# A fast and flexible toolbox for tracking brain connections in diffusion MRI datasets using GPUs - 4258

Moisés Hernández-Fernández[1], Istvan Reguly[2,3], Mike Giles[2], Saad Jbabdi[1], Stephen Smith[1] and Stamatios N. Sotiropoulos[1]

[1]FMRIB Centre, University of Oxford, UK [2]Oxford e-Research Centre, University of Oxford, UK
[3]Faculty of Information Technology and Bionics, Pazmany Peter Catholic University, Hungary
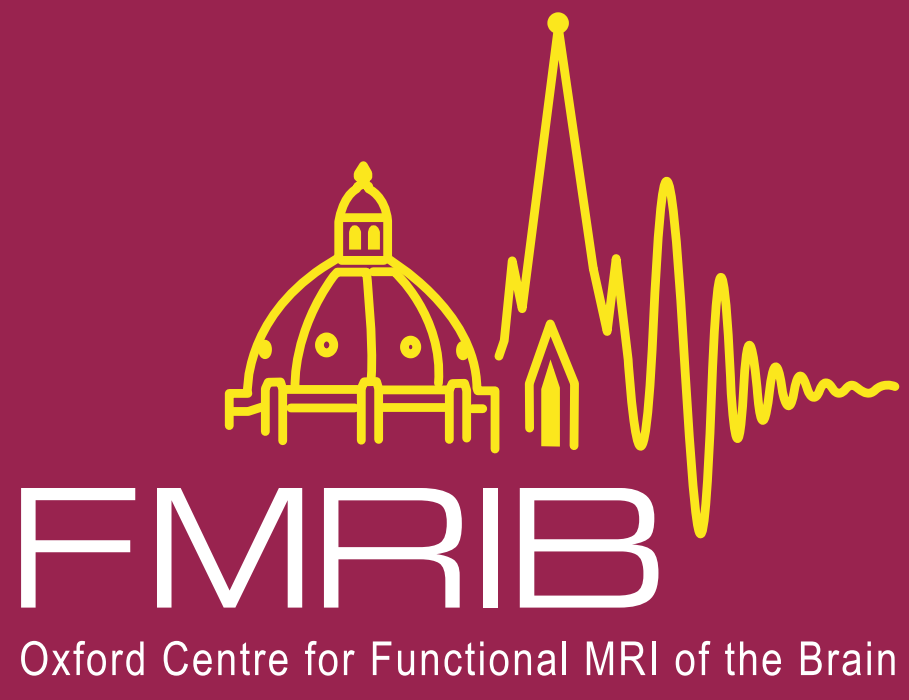
1. Introduction: A GPU based tractography toolbox that reduces computation times by two orders of magnitude

Tractography methods have great potential for mapping brain connections, however, they can be very time-consuming and restrict the potential of the technology. This is particularly true when probabilistic streamline tractography is used for:
- Processing databases that comprise **hundreds** [1,2] to **tens of thousands of subjects** [3], and/or
- Performing **whole-brain** exploratory analysis and building **connectomes** [4]

We present a **parallel algorithm** that performs **probabilistic streamline tractography** using Graphics Processing Units (**GPUs**) and **reduces computation** times by **two orders of magnitude**. The toolbox will be publically available in the **next version of FSL**.

## 2. Parallel tractography algorithm: challenges and limitations

The algorithm implements probabilistic streamline tractography using the **CUDA** parallel programming model [5] and generates spatial distributions of streamlines from seed points. Thousands of streamlines can be computed in parallel on a single GPU.

Parallelisability requirements for obtaining a good *performance* on GPUs:
1. Problem must be divisible into thousands of simple sub-problems.
2. Data Level Parallelism (DLP): Same instructions over multiple data sets.
3. Regular memory access patterns. Threads accessing continuous memory positions.
4. Semi-independence of threads. Little communication/synchronisation.

Parallelising probabilistic tractography algorithms does not inherently fulfil these requirements.

