A TESTING FRAMEWORK FOR FIBER TRACTOGRAPHY

Limin (Kevin) Xu

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

Abstract

This report outlines a toolkit that has been developed for simulating DTI data as well as allowing the user to compare various nerve fiber tracking algorithms.

Identifying and visualizing the nerve fiber tracts in the human brain using biomedical imaging, such as diffusion tensor imaging (DTI) can improve the diagnoses, understanding and treatment of a large variety of diseases. The process of extracting nerve fiber tracts is called fiber tractography and it has been widely used to diagnose various neuro-degenerative diseases. There are several existing DTI visualization toolkits, however they are not designed for testing and comparing different nerve fiber tracking algorithms.

In has been concluded that the FiberTk toolkit is built which is capable of generating and visualizing user-specified nerve fiber tracts, evaluating various nerve fiber tracking algorithms. It is recommended to enhance the toolkit with various future developments including generating analysis report, creating more complicated nerve fiber shapes, allowing user to specify the starting points for nerve fiber tracking algorithms and non-linear interpolation for changing the fiber radius.

1. Introduction

In modern biomedical sciences, the in vivo identification and analysis of anatomical structures of the human brain still remains a tremendously challenging research field. Identifying and visualising the nerve fiber tracts in the brain using various biomedical imaging technologies, such as diffusion tensor imaging (DTI) can improve the diagnoses, understanding and treatment of a large variety of diseases. The process of extracting nerve fiber tracts which consists of millions of parallel nerve fibers is called fiber tractography [1]. It has been widely used to diagnose various neuro-degenerative diseases and to investigate the development of white matter tracts in adolescents and adults [2].

The aim of this project is to develop a toolkit that simulates DTI data as well as allowing the user to compare various nerve fiber tracking algorithms. It would involve design and development of a graphical user interface for drawing the nerve fiber tracts in a 3D environment and testing and representing the existing fiber tracking algorithms in a constructive manner. In the end, Fiber ToolKit (FiberTK) is developed, a toolkit for simulating DTI data and evaluating various nerve fiber tracking algorithms.

2. Project Background

2.1. Diffusion Tensor Imaging (DTI)

Diffusion tensor imaging (DTI), also known as diffusion-weighted MRI imaging (DWI) [1] or diffusion tensor magnetic resonance imaging (DT-MRI) [3, 4, 5] is a new imaging technology used in biomedical imaging. The principle is based on measuring the phase differences between spinning water molecules due to the fact that different molecules diffusing in different directions experience different magnetic fields [4]. The output is a set of Diffusion Weighted Images which is used to render the information about how water diffuses in biological tissues containing a large number of fibers, such as brain white matter, into intricate three dimensional (3D) representations of tissue architecture [5]. Thus, it can be exploited to visualize and extract information about the brain white matter and nerve fibers [6] in order to understand its inter-structure.

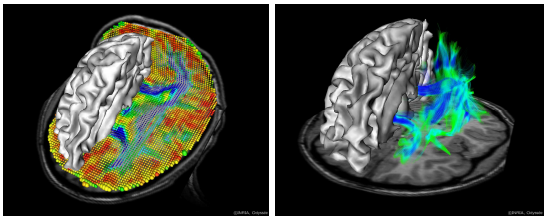

Figure 1: Left: white matter segmentation superimposed on anatomical and DTI data. Right: Major neural fibers bundles (corpus callosum, corona radiata) with an anatomical axial slice and segmentation of the white matter [7].

Figure 1 illustrates some of the three dimensional representation of the internal brain structures and tissue architectures which is generated from DTI data. All these images were generated from datasets acquired at the Center for Magnetic Resonance Research,

University of Minnesota, Minneapolis, MN USA and at the CEA, SHFJ Orsay FRANCE.

2.2. Fiber Tractography

Another important field of study related to DTI is fiber tactography. Fiber tractography is the process of extracting nerve fibers. There are several tractography techniques that have been developed in the past to simulate the basic structure of the nerve fiber tracts in the brain, including streamline fiber tracking techniques, diffusion tensor deflection strategy, probabilistic Monte-Carlo method, fast marching tractography based on level set principles, and diffusion simulation-based tractography [5]. Figure 2 demonstrates the extraction of nerve fibers from DTI data and figure 3 illustrates both the front view and side view of a three dimensional representation of fiber bundles which is generated from fiber tractography.

