Imaging Applications of the Sparse FFT

by

Lixin Shi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

© Massachusetts Institute of Technology 2013. All rights reserved.

Author ........................................

Department of Electrical Engineering and Computer Science
May 22, 2013

Certified by .................................

Dina Katabi
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ..............................

Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students
Imaging Applications of the Sparse FFT

by

Lixin Shi

Submitted to the Department of Electrical Engineering and Computer Science on May 22, 2013, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

The sparse Fourier transform leverages the intrinsic sparsity of the frequency spectrum in many natural applications to compute the discrete Fourier Transform (DFT) in sub-linear time. Consequently, it has the potential to enable Big Data applications. In this thesis, we focus on extending the sparse Fourier transform (sparse FFT) to two imaging applications: 4D Light Field and Magnetic Resonance Spectroscopy. Directly applying sparse FFT to these applications however will not work. We need to extend the sparse FFT algorithm to address the following challenges: First, both applications are sample-intensive. It is time consuming, costly, and difficult to acquire samples. So, we need a new sparse FFT algorithm that minimizes the number of required input samples instead of purely focusing on the running time. Second, for these applications the spectra are not very sparse in the discrete Fourier domain. The sparsity is much greater in the continuous Fourier domain. Hence, we need a new sparse FFT algorithm that can leverage the sparsity in the continuous domain as opposed to the discrete domain.

In this thesis, we design a sparse FFT algorithm suitable for our imaging applications. Our algorithm contains two phases: it first reconstructs a coarse discrete spectrum and then refines it using gradient descent in the continuous Fourier domain. In our experiments, we showed high-quality reconstruction of 4D light field with only 10% 20% of the samples, and a reduction of the MRS acquisition time by a factor of 3× 4×.

Thesis Supervisor: Dina Katabi
Title: Professor of Electrical Engineering and Computer Science
Disclaimer

Both of the projects, light field and MRS, are joint projects with other students. I really enjoy the collaborations and appreciate their help.

The light field project is joint work with Haitham Al-Hassanieh and Myers Abraham Davis. The principles of the algorithm were developed during discussions and brainstorming sessions between Haitham and me. Beyond that, I implemented the light field reconstruction algorithm, reconstructed the light field data sets and did the empirical evaluation. Abe worked on the light field capturing part and I learned a lot of graphics background from discussing with him.

The MRS project is joint work with Haitham Al-Hassanieh and Ovidiu C. Andronesi. Both Haitham and me shared credit for the design of the MRS reconstruction algorithm. I implemented the MRS reconstruction algorithm and did the visual and quantitative empirical studies for the effectiveness of our reconstruction. Ovidiu educated us with essential MRS background and helped acquire data sets for our experiments.
Acknowledgments

First and foremost, I would like to thank my adviser Dina. Without her guidance I cannot finish this thesis. In the past two years, she provided me with invaluable guidance and generous help; whenever I feel lost or have problems, I would turn to her and she always gives me prompt and helpful feedbacks. Her guidance and help to me go beyond research and I enjoy working with her and learning from her. I hope to learn more from her in the coming years.

I am in debt to Haitham, with whom I work on the sparse FFT related projects. He gave me considerable support and inspiration throughout the projects. I would like to express my gratefulness to Haitham, Dina, Piotr and Eric who came up with the wonderful idea of sparse FFT and opened so many new research topics. I owe many thanks to my collaborators in these projects: Abe, Ovidiu, Badih, Fredo and Elfar. Only with their kind assistance and hard work that I can finish this thesis. I am also thankful to the people who provided a lot of help to me while I was doing the project: Andrew Adams, Trina Kok, Diego Cifuentes and Ross Knepper.

I would like to thank all of my group-mates: Rahul, Shyam, Nabeel, Nate, Jue, Haitham, Swarun, Fadel, Omid, Ezz and Diego for their great company during my first two years at MIT.

I really would like to thank my parents, Yiting and Yane, and my sister Hong, for their extraordinary supports over all of my 22 years, for their understanding of me leaving home for 6 years (starting from my undergraduates) and not being not able to accompany them when they need me.

I cannot express enough gratitude to Miya, to whom I can lean when I am helpless, and who always makes my life beautiful.
# Contents

1 Introduction .................................................. 19
   1.1 Challenges .............................................. 20
   1.2 Reconstruction Algorithm ................................. 21
   1.3 Results ................................................ 22

2 Imaging Applications Background ............................. 25
   2.1 4D Light Field ........................................... 25
   2.2 Magnetic Resonance Spectroscopy (MRS) .................... 26

3 Related Work .................................................. 29
   3.1 Sparse FFT .............................................. 29
   3.2 Light Field .............................................. 30
   3.3 Magnetic Resonance Spectroscopy .......................... 31

4 Sparsity in the Discrete vs. Continuous Fourier Domain ....... 35

5 The Reconstruction Algorithm ................................ 39
   5.1 Notation ................................................ 39
   5.2 Overview ................................................. 39
   5.3 On-Grid Recovery ......................................... 42
      5.3.1 Compute Projections: ................................. 43
      5.3.2 Recover Frequency Positions: ......................... 44
      5.3.3 Recover Frequency Coefficients: ....................... 44
   5.4 Off-Grid Recovery ......................................... 47
5.4.1 Gradient Search ........................................... 50
5.5 Algorithm Discussion ....................................... 50
  5.5.1 Speeding Up the Algorithm .......................... 50
  5.5.2 Add, Delete or Merge Peaks ......................... 52
  5.5.3 Leveraging Light Field Priors ....................... 53

6 Light Field Reconstruction ................................. 55
  6.1 Implementation ........................................... 55
    6.1.1 Data Sets ........................................ 55
    6.1.2 Sampling .......................................... 55
    6.1.3 Algorithm Implementation ......................... 56
  6.2 Results .................................................. 57
    6.2.1 The Stanford Bunny ............................... 57
    6.2.2 Amethyst .......................................... 57
    6.2.3 Crystal Ball ...................................... 59
    6.2.4 Gnome .............................................. 59
    6.2.5 Informal comparison with Levin and Durand [2010] . 60
    6.2.6 Extending views ................................ 62
    6.2.7 Refocusing ........................................ 62
  6.3 Discussion ............................................... 63
    6.3.1 Viewpoint Denoising .............................. 63
    6.3.2 Importance of continuous off-grid recovery ... 64

7 Magnetic Resonance Spectroscopy Reconstruction ........ 67
  7.1 Reconstruction Algorithm ............................... 67
  7.2 Experiment Setup ...................................... 68
  7.3 Results ................................................ 69
    7.3.1 Reconstructed Spectra ............................ 69
    7.3.2 Quantitative Analysis ............................ 70

8 Conclusion .................................................. 75
List of Figures

1-1 A demonstration of windowing effect in real data sets: (a) in an example slice of light field spectrum, the cross tails over the two axes are the windowing artifacts. (b) In MRS, the windowing artifacts are more profound in one dimension ($f_1$) since we suffer from insufficient samples in this dimension and we have enough samples in the other. The vertical bars in the image are the windowing effects. . . . . . . . . 21

2-1 Light Field Demonstration: (a) The 4D light field describes the light rays between plane $(u, v)$ and $(x, y)$. Each value $L(x_0, y_0, u_0, v_0)$ describes the ray from position $(x_0, y_0)$ to position $(u_0, v_0)$. (b) An example of a captured light field data set, where each small image is captured by a specific angular view point. They are ordered in the same way as they are captured in the $(u, v)$ domain. To display the images we subsample $u$ and $v$ by 3 (from 17x17 to 6x6) . . . . . . . . 26

2-2 MRS Demonstrations: This is an example of a localized 2D COSY experiment process, the same technique we focus on in this thesis. (a) It first localizes a voxel in the human brain (or other organs) whose spectrum we want to examine. (b) The scanning process is done by repeatedly sending two pulses and iteratively changing the interval between them (the $t_1$ interval). (c) The reconstruction is done by taking FFT of the whole $t_1, t_2$ signal. The spectrum shown here is recovered by our algorithm so that there are no windowing artifacts. . 27
4-1 The windowing effect: limiting the samples in the time domain to an aperture $A$ is equivalent to convolving with a sinc function. (a) If frequency spike lies on a DFT grid point, once the sinc is discretized, it disappears and the original sparsity of the spectrum is preserved. (b) If the frequency spike is off the grid, once we discretize we get a sinc tail and the spectrum is no longer as sparse as in the continuous domain. 37

4-2 A 2D angular slice of the 4D light field spectrum It is taken from the Stanford crystal ball light field at $(\omega_x, \omega_y) = (50, 50)$. (a) In the discrete Fourier domain, we have sinc tails and the spectrum is not very sparse. (b) In the continuous Fourier domain, as reconstructed by our algorithm, the spectrum is much sparser. It is formed of 4 peaks which do not fall on the grid points of the DFT. .................. 38

5-1 Flow chart of the 2D sparse FFT reconstruction algorithm. The algorithm takes a set of sampled line segments. The first stage, On-Grid Recovery, has 3 steps: computing the projections, recovering the integer positions, and recovering the coefficients of the large frequencies. This stage outputs a list of on-grid positions and coefficients of the large frequencies. The second stage is Off-Grid Recovery. In this step, the gradient search tries to shift the positions of the frequencies to non-integer locations and recover their coefficients. We keep repeating this gradient search until we get a small enough error. This stage will output a list of off-grid frequency positions and their coefficients which can then be used to reconstruct the full 2D slice. .................. 43

5-2 Sampling Patterns: Our algorithms samples the $(u, v)$ angular domain along line segments. (a) Box and 2 diagonals (b) Middle row and middle column and 2 diagonals (c) Box and 2 lines of slopes $= \pm 2$. Note that in this case the line wraps around. .................. 43
5-3 Computing the DFT of a line segment of a 2D signal is equivalent to projecting the 2D spectrum on the line segment. (a) Row projection (b) Column projection (c) Diagonal projection

5-4 Two examples of the voting procedure used to recover the discrete positions of the large frequencies from projections on the line segments. The 2D spectrum is projected on a row, a column and a diagonal. Each large projection votes for the frequencies that map to it. (a) Frequencies (5,5), (5,9), and (9,5) are large and they all get 3 votes. (b) Frequencies on the diagonal are large and only these frequencies get 3 votes.

5-5 A demonstration of gradient search process: The gradient descent algorithm will shift the frequency by a small step in every iteration to minimize the error. (a) One frequency (4,5) is shifted to an off the grid position that best minimizes the error. It converges after 3rd iteration. (b) One fake frequency (blue) which was the result of an error in the first stage (On-Grid Recovery) and one actual large frequency (red). The algorithm keeps shiting the fake frequency until it crosses the integer boundary and merges with an actual large frequency thus correcting the error of the first stage.

6-1 Sampling Pattern of the Crystal Ball data set and the Gnome data set: All of the images from different view points of light field are arranged by their \((u, v)\) positions. The images that is used as input samples to our algorithm are highlighted while others are darkened. Crystal Ball and other Stanford light field archive data sets use the box-and-X sampling pattern, while Gnome uses box-and-double-X pattern.
6-2 **The reconstruction of Stanford Bunny data set from one** $(u, v)$ **viewpoint.** On the left side are the reference figure and the reconstruction from [25]. The sampling pattern we used is the box-and-X pattern (fig. 5-2(a)). Though we used more samples than Levin & Durand used, we did a much better job in terms of less blurring, preserving the textures and having less noise.

