

Fiber Tractography Using Brain DT-MRI Data and VTK Visualization (June 2003)

Murat Aksoy, Sancar Adalı and Hasan Ayaz

Abstract—Diffusion Tensor-MRI is a promising modality of MRI that takes advantage of Brownian motion of water molecules in order to analyze fiber tract anatomy of the human brain. Displaying connectivity data requires tracking nerve fibers and various methods has been proposed. We have implemented a fiber tracking environment using Visualization Toolkit(VTK) for rendering and processing 2D and 3D images in the environment. The environment allows the user to use three different methods for fiber tractography: Fast Marching, Principal Diffusion Direction Tracking methods using Euler and Fourth order Runge-Kutta methods.

Index Terms—DT-MRI, Fiber Tractography, Fast Marching, VTK, Visualization, Euler Method, Fourth Order Runge-Kutta Method

I. INTRODUCTION

MRI Imaging has been a field of remarkable innovation in the last decade. Diffusion Tensor MRI is a recent extension of MRI which measures motion of hydrogen molecules in water to determine diffusion of water in the tissues. From this diffusion information existence of fiber structures - such as white brain matter or skeletal muscle fibers -can be inferred that has tissue components blocking random diffusion of water molecules known as Brownian motion.

Various methods exist for fiber tractography, which is the process of connecting those pixels according to DT-MRI data. The DT-MRI data can be stored as diffusion tensors and either these tensors or mathematical structures calculated from these tensors such as eigenvectors, can be useful in determining connectivity data. Important examples of fiber tractography are summarized in the next part.

Manuscript received June 15, 2003.

Murat Aksoy is with Dept. of Electrical and Electronics Engineering Bogazici University, Bebek Istanbul TURKEY (e-mail: muraksoy@hotmail.com).

Sancar Adalı is with Dept. of Electrical and Electronics Engineering Bogazici University, Bebek Istanbul TURKEY (e-mail: adali@boun.edu.tr).

Hasan Ayaz is with the Intelligent Systems Laboratory, Dept. of Electrical and Electronics Engineering Bogazici University, Bebek Istanbul TURKEY, (e-mail: hayaz@yahoo.com).

II. RELATED WORK

A. Literature Survey on Fiber Tractography

An obvious method is to follow the direction of maximum diffusion. A good estimate of this is the eigenvector corresponding to the maximum eigenvalue of the diffusion tensor. This is referred to in literature as Principal Diffusion Direction Tracking. Such techniques use Fourth-Order Runge Kutta Methods or Euler Methods to calculate the points from the eigenvector. A more brute-force method would be to solve diffusion equation. Fast- marching method can also be applied to solve the problem of fiber tracking. This method provides a fast way to propagate interfaces that never go back. It can be used to find the shortest path between two points depending on the tensor in the neighborhood of the propagation interface. Parker has implemented this in [Parker]. There exist also a method using Markov random field regularization. This method. A novel method is presented in [Björnemo&Brun,2002] incorporating uncertainty in tractography mechanism. After segmenting white matter using EM algorithm and they modify PPD tracking paradigm by regularizing the diffusion tensor by adding a component from the previous tracking direction and adding a stochastic part to the eigenvector corresponding to maximum eigenvalue. The stochastic part is weighted by a variable determined from the Sequential Importance Sampling Resampling framework which models the propagation as the random self-avoiding walk. For example, the uncertainty of propagation increases if the propagation front encounters a planar, or spherical tensor. In such a case, the weight belonging to the stochastic term start to decrease and until a more certain propagation direction could be “sampled”. The EM algorithm has also been applied to this problem in [Inati et al.] They declare that their method is insensitive to accumulation of error unlike stepwise methods such as PPD methods, and therefore the method results in longer tracts. Also multiple pathways can be determined with a measure of the likelihood of each being the right pathway.