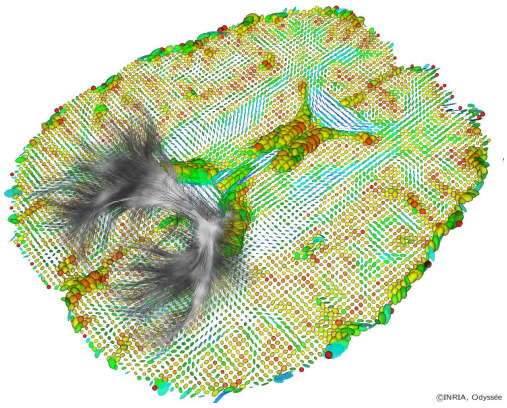

Figure 2: Extraction of fibers in the splenium of the corpus callosum from DTI data [7].

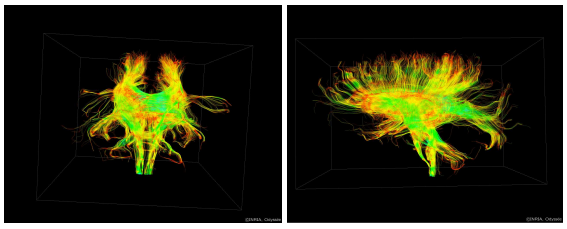

Figure 3: Example of fibers bundles generated from fiber tractography [7].

Both DTI imaging and fiber tractography is important for not only understanding the inter-structure of a brain, but also applying the knowledge to diagnose various neuro-degenerative diseases such as multiple sclerosis.

2.3. Previous work

There are several existing software packages developed for displaying the DTI for medical applications, such as 3DMRI [5, 8], 3D slicer [5, 9] and DTIChecker [5, 10]. However, most of these software applications are designed for generating the DT-MRI and fiber tractography from the existing data. In other words, they are not designed for comparing different nerve fiber tracking algorithms. Furthermore, the user can only manipulate the visualization of the data that is pre-loaded from the database; the software does not provide any functionality for the user to manipulate the data directly or to compare the fiber tractography from different nerve fiber tracking algorithms. Due to the limitations of existing software toolkits, a new toolkit is required for simulating DTI data as well as analyzing and evaluating various nerve fiber tracking algorithms.

3. Project Design

3.1. Specification and Structural Design

According to the specification, the software needs to have the functionality of defining, saving and loading a diffusion tensor field. The diffusion tensor field is defined with user specified parameters such as the size of the diffusion tensor field and distance between each tensor point. Furthermore, the toolkit is also required to have the functionality of defining, saving and loading simulated fiber tracts. Simulated fiber tracts are representations of millions of parallel brain nerve fibers in the diffusion tensor field. After a simulated fiber tract has been defined, the toolkit should also provide the functionality for the user to manipulate the fiber tract easily; such as to change the position or the shape of a fiber tract, and to duplicate and delete a fiber tract. After defining both diffusion tensor field and some fiber tracts, the user can then run various nerve fiber tracking algorithms and do some comparison and analysis between various nerve fiber tracking algorithms.

With a good understanding of the user's requirements, a structural layout of the FiberTK has been designed based on the specification which is shown in figure 4.

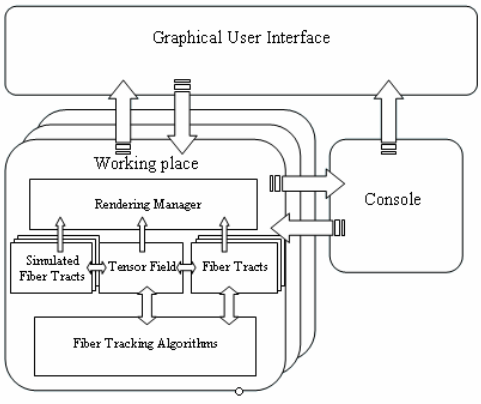

Figure 4: Structural Design of FiberTK.

The basic structure of FiberTK can be divided into two layers. On the top layer, there is the Graphical User interface. It allows the user to interact with individual internal work spaces and displays the graphical outputs on the screen. The bottom layer consists of a number of work spaces. Each work space contains a tensor field and a set of simulated fiber tracts. The tensor field is a set of simulated DTI data which is generated from user defined fiber tracts. The User can then run various nerve fiber tracking algorithms based on the tensor field and generates another set of fiber tracts for comparing the algorithms. All these data will be sent to the rendering manager and displayed through the graphical user interface.

In addition, in order to increase the usability and to display the current state of the software, a console is added to keep a record of user's actions and the current progress information. Error messages will also be displayed through the console window.

