Accelerating Cardiac MRI Compressed Sensing Image Reconstruction using Graphics Processing Units

A Thesis Presented
by

Majid Sabbagh

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements
for the degree of

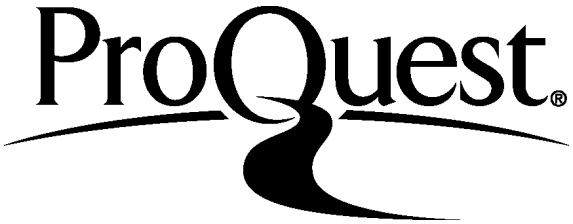
Master of Science

in

Electrical and Computer Engineering

Northeastern University
Boston, Massachusetts

April 2016



To my family.

Contents

List of Figures iv

List of Tables vi

List of Acronyms vii

Acknowledgments viii

Abstract ix

1 Introduction 1
1.1 Thesis Organization . . . 2

2 Background 3
2.1 Magnetic Resonance Imaging (MRI) Scanners and Imaging Time . . . 3
2.2 Parallel Magnetic Resonance Imaging (pMRI) . . . 4
2.3 Compressed Sensing (CS) Imaging and Reconstruction . . . 5
2.4 ℓ1-ESPIRiT CS reconstruction algorithm . . . 6
2.5 Related Work . . . 7

3 Materials and Methods 9
3.1 Data Acquisition . . . 9
3.2 Data Undersampling . . . 9
3.3 Patient Study . . . 10
3.4 ℓ1-ESPIRiT Implementation . . . 11
3.5 Performance and Quality Evaluation Approach . . . 16

4 Results and Discussion 17
4.1 Imaging and Reconstruction Time . . . 17
4.2 Quality Comparison . . . 24

5 Conclusion and Future Work 26
5.1 Conclusion . . . 26
5.2 Limitations and Future Work . . . 26

Bibliography **28**

List of Figures

3.1 (a) Pattern of fully sampled dataset compared with (b) pattern of variable density Poisson disc undersampled data to achieve about 6 fold reduction in 3D-SSFP acquisition time. 10
3.2 The main stages of the ℓ1-ESPIRiT CS reconstruction algorithm are FFTMOD, CC, ECALIB, and PICS. In the FFTMOD stage, the k-space is modulated along specified directions. In the CC stage, the data from up to 25 parallel receiver coils is compressed. In the ECALIB stage, the coils sensitivities are estimated. In the PICS stage, the unmeasured data estimation for image reconstruction is performed. . . . 12
3.3 On average the PICS stage contributed to 87% of the CPU reconstruction time in five patient datasets. The execution time of the PICS stage was 11.37 ± 3.66 minutes. 15

4.1 Execution times of the 4 main stages in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU without OpenMP parallelization for 3D datasets in 5 patients (ordered according to increasing data size). 18
4.2 Execution times of the 4 main stages in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP parallelization for 3D datasets in 5 patients (ordered according to increasing data size). 19
4.3 Execution times of the 4 main stages in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP + k20m GPU for 3D datasets in 5 patients (ordered according to increasing data size). 19
4.4 Execution times of the 4 main stages in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP + k40m GPU for 3D datasets in 5 patients (ordered according to increasing data size). 20
4.5 Execution times of the PICS stage in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP + k20m GPU and on the CPU with OpenMP + k40m GPU for 3D datasets in 5 patients. 21
4.6 The execution times for the main stages of the ℓ1-ESPIRiT CS reconstruction algorithm on CPU, CPU with OpenMP, CPU with OpenMP plus k20m GPU and CPU with OpenMP plus k40m GPU. In all implementations, the PICS stage is the major contributor to the execution time for all datasets. 22
4.7 Speed up factor vs. dataset size for CPU with OpenMP + k40m GPU implementation over CPU with and without OpenMP implementations. 24

4.8 An example of 3D-SSFP images in sagittal view acquired from a patient with congenital heart disease: (a) raw image before CS reconstruction, (b) ℓ1-ESPIRiT CS reconstructed image on CPU, (c) ℓ1-ESPIRiT CS reconstructed image on GPU, and (d) the absolute difference between the reconstructed images on CPU and GPU. The mean absolute difference between the CPU and GPU reconstructed images was 1.37e-05 with the maximum difference of 6.49e-05. 25

List of Tables

3.1 The ages of the patients under study and the dimensions of the acquired datasets in 3 orthogonal directions. 11
3.2 The parameters for different stages of the $\ell 1$-ESPIRiT algorithm and their corresponding values in our experiments. 14

4.1 Execution times of the PICS stage on CPU and GPU as well as total execution times of $\ell 1$-ESPIRiT CS algorithm on CPU, CPU with OpenMP parallelization, and CPU with OpenMP parallelization plus GPU for PICS for 3D datasets in 5 patients. Note that the dataset dimensions reported here are after pre-processing to reduce the dataset sizes. 23

List of Acronyms

CPU Central Processing Unit

GPU Graphics Processing Unit

MIC Many Integrated Core

FPGA Field Programmable Gate Array

MRI Magnetic Resonance Imaging

RF Radio Frequency

CS Compressed Sensing

FT Fourier Transform

FFT Fast Fourier Transform

DWT Discrete Wavelet Transform

3D-SSFP Three-Dimensional Steady-State Free Precession

DC Direct Current

FFTMOD FFT Modulation

CC Coil Compression

ECALIB ESPIRiT Calibration

PICS Parallel Image Compressed Sensing

BART Berkeley Advanced Reconstruction Toolbox

GCC GNU Compiler Collection

NVCC NVIDIA CUDA Compiler

Acknowledgments

I would like to express my gratitude to my advisor Professor Miriam Leeser and to my co-advisor Professor Mehdi H. Moghari for their support, advice, constructive criticism, and encouragement during this research. I would also like to thank Professor Andrew J. Powell for his support in this research.

I am grateful to God for all the helps and the energy, good health, and wellbeing that were necessary to complete this thesis. Finally, I want to thank my family and friends who helped me throughout my years of study.

Abstract

Accelerating Cardiac MRI Compressed Sensing Image Reconstruction using Graphics Processing Units

by

Majid Sabbagh

Master of Science in Electrical and Computer Engineering

Northeastern University, April 2016

Prof. Miriam Leeser, Advisor

Prof. Mehdi H. Moghari, Co-Advisor

Cardiac magnetic resonance imaging (MRI) has become a crucial part of monitoring patients with congenital heart diseases. An important limitation of cardiac MRI using the prominent 3D steady-state free precession (3D-SSFP) sequence is its long scan time. Compressed sensing (CS) algorithm reduces the scan time by undersampling the data but increases the image reconstruction time because a non-linear optimization problem must be iteratively solved to estimate the missing data and reconstruct the images. The growing demand for reducing the examination time in cardiac MRI led us to investigate opportunities to accelerate this non-linear optimization problem to facilitate the migration of CS into the clinical environment. Using undersampled 3D-SSFP datasets acquired from five patients, we compared the speed and output quality of CS image reconstruction algorithm using a Central Processing Unit (CPU), a CPU with OpenMP parallelization, and two different Graphics Processing Unit (GPU) platforms. Reconstruction time had a mean of 13.1 minutes with a standard deviation of 3.8 minutes for the CPU, a mean of 11.5 minutes with a standard deviation of 3.6 minutes for the CPU with OpenMP parallelization, a mean of 2.2 minutes with a standard deviation of 0.3 minutes for the CPU with OpenMP plus NVIDIA k20m GPU, and a mean of 1.7 minutes with a standard deviation of 0.3 minutes for the CPU with OpenMP plus NVIDIA k40m GPU. The quality of images reconstructed on GPU and on CPU, as assessed by image subtraction, was comparable. Furthermore, necessary steps for implementation of rapid CS image reconstruction in the clinical environment are discussed.

Chapter 1

Introduction

Magnetic resonance imaging (MRI) is one of the main imaging modalities to generate images of the anatomy and physiological function of the body. Specifically, cardiac MRI has become an essential part of procedural planning and monitoring of children and adults with congenital heart disease [1][2].

An important cardiac MR sequence for acquiring high-resolution anatomic datasets of the entire thorax is the electrocardiogram and respiratory-gated three-dimensional steady-state free precession (3D-SSFP) sequence [3][4]. However, a notable limitation of this sequence is its long imaging time. Imaging by this method typically takes about 10-15 minutes for 1.2 mm^3 isotropic resolution [5]. During such a long scan time, the patient's heart rate, breathing pattern, and position may change which could lead to a non-diagnostic image quality. To avoid image distortions due to these unwanted movements, clinicians may use sedation for patients which could cause serious neurological damage [6]. Also, sedating patients imposes additional costs on MRI. To reduce the imaging time and minimize the negative effects of variations in heart rate, breathing pattern, and gross motion of patients on image quality; compressed sensing (CS) may be used as a promising acceleration technique [7].

Applying the CS technique to reduce the imaging time increases the post processing time, because the unmeasured data should be estimated using CS image reconstruction. CS image reconstruction is based on solving an iterative, computationally intensive, non-linear optimization problem to estimate the whole image from undersampled data. Thus CS reduces the scan time with the cost of increased image reconstruction time. To use CS imaging in clinical settings we need to accelerate the image reconstruction to make the combined CS-based imaging and reconstruction time lower than the non CS imaging methods.

