A Field Data Structure for Improved Interactive Exploration of Scientific Data Sets

Burkhard Wünsche
Department of Computer Science
The University of Auckland, Private Bag 92019
Auckland, New Zealand
Email: burkhard@cs.auckland.ac.nz

Abstract

Scientific data sets frequently consists of many multi-dimensional variables. Visualizing such data enables the user to understand and interpret the data. In many instances, however, visualizing multiple individual variables does not yield sufficient information and hidden structures can be only be revealed by deriving new measures from the existing variables. The derivation of new measures is complicated by the fact that input data may exist in various forms, such as discrete data at sample points, analytic data (in form of a mathematical function) and finite element data in which variables can be defined in either world or material coordinates.

This paper proposes a new data structure and accompanying user interface which enables the scientist to interactively derive new measures from the underlying data set. The data structure is efficient in the sense that data values are only evaluated at positions where they are needed. The capabilities of our data structure are demonstrated using as a case study the extraction of the nerve fiber structure of the brain from diffusion tensor data.

Keywords: Scientific visualization, data transformation, scalar fields, vector fields, tensor fields

1 Introduction

Technological advances over the past decade have enabled scientist to create ever larger and more complex sets scientific data. Examples are advanced numerical simulations, satellite measurements and medical imaging data. Consequently it has become increasingly difficult to understand, analyse and communicate the resulting data sets. Scientific Visualization is an attempt to simplify these tasks according to the motto "An image says more than a thousand words". Representing scientific data as an image improves the perception of features and pattern in the data, facilitates the navigation through and interaction with complex and disparate sets of data and improves the communication of scientific results with peers and the wider community. The visualization process can be divided into three stages: The *data transformation* stage converts raw data into a form more suitable for visualization. The subsequent *visualization mapping* converts the raw data into a number of graphical entities (*visualization icons*) which are displayed in the final *rendering* stage. The following paragraphs formally define the term *multi-dimensional data* and explain how such data can be transformed.

A multidimensional data set L_m^n consists of m *independent variables* representing the data domain and n *dependent variables* defined over the domain. In most applications the independent variables define a two or three dimensional spatial domain and the data set is simply called 2D and 3D data, respectively. An additional independent variable can be introduced by considering time. Both dependent and independent variables can be either *discrete* or *continuous* and can have a *finite* or an *infinite* range of values. Common examples for dependent variables are scalar fields ($n = 1$) such as temperature, vector fields ($n = 3$) such as velocity, and symmetric tensor fields ($n = 6$) such as stress. Many scientific data sets consist of multiple fields defined over the same domain resulting in a high-order space of dependent variables.

The data transformation step can refer to both the independent and the dependent variable. The independent variable can be transformed using, e.g., interpolation, sampling, and projections. The dependent variable can be transformed by reducing, modifying or expanding it.

Examples for data reduction techniques are the computation of vector magnitude, eigenvalues, eigenvectors and components. Other applicable operators include the dot product, matrix determinant, evaluating surface curvature and distance metrics. Data expansion is achieved using a

gradient operator for scalars and a Jacobian for vector data. Both of these operators give neighbourhood information at a point and can be utilized to detect local features in a data set (e.g., extrema, ridges). A more complete listing of data transformation techniques can be found in [7].

In order to explore a data set efficiently and effectively it is desirable to interactively choose and create new transformations. For example, the user might want to compare the difference in velocity magnitude or flow direction of two velocity fields. In this paper we present a data structure and user interface designed for that purpose.

Previous work related to this area seems to be extremely limited. Bryson et al. [3] describe a *Field Encapsulation Library* which provides a grid independent interface to gridded three dimensional field data. Moran and Henze [4] and [8] suggest a tool where new fields can be created from existing fields using a list of predefined operators. The computation is efficient in the sense that the new field is only computed at data point where it is needed. However, only fields defined over the same grid can be combined. Our work allows the combination of fields from different domains and permits a more powerful variety of derived fields. Furthermore, since we don't interpolate sample values but use the representation of the underlying source field, the resulting field values are more accurate.

The next section explains our field data structure and is followed by a presentation of the accompanying user interface. We conclude with a case study demonstrating the capabilities of our data structure.

