A White Matter Stochastic Tractography System

Release 1.00

Tri M. Ngo[1], Carl-Fredrik Westin[2]

December 14, 2007

[1]MIT, Cambridge
[2]Harvard Medical School, Cambridge

Abstract

White matter tractography enables studies of fiber bundle characteristics. Stochastic tractography facilitates these investigations by providing a measure of confidence regarding the inferred fiber bundles. This article presents a multithreaded ITK stochastic tractography filter that will enable novel studies of fiber tract abnormalities. Additionally, we provide an easy to use command line interface for the filter that is compatible with the 3D Slicer visualization environment.

Contents

1 Introduction

DTI data sets provide information about the diffusion of water at each voxel, or volume element, in the form of diffusion tensors. This information can be used to extract the underlying fiber tracks which produced the

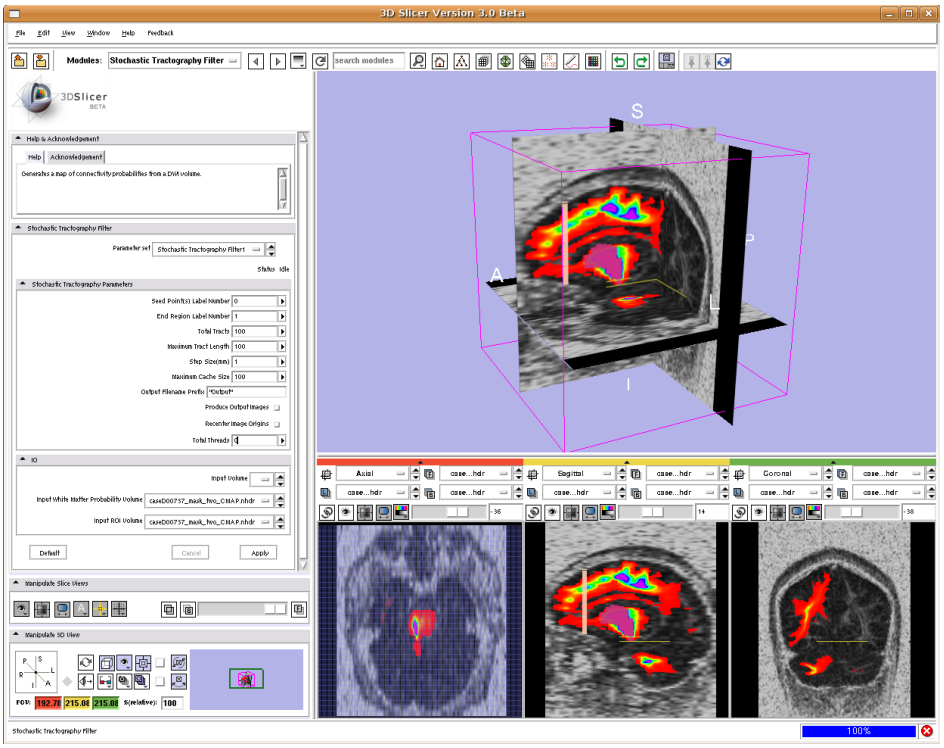

Figure 1: 3D Slicer environment displaying the Stochastic Tractography Module interface and a connectivity map overlaid on a fractional anisotropy image.

observations. This technique is known as DTI Tractography.

One possible method of performing tractography is to generate tracts which follow the direction of maximal water diffusion of the voxels they pass through [9, 1]. This method is sometimes called streamline tractography. Unfortunately streamline tractography does not provide information about the uncertainty of the generated tracks caused by noise or insufficient spatial resolution. Bayesian white matter tractography methods try to address this problem by performing tractography under a probabilistic framework. Several formulations of stochastic/probabilistic tractography have been suggested [3, 2, 10, 7, 8], however tools which enable widespread adoption of stochastic tractography in clinical studies are not currently available.

This paper presents an easy to use, multithreaded ITK filter for performing stochastic white matter tractography based on the algorithm described by Friman et al. [5, 6]. Additionally, we present a 3D Slicer [4] graphical user interface module which uses the stochastic tractography filter, further increasing its ease of use and further encouraging its application in clinical research (figure 1).

