A Toolkit for Visualizing Biomedical Data Sets

Burkhard C. Wünsche

Graphics Group, Dept. of Computer Science, University of Auckland, Auckland, New Zealand *

Abstract

Medical data sets now comprise a diverse range of measurements such as tissue densities, sensitivity to magnetization, blood flow velocity, and material strain. The size and complexity of medical data sets makes it increasingly difficult to understand, compare, analyze and communicate the data. Visualization is an attempt to simplify these tasks according to the motto "An image says more than a thousand words". Representing complex material properties, such as strain, as a single image improves the perception of features and pattern in the data, enables the recognition of relationship between different measures and facilitates the navigation through and interaction with complex and disparate sets of data.

This paper introduces a toolkit developed for exploring complex biomedical data sets. The contributions of this paper are threefold: we suggest a modular design which facilitates the comparison and exploration of multiple data sets and visualization. We introduce a novel field data structure which allows interactive creation of new fields and we present boolean filters as a universal visualization tool.

CR Categories: I.3.8 [Computer Graphics]: Applications; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics Data Structures and data types; J.3 [Computer Applications]: Life and Medical Sciences

Keywords: visualization, user interfaces, biomedicine, tensor fields

1 Introduction

The past decades have seen the introduction of a variety of medical imaging modalities such as as magnetic resonance imaging (MRI), computed tomography (CT), positron emission tomography (PET) and ultrasonography. In recent years this development has accelerated and a variety of new techniques for measuring tissue properties, fiber orientation and functional processes have been proposed [Tempany and McNeil 2001]. Consequently the available medical data sets now comprise measurements ranging from scalar fields such as tissue density (x-ray) and water content (MRI) to vector fields, such as blood flow velocity [Nayler et al. 1986], and tensor fields such as myocardial strain [Young et al. 1994b] and cellular water diffusion [Basser et al. 1994; Basser 1995].

*email:burkhard@cs.auckland.ac.nz

The size and complexity of medical data available today makes it difficult to analyze and to understand them. This is particularly true for multi-dimensional data such as vector and tensor fields, and for the simultaneous analysis of multiple data sets such as MRI, PET and CT. The understanding of the data can be improved by visualizing it.

A wide variety of visualization applications exists. The most popular applications are programming frameworks which cover diverse visualization tasks but can also be customized for specialized assignments. Popular visualization environments, such as AVS, IRIS Explorer, OpenDX (formerly IBM Data Explorer) and VTK exhibit modular and extensible components comprising the entire visualization process from data input and transformation to rendering. The underlying visualization model (*data flow paradigm*) consist of three processes: *Filtering* maps data into data, *mapping* converts the resulting data into geometric primitives and *rendering* creates images from these primitives.

We are particularly interested in the visualization of biomedical finite element models which are becoming increasingly important for understanding and simulating organ function and diseases [Bro-Nielsen and Cotin 1996; Höhne et al. 1996; Sagar 1996; Hunter et al. 1993]. Using the finite element data structure allows the definition of tissue properties in material coordinates, enables the selection of important structural components of the modeled organ (such as the inside or outside surface of the heart) and facilitates the computation of performance measures.

Most closely related to our research is *Amira*, originating from the Department for Scientific Visualization of the Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany [ZIB - Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany n. d.]. Amira is an integrated 3D visualization and volume modelling program for medicine, biology, and engineering. The program supports multiple coordinate systems, curvilinear grids, and creation of grids for FE simulation. Processing of 3D image data is supported by automatic and interactive segmentation tools. Advanced vector field and volume visualization tools are available and an expandable development version has been released.

In this paper we introduce a new toolkit for visualizing biomedical data. Our research concentrates on novel features which according to our knowledge are not or only partially supported in the previously mentioned tools. The paper commences with an introduction of finite element models and presents as example a model of the left ventricle of a human heart. This is followed by an overview of our toolkit and a presentation of some novel features. In detail we introduce a novel field data structure, tools for element, plane, and point selection, and a universal filter tool. We conclude with a short discussion of important issues considered when designing the colour map control and the rendering control of this toolkit.

2 Finite Element Models

Much of the functionality of our toolkit makes use of the geometry and associated data structures of a finite element model which are explained in the following. The geometry of a *finite element* (FE) model is described by a set of nodes and a set of elements, which have these nodes as vertices. The nodal coordinates are interpolated

over an element using *interpolation functions*. Curvilinear elements can be defined by specifying additionally nodal derivatives.

As an example of a finite element consider the cubic Hermite-linear Lagrange element in two dimensions shown in figure 1 (b). We first specify a parent element, shown in part (a) of the figure, which 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 linearly in the given parameter direction. In our example we assume that additionally derivatives in ξ_1-direction $\left(\frac{\partial u}{\partial \xi_1}\right)_i$ $(i = 1,\ldots,4)$ are specified at the element nodes. In this case a cubic Hermite interpolation is performed in that coordinate direction.

