Seismic Data Processing with the Parallel Windowed Curvelet Transform

by

Fadhel Alhashim

B.Sc. Software Engineering, King Fahad University of Petroleum and Minerals, Dhahran, 2004

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Geophysics)

THE UNIVERSITY OF BRITISH COLUMBIA

August, 2009

Abstract

The process of obtaining high quality seismic images is very challenging when exploring new areas that have high complexities. The to be processed seismic data comes from the field noisy and commonly incomplete. Recently, major advances were accomplished in the area of coherent noise removal, for example, Surface Related Multiple Elimination (SRME).(Verschuur et al., 1992).

Predictive multiple elimination methods, such as SRME, consist of two steps: The first step is the prediction step, in this step multiples are predicted from the seismic data. The second step is the separation step in which primary reflection and surface related multiples are separated, this involves predicted multiples from the first step to be "matched" with the true multiples in the data and eventually removed (Verschuur, 2006; Wang et al., 2008). Wang et al.,2008 have introduced a robust Bayesian wavefield separation method to improve on the separation by matching methods. This method utilizes the effectiveness of using the multi scale and multi angular curvelet transform (Candès et al., 2006; Ying et al., 2005) in processing seismic images. The method produced excellent results and improved multiple removal. A considerable problem in the seismic processing field is the fact that seismic data are large and require a correspondingly large memory size and processing time. The fact that curvelets are redundant also increases the need for large memory to process seismic data.

In this thesis we propose a parallel aproach based windowing operator that divides large seismic data into smaller more managable datasets that can fit in memory so that it is possible to apply the Bayesian separation process in parallel with minimal harm to the image quality and data integrity. However, by dividing the data, we introduce discontinuities. We take these discontinuities into account and compare two ways that different windows may communicate. The first method is to communicate edge information at only two steps, namely, data scattering and gathering processes while applying the multiple separation on each window separately. The second method is to define our windowing operator as a global operator, which

exchanges window edge information at each forward and inverse curvelet transform. We discuss the trade off between the two methods trying to minimize complexity and I/O time spent in the process.

We test our windowing operator on a seismic denoising problem and then apply the windowing operator on our sparse-domain Bayesian primary-multiple seperation.

Bibliography

Candès, E. J., L. Demanet, D. L. Donoho, and L. Ying, 2006, Fast discrete curvelet transforms: Multiscale Modeling and Simulation, **5**, no. 3, 861–899.

Verschuur, D. J., 2006, Seismic multiple removal techniques: past, present and future: EAGE publications b.v. ed.

Verschuur, D. J., A. J. Berkhout, and C. P. A. Wapenaar, 1992, Adaptive surface-related multiple elimination: Geophysics, **57**.

Wang, D., R. Saab, O. Yilmaz, and F. J. Herrmann, 2008, Bayesian-signal separation by sparsity promotion: application to primary-multiple separation: Technical Report TR-2008-1, UBC Earth and Ocean Sciences Department.

Ying, L., L. Demanet, and E. J. Candès, 2005, 3-D discrete curvelet transform: Proceedings SPIE wavelets XI, San Diego, **5914**, 344–354.

Table of Contents

List of Tables

List of Figures

Preface

This thesis was prepared with Madagascar, a reproducible research software package available at rsf.sf.net.

A large amount of code was developed in the Seismic Laboratory for Imaging and Modeling (SLIM). The numerical algorithms and applications are mainly written in Python. with a few experiments written in Matlab. Early experiments were conducted using SLIMpy (slim.eos.ubc.ca/SLIMpy) a Python interface that exploits functionalities of seismic data processing packages, such as MADAGASCAR, through operator overloading.

Acknowledgments

I would like to thank my supervisor Felix Herrmann for his support and guidance to make this thesis possible. Also, I would like to thank every member of the SLIM team for being extremely helpful and providing information and help whenever needed.

I am very fortunate to have Michael Bostock and Doug Oldenburg as part of my supervisory committee. They provided me with some of the most insightful courses I have ever had.

I would like to thank my Management at Saudi Aramco who gave me the chance to study at UBC and provided support and help. They have funded and sponsored me throughout my studying period at UBC.

I would like to especially thank my wife Ghadeer who helped me during my studies at UBC and was my biggest motivator. She provided me with the strength to acheive my goals and keep going. I also cannot thank my parents enough for encouraging me to continue my quest for knowledge. A special thanks also go to my brother and my sisters.

Finally, I will not forget my best friends that have always stood by my side, especially Maitham Alhubail and Abdullah Alnajjar.

Thank you all.

To my parents.

Chapter 1

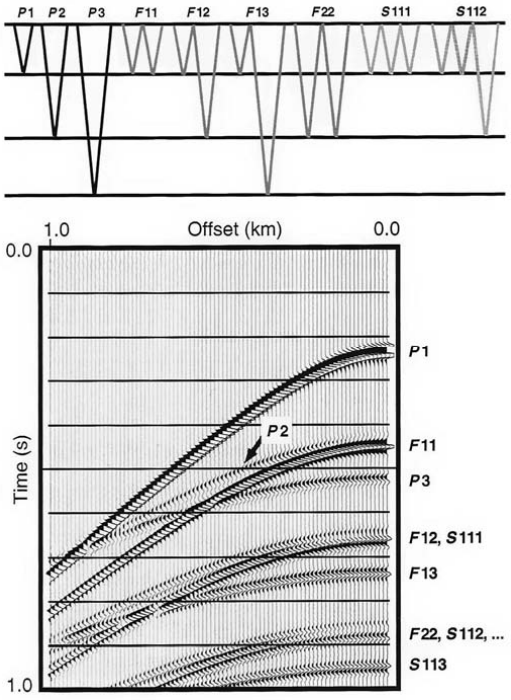
Introduction

In the early days of oil discovery, petroleum exploration was simply a matter of predicting the subsurface geology based on simple surface observations of signs such as seeps of oil or land formations related to salt domes and other basic observations. Not every prediction was accurate enough but it was a good start. Then, seismology came into the picture and improved the discovery in areas with less surface observations. The discovery was relatively easy and most of the simple structural traps have been discovered. Since then, geophysicists spent a considerable effort in developing and improving scientific methods of processing seismic data that gave huge success in targeting even more complex reservoirs.

Today, as we explore new regions, we face even more complex reservoirs to be discovered that consist of extremely thin strata or are very conformable to the surrounding strata. Such reservoirs may not be able to show on basic seismic data, they require the highest resolution possible. An added complexity to the problem is the fact that seismic data contain noise, ghosting, multiples and other kinds of events that obscure thin layers and complex structures in a seismic image. Such added noise and other undesired events do not completely destroy main features in images of basic structures. But a small amount of interfering coherent events can make a difference between finding an oil reservoir in a very complex subsurface structure and making profit, and, missing the target and losing millions of dollars. This is one of the major motivations in the oil industry to denoise complex seismic data and attenuate multiples (demultiple) or eliminate them (Ikelle and Amundsen, 2005).

Two major seismic events will be discussed in the coming chapters that are involved in the process of obtaining higher quality seismic images, specifically, in the process of surface-related multiple elimination. These events are primaries and multiples. Primaries are seismic events that have been reflected once before arriving at the receivers (or geophones). Primaries represent information about reflectors we are interested in that show the structure of subsurface. Since we are sending sound waves, any discontinuity (boundary, reflector...) in the path of these waves causes the wave to be

transmitted and reflected, which means that the recorded seismic data not only contain events that are reflected once but it contains events generated by waves traveling many possible paths before being recorded at seismic receivers. We call events that has been reflected more than once between the energy source and the receivers, multiples. Figure 1.1 shows a synthetic example of primaries and surface-related multiples. On the top a diagram showing the waves' paths and below is the seismic data.



(a)

Figure 1.1: Synthetic data consisting of three primaries and six surface related multiples. Top: the waves paths. Bottom: seismic data of the waves above. Notice that primaries reflect once while multiples reflect more than once resulting in some false horizons or reflectors on the seismic image. Adapted from Ikelle and Amundsen, 2005

As mentioned earlier, multiple attenuation or elimination (Verschuur et al., 1992) is an important part of seismic data processing and the process becomes complex with the presence of noise and other events. There is also the fact that seismic data can be incomplete due to physical or economical

factors. Many algorithms have been developed in the past, and still are, that aim to enhance seismic images with the available data by removing noise and filling in missing traces.

Seismic data is well known to be extremely large, and when this data is pre-processed data, the size becomes especially enormous and exceeds today's single computer memory (can easily exceed terabytes) which places a challenge on processing the data. To address this issue, it has always been part of seismic data processing to find the most efficient algorithm to deal with such huge amount of data without losing quality. In addition to finding efficient algorithms, parallelism is an obvious choice for handling huge data in any problem requiring more memory than a single computer can handle.

1.1 Theme

The main theme of this thesis is to utilize a robust technique for multiple elimination based on well-established sparsity-promoting methods and to make this method scalable to realistically sized data. The multiple elimination technique uses the redundant curvelet transform as a sparsifying domain. This redundancy , in conjunction with the large size of seismic data makes it impossible to fit seismic data in a single processing unit memory. To overcome these limitations, we design windowing schemes that distribute the data to multiple computing nodes such that it can fit in memory and run the algorithm with minimal quality loss. The trade-off between the image quality and running time is investigated.

1.2 Objectives

We have two main objectives: First, to allow a memory demanding iterative multiple elimination method to be applied to a large data set by dividing the data into smaller windows that fit in memory without losing data quality. Second, to compare two scenarios of using our windowing scheme in which, neighbor windows communicate and exchange information. The two scenarios are:

- Scenario A: Communication between windows occur at the windowing stage and in the final gathering stage only. After creating the windows and scattering the windowed data to different nodes, we run our primary-multiple separation on each window separately. This scenario has the advantages of simple implementation and fast running

time. But since we process each window separately we may lose data continuity and accuracy.

- Scenario B: Communication between windows occurs whenever there is a transform of the data to and from a sparse domain that assumes continuous data. In this scenario, we try to minimize the effect of partitioning our data by defining the sparsifying operator for each window such that they form an overlapping block diagonal operator. This scenario has the advantage of accuracy, but is more complex to implement and consumes a considerable amount of processing time in I/O to exchange information between windows.

Table 1.1 summarizes the advantages and disadvantages of the above scenarios.

We will test our windowing methods by applying the above scenarios to a seismic denoising problem. Then, we will apply the scenarios on our primary-multiple separation method.

	Advantages	Disadvantages
Scenario A	- speed - ease of implementation	- less accurate
Scenario B	- accuracy	- complex to implement - slow runtime (I/O between windows)

Table 1.1: Advantages and disadvantages of Scenario A: windowing data and applying separation seperatly, Scenario B: windowing data with overlapping operators.

1.3 Outline

The first chapter introduces the subject of high quality seismic imaging and the need to devise robust techniques to eliminate multiples. It contains a brief discussion on the concept of sparse data representation and introduces the curvelet transform used to represent seismic data sparsely.

In Chapter 2, we describe the denoising problem that we will use to test our technique. Then, we discuss a new development in wavefield separation by sparsity promotion (Saab et al., 2007; Wang et al., 2008) and explain the

basic idea behind the use of curvelets in our Bayesian formulation for the primary-multiple separation problem.

Chapter 3 introduces our windowing method to resolve the issue of scalability that makes it hard to process large seismic data in the redundant curvelet domain. Such windowing also allows us to process data in parallel. We compare two scenarios where we apply windowing and try to highlight trade-off between these two scenarios. Finally, we conduct several experiments on synthetic data starting with solving a seismic denoising problem, then, applying our windowing methods to our primary-multiple seperation method. The results are discussed in Chapter 4.

1.4 Theoretical background

In the world of seismic data processing, we are faced with many different challenges, one major challenge is the fact that we are trying to obtain a high quality image of the subsurface of the earth from a set of data that is noisy and incomplete due to physical and economical restrictions. There is also the fact that seismic data has bandwidth-limited wavefronts that come in many shapes, vary in frequency and directions and may contain conflicting dips and caustics (Herrmann and Hennenfent, 2008).

One of the most effective ways to face the above challenges is to work with seismic data in a sparse domain, which means that we transform the data to a domain that decomposes data into a form where most of the energy is concentrated in a small number of significant coefficients. This makes the denoising and primary-multiple separation problems simpler as we can work with a small set of coefficients that carry the most important energy from the data. In the case of data contaminated with white Gaussian noise, a good choice of transform will distribute the random noise into smaller less significant coefficients (Herrmann and Hennenfent, 2008; Hennenfent and Herrmann, 2008; Starck et al., 2002).

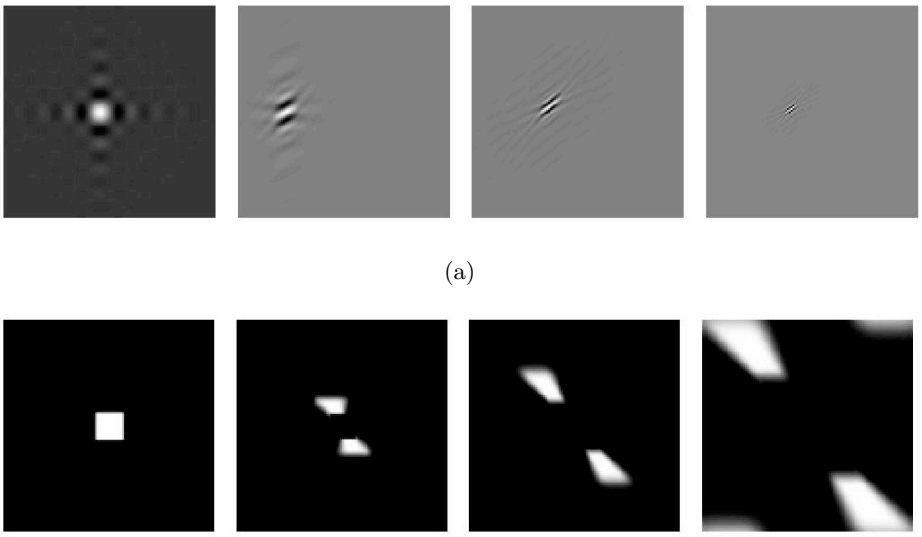
For a long time, seismologists have focused on exploiting the Fourier transform, which decomposes the seismic data into the sum of harmonic waves that have different frequencies. But the Fourier transform still has some shortcomings, it assumes that the wave fronts it is representing are plane monochromatic waves and it requires a large number of Fourier coefficients to represent wavefronts in seismic data. One way to over come the problem of representing wavefronts, is the use of the wavelet transform, which is a multi scale transform that produces localized coefficients. But wavelets do not have the sense of direction when it comes to two or higher

dimensional data, which means it cannot detect directional information on the wavefronts (Herrmann and Hennenfent, 2008).

