2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Probabilistic Brain Fiber Tractography on GPUs

Mo Xu*, Xiaorui Zhang*, Yu Wang*, Ling Ren*, Ziyu Wen*, Yi Xu*, Gaolang Gong†, Ningyi Xu‡ and Huazhong Yang*
*Department of Electronic Engineering
Tsinghua National Laboratory for Information Science and Technology, Tsinghua University
Email: {xumo08,zxr08}@mails.tsinghua.edu.cn; yu-wang@tsinghua.edu.cn
†State Key Laboratory of Cognitive Neuroscience and Learning, Beijing Normal University
‡Hardware Computing Group, Microsoft Research Asia

***Abstract*—Diffusion Tensor Magnetic Resonance Imaging (DT-MRI) is an emerging technique that explores the structural connectivity of the human brain. The probabilistic fiber tractography based on DT-MRI data behaves more robustly than deterministic approaches in the presence of fiber crossings, but requires more prohibitive computational time. In this work we present a GPU-based probabilistic framework for brain fiber tractography. The framework includes two main steps: 1) Markov-Chain Monte-Carlo (MCMC) sampling, and 2) probabilistic streamlining fiber tracking. We implement the Metropolis-Hastings sampling for local parameter estimation on GPU. In the probabilistic streamlining fiber tracking, we find that fiber lengths are exponentially distributed, and propose a novel segmenting strategy to improve the load balance. On mid-range GPUs, we achieve performance gains up to 34x and 50x over CPUs for the two steps respectively.**

***Keywords*-GPU; Probabilistic fiber tractography; DT-MRI; MCMC; Probabilistic Streamlining**

I. Introduction

Diffusion Tensor Magnetic Resonance Imaging (DT-MRI) is a non-invasive technique that measures the local diffusion of water molecules in vivo [1]. Within the white matter of the human brain, the random thermal movement of water is restricted along the direction of fiber bundles, imposing anisotropy on the molecular diffusion. DT-MRI can characterize this anisotropy and provide information on the presence and orientation of fibrous tissue. Based on this information, researchers can locate the fiber pathways underlying the human brain using tractography algorithms [2] [3]. The reconstructed fibers are of great value in neuroscience research and clinical applications such as surgical planning and diagnosis of neurological disorders [4] [5] [6] [7].

Traditional deterministic tractography algorithms are analogous to the method for determining streamlines in fluid dynamics. They draw fiber paths by repeatedly stepping in the principal diffusion orientation [5]. Such methods succeed in reconstructing the fiber bundles in clinically acceptable time, but have several drawbacks. First, They are very sensitive to noise, which may lead to large, accumulative errors in the trajectory of fibers [8]. Second, they may be disturbed by the presence of fiber crossings or bifurcations [9] [7], which is rather common, since the diameter of an axon is well beyond the resolution of a current MRI scan [10]. Third, they do not provide the confidence in the estimated fiber paths; and if not interpreted carefully, they might even give an impression of false certainty [11](finding "fibers" that do not exist at all).

In response, Behrens et al. proposed a probabilistic framework for brain fiber tractography, which takes into account the uncertainty stemming from noise and diffusion model imperfections [12] [13]. Based on Bayesian estimation, they estimate both the local probability density functions (*pdf*) on parameters of interest in a diffusion model and the global connectivity, i.e. the probability of the existence of a connection between two voxels. This framework is adopted in FMRIB Software Library (FSL), one of the most popular open-source tools for brain imaging analysis [14] [15].

However, the probabilistic analysis of one brain at regular spatial and angular resolution costs nearly one day on common CPUs. The intensive computation in the probabilistic method inhibits the research and their application in routine clinical tasks [7]. The two most time-consuming steps of probabilistic fiber tractography are iteratively drawing samples from complicated and high dimensional *pdf*s using Markov-Chain Monte-Carlo (MCMC), and the subsequent probabilistic streamlining fiber tracking based on these random samples.

Recent advances in Graphic Processing Units (GPUs) provide a solution to such difficulties. The GPU, with massively parallel processors, proves suitable for general-purpose data-parallel scientific computation [16]. In this work, we propose a GPU-based framework for probabilistic fiber tractography. We test our GPU-based framework on two open datasets, and achieve performance gains up to 34x and 50x for MCMC sampling and probabilistic fiber tracking on mid-range GPU over CPUs. The main contributions of our work are:

- We for the first time accelerate the Bayesian framework for probabilistic brain fiber tractography on GPUs.
- We find that the fiber lengths in probabilistic tractography are exponentially distributed. Based on this finding, we propose a flexible segmentation strategy to balance the load between GPU SIMD threads while maintaining low communication overhead.

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.92

The rest of our paper is organized as follows: we review the related works in Section II, and describe the preliminary methods in Section III. In Section IV, we present our GPU-based framework. Experimental results are provided in Section V. Section VI concludes the paper.

II. Related Work

Most previous work on GPU-based fiber tractography focuses on enhancing the visualization of the results [17] [18] [19] [20] [21]. Besides, great efforts have been made to improve the interactivity of the fiber tractography [22] [23] [24] [8] [25]. All of them use deterministic streamlining algorithms. Mittmann et al. presented a GPU acceleration for deterministic streamlining method [26]. In their implementation, CPU performs a reduction step immediately after every trajectory point calculation on the GPU. Thus the frequent communication between CPU and GPU degrades the overall performance.

Based on Behrens' full Bayesian framework, Friman et al. proposed an empirical Bayesian model [27]. They replaced the MCMC sampling with point estimation for computational tractability. However, the equivalence of these two methods is still under investigation [11] [27]. McGraw accelerated this modified method on the GPU [28].

The GPU-based MCMC technique has recently been used in several other applications, such as high-dimensional optimization [29], chemical kinetic models [30], signal detection and pattern recognition [31], and network learning [32]. A package, *cudaBayesreg*, for Bayesian analysis based on CUDA has been built with the Gibbs sampler [33]. But Gibbs sampler does not fit our application, since it is hard to obtain the conditional distribution for each parameter. GPU-based Metropolis-Hastings sampler has not been fully discussed in these works.

III. Preliminary Method

In this section, we introduce the basic method for probabilistic brain fiber tractography. This method is based on Bayesian estimation, and can be divided into two steps: the local parameter estimation and the global connectivity estimation. Given the data from the DT-MRI scan, and the parameterized diffusion model, the local parameter estimation is to estimate the *pdf*s of the parameters in the diffusion model of each voxel. Then based on these local *pdf*s, the global connectivity estimation is to estimate the probability of the existence of a connection between all voxel-pairs in the brain.

