A TESTING FRAMEWORK FOR FIBER TRACTOGRAPHY

Langping (Kevin) Wei

Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand

Abstract

The in vivo visualization of the complex human brain is always a tremendous challenge for biomedical imaging tools. Diffusion Tensor Imaging (DTI) is a new imaging technique based on monitoring water molecule diffusions in 3D space. Fiber tractography is the process of extracting nerve fiber tracts. In combination with DTI, fiber tractography can be used for medical research and the diagnosis of various diseases.

A number of algorithms have been proposed for fiber tracking. However, few software tools have been developed for testing these algorithms. This project aimed to develop a comprehensive testing framework for the various fiber tracking algorithms.

Based on research and user testing, FiberTK is a toolkit developed for creating and visualizing nerve fiber tracts. Various fiber tracking algorithms can be applied in the program and results are to be compared and analyzed visually.

It is concluded that the project has been carried out in an appropriate manner and the resulting software has met its original requirement. However, future developments such as Automatic Analysis Report are possible and encouraged for the project.

1. Introduction

Diffusion Tensor Imaging (DTI) is a new imaging technology based on measuring the intrinsic properties of water molecule diffusing in 3D space [1]. An important field of study related to DTI is fiber tractography, which is defined as the process of extracting fibers from soft fibrous tissues such as nerves, muscles and tendons [2]. In combination with DTI, fiber tractography is commonly used in medical diagnostics and research [3].

There have been many methodologies suggested for tracking nerve fiber tracts in the brain. However, few software tools have been developed for comparing these methods [1]. Based on the properties of nerve fiber tracts, this project aimed to develop a toolkit for the comparison of various fiber tracking algorithms.

The reason for developing such a testing framework was that the ability to evaluate and test various existing fiber tracking algorithms is useful in medical research, especially when one wants to apply one or more of these algorithms to a particular problem [3]. Knowing how well a particular algorithm performs under a particular circumstance is helpful in determining its suitability under that particular situation.

Developed in C/C++ in conjunction with Tcl/Tk, FiberTK is a toolkit developed in this project that allows users to create simulated DTI data and run various fiber tracking algorithms. Results are to be visually compared and analyzed.

FiberTK was being developed as part of the Year 4 Software Engineering Project at the University of Auckland (UoA).

2. Project Background

2.1. Diffusion Tensor Imaging (DTI)

Biomedical imaging is an essential tool for medical research and analysis. The complexity of the human brain presents significant challenges in *in vivo* visualization of the brain structure using such tools [3]. Magnetic Resonance Diffusion Tensor Imaging (MR-DTI) is a new imaging technique and it is based on measuring the extent of water molecules diffusing in brain fibers [1]. Images in DTI are created by monitoring the phase differences among water molecules during diffusion. These phase differences occur due to the different magnetic fields experienced by different water molecules when diffusing in different directions [2].

DTI can be used to differentiate various types of structural tissues in the brain, one of which is the nerve fiber tracts [1]. Consisting of millions of parallel-orientated nerve fibers, nerve fiber tracts can be easily identified based on the fact that water diffusion in nerve fibers is single-directional [1]. DTI is capable of providing directional information for fiber tracking. Figure 1 below illustrates fiber tracts in DTI using the 4th order Runge-Kutta method [2].

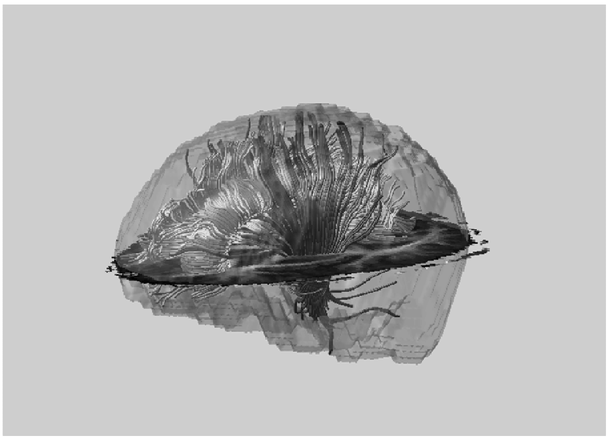


Figure 1: Fiber Tracts Generated Using the 4th Order Runge-Kutta Method. Reproduced From [2]

2.2. Fiber Tractography

Fiber Tractography is an important field of study related to DTI. It is the process of extracting nerve fibers. There have been a number of tractography techniques developed in the past that simulate the basic structure of nerve fiber tracts in the brain. These include streamline fiber tracking technique, diffusion tensor deflection strategy, probabilistic Monte-Carlo method, fast marching tractography based on level set principles, and diffusion simulation-based tractography [4]. All of these techniques vary in terms of performances due to the nature of the techniques themselves. Figure 2 below illustrates the extraction of nerve fibers from a DTI data set.