6-3 **The color map of the Y channel for the difference between the reference and the reconstructed view from Stanford Bunny dataset.** In about half of the pixels the differences are just zeros; there are some unstructured noises and it is hard to tell whether they come from the reference figure or our reconstruction; and there are structured noises on the edge of the bunny as well.

6-4 **The reconstruction of Amethyst data set(right), the reference figure(left) and the reconstruction from [25] (middle).** We are using the box-and-X sampling pattern (fig. 5-2(a)), which is more than the number of samples Levin & Durand used. However, we are able to reconstruct this highly non-Lambertian view and it is hard to distinguish our reconstruction from the full captured light field.

6-5 **One example of reconstructed view from Amethyst data set where we lose some reflection details.** The specular reflection only occurs in views outside our input set.

6-6 **One** $(u, v)$ **view from the reconstruction of the Crystal Ball data set.** We are using box-and-X sampling pattern (as shown in the blue box in the bottom center) and the red dot shows the position of reconstructed view in angular domain. Despite the fact that the scene is extremely non-Lambertian and has complex structures, we could still reconstruct the images with most of its details.
6-7 **One** 
\((u, v)\) **view from the reconstruction of Gnome data set.**
We use sample patterns from fig. 5-2(e), as shown by the blue box in the bottom right. The red marker shows where the view is in angular domain. Although the captured data set is noisy, we could still reconstruct the data set with good details.

6-8 **Extending views:** We extend our reconstruction of Stanford Bunny dataset (fig 6-2) and extend the camera views. The original view is \(0 \leq u \leq 16\) and \(0 \leq v \leq 16\), and here we show our extension to \((-2, -2)\) and \((18, 18)\).

6-9 **Refocus to different depth planes:** We use our reconstruction of the Amethyst data set and refocus it to the front and rear depth planes.

6-10 **Viewpoint denoising demonstration:** Top: We see noise in the \(u,v\) dimensions of our reference data caused by registration errors. This error shows up as camera shake in the reference images. Bottom: Our algorithm effectively removes this noise in the reconstruction, essentially performing camera stabilization.

6-11 **Comparison of our final reconstruction in the continuous domain to two alternative reconstructions in the discrete domain:** We compare our result (right) with the output of only the first step of our algorithm (left), as well as the discrete approximation of our result with sinc tails removed (middle).

6-12 **A demonstration of the ghost effect caused by removing sinc tails in frequency domain.** We removed sinc tail from the spectrum of Stanford Bunny dataset, got the reconstruction and selected an inset from it (the same inset as in fig. 6-11). The figure shows how this inset changes across \((u,v)\) aperture (note that we subsampled the 17x17 aperture by 2) and the ghost effect gets stronger as we go from the center of the aperture to the edge.
7-1 Phantom/In vivo Data Result, picking one $f_2$ column with $f_2 = 4.4\text{ppm}$. It shows that our reconstruction zeros out the tails.

7-2 Reconstruction of a phantom data set and comparisons with FFT, Full FFT and CS. The brain phantom contains NAA, Lac, 2HG, Glu, Myo, Cho, GABA. Our reconstruction largely reduces the tails while keeping most of the cross-diagonal peaks. The only peak that exists in Full FFT but is missed by our reconstruction is 2HG. If we look at other peaks, for example, Myo, we could find that it gets successfully recovered by our algorithm, but it is totally missing in FFT and CS could only recover a very much weakened version of Myo.

7-3 Reconstruction of an in vivo data set and comparisons with FFT, Full FFT and CS. The metabolites that can be detected are NAA, Cho, Asp and Glu. Our reconstruction could eliminate the heavy windowing artifacts and disentangle the cross-diagonal peaks from tails. For Asp and Cho, they are mixed with windowing artifacts in Full FFT and it is hard to distinguish between them and the tails; in our reconstruction they are well separated.
# List of Tables

5.1 **Notation:** These are the notations used in chapter 5 .................................. 40

7.1 **Notation:** These are the notations used in chapter 7 ................................. 67

7.2 **The Acquisition Time:** We tried three different acquisition patterns with $S_1/N_1$ equals 64/180, 60/160 and 72/256. Full ACQ (Acquisition) time and reduced ACQ (Acquisition) time are both measured in minutes and exclude the time to localize a voxel in the brain. We showed a factor of 3 - 4 reduction in acquisition time from full acquisitions. .......................................................... 71

7.3 **Line Width of NAA** (ppm): Our reconstruction could reduce the line width of Full FFT by 26% .......................................................... 73

7.4 **Signal-to-Artifacts Ratio** (dB): Our reconstruction could improve the Signal-to-artifacts ratio by 13.75dB for in vivo experiment .......................... 73
Chapter 1

Introduction

The Fast Fourier Transform (FFT) algorithm is a powerful tool whose applications include video and audio processing [44, 18, 4], radar and GPS systems [19, 8], medical imaging, spectroscopy [22, 34], the processing of seismic data by the oil and gas industries [47], and many other engineering tasks. However, the emergence of big data problems, in which the processed datasets can exceed terabytes [39], has rendered the FFT’s runtime $^1$ too slow. The Sparse FFT algorithm [20, 21] addresses this problem by providing a sub-linear algorithm leveraging the sparsity in the Fourier domain. For input size $n$ with sparsity $k$, sparse FFT reconstructs the frequency domain in $O(k \log n \log (n/k))$ time.

The fast computation speed makes the sparse FFT promising for many data-intensive applications. In this thesis, we mainly focus on adapting the sparse FFT to two imaging applications, 4D light fields [28, 15] and 2D Magnetic Resonance Spectroscopy (MRS) [29]. In 4D light fields, a 2D arrays of cameras is used to capture the 4 dimensional properties of light rays. 4D light fields open up a variety of interesting applications including interactive viewpoint changes [28, 15], depth extraction [1], and refocusing [33]. In 2D MRS, an MRI machine is used to scan a specific area of the brain and output the molecular composition of that area. It allows doctors to detect new biomarkers that indicate certain diseases, e.g., 2-hydroxyglutarate (2HG) [2, 11]

$^1$The time complexity of FFT is $O(n \log n)$ for input of size $n$
as an indicator of gliomas, and hence, which can greatly benefit clinical diagnosis.

Both of the two applications however are sample-intensive. Their data acquisition process is either time-consuming, cumbersome or costly. In MRS, the acquisition is an iterative scanning process that takes 25–40 min. Since patients cannot stay still in the MRI machine for that long, 2D MRS is restricted to research and is not used today for clinical diagnosis. In light fields, despite various attempts [13, 6, 27, 30], the dominating way to capture a good light field is still by using costly lenslet arrays, bulky camera arrays or time-consuming robot gantries. In these two applications and many other applications, the main goal is to reduce the number of samples to make the acquisition more practical while keeping a good reconstruction quality.

1.1 Challenges

Sparse FFT seems like a natural fit to these applications, since their frequency spectra are sparse, and sparse FFT only needs a subset of samples to reconstruct a sparse frequency spectrum. Nevertheless, directly applying sparse FFT is not desirable because of two major challenges:

First, original sparse FFT is not the best solution to reduce the required number of input samples since it takes more samples than optimally needed. The original sparse FFT is optimal in terms of running time, but is not optimal in terms of sample complexity. The sample complexity of sparse FFT is a logarithmic factor away from the optimal one \(^2\) and the large constant in the asymptotic big-O notation of sample complexity makes it not ideal for sample-intensive applications.

Second, many real applications suffer from the windowing effect which significantly reduces the sparsity of the spectrum. Because of the limitation in acquisition time, we could only get a finite window of samples. This is equivalent to multiplying the signal by a rectangular window whose length is the distance between the first and the last samples. Since multiplication translates into convolution in the frequency domain,

\(^2\)The sparse FFT has a sample complexity of \(\Theta(k \log n \log(n/k))\), whereas the optimal sample complexity is \(\Omega(k \log(n/k))\) [36].
Figure 1-1: A demonstration of windowing effect in real data sets: (a) in an example slice of light field spectrum, the cross tails over the two axes are the windowing artifacts. (b) In MRS, the windowing artifacts are more profound in one dimension ($f_1$) since we suffer from insufficient samples in this dimension and we have enough samples in the other. The vertical bars in the image are the windowing effects.

This windowing effect essentially convolves the spectrum with the sinc function, which is the Fourier transform of a rectangle function. The slowly decaying tail of the sinc function imposes a tail for each frequency peak. In imaging applications where acquisition is a major bottleneck, the spectra are more sensitive to these windowing artifacts since the acquisition window is very small. Figure 1-1 shows an example of how the sinc tail reduces the sparsity in both light field and MRS. Our key observation here is that the sparsity of many natural applications is much greater in the continuous Fourier spectrum than in the discrete Fourier spectrum, because the aforementioned sinc tail blurs the Fourier domain with non-ignorable artifacts.

### 1.2 Reconstruction Algorithm

The main contribution of this thesis is a new sparse FFT algorithm that takes care of both the sample complexity and the windowing effect. We introduce a two-phase algorithm that first does a coarse sparse discrete estimate of the spectrum, and then refines the reconstruction in the continuous domain.
The first step leverages the fact that subsampling in time domain is aliasing in frequency domain. It takes different sets of subsamples with different subsampling ratios, which correspond to different ways of folding entries in the frequency domain. It then uses a voting technique similar to [21] to discover the big frequency entries and their values. The use of these subsampling-aliasing filters instead of the Gaussian filters in the original sparse FFT algorithm [21, 20] allows the algorithm to save a logarithmic factor in the number of samples used.

The second step uses a gradient descent approach to approximate the continuous frequency spectrum. It iteratively shifts the positions of the big frequency entries to find an estimate of their positions in the continuous Fourier domain. In each iteration, we shift the entry along the direction that best minimizes the square error of our reconstruction of the known input samples. The output, then, is a set of frequency entries whose positions are not necessarily on the discrete Fourier transform (DFT) grid.

1.3 Results

We apply the algorithm to multiple data sets of two imaging applications, 4D light field [28, 15], and Magnetic Resonance Spectroscopy [29] which suffer heavily from these two challenges (sec. 2). Our results show the following:

- We show a reduction in the number of required samples which makes the acquisition more practical.
  - We can reconstruct a light field spectrum using only 10%-20% of the input images. Our sampling uses a camera moving along a 1D trajectory instead of scanning the full 2D grid positions which makes capturing the lightfield significantly easier.
  - In MRS, we reduce the time a patient has to spend in an MRI machine in order to get a full magnetic resonance spectrum from 30-40 minutes to 10 minutes.

22
• We demonstrate an improvement in the reconstructed spectra even comparing with full FFT, because of the fact that we are reconstructing in the continuous Fourier Domain.

  - In light field, we are effectively denoising the noise introduced by imprecise camera locations ("jitter noise") and light flickering.
  - In MRS, we improve the signal to artifacts ratio by a maximum of 13.75 dB and can recover peaks which are otherwise immersed in the sinc tails.

• We also demonstrate that sparse FFT enables some applications that cannot be done otherwise. One example is extending the light field to extra view points, we show a 53% increment of view aperture while still roughly keeping the same reconstruction quality.

