Interactive Volume Rendering of Thin Thread Structures within Multivalued Scientific Datasets

Andreas Wenger, Daniel F. Keefe, Song Zhang, David H. Laidlaw

Abstract— **We present a threads and halos representation for interactive volume rendering of vector-field structure and describe a number of additional components that combine to create effective visualizations of multivalued 3D scientific data. After filtering linear structures, such as flow lines, into a volume representation, we use a multi-layer volume rendering approach to simultaneously display this derived volume along with other data values. We demonstrate the utility of threads and halos in clarifying depth relationships within dense renderings, and we present results from two scientific applications: visualization of second-order tensor valued magnetic resonance imaging (MRI) data and simulated 3D fluid flow data. In both application areas, the interactivity of the visualizations proved to be important to the domain scientists. Finally, we describe a PC-based implementation of our framework along with domain specific transfer functions, including an exploratory data culling tool, that enable fast data exploration.**

Keywords—
Scientific Visualization, Diffusion Tensor Imaging (DTI), Fluid Flow Visualization, Medical Imaging, Direct Volume Rendering, Volume Graphics, Volume Shading, Multi-textures, PC Graphics Hardware

I. Introduction

We describe a thread and halo technique suitable for interactive volume rendering of thin linear structures together with a number of components that make it useful for visualization of multivalued 3D scientific data. Two scientific applications drive our volume-rendering work: understanding brain anatomy and pathology, and understanding blood flow in coronary arteries. These driving applications have provided the problems, and, as Brooks suggests, the extent to which our application facilitates the solution of these problems helps to evaluate and guide our algorithm and tool development [1].

Creating comprehensive and accurate visualizations for exploring 3D multivalued data is challenging. The first challenge is to create visualizations in which the data nearer to the viewer does not excessively obscure that farther away. The second challenge is to represent many values and their interrelationships at each spatial location.

Interactive volume rendering with user controlled transfer functions can provide a promising approach for overcoming much of the obscuration problem. By using transparency effectively, transfer functions can be designed that show important features in a dataset throughout a volume. Interactive control allows a scientist to weight the relative importance (usually tied to an opacity level) of data values or features while exploring the dataset.

In our work, we rely heavily on volume rendering techniques. In fact, we use a multi-layer volume rendering approach, similar to [2] to enable us to fully represent multivalued datasets. We also make heavy use of transfer functions and provide interactive controls that are tailored to our application domains.

The authors can be reached at the Brown University Department of Computer Science, Brown University, Providence, RI 02912, {aw,dfk,sz,dhl}@cs.brown.edu

The key contribution of our work is a clear volumetric vector-field representation that can be rendered interactively. Datasets that can benefit from this representation are common in fluid flow research and medical imaging. Our thread and halo representation, shown in Figure 1, together with direct volume rendering, provides clear visual indications of complex linear forms, depth relationships among multiple densely packed threads, and changing data values along the length of the thread. In this paper, we demonstrate that our threads and halos technique can be incorporated into a multi-layer volume rendering scheme and displayed at interactive frame rates on modern consumer graphics cards. We also describe the benefits of such an implementation for our scientific collaborators.

In the next section we discuss related work. We then describe our layered volume rendering framework, threads and halos, and our interactive controls. Results for our two driving applications are then presented and discussed along with some conclusions from this work.

II. Related Work

Below we survey relevant work in diffusion tensor field visualization, vector field visualization, and hardware-accelerated volume rendering.

A. Visualization of Diffusion Tensor Fields

There are several approaches to visualizing diffusion tensor imaging (DTI) datasets. Pierpaoli *et al.* [3] used a 2D array of ellipsoids to visualize a 2D diffusion tensor field. To give a more continuous visual appearance, Laidlaw *et al.* [4] normalize the ellipsoids. They also use concepts from oil painting, mapping data components onto brush strokes and building up the strokes in multiple layers, to represent more values in the data. None of these 2D methods generalize well to 3D.

Kindlmann *et al.* [5] attacked the problem of obscuring data points in 3D with a direct-volume-rendering approach: at every data point they assign an opacity and color based on the underlying diffusion tensor dataset. However, it is still difficult to pick out anatomically distinct regions of the brain and understand their connectivity. The direct volume rendering portion of our work is similar to this approach, but makes connectivity information more apparent by using a coloring and lighting scheme based on diffusion magnitude and diffusion anisotropy measurements.

Delmarcelle and Hesselink [6] introduced hyperstreamlines, a method that captures all of the attributes of tensors along a path. In our visualizations we represent the type of diffusion along a path through color coding rather than a change in cross section shape.

