Strategies for Direct Volume Rendering of Diffusion Tensor Fields

Gordon Kindlmann, David Weinstein, David Hart

Scientific Computing and Imaging
Department of Computer Science, University of Utah
{gk|dmw|dahart}@cs.utah.edu

Abstract

Diffusion-weighted magnetic resonance imaging is a relatively new modality capable of elucidating the fibrous structure of certain types of tissue, such as the white matter within the brain. Volume rendering is one tool for interpreting this data, because volume rendering permits the visualization of three-dimensional structure without a prior segmentation process. In order to use volume rendering, however, we must develop methods for assigning opacity and color to the data, and create a method to shade the data to improve the legibility of the rendering. Previous work introduced three such methods: barycentric opacity maps, hue-balls (for color), and lit-tensors (for shading). The current paper expands on and generalizes these methods, describing and demonstrating further means of generating opacity, color, and shading from the tensor information. We also propose anisotropic reaction-diffusion volume textures as an additional tool for visualizing the structure of diffusion data. The patterns generated by this process can be visualized on their own, or they can be used to supplement the volume rendering strategies described in the rest of the paper. Finally, because interpolation between data points is a fundamental issue in volume rendering, we conclude with a discussion and evaluation of three distinct interpolation methods suitable for diffusion tensor MRI data.

1 Introduction

A fundamental property of biological tissue is the ability of water molecules to move within it by the action of Brownian motion. Rather than being one fixed velocity, this movement, called diffusion, is often anisotropic – happening faster in some directions than others. To a good approximation, the diffusion rate's directional dependence can be represented with a 3×3 real-valued symmetric matrix. This matrix representation of the diffusion tensor can be calculated from a sequence of diffusion-weighted MRI images.

To provide a feel for measured diffusion tensor data, a slice of a human brain dataset is portrayed in Figure 1. Each sub-image in the matrix of images is a gray-scale representation of the corresponding component of the tensor matrix, with medium gray representing zero. In the brain interior, the on-diagonal components of the tensor matrix are positive, while the off-diagonal components can be either positive or negative. This method of portraying the raw tensor data is not novel, nor is it a very intuitive way to display the orientation and shape of the diffusion tensors (in the same way that looking at the individual components of a vector field gives a poor sense of the field's structure).

All 3×3 real-valued symmetric matrices have three real eigenvalues and three real-valued orthogonal eigenvectors [29]. The diffusion tensor matrix enjoys the additional constraint of having non-negative eigenvalues, implying it can be unambiguously represented as an ellipsoid. The ellipsoid's major, medium, and minor axes are along the tensor's eigenvectors, with the scalings along the axes being the eigenvalues. Such an ellipsoid is the image of the unit sphere under the linear transform induced by the tensor's ma-

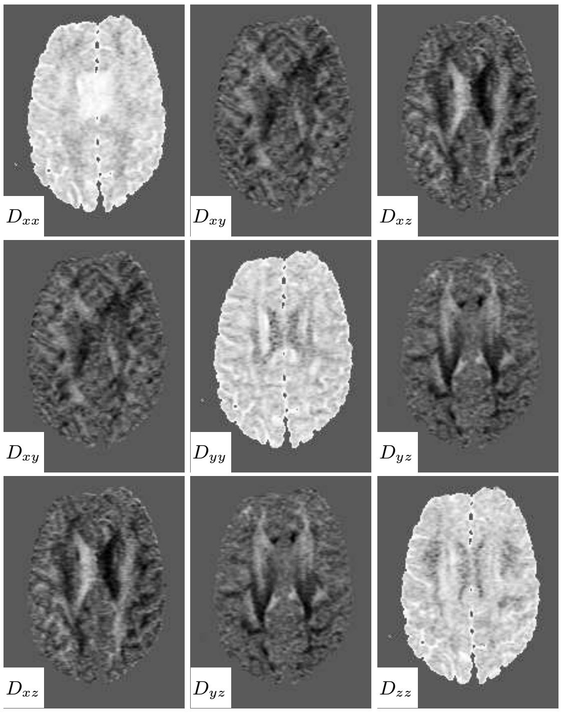


Figure 1: Matrix of images showing the individual tensor components within one dataset slice.

trix representation[1].

The ellipsoid provides a concise and elegant way to visualize the tensor because it has a simple shape, and it has just as many degrees of freedom as the diffusion tensor. As such, we will use the ellipsoid representation for demonstration purposes in figures. Also, previous work in diffusion tensor visualization has used arrays of ellipsoids to depict the tensor field within a two dimensional region. Another tensor visualization method, hyperstreamlines, succeeds in faithfully depicting the tensor along one-dimensional paths in a volume dataset. These methods are useful because they produce a means of visually decoding *all* the tensor's degrees of freedom at some set of locations in the field.

We believe that in this context, however, the most informative visualizations do not necessarily come from striving to pack many dimensions of information into one image. Rather, it may be desirable to create renderings of tensor datasets by displaying only *some* of the information, but *everywhere* within a volume. The goal of this research is creating an understanding of the fibrous structure of white matter throughout the brain. Because the white matter fiber tracts connect major regions of the brain, a detailed understanding of their structure could foster advances in neuroanatomy, in surgical planning, and cognitive science [20, 7, 19]. Fortunately, developments in magnetic resonance imaging have made it possible to accurately measure the water diffusion tensor within living brain tissue [2]. The white matter fiber tracts can be distinguished from their surroundings based on properties of the measured diffusion tensor, such as its anisotropy. Visualizing the fiber tracts is inherently a three-dimensional problem because of their curving, intricate structure. A technique that allows us to visualize the large scale patterns across the entire dataset is ideal.

Since this has historically been the goal of direct volume rendering for *scalar* data, we have explored the use of direct volume rendering for diffusion tensor visualization. To make this possible, the various ingredients of the direct volume rendering algorithm need to be supplied from the tensor data. The barycentric opacity map, lit-tensor, and hue-ball techniques introduced in [17] are specific approaches to performing three tasks fundamental to volume rendering: determining opacity, calculating shading and assigning material color.

In the current paper, we generalize our previous techniques to offer more choices in how to accomplish the basic tasks of volume rendering. While barycentric opacity maps can control which types of anisotropy appear in the final image, *barycentric color maps* can add information about how anisotropy varies across structures. Hue-balls can take the role of assigning color, but the underlying principle of deflection can also be used to assign opacity with a *deflection opacity map*. Lit-tensors are one method of shading, but a more simplistic method based on the *gradient of opacity* is also described, as well as mixtures of the two shading approaches. For simplicity, we have taken a ray casting approach to volume rendering tensor fields. After interpolating the tensor information at each sample point along a ray cast through the volume, we apply any of the various methods described in this paper to determinine opacity, color, and shading. We then composite the results to determine the pixel color.

In addition, we describe a new method for visualizing diffusion tensor fields. By simulating a reaction-diffusion process between two interacting chemicals, on a domain effectively warped by the underlying tensor data, we are able to produce a volumetric solid texture that follows the structure of the diffusion tensor field. The texture is composed of a large number of pseudo-ellipsoidal "spots", each of which reflect the magnitude and orientation of the eigenvectors in a local neighborhood. This texture can then be mapped onto the surfaces generated by a diffusion tensor volume rendering, or it can be inspected as a stand-alone visualization.

We finish with a brief discussion of an issue highly relevant to the task of volume rendering: interpolation. With scalar data, the matter of interpolation usually becomes a choice among the wide variety of available reconstruction kernels– trilinear, tricubic, windowed sinc, etc. In all cases it is obvious that the original sampled data values are the quantity to be interpolated. With diffusion tensor data, there is still the same choice of reconstruction kernel, but there is the independent question of which tensor-related quantity should be interpolated. One can interpolate the raw diffusion-weighted images acquired by the MRI scanner, the individual components of the tensor matrix calculated from them, or a quantity derived from the diffusion tensor, such as an eigenvector. We discuss and analyze three distinct interpolation schemes based on these different options.

2 Previous Work

Much previous work in tensor visualization has started by simplifying the data to a scalar or vector field, to which established visualization techniques can be applied. That is, the tensor is viewed only in terms of some salient scalar or vector characteristic. For example, *tensor field lines* allow one to see the patterns in the vector fields composed of the eigenvectors of a tensor matrix [12]. In the medical community there is much interest in visualizing two-dimensional slices of MR diffusion tensor data by colormapping the direction of the principal eigenvector (the eigenvector associated with the largest eigenvalue) [23, 15, 26, 8]. One tool for visualizing general (non-symmetric) second order tensor fields [5] proceeds by multiplying a fixed user-specified vector by the tensor field as sampled on some restricted domain (such as a plane) which acts as a probe to query specific regions of the field. Surface deformations or other vector visualization techniques are used to visualize the resultant vector field.

When the tensor visualization is not accomplished by showing only some of the information at all locations, it is often done by showing all the tensor information in a restricted subset of locations. A natural choice has been the ellipsoid representation of the tensor [25, 28, 16, 34], though rectangular prisms (with geometry determined by the eigensystem) also work very well [39]. A recent advance along these lines was inspired by artists who vary the characteristics of discrete brush strokes to convey information [18]. Through a carefully designed mapping from tensor attributes to brush stroke qualities, a two-dimensional MR diffusion tensor dataset can be rendered as an image with rich information content. Furthermore, the image can be understood at a range of scales, showing both the overall shape of the anisotropic regions, as well as the degree and direction of anisotropy at one particular location.

Another method of tensor visualization by explicit representation is hyperstreamlines [10, 11]. Streamlines are advected through a vector field of one of the eigenvectors, but instead of simply drawing a line to indicate the path, a surface is formed whose cross-section indicates the orientation of the other two eigenvectors and their associated eigenvalues. As with the ellipsoids, this type of representation must be unobstructed to be interpreted, so the density of hyperstreamlines in the volume must be low in order to avoid visual cluttering. Also, as with any scheme in which high-dimensional information is carefully packed into a single image, it can take some time to learn how to "read" these visualizations.

One could argue that density of visual information is what limits the number of hyperstreamlines that can go into a single visualization, or prevents a stack of ellipsoid-based two-dimensional visualizations from being readily composited to form a volume rendering.

[1] This is not the only unambiguous ellipsoidal representation. One could also represent the tensor with the unit sphere's pre-image, or, for a tensor M one could also use the set of points $\mathbf{x}$ such that $\mathbf{x}^{\mathrm{T}}\mathrm{M}\mathbf{x} = 1$, as is done by Strang [29].

However, volume rendering is precisely what is needed for our application. Three-dimensional rendering of tensor fields will almost certainly *require* the elision of some of the tensor information; the challenge is to choose which tensor characteristics to display and how to do so.