In this thesis, parallel processing is used to accelerate CS reconstruction for 3D cardiac MR imaging datasets acquired from patients. Specifically, we used OpenMP programming for parallel processing on a Central Processing Unit (CPU). In addition, after determining that the iterative estimation step is the major contributor to reconstruction time, an NVIDIA Graphics Processing Unit (GPU) platform [8] is used to accelerate this step using CUDA with the goal of reducing the total reconstruction time. We used Berkeley Advanced Reconstruction Toolbox (BART) [9] to implement the ℓ1-ESPIRiT CS reconstruction algorithm [10] on GPUs. This thesis makes the following contributions:

- Integrate the ℓ1-ESPIRiT algorithm using tools from BART, chaining and configuring the proper blocks to create a functional CS reconstruction
- Adapt the acquired patient datasets to the BART framework for GPU implementation using lossless compression
- Apply the BART framework to patient datasets of different sizes for CS MRI
- Study different implementation methods for CS reconstruction using parallelism (OpenMP and CUDA)
- Extend the OpenMP parallelism in BART
- Evaluate execution time and quality of reconstructed images of 5 patients
- Identify potential steps for further acceleration in reconstruction time
- Facilitate the use of CS MRI in clinical settings

1.1 Thesis Organization

This thesis is organized as follows. In Chapter 2, we present background on MRI and compressed sensing image reconstruction algorithm as well as related work. Chapter 3 introduces the experimental setup, patient study, and the ℓ1-ESPIRiT CS reconstruction algorithm. Chapter 4 describes the experiments we performed and the comparison of execution times and reconstructed image quality for different implementations. We conclude and discuss the limitations of our study and future work in Chapter 5.

Chapter 2

Background

In this chapter, the MRI scanner structure and the MR image formation process are described. Also, the compressed sensing method and different categories of reconstruction algorithms are explained. Then, the recent related work is discussed.

2.1 Magnetic Resonance Imaging (MRI) Scanners and Imaging Time

MRI scanners use strong magnetic fields to align the nuclear spins in the hydrogen atoms of the human body's water molecules. Following that, radio transmitters are used in the scanners to excite those nuclear spins for releasing radio frequency (RF) energy before going back to equilibrium. Gradient coils are used to create gradients in the main magnetic field for mapping different scan locations to different frequencies. The RF energy that the nuclear spins release are sensed using the receive RF coils. Then the sensed signals are sampled in the frequency domain to extract the contribution of each spin to the frequency spectrum. In this process, we can either acquire all of the samples from the MRI scanner to form the full image without aliasing, eliminating the need for image reconstruction, or randomly acquire a portion of the samples using undersampling techniques, such as compressed sensing (CS), to form a partial and then compensate the aliasing introduced by undersampling using an image reconstruction algorithm. The image reconstruction algorithm reconstructs the full image with the cost of more post-processing [11].

An important method to acquire cardiac MR images is steady state free precession (SSFP) sequence. SSFP is a variant of gradient echo imaging that produces bright blood images with great contrast between myocardium and blood within the heart [12]. The 3D-SSFP is a very commonly used acquisition method in 3D cardiac MRI because of its high signal to noise ratio. The acquisition

time for a 3D-SSFP dataset can be reduced from about 15 minutes to about 3.5 minutes using the CS scanning method, however the corresponding CS image reconstruction can take up to 18 minutes on a CPU making the overall CS-based imaging and reconstruction time more than 20 minutes. To use CS imaging in clinical settings we need to accelerate the image reconstruction to make the overall CS-based imaging and reconstruction time lower than one minutes.

2.2 Parallel Magnetic Resonance Imaging (pMRI)

Parallel magnetic resonance imaging (pMRI) takes advantage of spatial sensitivity information inherent in an array of multiple receiver surface coils to partially replace time-consuming spatial encoding, which is normally performed by switching magnetic field gradients. In this way, only a fraction of phase-encoding steps have to be acquired, directly resulting in an accelerated image acquisition while maintaining full spatial resolution and image contrast. Besides increased temporal resolution at a given spatial resolution, the time savings due to pMRI can also be used to improve the spatial resolution in a given imaging time. Furthermore, pMRI can diminish susceptibility-caused artifacts by reducing the echo train length of single- and multi-shot pulse sequences. There are different pMRI methods which can be categorized as follows:

- The pMRI methods based on regular undersampling patterns and linear reconstructions: In these methods, the samples are acquired using a non-random pattern from the MRI scanner. This can be done by acquiring equidistantly k-space [13] lines and then reconstructing the image using a linear algorithm. The reconstruction can either take place in image space, for example in SENSE [14] or PILS [15], which consists of an unfolding or inverse procedure, or in k-space, for example in SMASH [16] or GRAPPA [17], which consist of a calculation of missing k-space data. Hybrid techniques, such as SPACE RIP [18], are also available which combine the procedures from different regular imaging techniques. However, if we study these reconstruction algorithms from a linear system perspective, many common blocks and features can be found among them which can help one to combine or optimize different algorithms to create a faster or more accurate reconstruction algorithm for a particular application [19]. Also, it facilitates the decomposition of algorithms into building blocks and development of a common implementation framework.

- The pMRI methods based on random undersampling patterns and nonlinear reconstructions: In these methods, the samples are acquired using a random pattern from the MRI scanner.

This can be done by random undersampling methods such as variable density Poisson disc sampling [20] in CS MRI. The unmeasured data is then estimated by solving an iterative non-linear optimization problem. The $\ell 1$-ESPIRiT algorithm [10] is an example of the CS reconstruction methods.

The advantage of pMRI methods based on regular sampling is that, the image reconstruction is done using a linear algorithm while in the pMRI techniques with random sampling, the image reconstruction is a nonlinear procedure. The typical undersampling factor or acceleration rate in the regular sampling is about 2-4. To push the acceleration rate higher to about 6-8 to further reduce the acquisition time, pMRI techniques using CS are commonly used in MRI.

2.3 Compressed Sensing (CS) Imaging and Reconstruction

CS arose in the literature of information theory and approximation theory as a high-level mathematical idea [21][22][23]. The aim is to measure a relatively small number of signal samples using a *random* sampling method, in fact much smaller than the number of samples nominally defining the signal. However, since the underlying signal is compressible, the unmeasured signal samples could be estimated from the few acquired samples. Therefore, the signal can be reconstructed with good accuracy from relatively few measurements by a nonlinear procedure. In MRI, we look at a special case of CS where the sampled linear combinations are individual Fourier coefficients (k-space samples). CS can allow accurate reconstructions from a small subset of k-space, rather than an entire k-space grid if all of the following requirements are fulfilled [24]:

- Transform sparsity: The target image should have a sparse representation in a certain transform domain (i.e., it must have a few non-zero elements in a given domain).
- Incoherence of undersampling artifacts: The artifacts in linear reconstruction caused by k-space undersampling should be incoherent (white Gaussian noise) in the sparsifying transform domain.
- Nonlinear reconstruction: The image should be reconstructed by a nonlinear procedure that enforces both sparsity of the image representation and consistency of the reconstruction with the acquired samples.

Real-world images images with non-random structures or natural images [25] can usually be compressed with little or no perceptible loss of information. Transform-based compression is a

widely used method adopted in different image standards. This strategy first applies a sparsifying transform, mapping image content into a vector of sparse coefficients, and then encodes the sparse vector by approximating the most significant coefficients and ignoring the smaller ones. The discrete wavelet transform (DWT) is a common sparsifying transform [26]. Most MR images are sparse in an appropriate transform domain so the first condition is met for MR images. The fact that incoherence is important, means that MR acquisition can be designed to achieve incoherent undersampling. Finally, for the third requirement, there are various non-linear algorithms for CS reconstruction. We used ℓ1-ESPIRiT in our study.

To summarize, in CS reconstruction algorithm, an iterative non-linear optimization problem is solved to estimate and reconstruct images from very few randomly sampled image data. This algorithm is based on two principles: a) the samples should be randomly acquired, and b) for image reconstruction, the images should be sparse in a certain domain. This algorithm then reconstructs the images by iteratively optimizing a non-linear cost function ensuring that the output images are 1) consistent with the acquired samples and 2) sparse in a given domain after each iteration.

2.4 ℓ1-ESPIRiT CS reconstruction algorithm

As mentioned in Section 2.2, there are different types of reconstruction algorithms with various properties. These algorithms can be categorized as follows:

- Reconstruction algorithms based on explicit knowledge of the coil sensitivities. An example is SENSE [27][14].
- Reconstruction algorithms based on local kernels in k-space, which exploit the learned correlation between multiple channels in neighboring points in k-space. Examples are GRAPPA [17] and SPIRIT [28].
- Reconstruction algorithms that can combine the advantages of the first and second categories.

Algorithms based on explicit knowledge of the sensitivities allow optimal reconstruction (e.g. in terms of minimum mean square error or minimum variance unbiased estimation, when used with exact sensitivities). However, it is often hard to accurately and robustly measure the sensitivities and even small errors can result in inconsistencies that lead to visible artifacts in the image. On the other hand, algorithms based on learned correlations tend to fail for high acceleration factors, but are much more robust to errors. This later property makes them the preferred choice in clinical practice

today. The ℓ1-ESPIRiT reconstruction, which is used in this research, combines the advantages of SENSE with robustness to certain errors similar to GRAPPA [10].

2.5 Related Work

Different approaches have been proposed for accelerating CS reconstruction on various types of 1- or 2-dimensional data. Some researchers suggested using the Barzilai-Borwein and Split Bergman formulations on a single GPU or multiple GPUs. Park et al. employed Barzilai-Borwein formulation and a GPU implementation to speed up CS reconstruction up to 7-fold compared to a multi-core CPU implementation [29]. Smith et al. [30] used a Split Bergman solver in the CS reconstruction algorithm and implement it on a GPU platform to achieve up to 27-fold speed up compared to a multi-core CPU implementation. A multi-GPU implementation was proposed by Quan et al. achieving up to 7-fold speed up compared to a CPU implementation [31]. GPU implementation was also used for accelerating CS reconstruction on 2D cine MRI datasets [32]. Kulkarni et al. and a few other research groups accelerated 1D or 2D CS reconstruction process using Field Programmable Gate Arrays (FPGAs) [33][34][35]. Although these efforts were successful in speeding up the CS reconstruction, they are limited to 1D or 2D datasets and missed the general and expandable structure for 3D CS reconstruction.

