

To the Graduate Council:

I am submitting herewith a thesis written by Nathaniel Richard Fout entitled "Integrated Visualization of Diffusion Tensor and Functional MRI." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Jian Huang
Major Professor

We have read this thesis and
recommend its acceptance:

Brad Vander Zanden
Jens Gregor

Accepted for the Council:

Dr. Anne Mayhew
Vice Provost and
Dean of Graduate Studies

(Original signatures are on file with official student records.)

Integrated Visualization of Diffusion Tensor and Functional MRI

A Thesis
Presented for the
Master of Science Degree
The University of Tennessee, Knoxville

Nathaniel Richard Fout
August 2003

DEDICATION

This thesis is dedicated to my loving wife, Maria. She not only encourages me to pursue my dreams, but also ensures their completion by taking part in them. I owe her a debt of gratitude that I will never be able to repay.

ACKNOWLEDGMENTS

I would like to thank several people who have helped me to achieve this accomplishment. First and foremost I would like to thank Dr. Jian Huang, who challenged me again and again to excel, both in his classes and in our research. He introduced me to and educated me in the fields of graphics and visualization, as well as allowed me to participate in exciting research in these fields.

As this project was not the work of one individual, but rather a team effort, I would like to thank Dr. Jian Huang, Dr. Zhaohua Ding, and Frederic Kuck for their work and guidance. Without their continued cooperation, labor, and wealth of ideas this thesis would not exist.

I would also like to thank Dr. Brad Vander Zanden and Dr. Jens Gregor for serving on my committee and for their demanding classes, which prepared me well for the future.

ABSTRACT

To understand the morphology, structure, and function of the human brain and the underlying relationships therein has long been a goal of mankind. Technologies are constantly emerging and evolving in an effort to realize this goal, with each new development potentially providing another piece of the puzzle. Among those technologies are advanced imaging modalities such as Diffusion Tensor MRI (DT-MRI) [2] and Functional MRI (fMRI) [8], whose purpose is to provide deeper insight into the neural network connecting functional units of the cerebral cortex. More specifically, DT-MRI captures a description of the fibrous structures (such as nerve fibers) in the brain, whereas fMRI studies the activity of various parts of the cerebral cortex when tasks are performed, thereby identifying the functional regions of the cerebral cortex and their correlation. Without knowledge of the underlying neural connections, however, the relationship between functional regions revealed by fMRI lacks structural support. Therefore, if DT-MRI could be coupled with fMRI, our view of the complex network within the brain could be revolutionized. Based on a comprehensive study of tensor field reconstruction, we have developed new methods for (i) white matter segmentation, (ii) nerve fiber reconstruction directly from tensors, (iii) discovery of globally optimal neuronal pathways, (iv) construction of consistent nerve bundles, and finally (v) integrated visualization of fMRI Regions Of Interest (ROI) with globally optimal neural networks constructed using techniques developed in (i) through (iv).

TABLE OF CONTENTS

1. Introduction.....	1
2. Background.....	4
2.1 Acquisition of DT-MRI and fMRI Images.....	4
2.2 DT-MRI Visualization.....	6
2.3 fMRI Visualization.....	11
3. Tensor Field Reconstruction.....	13
4. Visualization Methods.....	15
4.1 White Matter Surface Extraction.....	15
4.2 Bayesian Fiber Tracking in Tensor Fields.....	18
4.3 Optimum Neuronal Pathways.....	23
4.3.1 Finding the Optimal Pathway.....	24
4.3.2 Reconstructing Fibers Through the Pathway.....	26
4.4 Bundling of Nerve Fibers.....	27
5. Implementation of the Visualization.....	29
5.1 Design Goals.....	29
5.2 Design Decisions.....	31
5.3 Class Definitions.....	33
5.3.1 Tensor Class.....	33
5.3.2 TensorVolume Class.....	34
5.3.3 WhiteMatter Class.....	38
5.3.4 Fiber Class.....	38
5.3.5 FiberBundle Class.....	40
5.3.6 FiberNetwork Class.....	40
5.3.7 PGraph Class.....	41
5.3.8 VPath Class.....	45
5.3.9 RendererOGL Class.....	45
5.4 Parallel Implementations.....	46
5.4.1 White Matter Extraction in Parallel.....	47
5.4.2 Bayesian Fiber Tracking in Parallel.....	47
6. Results and Discussion.....	48
6.1 White Matter Surface Extraction.....	48
6.2 Bayesian Fiber Tracking.....	50
6.3 Fiber Bundling.....	53
6.4 Globally Optimal Pathways.....	53
6.5 Integrated Visualization.....	56
7. Conclusions.....	58

List of References.....	59
Vita.....	62

LIST OF FIGURES

Figures	Page
1. Discovery of the white matter surface.....	16
2. Action of Bayesian conditional probability in fiber tracking.....	22
3. Components of Nima.....	32
4. Implementation of the <i>ExtractWM</i> method.....	35
5. Implementation of the <i>Bundle</i> method.....	42
6. Implementation of the <i>FindOptimumPath</i> method.....	44
7. White Matter Surface Extraction.....	49
8. White Matter Surface (high resolution).....	49
9. Fiber Tracking Results.....	51
10. Large Scale Fiber Integration.....	52
11. Large Scale Fiber Bundling.....	54
12. K-most-similar Bundling.....	54
13. Optimal Pathway Search.....	55
14. Integrated Visualization.....	57

LIST OF ABBREVIATIONS

DT-MRI	Diffusion Tensor MRI
fMRI	Functional MRI
GHz	GigaHertz
GwS	Gaussian-windowed sinc
MRI	Magnetic Resonance Imaging
ROI	Regions of Interest
SNR	Signal-to-Noise Ratio
TDE	Tensor-Derived Ellipsoid

1. Introduction

To understand the morphology, structure, and function of the human brain and the underlying relationships therein has long been a goal of mankind. Technologies are constantly emerging and evolving in an effort to realize this goal, with each new development potentially providing another piece of the puzzle. Among those technologies are advanced imaging modalities such as Diffusion Tensor MRI (DT-MRI) [2] and Functional MRI (fMRI) [8], whose purpose is to provide deeper insight into the neural network connecting functional units of the cerebral cortex. More specifically, DT-MRI captures a description of the fibrous structures (such as nerve fibers) in the brain, whereas fMRI studies the activity of various parts of the cerebral cortex when tasks are performed, thereby identifying the functional regions of the cerebral cortex and their correlation. Without knowledge of the underlying neural connections, however, the relationship between functional regions revealed by fMRI lacks structural support. Therefore, if DT-MRI could be coupled with fMRI, our view of the complex network within the brain could be revolutionized. As a viable way to integrate the visualization of these imaging modalities has yet to be developed, this goal is currently unattainable.

Integrated visualization of DT-MRI and fMRI, as an emerging subject, faces two major research challenges. First, unlike volume rendering a scalar dataset (like anatomical MRI), direct, intuitive and effective visual analogies to tensor fields do not readily exist in 3D. Various methods do exist to visualize diffusion tensors, but the renderings allow mostly qualitative evaluation while providing less than adequate information from which fMRI studies can benefit. Some researchers have developed methods to reconstruct nerve fibers in the

brain as well. Those methods rely on reducing the tensor field to a vector field of major eigenvectors, within which conventional streamline tracing procedures are performed to obtain the fibers. Unfortunately, a tensor in general cannot be reliably reduced to a vector field, such as in cases where a tensor is depolated (i.e. the ellipsoid represented by the tensor matrix is not prolate, “cigar-shaped”, but oblate, “disc-shaped”, or spherical). As a result, existing approaches to visualize DT-MRI cannot effectively support neurological studies, where hypotheses need to be formulated and verified in a quantitative and reliable manner. A well-developed mathematical framework to visualize diffusion tensors is necessary. The second research challenge is how to develop a systematic study to support integrated visualization of DT-MRI and fMRI. After all, effective fMRI visualizations must be developed based upon quantitative results from DT-MRI datasets. Hence, approaches that merely allow qualitative visual inspection are insufficient.

To address the above challenges, we present an approach for modeling the brain from DT-MRI datasets using a new mathematical framework based on Bayes Rule of conditional probability and global optimization by dynamic programming. The models obtained by our approach incorporate individual nerve fibers, fiber bundles and the white matter surface depicting the collective boundary of all nerve bundles. Nerve fibers are constructed directly from tensor fields without simplifying tensors to vectors, and to address the crippling effects of noise on neural pathway reconstruction we designed an efficient dynamic programming method to discover globally optimized neural pathways. This is an improvement over previous approaches that rely solely on signal smoothing by low-pass filtering. In addition, our dynamic programming approach (using Bayesian fiber tracking, fiber bundling and white matter surface extraction) is the first to reliably support the

testing of hypotheses about the nature of the connection between any two functional areas on the cerebral cortex. Our approach also handles cases where nerve fibers may cross, merge or diverge. Finally, by using these techniques, we are able to integrate the visualization of DT-MRI and fMRI datasets.

To summarize, our proposed framework supports quantitative investigations, through visual interaction, by modeling the brain from DT-MRI and fMRI. Based on a comprehensive study of tensor field reconstruction, we have developed new methods for (i) white matter segmentation, (ii) nerve fiber reconstruction directly from tensors, (iii) discovery of globally optimal neuronal pathways, (iv) construction of consistent nerve bundles, and finally (v) integrated visualization of fMRI Regions Of Interest (ROI) with globally optimal neural networks constructed using techniques developed in (i) through (iv).

In the next section we give a brief description of diffusion tensors, followed by a discussion of previous work in the areas of DT-MRI and fMRI visualization. We then discuss in Section 3 issues regarding tensor fields in general and the reconstruction of tensor-derived quantities. Section 4 presents in greater detail our approaches and visualization algorithms, and in Section 5 we describe an implementation of these methods. Finally, in Section 6 we present and discuss our results, and in Section 7 we draw our final conclusions.

2. Background

2.1 Acquisition of DT-MRI and fMRI Images

Water molecules undergo random motion in biological tissues, commonly referred to as the diffusion process. Local diffusion in three dimensions is described by a symmetric rank-2 tensor matrix T_{xyz} of size 3x3, and can be represented as:

$$T_{xyz} = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{pmatrix}$$

A useful analogy to the diffusion tensor is to consider a diffusion tensor matrix as defining an ellipsoid of a certain size, shape and orientation, such that the surface represents the root-mean-square (RMS) diffusive displacement. This ellipsoid, which we call the tensor-derived ellipsoid (TDE), provides a useful analogy that is exploited in many DT-MRI visualization algorithms. Clearly, the DTE defined by a tensor matrix can be an isotropic sphere or anisotropic shapes such as a disc or spindle. The trace of the diffusion tensor matrix is rotationally invariant and serves as the basis for the directionally averaged diffusivity $\bar{D}$, defined as $\bar{D} = (D_{xx} + D_{yy} + D_{zz})/3$. In relation to a reference, or world, coordinate system, the ellipsoid is described by a rotation matrix R and a diagonal matrix D , where $T = RDR^{-1}$. The diagonal matrix elements are the eigenvalues of the diffusion tensor, and are the principal diffusivities of the rotated ellipsoid. The columns of R contain the eigenvectors of the

system and are associated with the eigenvalues of the diagonal matrix, respectively.

DT-MRI measures varying substance diffusion processes among different tissues in the human brain [2], relying on the fact that different tissue types, especially areas with highly consistent neuronal fiber bundles, have contrasting anisotropic diffusion rates in different spatial directions. Although still on a coarse spatial scale, diffusion MRI is revolutionary in providing the first physical data describing structural information within brain tissues, which show up as homogeneous in other modalities.