3 Methods

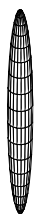
3.1 Barycentric Mapping

In volume rendering scalar data, the domain of the transfer function is nearly always the range of scalar data values. In volume rendering three-dimensional diffusion tensor data, however, it makes little sense to use the data values as the domain of the transfer function, since they live in a six-dimensional space: a 3×3 symmetric matrix has six degrees of freedom. From the tensor data, we can derive a simpler quantity living in a lower dimensional space, then specify transfer functions that map from this space to color and opacity. Therefore, the derived quantity has to vary significantly between the regions of interest and the regions that would only serve to obscure or cloud a visualization.

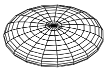
In the context of visualizing the shape of the white matter tracts in the human brain, such a quantity is anisotropy, since the fibers have anisotropy distinct from the isotropic gray matter that surrounds them. Assigning opacity to regions with high anisotropy while assigning low or no opacity to isotropic regions helps visualize the fiber tracts and ignore the gray matter on the exterior of the brain.

The literature provides various metrics for anisotropy based on the tensor matrix's three sorted eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ [33, 37, 25]. We have chosen to use the ones by Westin *et al.* due to the simple geometric motivation behind them. Metrics for three different kinds of anisotropy are given:

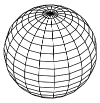
$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \tag{1}$$



$$c_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2}$$



$$c_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{3}$$

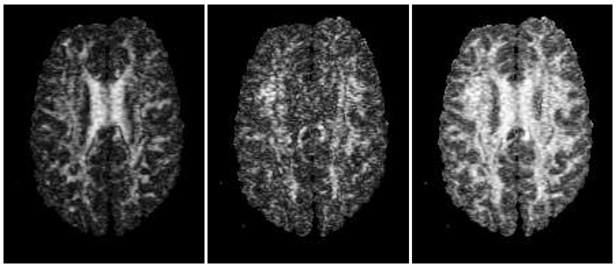


It can be shown that all the metrics fall in the range $[0, 1]$, and that they sum to unity: $c_l + c_p + c_s = 1$. The ellipsoids drawn next to the anisotropy metrics indicate the shape of diffusion tensor for which that metric will be high; it will be near zero for the other two shapes. Where only c_l is high, the tensor field is said to be *linearly anisotropic*; where only c_p is high, the tensor field is *planarly anisotropic*. The last metric, c_s is actually for *isotropy*; $c_s = 1$ only when all the eigenvalues are equal. Therefore, a single anisotropy metric called the "anisotropy index" is defined as:

$$c_a = 1 - c_s = c_l + c_p = \frac{\lambda_1 + \lambda_2 - 2\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{4}$$

To see how the anisotropy can vary in measured data, Figure 2 shows the metrics c_l, c_p, and c_a evaluated over the same dataset slice seen in previous figures, with brighter areas indicating higher anisotropy.

In light of the normalization built into c_l, c_p, and c_s, we propose the use of barycentric coordinates to depict the space of possible anisotropies, as shown in Figure 3. For every point in the triangle, there is a corresponding ellipsoid for which the anisotropy measures



(a) c_l (b) c_p (c) c_a

Figure 2: Different anisotropy metrics on a slice.

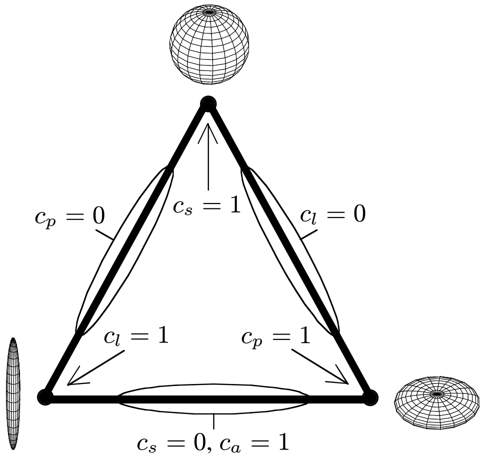


Figure 3: Barycentric space of anisotropies.

(c_l, c_p, and c_s) evaluate to the point's barycentric coordinates. In the figure, the three ellipsoids accompanying the corners of the triangle are representative of the ellipsoids that correspond to those corners. At each vertex of the triangle, one of the anisotropy measures is one, while the two others are both zero. Along the sides of the triangle, one of the anisotropy measures is zero, and the other two measures sum to one.

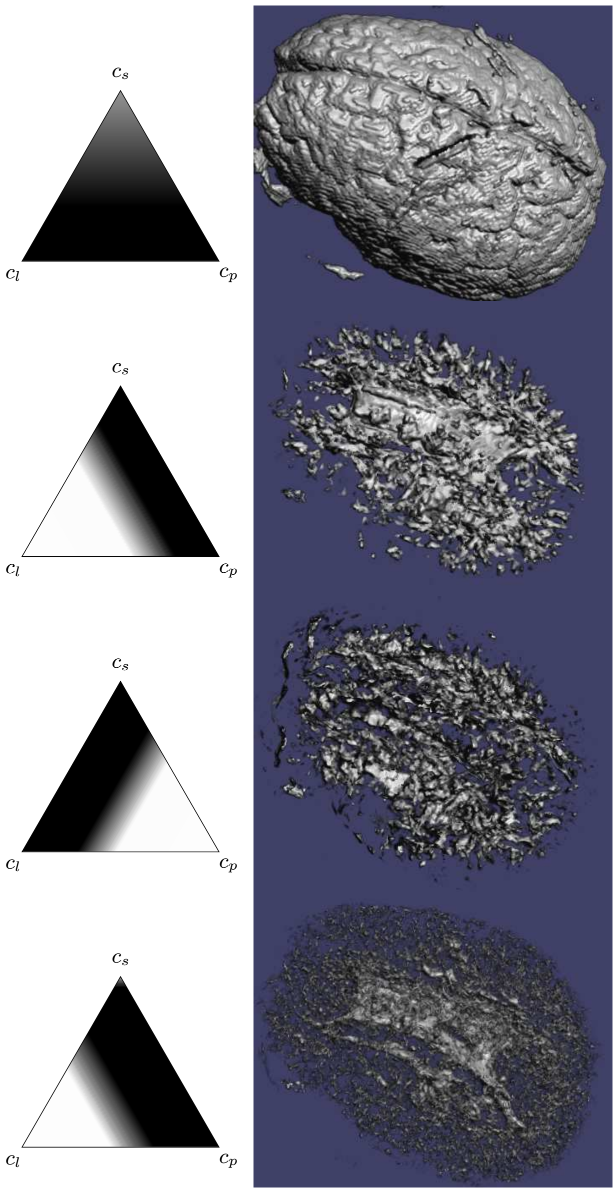


Figure 4: Examples of barycentric opacity maps and resulting volumes.

Barycentric opacity functions use this barycentric space of anisotropy as their domain, assigning an opacity between 0.0 and 1.0 to each location inside the triangle (or to each entry in a two-dimensional lookup table that represents the triangle). During rendering, a given sample point's opacity is found by looking up the opacity at the location determined by the anisotropy of the diffusion tensor at that point.

Figure 4 demonstrates some barycentric opacity maps. Each opacity map is depicted by gray-scale representation: brighter regions in the triangle correspond to higher opacity assignment. For the purposes of this figure, the effect of the opacity map is demonstrated by applying the map to the the tensor dataset, resulting in a *scalar* volume of opacity values. This new scalar volume is visualized with a *linear* opacity function, and shaded according to the gradient of opacity values. One can see that, analogous to Figure 2, appropriately chosen opacity functions allow one to see the form of structures in the dataset that have one predominant type of anisotropy.

Because of its expressive power, the barycentric space also makes sense as the domain of the color function, which assigns color to each sample point in the volume rendering according to its anisotropy. Most importantly, different kinds of anisotropy receiving equal opacity can be disambiguated by assigning different colors. Also, to the extent that various classes of white-matter tissue are found to have a characteristic anisotropy throughout the volume, they can be color-coded with an appropriate barycentric color map. Volume renderings made with both barycentric opacity and color maps allow an extra dimension of information about the diffusion tensor to be represented in the volume rendering. Figure 5 shows two examples of these.

3.2 Lit-Tensors

Streamlines used in vector visualization are sometimes hard to interpret because they lack the shading cues that we are accustomed to seeing on surfaces. However, illuminated streamlines ("lit-lines") have shading and highlights that give information about their direction and curvature, creating the appearance of shiny filaments [40]. In the case of diffusion tensor visualization, we have made tensors opaque and colored based on their anisotropy, but we lacked a means of shading tensors in the presence of a virtual light source. We designed a shading technique termed *lit-tensors* which can indicate the type and orientation of anisotropy, by following certain constraints:

1. In regions of complete linear anisotropy, the lighting model should be identical to that of illuminated streamlines. Complete linear anisotropy means movement by diffusion is constrained to one dimension, so it is sensible for the lighting model to degenerate to one already developed for vector fields.

2. In regions of complete planar anisotropy, the lighting model should be the same as with traditional surface rendering. The obvious choice for the "surface normal" for a planar anisotropic tensor is the third eigenvector, perpendicular to the plane formed by the span of the first two eigenvectors (associated with the largest two eigenvalues).

3. There has to be a smooth interpolation between these two extremes. Since tensor data can exhibit a wide variety of anisotropies, allowing small variations in anisotropy to lead to large changes in shading will probably create a more confusing image.

This can be seen as a problem of how to interpolate illumination between different codimensions. The codimension of the diffusion tensor's representative ellipsoid is two in the linear anisotropy case, and one with planar anisotropy. Previous work [1] has rigorously developed illumination methods for general manifold dimension and codimension, but did not cover cases part-way between different codimensions. Unlike that work, no claim to physical accuracy or plausibility is made for the model presented here; it is just one simple way of satisfying the constraints above.

3.2.1 Lit-Tensor Formulation

We take as our starting point the Blinn-Phong lighting model [4]:

$$\begin{aligned} I &= I_{\mathrm{ambient}} + I_{\mathrm{diffuse}} + I_{\mathrm{specular}} \\ &= k_{\mathrm{a}} A_\lambda O_\lambda + I_\lambda (k_{\mathrm{d}} O_\lambda \mathbf{L} \cdot \mathbf{N} + k_{\mathrm{s}} (\mathbf{H} \cdot \mathbf{N})^n) \end{aligned} \qquad (5)$$

k_{a}, k_{d}, and k_{s} control the contributions of ambient, diffuse, and specular reflection to the final image. Following Foley *et al.* [13], we add the subscript λ to those variables which vary according to color. For example, there are separate values I_{r}, I_{g}, I_{b}, for the red, green, and blue components of the directional light source. The ambient light color is A_λ. Instead of representing the intrinsic object color with different k_{a} and k_{d} for red, green, and blue, we use O_λ for object color and keep k_{a} and k_{d} as separate controls. In our case, the intrinsic object color is determined by any of the methods described in this paper (barycentric maps, hue-balls, or reaction-diffusion textures). $\mathbf{L}$ is the vector pointing towards the directional light source, $\mathbf{V}$ points towards the eye, and $\mathbf{N}$ is the surface normal. Note that instead of using $(\mathbf{V} \cdot \mathbf{R})^n$ for the specular component, where $\mathbf{R}$ is the reflection of $\mathbf{L}$ across $\mathbf{N}$, we are using the "half-way" vector $\mathbf{H}$ in $(\mathbf{H} \cdot \mathbf{N})^n$. $\mathbf{H}$ is the normalized average of $\mathbf{L}$ and $\mathbf{V}$, and n is the shininess exponent.