2 Algorithm Overview

The stochastic tractography algorithm implemented in this paper is based on Friman's [6] approach with some modifications to the stopping criteria. This paper provides a brief overview of the algorithm. Please refer to Friman's [6] original paper for a more complete explanation of the theory. Figure 2 provides a flow chart demonstrating key steps in the algorithm.

A fiber tract is modeled as a sequence of unit vectors. The orientation of these unit vectors is determined

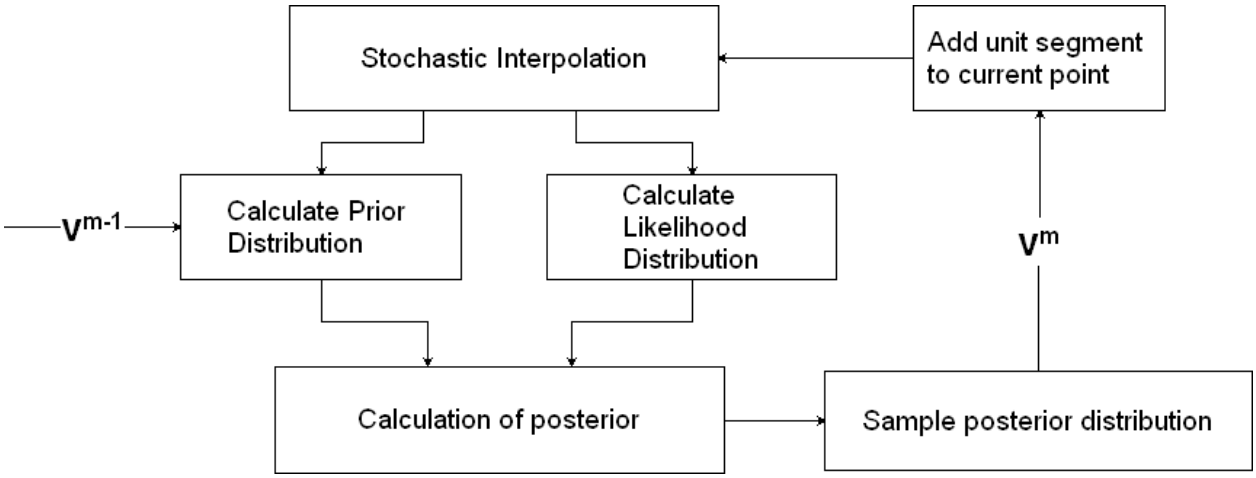

Figure 2: A flow chart demonstrating key steps in the stochastic tractography algorithm

by sampling a posterior fiber orientation distribution which is dependent on the local diffusion data as well as the orientation of the unit vector in the previous step. The posterior distribution is a normalized product of the prior likelihood of the fiber orientation and the likelihood of that fiber orientation given the local diffusion data.

Friman uses a subset of the tensor model which is called a constrained diffusion model. In this model, the two smallest eigenvectors of diffusion tensor are equal, constraining the shape of the diffusion tensor to be linearly anisotropic. The constrained model rules out the possibility of nonlinear, or non-cylindrical anisotropic diffusion distributions. Deviations from linearly anisotropic diffusion distributions are captured as uncertainty in the fiber orientation. The constrained model is combined with a Gaussian DWI noise model to obtain a fiber orientation likelihood function. The parameters for the constrained model are derived from a weighted least squares estimation of the parameters for the log tensor model.

The orientation of each vector depends only on the previous vector. This dependency is formulated in the prior on the fiber orientation. Prior knowledge about the regularity of the fiber tract can encoded in this prior probability. The prior also serves to prevent the fiber from backtracking, since the likelihood distribution alone is axially symmetric.

