Sparse regularization in MRI iterative reconstruction using GPUs

Yue Zhuo[1], Bradley Sutton[2]
[1]*Student Member, IEEE,* [2]*Member, IEEE*
Department of Bioengineering
University of Illinois at Urbana-Champaign
Urbana, IL, United States

Xiao-Long Wu[1], Justin Haldar[1], Wen-mei Hwu[3], Zhi-pei Liang[3]
[1]*Student Member, IEEE,* [3]*Fellow, IEEE*
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL, United States

***Abstract* — Regularization is a common technique used to improve image quality in inverse problems such as MR image reconstruction. In this work, we extend our previous Graphics Processing Unit (GPU) implementation of MR image reconstruction with compensation for susceptibility-induced field inhomogeneity effects by incorporating an additional quadratic regularization term. Regularization techniques commonly impose the prior information that MR images are relatively smooth by penalizing large changes in intensity between neighboring voxels. However, the associated computations often increase data access and the overall computational load, which can lead to slower image reconstruction. This motivates us to adopt a GPU-enabled implementation of spatial regularization using sparse matrices. This implementation enables the computations for the entire reconstruction procedure to be done on the GPU, which avoids the memory bandwidth bottlenecks associated with frequent communications between the GPU and CPU. Both the CPU and GPU code of this implementation will be available for release at the time of the conference.**

Keywords- Regularization, Sparse matrix, Graphics Processing Unit (GPU), CUDA, magnetic resonance imaging (MRI), Iterative reconstruction

I. INTRODUCTION

Susceptibility-induced magnetic field inhomogeneity exist near the interface of the air and tissue in the human brain in functional magnetic resonance imaging (fMRI) [1-5]. These inhomogeneities lead to susceptibility artifacts in reconstructed images such as geometric distortion and signal loss. Many methods are available to compensate for the susceptibility artifacts [6-14]. We have previously developed and validated an iterative reconstruction algorithm compensating for the susceptibility artifacts in MRI. This method relies on an advanced imaging model including the magnetic field map and its gradient combined with a conjugate gradient reconstruction [15-16]. In addition, a regularization term is used in the iterative reconstruction in order to enforce a spatial smoothness constraint on the reconstructed image. A sparse representation of the regularization constraint is used. Sparse representations are commonly used to represent regularization term that affect a neighborhood of each voxel and are therefore sparse by construction.

However, the iterative reconstruction method with field inhomogeneity correction is time-consuming due to the large data sizes and the tremendous computational demand for performing a Fourier-like transform operation without using FFT. Thus, accelerating the iterative reconstruction method with field correction adopting the parallel programming facility on GPU's can increase the practicality of the algorithm. We have previously proposed a GPU-enabled implementation without [17-19] and with [20-21] correcting for magnetic field inhomogeneities using our iterative reconstruction algorithm. In this work, we extend the iterative reconstruction by adding a regularization term to penalize the roughness of an estimated image. Sparse matrices are represented using the compressed sparse row (CSR) format for the fast sparse matrix-vector multiplication on GPUs, as we will describe in the next section. The addition of the regularization term yields better noise reduction by enforcing high similarity between neighboring voxels.

Similar regularization work using sparse matrix-vector multiplication (SpMV) is referred in [17-18]. However, it was not using the best known method for SpMV kernels on GPU's. Besides, the particular characteristic of the sparse data in our application are not taken into account either. Yet in the proposed work in this paper, we choose the best known SpMV based on the sparse matrix structure in regularization term kernel for spatial smoothness during the MRI reconstruction in [22-23].

In the field of the parallel programming on GPU's, the sparse matrix operation is challenging due to the huge matrix sizes and irregular data accesses to memory. Since the number of nonzero elements in a sparse matrix is very low, special data compression formats are used to relieve the memory size limitations from a computer system. This, though, bypasses the bandwidth and memory limitations, it introduces branch divergences for GPU hardware during the computation process, which is one key characteristic of vector machines running SPMD (Single Program, Multiple Data) programs. A branch divergence usually occurs when "if-else" control structures are used in a program. The consequence is that the "if-" and "else-part" code segments are executed in serial by different threads. It happens even when only one thread takes one path and the remaining threads take the other. More specifically, this doubles the execution time. However, if we can make all threads take the same path, for instance, to take the "if-" or "else-part" only, the branch divergence can be skipped. Therefore, this property of sparse matrix operation makes the accelerating of parallel programming on GPU difficult and thus it is also one of the motivations for our work as well.