Several improvements to the basic concept of integrating

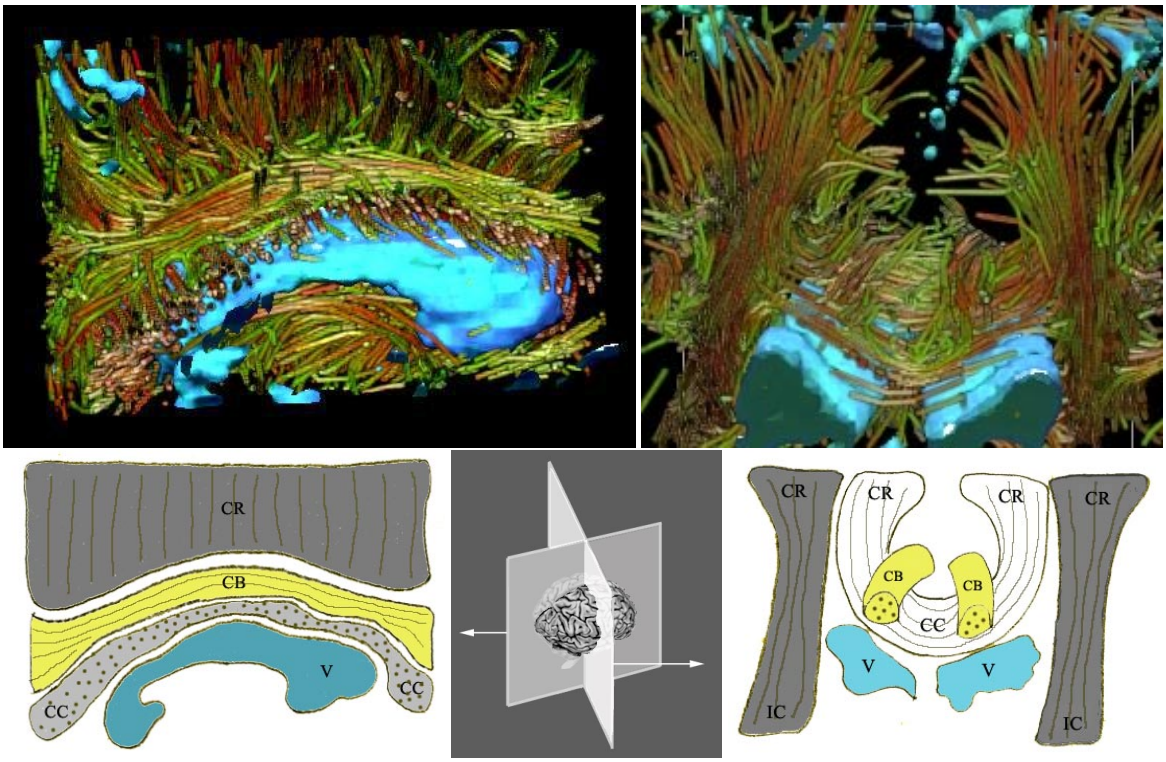

Fig. 1. Interactive renderings of a human brain dataset. The renderings (top) show collections of threads consistent with major white-matter structures: IC=internal capsule, CR=corona radiata, CB=cingulum bundle, CC=corpus callosum, diagrammed on the bottom. Components of the tensor-valued data control thread direction, color, and density as described in the text. Direct volume rendering simultaneously shows the ventricles (labeled V) in blue for anatomical context.

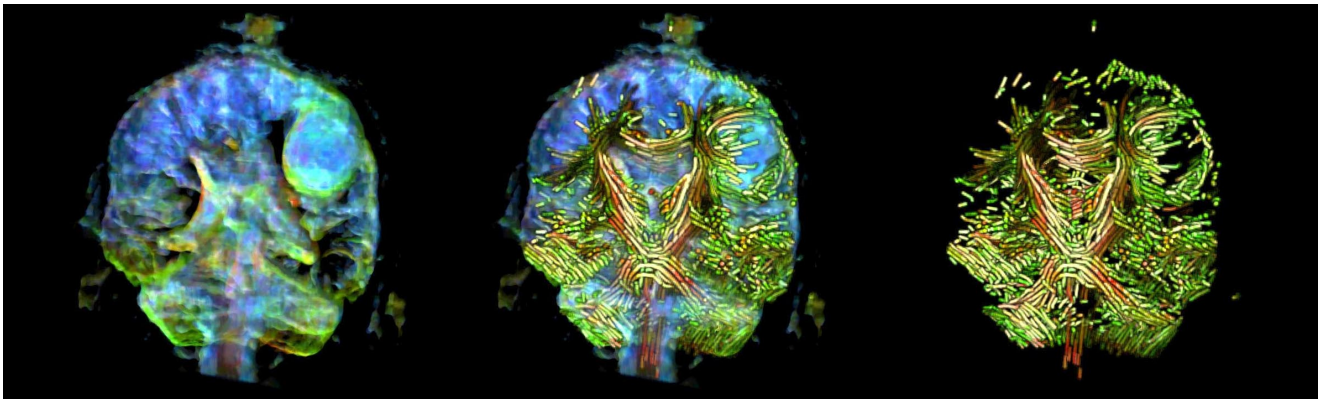

Fig. 2. Left: a direct-volume-rendered layer showing regions with different diffusion anisotropy. Right: a thread layer showing both the anisotropy and diffusion direction. The two layers are combined in the center image which shows significantly more information and requires little additional visual bandwidth.

paths along the principle eigenvector field were suggested in [7] and [8], mainly to stabilize the propagation in isotropic regions. Basser *et al.* [9] calculated the trajectories of neural fibers in brain white matter that were generated from the diffusion tensor field by integrating along the eigenvector with the largest eigenvalue. Zhang *et al.* [10] used this method to generate streamtubes to visualize continuous directional information in the brain. We extend Zhang et al.'s algorithm to continue streamtubes through areas with planar anisotropy. In addition, we filter the resulting paths into a densely packed thread volume rather than representing them as polygonal models.

B. Visualization of Vector Fields

Of the extensive work on creating effective vector field visualizations, the following two papers are most closely related to our work. Interrante and Grosch [11] visualized 3D flow with volume line integral convolution (LIC). As they demonstrated with offline rendering, their visibility-impeding halos improve depth perception and help make complex 3D structures easier to analyze. Our technique builds on this work to produce a similar effect interactively.