Because a streamline is one-dimensional, at any given point along it there is an infinite set of normals, all perpendicular to the tangent direction $\mathbf{T}$, radiating outwards in a circle. If naively using Equation 5 to illuminate a streamline, one must find the normal which is in the plane spanned by $\mathbf{L}$ and $\mathbf{T}$ to evaluate $\mathbf{L} \cdot \mathbf{N}$. Similarly, another specific normal must be found to evaluate $\mathbf{H} \cdot \mathbf{N}$. The insight which makes lit-lines simple is that one does not need to actually find a specific normal in order to evaluate a dot product with it. With the Pythagorean theorem, the dot product with $\mathbf{N}$ can be expressed in terms of the tangent $\mathbf{T}$:

$$\mathbf{U} \cdot \mathbf{N} = \sqrt{1 - (\mathbf{U} \cdot \mathbf{T})^2} \qquad (6)$$

where $\mathbf{U}$ is either $\mathbf{L}$ or $\mathbf{H}$, for the diffuse and specular terms, respectively.

The relevant property of Equation 6 is that the lighting calculation depends on a tangent vector $\mathbf{T}$ that gives the object's direction, instead of its surface normal $\mathbf{N}$. The direction and orientation of a diffusion tensor is determined by not one, but two vectors: the first and second eigenvectors[2]. Both of these could be interpreted as tangents, but their relative importance is determined by the magnitudes of the corresponding eigenvalues. To control the relative importance of the first two eigenvectors in determining the tensor's orientation, we introduce a parameter c_θ that characterizes *anisotropy type*. Assuming that the eigenvalues are ordered $\lambda_1 \geq \lambda_2 \geq \lambda_3$, we define

$$c_\theta = \frac{\pi}{2} \frac{c_p}{c_a} = \frac{\pi(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 - 2\lambda_3} \qquad (7)$$

As anisotropy varies from completely linear ($c_l = 1$; $c_p = 0$) to completely planar ($c_l = 0$; $c_p = 1$), c_θ varies from 0 to $\frac{\pi}{2}$. The role of c_θ is to control how much the second eigenvector contributes to the lighting of the diffusion tensor. In the linear case, only the first eigenvector determines the tensor orientation, and in the planar case, both the first and second eigenvectors matter equally.

The expression to be used in lieu of dot products with $\mathbf{N}$ is:

$$\text{“}\mathbf{U} \cdot \mathbf{N}\text{”} = \sqrt{1 - (\mathbf{U} \cdot \mathbf{e}_1)^2 - (\mathbf{U} \cdot \mathbf{e}_2 \sin(c_\theta))^2} \qquad (8)$$

In the case of linear anisotropy, $\sin(c_\theta) = \sin(0) = 0$, so the contribution from $\mathbf{e}_2$ vanishes, and the expression reduces to the

[2] Because the eigenvectors always form an orthogonal basis, and because we are adopting two-sided lighting, the third eigenvector does not contribute any additional information.

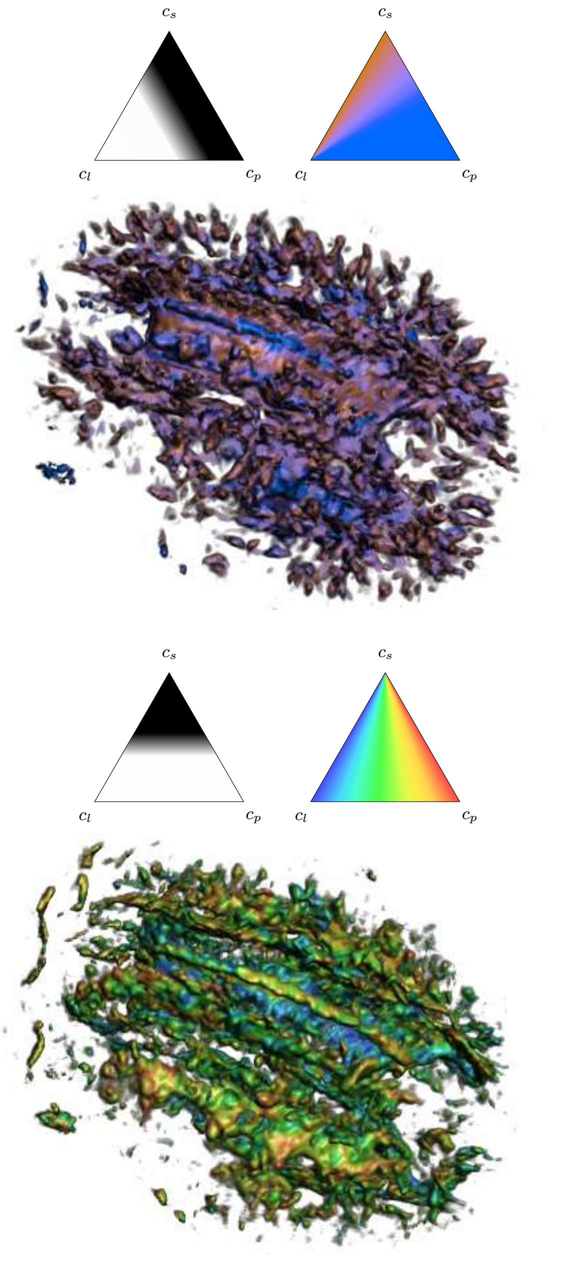


Figure 5: Examples of barycentric color maps and resulting renderings.

formula for lit-lines (Equation 6), with the principal eigenvector $\mathbf{e}_1$ taking the role of the tangent $\mathbf{T}$. This is appropriate, since in linear anisotropy, the principal eigenvector points in the direction of movement, as does a streamline's tangent vector.

In planar anisotropy, $\sin(c_\theta) = \sin(\frac{\pi}{2}) = 1$, and the contributions of the two dot products are equal. This means that for any other vector $\mathbf{W}$ such that

$$(\mathbf{W} \cdot \mathbf{e}_1)^2 + (\mathbf{W} \cdot \mathbf{e}_2)^2 = (\mathbf{U} \cdot \mathbf{e}_1)^2 + (\mathbf{U} \cdot \mathbf{e}_2)^2 \qquad (9)$$

Equation 8 will have the same value. Therefore, in planar anisotropy the lighting model is rotationally symmetric around $\mathbf{e}_3$. Rotational symmetry in this case is actually an important feature of the lighting model. In planar anisotropy, the diffusion tensor ellipsoid degenerates to a disc, and *any* vector in the plane spanned by the disc is an eigenvector. Because of this numerical instability, the calculated directions of the first and second eigenvectors will be essentially random. The illumination should not be sensitive to this arbitrary orientation, and should only be a function of the third eigenvector. In fact, one can use the Pythagorean theorem to show that if $c_\theta = \frac{\pi}{2}$, Equation 8 gives an exact formula for $\mathbf{U} \cdot \mathbf{e}_3$. Interpreting both $\mathbf{e}_1$ and $\mathbf{e}_2$ as surface tangents, then the surface normal $\mathbf{N}$ is aligned along $\mathbf{e}_3$. Therefore the model contains standard surface shading as a special case.

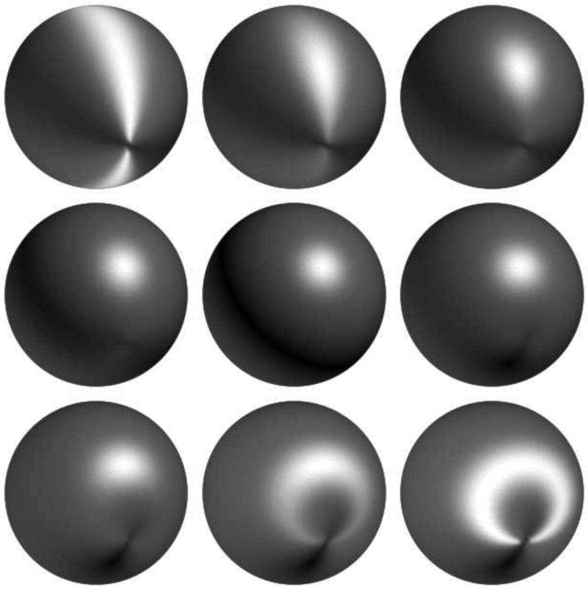


Figure 6: Sequence of volumes of differing anisotropy, rendered with lit-tensors. Anisotropy varies gradually between the nine volumes, going in scanline order.

To demonstrate lit-tensors, Figure 6 shows nine different synthetic diffusion tensor datasets that were direct volume rendered with a fixed viewpoint and light. The anisotropy index c_a of the sphere is also constant in every case, but c_θ is changing. The dataset in the upper left has complete linear anisotropy in a concentric circular pattern (along lines of latitude). The dataset in the middle has complete planar anisotropy (and hence looks just like a standard surface rendering). The dataset in the lower left has complete linear anisotropy along lines of longitude, going from pole to pole. The images provide a convincing sense of surface anisotropy, which is not a typical trait in direct volume renderings.

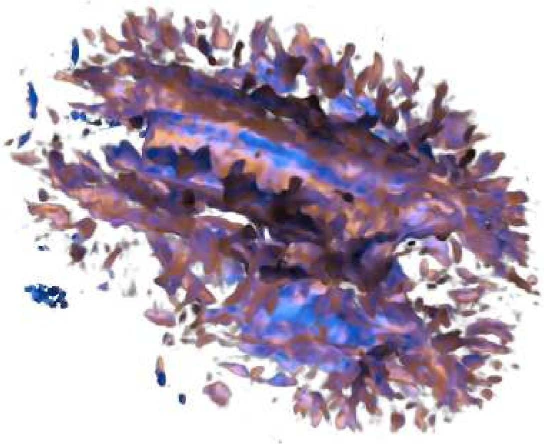


Figure 7: Lit-tensor model shading whole brain, with same opacity and light direction as the first image in Figure 5.

