



Using GPUs to accelerate computational diffusion MRI: From microstructure estimation to tractography and connectomes

Moises Hernandez-Fernandez^{a,b,*}, Istvan Reguly^c, Saad Jbabdi^a, Mike Giles^d, Stephen Smith^a, Stamatis N. Sotiropoulos^{a,e}

^a Wellcome Centre for Integrative Neuroimaging - Centre for Functional Magnetic Resonance Imaging of the Brain (FMRIB), University of Oxford, Oxford, United Kingdom

^b Center for Biomedical Image Computing and Analytics (CBICA), Department of Radiology, University of Pennsylvania, Philadelphia, PA, United States

^c Faculty of Information Technology and Bionics, Pazmany Peter Catholic University, Budapest, Hungary

^d Mathematical Institute, University of Oxford, Oxford, United Kingdom

^e Sir Peter Mansfield Imaging Centre, School of Medicine, University of Nottingham, Nottingham, United Kingdom

ARTICLE INFO

Keywords:

GPGPU
Scientific computing
Biophysical modelling
Non-linear optimisation
Bayesian inference
Fibre orientations
Fibre dispersion
Brain connectivity
Medical imaging

ABSTRACT

The great potential of computational diffusion MRI (dMRI) relies on indirect inference of tissue microstructure and brain connections, since modelling and tractography frameworks map diffusion measurements to neuroanatomical features. This mapping however can be computationally highly expensive, particularly given the trend of increasing dataset sizes and the complexity in biophysical modelling. Limitations on computing resources can restrict data exploration and methodology development. A step forward is to take advantage of the computational power offered by recent parallel computing architectures, especially Graphics Processing Units (GPUs). GPUs are massive parallel processors that offer trillions of floating point operations per second, and have made possible the solution of computationally-intensive scientific problems that were intractable before. However, they are not inherently suited for all problems. Here, we present two different frameworks for accelerating dMRI computations using GPUs that cover the most typical dMRI applications: a framework for performing biophysical modelling and microstructure estimation, and a second framework for performing tractography and long-range connectivity estimation. The former provides a front-end and automatically generates a GPU executable file from a user-specified biophysical model, allowing accelerated non-linear model fitting in both deterministic and stochastic ways (Bayesian inference). The latter performs probabilistic tractography, can generate whole-brain connectomes and supports new functionality for imposing anatomical constraints, such as inherent consideration of surface meshes (GIFTI files) along with volumetric images. We validate the frameworks against well-established CPU-based implementations and we show that despite the very different challenges for parallelising these problems, a single GPU achieves better performance than 200 CPU cores thanks to our parallel designs.

1. Introduction

General-purpose computing on graphics processing units (GPGPU) has lead to a significant step forward in scientific computations. GPUs are massive parallel processors with thousands of cores. Mainly driven by the computer game industry, and more recently by deep learning applications (Schmidhuber, 2015), GPUs have evolved rapidly in the last decade, offering now over 15 TeraFLOPS (15×10^{13} floating operations per second) in single precision of performance (NVIDIA, 2017). Even if their full potential is not used, their suitability for scientific computing has become more and more evident in projects that involve large

amounts of data. For instance, the 1000 Genomes Project (Auton et al., 2015; Sudmant et al., 2015) and the Human Connectome Project (Van Essen and Ugurbil, 2012; Van Essen et al., 2012; Sotiropoulos et al., 2013) have generated Petabytes of data. The computations performed for the analysis of all this data can take months on typical computer clusters, but GPU accelerated solutions can accelerate massively these computations (Klus et al., 2012; Hernández et al., 2013).

In the field of medical imaging, GPUs have been used in several computational domains (Eklund et al., 2013), including image reconstruction (Stone et al., 2008; Uecker et al., 2015) image segmentation (Smistad et al., 2015; Alsmirat et al., 2017), image registration

* Corresponding author. Wellcome Centre for Integrative Neuroimaging - Centre for Functional Magnetic Resonance Imaging of the Brain (FMRIB), University of Oxford, Oxford, United Kingdom.

E-mail address: moisesf@fmrrib.ox.ac.uk (M. Hernandez-Fernandez).

<https://doi.org/10.1016/j.neuroimage.2018.12.015>

Received 28 June 2018; Received in revised form 20 November 2018; Accepted 7 December 2018

Available online 8 December 2018

1053-8119/Crown Copyright © 2018 Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

(Shamonin, 2014), and in the analysis of functional MRI (Eklund et al., 2014) and diffusion MRI data (Xu et al., 2012; Hernández et al., 2013; Chang et al., 2014; Hernandez-Fernandez et al., 2016; Harms et al., 2017).

However, using GPUs is not always straightforward. The GPU architecture is completely different to the traditional single or multi-core CPU architectures, it is not inherently suited for all types of problems, and bespoke computational frameworks may need to be developed to take advantage of their full potential. Some of the challenges that need to be considered for achieving an efficient design include: balanced parallelisation of an algorithm, good organisation of threads and grouping, appropriate usage of memory resources, appropriate memory access patterns, and correct communication and synchronisation between threads. Furthermore, programming GPUs requires specific programming models that offer control over the device resources, but may increase the difficulty for designing parallel solutions. Low-level programming models, such as the Compute Unified Device Architecture (CUDA) (Nickolls et al., 2008), offer a high degree of control over the resources, and the possibility of achieving very efficient solutions. A more detailed description of the GPU architecture, the CUDA programming model, and some considerations for GPU programming are included in the Supplementary material.

Despite the challenges, in this paper we illustrate the potential of GPUs for two neuroimaging applications spanning different parallelisation strategies. Specifically, we design and implement parallel computational frameworks for analysing diffusion magnetic resonance imaging (dMRI) data (Alexander et al., 2017; Jeurissen et al., 2017; Sotiropoulos and Zalesky, 2017). The great potential of dMRI is that it uniquely allows studying the human brain non-invasively and in vivo. However, it relies on indirect inference from the data, and typically, modelling frameworks are necessary to map dMRI measurements to neuroanatomical features, which can be computationally expensive. The computational cost is becoming even higher given the trend of increasing data sizes. New MRI hardware and sequences for pushing spatial and angular resolution (Vu et al., 2015; Setsompop et al., 2018) can considerably increase the size of a single subject dataset. At the same time, big imaging data repositories with a large number of subjects are being created, such as the Human Connectome Project (HCP) (Van Essen and Ugurbil, 2012; Van Essen et al., 2012; Sotiropoulos et al., 2013), where 1200 datasets are included, its Lifespan and Disease extensions (<https://www.humanconnectome.org>) and UK Biobank (Miller et al., 2016; Alfaro-Almagro et al., 2018) where a total of 100,000 subjects are being scanned. Limitations in computing can restrict data exploration and even methodology development.

A common application of dMRI analysis is tissue microstructure estimation. Even if the diffusion tensor imaging model (DTI) (Basser et al., 1994a,b) is by far the most popular framework for extracting microstructure indices and can be fitted linearly to data, it has major limitations such as the inability to capture partial volume, leading to non-specific markers of tissue structural changes (Basser et al., 2000; Poupon et al., 2000; Wiegell et al., 2000; Alexander et al., 2001; Pierpaoli et al., 2001; Seunarine and Alexander, 2014). To overcome these limitations, multi-compartment biophysical models are being developed, where the diffusion signal attenuation is represented as a mixture of signals obtained from different tissue components (Szafer et al., 1995; Niendorf et al., 1996; Stanisz et al., 1997; Assaf and Cohen, 1998; Mukherjee et al., 1999; Assaf et al., 2008; Alexander et al., 2010; Sotiropoulos et al., 2012; Zhang et al., 2012). Multi-compartment dMRI models are commonly non-linear functions of the signal, and non-linear optimisation algorithms are typically used for fitting the model to the diffusion-weighted measurements (Motulsky and Ransnas, 1987; Kelley, 1999). These algorithms use iterative optimisation procedures for finding a global solution, leading to potentially large computational times. Furthermore, if a stochastic (instead of deterministic) optimisation method is required, such as Bayesian approaches (Tarantola, 2005), computation requirements are even heavier. For instance, using a cluster

with 72 CPU cores, fitting the NODDI-Bingham model (Tariq et al., 2016) to a single subject of the UK Biobank dataset with the current available Matlab toolbox (Microstructure Imaging Group - University College London, 2017) requires more than 6 h for deterministic fitting, and fitting the ball and sticks model (Behrens et al., 2003, 2007) to a single subject of the HCP dataset with FMRIB's Software Library (FSL) (Jenkinson et al., 2012) requires 5 h for stochastic fitting using MCMC on a large cluster with 145 CPU cores (see Table 1).

Optimisation methods used for fitting voxel-wise biophysical models to dMRI data are inherently parallelisable and in general well suited for GPU design, since the computational modelling is applied independently to different voxels. The large number of independent elements in the data and the fact that identical procedures need to be performed over each of these elements make GPUs a perfect candidate for processing these datasets, as they have been designed for exploiting the data level parallelism by executing the same instructions over different data simultaneously (SIMD (Flynn, 1972)). However, due to the heavy tasks involved in the optimisation procedures, the design of an optimal parallel solution is non-trivial.

A number of GPU frameworks for accelerating these routines have been developed in the past by ourselves and others focusing on specific models for fibre orientation or diffusion tensor estimation (Xu et al., 2012; Hernández et al., 2013; Chang et al., 2014). In this paper we reformulate our previous proposed approach (Hernández et al., 2013) and provide a generic and model independent toolbox for model fitting using GPUs. The toolbox provides a flexible and friendly front-end for the user to specify a model, define constraints and any prior information on the model parameters, and choose a non-linear optimisation routine, ranging from deterministic Gauss-Newton type approaches to stochastic Bayesian methods based on Markov Chain Monte Carlo (MCMC). It then automatically generates a GPU executable file that reflects all these options. This achieves flexibility in model fitting and allows a single GPU to achieve a better performance than 200 CPU cores.

To further explore the potential of GPUs for computational dMRI, we present another parallel framework for white matter tractography and connectome estimation, a common dMRI application with completely different features and challenges compared to the voxel-wise biophysical modelling. We focus here on probabilistic tractography approaches, which for certain applications can be very time consuming (Behrens et al., 2007). For instance, the generation of a “dense” connectome (Sporns et al., 2005) from a single subject using high-resolution data from the HCP can take more than 9 h on a large CPU cluster (see Table 1). In the case of tractography algorithms, the main challenge for a GPU parallel solution comes from the fact that the required data for propagating each streamline (set of voxels with distributions of fibre orientations) is not known in advance, as their paths are estimated dynamically on the fly. This makes the allocation of GPU resources difficult, and therefore, the a-priori assessment of the parallelisability of the application challenging. Moreover, the streamlines propagation is likely to be asynchronous as they may have imbalanced execution length, which induces *thread divergence* and causes performance degradation on GPUs.

Table 1

Execution times and memory requirements of some dMRI applications processing datasets from the UK Biobank project and the Human Connectome Project. Processing times are reported using several CPU cores from several modern processors (Intel Xeon E5-2660 v3 processors).

Application (Single subject)	Computational resources	Time	Memory Required
Ball & 2 sticks model (MCMC) - UK Biobank	72 cores	0.73 h	2.5 GB
Ball & 2 sticks model (MCMC) - HCP	145 cores	5 h	8 GB
NODDI-Bingham. Multi-compartment model - Biobank	72 cores	6.75 h	2.5 GB
Brain Connectome (dense) - HCP	100 cores	9.5 h	35 GB

Furthermore, these methods have typically high memory requirements and include relatively heavy tasks, particularly for large datasets and whole-brain explorations, making the design of an efficient GPU-accelerated solution (which ideally comprises light and small tasks) even less straightforward. Preliminary GPU parallel tractography frameworks have been proposed in the past (Mittmann et al., 2008; Xu et al., 2012); however, our tractography parallel framework achieves accelerations of more than 200 times compared to CPU-based implementations and includes novel features that allow even more accurate anatomical constraints to be imposed, such as the inherent support of surface meshes (GIFTI files (Harwell et al., 2008)), and the possibility of generating dense connectomes.

In summary, we illustrate that, despite differences in parallelisability challenges, well-thought GPU-based designs that are carefully implemented can offer the same performance as hundreds of CPU cores, within the different contexts of tissue microstructure estimation, and tractography and connectome generation. The developed frameworks will be released upon publication within the FSL software library (Jenkinson et al., 2012)¹.

2. Material and methods

2.1. Biophysical modelling on GPUs

2.1.1. Framework description

Tissue microstructure estimation from dMRI is typically performed on a voxel-by-voxel basis, where a biophysical model is fitted. Excluding the DTI model, which can be easily and quickly fitted using linear least squares, most models are non-linear and numerical optimisation routines are required. Non-linear optimisation is typically computationally demanding and can be very time consuming, particularly since advanced multi-compartment models (Alexander et al., 2017) require larger than average datasets (multiple *b-values* or high angular resolution).

Given the large number of voxels and the relatively low memory requirements of these independent tasks, such an application is well-suited for implementation on GPUs. To take advantage of the inherent parallelisability of the problem and yet cover all possible dMRI models, we have developed a generic toolbox for designing and fitting nonlinear models using GPUs and CUDA. The toolbox, CUDA diffusion modelling toolbox (cuDIMOT), offers a friendly and flexible front-end for the users to implement new models without the need for them to write CUDA code or deal with a GPU design, as this is performed by the toolbox automatically (Fig. 1). The user only specifies a model using a C-like language header. This model specification includes the model parameters, constraints and priors for these parameters, the model predicted signal function, and optionally the partial derivatives with respect to each parameter if they can be provided analytically (cuDIMOT offers the option for numerical differentiation). Once the model specification has been provided, the toolbox integrates this information with the parallel CUDA design of the corresponding fitting routines at compilation time, and it generates a GPU executable. The fitting routines include options for both deterministic (e.g. Gauss-Newton type) and stochastic (e.g. Bayesian inference using MCMC) optimisation.