Zöckler *et al.* [12] introduced illuminated field lines to visualize 3D vector fields. Our illuminated thread representation is similar, but our volumetric rendering approach renders at a rate independent of the tube or line complexity and combines with

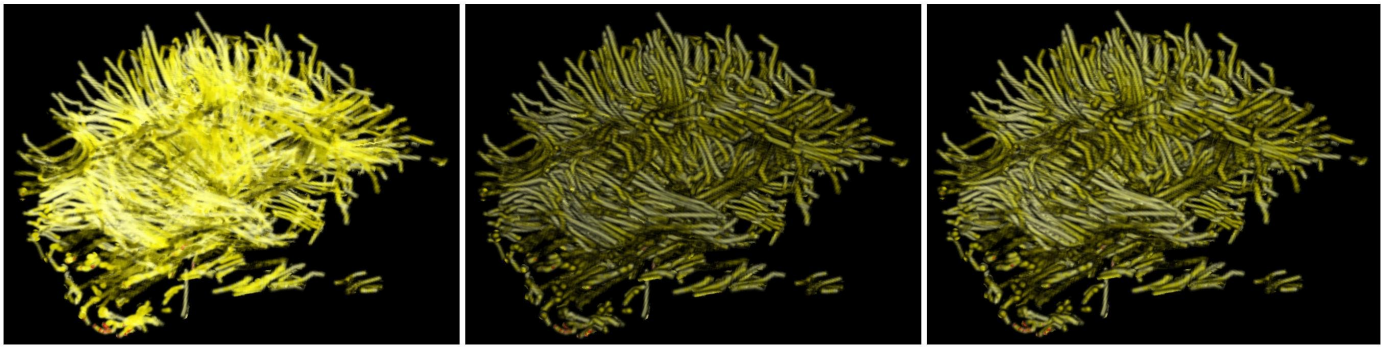

Fig. 3. Three thread volumes showing brain connectivity information (the front of the head points right). Shown without halos on the left, with halos in the center, and after slightly shifting the halos away from the viewer to brighten the threads on the right.

our other volumetric layers to create visualizations that convey more information.

Li *et al.* [13] presented a volumetric texture based method to visualize 3D vector fields. They scan convert properties of the streamlines into a volumetric texture and then use these properties to look up color and opacity information in an appearance texture. Our approach differs in that our threads, while individually less visually complex, are much thinner and more densely packed; we also represent data more complicated than vector fields.

C. Hardware-Accelerated Volume Rendering

Cabral *et al.* [14] introduced a 3D texture approach for volume rendering using view-aligned slices that exploit trilinear interpolation. In addition, we make significant use of hardware texture compression to reduce texture memory consumption.

There are also several volume-rendering implementations that make use of dedicated hardware [15] or distributed hardware [16], that are capable of visualizing multi-valued volumetric datasets.

Our multi-layer volume rendering is closely related to the two-level volume rendering presented by Hauser *et al.* [2]. In this scheme, the two levels are an object level and a global level. Different rendering styles, such as direct volume rendering, maximum intensity projection, or value integration, are used for each level. Our system is based on the same concept of rendering multiple volumes of information into the same visualization space. However, we use separate volumetric datasets for our layers of information, rather than classifying a single volume of data as either focus or context. In some cases our halos and threads could be conceptualized as together forming a *focus* level for the visualization with any additional direct volume rendered layers forming the *context* level.

Kniss *et al.* [17] use interactive transfer functions operating on directional derivative values to select boundaries in scalar-valued datasets. We use this technique to visualize our scalar-valued datasets, although with less sophisticated interactive manipulation widgets. In [18], multi-dimensional transfer functions and dual-domain interaction are applied to multivariate meteorological data. They found, as we did, that multi-dimensional transfer functions provide a powerful tool to explore multivariate datasets.

Lum and Ma [19] implemented a hardware-accelerated parallel nonphotorealistic volume renderer that uses multi-pass rendering on consumer-level graphics cards. Their system emphasizes edges or depth ordering using artistically motivated techniques. Like Lum and Ma, we utilize multiple rendering passes to enhance visual cues, but our rendering is targeted to exploratory visualization of multi-valued data, which has significant implications for the interface, implementation, and results.

Stompel *el al.* [20] use nonphotorealistic (NPR) volume rendering to more effectively visualize multivariate volumetric data. They use stroke rendering to display a vector field simultaneously with a scalar field and produce several NPR effects, including silhouettes. Their silhouettes help to emphasize depth discontinuities just like our halos but will not work for features as small as our threads because a reliable gradient cannot be calculated.

D. Hair, Fur, and Thread Rendering

Several hair and fur rendering algorithms inspired our work. Kajiya and Kay [21] introduced texels to render realistic-looking fur. Kajiya and Kay also developed a Phong-like lighting model for fur; our approach is similar but targets free-floating threads. Instead of providing parameters for lighting, we store derived values from the multivalued datasets along with tangent and density values throughout the volume.

Lengyel [22] uses a volumetric texture approach to render short threads in real time. Unlike his short threads, our data-defined threads remain individually distinguishable. Like Lengyel, we use Banks' [23] hair-lighting model but with a different implementation appropriate for volume rendering.

III. A Layered Volume-Rendering Framework

Our visualization framework has four steps. We begin with primary multivalued volumetric data.

Calculate Derived Datasets. Since the primary data is often difficult to interpret directly, our first step is to calculate derived volumes of data with more intuitive interpretations. For example, DTI datasets are second-order tensor fields. It is often useful to decompose these into several scalar and vector fields to reduce the problem of rendering a tensor field into one of rendering several simpler fields.

Define Visual Abstractions. In the abstraction step, we group

the data volumes into layers of visual representations that are filtered into and stored in separate volumes. The most simple abstraction is a straight mapping of a scalar value, such as speed, to a volume. Threads are another abstraction that are good at illustrating vector fields. Any conversion of the derived data to a clearer visual representation fits into this step of the framework.

Map Data with Interactive Transfer Functions. The mapping step defines transfer functions that assign color and opacity to the volume layers and shader programs that control lighting parameters.

Visualize and Explore. In the final step of the framework, we render the multiple volumes together in the same visualization space and use interaction widgets to control the appearance of each layer.

