# DTI APPLICATION WITH HAPTIC INTERFACES

*Murat Aksoy* [1], *Neslehan Avcu* [2], *Susana Merino-Caviedes* [3], *Engin Deniz Diktaş* [4], *Miguel Ángel Martín-Fernández* [3], *Sıla Girgin* [4], *Ioannis Marras* [5], *Emma Muñoz-Moreno* [3], *Erkin Tekeli* [4], *Burak Acar* [4], *Roland Bammer* [1], *Marcos Martin-Fernandez* [3], *Ali Vahit Sahiner* [4], *Suzan Üskudarli* [4]

[1] Stanford University, School of Medicine, Dept. of Radiology, LUCAS MRS/I Center, USA
[2] Dokuz Eylül University, EE Dept., Izmir, Turkey
[3] University of Valladolid, Valladolid, Spain
[4] VAVlab, Boğaziçi University, İstanbul, Turkey
[5] AIIA lab, Aristotle University of Thessaloniki, Greece

## ABSTRACT

Diffusion Tensor Magnetic Resonance Imaging (DTI) is a rather new technique that allows in vivo imaging of the brain nervous structure. The DTI data is a 3D second-order positive semi-definite tensor field. DTI analysis and visualization is a challenging field due to the lack of standards and high-dimensionality of the data. This project focused on 1) implementation of a base DTI tool compatible with the tensor field standard (STAC) as proposed by the SIMILAR NoE WP10 group, 2) developing haptic interfaces for effective visualization and investigation of aforementioned 3D tensor fields. Both goals have been achieved, yet their integration could not be completed with the Enterface2007 workshop. Howeever, the know-how built during the workshop and the codes generated are invaluable resources for developing the final application. VAVlab (Boğaziçi University, EE Dept., İstanbul, Turkey) is currently working on the integration of these components into a complete application.

## KEYWORDS

DTI – Tensor Fields – Tractography – Tensor Standards – Haptic Interfaces – 3D Interactive Interfaces for Radiology.

## 1. INTRODUCTION

Diffusion Tensor Magnetic Resonance Imaging (DTI) is a rather new technique that enables researchers and physicians to image fiberous structures, such as nerves in brain, in vivo. Previously, such investigations could only be done in vitro, with the unavoidable consequences of dissection of the brain. Furthermore, in vivo imaging of fiber networks in brain is likely to facilitate the early diagnosis, surgery planning and/or follow-up screening [1, 2, 3].

DTI is based on measuring the MR signal attenuation due to random walk (diffusion) of water molecules in restricted media, namely inside the myelin coated nerve fibers, in response to a special diffusion weighting gradient magnetic fields. Thus measured 3D data is called the Diffusion Weighted Imaging (DWI). Several DWI sets can be combined to compute the DTI datasets, which is a second order, symmetric positive semi-definite tensor field in 3D. Each tensor represents the local physical diffusion process upto a second order approximation.

The challenge in DTI is not only in data acquisition, which is an active research area, but also in post-processing and visualizing tensor fields and user interfaces for effective communication of the information content.

The tensor data is a high dimensional volumetric data which is significantly different then volumetric scalar fields. For conventional 3D scalar fields, one has to be careful to set the relation between the data grid and the world (patient) reference frame correct. However, for a tensor field, one has to be careful in managing the relation between the DTI reference frame (the reference frame with respect to which the tensors are **defined**) and the world reference frame as well. Furthernore, the application of simple geometric transformations to a tensor field can be tricky as one has to transform the data itself together with the data grid. Such difficulties lead to significant problems in the absence of a tensor standard. Consequently, the SIMILAR NoE WorkPackage10 had initiated an effort to define a general tensor standard for use with DTI data as well in other fields. The standard proposed is called STAC (Similar Tensor Array Core) and is designed to incorporate the essential components to define a tensor field without ambiguity [4].

Fiber tractography is a well-known and widely used approach for visualization and analysis of DTI data. As will be explained in more detail below, it consists of numerical integration of the principal diffusion direction (PDD) as represented by the tensor data. The output of fiber tractography is a set of 3D streamlines representing fiber bundles in brain. The required interface for fiber tractography should enable the users to answer questions like "*Where do the fibers passing through region A end?*", "*Are there any fibers connecting region A and region B?*", "*What is the shape of the fiber bundles around this lesion?*", etc. All these questions require an intuitive, easy to navigate user interface that allows the users to see volumetric cross-sections they choose and the to shape and position region-of-interests (ROIs).

DTInteract is a proof-of-concept application developed during eNTERFACE 2007, İstanbul , Turkey. Its major goals are *i)* to incorporate the STAC into the volumetric analysis and visualization framework (VAVframe) being developed at Boğaziçi University, VAVlab, İstanbul, Turkey, and *ii)* to develop and test the use of haptic interfaces for DTI visualization and analysis.

## 2. DTI DATA

### 2.1. Theory of DTI Tensor Calculation

It is of utmost importance to understand what the DT-MRI signal represents in order to develop adequate analysis and visualization methods. DT-MRI measures the average signal attenuation within a small subvolume (i.e. a voxel) due to water molecules spinning out-of-phase.

The basis of MRI is to perturb the water molecules (the dipoles) that were aligned in a constant magnetic field ($B_0$ approx 1-7 Tesla) and let them re-orient themselves with $B_0$ during which the dipoles rotate around $B_0$ according to the Bloch's Equation. This rotation causes a temporal change in total magnetic field which induces a time-varying current at the receiving coils of the MR scanner. The time it takes for the dipoles to fully relax depends on the environment (i.e. the tissue). Thus, the amount of current induced is proportional to the number of dipoles (i.e water concentration) and the tissue type. These current measurements are transformed to monochromatic images in which each pixel value is also a function of water concentration and tissue type

In DT-MRI, extra spatially varying magnetic fields, the so called Diffusion Weighting Gradients (DWG), $G$, are applied together with $B_0$. Due to this added $G$ field, the water molecules under continuous brownian motion experience different total magnetic field at different locations. This causes them to rotate at different frequencies, i.e. to be out-of-phase. The effect of out-of-phase dipoles on the induced current is attenuation of the MR signal. So, the amount of attenuation in the received signal (equivalently in the diffusion weighted MR images) is a function of the brownian motion (i.e diffusion) of water molecules and the applied $G$ field.