Diffusion tensors are measured by sensitizing the magnetic resonance signal intensity to the random motion of water molecules. The 3D diffusion tensor datasets analyzed in this study were acquired with a 1.5 Tesla GE Signa LX magnetic resonance scanner, using a pulsed gradient spin echo pulse sequence. To minimize motion artifacts and reduce scanning time, single shot echo-planar imaging was used. The size of the scanned volume was $20 \times 20 \times 9 \text{ cm}^3$, generating a data set of $64 \times 64 \times 18$ voxels. For each voxel, seven measurements are made: one with no diffusion weighting and six with diffusion sensitization along non-collinear directions. The six independent coefficients of the diffusion tensor matrix are calculated using multivariate linear regression. To improve the signal-to-noise ratio (SNR), four repeated scans were used for each data set, and the final diffusion tensor matrix is calculated from the arithmetic mean of the diffusion tensor measurements.

fMRI studies the functional activities of a subject when a designated task is performed [8]. On each voxel in an fMRI dataset, depending on the number of ROIs under investigation, dozens of correlation coefficients are computed using statistical approaches and stored for later analysis. The coefficients describe whether an ROI is

stimulated, inhibited or irrelevant when a certain test task, such as answering a question, is performed by the subject.

fMRI data for this study was obtained using a 1.5 Tesla Signa MR imaging system. Prior to functional imaging, 14 axial-oblique T_1 weighted anatomic images are taken parallel to the inter-commissural line based on sagittal localizer images. Axial-oblique images are obtained at the same relative slice location in each subject, extending from the inferior aspect of the temporal lobes to the parietal convexity. Activation images are collected using single shot, gradient echo, echo planar acquisitions in the same slice locations used for the anatomical images. In conducting the experiments an event-related Average Single Trial protocol was used. Correction of images for motion was accomplished using Statistical Parametric Mapping software, followed by spatial and temporal image filtering. Statistical maps showing distributions of activation were derived using the standard univariate analysis method of the General Linear Model on a voxel by voxel basis.

2.2 DT-MRI Visualization

Various methods have been proposed for visualization of DT-MRI data. In this section we will discuss these methods, especially those that relate directly to the techniques developed herein. The methods are sorted into three representative groups: global visualization, white matter segmentation, and fiber tractography.

Global visualization methods attempt to present a comprehensive view of the entire data set (or at least a significant part of it), displaying all or most of the information contained in the tensor field. The technique developed in [9] falls into this category; this method simply displays on each voxel a small ellipsoidal icon. Although this

visualization displays all information contained in the tensor matrix, images of more than a few tensors quickly become cluttered. Also, the resulting visualization lacks continuity and cohesion, a result of the discrete nature of the representation. In [11] two techniques are discussed, one of which also uses ellipsoidal icons. The first method begins by normalizing the ellipsoids on each voxel so that they are about the same size. Then the visualization displays a single slice of these ellipsoids. The second method is based on principles from oil painting. This method also operates on a single slice to produce a 2D visualization, and encodes all information contained in the tensor. The image is created by applying layers of brush strokes, where the attributes of the brush stroke (color, direction, length, texture, etc.) encode various properties of the tensor matrix. The intent of this method is to encode the tensor information in such a way that at different scales different patterns are visible. However, extension of these methods to a 3D volume would result in an incomprehensible image, primarily due to cluttering.

Another global visualization of the tensor field based on direct volume rendering is developed in [10]. In this framework, the transfer function (a mapping from data values to color/opacity) is split into two separate mappings. The first, called *Hue-balls*, maps the diffusion action of the tensor to a color; the second, called the Barycentric Opacity Mapping, maps the anisotropy of the tensor to an opacity. Finally, the canonical Phong lighting model is modified to handle diffusion tensors. This lighting model, called *Lit-Tensors*, treats prolate tensor ellipsoids as illuminated streamlines and oblate ellipsoids as normal surfaces. Modifications to the lighting model include provisions to interpolate between these two models of lighting, depending on the nature of the tensor ellipsoid. The results of these enhancements allow effective and

informative renderings of the white matter, showing various known structures. The only drawback is that it is difficult to distinguish individual structures when they overlap in the renderings. This is an artifact of the direct volume rendering process, which yields only implicit surfaces, from which it is difficult to disambiguate spatial relationships.

Methods to visualize white matter attempt to segment the volume into two partitions: white matter and all other structures. The motivation for this spatial segmentation is that the white matter defines regions of high diffusion, indicating the presence of nerve fibers. Therefore, important contextual information is provided by discovering the boundary of the white matter. While it is generally agreed that areas of anisotropic diffusion indicate white matter and areas of isotropic diffusion indicate other regions (such as gray matter), what definition of anisotropy best segments the white matter is a point of contention. In [19], Zhukov *et al.* propose an anisotropy index based on tensor invariants (i.e. properties of the tensor that are invariant to the frame of reference), of which there will be three (since the tensor is of rank 2). One set of tensor invariants is the eigenvalues, and most other methods define anisotropy in terms of these. In [19] the invariants are chosen to be the coefficients of the characteristic equation whose roots define the eigenvalues. These invariants are more efficient to compute than eigenvalues, and are also less susceptible to noise. Using a definition of anisotropy based on these invariants, white matter surfaces were successfully extracted using Marching Cubes [12] followed by level set smoothing. However, the use of Marching Cubes along with level set smoothing in order to obtain a coherent, smooth surface blurs details and causes artifacts, such as unexplained holes. The general appearance of these images matches that of known white matter

structure; however, the accuracy of the boundary locations is questionable.

The last area of DT-MRI visualization is fiber tractography, in which paths are traced through the tensor volume in an attempt to recreate the fiber, or fascicle, pathways. Normally the fibers obtained from these methods depend very strongly on the location of the seed points, and are very noise sensitive, since one wrong step will most likely cause the following steps to be wrong as well.

The simplest approach, described in [20], reduces the tensor field to a vector field consisting of only the major eigenvector of the tensors. Streamlines are then traced through the volume as usual. This method makes the critical assumptions that the major eigenvector is 1) unique and 2) always gives the direction of the fiber that was tracked from the previous step. Also, important information is discarded, resulting in an incomplete picture of the tensor field. In [6] hyperstreamlines make use of all the tensor information by tracing out an elliptical tube along the trajectory defined by the major eigenvector. The dimensions and orientation of the tube reflect the medium and minor eigenvalues and eigenvectors, and the color along the tubes length indicates the major eigenvalue. This visualization includes all the key data; however, it still makes the same assumptions as before. In addition, the volumetric paths carved out by hyperstreamlines introduce occlusion problems in the resulting image; to preserve visual clarity only a few hyperstreamlines can be displayed at one time.

In [16] the path is again represented by simple lines, called tensor lines. The difference between tensor lines and streamlines is that tensor lines address the first assumption from above; a unique major eigenvector is not assumed. Instead, the tensor is characterized in one of three ways. If the tensor displays linear anisotropy, then there does exist

a unique major eigenvector, and this is the direction of propagation. If the tensor is planar anisotropic, then there exists a plane in which the major eigenvector is restrained to reside. If the tensor is spherical, then the major eigenvector is completely ambiguous. Tensorlines handle these cases by simulating the path as being traced out by a particle possessing momentum. The outgoing direction at any stage is determined by the classification of the tensor at that point and the incoming direction of the particle. In this way the tensorline proceeds straight through a region of isotropy, whereas streamlines and hyperstreamlines will possibly make erratic turns based on arbitrary determinations of the major eigenvector. This advantage allows tensorlines a more stable fiber trace (especially when tracking near the white matter boundaries), producing fibers which are more consistent with known structures. However, this method has the potential to introduce continuities where none exist, as pointed out in [3]. In this work, vector field integration using the major eigenvector is again promoted as the mathematically correct method of fiber tractography. The authors note the limitations of this scheme, but caution against methods which employ history, bending energy, regularization of the fiber trajectories, or smoothing of the data. The authors argue that these techniques may artificially introduce smooth, continuous, straight fiber segments which do not exist in reality. Other contributions from [3] include the generation of synthetic DT-MRI data sets as a way of evaluating fiber tracking methods, and a comprehensive discussion of the pitfalls inherent in DT-MRI tractography.

Poupon et al. [13] present a method of fiber tracing which is based on regularization of the direction map (a 3D graph representing voxel connectivity) using a priori knowledge of the nature of fiber geometry. This method proceeds in two parts: first the direction map is regularized,

and second fibers are tracked by simply following the direction map. Regularization of the map depends on the local curvature and bending energy calculations. The result is that fiber paths retain a directional coherency. One advantage of this algorithm is that it allows for fiber branching, in which fibers diverge and are tracked in parallel. This method also addresses the partial volume effect (when the tensor is the average of more than one fiber). The main disadvantage is that regularization can change the connectivity of the direction map. The degree of regularization is specified by the user, and it is not clear when the fidelity of the data set has been compromised by this procedure.

Recent work by Zhukov and Barr [18] also uses regularization to create smooth fibers. In their method a moving oriented least-squares filter is applied to tensor interpolation on the fly. Again the major eigenvector gives the direction of fiber tracking.

2.3 *fMRI Visualization*

As fMRI is a relatively new imaging modality, research into visualization of fMRI data is in the early stages of development. Paramount to visualization of fMRI data is correct interpretation of the signals from which the data set is generated. A comprehensive review of the current understanding of fMRI technology and what it really tells us about the underlying neuronal activity is given by Heeger and Ress [8]. A discussion of previous work in fMRI visualization follows.

In [1], Konig *et al.* call attention to the difficulties of direct volume rendering fMRI data together with anatomical data. They then propose enhancements to direct volume rendering in order to overcome these difficulties. The primary obstacle to this multimodal visualization is that although transparency allows regions of interest located within the

anatomy to be seen, it is very difficult to ascertain their position without compromising anatomical detail. To address this problem they propose two enhancements to direct volume rendering: transfer function volumes and multiple views. Transfer function volumes define not one transfer function for the entire data set, but rather a volume of transfer functions, assigned on a voxel-by-voxel basis. In this way special transfer functions can be used to highlight regions of interest or cut out irrelevant regions that saturate the opacity channel. The second enhancement is to display additional views of the objects. In their implementation of this idea, three additional views are generated and displayed below and behind the objects in the scene. These views provide additional three-dimensional contextual information about the regions of interest without having to continually rotate the scene. In addition to showing the object from a different perspective, these additional views can employ alternative projections or even non-volume rendering techniques such as contour maps to complement the primary view.

Whereas in [1] only the location of the regions of interest are emphasized, in [17] Welsh *et al.* also emphasize the relationships among these regions. Statistical exploration of these relationships is conducted via correlation analysis and principal component analysis on a regional basis (as opposed to a voxel basis). The visualization is accomplished by rendering the regions of interest as polygonal objects, with an iso-surface of the brain providing contextual information. The relationships among regions of interest are displayed through color channels, and a GUI is provided to allow interactive exploration.

3. Tensor Field Reconstruction

Reconstruction of scalar fields is typically accomplished by interpolation, with the interpolation kernel determining the speed and accuracy of the reconstruction. On one extreme is the box kernel, which gives very fast but inaccurate reconstruction; on the other extreme is the sinc kernel, which is slow but gives accurate reconstruction. A common interpolation kernel is the triangle kernel (a.k.a trilinear interpolation), which offers a trade-off between accuracy and speed. Assuming a band-limited, noise-free scalar field and an adequate sampling rate, the sinc kernel will perfectly reconstruct the original scalar field. However, if the field is noisy reconstruction of the original signal is problematic, despite the choice of interpolation kernel.

Reconstruction of vector fields is analogous to reconstruction in scalar fields, with the interpolation proceeding component-wise. Reconstruction of tensor fields, however, presents new problems not encountered in scalar or vector fields. These new problems stem from the fact that methods to visualize tensor fields do not use the tensor directly, but instead use quantities derived from the tensor. For instance, scalar quantities derived from tensors can be subjected to established scalar visualization algorithms. Therefore the issue arises of whether one should interpolate the derived quantities or the tensors. If the dimensionality of the derived quantity is less than that of the tensor, then it is more efficient to pre-compute that quantity at each voxel and then interpolate; that is, we perform post-interpolation. The alternative is to first interpolate the tensor component-wise, and then to calculate the derived quantities. This pre-interpolation process is analogous to reconstruction in vector fields. As far as accuracy is concerned, only mathematical quantities derived from the components of a tensor by

linear combinations can be correctly reconstructed by post-interpolation. Other quantities will not be reconstructed correctly by this process, resulting in the introduction of an unknown amount of error in the reconstruction. The degree of error depends on several factors, but in general if the derived quantity has a much higher spectrum bandwidth than the tensor field then the quality of the reconstruction will be poor. Some frequently used quantities that exemplify this behavior are eigenvectors, eigenvalues, and anisotropy indices.