IV. Threads and Halos

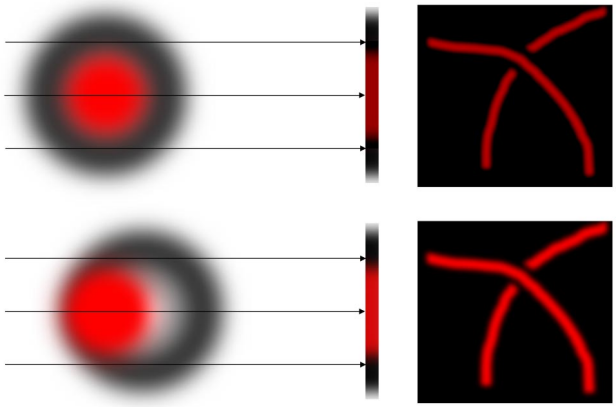

Fig. 4. Black arrows illustrate the path of virtual viewing rays through a cross section of a volumetric thread and halo. When the rays, coming from the viewer's perspective, pass through part of the dark halo volume before reaching the red thread, the resulting color is a darker red than when the halo volume is shifted away from the viewer (the bottom case).

We represent continuous directional information using threads. We densely populate the volume with threads so as to represent as much of the underlying data as possible. To clarify individual threads, we add a "halo," modeled after those presented by Interrante and Grosch [11]. Compare the images in Figure 3 to see the depth clarifying effect of using halos with the threads.

The threads and halos are each stored in a volume texture that is precomputed. There is exactly one halo in the halo volume for each thread in the thread volume. Each halo follows the same path as its thread but has a slightly larger radius. Figure 4 demonstrates how the halos extend beyond the sides of the threads when rendered to obscure any geometry behind them.

Unfortunately, the halo also slightly obscures the thread from a frontal view. This has the effect of darkening the thread rather than completely hiding it since the halo is semi-transparent and there is only a small amount of it in front of the thread. This is seen in the darker, middle image in Figure 3. We compensate for the darkening effect to produce images like the one on the right of the figure by shifting the entire halo volume so that there is less halo between the thread and the viewer.

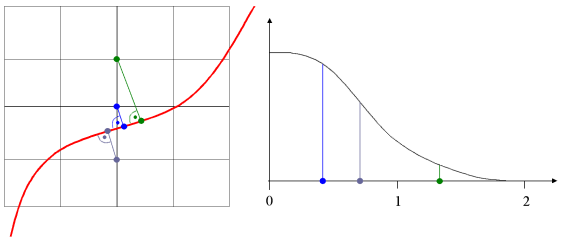

Fig. 5. Filtering a thread into a volume (2D view). For each voxel within a radius of two voxels from a thread, we use the shortest distance to the thread as the input to the filter (at right). The grid on the left and the horizontal axis on the right both show single-voxel spacing.

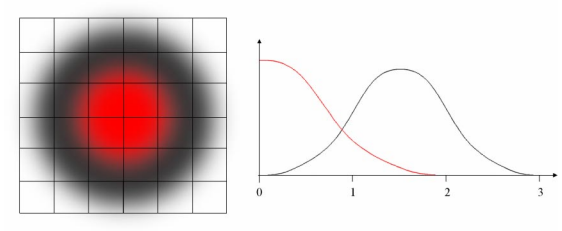

Fig. 6. Filtering a halo into a volume (2D view); red depicts the thread and black the halo around it. The red curve is the filter for the thread and the black curve is the filter with which the halo is generated. The grid on the left and the horizontal axis on the right both show single-voxel spacing.

As illustrated in Figure 4, the virtual viewing rays that pass through only the halo on the silhouette of each thread will produce a black color. If a portion of the halo exists in front of the thread with respect to the viewer, as in the top of the figure, then the viewing ray passing through this area will produce a color partly saturated with red and partly with black, resulting in dark red. If the halo is offset away from the viewer at a distance of one voxel, then the viewing rays pass through far less, or none in the case of the middle ray, of the halo volume before reaching the red color of the thread. The result is a brighter red color for the thread.

The threads and halos are filtered into volumes using a cubic B-spline filter that has a support of four voxels for the thread (see Figure 5) and six voxels for the halos (see Figure 6). Since the threads and halos are represented in volumes, the rendering time is independent of the number of threads displayed. However, the diameter of the threads is limited by the resolution of the volume.

Lighting for the threads is calculated using a restricted version of the lighting model in [23], which defines intensity I as

$$I = k_d I_t \left(\sqrt{1-(T\cdot L)^2}\right)^p + k_s \left(\sqrt{1-(T\cdot H)^2}\right)^n \quad (1)$$

Here I_t is the combined output color from the transfer functions, T the tangent, L the light vector, H the vector halfway between the light vector and the view vector, n the specular exponent, p the excess-brightness diffuse exponent, and k_d and k_s the diffuse and specular contributions respectively. Our restricted lighting model assumes directional lighting and a constant halfway vector for the entire volume.

We have implemented several thread lighting models, including those of Kajiya and Kay [21], and Banks [23] with excess

Fig. 7. A sequence of renderings of a thread density volume with increasing length threshold from left to right. The rightmost image shows only long threads.

brightness exponents of $p = 2$ and $p = 4$. Kajiya and Kay's lighting model is similar to a Banks model with $p = 1$. Banks actually uses a value of around 4.8. With a small exponent, the threads become brighter and the lighting is less dramatic. We found $p = 2$ to be a good value for our applications and also a speedy one, since it does not require a square-root calculation.

V. Layering Volumes

Our volume-renderer implementation uses a single stack of view-aligned texture-mapped slices, rendered back to front and blended with weights $1-\alpha$ and α. Each slice is rendered multiple times, once for each volume layer. Layers of each slice are also blended with the weights $1-\alpha$ and α. We render our direct volume rendered layers first, the halos second, and the threads third. For our applications, this is equivalent to rendering the layers in the order of the scale of their largest structures. The direct-volume rendered layers tend to reveal large structures and surfaces that are easy to make out underneath the more finely detailed thread and halo layers. Our intuition is that in most cases this represents an acceptable heuristic to use when determining layer ordering.