A. Local Parameter Estimation

1) Diffusion Model: Diffusion models predict how the local water diffusion profiles determine the voxel intensities μ_i in the diffusion weighted images. Three models are listed in Table I. The gradient directions $\hat{\mathbf{r}}_\mathbf{i}$ and the b-value b_i are known experimental parameters. In fiber tractography, the diffusion tensor $\mathbf{D}$ or the local fiber direction $\hat{\mathbf{v}}$ is the parameter of interest. The direction of fiber $\hat{\mathbf{v}}$ can also be expressed in spherical coordinate with (θ, ϕ).

Table I
THREE DIFFUSION MODELS

Tensor model	$\mu_i = S_0 e^{-b_i \hat{\mathbf{r}}_\mathbf{i}^\mathbf{T} \mathbf{D} \hat{\mathbf{r}}_\mathbf{i}}$
Constrained model	$\mu_i = S_0 e^{-\alpha b_i} e^{-\beta b_i (\hat{\mathbf{r}}_\mathbf{i}^\mathbf{T} \hat{\mathbf{v}})^2}$
Compartment model	$\mu_i = S_0 \left[(1-f) e^{-b_i d} + f e^{-b_i d (\hat{\mathbf{r}}_\mathbf{i}^\mathbf{T} \hat{\mathbf{v}})^2} \right]$

Among the three models, the constrained model and the compartment model (also called single partial volume model) consider the effect of the underlying fiber tract. Yet both of them only consider single underlying fiber tract. Due to the relatively low spatial resolution of DTI data, voxels may contain several different fiber tracts. To model multiple fibers within a voxel, Behrens proposes the multiple partial volume model [12], where a voxel can be divided into several sub-voxels, and each of them has only one fiber direction through it. It assumes that the MR signal μ_i from a voxel is the sum of those from their sub-voxels,

$$\mu_i = \sum_{j \in sub-voxels} \mu_{ij}$$

So according to the compartment model, the predicted signal for each diffusion-weighted measurement at each voxel is:

$$\mu_i = S_0 \left[(1 - \sum_{j=1}^{N} f_j) e^{-b_i d} + \sum_{j=1}^{N} f_j e^{-b_i d (\hat{\mathbf{r}}_\mathbf{i}^\mathbf{T} \hat{\mathbf{v}}_\mathbf{j})^2} \right] \quad (1)$$

In our work, we use this multiple partial volume model, and let $N = 2$ to avoid over fitting. This model is also adopted by FSL.

Given this model M, our mission is to obtain the *pdf*s of the parameters from the observed signals $\mathbf{Y}$. Following the Bayes' Theorem, it is proven in [12] that the marginal posterior *pdf* of the parameter subset of interest $\omega_I = \{f_1, f_2, \theta_1, \theta_2, \phi_1, \phi_2\}$ can be calculated using the following integral,

$$\begin{aligned} P(\omega_I | \mathbf{Y}, M) &= \int_{\mathbf{\Omega}_{-I}} P(\omega | \mathbf{Y}, M) d\omega_{-I} \\ &\propto \int_{\mathbf{\Omega}_{-I}} P(\mathbf{Y} | \omega, M) P(\omega | M) d\omega_{-I} \end{aligned} \quad (2)$$

where $\omega_{\neg I}$ refers to all the other parameters.

2) MCMC Sampling: The integral in (2) is analytically intractable. So we use MCMC method to obtain some samples for our objective distribution. We first construct a Markov chain with the target *pdf* as its stationary distribution. By taking samples from the Markov chain, we obtained a estimation of $P(\omega | \mathbf{Y}, M)$, from which we can obtain the estimation of our target distribution $P(\omega_I | Y, M)$

Generally, there are two widely used MCMC algorithms: the Gibbs Sampler and the Metroplis-Hastings (MH) Sampler [34]. The Gibbs Sampler does not fit our problem since it is not possible to obtain the full conditional distributions for each parameter, which are essential for Gibbs Sampler [35]. So we choose MH sampler.

The procedures of one MH step are as follows:

- Propose the candidate state ω' from the current state $\omega^{(t)}$ with the proposal distribution $P(\omega^{(t)}, \omega')$.
- Compute the ratio

$$r = \frac{P(\omega'|\mathbf{Y}, M)}{P(\omega^{(\mathbf{t})}|\mathbf{Y}, M)}$$

- Decide whether to accept ω'. Accept ω' as $\omega^{(t+1)}$ with probability $min(r, 1)$. If ω' is not accepted, $\omega^{(t+1)} = \omega^{(t)}$.

The proposal distribution $P(\omega^{(t)}, \omega')$ here is a zero mean Gaussian distribution $N(0, \sigma^2)$. The variance parameter σ^2 should be adjusted according to the acceptance rate to ensure that the acceptance rate is neither too high nor too low (somewhere between 25% and 50%).

B. Global Connectivity Estimation

In this subsection we introduce how to estimate the global connectivity with the local *pdf*s at each voxel obtained from local parameter estimation.

In the tracking process, single partial volume model assumes only one fiber direction, and multiple partial volume model allows for multiple fiber directions. In this study, we adopt the multiple partial volume model to allow for the presence of fiber crossings or bifurcations. There is only a slight difference between the two models, so we first introduce the single partial volume model, and then extend it to the multiple partial volume model.

1) Single Local Fiber Direction: $P(\exists A \rightarrow B|\mathbf{Y})$ denotes the probability of a connection existing between points A and B, given data $\mathbf{Y}$. According to Bayes' theorem,

$$P(\exists A \rightarrow B|\mathbf{Y}) = \int_0^{2\pi}\int_0^{\pi}\cdots\int_0^{2\pi}\int_0^{\pi} P(\exists A \rightarrow B|(\theta,\phi)_{\mathbf{x}})$$
$$P((\theta,\phi)_{x_1}|\mathbf{Y})\cdots P((\theta,\phi)_{x_m}|\mathbf{Y})d\theta_{x_1}d\phi_{x_1}\cdots d\theta_{x_m}d\phi_{x_m} \quad (3)$$

where the subscript $\mathbf{x}$ refers to every voxel in the brain. Note that $P(\exists A \rightarrow B|(\theta,\phi)_{\mathbf{x}})$ can be simply calculated using "deterministic streamlining" algorithm, which will be introduced in the next sub-section.

