A NOVEL APPROACH TO BLIND SOURCE SEPARATION AND EXTRACTION IN AUDIO

By

Xun Wang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Applied Mathematics — Doctor of Philosophy

2015

ABSTRACT

A NOVEL APPROACH TO BLIND SOURCE SEPARATION AND EXTRACTION IN AUDIO

By

Xun Wang

In this thesis, the blind source separation (BSS) on digital audio signals via background learning by several different methodologies is carefully studied. In the daily auditory, acoustic audio signals are usually mixtures of different sources, including foreground and background noises.

Most of the time, we only want to receive the foreground sources and get rid of the background ones. Because of the randomness of various situations, it is very difficult to perform this separation without knowing the detailed information. Even if the background noises are not dominating the foreground sources, or even much weaker, it is still a difficult problem, especially for the case that there are more than three sources including the noise. This also makes it even more difficult to separate different sources from mixed signals. In this thesis, a novel approach to solve cancellation kernels is provided by using a modified sigular value decomposition method. The main focus is to use this new technique to estimate the cacellation kernels with good results in short computational time.

In this work, some background information for blind source separation of audio will be first introduced. Next, the knowledge of four different methods for solving this type of problems is mentioned. Split Bregman has been studied by others in solving cancellation kernels for the separation of speech signals. We apply proximity operator method to solve the cancellation kernels for BSS of audio signal processing. It has been applied to study image processing by other researchers. Quadratic programming method has been applied to solve

cancellation kernels by Wang and Zhou. We provide a new approach to bring sparseness to cancellation kernels by using quadratic programming.

We developed a modified singular value decomposition (SVD) algorithm based on the numerical experiments. It is a new technique to estimate cancellation kernels for BSS of audio signals. The detailed information and schemes are presented in Chapter 3. Then, in the fouth chapter, there are different numerical simulation examples according to different scenarios. We compare the results of our modified SVD method with others methods, and conclude that our modified SVD is the best approach.

To Sihua and my parents

ACKNOWLEDGMENTS

I owe my great gratitude to my two advisors, Dr. Yang Wang and Dr. Zhengfang Zhou, without whom this thesis would not have been possible. They introduced me to the field of mathematics and supported my work in times of challenge and difficulty. They are not just my academic advisors, but also life mentors who make me the person I am today.

I would also love to express my gratitude to my other three committee members. Dr. Andrew Christlieb, Dr. Mark Iwen, and Dr. Tien-Yien Li. They have shared their great wisdom with me and taken time to give me feedback on this thesis. I sincerely appreciate the help from Dr. Peiru Wu. There are many faculty and staff in this great department that I am grateful to, for they provide me rapport and encouragement throughout my years here at Michigan State.

I would also like to thank my family for their unquestionable love and support. Thank you for believing in me all the time, and thank you for not giving me pressure when I was at my low point.

Last but not the least, so many thanks to my dear girlfriend Sihua Hu who inspires me to finish my thesis.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

The acoustic signal processing system has been studied by researchers for several decades. Within the research community, people have achieved many remarkable results, especially in the past decade. During this process, one of the major goals is to get the estimation of one or more source signals as accurate as possible, or at least to the level that people could tolerate the noise. Especially nowadays, the high performance computing technology can rapidly implement a lot of previously time-consuming methods.

For audio recording mixtures, it is very difficult to separate different source signals from mixtures. There are many causes for noise, such as audio recording process, background noise from other sources, or audio propagation, and so on. These random origins make it difficult to conduct noise removal. Blind source separation is the separation of a set of signals from the set of mixed signals, without the aid of information (or with very little information) about the source signals or the mixing process. There are many applications of blind source separation to modern technology, such as biomedical, telecommunications, astrophysics and image processing, etc.

1.1 Blind Source Separation

Blind source separation (BSS) is a major area of research in signal processing. A vast literature exists, particularly in audio signal processing. The goal of BSS is to separate source

signals from their mixtures without assuming detailed knowledge about the sources and the mixing process. In this paper we focus on a particular application of BSS in audio, namely to extract a desired audio source from mixtures involving noise, background or unwanted sources. Such extractions can be extremely challenging when the unwanted sources are much stronger than the source one wishes to extract, which can be the case in many practical applications such as mobile and car phones. Standard BSS techniques often do not work at all in such settings.

We first describe our model in its most general form. The basic setup for BSS in audio has n audio sources $S_1(t), \ldots, S_n(t)$ and m mixtures $X_1(t), \ldots, X_m(t)$, with the model

$$X_k(t) = \sum_{i=1}^{L} \sum_{j=1}^{N_{k,i}} a_{k,i,j} S_i(t - d_{k,i,j}), \qquad k = 1, \ldots, M \tag{1.1.1}$$

where $a_{k,i,j}$ are the mixing coefficients and $d_{k,i,j}$ are the delays from the source i to the recording or sensing device k. The mixing with multiple delays are a result of both the direct signal and the reverberations (echoes) in the ambient environment. In audio applications the sensors are microphones, and these mixtures are recorded by placing m microphones in various locations. In BSS these coefficients are unknown. With the presence of reverberations we often refer to the mixtures X_k in (1.1.1) as *convolutive mixtures*. Mathematically a more concise way to formulate the model (1.1.1) is

$$X_k = \sum_{i=1}^{L} A_{k,i} * S_i, \qquad k = 1, \ldots, M \tag{1.1.2}$$

where $*$ denotes convolution and

$$A_{k,i}(t) = \sum_{j=1}^{N_{k,i}} a_{k,i,j}\delta(t - d_{k,i,j}) \tag{1.1.3}$$

are the *convolutive kernels* or *filters* that represent the mixing of both the direct signals and their reverberations. BSS techniques aim to estimate these convolutive kernels $A_{k,i}$.

Among the most popular techniques for BSS is the *independent component analysis (ICA)* model, where the source signals S_k, modeled as random variables, are assumed to be independent, see the reviews [13, 26] and the references therein for more details. Under this model, many techniques such as Joint Approximate Diagonalization Eigenmatrices (JADE) [11] and Information Maximization (Infomax) [6, 12], as well as their refinements, have been developed. It is safe to say that the ICA model forms the overwhelming bulk of the research in BSS. An alternative technique, which is gaining popularity, is the *Degenerative Unmixing and Estimation Technique (DUET)*, which uses time-frequency separations to achieve BSS and has the advantage that it can work in the degenerative case $M < L$, see e.g. [30, 60]. All these techniques have their respective strengths and weaknesses, which are well documented in the literature so we will not go into details here. While these techniques can often yield good results in BSS, they all have certain limitations that pose challenges for many source extraction and background removal applications. A major challenge is that they do not work well for the case we focus on, namely for source extraction where the desired source signals are very weak in comparison to the unwanted sources in the mixtures. Computational cost can be another issue for many practical applications such as mobile and car phone where removing background sources must be done in real time. The batch processing techniques used by many of the ICA techniques do not adapt well to dynamic real-time processing.

An alternative approach based on background learning and cancellation kernels for source extraction was first proposed by Wang and Zhou [61]. The idea is that if the desired source we wish to extract (we call it the *target sources*) is inactive (silent) for a brief moment, then one may during this brief moment "learn" the ambient environment in the sense of attaining information about the convolutive kernels for the unwanted sources (we call them the *background sources*). This will allow the extraction of the target source through cancellation of the background sources. Typically one second or even less of silent moment by the target source is sufficient for the extraction. In [61] background learning was achieved through quadratic programming and regularization by imposing certain nonnegativity condition which we shall describe in detail later. However, there are issues that needed to be addressed. A major issue was the computational speed, as quadratic programming can be quite slow. Another main issue is that although the imposition of nonnegativity regularization seems to yield excellent result, it is physically unrealistic as in real world mixtures the convolutive kernels will often have negative coefficients. A modified version of this model was proposed in Yu et al [63], in which an L^1 regularization model was used in place of the nonnegativity. Not only it would allow negative coefficients in the kernel, but now a split Bregman algorithm to compute the kernels for the extraction. Computationally, the use of the split Bregman algorithm is substantially more efficient than the quadratic programming approach in [61].

Still, even with the much improved computational efficiency there remain some issues that need to be addressed or understood. The L^1 minimization model often leads to sparsity. But again, just like the nonnegativity assumption is physically unrealistic, so is the sparsity assumption, as in a typical echoic environment the convolutive kernels are frequently *not sparse*. The quality of the extracted sources using the nonnegative model and the L^1 model

are fairly comparable. The L^1 model does seem to suppress the background a bit more, but the extracted target source often exhibit subtle artifacts as opposed to the usually artifact-free extraction from the nonnegativity model. Another issue we try to understand is why the nonnegativity model works even though it is not a realistic physical model. Given the quality of the extracted target sources using the nonnegativity model we are very interested in finding an algorithm that greatly improves the computational efficiency of the nonnegativity model. In this thesis we attempt to address these issues. We develop a new algorithms based on a regularized SVD algorithm. Other than our SVD mehtod, we also apply proximity operator in solving cancellation kernels for BSS. Based on our numerical results, proximity operator is slightly faster than split Bregman while they have the same level of relative error. Later we perform extensive comparisons of various different algorithms. As far as our results can show, the SVD algorithm yields the best results, and it is also the most computationally efficient.

This thesis focuses on background source removal through learning and cancellation kernels. In the rest of this thesis, we shall introduce more details on our mathematical model as well as further background information about various techniques for BSS. Overviews of various algorithms (split Bregman, Promixity operator etc.) will be provided. In the last part of the thesis, several numerical examples will be shown.