Friman's approach is a Bayesian inference algorithm similar to Behrens's but with some important optimizations [6]. In contrast with Behrens's two-compartment observation model, the constrained model used by Friman is derived from the thoroughly studied tensor model of diffusion. The advantage of using the constrained model is that it is relatively easy to estimate the parameters for the model. The parameters for the constrained model are obtained after the tensor model has been fit to the diffusion data. Since the parameters for the tensor model are easily obtained through many computationally efficient ways, the constrained model's parameters are likewise easy to obtain. The constrained model can be fit to every voxel within a matter of seconds whereas Behrens's model takes a couple of hours [6]. Additionally Friman avoids using MCMC techniques by assuming that parameters other than the principle diffusion direction take on their ML estimates with certainty within each voxel. Friman demonstrates that eliminating this source of uncertainty has little effect on the resulting posterior fiber orientation distribution.

In Friman's paper, tracking is terminated when an encountered voxel's diffusion distribution anisotropy is below some threshold. However, since the stochastic tractography algorithm takes into account this uncertainty with an increase in the spatial variance of sampled fibers, this termination criterion seems arbitrary and contradictory with the goals of stochastic tractography, which is to enable sampling of white matter fiber

tracts in regions of uncertainty. Thus we replace this termination criterion with one which terminates tractography based on the posterior probability that a fiber tract exists within the current voxel. This probability is equivalent to the probability that the current voxel is in white matter. A map of white matter posterior probability can be obtained through a soft segmentation of an anatomical image of the brain co-registered with the DWI data. Alternatively, the soft segmentation can also be performed on the B0 image of the DWI data set, thus eliminating the need for additional data. Although this alternative termination criteria may seem equivalent to using an anisotropy threshold criterion, because white matter generally has higher anisotropy than gray matter, the alternative method does not exclude regions of white matter which have low anisotropy due to crossing fibers. This criteria should enable the algorithm to detect more tracts in white matter than under the anisotropy termination criteria.

3 Implementation Details

The stochastic tractography algorithm is a Monte Carlo algorithm which samples the high dimensional parameter space of fiber tracts. This is a large space because fiber tracts are characterized by a sequence of segment orientations, each of which is a separate parameter describing the fiber tract. As such, it may take many samples to accurately approximate the posterior distribution of these parameters. However, since these samples are IID, they can be generated in parallel. Implementing the filter in a multithreaded fashion enables parallel sampling of the tract distribution.

ITK provides a framework for implementing multithreaded algorithms. The ITK multithreading framework assumes that the output region can be divided into disjoint sections with each thread working exclusively on their own section of the output image. This design prevents threads from simultaneously writing to the same memory region, which may cause unexpected results. However, since the stochastic tractography algorithm generates tracts that may span the entire output image, dividing the output region into disjoint sections is not possible. Additionally, in order to obtain statistics on these tracts, we need to output the generated tracts as well as the resultant connectivity image. Thus the existing ITK framework for implementing multithreaded filters is not very useful for our stochastic tractography filter. Fortunately ITK also provides basic multithreading functions which allowed us to create a custom multithreaded design that is still within the ITK framework.

Each thread of stochastic tractography filter is an instance of the stochastic tractography algorithm. The block diagram in figure 3 demonstrates graphically the architecture of the ITK stochastic tractography filter. Every thread allocates it own independent memory for the tract that it is currently generating. Once the tract has terminated, the thread stores a memory pointer to the completed tract in a tract pointer container that is shared among all threads. The tract pointer container is protected by a mutex, which serializes write operations so that only one thread can store its completed tract in the vector at a time. Once the filter has generated enough samples, the tracts can be transferred to an output image to create a connectivity map. Additionally other statistics can be computed on the tracts. In essence, we divide the process into two sections, a multithreaded portion that samples the tracts and a single threaded portion which accumulates the tracts and calculates relevant statistics on them.

The most computationally expensive part of the algorithm is the calculation of the likelihood distribution. The algorithm must compute probabilities for 2,562 possible fiber orientations in a voxel. Fortunately, this likelihood distribution is a deterministic function of the diffusion observations within that voxel. The filter runs much faster, at the cost of additional memory, by caching the generated likelihood distribution for later access. Caching is effective because in highly anisotropic regions of the brain, the sampled tracts are expected to be dense causing many of the sampled tracts to visit the same voxels many times.

