THREE DIMENSIONAL IMAGE RECONSTRUCTION (3DIRECT) OF SPARSE SIGNALS WITH MRI APPLICATION

by

MELINDA M AU

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2016

I dedicate this thesis to my best friend and future husband Kyle. I love you.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Ren-Cang Li for his continuous support, patience and immense knowledge he bestowed upon me. His guidance was unwavering, and his intellectual curiosity allowed me to pursue my research interests whereever they may have led. I could not have imagined a better advisor and mentor to help me along my journey of completing my PhD.

I would like to thank the rest of my thesis committee: Dr. Jianzhong Su, Dr. Guojun Liao, Dr. Hristo Kojouharov, and Dr. Gaik Ambartsoumian, for their insightful comments and encouragement. A special note of thanks to Dr. Su for his connections to University of Texas South Western Medical Center for the data sets used in this thesis.

Further, I would like to thank Dr. Ray Johnson, Dr. David Hunn, Dr. Robie Samanta Roy, Dr. David Pustai and Robert McCarty for the opportunity to complete my dream of obtaining my doctorate. Your support, funding and mentorship has allowed me to have some of the best years in my career. I am forever grateful.

Lastly, it goes without saying, but thank you to my family and friends. Mom and Dad, you have always encouraged me to push myself and reach for my dreams. I would not be here today if it wasn't for you both. To my friend Sarah, who helped me through late night study sessions, comprehensive exams and being my sounding board, you made this experience all the more enjoyable. Lastly, to my husband-to-be Kyle, thank you for always supporting me, here's to many more years of happiness.

October 21, 2016

ABSTRACT

THREE DIMENSIONAL IMAGE RECONSTRUCTION (3DIRECT) OF SPARSE SIGNALS WITH MRI APPLICATION

MELINDA M AU, Ph.D.

The University of Texas at Arlington, 2016

Supervising Professor: Ren-Cang Li

Sparse signal reconstruction has been steadily gaining tremendous attention, specifically in applications of compressed sensing as well as feature selection in signal processing methods [*IEEE Signal Processing*, Vol. 25 (2008) pp.21-30]. Standard techniques to solve these problems such as the nonlinear Conjugate Gradient method have been used successfully on small and medium sized problems. However, as the desire to collect more data becomes the trend, and the need to process information more quickly and reliably becomes more prevalent than before, these standard methods become inadequate.

In this thesis, we present a new approach for sparse signal reconstruction called the *Three Dimensional Image Reconstruction* (3DIRECT) method. This method minimizes total variation using the interior point method via the log-barrier function to recover an image sparsely sampled in the frequency domain. Furthermore, this method is able to recover all of the slices of 3D images simultaneously, as oppose to the current state-of-the-art methods which traditionally recover 3D images one slice at a time.

We applied our method to MRI data and observed the following results. First, our method improves upon the speed of existing interior point methods by leveraging object oriented coding, as well as analyzing and improving the stopping criteria in the Conjugate Gradient (CG) method. The 3DIRECT method also benefits from speed improvements due to the structure of the optimization model. The 3D model requires less calls to iterative methods compared to the 2D models. Thus, our method achieves similar or better results than current state-of-the-art methods in less time, e.g. for a $128 \times 128 \times 128$ image using 17% sampling, 3DIRECT is 22% faster in mean and 33% faster in minimum observed run times.

Finally, our method is able to achieve a cleaner recovered MRI image under certain conditions, due to the ability to exploit physical information in the z-direction. We provide an algorithm which is able to detect this superior region of performance, and provide an indication of when to use the 3DIRECT method versus traditional 2D methods. In cases, where the image has homogeneity in the diameter of the cross-section from slice to slice, the 3DIRECT method provides the best results.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

We are interested in solving a 3D sparse image reconstruction problem modeled by

$$\min_X \mathrm{TV}(X) \quad \text{subject to } \|\mathcal{A}(X) - B\|_2 \leq \epsilon, \tag{1.1}$$

where $\mathcal{A} : \mathbb{R}^{n\times n\times n} \mapsto \mathbb{R}^{n\times n\times n}$ is a sparsifying linear transform, $B \in \mathbb{R}^{n\times n\times n}$ are the measured observations of sampled pixels of an image in the k-space, and $\mathrm{TV} : \mathbb{R}^{n\times n\times n} \mapsto \mathbb{R}$ represents the total variation of X and is the sum of the magnitudes of the gradient in the $x-, y-$ and $z-$directions [14]. How the total variation function, TV, is computed will be discussed later.

We plan to solve minimization problem (1.1) using an extension of the 2D reconstruction method proposed in [8], which we call the 3DIRECT method. The 3DIRECT method casts (1.1) into a Second Order Cone Problem (SCOP), and then solves it using the interior point method via the log-barrier function based on techniques from [4]. Our method is able to do this by minimizing over the 3D total variation, and expressing our systems in such a way that the optimization method can recover all the slices of the 3D image simultaneously [32, 53].

Furthermore, our method uses *Object Oriented* (OO) coding to improve the efficiency of the finite difference operators used in the calculation of total variation [39]. We were able to achieve the same functionality in *half the time* for one finite difference calculation. As these calculations are repeatedly called in several subroutines, this improves the overall reconstruction time dramatically.

In addition to the OO efficiencies, we analyzed the CG convergence properties, and devise a better stopping criteria than the current state-of-the-art method in [8]. This reduces the number of iterations of CG while maintaining recovered image quality. Also, our method processes all the slices of the 3D image simultaneously; therefore, there are also less iterations of each subroutine per recovered image. Thus, further timing benefits are achieved by the 3DIRECT method.

In terms of performance, several different examples using various sample schemes for the linear transform operator, $\mathcal{A}$, were examined [9]. More about the specifics of $\mathcal{A}$ and sample schemes will be discussed in later chapters. However, two important results were achieved. One is the 3DIRECT method achieved similar performance to the 2D method in [8] 22% faster in mean and 33% faster in minimum observed run times, when using a 2D sample scheme. Also, our method was able to exploit a 3D sample scheme and achieve superior image clarity in a certain range of slices for a specific brain scan. This range can be predicted via an algorithm we devised which uses the starting image as its input. Because of this, we are able to predict applications where the 3DIRECT method will achieve better performance than 2D methods such as those in [8] and [34], at low sampling percentages on the order of 17% as oppose to the observed 40% required to get quality image results in [34]. Thus, our method provides a means to recover sparsely sampled 3D images (signals) simultaneously while providing speed improvement, and in certain conditions, better recovered image quality.

CHAPTER 2

THE PROBLEM

An important application of minimization problem (1.1) is MRI reconstruction. Often collecting MRI data is limited by hardware constraints and the limitations caused by patient discomfort [36, 46]. For instance, a signal sampled in the Fourier space is band-limited by half the sample rate. Thus, if you wish to reconstruct a signal without aliasing at a given frequency, your sampled rate must be at least twice the said frequency, which can results in a large number of samples [50, 59]. This is where compressed sensing can help speed up the process. Compressed sensing is a theory that with high probability, ensures how a signal can be defined by a small number of random samples than typically required by the traditional sampling schemes [18].

So, if we are given an image sparsely sampled in the frequency domain, how do we recover it? In [9], it was proven that recovery can be done by solving a convex optimization problem. Further, it develops a random sampling theorem that allows the reconstruction problem to overcome the uncertainty principle, which states that it is hard to localize a signal in time and frequency. Later on, [7] developed the *restricted isometry property* which was used as the foundation for proving several properties for compressed sensing. While we do not go into detail on these theoretically, they are fundamental in the ability to reconstruct sparse signals using nearly orthogonal operators in convex optimization problems such as (1.1).

It is important to understand how the formulation of (1.1) relates to the MRI application. Often times when MRI data is collected, it represents random samplings of Fourier coefficients referred to as k-space samples. Thus, for 2D problems, the

operator $\mathcal{A}$ represents a random sampling of the 2D Fourier transform [28, 38]. This random sampling will be referred to as the *mask* [42].

Current solution methods for the minimization problem process the 3D image one slice at a time using a 2D transformation operator and a 2D mask. The drawback to this is that the overall mask is independent from slice to slice. Therefore, this approach cannot exploit shared similarities of the cross-sections of the MRI image. Thus, each slice must be sampled at a minimum percentage to get desired recovered results. Note the use of minimum here is dependent on the end user desired recovered image quality.

In our method we propose using a 3D subsampled Fourier Transformation operator, and exploit using a 3D mask. In doing so we can now select the mask to exploit physical similarities between adjacent MRI slices. This results in being able to reduce the sample percentage while maintaining good results.

It should be noted, how a mask is chosen is its own field of study in compressed sensing, and is gaining lots of attention [9, 11]. For the purposes of this research we will show the results of a few masks and discuss the implications of each. First, let's consider a single slice of a brain fully sampled, and then take a look at what happens to the image if it is transformed into the k-space domain, sparsely sampled, and then transformed back to the image domain [20].

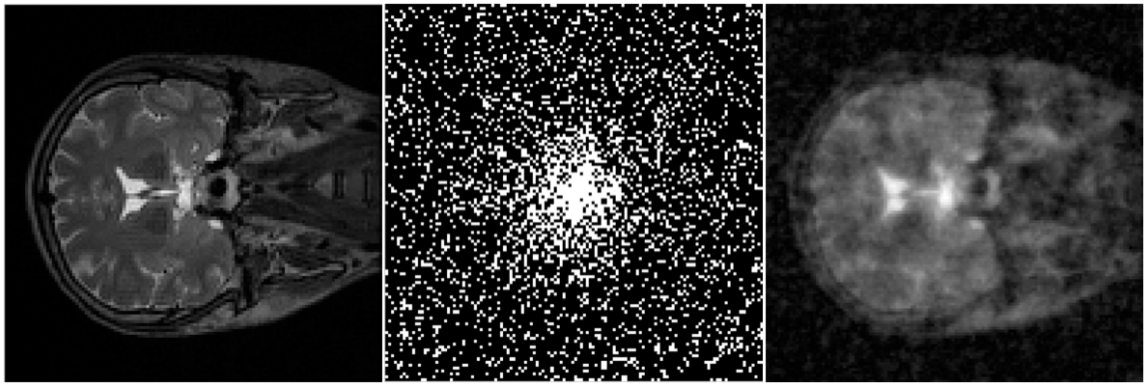

Figure 2.1. Artifacts of the 2D Subsample k-space Fourier Operator. Left: Fully Sampled Brain Image; Center: 2D Mask (20%); Right: Image artifacts caused by the defined sampling.

Figure 2.1 demonstrates the impacts of the mask embedded in the sparse transformation operator $\mathcal{A}$. The otherwise clear brain image now has aliased artifacts causing distortion of the image, which is due to the nature of compressed sensing [17]. In an ideal world all the samples would be collected, and there would be no need to devise algorithms to clear up the sampled image. However real world constraints, such as hardware limitations and patient discomfort, do not allow all samples to be collected. This leads to a classic image reconstruction problem, and is where the TV operator in Problem (1.1) comes into play. Given the observations measured in the right hand side of Figure 2.1, denoted as our matrix B in (1.1), the minimization of the total variation of the image leads to a clearer picture, and thus allows the doctors to see the details currently obscured by aliased artifacts.

It should be noted the mask used in Figure 2.1 was a 20% random sampling distributed around the center frequencies of the image. One can view a 2D Fourier Transform as the 0 frequency component centered in the middle of the grid with higher frequency components as you move towards the edges. The location of the

frequency bin on the grid indicates how much vertical and horizontal components are in the image [40]. For instance consider the following image provided by [5].

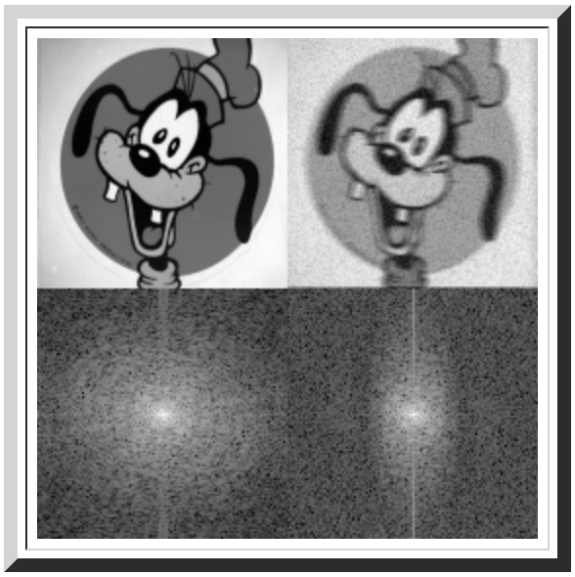

Figure 2.2. Top Left: Clear Image; Bottom Left: Fourier Transform of the Clear Image; Top Right: Fuzzy Image; Bottom Right: Fourier Transform of the Fuzzy Image.

Figure 2.2 demonstrates how different images appear in the k-space. The top left image is a clear image of Goofy with its k-space domain directly below showing frequencies centered in the middle and radiating out. The image on the right is a blurred image of Goofy. Upon inspection of the corresponding k-space domain, we see the horizontal frequencies have been attenuated. This is due to the fact that the source in [5] is applied a smoothing filter to the image in the horizontal direction only. Using this intuition about looking at images in the k-space we can now examine what a typical MRI image looks like in the k-space. Observe Figure 2.3, where a single slice of a brain scan is transformed into the k-space, subsampled, and transformed back into the image domain.

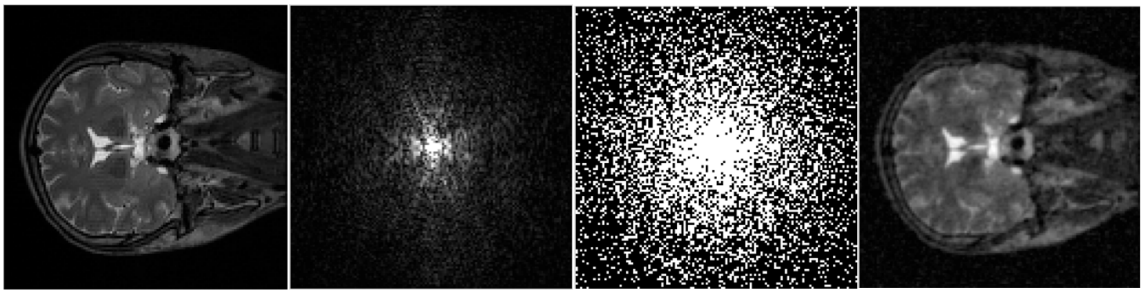

Figure 2.3. From Left to Right: Original Image; Fully Sampled k-space domain; 35% mask; Recovered Image with Artifacts.

Notice in Figure 2.3 the 2D fully sampled k-space domain shows the MRI frequency components are centered in the grid with some clear symmetries in the vertical and horizontal direction. This would lead one to believe the best Fourier coefficients to sample in our mask are those with a strong concentration around the middle frequencies. Using a 35% mask with a wider center a new recovered image is displayed on the right in Figure 2.3 . This image is much clearer than the recovered image from Figure 2.1, and it will be easier to minimize the total variation in accordance with (1.1). This is why the mask plays a critical role in the performance of the minimization problem [13].

Taking a step back from the mask implications, we now briefly discuss the ramifications of using a 3D Fourier Transform to process the 3D image all at once, versus using a 2D Fourier Transform and processing slice by slice. In order to see this we will process a 3D brain image that is $128 \times 128 \times 128$ in size. We will do this in two ways: first we will process each slice using a 2D Fourier Transform under the 21% mask, and second we will process the same data using a 3D Fourier Transform under the same mask as the 2D case. The results of this are displayed in Figure 2.4.

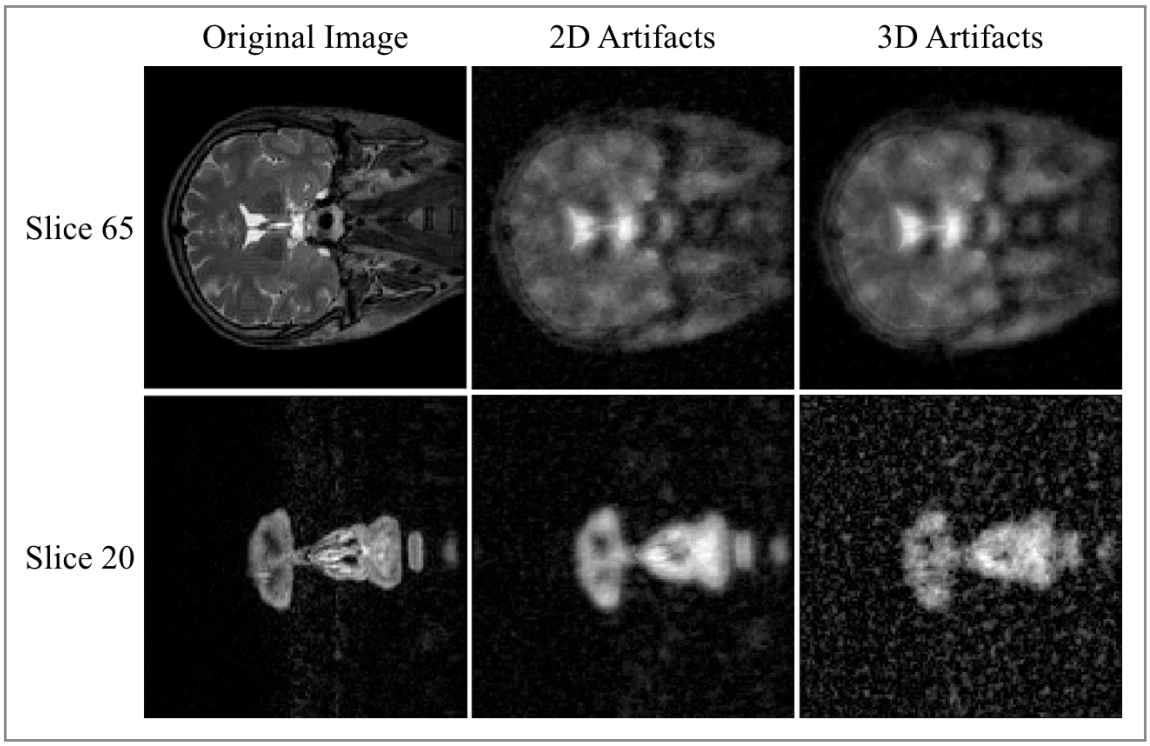

Figure 2.4. An Example of 3D Fourier Transform Effects: Slice 65 shows the 2D artifacts have more noise i.e. small blips on image and background, and also have less defined edges of the brain cross section. The reverse of this is true for slice 20 as the 3D artifacts distort early and end slices due to mixing of the brain image along the z-direction.

Figure 2.4 shows the effects of taking a masked Fourier Transform of a 3D brain image. For the purposes of this thesis, we choose to look at a middle slice 65, and an early slice 20. The original data, on the left, provides a clear picture of the slice. The center column represents the effects of taking a 2D Fourier Transform and processing the image slice by slice. Notice there is typical aliasing effects that occurs with subsampling in the k-space [43]. What is important to take away from Figure 2.4 are the 3D artifacts that appear in slice 20.

The 3D image near the beginning and later slices suffers from higher background noise levels. This is because, similar to our Goofy example [5], the 3D transform now considers image frequencies occurring in the third dimension or as we call it the tubal dimension. Thus, the background blips occurring in slice 20 in Figure 2.4, are due to the fact slices 30 through 100 contain fuller images of the brain and those frequencies are spilling over into the smaller cross-sections [19]. The implications for this as it relates to (1.1) means the 3D method has more initial total variation to remove from the early and later slices of our image. You will see the results of this and a recommended course of action in later Chapter 6. Next we will provide some background information on solution methods for solving (1.1).

CHAPTER 3

REVIEW OF NUMERICAL OPTIMIZATION TECHNIQUES

Before beginning this chapter, we will make note that we are going to discuss the background and derivation of the optimization methods used to solve

$$\min_x f(x) \text{ subject to } \|Ax - b\|_2 \leq \epsilon,$$

where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ and $f(x) : \mathbb{R}^n \mapsto \mathbb{R}$ is some convex objective function. We will discuss how these methods can be extended for the 2D and 3D MRI reconstruction applications in Chapters 4 and 5.

Included in this chapter is an explanation of standard convex optimization techniques in order to solve a minimization problem with quadratic inequality constraints. The techniques include the Conjugate Gradient (CG) method, the Newton method and the interior point method using the log-barrier function.

3.1 The Linear Conjugate Gradient Method

When we discuss the interior point method in Section 3.3, it will become apparent we have a large linear system, $Ax = b$, to solve, where A is symmetric positive-definite. Typically this is a large system and we don't know the exact solution. Therefore, we wish to find an x such that the residual

$$r = Ax - b \tag{3.1}$$

is less than some specified tolerance. According to [45, pp.101-111, pp.120-125], this is also equivalent to solving the following minimization problem

$$\min_x f(x) \text{ with } f(x) \coloneqq \frac{1}{2}x^T Ax - b^T x. \tag{3.2}$$

Observe that r is the gradient of f in (3.2), which means the opposite direction of r, the *negative gradient direction*, will be the fastest descent direction. When this is done iteratively we call this method the *steepest descent method*.

The CG method improves upon this approach by iteratively constructing a set of A-orthogonal search directions denoted by $\{p_{(0)}, p_{(1)}, ..., p_{(n)}\}$. To say the set of vectors, $\{p_{(0)}, p_{(1)}, ..., p_{(n)}\}$, are A-orthogonal means

$$p_i^T A p_j = 0, \quad \forall i \neq j.$$

The importance of this property comes from the fact that $f(x)$ can be minimized successively along the individual search directions, and yet each approximation is the optimal solution from the subspace spanned by all the search direction vectors up to that point [48]. This is written as

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} p_{(i)}, \tag{3.3}$$

where $\alpha_{(i)}$ is how far we wish to step along the direction $p_{(i)}$. This is equivalent to finding the minimum of $g(\alpha_{(i)})$, where

$$\begin{aligned} g(\alpha_{(i)}) &= f(x_{(i)} + \alpha_{(i)} p_{(i)}) \\ &= \frac{1}{2}(x_{(i)} + \alpha_{(i)} p_{(i)})^T A(x_{(i)} + \alpha_{(i)} p_{(i)}) - b^T(x_{(i)} + \alpha_{(i)} p_{(i)}) \\ &= \frac{1}{2}(x_{(i)}^T + \alpha_{(i)} p_{(i)}^T) A(x_{(i)} + \alpha_{(i)} p_{(i)}) - b^T x_{(i)} - \alpha_{(i)} b^T p_{(i)} \\ &= \frac{1}{2}(x_{(i)}^T A x_{(i)} + \alpha_{(i)} x_{(i)}^T A p_{(i)} + \alpha_{(i)} p_{(i)}^T A x_{(i)} + \alpha_{(i)}^2 p_{(i)}^T A p_{(i)}) - b^T x_{(i)} - \alpha_{(i)} b^T p_{(i)} \\ &= \frac{1}{2}\alpha_{(i)}^2 p_{(i)}^T A p_{(i)} + \alpha_{(i)} p_{(i)}^T (A x_{(i)} - b) + (\frac{1}{2} x_{(i)}^T A x_{(i)} - b^T x_{(i)}). \end{aligned}$$

The minimum occurs when $g'(\alpha_{(i)}) = 0$, i.e.

$$\alpha_{(i)} p_{(i)}^T A p_{(i)} + p_{(i)}^T (A x_{(i)} - b) = 0.$$

We can solve the above equation for $\alpha_{(i)}$ to get

$$\alpha_{(i)} = \frac{p_{(i)}^T r_{(i)}}{p_{(i)}^T A p_{(i)}}. \tag{3.4}$$

We now have a way to iteratively compute our approximations $x_{(i)}$. Notice we also now have an expression for the residuals given by

$$\begin{aligned} r_{(i)} &= A x_{(i)} - b \\ &= A(x_{(i-1)} + \alpha_{(i-1)} p_{(i-1)}) - b \quad \text{by (3.3)} \\ &= A x_{(i-1)} - b + \alpha_{(i-1)} A p_{(i-1)} \\ &= r_{(i-1)} + \alpha_{(i-1)} A p_{(i-1)}. \end{aligned} \tag{3.5}$$

Observe (3.3), (3.4), and (3.5) are dependent on search directions $p_{(i)}$. As stated before the opposite direction r will be the fastest descent direction. CG sets $p_{(0)} = -r_{(0)}$ and iteratively defines the next search direction as

$$p_{(i)} = -r_{(i)} + \beta_{(i)} p_{(i-1)}. \tag{3.6}$$

Thus $\alpha_{(i)}$ can be simplified to

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{p_{(i)}^T A p_{(i)}}, \tag{3.7}$$

and $\beta_{(i)}$ is given by [45, pp.101-111, pp.120-125] as

$$\beta_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i-1)}^T r_{(i-1)}}. \tag{3.8}$$

Thus, the CG method does not need to store the previous search directions, and in doing so reduces the complexity to $\mathcal{O}(km)$, where m is the number of nonzero entries of A and k is the number of CG steps [48]. Combining formulas (3.3), (3.5), (3.6), (3.7), and (3.8), we get the following CG iterations that continue until $\frac{\|r_{(i)}\|}{\|r_{(0)}\|}$ is less than some tolerance or the maximum number of iterations has been exceeded [25].

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{p_{(i)}^T A p_{(i)}}, \tag{3.9}$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} p_{(i)}, \tag{3.10}$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A p_{(i)}, \tag{3.11}$$

$$\beta_{(i)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}, \tag{3.12}$$

$$p_{(i+1)} = r_{(i+1)} + \beta_{(i)} p_{(i)}, \tag{3.13}$$

where $x_{(0)} = 0$ and $p_{(0)} = -r_{(0)} = b - A x_{(0)}$.

The expressions in (3.9) through (3.13) outline the iterative CG method that will be used within the Newton method to solve a system of the form $Ax = b$ where A is the Hessian matrix derived in the interior point method using the log-barrier function [35].

3.2 The Newton Method for Unonstrained Minimization

The Newton method is a well-known method for finding roots or zeros of functions. In this application it is used during each log-barrier iteration to iteratively solve the log-barrier form of the minimization problem (1.1) by forming a series of quadratic approximations. This form will be discussed in Section 3.3. So for now consider a real-valued function $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$. We are all familiar with the Taylor approximation of a single variable function, $g(y)$ near y, as follows

$$g(y + \Delta y) \approx g(y) + g^{'}(y)\Delta y + \frac{1}{2}g^{''}(y)\Delta y^2.$$

The analog of this for a multivariable function is

$$f(x + \Delta x) \approx f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x)\Delta x, \tag{3.14}$$

where ∇f represents the gradient and $\nabla^2 f$ represents the Hessian of our function [2]. Since we are seeking a Δx to minimize $f(x + \Delta x)$, we take the derivative of the right hand side of (3.14) and set it equal to zero

$$\begin{aligned} 0 &= \frac{d}{d\Delta x}\Big(f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x)\Delta x\Big) \\ &= \nabla f(x)^T + \Delta x^T \nabla^2 f(x). \end{aligned} \tag{3.15}$$

Since the Hessian, $\nabla^2 f(x)$, is positive definite, Equation (3.15) implies

$$\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x). \tag{3.16}$$

Equation (3.16) is called the Newton step. Thus we can iterate on x by $x_{n+1} = x_n + \Delta x_n$ where $\Delta x_n = -\nabla^2 f(x_n)^{-1} \nabla f(x_n)$. However, this formulation assumes we will step the full length of Δx_n. Often times in unconstrained minimization it is beneficial to control how far to step along the descent direction. A common way to control this is to decrease the step size until the objective function decreases in accordance with the expected change from the local gradient. To do this we add a parameter t and get the modified Newton iteration

$$x_{n+1} = x_n + t \cdot \Delta x_n. \tag{3.17}$$

Equation (3.17) now begs the question: when do we stop iterating? A typically stopping criterion involves the Newton decrement, denoted by $\lambda(x)$ [4]. It is expressed as

$$\lambda^2(x) = \left[\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right]^{\frac{1}{2}}. \tag{3.18}$$

A typical stopping criterion is given by

$$\frac{\lambda^2(x)}{2} \leq \epsilon, \tag{3.19}$$

where ϵ is a specified tolerance. We outline the Newton method in Algorithm 3.1.

Algorithm 3.1 The Newton method

Input: given $x \in \mathbf{dom}\ f$, tolerance $\epsilon > 0$.

Output: x that approximately minimizes the objective function.

1: $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$
2: $\lambda^2 = \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$
3: **while** $\frac{\lambda^2}{2} > \epsilon$ **do**
4: *Line Search:* Find appropriate t (see subsection 3.2.1)
5: $x = x + t\Delta x$
6: Update Δx and λ^2
7: **end while**

3.2.1 Backtracking Line Search

There are many line search methods used in practice to determine an appropriate t in line 4 of Algorithm 3.1. A popular effective method is the *Backtracking Line Search* (BLS). The idea behind the BLS is: initialize t to a unit step size, and continually reduce by a given factor until a stopping criterion is met. It is typical to specify a parameter $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$, where α is a stopping criterion and β is a reduction factor of t. Specifically, the BLS method is shown in Algorithm 3.2 [4].

Algorithm 3.2 Backtracking Line Search

Input: $\alpha \in (0, 0.5), \beta \in (0, 1)$, and descent direction Δx

Output: $t \in (0, 1]$.

1: $t = 1$
2: **while** $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$ **do**
3: $t = \beta t$
4: **end while**

To understand where the stopping criterion $f(x+t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$ comes from in Algorithm 3.2, consider the following. Notice $\nabla f(x)^T \Delta x < 0$ and $\alpha \in (0, 0.5)$ implies

$$\nabla f(x)^T \Delta x < \alpha \nabla f(x)^T \Delta x < 0.$$

Further,

$$\nabla f(x)^T \Delta x = \lim_{t \to 0} \frac{f(x+t\Delta x) - f(x)}{t}. \tag{3.20}$$

Thus $\exists\ \hat{t} > 0$ such that $\forall\ t \in (0, \hat{t})$

$$\begin{aligned} f(x+t\Delta x) &\approx f(x) + t\nabla f(x)^T \Delta x \\ &< f(x) + \alpha t \nabla f(x)^T \Delta x, \end{aligned} \tag{3.21}$$

since $\alpha \in (0, 0.5)$ and Δx is a descent direction. Since, $t = \beta^n$ for some $n \in \mathbb{N}$, and $\beta \in (0, 1)$, we see the line search will terminate in a finite number of steps. Algorithm 3.2 will be used along with the Newton method in Algorithm 3.1.

3.3 The Interior-Point Method using Log Barrier Function

We now return our attention to Problem (1.1). Recall we wish to solve a problem of the form

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to } & f_i(x) \leq 0 \text{ for } i = 1, ..., m. \end{aligned} \tag{3.22}$$

However, in the Newton method, discussed in Section 3.2, there are no constraints. This is where the log-barrier function enables us to take the inequality constraints in (3.22), and embedded them into the objective function, and then iteratively solve using the Newton method.

Consider the following indicator function $\mathscr{I} : \mathbb{R} \mapsto \mathbb{R}$,

$$\mathscr{I}(x) = \begin{cases} 0, & x \leq 0, \\ \infty, & x > 0. \end{cases} \tag{3.23}$$

One can envision changing (3.22) into the following

$$\min_x f_0(x) + \sum_{i=1}^{m} \mathscr{I}(f_i(x)) \tag{3.24}$$

Notice if an inequality constraint goes greater than 0 the indicator function, $\mathscr{I}(f_i(x))$, is infinity. Thus, the minimization problem is penalized for not meeting all inequality constraints which are now cast as part of the objective function itself.

There is still one issue with problem (3.24), that is the objective function is not differentiable which is required for the Newton method. Thus we need to approximate this function with a smooth and differentiable function. Consider the function

$$\hat{\mathscr{I}}(x) = \frac{-1}{\tau} \log(-x), \tag{3.25}$$

where $\mathbf{dom}\ \hat{\mathscr{I}} = \{x \in \mathbb{R} | x < 0\}$, and $\tau > 0$ is an accuracy parameter [4]. Clearly this is differentiable, but does it penalize the objective function similar to $\mathscr{I}(x)$? Let's look at some plots below for various values of τ.

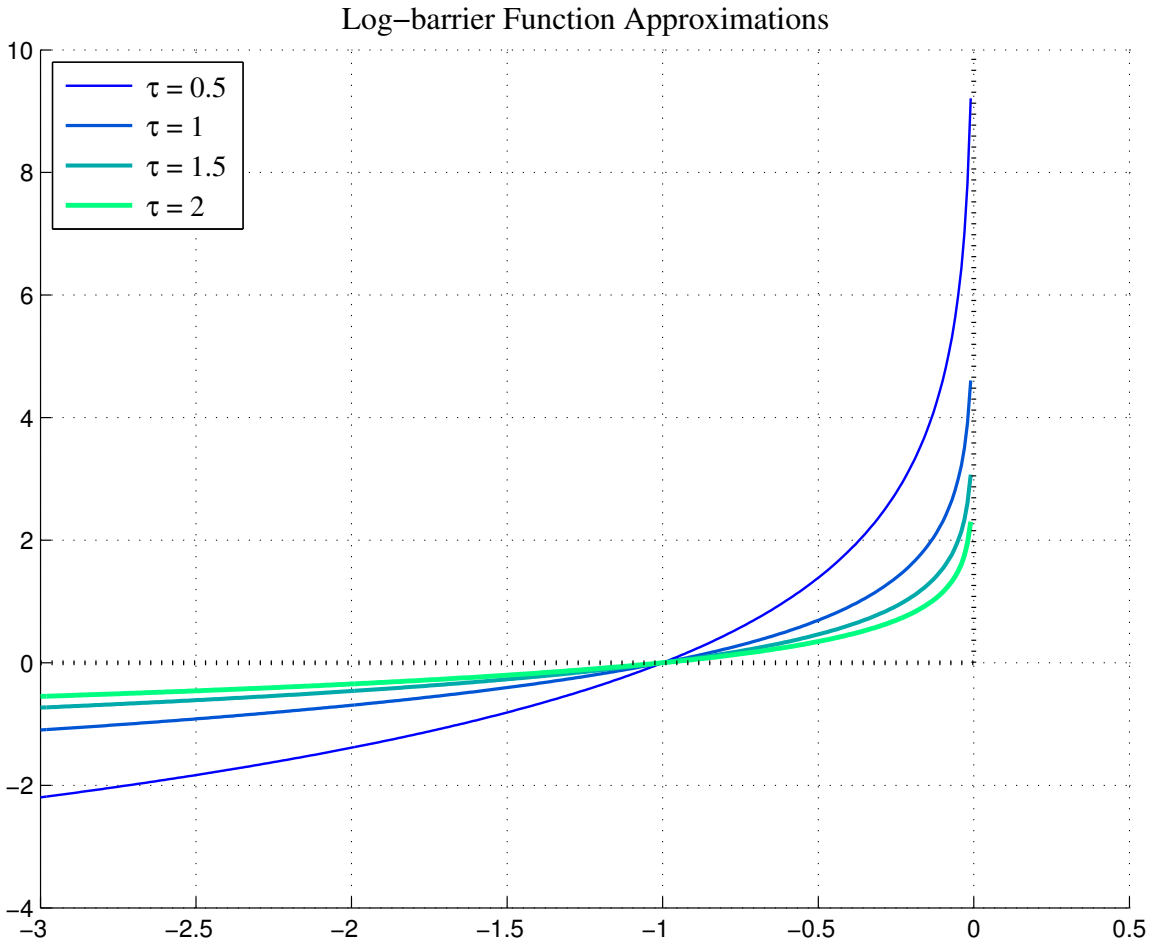

Figure 3.1. The Log-Barrier Function (3.25) for Various τ Values.