For direct volume rendered layers, a Phong lighting model (Eq. (2)) is used. As in the thread lighting model described above, I_t is the combined output color from the transfer functions, N is the normal, L the light vector, H the halfway vector, k_a the ambient contribution, k_d the diffuse contribution, k_s the specular contribution, and n the specular exponent:

$$I = k_a I_t + k_d I_t (N \cdot L) + k_s (N \cdot H)^n \qquad (2)$$

Visualizing multiple layers of volumetric data requires an extensive amount of texture memory. We utilize the OpenGL extension ARB_texture_compression, which provides a 4:1 compression ratio. Thus, a 256^3 eight bit per channel RGBA texture can be reduced from 64MB to 16MB. With this scheme, we can fit the multiple volume textures required in memory on commodity graphics cards.

VI. Exploratory Culling and Transfer Function Manipulation

Interactive editing of transfer functions has become commonplace in volume rendering applications. We describe several application specific transfer function modes in the following section.

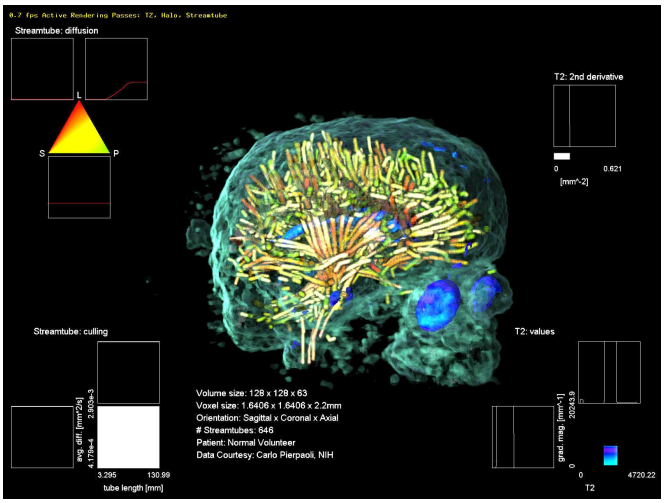

Fig. 8. The interactive exploration tool. Clockwise from upper left are a 2D barycentric widget, a 1D widget, a 2D Cartesian widget, and a 2D Cartesian culling widget.

Exploratory culling applies a transfer function to an entire thread and halo based on attributes of the linear structure it represents. This is the approach that Wei *et al.* take in [24] to cull discrete particle tracks based on the track's energy. Doleisch *et al.* [25] introduce a more general framework in which degree-of-interest functions are defined to perform similar tasks. In our approach the entire thread and its halo are classified according to a metric i.e. thread length which is mapped to a $0\ldots255$ domain. Every voxel in the data volume belongs to one class of threads. The cost of classifying the threads and halos for this example is an additional byte per voxel and a 1D transfer function that takes up 256 additional bytes of texture memory.

In our brain visualizations, both long and short threads are important and provide different types of insight into the data. We use this culling feature to interactively select a subset of threads to display based on their average diffusion rate or their length, as seen in Figure 7. This approach is a significant advance over the state of the art in this application area. Similar culling in Zhang *et al.*'s [26] approach required an entire preprocessing step taking between several minutes to several hours.

We provide several on-screen widgets, shown in Figure 8, to

control transfer functions of the form 1D, 2D, and 2D barycentric. Colors are manipulated within the hue, saturation, value, and transparency (HSVα) color space. For the 1D and 2D transfer function widgets, color and opacity can be controlled interactively along each axis. In the multidimensional cases, the colors of the axes are averaged, while the opacities are combined multiplicatively. In our informal trials, these combination methods seemed most intuitive. The 2D barycentric manipulation widget, shown in the top left of Figure 8, is ideal for the brain visualization application since the space maps naturally to the anisotropy metrics defined in [27].

VII. Neuroimaging Results and Discussion

Our neuroimaging data are acquired using magnetic resonance imaging (MRI) and are of two types: second-order tensor-valued water-diffusion-rate images and scalar-valued anatomical images. At each point in a volume, the tensor-valued data capture the rate at which water is diffusing through tissues. That rate is different in different areas – in regions of pure fluid, it is fast; in tissues like bone, it is slow. The rate of diffusion can also be directionally dependent, particularly in fibrous tissues like axon tracts and muscles, diffusing more quickly along the fibers than across them. The scalar-valued data are typical T2-weighted MR images.

Within the second-order tensor field measuring the water diffusion rate, each value D is a symmetric tensor with real, positive eigenvalues. From D we derive several other measures. First, three scalar anisotropy measures introduced by Westin [27], c_l, c_p, and c_s, describe how close to a line, a plane, or a sphere the corresponding ellipsoid shape is for a given measurement. Second, the trace of D, $\mathrm{Tr}(D)$, is equivalent to the sum of the eigenvalues of D and gives a scalar measure of the overall diffusion rate. Third, the gradient of the trace, $\nabla\mathrm{Tr}(D)$ and its magnitude, $|\nabla\mathrm{Tr}(D)|$, describe how the diffusion rate is changing and in what direction; we use these quantities in lighting calculations for the direct volume rendered layer i.e. in Figure 2.

The fourth category of derived data is a set of threads and halos through the tensor field that represent the directions of diffusion. These are calculated and distributed within the volume as described by Zhang *et al.* [10][28][26]. They follow the direction of fastest diffusion in linear regions. In planar regions, they stay within the plane formed by the major and medium eigenvectors, following whichever is more consistent with the path to that point. They are not present in isotropic regions.

From the T2-weighted image scalar field we derive the gradient of the value and the gradient magnitude, which help define how fast the value is changing and in which directions. We use these quantities in lighting calculations. We also derive the second directional derivative to help define boundaries between homogeneous regions.

