



MAGNETIC RESONANCE IMAGE RECONSTRUCTION
USING COMPRESSED SENSING ON EMBEDDED
PROCESSING PLATFORMS

By

Yassin Amer El-Sayed Amer

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Systems & Biomedical Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2013

MAGNETIC RESONANCE IMAGE RECONSTRUCTION
USING COMPRESSED SENSING ON EMBEDDED
PROCESSING PLATFORMS

By
Yassin Amer El-Sayed Amer

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Systems & Biomedical Engineering

Under the Supervision of

Prof. Dr. Yasser M. Kadah

.....
Professor of Biomedical Engineering
Systems & Biomedical Engineering
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2013

MAGNETIC RESONANCE IMAGE RECONSTRUCTION
USING COMPRESSED SENSING ON EMBEDDED
PROCESSING PLATFORMS

By
Yassin Amer El-Sayed Amer

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Systems & Biomedical Engineering

Approved by the
Examining Committee

Prof. Dr. Yasser M. Kadah, Thesis Main Advisor

Prof. Dr. Nahed H. Solouma, Internal Examiner

Prof. Dr. Mohamed I. El-Adawy, External Examiner

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2013

Engineer's Name: Yassin Amer El-Sayed Amer
Date of Birth: 5/3/1989
Nationality: Egyptian
E-mail: yassin.amer@eng.cu.edu.eg
Phone: 01069699880
Address: Quweisna – Almenoufia Governorate
Registration Date: 1/10/2010
Awarding Date: / /
Degree: Master of Science
Department: Systems and Biomedical Engineering



Supervisors:

Prof. Dr. Yasser M. Kadah

Examiners:

Prof. Dr. Yasser M. Kadah (Thesis main advisor)
Prof. Dr. Nahed H. Solouma (Internal examiner)
Prof. Dr. Mohmaed I. El-Adawy (External examiner)

Title of Thesis:

Magnetic Resonance Image Reconstruction Using Compressed Sensing On Embedded Processing Platforms

Key Words:

Magnetic Resonance Imaging; Image Reconstruction; Compressed Sensing; L0 Norm BeagleBoard; OMAP; DSP.

Summary:

Magnetic Resonance Imaging (MRI) is an essential medical imaging modality. Enduring a diagnostic session in an MR machine means lying motionless for a long time (up to 45 minutes) which is very uncomfortable besides the countless ear-ringing, bangs, knocks and the image artifacts which may appear due to motion. So, reducing the scan time (Acquisition time) of an MR image is a very important issue. A newly emerged theory known as Compressed Sensing (CS) is a novel sampling paradigm which reduces the number of measurements needed for image reconstruction with no significant degradation in image quality. Applying CS for MRI offers significant scan time reductions and the availability of embedded reconstruction platforms based on such techniques will be very beneficial in size and cost reduction. An MR image reconstruction algorithm based on CS was built, tested and modified in order to produce images of higher quality in shorter reconstruction times under the same sampling conditions, and its performance was tested on different platforms including an embedded platform based on OMAP processor. This work showed good results for the quality of images reconstructed from highly undersampled k-space (up to 90 % reduction) using CS also the performance of the algorithm on the embedded platform was interesting and points to several future directions for performance optimization in utilizing such embedded platforms in practical medical imaging applications.

Acknowledgment

Firstly, I would like to thank ALLAH, the source of every gift.

Secondly, I would like to thank Prof. Yasser Kadah for his continuous support, valuable guidance and great science which you will always find when you need them. Also an important part of this research would not have been possible without the support of Eng. Mostafa El-Tager and Eng. Ehab El-Alamy.

Finally for my family especially my parents, you are my real gift.

Dedication

To Egypt.

Table of Contents

ACKNOWLEDGMENT	I
DEDICATION	II
TABLE OF CONTENTS	III
LIST OF TABLES	V
LIST OF FIGURES	VI
NOMENCLATURE	VIII
ABSTRACT	IX
CHAPTER 1 : INTRODUCTION	1
1.1. PROBLEM OVERVIEW	1
1.2. THESIS OBJECTIVE	1
1.3. THESIS ORGANIZATION	1
CHAPTER 2 : MR IMAGE ACQUISITION	3
2.1. SPIN ECHO (SE) PULSE SEQUENCE	3
2.2. SPATIAL ENCODING	3
2.3. DATA SPACE	4
2.4. SCAN TIME.....	5
CHAPTER 3 : COMPRESSED SENSING	7
3.1. TECHNIQUE OVERVIEW	7
3.2. SPARSITY	7
3.3. INCOHERENCE	8
3.4. COMPRESSED SENSING MRI	8
3.5. IMAGE RECOVERY (RECONSTRUCTION PROBLEM).....	10
CHAPTER 4 : BEAGLEBOARD (EMBEDDED PLATFORM)	13
4.1. SYSTEM OVERVIEW	13
4.2. PROCESSOR	14
4.3. USAGE SCENARIOS.....	15
CHAPTER 5 : PERFORMANCE ENHANCEMENT OF CS ALGORITHM	17
5.1. CS ALGORITHM ARCHITECTURE.....	17
5.2. IMAGE QUALITY EVALUATION METRICS	19
5.2.1. Mean Squared Error (MSE).....	19
5.2.2. Geometric Average Error (GAE)	19
5.2.3. Quality Index (QI)	20
5.3. CS ALGORITHM PERFORMANCE.....	20
5.4. CS USING L0-NORM MINIMIZATION	21

5.5.	CS USING FOURIER TRANSFORM AS THE SPARSE TRANSFORM FOR MR REAL	DATA	26
5.6.	DISCUSSION		27
CHAPTER 6 : EMBEDDED IMPLEMENTATION.....			39
6.1.	CS ALGORITHM PREPARATION		39
6.2.	EMBEDDED RECONSTRUCTION.....		40
6.2.1.	Arm Based Reconstruction System		41
6.2.2.	Hybrid processor based Reconstruction System.....		41
6.3.	RESULTS AND DISCUSSION.....		41
CONCLUSIONS AND FUTURE WORK			43
REFERENCES			45
APPENDIX A: GETTING BEAGLEBOARD READY			49

List of Tables

Table 2.1: Different MRI Scan Times.....	5
Table 5.1: Real MR image MSE with Fourier as a sparse transform.....	36
Table 5.2: Real MR image QI with Fourier as a sparse transform.....	37
Table 5.3: Real MR image MSE with Wavelet as a sparse transform	37
Table 5.4: Real MR image QI with Wavelet as a sparse transform	37
Table 5.5: Clear Shepp Logan MSE.....	37
Table 5.6: Clear Shepp Logan QI.....	37
Table 5.7: Noised Shepp Logan MSE	38
Table 5.8: Noised Shepp Logan QI	38
Table 5.9: Noised Shepp Logan SNR.....	38
Table 5.10: Real MR image reconstruction time with Fourier as sparse transform	38
Table 5.11: Real MR image reconstruction time with Wavelet as sparse transform	38
Table 6.1: Processing time.	42
Table A.1: Environment Variables.....	55

List of Figures

Figure 2.1: Pulse sequence diagram [1].	3
Figure 2.2: Spatially Encoded Sequence [1].	4
Figure 2.3: Data Space [1].	4
Figure 3.1: (a) A gray-level image. (b) Image Wavelet transform. (c) Reconstruction after zeroing out [9].	8
Figure 3.2: (a) A sparse signal. (b) Its K-Space. (c) Incoherent interference due to random undersampling. (d) Aliasing due to uniform undersampling. (e), (f) Isolation of Strong components. (h), (g) Lowering interference and weak component isolation [6].	9
Figure 3.3: (a) MR Images. (b) Images in the sparse domain [6].	9
Figure 3.4: (a) A sparse signal. (b) Reconstruction using l_1 minimization. (c) Reconstruction using l_2 minimization [9].	11
Figure 4.1: BeagleBoard [30].	13
Figure 4.2: OMAP35xx Block Diagram.	14
Figure 5.1: Conjugate gradient Algorithm block diagram.	18
Figure 5.2: (a) 10% pattern. (b) 33.3% pattern.	18
Figure 5.3: (a) Shepp logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	22
Figure 5.4: (a) Noised Shepp logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	23
Figure 5.5: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using CS with undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	24
Figure 5.6: MSE & QI calculated for CS reconstructions compared to full k-space reconstruction for the three data types. (a) & (b) MSE and QI of Shepp logan. (c) & (d) MSE and QI of Noised Shepp Logan. (e) & (f) MSE and QI of Brain Image.	25
Figure 5.7: (a) 10% undersampling pattern. (b) 33.3% undersampling pattern.	26
Figure 5.8: (a) Shepp Logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	29
Figure 5.9: (a) Noised Shepp Logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	30
Figure 5.10: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	31
Figure 5.11: Quality comparison between using CS with L1-Norm & L0-Norm, (a) & (b) MSE and QI of Shepp Logan results. (c) & (d) MSE and QI of Noised Shepp Logan results. (e) & (f) MSE and QI of Brain image results.	32
Figure 5.12: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using CS with Fourier transform and L1-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	33
Figure 5.13: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using Fourier based CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.	34

Figure 5.14: Quality Comparison, (a) & (b) MSE & QI for brain image reconstructions using Fourier based CS with L0-Norm & L1-Norm. (c) & (d) MSE & QI for Fourier based CS vs. Wavelet based CS with L1-Norm. (c) & (d) MSE & QI for Fourier based CS vs. Wavelet based CS with L0-Norm.	35
Figure 5.15: Performance comparison (a) Processing time for reconstructions using Wavelet based CS with L0-Norm & L1-Norm. (b) Processing time for using Fourier based CS with L0-Norm & L1-Norm. (c) Processing time for using L1-Norm with Wavelet and Fourier. (d) Processing time for using L0-Norm with Wavelet and Fourier.	36
Figure 6.1: (a) Original image. (b) Sampling pattern. (c) Zero filling with density compensation reconstruction.	39
Figure 6.2: CS algorithm block diagram	40
Figure 6.3: Communication with DSP.	40
Figure 6.4: CS based reconstructed image.	42
Figure A.1: Kernel menu configuration.	52
Figure A.2: Kernel menu configuration.	52
Figure A.3: Kernel menu configuration.	53
Figure A.4: SD card partitions.....	54
Figure A.5: "boot.cmd" file.....	55
Figure A.6: using minicom command.	56

Nomenclature

ARM	Advanced RISC Machines
BB	BeagleBoard
BIOS	Basic Input/Output System
CG	Conjugate Gradient
CS	Compressed Sensing
DSP	Digital Signal Processor
GAE	Geometric Average Error
GPP	General Purpose Processor
HLOS	High Level Operating System
JPEG	Joint Photographic Experts Group
MLO	Memory Loader
MRCP	Magnetic Resonance Cholangiopancreatography
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
OMAP	Open Media Applications Platform
OS	Operating System
QI	Quality Index
RAM	Random Access Memory
RTOS	Real Time Operating System
SD	Secure Digital
SE	Spin Echo
SNR	Signal to Noise Ratio
TE	Time to Echo
TI	Texas Instruments
TMG	Temporomandibular Joint
TPSF	Transform Point Spread Function
TR	Repetition Time
ZF-w/dc	Zero Filling with Density Compensation

Abstract

Magnetic Resonance Imaging (MRI) is an essential medical imaging modality. Enduring a diagnostic session in an MR machine means lying motionless for a long time (up to 45 minutes) which is very uncomfortable besides the countless ear-ringing, bangs, knocks and the image artifacts which may appear due to motion. So, reducing the scan time (Acquisition time) of an MR image is a very important issue. A newly emerged theory known as Compressed Sensing (CS) is a novel sampling paradigm which reduces the number of measurements needed for image reconstruction with no significant degradation in image quality. Applying CS for MRI offers significant scan time reductions and the availability of embedded reconstruction platforms based on such techniques will be very beneficial in size and cost reduction. An MR image reconstruction algorithm based on CS was built, tested and modified in order to produce images of higher quality in shorter reconstruction times under the same sampling conditions, and its performance was tested on different platforms including an embedded platform based on OMAP processor. This work showed good results for the quality of images reconstructed from highly undersampled k-space (up to 90 % reduction) using CS also the performance of the algorithm on the embedded platform was interesting and points to several future directions for performance optimization in utilizing such embedded platforms in practical medical imaging applications.

Chapter 1 : Introduction

1.1. Problem Overview

Magnetic Resonance Imaging (MRI) is one of the common and powerful medical imaging procedures used at the current time due to its ability to show soft tissue structures, such as ligaments and internal organs like brain. MRI can be safely used with people who are vulnerable to the effects of ionizing radiation, such as pregnancy cases and babies also to be used to image in any plane. All of this has made MRI an essential medical imaging modality but having an MRI scan is very uncomfortable because of the long scan times (20-60 minutes or more) which means lying motionless inside the gantry during this period besides hearing noisy sounds and this may prevent many claustrophobic people from undergoing an MRI scan. MR images also may suffer motion artifacts and the patient may repeat the scan many times till having a good image which will be very uncomfortable and costly. So using techniques that reduce MRI scan times will help to raise the efficiency of this important modality.

1.2. Thesis Objective

This work aims to reduce MRI scan time by minimizing the number of measurements used to reconstruct an MR image through the use of a newly emerged theory called Compressed Sensing (CS) which uses nonlinear reconstruction themes to build images of good quality (with the same resolution) using far smaller number of measurements, and build a cheap and fast reconstruction platform using CS, and based on the use embedded and special function processors.

1.3. Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 2 provides a background about MR image acquisition process.
- Chapter 3 provides an overview on compressed sensing and its usage with MRI.
- Chapter 4 provides an overview about the embedded platform used (BeagleBoard).
- Chapter 5 provides a detailed description for the methods used to enhance the performance of the CS algorithm with MR images.
- Chapter 6 provides a detailed description for the embedded implementation of the CS based reconstruction algorithm.
- Appendix A provides detailed steps for building an operating system for BeagleBoard and getting it ready to be used.

Chapter 2 : MR Image Acquisition

The purpose of this chapter is to give a brief overview about the MR image acquisition process, the image data space, k-Space, and acquisition time.

2.1. Spin Echo (SE) Pulse Sequence

Spin Echo (SE) is the most frequently used pulse sequence (a sequence of radio frequency pulses) during an MR study. The sequence starts with the 90° which causes the magnetization vector M_Z to be flipped into the x-y plane. After the 90° pulse the spins will get out of phase with each other due to magnetic field inhomogeneity. At a certain time τ after the 90° pulse, when the spins have gotten out of phase, a 180° pulse is applied. Now all the spins flip 180° in the x-y plane and they continue precessing, but now in the opposite direction [1]. The pulse sequence diagram is shown in Figure 2.1.

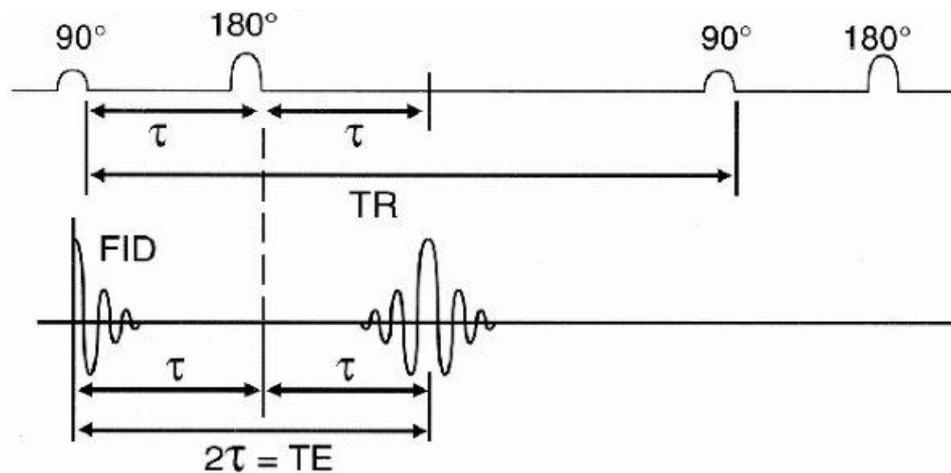


Figure 2.1: Pulse sequence diagram [1].

From Figure 2.1 we start off with a 90° RF pulse to flip the spins into the x-y plane. We wait a time τ and apply a 180° RF pulse. Then we wait a long time, TR (Repetition Time), and repeat the process [1]. Also the time to echo (TE) is the time after the 90° pulse when we get maximum signal again.

2.2. Spatial Encoding

The signals received from a patient contain information about the entire part of the patient being imaged but they do not have any particular spatial information, and to determine the origin of each component of the signal we use the gradients for x, y, and z directions Figure 2.2. These gradients are called:

- The slice-select gradient
- The frequency-encoding gradient
- The phase-encoding gradient

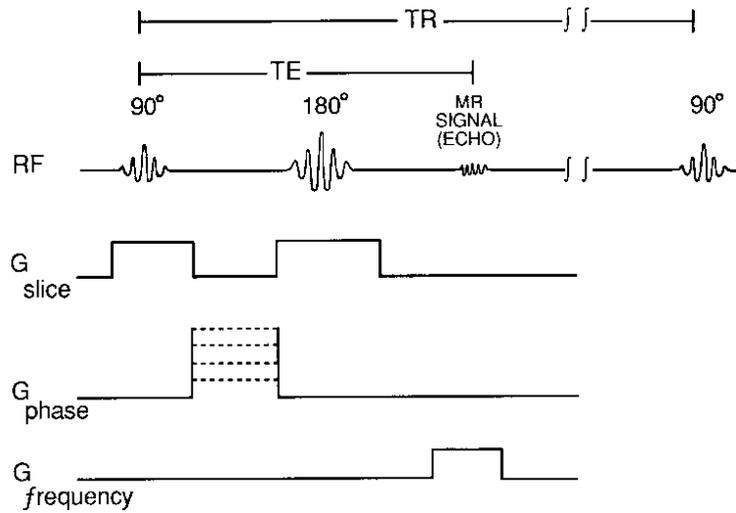


Figure 2.2: Spatially Encoded Sequence [1].

2.3. Data Space

It is the analog version of the k-space, and it is composed of rows of acquired signals at different phase encoding gradients (Figure 2.3) as follows [1]:

1. With TR#1, we have no phase shift. After the frequency-encoding step, a signal is received and placed into the center of the data space.
2. With TR#2, we have no phase shift. After the frequency-encoding step, a signal is received and placed into one above the center of the data space.
3. With TR#3, we have no phase shift. After the frequency-encoding step, a signal is received and placed into one below the center of the data space.
4. Continue till filling the data space in this manner.

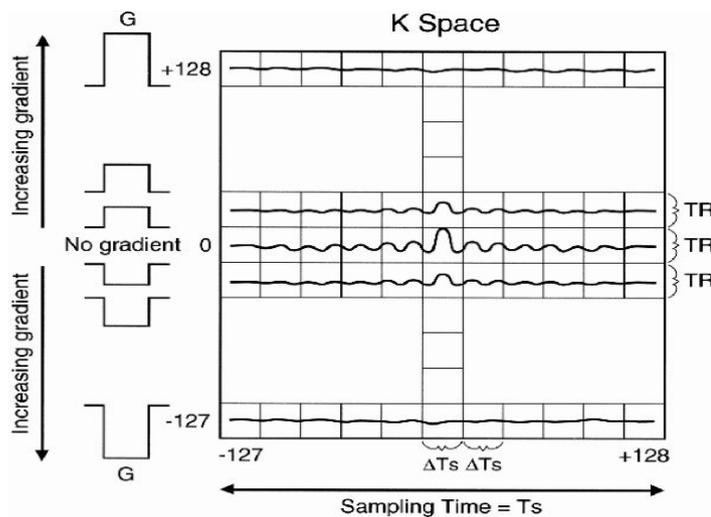


Figure 2.3: Data Space [1].