The rest of the thesis is organized as follows: In chapter 2 we will give a short primer on the imaging applications we are going to focus on: 4D light field and Magnetic Resonance Spectroscopy (MRS). In chapter 3 we discuss related work with this thesis. Following this, in chapter 4 we elaborate on our idea of exploiting sparsity in the continuous Fourier domain, and in chapter 5 we leverage our observations to design the main reconstruction algorithm. For simplicity in describing the algorithm we use the context of light field but it also applies to MRS. In chapter 6 and chapter 7 we present the experimental result for light field and MRS respectively. Finally, chapter 8 concludes the thesis.
Chapter 2

Imaging Applications Background

2.1 4D Light Field

A 4D light field \( L(x, y, u, v) \) characterizes the light rays between two parallel planes (figure. 2-1). One plane is the \((u, v)\) plane where a camera (lenslet) can move along or an array of cameras (lenslets) can reside on. The other plane is \((x, y)\) which is essentially the focal plane of the cameras (lenslets). \( uv \) spans camera plane and \( xy \) spans image plane, which we will refer to hereafter as angular and spatial dimensions respectively. Each \((u, v)\) coordinate corresponds to the location of the viewpoint of the camera and each \((x, y)\) coordinate corresponds to a pixel location. This gives a 4D description of the light rays and enables many promising computational photography applications, including interactive viewpoint changes [28, 15], depth extraction [1], and refocusing [33].

Many of the interesting properties of the light field come from its frequency domain \( \hat{L}(\omega_x, \omega_y, \omega_u, \omega_v) \). One important conclusion is the Fourier Slice Photography theorem [32], saying that the non-zero part of the light field spectrum is only a 3D subspace of the whole 4D light field. This subsumes that the scene is Lambertian, i.e., the reflection from any point has isotropic luminance. The fact that energies are largely constrained in a 3D subspace is referred to as the dimensionality gap henceafter. This essentially proved that Lambertian views have very sparse frequency domain.

However, the major problem with light field representation is the large data volume
Figure 2-1: **Light Field Demonstration**: (a) The 4D light field describes the light rays between plane \((u, v)\) and \((x, y)\). Each value \(L(x_0, y_0, u_0, v_0)\) describes the ray from position \((x_0, y_0)\) to position \((u_0, v_0)\). (b) An example of a captured light field data set, where each small image is captured by a specific angular view point. They are ordered in the same way as they are captured in the \((u, v)\) domain. To display the images we subsample \(u\) and \(v\) by 3 (from 17x17 to 6x6)

it has to present and the resulting difficulty in capturing a light field. The state-of-the-art cameras can easily get good resolution and bandwidth in the spatial domain, but it is hard to get enough samples (i.e., camera/lenslet positions) in the angular domain. Consequently the resolution and bandwidth are intrinsically limited in the angular dimensions.

### 2.2 Magnetic Resonance Spectroscopy (MRS)

Comparing with the traditional Magnetic Resonance Imaging (MRI) [17] techniques, which gives a 3D medical imaging by a collection of 2D slices, Magnetic Resonance Spectroscopy is able to provide a spectrum representing the levels of different metabolites in a human organ (e.g., brain). In vivo MRS techniques place human body into static magnetic fields (commonly 1.5T-3T) and then emit oscillating magnetic pulses to excite different metabolites. Since different metabolites response in different frequencies of the pulse, they will correspond to different peaks of the resulting spectrum.
Figure 2-2: **MRS Demonstrations:** This is an example of a localized 2D COSY experiment process, the same technique we focus on in this thesis. (a) It first localizes a voxel in the human brain (or other organs) whose spectrum we want to examine. (b) The scanning process is done by repeatedly sending two pulses and iteratively changing the interval between them (the $t_1$ interval). (c) The reconstruction is done by taking FFT of the whole $t_1, t_2$ signal. The spectrum shown here is recovered by our algorithm so that there are no windowing artifacts.

The specific MRS technique that we are going to look into is the 2D Correlation Spectroscopy (COSY) technique [42]. Figure 2-2 shows the basic procedure of a COSY experiment. The acquisition is divided into different scans, in each scan we send two pulses $p_1$ and $p_2$, where the time interval between them will change across scans. The consequence is that the two time dimensions are acquired differently: $t_2$ (henceforth, *directly measured dimension*) is acquired immediately in each run, and $t_1$ (henceforth, *indirectly measured dimension*) is acquired point-by-point by repeating scans fixing different $t_1$ values. Finally we perform 2D FFT on the collected 2D data and get the frequency domain $(f_1, f_2)$. One important indication of this acquisition methodology is that we can freely subsample $t_1$ domain for a reduction of acquisition time, but subsampling $t_2$ will give no gain in terms of saving time.

Figure 2-2(c) shows a typical 2D COSY frequency domain. Note that we recovered it using our algorithm so that the windowing artifacts have already been removed. Comparing with its 1D counterpart, 2D COSY frequency domain is intrinsically sparse because it disentangles the overlapped and crowded peaks in 1D spectrum and posi-
tions them in a more spacious 2D layout. Only the cross-diagonal peaks are of interest because their positions \((f_1, f_2)\) indicate the J-coupling [5] frequencies and further indicate which metabolite it is. The frequency positions of human metabolites are well known [16] and are very sparse in the continuous frequency domain.

The major challenges of 2D COSY align with our goal: the repetitive acquisition methodology yields an unacceptable acquisition time for patients (full acquisition needs 25–40 minutes); and the heavy tails in the frequency domain (figure 1-1) obscure the useful cross-diagonal peaks.
Chapter 3

Related Work

In this chapter we will briefly outline the related work in three areas: sparse FFT, light field and MRS.

3.1 Sparse FFT

We build on recent work on the sparse Fourier transform, which shows that it is possible to compute the Fourier representation of a sparse signal using only a subset of its samples [14, 20, 21]. In our imaging applications, sparsity is well documented: the light field is sparse in the angular domain, and 2D COSY is sparse in the $f_1$ dimension. It should be possible to leverage this sparsity to recover the signal for both applications. The existing sparse Fourier algorithms, however, assume the signal is very sparse, i.e., the non-zero angular frequencies should be less than 2% to 3% (see Figure 3 in [21]). Typically, the discrete spectrum of the angular dimension in light field and the $f_1$ dimension in 2D COSY are not sufficiently sparse (as shown in figure 1-1). Thus, simply applying one of the existing sparse Fourier algorithms to light-field reconstruction would only recover the large frequencies and miss many of the small frequencies, producing a poor approximation. We introduce a new two-step algorithm that can handle this.
3.2 Light Field

The related work in light field solves one of the two problems: how to capture a light field efficiently given all the practical limitations, and how to reconstruct light field using different priors given limited samples.

Capturing Different capturing techniques present different trade-offs on angular/spatial resolution and bandwidth, the speed of capture and capture noise. Based on the trade-offs, we divide the capture techniques into two main categories:

The first category uses a plenoptic camera (i.e. light field camera), e.g., [1, 33, 13], or an array of camera [46] to capture the images. The upside is that the whole light field can be captured in one shot, and consequently the capture is easy and the object will likely to keep stable. The downside is that the spatial layout of cameras/lenslets is limited such that two different positions cannot overlap. Specifically in a plenoptic camera the total number of samples are limited so more lenslets (angular samples) means less pixels (spatial samples); in a camera array each camera will physically take some space that limits the resolution of angular samples. As a complement, [43] presents a coded-mask technique that multiplexes 4D information into 2D Fourier domain, but still fundamentally captures a full 4D dataset.

The second category captures the images one by one, instead of capturing them all at once. Robotic gantries [28] and handheld cameras [15, 7, 9] are two representatives. They are both more flexible in terms of angular/spatial resolution and bandwidth than the plenoptic and camera array techniques, but they exhibit other problems. First they are inappropriate for moving subjects; second they are hard to control the precision of angular positions. Mechanical controls of camera positions might be insufficient, so they often rely on image registration and other vision techniques to locate the camera positions either by on-line feedback or ex post estimation. But as pointed out in [9], even with visual feedback it is hard to capture more than a 1D trajectory. As a consequence, our algorithm chooses to take 1D trajectory as the sampling pattern.
Reconstruction  It is well accepted that light field is sparse so that we could reconstruct light field from only a subset of angular samples (or coded angular samples). The fundamental differences between them is the prior sparsity assumptions made by the reconstructions, as argued by [25, 27].

The most popular sparsity assumption is the dimensionality gap assumption (as introduced in sec 2.1) plus known depth. This prior knowledge subsumes that the scene is Lambertian so we have a well-formulated relationship such that given depth and spatial frequency position \((\omega_x, \omega_y)\) we could infer angular frequency position \((\omega_w, \omega_v)\). Representative work includes reconstruction from coded aperture [26, 43] or plenoptic camera [6].

Another sparsity assumption only assumes dimensionality gap but with no depth information. Basically this translates to the assumption that the energy of the light field is focused on a known 3D subspace so that it could be reconstructed by linear inference from only a 3D input. [25] uses only 1D trajectory of angular samples and could reconstruct Lambertian views; for a comparison with our technique please see section 6.2.5. However, the Lambertian assumption is often not true in a real scene with specularities, caustics, reflections and non-linear parallax.

A very recent work [30] assumes a different sparsity structure. It assumes that the light field can be sparsely represented using a dictionary of pre-trained light fields, i.e., new light fields will have similar structure to those in their training data.

In contrast to the prior knowledge listed above, we do not need any assumptions about the structure of the sparsity. We do not need to assume the dimensionality gap or extra information of depth, and we do not need a training data set. We use a 1D trajectory of the angular samples and the length of the 1D trajectory will constrain the level of sparsity we could reconstruct.

3.3 Magnetic Resonance Spectroscopy

In-Vivo 2D COSY  Although 2D Nuclear Magnetic Resonance (NMR) spectroscopy is widely used in chemistry, 2D Magnetic Resonance Spectroscopy mostly remains in
research and is rarely used in clinic practice. On one hand, 2D MRS techniques, especially 2D COSY, show great clinic potentials: it could unravel the overlapping resonances from metabolites in the 1D spectrum, such as GABA [37] and glutamate [45]; and it can detect new biomarkers which cannot be discovered by 1D techniques, e.g. 2-hydroxyglutarate (2HG) [2, 11] as an indicator of gliomas. On the other hand, although the developments in the methodology (e.g., new pulses [42, 3]) open up the possibility of in-vivo usage of 2D COSY, they all follow the pattern of iterative scanning (figure 2-2(b)) and require long acquisition time. Any direct reduction in time window will cause a loss in the reconstruction: reduction in directly measured dimension ($t_2$) or number of averages will reduce the SNR, and reduction in indirectly measured dimension ($t_1$) will exacerbate the windowing artifacts.

**Sparsity Based Reconstruction** The sparsity of the magnetic resonance spectra is well observed and there is a rich literature on the application of sparsity in NMR spectra: for example, compressive sensing techniques [23, 10] explore the unstructured sparsity, and multi-dimensional decomposition techniques [35] assume that the magnetic resonance spectrum is intrinsically low rank thus can be sparsely represented. Similarly, both unstructured sparsity [29] and structured sparsity [12] were explored for MRI techniques [29]. However, it is difficult to sparsely reconstruct in-vivo 2D MRS spectra and there is few work on it. One important reason is that it is not really sparse in discrete Fourier domain, because of the heavy windowing effects along indirectly measured dimension (figure 1-1) as well as other noises due to imperfection in acquisition. For instance, compressive sensing is argued to perform well in 2D COSY experiment with perfect conditions (simulation with no windowing artifacts) [38, 10], but we show it fails to give good reconstruction of real in-vivo data (section 7.3).