3.2. User Interface Design

The Layout of the Graphical User Interface is requested as the standard window user interface which consists of elements such as menus, toolbars, tool menus, and work spaces.

3.2.1. Work space

Due to the fact that the toolkit is implemented in a 3D environment with three dimensional co-ordinate systems, and that the screen of a computer is only two dimensional, a way of defining the third dimensional position needs to be considered. After investigating and examining some existing 3D CAD programs such as PTC ProDESKTOP 2000i2 [11] and Discreet 3D Studio Max4 [12], it has been concluded that two possible solutions can be implemented. Either use a single viewport, which has only one display window, or use multiple viewports, consisting of several but usually four display windows. With the single viewport solution, the user needs to define a working plane in the 3D environment first, and then construct the 3D object based on the working plane. On the other hand, the graphical user interface can be separated into four quadrants, with one quadrant displaying a 3D model in perspective mode and the other three displaying orthographic projected views, including the top, the front and the side views respectively.

Considering both advantages and limitations for each solution and user's expectations of the software, it has been decided to use a single viewport for the graphical user interface by default at the beginning of the project, and the software also has the functionality of changing the single viewport to multiple viewports. However, after gaining some experience in programming, it is discovered that the multiple viewport system is much easier to implement and the user will have better control over the 3D model as each orthographic projected view only consists of two dimensional co-ordinates. In addition, it is more efficient to construct a 3D model as the user can specify the co-ordinates directly into three different viewports. Therefore, the work space is implemented with multiple viewports that allow the user to define and manipulate fiber tracts in all three orthographic projected views but not the perspective view.

3.2.2. Console Window

The console window is embedded in the graphical user interface and it is used as a message window. It gives extra information on the current progress of the software with some additional tips to help the user to achieve some specific tasks. All the user's actions will be recorded and the user can refer back to them later on. More importantly, it displays error messages in red on any illegal operations or invalid inputs.

3.3. Curve Algorithms

The shape of a fiber tract can be defined by a curve as its basic structure. There are many curve algorithms for interpolating a curve, including Hermite Curves (refer to appendix for formula), Bezier Curves (refer to appendix for formula), Uniform B-Spline Curves, Catmull-Rom Spline Curve, Non-uniform B-splines, Non-uniform Rational B-spline [NURBS] and so on [13]. Considering the complexity and property of each algorithm as shown in the table 1, it is decided to implement FiberTK with both Uniform B-Spline Curves and Catmull-Rom Spline Curve.

	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: The table of individual properties among various curve algorithms.

3.3.1. B-spline Curve Algorithm

The Uniform B-Spline Curve is best for use in interactive curve modeling [14]. It defines a curve in such a way that neither of the control points will lie on the curve. However, it is easy to implement and have C2 continuity [13], which means that not only the end point of each curve segment must match, but the tangent, "speed" and "acceleration" around the curve with respect to t must also match. Furthermore, it is more flexible and has local control, which means changing one of the control points will not affect the entire curve. Figure 5 illustrate an example of a cubic B-spline curve which is implemented in the FiberTK.

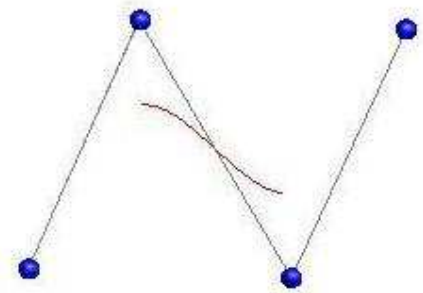

Figure 5: An example of a cubic B-Spline curve which is generated by four control points.

Another very important feature is that each curve segment of the B-Spline curve lies within the convex hull of its associated control points. The convex hull can be defined as a set of points which is the smallest set of points, such that any line connected by any pair of the points lie entirely within the set. Figure 6 illustrate this by drawing the convex hull around four control points. This allows the user to predict the position of the curve more easily.

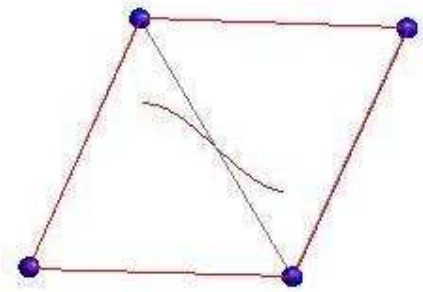

Figure 6: An example of a convex hull around four control points.