Figure 3.1 shows that for large values of τ, we can get close approximations of the indicator function $\mathscr{I}$. Using the fact that function (3.25) can be used to approximate the indicator function, we can now recast (3.22) as

$$\min_x f_0(x) + \frac{1}{\tau}\sum_{i=1}^{m} -\log\left(-f_i(x)\right) \tag{3.26}$$

Recall the Newton method wishes to find a change in descent direction that minimizes the objective function around a point x. So we write

$$f_0(x + \Delta x) \approx f(x) + \langle g_x, \Delta x\rangle + \frac{1}{2}\langle H_x \Delta x, \Delta x\rangle, \tag{3.27}$$

where g_x and H_x represent the gradient and Hessian of the original objective function, respectively. Specifically, we define

$$\phi(x) = -\sum_{i=1}^{m} \log\left(-f_i(x)\right) \tag{3.28}$$

to be the *log-barrier* function for (3.22). We also will note here for later reference, that the gradient, $\nabla\phi(x)$, and Hessian,$\nabla^2\phi(x)$, for the log-barrier function are

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{-f_i(x)} \nabla f_i(x) \tag{3.29a}$$

and

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{f_i(x)^2} \nabla f_i(x) \nabla f_i(x)^T + \sum_{i=1}^{m} \frac{1}{-f_i(x)} \nabla^2 f_i(x). \tag{3.29b}$$

Therefore, given our starting point x that is feasible, the direction Δx along which a new approximation to minimize (3.27) is the solution to the following linear system

$$H_x \Delta x = -g_x. \tag{3.30}$$

One might notice the objective function in (3.26) is only an approximation to the objective function in (3.22), whose results depend on the parameter τ. Thus, the larger τ gets, the more accurate the minimizer of (3.26) is as an approximation to (3.22). However, for the Newton method, a large τ impacts the Hessian as x approaches the boundary.

Since τ is a parameter that can change, it is logical to iterate. Thus, we have to solve

$$\min_x f_0(x) + \frac{1}{\tau}\phi(x) \tag{3.31}$$

for many $\tau > 0$ via the Newton method. The solution of (3.31), denoted by $x^*(\tau)$, is one point in a set of points defining the *central path*. Without going into too much detail, we will note the central path satisfies the KKT conditions of (3.31), thus guarantees our iterative solution will be optimal [4]. Often finding $x^*(\tau)$ is referred to as the *centering step*.

The pseudo code for the interior point method using the log-barrier function is shown in Algorithm 3.3.

Algorithm 3.3 Interior Point Method Using the Log-Barrier Function

Input: x strictly feasible, $\tau > 0, \mu > 0, \epsilon > 0$.

Output: minimizer $x^*(\mu^m\tau)$ for some $m > 0$.

1: **while** $m/\tau < \epsilon$ **do**
2: *Centering Step:* $x^*(\tau) = \arg\min_x f_0 + \frac{1}{\tau}\phi$
3: *Increase* $\tau : \tau = \mu\tau$
4: $m = m + 1$
5: **end while**

Line 2 in Algorithm 3.3 is solve by the Newton method with BLS given an initial starting point x_0.

CHAPTER 4

CURRENT STATE-OF-THE ART: 2D IMAGE RECOVERY

Now that we have reviewed some fundamental optimization techniques, in this chapter we discuss the current state-of-the-art in solving MRI reconstruction problems. Two popular approaches to solving the 2D minimization problem are proposed in [8] and [34].

In [34], the authors recast the problem into a 2D ℓ_1-regularized least squares problem, and solve the latter via the non-linear CG method [45, pp.101-111, pp.120-125]. In [8], the authors solve the 2D formulation of the total variation problem with quadratic constraints using the interior point method via the log-barrier function reviewed in Chapter 3.

We will briefly outline each method and show some results. More in-depth results and analysis of each method will be presented in Chapter 6 to compare performance against our 3DIRECT method. Recall our operator $\mathcal{A}$ is the sparse Fourier Transform discussed in Section 2.

4.1 SparseMRI: Minimizing the ℓ_1 norm

The method proposed in [34] is entitled *SparseMRI*, and it attempts to solve the following

$$\min_{X \in \mathbb{R}^{n \times n}} \|AX - B\|_2^2 + \lambda \|CX\|_1, \tag{4.1}$$

where $A \in \mathbb{R}^{n \times n}$, $C \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{n \times n}$, and $\lambda > 0$ is a preselected penalty parameter. Usually m and n are large but may not be necessarily the same. The formulation of this ℓ_1-regularized least squares problem lends itself nicely to signal reconstruction in

compressed sensing applications [29, pp.312-321]. The use of the $\|\cdot\|_2$ in (4.1) ensures the solution x is consistent with the measured data [3]. The use of the $\|\cdot\|_1$ ensures x is also sparse in a certain domain, for instance the k-space for MRI applications using wavelet transformations [10, 15, 60].

The norm $\|\cdot\|_1$ is similar to attempting to reduce the total variation of the image [16]. Thus, solutions to problem (4.1) will appear similar to solutions of (1.1). However, this norm is not differentiable. So, [34] approximates $\|\cdot\|_1$ before applying the nonlinear CG method to solve (4.1) [45, pp.101-111, pp.120-125].

In order to compute the recursive formulas for the nonlinear CG method, we need to calculate the gradient of $\|\cdot\|_1$. Unfortunately this gradient does not always exist, making any optimization method, e.g., those based on gradients, unusable. One common way to overcome the difficult is to smoothen out the norm. Consider

$$\gamma(z) = \|z\|_1 = \sum_{i=1}^{m} |z_i| \quad \text{for } z = [z_i] \in \mathbb{C}^m,$$

where z may be complex such as in SparseMRI. The partial gradient of $\gamma(z)$ with respect to z_i at $z_i = 0$ does not exist [7, 22]. The simple idea of smoothening it is as follows. Let μ be a tiny positive number (in the order of $O(10^{-6})$ or smaller), and approximate $|z_i|$ by

$$|z_i| \approx \sqrt{\bar{z}_i z_i + \mu}, \tag{4.2a}$$

$$\gamma(z) \approx g(z) := \sum_{i=1}^{m} \sqrt{\bar{z}_i z_i + \mu}, \tag{4.2b}$$

where $\bar{z}_i$ is its complex conjugate. Suppose now z_i is perturbed to $z_i + p$, where p is an infinitesimal quantity. We have, up to the first order of p,

$$\begin{aligned}(\bar{z}_i + \bar{p})(z_i + p) + \mu &= (\bar{z}_i z_i + \mu)\left[1 + \frac{\bar{z}_i p + \bar{p} z_i}{\bar{z}_i z_i + \mu}\right], \\ \sqrt{(\bar{z}_i + \bar{p})(z_i + p) + \mu} &= \sqrt{\bar{z}_i z_i + \mu}\left[1 + \frac{1}{2}\frac{\bar{z}_i p + \bar{p} z_i}{\bar{z}_i z_i + \mu}\right] \\ &= \sqrt{\bar{z}_i z_i + \mu} + \frac{1}{2}\frac{\bar{z}_i p + \bar{p} z_i}{\sqrt{\bar{z}_i z_i + \mu}},\end{aligned}$$

yielding the partial gradient with respect to z_i as $\frac{z_i}{\sqrt{\bar{z}_i z_i + \mu}}$ and consequently

$$\nabla g(z) = D^{-1} z \quad \text{with} \quad D = \text{diag}(\sqrt{\bar{z}_1 z_1 + \mu}, \ldots, \sqrt{\bar{z}_m z_m + \mu}\,).$$

$\nabla g(z)$ is the direction along which $g(z)$ increases the fastest at z.

We now return our attention to $\|Cx\|_1$ in (4.1). Denote the i^{th} component of Cx by $(Cx)_i$. Let μ be as before, and use

$$|(Cx)_i| \approx \sqrt{(Cx)_i^H (Cx)_i + \mu}, \quad \|Cx\|_1 \approx g(x) := \sum_i \sqrt{(Cx)_i^H (Cx)_i + \mu}.$$

Suppose now x is perturbed to $x + p$, where p is an infinitesimal vector. We have, up to the first order of p,

$$\sqrt{(C[x+p])_i^H (C[x+p])_i + \mu} \approx \sqrt{(Cx)_i^H (Cx)_i + \mu} + \frac{1}{2}\frac{(Cx)_i^H (Cp)_i + (Cp)_i^H (Cx)_i}{\sqrt{(Cx)_i^H (Cx)_i + \mu}}.$$

Therefore

$$g(x+p) \approx g(x) + \sum_i \frac{1}{2}\frac{(Cx)_i^H (Cp)_i + (Cp)_i^H (Cx)_i}{\sqrt{(Cx)_i^H (Cx)_i + \mu}}.$$

Use $(Cp)_i = \sum_j c_{ij}p_j$ and let $d_i = \sqrt{(Cx)_i^H(Cx)_i + \mu}$ to get

$$g(x+p) \approx g(x) + \sum_j \frac{1}{2} \sum_i [(Cx)_i^H d_i^{-1} c_{ij} p_j + \bar{p}_j \bar{c}_{ij} d_i^{-1} (Cx)_i],$$

giving the j^{th} component of $\nabla g(x)$ as

$$\sum_i \bar{c}_{ij} d_i^{-1} (Cx)_i = (C^H D^{-1} Cx)_j.$$

Therefore

$$\nabla g(x) = C^H D^{-1} Cx, \tag{4.3}$$

where $D = \text{diag}(\sqrt{(Cx)_1^H(Cx)_1 + \mu}, \ldots, \sqrt{(Cx)_m^H(Cx)_m + \mu})$. This gives the following,

$$\nabla f(x) \approx 2A^H(Ax - b) + \lambda C^H D^{-1} Cx.$$

Similarly we evaluate $f(x)$ using the approximation in (4.2a) as

$$f(x) \approx (Ax-b)^H(Ax-b) + \lambda |||Cx|||,$$

where $|||a||| = \sum_i \sqrt{\bar{a}_i a_i + \mu}$. Putting it all together, Algorithm 4.1 solves the nonlinear ℓ_1-regularized least squares problem (4.1).

Algorithm 4.1 Nonlinear CG with Backtracking Line Search for (4.1)

Input: measured data b, consistency matrix A, sparsity matrix C, ℓ_1 penalty λ, initial guess x.

Output: solution x to least squares problem (4.1)

Specify Parameters

$k = 0$, $\mu = 10^{-6}$, $tol = 10^{-9}$, $M = 100$, $\alpha = 0.5$, $\beta = 0.6$, $t_0 = 1$

Compute Initial Gradient

$g_0 = 2A^H(Ax - b) + \lambda \cdot C^H D^{-1} Cx, \quad dx = -g_0$

Iterations

while $\|g_0\|_2 > tol$ and $k < M$ **do**

  Backtracking Line Search

  $f_0 = \|Ax - b\|_2^2 + \lambda |||Cx|||$

  $t = t_0, \qquad f_1 = \|A(x + t \cdot dx) - b\|_2^2 + \lambda |||C(x + t \cdot dx)|||$

  $lsIter = 0$

  while $f_1 > f_0 + \alpha t \cdot \text{real}(g_0^H dx)$ **do**

    $lsIter = lsIter + 1$

    $t = \beta \cdot t, \qquad f_1 = \|A(x + t \cdot dx) - b\|_2^2 + \lambda |||C(x + t \cdot dx)|||$

  end while

  Update nonlinear CG Recursive Formulas

  $x = x + t \cdot dx$

  $g_1 = 2A^H(Ax - b) + \lambda \cdot C^H D^{-1} Cx, \qquad \gamma = \frac{\|g_1\|_2}{\|g_0\|_2}$

  $dx = -g_1 + \gamma dx, \qquad g_0 = g_1, \qquad k = k + 1$

end while

Note that $k, \mu, tol, M, \alpha, \beta,$ and t_0 are shown as fixed parameters, however these can be passed in as inputs. In addition, the maximum iterations for the line search and the CG method can be different. Algorithm 4.1 is just a simplified version.

To show how the method in [34] recovers images from sparsely sampled images in the k-space, we will consider a 2D example of a 512×512 image [31, 52]. The image is subsampled using a 2D Fourier Transform mask that keeps entire columns of

Fourier data. The sparse domain is a Daubechies 4 Wavelet transform [6]. Figure 4.1 shows the ability of Algorithm 4.1 to clear up the artifacts from subsampling.

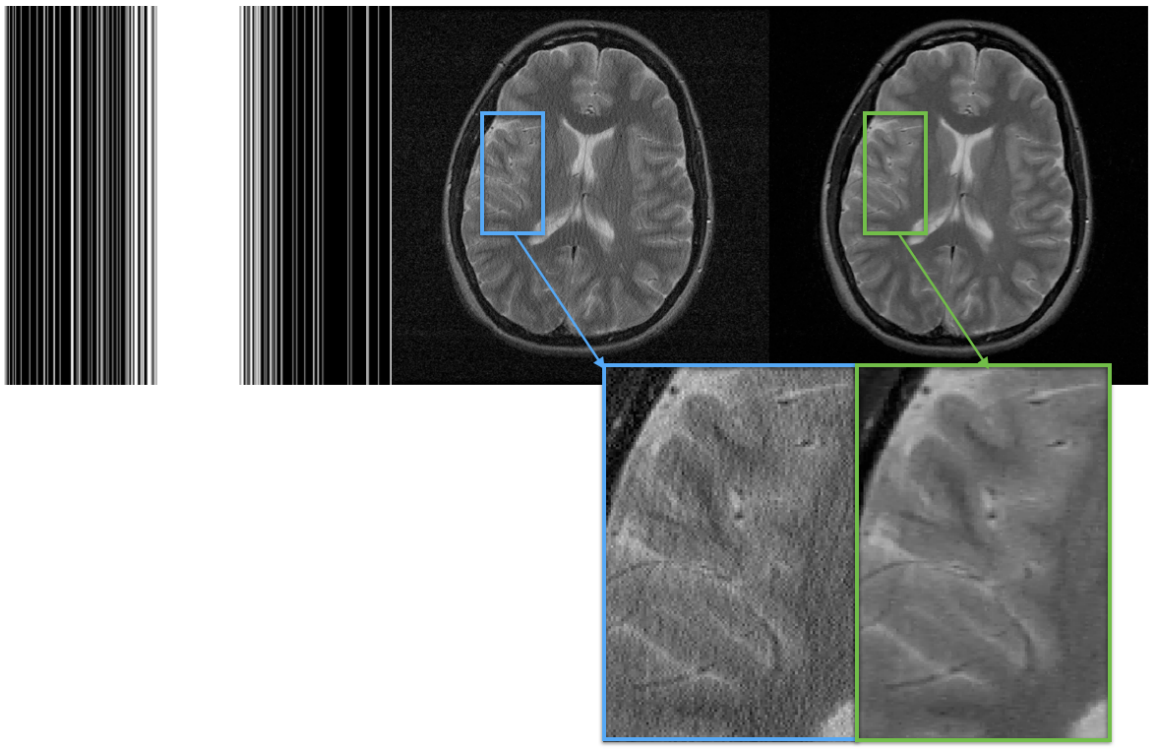

Figure 4.1. The Recovered Image is Sharpe. Left: Mask; Center: Input Image; Right: Recovered Image .

On the left in Figure 4.1 is the 2D Fourier Mask where the white indicates the data kept, and the black indicates the Fourier samples that are thrown away. As a result of the subsampling scheme, the center image appears with noisy artifacts. Finally, the image is cleaned up using the nonlinear CG method, and as a result the noise is removed. This is just an example of how the algorithm works to denoise an image by minimizing over the-ℓ_1 norm. The mask sampling precentage was 40% which is rather a large amount of data to sample and store especially when capturing

3D data. We will see further impacts of this when looking at the numerical results in Chapter 6.

4.2 ℓ_1-Magic: Total Variation with Quadratic Constraints

We now turn our attention to the approach discussed in [8] which is called ℓ_1-*Magic:* TV_2. For the purposes of this thesis we will refer to the method simply as ℓ_1-Magic. This approach considers the following 2D problem

$$\min_X \text{TV}(X) \text{ subject to } \|\mathcal{A}(X) - B\|_2 \leq \epsilon, \tag{4.4}$$

where $\mathcal{A} : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is the 2D Sparse Fourier Transform as describe in Chapter 2, $B \in \mathbb{R}^{n\times n}$ is the observed image pixel in the k-space, and TV is a function that computes the total variation in the $x-$ and $y-$direction.

In order to use the Newton method we must define the TV function in (4.4). In [47], the Total Variation of a 2D discrete image is defined as the sum of the finite differences of pixels with their vertical and horizontal neighboring pixels [37, 49].

To define this mathematically, we will consider a two-dimensional image $X \in \mathbb{R}^{n\times n}$, whose ij^{th} pixel will be denoted as X_{ij}, and define the following operators

$$\mathscr{D}_{h;ij}X = \begin{cases} X_{i,j+1} - X_{ij}, & i < n, \\ 0, & i = n, \end{cases} \qquad \mathscr{D}_{v;ij}X = \begin{cases} X_{i+1,j} - X_{ij}, & j < n, \\ 0, & j = n, \end{cases}$$

where $\mathscr{D}_{h;ij}$ and $\mathscr{D}_{v;ij}$ can be thought of as the horizontal and vertical finite difference operators. We further define the vector $\mathscr{D}_{ij}X \in \mathbb{R}^2$ as

$$\mathscr{D}_{ij}X = \begin{bmatrix} \mathscr{D}_{h;ij}X \\ \mathscr{D}_{v;ij}X \end{bmatrix}. \tag{4.5}$$

This vector, $\mathscr{D}_{ij}X$, serves as a discrete gradient of our image at the ij^{th} pixel, and thus the total variation in (4.4) is defined as follows

$$\mathrm{TV}(X) = \sum_{ij} \sqrt{(\mathscr{D}_{h;ij}X)^2 + (\mathscr{D}_{v;ij}X)^2} = \sum_{ij} \|\mathscr{D}_{ij}X\|_2. \tag{4.6}$$

4.2.1 The Interior Point Method for SOCPs

Before applying the log-barrier function to transform (4.4), we will first recast it as a second-order cone problem [8, 33]. Using (4.6), (4.4) can be recast as

$$\begin{aligned} \min_{X,T} \quad & \sum_{ij} T_{ij} \\ \text{subject to} \quad & \|\mathscr{D}_{ij}X\|_2 \leq T_{i,j}, i, j = 1, \ldots, n, \\ & \|\mathcal{A}(X) - B\|_2 \leq \epsilon. \end{aligned} \tag{4.7}$$

It is important to notice the problem now has two parameters to optimize over, T and X, where $T = [T_{ij}]$ is an $n \times n$ matrix. So, now we have two inequality constraints which need to be embedded into the objective function via the log-barrier function. In order to do this we will define

$$f_{T_{ij}} = \frac{1}{2}(\|\mathscr{D}_{ij}X\|_2^2 - T_{ij}^2), \quad i, j = 1, \ldots, n, \tag{4.8a}$$

$$f_\epsilon = \frac{1}{2}(\|\mathcal{A}(X) - B\|_2^2 - \epsilon^2). \tag{4.8b}$$

Note the factor $\frac{1}{2}$ in (4.8a) and (4.8b) are there to cancel out constants that appear later on when we calculate the gradient. We can now apply the log-barrier function from (3.28), and make the inequality constraints implicit in the objective as follows

$$F(z) \coloneqq \langle c_0, z\rangle - \frac{1}{\tau}\Big[\sum_{ij} \log(-f_{T_{ij}}(z)) + \log(-f_\epsilon(z))\Big], \tag{4.9}$$

where $\tau > 0$ is an accuracy parameter, $c_0 = \begin{bmatrix} 0_{n^2} \\ 1_{n^2} \end{bmatrix}$ and $z = \begin{bmatrix} x \\ t \end{bmatrix}$, and

$$x \coloneqq X(:) = \begin{bmatrix} X_{11} \\ \vdots \\ X_{n1} \\ \vdots \\ X_{1n} \\ \vdots \\ X_{nn} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{n^2} \end{bmatrix}, \text{ and } t \coloneqq T(:) = \begin{bmatrix} T_{11} \\ \vdots \\ T_{n1} \\ \vdots \\ T_{1n} \\ \vdots \\ T_{nn} \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ t_{n^2} \end{bmatrix}. \tag{4.10}$$

We use 0_{n^2} and 1_{n^2} to denote column vectors of 0's and 1's of length n^2, respectively. Next, we need to compute the gradient and Hessian of the function (4.9). Using the gradient of the log-barrier function from (3.29a), the gradient of $F(z)$ can be expressed as

$$g_z = c_0 + \frac{1}{\tau}\Big[\sum_{ij} \frac{1}{-f_{T_{ij}}(z)} \nabla f_{T_{ij}}(z) + \frac{1}{-f_\epsilon(z)} \nabla f_\epsilon(z)\Big], \tag{4.11}$$

and the Hessian by

$$\begin{aligned} H_z = \frac{1}{\tau}\Big[\sum_{ij} \frac{1}{f_{T_{ij}}(z)^2} \nabla f_{T_{ij}}(z)[\nabla f_{T_{ij}}(z)]^T + \sum_{ij} \frac{1}{-f_{T_{ij}}(z)} \nabla^2 f_{T_{ij}}(z) \\ + \frac{1}{f_\epsilon(z)^2} \nabla f_\epsilon(z)[\nabla f_\epsilon(z)]^T + \frac{1}{-f_\epsilon(z)} \nabla^2 f_\epsilon(z)\Big]. \end{aligned} \tag{4.12}$$

Again if we assume a feasible z, the direction Δz along which $F(z)$ is to be approximately minimized, is the solution to the following system of linear equations

$$H_z \Delta z = -g_z. \tag{4.13}$$

This linear system is reduced and then solved by the linear CG method. Applying this to a 512×512 brain image using a radial sampling scheme of 10% results in Figure 4.2.

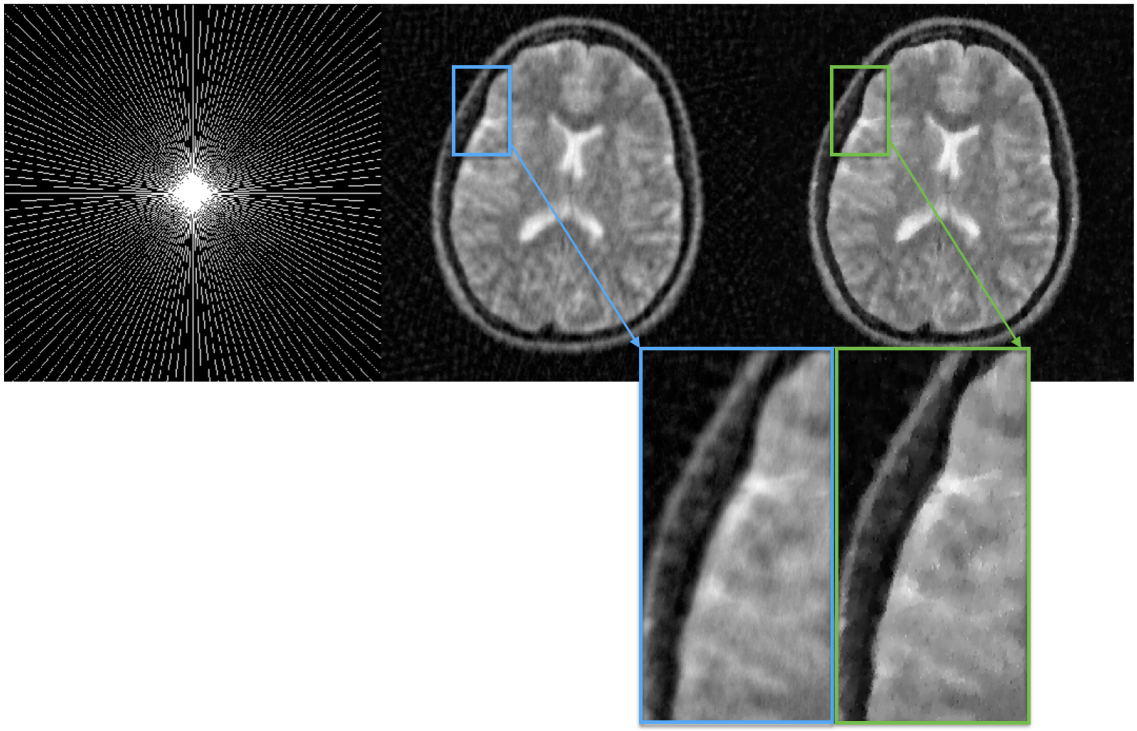

Figure 4.2. The Recovered Image is More Clear. Left: Mask; Center: Input Image; Right: Recovered Image.

Figure 4.2 shows the artifacts from subsampling have been reduced by minimizing over the total variation. Another observation is the sampling scheme used is

radial lines as oppose to columns of the Fourier domain. As stated in Chapter 2, the type of mask can improve or hurt results. The mask is dependent on the transformation used, $\mathcal{A}$, and the data set being operated on. This will become clear in the Chapter 6.

CHAPTER 5

3D IMAGE RECOVERY

Now that we understand the current state-of-the-art in 2D imaging, we will examine how to handle a 3D image all at once as oppose to running 2D methods over each slice of the data one at a time. We call our method 3DIRECT. Consider the following problem

$$\min_X \mathrm{TV}(X) \text{ subject to } \|\mathcal{A}(X) - B\|_2 \leq \epsilon, \tag{5.1}$$

where TV is a function that computes the 3D Total Variation of an image $X \in \mathbb{R}^{n\times n\times n}$, and $\mathcal{A} : \mathbb{R}^{n\times n\times n} \mapsto \mathbb{R}^{n\times n\times n}$ is again a linear map. For the purposes of implementation it is more precisely the subsampled 3D Fourier Transformation [21, 54].

Similarly to the 2D ℓ_1-Magic method, we will explicitly define a function TV : $\mathbb{R}^{n\times n\times n} \mapsto \mathbb{R}$. Consider now a 3D $n \times n \times n$ image X, whose ijk^{th} pixel will be denoted as X_{ijk}, and define the following operators

$$\mathscr{D}_{h;ijk}X = \begin{cases} X_{i(j+1)k} - X_{ijk}, & i < n, \\ 0, & i = n, \end{cases} \qquad \mathscr{D}_{v;ijk}X = \begin{cases} X_{(i+1)jk} - X_{ijk}, & j < n, \\ 0, & j = n, \end{cases}$$

$$\mathscr{D}_{t;ijk}X = \begin{cases} X_{ij(k+1)} - X_{ijk}, & k < n, \\ 0, & k = n, \end{cases}$$

where now we have a third finite difference operator, $\mathscr{D}_{t;ijk}$. This operator considers the pixel variation from slice to slice down what we call a *tube*. Hence $\mathscr{D}_{t;ijk}$ is the *tubal operator*. Further, we denote a vector $\mathscr{D}_{ijk}X \in \mathbb{R}^{3\times 1}$ as

$$\mathscr{D}_{ijk}X = \begin{bmatrix} \mathscr{D}_{h;ijk}X \\ \mathscr{D}_{v;ijk}X \\ \mathscr{D}_{t;ijk}X \end{bmatrix}. \tag{5.2}$$

This vector, $\mathscr{D}_{ijk}X$, serves as a 3D discrete gradient of the image at the ijk^{th} pixel, and thus the total variation in (5.1) can be expressed as

$$\mathrm{TV}(X) = \sum_{ijk} \sqrt{(\mathscr{D}_{h;ijk}X)^2 + (\mathscr{D}_{v;ijk}X)^2 + (\mathscr{D}_{t;ijk}X)^2} = \sum_{ijk} \|\mathscr{D}_{ijk}X\|_2. \tag{5.3}$$

5.1 The Interior Point Method for SOCPs

We will first use (5.3) to recast problem (5.1) as a SOCP

$$\begin{aligned} \min_{X,T} \quad & \sum_{ijk} T_{ijk} \\ \text{subject to} \quad & \|\mathscr{D}_{ijk}X\|_2 \le T_{ijk}, \quad i,j,k = 1,\dots,n, \\ & \|\mathcal{A}(X) - B\|_2 \le \epsilon. \end{aligned} \tag{5.4}$$

We now stretch X and T into vectors columwise within a slice, and then each stretched slice is stacked in order from first to last slice, i.e., arrange their entires, X_{ijk} and T_{ijk}, in the *lexicographical order* as follows

$$x \coloneqq X(:) = \begin{bmatrix} X_{111} \\ \vdots \\ X_{11k} \\ \vdots \\ X_{n1k} \\ \vdots \\ X_{1nk} \\ \vdots \\ X_{nnk} \\ \vdots \\ X_{nnn} \end{bmatrix}, \text{ and } t \coloneqq T(:) = \begin{bmatrix} T_{111} \\ \vdots \\ T_{11k} \\ \vdots \\ T_{n1k} \\ \vdots \\ T_{1nk} \\ \vdots \\ T_{nnk} \\ \vdots \\ T_{nnn} \end{bmatrix}. \tag{5.5}$$

Before embedding the inequality constraints via a log-barrier function, we will define

$$f_{T_{ijk}} = \frac{1}{2}(\|\mathscr{D}_{ijk}X\|_2^2 - T_{ijk}^2), \quad i,j,k = 1,\ldots,n, \tag{5.6a}$$

$$f_\epsilon = \frac{1}{2}(\|\mathcal{A}(X) - B\|_2^2 - \epsilon^2). \tag{5.6b}$$

We can now apply the log-barrier function from (3.28), and make the inequality constraints implicit in the objective function using (5.6a) and (5.6b) as follows

$$F(z) \coloneqq \langle c_0, z\rangle - \frac{1}{\tau}\Big[\sum_{ijk}\log(-f_{T_{ijk}}(z)) + \log(-f_\epsilon(z))\Big], \tag{5.7}$$

where again $\tau > 0$ is an accuracy parameter and $c_0 = \begin{bmatrix} 0_{n^3} \\ 1_{n^3} \end{bmatrix}$, $z = \begin{bmatrix} x \\ t \end{bmatrix}$. We use 0_{n^3} and 1_{n^3} to denote column vectors of 0's and 1's of length n^3, respectively. Next, we express the gradient, g_z, and Hessian, H_z, for (5.7) as

$$g_z = c_0 + \frac{1}{\tau}\Big[\sum_{ijk} \frac{1}{-f_{T_{ijk}}(z)} \nabla f_{T_{ijk}}(z) + \frac{1}{-f_\epsilon(z)} \nabla f_\epsilon(z)\Big], \tag{5.8}$$

and

$$\begin{aligned} H_z = \frac{1}{\tau}\Big[\sum_{ijk} \frac{1}{f_{T_{ijk}}(z)^2} \nabla f_{T_{ijk}}(z)[\nabla f_{T_{ijk}}(z)]^T + \sum_{ijk} \frac{1}{-f_{T_{ijk}}(z)} \nabla^2 f_{T_{ijk}}(z) \\ + \frac{1}{f_\epsilon(z)^2} \nabla f_\epsilon(z)[\nabla f_\epsilon(z)]^T + \frac{1}{-f_\epsilon(z)} \nabla^2 f_\epsilon(z)\Big]. \end{aligned} \tag{5.9}$$

Thus, if we assume a feasible z, the direction Δz along which $F(z)$ is to be approximately minimized is the solution to the following system of linear equations

$$H_z \Delta z = -g_z. \tag{5.10}$$

The complete interior point method using the log-barrier function is outline in Algorithm 5.1.

Algorithm 5.1 Interior Point Method Using the Log-Barrier Function

Input: feasible starting point $z_0, \tau_0 > 0, \mu > 0, \epsilon > 0, n_{max} > 0$.

Output: minimizer z_k.

$k = 0, \tau = \tau_0$
while $m/\tau < \epsilon$ **do**
  Find $z_k = \min_z F(z)$ by the Newton Method, where $F(z)$ is defined in (5.7) as follows:
  $n_{stop} = 0, k = 0$
  while $n_{stop} = 0$ **do**
    Compute g_z and create operator $H_z(z)$
    Find Newton Step Δz by using CG to solve: $H_z \Delta z = -g_z$
    $\lambda^2 = g_z^T H_z^{-1} g_z$
    BLS: Find appropriate t (see subsection 3.2.1)
    $z_k = z_k + t\Delta z$
    $k = k + 1$
    $n_{stop} = (\frac{\lambda^2}{2} < \epsilon)$ or $(k > n_{max})$
  end while
  $\tau = \mu\tau$
  $k = k + 1$
end while

In Sections 5.2 and 5.3 we provide detail on how to express g_z and H_z in order to efficiently implement the 3DIRECT method.