Different methods have been proposed to accelerate CS reconstructions for 3D MRI and computed tomography datasets using GPU, FPGA, and Intel's Many Integrated Core (MIC) architectures. Nett et al. proposed a GPU implementation of a CS reconstruction algorithm for Computed Tomography (CT) scans [36]. Similarly, Chen et al. implemented a CS reconstruction algorithm on FPGAs for CT datasets [37]. Kim et al. investigated the development and optimization of a CS reconstruction algorithm on CPU, GPU, and Intel's MIC architectures for 3D MRI datasets [38]. Compared to CPUs, they showed about a 5-fold speed up using Intel's MIC platform. In [39] Stone et al. reported 21-fold speed up of MRI reconstruction time with a GPU implementation compared to a CPU implementation for human brain datasets. Nam et al. also showed 58-fold speed up compared to a serial C++ CPU implementation using a GPU implementation [40]. These studies, however, are not applied and analyzed on patient datasets with various dimensions. The aim of this study was to accelerate the ℓ1-ESPIRiT CS reconstruction algorithm for 3D MRI patient datasets using GPUs and evaluate the result for clinical usage. We used CPU and GPU compute nodes from the Northeastern University Discovery Cluster [41] for our experiments. For accelerating the CS reconstruction, we used OpenMP [42] which supports a shared memory multi-threading parallel programming model.

Also, we used the CUDA framework for parallel processing on NVIDIA Tesla k20m/k40m GPUs to further speed up the CS reconstruction.

Chapter 3

Materials and Methods

In this chapter the experimental setup and the methods that we used for undersampling the data, reconstructing it, and evaluating the result are described. The numerical results reported in Chapter 3 and Chapter 4 are in the format of *mean* $\pm$ *standard deviation*.

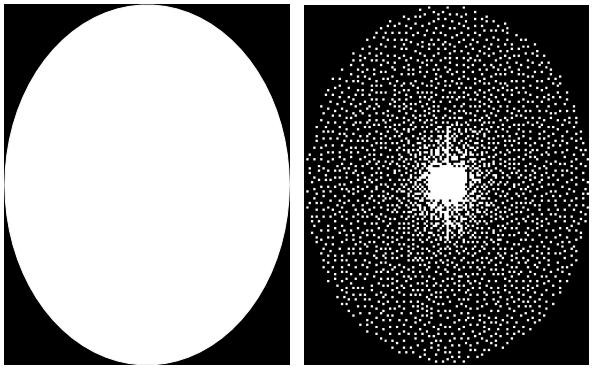
3.1 Data Acquisition

Cardiac MR imaging using 3D-SSFP sequence takes about 10-15 minutes [3][4]. This long acquisition time is due to the fact that the cardiac and respiratory motions should be compensated for acquiring a sharp image. To freeze the cardiac motion, the data is segmented and each segment is acquired at a specific cardiac phase where the heart has the least motion. This means that for acquiring the whole dataset, multiple cardiac cycles or heart beats are needed. Another motion that should be compensated is the respiratory motion. Based on the patient's breathing pattern, the acquisition is confined to the respiratory end expiration. These acquisition strategies make the 3D-SSFP sequence a time consuming acquisition method, however it provides very high signal to noise ratio (SNR). To reduce the acquisition time, one could only sample a subset of data and estimate the unmeasured data using a compressed sensing (CS) reconstruction algorithm.

3.2 Data Undersampling

We used CS technique in cardiac MR imaging to reduce the imaging time of 3D-SSFP sequence. The 3D-SSFP data was randomly undersampled using a variable density Poisson disc sampling pattern [20]. Poisson disk sampling produces points that are tightly packed but regularly distributed,

and the distance among the points cannot be smaller than a specified threshold. The reason that the variable density Poisson disk sampling method is used to undersample the k-space [13] is that, the energy distribution is not equal in different sections of k-space. In fact, the density of energy varies from central to peripheral sections. The center or direct current (DC) part of k-space carries the majority of signal energy and the peripheral sections contain a small amount of total energy. Therefore, the central 2% of k-space is fully sampled and from the peripheral sections 16% of data is randomly sampled as shown in Figure 3.1. In total 18% of the data is sampled and the imaging time is reduced by a factor of 6.



a) Fully sampled data b) Undersampled data

Figure 3.1: (a) Pattern of fully sampled dataset compared with (b) pattern of variable density Poisson disc undersampled data to achieve about 6 fold reduction in 3D-SSFP acquisition time.

3.3 Patient Study

To compare different implementations of $\ell 1$-ESPIRiT CS reconstruction, 5 patients (age 22.6 $\pm$ 11.1 years, 3 females) with congenital heart disease referred for cardiac MRI exams were recruited. For each patient, a 3D-SSFP sequence was acquired with the following parameters: field of view about 386$\times$230$\times$120 mm^3, voxel size 1.5 mm^3 reconstructed to 0.75 mm^3, flip angle 90°, echo time 2.4 ms, repetition time 4.7 ms, and bandwidth 542 Hz. The datasets acquired from patients had different dimensions because the patients had different thorax sizes. The ages of patients and dataset dimensions are shown in Table 3.1.

The Boston Children's Hospital Committee on Clinical Investigation approved this study, and written informed consent was obtained from the patients.

Table 3.1: The ages of the patients under study and the dimensions of the acquired datasets in 3 orthogonal directions.

		Dataset Dimension		
	Age (yrs)	Frequency encode (read-out)	Phase encode	Slice encode
Patient 1	14	512	128	113
Patient 2	18	512	145	96
Patient 3	18	512	141	128
Patient 4	21	512	170	121
Patient 5	42	512	157	124

3.4 ℓ1-ESPIRiT Implementation

The Berkeley Advanced Reconstruction Toolbox (BART) [9] was used for implementing the ℓ1-ESPIRiT CS reconstruction on CPU and on GPU. BART is a framework for computational MRI which has a programming library that includes common operations on multi-dimensional arrays, such as Fourier and wavelet transforms, for CPU and for GPU. It also has generic implementations of iterative optimization and reconstruction algorithms and a toolbox of command line programs which provide direct access to basic operations and efficient implementations of many CS calibration and reconstruction algorithms.

We implemented the ℓ1-ESPIRiT CS reconstruction algorithm in four main stages as shown in Figure 3.2. In the first stage, the undersampled data was modulated by a complex exponential in the frequency domain (FFTMOD). Then the data from the parallel receiver coils were projected onto eight orthogonal virtual channels in the coil compression stage (CC). For our datasets the samples are acquired from a maximum of 25 parallel receiver coils. In the next stage, the coil sensitivities were estimated using the ESPIRiT calibration method (ECALIB). Finally, in the Parallel Imaging Compressed Sensing reconstruction stage (PICS), an iterative non-linear optimization problem was solved to estimate the unmeasured data and reconstruct the complete image.

To implement ℓ1-ESPIRiT CS reconstruction, a Matlab script was written to call, configure and properly connect four different command line programs, `fftmod`, `cc`, `ecalib`, and `pics`

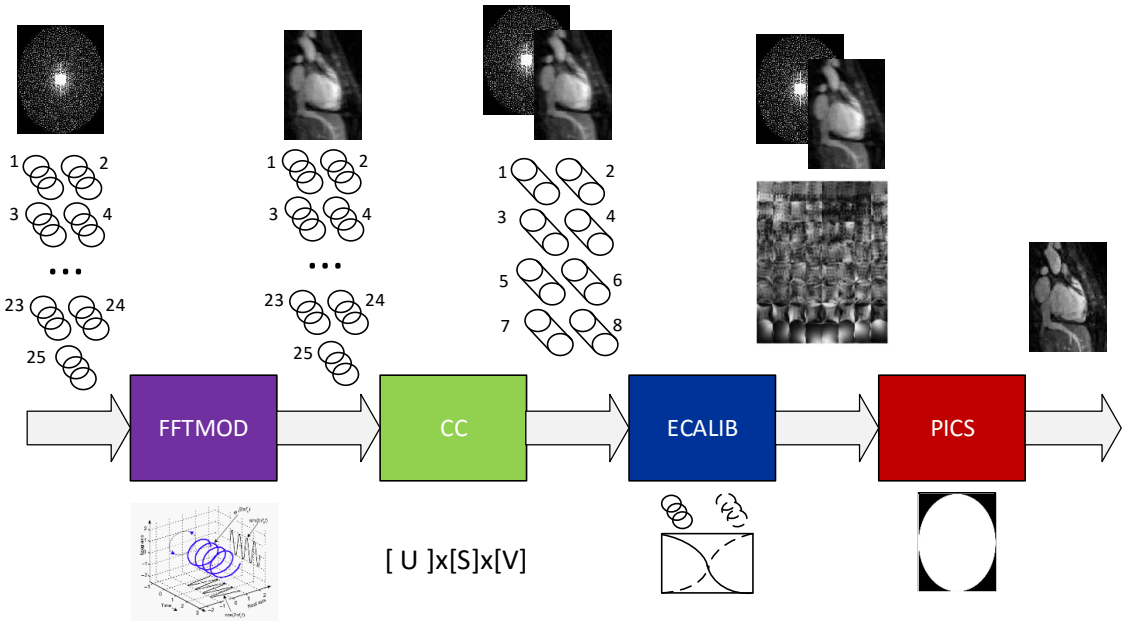


Figure 3.2: The main stages of the $\ell 1$-ESPIRiT CS reconstruction algorithm are FFTMOD, CC, ECALIB, and PICS. In the FFTMOD stage, the k-space is modulated along specified directions. In the CC stage, the data from up to 25 parallel receiver coils is compressed. In the ECALIB stage, the coils sensitivities are estimated. In the PICS stage, the unmeasured data estimation for image reconstruction is performed.

