Visibility Histograms and Visibility-Driven Transfer Functions

Carlos D. Correa, *Member, IEEE,* and Kwan-Liu Ma, *Senior Member, IEEE*

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 general notion of visibility histograms, which are multi-dimensional graphical representations of the distribution of visibility in a volume-rendered image. In this paper, we explore the 1D and 2D transfer functions that result from intensity values and gradient magnitude. With the help of these histograms, users can manage a complex set of transfer function parameters that maximize the visibility of the intervals of interest and provide high quality images of volume data. We present a semi-automated method for generating transfer functions, which progressively explores the transfer function space towards the goal of maximizing visibility of important structures. Our methodology can be easily deployed in most visualization systems and can be used together with traditional 1D and 2D opacity transfer functions based on scalar values, as well as with other more sophisticated rendering algorithms.

Index Terms—Transfer Functions, Volume Rendering, Visibility, View-point dependent rendering, Histograms

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 rendering 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 [3], or to enhance the rendering process with ghost and cutaway views [33].

Visibility can be studied as a more fundamental quantity that measures the quality of transfer functions. In our previous work [7], we introduced the notion of *visibility histograms* as an interactive aid for generating effective transfer functions, which we called collectively *visibility-driven transfer functions*. These visibility histograms represent the contribution of each sample in the final resulting image. As such, these histograms are opacity dependent, as can be seen in Figure 1, and viewpoint-dependent, as seen in Figure 2. In this extended paper, we generalize the notion of visibility histograms along a number of dimensions, including multi-dimensional histograms, in particular 2D histograms that highlight boundaries (i.e., based on intensity and gradient magnitude) and viewpoint-independent visibility histograms, which encode the average visibility along multiple viewing directions.

One key property of these histograms is their ability to act as immediate feedback on the quality of the volume rendered image, as the user changes the rendering parameters (opacity, sampling distance, viewpoint, etc.) Therefore, a key challenge is the ability to compute them in real-time. In this paper, we present an efficient method for computing 2D visibility histograms in programmable graphic processing units (GPUs), using scattering operations. In our previous work, we were limited by 1D transfer functions using intensity alone. In this paper, we consider multiple dimensions for the creation of more effective transfer functions. On one hand, 2D visibility histograms provide immediate feedback for the handling of 2D histograms of intensity vs. gradient magnitude, helping the user understand how arcs in these histograms correspond to boundaries and their relative contribution to the final image. On the other hand, visibility guides the semi-automatic generation of transfer functions, replacing the tedious exploration of transfer function parameters with an automatic approach that maximizes the visibility of features of interest. As we incorporate more dimensions for classification and rendering, such as gradient magnitude, the exploration becomes more tedious. With a visibility-guided approach, the user does not need to handle them directly but only specify the desired visibility for a number of intervals of interest.

Examples of visibility histograms are shown in Figure 1. A red plot describes the opacity mapping, where the x-coordinate is the intensity value and the y-coordinate is opacity. The purple plot represents the visibility histogram. In Figure 1(a) we see that, even though the opacity mapping gives more importance to the bones, they are barely visible, due to the occluding muscle tissue. By reducing the opacity of occluding tissue, we get more visibility of veins and bones, as seen in Figure 1(a-right). Figure 1(b) shows the histogram for a

• *Carlos D. Correa and Kwan-Liu Ma are with the University of California, Davis. E-mail: {cdcorrea, ma}@cs.ucdavis.edu*

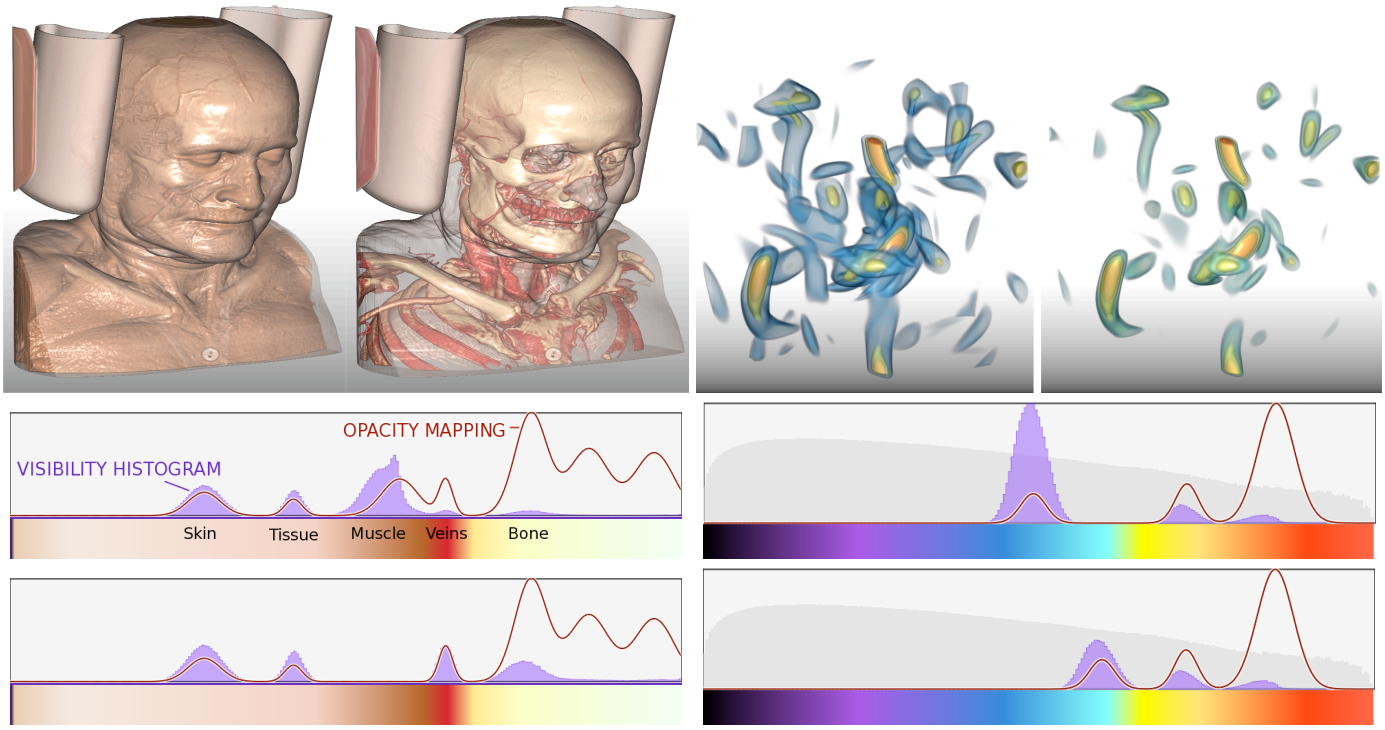


Fig. 1. Visibility histogram for two datasets. (a) The visibility histogram (purple plot) on a CT head data set reveals that the opacity mapping, as defined by the user (red line strip), cannot show bone structures clearly. This is due to muscle and tissue occluding underneath layers. The user then manipulates the opacity of the skin and muscle intervals until the bone tissue has enough visibility. The VH provides immediate feedback that helps the user converge to a solution. (b) For a vorticity simulation dataset, a different effect occurs. In this case, different intervals result in decreasing visibility, but no particular interval acts as a strong occluder. We see that the visibility peaks for the yellow and red isosurfaces do not change when moving the opacity from the blue to cyan intervals. This is often the case when visualizing quantities that vary smoothly, i.e., when there are no maxima in the gradient with respect to intensity value.

flow simulation dataset. Here, a different behavior is seen. Because the scalar field varies smoothly, no particular interval is a strong occluder. Notice how the visibility of the regions selected towards the right of the histogram are not affected by the change in the isosurface (from blue to cyan). Accordingly, the same intervals have relatively the same visibility in both images.