5.2 g_z

This section provides details on computing the gradient in (5.8). Recall our optimization variable z is comprised of x and t. Thus we have two partial derivatives to concern ourselves with when computing $\sum_{ijk} \frac{1}{-f_{T_{ijk}}(z)} \nabla f_{T_{ijk}}(z)$ and $\frac{1}{-f_\epsilon(z)} \nabla f_\epsilon(z)$.

First notice X is organized by slices $X(:,:,k)$ for $k = 1, \cdots, n$, and the $\mathscr{D}_{h;ijk}$ and $\mathscr{D}_{v;ijk}$ operate within a slice. Therefore, it suffices to analyze the behavior of these operators within a single 2D slice, and then extend it to three dimensions.

For a fixed slice $\hat{k} \in [1, n]$, we first define the row vectors $D^{(\hat{k})}_{h;ij}, D^{(\hat{k})}_{v;ij} \in \mathbb{R}^{1\times n^2}$ such that

$$D^{(\hat{k})}_{h;ij} x_{\hat{k}} \coloneqq \mathscr{D}_{h;ij\hat{k}} X, \tag{5.11a}$$

$$D^{(\hat{k})}_{v;ij} x_{\hat{k}} \coloneqq \mathscr{D}_{v;ij\hat{k}} X, \tag{5.11b}$$

where $x_{\hat{k}} \in \mathbb{R}^{n^2}$ represent the image vector for fixed slice $\hat{k}$. We can now express $\hat{f}_{T_{ij}}$ for a fixed slice $\hat{k}$ as

$$\begin{aligned} \hat{f}_{T_{ij}} &= \frac{1}{2}\left(\left\| \begin{bmatrix} D^{(\hat{k})}_{h;ij} \\ D^{(\hat{k})}_{v;ij} \end{bmatrix} x_{\hat{k}} \right\|_2^2 - T^2_{ij\hat{k}}\right), \\ &= \frac{1}{2}(\|\hat{D}_{ij} x_{\hat{k}}\|_2^2 - T^2_{ij\hat{k}}), \quad i, j = 1, \ldots, n, \end{aligned} \tag{5.12}$$

where $\begin{bmatrix} D^{(\hat{k})}_{v;ij} \\ D^{(\hat{k})}_{v;ij} \end{bmatrix} = \hat{D}_{ij}$. Note, $\hat{D}_{ij}$ has a dependence on $\hat{k}$, but for readability will leave off this distinction. Expanding (5.12) gives

$$\begin{aligned} f_{T_{ij\hat{k}}} &= \frac{1}{2}(\|\hat{D}_{ij} x_{\hat{k}}\|_2^2 - T^2_{ij\hat{k}}), \\ &= \frac{1}{2}((\hat{D}_{ij} x_{\hat{k}})^T (\hat{D}_{ij} x_{\hat{k}}) - T^2_{ij\hat{k}}), \\ &= \frac{1}{2}(x^T_{\hat{k}} \hat{D}^T_{ij} \hat{D}_{ij} x_{\hat{k}} - T^2_{ij\hat{k}}), \quad i, j = 1, \ldots, n. \end{aligned} \tag{5.13}$$

Recall we wish to find $\sum_{ij\hat{k}} \frac{1}{\hat{f}_{T_{ij}}(z)} \nabla \hat{f}_{T_{ij}}(z)$ for fixed slice $\hat{k}$. Before computing this we will define a new function $\psi(y) = y^T M y$ for some square matrix M and vector y. We can compute the gradient as follows

$$\begin{aligned}
\frac{\partial \psi}{\partial y_k} &= \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij} \frac{\partial (y_i y_j)}{y_k} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}(\delta_{ik} y_j + y_i \delta_{jk}) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}\delta_{ik} y_j + \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij} y_i \delta_{jk} \\
&= \sum_{j=1}^{n} m_{kj} y_j + \sum_{i=1}^{n} m_{ik} y_i \\
&= (My)_k + (M^T y)_k \\
&\Rightarrow \nabla \psi(y) = (M + M^T) y, \qquad (5.14)
\end{aligned}$$

where δ_{ij} is the Dirac δ-symbol, i.e. $\delta_{ij} = 1$ for $i = j$ and 0 otherwise. Thus, using (5.14) and $M = \hat{D}_{ij}^T \hat{D}_{ij}$ we can write

$$\begin{aligned}
\sum_{ij} \frac{1}{\hat{f}_{T_{ij}}(z)} \nabla \hat{f}_{T_{ij}}(z) &= \sum_{ij} \frac{1}{\hat{f}_{T_{ij}}(z)} \begin{bmatrix} \hat{D}_{ij}^T \hat{D}_{ij} x_{\hat{k}} \\ -T_{ij\hat{k}} \hat{\delta}_{ij} \end{bmatrix}, \\
&= \begin{bmatrix} \hat{D}_h^T \hat{F}_t^{-1} \hat{D}_h x_{\hat{k}} + \hat{D}_v^T \hat{F}_t^{-1} \hat{D}_v x_{\hat{k}} \\ -\hat{F}_t^{-1} t \end{bmatrix},
\end{aligned} \qquad (5.15)$$

where $\hat{\delta}_{ij} \in \mathbb{R}^{n^2}$ is a vector that is 1 in the ij^{th} entry and zero elsewhere, and $\hat{F}_t^{-1} = \texttt{diag}(1 \oslash \hat{f}_T)$, where $\hat{f}_T \in \mathbb{R}^{n^2}$ contains the $f_{T_{ij\hat{k}}}$ elements listed in the lexicographical order according to their $ij\hat{k}$ indices. The $\oslash$ notation is used to indicate that element-wise division, i.e. given vectors x and y, $(x \oslash y)_i = \frac{x_i}{y_i}$.

Lastly, $\hat{D}_h, \hat{D}_v \in \mathbb{R}^{n^2 \times n^2}$ are comprised of the row vectors $D_{v;ij}^{(\hat{k})}$ and $D_{v;ij}^{(\hat{k})}$, for $i, j = 1, \cdots, n$, also listed in lexicographical order. Based on this ordering, and the definition of the vectors in (5.18a) and (5.18b), we can express

$$\hat{D}_h = \hat{B} \otimes I_n, \tag{5.16a}$$

$$\hat{D}_v = I_n \otimes \hat{B}, \tag{5.16b}$$

where

$$\hat{B} = \begin{bmatrix} -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & & -1 & 1 \\ & & & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{5.17}$$

and $\otimes$ denotes the *Kronecker product*. We are now ready to expand our analysis to the tubal direction. We define the row vectors $D_{h;ijk}, D_{v;ijk}, D_{t;ijk} \in \mathbb{R}^{1 \times n^3}$ such that

$$D_{h;ijk}x \coloneqq \mathscr{D}_{h;ijk}X, \tag{5.18a}$$

$$D_{v;ijk}x \coloneqq \mathscr{D}_{v;ijk}X, \tag{5.18b}$$

$$D_{t;ijk}x \coloneqq \mathscr{D}_{t;ijk}X. \tag{5.18c}$$

Using the results from (5.16a) and (5.16b), we define

$$D_h = \begin{bmatrix} \hat{D}_h & & & \\ & \hat{D}_h & & \\ & & \ddots & \\ & & & \hat{D}_h \end{bmatrix} = I_n \otimes \hat{D}_h, \quad D_v = \begin{bmatrix} \hat{D}_v & & & \\ & \hat{D}_v & & \\ & & \ddots & \\ & & & \hat{D}_v \end{bmatrix} = I_n \otimes \hat{D}_v, \tag{5.19}$$

where $D_h, D_v \in \mathbb{R}^{n^3 \times n^3}$. We are able to do this because the horizontal and vertical finite differences do not interact between slices. Therefore, all we need now is to define the $n^3 \times n^3$ matrix D_t that computes the tubal finite differences. Since these are the differences of the same X_{ijk} pixel from slice to slice, we can write D_t as

$$D_t = \hat{B} \otimes I_{n^2}, \tag{5.20}$$

where B is defined in (5.17). By similar analysis for a fixed slice $\hat{k}$, we can now write

$$\sum_{ijk} \frac{1}{f_{T_{ijk}}(z)} \nabla f_{T_{ijk}}(z) = \begin{bmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x \\ -F_t^{-1} t \end{bmatrix}. \tag{5.21}$$

where $F_t^{-1} = \texttt{diag}(1 \oslash f_T)$, $f_T \in \mathbb{R}^{n^3}$ contains the elements of $f_{T_{ijk}}$, and we are highlighting the impacts of the tubal difference operator in blue color. This is done to contrast the differences from the 2D case, and will continue this convention in subsequent chapters.

We now turn our attention to computing $\frac{1}{f_\epsilon(z)} \nabla f_\epsilon(z)$ in (5.8). For ease of implementation we will note that, $\mathcal{A}(X)$ is equivalent to a matrix-vector multiplication, Ax, if we formally write $Ax \coloneqq (\mathcal{A}(X))(:)$. Similarly we can compare this resulting vector, Ax to the observations in vector $b \coloneqq B(:)$. Using this implementation and the fact that f_ϵ only depends on x, we will first expand the equivalent f_ϵ expression as

$$\begin{aligned} \|Ax - b\|_2^2 - \epsilon &= (Ax - b)^T (Ax - b) - \epsilon \\ &= x^T A^T A x - x^T A^T b - b^T A x + b^T b - \epsilon \\ &= b^T b - 2A^T b x + x^T A^T A x - \epsilon. \end{aligned} \tag{5.22}$$

Now applying (5.14) to (5.22) with $M = A^T A$, we get

$$\begin{aligned}\nabla(\|Ax-b\|_2^2-\epsilon) &= -2A^Tb + (A^TA + (A^TA)^T)x \\ &= -2A^Tb + 2A^TAx \\ &= 2A^T(Ax-b) \\ &= 2A^Tr, \end{aligned} \tag{5.23}$$

where $r = Ax - b$ is the residual. We can now apply the factor $\frac{1}{2}$ from (5.8), and deduce

$$\frac{1}{f_\epsilon(z)}\nabla f_\epsilon(z) = \frac{1}{f_\epsilon(z)}\begin{bmatrix} A^Tr \\ 0_{n^3} \end{bmatrix}, \tag{5.24}$$

where $0_{n^3} \in \mathbb{R}^{n^3}$ and A^T is equivalent to applying the transpose operation of $\mathcal{A}$. We will denote this as

$$A^Tr \coloneqq (\mathcal{A}^T(R))(:), \tag{5.25}$$

where $\mathcal{A}^T : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ and $R = \mathcal{A}(X) - B$.

We are now ready to express g_z for our MRI reconstruction problem. Substituting the results of (5.21) and (5.24) into (5.8), we find the gradient of the objective function is

$$
\begin{aligned}
g_z &= c_0 + \frac{1}{\tau}\Big[\sum_{ijk} \frac{1}{-f_{T_{ijk}}(z)}\nabla f_{T_{ijk}}(z) + \frac{1}{-f_\epsilon(z)}\nabla f_\epsilon(z)\Big] \\
&= \begin{bmatrix} 0_{n^3} \\ 1_{n^3} \end{bmatrix} + \frac{-1}{\tau}\Big(\begin{bmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x \\ -F_t^{-1} t \end{bmatrix} + \frac{1}{-f_\epsilon}\begin{bmatrix} A^T r \\ 0_{n^3} \end{bmatrix}\Big) \\
&= \frac{-1}{\tau}\begin{bmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x + \frac{1}{f_\epsilon} A^T r \\ -\tau 1_{n^3} - F_t^{-1} t \end{bmatrix}.
\end{aligned}
\tag{5.26}
$$

5.3 H_z

In order to compute the Hessian in (5.9), we need to express four quantities:

1. $\sum_{ijk} \frac{1}{f_{T_{ijk}}(z)^2}\nabla f_{T_{ijk}}(z)[\nabla f_{T_{ijk}}(z)]^T$
2. $\sum_{ijk} \frac{1}{-f_{T_{ijk}}(z)}\nabla^2 f_{T_{ijk}}(z)$
3. $\frac{1}{f_\epsilon(z)^2}\nabla f_\epsilon(z)[\nabla f_\epsilon(z)]^T$
4. $\frac{1}{-f_\epsilon(z)}\nabla^2 f_\epsilon(z)$

Since we already have an expression for $\sum_{ijk} \frac{1}{-f_{T_{ijk}}(z)}\nabla f_{T_{ijk}}(z)$ and $\frac{1}{-f_\epsilon(z)}\nabla f_\epsilon(z)$ we will begin finding quantities 1 and 3 from the list above. First let's use expression (5.21) to get

$$
\sum_{ijk} \frac{1}{f_{T_{ijk}}(z)^2}\nabla f_{T_{ijk}}(z)[\nabla f_{T_{ijk}}(z)]^T = \begin{bmatrix} \hat{x} \\ -F_t^{-1} t \end{bmatrix}\begin{bmatrix} \hat{x}^T & -F_t^{-1} t^T \end{bmatrix}, \tag{5.27}
$$

where $\hat{x} = D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x$. Notice the top left block of the resulting matrix in (5.27) is given by

$$
\begin{aligned}
&\hat{x}\hat{x}^T \\
&= (D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x)(x^T D_h^T F_t^{-1} D_h + x^T D_v^T F_t^{-1} D_v + x^T D_t^T F_t^{-1} D_t) \\
&= D_h^T F_t^{-1} D_h x x^T D_h^T F_t^{-1} D_h + D_h^T F_t^{-1} D_h x x^T D_v^T F_t^{-1} D_v + D_h^T F_t^{-1} D_h x x^T D_t^T F_t^{-1} D_t \\
&\quad + D_v^T F_t^{-1} D_v x x^T D_h^T F_t^{-1} D_h + D_v^T F_t^{-1} D_v x x^T D_v^T F_t^{-1} D_v + D_v^T F_t^{-1} D_v x x^T D_t^T F_t^{-1} D_t \\
&\quad + D_t^T F_t^{-1} D_t x x^T D_h^T F_t^{-1} D_h + D_t^T F_t^{-1} D_t x x^T D_v^T F_t^{-1} D_v + D_t^T F_t^{-1} D_t x x^T D_t^T F_t^{-1} D_t \\
&= B F_t^{-2} B^T,
\end{aligned} \tag{5.28}
$$

where $B = D_h^T \Sigma_h + D_v^T \Sigma_v + D_t^T \Sigma_t$ and $\Sigma_h = \texttt{diag}(D_h x), \Sigma_v = \texttt{diag}(D_v x)$ and $\Sigma_t = \texttt{diag}(D_t x)$. Similarly, we see the top right block of the matrix in (5.27) can be written as

$$
\begin{aligned}
-\hat{x} F_t^{-1} t^T &= -(D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x) F_t^{-1} t^T \\
&= -B \tilde{T} F_t^{-2},
\end{aligned} \tag{5.29}
$$

where $\tilde{T} = \texttt{diag}(t)$ and $F_t^{-2} = \texttt{diag}(1 \oslash f_T^2)$. It is obvious we can now write the remaining blocks of the matrices

$$
\sum_{ijk} \frac{1}{f_{T_{ijk}}^2} \nabla f_{T_{ijk}}(z) [\nabla f_{T_{ijk}}(z)]^T = \begin{bmatrix} B F_t^{-2} B^T & -B \tilde{T} F_t^{-2} \\ -F_t^{-2} \tilde{T} B^T & F_t^{-2} \tilde{T}^2 \end{bmatrix}. \tag{5.30}
$$

Now we will examine how to express $\frac{1}{f_\epsilon(z)^2} \nabla f_\epsilon(z) [\nabla f_\epsilon(z)]^T$. Recall $\frac{1}{f_\epsilon(z)} \nabla f_\epsilon(z)$ from (5.24), and plug it in to get

$$\begin{aligned}\frac{1}{f_\epsilon^2(z)}\nabla f_\epsilon(z)[\nabla f_\epsilon(z)]^T &= \frac{1}{f_\epsilon^2(z)}\begin{bmatrix}A^Tr\\0\end{bmatrix}\begin{bmatrix}r^TA & 0\end{bmatrix}\\ &= \frac{1}{f_\epsilon^2(z)}\begin{bmatrix}A^Trr^TA & 0\\0 & 0\end{bmatrix}.\end{aligned} \tag{5.31}$$

All that remains to finish computing the Hessian, H_z, is $\sum_{ijk}\frac{1}{-f_{T_{ijk}}(z)}\nabla^2 f_{T_{ijk}}(z)$ and $\frac{1}{-f_\epsilon(z)}\nabla^2 f_\epsilon(z)$. Observe the Hessian is the Jacobian of the gradient. Thus, given $\phi(y) = y^TMy$, we apply (5.14) to the gradient $\nabla\phi = (M + M^T)y$, and see the Hessian is $\nabla^2\phi = M^T + M$. Using this fact we get

$$\sum_{ijk}\frac{1}{-f_{T_{ijk}}(z)}\nabla^2 f_{T_{ijk}}(z) = \begin{bmatrix}D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + D_t^T(-F_t^{-1})D_t & 0\\0 & F_t^{-1}\end{bmatrix}, \tag{5.32}$$

and

$$\frac{1}{-f_\epsilon(z)}\nabla^2 f_\epsilon = \frac{1}{-f_\epsilon(z)}\begin{bmatrix}A^TA & 0\\0 & 0\end{bmatrix}. \tag{5.33}$$

Recall $A^TAx = (\mathcal{A}^T(\mathcal{A}(X)))(:)$, but for consistency of notation we will use A^TA in writing the Hessian. We now have all the information we need to find the Hessian. Using the expressions from (5.30), (5.31), (5.32), and (5.33) to put into (5.9) shows

$$
\begin{aligned}
H_z &= \frac{1}{\tau}\Big[\sum_{ijk}\frac{1}{f_{T_{ijk}}(z)^2}\nabla f_{T_{ijk}}(z)(\nabla f_{T_{ijk}}(z))^T + \sum_{ijk}\frac{1}{-f_{T_{ijk}}(z)}\nabla^2 f_{T_{ijk}}(z) \\
&\quad + \frac{1}{f_\epsilon(z)^2}\nabla f_\epsilon(z)(\nabla f_\epsilon(z))^T + \frac{1}{-f_\epsilon(z)}\nabla^2 f_\epsilon(z)\Big] \\
&= \frac{1}{\tau}\Bigg(\begin{bmatrix} BF_t^{-2}B^T & -B\tilde{T}F_t^{-2} \\ -F_t^{-2}\tilde{T}B^T & F_t^{-2}\tilde{T}^2 \end{bmatrix} + \begin{bmatrix} D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + D_t^T(-F_t^{-1})D_t & 0 \\ 0 & F_t^{-1} \end{bmatrix} \\
&\quad + \begin{bmatrix} f_\epsilon^{-2}A^T r r^T A & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -f_\epsilon^{-1}A^T A & 0 \\ 0 & 0 \end{bmatrix}\Bigg) \\
&= \frac{1}{\tau}\begin{bmatrix} H_{11} & -B\tilde{T}F_t^{-2} \\ -F_t^{-2}\tilde{T}B^T & F_t^{-2}\tilde{T}^2 + F_t^{-1} \end{bmatrix},
\end{aligned} \tag{5.34}
$$

where

$$
\begin{aligned}
H_{11} &= BF_t^{-2}B^T + D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + D_t^T(-F_t^{-1})D_t \\
&\quad + f_\epsilon^{-2}A^T r r^T A - f_\epsilon^{-1}A^T A.
\end{aligned} \tag{5.35}
$$

In order to simplify notation, we will define

$$\Sigma_{12} = -\tilde{T}F_t^{-2}, \tag{5.36a}$$

$$\Sigma_{22} = F_t^{-2}\tilde{T}^2 + F_t^{-1}, \tag{5.36b}$$

and thus

$$H_z = \frac{1}{\tau}\begin{bmatrix} H_{11} & B\Sigma_{12} \\ \Sigma_{12}B^T & \Sigma_{22} \end{bmatrix}. \tag{5.37}$$

5.4 Reduce the System

Now that we have the 3D expressions for g_z and H_z stated in (5.26) and (5.37), each step of the Newton method needs to solve a large scale system. Substituting (5.26) and (5.37) in (5.10) yields

$$\begin{bmatrix} H_{11} & B\Sigma_{12} \\ \Sigma_{12}B^T & \Sigma_{22} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \end{bmatrix} = \begin{bmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + D_t^T F_t^{-1} D_t x + f_\epsilon^{-1} A^T r \\ -\tau - F_t^{-1} t \end{bmatrix} := \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}. \tag{5.38}$$

Here we have defined w_1 and w_2 to make the following computations more readable. In order to reduce the system of equations, we eliminate Δt in Equation (5.38). Notice

$$\Sigma_{12}B^T\Delta x + \Sigma_{22}\Delta t = w_2 \quad \Rightarrow \quad \Delta t = \Sigma_{22}^{-1}(w_2 - \Sigma_{12}B^T\Delta x). \tag{5.39}$$

Using this, we reduce the system of equations as follows

$$\begin{aligned} H_{11}\Delta x + B\Sigma_{12}\Delta t &= w_1 \\ \Longrightarrow\ & H_{11}\Delta x + B\Sigma_{12}[\Sigma_{22}^{-1}(w_2 - \Sigma_{12}B^T\Delta x)] = w_1 \\ \Longrightarrow\ & H_{11}\Delta x + B\Sigma_{12}\Sigma_{22}^{-1}w_2 - B\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}B^T\Delta x = w_1 \\ \Longrightarrow\ & (H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T)\Delta x = w_1 - B\Sigma_{12}\Sigma_{22}^{-1}w_2 \\ \Longrightarrow\ & \hat{H}_{11}\Delta x = \hat{w}_1, \end{aligned} \tag{5.40}$$

where $\hat{w}_1 = w_1 - B\Sigma_{12}\Sigma_{22}^{-1}w_2$ and $\hat{H}_{11} = H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T$. However, in order to implement this, we can expand $\hat{H}_{11}$ and get a more simple expression in order to coding of the problem easier. Notice the following expansion for $\hat{H}_{11}$:

$$
\begin{aligned}
\hat{H}_{11} &= H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T \\
&= BF_t^{-2}B^T + D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + f_\epsilon^{-2}A^Trr^TA - f_\epsilon^{-1}A^TA \\
&\quad - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T \\
&= (D_h^T\Sigma_h + D_v^T\Sigma_v + D_t^T\Sigma_t)F_t^{-2}(D_h^T\Sigma_h + D_v^T\Sigma_v + D_t^T\Sigma_t)^T \\
&\quad + D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + D_t^T(-F_t^{-1})D_t + f_\epsilon^{-2}A^Trr^TA - f_\epsilon^{-1}A^TA \\
&\quad - (D_h^T\Sigma_h + D_v^T\Sigma_v + D_t^T\Sigma_t)\Sigma_{12}^2\Sigma_{22}^{-1}(D_h^T\Sigma_h + D_v^T\Sigma_v + D_t^T\Sigma_t)^T \\
&= (D_h^T\Sigma_h + D_v^T\Sigma_v + D_t^T\Sigma_t)F_t^{-2}(\Sigma_{\delta h}D_h + \Sigma_v D_v + \Sigma_t D_t) \\
&\quad + D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + D_t^T(-F_t^{-1})D_t + f_\epsilon^{-2}A^Trr^TA - f_\epsilon^{-1}A^TA \\
&\quad - (D_h^T\Sigma_h + D_v^T\Sigma_v D_t^T + D_t^T\Sigma_t)\Sigma_{12}^2\Sigma_{22}^{-1}(\Sigma_{\delta h}D_h + \Sigma_v D_v + \Sigma_t D_t) \\
&= D_h^T\Sigma_{\delta h}F_t^{-2}\Sigma_{\delta h}D_h + D_h^T\Sigma_{\delta h}F_t^{-2}\Sigma_v D_v + D_h^T\Sigma_{\delta h}F_t^{-2}\Sigma_t D_t \\
&\quad + D_v^T\Sigma_v F_t^{-2}\Sigma_{\delta h}D_h + D_v^T\Sigma_v F_t^{-2}\Sigma_v D_v + D_v^T\Sigma_v F_t^{-2}\Sigma_t D_t \\
&\quad + D_t^T\Sigma_t F_t^{-2}\Sigma_{\delta h}D_h + D_t^T\Sigma_t F_t^{-2}\Sigma_v D_v + D_t^T\Sigma_t F_t^{-2}\Sigma_t D_t \\
&\quad + D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + D_t^T(-F_t^{-1})D_t + f_\epsilon^{-2}A^Trr^TA - f_\epsilon^{-1}A^TA \\
&\quad - D_h^T\Sigma_{\delta h}\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_{\delta h}D_h - D_h^T\Sigma_{\delta h}\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_v D_v - D_h^T\Sigma_{\delta h}\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_t D_t \\
&\quad - D_v^T\Sigma_v\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_{\delta h}D_h - D_v^T\Sigma_v\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_v D_v - D_v^T\Sigma_v\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_t D_t \\
&\quad - D_t^T\Sigma_t\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_{\delta h}D_h - D_t^T\Sigma_t\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_v D_v - D_t^T\Sigma_t\Sigma_{12}^2\Sigma_{22}^{-1}\Sigma_t D_t \\
&= D_h^T(\Sigma_{\delta h}^2\Sigma_b - F_t^{-1})D_h + D_v^T(\Sigma_v^2\Sigma_b - F_t^{-1})D_v + D_t^T(\Sigma_t^2\Sigma_b - F_t^{-1})D_t \\
&\quad + D_h^T(\Sigma_{\delta h}\Sigma_v\Sigma_b)D_v + D_h^T(\Sigma_{\delta h}\Sigma_t\Sigma_b)D_t + D_v^T(\Sigma_{\delta h}\Sigma_v\Sigma_b)D_h + D_v^T(\Sigma_v\Sigma_t\Sigma_b)D_t \\
&\quad + D_t^T(\Sigma_{\delta h}\Sigma_t\Sigma_b)D_h + D_t^T(\Sigma_v\Sigma_t\Sigma_b)D_v + f_\epsilon^{-2}A^Trr^TA - f_\epsilon^{-1}A^TA,
\end{aligned}
\tag{5.41}
$$

where $\Sigma_b = F_t^{-2} - \Sigma_{12}^2\Sigma_{22}^{-1}$.

Notice the tubal finite difference operator added a total of six extra terms in order to account for the interaction of the three gradient directions. Three dimensional systems can now be solved in the same way as the 2D systems using this new $\hat{H}_{11}$ which is expressed as

$$\hat{H}_{11}\Delta x = \hat{w}_1 \tag{5.42}$$

Applying this method to a $128 \times 128 \times 128$ image, we obtain the following results in Figure 5.1.

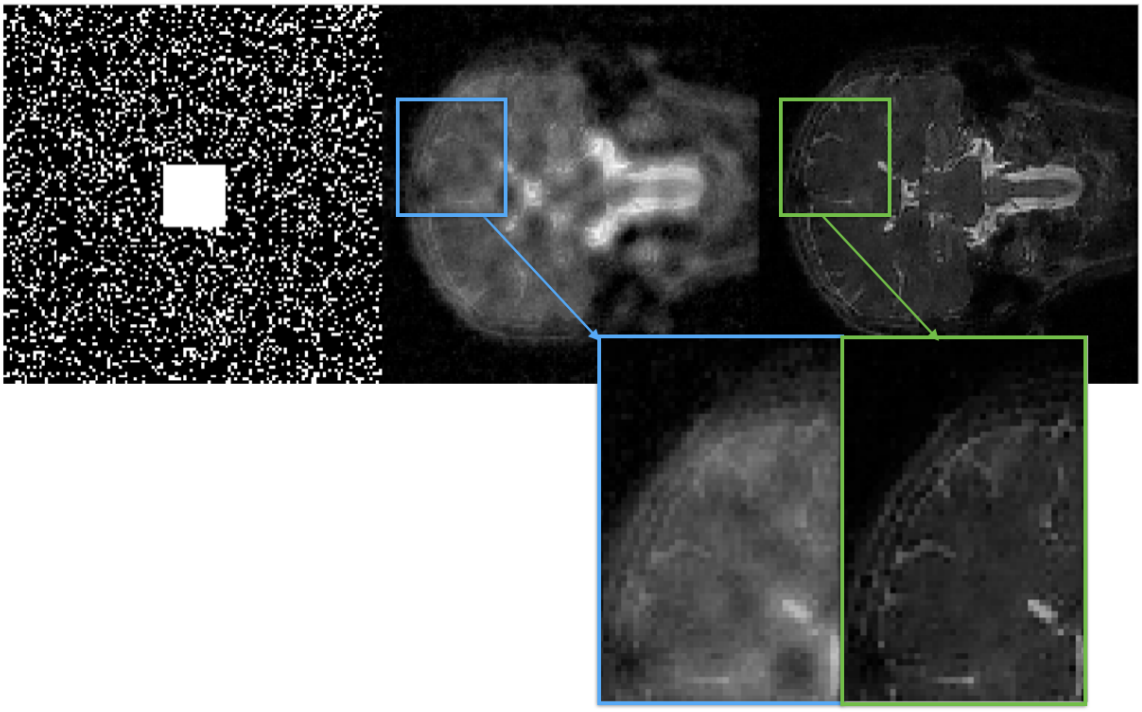

Figure 5.1. Results of slice 87. Left: Mask; Center: Input Image; Right: Recovered Image.

In Figure 5.1, the 20% sample scheme was one of many tried on the 3D image data. We will see how the SparseMRI, ℓ_1-Magic and the 3DIRECT methods compare against each other when run under the same sample schemes.

5.5 Efficiency Improvements

In addition to processing a 3D image all at once vs. slice by slice, further improvements were made to the code that was originally provided for the 2D case in [8]. In this code there were two major things we attempted to improve upon. These are the speed of the large matrix multiplications that occur when applying D_h and D_v, and improving CG stopping criteria.

It was observed that the CG method in [8] used to solve $\hat{H}_{11}x = \hat{w}_1$, always achieved the preset maximum number of iterations, and often times had large residual errors when it was stopped. This typically implies our linear operator has a large condition number. For our specific problem we have condition numbers on the order of 10^{17} which implies our systems are numerically singular. There were several ideas we tried to improve convergence, including:

- Regularization,
- Preconditioning.

Regularization is a popular method for helping solve ill-conditioned least squares problems. Instead of solving the system $Ax = b$, it attempts to solve the following minimization problem

$$\min_x \|Ax - b\|^2 + \rho\|x\|^2,$$

where the constant $\rho > 0$ depends on the specific problem being solved [58]. This formulation attempts to solve the least squares problem while penalizing solutions with high norms. For our purposes, we selected various values for $\rho \in [10^{-10}, 10^{-2}]$,

but saw no improvement in the image quality or resolution of detail in the image. Next, we tried implementing a *preconditioner*. A preconditioner attempts to solve an equivalent system whose condition number is usually made close to 1, as

$$M^{-1}Ax = M^{-1}b,$$

where the preconditioner M is symmetric positive definite [58]. There are several predconitioner matrices one can choose from. The most simple is $M = \texttt{diag}(A)$. Recall for the MRI application, $\hat{H}_{11}$ depends on $\mathcal{A}$ which is a linear operator that is dependent upon the mask chosen to subsample the k-space. Thus we need a way to approximate $\texttt{diag}(\hat{H}_{11})$. To do this we first simplify (5.41) to

$$\hat{H}_{11} = D + f_\epsilon^{-2} A^T r r^T A - f_\epsilon^{-1} A^T A, \tag{5.43}$$

where

$$\begin{aligned} D = D_h^T(\Sigma_{\delta h}^2 \Sigma_b - F_t^{-1}) D_h + D_v^T(\Sigma_v^2 \Sigma_b - F_t^{-1}) D_v \\ + D_h^T(\Sigma_{\delta h} \Sigma_v \Sigma_b) D_v + D_v^T(\Sigma_{\delta h} \Sigma_v \Sigma_b) D_h. \end{aligned} \tag{5.44}$$

By its construction, finding $\texttt{diag}(D) \coloneqq \Lambda$ is straightforward. Also notice,

$$\texttt{diag}(f_\epsilon^{-2} A^T r r^T A) = f_\epsilon^{-2} (A^T r)^{②} \in \mathbb{R}^{n \times 1}$$

is also easy to handle, where given a vector y, $y^{②}$ indicates to square the entires, i.e. $(y^{②})_i = y_i^2$. Thus the approximation portion comes from the term $f_\epsilon^{-1} A^T A$.

We know from the MRI application A is the matrix representation of $\mathcal{A}$ which is a sparse Fourier Transform operator. It exists, but usually not explicitly written down and stored. The fully sampled operator is orthogonal thus $\mathcal{A}^T \mathcal{A} = I$. At the

other extreme none of the samples are kept, and then the diagonal of $\mathcal{A}^T\mathcal{A}$ contributes nothing to $\hat{H}_{11}$. So we take

$$\texttt{diag}(\hat{H}_{11}) \approx \Lambda + f_\epsilon^{-2}\texttt{diag}((A^T r)^{②}) - f_\epsilon^{-1}kI, \tag{5.45}$$

is likely a decent approximation for the diagonal of $\hat{H}_{11}$, where k is an integer from 1 to n^2 corresponding to the number of sampled pixels in the k-space. Again, we found despite the value of k used in recovery process, the numerical solution did not improve the recovered image quality or error relative to the fully sampled image.