The visualization methods described herein (as well as other pre-existing tensor visualization methods) rely on quantities of this nature. This fact, combined with the inherent error in DT-MRI data sets due to noise, necessitates accurate reconstruction by pre-interpolation. In addition to this, the interpolation kernel must be chosen with these facts in mind. Trilinear interpolation, although efficient to compute, is insufficient. The sinc kernel gives much better reconstruction, but is susceptible to the high level of noise present in typical DT-MRI data sets. In the end it was found that a Gaussian-windowed sinc (GwS) function provides the dual functionality of handling noise while providing accurate reconstruction. The spatial extent of the GwS reconstruction kernel used herein is 2.0 in voxel sizes along each of the coordinate axes.

4. Visualization Methods

This section describes visualization algorithms designed and/or selected to meet the needs of an integrated visualization of the human brain. The methods are: white matter surface reconstruction, Bayesian fiber tracking, globally optimal pathway reconstruction, and fiber bundling. Each method is given a motivational introduction, a detailed description, and a discussion of what role it plays in the integrated visualization.

4.1 White Matter Surface Extraction

White matter, present throughout much of the central area of the brain, defines key regions pertaining to the structure and function of the human mind. White matter consists of collections of nerve bundles within which paths of high diffusion are found. The discovery and extraction of the boundary representing white matter provides an important high-level framework, establishing a domain within which nerve fiber/bundle operations can be applied.

The key to finding the white matter boundary is to find a metric that classifies tensors as either on the boundary or not. Several definitions of what constitutes the white matter boundary have been proposed, and the debate as to which metric is better remains open. Some metrics are derived from diffusion rates, whereas most are defined in terms of the anisotropy of diffusion. A satisfactory metric would extract a 3D surface which resembles the commonly known "butterfly" shape of white matter, and this is the qualitative criterion we use to discriminate among the various plausible alternatives. Several metrics were tested by using a flooding algorithm (Figure 1) in the following manner: a point is

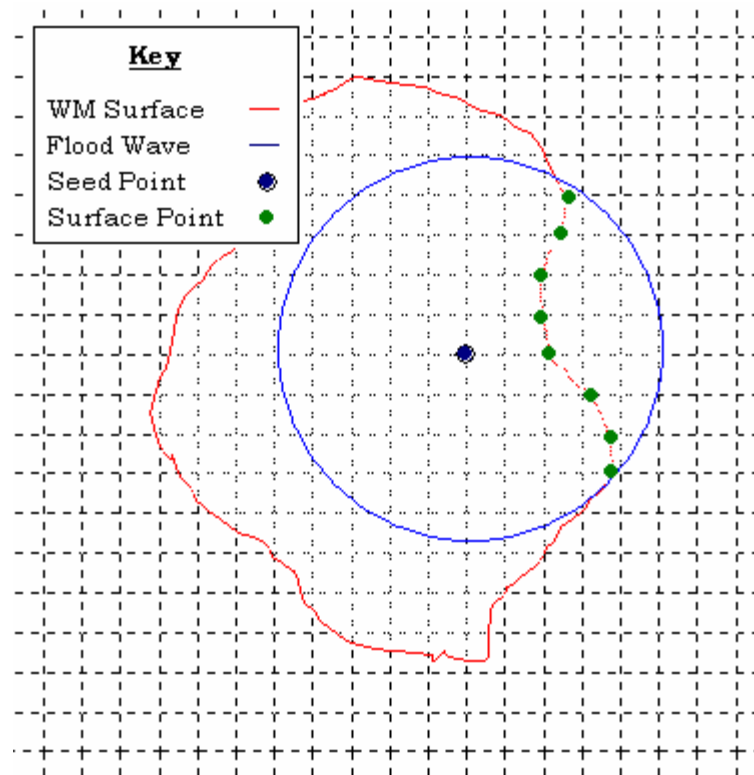


Figure 1: Discovery of the white matter surface

identified as being inside the white matter, and from that point we flood outward in all directions. The waves spread until they cross the white matter boundary, as determined by the metric under consideration. Points are then generated to mark the position of the surface. Beyond the surface boundary flooding ceases. Once all "particles" of the wave have crossed the white matter boundary (or the data set boundary) the resulting surface consisting of points is compared to the known shape of the white matter. Given this testing method, experimentation revealed only one suitable metric, given by the following definition of anisotropy, A_σ :

$$A_\sigma = \frac{1}{\sqrt{6\bar{D}}} \times \sqrt{\sum_{i=x,y,z} (D_{ii} - \bar{D})^2 + 2(D_{xy}^2 + D_{xz}^2 + D_{yz}^2)} \quad (1)$$

where $\bar{D} = (D_{xx} + D_{yy} + D_{zz})/3$. A_σ is based on the average of diffusivity in all directions, as in $A_\sigma = \sigma(T)/(\sqrt{2\bar{D}})$. $\sigma(T)$ is the standard deviation of the diffusion coefficients measured by the DT-MRI and then encoded as the tensor matrix. The denominator of A_σ can be considered as the average radius of the ellipsoid (average of the eigenvalues). As a measure of anisotropy, this definition of A_σ is ideal in that it isolates the energy of anisotropy from the energy of isotropy (the baseline) and computes the ratio of the two, thus allowing a single index to describe the anisotropy of the diffusion tensor.

Since A_σ cannot be computed as a linear combination of the components in a tensor matrix, we cannot use methods like Marching Cubes that assume a piece-wise linearity between voxels to extract the white matter surface. The flooding algorithm described previously is sufficient to find the white matter surface; however, the algorithm is very

slow in practice. An alternative that allows parallel implementation is to divide each voxel into subvoxels, on which A_σ is computed from the interpolated tensor matrix. Only subvoxels with a value close to the iso-value are extracted as points on the surface. This method was first proposed in [4] and is known as *Dividing Cubes*. In practice the algorithm produces isolated pockets of subvoxels falsely marked as boundary points in addition to the white matter surface. Most likely these pockets of false surface points are due to either signal noise or isolated segments of nerve fibers outside the white matter boundary. We remove all those extraneous structures using a flooding operation originating from within the white matter. The flooding operation also allows us to identify the interior of white matter from the exterior.

Once a subvoxel has been identified as a surface point, the normal is computed by central differencing A_σ . Although the resulting white matter surface is a point model, one can use computational geometry approaches to construct a surface mesh from the surface points.

Extraction of the white matter surface provides a visual context which greatly aids in an integrated visualization of the brain. Moreover, it provides a quantitative focus for the remaining visualization methods.

4.2 Bayesian Fiber Tracking in Tensor Fields

Tracking fibers within the white matter provides both a candidate region for planting seeds and a simple stopping criterion; however, we still need to know which direction to follow as numerical integration proceeds. Unlike vector fields, where the local direction of flow is defined without ambiguity, tensor fields require a more sophisticated computation to make this decision. One common method is to reduce the tensor to its major eigenvector. This reduction defines two possible

directions (the eigenvector and its negative), of which the one that is more consistent with the recent integration path is chosen. This method is equivalent to streamline tracing. Unfortunately, this approach only works well for tensors with distinctive major eigenvectors. For depredated TDE's (tensor-derived ellipsoids) in areas where fibers join or cross each other and areas close to the surface of nerve bundles, it is difficult to reliably trace streamlines in this way. While some estimation methods can be used to circumvent this problem, for instance, based on the directional momentum from previous steps (as in the tensorline approach [16]), we believe a more rigorous mathematical analysis of the governing principles to trace streamlines in a tensor field is necessary.

In order to develop a mathematical framework for fiber tracking in tensor fields, we first make the observation that the tensor, or equivalently the TDE, tells us the likelihood of diffusion in a given direction; that is, the surface of the TDE represents a 3D probability distribution. When we consider all the points of the TDE in spherical coordinates, where points are represented as $P = (\theta_p, \phi_p, r_p)$, then r_p is the relative probability of diffusion in the direction given by the angles θ_p and ϕ_p . The generality of this interpretation of diffusion tensors provides the mathematical foundation for a rigorous treatment of the fiber tracking process. At this point it is worth noting that the streamline method mentioned previously is equivalent to following the direction of highest relative probability.

As we mentioned above, simply following the direction of highest probability is not enough to track fibers in all situations. In order to achieve this, we make the observation that the tensor field inside the white matter is a description of real, contiguous pathways. We extend our mathematical framework to take advantage of this pathway coherency by employing the concepts of conditional probability,

specifically Bayes rule on conditional probability [15]. Based on Bayes rule, we dynamically determine the local direction to follow considering both the local tensor and the incoming direction of the fiber. According to Bayes rule, in an event space U , such as all possible outgoing directions from a voxel, there exists a partition, B_i ($i=1..n$), of U , with B_i being each possible outgoing direction. In addition, a fiber may or may not arrive at a local voxel from a certain direction, referred to as event A . Bayes rule states that the probability of event B_i happening given that event A has happened, $P(B_k|A)$, is computed as:

$$P(B_k|A) = \frac{P(B_k)P(A|B_k)}{\sum_{i=1}^n P(B_i) \times P(A|B_i)} \quad (2)$$

where $P(B_k)$ is the probability of event B_k taking place under all conditions, and $P(A|B_k)$ is the probability of event A happening given the condition that event B_k has already happened. $P(B_k)$ is given by the 3D probability distribution for the local TDE. To compute Equation (2), the only remaining unknown is the conditional probability, $P(A|B_k)$, which represents the probability of a fiber, i.e. event A , bending its direction to B_k . We can model $P(A|B_k)$ based on the concept of bending energy, i.e. the “energy” needed to bend a nerve fiber. In $P(A|B_k)$, the maximal value is achieved when the angle between A and B_k is zero, i.e. no bending, while the value of $P(A|B_k)$ decreases as the angle widens. The profile of such a fall-off can be any smooth function demonstrating this type of behavior. We used the *cosine* function to model bending energy because

it can be computed as a simple dot product between the two vectors. Obviously, choosing a uniform bending energy function would reduce this method to the previous approaches of simplifying a tensor field to a vector field. Computing Equation (2) is straightforward; however, since we are only interested in the relative probabilities so as to find out the direction with maximal conditional probability, it is more efficient to compute only the numerator. In practice, one cannot check the probability of all possible outgoing directions. We sample the spherical coordinate space on a 40 by 40 grid in all our experiments. That is, in each step of numerical integration, after the local tensor is properly interpolated, the probabilities of 1600 candidate outgoing directions are computed, based on the incoming direction.

Figure 2 demonstrates the effectiveness of Bayesian fiber tracking in terms of probability distribution functions, where for each spherical direction (θ, ϕ) a relative probability is computed. In Figure 2 the incoming direction is $(0, \pi/2)$, which in rectangular coordinates is the X -axis. For each general type of ellipsoid (prolate, oblate, and spherical) Bayes rule is computed for all outgoing directions.

Bayesian fiber tracing addresses the major limitation of tensor deprolation encountered by previous methods of fiber tracking. While a prolate ellipsoid provides a distinctive single direction of highest diffusion, an oblate ellipsoid or a spherical ellipsoid does not, making it ambiguous to search for a direction of highest diffusion. In these situations, the use of Bayes rule advances the search of a proper outgoing direction from a static binary decision to a quantitatively measured dynamic choice. This benefit leads to a more stable trace of fibers, resulting in a large degree of consistency and smoothness. Additionally, Bayesian fiber tracking can benefit from oriented regularization [18] to further address local noise if necessary.

