Multi-GPU Reconstruction of Dynamic Compressed Sensing MRI

Tran Minh Quan, Sohyun Han, Hyungjoon Cho, and Won-Ki Jeong

E-mail: {quantm, hansomain, hcho, wkjeong}@unist.ac.kr
Ulsan National Institute of Science and Technology (UNIST) *

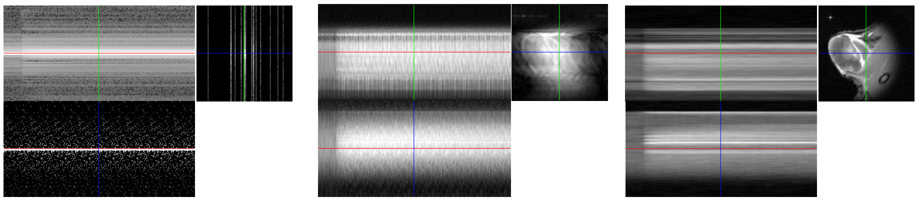
Abstract. Magnetic resonance imaging (MRI) is a widely used in-vivo imaging technique that is essential to the diagnosis of disease, but its longer acquisition time hinders its wide adaptation in time-critical applications, such as emergency diagnosis. Recent advances in compressed sensing (CS) research have provided promising theoretical insights to accelerate the MRI acquisition process, but CS reconstruction also poses computational challenges that makes MRI less practical. In this paper, we introduce a fast, scalable parallel CS-MRI reconstruction method that runs on graphics processing unit (GPU) cluster systems for dynamic contrast-enhanced (DCE) MRI. We propose a modified Split-Bregman iteration using a variable splitting method for CS-based DCE-MRI. We also propose a parallel GPU Split-Bregman solver that scales well across multiple GPUs to handle large data size. We demonstrate the validity of the proposed method on several synthetic and real DCE-MRI datasets and compare with existing methods.

1 Introduction

Magnetic resonance imaging (MRI) has been widely used as an in-vivo imaging technique due to its safety to living organisms. Because the acquisition process is the major bottleneck of MRI, many acceleration techniques have been developed using parallel imaging techniques [2]. Recently, the compressed sensing (CS) theory [3] has been successfully adopted to MRI to speed up the acquisition process [10]. However, CS-MRI introduces additional computational overhead in the reconstruction process because the ℓ_1 minimization is a time-consuming nonlinear optimization problem. Therefore, there exists a need to develop fast CS reconstruction methods to make the entire MRI reconstruction process practical for time-critical applications.

One research direction for accelerating CS reconstruction has focused on developing efficient numerical solvers for the ℓ_1 minimization problem [7, 5, 1]. The other direction has been leveraging the state-of-the-art parallel computing hardware, such as the graphics processing unit (GPU), to push the performance to the limit [9, 13, 1]. We believe that GPU acceleration is the most promising approach to make CS-MRI reconstruction clinically feasible, but multi-GPU acceleration has not been fully addressed in previously published literature.

* Corresponding author: wkjeong@unist.ac.kr



(a) ×8 subsampled k-space (b) Zero padding recon. (c) CS-DCE-MRI recon.

Fig. 1: CS-DCE-MRI examples. Red line: time axis, Blue–green lines: x–y axis

In this paper, we introduce a novel multi-GPU reconstruction method for CS-based dynamic contrast-enhanced (DCE) MRI. DCE-MRI is widely used to detect tumors, and its diagnostic accuracy highly depends on the spatio-temporal resolution of data. Fig. 1 shows an example of CS-based DCE-MRI reconstruction from a sub-sampled k-space data. CS has been successfully adopted to DCE-MRI to increase the resolution of data, but CS also imposes a high computational burden due to the increasing data size. In addition, we observed that a popular numerical method for CS reconstruction, i.e., Split-Bregman, is not directly applicable to the CS-based DCE-MRI formulation because the time and spatial domains of DCE-MRI data cannot be treated equally.