This phenomenon is described by the superposition of the Bloch's Equation and the Diffusion Equation whose solution is

$$M_k = M_0 \exp\left(-\sum_{i=x,y,z}\sum_{j=x,y,z} \mathbf{b}_k^{i,j}\mathbf{D}_{i,j}\right) \quad (1)$$

Here, $M_k$ is the pixel intensity in the case when diffusion weighting is applied using the $k_{th}$ diffusion encoding direction, $M_0$ is the pixel intensity when there is no diffusion weighting, $\mathbf{D}$ is the $3 \times 3$ diffusion tensor, $\mathbf{b}_k$ is a $3 \times 3$ symmetric matrix that is determined by the applied diffusion weighting direction and strength and i,j are matrix indices. The six independent components of the symmetric diffusion tensor $\mathbf{D}$ can be computed using at least 6 linearly independent equations of this form where

$$\mathbf{b}_k = b_{\text{nom}}\mathbf{r}_k\mathbf{r}_k^T \quad (2)$$

[1]

$b_{nom}$ is a user-determined nominal b-value which is used to adjust the amount of diffusion weighting. This can be obtained from the header of the DICOM files containing the diffusion weighted images (DICOM tag (0019,10b0)). An example set of diffusion encoding directions ($\mathbf{r}_k$) is

$$\begin{array}{llll} & & \mathbf{r}_0 = & [0\ 0\ 0]^T \\ \mathbf{r}_1 = & [0.707\ 0.707\ 0]^T & \mathbf{r}_2 = & [0\ 0.707\ 0.707]^T \\ \mathbf{r}_3 = & [0.707\ 0\ 0.707]^T & \mathbf{r}_4 = & [-0.707\ 0.707\ 0]^T \\ \mathbf{r}_5 = & [0\ -0.707\ 0.707]^T & \mathbf{r}_6 = & [0.707\ 0\ -0.707]^T \end{array}$$

The diffusion weighted MR images are stored in `dwepi.*.grads` files.

Taking the natural logarithm and using linear algebra, for each pixel, a linear system can be obtained from multiple diffusion weighted measurements:

$$\underbrace{\begin{pmatrix} \ln M_1 \\ \ln M_2 \\ \cdots \\ \ln M_n \end{pmatrix}}_{\mathbf{S}} = \underbrace{\begin{pmatrix} 1 & -\mathbf{b}_1^{xx} & -\mathbf{b}_1^{yy} & -\mathbf{b}_1^{zz} & -2\mathbf{b}_1^{xy} & -2\mathbf{b}_1^{xz} & -2\mathbf{b}_1^{yz} \\ 1 & -\mathbf{b}_2^{xx} & -\mathbf{b}_2^{yy} & -\mathbf{b}_2^{zz} & -2\mathbf{b}_2^{xy} & -2\mathbf{b}_2^{xz} & -2\mathbf{b}_2^{yz} \\ \cdots & & & & & & \\ 1 & -\mathbf{b}_n^{xx} & -\mathbf{b}_n^{yy} & -\mathbf{b}_n^{zz} & -2\mathbf{b}_n^{xy} & -2\mathbf{b}_n^{xz} & -2\mathbf{b}_n^{yz} \end{pmatrix}}_{\mathbf{M}} \underbrace{\begin{pmatrix} \ln M_0 \\ \mathbf{D}_{xx} \\ \mathbf{D}_{yy} \\ \mathbf{D}_{zz} \\ \mathbf{D}_{xy} \\ \mathbf{D}_{xz} \\ \mathbf{D}_{yz} \end{pmatrix}}_{\mathbf{d}}$$

[1] $\mathbf{b}_k^{i,j}$ is the (i,j) component of b-matrix $\mathbf{b}_k$

Then, the unknown $\mathbf{d}$ vector can be found using pseudo-inverse:

$$\mathbf{d} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{S} \quad (3)$$

### 2.2. STAC: Similar Tensor Array Core

Diffusion Tensor (DT) images are a relatively new kind of images, that requires special features for their storage. Nowadays, there is not a standard format for this sort of data, and therefore specific writers and readers must be implemented to manage data acquired by different equipments. In order to deal with this, a standard has been proposed in [4] that is valid not only for diffusion tensor images, but for general tensor fields. It is named the Similar Tensor Array Core (STAC).

The standard consists of two parts: the general tensor standard array core (STAC) that is the basic type of storage for tensor fields, and the extensions that provides more specific functionalities for specific tensor types. By now, we have implemented the core header that is the main part of the tensor standard. The definition and implementation of an appropriate extension for DT-MRI will be considered as a future task.

The standard core header contains the following fields:

- `array_dimensionality`: It is the number of dimensions of the field.
- `array_size`: It is the size of the field.
- `array_index_names`: It contains the name of the index of the field, therefore is an array n values, where n is the `array_dimensionality`. It is an optional field that can be useful for processing issues.
- `array_tensor_metric`: It is a metric tensor that describes the metric in the array space. Thus, diagonal elements represents the scaling factor in each dimension. It is a $n \times n$ metric tensor, where $n$ is the `array_dimensionality`. It is not a mandatory field, by default is set to the identity.
- `tensor_order`: It is a scalar number describing the order of the tensor.
- `tensor_index_types`: Each tensor index can be contravariant or convariant. In this field, an array of $m$ elements, where $m$ is the `tensor_order`, describes what kind is each of the index.
- `tensor_index_names`: The name of the index of the tensor elements. It is an optional field, useful for processing tasks.
- `tensor_name`: A name can be given to the tensor. It is an optional field.
- `description`: Description of the tensor field can be included. It is an optional field.

We have implemented both reader and writer for tensor standard data storage. Both of them are included as methods in the VavDTI class. Two files are required to store the data: the first, that has extension `.stach`, contains the header, whereas the second contains the raw data and it has `.stacd` extension. Both files have the same name, they only differs in the extension.

#### *2.2.1. The Reader*

The `readerSTAC` method receives as an input the file name (without extension), and it assigns the appropriate values to the VavDTI attributes. The `array_size` is assigned to the `dimArray` variable, the `spacingArray` elements are set as the values of the diagonal of the metric tensor, and a VavTensor is built reading the data file. A value of '0' is returned by the function if no error happens.

