K1. High Performance Compressed Sensing MRI Image Reconstruction

Ahmed Abdel Salam, Fadwa Fawzy, Norhan Shaker, Yasser M.Kadah
Biomedical Engineering, Cairo University, Cairo, Egypt, ymk@k-space.org
Computer and Electronics, Cairo Higher institute for engineering, Cairo, Egypt, a.asalam@chi.edu.eg

ABSTRACT

Magnetic resonance imaging (MRI) has been used extensively for clinical purposes to depict anatomy because of its non-invasive nature to human body. It is always desirable to enhance the resolution of MR images in order to confirm the presence of any suspicious behavior inside the body while keeping the imaging time short. At present, MR imaging is often limited by high noise levels, significant imaging artifacts and/or long data acquisition (scan) times. Advanced image reconstruction algorithms can mitigate these limitations and improve image quality. In this paper we aim to enhance image quality and shorten imaging time using Compressed sensing (CS) and parallel computing techniques.

Keywords: Compressed Sensing(CS), Image reconstruction, Parallel Computing

I. INTRODUCTION

One of the main drawbacks of MRI scans is the time it takes to acquire the scan. During this time the patient must remain completely still, which often requires general anaesthesia. Compressed sensing allows the MRI scan to sample much less densely, which lowers sensing time[1]. Compressed sensing method replaces the conventional sampling and reconstruction operations with a more general linear measurement scheme coupled with an optimization in order to acquire certain kinds of signals at a rate significantly below Nyquist. And it goals to reduce the number of measurements required to completely describe a signal by exploiting its compressibility. This at the cost of much higher image reconstruction times.

Image reconstruction is a complicated process in which a large number of calculations is needed to retrieve information that has been lost or obscured in the imaging process itself. Sequential techniques for image reconstruction programs take long time to end the process. So we aim at finding bottle nicks which consume time and optimize it using OpenMP and Intel Parallel Studio to speed up the reconstruction process.

In this paper, we proposed an accelerated reconstruction procedure for combining CS with parallel computing techniques using OpenMP and 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.

II. MATERIAL AND METHODS

A. Intel Parallel Advisor - Survey Tool -

Intel® Parallel Studio is a comprehensive tool suite that provides C/C++ developers using Microsoft Visual Studio with tools for serial and threaded application design. Offering innovative features and capabilities, the suite equips developers to design, build and debug, verify, and tune their applications. first step in parallel process is to find bottle necks within our code ,Survey tool which is one of intel parallel advisor tools enables us to identify those hot spots as shown in table 1

Table 1 Survey Tool output attributes

Total time %	Self time	Total time
shows the top-down running total for the program, starting at 100%.	shows the time used by that function or loop.	Show the total time consumed in this function

B. Compressed sensing

The second step in parallel process is to redesign those hot spots to be in a parallelizable form can benefit from OpenMP work sharing techniques. This step requires deep understanding of CS algorithm.
Based on the CS theory, an image x can be reconstructed from a reduced set of incomplete k-space data y, where [1]

$$y = \phi x \qquad (1)$$

with ϕ being the randomly sampled Fourier transform operator.
Given x of length N, only M measurements (M<N) is required to fully recover x when x is K-sparse (K<M<N).

However, three conditions named CS1-3 are to be satisfied for the this statement to be true[1]

1) **Sparsity** which means that the desired signal has a sparse representation in a known transform domain. while the sparsity of the image is the percentage of transform coefficients sufficient for diagnostic-quality reconstruction. In our case we found that brain images have a good sparse representation in the wavelet domain at reconstructions involving 5-10% of the coefficients.

2) **Incoherence** which means that undersampled sampling space must generate noise-like aliasing in that transform domain. There are numerous ways to design incoherent sampling trajectories. We focus on Cartesian sampling because it is by far the most widely used in practice. It is simple and also highly robust to numerous sources of imperfection. Non-uniform undersampling of phase encodes in Cartesian imaging has been proposed in the past as an acceleration method because it produces incoherent artifacts[1] and that is exactly what we are looking for. Undersampling phase-encode lines offers pure randomness in the phase-encode dimensions, and a scan time reduction that is exactly proportional to the undersampling. Finally, implementation of such an undersampling scheme is simple and requires only minor modifications to existing pulse sequences.