Due to the fact that the software is developed for 3D environment, the algorithm implemented in FiberTK is a cubic B-spline curve algorithm with a function degree of three, which means that there are four terms in the polynomial function with a dominant term of t^3 and needs four control points to define each curve segment. Every point on the curve can be calculated using the formula in the equation below:

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

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

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

This formula can also be written in the matrix form, as shown in the equation below:

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

3.3.2. Catmull-Rom Spline Curve Algorithm

On the other hand, the Catmull-Rom spline curve is different compared to the B-spline curve. It does not have the property such that any of the curve segments lies within the convex hull. However, it generates a smooth curve that passes through a set of control points and it is best at interpolating a number of positions [14]. Like the B-Spline, it also has local control and has the continuity of C2 [13]. Figure 7 illustrates an example of a cubic Catnull_Rom spline curve which is implemented in the FiberTK.

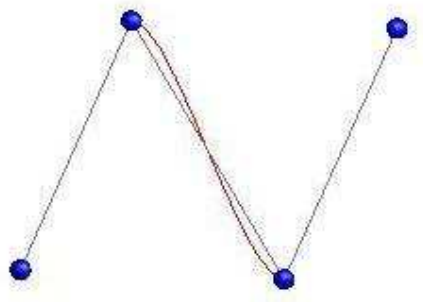

Figure 7: An example of a cubic Catmull-Rom Spline curve which is generated by four control points.

More importantly, the tangent of any point is equal to the previous point minus the next point divided by two, which can be written as the following equation.

$$V_i = (P_{i+1} - P_{i-1})/2 \quad (3)$$

As with the B-Spline, the *Catmull-Rom Spline curve* algorithm implemented in FiberTK is also a cubic function with degree of three, and also needs four control points to define each curve segment. Every point on the curve can be calculated using the formula in the equation below:

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

This formula can also be written in the matrix form as shown in the equation below:

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

3.4. Simulated Fiber Tracts

Once the curve is defined in 3D space, the next step is to generate a three dimensional representation of the fiber tract. In another words, to construct a cylindrical tubed shape to represent a fiber tract in 3D. In order to achieve this, the tangent, principle normal and binormal of the curve must be calculated in order to align each cross-section properly with its neighbors so that the structure does not twist [15] as illustrated in figure 8. The tangent vector, **T** can be calculated by differentiating the first derivatives of both curve algorithms.

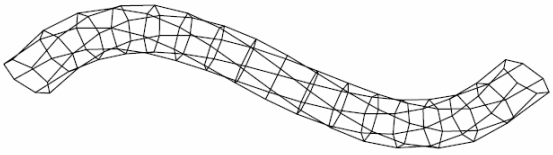

Figure 8: A wire framed view of a three dimensional representation of a fiber tract [15].

From equation (1):

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

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

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

And from equation (4):

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

From these two equations, the tangent of any point on the curve can be calculated. After the tangent is calculated, the principle normal vector, **V** and the binormal vector, **U** can be calculated by using the following equations.

$$U = T \times UP \quad (8)$$

$$V = T \times U \quad (9)$$

Where UP is the up vector [0, 1, 0].

After the calculation of the tangent, principle normal and binormal of a point on the curve, these three vectors can then be used to align the cross-section of the fiber tract. This is achieved by firstly drawing a unit circle with the radius of one at the origin. Next, scale it to a specific radius, and then rotate it to align the tangents of the fiber tract by using the three vectors **T**, **U** and **V**. Finally, translate the circle to its proper position in 3D space as shown in figure 9.

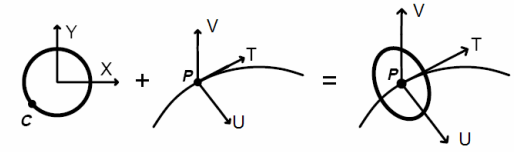

Figure 9: Positioning and orienting a cross-section [15].

However, there is a problem with this approach. Due the fact that when the angle between the tangent vector and the up vector is too small or close to 180 degree, the binormal vector calculated using the equation (8) will not be orthogonal to the tangent vector. Therefore the shape of the fiber tract will be deformed. However, this problem can be solved by using the previous principle normal vector to substitute the up vector. The new equation can be written as:

For P_0:

$$U_i = T_i \times UP \quad (10)$$

For P_i where $i \neq 0$:

$$U_i = T_i \times V_{i-1} \quad (11)$$

$$V_i = T_i \times U_i \quad (12)$$

Where UP is the up vector [0, 1, 0].