**Reducing Windowing Artifacts** Heavy windowing artifacts will obscure actual cross-diagonal peaks and blur the whole spectrum. There are some standard ways to deal with this in NMR and MRS practice: one way is to smoothen the time window in the indirectly measured dimension by multiplying SINE(sin) or QSINE(sinc$^2$) win-
dow [5]; another way is to enlarge the time window by linearly predicting the samples outside the acquisition window [24]. However, these techniques could only weaken the windowing artifacts but could not eliminate them; our algorithm will instead try to fit the spectrum into its continuous form with no windowing artifacts and thus remove the artifacts.
Chapter 4

Sparsity in the Discrete vs. Continuous Fourier Domain

The difference between the sparsity in the discrete and continuous Fourier domain is due to the windowing effect. This is a general phenomenon that occurs when one computes the discrete Fourier transform (DFT) of a natural signal using a relatively small number of samples. Since it is not limited to either light field or MRS, we will explain the concept using one-dimensional signals. It naturally extends to higher dimensions.

Consider computing the discrete Fourier transform of a time signal $y(t)$. To do so, we would sample the signal over a time window $[-\frac{A}{2}, \frac{A}{2}]$, then compute the DFT of the samples. Since the samples come from a limited window, it is as if we multiplied the original signal $y(t)$ by a boxcar function that is zero everywhere outside the acquisition window. Multiplication in the time domain translates into convolution in the Fourier domain. Since acquisition multiplies the signal by a boxcar, the resulting DFT will return the spectrum of the original signal $y(t)$ convolved with a sinc function.

Convolution with a sinc, in most cases, significantly reduces the sparsity of the original signal. To see how, consider a simple example where the signal $y(t)$ is one sinusoid, i.e., $y(t) = \exp(-2j\pi\omega t)$. The frequency domain of this signal has a single impulse at $\omega$. Say we sample the signal over a window $[-\frac{A}{2}, \frac{A}{2}]$, and take its DFT. The spectrum will be convolved with a sinc, as explained above. The DFT will discretize
this spectrum to the DFT grid points located at integer multiples of $\frac{1}{A}$. As can be seen in Fig. 4-1(a), if $\tilde{\omega}$ is an integer multiple of $\frac{1}{A}$, the gird points of the DFT will lie on the zeros of the sinc(·) function and we will get a single spike in the output of the DFT. However, if $\tilde{\omega}$ is a not an integer multiple of $\frac{1}{A}$, then the output of the DFT will have a sinc tail as shown in Fig. 4-1(b).

The same argument applies to MRS and light field. Here we pick the example of the DFT of the light field in the angular domain. The light field is sampled using a limited 2D window in the $uv$ coordinates. As a result, the DFT of each 2D angular slice, $\hat{L}_{\omega_u, \omega_v}(\omega_u, \omega_v)$, is convolved with a 2D sinc function reducing the overall sparsity. Fig. 4-2(a) shows the DFT of an angular slice from the crystal ball light field. The slice is sparse; however, as can be seen from the figure, these peaks exhibit tails that decay very slowly. These tails reduce the sparsity of the light field and prevent a good quality reconstruction. Fig. 4-2(b) shows an approximate version of the continuous Fourier transform of the same slice, as reconstructed by our algorithm. The continuous Fourier transform removes the windowing effect recovering the original sparsity in the light field.
Figure 4-1: The **windowing effect**: limiting the samples in the time domain to an aperture $A$ is equivalent to convolving with a sinc function. (a) If frequency spike lies on a DFT grid point, once the sinc is discretized, it disappears and the original sparsity of the spectrum is preserved. (b) If the frequency spike is off the grid, once we discretize we get a sinc tail and the spectrum is no longer as sparse as in the continuous domain.
Figure 4-2: A 2D angular slice of the 4D light field spectrum. It is taken from the Stanford crystal ball light field at $(\omega_x, \omega_y) = (50, 50)$. (a) In the discrete Fourier domain, we have sinc tails and the spectrum is not very sparse. (b) In the continuous Fourier domain, as reconstructed by our algorithm, the spectrum is much sparser. It is formed of 4 peaks which do not fall on the grid points of the DFT.
Chapter 5

The Reconstruction Algorithm

Our reconstruction algorithm is essentially the same for light field and MRS. For the sake of simplicity, in this chapter we will talk in the context of reconstructing the light field; MRS reconstruction will use the same algorithm with very slightly changes, and section 7.1 will highlight the differences.

5.1 Notation

Following our notations, we use \( L(x, y, u, v) \) to denote a 4D light field. \( \hat{L}(\omega_x, \omega_y, \omega_u, \omega_v) \) characterizes the 4D spectrum of this light field. We will use \( \hat{L}_{\omega_x, \omega_y}(\omega_u, \omega_v) \) to denote a 2D angular slice of this 4D spectrum for a fixed spatial frequencies \( \omega_x, \omega_y \). Similarly, \( L_{u,v}(x,y) \) will denote the 2D image captured by a camera at \( (u, v) \) coordinates. Table 5.1 presents a list of terms used throughout this paper.

5.2 Overview

The purpose of our reconstruction algorithm is to recover the entire 4D light field \( L(x, y, u, v) \) from a few 1D lines in the viewpoint coordinates. The algorithm focuses on reducing sampling along the viewpoint dimensions, and for each \( (u, v) \) viewpoint we have an image \( L_{u,v}(x,y) \) at full resolution.

The high-level structure of the algorithm is captured in the pseudocode in Alg. 5.2.1,
under the function \textsc{SparseLightField}. First, for each input image, $L_{uv}(x, y)$, the algorithm computes its 2D DFT. The algorithm then operates on an intermediate domain that represents the spatial frequencies $(\omega_x, \omega_y)$ across the viewpoints $(u, v)$. For each spatial frequency $(\omega_x, \omega_y)$, the algorithm has its value at a few viewpoints that lie on the set $S$ of the sampled 1D segments, which we will refer to as $\hat{L}_{\omega_x, \omega_y}(u, v)|_S$. Given this input, our objective is to recover for each spatial frequency $(\omega_x, \omega_y)$, its 2D angular spectrum $\hat{L}_{\omega_x, \omega_y}(\omega_u, \omega_v)$. Once we recover the 2D angular spectrum of all of the spatial frequencies, we have the full 4D light field spectrum $\hat{L}(\omega_x, \omega_y, \omega_u, \omega_v)$. We can then take the inverse Fourier transform to recover the full 4D light field $L(x, y, u, v)$.

Thus, for the rest of the description we can focus on a single spatial frequency $(\omega_x, \omega_y)$, and recover its 2D angular spectrum $\hat{L}_{\omega_x, \omega_y}(\omega_u, \omega_v)$ from its 1D set of sampled viewpoints $\hat{L}_{\omega_x, \omega_y}(u, v)|_S$. A shown in in Alg. 5.2.1, This process is performed by the function \textsc{2DSparseFFT} and has two stages:

- **On-Grid Recovery:** recovers the 2D spectrum assuming frequencies are only

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u, v$</td>
<td>angular/camera plane coordinates</td>
</tr>
<tr>
<td>$x, y$</td>
<td>spatial plane coordinates</td>
</tr>
<tr>
<td>$\omega_x, \omega_y$</td>
<td>spatial frequencies</td>
</tr>
<tr>
<td>$\omega_u, \omega_v$</td>
<td>angular frequencies</td>
</tr>
<tr>
<td>$L(x, y, u, v)$</td>
<td>4D light field kernel</td>
</tr>
<tr>
<td>$\hat{L}(\omega_x, \omega_y, \omega_u, \omega_v)$</td>
<td>4D light spectrum</td>
</tr>
<tr>
<td>$\hat{L}_{\omega_x, \omega_y}(\omega_u, \omega_v)$</td>
<td>a 2D slice for fixed spatial frequencies</td>
</tr>
<tr>
<td>$\hat{L}_{\omega_x, \omega_y}(u, v)$</td>
<td>a 2D angular slice of the 4D light spectrum</td>
</tr>
<tr>
<td>$X$</td>
<td>2D slice $= \hat{L}_{\omega_x, \omega_y}(u, v)$</td>
</tr>
<tr>
<td>$S$</td>
<td>set of samples $(u, v)$</td>
</tr>
<tr>
<td>$X</td>
<td>_S$</td>
</tr>
<tr>
<td>$x_S$</td>
<td>$X</td>
</tr>
<tr>
<td>$P$</td>
<td>set of frequency positions $(\omega_u, \omega_v)$</td>
</tr>
<tr>
<td>$\hat{x}_P$</td>
<td>$1 \times</td>
</tr>
<tr>
<td>$F$</td>
<td>set of positions and coefficients $(\omega_u, \omega_v, a)$</td>
</tr>
<tr>
<td>$[N]$</td>
<td>the set ${0, 1, \ldots N - 1}$</td>
</tr>
<tr>
<td>$y$</td>
<td>1D signal or line segment</td>
</tr>
<tr>
<td>$M \times M$</td>
<td>number of image pixels in spatial domain</td>
</tr>
<tr>
<td>$N \times N$</td>
<td>number of camera locations</td>
</tr>
</tbody>
</table>

Table 5.1: **Notation:** These are the notations used in chapter 5
\textbf{procedure} SPARSELIGHTFIELD(\textbf{L}\textsubscript{|S})
\begin{align*}
\hat{L}_{u,v}(\omega_x, \omega_y) &= \text{FFT}(L_{u,v}(x,y)) \text{ for } u, v \in S \\
\text{for } \omega_x, \omega_y \in [M] \text{ do} & \\
\hat{L}_{\omega_x, \omega_y}(\omega_y, \omega_v) &= \text{2DSPARSEFFT}(\hat{L}_{\omega_x, \omega_y}(u,v)\lvert_S) \\
L(x,y,u,v) &= \text{IFFT}(\hat{L}_{\omega_x, \omega_y, \omega_u, \omega_v}) \\
\text{return } L
\end{align*}
\textbf{procedure} 2DSPARSEFFT(X\textsubscript{|S})
\begin{align*}
F, e &= \text{ONGRIDRECOVERY}(X\textsubscript{|S}) \\
F, e &= \text{OFFGRIDRECOVERY}(X\textsubscript{|S}, F, e) \\
X(u,v) &= \sum_F a \cdot \exp\left(2j\pi \frac{\omega_u u + \omega_v v}{N}\right) \text{ for } u, v \in [N] \\
\hat{X} &= \text{FFT}(X) \\
\text{return } \hat{X}
\end{align*}
\textbf{Algorithm 5.2.1:} Light Field Reconstruction Algorithm

located on integer multiples of the discrete Fourier grid. This step only gives an initial estimate of the spectrum and alone cannot give a good quality reconstruction.

- **Off-Grid Recovery:** refines the frequencies discovered in the previous stage, allowing them to take non-integer positions. As such it attempts to approximate the continuous spectrum.

Each of the above stages outputs a list \(F\) of the large Fourier coefficients and their positions \((a, \omega_u, \omega_v)\). In the first stage, \((\omega_u, \omega_v)\) are integers (i.e., the coefficients lie on the DFT grid) whereas in the second stage they can be non-integers.