3.2.2 Lit-Tensors mixed with Opacity Gradient Shading

The spheres shown in Figure 6 have well-behaved anisotropy in the following sense: the changes in the orientation of anisotropy are directly correlated to the changes in the orientation of the sphere's surface normal. Experience has shown that measured diffusion tensor data is generally not so well-behaved, so that still images created using lit-tensors tend to be confusing[3]. The underlying problem is that lit-tensors were designed to indicate anisotropy type and direction, not the shape of the structure made opaque by the opacity function. On a complex structure with significant self occlusion, the lack of surface shape cues can lead to a rather ambiguous, watercolor effect, as seen in Figure 7.

Our current solution to this problem is to perform a separate (and significantly simpler) shading calculation, using standard Phong shading with the normalized gradient of opacity serving as the surface normal. This is accomplished by a two-step pre-process: the opacity at each data point is determined, and then the gradient of the opacity field is calculated. The normalized negative gradient of opacity is stored at each sample point. During volume rendering, these directions are interpolated to determine a surface normal at every point along the ray. The interpolated surface normal is used in the Phong shading equation. This results in an image in which shading is entirely determined by the opacity assignment and the shape of the opaque structures selected by it.

However, it is also possible to arbitrarily mix the results of lit-tensor shading and opacity gradient shading, as shown in Figure 8. Both shading calculations are performed, and then the results are mixed on a per-voxel basis by a user-defined parameter. This sort of mixing is quite different than varying the anisotropy type as was done in Figure 6. Instead of one specular highlight changing shape gradually, there are two different specular highlights which cross-blend.

The range of possibilities illustrated by Figure 8 demonstrates an important difference between scalar and tensor volume rendering. In scalar volume rendering, opacity is nearly always determined as a function of the (scalar) data value, hence the opacity gradient is always aligned with the gradient of original data value. This means that the data value gradient can be computed only once per dataset and used to shade the output of any opacity function. Unfortunately, such a pre-process is not possible with tensor data under

[3]Animations that vary viewpoint location can disambiguate surface shape while enhancing the effect of lit-tensors by showing motion of the specular highlights.

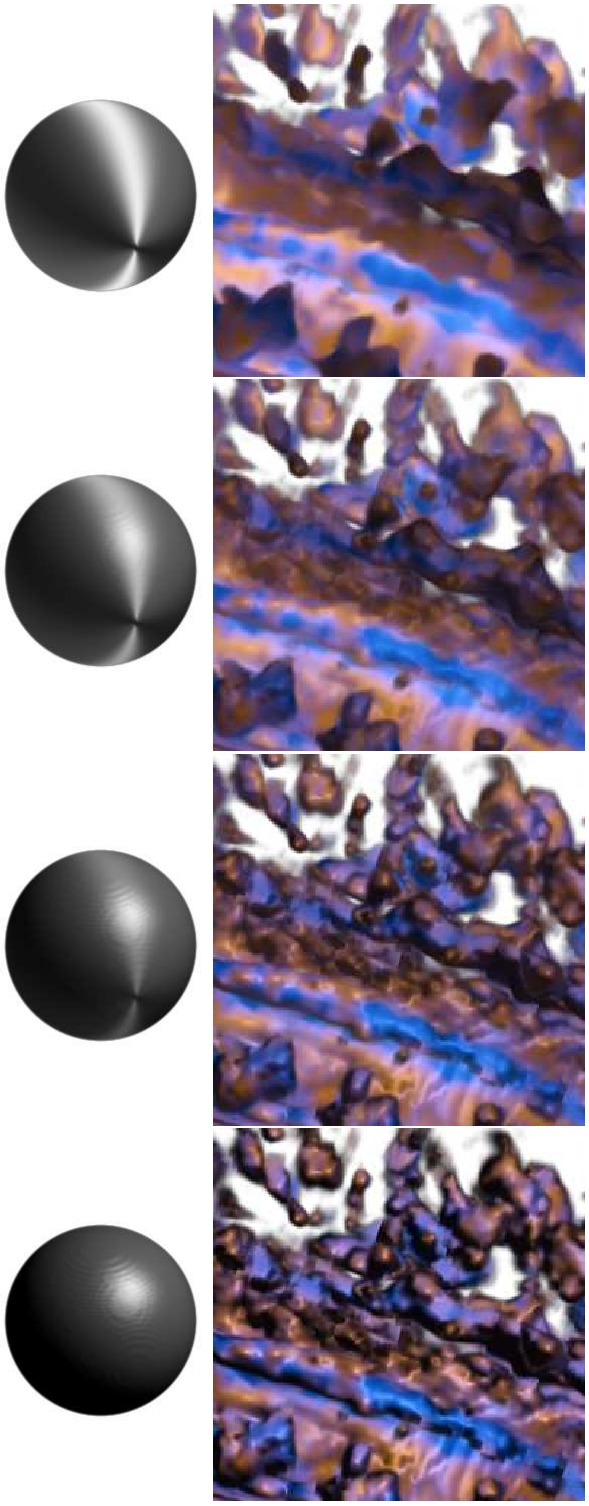


Figure 8: Mixing between lit-tensor and opacity gradient shading, for a synthetic sphere (left), and a portion of brain data (right). Going from top to bottom, the contribution of lit-tensors is 1.00, 0.66, 0.33, and 0.00.

barycentric opacity maps, as the domain of the opacity function is a multi-dimensional space which varies non-linearly with the tensor matrix component values. On the other hand, given the overall computational expense of tensor volume rendering, we have found the cost of having to compute the opacity gradient once per opacity function to be acceptable.

3.3 Hue-balls and Deflection Mapping

3.3.1 Deflection caused by Tensors

The idea underlying hue-balls is that the mapping from tensors to color should not first reduce the tensor to a vector, such as one of its eigenvectors. The intent is to maximize the continuity of the mapping across the range of possible anisotropies. Color determined by the direction of the principal eigenvector, for instance, is discontinuous in regions of low anisotropy, and even high planar anisotropy. Hue-balls color tensors according to their action as a linear operator. At all locations in the tensor field, a single user-specified input vector is multiplied by the diffusion tensor matrix to create an output vector. The tensor is assigned color by using the direction of the output vector as the lookup into a smoothly varying spherical colormap. We use the term *hue-ball* to describe a spherical colormap used in this way. Throughout a region of high spatial coherence in the tensor field, multiplying by the tensor will tend to give the same result, and the assigned color will be nearly uniform. Discerning coherent structures in the tensor field becomes a task of visually detecting color coherence in the rendered image. A closely analogous approach in vector visualization uses a two dimensional hue-saturation colormap on the sphere to visualize perturbation velocity in an application of direct volume rendering to computational fluid dynamics [35].

The properties of the tensor's matrix representation are important for understanding how the hue-ball functions. It is useful to consider the input vector in the basis formed by the eigenvectors. Given a tensor matrix M with unit-length eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$, an input vector $\mathbf{v}$ can be expressed as

$$\begin{aligned} \mathbf{v} &= (\mathbf{v}\cdot\mathbf{e}_1)\,\mathbf{e}_1 + (\mathbf{v}\cdot\mathbf{e}_2)\,\mathbf{e}_2 + (\mathbf{v}\cdot\mathbf{e}_3)\,\mathbf{e}_3 \\ &= (\mathbf{v}\cdot\mathbf{e}_1, \mathbf{v}\cdot\mathbf{e}_2, \mathbf{v}\cdot\mathbf{e}_3) \end{aligned} \tag{10}$$

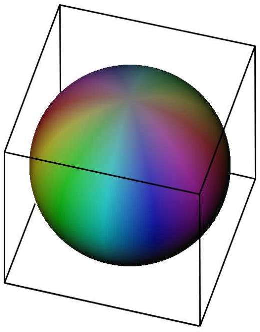
Then the output vector $\mathrm{M}\mathbf{v}$ can be expressed as

$$\begin{aligned} \mathrm{M}\mathbf{v} &= \mathrm{M}(\mathbf{v}\cdot\mathbf{e}_1)\,\mathbf{e}_1 + \mathrm{M}(\mathbf{v}\cdot\mathbf{e}_2)\,\mathbf{e}_2 + \mathrm{M}(\mathbf{v}\cdot\mathbf{e}_3)\,\mathbf{e}_3 \\ &= \lambda_1(\mathbf{v}\cdot\mathbf{e}_1)\,\mathbf{e}_1 + \lambda_2(\mathbf{v}\cdot\mathbf{e}_2)\,\mathbf{e}_2 + \lambda_3(\mathbf{v}\cdot\mathbf{e}_3)\,\mathbf{e}_3 \\ &= (\lambda_1\mathbf{v}\cdot\mathbf{e}_1, \lambda_2\mathbf{v}\cdot\mathbf{e}_2, \lambda_3\mathbf{v}\cdot\mathbf{e}_3) \end{aligned} \tag{11}$$

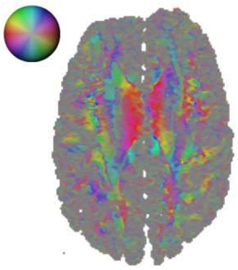
where λ_i is the eigenvalue corresponding to eigenvector $\mathbf{e}_i$. The coordinates of the output vector in the eigenvector basis are the input vector's coordinates, scaled by the corresponding eigenvalues. We term the change in direction between the input and output vectors the *deflection* caused by the tensor.

Equation 11 indicates that the vector is always deflected towards the principal eigenvector, since the coordinate of the input vector in the principal eigenvector direction will by definition grow proportionally larger than the components along the other eigenvectors. There is also a relationship between the amount of deflection and the tensor's anisotropy. Because the anisotropy of a tensor is in general related to the disparity among its three eigenvalues, multiplying a vector by a tensor with high anisotropy will cause a greater relative change among its coordinates, and hence a greater deflection. However, since the diffusion tensor matrix has non-negative eigenvalues, multiplying by it cannot change the sign of any of the vector's coordinates. Both the input and output vectors will be in the same octant of the eigenvector basis, so the angle between input and output vectors cannot exceed 90 degrees.

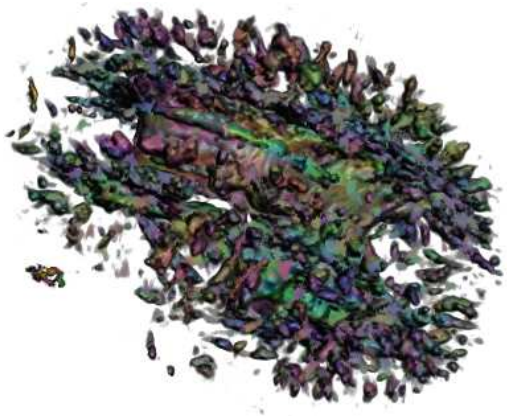
3.3.2 Hue-ball Color Mapping



(a) Hue-ball



(b) Slice mapped by hue-ball



(c) Volume colored by hue-ball

Figure 9: Hue-ball acting on one slice of a dataset, and as used in volume rendering.