2.4. Scan Time

The scan time is an important factor in MRI systems. It is directly proportional to the size of the image and depends on also the type of the study being performed. It can be calculated through the simple formula in Eq. (2.1) [2], Table 2.1 shows ranges of scan time for different studies [3].

$$\text{Scan Time} = T_R \times \text{number of phase-encoding steps} \times \text{number of signal averages.} \quad (2.1)$$

Many ways are used in order to reduce the scan time but this comes with the reduction of image quality. One of this ways is to shorten TR and according to this reduction, the SNR will decrease according to the nature that $\text{SNR} \propto \sqrt{\text{Time}}$ [4] also the contrast of image is changed with T_R and if it changes for the worse this will not be useful. Another way to reduce scan time is to reduce the number of phase encoding steps but this causes the volume effects to be worse [4]. So new methods are needed to reduce the scan time without affecting the image quality.

Table 2.1: Different MRI Scan Times.

Scan Type	Scan Time
MRI of the Brain	20-45 minute
MRI of the Orbits	20-35 minute
MRI of the TMJ	45-60 minute
MRI of the Soft Tissue Neck	25-35 minute
MRI of the Cervical Spine	20-35 minute
MRI of the Upper Extremity	20-45 minute
MRI of the Thoracic Spine	25-45 minute
MRI of the Chest	25-45 minute
MRI of the Abdomen	25-45 minute
MRI MRCP	50-60 minute
MRI of the Lumbar Spine	20-35 minute
MRI of the Pelvis	20-35 minute
MRI of the Lower Extremity	20-35 minute
MRI Run Off	50-60 minute
MRI Arthrogram	30-60 minute

Chapter 3 : Compressed Sensing

Imaging speed is limited by many constraints like physical factors (e.g. slew rate in MRI), physiological factors and processing speed [5]. Any imaging system contains two main stages the first is data collection and the second is image reconstruction. The data collection stage depends on the resolution of image collected and field of view [6], [7]. The time needed for image reconstruction depends on the processing power of the machine and complexity of the reconstruction algorithm and of course the size of data [8]. In order to enhance the imaging speed this will be done in one of the previous stages or in both of them, and CS works mainly in the first stage of data acquisition in addition to a modification in the reconstruction process.

The purpose of this chapter is to give a detailed description for the CS algorithm and the natural fit between MRI and CS.

3.1. Technique overview

Conventional sampling approaches of signals or images follow Shannon's theorem which states that the sampling rate must be at least twice the maximum frequency present in the signal in order to be able to completely recover the signal (Nyquist rate) [9], and this underlies nearly all signal acquisition protocols including those used in medical imaging devices. For some signals like images which are not bandwidth-limited, the sampling rate is determined by the desired spatial resolution [9]. However, it is common to use an antialiasing filter to limit the bandwidth of the signal so that Shannon's theorem applies.

Compressed Sensing (CS) is a novel sampling paradigm that goes against the commonly known sampling wisdom, and tries to reduce the measurements needed to reconstruct the signal or image without significantly degrading its quality [10–12]. CS depends on the broad success of lossy compression techniques for signals and images which raises a very natural question: why to go to so much effort to acquire all the data when most of what we get will be thrown away? Can we just directly measure the part that will not end up being thrown away? [11]. So CS is a compressed data acquisition protocol which cares only about acquiring the data that will not be thrown away by lossy compression. In order for CS to be applicable the signal or image should obey two key requirements which are Sparsity and Incoherence [9].

3.2. Sparsity

For CS to be applied the Sparsity condition should exist for the object (signal or image) of interest. Sparsity means that the underlying object has a sparse representation in a known domain.

Many natural signals have sparse representation in if it is expressed in a certain domain. For example if we considered the image in Figure 3.1(a) which represents a gray-level image, and contains pixel values from 0 to 255 and its wavelet transform in (b), we find that despite of having nearly all pixels with non zero value, the wavelet transform provides a concise representation with many near zero coefficients and relatively small few large coefficients [9].

If the image is reconstructed after zeroing out most of the small coefficients in wavelet domain (97.5 % of coefficients) Figure 3.1(c), we see that the difference is hardly noticeable. Sparsity is what underlies most modern lossy coders like JPEG-2000 and others by first applying 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 [9], [13].

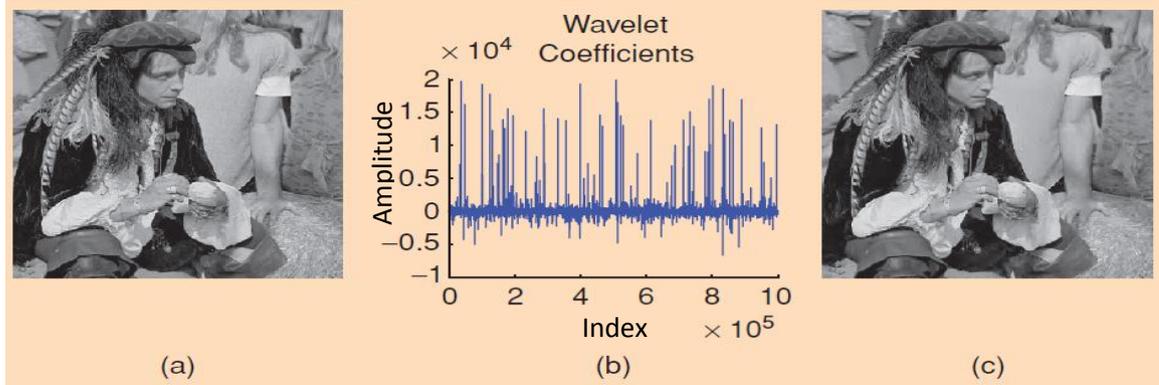


Figure 3.1: (a) A gray-level image. (b) Image Wavelet transform. (c) Reconstruction after zeroing out [9].

3.3. Incoherence

The second condition for CS to be applied is the incoherence which means that the artifacts caused in linear reconstruction due to reduction in data collected should be noise like in the sparsifying domain [5], and this depends on the undersampling scheme used. To be easily understood a 1D sinusoidal signal undersampling example is considered in Figure 3.2; in Figure 3.2(b) we can see the two used undersampling schemes (random and uniform 8-fold undersampling) [6]. The results from uniform undersampling (d) have coherent interference which prevents recovery, but in (c) the interference due to random undersampling can be separated and the signal can be recovered through two stages including strong components recovery using simple thresholding (e), (f), and weak components recovery is done by subtracting the interference calculated for the recovered strong components from the complete signal interference and then component isolation using thresholding (h), (g) [6].

3.4. Compressed Sensing MRI

For successful application of CS in MRI, MR images should obey: (1) to be naturally compressible by sparse coding in a certain transform domain, and (2) the aliasing artifacts due to k-space undersampling be incoherent (noise like) in that transform domain [5]. The first condition applies for MRI as most MR images are sparse in an appropriate transform domain Figure 3.3. First, brain images look sparse in wavelet domain, angiograms in Finite difference, and dynamic heart in temporal frequency [6].

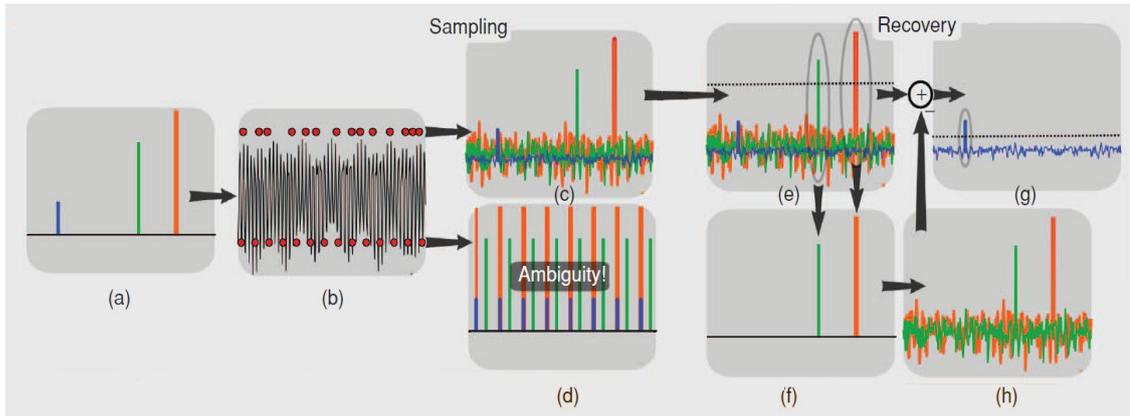


Figure 3.2: (a) A sparse signal. (b) Its K-Space. (c) Incoherent interference due to random undersampling. (d) Aliasing due to uniform undersampling. (e), (f) Isolation of Strong components. (h), (g) Lowering interference and weak component isolation [6].

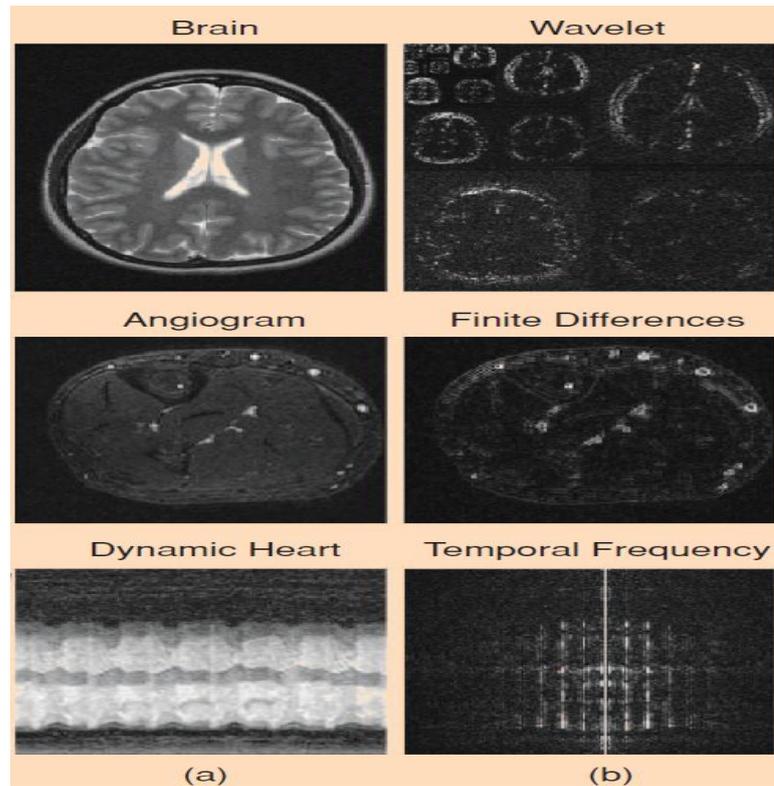


Figure 3.3: (a) MR Images. (b) Images in the sparse domain [6].

Also as included in the 1D signal example in Figure 3.2, we see that a complete random set of samples will be sufficient for incoherent interference [10], [11]. Random point k-space sampling in all dimensions is generally impractical as the k-space trajectories have to be relatively smooth due to hardware and physiological considerations [5], and therefore sampling trajectories must follow relatively smooth lines and curves. Non-Cartesian sampling schemes can be highly sensitive to system

imperfections [6]. So considering Cartesian grid sampling will be more practical where the sampling is restricted to undersampling the phase-encodes and fully sampled readouts [5]. Alternative sampling trajectories are possible and some very promising results have been presented by [14–16] (radial imaging), and by [17], [18] (spiral imaging).

Furthermore, a uniform random distribution of samples in spatial frequency does not take into account the energy distribution of MR images in k-space, which is far from uniform. Most energy in MR imagery is concentrated close to the center of k-space and rapidly decays towards the periphery of k-space [6]. Therefore, realistic designs for CS in MRI should have variable density sampling with denser sampling near the center of k-space, matching the energy distribution in k-space [6]. All those key features of MR images have enabled the use of CS with MRI.

3.5. Image Recovery (Reconstruction Problem)

When using CS, the image should be reconstructed using a nonlinear reconstruction that enforces both the sparsity of the image and consistency of the reconstructed data with the acquired samples [5]. In MRI, CS can be considered to be a special case as the samples are simply individual Fourier coefficients (k-space samples) not pixel values [5].

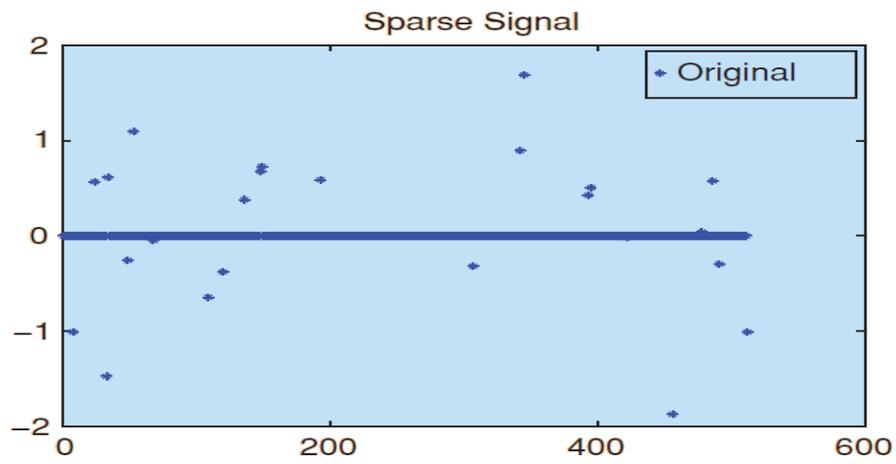
When applying CS with MRI, we only need to acquire a subset S of k-space coefficients and the reconstruction is obtained through solution of the following optimization problem:

$$\text{minimize } \|\psi m\|_1 \quad \text{subject to } \|F_S m - y\|_2 < \varepsilon, \quad (3.1)$$

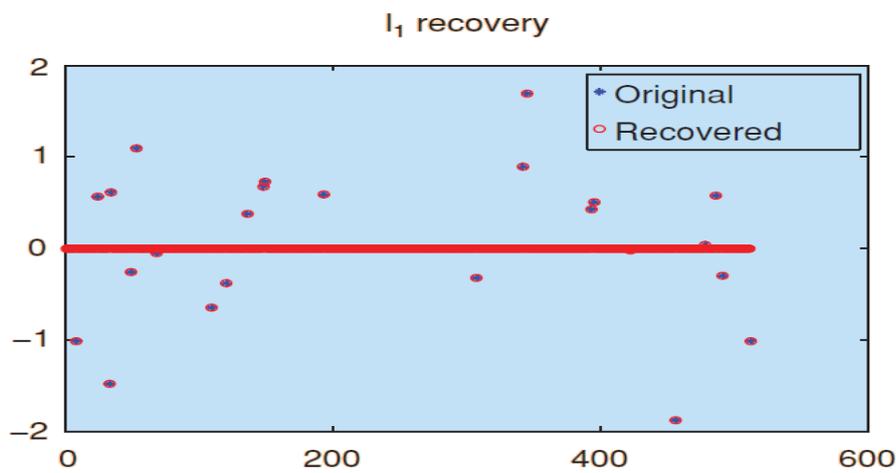
While F_S denotes the Fourier transform evaluated just at frequencies in the subset S , ψ is the sparse transform, m is the reconstructed image, y is the measured k-space data from the MR scanner, and ε controls the fidelity of the reconstructed data [5], [9].

In Error! Reference source not found., the objective function is the l_1 norm which is defined as $\|x\|_1 = \sum_i |x_i|$, and minimizing $\|\psi m\|_1$ promotes sparsity [5], [19]. The constraint $\|F_S m - y\|_2 < \varepsilon$ controls the data consistency. In other words Eq. (2.1) finds a solution that is compressible by ψ [5]. The use of l_1 norm as a sparsity-promoting function traces back several decades. A leading early application was reflection seismology, in which a sparse reflection function (indicating meaningful changes between subsurface layers) was sought from bandlimited data [20], [21]. An example for reconstruction of a 1D signal using l_1 norm vs. l_2 norm minimization is shown in Figure 3.4 [9]. As seen in the example minimizing the l_1 norm shows perfect reconstruction.

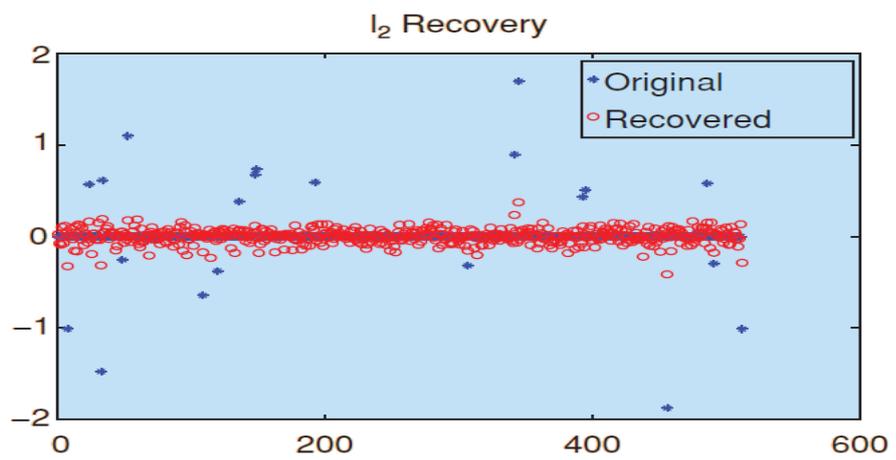
Special purpose methods for solving problem in Eq. (3.1) have been a focus of research interest since CS was first introduced. Proposed methods include: interior point methods [19], [22], projections onto convex sets [23], iterative soft thresholding [24–26], iteratively reweighted least squares [15], [27], and non-linear conjugate gradients and backtracking line-search [5], [14], [16], [28].



(a)



(b)



(c)

Figure 3.4: (a) A sparse signal. (b) Reconstruction using l_1 minimization. (c) Reconstruction using l_2 minimization [9].

Chapter 4 : BeagleBoard (Embedded Platform)

Embedded computing is a rapidly growing field. This field has exploded with the wide adoption of smartphones and most recently, the creation of multimedia devices. For developers interested in learning more about embedded computing or working to design a new embedded device, finding cost effective hardware on which to experiment can be a challenge. The BeagleBoard is one answer to this challenge [29].

This Chapter gives a detailed description for the BeagleBoard which is used the embedded platform for our experiment.

4.1. System Overview

The BeagleBoard is a low cost USB powered fanless computer. It is based on the OMAP35xx architecture, uses Texas Instruments ARM-8 and designed specifically to address the open source community Figure 4.1 [30].

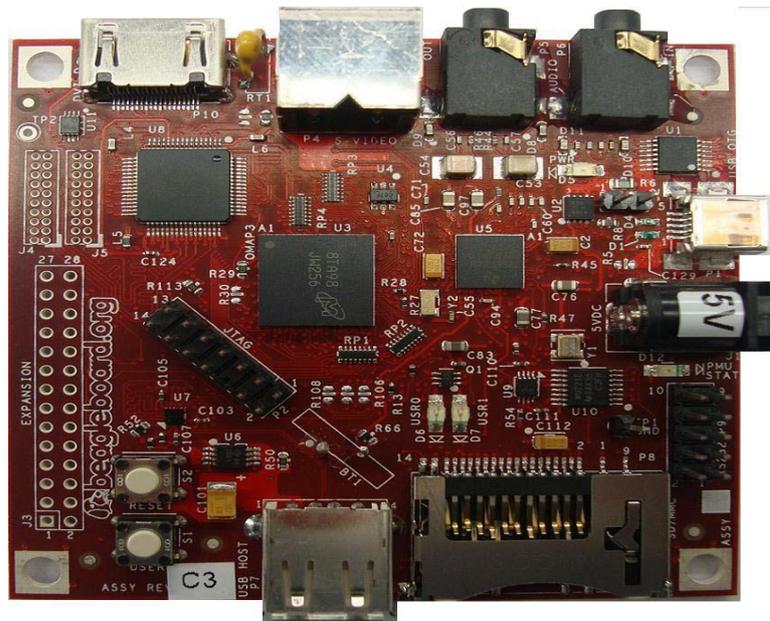


Figure 4.1: BeagleBoard [30].