1.2 Mathematical Model for Background Cancellation

In practice the time is discretized through sampling. It is convenient to write the convolutive mixtures in the discrete form:

$$X_k(t) = \sum_{i=1}^{L} \sum_{j=0}^{N} a_{k,i,j} S_i(t-j), \qquad k = 1, \ldots, M, \tag{1.2.4}$$

where $t \in \mathbb{Z}$ and $T_0 \leqslant t < T_1$. The integer N denotes the maximal delay in the mixture, which depends on the ambient environment and the sampling rate. We shall view a signal as an element in $l^\infty(\mathbb{Z})$. We shall adopt the notations used in [61]: For each $\mathbf{u} \in l^\infty(\mathbb{Z})$ we use $\mathbf{u}(j)$ to denote its j-th entry and $\text{supp}\,(\mathbf{u})$ its support, i.e. the set of indices of its nonzero entries. For any $\mathbf{u}, X \in l^\infty(\mathbb{Z})$, where $\mathbf{u}$ is finitely supported, the convolution $\mathbf{u} * X \in l^\infty(\mathbb{Z})$ is defined by

$$\mathbf{u} * X(t) = \sum_{j \in \mathbb{Z}} \mathbf{u}(j) X(t-j).$$

Again we may rewrite the model (1.2.4) simply as

$$X_k = \sum_{i=1}^{L} \mathbf{a}_{k,i} * S_i, \qquad k = 1, \ldots, M, \tag{1.2.5}$$

where $\mathbf{a}_{k,i} \in l^\infty(\mathbb{Z})$ is supported on $[0, N]$ with $a_{k,i,j}$ as its j-th element.

Suppose that the sources $S_1, \ldots, S_J$ are the background sources we wish to remove and $S_{J+1}, \ldots, S_L$ are the target sources we wish to preserve from the mixtures X_k, $1 \leqslant k \leqslant M$. For any finitely supported $\mathbf{b}_k \in l^\infty(\mathbb{Z})$ we have

$$\mathbf{b}_k * X_k = \sum_{i=1}^{L} \mathbf{b}_k * \mathbf{a}_{k,i} * S_i. \tag{1.2.6}$$

If we can find $\mathbf{b}_k$ such that $\sum_{k=1}^{M} \mathbf{b}_k * \mathbf{a}_{ki} = 0$ for $1 \leqslant i \leqslant J$, then by (1.2.5) one can easily verify that

$$\sum_{k=1}^{M} \mathbf{b}_k * X_k = \sum_{i=J+1}^{L} \Big(\sum_{k=1}^{M} \mathbf{b}_k * \mathbf{a}_{k,i} \Big) * S_i. \tag{1.2.7}$$

Note that the unwanted background sources are now removed, leaving only the target sources in place. We shall call those $\mathbf{b}_k$ *cancellation kernels*. It was shown in [61] that cancellation kernels always exist. The following proposition was proved in [61], and we include the proof here for self-containment.

Proposition 1.2.1 ([61]) *Let $\mathbf{u}_{k,i} \in l^\infty(\mathbb{Z})$ for $1 \leqslant k \leqslant M$ and $1 \leqslant i \leqslant J$ have finite support. Suppose that $M > J$. Then there exists finitely supported cancellation kernels $\mathbf{b}_1, \ldots, \mathbf{b}_M \in l^\infty(\mathbb{Z})$ not all zero such that*

$$\sum_{k=1}^{M} \mathbf{b}_k * \mathbf{u}_{k,i} = 0, \qquad i = 1, \ldots, J. \tag{1.2.8}$$

Proof. Taking the Fourier transform we need to show the existence of finitely supported $\mathbf{b}_k$ such that

$$\sum_{k=1}^{M} \widehat{\mathbf{b}}_k(\xi) \widehat{\mathbf{u}}_{k,i}(\xi) = 0, \qquad i = 1, \ldots, J.$$

Let $\mathcal{F}$ be the field of all real trigonometric rational functions, i.e. functions of the form f/g where both f, g are trigonometric polynomials with real coefficients. Set $\mathbf{A} = [\widehat{\mathbf{u}}_{k,i}]$ with rows indexed by i and columns indexed by k, which is $J \times M$. Since $M > J$ there exists a $Z = [f_1, \ldots, f_M]^T$ in $\mathcal{F}^M$ such that $\mathbf{A}Z = 0$. Now let $F(\xi)$ be a trigonometric polynomial that is the common denominator of all trigonometric rational functions f_k, $1 \leqslant k \leqslant M$ and $G_k(\xi) = F(\xi) f_k(\xi)$. Each G_k is a trigonometric polynomial with real coefficients. Let

$\mathbf{b}_k \in l^\infty(\mathbb{Z})$ such that $\widehat{\mathbf{b}}_k = G_k$. Then

$$\sum_{k=1}^{M} \mathbf{b}_k * \mathbf{u}_{k,i} = 0, \qquad i = 1, \ldots, J.$$

■

In general cancellation kernels are not unique. Practically the question is how can they be reconstructed. The key idea in [60] and several subsequent papers (e.g., [63]) is that cancellation kernels can be estimated if there is a period of silence by the target sources using various optimization methods.

Observe that if $\mathbf{b}_1, \ldots, \mathbf{b}_M$ are cancellation kernels then so are $\tau_q \mathbf{b}_1, \ldots, \tau_q \mathbf{b}_M$ for any q. By shifting we can thus normalize the cancellation kernels so that all supp $\mathbf{b}_k$ are nonnegative and at least one $\mathbf{b}_k(0) \neq 0$. We shall call such cancellation kernels *normalized*. The general framework for target source extraction we propose in this thesis is to compute the normalized cancellation kernels via background learning. This can be achieved by utilizing an interval of silence for the target sources we wish to extract. Once we have the cancellation kernels the extraction of target can be made through background cancellations. Assume that the target sources $S_{J+1}(t), \ldots, S_L(t)$ are silent in the time interval $a \leqslant t \leqslant b$, i.e. $S_{J+1}(t) = \cdots = S_L(t) = 0$. Let $\mathbf{b}_1, \ldots, \mathbf{b}_M$ be the cancellation kernels. It follows from (1.2.8) that

$$\sum_{k=1}^{M} \mathbf{b}_k * X_k = 0$$

for $a+N \leqslant t \leqslant b$. N is the maximum delay in the mixture which depends on the ambient environment and the sample rate. Thus we can use this interval to learn about the background

and estimate the normalized cancellation kernels by minimizing the cost function

$$F(\mathbf{b}_1, \ldots, \mathbf{b}_M, a, b) := \sum_{a+N \leqslant t \leqslant b} \left| \sum_{k=1}^{M} \mathbf{b}_k * X_k(t) \right|^2, \tag{1.2.9}$$

subject to the constraint $\sum_{k=1}^{M} |\mathbf{b}(0)| = 1$. In practice, we replace the nonlinear constraint $\sum_{k=1}^{M} |\mathbf{b}(0)| = 1$ with the relaxed linear constraint $\sum_{k=1}^{M} \mathbf{b}(0) = 1$, and it seems to work equally well. Once the cancellation kernels are obtained, target sources can be extracted in real time by (1.2.8) through simple convolutions

$$\sum_{k=1}^{M} \mathbf{b}_k * X_k = \sum_{i=J+1}^{L} \Big(\sum_{k=1}^{M} \mathbf{b}_k * \mathbf{u}_{k,i} \Big) * S_i.$$

The cost function $F(\mathbf{b}_1, \ldots, \mathbf{b}_M, a, b)$ with constraint $\sum_{k=1}^{M} \mathbf{b}(0) = 1$ can be minimized easily as it is a quadratic function. It falls into the general category of

$$\min_x \|Au - f\|_2^2, \tag{1.2.10}$$

where $u \in \mathbb{R}^n$ and A is $m \times n$. If it is this simple then there clearly is not any need to go any further. Unfortunately, our numerical experiments have shown that without further constraints the cancellation kernels obtained by minimizing F do not work at all. There could be several problems associated with simply minimizing the cost function. One problem is that the condition number in this setup is typically high. Another problem is the lack of uniqueness even when all cancellation kernels are normalized. But since any cancellation kernels can be used to remove the background sources, this may not be a major problem. A more serious problem is potential *over-fitting*. To overcome these difficulties one must

impose proper level of constraints and regularization.

One possible solution to overcome the over-fitting problem and ill-posed-ness of the problem is to impose a penalty term that serves as both the regularizer and the sparsity inducer. In our experiments the minimizers of (1.2.9) are almost never sparse, particularly for real life recorded mixtures. There are many ways to achieve sparsity, such as adding an l^1-norm penalty term, see e.g. [33] and the references therein. The well-known ROF total variation model is the standard from which many other models had originated. This penalty helps somewhat, and the resulting background removal is often adequate. However, the problem with this approach is that it adds a penalty term that does not seem natural to us. And It can affect the performance. Finding the right sparsity or other conditions to impose is the most important aspect of this approach to source extraction, and at this point is still a work in progress. The main contribution of this thesis is to introduce a new SVD-based regularization algorithm, and to a lesser extend, a proximity operator based algorithm, to address the problems that are encountered in background removal.

Chapter 2

Background on Previous Work and New Approach

2.1 ICA for BSS

Independent component analysis (ICA) was first brought up by Herault and Jutten around 1986 [23], because it is similar to another method, principal component analysis (PCA). Meanwhile, many people, such as Giannakis [21], Cardoso [10], Inouye and Matsui [27], and Fety, contributed to the development of the definition and properties in this area. *ICA* was well defined by Comon in [17], Jutten and Herault [29] precisely, with detailed descriptions of other concepts as well.

Definition 1 *The* ICA *of a random vector y of size N with finite covariance V_y is a pair $\{F, \Delta\}$ of matrices such that*

(a) the covariance factorizes into

$$V_y = F\Delta^2 F^*, \tag{2.1.1}$$

where Δ is diagonal real positive, F is full column rank ρ, and F^* is the transposition of F.

(b) the observation can be written as $y = Fz$, where z is a $\rho \times 1$ random vector with covariance Δ^2 and whose components are 'the most independent possible', in the sense of

the maximization of a given 'contrast function' that will be subsequently defined.

