Compressed Sensing MRI with Multichannel Data Using Multicore Processors

Ching-Hua Chang and Jim Ji*

Compressed sensing (CS) is a promising method to speed up MRI. Because most clinical MRI scanners are equipped with multichannel receive systems, integrating CS with multichannel systems may not only shorten the scan time but also provide improved image quality. However, significant computation time is required to perform CS reconstruction, whose complexity is scaled by the number of channels. In this article, we propose a reconstruction procedure that uses ubiquitously available multicore central processing unit to accelerate CS reconstruction from multiple channel data. The experimental results show that the reconstruction efficiency benefits significantly from parallelizing the CS reconstructions and pipelining multichannel data into multicore processors. In our experiments, an additional speedup factor of 1.6–2.0 was achieved using the proposed method on a quad-core central processing unit. The proposed method provides a straightforward way to accelerate CS reconstruction with multichannel data for parallel computation. Magn Reson Med 64:1135–1139, 2010. © 2010 Wiley-Liss, Inc.

Key words: compressed sensing; array coils; parallel computing; multicore processors; image reconstruction

MRI using conventional Fourier imaging is relatively slow compared with other imaging modalities because the data acquisition speed is limited by hardware capability and signal-to-noise ratio factors. Compressed sensing (CS) is a new sampling and reconstruction technique that allows a sparse signal to be reconstructed from a set of randomly undersampled data, which in turn shortens the data acquisition time (1–3). Recently, Lustig et al. (4) have demonstrated the use of CS in a number of MRI applications with remarkable acceleration.

Because array systems receive signals from multichannel simultaneously, they offer improved signal-to-noise ratio (5,6) or accelerated speed in parallel imaging. Several groups including ours have investigated methods to integrate CS and parallel imaging with array systems (7–12). In these methods, CS and parallel imaging are generally coupled in a large linear system that contains data from all channels. Another novel approach is to apply the CS algorithm to reconstruct an aliased image in each channel, followed by a conventional sensitivity encoding (SENSE) method (13). However, an alternative approach to integrate them is to apply the CS reconstruction to each channel individually, followed by a simple channel combination using a sum-of-squares method.

One significant problem in the aforementioned methods is the computational complexity. CS reconstruction involves nonlinear optimization, which can be time consuming even for regular-sized data and images. With multichannel array systems, the computational complexity will scale with the number of channels. This problem can be partially addressed with improved performance of the central processing units (CPU). Unfortunately, computation power growth using higher CPU clock frequency is limited by the power consumption and physical wire size. Modern processor design is moving toward multicore architecture. The ubiquitously available duo-core, quad-core CPU and graphics processing unit (GPU) offer new platforms for implementing parallel algorithms to accelerate image reconstructions. In Wiggers et al. (14), a conjugate gradient solver was implemented on NVIDIA's G80 GPU (NVIDIA, Santa Clara, CA) G80 GPU. Borghi et al. (15) used the multicore platform to solve the CS reconstruction, which involves l_1 minimization. Other advanced MRI reconstruction algorithms were also implemented on NVIDIA's Quadro FX 5600 GPU (16,17). These previous works shows the tremendous potential of multicore processors for speeding up MRI reconstruction. However, most of the work is for single-channel data. In addition, they usually require significant efforts to program the parallel algorithms on the GPUs. For the multichannel parallel imaging with CS algorithm, the methods delineated elsewhere (7–12) cannot be parallelized straightforwardly as the linear system is coupled. However, the methods discussed in other articles (14–17) can be potentially adopted to accelerate these reconstructions. In contrast to the previous cases, the method by Liang et al. (13) can directly benefit from parallel computing because processing of different channels is decoupled before using the SENSE reconstruction.

In this article, we proposed an accelerated reconstruction procedure for combining CS with the array receive systems, using widely available multicore CPUs to accelerate the image reconstruction. The results show that the reconstruction algorithm can benefit significantly from the parallel computing and multicore architecture. This work is developed from a preliminary conference report (18).

MATERIALS AND METHODS

Based on the CS theory, an image $\mathbf{x}$ can be reconstructed from a reduced set of incomplete k-space data $\mathbf{y}$, where $\mathbf{y} = \Phi\mathbf{x}$ with Φ being the randomly sampled Fourier

Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843–3128.