Recently, the curvelet transform (Candès et al., 2006; Ying et al., 2005) was introduced and have successfully resolved the shortcomings mentioned above in the Fourier and wavelet transforms (Herrmann and Hennenfent, 2008). What makes the curvelet transform successful is the fact that curvelets are localized multi scale and multi-directional, they look like localized plane waves that are oscillating in one direction and smoothly varying in the perpendicular direction. Figure 1.2 shows four curvelets in the spatial and frequency domain, all of these curvelets have the same angle but different scales. They are arranged from coarsest to finest scale from left to right. notice that they are localized in the frequency domain. Curvelets follow a parabolic scaling principle, i.e. a curvelet's length and width are related according to $length = width^2$. The number of angles in the transform doubles every other scale. Without prior information, curvelets can find the location and direction of a wave front when a wave front and a curvelet that have the same direction and frequency content produce a large inner product between them (Herrmann and Hennenfent, 2008), this is illustrated in Figure 1.3.

One way to show how efficiently a sparse transform represent data is to take the data into the transform's domain and sort the coefficients in a descending order. This shows how rapid the coefficients decay. The rapid decay means that we can represent our data with few large coefficients. Fig. 1.4(b) shows a set of sorted coefficients of a synthetic seismic data in the Fourier, wavelet and curvelet domains. The x-axis represent the percentage of the number of coefficients and the y-axis shows the coefficients amplitude. We can see clearly that the curvelet domain has the fastet decaying rate. This means that the same seismic data can be represented with a smaller percentage of curvelet coefficients compared to wavelets and Fourier coefficients.

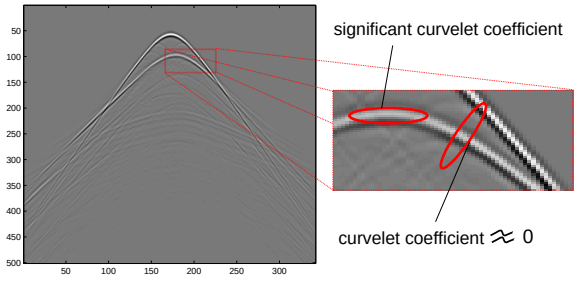
The fact that the curvelet transform represent seismic data in a small number of large coefficients that are parameterized by location, scale and angle, makes it unlikely for primaries and multiples to overlap. This makes it possible to successfully attempt to develop robust multiples attenuation algorithms. The computational complexity of the curvelet transform for a data of size M is $O(M \log M)$ and is not a major issue. However, the curvelet transform is an overcomplete signal representation and is 8 times redundant in 2D and 24 times redundant in 3D. This, in conjunction with the large size of seismic data, are the motivation for processing in parallel (Thomson et al., 2006).



(a)

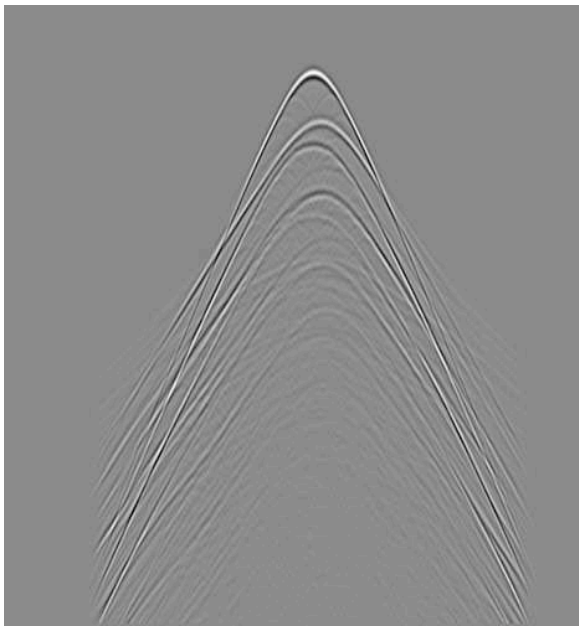
(b)

Figure 1.2: Four curvelets with the same angle but different scales starting from coarsest scale to finest scale left to right in the a) spatial domain b) Fourier or frequency domain

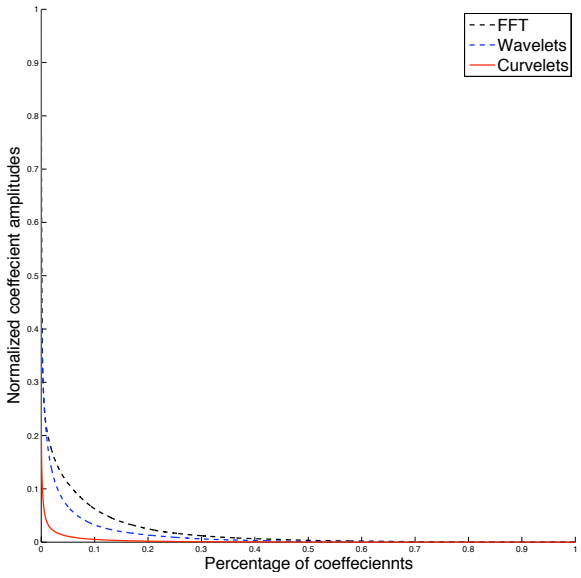


(a)

Figure 1.3: Principle of alignment. Curvelets and wavefronts that locally have the same frequency content and direction produce large inner products (significant coefficients). Adapted from Herrmann and Hennenfent, 2008.



(a)



(b)

Figure 1.4: a) synthetic seismic data b) coefficients of transforms sorted in a descending order. The x-axis represents the percentage of coefficients and the y-axis represents the amplitudes normalized to 1. This plot shows how the curvelet coefficients, shown in red, decay more rapidly than FFT and wavelet coefficients, shown in black and blue respectively

Bibliography

Candès, E. J., L. Demanet, D. L. Donoho, and L. Ying, 2006, Fast discrete curvelet transforms: Multiscale Modeling and Simulation, **5**, no. 3, 861–899.

Hennenfent, G. and F. J. Herrmann, 2008, Simply denoise: wavefield reconstruction via jittered undersampling: Geophysics, **73**, no. 3.

Herrmann, F. J. and G. Hennenfent, 2008, Non-parametric seismic data recovery with curvelet frames: Geophysical Journal International, **173**, 233–248.

Ikelle, L. T. and L. Amundsen, 2005, Introduction to petroleum seismology (investigations in geophysics, no. 12): Society of Exploration Geophysicists.

Saab, R., D. Wang, O. Yilmaz, and F. J. Herrmann, 2007, Curvelet-based primary-multiple separation from a Bayesian perspective: SEG Technical Program Expanded Abstracts, 2510–2514, SEG.

Starck, J.-L., E. J. Candès, and D. L. Donoho, 2002, The curvelet transform for image denoising: IEEE Transactions on Image Processing, **11**, no. 6, 670–684.

Thomson, D., G. Hennenfent, H. Modzelewski, and F. J. Herrmann, 2006, A parallel windowed fast discrete curvelet transform applied to seismic processing: SEG Technical Program Expanded Abstracts, 2767–2771, SEG.

Verschuur, D. J., A. J. Berkhout, and C. P. A. Wapenaar, 1992, Adaptive surface-related multiple elimination: Geophysics, **57**.

Wang, D., R. Saab, O. Yilmaz, and F. J. Herrmann, 2008, Bayesian-signal separation by sparsity promotion: application to primary-multiple separation: Technical Report TR-2008-1, UBC Earth and Ocean Sciences Department.

Ying, L., L. Demanet, and E. J. Candès, 2005, 3-D discrete curvelet transform: Proceedings SPIE wavelets XI, San Diego, **5914**, 344–354.

Chapter 2

Curvelet based seismic data processing

2.1 Curvelet based denoising

Seismic data come from the field contaminated with various types of noise that effect the final processed seismic image.

The problem of incoherent noise elimination can be cast into the following optimization problem:

$$\min \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \sigma, \tag{2.1}$$

where $\mathbf{A}$ is sparsifying operator that we choose to be the curvelet synthesis operator. The vector $\mathbf{b}$ is our data with added white Gaussian noise, and the positive parameter σ is an estimate of the noise level in the data. (van den Berg and Friedlander, 2008).

Figure 2.1 shows a synthetic data set that we will use to illustrate the denoising problem. Figure 2.1a shows the noise free synthetic data and Figure 2.1b shows the data after adding white Gaussian noise with standard deviation 1.4.

We solved the above denoising problem using SPGL1 (Berg and Friedlander, 2007), a solver for large-scale one-norm regularized least squares problems. Figure 2.1c shows our denoised output, which has an SNR value of 9.5. This was obtained after running for only 14 iterations.

We will use the above denoising problem later in this thesis to test operators that will be introduced in the next chapter.

2.2 Curvelet based primary-multiple separation

2.2.1 Introduction

Removal of multiples from seismic data is a vital part of producing high-quality seismic images. In this thesis our goal is to successfully eliminate

multiples from large seismic data in the presence of noise and possibly incomplete data. Major advances were accomplished in the multiples elimination area, e.g., Surface Related Multiple Elimination (SRME) (Verschuur et al., 1992) Predictive multiple elimination methods, such as SRME, consist of two steps: The first step is the prediction step, in this step multiples are predicted from the seismic data. The second step is the separation step in which primary reflection and noise are separated, this involves predicted multiples from the first step to be "matched" with the true multiples in the data and eventually removed (Verschuur, 2006; Wang et al., 2008).

In some situations, the subsurface produces three dimensional complexity on two dimensional data, which causes multiple predictions to be inaccurate. An added complication is the possibility of having ghosts, unbalanced amplitudes in multiple predictions (Herrmann et al., 2007a) and incomplete data (Herrmann et al., 2007c). Many attempts have been made to improve the process of multiple elimination by either producing more accurate predictions of the multiples (Herrmann, 2008; Herrmann et al., 2007c; Verschuur and Berkhout, 1997) or by developing more robust separation techniques (Herrmann et al., 2007a).

2.2.2 Bayesian primary-multiple separation by sparsity promotion

In a recent development, Wang et al., (2008) have introduced a robust Bayesian wavefield separation method to improve on the separation by matching methods. We will use this method as our main multiple elimination technique.

The separation problem is set up as a probabilistic framework. Seismic data come from the field as a mixture of primaries, multiples, noise and other recorded signals. Our main goal is to recover primaries from a mixture of primaries and multiples. The following will be our forward model:

$$\mathbf{b} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{n}, \tag{2.2}$$

where vector $\mathbf{b}$ is our total data and $\mathbf{s}_1$ and $\mathbf{s}_2$ denote the primaries and multiples respectively. The vector $\mathbf{n}$ is white Gaussian noise with each component being zero-mean Gaussian with standard deviation σ (Wang et al., 2008). We define the SRME predicted multiples as:

$$\mathbf{b}_2 = \mathbf{s}_2 + \mathbf{n}_2, \tag{2.3}$$

where $\mathbf{n}_2$ is the error in the SRME prediction, which we also assume to be

white Gaussian. The primaries can then be written as

$$\begin{aligned} \mathbf{b}_1 &= \mathbf{b} - \mathbf{b}_2 \\ &= \mathbf{s}_1 + \mathbf{n} - \mathbf{n}_2 \\ &= \mathbf{s}_1 + \mathbf{n}_1, \end{aligned} \tag{2.4}$$

We assume that $\mathbf{n}$ and $\mathbf{n}_2$ are independent (Wang et al., 2008). We can rewrite the unknown signals $\mathbf{s}_1$ and $\mathbf{s}_2$ as follows

$$\begin{aligned} \mathbf{s}_1 &= \mathbf{A}\mathbf{x}_1 \qquad \text{and} \\ \mathbf{s}_2 &= \mathbf{A}\mathbf{x}_2, \end{aligned} \tag{2.5}$$

where $\mathbf{A}$ is a sparse domain synthesis matrix, We choose $\mathbf{A}$ to be the curvelet synthesis matrix (Candès et al., 2006; Herrmann, 2006; Herrmann et al., 2007b). This gives us the following system:

$$\begin{aligned} \mathbf{b}_1 &= \mathbf{A}\mathbf{x}_1 + \mathbf{n}_1 \\ \mathbf{b}_2 &= \mathbf{A}\mathbf{x}_2 + \mathbf{n}_2. \end{aligned} \tag{2.6}$$

