Interactive segmentation of white-matter fibers using a multi-subject atlas

Nicole Labra [1,2], Miguel Figueroa[1,2], Pamela Guevara[1]
Delphine Duclap [3], Josselin Houenou [3,4], Cyril Poupon [3] and Jean-François Mangin [3]

[1]Dept. of Electrical Engineering, Universidad de Concepción, Concepción, Chile
[2] Center for Optics & Photonics, Universidad de Concepción, Concepción, Chile
[3] I2BM, Neurospin, CEA, Gif-sur-Yvette, France
[4] INSERM, U955 Unit, Paris, France

***Abstract*— We present a fast algorithm for automatic segmentation of white matter fibers from tractography datasets based on a multi-subject bundle atlas. We describe a sequential version of the algorithm that runs on a desktop computer CPU, as well as a highly parallel version that uses a Graphics Processing Unit (GPU) as an accelerator. Our sequential implementation runs 270 times faster than a C++/Python implementation of a previous algorithm based on the same segmentation method, and 21 times faster than a highly optimized C version of the same previous algorithm. Our parallelized implementation exploits the multiple computation units and memory hierarchy of the GPU to further speed up the algorithm by a factor of 30 with respect to our sequential code. As a result, the time to segment a subject dataset of 800,000 fibers is reduced from more than 2.5 hours in the Python/C++ code, to less than one second in the GPU version.**

I. INTRODUCTION

The study of white matter (WM) organization and structure is an important research area in the neuroscience study of normal brain development, as well as neurological and psychiatric disorders. Studies of different types of disorders have been performed using this kind of approach [12].

Diffusion-weighted Magnetic Resonance Imaging (dMRI) [8] techniques are the most used for the analysis of human brain white matter *in vivo*. Several methods have been proposed for the analysis of this kind of data. Some methods can be performed by manual expert regions of interest (ROI) placement [1], others use unsupervised clustering [5], [2], and an increasing collection of methods apply hybrid techniques by combining learning or classification algorithms with anatomical information [10], [9], [13]. In [4], an automatic WM segmentation based on a multi-subject WM bundle atlas method is proposed. This method produces good classification results because the atlas embeds information about fiber shape, length and position information, as well as inter-subject variability.

Some of these methods can deal with the new generation of tractography datasets containing a huge number of fibers, but suffer from long processing times. In some applications, useful for neuroscience studies, the user needs to iteratively explore the parameter space, or requires fast multi-subject segmentation in order to find structural similarities between fibers of different subjects. In order to satisfy this need for medium and large tractography datasets (50,000 to 2,000,000 fibers), we can use faster and more direct algorithms by exploiting the high-performance computing capabilities of specialized parallel hardware.

In this work, we describe a new fast algorithm for automatic WM segmentation using the method proposed in [4]. In previous work [7], we parallelized the original classification algorithm using the CUDA language and run it on a GPU. By exploiting the memory hierarchy and fine-grained parallelism of the GPU, we achieved high performance at an affordable cost [6], [11]. Our results made segmentation and visualization of WM tracts more practical, albeit still too slow for interactive applications (35 seconds on a GPU for a subject dataset of 800,000 fibers). The algorithm presented in this paper uses a preprocessing stage that can quickly discard more than 80% of the fibers in the subject dataset before running the original classification algorithm, without changing the final results. The new algorithm classifies 800,000 fibers using the entire atlas in less than one second, and in less than 350ms when only one bundle is considered.

II. MATERIAL AND METHODS

A. Diffusion and tractography datasets

For this analysis, we used healthy subjects from a HARDI database. Acquisitions were obtained using a Siemens Magnetom TrioTim 3T MRI scanner, 12-channel head-coil. The protocol included a high-resolution T1-weighted acquisition (echo time (ET) 2.98ms, repetition time (RT) 2300ms, 160 sagittal slices, 1.0x1.0x1.1mm) and a DW-EPI sequence along 41 directions (2.0x2.0x2.0mm, b=1000s/mm^2, plus one b=0 image, ET 87ms, RT 14000ms, 60 axial slices). The data were processed using the BrainVISA/Connectomist-2.0 tool [3]. They were preliminary corrected for all sources of artifacts, and outliers were removed. A preprocessing step normalizes the tractography dataset to the bundle atlas space (Talairach space) using an affine transformation and resamples each fiber using 21 equally-spaced points.