As said above, given the list of Fourier coefficients and their positions, one can invert the Fourier transform to recover the full light field. To do so, however, we cannot simply use the inverse DFT (i.e., the IFFT) because the frequency positions \((\omega_u, \omega_v)\) are no longer discrete. We have to invert the continuous Fourier transform which is the sum of the complex exponentials weighted by the frequency coefficients.

Thus, given a set \(F\) of non-discrete frequencies \((\omega_u, \omega_v)\) with value \(a\), we can calculate each unknown sample \(\hat{L}_{\omega_x, \omega_y}(u,v)\) as:

\begin{align*}
\hat{L}_{\omega_x, \omega_y}(u,v) &= \sum_{(a, \omega_u, \omega_v) \in F} a \cdot \frac{1}{N} \exp\left(2j\pi \frac{u\omega_u + v\omega_v}{N}\right).
\end{align*} (5.1)
An alternative would be to reconstruct the sinc tails in the frequency domain, discretize the frequencies and then take the inverse FFT. The frequency reconstruction is given by the following equation:

$$\hat{L}_{\omega_x, \omega_y}(\omega'_u, \omega'_v) = \sum_{(a, \omega_u, \omega_v) \in P} \text{sinc}_N(\omega'_u - \omega_u) \cdot \text{sinc}_N(\omega'_v - \omega_v)$$

(5.2)

where \( \text{sinc}_N(\cdot) \) is the discrete sinc function

$$\text{sinc}_N(x) = \exp \left(-\pi j \frac{N - 1}{N} x\right) \cdot \frac{\sin(\pi x)}{N \sin \left(\frac{\pi x}{N}\right)}$$

(5.3)

The above equation allows us to reconstruct the samples for any \((u, v)\) position; even the ones outside the aperture of the recorded views. Simply by setting \((u, v)\) in the above equation to a new viewpoint, we are able to extend the views. This would not be possible if \((\omega_u, \omega_v)\) were integers since then the above equation would be periodic modulo \(N\) and trying to reconstruct views outside our aperture would yield the same views as inside the aperture. In contrast, by finding the off-the-grid positions of the large-amplitude frequencies we can extend our reconstruction to viewpoints outside the aperture.

Below, we describe in detail On-Grid Recovery and Off-Grid Recovery. To simplify notations, we will use \(X\) to denote the 2D slice \(\hat{L}_{\omega_x, \omega_y}(u, v)\) in the intermediate domain and \(\hat{X}\) to denote its DFT \(\hat{L}_{\omega_x, \omega_y}(\omega_u, \omega_v)\). Thus, our algorithm takes as input a sparse set of samples of \(X\). We will denote these samples as \(X_{|S}\) where the set \(S\) corresponds to the coordinates of these samples (positions of the \((u, v)\) viewpoints). The sampling pattern forms a set of line segments. Different sampling patterns are shown in Fig. 5-2.

Fig. 5-1 shows a flow chart of the entire reconstruction.

### 5.3 On-Grid Recovery

The On-Grid recovery process can be divided into three steps: compute projections, recover frequency positions and recover frequency coefficients.
Figure 5-1: **Flow chart of the 2D sparse FFT reconstruction algorithm.** The algorithm takes a set of sampled line segments. The first stage, On-Grid Recovery, has 3 steps: computing the projections, recovering the integer positions, and recovering the coefficients of the large frequencies. This stage outputs a list of on-grid positions and coefficients of the large frequencies. The second stage is Off-Grid Recovery. In this step, the gradient search tries to shift the positions of the frequencies to non-integer locations and recover their coefficients. We keep repeating this gradient search until we get a small enough error. This stage will output a list of off-grid frequency positions and their coefficients which can then be used to reconstruct the full 2D slice.

**5.3.1 Compute Projections:**

As explained above our input samples lie on line segments (Fig. 5-2). We perform a 1D DFT for each line segment. Per the slicing theorem, this yields the projection of the 2D spectrum onto the corresponding line in the Fourier domain. Said differently, let \( y \) be a 1D line segment of the 2D slice \( X \). Then, \( \hat{y} \), the frequency spectrum of \( y \), is a projection of \( \hat{X} \) onto this line, that is each point in \( \hat{y} \) is a summation of the \( N \) frequencies that lie on a line orthogonal to \( y \) as shown in Fig. 5-3. For simplicity, we will illustrate only 3 line segments: a row, a column, and a diagonal, but our
practical sampling patterns usually involve 6 lines (Fig. 5-2.)

Figure 5-3: **Computing the DFT of a line segment of a 2D signal is equivalent to projecting the 2D spectrum on the line segment.** (a) Row projection (b) Column projection (c) Diagonal projection

### 5.3.2 Recover Frequency Positions:

Now that we have projections of the 2D spectrum on line segments, we want to find the positions of the large frequencies. To do so, we use a voting based approach where large projections vote for the frequencies that map to them, similar to a Hough transform. Since the spectrum is sparse, most of the projections do not have power and hence only few frequencies get votes each time. After performing a few projections, the large-amplitude frequencies will have the largest number of votes.

Consider a simple example shown in Fig. 5-4(a). The 2D spectrum has only 3 large frequencies at (5, 5), (5, 9) and (9, 5). When we project on a row, the 5th and 9th entry of the projection will be large and this projection will vote for all frequencies in the 5th and 9th columns. Similarly, when we project onto a column, the projection will vote for all frequencies in the 5th and 9th rows. At this point, only frequencies (5, 5), (5, 9), (9, 5), (9, 9) have two votes. However, when we project on the diagonal, frequency (9, 9) will not get a vote since its projection is not large. After 3 projections only the large frequencies will get 3 votes. Another example is shown in Fig. 5-4(b).

### 5.3.3 Recover Frequency Coefficients:

Now that we have a list of large frequency positions $P = \{(\omega_u, \omega_v)\}$, we want to find the Fourier coefficient $a$ corresponding to each of these positions and return
Figure 5-4: Two examples of the voting procedure used to recover the discrete positions of the large frequencies from projections on the line segments. The 2D spectrum is projected on a row, a column and a diagonal. Each large projection votes for the frequencies that map to it. (a) Frequencies (5,5), (5,9), and (9,5) are large and they all get 3 votes. (b) Frequencies on the diagonal are large and only these frequencies get 3 votes.
the list $F = \{(a, \omega_u, \omega_v)\}$. We have as input a set of samples $X(u, v)$ for $(u, v) \in S$. As described earlier in Eq. 5.1, each of these samples is the sum of complex exponentials weighted by the frequency coefficients. Since we know the positions of these frequencies, we know the complex exponentials in Eq. 5.1. Hence, we can view each of the input samples as a linear combination of the frequency coefficients we want to recover. So by constructing and solving a set of linear equations we can recover these frequency coefficients.

To construct the system of linear equations, we concatenate the input $(u, v)$ samples $X|_S$ into an $|S| \times 1$ vector which we denotes as $x_S$. We let $\hat{x}_P$ be a $|P| \times 1$ vector of the frequency coefficients which we want to recover. Each coefficient in $\hat{x}_P$ corresponds to the frequency position $(\omega_u, \omega_v)$ in $P$. Finally, let $A_P$ be a matrix of $|S| \times |P|$ entries. Each row corresponds to a $(u, v)$ sample and each column corresponds to an $(\omega_u, \omega_v)$ frequency and the value of each entry will be a complex exponential:

$$A_P((u, v), (\omega_u, \omega_v)) = \exp \left(\frac{2j\pi u\omega_u + v\omega_v}{N}\right) \tag{5.4}$$

Thus our system of linear equations becomes:

$$x_S = A_P\hat{x}_P \tag{5.5}$$

Our goal is to recover the values that best minimize the residual error on the input samples $x_S$. This error depends on the positions $P$ of the large frequencies and can be written as:

$$e(P) = \|x_S - A_P\hat{x}_P\|^2 \tag{5.6}$$

and $\hat{x}_P$ that best minimize this error is given by

$$\hat{x}_P^* = A_P^\dagger x_S \tag{5.7}$$

where $A_P^\dagger$ is the pseudo-inverse of $A_P$. Once we calculate $\hat{x}_P^*$, each coefficient will correspond to the position of a large frequency $(\omega_u, \omega_v) \in P$. This procedure of recov-
ering frequency coefficients given their positions does not assume that the frequencies are integers which allows us to use it again in the Off-Grid recovery process.

A pseudocode for this On-Grid recovery process can be found in Alg. 5.3.1. The description so far has ignored the need for a threshold, to allow us to focus on large coefficients and make the system well-conditioned. If we set this threshold too large, the voting process might miss some frequencies. If we set it too small then some empty frequencies will get enough votes to be considered large in which case the number of large frequencies $|P|$ will be more than the number of input samples $|S|$ and the system of linear equations (5.7) will be under-determined. Thus, to set this power threshold, we start from a small value and adaptively increase it until the system of linear equations becomes well determined i.e. we pick the smallest threshold such that $\text{rank}(A_P) \geq |P|$.

5.4 Off-Grid Recovery

The Off-Grid recovery process refines the frequencies discovered in the On-Grid stage, allowing them to take non-integer positions. As such it attempts to approximate the continuous spectrum.

We formalize the computation of non-integer frequencies as an optimization problem and we use a gradient descent algorithm based on finite differences to solve this problem.

Our objective is to minimize the residual error between the original set of input samples $X|S$ and our reconstruction. Again, we concatenate the input samples into a $|S| \times 1$ vector which we denote $x_S$ and call $x^*_S$ our reconstruction of these samples. The unknowns are in $P$, the list of positions of large-amplitude frequencies and the optimization is:

$$\text{minimize } c(P) = \|x_S - x^*_S(P)\|^2$$  \hspace{1cm} (5.8)