The device can be connected to the USB port of a PC or laptop for experimentation. One great feature of the Beagle Board is that its capabilities can be expanded by the addition of various peripherals. These expansion capabilities include support for stereo audio, an interface for SD memory cards, the ability to be powered via USB style cell phone chargers and power supplies, DVI-D for connection to computer monitors, and different input devices like keyboards and pointing devices [29], [30]. There are certified and third party peripherals available including a 5V power supply and an Ethernet connection.

The device can be used for a variety of applications. Some of the ones mentioned on the beagleboard.org web site include multimedia player, game console, home automation, and kitchen computer.

In order to use the BeagleBoard, it should be loaded with an operating system which may be Linux or Windows based. Many software projects are available to create a version of Android for the BeagleBoard and OMAP3 platforms and also several Linux distributions being ported to the Beagle Board including Debian and Gentoo [29] (see Appendix A). All of this makes it not only a powerful learning tool but potentially a powerful and inexpensive prototyping tool as well.

The BeagleBoard is an exciting project that provides an extremely low-cost hardware solution for developers to learn about embedded computing. It can also potentially provide the perfect platform for prototyping the next generation of embedded devices.

4.2. Processor

The OMAP35x family of high-performance, applications processors are based on the enhanced OMAP™ 3 architecture and are integrated on TI's advanced 65-nm process technology. The architecture is designed to provide best-in-class video, image, and graphics processing sufficient to support video streaming, conferencing and gaming. The architecture of OMAP35xx is designed to provide maximum flexibility in a wide range of end applications including medical imaging. The device can support numerous HLOS and RTOS solutions including Linux and Windows Embedded CE Figure 4.2 [31].

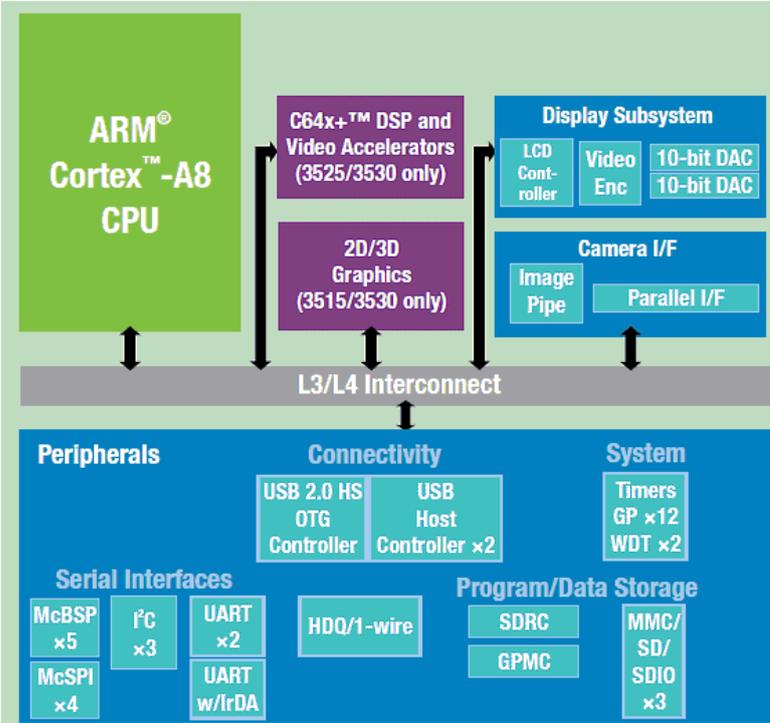


Figure 4.2: OMAP35xx Block Diagram.

This OMAP device includes state-of-the-art power-management techniques required for high-performance mobile products.

The following subsystems are part of the device:

- Microprocessor unit (MPU) subsystem based on the ARM® Cortex™-A8 microprocessor.
- IVA2.2 subsystem with a C64x+ digital signal processor (DSP) core.
- POWERVR SGX™ Graphics Accelerator subsystem for 3D graphics acceleration to support display and gaming effects.
- Camera image signal processor (ISP) that supports multiple formats and interfacing options connected to a wide variety of image sensors.
- Display subsystem with a wide variety of features for multiple concurrent image manipulation, and a programmable interface supporting a wide variety of displays. The display subsystem also supports NTSC/PAL video out.
- Level 3 (L3) and level 4 (L4) interconnects that provide high-bandwidth data transfers for multiple initiators to the internal and external memory controllers and to on-chip peripherals.

The device also offers:

- A comprehensive power and clock-management scheme that enables high-performance, low-power operation, and ultralow-power standby features. The device also supports SmartReflex™ adaptative voltage control. This power management technique for automatic control of the operating voltage of a module reduces the active power consumption.
- A memory stacking feature using the package-on-package (POP) implementation.

4.3. Usage Scenarios

When loading BeagleBoard with an operating system the ARM processor will work as the GPP, and to allow passing data and messages from the GPP to DSP, an inter-communication system will be built for the two sides. The CS based MRI reconstruction system will be implemented to run on the BeagleBoard in two modes as follows:

- Complete processing on the GPP (ARM).
- Hybrid processing on both sides of the processor (GPP and DSP).

Chapter 5 : Performance Enhancement of CS Algorithm

5.1. CS Algorithm Architecture

The problem in Eq. (3.1) represents a convex optimization problem. Finding a solution to this equation requires a highly efficient optimization method due to the large size of the parameter space. A suitable approach for such problems is the conjugate gradient method. It has initially been presented by Hestenes and Stiefel in 1952 for the solution of linear systems and in the meantime successfully applied to MRI reconstruction problems [32]. The method has been extended to nonlinear optimization by Fletcher and Reeves in 1964 and since then a number of optimized nonlinear conjugate gradient approaches have been developed [33]. Recently, Hager and Zhang [34] presented a version with improved convergence properties, which will be appropriate to solve Eq. (3.1). In this work we used the nonlinear conjugate gradient and backtracking line search to solve this problem similar to [5], [14], [16], [28] as it is characterized with low memory requirements and strong convergence.

Considering the unconstrained problem in the so-called Lagrangian form:

$$\operatorname{argmin}_m \|F_u m - y\|_2^2 + \lambda \|\psi m\|_1, \quad (5.1)$$

Where λ is a regularization parameter that determines the trade-off between the data consistency and the sparsity [5]. λ can be selected so that the solution of Eq. (5.1) will be the same as that of Eq. (3.1). The conjugate gradient algorithm implemented is shown in Figure 5.1 and $f(m)$ is the objective function as defined in Eq. (5.1).

“mingrad” and “maxiter” are used as stopping criteria for the algorithm by gradient magnitude and number of iterations respectively, α and β are line search parameters and are arbitrary selected (defaults are $\alpha = 0.05$ and $\beta = 0.6$). γ is the conjugate gradient update parameter and it can be calculated through many methods [33] and the selected method was that used in the first conjugate gradient method proposed by Fletcher and Reeves [35] as shown in Eq. (5.2).

$$\gamma = \frac{\|\mathbf{g}_{k+1}\|_2^2}{\|\mathbf{g}_k\|_2^2}. \quad (5.2)$$

Matlab (The MathWorks, Inc., Natick, MA, USA), was used for the implementation of CS algorithm. After implementation the algorithm was tested on two types of data the first is the SheppLogan phantom by computing its k-space at the wanted locations only using its continuous Fourier formulas, the second type of data is real brain MR image data.

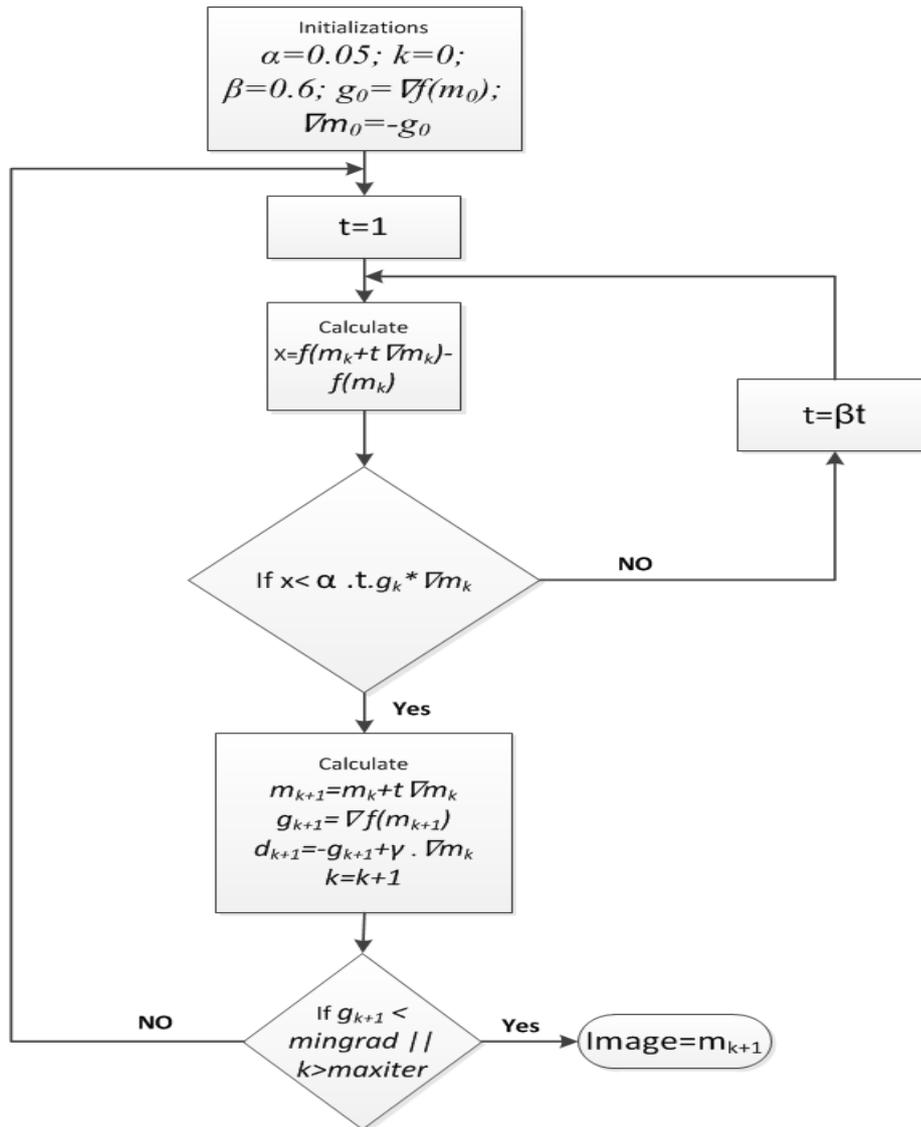


Figure 5.1: Conjugate gradient Algorithm block diagram.

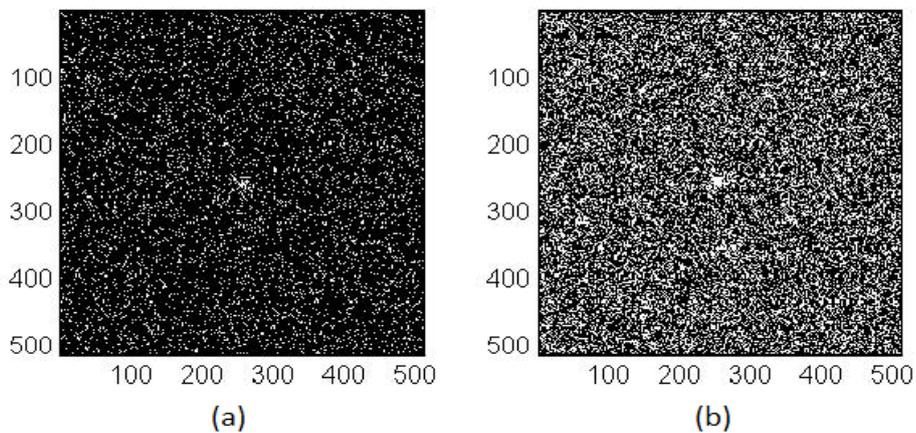


Figure 5.2: (a) 10% pattern. (b) 33.3% pattern.

In order to maximize the incoherence for a given number of samples, random sampling was chosen which results in a good, incoherent, and near-optimal solution [5]. A Monte-Carlo algorithm was used to generate the undersampling pattern which uses a grid size based on the desired resolution, and this grid is undersampled using a constructed probability density function and randomly draw indices from that density. The quality of the generated undersampling pattern is judged using the Transform Point Spread Function (TPSF) which is defined as $TSPF(i, j) = (\psi^* F_S^* F_S \psi)(i, j)$ [5], [6], and the pattern with the lowest peak interference was selected. An example of generated undersampling patterns is shown in Figure 5.2.

As mentioned before the constructed CS algorithm was tested on a Shepp Logan phantom data and on a real MR data. Firstly for the Shepp Logan trial, the continuous Fourier data was calculated at specific locations according to the generated undersampling patterns of ratios of (33.3%, 30%, 25%, 20%, 15%, and 10%) and with a desired resolution of 512*512 pixels and these data was prepared for testing in two modes the first was using it directly with the algorithm and the other was by adding a Gaussian noise ($\mu=0.002$ & $\sigma^2=0.002$) to the Fourier data (Signal to Noise Ratio: 2.3 dB) and then testing it with the algorithm. The sparsifying transform used for the Shepp Logan images was the Finite Difference transform. Secondly the Algorithm was tested on real MR data which was a brain MR image with a resolution 512*512 pixels with the same undersampling patterns used with the Shepp Logan trial but with Wavelet transform as the sparsifying transform.

5.2. Image Quality Evaluation Metrics

Here, the metrics used to evaluate the quality of the produced images by compressed sensing are mentioned.

5.2.1. Mean Squared Error (MSE)

The MSE [36], [37] has been widely used to quantify image quality. It measures the quality change between the original and processed images in an M*N window. When it is used alone, it does not correlate strongly enough with perceptual quality. It should be used, therefore, together with other quality metrics and visual perception. MSE is defined by Eq. (5.3).

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (g_{i,j} - f_{i,j})^2. \quad (5.3)$$

5.2.2. Geometric Average Error (GAE)

The value of GAE [37] is approaching zero if there is a very good transformation (small differences) between the original and processed images; otherwise, the value of GAE is high. GAE is defined by Eq. (5.4).

$$GAE = \left(\prod_{i=1}^M \prod_{j=1}^N \sqrt{g_{i,j} - f_{i,j}} \right)^{1/MN}. \quad (5.4)$$

5.2.3. Quality Index (QI)

It models any distortion as a combination of three different factors, which are loss of correlation, luminance distortion, and contrast distortion, and is defined as:

$$Q = \frac{\sigma_{gf}}{\sigma_f \sigma_g} \cdot \frac{2\bar{f}\bar{g}}{(\bar{f})^2 \cdot (\bar{g})^2} \cdot \frac{\sigma_f \sigma_g}{\sigma_f^2 + \sigma_g^2}, \quad (5.5)$$

Where \bar{g} and \bar{f} represent the mean of the original and processed values with their standard deviations σ_g and σ_f of the original and processed values of the analysis window, and σ_{gf} represents the covariance between the original and processed windows. QI is computed for a sliding window of size 8*8 without overlapping. Its highest value is 1 if $g_{i,j} = f_{i,j}$, whereas its lowest value is -1 if $f_{i,j} = 2\bar{g} - g_{i,j}$ [37].

5.3. CS Algorithm Performance

The Algorithm was tested during all trials on a PC containing 8 GB RAM and an Intel® Core™ i7-2630QM CPU 2.00GHz (Intel Corporation, USA). Figure 5.3 shows the results of reconstruction for Shepp Logan for patterns of ratios (33.3%, 30%, 25%, 20%, 15%, and 10%) from the top of figure. Figure 5.4 shows the results for the reconstruction of the noised Shepp Logan with the same ratios used with the clear Shepp Logan. Figure 5.5 shows the results for the brain MR image with ratios of 33.3% to 10% from top of figure. From these results we can see that the quality of images reconstructed from incomplete k-space is very good compared to that produced from the complete k-space. The quality of produced images using CS is evaluated using the metrics mentioned in the previous section. Figure 5.6 shows in the left column the mean squared error drawn versus the undersampling ratios and in the right column the quality index versus undersampling ratios for SheppLogan, noised SheppLogan, and brain MR image from top to down respectively. The values of MSE and QI are shown in Table 5.3 to Table 5.8 at the end of chapter. From the calculated metrics it seems that the reconstructed images using CS for Shepp Logan phantom is with medium quality as it has a slightly high MSE and medium QI (near zero) which needs to have a value near one for good approximation of image. The algorithm shows approximately the same behavior with the noised Shepp Logan as in Figure 5.4 and Figure 5.6 from the side of quality metrics except that the produced images may be visually better than the noised images reconstructed from the complete k-space. The results for the real brain MR image show a very good performance as the reconstructed images using CS from small undersampling ratios have a very small MSE compared to the image produced from the complete k-space and the QI is very close to one which indicates that the reconstruction using CS is a very good approximation for the data.

5.4. CS using L0-Norm Minimization

In the objective function of the CS optimization problem in Eq. (3.1) and Eq. (5.1), we used the L1-norm minimization of the sparse representation of the reconstructed image. This part of the objective function is the part which enforces the sparsity of the produced image in the sparsifying transform domain. The L1-norm is calculated as mentioned before by summation of the absolute values of the array of interest and this cares about the values of coefficients of the image in the sparse domain, but if we can reduce the order of the norm in the objective function to care about the number of non-zero elements in the sparse representation, this will more express the sparsity of the vector of coefficients and minimizing this number will produce sparser images than those produced by minimizing L1-norm.

Reducing the order of the norm means that the L0-pseudo norm will be used which means the number of non-zero elements of the array of interest and the new optimization problem will be as follows:

$$\text{minimize } \|\psi m\|_0 \quad \text{subject to } \|\mathbb{F}_S m - y\|_2 < \varepsilon. \quad (5.6)$$

Solving Eq. (5.6) which is a non-convex optimization problem is generally infeasible [38]. Replacing the L0-pseudo norm by L1-norm as shown before is one of the solutions to this difficulty. A smoothed representation of the L0-norm by approximation using continuous function may enhance the performance of the CS algorithm and make it computationally inexpensive [38].

The used smoothed L0-norm is computed as follows:

Consider

$$f(x) = \begin{cases} 1, & |x| \ll \sigma \\ 0, & |x| \gg \sigma \end{cases}, \quad (5.7)$$

Define the continuous multivariate function $g(x)$ as:

$$g(x) = \sum_{i=1}^N f(x_i), \quad (5.8)$$

L0-norm is calculated through

$$\|x\|_0 = N - g(x). \quad (5.9)$$

Using the new definition of the L0-norm in Eq. (5.9), the CS algorithm was implemented by solving the optimization problem in Eq. (5.6) using the same techniques used with the solution of problem in Eq. (3.1) (nonlinear Conjugate Gradient). Also the algorithm was tested on the same data for the algorithm of Eq. (3.1) (Shepp Logan, Noised Shepp Logan, and real MR data).