*Correspondence to: Jim Ji, Ph.D., Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128. E-mail: jimji@tamu.edu

Received 10 December 2009; revised 22 February 2010; accepted 23 March 2010.

DOI 10.1002/mrm.22481
Published online 17 June 2010 in Wiley Online Library (wileyonlinelibrary.com).

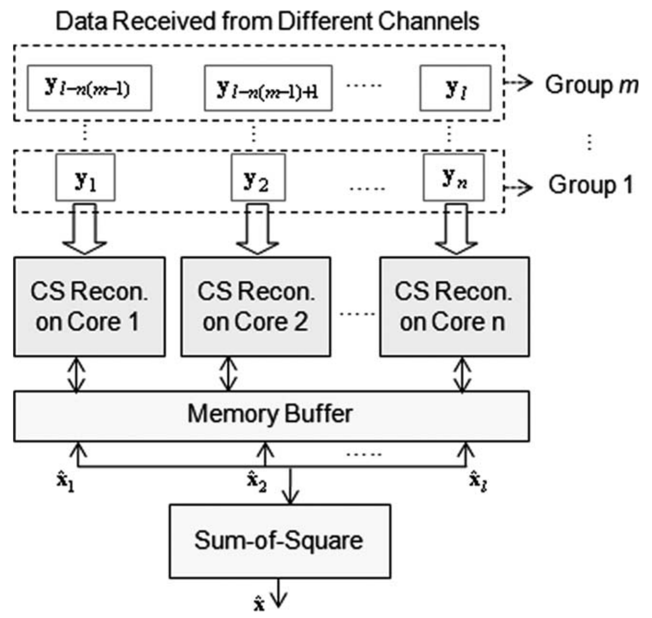

FIG. 1. Illustration of the proposed algorithm on a multicore CPU. Undersampled multichannel data are input to the *n* cores. The final image is obtained by combining all individual channel images.

transform operator implemented by encoding gradients. Specifically, the image **x** can be recovered by the following constrained optimization problem

$$\min\ (\|\Psi\mathbf{x}\|_1 + \alpha\ \mathrm{TV}(\mathbf{x})) \qquad \text{subject to } \|\mathbf{y} - \Phi\mathbf{x}\|_2 < \varepsilon \quad [1]$$

Here, Ψ is a linear transform such as the wavelet transform and TV(**x**) is the total variation of the image. Both terms reflect the sparsity of the image. Note that parameter α is to balance the two terms and parameter ε corresponds to the data fidelity, which can be set to zero for simplicity. With an array receiver system, a *k*-space data set will be acquired from each channel. An image $\mathbf{x}_k$ can be reconstructed from the *k*th channel data, using the CS method. Then, the images from all channels can be combined using the well-known sum-of-squares method (5). A desirable feature of this approach is that individual channels are not coupled, which allows parallel reconstruction using multicore architecture.

The overall reconstruction algorithm is illustrated in Fig. 1. Here, $\mathbf{y}_k$, $k = 1, 2, \ldots, l$, are the undersampled data received from the *l* parallel channels. The data sets $\mathbf{y}_k$ are clustered into *m* groups. Each of the first $m - 1$ groups contains *n* data sets, and the last group *m* contains the rest $l - n(m - 1)$ data sets. Note that *n* corresponds to the number of CPU cores intended to be used. In reconstruction, *m* groups will be processed sequentially, i.e., the first group of *n* data sets is fed to the *n* available processing cores simultaneously, followed by the second group, and so on. After all *m* groups are processed and all *l* individual channel images are reconstructed, the final image is obtained by the sum-of-squares combination. In each processing core, the reconstruction for the *k*-th channel data is obtained by recasting Eq. 1 as

$$\min\ \|\Phi\hat{\mathbf{x}}_k - \mathbf{y}_k\|_2 + \lambda_1\|\Psi\hat{\mathbf{x}}_k\|_1 + \lambda_2\ \mathrm{TV}(\hat{\mathbf{x}}_k) \qquad k = 1, 2, \ldots, l \quad [2]$$

In this article, Sparse MRI software (19) was used for optimization, whose algorithms are based on a nonlinear conjugate gradient method. The regularization parameters λ_1 and λ_2 are set to 0 and 0.001 experimentally.