Figure 9 shows the mapping from the scalar- and tensor-valued volumes onto a direct volume rendered layer, a thread layer, and a halo layer as seen in Figure 8. The first layer directly renders the T2-weighted image. The hyper-intense ventricle regions were selected by interactively editing two transfer functions that are combined multiplicatively to yield α values for the layer. Color is specified only through the transfer function controlling the T2 image value and the gradient magnitude.

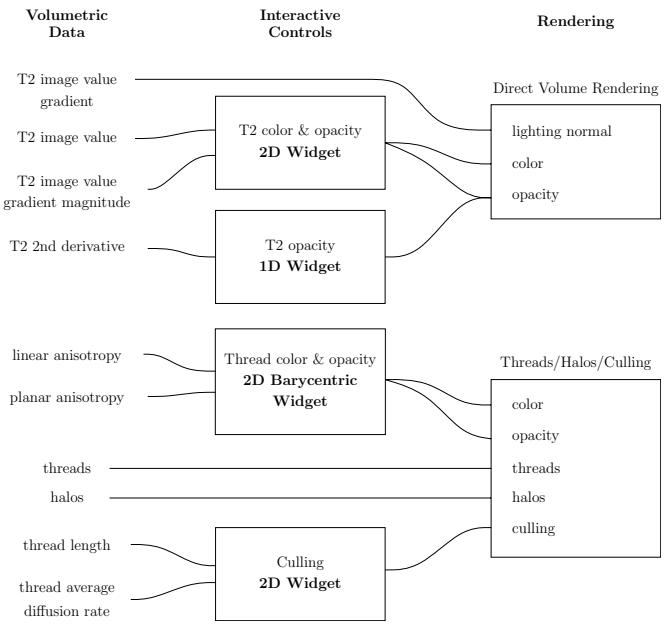

Fig. 9. A data-flow diagram of the rendering setup for Figure 8.

The second layer renders halos for the threads, and the third layer renders the threads. The visible portions for these layers are interactively selected via three criteria. First, a transfer function maps the anisotropy metrics to an α value. For this rendering, areas of all types of anisotropy are shown. Second, each thread and halo can be selected via exploratory culling based on the thread's length and on the average diffusion rate along it. In this rendering all threads are shown. Third, the thread density is provided directly by the precalculated thread volume. Likewise, halo density is provided directly by the halo volume. Each of the results for this dataset is rendered with 256^3 volume textures.

Our neuroscientist collaborators gained several insights into their data through these visualizations. Figure 1 shows detail of a diffusion dataset from a normal volunteer. A number of large white-matter structures are clearly visible, including the corpus callosum, internal capsule, corona radiata, and cingulum bundle.

Figure 2 shows a dataset from a patient with a brain tumor. Direct volume rendering captures the tumor as an opaque mass and threads show the diffusion direction and variation in anisotropy around it. Note the cradle of threads surrounding the tumor. Using this exploratory visualization has enabled our collaborators to discover a relationship between the different types of anisotropy around tumors. In particular, there is a notable increase in planar anisotropy (shown as green) in the area around the tumor [29].

VIII. Simulated Blood Flow Results and Discussion

Our second scientific application involves simulated fluid flow data on incompressible flow through an idealized model of a bifurcating coronary artery. We have been studying how the flow structure is related to atherosclerotic lesion formation. Lesions tend to form just downstream of branches. They form on the wall of each branch opposite the exit point of the other

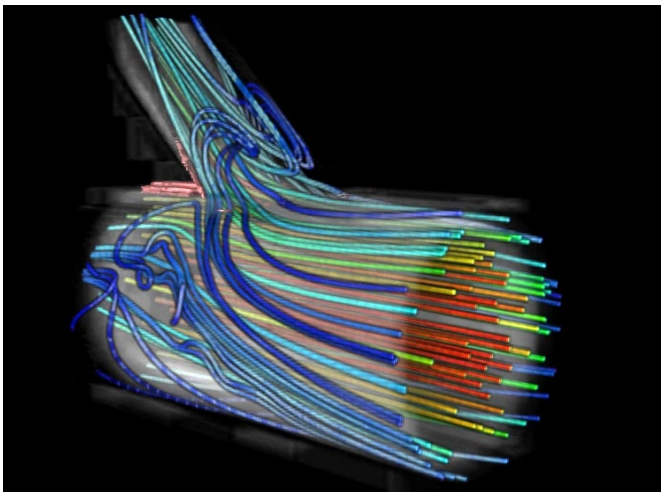

Fig. 10. Simulated flow, from right to left, through a model of a branching coronary artery. Several complex structures can be seen, including reversal of flow direction, as illustrated by the blue-haloed threads in the side branch immediately downstream from the bifurcation. The semi-transparent white shell represents vorticity magnitude and gives near-the-wall context for the threads.

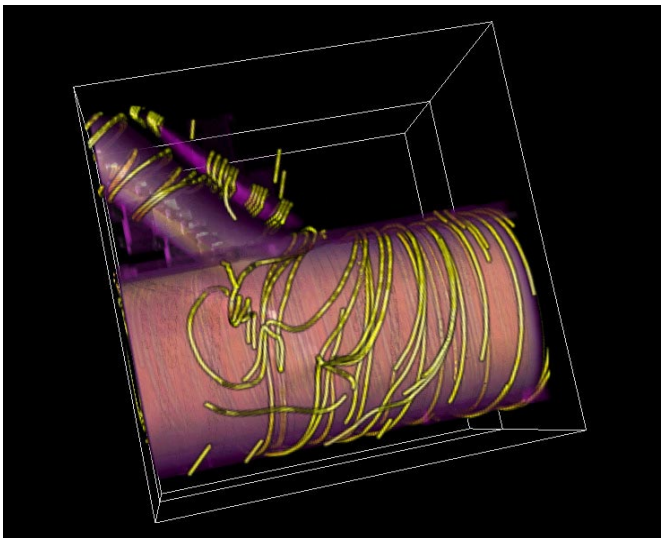

Fig. 11. Integral curves through the vorticity vector field together with the velocity magnitude field for the flow illustrated in Figure 10. These vortex lines give additional clues to the flow structure, particularly in areas where it curves and changes speed.

branch. We hypothesize that upstream flow structures may provide important insight into why this happens. Our renderings depict one time-step of the simulated, pulsatile flow.

