IMPATIENT MRI: ILLINOIS MASSIVELY PARALLEL ACCELERATION TOOLKIT FOR IMAGE RECONSTRUCTION WITH ENHANCED THROUGHPUT IN MRI

Xiao-Long Wu[1], Jiading Gai[2], Fan Lam[1,2], Maojing Fu[1,2], Justin P. Haldar[1,2], Yue Zhuo[2,3],
Zhi-Pei Liang[1,2], Wen-Mei Hwu[1,2], Bradley P. Sutton[2,3]

[1]Department of Electrical and Computer Engineering, [2]Beckman Institute, [3]Bioengineering Department,
University of Illinois at Urbana-Champaign

ABSTRACT

Much progress has been made in the design of efficient acquisition trajectories for high spatial and temporal resolution in magnetic resonance imaging (MRI). Additionally, significant developments in image reconstruction have enabled the reconstruction of reasonable images from massively undersampled or noisy data that is corrupted by a variety of physical effects, including magnetic field inhomogeneity. Translation of these techniques into clinical imaging has been impeded by the need for expertise and computational facilities to realize the potential of these methods. We present the Illinois Massively Parallel Acceleration Toolkit for Image reconstruction with ENhanced Throughput in MRI (IMPATIENT MRI), a reconstruction utility that enables advanced techniques within clinically relevant computation times by using the computational power available in low-cost graphics processing cards.

Index Terms— magnetic resonance imaging, graphics processing cards, field inhomogeneity, image regularization

1. INTRODUCTION

Since the inception of magnetic resonance imaging (MRI), technology has continually evolved to improve tradeoffs between spatial resolution, temporal resolution, signal-to-noise ratio, image contrast, and examination time. Although it is still relatively young, MRI has reached a juncture where significant future advances are limited by traditional data acquisition strategies and image reconstruction schemes that ignore the non-idealities of the MR acquisition. Proprietary algorithms and a lack of access to sufficient computing power in the clinic, and sometimes even in the laboratory, have held back widespread adoption of advanced MRI acquisition and reconstruction schemes. This abstract seeks to remove this barrier to advancement by creating and validating an open source image reconstruction software package that leverages the computational power of the graphics hardware that is available in many PCs and reasonably affordable in stand-alone computational units.

Throughout the late 1990s and early 2000s, mainstream microprocessors such as the Intel Pentium and the AMD Opteron families have driven rapid performance increases and cost reductions in science and engineering. Development slowed, however, in 2004 due to constraints on power consumption. Since that time, many-core processors such as graphics processing units (GPUs) have led the advances in computational throughput for science and engineering applications. We are at a crossroads where computational power is available through multi-core CPU and many-core GPU-equipped PCs and clusters. However, significant infrastructure investment is required to enable the medical imaging community to leverage these computational resources for furthering imaging science. This abstract introduces the Illinois Massively Parallel Acceleration Toolkit for Image reconstruction with ENhanced Throughput in MRI (IMPATIENT MRI), an open source GPU-based advanced image reconstruction package that will be available to the public through the internet by the time of the conference. This package builds on our previous work in GPU-based acceleration of MR algorithms [1,2], integrating new features and improved computational efficiency.

IMPATIENT MRI is currently equipped to solve optimization problems of the form

$$\hat{\boldsymbol{\rho}} = \arg\min_{\boldsymbol{\rho}} \frac{1}{2}\left\|\mathbf{E}\boldsymbol{\rho} - \mathbf{d}\right\|_2^2 + \beta\left\|\mathbf{W}\mathbf{C}\boldsymbol{\rho}\right\|_2^2 \qquad (1)$$

Where ρ is the vector of image voxel coefficients to be reconstructed, **E** is a matrix that models the linear data acquisition physics for MRI, **d** is the vector of measured data, **C** is a regularization matrix, **W** is an optional diagonal weighting matrix, and β is a regularization parameter. This cost function consists of two terms, one imposing data consistency, and the other enforcing image regularity. Traditional MRI models data collection using the Fourier transform, and collects data on a Cartesian (i.e., rectilinear) grid to allow **E** to be represented using the fast Fourier transform (FFT). In contrast, our implementation can more accurately model advanced data acquisition strategies and non-ideal data acquisition physics. Our implementation includes capabilities for using non-Cartesian Fourier sampling trajectories (with and without FFT-based approximations of the non-uniform Fourier transform), B_0 field inhomogeneity compensation [3], and SENSE-based modeling for multichannel receivers [4]. Regularization can be important for stabilizing reconstructions when data-consistency constraints alone do not result in a sufficiently well-posed inverse problem. Our implementation allows **C** to be a general sparse matrix, with additional fast implementations available for certain commonly-used regularization functions. Use of the **W** matrix enables more advanced reconstructions that incorporate prior information (e.g., [5]) or that solve nonlinear problems [6] (e.g., the popular l_1 and total variation regularization schemes that appear in compressed sensing [7]). In addition, we offer two forms of the estimation algorithm: one based on a brute force exact approach and one Toeplitz-based reconstruction scheme [8]. The following sections describe the implementation of IMPATIENT MRI and illustrate its computational capabilities.