An important factor to take into account in the design of the framework is its generic and model-independent aspect. In order to achieve an abstraction of the fitting routines, these are implemented in a generic way excluding functions that are model-dependent. The management of threads, definition of grid size and data distribution are also challenging aspects that cuDIMOT automates. The fitting routines, deterministic and stochastic, are implemented in different CUDA kernels. The toolbox implements the different kernels, deals with the arduous task of distributing

the data among thousands of threads, uses the GPU memory spaces efficiently, and even distributes the computation among multiple GPUs if requested. Two levels of parallelism are used in our design (see Fig. 2). A first level distributes the fitting process of all the voxels amongst CUDA warps (groups of 32 CUDA threads). Specifically, the fitting process of a few voxels is assigned to a CUDA block (a group of CUDA warps), and each warp fits the model to a single voxel. In a second level of parallelisation, the computation of the most expensive within-voxel tasks is distributed amongst threads within a warp, including the computation of the model predicted signal and residuals, and the partial derivatives with respect to the model parameters across the different measurement points. More details about the parallel design and implementation of cuDIMOT are provided in the [Supplementary material](#).

Additionally, a higher-level of parallelism can be further used to enhance even more the performance, using very large groups of voxels and a multi-GPU system. We can divide a single dataset into groups of voxels and assign each group to a different GPU. The different GPUs do not need to communicate because the groups of voxels are completely independent, apart from the final step of outputting the results.

In terms of optimisation routines, cuDIMOT offers a number of deterministic and stochastic model-fitting approaches, including greedy optimisation using Grid-Search, non-linear least-squares optimisation using Levenberg-Marquardt (LM) and Bayesian inference using Markov Chain Monte Carlo (MCMC).

MRI models can have free parameters, which are estimated, and fixed parameters, which reflect measurement settings or features that are known. cuDIMOT allows such fixed parameters to be defined, and these may be common to all voxels or not (CFP or common fixed parameters and FixP or fixed parameters in [Supplementary Fig. 3a](#)). For instance, in typical diffusion MRI, the diffusion-sensitising gradient strengths (*b* values) and associated directions (*b* vectors) would be CFPs, whereas for diffusion-weighted steady-state free precession (DW-SSFP) (McNab and Miller, 2008), the flip angle (α) and repetition time (TR) would be CFPs, while the longitudinal and transverse relaxation times (T1 and T2) would be FixP, as they vary across voxels. Using a simple syntax, a list with all the information is passed to the toolbox through the designer interface. This information is parsed by cuDIMOT and used at execution time, when the model user must provide maps with these parameters. This generic interface allows the users to combine data from dMRI with data from other modalities, such as relaxometry (Deoni, 2010), and develop more complex models (Foxley et al., 2015, 2016; Tendler et al., 2018), or, even use cuDIMOT in different modalities where nonlinear optimisation is required.

Prior information or constraints on the parameters of a model can be integrated into the fitting process using the toolbox interfaces, where a simple syntax is used for enumerating the type and the value of the priors (see [Supplementary Fig. 3b](#)). Upper and lower limits or bounds can be defined for any parameter (transformations are implemented internally and described in the [Supplementary material](#)), and priors can be any of the following types:

- A Normal distribution. The mean and the standard deviation of the distribution must be specified.
- A Gamma distribution. The shape and the scale of the distribution must be specified.
- Shrinkage prior or Automatic Relevance Determination (ARD) (MacKay, 1995). The factor or weight of this prior must be provided in the specification.
- Uniform distribution within an interval and uniform on a sphere (for parameters that describe an orientation).

For stochastic optimisation, a choice can also be made on the noise distribution and type of the likelihood function (Gaussian or Rician).

Because high-dimensional models are difficult to fit without a good initialisation, the toolbox offers an option for cascaded fitting, where a simpler model is fitted first, and the estimated parameters are used to

¹ Current versions of the toolboxes are publicly available on: <https://users.fmrib.ox.ac.uk/~moisesf/cudimot/index.html> and https://users.fmrib.ox.ac.uk/~moisesf/Probtrackx_GPU/index.html.

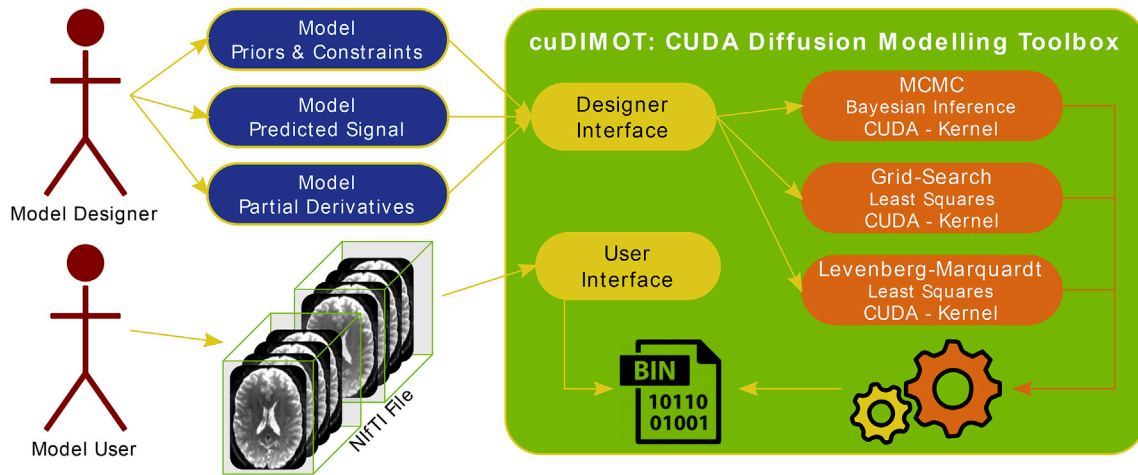


Fig. 1. General design of CUDA Diffusion Modelling Toolbox (cuDIMOT). Two types of users interact with the toolbox through interfaces, a model designer and a model user. The model designer provides the model specification (parameters, priors, constraints, predicted signal and derivatives), whereas the model user interacts with the toolbox for fitting the model to a dataset. The toolbox provides CUDA kernels that implement several fitting routines. These kernels are combined with the model specification at compilation time for generating a GPU executable application.

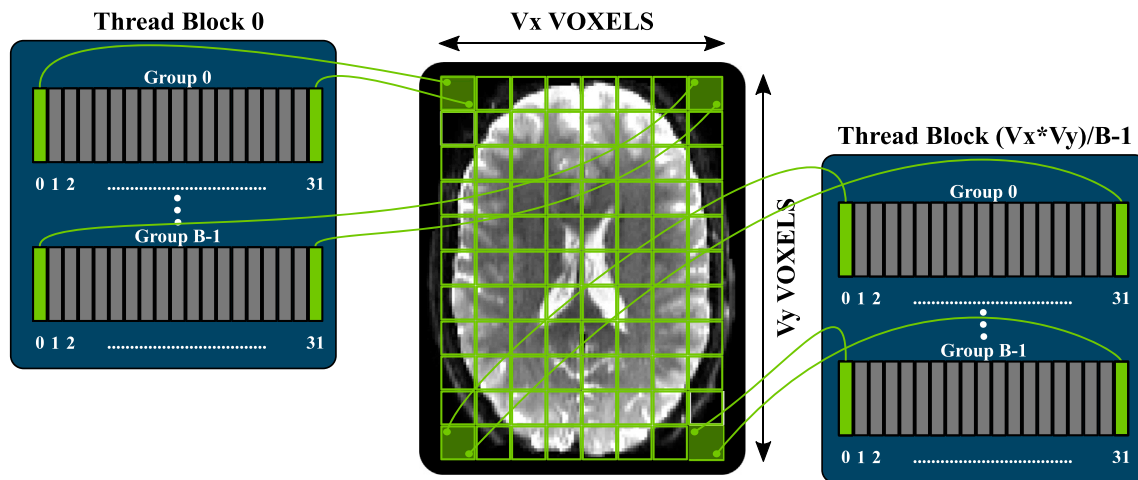


Fig. 2. Parallel design of cuDIMOT for fitting dMRI models on a GPU. The V voxels of a dataset are divided into groups of B voxels (voxels per block), and the fitting process of each of these groups is assigned to different CUDA blocks. Inside a block, a warp (32 threads) collaborate for within-voxel computations.

initialise the parameters of a more complex model. 3D volumes can be used for specifying the initialisation value of the model parameters in every voxel.

Once a model has been defined and an executable file created, a user still has flexibility in controlling a number of fitting options, including:

- Choosing fitting routines: Grid-Search, Levenberg-Marquardt or MCMC. A combination of them is possible, using the output of one to initialize the other.
- Selecting number of iterations in Levenberg-Marquardt and MCMC (burn-in, total, sample thinning interval).
- Using Gaussian or Rician noise modelling in MCMC.
- Choosing model parameters to be kept fixed during the fitting process.
- Choosing model selection criteria to be generated, such as BIC and AIC.

2.1.2. Exploring microstructure diffusion MRI models with cuDIMOT

We used cuDIMOT for implementing a number of diffusion MRI models and assess the validity of the results. We have implemented the Neurite Orientation Dispersion and Density Imaging (NODDI) model,

using Watson (Zhang et al., 2012) and Bingham (Tariq et al., 2016) distributions for characterising orientation dispersion.

We implemented NODDI-Watson with cuDIMOT using the designer interface. This model assumes the signal comes from three different compartments: isotropic compartment, intra-cellular compartment and extra-cellular compartment. The model has five free parameters: the fraction of the isotropic compartment f_{iso} , the fraction of the intra-cellular compartment relative to the aggregate fraction of the intra-cellular and extra-cellular compartments f_{intra} , the concentration of fibre orientations κ (the lower this value the higher the dispersion), and two angles for defining the mean principal fibre orientation θ and ϕ . The concentration parameter κ can be transformed and expressed as the orientation dispersion index $OD \in [0, 1]$:

$$OD = \frac{2}{\pi} \arctan\left(\frac{1}{\kappa}\right) \quad (1)$$

We implemented the model predicted signal of NODDI-Watson as in (Zhang et al., 2011), providing analytically the derivatives for f_{iso} , f_{intra} . We used numerical differentiation to evaluate the partial derivatives of the rest of parameters. We used numerical approximations (e.g. for the Dawson's integral) as in (Press et al., 1988), and we performed the same

cascaded steps as the Matlab NODDI toolbox (Microstructure Imaging Group - University College London, 2017). First, we fit the diffusion tensor model for obtaining the mean principal fibre orientation (θ and ϕ). Second, we run a Grid-Search algorithm testing different combination of values for the parameters f_{iso} , f_{intra} and κ . Third, we run Levenberg-Marquardt algorithm fitting only f_{iso} , f_{intra} and fixing the rest of parameters. Finally, we run Levenberg-Marquardt, fitting all the model parameters. The only difference is that the Matlab toolbox uses an active set optimisation algorithm (Gill et al., 1984) instead of Levenberg-Marquardt.

The NODDI-Bingham model assumes the same compartments as NODDI-Watson. However, this model can characterise anisotropic dispersion, and thus it has two concentration parameters κ_1 , κ_2 , and an extra angle ψ which is orthogonal to the mean orientation of the fibres, and encodes a rotation of the main dispersion direction. When $\kappa_1 = \kappa_2$ the dispersion is isotropic, and when $\kappa_1 > \kappa_2$ anisotropic dispersion occurs. In this case, the orientation dispersion index OD is defined as:

$$OD = \frac{2}{\pi} \arctan \left(\sqrt{\left(\frac{1}{\kappa_2} \right) \left(\frac{1}{\kappa_1} \right)} \right) \quad (2)$$

and an index $DA \in [0, 1]$ reflecting the factor of anisotropic dispersion can be defined as:

$$DA = \frac{2}{\pi} \arctan \left(\frac{\kappa_1 - \kappa_2}{\kappa_2} \right) \quad (3)$$

We implemented NODDI-Bingham using cuDIMOT in a similar manner as the previous model (only providing the analytic derivatives for the f_{iso} and f_{intra} parameters). For implementing the confluent hypergeometric function ${}_1F_1$ of a matrix argument, included in the predicted signal of the intra-cellular and extra-cellular compartments of the model, we use the approximation described in (Kume and Wood, 2005). We use the same optimisation steps as in the NODDI-Watson: diffusion tensor fitting, Grid-Search, and Levenberg-Marquardt twice.

2.2. Probabilistic tractography and connectomes on GPUs

Contrary to voxel-wise model fitting, white-matter tractography, and particularly whole-brain connectome generation, are not inherently suited for GPU parallelisation. Very high memory requirements, *uncoalesced memory accesses* and threads divergence (irregular behaviour of threads in terms of accessed memory locations and life duration) are some of the major factors that make a GPU parallel design of such an application challenging. Nevertheless, we present a framework that parallelises the propagation of multiple streamlines for probabilistic tractography and overcomes the aforementioned issues using an overlapping pipeline-design.

2.2.1. Framework description

We design a parallel design and develop a GPU-based framework for performing probabilistic tractography. Our application includes the common tractography functionality, including for instance options to set:

- The number of streamlines propagated from each seed point, i.e., the number of samples.
- Streamline termination criteria (maximum number of steps, curvature threshold, anisotropy threshold, tract loop detection)
- A number of numerical integration approaches, including Euler's method and 2nd order Runge-Kutta method (Basser et al., 2000), with a subsequent choice of step length.
- Propagation criteria and anatomical constraint rules (seed, waypoint, termination, stopping, and target masks) (Smith et al., 2012).
- Ability to accept tracking protocols in either diffusion or structural/standard space.

Connectome generation is also inherently supported (Sporns et al., 2005) and three options are available (see Fig. 3) (Li et al., 2012; Donahue et al., 2016):

- Connectivity matrix between all the seed points and all the other seed points. A typical use is for studying the connectivity from all grey matter to all grey matter (Glasser et al., 2013).
- Connectivity matrix between all the points of two different masks, which are independent of the seed points mask. A typical use is for studying the connectivity between grey matter regions, when seeding from all white matter.
- Connectivity matrix between all the seed points and all the points specified in a different mask. A typical example is to use the whole brain as a target mask for studying the connectivity profile of the grey matter in a specific seed, and use it for connectivity-based classification (Johansen-Berg et al., 2004).