$x^*_S(P)$ can be obtained from the frequency positions $P$ and the input $x_S$ using the same linear approach as in the previous section. The matrix $A_P$ is constructed using the same formula, and the only difference is that $P$ might now contain non-integer
procedure \textsc{OnGridRecovery}(\textit{X}_{iS})
\begin{align*}
\hat{\textit{y}}_1 &= \text{PROJECTLINE}(\textit{X}_{iS}, 0, 1, 0, 0) \\
\hat{\textit{y}}_2 &= \text{PROJECTLINE}(\textit{X}_{iS}, 1, 0, 0, 0) \\
\hat{\textit{y}}_3 &= \text{PROJECTLINE}(\textit{X}_{iS}, 1, 1, 0, 0) \\
P &= \text{RECOVERPOSITIONS}(\hat{\textit{y}}_1, \hat{\textit{y}}_2, \hat{\textit{y}}_3) \\
F, e &= \text{RECOVERCOEFFICIENT}(\textit{X}_{iS}, P) \\
\text{return } F, e
\end{align*}

procedure \textsc{PROJECTLINE}(\textit{X}_{iS}, \alpha_u, \alpha_v, \tau_u, \tau_v)
\begin{align*}
y(i) &= X(i \alpha_u + \tau_u, i \alpha_v + \tau_v) \text{ for } i \in [N] \\
\hat{y} &= \text{FFT}(y) \\
\text{return } \hat{y}
\end{align*}

procedure \textsc{RECOVERPOSITIONS}(\hat{\textit{y}}_1, \hat{\textit{y}}_2, \hat{\textit{y}}_3)
\begin{align*}
V_1 &= \text{VOTE}(\hat{\textit{y}}_1, 0, 1, 0, 0, \theta) \\
V_2 &= \text{VOTE}(\hat{\textit{y}}_2, 1, 0, 0, 0, \theta) \\
V_3 &= \text{VOTE}(\hat{\textit{y}}_3, 1, 1, 0, 0, \theta) \\
P &= V_1 \cap V_2 \cap V_3 \\
\text{return } P
\end{align*}

procedure \textsc{VOTE}(\hat{y}, \alpha_u, \alpha_v, \tau_u, \tau_v, \theta)
\begin{align*}
I &= \{(i \alpha_u + \tau_u, i \alpha_v + \tau_v) : ||\hat{\textit{y}}(i)|| > \theta\} \\
V &= \{(u, v) : u = -\ell \alpha_v + u', v = \ell \alpha_u + v', \ell \in [N], (u', v') \in I\} \\
\text{return } V
\end{align*}

procedure \textsc{RECOVERCOEFFICIENT}(\textit{X}_{iS}, P)
\begin{align*}
\mathbf{A} &= \mathbf{0}_{|S| \times |P|} \\
\mathbf{x}_S &= \mathbf{0}_{|S| \times 1} \\
\text{for } i \in \{1, \ldots, |S|\} \text{ do} & \\
(u, v) &= S_i \\
\mathbf{x}_S(i) &= \mathbf{X}(u, v) \\
\text{for } k \in \{1, \ldots, |P|\} \text{ do} & \\
(\omega_u, \omega_v) &= P_k \\
\mathbf{A}(i, k) &= \exp (2j\pi \frac{\omega_u u + \omega_v v}{N}) \\
\hat{\mathbf{x}}_P &= \mathbf{A}^\dagger \mathbf{b} \quad \triangleright \text{is the pseudo-inverse of } \mathbf{A} \\
e &= ||\mathbf{x}_S - \mathbf{A} \hat{\mathbf{x}}_P||^2 \\
F &= \{(a, \omega_u, \omega_v) : \text{of corresponding entries in } F \text{ and } \hat{\mathbf{x}}_P\} \\
\text{return } F, e
\end{align*}

\textbf{Algorithm 5.3.1: On-Grid Recovery}
Figure 5-5: A demonstration of gradient search process: The gradient descent algorithm will shift the frequency by a small step in every iteration to minimize the error. (a) One frequency (4,5) is shifted to an off the grid position that best minimizes the error. It converges after 3rd iteration. (b) One fake frequency (blue) which was the result of an error in the first stage (On-Grid Recovery) and one actual large frequency (red). The algorithm keeps shifting the fake frequency until it crosses the integer boundary and merges with an actual large frequency thus correcting the error of the first stage.

frequencies. Once we estimate \( \hat{x}_P^* \), we can reconstruct our original input samples as:

\[
x^*_S = A_P \hat{x}_P^* = A_P A_P^\dagger x_S.
\]

Hence our optimization problem becomes:

\[
\text{minimize } e(P) = ||x_S - A_P A_P^\dagger x_S||^2
\]  

(5.9)

where the list of frequency positions \( P \) defines the coefficients of the matrix \( A_P \) in a non-linear manner.
5.4.1 Gradient Search

To solve the above optimization problem we use a gradient descent algorithm based on finite differences. Each iteration of the algorithm updates the list of frequency positions $P$. For each recovered frequency position in $P$, we fix all other frequencies and shift the position of this frequency by a small fractional step $\delta \ll 1$. We shift it in all 8 directions as shown in Fig. 5-5 and compute the new error $e(P)$ given the new position. We then pick the direction that best minimizes the error $e(P)$ and change the position of the frequency in that direction. ¹ We repeat this for every frequency position in $P$.

The gradient descent ensures that from iteration $i$ to iteration $i + 1$, we are always reducing the residual error, i.e., $e(P^{(i+1)}) < e(P^{(i)})$. The algorithm keeps iterating over the frequencies until the error $e(P)$ falls below a minimum acceptable error $\epsilon$. Once we have a final list of positions, we can use the same procedure described in On-Grid recovery to recover the coefficients of these frequencies. A pseudocode of this gradient search and Off-Grid recovery is provided in Alg. 5.4.1.

5.5 Algorithm Discussion

5.5.1 Speeding Up the Algorithm

The algorithm, as described hitherto, is slow. The major bottleneck is that we have to calculate the pseudo-inverse of matrix $A_P$ each time when we calculate the gradient based on finite differences (equ. 5.7). Suppose we use the widely-used Strassen algorithm [41] to compute matrix multiplication and inversion, this means that each time we shift the off-grid position of the peak, the time complexity we need to bear is $O(k^{\log_2 7}) = O(k^{2.807})$, where $k$ is the number of peaks.

The key observation here is that each time when we calculate the pseudo-inverse, we do not need to calculate from scratch, since only one column of $A_P$ is changed.

¹It is possible that none of the directions can minimize error in which case we do not change the position of this frequency.
procedure OFFGRIDRECOVERY(XiS, F, e)
\[ F(0) = F \]
\[ e(0) = e \]
\[ i = 0 \]
while \( e > e \) do
\[ F(i+1), e(i+1) = \text{GRADIENTSEARCH}(X_iS, F(i), e(i)) \]
\[ i + + \]
return \( F(i), e(i) \)

procedure GRADIENTSEARCH(XiS, F, e)
\[ P = \{(\omega_u, \omega_v) : (a, \omega_u, \omega_v) \in F\} \]
for \((\omega_u, \omega_v) \in P\) do
\[ (\Delta u, \Delta v) = \text{GETGRADIENT}(X_iS, P, e, \omega_u, \omega_v) \]
\[ (\omega_u, \omega_v) = (\omega_u, \omega_v) + (\delta \Delta u, \delta \Delta v) \]
\[ F', e' = \text{RECOVERCOEFFICIENT}(X_iS, P) \]
return \( F', e' \)

procedure GETGRADIENT(XiS, P, e, \omega_u, \omega_v)
\[ \Delta = \{(-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1, 0), (1,1)\} \]
for \((du, dv) \in \Delta\) do
\[ P' = P - \{(\omega_u, \omega_v)\} \]
\[ P' = P \cup \{(\omega_u + \delta du, \omega_v + \delta dv)\} \]
\[ F', e' = \text{RECOVERCOEFFICIENT}(X_iS, P') \]
\[ \delta e_{du, dv} = (e - e')/||\(du, dv)\|| \]
\[ (du^*, dv^*) = \arg\max_{(du, dv)\in \Delta} \delta e_{du, dv} \]
return \( (du^*, dv^*) \)

Algorithm 5.4.1: Gradient Search Algorithm
as we are shifting only one peak position at each time. This is defined as a rank-1 update of the matrix and it is well known that the Moore-Penrose pseudo-inverse of a rank-1 updated matrix is also a rank-1 update, i.e.,

$$(A + uv^T)^\dagger = A^\dagger + G$$

(5.10)

where $u$ and $v$ are column vectors and $G$ is a rank-1 update calculated from $u$ and $v$. The exact form of $G$ can be found in [31]. The calculation of $G$ only involves matrix-vector multiplication so the new time complexity is $O(k^2)$.

In our calculation $k$ is often tens or hundreds, so this trick itself could speed up the algorithm by a factor of tens to one hundred. And we could try to play with the parameters, e.g., the finite difference when calculating gradient and the error threshold when deciding the convergence, to reduce the number of iterations needed.

### 5.5.2 Add, Delete or Merge Peaks

In our off-grid recovery step, the gradient search algorithm will be able to not only change the positions of the peaks, but also change the set of peaks. There are three operations that are possible:

**Merge two peaks:** When two peaks are very near to each other, we will merge these two peaks into one peak. We do this by manually setting a threshold $D$, so that when the distance of the two peaks becomes smaller than $D$, we will merge them into one peak. We should not keep two peaks that is very near to each other, because this will make two columns of $A_P$ roughly the same which will render $A_P$ misconditioned. In figure 5-5(b), we give an example of merging two peaks.

**Add and Delete Peaks:** We will periodically check if there are any missing or fake peaks and the algorithm will try to correct it by adding or deleting corresponding peaks.

In each checkpoint, we will have a reconstruction $x^*_5(P)$ of the samples we have as
input. Now we run the on-grid voting algorithm, but with input $x_s - x^*_s(P)$. If any frequency wins the voting, it means that this is a big frequency in the spectrum of the difference between our reconstruction and ground truth. Such a big frequency is either a missing or a fake peak. We will then compare this frequency with the peaks we have in $P$, and see if it is already in $P$. If it is in $P$, then this is a fake peak and we should remove it from $P$; otherwise we will add it to $P$.

This checking is done periodically (e.g., every 10 iterations), instead of every iteration to avoid unnecessary computation.

5.5.3 Leveraging Light Field Priors

Our algorithm does not need any prior knowledge other than sparsity. However, we can make use of the common light field prior such as Lambertian and known depth to improve the performance.

Lambertian prior is essentially a restriction of where the peak positions can be. It can be loosely formulated as the following equation:

$$\alpha \frac{\omega_u}{\omega_x} = \beta \frac{\omega_v}{\omega_y} = f(d) \pm \epsilon$$

(5.11)

where $\alpha$ is the ratio between the unity one in $u$ dimension and $x$ dimension, $\beta$ is the ratio of unity one in $v$ over $y$ dimensions, and $f(d)$ is a deterministic function of depth $d$ (the exact form can be found in [32]). $\epsilon$ is the error term that takes care of the factors such as imperfection of acquisition, slightly non-Lambertian scene, etc.

The way we leverage Lambertian prior is by putting equation 5.11 as a constraint while in the on-grid phase and off-grid phase, i.e., we will not allow a peak to be more than $\epsilon$ further away than the expected line.

In furtherance, if we know the depth, it means that we know the position of $\omega_u, \omega_v$ in each $\hat{L}_{\omega_x,\omega_y}(\omega_u, \omega_v)$ slice, according to equation 5.11. Ideally we could directly recover the spectrum by the RECOVERCOEFFICIENT procedure in algorithm 5.3.1; a better approach will be using that as the initial value and perform off-grid gradient search on it to correct any imprecision of the prior.

53
Chapter 6

Light Field Reconstruction

6.1 Implementation

6.1.1 Data Sets

We have four light field data sets. Three of them, the Stanford bunny (sec. 6.2.1), the Amethyst (sec. 6.2.2) and the Crystal Ball (sec. 6.2.3) were from the Stanford light field archive [40]. Each of these data sets have $17 \times 17$ view points in angular domain; in spatial domain they vary in the spatial resolution.

The forth data set, Gnome (sec. 6.2.4), was captured by ourselves using a robotic gantry similar to the one from [40]. This data set has a much larger angular domain, which is $52 \times 52$, and it shows that our algorithm scales gracefully with the number of view points.

6.1.2 Sampling

The input to our algorithm is a set of images taken along 1D viewpoint trajectories. For all of the data set, we have full ground truth references for comparison, and we take only the images on our sampling pattern for reconstruction. The original images which are not used as sample input to our algorithm are then used as a comparison reference for our reconstruction.