B. Multi-subject atlas

We use the multi-subject atlas proposed in [4], obtained from a HARDI database of 12 subjects, where inter-subject clusters were manually labeled to identify known WM tracts.

978-1-4244-7929-0/14/$26.00 ©2014 IEEE

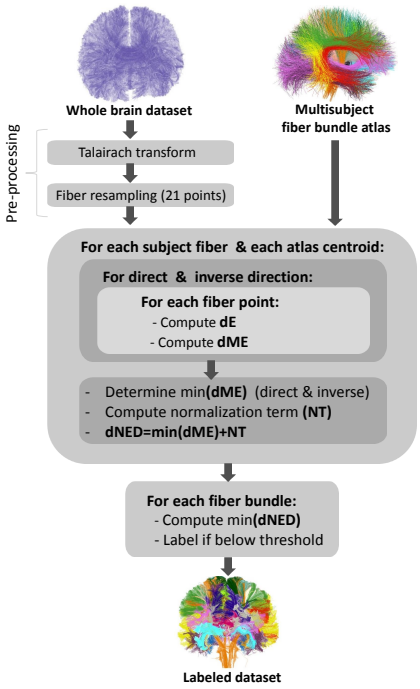

Fig. 1. White matter segmentation algorithm based on a multi-subject bundle atlas. First, the algorithm computes the Euclidean distance (dE) between corresponding points of each subject fiber and atlas centroid and selects the maximum value (dME). It traverses the fibers in both directions and selects the minimum distance. A normalization term (NT) is added to each distance to obtain the normalized Euclidean distance ($dNED$). Finally, fibers are labeled by comparing $dNED$ to a per-bundle threshold.

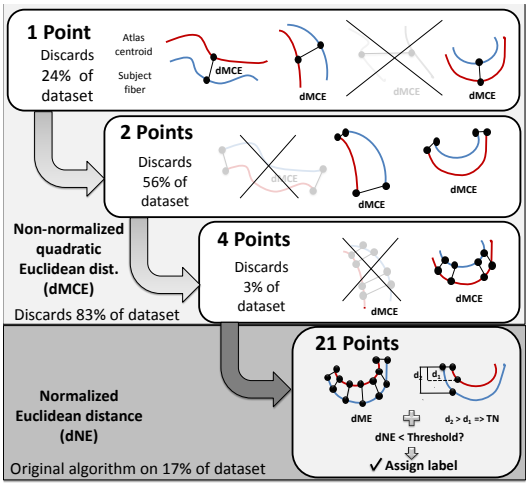

Fig. 2. Interactive WM segmentation algorithm based on a simplified distance tree. Most fibers that do not belong to any bundle are quickly discarded by a preclassification stage using a subset of points and a simplified distance metric. Preclassified fibers are then processed using the original algorithm.

The atlas represents the variability of the bundle shape and position across subjects. We use 26 of the deep WM bundles (**LNAO-DWM12**): arcuate fasciculus, corpus callosum, cigulum fascicles, fornix, uncinate, inferior fronto-occipital, inferior longitudinal and corticospinal tracts, for a total of 9,085 centroids.

C. WM segmentation based on a multi-subject bundle atlas

Guevara et al. [4] proposed an automatic WM segmentation method based on the multi-subject WM bundle atlas described in Figure 1. A similarity measure termed *normalized Euclidean distance* ($dNED$) between each fiber of the tractography dataset and each centroid of the multi-subject atlas is used for classifying fibers into known bundles. The algorithm first computes the regular Euclidean distance between each subject fiber and each atlas centroid. Because the spatial orientation of fibers on the dataset is unknown, both possible orientations are considered: direct and inverse direction. The distance is then normalized by adding a extra term to penalize the length difference between the fiber and the centroid, and the minimum between the direct and inverse distances $dNED$ is obtained for each fiber-centroid pair and compared with a precomputed threshold for each bundle. If $dNED$ is lower than the threshold, the fiber is classified as belonging to that bundle.

The algorithm is flexible and robust, and can be adapted to different anatomical reconstructions. However, it yields excessively long processing times for large tractography datasets. Indeed, the original implementation written in C++ (for data processing) and Python (for file handling), takes more than 2.5 hours to segment a subject dataset of 800,000 fibers. As a first step towards reducing the execution time, profiled the algorithm and rewrote it from scratch in C, applying advanced compiler optimizations such as loop unrolling and scheduling, software pipelining, and reordering of floating-point operations. The resulting code classifies the same 800,000-fiber dataset in approximately 11 minutes. We use this optimized C implementation as a reference to evaluate our new interactive algorithm.