From Eq.3 we can see that $P(\exists A \rightarrow B|\mathbf{Y})$ reduces to $P(\exists A \rightarrow B|(\theta,\phi)_{\mathbf{x}})$ when the local *pdf*s on fiber direction $P((\theta,\phi)_{\mathbf{x}}|\mathbf{Y})$ are delta functions. This means when there is no uncertainty in the local fiber direction, Eq.3 reduces to the deterministic streamlining solution. In our case, we again need to compute these high dimensional integrals using Monte Carlo methods. Since we have already obtained the samples from the posterior *pdf*s $P((\theta,\phi)_{\mathbf{x}}|\mathbf{Y})$ at each point in the previous MCMC step, all we need to do is to construct the streamlines from A. This method is referred to as "probabilistic streamlining" [12]. The probabilistic streamlining algorithm is done by invoking deterministic streamlining for many times.

Having obtained the probabilistic streamlines from the seed point A with all the samples, we may then get the connectivity $P(\exists A \rightarrow B|\mathbf{Y})$ by simply counting the number of streamlines passing through B, and dividing it by the total number of the streamlines.

2) Multiple Local Fiber Directions: To make the above procedure amenable to the multi-fibre case, the only needed adaptation is that, at each step, a right direction should be chosen. In this choice, we aim at maintaining the original orientation of the streamline through crossing regions [13].

3) Deterministic Streamlining Tracking Algorithm: Now we introduce the deterministic streamlining tracking algorithm. Before the tracking, we must set a step length and some termination criteria. Common termination criteria for deterministic streamlining include:

- a lowest anisotropy (f), to prevent the fiber from stepping into an area with low signal-to-noise ratio;
- a maximum number of steps, to avoid dead loops;
- a maximum angle formed by two subsequent fiber segments, to eliminate trajectories with sharp turns.

Since the probabilistic method is robust to noise, the lowest anisotropy criterion is not a must [12]. Thus we only use the last two criteria in our implementation.

The streamlining algorithm starts from a seed point A, and iteratively steps in the current direction by the preset step length, until one or more stop criteria are met. If B lies on this path, we say that a connection exists from A to B, and $P(\exists A \rightarrow B|(\theta,\phi)_{\mathbf{x}}) = 1$.

IV. The GPU-based Framework

Fig.1 shows the overall workflow of our GPU-based framework for probabilistic fiber tractography. The two major steps are implemented on the GPU. The first step is the MCMC sampling to draw samples from the *pdf*s of the diffusion magnitude and directions of each voxel. The inputs include a 4-D volume of the scanned DT-MRI data of a brain, a vector of b-values and a vector of gradient directions. By the MCMC sampling, we get several samples of the six parameters of interest for each voxel in the brain. The six parameters are the magnitudes and the directions of the two principal local fibers ($f_1, f_2, \theta_1, \theta_2, \phi_1, \phi_2$). The second step is the estimation of global connectivity by probabilistic streamlining fiber tracking. In this step, the deterministic streamlining tracking algorithm is performed iteratively to all sample volumes. Each sample generates a series of fiber paths from each voxels in the brain. We can further get the connectivity matrix P, in which P_{ij}

represents the probability that there exist a connection from i to j.

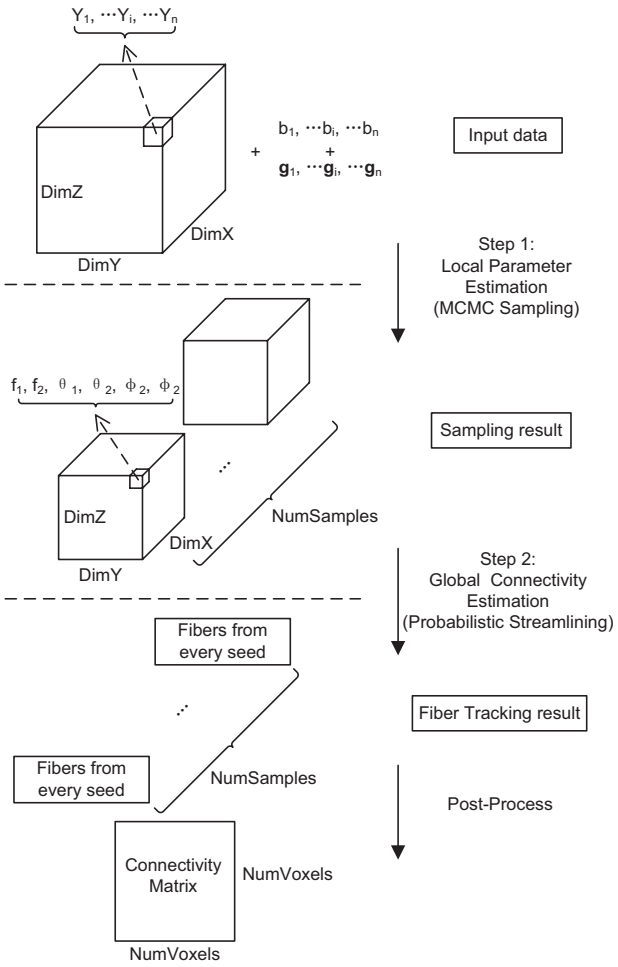


Figure 1. The Overall Workflow of the GPU-based Framework. The inputs include a 4-D volume ($DimX \times DimY \times DimZ \times n$) of the scanned DT-MRI data of a brain, a vector of b-values and a vector of gradient directions. The step 1 generates six 4D volumes ($DimX \times DimY \times DimZ \times NumSamples$) of the samples to illustrate the possible local fiber directions in each voxel. In step 2, streamlining tracking algorithm is performed to all sample volumes. The output can be fiber paths or/and the connectivity matrix ($NumVoxels \times NumVoxels$)

A. MCMC Sampling

The overall workflow of the MCMC process is shown in Fig.2. We use one thread for the MCMC of one voxel, since the MCMC processes for different voxels are completely independent of each other.