from the BART framework. The `pics` program which was the main part the ℓ1-ESPIRiT CS reconstruction, included CPU and GPU implementations of 3D fast Fourier transform (FFT) and inverse 3D FFT for converting the data from image domain to frequency domain and vice versa on CPU and on GPU. Also, CPU and GPU implementations of divergence-free wavelet transform [43] were used to sparsify the data, so that the the unmeasured samples could be estimated in the wavelet domain on CPU and on GPU. These implementations were based on the custom designed libraries from BART as well as third-party libraries, such as linear algebra package (LAPACK) and CUFFT. This script was also used for reading the datasets and extracting the raw data for processing, coarse grain profiling of the execution times, writing back the results to the appropriate files, and visualizing the results. The parameters for different stages of the ℓ1-ESPIRiT algorithm were configured as shown in Table 3.2. The values of these parameters were fixed in all of our experiments. The description of the parameters in Table 3.2 is as follows:

- FFTMOD parameter: The parameter for determining the dimensions for which we apply modulation is encoded as a three bit binary pattern. Each bit in this pattern corresponds to a certain dimension and if a bit is set to 1, modulation is applied along that dimension. Therefore, we pass `6` which is `110` in binary to the `fftmod` program to apply modulation along the second and third dimensions.

- CC parameter: The parameter for specifying the number of target virtual channels for compression. To perform compression to 8 virtual channels, we pass `-p 8` to the `cc` program.

- ECALIB parameters: The parameters for determining the number of maps and the transition type for calibration. To estimate coil sensitivities using ESPIRiT calibration to generate 1 sensitivity map with smooth transitions using Soft-SENSE algorithm [10], we pass `-S` (to specify the algorithm) and `-m1` (to specify the number of maps) to the `ecalib` program.

- PICS parameters: The parameters for determining the number of iterations in the estimation step, the type of the algorithm for regularization and data sparsifying, and the regularization threshold. To perform parallel imaging compressed sensing reconstruction with 100 iterations using l1-wavelet method and 0.01 as regulation parameter, we pass `-i 100 -l1 -r 0.01` to the `pics` program.

For executing the CS reconstructions, we used the Northeastern University Discovery Cluster compute nodes. The data was reconstructed using the the ℓ1-ESPIRiT CS algorithm on:

Table 3.2: The parameters for different stages of the ℓ1-ESPIRiT algorithm and their corresponding values in our experiments.

Stage	Parameter	Value
FFTMOD	Modulation direction	`6`
CC	Number of target virtual channels	`-p 8`
ECALIB	Transition type	`-S`
	Number of maps	`-m1`
PICS	Number of iterations	`-i 100`
	Regularization method	`-l1`
	Regularization threshold	`-r 0.01`

1. **A CPU without OpenMP parallelization**: In this implementation, no parallelization is used and all of the stages were serially executed. All the stages were implemented in C/C++ and compiled using the GNU compiler collection (GCC) [44] with `-O3 -ffast-math` optimization flags for level-3 optimization and also fast floating point mathematical operations. The CPU node had dual Intel E5-2650 eight core CPUs @ 2.00 GHz and 128 GByte of RAM.

2. **A CPU with OpenMP parallelization**: In this implementation, 32 threads was used for OpenMP parallelization in all of the stages. We extended the OpenMP parallelization to the main `for` loop of the `fftmod` program, using the `parallel for` pragma. All the stages were implemented in C/C++ and compiled with `-O3 -ffast-math -fopenmp` optimization flags using the GCC. The `fopenmp` flag was used for activating the OpenMP parallelization. The CPU node had dual Intel E5-2650 eight core CPUs @ 2.00 GHz and 128 GByte of RAM.

3. **A CPU with OpenMP + an NVIDIA k20m GPU**: In this implementation, only the PICS stage was accelerated on a k20m GPU using CUDA programming; the FFTMOD, CC, and ECALIB stages were performed on a CPU with OpenMP parallelization. The PICS code was compiled using NVIDIA CUDA Compiler (NVCC) [45] with `-arch=sm_35 -m64` flags to target an NVIDIA GPU device with compute capability of 3.5 and a 64-bit host architecture.

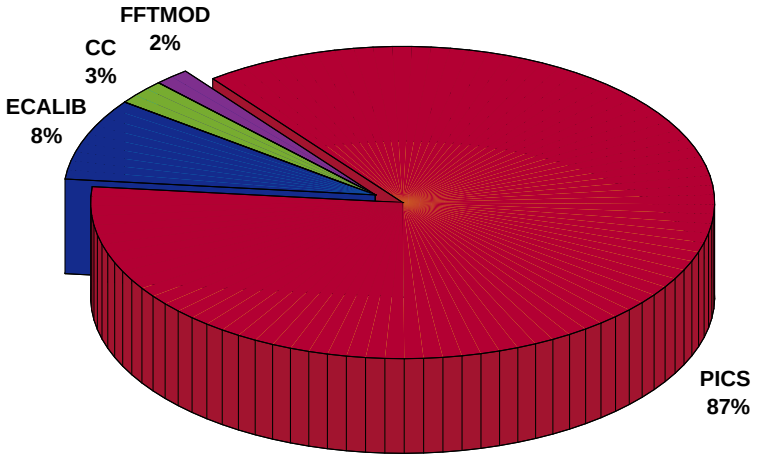


Figure 3.3: On average the PICS stage contributed to 87% of the CPU reconstruction time in five patient datasets. The execution time of the PICS stage was 11.37 $\pm$ 3.66 minutes.

All other stages were compiled similar to the CPU with OpenMP implementation. The CPU node had dual Intel E5-2650 eight core CPUs @ 2.00 GHz and 128 GByte of RAM and the GPU node had NVIDIA Tesla K20m GPU with 2496 computing cores and 4.8 GB of global memory.

4. **A CPU with OpenMP + an NVIDIA k40m GPU**: In this implementation, only the PICS stage was accelerated on a k40m GPU using CUDA programming; the FFTMOD, CC, and ECALIB stages were performed on a CPU with OpenMP parallelization. Similar to the third implementation, the PICS code was compiled using NVCC with `-arch=sm_35 -m64` flags, all other stages were compiled similar to the CPU with OpenMP implementation. The CPU node had dual Intel E5-2690 eight core CPUs @ 2.60 GHz and 128 GByte of RAM and the GPU node had NVIDIA Tesla K40m GPU with 2880 computing cores and 12 GB of global memory.

In all of the experiments, the number of iterations for the estimation part of the reconstruction algorithm is set to 100 so that the reconstruction time could be evaluated fairly for different datasets. As shown in Figure 3.3, on average the PICS stage took 87% of the reconstruction time on CPU in all datasets, so the PICS stage was accelerated using GPU.

3.5 Performance and Quality Evaluation Approach

The execution times of $\ell 1$-ESPIRiT CS reconstruction for 1) CPU, 2) CPU with OpenMP parallelization, 3) CPU with OpenMP + k20m GPU, and 4) CPU with OpenMP + k40m GPU, implementations were evaluated by coarse grain profiling of the four main stages in the algorithm (i.e., FFTMOD, CC, ECALIB, and PICS), including all of the file reads/writes, computations, CPU/GPU memory access latencies and Matlab script overhead. To ensure that the results were statistically robust, the reconstructions were repeated 10 times for each dataset in all four implementations, and the mean execution time was reported. Then, the bottlenecks in the execution times were determined through profiling. Mean absolute differences were calculated between reconstructed images from the implementations to quantitatively compare their reconstructed image quality. When image quality is similar, the mean absolute difference should be close to zero.

A two-tailed paired Student's t-test [46] was used to compare the execution times. A p-value$\leq$0.05 was considered statistically significant.

Chapter 4

Results and Discussion

In this chapter, we present our experimental results for CS reconstruction using CPU and GPU. We use *execution time in minutes* as a metric for comparing the performance in CPU and GPU implementations. Moreover, we evaluate the achieved speed up of reconstruction by accelerating the PICS stage using GPU. Finally, we use *mean absolute difference* for quantitative comparison of the quality of the reconstructed images on CPU and on GPU.

4.1 Imaging and Reconstruction Time

The acquisitions and reconstructions were successfully completed on the five patient datasets described in Chapter 3. The acquisition time for the MRI 3D-SSFP patient datasets after six fold undersampling was 3.4 ± 1.0 minutes. The acquisition time reduced from about 18 minutes to about 3.5 minutes using the CS technique.

We evaluated the execution time of the $\ell 1$-ESPIRiT CS reconstruction algorithm using four different implementations as follows:

1. **A CPU without OpenMP parallelization**: In this implementation all of the stages has been executed serially. The execution time of the PICS stage which performed the iterative part of the $\ell 1$-ESPIRiT CS reconstruction, was 11.37 ± 3.66 minutes. The execution time of ECALIB, CC, and FFTMOD stages were 1.11 ± 0.09 minutes, 0.34 ± 0.05 minutes, and 0.24 ± 0.03 minutes respectively. Figure 4.1 shows the execution times of each stage in this implementation.