3.5. Tensor Field

The Tensor Field is a set of simulated DTI data. Once the user has defined some fiber tracts, the tensor field can then be generated from the tangents of the fiber tracts. Any tensor points that lie within the simulated fiber tracts will be deformed with a high diffusion weight along the tangents of the fiber tracts as shown in figure 10.

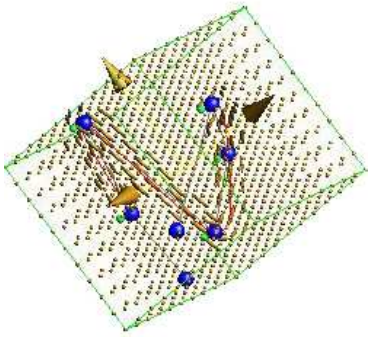

Figure 10: An example of a tensor field in 3D space.

This is achieved by firstly checking whether the tensor point lies in the simulated fiber tract or not. Then the tangent, principle normal and binormal is calculated as the three eigenvectors. The pseudo code in figure 11 briefly explains how this is done.

```
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 closest 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 the tensor field.

If the tensor point lies in the fiber tract, three eigenvectors are calculated as the tangent, principle normal and binormal of the corresponding point on the curve for orienting the tensor point. The diffusion tensor **D** is then calculated by the following formula.

$$D = S^{-1} \Lambda S \quad (13)$$

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

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

$where \lambda_1, \lambda_2, \lambda_3$ are the eigenvectors and

e_1, e_2, e_3 are the eigenvalures.

Due to the fact that the diffusion tensor matrix is symmetrical, only six values are stored in the data structure for each diffusion tensor matrix. Figure 12 demonstrates how the diffusion tensor matrix is stored.

$$D = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \Rightarrow T[a, d, f, b, e, c]$$

Figure 12: Every diffusion tensor matrix is stored in a single dimensional array with a length of six.

Furthermore, when two fiber tracts intersect with each other, their tensor eigenvalues and eigenvectors are added together which generates a disc shape as illustrated in figure 13. As a result, various nerve fiber tracking algorithms will perform differently depending on their individual properties.

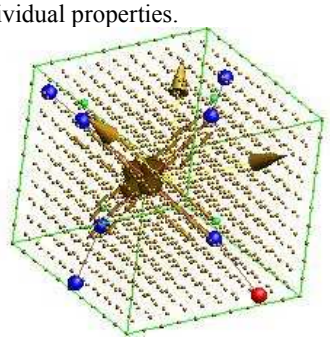

Figure 13: An example of two fiber tracts intersecting each other.

After the tensor field is generated, various nerve fiber tracking algorithms can then be run for further analysis and evaluation. Three nerve fiber tracking algorithms were enclosed in this project, including stream line, tensor deflection and tensor line [16]. All these nerve fiber tracking algorithms were provided by Ms Jing Li from the graphic research group of The University of Auckland.

3.6. Implementation Language

FiberTK was implemented using various programming languages which support multi-platform. These include C/C++, OpenGL, Tcl/Tk, SWIG and TkOGL.

3.6.1. C/C++

C is a general-purpose language originally developed for the UNIX operating system. It is both a high-level and low-level language with better performance and better control over low-level mechanisms such as direct access to hardware.

3.6.2. OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphical applications [17, 18]. The advantage of using OpenGL is that it is a multi-platform, high performance 3D graphics API [17].

3.6.3. Tcl/Tk and TkOGL

Tcl is the abbreviation for Tool Command Language and Tk is the graphical user interface toolkit of Tcl [19]. Tcl/Tk is used because it has a simple and programmable syntax and can be used either as a standalone application or embedded in application programs [20]. Furthermore, Tcl is open source and it can create powerful GUIs incredibly quickly.

TkOGL is a package extension of Tcl/Tk that enables a user to utilize OpenGL. It wraps the GL and GLU commands through SWIG, and it compiles the wrapper code and generates the Tcl extension packages.

3.6.4. SWIG