One of the challenges in GPU implementation of MCMC sampling is to generate a large number of pseudorandom numbers. Since it takes number of runs (burn-in period) until the Markov chain approaches stationary, we begin to draw samples after $NumBurnin$ steps. To decrease the dependence between samples of the chain, we take samples

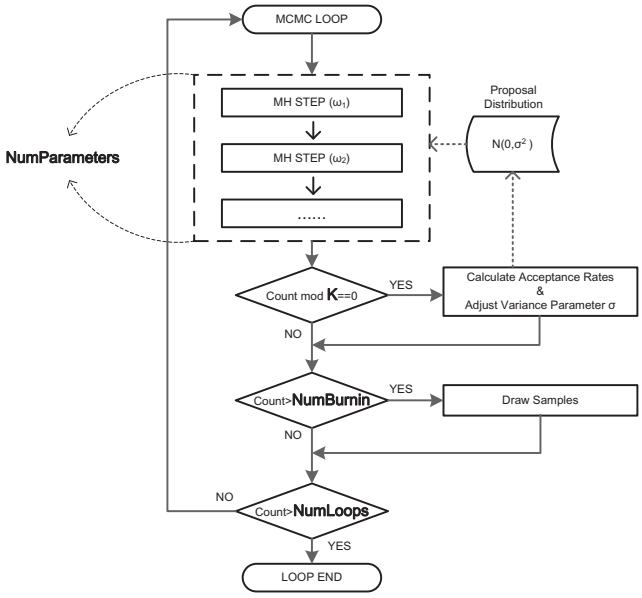


Figure 2. The MCMC sampling workflow. In each loop, the MH step is repeated $NumParameters$ times. For each K loops, the acceptance rates are calculated and the proposal distribution is updated. We begin to take samples after $NumBurnin$ steps. There are totally $NumLoops$ in the MCMC sampling process.

with an interval of L steps. Thus, the total number of loops is

$$NumLoops = NumBurnIn + NumSamples \times L$$

In each loop of Metroplis-Hastings sampling, we need two random numbers. One is for proposing the new candidate ω', and should meet the Gaussian distribution. The other is for calculating the acceptance rate, and should meet the uniform distribution. We get the Gaussian distributed random number by box-muller transformation [36] using two uniformly random number. Thus the total number of all random numbers needed is

$$NumVoxels \times NumLoops \times NumParameters \times 3$$

In our method, there are 9 parameters in ω to be estimated. Let $NumBurnIn$ be 500, L be 2, and $NumSamples$ be 250 as an example. In the two datasets we use, the $NumVoxels$ can be very huge, over 200,000. Hence the total memory needed for random numbers easily exceeds 20GB.

Thus, it is impractical to transfer the pre-generated random number from CPU to CPU. Instead, we use the combined Tausworthe algorithm described in [37] to efficiently generate pseudorandom numbers on the GPU.

B. Probabilistic Streamlining Fiber Tracking

The probabilistic streamlining fiber tracking algorithm (Algorithm 1) is a triple-nested loop. The innermost loop is

Algorithm 1 Probabilistic Streamlining Fiber Tracking Algorithm on GPUs

```
Require: NumIterations[NumSegments]
  InitializeGPU();
  for Every sample volume do
    Copy3DImagesToGPU();
    for i = 0 to NumSegments do
      SendStartPointsToGPU();
      NumThreads = NumStartPoints;
      LaunchGPUKernel(NumThreads,
      NumIterations[i])
      GPU kernel:
      for j = 0 to NumThreads in Parallel do
        GetStartPoint();
        for k = 0 to NumIterations[i] do
          Interpolation();
          StepToNextPoint();
          if Meet the Stop Criteria then
            break
          end if
        end for
        SetEndPoint();
      end for
      end GPU kernel
      ReadEndPointFromGPU();
      Reduction();
    end for
  end for
```

the deterministic streamlining algorithm, which sequentially locates the fiber pathway from one single seed using one sample. The processing for different samples or different seed points are independent of each other and can be parallelized. We parallelize the tracking tasks of different seed points, and serialize the tasks of different samples, to reduce the memory requirement. Each fiber pathway is located by a single thread in GPU, and the sample volume is shared by all the threads as read-only 3D images .

The greatest challenge is to improve the loadbalance on GPUs. In GPUs, a group of consecutive threads (referred to as a wavefront or a warp) execute instructions in a SIMD manner, which means that their running time is that of the slowest thread. Unfortunately, the fiber lengths (i.e. the number of steps) vary greatly in our problem. We depict the load of each thread in Fig. 6(a). We can see that it is very likely that consecutive SIMD threads have quite imbalanced workload. This greatly inhibits the full utilization of hardware resources.

Therefore, we propose dividing a streamlining into several segments to improve the loadbalance between GPU threads. Specifically, instead of finding a whole fiber pathway, a GPU kernel only locates a number of points from the given seed point and direction. Then a new kernel is launched to continue the streamlining only for those unfinished paths. We express the segmentation strategy with a *segmentation array*: $NumIteration[NumSegments]$, in which element $NumIteration[i]$ is the number of tracking iterations in the i^{th} GPU kernel launch. The sum of *segmentation array* determines the maximum number of steps ($MaxStep$) in the streamlining tracking algorithm.

Segmentation of the task can better utilize GPU hardware, but introduce the overhead of CPU reduction and CPU-GPU communication. Finer-grained segmentation strategy definitely results in less wasted hardware resources and higher overhead. The length m and values of the *segmentation array* are both to be determined. Thus it seems impossible to find a theoretically optimal strategy, given so many variables. Instead, we first get some insights by studying the following extreme cases, and then propose a heuristic strategy to approximate the optimal solution.

Extreme case 1: Minimize the Segment

One extreme segmentation strategy is no actual segmenting at all, equivalently $NumIteration[0] = MaxStep$ and $NumIteration[i] = 0$ for $i = 1, \cdots, m$. In this strategy, SIMD threads suffer from extremely imbalanced workloads. This straightforward implementation is shown in Fig. 3. In the figure, time advances along the vertical axis. The data needed and returned by GPU is transferred through PCI express bus. The width of the rectangle denotes the number of seed voxels, which is also the number of threads in the GPU kernel. The length of the rectangle denotes the number of steps. Thus the area of the rectangle represents the execution time.

Sorting the Load

One plausible solution for load imbalance is to predict and sort the load according to previous executions. In this way, consecutive SIMD threads are more likely to have similar loads. However, the effectiveness of this method depends on the similarity of fiber lengths from different samples of one seed. Unfortunately, experiments show that this method does