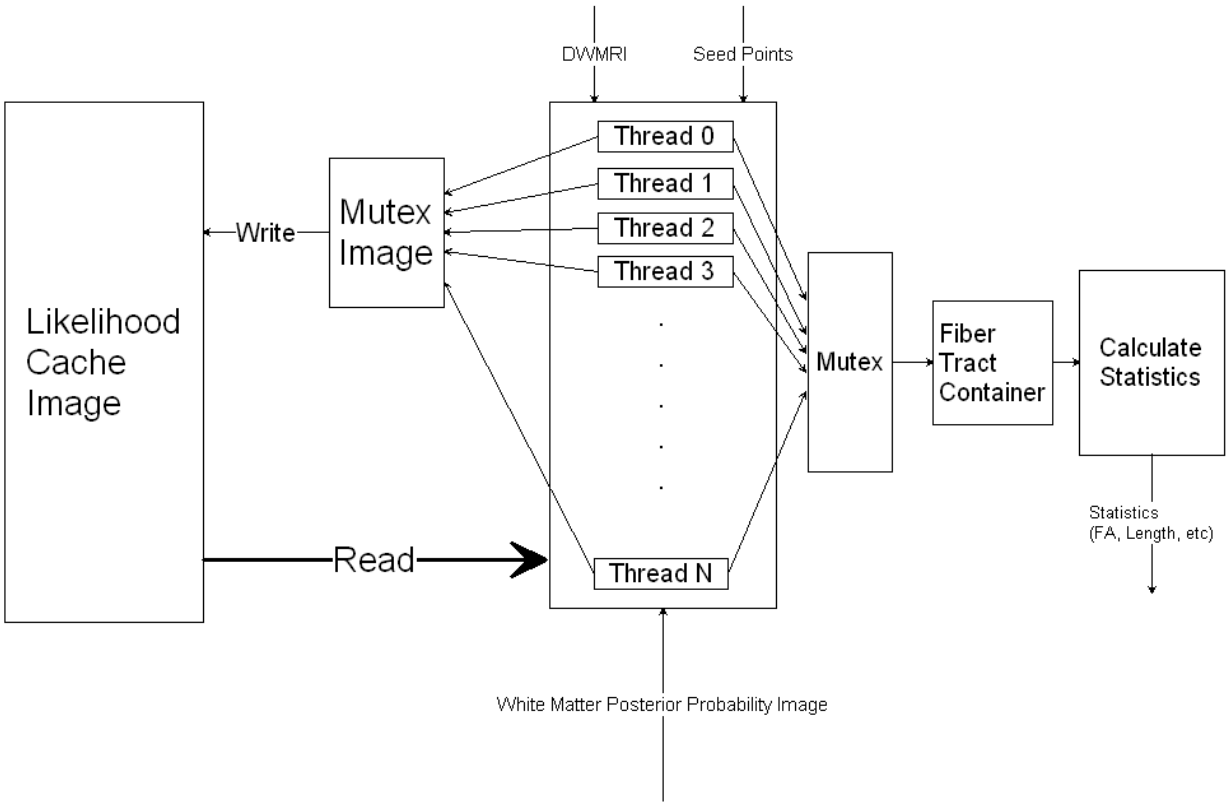

Figure 3: A block diagram of the filter showing its shared likelihood cache and multithreaded architecture.

The cache is implemented as an image whose voxels are re-sizable arrays. ITK's optimized pixel access capabilities enable quick access to the likelihood distribution associated with any voxel in the image. On creation, every voxel in the likelihood cache image is initiated to a zero length array. Whenever the algorithm encounters a voxel, it first checks to see if the likelihood cache contains this voxel by testing if associated array is zero length. If the voxel has never been visited, the associated array is resized and the computation of the likelihood distribution associated with this voxel is stored inside the newly resized array.