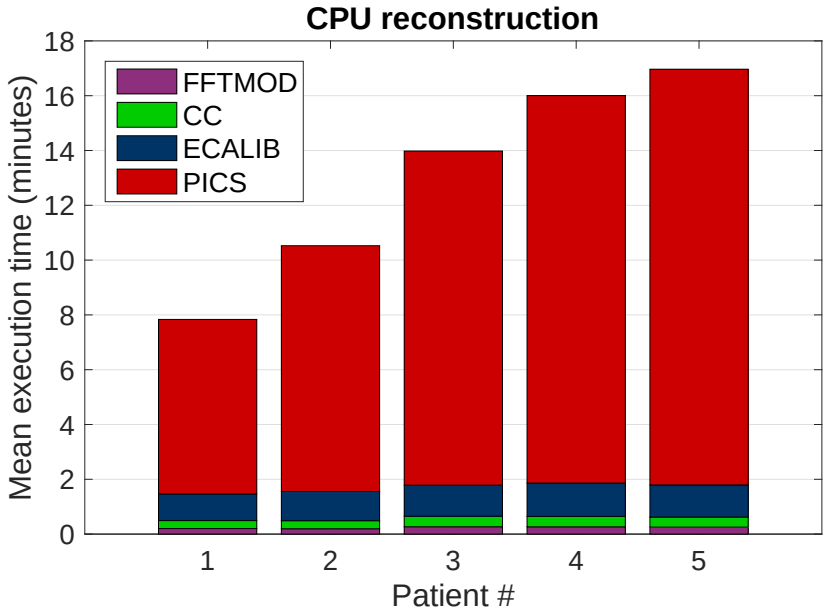


Figure 4.1: Execution times of the 4 main stages in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU without OpenMP parallelization for 3D datasets in 5 patients (ordered according to increasing data size).

2. **A CPU with OpenMP parallelization**: In this implementation, OpenMP parallelization used in all of the stages. The execution times of PICS, ECALIB, CC, and FFTMOD with OpenMP parallelization were 10.60 $\pm$ 3.52 minutes, 0.59 $\pm$ 0.04 minutes, 0.16 $\pm$ 0.03, and 0.18 $\pm$ 0.03 minutes respectively. In this implementation the OpenMP parallelization is extended in the FFTMOD stage, which on average reduced the execution time of this stage about 20%. However, the FFTMOD stage still had slightly higher execution time compared to the CC stage. This is due to the fact that OpenMP parallelization is used to a greater extent in CC than in FFTMOD. Figure 4.2 shows the execution times of each stage in this implementation.

3. **A CPU with OpenMP + an NVIDIA k20m GPU**: In this implementation, OpenMP parallization used in all of the stages. The execution times of PICS, ECALIB, CC, and FFTMOD were 1.24 $\pm$ 0.27 minutes, 0.65 $\pm$ 0.03 minutes, 0.15 $\pm$ 0.02 minutes, and 0.16 $\pm$ 0.03 minutes respectively. Figure 4.3 shows the execution times of each stage in this implementation.

4. **A CPU with OpenMP + an NVIDIA k40m GPU**: In this implementation, OpenMP parallization used in all of the stages. The execution times of PICS, ECALIB, CC, and FFTMOD were 0.97 $\pm$ 0.20 minutes, 0.44 $\pm$ 0.02 minutes, 0.12 $\pm$ 0.01 minutes, and 0.14 $\pm$ 0.02 minutes

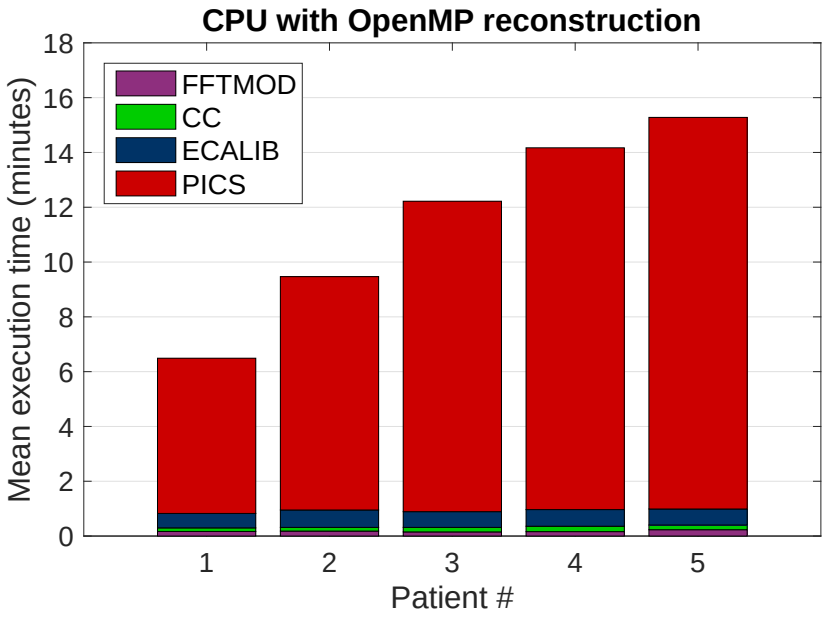


Figure 4.2: Execution times of the 4 main stages in the $\ell 1$-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP parallelization for 3D datasets in 5 patients (ordered according to increasing data size).

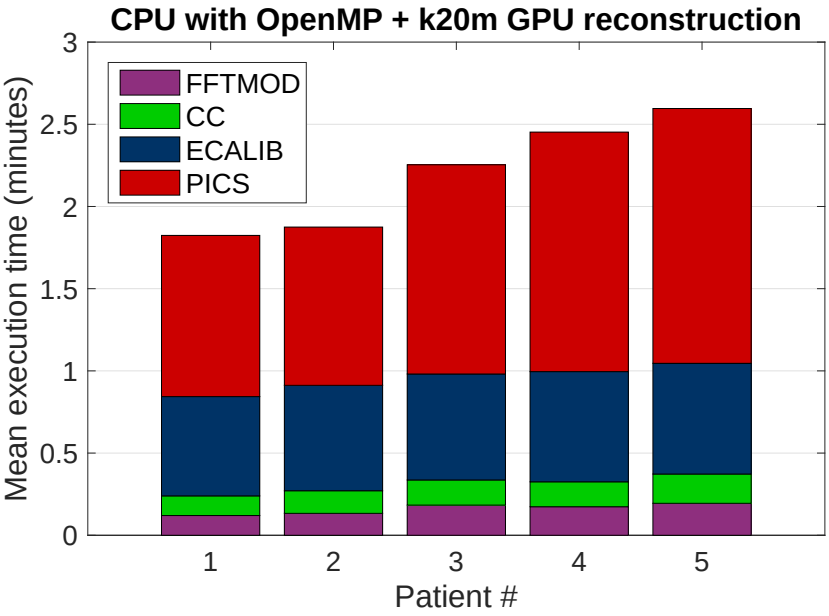


Figure 4.3: Execution times of the 4 main stages in the $\ell 1$-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP + k20m GPU for 3D datasets in 5 patients (ordered according to increasing data size).

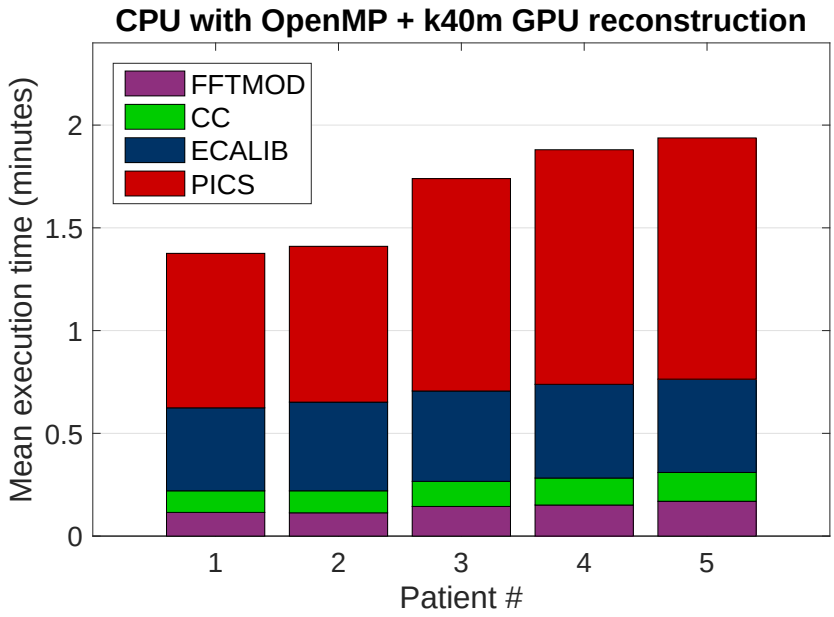


Figure 4.4: Execution times of the 4 main stages in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP + k40m GPU for 3D datasets in 5 patients (ordered according to increasing data size).

respectively. Figure 4.3 shows the execution times of each stage in this implementation.

We also compared the execution times of PICS on an NVIDIA k20m GPU and an NVIDIA on k40m GPU. Figure 4.5 shows that by using the k40m GPU to accelerate the PICS stage, the execution time of this stage was sped up by a factor of 1.3 on average in 5 datasets compared to using the k20m GPU. For CS reconstruction of the five datasets on the GPUs, a mean of 591 $\pm$ 101 MB of GPU memory was utilized.

Figure 4.6 shows the CS reconstruction times for CPU, CPU with OpenMP, and CPU with OpenMP + GPU implementations in one graph. The CPU with OpenMP decreased the execution time of ℓ1-ESPIRiT CS reconstruction algorithm on average 12% compared with CPU without OpenMP. The CPU with OpenMP plus k20m GPU and k40m GPU reduced the execution time of the CS algorithm on average 83% and 87% respectively compared to the CPU without OpenMP. Implementing only the PICS stage on a GPU sped up the CS reconstruction about 5.2$\times$ with the k20m GPU and 6.8$\times$ with the k40m GPU compared to the CPU with OpenMP. Therefore, among all implementations, the best results were achieved using the k40m GPU.

