Multidimensional Transfer Functions for Volume Rendering

Article *in* IEEE Transactions on Visualization and Computer Graphics · August 2002

DOI: 10.1109/TVCG.2002.1021579 · Source: IEEE Xplore

3 authors, including:

Joe Kniss
University of New Mexico
47 PUBLICATIONS **2,880** CITATIONS

SEE PROFILE

Gordon Kindlmann
University of Chicago
82 PUBLICATIONS **5,842** CITATIONS

SEE PROFILE

Multi-Dimensional Transfer Functions for Interactive Volume Rendering

Joe Kniss Gordon Kindlmann Charles Hansen

Scientific Computing and Imaging Institute
School of Computing, University of Utah
{jmk|gk|hansen}@cs.utah.edu

1 Abstract

Most direct volume renderings produced today employ one-dimensional transfer functions, which assign color and opacity to the volume based solely on the single scalar quantity which comprises the dataset. Though they have not received widespread attention, multi-dimensional transfer functions are a very effective way to extract materials and their boundaries for both scalar and multivariate data. However, identifying good transfer functions is difficult enough in one dimension, let alone two or three dimensions. This paper demonstrates an important class of three-dimensional transfer functions for scalar data, and describes the application of multi-dimensional transfer functions to multivariate data. We present a set of direct manipulation widgets that make specifying such transfer functions intuitive and convenient. We also describe how to use modern graphics hardware to both interactively render with multi-dimensional transfer functions and to provide interactive shadows for volumes. The transfer functions, widgets, and hardware combine to form a powerful system for interactive volume exploration.

Keywords: volume visualization, direct volume rendering, multi-dimensional transfer functions, direct manipulation widgets, graphics hardware

2 Introduction

Direct volume rendering has proven to be an effective and flexible visualization method for three-dimensional (3D) scalar fields. Transfer functions are fundamental to direct volume rendering because their role is essentially to make the data visible: by assigning optical properties like color and opacity to the voxel data, the volume can be rendered with traditional computer graphics methods. Good transfer functions reveal the important structures in the data without obscuring them with unimportant regions. To date, transfer functions have generally been limited to one-dimensional (1D) domains, meaning that the 1D space of scalar data value has been the only variable to which opacity and color are assigned. One aspect of direct volume rendering which has received little attention is the use of multi-dimensional transfer functions.

Often, there are features of interest in volume data that are difficult to extract and visualize with 1D transfer functions. Many medical datasets created from CT or MRI scans contain a complex combination of boundaries between multiple materials. This situation is problematic for 1D transfer functions because of the potential for overlap between the data value intervals spanned by the different boundaries. When one data value is associated with multiple boundaries, a 1D transfer function is unable to render them in isolation. Another benefit of higher dimensional transfer functions is their ability to portray subtle variations in properties of a single boundary, such as its thickness. When working with multivariate data, a similar difficulty arises with features that can be identified only by their unique combination of multiple data values. A 1D transfer function is simply not capable of capturing this relationship.

Unfortunately, using multi-dimensional transfer functions in volume rendering is complicated. Even when the transfer function is only 1D, finding an appropriate transfer function is generally accomplished by trial and error. This is one of the main challenges in making direct volume rendering an effective visualization tool. Adding dimensions to the transfer function domain only compounds the problem. While this is an ongoing research area, many of the proposed methods for transfer function generation and manipulation are not easily extended to higher dimensional transfer functions. In addition, fast volume rendering algorithms that assume the transfer function can be implemented as a linear lookup table (LUT) can be difficult to adapt to multi-dimensional transfer functions due to the linear interpolation imposed on such LUTs.

A previous paper [19] demonstrated the importance and power of multi-dimensional transfer functions. This paper extends that work with a more detailed exposition of the multi-dimensional transfer function concept, a generalization of multi-dimensional transfer functions for both scalar and multivariate data, as well as a novel technique for the interactive generation of volumetric shadows. To resolve the potential complexities in a user interface for multi-dimensional transfer functions, we introduce a set of direct manipulation widgets which make finding and experimenting with transfer functions an intuitive, efficient, and informative process. In order to make this process genuinely interactive, we exploit the fast rendering capabilities of modern graphics hardware, especially 3D texture memory and pixel texturing operations. Together, the widgets and the hardware form the basis for new interaction modes which can guide users towards transfer function settings appropriate for their visualization and data exploration interests.

3 Previous Work

3.1 Transfer Functions

Even though volume rendering as a visualization tool is more than ten years old, only recently has research focused on making the space of transfer functions easier to explore. He *et al.* [12] generated transfer functions with genetic algorithms driven either by user selection of thumbnail renderings, or some objective image fitness function. The Design Gallery [26] creates an intuitive interface to the entire space of all possible transfer functions based on automated analysis and layout of rendered images. A more data-centric approach is the Contour Spectrum [1], which visually summarizes the space of isosurfaces in terms of metrics like surface area and mean gradient magnitude, thereby guiding the choice of isovalue for isosurfacing, and also providing information useful for transfer function generation. Another recent paper [21] presents a novel transfer function interface in which small thumbnail renderings are arranged according to their relationship with the spaces of data values, color, and opacity.

The application of these methods is limited to the generation of 1D transfer functions, even though 2D transfer functions were introduced by Levoy in 1988 [25]. Levoy introduced two styles of transfer functions, both two-dimensional, and both using gradient magnitude for the second dimension. One transfer function was intended for the display of interfaces between materials, the other

for the display of isovalue contours in more smoothly varying data. The previous work most directly related to our approach for visualizing scalar data facilitates the semi-automatic generation of both 1D and 2D transfer functions [17, 32]. Using principles of computer vision edge detection, the semi-automatic method strives to isolate those portions of the transfer function domain which most reliably correlate with the middle of material interface boundaries. Other work closely related to our approach for visualizing multivariate data uses a 2D transfer function to visualize data derived from multiple MRI pulse sequences [23].

Scalar volume rendering research that uses multi-dimensional transfer functions is relatively scarce. One paper discusses the use of transfer functions similar to Levoy's as part of visualization in the context of wavelet volume representation [30]. More recently, the VolumePro graphics board uses a 12-bit 1D lookup table for the transfer function, but also allows opacity modulation by gradient magnitude, effectively implementing a separable 2D transfer function [31]. Other work involving multi-dimensional transfer functions uses various types of second derivatives in order to distinguish features in the volume according to their shape and curvature characteristics [15, 37].

Designing colormaps for displaying non-volumetric data is a task similar to finding transfer functions. Previous work has developed strategies and guidelines for colormap creation, based on visualization goals, types of data, perceptual considerations, and user studies [3, 35, 39].

3.2 Direct Manipulation Widgets

Direct manipulation widgets are geometric objects rendered with a visualization and are designed to provide the user with a 3D interface [5, 14, 34, 38, 41]. For example, a frame widget can be used to select a 2D plane within a volume. Widgets are typically rendered from basic geometric primitives such as spheres, cylinders, and cones. Widget construction is often guided by a constraint system which binds elements of a widget to one another. Each sub-part of a widget represents some functionality of the widget or a parameter to which the user has access.

3.3 Hardware Volume Rendering

Many volume rendering techniques based on graphics hardware utilize texture memory to store a 3D dataset. The dataset is then sampled, classified, rendered to proxy geometry, and composited. Classification typically occurs in hardware as a 1D table lookup.

2D texture-based techniques slice along the major axes of the data and take advantage of hardware bilinear interpolation within the slice [4]. These methods require three copies of the volume to reside in texture memory, one per axis, and they often suffer from artifacts caused by under-sampling along the slice axis. Trilinear interpolation can be attained using 2D textures with specialized hardware extensions available on some commodity graphics cards [6]. This technique allows intermediate slices along the slice axis to be computed in hardware. These hardware extensions also permit diffuse shaded volumes to be rendered at interactive frame rates.

3D texture-based techniques typically sample view-aligned slices through the volume, leveraging hardware trilinear interpolation [11]. Other proxy geometry, such as spherical shells, may be used with 3D texture methods to eliminate artifacts caused by perspective projection [24]. The pixel texture OpenGL extension has been used with 3D texture techniques to encode both data value and a diffuse illumination parameter which allows shading and classification to occur in the same look-up [28]. Engel *et al.* showed how to significantly reduce the number of slices needed to adequately sample a scalar volume, while maintaining a high quality rendering, using a mathematical technique of pre-integration and hardware extensions such as dependent textures [10].

Another form of volume rendering graphics hardware is the Cube-4 architecture [33] and the subsequent VolumePro PCI graphics board [31]. The VolumePro graphics board implements ray casting combined with the shear warp factorization for volume rendering [22]. It features trilinear interpolation with supersampling, gradient estimation, shaded volumes, and provides interactive frame rates for scalar volumes with sizes up to 512^3.

4 Multi-Dimensional Transfer Functions

Transfer function specification is arguably the most important task in volume visualization. While the transfer function's role is simply to assign optical properties such as opacity and color to the data being visualized, the value of the resulting visualization will be largely dependent on how well these optical properties capture features of interest. Specifying a good transfer function can be a difficult and tedious task for several reasons. First, it is difficult to uniquely identify features of interest in the transfer function domain. Even though a feature of interest may be easily identifiable in the spatial domain, the range of data values that characterize the feature may be difficult to isolate in the transfer function domain due to the fact that other, uninteresting regions, may contain the same range of data values. Second, transfer functions can have an enormous number of degrees of freedom. Even simple 1D transfer functions using linear ramps require two degrees of freedom per control point. Third, typical user interfaces do not guide the user in setting these control points based on dataset specific information. Without this type of information, the user must rely on trial and error. This kind of interaction can be especially frustrating since small changes to the transfer function can result in surprisingly large and unintuitive changes to the volume rendering.

Rather than classifying a sample based on a single scalar value, multi-dimensional transfer functions allow a sample to be classified based on a combination of values. Multiple data values tend increase the probability that a feature can be uniquely isolated in the transfer function domain, effectively providing a larger vocabulary for expressing the differences between structures in the dataset. These values are the axes of a multi-dimensional transfer function. Adding dimensions to the transfer function, however, greatly increases the degrees of freedom necessary for specifying a transfer function and the need for dataset specific guidance.

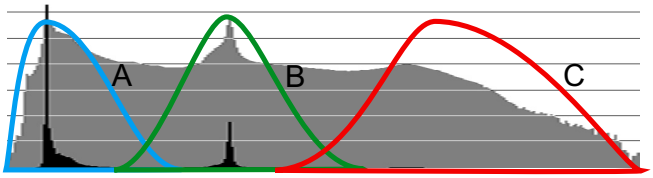
In the following sections, we demonstrate the application of multi-dimensional transfer functions to two distinct classes of data: scalar data and multivariate data. The scalar data application is focused on locating surface boundaries in a scalar volume. We motivate and describe the axes of the multi-dimensional transfer function for this type data. We then describe the use of multi-dimensional transfer functions for multivariate data. We use two examples, color volumes and meteorological simulations, to demonstrate the effectiveness of such transfer functions.