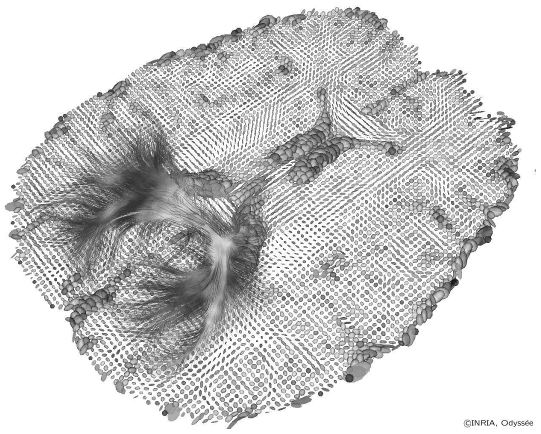


Figure 2: Extraction of fibers in the splenium of the corpus callosum from DTI data [17].

Together with DTI, fiber tractography is extremely useful in biomedical research and the diagnosis of various neurodegenerative diseases such as multiple sclerosis and schizophrenia [3].

2.3. Previous work

There have been a number of exiting software packages for Fiber Tractography. However most of these packages have a strong emphasis on visualizations because they were built to fit a different purpose (i.e. they are unsuitable for the purpose of testing different fiber tracking algorithms).

One of the representative software packages in this field is the 3DMRI [5], which is essentially a C++ built toolkit for visualizing fiber tracts in an isosurface of human brain [5]. Images are generated from the brain MRI scans. Figure 3 below shows a screenshot of the 3DMRI software package.

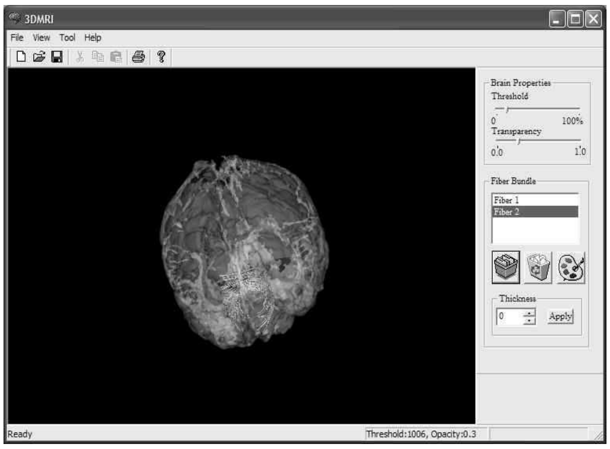


Figure 3: Screenshot of 3DMRI Fiber Visualization Toolkit. Reproduced from [4]

Other similar applications include 3D Slicer [6], DTIChecker [7] and DdDTI [8]. However, due to the nature of these software packages they have not incorporated the ability of testing any of the fiber tracking algorithms.

3. Project Design

3.1. Specification and Structural Design

The project aimed to develop a testing framework for evaluating various fiber tracking algorithms. This testing framework was developed in the form of a Graphical User Interface (GUI). Users are able to create simulated fiber objects which are represented by tubes in 3D space. Users can also manipulate the artificial fibers in a similar fashion to manipulating a word document (i.e. they could cut, copy, paste, save and load fibers). Once a user is satisfied with the fibers, various fiber tracking algorithms can be applied. Furthermore, results can be displayed, compared and analyzed visually.

According to these user requirements, the structural design of FiberTK is shown in Figure 3.1 below.

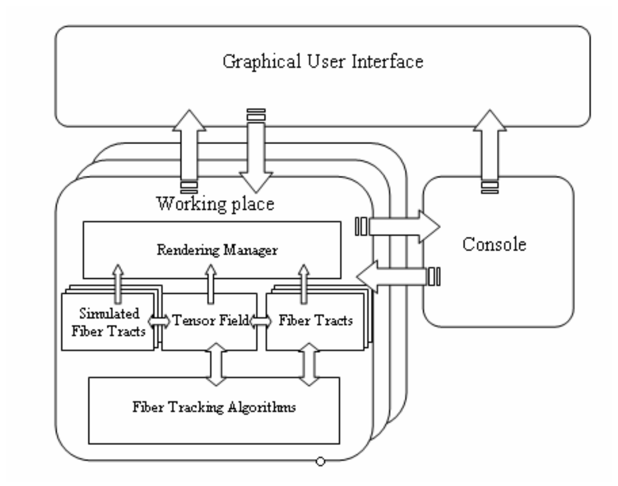


Figure 4: Structural Design of FiberTK.

The structure of FiberTK consists of two distinct but highly relative layers. The Graphical User Interface (UI) layer interacts with the users by observing user actions such as button press and mouse movements. It is also responsible for output display of the program.

The layer underneath consists of a number of objects. Simulated fiber tracts and tensor fields are to be defined by users. Various fiber tracking algorithms can then be applied to create fiber tracts. User-created (simulated) fiber tracts, tensor fields and fiber tracts are all managed by the Rendering Manager, which is responsible for rendering outputs to the UI.

3.2. User Interface Design

Based on research done earlier in the project, the user interface of FiberTK is designed and developed. To some extent, this interface is essentially a simplified 3D drawing environment/platform where users can create /modify fiber tracts as well running various tracking algorithms. Therefore, similar software packages were studied. These included Adobe PhotoShop [9], AutoDesk 3D Max [10] and ProDesktop [11].