A challenge that we had in the first step of implementing the ℓ1-ESPIRiT was that BART

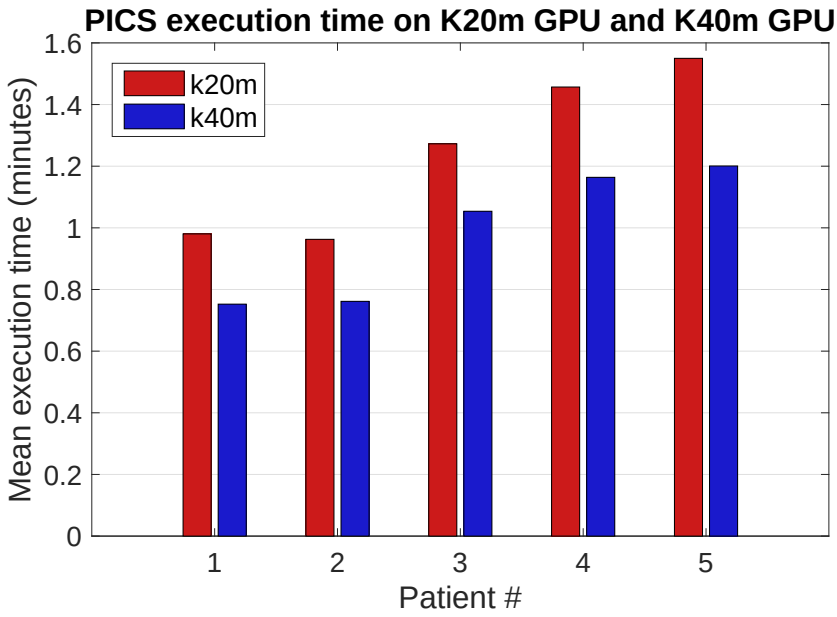


Figure 4.5: Execution times of the PICS stage in the ℓ1-ESPIRiT CS reconstruction algorithm on the CPU with OpenMP + k20m GPU and on the CPU with OpenMP + k40m GPU for 3D datasets in 5 patients.

was not supporting the dimensions of our datasets for GPU implementation. Therefore, we had to either change the BART structure or reduce the dataset dimensions. We chose the second option. We applied a lossless compression on the datasets to avoid loss of quality. For that a Matlab script was written to remove the redundant zero-fill data from the frequency encode direction; making the actual dimension along the frequency encode direction for all of the datasets to be 256 instead of 512. Hence we cut the dataset sizes by half. By adding this pre-processing stage, we could use the BART framework without modification for GPU implementation.

Table 4.1 shows the execution times of the PICS stage and the complete reconstruction process for CPU, CPU with OpenMP, and CPU with OpenMP + GPU. The mean processing time to reconstruct the five 3D patient datasets was 13.08 $\pm$ 3.82 minutes on the CPU, 11.52 $\pm$ 3.58 minutes on the CPU with OpenMP, 2.20 $\pm$ 0.34 minutes on the CPU with OpenMP + k20m GPU, and 1.67 $\pm$ 0.26 minutes on the CPU with OpenMP + k40m GPU. The pairwise differences between the execution times on CPU, CPU with OpenMP, and CPU with OpenMP + GPU for all datasets were significant (p-value$<$0.05).

The total execution time of CS reconstruction in different implementations varied for the

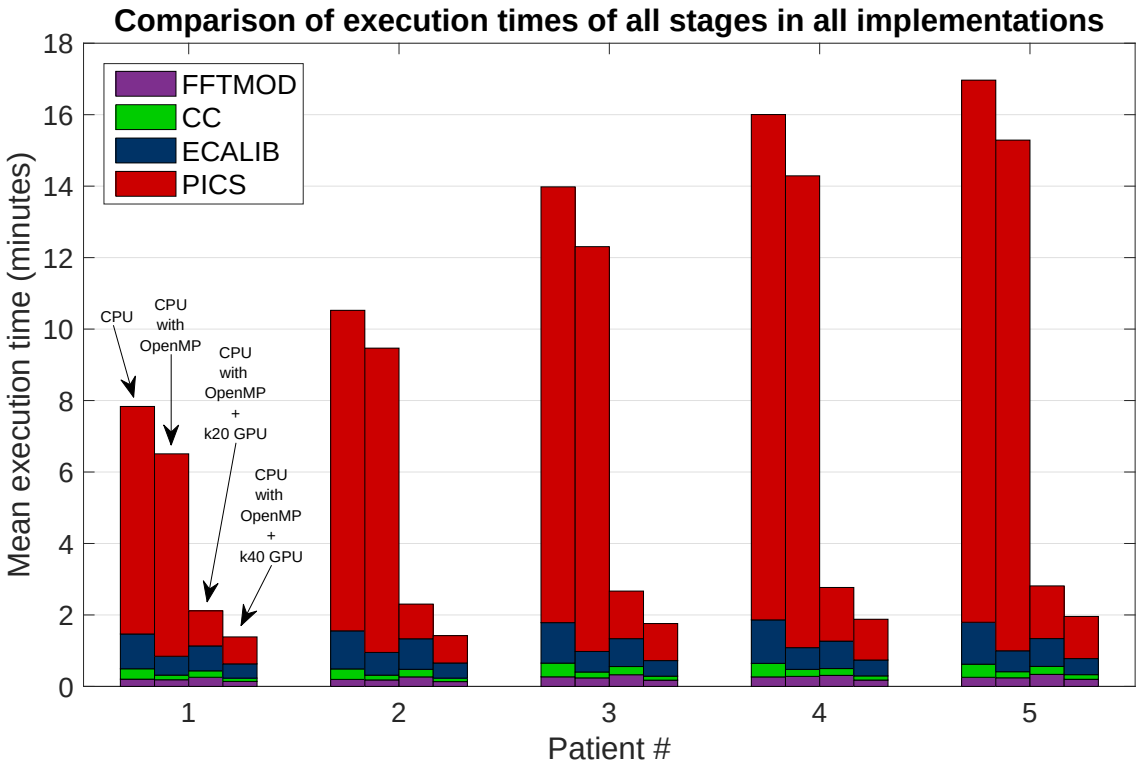


Figure 4.6: The execution times for the main stages of the ℓ1-ESPIRiT CS reconstruction algorithm on CPU, CPU with OpenMP, CPU with OpenMP plus k20m GPU and CPU with OpenMP plus k40m GPU. In all implementations, the PICS stage is the major contributor to the execution time for all datasets.

datasets with different sizes. However, this variation was smaller on the CPU with OpenMP + GPU compared to the CPU implementation (0.3 minutes for the k20m and k40m GPUs vs. 3.8 minutes for CPU). We observed similar behavior from a different perspective. As shown in Figure 4.7, we studied the trend in speed up factors by accelerating the PICS stage using the k40m GPU and the k20m GPU over the CPU implementations when the size of datasets increases. Figure 4.7 shows that, by increasing the dataset size the speed up factor also increases, however the rate of the increase in speed up factor reduces at the same time up to a point that for dataset 5 we observe a saturation in the speed up curve. Since the k20m GPU has less memory capacity, the saturation point for that appears earlier. We envision that if we use datasets with considerably larger sizes, we could even observe a decrease in speed up factor similar to the results illustrated by Smith et al. [30]. This could stem from the fact that the GPU computation limits are not yet reached and the speed up is only bounded by the data transfer bandwidth and latency to the GPU. The reason that we might see a decrease in speed up factor by further increasing in the dataset size is that, the GPU memory has limited capacity which is usually smaller than CPU memory capacity. Therefore, when the dataset size is too large that prevents holding enough data on GPU memory for proceeding computation, the latency overhead of transferring the data from CPU memory to GPU memory comes into the picture. This latency increases the GPU execution time significantly, leading to a decreased speed up factor.

Table 4.1: Execution times of the PICS stage on CPU and GPU as well as total execution times of ℓ1-ESPIRiT CS algorithm on CPU, CPU with OpenMP parallelization, and CPU with OpenMP parallelization plus GPU for PICS for 3D datasets in 5 patients. Note that the dataset dimensions reported here are after pre-processing to reduce the dataset sizes.

	PICS execution time (minutes)				Total execution time (minutes)			
Data Size	CPU	CPU+OpenMP	k20m GPU	k40m GPU	CPU	CPU+OpenMP	+ k20m GPU	+ k40m GPU
256×128×113	6.36	5.66	0.98	0.75	7.83	6.49	1.82	1.37
256×145×96	8.97	8.51	0.96	0.76	10.52	9.47	1.87	1.41
256×141×128	12.19	11.33	1.33	1.03	13.98	12.22	2.25	1.74
256×170×121	14.14	13.20	1.46	1.14	16.00	14.17	2.45	1.88
256×157×124	15.17	14.29	1.55	1.17	16.97	15.28	2.59	1.93