The primary data for this application area is a 3D velocity field. From this, a number of quantities are useful to derive. Speed is one. Another is vorticity, a vector field, that is a component of the first derivative of the velocity and captures the local rotational component of the flow. The vorticity vector points along the axis of rotation and its magnitude indicates the rate of rotation.

Figure 10 shows an idealized model of a branching coronary artery. Flow is from right to left, starting in the open end and proceeding down the main artery and into the side branch. Haloed threads, colored according to speed, are rendered together with a diaphanous shell showing relatively low-vorticity regions. A more opaque pink section right at the point of bifurcation shows the region of highest vorticity.

The same flow is rendered in Figure 11. In this image yellow threads show vortex lines integrated through the vorticity vector field. The semi-transparent purple form shows low-speed regions.

These two images together show important flow features not seen with other visualization methods, including near-wall kinks in vortex lines and localized looping structure in the vorticity. The interactive lighting in this visualization helped to make the correlation between these features more apparent to our collaborators. The kinks tended roughly to fit into the upstream edges of separation pockets evident in velocity images. Both of these datasets were rendered with 256^3 volume textures.

Based on the results of these visualizations, we attempted to create an even more compelling visualization of the correlation between velocity and vorticity by layering velocity and vorticity lines together in the visualization in Figure 12. This visualization has two separate sets of thread and halo volumes. To fit both sets of threads and halos in texture memory we had to crop the volume size to $256 \times 256 \times 128$. Visualizing the two vector

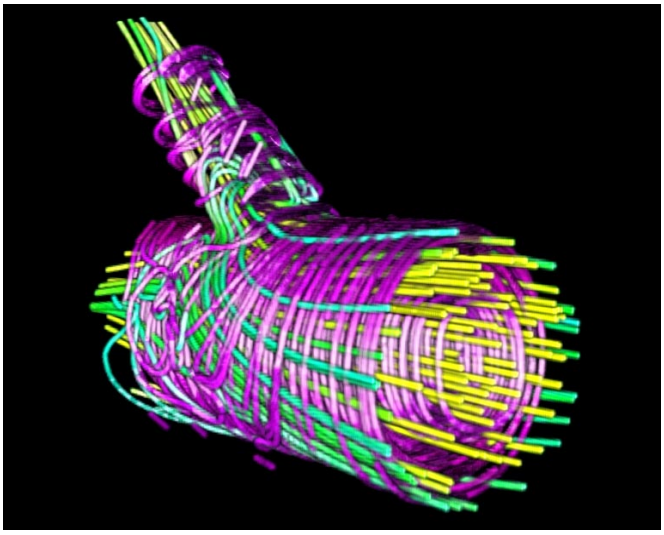

Fig. 12. Integral curves through both the velocity (yellow and green) and vorticity (purple and pink) vector fields for the same flow as illustrated in Figure 10. Correlations among these vector fields can represent important flow structures.

fields simultaneously clarifies correlations among features quite dramatically. This mass of complex linear flow structures is difficult to interpret without the aid of the depth cue enhancing halos.

Typical rendering rates for both of our application areas are four to five frames per second. Clipping planes help isolate regions of interest and increase the frame rate so that zooming in on those sections does not significantly slow rendering rates. In most instances, close to a one frame per second reduction in the frame rate can be seen for each additional volume layer that is rendered.

IX. Conclusion

In this paper we present a novel technique for generating and interactively rendering a thread and halo representation for mul-

tivariate datasets that include some fields with important linear features, such as flow lines. This work was motivated by recent advances in graphics hardware, volume rendering, halo and other art-based rendering effects, and hair lighting models. In addition to presenting our approach to interactively volume render thin threads with halos, we also present a successful fusion of these varied recent research results in the form of an application motivated by multiple scientific problems. This is an important secondary contribution of our work.

We have been driven in our exploration of these concepts by our collaborations with scientists in both of our application domains. Feedback from both groups indicates that they find our interactive volume renderer effective in exploring both kinds of multivalued datasets. Currently these data are often not well understood and exploring them will lead to new scientific hypotheses and insights.

X. Acknowledgments

The authors thank the Brown Scientific Visualization Group, Graphics Group, and Technology Center for Advanced Scientific Computing and Visualization. Special thanks go to George Karniadakis and his group at Brown, Peter Richardson at Brown, Susumu Mori at Johns Hopkins, Mark Bastin at the University of Edinburgh, and Thomas Deisboeck at MGH for their data and feedback. Thanks also to Katrina Avery and Morriah Horani for superb editorial input. This work was partially supported by NSF (CCR-0093238) and the Human Brain Project (EB00232: NIBIB and NIMH).

References

[1] F. P. Brooks, "The computer scientist as toolsmith II," *CACM*, vol. 39, no. 3, pp. 61–68, 1996.

[2] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Groeller, "Two-Level Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 3, pp. 242–252, 2001.

[3] C. Pierpaoli and P. Basser, "Toward a Quantitative Assessment of Diffusion Anisotropy," *Magnetic Resonance Magazine*, pp. 893–906, 1996.

[4] D. H. Laidlaw, E. T. Ahrens, D. Kremers, M. J. Avalos, R. E. Jacobs, and C. Readhead, "Visualizing Diffusion Tensor Images of the Mouse Spinal Cord," in *Proceedings IEEE Visualization '98*. IEEE, 1998, pp. 127–134.