Our contributions are two-fold. On one hand, we present the general notion of a visibility histogram, which represents the visibility of the sample values from a given viewpoint. We explore several dimensions of visibility histograms, depending on the amount of information and dimensionality of their input parameters. For example, visibility histograms can be 1D (typically intensity value), 2D (intensity vs. gradient magnitude) or n-dimensional. Visibility histograms are typically view-dependent, when computed for a single projection camera, but can be omni-directional, when computed for a plenotpic camera model. 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 lets the user manipulate the parameter space of the transfer functions more intuitively. Users can initiate linear searches of specific parameters that converge to optimal solutions with respect to visibility, without requiring much manual intervention.

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. In addition, efficient classification of complex volume data usually requires extra dimensions, such as gradient magnitude, to isolate structures of interest. The semi-automatic design handles the task of exploring the extra dimensions transparently without the need for complex classification spaces or the user manual control of multi-dimensional widgets. 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.

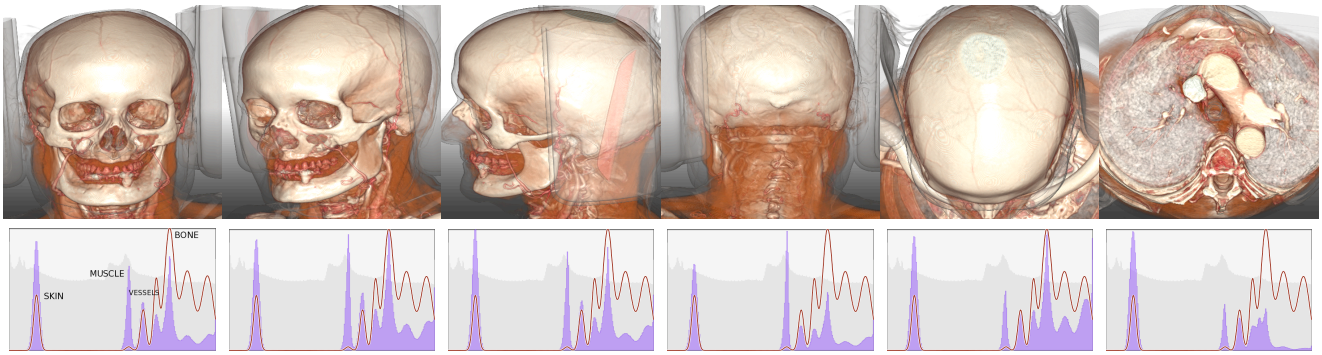


Fig. 2. View-dependent visibility histogram for a CT data set as the user changes view. Notice the increase and decrease of some of the intervals of interest (e.g., skin, muscle tissue or bones), depending on how visible they are.

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 [21], depending on whether they derive their parameters from the original data or the resulting images, respectively. 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 [17], [14] and curvature [13], [15]. 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. [16] 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 different opacity and lighting parameters [18]. Sereda et al. [29] generalize this notion and compute low and high values for each sample as the result of path tracing along the direction of the gradient. The resulting histogram, called the LH histogram, represents the boundaries of materials as blobs instead of arcs, which are also more robust to noise and bias. 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 [25]. Lundström et al. suggest the use of local histograms [19] to represent the spatial distribution of scalar samples. These histograms, however, are discrete projections of an otherwise continuous scalar or vector function. Carr [5] showed that, in fact, representing distribution via histograms assumes that the reconstruction follows nearest neighbor interpolation, and suggests isosurface statistics instead. This computation was later revisited by Scheidegger et al. [27], who weight the statistics by the gradient magnitude. A similar derivation was obtained by Bachthaler and Weiskopf [1], who pose the problem as a density transformation to define continuous scatterplots. In our paper, we treat histograms as discrete representations. However, we believe a visibility continuous scatterplot can be derived by considering a view-dependent density transformation and following their general approach [1].

Instead of extracting material boundaries, a number of techniques have been proposed that classify the local structure of volume data. These include shape-based [26], size-based [6] and topology-based [11] transfer functions. The exploration of the histograms that ensue with such dimensions has been rather limited and, in most cases, these methods demand new visual and interaction metaphors. As an alternative to histograms, Bajaj et al. propose the Contour Spectrum [2], which depicts a set of data attributes as a series of 1D plots for fast isosurfacing. To alleviate the complexity of handling multi-dimensional visual parameters, Tzeng et al. allow the users to "paint" directly on the volume and derive high-dimensional transfer functions using a neural network classifier [32]. Rezk-Salama et al. present high-level semantics that abstract parametric models of transfer functions [22]. In our paper, we follow a similar goal-oriented approach. However, instead of high-level semantics, we use a low-level quantity, visibility, to measure the quality of transfer functions. Through optimization, visibility-guided classification overcomes the need for managing a large number of rendering parameters.

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 [12]. 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 [20]. 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 [9]. 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 [34].

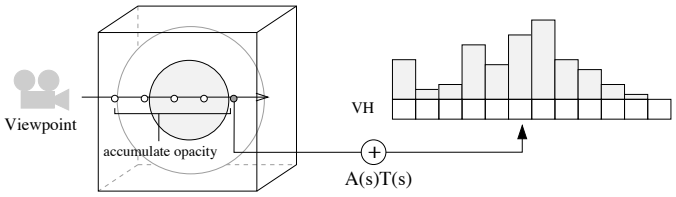


Fig. 3. Computation of visibility histograms. Given a viewpoint, the total opacity of a given sample, computed as the product of the original opacity and the transfer function and the accumulated opacity, is added to the corresponding bin in the histogram.

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 [12], [20], [34]. 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 [3]. 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. [31], 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. Furthermore, we extend the notion of visibility for viewpoint independent classification.

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 [24]. Interactive cutaways achieve visibility by removing occluding surfaces [8]. Importance-driven volume rendering achieves a similar effect by mapping the importance of features to levels of sparseness, which control the visibility of features [33]. Context-preserving volume rendering combines NPR operations with the accumulated opacity, or visibility, to reduce the opacity of unimportant objects [4]. 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 [23]. 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 can also be measured as the accumulated opacity of a sample p to the eye position E :

$$T(p) = e^{-\int_p^E \tau(t)dt} \tag{1}$$

where $\tau(t)$ is the attenuation coefficient of a sample, usually represented as an opacity transfer function A. Therefore, the visibility of a sample depends on the opacity of the sample, $A(X(p))$, usually defined by the user, and the viewpoint, which affects the accumulated opacity in front of the sample, where $X(p)$ is a classification value for point p, typically the scalar or intensity value $V(p)$.