4.1 Scalar Data

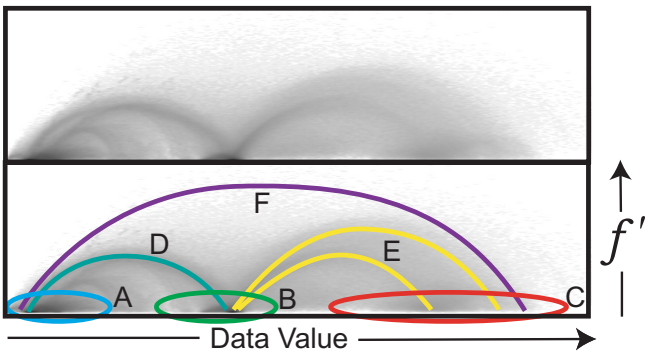
For scalar data, the gradient is a first derivative measure. As a vector, it describes the direction of greatest change. The normalized gradient is often used as the normal for surface based volume shading. The gradient magnitude is a scalar quantity which describes the local rate of change in the scalar field. For notational convenience, we will use f' to indicate the magnitude of the gradient of f, where f is the scalar function representing the data:

$$f' = \|\nabla f\| \tag{1}$$

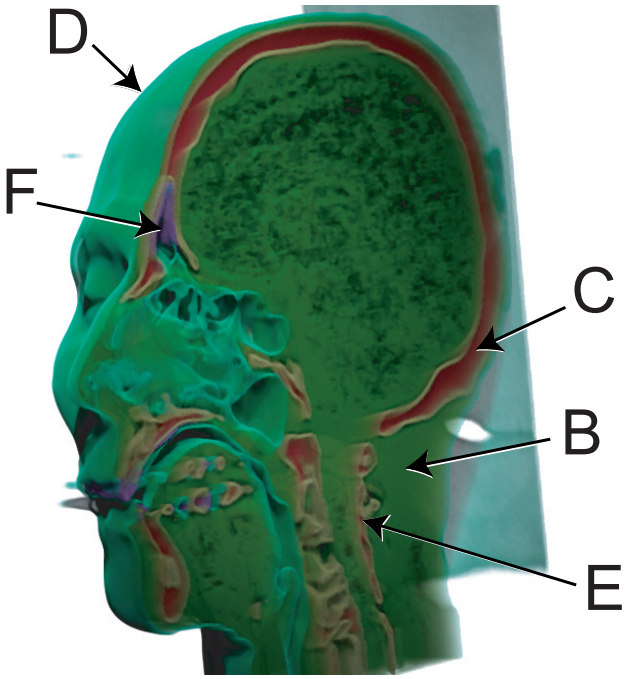
This value is useful as an axis of the transfer function since it discriminates between homogeneous regions (low gradient magnitudes) and regions of change (high gradient magnitudes). This



(a) A 1D histogram. The black region represents the number of data value occurrences on a linear scale, the grey is on a log scale. The colored regions (A,B,C) identify basic materials.

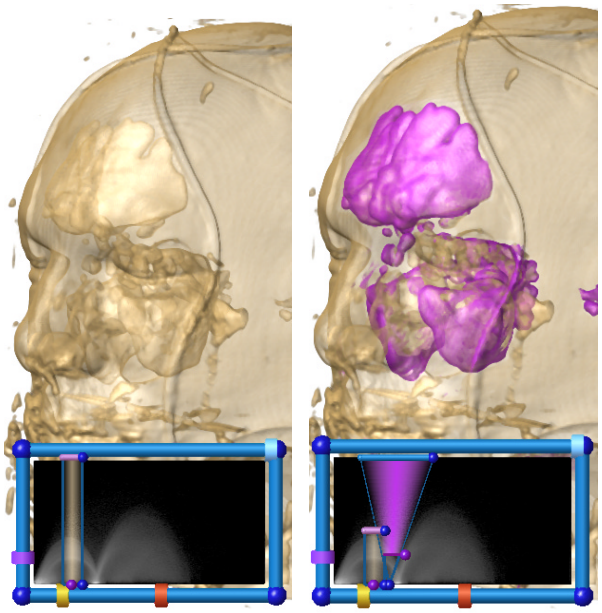


(b) A log-scale 2D joint histogram. The lower image shows the location of materials (A,B,C), and material boundaries (D,E,F).



(c) A volume rendering showing all of the materials and boundaries identified above, except air (A), using a 2D transfer function.

Figure 1: Material and boundary identification of the Chapel Hill CT Head with data value alone (a) versus data value and gradient magnitude (*f'*), seen in (b). The basic materials captured by CT, air (A), soft tissue (B), and bone (C) can be identified using a 1D transfer function as seen in (a). 1D transfer functions, however, cannot capture the complex combinations of material boundaries; air and soft tissue boundary (D), soft tissue and bone boundary (E), and air and bone boundary (F) as seen in (b) and (c).



(a) 1D transfer function (b) 2D transfer function

Figure 2: The frontal and maxillary sinuses of the Visible Male CT. While a 1D transfer function can show the sinuses along with the skin, it cannot capture them in isolation. Only a higher dimensional transfer function, in this case a 2D transfer function using data value and gradient magnitude, can uniquely classify them.

effect can be seen in Figure 1. Figure 1(a) shows a 1D histogram based on data value and identifies the three basic materials in the Chapel Hill CT Head; air (A), soft tissue (B), and bone (C). Figure 1(b) shows a log-scale joint histogram of data value versus gradient magnitude. Since materials are relatively homogeneous, their gradient magnitudes are low. They can be seen as the circular regions at the bottom of the histogram. The boundaries between the materials are shown as the arches; air and soft tissue boundary (D), soft tissue and bone boundary (E), and air and bone boundary (F). Each of these materials and boundaries can be isolated using a 2D transfer function based on data value and gradient magnitude. Figure 1(c) shows a volume rendering with the corresponding features labeled. The air/bone boundary, (F) in Figure 1 is a good example of a surface which cannot be isolated using a simple 1D transfer function. This type of boundary appears in CT datasets as the sinuses and mastoid cells. Figure 2 compares attempts at emphasizing the frontal and maxillary sinuses of the Visible Male CT using a 1D transfer function and a 2D transfer function.

Often, the arches that define material boundaries in a 2D transfer function overlap. In some cases this overlap prevents a material from being properly isolated in the transfer function. This effect can be seen in the circled region of the 2D data value/gradient magnitude joint histogram of the human tooth CT in Figure 3(a). The background/dentin boundary (F) shares the same ranges of data value and gradient magnitude as portions of the pulp/dentin (E) and the background/enamel (H) boundaries. When the background/dentin boundary (F) is emphasized in a 2D transfer function, the boundaries (E) and (H) are erroneously colored in the volume rendering, as seen in Figure 3(c). A second derivative measure enables a more precise disambiguation of complex boundary configurations such as this. Some edge detection algorithms (such as Marr-Hildreth [27]) locate the middle of an edge by detecting

a zero-crossing in a second derivative measure, such as the Laplacian. We compute a more accurate but computationally expensive measure, the second directional derivative along the gradient direction, which involves the Hessian (**H**), a matrix of second partial derivatives. We will use f'' to indicate this second derivative.

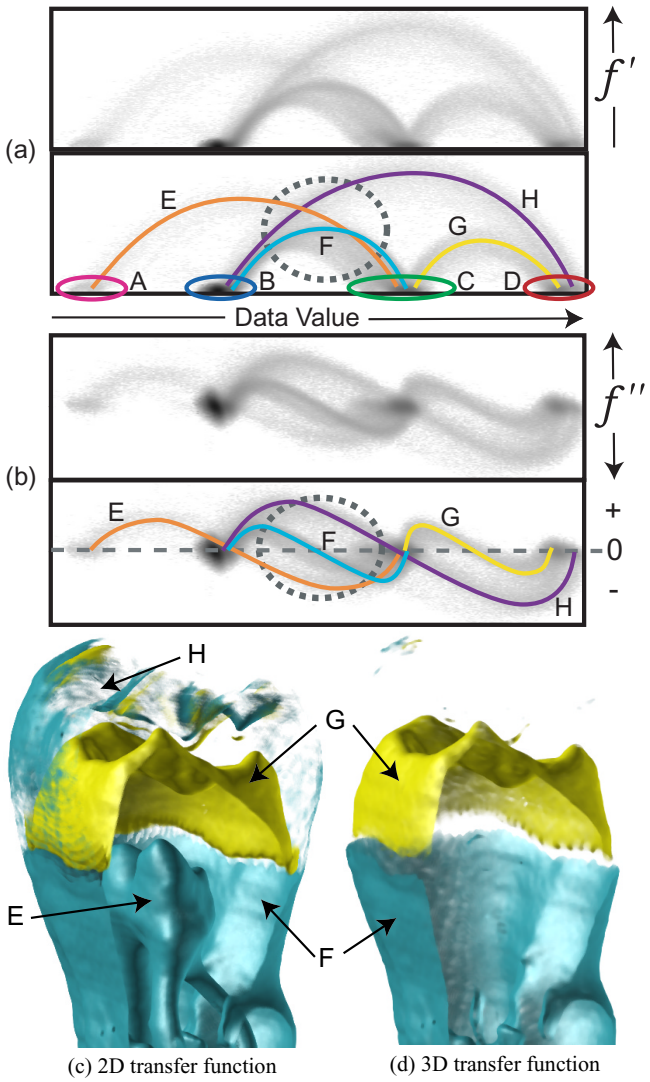
$$f'' = \frac{1}{\|\nabla f\|^2}(\nabla f)^{\mathrm{T}}\mathbf{H}f\nabla f \qquad (2)$$

More details on these measurements can be found in previous work on semi-automatic transfer function generation [16, 17]. Figure 3(b) shows a joint histogram of data value verses this second directional derivative. Notice that the boundaries (E), (F), and (G) no longer overlap. By reducing the opacity assigned to non-zero second derivative values, we can render the background/dentin boundary in isolation, as seen in Figure 3(d). The relationship between data value, gradient magnitude, and the second directional derivative is made clear in Figure 4. Figure 4(a) shows the behavior of these values along a line through an idealized boundary between two homogeneous materials (inset). Notice that at the center of the boundary, the gradient magnitude is high and the second derivative is zero. Figure 4(b) shows the behavior of the gradient magnitude and second derivative as a function of data value. This shows the curves as they appear in a joint histogram or a transfer function.