A number of design decisions were made according to similar 3D drawing software packages.

3.2.1. Overall Design

Like many software packages, FiberTK has a menu bar, two dockable toolbars and the main working space. Each drawing window is implemented as an internal window inside the main working space and user-defined fiber tracts can be copied and pasted across different drawing windows. Each internal drawing window can be created, maximized, minimized and closed.

The shape of a fiber is defined by a curve in 3D space. And a curve is defined by a set of control points. By allowing users to define a set of control points in 3D space, a fiber tract can be precisely defined.

3.2.2. Internal Drawing Window

A number of design decisions were also made with regard to the design of the internal drawing windows.

Due to the fact that fiber tracts are to be drawn on a 3D space, an intuitive way of allowing users to specify control points in 3D space played an important role in the entire design of the program.

Control points are to be defined and drawn on viewports. There are two fundamental types of viewports available for 3D scene rendering, namely the single viewport and the multiple viewport. Single viewport is the viewing of designed object from the perspective view while multiple viewport splits the screen into four different windows, each displaying a different view (i.e. front, side, top and perspective view). Figure 5 below demonstrates the difference between single and multiple viewports.

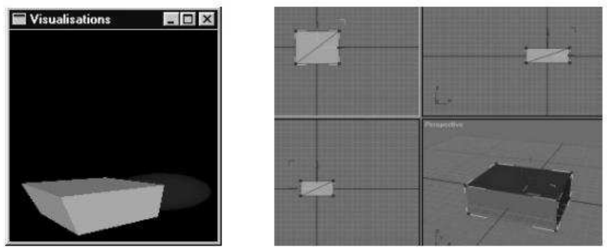


Figure 5: Single Viewport (Left) vs. Multiple Viewport (Right). Adapted from [12]

Single viewport has the advantage of ease of use while multiple viewport is relatively more difficult to understand. Multiple viewport, on the other hand, is much more powerful in terms of accuracy because experienced users can refer to different views at the same time. Further more, multiple viewport gives users better control over the entire 3D model by allowing them to manipulate the model in all directions with precision. A series of user testing were carried out and it was found that multiple viewport was a more intuitive approach. It was therefore decided that the multiple viewport approach was more appropriate.

While multiple viewport is the default structure, users can also switch to single viewport for viewing by clicking on the "Change Viewport" button on the toolbar.

3.2.3. Console Window

A console window is also implemented in the program for displaying messages to users. Messages to be displayed include confirming user actions (e.g. when a new file has been created), tips to assist users, and error messages (displayed in red) when a user has made a mistake. The console window helps communicating with users better.

3.3. Curve Algorithms

There are a number of curve drawing algorithms available for rendering a curve from points. A set of control points define a curve, a curve (in 3D space) and a radius define a (uniform) tube. Deciding what curve algorithms to use for drawing curves plays another important part on the application. Four different types of curve algorithms were studied and evaluated. Table 1 below illustrates the properties of different curve drawing algorithms.

	Bezier	Hermite	B-Spline	Catmull-Rom Spline
Continuity	C1	C1	C2	C2
Local control	no	no	yes	yes
Curve segment lies within the convex hull	yes	yes	yes	no
Interactive curve modeling	Good	Good	Very good	Good
Interpolate a number of position	Poor	Poor	Poor	Best
Interpolate with tangent control	Often used	Often used	Good	Good

Table 1: Table for different curve drawing algorithms

It was decided that both Uniform B-Spline Curves algorithm and Catmull-Rom Spline Curve algorithm are to be implemented due to their high continuity and their abilities of local control. More explanation will be presented below.

3.3.1. Uniform B-spline Curve Algorithm

The Uniform B-Spline Curve is best for interactive curve modeling [13]. A curve implemented using Uniform B-Spline is defined in such a way that neither of the control point will lie on the curve. B-Spline Curve gives good local control, which means that changing one of the control points on the curve will not have effects the entire curve. Figure 6 illustrates an example of a cubic B-spline curve implemented in FiberTK.

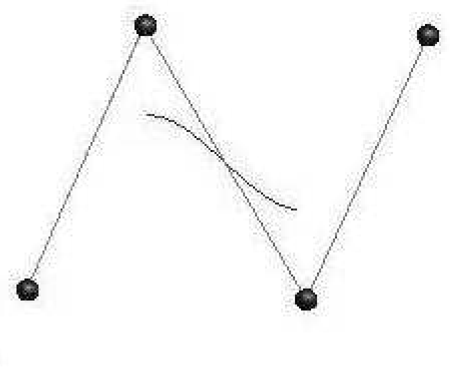


Figure 6: An example of a cubic B-Spline curve which is generated by four control points.