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:

$$VH(x) = A(x) \int_{p \in \Omega} \delta(p,x) T(p) dp \tag{2}$$

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

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

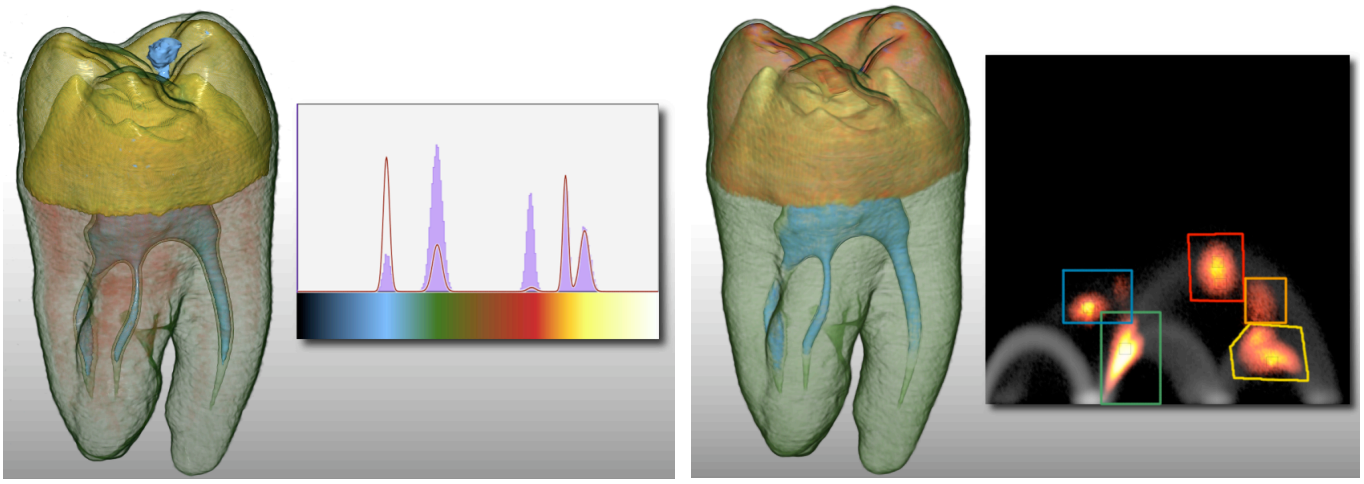
In practice, the histogram is computed at discrete bins, and the accumulated opacity is discretized with front-to-back compositing at discrete intervals. For a point p:

$$\begin{aligned} VH[x] &= VH[x] + T(p)A(x) \\ T(p+\Delta p) &= T(p)A(x) + (1 - T(p)) \end{aligned} \tag{4}$$

where $x = round(V(p)) \in [1,\ldots,N]$ are the quantized intensity values in the range $1,\ldots,N$. In the examples used throughout this paper, we use histograms of $N = 256$ bins.

Fig. 3 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 [3]. 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. 1(a), where muscle and fatty tissue dominate in terms of visibility. 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. 1(b) for a vorticity simulation data set. 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. Since the quantity (in this case vorticity) varies smoothly, we do not see the presence of a strong occluder.



(a) 1-D Visibility Histogram (b) 2-D Visibility Histogram

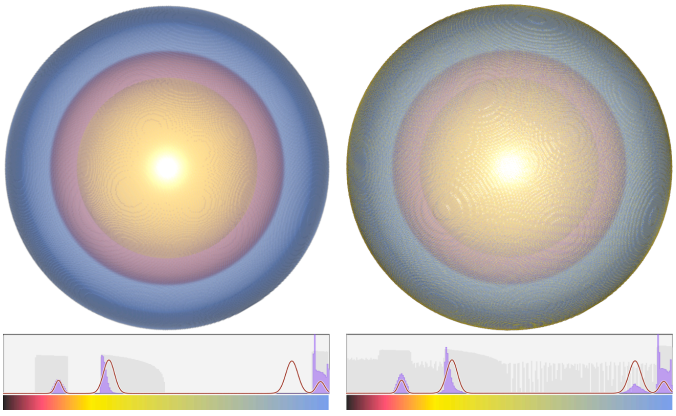
Fig. 4. Visibility Histograms for a tooth data set. (a) A 1-D Visibility Histogram shows that the green isosurface dominates visibility (highest purple peak). The interval in red, although with a lower opacity, has a significant relative visibility compared to other intervals. This shows the presence of a strong occluder (b) A 2-D Visibility Histogram for intensity and gradient magnitude as axes. Now, different structures can be more easily discernible. The 2D visibility histogram is depicted as a heat map, where bright colors indicate higher visibility and darker colors indicate low visibility.

3.1 Multi-dimensional Visibility Histograms

In general, transfer functions are multi-dimensional mappings from intensity and other variables, such as first and second derivatives to color and opacity. Therefore, in general, the opacity mapping can be defined $A : R^n \mapsto R$, for n dimensions, and the n-dimensional visibility histogram, as:

$$VH[x_1,\ldots,x_n] = VH[x_1,\ldots,x_n] + T(p)A(x_1,\ldots,x_n) \qquad (5)$$

for $(x_1,\ldots,x_n) = X(p)$, a vector of classification values for each point p. An example of such visibility histogram is for 2D transfer functions where one variable is intensity and the other is gradient magnitude. For higher-dimensional transfer functions, computing a visibility histogram becomes memory intensive and costly to compute. For this reason, we limit our implementation to 2D histograms. Figure 4 shows an example of a 1D and a 2D visibility histogram on a tooth CT data set. In Figure 4(b), two dimensions are plotted: intensity on the X-axis and gradient magnitude on the Y-axis. We can see a number of isosurfaces selected from the 2D data histogram (gray points) using mixtures of Gaussian widgets (the corresponding color is highlighted). The visibility histogram appears as a set of blobs encoded with a "heat" color map. The more intense blobs correspond to highly visible points. For example, the green isosurface dominates visibility. However, since we use gradient magnitude, only the boundary is rendered, allowing visibility of inner structures such as the blue isosurface. Other isosurfaces, such as the orange structure are occluded by the red and green isosurfaces and therefore the visibility is reduced. The visibility visualization provides an additional insight on the structure and shape of complex 2D histograms.



(a) Pre-Interpolation (b) Post-Interpolation

Fig. 5. (a) Pre-interpolated histograms corresponds to rendering using nearest neighbor interpolation. The third opacity peak does not exhibit visibility since there is nodata in such interval (b) Post-interpolated histograms show more visibility of intensities seemingly internal (yellow), due to the boundary added by interpolation. In addition, the third peak now exhibits visibility, corresponding to interpolated layers (green). These can only be seen in the post-interpolated data histogram (gray plot)

3.2 Pre- and Post-Interpolation Histograms

One of the issues with these visibility histograms is the handling of fractional values. Eq. (5) considers samples s along a ray, which may fall between two volume voxels with intensities a and b. If we assume linear interpolation, the visibility value will be added to the intensity value interpolated between a and b, $V(p) = wa + (1-w)b$, for an

