# Fast MR Image Reconstruction using Graphics Processing Units

J. P. HALDAR[1], S. S. STONE[1], H. YI[1], S. C. TSAO[1], B. P. SUTTON[2], W-M. W. HWU[1], AND Z-P. LIANG[1]

[1]ELECTRICAL AND COMPUTER ENGINEERING, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN, URBANA, IL, UNITED STATES, [2]BIOENGINEERING, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN, URBANA, IL, UNITED STATES

## INTRODUCTION

The computational cost of image reconstruction is an important practical issue. Although conventional fast Fourier transform (FFT) reconstructions are fast, many important emerging algorithms are computationally more expensive, including notable iterative algorithms designed for non-Cartesian *k*-space data, parallel imaging data, or data contaminated by physical effects like field inhomogeneity. Significant efforts from the MR image reconstruction community have been put forth in recent years to increase the computational speeds of various reconstruction algorithms, using clever approximations to reduce the complexity of the algorithms themselves. In this work, we show that performing processing on graphics processing units (GPUs) can significantly improve computational performance and can open the door for using complex, exact computations for image reconstruction. GPUs offer vast parallel computation resources and increased memory bandwidth relative to standard central processing units (CPUs). Moreover, GPUs are inexpensive, small, and found in all modern computers, making them a powerful alternative to standard computer cluster technology. We are implementing several image reconstruction algorithms on the NVIDIA GeForce 8800 GTX GPU, and our preliminary experiences indicate the tremendous potential of this approach to fast image reconstruction.

## MATERIALS AND METHODS

Modern GPUs have massive parallel computational resources and high memory bandwidth, provide nearly complete support for IEEE standard math operations, and can be programmed with standard programming languages. Some key, relevant computational features of the GeForce 8800 GPU are summarized below:

| **Features of a GeForce 8800 GTX GPU** | |
|---|---|
| **Processing Power** | 128 processing cores, each operating at 1.35 GHz. Capability for 367 billion floating point operations per second (GFLOPS). Special onboard functional units for specific common mathematical operations. |
| **Parallel Computing Capabilities** | Up to 12,288 simultaneous threads per GPU, and an unlimited number of GPUs can be used in parallel. |
| **Memory** | 768 MB DRAM on chip, with 80 GB/s total memory bandwidth. |
| **Programming Language** | CUDA (Advanced C with simple extensions). |

Using the CUDA programming environment, we were able to code our algorithms in C. To utilize the parallel processing capabilities of the GPU, each algorithm is broken down into separate blocks that can be computed simultaneously in parallel. For example, in algorithms containing nested loops, different iterations of the loops are mapped onto different cores for simultaneous processing. This initial parallelization improves computational performance by roughly a factor of 10. Further GPU tuning is very algorithm-dependent, but can improve performance significantly [1]. After profiling the computational load of our algorithms by running standard test cases, we found that memory accesses and trigonometric function evaluations were significant performance bottlenecks. We optimized memory accesses by placing frequently-accessed variables in on-chip constant caches rather than in the global memory; this improved computational performance by several times. To improve the speed of trigonometric functions (which are used heavily in MR algorithms because of the Fourier encoding process), we used the special functional units (SFUs) of the GPU to allow sine and cosine evaluations to execute as a single instruction. Use of these SFUs comes at the expense of some numerical accuracy; however, based on empirical evidence, this level of discrepancy is negligible for the types of problems we've considered.

## RESULTS

Image reconstruction from high-resolution 3D non-Cartesian data is computationally demanding, particularly when undersampled trajectories are used with parallel imaging and/or regularized reconstruction methods. Gridding [2] is a well-known and widely-used technique impacting both iterative and non-iterative reconstruction from non-uniformly sampled *k*-space data, and functions by interpolating non-uniform data onto a Cartesian grid for subsequent processing via the fast Fourier transform (FFT). However, gridding is simply a fast approximation of direct conjugate-phase (DCP) reconstruction [3], which has previously been considered too slow for practical use. Using the techniques described in [4], we implemented DCP reconstruction on the GPU, and have achieved a computational speed of over 150 GFLOPS. In comparison, our implementation of the same DCP algorithm on a standard 2.66 GHz CPU achieves only 0.5 GFLOPS. In practical terms, this means that a DCP computation requiring 2 days of runtime on a standard CPU can now be done on a GPU in 10 minutes. Using these GPU implementations, we can perform DCP reconstruction from 3.2 million non-Cartesian data samples onto a 128x128x128 3D image matrix in only 7.5 minutes, and can perform DCP reconstruction from 75 thousand non-Cartesian samples onto a 256x256 2D image matrix in around 1 second. Though DCP reconstruction on the GPU is not faster than highly-optimized gridding on a CPU [5], it does not require any algorithmic approximations, and can make fast and precise iterative methods such as [6] even better. While the speed of the gridding procedure may still be preferable in many situations, acceleration of gridding on the GPU is expected to yield similar improvements in computational speed.

Besides DCP reconstruction, we have also begun GPU implementation of fast iterative algorithms such as the method of conjugate gradients (CG) and LSQR. Based on initial projections, we expect to be able to reconstruct 176 image slices with 256x256 in-plane resolution from non-uniformly undersampled (by a factor of 4) multi-slice parallel imaging data with 4 receiver coils in a few tens of seconds. This reconstruction now takes more than 20 minutes on a serial implementation using a CPU.

## CONCLUSION

In this work, we show that significant improvements in image reconstruction speed can be achieved by performing data-parallel computations on GPUs. Leveraging the resources of a single modern GPU, we can achieve computational performance that is hundreds of times faster than what is reported on a single modern CPU.

## REFERENCES

[1] S. Ryoo, *GPGPU* 2007.
[2] J. Jackson, *IEEE Trans Med Imag* 1991;10:473-478.
[3] A. Macovski, *Magn Res Med* 1985;2:29-40.
[4] S. Stone, *GPGPU* 2007.
[5] P. Beatty, *IEEE Trans Med Imag* 2005;24:799-808.
[6] F. Wajer, *Proc ISMRM* 2001;767.