Now, given the SRME predictions $\mathbf{b}_1$ and $\mathbf{b}_2$, we want to maximize the conditional probability:

$$P(\mathbf{x}_1, \mathbf{x}_2 | \mathbf{b}_1, \mathbf{b}_2) = P(\mathbf{x}_1, \mathbf{x}_2) P(\mathbf{n}) P(\mathbf{n}_2) / P(\mathbf{b}_1, \mathbf{b}_2), \tag{2.7}$$

this is reformulated into the minimization function:

$$\min_{\mathbf{x}_1, \mathbf{x}_2} f(\mathbf{x}_1, \mathbf{x}_2) \tag{2.8}$$

$$f(\mathbf{x}_1, \mathbf{x}_2) = \lambda_1 \|\mathbf{x}_1\|_{1,\mathbf{w}_1} + \lambda_2 \|\mathbf{x}_2\|_{1,\mathbf{w}_2} + \|\mathbf{A}\mathbf{x}_2 - \mathbf{b}_2\|_2^2 + \eta \|\mathbf{A}(\mathbf{x}_1 + \mathbf{x}_2) - \mathbf{b}\|_2^2,$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are curvelet coefficients of the primaries and multiples respectively. The parameters λ_1 and λ_2 allow us to input priori information related to the expected sparsity of the estimated primaries and multiples respectively. The vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are the weights chosen based on empirical findings (Herrmann et al., 2007a) to be $\mathbf{w}_1 = \max\{|\mathbf{A}^{\mathrm{T}}\mathbf{b}_2|, \epsilon\}$ and $\mathbf{w}_2 = \max\{|\mathbf{A}^{\mathrm{T}}\mathbf{b}_1|, \epsilon\}$, where ϵ is a noise dependant constant. The parameter η controls the trade-off between coefficients vectors' sparsity and the misfit between the total data and the sum of the primaries and multiples. Reducing η increases the thresholding operators' aggressiveness, and increasing η reduces the thresholding operators' aggressiveness. To solve the above minimization problem, an iterative thresholding algorithm was derived from

Algorithm: Bayesian iterative method for wavefield separation

input: $\mathbf{b}_1$,$\mathbf{b}_2$,λ_1, λ_2, η, niter
$\tilde{\mathbf{x}}_1 = 0$,$\tilde{\mathbf{x}}_2 = 0$

threshold $\mathbf{w}_1 = \frac{\lambda_1 \left| \mathbf{A}^T \mathbf{b}_2 \right|}{2\eta}$
threshold $\mathbf{w}_2 = \frac{\lambda_2 \left| \mathbf{A}^T \mathbf{b}_1 \right|}{2(\eta+1)}$

$\hat{\mathbf{b}_1} = \mathbf{A}^T \mathbf{b}_1$
$\hat{\mathbf{b}_2} = \mathbf{A}^T \mathbf{b}_2$

for i = 1 : niter

$\mathbf{x}_1 = \hat{\mathbf{b}_2} - \mathbf{A}^T \mathbf{A} \tilde{\mathbf{x}}_2^n + \hat{\mathbf{b}_1} - \mathbf{A}^T \mathbf{A} \tilde{\mathbf{x}}_1^n + \tilde{\mathbf{x}}_1^n$
$\mathbf{x}_2 = \hat{\mathbf{b}_2} - \mathbf{A}^T \mathbf{A} \tilde{\mathbf{x}}_2^n + \frac{\eta}{\eta+1} \left(\hat{\mathbf{b}_1} - \mathbf{A}^T \mathbf{A} \tilde{\mathbf{x}}_1^n \right)$

$\tilde{\mathbf{x}}_\mathbf{1} = \frac{x_1}{|x_1|} \cdot \max\left(0, |\mathbf{x}_1| - |\mathbf{w}_1|\right)$
$\tilde{\mathbf{x}}_\mathbf{2} = \frac{x_2}{|x_2|} \cdot \max\left(0, |\mathbf{x}_2| - |\mathbf{w}_2|\right)$

end

Table 2.1: The iterative Bayesian wavefield separation algorithm introduced in Wang et al., 2008; Saab, 2008

the work of Daubechies et al. (2004) and so starting from initial estimates $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$ for several iterations, the n$^{\text{th}}$ iteration becomes

$$\begin{aligned} \mathbf{x}_1^{n+1} &= \mathbf{T}_{\frac{\lambda_1 w_1}{2\eta}} \left[\mathbf{A}^T \mathbf{b}_2 - \mathbf{A}^T \mathbf{A} \mathbf{x}_2^n + \mathbf{A}^T \mathbf{b}_1 - \mathbf{A}^T \mathbf{A} \mathbf{x}_1^n + \mathbf{x}_1^n \right] \\ \mathbf{x}_2^{n+1} &= \mathbf{T}_{\frac{\lambda_2 w_2}{2(1+\eta)}} \left[\mathbf{A}^T \mathbf{b}_2 - \mathbf{A}^T \mathbf{A} \mathbf{x}_2^n + \mathbf{x}_2^n + \frac{\eta}{\eta+1} \left(\mathbf{A}^T \mathbf{b}_1 - A^T \mathbf{A} \mathbf{x}_1^n \right) \right], \end{aligned} \tag{2.9}$$

where $\mathbf{T}_u$ is the element wise soft thresholding defined as

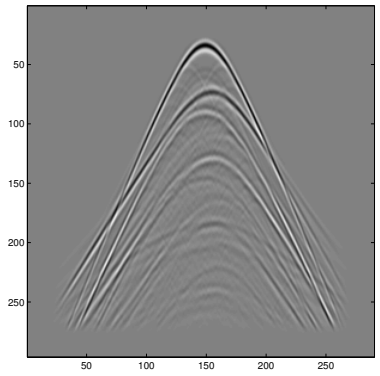
$$\mathbf{T}_{u_i}(v_i) := \frac{v_i}{|v_i|} \cdot \max\left(0, |v_i| - |u_i|\right), \tag{2.10}$$

The above algorithm, shown in Table 2.1, have proved to be a robust method for separating coherent sparse signal components. Using an initial prediction with moderate errors as input, the algorithm produces improved estimates of these predictions. Because the algorithm takes advantage of the curvelet domain, the algorithm has fast convergence and gives excellent quality output data(Wang et al., 2008). Figure 2.3 shows an example of running our Bayesian separation on a 2D data set. We ran the separation for 10 iterations using the parameters $\{\lambda_1^* = 0.8, \lambda_2^* = 1.2, \eta^* = 1.2\}$. These parameters values were found empirically. Notice the improvement in the estimated primaries, where we can see less multiples residual.

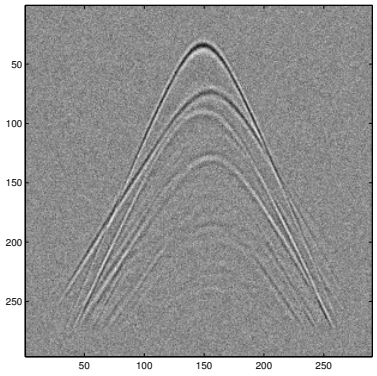
Table 2.2 shows sensitivity analysis for our Bayesian separation method. The table shows SNR values calculated against the "multiple-free" ground truth data, which is generated using the same simulation of the total data, only an energy absorbing boundary condition is enforced to prevent the generation of multiples. We can see from these SNR values that our separation technique is robust against changes in the control parameters. Parameters combinations that aggressively over threshold or under threshold are not included, since they produce extremely low SNR values.

SNR (dB)	$\{\lambda_1^*, \lambda_2^*\}$	$\{2 \cdot \lambda_1^*, \lambda_2^*\}$	$\{\lambda_1^*, 2 \cdot \lambda_2^*\}$	$100 \cdot \{\lambda_1^*, \lambda_2^*\}$
η^*	11.49	11.11	11.40	-
$\frac{1}{2} \cdot \eta^*$	11.29	10.38	11.15	-
$2 \cdot \eta^*$	10.90	11.38	10.81	-
$100 \cdot \eta^*$	-	-	-	10.99

Table 2.2: Calculated SNR of estimated primaries using the Bayesian separation. SNR was calculated against the ground truth "multiple-free" primaries.

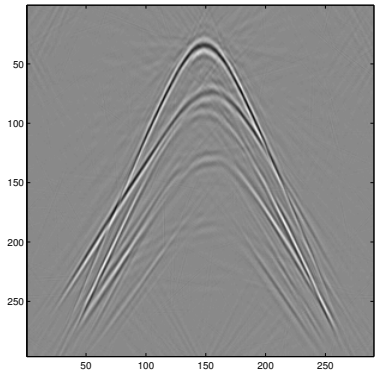


(a)



,1

(b)



(c)

Figure 2.1: a) Noise free data b) same data from a with added Gaussian noise with $\sigma = 1.4$ c) data after noise removal using SPGL1 and the curvelet domain

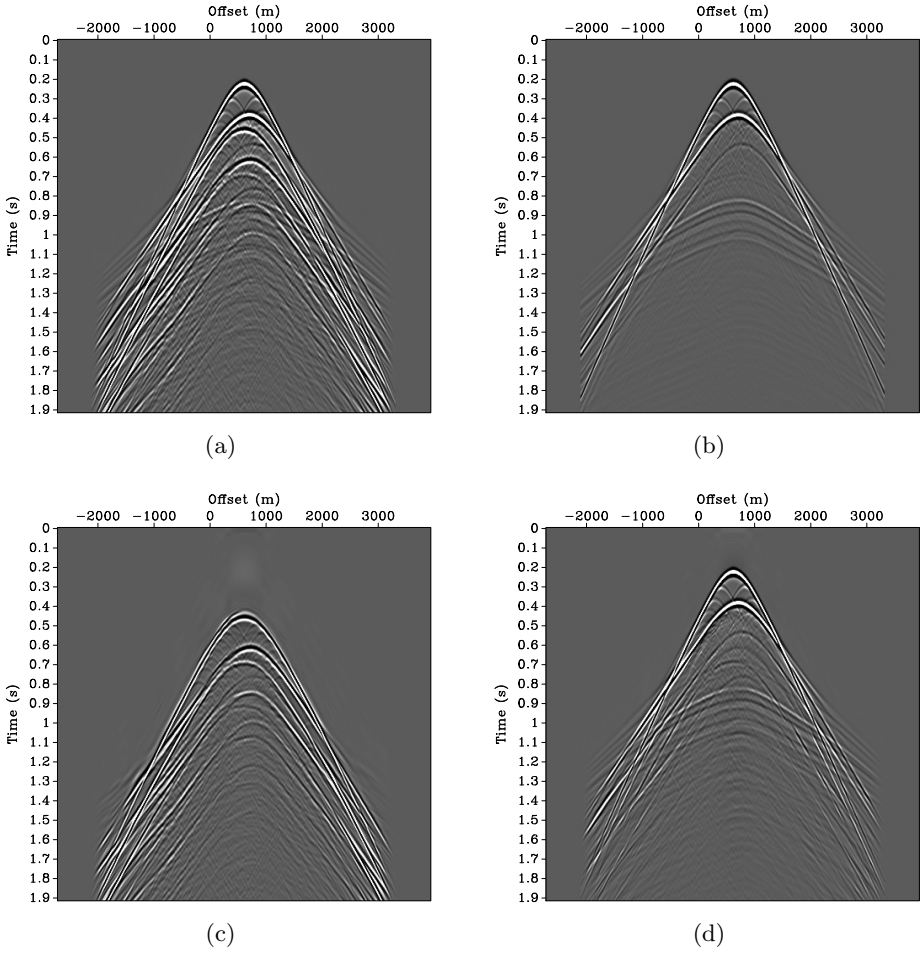


Figure 2.2: 2D synthetic data a) total data b) ground truth primaries c) SRME predicted multiples d) SRME predicted primaries

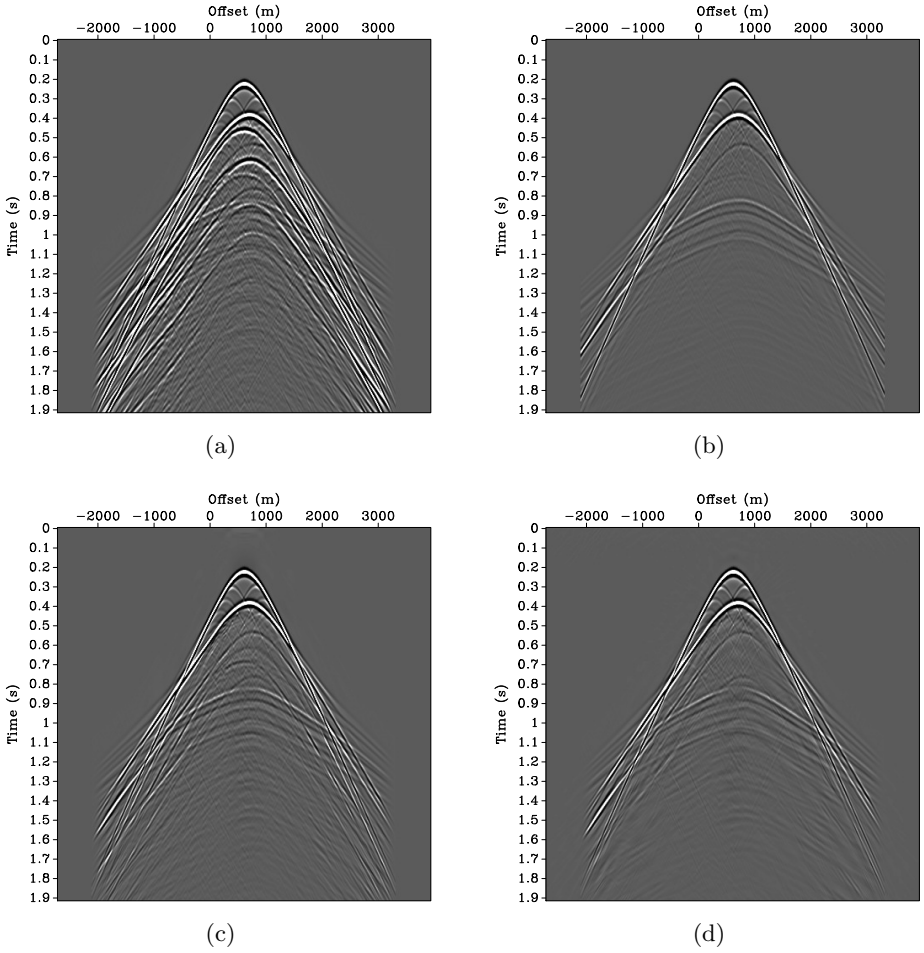


Figure 2.3: 2D synthetic data a) total data b) ground truth primaries c) SRME predicted primaries d) Our bayesian estimated primaries

Bibliography

Berg, E. v. and M. P. Friedlander, 2007, SPGL1: A solver for large-scale sparse reconstruction. (http://www.cs.ubc.ca/labs/scl/spgl1).