The rest of the paper is organized as follows. We first present the methods used, including a description of the reconstruction algorithm and a discussion on the choice of the sparse matrix representation for fast GPU implementation. Then we present the results, analyze the effect of regularization on image quality in presence of noise and compare the execution times for CPU and GPU implementations.

II. METHOD

In this section, we briefly recall the iterative reconstruction algorithm and the original GPU implementation (with no regularization constraint). Further, we discuss sparse matrix representations and justify our choice for GPU implementation.

A. *Iterative reconstruction*

The proposed iterative reconstruction models the field inhomogeneities in the imaging system. The reconstruction algorithm aims at finding an estimate of the image intensity, ρ, defined by:

$$\hat{\rho} = \arg\min_{\rho} \|F\rho - d\|_2^2 + \beta R(\rho) \tag{1}$$

where F is the imaging model (including the magnetic field map and its gradients); d is the measured k-space data. The reader is referred to [16, 20] for more details on the imaging model (also called system matrix). The second term in Eq. (1) corresponds to the regularization constraint. It ensures that the image intensity estimate, ρ, verifies the constraint defined by R(•). In our implementation R(•) is a function used to compare the intensity of each voxel with the intensity of its neighboring voxels (i.e., 8-connected neighbors). By nature, the value in R function is sparse and is therefore represented as sparse matrix. Note that the weighting factor, beta, is used to control the trade-off between the least square criterion and the regularization term. While the first term in Eq. (1) measures the similarity between the acquired k-space data and the k-space data simulated using the image estimate and the imaging model, the second term penalizes images that are not spatially smooth. Iterative estimation is performed using the conjugate gradient algorithm as in [24]. We have previously studied the convergence of the algorithm [16] and shown that 8 to 10 iterations are typically enough to obtain a reasonable image estimate. In the next section, we describe the GPU implementation of the conjugate gradient reconstruction algorithm.

B. *GPU implementation of the conjugate gradient algorithm*

The recent expansion of graphics processing units (GPU) has allowed significant speedups in image reconstruction for medical imaging. Our previous implementation of the conjugate gradient algorithm (without regularization) has shown speedups of up to two orders of magnitude compared to CPU execution time [20-21]. The implementation was performed on the NVIDIA GeForce GTX 280 GPU, which can achieve 933 GFLOPS of peak theoretical performance and has an access bandwidth of 141.7 GB/s. In addition, the on-chip memory on GTX280 allows for acceleration of data transfer and reduced need for off-chip memory bandwidth [25]. The compute unified device architecture 3.0 (CUDA) was used as it supports the single-program, multiple-data (SPMD) parallel execution [25-28]. These features are utilized to provide a GPU-optimized reconstruction algorithm. The traditional CPU platform used for comparison was a quad-core 2.37 GHz AMD Opterons with 8 GB of memory.

C. *Sparse Matrix-Vector Multiplications on GPU's*

Sparse matrix-vector/-matrix multiplications are widely used on various fields of scientific computations. The key features of the problem are 1) the multiplication and addition of elements from matrices are naturally independent, 2) a big majority of zero elements do not need computation, and 3) the matrix size is much larger as compared with the problem of dense matrix multiplications. These characteristics imply that this problem can be conquered in a parallel fashion with suitable algorithm design and hardware support. Sparse matrix-vector multiplications on GPUs have been well-studied in [22-23]. Bell and Garland studied several commonly-used matrix formats, including Diagonal format, ELLPACK (ELL) format, Coordinate (COO) format, and Compressed Sparse Row (CSR) format, and their implementations on GPUs. The main point of these representations is to remove the redundant elements, namely zero values, while keeping the representations readable, easy to manipulate, and dense for extraordinarily big matrix sizes. However, since each format is designed for specific matrix characteristic in order to have the best performance, it is important to choose the right format accordingly. For our application, we choose CSR format and the corresponding GPU implementation CSR vector kernel from [23].