interpolation weight w. In 3D, this is typically the product of tri-linear interpolation, as a weighted sum of the 8 nearest neighbors in 3D space. In the case of air-material boundaries, this introduces values that may not exist in the actual data. However, these are values that appear in the final rendered image, since the GPU-based renderer also interpolates values at fractional positions. We call these histograms *interpolated* visibility histograms. Although these represent accurately the actual intensities that are used in the rendered image, they may cause confusion about the actual distribution of visibility with respect to the intensity values present in the data set. As an alternative, one can force the visibility histogram to only consider intensity values present in the original data. There are two ways to do this. One, similar to the approach followed by Bordoloi et al. [3], is to align the samples as we gather the visibility histograms with voxel centers. Later on, we show that this can be achieved with a shear on the slices used to sample the visibility function. Another possibility is to include fractional histogram values. For a sample p, instead of adding the visibility to the interpolated intensity value $V(p)$, we add the fraction $wV(p)$ to the histogram bin corresponding to intensity a and $(1-w)V(p)$ to the histogram bin corresponding to b. Figure 5 shows the difference between interpolated and fractional visibility histograms for a sphere data set, where the higher intensities (blue) are in the outer layers of the sphere. With pre-interpolation, the visibility histogram is limited to those values present in the original data. For example, the third opacity peak does not get any visibility since there are no intensities within that interval in the original data. The visibility of the blue layers dominates and the yellow layers are barely visible. This histogram is similar to the distribution of visibility of rendering the volume data using nearest neighbor interpolation. In contrast, interpolated histograms also include the values introduced by the interpolation. We see that, although the blue layers still dominate visibility, the yellow layers appear more visible, since there is a very thin layer of yellow intervals due to the interpolation between the high values (blue) and empty space (zero value). Also, the third peak exhibits some visibility, accounting for the interpolated values in the final image (green layers). Although post-interpolated histograms introduce visibility of intensities that may not be in the original data, it represents accurately the distribution of visibility in the final image when using post-classification transfer functions. This can be seen in the post-interpolated data histogram (gray plot), which now shows the presence of intensity values along the entire spectrum, as introduced by interpolation. Throughout the paper, we rely on post-interpolation histograms.

3.3 Viewpoint-independent Visibility Histograms

The notion of visibility described thus far is inherently viewpoint dependent. Although this is important for finding the best image for a given view, it is also desirable to find a transfer function that can guarantee visibility of important features independently of the viewpoint. To this purpose we introduce two new types of histograms: *omni-directional* and *radial* visibility histograms.

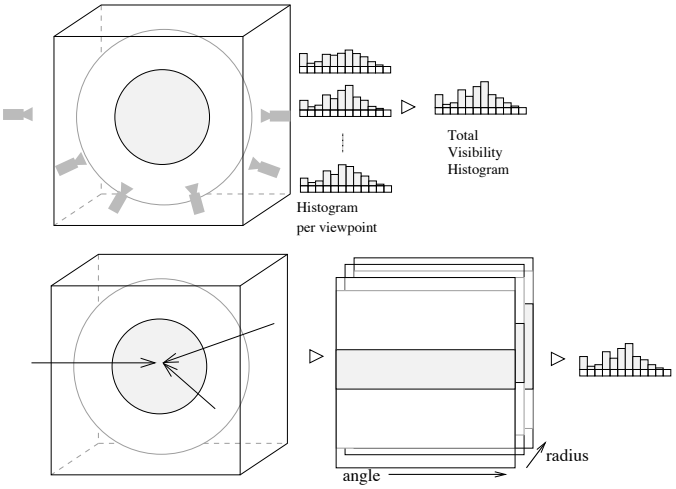


Fig. 6. Approach for viewpoint independent visibility histograms. Top: Omni-directional visibility histograms are obtained by summing the histograms of a number of viewpoints around the sphere (or cylinder) surrounding the volume. Bottom: Radial visibility histograms, a cheaper alternative, first transform the volume into spherical or cylindrical coordinates, then compute the histogram using the method described above.

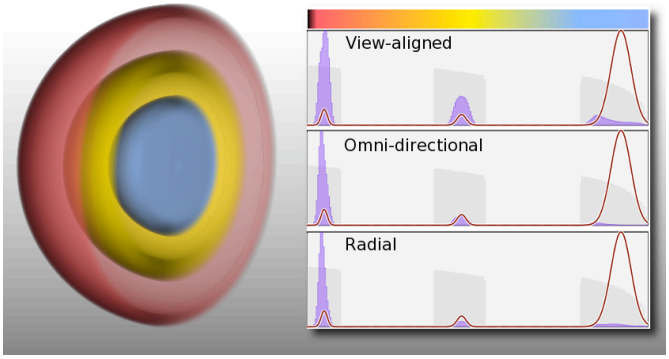


Fig. 7. View-aligned vs. View-independent visibility histograms.

3.3.1 Omni-directional Visibility Histogram (OVH)

An omni-directional visibility histogram encodes the distribution of visibility from all possible viewpoints. That is,

$$OVH(x) = \int_{\omega \in W} VH(x) \tag{6}$$

for all directions ω in an enclosing surface W, typically the circumscribing sphere. Some datasets, namely medical, have a clearly defined *up* direction, and viewpoints can be constrained to be defined in a circumscribing cylinder instead of a sphere. As shown in Figure 6(top), the approach to obtain such data set is to sum all the histograms along for a discrete sampling of all possible viewpoint directions. This, however, is an expensive operation. Fortunately, it only needs to be computed when the user changes the opacity function and not when the viewpoint is changed.

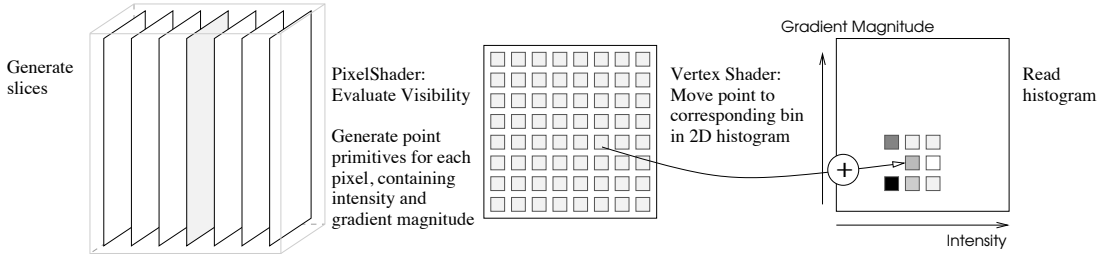


Fig. 8. GPU-based computation of 2D visibility histogram. First, we generate view-oriented slices. For each slice, in a pixel shader, we evaluate the visibility equation and keep the intensity and gradient magnitude values. For each of the pixels, we generate a point primitive. In a vertex shader, we transform the position of each of this point to 2D coordinates corresponding to the bin locations in the 2D histogram (i.e., the x coordinate is the intensity value and the y coordinate is the gradient magnitude). These values are added up for all slices and finally read back to the CPU as the visibility histogram.

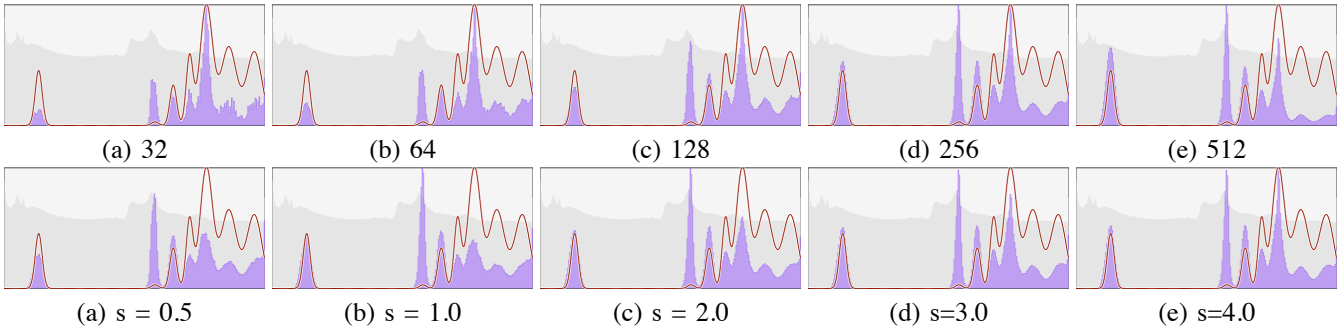


Fig. 9. Effect of window size and sampling resolution. The window size determines the dimensions of the view-aligned slices, while the resolution determines the number of slices along the view direction. We assume a base sampling of $1/512$ (i.e. 512 slices) and s the proportion of these slices used for histogram computation. As the size of the view-dependent slices decreases, the histogram is more prone to aliasing (top row). As the number of slices decreases, the true visibility of some intervals (approximated as the one with highest sampling), may be misrepresented. Notice, for example, the third peak from the left, which has more visibility than it appears to have in low resolution histograms.