2. METHODS

2.1 Modeling Data Acquisition

Parallel imaging is performed by placing an array of receiver coils around the object to be imaged, with each receiver coil lending spatially distinct reception profiles to the acquired data sets. The collection of data with different coil sensitivities enables the reconstruction of alias-free reduced encoding data sets. For the *i*th receiver coil, data acquisition can be modeled as $\mathbf{FS}_i$, where $\mathbf{S}_i$ is a diagonal matrix containing the spatial sensitivity profile of the coil, and $\mathbf{F}$ is a matrix that describes the effects of k-space encoding and magnetic field inhomogeneity. If the data received by each coil is concatenated to make a single data vector $\boldsymbol{d}$, then the net encoding function is given as: $\mathbf{E} = [\mathbf{FS}_1; \ldots ; \mathbf{FS}_L]$, for an acquisition with an L-channel receiver. This parallel imaging is implemented in the reconstruction code as a wrapper around the single-coil operations for each of the included algorithms. The sensitivity maps are stored as a 1D vector to save memory as $\mathbf{S_i}$ is a diagonal matrix. Pre-multiplication by $\mathbf{S_i}$ for the forward transform and post-multiplication by $\mathbf{S_i^H}$ for the adjoint operator is performed by point-wise vector-vector multiplication.

2.2 Incorporation of *a priori* information

As mentioned above, the regularization term is essential for the reconstruction of practical MR images. IMPATIENT MRI treats the regularization matrix $\mathbf{C}$ as a general sparse matrix and also incorporates specialized implementations for commonly used regularization functions. In this paper, we present the implementation for a dual direction finite difference operator which can be expressed as $\mathbf{C} = [\mathbf{D}_H ; \mathbf{D}_V]$, where $\mathbf{D}_H$ and $\mathbf{D}_V$ denote the finite difference of every pixel pair along the horizontal and vertical directions of the image respectively. We have explored the following two ways to implement this regularization function:

1. Form matrices $\mathbf{D}_H$, $\mathbf{D}_V$ explicitly and store them as sparse matrices. While it may take extra effort to optimize sparse matrix storage and calculation, sparse matrices are widely applicable to finite difference calculation along any arbitrary direction and they are configurable to accommodate multiple forms of regularization terms.

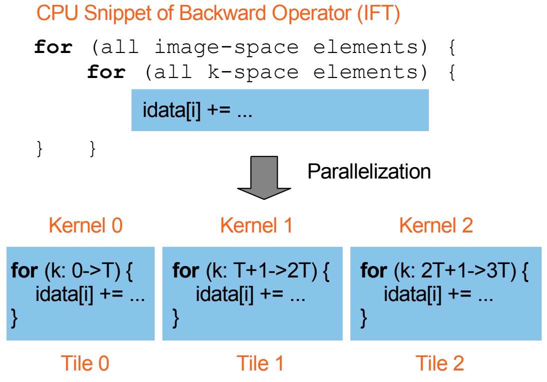
2. Tailor explicit finite difference calculations according to the specific choice of $\mathbf{C}$. This alternative requires no effort in the management of sparse matrix, reduces memory access redundancy in CUDA kernels and can be easily transformed from C codes into CUDA codes.

In the current implementations, the weighting matrix $\mathbf{W}$ is predetermined for l_2 regularization, and can be automatically updated for l_1 regularization. In the future, we will also enable customized input option to suit the user's specific requirements.

2.3 Implementation of Image Reconstruction in the GPU

The key implementation in the GPU code involves different levels of manipulation and optimizations. Based on our previous work [2,9], we already adopted the GPU constant memory to avoid frequent accessing of the slow global memory. In this work, we further optimize the code and extend it to manage larger images like 256^2 and 512^2. Due to limited space, we'll describe only key optimizations and provide explanation about the extension for handling large images. Detailed information can be found in [10].