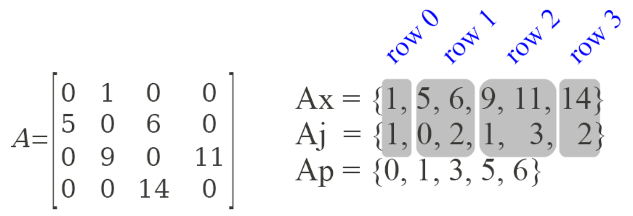
row 0
row 1
row 2
row 3
A = [0 1 0 0; 5 0 6 0; 0 9 0 11; 0 0 14 0]
Ax = {1, 5, 6, 9, 11, 14}
Aj = {1, 0, 2, 1, 3, 2}
Ap = {0, 1, 3, 5, 6}

Figure 1. CSR representation example

Figure 1 shows an example matrix in CSR format. CSR format takes three arrays to represent a sparse matrix. The first array Ax stores nonzero values in row-major order. The second array Aj stores the corresponding column indexes in array Ax. The numbers of nonzeros in all rows and the corresponding row indexes are encoded in the third array Ap as paired value list. For example, the first pair (0, 1) shows the number of nonzeros in the first row is 1. The second pair (1, 3) shows the number of nonzeros in the 2nd row is 2. Therefore, the sizes of Ax and Aj are equivalent to the number of nonzeros. The size of Ap equals to the number of rows plus one. For a matrix of size M x N with V nonzero elements, the required space for the CSR encoding is thus 2V+M+1. Since V is usually small, M will dominate the final space. In our application, this space requirement is affordable because M, N, and V are roughly 16K, 4K and 10K, respectively. The space requirement is close to 36K. Because in our algorithm, we need both a generic matrix-vector multiplication and a transposed one, we can not benefit from using a matrix with a smaller row size and larger column size. However, this characteristic is useful for other applications using only one of both.