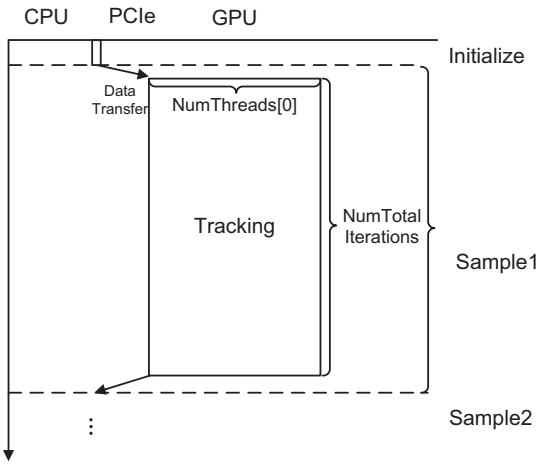
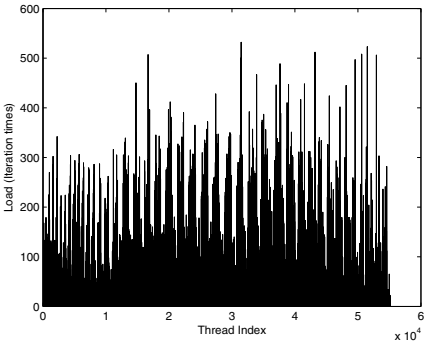
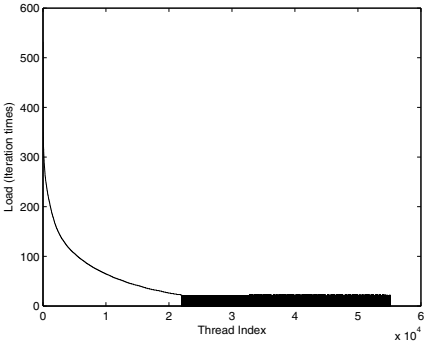


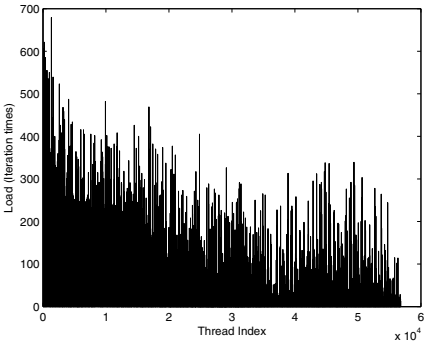
Figure 3. The Straightforward GPU Implementation of Probabilistic Fiber Tracking. Each thread track a whole fiber. Time advances vertically while CPU thread and parallel GPU threads are stacked horizontally. On the GPU side, the area of rectangles reflects the execution time.



(a) Load of Threads in Original Sequence



(b) Sorted Load



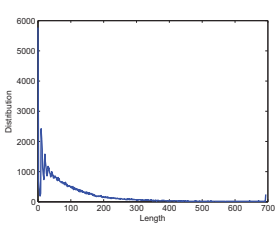
(c) Applying to Another Sample

Figure 4. Work Loads before and after sorting.

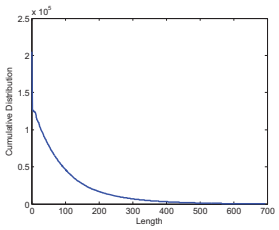
not bring any notable improvement at all. Fig.4(a) shows the loads of all threads in the original sequence and Fig.4(b) shows the loads in the sorted sequence. When applying this sequence to another sample, we can see in Fig.4(c) that although the general trends match, there still exists high variance between the loads of neighboring threads.

Extreme case 2: Maximize the Segments

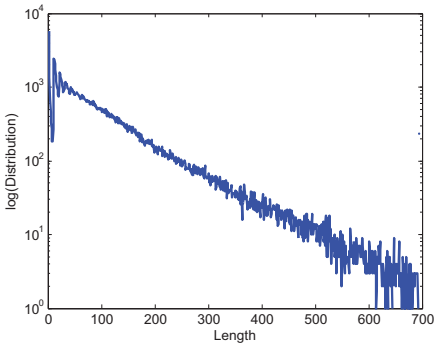
Another extreme segmentation strategy is that CPU reduction is performed at every advance of the trajectory, or equivalently $NumIteration[i] = 1$ for all $i = 0, 1, ..., n$. This strategy is adopted in [26]. In this strategy, all the threads



(a) The distribution of fiber lengths



(b) The cumulative distribution of fiber lengths



(c) The semi-log distribution of fiber lengths

Figure 5. Fiber Lengths Distributions. Figure (c) shows that the fiber length is exponentially distributed.

enjoy perfect loadbalance, and no GPU hardware resources are wasted, but CPU-GPU transfer time is maximized. In Section V, we will demonstrate that in this extreme case, the communication overhead overwhelm the benefit of GPU loadbalance.

Our strategy: Segments with Increasing Intervals

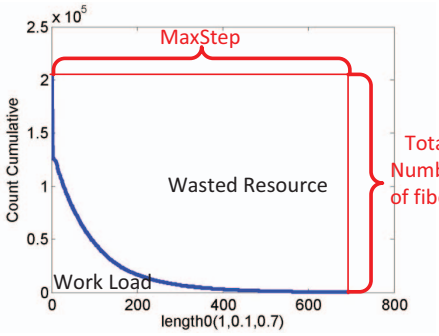
By investigating the histogram of the lengths of all fibers tracked from one sample, we find that the fiber lengths follow an exponential distribution, which is mathematically defined as:

$$p(x; \lambda) = \lambda e^{-\lambda x} \tag{4}$$

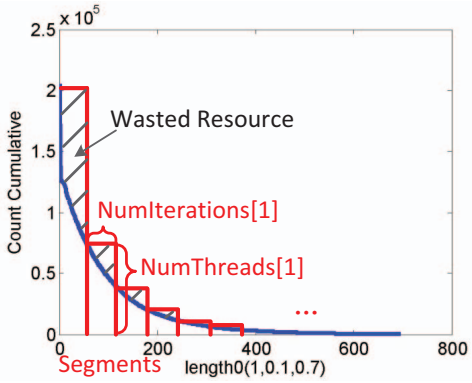
where x is the length of fiber. In [26] and experiments on three other data sets, similar distribution of fiber length is observed. Fig.5(a) and Fig.5(b) show the distribution and "cumulative" distribution $P(L > x)$ of actual fiber lengths respectively. Fig.5(c) is the semi-log plot of the distribution, which clearly indicates the exponential distribution.

We use the cumulative distribution figure to illustrate the load of GPUs and the resources that are wasted due to the load imbalance. In Fig.6(a), the *minimize segments* strategy is adopted. In this case, the maximum of fiber length reflects the actual number of iterations. The area beneath the cumulative function curve is the necessary workload.

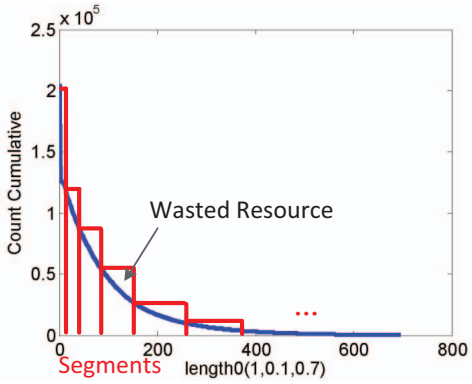
If all threads execute in a SIMD manner, the total execution time can be measured as the area of the rectangle in Fig.6(a), where most of the resources are wasted. However in GPUs, only 32 or 64 threads behave in a SIMD manner,



(a) Minimize the segments



(b) Segments with equal iterations



(c) Segments with Increasing iterations

Figure 6. The Load and the Utilization. The cumulative distribution is from tracking the smaller dataset, with step length = 0.1, and angular threshold = 0.7

so the total execution time is far less than what the rectangle represents. Despite this, we can still use Fig.6 to illustrate some guidance for choosing better segmentation strategies.