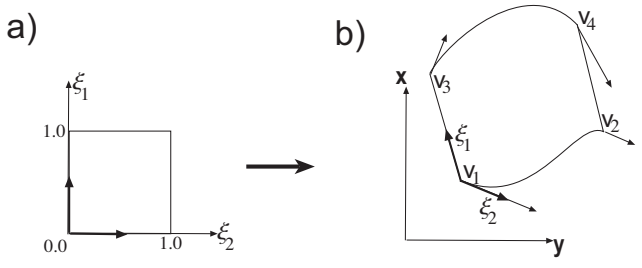

Figure 1: A cubic Hermite-linear Lagrange finite element.

The cubic Hermite-linear Lagrange interpolation of u over the entire 2D parameter space is then defined by the tensor products of the interpolation functions in each parameter direction:

$$\begin{aligned} & u(\xi_1,\xi_2) = \\ & H_1^0(\xi_1)L_1(\xi_2)u_1 + H_2^0(\xi_1)L_1(\xi_2)u_2 \\ + \quad & H_1^0(\xi_1)L_2(\xi_2)u_3 + H_2^0(\xi_1)L_2(\xi_2)u_4 \\ + \quad & H_1^1(\xi_1)L_1(\xi_2)\left(\frac{\partial u}{\partial \xi_1}\right)_1 + H_2^1(\xi_1)L_1(\xi_2)\left(\frac{\partial u}{\partial \xi_1}\right)_2 \\ + \quad & H_1^1(\xi_1)L_2(\xi_2)\left(\frac{\partial u}{\partial \xi_1}\right)_3 + H_2^1(\xi_1)L_2(\xi_2)\left(\frac{\partial u}{\partial \xi_1}\right)_4 \end{aligned} \tag{1}$$

where

$$L_1(\xi) = 1-\xi, \quad \text{and} \quad L_2(\xi) = \xi \tag{2}$$

are the one-dimensional linear Lagrange basis functions, and

$$\begin{aligned} H_1^0(\xi) &= 1-3\xi^2+2\xi^3, & H_1^1(\xi) &= \xi(\xi-1)^2 \\ H_2^0(\xi) &= \xi^2(3-2\xi), & H_2^1(\xi) &= \xi^2(\xi-1) \end{aligned} \tag{3}$$

are the one-dimensional cubic Hermite basis functions.

The geometry of an element in world coordinates (figure 1 (b)) is obtained by specifying the world-coordinates $\mathbf{v}_i$ and the ξ_1-tangents $\left(\frac{\partial \mathbf{v}}{\partial \xi_1}\right)_i$ $(i = 1,\ldots,4)$ of the element vertices and interpolating them as above.

2.1 A Model of the Left Ventricle

The Department of Physiology of the University of Auckland has created a model of the human left ventricle and the associated strain tensor field using *tagged Magnetic Resonance Imaging* (tagged MRI). The geometry of the left-ventricle, shown in figure 2, is described by 16 finite elements and uses bicubic Hermite interpolation in circumferential and longitudinal directions with linear interpolation in radial direction [Young et al. 1994b; Young et al. 1995].

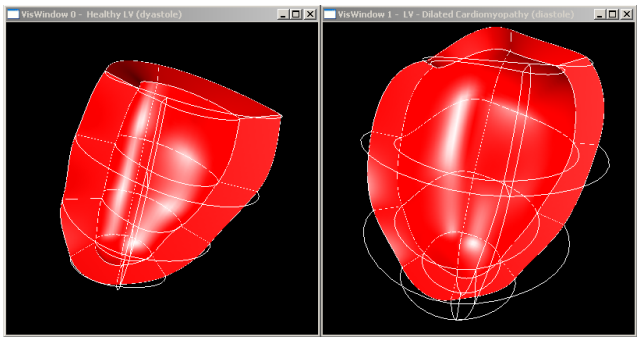

Figure 2: Finite Element Model of a healthy (left) and a diseased (right) left ventricle.

The authors obtain the ventricular geometry by tracing ventricular contours on MRI slices and by fitting the FE mesh to it. Strain information is obtained from tagged MRI images. When the heart deforms the tag lines deform with it making it possible to compute the displacement field of the myocardium from which the strain tensor is derived [Young et al. 1994a]. The strain field is defined as a regular grid of $10 \times 10 \times 6$ sample points over the material coordinates of the tensor. The strain tensors themselves are defined with respect to the material coordinate system of the corresponding elements.

The left ventricular model is used to demonstrate several of the features of our toolkit which is introduced in the next section.

3 The Visualization Toolkit

The top-level control of the visualization toolkit manages lists of models, visualization controls, visualization windows, rendering controls, and colour maps which are used for the creation of visualization icons. Figure 3 shows a screen shot of the toolkit at work.

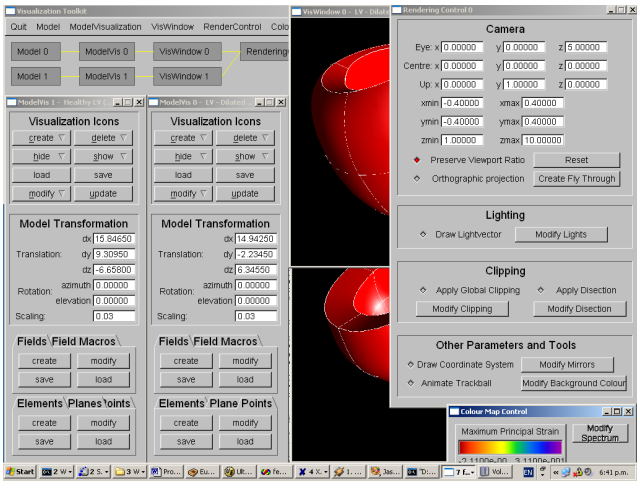

Figure 3: An example of the visualization toolkit at work. The image shows the toolkit control at the top-left, 2 visualization windows partially covered in the middle, 2 visualization controls at the left, and one rendering control in the top-right and the colour map control partially covered in the bottom-right.