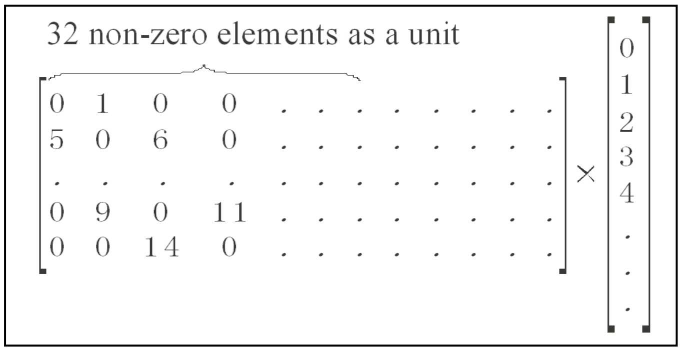

Figure 2. GPU execution behavior on CSR matrix data

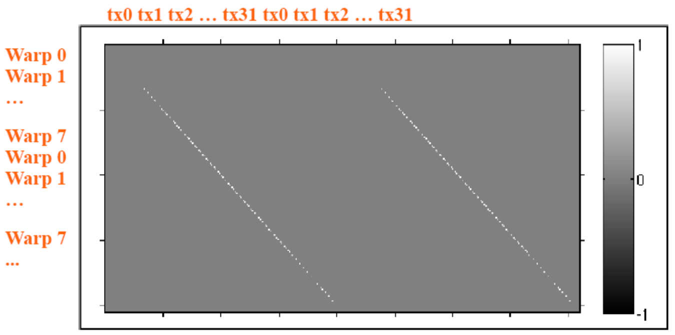

Figure 3. Partial data distribution of our matrix data

In the CSR vector kernel, the nonzero elements of each row in a matrix is served by one warp (32 threads). Thus each row is manipulated in the unit of 32 nonzero elements and 32 multiplication results are added into the final sum of each row, as demonstrated in Figure 2. Using Figure 3 as an example (partial data of our regularization matrix for spatial smoothness), where the white dots are nonzero elements. For the simplicity of explanation, here we use 256 threads (8 warps) as one block to process the given image. For each warp, the 32 threads will be manipulating the nonzero elements on each specified row until no elements on the row. Then, it jumps to the next (total warp number + current row ID)th row until no more rows. For instance, the threads in Warp 0 will jump to the (8 + 0)th row after it finishes the 0^{th} row. The threads in Warp 1 will jump to the (8+1)th row.

Therefore, as we can see if the standard deviation of the numbers of nonzeros in all rows is high, the load imbalance situation can degrade the performance. Taking one extreme case as an example, if the nonzero elements in the rows assigned to Warp 0 are very big as compared to others, Warp 0 will be the only active warp in a block after others finish their jobs soon. This situation, in a sense, is equivalent to declaring a thread block with only one warp and having resources occupied by more than one warp. Yet if the standard deviation of the numbers of nonzeros in all rows is low, most warps can finish their jobs in a smaller time window. The GPU hardware resources can be fully utilized.

Figure 3 also shows the data distribution of the matrix data in our application. As we can see, each row contains roughly similar number of nonzero elements. By this characteristic, CSR format is very suitable as the computational format in our case since little load imbalance can happen.

III. RESULTS AND DISCUSSIONS

Some of the preliminary results are presented in this section. The performances for the iterative MR image reconstruction with the regularization between CPU and the fast implementation on GPU are compared for two tested dataset with matrix size 64x64x1 and 128x128x1. The image reconstruction results with the regularization using the proposed implementation are reported here.

A. Performance comparison

The performance of the iterative reconstruction method on CPU and proposed accelerating reconstruction with field inhomogeneity compensation using GPU are compared in Table 1.

TABLE I. PERFORMANCE COMPARISON BETWEEN GPU AND CPU FOR SPMV

Data size	**Performance comparison for SpMV**		
	***GPU time* (ms)**	***CPU time* (ms)**	***Speedup***
64x64x1	3.694	0.705	0.19x
128x128x1	14.611	2.836	0.19x

As shown in Table I, the reconstruction time of the GPU fast implementation is compared with the reconstruction on CPU. Although we currently found the GPU-based SpMV kernel is still slower than the CPU-based version in the preliminary results, the major performance bottleneck is not limited by the SpMV kernels, even when the matrix sizes are increased.

TABLE II. PERFORMANCE COMPARISON BETWEEN GPU AND CPU FOR OVERALL ITERATIVE CG RECONSTRUCTION

Data size	**Performance comparison for CG**		
	***GPU time* (ms)**	***CPU time* (ms)**	***Speedup***
64x64x1	124.066	24916.102	200.83x
128x128x1	435.753	95542.203	219.26x

The overall performance for iterative CG reconstruction with susceptibility gradients are shown in Table II. The results clearly show that the proposed fast implementation on GPU achieves speedup about 200 comparing with implementation on CPU with the tested data set.

B. Image quality comparison

The GPU implementation results are shown in Figure 4 for comparison between (a) the implementation on CPU and (b) fast implementation on GPU. We can see that the differences between the results from GPU and CPU are negligible. Besides, the image quality was improved by including the within-plane and through-plane susceptibility gradients into the reconstruction model. Effects of magnetic field inhomogeneity (including both geometric distortion and signal loss) are corrected for especially in the frontal-orbital area (as previously reported in [15]).