D. New interactive WM segmentation algorithm

Figure 2 shows the new interactive algorithm, which is divided into two steps: preclassification and classification. The original method works using 21 3D points per subject fiber and atlas centroid, and spends most of its time computing the normalized Euclidean distance between each fiber-bundle pair. The preclassification stage quickly discards fibers that do not belong to a bundle by using a simplified non-normalized quadratic Euclidean distance $dMCE$ that uses only a subset of the 21 points that represent each fiber and omits the normalization term. Each step of the preprocessing stage compares this distance to the square of each bundle threshold TH^2 and selects only fibers for which $dMCE < TH^2$. It can be shown that $dNED^2 > dMCE$, therefore the simplified metric will not discard a fiber that the original metric would classify as belonging to a bundle.

Our algorithm repeats the preclassification step on the fibers selected by the previous step, doubling the number of points per fiber in each run, until the number of discarded fibers is negligible. We empirically chose the number of points used in each preclassification step by assessing the execution time with several subset combinations. The best

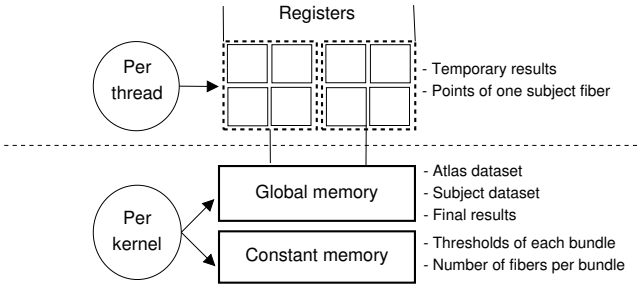

Fig. 3. Distribution of data in GPU memory. Our implementation uses registers to store temporary results and a fiber per thread. Global memory is used to store datasets and results, and constant memory is used to store thresholds and constant values.

results were achieved when using the center point in the first step to quickly discard fibers that are located far from the centroid. The second and third steps differentiate fiber shape by using the two extremes and 4 intermediate points, respectively. The final classification stage segments the remaining fibers using the original distance metric and algorithm.

In our experiments, the preclassification stage discards 83% of the fibers in the dataset. Thus, the original algorithm runs on only 17% of the data, and assigns a label to 14% of the fibers on average. The RAM available to the CPU can only accommodate 200,000 fibers simultaneously, therefore larger datasets must be processed in blocks.

E. Parallel implementation on a GPU

In order to further reduce the execution time, we wrote a parallelized version of the new algorithm using the CUDA language and run it on an NVIDIA GPU. A GPU is a single-instruction multiple-data (SIMD) processor composed of hundreds of cores, which are much simpler than traditional CPU cores. CUDA programs are composed of kernels, which are functions that execute multiple identical threads. All threads in a CUDA kernel run in parallel on multiple cores, by executing the same instruction on multiple data.

Our application consists of a main kernel, which concentrates most of the computationally-intensive code. The kernel spawns as many threads as there are fibers in the subject dataset. The program starts by allocating global memory on the video card that hosts the GPU, leaving some extra space available for final results. Then, the CPU transfers the entire atlas and subject datasets to the global memory and invokes the kernel on the GPU.

The NVIDIA GPU architecture features three levels of memory hierarchy: registers and local memory that can be privately accessed by each thread being executed on the GPU; shared memory, which is accessed by all threads belonging to the same thread block; and constant memory, texture memory and global memory, which are available for all threads on the GPU. Our implementation carefully places the data within the hierarchy depending on how the algorithm uses it. Figure 3 shows the data placement, which uses local registers, constant memory, and global memory.

Global memory is flexible and large (6GB in our video card), but it has a high latency and its bandwidth is limited. We use it as an intermediate stage to transfer the atlas, fiber dataset, and computation results between the CPU and the GPU. Because the atlas contains only 9,085 centroids, the entire set fits in the global memory.

Registers offer local thread storage with high aggregated bandwidth and low latency. Because we assign one thread to each fiber, we use registers to store each fiber point, as well as temporary results such as intermediate sums, classification labels per stage, counters, Euclidean distance between points within a fiber, and the normalization term. Data exists in the memory for the duration of the thread, which is the lifetime of the kernel.