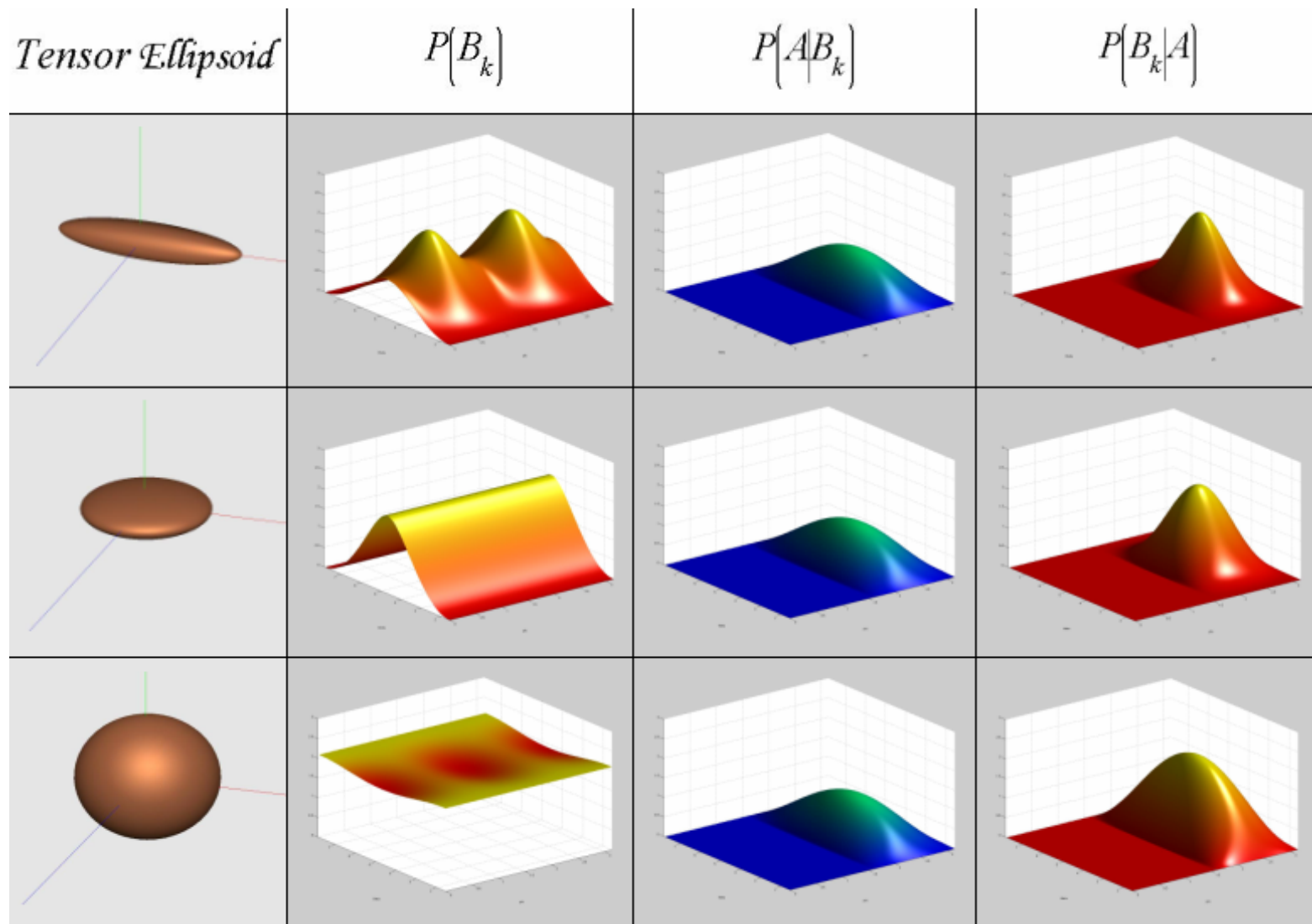


Figure 2: Action of Bayesian conditional probability in fiber tracking

Finally, the value of Bayesian fiber tracking goes beyond just being a rigorous way to trace fibers in tensor fields. The paramount advantage offered by Bayesian tracking is to associate a probability with each outgoing direction. With this information it is possible to undertake the task of constructing globally optimal pathways in DT-MRI, a key ingredient in our integrated visualization.

4.3 Optimum Neuronal Pathways

The most useful information DT-MRI provides to aid fMRI visualization is the neural network connecting various regions on the cerebral cortex. But to obtain this information, three problems must be solved, namely: (i) noise and numerical errors affecting the accuracy of fiber tracing; (ii) handling branching and crossing of fibers, and (iii) identification of optimal pathways connecting any two given end points, if such a pathway is determined to exist.

First, fiber tracking is based on numerical integration, and as such accumulates the effects of noise and numerical errors. Hence, the accuracy of a reconstructed fiber is hard to determine. While local noise reduction methods [19] have been shown to be effective, the global effects of error accumulation are still unknown. Second, nerve fibers may cross, merge and diverge. Currently, there is no reliable way to handle such situations. Bayesian fiber tracking cannot solve this problem by itself either, since this requires considering multiple potential choices of similar probabilities, where choosing the maximal probability seems quite arbitrary. The last unresolved issue is that queries regarding optimal paths will most likely be posed in such a way as to not be amenable to the canonical method of fiber tracking. Whether two spatial locations are connected and how they are connected cannot be answered

by local studies of where a fiber goes from the current voxel. It is not clear if any existing approaches can reliably solve all three of these problems.

4.3.1 Finding the Optimal Pathway

All of the above questions involve a core element of global optimization; therefore, we address these issues by resorting to the paradigm of dynamic programming [5]. The essential problem we study is then formulated as:

“Given two end points, S and D , is there a nonempty set of plausible fibers connecting S and D and if there is, which is the optimal fiber.”

The metric of optimality, or cost, can be defined in many ways without loss of generality in our overall algorithm. Plausible choices include shortest path, shortest path weighted by diffusion rates along the path, maximal anisotropy along the path, etc. Combinations of these options are also possible. Using any of these definitions of cost, denoted as ‘ m ’, we subdivide the global problem recursively according to Equation (3):

$$m(S,D) = \min_{\forall N_S} [m(S,N_S) + m(N_S,D)] \quad (3)$$

where N_S represents all neighbors of S . The sub-problems are to determine the optimal path from N_S to D , which in turn is based on the

optimal path based on the costs $m(N_S, N_{N_S})$ and $m(N_{N_S}, D)$ for all neighbors of N_S . We define $m(S, S) = 0$. The optimal fiber should result in a minimal value of $m(S, D)$. In assigning the initial costs, m , between voxel neighbors we use Bayes rule. Since each voxel has 26 direct neighbors in 3D discrete space, using Bayes rule we can efficiently determine for each voxel S , given an input direction from a neighbor, N_S , the probability of the fiber leaving in the other directions. For all voxels directly neighboring each other, not all pairs have a high enough probability to have a fiber connecting them directly; therefore, using the computed probabilities, low likelihood direct connections are penalized with large m values. As an example, consider shortest path as the optimal criterion. The initial cost, m , between neighboring voxels, should be the distance between two voxels, weighted by the inverse of the probability of such a connection to exist. One must note that between any pairs of neighboring voxels, for different incoming directions and outgoing directions, there are different m values on the same link connecting the two voxels. All such m values must be stored for later use.

Since our problem is 3D in nature, the classical way of using dynamic programming would require creating a cost table that stored the value m for all possible voxel pairs in the volume. This results in a table too large to be practical, and so our algorithm builds the table dynamically. The entire table is not needed anyway, as the search is limited to only the white matter region. We could build a static table containing only voxels in the white matter; however, by building the table dynamically, we have the ability to limit the search radius. This can be especially useful in speeding up the search by eliminating unlikely paths from consideration. This option must be used with care; otherwise the

optimum discovered may be a local, not global, optimum. One appropriate use of this option is the shortest path cost metric.

The essence of dynamic programming is to compute any sub-problem only once. The results of precomputed sub-problems are reused by all problems sharing those sub-problems. In our case, we are not interested in every possible nerve fiber in existence in the brain. Instead, our interests mostly revolve around those fibers connecting a relatively small number (usually around a dozen) of ROIs. This allows us to significantly limit the overall size of the problem dealt with in dynamic programming. We only need to construct a very small subset of the whole problem space, which is discovered on the fly as the optimal paths are being constructed.

4.3.2 Reconstructing Fibers Through the Pathway

The result of the dynamic programming process is a sequence of voxels that a certain path must go through. We then trace fibers through these voxels, connecting the two ends. A straightforward way to reconstruct these fibers is to do numerical integration by always following the direction pointing towards to the center of the next voxel in sequence. However, this simple approach produces fibers with very jagged geometries. A more sophisticated approach is to slightly modify the Bayesian tracking method in Section 3.3. Let us rewrite Equation (2) below:

$$P(B_k|A) = \frac{P(B_k)P(A|B_k)}{\sum_{i=1}^n P(B_i) \times P(A|B_i)} \quad (2)$$

Here, in global optimal path reconstruction, the only change we make is that event A now becomes the direction pointing towards the next voxel in sequence. We still use the semantic of bending energy to attenuate the probabilities delineated by the reconstructed local tensor, i.e. $P(B_k)$. Furthermore, to smoothly track tight turns we use an adaptive step size in the numerical integration, linearly correlated with the angle between the incoming direction and the known intended direction. This approach provides stable fiber paths that closely follow the specified path sequence.

4.4 Bundling of Nerve Fibers

It is very difficult to visually sort through large numbers of neural fibers in search of distinctive patterns. By bundling similar fibers into groups, nerve pathways become visible on a more manageable scale, facilitating fast comprehension of the structures within. In a way, having a number of fibers of consistent pattern is also a strong confirmation of the existence of a real neural connection.

We use our K-most-similar bundling method developed previously in [7] for fiber bundling. In that method, a fiber considers its neighboring fibers as candidates for bundling. The pair-wise similarities between a fiber and each of its neighbors are computed. Neighboring fibers with a similarity below a threshold are not considered in the bundling process. Among the remaining neighbors, the K neighboring fibers most similar to the fiber in question are bundled together with that fiber. If a K-most-similar neighboring fiber already belongs to a bundle, all the K-most-similar fibers are coalesced into that existing bundle. In practice, we use a K value of four. The similarity metric we use is the spatial proximity metric, based on computing the corresponding segments of two fibers

and then the averaging the distance between the two corresponding fiber segments [7]. The corresponding segment of two fibers is defined as the portion of those fibers where a point-wise correspondence exists. This criterion is effective in grouping fibers that visually “look alike”, providing results that correlate well with real structures.

To display bundles, we color code fibers in the same bundle using a common color. This way, no addition modeling is necessary and all of the original fibers are shown.

5. Implementation of the Visualization

This section describes the implementation of the software system called Nima (Greek for fiber). The primary objective of Nima is to realize the goal of integrated visualization via the techniques developed in Section 4. We begin with a review of the system requirements and a discussion of the design decisions, followed by a discussion of how each of the methods in Section 4 is implemented in Nima.

5.1 *Design Goals*

The system requirements for Nima fall into two different categories: first, there are the computational requirements, which dictate the design of the algorithms, and second there are the rendering requirements, which mandate the design of the renderer. The computational requirements for the system are:

1. Accurate reconstruction of the tensor field
2. Extraction of high resolution point models of the white matter surface in a reasonable amount of time
3. Efficient and accurate integration of the tensor field using Bayesian Fiber Tracking, including flexibility in seed point(s) selection, termination criteria, probability function calculation and sampling, and accuracy
4. Computation of the globally optimum path via adaptive dynamic programming in a reasonable amount of time, allowing flexibility in the maximum path length limit and the maximum path weight limit

5. Point-to-point integration of fibers through the optimum path using an alternate probability function
6. Efficient bundling of large numbers of fibers, allowing flexibility in the bundling parameter, K , and bundling criteria
7. Presentation of quantitative results for use in case studies

The rendering requirements are:

1. Ability to render all objects of the integrated visualization in real-time
2. Ability to manipulate rendering properties such as color, transparency, lighting, shading, etc.
3. Rendering of white matter point structures with normals based on tensor field gradients
4. Rendering of integrated neural fibers with correct shading, allowing modulation of rendering properties along the fiber length based on tensor field properties
5. Rendering of the optimum path, along with other quantities which aid in the visualization of the optimum path search
6. Rendering of fiber bundles; that is, rendering fibers in a common color to indicate their membership in a bundle
7. Generation of high quality images that clearly depict all aspects of the visualization

These requirements represent the basic goals of the system and its current state of implementation. Additional goals include a graphical user interface, stereo viewing, as well as several other small additions.