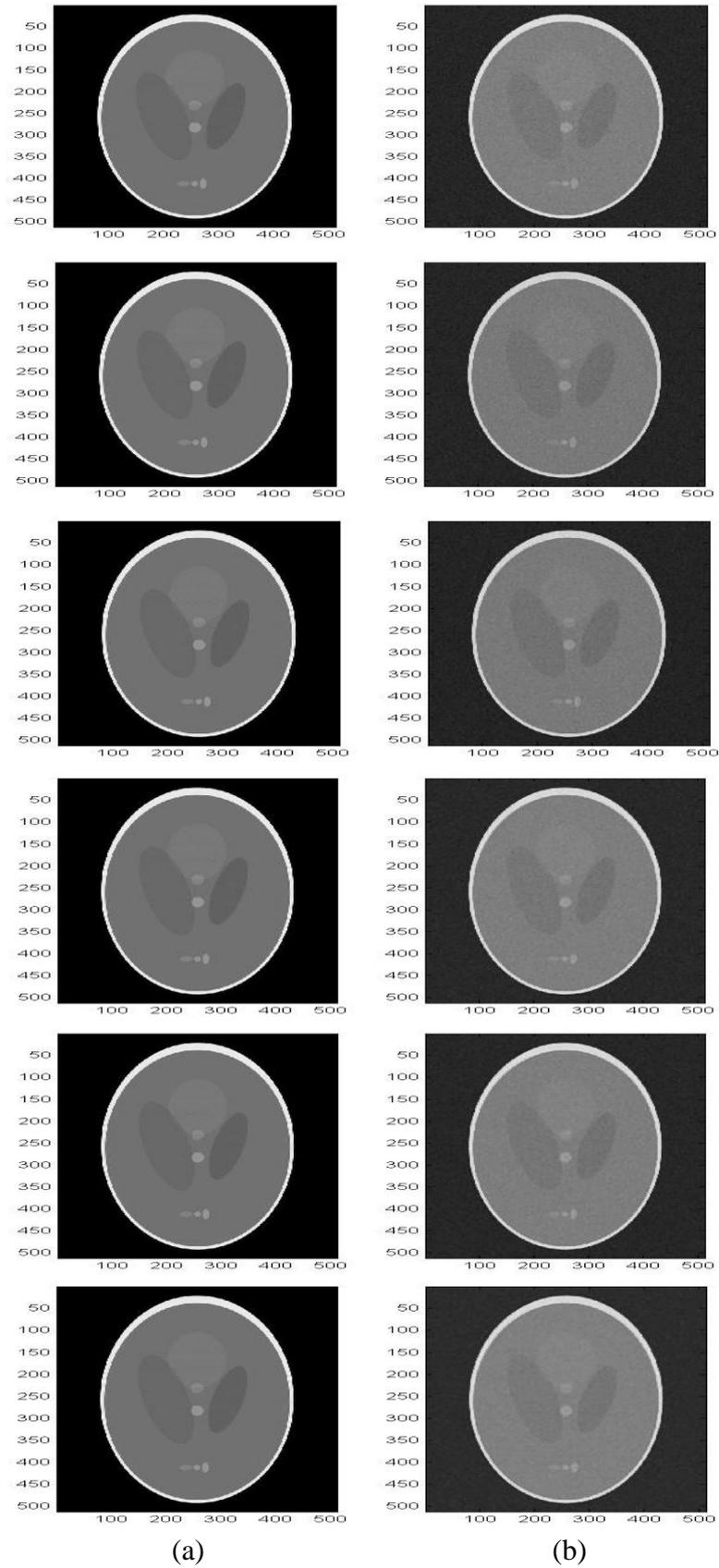


Figure 5.3: (a) Shepp logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.

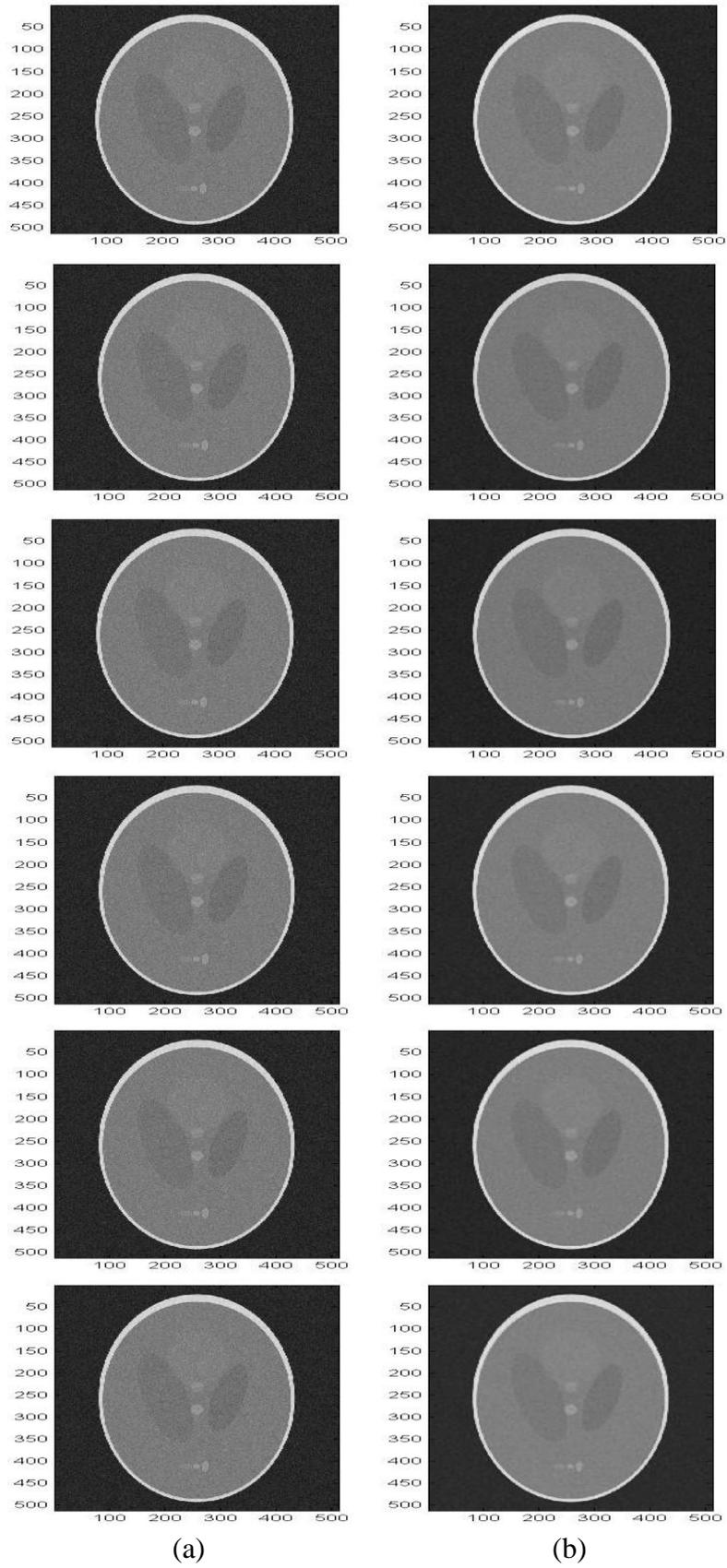


Figure 5.4: (a) Noised Shepp logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.

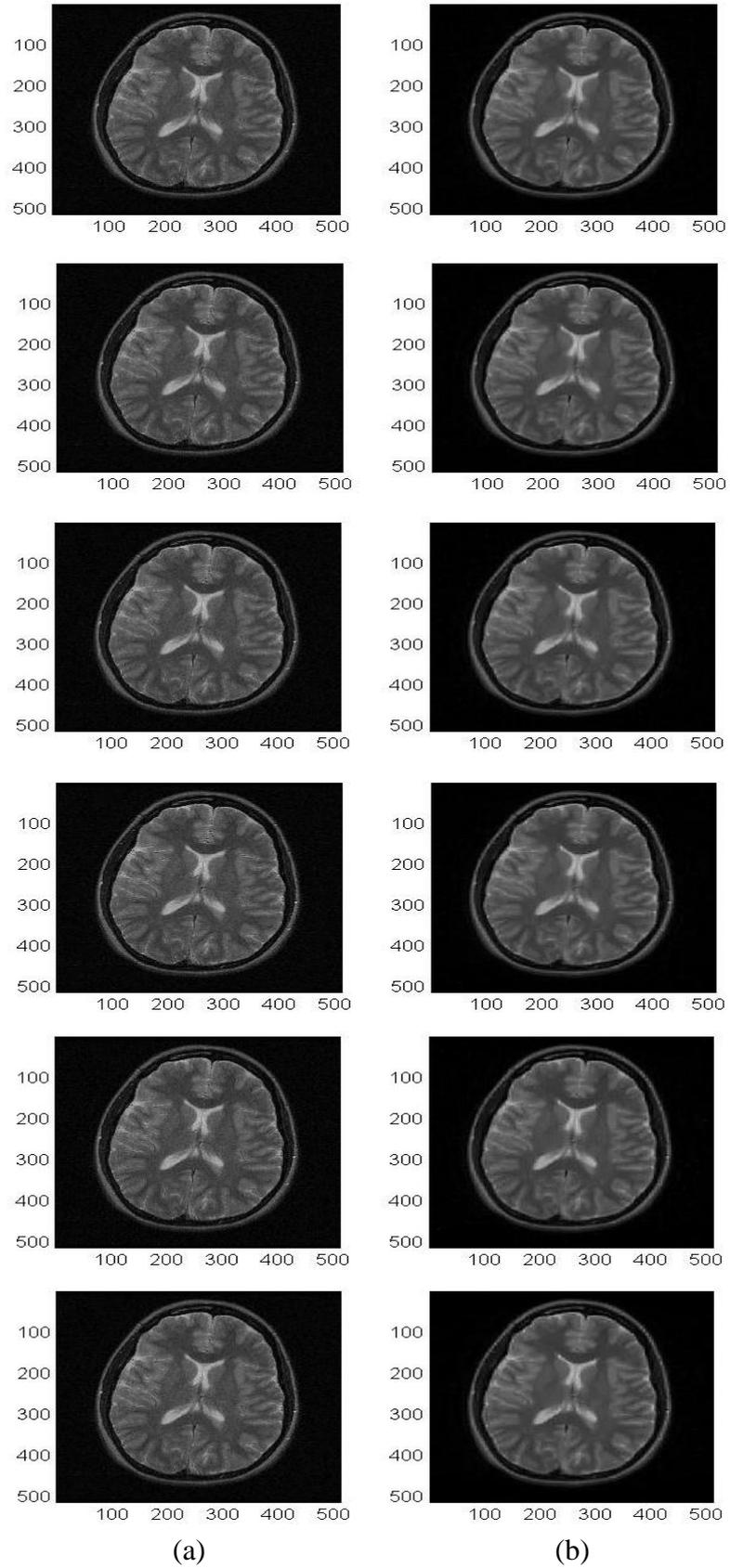
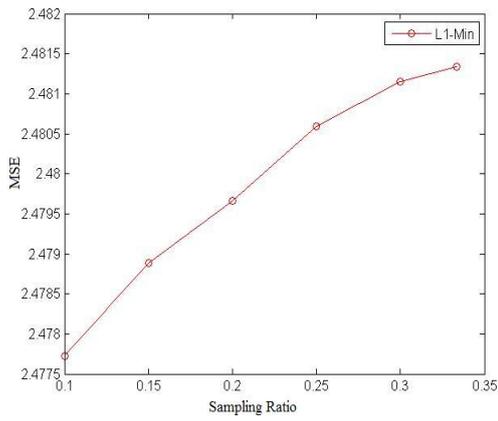
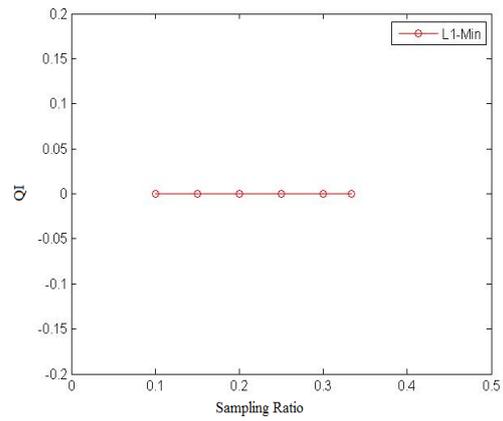


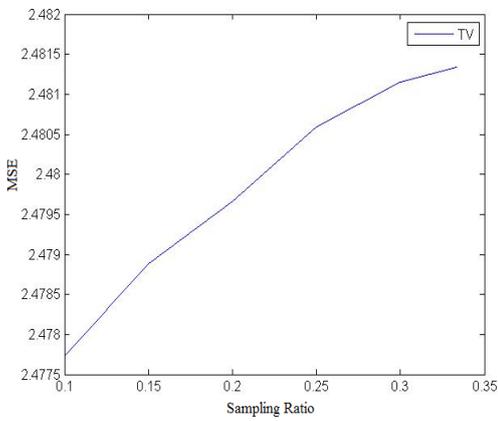
Figure 5.5: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using CS with undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.



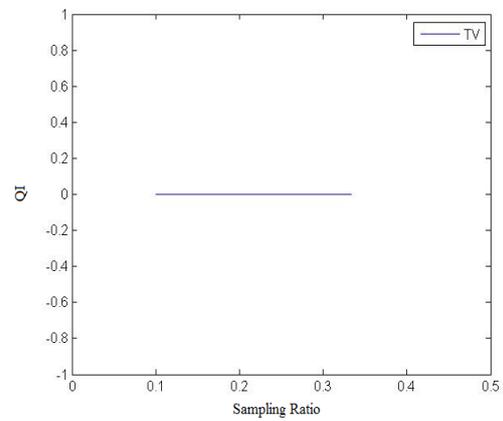
(a)



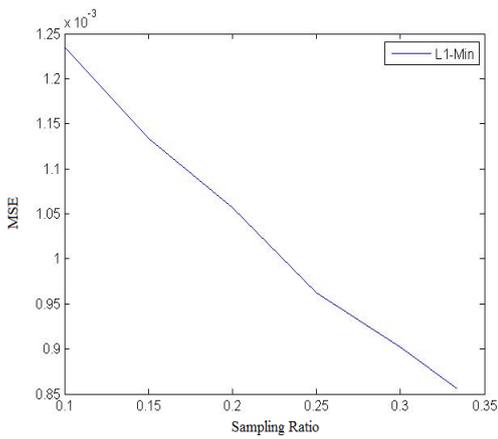
(b)



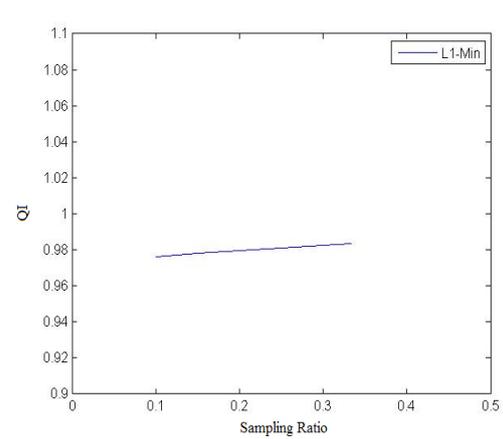
(c)



(d)



(e)



(f)

Figure 5.6: MSE & QI calculated for CS reconstructions compared to full k-space reconstruction for the three data types. (a) & (b) MSE and QI of Shepp logan. (c) & (d) MSE and QI of Noised Shepp Logan. (e) & (f) MSE and QI of Brain Image.

The undersampling patterns used here are generated using the same Monte-Carlo method, but we used a modification on those ones used with the real MR data where we concentrated most of the samples in the central region as in Figure 5.7 taking in consideration that the MR k-space has most of its power in this region.

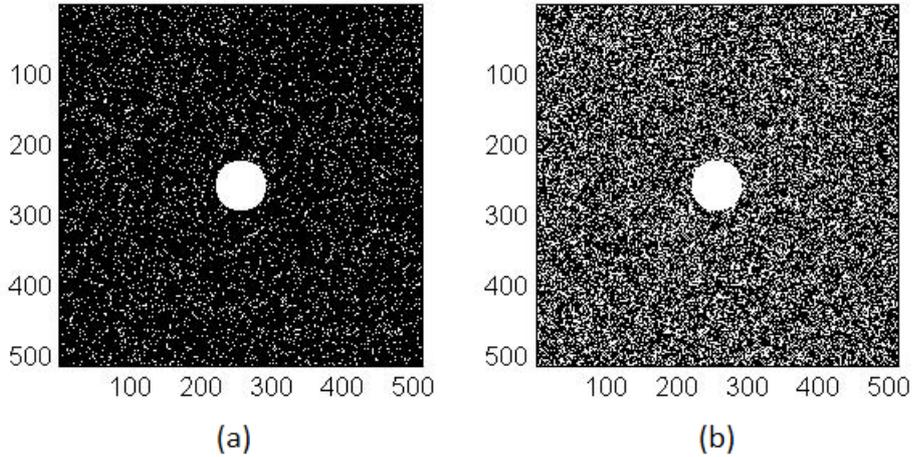


Figure 5.7: (a) 10% undersampling pattern. (b) 33.3% undersampling pattern.

The algorithm here also was implemented and tested on the same platform used for the original CS algorithm. Figure 5.8 shows the results of using L0-norm with the finite difference transform on the Shepp Logan for undersampling ratios of (33.3%, 30%, 25%, 20%, 15%, and 10%) from the top of figure and as we see, the produced images seem to be with good quality compared the image produced from complete k-space reconstruction. Figure 5.9 shows the results of using L0-norm with the finite difference transform on the noised Shepp Logan for the same undersampling ratios. Figure 5.10 shows the results of using L0-norm with the Wavelet transform on the real MR image. Figure 5.11 shows quality comparison for using the L0-norm and L1-norm in reconstruction including the mean squared error in the left column and the quality index in the right column.

5.5. CS using Fourier Transform as the Sparse transform for MR Real Data

In this section we propose the use of the Fourier transform of the image of interest as the sparse transform, testing it with the use of both L1-norm and smoothed L0-norm minimization, and comparing it with the results of the same trials using the Wavelet transform as the sparse transform. The optimization problem of CS when using Fourier transform as the sparse transform will be as follows:

$$\text{minimize } \|F \cdot m\|_1 \quad \text{subject to } \|F_s m - y\|_2 < \varepsilon. \quad (5.10)$$

Where F indicates the complete forward Fourier transform. The algorithm of this technique was tested using the MR real data of the same resolution used in the previous trials, and tested on the same computing platform used for the original CS algorithm. Figure 5.12 shows the results of using the Fourier transform as the sparse transform

with L1-norm minimization. Figure 5.13 shows the results of using the Fourier transform as the sparse transform with L0-norm minimization. Figure 5.14 is quality comparison between using L1 and L0-norms with the Fourier transform as the sparse transform, using Wavelet and Fourier with L1-norm, and using Wavelet and Fourier with L0-norm. The left column shows the mean squared error and the right shows the quality index. Figure 5.15 shows performance comparison (Processing time) for the different versions of the CS algorithm.

Table 5.1 to Table 5.11 contain all the data represented in the comparison figures including mean squared errors and quality index for both real brain MR and simulated data (clear and noised Shepp Logan), signal to noise ratio for noised Shepp Logan results, and processing time of all trials.

5.6. Discussion

The basic CS algorithm gave excellent results compared to images reconstructed from complete k-space in both simulated data (Shepp Logan phantom) in Figure 5.3 and Figure 5.4, and real brain MR data in Figure 5.5. The quality of reconstructed images is good according to the quality metrics measured in Figure 5.6 which show good mean squared errors and quality indices with the best performance with the real MR image. We can see that the MSE for the noised Shepp Logan gets bad as the undersampling ratio increases and the reason for this may be that increasing the sampling ratio here is for both the signal and the noise which results in acquiring more information about the sampled signal (the noised image) and as a result good recovery for the sampled signal which is an extra noised image with respect to the original Shepp Logan image (as MSE and QI are calculated with respect to the clear image). Also we see that the quality indices for clear and noised Shepp Logan are near zero which means a medium quality of the produced images, but for the real MR data it is near one which means that we have a good reconstruction from the side of correlation, luminance, and contrast distortion.

Using the L0-norm penalized reconstruction gave the expected performance for both the simulated data and the real MR data as shown in Figure 5.8 for the clear Shepp Logan image, and Figure 5.10 for the real MR data. When comparing the MSE and the quality index for the reconstructed images using L0-norm based CS compared to image produced from complete k-space in Figure 5.11, we can see that the L0-norm penalization produces images of lower mean squared errors than those produced with L1-norm penalized reconstruction except for noised Shepp Logan images it gives higher mean squared errors for the same reason mentioned in the last paragraph. Also we find here that the quality of the produced images using L0-norm based CS from the side of contrast and luminance (QI) is slightly higher than that images produced using the L1-norm based CS. And as shown in Figure 5.15 (a) we can say that using L0-norm with wavelet as a sparse transform in CS algorithm has no benefit from the side of computation time as it has approximately the same computation time of using L1-norm and this is considered a good thing as using L0-norm now has become computationally inexpensive or at least comparable with using L1-norm.