We have also included an extra feature that is not typically found in tractography toolboxes, but can be important in defining anatomically accurate constraints. We included in our parallel framework the possibility of using surfaces, as well as volumes, for defining seed and regions of interest (ROIs). We implement support for the GIFTI format (Harwell et al., 2008), according to which surfaces are defined by meshes of triangles. Three spatial coordinates define each triangle vertex in a 3D space. Surface vertices can be used for defining seed points, and mesh triangles can be used for defining stopping/constraint masks. If the latter, a streamline needs to be checked upon crossing the surface meshes. We implement the method described in (O'Rourke, 1998) (ray-plane intersection) for checking if the segments of a streamline intersects a triangle (details of the method are presented in the [Supplementary material](#)).

When designing a parallel solution, we first notice that path

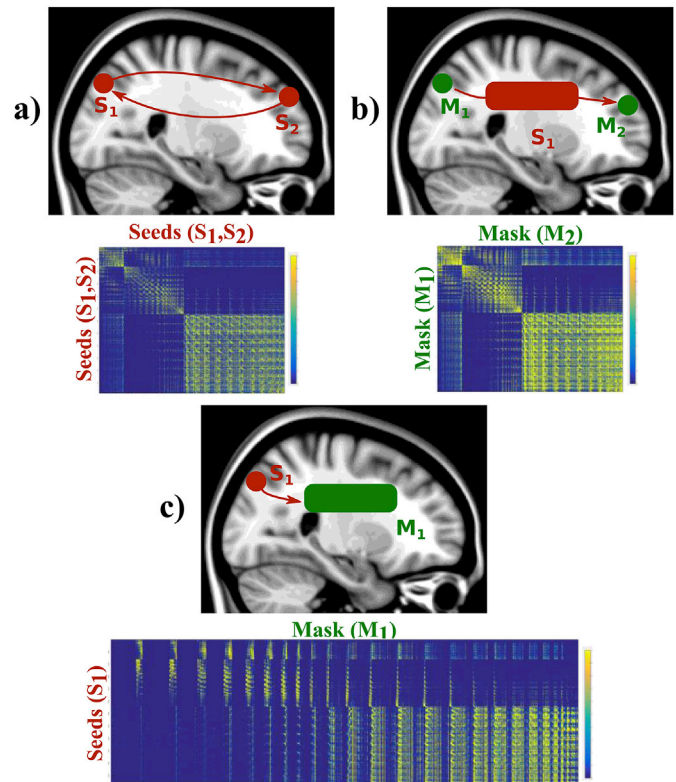


Fig. 3. Connectivity matrices modes offered by the GPU-accelerated tractography framework. The framework can generate connectivity matrices from a) all seed to all seed points, b) all points in a mask to all points in another mask seeding from an independent region, or c) all seed points to all points in a different mask.

propagation is completely independent across streamlines, and thus it can be in principle parallelised. To reflect that, we can create as many CUDA threads as the required number of streamlines, in fact twice the number of streamlines, as we propagate from each seed point towards both directions indicated by the local fibre orientation. Thus, for D seed points and F streamlines per seed, we create $2 \times D \times F$ threads in total (see Fig. 4). Nevertheless, there are complexities that make such a design challenging and considerably reduce efficiency if not addressed, as we explain below. These include heavy tasks, thread divergence and memory allocation challenges.

A first consideration is the high complexity of some of the routines included for implementing the offered functionality. For instance, the use of surfaces involves the execution of an intersection detection algorithm, while the streamline propagation includes interpolation and space transformation routines. Furthermore, checking anatomical constraints during propagation increases the complexity of the algorithm, and induces a significant number of conditional branches. Having a single CUDA kernel for performing all these tasks leads to substantially heavy threads, which consume a lot of computational resources and consequently cause low occupancy in a GPU Streaming Multiprocessor (SM, see Supplementary Fig. 1). To solve this issue, we split the application into multiple tasks, each of which is implemented in a different CUDA kernel. A pipelined design is used to execute these kernels, running them serially one after the other. A first kernel propagates the streamlines, and subsequently, kernels for checking anatomical constraints, generating path distribution maps and generating connectomes are executed. Further details of these kernels and the executing pipeline of the application are included in the Supplementary material.

Another main challenge is related to memory requirements. It may be impossible to use GPUs if the memory demands exceed the available device memory, and typically, this is true across all levels in the GPU memory hierarchy. Tractography algorithms need a distribution of fibre orientations for sampling. As we cannot predict the streamline track locations in advance, the fibre orientation distributions of all voxels need to be allocated in memory. The amount of required memory depends on the size (spatial dimensions) of the dataset and for probabilistic tracking on the number of samples in the orientation distributions. For instance, the required memory for simply allocating the samples of a Human Connectome Project dataset (Van Essen and Ugurbil, 2012; Van Essen et al., 2012; Sotiropoulos et al., 2013) is approximately 1.5 GB. Moreover, the 3D streamline coordinates need to be stored, but the number of steps that a streamline will take cannot be predicted in advance. We therefore need to allocate enough memory for storing the maximum possible number of coordinates for each streamline. Additionally, volumes and/or surfaces may be used for defining seeds, anatomical constraints and connectome matrix elements, and all of them need also to be allocated in memory. Our strategy for overcoming this issue was to allocate all the required memory in the GPU without considering the streamline coordinates, and

then propagate the maximum number of streamlines that can be computed in parallel given the amount of memory left. If all the requested streamlines cannot be computed in parallel (which is the most typical scenario), the application iterates over different streamline sets.

Another challenge that limits the performance is thread divergence. The streamlines may be initialised at different seed points and they can propagate to different regions and for different lengths. This causes accesses to different memory locations when sampling, i.e., uncoalesced memory accesses. Furthermore, streamlines may terminate at different time points. This causes asynchronous thread termination and a possible waste of computational resources, as some threads finish their execution before others and stay idle, wasting GPU resources. This situation persists until all the threads of the same CUDA block finish their execution and the GPU scheduler switches the block with another block of threads. For this reason, we set the block size K to a small size of 64 threads (2 warps). Although with this configuration full SM occupancy is not achieved (because there is a limit in the number of blocks per SM), there can be fewer divergences than having larger blocks. Moreover, the GPU can employ the unused resources by overlapping other tasks/kernels in parallel.

We indeed take advantage of our pipelined design and use *CUDA streams* to overlap the computation of different sets of streamlines. CUDA streams are queue instances for managing the order of execution of different tasks (NVIDIA, 2015a) and offer the possibility of running concurrently several tasks (kernels execution and/or CPU-GPU memory transfers) on the same GPU. Our framework divides the streamlines into a few sets, and uses a number of OpenMP (Chapman et al., 2008) threads to execute the pipeline of several streamline sets on different CUDA streams concurrently (see Supplementary Fig. 6b).

2.3. Diffusion-weighted MRI data

The GPU designs presented in this paper have already been used to process hundreds of datasets. Here, we illustrate performance gains on various exemplar data with both high and low spatial resolutions.

For testing the diffusion modelling framework (cuDIMOT) we used data from UK Biobank (Miller et al., 2016). Diffusion-weighting data were acquired using an EPI-based spin-echo pulse sequence in a 3T Siemens Skyra system. A voxel size of $2.0 \times 2.0 \times 2.0 \text{ mm}^3$ was used ($TR = 3.6 \text{ s}$, $TE = 92 \text{ ms}$, 32-channel coil, 6/8 partial Fourier) and 72 slices were acquired. Diffusion weighting was applied in $M = 100$ evenly spaced directions, with 5 directions $b = 0 \text{ s/mm}^2$, 50 directions $b = 1000 \text{ s/mm}^2$ and 50 directions $b = 2000 \text{ s/mm}^2$. A multiband factor of 3 was employed (Moeller et al., 2010; Setsompop et al., 2012). A T1 structural image (1 mm isotropic) of the same subject was used for creating a white & grey matter mask, which was non-linearly registered to the space of the diffusion dataset (Andersson et al., 2007), and applied to the map of the estimated parameters before showing the results included in this paper. For creating this mask, a brain extraction tool (Smith, 2002) and a tissue segmentation tool (Zhang et al., 2001) were used.

For testing the GPU probabilistic tractography framework, data were acquired on a 3T Siemens Magnetom Prisma using HCP-style acquisitions (Sotiropoulos et al., 2013). Diffusion-weighting was introduced using single-shot EPI, using an in-plane resolution of $1.35 \times 1.35 \text{ mm}^2$ and 1.35 mm slice thickness ($TR = 5.59 \text{ s}$, $TE = 94.6 \text{ ms}$, 32-channel coil, 6/8 partial Fourier). 134 slices were acquired in total and diffusion weighting was applied in $M = 291$ evenly spaced directions, with 21 directions $b = 0 \text{ s/mm}^2$, 90 directions $b = 1000 \text{ s/mm}^2$, 90 directions $b = 2000 \text{ s/mm}^2$ and 90 directions $b = 3000 \text{ s/mm}^2$.

2.4. Hardware features

We used an Intel host system with NVIDIA GPUs and a large cluster of Intel processors for testing our parallel designs. The system has a dual NVIDIA K80 accelerator (Error Correcting Codes ECC enabled),

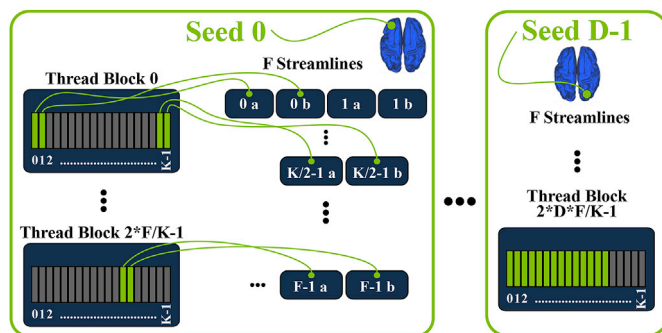


Fig. 4. GPU parallel design of the streamline probabilistic tractography framework. For each of the D seeds and for each of the F streamlines per seed, we create two CUDA threads (a and b), which are distributed amongst blocks of K threads.

connected to the host via PCI express v3. A single GPU was used for the experiments. The system has tens of CPU nodes, and each node is comprised of 2 Intel Xeon E5-2660 v3 2.60 GHz processors, each with 10 cores (20 CPU cores per node), and 384 GB (24 × 16 GB) RDIMM memory. Major features of the NVIDIA K80 accelerators and the Intel processors are summarized in [Supplementary Table 1](#) (NVIDIA, 2014a, 2015b).

The systems run Centos 6.8 Linux. We compiled our code using CUDA 7.5 (V.7.5.17) and gcc 4.4.7 compilers.

3. Results

3.1. Tissue microstructure modelling with GPUs

We fit the NODDI-Watson model to a UK Biobank dataset using three approaches, the NODDI Matlab toolbox ([Microstructure Imaging Group - University College London, 2017](#)), AMICO ([Daducci et al., 2015](#)) and cuDIMOT. Although Matlab applications are not as optimised as C/C++ applications, the only available version of NODDI is implemented in Matlab. Despite this issue, the NODDI toolbox can parallelise the fitting process distributing groups of voxels among several CPU threads. AMICO reformulates the problem as a linear system via convex optimisation and accelerates computations by performing discrete searches in the multi-dimensional space of the problem. [Fig. 5a](#) shows maps with the estimated parameters from each approach and the respective execution times. Both cuDIMOT and AMICO achieved accelerations of more than two orders of magnitude compared to NODDI toolbox (cuDIMOT 352x and AMICO 160x) using a single NVIDIA K80 GPU and a single CPU core respectively. cuDIMOT was 2.2 times faster than AMICO.

To compare the estimates, we treat the NODDI toolbox results as ground truth, and we calculate the percentage absolute difference with the estimates obtained from the other two approaches. [Fig. 5b](#) shows higher differences with AMICO than with cuDIMOT for some of the estimated parameters. The differences between the Matlab implementation and cuDIMOT are insignificant, except for the parameter f_{iso} in certain parts of the white matter. However, the values of f_{iso} are very low in the white matter, and the absolute difference between the Matlab toolbox and cuDIMOT are very small (~ 0.003) (See [Supplementary Fig. 8](#)). The absolute differences between the Matlab toolbox and AMICO are also small, but more significant (~ 0.03).

To further explore these findings, [Fig. 6](#) shows scatterplots of the estimated values throughout the brain. The correlations between the Matlab implementation and AMICO, and between the Matlab toolbox and cuDIMOT are presented. cuDIMOT results are highly correlated to the results from the Matlab tool ([Supplementary Fig. 9](#) includes Bland-Altman plots). The discretisation approach used in AMICO is evident in these scatterplots, particularly for the f_{intra} parameter, with discretisation effects. For cuDIMOT, correlations with ground truth is higher. We only find some differences in a few grey matter voxels where OD takes relatively high values ($OD > 0.85$, i.e. very high fibre orientation dispersion). We compared the distribution of the estimated values for the parameter κ , from which OD is derived (See [Supplementary Fig. 10a](#)), and we found that for low values of κ ($\kappa < 1$), the Matlab toolbox seems to get trapped in a local minimum ($\kappa = 0.5$), whereas in cuDIMOT the value of the parameter is pushed towards the lower bound ($\kappa = 0.0$). Moreover, we found that this happens in a very low proportion of voxels located at the interface between grey matter and CSF (See [Supplementary Fig. 10b](#)). We believe that these differences are due to the numerical approximations used. In the cuDIMOT implementation we approximate the Dawson's integral as in ([Press et al., 1988](#)). Likely, the Matlab toolbox is using a different approximation. Overall, these results indicate that cuDIMOT achieves very similar performance to the NODDI Matlab toolbox, and compared with AMICO, cuDIMOT is faster and obtains more accurate results.

