Accelerating Fibre Orientation Estimation from Diffusion Weighted Magnetic Resonance Imaging using GPUs

Moisés Hernández, Ginés D. Guerrero, José M. Cecilia and José M. García
Grupo de Arquitectura y Computación Paralela. University of Murcia (Spain)
{moises, gines.guerrero, chema, jmgarcia}@ditec.um.es

Alberto Inuggi
Dpto. Psicología Básica y Metodología. University of Murcia (Spain)
a.inuggi@bcbl.eu

Stamatios N. Sotiropoulos
Oxford Centre for Functional MRI of the Brain
Department of Clinical Neurology. University of Oxford (UK)
stam@fmrib.ox.ac.uk

Abstract

Diffusion Weighted Magnetic Resonance Imaging (DW-MRI) and tractography approaches are the only tools that can be utilized to estimate structural connections between different brain areas, non-invasively and in-vivo. A first step that is commonly utilized in these techniques includes the estimation of the underlying fibre orientations and their uncertainty in each voxel *of the image. A popular method to achieve that is implemented in the FSL software, provided by the FMRIB Centre at University of Oxford, and is based on a Bayesian inference framework. Despite its popularity, the approach has high computational demands, taking normally more than 24 hours for analyzing a single subject. In this paper, we present a GPU-optimized version of the FSL tool that estimates fibre orientations. We report up to 85x of speed-up factor between the GPU and its sequential counterpart CPU-based version.*

1. Introduction

The brain is one of the most complex biological systems. It is comprised of billions of inter-connected processing units called *neurons*. Information between *neurons* is transferred by processes called *axons*, which conduct electrical impulses away from neurons' cell bodies. *Axons* are usually bundled together and form coherent structures or fibre tracts, that mediate information flow between different brain regions and are very important for brain's function. Many neurological diseases are associated with the interruption and damage of fibre tracts.

Traditional neuroimaging techniques do not provide enough information about such structural connections between brain regions. However, with the advent of Diffusion-Weighted Magnetic Resonance Imaging (DW-MRI) [11] and variants, such as Diffusion Tensor Imaging (DTI) [2], estimation of fibre tracts has become feasible.

DW-MRI is sensitive to the diffusion motions of water molecules, whose features vary throughout the different brain tissues; white matter, which mostly comprises of *axons*, gray matter, which contains mainly cell bodies and cerebrospinal fluid (CSF)-filled regions. Particularly in white matter, water diffuses preferably along rather than across the *axons* [4]. By applying strong magnetic field gradients in several and different directions, it is possible to map these preferred diffusion orientations (PDOs) in each image volume element (or *voxel*). PDOs are commonly assumed to give local fibre orientation estimates, i.e. the major axon orientations within an image *voxel* [15]. Using tractography approaches, PDOs can then be utilized to reconstruct the underlying fibre tracts [3]. Tractography is currently the only tool that allows the study of structural connections in the brain, non-invasively and in-vivo [10].

Various model-based and model-free approaches have been suggested to estimate the PDOs [17]. A popular approach that also takes care of within-voxel fibre crossings, a common problem in tractography, is the ball & stick model [5, 6]. The approach has been implemented in the FSL software (developed by the FMRIB centre at the Uni-

versity of Oxford) [18] and the respective tool is called *Bedpost*. *Bedpost* application estimates the PDOs, as well as their uncertainty given the data, using a Bayesian inference framework. Despite its popularity, its main drawback is the long computation time, since depending on the parameters of the MRI sequence, the analysis of a single subject can take more than 24 hours on a CPU.

Modern Graphics Processing Units (GPUs) are massively parallel processors that are capable of supporting thousands of threads running in parallel, reaching theoretical peak performance up to a TeraFLOP (10^{12} Floating Point operations per second). Many general purpose applications have been successfully ported to these platforms, obtaining considerable accelerations [7, 8].

In this paper, we discuss the parallelization of the *Bedpost* application on NVIDIA GPUs by using the CUDA programming model [12]. We identify two main stages in this application that are studied separately. The first is the *Levenberg-Marquardt* algorithm which is designed by using a data-parallelism approach. The second is the *Markov Chain Monte Carlo (MCMC)* algorithm in which several techniques are proposed to increase the parallelism, and also avoiding long-stall warp serialization. These implementations provide us a speed-up factors of 127x and 56x for the *Levenberg-Marquardt* and the *MCMC* algorithms respectively, and an overall speed-up factor of the *Bedpost* application reaching up to 85x compared to the sequential counterpart version.