→ GPU implementation is not straightforward:
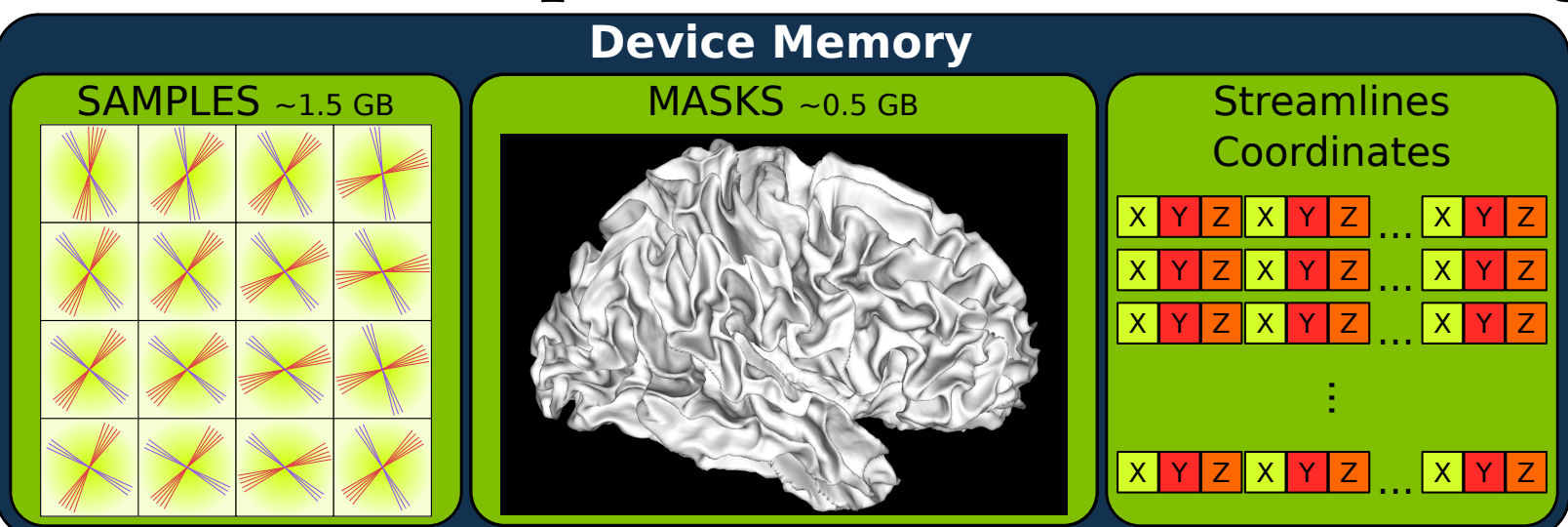
- Limitation on the number of streamlines that can be computed in parallel: high memory demands. All data need to be in the device memories for online calculations.