SWIG stands for Simplified Wrapper and Interface Generator, a software development tool that integrates C/C++ with a variety of high-level programming languages and various common scripting languages such as Perl, Python, Tcl/Tk and Ruby [21].

3.6.5. Software Architecture

The fundamental software architecture of FiberTK is illustrated in figure 14, as all the underlying principles including various curve algorithms, nerve fiber tracking algorithms and OpenGL functions are implemented in C/C++. It is then compiled into Dynamic Link Library (.dll) file in Windows or Shared Object (.so) file in Linux using SWIG which wraps all the functions in C/C++ and makes it callable by the graphical user interface. The graphical user interface is constructed using Tcl/Tk and TKOGL. At runtime, the Tcl/Tk code would load the library file and the output is displayed through the graphical user interface.

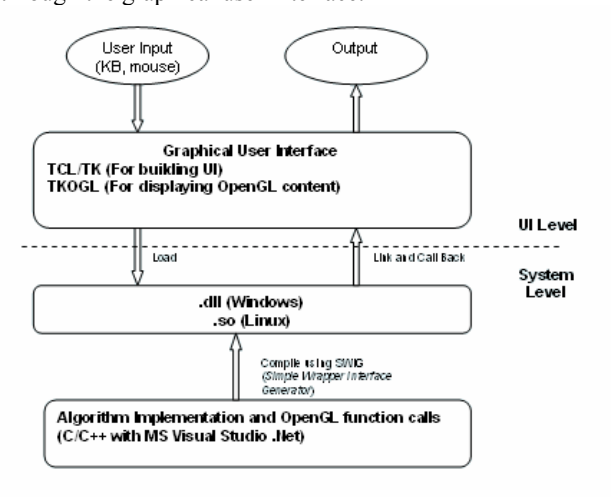

Figure 14: Software Architecture of FiberTk.

4. Project Implementation

4.1. Graphic User Interface

Figure 15 is a screenshot of the graphical user interface. It consists of five parts: the menu bar, tool bar, toolbox, work spaces and console window.

4.1.1. Menu

The menu bar contains a list of drop-down menus including file, edit, view, tool, window and help. Each menu also contains a list of sub-menus and functionalities for the user to manipulate fiber tracts and evaluating nerve fiber tracking algorithms.

4.1.2. Toolbox

The toolbox contains a list of buttons for various functions and it is placed on the left side of the user interface. Buttons in the toolbox can be divided into several categories. The first category is the selecting tool buttons such as select fiber, select control point and select radius control point. The next category is the modifying tool buttons. This includes delete fiber, insert control point and delete control point. The third category is the drawing tool buttons which only consist of two buttons, B-Spline curve algorithm and Catmull-Rom Spline curve algorithm. All these buttons are for defining and manipulating fiber tracts.

The last category is the nerve fiber tracking algorithms tool buttons. This consists of tensor field, stream line, tensor deflection and tensor line. All these buttons are used for running various nerve fiber tracking algorithms.

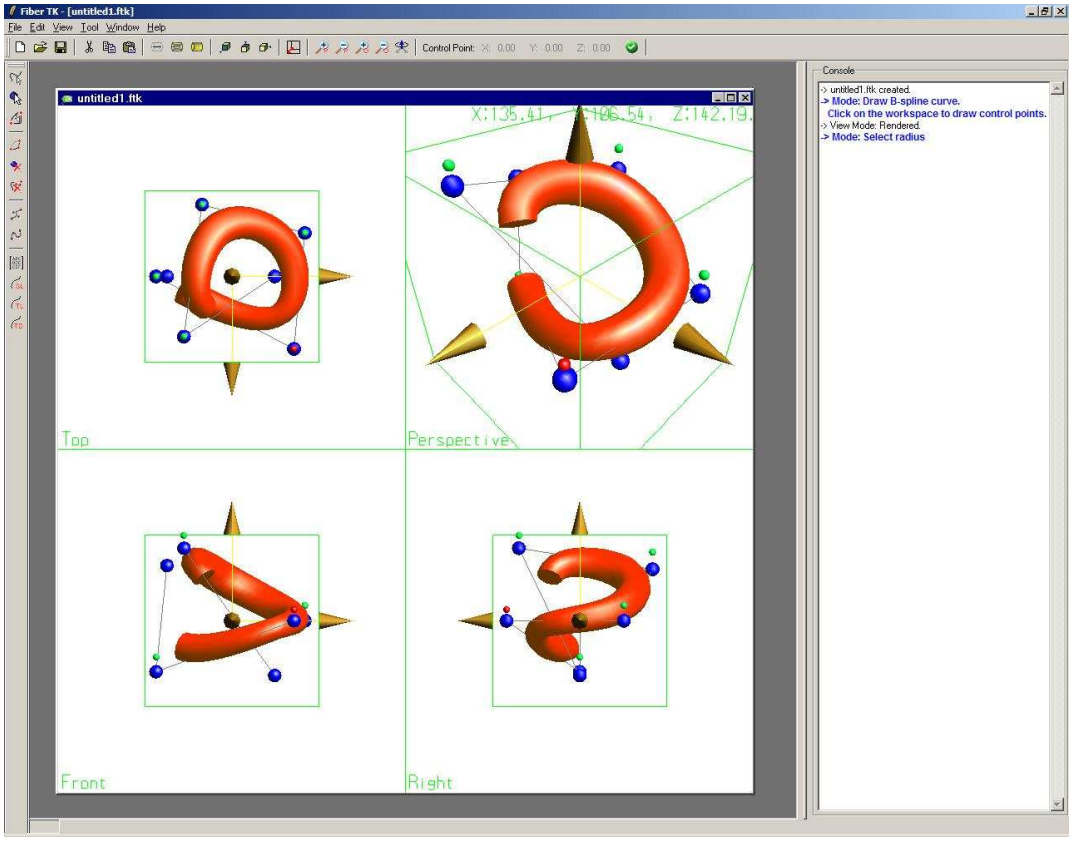

Figure 15: The Graphical User Interface of FiberTK

4.1.3. Toolbar

The toolbar consists of some frequently used buttons that provides some of the basic functions such as new, load, save and so on. A separator is used to group icons with similar functions such as cut, copy and paste. Apart form these buttons, it also contains a large amount of buttons for manipulating views such as changing the viewing mode of the fiber tract from skeleton view to wire framed view, or to full rendered view; setting the viewport back to original position with alignment of a specific axis; rotating the 3D model in the viewport; changing from single viewport to multiple viewports; and zooming in and out of various viewports including both perspective and orthographical views.

4.2. Drawing a Fiber

The user can define a fiber tract by selecting one of the drawing fiber tract tool and start defining the position of the control point in any of the three orthographical viewports. The perspective viewport is only for viewing. The user can also swap to single perspective viewport for a close inspection of the 3D model.

4.3. Manipulating a Fiber