This prompted us to another preconditioner expressed as

$$M = \Lambda + f_\epsilon^{-2}A^T r r^T A - f_\epsilon^{-1}kI, \tag{5.46}$$

where we kept the entire portion of the $f_\epsilon^{-2}\nabla f_\epsilon(z)\nabla f_\epsilon^T(z)$ term as it is of rank 1. We can find an explicit expression for M^{-1}. First we claim given vectors x and y of same length

$$\Big(I + xy^T\Big)^{-1} = I - \frac{xy^T}{1 + y^T x}, \tag{5.47}$$

provided $1 + y^T x \neq 0$. To see this, we observe

$$\begin{aligned}
\Big(I + xy^T\Big)\Big(I - \frac{xy^T}{1 + y^T x}\Big) &= I - \frac{xy^T}{1 + y^T x} + xy^T - \frac{xy^T xy^T}{1 + y^T x} \\
&= I - \frac{-xy^T + xy^T + (y^T x)xy^T - (y^T x)xy^T}{1 + y^T x} \\
&= I.
\end{aligned}$$

We rewrite M in (5.46) as

$$\begin{aligned} M &= \hat{\Lambda} + \alpha a a^T \\ &= \hat{\Lambda}\Big(I + \alpha \hat{\Lambda}^{-1} a a^T\Big), \end{aligned} \tag{5.48}$$

where $\alpha = f_\epsilon^{-2}, a = A^T r$ and $\hat{\Lambda} = \Lambda - f_\epsilon^{-1} k I$. Using (5.47) and (5.48), we see

$$\begin{aligned} \Big(\hat{\Lambda} + \alpha a a^T\Big)^{-1} &= \Big(\hat{\Lambda}(I + \alpha \hat{\Lambda}^{-1} a a^T)\Big)^{-1} \\ &= \Big(I + (\alpha \hat{\Lambda}^{-1} a) a^T\Big) \hat{\Lambda}^{-1} \\ &= \Big(I - \frac{\alpha \hat{\Lambda}^{-1} a a^T}{1 + \alpha a^T \hat{\Lambda}^{-1} a}\Big) \hat{\Lambda}^{-1} \\ &= \hat{\Lambda}^{-1} - \frac{\alpha}{1 + \alpha a^T \hat{\Lambda}^{-1} a} (\hat{\Lambda}^{-1} a)(a^T \hat{\Lambda}^{-1}). \end{aligned} \tag{5.49}$$

To implement this efficiently, we notice that $\hat{\Lambda}$ is a diagonal matrix and it allows us to write the ratio in (5.49) as

$$\frac{\alpha}{1 + \alpha a^T \hat{\Lambda}^{-1} a} = \frac{f_\epsilon^{-2}}{1 + f_\epsilon^{-2} \sum_i \hat{\Lambda}_{ii}^{-1} a_i^2} = \frac{1}{f_\epsilon^2 + \sum_i \hat{\Lambda}_{ii}^{-1} a_i^2} \coloneqq \kappa. \tag{5.50}$$

Therefor M^{-1} applied to a vector z is given by

$$M^{-1} z = \hat{\Lambda}^{-1} z - \kappa (\hat{\Lambda}^{-1} a)(a^T \hat{\Lambda}^{-1}) z, \tag{5.51}$$

which cost very little to implement. Incorporating the CG method outlined in Section 3.1 with the precoditioner (5.51) did not result in significant improvements in the residual error. This prompted us to check if our approximation for $\mathcal{A}^T \mathcal{A}$ used in (5.45) was good enough. To do so we ran a problem using a much smaller value of n, and attempted to solve the system directly by actually constructing the sparse

Fourier Matrix. This was done by iterating over the n^2 basis vectors as shown in Algorithm 5.2.

Algorithm 5.2 Construct Sparse Fourier Matrix

Output: $K \in \mathbb{R}^{n^2 \times n^2}$

1: $v = \texttt{zeros}(n^2, 1)$
2: **for** $i = 1 : n^2$ **do**
3: **if** $i = 1$ **then**
4: $v_i = 1$
5: **else**
6: $v_{i-1} = 0; v_i = 1$
7: **end if**
8: $K_{(:,i)} = \mathcal{A}^T(\mathcal{A}(v))$ note: $\mathcal{A}^T$ and $\mathcal{A}$ are subroutines
9: **end for**

Using this we then directly constructed $\hat{H}_{11}$ as

$$\hat{H}_{11} = D + f_\epsilon^{-2} A^T r r^T A - f_\epsilon^{-1} K, \tag{5.52}$$

where K is the output of Algorithm 5.2. Then, using MATLAB's backslash solver found Δx by $\Delta x = \hat{H}_{11} \backslash \hat{w}_1$. While this method is not practical for implementing on real world applications, it did reveal that the final results achieved by the interior point method with the log-barrier function were similar to that of the normal, regularized, and preconditioning methods. Thus, for the MRI application and images tested, it was observed that running a standard CG method was the best for ease of implementation and quality reconstruction of the recovered image. Thus, we analyzed the stopping criterion of the CG method.

Running various masks and observing the error tolerance returned from the CG method showed, the lower bound for the error was of the order of $O(10^{-3})$. Thus we changed to stopping tolerance from $O(10^{-8})$ as used by ℓ_1-Magic, to something more in line with the observed error for MRI image recovery. Also ℓ_1-Magic checks CG resolution expressed as $\frac{\|r_{(i)}\|}{\|r_{(0)}\|}$, and if its greater than $\frac{1}{2}$, it returns to the previous CG iterate and continues the Newton iterations. However, we noticed this did not improve the recovered image quality. Thus we implemented a flag to check when this condition happens, and then stop the remaining Newton iterations based on the CG resolution being larger then the $\frac{1}{2}$ threshold. Both these changes resulted in fewer CG and Newton iterations with similar recovered image quality as seen when allowing maximum iterations to be obtained using more stringent stopping but unrealistic criteria. Hence we obtained similar results quicker then ℓ_1-Magic.

Next, we spent time focusing on how to speed up the calculations themselves. The slowest lines of code are the application of the finite difference matrices used to find g_z and H_z. The code in [8] chose to implement the discrete gradient matrices as sparse matrices using MATLAB's `spdiags`. This command uses the fact the matrices D_h and D_v can be more compactly stored in memory as sparse diagonal matrices. Further, MATLAB can perform matrix multiplication very efficiently using the sparse matrix array structure it has built in. However, we improved the speed on these calculations by noticing MATLAB had more efficient built-in functions such as the `diff` operator, that, if used intelligently, can improve the speed of these matrix multiplications dramatically.

An example of implementing a change in the $2D$ operator given in [8] is

$$D_v = \texttt{spdiags}([\texttt{reshape}([-\texttt{ones}(n-1,n);\texttt{zeros}(1,n)],N,1)...$$
$$\texttt{reshape}([\texttt{zeros}(1,n);\texttt{ones}(n-1,n)],N,1)],[01],N,N);$$

versus our improved implementation

$$v = [\texttt{diff}(X, 1, 1); \texttt{zeros}(1, n)]; \quad D_v = v(:);$$

where for a single iteration we see a 98% increase in speed. Similarly for D_h we see

$$D_h = \texttt{spdiags}([\texttt{reshape}([-\texttt{ones}(n, n-1); \texttt{zeros}(n, 1)], N, 1)...$$
$$\texttt{reshape}([\texttt{zeros}(n, 1); \texttt{ones}(n, n-1)], N, 1)], [0n], N, N);$$

in [8] versus

$$h = [\texttt{diff}(X, 1, 2); \texttt{zeros}(n, 1)]; \quad D_h = h(:);$$

This benefit is further enlarged by the fact each finite difference matrix is invoked multiple times during the iterations. Assuming we run one iteration of the 2D interior point method with log-barrier function, we have to perform 12 matrix multiplications per one complete call.

To see the timing benefits from intelligent use of built-in MATLAB function calls, we collected statistics for a 512×512 MRI reconstruction application. The averages are reported in Table 5.1 and 5.2.

Method	Total Number of Appl. of D_v and D_h	Average Iterations for 2D MRI Method	Total Mult.
Log Barrier	4	16	64
Newton	6	60	360
CG	2	20	40
Total	**12**	**96**	**464**

Table 5.1. Average Number of Observed Iterations for 512×512 2D MRI Application

Method	Total Mult.	Original Times (s)		Improved Times (s)	
		D_h	D_v	D_h	D_v
		1.3×10^{-1}	1.3×10^{-1}	1.0×10^{-3}	1.9×10^{-3}
Log Barrier	64	8.32	8.32	0.064	0.122
Newton	360	46.8	46.8	0.36	0.684
CG	40	5.2	5.2	0.04	0.076
Total	**464**	**60.32**	**60.32**	**0.464**	**0.882**

Table 5.2. Timing Benefits for 512×512 2D MRI Application. Note the times under the finite difference operators in the header row, indicate the time to run one instance of said operator.

Based on these results, we reduced the overall computation time by almost 2 mins for the 2D MRI application which typically can run 4.5 minutes using a 17% sample scheme on a 512×512 image; that's *nearly twice as fast*. Obviously the amount of speed improvement is application dependent, but it is clear our efficiencies in implementation of the finite difference matrices is a vast improvement over sparse matrix multiplication.

It should be noted that although we included timing for the 2D case, these benefits are further magnified when we move to the 3D case. We implemented all three finite difference matrices via function calls using MATLAB's `diff` command. The function calls are much more efficient and therefore used in collecting the numerical results later in Chapter 6.

CHAPTER 6

NUMERICAL RESULTS

In this chapter we focus on the results of recovering a 3D brain image from subsampled data in the k-space. We will discuss how the simulation is initialized, metrics used for image comparison, and provide simulation results for different sample schemes:

- Symmetric Low Frequency Sample Scheme,
- 2D Favorable Sample Scheme Using 15% 2D Peak FFT Bins,
- 3D Favorable Sample Scheme Using 15% 3D Peak FFT Bins,
- 2D vs. 3D Favorable Peaks for Varying Sampling Percentages,
- A Practical 2D Sample Scheme Implementation,
- A Practical 3D Sample Scheme Implementation, and
- A Comparison of 2D and 3D Practical Schemes.

6.1 Initialization

The image data that will be presented here was provide by University of Texas Southwestern Medical Center[1]. It is a 3D brain image that was collected in 2010. The image contains $128 \times 128 \times 128$ complex data that was fully sampled in the k-space. While this is not normally collected in practice, it allows for us to subsample the data under different schemes, and observe the impact on image recovery. Figure 6.1 demonstrates how this initialization of data is done.

[1]A special thank you to Dr. Jian-xiong Wang of UT Southwestern Medical Center for providing the image data.

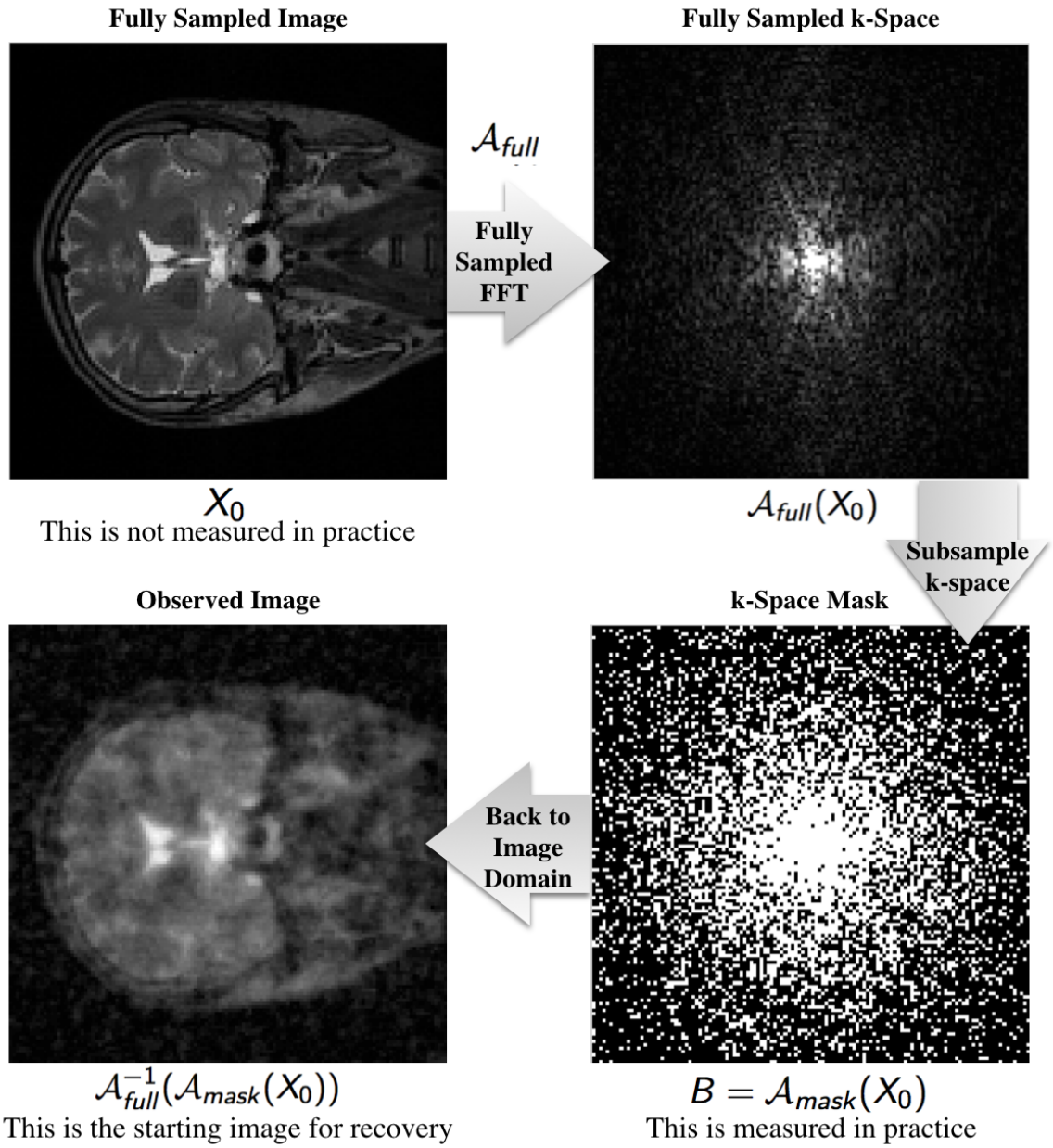

Figure 6.1. A demonstration of how fully sampled simulation data is subsampled to provide the starting image, or ℓ_2 image, for testing reconstruction methods.

The observed image and its subsampled k-space measurements are the initial data for each of the methods. We will now briefly discuss the structure of our simulation.

Our simulation was developed in MATLAB. It initializes images to be tested, provides a selection of various sample schemes, and then invokes each of the three methods: SparseMRI, ℓ_1-Magic, and 3DIRECT. It then provides various plots of different image metrics, and displays the recovered images. The image metrics will be discussed in Section 6.2. Both SparseMRI and ℓ_1-Magic were downloaded and installed from their respective sources, and then modified to iterate over a 3D image.

SparseMRI was developed by Michael Lustig and his collaborators. The code was developed and implemented in MATLAB based on the algorithms described in [34]. This code can be downloaded from:

`people.eecs.berkeley.edu/~mlustig/Software.html.`

The main script that was modified to handle the 3D brain image to run slice by slice is `demo_2D.m`. This script uses the complex k-space data as its initial data. This is because `demo_2D.m` was written to solve (4.1) in the form

$$\min_{z\in\mathbb{R}^n} \|\mathcal{A}(C^{-1}z) - b\|_2^2 + \lambda\|z\|_1, \tag{6.1}$$

where $z = Cx$. Contrastingly, ℓ_1-Magic uses the image data, which is purely real, as its input. Similarly, this method was also modified to run slice by slice. It was developed by Candes, Tao, Romberg and their collaborators, and is also implemented in MATLAB. It contains several convex optimization routines including the standard interior point method. This code can be downloaded from:

`statweb.stanford.edu/~candes/software.html.`

The main function modified here to handle the 3D image slice by slice is `tvqc_2Dexample.m`. Again it was setup to run each slice of the 3D data and report out the results.

The 3DIRECT method was based on extending and improving the following functions and subfunctions from ℓ_1-Magic:

- `tvqc_2Dexample.m`: main function,
- `A_fhp.m`: performs inverse transformation of $\mathcal{A}$,
- `A_f.m`: performs transformation of $\mathcal{A}$,
- `tvqc_logbarrier.m`: sets up log-barrier iterations,
- `tvqc_newton.m`: runs Newton Iterations,
- `cgsolve.m`: runs CG iterations,
- `TVdiff.m`: calculates total variation, and
- `H11.m`: calculates $\hat{H}_{11}z$.

By modifying the above functions to implement the 3DIRECT method, a new suite of MATLAB functions were created. 3DIRECT also uses the real image data as its input. We will next discuss the image metrics used for comparing the recovered images of each of the methods.

6.2 Image Comparison Metrics

Comparing two images is rather subjective. However, recent researchers have proposed some metrics to try and help quantify how similar two images are. One such metric is *peak signal-to-noise ratio* (PSNR). It measures the difference between a signal's maximum power and its noise floor. It is typically expressed in units of decibels where a higher PSNR typically indicates a better image reconstruction. We say typically, because ultimately human perception is the deciding factor on how two images compare [30].

Given an $m \times n$ 2D image X and Y, we first compute the mean squared error (MSE) as

$$\text{MSE} = \frac{1}{n \cdot m} \sum_{i=1}^{m} \sum_{j=1}^{n} (X(i,j) - Y(i,j))^2. \tag{6.2}$$

PSNR is defined as

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{I_{\max}^2}{\text{MSE}} \right), \tag{6.3}$$

where $I_{\max}^2$ is maximum possible pixel value of the image on Y. Another popular metric is the *structural similarity index* (SSIM). SSIM was developed to improve upon the deficiencies of PSNR [56, 57]. The SSIM measure between two images, , X and Y, of the same size is given by

$$\text{SSIM}(X,Y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{6.4}$$

where

- μ_x is the average of window size l of X which moves pixel-by-pixel over the entire image,
- μ_y is the average of window size l of Y which moves pixel-by-pixel over the entire image,
- σ_x^2 is the variance of X,
- σ_y^2 is the variance of Y,
- σ_{xy} is the covariance of X and Y, and
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are stabilization variables for weak denominator, where L is the dynamic range of pixel values and $k_1 = 0.01, k_2 = 0.03$.

The code for SSIM can be downloaded from

`www.mathworks.com/matlabcentral/answers/9217-need-ssim-m-code`

The last metric used to compare images is human perception. Despite what PSNR or SSIM indicates, the user gets the final say in how well a reconstructed image appears. This will become clear when looking at a few recovered images, and their metrics.

6.3 Sample Scheme: Symmetric Low Frequency Bins

The symmetric low frequency sample scheme allows us to reasonably assure the methods recover the images as expected. This scheme grabs the low frequency bins symmetrically about the zero frequency bin or *direct current* (DC) bin. It is called the DC bin because it represents the DC voltage offset when looking at a 1D signal in signal processing. It allows a comparison of the three methods performances under the similar starting starting images. This is because the symmetry of the sample scheme allows similar frequency components to be sampled, and these sampled frequencies correspond to the main image structure and don't include noise or fine details which can make recovery more challenging.

The subsampled image shown in this section is for a 10.26% mask, which is a square centered in each slice, whose sides measure 41 bins long (i.e. $41/128 = .1026$). In Figure 6.2, the PSNR results of the starting image and the recovered image for each method are shown.

When examining the results, we note that the legend indicating the solid line with square markers, labeled ℓ_2 for each of the methods, is the PSNR of the input image. The dotted lines with the triangle markers are the PSNR values of the recovered image. Ideally the dotted line should be above the solid line if the algorithm improved upon the image. Based on this, SparseMRI did not improve the PSNR, despite all three methods starting out with similar PSNR for the starting or ℓ_2 image.

This is due to the fact that SparseMRI has been observed to require about 40% or higher sampling rates to get good quality in its recovered image.

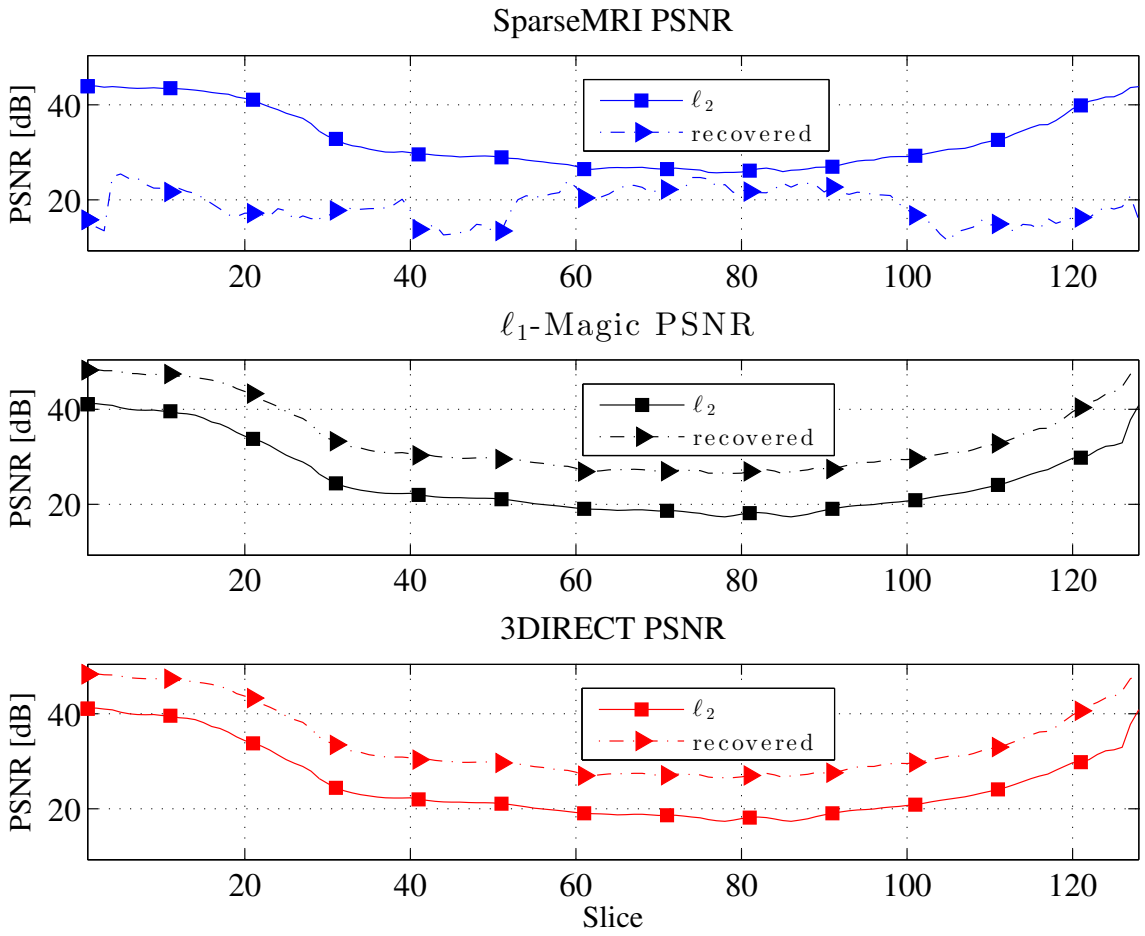

Figure 6.2. PSNR of Initial ℓ_2 Image vs. PSNR of Reconstructed Image using Symmetric Low Frequency Bins. Large PSNR should correspond to better image quality. Thus, based on these results, the SparseMRI method did not do a good job recovering the image ℓ_1-Magic and 3DIRECT have similar performance.

In addition SparseMRI and ℓ_1-Magic slightly differ in PSNR for the ℓ_2 image. This is because of the way the methods instantiate their sparse 2D Fourier Transform operator. Since SparseMRI tries to minimize the $\|\cdot\|_1$-norm under a wavelet

transformation in the k-space, it retains complex data. Contrastingly, ℓ_1-Magic tries to minimize over total variation in the image domain, thus it uses the real part of the k-space spectrum. Because the test data set we have was collected in the complex k-space, there is some energy contained in the complex data, hence the slight difference in the PSNR of the starting images for the 2D methods.

Figure 6.2 verifies that the 2D methods and 3DIRECT returned similar PSNR for the recovered image. This was the goal: to verify the new 3DIRECT method agrees with the well-accepted ℓ_1-Magic method under a symmetric low frequency sample scheme. This agreement is also reenforced by the fact that the SSIM values for the two methods are very similar, as seen in Figure 6.3.

We also observed in Figure 6.3, that Sparse MRI has the highest SSIM of its ℓ_2 starting image. This is due to how this method implements its 2D Fourier Operator. Unlike, ℓ_1-Magic and 3DIRECT, who subsample the complex k-space and then return the real portion of the inverse Fourier operator for the image domain, SparseMRI returns the absolute value of the complex subsampled k-space inverse Fourier Transform. Because of this, it retains some additional information stored in the k-space. However, the recovery results of SparseMRI suffer from low sampling percentage, making the SSIM of the recovered image worse than its starting image.

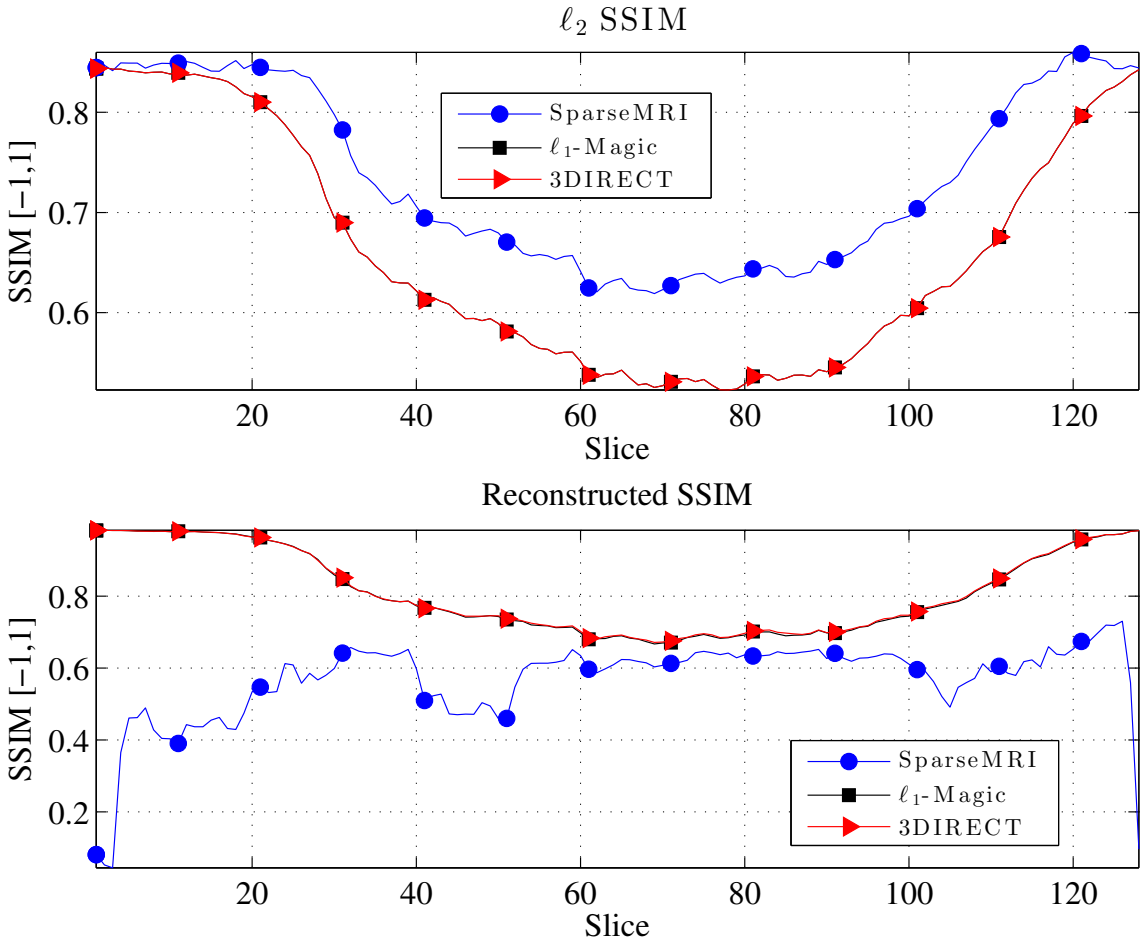

Figure 6.3. SSIM using DC Bins. Higher SSIM should mean the image is closer in structure to the original image. Again, by selecting symmetric bins around the DC bin, we have verified the implementation and similar performance of the methods.

We note that again SparseMRI did not do a good job in recovering the image based on its lower SSIM of the recovered image, and suffered at the end slices. Lastly, we use the human perception metric to indeed verify the indications of PSNR and SSIM. For reference, slices 10 and 87 are picked as arbitrary slices for plotting.

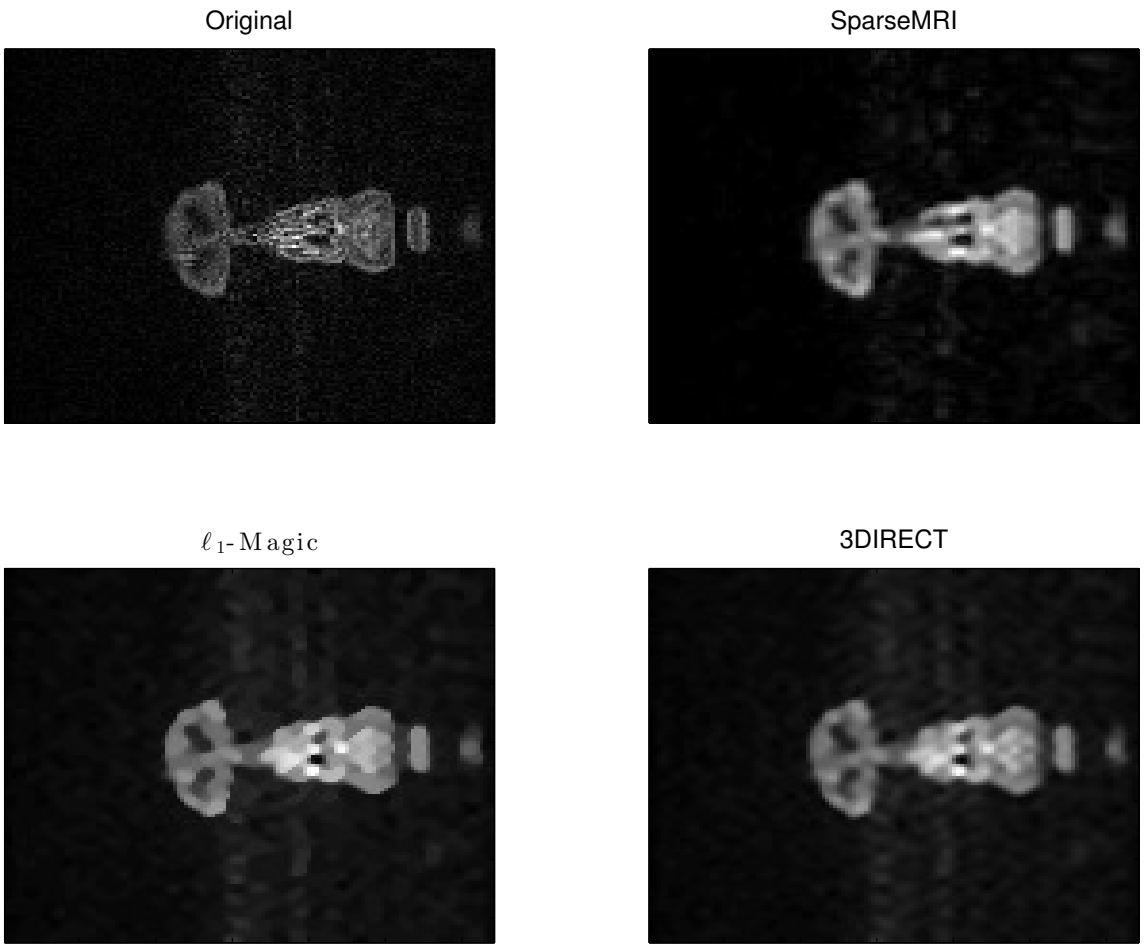

Figure 6.4. Recovered image using DC Bins (slice 20). With PSNR and SSIM in agreement, the last metric we use to validate performance is human perception. ℓ_1-Magic and 3DIRECT most closely resemble the original image, where SPARSE MRI loses structural detail and background information, which confirms the SSIM results.

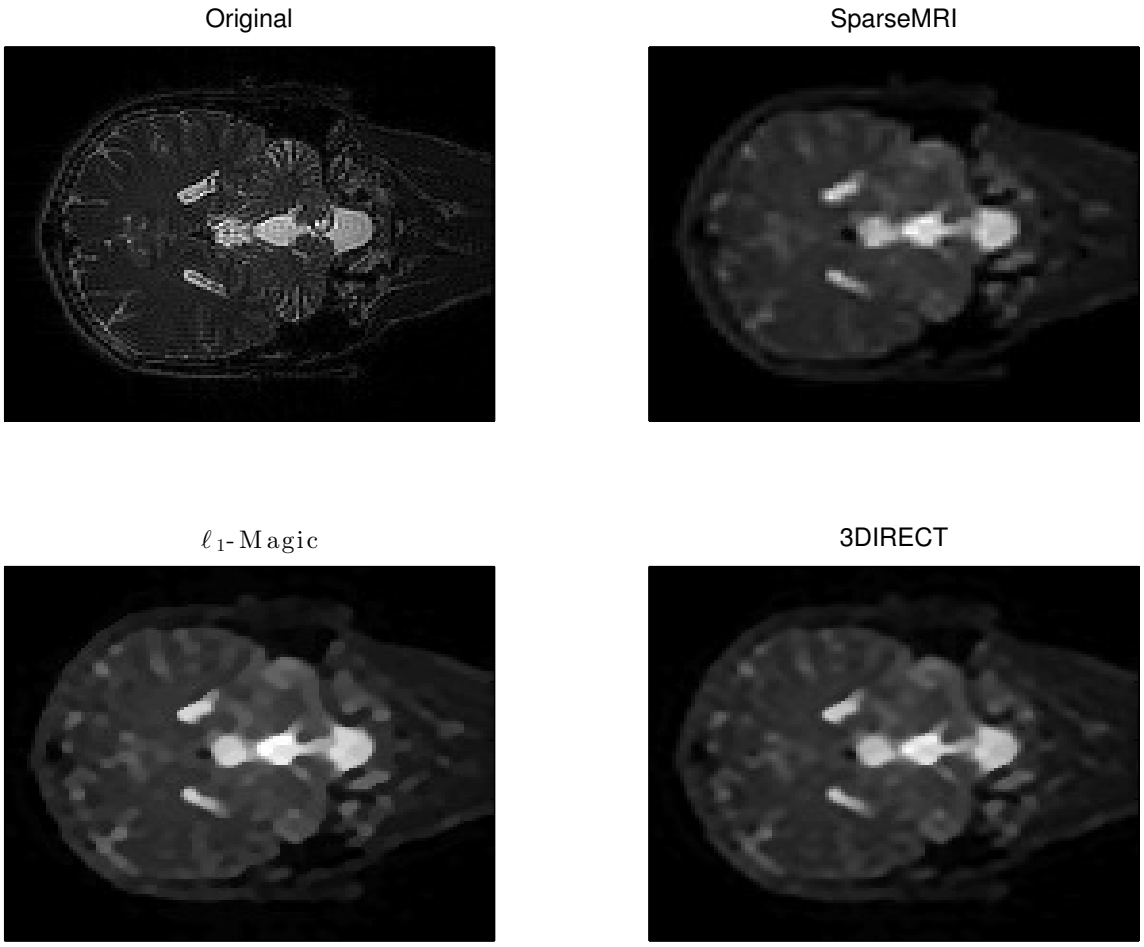

Figure 6.5. Recovered image using DC Bins (slice 87). With PSNR and SSIM in agreement, the last metric we use to validate performance is human perception. As shown here, all the methods have similar recovered images which is in agreement with the SSIM results.