The main contributions of this paper can be summarized as follows. First, we present a modified Split-Bregman iteration to solve CS-based time-varying 2D DCE-MRI reconstruction problems. The proposed method splits time (1D) and spatial (2D) operators in the sub-optimization problem in the original Split-Bregman method, which can be efficiently solved by a single-step implicit integration method. The proposed method runs 3× faster than the conjugate gradient method and 5× faster than the fixed-point iteration method, which makes the entire reconstruction process converge up to about 7× faster than the state-of-the-art GPU DCE-MRI method [1]. Second, we introduce a scalable GPU implementation of the proposed reconstruction method and further extend to a multi-GPU system by leveraging advanced communication and latency hiding strategies. We demonstrate the performance of the proposed method on several CS-based DCE-MRI datasets with up to 12 GPUs.

2 Method

The input to our system is 2D DCE MRI data (a collection of 2D k-space data over time) defined on a 3D domain by the parameter (x, y, t), where x and y are spatial coordinates for 2D k-space data, and t denotes a temporal coordinate on the time axis. In the following, the subscript of an operator represents its dimension, e.g., F_{xy} stands for a Fourier operator applied to a 2D x-y grid.

2.1 Compressed Sensing formulation for DCE-MRI

Let U be a collection of 2D MRI k-space data u_i, i.e., $U = \{u_1, u_2, ..., u_n\}$, acquired over time. Then the general CS formulation for DCE-MRI reconstruction problem can be described as follows:

$$\min_U \{J(U)\} \quad s.t. \quad \sum_i \|Ku_i - f_i\|_2^2 < \sigma^2 \tag{1}$$

where $J(U)$ is the ℓ_p-norm energy function to minimize (i.e., regularizer), f_i is the measurement at time i from the MRI scanner, and $K = RF$ is the sampling matrix that consists of a sampling mask R and a Fourier matrix F for 2D data. Since DCE-MRI does not change abruptly along the time axis, we enforce the temporal coherency by introducing a total variation energy along t axis and decouple the sparsifying transform on the x-y plane and temporal axis as follows:

$$J(U) = \|W_{xy}U\|_1 + \|\nabla_{xy}U\|_1 + \|\nabla_t U\|_1 \tag{2}$$

where W is the wavelet transform. Then Eq. 1 can be solved as a constrained optimization problem using a Bregman iteration [5] where each update of U^{k+1} in line 3 solves an unconstrained problem as shown in Algorithm 1. Using the Split Bregman algorithm [5], we can decouple ℓ_1 and ℓ_2 components in line 3 in Algorithm 1 and iteratively update using a two-step process (more details can be found in [5]). Note that in the original Split-Bregman algorithm, U and $J(U)$ are defined on the same dimensional grid, i.e., if U is a 3D volume then $J(U)$ is a hybrid of 3D total variation (TV) and wavelet regularizers, which allows a closed-form solution for U^{k+1}. However, because our $J(U)$ consists of 1D and 2D operators, the same closed-form solution is not applicable. To be more specific, U^{k+1} can be updated by solving the linear system defined as follows:

Algorithm 1 Constrained CS Optimization Algorithm

```
1: $k = 0, u_i^0 = f_i^0 = \mathbf{0}$ for all $i$
2: while $\sum_i \left\| R_i F_{xy} u_i^k - f_i \right\|_2^2 > \sigma^2$ do
3:     $U^{k+1} = \min_U \{J(U) + \frac{\mu}{2} \sum_i \left\| R_i F_{xy} u_i - f_i^k \right\|_2^2\}$
4:     $f_i^{k+1} = f_i^k + f_i - R_i F_{xy} u_i^{k+1}$ for all $i$
5: end while
```

$$(\mu F_{xy}^{-1} R^T R F_{xy} - \lambda \mathit{\Delta}_{xy} - \theta \mathit{\Delta}_t + \omega) U^{k+1} = rhs^k \tag{3}$$

where the parameters λ, θ, ω and μ are used to control the amount of regularization energy, and refer to [5] for the definition of rhs^k. Goldstein et al. [5] proposed a closed form solution to invert the left-hand side of the given linear system using the forward and inverse 3D Fourier transforms, but we cannot apply the same method since the k-space data is 2D in our formulation.