2. Diffusion-Weighted MRI and the ball & stick model

Water molecules are in constant random motion known as diffusion, which is solely driven by thermal energy. Diffusion in the brain is interesting because the presence of tissue microstructures hinders these motions. In white matter, this hindrance is systematic, due to the presence of axonal membranes and myelin sheaths [4] and there is a preference for diffusion towards a particular direction along the main axon orientation. Such a diffusion profile is called anisotropic. On the contrary, there is no systematic structure in gray matter or CSF-filled regions. Diffusion appears to be equally likely towards all directions and the profile is called isotropic. In the anisotropic regions, the Preferred Diffusion Orientation (PDO) has a biophysical meaning and provides an estimate for the main orientation of the underlying *axons*. These estimates are utilized by tractography approaches to reconstruct long-range fibre tracts.

To estimate the PDOs a set of diffusion-weighted magnetic resonance images are needed. Each DW image has a contrast that depends on diffusion motions along a specific direction, the direction of an applied diffusion-sensitizing magnetic field gradient [19]. Many DW images are commonly acquired along K different directions to effectively sample the signal on a unit sphere domain. The ball & stick model explains this signal in each *voxel* of the brain volume, using a multi-compartment decomposition. It assumes a fully isotropic compartment (the ball) and $L \geq 1$ perfectly anisotropic compartments (the sticks). The orientations of these sticks provide the PDOs in a *voxel*.

Equation 1 shows the signal model when each of the $k = 1 : K$ gradient directions is applied [5, 6]:

$$S_k = S_0[(1 - \sum_{j=1}^{L} f_j) \exp(-b_k d) + \sum_{j=1}^{L} f_j \exp(-b_k d (g_k^T v_j)^2)] + e \quad (1)$$

S_0 is a baseline signal measured without any diffusion weighting; b_k depends on the magnitude and duration of the k^{th} diffusion-sensitizing gradient, g_k indicates the direction of this gradient, d is the diffusivity, and finally $f_j \in [0, 1]$ and v_j describe the volume fraction and orientation of the j^{th} stick, with:

$$v_j = \left[\begin{array}{ccc} \sin(\theta_j)\cos(\varphi_j) & \sin(\theta_j)\sin(\varphi_j) & \cos(\theta_j) \end{array} \right]^T \quad (2)$$

and $\theta_j \in [0, \pi]$ and $\varphi_j \in [0, 2\pi]$. Equation 1 also contains an error term e that represents noise and can be assumed to follow a zero-mean Normal distribution.

Bedpost application inverts the above model using a Bayesian framework. Equation 1 can be used to obtain the likelihood, the conditional distribution of the data S_k given the model parameters. Bayes theorem allows us to calculate the posterior probability of the parameters given the data $P(\textit{parameters} \mid \textit{data})$ [5]. Thus a distribution is estimated for each of the model parameters S_0, d, θ_j, φ_j and f_j, $j = 1 : L$. This is performed using a *Markov Chain Monte Carlo (MCMC)* algorithm [1], which is initialized using a *Levenberg-Marquardt* fit of the model to the data [16].

3. Description of the Bedpost application

The input of the *Bedpost* application is a brain's image of a given subject. This image is composed of several *slices* that may have different sizes depending on the area of the brain that they represent. Moreover, these slices are composed of several *voxels*, which are processed sequentially on the CPU (see Figure 1).

Algorithm 1 summarizes the sequential steps of this application: (1) An initial estimation of the parameters through *Levenberg-Marquardt* algorithm, and (2) the estimation of the posterior distribution of the model parameters given the data through a *Markov Chain Monte Carlo (MCMC)* algorithm.

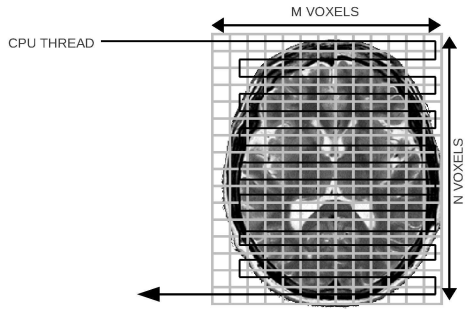


Figure 1. A sequential thread computing $M * N$ *voxels* of a *slice*.