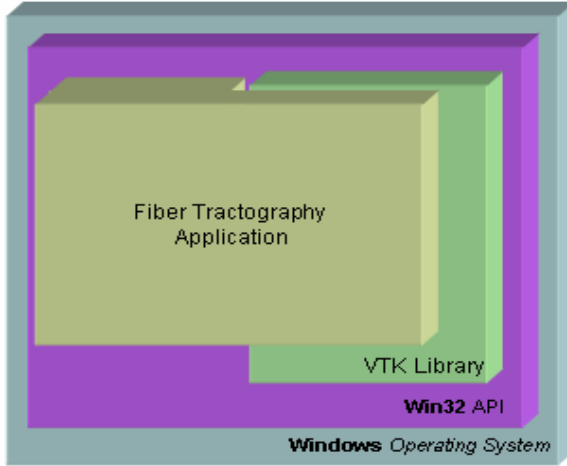
III. VISUALIZATION TOOLS AND PROGRAMMING ENVIRONMENT

Working environment is selected to be Windows Operating System. There are two main methods to create a C++ executable in Microsoft Visual Studio. First method uses Win32 SDK and the second uses MFC (Microsoft Foundation

Classes). Both has pros and cons and both are the most efficient and flexible methods for creating Windows executables. Among these two methods, Win32 SDK selected because of gives direct access Win32 API (Application Programming Interface). Win 32 API is the underlying core component of Windows Operating System over the Hardware Abstraction Layer. MFC, also access it through Win 32 SDK and appends extra code for automatic handling of the hierarchical item and garbage collection. Thus using Win32 SDK, we can have a more efficient executables in terms of both performance and executable size.

VTK 4.0 (Visualization Toolkit) Library is being used for rendering and processing 2D and 3D images for the program. VTK is used as a set of C++ class library in the project. The library visualization system that supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques like implicit modelling and mesh smoothing.

Looking at the basic structure of the vtk library, it is understood that it is an higher visualization system than OpenGL which is a low level computer graphics rendering system. The architecture of our system is summarized in figure 2. As seen in the figure (), our application has both accesses to Win32 API and Vtk library. A more detailed discussion of the usage of VTK library will follow in Section V.



IV. METHODS OF FIBER TRACTOGRAPHY

1) Euler and Higher-Order Runge Kutta Methods

Stepwise tractography methods model fiber tracts as a 3D discrete curve to be drawn step by step. In each step the following formula is used to go the next sample of the curve:

$$r_{n+1} = r_n + h * V_n$$

where r_n is the vector corresponding to the voxel at nth step when starting from r_0 , h is the step size parameter and V_n is the vector in the direction the tract will follow [3]. The

objective here is to find V_n for any n given initial conditions r_0 and V_0 . The most basic method to do that is Euler's Method.

Euler Method solves the problem of ordinary differential equations by assuming a constant step size and evaluating the function $f(x,y)$ at y_n in the differential equation $dy/dx=f(x)$ in order to find the finite difference and add to y_n . Of course we will be using generalization of this method to 3D and our finite difference V_n will be the eigenvector corresponding to the maximum eigenvalue at the nth step.

Runge-Kutta Methods are generalizations of Euler's method for finding smoother and more reliable approximations of the solution of differential equations. We have implemented Fourth-order Runge-Kutta Method that could be formulated as:

$$V_n = \frac{1}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \quad \text{where } k_1, k_2, k_3, k_4 \text{ are found as:}$$

$$\vec{k}_1 = \frac{\vec{V}_0 \cdot \vec{E}(\vec{r}_0)}{|\vec{V}_0 \cdot \vec{E}(\vec{r}_0)|} E(\vec{r}_0)$$

$$\vec{k}_2 = \frac{\vec{V}_0 \cdot \vec{E}(\vec{r}_0 + \frac{h}{2} * \vec{k}_1)}{|\vec{V}_0 \cdot \vec{E}(\vec{r}_0 + \frac{h}{2} * \vec{k}_1)|} E(\vec{r}_0 + \frac{h}{2} * \vec{k}_1)$$