2 The Field Data Structure

In order to unify different field representations we require that the domain of the entire data set (the *model*) is a *finite element* mesh. This requirement was motivated by our work in bioengioneering where finite element models are common. Employing the material coordinate system of a FE model leads to more efficient and accurate visualizations [7]. Note that this prerequisite does not limit the range of input data sets since, for example, the domain of a mathematical function or MRI data set can be represented by a single trilinear interpolated cuboidal element enclosing the region of interest.

The geometry of a *finite element* (FE) model is defined by coordinates and interpolation functions. This is achieved by first specifying a parent element (figure 1 (a)), which in 2D is a square in ξ-parameter space. The coordinates ξ_i ($0 \leq \xi_1, \xi_2 \leq 1$) are called the element or *material coordinates*. The value of some variable u (e.g., temperature) at the material coordinates ξ is then specified by interpolating the variables u_i and, depending on the degree of the interpolation, its derivatives in ξ_j-direction $\left(\frac{\partial u}{\partial \xi_j}\right)_2$ ($i = 1, \ldots, 4;\ j = 1, 2$) at the element nodes. The geometry of an element in world coordinates (figure 1 (b)) is then obtained by specifying the world-coordinates v_i and the ξ_j-tangents $\left(\frac{\partial v}{\partial \xi_j}\right)_i$ ($i = 0, \ldots, 4;\ j = 1, 2$) of the element vertices and interpolating them.

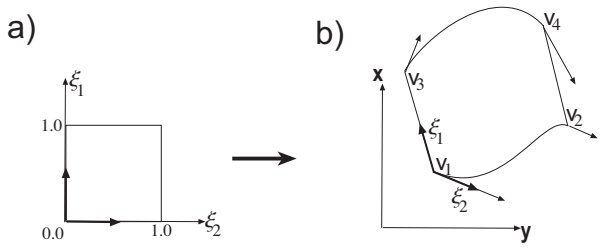

Figure 1. A Biubic Finite Element.

When combining fields specified in different coordinate systems (ie. world coordinates (x,y,z) or different material coordinates) a mapping between coordinate system must be accomplished. The mapping from material to world coordinates is achieved by using the finite element interpolation. The reverse mapping requires a multi-dimensional Newton method. We found that 3 iterations are usually enough to find a material point inside an element for a given point in world coordinates.

This theoretical background allows us now to define fields with differently defined domains and to combine them using various operators. Our data structure consists of an abstract field class that is first subclassed into a (symmetric) tensor field, a vector field, a scalar field, and a general n-d field class as shown in figure 2. These classes contain attributes and methods common to their subclasses. For example, for every symmetric tensor field the algorithm to compute eigenvectors and eigenvalues is identical, and for every vector field it must be distinguished whether it is signed or unsigned. All of these classes are then subclassed into *defined fields*, *derived fields*, *analytic fields*, or *expression fields*.

A defined field is associate with a sampling grid and a set of interpolation functions. The interpolation functions chosen for a derived field depend on the spatial variation and continuity requirements of the field. In particular the interpolation functions for derived fields are not necessarily the same as the ones used to interpolate the geometry of the underlying model.

A derived field is associated with a parent field and contains a function specifying how a field value is derived from the corresponding parent field value. As an example consider an eigenvalue field which has a tensor field as a parent. The eigenvalue field contains a link to the associated tensor field, a variable specifying whether the major, medium, or minor eigenvalue is selected and a method to compute the

eigenvalue at a point. Other examples of derived fields are eigenvector fields (major, medium, or minor), vector length fields, vector angle fields (specifying the angle with any of the world or material axes), gradient fields, and vector and tensor component fields. For most FE models the user is interested in the components of a tensor with respect to the material coordinate system of the model so that a basis transformation is performed if the tensor is defined with respect to a different coordinate system.

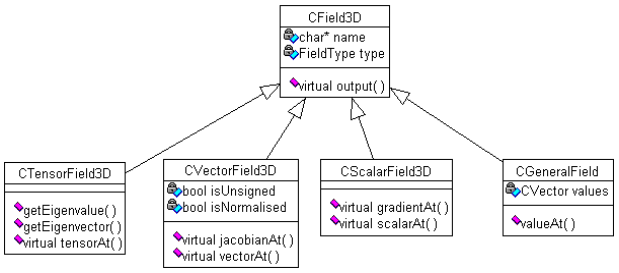

Figure 2. Top-Level class diagram of the field data structure.