The algorithm was tested on a platform that contained a standard Intel Core 2 Quad Q8200 2.33-GHz CPU, with a 4-MB L2 cache and 4-GB DDR2 memory. All computer simulation and image reconstruction were performed on Matlab (MathWorks, Natick, MA; build 7.7.0.471). In the default mode, there is no multithread computation supporting of the Parallel Computing Toolbox; a Matlab process is only able to utilize 25% of quad-core CPU. To initialize the parallel process as shown in Fig. 1, a method similar to the message passing interface in (20) was used, which allows multiple Matlab processes run on parallel computer clusters or multicores. In the proposed method with quad-core CPU, four instances of Matlab are started at the beginning, each initiating a process. A primary process in one instance will save the *k*-space data and reconstruction parameters for each channel in a separate data file and in the end will combine all channel images. As soon as the files are ready, the primary and the other three salve processes will simultaneously access and reconstruct the channel data. This parallel processing will finish until all channel files are accessed and processed. At this point, the primary process, after realizing that there are no more individual channel data to be processed, will perform the sum-of-squares combination. To simplify programming and boost the efficiency, all processes communicate through file reading/writing (R/W) only. This also avoids the potential conflicts between processes and ensures that different processes do not access the same channel data.

To test the proposed algorithm, both simulated and in vivo data were used. The multiple-channel (4-channel, 8-channel, and 16-channel) *k*-space data were simulated using the "Shepp–Logan" phantom and gaussian channel sensitivities. For the four-channel data, five data sets with different sizes (32 × 32, 64 × 64, 128 × 128, 256 × 256, and 512 × 512) were synthesized. This was to test the effect of image size on saving the computation time and the overhead time. The individual channel data were undersampled in the *k*-space with radial sampling pattern. The undersampling factor was about 40%, which meant that only 40% of the total data were kept. For the 8-channel and 16-channel simulations, only 256 × 256 images were used, with an undersampling factor of 33%. In addition, for simulated data, four sets of results were generated by using one, two, three, or four cores in the CPU. The total reconstruction times were recorded and compared. The total time includes time for launching and communicating through file R/W and computation starvation, which is defined as the minimum absolute time that the primary or slaves wait for the data before continuing on the next operations. Note

Table 1
Results From the Simulated Four-Channel Study*

	Cores			
Size	1	2	3	4
a: Total computation time				
16 × 16	12.3	7.7	7.4	6.2
32 × 32	17.8	13.2	11.3	10.1
64 × 64	40.3	26.8	26.3	22.5
128 × 128	125.0	76.8	81.2	59.3
256 × 256	567.0	325.6	334.6	248.9
b: Computation speedup factors				
128 × 128	0.8	1.4	1.4	1.8
256 × 256	0.8	1.3	1.3	2.0
512 × 512	0.7	1.2	1.1	1.6

*__a:__ The total computation time (in seconds) in the simulated four-channel study with a quad-core CPU when different numbers of cores are used. **b:** Computational speedup factors of the proposed method in the four-channel study compared with the reconstruction using the quad-core CPU without the proposed method. Note that the acceleration is greater than 1 even with two cores, using the proposed method.

that if multiple cores were working simultaneously, we only recorded the maximum stalled time and the maximum overhead time on file R/W among different cores of CPU. In addition, the computation speedup factor was evaluated by comparing the reconstruction time with and without the proposed method.

Finally, an eight-channel in vivo brain MR data set was acquired and tested. The *k*-space data with size of 256 × 256 from each channel were acquired in full field of view, i.e., without undersampling. Then, the radial sampling was simulated by decimation with an undersampling factor of 33%. The undersampled eight-channel data were then input to the quad-core processor for image reconstruction, as shown in Fig. 1. To evaluate the efficiency of using multicore parallel reconstruction, the total reconstruction times for both simulated data and in vivo data were compared with the time required using the conventional reconstruction method, where the quad-core CPU is used without the proposed parallelization. To reduce the performance variation, each reconstruction test was repeated 10 times and the average time cost was calculated and shown.

