Sub-second Compressed Sensing Reconstruction for Large Array Data Using GPUs

Ching-Hua Chang[1] and Jim Ji[1]
[1]*Texas A&M University, College Station, Texas, United States*

Purpose: Many clinical applications can benefit from integrating compressed sensing (CS) with parallel receive systems because of the potential to accelerate the data acquisition and/or improve reconstruction quality [1-5]. However, the increased computational complexity required by the advanced CS reconstruction algorithms, particular with large arrays and 3D imaging, is time consuming. Several groups have reported using graphics processing units (GPUs) to accelerate CS reconstruction. However, none has been applied to CS-MRI with parallel imaging [6-7]. The work reported in this paper is to accelerate CS reconstruction from large array data using an alternating direction algorithm and graphics processing units (GPUs).

Methods: In multichannel receiver system, the data from the each channel can be modeled as $v_i = \Phi u_i$ where Φ represents the randomize Fourier encoding matrix, v_i is the acquired data, and u_i is the coil image. Based on CS theory, the image can be recovered by minimizing the Lagrangian function $\|\Phi u_i - v_i\|_2 + \lambda TV(u_i)$, where TV denotes the total variation operator. In this work, this was accomplished with an alternating direction method (ADM) in [8], which involves simple shrinkages and FFTs in a small number of iterations. Note that u_i can be the aliased, reduced-FOV images, which can then by processed by the conventional SENSE reconstruction as shown in [5], or they can be full-FOV images, which can be combined using the sum-of-squares reconstruction. The advantage of this method is that each independent channel image can be parallelized and processed with GPU. In addition it does not require channel sensitivity or extra reference data. This makes it very suitable for process parallel imaging data from large arrays with the CS sampling and reconstruction.

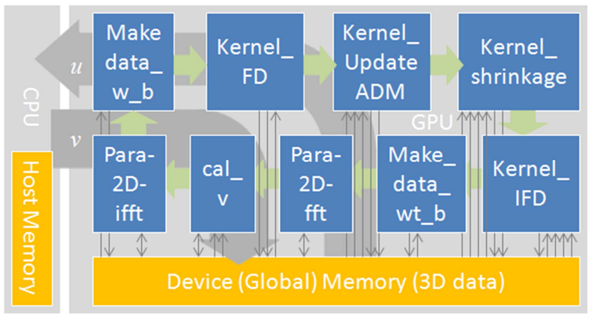

Figure 1. Block diagram and data flow on the GPU for accelerated compressive sensing reconstruction.

The proposed GPU implementation is illustrated in Figure 1. The major sub functions (kernels) include making boundaries (Makedataw_b), finite difference (Kernel_FD), updating ADM multipliers (Kernel_UpdataADM), shrinkages (Kernel_shrinkage), inversed FD (Kernel_IFD), asking data without boundaries (Makedata_wt_b), and 2D fast Fourier transforms (Para_2D_FFTs). FFTs are supported by CUFFT library in CUDA and can simultaneously launch. All other kernels are programmed separately in house. The green arrows represent the data flow on GPU and the gray arrows show the read/write access from the global memory. The fast on-chip shared memory can be utilized to further improve the throughput. In the implementation, one-pixel periodic boundary is prepared for the desired images in the GPU global memory. Therefore, stencil computations arise within these major kernels, and the reconstruction from multichannel 3D data, can be highly parallelized and accelerated. The proposed method was tested using an NVIDIA Geforce GTX 650Ti with 2G DDR5 memory on a desktop computer with an Intel Quadcore i7-3770 3.6 GHz CPU and 32G DDR3 memory. Comparisons are made between reconstruction times with CPU alone or with GPU. First, a 3-D 128×128×16 image was reconstructed from an 8-ch data that are randomly sampled with an acceleration factor of 6. This test is used to profile kernel functions of CPU version compared with the GPU version. In addition, a 3-D, 4-ch data set with 256×256×32 matrix size are truncated to create image data size of 240×240×32, 128×128×16, 64×64×16, and 32×32×4. Each data is then randomly subsampled with an acceleration factor of 6. Speedup factors are calculated based on the total reconstruction times required by CPU alone or with GPU.

Results: Fig.2 shows the run time of CPU (in blue) and GPU (in orange) for the 8-ch data. The speed up factor is about 16. Note that the most improvements are on kernel FD and shrinkage, with a factor of above 30. The inserted pie chart in (b) shows the run time percentage of kernel functions, which shows the FD/IFD, FFT, and shrinkage contribute to 80% of the computations. Table 2 shows the run time and the speedup factor as the data size varies. Note that as the image size becomes larger, the speedup factor increases.

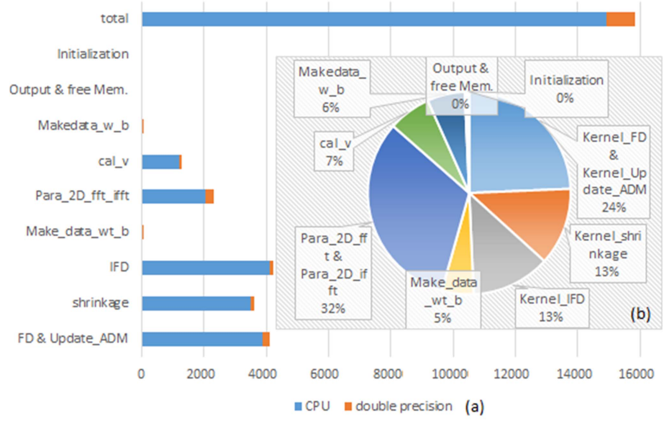

Figure 2: (a) Reconstruction time for CPU vs. GPU (in ms, 40 iterations) for a 128×128×16 3D image from a 8-ch dataset. (b) Inserted pie chart shows the portions of computation loads of the major sub-functions on the

Table 1: Reconstruction time and speedup with a 4-ch dataset (in ms).

[Nx Ny Nz]	CPU	GPU	speedup
256×256×32	67229	2385	28.5
240×240×32	60678	2746	22.1
128×128×16	8795	315	27.9
64×64×16	2497	222	11.2
32×32×4	139	144	0.9

Discussion and Conclusion: Reconstruction time is one of the bottleneck in clinical application of advanced compressive sensing technique with large parallel array data. Multi-core architectures such as GPUs have the potential to make real-time CS-MRI reconstruction from parallel array data possible. The work here demonstrated that it is possible to reconstruct 3D image from parallel imaging data within 1 second using the GPUs, Memory coalescing, constant and shared memory will be explored in the future work to further speedup the reconstruction.

References:

[1] J. Gai, et al. *J Parallel Distributed Comput*. v. 73, pp. 686-697, 2013.
[2] S. Nam, et al. *Mag. Reson. Med.,* v. 69, pp. 91-102, 2013.
[4] D. Liang, et al. *Mag. Reson. Med.,* v. 62, pp.1574-1584, 2009.
[6] D. S. Smith, et al. *Inter. J. Biomed. Imag.* v. 2012, Article ID 864827, 2012.
[8] Y. Junfeng, et al. *IEEE J. Sel. Top. Sig. Proc.,* v.4, pp. 288-297, 2010.
[3] M. Murphy, et al. *IEEE Trans. Med. Imag.*, v. 31, pp. 1250-1262, 2012.
[5] T. S. Sorensen, et al. *IEEE Trans. On Med. Imag.,* v. 28, pp. 1974-1985, 2009.
[7] M. Murphy, et al. *IEEE Trans. On Med. Imag.,* v.31, no.6, pp.1250-1262, 2012.