The B-spline curve algorithm implemented in FiberTK has a function degree of three, which means that four control points are needed in order to define one curve segment. In terms of polynomial function of the curve, four terms with a dominant term of t^3 are present in the function. Each point on the curve can be calculated using the formula below:

$$P(t) = \sum P_i + B_4(t-i)$$

$$B_4(t) = \frac{1}{6}\begin{cases} t^3 & 0 \le t \le 1 \\ v(2-t) & 1 \le t \le 2 \\ v(t-2) & 2 \le t \le 3 \\ (4-t)^3 & 3 \le t \le 4 \\ 0 & otherwise \end{cases} \qquad (1)$$

$$where \quad v(s) = (3s^3 - 6s^2 + 4)$$

The formula can also be written in the form of matrix:

$$p(t) = (1 \quad t \quad t^2 \quad t^3)\frac{1}{6}\begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix}\begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} \qquad (2)$$

3.3.2. Catmull-Rom Spline Curve Algorithm

The Catmull-Rom Spline curve algorithm defines a curve by interpolating the curve through a set of control points. Similar to the B-Spline curves, it gives good local control. Figure 7 below shows an example of a cubic B-spline curve implemented in FiberTK.

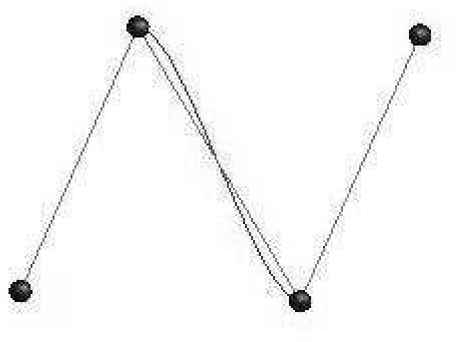


Figure 7: An example of a cubic Catmull-Rom Spline curve which is generated by four control points.

The Catmull-Rom Spline curve algorithm implemented in FiberTK also has a function degree of three. Each point on the curve can be calculated using the formula below:

$$C_4(t) = (-\frac{1}{2}t^3 + t^2 + -\frac{1}{2}t)P_1 + (\frac{3}{2}t^3 - \frac{5}{2}t^2 + 1)P_2$$
$$+ (-\frac{3}{2}t^3 + 2t^2 + \frac{1}{2}t)P_3 + (\frac{1}{2}t^3 - \frac{1}{2}t^2)P_4 \qquad (4)$$

The formula can also be written in the form of matrix:

$$p(t) = (1 \quad t \quad t^2 \quad t^3)\frac{1}{2}\begin{pmatrix} 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ 1 & 3 & -3 & 1 \end{pmatrix}\begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} \qquad (5)$$

3.4. Simulated Fiber Tracts

The simulated fiber tract can be generated from the user-defined curve to give users an *in vivo* visualization of the fiber. A simulated fiber tract is represented by a cylindrical tube shape constructed from a series of cross-section circles. Figure 8 below illustrates an example of a simulated fiber tract.

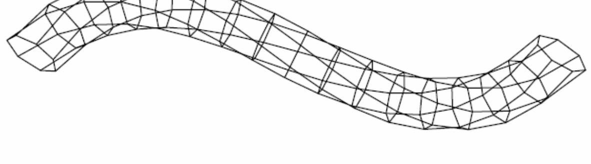


Figure 8: A wire framed view of a three dimensional representation of a fiber tract.

In order to achieve a uniform tube shape, the alignment of each cross-section plane plays a significant role. The tangent, the principle normal and the binormal of the curve must firstly be calculated in order for each cross-section plane to align with its neighbours properly.

The tangent vector **T** can be obtained by differentiating the first derivatives of both of the curve algorithms. From equation (1) one can see that:

$$P(t)' = \sum P_i + B_4(t-i)'$$

$$B_4(t)' = \begin{cases} 3t^2 & 0 \le t \le 1 \\ -v'(2-t) & 1 \le t \le 2 \\ v'(t-2) & 2 \le t \le 3 \\ -3(4-t)^2 & 3 \le t \le 4 \\ 0 & otherwise \end{cases} \qquad (6)$$

$$where \quad v'(s) = (9s^2 - 12s)$$

And from equation (4):

$$C_4(t)' = (-\frac{3}{2}t^2 + 2t + -\frac{1}{2})P_1 + (\frac{9}{2}t^2 - 5t)P_2$$
$$+ (-\frac{9}{2}t^2 + 4t + \frac{1}{2})P_3 + (\frac{3}{2}t^2 - t)P_4 \qquad (7)$$

The tangent of any point on the curve can therefore be calculated using the equations above. The principle normal vector (**V**) and the binormal vector (**U**) can also be evaluated easily when "T" is known:

$$U = T \times UP \qquad (8)$$
$$V = T \times U \qquad (9)$$

Where UP refers to the Up Vector [0, 1, 0].

These three vectors can then be used to rotate the cross-section plane to form a uniform tube. Firstly a unit circle is created and scaled with a specified radius. The circle is then rotated using the T, U and V vectors. Finally the circle is translated to be positioned in the curve. Figure 9 below illustrates this process:

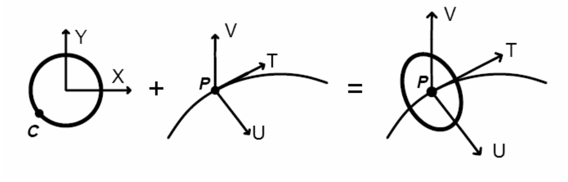


Figure 9: Positioning and orienting a cross-section [14].