4.2 Multivariate data

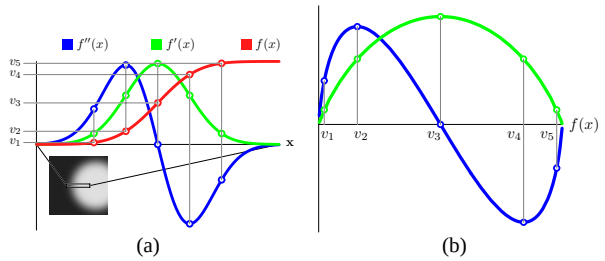
Multivariate data contains, at each sample point, multiple scalar values that represent different simulated or measured quantities. Multivariate data can come from numerical simulations which calculate a list of quantities at each timestep, or from medical scanning modalities such as MRI, which can measure a variety of tissue characteristics, or from a combination of different scanning modalities, such as MRI, CT, and PET. Multi-dimensional transfer functions are an obvious choice for volume visualization of multivariate data, since we can assign different data values to the different axes of the transfer function. It is often the case that a feature of interest in these datasets cannot be properly classified using any single variable by itself. In addition, we can compute a kind of first derivative in the multivariate data in order to create more information about local structure. As with scalar data, the use of a first derivative measure as one axis of the multi-dimensional transfer function can increase the specificity with which we can isolate and visualize different features in the data.

One example of data that benefits from multi-dimensional transfer functions is volumetric color data. A number of volumetric color datasets are available, such as the Visible Human Project's RGB data. The process of acquiring color data by cryosection is becoming common for the investigation of anatomy and histology. In these datasets, the differences in materials are expressed by their unique spectral signature. A multi-dimensional transfer function is a natural choice for visualizing this type of data. Opacity can be assigned to different positions in the 3D RGB color space. Figure 5(a) shows a joint histogram of the RGB color data for the Visible Male; regions of this space that correspond to different tissues are identified. Regions (A) and (B) correspond to the fatty tissues of the brain, white and gray matter, as seen in Figure 5(b). In this visualization, the transition between white and grey matter is intentionally left out to better emphasize these materials and to demonstrate the expressivity of the multi-dimensional transfer function. Figure 5(c) shows a visualiation of the color values that represent the muscle structure and connective tissues (C) of the head and neck with the skin surface (D) given a small amount of opacity for context. In both of these figures, a slice of the original data is mapped to the surface of the clipping plane for reference. Figure 6 shows a visualization of the kidney from the Visible Male RGB data.



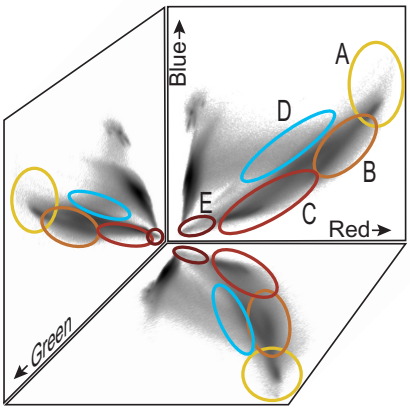
(c) 2D transfer function (d) 3D transfer function

Figure 3: Material and boundary identification of the human tooth CT with data value and gradient magnitude (*f'*), seen in (a), and data value and second derivative (*f"*), seen in (b). The background/dentin boundary (F) cannot be adequately captured with data value and gradient magnitude alone. (c) shows the results of a 2D transfer function designed to show only the background/detin (F) and dentin/enamel boundaries (G). The background/enamel (H) and dentin/pulp (E) boundaries are erroneously colored. Adding the second derivative as a third axis to the transfer function disambiguates the boundaries. (d) shows the results of a 3D transfer function that gives lower opacity to non-zero second derivative values.

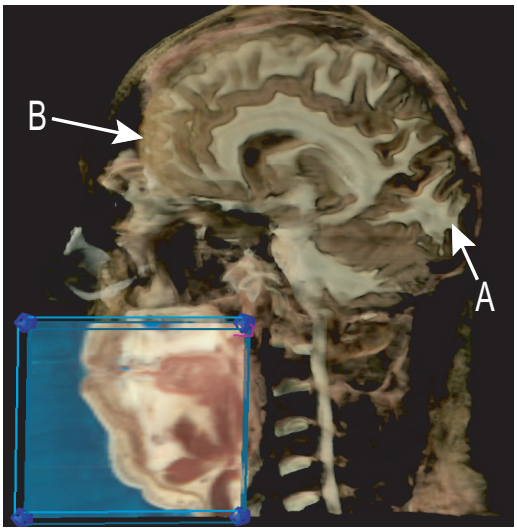


(a) (b)

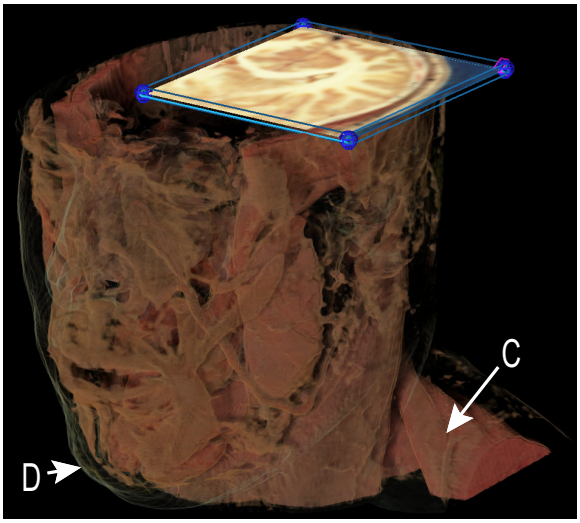
Figure 4: The behavior of primary data value (f), gradient magnitude (f'), and the second directional derivative (f'') as a function of position (a) and as a function of data value (b).



(a) Histograms of the Visible Male RGB dataset.



(b) The white (A) and gray (B) matter of the brain.



(c) The muscle and connective tissues (C) of the head and neck, showing skin (D) for reference.

Figure 5: The Visible Male RGB (color) data. The opacity is set using a 3D transfer function, color is taken directly from the data. The histogram (a) is visualized as projections on the primary planes of the RGB color space.

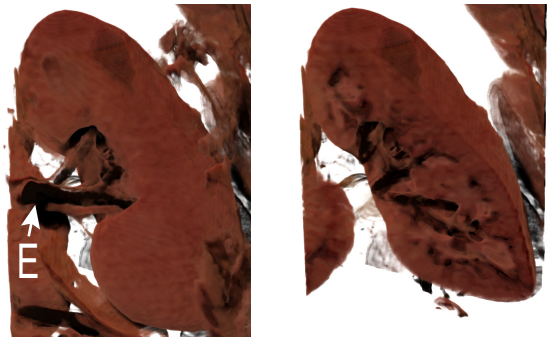


Figure 6: A kidney from the Visible Male RGB dataset. The renal vein is labeled (E). A clipping plane reveals internal structure (right).

Our choice of RGB for the transfer function axes is rather arbitrary; it is simply the most direct use of the color data. Other natural choices for color representation are the HSV or HLS spaces, or a CIE colorimetric space, if calibration data is available. Any color space is fine as long as it is possible to convert to RGB for display. It is important to note, however, that materials which are indistinguishable in the RGB color space will also be indistinguishable in any other color space. The choice of color space representation for the transfer function should be made on the basis of ease of use. Some color spaces, such as HSV, are better geared for human navigation. Our experience, however, has shown that tissue colors in cryosection are sometimes not what we expect. This can be seen in Figure 6, where the color in the renal vein (E) is essentially black, rather than red as we might expect blood to be. For this reason, our exploration of this dataset has been largely guided by probing and dual-domain interaction, which are described in the next section. We have also found it impractical to manipulate the transfer function in the full 3D space that it defines. Instead, we only manipulate the transfer function using two axes at a time. The placement of classified regions is very similar to that shown in Figure 5(a), where each classified region is represented as a projection onto two of the transfer function axes.

The kind of first derivative that we compute in multivariate data is based on previous work in color image segmentation [8, 36, 7]. While the gradient magnitude in scalar data represents the magnitude of local change at a point, an analogous first derivative measure in multivariate data captures the total amount of local change, across all the data components. This derivative has proven useful in color image segmentation because it allows a generalization of gradient-based edge detection. In our system, we use this first derivative measure as one axis in the multi-dimensional transfer function in order to isolate and visualize different regions of a multivariate volume according to the amount of local change, analogous to our use of gradient magnitude for scalar data.

If we represent the dataset as a multivariate function $\mathbf{f}(x, y, z) : \mathbb{R}^3 \rightarrow \mathbb{R}^m$, so that

$$\mathbf{f}(x, y, z) = (f_1(x, y, z), f_2(x, y, z), \cdots, f_m(x, y, z))$$

then the derivative $\mathbf{Df}$ is a matrix of first partial derivatives:

$$\mathbf{Df} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ & \vdots & \\ \frac{\partial f_m}{\partial x} & \frac{\partial f_m}{\partial y} & \frac{\partial f_m}{\partial z} \end{bmatrix}$$

By multiplying $\mathbf{Df}$ with its transpose, we can form a 3×3 tensor $\mathbf{G}$ which captures the directional dependence of total change:

$$\mathbf{G} = (\mathbf{Df})^{\mathrm{T}} \mathbf{Df} \tag{3}$$

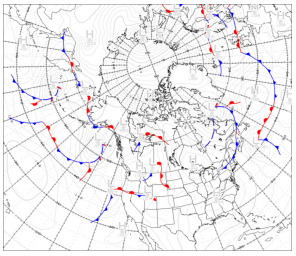
In the context of color edge detection [8, 36, 7], this matrix (specifically, its two-dimensional analog) is used as the basis of a quadratic function of direction $\mathbf{n}$, which Cumani [7] terms the *squared local contrast* in direction $\mathbf{n}$:

$$S(\mathbf{n}) = \mathbf{n}^{\mathrm{T}} \mathbf{G} \mathbf{n}$$

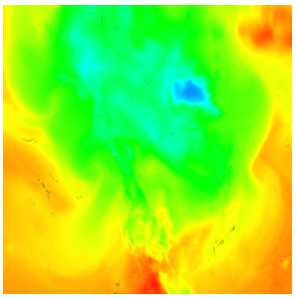
$S(\mathbf{n})$ can be analyzed by finding the principal eigenvector (and associated eigenvalue) of $\mathbf{G}$ to determine the direction $\mathbf{n}$ of greatest local contrast, or fastest change, and the magnitude of that change. Our experience, however, has been that in the context of multi-dimensional transfer functions, it is sufficient (and perhaps preferable) to simply take the L2 norm of $\mathbf{G}$, $\|\mathbf{G}\|$, which is the square root of the sum of the squares of the individual matrix components. As the L2 norm is invariant with respect to rotation, this is the same as the L2 norm of the three eigenvalues of $\mathbf{G}$, motivating our use of $\|\mathbf{G}\|$ as a directionally independent (and rotationally invariant) measure of local change. Other work on volume rendering of color data has used a non-rotationally invariant measure of $\mathbf{G}$ [9]. Since it is sometimes the case that the dynamic range of the individual channels (f_i) differ, we normalize the ranges of each channel's data value to be between zero and one. This allows each channel to have an equal contribution in the derivative calculation.