5.2 Design Decisions

The original Nima system consisted of one large single-threaded, single-CPU program (called the Nima root) that implemented all the required methods and attempted to meet all computational and rendering requirements. Figure 3 shows the design of this system in terms of class data, member functions, and data flow. However, this one program was not sufficient to meet all computational requirements, namely those involving reasonable completion time. As a solution to this problem, several auxiliary programs (called Nima branches) were created. These programs are parallel message-passing programs that implement one specific method. The data generated by these programs is passed to the root program via files (future implementations will use more sophisticated communication based on remote procedure calls (RPCs) or sockets (Client-Server model)).

The choice of development language for the root program was C++, since the system was anticipated to be large and the Object-Oriented Paradigm fit well with the problem space and requirement lists. Also contributing to this decision was the fact that the OS environment was chosen to be Windows NT, and future work included a graphical interface which would be MFC-based. The auxiliary parallel programs were developed in C, since they were small, focused on one task, needed to be as efficient as possible, ran in the Linux/Unix OS environment, and used MPI for message-passing. All rendering was done using OpenGL, primarily to increase portability, but also to decrease development time. The trade-off between OpenGL and Direct3D is that Direct3D is integrated into the Windows OS class environment and in many cases allows a more fine-grained control of the rendering pipeline, as well as

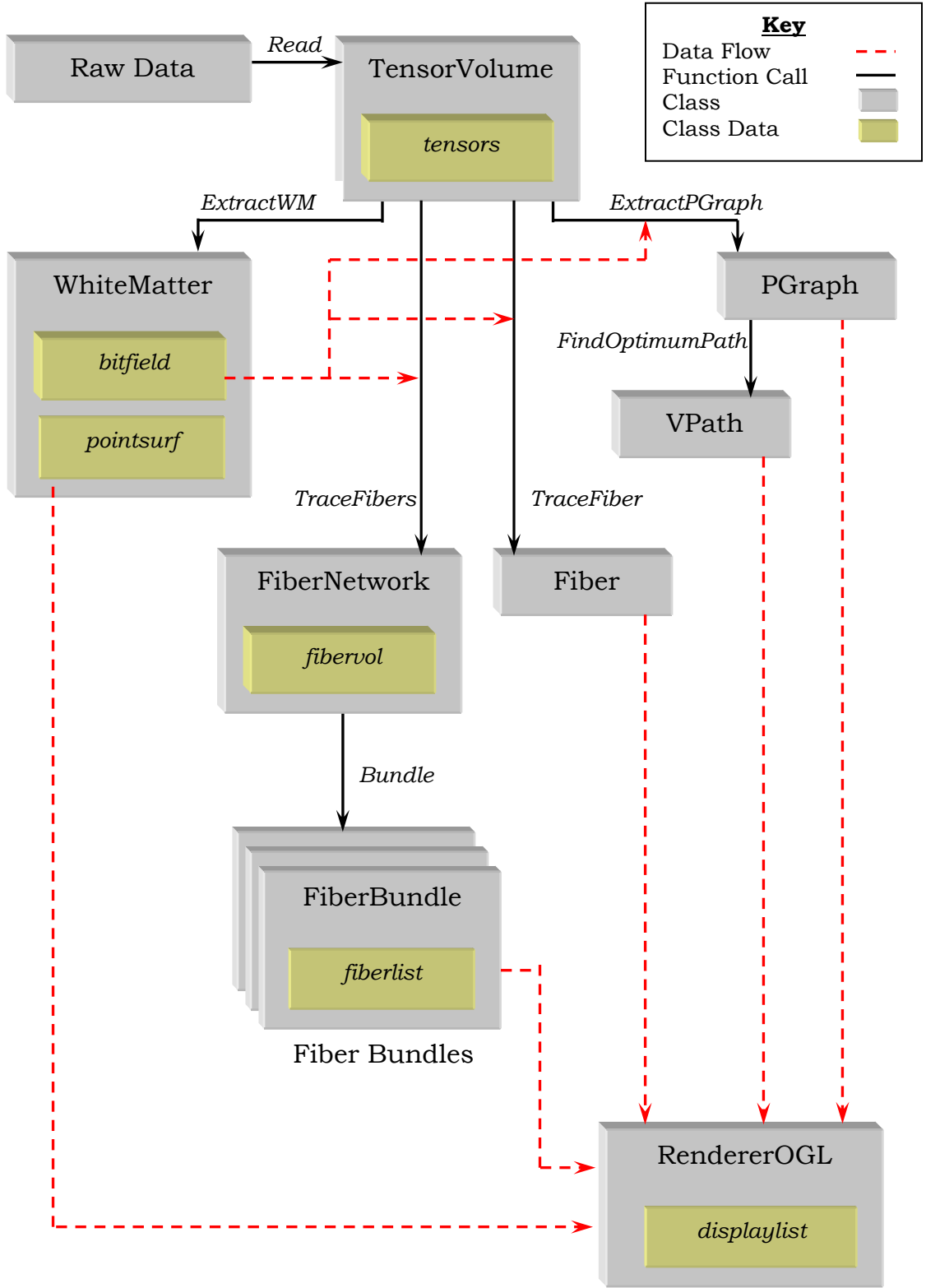


Figure 3: Components of Nima

access to newer graphics card features. Since none of these advantages were necessary in Nima, we chose not to use Direct3D.

5.3 *Class Definitions*

The root program employs several classes in order to accomplish its task of integrated visualization. Only the most important classes will be described here, as there are many secondary classes that encapsulate various low-level data and associated operations. For each class a short synopsis is given, followed by a list of key member functions and their descriptions. When necessary pseudo code will be given to further elucidate the algorithms. Figure 3 shows how the classes work together to produce the complete visualization.

5.3.1 Tensor Class

The Tensor class encapsulates a 3 dimensional symmetric tensor matrix (rank-2) of size 3x3. Operations include:

double A_sigma()

Calculates the anisotropy index (A_σ).

double D_bar()

Calculates the mean diffusion rate ($\bar{D}$).

5.3.2 TensorVolume Class

The TensorVolume class defines a 3-dimensional tensor volume. This class serves as a container for the Tensor class. Operations include:

Tensor InterpolateTensor(Point_3D& pt)

Reconstructs the tensor at the point *pt* using the GwS interpolation kernel with radius 2.0.

*unsigned int ExtractWM(WhiteMatter& wm,
double aSigThd,
Point_3D ptSeed)*

Extracts the white matter surface using the seed point *ptSeed* and the A_σ threshold *aSigThd*. The return value is the number of points extracted, and the points are returned in the *wm* parameter. Psuedo code for this algorithm is given in Figure 4. If no seed point is given then one is generated automatically by testing points near the center of the data volume for A_σ values greater than the threshold. If one is found then a second test must be made to ensure that the point is not just part of a small pocket of false positives. This is done by starting the extraction algorithm and checking to make sure some minimum number of points is extracted (the pockets of false positives are small and do not generate many points). The correct A_σ threshold is constant for data taken using the same DT-MRI routine, and is determined by experimentation on a trial data set of the routine.

```

Unsigned int ExtractWM(WhiteMatter& wm, double aSigThd,
                        Point_3D ptSeed) {
    FloodElement *felem;
    FloodStack fstack;
    BitField touchedpts;

    BitField.ZeroAll();
    felem = new FloodElement(ptSeed, Null_Direction);
    touchedpts.MarkOne(ptSeed);
    fstack.push(felem);

    while (!fstack.IsEmpty()) {
        felem = fstack.pop();
        if ((InterpolateTensor(felem->GetPoint())->A_sigma) < aSigThd) {
            for (all neighbors n of felem->GetPoint()) {
                if (touchedpts.IsZero(n)) {
                    felem = new FloodElement(n, n-felem->GetPoint());
                    touchedpts.MarkOne(n);
                    fstack.push(felem);
                }
            }
        } else {
            wm.AddPoint(InterpolateSurfacePoint(felem->GetPoint(),
                                                  felem->GetPrevPoint(), aSigThd));
        }
    }
    wm.SetBitField(touchedpts);
}

```

Figure 4: Implementation of the *ExtractWM* method

```

unsigned int TraceFiber (Fiber& fib,
                        Point_3D ptSeed,
                        WhiteMatter& wm,
                        Path path,
                        Param_FT& intParams)

```

Integrates a single fiber through the tensor field using 4th order Runge Kutta with adaptive step size. The fiber is traced in two directions from the seed point, as determined by the major eigenvector and its negative at the local TDE (since there is no incoming direction for the first step, Bayesian fiber tracking must begin at the second step). The resultant points are returned in the *fib* parameter. Included with each point is the tangent vector (necessary for rendering), anisotropy index, and mean diffusion rate at that point. The *intParams* parameter is a structure which contains various integration settings, including termination criteria, step size (if not using adaptive step size), and probability function sample rate. If a *path* parameter is given, then the Bayes rule is calculated according to Section 4.3.2, and the end of the path is included in the termination criteria. The *wm* parameter is necessary only if exiting the white matter constitutes a termination criterion.

```

unsigned int TraceFibers(FiberNetwork& fnet,
                        BitField& seedpool,
                        WhiteMatter& wm,
                        Param_FT& intParams)

```

Traces a group of fibers through the tensor volume according to the integration parameters given in *intParams*. The seed points are given by the parameter *seedpool*, which is a 3D bit field having one bit for each voxel. A 1 indicates a seed point. The bit field of seed points is usually defined by taking the bit field representing the white matter (discussed in Section 5.3.3) and performing various operations, such as masking all but a single slice, or taking the top 10% anisotropy. As in *TraceFiber*, the parameter *wm* is used for termination of fibers. The fibers are returned in the parameter *fnet* in a 3D array based on their seed points.

```
void ExtractPGraph(PGraph& pgraph,  

WhiteMatter& wm,  

double problim)
```

Extracts a probability graph for use in the optimum path search. The graph is defined only within the white matter, which is why the *wm* parameter is necessary. The parameter *problim* gives a lower probability limit to voxel-voxel connections in the graph; connections with a probability less than this will not become edges in the graph. Connections with a probability greater than this will become edges in the graph, and a relative probability will be associated with that edge.

5.3.3 WhiteMatter Class

The WhiteMatter class defines the white matter of the brain as given by the DT-MRI data set. The class data consists of instantiations of two basic classes, *ptsurf* (an instance of the PointSurface class) and *bitfield* (an instance of the BitField class). *ptsurf* contains the points and normals that are used to render the white matter surface, and *bitfield* contains a 3D mask of the tensor field which corresponds to the spatial extent of the white matter. In the bit field, a 1 indicates that a point is inside the white matter. Operations include:

bool IsInsideWM(Point_3D pt)

Determines if a given point is inside the white matter, according to the *bitfield* member.

void RenderOGL()

Called by the Renderer_OGL class to render the white matter using OpenGL functionality. In the implementation this operation simply calls the RenderOGL operation of the *ptsurf* member, which then uses vertex arrays to render the points.

5.3.4 Fiber Class

The Fiber class defines a single brain fiber. The data consists of a list of the fiber bundles to which this fiber belongs, the seed point from which it was traced and the points which make up the fiber. As

mentioned in Section 5.3.2, the tangent of each point is also stored for shading calculations. Operations include:

void RenderOGL()

Renders the fiber using the OpenGL interface. The fiber is rendered as a line segment using vertex arrays, but since the definition of a normal to a line is ambiguous, shading cannot be done using the usual OpenGL shading functions. Instead, shading is accomplished by using the illuminated streamlines approach. Basically, we take the plane of normals to the line at a point and choose the normal which is coplanar with the viewing vector or reflection vector. In order to define the plane of normals it is necessary to know the tangent of the line at each point, which explains why it is calculated and stored by the integration routines.

*static double Similarity(Fiber& fibA, Fiber& fibB,
SiEnum simcrit,
double simlim)*

Calculates the similarity between the corresponding segments of the fibers *fibA* and *fibB*, according to the criterion defined by the enumerated value of *simcrit*. The minimum similarity threshold is given by *simlim*; if the calculated similarity is not greater than this value then it is clamped to 0. The return value is the similarity, which ranges from 0 to 1 (1 means the fibers are identical). This function is used to bundle fibers.