The *model* used in our toolkit is always a finite element model. If the visualized data is not associated with a FE model a single trilinear element representing the bounding box of the data volume is used as a default model.

The *visualization control* contains a display list of *visualization icons* and a set of transformation parameters. Visualization icons are high-level graphic primitives representing a visualization. Our toolkit implements most common icons such as particle systems, vector glyphs, streamlines and streamtubes, hyperstreamlines, colour mapped surfaces, height fields, tensor glyphs and various types of line integral convolution textures. Annotations used to identify features and to explain relationships can also be created. Examples are legends, labels, and markers.

Transformations are necessary to align two data sets, e.g., MRI and PET data, or to represent two models, such as a healthy and a sick heart, at the same scale. The model visualization also contains data fields associated with the model (including any interactively defined new measures) and a list of element, plane and point sets used to define the location of visualization icons. The advantage of this design is that the user can simultaneously run two visualizations (e.g., from two different research groups) with different icons and fields for the same model.

A visualization window displays a model visualization with rendering parameters provided by a rendering control object. A rendering control contains a view, a trackball, lighting information, mirrors, and global clipping planes. The same rendering control can be used for different windows which is useful, for example, when comparing two different models. Vice versa the same model visualization can be displayed in different windows with different rendering parameters, for example, in order to display two different sides of the model simultaneously or in order to give a global and a detail view. Each model visualization is associated with exactly one model. A model can have several model visualizations which enables the user to display different visualizations of the same model at the same time.

The toolkit contains a list of colour maps which are used by the visualization icons in a model visualization. The decision to make the colour maps "global" was motivated by users who found it easier to interpret visualizations when identical colour maps were used for different visualizations of the same or different models. A typical example is the comparison of the strain fields in a sick and a healthy left ventricle.

The top-level control of the toolkit, shown in the top-left of figure 3, displays the relationship between its components graphically and allows the user to hide, show, add and delete additional components. The entire visualization toolkit was written in C++ using OpenGL and FLTK, a LGPL'd C++ graphical user interface toolkit for X (UNIX), OpenGL, and WIN32 (Microsoft Windows NT 4.0, 95, or 98) [Spitzak n. d.].

In the following we explain several of the novel features of this toolkit in more detail and show examples how they can be employed to create effective visualizations.

4 Computing Model Properties from the FE Geometry

If the model geometry is defined by finite elements it is possible to efficiently compute various volume, surface and length measures. The volume of a single element is obtained by integrating the identity function over the finite element in world coordinates. The calculation is simplified by using the substitution rule of multidimensional integration [Heuser 1981, p.478]

$$\int_{\mathbf{g}(T)} f(\mathbf{x})\, d\mathbf{x} = \int_T f(\mathbf{g}(\mathbf{x}))\, |det \mathbf{J}_\mathbf{g}(\mathbf{t})|\, d\mathbf{t} \qquad (4)$$

where f is the identity function, T is the unit cube representing the domain of the parent element, $\mathbf{g}$ is the transformation function from ξ-coordinates to world coordinates and $\mathbf{J_g}$ is its Jacobian. The resulting integral can be evaluated efficiently using Gaussian Quadrature [Burnett 1987]. Depending on the degree of the polynomial interpolation functions it is possible to determine the degree of the ξ-coordinates in the polynomial integrand. For example, if a cubic Hermite interpolation is used 5 gauss points in the corresponding coordinate direction are sufficient to achieve exact integration.

As an example consider the heart model at end-diastole (maximum expansion) and end-systole (maximum contraction) shown in figure 4. An important measure is the volume and the fraction of ejected blood. Using our toolkit it is possible to model the left-ventricular cavity by finite elements. This is done be creating centroids for any four vertices on the surface of the ventricular cavity with common longitudinal ξ-coordinate. Connecting these centroids to the corresponding points on the cavity surface results in 16 finite elements for the left ventricular cavity. Applying the above computation to the elements forming the cavity yields the desired results.

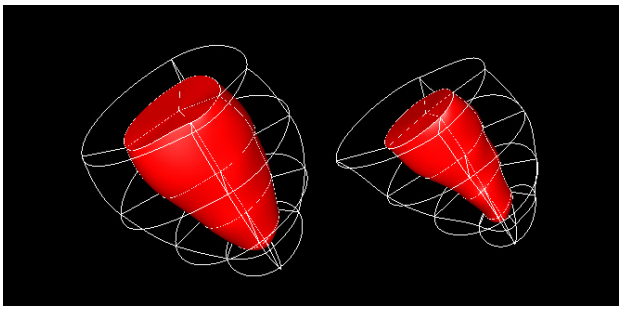

Figure 4: Left ventricular cavity of the healthy heart at end-diastole (left) and end-systole (right).