Tiled processing [11] has been widely used in CUDA programming to deal with large data. Yet due to the complex hardware memory hierarchy and the need for algorithm-level information, this transformation must be manipulated by programmers. Figure 1 depicts the CPU code snippet and resultant GPU tiled computation of the adjoint operator of our brute force approach. The outer loop of the CPU code is removed after parallelization. Each loop iteration of the outer loop is handled by one thread. The whole image reconstruction process is conducted piece by piece or kernel by kernel such that we can take advantage of the constant memory for frequently accessed data. We choose the k-space trajectory data for the adjoint operator and the image-space pixel locations for the forward operator. Since only 64KB is available for the constant memory, for single-precision computation with 2-D trajectory data, the maximum number for tile size would seem to be 8192 which equals to 65536 B / (2 dimension trajectory variables x 4 bytes for single precision data). Unfortunately, the compiler will implicitly use up several to make it less than 64KB and to be in favor of regular thread processing, the number had better be the multiple of power of two. So 4096 is the final tile size in each kernel invocation (or 2048 for double-precision calculations).



```
for (all image-space elements) {
    for (all k-space elements) {
        idata[i] += ...
}   }
```

```
for (k: 0->T) {
    idata[i] += ...
}
```

```
for (k: T+1->2T) {
    idata[i] += ...
}
```

```
for (k: 2T+1->3T) {
    idata[i] += ...
}
```

Figure 1. Tiled processing on the adjoint operator

For the adjoint operator, the output image-space data is updated based on the input k-space data. Every output pixel is contributed to by every input point. A choice can be made on how to approach the calculation, either from an input-oriented or output-oriented perspective, as shown in Figure 2. Taking an input-oriented perspective can cause a big performance loss, due to the sharing of outputs among threads. More specifically, when multiple threads are accessing the same output point, their requests have to be processed in serial in order to avoid erroneous results. For resolving this in GPU programming, atomic operations are widely used. Although this is faster in the latest Nvidia Fermi architecture, we can still benefit by avoiding this situation. Therefore, we choose an output-oriented approach.

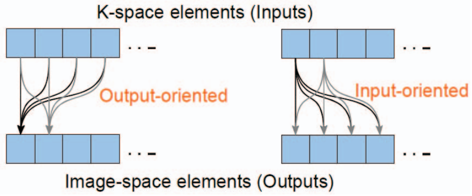


Figure 2. Output-oriented vs. input-oriented perspectives [12]

Table 1 lists the optimization guidelines we propose for performance tuning. We start from checking if the kernel is compute-bound or memory-bound. More specifically, for compute-bound kernels the computation operations are the main performance limiting factor. Hence we can either reduce the instructions through the suggested code transformation or move the kernel to a more advanced GPU device. On the other hand, memory-bound kernels are limited by the data access latency or bandwidth. Memory latency is the time to access data, which can be hidden by introducing more instruction mix to achieve instruction level parallelism. Memory bandwidth is the throughput that the memory can provide, which can be reduced by using faster constant or shared memory or fully utilize the given bandwidth by regularizing the memory access pattern.

Table 1. Optimization guidelines for performance tuning

Compute-bound	Memory-bound	
Instruction reduction	**Memory latency hiding**	**Memory bandwidth reduction**
Reduce branches and loop counting (Loop unrolling)	Automatic instruction scheduling (Loop unrolling)	Tiled compute (Using high-bandwidth memory)
Common subexpression elimination (Using registers)	Manual instruction scheduling (Data prefetching, Double buffering)	Memory layout transformation (Coalescing/Bank conflicts/Working size)
		Reuse data (Using high-bandwidth memory)

Again taking the adjoint operator of the brute force algorithm as an example, which is listed in Figure 1, the kernel is roughly compute-bound because of 17 floating-point operations and 3 global memory accesses. Since the trajectory data are read-only and used by all threads, they are stored in the constant memory to leverage the on-chip high bandwidth. More registers are used to eliminate calculations from common sub-expressions and to avoid accesses to global memory. Coalesced memory access is taken into account, based on the idea of Structure-of-Array memory layout. Loop unrolling is applied for further parallelism in terms of the granularities of instructions and loop bodies to hide memory latency. After these optimizations, we get another ~25% to ~34% speedup compared to our first GPU code snippet in [2].

3. RESULTS

3.1 Description of data sets

Two different data sets are used to investigate the performance of the IMPATIENT MRI code.