3) **Non-linear Reconstruction,** We now describe in more detail the processes of nonlinear image reconstruction appropriate to the CS setting. let Ψ denote the linear operator that transforms from pixel representation into a sparse representation, The reconstruction is obtained by solving the following constrained optimization problem[1]

$$\text{minimize } \|\Psi x\|_1 \qquad (2)$$
$$\text{s.t. } \|\phi x - y\|_2 < \varepsilon$$

Here y is the measured k-space data from the scanner and ε controls the fidelity of the reconstruction to the measured data. The threshold parameter ε is usually set below the expected noise level.
The objective function in Eq. 2 is the ℓ1 norm, which is defined as [1]

$$\|x\|_1 = \Sigma |x_i| \qquad (3)$$

Minimizing $\|\Psi x\|_1$ promotes sparsity [2]. The constraint $\|\phi x - y\|_2 < \varepsilon$ enforces data consistency.
Eq. 2 finds a solution which is compressible by the transform . When finite-differences is used as a sparsifying transform, the objective in Eq. 2 is often referred to as Total-Variation (TV)[3], since it is the sum of the absolute variations in the image. The objective then is usually written as TV (x). Even when using other sparsifying transforms in the objective, it is often useful to include a TV penalty as well (4). This can be considered as requiring the image to be sparse by both the specific transform and finite-differences at the same time. In this case Eq. 2 [1] is written as

$$\text{minimize } \| \Psi x \|1 + \alpha \, TV(x) \quad (4)$$
$$\text{s.t. } \| \phi x - y \|2 < \varepsilon$$

where α trades Ψ sparsity with finite-differences sparsity. Special purpose methods for solving Eq. 4 have been a focus of research interest since CS was first introduced. we are interested in iteratively reweighted least squares[5],[6] because iterative methods can benefit from our parallel approach . The algorithm was tested on a platform that contained a standard Intel Corei7-2630QM CPU @ 2.00GHz, with 6-GB DDR2 memory.

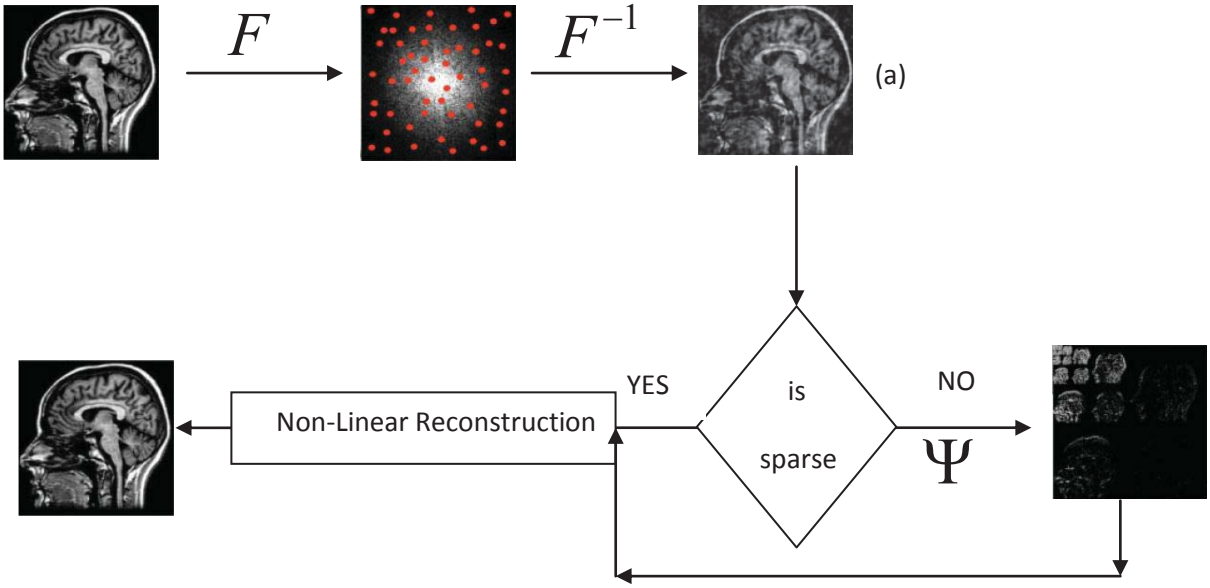

figure 1, CS flow chart

figure 1 show the flow chart of our implementation , we begin with k-space data and apply inverse fourier to obtain image (a) , then we check the sparsity using L1-norm such that minimum L1-norm corresponding to highest sparsity. if the image sparse we go directly to non linear image reconstruction technique , if not we need to transform the image to a different domain in which it has sparse representation. it's all about finding the right Ψ like STFT[8], Gabor[9], Wavelet Transform[10].