Candès, E. J., L. Demanet, D. L. Donoho, and L. Ying, 2006, Fast discrete curvelet transforms: Multiscale Modeling and Simulation, **5**, no. 3, 861–899.

Daubechies, I., M. Defrise, and C. De Mol, 2004, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint: Communications on Pure and Applied Mathematics, **57**, no. 11, 1413–1457.

Herrmann, F. J., 2006, A primer on sparsity transforms: curvelets and wave atoms: Presented at the SINBAD 2006.

———, 2008, Curvelet-domain matched filtering: SEG Technical Program Expanded Abstracts, 3643–3647, SEG.

Herrmann, F. J., U. Boeniger, and D. J. Verschuur, 2007a, Non-linear primary-multiple separation with directional curvelet frames: Geophysical Journal International, **170**, no. 2, 781–799.

Herrmann, F. J., G. Hennenfent, and P. P. Moghaddam, 2007b, Seismic imaging and processing with curvelets: Presented at the EAGE Technical Program Expanded Abstracts, EAGE.

Herrmann, F. J., D. Wang, and G. Hennenfent, 2007c, Multiple prediction from incomplete data with the focused curvelet transform: SEG Technical Program Expanded Abstracts, 2505–2509, SEG.

van den Berg, E. and M. P. Friedlander, 2008, Probing the Pareto frontier for basis pursuit solutions: Technical Report TR-2008-01, UBC Computer Science Department.

Verschuur, D. J., 2006, Seismic multiple removal techniques: past, present and future: EAGE publications b.v. ed.

Verschuur, D. J. and A. J. Berkhout, 1997, Estimation of multiple scattering by iterative inversion, part II: Practical aspects and examples: Geophysics, **62**, no. 5, 1596–1611.

Verschuur, D. J., A. J. Berkhout, and C. P. A. Wapenaar, 1992, Adaptive surface-related multiple elimination: Geophysics, **57**.

Wang, D., R. Saab, O. Yilmaz, and F. J. Herrmann, 2008, Bayesian-signal separation by sparsity promotion: application to primary-multiple separation: Technical Report TR-2008-1, UBC Earth and Ocean Sciences Department.

Chapter 3

Parallel windowed seismic data processing

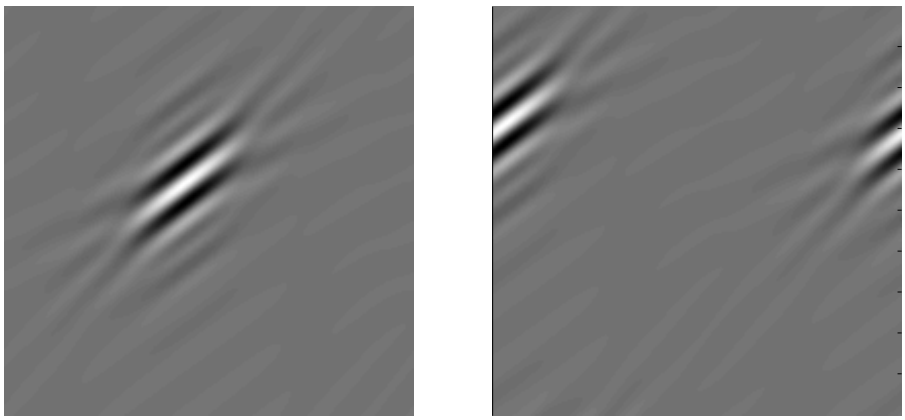
3.1 Parallel windowed curvelet transform

It is well known that seismic data sets are extremely large and can easily reach the size of several terabytes. Also, the two dimensional curvelet transform is 8 times redundant and the three dimensional transform is 24 times redundant; this means we will be dealing with 8 times the size of 2D data and 24 times the size of 3D data. This, in conjunction with the large size of seismic field data, makes it impossible to directly apply the curvelet transform to the full data into memory on a single processing unit. Currently, users are forced to work out of core to process relatively small 3D data sets.

Even though an MPI implementation of the curvelet transform exists (Ying et al., 2005), it has limitations in scalability since the curvelet transform is based on the Fast Fourier Transform, which requires large amounts of communication when the data set is distributed on different processing nodes (Thomson et al., 2006). Our sparsity promoting methods require repeated evaluation of matrix-vector multiplications, which multiplies the amount of communication required between processing nodes.

A possible scalable solution is to define a windowing operator that divides data into smaller manageable windows. Each window is dealt with separately, and when all windows are processed and ready, they are joined (gathered) to form the final output. This is a practical solution that minimizes communication between nodes. But it is likely to have problems at the borders of these windows, such as artifacts, dimming and/or other types of problems. For instance, problems arise from the fact that we may divide the data near a point where a curvelet coefficient would be located and so when we take the forward curvelet transform, the curvelet wraps around to the opposite border of the transformed data block, see Figure 3.1. This reduces the quality of our process and may introduce artifacts . Another issue is the fact that the curvelet transform is based on the fast Fourier

transform, which produces spectral leakage from aperiodic data. Spectral leakage is basically the spreading of the to-be-transformed signal's energy into other frequencies. A well known solution is the use of tapering which is basically multiplying the data by some function that smoothly reduces the edges' amplitudes to zero which minimizes discontinuity. This means we can improve our windowing operator by adding a tapering operator that affects the edges of each of the resulting windows. Figure 3.2 demonstrates spectral leakage from a monochromatic sinusoidal wave and shows the effect of a window function and how it limits the leakage.



(a) (b)

Figure 3.1: a) a centered wrapping curvelet b) the same wrapping curvelet located near the border. Notice how the curvelet wraps to the opposite border.

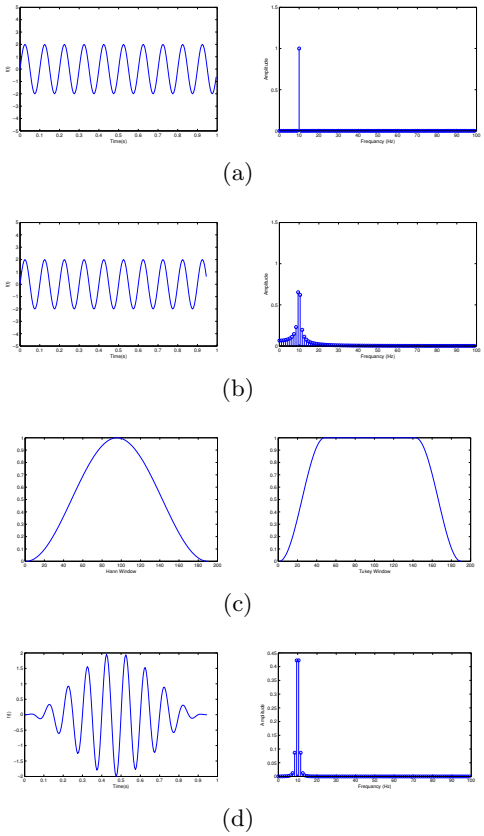
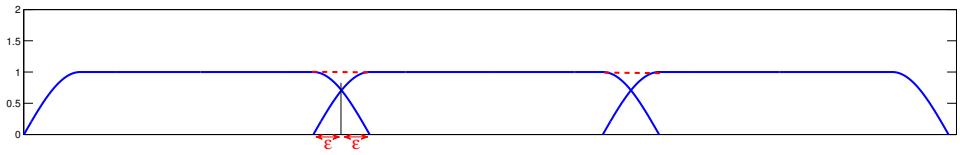


Figure 3.2: Spectral leakage a) a periodic function on the left and its Fourier transform on the right. b) an aperiodic function on the left and its Fourier transform on the right. c) different window functions d) The aperiodic function from (b) after multiplying by the second window function from c), and its Fourier transform on the right. We can see that applying the window function reduces spectral leakage

We can define our windowing operator to contain controllable overlapping regions at the windows' edges. This means that our forward windowing will contain information from adjacent windows. These windows will overlap in a region of total width of 2ϵ. The overlapping regions communicate via the way our tapering is defined and a suitable way of defining the tapering operator is by making sure it satisfies the following relation

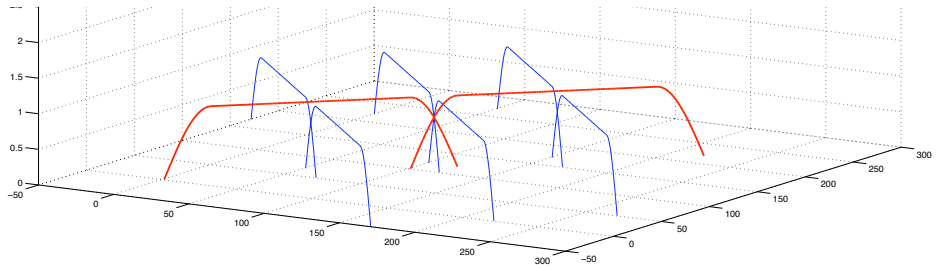
$$\mathbf{T}_1^2 + \mathbf{T}_2^2 = 1, \tag{3.1}$$

where $\mathbf{T}_1$ and $\mathbf{T}_2$ are overlapping tapering functions applied to adjacent windows. This relation guarantees that we have partition of unity, where all windows add up quadratically to one, preserving the system's energy. This allows us to define our windowing operator as a linear operator. Figure 3.3 illustrates how our tapering function looks like.



(a)

Figure 3.3: Three tapering windows with two overlapping regions. The overlapping region is 2ϵ wide. We can see the tapering functions being 1 everywhere except at the overlapping region where the sum of their square produces 1. The red dashed line represent that quadratic sum.



(a)

Figure 3.4: A simplified illustration of how overlapping and tapering would look like in 3D.

There are many functions that are suitable for tapering and satisfy the

above property (Mallat, 1998). We chose the following simple tapering operator:

$$\mathbf{T}_n = \sin(\frac{(N-n)\pi}{2(2\epsilon-1)}), \qquad n = \{N, N-1, ..., N-2\epsilon+1\}. \tag{3.2}$$

Here, N is the total width of the overlapping windows. Now we have set up our windowing operator with overlapping regions and a tapering operator applied to these windows' edges such that perfect reconstruction is ensured and energy of the system is preserved since our windowing and tapering operators satisfy

$$\mathbf{W}^*\mathbf{T}^*\mathbf{T}\mathbf{W} = \mathbf{I}, \tag{3.3}$$

where $\mathbf{W}$ is our windowing operator that divides our data into overlapping regions, and $\mathbf{W}^*$ is the adjoint windowing operator, which gathers the windowed data into one data set, adding the overlapping regions during the process. The operators $\mathbf{T}$ and $\mathbf{T}^*$ are our forward and adjoint tapering operators respectively. Finally, the matrix $\mathbf{I}$ represents the identity matrix. Figure 3.5 illustrates the way our combination of windowing and tapering works in 2D.

Now we have a windowing operator that permits us to fit a large data set in a distributed memory and perform operations in parallel to speed up our separation algorithm. We can define our parallel forward curvelet transform as:

$$\mathbf{A}^{\mathrm{T}} \quad = \quad [\mathbf{C}]\mathbf{T}\mathbf{W}. \tag{3.4}$$

The block diagonal matrix $[\mathbf{C}]$ is our forward curvelet transform matrix. Similarly, the parallel inverse curvelet transform can be defined as:

$$\mathbf{A} \quad = \quad \mathbf{W}^*\mathbf{T}^*[\mathbf{C}^*], \tag{3.5}$$

where the block diagonal matrix $[\mathbf{C}^*]$ is our inverse curvelet transform matrix.

3.2 Parallel windowed curvelet transform usage

We consider two options to incorporate the windowing operator with our sparsity promoting techniques. The first option, we will call scenario A, is to apply the forward windowing operator with overlaps and tapering and then

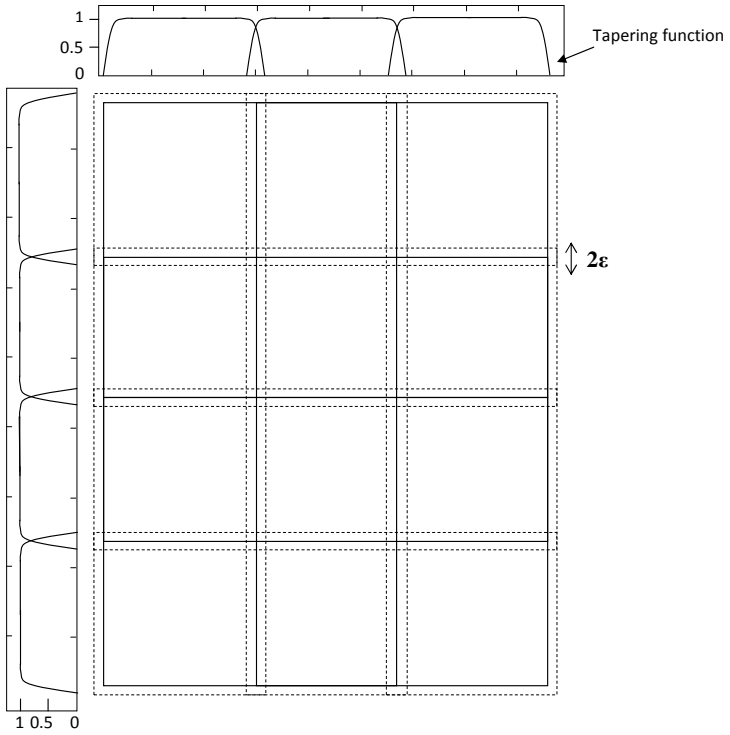


Figure 3.5: Windowing and tapering operators illustrated. Solid lines are windows boundaries. Dotted lines are overlapping regions. The overlapping region is of width 2ϵ.

process each window independently for the required number of iterations and once all windows have been processed, we simply apply the adjoint windowing operator.