However, in practice it was found that when the angle between the tangent vector (T) and the up vector (U) gets too small or gets close to 180 degree, the binormal vector calculated using the equation (8) will not be orthogonal to the tangent vector any more. This meant that the rotation of the circle plane would not be accurate which in turn affects the uniformity of the fiber. A solution to this would be to substitute the up vector (U) with the previous principle normal vector (V_{i-1}). The new equation is presented below:

For P_0:

$$U_i = T_i \times UP \tag{10}$$

For P_i where $i \neq 0$:

$$U_i = T_i \times V_{i-1} \tag{11}$$

$$V_i = T_i \times U_i \tag{12}$$

Where UP refers to the Up Vector [0, 1, 0].

3.5. Tensor Field

The tensor field can be generated from the tangents of the fiber tracts created by users. Any tensor points that lie within the simulated fiber tracts will be deformed with a high diffusion factor along the direction of the tangents. Figure 10 below demonstrates an example from the actual FiberTK program.

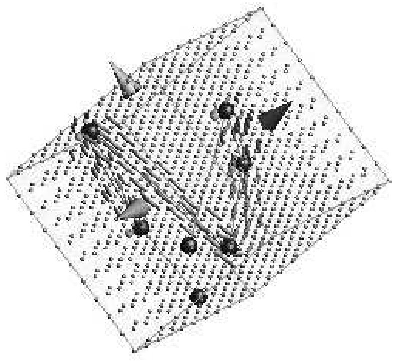


Figure 10: An example of a tensor field in 3D space.

The pseudo code in figure 11 below shows how this is achieved. Firstly, every single tensor point is checked to see if it lies within the simulated fiber tract. If this is true, three eigenvectors are calculated from the tangent (T), the principle normal (V) and the binormal (U) of the corresponding point. These eigenvectors are then applied to the point for re-scaling and orienting.

```
For every tensor point
    //tensor is the data structure for storing the matrix of diffusion tensor
    Reset tensor to 0
    For every fiber tract
        Find the closed point in distance
        Compute the distance between the tensor point and fiber tract
        If the distance is less than the radius
            Compute tangent, principle normal, and binormal vectors
            Compute tensor values
        End if
    End loop
End loop
```

Figure 11: Pseudo code for computing tensor field

The diffusion tensor **D** can be calculated as shown below.

$$D = S^{-1}\Lambda S \tag{13}$$

$$where \qquad S = \begin{bmatrix} \lambda_{1x} & \lambda_{1y} & \lambda_{1z} \\ \lambda_{2x} & \lambda_{2y} & \lambda_{2z} \\ \lambda_{3x} & \lambda_{3y} & \lambda_{3z} \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} e_1 & 0 & 0 \\ 0 & e_2 & 0 \\ 0 & 0 & e_3 \end{bmatrix}$$

When there is an intersection between two or more fiber tracts, their tensor eigenvalues and eigenvectors are added together to form a disc shape as illustrated in figure 12 below. Different tracking algorithms usually perform differently in this particular situation.

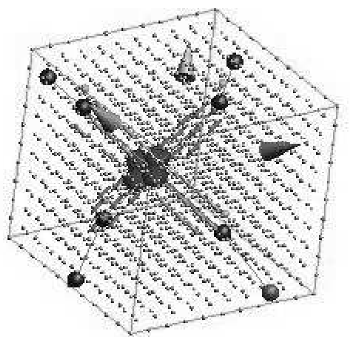


Figure 12: An example of two fiber tracts intersecting each other.

3.6. Fiber Tracking Algorithms

Once the tensor field is computed, various nerve fiber tracking algorithms can be applied to the fiber tracts. Three tracking algorithms were implemented in this project, namely the stream line, tensor deflection and tensor line algorithms [15]. Figure 13 below shows the different results from different tracking algorithms.

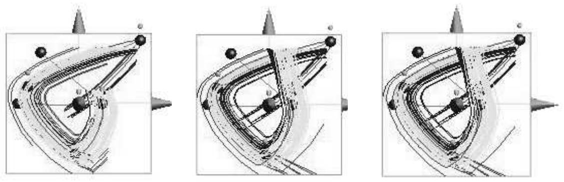


Figure 13: Various Fiber Tracking Algorithms (From Left to Right: Stream Line, Tensor Line and Tensor Deflection)

3.7. Implementation Language

FiberTK is implemented in a number of programming languages and APIs. The project aimed to develop a toolkit that runs on both Windows and Linux. As a result, the choices for implementation languages are important. After a significant amount of research and practices, five programming languages and packages were used in implementation.

3.7.1. C/C++ and OpenGL

C/C++ is always the first choice for graphic applications due to its speed, full functionality and greater support for underlying hardware. The use of OpenGL is also specified in the project requirement since fiber tracking algorithms provided were also written in OpenGL. Both C/C++ and OpenGL are open source and both support multi-platform.

3.7.2. Tcl/Tk and TkOGL

Tool Command Language (TCL) and Toolkit (TK), commonly known as Tcl/Tk, is a powerful, platform-independent scripting language for building user interfaces [19]. Its ease of use and comprehension of support make it ideal for implementing the UI in this project.