Using a shared likelihood cache between multiple concurrent threads creates additional complexities. Simultaneous writes to the cache would cause unexpected behavior. Additionally there is the possibility of one thread reading an incomplete cache entry while another thread is trying to write it. One possible solution is to ensure that only one thread can read or write to the likelihood cache at a time. This is easily implemented by serializing access to the likelihood cache using a mutex. A mutex serves as a lock on data. A thread will wait to obtain a lock on the data before it proceeds to the next section of code. Inside this section, which is called the critical section, the thread holds the lock ensuring exclusive access to the otherwise shared data. All other threads must wait and idle while the thread which owns the lock finishes it operations. Since threads must access the likelihood cache very often, this results in a situation where many threads are waiting for other threads to finish accessing the likelihood cache. The serialized access to the likelihood cache creates a bottleneck, which in the worse case would result in performance that is only marginally better than a single threaded version of the filter.

Access collisions to the likelihood cache can be reduced if we increase the granularity of the lock. Instead of using one large lock for the entire likelihood cache image, we use a lock for each voxel. The probability

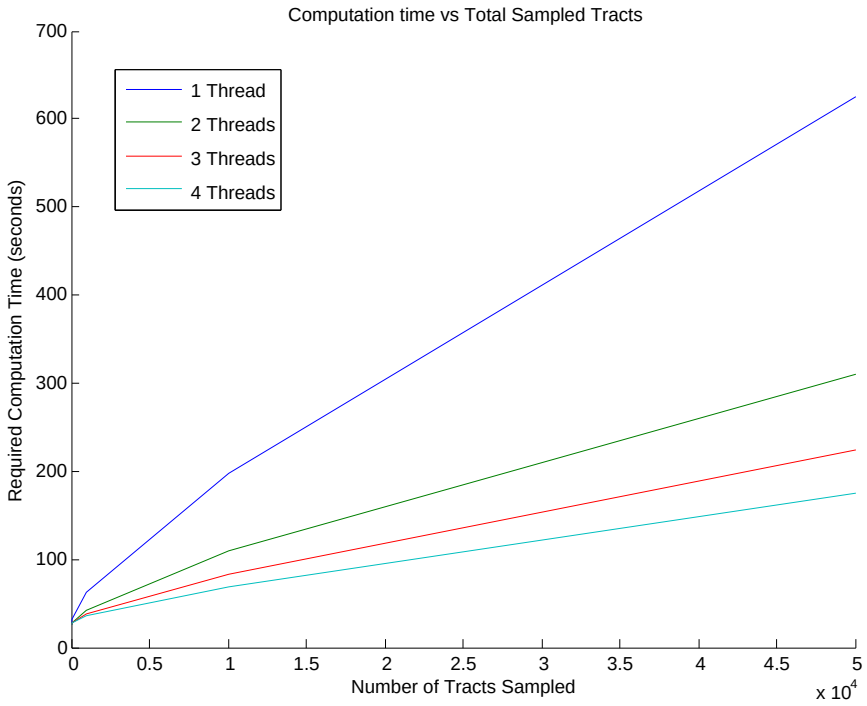

Figure 4: A graph displaying the amount of time needed to sample a number of tracts. Each line represents the algorithm's performance using different numbers of threads. This test was run on a 4 processor machine.

of two threads accessing the same voxel simultaneously is much less than the probability of two threads accessing any part of the likelihood cache. These per voxel locks are conveniently constructed using an ITK image whose voxel data type is a mutex. Similar to the likelihood cache, this collection of mutexes is indexed by coordinates which correspond to the coordinates of the voxels in the DWI input data. Again, access to the mutex image is fast due to ITK's optimized access operators for data types indexed by coordinates. The only cost to using this high resolution mutex image is the additional memory required to store pixel mutexes. However this cost is small since a mutex is essentially a Boolean variable. The mutex image allows different voxels in the likelihood cache image to be updated simultaneously, increasing the rate that the likelihood cache is filled. The advantage of using a mutex image is most evident when tracking in highly isotropic regions, where collisions are very unlikely to occur, since the sampled paths are very dispersed. Even on uni-processor systems, using multiple threads may improve performance since the rate of encountering an unvisited voxel may be higher, thus filling the likelihood cache faster. Figure 4 demonstrates the required computation time for a given number of tracts under different number of threads.