The second option, we will call scenario B, is to apply the forward windowing operator with overlaps and tapering just like in scenario A, only this time we allow the overlapping edges of the windows to communicate at each iteration in our process.

In both scenarios we solve a set of nonlinear minimization problems. For instance, instead of solving the denoising problem from the previous chapter (Eqn 2.1) as a single system we solve the following set of problems seperatly:

$$\min \|\mathbf{x}_1\|_1 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x_1} - \mathbf{b_1}\|_2 \leq \sigma \tag{3.6}$$

$$\min \|\mathbf{x}_2\|_1 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x_2} - \mathbf{b_2}\|_2 \leq \sigma \tag{3.7}$$

$$\begin{aligned}
\min \|\mathbf{x}_3\|_1 \quad & \text{subject to} \quad \|\mathbf{A}\mathbf{x_3} - \mathbf{b_3}\|_2 \leq \sigma \\
& \vdots \\
\min \|\mathbf{x}_{n-1}\|_1 \quad & \text{subject to} \quad \|\mathbf{A}\mathbf{x_{n-1}} - \mathbf{b_{n-1}}\|_2 \leq \sigma \\
\min \|\mathbf{x}_n\|_1 \quad & \text{subject to} \quad \|\mathbf{A}\mathbf{x_n} - \mathbf{b_n}\|_2 \leq \sigma
\end{aligned}$$

where n is the number of windows. The vectors $b_1, b_2, \cdots, b_n$ represent our windowed data, where each vector is a window of our n windows. The vectors $x_1, x_2, \cdots, x_n$ represent our denoised outputs. In scenario A, each instance of the above problems is solved without communicating with any other instances, while in scenario B, each instance is solved independently but communicates and updates the edges of neighboring windows.

Each of the above two scenarios have advantages and disadvantages. The first scenario has the advantages of simple implementation and reduced I/O time between nodes. But it only exchanges window edge information at the very end and might affect the accuracy of our estimation during the separation iterations. The second scenario insures that the edges have up-to-date information about the region it is overlapping at each iteration. But this scenario is complex to implement and requires a considerable amount of communication between nodes. Some of this time is spent by each window waiting for all of its edges to be ready before going to the next step in the algorithm. So for a 2D data set, a central window will be waiting for four other windows to update edge information and in 3D, a window will wait for 12 other windows, unlike Scenario A, where all windows work in parallel

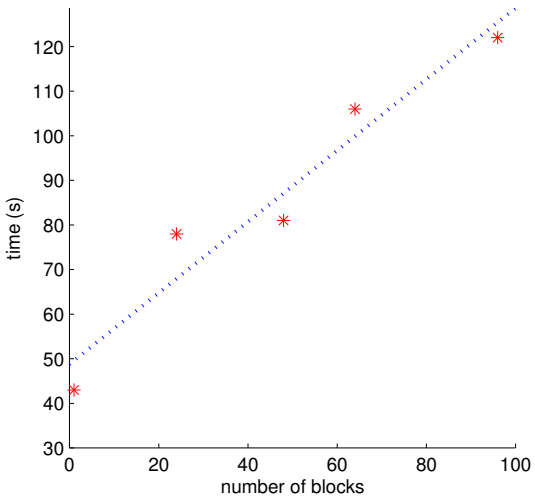
until the end. Another area time is spent on in Scenario B is the time spent to exchange data between windows which involves plenty of I/O operations.

3.3 Scalability

To test the scalability of our approach, we applied the forward curvelet transform on a 3D cube of size 128X128X128 (shots X traces X time samples). We then compared the time it takes to transform this block with the time it takes to apply our parallel curvelet transform on a bigger block windowed into several 128X128X128 blocks. This ensures a fair comparison, since applying the curvelet transform to different block sizes requires different processing time. Our testing blocks contained 24, 48, 64 and 96 blocks of size 128X128X128. For instance the 96 blocks composed a bigger block of size 1536X512X256. Figure 3.6 shows the results of our experiments. The x-axis represent the number of blocks transformed in parallel. The y-axis is the time it takes to apply the forward transform in seconds, including the time it takes to window and taper the blocks. The red stars are the actual times for our experiments. We can see that as we increase the number of blocks the time it takes to perform the transform naturally increases. We can see that it takes 96 blocks about three times the time it takes a single block to be transformed which is a very good ratio.

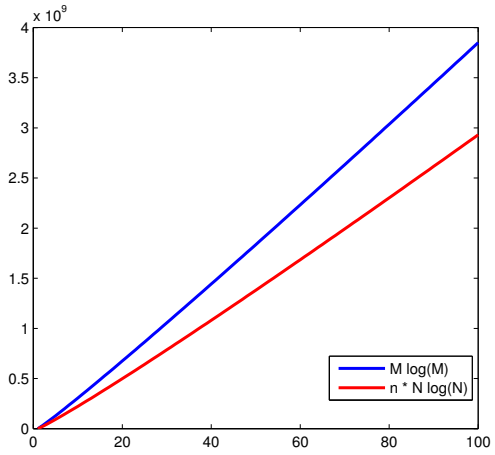
Part of the processing time increase is due to the fact that we have more windows to extract and more windows to apply tapering on. Also, we need more I/O operations and file management. The processing time can be further optimized by optimizing the windowing and tapering implementation, and enhancing I/O and file management techniques.

We also applied a single forward curvelet transform to the largest data set in our experiments (1536X512X256) without parallelization. It took 422 seconds to complete, compared to 123 seconds when using our parallel curvelet transform on the same data set. This gives a perspective on how much speed we can gain with parallelism. Considering the time complexity of $O(M \log M)$ for the curvelet transform, Figure 3.7 Compares the time complexity for applying the forward transform on a single block of data of size M=1536X512X256 with the time complexity of applying the forward transform on 96 blocks of size N=128X128X128.



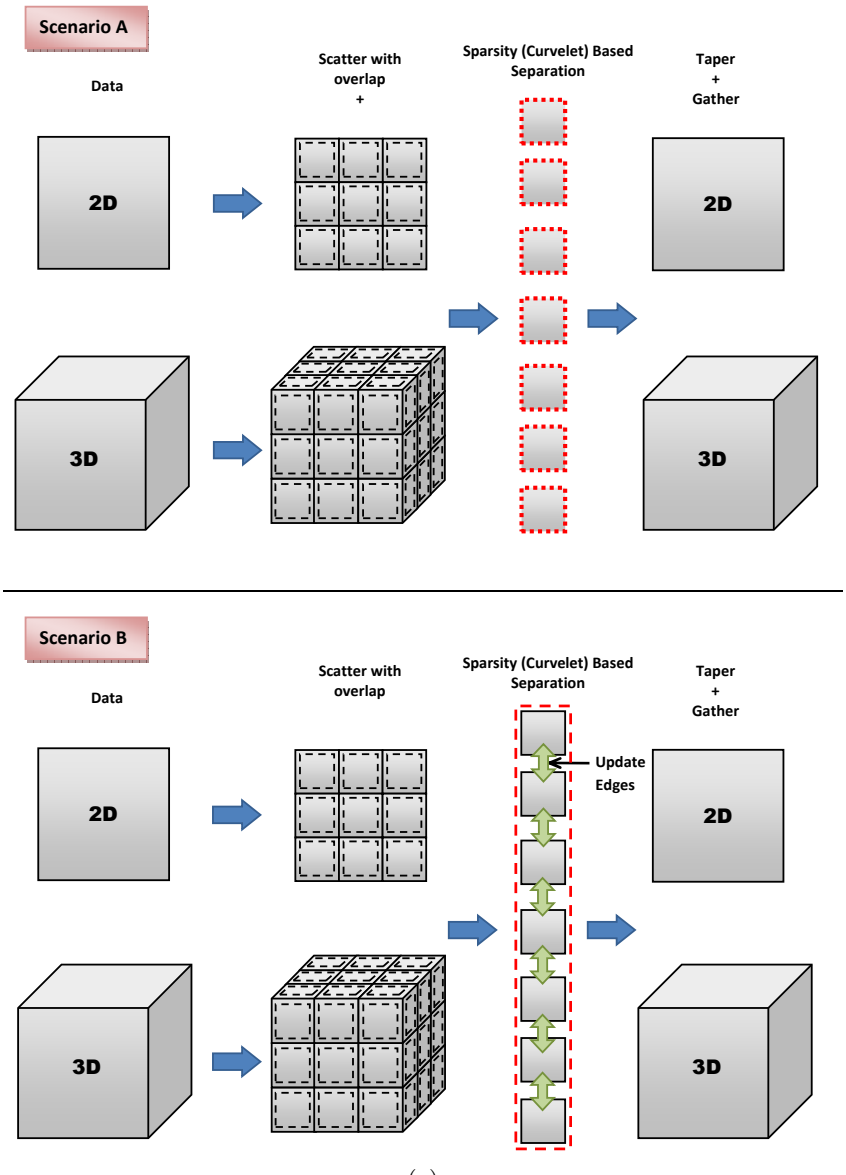
(a)

Figure 3.6: Scalability plot showing timing of a single forward curvelet transform. The x-axis represents the number of blocks transformed in parallel. The y-axis is the time it takes to apply the forward transform including the time it takes to window and taper the blocks. The red stars are the actual times for our experiments.



(a)

Figure 3.7: The Forward curvelet transform complexity. Comparing the time complexity for applying the forward transform on a single block of data of size M (in blue, M=1536X512X256)with the time complexity of applying the forward transform on n blocks of size N (in red, N=128X128X128 and n=96) where N < M.



(a)

Figure 3.8: The two compared scenarios, the dashed red line represent the Bayesian solver. Scenario A) Applying the Bayesian separation at each window separately. Scenario B) Emulating the separation as if the data was not windowed by exchanging edge information between windows.

Bibliography

Mallat, S., 1998, A wavelet tour of signal processing: Acadamic Press.

Thomson, D., G. Hennenfent, H. Modzelewski, and F. J. Herrmann, 2006, A parallel windowed fast discrete curvelet transform applied to seismic processing: SEG Technical Program Expanded Abstracts, 2767–2771, SEG.

Ying, L., L. Demanet, and E. J. Candès, 2005, 3-D discrete curvelet transform: Proceedings SPIE wavelets XI, San Diego, **5914**, 344–354.

Chapter 4

Parallel seismic data processing with the parallel windowed curvelet transform.

In this chapter, we put our windowing method in action, compare the two scenarios defined in the previous chapter and present some findings. First, we tested our windowing technique with a denoising problem and observed our windows edges effect on the solution. Then, we applied our windowing operators in combination with our Bayesian separation method on 2D and 3D data sets. In every experiment, we calculated SNR and compared results.

4.1 Parallel windowed seismic data denoising

We ran multiple experiments using SPGL1 (Berg and Friedlander, 2007) to solve the denoising problem introduced in Chapter 2 using our parallel windowed curvelet transform, in each experiment, we change a set of parameters, namely, the sparsifying operators, window sizes and overlap sizes. We then calculated the SNR against the noise free data according to :

$$SNR = 20 \log \frac{\| \frac{\mathbf{d}}{\|\mathbf{d}\|_2} \|_2}{\| \frac{\mathbf{d}_n}{\|\mathbf{d}_n\|_2} - \frac{\mathbf{d}}{\|\mathbf{d}\|_2} \|_2}, \tag{4.1}$$

where $\mathbf{d}$ denotes our noise free data and $\mathbf{d}_n$ is our denoised solution. Table 4.1 shows the SNR values from our experiments. The first column specifies which scenario was used. Scenario A: solving each window independently with no edge information update during separation. Scenario B: solving each window independently with edge information updates occurring at each iteration. Each pair of rows share the same parameters first for Scenario A and second for scenario B. The second column shows the

number of windows in each dimension. So for example 2X4 is interpreted as dividing the data into 2 windows along the time axis and 4 windows along the receiver axis. We can see that we have three groups based on overlap values of 6,10 and 16.

Figure 4.1 shows the noise free data and the data after adding noise on the top. On the bottom, we can see the two results from scenario A on the left and scenario B on the right. Both were windowed into 16 equal windows.

Looking at the SNR values in Table 4.1, we notice several points. First, we notice that in each of the scenarios the different window sizes result in different SNR values with considerable variations. For example, running scenario A with overlap value $\epsilon = 16$ resulted in SNR values that range between 6.89 and 8.11. Another example is the SNR values resulting from scenario B with $\epsilon = 16$ that range from 7.26 to 10.34. These variations in SNR are due to the fact that each time, we are solving a different non-linear problem and when we change the windows' sizes, we also change the location where we divide the data, and hence, the likelihood that significant coefficients are located near the edges of the windows or in the overlapping region. Comparing SNR values from scenario A and B, we notice that scenario B results in higher SNR values compared to corresponding values from scenario A. This is due to the overlap in our curvelet transform which minimizes the effect of dividing the data. The improvement in SNR reached a difference of up to 2.23 dB, which is a considerable amount of improvement. Some results showed more improvement than others, because the combinations of window sizes and overlaps affects the data differently in terms of the amount of significant curvelet coefficients near the borders. But in general, scenario B produced higher SNR values throughout our experiments.

4.2 Parallel windowed primary-multiple separation

Our set of primary-multiple separation experiments were conducted on synthetic data that contained 361 shots, 361 traces/shot, 501 time samples/-trace with sample intervals $\Delta t = 4ms$.