[5] G. L. Kindlmann and D. M. Weinstein, "Hue-Balls and Lit-Tensors for Direct Volume Rendering of Diffusion Tensor Fields," in *Proceedings IEEE Visualization '99*. IEEE, 1999, pp. 183–189.

[6] Th. Delmarcelle and L. Hesselink, "Visualizing Second-Order Tensor Fields with Hyperstreamlines," *IEEE Computer Graphics and Applications*, vol. 13, no. 4, pp. 25–33, 1993.

[7] D. M. Weinstein, G. L. Kindlmann, and E. C. Lundberg, "Tensorlines: Advection-Diffusion based Propagation through Diffusion Tensor Fields," in *Proceedings IEEE Visualization '99*. IEEE, 1999, pp. 249–254.

[8] L. Zhukov and A. H. Barr, "Oriented Tensor Reconstruction: Tracing Neural Pathways from Diffusion Tensor MRI," in *Proceedings IEEE Visualization 2002*. IEEE, 2002, pp. 387–394.

[9] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In Vivo Fiber Tractography Using DT-MRI Data," *Magnetic Resonance in Medicine*, vol. 44, pp. 625–632, 2000.

[10] S. Zhang, C. T. Curry, D. S. Morris, and D. H. Laidlaw, "Streamtubes and Streamsurfaces for Visualizing Diffusion Tensor MRI Volume Images," in *Works in Progress IEEE Visualization 2000*. IEEE, 2000.

[11] V. Interrante and Ch. Grosch, "Strategies for Effectively Visualizing 3D Flow with Volume LIC," in *Proceedings IEEE Visualization '97*. IEEE, 1997, p. 421.

[12] M. Zöckler, D. Stalling, and H.-Ch. Hege, "Interactive visualization of 3D-Vector fields using illuminated streamlines," in *Proceedings IEEE Visualization '96*. IEEE, 1996, pp. 107–113.

[13] G. Li, U.D. Bordoloi, and H. Shen, "Chameleon: An interactive texture-based rendering framework for visualizing three-dimensional vector fields," in *Proceedings IEEE Visualization 2003*. IEEE, 2003, pp. 241–248.

[14] B. Cabral, N. Cam, and J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," in *ACM Symposium On Volume Visualization*. ACM, 1994, pp. 91–98.

[15] Y. Wu, V. Bhatia, H. Lauer, and L. Seiler, "Shear-image order ray casting volume rendering," in *Proceedings of the 2003 symposium on Interactive 3D graphics*. 2003, pp. 152–162, ACM Press.

[16] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, Ch. Hansen, and P. Shirley, "Interactive ray tracing for volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 238–250, 1999.

[17] J. M. Kniss, G. L. Kindlmann, and Ch. Hansen, "Multidimensional Transfer Functions for Interactive Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 270–285, 2002.

[18] J. M. Kniss, C. Hansen, M. Grenier, and T. Robinson, "Volume rendering multivariate data to visualize meteorological simulations: A case study," in *Proceedings of Eurographics - IEEE TCVG Symposiumon Visualization*, 2002, pp. 189–194.

[19] E. B. Lum and K.-L. Ma, "Hardware-Accelerated Parallel Non-Photorealistic Volume Rendering," in *Proceedings of the Non-Photorealistic Animation and Rendering Conference 2002*, 2002, pp. 67–74.

[20] A. Stompel, E. B. Lum, and K.-L. Ma, "Visualization of Multidimensional, Multivariate Volume Data Using Hardware-Accelerated Non-Photorealistic Rendering Techniques," in *10 th Pacific Conference on Computer Graphics and Applications (PG'02)*, 2002, p. 394.

[21] J. T. Kajiya and T. L. Kay, "Rendering Fur with Three Dimensional Textures," in *Proceedings SIGGRAPH '89*. ACM, 1989, pp. 271–280.

[22] J. E. Lengyel, "Real-Time Fur," in *Eurographics Rendering Workshop 2000*. ACM, 2000, pp. 243–256.

[23] D. C. Banks, "Illumination in Diverse Codimensions," in *Proceedings SIGGRAPH '94*. ACM, 1994, pp. 327–334.

[24] X. Wei, A. Kaufman, and T. J. Hallman, "Case Study: Visualization of Particle Track Data," in *Proceedings IEEE Visualization 2001*. IEEE, 2001, pp. 465–468.

[25] H. Doleisch, M. Gasser, and H. Hauser, "Interactive feature specification for focus+context visualization of complex simulation data," in *Proceedings of the Symposium on Data Visualisation 2003*, 2003, pp. 239–248.

[26] S. Zhang, Ç. Demiralp, and D. H. Laidlaw, "Visualizing Diffusion Tensor MR Images Using Streamtubes and Streamsurfaces," *IEEE Transactions on Visualization and Computer Graphics*, 2003, inpress.

[27] C.-F. Westin, S. Peled, H. Gubjartsson, R. Kikinis, and F.A. Jolesz, "Geometrical Diffusion Measures for MRI from Tensor Basis Analysis," in *Proceedings of ISMRM 1997*, 1997, p. 1742.

[28] S. Zhang, Ç. Demiralp, D. Keefe, M. DaSilva, B. D. Greenberg, P. J. Basser, C. Pierpaoli, E. A. Chiocca, T. S. Deisboeck, and D. Laidlaw, "An immersive virtual environment for DT-MRI volume visualization applications: A case study," in *Proceedings IEEE Visualization 2001*. IEEE, 2001, pp. 437–440.

[29] S. Zhang, M. E. Bastin, D. H. Laidlaw, S. Sinha, P. A. Armitage, and T. S. Deisboeck, "Visualization and analysis of white matter structural asymmetry in diffusion tensor MR imaging data," *Magnetic Resonance in Medicine*, vol. 51, no. 1, pp. 140–147, 2004.