RESULTS

Table 1a lists the total reconstruction time for all experiments with the simulated four-channel data, including both time cost for file R/W and the CS reconstruction. As shown, using two cores leads to significant reduction in the computation time per core, with the reduction factor of about 1.7. In addition, using four cores gives the shortest reconstruction time, where the average reduction factor is about 2.4. It is worth mentioning that for the simulated four-channel data, there is very little benefit from using three cores compared with using two cores. This is because in both cases, two groups of data sets need to be processed. With three cores, there are two cores stalling during the second group, which contains only one channel data being processed. Table 1b shows the computation speedup factors of the proposed method. Note that the proposed parallel computation provides about 1.3 times speedup when using two cores and about 1.8 times speedup when using four cores, shown in Table 1b factor 128 × 128 data sets.

Figure 2 illustrates the total reconstruction time in the four-channel simulated study in these parts: CS reconstruction, computation starvation time (stall), and communication time through file R/W. Comparing subplots for large image sizes with that of image size 32, the overheads of file R/W and the stall time take a larger

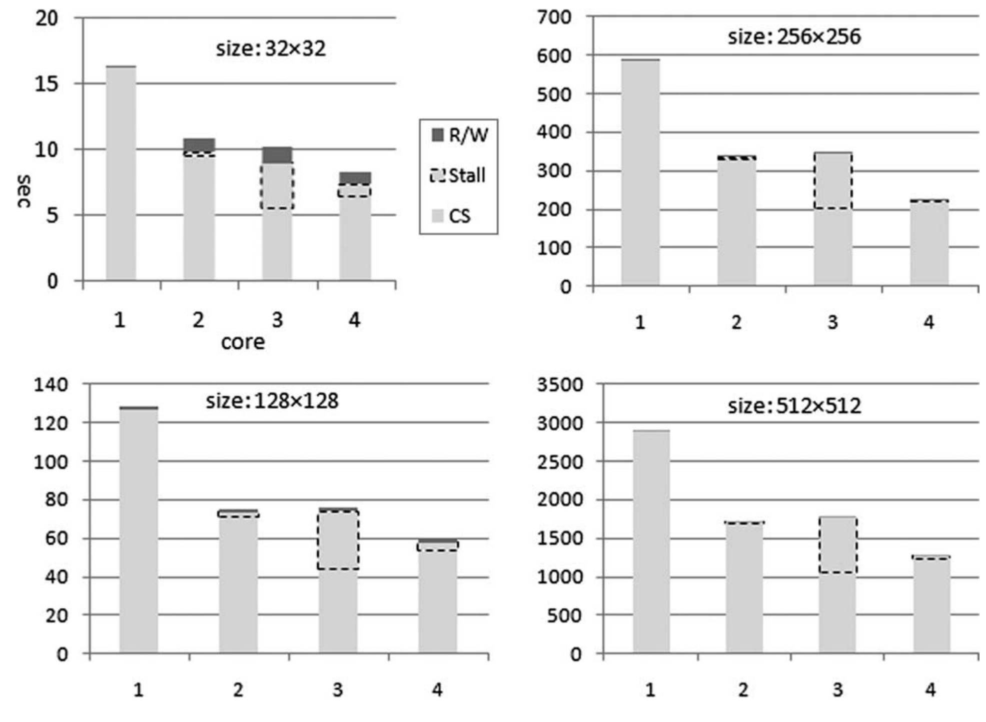

FIG. 2. Total computation time shown in portions: CS reconstruction, computation starvation (cores stall), and file R/W. The horizontal axis represents the numbers of cores in use and the vertical axis represents the computation time.

Table 2
Results From the Simulated 8-Channel and 16-Channel Studies*

Data	Cores 1	2	3	4
a: Total computation time				
8-Channel	1194.0	675.4	579.4	513.7
16-Channel	2317.8	1358.1	1170.1	1017.5
b: Computation speedup factors				
8-Channel	0.8	1.3	1.6	1.7
16-Channel	0.8	1.3	1.6	1.8

***a:** Total computation time (in seconds) when different numbers of cores are used. **b:** Computational speedup factors compared with the reconstruction using the quad-core CPU without the proposed method.

percentage of the total computation time. However, when images become larger, the percentages of these overheads are dramatically reduced and the parallelization of CS reconstructions gains more benefits. Also, using two cores has less stall time, which is implicitly contained in CS reconstruction time, compared with using three cores and four cores. Therefore, it provides maximum efficiency improvement per core, whereas using three cores has the minimum efficiency improvement per core.