We started by applying our code on shot record number 181 for our 2D experiment. Then, applied the code on the 3D dataset using the 3D curvelet transform (Ying et al., 2005). Our input dataset consisted of predicted (SRME) primaries $\mathbf{s}_1$ and predicted multiples $\mathbf{s}_2 = \mathbf{d} - \mathbf{s}_1$, where $\mathbf{d}$ is our total data, and "multiple free" data $\mathbf{s}_p$, which is generated using the same simulation of the total data, only an energy absorbing boundary condition

Scenario	Window	SNR
$\epsilon = 6$		
A	1X2	7.30
B	1X2	7.70
A	2X2	7.39
B	2X2	7.90
A	1X4	7.80
B	1X4	8.16
A	4X4	8.06
B	4X4	8.27
$\epsilon = 10$		
A	1X2	7.90
B	1X2	9.35
A	2X2	7.73
B	2X2	7.91
A	1X4	7.04
B	1X4	8.05
A	4X4	7.94
B	4X4	8.12
$\epsilon = 16$		
A	1X2	6.89
B	1X2	7.26
A	2X2	6.97
B	2X2	7.38
A	1X4	8.11
B	1X4	10.34
A	4X4	7.02
B	4X4	7.50

Table 4.1: Calculated SNR for the denoise problem. The first column specifies which scenario was used. Scenario A: no edge information update during separation. Scenario B: edge information update occurs at each transform call. each pair of rows share the same parameters used in scenario A and B. The window column shows the number of windows at each dimension, these windows have identical sizes.

is enforced to prevent the generation of multiples. The "multiple free" data were used as a reference to calculate SNR according to Eqn 4.1.

Figure 4.2 shows the total data, predicted multiples, predicted SRME primaries and estimated primaries after applying the Bayesian separation.

4.2.1 2D data experiments

Table 4.3 shows SNR calculated against the true primaries with different parameters. The first column specifies which scenario was used. The second column shows the number of windows in each dimension. We can see that we have three groups based on overlap values of 10,15 and 20. For comparison it is important to mention that the SNR value of the SRME predicted primaries is 9.82, and the SNR of the estimated primaries after applying the Bayesian separation without any windowing is 11.49.

Looking at Table 4.3 we notice the following: First, we notice that all SNR values in the table are extremely close and unlike the case with the denoising problem the changes in window sizes for each of the scenarios does not have considerable variations in the output SNR values. The differences between the lowest and highest SNR values in scenario A and scenario B are 0.07 and 0.079 respectively. This is a major advantage of our Bayesian separation technique where we threshold the primaries against a fixed threshold based on the multiples, and threshold the multiples with a fixed threshold based on the primaries. In both cases, the primaries and multiples are windowed and tapered. Another reason that explains these results, is the fact that in the denoising problem, the solver uses a cooling method where it starts with a very sparse solution, this sparseness is translated into a small number of curvelets, if some of this small number is located near windows edges, they will have distingushable presense and create artifacts. But the Bayesian separation works with a large number of coefficients such that they contribute in reducing the possibility of having artifacts near the window edges because we have enough curvelets to diminish the effect of wrpping around the window edges.

We also notice that the more windows we have the lower SNR values we get, one reason for this is the fact that we increase the discontinuities introduced in the data and hence the probability of a curvelet being located near the borders. Also, notice that as we increase the overlap, we get higher SNR. Because large overlaps allow windows to have more information and be tapered more smoothly than smaller overlaps.

Our goal is to compare the two scenarios we have. And looking at the SNR values we can see that, as expected, in general Scenario B, which

updates windows' edges, has higher values. But the difference between the two scenarios is in the hundredth decimal place, with a maximum of 0.0276, which is considered small.

Figure 4.4 shows one of our worst cases where we divide the data into 16 equal windows. Figure 4.4b shows the estimated primaries from our Bayesian method after windowing into 16 windows without applying any tapering or overlap, we can clearly see introduced artifacts along the windows' edges. Figure 4.4c shows the estimated primaries after applying Scenario A. We can clearly see that the artifacts are gone now and the data looks cleaner. Same applies to Figure 4.4d, which shows the estimated primaries after applying Scenario B. Comparing Figure 4.4c and Figure 4.4d we can see that both results are extremely close and differentiating between them is difficult. This is consistant with the SNR results in Table 4.3. Looking at different scenarios we get the same observation, i.e. Scenario A performs very well with very close output to the one from Scenario B. Figuers 4.5 and 4.6 also show the same outcome using different windows' sizes. This means that in terms of SNR and image quality, we can use Scenario A and save a considerable amount of processing time that is consumed as I/O time between processing nodes.

Our experiments with 2D data show that Scenario A produces excellent results compared to the more complex but relatively more accurate Scenario B in the context of our Bayesian separation technique. We further investigate the two scenarios using 3D data and the 3D curvelet transform.

Scenario	Window	Overlap ϵ	SNR	Difference
A	2X2	10	**11.41**	
B	2X2	10	**11.43**	0.02
A	2X3	10	11.46	
B	2X3	10	11.48	0.02
A	3X2	10	11.37	
B	3X2	10	11.38	0.01
A	4X4	10	11.39	
B	4X4	10	11.41	0.02
A	1X4	10	11.40	
B	1X4	10	11.41	0.01
A	4X1	10	11.38	
B	4X1	10	11.39	0.01
A	2X2	15	**11.44**	
B	2X2	15	**11.46**	0.02
A	2X3	15	11.43	
B	2X3	15	11.46	0.03
A	3X2	15	11.39	
B	3X2	15	11.40	0.01
A	4X4	15	11.41	
B	4X4	15	11.43	0.02
A	1X4	15	11.42	
B	1X4	15	11.43	0.01
A	4X1	15	11.40	
B	4X1	15	11.41	0.01
A	2X2	20	11.44	
B	2X2	20	11.46	0.02
A	4X4	20	11.41	
B	4X4	20	11.42	0.01

Table 4.2: Calculated SNR for the 2D synthetic data set. The first column specifies which scenario was used. Scenario A: no edge information update during separation. Scenario B: edge information update occurs at each transform call. each pair of rows share the same parameters used in scenario A and B. The difference column carry the difference between the SNRs of scenario A and B for each pair. The window column shows the number of windows at each dimension, these windows have identical sizes. Notice that the more windows we have the more discontinuities in the data which lowers the SNR. Also, notice that increasing the overlap amount improves the SNR. All results were achieved after 10 iterations of the solver.

4.2.2 3D data experiments

The scalability issue we are attempting to solve is more crucial with 3D data since it requires more memory. We applied the Bayesian separation on our synthetic 3D data and calculated the SNR for each experiment the same way we did for 2D data. The SNR value of the predicted SRME primaries is 9.928 and the SNR after applying the Bayesian separation without windowing is 11.466.

Table 4.3 shows calculated SNR values from our experiments. Just like in the 2D case we can see that the variations between SNR values are extremely small. Scenario B showed relatively higher SNR values than scenario A, yet the difference is negligible.

Figure 4.9 shows a slice taken from the 3D cube at shot point 181. Figure 4.9a shows the estimated primaries after applying Bayesian separation without any windowing. Figure 4.9b shows the estimated primaries after windowing, with 4X4X2 windows, without any overlapping nor tapering. Notice the large amount of introduced artifacts, especially around the windows' edges. Figuers 4.9c and 4.9d show the estimated primaries using scenarios A and B, respectively. Notice that there are almost no artifacts around windows' edges. Note that this is one of the worst cases in Table 4.3.

Scenario	Window	Overlap ϵ	SNR	Difference
A	2X2X2	15	11.50	
B	2X2X2	15	11.51	0.01
A	2X4X4	15	11.50	
B	2X4X4	15	11.51	0.01
A	4X4X2	15	11.46	
B	4X4X2	15	11.47	0.01

Table 4.3: Calculated SNR for the 3D synthetic data set. The first column specifies which scenario was used. Scenario A: no edge information update during separation. Scenario B: edge information update occurs at each transform call. each pair of rows share the same parameters used in scenario A and B. The difference column carry the difference between the SNRs of scenario A and B for each pair. The window column shows the number of windows at each dimension, these windows have identical sizes. Notice that the more windows we have the more discontinuities in the data which lowers the SNR.

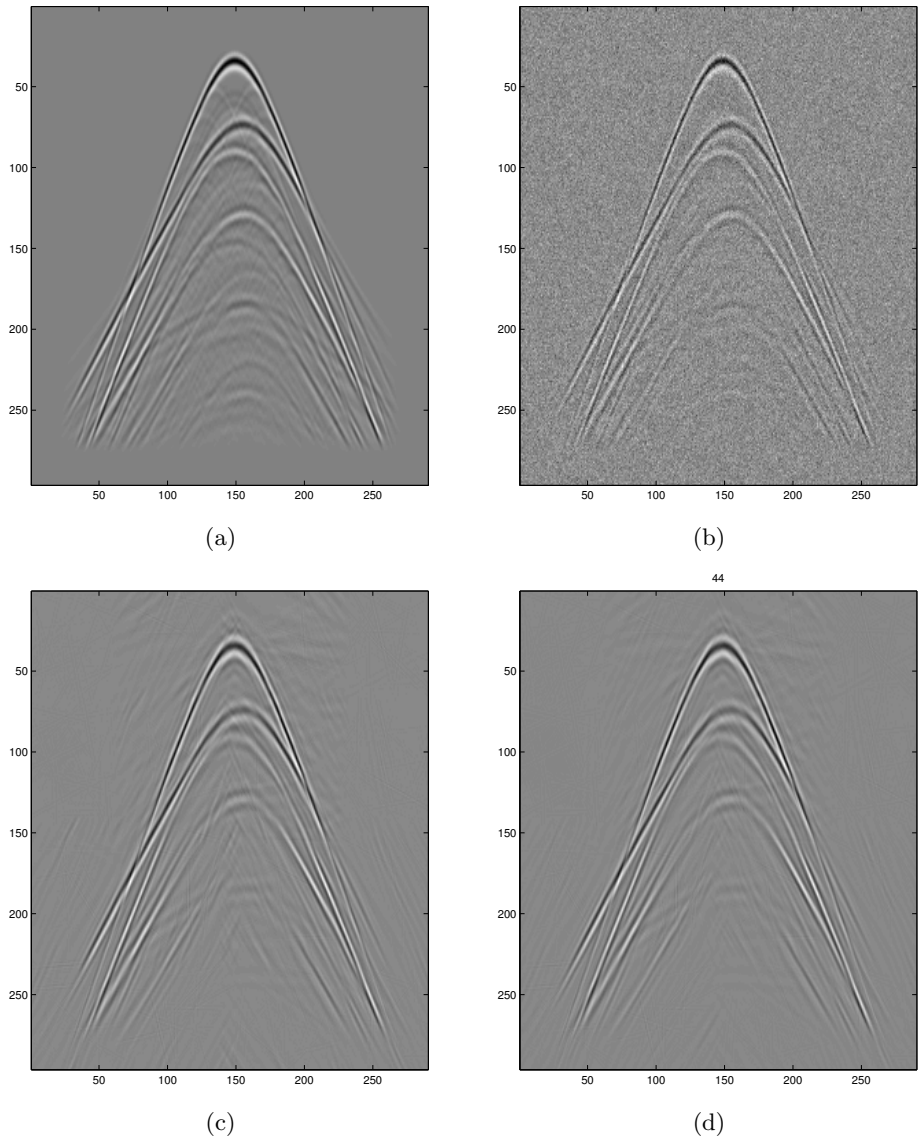


Figure 4.1: a) Noise free data b) same data from a with added Gaussian noise with $\sigma = 1.4$ c) Denoised data using Scenario A d) Denoised data using Scenario B

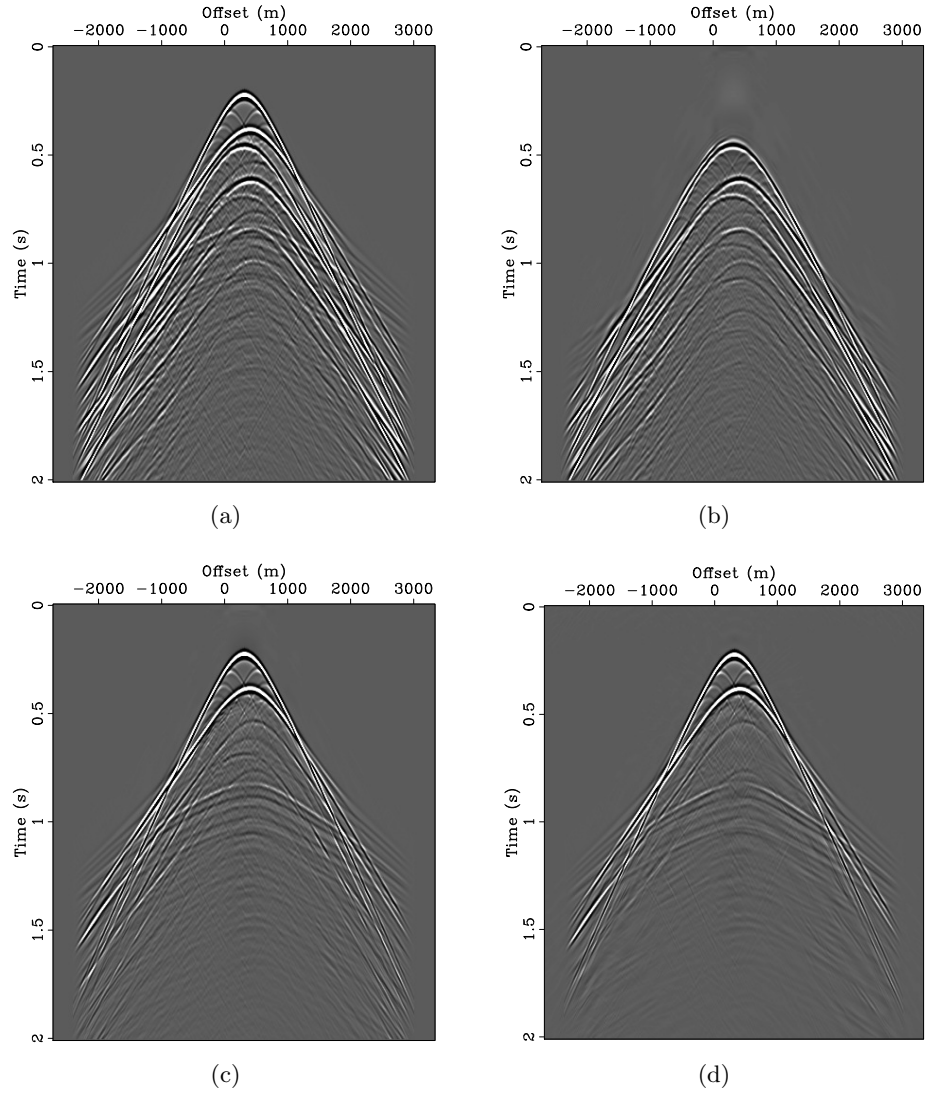


Figure 4.2: 2D synthetic data a) total data b) SRME predicted multiples c) SRME predicted primaries d) estimated primaries using Bayesian separation