Since the left-hand side of Eq. 3 is not directly invertible, we can use iterative methods, such as the fixed-point iteration or the conjugate gradient method, that converge slower than the explicit inversion in the original method. In order to speed up the convergence, we propose a *single iteration* method, which further reduces the iteration cost by splitting the left-hand side of Eq. 3 and solve for

U^{k+1} directly in a single step. If we separate 1D and 2D operators in the left-hand side, then Eq. 3 can be represented as follows:

$$(A_1 + A_2)U^{k+1} = rhs^k \tag{4}$$

where $A_1 = (\mu F_{xy}^{-1} R^T R F_{xy} - \lambda \Delta_{xy})$ and $A_2 = -\theta \Delta_t + \omega$. If we treat A_2 as the operator applied to U from the previous iteration, i.e., U^k at (k+1)-th iteration, then Eq. 4 can be expressed as follows:

$$A_1 U^{k+1} + A_2 U^k = rhs^k. \tag{5}$$

Because U^k is known, we can move it to the right-hand side to make the update rule for U^{k+1}

$$U^{k+1} = A_1^{-1}(rhs^k - A_2 U^k). \tag{6}$$

This assumption holds for a sufficiently large k because U^k and U^{k+1} will converge. Then, A_1 can be inverted by making the system circulant as shown in [5]

$$U^{k+1} = F_{xy}^{-1} \mathcal{K}^{-1} F_{xy}(rhs^k + (\theta \Delta_t - \omega)U^k) \tag{7}$$

where $\mathcal{K}$ is the diagonal operator defined as $\mathcal{K} = \mu R^T R - \lambda F_{xy} \Delta_{xy} F_{xy}^{-1}$.

We evaluated the convergence rates of three different minimization strategies for U^{k+1} – fixed-point iteration, conjugate gradient, and single iteration methods (Fig. 2). We observed that the conjugate gradient method converges about twice faster than the fixed-point iteration, and our single iteration converges about 2–3-fold faster than the conjugate gradient method to reach the same PSNR.

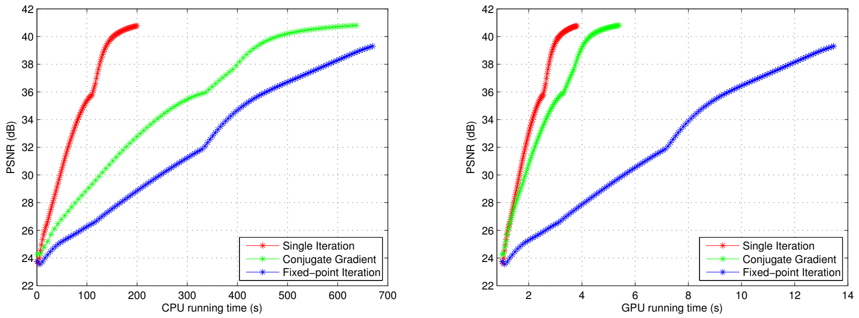


Fig. 2: PSNR vs. CPU (left) and GPU running times (right) of various methods.

2.2 Multi-GPU Implementation

As the temporal resolution of DCE-MRI increases, the entire k-space can not be mapped onto a single GPU. We decompose the domain along the t-axis into chunks where each chunk consists of three parts: *interior*, *exterior*, and *ghost*, similar to [12]. Interior and exterior regions form a computational domain for a given chunk, and ghost is the region extends to its neighbor chunk that is

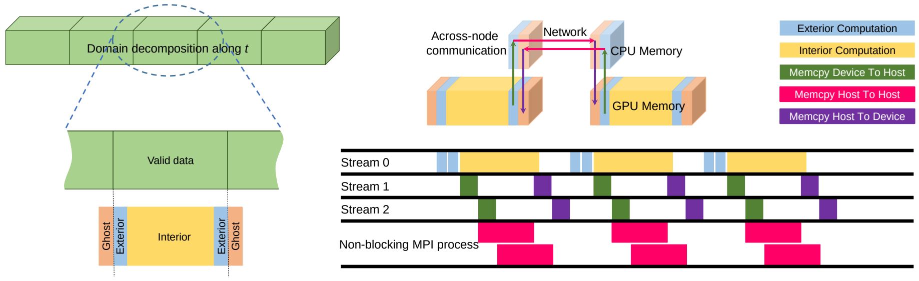


Fig. 3: Ghost region communication between neighbour GPUs

required for computation (Fig. 3 left). In our method, we increase the ghost and exterior size (up to 5 slices in our experiment) in order to run the algorithm multiple iterations without communication.