Figure 5.12 and Figure 5.13 show the results of using Fourier transform as the sparse transform with the real MR image, the results show that using Fourier with L1-norm penalized reconstruction gave better results than using Wavelet with the same type of reconstruction and approximately the same quality of images if using it with L0-norm penalized reconstruction. This is verified in quality comparison figures in

Figure 5.14 which show that using Fourier based CS with L1-norm gives a smaller mean squared error than Wavelet and approximately the same mean error with L0-norm. With respect to the quality indices for both transforms we see that they are excellent also and all of them are near one for both transforms (slightly higher for using Fourier based CS). Figure 5.15 shows the computation time for the different trials with the real MR image and we can see approximately all the trials have the same computation time except for the L1-norm penalized CS based on Fourier reconstruction which gives the best performance among all trials.

Through previous results we can say that CS algorithm is a good reconstruction technique but the quality of images should be investigated in other ways to be sure of the efficiency of the algorithm in full recovery of the image. One way may be to investigate the results of the algorithm in reconstruction of diseased brain images and see the effect of incomplete sampling on appearance of images or to find another quality factor which better describes the reconstructed images using CS. Also other ways need to be investigated to reduce the computation time of CS algorithm like reducing the number of iterations of the steepest descent CG by improving the stopping criterion used in the algorithm.

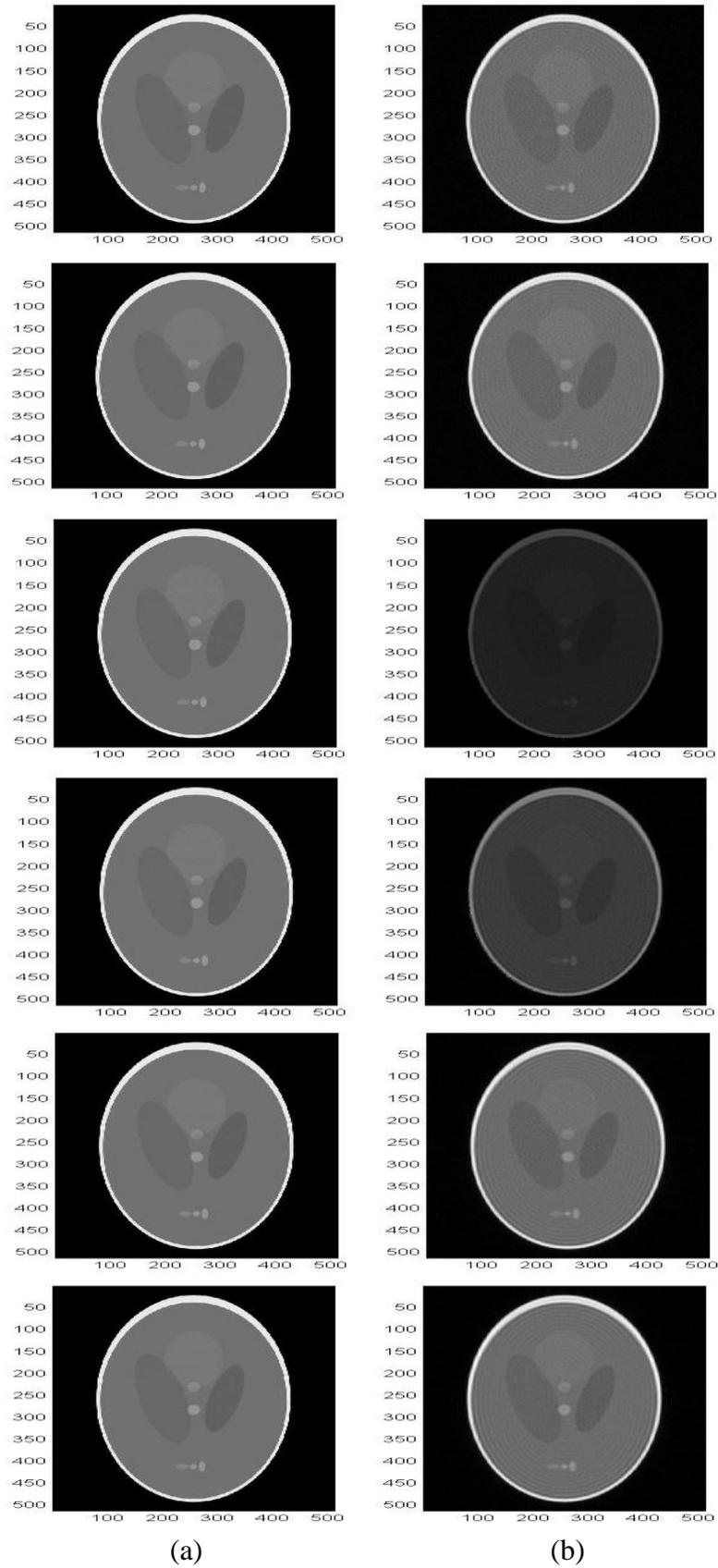


Figure 5.8: (a) Shepp Logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.

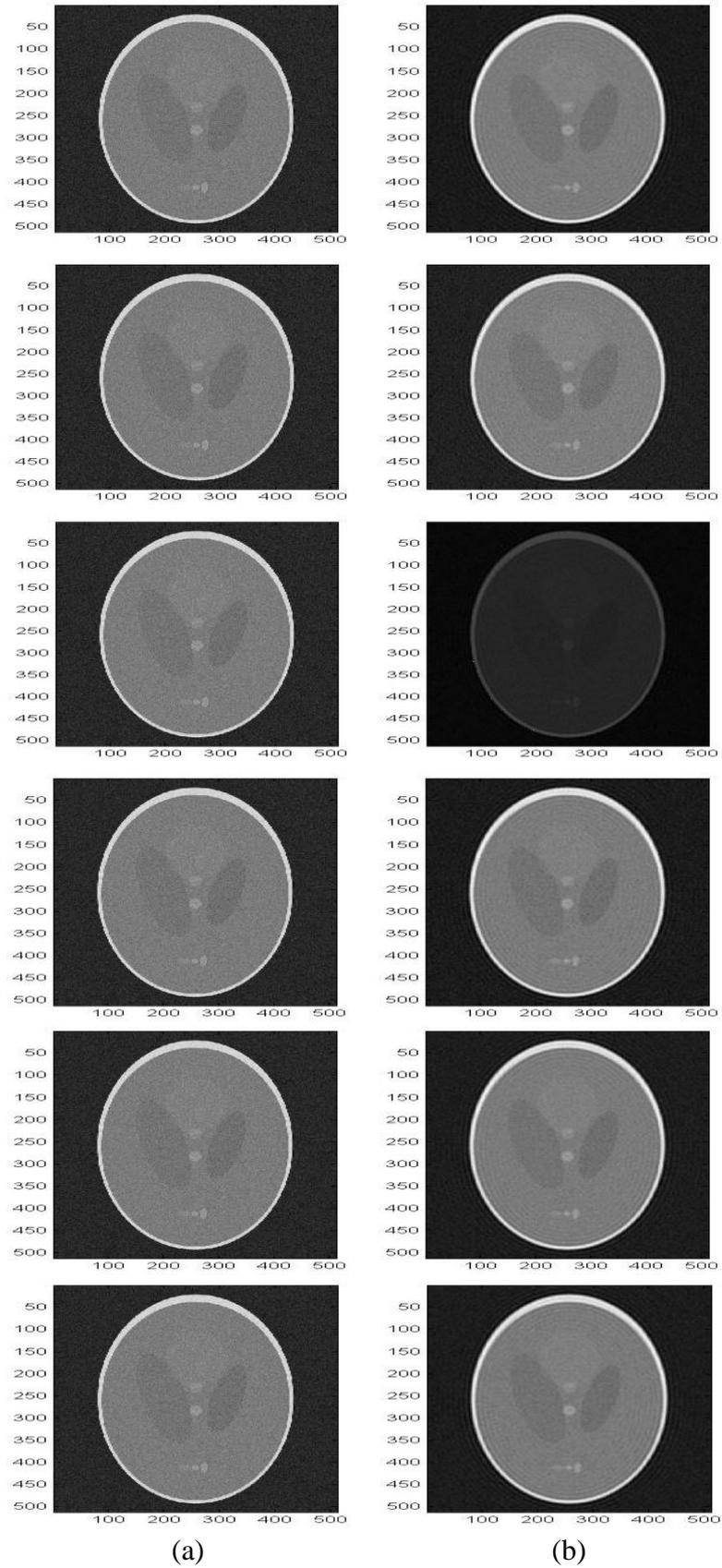


Figure 5.9: (a) Noised Shepp Logan image reconstructed from 100% of k-space. (b) Reconstruction using CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.

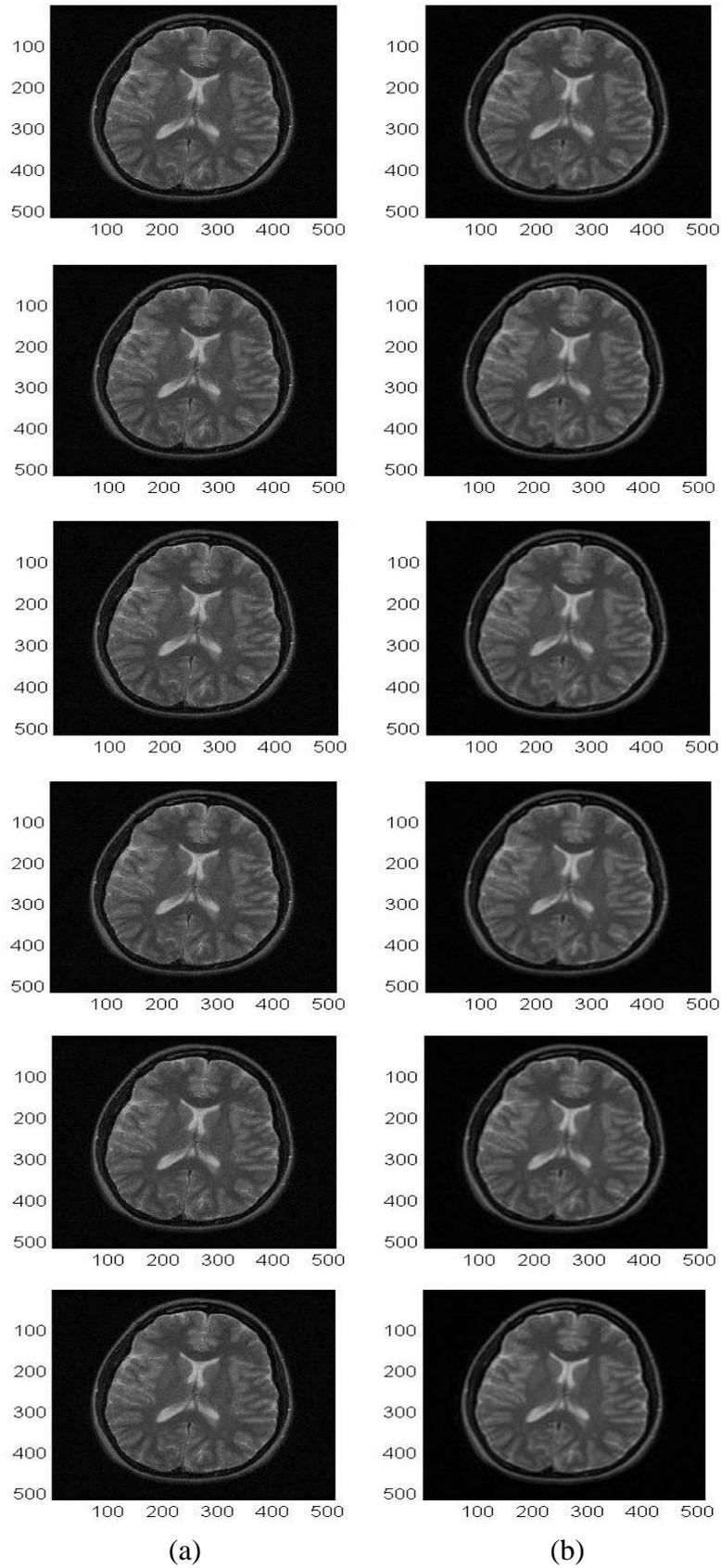


Figure 5.10: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.

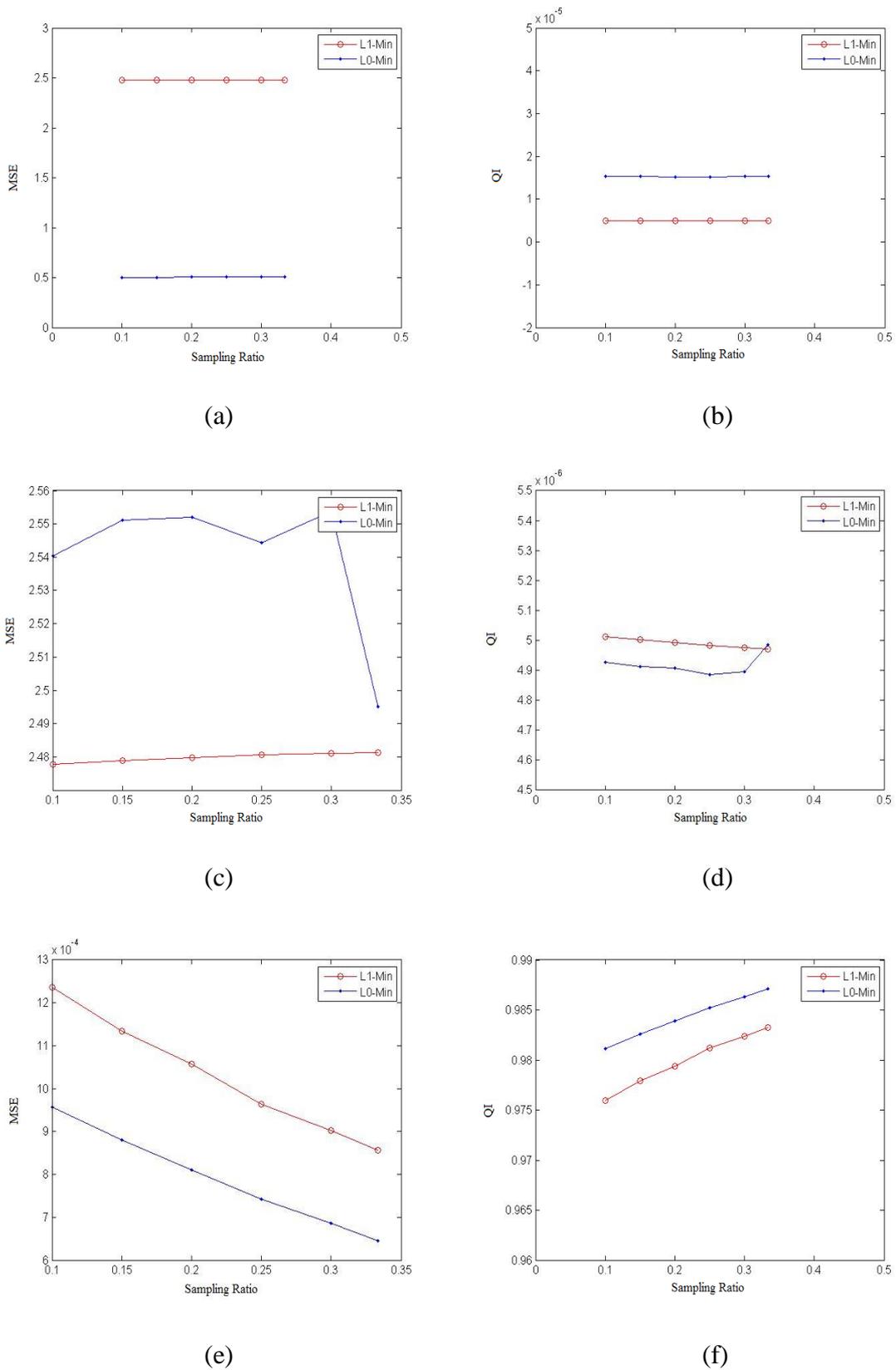


Figure 5.11: Quality comparison between using CS with L1-Norm & L0-Norm, (a) & (b) MSE and QI of Shepp Logan results. (c) & (d) MSE and QI of Noised Shepp Logan results. (e) & (f) MSE and QI of Brain image results.

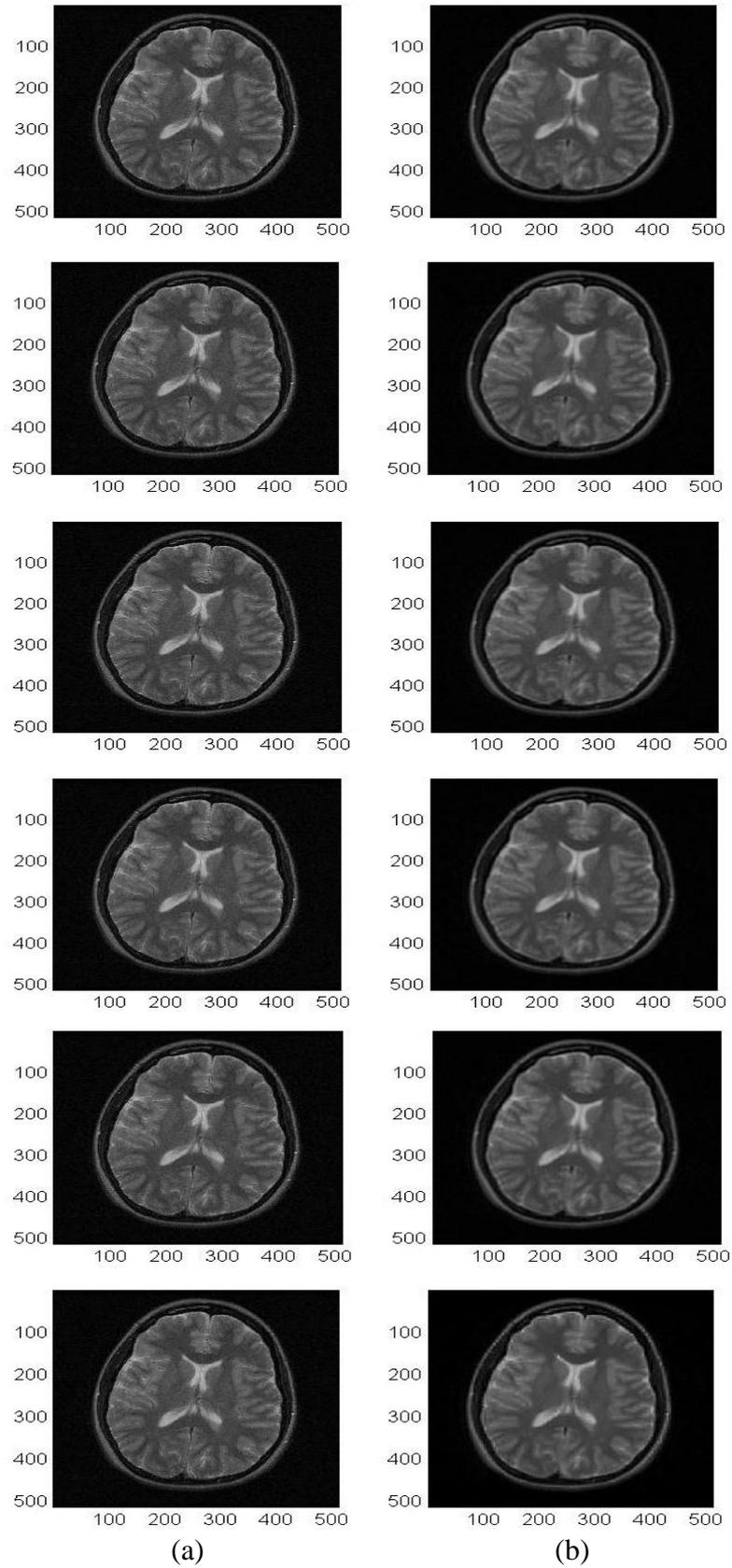


Figure 5.12: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using CS with Fourier transform and L1-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.