Similar methods to compute surface and length measures are also implemented [Wünsche 2002b].

5 The Field Data Structure

While the domain of a visualization is always a finite element model fields can be specified independent of that. For example, in some application it is desirable to mix a numerical model of an organ with MRI and CT raw data, which are usually specified in Cartesian device coordinates.

When combining fields specified in different coordinate systems a mapping between coordinate systems 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 [Press et al. 1992]. We found that 3 iterations are usually enough to find a material point inside an element for a given point in world coordinates.

Using this mapping makes it possible to use fields with different domains and to combine them using various operators. The data structure for a field variable 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 5. 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.

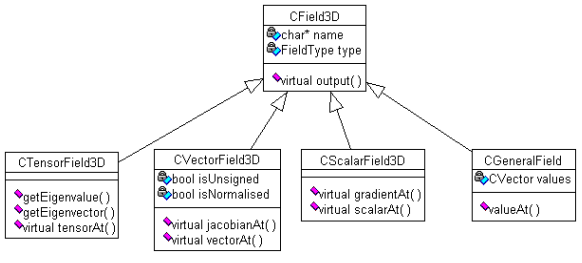

Figure 5: Top-Level class diagram of the field data structure.

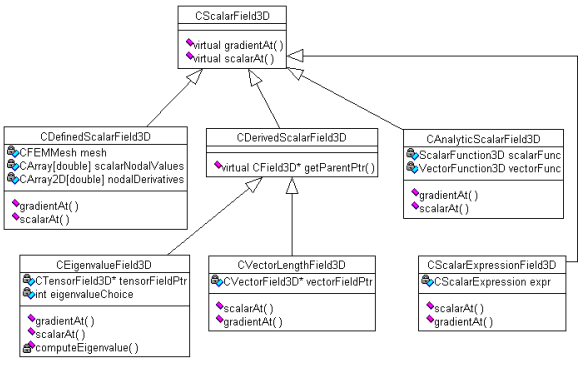

Figure 6: Class diagram of (a subset of) the scalar field data structure.

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 6 demonstrates the subclassing of the field data structure in figure 5 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 a field. For a regular trilinearly interpolated sample grid finite differences can be employed, for analytic functions a numerical differentiation can be used and for higher-order finite element meshes the derivatives of the interpolation functions can be used to obtain the gradient of the field.

The advantages of 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, when sampling an analytic field in order to create a new field over a given 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 [Wünsche 2002b]).

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.

5.1 GUI For Editing Fields

The graphical user interface for field creation, pictured on the right of figure 7, contains three output text components listing the currently defined scalar, vector and tensor fields. The user can create a new field by typing a simple mathematical expression into the input text field below.

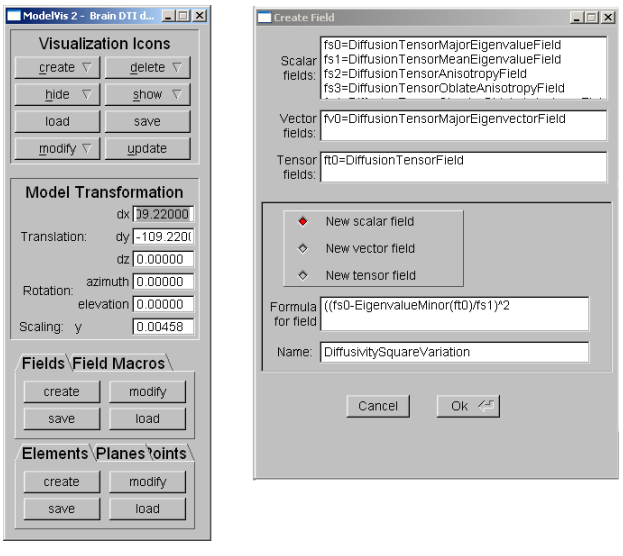

Figure 7: The control window for a model visualization (left) and the user interface for creating new fields (right).

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 the 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. Section 7 demonstrates how such an expression can be used for the visual segmentation of an image.

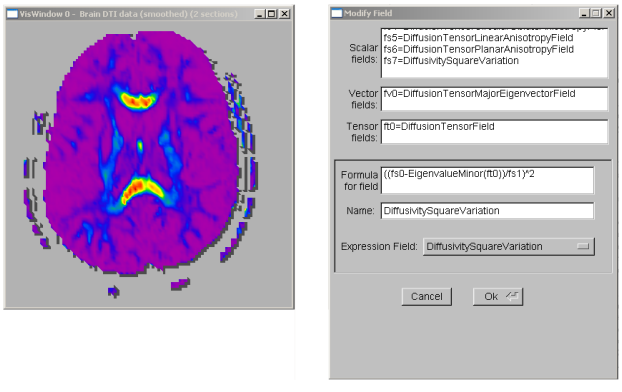

Figure 8: The visualization of the field defined in figure 7 (left) and the user interface used to edit an expression field (right).

The left part of figure 8 shows a visualization obtained by using the field in figure 7. 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 8. Using the update button of the model control (figure 7 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.