Algorithm 1 The sequential pseudo-code of `Bedpost`

```
1: for all slice do
2:   for all voxel do
3:     LevenbergMarquardt()
4:     MCMC()
5:   end for
6: end for
```

The *Levenberg-Marquardt* algorithm is based on an iterative optimization procedure that minimizes the sum of squared model residuals. This method applies several matrix operations, with the inverse matrix operation being the most time consuming part. The matrices have a size that depends on the number of parameters to be estimated.

The *MCMC* algorithm takes as input the values calculated in the *Levenberg-Marquardt* step. The algorithm then proposes in an iterative fashion random values for each parameter to be estimated, drawn from Normal proposal distributions. Actually, two random numbers are generated for each parameter at each iteration of the *MCMC* algorithm, one drawn from a uniform and one from a normal distribution. It is worth noticing the computational cost of this random generation as hundreds of millions of random numbers need to be generated. The proposed parameter values are accepted or rejected depending on a Metropolis acceptance criterion that utilizes the respective posterior probability values.

4. Parallel design of the Bedpost application in CUDA

A first alternative for a parallel design of *Bedpost* application is motivated by the parallel nature of processing all *slices*. This is the philosophy followed by the FSL developers [18]. They run each *slice* independently by using several different processors on a large cluster using the SunGridEngine software application [9]. This design, however, is based on task parallelism, and thus it involves heavy tasks assigned to each processor. The task-based parallelism is not theoretically well-suited for the GPU programming, where a data-based approach can lead to better performance, taking advantage of the thousands of light-weight threads that can run in parallel (for more details about CUDA programming model refer to [12])

Our data-based parallelism approach for this problem is obtained by thinking about how data can be partitioned. As previously mentioned, each *slice* is composed of many *voxels*, and those *voxels* can be processed independently. Therefore, a CUDA kernel processes all *voxels* within a *slice* in parallel.

Four different kernels run sequentially, with parallelization occuring within each kernel (see Algorithm 2). The first CUDA kernel performs the *Levenberg-Marquardt* algorithm. This kernel follows the design previously commented, mapping each CUDA thread to a *voxel*, and having as many threads as *voxels* contained in a particular *slice*. Each thread estimates the initial values of all model parameters for each *voxel*.

Algorithm 2 CUDA Design of the `Bedpost` application.

```
1: for all slice do
2:   bloqs := voxels/blockSize1
3:   levenbergKernel(bloqs, blockSize1)
4:   genNumsKernel(bloqs, blockSize1)
5:   adjustNumsKernel(bloqs, blockSize1)
6:   bloqs := (voxels * K)/blockSize2
7:   mcmcKernel(bloqs, blockSize2)
8: end for
```

The other three CUDA kernels implement the MCMC step. The first one (*genNumsKernel* in the Algorithm 2) is responsible for the generation of random numbers. This is a critical issue for performance as the algorithm needs to generate many random numbers from a pseudo-random sequence. We decided to pre-calculate those random numbers in an homogenous separate kernel instead of having a coarse-grained kernel with a heterogeneous set of instructions. Random numbers are generated by using the `CURAND` library [14]. After their generation, the pseudo-random numbers are adjusted in the second kernel (*adjustNumsKernel* in the Algorithm 2) to a uniform and normal distribution as global synchronization is required. Both previous kernels follow the design described above.

Finally, the last kernel implements the `MCMC` computation (*mcmcKernel* in the Algorithm 2). The design of this kernel is a bit different from the previous ones. Many signal evaluations must be performed for each voxel, concretely as many evaluations as gradient directions are applied. In order to obtain light-weight threads, we decide that each thread models a predicted signal for a single diffusion-sensitizing direction k. This increases the parallelism by

having $\#voxels \times k$ threads, and thus the work can be distributed among them. Moreover, *voxels* are equally distributed among CUDA thread-blocks, according to the number of gradient directions (see Figure 2).

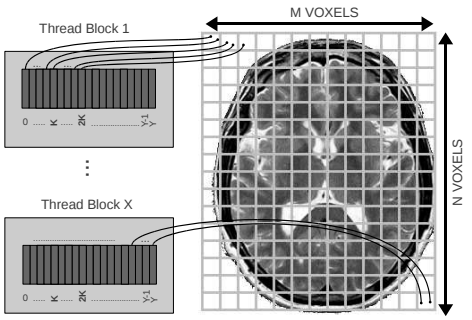


Figure 2. $M*N*K$ threads running in parallel for the processing of $M*N$ *voxels* of a *slice* with K gradient directions, grouped into X blocks of size Y, where Y is a multiple of K.