- <u>Too heavy Tasks</u>: each streamline needs to compute complex tasks, including a segment-triangle intersection algorithm [6]. Light-weight threads are more suitable on GPUs.
- <u>Divergent behaviour</u> of the threads: different paths and different number of iterations. Irregular memory accesses patterns.
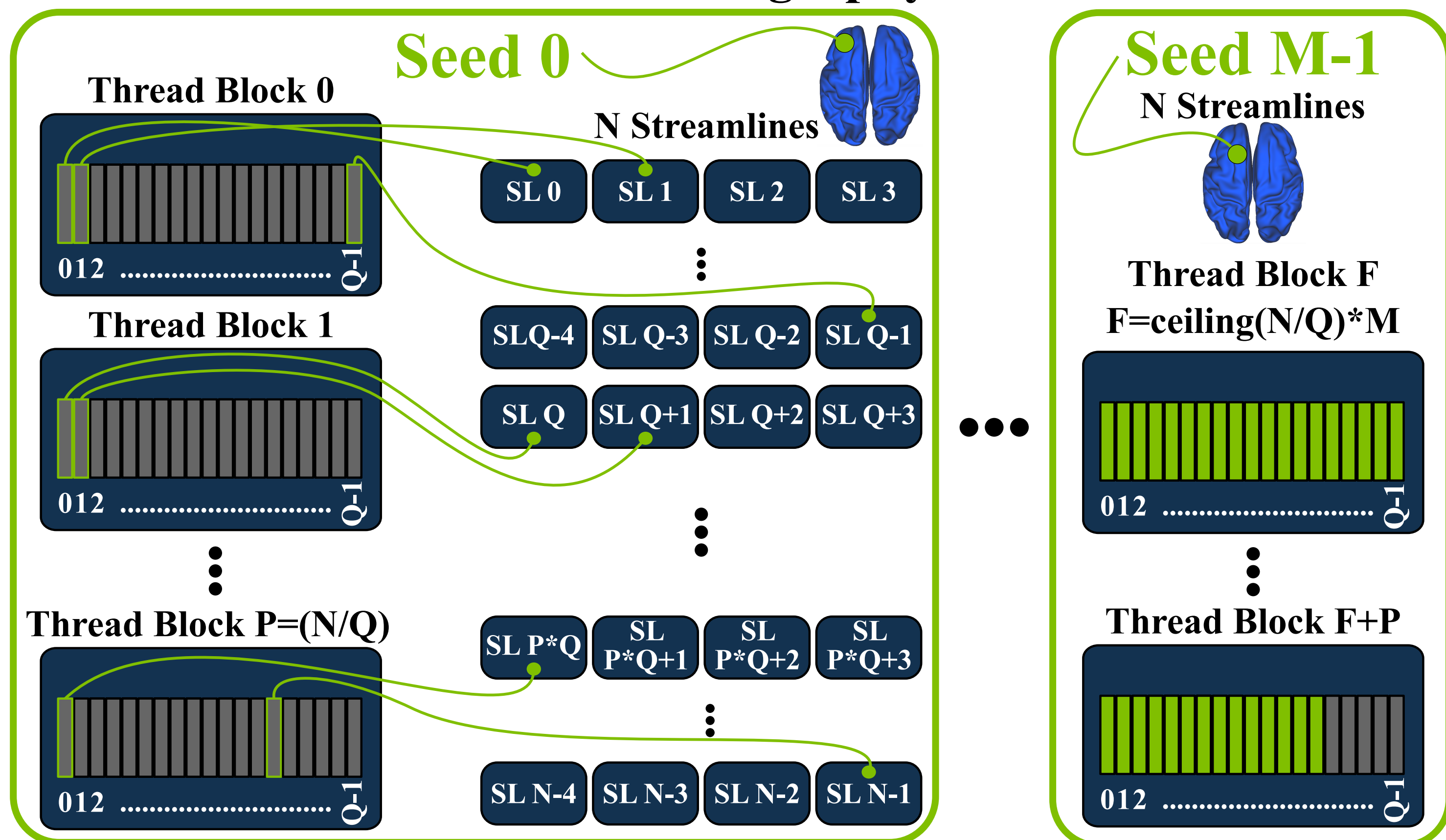- A lot of <u>memory transfers</u>: need to copy results from GPU to CPU.
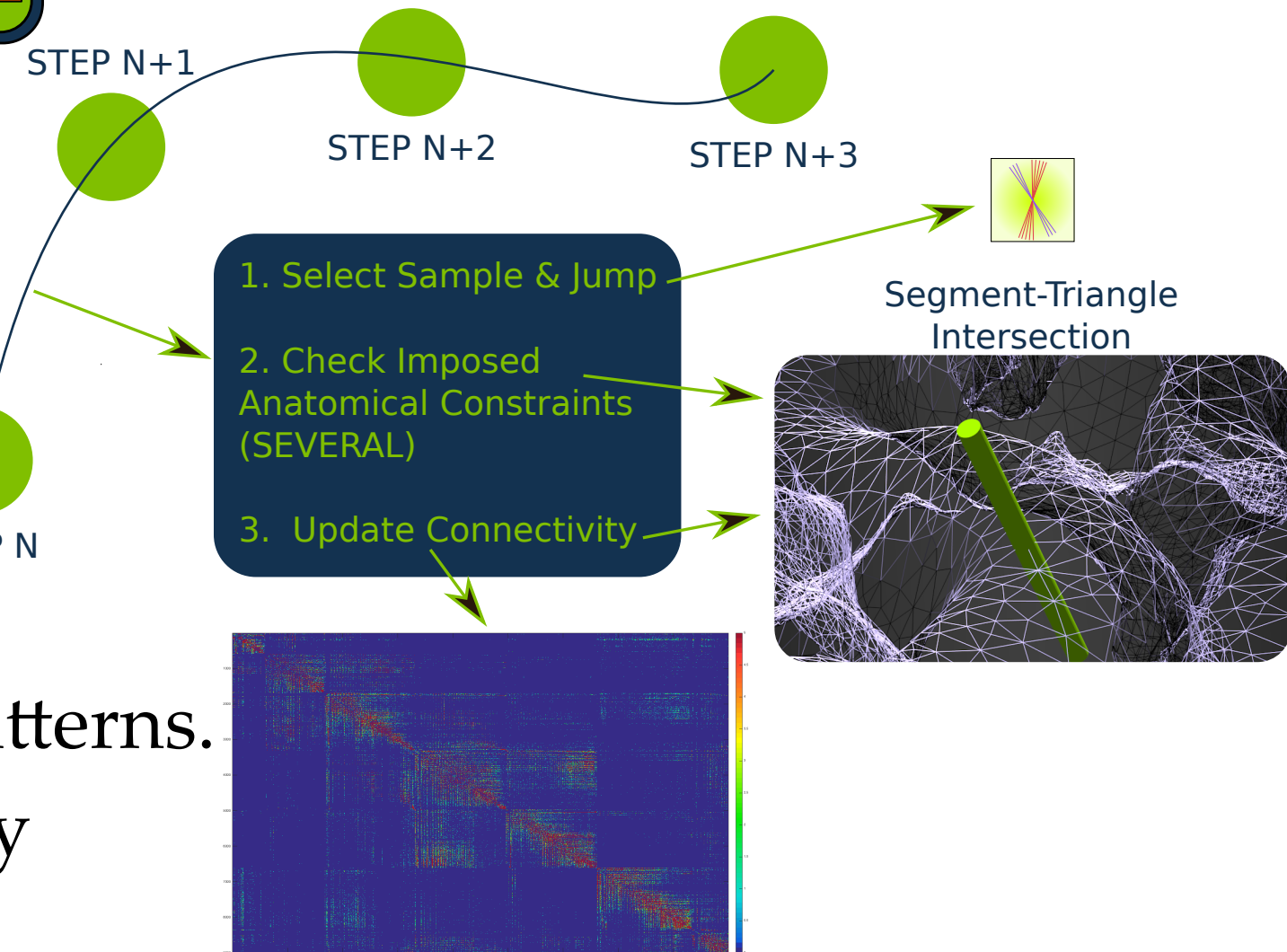
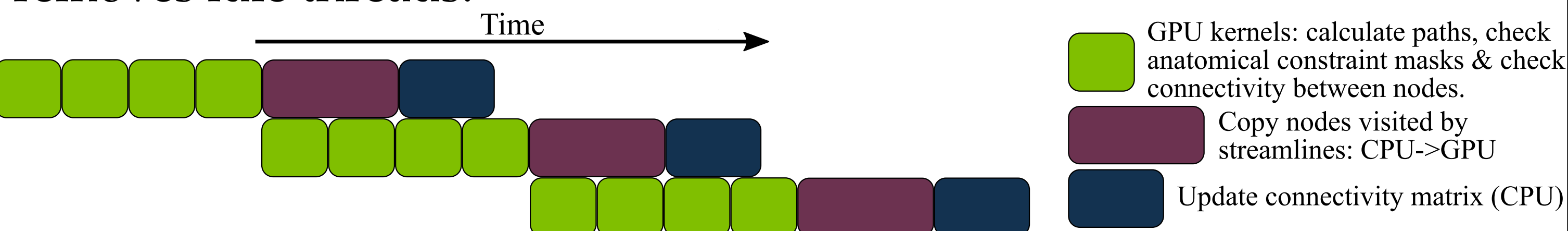## 3. Design and implementation of parallel tractography algorithm