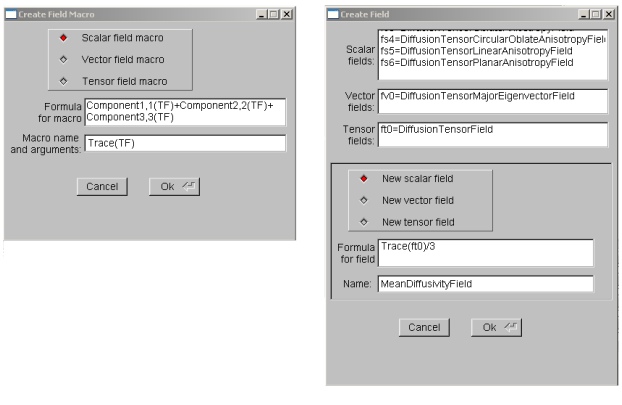

Figure 9: Defining a macro and using it to create a new field.

Frequently used expressions can be saved as a macro and the macro name can then be used during field creation. An example is given in figure 9.

6 Element, Plane and Point Selection

When visualizing a data set visualization icons must be created for selected elements, surfaces or points of interest. Rather than requiring the user to specify the domain of an icon each time a new one is created the toolkit keeps a list of previously selected sets of elements, surfaces and points. For example, when examining the left ventricle (figure 2) medical specialists are particularly interested in its outside surface (endocardial surface), the inside surface (epicardial surface) and the surface in the middle of the heart wall. Using the selection data structure the user can define these surfaces and reuse them for different visualization icons such as colour mapped surfaces and line integral convolution textures.

6.1 Element Selection

A collection of elements can be specified as a range of element indices, a list of element indices, as elements enclosed by a bounding box, and as elements intersected by a plane. Examples of the various selection mechanisms are given in figure 10.

A range of elements proves useful for applications where most or all of the elements of the model are selected. The element list and bounding box are usually employed for selecting a small region of interest. The set of elements intersected by a plane proves useful for the definition of colour mapped surfaces and height fields.

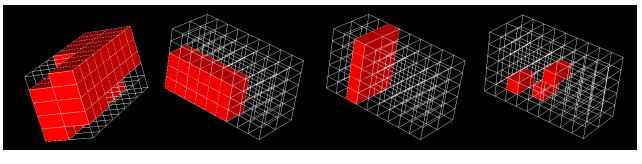

Figure 10: Examples of various element sets. From left to right: a range of elements, elements enclosed by a bounding box, elements intersected by a plane, and elements specified as a user defined list

6.2 Plane Selection

Planes can be specified in material and world coordinates. A plane in material coordinates is characterized by a constant ξ_i-parameter $(i = 1, \ldots, 3)$ and in case of curvilinear elements becomes a curved surface in world coordinates. The user can specify the range of elements for the plane using an previously defined element set.

A plane in world coordinates is either interactively positioned by the user or is specified as a plane parallel to the coordinate planes. In the latter case the rendered section of the plane is obtained by the intersection with an enlarged bounding box of the model. Examples of the various selection mechanisms are demonstrated in figure 11.

6.3 Point Selection

Many visualization icons work best by distributing them randomly or regularly over a region of interest. Typical examples are vector arrows or the seed points of streamlines. A set of points can be defined as a regular grid, a random selection of points, or an explicit listing of points. Any of these definitions can be performed either in material space or in world coordinates.

A regular grid in material space is specified by a set of elements and the number of points in each material coordinate direction. In

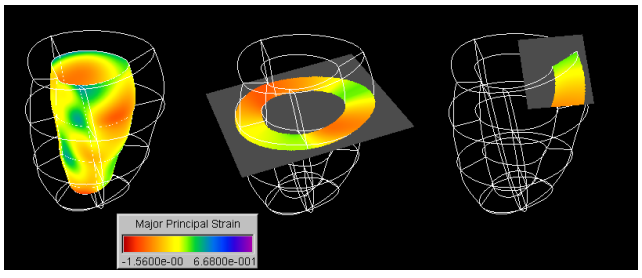

Figure 11: Examples of various plane sets (all colour mapped with the principal strain). From left to right: material coordinate planes ($\xi_3 = 0$) selected for all elements, a plane orthogonal to the world coordinate z-axis, an interactively defined plane in world coordinates.

world coordinates a regular grid is defined by a bounding box and the number of points in each coordinate direction. A selection of random points is specified by the total number of points. If points are randomly distributed over the material space a set of elements must be specified, whereas for random points in world space a bounding box is required. Finally a list of sample points can be specified directly either by their world coordinates or by element IDs and the corresponding ξ-coordinates.

Points can also be regularly or randomly distributed over a plane specified in either material or world coordinates. Regular 2D grids are well suited as start points for a bundle of streamlines or hyperstreamlines. The divergence or convergence of initially parallel sreamlines gives information which would be hard to observe when, for example, starting streamlines at random points.

When enumerating points defined over a world coordinate domain samples outside the model domain are skipped. Examples are given in figure 12.