TkOGL, which stands for OpenGL for Tk, is a software library written to support OpenGL in Tcl/Tk [20]. It is widely used in conjunction with Tcl/Tk for displaying OpenGL contents on the screen.

3.7.3. SWIG

The Simple Wrapper Interface Generator (SWIG) is an essential tool for communicating between the Tcl/Tk code and the underlying C/C++ code. To a wider extent, it integrates C/C++ with a variety of high-level programming languages and various common scripting languages such as Perl, Python, Tcl/Tk and Ruby by generating an interface file for these languages [21].

3.7.4. Software Architecture

The software architecture for FiberTK is shown in Figure 14 in the next page.

Data structure, various curve and tracking algorithms as well as the basic underlying structure for the program are implemented in C/C++ using Microsoft Visual Studio .Net framework. The source code is then compiled into a Dynamic Link Library (.dll) file in Windows or a Shared Object (.so) file in Linux using SWIG. SWIG is capable of wrapping all function calls in C/C++ as well as making it callable by other applications.

At the user interface level, Tcl/Tk is used for building the UI. At runtime, the library file is loaded into the program and user inputs become parameters to function calls into the library file. Any OpenGL output would be displayed the TkOGL widget.

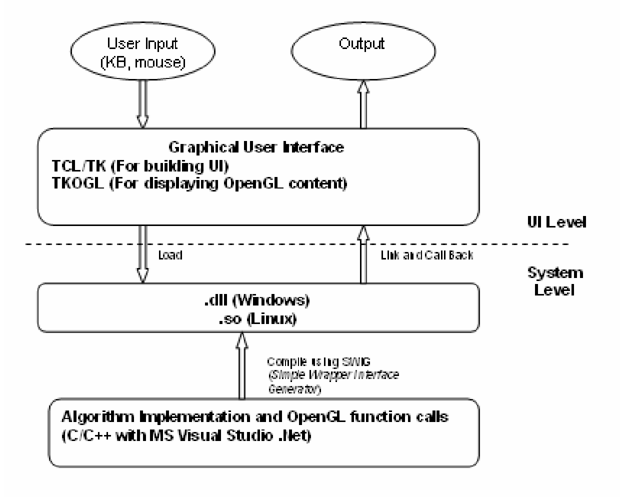

User Input (KB, mouse)
Output
Graphical User Interface
TCL/TK (For building UI)
TKOGL (For displaying OpenGL content)
UI Level
Load
Link and Call Back
System Level
.dll (Windows)
.so (Linux)
Compile using SWIG (Simple Wrapper Interface Generator)
Algorithm Implementation and OpenGL function calls (C/C++ with MS Visual Studio .Net)

Figure 14: Software Architecture of FiberTk.

4. Project Implementation

4.1. Graphic User Interface Overview

Figure 15 in the next page shows a screenshot of the graphical user interface for FiberTK. There are five main components in the UI: the menu bar, the tool bar, the tool box, the console and the main working window.

4.1.1. Menu

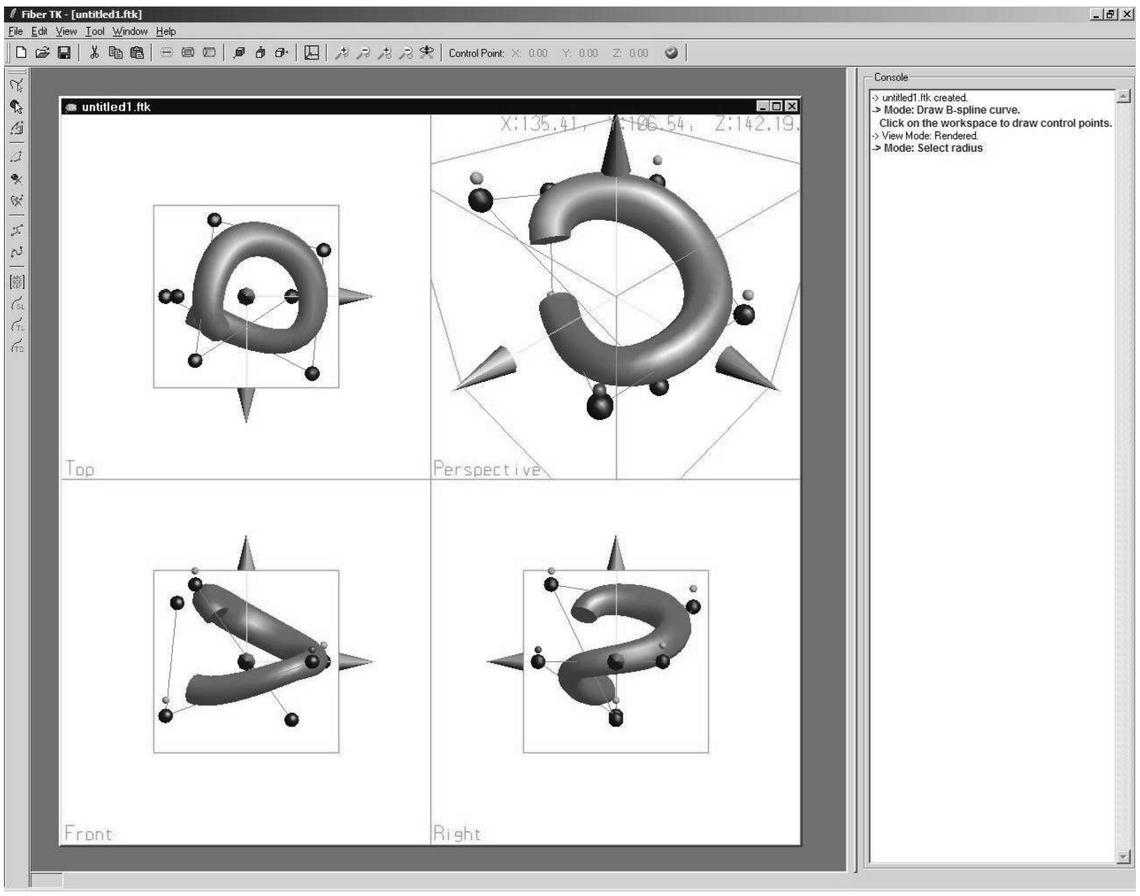
The menu bar is consisted of a list of drop-down menus including File, Edit, View, Tool, Window and Help. Each menu also contains a list of sub-menus for various functions relative to the software.