#### 2.2.2. The Writer

The `writerSTAC` method has as input the file name (without extensions) where the VavTensor data will be stored. First of all, the header file (`*.stach`) is written:

- `array_dimensionality`: Since VavDTI class is implemented for 3D arrays, it is set to 3.
- `array_size`: The size of the array is obtained from the `dimArray` value.
- `array_index_names`: This is an optional field, if no value is given for the array names is set to 'None' by default.
- `array_tensor_metric`: The metric information that provides the VavDTI class is the `spacingArray` variable. Therefore, the tensor metric is represented by a 3x3 diagonal matrix. Each diagonal element represents the spacing in each direction.
- `tensor_order`: The diffusion tensors are second order tensors, so this field is set to 2.
- `tensor_index_types`: The diffusion tensor is a contravariant tensor, so both index are of contravariant type.
- `tensor_index_names`: This is an optional field, if no value is given for the array names is set to 'None' by default.
- `tensor_name`: The tensor is named DT (Diffusion Tensor)
- `description`: We include a brief description of the tensor field in this field.

After writing the header file, the data are written in the `*.stacd` file in raw format.

### 2.3. VAVframe Data Structure

In VavFrame, three data classes are generated for three main data types. These are: VavScalar, VavVector and VavTensor for scalar, vectoral and tensor fields. And VavData class is generated as an abstract data class for these three data types of VavFrame. All data classes are inherited from this class.

VavData class includes 3-dimensional vectors for dimension, space and origin information for each direction x,y,z. And two vtk objects are member of it. They are used to store data fields. These variables are common for three data classes, but the size of data components are not the same for all of them. There are one component for VavScalar, three components for VavVector, nine components for VavTensor classes for each voxel in the field.

Data are stored in VtkImageData type object. It is one dimensional array, but data is in three dimension. So, we hold data in order of x,y,z. And, the common reference frame with respect to the anatomy is as follows:

- 1st dimension (X): Left → Right
- 2st dimension (Y): Posterior → Anterior
- 3st dimension (Z): Inferior → Superior

VavDTI class is created to hold DTI specific data and manage DTI operations. It includes dimension, space and origin information; diffusion tensor field, eigenvectors and eigenvalues of this tensor field, mean diffusivity, $B_0$ images, fractional anisotropy, lattice index, coherence index, diffusion weighted images, bmatrices and a header information. These variables are instances of VavScalar, VavVector and VavTensor classes according to their types. There are reader functions for diffusion tensor data from different formats in VavDTI class, such as Stanford DTI data and PISTE DTI data. Also it includes a conversion function from DWI to DTI and an eigen decomposition function for DTI data.

### 2.4. Stanford DTI Data

There are two types of DTI data format that is being used at Stanford University. The first one is the diffusion weighted image DICOM (DWI-DICOM) format, and the second one is the tensor raw data.

The DWI-DICOM data simply consists of the diffusion weighted images belonging to each slice, diffusion weighted direction and scan repetition. All these files are in DICOM format, which is currently the standard file format used in the area of medical imaging [2]. These DICOM files are ordered consecutively and the file format is `Ixxxx.dcm`, where `xxxx` is a number which starts with 0001 and goes up to the total number of images. The DICOM file ordering, from the outermost to the innermost loop, is as follows: scan repetition → diffusion weighting direction → slice (Figure 1).

The second file format consists of pre-calculated raw tensor data [3]. This consists of two sets of files: The first set is the `TensorElements.float.xxx` files, where `xxx` denotes the slice number. These contain the tensor element in the order: $\mathbf{D}_{xx}$, $\mathbf{D}_{yy}$, $\mathbf{D}_{zz}$, $\mathbf{D}_{xy}$, $\mathbf{D}_{xz}$, $\mathbf{D}_{yz}$. In this case, the pixels are the innermost loop, i.e., for an image resolution of $n \times n$, the $n^2$ $\mathbf{D}_{xx}$ elements are written first, followed by $n^2$ $\mathbf{D}_{yy}$ elements, and so on.

The second set is the `Tensor.float.xxx` files. Here, the following $n \times n$ images are written to the file consecutively: Mean diffusivity (units x$10^6$ $\mathrm{mm}^2$/s), maximum eigenvalue, medium eigenvalue, minimum eigenvalue, x component of the maximum eigenvector (values between -1 and 1), y component of the maximum eigenvector, z component of the maximum eigenvector, Fractional Anisotropy (FA, multiplied by 1000) and b=0 image (arbitrary units).

### 2.5. PISTE DTI Data

DT-MRI based tractography techniques have been proposed to propagate fiber trajectories in diffusion tensor fields. For the accuracy and acceptability of these tracking methods, there must be a general data set to evaluate the algorithms and their results. PISTE, developed on behalf of the ISMRM Diffusion/Perfusion Study Group, following an initiate at the 2003 Toronto meeting, is intended as a resource for researchers interested in Diffusion Tensor Magnetic Resonance Imaging and Tractography. The aim is to provide a general database of simulated common fiber tract trajectories that can be used for testing, validating and comparing various tractography algorithms.

To evaluate the performance of tractography algorithms and analyze the influence of several factors on tracking, PISTE includes several datasets differing with respect to

- signal-to-noise ratio,
- tensor field anisotropy,
- fiber geometry,
- interpolation and
- step-size.