Data Set 1: The first data set is a high-resolution, multi-shot spiral functional MRI acquisition acquired on a Siemens 3 T Trio with a 32-channel head coil. A gradient echo acquisition with 25 ms echo time, 2s TR, and 10 slices of coverage was acquired while a subject was at rest in accordance with the institutional review board. The spirals are designed according to [13] using a maximum gradient amplitude of 22 mT/m and a maximum slew rate of 140 mT/m/ms. The data was acquired at several resolutions with image sizes of $[256, 512]^2$ with a 10-shot spiral acquistion. The 32-channel sensitivity maps and magnetic field inhomogeneity map are acquired from low-resolution 24-shot spiral FLASH acquisitions with matrix size 128^2 and TE's of 2 and 2.5 ms.

Data Set 2: For the second dataset, we use the Shepp-Logan phantom to simulate an undersampled reconstruction using one shot of an 8-shot 256^2 spiral trajectory, designed similarly to those above. The simulations use an 8-channel array and field inhomogeneity map from the ISMRM data recon challenge for an abdomen scan (Double Vision data) [14]. A small amount of noise is added to the data.

3.2 Reconstruction results for CPU/GPU implementations

All the data sets are processed by a machine with Intel Xeon E5520 CPU having 8 logical threads and one GTX 480 (Fermi) GPGPU having 480 processing cores.

Data Set 1: Figure 3 gives the reconstruction results for IMPATIENT MRI using a 512^2 matrix size and a full SENSE reconstruction with and without field inhomogeneity correction. Notice that the field inhomogeneity correction corrects for the blurring induced by magnetic field inhomogeneities, as indicated by the arrows. Table 2 gives the GPU reconstruction time results from an implementation of the Toeplitz-based method in single-precision mode with 20 CG iterations. All images are with SENSE reconstruction and 32-channel data. We have not included data-independent precomputations in our timing results. A Hanning window was used for the time segmentation interpolator with 15 time segments used.

Comparison with fast methods for CPU reconstruction

To provide a comparison with a common choice that has been used to speed up the calculation of the image, we compared our GPU reconstruction time to a time-segmented non-uniform FFT (NUFFT) reconstruction including SENSE [3]. The SENSE NUFFT reconstruction times on a CPU are listed in Table 2.

Data Set 2: Figure 4 compares the reconstruction results from the GPU reconstruction with and without TV. Figure 4a shows the sum-of-squares image, which is obtained by reconstructing each of the 8 coils individually without using sensitivity maps. Then we combine the images from the 8 different channels using a sum-of-squares and conclude with a square-root. Figure 4b and Figure 4c shows the SENSE reconstruction (8 coils) without and with TV, respectively.

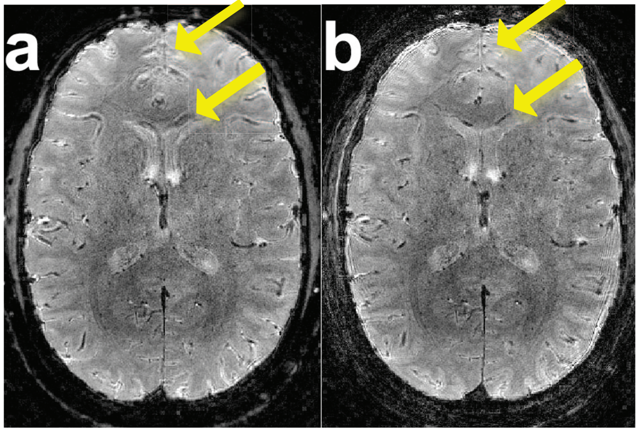


Figure 3. Resultant images for a) SENSE reconstruction without field correction, b) SENSE with FM correction.

Table 2. Execution time between the CPU and GPU implementations.

Data set	CPU	GPU	Speedup
256x256	2.1 hrs	28.55 sec	265
512x512	43.3 hrs	145.54 sec	1071

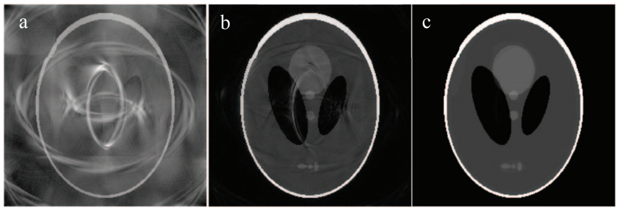


Figure 4. Results with and without TV regularization for data set 2

4. DISCUSSION

Clinical MRI has been limited by acquisitions that can be quickly reconstructed using standard FFT approaches and Cartesian sampling. The IMPATIENT MRI software provides a general image reconstruction approach for advanced data acquisitions in MRI, including: non-Cartesian sampling schemes, long image readouts that suffer from magnetic susceptibility image distortion artifacts, and parallel multi-coil acquisitions that can result in reduced image acquisition time. With the availability of this powerful image reconstruction environment, the use of advanced image acquisitions is possible while maintaining reconstruction times that allow for review of image quality while the patient is still in the imaging suite.