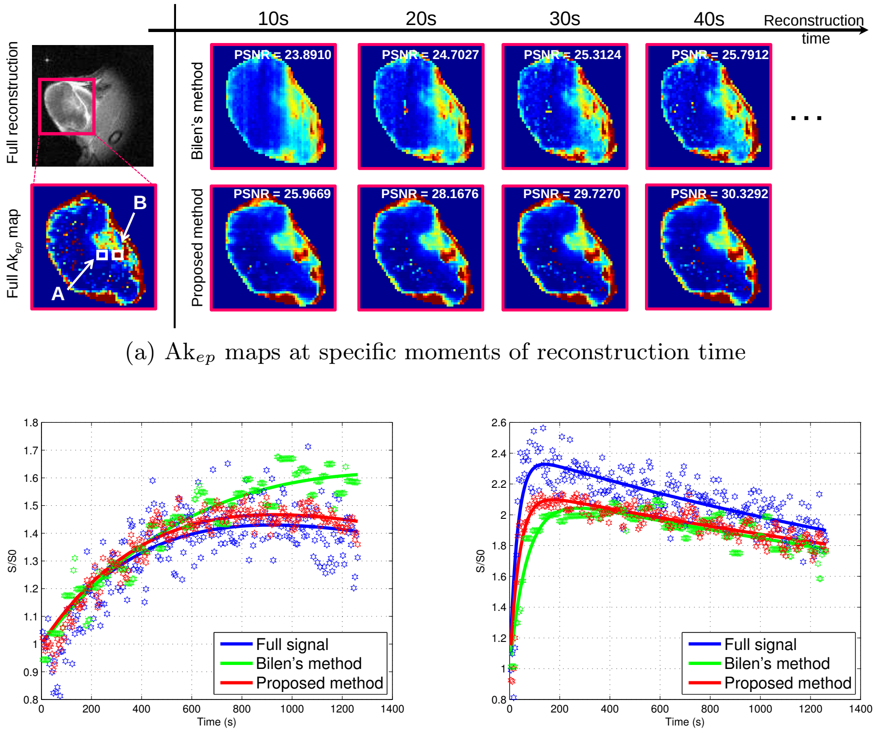
Data communication between adjacent GPUs is performed as follows (see Fig. 3 bottom right): First, we perform the computation on the exterior regions, which are the ghost regions of the neighborhood GPU, using the main stream (stream 0, Fig. 3 cyan). Note that we can run n iterations in this step for the ghost size n (one slice of ghost region is invalidated per each iteration, so we can run up to n iterations). Next, the other concurrent streams (stream 1 and 2) will perform the peer copies from GPU (device) to CPU (host) memory, one stream per an exterior region (Fig. 3 green). Because streams run in parallel, stream 0 can continue to compute on the interior region during the communication (Fig. 3 yellow). Then each exterior region data on the CPU will be transferred to the ghost region of its neighborhood CPU via MPI send and receive (Fig. 3 magenta). Because the MPI process becomes available right after invoking the GPU asynchronous calls, it can be used to perform a non-blocking transaction across the GPUs (note that stream 2 is overlapped with MPI transfer). Finally, the valid ghost data is copied from the host to device asynchronously to complete one round of ghost communication (Fig. 3 purple). By doing this, we can effectively hide the communication latency by allowing the ghost transfers occur on the different GPU streams while the main stream continues the computation on the interior region. If GPUs are in the same physical node, this communication can be implemented using asynchronous peer-to-peer device communication via PCI bus. In addition, if the system supports infiniband network, we can use NVIDIA GPUDirect for RDMA communication.