For each setup there are 5 datasets provided:

- A T2-weighted image;
- The 6 elements of the diffusion tensor;

[2] http://medical.nema.org/

[3] http://rsl.stanford.edu/moseley/tensorcalc/

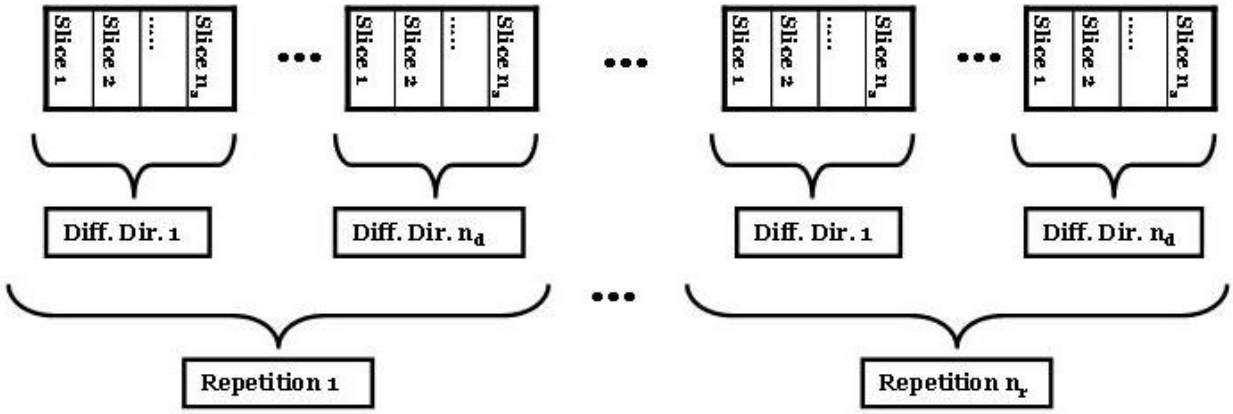


Figure 1: *Structure of DICOM files for DWI data. The data is acquired $n_r$ times for each one of the $n_d$ diffusion gradient directions. $n_d$ must be at least 6 to recover the diffusion gradient and $n_r$ is usually selected to be 4. The repeated data acquisition is performed for noise suppression.*

- The 3 eigenvalues and eigenvectors;
- An image containing the tract initiation point or region of interest;
- A calibration image showing the positive X,Y and Z directions.

## 3. DTINTERACT

### 3.1. System Components

DTInteract application is a proof-of-concept C++ application built on the VAVframe infrastructure. It consists of three main components, the data standardization, fiber tractography, haptic interfaces for tractography. The first component has been explained in Section 2. Its major concern is to load 3D tensor fields (and related data) into a common workplace, which is chosen to be the VAVframe environment. We have currently dealt with two datasets, the DTI data provided by Stanford University, LUCAS MRS/I Center, CA[4] and the PISTE phantom DTI dataset. The second component is composed of $4^{th}$ order Runge-Kutta based fiber tractography (PDD tracking). This is the most well-known and widely used method for tractography. Despite its drawbacks, we have decided to use it in this proof-of-concept application. The final component is the haptic interface. It is used to position orthogonal slices (axial, sagittal, coronal) in 3D, as well as to shape and locate 3D ROIs (region-of-interest). These will be discussed in more detail in the following.

### 3.2. Base Application

The programming language used for the interface development is C++, given the availability of libraries of classes written in this language for visualization, haptics and user interface development. In addition, C++ performance, although not as efficient as C, is comparable to it.

Three C++ libraries have been used: VTK (Visualization Toolkit) [5], CHAI-3D [6] and Qt [7], which will be described below. VTK and CHAI-3D are open source and free. The open source version of Qt was employed for this project.

VTK is a widely used C++ class library for visualization tasks and image processing, and it provides high level classes for computer graphics. It makes use of the underlying low level graphics library, like for example OpenGL. VTK provides tools for creating complex visualization scenes, and is multiplatform.

The application uses a Sensable Phantom Omni as haptic input device for the interaction with the user. In order to control it, the library CHAI-3D is employed, which provides classes for the management of a wide variety of haptic devices and real-time simulations.

Qt is a C++ object library with a rich variety of resources for user interfaces development. It also provides a signal-slot paradigm for communication between objects. Signals and slots can be connected, so that when a signal is sent, the slots it is attached to are executed. This mechanism provides the fundamental communication system of our interface.

The main classes that intervene in the application are VavDTI, TestApplication, VavDTIViewer and the classes involved in haptic management, which will be explained here below.

#### *3.2.1. VavDTI*

This class is able to load and save DTI data in Stanford and STAC formats, extract some of the tensor characteristics. The methods for computing tractographies and visualizing the different DTI data are also included in this class.

#### *3.2.2. TestApplication*

This class provides the interface management and interconnectivity of the different parts of the application. It gives functionality to the interface and manages the methods provided by VavDTI. It also provides functions for connecting the haptic classes for user interaction.

#### *3.2.3. VavDTIViewer*

This class inherits from the class vav3DViewer, which has as a consequence that it encapsulates both a QVTKWidget and a vtkRenderer object. This class is used to provide a display for the visualization of the DTI components and provide haptic interaction to the application, along with the class TestApplication.

The application uses orthogonal slices of scalar voumes for the fractional anisotropy, mean diffusivity and B0 coefficients. The major eigenvector field is visualized using lines at each point of the slices, and the tracts are visualized as tubes (Figure 2).

[4] Courtesy of Prof. Roland Bammer, PhD.

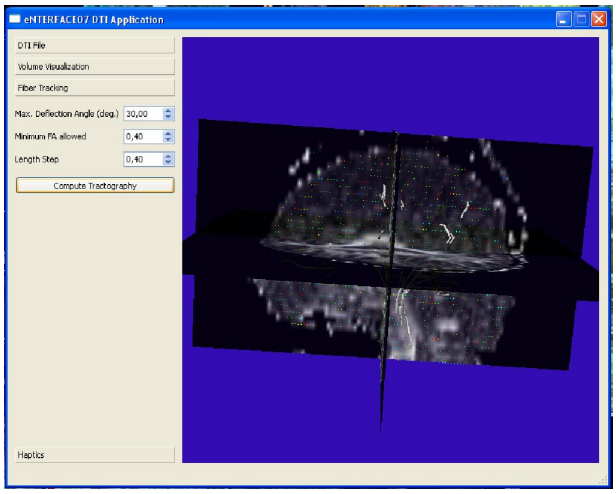


Figure 2: *GUI for the base DTI Application.*

### 3.3. DTI Tractography

Since water diffusion in brain is constrained by the myelin covering of the axons, it is possible to assess the fiber structure by tracking the direction given by the DT major eigenvector. This eigenvector represents the maximum diffusion direction, and for this reason, it is supposed to be tangent to the fiber bundles at each voxel. In this way, the integration of the vector filed will provide the diffusion path. Some numerical methods can be used in order to integrate the vector fields, such as Euler or Runge-Kutta integration methods. In this implementation, we have considered a $4^{th}$ order Runge-Kutta integration method, which is detailed just below.