We also performed similar comparisons for the NODDI-Bingham model ([Microstructure Imaging Group - University College London,](#)

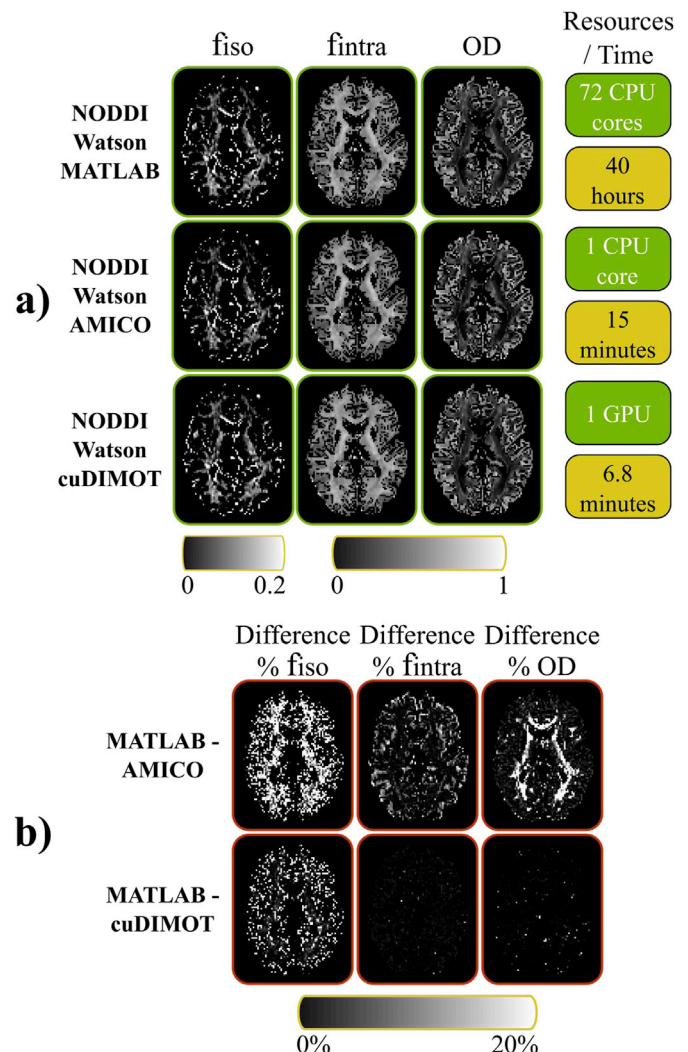


Fig. 5. Comparison of three different tools fitting the NODDI-Watson model. (a) The results from each tool are presented in different rows. The first 3 columns show the map of the estimates for the parameters f_{iso} , f_{intra} and the index OD . The 4th column shows the employed computational resources and the execution times. (b) Differences, in percentage, of the estimated values between the Matlab toolbox and the other approaches.

2017). Using a single GPU, cuDIMOT was found to be 7 times faster than the Matlab implementation running on a cluster with 72 cores² ([Fig. 7](#)). We obtain very similar results from both tools; however, the percentage absolute differences (bottom row in [Fig. 7](#)) are on average higher compared to the NODDI Watson model. To gain further insight, [Fig. 8](#) shows scatterplots of the parameter values estimated using both methods throughout the brain. In all the parameters, the correlation coefficient was higher than 0.984 in the white matter, and 0.977 in the grey matter. Notably, we found some voxels where one toolbox returns a very low DA (near zero) but not the other. We found that these voxels represent a very low proportion of the whole dataset, 0.2%, and they are at the interface between white matter and CSF. We believe that the source of these differences come from:

² Note the almost sixfold difference between fitting NODDI-Bingham vs. NODDI-Watson using the Matlab toolbox, despite the higher complexity of NODDI-Bingham. This is due to the inefficient approximation of the confluent hypergeometric function of a scalar argument used in the NODDI-Watson Matlab implementation. For this reason, the comparisons in performance gains with cuDIMOT are more meaningful in the NODDI-Bingham case.

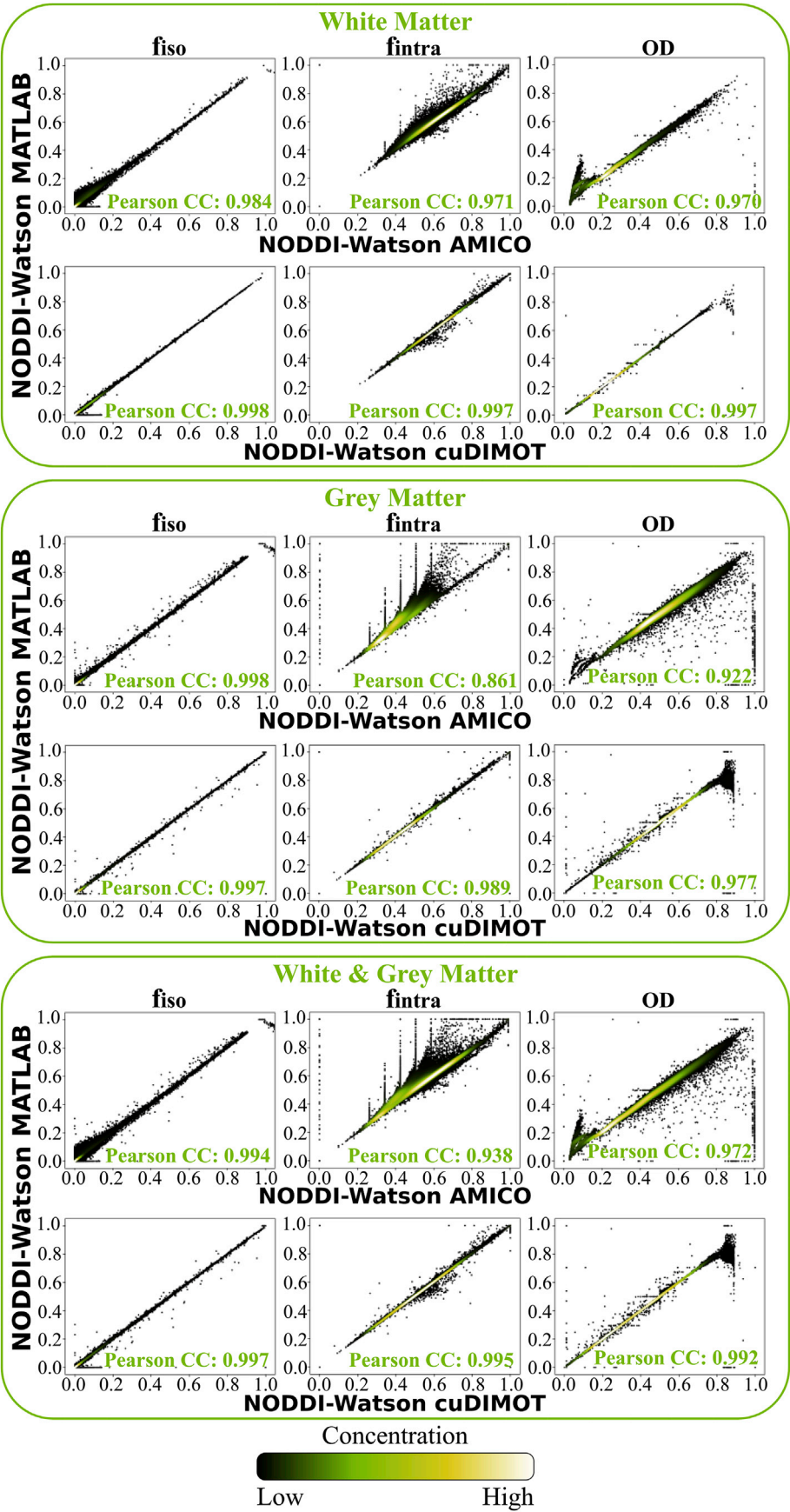


Fig. 6. Correlations between the results from NODDI Matlab toolbox and AMICO/cuDIMOT fitting the NODDI-Watson model in the white matter, grey matter and the combination of white & grey matter.

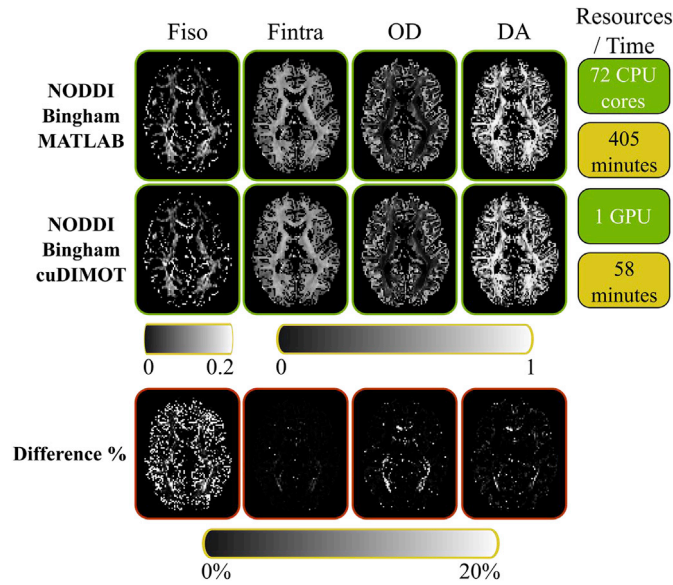


Fig. 7. Comparison of a Matlab tool and cuDIMOT fitting the NODDI-Bingham model. The first 2 rows show the map of the estimates for the parameters f_{iso} , f_{intra} , the indices OD and DA , the used computational resources and execution times for each tool. The bottom row shows the differences, in percentage, of the estimated parameters between the Matlab tool and cuDIMOT.

- Using a different approximation of the hypergeometric function. In cuDIMOT we use a Saddlepoint approximation (Kume and Wood, 2005) and in the Matlab toolbox the function is approximated as in (Koev and Edelman, 2006).
- Different non-linear optimisation method. We use Levenberg-Marquardt whereas the Matlab toolbox uses the active set algorithm included in the `fmincon` function (Gill et al., 1984).

We also found a few voxels where DA is estimated with values around 0.5 in cuDIMOT, whereas in the Matlab toolbox the values are different. This seems to be related to the initial Grid-Search routine and the values that define the grid for the second concentration parameter κ_2 . Both Matlab toolbox and cuDIMOT reparametrise this parameter as $\beta = \kappa_1 - \kappa_2$. However, in cuDIMOT we include in the grid a set of values (from 0 to 16), whereas Matlab toolbox uses a single constant value to initialise this parameter, defined by the coefficient between the second and third eigenvalues of the diffusion tensor. Nevertheless, overall we obtain very high correlations between both toolboxes.

To assess speed-ups achieved by cuDIMOT, we implemented several dMRI models. We report in Table 2 the speedups obtained by cuDIMOT using a single NVIDIA K80 GPU, compared to the commonly used tools for fitting these models running on 72 CPU cores, including C++ and Matlab implementations. A Biobank dataset was used for this experiment. We considered the following models:

- Ball & 1 stick (Behrens et al., 2003, 2007)
- Ball & 2 sticks
- Ball & 1 stick (with gamma-distribution for the diffusivity (Jbabdi et al., 2012))
- Ball & 2 sticks (with gamma-distribution for the diffusivity)
- NODDI-Watson
- NODDI-Bingham

On average (and excluding NODDI-Watson implementation), using a single GPU cuDIMOT achieves accelerations of 4.3x.

To illustrate the flexibility of cuDIMOT in defining new models, and the benefits from accelerations that allow extensive model comparison, even with stochastic optimisation, we used cuDIMOT to test whether

crossing or dispersing models are more supported by the data. We performed a comparison of six diffusion MRI models and used the BIC index for comparing the performance. The models included in this test were:

- Ball & 1 stick (with gamma-distribution for the diffusivity (Jbabdi et al., 2012))
- Ball & 2 sticks (with gamma-distribution for the diffusivity)
- NODDI-Watson
- Ball & racket (Sotiropoulos et al., 2012)
- NODDI-Bingham
- NODDI-2-Binghams: we implement an extension of the NODDI-Bingham model (Tariq et al., 2016) for including two fibre orientations with the model signal given by:

$$S_m = S_0 \left[\frac{f_{iso} S_m^{iso} + (1 - f_{iso})(1 - f_{an2})(f_{intra1} S_m^{intra1} + (1 - f_{intra1}) S_m^{extra1})}{(1 - f_{iso})(f_{an2})(f_{intra2} S_m^{intra2} + (1 - f_{intra2}) S_m^{extra2})} \right] \quad (4)$$

S_m^{iso} , S_m^{intra1} , S_m^{intra2} , S_m^{extra1} and S_m^{extra2} are defined as in the NODDI-Bingham model.

The model has a total of 14 free parameters:

- Compartments fraction: f_{iso} , f_{an2} , f_{intra1} , f_{intra2}
- First fibre distribution: κ_{1-1} , κ_{1-2} , θ_1 , ϕ_1 , ψ_1
- Second fibre distribution: κ_{2-1} , κ_{2-2} , θ_2 , ϕ_2 , ψ_2

In all cases we ran an initialisation routine (Grid-Search or the output of the fitting process of another model), we run Levenberg-Marquardt and MCMC. cuDIMOT calculates the BIC from the mean of the parameter estimates. We first classify the six models into two groups, one group with the models that do not characterise the dispersion of fibre orientations, which include the ball & stick(s) models, and another group with the models that characterise the dispersion. The second row in Fig. 9 shows a colour-coded map indicating in what voxels each group gets a better BIC (lower), i.e. its complexity is better supported by the data. Using dispersion models the diffusion signal is better explained, and the obtained BIC is lower in the majority of brain regions. The last row of Fig. 9 compares the four considered dispersion models. The dominant model that gets a lower BIC is NODDI-Bingham (55% of the voxels) followed by NODDI-Watson (24% of the voxels), consistent with the results presented by (Ghosh et al., 2016). Interestingly, 5% of the voxels, particularly in the centrum semiovale, support the existence of dispersing populations crossing each other.