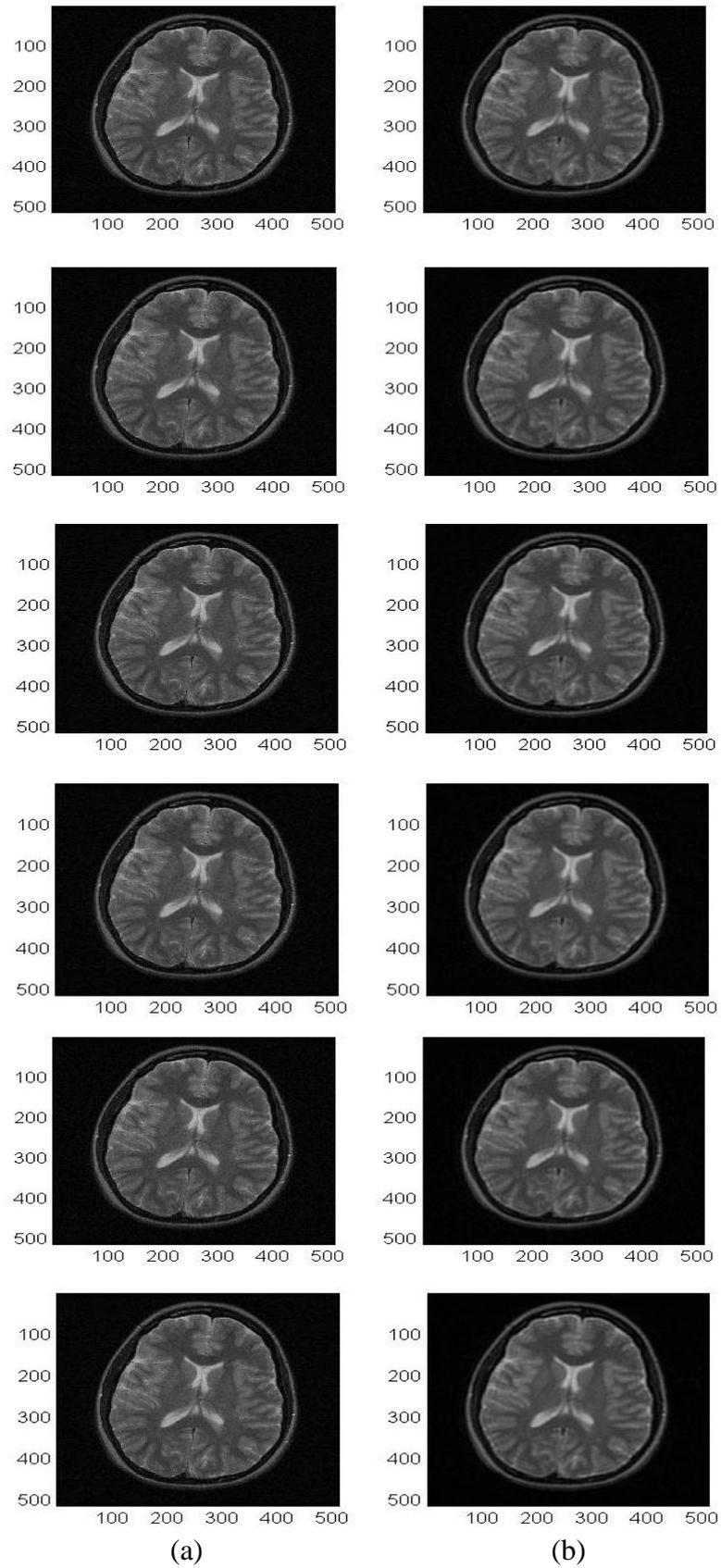
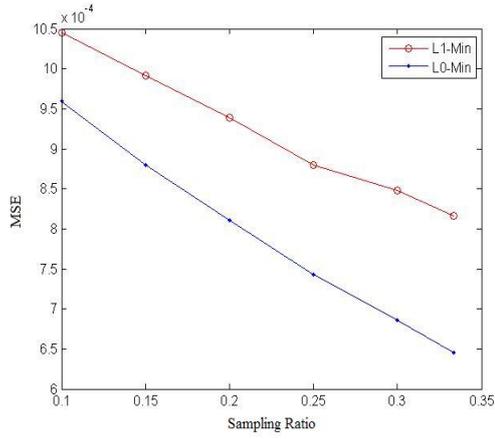
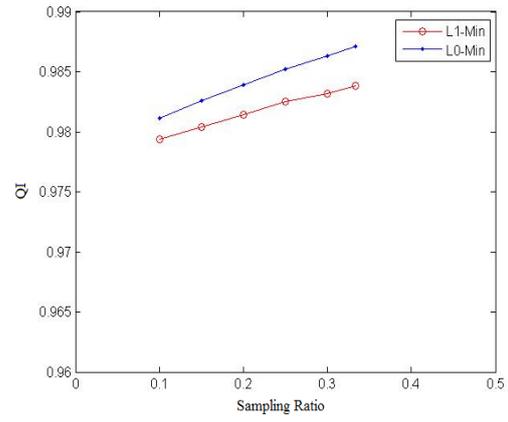


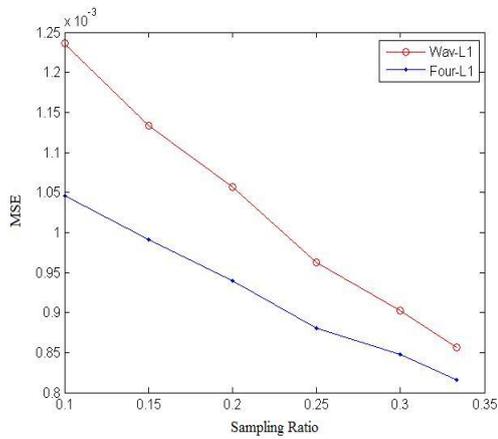
Figure 5.13: (a) Brain image reconstructed from 100% of k-space. (b) Reconstruction using Fourier based CS with L0-Norm for undersampling ratios of 33.3, 30, 25, 20, 15, 10% of k-space respectively from top to bottom.



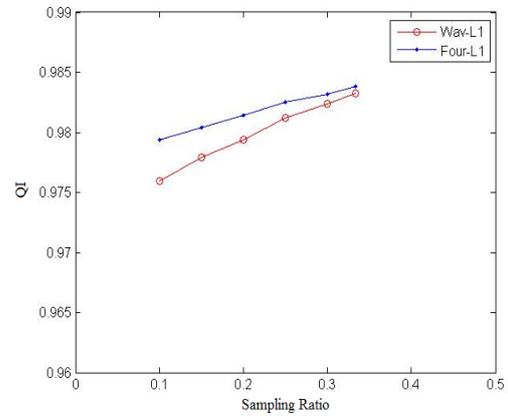
(a)



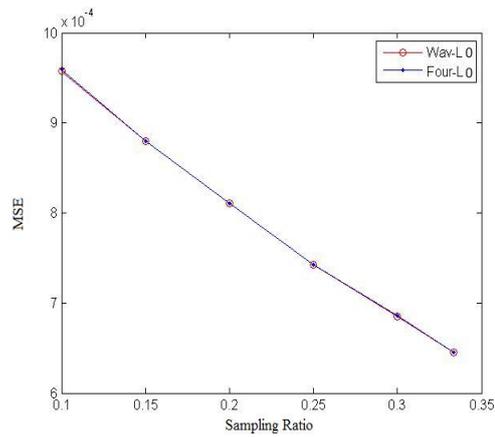
(b)



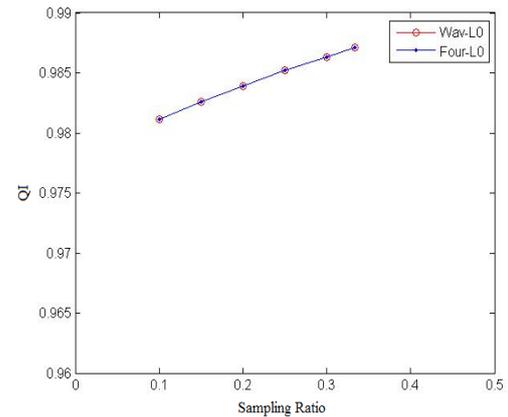
(c)



(d)



(e)



(f)

Figure 5.14: Quality Comparison, (a) & (b) MSE & QI for brain image reconstructions using Fourier based CS with L0-Norm & L1-Norm. (c) & (d) MSE & QI for Fourier based CS vs. Wavelet based CS with L1-Norm. (e) & (f) MSE & QI for Fourier based CS vs. Wavelet based CS with L0-Norm.

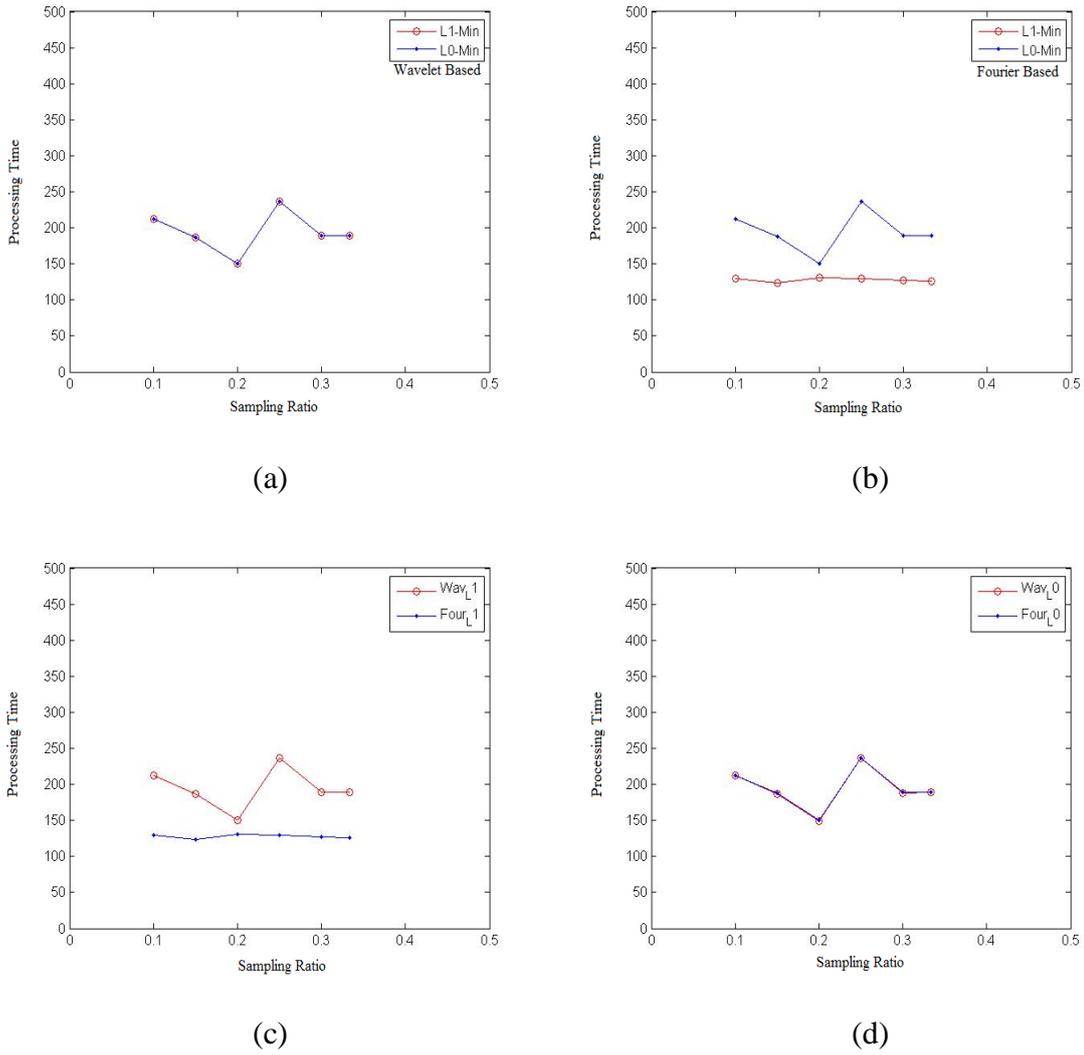


Figure 5.15: Performance comparison (a) Processing time for reconstructions using Wavelet based CS with L0-Norm & L1-Norm. (b) Processing time for using Fourier based CS with L0-Norm & L1-Norm. (c) Processing time for using L1-Norm with Wavelet and Fourier. (d) Processing time for using L0-Norm with Wavelet and Fourier.

Table 5.1: Real MR image MSE with Fourier as a sparse transform

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	8.16e-4	8.47e-4	8.8e-4	9.39e-4	9.9e-4	0.001
L0-Norm	6.45e-4	6.85e-4	7.42e-4	8.1e-4	8.79e-4	9.59e-4

Table 5.2: Real MR image QI with Fourier as a sparse transform

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	0.9837	0.9831	0.9825	0.9814	0.9804	0.9793
L0-Norm	0.9870	0.9862	0.9852	0.9839	0.9825	0.9811

Table 5.3: Real MR image MSE with Wavelet as a sparse transform

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	8.56e-4	9.02e-4	9.62e-4	10.5e-4	11.33e-4	12.35e-4
L0-Norm	6.45e-4	6.85e-4	7.42e-4	8.1e-4	8.79e-4	9.57e-4

Table 5.4: Real MR image QI with Wavelet as a sparse transform

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	0.9833	0.9823	0.9812	0.9793	0.9778	0.9759
L0-Norm	0.9870	0.9863	0.9852	0.9839	0.9825	0.9811

Table 5.5: Clear Shepp Logan MSE

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	2.4813	2.4811	2.4805	2.4796	2.4788	2.4777
L0-Norm	0.5046	0.5041	0.5056	0.5063	0.5033	0.5030

Table 5.6: Clear Shepp Logan QI

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	4.97e-6	4.97e-6	4.98e-6	4.99e-6	5e-6	5.01e-6
L0-Norm	1.53e-5	1.53e-5	1.52e-5	1.51e-5	1.52e-5	1.52e-5

Table 5.7: Noised Shepp Logan MSE

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	2.4813	2.4811	2.4805	2.4796	2.4788	2.4777
L0-Norm	2.4950	2.5537	2.5444	2.5520	2.5510	2.5403

Table 5.8: Noised Shepp Logan QI

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	4.97e-6	4.97e-6	4.98e-6	4.99e-6	5e-6	5.01e-6
L0-Norm	4.98e-6	4.89e-6	4.88e-6	4.9e-6	4.91e-6	4.91e-6

Table 5.9: Noised Shepp Logan SNR

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	0.00427	0.00427	0.00427	0.00426	0.00426	0.00426
L0-Norm	0.00417	0.00409	0.00410	0.00408	0.00408	0.00409

Table 5.10: Real MR image reconstruction time with Fourier as sparse transform

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	126.41	126.68	130.05	131.34	124.01	129.74
L0-Norm	188.99	189.20	237.00	150.50	187.94	212.77

Table 5.11: Real MR image reconstruction time with Wavelet as sparse transform

Undersampling Ratio	33%	30%	25%	20%	15%	10%
L1-Norm	189.03	188.99	236.57	150.00	187.14	211.83
L0-Norm	189.61	188.40	236.42	149.62	186.97	212.28

Chapter 6 : Embedded Implementation

In this chapter we investigate the use of an embedded platform based on the OMAP processor (BeagleBoard) for a challenging image reconstruction algorithm for MRI based on the compressed sensing. We compare straightforward implementations of the compressed sensing reconstruction algorithm on different processing platforms including embedded processors to verify the performance of such platforms. The performance of the algorithm on the embedded platform was compared to the performance on two large processing platforms containing Intel® Core™ Duo Processor and Intel® Core™ *i7-2630QM* CPU 2.00GHz (Intel Corporation, USA).

6.1. CS Algorithm Preparation

The CS algorithm to be tested on the embedded platform was the initial CS algorithm using the L1-norm minimization in the solution of the optimization problem in Eq. (3.1) and implemented using the standard *C* language and *gcc* 4.6.3 (Free Software Foundation, Inc., Boston, USA) on *Ubuntu* 11.10 (Canonical Ltd., London, United Kingdom) using a platform containing an Intel® Core™ Duo Processor *T2450* 2.00 GHz (Intel Corporation, USA). The memory used by the program was optimized and reduced in order to be suitable for the limited memory of BeagleBoard.

Due to memory considerations we used an angiography-like simulated image with a size of 100×100 pixels and containing randomly generated vessels with different sizes and magnitudes as shown in Figure 6.1 (a). The k-space of the image was undersampled with a factor of 20 with a randomly generated sampling pattern shown in Figure 6.1 (b), and as an initial guess for the algorithm we used a zero filling with density compensation (ZF-w/dc) reconstructed image Figure 6.1 (c). ZF-w/dc is the reconstruction by zero-filling the missing k-space data and k-space density compensation [5]. The algorithm block diagram is shown in Figure 6.2.

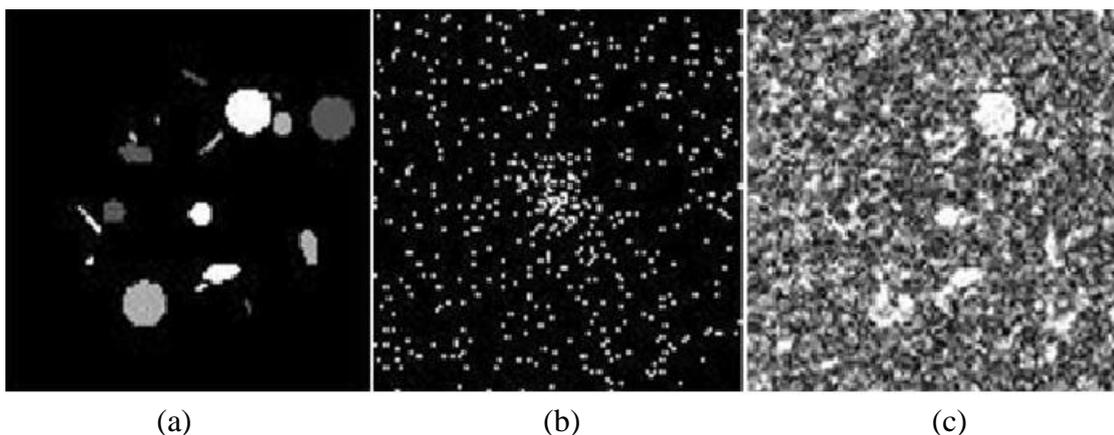


Figure 6.1: (a) Original image. (b) Sampling pattern. (c) Zero filling with density compensation reconstruction.

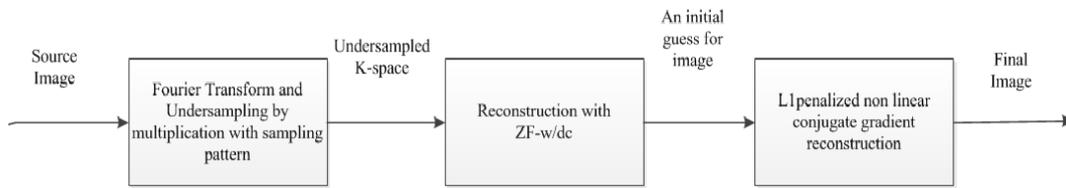


Figure 6.2: CS algorithm block diagram

6.2. Embedded Reconstruction

In order to run the OMAP we need first to load it with an embedded operating system (Windows based or Linux based), we build Ångström system which is a complete Linux distribution and includes the kernel, a base file system, basic tools and a package manager to install software from a repository. It uses the Open Embedded (OE) platform, a tool-chain that makes cross-compiling and deploying packages easy for embedded platforms, also an inter-processor communication system (DSP/BIOS Link) Fig.3, was built between the two processors (ARM and DSP) to allow passing messages and data for testing algorithm from the ARM (that works as a general purpose processor GPP) side to the DSP side to perform it Figure 6.3 [8].

The algorithm was tested on the embedded platform in two modes the first was on the GPP (ARM) and the second was implemented using the two processors of the board (ARM & DSP) [8].