After a fiber tract is defined, the user can then manipulate it using various tools. The user can change the position of a specific control point by first selecting the control point using the select control point tool. Once a control point is selected, it will be rendered in red to indicate that it has been selected. The user can then drag the control point to a new position. The user can also select and drag the entire fiber tract to move it around in the work space by using the select fiber tool. The color of the fiber will also change to purple to indicate that it has been selected. The user can also modify the radius of the fiber tract by selecting the select radius control point tool. This can be achieved by selecting and dragging the radius control point which is sitting on top of each control point, except the first one and the last one. The radius of each segment of the fiber tract is defined by the distance between the radius control point and the control point. By modifying the position of the radius control point, the radius of the fiber tract changes accordingly.

Furthermore, the user can also delete and insert a control point by using the delete control point tool and

insert control point tool respectively. When the user tries to insert a control point, depending on the current number on control points on the fiber tract, the position of the newly inserted control point will vary. When there is only one control point, the position of the new control point will be the same as the existing control point. On the other hand, when there are multiple control points, the new control point will be inserted after the selected control point and the position is calculated with the following equation.

$$P_{new} = (P_{selected} + P_{selected+1}) / 2 \quad (14)$$

However, there is a special case when the last control point is selected, then the equation becomes:

$$P_{new} = (P_{selected} + P_{selected-1}) / 2 \quad (15)$$

Lastly, the user can also delete the entire fiber by using the delete fiber tool.

4.4. Generating Tensor Field

Once the user has defined some fiber tracts, a tensor field can then be generated by using the create tensor field tool. This may take a few seconds for the algorithms to compute and the progress information will be displayed through the console window. The difference between this button and other buttons on the toolbar is that it acts like a switch. The second time the user clicks this button, the tensor field will be removed from the work space. The user can turn the button on and off accordingly.

4.5. Running Tracking Algorithms

After the tensor field is generated, the user can then run various nerve fiber tracking algorithms. There are three buttons on the toolbar for running the algorithms including stream line, tensor deflection and tensor line. All these buttons work the same way as the create tensor field tool button, as the user can turn them on and off depending on what is required.

5. Conclusions

In conclusion, a toolkit FiberTk is built which is capable of generating and visualizing user-specified nerve fiber tracts and evaluating various nerve fiber tracking algorithms. It can be used in biomedical research and analysis to improve diagnostics of neuro-degenerative diseases such as Multiple Sclerosis and Schizophrenia

6. Future work

The future enhancement of this project is further implementation on variable radius. Current interpolation of the radius is linear interpolation, which needs further enhancement. It is suggested that the Catmull-Rom Spline should be used in order to increase the curvature around the surface of the fiber tract.

Apparently, there is one shape for representing the nerve fiber tract and it is very limited. However, more complicated fiber shapes exists in real world applications such as generating a three dimensional representation of the internal structure of the brain. It is encouraged to increase the capability of creating different shaped fiber tracts.

Another important future enhancement can be to automatically generate an analysis report for evaluating various nerve fiber tracking algorithms. This can include the calculation of the mean standard error and percentage of deflected fibers.