3.2. Tractography with GPUs

In order to validate the GPU-accelerated probabilistic tractography framework we performed various tests and compared the results with the results obtained using a CPU-based tractography application (Smith et al., 2004; Behrens et al., 2007), as implemented in FSL, for both white matter tract reconstruction and connectome generation. Given the stochastic nature of probabilistic tractography methods, we expect some variability in the results of both frameworks, but given the high number of samples that we have used in the tests, we expect the results to have converged and the variability to be small. Nevertheless, we run every experiment 10 times and we compare the differences between CPU and GPU results with the run-rerun variability.

Fig. 10a shows some quantitative comparisons of reconstructed tracts using both implementations. We reconstructed 27 major white matter tracts as in (de Groot et al., 2013) (12 bilateral and 3 commissural ones) using standard space tracking protocols and constraints (see Table 3 for a list of reconstructed tracts). In a different test, we generated dense connectomes using the HCP grayordinates (91K seed points (Glasser et al., 2013)). To quantify these comparisons, we present the run-rerun variability of each framework independently and the distribution of correlation coefficients between the CPU-based and the GPU-based

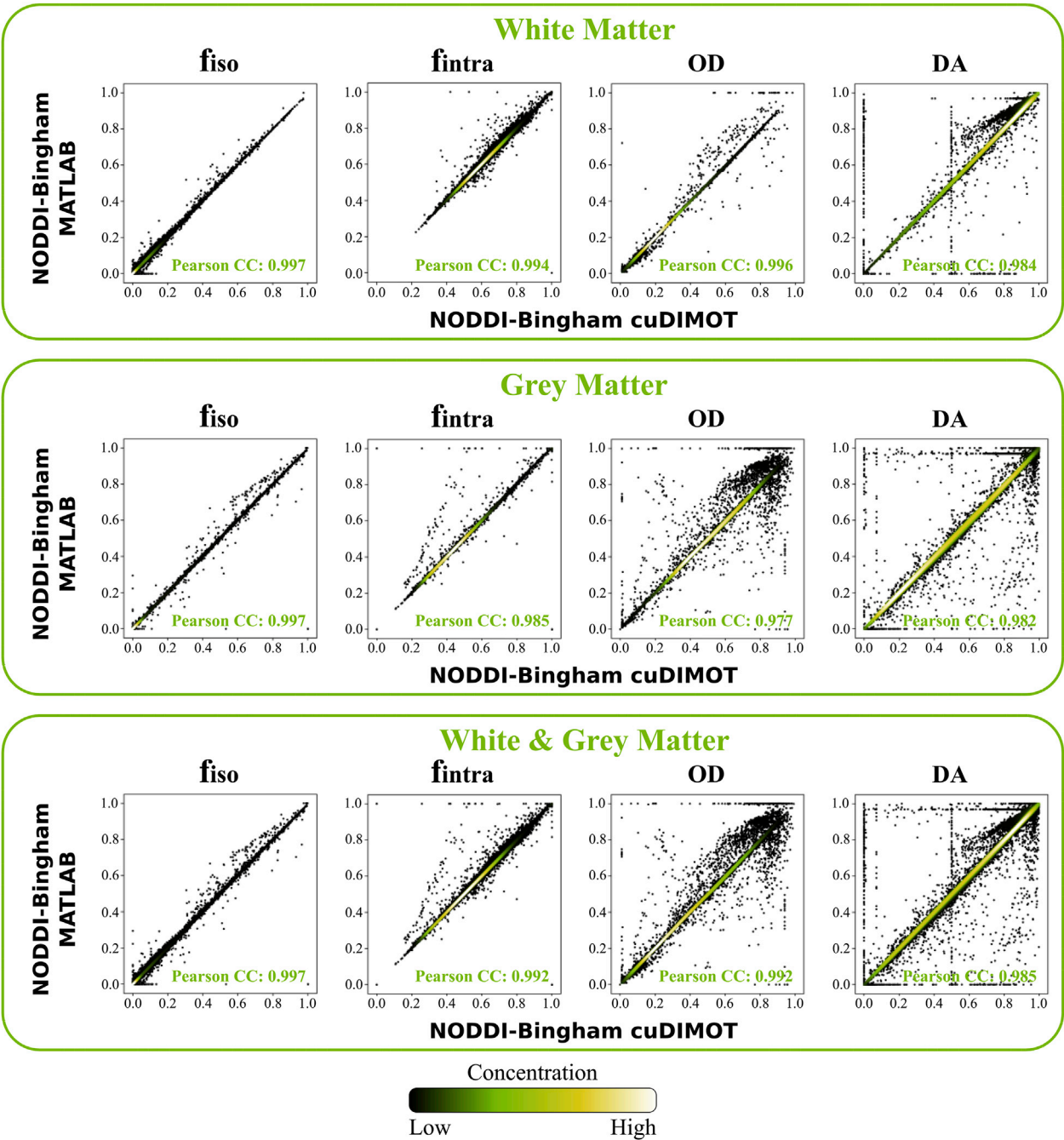


Fig. 8. Correlations between the results from a Matlab tool and cuDIMOT fitting the NODDI-Bingham model in the white matter, grey matter and the combination of white & grey matter.

Table 2
Speedups obtained by cuDIMOT, fitting several dMRI models to a dataset from the UK Biobank on a single K80 NVIDIA GPU, compared with the commonly used tools that implement these models and executed on a computing cluster using 72 CPU cores (and a single CPU thread per core).

	Ball & 1-stick C++	Ball & 2-sticks C++	Ball & 1-stick + Gamma C++	Ball & 2-sticks + Gamma C++	NODDI Watson Matlab	NODDI Bingham Matlab
Common Tools 72 CPU cores	720s	1380s	1260s	2520s	2400m	405m
cuDIMOT single NVIDIA K80 GPU	187s	423s	324s	679s	6.8m	58m
Speedup	3.85x	3.26x	3.88x	3.7x	352x	6.98x

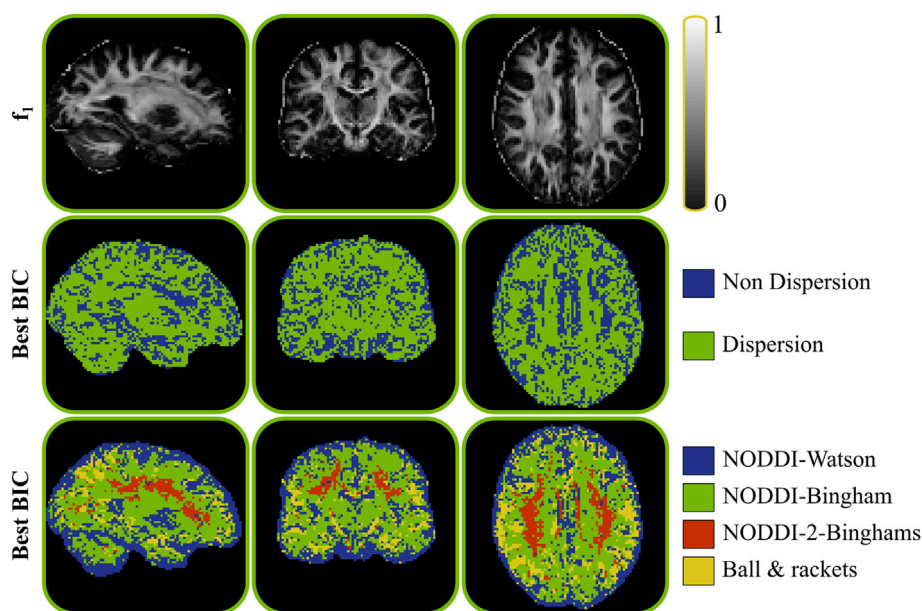


Fig. 9. Model performance comparison. The first row shows a map for reference with the estimated fraction of the principal fibre in the ball & 2 sticks model. The second and third rows show color-coded maps indicating in what locations a model or a group of models get the best BIC.

frameworks. In the reconstruction of the 27 tracts the correlation was calculated voxel-wise. In the generation of the dense connectome, the correlation was calculated from all the elements of the connectivity matrix. The individual run-rerun correlation coefficients are higher than 0.999 in all the cases, for both the CPU and the GPU frameworks. Importantly, the correlation coefficients between CPU and GPU are higher than 0.998, illustrating that the two implementations provide the same solution. Even if these correlations are slightly lower than between the individual run-rerun results (CPU vs. CPU and GPU vs. GPU), this is expected, as some mathematical operations have different implementations (e.g. rounding modes) and different precision in a GPU compared with a CPU (Whitehead and Fit-florea, 2011). Fig. 10b shows a qualitative comparison of the reconstruction of six exemplar tracts using both frameworks.

We evaluated the optimal number of CUDA streams (and OpenMP threads) for each test (see Supplementary Fig. 11). The most efficient configurations were 8 CUDA streams for generating dense connectomes, obtaining a total gain of 1.35x respect using 1 CUDA stream, and 4 CUDA streams for reconstructing tracts, obtaining gains of only 1.12x. Fig. 11a reports computation times reconstructing the 12 bilateral tracts and the 3 commissural tracts individually. A single CPU core was used for running the CPU-based framework, and a single GPU and four CUDA streams for processing the tracts with the GPU-accelerated framework. On average, a speedup of 238x was achieved, in the range of 80x to 357x. In all cases, except for the reconstruction of the Acoustic Radiation (AR), the GPU-based application achieves accelerations of more than two orders of magnitude. In general, if the reconstruction of a tract involves several anatomical constraints that makes the algorithm to stop or discard streamlines at early steps, including tight termination and exclusion masks, the GPU-based framework performs worse, as these masks are not checked until the propagation of the streamlines has completely finished (see Supplementary Fig. 6a). The reconstruction of the Acoustic Radiation uses a very tight exclusion mask and thus the achieved performance is lower compared with the reconstruction of other tracts.

Fig. 11b reports the total execution time reconstructing all the tracts. When the CPU-based tool is used, the reconstruction of several tracts can be parallelised. Tracts are completely independent and thus their reconstruction can be processed by different threads. A total of 27 CPU cores were used in this case, using different CPU threads for reconstructing different tracts. A single GPU and four CUDA streams were used

again for processing the tracts with the GPU-accelerated framework, processing sequentially the different tracts. A speedup of 26.5x was achieved using the GPU-accelerated solution.

We use the CPU-based and the GPU-based frameworks for generating a dense connectome. 91,282 seed points and 10,000 samples per seed point were employed, having a total of 912.82 million streamlines. For generating the connectome with the CPU-based application we used 100 CPU cores, each one propagating 100 streamlines from each seed point. This process took on average 3.38 h. At the end of the process, the different generated connectomes (on different CPU cores) need to be added. This merging process took on average 6.1 h (due to the size of the final connectivity matrices).

We used 1 single GPU and 8 CUDA streams for generating the connectome with the GPU-based application. The process took on average 2.95 h. Fig. 11c reports these execution times and the speedup achieved by the GPU-based framework, with and without considering the merging process required by the CPU multi-core application. Without considering the merging process both applications reported similar execution times. Considering the merging process, the GPU application was more than three times faster than the CPU multi-core application.

Apart from the computational benefits, we have added new functionality to the GPU tractography toolbox. A novel feature is the possibility of using surfaces for imposing more accurate anatomical constraints. For instance, we can use a pial surface as termination mask for avoiding the propagation of streamlines outside this surface, and avoiding non-plausible connections along the CSF in the subarachnoid space. As shown in the results of Fig. 12a, surfaces allow us to describe more accurately the cortical folding patterns and allow more accurate constraints to be imposed.

A more sophisticated termination mask mechanism has also been added to the GPU framework. Commonly, termination masks force the algorithm to stop the propagation of the streamlines the first time they hit the mask, but sometimes it is reasonable to allow the propagation until certain conditions are met (see Fig. 12b). For instance, to increase the chances of getting “direct” connections, it is desired that a streamline crosses the WM/GM boundary no more than twice when reconstructing cortico-cortical networks, once at the starting point and once at the end point. However, it does not seem plausible to have pathways running in and out of the WM/GM boundary or in parallel along the cortex connecting several regions. Thus, a special type of termination masks can be