$$\vec{k}_3 = \frac{\vec{V}_0 \cdot \vec{E}(\vec{r}_0 + \frac{h}{2} * \vec{k}_2)}{|\vec{V}_0 \cdot \vec{E}(\vec{r}_0 + \frac{h}{2} * \vec{k}_2)|} E(\vec{r}_0 + \frac{h}{2} * \vec{k}_2)$$

$$\vec{k}_4 = \frac{\vec{V}_0 \cdot \vec{E}(\vec{r}_0 + h * \vec{k}_3)}{|\vec{V}_0 \cdot \vec{E}(\vec{r}_0 + h * \vec{k}_3)|} E(\vec{r}_0 + h * \vec{k}_3)$$

where $\vec{E}(\vec{r}_0)$ is the continuous 3D function interpolated from DT-MRI maximum eigenvector data. This interpolation will be a trilinear interpolation using eight sample points around endpoint of \vec{r}_0 .

a) Considerations in Euler and Fourth-Order Runge Kutta Methods

One of the main considerations in Euler and Higher Order Methods is consistency in choosing the sign and direction of V_n . In order to maintain consistency, the dot product of the V_{n-1} and V_n divided by the dot products' absolute value is multiplied by the result of $\vec{E}(\vec{r})$ function. When V_{n-1} and V_n have same signs, the signs stay the same, when they have opposite signs, the signs reverse, so that the direction of the path of propagation doesn't reverse in consequent steps and the method is saved from going erratically back and forth. An important problem in such methods is that tracking is sensitive to noise and in images with low SNR, the trajectory

might veer off. To ameliorate this problem, intrinsic parameters such as curvature of the 3D tract curve are calculated at each step. If the curvature of the tract is above a threshold (one that could extract from physical attributes of the fiber tracts in the brain) then the propagation will stop. Other conditions for the stopping of the propagation includes low anisotropy measures in our case RA as defined in[]

2) Fast- Marching Method

Fast marching method was implemented as another tractography method. In the fast marching algorithm, the main purpose is to track the motion of an interface as it evolves. If we denote $\gamma(t)$ as the family of curves as the initial curve $\gamma(0)$ evolves in the direction of the normal vector with speed F , then this evolution of the curve can be formulated as :

$$\vec{x}_t = F(\kappa) \cdot \vec{n}$$

Where κ is the curvature of the curve at that point and n is the unit normal to the curve. This function simply states that the change in the curve with respect to time is the most in areas where the curvature-dependent speed function and the unit normal to the curve are most aligned.

However, in such a curve evolution, for $F=1$, almost in all of the curves, the smoothness of the curve is lost, and we require a weak solution, because the solution weakly satisfies the definition of differentiability. (A weak solution is the one that satisfies the integral form of the original equation. Due to the fact that such a solution does not require the solution to be differentiable with respect to the independent variable, the possible range of solutions broadens).

If we think of the curve as an interface separating two regions, the solution we want is the shortest distance or the first arrival. In another sense, the set of points in the front of the curve must always correspond to the points that are located at a distance t away from the original curve. A way to obtain this solution is through enforcing an ‘‘entropy’’ condition for the propagating interface.

If we take the speed function in the form :

$$F = 1 - \varepsilon \kappa$$

For $\varepsilon > 0$, we will have a smooth flow. The limit of this solution as $\varepsilon \rightarrow 0$, which is also known as the viscous limit, will be the entropy solution for the constant speed case. In order to find this solution, techniques used in hyperbolic conservation laws are used. An equation for $u(x,t)$ of the form :

$$u_t + [G(u)]_x = 0$$

is known as the hyperbolic conservation law. The solution to these equations can develop discontinuities, known as ‘‘shocks’’. In order to avoid these sudden compressions or expansions, a diffusive term is added to the right side of this equation. For example, in the case of Burger’s equation :

$$u_t + uu_x = \varepsilon u_{xx}$$

The term εu_{xx} acts as a smoothing term and for $\varepsilon > 0$, the solution remains smooth for all time.