The goal of *ICA* is to find a linear representation of non-Gaussian data so that the components are statistically independent, or as independent as possible. Hyvärinen and Oja provided a thorough review of ICA algorithm applied in BSS [26].

Assume that we observed n linear mixtures $x_1, \ldots, x_n$ of n independent components $s_1, \ldots, s_n$. As the simplest case, ICA model is written as

$$X_j = a_{j1}S_1 + a_{j2}S_2 + \cdots + a_{jn}S_n, \quad 1 \leqslant j \leqslant n, \tag{2.1.2}$$

we can drop the time index t. It can be transfered as matrix form

$$X = AS, \tag{2.1.3}$$

where X is the matrix of all mixtures $\{X_j\}_{j=1}^n$, A is the matrix of the coefficient $\{a_{jk}\}_{j,k=1}^n$, and S is the matrix of $\{S_j\}_{j=1}^n$. A is unknown, and S is the solution we would like to estimate. This is how it is related to BSS. Since the independence implies uncorrelatedness, the constraint of independence reduces the number of free parameters, and simplifies the problem.

The immidiate question is why the sources are required to be non-Gaussian. It can be simply proved by the following counter example.

Assume that the mixing matrix A is orthogonal and the sources $\{S_j\}_{j=1}^n$ are independent and Gaussian in (2.1.3). Then mixtures $\{X_j\}_{j=1}^n$ are uncorrelated and Gaussian which inplies

they are of unit variance. For x_1 and x_2, the joint density is

$$p(x_1, x_2) = \frac{1}{2\pi} e^{-\frac{x_1^2+x_2^2}{2}},$$

which is a symmetric function. It doen't contain any information on the directions of the columns of the mixing matrix A.

Define $z = A^T w$, then $y = w^T X = w^T AS = z^T S$. So y is a linear conbination of s_k with weights z_k. If w is a vector maximizes the non-Gaussianity of $w^T X$, w would correspond to a z which has only one non-zero component proved in [26]. That means $w^T S = z^T S$ is one of the independent components.

In order to measure the non-Gaussianity of $w^T X$ in ICA model, kurtosis is defined [26]. If y is a non-Gaussian random variable with zero mean and unit variance, then kurtosis is defined by

$$kurt(y) = E\{y^4\} - 3(E\{y^2\})^2.$$

By maximizing the the non-Gaussianity, it needs to solve minima for negative or maxima for positive kurtosis. Detailed solutions have been provided in [20, 26]. Negentropy as another measurement is introduced for non-Gaussianity [26]. Minimization of mutual information is designed to minimize the mutual information, which is equivalent to maimizing the non-Gaussianity. Thorough disscussions have been provided in [19, 26]

As mentioned in previous 1.1, there are many techniques designed under *ICA* model, such as JADE [11], Infomax [6, 12, 43]. There are also many extensions of *ICA* algorithm, such as topographic ICA, multidimensional ICA, kernel ICA, tree-dependent component analysis, subband decomposition ICA [13].

There are two ambiguities of *ICA* provided in [13, 26], which are also what we have in

BSS. The first one is that the sources $\{S_j\}_{j=1}^n$ can't be unique, since both A and S are unknown in (2.1.3). The solution would be a scalar multiple of the souces. The second one is that the solution to mixing matrix A could be AP^{-1} and P is a permutation matrix, because there is no order of S and the order can be freely changed.

This helps to understand and solve the BSS problem (1.2.4). In our assumption, all sources are independent and non-Gaussian. Because we use human speech and music as sources, the fact that voices of different people are independent and music is independent to human voice is widely accepted. This topic still deserves further investigation, though it is not the focus of our study.

2.2 DUET for BSS

Degenerate unmixing estimation technique (DUET) firstly appeared in [30]. Jourjine, Rickard and Yilmaz started work on BSS when there are less mixtures than sources, which is degenerated blind source separation. This is challenging because the mixing matrix A in (2.1.3) is not invertible and the traditional work of estimating the inverse matrix of A would not work any more.

Different from models under ICA using the kurtosis or the information optimizations to find the mixing coefficients and the delays, DUET uses the time-frequency orthogonality property of the source signals, W-Disjoint Orthogonality (WDO) [30, 60]. DUET allows for the reconstruction of multiple sources from only two mixtures. Furthermore, it is extremely fast, making it suitable for both batch and dynamic source separation.

In [60], the detailed procedures for the computation of sources are provided. We only list some important steps. Given a windowing function $W(t)$ and a function $f(t)$, the windowed

Fourier transform of f with window W is defined by

$$\widehat{f}^W(\omega,\tau) := \int_R W(t-\tau)f(t)e^{-i\omega t}dt. \tag{2.2.4}$$

Two functions $S_1(t)$ and $S_2(t)$ are called *W-disjoint orthogonal* if the supports of the windowed Fourier transforms of $\widehat{S}_1^W(\omega,\tau)$ and $\widehat{S}_2^W(\omega,\tau)$ are disjoint, i.e.,

$$\widehat{S}_1^W(\omega,\tau)\cdot\widehat{S}_2^W(\omega,\tau) = 0. \tag{2.2.5}$$

for all ω and τ. If any two functions in $S_k(t)_{k=1}^n$ satisfy the WDO property then we say $S_k(t)_{k=1}^n$ satisfy the WDO property.

As an example, if we consider the anechoic model as a simple example of 1.1.1

$$X_k(t) = \sum_{i=1}^{n} a_{k,i}S_i(t-d_{k,i}), \quad k=1,\ldots,m. \tag{2.2.6}$$

When $m=2$ and the constant window function $W(t)=1$, we have

$$\begin{aligned} X_1(t) &= \sum_{k=1}^{n} a_{1,k}S_k(t-d_{1,k}) \\ X_2(t) &= \sum_{k=1}^{n} a_{2,k}S_k(t-d_{2,k}) \end{aligned}$$

Without loss of generality, we can assume $a_{1k}=1$, $d_{1k}=0$, and change the notation

$a_{2k} = a_k$ and $d_{2k} = d_k$. Then we have

$$\begin{aligned} X_1(t) &= \sum_{i=1}^{n} S_k(t) \\ X_2(t) &= \sum_{i=1}^{n} a_k S_k(t - d_k) \end{aligned}$$

With $W(t) = 1$, we have

$$\begin{aligned} \widehat{X}_1^W(\omega, \tau) &= \sum_{k=1}^{n} \widehat{S}_k^W(\omega, \tau) \\ \widehat{X}_2^W(\omega, \tau) &= \sum_{k=1}^{n} a_k e^{-id_k\omega} \widehat{S}_k^W(\omega, \tau) \end{aligned}$$

By the WDO property with any given ω, the function

$$F(\omega, \tau) := \frac{\widehat{X}_2^W(\omega, \tau)}{\widehat{X}_1^W(\omega, \tau)}$$

can only take values in the finite set $\{a_k e^{-id_k\omega} : 1 \leqslant k \leqslant n\}$, which forms the the basis for DUET. Define the amplitude-phase function,

$$\Lambda(\omega, \tau) := (|F(\omega, \tau)|, -\omega^{-1}\Theta(F(\omega, \tau))),$$

where $\Theta(z)$ denotes the angle of z, $-\pi \leqslant \Theta \leqslant \pi$. If there is no wrap-around in $\Theta(F(\omega, \tau))$, then Λ only takes values in the finite set $\{(a_k, d_k) : 1 \leqslant k \leqslant n\}$. Thus each $\widehat{S}_k^W(\omega, \tau)$ can

be solved via following assignment

$$\widehat{S}_k^W(\omega,\tau) = \begin{cases} \widehat{S}_k^W(\omega,\tau) & \text{if } \Lambda(\omega,\tau) = (a_k, d_k), \\ 0 & \text{otherwise.} \end{cases}$$

The fatal problem of DUET is the phase wrap around problem, which causes the faulty assignment of $\widehat{S}_k^W(\omega,\tau)$. Wang, Yilmaz and Zhou [60] introduced a new method to define the amplitude-phase function $\bar{\Lambda}$ with over-sampled FFT. If the window W size is N, in stead of the normal FFT computation of size N data $g_\tau := W(t-\tau)X_i(t)$ that it computes M-point FFT of $\tilde{g_\tau}$ by padding $M-N$ zeros to replace. The numerical experiments in [60] provide extremely good results for 2 sources case. Moreover, the modified DUET improves the results than normal DUET, though it is still not perfect in the separation of three or more sources.

2.3 Cancellation Kernels

From previous discussion and derivation, there is a general idea of cancellation kernels. In this part, it is summarized. For the model (1.2.5)

$$X_k = \sum_{i=1}^{L} \mathbf{a}_{k,i} * S_i, \qquad k = 1, \ldots, M,$$

where $\mathbf{a}_{k,i} \in l^\infty(\mathbb{Z})$ is supported on $[0, N]$ with $a_{k,i,j}$ as its j-th element. The sequence $\{\mathbf{b}_k\}_{k=1}^L \in l^\infty(\mathbb{Z})$, which also satisfy the condition that $\sum_{k=1}^M \mathbf{b}_k * \mathbf{a}_{ki} = 0$ for $1 \leqslant i \leqslant J$, is called cancellation kernels.

By Proposition 1.2.1, it has been proved that the existence of cancellation kernels. There-

fore, (1.2.7) can be verified as

$$\sum_{k=1}^{M} b_k * X_k = (\sum_{i=1}^{J} + \sum_{i=J+1}^{L})(\sum_{k=1}^{M} b_k * a_{k,i}) * S_i \tag{2.3.7}$$

$$= \sum_{i=J+1}^{L} (\sum_{k=1}^{M} b_k * a_{k,i}) * S_i. \tag{2.3.8}$$

In this way, background sources are eliminated. Research can be further conducted for separation of sources $\{S_i\}_{J+1}^{L}$, if there are more than one foreground sources, i.e. $J+1 < L$.