Additionally, to compute the weighted least squares estimates for the log tensor model parameters, we must first estimate the weights. These weights are found by calculating a least squares estimate of the true intensities of each voxel. The design matrix used in this least squares estimation is a function solely of the magnetic gradient directions and associated b-values, which are the same for every voxel in the image. Since the same design matrix is used for each voxel in the least squares calculation, a common optimization is to orthogonalize the design matrix by computing its QR decomposition. While this operation is computationally expensive, it is performed only once for the entire DWI image. The orthogonalized design matrix reduces the cost of computing the weights for every voxel.

4 User's Guide

4.1 ITK Stochastic Tractography Filter

The Stochastic Tractography Filter is implemented as a multithreaded image filter in ITK under the class name itk::StochasticTractographyFilter. The filter is templated over the DWI and white matter probability map input image types and also on the connectivity map image type. The filter expects the DWI input image type to be an ITK VectorImage Type. The code below demonstrates how to instantiate the Stochastic Tractography Filter.

```
//Define Types
typedef itk::VectorImage< unsigned short int, 3 > DWIVectorImageType;
typedef itk::Image< float, 3 > WMPImageType;
typedef itk::Image< unsigned int, 3 > CImageType;
typedef itk::StochasticTractographyFilter< DWIVectorImageType, WMPImageType,
  CImageType > PTFilterType;
//Allocate Filter
PTFilterType::Pointer ptfilterPtr = PTFilterType::New();
```

The filter's required inputs and parameters must be set before it can be run. Table 1 lists filter methods that should be called to set the required inputs and parameters, and a short description of what each methods expects as arguments.

The code below is a continuation of the demonstration above and shows how to setup the filter's required inputs and parameters. The inputs to these methods are provided by ITK's image readers.

```
ptfilterPtr->SetInput( dwireaderPtr->GetOutput() );
ptfilterPtr->SetWhiteMatterProbabilityImageInput( wmpreader->GetOutput() );
ptfilterPtr->SetbValues(bValuesPtr);
ptfilterPtr->SetGradients( gradientsPtr );
ptfilterPtr->SetMeasurementFrame( measurement_frame );
ptfilterPtr->SetMaxTractLength( maxtractlength );
ptfilterPtr->SetTotalTracts( totaltracts );
ptfilterPtr->SetMaxLikelihoodCacheSize( maxlikelihoodcachesize );
ptfilterPtr->SetSeedIndex( seedindex );
```

The filter can then be run by calling the Update method.

```
ptfilterPtr->Update();
```

For the specified seed voxel, the filter outputs a connectivity map and a container holding all of the sampled tracts used to generate the connectivity map. The container of sampled tracts can be further processed outside of the stochastic tractography filter to obtain various statistic on the sampled tracts. Additional seed voxels can be included in the seed region by changing the seed voxel index and rerunning the filter. The statistics for a multi-voxel seed region can be analyzed by accumulating statistics for all seed voxels within the seed region. These outputs can be accessed by calling the `GetOutput` and `GetOutputTractContainer` methods after calling the `Update` method. The code below continues the example above and demonstrates how to obtain the filter's outputs.

Filter Member Method	Description
SetInput	DWI Image: An ITK VectorImage consisting of a vector of DWI measurements including the baseline b0 measurements, at each voxel.
SetWhiteMatterProbabilityImageInput	White Matter Probability Input: An ITK image whose voxel values range from 0 and 1 representing the posterior probability that the voxel is a white matter.
SetbValues	b-Values: An ITK VectorContainer whose elements are the corresponding b-values for the DWI input image. The b0 measurements must have a 0 b-value.
SetGradients	magnetic gradient directions: An ITK VectorContainer whose elements are 3 dimensional vnl vectors. These vectors should be unit length.
SetMeasurementFrame	DWI Measurement Frame: A 3x3 vnl matrix which transforms the gradient directions to the physical reference frame of the image. For instance multiplying a magnetic gradient direction vector by the Measurement Frame Matrix will take the vector to the RAS reference frame if RAS is the physical frame of the DWI image.
SetMaxTractLength	Maximum Tract Length: A positive integer that sets the maximum length of a sampled tract. This can also be interpreted as the number of segments which comprise the tract when using the default step size of 1 unit in the physical frame of the DWI image.
SetTotalTracts	Total Sampled Tracts: A positive integer that sets the total number tracts to sample from the seed voxel.
SetMaxLikelihoodCachSize	Maximum Likelihood Cache Size(MB) A positive integer that sets the maximum size of the Likelihood Cache in megabytes.
SetSeedIndex	Seed Voxel Index: The discrete index of the seed voxel, in the (IJK) reference frame of the image to start tractography.