#### *3.3.1. $4^{th}$ order Runge-Kutta method*

The integration method must start from an initial or seed point $r_o$, from which the diffusion path is defined. The points that belongs to the path are iteratively computed according to the next equation [8]:

$$\mathbf{r}_{n+1} = \mathbf{r}_n + h\mathbf{V}_{n+1} \tag{4}$$

$h$ stands for the step length parameter, that is a parameter set by the user in our implementation. $\mathbf{V}_n$ is a vector computed by the next equation that is actually the $4^{th}$ order Runge-Kutta method:

$$\mathbf{V}_{n+1} = \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{5}$$

Where $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ are defined as a function of a continuous **V**ector field $\mathbf{E}(\mathbf{r})$:

$$\begin{aligned}
\mathbf{k}_1 &= \frac{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n)}{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n)} \cdot \mathbf{E}(\mathbf{r}_n) \\
\mathbf{k}_2 &= \frac{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n + \frac{h}{2}\cdot\mathbf{k}_1)}{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n + \frac{h}{2}\cdot\mathbf{k}_1)} \cdot \mathbf{E}(\mathbf{r}_n + \frac{h}{2}\cdot\mathbf{k}_1) \\
\mathbf{k}_3 &= \frac{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n + \frac{h}{2}\cdot\mathbf{k}_2)}{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n + \frac{h}{2}\cdot\mathbf{k}_2)} \cdot \mathbf{E}(\mathbf{r}_n + \frac{h}{2}\cdot\mathbf{k}_2) \\
\mathbf{k}_4 &= \frac{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n + h\cdot\mathbf{k}_3)}{\mathbf{V}_n\mathbf{E}(\mathbf{r}_n + h\cdot\mathbf{k}_3)} \cdot \mathbf{E}(\mathbf{r}_n + h\cdot\mathbf{k}_3)
\end{aligned}$$

This definition of the coefficients ensures that the angle between consecutive vectors is smaller than 90°.

#### *3.3.2. Implementation*

The Runge-Kutta method is implemented by the RungeKuttaIntegration method, that has three input parameters, the curvature threshold, the step length and the anisotropy threshold, that are more detailed below. It returns 0 value if there is not error. The resulting streamlines are stored in the streamlineList attribute of the VavDTI class that is a vtkPolyData object. Next, we comment some implementation details.

- `Seed Points`: In our implementation, we consider as seed point every voxel whose anisotropy is higher than a given threshold value set by the user. Since fibers belongs to the white matter, that is the area where anisotropy is higher, it only makes sense to compute streamlines in areas of high anisotropy. Moreover, the voxels that belongs to a previously computed streamline are not used as seed points.
- `Initial conditions`: For each seed point $r_o$, a streamline is computed following the Runge-Kutta integration method. The initial value of $\mathbf{V}_n$ is $\mathbf{V}_o = \mathbf{e}(\mathbf{r}_o)$, that is, the eigenvector in the initial point.
- `Stop criteria`: The streamline computation from a given seed point stops when one of the following conditions is achieved:
  - The line arrives to a region of lower anisotropy, that is, the fractional anisotropy is lower than the threshold set by the user.
  - The angle between successive streamlines segments is higher than a threshold value set by the user. This criteria is based on a priori anatomical knowledge: the fiber curvature is small.
- `Interpolation`: Let's notice that the Runge-Kutta algorithm considers a continuous vector field, but the eigenvector field is defined in a discrete grid. For this reason, interpolation of the vector field is required. In this implementation, we have computed a linear interpolation among the nearest eigenvectors.
- `Tracking Parameters`: Three parameters are set by the user to carry out the tractography algorithm:
  - Fractional anisotropy threshold: The voxels whose anisotropy is higher than this threshold will be considered as white matter, and the fibers will be computed from them. By default is set to 0.15.
  - Curvature threshold: It represents the maximum angle of curvature allowed for a streamline. If the angle is higher, the streamline tracking is stopped. The default value is $\pi/6$ rad.
  - It is the value of the step length $h$ in the Runge-Kutta algorithm. By default is set to 0.9, that is one half of the voxel size in the smallest dimension.

### 3.4. Haptic Interface

Haptic devices are capable of enhancing the space comprehension in a user interaction with the virtual environment, by providing many Degrees-of-Freedom (DoF) for the input as well as force-feedback, thus achieving an increased realism for the perception of the scene contents. The primary goal of haptic introduction into the DTI application is to allow the user to interact with both the volumetric and DTI data in a better way.

#### *3.4.1. Application Environment*

The haptic device that was used in this project was the Phantom Omni from SensAble. Phantom Omni provides 6-DoF input (position + orientation) and 3-DoF of force-feedback. In order to control the haptic device, the CHAI-3D library was used. In addition, VTK and Qt were used for visualization and GUI, respectively.

Since VTK and CHAI-3D consume each much time for their own main-loops, Qt timers were used to provide the necessary time distribution for both loops. The correlation between the two loops is accomplished by Qt's signal-slot mechanism. Whenever an event occurs in the haptic environment, such as creation or modification of a new ROI, a Qt-signal is sent from the haptic loop, which is later caught by a QVTK widget for further processing. The integration of Qt, VTK and CHAI-3D has been the primal concern in this effort.