Constant memory is globally accessible, but it is smaller than global memory (only 64KB) and cannot be modified. It has very low latency and reduces the bandwidth requirements when multiple threads need to access the same memory location simultaneously. We use this memory to store parameters such as the threshold values and the number of centroids for each bundle in the atlas, which are constants and are used by many threads at the same time.

The threads perform the preclassification and classification stages for each subject fiber. On each preclassification step, each thread computes $dMCE$ between one fiber and all the centroids that belong to each atlas bundle. If a fiber is discarded with respect to a centroid, the thread immediately starts evaluating another centroid. If a fiber is preclassified, the evaluation continues in the subsequent steps. If a fiber passes all preclassification steps of a centroid, the thread runs the original algorithm to compute $dNED$ and determine whether it belongs to the respective bundle. Finally, the classification results are transferred back to the CPU memory.

III. RESULTS

We tested our software on an Intel Core i7-3820 CPU (3.6GHz, 8GB) running Ubuntu 12.04 and equipped with an NVIDIA Quadro 6000 GPU with 6GB of GDDR5 RAM and 448 CUDA cores. The parallel version was written using the CUDA Toolkit 4.0. Our results were validated using a real subject tractography dataset. All versions of the algorithm produce the same results as the reference C implementation.

The graph in Figure 4 compares the execution time of the original algorithm in Python/C++, the optimized C version, the new algorithm in C, and the parallel version in CUDA, for dataset sizes ranging from 9,500 to 1,600,000 fibers.

For medium and large datasets of more than 100,000 fibers, our new sequential algorithm runs 21 times faster than the optimized C version of the original algorithm, using the same hardware. In turn, the parallel algorithm runs 30 times faster than our sequential version and 638 times faster than the original C, thus drastically improving the execution time. The original algorithm can segment a subject dataset of 800,000 fibers in 2.5 hours with the Python/C++ implementation, and in 11 minutes with the optimized C version. Our new algorithm runs in 30s using only the CPU, and in just 970ms using the GPU. These drastically reduced

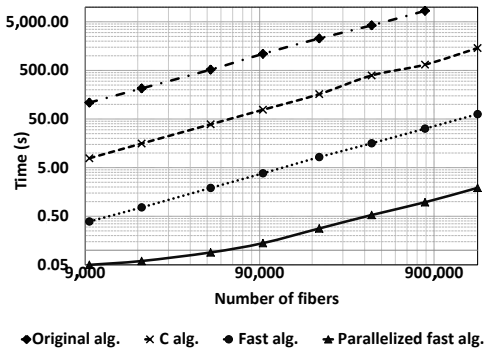

Fig. 4. Execution time of the algorithms for different datasets.

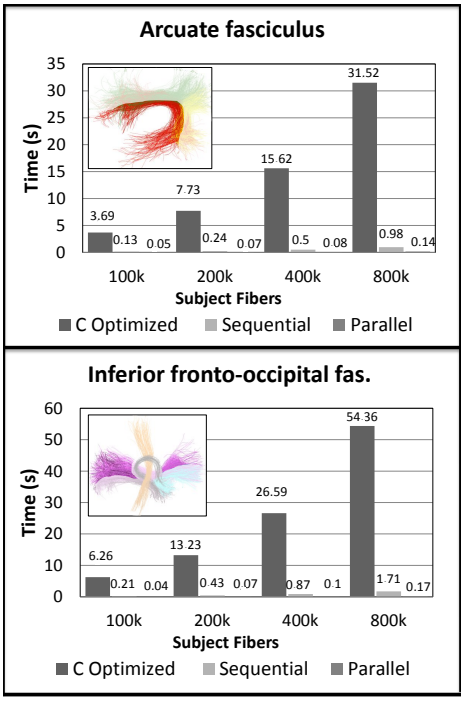

Fig. 5. Execution time of sequential and parallel fast algorithm for individual bundles.

execution times enable true interactive use of segmentation and visualization for tractography datasets of this size.