The free parameters in the hue-ball method of assigning colors to tensors are the color assignment on the sphere, and the input vector to use for multiplication with the diffusion tensor matrix. For the sake of simplicity we have used only the hue-ball mapping shown in Figure 9(a). The sphere has a band of saturated colors around its equator, with saturation decreasing to the top and bottom poles, which are a medium gray. All the colors have the same “lightness” in the HSL color space [13], since for the sake unambiguous visualization, it is less confusing if the hue-ball varies only in color, letting the shading model control intensity[4]. All the hues appear twice on the hue-ball so as to create 180 degree rotational symmetry.

To illustrate how the hue-ball colors measured tensor data, the same dataset slice which was shown in Figure 1 has been mapped by the HSL hue-ball described above and is shown in Figure 9(b). Some previous techniques for colormapping diffusion tensor data assign color based on the direction of the principle eigenvector, and then modulate the color by some scalar anisotropy measure, so that isotropic regions are suppressed. Using an appropriately chosen hue-ball mapping, with the input vector pointing to a neutral color, this happens automatically, so no anisotropy calculation is needed. Finally, a full volume rendering is shown with hue-ball coloring in Figure 9(c).

[4]If one is seeking a truly constant luminance colormap, HSL colorspace is too simplistic.

3.3.3 Deflection Opacity Mapping

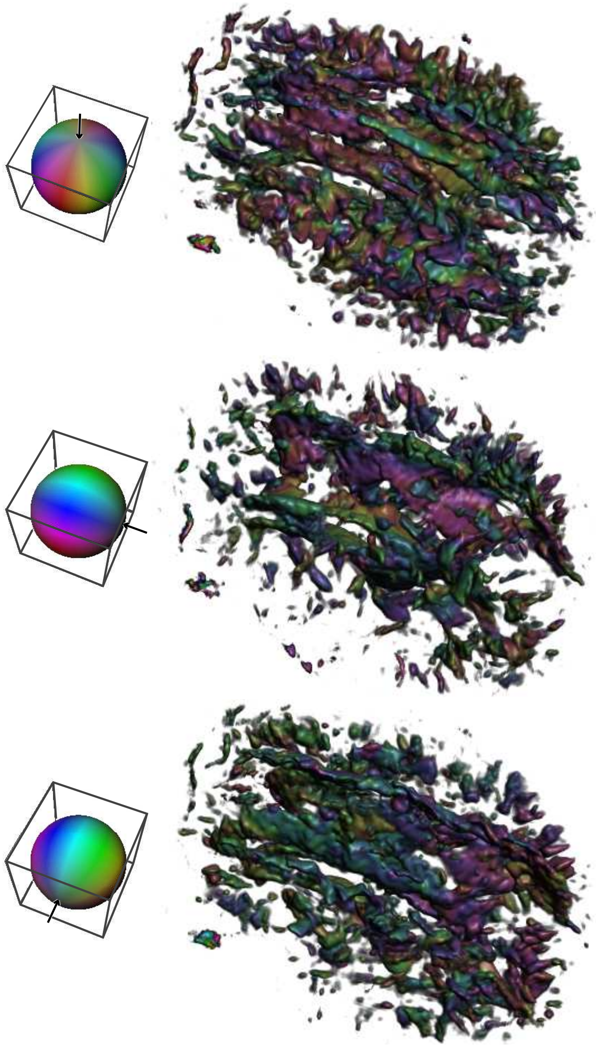


Figure 10: Assigning opacity based on deflection, with hue-ball coloring. Next to each image is a hue-ball image which indicates its orientation, and the direction of the input vector. Maximum opacity was assigned for deflection angles around six degrees and higher.

Based on the discussion in Section 3.3.1 about the relationship between anisotropy and deflection, we have also explored assigning opacity based on the amount of deflection. This opacity assignment is controlled by two user parameters– the input vector to use, and a simple (scalar) mapping from the amount of deflection to opacity. Assuming the regions of interest are anisotropic, the mapping should give no opacity if there was no deflection, and increase opacity with the amount of deflection. Figure 10 shows how the direction of the input vector emphasizes different features according to their anisotropy orientation. Because these images use the hue-ball for coloring, their rendering did not require solving for any eigensystems. They represent the quality of image possible with most numerically inexpensive methods.

3.4 Reaction-Diffusion Textures

3.4.1 Introduction

Our goal in this section is to use reaction-diffusion textures as a means of visualizing three-dimensional diffusion tensor data. We start by describing a simple model of reaction-diffusion texture that works in two and three dimensions, and then discuss how to modify its calculation to make the texture reflect measured diffusion tensor data. Then, we describe how to render the three-dimensional textures as a stand-alone method for diffusion tensor visualization, as well as how to integrate them into the rendering methods described in previous sections. The use of reaction-diffusion textures for this purpose is closely related to previous work which tuned spot noise to portray local characteristics of scalar and vector fields [36], or which used three-dimensional line integral convolution of spot noise to perform flow visualization of volumetric vector data [14, 27, 6]. In our case, instead of tuning noise, we are tuning what emerges as a well-organized pattern. The pattern closely corresponds to a field of ellipsoids, the traditional means of diffusion tensor visualization.

The origin of reaction-diffusion textures is a paper by Alan Turing [30] that sought to mathematically model the formation of the immense variety of growth and pigmentation patterns found in the animal kingdom. Turing's paper describes a pair of non-linear partial differential equations modeling the reactions between two chemicals (called "morphogens"), which diffuse at different rates and interact according to certain rules of activation and inhibition. Reaction-diffusion became a popular method in computer graphics for generating textures with the development of methods for normalizing the density of texture on parameterized surface and polygonal models, and for generating a rich variety of texture patterns [31, 38]. Reaction-diffusion textures have been applied to a wide variety of other contexts as well [22].

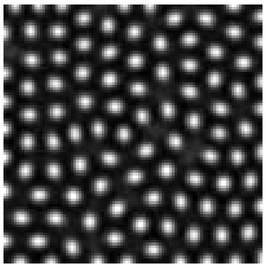
The reaction-diffusion equations Turing proposed are quite simple. The concentrations of the two morphogens are represented by a and b, and the differential equations tell how to increment a and b as function of their reaction and diffusion. The initial condition at $t = 0$ is that $a = b = 4$ everywhere.

$$\begin{aligned} \frac{\partial a}{\partial t} &= s(k(16 - ab) + d_a \nabla^2 a) \\ \frac{\partial b}{\partial t} &= s(k(ab - b - 12 + \beta) + d_b \nabla^2 b) \end{aligned} \tag{12}$$

The scaling factor k controls the size of the reaction terms of the equations relative to the diffusion terms, determining the size of the emergent patterns. Larger values produce patterns that stabilize more quickly, but have smaller characteristic size. The diffusion rates d_a and d_b control how fast the two chemicals can spread in the medium. The overall speed of the pattern's emergence is controlled by s: higher values make the system run faster, but values too large can lead the system into divergent instability. The remain ingredient is β, a pattern of uniformly distributed random values in a small interval centered around 0. It is this pattern that pushes the system away from the unstable equilibrium of the initial conditions, and towards the final texture. In practice, these reaction-diffusion systems are simulated on a regular two-dimensional discrete grid, in which case the Laplacian ∇^2 of a chemical c can be measured with a discrete convolution mask L:

$$\nabla^2 c = L * c = \begin{bmatrix} 0 & 1.0 & 0 \\ 1.0 & -4.0 & 1.0 \\ 0 & 1.0 & 0 \end{bmatrix} * c \tag{13}$$

Figure 11 shows a simple two-dimensional texture that was generated using the equation above and the indicated parameter settings.



$$\begin{aligned} s &= 1.0 \\ k &= 0.0125 \\ d_a &= 0.125 \\ d_b &= 0.03125 \\ \beta &\in [-0.1, 0.1] \\ \#\ \text{iter.} &= 30000 \end{aligned}$$

Figure 11: Amount of chemical a in solution to Equation 12 on a 100×100 grid with indicated parameter settings; black is about 3.0, white is about 7.5.

An important property of Equation 12 is that they are general with respect to dimension. Specifically, they work equally well to create a volumetric *solid texture* [24]. Such a texture can be calculated once and then mapped onto any surface placed within the volume. Or, in the context of volume rendering, the texture volume value can modulate the material color calculated at each sample point. The only implementation change is that the Laplacian is now measured with a three-dimensional convolution mask, shown in Figure 12.

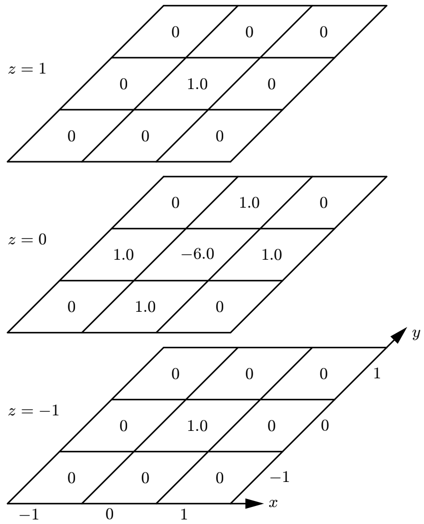


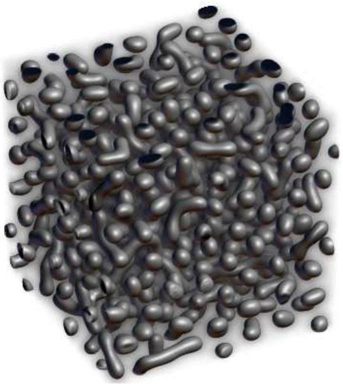
Figure 12: Three-dimensional convolution mask L for measuring Laplacian $\nabla^2 c$.

The result of running the three-dimensional reaction-diffusion equation is shown in Figure 13. Note that none of the simulation parameters have changed, and the resulting texture is a close three-dimensional analog to what was seen in Figure 11 (though some of the spots have joined together into more distended blobs).

3.4.2 Tuning the Texture with Tensor Data

Suppose we have a single chemical c in an isotropic medium with diffusivity d (a scalar). The rate of change in c due to non-steady-state diffusion is governed by Fick's second law [21]:

$$\begin{aligned} \frac{\partial c}{\partial t} &= \nabla \cdot (d \nabla c) \\ &= d \nabla^2 c \end{aligned} \tag{14}$$



$$
\begin{aligned}
s &= 1.0 \\
k &= 0.0125 \\
d_a &= 0.125 \\
d_b &= 0.03125 \\
\beta &\in [-0.1, 0.1] \\
\text{\# iter.} &= 30000
\end{aligned}
$$

Figure 13: Amount of chemical a in solution to Equation 12 on a $100 \times 100 \times 100$ grid with indicated parameter settings, volume rendered with colored lights and depth cueing.