Table 2a shows the time of image reconstructions of simulated 8-channel data and 16-channel data using different number of cores of CPU. As shown in Table 2b, the computation speedup factor of using the proposed method is calculated. The range of speedup factors in Table 2b is similar to that of Table 1b, given minimum 0.8 times speedup and maximum 1.8 times speedup, compared with the conventional method using the quad-core CPU without the proposed parallelization.

Figure 3 shows the image reconstruction in the eight-channel in vivo human brain imaging experiments. The sum-of-squares reconstruction from the fully sampled 256×256 data is shown in Fig. 3a, and the result from using the proposed method is shown in Fig. 3b. In this experiment, it took 718 sec to reconstruct the image using the proposed algorithm, whereas it took 1161 sec without the proposed algorithm. The speedup factor is about 1.6. Note that the image reconstruction quality with the proposed algorithm is close to the one from the fully sampled data.

DISCUSSION

We described an innovative reconstruction procedure for CS imaging with MR array receiver systems. The proposed method reconstructs images using the under-sampled data from each channel individually and combines them via the sum-of-squares method. Multicore CPUs were used to reconstruct CS images simultaneously, which significantly shortened the reconstruction time. In our experiments with simulated four-channel data, using two cores of CPU gave maximum efficiency improvement per core, whereas using four cores gave the fastest reconstruction. However, the efficiency was not doubled as we changed the number of cores from two to four. This is because the memory bandwidth and the memory size are fixed and all cores share the same memory.

The proposed algorithm was tested for single slice two-dimensional imaging in this work. However, it can be directly applied to multislice two-dimensional imaging where parallelization can be used on multiple slices instead of multiple channels. For more computationally demanding cases such as three-dimensional imaging or multichannel multislice imaging, the efficiency gain from the parallelization will be even more beneficial. In CS image reconstruction studied in this article, the overhead is only a small portion of the total reconstruction time. Significant computation time reductions were achieved using a quad-core CPU, especially for higher computation complexity of the reconstruction algorithms using wavelet transforms. The computational time can be further shortened significantly by implementing the algorithms in C/C^{++}. Such improvement will be

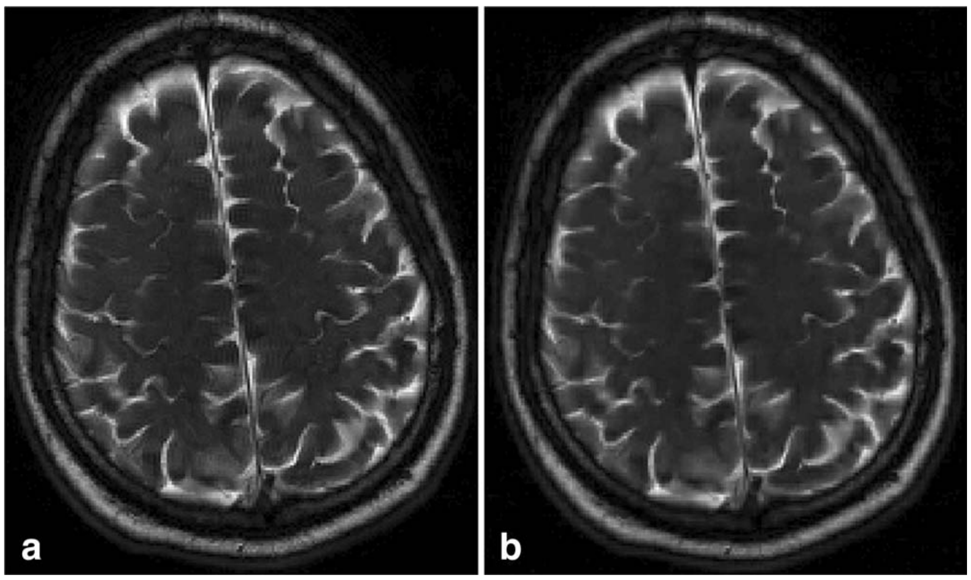

FIG. 3. Images reconstructed from the eight-channel in vivo data, using **(a)** sum of squares from fully sampled data and **(b)** the proposed method from 33% of the total data. The time to reconstruct **(b)** is about 718 sec, whereas it takes about 1161 sec without the proposed method.

