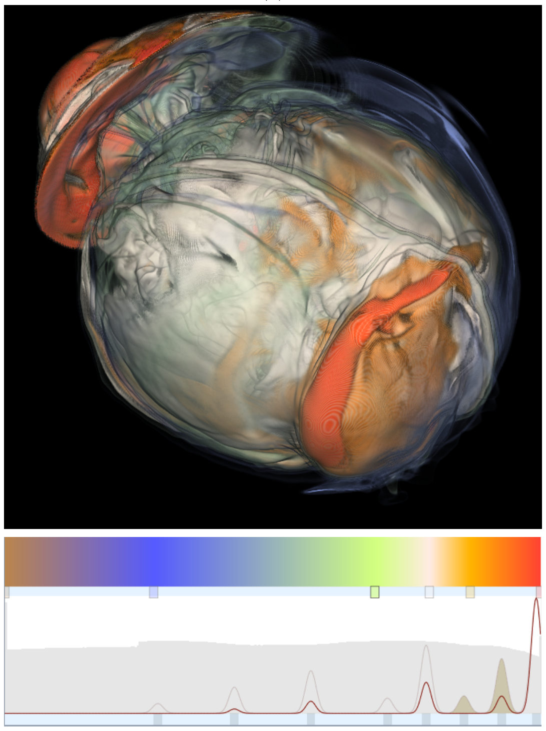
[23] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. In *Proc. IEEE Visualization 2005*, page 63, 2005.

[24] I. Viola, A. Kanitsar, and M. E. Groller. Importance-driven volume rendering. In *Proc. IEEE Visualization 2004*, pages 139–146, 2004.

[25] Y. Wu and H. Qu. Interactive transfer function design based on editing direct volume rendered images. *Visualization and Computer Graphics, IEEE Transactions on*, 13(5):1027–1040, Sept.-Oct. 2007.


(a)


(b)


(c)

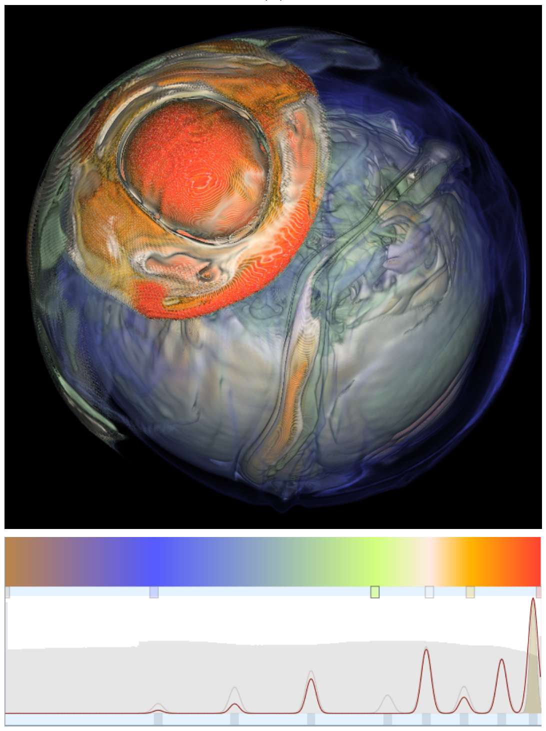

(d)

Figure 9: VDTF of the entropy of a supernova simulation. Highest entropy is red, the lowest blue. (a) The initial transfer function attempts to depict a series of isosurfaces whose importance (opacity) increases linearly. We can see that most of the important surfaces are occluded by the blue surface. (b) After optimization, the occluding surfaces are given low opacity, resulting in better visibility of the inner layers. The red layers are still difficult to observe, given that the user considers all of these intervals as important. (c) The user then reduces the expected opacity of the green and orange layers in an attempt to improve visibility. Rather than manually finding a good trade off, the user lets the system perform the search automatically. The resulting image now provides more visibility to the layers of interest (white and red). (d) Since VDTF are view-dependent, the user now selects a different viewpoint. The system lets the user explore the transfer function space. Increasing the opacity of unimportant layers (blue and green) is now possible since it does not affect much the visibility of the red layer, completely visible from this vantage point.