For the Stanford light field archive data sets, we use the box-and-X pattern de-
scribed by figure 5-2(a); for the Gnome data set, we use the box-and-double-X pattern described by figure 5-2(c). Figure 6-1 shows an illustration of the samples we take. Both of the sampling patterns use 6 lines so the sampling ratio is roughly $6N/N^2 = 6/N$, where $N$ is the size of the angular window along either $u$ or $v$ dimension. Note that the sampling ratio goes down when $N$ increases, in Gnome data set we only use less than 12% of the samples.

Figure 6-1: Sampling Pattern of the Crystal Ball data set and the Gnome data set: All of the images from different view points of light field are arranged by their $(u, v)$ positions. The images that is used as input samples to our algorithm are highlighted while others are darkened. Crystal Ball and other Stanford light field archive data sets use the box-and-X sampling pattern, while Gnome uses box-and-double-X pattern.

### 6.1.3 Algorithm Implementation

We implemented our algorithm using C++ with the matrix operations implemented using the Eigen library. Our code was designed for flexible experimentation and is currently slow, the speed up tricks mentioned by section 5.5 is not implemented for simplicity. However, the algorithm is highly parallelizable so that we run it on a PC cluster. For our images with YUV channels, each channel can be assigned to different machines to run; and in order to speed up we subsample the U and V dimensions by 2. Moreover, the $\omega_u, \omega_v$ slices of a light field can be divided among different machines, and the results are collected once all of the machines have finished.
Figure 6-2: The reconstruction of Stanford Bunny data set from one \((u, v)\) viewpoint. On the left side are the reference figure and the reconstruction from [25]. The sampling pattern we used is the box-and-X pattern (fig. 5-2(a)). Though we used more samples than Levin & Durand used, we did a much better job in terms of less blurring, preserving the textures and having less noise.

6.2 Results

6.2.1 The Stanford Bunny

We use the Stanford Bunny data set as our simplest test case. The reason why it is simple is that it is mostly Lambertian and the distances between adjacent \(u, v\) samples are very small so there is little aliasing.

The size of angular dimensions are \(17 \times 17\) and the size of spatial dimensions are subsampled to \(512 \times 512\) form \(1024 \times 1024\).

Our reconstruction of the Bunny is difficult to distinguish from the full light field captured by [40], as shown in figure 6-2. As we look into the difference between our reconstruction and the reference, figure 6-3 shows that the reconstruction error is very small.

6.2.2 Amethyst

The Amethyst data set is an example of a strongly non-Lambertian scene. It has specularities and refraction. Given these difficulties, we still have very good reconstruction which is almost indistinguishable from the reference, as shown in figure 6-4.

Interestingly, some reflections move so fast that they never appear in our sampled
Figure 6-3: The color map of the Y channel for the difference between the reference and the reconstructed view from Stanford Bunny dataset. In about half of the pixels the differences are just zeros; there are some unstructured noises and it is hard to tell whether they come from the reference figure or our reconstruction; and there are structured noises on the edge of the bunny as well.

Figure 6-4: The reconstruction of Amethyst data set (right), the reference figure (left) and the reconstruction from [25] (middle). We are using the box-and-X sampling pattern (fig. 5-2(a)), which is more than the number of samples Levin & Durand used. However, we are able to reconstruct this highly non-Lambertian view and it is hard to distinguish our reconstruction from the full captured light field.

input. Figure 6-5 gives an example of this, and we could not reconstruct these fast movement. However, we believe that it would be very difficult for any algorithm to infer them because there is no clue about them in the input.
Figure 6-5: **One example of reconstructed view from Amethyst data set where we lose some reflection details.** The specular reflection only occurs in views outside our input set.

### 6.2.3 Crystal Ball

The Crystal Ball data set is our hardest data set. First, the scene is extremely non-Lambertian, since there are caustics, reflections, specularities, and nonlinear parallax. Second, the cartoon edges in the spatial domain are very sharp so that they create high frequencies, which harms the sparsity in high frequency $\hat{L}_{\omega_z, \omega_y}(\omega_u, \omega_v)$ slices (herein high frequency means high $\omega_z, \omega_y$ value) and also easily causes spatial domain to be aliased.

Despite the difficulty of the scene, we are able to reproduce most of the complex properties that make this scene so challenging (figure 6-6). If we look closely at the insets, we could find small structured noise in the reconstruction. We believe that this happens because the underlying spectrum is less sparse, and because of the intrinsic challenges brought by this scene.

### 6.2.4 Gnome

The Gnome data set is mostly Lambertian (there are just a few specularities such as the one highlighted in figure 6-7), and consequently its difficulty might lie between
Figure 6-6: One \((u, v)\) view from the reconstruction of the Crystal Ball data set. We are using box-and-X sampling pattern (as shown in the blue box in the bottom center) and the red dot shows the position of reconstructed view in angular domain. Despite the fact that the scene is extremely non-Lambertian and has complex structures, we could still reconstruct the images with most of its details.

the Stanford Bunny and the Amethyst data sets. However, this data set is more challenging than it looks on its face, because of the level of noise in our input. We could see shot noise, light flickering and registration errors ("jitter error", i.e., the camera is not precisely on its expected position).

The reconstruction still largely recovers the details of the ground truth. More importantly, the algorithm does kind of denoising, making the output arguably better than the reference images. Since the flickering nose and jitter noise are scattered as non-sparse noise in frequency domain, they are filtered out by the algorithm automatically; on the other hand, the unstructured shot noise gets restructured in our reconstruction. We have a more detailed discussion of noises in section 6.3.

6.2.5 Informal comparison with Levin and Durand [2010]

In this subsection, we will present an informal comparison with the reconstruction from [25]. Both of us reconstruct light fields from a 1D set of input images, but note
Figure 6-7: One \((u, v)\) view from the reconstruction of Gnome data set. We use sample patterns from fig. 5-2(c), as shown by the blue box in the bottom right. The red marker shows where the view is in angular domain. Although the captured data set is noisy, we could still reconstruct the data set with good details.

that these results should be taken with a grain of salt because we used more views than they do. An exact comparison is hard because their method needs to store the reconstruction matrix in memory, which limits the number of input samples, while we need full 1D segments that cover the synthetic aperture. On the other hand, if we count the extended views as reconstructions as well (sec.6.2.6), then we have far less sampling ratio that them. Figures 6-2 and 6-4 show our comparisons. It is promising and shows that we are less blurry and do not have ringing artifacts as in their Amethyst results.

Here is a brief summary of our pros and cons: we use more samples and reconstruct better result; we do not need to use the Lambertian assumption; and we can extend
our views outside the aperture of the input images while their reconstruction is limited to the convex hull of the input images.

6.2.6 Extending views

Figure 6-8: Extending views: We extend our reconstruction of Stanford Bunny dataset (fig 6-2) and extend the camera views. The original view is $0 \leq u \leq 16$ and $0 \leq v \leq 16$, and here we show our extension to $(-2, -2)$ and $(18, 18)$

The fact that we are reconstructing the continuous Fourier domain frequency positions and values makes it possible to extend the views, i.e., reconstruct views outside the original window of input. We demonstrate this in the Stanford Bunny data set, where we extend each of the $u$ and $v$ dimensions by an addition of 4 views, and increase the size of our reconstructed aperture by 53% (see figure 6-8).

6.2.7 Refocusing

Our reconstruction of the whole light field enables us to refocus the images to different depth planes [33]. Figure 6-9 gives a demonstration of refocusing using our reconstructed light field of the Amethyst data set.
6.3 Discussion

6.3.1 Viewpoint Denoising

Figure 6-9: **Refocus to different depth planes:** We use our reconstruction of the Amethyst data set and refocus it to the front and rear depth planes.

![A Front Refocus](image1)

![A Rear Refocus](image2)

Figure 6-10: **Viewpoint denoising demonstration:** Top: We see noise in the $u, v$ dimensions of our reference data caused by registration errors. This error shows up as camera shake in the reference images. Bottom: Our algorithm effectively removes this noise in the reconstruction, essentially performing camera stabilization.

In this subsection we discuss our algorithm’s ability to “denoise” the view point noises. The view point noises include but are not limited to the flickering noise and jitter noise, i.e., any man-made noise that hurts the smoothness between view points. This results from the advantage of reconstruction based on sparsity. Noisy input tends
to create low power high frequencies that are not part of our scene. These frequencies make the spectrum less sparse and are usually zeroed out by our algorithm. Since our reconstruction is based on sparsity in the $\omega_u, \omega_v$ domain, we remove noise in $u, v$.

Figure 6-10 is a demonstration of our algorithm removing viewpoint jitter noise. It shows a $v, y$ slice of our light field; these slices are often named as epipolar plane images (EPI) in computer vision. An oscillation in the line structure of the EPI figure traces back to a noisy camera viewpoint, and it is obvious that in our reconstruction these noises get eliminated. Consequently, the reconstructed camera motion in this video is much smoother than the reference camera motion. One way to think of this effect is as a kind of video stabilization.

However, the ability to denoise $(u, v)$ is not unlimited. It is capped by the number of input samples we have and the original sparsity of the spectrum we are reconstructing. If the noise affects the sparsity of our scene too much, some of its power might be projected onto existing spikes from our signal, changing their estimated power. One example is the shot noise in Gnome data set, it is not eliminated by our algorithm, but translated to a structured noise along the dominant orientation of the scene, which is vertical in this case.

6.3.2 Importance of continuous off-grid recovery

To better understand what is the role of off-grid reconstruction in our reconstruction, we try to isolate the effect of estimating the frequency domain in continuous Fourier domain and examine what the reconstruction would be if the reconstruction is in discrete domain. We choose Stanford Bunny data set as our test field, because the result of our full algorithm is almost indistinguishable from the reference data, and we can more reliably assume that the continuous sparsity reconstructed by our algorithm reasonably fits the true sparsity.

In the off-grid recovery (second phase), we refine the off-grid positions of frequencies used to approximate the spectrum. However, during this refinement peaks may cross integer boundaries, and there are procedures merging and adding peaks to correct the initial estimation, therefore the error of our initial estimate (figure 6-11 left)
Figure 6-11: **Comparison of our final reconstruction in the continuous domain to two alternative reconstructions in the discrete domain**: We compare our result (right) with the output of only the first step of our algorithm (left), as well as the discrete approximation of our result with sinc tails removed (middle).

may reflect more than the effect of discretization. As an attempt to better isolate the effect of discrete v.s. continuous domain, we round off the output of our complete reconstruction algorithm to on-grid positions, and remove any sinc tails that are caused by off-grid frequencies. This approximates the discrete spectrum that is closest to our continuous spectrum while exhibiting the same sparsity. The IDFT of this spectrum is shown in figure 6-11(middle). We could see that the major effect still remaining is the ghost.