We observe that the fiber tracking starts with short iterations and huge number of threads, and ends with long iterations and few threads. Knowing the distribution, we come to the strategy that fill the $SegmentationArray$ with increasing integers, shown in Fig.7 and Fig.6(c). The wasted resource of this strategy is less than that in Fig.6(b).

CPU-GPU Overlap

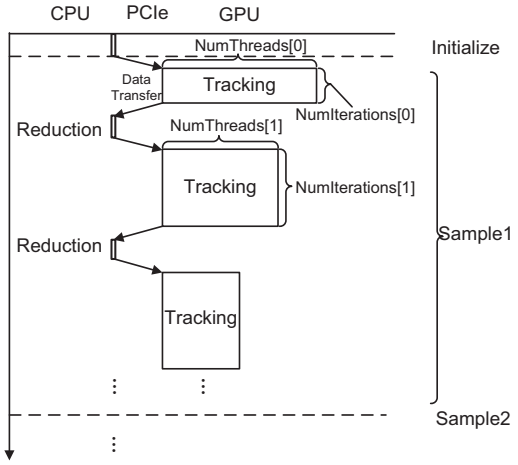


Figure 7. The segmentation method of Probabilistic Fiber Tracking on the GPU. The tracking task is segmented into many kernels. CPU compacts the unfinished pathways and launches new kernel.

To further reduce the overhead of CPU-GPU communication, we can overlap the tasks of CPU and GPU. Fig.8 demonstrates this method. Since the next tracking segment cannot be launched until the reduction of current segment is completed, overlapping the different segments in one sample is infeasible. Instead, we can track from two samples at the same time in a interleaving way. Meanwhile, the sample volume on the GPU also doubles. We leave this as future work.

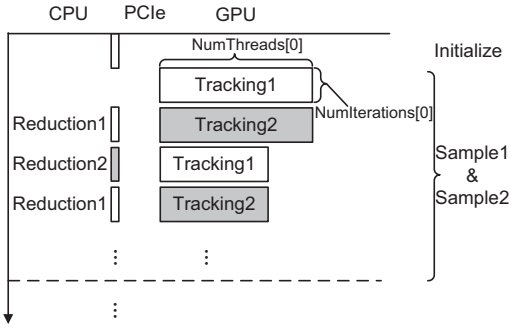


Figure 8. The Overlapped Sectional GPU Implementation of Probabilistic Fiber Tracking. The task of CPU and GPU can be overlapped.

V. Results

A. Experimental Setups

Our testing platform comprises a desktop PC featuring AMD Phenom X4 965 3.4GHz CPU and 8GB RAM, and AMD Radeon 5870 GPU. The C/C++ implementation is compiled using MS VC10.0 compiler with the O2 option, and the GPU implementation uses OpenCL with AMD APP SDK 2.0.

Table II
THE SPEEDUP OF PROBABILISTIC STREAMLINING

Dataset	Step length	Agular threshold	Longest fiber length	Total fiber length	Kernel time(s)	Reduction time(s)	Transfer time(s)	CPU time(s)	Speedup
1	0.1	0.9	453	113,822,762	3.02	0.78	2.94	289.6	43.0
	0.2	0.8	304	102,796,526	2.73	0.92	2.32	271.7	45.5
	0.3	0.85	286	109,408,821	2.71	0.78	2.33	306.6	52.7
2	0.1	0.9	777	305,396,623	6.78	3.77	4.29	739.6	52.0
	0.2	0.85	476	272,836,940	6.42	3.35	4.38	702.8	49.7
	0.3	0.8	517	2,913,939,11	6.63	3.38	4.37	784.5	54.5

The two testing data sets are downloaded from http://www.cabiatl.com/CABI/resources/dti-analysis/. The first dataset is 48 $\times$ 96 $\times$ 96 in size with a resolution of 2.5mm $\times$ 2.5mm $\times$ 2.5mm. The second dataset is 60 $\times$ 102 $\times$ 102 in size with a resolution of 2mm $\times$ 2mm $\times$ 2mm.

B. Performance

1) MCMC sampling: The performance of MCMC sampling is shown in Table.III. For each valid (white matter) voxel, we obtain 50 samples. The sample interval L is 2. The number of valid voxels are also provided in Table.III. For both the two data sets, GPU programs show about 34x speedup compared with CPU programs.

Table III
SPEEDUP OF DIFFUSION PARAMETER SAMPLING

Dataset	# of Voxels	CPU time(s)	GPU time(s)	Speedup
1	205,082	1383	41.3	33.6
2	402,194	2724	80.1	34.0

2) Probabilistic Streamlining: Table II shows the performance of probabilistic streamlining algorithm. The total work load is shown by the column $Totalfiberlength$. For GPU program, we list the GPU kernel time, CPU reduction time, CPU-GPU transfer time, and total running time. The test is preformed to the two datasets under varies step lengths and thresholds of turn angular (measured by the dot product of the two regular direction). The number of samples is 50. The overall speedups are listed in the right column. The segmentation strategy adopted here is our proposed increasing-interval strategy, where $NumIteration[]$={1, 2, 5, 10, 20, 50, 100, 200, 500, 1000}.

We then compare the performance of different segmentation strategies in Table IV. We denote the strategy with $NumIteration[i] = k$ for all $i = 0, 1, ..., \frac{MaxStep}{k} - 1$ as A_k, the strategy with $NumIteration[]$={1, 2, 5, 10, 20, 50, 100, 200, 500} as B, $NumIteration[]$={1, 1, 2, 2, 5, 5, 10, 10, 20, 20, 50, 50, 100, 100, 200, 200} as C. We can see that the CPU-GPU transfer costs a lot of time in finer-grained segmentation strategies, such as $A_1 \sim A_{10}$. When there are more steps in each segment, GPU kernel time generally becomes longer, because more GPU hardware recourses are wasted due to the imbalance load. $A_{MaxStep}$ represents that there is only one segment. Our proposed increasing-interval strategies (Strategy B and C) achieve the best performance. In these two strategies, both GPU kernel time and data transfer time are relatively short.