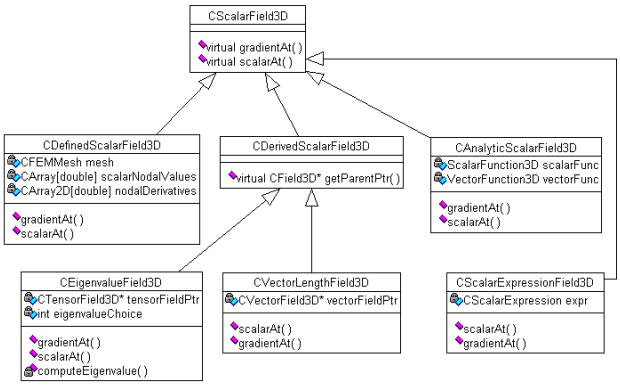

Figure 3. Class diagram of (a subset of) the scalar field data structure.

An analytic field is specified by an algebraic function defined over a domain in world coordinates or element coordinates. This type of field proves useful when creating test cases for our visualization algorithms and can be used in applications where the analytic solution to a problem is known.

Finally an expression field contains an arithmetic expression tree where the leaves are numeric constants or are fields themselves.

Figure 3 demonstrates the subclassing of the field data structure in figure 2 by showing a subset of the scalar field class hierarchy. Note that the computation of the gradient function is implemented in subclasses since the most suitable computational method depends on the type of field. For a regular trilinearly-interpolated sample grid finite differences can be employed, for analytic functions a numerical approximation can be used and for higher-order finite element meshes the derivatives of the interpolation functions can be used to get the coordinate derivatives of the field.

The advantages our field data structure are threefold:

- we eliminate problems with the interpolation of derived values. For example, directly interpolating the eigenvalues of a tensor over a finite element gives usually the wrong results. Instead we rather interpolate the tensor and compute the eigenvalues from the resulting tensor.
- we can combine arbitrary fields through arithmetic functions (e.g., the difference between two scalar fields) even if they are defined over different grids. Similarly, we can interactively derive new fields by choosing a parent field for a derived fields.
- No additional sample errors are introduced as would happen, for example, if sampling an analytic field in order to create a new field over a fixed grid structure.
- Entities defined over a finite element grid can be represented with respect of either the world coordinates or the material coordinates. This choice of representation increases the power of the visualization (see [7]).

The disadvantage of the described field structure is that

- the computation of a derived field value is slower than if the field values were precomputed at sample points.

3 Graphical User Interface

The graphical user interface for our field creation tool was implemented using FLTK, a LGPL'd C++ graphical user interface toolkit for X (UNIX), OpenGL, and WIN32. It forms part of a visualization toolkit we designed for visualizing biomedical datasets and models [7].

Figure 4 shows on the left the user interface used for controlling the fields and visualization of a model. The right hand side of the figure shows the user interface for creating a new field. The three output text components on the top list the currently defined scalar, vector and tensor fields. The user can create a new field by inputting a simple mathematical expression. Currently an expression can contain the following components:

Scalar field expressions for selecting eigenvalues and components of tensors, components of vectors, numerical constants, binary operators (+,-,*,/,^), unary operators (sin, cos,...), vector length, trace, angle of a vector with x,y,z,ξ_1,ξ_2, or ξ_3-coordinate axis.

Vector field Expressions for selecting eigenvectors of a tensor, gradient of a scalar field, binary operations (+,-,*), vector constants.

Tensor field binary operations (+,-,*), tensor constants.

We have also implemented a conditional expression `switch(<cond1>:field1;...;default:fieldN)`. Currently the conditions are restricted to boolean expressions containing scalar fields and comparison operators only. The next section demonstrates how such an expression can be used for the visual segmentation of an image.

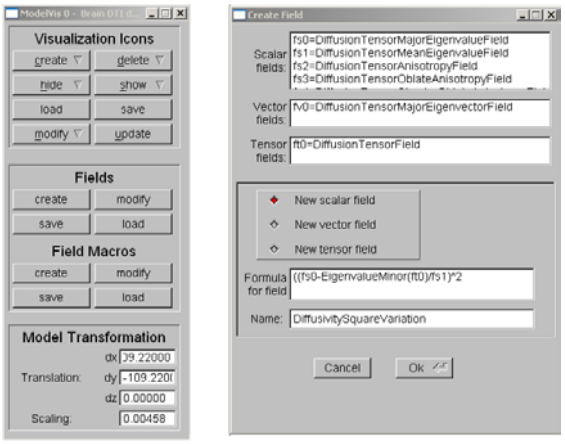

Figure 4. The control window for a model visualization (left) and the user interface for creating new fields (right).