The results in Figures **??**, 6.4 and 6.5 do confirm ℓ_1-Magic and 3DIRECT perform on par with one another when attempting to recover the images main structure without noise and fine image details contained in the higher frequencies. It also shows SparseMRI's weakness to low sampling rates, and that SSIM is a sensitive measure used to indicate performance. Lastly, it indicates that just grabbing 10% of the fre-

quency bins around the DC bin is a poor sampling scheme. This begs the question: what's the best sampling scheme? While there is a lot of research in this area, we will briefly touch on it in the next two subsections.

6.4 Sample Scheme: 15% of 2D Peaks

The importance of the peak FFT bin cases is that they give an idea of how good the methods can perform if dominant information is kept by the mask. These sample schemes are not practical to implement as it would require knowing all the data in the k-space in order to determine which bins to keep. However, it gives the reader an idea for how many samples must be kept in order to get good results.

The purpose of the 15% of 2D Peaks sample scheme is to examine how well each algorithm performs under a scheme that retains the maximum amount of information under a specified percentage for the 2D slice by slice approach used by the ℓ_1-Magic and SparseMRI. The mask for this scenario was generated by performing a 2D FFT of each slice of the fully sampled data, and then grabbing the top 15%, in magnitude, of all the n^2 Fourier samples per slice, giving an overall sampling rate of 15% for the 3D image. While this is not possible in practice, our simulation has access to a fully sampled data set which allowed us to a priori determine how the FFT energy is spread when no subsampling is done.

Selecting the top magnitude bins of each slice represents where most of the images data is stored. For the slice by slice approach, this meant the samples were concentrated in the middle of each slice unless the slice contained little to no brain cross-section, and then the samples got distributed across the slice as one would expect with background noise. This result is observed in Figure 6.6.

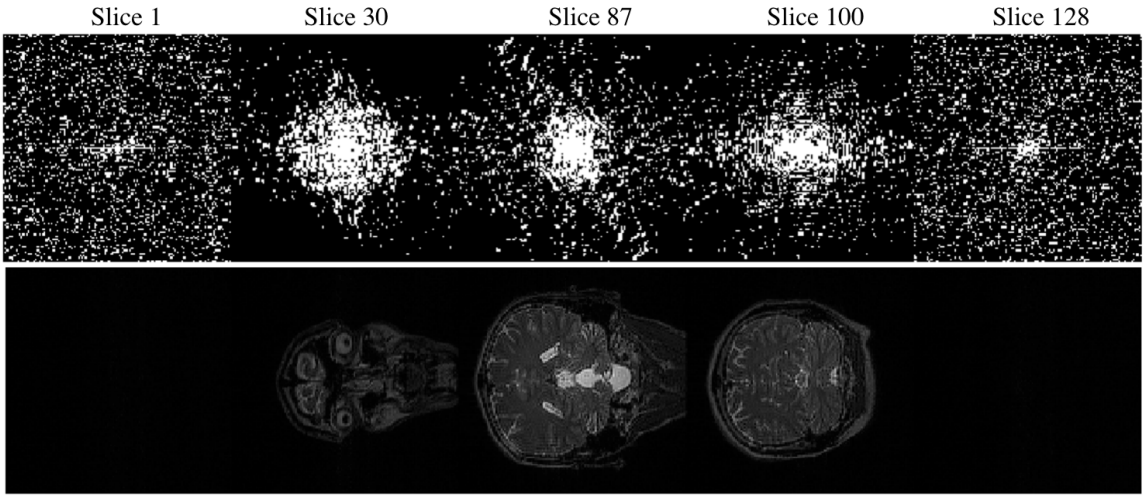

Figure 6.6. 15% of 2D FFT Peak Bins. By selecting the top 15% of the n^2 2D FFT bins per each slice, we are keeping the maximum amount of information in the k-space given a specified sampling rate using a 2D sample scheme. It is expected that this bin selection will allow this sample scheme to favor the 2D methods.

With the sample scheme in hand, each method was then run using their 2D and 3D FFT methods accordingly, keeping the bins as indicated from Figure 6.6. This means for 3DIRECT, that bins in the mask correspond to 2D frequencies for each slice, however, when selecting those equivalent bin locations after performing a 3D FFT, there is now tubal frequency information as well. While this is unfavorable to 3DIRECT, we perform this case to see how well the 2D methods can perform at a 15% sample scheme. It was observed that SparseMRI suffered from using the absolute value of the subsampled k-space, and thus had poor initial images. Using this sample scheme each method's ℓ_2 result for slice 87 and 25 are captured in Figure 6.7 and 6.8. These slices were picked arbitrarily for comparison purposes.

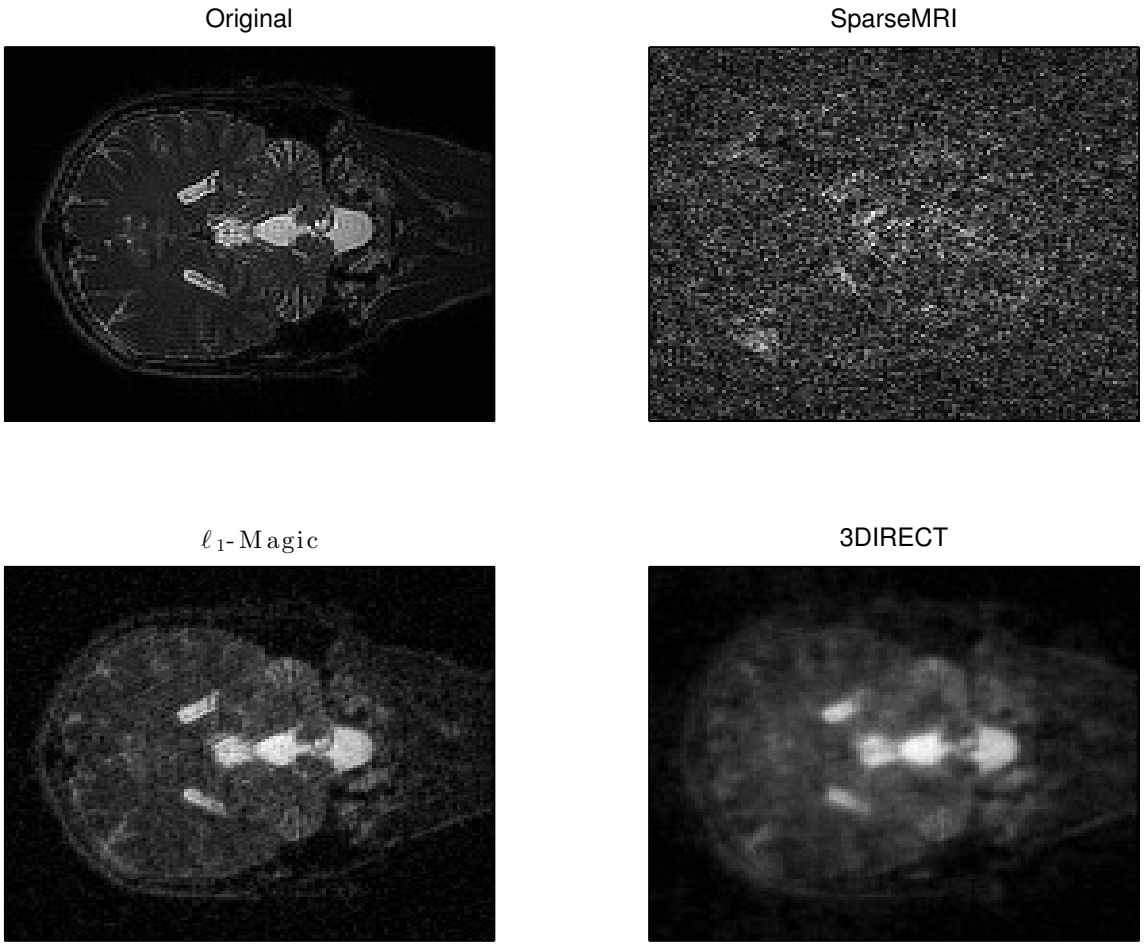

Figure 6.7. Initial ℓ_2 image using 15% of 2D FFT Peak Bins (slice 87). As expected the starting image for ℓ_1-Magic has the clearest picture. This is because the 15% of the bins selected, were based on its linear operator. 3DIRECT suffers from some blurring of the image, and SparseMRI has poor results.

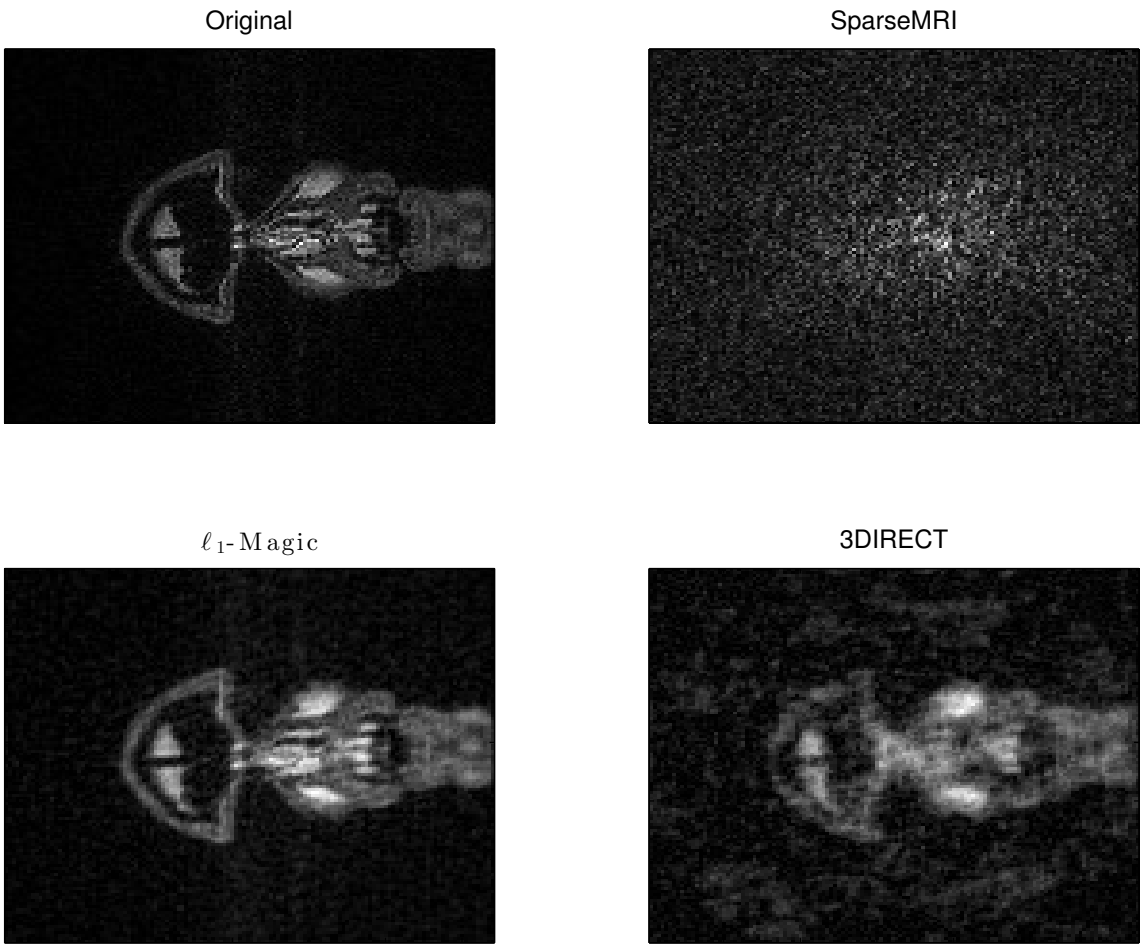

Figure 6.8. Initial ℓ_2 image using 15% of 2D FFT Peak Bins (slice 25). 3DIRECT still captures some of the image and the end slices, but suffers from noise.

These results are used as the initial guesses for each method respectively. Notice, SparseMRI has several noisy samples. ℓ_1-Magic has the best results as expected since this peak sample scheme was designed around its method. Lastly, the 3DIRECT method suffers from smearing of the image and noise at the end slices. This is because the 2D peaks partially coincide with the 3D peaks but miss key parts of the data. This will be obvious later when we see how the peak energy of the 3D FFT spreads.

Another way to view the ℓ_2 results is to analyze the PSNR of the image. Recall PSNR gives a measure of how well the image compares to the original data. Figure 6.9 shows how each of the slices of the image above compare to the fully sampled data.

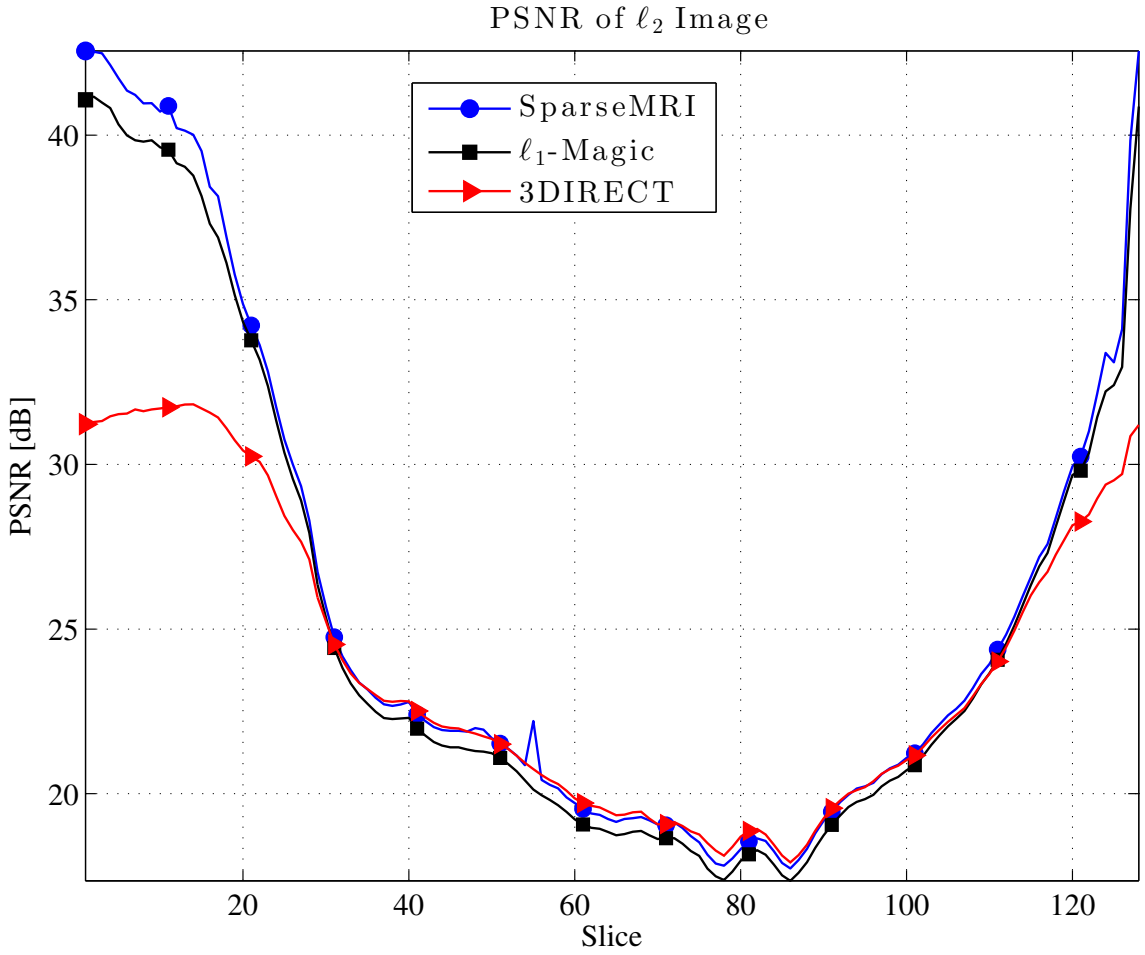

Figure 6.9. Comparison of PSNR of the ℓ_2 starting image using 15% of 2D FFT Peak Bins. Large PSNR should correspond to better image quality. The importance of these results is that it confirms PSNR is not always a good metric. For instance, we just saw a picture of slice 87 where ℓ_1-Magic had the clearest image; however the PSNR at slice 87 would indicate the ℓ_2 images should be similar. It is important to always rely on human perception.

The results above would indicate that ℓ_1-Magic and SparseMRI has the clearest results near the end slices, and has comparable results with the 3DIRECT method for the middle slices. However, when we look back to Figure 6.7, it is clear ℓ_1-Magic is clearly the superior image despite the fact at slice 87 they appear to have similar PSNR. This is important to understand that PSNR can be a measure for how well two images compare, but ultimately it is up to human perception. Keeping this in mind, Figure 6.10 and 6.11 display the reconstructed images.

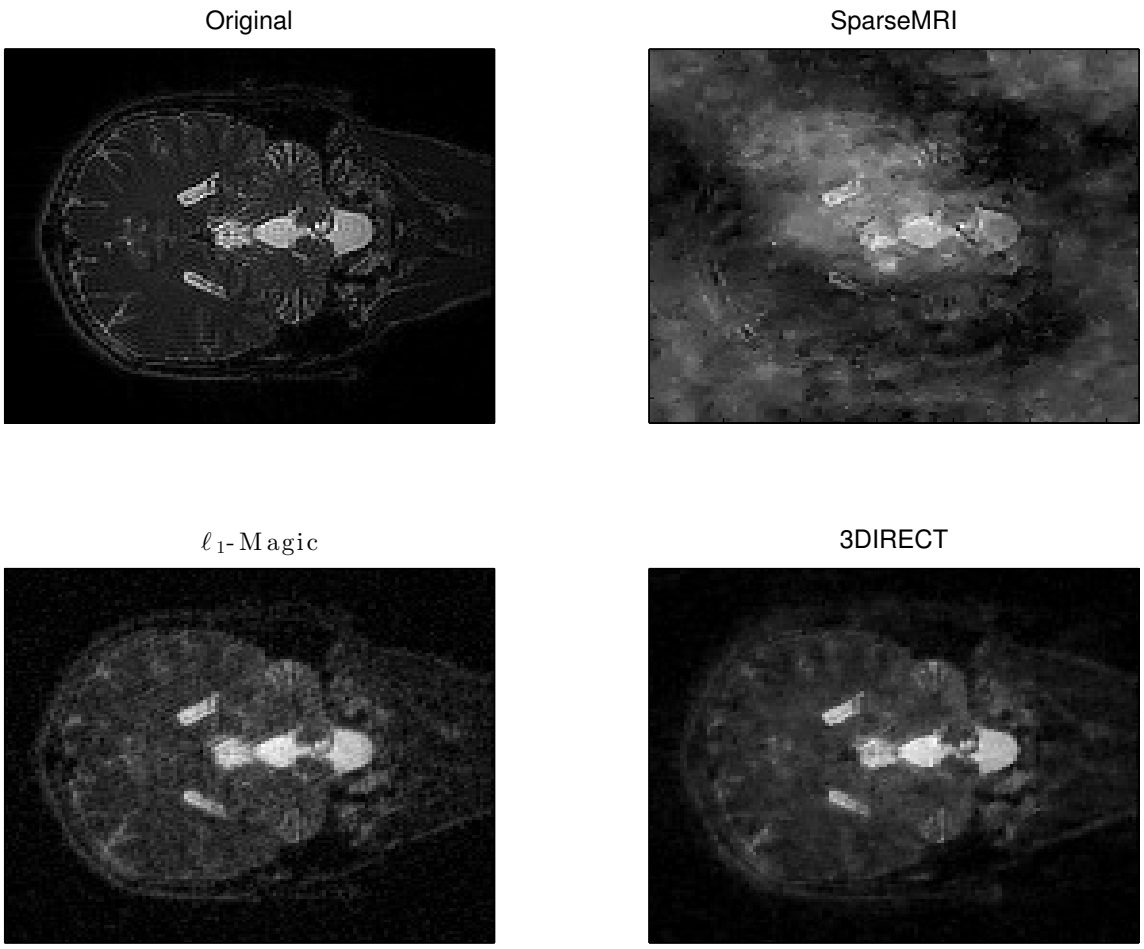

Figure 6.10. Reconstructed Image using 15% of 2D FFT Peak Bins (slice 87). These results show ℓ_1-Magic has a noise free image, but suffers from some graininess. SparseMRI was able to improve its image, but not enough to be useful. Finally 3DIRECT was able to minimize the smearing of the image when compared to Figure 6.7.

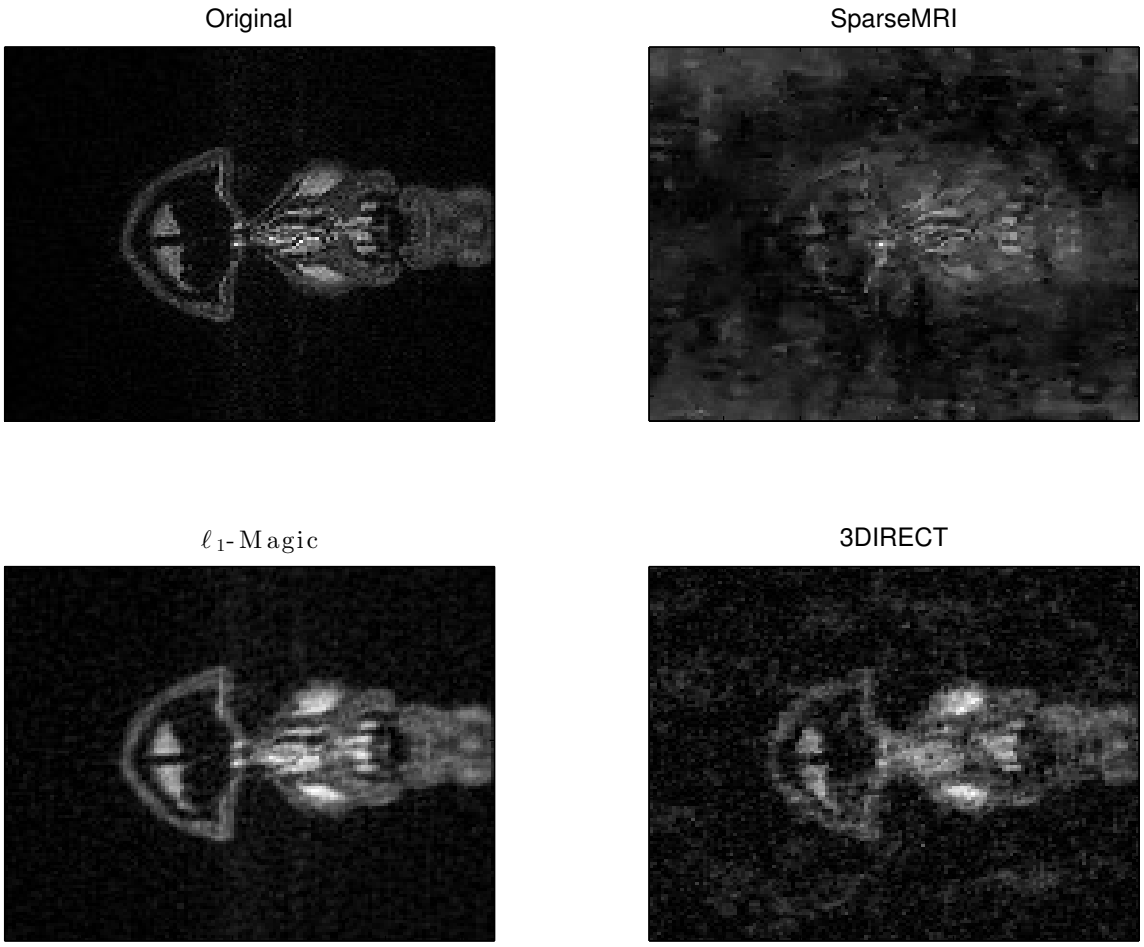

Figure 6.11. Reconstructed Image using 15% of 2D FFT Peak Bins (slice 25). For the end slices ℓ_1-Magic had the best results, while SparseMRI and 3DIRECT suffer from artifacts in the image.

It is clear that the SparseMRI method did not do a good job recovering the image at this low sampling percentage. The ℓ_1-Magic method obviously did the best as it had the best starting point. The 3DIRECT method, while able to reduce some of the noise, is still not a clear image near the end slices, but has comparable results to ℓ_1-Magic in the middles slices. In addition, 3DIRECT has consistent PSNR of

the recovered images. This is reflected in observing the PSNR of the reconstructed images in Figure 6.12.

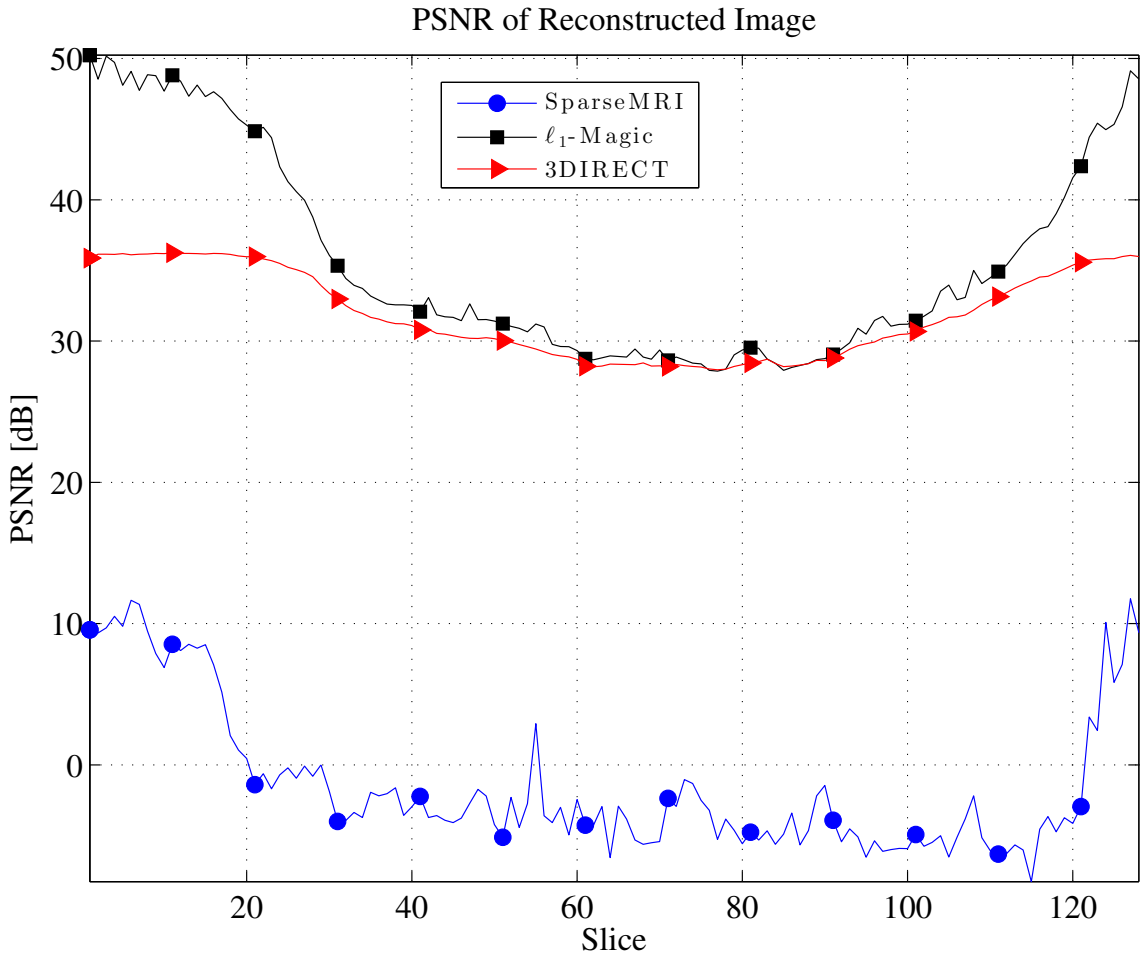

Figure 6.12. Comparison of Reconstructed PSNR using 15% of 2D FFT Peak Bins. Large PSNR should correspond to better image quality. Lastly, comparing the recovered image's PSNRs shows the 2D peak scheme benefited ℓ_1-Magic, specifically at the end slices. Meanwhile, the 3DIRECT method had comparable results in the middle slices.

In addition to PSNR and human perception, we can also use SSIM to compare our image results. Using the method outlined earlier, Figure 6.13 shows the SSIM for this simulation.

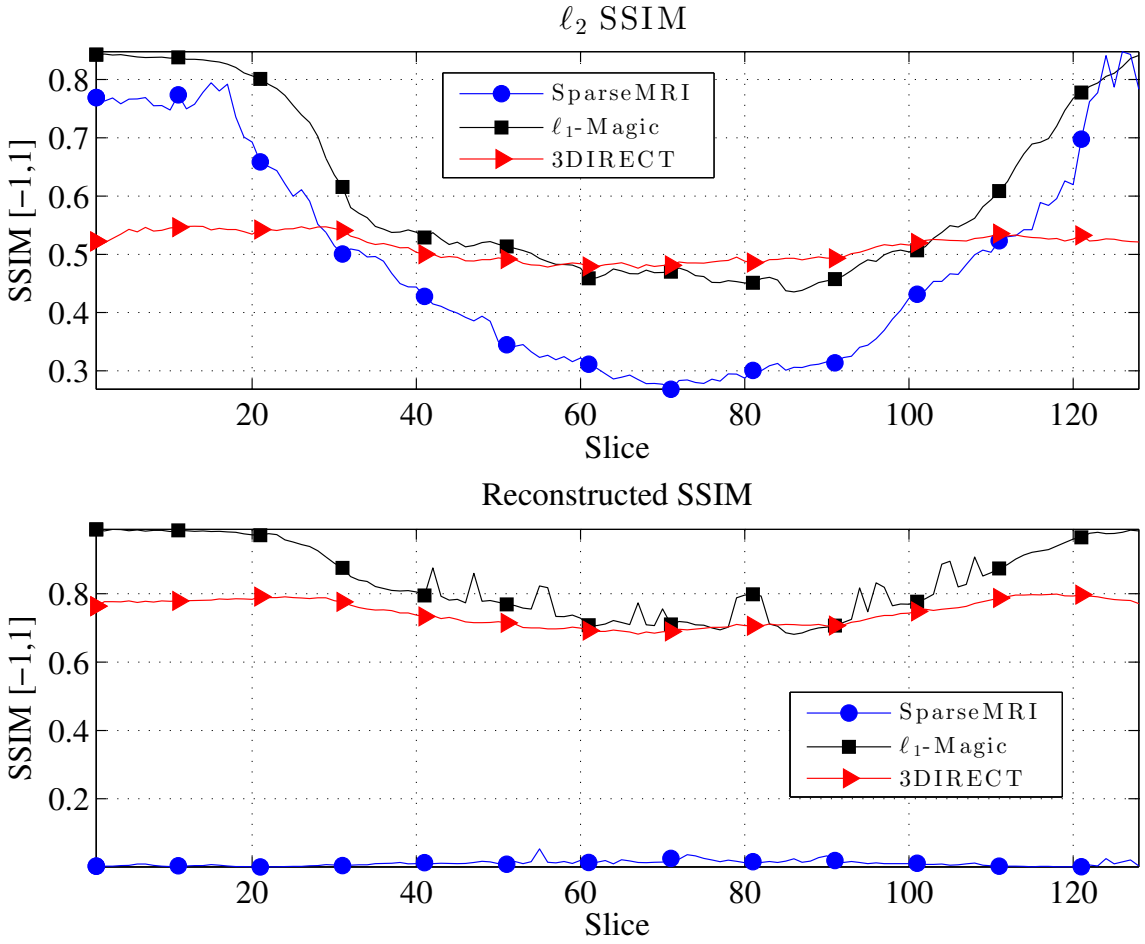

Figure 6.13. Comparison of Reconstructed SSIM using 15% of 2D FFT Peak Bins. Higher SSIM should mean the image is closer in structure to the original image. Again, ℓ_1-Magic consistently has the best results under this scheme, while the 3DIRECT method has comparable recovered images in the middle slices.

The SSIM result further confirms that the 2D peak sample scheme allows the ℓ_1-Magic to have the best performance.

Overall Observations: Using the 2D peaks as a sample scheme resulted in ℓ_1-Magic consistently providing the clearest image as we expected. The 3DIRECT method however was able to provide comparable results near the middles slices.

6.5 Sample Scheme: 15% of 3D Peaks

Similarly, to the previous 2D Peak FFT sample scheme, this scheme is intended to show how well the 3DIRECT method can perform. Using MATLAB's n-dimensional FFT, a 3D FFT was performed on the fully sampled data set, and then the top 15% in magnitude of the n^3 samples were kept. The results of this is the mask depicted in Figure 6.14 below.