5.3.5 FiberBundle Class

The FiberBundle class is a container class for fibers. The data consists of a list of fibers that belong to the bundle and the color of the bundle used for rendering. The only noteworthy operation is RenderOGL, which simply calls the equivalent function on the fibers it contains.

5.3.6 FiberNetwork Class

The FiberNetwork class is another container class for fibers. Whereas the FiberBundle class simply keeps a list of fibers, this class stores fibers according to their seed points. In practice, a group of fibers is integrated with the seed points defined systematically on a grid, which is usually derived from the white matter bit field. The result is a set of fibers arranged in a 3D array according to their seed points. This arrangement gives an organized way to store groups of fibers and facilitates bundling at a later time. Operations include:

```
unsigned int Bundle(FiberBundle **bundles,  
unsigned int K,  
SiEnum simcrit,  
double simlim)
```

Bundles the fibers contained in the fiber network, according to the criterion defined by the enumerated value of *simcrit*. The minimum similarity threshold is given by *simlim*, and the bundling parameter is *K*. The return value is the number of

bundles created, and the bundles are returned as an array in the *bundles* parameter. Pseudo code for this algorithm is given in Figure 5.

unsigned int Filter(unsigned int fsize_{lim})

Filters or removes fibers from the fiber network which contain fewer points than *fsize_{lim}*. This eliminates fibers which do not show enough information to be useful in the visualization, thereby increasing visual comprehension and rendering time. The return value is the number of fibers removed.

5.3.7 PGraph Class

The PGraph class defines a directed 3D graph where the outgoing edges of a node depend on which ingoing edge was taken to arrive at the node. The edges have relative probabilities associated with them, according to Bayes rule of conditional probability. Each node corresponds to a voxel inside the white matter. The graph is stored as a hash table, with each hash table entry containing a pointer to a PGraphNode. The PGraphNode contains a 2-dimensional array of size 26x26, where the rows are referenced by the encoded incoming direction and the columns are referenced by the encoded outgoing direction. To illustrate, consider the incoming direction (0,0,1). This is 1 of 26 possible incoming directions, and would be encoded as 3 (based on an arbitrary 1-1 mapping of vector to scalar). Thus the third row of the array would give for each of the 26 outgoing directions the probability of

```

unsigned int Bundle(FiberBundle **bundles, unsigned int K,
                   SiEnum simcrit, double simlim) {
    double sim[26];
    Fiber fibs[26], Klist[K];
    BundleList blist;
    Bundle *bp;

    for (all fibers f in the fiber network) {
        GetNeighbors(f,fibs);
        for (n=0; n<26; n++)  sim[n] = Similarity(f,fibs[n],simcrit,simlim);
        sort(sim,fibs);
        for (i=0; i<K; i++)  Klist[i] = fibs[i];
        bp = FindLargestBundle(f,Klist);
        if (bp == NULL) {
            bp = new Bundle;
            blist.AddBundle(bp);
            bp.AddFibers(Klist);
            bp.AddFiber(f);
        } else {
            for (i=0; i<K; i++) {
                if (Klist[i].GetBundle() != bp) bp.AddFiber(Klist[i]);
                if (f.GetBundle() != bp) bp.AddFiber(f);
            }
        }
    }
    ConvertBundleListToArray(blist,bundles);
}

```

Figure 5: Implementation of the *Bundle* method

a fiber following that direction, given the incoming direction (0,0,1). Say we want to follow the direction of maximum probability, and element in column 8 contains the highest value (of all elements in the 3rd row). We use the inverse of the vector to scalar mapping to decode 8 as direction (0,1,0). Then given incoming direction (0,0,1), element (3,8) of the 2D array gives the probability of taking outgoing direction (0,1,0).

Operations on the PGraph class include:

```
bool FindOptimumPath(VPath& optpath,
                     Voxel startvox,
                     Voxel endvox,
                     PwEnum wtfunc,
                     unsigned int pathlim)
```

Calculates the optimum path (which is returned in the parameter *optpath*) connecting *startvox* and *endvox* by an adaptive dynamic programming method using a hash table. The weight function is given by the enumerated value of *wtfunc*, and the limit of the length of the path searched for is given by *pathlim*. The return value is true if an optimum path was found and false if no path was found. The algorithm implemented by this method is shown in Figure 6.

```
void RenderOGL()
```

Renders the graph in OpenGL. The graph can be rendered in two different ways. The first way renders nodes as points and edges as lines connecting the points. Only the outgoing directions for one incoming direction are shown at a time, with

```

bool FindOptimumPath(VPath& optpath, Voxel startvox, Voxel endvox,
                    PwEnum wtfunc, unsigned int pathlim) {
    SetWtFunc(wtfunc);
    optpath = OptPath(startvox,endvox,startvox,endvox,-1,pathlim);
}

VPath OptPath(Voxel startvox, Voxel endvox, Voxel openvox,
              Voxel closevox, Voxel prevdir, unsigned int pathlim) {
    VPath *path, paths[26];
    Voxel Nbr[26];

    if ((pathlim == 0) || (endvox == GRAYMATTER) || (endvox == openvox)) {
        return NULL_PATH;
    }

    if (dynhashtable.Find(startvox,endvox,prevdir,path)) return &path;
    if (AreNeighbors(startvox,endvox) && AreConnected(startvox,endvox)) {
        path = new VPath;
        path = SetPathFromTo(startvox,endvox);
        dynhashtable.Insert(startvox,endvox,prevdir,path);
        return &path;
    }

    FindNeighbors(startvox, Nbr);
    for (all neighbors n of Nbr) {
        paths[n] = OptPath(startvox,Nbr[n],openvox,closevox,prevdir,1) +
                  OptPath(Nbr[n],endvox,openvox,closevox,startvox,pathlim-1);
    }

    return MinimumCostPath(paths);
}

```

Figure 6: Implementation of the *FindOptimumPath* method

interactive controls to toggle through the incoming directions. The second way renders only the nodes, which are shown as transparent blocks, in order to show only the extent of the graph. Rendering the graph gives contextual information when viewing the optimum path.

5.3.8 VPath Class

The VPath class defines a discrete path in 3D, and consists of an ordered list of voxels. It usually contains an optimum path computed from the PGraph. The only notable operation is RenderOGL, in which the path is rendered as a series of blocks. A reference to an instantiation of this class may be passed into the fiber tracking method of TensorVolume in order to obtain fibers that travel through the path.

5.3.9 RendererOGL Class

The RendererOGL class defines an OpenGL renderer. Its data consists of various rendering settings, as well as lists of registered objects. In normal operation of the Nima root program, the computational jobs are batched together at the beginning, thereby generating the data to be rendered. Then the objects which contain this data register themselves with the renderer. Following this the renderer enters the display loop in which the image is generated and interaction with the user can take place. A number of interfaces are defined in relation to the display loop callbacks. The most important is the RenderableObject interface. In order for an object to be rendered, it must implement this interface (by inheriting from the abstract class

RenderableObjectInterface and implementing it's one method, RenderOGL). It can then add itself to the renderer's RenderableObjectList, which we refer to as registration. The renderer calls the RenderOGL method on all objects in this list inside its display routine. Similar interfaces are defined for keyboard and mouse interaction, if an object wants to respond to these events. Future implementations incorporating a GUI may separate the renderer and the user interaction.

5.4 Parallel Implementations

Two of the four methods were implemented in parallel, namely the white matter extraction and Bayesian fiber tracking. The other methods, optimum path finding and fiber bundling, complete in a reasonable amount of time (less than 5 minutes). The original implementation of the white matter extraction took about 30 minutes to generate 2.25 million points on a 64x64x18 data set. This was judged unreasonable, and so a parallel implementation was devised.

Streamline tracing is usually a very efficient process; however, Bayesian fiber tracking is not. This is due to three key differences: first, reconstruction using the GwS filter is much more costly to compute than trilinear interpolation; second, integration using 4th order Runge Kutta is far more expensive than the 1st order Euler's method; and third, interpolating the tensor, computing the DTE, and taking 1600 samples of the probability function (which must be done at each step) is orders of magnitude slower than simply interpolating a vector. For these reasons, integration of large amounts of fibers takes an unreasonable amount of time, and so a parallel implementation was developed.

5.4.1 White Matter Extraction in Parallel

The algorithm used to extract white matter in the root program does not lend itself to a parallel implementation, since the process of flooding requires knowledge of the current front at all times. For reasons stated in Section 4.1, it is necessary to use the *Dividing Cubes* algorithm to discover the surface. This method finds the surface, but also picks up pockets of false surfaces outside the white matter, and so once the points are generated and collected a flooding algorithm is used to eliminate these extraneous points. This flooding algorithm runs very quickly, since the points are already discovered and stored according to the voxel in which they occur. Almost equal load balance is achieved using a round-robin allocation of voxels, as the points are not uniformly distributed in space. Each processor must have a copy of the tensor volume.

5.4.2 Bayesian Fiber Tracking in Parallel

The algorithm used to track multiple fibers in the root program is directly parallelizable. Each processor is given a subset of the seed points, the tensor volume, the integration parameters, and the white matter surface (if necessary). The seed points are again distributed in a round-robin manner, since some fibers will be longer than others, usually in a non-predictable way.

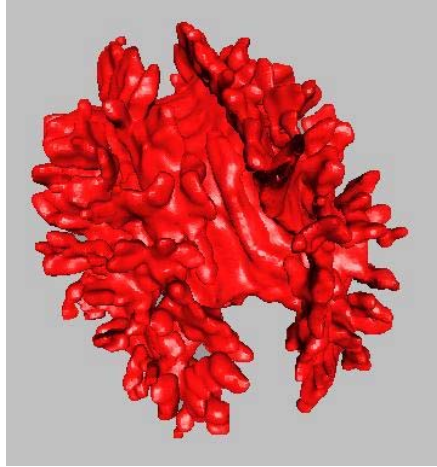
6. Results and Discussion

In this section we present the results of each technique developed in Section 4 obtained using the implementation described in Section 5, as well as examples of integrated visualization incorporating these techniques. As mentioned in Section 2.1, our main suite of test data is of 64x64x18 resolution, with both DT-MRI and fMRI measurements. Each voxel corresponds to a physical size of 3x3x5 mm³.

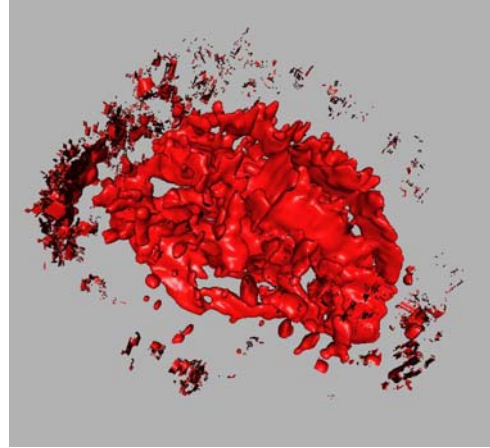
6.1 *White Matter Surface Extraction*

Figure 7 shows a point surface of white matter extracted from DT-MRI with dividing cubes. This model, generated using an A_c threshold of 0.22, consists of 2.25 million points. Every voxel was divided into 8x8x8 subvoxels. Our parallel implementation using round robin data distribution completes the whole process in 4 minutes on 30 2.4 GHz Pentium 4 processors.

The University of Utah graciously provided us with a DT-MRI dataset of resolution 148x190x160, without an associated fMRI dataset, however. We successfully extracted a surface of the white matter from this dataset as well (Figure 8), using 2x2x2 subdivision in each voxel. The result is very similar to that shown in Figure 7, but with significantly more details available.