Meteorological simulations are a good example of datasets with features that can only be identified using a combination of data values [18], and which additionally benefit from using $\|\mathbf{G}\|$ as an axis in the multi-dimensional transfer function. Air masses, for instance, are a phenomenon described primarily by differences in both temperature and humidity. The interfaces of these air masses, or fronts, are responsible for the formation of mid-latitude storms. In particular, cold fronts can produce severe weather including showers, thunderstorms, hail, high winds, and tornados. Naturally, the precise identification of these fronts are of interest to meteorologists. Figure 7 contains the results from a numerical meteorological simulation that uses a forcing function from measured data. Figure 7(a) is a surface map of the simulation domain for reference. Figure 7(b) shows the results of an expert frontal analysis using a technique which overlays slices through different scalar values, (f_1, f_2, *etc.*), of the dataset, similar to those shown in Figures 7(c) and 7(d). This type of analysis is difficult because the expert must create a mental image of frontal behavior based on these scalar visualizations. The task is greatly simplified by visualizing the data based on its unique combination of data values, in this case temperature and humidity. The frontal regions were identified by probing in the spatial domain, seen as the dotted line in Figure 7(f) and visualizing the data values in the transfer function domain, seen in Figure 7(e). While the frontal region is identified as (A) in Figure 7(e), the visualization is clearer when we show the regions which correspond to the air masses that meet at these fronts, identified as (B) and (C) in Figures 7(e) and 7(f). The rendering on the left in Figure 7(f) shows the air masses; the image on the right uses a similar transfer function, but excludes regions with low $\|\mathbf{G}\|$ values. Notice that the interfaces, or frontal regions, of these air masses are emphasized.

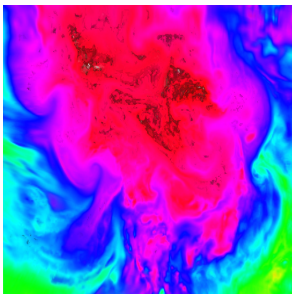
(a) Simulation map



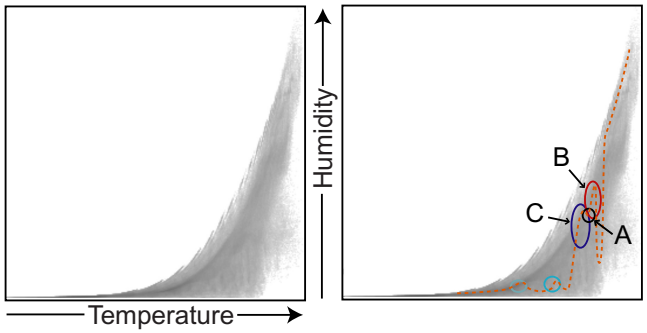
(b) Expert Frontal Analysis



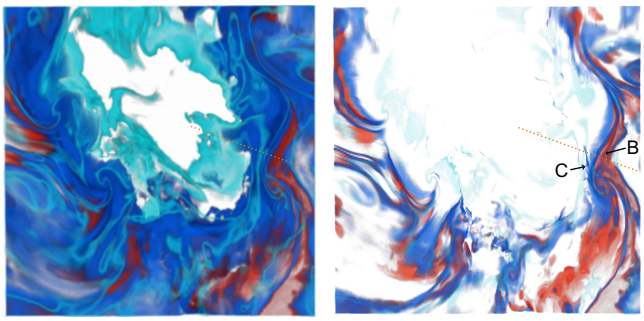
(c) Temperature



(d) Humidity



(e) 2D joint histogram of Temperature versus Humidity



(f) Warm (red) and cold (blue) air mass fronts

Figure 7: Frontal zones of a numerical weather simulation. (a) shows a reference map of the simulation domain. (b) shows the results of an expert analysis using scalar data visualizations similar to (c) and (d). (c) and (d) are slices through the dataset with a spectral color map. (e) shows a 2D log-scale joint histogram of temperature versus humidity. Region (A) shows the ranges of these data values that represent a mid-latitude front, (B) identifies the warm air mass, (C) identifies the cold air mass. (f) shows a volume rendering using a 3D transfer function which emphasizes regions (B) and (C). $\|\mathbf{G}\|$ from Equation 3 is used as the third axis of the transfer function for the rendering on the right to emphasize the portions of the air masses near the front. The dotted line shows a path through the dataset, the values along this line are shown in (e).

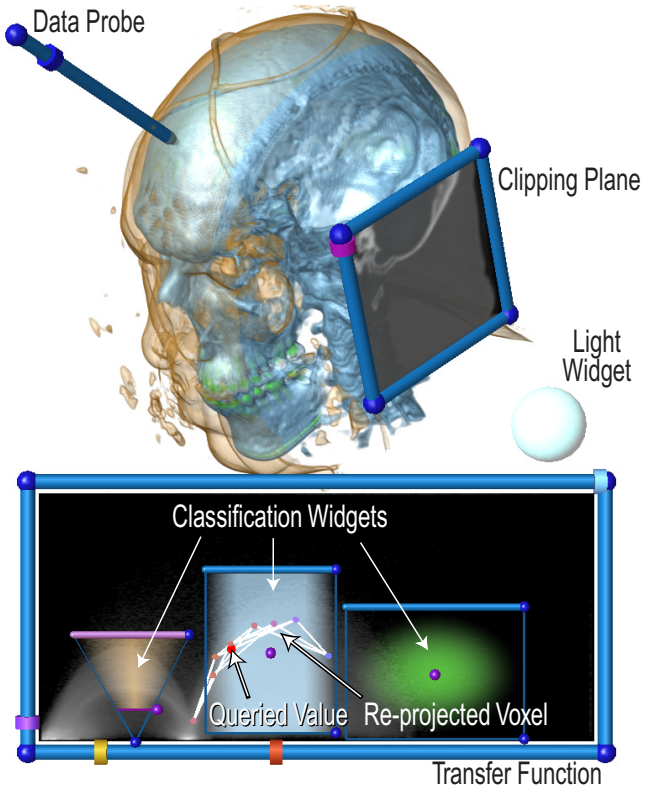


Figure 8: The direct manipulation widgets.

5 Interaction and Tools

While adding dimensions to the transfer function enhances our ability to isolate features of interest in a dataset, it tends to make the already unintuitive space of the transfer function even more difficult to navigate. This difficulty can be considered in terms of a conceptual gap between the spatial and transfer function domains. The spatial domain is the familiar 3D space for geometry and the volume data being rendered. The transfer function domain, however, is more abstract. Its dimensions are not spatial (*i.e.* the ranges of data values), and the quantities at each location are not scalar (*i.e.* opacity and three colors). It can be very difficult to determine the regions of the transfer function that correspond to features of interest, especially when a region is very small. Thus, to close this conceptual gap, we developed new interaction techniques, which permit interaction in both domains simultaneously, and a suite of direct manipulation widgets which provide the tools for such interactions. Figure 8 shows the various direct manipulation widgets as they appear in the system.

In a typical session with our system, the user creates a transfer function using a natural process of exploration, specification, and refinement. Initially, the user is presented with a volume rendering using a pre-determined transfer function that is likely to bring out some features of interest. This can originate with an automated transfer function generation tool [16], or it could be the default transfer function described later in Section 7. The user would then begin exploring the dataset.

Exploration is the process by which a user familiarizes him or herself with the dataset. A clipping plane can be moved through the volume to reveal internal structures. A slice of the original data can be mapped to the clipping plane, permitting a close inspection of the entire range of data values. Sample positions are probed in the spatial domain and their values, along with values in a neighborhood around that point, are visualized in the transfer function domain. This feedback allows the user to identify the regions of the transfer function that correspond to potential features of interest, made visible by the default transfer function or the sliced data. Once these regions have been identified, the user can then begin specifying a custom transfer function.

During the specification stage, the user creates a rough draft of the desired transfer function. While this can be accomplished by manually adding regions to the transfer function, a simpler method adds opacity to the regions in the transfer function at and around locations queried in the spatial domain. That is, the system can track, with a small region of opacity in the transfer function domain, the data values at the user-selected locations, while continually updating the volume rendering. This visualizes, in the spatial domain, all other voxels with similar transfer function values. If the user decides that an important feature is captured by the current transfer function, he or she can add that region into the transfer function and continue querying and investigating the volume.

Once these regions have been identified, the user can refine them by manipulating control points in the transfer function domain to better visualize features of interest. An important feature of our system is the ability to manipulate portions of the transfer function as discrete entities. This permits the modification of regions corresponding to a particular feature without affecting other classified regions.

Finally, this is an iterative process. A user continues the exploration, specification, and refinement steps until they are satisfied that all features of interest are made visible. In the remainder of this section we will introduce the interaction modalities used in the exploration and specification stages and briefly describe the individual direct manipulation widgets.

5.1 Probing and Dual-Domain Interaction

The concept of probing is simple: the user points at a location in the spatial domain and visualizes the values at that point in transfer function domain. We have found this feedback to be essential for making the connection between features seen in the spatial domain and the ranges of values that identify them in the transfer function domain. Creating the best transfer function for visualizing a feature of interest is only possible with an understanding of the behavior of data values at and around that feature. This is especially true for multi-dimensional transfer functions where a feature is described by a complex combination of data values. The value of this dataset-specific guidance can be further enhanced by automatically setting the transfer function based on these queried values.

In a traditional volume rendering system, setting the transfer function involves moving the control points (in a sequence of linear ramps defining color and opacity), and then observing the resulting rendered image. That is, interaction in the transfer function domain is guided by careful observation of changes in the spatial domain. We prefer a reversal of this process, in which the transfer function is set by direct interaction in the *spatial domain*, with observation of the transfer function domain. Furthermore, by allowing interaction to happen in both domains simultaneously, the conceptual gap between them is significantly lessened, effectively simplifying the complicated task of specifying a multi-dimensional transfer function to pointing at a feature of interest. We use the term “dual-domain interaction” to describe this approach to transfer function exploration and generation.