Equation 14 says that the amount of chemical c changes according to the divergence of its concentration gradient, and the diffusivity d. Since the diffusion is isotropic, the scalar d can be brought outside the divergence, and we get the $\nabla^2 c$ factor which appears in Equation 12. In an anisotropic medium, however, d is replaced by the diffusion tensor matrix D, which transforms how the gradient of concentration ∇c determines flux. D is the matrix calculated from the measured diffusion-weighted MRI images; it comprises the diffusion tensor field we wish to visualize by volume rendering. Generalizing Equation 14 by replacing d with D, we derive a convolution mask M that makes calculating the reaction-diffusion simulation simple.

$$
\begin{aligned}
\frac{\partial c}{\partial t} &= \nabla \cdot (D \nabla c) \\
&= \nabla \cdot \left(\begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{bmatrix} \begin{bmatrix} \frac{\partial c}{\partial x} \\ \frac{\partial c}{\partial y} \\ \frac{\partial c}{\partial z} \end{bmatrix} \right) \\
&= \nabla \cdot \begin{pmatrix} D_{xx}\frac{\partial c}{\partial x} + D_{xy}\frac{\partial c}{\partial y} + D_{xz}\frac{\partial c}{\partial z} \\ D_{xy}\frac{\partial c}{\partial x} + D_{yy}\frac{\partial c}{\partial y} + D_{yz}\frac{\partial c}{\partial z} \\ D_{xz}\frac{\partial c}{\partial x} + D_{yz}\frac{\partial c}{\partial y} + D_{zz}\frac{\partial c}{\partial z} \end{pmatrix} \\
&= D_{xx}\frac{\partial^2 c}{\partial x^2} + D_{yy}\frac{\partial^2 c}{\partial y^2} + D_{zz}\frac{\partial^2 c}{\partial z^2} + \qquad (15) \\
&\quad 2D_{xy}\frac{\partial^2 c}{\partial x \partial y} + 2D_{xz}\frac{\partial^2 c}{\partial x \partial z} + 2D_{yz}\frac{\partial^2 c}{\partial y \partial z} \\
&= M * c
\end{aligned}
$$

Equation 15 should be viewed as the linear combination of six different derivatives of the concentration c, each of which can be implemented with a convolution mask defined on the same $3 \times 3 \times 3$ grid seen in Figure 12. Using a combination of first and second central differences to evaluate the derivatives, we arrive at the mask M shown in Figure 14. This mask is different at each location in the field because it is built directly from the components of the diffusion tensor matrix.

Simply substituting the position-independent L of Figure 12 in Equation 12 with the position-dependent M shown in Figure 14 creates a reaction-diffusion texture that visualizes anisotropic diffusion tensor data. Depending on the magnitudes of values in D, one may have to adjust the s and k parameters in Equation 12 in order to produce a simulation that converges on spots of an appropriate size. Instead of having approximately spherical spots, the spots will be stretched into ellipsoids that reflect the diffusion tensor data in their local neighborhood.

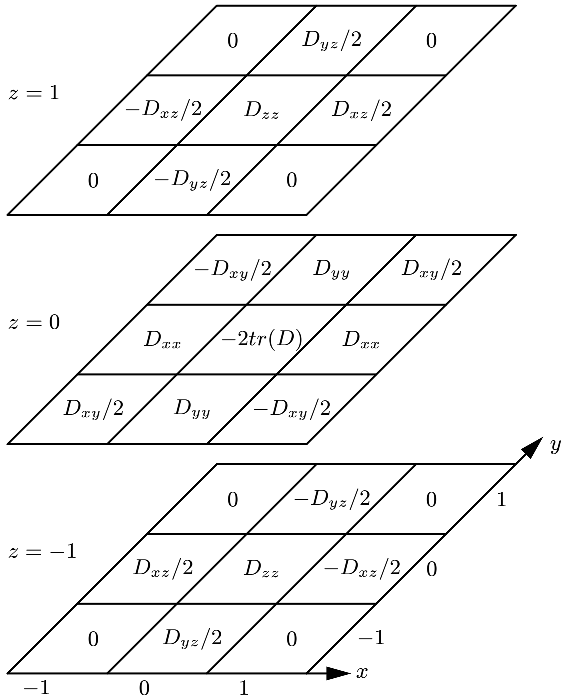
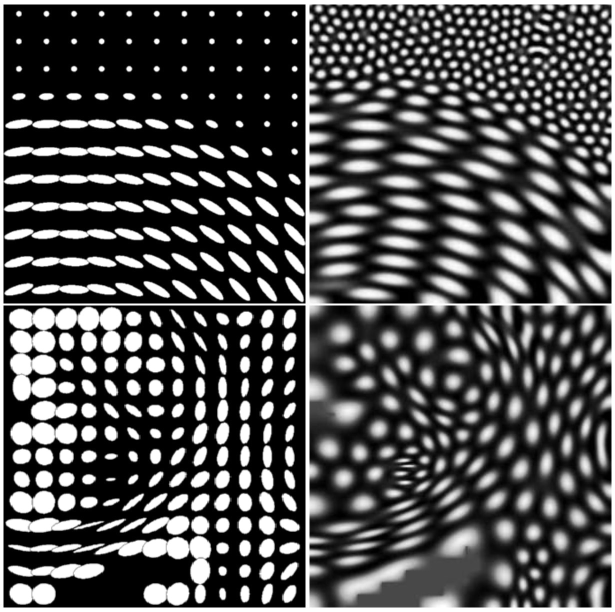


Figure 14: Three-dimensional convolution mask M for measuring $\nabla \cdot (D \nabla c)$ to determine amount of diffusion. $tr(D)$ is the trace of D, $D_{xx} + D_{yy} + D_{zz}$.

This simple substitution, however, is not exactly what we want. The random-walk nature of diffusion dictates that if we drop some ink into an anisotropic medium at a location where the diffusion tensor matrix has eigenvalues λ_1, λ_2, and λ_3, then the shape of the ink spot as it grows over time will approximate an ellipsoid whose axes are proportional to $\sqrt{\lambda_1}$, $\sqrt{\lambda_2}$, and $\sqrt{\lambda_3}$ [21]. That is, the ellipsoid discussed in previous sections– the image of the unit sphere under the tensor matrix– is not the same as the ellipsoid produced by running a diffusion simulation on an initial point of dye in a uniformly anisotropic medium. Because of this, we must run the reaction-diffusion simulation on a *different* tensor field, one in which the diffusion tensor matrix with eigenvalues λ_1, λ_2, λ_3 is replaced by a matrix with eigenvalues $\lambda_1^2, \lambda_2^2, \lambda_3^2$ (the eigenvectors are unchanged). This way the ellipsoids appearing in the results of the reaction-diffusion simulation will have the correct aspect ratio.

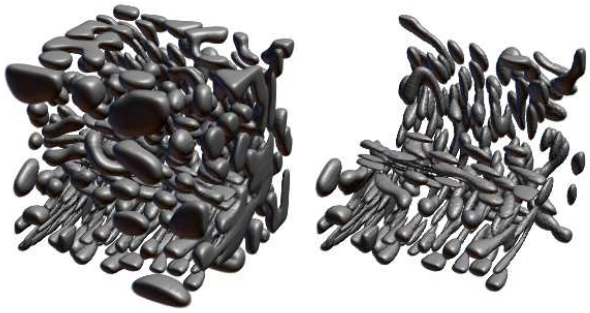
The above discussion also holds for the simpler case of two-dimensional diffusion tensor data. Instead of the whole convolution mask shown in Figure 14, we use its slice at $z = 0$. Figure 15 demonstrates how well two-dimensional reaction diffusion textures can represent diffusion tensor data. There are two major advantages to the use of reaction diffusion textures over a regular grid of ellipses. The first advantage is that the texture spots are packed together according to their size. Unlike with the ellipse arrays, there are no large gaps in which the tensor data is not visualized, nor do the texture spots ever overlap. While it is certainly the case that an algorithm could be developed for intelligently placing ellipses on the field, the benefit of using these textures is that the packing of texture spots arises automatically from the reaction-diffusion simulation. This benefit applies equally well to three-dimensional reaction-diffusion textures.

The second advantage of the reaction-diffusion textures is that the ellipses created in them are placed stochastically in a way that

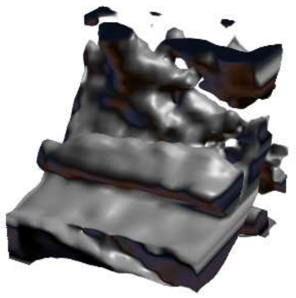


(a) Ellipse Array (b) Reaction-Diffusion

Figure 15: Comparison of regular array of ellipses (a) with anisotropic reaction-diffusion textures (b) for both a synthetic dataset (top row) and a portion of measured diffusion tensor data in the human brain (bottom row). Only 2×2 sub-matrices of the 3×3 diffusion tensor matrices were used to steer the textures in column (b).



(a) Unsegmented Texture (b) Segmented Texture



(c) Underlying Surface

Figure 16: Sample reaction-diffusion for three-dimensional tensor data. The texture is shown before (a) and after (b) segmenting. (c) shows a rendering using the same barycentric opacity map which was used for spot segmentation. The image in (b) shows both the structures of (c) and the structures' anisotropy orientation.

better allows the natural structure of the data to be seen. The reaction-diffusion texture facilitates visually tracking curved, twisting features more easily than on a regular grid. This issue is also addressed by research into how to position streamlines for the most effective flow visualization [32]. Because the ellipses are elongated, if they are placed in a way such that they approximately line up end to end (as happens in the circular synthetic dataset in Figure 15, column (a)), a non-existent linear structure spanning multiple ellipses is perceived, which accentuates the placement scheme of the ellipses, rather than the orientation of the underlying data.

3.4.3 Visualizing the Reaction-Diffusion Texture

Visualizing a three-dimensional texture generated from diffusion tensor data would be most useful if we had a way of removing the texture spots that occur in isotropic regions so that they do not obscure our view of the more interesting features. Fortunately, removing isotropic spots is a relatively easy operation to perform. Because of the uniform boundary and brightness of all the texture spots, it is trivial to choose a threshold for making the texture into a binary image in which all the spots are separated from each other. Next, we perform connected component analysis, defining adjacency by face neighbors. Since the spots are generally convex and thicker than a single voxel, each spot is correctly detected as one connected component. Then, we can determine the average value of a barycentric opacity map inside each spot, since we have defined the texture pattern as overlaying the tensor data. Finally, spots with average opacity below 0.5 are removed from the texture.