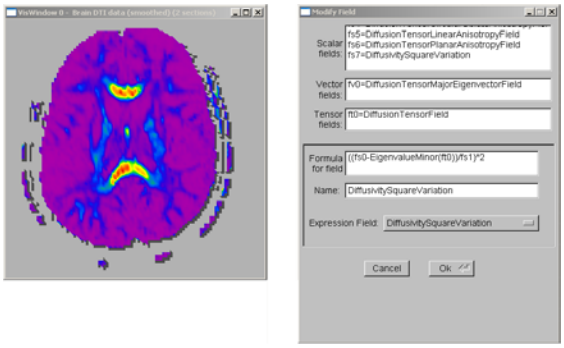

Figure 5. The visualization of the field defined in figure 4 (left) and the user interface used to edit an expression field (right).

The left part of figure 5 shows a visualization obtained by using the field in figure 4. If the user is not satisfied with the result the expression field can be edited in the modification window shown on the right of figure 5. Using the update button of the model control (figure 4 left) the user can recompute any visualization icons dependent on that field.

Expression fields also offer a convenient way to create visualizations for multiple versions of a field **F**. In order to do this define a new field **E** equal to one version of the field **F** and use **E** to derive other fields which are then visualized. If we want to visualize a different version of **F**, say **F'**, it's sufficient to set **E** equal to **F'** and to update all visualization icons. This property is useful, e.g., when comparing visualizations for raw and smoothed versions of the same data set.

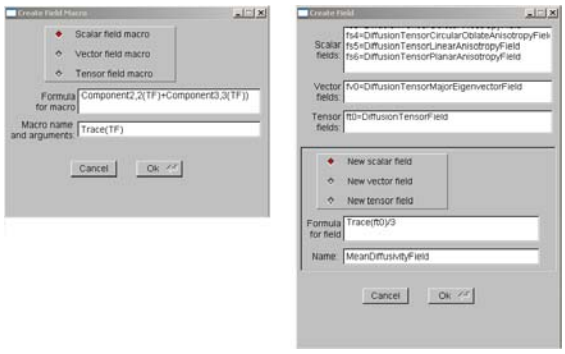

Figure 6. Defining a macro and using it to create a new field.

Frequently used expressions can also be saved as a macro and the macro name can then be used during field creation. An example is given in figure 6.

4 Case Study: Visualizing the Nerve Fiber Structure of the Brain

One recent advance in medical imaging has been the introduction of diffusion tensor imaging (DTI) which allows the measurement of water diffusion in the brain [1]. The diffusion tensor describes the spatial distribution of water molecules originating at a common location. Since the diffusion of water depends on the micro-structure of the tissue, visualizing the diffusion tensor field improves the understanding of the anatomy and the physiology of the brain. The results can be used in surgical planning, cognitive sciences, and the diagnosis and treatment of various white and gray matter disorders [5]. Extraction and visualizing the nerve fiber structure from the DTI data also represents a valuable teaching tool.

DTI uses diffusion weighted MR imaging (DWI) and almost completely suppresses water in blood vessels [Peter Basser, May 2000, private communication]. It therefore measures the diffusion of cerebral spinal fluid (CSF) and fluid within the tissue. The results of the measurement are the six components of the symmetric diffusion tensor **D** and the T_2 weighted signal intensity in the absence of diffusion sensitization. Images of water diffusion can pro-

vide pathophysiological information complementary to T_1 and T_2 weighted MRI images. The technique is sensitive to movements of the order of a few microns and is described in more detail in [1, 5].

In the brain DTI can be used to differentiate two types of structures. Fluid filled compartments are characterized by a very high *isotropic* diffusion, i.e., the diffusion is similar in all directions. In contrast *white matter* consists of nerve fibers which restrict the diffusion to one direction only due to the presence of cell membranes and myelin sheaths surrounding the axons. Fiber tracts, consisting of parallel nerve fibers, are therefore identified as areas of a high anisotropic diffusion. The orientation of such fiber tracts is determined from the principal directions (eigenvectors) of the diffusion tensor. Finally *gray matter* consists of neural cell bodies, support cells, intermingling nerve fibers and connecting contacts, and is characterized by a low, isotropic diffusion.