Table 1: ITK Stochastic Tractography Filter Required Inputs and Parameters

```
PTFilterType::TractContainerType::Pointer tractcontainer =
  ptfilterPtr->GetOutputTractContainer();
CImageType::Pointer cmap = ptfilterPtr->GetOutput();
```

4.2 Command Line Module Interface

The command line module interface provides an easy to use method of performing common tasks which use the ITK stochastic tractography filter. The command line module takes as input a DWI volume, a white matter probability map and a label map to produce a connectivity probability map and fractional anisotropy and length statistics for a selected seed region in the label map. Currently the command line module is designed to work only with NRRD formatted volumes due to its support of the diffusion measurement frame, but future revisions of the software will extend support to other formats.

The command line module is named `StochasticTractographyFilter`. Calling the executable with the `--help` flag will list all available inputs and options as well as a short description of each item. This section will demonstrate two typical usages of the command line module interface.

Given a DWI volume, an associated white matter probability map and a label map, the command line module can be used to generate an image that provides the probability of connectivity from an ROI in the label map to all other voxels in the DWI. The label map is an integer valued image that segments voxels into different classes or labels.

Let `case24` be the name of the subject we are interested in analyzing. The directory `case24` contains all relevant files for that subject. It will also hold the output files generated by the command line module. Before executing the command line module interface, a possible list of files in the `case24` directory may include:

```
case24_DWI.nhdr (DWI NRRD header)
case24_DWI.raw (DWI NRRD data)
case24_whitematterPB.nhdr (White Matter Probability Map NRRD header)
case24_whitematterPB.raw (White Matter Probability Map NRRD data)
case24_labelmap.nhdr (Label Map NRRD data)
case24_labelmap.raw (Label Map NRRD data)
```

Assuming that the starting ROI is labeled 15 inside the labelmap, the stochastic tractography filter can be run by executing the command below within the `case24` directory: To run the command line module, execute the command:

```
StochasticTractographyFilter -c 6500 -m 500 -t 200 -e 15 -l 15
-r -o case24_RUN0 case24_DWI.nhdr case24_whitematterPB.nhdr
case24_labelmap.nhdr
```

After the command completes, the `case24` directory will contain the following additional files:

```
case24_RUN0_CMAP.nhdr (Connectivity Map NRRD header)
case24_RUN0_CMAP.raw (Connectivity Map NRRD data)
case24_RUN0_TENSOR.nhdr (Tensor image NRRD header)
case24_RUN0_TENSOR.raw (Tensor image NRRD data)
case24_RUN0_COND.nhdr (Conditioned Connectivity Map NRRD header)
```

```
case24_RUN0_COND.raw (Conditioned Connectivity Map NRRD data)
case24_RUN0_CONDFAValues.txt (Conditioned Tract-Averaged FA values)
case24_RUN0_CONDLENGTHValues.txt (Conditioned Tract Length values)
```

The conditioned connectivity map is identical to the normal connectivity map when the start label and end labels are the same. However, if the label map contains ROIs designated by two labels, the conditioned connectivity map will be generated using only fibers which start in the start ROI and also pass through the second ROI. Assuming the second ROI is labeled 2 in the labelmap, the following command will isolate tracts which start in the start ROI and pass through the end ROI:

```
StochasticTractographyFilter -c 6500 -m 500 -t 200 -e 2 -l 15
-r -o case24_RUN0 case24_DWI.nhdr case24_whitematterPB.nhdr
case24_labelmap.nhdr
```

Now the conditioned connectivity maps and statistics are generated using only tracts which fulfill the condition of passing through both ROIs. This feature allows us to analyze the particular bundle of tracts which connect two regions.