(a)



(b)

Figure 7: White Matter Surface Extraction. (a) The point surface of 2.25 million points, extracted from a DT-MRI dataset (b) The result of dividing cube before the extraneous structures are removed

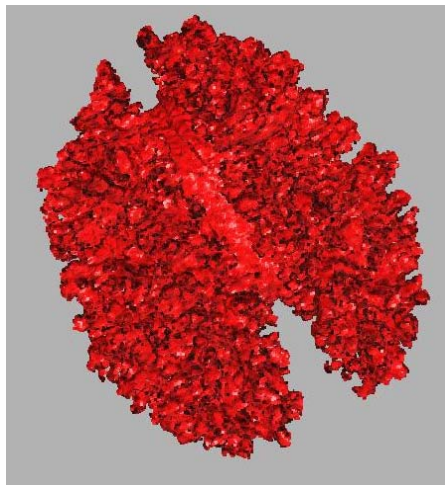


Figure 8: White Matter Surface (high resolution). A white matter surface, composed of one million points, extracted from the Utah DT-MRI dataset using an A_0 threshold of 0.25

6.2 *Bayesian Fiber Tracking*

In Figure 9, we demonstrate the differences in fiber reconstruction between Bayesian fiber tracking and vector fiber tracking. All tracking is performed without noise filtering. The fibers reconstructed by reducing the tensor field to a vector field are shown in blue, as opposed to red colored Bayesian tracking results. The yellow rectangle shows the region where seed points are located. All fibers are integrated using 4th order Runge Kutta with step sizes adaptively adjusted according to local diffusion rates.

In regions of high anisotropy, the fibers reconstructed using Bayesian tracking and vector based tracking are very similar. However, in the case of vector-based fibers, the combined effects of noise and depolarization produce increasingly inconsistent paths as the tracing progresses. This can be observed most easily in long fibers and is manifested as dramatic direction changes, doubling back and loops. In general, Bayesian fiber tracking shows more stable behavior at all locations within the white matter.

The overhead of Bayesian tracking is rather high, due to the repeated evaluation of Bayes rule and the GwS interpolation filter we used. In Figure 10 we constructed 50,000 fibers, which took 10 minutes using 20 2.4 GHz processors. The step size in the numerical integration was 1.0 mm.

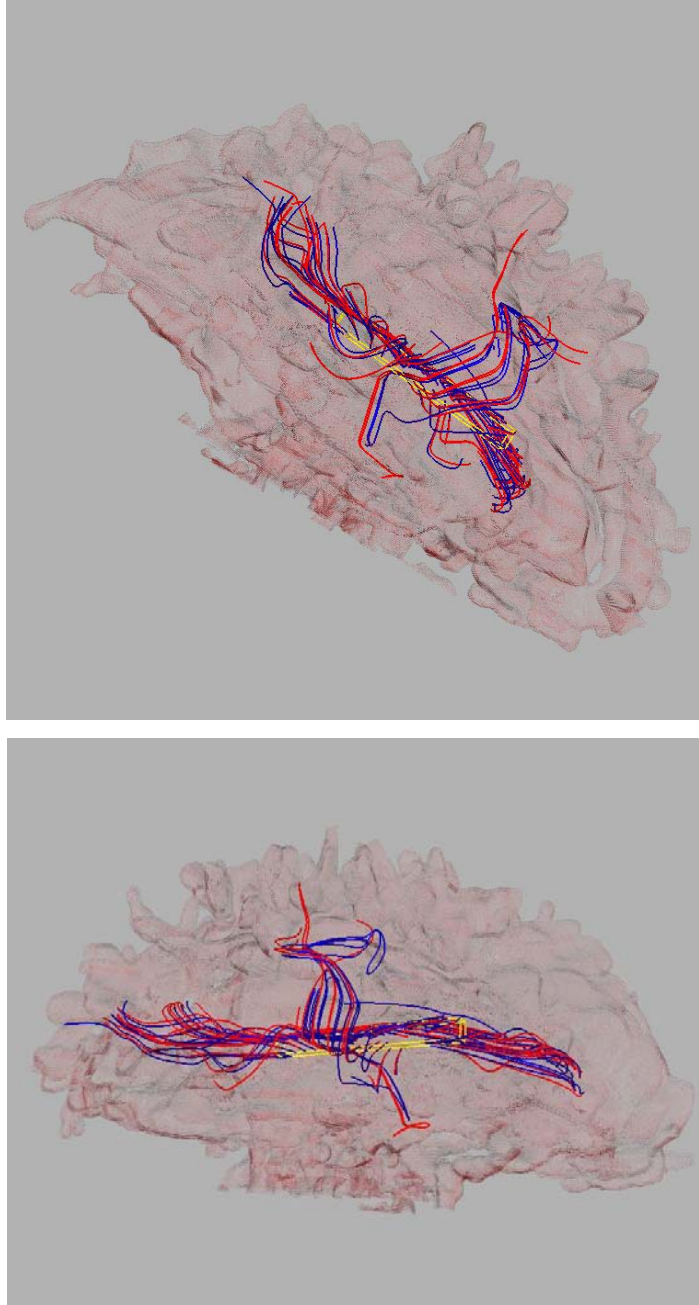


Figure 9: Fiber Tracking Results. Comparisons between Bayesian fiber tracking (red) and vector streamline tracking (blue). Two different view angles are shown.

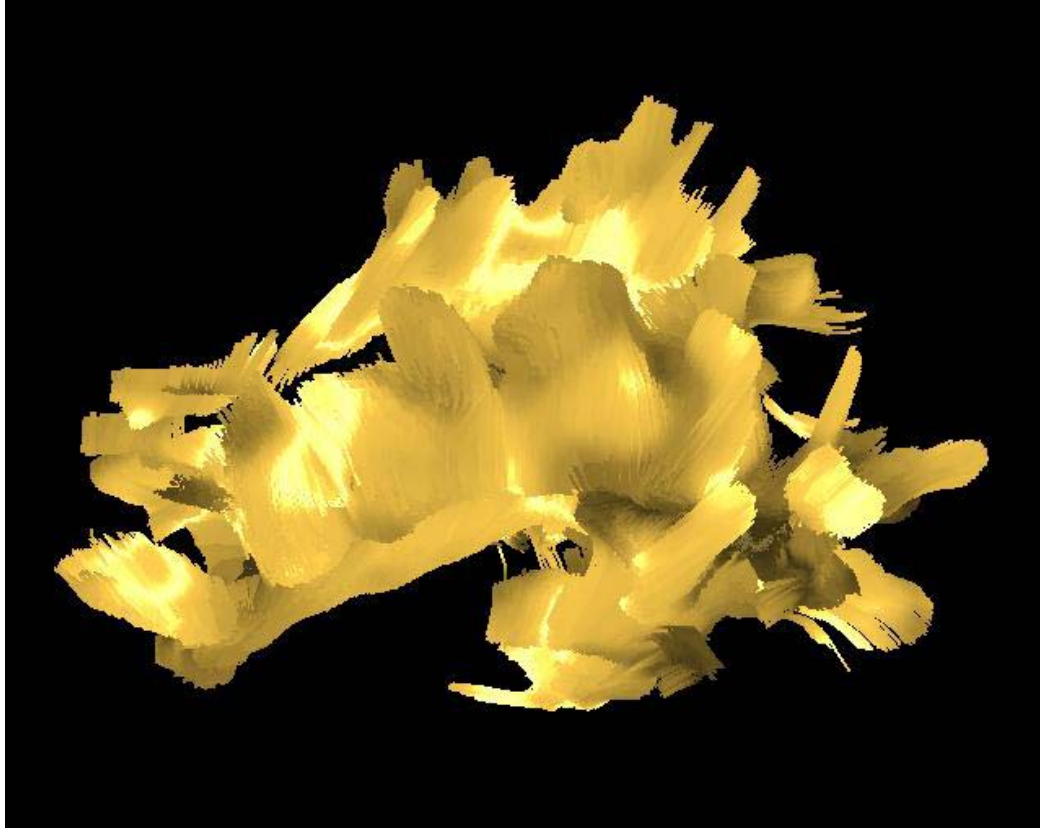


Figure 10: Large Scale Fiber Integration. 50,000 fibers constructed using the parallel implementation of Bayesian fiber tracking.

6.3 Fiber Bundling

Bundling is a rather efficient process. The 50,000 nerve fibers in Figure 10 were bundled into about 200 groups (shown in Figure 11) within one minute using a single 1.7 GHz P4 processor. To reveal some detailed white matter structures, in Figure 12 we show 10,000 fibers grouped into 45 color-coded bundles. The value of K used is four, with the similarity cutoff threshold set to 0.1.

6.4 Globally Optimal Pathways

Here we present the results of our procedures to construct globally optimal neural pathways (see Figure 13). To provide contextual information the white matter is delineated in gray semi-transparent cubes, and four regions on the cerebral cortex are rendered as solid cubes in orange. The query for an optimal pathway is specified with the S (green cube) and D (red cube) locations. Using dynamic programming, the optimal pathway is constructed and rendered as yellow cubes. Through this sequence of voxels depicting the optimal pathway, a bundle of nerve fibers was then constructed. During the process of dynamic programming to solve the query in Figure 13, 75,000 calls to sub problems were made, of which only 20% were calls to a new sub problem. The whole process took less than a minute to complete on a single 1.7 GHz P4 CPU.

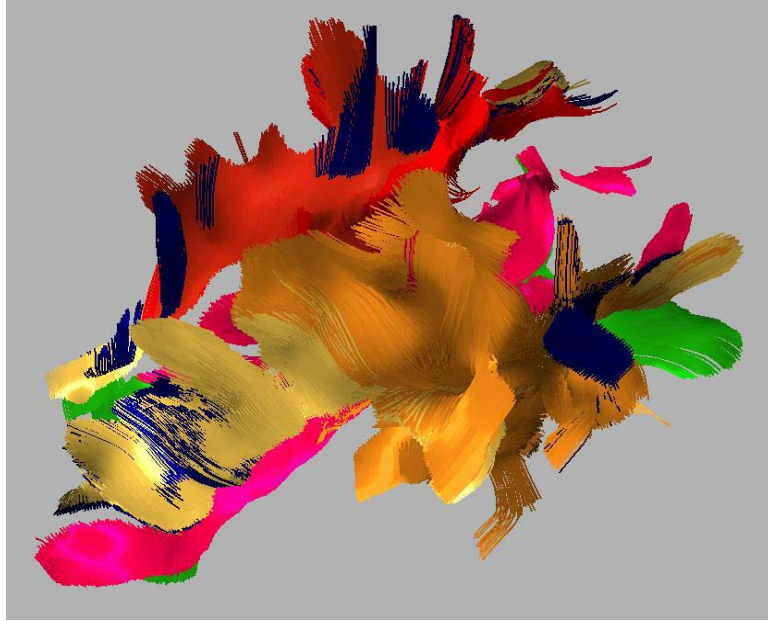


Figure 11: Large Scale Fiber Bundling. 50,000 fibers bundled into about 200 groups