The top of Figure 9 illustrates the specific steps of dual-domain interaction. When a position inside the volume is queried by the user with the data probe widget (a), the values associated with that position (multivariate values, or the data value, first and sec-

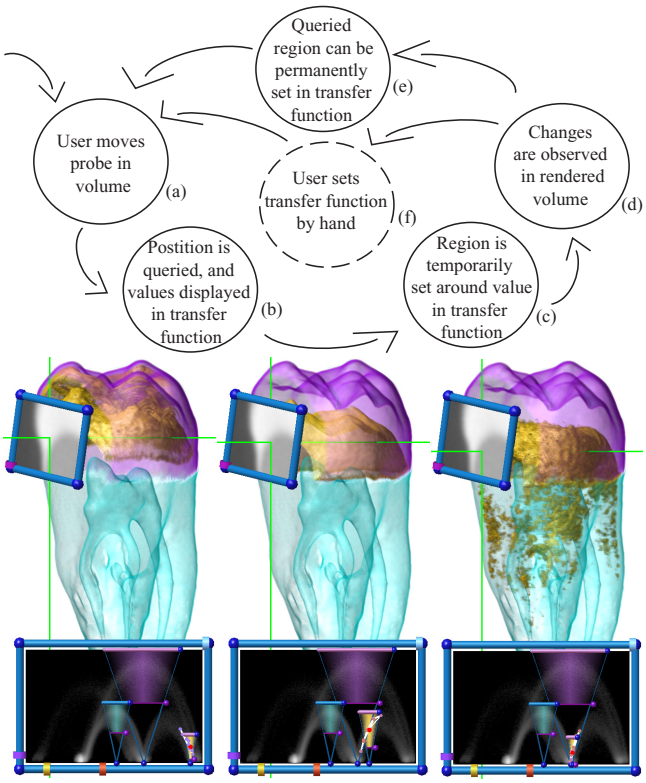


Figure 9: Dual-Domain Interaction

ond derivative) are graphically represented in the transfer function widget (b). Then, a small region of high opacity (c) is temporarily added to the transfer function at the data values determined by the probe location. The user has now set a multi-dimensional transfer function simply by positioning a data probe within the volume. The resulting rendering (d) depicts (in the spatial domain) all the other locations in the volume which share values (in the transfer function domain) with those at the data probe tip. If the features rendered are of interest, the user can copy the temporary transfer function to the permanent one (e), by, for instance, tapping the keyboard space bar with the free hand. As features of interest are discovered, they can be added to the transfer function quickly and easily with this type of two-handed interaction. Alternately, the probe feedback can be used to manually set other types of classification widgets (f), which are described later. The outcome of dual-domain interaction is an effective multi-dimensional transfer function built up over the course of data exploration. The widget components which participated in this process can be seen in the bottom of Figure 9, which shows how dual-domain interaction can help volume render the CT tooth dataset. The remainder of this section describes the individual widgets and provides additional details about dual-domain interaction.

5.2 Data Probe Widget

The data probe widget is responsible for reporting its tip's position in volume space and its slider sub-widget's value. Its pencil-like shape is designed to give the user the ability to point at a feature in the volume being rendered. The other end of the widget orients the widget about its tip. When the volume rendering's position or orientation is modified, the data probe widget's tip tracks its point in volume space. A natural extension is to link the data probe widget to a haptic device, such as the SensAble PHANTOM, which can provide a direct 3D location and orientation [29].

5.3 Clipping Plane Widget

The clipping plane is responsible for reporting its orientation and position to the volume renderer, which handles the actual clipping when it draws the volume. In addition to clipping, the volume widget will also map a slice of the data to the arbitrary plane defined by the clip widget, and blend it with the volume by a constant opacity value determined by the clip widget's slider. It is also responsible for reporting the spatial position of a mouse click on its clipping surface. This provides an additional means of querying positions within the volume, distinct from the 3D data probe. The balls at the corners of the clipping plane widget are used to modify its orientation, and the bars on the edges are used to modify its position.

5.4 Transfer Function Widget

The main role of the transfer function widget is to present a graphical representation of the transfer function domain, in which feedback from querying the volume (with the data probe or clipping plane) is displayed, and in which the transfer function itself can be set and altered. The balls at the corners of the transfer function widget are used to resize it, as with a desktop window, and the bars on the edges are used to translate its position. The inner plane of the frame is a polygon texture-mapped with the lookup table containing the transfer function. A joint histogram of data, seen with the images in Section 4, can also be blended with the transfer function to provide valuable information about the behavior and relationship of data values in the transfer function domain.

The data values at the position queried in the volume (either via the data probe or clipping plane widgets) are represented with a small ball in the transfer function widget. In addition to the precise location queried, the eight data sample points at the corners of the voxel containing the query location are also represented by balls in the transfer function domain, and are connected together with edges that reflect the connectivity of the voxel corners in the spatial domain. By "re-projecting" a voxel from the spatial domain to a simple graphical representation in the transfer function domain, the user can learn how the transfer function variables (data values at each sample point) are changing near the probe location. The values for the third, or unseen axis, are indicated by coloring the balls. For instance, with scalar data, second derivative values which are negative, zero, or positive are represented by blue, white, and yellow balls, respectively. When the projected points form an arc, with the color varying through these assigned colors, the probe is at a boundary in the volume as seen in Figure 8. When the re-projected data points are clustered together, the probe is in a homogeneous region. As the user gains experience with this representation, he or she can learn to "read" the re-projected voxel as an indicator of the volume characteristics at the probe location.

5.5 Classification Widgets

In addition to the process of dual-domain interaction described above, transfer functions can also be created in a more manual fashion by adding one or more classification widgets to the main transfer function window. Classification widgets are designed to identify regions of the transfer function as discrete entities. Each widget type has control points which modify its position or size. Optical properties, such as opacity and color are modified by selecting the widgets inner surface. The opacity and color contributions from each classification widget are blended together to form the transfer function. We have developed two types of classification widget: triangular and rectangular.

The triangular classification widget, shown in Figures 2, 8, 9, and 11, is based on Levoy's "isovalue contour surface" opacity function [25]. The widget is an inverted triangle with a base point attached to the horizontal data value axis. The triangle's size and position are adjusted with control points. There are an upper and lower threshold for the gradient magnitude, as well as a shear. Color is constant across the widget; opacity is maximal along the center of the widget, and it linearly ramps down to zero at the left and right edges.

The triangular classification widgets are particularly effective for visualizing *surfaces* in scalar data. More general transfer functions, for visualizing data which may not have clear boundaries, can be created with the rectangular classification widget. The rectangular region spanned by the widget defines the data values which receive opacity and color. Like the triangular widget, color is constant, but the opacity is more flexible. It can be constant, or fall off in various ways: quadratically as an ellipsoid with axes corresponding to the rectangle's aspect ratio, or linearly as a ramp, tent, or pyramid.

As noted in the description of the transfer function widget, even when a transfer function has more than two dimensions, only two dimensions are shown at any one time. For 3D transfer functions, classification widgets are shown as their projections onto the visible axes. In this case, a rectangular classification widget becomes a box in the 3D domain of the transfer function. Its appearance to the user, however, as 2D projections, is identical to the rectangular widget. When the third axis of the transfer function plays a more simplified role, interactions along this axis are tied to sliders seen along the top bar of the transfer function. For instance, since our research on scalar data has focused on visualizing boundaries between material regions, we have consistently used the second derivative to emphasize the regions where the second derivative magnitude is small or zero. Specifically, maximal opacity is always given to zero second derivative, and decreases linearly towards the second derivative extremal values. How much the opacity changes as a function of second derivative magnitude is controlled with a single slider, which we call the "boundary emphasis slider." With the slider in its left-most position, zero opacity is given to extremal second derivatives; in the right-most position, opacity is constant with respect to the second derivative. We have employed similar techniques for manipulating other types of third axis values using multiple sliders.

While the classification widgets are usually set by hand in the transfer function domain, based on feedback from probing and re-projected voxels, their placement can also be somewhat automated. This further reduces the difficulty of creating an effective higher dimensional transfer function. The classification widget's location and size in the transfer function domain can be tied to the distribution of the re-projected voxels determined by the data probe location. For instance, the rectangular classification widget can be centered at the transfer function values interpolated at the data probe's tip, with the size of the rectangle controlled by the data probe's slider. The triangular classification widget can be located horizontally at the data value queried by the probe, with the width and height determined by the horizontal and vertical variance in the re-projected voxel locations. This technique produced the changes in the transfer function for the sequence of renderings in Figure 9.

5.6 Shading Widget

The shading widget is a collection of spheres which can be rendered in the scene to indicate and control the light direction and color. Fixing a few lights in view space is generally effective for renderings, therefore changing the lighting is an infrequent operation.

5.7 Color Picker Widget

The color picker is an embedded widget which is based on the hue-lightness-saturation (HLS) color space. Interacting with this widget can be thought of as manipulating a sphere with hues mapped around the equator, gradually becoming black at the top, and white at the bottom. To select a hue, the user moves the mouse horizontally, rotating the ball around its vertical axis. Vertical mouse motion tips the sphere toward or away from the user, shifting the color towards white or black. Saturation and opacity are selected independently using different mouse buttons with vertical motion. While this color picker can be thought of as manipulating an HLS sphere, no geometry for this is rendered. Rather, the triangular and rectangular classification widgets embed the color picker in the polygonal region which contributes opacity and color to the transfer function domain. The user specifies a color simply by clicking on that object, then moving the mouse horizontally and vertically until the desired hue and lightness are visible. In most cases, the desired color can be selected with a single mouse click and gesture.

6 Rendering and Hardware

While this paper is conceptually focused on the matter of setting and applying higher dimensional transfer functions, the quality of interaction and exploration described would not be possible without the use of modern graphics hardware. Our implementation relies heavily on an OpenGL extension known as *dependent texture reads*. This extension can be used for both classification and shading. In this section, we describe our modifications to the classification portion of the traditional 3D texture-based volume rendering pipeline. We also describe methods for adding interactive volumetric shading and shadows to the pipeline.

Our system supports volumes which are stored as 3D textures with one, two, or four values per texel. This is is due to memory alignment restrictions of graphics hardware. Volumes with three values per sample utilize a four value texture, where the fourth value is simply ignored. Volumes with more than four values per sample could be constructed using multiple textures.

6.1 Dependent Texture Reads

Dependent texture reads are a hardware extension that is a similar but more efficient implementation of a previous extension known as *pixel texture* [10, 13, 28, 40]. Dependent texture reads and pixel texture are names for operations which use color fragments to generate texture coordinates, and replace those color fragments with the corresponding entries from a texture. This operation essentially amounts to an arbitrary function evaluation with up to three variables via a lookup table. If we were to perform this operation on an RGB fragment, each channel value would be scaled between zero and one, and these new values would then be used as texture coordinates of a 3D texture. The color values produced by the 3D texture lookup replace the original RGB values. Nearest neighbor or linear interpolation can be used to generate the replacement values. The ability to scale and interpolate color channel values is a convenient feature of the hardware. It allows the number of elements along a dimension of the texture containing the new color values to differ from the dynamic range of the component that generated the texture coordinate. Without this flexibility, the size of a 3D dependent texture would be prohibitively large.