The focus of our work is to estimate the corresponding cancellation kernels as BSS technique. By comparing various optimization methods, including quadratic programming, split Bregman, proximity operator and sigular value decomposition (SVD), and corresponding numerical experiments in the next part, we can choose the ones which provide good results. At the end, the results of SVD provide us very good solutions compared to other optimization methods.

2.4 Optimization Methods

The essencial work of BSS is based on using optimization algarithms to solve the approximation system, which is solving the cancellation kernels in this thesis. At the early stage of optimization study, the most commonly used criteria are based on least square method, however, it has been proved in [52] that the total variation (TV) is more proper than L_2 norm. ROF model is introduced by *L. Rudin*, *S. Osher* and *E. Fatemi* in [54] formulated as

$$arg\min_{u}\{\lambda \parallel f-u \parallel_{L^2}^2 + |u|_{TV} : u \in \mathbb{R}^d\}, \tag{2.4.9}$$

where $f \in \mathbb{R}^d$, λ is the regularization parameter, and $|u|_{TV}$ is the total variation of u. It has successfully demonstrated the advantage of optimizing the local solution, and also is broadly accepted and used as the general model of signal processing, but it has been proved to be very difficult to minimize by conventional methods. In order to make the computation faster, L^1 has been adopted to replace the total variation term. The formula can be written as

$$arg\min_u\{\lambda \parallel f-u \parallel_{L^2}^2 + |u|_1 : u \in \mathbb{R}^d\}. \tag{2.4.10}$$

For estimating cancellatoin kernels, the formula can be rewritten as

$$arg\min_u\{\lambda \parallel Au-f \parallel_{L^2}^2 + |u|_1 : u \in \mathbb{R}^d\}. \tag{2.4.11}$$

In BSS of audio signals, ROF model with L^1 regularization term (2.4.11) is used to solve the cancellation kernels, as we mentioned previously in Section 1.1. u is the combination of cancellation kernels, which is different from the case of image processing that people use u as the true signal (2.4.10). A lot of previous existing algorithms in image processing can be used in BSS of audio signals. In the next part, we are going to introduce some important methods for solving BSS. Quadratic programming is specified for solving cancellation kernels in audio signal processing [60]. Split Bregam is an iteration method originally used in image processing [22]. Xin and Osher applied it in BSS of audio signal process [39, 40, 41, 63, 64]. Proximity operator method is another interation method used by Micchelli, Shen and Xu for BSS of image processing [37, 38]. We apply proximity operator method in solving the cancellation kernels for BBS of audio signal processing.

2.4.1 Quadratic Programming

Wang and Zhou developed the idea of using cancellation kernels and quadratic programming to solve BSS of audio signal processing via background learning [60]. We apply the method in the work to achieve the sparseness of cancellation kernels by restricting the number of nonzero elements in cancellation kernels, and the ratio between numbers of positive to negative elements. Yu et al proposed a modified model by replacing the nonnegativity of the regularization term with L^1 regularization and split Bregman method [63].

As we have already disscussed in 1.2 and above, the cancellation kernels exist so that

$$\sum_{k=1}^{M} \mathbf{b}_k * X_k = 0$$

for $a + N \leqslant t \leqslant b$. N is the maximum delay in the mixture which depends on the ambient environment and the sample rate. Therefor we can use this interval to learn the background and estimate the normalized cancellation kernels by minimizing the cost function (1.2.9)

$$F(\mathbf{b}_1, \ldots, \mathbf{b}_M, a, b) := \sum_{a+N \leqslant t \leqslant b} \left| \sum_{k=1}^{M} \mathbf{b}_k * X_k(t) \right|^2,$$

subject to the constraint $\sum_{k=1}^{M} \mathbf{b}(0) = 1$.

In numerical computation, we only need to choose a time interval with length around 1 second from $[a + N, b]$ for learning the background, for solving the cancellation kernels and extracting the foreground sources. In order to test the error of the result, we choose a

different time interval $[s_{testing}, e_{testing}]$ from $[a+N, b]$ to calculate the error by

$$Err(b_1, \ldots, b_M, a, b) := \sum_{s_{testing} \leqslant t \leqslant e_{testing}} \left| \sum_{k=1}^{M} \mathbf{b}_k * X_k(t) \right|^2 . \tag{2.4.12}$$

In the following part, the formula for the cases of two source and two mixtures and three sources and three mixtures are provided.

2.4.1.1 Two Sources and Two Mixtures

Now let us take the example of two sources and two mixtures case. The analysis can also be found in [60]. When $L = M = 2$, S_1 is the background and S_2 is the foreground. Assume that the foreground source $S_2 = 0$ in the time interval $a \leqslant t \leqslant b$. It simply needs to take $b_1 = a_{2,1}$ and $b_2 = -a_{1,1}$ for cancellation kernels, which satisfy

$$b_1 * a_{1,1} + b_2 * a_{2,1} = 0. \tag{2.4.13}$$

It leads to

$$b_1 * X_1 + b_2 * X_2 = (b_1 * a_{1,2} + b_2 * a_{2,2}) * S_2 = (a_{2,1} * a_{1,2} - a_{1,1} * a_{2,2}) * S_2.$$

Let

$$Y := a_{2,1} * X_1 - a_{1,1} * X_2. \tag{2.4.14}$$

Then $Y(t) = 0$, where $a + N \leqslant t \leqslant b$.

Let $A_n = a_{2,1}(n)$ and $B_n = a_{1,1}(n)$ for $0 \leqslant n \leqslant N$ and 0 otherwise.

Denote

$$A = [A_0, A_1, \ldots, A_N]^T,$$

$$B = [B_0, B_1, \ldots, B_N]^T,$$

and

$$F(t) = [X_1(t), X_1(t-1), \ldots, X_1(t-N),]^T,$$

$$G(t) = [X_2(t), X_2(t-1), \ldots, X_2(t-N),]^T.$$

We could estimate $a_{2,1}, a_{1,1}$ by minimizing

$$\begin{aligned} E(A, B, a, b) &= \sum_{a+N \leqslant t \leqslant b} \mid Y(t) \mid^2 \\ &= \sum_{a+N \leqslant t \leqslant b} [A^T F(t) - B^T G(t)]^2 \\ &= \sum_{a+N \leqslant t \leqslant b} [A^T F(t) F(t)^T A + B^T G(t) G(t)^T B - 2A^T F(t) G(t)^T B] \\ &= U^T D U, \end{aligned}$$

where

$$U = \begin{bmatrix} A \\ B \end{bmatrix}, \qquad D = \begin{bmatrix} D_{FF} & -D_{FG} \\ -D_{GF} & D_{GG} \end{bmatrix},$$

$$
\begin{aligned}
D_{FF} &= \sum_{a+N \leqslant t \leqslant b} F(t)F(t)^T, \\
D_{GG} &= \sum_{a+N \leqslant t \leqslant b} G(t)G(t)^T, \\
D_{FG} &= \sum_{a+N \leqslant t \leqslant b} F(t)G(t)^T, \\
D_{GF} &= \sum_{a+N \leqslant t \leqslant b} G(t)F(t)^T = D_{FG}^T,
\end{aligned}
$$

It results in

$$
\min_{A,B} A^T D_{FF} A + B^T D_{GG} B - 2A^T D_{FG} B \tag{2.4.15}
$$

subject to $A_0 + B_0 = 1$.

2.4.1.2 Three Sources and Three Mixtures

For three sources and three mixtures case, we assume S_1 and S_2 are backgrounds, and S_3 is foreground. Assume that the foreground source $S_3 = 0$ in the time interval $a \leqslant t \leqslant b$. Since $L = M = 3$, there exist cancellation kernels $\{b_k\}_{k=1}^3$ such that

$$
\sum_{k=1}^{3} b_k * X_k = \sum_{i=1}^{3} (\sum_{k=1}^{3} b_k * a_{k,i}) * S_i. \tag{2.4.16}
$$

So our goal is to remove S_1 and S_2 from mixtures X_1, X_2, X_3. Similarly as in previous case, let

$$
Y := b_1 * X_1 + b_2 * X_2 + b_3 * X_3. \tag{2.4.17}
$$

Then $Y(t) = 0$, where $a + N \leqslant t \leqslant b$.

Denote

$$b_1 = [b_1(0), b_1(1), \ldots, b_1(N)]^T,$$

$$b_2 = [b_2(0), b_2(1), \ldots, b_2(N)]^T,$$

$$b_3 = [b_3(0), b_3(1), \ldots, b_3(N)]^T,$$

and

$$F(t) = [X_1(t), X_1(t-1), \ldots, X_1(t-N)]^T,$$

$$G(t) = [X_2(t), X_2(t-1), \ldots, X_2(t-N)]^T,$$

$$H(t) = [X_3(t), X_3(t-1), \ldots, X_3(t-N)]^T.$$

We can estimate b_1, b_2, b_3 by minimizing

$$\begin{aligned} E(b_1, b_2, b_3, a, b) &= \sum_{a+N \leqslant t \leqslant b} | Y(t) |^2 \\ &= \sum_{a+N \leqslant t \leqslant b} [b_1^T F(t) + b_2^T G(t) + b_3^T H(t)]^2 \\ &= \sum_{a+N \leqslant t \leqslant b} [A^T F(t) F(t)^T A + B^T G(t) G(t)^T B - 2A^T F(t) G(t)^T B] \\ &= U^T D U, \end{aligned}$$

where

$$U = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \qquad D = \begin{bmatrix} D_{FF} & D_{FG} & D_{FH} \\ D_{GF} & D_{GG} & D_{GH} \\ D_{HF} & D_{HG} & D_{HH} \end{bmatrix},$$

$$
\begin{aligned}
D_{FF} &= \sum_{a+N\leqslant t\leqslant b} F(t)F(t)^T, \\
D_{GG} &= \sum_{a+N\leqslant t\leqslant b} G(t)G(t)^T, \\
D_{HH} &= \sum_{a+N\leqslant t\leqslant b} H(t)H(t)^T, \\
D_{FG} &= \sum_{a+N\leqslant t\leqslant b} F(t)G(t)^T,
\end{aligned}
$$

$$
\begin{aligned}
D_{FH} &= \sum_{a+N\leqslant t\leqslant b} H(t)H(t)^T, \\
D_{GF} &= \sum_{a+N\leqslant t\leqslant b} G(t)F(t)^T = D_{FG}^T, \\
D_{GH} &= \sum_{a+N\leqslant t\leqslant b} G(t)H(t)^T, \\
D_{HF} &= \sum_{a+N\leqslant t\leqslant b} H(t)F(t)^T = D_{FH}^T, \\
D_{HG} &= \sum_{a+N\leqslant t\leqslant b} H(t)G(t)^T = D_{GH}^T.
\end{aligned}
$$

It leads to

$$
\min_{b_1,b_2,b_3} U^T D U, \tag{2.4.18}
$$

subject to $b_1(0) + b_2(0) + b_3(0) = 1$.