#### *3.4.2. Haptic Slicer*

In the haptic environment the first problem to be solved is the representation of the three perpendicular and discriminated slices, as they exist in VTK environment. As a result, the user will be able to move those slices in the local coordinate system according to this specific perpendicular schema. The basic model for these slices is stored as an external WRL file. The haptic environment loads the above file during the initialization through a Qt-signal and afterwards reads the position of the slices from the VTK. This position is in the form of a vector $\vec{P}_{VTK} = (p_x, p_y, p_z)$ and the values $p_x$, $p_y$, $p_z$ correspond to the position of the plane $XY$, $YZ$ and $XZ$, respectively. As mentioned before, the user will be able to rotate the entire scene using the haptic cursor. This means that if the user has already changed the position of the slices in VTK environment before the initialization of the haptic environment or has rotated the entire scene using the haptic device, the application should be able to move the slices in the appropriate motion direction. One should wonder here how can the haptic understand the direction in which should it move the slices. The haptic proxy, during collision detection, returns the 3D point in the local coordinate system. This information is not adequate in order to find the direction of motion for the slice with which the proxy has collided. One solution is to analyze the vector of the haptic cursor in order to find the direction according to which the user "pushes" the current slice. On could ask what happens if we have only collision detection and almost zero vector force in this point? A better solution for this problem would be to replace each one of the prototype slices with two discriminate slices on either side that are extremely close to the prototype slice's position. Thus, if for example the position of the $XY$ plane is $p_x$ then the two slices will have the positions $p_x - 0.005$ and $p_x + 0.005$ , respectively. The small distance between the slices makes this trick invisible to the user and enables the application to work properly without the need to store any immediate information regarding the rotations of the scene. It also doesn't depend on the rotations the user has performed through the VTK environment. In this case, the resulting wrl file consists of six slices. Each slice has a specific name p.e. for the $XY$ plane we have two slices: XY_Plane_Back and XY_Plane_Front. The two slices can be moved along the vector $\vec{V} = (0, 0, 1)$ by adding a positive number for the first one while adding a negative number for the second one, to their current positions. In case the user has loaded the haptic environment but has rotated the scene in VTK, a Qt-signal is enough for the calculation of the new positions in order for the application to obtain balance between the two autonomous environments. The range of this position vector depends on the size of the volumetric dataset, which has been loaded in the VTK environment. To overcome this limitation, the wrl file includes slices of size 2, meaning that each slice could be positioned between $[-1, 1]$. Thus, the vector $\vec{P}_{\mathrm{VTK}}$ is transformed in the haptic environment to $\vec{P}_{\mathrm{haptic}}$ as follows:

$$\vec{P}_{\mathrm{haptic}} = (\frac{p_x}{N_z}, \frac{p_y}{N_x}, \frac{p_z}{N_y}) \tag{6}$$

where $N_z$, $N_x$, $N_y$ are the z,x,y dimensions of the plane perpendicular correspondingly to each slice. When the user rotates the scene in the local coordinate system of the slices, the coordinates remain unaltered, so the vectors along which the slices that could be moved do not changed.

The haptic slicer application consists of two display windows: the first window is an OpenGL window that contains the haptic scene whereas the second one is a VTK-window that contains the main environment for the DTI application.

In order to control the VTK world, several objects are placed into the haptic scene. Three orthogonal planes in the haptic world represent the slices in the VTK-world. Pushing these slices without pressing a key rotates the scene, while pushing them while pressing the key, translates the planes along that direction so that different slices in the VTK-world can be seen.

#### *3.4.3. Haptic ROI Manipulations*

The second aim of the haptic environment was the creation of the ROI's. By default, a ROI is a sphere which is provided to the system as an external wrl file. In this point we should mention that in the beginning the slices consist of a separate and autonomous object in the local coordinate system. When the user loads a ROI then the sphere becomes a child of this object and it is placed in an default position. When the haptic device collides with an ROI and the user presses the button in the haptic device then the current ROI becomes a child of the world object and afterwards it is moved accordingly by the haptic cursor. In that way, the user is able to "capture" a ROI, with the haptic, and can rotate the scene in order to find the desirable without placing the ROI somewhere else in the scene. The realistic perception of the 3D scene is one of the main advantages of the haptic device. When the user chooses an appropriate position for the ROI, he releases the button of the haptic device and thus the ROI becomes again a child of the local coordinate system now having the position the user decided. As mentioned before, the placement the ROI results in a different coordinate system whose child we wish to become, having a specific position.

The next step is to use the haptic to place the ROI mesh in a specific location $\vec{P}_{\mathrm{roi}}$ and afterwards set the ROI mesh as a child of the slice system, by locating it in the same position $\vec{P}_{\mathrm{roi}}$. To achieve that we have to set the local position of ROI mesh, which is relative to the slice system, equal to the global position of the proxy, since the position of an object is relative to its parent. If the slice system has no rotation, the two mentioned positions should be the same., If the slice system has a rotation, we need to compute the position of the proxy relative to slice system's frame and set ROI mesh's position equal to that. The slice system is constantly moving so we need to calculate its position for every iteration. Let $\vec{P}_{\mathrm{roi,final}}$ be the position of the ROI in the slice system local coordinate system, $R_{\mathrm{scene}}$ be the global rotation matrix of the scene and $\vec{P}_{\mathrm{device}}$ be the global position of the device. Then the appropriate transformation is defined as:

$$\vec{P}_{\mathrm{roi,final}} = [R_{\mathrm{scene}}]^T * (\vec{P}_{\mathrm{device}} - \vec{P}_{\mathrm{roi}}) \tag{7}$$

where $[.]^T$ denotes the transpose of a matrix.

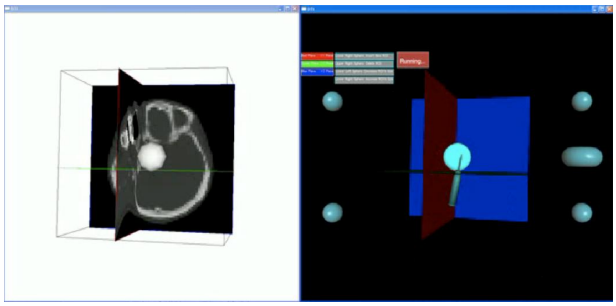

Figure 3: *The snapshot of the haptic interface demonsration. The left panel is the VTK display which works in parallel with the right panel which is a graphical representation of the data volume, three orthogonal slices and the spherical ROI. The tip of the 3D haptic pointer is shown touching (holding) the ROI for positioning.*

Apart from creating a ROI, the user is able to destroy, enlarge or shrink a ROI by selecting it with the haptic cursor and applying the desired procedures. At the right part of the haptic world there are two spheres that can be used for the addition and removal of the sphere-shaped ROIs. When the bottom-right sphere is selected, a ROI is added to the center of the scene. This ROI can be translated by touching it, pressing the haptic button and moving it around the scene. If the ROI is placed inside the bounding box of the volumetric data, it will stay there. But if the user tries to place the ROI outside the bounding box, it will automatically be placed at the center of the scene. In order to remove the ROI is removed, someone should drag and drop it to the upper thus erasing it from the scene.