In the solution of the ‘‘hyperbolic conservation law’’, the curves on which the solution $u(x,t)$ always stays the same are called characteristics. In some cases, these characteristics may exhibit converging behaviour, known as ‘shocks’; or they may diverge, leaving a ‘‘gap’’ in the (x,t) plane. The solution to this problem is to choose $u(x,t)$ to be the one obtained as the limit of the solution as the diffusive term added to the right of the equation (the term εu_{xx} in the case of Burger’s equation) vanishes. This solution is similar to the one that we want to obtain for the curve evolution equation. Thus, the solution schemes obtained for the hyperbolic conservation law can also be applied to our problem.

Integration of the hyperbolic equation allows us to get the conservation form of the equation:

$$\frac{d}{dt} \int_a^b u dx = G(u(a,t)) - G(u(b,t))$$

Our aim is to construct weak solutions to this conservation form which also satisfy the entropy condition and don’t contain any discontinuities. A scheme is in conservation form if there exists a numerical flux function $g(u_i, u_i)$ or $g(u_i, u_{i+1})$ which approximates the values for $G_{i-1/2}$ or $G_{i+1/2}$ such that

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{G_{i+1/2} - G_{i-1/2}}{\Delta x}$$

Such a solution confines the shocks to a few grid points. The entropy condition can also be satisfied by choosing a scheme that takes three arguments $u_i^n, u_i^{n-1}, u_i^{n+1}$, and that is a monotonically nondecreasing function of its arguments. Such a conservative, monotone scheme also satisfies the entropy condition. Thus, in order to construct a viable scheme, we need to make sure that it is in conservation form and it is a monotone increasing function of its arguments. One of the approximating functions used in a scheme is :

$$g(u_1, u_2) = (\max(u_1, 0))^2 + (\min(u_2, 0))^2$$

This entropy-satisfying and relatively few diffusing causing scheme in the conservation form allows us to approximate the gradients in both the initial value and boundary value formulations. After finding this gradient, we can come back to the relation between the level set methods and fast marching.

The main challenge in this part of the project is to track a surface that is evolving. Assume that our initial surface is denoted by Γ . Now consider the function $\phi(x,t) = \mp d$, where d is the distance of the point x from the evolving curve at time t . Plus sign indicates that x is outside the surface, and minus sign says it is inside. Thus, it is possible to say that the zero level set of the function $\phi(x,t=0)$ gives us the original curve Γ . By chain rule, an equation for the evolution of the interface can be produced :

$$\phi_t + F|\nabla\phi| = 0$$

$$\phi(x, t = 0) = \text{given}$$

This partial differential equation states that our initial curve changes (that is, evolves) with respect to time, and this evolution is in the direction of the negative gradient of the evolving surface and weighted by the speed function. This provides an outward propagation when $F > 0$.

As described before, a careful approximation of the gradient must be used in order to get a correct weak solution. The application of the result found above to the gradient in our case produces :

$$\phi_{ij}^{n+1} = \phi_{ij}^n - \Delta t (\max(D_{ij}^{-x}\phi, 0)^2 + \min(D_{ij}^{+x}\phi, 0)^2 + \max(D_{ij}^{-y}\phi, 0)^2 + \min(D_{ij}^{+y}\phi, 0)^2)$$

$$\text{where } D_{ij}^{+x}\phi = (\phi_{i+1,j} - \phi_{ij}) / \Delta x$$

However, in the formulation above, when we try to get the solution for $\phi(x, t)$, we find the solution for all x 's, not just for the level set that corresponds to our evolving curve (remember our curve corresponds to level set for $x=0$). This procedure is computationally expensive. Narrow band approaches are applied in order to make up for this cost.

In the narrow band approach, operations are only performed on the neighbouring the zero level set. The main idea in this approach is to name the points as alive, if they are inside the band; "land mines", if they are near its boundary or "far away", if they are outside the band. Work is performed only on the alive points, and the band is reconstructed once land mine points are reached. An extreme one cell version of this method leads to the "fast marching algorithm".