6.2 Classification

Dependent texture reads are used for the transfer function evaluation. Data values stored in the color components of a 3D texture are interpolated across some proxy geometry, a plane for instance.

These values are then converted to texture coordinates and used to acquire the color and alpha values in the transfer function texture per-pixel in screen space. For eight bit data, an ideal transfer function texture would have 256 color and alpha values along each axis. For 3D transfer functions, however, the transfer function texture would then be $256^3 \times 4$ bytes. Besides the enormous memory requirements of such a texture, the size also affects how fast the classification widgets can be rasterized, thus affecting the interactivity of transfer function updates. We therefore limit the number of elements along an axis of a 3D transfer function based on its importance. For instance, with scalar data, the primary data value is the most important, the gradient magnitude is secondary, and the second derivative serves an even more tertiary role. For this type of multi-dimensional transfer function, we commonly use a 3D transfer function texture with dimensions $256 \times 128 \times 8$ for data value, gradient magnitude, and second derivative respectively. 3D transfer functions can also be composed separably as a 2D and 1D transfer function. This means that the total size of the transfer function is $256^2 + 256$. The tradeoff, however, is in expressivity. We can no longer specify a transfer function based on the unique combination of all three data values. Separable transfer functions are still quite powerful. Applying the second derivative as a separable 1D portion of the transfer functions is quite effective for visualizing boundaries between materials. With the separable 3D transfer function for scalar volumes, there is only one boundary emphasis slider which affects all classification widgets as opposed to the general case where each classification widget has its own boundary emphasis slider. We have employed a similar approach for multi-variate data visualization. The meteorological example used a separable 3D transfer function. Temperature and humidity were classified using a 2D transfer function and the multi-derivative of these values was classified using a 1D transfer function. Since our specific goal was to show only regions with high values of $\|\mathbf{G}\|$, we only needed two sliders to specify the beginning and ending points of a linear ramp along this axis of the transfer function.

6.3 Surface Shading

Shading is a fundamental component of volume rendering because it is a natural and efficient way to express information about the shape of structures in the volume. However, much previous work with texture-memory based volume rendering lacks shading. Many modern graphics hardware platforms support multi-texture and a number of user defined operations for blending these textures per-pixel. These operations, which we will refer to as fragment shading, can be leveraged to compute a surface shading model.

The technique originally proposed by Rezk-Salama *et al.* [6] is an efficient way to compute the Blinn-Phong shading model on a per-pixel basis for volumes. This approach, however, can suffer from artifacts caused by denormalization during interpolation. While future generations of graphics hardware should support the square root operation needed to renormalize on a per-pixel basis, we can utilize *cube map dependent texture reads* to evaluate the shading model. This type of dependent texture read allows an RGB color component to be treated as a vector and used as the texture coordinates for a cube map. Conceptually, a cube map can be thought of as a collection of six textures that make up the faces of a cube centered about the origin. Texels are accessed with a 3D texture coordinate (s,t,r) representing a direction vector. The accessed texel is the point corresponding to the intersection of a line through the origin in the direction of (s,t,r) and a cube face. The color values at this position represent incoming diffuse radiance if the vector (s,t,r) is a surface normal or specular radiance if (s,t,r) is a reflection vector. The advantages of using a cube map dependent texture read is that the vector (s,t,r) does not need to be normalized, and the cube map can encode an arbitrary number of lights or a full environment map. This approach, however, comes at the cost of reduced performance. A per-pixel cube map evaluation can be as much as three times slower than evaluating the dot products for a limited number of light sources in the fragment shader stage.

Surface based shading methods are well suited for visualizing the boundaries between materials. However, since the surface normal is approximated by the normalized gradient of a scalar field, these methods are not robust for shading homogeneous regions, where the gradient magnitude is very low or zero and its measurement is sensitive to noise. Gradient based surface shading is also unsuitable for shading volume renderings of multivariate fields. While we can assign the direction of greatest change for a point in a multivariate field to the eigenvector (e_1) corresponding to the largest eigenvalue (λ_1) of the tensor $\mathbf{G}$ from Equation 3, e_1 is only a valid representation of orientation, not the absolute direction. This means that the sign of e_1 can flip in neighboring regions even though their orientations may not differ. Therefore, the vector e_1 does not interpolate, making it a poor choice of surface normal. Furthermore, this orientation may not even correspond to the surface normal of a classified region in a multivariate field.

6.4 Shadows

Shadows provide important visual queues relating to the depth and placement of objects in a scene. Since the computation of shadows does not depend on a surface normal, they provide a robust method for shading homogeneous regions and multivariate volumes. Adding shadows to the volume lighting model means that light gets attenuated through the volume before being reflected back to the eye.

Our approach differs from previous work [2] in two ways. First, rather than creating a volumetric shadow map, we utilize an off screen render buffer to accumulate the amount of light attenuated from the light's point of view. Second, we modify the slice axis to be the direction halfway between the view and light directions. This allows the same slice to be rendered from both the eye and light points of view. Consider the situation for computing shadows when the view and light directions are the same, as seen in Figure 10(a). Since the slices for both the eye and light have a one to one correspondence, it is not necessary to precompute a volumetric shadow map. The amount of light arriving at a particular slice is equal to one minus the accumulated opacity of the slices rendered before it. Naturally if the projection matrices for the eye and light differ, we need to maintain a separate buffer for the attenuation from the light's point of view. When the eye and light directions differ, the volume would be sliced along each direction independently. The worst case scenario happens when the view and light directions are perpendicular, as seen in Figure 10(b). In the case, it would seem necessary to save a full volumetric shadow map which can be resliced with the data volume from the eye's point of view providing shadows. This approach, however, suffers from an artifact referred to as attenuation leakage. The visual consequences of this are blurry shadows and surfaces which appear much darker than they should due to the image space high frequencies introduced by the transfer function. The attenuation at a given sample point is blurred when light intensity is stored at a coarse resolution and interpolated during the observer rendering phase.

Rather than slice along the vector defined by the view direction or the light direction, we modify the slice axis to allow the same slice to be rendered from both points of view. When the dot product of the light and view directions is positive, we slice along the vector halfway between the light and view directions, seen in Figure 10(c). In this case, the volume is rendered in front to back order with respect to the observer. When the dot product is negative, we slice along the vector halfway between the light and the inverted view directions, seen in Figure 10(d). In this case, the volume is

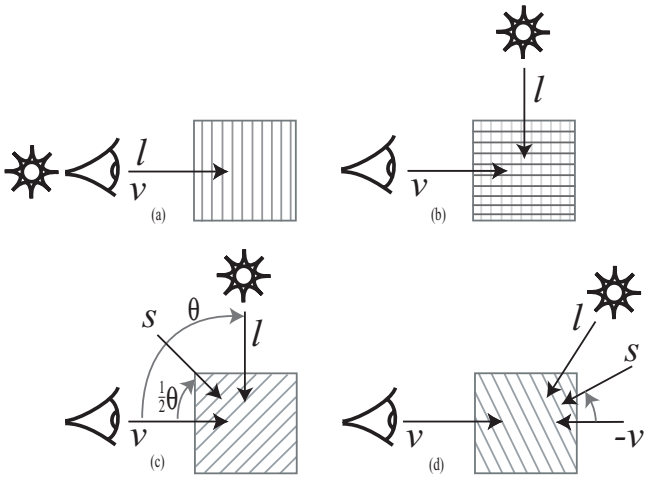


Figure 10: Modified slice axis for light transport.

rendered in back to front order with respect to the observer. In both cases the volume is rendered in front to back order with respect to the light. Care must be taken to insure that the slice spacing along the view and light directions are maintained when the light or eye positions change. If the desired slice spacing along the view direction is d_v and the angle between v and l is θ then the slice spacing along the slice direction is

$$d_s = \cos(\frac{\theta}{2})d_v. \quad (4)$$

This is a multi-pass approach. Each slice is first rendered from the observers point of view using the results of the previous pass from the light's point of view, which modulates the brightness of samples in the current slice. The same slice is then rendered from light's point of view to calculate the intensity of the light arriving at the next layer.

Since we must keep track of the amount of light attenuated at each slice, we utilize an off screen render buffer, known as a *pixel buffer*. This buffer is initialized to $1 - light\,intensity$. It can also be initialized using an arbitrary image to create effects such as spotlights. The projection matrix for the light's point of view need not be orthographic; a perspective projection matrix can be used for point light sources. However, the entire volume must fit in the light's view frustum. Light is attenuated by simply accumulating the opacity for each sample using the over operator. The results are then copied to a texture which is multiplied with the next slice from the eye's point of view before it is blended into the frame buffer. While this copy to texture operation has been highly optimized on the current generation of graphics hardware, we have achieved a dramatic increase in performance using a hardware extension known as *render to texture*. This extension allows us to directly bind a pixel buffer as a texture, avoiding the unnecessary copy operation.

This approach has a number of advantages over previous volume shadow methods. First, attenuation leakage is no longer a concern because the computation of the light transport (slicing density) is decoupled from the resolution of the data volume. Computing light attenuation in image space allows us to match the sampling frequency of the light transport with that of the final volume rendering. Second, this approach makes far more efficient use of memory resources than those which require a volumetric shadow map. Only a single additional 2D buffer is required as opposed to a potentially large 3D volume. One disadvantage of this approach is that due to the image space sampling, artifacts may appear at shadow boundaries when the opacity makes a sharp jump from low to high. This can be overcome by using a higher resolution for the light buffer than for the frame buffer. We have found that 30 to 50 percent additional resolution is adequate.

As noted at the end of the previous section, surface based shading models are inappropriate for homogeneous regions in a volume. However, it is often useful to have both surface shaded and shadowed renderings regardless of whether or not homogeneous regions are being visualized. To insure that homogeneous regions are not surface shaded, we simply interpolate between surface shaded and unshaded using the gradient magnitude. Naturally, regardless of whether or not a particular sample is surface shaded, it is still modulated by the light attenuation providing shadows. In practice we have found that interpolating based on $1 - (1 - \|\nabla f\|)^2$ produces better results, since mid-range gradient magnitudes can still be interpreted as surface features. Figure 11 shows a rendering which combines surface shading and shadows in such a way. Figure 1 shows a volume rendering using shadows with the light buffer initialized to simulate a spotlight. Figures 2 and 3 show volume renderings using only surface based shading. Figures 5, 6, and 7 only use shadows for illumination.