If we can bring the sparseness to cancelation kernels b_k, we probably will have a better computational time than the ones without sparseness. Here we try to get the approximation of them with the minimum number of nonzero elements. The more zeros we have, the less computations we need in solving convolutions. Though it is supposed to be computationally faster and more efficient, the numerical experiments show that the sparseness can not really

improve the results, and it is even more time-consuming sometimes. The detailed description for introducing sparseness is provided below, while the algorithm is provided in the next chapter.

2.4.1.3 Introducing Sparseness

Since the numerical results of the cancelation kernels are not unique for different local intervals, one big challenge becomes the evaluation of different cancelation kernels solved from different intervals. This is the also the part of the original goal of considering the sparseness for all cancelation kernels and pick up the important positions of cancellation kernels.

The main idea of this algorithm is to replace unimportant elements in cancelation kernels with zeros. So these unimportant ones do not have much impact on the results when we evaluate the error term (2.4.12), and keep all the positions of important ones' that those can affect the error term (2.4.12), while we also keep the ones with minimum error term through the process by comparing with previous step as well.

This idea is borrowed from a specific method called support vector machine, from machine learning. It is described in [24] and [42]. The compuation process is listed as below.

1. First of all, we take two non-overlapped smaller intervals, $[s_1, e_1]$ and $[s_2, e_2]$, from the interval $[a + N, b]$, where the foreground sources are silent. $[s_1, e_1]$ is the training set, and $[s_2, e_2]$ is the testing set.

2. Choosing small intervals $[s_{training}, e_{training}] \subset [s_1, e_1]$ and $[s_{testing}, e_{testing}] \subset [s2, e2]$.

3. Solving the corresponding cancellation kernels $\{b_k\}$ on $[s_{training}, e_{training}]$ and taking the error by (2.4.12) on $[s_{testing}, e_{testing}]$.

In the following steps, we are going to find the minimum error with its corresponding

sparsed cancellation kernels.

4. Eliminating the smallest p elements of $\{b_k\}$ in absolute value.

5. Resolving the minimization problem on interval $[t_{training}, e_{training}]$ to find new cancelation kernels, with the constraint that the eliminated positions of $\{b_k\}$ as zeros.

6. Calculating the new error from new cancellation kernels by (2.4.12) on $[s_{testing}, e_{testing}]$.

We repeat the above procedures from 4 to 6 untill we get the minimum error with minimum number of nonzero elements of $\{b_k\}$, since we at least can get rid of p nonzero elements each time. Then, we record all the nonzero positions of $\{b_k\}$.

Starting over from step 1. by randomly choosing another pair of training set $[s_{training}, e_{training}]$ and testing set $[s_{testing}, e_{testing}]$. Follow what are shown above repeatedly Q_1 iteration times, till we get the minimum error during each step. In our numerical examples, it turns out $Q_1 \geqslant 40$ at least to provide good results for cancellation kernels which can remove the background.

7. Geting the frequency of each element in cancelation kernels $\{b_k\}$, V_{Freq}, corresponding to the minimum errors in each step at the end.

In the second stage, we start with previously captured nonzero positions with minimum errors, and estimate the cancelation kernels on the first learning interval, through increasing the nonzero positions at certain percentage in each step to compare the error term by (2.4.12), so that we get the cancelation kernels b_k with minimum error at the end.

Detailed steps are listed as below.

1. Starting to estimate $\{b_k\}$ with only q highest frequency elements as nonzero and leave the rest elements as zero, by minimization function (1.2.9) on $[s_1, e_1]$, where q is a positive integer. Then, dividing $[s_1, e_1]$ into Q_2 equal subintervals, $I_1, \ldots, I_{Q_2}$, where Q_2 is a pre-selected number . Now, we find corresponding error vector $Error$ by function (2.4.12)

and estimated cancellation kernels, on all of I_k, $k = 1, \ldots, Q_2$. Meanwhile, takeing the means of all these errors, $Error_{mean}$.

2. Choosing next $\triangle\ q$ highest frequency elements in cancelation kernels, except the q elements already chosen. Then using these $q+\ \triangle\ q$ nonzero elements to estimate a set of new $\{b_k\}$ by minimization function (1.2.9), on interval $[s_1, e_1]$. Plus we could get a new error vector, $Error_{q+\triangle q}$ and its mean, $Error_{mean,q+\triangle q}$.

3.Running the same procedure as above step 2 iteratively, till we take all the elements with positive frequency of $\{b_k\}$ in V_{Freq}.

4. Taking the set of $\{b_k\}$ with the minimum mean error. Now we can get the foreground sources on the desired time interval by using $\{b_k\}$.

In general, this process can provide the sparse cancellation kernels with important positions, though it takes a long time to go through the interative quadratic programming procedure. The algorithm is listed in Section 3.2.1.

2.4.2 Split Bregman Method

Bregman iteration is originally used in functional analysis for finding extrema of convex functionals [5]. Furthermore, Osher firstly used Bregman iteration in image processing [47]. However, it needs to solve a optimization equation each iterative step which is very expensive computationally.

In [22, 47], authors provided a thorough background introduction of Bregman iteration. Starting with the assumption that E and H are two convex energy functionals, defined over $\mathbb{R}^n$ with $\min_{u\in\mathbb{R}^n} H(u) = 0$, where H is differentiable for simplicity. It leads to the

unconstraint minimization problem,

$$arg\min_u\{E(u)+\lambda H(u):u\in\mathbb{R}^d\}, \tag{2.4.19}$$

which was modified by iteratively solving

$$\begin{aligned} u^{k+1} &= \min_u\{D_E^p(u,u^k)+\lambda H(u)\} && (2.4.20)\\ &= \min_u\{E(u)-<p^k,u-u^k>+\lambda H(u)\}, && (2.4.21)\end{aligned}$$

where $D_E^p(u,v)=E(u)-E(v)-<p,u-v>$, with p is in the subgradient of E at v, and $p^{k+1}=p^k-\nabla H(u^{k+1})$ in [5].

Then Goldstein and Osher introduced the application of split Bregman method for L_1 regularized problems in [22] with a thorough introduction of split Bregman iteration for image processing. It splits the computation by introducing a intermediate term, which simplifies the computation. In [9], Cai et al proved the convergence of the split Bregman iteration under certain assumptions. Later on, Osher et al proposed the linearized Bregman iteration method by transforming the minimization problem in each interative step to a linear approximation for better computational time [8, 48].

Xin and Osher, et al also contributed a lot in applying split Bregman iteration in audio signal processing, [39, 40, 41, 63, 64]. Their work brought up the attention to the importance of fast computation in BSS. Split Bregman method will be numerically demonstrated and compared with the work we have done in following chapters.

2.4.2.1 Bregman Iteration

In the literature [22], it provided a thorough background introduction of Bregman iteration. Starting with the assumption that E and H are two convex energy functionals, defined over $\mathbb{R}^n$ with $\min_{u\in\mathbb{R}^n} H(u) = 0$, where H is differentiable for simplicity. It leads to the unconstraint minimization problem,

$$arg\min_u\{E(u) + \lambda H(u) : u \in \mathbb{R}^d\}, \tag{2.4.22}$$

which was modified by iteratively solving

$$\begin{aligned} u^{k+1} &= \min_u\{D_E^p(u, u^k) + \lambda H(u)\} && (2.4.23)\\ &= \min_u\{E(u) - < p^k, u - u^k > +\lambda H(u)\}, && (2.4.24) \end{aligned}$$

where $D_E^p(u, v) = E(u) - E(v) - < p, u - v >$, with p is in the subgradient of E at v, and $p^{k+1} = p^k - \nabla H(u^{k+1})$ in [5].

In [47], authors analyzed the convergence of Bregman iteration, especially, under fairly weak assumptions on E and H, that $H(u^k) \to 0$ as $k \to 0$. There is a convergence result restated in [22].

Theorem 2.4.1 *Assume that E and H are convex functionals, and that H is differentiable. We also assume that solutions to the sub-problems in (2.4.23) exist. We then have*

1. *Monotonic decrease in H: $H(u^{k+1}) \leqslant H(u^k)$,*
2. *Covergence to a minimizer of H: $H(u^k) \leqslant H(u^*) + E(u^*)/k$,*

where u^* is the minimizer of $H(u)$.

2.4.2.2 Constrained Optimization

For problem

$$arg\min_u E(u) \quad such \quad that \quad Au = b, \tag{2.4.25}$$

where A is a linear operator and b is a vector. By applying formula (2.4.22), it can be transformed to unconstrained problem using $\| \cdot \|_2$,

$$arg\min_u \{E(u) + \frac{\lambda}{2} \| Au - b \|_2^2\}, \tag{2.4.26}$$

where λ is a positive constant.

