GPU-Accelerated Radial Image Reconstruction with an Improved Parallel Gridding Method

Jiangsheng Yu[1], Yiqun Xue[1], Xia Zhao[1], Ping Wang[1], Walter Witschey[1], and Hee Kwon Song[1]
[1]*University of Pennsylvania, Philadelphia, PA, United States*

Introduction: Radial acquisition is becoming increasingly popular in MRI due in part to its reduced sensitivity to motion and its capability for high temporal resolution imaging via azimuthal undersampling. In some applications such as DCE-MRI, a continuous scan on the order of ~10 min is typically performed and yields a large dynamic series data set. For example, in a typical 3D DCE-MRI experiment with 32 slices, five channels, 6000 views and 384 readout samples, the size of acquired data set is ~3.0GB. It takes ~30 min on a modern desktop PC to reconstruct this large volume data set. Recently, graphic processing unit (GPU) has been utilized to accelerate the radial image reconstruction. One of the challenges of gridding using massively parallel processing is the issue of synchronization, in which multiple acquired points are gridded onto the same Cartesian grid location. Several approaches were reported to address the synchronization problem in GPU-accelerated gridding reconstruction [1-3]. In this work, a simple and effective parallel gridding algorithm is proposed and tested on two GPU systems for the reconstruction of DCE-MRI dynamic series.

Methods: Convolution gridding is commonly used following radial data acquisition, in which the non-uniformly sampled k-space data is first density compensated, convolved with a kernel to generate the Cartesian k-space points, and then FFT applied to generate the final image [4]. Typically, the gridding algorithm is implemented by distributing the acquired k-space data point to its nearby neighbors, weighted according to the convolution kernel. Each Cartesian k-space point is an accumulation of the contributions from multiple neighboring radial samples (donors), particularly in the highly oversampled region near k-space center. This multiple-donor scenario causes a synchronization problem when GPU is utilized to parallelize the gridding process. For example, if each radial point is assigned a GPU thread, a race condition occurs when two threads try to access the same memory indexed by the Cartesian point. As shown in Figure 1(a), the race condition can occur between acquired points of different views or neighboring points of the same radial view where the Cartesian distribution points are overlapped. Although it is possible to perform atomic memory operation to sequentialize the memory access, this process is very time-consuming and offsets the GPU parallel processing advantages. An alternative approach is a Cartesian k-space driven technique, which assigns a GPU thread for each Cartesian k-space point and searches the entire acquired dataset for its neighboring points within the distance of the convolution kernel [1]. As illustrated in Figure 1(b), this approach avoids the race condition by sequentially collecting the contributions from donors, however, the computation complexity is significantly increased. Some improved methods have been proposed but all involve some overhead of sorting and binning [2]. We present here a simple yet effective method for parallel gridding the dynamic radial k-space data. At the core level, each GPU core is assigned to a 2D radial k-space dataset for each slice or channel. A simple divide-and-conquer approach is used to utilize the thread level parallelism, in which the sampling points of each radial view are divided into multiple groups whose Cartesian neighboring points are non-overlapping. As shown in Fig.1(c), for example, for a 5x5 kernel, a 384-point radial view can be divided into 8 groups (an interleave of every 8 points) with a group size of 48 points, and the number of threads for parallel gridding is 48*5*5=1200, which takes the full advantages of thread capacity of a GPU core.

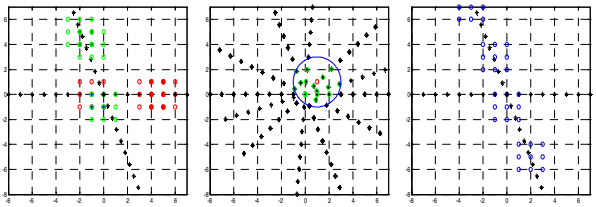

Fig. 1 (a) Synchronization problem in GPU parallel gridding (the solid circles represent the overlapped Cartesian points). (b) Cartesian k-space driven gridding method. (c) The proposed divide-and-conquer method in which the sampling points of each radial view is divided into multiple groups whose Cartesian points are non-overlapping.

Results and Discussion:

Table 1 shows the comparison of reconstruction times on two computer systems for one time frame of a dynamic image series with 32 slices, 5 channels, 60 views, 384 readout samples. To reconstruct a 100 time-point dynamic series with a 7x7 gridding kernel on system 2, it takes the CPU 29.9 min and the GPU 26.9 seconds, respectively, resulting an acceleration factor of 67. It should be noted that the GPU performance can be further improved by distributing the data to utilize all available GPU cores (one time frame in Table 1 only uses 32 (slices) x 5 (channels) = 160 cores).

Table 1 Comparison of reconstruction time for one dynamic series image at two GPU systems (1a : Intel Core i5 CPU M460 @2.53GHz 2.53GHz ; 1b : Intel Xeon CPU E5603 @ 1.60GHz, 1.60GHz 2a : Nvida Geforece M305 2b : Nvida Tesla C2050).

Time (seconds)	System 1 (laptop)			System 2 (desktop)		
	CPU[1a] (core i5)	GPU[2a] (16 cores)	acceleration factor	CPU[1b]	GPU[2b] (448 cores)	acceleration factor
Gridding (5x5 kernel)	12.05	1.64	**7**	8.80	0.116	**76**
Gridding (7x7 kernel)	22.80	3.84	**6**	16.61	0.243	**68**
3D fftw (384x384x32)	1.80	0.59	**3**	1.33	0.0259	**51**

Conclusion: The current work presents a simple and effective parallel gridding method for GPU-accelerated radial image reconstruction. The parallel gridding takes advantages of GPU core level and thread level parallelism. At the core level, the 2D radial k-space data for each slice or channel is assigned to a different core. A divide-and-conquer method is proposed to separate each radial view into multiple interleaved groups whose Cartesian neighboring points are non-overlapping. The tests on the reconstruction of DCE-MRI dynamic series show that an acceleration factor of 67 can be achieved on an Nvida Tesla C2050 GPU system.

Acknowledgments: American Cancer Society RSG-08-118-01-CCE; NIH P41-RR02305; NIH R01-CA125226; NIH UL1RR024134.

Reference: (1) C. I. Rodrigues, et al, Proceedings of the 2008 Conference on Computing Frontiers, ACM, 2008:273-282. (2) N. M. Obeid et al. ISMRM 2011:2547. (3) A Lu, et al, MRM, 2010; 63:1583-1593. (4) J.I. Jackson, et al, IEEE Transactions on Medical Imaging, 1991; 10:473-478.