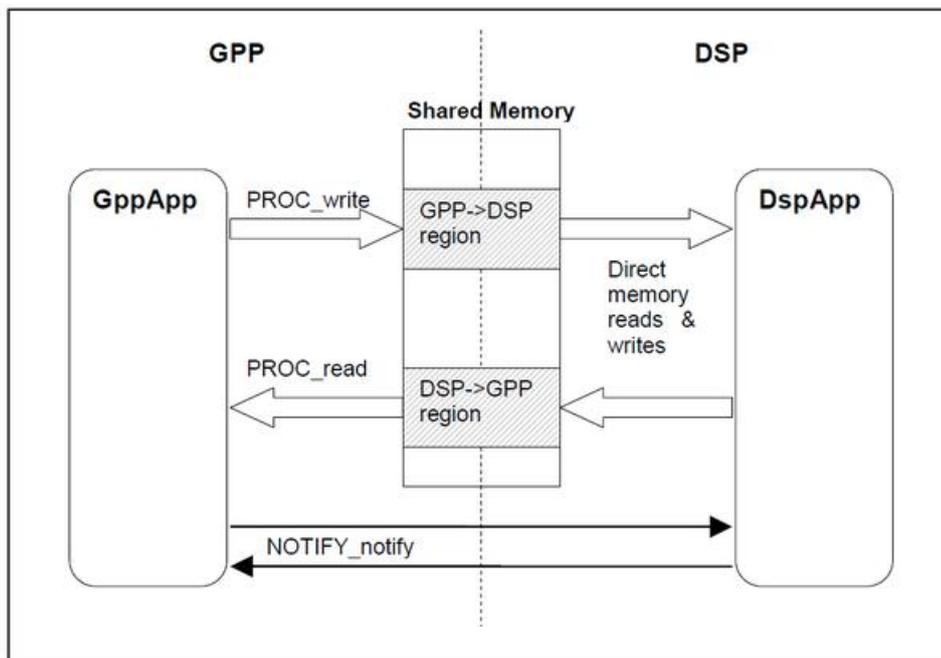


Figure 6.3: Communication with DSP.

6.2.1. Arm Based Reconstruction System

Here the CS algorithm was tested only on the ARM processor and compiled using *EGLIBC* 2.16 (Linux Foundation, USA). All calculations were performed in a straight forward sequential manner using the GPP. The algorithm test data including the image file, the undersampling pattern, and the probability density function for the density compensated reconstruction was transferred to the BeagleBoard though Ethernet [8].

6.2.2. Hybrid processor based Reconstruction System

The CS algorithm here was divided into two parts, the first is performed on the ARM processor and the second is performed on the DSP. The part to be on the ARM processor includes all preparation processes for the CS algorithm including memory allocations, read and write of test data to the shared memory with the DSP. The part to be on the DSP includes the core processes, iterations, and computations of the CS algorithm and it uses all the data written by the ARM in the shared memory. All the memory needed by the algorithm was fixed and preallocated from the ARM side [8].

6.3. Results and Discussion

After testing the algorithm on the BeagleBoard we get the reconstructed image using CS as shown in Figure 6.4 and it was identical to the one produced by the same algorithm tested on the large processing platforms.

The performance of the algorithm after trying it on the different platforms is shown Table 6.1. All the processors give longer processing time than the time expected for this algorithm especially on the DSP while that on the ARM processor was found to be surprisingly close to significantly larger processing platforms. This is apparently due to the dependence of the algorithm on 2D Fourier transform and the prolonged loops which take long processing time. The excessive processing time obtained when running the same algorithm on the DSP was difficult to explain at first until further research was done and that revealed the different architecture of this platform that requires very different coding strategy to take advantage of the available computing hardware on the processor. Hence, simple porting of code running on other general purpose processors is not a good strategy to develop efficient code on DSPs.

Difficulties were also found in attempting to transfer data between the ARM and DSP parts of the OMAP processor. The data passing interface allowed limited data packets that barely allowed the 100×100 sized image to be transferred for processing. This is clearly a very challenging problem facing the porting of such algorithms into embedded DSPs.

It should be noted that the algorithm used was implemented using serial code. This did not clearly take advantage of the number of processors available on the processing platform used. Hence, the difference between the first two Intel-based platforms with 2 and 8 processors can be attributed only to differences in clock speed rather than number of processor.

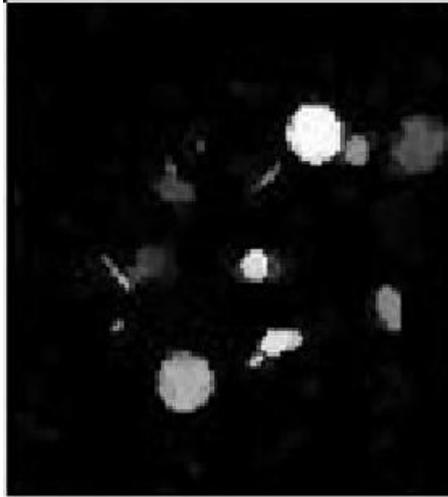


Figure 6.4: CS based reconstructed image.

Table 6.1: Processing time.

Module	Processing Time (min)
Core duo	22.5
Core i7	16.5
ARM	30.1
DSP	About 1000

Conclusions and Future Work

The results confirmed the theory of compressed sensing as a powerful method of image reconstruction under very low sampling conditions. The performance of the CS algorithm can be enhanced in both speed and quality of reconstructions through the use of some modifications as using l_0 penalized reconstruction which gives better images than those produced by l_1 penalized reconstructions. The used smoothed version of the l_0 -norm introduced a success in both the quality of reconstructed images which were better than their counterparts in the l_1 penalized reconstruction and processing time that was found to be very close to the time of using l_1 penalized problem.

Using Fourier transform as a sparse transform was found to give better results than Wavelet transform if using an l_1 penalized reconstruction and approximately the same results if using the l_0 penalization. Also it was found that the Fourier transform is time consuming if used with the l_0 penalization and speeds up the algorithm if used with l_1 penalization. The best performance for the CS algorithm in both quality and processing time was found to be achieved if using the l_0 penalized reconstruction with Wavelet or the l_1 penalized reconstruction with Fourier.

A further research will be done in order to enhance the performance of the compressed sensing algorithm through taking in consideration the symmetry of encoded state (k-space) at which the MR images are acquired. This may increase the number of acquired samples in the k-space from the same low sampling conditions of CS.

The processing time of the algorithm is compared on different processing platforms with results indicating interesting performance for the embedded ARM processor part of the OMAP processor. Also, the results indicated that the porting of such sophisticated algorithm to the DSP was not straightforward and that simple porting resulted in a very poor performance. So, special coding methods that take advantage of the architecture of the DSP to utilize the vectored computational hardware and pipelining must be carefully mapped onto the algorithm before it is ported. Further investigation is needed to develop specific porting instructions to allow the performance of the DSP to reach its theoretical limit. Targeting new embedded platforms that allow direct communication and debugging on DSP and containing multicores DSP's will be an interesting path to follow to allow further investigation for the performance of such special function processors with the challenging and computationally expensive algorithms like compressed sensing.

References

- [1] R. H. Hashemi, W. G. Bradley, and C. J. Lisanti, *MRI: The Basics*. Lippincott Williams & Wilkins, 2010, p. 400.
- [2] D. Weishaupt, V. D. Koechli, and B. Marincek, *How does MRI work?: An Introduction to the Physics and Function of Magnetic Resonance Imaging*. Springer, 2008, p. 182.
- [3] Advanced Imaging, “MRI Frequently Asked Questions.” [Online]. Available: <http://www.advancedimagingofmt.com/index.php?page=mri-faq>.
- [4] E. M. Haacke, R. W. Brown, M. R. Thompson, and R. Venkatesan, *Magnetic Resonance Imaging: Physical Principles and Sequence Design*. Wiley-Liss, 1999, p. 914.
- [5] M. Lustig, D. Donoho, and J. M. Pauly, “Sparse MRI: The application of compressed sensing for rapid MR imaging.,” *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–95, Dec. 2007.
- [6] 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, Mar. 2008.
- [7] G. A. Wright, “Magnetic resonance imaging,” *IEEE Signal Processing Magazine*, vol. 14, no. 1, pp. 56–66, 1997.
- [8] Y. A. Amer, M. A. El-Tager, E. A. El-Alamy, A. Abdel-Salam, and Y. M. Kadah, “Embedded magnetic resonance image reconstruction using compressed sensing,” in *2012 Cairo International Biomedical Engineering Conference (CIBEC)*, 2012, pp. 35–38.
- [9] E. J. Candes and M. B. Wakin, “An Introduction To Compressive Sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.
- [10] E. J. Candes, J. Romberg, and T. Tao, “Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information,” *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.
- [11] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
- [12] E. J. Candes 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, Dec. 2006.

- [13] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, vol. 642. Boston, MA: Springer US, 2002.
- [14] T. Chang, L. He, and T. Fang, “MR Image Reconstruction from Sparse Radial Samples Using Bregman Iteration,” *Proc. of the 14th Annual Meeting of ISMRM, Seattle*, vol. 4, p. 696, 2006.
- [15] J. C. Ye, S. Tak, Y. Han, and H. W. Park, “Projection reconstruction MR imaging using FOCUSS,” *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 57, no. 4, pp. 764–75, Apr. 2007.
- [16] K. T. Block, M. Uecker, and J. Frahm, “Undersampled radial MRI with multiple coils. Iterative image reconstruction using a total variation constraint,” *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 57, no. 6, pp. 1086–98, Jun. 2007.
- [17] J. H. L. M. Lustig, “Faster imaging with randomly perturbed, undersampled spirals and ℓ_1 reconstruction,” *Proc. of the 13th Annual Meeting of ISMRM, Miami Beach*, p. 685, 2005.
- [18] J. M. Santos, C. H. Cunningham, M. Lustig, B. A. Hargreaves, B. S. Hu, D. G. Nishimura, and J. M. Pauly, “Single breath-hold whole-heart MRA using variable-density spirals at 3T,” *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 55, no. 2, pp. 371–9, Feb. 2006.
- [19] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic Decomposition by Basis Pursuit,” *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, Jan. 1998.
- [20] J. F. Claerbout and F. Muir, “ROBUST MODELING WITH ERRATIC DATA,” *GEOPHYSICS*, vol. 38, no. 5, pp. 826–844, Oct. 1973.
- [21] F. Santosa and W. W. Symes, “Linear Inversion of Band-Limited Reflection Seismograms,” *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1307–1330, Oct. 1986.
- [22] S.-J. Kim, K. Koh, M. Lustig, and S. Boyd, “An Efficient Method for Compressed Sensing,” in *2007 IEEE International Conference on Image Processing*, 2007, vol. 3, pp. III – 117–III – 120.
- [23] E. J. Candes and J. K. Romberg, “Signal recovery from random projections,” in *Electronic Imaging 2005*, 2005, pp. 76–86.
- [24] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

- [25] J.-L. Starck, M. Elad, and D. L. Donoho, “Image decomposition via the combination of sparse representations and a variational approach,” *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1570–1582, Oct. 2005.
- [26] M. Elad, B. Matalon, and M. Zibulevsky, “Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization,” *Applied and Computational Harmonic Analysis*, vol. 23, no. 3, pp. 346–367, Nov. 2007.
- [27] D. L. Donoho, M. Elad, and V. N. Temlyakov, “Stable recovery of sparse overcomplete representations in the presence of noise,” *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 6–18, Jan. 2006.
- [28] M. M. Bronstein, A. M. Bronstein, M. Zibulevsky, and H. Azhari, “Reconstruction in diffraction ultrasound tomography using nonuniform FFT.,” *IEEE transactions on medical imaging*, vol. 21, no. 11, pp. 1395–401, Nov. 2002.
- [29] Linux, “Embedded Computing with Beagle Board,” 2011. [Online]. Available: <http://www.start-linux.com/embedded-linux-boards/embedded-computing-with-beagle-board>.
- [30] T. Instruments, “BeagleBoard System Reference Manual Rev C4,” 2009.
- [31] T. Instruments, “OMAP35x Applications Processor Technical Reference Manual,” 2012.
- [32] K. P. Pruessmann, M. Weiger, P. Börnert, and P. Boesiger, “Advances in sensitivity encoding with arbitrary k-space trajectories.,” *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 46, no. 4, pp. 638–51, Oct. 2001.
- [33] H. Z. William W Hager, “A survey of nonlinear conjugate gradient methods,” *Pacific journal of Optimization*, vol. 2, no. 1, pp. 35–38, 2006.
- [34] W. W. Hager and H. Zhang, “A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search,” *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, Jan. 2005.
- [35] R. Fletcher and C. Reeves, “Function minimization by conjugate gradients,” *The Computer Journal*, vol. 7, no. 2, pp. 149–154, Feb. 1964.
- [36] Y. Zhang, H. D. Cheng, J. Huang, and X. Tang, “An effective and objective criterion for evaluating the performance of denoising filters,” *Pattern Recognition*, vol. 45, no. 7, pp. 2743–2757, Jul. 2012.
- [37] C. Loizou, *Despeckle Filtering Algorithms and Software for Ultrasound Imaging (Synthesis Lectures on Algorithms and Software in Engineering)*. Morgan and Claypool Publishers, 2008, p. 166.

- [38] C. S. Oxvig, P. S. Pedersen, T. Arildsen, and T. Larsen, “Improving Smoothed l_0 Norm in Compressive Sensing Using Adaptive Parameter Selection,” *Computing Research Repository*, p. 7, Oct. 2012.

Appendix A: Getting BeagleBoard Ready

As mentioned before, BeagleBoard can be powered to act like a minicomputer by loading it with an operating system (OS). This OS may be a Linux or Windows based OS and in our work we used a Linux based OS. A complete Linux distribution: includes the kernel, a base file system, basic tools and even a package manager to install software from a repository was used (one of projects of Yocto Project). It is optimized for low-power controllers like the one in BB and intends to be small and basic system to modify on your needs.

The Yocto Project is an umbrella project covering a fairly wide swath of embedded Linux technologies. It is not a Linux distribution. The Yocto Project™ is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture. It's a complete embedded Linux development environment with tools, metadata, and documentation - everything you need. The tools are easy to get started with, powerful to work with (including emulation environments, debuggers, an Application Toolkit Generator, etc.) and they allow projects to be carried forward over time without causing you to lose optimizations and investments made during the project's prototype phase and to focus on their specific product features and development. The Yocto Project fully supports a wide range of hardware and generates images for many kinds of devices and supports device emulation through the QEMU Emulator.

Needed things to develop the Yocto project environment:

- A host system running a supported Linux distribution (i.e. recent releases of Fedora, openSUSE, CentOS, and Ubuntu). If the host system supports multiple cores and threads, you can configure the Yocto Project build system to decrease the time needed to build images significantly.
- The right packages.
- A release of the Yocto Project.

Environment Development

Packages and package installation vary depending on your development system and on your intent. The next sections list – for Ubuntu - the required packages needed to build an image that runs on BeagleBoard (the ARM side). The Yocto project has many distributions like Poky and Angstrom and we here use Poky distribution.

1. The Packages

The essential packages you need for Ubuntu can be acquired through the following command:

```
$ sudo apt-get install gawk wget git-core diffstat unzip  
texinfo \build-essential chrpath libsdl1.2-dev xterm
```

2. Building an Image

In the development environment you will need to build an image whenever you change hardware support, add or change system libraries, or add or change services that have dependencies.

Use the following commands to build your image. The OpenEmbedded build process creates an entire Linux distribution, including the toolchain, from source.

```
$ wget http://downloads.yoctoproject.org/releases/yocto/yocto-1.4/poky-dylan-9.0.tar.bz2
$ tar xjf poky-dylan-9.0.tar.bz2
$ cd poky-dylan-9.0
$ source oe-init-build-env
```

Notes

- The build process using Sato currently consumes about 50GB of disk space. To allow for variations in the build process and for future package expansion, we recommend having at least 100GB of free disk space.
- The first command retrieves the Yocto Project release tarball from the source repositories using the `wget` command.
- The second command extracts the files from the tarball and places them into a directory named `poky-dylan-9.0` in the current directory.
- The third and fourth commands change the working directory to the **Source Directory** and run the Yocto Project *oe-init-build-env* environment setup script. Running this script defines OpenEmbedded build environment settings needed to complete the build. The script also creates the **Build Directory**, which is *build* in this case and is located in the Source Directory. After the script runs, your current working directory is set to the Build Directory. Later, when the build completes, the Build Directory contains all the files created during the build.

Examine your *local.conf* file in your project's configuration directory, which is found in the **Build Directory**. The defaults in that file should work fine. However, there are some variables of interest at which you might look. By default, the target architecture for the build is *qemux86*, which produces an image that can be used in the *QEMU* emulator and is targeted at an Intel® 32-bit based architecture. To change this default, edit the value of the *MACHINE* variable in the configuration file before launching the build (select that one of BeagleBoard).

Another couple of variables of interest are the *BB_NUMBER_THREADS* and the *PARALLEL_MAKE* variables. By default, these variables are commented out. However, if you have a multi-core CPU you might want to uncomment the lines and set both variables equal to twice the number of your host's processor cores. Setting these variables can significantly shorten your build time.

Continue with the following command to build an OS image for the target, which is *core-image-sato* in this example.

```
$ bitbake -k core-image-sato
```

Depending on the number of processors and cores, the amount of RAM, the speed of your Internet connection and other factors, the build process could take several hours the first time you run it. Subsequent builds run much faster since parts of the build are cached.

Upon finishing you can find your image data in the following path (`yocto/poky-danny-8.0/build/tmp/deploy/images`) this data includes the following important files which will be loaded on our board:

- `uImage`
- `u-boot.bin`

- MLO-beagleboard
- core-image-sato-beagleboard.tar.bz2

Compiling Kernel

A kernel should be compiled and prepared to be added to system modules on BeagleBoard. One can bring a kernel from kernel.org, compile it and it will work properly on the board but for BeagleBoard there is a good kernel which has better compatibility with our system. This kernel is called Balister kernel. Philip Balister kernel is contained in the following link you just needs to zip its contents (<https://github.com/balister/linux-omap-philip>).

Before starting kernel compiling you need to make sure from installing the ARM compilers (code sourcery) from (<http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/>). Once you install them you will need to add the path of compilers (default is: /opt/CodeSourcery/Sourcery_G++_Lite/bin) to the system path using the export command in Ubuntu as follows:

```
export PATH="$PATH:/opt/CodeSourcery/Sourcery_G++_Lite/bin"
```

but take care while you perform this. If not properly executed, can destroy your system path (this step can be included in the second item in the following steps as will be shown).

Steps:

1. Change your directory to balister kernel folder.
2. Create the environment variables which include the architecture and cross compilers and the code sourcery path (if you did not perform before) in a file and call it env-vars as follows:

```
export PATH=$PATH:/opt/CodeSourcery/Sourcery_G++_Lite/bin
export CROSS_COMPILE=arm-none-linux-gnueabi-
export ARCH=arm
```

3. Source the file you created in 1 using the following command:

```
source env-vars
```

4. Copy the file named “*defconfig*” to “*.config*”.
5. Type the following command “`make menuconfig`”.
6. On execution of the previous command you will have the menu in Figure A.1.
7. Select “system type”.
8. Select “TI OMAP common features” as in Figure A.2.
9. Mark labels beside “IOMMU Module” and “IOMMU_IVA2” as in Figure A.3.
10. Save and exit the configuration.
11. Now build the kernel and compile its modules using the following commands:

```
make -j4 uImage
make -j4 modules
```

12. Now install your kernel modules using the following command:

make `INSTALL_MOD_PATH=path` `modules_install`
but replace “**path**” with the path you want and at which the kernel modules will be created.