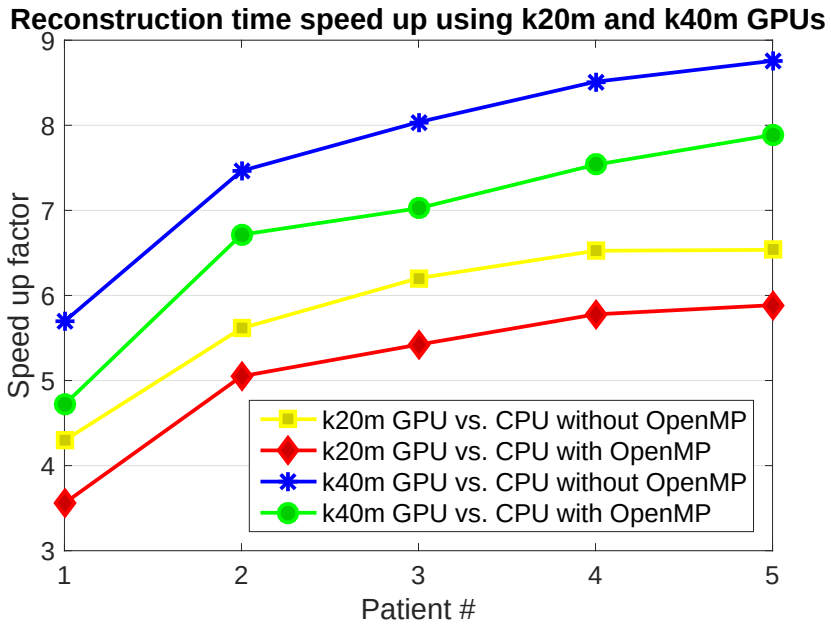
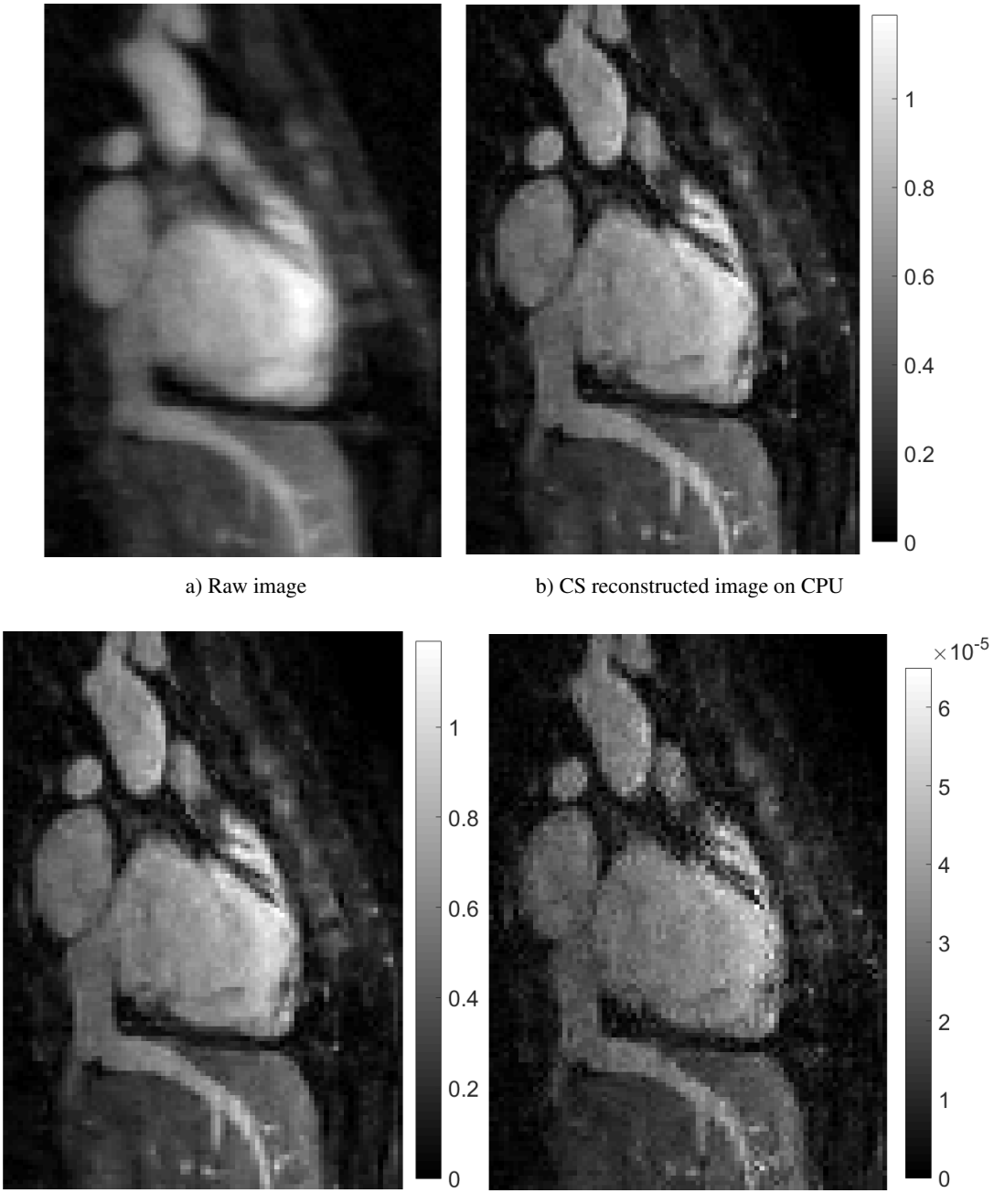


Figure 4.7: Speed up factor vs. dataset size for CPU with OpenMP + k40m GPU implementation over CPU with and without OpenMP implementations.

4.2 Quality Comparison

Figure 4.8 shows 3D-SSFP images acquired from a patient before and after image reconstruction using a CPU and a GPU, and the difference between the images reconstructed on CPU and GPU. For all 5 patient datasets, the mean absolute difference between the CPU and GPU reconstructed images was 2.2e-06 and the mean maximum absolute difference was 6.9e-04. The mean absolute difference between the images reconstructed on CPU and GPU were on the order of 1e-05, and the images were very similar. This small mean absolute difference is potentially due to the fact that different order of operations could lead to different results in floating-point arithmetic. In our study, the PICS stage was implemented using sequential coding on the CPU and parallel coding on the GPU, which yielded to different order of operations and therefore unequal results [47].



a) Raw image

b) CS reconstructed image on CPU

c) CS reconstructed image on GPU

d) Difference image (b-c)

Figure 4.8: An example of 3D-SSFP images in sagittal view acquired from a patient with congenital heart disease: (a) raw image before CS reconstruction, (b) ℓ1-ESPIRiT CS reconstructed image on CPU, (c) ℓ1-ESPIRiT CS reconstructed image on GPU, and (d) the absolute difference between the reconstructed images on CPU and GPU. The mean absolute difference between the CPU and GPU reconstructed images was 1.37e-05 with the maximum difference of 6.49e-05.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, the goal was to increase the efficiency of cardiac MR imaging. Long cardiac MR imaging time can have multiple unfavorable effects. During a long imaging time, patients heart rate, breathing pattern, and body position may change which may lead to a non-diagnostic image quality. To prevent these distortions, clinicians may have to use sedation for some patients. Sedating patients for a long time could cause serious neurological damage and also increase the MRI cost. These issues motivated us to look into improving the efficiency of cardiac MRI in clinical settings by reducing the MRI examination time using the CS reconstruction algorithm. We acquired five undersampled patient datasets and accelerated the corresponding CS reconstruction algorithm using GPUs.

To the best of our knowledge, this is the first study which evaluates ℓ1-ESPIRiT CS reconstruction algorithm on a GPU for 3D cardiac MRI datasets with different sizes acquired from patients. We showed that utilization of a GPU reduces the CS image reconstruction time to less than 1.6 minutes without compromising image quality. This advance should facilitate the use of CS MRI in clinical settings and could make cardiac MR imaging more efficient.

5.2 Limitations and Future Work

The number of subjects in our study was small. We are planning to apply the CS MRI algorithm on more patients to expand our study and further evaluate the CS MRI methods in clinical settings.

We want to reduce the reconstruction time further to less than a minute per 3D dataset. For that we envision the following steps:

- Use more advanced GPUs, for example NVIDIA k80 GPUs.
- Implement the other stages of the $\ell 1$-ESPIRiT algorithm on GPUs. Currently, only the PICS stage was implemented on GPUs while other stages were executed on a CPU with OpenMP parallelization.
- Enhance the library components that are used in BART. In addition to custom designed library components, BART uses third party implementations of linear algebra package (LAPACK) for linear algebraic and matrix operations, such as Cholesky decomposition, and the CUDA library for the GPU implementation of FFT. We plan to upgrade these libraries to the latest versions or replace them with more optimized ones to improve reconstruction performance. This may require changes to the structure of BART to support the new versions of these libraries.
- Change the Matlab interface to a C/C++ interface to reduce the considerable latency overhead that is associated with the Matlab environment.
- Connect the MRI scanner to a computer cluster and automate the reconstruction process. This way, the acquired undersampled datasets will directly sent to a computer cluster directly from the scanner, then the reconstruction will be executed on the cluster, and finally the reconstructed images will be retrieved from the cluster to the clinician's workstation for diagnosis.
- Expand to multiple GPUs. A single GPU node was used in our study for accelerating the $\ell 1$-ESPIRiT reconstruction algorithm. A multi-GPU implementation using message passing interface and CUDA can be used for further improvement in the reconstruction time for 3D or higher dimensional datasets.
- Compare the performance of different reconstruction algorithms on patient datasets.

Bibliography

[1] P. J. Kilner, T. Geva, H. Kaemmerer, P. T. Trindade, J. Schwitter, and G. D. Webb, "Recommendations for cardiovascular magnetic resonance in adults with congenital heart disease from the respective working groups of the European Society of Cardiology," *European Heart Journal*, vol. 31, no. 7, pp. 794–805, 2010.

[2] R. C. Hendel, M. R. Patel, C. M. Kramer, M. Poon, J. C. Carr, N. A. Gerstad, L. D. Gillam, J. Hodgson, R. J. Kim, J. R. Lesser, E. T. Martin, J. V. Messer, R. F. Redberg, G. D. Rubin, J. S. Rumsfeld, A. J. Taylor, G. Weigold, P. K. Woodard, R. G. Brindis, P. S. Douglas, E. Peterson, M. J. Wolk, and J. M. Allen, "ACCF/ACR/SCCT/SCMR/ASNC/NASCI/SCAI/SIR 2006 appropriateness criteria for cardiac computed tomography and cardiac magnetic resonance imaging," *Journal of the American College of Cardiology*, vol. 48, no. 7, pp. 1475–1497, 2006.