Table IV
THE COMPARISON OF DIFFERENT SEGMENTATION STRATEGIES (UNIT: SECOND)

Strategy	Kernel time	Reduction time	Transfer time	Total time
A_1	9.16	8.21	41.21	58.6
A_2	7.84	4.18	21.14	33.3
A_5	6.91	3.78	11.35	22.0
A_{10}	7.81	3.29	7.86	19.0
A_{20}	9.46	2.37	5.17	17.0
A_{50}	14.42	1.65	2.27	18.3
A_{100}	23.27	1.52	1.62	26.4
A_{200}	39.45	1.63	1.14	42.2
$A_{MaxStep}$	58.52	0	0	58.5
B	7.06	3.33	4.09	14.5
C	6.55	3.38	4.73	14.7

C. Biological results

In this section we demonstrate some examples of the tracking results. All results are generated from Dataset 2, and we select several long fibers. The A, B and C in Fig.9 shows part of the reconstructed corpus callosum, which is the largest white matter structure in the human brain, and connects the left and right cerebral hemispheres. These results are in accordance with the results of previous neuroscience studies [28], shown in Fig.10. Fig.11 and Fig.12 shows the final tracking result. CPU and GPU results are substantially the same.

VI. DISCUSSION AND CONCLUSION

We present a GPU-based Bayesian framework for probabilistic brain fiber tractography. We accelerate the two main steps: MCMC sampling and probabilistic fiber tracking by 34x and 50x, respectively. We find that the reconstructed fiber lengths are exponentially distributed, and propose a novel strategy for segmenting the tracking process to improve loadbalance while maintaining low communication overhead. The reconstructed fiber pathways are in accordance with previous studies. An open biological question is whether the observed exponential distribution of predicted fiber lengths has some implications for real fibers, or merely

an artifact of the modeling procedure. More data set should be involved to investigate this problem.

Our GPU-based framework has considerable scalability, since the communication of parallel threads is negligible. Little adaptation is needed to extend the current implementation to the multi-GPU version, and proportional performance gains can be expected. The great speedups are partly because this application involves massive data-parallel computation, which is favorable for GPU architecture.

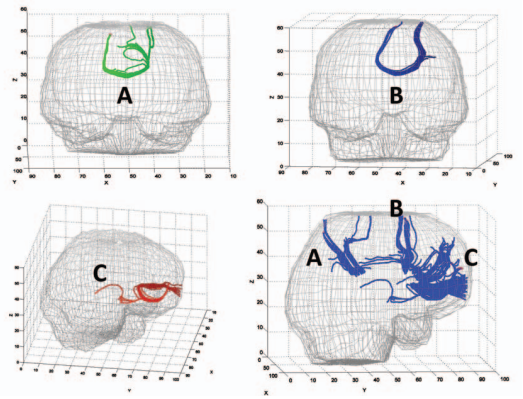


Figure 9. Reconstructed Fiber Pathways: corpus callosum

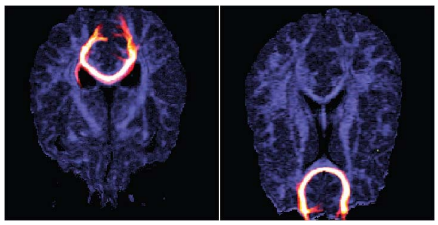


Figure 10. Biological results from similar studies for reference [28]

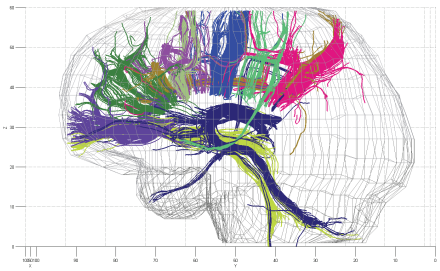


Figure 11. Result: Fibers whose length $>$ 100, from dataset 2.

VII. ACKNOWLEDGEMENT

This work is supported by Microsoft Research Asia and AMD China University Program. This work is also partially supported by National Natural Science Foundation of China (No.60870001). Thanks to the reviewers of IPDPS and IPDPS Workshop on HiCOMB for their precious suggestions.

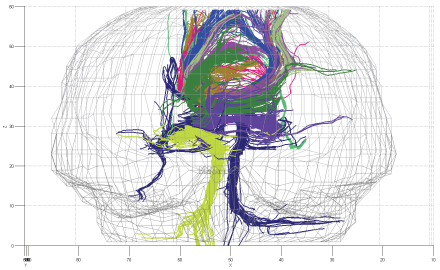


Figure 12. Result: Fibers whose length $>$ 100, from dataset 2.

REFERENCES

[1] H. Johansen-Berg and M. Rushworth, "Using diffusion imaging to study human connectional anatomy," *Annual review of neuroscience*, vol. 32, pp. 75–94, 2009.

[2] S. Mori, B. Crain, V. Chacko, and P. Van Zijl, "Three-dimensional tracking of axonal projections in the brain by magnetic resonance imaging," *Annals of neurology*, vol. 45, no. 2, pp. 265–269, 1999.

[3] T. Conturo, N. Lori, T. Cull, E. Akbudak, A. Snyder, J. Shimony, R. McKinstry, H. Burton, and M. Raichle, "Tracking neuronal fiber pathways in the living human brain," *Proceedings of the National Academy of Sciences*, vol. 96, no. 18, p. 10422, 1999.

[4] D. Jones, S. Williams, D. Gasston, M. Horsfield, A. Simmons, and R. Howard, "Isotropic resolution diffusion tensor imaging with whole brain acquisition in a clinically acceptable time," *Human brain mapping*, vol. 15, no. 4, pp. 216–230, 2002.

[5] P. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In vivo fiber tractography using dt-mri data," *Magnetic Resonance in Medicine*, vol. 44, no. 4, pp. 625–632, 2000.

[6] C. van Pul, J. Buijs, A. Vilanova, F. Roos, and P. Wijn, "Infants with perinatal hypoxic ischemia: Feasibility of fiber tracking at birth and 3 months1," *Radiology*, vol. 240, no. 1, p. 203, 2006.

[7] J. Klein, A. Grötsch, D. Betz, S. Barbieri, O. Friman, B. Stieltjes, H. Hildebrandt, and H. Hahn, "Qualitative and quantitative analysis of probabilistic and deterministic fiber tracking," *Medical Imaging 2010: Image Processing*, vol. 7623, pp. 76232A–1, 2010.

[8] E. van Aart, A. Jalba, A. Vilanova, and B. Mesman, "Acceleration of a geodesic fiber-tracking algorithm for diffusion tensor imaging using cuda," 2010.