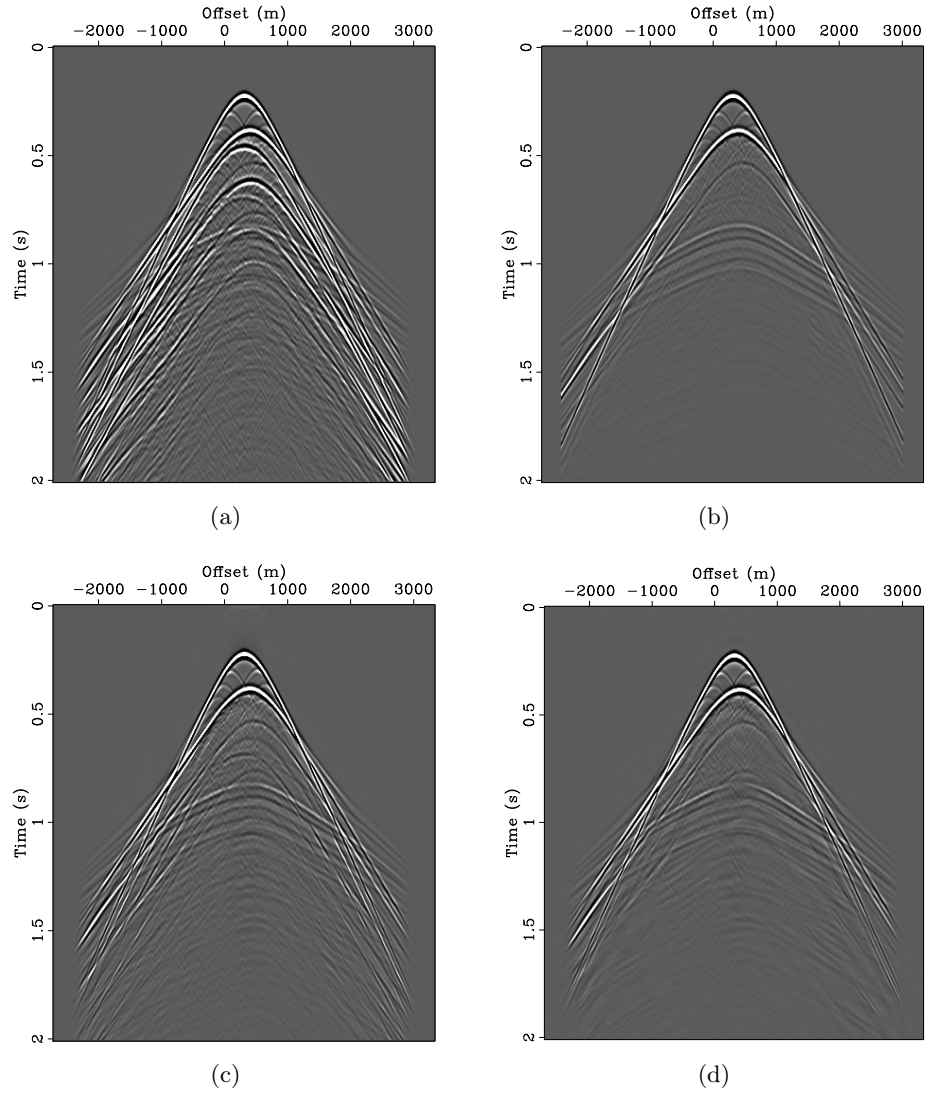


Figure 4.3: 2D synthetic data a) total data b) true 'multiple-free' primaries c) SRME predicted primaries d) estimated primaries using Bayesian separation.

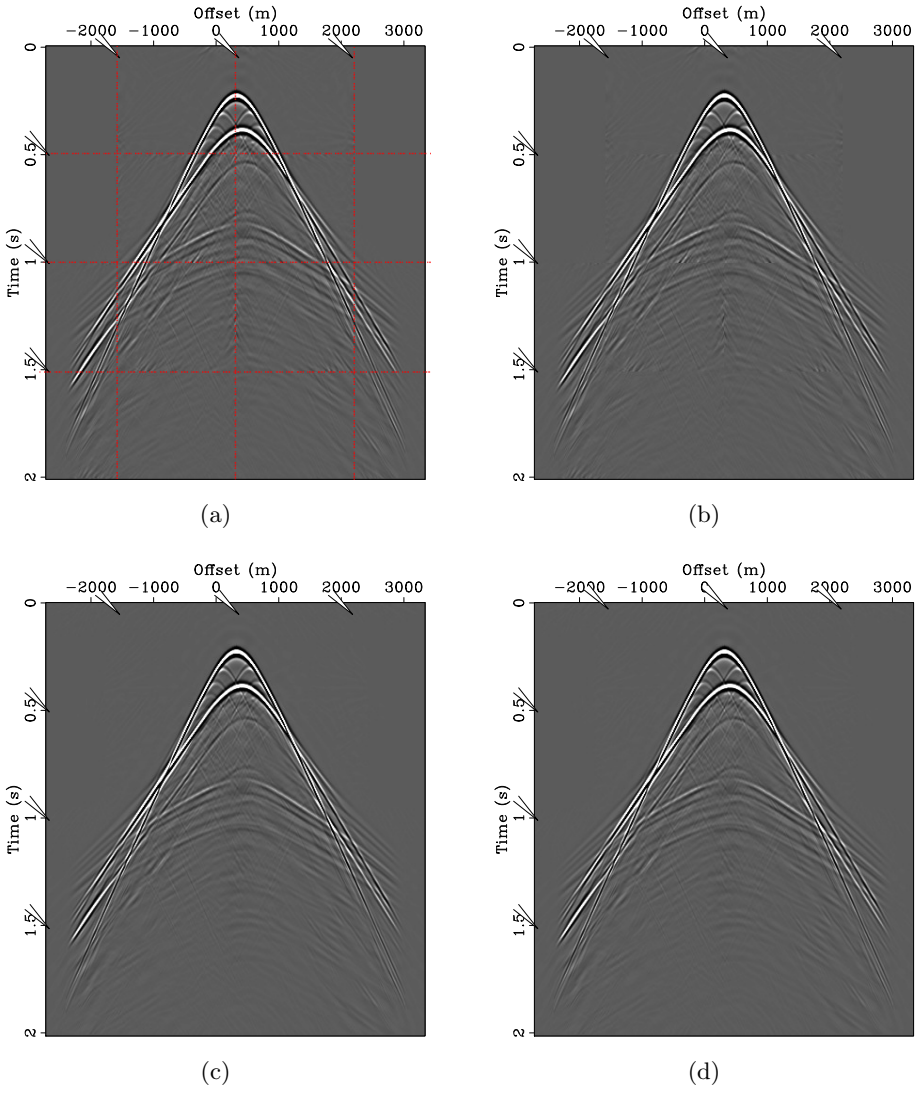


Figure 4.4: Estimated primaries a) illustration of how data will be divided, red lines are window edges. b) applying the Bayesian separation without any overlapping, tapering or edge updates. Notice the introduced artifacts along the edges indicated by the pointer. c) Scenario A (no edge updates) with overlap $\epsilon = 15$ d) Scenario B (edge updates) with overlap $\epsilon = 15$

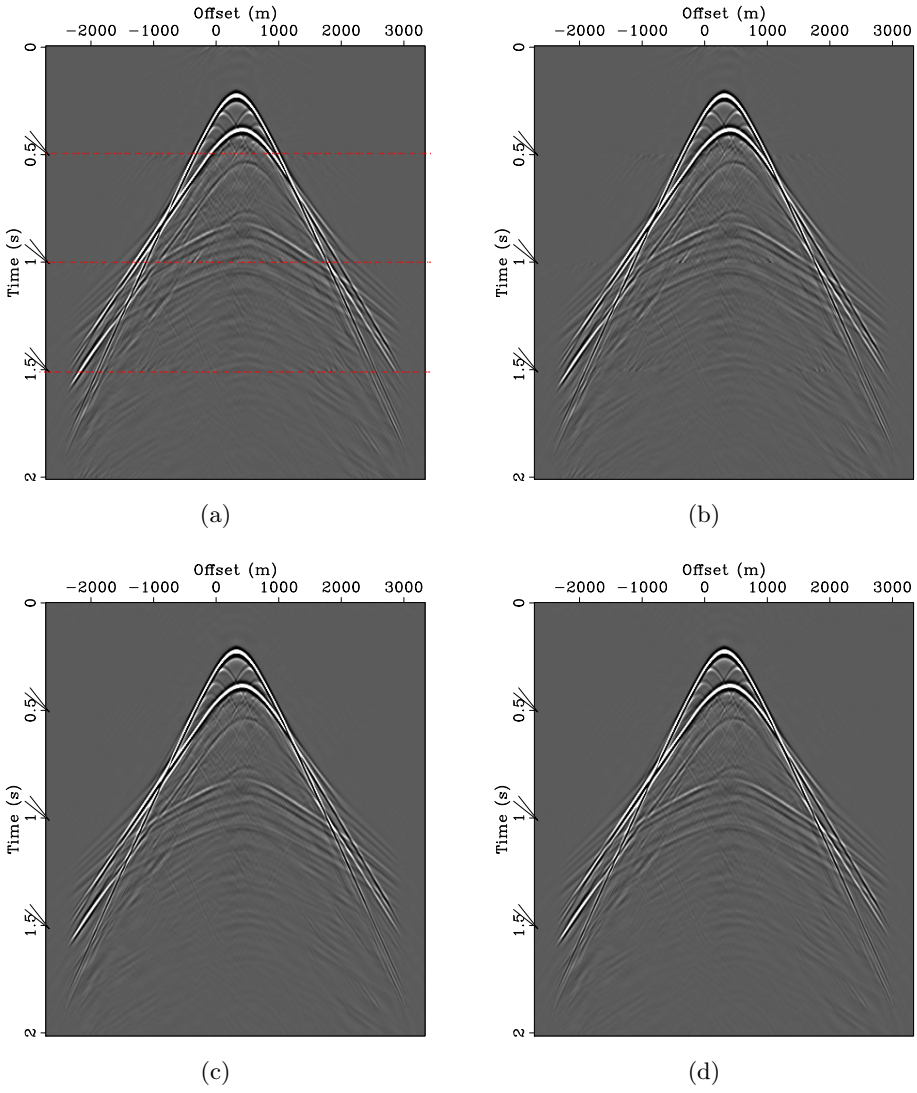


Figure 4.5: Estimated primaries a) illustration of how data will be divided, red lines are window edges. b) applying the Bayesian separation without any overlapping or tapering. Notice the introduced artifacts along the edges indicated by the pointer. c) Scenario A (no edge updates) with overlap $\epsilon = 15$ d) Scenario B (edge updates) with overlap $\epsilon = 15$