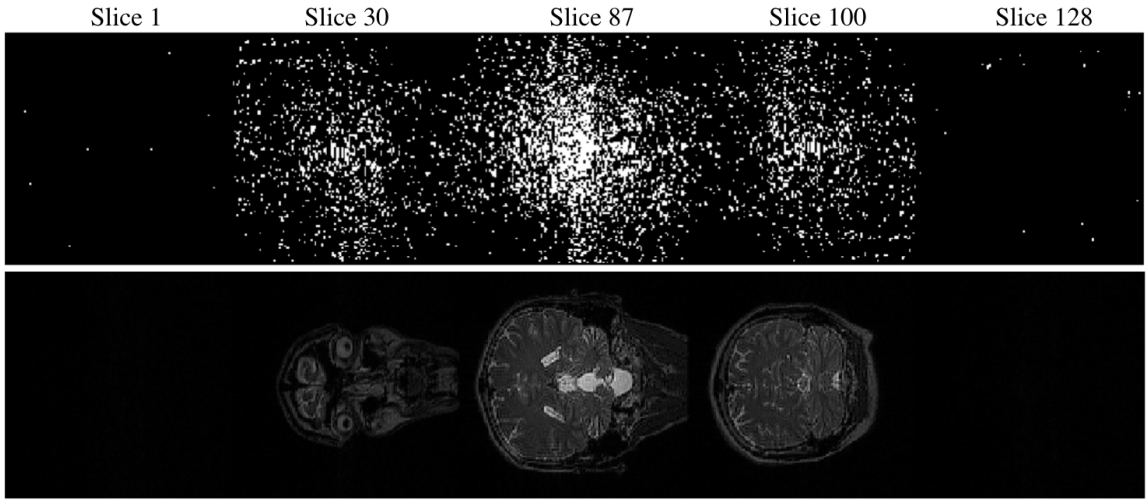

Figure 6.14. 15% of 3D FFT Peak Bins. The first row shows the location of the bins and how it varies over the slices. The second row shows what the image in the corresponding slice above looks like in the image domain. We now select a sample scheme that should benefit the 3DIRECT method. This scheme performs a 3D FFT on the data, and keeps the top 15% of the n^3 bins.

Similarly, to the 2D case, the peak 15% samples are concentrated in the middle of each slice. However, now the majority of the peak bins are located in the center slices, and more wildly spread apart. This is because the FFT in the tubal dimension compresses and mixes slices of the image data to the DC bin centered at the middle of the $128 \times 128 \times 128$ cube. One might expect, since the data is further compressed, that more information can be captured within the 15% peak bins. To see this, let's examine the ℓ_2 results in Figures 6.15 and 6.16. Note, ℓ_1-Magic and SparseMRI still used a 2D FFT, whose subsampling coincides with the results from selecting the top 15% in magnitude of the 3D FFT samples. While, we recognize this is unfair to sample 2D frequency bins based on 3D FFT information, the purpose of this scheme is to examine how well 3DIRECT can perform using maximal information, and observe the effects on the 2D methods, where again SparseMRI suffers from using the absolute value of the complex observations.

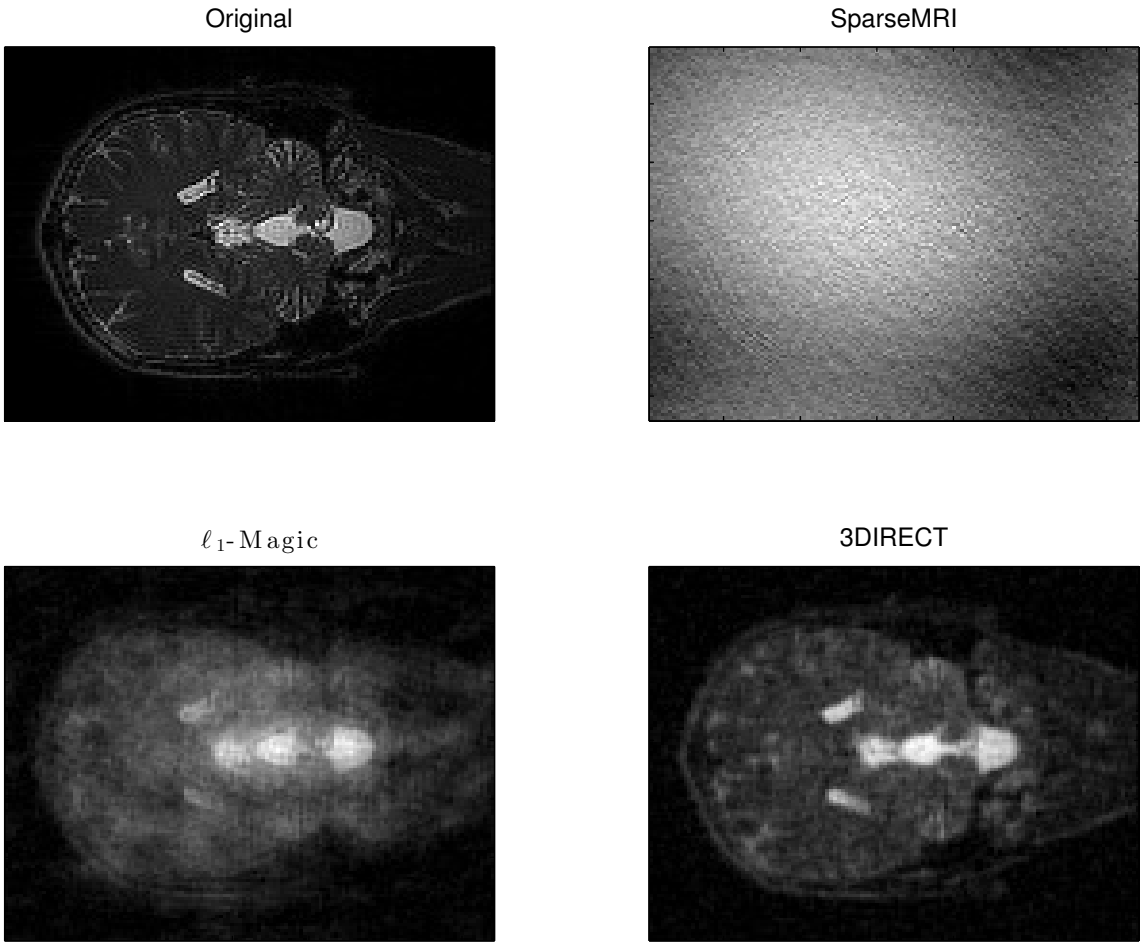

Figure 6.15. Initial ℓ_2 image using 15% of 3D FFT Peak Bins (slice 87). As expected the 3DIRECT method has the clearest starting image.

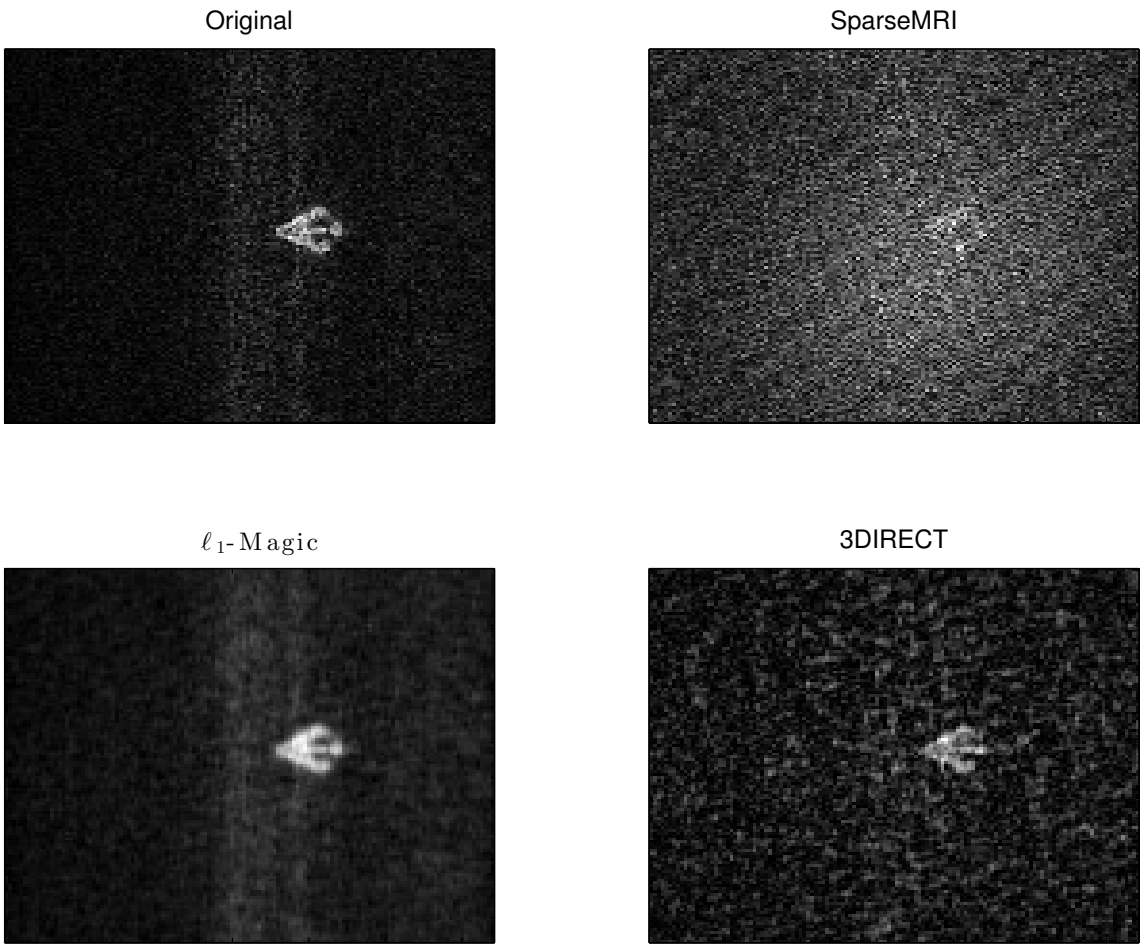

Figure 6.16. Initial ℓ_2 image using 15% of 3D FFT Peak Bins (slice 10). However, an interesting observation is that the 3DIRECT method suffers from some noise in the earlier and later slices.

Figure 6.15 makes it obvious that the 3D peak sample scheme is non-ideal for the 2D methods during the middle slices, and Figure 6.16 shows the 3D sample scheme captures the end slices with good image quality. PSNR results are shown in Figure 6.17. Notice, this is a case where PSNR is a poor metric of how clear an image appears. It shows SparseMRI has the highest PSNR, however it's image is illegible.

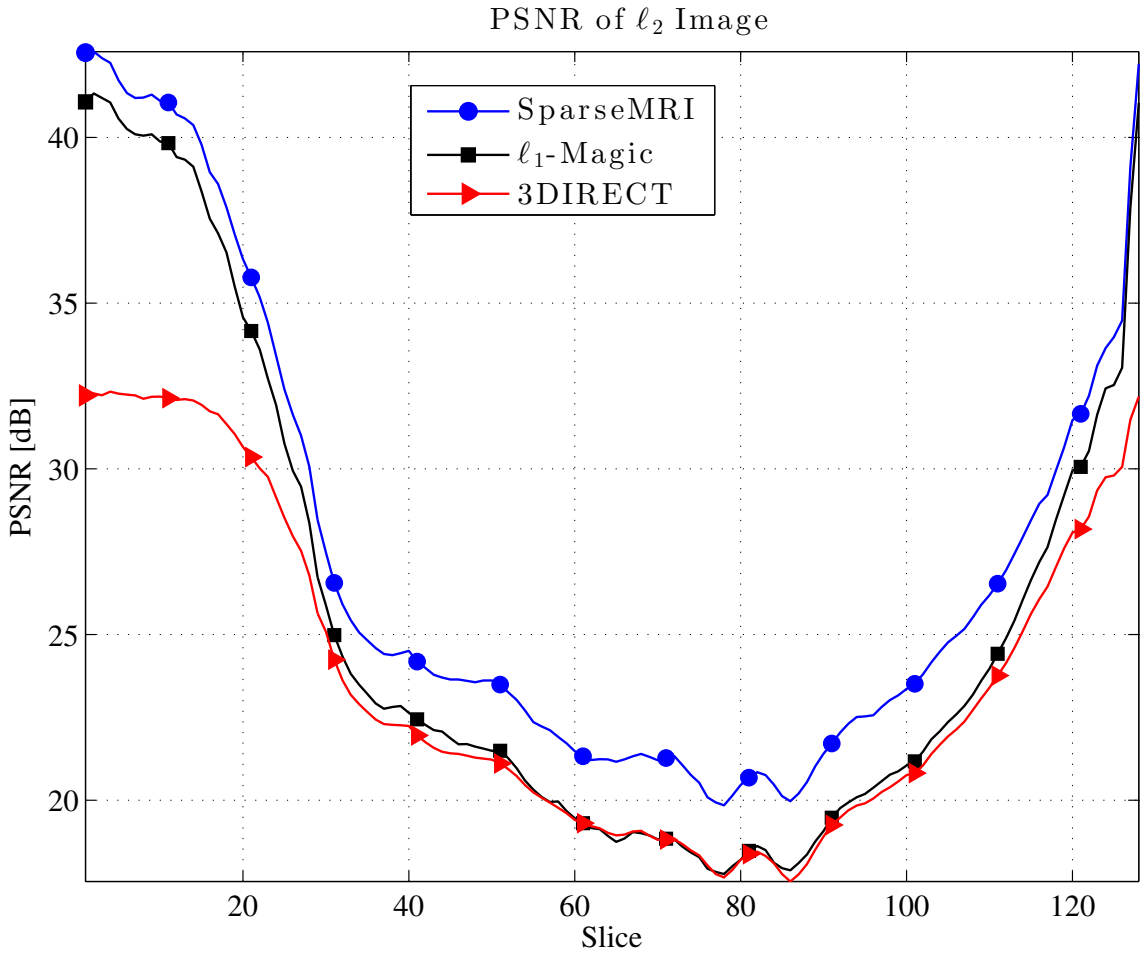

Figure 6.17. Comparison of PSNR of the Initial ℓ_2 Image using 15% of 3D FFT Peak Bins. Large PSNR should correspond to better image quality. The results here, show PSNR is a poor metric for how clear the starting images appear. Thus, we must rely on human perception.

Thus, we will ignore PSNR as a useful metric for this case and instead examine the recovered image for an early and middle slice.

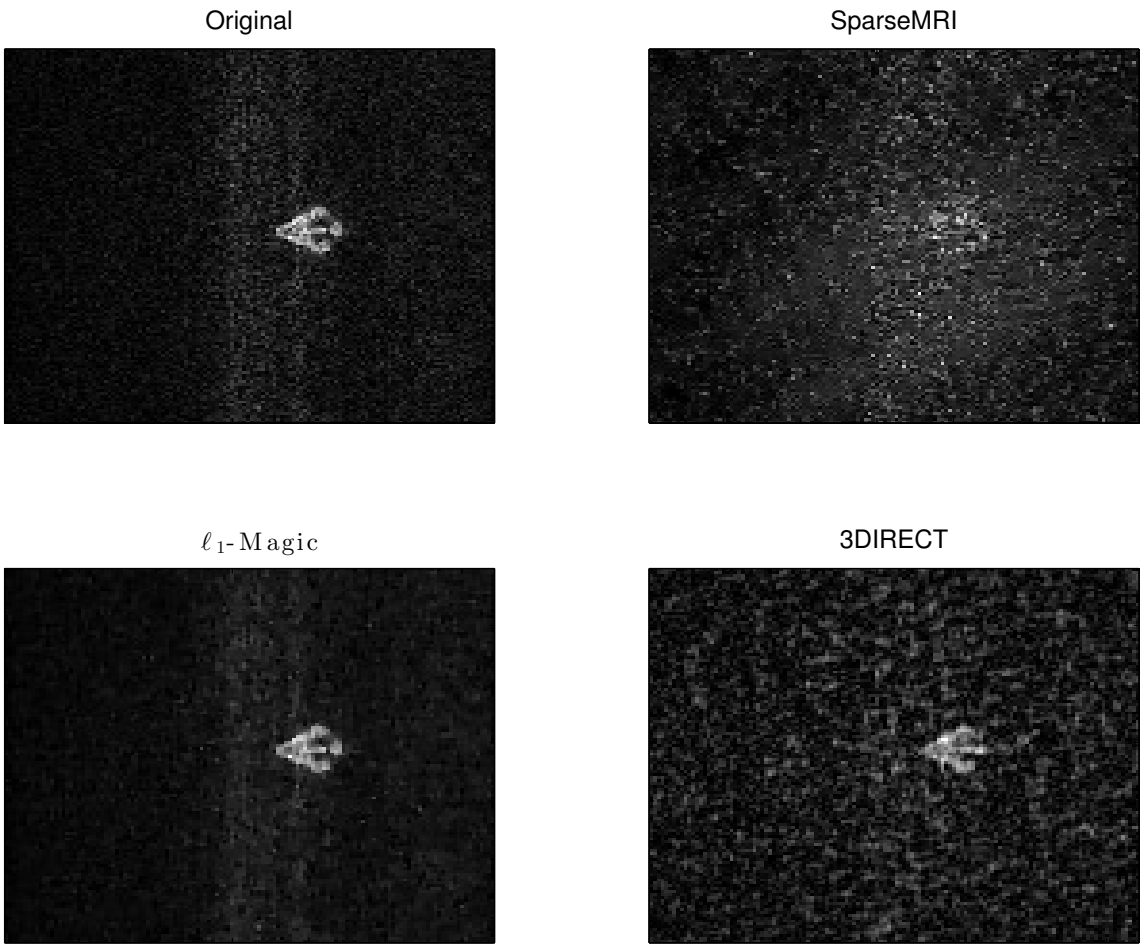

Figure 6.18. Reconstructed Image using 15% of 3D FFT Peak Bins (slice 10). Even after reconstruction, 3DIRECT still suffers from some noise, while ℓ_1-Magic seems to have clearer results. This is due to mixing of frequencies down the tubal direction.

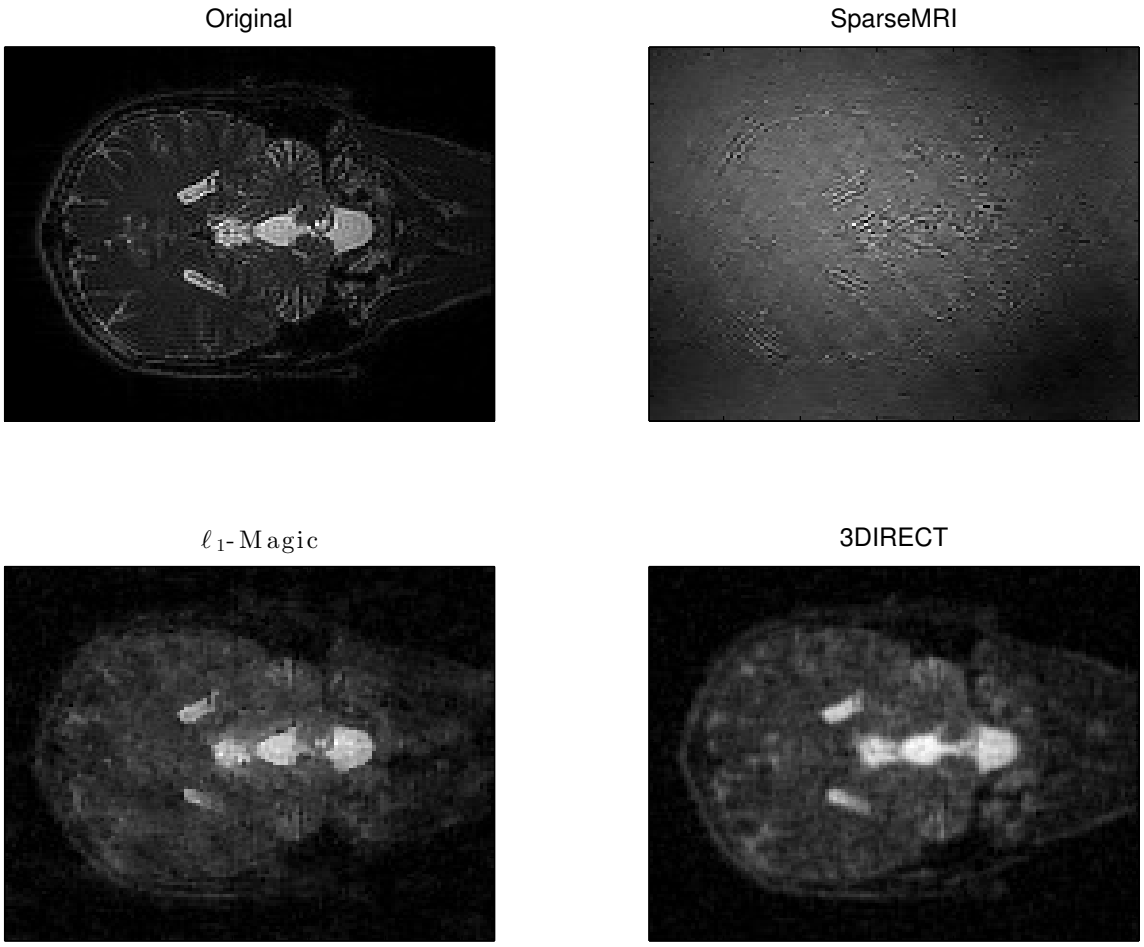

Figure 6.19. Reconstructed Image using 15% of 3D FFT Peak Bins (slice 87). In the middle slices 3DIRECT is clearly the best method to use given the 3D peak scheme.

As expected, the 3DIRECT image is the best in the middle slices and had some distortion for the end slices. This is due to the fact that the FFT in the tubal direction mixes the large cross-sections of the brain with the smaller ones. Meanwhile ℓ_1-Magic looks good at the early slices because it benefits from the way the 3D FFT samples are shifted about zero frequency. When looking at the SSIM metric we get the results in Figure 6.20.

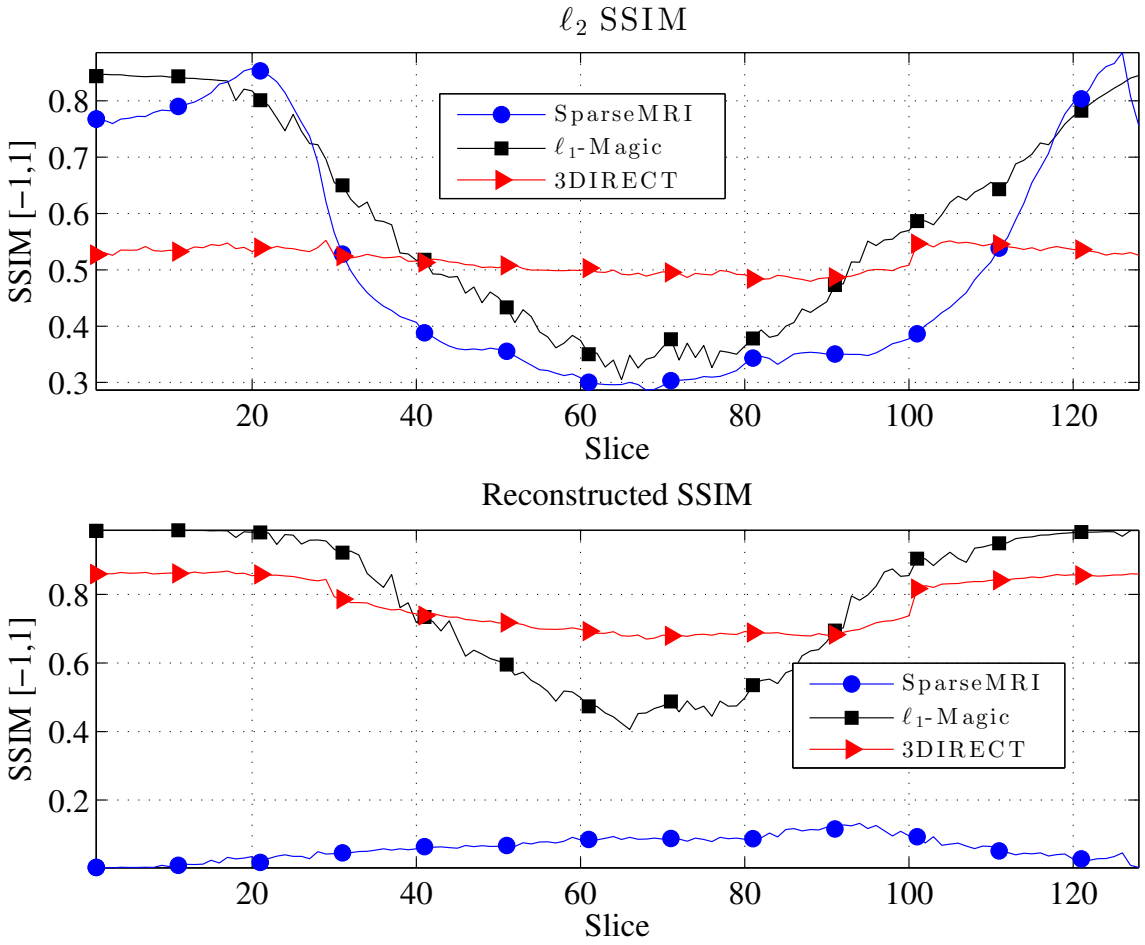

Figure 6.20. Comparison of Reconstructed SSIM using 15% of 3D FFT Peak Bins. Higher SSIM should mean the image is closer in structure to the original image. The SSIM of the image confirms what we observed earlier. That is, 3DIRECT outperforms the other 2D methods in the middle slices, and has challenges with mixing of the tubal frequencies for the end slices.

The recovered SSIM indicates that the 3DIRECT method is clearly all around better, but slightly trails ℓ_1-Magic at the end slices. This is due to the fact that the 3D peak sample scheme using 2D FFT operators samples the end slices at high rates, resulting in better image recovery.

Overall Observations: The 3D peak scheme allows the 3DIRECT method to have superior results. Both the SparseMRI and ℓ_1-Magic only benefit at the end slice due to how the 3D peak FFT frequencies coincide with the 2D frequency bin locations. It is also clear that the 3DIRECT method is the most resilient when it comes to selecting 15% peaks as the sample scheme.

6.6 Sample Scheme: 2D vs. 3D Favorable Peaks for Varying Sampling Percentages

Now that we've observed the impacts of sampling 2D and 3D peaks, one might ask: how does a 2D method using a 2D favorable sampling scheme compare to a 3D method using a 3D favorable sampling scheme? That is what this section aims to do.

Presented here are the results for ℓ_1-Magic using the peaks based on the 2D FFT for a specified percentage. This same percentage is also used to grab the 3D peaks based on the 3D FFT to be used in 3DIRECT. We used various sampling percentages, and observed how the SSIM of the recovered image changed.

In Figure 6.21, we display the SSIM of the recovered image for ℓ_1-Magic under different sampling percentages against one another. The same was done for 3DIRECT. Both methods saw improvement in SSIM as sampling percentage of the respective peak bins increased. Specifically ℓ_1-Magic had consistent SSIM at the slices near the end of the 3D image despite the sampling percentage. It also had a clear dip in SSIM and image quality near the middle slices. This is expected as the middle slices require higher a sampling percentage to capture the image. Further, ℓ_1-Magic had lots of variation in its SSIM.

In contrast, 3DIRECT saw improvement in SSIM for all the slices as sampling percentage increased. It also did not suffer from the dips in SSIM in the middle slices as we saw in ℓ_1-Magic. This is because the 3D peaks of the FFT contain tubal

frequency information which provides information on multiple slices, and not just the slice its k^{th} indices indicates. This also contributes to the smoothness of the SSIM.

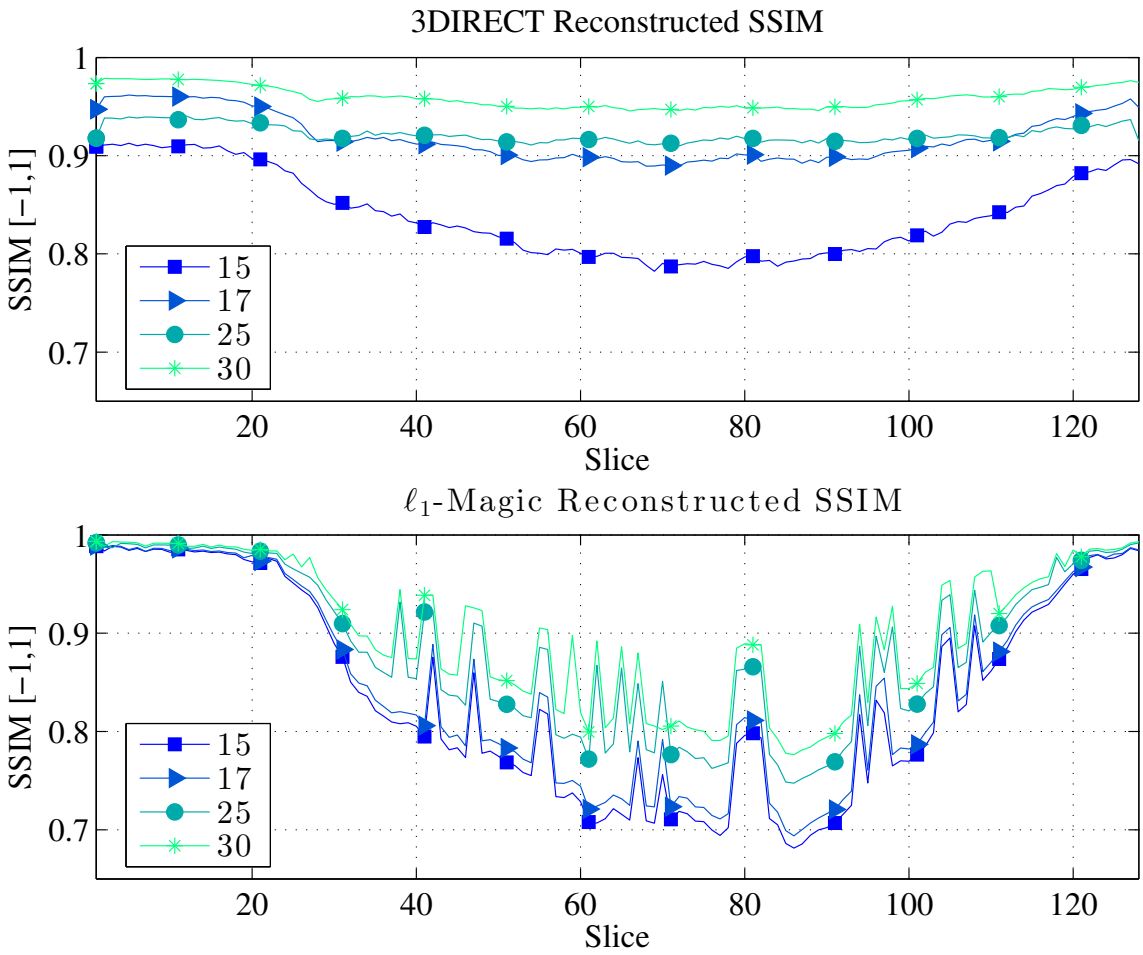

Figure 6.21. 2D vs. 3D Favorable Peaks for Varying Sampling Percentages. SSIM improves as sampling percentage increases. 3DIRECT has smooth SSIM and benefits from the multi-slice frequency information encoded in the 3D peak bins. Contrastingly, ℓ_1-Magic SSIM suffers in the middle slices due to the fact its 2D peaks require larger sampling percentages as the image cross-section increases.

We now examine how each of the methods did relative to one another for the specified sampling percentages in Figure 6.22. It was observed that ℓ_1-Magic per-

formed better at the end slices. This will become obvious later in section 6.8 when we display the impacts of using the tubal frequencies in the 3DIRECT method. However, we note here that 3DIRECT outperformed ℓ_1-Magic in over 65% of the slices regardless of sampling percentage. Further, 3DIRECT also had more consistent SSIM over the slices making the recovered image quality consistent for the end user. It also had higher minimum SSIM over ℓ_1-Magic minimum SSIM for the recovered image for all the sampling percentages.

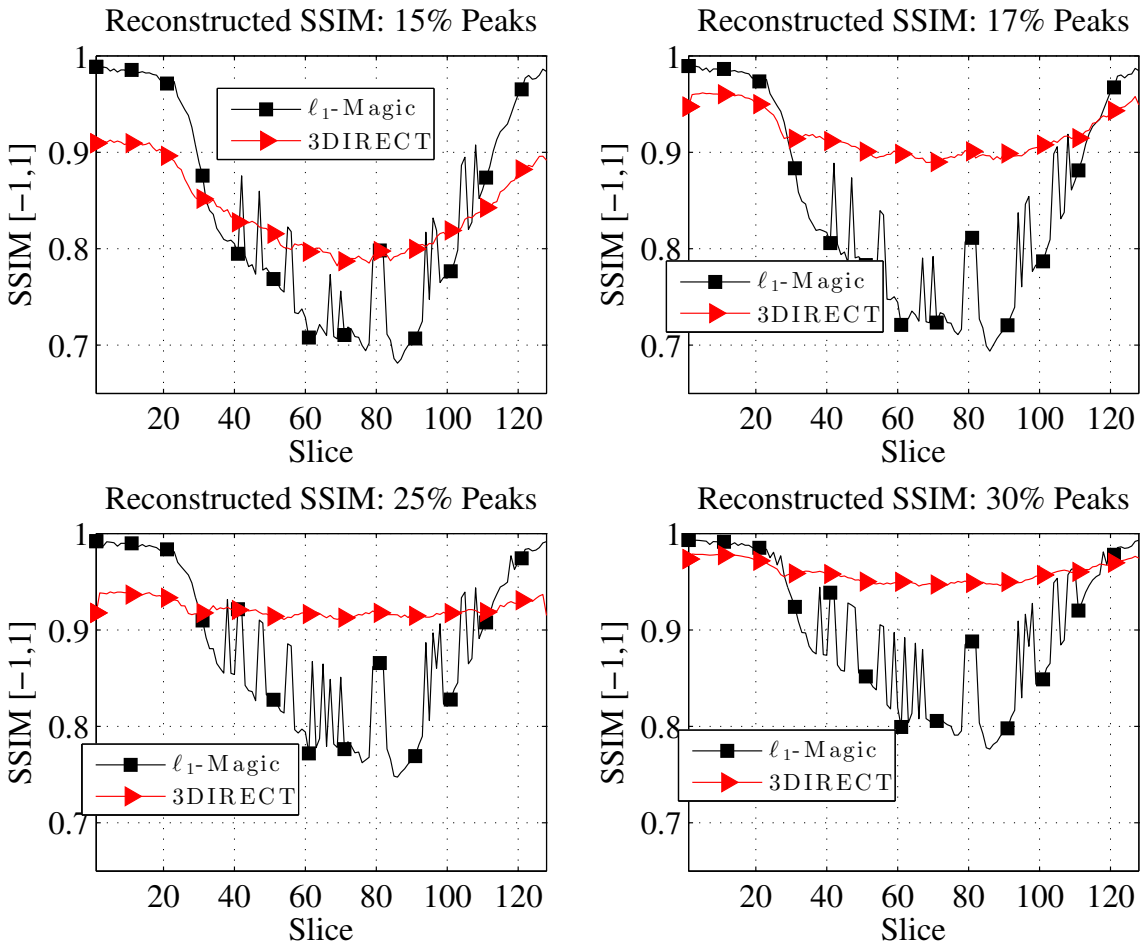

Figure 6.22. 2D vs. 3D Favorable Peaks for Varying Sampling Percentages. The SSIM of the recovered image for 3DIRECT is more robust to changing sampling percentages.