5. CONCLUSION

The IMPATIENT MRI software presented in this abstract will enable the use of advanced acquisitions in the clinical environment. Potential improvements in both image quality and acquisition speed will be available through the use of this dedicated reconstruction utility that provides flexible and optimized implementation of advanced image reconstruction features, including: correction of magnetic field susceptibility-induced image distortions, parallel imaging using SENSE, and incorporation of *a priori* information such as a roughness penalty. This powerful image reconstruction utility will enable the use of advanced image acquisition and reconstruction approaches in clinically feasible reconstruction time. In addition, the open-source code will allow engineers to implement additional physics specific to their own applications into the code while leveraging the advanced reconstruction tools already available.

Software package web site: http://impact.crhc.illinois.edu/mri.php

6. REFERENCES

[1] S. S. Stone, J. P. Haldar, S. C. Tsao, W.-m. W. Hwu, B. P. Sutton, Z.-P. Liang. “Accelerating Advanced MRI Reconstructions on GPUs,” *J. Parallel Distrib. Comput.*, vol. 68, pp. 1307-1318, 2008.

[2] Y. Zhuo, X.-L. Wu, J. P. Haldar Z.-P. Liang, W.-m. W. Hwu, B. P. Sutton. “The Role of GPUs in Advancing Clinical Imaging with Magnetic Resonance Imaging,” in *GPU Computing Gems*, W.-M. W. Hwu Ed., Elsevier Inc., 2011. *In Press*.

[3] B. P. Sutton, D. C. Noll, J. A. Fessler. “Fast, Iterative Image Reconstruction for MRI in the Presence of Field Inhomogeneities,” *IEEE Trans. Med. Imaging*, vol. 22, pp. 178-188, 2003.

[4] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger. “SENSE: Sensitivity encoding for fast MRI,” Magn Reson Med, vol. 42, pp. 952-962, 1999.

[5] J. P. Haldar, D. Hernando, S.-K. Song, and Z.-P. Liang, “Anatomically Constrained Reconstruction from Noisy Data,” Magn Reson Med, vol. 59, pp. 810-818, 2008.

[6] M. Nikolova and M. K. Ng, "Analysis of Half-Quadratic Minimization Methods for Signal and Image Recovery," SIAM J. Scientific Computing, vol. 27, pp. 937-966, 2005.

[7] M. Lustig, D. Donoho, and J. M. Pauly, “Sparse MRI: The application of compressed sensing for rapid MR imaging,” Magn Reson Med, vol. 58, pp. 1182–1195, 2007.

[8] J. A. Fessler, S. Lee, V. T. Olafsson, H. R. Shi, D.C. Noll. Toeplitz-based iterative image reconstruction for MRI with correction for magnetic field inhomogeneity. IEEE Trans Sign Proc, 53(9): 3393-3402.

[9] Y. Zhuo, X.-L. Wu, J. P. Haldar, W. Hwu, Z.-P. Liang, B. P. Sutton, “Accelerating Iterative Field-Compensated MR Image Reconstruction on GPUs,” Proceedings of the IEEE Intl Sym on Biomedical Imaging (ISBI), April, 2010.

[10] X.-L. Wu, Y. Zhuo, J. Gai, F. Lam, M. Fu, J. P. Haldar, W. Hwu, Z.-P. Liang, B. P. Sutton, “Advanced MRI Reconstruction Toolbox with Accelerating on GPU,” Proceedings of the IS&T/SPIE Electronic Imaging 2011 Conference on "Parallel Processing for Imaging Applications", January 2011.

[11] D. B. Kirk, W. Hwu, “Programming Massively Parallel Processors: A Hands-on Approach,” Morgan Kaufmann, 1st edition (February 5, 2010).

[12] Wen-mei Hwu, “ECE 598 HK : Computational Thinking for Manycore Processors,” http://courses.engr.illinois.edu/ece598/hk/.

[13] G. H. Glover. Simple analytic spiral K-space algorithm. Magn Reson Med. 1999 Aug;42(2):412-5.

[14] Data Reconstruction Challenge. 2010 Intl Soc Magn Reson Med. (ismrm.org/mri_unbound)