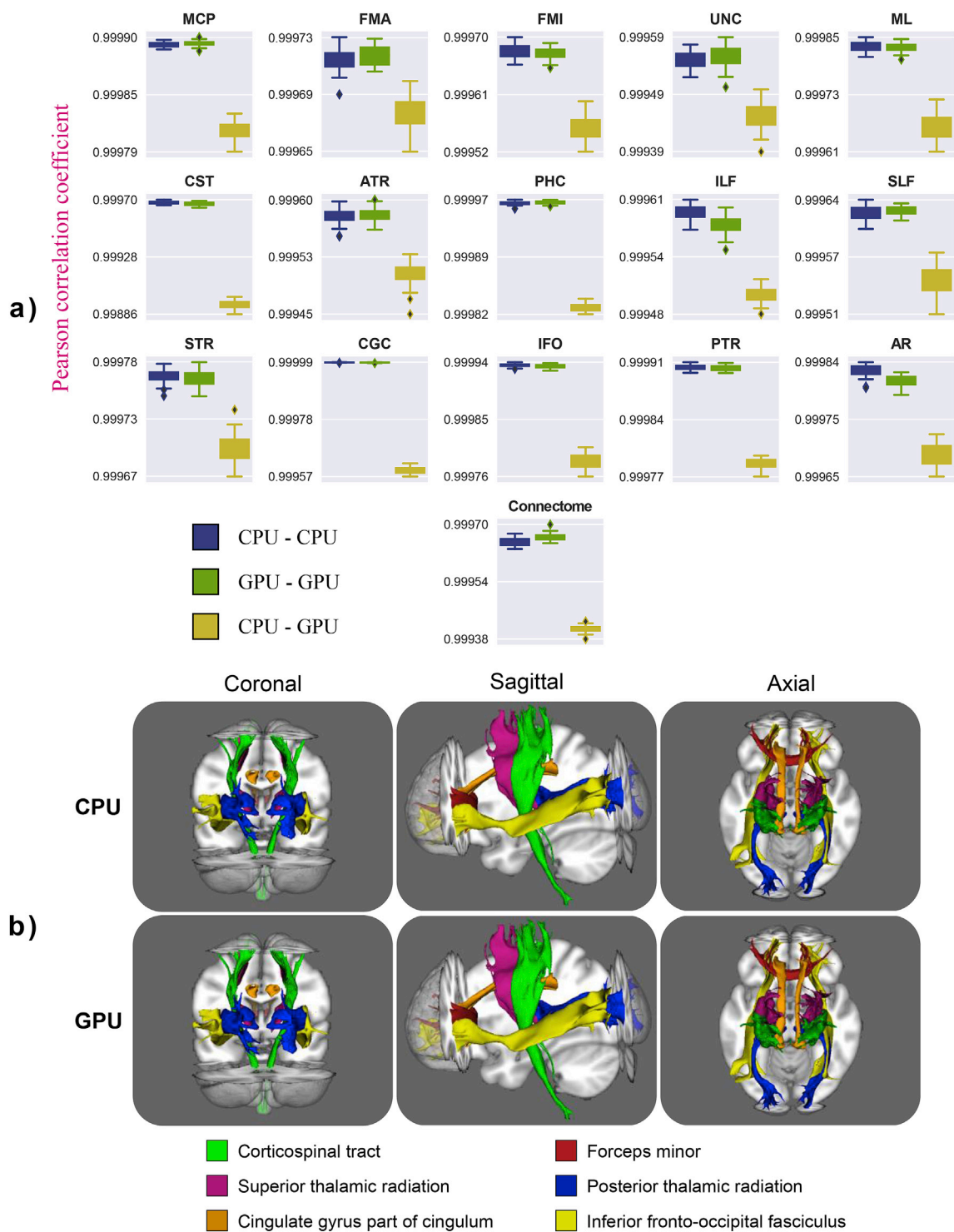


Fig. 10. (a) Run-rerun variability of CPU-based and GPU-based probabilistic tractography frameworks and distribution of the correlation coefficients between both frameworks. Results are showed in the reconstruction of 12 bilateral tracts, 3 commissural tracts and in the generation of a dense connectome. Each experiment was run 10 times. The 45 combinations of correlations between re-runs were considered and 45 out of the 100 combinations of CPU vs. GPU correlation coefficients were chosen randomly. (b) Coronal, sagittal and axial views comparing CPU-based and GPU-based frameworks performing probabilistic tractography and reconstructing some major white matter tracts. Each colour represents a different white matter tract. These paths are binarised versions of the path distributions after being thresholded at 0.5%.

used for stopping the streamlines when they cross a surface twice. Similarly, to encourage direct cortico-subcortical connections, it is undesirable that a streamline visits several subcortical regions, but ideally we would like a streamline to be able to propagate within a subcortical region. As in (Smith et al., 2012), our framework can use the special

termination masks for stopping the streamlines upon exiting these regions, while allowing propagation within them. Fig. 12c shows the effect of imposing these anatomical constraints when generating a dense connectome. The special termination mask is defined with a WM/GM boundary surface, and it also includes several subcortical structures

Table 3

List of reconstructed tracts sorted by number of propagated streamlines. Different number of seed points and samples are used for reconstructing the tracts. Some tracts have a bilateral homologue (+) and some others no (–).

Tract Name	Acronymic	Number of seeds	Samples per seed	Number of streamlines	Left/Right
Uncinate fasciculus	UNC	1692	1200	2,030,400	+
Medial lemniscus	ML	1926	1200	2,311,200	+
Corticospinal tract	CST	723	4000	2,892,000	+
Anterior thalamic Radiation	ATR	3181	1000	3,181,000	+
Parahippocampal part of cingulum	PHC	1887	3000	5,661,000	+
Middle cerebellar peduncle	MCP	2075	4400	9,130,000	–
Forceps major	FMA	18,159	600	10,895,400	–
Inferior longitudinal fasciculus	ILF	9207	1200	11,048,400	+
Forceps minor	FMI	19,195	600	11,517,000	–
Superior longitudinal fasciculus	SLF	32,831	400	13,132,400	+
Superior thalamic radiation	STR	21,019	800	16,815,200	+
Cingulate gyrus part of cingulum	CGC	1137	20,000	22,740,000	+
Inferior fronto-occipital fasciculus	IFO	15,412	4400	67,812,800	+
Posterior thalamic radiation	PTR	3669	20,000	73,380,000	+
Acoustic radiation	AR	23,105	10,000	231,050,000	+

(accumbens, amygdala, caudate, cerebellum, hippocampus, pallidus, putamen and thalamus). The connectivity pattern from the sensori-motor part of the thalamus without and with advanced termination masks is illustrated. In the former case, the streamlines can cross the cortex or subcortical structures several times and continue propagating, generating a number of false positives (for instance see hotspots along the frontal medial surface). In the latter case, this situation is avoided, and a more realistic connectivity map is obtained, connecting the sensorimotor part of the thalamus to sensorimotor cortical regions.

Given the speed and facility to run-rerun probabilistic tractography using the developed GPU toolbox, we performed a convergence study. We evaluated the number of samples that are needed per seed point when generating a dense connectome in order to achieve convergence. To do that, we generated a dense connectome multiple times using a different number of samples per seed point. Fig. 13 shows the correlation coefficients with respect to an assumed converged dense connectome, which was generated using 100,000 samples per seed. The figure also shows the correlation coefficients between consecutive runs in terms of number of samples per seed. It seems that even with 1000 samples and 18 min run, the results are almost converged. Using 10,000 samples per seed achieves convergence, while the time for generating the connectome is still reasonable, less than 3 h using a single GPU.

4. Discussion

We have presented GPU-based parallel computational frameworks for accelerating the analysis of diffusion MRI, spanning from voxel-wise biophysical model fitting to tractography and whole-brain connectome generation. Despite the difference in the inherent parallelisability of these applications, GPUs can offer considerable benefits when challenges are carefully considered. Performances similar to 200 CPU cores were achieved using a single GPU, which change the perspective of what is

computationally feasible. The GPU toolboxes will be publically released as part of FMRIB's Software Library (FSL).

The accelerations achieved by the designs proposed here can be tremendously beneficial. Big databases arise more and more often from large consortiums and cornerstone projects worldwide. Hundreds or even thousands of datasets need to be processed. The throughput of the parallel designs using a single or a multi-GPU system is higher than a CPU multi-core system. Very large recent studies such as the Human Connectome Project (HCP) (Van Essen and Ugurbil, 2012; Van Essen et al., 2012; Sotiropoulos et al., 2013), (data from 1200 adults), the Developing Human Connectome Project (dHCP) (data from 1000 babies) and UK Biobank (Miller et al., 2016; Alfaro-Almagro et al., 2018) (data from 100,000 adults) are using our parallel designs for processing these datasets on GPU clusters. For instance, a 10-GPU cluster has been built for processing the most computationally expensive tasks of the UK Biobank pipeline. The cluster allows fitting the ball & sticks model to 415 datasets per day. Running the same tasks with a cluster of 100 CPU cores, only 25 datasets could have been processed per day. Moreover, to obtain a similar throughput as the 10-GPU cluster, more than 1600 CPU cores would have been necessary. Nevertheless, there are cloud computing platforms that provide on-demand computational resources, including GPUs. Recent studies have presented the pros and cons of using these services for running neuroimaging applications, including cost comparisons (Madhyastha et al., 2017).

We made a price-performance comparison between the multi-CPU and single-GPU configurations, i.e. assessed the relative performance gains of the GPU designs per unit cost. Indicative costs are detailed in Supplementary Table 1, suggesting a price of ~£10800 for 72 CPU cores and £4960 for a dual GPU (K80). It should be noted that these prices reflect a GPU and CPU model and can change depending on choice and generations. They however reflect reasonably the current costs. On average, cuDIMOT on a single GPU (and single CPU core) was 4.3x faster than 72 CPU cores. Thus, in terms of price-performance ratio the parallel solution on a single GPU offers 17.6 times better speedup/pound than the 72 CPU core system. For generating a dense connectome, the GPU system had a total cost of £3680 (a single GPU and 8 CPU cores) and was 3.2x faster than 100 CPU cores, offering 13.04 times better speedup/pound.

Apart from increasing feasibility in big imaging data exploration, the designs presented here can assist in model exploration and development, as estimation and testing is an inherent part of model design. Moreover, the response time for analysing a single dataset is dramatically reduced from several hours/days to few minutes using a single GPU, and close to real-time processing could make these methods more appealing for clinical practice.

4.1. GPU-based biophysical modelling

We have presented a generic modelling framework, cuDIMOT, that provides a model-independent front-end, and automatically generates a GPU executable file that includes several fitting routines for user-defined voxel-wise MRI models. Although parallel designs for processing MRI voxel-wise applications are straightforward, for instance creating as many threads as voxels, challenges need to be considered for achieving efficient solutions on GPUs. Here we have proposed a second level of parallelisation where the most expensive within-voxel tasks are distributed amongst threads within a CUDA warp. We have used cuDIMOT to explore diffusion models that characterise fibre orientation dispersion, and we have shown that it can be very useful for exploring, designing and implementing new dMRI protocols and models. It is easy to use and generates very efficient GPU-accelerated solutions.

Some toolboxes with the same purpose as cuDIMOT have been recently presented (Harms et al., 2017; Fick et al., 2018). In (Fick et al., 2018) a python generic toolbox for fitting multi-compartments dMRI models is proposed, but it does not include any parallelisation strategy. In (Harms et al., 2017) a toolbox for parallelising the optimisation routines is proposed. Even if the initial version did not include the option for

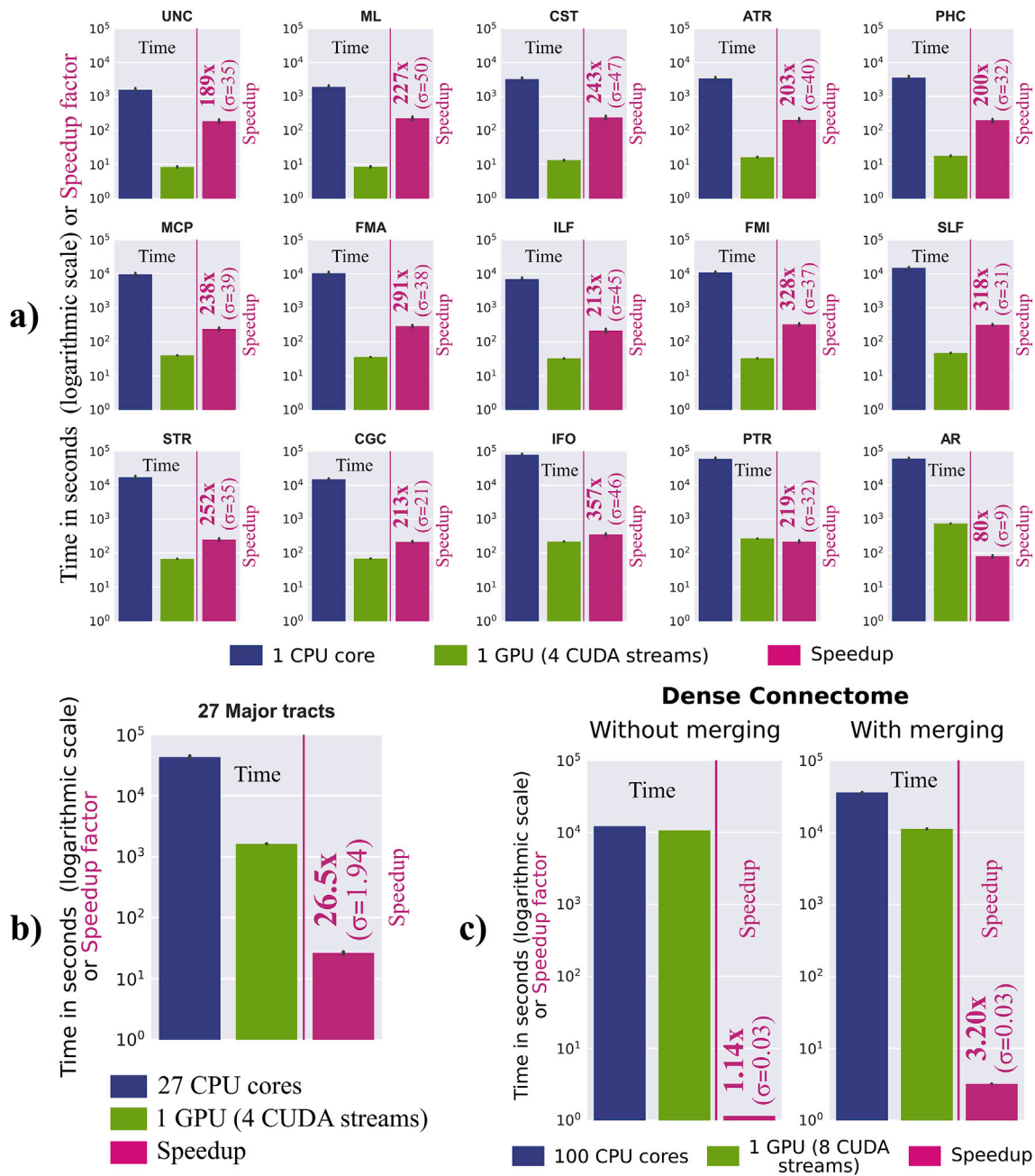


Fig. 11. (a) Execution times (in logarithmic scale) and speedup (standard deviation σ is also shown) in the reconstruction of 12 bilateral tracts and 3 commissural tracts comparing a GPU-based with a CPU-based probabilistic tractography framework. (b) Execution times (in logarithmic scale) and speedup (and its standard deviation σ) reconstructing a total of 27 tracts and (c) generating a dense connectome, comparing a single GPU with several CPU cores.

performing stochastic optimisation or the ability to add priors on the model parameters, an MCMC routine has been very recently included (Harms and Roebroek, 2018). It is implemented in a more generic (non-GPU specific) programming model (OpenCL (Stone et al., 2010) rather than CUDA), allowing the parallelisation on both multi-core CPUs and GPUs, but potentially achieving lower performance on NVIDIA GPUs as some type of instructions can differ in the implementation. For instance, CUDA Shuffle instructions (NVIDIA, 2014a), which are used in cuDIMOT kernels (see implementation details in the [Supplementary material](#)), allow sharing data between threads within a warp and offer performance improvement, but are not supported in OpenCL (Khronos OpenCL Working Group, 2012), where same results must be achieved with slower operations. Therefore, it is expected that CUDA implementations on NVIDIA GPUs will be more efficient than OpenCL

counterparts, this however remains to be tested explicitly for our design. It would be of interest for future work to directly compare the two implementations.