At the left part of the haptic world there are two additional spheres that are used to increase and decrease the size of the ROI. By selecting the ROI and dropping it at the upper sphere it grows. If it is dropped at the lower sphere, the ROI shrinks.

The step factor for changing the size of a ROI is determined from the user. For the time being, the shape of the ROI's are just spheres and their shapes cannot be modified. Figure 3 shows a screenshot of the haptic interface demonstration.

## 4. DISCUSSION

The DTI project's goals can be grouped into three: The implementation of a baseline application, the implementation of the STAC standard and the development of haptic interaction tools. The development was done within the VAVframe framework. VAVframe is a C++ library that uses VTK, ITK and Qt. VAVframe has two goals: To provide a unified and standardized environment for code development and documentation, to function as a repository of the R&D at VAVlab (`http://www.vavlab.ee.boun.edu.tr`) for the continuation and progress of research efforts. It is a common problem of research labs that many invaluable contributions that cost significant time and money, become unusable as soon as the individuals who developed them move. VAVframe aims at avoiding this problem. The project team achieved all goals, yet the integration of these components could not be completed during the workshop.

The major problem that we have encountered was the integration of the VTK library, that VAVframe uses extensively, and the open-source CHAI-3D library used for haptic devices. Part of the project team worked exclusively on finding solutions for this integration. The current implementation is a proof-of-concept application which allows the users to rotate 3D volume, slide cross-sectional planes, position a spherical ROI and change its size. One of the unfulfilled goals was to develop haptic interface to modify the shape of 3D ROI. Using FEM for shape-modification seems a promising approach but it will certainly add some overhead due to real-time calculations. This is left for future work. In addition to the haptic controls discussed so far, haptics could also be used to move along individual fibers (like a bead sliding along a space-curve) for enhancing the perception of the topological properties of the fibers by the user, like curvature, torsion of the curve representing the fiber. In such a case, the haptic device could be used as a play-back device that moves the user's hand along the fiber or as a passive device that applies constraint forces whenever the haptic cursor tends to move away from the curve. Another option would be to send high force-magnitudes to the haptic device in regions where the fiber intensity is high so that the user can "feel" regions of high neural activity & concentration in the brain.

The base implementation and integration of the STAC with the software was completed. However, due to the unforeseen delays in the development of haptic interface, we could not demonstrate the final goal of the project. The goal was to demonstrate the use of real-time haptic ROI manipulations (shaping and positioning) with tractography results within a dynamic ROI query paradigm. The target application was envisioned to display the fiber tracts that intersect with a 3D ROI which is manipulated with a haptic device, in real time. Although this goal could not be achieved, the invaluable know-how that was developed during the workshop regarding the integration of different software libraries is a big step forward.

## 5. CONCLUSION

A proof-of-concept DTI application was aimed during the project. Several issues, including the integration of STAC, implementation of DWI to DTI conversion (tensor estimation), integration of CHAI-3D and VTK through Qt were solved during the project. Although the final application could not be completed, the know-how generated during the workshop is invaluable. Current efforts are concentrated on the integration of these components with the addition of original DTI analysis algorithms.

The project was run by VAVlab, Boğaziçi University, İstanbul, Turkey, in collaboration with Stanford University, School of Medicine, Department of Radiology, LUCAS MRS/I Center, CA, USA, University of Valladolid, Valladolid, Spain and Aristotle University of Thessaloniki, Thessaloniki, Greece.

## 6. ACKNOWLEDGMENTS

This work is conducted as part of EU $6^{th}$ Framework Programme, SIMILAR NoE, eNTERFACE 2007 workshop organized at Boğaziçi University, İstanbul, Turkey. The project is in part supported by SIMILAR NoE grants and Tubitak KARIYER-DRESS (104E035) research grants.

## 7. REFERENCES

[1] P. Basser, "Inferring microstructural features and the physiological state of tissues from diffusion-weighted images", *NMR Biomed*, vol. 8, pp. 333–344, 1995. 1

[2] P. Basser and C. Pierpaoli, "Microstructural and physiological features of tissues elucidated by quantitative diffusion tensor MRI", *J. Magn. Reson.*, vol. B 111, pp. 209–219, 1996. 1

[3] R. Bammer, B. Acar, and M. Moseley, "In vivo MR Tractography Using Diffusion Imaging", *European J. of Radiology*, vol. 45, pp. 223–234, 2002. 1

[4] A. Brun, M. Martin-Fernandez, B. Acar, E. M. noz Moreno, L. Cammoun, A. Sigfridsson, D. Sosa-Cabrera, B. Svensson, M. Herberthson, and H. Knutsson, "Similar Tensor Arrays - A framework for storage of tensor data", tech. rep., Las Palmas, Spain, 2006. 1, 2

[5] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit. An Object Oriented Approach to 3D Graphics.* Kitware, Inc., 2002. 4

[6] "The Open Source Haptics Project", August, 8 2002. http://www.chai3d.org. 4

[7] J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt 3*. Pearson Education, 2004. 4

[8] C. R. Tench, P. S. Morgan, M. Wilson, and L. D. Blumhardt, "White Matter Mapping Using Diffusion Tensor MRI", *Magnetic Resonance in Medicine*, vol. 47, pp. 967–972, 2002. 5

## 8. BIOGRAPHIES

**Murat Aksoy** is a PhD student at Stanford University LUCAS MRS/I Center, CA, USA.
Email: maksoy@stanford.edu

**Neslehan Avcu** graduated from Dokuz Eylül University, Department of Electrical and Electronics Engineering, in 2006 and currently she is a MS student in Dokuz Eylül University, The Graduate School of Natural and Applied Science, Electrical and Electronics Engineering Department. She has been a scholarship student, funded by TUBITAK since 2006. She is now a research assistant in Control Laboratory, and researcher in Control and Intelligent Systems Research Laboratory, DEU, Department of Electrical and Electronics Engineering. Her research interests include ANNs and their signal processing applications, cognitive science, fuzzy logic and pattern classification.
Email: neslihan.avcu@deu.edu.tr