Figure 16 shows a small texture volume before and after segmenting out the isotropic spots, as well as structures that were selected by the barycentric opacity map. Renderings of segmented textures are useful visualizations in their own right, since as a whole, they can show the shape of the anisotropic structures in a flexible way, while the individual spots in the structure represent the local properties of the diffusion tensor field.

Figure 17(a) shows a segmented texture for half of the brain dataset shown in previous figures, and Figure 17(b) shows the (unsegmented) texture applied to a volume rendering of the tensor data. For this volume rendering, applying the texture was a simple matter of modulating the calculated material color at each point by the corresponding value in the texture volume; a more sophisticated technique like bump-mapping is not as straight-forward to accomplish. The benefit of texture-mapping the reaction-diffusion pattern onto the surface is that now the direction of anisotropy is indicated on what would be an otherwise isotropic surface.

4 Diffusion Tensor Interpolation

One important decision to make when rendering and manipulating datasets is the method of interpolation. In scalar volume rendering, the usual technique is to resample data values by trilinear interpolation, and then map them through the transfer function. One could also resample vectors by interpolating the vector data componentwise. It is less clear, however, how best to interpolate diffusion tensor data for direct volume rendering. This is because sometimes, as in the case of barycentric maps or lit-tensors, the underlying tensor matrix is not required for rendering, only some quantity derived from its eigensystem. In these cases, it would seem best to interpolate pre-computed eigenvalues or eigenvectors.

Other approaches are possible. It is extremely convenient and simple to interpolate the 3×3 diffusion tensor matrices componentwise; this is useful for cases where the eigensystem is not required (as with hue-balls and deflection opacity maps). On the other hand, it is worth noting that the tensor matrix itself is not actually measured directly by the MRI machine– it is computed from a set of measured "diffusion-weighted" images. Since we are accustomed to interpolating measured *scalar* CT or MRI data as part of volume rendering it, one could argue that interpolating the raw diffusion-weighted images is the most defensible choice – previous work has followed this path [7]. We believe the issue of tensor interpolation deserves more research, and we present our initial investigation into three different schemes for tensor interpolation.

Working at the lowest level of representation in the tensor data, we can interpolate the measured diffusion-weighted images which come off the MRI machine; we call this approach *channel interpolation*. Our tensor data was derived from a seven measured images ("channels"), notated A_i, $i = 0 \ldots 6$. Each channel corresponds to a pair of gradient encoding directions used during the scan acquisition. In channel interpolation, values from these seven images are stored at each sample point in the volume. These values are interpolated to produce image values at intermediate points in the volume, from which the diffusion tensor is calculated [3]. First, knowing the direction-independent diffusion weighting b (about 900 sec/mm^2) we calculate a set of log image values I_i, $i = 1 \ldots 6$:

$$I_i = \frac{ln(A_i) - ln(A_0)}{b} \tag{16}$$

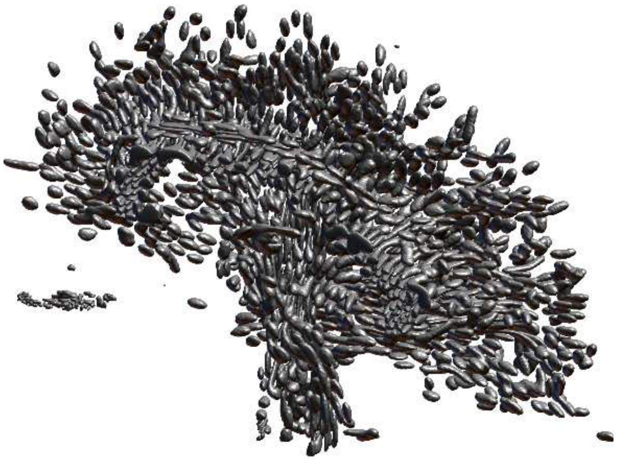
From these, the diffusion tensor D is calculated:

$$\begin{aligned} D_{xx} &= -I_1 - I_2 + I_3 + I_4 - I_5 - I_6 \\ D_{xy} &= -I_5 + I_6 \\ D_{xz} &= -I_1 + I_2 \\ D_{yy} &= +I_1 + I_2 - I_3 - I_4 - I_5 - I_6 \\ D_{yz} &= -I_3 + I_4 \\ D_{zz} &= -I_1 - I_2 - I_3 - I_4 + I_5 + I_6 \end{aligned} \tag{17}$$

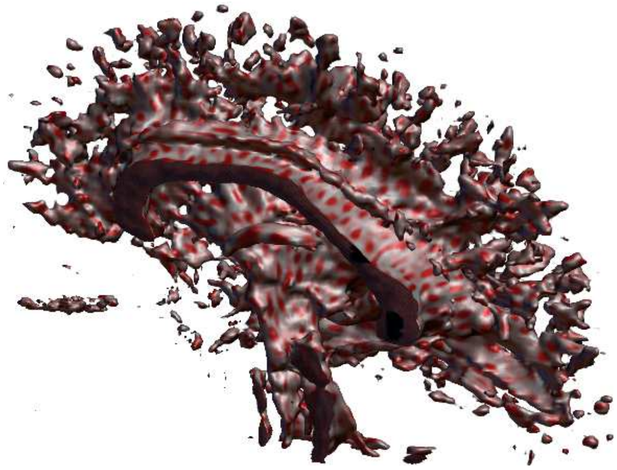
While this approach to interpolation has the benefit of working with the original measured data, there is considerable computational expense associated with evaluating the natural logs in the calculation of I_i. Avoiding this expense is achieved with *matrix interpolation*, wherein the tensor matrix D is calculated once per sample point, and then interpolated component-wise to produce a tensor at intermediate locations. Were the components of the tensor matrix simply linear combinations of the measured channels, then matrix interpolation would be equivalent to channel interpolation. However, they are different because of the non-linearities introduced by Equation 16.

As mentioned above, it is also conceivable that rather than interpolate channels or matrices, we interpolate the necessary information derived from them, such as eigenvalues or eigenvectors, a process we term *eigensystem interpolation*. In our work, this has immediate relevance for the barycentric map methods of assigning color and shading, and the lit-tensor shading model. This style of interpolation has the tremendous benefit of doing all the eigensystem calculations once as a pre-process, and storing the results at each dataset point.

Despite its obvious benefits, there is a subtle issue of *correspondence*, which complicates eigensystem interpolation. Given a set of three eigenvectors at one sample point $\mathbf{x}_1$, and another three eigenvectors at a second sample point $\mathbf{x}_2$, we do not immediately know the correspondence between the two sets of eigenvectors. However, it is necessary to define some correspondence between the eigensystems at the two sample points in order to perform eigenvector interpolation. Knowing the *continuous* tensor field from which the discrete dataset is sampled, we could possibly learn the "correct" correspondence. As the tensor field is sampled continuously on a path between $\mathbf{x}_1$ and $\mathbf{x}_2$, we could determine if (for example) the principal eigenvector at $\mathbf{x}_1$ smoothly becomes the principal eigenvector at $\mathbf{x}_2$. An analogous correspondence problem complicates



(a) Segmented Texture



(b) Texture mapped onto to surface rendering

Figure 17: Texture-mapping with reaction-diffusion texture.

eigenvalue interpolation. Efficiently determining the "correct" correspondence from the measured data, and characterizing those situations where the "correct" correspondence may be undefined (such as passing through a region of total or nearly total isotropy), is a topic of our current research.

We have evaluated one specific form of eigensystem interpolation that makes an assumption about the sampling density of the data. In *eigenvalue interpolation*, we interpolate eigenvalues based on the correspondence induced by sorting. Thus, the largest eigenvalues at two sample points are assumed to correspond, and are interpolated with standard scalar interpolation; likewise for the middle and smallest eigenvalues[5]. Because we learn the association between eigenvalues and eigenvectors as part of the eigensystem calculation, we note that the correspondence induced by eigenvalue sorting also determines a correspondence between eigenvectors (useful for lit-tensors), although such a correspondence could also be determined by a more complicated analysis of similarities in eigenvector orientation.

The "correct" eigenvalue correspondence is undefined if we know *a priori* there is a point (called a *critical point* [9]) between the two samples where two eigenvalues are equal (one of the anisotropy measures c_l or c_p are zero). This means the magnitudes of two eigenvalues crossed paths and changed sorting order. The eigenvectors associated with the (double) eigenvalue are also not uniquely defined at critical points. Fortunately, none of the methods presented in this paper crucially depend on non-unique eigenvectors or eigenvalues. For instance, the lit-tensor shading model (Section 3.2.1) becomes insensitive to eigenvector direction precisely when the direction is ill-defined, and the barycentric anisotropy coordinates vary continuously across critical points in continuous tensor data.

Thus, the sampling density assumption made in eigenvalue interpolation is that any important feature located near the critical point is also represented by neighboring sample points. More generally, the anisotropy properties that identify a feature need to be present at nearby sample points, instead of falling between them. We justify this assumption by noting that the features we are interested in visualizating tend to have characteristic size larger than the sampling grid resolution of the tensor data.

4.1 Evaluation

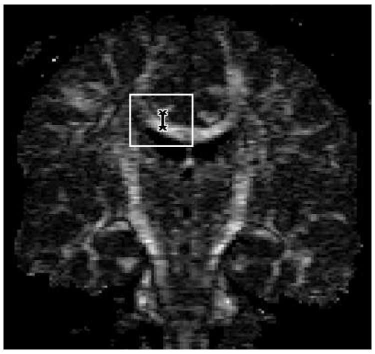


Figure 18: Interpolation path between cingulum bundle and corpus callosum. Anisotropy at 100 points along this path was evaluated using the three different interpolation methods.

One way to appreciate the difference between the three different interpolation methods described is to use them to interpolate along a path between two features of interest, and watch how a quantity derived from the tensor varies along the path. Because of the importance of barycentric anisotropy in our methods, we have plotted the variation in anisotropy along a path. The path we chose runs from the cingulum bundle down to the corpus callosum, as shown in Fig 18. The first point is in the cingulum bundle, an anatomic structure with linear anisotropy along the y-axis. The second point lies just seven voxels away in the corpus callosum, an anatomic structure with linear anisotropy along the x-axis. Between these two points is a region of low linear anisotropy, and somewhat higher planar anisotropy. The paths in barycentric anisotropy space shown in Figure 19 were traced by interpolating 100 uniformly spaced points in the tensor field on the line between the two points. Note that the channel and matrix interpolants follow very similar curved paths, while eigenvalue interpolation follows a straighter, simpler path.