5. Performance Evaluation

This section evaluates all kernels previously explained for the *Bedpost* application on GPUs. Intel-based host machine provide service to a GPU system based on an Nvidia Tesla C2050 [13]. The main processor of this machine is an Intel Xeon E5620 2.40GHz and 4GB of main memory where the sequential experiments were developed on. Finally, we used for our tests the 4.0 CUDA version and gcc 4.4.3 with option -03 enable.

The input data for the *Bedpost* application, i.e. the number of *slices*, *voxels*, and diffusion gradients is determined by the acquisition protocol during imaging. We vary the number of parameters to be estimated in the range of eight to fourteen.

Figure 3 shows a performance comparison between CPU and GPU when executing Levenberg-Marquardt varying the number of parameters modeled and images size. Figure 3 shows a performance gain proportional to the number of *voxels*, as a major number of threads run concurrently on the GPU. For eight parameters configuration, we report up to 20x speed-up factor, and up to 56x for fourteen parameters compared to the sequential counterpart version.

The generation of pseudorandom number is a critical issue in the performance of the *Bedpost* application. The CPU version uses the `rand()` function that belongs to the C++ library `CSTDLIB` of C++, while the GPU version uses the `CURAND` library from NVIDIA. We report up to 140x speed-up factor when up to 380 million of pseudo-random numbers are generated in parallel.

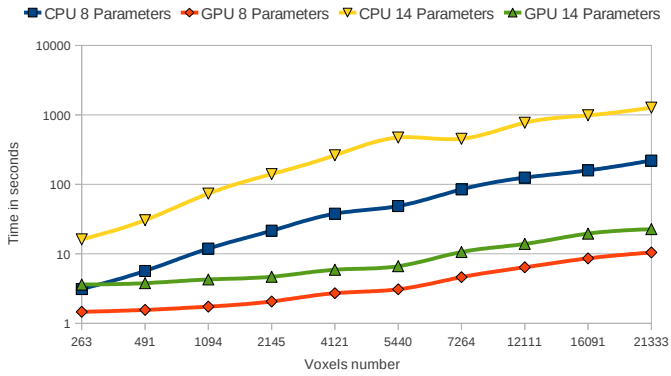


Figure 3. CPU and GPU execution times (in log scale) running Levenberg-Marquardt with 8 and 14 parameters.

Moreover, an extra 14x performance gain is achieved by generating the normal and uniform distribution of pseudo-random numbers on the GPU.

For the *MCMC* algorithm we vary the number of iterations and also the parameters to be modeled by using two and four fibres. In this case, the GPU defeats by a wide margin the CPU reaching up to 127x speed-up factor.

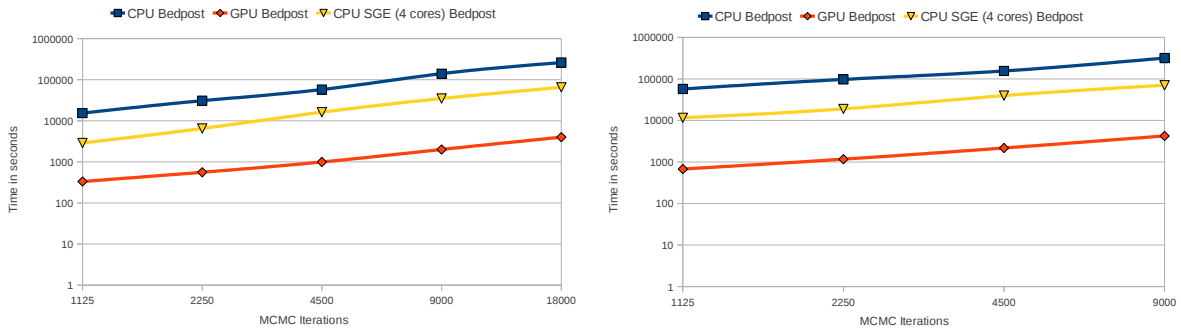
Finally Figure 4 shows the performance evaluation in CPU, CPU with four cores using the SGE software, and in GPU, for the *Bedpost* application with a image dimensions of 128x128x47 with 35 gradient directions, two and four fibres modelled and varying the number of *MCMC* iterations. The maximum speedup obtained is up to 85X comparing a core of CPU with GPU, and 18X comparing four cores of CPU with GPU.