Consider a front moving with the speed function $F=F(x,y,z)$

$$\phi_t = F(x, y, z)|\nabla\phi|$$

$$\phi(x, t = 0) = \Gamma$$

Now consider the two dimensional case where the evolving structure is a curve. If we denote $T(x,y)$ as the time in which the evolving curve crosses the point (x,y) :

$$|\nabla T|F = 1$$

This is a form of the Eikonal equation, and the viscosity solutions are connected with the solution of this equation, where, as has already been stated above, selection of monotone and consistent schemes will lead to the schemes that select correct viscous limit.

In the fast marching equation, for the approximation to the gradient for T , the following method, proposed by Rouy and Tourin, is used (instead of the difference approximation that was proposed above):

$$\left[\begin{array}{l} \max(D_{ijk}^{-x}T, -D_{ijk}^{+x}T, 0)^2 + \\ \max(D_{ijk}^{-y}T, -D_{ijk}^{+y}T, 0)^2 + \\ \max(D_{ijk}^{-z}T, -D_{ijk}^{+z}T, 0)^2 \end{array} \right]^{\frac{1}{2}} = \frac{1}{F_{ijk}}$$

The equation above is a quadratic equation for T_{ijk} , and can be solved by iteration. One can start with initial T_{ijk} values for all grids, and by looking at the neighbouring T_{ijk} values, (namely $T_{i-1,j,k}$ $T_{i+1,j,k}$ $T_{i,j-1,k}$ $T_{i,j+1,k}$ $T_{i,j,k-1}$ $T_{i,j,k+1}$), every T value for each grid is updated, and this process containing all the grid points is repeated n times until convergence. Such an operation requires one loop for the iteration and three loops for the coordinates, thus making the computational complexity as high as $O(N^4)$.

The fast marching algorithm proposes the use of the causality relation in this algorithm. In order to explain this, we should first consider the nonnegativity of the left side of the equation (it can be seen that the outcome of any of the max operations is a number that is equal to or greater than zero). In order to understand this more clearly, consider only one dimensional version of this equation, i.e. :

$$\max(D_i^{-x}T, -D_i^{+x}T, 0) = \frac{1}{F_i}$$

If the curve (or in our case the point) is propagating to the right, the first term in the max operation is taken, and the equation becomes :

$$\frac{T(x) - T(x - \Delta x)}{\Delta x} = \frac{1}{F_i}$$

In this case, $T(x)$ is always greater than $T(x-\Delta x)$ (of course $F_i > 0$). Thus, the value of T always increases as the point propagates to right. In the case of a propagating curve to the left, the term in the middle is always chosen, and the same reasoning applies. The 0 term only has a regularizing role.

Thus, the Fast Marching method relies on propagating the curve starting from the smallest T value. By looking at the narrow band values around our initial curve, we expand the curve by taking the grid that has the smallest T value, and freeze this point. After that, new points are added to the set of narrow band points, and the same operation is repeated over and over. Since the recomputation of T at any of the new narrow band points cannot yield results smaller than the frozen points, we can systematically move forward. This means that we won't have to go back and revisit a frozen point and compare its T value with the new T values.

The fast marching algorithm can be summarized as follows :

- 1) Set the initial points as "known"
- 2) Tag the points that are 1 pixel away as "trial"
- 3) Tag all other points as "far away"

- 4) Determine the trial point with the smallest T value
- 5) Add this point to the known, remove it from trial
- 6) Tag as “trial” that are the neighbours of the new added point and that are not “known”
- 7) Recompute the T values of the new points by solving the quadratic equation
- 8) Return to 4

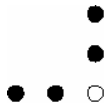
For the fiber tractography application of this algorithm, the speed function F will be :

$$F(r) = A|\varepsilon_1(r) \cdot n(r)|$$

where $\varepsilon_1(r)$ is the eigenvector corresponding to the largest eigenvalue, and $n(r)$ is the unit normal to the curve. The normal to the curve is calculated using the gradient of the arrival times :

$$n(r) = \frac{\nabla T(r)}{|\nabla T(r)|}$$

However, in order to calculate the gradient of T, we also need the arrival time information about the point that we are on. Thus, the right side of the equation also becomes a function of T. In order to understand how this situation is dealt with, examine the figure below :