6.7 Sample Scheme: A Practical 2D Implementation

Now that we have examined two sample schemes, although impractical, to understand the effects of the mask on image recovery, next we will look at one practical implementation of a 2D sample scheme [24, 26, 51]. The sampling scheme was proposed in [34] and it uses a radial sample scheme which concentrates on collecting 2D

center frequencies and some other adjacent frequencies [12]. This can be viewed in Figure 6.23.

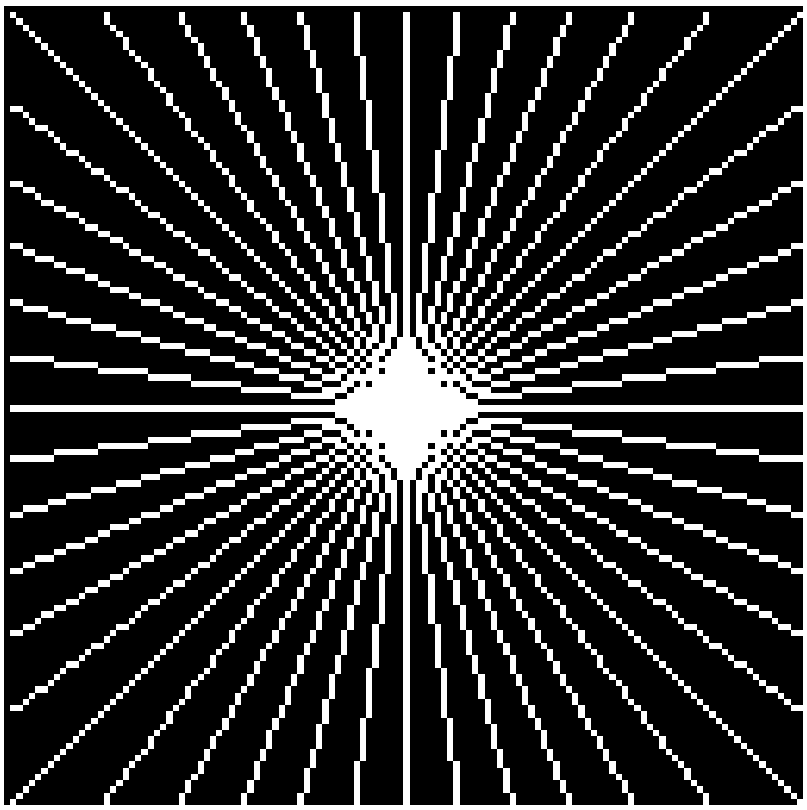

Figure 6.23. A Practical 2D Sample Scheme. The white pixels correspond to the sampled k-space pixels.

The sample scheme in Figure 6.23 was then replicated for each slice, and the location of the bins to be sampled were stored in the 3D mask. For SparseMRI and ℓ_1-Magic, each method performed 2D FFTs and used the information from the corresponding slice in the mask. The 3DIRECT method performed a 3D FFT, and kept the same samples as specified by mask all at one time. The following reconstructed images were achieved in Figure 6.24 for slice 87.

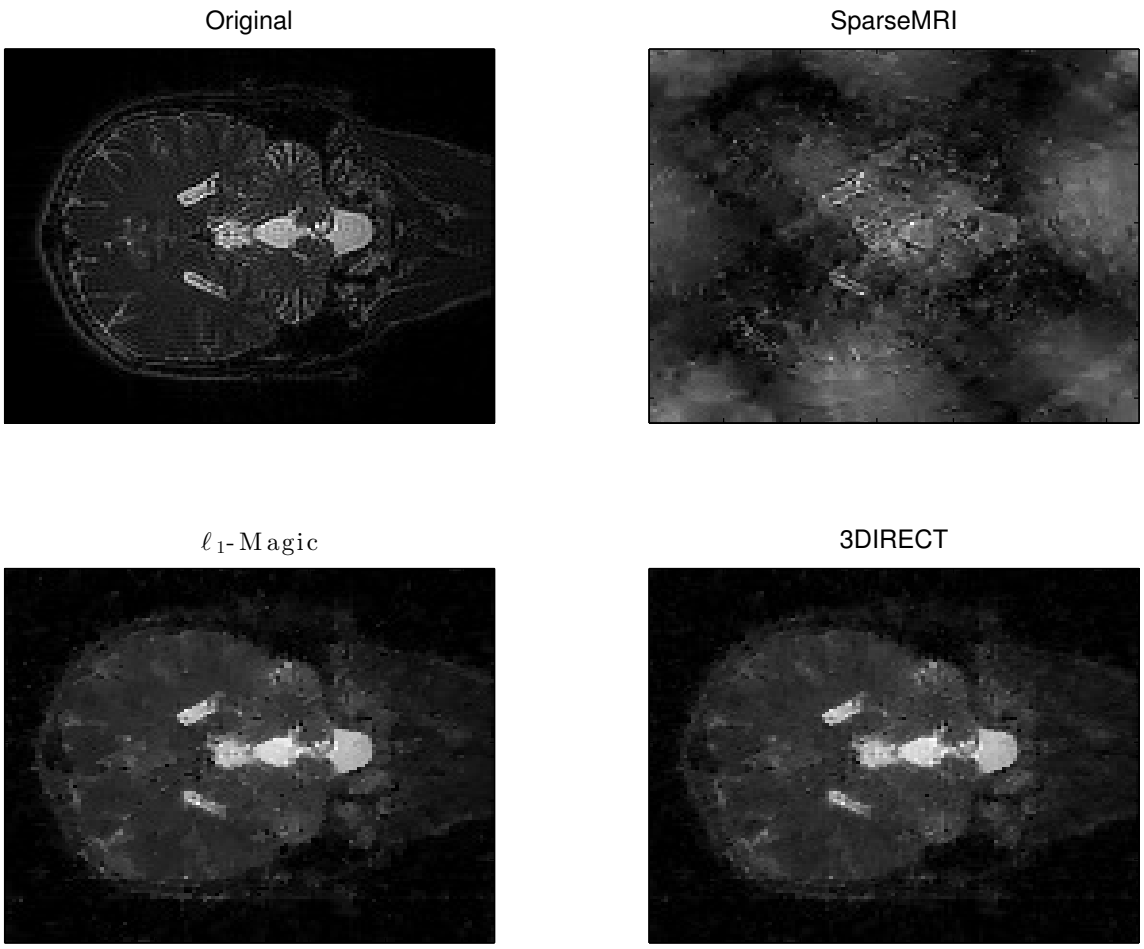

Figure 6.24. Reconstructed Image using 2D Radial Scheme at 17% (slice 87). Both ℓ_1-Magic and 3DIRECT appear to have similar recovered images for this slice.

Clearly SparseMRI did not perform well under this scheme, meanwhile ℓ_1-Magic and 3DIRECT have similar performance. This is because the 2D Radial scheme is symmetric when doing 2D and 3D FFTs [23]. Thus, both methods start out with the similar initial ℓ_2 images. SSIM for all recovered slices is shown in Figure 6.25.

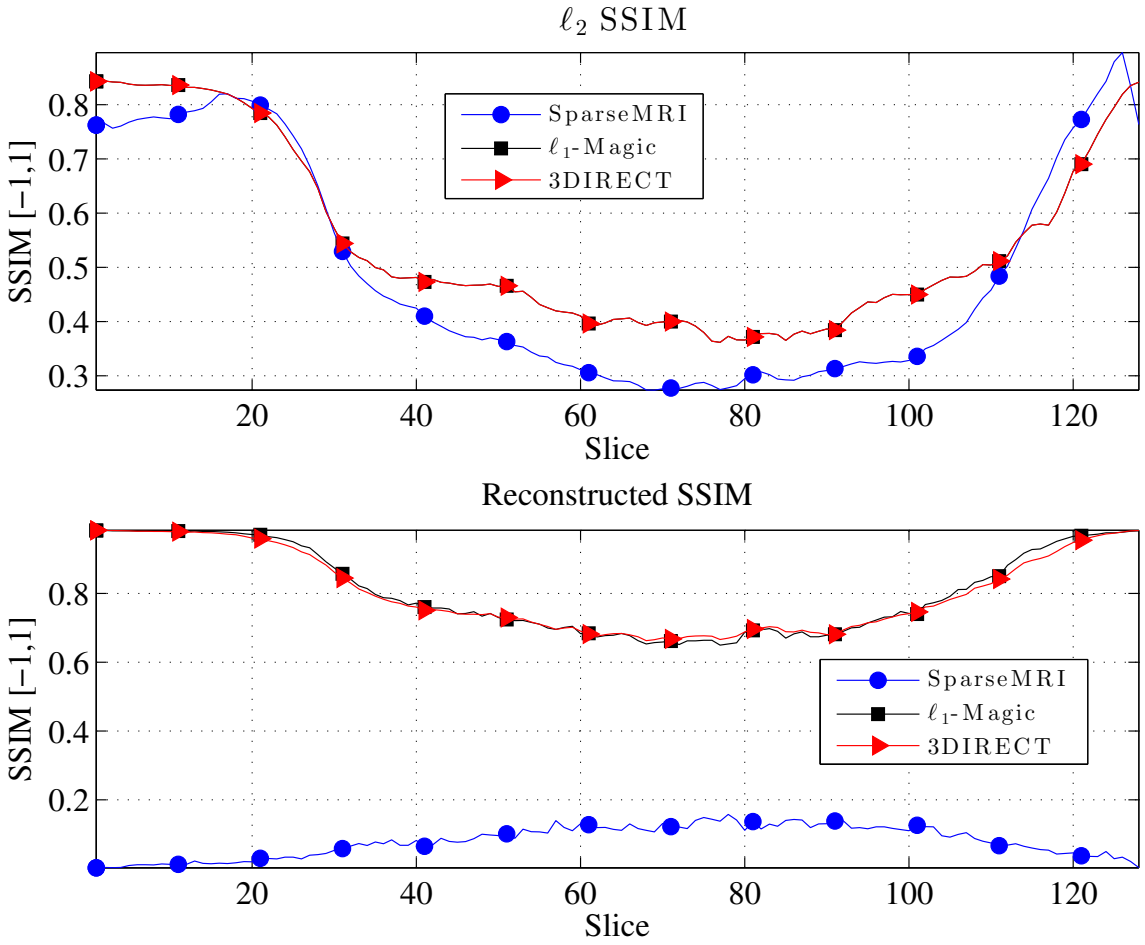

Figure 6.25. Comparison of Reconstructed SSIM using 2D Radial Scheme. Higher SSIM should mean the image is closer in structure to the original image. The SSIM of the recovered image indicates that ℓ_1-Magic and 3DIRECT have similar results for all slices, which indeed was confirmed in simulation via human perception.

Notice both ℓ_1-Magic and 3DIRECT have very comparable results in SSIM and human perception for every slice. The next logical question is then: which algorithm achieved the results fastest? To see the answer we examine Figure 6.26.

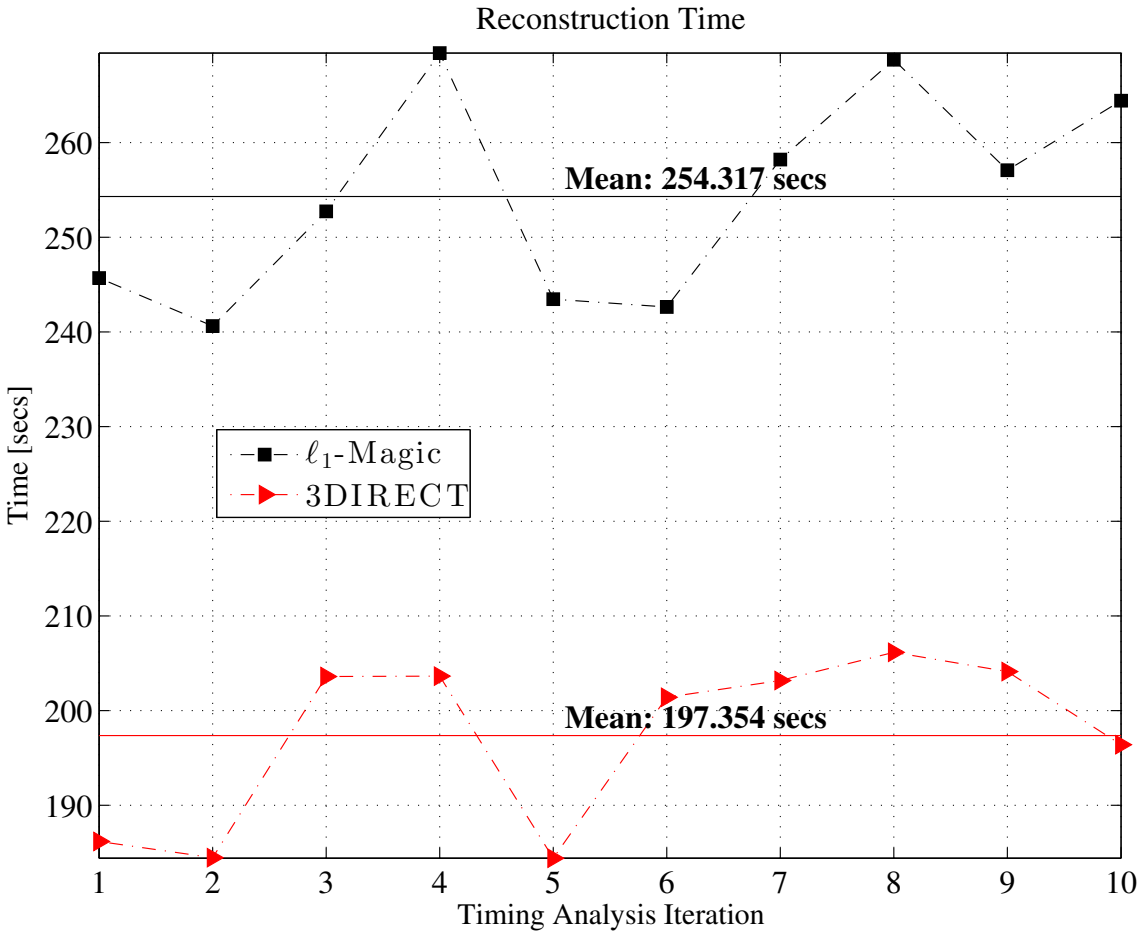

Figure 6.26. Timing Comparison of ℓ_1-Magic and 3DIRECT using 2D Radial Scheme. Under this scheme, ℓ_1-Magic and 3DIRECT achieved similar results, but 3DIRECT was 22% *faster*.

To achieve the results in the Figure 6.26, each optimization method was run with the same starting conditions 10 times in order to see on average how long each method took. It is clear that the 3DIRECT method achieved the same results under the 2D Radial scheme as ℓ_1-Magic but was 22% *faster in mean* and 33% *faster in minimum* times for this specific example. As the images get larger, this speed increase becomes more prominent.

Overall Observations: ℓ_1-Magic and 3DIRECT had comparable results under a practical 2D sampling scheme. However, 3DIRECT was 22% faster than ℓ_1-Magic, making it a more ideal choice for image recovery.

6.8 Sample Scheme: A Practical 3D Implementation

We will now examine a practical 3D sampling scheme. It's been observed that collecting frequencies near DC is important. For the 3D FFT this means concentrating our samples near the center of the cube [55]. However, we also know that the information isn't completely contained in the DC bins, as the images in the DC sample scheme lacks some detail. Thus, other frequencies must be included as well. Unfortunately we don't know which frequencies those are, so we attempt to implement a scheme that grabs a majority of the low 3D frequencies as well as some other frequencies [1, 27, 41]. To do this we use a 3D radial burst whose center starts in the center of the cube. An example of such a burst is depicted in Figure 6.27.

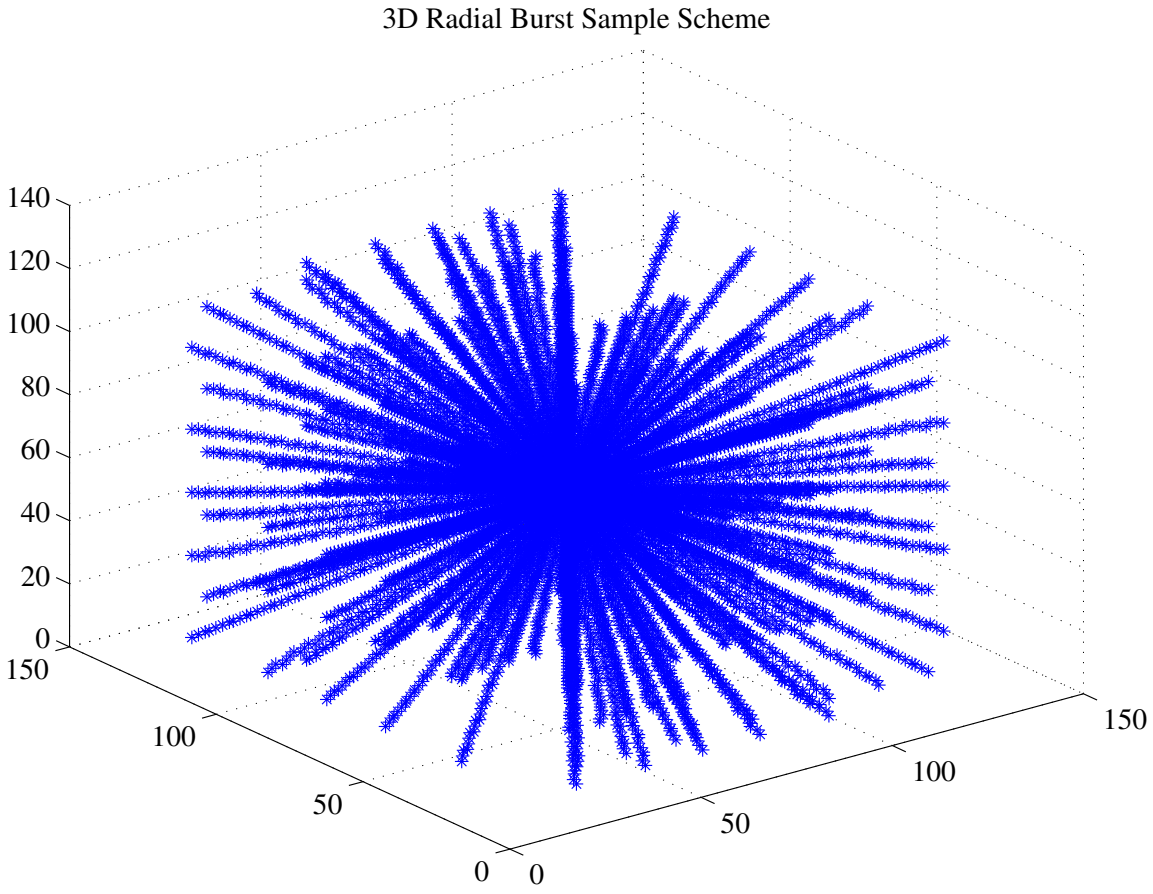

Figure 6.27. A Practical 3D Sample Scheme that Concentrates its Samples Near the 3D Lower Frequencies.

During the simulation it was observed that PSNR was not always a good metric to compare images. Thus, we include here some pictures of the recovered image and the SSIM results for a 3D radial burst at 17% sampling. Note, SparseMRI and ℓ_1-Magic still performed 2D FFTs per slice, but sampled the bins as indicated by the corresponding slice of the 3D mask. Again we note this is an unfair mask to apply to the 2D methods, as it was designed to highlight the 3DIRECT method.

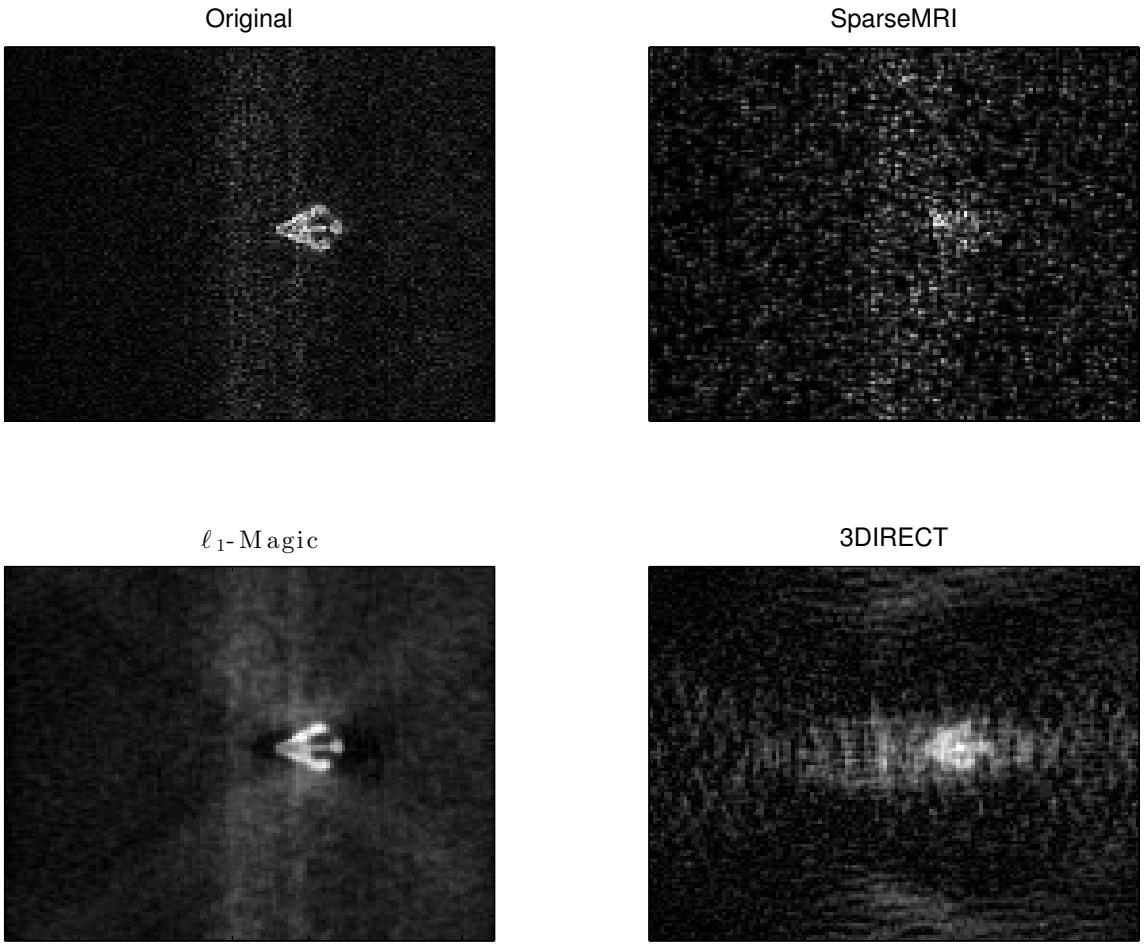

Figure 6.28. 3D Radial Recovered Image (slice 10). Similar to the 3D peak sample scheme, the practical 3D scheme causes 3DIRECT to have some mixing in earlier slices for this particular image.

Note, ℓ_1-Magic has a cleaner image than the 3DIRECT, whose tubal frequencies are causing smearing at the end slices.

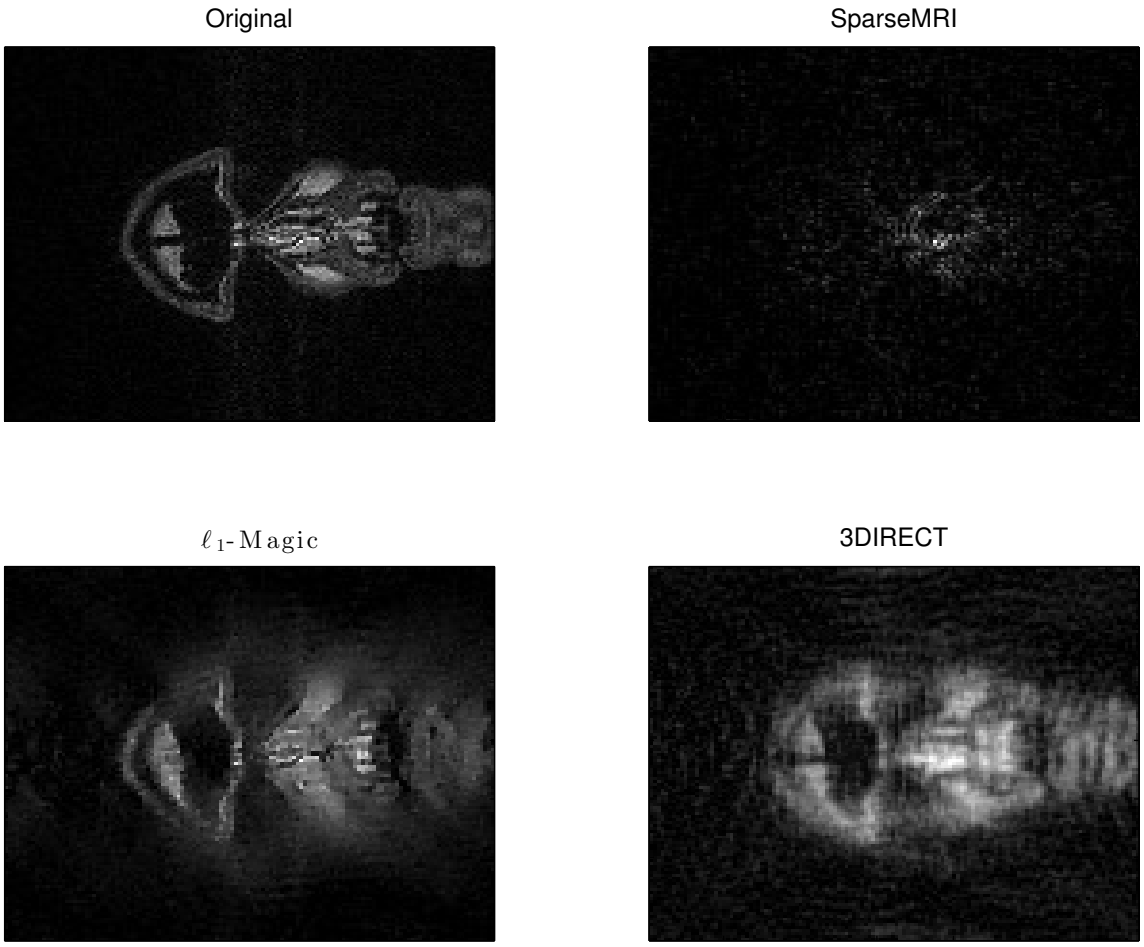

Figure 6.29. 3D Radial Recovered Image (slice 25). By slice 25, 3DIRECT is beginning to outperform the 2D methods.

Notice by slice 25 the ℓ_1-Magic and 3DIRECT method have comparable results. ℓ_1-Magic has less background noise but more smearing, and 3DIRECT has more background noise with more defined edges.

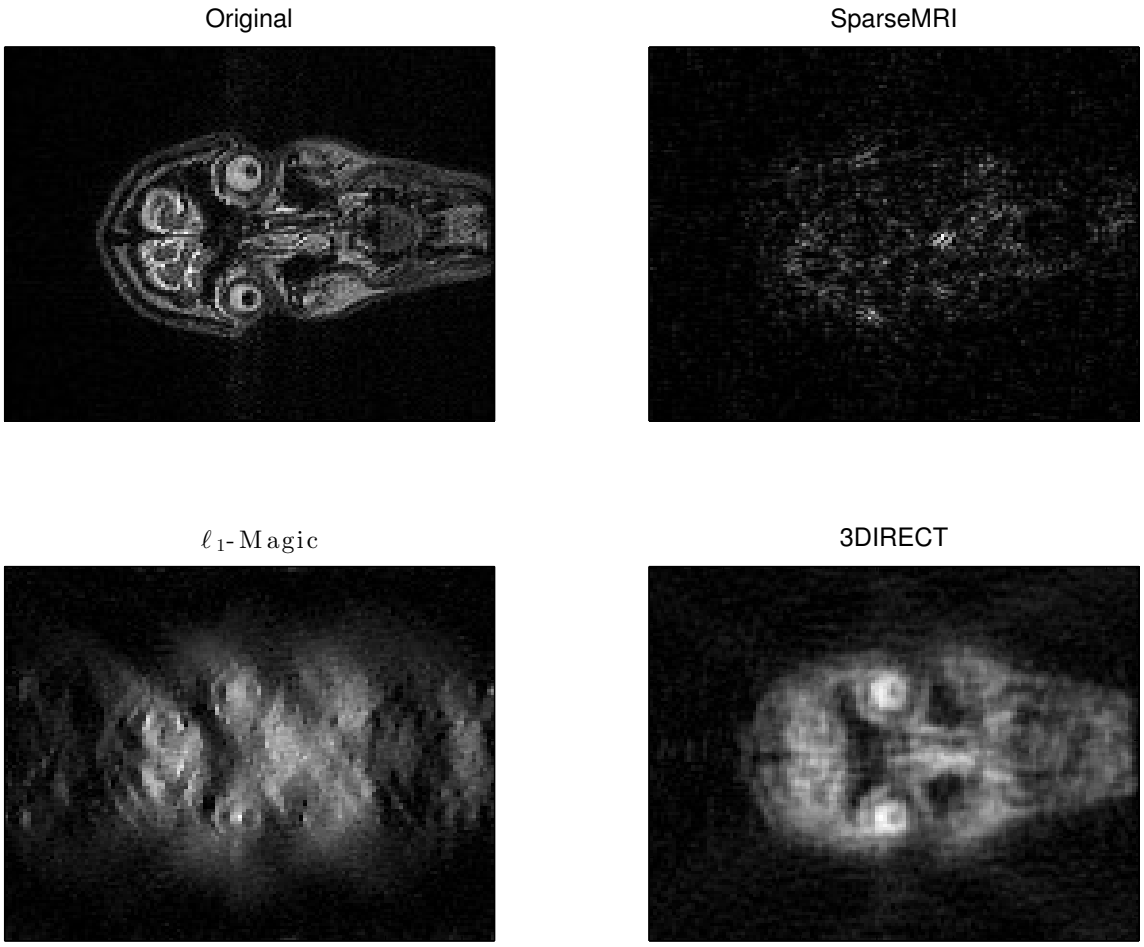

Figure 6.30. 3D Radial Recovered Image (slice 30). By slice 30, 3DIRECT is clearly the best image.

By slice 30 it is clear the 3DIRECT method has the best results and the 2D methods provide poor recovered images.

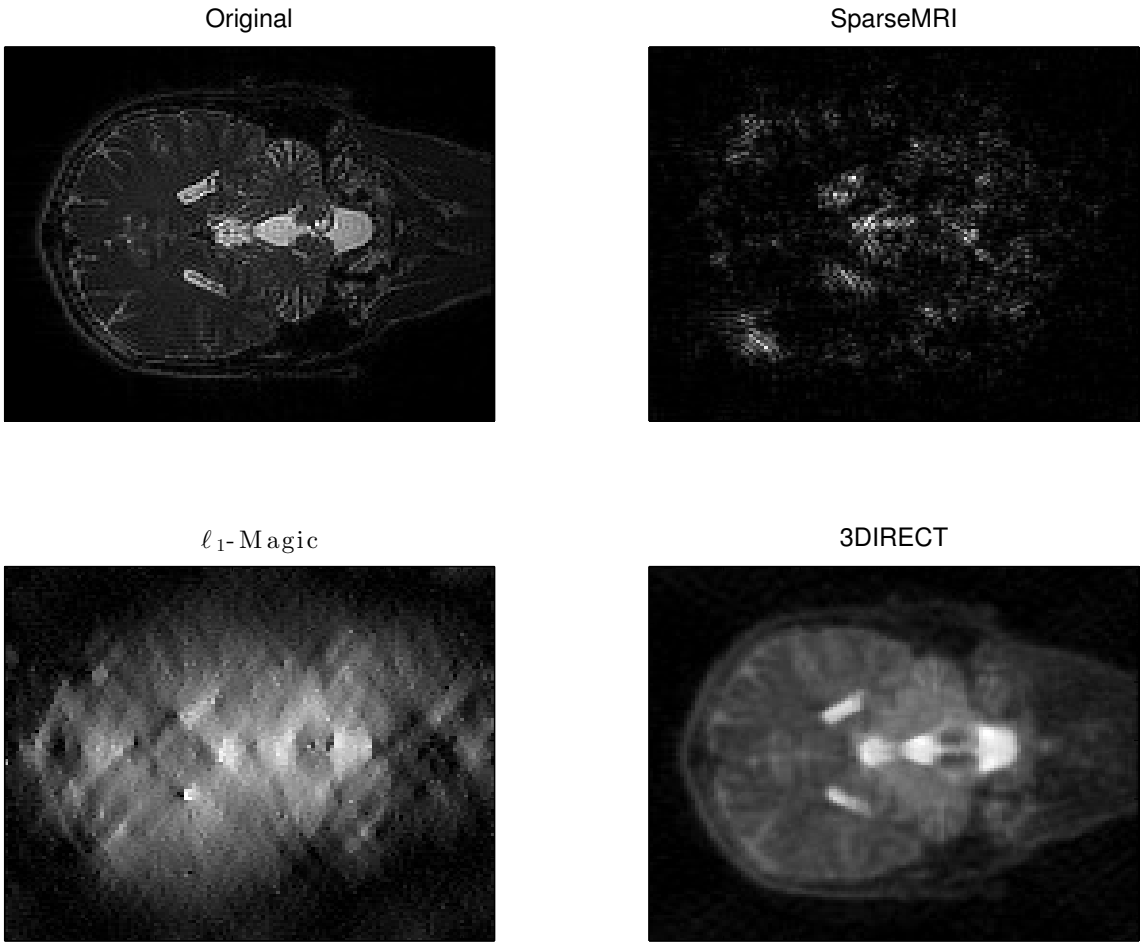

Figure 6.31. 3D Radial Recovered Image (slice 87). By the middle slices, 3DIRECT is able to exploit physical similarities between slices and recover the image with good quality.