Introducing scalar measures for the above diffusion properties makes it possible to gain in vivo information about the anatomy, microstructure and physiology of the brain. In order to derive these measures it is convenient to order the three eigenvalues of the diffusion tensor $\mathbf{D}$ by size with λ_1 being the biggest and λ_3 being the smallest [5]. The *maximum diffusivity* is then given by $\lambda_{max} = \lambda_1$. A high isotropic diffusion can be measured by the *mean diffusivity* [1] which is defined as the average eigenvalue of the diffusion tensor and is efficiently computed by using the first tensor invariant

$$\lambda_{mean} = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} = \frac{D_{11} + D_{22} + D_{33}}{3} \quad (1)$$

The diffusion anisotropy is measured by [2]

$$\lambda_{anisotropy} = ((\mathbf{D} - \lambda_{mean}\mathbf{I})^T (\mathbf{D} - \lambda_{mean}\mathbf{I}))/\lambda_{mean}^2 \quad (2)$$

Note that both expressions can be efficiently calculated without computing the eigenvalues and eigenvectors.

Alternative measures have been proposed by Westin et al. [6]. The authors define a *linear isotropy* c_l, a *planar isotropy* c_p, and an *isotropy* c_s as

$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}\ ;\quad c_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3}\ ;\quad c_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \quad (3)$$

The measures fall in the range $[0, 1]$ and sum up to 1 and define therefore a barycentric space of anisotropies.

Using our field creation tool we can interactively define and experiment with various diffusion measures. Figure 7 (a) shows a visualization of the mean diffusivity. The yellow and red coloured regions indicate fluid filled compartments. Part (b) of the figure shows a visualization of the diffusion anisotropy. The following white matter structures are indicated by numbers: Genu (1) and splenium (2) of the corpus callosum, genu (4), anterior limb (5) and posterior limb (3) limb of the internal capsule, external capsule (6), fornix (7), and optic radiation (8). The dark blue coloured region enclosed by the corpus callosum and the internal capsule represents the thalamus and the lateral ventricle. The yellow dot in its centre is the fornix. The anisotropic regions in the periphery of the brain are due to eddy currents induced in the gradient/magnet system during DTI [5].

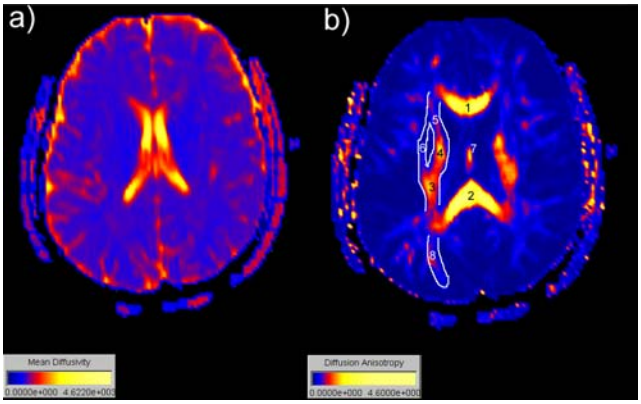

Figure 7. Horizontal slice showing the mean diffusivity (a) and diffusion anisotropy (b).

Regions of gray matter, white matter, and CSF can be displayed simultaneously using the segmentation function

$$\lambda_{segmented} = \begin{cases} 1 & \text{if } \lambda_{anisotropy} > 0.25 \\ 2 & \text{if } \lambda_{mean} > 10^{-3} \\ 3 & \text{if } \lambda_{mean} < 10^{-3} \text{ and } \lambda_{anisotropy} < 0.25 \\ 0 & \text{otherwise} \end{cases}$$

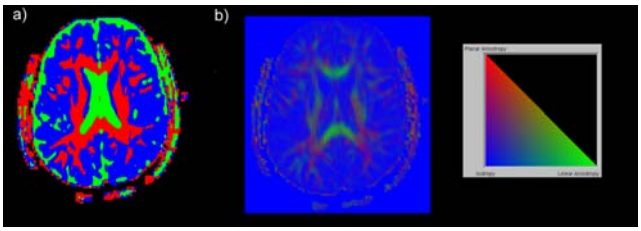

Figure 8. Horizontal slice coloured using a segmentation function (a) and a barycentric colour map (b).

The conditions for the values 1,2, and 3 are chosen so that they indicate white matter, CSF and gray matter. Figure 8 (a) shows the resulting segmentation using the colours red, green and blue, respectively. In contrast to the previous images this image allows the identification of the thalamus as the two blue region between the lateral ventricle in green and the internal capsule in red. The function is implemented using a conditional expression. The user can interactively adjust values to improve the segmentation result.