**Susana Merino-Caviedes** graduated in Telecommunications Engineering from the University of Valladolid in 2005, and she is currently a PhD student in the aforementioned university. She is a member of the Image Processing Laboratory at the University of Valladolid. She is a participant of the SIMILAR NoE and a national research project focused on DTI. Her research interests are medical image processing, DTI visualization, user interfaces, image segmentation and level set algorithms.
Email: smercav@lpi.tel.uva.es

**Engin Deniz Diktaş** was born in İstanbul-Turkey, in 1978 and is currently a PhD student at Computer Engineering Department at Boğaziçi University. He received his BS degree in Mechanical Engineering and MS degree in Systems & Control Engineering from Boğaziçi University. His research interests include computer graphics, computational geometry, haptic rendering & interfaces and real-time simulation.
Email: denizdiktas@gmail.com

**Miguel Ángel Martín-Fernández** received his M.S. degree in Electronical Engineering from the University of Valladolid in 1999, and currently he is a teaching assistant and PhD student at the same received University. He has participated in several national and international research projects as member of the Image Processing Laboratory at the University of Valladolid. His research work is related to medical image processing, specially registration algorithms and analysis of medical images.
Email: migmar@tel.uva.es

**Sıla Girgin** was born in Bayburt, Turkey in 1982. She received the BS degree from the Department of Computer Engineering of Boğaziçi University, in 2005. She is currently a researcher and MS student at the same university. She is a member of VAVLab (Volumetric Analysis and Visualization Laboratory) in the Departments of Electrical and Electronics Engineering and MediaLab in the Department of Computer Engineering, at the Boğaziçi University. Her current interests are medical information visualization, computer graphics and computer vision.
Email: silagirgin@gmail.com

**Ioannis Marras** was born in Karditsa, Greece in 1980. He received the BS degree from the Department of informatics of Aristotle University of Thessaloniki in 2005. He is currently a researcher and teaching assistant and studies towards the Ph.D. degree at the Department of Informatics, in the Artificial Intelligence Information Analysis (AIIA) laboratory, at the Aristotle University of Thessaloniki. His current research interests lie in the areas of medical image analysis, signal and image processing, pattern recognition and computer vision.
Email: imarras@aiia.csd.auth.gr

**Emma Muñoz-Moreno** received her M.S. degree in Electronical Engineering from the University of Valladolid in 2003, and currently she is a PhD student at the same received University. She has participated in several national and international research projects as member of the Image Processing Laboratory at the University of Valladolid. Her research work is related to medical image processing, specially registration algorithms of DT-MRI and surgical simulation.
Email: emunmor@lpi.tel.uva.es

**Erkin Tekeli** was born in İzmit, Turkey in 1980. He received the BS degree from the Department of Mechatronics Engineering of Sabanci University, in 2004 and his MS degree from the Department of Electrical and Electronics Engineering and Computer Sciences of Sabanci University, in 2007. He is currently a Phd. student at the Department of Electrical and Electronics Engineering of Boğaziçi University. He is a member of VAVLab (Volumetric Analysis and Visualization Laboratory) in the Departments of Electrical and Electronics Engineering at the Boğaziçi University. His current interests are computer vision, medical image processing, shape spaces, content based image retrieval, texture analysis and segmentation.
Email: erkin.tekeli@boun.edu.tr

**Roland Bammer**, PhD, is an assistant professor at Stanford University, LUCAS MRS/I Center, CA, USA.
Email: rbammer@stanford.edu

**Burak Acar** was born in Izmir in 1972. He received his BS, MS and PhD degrees, all in Electrical & Electronics Engineering, from Bilkent University, Ankara, Turkey, in 1994, 1996 and 2000, respectively. He worked on ECG signal processing during his PhD. He worked at University of London, St. George's Hospital Medical School, London, UK, between 1998-1999. He was at Stanford University Medical School, Department of Radiology, 3D Lab, CA, USA, between 2000-2003, where he worked on medical image analysis and computer aided detection/diagnosis. He joined Boğaziçi University, Electrical & Electronics Engineering Department in 2003. His main research interests are 3D data (ranging from images to tensors) analysis and visualization with emphasis on medical applications. Further information can be found at `http://www.vavlab.ee.boun.edu.tr`.
Email: acarbu@boun.edu.tr

**Marcos Martin-Fernandez** received the Ingeniero de Telecomunicacion degree and the PhD degree from the University of Valladolid, Valladolid, Spain, in 1995 and 2002 respectively. He is Associated Professor at the ETSI Telecomunicacion, University of Valladolid where he is currently teaching and supervising several Master and PhD students. From March 2004 to March 2005, he was a Visiting Assistant Professor of Radiology at the Laboratory of Mathematics in Imaging (Surgical Planning Laboratory, Harvard Medical School, Boston, MA). His research interests are statistical methods for image segmentation and filtering in multidimensional (tensor) signal processing. He is also with the Laboratory of Image Processing (LPI) at the University of Valladolid where he is currently doing his reseach. He was granted with a Fullbright fellowship during his visit at Harvard. He is reviewer of several scientific journals and member of the scientific committee of the IEEE International Conference on Image Processing (ICIP), the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), European Signal Processing Conference (EUSIPCO) and IEEE International Symposium on BioInformatics and BioEngineering (BIBE). He also participates in fellowship evaluation for the National Competence Centre in Biomedical Imaging (NCCBI), Switzerland. He has more than 50 papers in scientific Journals and Conferences.
Email: marcma@tel.uva.es

**Ali Vahit Sahiner** was born in Ankara, Turkey in 1956. He received his PhD. in Computer Science from University of Westminster, London, UK, in 1991. He is currently working as an instructor at Boğaziçi University, İstanbul, Turkey. His interests include Computer Graphics, Visualization and Animation, Video and TV technologies, Graphic design and Typography.
Email: alivahit.sahiner@boun.edu.tr

**Suzan Üskudarli** is a faculty member of The Computer Engineering Department at Boğaziçi University. After receiving her doctorate degree at University of Amsterdam, she worked in Palo Alto, California. She joined Boğaziçi University in 2005. Her interests include software engineering and cooperation technologies. She is part of the VAVlab team, where she contributes to the software development aspects.
Email: suzan.uskudarli@boun.edu.tr