Our framework offers a C-like interface. We are planning as a future extension to design an even more user-friendly Application Programming Interface (API) and a python parser to communicate this API with cuDIMOT.

Our toolbox has been designed for processing datasets with any number of voxels and measurements. If the memory required for storing the data of all the voxels exceeds the device memory, the framework divides the data into several subsets (according to the amount of device memory available), and these subsets are processed one after the other. However, before being processed, each subset needs to be copied into device memory. This can create a performance penalty if the device

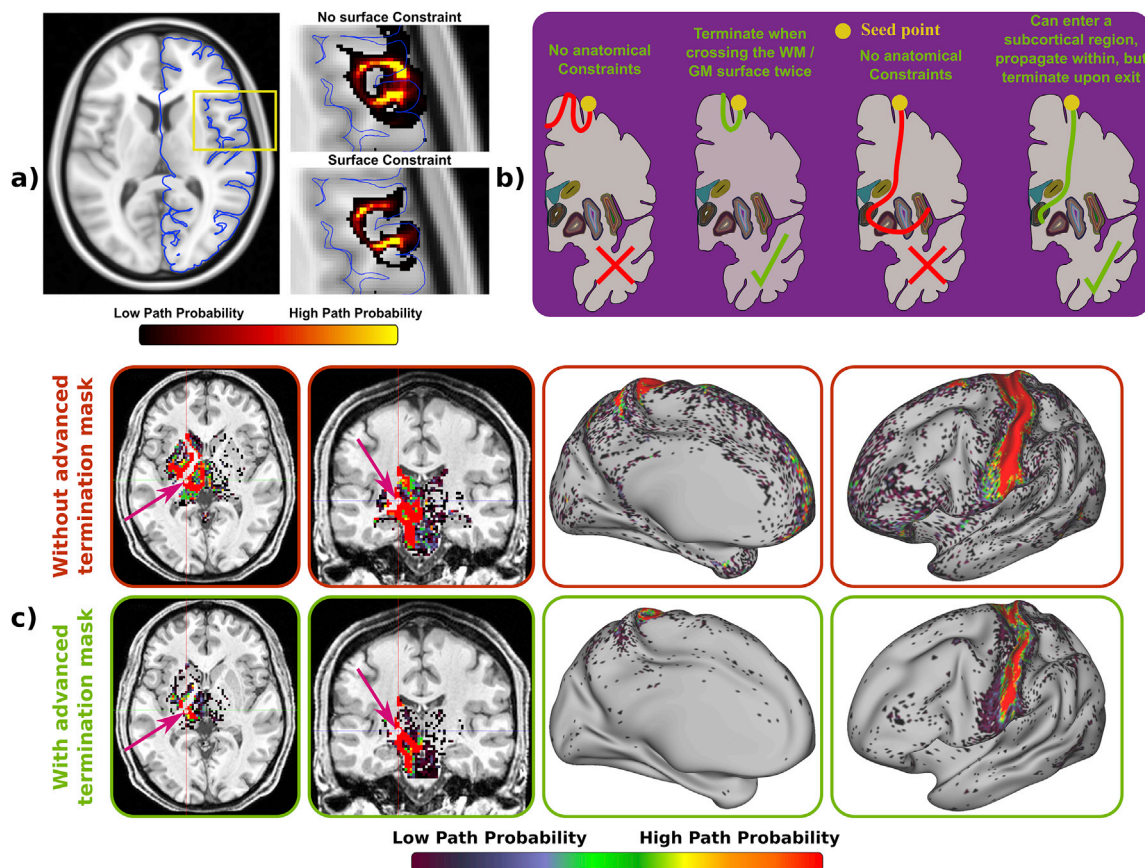


Fig. 12. (a) Example of the use of surfaces for imposing anatomical constraints. Probabilistic tractography is performed using as seed and target points the right inferior frontal gyrus. Without using a surface constraint, wrong paths that jump between neighbouring gyri can be generated. (b) Advanced termination masks. The tractography framework adds the possibility of stopping the streamlines when they cross a surface twice and/or streamlines can be propagated inside a subcortical region but the framework stops them upon exit. (c) Connectivity from a voxel inside the left Thalamus (fuchsia arrow) using and not using advanced terminations masks. The first two columns show the connectivity with other subcortical structures. The last two columns show the connectivity with all the vertices on the left hemisphere cortex.

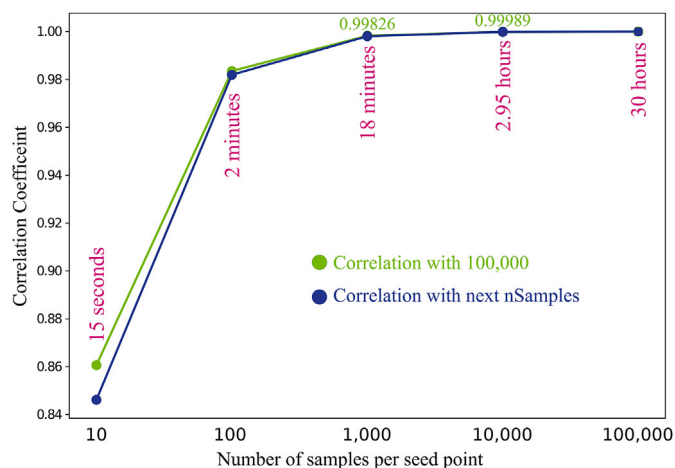


Fig. 13. Correlation coefficient, generating a dense connectome (all greyordinates to all greyordinates) on the GPU-based framework, between re-runs, modifying the number of samples per seed point. The figure reports the correlation coefficients with respect to a dense connectome generated with 100,000 samples (green) and with respect to the connectome generated with the next number of samples in the plot (blue). The figure also shows the execution times generating these dense connectomes on a single NVIDIA K80 GPU.

memory capacity is small (1 or 2 GB) because a large number of CPU to GPU transfers is required, and these transfers are expensive (using the PCIe interconnection bus). This is however not a problem in the new NVIDIA architectures, where global memory space is larger (up to 24 GB in Pascal architecture (NVIDIA, 2016)) and a new CPU-GPU interconnection bus is incorporated (NVLink (NVIDIA, 2014b)).

There is a limitation in cuDIMOT on the number of parameters of a model. In the Levenberg-Marquardt routine the number of model parameters is limited to 31. The cause of this limitation is in the implementation of a LU solver (see Supplementary Fig. 4), where each thread of a warp processes a column of the matrix for solving the system. For a model with P parameters, $P + 1$ threads are required, and a warp has 32 threads. In the MCMC routine there is also a limitation on the number of model parameters. The framework stores in the GPU Shared memory the parameters and some associated information (priors, number of proposals accepted/rejected, and standard deviation of the proposals distribution). Thus, a model is limited to a maximum number of around 300 parameters (the exact number depends on the size of Shared memory of the specific GPU and on the precision used to store the parameters, single or double).

A large number of dMRI modelling approaches have been proposed in the literature, but it seems that no single approach can explain all complex microstructure patterns (Ferizi et al., 2015; Ghosh et al., 2016). Thus, applications that consider several models for selecting the best one in each voxel seems to be a potential solution. Given the computational cost of fitting these models, parallel solutions like cuDIMOT will be essential for performing this type of analysis. We believe that cuDIMOT is

going to be very useful in the development and improvement of new diffusion MRI models, which may explain the complexity of the diffusion process, extract useful biophysical parameters and contribute to the development of new biomarkers.

4.2. GPU-based tractography

We have also developed and presented a probabilistic tractography framework that achieves a higher performance than 200 CPU cores, and can handle situations ranging from simple white matter tracking tasks to dense connectome generation. The implementation offers the possibility of defining tractography protocols with either volumes or surfaces, and the possibility of using advanced termination masks that allow more accurate anatomical constraints. We have shown the benefits of using this extended functionality.

Our GPU framework parallelises a stochastic tractography algorithm, and we have reported the speed-ups achieved by our parallel solution for the number of samples required to achieve convergence in the stochastic estimation (as shown in Fig. 13). Fewer samples would bring the computing times down, for both the sequential and the parallel solution, but the relevant speed-ups would still be applicable.

Tractography algorithms pose particular challenges for designing GPU solutions. Each GPU thread accesses different memory locations during its execution, and these accesses cannot be anticipated, as the propagation movements are decided on the fly. Moreover, given the stochastic nature for choosing the orientation samples, the threads may diverge even if their streamlines are initialised from the same seed point. This behaviour leads to uncoalesced memory accesses and imbalanced execution length of the threads, and consequently, to a waste of GPU resources.

We studied the execution length distribution across streamlines reconstructing the same tract. In many scenarios, many generated streamlines terminate relatively quickly (before 100 steps), as they meet a termination criterion. However, there are other streamlines that take considerably more steps. This trend has also been confirmed before in Mittmann et al. (2008) and Xu et al. (2012), and it is also supported by the underlying anatomy: the majority of white matter connections are short in length (Donahue et al., 2016). To avoid a waste of resources, we explored the approach proposed in (Mittmann et al., 2008; Xu et al., 2012), where the kernel that propagates the streamlines is stopped after a certain number of steps and the threads that are idle, i.e., the threads with terminated streamlines, are removed. When the kernel is launched again on the GPU, there will be only threads with streamlines still propagating, and thus, the device resources will be used more efficiently. This also allows other streamlines to start to be processed, as memory resources are freed after removing terminated streamlines. We tried several strategies for deciding the number of propagation steps to use before stopping the kernel. The process for removing idle threads is executed on the host and it has some extra cost that may cancel out the gains of this approach. Contrary to suggestions in (Mittmann et al., 2008; Xu et al., 2012), where gains of 4x were reported when using streamlines removal strategies, we could not find enough supporting evidence that this approach results in significant performance gains. This strategy barely reduces execution times in our framework, and in some cases, it even increases them. We believe that the extended and more complex functionality offered by our tractography framework, compared with previous designs, is the main reason for these differences. Given the complex functionality, more information needs to be stored and reloaded by the GPU threads every time that the CUDA kernel is stopped, causing a higher overload. These previous studies have reported final performance gains in the range of 40x–50x, which is considerably lower than what we found here.

Another challenging requirement of the parallel tractography application is the large amount of required memory. Given the non-predictable path and length of the streamlines, all the orientation

samples and memory for storing the maximum possible number of visited coordinates need to be allocated. The GPU global memory is used for storing this data. This restricts the number of streamlines that can be propagated in parallel. However, most modern GPUs have at least 3 GB of global memory, and they still can run a considerable number of streamlines (~40,000) in parallel. When a more complex and demanding functionality is used, such as generating a dense connectome, this number can be reduced by 60% (as more data per streamline is needed) and GPUs with at least 5 GB should be used for achieving a good performance.

It should be noted that the strategy used here to generate connectomes might not be necessarily optimal, as a number of open questions remain to be answered when building connectomes using diffusion MRI (see (Sotiropoulos and Zalesky, 2017) for a recent review). Nevertheless, our GPU implementation is flexible for defining a number of strategies (e.g. see Fig. 3) and contributes to research for resolving some of the open challenges by enabling faster explorations.

Acknowledgments

We would like to acknowledge financial support from the UK Engineering and Physical Sciences Research Council (EP/L023067/1). The Wellcome Centre for Integrative Neuroimaging is supported by core funding from the Wellcome Trust [203139/Z/16/Z]. Part of this project was awarded the NVIDIA 2016 GPU centre of excellence achievement, and the prize was used for partially funding this research. We also acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for aspects of development of the presented toolboxes. Data were provided by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University. More details at: <https://www.humanconnectome.org/study/hcp-young-adult/document/hcp-citations>.

Appendix A. Supplementary data

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.neuroimage.2018.12.015>.

References

- Alexander, A.L., et al., 2001. Analysis of partial volume effects in diffusion-tensor MRI. *Magn. Reson. Med.* 45 (5), 770–780.
- Alexander, D.C., et al., 2010. Orientationally invariant indices of axon diameter and density from diffusion MRI. *Neuroimage* 52 (4), 1374–1389.
- Alexander, D.C., et al., 2017. Imaging brain microstructure with diffusion MRI: practicality and applications. *NMR Biomed.* e3841.
- Alfaro-Almagro, F., et al., 2018. Image processing and Quality Control for the first 10,000 brain imaging datasets from UK Biobank. *Neuroimage* 166, 400–424. Elsevier Inc.
- Alsmirat, M.A., et al., 2017. Accelerating compute intensive medical imaging segmentation algorithms using hybrid CPU-GPU implementations. *Multimed. Tool. Appl.* 76 (3), 3537–3555.
- Andersson, J.L.R., Jenkinson, M., Smith, S., 2007. Non-linear Registration, Aka Spatial Normalisation. *FMRIB Technical Report TR07JA2*.
- Assaf, Y., et al., 2008. AxCaliber: a method for measuring axon diameter distribution from diffusion MRI. *Magn. Reson. Med.* 59 (6), 1347–1354.
- Assaf, Y., Cohen, Y., 1998. Non-mono-exponential attenuation of water and N-acetyl aspartate signals due to diffusion in brain tissue. *J. Magn. Reson.* 131 (1), 69–85.
- Auton, A., et al., 2015. A global reference for human genetic variation. *Nature* 526 (7571), 68.
- Basser, P.J., et al., 2000. In vivo fiber tractography using DT-MRI data. *Magn. Reson. Med.* 44 (4), 625–632.
- Basser, P.J., Mattiello, J., LeBihan, D., 1994a. Estimation of the effective self-diffusion tensor from the NMR spin echo. *J. Magn. Reson., Ser. B* 247–254.
- Basser, P.J., Mattiello, J., LeBihan, D., 1994b. MR diffusion tensor spectroscopy and imaging. *Biophys. J.* 66 (1), 259–267.
- Behrens, T.E.J., et al., 2003. Characterization and propagation of uncertainty in diffusion-weighted MR imaging. *Magn. Reson. Med.* 50 (5), 1077–1088.
- Behrens, T.E.J., et al., 2007. Probabilistic diffusion tractography with multiple fibre orientations: what can we gain? *Neuroimage* 34 (1), 144–155.