4.1.2. Toolbar

The dockable toolbar positioned horizontally is used for frequently-used functions. Similar functions are grouped together and different function groups are separated by separators.

There are seven main function groups in the toolbar:

- New, Open and Save functions
- Cut, Copy and Paste functions
- Different view mode functions (Skeleton View, Wireframe View and Rendered View)
- Different viewing angle functions (Viewing from X, Y and Z axis)
- Changing Viewport function (Switch between single viewport and multiple viewports)
- Zoom in, Zoom Out and Rotation Mode functions
- Defining control point position function


Fiber TK - [untitled1.ftk]
File Edit View Tool Window Help
Control Point: X: 0.00 Y: 0.00 Z: 0.00
untitled1.ftk
X:135.41, Y:106.54, Z:142.19
Top
Perspective
Front
Right
Console

```
-> untitled1.ftk created.
-> Mode: Draw B-spline curve.
   Click on the workspace to draw control points.
-> View Mode: Rendered.
-> Mode: Select radius
```

Figure 15: The Graphical User Interface of FiberTK

4.1.3. Toolbox

The dockable toolbar positioned vertically is responsible for drawing and manipulating fiber tracts. There are four main function categories in the toolbox.

- Selecting Functions: Select Entire Fiber, Select Control Point and Select Radius
- Insert and Delete Functions: Insert Control Point, Delete Control Point and Delete Entire Fiber
- Drawing Functions: Draw using B-Spline Algorithm and Draw using Catmull-Rom Spline Algorithm
- Tensor field functions: Generate Tensor field, Apply the three algorithms

4.2. Drawing a Fiber

Creating fiber tracts is made easy in this program by allowing users to define a series of control points in any of the working planes except for the perspective one. A curve segment will be generated automatically for every four defined control points. Users can also switch to different viewing modes or angles once a curve is defined.

The perspective viewport is for viewing only. Users are not allowed to define control points in this view mode simply because positions in perspective viewing are difficult to determine.

4.3. Manipulating a Fiber

Once a curve/fiber tract is created, it can be manipulated in different ways. Here's a list of functions associated with curves.

- Cut, Copy and Paste a curve/fiber
- Modify the position of the control point of the curve
- Modify the position of the entire curve
- Modify the radius of the fiber
- Delete a control point in the curve
- Delete the entire curve
- Insert a control point

4.4. Generating Tensor Field

Once the user is satisfied with the fiber created, the tensor field can be generated using the Generate Tensor Field function in the toolbox. All tensor points are displayed inside the tensor field as shown in Figure 10 above. Any tensor point that lies within the simulated fiber tract will be deformed with a high diffusion factor along the direction of the tangents.

Users can rotate, zoom in, and zoom out and the tensor field for viewing.

4.5. Running Tracking Algorithms

Once the tensor field is computed, users can apply various tracking algorithms to the fiber tracts. This is done by the last three buttons in the toolbox. Three tracking algorithms were implemented; the stream line, tensor deflection and tensor line algorithms.

These buttons were implemented as toggle buttons, which means that first clicking displays the corresponding result and the next clicking turns the display off. This gives users better control over the results they wish to view (i.e. a user might want to view result from the tensor line algorithm only and one could do this by switching all other views off)

Results from various tracking algorithms are to be compared and analyzed visually only at this stage.

5. Conclusions

- The project aimed to develop a comprehensive testing framework for testing different fiber tracking algorithms.
- Written in C/C++ and TCL/TK, FiberTK is toolkit developed for the project that can be used for generating and visualizing nerve fiber tracts.
- Various fiber tracking algorithms can be applied in the application. Results are to be compared and analyzed visually.
- Various design approaches were used to increase the usability of the software. Design decisions were made accordingly.