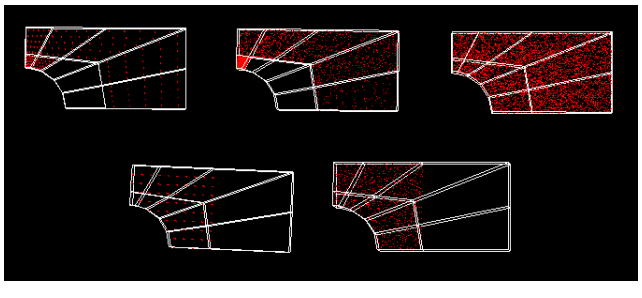

Figure 12: Examples of methods for point sampling. From left to right and top to bottom: regular grid in material coordinates, random points in material coordinates, random points in material coordinates volume weighted, regular grid in world coordinates, and random points in world coordinates. The first three methods employ an element selection mechanism, the last two methods a bounding box in world coordinates.

Points specified in material coordinates are usually more efficient since they can be used immediately as parameters of the field interpolation functions (if the field uses a FE interpolation). In contrast, a world coordinate point must be transformed into material coordinate first. Samples in material coordinates may also result in a more informative visualization since the material space often corresponds to the underlying structure of a model (e.g., the anatomic structure of the ventricle). A disadvantage of sample point specified in material space is that sample density in world coordinates varies according to the volume of an element (top-left of figure 12). Note, however, that this effect is sometimes desirable since small elements are frequently used in finite element modelling to represent regions with large field variations which are of particular interest to the user.

7 Filters

In many instances visualization icons are only required in regions with interesting field properties. Such regions can be specified by using *filters* which are boolean expressions defined over the model domain containing three types of terms: a comparison between scalar fields or constants, e.g., "$density(\mathbf{x}) > 0.5$", a range expression, e.g., "$0 < density(\mathbf{x}) < 1$", or a probabilistic expression where the probability that the expression is true is determined by the value of a scalar field. The maximum value of the scalar field gives a probability of one and the minimum value a probability of zero.

The toolkit uses filters for three tasks: The first task is the definition of point selections. Any of the previously introduced point selection mechanisms can be supplemented with a filter. A sample point is selected only if the boolean expression at that point is true. This tool is useful for the creation of visualization icons at regions of interest, e.g., points where the blood flow velocity exceeds a certain limit.

The second task is the definition of conditional expressions in the field data structure introduced in section 5. All conditions of such an expression are represented by filter objects.

We have used conditional expressions for the visual segmentation of data sets. For example, using the mean diffusivity λ_{mean} and the diffusion anisotropy $\lambda_{anisotropy}$ of diffusion tensor data [Wünsche and Lobb 2001b] it is possible to characterize three types of brain tissue by

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

The expression represents a conditional field and can be visualized using a colour map with different hues for the values 0,1,2, and 3. The conditions for the values 1,2, and 3 are chosen so that they indicate white matter, cerebral spinal fluid and gray matter, respectively. Figure 13 (b) shows the resulting segmentation using the colours red, green and blue, respectively. The user can interactively adjust values to improve the segmentation result.

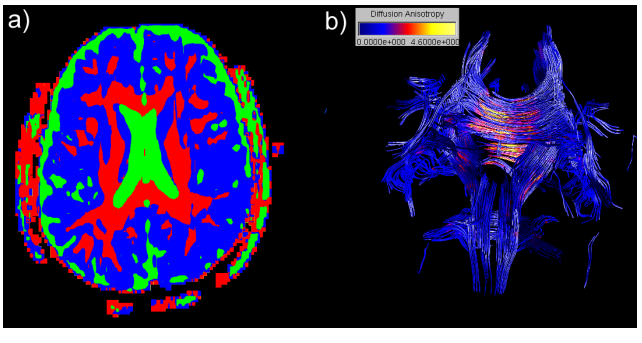

Figure 13: (a) Horizontal slice of the brain segmented into regions of white matter, gray matter, and fluid filled compartments. (b) Visualization of the nerve fiber structure of the brain using streamtubes.

Finally filters are also useful for specifying the shape of a visualization icon. For example, when defining streamlines and streamtubes filters can be used as an integration condition. The integration along a vector field is continued as long as the filter at the current point is true. Figure 13 (b) shows the result of using this tool for the extraction of the nerve fiber structure from a diffusion tensor data set [Wünsche and Lobb 2001b]. Streamtubes are integrated in the maximum diffusion direction of cellular fluid until the average diffusivity or diffusion anisotropy falls below a given thresholds.

8 The Colour Map Control

A popular method to visualize a scalar field over a one-dimensional or two-dimensional domain is colour mapping (e.g., [Cox 1988]). The technique associates a range of scalar field values with a colour spectrum and displays it by rendering the domain of the scalar field in the appropriate colours. Colour maps can also be used to map scalar information onto other visualization icons such as streamlines and streamtubes.

The visualization toolkit contains a global list of colour maps which can be used for multiple icons of the same or different models. A colour map consist of a colour spectrum, a range of field values associated with the spectrum, and a default min and max colour indicating values above or below the specified range.

A selection of colour spectra implemented is shown in the left part of figure 14: spectra with hue variations only, such as the rainbow colour scale, are best suited for illuminated surfaces since the surface shading variations do interfere with the brightness variations of a colour spectrum. Spectra with brightness variation only, such as the linear gray scale, are best suited for visualizations employing a large number of different visualization icons since that way interference between icon colours is minimized. Finally colour spectra with hue and brightness variations maximize the number of perceivable different field values.