Furthermore, we apply Bregman iteration (2.4.23) to derive,

$$\begin{aligned} u^{k+1} &= \min_u \{E(u) - < p^k, u - u^k > + \frac{\lambda}{2} \| Au - b \|_2^2\}, & (2.4.27) \\ p^{k+1} &= p^k - \lambda A^T(Au^{k+1} - b), & (2.4.28) \end{aligned}$$

where A^T is the transpose of A, and $< \cdot, \cdot >$ is the normal inner product. When A is linear, it is equivalent to solve,

$$\begin{aligned} u^{k+1} &= \min_u \{E(u) + \frac{\lambda}{2} \| Au - b \|_2^2\}, & (2.4.29) \\ b^{k+1} &= b^k + b - Au^k. & (2.4.30) \end{aligned}$$

The following theorem from [22] states that the solution, u^*, through this process (2.4.29) - (2.4.30), is the solution of (2.4.25).

Theorem 2.4.2 *Let H: $\mathbb{R}^d \to \mathbb{R}$ be convex, and A: $\mathbb{R}^n \to \mathbb{R}^m$ be linear. Consider the algorithm (2.4.29) - (2.4.30). Suppose that u^*, satisfies that $Au^* = b$. Then u^* is a solution*

to the original constrained problem (2.4.25).

2.4.2.3 Split Bregman

Bregman iteration is a good tool for solving optimization problems, however, it needs to solve the minimization

$$D_E^p(u, u^k) + \lambda H(u)$$

in (2.4.23) at each iterative step, which can computationally expensive. The split Bregman iteration method introduces an intermediate variable which makes the ROF model separable and easier to solve. Instead of considering the original Bregman problem as a constraint problem, according to [22], the split Bregman method aims to solve the unconstrained problem

$$arg\min_{u,d}\{|d| + H(u) + \frac{\lambda}{2} \parallel \Phi(u) - d \parallel_2^2\}. \tag{2.4.31}$$

By introducing $E(u, d) = |d| + H(u)$, and $A(u, d) = d - \Phi(u)$, it is clear that (2.4.31) is simply an implementation of (2.4.26). Correspondingly, we should have the following iteration,

$$\begin{aligned}
(u^{k+1}, d^{k+1}) &= arg\min_{u,d}\{E(u, d) - < p_u^k, u - u^k > - < p_d^k, d - d^k > + \frac{\lambda}{2} \parallel \Phi(u) - d \parallel_2^2\}, \\
p_u^{k+1} &= p_u^k - \lambda(\nabla\Phi)^T(\Phi(u^{k+1}) - d^{k+1}), \\
p_d^{k+1} &= p_d^k - \lambda(d^{k+1} - \Phi(u^{k+1})).
\end{aligned}$$

Similarly, following (2.4.29) - (2.4.30), by introducing $b^k = p_d^k/\lambda$, we notice that $p_d^k = \lambda b^k$, and $p_u^k = -\lambda(\nabla\Phi)^T b^k$. Then it leads to the equivalent algorithm from [41], by setting $d^0 = 0$,

$u^0 = 0$, and $b^0 = 0$,

$$
\begin{aligned}
d^{k+1} &= arg\min_d\{\frac{1}{\lambda}J(d) - < b^k, d - d^k > + \frac{1}{2} \parallel \Phi(u^k) - d \parallel_2^2, \} && (2.4.32)\\
u^{k+1} &= arg\min_u\{\frac{1}{\lambda}H(u) + < b^k, \Phi(u - u^k) > \\
&\quad + \frac{1}{2} \parallel d^{k+1} - \Phi(u^k) \parallel_2^2\}, && (2.4.33)\\
b^{k+1} &= b^k - (d^{k+1} - \Phi(u^{k+1})). && (2.4.34)
\end{aligned}
$$

Convergence theorem is derived in [9], for the case of $J(u) = \mu \parallel u \parallel_1$, also restated in [41] as below,

Theorem 2.4.3 *Assume that there exists at least one solution u^* of*

$$arg\min_u\{J(\Phi(u)) + H(u)\}, \tag{2.4.35}$$

where J is convex but not necessarily differentiable, H is convex and differentiable, and Φ is a linear operator. Then we have the following properties for the split Bregman iteration (2.4.32) - (2.4.34),

$$\lim_{k\to\infty}\{\mu \parallel \Phi(u^k) \parallel_1 + H(u^k)\} = \mu \parallel \Phi(u^*) \parallel_1 + H(u^*). \tag{2.4.36}$$

Furthermore,

$$\lim_{k\to\infty} \parallel u^k - u^* \parallel_2 = 0,$$

if u^* is the unique solution.

2.4.3 Proximity Operator

We apply this method in BSS of audio signal processing to solve the cancellation kernels. The major conclusions of proximity operator are listed in this chapter. The detailed algorithm is the the next chapter and numerical results are provided at the end.

Proximity operator was firstly introduced by Moreau in [34], and further investigated in [35] [36]. Combettes also contributed to the initiation of using proximity operator in [14]. Micchelli, Shen and Xu applied proximity operator method for BSS in image processing [37, 38]. They also provided a detailed analysis in comparing proximity operator with split Bregman in image processing to show that proximity operator is better than split Bregman.

2.4.3.1 Definitions

Definition 2 *Let ψ be a real-valued convex function on $\mathbb{R}^d$. The proximity operator of ψ is defined for $x \in \mathbb{R}^d$ by*

$$prox_{\psi} x := arg\min\{\frac{1}{2} \parallel u - x \parallel_2^2 + \psi(u) : u \in \mathbb{R}^d\}. \tag{2.4.37}$$

In ROF model (2.4.9), the major difficulty is in minimizing the total variation term $|\cdot|_{TV}$. C. Micchelli, L. Shen, and Y. Xu proposed a different way of solving the ROF model for image processing, [37, 38], and concluded that it is faster than split Bregman in the specific form,

$$arg\min_u\{\frac{1}{2} \parallel u - x \parallel_2^2 + \mu|u|_{TV} : u \in \mathbb{R}^d\}, \tag{2.4.38}$$

where $x \in \mathbb{R}^d$ denotes the noisy signal, μ is the regularization parameter, by noting that the solution of the ROF model can be viewed as the proximity operator of $|\cdot|_{TV}$ evaluated

at the noisy signal x. Since it cannot be easily computed, it is treated that $|\cdot|_{TV}$ is the composition of convex function, such as the norm $\|\cdot\|_1$ for the anisotropic total-variation or a certain combination of the norm $\|\cdot\|_2$ for the isotropic total variation, with a linear transformation (the first-order difference operator). This approach will be adopted in next chapter for solving BSS accoustic signal processing.

Definition 3 *Let ψ be a real-valued convex function on $\mathbb{R}^d$. The subdifferential of ψ at $x \in \mathbb{R}^d$ is defined by*

$$\partial\psi(x) := \{y : y \in \mathbb{R}^d \quad and \quad \psi(z) \geqslant \psi(x) + < y, z - x > \quad for \quad all \quad z \in \mathbb{R}^d\}. \quad (2.4.39)$$

Elements in $\partial\psi(x)$ are called subgradients.

If $\lambda > 0$ and $x \in \mathbb{R}$, then the well-known soft thresholding operator with $\frac{1}{\lambda}$ as the threshold is,

$$prox_{\frac{1}{\lambda}|\cdot|}x = \max\{|x| - \frac{1}{\lambda}, 0\}sign(x). \quad (2.4.40)$$

Furthermore, if $\lambda > 0$ and $x \in \mathbb{R}^m$, we have

$$prox_{\frac{1}{\lambda}\|\cdot\|_1}x = [prox_{\frac{1}{\lambda}|\cdot|}x_1, prox_{\frac{1}{\lambda}|\cdot|}x_2, \ldots, prox_{\frac{1}{\lambda}|\cdot|}x_m]^T. \quad (2.4.41)$$

It is apparent to directly verify that if ψ is a convex function on $\mathbb{R}^d$ and $x \in \mathbb{R}^d$, then

$$y \in \partial\psi(x)$$

if and only if

$$x = prox_{\psi}(x + y).$$

2.4.3.2 Fixed Point Algorithm Based on Proximity Operator

For a convex function φ on $\mathbb{R}^m$ and an $m \times d$ matrix B, authors in [37] define a convex function

$$\psi(x) := (\varphi \circ B)(x). \tag{2.4.42}$$

Since the aim is to compute the proximity operator $prox_{\psi}$, i.e., $prox_{\varphi \circ B}$, then consider the minimization problem,

$$arg\min_{u}\{\frac{1}{2} \parallel u - x \parallel_2^2 + (\varphi \circ B)(u) : u \in \mathbb{R}^d\}, \tag{2.4.43}$$

where $x \in \mathbb{R}^d$ is given. Then the ROF model corresponds to special cases of this minimization problem.

By replacing ψ with $\varphi \circ B$, we have following several properties:

$$prox_{\varphi \circ B}x \in x - \partial(\varphi \circ B)(prox_{\varphi \circ B}x), \tag{2.4.44}$$

$$\partial(\varphi \circ B) = B^T \circ (\partial\varphi) \circ B, \tag{2.4.45}$$

$$prox_{\varphi \circ B}x \in x - B^T\partial\varphi(Bprox_{\varphi \circ B}x). \tag{2.4.46}$$

By introducing the affine transformation $A : \quad \mathbb{R}^m \to \mathbb{R}^m$ for all $y \in \mathbb{R}^m$ as

$$Ay := Bx + (I - \lambda BB^T)y \tag{2.4.47}$$

for a fixed $x \in \mathbb{R}^d$, then one can define the operator $A: \quad \mathbb{R}^m \to \mathbb{R}^m$ given by

$$H := (I - prox_{\frac{1}{\lambda}\varphi}) \circ A. \tag{2.4.48}$$

Theorem 2.4.4 *If φ is a convex function on $\mathbb{R}^m$, B is a $m \times d$ matrix, $x \in \mathbb{R}^d$ and λ is a positive number then*

$$prox_{\varphi \circ B} x = x - \lambda B^T v \tag{2.4.49}$$

if and only if $v \in \mathbb{R}^m$ is a fixed-point of H.