3 Result

We used an NVIDIA GeForce GTX 680 GPU with Jacket for MATLAB for the single GPU evaluation (same environment as Bilen's). We ran our scalability test on a 6-node GPU cluster system with two NVIDIA K20m per node, 12 GPUs in total. Gd-DTPA contrast agent was used for data preparation, and the CS-undersampling factor is ×8.

3.1 Image Quality and Running time Evaluation

We compared our methods with IMPATIENT [4], kt-SPARSE [11], kt-FOCUSS [8], GPU Split-Bregman [9] and GPU ADMM [1], and our method is comparable to or outperforms those. Among them, we discuss Bilen et al. [1] in detail because it is the most recent work close to our method in the sampling strategy, energy function, numerical method, and GPU implementation. For a direct comparison with Bilen's, the wavelet term in the formulation is ignored in this experiment.



(a) Ak$_{ep}$ maps at specific moments of reconstruction time

(b) Temporal profile of A

(c) Temporal profile of B

Fig. 4: Comparison between Bilen's method and the proposed solution

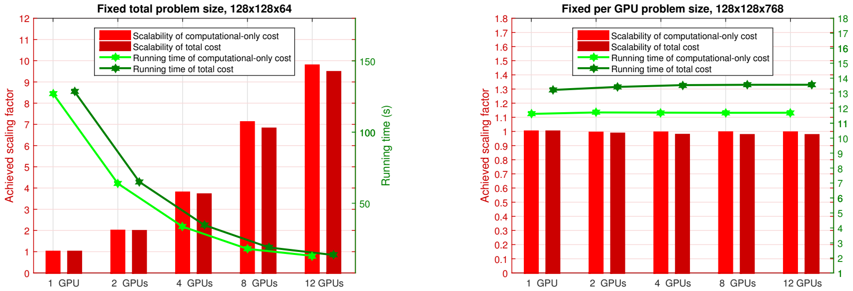
Fig. 4 (a) compares Ak$_{ep}$ maps [6] of reconstructed images at different points in time. The Ak$_{ep}$ maps in the bottom row (our method) is much closer to the ground truth (Full Akmap on the left) than those in the upper row at the same point in time, which implies our method converges faster. Fig. 4 (b) and (c) show the least-square fitting curve to the temporal profile of the region of interest in the Ak$_{ep}$ map (A and B, high vascularity regions), which is also a commonly used quality measure to characterize the perfusion and permeability in DCE-MRI data. For this test, we ran each method until its reconstruction image reaches the same PSNR. While both methods generate the curves similar to that of the full reconstruction, our result is more accurate (i.e., close to the full reconstruction curve), which is due to the different numerical characteristics of the algorithm. Table 1 shows the running time of each method on several tumor datasets until convergence (i.e., reaching a steady state). The result also confirms that our method converges much faster than Bilen's, up to 6.9× in GPU implementation.

Table 1: Running times of Bilen's and the proposed method on various datasets of size 128 × 128 × 256.

Metrics	Tumor 1		Tumor 2		Tumor 3		Tumor 4	
	Bilen	Proposed	Bilen	Proposed	Bilen	Proposed	Bilen	Proposed
PSNR(dB)	30.598	30.636	40.169	39.931	39.678	39.479	33.559	34.174
CPU time(s)	448.451	360.689	797.056	340.143	719.187	327.868	715.004	407.461
GPU time(s)	292.341	61.637	475.155	68.002	394.241	66.125	367.759	63.276

3.2 Multi-GPU Performance Evaluation

In this evaluation, we check the scalability up to 12 GPUs on a distributed GPU cluster. We first measure the computation-only time to obtain the best possible running times, and then measure the total running times including data communication (i.e., ghost region exchange). As shown in Fig. 5, the total time including the ghost exchange is approximately close to the computational-only time, in both strong and weak scaling tests. This result confirms that our multi-GPU implementation can effectively hide the communication latency while performing the CS DCE-MRI reconstruction solver on distributed systems.