B. OpenMP

OpenMP is a shared-memory application programming interface (API) whose features are based on prior efforts to facilitate shared-memory parallel programming[11]. OpenMP is intended to be suitable for implementation on a broad range of SMP architectures. OpenMP comprised of three primary API components: Compiler Directives, Runtime Library Routines and Environment Variables. Multithreaded programs can be written in various ways, some of which permit complex interactions between threads. OpenMP attempts to provide ease of programming and to help us avoid a number of potential programming errors, by offering a structured approach to multithreaded programming. It supports the so-called fork-join programming model as shown in figure(2)

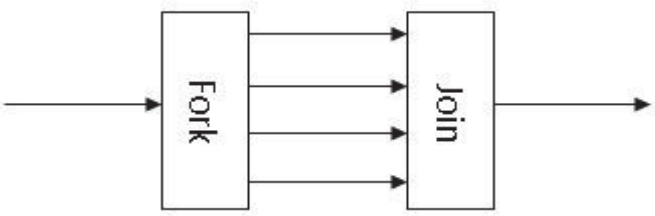

figure 2, fork-join model of OpenMP

Methods

1) We take advantage of modern parallel architectures in designing our parallel version of CS algorithm. The main data structures in CS is 2D matrices. Due to element wise operations , we can advantage of the element-level parallelism. we can divide data element among threads without concern of data dependency. fortunately most loops in CS do not contain data-dependency, which helps maintain high vector efficiency.

2) We have composite loops, in order to keep data dependency and better memory access we choose to parallel the inner-most loop, it guarantees sequential memory access unlike the outer-most loop which requires access non sequential memory location

3) OpenMP has three schedule to handle loops: static, dynamic and guided. static schedule provides good load balancing. dynamic schedule may provide better load balancing, but it costs high thread communication overhead.

4) It is not trivial to parallelize FFT due to its non sequential memory access patterns and bit-reversal element shuffling. However, this has been studied for decades and optimization techniques are well known as shown[7],details of which are out of scope of this paper.

5) We have bottlenecks which composed of three arithmetic operation for two input matrices, A and B, and two output matrices, C and D. first we try the trivial solution of computing each stage before proceeding to the next stage. but we found it inefficient due to high memory traffic as each stage needs to sweep through the entire matrix

6) Back to the element level, we can read in one element from A and B and perform the three arithmetic operation while the data kept in the register then store the result in C and D. this optimization is called loop fusion

III. Results

Table 2 lists the total reconstruction time for all experiments including time cost for file read and output file write using two cores leads to significant reduction in the computation time per core, with the reduction factor of about 1.2. In addition, using eight cores gives the shortest reconstruction time, where the average reduction factor is about 2.1.

Table 2 total reconstruction time across different hardware

Size	Cores			
512 * 512	***1***	***2***	***4***	***8***
	1010	840	591	480

figure 2a shows the reconstructed image from serial compressed sensing[12] while figure 3b shows the reconstructed image from parallel algorithm. Note that the image reconstruction quality with the proposed algorithm is close to the one from the serial algorithm.