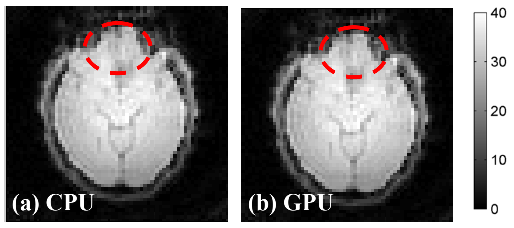

Figure 4. Image comparisons between GPU and CPU

For the computation time, the current preliminary implementation on GPU with sparse representation is still slower than the CPU. However, this implementation enables the regularization procedure executed on GPU instead of on CPU. And hence this implementation avoids the bottleneck of the communication between CPU and GPU (for regularization in the forward and backward operations during each iteration). Besides, the proposed implementation of regularization for spatial smoothness can be easily extend to include more flexible regularization methods, such as support constraints, reference images, etc. Nevertheless, the preliminary GPU code for regularization is a promising implementation. We are currently working on the optimization of the GPU code and expect to be able to accelerate the computation further when releasing the code (CPU and GPU code with sparse regularization).

IV. CONCLUSION

In summary, we present an iterative reconstruction for MR images using a regularization constraint for spatial smoothness and implemented on Graphics Processing Unit (GPU) hardware. In MRI, iterative image reconstruction allows for accurate modeling of the imaging system and therefore can improve image quality compared to non-iterative methods. We have demonstrated in our earlier work the benefits of using an imaging model including the magnetic field map and its gradients in order to compensate for the susceptibility artifacts.

In this work we extend our previous reconstruction model by adding a regularization term in our previously published GPU-enabled reconstruction method. Regularization can provide better noise reduction properties and thus better images when the acquired signal is noisy. Implementation on GPUs significantly reduces computation times to clinically practical times. Simulations show that the proposed implementation enables the regularization on GPU by using the sparse matrix representation and therefore efficiently enforces spatial smoothness. Future work includes more detailed quantitative comparison between GPU and CPU for the sparse regularization on GPU.

ACKNOWLEDGMENT

We would like to acknowledge the high-performance computing resources provided by Institute for Advanced Computing Applications and Technologies (IACAT).

REFERENCES

[1] K. Sekihara, M. Kuroda, H. Kohno. Image restoration from non-uniform magnetic field influence for direct Fourier NMR imaging. *Phys Med Biol.* 1984; Jan; 29(1): p.15-24.

[2] D.C. Noll, C.H. Meyer, J.M. Pauly, D.G. Nishimura, A. Macovski. "A homogeneity correction method for magnetic resonance imaging with time-varying gradients," *IEEE Trans Med Imaging*, 1991; 10(4): p.629-637.

[3] P. Jezzard, et al. "Correction for geometric distortion in echo planar images from B0 field variations," *Magn Reson Med*, 1995; 34: p.65-73.

[4] L.C. Man, et al. Improved automatic off-resonance correction without a field map in spiral imaging. *Magn Reson Med*, 1997; 37: p.906-913.

[5] H. Schomberg, "Off-resonance correction of MR images," *IEEE Trans Med Imaging*, 1999; 18(6): p.481-495.

[6] D.C. Noll, J.A. Fessler, B.P. Sutton. "Conjugate phase MRI reconstruction with spatially variant sample density correction," *IEEE Trans Med Imaging*, 2005; 24(3): p.325–36.

[7] C.B. Ahn, J.H. Kim, Z.H. Cho. High-speed spiral-scan echo planar NMR imaging-I. *IEEE Trans Med Imaging*, 1986: 5(1): p.2-7.

[8] C.H. Meyer, B.S. Hu, D.G. Nishimura DG, A. Macovski. Fast spiral coronary artery imaging. *Magn Reson Med*, 1992 Dec; 28(2): p.202-13.

[9] J.I. Jackson, C.H. Meyer, D.G. Nishimura, A. Macovski. Selection of a convolution function for Fourier inversion using gridding. *IEEE Trans Med Imaging*. 1991; 10(3): p.473-8.

[10] 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*, 2003 Feb; 22(2): p.178–88.

[11] D.C. Noll, J.A. Fessler, B.P. Sutton. Conjugate phase MRI reconstruction with spatially variant sample density correction. *IEEE Trans Med Imaging*, 2005 Mar; 24(3): p.325-36.

[12] 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 Signal Process,* 2005 Sep; 53(9): p.3393-3402.