6. Future work

There are four possible future developments for this project.

- Further development in variable radius. At present, variable radius is implemented as linear interpolating in this project. A non-linear interpolation would give the curve a much smoother shape.
- More complex fiber shapes. At the moment, only a tube shape is implemented to represent real fiber tracts. In practice this is unrealistic because real fiber tracts usually consist of different shapes and their internal structure is far more complex than a simple tube. A more sophisticated structure to represent fiber tracts is strongly encouraged as a future work for this project.
- Analysis report. This project allows users to compare results from various tracking algorithms visually only, which is obviously insufficient and ineffective. Analysis reports could therefore be implemented. This could include a report that calculates the Standard Mean Error (SME) of the computed fiber tract compared to the actual fiber tract previously created by the user. In addition, the percentage of computed fiber tracts that are outside the simulated fiber tract can also be counted and act as a benchmark for the performance of a particular tracking algorithm.
- User defined starting points. At present, the starting point for the tracking algorithms are hard-coded into the underlying C++ code. Allowing users to specify starting points would be helpful to the users.

Acknowledgements

The student would like to thank the project supervisor, Dr Burkhard Wuensche for providing the valuable advices, support and guidelines to make this project successful.

Second Examiner, Mr. Chris Smaill for providing helpful feedback and advices on the project implementation as well as on the project reports.

Ms Jing Li for providing the necessary nerve fiber tracking algorithms and other relative information that is much helpful to the project.

Project partner Li Min (Kevin) Xu for his dedication, enthusiasm and contribution to the project.

And other software Engineering students including David Huai and Yin Zili for doing usability testing on the prototype of the graphical user interface.

7. References

[1] Burkhard, C. W. and Richard, L. (2004).The 3D Visualization of Brain Anatomy from Diffusion-Weighted Magnetic Resonance Imaging Data. Division for biomedical imaging and visualization, Department of computer science. University of Auckland.

[2] Acar, B. and Bammer, R. *MR-DTI FIBER TRACTOGRAPHY and Beyond. (2003).* Retrieved May 7, 2005: http://www.vavlab.ee.boun.edu.tr/ProjectsNEW/links_DTI.html

[3] Nimsky, C., Ganslandt, O., Hastreiter, P., Wang, R., Benner, Thomas P., Sorensen, A. G., and Fahlbusch, R. (2005). Preoperative and Intraoperative Diffusion Tensor Imaging-based Fiber Tracking in Glioma Surgery. *Neurosurgery*. January, 56(1):130-138.

[4] Jun, Z., Hao, J., Ning, K. and Ning, C. (2005). Fiber Tractography in Diffusion Tensor Magnetic Resonance

Imaging: A Survey and Beyond. Laboratory for high performance scientific computing and computer simulation, Department of computer science, University of Kentucky, Lexington, KY 40506-0046, USA

[5] Hua, K. *3DMRI* (2003). Retrieved May 7, 2005: http://cmrm.med.jhmi.edu/DTIuser/DTIuser.asp

[6] 3D Slicer. (2004). Retrieved May 7, 2005: http://www.slicer.org/

[7] DTIChecker (2004). Retrieved May 7, 2005: http://www.ia.unc.edu/dev/download/fibertracking/index.htm

[8] DdDTI. (2004). Retrieved May 7, 2005: http://neuroimage.yonsei.ac.kr/dodti/

[9] Adobe PhotoShop (2005). Retrieved May 7, 2005: http://www.adobe.com/products/photoshop/main.html

[10] AutoDesk 3D Max (2005). Retrieved May 7, 2005: http://www4.discreet.com/3dsmax/

[11] ProDesktop (2005). Retrieved May 7, 2005: http://www.prodesktop.net/

[12] Perspective View (2003). Retrieved May 7, 2005: http://www.vmbollig.de/msts/tut_en/construction_1/box_in_mitte.jpg

[13] Hill, F.S. (2001). *Computer Graphics Using OpenGL.* Second edition. Prentice Hall, United States of America.

[14] Bloomenthal, J. Calculation of Reference Frame along a Space Curve. Graphics Gems, Vol1.

[15] Li, J. (2005). An Analysis of Algorithms for In Vivo Fiber Tractography Using DW-MRI Data. The Computer Science Department. The University of Auckland.

[16] Beaulieu, C. (2002). The basis of anisotropic water diffusion in the nervous system – a technical review. NMR Biomed. 15:435-455.

[17] Center for Magnetic Resonance Research, University of Minnesota, Minneapolis, MN USA and at the CEA, SHFJ Orsay FRANCE.

[18] Burkhard, C. W. and Richard, L. (2004). Curves and Surfaces. Course notes for paper 715.415. University of Auckland.

[19] Welch, B. B. *Practical Programming in TCL and TK*. 3rd Edition. Prentice Hall PTR. 2000.

[20] TKOGL – A TK OpenGL widget. Retrieved May 7, 2005: http://tcltk.free.fr/tkogl/

[21] "SWIG" Retrieved September 11, 2005, from http://www.swig.org/