FreeSurfing on the GPU: Accelerating Brain MRI Processing

R. G. Edgar, T. Witzel, N. Schmansky, B. Fischl

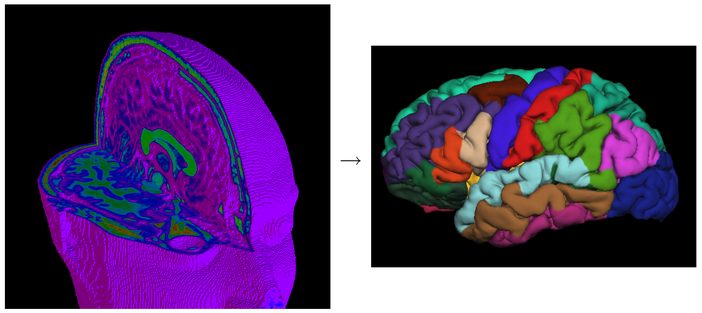
Athinoula A. Martinos Center for Biomedical Imaging, Massachusetts General Hospital, 149 Thirteenth Street, Charlestown, MA 02129

rge21@nmr.mgh.harvard.edu

http://surfer.nmr.mgh.harvard.edu/

The FreeSurfer Suite

FreeSurfer is a set of automated tools developed at the Athinoula A. Martinos Center for Biomedical Imaging which reconstruct the brain's cortical surface from structural MRI data, and overlay functional MRI data onto the reconstructed surface. A researcher provides a 'raw' MRI scan of the entire head as input, and Freesurfer computes a segmented brain image:



FreeSurfer is used by thousands of researchers worldwide, to study a huge variety of different conditions. These include Autism Spectrum Disorder, Alzheimer's Disease, Huntington's Disease, schizophrenia and epilepsy.

The Need for Speed

The current pipeline takes around eight hours to run on a single core of a Nehalem-class Xeon processor. Although this is adequate for academic research, clinical usage requires a processing time of one hour or less. Any longer, and the patient will have to go home and return for a second appointment.

Acceleration with CUDA

To achieve our goal of clinical usability, we turned to GPU acceleration using NVIDIA's CUDA technology. The `thrust` package was used for a number of operations, especially GPU-based parallel reductions.

Acceleration of individual routines in the FreeSurfer suite was generally straightforward. Many consisted of a large loop over all the image voxels (typically 256^3), with an independent operation being applied to each. These mapped onto the CUDA programming model very well, with one thread taking responsibility for each voxel or column of voxels. Individual routines generally achieved 10-20$\times$ speed-ups with relatively little effort at optimisation.

Marshalling the data for the GPU was a more challenging task. The CPU code made extensive use of three dimensional arrays of structures, with both xyz and zyx ordering. Consider the datastructure describing a non-linear transformation:

```
typedef struct {
  int width, height, depth; // Volume size
  GCA_MORPH_NODE ***nodes;  // Volume data
  // Other scalar data...
} GCA_MORPH;
```

where each GCA_MORPH_NODE was a 254 byte structure. For the GPU, the GCA_MORPH had to be repackaged into linear arrays. This made transfers very slow, taking upwards of 0.5 s in this case, of which only 100 ms would be spent on the PCIe bus. In contrast, the 'computation' time of a given routine would typically be around 200 ms on the CPU, and 20 ms on the GPU. Overall speed-ups could only be achieved when significant portions of a program were GPU accelerated. Our experience highlights the need for new programming paradigms which expose an array-of-structures model to the programmer, but which implement structure-of-arrays in the machine.

We developed a templated datastructure to manage volumetric data on the GPU. This splits into 'management' and 'mutator' classes. The 'management' class is used by CPU code and takes care of memory allocation and data transfer. The 'mutator' class is used in GPU code, and provides methods which enable the programmer to access the data via three dimensional indices, even though the data is stored in pitched linear memory. We believe that this paradigm of 'manager' and 'mutator' will be useful even in purely CPU-based code.

Results

We benchmarked our GPU implementation on a Tesla C2050 card, comparing the times to a single core of an Intel Xeon W5580 (3.2 GHz) based workstation. Some sample wall times are shown in the following table, together with the overall pipeline wall time. Some programs run multiple times, hence the savings from individual programs do not add up to the overall reduction in wall time.

Program	CPU (h:mm)	GPU (h:mm)
Linear Registration	0:19	0:04
Non-linear registration	1:50	0:19
Template Surface	1:00	0:16
Segmentation Statistics	0:09	0:04
Full pipeline	7:55	4:18

While individual routines are typically 10-20$\times$ faster on the GPU, PCIe transfer times and serial sections limit the overall speed-ups of each program to around 5$\times$. The overall pipeline speed-up is further reduced by the programs which have not been accelerated.

Testing was performed at multiple levels. Individual routines were tested by comparing results from the CPU and GPU implementations run on known inputs. We also ensured that the inevitable drift in results at the level of individual routines did not affect the final output by comparing the segmentations produced by the GPU pipeline to manual segmentations of thirty brains.

Conclusions

We have made significant progress towards enabling the FreeSurfer pipeline to run in one hour. A small number of programs dominate the GPU accelerated pipeline, and we are currently studying how to add CUDA acceleration to these.

This work also highlights the need for new programming paradigms which provide better abstractions for scientific programmers. A particular issue in this is the choice between array-of-structures and structure-of-arrays. Current languages require too much low-level manipulation of datastructures, encouraging the adoption of the former approach (which is more natural) when the latter typically offers far greater performance.

FreeSurfer is a suite of tools used by thousands of medical researchers worldwide. By adding GPU acceleration, we have come significantly closer to our ultimate goal of allowing FreeSurfer to be used in a clinical setting.

Acknowledgements

Support for this research was provided in part by the National Center for Research Resources (P41-RR14075, and the NCRR BIRN Morphometric Project BIRN002, U24 RR021382), the National Institute for Biomedical Imaging and Bioengineering (R01EB006758), the National Institute on Aging (AG022381), the National Center for Alternative Medicine (RC1 AT005728-01), the National Institute for Neurological Disorders and Stroke (R01 NS052585-01, 1R21NS072652-01), and was made possible by the resources provided by Shared Instrumentation Grants 1S10RR023401, 1S10RR019, and 1S10RR023043. Additional support was provided by The Autism & Dyslexia Project funded by the Ellison Medical Foundation.