Theorem (2.4.4) shows that the minimization problem (2.4.43) corresponds to a fixed point of H, which involves the proximity operator of the convex function $\frac{1}{\lambda}\varphi$ and a linear transformation B. As a direct result, there is the following proposition.

Proposition 2.4.5 *If φ is a convex function on $\mathbb{R}^m$, B is a $m \times d$ matrix, $x \in \mathbb{R}^d$ and λ is a positive number, then H has a fixed point.*

For a Lipschitz continuous function φ with Lipschitz constant K that is

$$|\varphi(u) - \varphi(v)| \leqslant K \parallel u - v \parallel, \quad for \quad all \quad u, v \in \mathbb{R}^m. \tag{2.4.50}$$

Let Lip_φ represent the smallest constant K in this inequality. There is following lemma.

Lemma 2.4.6 *If φ is a Lipschitz continuous convex function and $y \in \partial\varphi(x)$ for some $x \in \mathbb{R}^m$, then*

$$\parallel y \parallel_* \leqslant Lip_\varphi, \tag{2.4.51}$$

where $\parallel \cdot \parallel_*$ represents the dual norm.

Let

$$C_\varphi := \{z : z \in \mathbb{R}^m, \| z \|_* \leqslant Lip_\varphi/\lambda\}. \tag{2.4.52}$$

The following lemma is established in [37].

Lemma 2.4.7 *If φ is a Lipschitz continuous convex function and B is an $m \times d$ matrix, then H maps R^m into C_φ. If λ is positive number, then all the fixed points of H are in C_φ.*

2.4.3.3 Nonexpansivity of H

Definition 4 *A nonlinear operator $\mathcal{P}$: $\mathbb{R}^d \to \mathbb{R}^d$ is called nonexpansive if*

$$\| \mathcal{P}(x) - \mathcal{P}(y) \|_2 \leqslant \| x - y \|_2, \tag{2.4.53}$$

for all x, $y \in \mathbb{R}^d$.

P. Combettes, and V. Wajs in [15] proved that

$$\| (I - prox_\psi)(x) - (I - prox_\psi)(y) \|_2^2 \leqslant < x - y, (I - prox_\psi)(x) - (I - prox_\psi)(y) > . \tag{2.4.54}$$

By applying Cauchy-Schwarz inequality to (2.4.54), it shows $I - prox_\psi$ is nonexpansive. H is nonexpansive based on next lemma.

Lemma 2.4.8 *If φ is a convex function on $\mathbb{R}^d$, B is an $m \times d$ matrix, and λ is positive number such that $\| I - \lambda BB^T \|_2 \leqslant 1$, then H is nonexpansive.*

Definition 5 *A nonlinear operator $\mathcal{P}$: $\mathbb{R}^d \to \mathbb{R}^d$ is called nonexpansive averaged if there are a number $\kappa \in (0, 1)$ and a nonexpansive operator Q such that*

$$\mathcal{P} = \kappa I + (1 - \kappa)Q. \tag{2.4.55}$$

$\mathcal{P}$ is specifically called nonexpansive κ-averaged. By direct computation, one can verify that the composition of two nonexppansive averaged operators is also nonexpansive averaged.

Then these general results can be applied on H defined in (2.4.48).

Proposition 2.4.9 *If φ is a convex function on $\mathbb{R}^d$, B is an $m \times d$ matrix, and λ is positive number such that $\| I - \lambda BB^T \|_2 \leqslant 1$, then H is nonexpansive κ-averaged for some $\kappa \in (0,1)$.*

In order to show the convergence, we need to construct Picard sequence. For an operator $\mathcal{P} : \mathbb{R}^m \to \mathbb{R}^m$, with initial point $v^0 \in \mathbb{R}^m$, we have the following Picard sequence,

$$v^{n+1} := \mathcal{P}(v^n), \tag{2.4.56}$$

for $n \in \mathbb{N}$. For $\kappa \in (0,1)$, the definition of κ-averaged operator $\mathcal{P}_\kappa$ for $\mathcal{P}$ is given by

$$\mathcal{P}_\kappa := \kappa I + (1-\kappa)\mathcal{P}. \tag{2.4.57}$$

Let $\mathcal{P}_0 = \mathcal{P}$.

Z. Opial in [45], shows the convergence of the Picard sequence of $\mathcal{P}_\kappa$.

Lemma 2.4.10 *If C is a closed and convex set in $\mathcal{R}^m$ and $\mathcal{P} : C \to C$ is a nonexpansive mapping having at least one fixed point, then for $\kappa \in (0,1)$, $\{\mathcal{P}_k\}$ is nonexpansive, maps C to itself, and has the same set of the fixed point as $\mathcal{P}$. Moreover, for any $u \in C$ and $\kappa \in (0,1)$, the Picard sequence of $\mathcal{P}_\kappa$ converges to a fixed point of $\mathcal{P}$.*

Correspondingly, to H, authors in [37] came up with the result.

Theorem 2.4.11 *If φ is a Lipschitz continuous convex function, B is an $m \times d$ matrix, and λ is positive number, such that $\| I - \lambda BB^T \|_2 \leqslant 1$, then for any initial point in C_φ and any $\kappa \in (0,1)$, the Picard sequence of H_κ converges to a fixed point of H.*

As a consequence of (2.4.9), the authors in [37] derived the following convergence theorem with a slighly stronger condition.

Theorem 2.4.12 *If φ is a Lipschitz continuous convex function, B is an $m \times d$ matrix, and λ is positive number, such that $\| I - \lambda BB^T \|_2 < 1$, then for any initial point in C_φ, the Picard sequence of H converges to a fixed point of H.*

They also showed that H is averaged nonexpansive as far as

$$\lambda < \frac{1}{4\sin^2 \frac{(N-1)\pi}{2N}} \tag{2.4.58}$$

where $N = \sqrt{d}$.

2.4.3.4 Algorithm

By setting $\varphi(x) = \mu \| x \|_1$, the ROF model (2.4.9) becomes

$$arg\min_u \{\frac{1}{2} \| u - x \|_2^2 + \mu \| Bu \|_1 : u \in \mathbb{R}^d\}. \tag{2.4.59}$$

The algorithm can be generalized in [37] as

Given x; $\lambda < \frac{1}{4\sin^2 \frac{(N-1)\pi}{2N}}$, $\mu > 0$, $\kappa \in [0, 1)$

Initialize: $v^0 = 0$

For $n = 0, 1, \ldots$

$v^{n+1} := \kappa v^n + (1 - \kappa)(I - prox_{\frac{\mu}{\lambda}\|\cdot\|_1})(Bx + (I - \lambda BB^T)v^n)$

End

Write the output of v^n from the above loop as v^∞ and compute

$prox_{\varphi \circ B} x = x - \lambda B^T v^\infty$.

The convergence is guaranteed by the following proposition.

Proposition 2.4.13 *If $\lambda > 0$ and a, $x^0 \in \mathbb{R}^m$, then the iterative scheme*

$$x^{k+1} = (I - prox_{\frac{1}{\lambda}\|\cdot\|_1})(x^k + a), \quad k = 0, 1, \ldots, \tag{2.4.60}$$

converges to its limit in a finite number of steps and for $i = 0, 1, \ldots, m$.

2.4.4 Singular Value Decomposition

In this part we will only review some important properties of singular value decomposition. The detailed techniques by applying singular value decomposition are provided in the next chapter.

Singular value decomposition (SVD) is a classic topic in matrix analysis and numeric computation. Compared with previous iteration methods, it solves the solution of the system directly. The detailed methodology and adjustment will be provided in the next chapter as our new approach to ROF model.

For a given $m \times n$ real or complex matrix A, there exist a factorization form of A,

$$A = U\Sigma V^*, \tag{2.4.61}$$

where U is a $m \times m$ unitary matrix, Σ is a $m \times n$ rectangular diagonal matrix with nonnegative real numbers on the diagonal, and V^* is the conjugate transpose of V as a $n \times n$ unitary matrix.

The diagonal entries σ_j of Σ are so called singular values of A, by convention arranged in non-increasing order.

SVD can be used for computing the pseudoinverse of a matrix A. Indeed, the pseudoinverse of matrix A with its SVD $A = U\Sigma V^*$ is

$$A^+ = V\Sigma^+ U^*,$$

where Σ^+ is the pseudoinverse of Σ, formed by replacing every non-zero diagonal entry by its reciprocal and transposing the resulting matrix, and U^* is the conjugate transpose of U.

Another important property of SVD is that if A is a symmetric real $n \times n$ matrix, there is an orthogonal matrix V and a diagonal matrix D such that $A = VDV^T$. Then this is also the eigenvalue decomposition. The diagonal elements of D are the eigenvalues of A in descending order.