(a) (b)

Fig. 5: Strong (a) and weak scaling (b) on a distributed GPU cluster.

4 Conclusion

In this paper, we presented our new CS-based DCE-MRI reconstruction system for the multi-GPU computing environment. The proposed method delivered a new numerical method in order to apply the Split-Bregman algorithm to CS-based time-variant DCE-MRI problem. We also introduced a scalable implementation of the proposed CS-MRI reconstruction method on a distributed multi-GPU system. As discussed, the proposed method outperforms the existing GPU CS-reconstruction algorithm in quality and running time. For future work, we plan to extend the proposed CS-MRI method to large-scale dynamic 3D DCE-MRI reconstruction. Assessing the clinical feasibility of the proposed method would be another interesting future research.

Acknowledgements. This work was partially supported by the National Research Foundation of Korea grant NRF-2012R1A1A1039929 and NRF- 2013K2A1 A2055315.

References

1. Bilen, C., Wang, Y., Selesnick, I.: High-speed compressed sensing reconstruction in dynamic parallel MRI using augmented lagrangian and parallel processing. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2(3), 370–379 (Sep 2012)
2. Blaimer, M., Breuer, F., Mueller, M., Heidemann, R.M., Griswold, M.A., Jakob, P.M.: SMASH, SENSE, PILS, GRAPPA: how to choose the optimal method. Topics in magnetic resonance imaging : TMRI 15(4), 223–236 (Aug 2004)
3. Donoho, D.: Compressed sensing. IEEE Transactions on Information Theory 52(4), 1289–1306 (2006)
4. Gai, J., Obeid, N., Holtrop, J.L., Wu, X.L., Lam, F., Fu, M., Haldar, J.P., Hwu, W.m.W., Liang, Z.P., Sutton, B.P.: More IMPATIENT: a gridding-accelerated toeplitz-based strategy for non-cartesian high-resolution 3D MRI on GPUs. Journal of parallel and distributed computing 73(5) (May 2013)
5. Goldstein, T., Osher, S.: The split bregman method for $\ell 1$–regularized problems. SIAM Journal on Imaging Sciences 2(2), 323–343 (Jan 2009)
6. Hoffmann, U., Brix, G., Knopp, M.V., Hess, T., Lorenz, W.J.: Pharmacokinetic mapping of the breast: a new method for dynamic MR mammography. Magnetic Resonance in Medicine 33(4), 506–514 (Apr 1995)
7. Jung, H., Sung, K., Nayak, K.S., Kim, E.Y., Ye, J.C.: k–t FOCUSS: a general compressed sensing framework for high resolution dynamic MRI. Magnetic Resonance in Medicine 61(1), 103116 (2009)
8. Jung, H., Ye, J.C., Kim, E.Y.: Improved k–t BLAST and k–t SENSE using FOCUSS. Physics in Medicine and Biology 52(11), 3201 (Jun 2007)
9. Kim, D., Trzasko, J., Smelyanskiy, M., Haider, C., Dubey, P., Manduca, A.: High–performance 3D compressive sensing MRI reconstruction using many–core architectures. Journal of Biomedical Imaging 2011, 2:12:11 (Jan 2011)
10. Lustig, M., Donoho, D., Santos, J., Pauly, J.: Compressed sensing MRI. IEEE Signal Processing Magazine 25(2), 72–82 (2008)
11. Lustig, M., Donoho, D., Pauly, J.M.: Sparse MRI: the application of compressed sensing for rapid MR imaging. Magnetic Resonance in Medicine 58(6), 1182–1195 (2007)
12. Micikevicius, P.: 3D finite difference computation on GPUs using CUDA. In: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. p. 7984. GPGPU-2, ACM, New York, NY, USA (2009)
13. Murphy, M., Alley, M., Demmel, J., Keutzer, K., Vasanawala, S., Lustig, M.: Fast - SPIRiT compressed sensing parallel imaging MRI: scalable parallel implementation and clinically feasible runtime. IEEE Transactions on Medical Imaging 31(6), 1250–1262 (2012)