To reduce divergences of threads, streamlines from different seeds can not be assigned to the same Thread Block (Group of threads).

Probabilistic Tractography on GPUs

**MxN streamlines (SLs) are launched from M seeds.
Each SL is assigned to a thread.**

The parallel algorithm reorganises dynamically the threads, in such a way that threads within same warp (group of 32 threads) access similar memory positions most of the time (to rely on efficient caching), and removes idle threads.

GPU kernels: calculate paths, check anatomical constraint masks & check connectivity between nodes.
Copy nodes visited by streamlines: CPU->GPU
Update connectivity matrix (CPU)

The complex tasks of a streamline are divided into simple CUDA kernels. The parallel algorithm iterates over groups of streamlines. Some tasks are assigned to the CPU. GPU & CPU processes and Memory transfers are overlapped.

4. New features of the algorithm

All the **functionality of FSL**'s respective CPU tractography tool [7]. Can handle both volume and surface (**NIFTI and GIFTI**) files. The toolbox offers novel flexibility and functionality features:

An option to use **3D surfaces** (GIFTI files) for imposing more realistic **anatomical constraints** (for instance using the WM/GM boundary surface as a termination mask).
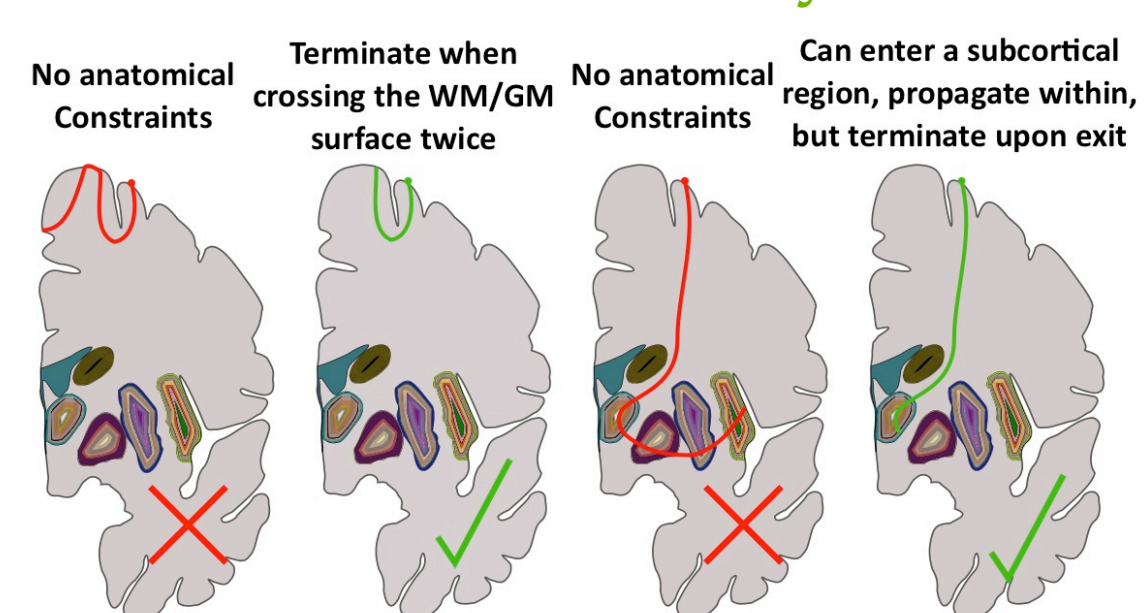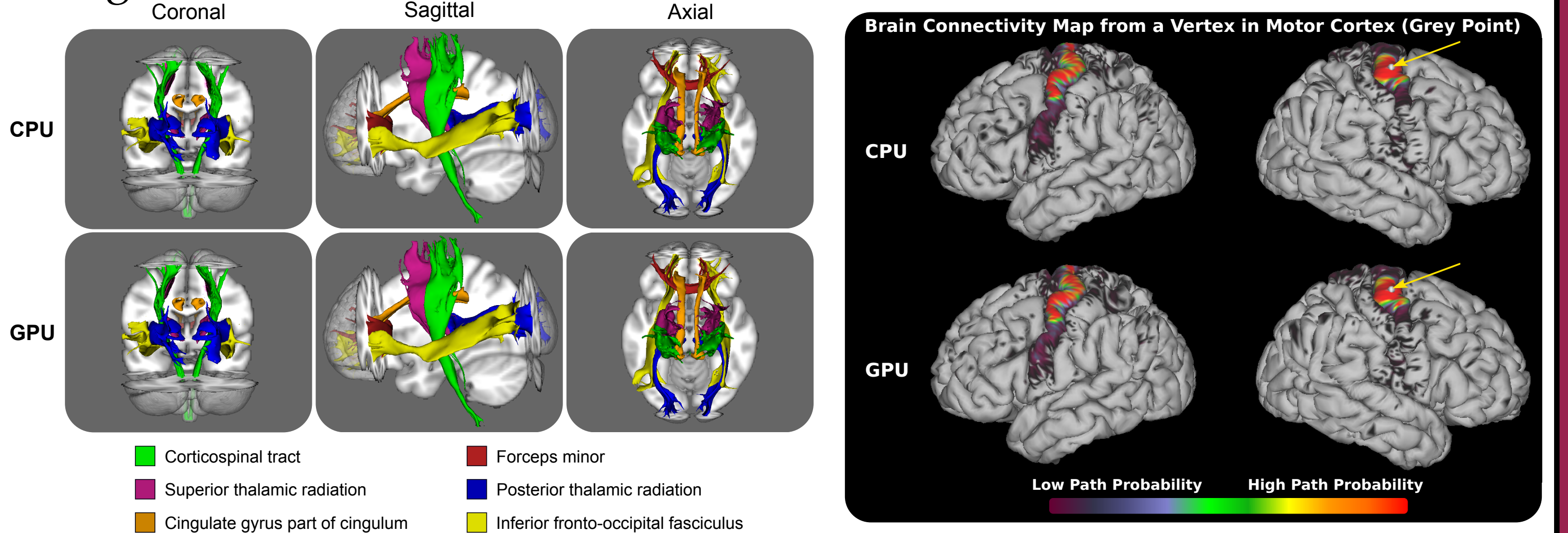