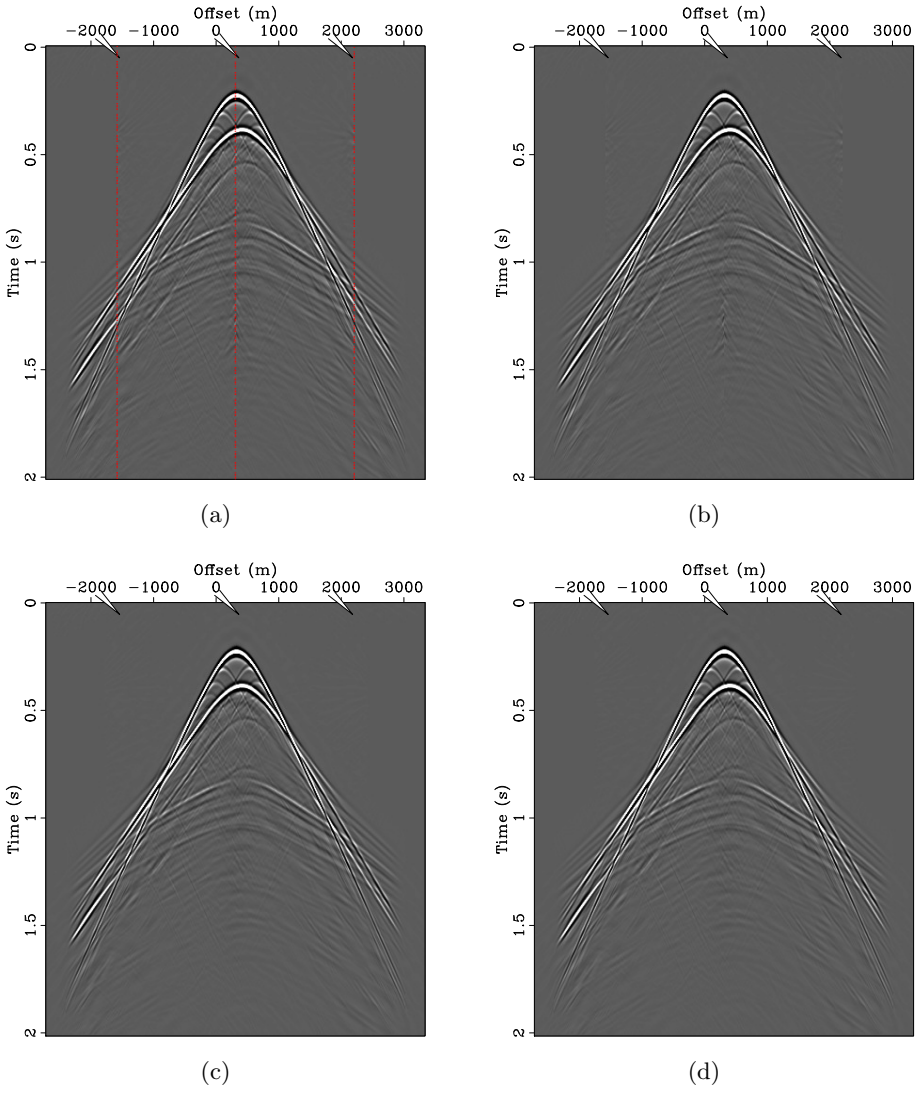
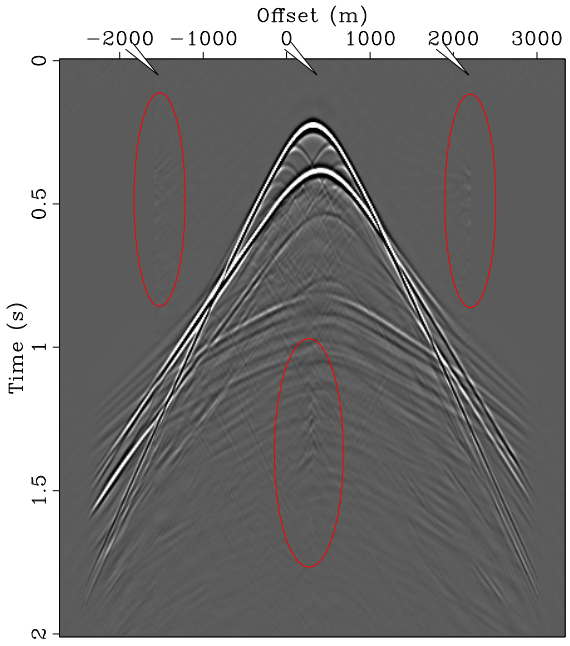
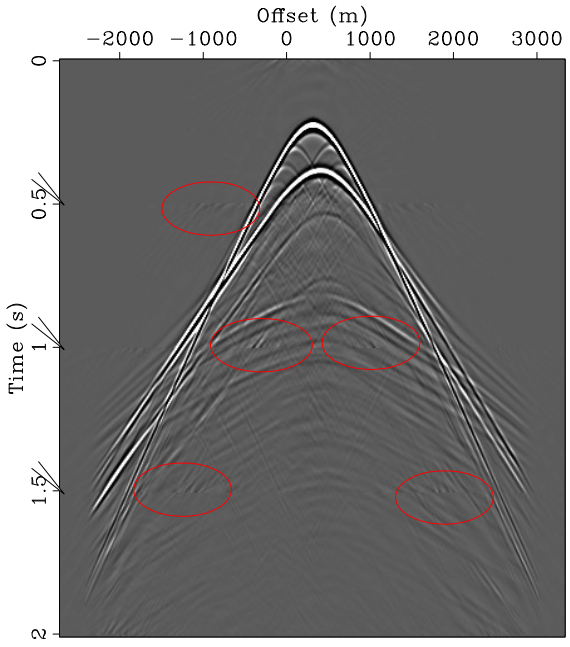


Figure 4.6: Estimated primaries a) illustration of how data will be divided, red lines are window edges. b) applying the Bayesian separation without any overlapping or tapering. Notice the introduced artifacts along the edges indicated by the pointer. c) Scenario A (no edge updates) with overlap $\epsilon = 15$ d) Scenario B (edge updates) with overlap $\epsilon = 15$

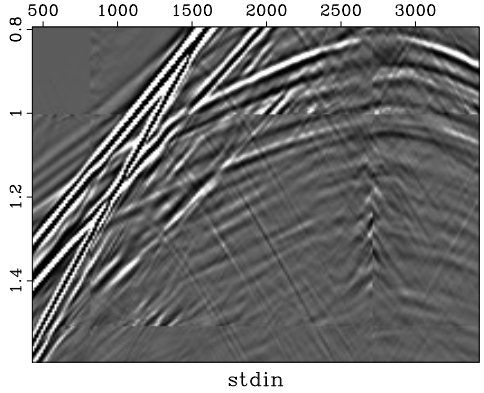


(a)

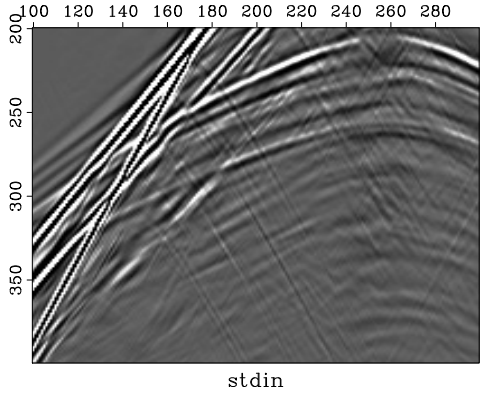


(b)

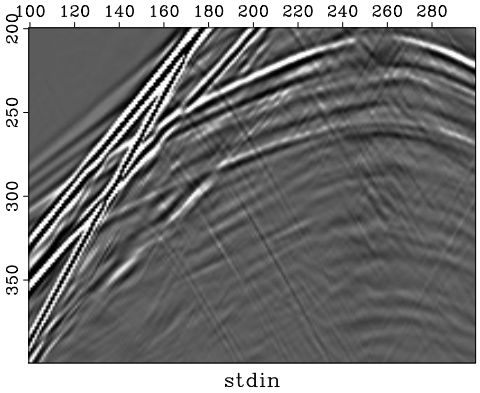
Figure 4.7: Wide vs tall windows. a) tall windows without overlap and tapering b) wide windows without overlap and tapering. The red ovals highlight some artifacts.



(a)



(b)



(c)

Figure 4.8: A closer look of the estimated primaries generated using 16 windows and a) no overlap nor tapering b) applying Scenario A c) applying Scenario B. The images were clipped to 0.8 to make comparision easier.

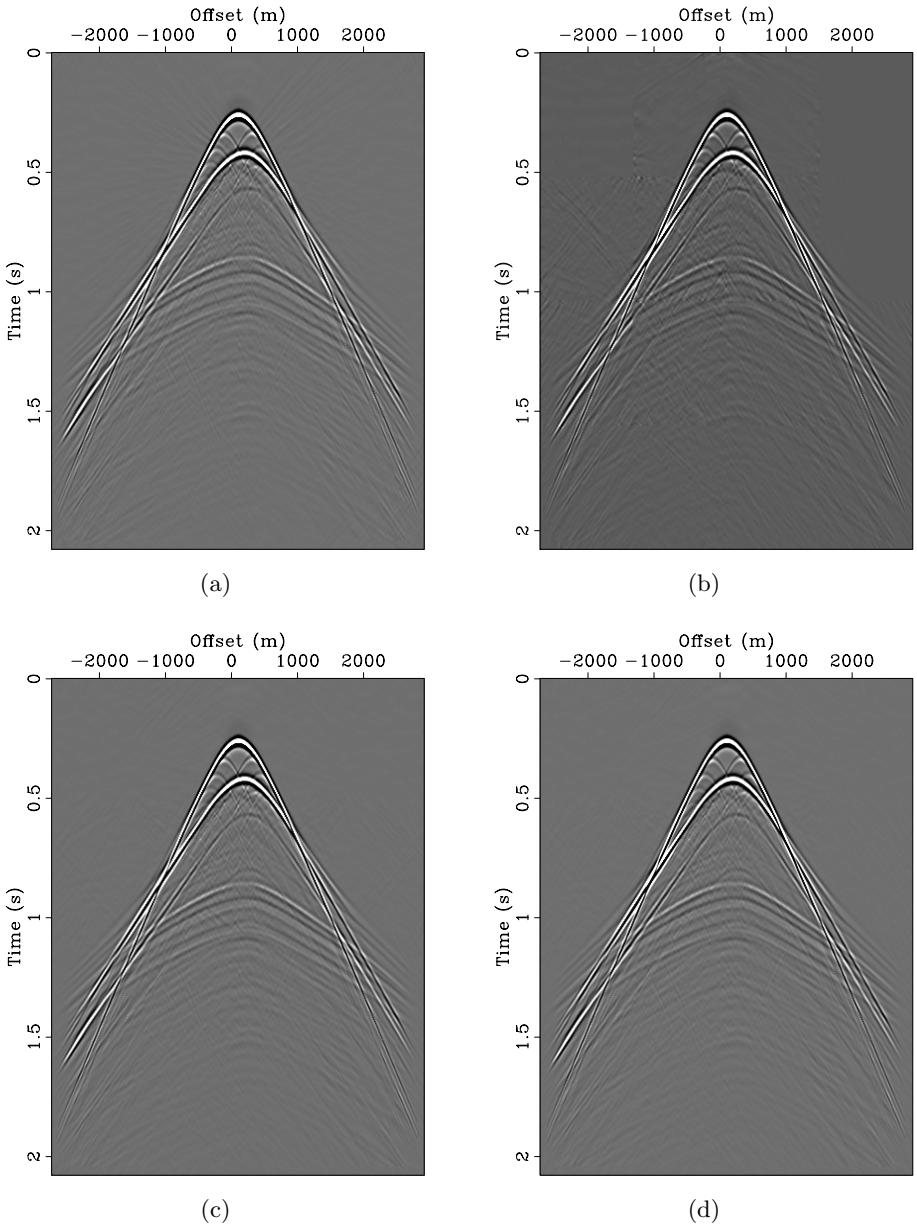


Figure 4.9: A slice from our synthetic 3D data cube. a) Estimated primaries using Bayesian separation without windowing. b) Estimated primaries after windowing into 4X4X2 windows without overlaps or tapering. c) Estimated primaries using Scenario A (no edge update) d) Estimated primaries using Scenario B (window edges exchange information)

Bibliography

Berg, E. v. and M. P. Friedlander, 2007, SPGL1: A solver for large-scale sparse reconstruction. (http://www.cs.ubc.ca/labs/scl/spgl1).

Ying, L., L. Demanet, and E. J. Candès, 2005, 3-D discrete curvelet transform: Proceedings SPIE wavelets XI, San Diego, **5914**, 344–354.

Chapter 5

Conclusion

5.1 Parallel curvelet domain seismic data processing

Our goal in this thesis is to solve the problem of scalability of the curvelet transform in iterative seismic processing techniques such as the combination of SRME (Verschuur et al., 1992) and Bayesian separation used in (Verschuur, 2006; Wang et al., 2008). We have introduced a windowing technique that allows us to divide large seismic data into smaller data sets that make it possible to fit the huge data and redundant curvelet coefficients into memory. Since we divide the data, we introduce discontinuities at the windows' edges, hence, we need to address the issue of curvelet coefficients located at the boundaries of these windows. We applied a window tapering function to our windowing operator. We designed the windows to overlap such that the sum of the tapered overlapping regions preserve the system's energy. Once we had a windowing operator we compared two scenarios of applying it on our denoising and multiple elimination problems. The first scenario divides the data with our windowing operator and then process each window independently. Finally, when each window is processed, the adjoint windowing is applied and the final data set is gathered. The second scenario redefines the sparsifying operator such that it allows the windows to be independently processed but update their overlapping edges.

The advantages of using the first scenario are simplicity in implementation and speed. There is minimum I/O time between the windows. The main advantage of the second scenario is accuracy. But this scenario spends a considerable amount of time in I/O. Not only does the edge information exchange consume time, but each window has to wait for all neighboring windows to finish before updating.

From our experiments, the second scenario showed a considerable advantage when solving the denoising problem. Our Bayesian separation showed extremely small differences between the two scenarios and showed the advantage of allowing us to use the faster and simpler windowing technique

without losing considerable degree of accuracy.

5.2 Open and future research

The use of curvelets (Candès et al., 2006) as a sparsifying domain in seismic imaging has opened great opportunities in geophysical applications such as seismic image regularization, ground roll removal and primary multiple separation (Yarham et al., 2007; Herrmann, 2008; Yan, 2008; Hennenfent and Herrmann, 2005, 2006).

Our application of curvelets on primary-multiple separation is based on general sparsity concepts. This gives the chance to experiment with different sparsifying domains that might come up in the future. Another possible research area would be to develop an automated way to choose parameters in our algorithms, such as curvelet parameters (e.g. scale, angle) and windowing parameters such as window dimensions and the amount of overlap. Also, the protocols, by which our method manages data and I/O can be considered as a good area for improvement.

Bibliography

Candès, E. J., L. Demanet, D. L. Donoho, and L. Ying, 2006, Fast discrete curvelet transforms: Multiscale Modeling and Simulation, **5**, no. 3, 861–899.

Hennenfent, G. and F. J. Herrmann, 2005, Sparseness-constrained data continuation with frames: applications to missing traces and aliased signals in 2/3-D: SEG Technical Program Expanded Abstracts, 2162–2165, SEG.

———, 2006, Application of stable signal recovery to seismic data interpolation: SEG Technical Program Expanded Abstracts, 2797–2801, SEG.

Herrmann, F. J., 2008, Adaptive curvelet-domain primary-multiple separation: Presented at the SINBAD 2008. (SINBAD 2008).

Verschuur, D. J., 2006, Seismic multiple removal techniques: past, present and future: EAGE publications b.v. ed.

Verschuur, D. J., A. J. Berkhout, and C. P. A. Wapenaar, 1992, Adaptive surface-related multiple elimination: Geophysics, **57**.

Wang, D., R. Saab, O. Yilmaz, and F. J. Herrmann, 2008, Bayesian-signal separation by sparsity promotion: application to primary-multiple separation: Technical Report TR-2008-1, UBC Earth and Ocean Sciences Department.

Yan, J., 2008, Wavefield reconstruction using simultaneous denoising interpolation vs. denoising after interpolation: Presented at the SINBAD 2008.

Yarham, C., G. Hennenfent, and F. J. Herrmann, 2007, Curvelet applications in surface wave removal: Presented at the EAGE Technical Program Expanded Abstracts, EAGE.