- Chang, L.C., et al., 2014. GPU acceleration of nonlinear diffusion tensor estimation using CUDA and MPI. *Neurocomputing* 135, 328–338. Elsevier.
- Chapman, B., Jost, G., Pas, R. Van Der, 2008. Using OpenMP: Portable Shared Memory Parallel Programming. The MIT press.
- Daducci, A., et al., 2015. Accelerated microstructure imaging via convex optimization (AMICO) from diffusion MRI data. *Neuroimage* 105, 32–44.
- Deoni, S.C.L., 2010. Quantitative relaxometry of the brain. *Top. Magn. Reson. Imag.*: *TMRI* 21 (2), 101–113.
- Donahue, C.J., et al., 2016. Using diffusion tractography to predict cortical connection strength and distance: a quantitative comparison with tracers in the monkey. *J. Neurosci.* 36 (25), 6758–6770.
- Eklund, A., et al., 2013. Medical image processing on the GPU - past, present and future. *Med. Image Anal.* 17 (8), 1073–1094.
- Eklund, A., et al., 2014. BROCCOLI: software for fast fMRI analysis on many-core CPUs and GPUs. *Front. Neuroinf.* 8.
- Van Essen, D.C., et al., 2012. The Human Connectome Project: a data acquisition perspective. *Neuroimage* 62 (4), 2222–2231.
- Van Essen, D.C., Ugurbil, K., 2012. The future of the human connectome. *Neuroimage* 62 (2), 1299–1310. Elsevier Inc.
- Perizi, U., et al., 2015. White matter compartment models for in vivo diffusion MRI at 300mT/m. *Neuroimage* 118, 468–483.
- Fick, R., Wassermann, D., Deriche, R., 2018. Mipy: an open-source framework to improve reproducibility in brain microstructure imaging. In: Annual Meeting of the Organization for Human Brain Mapping.
- Flynn, M.J., 1972. Some computer organizations and their effectiveness. *IEEE Trans. Comput.* 100 (9), 948–960.
- Foxley, S., et al., 2015. Improved tract identification of post-mortem human brain with high-resolution DTI at 7T. In: Annual Meeting of the Organization for Human Brain Mapping.
- Foxley, S., et al., 2016. A comparison of multiple acquisition strategies to overcome B1 inhomogeneities in diffusion imaging of post-mortem human brain at 7T. In: International Society for Magnetic Resonance in Medicine 24th Annual Meeting.
- Ghosh, A., Alexander, D., Zhang, H., 2016. Crossing versus fanning: model comparison using HCP data. In: Computational Diffusion MRI. Springer, pp. 159–169.
- Gill, P.E., et al., 1984. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Trans. Math Software* 10 (3), 282–298.
- Glasser, M.F., et al., 2013. The minimal preprocessing pipelines for the Human Connectome Project. *Neuroimage* 80, 105–124.
- de Groot, M., et al., 2013. Improving alignment in Tract-based spatial statistics: evaluation and optimization of image registration. *Neuroimage* 76, 400–411.
- Harms, R.L., et al., 2017. Robust and fast nonlinear optimization of diffusion MRI microstructure models. *Neuroimage* 155, 82–96. Elsevier.
- Harms, R.L., Roebroeck, A., 2018. Robust and Fast Monte Carlo Markov Chain Sampling of Diffusion MRI Microstructure Models bioRxiv.
- Harwell, J., et al., 2008. 'GIFTI': geometry data format for exchange of surface-based brain mapping data'. In: Annual Meeting of the Organization for Human Brain Mapping.
- Hernandez-Fernandez, M., et al., 2016. A fast and flexible toolbox for tracking brain connections in diffusion MRI datasets using GPUs. In: Annual Meeting of the Organization for Human Brain Mapping.
- Hernández, M., et al., 2013. Accelerating fibre orientation estimation from diffusion weighted magnetic resonance imaging using GPUs. *PLoS One* 8 (4), e61892.
- Jbabdi, S., et al., 2012. Model-based analysis of multishell diffusion MR data for tractography: how to get over fitting problems. *Magn. Reson. Med.* 68 (6), 1846–1855.
- Jenkinson, M., et al., 2012. Fsl. *Neuroimage* 62 (2), 782–790.
- Jeurissen, B., et al., 2017. Diffusion MRI fiber tractography of the brain. *NMR Biomed.* e3785.
- Johansen-Berg, H., et al., 2004. Changes in connectivity profiles define functionally distinct regions in human medial frontal cortex. *Proc. Natl. Acad. Sci. Unit. States Am.* 101 (36), 13335–13340.
- Kelley, C.T., 1999. Iterative Methods for Optimization. Society for Industrial and Applied Mathematics, Siam.
- Khronos OpenCL Working Group, 2012. The OpenCL Specification Version 1.2.
- Klus, P., et al., 2012. BarraCUDA-a fast short read sequence aligner using graphics processing units. *BMC Res. Notes* 5 (1), 27.
- Koev, P., Edelman, A., 2006. The efficient evaluation of the hypergeometric function of a matrix argument. *Math. Comput.* 75 (254), 833–846.
- Kume, A., Wood, A.T.A., 2005. Saddlepoint approximations for the Bingham and Fisher-Bingham normalising constants. *Biometrika* 92 (2), 465–476.
- Li, L., et al., 2012. The effects of connection reconstruction method on the interregional connectivity of brain networks via diffusion tractography. *Hum. Brain Mapp.* 33 (8), 1894–1913.
- MacKay, D.J.C., 1995. Developments in probabilistic modelling with neural networks - ensemble learning. In: Neural Networks: Artificial Intelligence and Industrial Applications. Springer, pp. 191–198.
- Madhyastha, T.M., et al., 2017. Running neuroimaging applications on amazon web services: how, when, and at what cost? *Front. Neuroinf.* 11.
- McNab, J.A., Miller, K.L., 2008. Sensitivity of diffusion weighted steady state free precession to anisotropic diffusion. *Magn. Reson. Med.* 60 (2), 405–413.
- Microstructure Imaging Group - University College London, 2017. NODDI Matlab toolbox. Available at: <http://mig.cs.ucl.ac.uk/index.php?n=Tutorial.NODDIMatlab>.
- Miller, K.L., et al., 2016. Multimodal population brain imaging in the UK Biobank prospective epidemiological study. *Nat. Neurosci.* 19 (11), 1523–1536.
- Mittmann, A., Comunello, E., von Wangenheim, A., 2008. Diffusion tensor fiber tracking on graphics processing units. *Comput. Med. Imag. Graph.* 32 (7), 521–530.
- Moeller, S., et al., 2010. Multiband multislice GE-EPI at 7 tesla, with 16-fold acceleration using partial parallel imaging with application to high spatial and temporal whole-brain fMRI. *Magn. Reson. Med.* 63 (5), 1144–1153.
- Motulsky, H.J., Ransnas, L. a., 1987. Fitting curves to data using nonlinear regression: a practical and nonmathematical review. *Faseb. J.* 1 (5), 365–374.
- Mulkern, R.V., et al., 1999. Multi-component apparent diffusion coefficients in human brain. *NMR Biomed.* 12 (1), 51–62.
- Nickolls, J., et al., 2008. Scalable Parallel Programming with CUDA. *ACM QUEUE*, pp. 40–53 (March/April).
- Niendorf, T., et al., 1996. Biexponential diffusion attenuation in various states of brain tissue: implications for diffusion-weighted imaging. *Magn. Reson. Med.* 36 (6), 847–857.
- NVIDIA, 2014a. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110/210.
- NVIDIA, 2014b. NVIDIA NVLink High-speed Interconnect: Application Performance.
- NVIDIA, 2015a. Cuda C Programming Guide v7, vol. 5.
- NVIDIA, 2015b. TESLA K80. GPU ACCELERATOR, Board Specification.
- NVIDIA, 2016. NVIDIA Tesla P100 Whitepaper. The Most Advanced Datacenter Accelerator Ever Built. Featuring Pascal GP100, the World's Fastest GPU.
- NVIDIA, 2017. NVIDIA TESLA V100 GPU ARCHITECTURE: the World's Most Advanced Data Center GPU.
- O'Rourke, J., 1998. Search and intersection. In: Computational Geometry in C. Cambridge university press.
- Pierpaoli, C., et al., 2001. Water diffusion changes in wallerian degeneration and their dependence on white matter architecture. *Neuroimage* 13 (6), 1174–1185.
- Poupon, C., et al., 2000. Regularization of diffusion-based direction maps for the tracking of brain white matter fascicles. *Neuroimage* 12 (2), 184–195.
- Press, W., et al., 1988. Numerical Recipes in C: the Art of Scientific Computing. Cambridge University Press.
- Schmidhuber, J., 2015. Deep Learning in neural networks: an overview. *Neural Network.* 61, 85–117.
- Setsonpop, K., et al., 2012. Blipped-controlled aliasing in parallel imaging for simultaneous multislice echo planar imaging with reduced g-factor penalty. *Magn. Reson. Med.* 67 (5), 1210–1224.
- Setsonpop, K., et al., 2018. High-resolution in vivo diffusion imaging of the human brain with generalized slice dithered enhanced resolution: simultaneous multislice (gSlider-SMS). *Magn. Reson. Med.* 79, 141–151.
- Seunarine, K.K., Alexander, D.C., 2014. Multiple fibers: beyond the diffusion tensor. In: Diffusion MRI: from Quantitative Measurement to in Vivo Neuroanatomy. Academic Press, pp. 105–123.
- Shamonin, D., 2014. Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease. *Front. Neuroinf.* 7.
- Smistad, E., et al., 2015. Medical image segmentation on GPUs - a comprehensive review. *Med. Image Anal.* 20 (1), 1–18.
- Smith, R.E., et al., 2012. Anatomically-constrained tractography: improved diffusion MRI streamlines tractography through effective use of anatomical information. *Neuroimage* 62 (3), 1924–1938.
- Smith, S.M., 2002. Fast robust automated brain extraction. *Hum. Brain Mapp.* 17 (3), 143–155.
- Smith, S.M., et al., 2004. Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage* 23, S208–S219.
- Sotiropoulos, S.N., et al., 2013. Advances in diffusion MRI acquisition and processing in the human connectome project. *Neuroimage* 80, 125–143.
- Sotiropoulos, S.N., Behrens, T.E.J., Jbabdi, S., 2012. Ball and rackets: inferring fiber fanning from diffusion-weighted MRI. *Neuroimage* 60 (2), 1412–1425.
- Sotiropoulos, S.N., Zalesky, A., 2017. Building connectomes using diffusion MRI: why, how and but. *NMR Biomed.* e3752.
- Sporns, O., Tononi, G., Kötter, R., 2005. The human connectome: a structural description of the human brain. *PLoS Comput. Biol.* 1 (4), e42.
- Stanisz, G.J., et al., 1997. An analytical model of restricted diffusion in bovine optic nerve. *Magn. Reson. Med.* 37 (1), 103–111.
- Stone, E., Gohara, D., Shi, G., 2010. OpenCL: a parallel programming standard for heterogeneous computing systems. *Comput. Sci. Eng.* 12 (3), 66–73.
- Stone, S.S., et al., 2008. Accelerating advanced MRI reconstructions on GPUs. In: Proceedings of the 5th Conference on Computing Frontiers, pp. 261–272.
- Sudmant, P.H., et al., 2015. An integrated map of structural variation in 2,504 human genomes. *Nature* 526 (7571), 75–81.
- Szafer, A., et al., 1995. Diffusion-weighted imaging in tissues: theoretical models. *NMR Biomed.* 8 (7), 289–296.
- Tarantola, A., 2005. Inverse Problem Theory and Methods for Model Parameter Estimation. SIAM.
- Tariq, M., et al., 2016. Bingham-NODDI: mapping anisotropic orientation dispersion of neurites using diffusion MRI. *Neuroimage* 133, 207–223.
- Tendler, B., et al., 2018. Development of a diffusion-weighted SSFP acquisition and processing pipeline to quantify the diffusion properties of the post-mortem ALS brain at 7T. In: International Society for Magnetic Resonance in Medicine 27th Annual Meeting.
- Uecker, M., et al., 2015. Berkeley advanced reconstruction toolbox. In: Proceedings of the International Society for Magnetic Resonance in Medicine.
- Vu, A.T., et al., 2015. High resolution whole brain diffusion imaging at 7T for the Human Connectome Project. *Neuroimage* 122, 318–331.

- Whitehead, N., Fit-Florea, A., 2011. Precision & Performance : Floating Point and IEEE 754 Compliance for NVIDIA GPUs.
- Wiegell, M.R., Larsson, H.B.W., Wedeen, V.J., 2000. Fiber crossing in human brain depicted with diffusion tensor MR imaging. *Radiology* 217 (3), 897–903.
- Xu, M., et al., 2012. Probabilistic brain fiber tractography on GPUs. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012, pp. 742–751.
- Zhang, H., et al., 2011. Axon diameter mapping in the presence of orientation dispersion with diffusion MRI. *Neuroimage* 56 (3), 1301–1315.
- Zhang, H., et al., 2012. NODDI: practical in vivo neurite orientation dispersion and density imaging of the human brain. *Neuroimage* 61 (4), 1000–1016.
- Zhang, Y., Brady, M., Smith, S., 2001. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE Trans. Med. Imag.* 20 (1), 45–57.