Furthermore, there is a limitation on the current software in which the starting point for the tracking algorithms is randomly selected. A further improvement would be to allow the user to define the starting point for tracking algorithms

Acknowledgements

The student would like to thank the supervisor, Dr Burkhard Wuensche for giving us the opportunity to practice and develop our programming and design skills, as well as all the help and advice that he has given us along the way.

Second Examiner, Mr Chris Smaill for giving us much useful advice on how to write both the interim report and the final report.

Ms Jing Li for providing us with various nerve fiber tracking algorithms and giving us various useful advices on the project.

Project partner, Langping Wei for always encouraging and supporting me on this project and I really appreciate his devotion to his part of the project. And other software engineering students including David Huai and Yin Zili for giving us useful advice on this project and helping us perform the usability test on the prototype of our 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] 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.

[3] "Diffusion Tensor Imaging". Retrieved September 3,

2005, from http://www.sci.utah.edu/research/diff-tensor-imaging.html
[4] Burak, A. and Roland, B. (2003). "MR-DTI FIBER TRACTOGRAPHY and Beyond". Retrieved September 3, 2005 from http://www.vavlab.ee.boun.edu.tr/ProjectsNEW/links_DTI.html
[5] 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
[6] Beaulieu, C. (2002). The basis of anisotropic water diffusion in the nervous system – a technical review. NMR Biomed. 15:435-455.
[7] Center for Magnetic Resonance Research, University of Minnesota, Minneapolis, MN USA and at the CEA, SHFJ Orsay FRANCE.
[8] "DIT Software". Retrieved September 5, 2005, from http://cmrm.med.jhmi.edu/DTIuser/DTIuser.asp
[9] "3D Slicer: Medical Visualization and Processing Environment for Research". Retrieved May 4, 2005, from http://www.slicer.org
[10] "Fiber Tracking". Retrieved September 5, 2005, from http://www.ia.unc.edu/dev/download/fibertracking/index.htm
[11] "Autodesk 3ds Max". Retrieved September 5, 2005, from http://www4.discreet.com/3dsmax/
[12] "Orthographic projection". Retrieved May 5, 2005, from http://www.ider.herts.ac.uk/school/courseware/graphics/engineering_drawing/orthographic_projection.html
[13] Burkhard, C. W. and Richard, L. (2004). Curves and Surfaces. Course notes for paper 715.415. University of Auckland.
[14] Hill, F.S. (2001). *Computer Graphics Using OpenGL.* Second edition. Prentice Hall, United States of America.
[15] Bloomenthal, J. Calculation of Reference Frame along a Space Curve. Graphics Gems, Vol1.
[16] Li, J. (2005). An Analysis of Algorithms for In Vivo Fiber Tractography Using DW-MRI Data. The Computer Science Department. The University of Auckland.
[17] "The Industry's Foundation for High Performance Graphics'. Retrieved September 10, 2005, from http://www.opengl.org/
[18] "OpenGL Tutorial". Retrieved September 10, 2005, from http://www.eecs.tulane.edu/www/Terry/OpenGL/Introduction.html
[19] "Tcl/Tk.free.fr". Retrieved September 11, 2005, from http://tcltk.free.fr/man/TkCmd/text.php3
[20] "Tcl Developer Exchange". Retrieved September 11, 2005, from http://www.tcl.tk/software/tcltk/
[21] "SWIG" Retrieved September 11, 2005, from http://www.swig.org/

Appendix

Cubic Bezier Curve in Matrix form [14]:

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

Cubic Hermite Curve in Matrix form [14]:

$$p(t) = \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{pmatrix} \begin{pmatrix} P_1 \\ P_4 \\ V_1 \\ V_4 \end{pmatrix}$$

Cubic B-Spline Curve in Matrix form [14]:

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

Cubic Catmull-Rom Spline Curve in Matrix form [14]:

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