3.3.2 Radial Visibility Histogram (RVH)

As an alternative to omni-directional visibility, one can also use radial visibility to obtain a viewpoint-independent histogram. In this case, we measure the visibility along radial rays, instead of view-aligned rays. This can be obtained easily by transforming the volume points from cartesian coordinates to spherical or cartesian coordinates. Therefore, the corresponding histograms become:

$$RVH(x) = A(x) \int_{q \in \Omega} \delta(q,x) T(q) |J_U| dq \qquad (7)$$

where $q = U(p)$ is the transformation of the samples to a new coordinate system defined by the transformation U, e.g., radius r, azimuth ϕ and angle θ for spherical and r and θ for cylindrical coordinates. $|J_U|$ is the determinant of the Jacobian of the transformation, which is $|J_U| = r^2 \sin\phi$ and $|J_U| = r$ for spherical and cylindrical coordinates, respectively. This ensures that the volume element is correctly scaled as we approach the center of the sphere or cylinder. In practice, we can do this by considering the visibility of a transformed volume where the coordinates x, y, z are replaced by cylindrical or spherical coordinates and computing the visibility histogram as defined above (taking into account the Jacobian of the transformation). This is depicted in Figure 6 (bottom).

One of the issues of radial visibility functions is that they may fail to capture the true visibility of structures that cannot be seen radially, but that a different viewpoint would show. The visibility will be misrepresented as zero. This is the case when the intensity value is concentrated in an eccentric region of the volume. In our experiments, we did not find a data set that had such characteristics. An approach to overcome this issue is to employ a hybrid visibility that combines both the OVH and RVH. This derives from the observation that the OVH is a sum of all the RVH that can be obtained by moving the center of the spherical or cylindrical coordinates, i.e.:

$$OVH(x) = \int_{p \in \Omega} RVH_p(x) dp \qquad (8)$$

where $RVH_p(x)$ is the radial visibility histogram with respect to a center point p in the volume domain Ω. Therefore, one can approximate the OVH by considering only a reduced number of center points p (e.g., 8, one for each octant).

An example of view-dependent and view-independent visibility histograms is shown in Figure 7 for a hemisphere data set. In this case, a view-aligned histogram shows visibility of the internal hemispheres (yellow and blue) since they are

visible from this view. However, they will not be visible for half of the viewpoints where the red hemisphere becomes an occluder. A transfer function designed from this viewpoint would require to be redesigned for a different viewpoint. A view-independent histogram circumvents this problem. Both omni-directional and radial VH capture the reduced visibility of the inner layers and the dominance of the red hemisphere. The RVH, however, over estimates the visibility of the blue intervals (notice that it is centered whereas the OVH is more skewed towards the lower intensities).

3.4 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, 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 the accumulated opacity, which is in turn required for the next slice.

3.4.1 Using Gathering

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. [10]. It 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.

3.4.2 Using Scattering

Another implementation uses scattering operations on the GPU. In this case, we exploit the vertex texture fetch capabilities of modern GPUs to scatter the pixel points to the right bin in the histogram. Since it allows us to place each point to the correct bin, we can extend it easily for 2D visibility histograms. The overall process is depicted in Figure 8. Our implementation is based on the approach by Scheuermann and Hensley [28]. Similar to the above technique, we first slice the volume into view-aligned slices. Each of the pixels in one of these slices contain both the intensity value and gradient magnitude. We then generate point primitives, one per pixel, which are translated in a vertex shader to the corresponding bin in the 2D histogram, i.e., the new x-coordinate becomes the intensity value and the y-coordinate is the gradient magnitude. Using blending, we can add up all the contributions for each slice, and the 2D histogram is transferred to the CPU by reading the screen area of a view port of size $N \times N$, where N is the number of bins. We use gradient magnitude as a second dimension due to its popularity and ease of computation. However, any variable can be mapped to the second dimension.

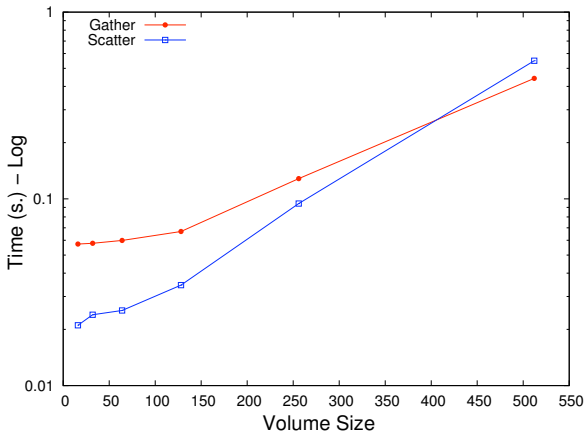


Fig. 10. Evaluation of GPU-based histogram algorithms. Timing comparison between gather (red) and scatter algorithm (green) in seconds (log scale) vs. size of view-dependent volume. Note that scatter algorithm is faster for sampling rates less than $1/512$.

Fig. 10(a) shows a comparison of these two methods in terms 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 [28] 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. Figure 9 summarizes the effects of slice size and slice resolution (reciprocal of number of slices). Small slice sizes make the histogram more prone to aliasing than a reduced number of slices. In both cases, the true visibility of certain intervals may be misrepresented.

4 Visibility-Driven Transfer Functions

As described above, visibility histograms provide the basis for generating visibility-driven transfer functions. In general, the approach for generating a transfer function can be seen as a way to maximize visibility of intensities of interest. This can incorporated in current visualization systems as a feedback mechanism for user-defined classification, or as a goal for optimization in semi-automated transfer function design.

4.1 User-defined 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 Fig. 1(a). In the first attempt to create a meaningful image, the user may set the opacity of bone to high and of skin and muscle to medium opacity. Despite this definition, the resulting image is not effective to show bone. This can be seen immediately in the visibility histogram. As a next step, the user can decrease the opacity of muscle tissue until sufficient visibility for the bone is obtained. The result in Fig.1(a-right) shows a better depiction of the veins and bone structures, as seen in the shape of the VH.

Furthermore, the VH is a useful tool for measuring the quality of volume rendered images. Although image quality is certainly the product of numerous factors, the visibility of structures of interest is at the core of volume rendering. Therefore, the selection of isosurfaces and rendering parameters can be justified in the grounds that they provide enough visibility of interesting features.

One of the issues with manual classification is that, despite being guided by visibility, the transfer functions can contain a large number of parameters that require to be managed individually. As an alternative, we present a more automatic approach.

4.2 Semi-automatic Transfer Functions

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.