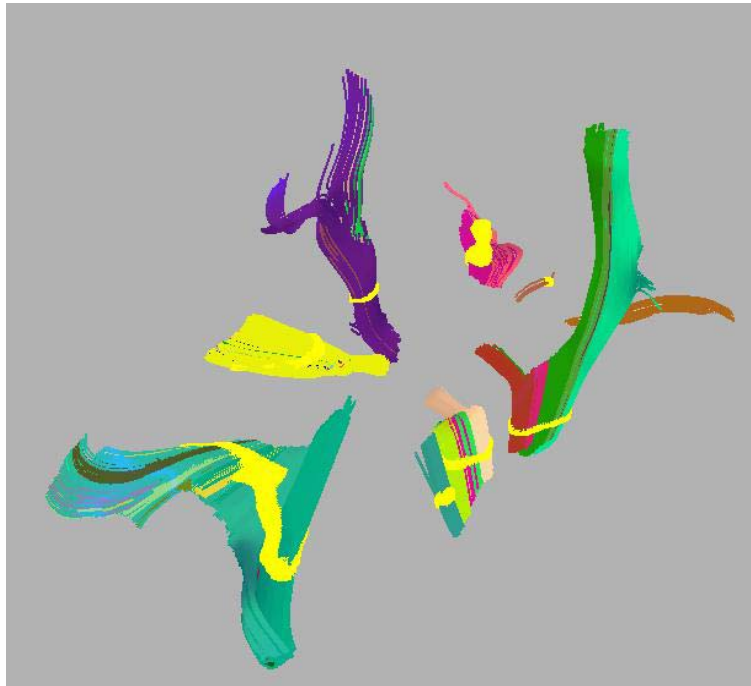


Figure 12: K-most-similar Bundling. 10,000 fibers grouped into 45 bundles

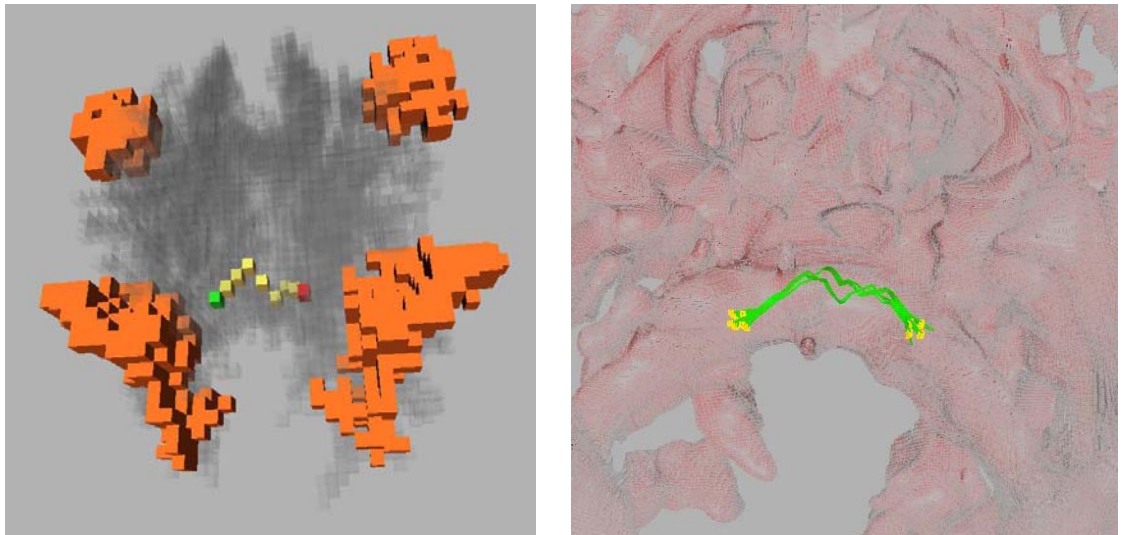


Figure 13: Optimal Pathway Search. (Left) The optimal pathway reconstructed, queried with two end voxels (red and green cubes) and (Right) the optimal nerve bundles traced within the optimal path. The yellow dots are the end points on each resulting fiber.

6.5 *Integrated Visualization*

Using the techniques developed in this paper, we were able to query the DT-MRI dataset with any number of ROIs from fMRI. In Figure 14 we show one example of visualizing fMRI ROIs supported by DT-MRI. Although all definitions of cost, or optimality metric, can be used in our approach, we do not know which one truly matches real nerve connections. The result in Figure 14 was obtained using diffusion rate weighted shortest path as the metric of optimality. Of course, the initial cost on each link connecting neighboring voxels is weighted by the inverse of the probability of such a connection's existence, based on Bayes rule, to penalize low probability connections. According to our result, not all four ROIs are directly connected.

The computation time required by the dynamic programming process depends on the number of calls to sub problems made, and therefore, the length of the optimal path being constructed. For instance, the result in Figure 14 took less than 10 minutes, where about one million calls to sub problems in total were made during the process. Fortunately, more than 90% of those calls were to sub problems already solved in previous steps.

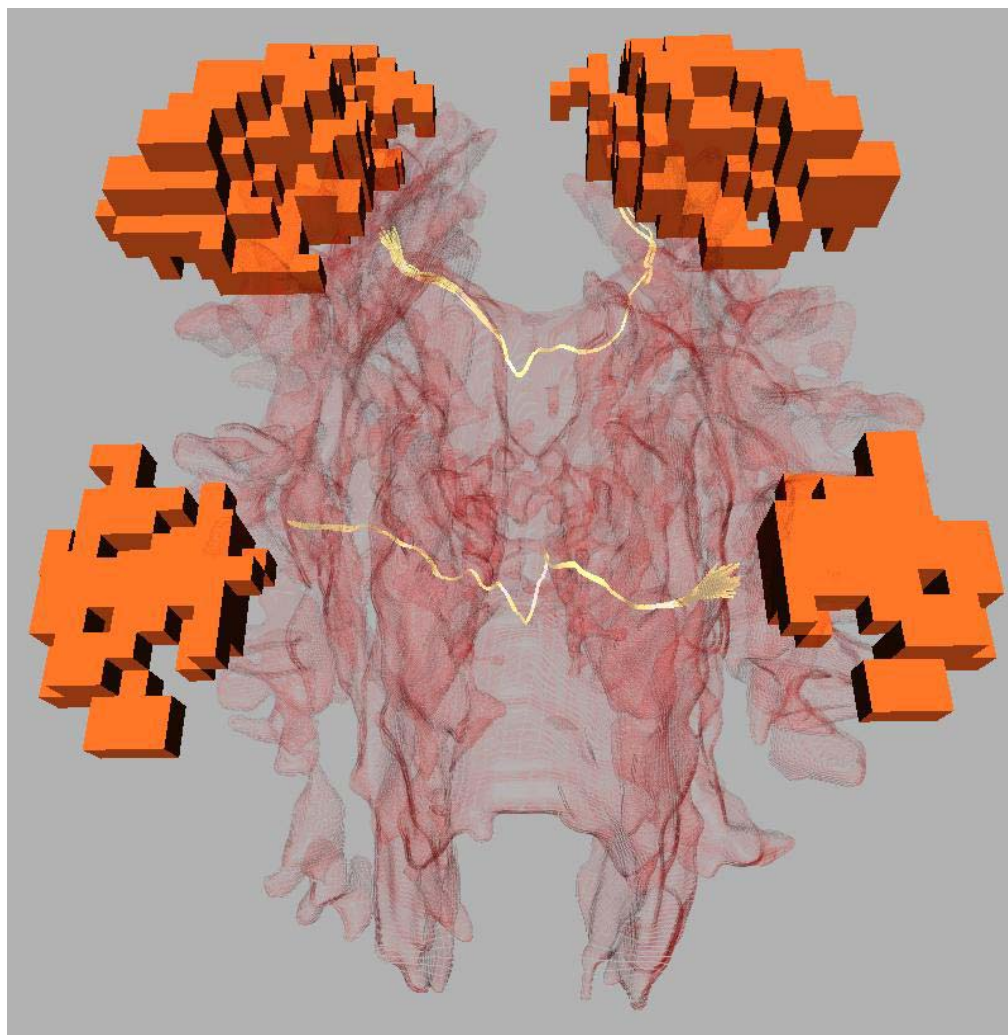


Figure 14: Integrated Visualization. The optimal fiber bundles (yellow) reconstructed, between four 3D ROIs (solid orange blocks) on the cerebral cortex, shown within semitransparent white matter (red).

7. Conclusions

We have proposed a framework in which diffusion tensor MRI datasets can be modeled for the purpose of integrated visualization with functional MRI. The core contribution revolves around a rigorous mathematical framework based on applying Bayes rule of conditional probability to conduct fiber tracking within a tensor field directly and the introduction of the paradigm of dynamic programming to globally optimal pathway reconstruction. In addition to this, several supporting techniques, such as proper methods of reconstruction in tensor fields and discovery of white matter boundaries have also been studied. Finally, our previous bundling method is tested and shown to scale up efficiently to a large number of fibers.

Future research on integrated visualization of multiple imaging modalities has significant potentials, as more imaging modalities are now being developed and perfected. Many of such problems are just emerging, and so the demand for integrated visualization will continue to increase.

List of References

List of References

1. A. Konig, H. Doleisch, and E. Groller. *Multiple Views and Magic Mirrors - fMRI Visualization of the Human Brain*. in *Proceedings of Spring Conference on Computer Graphics and its Applications*. 1999. Budmerice, Slovakia.
2. Bassler, P.J., J. Mattiello, and D. Le Bihan, *MR diffusion tensor spectroscopy and imaging*. Biophysics Journal, 1994. **66**: p. 259-267.
3. Bassler, P.J., et al., *In vivo fiber tractography using DT-MRI Data*. Magn Reson Med, 2000. **44**(625-632).
4. Cline, H., et al., *Two algorithms for the three-dimensional reconstruction of tomograms*. Medical Physics, 1988. **15**(3): p. pp. 320-327.
5. Cormen, T., C. Leiserson, and R. Rivest, *Introduction to Algorithms*. 1990: The MIT Press; McGraw-Hill Book Company. ISBN 0-262-03141-8
6. Delmarcelle, T. and L. Hesselink, *Visualizing second-order tensor fields with hyper streamlines*. IEEE Computer Graphics and Applications, 1993: p. 25-33.
7. Ding, Z., J.C. Gore, and A. Anderson. *Case study: reconstruction, visualization, and quantification of neuronal fiber pathways*. in *Proc IEEE Visualization*. 2001.
8. Heeger, D.J. and D. Ress, *What does fMRI tell us about neuronal activity?* Nature Reviews Neuroscience, 2002. **3**: p. 142-151.
9. Kerlick, G.D. *Moving iconic objects in scientific visualization*. in *Proc. of IEEE Visualization*. 1990.
10. Kindlmann, G. and D. Weinstein. *Hue-balls and lit tensors for direct volume rendering of diffusion tensor fields*. in *IEEE Visualization Conference*. 1999.
11. Laidlaw, D.H., et al. *Visualizing Diffusion Tensor Images of the Mouse Spinal Cord*. in *Proceedings of IEEE Visualization*. 1998. Research Triangle Park, North Carolina.
12. Lorensen, W. and H. Cline. *Marching Cubes: a high resolution 3D surface construction algorithm*. in *Proc. of SIGGRAPH*. 1987.
13. Poupon, C., et al., *Regularization of diffusion-based direction maps for the tracking of brain white matter fascicles*. NeuroImage, 2000. **12**: p. 184-195.

14. Shimony, J., et al., *Quantitative Diffusion-Tensor Anisotropy Brain MR Imaging: Normative Human Data and Anatomic Analysis*. Radiology, 1999. **212**: p. 770-784.
15. Walpole, R., R. Myers, and S. Myers, *Probability and statistics for engineers and scientists*. 1998: Prentice Hall. ISBN 0-13-840208-6
16. Weinstein, D., G. Kindlmann, and E. Lundberg. *Tensorlines: Advection-Diffusion based Propagation through Diffusion Tensor Fields*. in *Proc. of IEEE Visualization Conference*. 1999. San Francisco, CA.
17. Welsh, T., et al. *Graphical strategies to convey functional relationships in the human brain: a case study*. in *Proc. of IEEE Visualization*. 2001. San Diego, CA.
18. Zhukov, L. and A. Barr. *Oriented Tensor Reconstruction: Tracing Neural Pathways from Diffusion Tensor MRI*. in *Proceedings IEEE Visualization*. 2002. Boston, MA.
19. Zhukov, L., et al. *Tensor Invariants for Modeling and Segmentation of Diffusion Weighted MRI Data*. in *SPIE Medical Imaging*. 2001.
20. R. Dickinson. *A Unified Approach to the Design of Visualization Software for the Analysis of Field Problems*. in *SPIE Medical Imaging*. 1989.

VITA

In the summer of 1996 Nathaniel Fout received a four-year academic scholarship to the University of Tennessee, and in the fall of that year began attending. In 1999 he joined Dow Chemical as an alternating term coop (ATC), and in 2000 completed his three terms. While at Dow, he became the first student in the ATC program to receive a Special Recognition Award. In the summer of 2002 he graduated Summa Cum Laude with a Bachelor of Science in Chemical Engineering. He also received minors in Chemistry and Computer Science.

Following graduation he immediately entered graduate school in Computer Science at the University of Tennessee. That same fall he joined the Visualization Group, which was at that time just beginning. At the present time he continues to work as a graduate researcher for the Visualization Group. After graduation he plans to join the Ph.D. program in Computer Science at the University of California in Davis.