In order to explore typical use of segmentation in interactive visualization tools, we evaluated the time to run the algorithm on the entire dataset using a single atlas bundle. In this case, the algorithm only decides whether each fiber belongs to the selected bundle. Figure 5 shows the results for 2 representative bundles with the two implementations of our algorithm, and the original optimized C version. Both implementations of the new algorithm greatly improve the execution time. In fact, even on a CPU our algorithm segments an 800,000-fiber dataset in under 1.7s, while the original C uses 54s. The execution time on the GPU is below 170ms. Because of the time used to exchange data between CPU and GPU, the speedup achieved by the parallel version is smaller than in the segmentation with the entire atlas.

IV. CONCLUSIONS

We have developed a fast automatic segmentation algorithm for classification of white matter fibers. Even a sequential version of our algorithm running on a CPU reduces the original segmentation time by a factor of 21 compared to a highly-optimized implementation of the original method. A parallel implementation running on a GPU achieves an overall reduction in the execution time of a factor of 630. Thus, segmentation of large datasets that previously would take 11 minutes, can now be performed in less than one second. Our sequential and parallel implementations can also perform segmentation on a per-bundle basis in less than 1.7s and 170ms, respectively, for the same number of fibers. These improvements in execution time enable truly interactive visualization, analysis and segmentation of large datasets.

V. ACKNOWLEDGMENTS

This work was partially funded by FONDECYT grants 112101 and 11121644, and PIA-CONICYT PFB0824. Thanks to Marion Leboyer for providing the testing HARDI database.

REFERENCES

[1] M. Catani, R. J. Howard, S. Pajevic, and D. K. Jones. Virtual in vivo interactive dissection of white matter fasciculi in the human brain. *Neuroimage*, 17(1):77–94, Sep 2002.

[2] L. Dodero, S. Vascon, L. Giancardo, A. Gozzi, D. Sona, and V. Murino. Automatic white matter fiber clustering using dominant sets. In *Pattern Recognition in Neuroimaging (PRNI), 2013 International Workshop on*, pages 216–219, June 2013.

[3] D. Duclap, A. Lebois, B. Schmitt, O. Riff, P. Guevara, L. Marrakchi-Kacem, V. Brion, F. Poupon, J.-F-Mangin, and C. Poupon. Connectomist-2.0: a novel diffusion analysis toolbox for BrainVISA. In *ESMRMB 2012*, 2012.

[4] P. Guevara, D. Duclap, C. Poupon, L. Marrakchi-Kacem, P. Fillard, D. Lebihan, M. Leboyer, J. Houenou, and J-F. Mangin. Automatic fiber bundle segmentation in massive tractography datasets using a multi-subject bundle atlas. *Neuroimage*, 61(4):1083–1099, Jul 2012.

[5] P. Guevara, C. Poupon, D. Riviére, Y. Cointepas, M. Descoteaux, B. Thirion, and J.-F. Mangin. Robust clustering of massive tractography datasets. *NeuroImage*, 54(3):1975–1993, Feb 2011.

[6] D. B. Kirk and W.-M. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. M. Kaufmann, 2010.

[7] N. Labra, M. Figueroa, P. Guevara, D. Duclap, J. Houenou, F. Poupon, and J-F. Mangin. Gpu-based acceleration of an automatic white matter segmentation algorithm using cuda. In *IEEE Eng Med Biol Soc, EMBS conference*, pages 89–92, 2013.

[8] D. Le Bihan, J. F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabriat. Diffusion tensor imaging: concepts and applications. *J Magn Reson Imaging*, 13(4):534–546, Apr 2001.

[9] H. Li, Z. Xue, L. Guo, T. Liu, J. Hunter, and S.T. Wong. A hybrid approach to automatic clustering of white matter fibers. *Neuroimage*, 49(2):1249–1258, Jan 2010.

[10] L.J. O'Donnell and C.-F. Westin. Automatic tractography segmentation using a high-dimensional white matter atlas. *IEEE Transactions on Medical Imaging*, 26(11):1562–1575, Nov. 2007.

[11] J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley, 2012.

[12] Samuel Sarrazin, Cyril Poupon, Julia Linke, and et al. A multicenter tractography study of deep white matter tracts in bipolar i disorder: Psychotic features and interhemispheric disconnectivity. *JAMA Psychiatry*, Feb 2014.

[13] D. Wassermann, L. Bloy, E. Kanterakis, R. Verma, and R. Deriche. Unsupervised white matter fiber clustering and tract probability map generation: Applications of a gaussian process framework for white matter fibers. *Neuroimage*, 51:228–241, Jan 2010.