In order to further minimize interference between different visualization icons we have created a graphical user interface for creating new colour maps. If a colour map is changed all visualization icons using this map are marked as changed and can be updated with a single button press. The automatic update proves useful when adding a new model to an existing visualization. For example, if a visualization has been created for the model of the healthy left ventricle and the user wants to compare the results with the sick left ventricle it is sufficient to change the range of all colour maps to reflect the range of field values for both models.

In order to minimize artifacts due to colour interpolation the colour maps are implemented using a one-dimensional texture map. This enabled us to introduce *colour map markers* as a new feature. Markers are inserted into the colour spectrum within the specified range and appear as isocontours on a colour mapped surface as illustrated in the middle part of figure 14. As an additional new tool we suggest cyclical colour maps, which map several cycles of a colour spectrum over the specified mapping range. We have found cyclical colour maps are especially useful when trying to understand the fine structure of a scalar field and to uncover symmetries and discontinuities [Wünsche 2002b]. The example in the right part of figure 14 shows clearly some discontinuities of the visualized scalar field along the element boundaries. Also note that the contour density and contour normal direction of a surface mapped with a cyclical colour map indicates the magnitude and direction, respectively, of the visualized scalar field.

9 The Rendering Control

The rendering control contains the view parameters, a trackball, lighting information, mirrors, and global clipping planes. The same

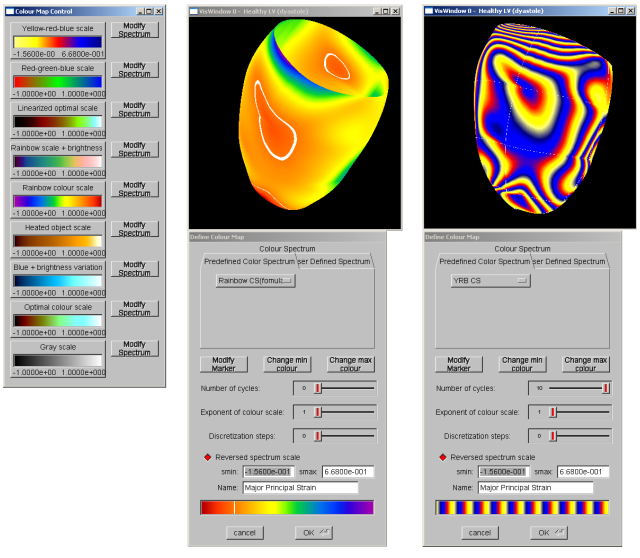

Figure 14: Subset of the colour spectra available in the toolkit (left). Isocontour separating contracting and expanding regions of the heart muscle created by using a spectrum marker (middle). A cyclical colour map (right).

rendering control can be used for different windows which is practical, for example, when comparing two different models.

A useful feature is the animation of models. Animations are valuable when using large numbers of icons distributed over a 3D domain. Rotating the model around its axis enables the brain to differentiate icons in the foreground and background. Consequently our toolkit incorporates a function to animate the trackball used to rotate the model. A fly-through is also available.

It is important to mention that all visualization icons have a default setting specifying whether they are illuminated or not. The user can change this setting if required. For most visualization icons lighting is enabled since shading is an important shape cue in 3D vision [Wünsche and Lobb 2001a]. However, illuminating a surface makes it difficult to perceive the object colour so that lighting is disabled for colour mapped objects as long as they are flat or sufficiently smooth.

10 Conclusion

This paper described a visualization toolkit for biomedical data sets with three novel features: The first feature is a modular design with separate objects describing the underlying model, the visualization including data fields, rendering parameters, and visualization windows. A visualization is achieved by defining relationships, subject to some constraints, between these objects. The design facilitates the definition of simultaneous visualizations of multiple models such as the simultaneous display of a sick and a healthy heart or the simultaneous display of a global and detailed view of an object. Using the same rendering parameters ensures that both models are displayed using the same view, scaling, orientation and lighting.

As a second feature we introduce a generalized field structure. The user can define scalar fields, vector fields, and tensor fields either by sets of sample points and interpolation functions or as analytic functions. The fields can be used to derive new fields using a set of predefined operators and general algebraic expressions. Examples are gradient fields and eigenvalue and eigenvector fields. The advantage of this construction is that fields are only evaluated if

they are used and that the user can interactively construct new non-standard fields when required for a given application. We found the feature especially useful when exploring tensor fields since the formation of new measures such as diffusion anisotropy can be used to extract anatomical structures. Fields can be represented with respect to both material and world coordinates.

The third novel feature of our toolkit are boolean filters which are used to control the positioning and shape of visualization icons. Boolean filters are also used for the creation of conditional fields which can be used for the segmentation of data.

Lastly the toolkit features a global colour map control and a model dependent point, surface and element selection mechanisms. The global colour map was motivated by the observation that users found it easier to derive qualitative and quantitative information when using the same colour scale for different fields in different models. Defining new colour maps is often necessary to avoid colour clashes when displaying multiple visualization icons simultaneously and gives the user additional freedom when exploring the data set. As novel modification we suggested spectrum markers, a technique to add isocontours to a colour map, and cyclical colour maps which are useful to extract structure and symmetries from fields. Model dependent point, surface, and element selection mechanism facilitate the placement and mixing of visualization icons.