Finally we can also visualize nerve fiber tracts by integrating streamlines in the direction of the maximum diffusivity in regions of high diffusion anisotropy. Since similar diffusion properties can occur due to noise or eddy currents the maximum diffusivity at any step during the streamline integration must exceed a certain predefined minimum value and the streamline must exceed a specified minimum length [7]. An example is given in figure 9. In order to improve perception of the 3D geometry we generate streamtubes by fitting a cylinder with constant radius around each streamline. In order to reduce complexity the maximum number of streamtubes intersecting a cell is limited to eight.

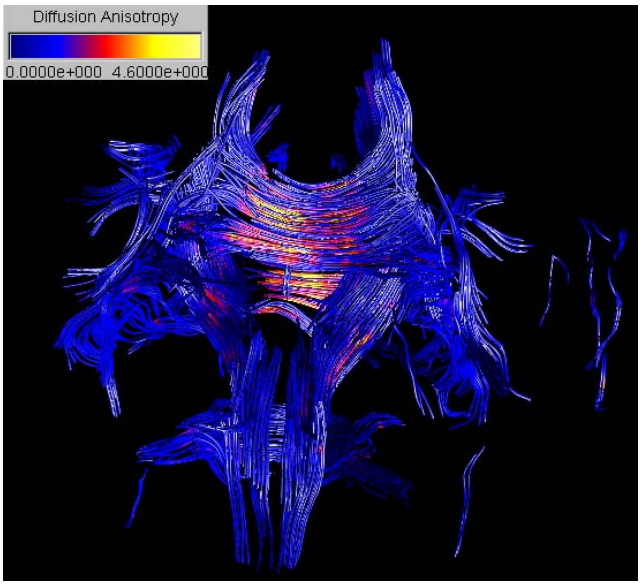

Figure 9. Fiber tracts visualized using cylindrical streamtubes colour mapped with the diffusion anisotropy.

5 Conclusion

Scientific data sets are continuously increasing in size and complexity. In this paper we presented a field data structure, which allows the user to interactively derive new measures in order to reveal structure in the data. The data structure is efficient in the sense that derived measures are only computed when required for the chosen visualization icon.

In contrast to previous research efforts our data structure allows the combination of different field types. A graphical user interface enables the user to define new measures as simple mathematical expression. Conditional expression can be used to visually segment data.

We demonstrated the capabilities of our data structure by visualizing the diffusion tensor field in a brain. By defining appropriate measures we visualized anatomical structures hidden in the data. We intend to use our visualization environment for the exploration of various white matter diseases and hope to gain new insight into the progress of neurodegenerative diseases by employing the presented techniques.

6 Acknowledgments

We would like to thank Peter J. Basser from the National Institute of Health, Bethesda, MD, for valuable discussions and Carlo Pierpaoli for providing us with the diffusion tensor data set of a healthy brain.

References

[1] P. J. Basser, J. Mattiello, and D. Le Bihan. MR diffusion tensor spectroscopy and imaging. *Biophysical Journal*, 66:259–267, 1994.

[2] Peter J. Basser and Carlo Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI. *Journal of Magnetic Resonance, Series B*, 111(3):209–219, 1996.

[3] Steve Bryson, David Kenwright, and Michael Gerald-Yamasaki. FEL: The field encapsulating library. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings of Visualization '96*, pages 241 – 247. IEEE, October 1996.

[4] Patrick J. Moran and Chris Henze. Large field visualization with demand-driven calculation. In David Ebert, Markus Gross, and Bernd Hamann, editors, *Proceedings of Visualization '99*, pages 27 – 33. IEEE, October 1999.

[5] Carlo Pierpaoli, Peter Jezzard, Peter J. Basser, Alan Barnett, and Giovanni Di Chiro. Diffusion tensor MR imaging of the human brain. *Radiology*, 201(3):637 – 648, December 1996.

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

[7] Burkhard C. Wünsche. *The Visualization of Tensor Fields in Biological Tissue*. PhD thesis, University of Auckland. (To be published).

[8] Burkhard C. Wünsche and Richard Lobb. A toolkit for the visualization of stress and strain tensor fields in biological tissue. In *Proceedings of VIP'99*, pages 6 – 15, December 1999.