The white point is a trial point, black points are known, and the other points outside the border are far away. If we denote the arrival times of known nodes as T_x and T_y , the equation becomes :

$$(T - T_x)^2 + (T - T_y)^2 = \frac{1}{F_{ij}}$$

instead of F, we put the inner product of the unit normal and

the vector field, which can be denoted as $\begin{bmatrix} F_x \\ F_y \end{bmatrix}$:

$$\sqrt{(T - T_x)^2 + (T - T_y)^2} = \frac{1}{\frac{(T - T_x)F_x + (T - T_y)F_y}{\sqrt{(T - T_x)^2 + (T - T_y)^2}}}$$

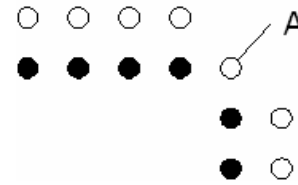
$$T = \frac{1 + T_x F_x + T_y F_y}{F_x + F_y}$$

Thus, the quadratic equation becomes a linear one. The signs of F_x and F_y change according to the direction of the gradient, thus the denominator of the equation above never becomes zero. If it becomes zero, the meaning of this is that the vector at that point is perpendicular to the normal. At this stage, we assume that there's no diffusion along that direction, and do

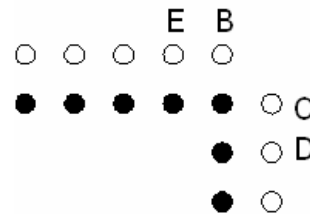
not consider that dimension in the solution of the equation. (T_i and F_i become zero).

Another important point is that, since in the DT-MRI data, there's no difference between a vector to be in one direction and the opposite one, two equations are solved for T : one considering F vector in one direction and the other in the other direction. Obviously, one of these vectors will point to the inner part of the surface and will give a smaller values, and the other one will point outward from our level curve. It is clear that the latter one is correct, and in fact this is the one that gives a greater value than the other one. (Since outward propagation gives always greater T values – the main idea of the upwind scheme).

One last remark which is about the update of the T values of the trial points : Should we use the T values of the other trial points, or is it sufficient if we only use T values belonging to the known points. The answer is the second choice, it suffices if we only consider known points.



In the figure above, suppose A turned out to be the trial point with the smallest T value, and we have updated it to known:



The new trial points, B and C are updated accordingly, but the crucial question is that, should E and D be affected by B and C. The answer is no, and the reasoning behind this is the same as the problem above, and it goes like this : Say, if E is affected by B, that this is to assume that E has a greater T value than B (if T value of B is included in the equation, this will any way give a greater value). But this means that, in the next iteration, we will never choose E, because B has a smaller T value. Then we really don't have to care about the correct value of E, it may be corrected when B becomes an alive point at some time. If the case is the other way around, that is, if E has a smaller T value, than it is never affected by B, and B should not be included in the equation. Thus, at each propagation of the curve, only the points at the 4 neighbourhood of the new known point is updated, the others are kept the same.

The propagation of the curve will start from initial “seed” points. The arrival times for all the other points from these initial points are be calculated by the algorithm defined above, and we are going to have a map filled with arrival times. Among these, the points that are well connected to the seeds

will have smaller arrival times. After choosing another point on the map, the most likely connection between this point and the seeds will be formed by following the steepest descent of T from the chosen point and the seed point. Thus, the most likely connected between a point and the seed point will be determined.

V. VISUALIZATION IMPLEMENTATION

To create a rendering pipeline, vtk classes were employed. Among these classes vtkRenderWindow provides connection between Win32 and vtk by specifying the window area, where graphics drawing takes place. vtkRender class is combined to vtkRenderWindow class instance at run-time. This class has methods that execute rendering or calculation & rastering of pixels.