We can formulate a transfer function as a parametric model of opacity $A(X(p),\Theta)$, where Θ is a vector of parameters, p is a sample position and $X(p)$ is a vector of sample values. For the case of 1D transfer functions, $X(p)$ is the scalar value at voxel p. In general, $X(p)$ is an n-dimensional vector containing all the classification dimensions, e.g., intensity and gradient magnitude. The quality of a transfer function can be seen as the reciprocal of a cost or energy function: $E:\Theta\mapsto R$, which is minimized when the transfer function is considered best. In this paper, we define the cost of a transfer function in terms of two components: user satisfaction and visibility.

Let us define $A_0(X(p))$ an initial transfer function, defined by the user or computed automatically using a gradient analysis as proposed in [14] or a topological decomposition [11].

We define the following energy components, designed to highlight certain desired aspects 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:

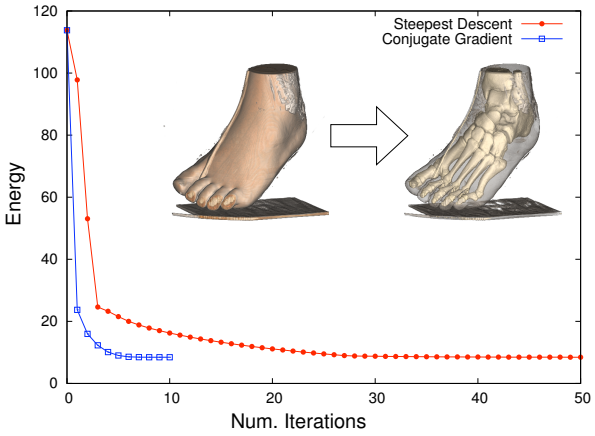


Fig. 11. Convergence of different algorithms for transfer function optimization. We plot energy vs. number of iterations. Conjugate gradient reaches a good solution in less than 10 iterations while steepest descent requires up to 5 times the number of iterations.

$$E_S(\Theta)=\sum_{p\in\Omega}\left(A(X(p),\Theta)-A_0(X(p))\right)^2 \tag{9}$$

where $A_0(X(p))$ is the opacity function defined by the user, and $A(X(p),\Theta)$ 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 A_0, we can weight the visibility of a sample by its opacity:

$$E_V(\Theta)=-\sum_{p\in\Omega}A_0(X(p))T(p,\Theta) \tag{10}$$

where $T(p,\Theta)$ is the visibility of point p, as defined in Eq.1, for the opacity mapping $A(X(p),\Theta)$. Note that the sign of this component is negative, which is minimized as the visibility of important values increases.

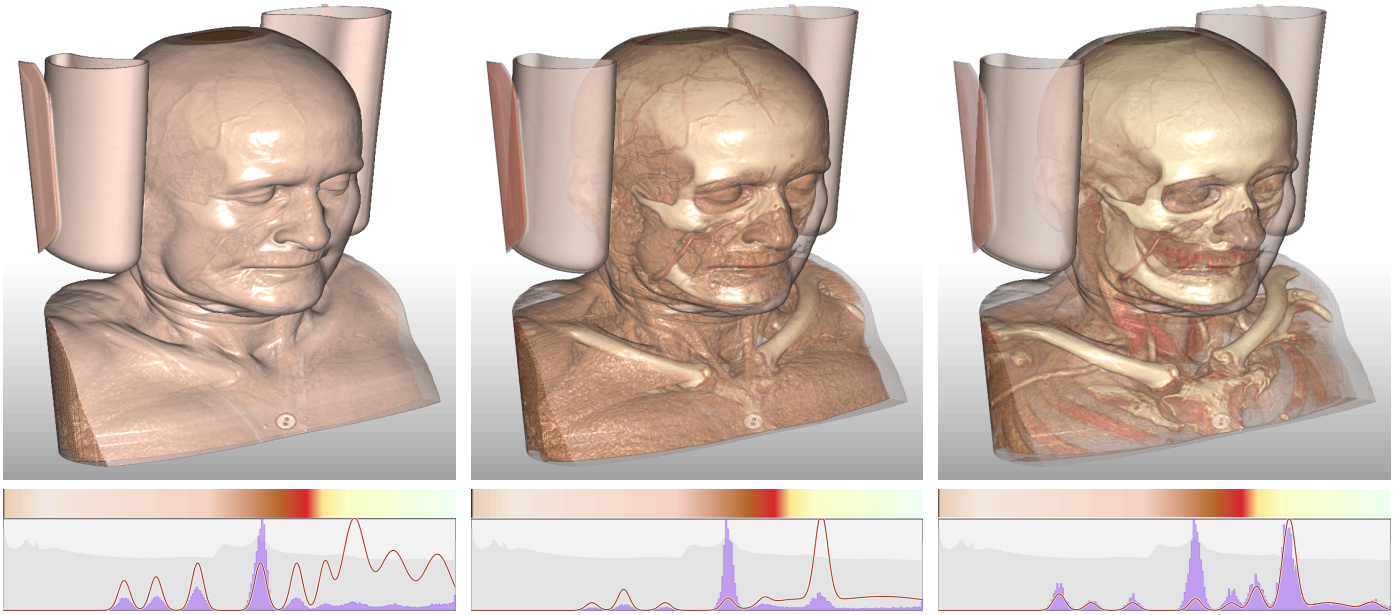
Constraints. Finally, we introduce constraints on the parameter space Θ. These constraints are used to control the minimum and maximum values opacities of value intervals in the final opacity function. This is particularly important for providing context in the resulting image. Without constraints, it 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:

$$E_C(\Theta)=\sum_{i\in||\Theta||}\left[\theta^i_{min}-\theta_i\right]^2_+ +\left[\theta_i-\theta^i_{max}\right]^2_+ \tag{11}$$

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

After examination of these components, we define a good transfer function as one that satisfies the minimization problem

$$\text{argmin}_\Theta\,\beta_1E_S(\Theta)+\beta_2E_V(\Theta)+\beta_3E_C(\Theta) \tag{12}$$



(a) $\beta_1 = 0.05, \beta_2 = 0.95$ (b) $\beta_1 = \beta_2 = 0.5$ (c) $\beta_1 = 0.95, \beta_2 = 0.05$

Fig. 12. Three results of automatic classification of a CT head dataset, depending on the optimization weights ($\beta_3 = 1$ for all three cases). (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 muscle 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.

where β_1, β_2 and β_3 are the weights for the matching, visibility and constraint terms, respectively.

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

$$A(x) = \sum_i \alpha_i G_{\mu_i,\sigma_i}(x) \tag{13}$$

where $G_{\mu,\sigma}(x)$ is a Gaussian function of mean μ and standard deviation σ. The parameter space corresponds to $\Theta = \{\alpha_i, \mu_i, \sigma_i : i \in [1 \ldots N]\}$, for N Gaussian functions. 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. Other parametric models can be considered, such as triangular, rectangular functions, linear ramps, etc.