4.3 3D Slicer Interface

To encourage the algorithm's adoption in clinical studies, we created an interactive GUI module for the 3D Slicer medical image visualization program was created which interfaces with the ITK stochastic tractography filter.

The module was implemented using the command line module interface provided by the 3D Slicer environment. This interface greatly eased the adaption of the command line interface into a graphical interface that could be included with 3D Slicer. The command line interface and the graphical interfaces are both completely described using an XML file. This XML file description is then parsed by a program provided by 3D Slicer which generates code that can be included with the command line interface to create what 3D Slicer refers to as a command line module. Command line modules can be run independently of 3D Slicer but can also be incorporated in to the 3D Slicer graphical interface. This enables the stochastic tractography algorithm to function as an easy to use extension in the 3D Slicer program as well as a stand alone program suitable for processing a large numbers of data sets non-interactively.

5 Conclusion

In this paper we have implemented a stochastic tractography system that can be used to analyze white matter architecture from DTI images. The multithreaded stochastic tractography ITK filter allows parallel sampling of fiber tracts, reducing computation time on multi-processor systems. Finally, we have created easy to use command line and 3D Slicer graphical interfaces to facilitate algorithm's use in clinical studies.

6 Acknowledgments

We would like to acknowledge the support from the following NIH grants: P41 RR13218, R01 MH074794, and U54 EB005149 (National Alliance for Medical Image Computing, NA-MIC).

References

[1] PJ Basser and S. Pajevic. In vivo fiber tractography using dt-mri data. *Magn Reson Med*, 44:625–632, 2000. 1

[2] T.E.J Behrens, M.W. Woolrich, M. Jenkinson, H. Johansen-Berg, R. G. Nunes, S. Clare, P.M. Matthews, J.M Brady, and S.M. Smith. Characterization and propagation of uncertainty in diffusion-weighted mr imaging. *Magnetic Resonance in Medicine*, 50:1077–1088, 2003. 1

[3] M. Björnemo, A. Brun, R. Kikinis, and C.-F. Westin. Regularized stochastic white matter tractography using diffusion tensor MRI. In *Fifth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, pages 435–442, Tokyo, Japan, 2002. 1

[4] Brigham and Women's Hospital. 3d slicer medical visualization and processing environment for research. *http://www.slicer.org/*. 1

[5] O. Friman and C.-F. Westin. Uncertainty in fiber tractography. In *Eighth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'05)*, Lecture Notes in Computer Science 3749, pages 107–114, Palm Springs, CA, USA, October 2005. 1

[6] Ola Friman, Gunnar Farneback, and Carl-Fredrik Westin. A Bayesian approach for stochastic white matter tractography. *TMI*, 25(8):965–978, 2006. 1, 2

[7] Derek K. Jones and Carlo Pierpaoli. Confidence mapping in diffusion tensor magnetic resonance imaging tractography using a bootstrap approach. *Magnetic Resonance in Medicine*, (5):1143–1149, 2005. 1

[8] M. Lazar and A. Alexander. Bootstrap white matter tractography (boot-trac). *NeuroImage*, 24:524–532, 2005. 1

[9] S. Mori, B. Crain, V.P. Chacko, and PCM van Zijl. Three dimensional tracking of axonal projections in the brain by magnetic resonance imaging. *Ann Neurol*, 45:265–269, 1999. 1

[10] D.S. Tuch, M.R. Wiegell, T.G. Reese, J.W. Belliveau, and V.J. Weeden. Measuring cortico-cortical connectivity matrices with diffusion spectrum imaging. In *Proc. of the International Society for Magnetic Resonance Medicine*, page 502, Glasgow, Scotland, 2001. 1