To understand why forcing reconstruction in discrete Fourier domain results in ghosts in primal domain, we pick the inset in figure 6-11 and compare it across the \((u, v)\) domain. It turns out that the ghost effect is most severe at the edges of the \((u, v)\) domain. Recall that the discrete Fourier transform assumes that signals are periodic in the primal domain, and we attempt to reconstruct the spectrum using a small number of frequencies. As a result, the IDFT will attempt to smooth between images at opposite ends of the primal domain. Consequently, when viewpoints near the center are averaged (smoothed) with their neighbors the artifact is less noticeable because their neighbors are very similar. However, when this smoothing wraps around the edges of our aperture, we average between more dissimilar images and the ghosting becomes more severe. In other words, if we want to approximate the frequency domain
Figure 6-12: A demonstration of the ghost effect caused by removing sinc tails in frequency domain. We removed sinc tail from the spectrum of Stanford Bunny dataset, got the reconstruction and selected an inset from it (the same inset as in fig. 6-11). The figure shows how this inset changes across \((u, v)\) aperture (note that we subsampled the 17x17 aperture by 2) and the ghost effect gets stronger as we go from the center of the aperture to the edge.

by sparsity in the discrete domain, it will inevitably smoothen the opposite ends of the aperture and create ghosts.
Chapter 7

Magnetic Resonance Spectroscopy Reconstruction

In this chapter, we will talk about our reconstruction of 2D COSY spectra. In our following descriptions, we use $M(t_1, t_2)$ to denote the time domain signal, and $\widehat{M}(f_1, f_2)$ to denote its frequency domain. Similar to light field, we use $\widehat{M}_{f_2}(f_1)$ to denote a 1D $f_1$ spectrum with a fixed value $f_2$, and $\widehat{M}_{f_2}(t_1)$ represents the time domain of $\widehat{M}_{f_2}(f_1)$. Table 7.1 shows the notations used in this chapter.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1, f_1$</td>
<td>the indirectly measured dimension and its Fourier domain</td>
</tr>
<tr>
<td>$t_2, f_2$</td>
<td>the directly measured dimension and its Fourier domain</td>
</tr>
<tr>
<td>$M(t_1, t_2)$</td>
<td>the time domain of MRS signal</td>
</tr>
<tr>
<td>$\widehat{M}(f_1, f_2)$</td>
<td>the frequency spectrum of MRS</td>
</tr>
<tr>
<td>$\widehat{M}_{f_2}(f_1)$</td>
<td>a 1D spectrum of $\widehat{M}(f_1, f_2)$ along $f_1$ by fixing $f_2$</td>
</tr>
<tr>
<td>$\widehat{M}_{f_2}(t_1)$</td>
<td>the primal domain of $\widehat{M}_{f_2}(f_1)$</td>
</tr>
<tr>
<td>$N_1$</td>
<td>the number of full acquired samples along $t_1$ domain</td>
</tr>
<tr>
<td>$S_1$</td>
<td>the number of samples used by our reconstruction along $t_1$ domain</td>
</tr>
</tbody>
</table>

Table 7.1: Notation: These are the notations used in chapter 7

7.1 Reconstruction Algorithm

The algorithm we used for 2D COSY reconstruction is essentially the same with the one we used in light fields (chapter 5). However, we want to highlight a few points:
• The dimensionality: 2D COSY reconstruction has different dimensionalities with 4D light field. However, their reconstruction procedures resemble each other. In 4D light field, we reconstruct the frequency domain by first doing full FFT along $x, y$ domain, and then perform our reconstruction on angular domain slice by slice. Analogically, in 2D COSY reconstruction, we first perform full FFTs along $t_2$ domain, and then perform our reconstructions on $\widehat{M}_{f_2}(t_1)$ with each value of $f_2$.

• The on-grid recovery phase: the projection trick used in chapter 5 does not hold here because now each recovery is only 1D. However, instead of projecting a 2D slice into 1D lines, herein we project a 1D line into subsampled lines. In 2D reconstruction the FFT of a line equals the projection of frequency values onto that line; analogically in 1D reconstruction subsampling in time domain translates into aliasing in frequency domain. The voting scheme thereafter is largely the same here: each folded frequency will vote for all frequencies which are collapsed into it, and the one with the largest votes will win.

• The sinc tails: it is worth noting that in MRS reconstruction we zero out all of the sinc tails, as opposed to keeping them in light field reconstruction. The reason is that in light field we want to reconstruct the time domain and removing sinc tails will result in ghost effects (section 6.3.2); while in MRS the reconstruction is in frequency domain and the tails will obscure the cross-diagonal peaks.

The other parts of the algorithm remain the same except for a trivial reduction from 2D case to 1D case.

7.2 Experiment Setup

All of our experiments were performed on a whole-body 3T MR scanner (Tim Trio Siemens, Erlangen), and we used the COSY-LASER sequence (repetition time TR = 1.5s, echo time TE = 30 ms) [3] to acquire 2D COSY spectra. We programmed
our own $t_1$ sampling pattern through the functionalities provided by the machine. In each data we did a full acquisition which contained all $t_1$ samples, and also separately acquired a subset of $t_1$ samples following our sampling pattern. We compared our reconstruction from the subsampled acquisitions with reconstructions from the full acquisitions, since they are operated on the same phantom or volunteer. We collected different data sets using different $N_1$ and $S_1$ to examine our performance.

For each data set we collected, we perform the following reconstructions:

- **Sparse FFT**: We perform our reconstruction algorithm as described in section 7.1 on the subsampled acquisition which has only $S_1$ samples out of $N_1$.

- **Full FFT**: We directly pick all of the $N_1$ samples, perform QSINE window and linear prediction on $t_1$ domain and then do FFT.

- **FFT**: The same as Full FFT, except that only the first $S_1$ samples are picked from the full $N_1$ samples.

- **CS**: We perform Compressive sensing (iterative soft thresholding [10]) using $S_1$ randomly selected samples from the $N_1$ samples. Note that QSINE window is performed before compressive sensing, but linear prediction cannot be applied since we do not have continuous blocks of samples in $t_1$.

7.3 Results

7.3.1 Reconstructed Spectra

In this subsection we pick one phantom example and one in vivo example to show our quality of reconstruction of spectra. In the brain phantom $N_1 = 180, S_1 = 64$ and in the in vivo data $N_1 = 160, S_1 = 60$. The results are summarized in figures 7-1, 7-2, 7-3.

From the spectra we can clearly see that we removed the large windowing artifacts along $f_1$ dimension. Comparing with FFT and CS, we are able to recover more peaks and get a cleaner spectrum; even comparing with Full FFT, we are able to reconstruct
Figure 7-1: **Phantom/In vivo Data Result**, picking one $f_2$ column with $f_2 = 4.4\text{ppm}$. It shows that our reconstruction zeros out the tails most of the peaks, and more importantly, we can disentangle cross-diagonal peaks from heavy windowing artifacts.

### 7.3.2 Quantitative Analysis

In this subsection we evaluate our reconstruction using some quantitative metrics.

**Acquisition Time:** Table 7.2 shows the reduction of acquisition time from our reconstruction. Generally the acquisition time is reduced by a factor of 3–4 from full acquisitions, the average time is reduced from about 30min to 10min.
Figure 7-2: Reconstruction of a phantom data set and comparisons with FFT, Full FFT and CS. The brain phantom contains NAA, Lac, 2HG, Glu, Myo, Cho, GABA. Our reconstruction largely reduces the tails while keeping most of the cross-diagonal peaks. The only peak that exists in Full FFT but is missed by our reconstruction is 2HG. If we look at other peaks, for example, Myo, we could find that it gets successfully recovered by our algorithm, but it is totally missing in FFT and CS could only recover a very much weakened version of Myo.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>$S_1$</th>
<th>$N_1$</th>
<th>Full ACQ Time</th>
<th>Reduced ACQ Time</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>180</td>
<td>27</td>
<td>9</td>
<td>67%</td>
</tr>
<tr>
<td>2</td>
<td>60</td>
<td>160</td>
<td>24</td>
<td>9</td>
<td>62.5%</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
<td>256</td>
<td>38</td>
<td>10.5</td>
<td>72%</td>
</tr>
</tbody>
</table>

Table 7.2: The Acquisition Time: We tried three different acquisition patterns with $S_1/N_1$ equals 64/180, 60/160 and 72/256. Full ACQ (Acquisition) time and reduced ACQ (Acquisition) time are both measured in minutes and exclude the time to localize a voxel in the brain. We showed a factor of 3-4 reduction in acquisition time from full acquisitions.
Figure 7-3: **Reconstruction of an in vivo data set and comparisons with FFT, Full FFT and CS.** The metabolites that can be detected are NAA, Cho, Asp and Glu. Our reconstruction could eliminate the heavy windowing artifacts and disentangle the cross-diagonal peaks from tails. For Asp and Cho, they are mixed with windowing artifacts in Full FFT and it is hard to distinguish between them and the tails; in our reconstruction they are well separated.
**Line Width:** We measured the line width of peaks. In FFT, Full FFT and CS, the peaks themselves are convolved with sinc functions and get mixed with the heavy ringing tails from strong diagonal peaks. As a consequence, all of these artifacts will blur the cross-diagonal peaks and their line widths get undesirably bigger. In table 7.3, we show a reduction of 26% and 27% of line width in phantom data and in vivo data respectively.

<table>
<thead>
<tr>
<th></th>
<th>Phantom</th>
<th>In Vivo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse FFT</td>
<td>0.17</td>
<td>0.15</td>
</tr>
<tr>
<td>Full FFT</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>FFT</td>
<td>0.15</td>
<td>0.15</td>
</tr>
<tr>
<td>CS</td>
<td>0.21</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Table 7.3: **Line Width of NAA** (ppm): Our reconstruction could reduce the line width of Full FFT by 26%

<table>
<thead>
<tr>
<th></th>
<th>Phantom</th>
<th>In Vivo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse FFT</td>
<td>13.78</td>
<td>2.05</td>
</tr>
<tr>
<td>Full FFT</td>
<td>4.29</td>
<td>-11.7</td>
</tr>
<tr>
<td>FFT</td>
<td>0.84</td>
<td>-11.6</td>
</tr>
<tr>
<td>CS</td>
<td>-0.65</td>
<td>-13.3</td>
</tr>
</tbody>
</table>

Table 7.4: **Signal-to-Artifacts Ratio** (dB): Our reconstruction could improve the Signal-to-artifacts ratio by 13.75dB for in vivo experiment

**Signal-to-Artifacts Ratio:** We use *Signal-to-Artifacts Ratio* to measure the improvement of signal quality. The way we calculate it is adding up the energy of all cross-diagonal signals and dividing it by the energy of all artifacts and noises. Note that the diagonal which has strong energies is excluded from both the numerator and the denominator.

Table 7.4 presents our result. It shows that we improved the signal-to-artifacts ratio dramatically, by 9.5dB in phantom data and 13.75dB in in vivo data. This is because we effectively removed the sinc tails which are the major source of artifacts.
Chapter 8

Conclusion

In this thesis we extended the sparse FFT algorithm and showed its potentials in two imaging applications: 4D light field and Magnetic Resonance Spectroscopy. We addressed two major challenges: first, these applications are all sample-intensive and need to optimize the number of input samples; second, the spectra are more sparse in the continuous Fourier domain since the discretization will introduce windowing effects. In order to address these challenges, we designed a two-phases algorithm: it first reconstructs a coarse discrete spectrum and then refines it using gradient descent in the continuous Fourier domain. Our algorithm enables high-quality reconstruction of 4D light field with only a 1D trajectory of samples in the angular domain; and it reduces the acquisition time of MRS by a factor of 3-4. Moreover, it demonstrated improvement in the reconstruction quality even comparing with the fully acquired data in both applications. We believe our algorithm could be applied to other applications such as general NMR spectroscopy, computational photography beyond light field reconstruction, videos, etc.
Bibliography