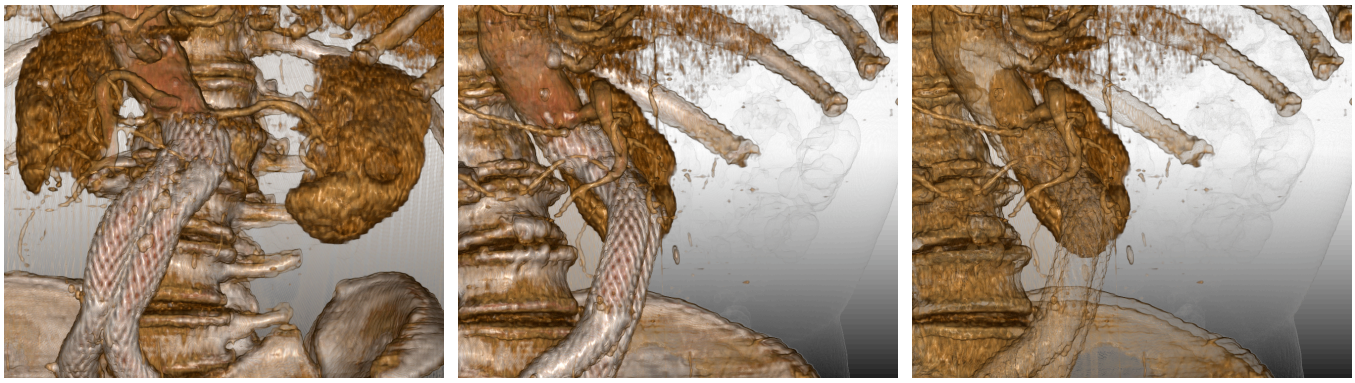
4.3 Optimization Algorithm

To solve the 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,

$$\Theta_{k+1} = \Theta_k + \gamma \Lambda_k \tag{14}$$

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 converges 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.11 We see that conjugate gradients converge more rapidly to the optimal solution that steepest descent, making it more attractive for interactive systems.

Fig. 12 shows the result of optimization for a CT head data set. From left to right, we show the classification when shifting the weight on Eq. 12 from user satisfaction to visibility. In Fig.12(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.12(b)), the resulting opacity function now provides visibility of the bone tissue, while satisfying the user opacity function. Notice how the skin and muscle are still represented, and we get a clearer view of bone structures, but the muscle layers still dominate. Finally, Fig.12(c) shows the case where the weights favor visibility over user satisfaction. Now the opacity of the skin is reduced considerably while muscle tissue is almost transparent, but we obtain a clear view of the bone.



(a) (b) (c)

Fig. 13. View-dependent nature of visibility-driven transfer functions for an abdominal CT data set (a) A transfer function is obtained automatically to maximize the visibility of the kidneys (b) when the user changes the viewpoint, the kidney is no longer visible. (c) A re-optimization yields a better depiction of the kidney, by decreasing the opacity of occluding vessels and the stent graft.

4.4 Transfer Function Refinement

Because there may be many different local minima in the parameter space, these iterative algorithms cannot guarantee to converge to the optimal solution. Furthermore, in many cases the user wants to explore the evolution of a number of isosurfaces in the context of other that do not change. We enable the user to initiate transfer function searches along individual directions, i.e., changing one parameter at a time. Instead of requiring to manually change the direction until visibility is maximized, the system finds the parameter that minimizes the energy cost in a given direction. Given a set of transfer function parameters Θ_0 and a direction $\delta\Theta$, the resulting transfer function is the one that results in a local minima in the energies $E(\Theta_0 + t\delta\Theta)$, for $t \in [1 \ldots n]$, where n is the maximum number of iterations. This mechanism has been used to increase and decrease the opacity of specific isosurfaces without compromising much the visibility of important regions.

4.5 Automatic Gradient Modulation

Since the optimization space can be defined along any set of parameters, we can include extra dimensions that may complicate the transfer function space. One example is gradient magnitude modulation, designed to highlight material boundaries without requiring to interact directly with a 2D transfer function. A common approach is to define gradient modulation as a global operation, where the opacity of a point p is defined as $A(V(p))||\nabla V(p)||^k$, where $V(p)$ is the scalar value and $||\nabla V(p))||$ is the magnitude of the scalar field gradient at point p, and k is a sensitivity parameter. In general, one would like to set an adaptive exponent, $K(x)$ (instead of a global k), depending on the intensity value x. Since this implies an additional dimension, the classification process doubles in complexity. For our semi-automatic approach, we can include this factor as an extra parameter in Θ to be considered in the optimization. Figure 12 and Figure 15(c) depict some of the results obtained when including gradient magnitude as a parameter.

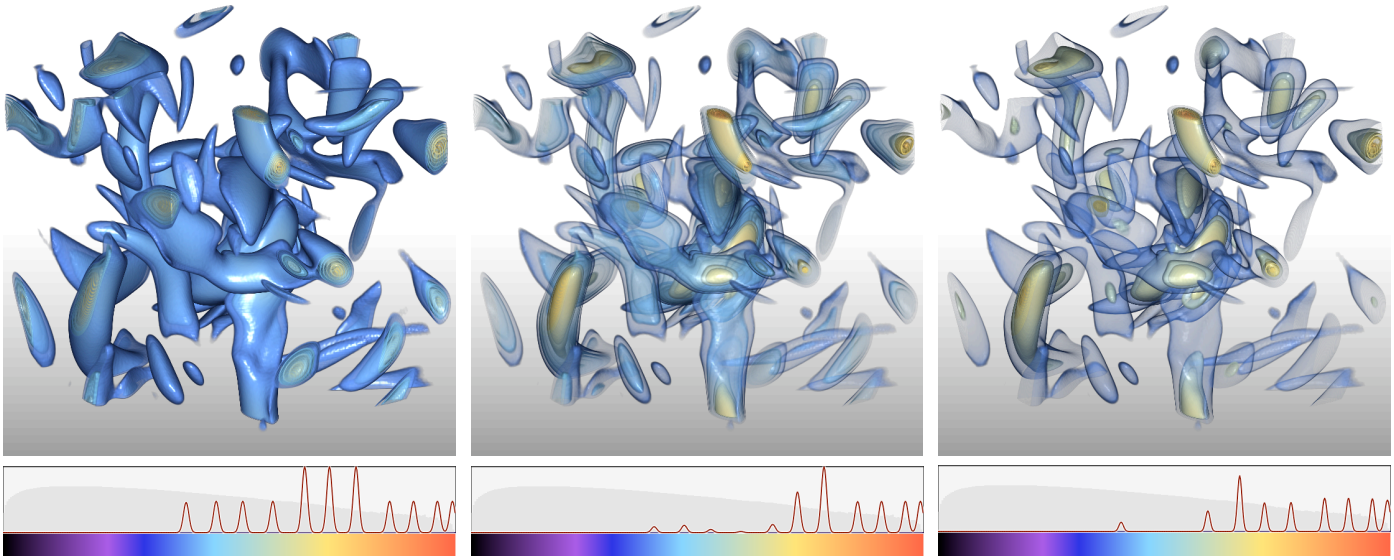
5 Results

We applied our approach on a number of datasets, including anatomical and flow simulation data.

Figure 13 shows a classification of an abdominal CT data set. This example shows the view-dependent nature of visibility histograms. First, we show the result of classification when most of the importance is given to the kidney (a), but when we change the view (b), the kidney is occluded by the aorta and stent graft. A re-optimization (c) yields a transfer function that highlights most of the kidney. In this case, we combined the intensity values to gradient magnitude to provide the best visibility. The user does not need to control any parameters for gradient magnitude modulation as it is done by the system in the hopes of improving visibility. Note that the visibility of the kidney and other structures with the same intensity is not sacrificed.

Fig. 14 shows the classification of a vortex dataset [30]. 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.14(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.14(c) shows the effect of shifting the importance to the left (outer layers of features). The transfer function is adapted interactively to reflect the change in importance. Note how the outer layers in blue become more transparent, while the isosurfaces in green become visible.