[13] G. Liu, S. Ogawa. EPI image reconstruction with correction of distortion and signal losses. *J Magn Reson Imaging*. 2006 Sep; 24(3): p.683-9.

[14] J.A. Fessler, D.C. Noll. Model-based MR Image Reconstruction with Compensation for Through-Plane Field Inhomogeneity. *Proc. IEEE Intl. Symp. Biomed. Imag*, 2007 April; p.920-923.

[15] B.P. Sutton, D.C. Noll, and J.A. Fessler, Compensating for within voxel susceptibility gradients in BOLD fMRI, *Proc Int Soc Mag Res Med*, 2004; p.349.

[16] Y. Zhuo, B.P. Sutton. Iterative Image Reconstruction Model Including Susceptibility Gradients Combined with Z-shimming Gradients in fMRI. *Proc IEEE Eng Med Biol Soc, Minneapolis*, 2009 Sep; p.5721-5724.

[17] S.S. Stone, H. Yi, J. P. Haldar, WmW. Hwu, B.P. Sutton, Z.P. Liang. How GPUs Can Improve the Quality of Magnetic Resonance Imaging, *First workshop on General purpose processing on Graphics processing units (GPGPU),* 2007 Oct.

[18] S.S. Stone, J.P. Haldar, S.C. Tsao, WmW. Hwu, Z.P. Liang, B.P. Sutton. Accelerating Advanced MRI Reconstructions on

GPUs, *J Parallel Distrib Comput*. 2008 Oct; 68(10): p.1307-1318.

[19] WmW. Hwu, D. Nandakumar, J.P. Haldar, I.C. Atkinson, B.P. Sutton, Z.P. Liang, K.P. Thulborn. Accelerating MR Image Reconstruction on GPUs. *Proc IEEE Intl. Symp. Biomed. Imag*, 2009 June; p.1283-1286.

[20] Yue Zhuo, Xiao-Long Wu, Justin P. Haldar, Wen-mei Hwu, Zhi-Pei Liang, Bradley P. Sutton. "Accelerating iterative field-compensated MR image reconstruction on GPUs", *IEEE ISBI 2010 Proceedings*, 2010 Apr.

[21] Yue Zhuo, Xiao-Long Wu, Justin P. Haldar, Wen-mei Hwu, Zhi-Pei Liang, Bradley P. Sutton. "Multi-GPU Implementation for Iterative MR Image Reconstruction with Field Correction", *Proc Int Soc Mag Res Med*., 2010; p.2942.

[22] Michael Garland, "Sparse Matrix Computations on Manycore GPU's," *Proceedings of the 45th annual conference on Design automation*, 2008 June.

[23] Nathan Bell and Michael Garland, "Efficient Sparse Matrix-Vector Multiplication on CUDA, " *NVIDIA Technical Report NVR-2008-004*, 2008 Dec.

[24] Fessler, J.A., Booth, S.D., "Conjugate-Gradient Preconditioning Methods for Shift-Variant PET Image Reconstruction*", IEEE TIP(8), No. 5*, 1999 May; p.688-699.

[25] J. Nickolls, I. Buck. NVIDIA CUDA software and GPU parallel computing architecture. *Microprocessor Forum*, 2007 May.

[26] S. Ryoo, C. I. Rodrigues, S.S. Baghsorkhi, S.S. Stone, D.B. Kirk, WmW. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. *Symp Princ Practice Parallel Programming (PPOPP),* 2008.

[27] F. Wajer, K.P. Pruessmann. Major Speedup of Reconstruction for Sensitivity Encoding with Arbitrary Trajectories. *Proc. 9th Intl. Soc. Mag. Reson. Med*, 2001; p.767.

[28] V. Kindratenko, J. Enos, G. Shi, M. Showerman, G. Arnold, J. Stone, J. Phillips, WmW. Hwu, GPU Clusters for High-Performance Computing, in Proc. Workshop on Parallel Programming on Accelerator Clusters, *IEEE International Conference on Cluster Computing*, 2009.