7 Discussion

Using multi-dimensional transfer functions heightens the importance of densely sampling the voxel data in rendering. With each new axis in the transfer function, there is another dimension along which neighboring voxels can differ. It becomes increasingly likely that the data sample points at the corners of a voxel straddle an important region of the transfer function (such as a region of high opacity) instead of falling within it. Thus, in order for the boundaries to be rendered smoothly, the distance between view-aligned sampling planes through the volume must be very small. Most of the figures in this paper were generated with sampling rates of about 3 to 6 samples per voxel. At this sample rate, frame updates can take nearly a second for a moderately sized ($256 \times 256 \times 128$) shaded and shadowed volume. For this reason, we lower the sample rate during interaction, and re-render at the higher sample rate once an action is completed. During interaction, the volume rendered surface will appear coarser, but the surface size and location are usually readily apparent. Thus, even with lower volume sampling rates during interaction, the rendered images are effective feedback for guiding the user in transfer function exploration.

While the triangular classification widget is based on Levoy's iso-contour classification function, we have found it necessary to have additional degrees of freedom, such as a shear. Shearing the triangle classification along the data value axis, so that higher values are emphasized at higher gradients, allows us to follow the center of some boundaries more accurately. This is a subtle but basic characteristic of boundaries between a material with a narrow distribution of data values, and another material with a wide value distribution. This pattern can be observed in in the boundary between soft tissue (with a narrow value distribution) and bone (wide value distribution) of the Visible Male CT, seen in Figure 12. Thresholding the minimum gradient magnitude allows better feature discrimination as can be seen in Figure 2(b).

A benefit of using multi-dimensional transfer functions is the ability to use a "default" transfer function which is produced without any user interaction. Given our interest in visualizing the boundaries between materials, this was achieved by assigning opacity to high gradient magnitudes and, in the case of scalar data, low-magnitude second derivatives, regardless of data value, while varying hue along the data value. This default transfer function is intended only as a starting point for further modification with the widgets, but often it succeeds in depicting the main structures of

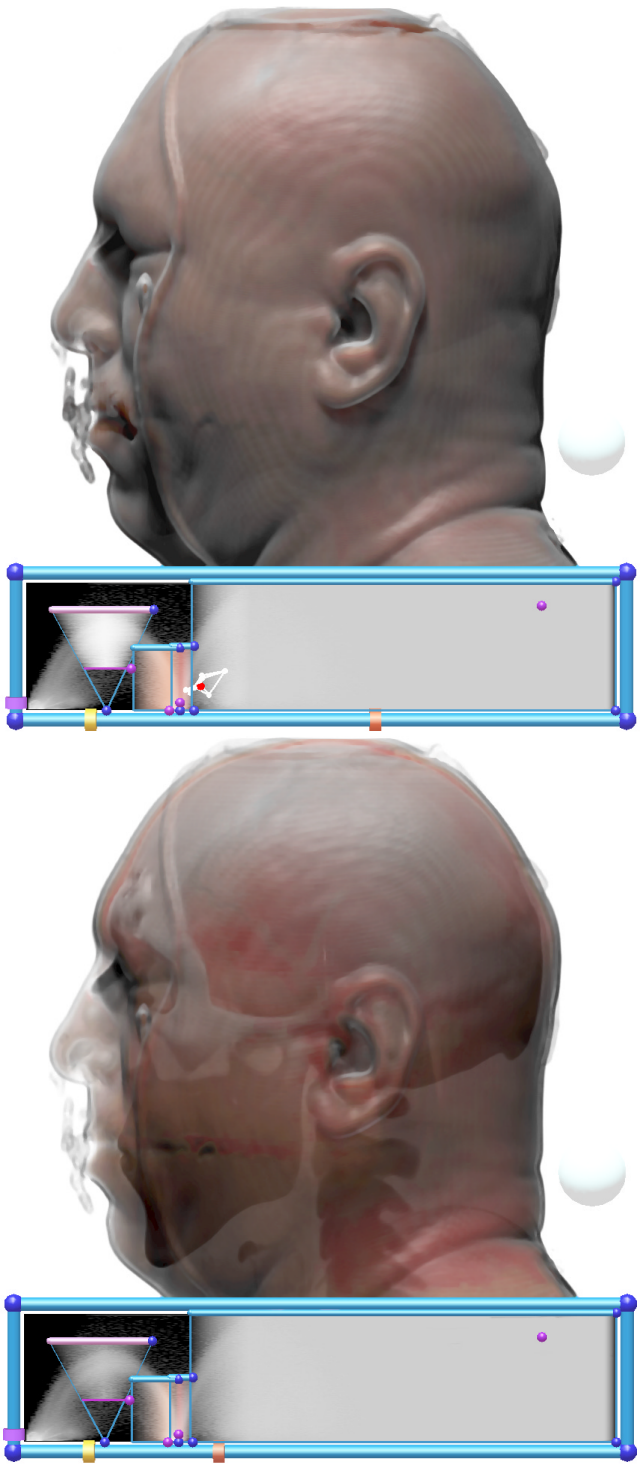


Figure 11: Volume renderings of the Visible Male CT (frozen) demonstrating combined surface shading and shadows.

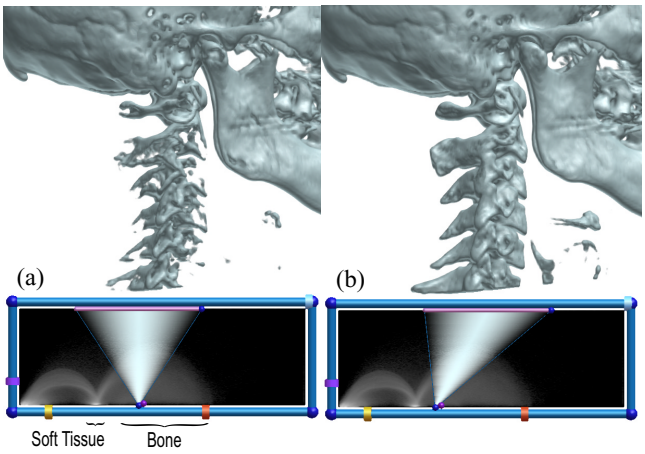


Figure 12: The soft tissue/bone boundary of the Visible Male CT. It is necessary to shear the triangular classification widget to follow the center of this boundary.

the volume, as seen in Figure 13. Other application areas for volume rendering may need different variables for multi-dimensional transfer functions, with their own properties governing the choices for default settings.

While multi-dimensional transfer functions are quite effective for visualizing material boundaries, we have also found them to be useful for visualizing the materials themselves. For instance, if we attempt to visualize the dentin of the Human Tooth CT using a 1D transfer function, we erroneously color the background/enamel boundary, seen in Figure 14(a). The reason for this can be seen in Figure 3(a), where the range of data values which define the background/enamel boundary overlap with the dentin's data values. We can easily correct this erroneous coloring with a 2D transfer function that only gives opacity to lower gradient magnitudes. This can be seen in Figure 14(b).

A further benefit of dual-domain interaction is the ability to create feature-specific multi-dimensional transfer functions which would be extremely difficult to produce by manual placement of classification widgets. If a feature can be visualized in isolation with only a very small and accurately placed classification widget, the best way to place the widget is via dual-domain interaction. This is the case for visualizing different soft tissues in CT data, such as the white matter of the brain in the Visible Male CT, shown in Figure 15.

Dual-domain interaction has utility beyond setting multi-dimensional transfer functions. Dual-domain interaction also helps answer other questions about the limits of direct volume rendering for displaying specific features in the data. For example, the feedback in the transfer function domain can show the user whether a certain feature of interest detected during spatial domain interaction is well-localized in the transfer function domain. If re-projected voxels from different positions, in the same feature, map to widely divergent locations in the transfer function domain, then the feature is not well-localized, and it may be hard to create a transfer function which clearly visualizes it. Similarly, if probing inside two distinct features indicates that the re-projected voxels from both features map to the same location in the transfer function domain, then it may be difficult to selectively visualize one or the other feature.

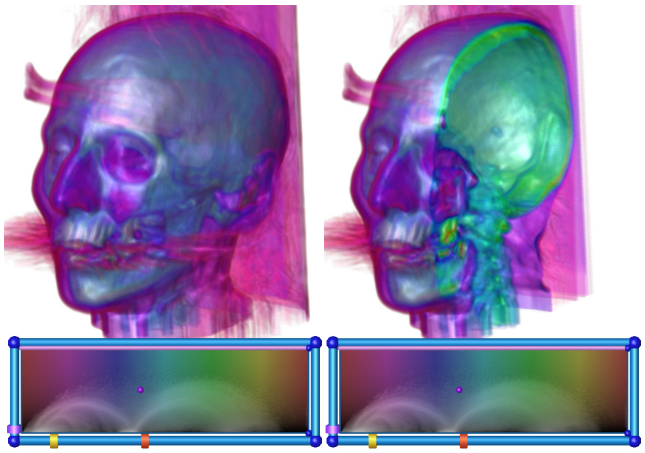


Figure 13: A default transfer function for scalar data applied to the Chapel Hill CT. Hue varies along the data value axis and opacity varies along the gradient magnitude axis. A clipping plane reveals internal structure (right).

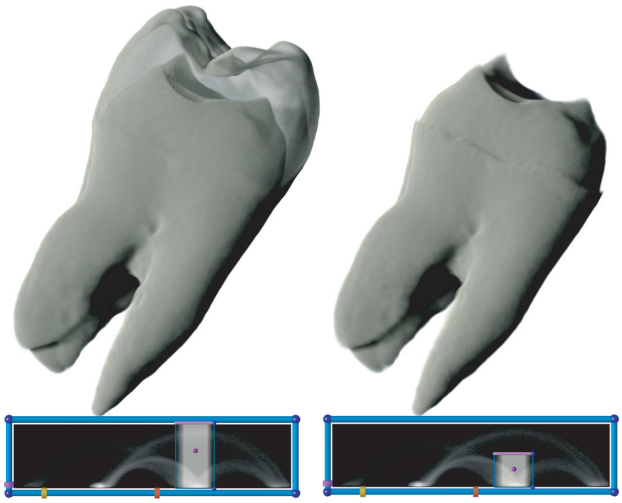


Figure 14: The dentin of the Human Tooth CT. (a) shows that a 1D transfer function, simulated by assigning opacity to data values regardless of gradient magnitude, will erroneously color the background/enamel boundary. A 2D transfer function, shown in (b) can avoid assigning opacity to the range of gradient magnitudes that define this boundary.