We used our visualization toolkit in the past successfully to explore tensor fields in the heart and the brain [Wünsche 2002a; Wünsche 2002b] and we hope that future research will further increase our understanding of the working and the diseases of these organs.

11 Acknowledgments

We would like to thank Alistair A. Young from the Department of Physiology of the University of Auckland, Auckland, New Zealand, for valuable discussions and for providing us with the models of the left ventricle. Our thanks also goes to Dr. Richard White of the Cleveland Clinic, Cleveland, Ohio, USA, who kindly provided tagged MRI data of a left ventricle diagnosed with dilated cardiomyopathy and to 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

BASSER, P. J., MATTIELLO, J., AND BIHAN, D. L. 1994. MR diffusion tensor spectroscopy and imaging. *Biophysical Journal 66*, 259–267.

BASSER, P. J. 1995. Inferring microstructural features and the physiological state of tissues from diffusion-weighted images. *NMR in Biomedicine 8*, 7-8, 333–344.

BRO-NIELSEN, M., AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum (Proceedings of Eurographics '96) 15*, 3, C57 – C66. Futoroscope - Poitiers, France August 26 – 30, 1996, ISSN 0167-7055.

BURNETT, D. S. 1987. *Finite Element Analysis - From Concepts to Applications*. Addison-Wesley Publication Company Inc.

COX, D. J. 1988. Using the supercomputer to visualize higher dimensions: An artist's contribution to scientific visualization. *Leonardo 22*, 233 – 242.

HEUSER, H. 1981. *Lehrbuch der Analysis*, vol. 2. B.G. Teubner.

HÖHNE, K. H., PFLESSER, B., POMMERT, A., RIEMER, M., SCHIEMANN, T., SCHUBERT, R., AND TIEDE, U. 1996. A 'virtual body' model for surgical education and rehearsal. *Computer 29*, 1 (Jan.), 25–31.

HUNTER, P. J., NIELSEN, P. M. F., SMAILL, B. H., LEGRICE, I. J., AND HUNTER, I. W. 1993. An anatomical heart model with applications to myocardial activation and ventricular mechanics. In *High-Performance Computing in Biomedical Research*, T. C. Pilkington, Ed. CRC Press, ch. 1, 3–26.

NAYLER, G. L., FIRMIN, D. B., AND LONGMORE, D. B. 1986. Blood flow imaging by CINE magnetic resonance. *Journal of Computer Assisted Tomography 10*, 715–722.

PRESS, W. H., VETTERLING, W. T., TEUKOLSKY, S. A., AND FLANNERY, B. P. 1992. *Numerical Recipes in C - The Art of Scientific Computing*, second ed. Cambridge University Press.

SAGAR, M. A. 1996. *Creating Anatomic Models for Bioengineering Virtual Environments*. PhD thesis, Department of Engineering Science, The University of Auckland, Auckland, New Zealand.

SPITZAK, B. The FLTK home page. URL: http://www.fltk.org.

TEMPANY, C. M. C., AND MCNEIL, B. J. 2001. Advances in biomedical imaging. *The Journal of the American Medical Association 285*, 5 (Feb.), 556–561. URL: `http://jama.ama-assn.org/issues/v285n5/ffull/jsc00355.html`.

WÜNSCHE, B. C., AND LOBB, R. 2001. A scientific visualization schema incorporating perceptual concepts. In *Proceedings of IVCNZ'01*, 31 –36.

WÜNSCHE, B. C., AND LOBB, R. 2001. The visualization of diffusion tensor fields in the brain. In *Proceedings of METMBS'01*, CSREA Press, 498 – 504.

WÜNSCHE, B. C. 2002. The visualization and measurement of left ventricular deformation. In *Proceedings of APBC '03, The First Asia-Pacific Bioinformatics Conference*. (accepted for publication).

WÜNSCHE, B. C. 2002. *The Visualization of Tensor Fields in Biological Tissue*. PhD thesis, University of Auckland. (To be published).

YOUNG, A. A., IMAI, H., CHANG, C.-N., AND AXEL, L. 1994. Two-dimensional left ventricular deformation during systole using magnetic resonance imaging with spatial modulation of magnetization. *Circulation 89*, 2 (Feb.), 740 – 752.

YOUNG, A. A., KRAMER, C. M., FERRARI, V. A., AND AD NATHANIEL REICHEK, L. A. 1994. Three-dimensional left ventricular deformation in hypertrophic cardiomyopathy. *Circulation 90*, 2 (Aug.), 854 – 867.

YOUNG, A. A., KRAITCHMAN, D. L., DOUGHERTY, L., AND AXEL, L. 1995. Tracking and finite element analysis of stripe deformation in magnetic resonance tagging. *IEEE Transactions on Medical Imaging 14*, 3 (Sept.), 413 – 421.

ZIB - KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK BERLIN, GERMANY. The amira homepage. URL: http://www.amiravis.com/.