The option to add anatomical constraints as in [8]. For instance, allow streamlines to enter a subcortical region, propagate within it, but terminate upon exit.
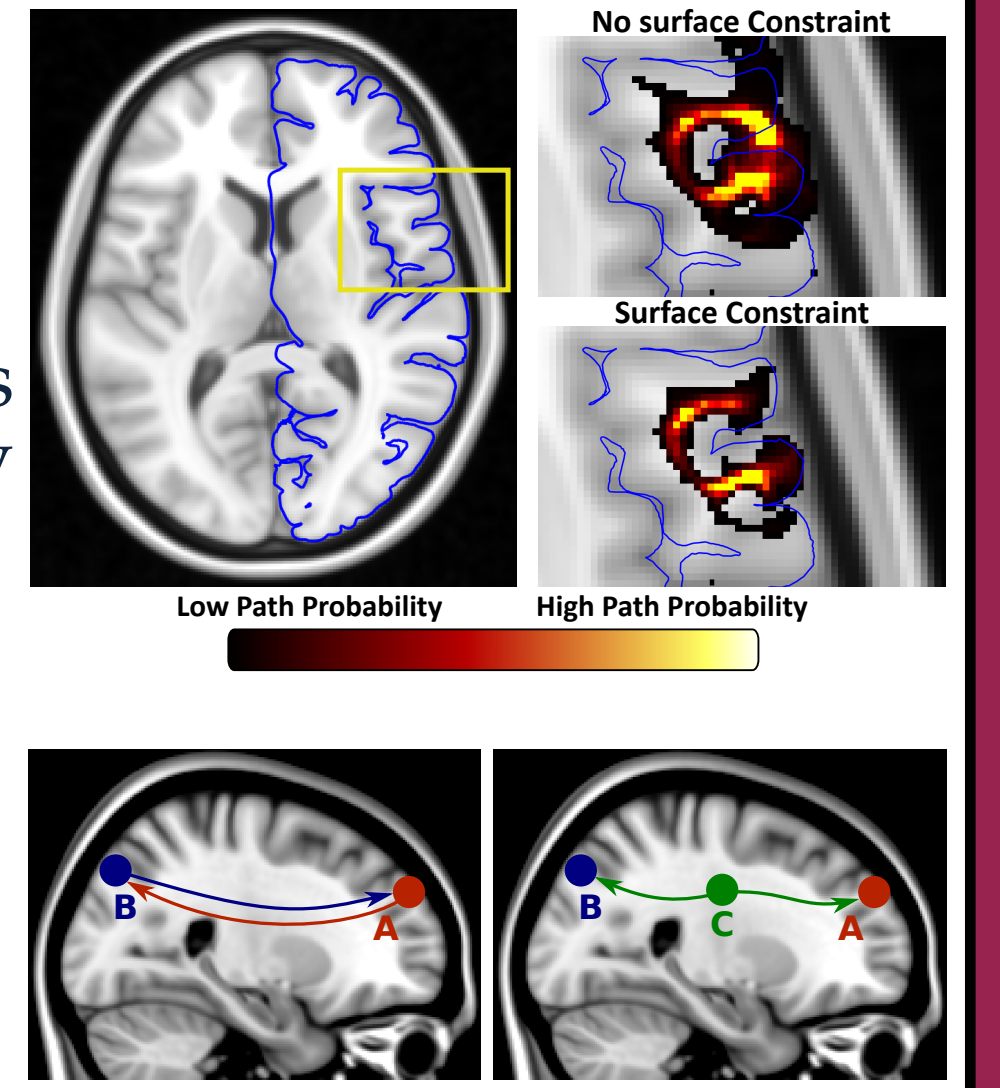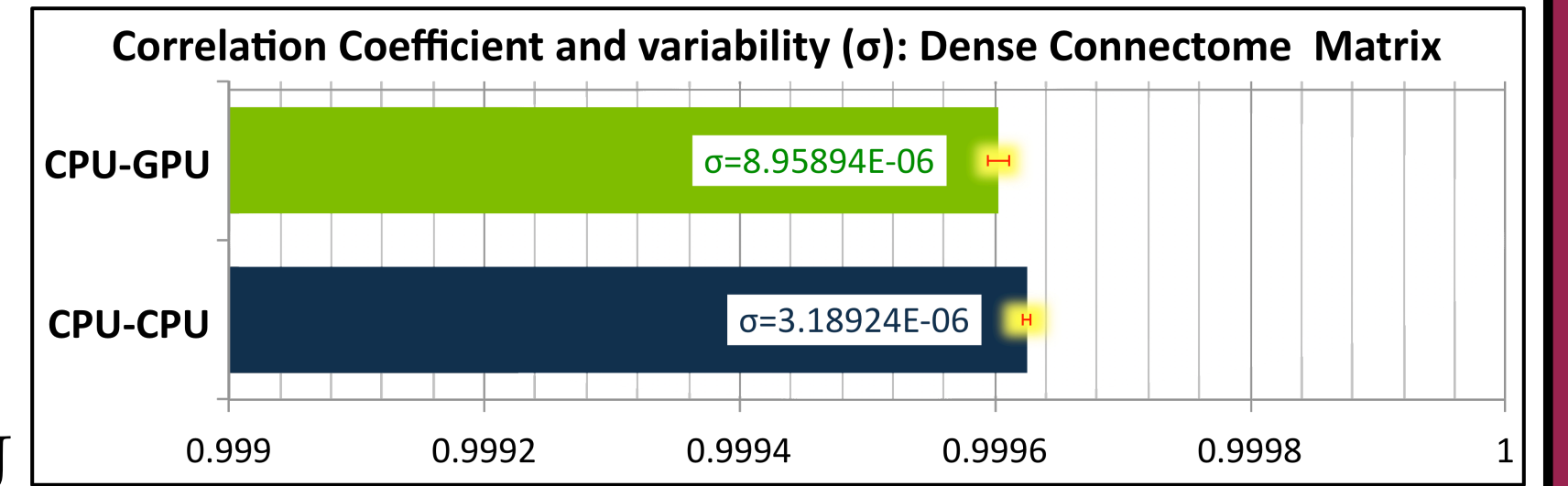
The possibility to generate **"dense" connectome matrices**, either seeding from WM or from WM/GM boundary: generation of tens of millions of streamlines between tens of thousands of "grayordinate" locations [9].