complementary to the gains achieved by parallelization. In addition, other existing methods using GPU architecture, open multiprocesses, and more advanced synergetic integration of multicore CPUs with GPUs can be potentially used to further accelerate the CS algorithms for two-dimensional or three-dimensional multichannel data (14–17). These possibilities will be further explored in the future research to fully realize the potential of parallel computations for processing large, multichannel data in MRI.

REFERENCES

1. Candès EJ, Romberg J, Tao T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans Inf Theory 2006;52:489–509.
2. Candès EJ, Romberg JK, Tao T. Stable signal recovery from incomplete and inaccurate measurements. Commun Pure Appl Math 2006; 59:1207–1223.
3. Donoho D. Compressed sensing. IEEE Trans Inf Theory 2006;52: 1289–1306.
4. Lustig M, Donoho D, Pauly JM. Sparse MRI: the application of compressed sensing for rapid MR imaging. Magn Reson Med 2007;58: 1182–1195.
5. Roemer P, Edelstein W, Hayes C, Souza S, Mueller O. The NMR phased array. Magn Reson Med 1990;16:192–225.
6. Wright S, Wald L. Theory and application of array coils in MR spectroscopy. NMR Biomed 1997;10:394–410.
7. King K. Combining compressed sensing and parallel imaging. In: Proceedings of the 16th Annual Meeting of ISMRM, Toronto, 2008. p 1488.
8. Wu B, Millane RP, Watts R, Bones P. Applying compressed sensing in parallel MRI. In: Proceedings of the 16th Annual Meeting of ISMRM, Toronto, 2008. p 1480.
9. Liu B, Sebert FM, Zou Y, Ying L. Sparse SENSE: randomly-sampled parallel imaging using compressed sensing. In: Proceedings of the 16th Annual Meeting of ISMRM, Toronto, 2008. p 3154.
10. Zhao C, Lang T, Ji J. Compressed sensing parallel imaging. In: Proceedings of the 16th Annual Meeting of ISMRM, Toronto, 2008. p 1478.
11. Marinelli L, Hardy CJ, Blezek DJ. MRI with accelerated multi-coil compressed sensing. In: Proceedings of the 16th Annual Meeting of ISMRM, Toronto, 2008. p 1484.
12. Liang D, Liu B, Ying L. Accelerating sensitivity encoding using compressed sensing. In: Proceedings of the IEEE Engineering Medicine Biology Society, Vancouver, 2008. p 1667–1670.
13. Liang D, Liu B, Wang J, Ying L. Accelerating SENSE using compressed sensing. Magn Reson Med 2009;62:1574–1584.
14. Wiggers W, Bakker V, Kokkeler A, Smit G. Implementing the conjugate gradient algorithm on multi-core systems. In: Proceedings of the IEEE International Symposium on System-on-Chip, Tampere, Finland, 2007. p 1–4.
15. Borghi A, Darbon J, Peyronnet S, Chan T, Osher S. A simple compressive sensing algorithm for parallel many-core architectures. CAM Rep. 2008:08–64.
16. Stone S, Haldar J, Tsao S, Hwu W, Sutton B, Liang Z. Accelerating advanced MRI reconstructions on GPUs. J Parallel Distrib Comput 2008;68:1307–1318.
17. Knoll F, Unger M, Ebner F, Stollberger R. Real time elimination of undersampling artifacts using 3D total variation on graphics hardware. In: Proceedings of the 17th Annual Meeting of ISMRM, Hawaii, 2009. p 2838.
18. Chang C, Ji J. Compressed sensing MRI with multi-channel data using multi-core processors. In: Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Minneapolis, 2009. p 2684–2687.
19. Lustig M. SparseMRI V0.2, Available at: http://www.stanford.edu/~mlustig/SparseMRI. html. Accessed February 1, 2010.
20. Kepner JV. Parallel MATLAB for multicore and multinode computers. Philadelphia: Society for Industrial and Applied Mathematics; 2009, 273 p.