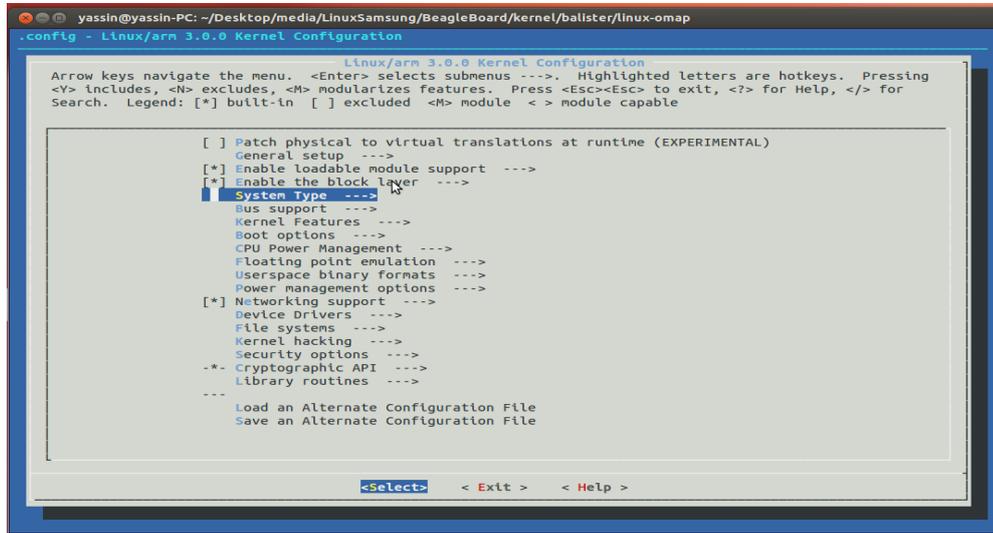


Figure A.1: Kernel menu configuration.

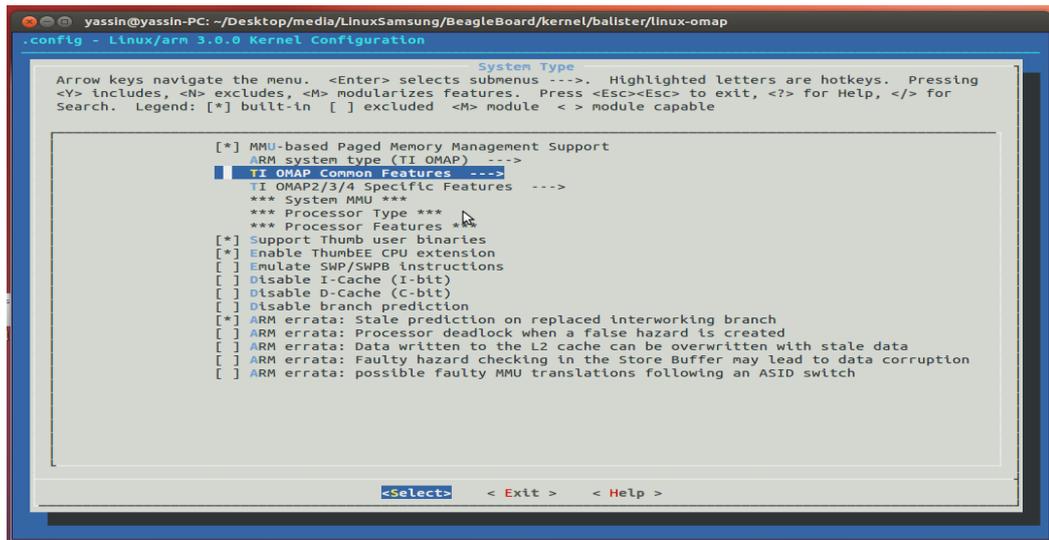


Figure A.2: Kernel menu configuration.

- Now you have your kernel ready at folder called “lib” at the path you selected in the previous step.

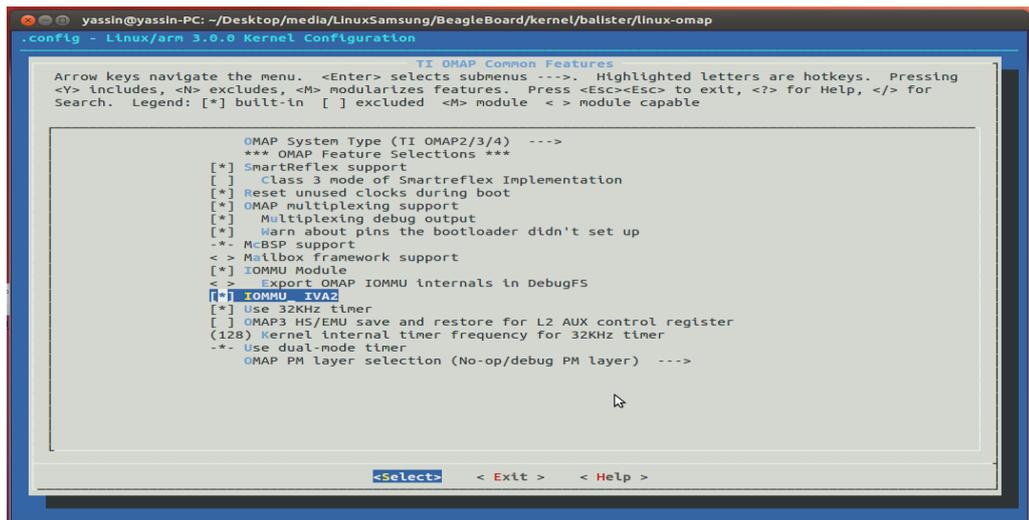


Figure A.3: Kernel menu configuration.

Building Inter-processor Communication RTOS

In order to perform inter-processor communication between ARM acting as GPP and the DSP we need a real time OS called SYSLINK to provide the needed API's used for this process to handle the shared memory between the two processors. SYSLINK can be downloaded from TI company website (http://software-dl.ti.com/dsp/dsp_public_sw/sdo_sb/targetcontent/syslink/index.html).

The SysLink product provides the following services to frameworks and applications:

- Processor Manager
- Inter-Processor Communication
- Utility modules

Steps:

1. Before starting the build process you should make sure of installing the following components on your host system:
 - a. TI Linux EZ Software Development Kit (EZSDK) for Sitara™ ARM® Microprocessors and you can download from (<http://www.ti.com/tool/linuxezsdk-sitara>) and a detailed installation for SDK is available here (<http://www.fedevl.com/welldoneblog/2011/09/c6a816x-installation-host-machine-linux-kernel-compilation/>).
 - b. TI inter-processor communication (IPC) and you can download from (http://downloads.ti.com/dsp/dsp_public_sw/sdo_sb/targetcontent/ipc/).
 - c. TI BIOS and you can download from (http://downloads.ti.com/dsp/dsp_public_sw/sdo_sb/targetcontent/bios/sysbios/6_34_04_22/index_FDS.html).
 - d. TI XDC tools and you can download from (http://downloads.ti.com/dsp/dsp_public_sw/sdo_sb/targetcontent/rtsc/3_24_05_48/index_FDS.html).

- e. TI C6000 code generation tools and you can download from (https://www-a.ti.com/downloads/sds_support/TICodegenerationTools/download.htm).
2. Extract the tar.gz file downloaded to any path you want using the following command:

```
tar -xzvf syslink_<version>.tar.gz -C path
```

and change your directory to Syslink directory.

3. Update the *product.mak* file for your environment (BB) by changing the following parameters:
 - a. Device.
 - b. Directories for different TI components (IPC, XDCtools, BIOS, and SDK).
4. Build the Syslink driver and libraries with the following command:

```
make syslink
```

5. Build the examples using the following command:

```
make examples
```

6. To assemble the SysLink executables and examples into a directory structure suitable for running on the device's file-system use the following command:

```
make install
```

Preparing an SD Card carrying a Linux image for BB-xm

The Linux image built on the host device is now ready for booting the BeagleBoard and in order to use you should prepare an SD card carrying the image to boot your BeagleBoard. As mentioned all the needed files for the image are on the following path where you installed your Yocto project Linux distribution (yocto/poky-danny-8.0/build/tmp/deploy/images). A micro SD card of at least 4 GB is needed and to have it loaded with the image it should be divided into two partitions as in Figure A.4. The first partition is boot partition and its format should be (**bootable** FAT16) with a size that does not exceed 50 MB, the second partition is the one for Linux file system and it should be in (EXT3) format with the remaining bytes in the card. One can use for example the Ubuntu disk utility to create the needed partitions on SD card.

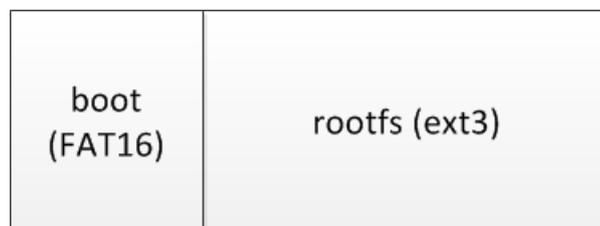


Figure A.4: SD card partitions

After creating partitions they will not be loaded with the Linux image. The bootable partition will contain a copy of the following files:

- X-loader “MLO-beagleboard” and rename it to MLO on the card.
- U-boot binary “u-boot.bin”.
- uImage.
- Boot script file “boot.scr” (the next paragraph shows how to create it).

A boot script file is created by setting the environment variable in a “cmd” file. Environment variables are the boot arguments and the boot command for the BB. A “cmd” file with any name like “boot.cmd” is created and filled with arguments in Table A.1. And the “boot.cmd” file will look like the snapshot in Figure A.5.

Table A.1: Environment Variables.

Variable	Value
bootargs	console=tty0 console=ttyO2,115200n8 root=/dev/mmcblk0p2 mem=200M@0x80000000 mem=256M@0x90000000 rootwait rootfstype=ext3 ro
bootcmd	mmc init; fatload mmc 0:1 0x80300000 uImage; bootm 0x80300000

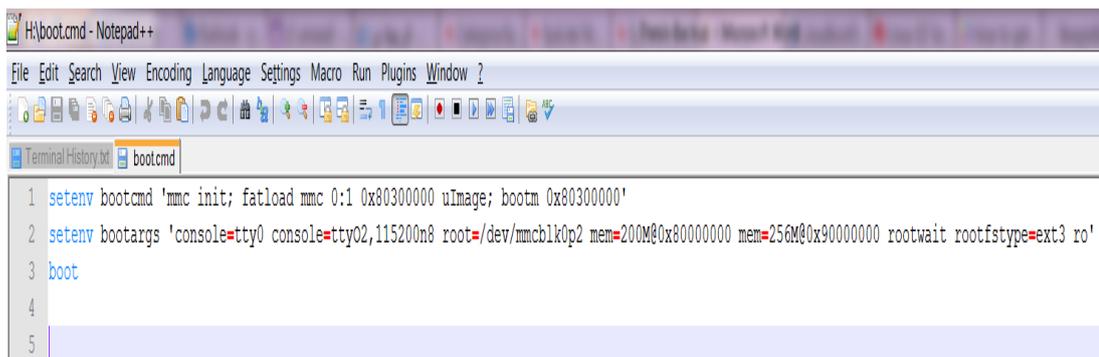


Figure A.5: “boot.cmd” file.

Use the following command to create “boot.scr” file:

```
mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n 'Execute uImage.bin' -d boot.cmd boot.scr
```

After preparing the boot partition now extract your file system on the second partition “rootfs”. The file system for your BB is the file called “core-image-sato-beagleboard.tar.bz2” in the same directory of the previous files. Then move the compiled Balister kernel to the file system partition by zipping the lib directory produced from kernel compile process and extracting it at this SD partition. Also copy the built Syslink to the file system partition to use it. Now all the components needed for using your BB are ready and all loaded to the SD card for plug and play. After booting the BB you should insert the Syslink module to system modules using “insmod” command and you will find the module in the following path (syslink/lib/-/-/-/-/ syslink.ko).

Booting BeagleBoard-XM using USB-Serial converter

1. Use the command `minicom -s` to configure the serial port settings.
2. In serial port setup Figure A.6 change:
 - a. Serial device to be `"/dev/ttyUSB0"`.
 - b. Hardware flow control to be `"No"`.
3. Save your setup as `df1`.
4. Exit.
5. Power the BeagleBoard using USB or 5V power adapter.
6. Wait till it requests BeagleBoard login and log in as root.

```
+-----+
| A -   Serial Device       : /dev/ttyS1          |
| B - Lockfile Location    : /var/lock           |
| C - Callin Program       :                    |
| D - Callout Program      :                    |
| E -   Bps/Par/Bits       : 115200 8N1         |
| F - Hardware Flow Control : No                |
| G - Software Flow Control : No                |
|                                     |
|   Change which setting?                 |
+-----+
| Screen and keyboard |
| Save setup as df1   |
| Save setup as..     |
| Exit                |
| Exit from Minicom   |
+-----+
```

Figure A.6: using minicom command.

Compiling a C code file and running it on the ARM processor

1. First make of the presence of cross compilers path (code sourcery) in the system path by echoing `PATH` and if not add it using `export` command as mentioned in earlier section.
2. Add the path of the Sourcery bin folder to the system `PATH`.
3. Compile your C code using `arm-none-linux-gnueabi-gcc` with the following command:

```
arm-none-linux-gnueabi-gcc -Wall file_name.c -lm -o
output_file_name
```

4. Copy the executable file to the file system of BeagleBoard on the SD-card or transfer it using Ethernet and run it from its terminal.

Running an algorithm on the DSP processor using SysLink

1. You should have the source of SysLink (you can get it from: http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/syslink/index.html).

2. You should divide your algorithm into two parts the 1st to be run on the ARM processor and the other on DSP (memory allocations and reading or writing files are the parts to be done on the ARM and calculations are the parts to be run DSP).
3. You can use any of the examples provided with the SysLink to write your code in order to use its architecture without the need to write it from scratch.
4. After finishing your code perform the build operation to get your executable files as mentioned in an earlier section.

المخلص

إن التصوير بالرنين المغناطيسي هو أحد أهم وسائل التصوير الطبي. يمثل الخضوع لجلسة تشخيصية داخل جهاز رنين مغناطيسي الإستلقاء بدون حركة لمدة طويلة قد تصل أو تتخطى ٤٥ دقيقة وهو أمر غير مريح، هذا بالإضافة إلى عدد لا يحصى من الأصوات المزعجة ورنين الاذن والتلف الذي قد يحدث في الصورة نتيجة حركة المريض، وبالتالي فإن خفض زمن الفحص لصورة الرنين المغناطيسي هو أمر في غاية الأهمية. هناك نظرية ظهرت حديثاً معروفة بإسم الإحساس المضغوط وهي عبارة عن تقنية جديدة للقياس والتي تقلل عدد القياسات المطلوبة لبناء الصورة بدون أي فقد يذكر في جودة الصورة. يتيح إستخدام هذه التقنية في التصوير بالرنين المغناطيسي تقليل زمن الفحص بدرجة كبيرة كما أن استخدام مثل هذه الطريقة مع أنظمة المعالجة المدمجة سيكون ذو منفعة كبيرة من ناحية تقليل الحجم والتكلفة. قما في هذه الرسالة بتنفيذ نظام لبناء صور الرنين المغناطيسي اعتماداً على نظرية الاحساس المضغوط وتم إختباره وتعديله لإنتاج صور ذات جوده عالية في وقت أقل بإستخدام نفس عدد القياسات المستخدمة في الخوارزم الأصلي وتم اختبار كفاءة تشغيله على عدد من الأنظمة منها نظام معالجة مدمج يعتمد على المعالج (OMAP). هذا العمل أظهر نتائج جيدة بالنسبة لكفاءة الصور المبنية من خلال فضاء ترددي مضغوط بنسبة تصل إلى ٩٠% بإستخدام الإحساس المضغوط كما أن تشغيله على نظام المعالجة المدمج أظهر نتائج جيدة بالإهتمام ووجهنا لأكثر من اتجاه لتحسين كفاءة تشغيل مثل تلك الأنظمة المدمجة في تطبيقات التصوير الطبي.



مهندس: ياسين عامر السيد عامر
تاريخ الميلاد: ١٩٨٩\٣\٥
الجنسية: مصري
تاريخ التسجيل: ٢٠١٠\١١\١١
تاريخ المنح: ١ ١
القسم: الهندسة الحيوية الطبية والمنظومات
الدرجة: ماجستير
المشرفون:

أ.د. ياسر مصطفى إبراهيم قدح

الممتحنون:

أ.د. ياسر مصطفى إبراهيم قدح (المشرف الرئيسي)
أ.د. ناهد حسين علي سلومة (الممتحن الداخلي)
أ.د. محمد إبراهيم العدوي (الممتحن الخارجي)

عنوان الرسالة:

بناء صور الرنين المغناطيسي باستخدام تقنية الإحساس المضغوط على أنظمة المعالجة المدمجة

الكلمات الدالة:

الرنين المغناطيسي، بناء الصور، الإحساس المضغوط، معالج الاشارات الرقمية ، الأنظمة المدمجة

ملخص الرسالة:

إن التصوير بالرنين المغناطيسي هو أحد أهم وسائل التصوير الطبي. يمثل الخضوع لجلسة تشخيصية داخل جهاز رنين مغناطيسي الإستلقاء بدون حركة لمدة طويلة قد تصل أو تتخطى ٤٥ دقيقة وهو أمر غير مريح، هذا بالإضافة إلى عدد لا يحصى من الأصوات المزعجة ورنين الاذن والتلف الذي قد يحدث في الصورة نتيجة حركة المريض، وبالتالي فإن خفض زمن الفحص لصورة الرنين المغناطيسي هو أمر في غاية الأهمية. هناك نظرية ظهرت حديثاً معروفة بإسم الإحساس المضغوط وهي عبارة عن تقنية جديدة للقياس والتي تقلل عدد القياسات المطلوبة لبناء الصورة بدون أي فقد يذكر في جودة الصورة. يتيح إستخدام هذه التقنية في التصوير بالرنين المغناطيسي تقليل زمن الفحص بدرجة كبيرة كما أن استخدام مثل هذه الطريقة مع أنظمة المعالجة المدمجة سيكون ذو منفعة كبيرة من ناحية تقليل الحجم والتكلفة. قمنا في هذه الرسالة بتنفيذ نظام لبناء صور الرنين المغناطيسي اعتماداً على نظرية الإحساس المضغوط وتم إختباره وتعديله لإنتاج صور ذات جودة عالية في وقت أقل بإستخدام نفس عدد القياسات المستخدمة في الخوارزم الأصلي وتم اختبار كفاءة تشغيله على عدد من الأنظمة منها نظام معالجة مدمج يعتمد على المعالج (OMAP). هذا العمل أظهر نتائج جيدة بالنسبة لكفاءة الصور المبنية من خلال فضاء ترددي مضغوط بنسبة تصل إلى ٩٠% بإستخدام الإحساس المضغوط كما أن تشغيله على نظام المعالجة المدمج أظهر نتائج جيدة بالإهتمام ووجهنا لأكثر من اتجاه لتحسين كفاءة تشغيل مثل تلك الأنظمة المدمجة في تطبيقات التصوير الطبي.

بناء صور الرنين المغناطيسي باستخدام تقنية الإحساس المضغوط على أنظمة
المعالجة المدمجة

اعداد
ياسين عامر السيد عامر

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة الماجستير
في
الهندسة الحيوية الطبية والمنظومات

يعتمد من لجنة الممتحنين:

المشرف الرئيسي

الأستاذ الدكتور: ياسر مصطفى إبراهيم قدح

المتحن الداخلي

الأستاذ الدكتور: ناهد حسين علي سلومة

المتحن الخارجي

الأستاذ الدكتور: محمد إبراهيم العدوي

كلية الهندسة - جامعة القاهرة
الجيزة - جمهورية مصر العربية

٢٠١٣

بناء صور الرنين المغناطيسي باستخدام تقنية الإحساس المضغوط على أنظمة
المعالجة المدمجة

اعداد

ياسين عامر السيد عامر

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة الماجستير
في
الهندسة الحيوية الطبية والمنظومات

تحت اشراف

أ.د. ياسر مصطفى إبراهيم قدح
أستاذ بقسم الهندسة الحيوية الطبية
والمنظومات

كلية الهندسة - جامعة القاهرة
الجيزة - جمهورية مصر العربية

٢٠١٣



بناء صور الرنين المغناطيسي باستخدام تقنية الإحساس المضغوط على أنظمة
المعالجة المدمجة

اعداد

ياسين عامر السيد عامر

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة الماجستير
في
الهندسة الحيوية الطبية والمنظومات

كلية الهندسة-جامعة القاهرة
الجيزة-جمهورية مصر العربية

٢٠١٣