GPU-Accelerated Gridding for Rapid Reconstruction of Non-Cartesian MRI

N. M. Obeid[1], I. C. Atkinson[2], K. R. Thulborn[2], and W-M. W. Hwu[1]
[1]Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL, United States, [2]Center for Magnetic Resonance Research, University of Illinois at Chicago, Chicago, IL, United States

Introduction: Non-Cartesian acquisitions are commonly used for MR imaging of short-T2 species, such as 23-sodium or 17-oxygen where an ultra-short echo time is essential for quantification, and for rapid 2-D and 3-D MR imaging of anatomy, physiology, or function. Since the Fast Fourier Transform (FFT) cannot be directly applied to non-Cartesian k-space data, alternate methods must be used for image reconstruction. Gridding-based image reconstruction allows the FFT to be used after interpolating the non-Cartesian data onto a Cartesian grid [1]. However, despite the fact that the algorithmic complexity of gridding is O(N) (Big-O notation), this step can still take significant time when there are millions of sample points or when gridding is performed multiple times as part of a multi-frequency reconstruction. We propose a method to perform gridding using a graphics processor (GPU) to achieve up to 29X acceleration for 3-D gridding.

Method: Sequentially computed gridding is traditionally done in an input-driven (or scatter) approach, where every sample point contributes to all of the grid points within the neighborhood defined by the gridding kernel width [1]. The contribution to grid points beyond the kernel width is zero, which results in an O(N) algorithm instead of an O(NM) one, where N is the number of non-Cartesian input sample points and M is the number of Cartesian grid output elements (figure 1.a). Using the scatter approach for parallel execution assigns each input point to a single processing element. This results in multiple processing elements potentially writing the same output grid point simultaneously, which will lead to incorrect results without time consuming synchronization. The alternative approach is an output-driven (or gather) algorithm, where every output is computed by a single processing element and all the processing elements share the input elements in a read-only manner (figure 1.b). Similar to the scatter approach, the gridding kernel defines the neighborhood that contributes to an output point. However, since the sample point locations cannot simply be inferred, every output element must check each of the N sample points to determine which fall within its cutoff before computing their contribution. The result is an O(NM) algorithm, despite the amount of useful computation being only on the order of O(N). Input binning [2] helps reduce the complexity of the gridding algorithm to O(N) by first sorting the sample points into bins of known dimensions and offsets within the k-space and uniform capacity. This enables each processing element to efficiently access only the bins that fall within the neighborhood of the grid point that is being computed. In contrast with the application described in [2], the sample distribution in k-space is often non-uniform, with the center of k-space highly oversampled. Maintaining bins of uniform capacity would require a large amount of padding (mock samples added to the bins to yield uniform sampling density), which can easily become a disabling factor in terms of memory requirements when the sampling density is highly non-uniform (figure 1.c). The solution is to allow bins to contain a variable number of sample points within them, without sacrificing rapid access to the bins by the output-bound processing elements (figure 1.d). To further improve the load imbalance due to varying bin sizes, the computation is distributed evenly between the CPU and the GPU to result in even better performance.

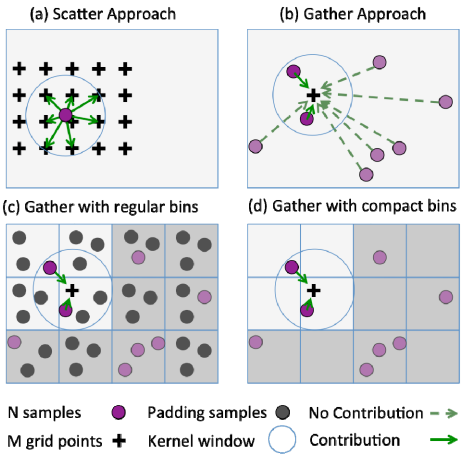

Figure 1. Depictions of the different gridding algorithms discussed in the methods section

	Image 1	Image 2	Image 3
Input samples	5111916	2655910	30144488
Output points	256^3	256^3	512^3
Kernel width (voxels)	6.74	8.53	3.00
Multi-Frequency Iterations	32	1	1
GPU time (s)	30.14	1.00	2.80
CPU time (s)	727.08	29.82	12.62
Speedup	**24.12X**	**29.82X**	**4.51X**

Table 1. Speedup results for three images

Experimental Results and Conclusion: Table 1 compares our GPU-accelerated algorithm to a hand-optimized, single-thread CPU version on a 2.27Ghz Intel Xeon CPU and an NVIDIA C2050 GPU for three 3-D non-Cartesian acquisitions (Images 1 and 2: flexTPI of 23-sodium **[3]**; Image 3: T2-weighted VIPR **[4]**). A reduction in execution time of 11.5 minutes (720.08 s to 30.14 s, >24X speedup) was obtained for a multi-frequency reconstruction with ~$5x10^6$ input samples (image 1). For a smaller dataset with ~$2.65x10^6$ input samples (image 2) a >29 X speedup was achieved. When applied to a large dataset with >$30x10^6$ input samples and a 512^3 grid matrix (image 3) a >4.5 X speedup was achieved. This speedup on the gridding of 3-D image reconstruction makes possible fast reconstruction of non-Cartesian MR imaging data.

References: **[1]** J.I. Jackson, et al, *IEEE Transactions on Medical Imaging*, 1991; 10:473-478 **[2]** C. I. Rodrigues, et al, *Proceedings of the 2008 Conference on Computing Frontiers, ACM*, 2008:273-282. **[3]** A Lu, et al, MRM, 2010; 63:1583-1593. **[4]** A Lu, et al. MRM 2005; 53:692-9.