[3] T. S. Sørensen, P. Beerbaum, J. Mosegaard, A. Rasmusson, T. Schaeffter, C. Austin, R. Razavi, and G. F. Greil, "Virtual cardiotomy based on 3-D MRI for preoperative planning in congenital heart disease," *Pediatric Radiology*, vol. 38, no. 12, pp. 1314–1322, 2008.

[4] T. S. Sørensen, H. Körperich, G. F. Greil, J. Eichhorn, P. Barth, H. Meyer, E. M. Pedersen, and P. Beerbaum, "Operator-independent isotropic three-dimensional magnetic resonance imaging for morphology in congenital heart disease: A validation study," *Circulation - American Heart Association*, vol. 110, no. 2, pp. 163–169, 2004.

[5] M. Fenchel, G. F. Greil, P. Martirosian, U. Kramer, F. Schick, C. D. Claussen, L. Sieverding, and S. Miller, "Three-dimensional morphological magnetic resonance imaging in infants and children with congenital heart disease," *Pediatric Radiology*, vol. 36, no. 12, pp. 1265–1272, 2006.

[6] L. Sun, "Early childhood general anaesthesia exposure and neurocognitive development," *British Journal of Anaesthesia*, vol. 105, no. suppl 1, pp. i61–i68, 2010.

[7] M. H. Moghari, D. Annese, T. Geva, and A. J. Powell, "Three-dimensional heart locator and compressed sensing for whole-heart MR angiography," *Magnetic Resonance in Medicine*, 2015 (in press).

[8] Nvidia. CUDA parallel computing platform. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html

[9] M. Uecker, F. Ong, J. I. Tamir, D. Bahri, P. Virtue, J. Y. Cheng, T. Zhang, and M. Lusting, "Berkeley advanced reconstruction toolbox," *ISMRM*, vol. 23, p. 2486, 2015.

[10] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "ESPIRiT an eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA," *Magnetic Resonance in Medicine*, vol. 71, no. 3, pp. 990–1001, 2014.

[11] G. A. Wright, "Magnetic resonance imaging," *IEEE Signal Processing Magazine*, vol. 14, no. 1, pp. 56–66, 1997.

[12] P. T. Norton, N. C. Nacey, D. B. Caovan, S. B. Gay, C. M. Kramer, and B. S. Jeun. Steady state free precession. [Online]. Available: https://www.med-ed.virginia.edu/courses/rad/cardiacmr/Techniques/SSFP.html

[13] A. D. Elster. What is k-space? [Online]. Available: http://mriquestions.com/what-is-k-space.html

[14] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger, "SENSE: Sensitivity encoding for fast MRI," *Magnetic Resonance in Medicine*, vol. 42, no. 5, pp. 952–962, 1999.

[15] M. A. Griswold, P. M. Jakob, M. Nittka, J. W. Goldfarb, and A. Haase, "Partially parallel imaging with localized sensitivities (PILS)," *Magnetic Resonance in Medicine*, vol. 44, no. 4, pp. 602–609, 2000.

[16] D. K. Sodickson and W. J. Manning, "Simultaneous acquisition of spatial harmonics (SMASH): Fast imaging with radiofrequency coil arrays," *Magnetic Resonance in Medicine*, vol. 38, no. 4, pp. 591–603, 1997.

[17] M. A. Griswold, P. M. Jakob, R. M. Heidemann, M. Nittka, V. Jellus, J. Wang, B. Kiefer, and A. Haase, "Generalized autocalibrating partially parallel acquisitions (GRAPPA)," *Magnetic Resonance in Medicine*, vol. 47, no. 6, pp. 1202–1210, 2002.

[18] W. E. Kyriakos, L. P. Panych, D. F. Kacher, C.-F. Westin, S. M. Bao, R. V. Mulkern, and F. A. Jolesz, "Sensitivity profiles from an array of coils for encoding and reconstruction in parallel (SPACE RIP)," *Magnetic Resonance in Medicine*, vol. 44, no. 2, pp. 301–308, 2000.

[19] W. S. Hoge, D. H. Brooks, B. Madore, and W. E. Kyriakos, "A tour of accelerated parallel MR imaging from a linear systems perspective," *Concepts Magn Reson Part A*, vol. 27A, no. 1, pp. 17–37, 2005.

[20] P. Lai, P. Ghedin, G. Pontone, and A. Brau, "Highly accelerated 3D myocardial late fadolinium enhancement MRI using ESPIRiT compressed sensing: initial feasibility," *Journal of Cardiovascular Magnetic Resonance*, vol. 16, no. 1, pp. 1–2, 2014.

[21] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[22] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, p. 12891306, 2006.

[23] E. Candès and T. Tao, "Near optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[24] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[25] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural Image Statistics*. Springer-Verlag London, 2009.

[26] D. Taubman and M. Marcellin, *JPEG2000 image compression fundamentals, standards and practice*. Springer US, 2002.

[27] J. B. Ra and C. Y. Rim, "Fast imaging using subencoding data sets from multiple detectors," *Magnetic Resonance in Medicine*, vol. 30, no. 1, pp. 142–145, 1993.

[28] M. Lustig and J. M. Pauly, "SPIRiT: Iterative self-consistent parallel imaging reconstruction from arbitrary k-space," *Magnetic Resonance in Medicine*, vol. 64, no. 2, pp. 457–471, 2010.

[29] J. C. Park, B. Song, J. S. Kim, S. H. Park, H. K. Kim, Z. Liu, T. S. Suh, and W. Y. Song, "Fast compressed sensing-based CBCT reconstruction using Barzilai-Borwein formulation for application to on-line IGRT," *Medical Physics*, vol. 39, no. 3, pp. 1207–1217, 2012.

[30] D. S. Smith, J. C. Gore, T. E. Yankeelov, and E. B. Welch, "Real-time compressive sensing MRI reconstruction using GPU computing and Split Bregman methods," *International Journal of Biomedical Imaging*, vol. 2012, no. 864827, pp. 1–6, 2012.

[31] T. M. Quan, S. Han, H. Cho, and W.-K. Jeong, "Multi-GPU reconstruction of dynamic compressed sensing MRI," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2015*, vol. 9351, 2015, pp. 484–492.

[32] M. S. Hansen, D. Atkinson, and T. S. Sorensen, "Cartesian SENSE and k-t SENSE reconstruction using commodity graphics hardware," *Magnetic Resonance in Medicine*, vol. 59, no. 3, pp. 463–468, 2008.

[33] A. M. Kulkarni, H. Homayoun, and T. Mohsenin, "A parallel and reconfigurable architecture for efficient OMP compressive sensing reconstruction," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, 2014, pp. 299–304.

[34] J. Stanislaus and T. Mohsenin, "Low-complexity FPGA implementation of compressive sensing reconstruction," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*, 2013, pp. 671–675.

[35] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-speed compressed sensing reconstruction on FPGA using OMP and AMP," in *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, 2012, pp. 53–56.

[36] B. E. Nett, J. Tang, and G.-H. Chen, "GPU implementation of prior image constrained compressed sensing (PICCS)," *Proc. SPIE*, vol. 7622, p. 762239, 2010.

[37] J. Chen, J. Cong, M. Yan, and Y. Zou, "FPGA-accelerated 3D reconstruction using compressive sensing," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2012, pp. 163–166.

[38] D. Kim, J. D. Trzasko, M. Smelyanskiy, C. R. Haider, A. Manduca, and P. Dubey, "High-performance 3D compressive sensing MRI reconstruction," in *Engineering in Medicine and*

Biology Society (EMBC), 2010 Annual International Conference of the IEEE, 2010, pp. 3321–3324.

[39] S. S. Stone, J. P. Haldar, S. C. Tsao, W.-m. W. Hwu, Z.-P. Liang, and B. P. Sutton, "Accelerating advanced MRI reconstructions on GPUs," *Journal of Parallel and Distributed Computing*, vol. 68, no. 10, pp. 1307–1318, 2008.

[40] S. Nam, M. Akakaya, T. Basha, C. Stehning, W. J. Manning, V. Tarokh, and R. Nezafat, "Compressed sensing reconstruction for whole-heart imaging with 3D radial trajectories: A GPU implementation," *Magnetic Resonance in Medicine*, vol. 69, no. 1, pp. 91–102, 2013.

[41] Northeastern university. Overview of discovery cluster. [Online]. Available: http://nuweb12.neu.edu/rc/?page_id=27

[42] ARB. The OpenMP® API specification for parallel programming. [Online]. Available: http://openmp.org/wp/

[43] F. Ong, M. Uecker, U. Tariq, A. Hsiao, M. T. Alley, S. S. Vasanawala, and M. Lustig, "Robust 4D flow denoising using divergence-free wavelet transform," *Magnetic Resonance in Medicine*, vol. 73, no. 2, pp. 828–842, 2015.

[44] I. Free Software Foundation. Gcc, the gnu compiler collection. [Online]. Available: https://gcc.gnu.org/

[45] NVIDIA. Nvidia cuda compiler driver nvcc. [Online]. Available: http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/#abstract

[46] D. W. Stockburger. One and two-tailed t-tests. [Online]. Available: http://www.psychstat.missouristate.edu/introbook/sbk25m.htm

[47] Y. Gu, T. Wahl, M. Bayati, and M. Leeser, "Behavioral non-portability in scientific numeric computing," in *Euro-Par 2015: Parallel Processing*. Springer, 2015, pp. 558–569.