Fig. 15 shows the classification of a supernova data set. Scientists are interested in visualizing the turbulent structures near the core of the supernova in an effort to understand how these explosions are formed. 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



(a) (b) (c)

Fig. 14. 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.

entropy (Fig.15a). After optimization, the resulting transfer function provides better visibility to the inner layers (Fig.15b). In this case, we use a 1D transfer function based on intensity alone. This proves to be insufficient to visualize the internal flow. When we include gradient magnitude as an optimization parameter, the system can explore an additional dimension to improve the visibility. The result after optimization, as shown in Figure 15(c), does a better job in depicting the intricate relationships between the flow in yellow and blue. See the accompanying video for an interactive demonstration of our approach.

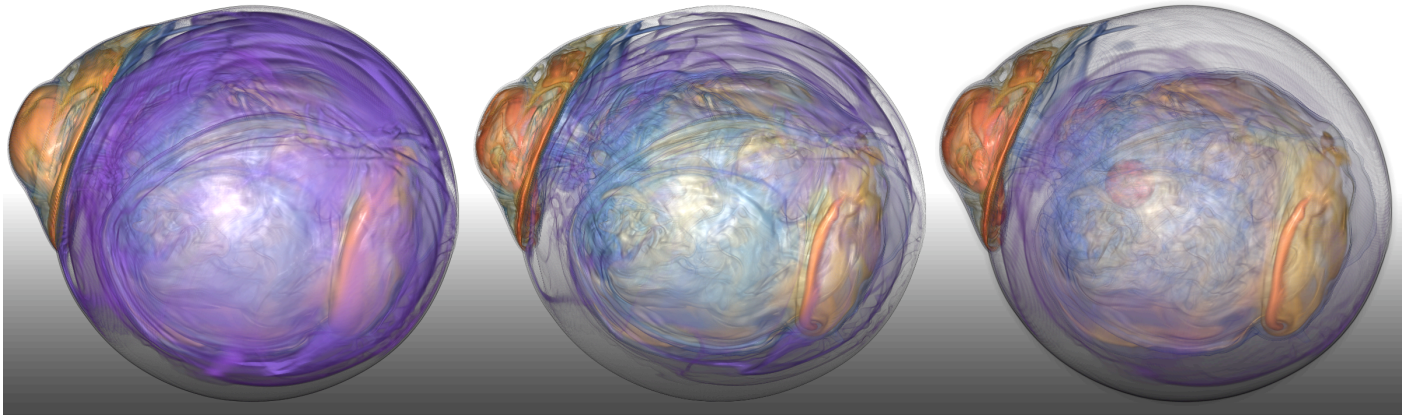
5.1 Limitations

The design of good transfer functions is still challenging. Even with additional dimensions, such as gradient magnitude and curvature, isolating structures of interest can be challenging. In medical imaging, for example, radiologists often turn to segmentation to extract features of interest. In this case, visibility needs to take into account the semantics of the data to provide better results. As seen in Figure 13(c), providing visibility of the kidney also highlights structures with similar intensity but that may not be important (e.g., in the spine). In addition, view-dependent effects, such as ghosted views and cutaways are often required to discard unimportant features that cannot be removed entirely with transfer functions. We believe that the optimization space can be easily extended to account for more sophisticated rendering techniques.

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 a general notion of visibility histograms and 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 proves 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



(a) (b) (c)

Fig. 15. Semi-automatic 1D and 2D transfer function for a supernova data set. (a) The initial transfer function, although gives more importance to inner flow, fails to show isosurfaces of interest near the supernova core. (b) Semi-automatic 1D transfer function generated using our approach. Although internal flow is more visibility, the reliance on a single dimension cannot improve the visibility of inner flow. (c) By considering an extra dimension, in this case gradient magnitude, we can obtain a better transfer function that now reveals the turbulent structure near the core. Semi-automatic generation alleviates the need for the user to tweak an extra dimension for each transfer function widget.

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.

ACKNOWLEDGMENTS

This research was supported by the U.S. National Science Foundation through grants CCF-0808896, OCI-0749227, OCI-0749217, CNS-0551727, OCI-0325934, OCI-0850566, CCF-0811422 and OCI-0905008, and the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FC02-06ER25777 and DOE-FG02-08ER54956. We want to thank the following for the data sets: OsiriX Foundation for the Head CTA dataset, Michael Meisner (Viatronix) for the stented abdominal aorta data set, GE Aircraft Engines for the tooth dataset, Deborah Silver for the turbulent flow dataset, the Terascale Supernova Initiative for the supernova entropy data set and Stefan Roettger for the bonsai data set and the volume data repository.

REFERENCES

[1] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1428–1435, 2008.

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

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

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

[5] H. Carr, D. Brian, and D. Brian. On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1259–1266, 2006.

[6] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380–1387, 2008.

[7] C. D. Correa and K.-L. Ma. Visibility-driven transfer functions. In *IEEE Pacific Visualization Symposium*, pages 177–184, 2009.

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] C. Rezk-Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, 2006.
[23] C. Rezk-Salama and A. Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum*, 25:596–606, 2006.
[24] P. Rheingans and D. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.
[25] S. Roettger, M. Bauer, and M. Stamminger. Spatialized transfer functions. In *Eurographics - IEEE VGTC Symposium on Visualization*, pages 271–278, 2005.
[26] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3d local intensity structure for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180, 2000.
[27] C. E. Scheidegger, J. M. Schreiner, B. Duffy, H. Carr, and C. T. Silva. Revisiting histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1659–1666, 2008.
[28] 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.
[29] P. Sereda, A. Vilanova Bartroli, I. W. O. Serlie, and F. A. Gerritsen. Visualization of boundaries in volumetric data sets using LH histograms. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):208–218, 2006.
[30] D. Silver and X. Wang. Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997.
[31] 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*, pages 495–502, 2005.
[32] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. A novel interface for higher-dimensional classification of volume data. In *Proc. IEEE Visualization 2003*, pages 505–512, 2003.
[33] I. Viola, A. Kanitsar, and M. E. Groller. Importance-driven volume rendering. In *Proc. IEEE Visualization 2004*, pages 139–146, 2004.
[34] Y. Wu and H. Qu. Interactive transfer function design based on editing direct volume rendered images. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1027–1040, Sept.-Oct. 2007.

Carlos D. Correa received the BS degree in computer science from EAFIT University, Colombia, in 1998, and the MS and PhD degrees in electrical and computer engineering from Rutgers University in 2003 and 2007, respectively. Currently, he is a postdoctoral researcher in the Department of Computer Science, University of California, Davis. His research interests include computer graphics, visualization and user interaction.

Kwan-Liu Ma is a professor of computer science and the chair of the Graduate Group in Computer Science (GGCS) at the University of California, Davis. He leads the VIDI research group and directs the DOE SciDAC Institute for Ultrascale Visualization, which involves researchers from three other universities and two DOE national laboratories. Professor Ma received his PhD degree in computer science from the University of Utah in 1993. His research interests include visualization, high-performance computing and user interface design. He is the paper chair of the IEEE Visualization Conference in 2008 and 2009. He is the founder of the IEEE Pacific Visualization Symposium. Professor Ma also serves on the editorial boards of the *IEEE Computer Graphics and Applications* and the *IEEE Transactions on Visualization and Graphics*. He is a senior member of the IEEE.