5. Comparing the GPU to the CPU implementation

Reconstruction of major tracts [10] using dMRI datasets from the HCP [1] and generation of dense connectome matrix with CPU and GPU.

Corticospinal tract
Superior thalamic radiation
Cingulate gyrus part of cingulum
Forceps minor
Posterior thalamic radiation
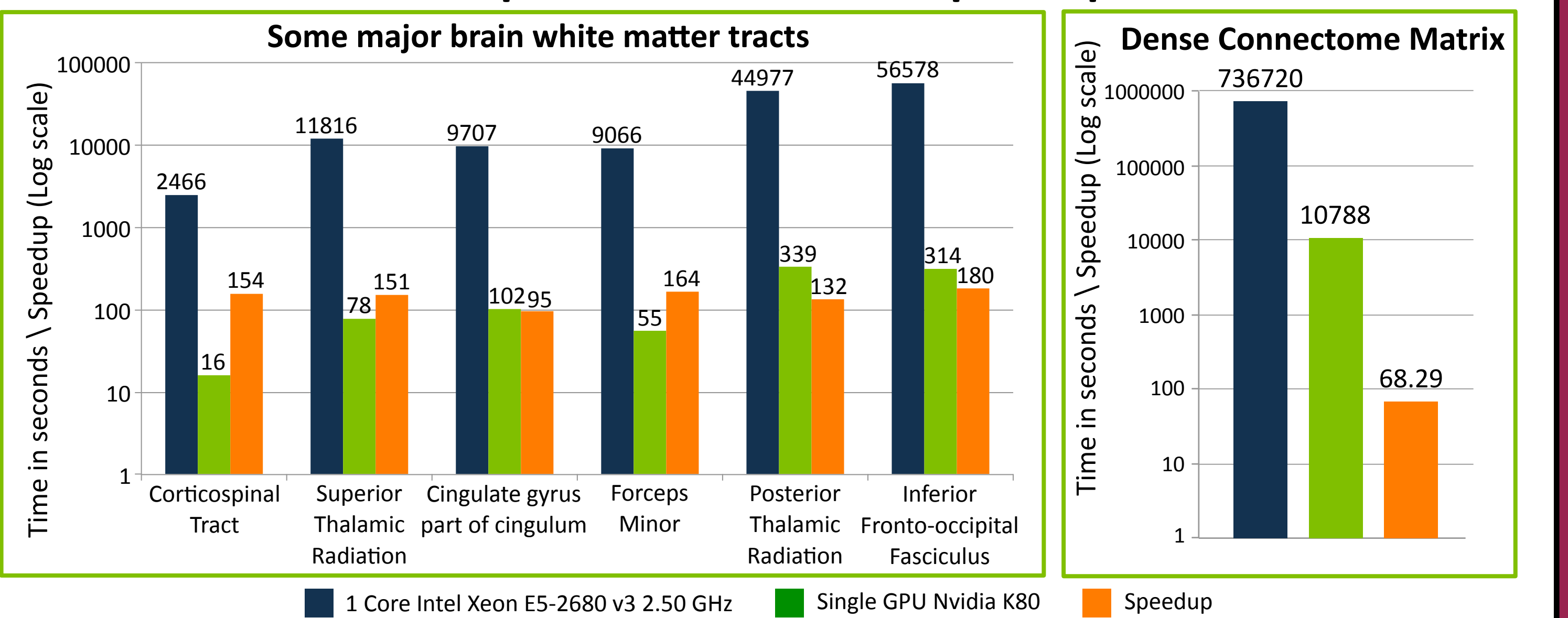Inferior fronto-occipital fasciculus

Given the stochastic nature of the process, we compare the mean of the correlation coefficients (and their variability) of the results from GPU and several CPU runs, with those from only several CPU runs. Very **high CPU-GPU correlations** (>0.99) are obtained for all the reconstructed tracts and the generated dense connectome matrices.

Correlation Coefficient and variability (σ): Dense Connectome Matrix
CPU-GPU σ=8.95894E-06
CPU-CPU σ=3.18924E-06

6. Reduction of computation times by two orders of magnitude

Computational Times & Speedup

Some major brain white matter tracts

	Corticospinal Tract	Superior Thalamic Radiation	Cingulate gyrus part of cingulum	Forceps Minor	Posterior Thalamic Radiation	Inferior Fronto-occipital Fasciculus
1 Core Intel Xeon E5-2680 v3 2.50 GHz	2466	11816	9707	9066	44977	56578
Single GPU Nvidia K80	16	78	102	55	339	314
Speedup	154	151	95	164	132	180

Dense Connectome Matrix

1 Core Intel Xeon E5-2680 v3 2.50 GHz	Single GPU Nvidia K80	Speedup
736720	10788	68.29

7. Conclusions

1. We have developed a GPU based tractography toolbox.
2. It will be publically available in the next version of FSL.
3. It reduces computation times by two orders of magnitude.
4. It includes new features: 3D surfaces for imposing anatomical constraints and generation of "dense" connectome matrices.

8. References

(1) Van Essen D.C. et al. *Neuroimage* 80: 62-79, 2013. (2) Sotiropoulos, S.N. et al. *Neuroimage* 80: 125-143, 2013. (3) UK Biobank: http://www.ukbiobank.ac.uk. (4) Sporns O. et al. *PLoS Comput Biol* 1(4): e42, 2005. (5) Nvidia. "CUDA C Programming Guide, version 7.5". 2015. (6) o'Rourke, J. "Computational geometry in C". Cambridge university press, 1998.
(7) Behrens T. E. et al. *NeuroImage* 34(1): 144-155, 2007. (8) Smith, R.E., et al. *Neuroimage* 62(3): 1924-1938, 2012. (9) Glasser, M.F., et al. *Neuroimage* 80: 105-124, 2013. (10) de Groot M. et al. *NeuroImage* 76: 400-411, 2013.

9. Acknowledgments