Another fundamental and useful class is vtkRenderWindowInteractor class. This class provides some kind of interaction between user and the rendering pipeline. It accepts simple commands from the user and transmit these to the rendering classes. These commands are zooming in/out and padding. Also, interactors can take input from keyboard if the focus is on the drawing window.

Our system uses Actors to input data to rendering pipeline. Actors serve to group rendering attributes such as surface properties (e.g. ambient, diffuse, specular color), representation (e.g. , surface or wireframe), texture maps, and geometric definitions.

After defining the actor, we need to define the data structure of vtk library since, the data is need to be handled by these. vtkPoints class represents 3D points. The data model for vtkPoints is an array of vx-vy-vz triplets accessible by (point or cell) id. vtkCellArray class is used to represent cell connectivity. The cell array structure is a raw integer list of the form: (n,id1,id2,...,idn, n,id1,id2,...,idn, ...) where n is the number of points in the cell, and id is a zero-offset index into an associated point list. This class is used in combination with vtkPoints class in forming vtkPolyData class. vtkPolyData is a data object that is a concrete implementation of vtkDataSet. vtkPolyData represents a geometric structure consisting of vertices, lines, polygons, and triangle strips. Point attribute values (e.g., scalars, vectors, etc.) also are represented. In forming the stream tubes, we have used vtkPolyData class for storing the coordinate information.

vtkImageData class is used as the storage of the image pixel values. This class can get input from a ANSIC array of dimension 3. This image, then, can be displayed by transferring this data to vtkImageActor class which is a child class vtkActor. Finally, the data in vtkImageActor is fed into the current rendering class.

For the main dialog of our application, there are three separate image views are placed at the top row as seen in figure 1. These views will display axial, sagittal and coronal 2D images of the brain. Slide-bars, under them will provide navigation through the frames. Each of these frames are a separate instance of the 'pencere' class that forms an individual rendering pipeline separate from each other.

The bottom left view will represent a 3D view of the combination of the previous 3 views. The information obtained from the processing algorithms provide the voxel locations to draw fibers. This view presents the extracted fiber to the user. It can be rotated and translated. However, hardware acceleration is required.

Thus, the rendering system uses hardware acceleration to model the whole system in three dimensions.

To set seeds and observing them in the images, a linked list data set is created which is formed from sets, destination and departure link lists. Each link list contains a base class, called point. This class contains the x,y and z value of the required seed point.. This data is separate from image data. vtkImageReslice class is the swiss knife of the vtk class, that is capable of slicing and rendering the volume data structure. This class is used in manipulating volume data.

ACKNOWLEDGMENT

We would like to thank Burak Acar,Ph.D for his invaluable support and advice during the project

REFERENCES

- [1] O. Ciccarelli,a G.J.M. Parker,b A.T. Toosy,a C.A.M. Wheeler-Kingshott,a G.J. Barker, a P.A. Boulby,c D.H. Miller,a and A.J. Thompsona, "From diffusion tractography to quantitative white matter tract measures: a reproducibility study"
- [2] Peter J Basser, Sinisa Pajevic, Carlo Pierpaoli and Akram Aldroubi, "Fiber Tract Following in the Human Brain Using DT-MRI Data", IEICE Trans . Inf. & Syst. Vol E85-D January 2002
- [3] C.R. Tench, P.S. Morgan, M. Wilson, and L.D. Blumhardt, "White Matter Mapping Using Diffusion Tensor MRI", Magnetic Resonance in Medicine 47:967–972 (2002)
- [4] G.J.M. Parker, "Tracing Fiber Tracts Using Fast Marching"
- [5] S. Zhang , C . Demiralp , M. DaSilva , D. Keefe , D. Laidlaw , B. D. Greenberg , P.J. Basser , C. Pierpaoli , E.A. Chiocca , T. S. Deisboeck , "Toward Application of Virtual Reality to Visualization of DT-MRI Volumes"