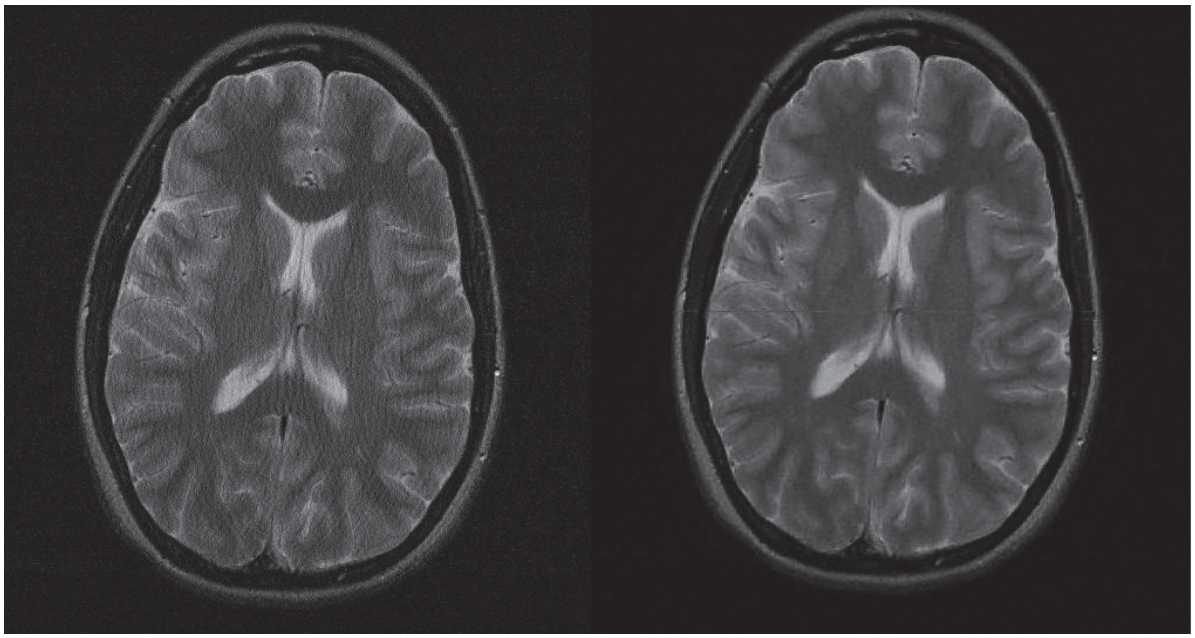

Figure 3 (a) reconstructed image form serial algorithm , (b) reconstructed image form parallel algorithm

IV. CONCLUSION

In this work, we developed a new implementation of the CS algorithm based on a parallel computing architecture. The new implementation offers less data dependency and more convenient structure to parallel computing approach. The new method was implemented to reconstruct real MRI images and results indicate significant reduction in time with reasonable image quality. This shows the potential of the new method to be used in medical image reconstruction algorithms with high complexity to speed up the reconstruction and utilize the already available processing power in newer personal computers.

REFERENCES

[1] M. Lustig, D. L. Donoho, J. M. Santos and J. M. Pauly. "Compressed Sensing MRI," *Signal Processing Magazine, IEEE,* vol. 25, pp. 72-82,March 2008;

[2] S. Chen, D. Donoho, M. Saunders "Atomic decomposition by basis pursuit," *Sci Comp*,vol. 20, pp. 33–61, 1999

[3] L. Rudin, S. Osher, E. Fatemi, " Non-linear total variation noise removal algorithm, " *Phys*. D 1992; 60:259–268.

[4] Y. Tsaig, D. Donoho, " Extensions of compressed sensing," *Signal Processing*, vol. 86, pp. 533–548, 2006

[5] Ye JC, Tak S, Han Y, Park HW. "Projection reconstruction MR imaging using FOCUSS," *Magnetic Resonance in Medicine,* vol. 57,pp. 764–775, 2007

[6] D. Donoho, M. Elad, V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise,". *IEEE Trans. Info. Thry*, vol. 52,pp.6–18, 2006

[7] A. C. Chow, G. C. Fossum, and D. A. Brokenshire, "A programming example: large FFT on the cell broadband.engine,",[Online].Available:http://www.ibm.com/ru/ondemand/innovation/special/nonflash/pdf/2053_IBM_CellIntro.pdf

[8] Tomazic, S.; Znidar, S, "A fast recursive STFT algorithm," *Electrotechnical Conference*, vol. 2, pp. 1025 - 1028, 1996.

[9] Q . Sigang, H. G. Feichtinger " Gabor-type matrices and discrete huge Gabor transforms*"* *Acoustics, Speech, and Signal Processing,"* vol. 2,pp. 1089-1092, May 1995

[10] G. Xinyi, L. Gengyin; M. Zhou, " Wavelet transform based approach to harmonic analysis," *Electrical Power Quality and Utilisation (EPQU,* pp. 1-6, Jan 2012

[11] B. Chapman, G. Jost, R. Van De Pas. *Using OpenMP Portable Shared Memory Parallel Programming,*

[12] [Online] available: http://www.stanford.edu/~mlustig/SparseMRI.html