6. Conclusions and Future Work

The brain is one of the most complex biological systems, and a proper understanding of its function is still a challenge in the scientific community. Our role as computer scientist is to provide efficient tools to help experts in these areas. FSL help to understand structural connections between regions within the brain. However the analysis of a single subject may take up to 24 hours.

In this paper, we contribute with the GPU paralelization of the *Bedpost* tool within the FSL framework, which is used to estimate fibre orientations from diffusion-weighted magnetic resonance images. We report an overall speed-up of up to 85x compared to the sequential counterpart version within FSL. This means, in practical terms, that our approach reduces the execution time of *Bedpost* down to 17 minutes for a single subject.

Moreover, given the adequacy of GPUs for the optimization of *Bedpost* application, our next step will be the GPU



(a) 2 Fibres

(b) 4 Fibres

Figure 4. Comparison of total execution time (in log scale) of *Bedpost* application in CPU, CPU with four cores using the SGE software, and in GPU with image dimensions 128x128x47, 35 gradient directions and modeling two and four fibres.

implementation of other tools included in FSL.

Acknowledgements

This research was supported by Fundación Séneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) under grant 00001/CS/2007, and also by the Spanish MEC and European Commission FEDER under grants CSD2006-00046 and TIN2009-14475-C04.

References

[1] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, Jan. 2003.

[2] P. J. Basser, J. Mattiello, and D. Le Bihan. Estimation of the Effective Self-Diffusion Tensor from the NMR Spin Echo. *Journal of Magnetic Resonance Series b*, 103(3):247–254, 1994.

[3] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi. In vivo fiber tractography using DT-MRI data. *Magnetic Resonance in Medicine*, 44(4):625–632, 2000.

[4] C. Beaulieu. The basis of anisotropic water diffusion in the nervous system - a technical review. *NMR in Biomedicine*, 15(7-8):435–455, 2002.

[5] T. E. 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(5):1077–1088, 2003.

[6] T. E. J. Behrens, H. Johansen-Berg, S. Jbabdi, M. F. S. Rushworth, and M. W. Woolrich. Probabilistic diffusion tractography with multiple fibre orientations: What can we gain? *NeuroImage*, 34(1):144–155, 2007.

[7] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov. Parallel Computing Experiences with CUDA. *IEEE Micro*, 28:13–27, July 2008.

[8] W. W. Hwu, editor. *GPU Computing Gems: Emerald Edition*. Morgan Kaufmann, 2011.

[9] Inc. Sun Microsystems. Sun N1 Grid Engine 6.1 User's Guide, 2007.

[10] S. Jbabdi and H. Johansen-Berg. Tractography: Where Do We Go from Here? *Brain Connectivity*, 3:169–183, September 2011.

[11] D. Le Bihan, E. Breton, D. Lallemand, P. Grenier, E. Cabanis, and M. Laval-Jeantet. MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders. *Radiology*, 161(2):401–407, Nov 1986.

[12] NVIDIA. *NVIDIA CUDA C Programming Guide 4.0*. 2011.

[13] NVIDIA Corporation. Whitepaper NVIDIA's Next Generation CUDA Compute Architecture: Fermi, 2009.

[14] NVIDIA Corporation. CUDA Toolkit 4.0 CURAND Guide, 2011.

[15] C. Pierpaoli, P. Jezzard, P. J. Basser, A. Barnett, and G. Di Chiro. Diffusion tensor MR imaging of the human brain. *Radiology*, (201):637–648, 1996.

[16] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 2 edition, 1992.

[17] K. K. Seunarine and D. C. Alexander. Multiple fibers: Beyond the diffusion tensor. In H. Johansen-Berg and T. Behrens, editors, *Diffusion MRI: From quantitative measurement to in vivo neuroanatomy*, pages 55–72. Elsevier, 2009.

[18] S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E. Behrens, H. Johansen-Berg, P. R. Bannister, M. De Luca, I. Drobnjak, D. E. Flitney, R. K. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. De Stefano, J. M. Brady, and P. M. Matthews. Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*, 23:208–219, 2004.

[19] E. O. Stejskal and J. E. Tanner. Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient. *Journal of Chemical Physics*, 42(1), 1965.