Finally, by the middle slices, the 3DIRECT method has the best recovered image.

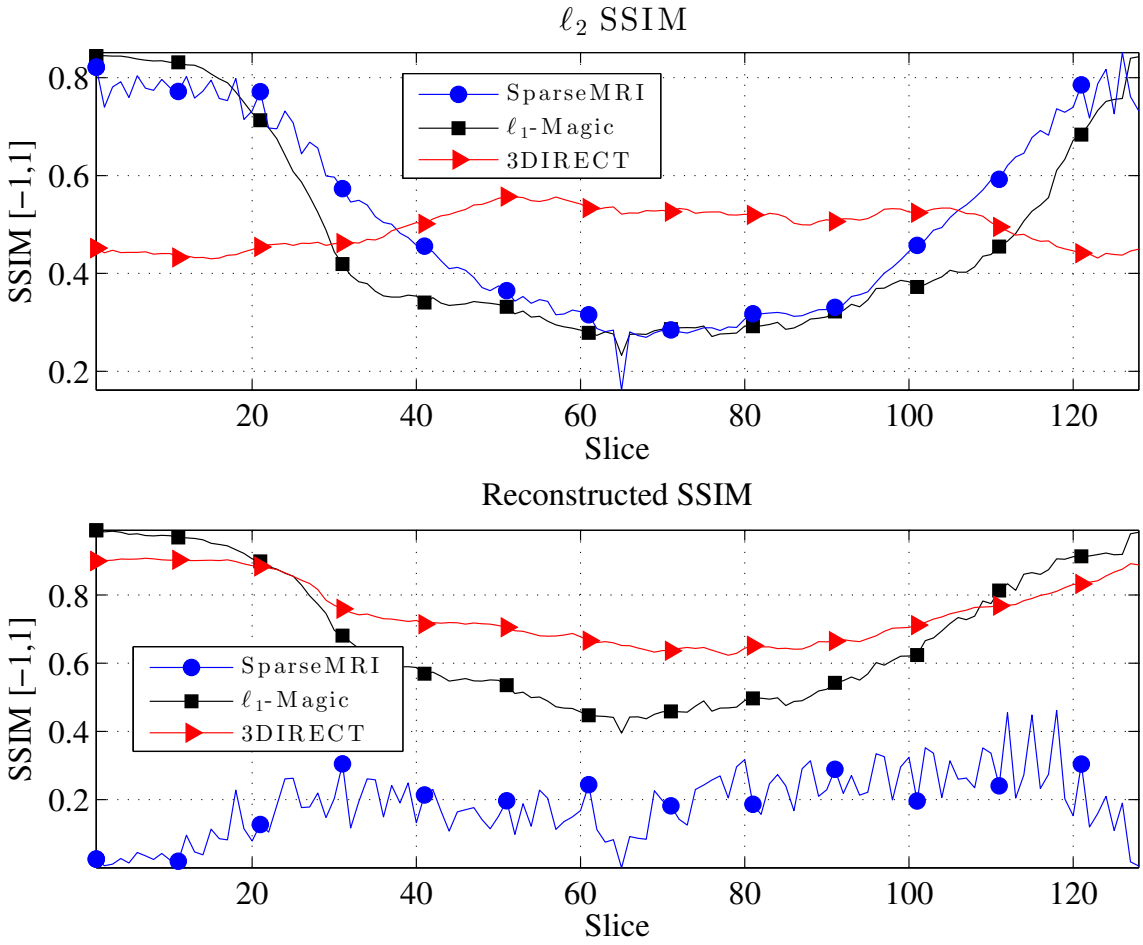

Figure 6.32. 3D Radial Recovered SSIM. Higher SSIM should mean the image is closer in structure to the original image. The results of SSIM confirm what was seen in recovered images; that 3DIRECT has superior image quality from slices 24-109.

The observations from Figures 6.28, 6.29, 6.30, and 6.31 are confirmed in Figure 6.32, as we see the 3DIRECT method outperforms both 2D methods from slice 24 to 109, which makes up 67% of the slices.

6.9 A Comparison of 2D and 3D Practical Schemes

As noted in the previous two sections, it is unfair to compare 2D and 3D methods using the same sample schemes. Therefore, we include in this section a more fair comparison of ℓ_1-Magic and 3DIRECT. To do this, we sample ℓ_1-Magic using a 2D radial pattern at 17% for each slice. We then sample 3DIRECT using a 3D star burst pattern at 17%. Lastly, we run a modified 3DIRECT method which minimizes over 3D total variation whose linear operator, $\mathcal{A}$, loops over each slice of the 3D image and performs a 2D FFT using the 2D start radial pattern at 17% for each slice. The idea behind the *modified 3DIRECT* method is that it tests how well the 3D total variation minimization does without the impacts of the tubal FFT effects seen in the regular 3DIRECT method.

The idea is that each method is sampled favorably according to its linear transformation operator, $\mathcal{A}$, and at similar rates. The results of this test show similar results that we have seen before, that is 3DIRECT has superior performance in the middle slices and ℓ_1-Magic has superior performance at the end slices. Meanwhile the modified 3DIRECT method, had poor recovery results in the middle slices. This is reflected in the comparison of the starting SSIM of the ℓ_2 image and the SSIM of the recovered image depicted in Figure 6.33. We also note that 3DIRECT as more consistent SSIM of the recovered image across all the slices.

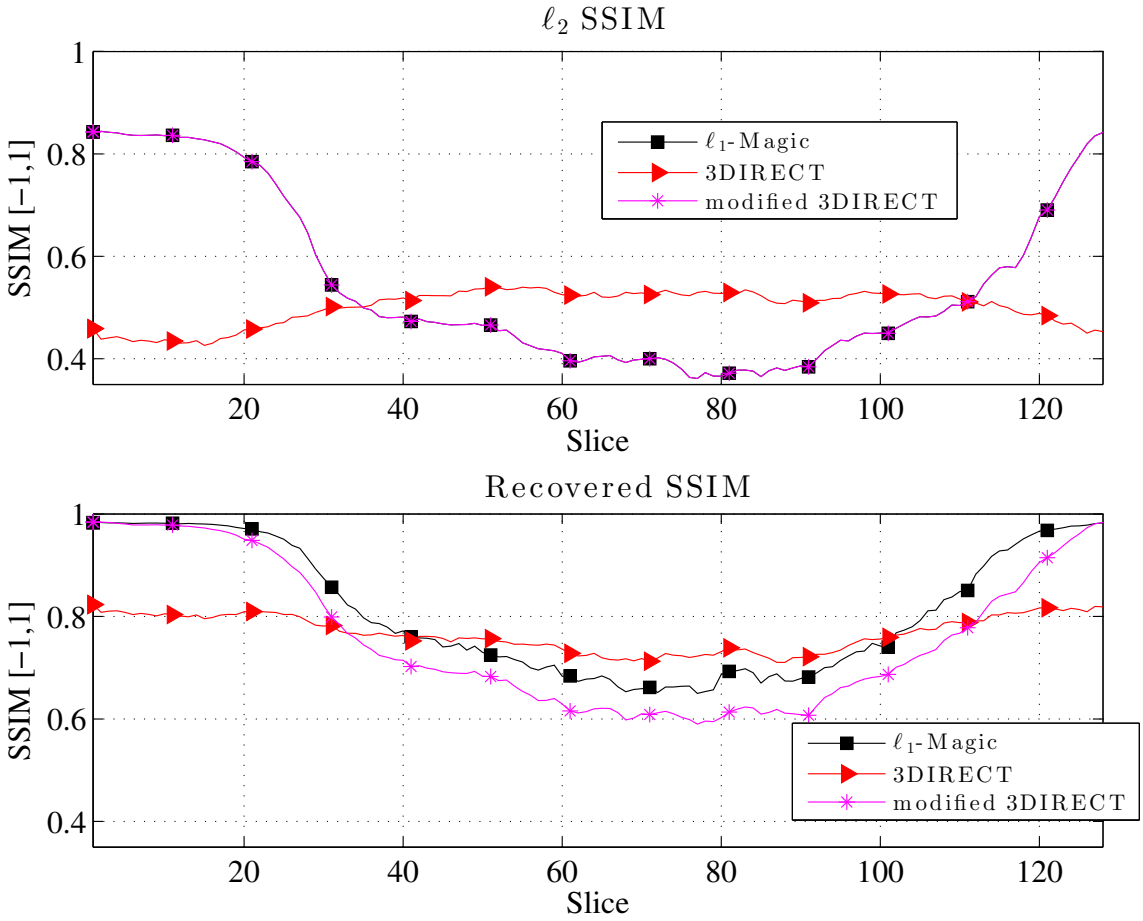

Figure 6.33. 2D Radial vs. 3D Star SSIM Comparison. 3DIRECT has superior performance in the middles slices, and suffers at the ends slices due to tubal frequency components mixing. The modified 3DIRECT had poor image quality in the middle slices. We also note that 3DIRECT as more consistent SSIM of the recovered image across all the slices.

Viewing a few of the recovered images for slices 10 and 87 confirm the results of the SSIM comparison.

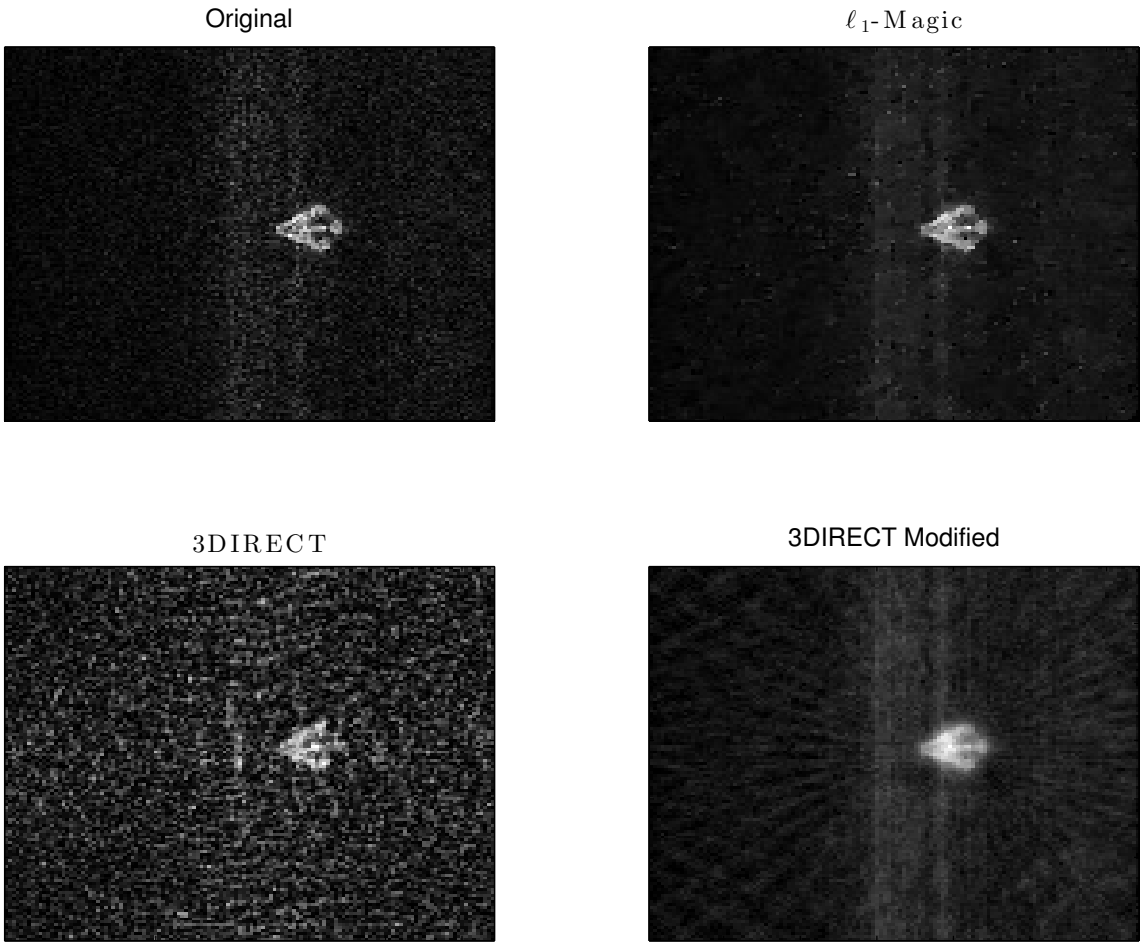

Figure 6.34. 2D Radial vs. 3D Star Recovered Image Comparison (slice 10). Tubal frequency mixing causes higher background noise for the end slices in the 3DIRECT method.

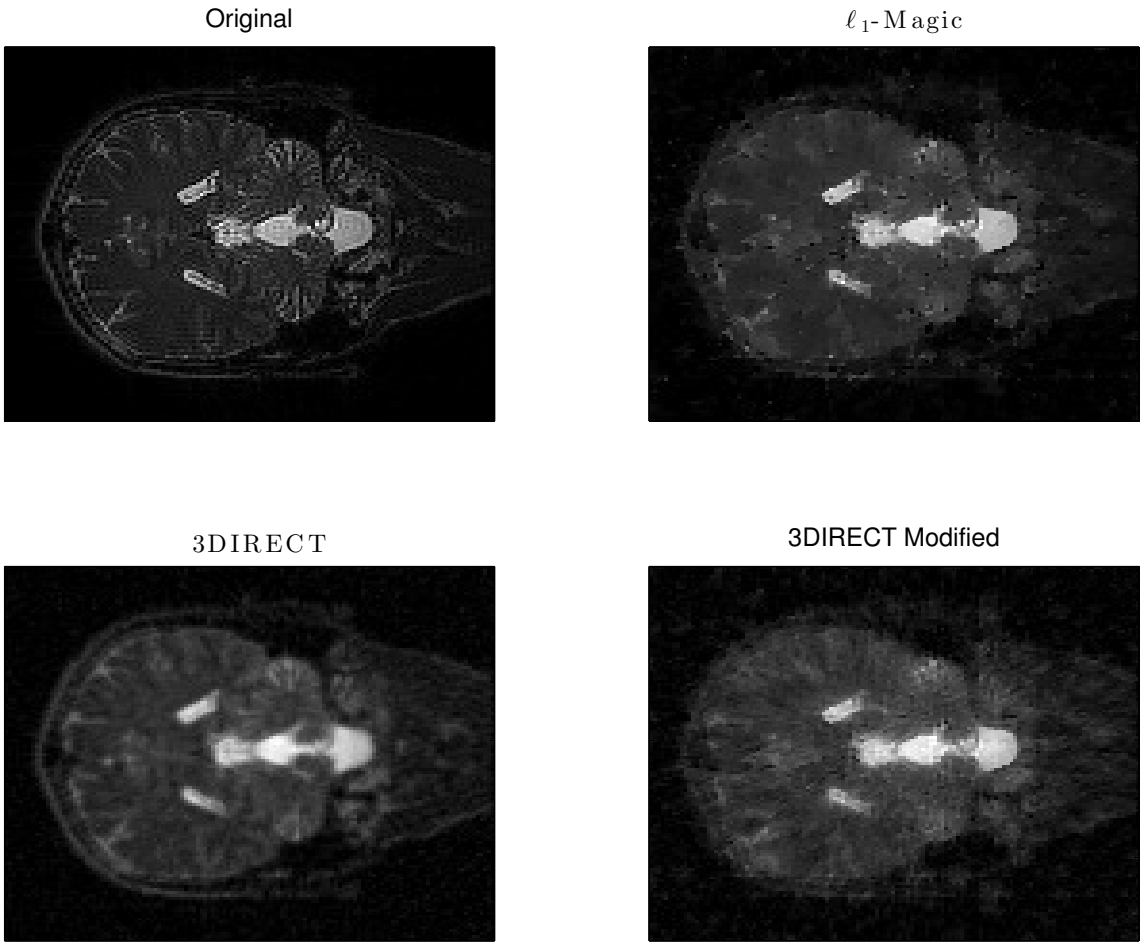

Figure 6.35. 2D Radial vs. 3D Star Recovered Image Comparison (slice 87). Tubal frequency mixing causes more details and a sharper image for the middles slices in the 3DIRECT method.

It is clear that 3DIRECT has superior performance in the middle slices for this 3D image over the ℓ-Magic and the modified 3DIRECT method. Thus it begs the question: can we detect where this performance region will occur prior to image recovery and without knowing the perfectly sampled image?

6.10 Detecting Edge Effects

It is clear the 3DIRECT method has superior results over the 2D methods when the image cross-sections occupy similar pixels in the image. When the smaller cross-sections are processed, they suffer from smearing from the large cross-sections in different slices due to the tubal dimension of the FFT. Thus the 3DIRECT method is ideal for applications where the image has more homogenous diameter in the z-direction or when the area of interest is in the middle of a scan [44]. So how can we detect where the 3DIRECT method will have superior results? That is the topic of this section.

If we go back and examine the SSIM in Figures 6.13, 6.20, 6.25, and 6.32 there is a clear correlation between how well the 3DIRECT image is recovered and its corresponding SSIM to the original image. However, in practice we won't have access to the perfectly sampled image. Thus, given a image X sparsely sampled in the k-space, one can examine its SSIM from slice to slice, and use a threshold to determine where optimal 3DIRECT performance will occur. This approach is depicted in Algorithm 6.1, which uses the SSIM software described in Section 6.2.

Algorithm 6.1 3DIRECT Performance Detections

Input: ℓ_2 Starting 3D Image X with n slices, threshold $\mu \in [0, 1]$

Output: starting 3DIRECT index i, ending 3DIRECT index j

1: $K = [0.010.03]$, myWin = `fspecial`('`gausssian`$'$, $11, 1.5)$, $L = 1$
2: **for** $i = 1 : n - 1$ **do**
3: SSIM(i) = `ssim`$(X(:, :, i), X(:, :, i + 1), k,$ myWin$, L)$
4: **end for**
5: indices = `find`(SSIM $> \mu$)
6: $i =$ indices$(1), j =$ indices(end)

Using Algorithm 6.1 on the 3D radial burst sampled image as our starting reference and $\mu = .7$ we get the results in Figure 6.36.

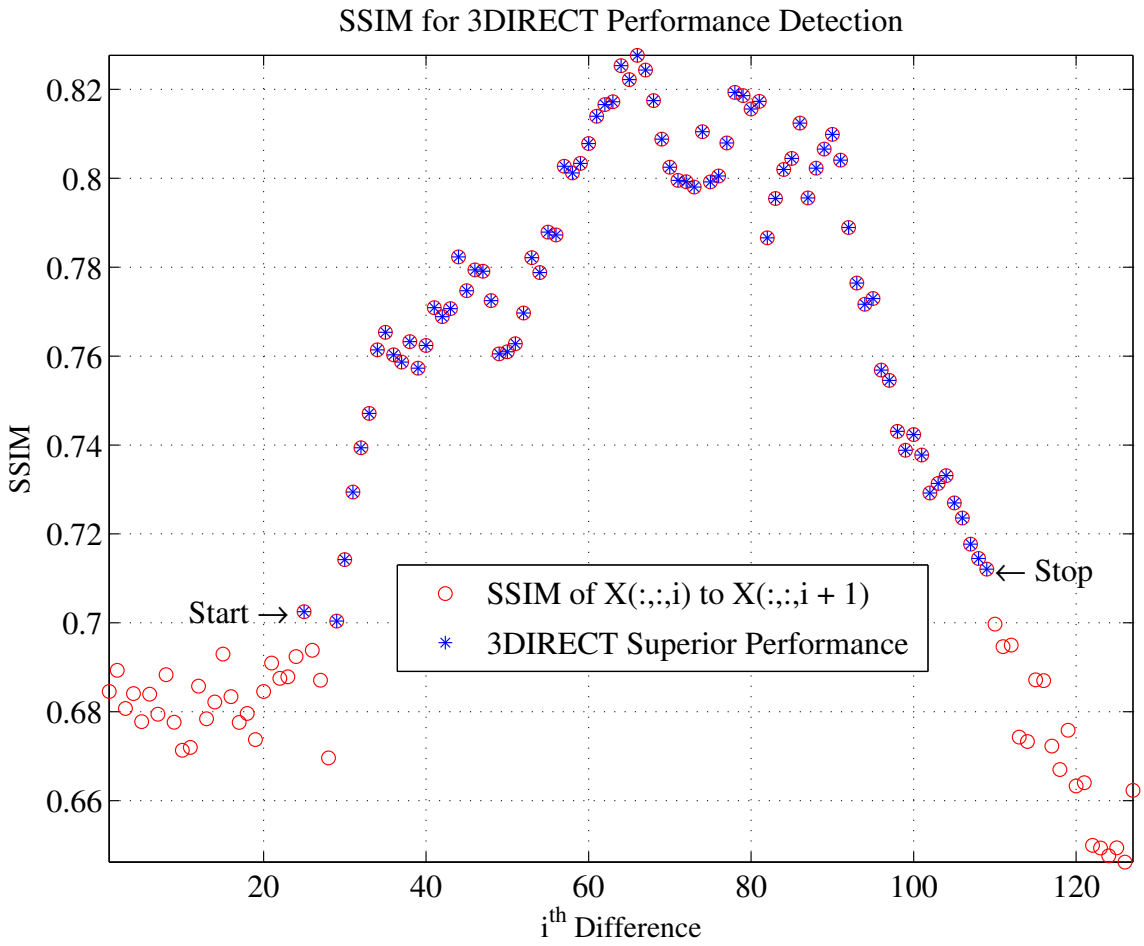

Figure 6.36. 3DIRECT Edge Detection on 3D Radial Sampled Image.

As indicated in Figure 6.36, Algorithm 6.1 determined slices 25 to 110 would have good recovery results using 3DIRECT. This is consistent with what we observed in the reconstruction images. Further, this also indicates slice 1-25 and 110-128 would recover well if processed together as two separate image recovery problems. Thus, in practice, Algorithm 6.1 can be used to decide how to parse large data sets in the

third dimension to run parallel image recoveries using the 3DIRECT method.

Overall Observations: The results indicate that the 3DIRECT method will outperform the 2D methods from slice 25 to 109 which is consistent with what we observed earlier when looking at the recovered images. Further, Algorithm 6.1 provides a way to a priori determine how to parse 3D images to run parallel image recovery that promotes optimal results by running independent 3D image recovery on segments of the larger 3D image.

CHAPTER 7

CONCLUSION

We've presented a new 3D image reconstruction method, and compared it to current state-of-the-art image recovery methods. Unlike its 2D counterparts, the 3DIRECT method can process and recover an entire 3D image at once as oppose to processing the image slice by slice. This comes with two advantages: speed and performance.

When using a 2D radial scheme which is symmetric about the 3D FFT shifts, the 3DIRECT and ℓ_1-Magic methods produce similar results. However, the 3DIRECT was 22% faster in mean and 33% faster in minimum observed run times. This is due to improvements we made to the 2D methods' inefficiencies, including: objected oriented coding of the finite difference matrices, and improved stopping tolerance for the CG method for MRI applications. In addition, our method also reduces the number of calls to the Newton and CG methods, due to its ability to treat the 3D image as a single system. These speed improvements will only continue to improve as the size of the image increases.

Secondly, the 3DIRECT algorithm can take advantage of similarities of the cross-sections of the image to be processed. This is due to the fact that it uses tubal frequency information in its minimization process. In doing so, intelligent 3D sample schemes can be exploited to achieve quality results with less sampling percentage in the k-space. This is especially beneficial for applications whose image has some homogeneity in the z-direction.

For applications where there is variation in the z-direction, and tubal frequencies can cause undesirable effects in slices with smaller image cross-sections, we presented a method to detect where the 3DIRECT algorithm performance would be best. Thus, for situations where the desirable area of interest coincides with the 3DIRECT high performance region, our method is the best choice.

REFERENCES

[1] K. Aatish and B. Zhang. Three dimensional fast Fourier transform CUDA implementation. Available online[1].

[2] T. M. Apostol. *Mathematical Analysis.* Addison-Wesley, 2nd edition, 1974.

[3] K. Block, M. Uecker, and J. Frahm. Undersampled radial MRI with multiple coils. iterative image reconstruction using a total variation constraint. *Magn. Reson. Med.*, 57(6):1086–1098, 2007.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[5] J. M. Brayer. Introduction to Fourier transforms for image processing. Available online[2].

[6] R. W. Buccigrossi and E. P. Simoncelli. Image compression via joint statistical characterization in the wavelet domain. *IEEE Trans. Image Process.*, 8:1688–1701, 1999.

[7] E. Candes. The restricted isometry property and its implications for compressed sensing. Applied & Computational Mathematics, California Institute of Technology. Available online[3], 2008.

[8] E. Candes and J. Romberg. ℓ_1-Magic: Recovery of sparse signals via convex programming. Available online[4]. 2005.

[1] cseweb.ucsd.edu/∼baden/classes/Exemplars/cse260_fa12/3DFFT.pdf
[2] www.cs.unm.edu/∼brayer/
[3] https://pdfs.semanticscholar.org/beb9/6e51a221577d5b53f8f0d102ce1427f270fd.pdf
[4] users.ece.gatech.edu/justin/l1magic/downloads/l1magic.pdf

[9] E. Candes, J. Romberg, and T. Tao. Robust uncertainty priciples: Exact signal reconstruction from high incomplete frequency information. *IEEE Trans. Inf. Theory*, 52(2):489–509, 2006.

[10] E. Candes, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, 2006.

[11] E. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inf. Theory*, 52(12), 2006.

[12] B. M. Dale, J. S. Lewin, and J. L. Duerk. Optimal design of k-space trajectories using a multi-objective genetic algorithm. *Magn. Reson. Med.*, 52(4):831–841, 2004.

[13] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57:1413–1457, 2004.

[14] D. C. Dobson and F. Santosa. Recovery of blocky images from noisy and blurred data. *SIAM J. Applied Mathematics*, 56(4):1181–1198, 1996.

[15] D. Donoho and Y. Tsaig. Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. IEEE Transactions on Information Theory, 2008.

[16] D .L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proc. Nat. Acad. Sci. USA*, 100(5):2197–2202, 2003.

[17] D. L. Donoho and B. F. Logan. Signal recovery and the large sieve. *SIAM J. Applied Mathematics*, 52(2):577–591, 1992.

[18] D. L. Donoho and P. B. Stark. Uncertainty priciples and signal recovery. *SIAM J. Applied Mathematics*, 49(3):906–931, 1989.

[19] S. C. Fain, W. Block, A. Charles, and A. B. Mistretta. Correction for artifacts in 3D angularly undersampled MRI projection reconstruction. *Proc. Intl. Soc. Mag. Reson.*, 9, 2001.

[20] J. A. Fessler and B. P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. *IEEE Trans. Signal Process.*, 51(2):560–574, 2003.

[21] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *IEEE*, 93(2):216–213, 2005.

[22] J. J. Fuchs. On sparse representations in arbitrary redundant bases. *IEEE Trans. Inf. Theory*, 50(6):1341–1344, 2004.

[23] G. H. Glover. Simple analytic spiral k-space algorithm. *Magn. Reson. Med.*, 42(2):412–414, 1999.

[24] G. H. Glover and J. M. Pauly. Projection reconstruction technique for reduction of motion effects in MRI. *Magn. Reson. Med.*, 28:275–289, 1992.

[25] G. E. Golub and C. F. Von Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.

[26] A. Greiser and M. von Kienlin. Efficient k-space sampling by density-weighted phase-encoding. *Magn. Reson. Med.*, 50(6):1266–1275, 2003.

[27] P. T. Gurney, B. A. Hargreaves, and D. G. Nishimura. Design and analysis of a practical 3d cones trajectory. *Magn. Reson. Med.*, 55(3):575–582, 2006.

[28] J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. *IEEE Trans. Inf. Theory*, 52(9):4036–4048, 2006.

[29] S. Hu, M. Lustig, and A. P. Chen. 3D compressed sensing for highly accelerated hyperpolarized 13C MRSI with in vivo applications to transgenic mouse models of cancer. *Magn. Reson. Med.*, 63(2), 2010.

[30] Q. Huynh-Thu and M. Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics Letters*, 44(13):800 – 801, 2008.

[31] S. J. Kim, K. Koh, M. Lustig, and S. Boyd. An efficient method for compressed sensing. In *IEEE International Conference on Image Processing (ICIP)*, 2007.

[32] D. A. Koff and H. Shulman. An overview of digital compression of medical images: Can we use lossy compression in radiology? *Can Assoc Radiol J.*, 57(4):211–217, 2006.

[33] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lbret. Application of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[34] M. Lustig, D. Donoho, and J. M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magn. Reson. Med.*, 58:1182–1196, 2007.

[35] M. Lustig and J. M. Pauly. SPIRiT: Iterative self-consistent parallel imaging reconstruction from arbitrary k-space. *Magn. Reson. Med.*, 64:457–471, 2010.

[36] Michael Lustig. SparseMRI. Available online[5]. 2008.

[37] J. Ma. Improved iterative curvelet thresholding for compressed sensing and measurement. *IEEE Trans. Instrum. Meas.*, 60(1):126–136, 2011.

[38] B. Madore, G. Glover, and N. Pelc. Unaliasing by Fourier-encoding the overlaps using the temporal dimension (UNFOLD), applied to cardiac imaging and fMRI. *Magn. Reson. Med.*, 42:813–828, 1999.

[39] MATLAB. Object-oriented programming in MATLAB. Available online[6].

[40] G. McGibney, M. R. Smith, S. T. Nichols, and A. Crawley. Quantitative evaluation of several partial Fourier reconstruction algorithms used in mriuation of several partial Fourier reconstruction algorithms used in MRI. *Magn. Reson. Med.*, 30(1):51–59, 1993.

[5] people.eecs.berkeley.edu/ mlustig/Software.html

[6] www.mathworks.com/discovery/object-oriented-programming.html

[41] R. Mir, A. Guesalaga, J. Spiniak, M. Guarini, and P. Irarrazaval. Fast three-dimensional k-space trajectory design using missile guidance ideas. *Magn. Reson. Med.*, 52(2):329–336, 2004.

[42] C. A. Mistretta, O. Wieben, J. Velikina, W. Block, J. Perry, Y. Wu, K. Johnson, and Y. Wu. Highly constrained backprojection for time-resolved MRI. *Magn. Reson. Med.*, 55(1):30–40, 2006.

[43] K. S. Nayak and D. G. Nishimura. Randomized trajectories for reduced aliasing artifacts. Available online[7]. 6th Annual Meeting of ISMRM, 1998.

[44] Q. Ning, C. Ma, F. Lam, B. Clifford, and Z. Liang. Spectral quantification of MRSI data using spatiospectral constraints. Available online[8]. University of Illinois, 2015.

[45] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2000.

[46] D. C. Peters, F. R. Korosec, T. M. Grist, W. F. Block, J. E. Holden, K. K. Vigen, and C. A. Mistretta. Undersampled projection reconstruction applied to MR angiography. *Magn. Reson. Med.*, 43:91–101, 2000.

[47] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[48] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Available online[9]. Carnegie Mellon University, 1994.

[49] J. L. Starck, M. Elad, and D. L. Donoho. Image decomposition via the combination of sparse representations and a variational approach. Stanford University, 2004.

[7] http://sipi.usc.edu/∼knayak/pdf/kspace-ismrm98-670.pdf
[8] http://web.engr.illinois.edu/∼qning2/papers/ismrm16_quant.pdf
[9] http://www.cs.cmu.edu/∼./quake-papers/painless-conjugate-gradient.pdf

[50] T. Strohmer and J. Tanner. Implementations of shannon's sampling theorem, a time-frequency approach. *Sampling Theory in Signal and Image Processing*, 4(1):1–17, 2005.

[51] C. M. Tsai and D. G. Nishimura. Reduced aliasing artifacts using variable-density k-space sampling trajectories. *Magn. Reson. Med.*, 43(3):452–458, 2000.

[52] M. Uecker. Nonlinear reconstruction methods for parallel magnetic resonance imaging. PhD Thesis University of Göttingen, 2009.

[53] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig. ESPIRiT–an eigenvalue approach to autocalibrating parallel MRI: where SENSE meets GRAPPA. *Magn. Reson. Med.*, 71(3):990–1001, 2014.

[54] S. S. Vasanawala, M. J. Murphy, M. T. Alley, P. Lai, K. Keutzer, J. M. Pauly, and M. Lustig. Practical parallel imaging compressed sensing MRI: Summary of two years of experience in accelerating body MRI of pediatric patients. pages 1039–1043. IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2011.

[55] F. Wajer. Non-cartesian MRI scan time reduction through sparse sampling. PhD thesis, Delft University of Technology, 2001.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structual similarity. *IEEE Trans. Image Process.*, 13(4):1–13, 2004.

[57] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. volume 2. Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, 2004.

[58] P. Whittle. *Prediction and Regulation by Linear Least-Square Methods*. University of Minnesota Press, 1983.

[59] J. C. Ye, S. Tak, Y. Han, and H. W. Park. Projection reconstruction MR imaging using FOCUSS. *Magn. Reson. Med.*, 57:764–755, 2007.

[60] P. Yin, Y. Lou, Q. He, and J. Xin. Miniminzation of ℓ_{1-2} for compressed sensing. *SIAM J. Sci. Comput.*, 37(1):A536–A563, 2015.