[9] G. Parker, H. Haroon, and C. Wheeler-Kingshott, "A framework for a streamline-based probabilistic index of connectivity (pico) using a structural interpretation of mri diffusion measurements," *Journal of Magnetic Resonance Imaging*, vol. 18, no. 2, pp. 242–254, 2003.

[10] P. Hagmann, J. Thiran, L. Jonasson, P. Vandergheynst, S. Clarke, P. Maeder, and R. Meuli, "Dti mapping of human brain connectivity: statistical fibre tracking and virtual dissection," *Neuroimage*, vol. 19, no. 3, pp. 545–554, 2003.

[11] O. Friman, G. Farneback, and C. Westin, "A bayesian approach for stochastic white matter tractography," *Medical Imaging, IEEE Transactions on*, vol. 25, no. 8, pp. 965–978, 2006.

[12] T. Behrens, M. Woolrich, M. Jenkinson, H. Johansen-Berg, R. Nunes, S. Clare, P. Matthews, J. Brady, and S. Smith, "Characterization and propagation of uncertainty in diffusion-weighted mr imaging," *Magnetic Resonance in Medicine*, vol. 50, no. 5, pp. 1077–1088, 2003.

[13] T. Behrens, H. Berg, S. Jbabdi, M. Rushworth, and M. Woolrich, "Probabilistic diffusion tractography with multiple fibre orientations: What can we gain?," *Neuroimage*, vol. 34, no. 1, pp. 144–155, 2007.

[14] S. Smith, M. Jenkinson, M. Woolrich, C. Beckmann, T. Behrens, H. Johansen-Berg, P. Bannister, M. De Luca, I. Drobnjak, D. Flitney, *et al.*, "Advances in functional and structural mr image analysis and implementation as fsl," *Neuroimage*, vol. 23, pp. S208–S219, 2004.

[15] M. Woolrich, S. Jbabdi, B. Patenaude, M. Chappell, S. Makni, T. Behrens, C. Beckmann, M. Jenkinson, and S. Smith, "Bayesian analysis of neuroimaging data in fsl," *Neuroimage*, vol. 45, no. 1, pp. S173–S186, 2009.

[16] J. Nickolls and W. Dally, "The gpu computing era," *Micro, IEEE*, vol. 30, no. 2, pp. 56–69, 2010.

[17] T. Peeters, A. Vilanova, and R. ter Haar Romeny, "Visualization of dti fibers using hair-rendering techniques," in *Proc ASCI*, pp. 66–73, 2006.

[18] M. Hlawitschka, S. Eichelbaum, and G. Scheuermann, "Fast and memory efficient gpu-based rendering of tensor data," *IADIS Computer Graphics and Visualization*, 2008.

[19] P. Kondratieva, J. Kruger, and R. Westermann, "The application of gpu particle tracing to diffusion tensor field visualization," in *Visualization, 2005. VIS 05. IEEE*, pp. 73–78, IEEE, 2005.

[20] G. Reina, K. Bidmon, F. Enders, P. Hastreiter, and T. Ertl, "Gpu-based hyperstreamlines for diffusion tensor imaging," in *Proceedings of EUROGRAPHICS-IEEE VGTC Symposium on Visualization 2006*, pp. 35–42, Citeseer, 2006.

[21] V. Petrovic, J. Fallon, and F. Kuester, "Visualizing whole-brain dti tractography with gpu-based tuboids and lod management," *IEEE transactions on visualization and computer graphics*, pp. 1488–1495, 2007.

[22] A. Köhn, J. Klein, F. Weiler, and H. Peitgen, "A gpu-based fiber tracking framework using geometry shaders," *Medical Imaging 2009: Visualization, Image-Guided Procedures, and Modeling*, vol. 7261, no. 1, p. 72611J, 2009.

[23] A. Mittmann, T. Nobrega, E. Comunello, D. Carvalho, and A. von Wangenheim, "Enabling interactive brain fiber tracking with the gpu," *XXI Brazilian Symposium on Computer Graphics and Image Processing*, 2008.

[24] W. Jeong, P. Fletcher, R. Tao, and R. Whitaker, "Interactive visualization of volumetric white matter connectivity in dt-mri using a parallel-hardware hamilton-jacobi solver," *IEEE transactions on visualization and computer graphics*, pp. 1480–1487, 2007.

[25] A. Mittmann, M. Dantas, and A. von Wangenheim, "Design and implementation of brain fiber tracking for gpus and pc clusters," in *21st International Symposium on Computer Architecture and High Performance Computing*, pp. 101–108, IEEE, 2009.

[26] A. Mittmann, E. Comunello, and A. von Wangenheim, "Diffusion tensor fiber tracking on graphics processing units," *Computerized Medical Imaging and Graphics*, vol. 32, no. 7, pp. 521–530, 2008.

[27] O. Friman and C. Westin, "Uncertainty in white matter fiber tractography," *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2005*, pp. 107–114, 2005.

[28] T. McGraw and M. Nadar, "Stochastic dt-mri connectivity mapping on the gpu," *IEEE transactions on visualization and computer graphics*, pp. 1504–1511, 2007.

[29] H. Zhou, K. Lange, and M. Suchard, "Graphics processing units and high-dimensional optimization," *Statistical Science*, vol. 25, no. 3, pp. 311–324, 2010.

[30] J. Niemi and M. Wheeler, "Efficient bayesian inference in stochastic chemical kinetic models using graphical processing units," *Arxiv preprint arXiv:1101.4242*, 2011.

[31] K. Miyazato, A. Kimura, S. Takagi, and J. Yamato, "Real-time estimation of human visual attention with dynamic bayesian network and mcmc-based particle filter," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pp. 250–257, IEEE, 2009.

[32] M. Linderman, R. Bruggner, V. Athalye, T. Meng, N. Bani Asadi, and G. Nolan, "High-throughput bayesian network learning using heterogeneous multicore computers," in *Proceedings of the 24th ACM International Conference on Supercomputing*, pp. 95–104, ACM, 2010.

[33] M. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West, "Understanding gpu programming for statistical computation: Studies in massively parallel massive mixtures," *Journal of Computational and Graphical Statistics*, vol. 19, no. 2, pp. 419–438, 2010.

[34] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *American Statistician*, pp. 327–335, 1995.

[35] B. Walsh, "Markov chain monte carlo and gibbs sampling," 2004.

[36] G. Box and M. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 1958.

[37] H. Nguyen, *Gpu gems 3*. Addison-Wesley Professional, 2007.