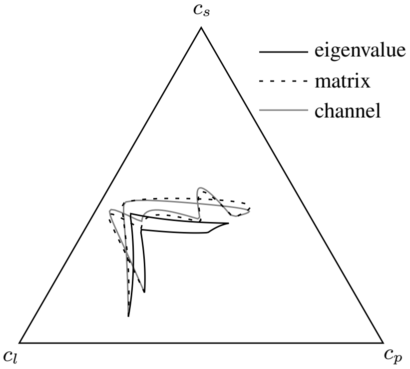


Figure 19: Interpolated paths between the cingulum bundle and the corpus callosum, as measured in barycentric anisotropy space.

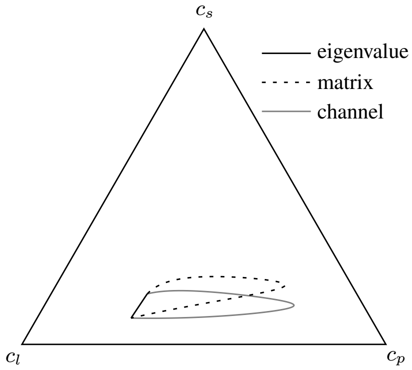


Figure 20: Interpolation between two non-neighboring voxels.

Figure 20 demonstrates the consequences of violating the sampling density assumption of eigenvalue interpolation. It shows the paths in barycentric anisotropy space for interpolating directly between the two endpoints in Figure 18, without sampling any data in the intervening field. While there is linear anisotropy at both endpoints, the direction of the principal eigenvector varies by 90 degrees, passing through a region in the dataset where c_l is near zero, so the eigenvalue correspondence induced by sorting is incorrect. The paths for channel and matrix interpolation methods do correctly veer towards low c_l anisotropy (albeit at a higher value of c_p), while the path for eigenvalue interpolation stays strictly within the space of high linear anisotropy.

From this evaluation, we conclude that channel and matrix interpolation tend to track each other closely, and that eigenvalue inter-

[5] This explanation implies the use of an interpolation kernel that overlaps only two samples at a time. Using a larger kernel would require determining the simultaneous correspondence over a larger set of eigensystems.

polation can work well as long as there is sufficient sampling density. Thus, we plan to incorporate better handling of eigenvalue interpolation in future versions of our rendering system. However, for the purposes of experimenting with the different rendering strategies presented here, some of which require the underlying tensor matrix, we have found it simplest (though by no means fastest) to perform matrix interpolation. All the volume rendered figures in this paper were created this way, with any necessary eigensystem calculation being done after interpolation.

5 Acknowledgments

This work was supported in part by awards from the National Science Foundation, the National Institutes of Health, and the Department of Energy. The authors would like to extend special thanks to Andrew L. Alexander, Assistant Research Professor in the Department of Radiology, University of Utah School of Medicine, for involving them in his research on diffusion tensor imaging, providing them with high-resolution datasets, and contributing to discussions on visualization and analysis methods.

References

[1] D.C. Banks. Illumination in diverse codimensions. In *Proceedings SIGGRAPH 94*, volume 28, pages 327–334. Addison Wesley, 1994.

[2] P.J. Basser, J. Mattiello, and D. Le Bihan. Estimation of the effective self-diffusion tensor from the NMR spin-echo. *Magnetic Resonance*, 103:247–254, 1994.

[3] P.J. Basser and C. Pierpaoli. A simplified method to measure the diffusion tensor from seven MR images. *Magnetic Resonance in Medicine*, 39:928–934, 1998.

[4] James F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics*, pages 192–198, July 1977.

[5] Ed Boring and Alex Pang. Interactive Deformations from Tensor Fields. In *Proceedings IEEE Visualization 98*, pages 297–304, 1998.

[6] Brian Cabral and Leith Leedom. Imaging vector fields using line integral convolution. In *Computer Graphics Proceedings, Annual Conference Series*, pages 263–270, 1993.

[7] Thomas Conturo, Nicolas Lori, Thomas Cull, Erbil Akbudak, Abraham Snyder, Joshua Shimony, Robert McKinstry, Harold Burton, and Marcus Raichle. Tracking neuronal fiber pathways in the living human brain. *Proceedings of the National Academy of Sciences*, 96:10422–10427, 1999.

[8] J. Coremans, R. Luypaert, F. Verhelle, T. Stadnik, and M. Osteaux. A Method for Myelin Fiber Orientation Mapping Using Diffusion-Weighted MR Images. *Magnetic Resonance Imaging*, pages 443–454, 1994.

[9] T. Delmarcelle. *The Visualization of Second-Order Tensor Fields*. PhD thesis, Stanford University, Stanford, CA, 1994.

[10] T. Delmarcelle and L. Hesselink. Visualization of second order tensor fields and matrix data. In *Proceedings IEEE Visualization 92*, pages 316–323, 1992.

[11] T. Delmarcelle and L. Hesselink. A unified framework for flow visualization. In R. S. Gallagher, editor, *Computer visualization: graphics techniques for scientific and engineering analysis*, pages 129–170. CRC Press, 1995.

[12] R. R. Dickinson. A unified approach to the design of visualization software for the analysis of field problems. In *Three-dimensional Visualization and Display Technologies (Proceedings of SPIE)*, pages 173–180, 1989.

[13] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics Principles and Practice*, pages 592–595, 722–731. Addison-Wesley, second edition, 1990.

[14] Victoria Interrante and Chester Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings IEEE Visualization 97*, pages 421–424, 1997.

[15] D. K. Jones, S. C. R. Williams, and M. A. Horsfield. Full representation of white-matter fibre direction in one map via diffusion tensor analysis. In *Proc. 5th Int. Soc. of Mag. Res. in Med.*, page 1743, 1997.

[16] G.D. Kerlick. Moving iconic objects in scientific visualization. In *Proceedings IEEE Visualization 90*, pages 124–130, 1990.

[17] 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*, 1999.

[18] David H. Laidlaw, Eric T. Ahrens, David Kremers, Matthew J. Avalos, Russell E. Jacobs, and Carol Readhead. Visualizing diffusion tensor images of the mouse spinal cord. In *Proceedings IEEE Visualization 98*, pages 127–134, 1998.

[19] N. Makris, A.J. Worth, G. Sorensen, G.M. Papadimitriou, O. Wu, T.G. Reese, V.J. Wedeen, T.L. Davis, J.W. Stakes, V.S. Caviness, E. Kaplan, B.R. Rosen, D.N. Pandya, and D.N. Kennedy. Morphometry of *in vivo* human white matter association pathways with diffusion weighted MRI. *Annals of Neurology*, 42(6):951–962, 1997.

[20] Susumu Mori and Peter B. Barker. Diffusion magnetic resonance imaging: Its principle and applications. *The Anatomical Record*, 257(3):102–109, June 1999.

[21] J.D. Murray. *Mathematical Biology*, chapter 9, 14, 15. Springer-Verlag, Berlin, 1993.

[22] G.M. Nielson, H. Hagen, and H. Müller. *Scientific Visualization*, chapter 17. IEEE Computer Society, Los Alamitos, California, 1997.

[23] Sinisa Pajevic and Carlo Pierpaoli. Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: Application to white matter fiber tract mapping in the human brain. *Magnetic Resonance in Medicine*, 42(3):526–540, 1999.

[24] D.R. Peachey. Solid texturing of complex surfaces. In *Proceedings SIGGRAPH 85*, volume 19, pages 279–286. Addison Wesley, 1985.

[25] C. Pierpaoli and P.J. Basser. Toward a Quantitative Assessment of Diffusion Anisotropy. *Magnetic Resonance in Medicine*, pages 893–906, 1996.

[26] Carlo Pierpaoli. Oh no! One more method for color mapping of fiber tract direction using Diffusion MR Imaging Data. In *Proc. 5th Int. Soc. of Mag. Res. in Med.*, page 1741, 1997.

[27] Han-Wei Shen, Christopher Johnson, and Kwan-Liu Ma. Visualizing vector fields using line integral convolution and dye advection. In *Proceedings IEEE 1996 Symposium on Volume Visualization*, pages 63–70, 1996.

[28] D. Silver, N. Zabusky, V. Fernandez, and M. Gao. Ellipsoidal quantification of evolving phenomena. *Scientific Visualization of Natural Phenomena*, pages 573–588, 1991.

[29] Gilbert Strang. *Linear Algebra and Its Applications*, chapter 5.5. Academic Press, Inc., Orlando, Florida, 1976.

[30] A.M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of Royal Society of London*, 237(B):37–72, 1952.

[31] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings SIGGRAPH 91*, volume 25, pages 289–298. Addison Wesley, 1991.

[32] Greg Turk and David Banks. Image-guided streamline placement. In *Proceedings SIGGRAPH 96*, volume 30, pages 453–460. Addison Wesley, 1996.

[33] Aziz M. Uluğ and Peter C.M. van Zijl. Orientation-independent diffusion imaging without tensor diagonalization: Anisotropy definitions based on physical attributes of the diffusion ellipsoid. *Journal of Magnetic Resonance Imaging*, 9(6):804–813, June 1999.

[34] C. Upson, R. Wolff, R. Weinberg, and D. Kerlich. Two and three dimensional visualization workshop. In *Course Number 19, SIGGRAPH 89*. Addison Wesley, 1989.

[35] Samuel P. Uselton. Volume Rendering for Computational Fluid Dynamics: Initial Results. Technical Report RNR-91-026, NAS-NASA Ames Research Center, September 1991.

[36] Jarke J. van Wijk. Spot noise: Texture synthesis for data visualization. In *Proceedings SIGGRAPH 91*, volume 25, page 309318. Addison Wesley, 1991.

[37] C-F. Westin, S. Peled, H. Gubjartsson, R. Kikinis, and F.A. Jolesz. Geometrical diffusion measures for MRI from tensor basis analysis. In *Proceedings 5th Annual ISMRM*, 1997.

[38] Andrew Witkin and Michael Kass. Reaction-diffusion textures. In *Proceedings SIGGRAPH 91*, volume 25, pages 299–308. Addison Wesley, 1991.

[39] Andrew J. Worth, Nikos Makris, Van J. Wedeen, Jr. Verne S. Caviness, and David N. Kennedy. Fusion of MRI data for Visualization of White Matter Bundles, 1998. http://neuro-www.mgh.harvard.edu/cma/staff/ajw/MICCAI98/MICCAI98.html.

[40] M. Zöckeler, D. Stalling, and H-C. Hege. Interactive visualization of 3D-vector fields using illuminated streamlines. In *Proceedings IEEE Visualization 96*, pages 107–113, 1996.