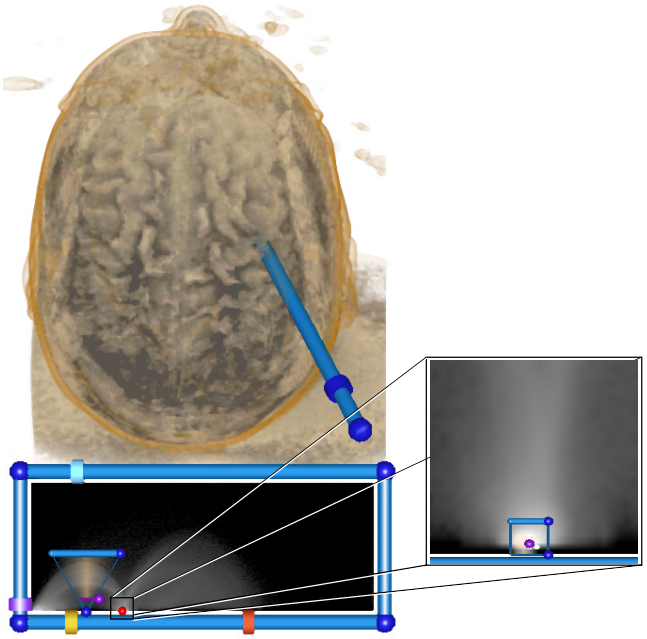


Figure 15: The brain of the Visible Male CT. The transfer functions were created using dual-domain interaction. A detail region shows how small the region that identifies this subtle feature is in the transfer function domain.

8 Future Work

One unavoidable drawback to using multi-dimensional transfer functions is the increased memory consumption needed to store all the transfer function variables at each voxel sample point. Future work can expand the dataset size by using parallel hardware rendering methods [20]. Surface based shading also has a dramatic impact on the data set size since we must store pre-computed normals, this requires an additional 4 bytes per-sample.

Using shadows for illumination is advantageous since it does not require a surface normal, thus eliminating the need for a normal volume. New shading models based on this approach might have the potential to create even more realistic and informative imagery. As future generations of graphics hardware provide even richer feature sets, it should soon be possible to create and implement better approximations of light transport through volumetric media at interactive frame rates.

Another area of future research would be to explore methods of surface normal generation using on-the-fly post-classification gradient estimation. This is a non-trivial problem since the transfer function can introduce very high frequencies or discontinuities, which can be problematic for creating normals that produce smooth shading. Such a method would have the potential to provide robust normals for surface shading multivariate volume visualizations.

Direct manipulation widgets and spatial interaction techniques lend themselves well to immersive environments. We would like to experiment with dual-domain interaction in a stereo, tracked, environment. We speculate that an immersive environment could make interacting with a 3D transfer function more natural and intuitive. We would also like to perform usability studies on our direct manipulation widgets and dual-domain interaction technique, as well as perceptual studies on 2D and 3D transfer functions for volume rendering.

9 Summary

This paper demonstrates the importance of multi-dimensional transfer functions for direct volume rendering applications. We present several examples for both scalar and more general multivariate datasets. We also identify the importance of interactive techniques. We introduce new interaction modalities and tools to make the process of specifying a high quality transfer function efficient and effective. These tools guide the user based on dataset specific information. We also present a number of methods for volume shading, including a novel technique for generating volumetric shadows.

10 Acknowledgments

The authors would like to thank Al McPherson from the ACL at LANL for fruitful and provocative conversations about volumetric shadows. This research was funded by grants from the Department of Energy (VIEWS 0F00584), the National Science Foundation (ASC 8920219, MRI 9977218, ACR 9978099), and the National Institutes of Health National Center for Research Resources (1P41RR12553-2). We would also like to thank NVIDIA, ATI and sgi for their making their latest generations of hardware available.

References

[1] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The Contour Spectrum. In *Proceedings IEEE Visualization 1997*, pages 167–173, 1997.

[2] Uwe Behrens and Ralf Ratering. Adding Shadows to a Texture-Based Volume Renderer. In *1998 Volume Visualization Symposium*, pages 39–46, 1998.

[3] Lawrence D. Bergman, Bernice E. Rogowitz, and Lloyd A. Treinish. A Rule-based Tool for Assisting Colormap Selection. In *Proceedings Visualization 1995*, pages 118–125. IEEE, October 1995.

[4] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *ACM Symposium On Volume Visualization*, 1994.

[5] D. Brookshire Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-Dimensional Widgets. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pages 183–188, 1992.

[6] C.Rezk-Salama, K.Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *Siggraph/Eurographics Workshop on Graphics Hardware 2000*, 2000.

[7] A. Cumani, P. Grattoni, and A. Guiducci. An edge-based description of color images. *GMIP*, 53(4):313–323, 1991.

[8] Silvano Di Zenzo. A Note on the Gradient of a Multi-Image. *Computer Vision, Graphics, and Image Processing*, 33(1):116–125, Jan 1986.

[9] David Ebert, Christopher Morris, Penny Rheingans, and Terry Yoo. Designing Effective Transfer Functions for Volume Rendering from Photographic Volumes. *IEEE TVCG*, (to appear) 2002.

[10] Klaus Engel, Martin Kraus, and Thomas Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Siggraph/Eurographics Workshop on Graphics Hardware 2001*, 2001.

[11] Allen Van Gelder and Kwansik Kim. Direct Volume Rendering with Shading via Three-Dimensional Textures. In *ACM Symposium On Volume Visualization*, pages 23–30, 1996.

[12] Taosong He, Lichan Hong, Arie Kaufman, and Hanspeter Pfister. Generation of Transfer Functions with Stochastic Search Techniques. In *Proceedings IEEE Visualization 1996*, pages 227–234, 1996.

[13] Wolfgang Heidrich, Rudiger Westermann, Hans-Peter Seidel, and Thomas Ertl. Applications of Pixel Textures in Visualization and Realistic Image Synthesis. In *Proceedings of the 1999 Symposium on Interacive 3D Graphics*, 1999.

[14] Kenneth P. Hernandon and Tom Meyer. 3D Widgets for Exploratory Scientific Visualization. In *Proceedings of UIST '94 (SIGGRAPH)*, pages 69–70. ACM, November 1994.

[15] Jiří Hladůvka, Andreas König, and Eduard Gröller. Curvature-Based Transfer Functions for Direct Volume Rendering. In Bianca Falcidieno, editor, *Spring Conference on Computer Graphics 2000*, volume 16, pages 58–65, May 2000.

[16] Gordon Kindlmann. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. Master's thesis, Cornell University, Ithaca, NY, January 1999 (http://www.cs.utah.edu/∼gk/MS).

[17] Gordon Kindlmann and James W. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In *IEEE Symposium On Volume Visualization*, pages 79–86, 1998.

[18] Joe Kniss, Charles Hansen, Michel Grenier, and Tom Robinson. Volume Rendering Multivariate Data to Visualize Meteorological Simulations: A Case Study. In *VisSym 2002, Joint EUROGRAPHICS - IEEE TCCG Symposium on Visualization*, (to appear) 2002.

[19] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets. In *IEEE Visualization 2001*, pages 255–262, 2001.

[20] Joe Kniss, Patrick S. McCormick, Allen McPherson, James Ahrens, Jamie Painter, Alan Keahey, and Charles Hansen. Interactive Texture-Based Volume Rendering for Large Data Sets. *IEEE Computer Graphics and Applications*, 21(4):52–61, July/August 2001.

[21] Andreas König and Eduard Gröller. Mastering Transfer Function Specification by Using VolumePro Technology. In Tosiyasu L. Kunii, editor, *Spring Conference on Computer Graphics 2001*, volume 17, pages 279–286, April 2001.

[22] Philip Lacroute and Marc Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform. In *ACM Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 451–458, July 1994.

[23] David H. Laidlaw. *Geometric Model Extraction from Magnetic Resonance Volume Data*. PhD thesis, California Institute of Technology, May 1995.

[24] Eric LaMar, Bernd Hamann, and Kenneth I. Joy. Multiresolution Techniques for Interactive Texture-Based Volume Visualization. In *Proceedings Visualization '99*, pages 355–361. IEEE, October 1999.

[25] Marc Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics & Applications*, 8(5):29–37, 1988.

[26] J. Marks, B. Andalman, P.A. Beardsley, and H. Pfister et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *ACM Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 389–400, August 1997.

[27] D. Marr and E. C. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London*, B 207:187–217, 1980.

[28] Michael Meissner, Ulrich Hoffmann, and Wolfgang Strasser. Enabling Classification and Shading for 3D Texture Mapping based Volume Rendering using OpenGL and Extensions. In *IEEE Visualization 1999*, pages 207–214, 1999.

[29] Timothy Miller and Robert C. Zeleznik. The Design of 3D Haptic Widgets. In *Proceedings 1999 Symposium on Interactive 3D Graphics*, pages 97–102, 1999.

[30] Shigeru Muraki. Multiscale Volume Representation by a DoG Wavelet. *IEEE Trans. Visualization and Computer Graphics*, 1(2):109–116, 1995.

[31] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro Real-Time Ray-Casting System . In *ACM Computer Graphics (SIGGRAPH '99 Proceedings)*, pages 251–260, August 1999.

[32] Hanspeter Pfister, Chandrajit Bajaj, Will Schroeder, and Gordon Kindlann. The Transfer Function Bake-Off. In *Proceedings IEEE Visualization 2000*, pages 523–526, 2000.

[33] Hanspeter Pfister and Arie E. Kaufman. Cube-4 - A Scalable Architecture for Real-Time Volume Rendering. In *IEEE Symposium On Volume Visualization*, pages 47–54, 1996.

[34] James T. Purciful. Three-Dimensional Widgets for Scientific Visualization and Animation. Master's thesis, University of Utah, June 1997.

[35] Penny Rheingans. Task-Based Color Scale Design. In *Proceedings Applied Image and Pattern Recognition*. SPIE, October 1999.

[36] Guillermo Sapiro. Color Snakes. *CVIU 68(2)*, pages 247–253, 1997.

[37] Yoshinabu Sato, Carl-Fredrik Westin, and Abhir Bhalerao. Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–179, April-June 2000.

[38] Paul S. Strauss and Rikk Carey. An Object-Oriented 3D Graphics Toolkit . In *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 341–349, July 1992.

[39] Colin Ware. Color Sequences for Univariate maps: Theory, Experiments, and Principles. *IEEE Computer Graphics and Applications*, 8(5):41–49, September 1988.

[40] Rudiger Westermann and Thomas Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *ACM Computer Graphics (SIGGRAPH '98 Proceedings)*, pages 169–176, August 1998.

[41] R. C. Zeleznik, K. P. Herndon, D. C. Robbins, N. Huang, T. Meyer, N. Parker, and J. F Hughes. An Interactive Toolkit for Constructing 3D Widgets. *Computer Graphics*, 27(4):81–84, 1993.