In general, if a matrix A with size $m \times n$, then $A^T A$ is a symmetric $n \times n$ matrix. The singular value decomposition of $A^T A$ can be written as

$$A^T A = VDV^T. \tag{2.4.62}$$

Chapter 3

Technical Discussions

3.1 Review of other algorithms

3.1.1 Split Bregman iteration

Based on previous discussion in 2.4.2.3, we use the model provided in [22], by Goldstein and Osher, instead of ROF model (2.4.9),

$$arg\min_u\{\frac{\mu}{2} \parallel Au - f \parallel_2^2 + J(u)\}. \tag{3.1.1}$$

Equivalently,

$$arg\min_u\{\frac{1}{2} \parallel Au - f \parallel_2^2 + \mu J(u)\}. \tag{3.1.2}$$

3.1.1.1 Algorithm of Split Bregman

By applying split Bregman schemes (2.4.32) - (2.4.34), we have the scheme,

$$\begin{aligned}
(u^{k+1}, d^{k+1}) &= arg\min_{u,d} E(u,d) - < p_u^k, u - u^k > - < p_d^k, d - d^k > + \frac{\lambda}{2} \parallel \Phi(u) - d \parallel_2^2, \\
p_u^{k+1} &= p_u^k - \lambda(\Phi)^T(\Phi(u^{k+1}) - d^{k+1}), \\
p_d^{k+1} &= p_d^k - \lambda(d^{k+1} - \Phi(u^{k+1})).
\end{aligned}$$

Let $b^k = p_d^k/\lambda$, we have $p_d^k = \lambda b^k$ and $p_u^k = -\lambda\Phi^T b^k$. Then the general split Bregman

iteration is,

$$\begin{aligned} d^{k+1} &= arg\min_d \frac{1}{\lambda}J(d) - < b^k, d - d^k > + \frac{1}{2} \parallel \Phi(u^k) - d \parallel_2^2, \\ u^{k+1} &= arg\min_u \frac{1}{\lambda}H(u) + < b^k, \Phi(u - u^k) > + \frac{1}{2} \parallel d^{k+1} - \Phi(u^k) \parallel_2^2, \\ b^{k+1} &= b^k - (d^{k+1} - \Phi(u^{k+1})). \end{aligned}$$

In [40, 41, 63, 64], authors used the tansformed (3.1.1) to

$$arg\min_u \{\frac{1}{2} \parallel Au - f \parallel_2^2 + \mu \parallel u \parallel_1\}. \tag{3.1.3}$$

In this case, $J(u) = \mu \parallel u \parallel_1$, $\Phi = I$, and $H(u) = \frac{1}{2} \parallel Au - f \parallel_2^2$. The iteration are

$$d^{k+1} = arg\min_d \frac{\mu}{\lambda} \parallel d \parallel_1 - < b^k, d - d^k > + \frac{1}{2} \parallel d - u^k \parallel_2^2 \tag{3.1.4}$$

$$\begin{aligned} u^{k+1} &= arg\min_u \frac{1}{2\lambda} \parallel Au - f \parallel_2^2 + < b^k, u - u^k > \\ &\quad + \frac{1}{2} \parallel d^{k+1} - u \parallel_2^2 \end{aligned} \tag{3.1.5}$$

$$b^{k+1} = b^k - d^{k+1} + u^{k+1}. \tag{3.1.6}$$

By solving (3.1.4) and (3.1.5) explicitly, it results in a simple algorithm,

Initialize $u^0 = d^0 = b^0 = 0$

While $\parallel u^{k+1} - u^k \parallel_2 > \epsilon$

$d^{k+1} = shrink(u^k + b^k, \frac{\mu}{\lambda})$

$u^{k+1} = (\varepsilon I + A^T A)^{-1}(A^T f + \lambda(d^{k+1} - b^k))$

$b^{k+1} = b^k - d^{k+1} + u^{k+1}$

end While

where $shrink(v,t) = (\tau_t(v_1), \tau_t(v_2), \cdots, \tau_t(v_{NL}))^T$ with $\tau_t(x) = sign(x)\max\{|x| - t, 0\}$.

The $shrink(v,t)$ is a soft threshhold function. Clearly, the step

$$u^{k+1} = (\varepsilon I + A^T A)^{-1}(A^T f + \lambda(d^{k+1} - b^k)) \tag{3.1.7}$$

brings in error as a regularization term. We will compare this with the the SVD method applied in the later part.

3.1.1.2 Two Sources and Two Mixtures

Let us start with simple determined case, 2 sources and 2 mixtures, following the same strategy as [40], [41], [63], [64]. As defined previously, 2 sources are background source S_1 and foreground source S_2, 2 mixtures are X_1 and X_2. It leads to mixing model

$$X_j = a_{j,1} * S_1 + a_{j,2} * S_2, \tag{3.1.8}$$

where $j = 1, 2$. In order to extract foreground source S_2, as we assumed previously, $S_2 = 0$, where $a \leqslant t \leqslant b$, we have to find cancelation kernels u_1 and u_2, such that

$$u_1 * X_1 + u2 * X_2 = u_1 * a_{1,1} * S_1 + u_2 * a_{2,1} * S_1 \approx 0. \tag{3.1.9}$$

An ideal pair of solutions are

$$u_1 \approx a_{2,1}, \quad u_2 \approx -a_{1,1}. \tag{3.1.10}$$

Based on previous discussion in quadratic programming, we can get the optimization

problem as

$$\begin{aligned}(u_1^*, u_2^*) &= arg \min_{(u_1,u_2)} \frac{1}{2} \parallel u_1 * X_1 + u_2 * X_2 \parallel_2^2 \\ &\quad +\mu(\parallel u_1 \parallel_1 + \parallel u_2 \parallel_1) \end{aligned} \tag{3.1.11}$$

subject to $u_1(1) + u_2(1) = 1$.

Equivalently, the authors proposed problem

$$\begin{aligned}(u_1^*, u_2^*) &= arg \min_{(u_1,u_2)} \frac{1}{2} \parallel u_1 * X_1 + u_2 * X_2 \parallel_2^2 + \frac{\eta^2}{2}(\sum_{j=1}^{2} u_j(1) - 1)^2 \\ &\quad +\mu(\parallel u_1 \parallel_1 + \parallel u_2 \parallel_1). \end{aligned} \tag{3.1.12}$$

In matrix form, (3.1.12) corresponds to (3.1.3) as

$$u^* = arg \min_u \{\frac{1}{2} \parallel Au - f \parallel_2^2 + \mu \parallel u \parallel_1\}, \tag{3.1.13}$$

where,

$$A = \begin{bmatrix} X_1(1) & X_1(2) & \cdots & \cdots & X_1(m-2) & X_1(m-1) & \eta \\ & X_1(1) & \cdots & \cdots & X_1(m-3) & X_1(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_1(1) & \cdots & X_1(m-d) & 0 \\ X_2(1) & X_2(2) & \cdots & \cdots & X_2(m-2) & X_2(m-1) & \eta \\ & X_2(1) & \cdots & \cdots & X_2(m-3) & X_2(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_2(1) & \cdots & X_2(m-d) & 0 \end{bmatrix}^T,$$

$$
\begin{aligned}
f &= [0 \quad 0 \quad \cdots \quad 0 \quad \eta]^T, \\
X_1 &= [X_1(1) \quad X_1(2) \quad \cdots \quad X_1(m-2) \quad X_1(m-1)]^T, \\
X_2 &= [X_2(1) \quad X_2(2) \quad \cdots \quad X_2(m-2) \quad X_2(m-1)]^T,
\end{aligned}
$$

$m-1$ represents the length of the audio signal, f with size $m \times 1$, and d represents the length of the cancellation kernel, X_1 and X_2 are two mixtures.

3.1.1.3 Three Sources and Three Mixtures

Similarly, in 3 sources and 3 mixtures case, we should have

$$
\begin{aligned}
(u_1^*, u_2^*, u_3^*) &= arg \min_{(u_1,u_2,u_3)} \frac{1}{2} \parallel u_1 * X_1 + u_2 * X_2 + u_3 * X_3 \parallel_2^2 + \frac{\eta^2}{2} (\sum_{j=1}^{3} u_j(1) - 1)^2 \\
&\quad + \mu(\parallel u_1 \parallel_1 + \parallel u_2 \parallel_1 + \parallel u_3 \parallel_1). \qquad (3.1.14)
\end{aligned}
$$

Corresponding to (3.1.13),

$$A = \begin{bmatrix} X_1(1) & X_1(2) & \cdots & \cdots & X_1(m-2) & X_1(m-1) & \eta \\ & X_1(1) & \cdots & \cdots & X_1(m-3) & X_1(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_1(1) & \cdots & X_1(m-d) & 0 \\ X_2(1) & X_2(2) & \cdots & \cdots & X_2(m-2) & X_2(m-1) & \eta \\ & X_2(1) & \cdots & \cdots & X_2(m-3) & X_2(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_2(1) & \cdots & X_2(m-d) & 0 \\ X_3(1) & X_3(2) & \cdots & \cdots & X_3(m-2) & X_3(m-1) & \eta \\ & X_3(1) & \cdots & \cdots & X_3(m-3) & X_3(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_3(1) & \cdots & X_3(m-d) & 0 \end{bmatrix}^T .$$

$$\begin{aligned} f &= [0 \quad 0 \quad \cdots \quad 0 \quad \eta]^T, \\ X_1 &= [X_1(1) \quad X_1(2) \quad \cdots \quad X_1(m-2) \quad X_1(m-1)]^T, \\ X_2 &= [X_2(1) \quad X_2(2) \quad \cdots \quad X_2(m-2) \quad X_2(m-1)]^T, \\ X_3 &= [X_3(1) \quad X_3(2) \quad \cdots \quad X_3(m-2) \quad X_3(m-1)]^T, \end{aligned}$$

where $m-1$ represents the length of the audio signal; f is in size $m \times 1$; d represents the length of the cancellation kernel; X_1, X_2, and X_3 are three different mixtures.

3.1.1.4 n sources and n mixtures

In general, if there n sources and n mixtures, we are going to solve the following problem,

$$\begin{aligned}(u_1^*, u_2^*, u_3^*, ..., u_n^*) &= arg \min_{(u_1,u_2,u_3,...,u_n)} \frac{1}{2} \parallel \sum_{j=1}^{n} u_j * X_j \parallel_2^2 + \frac{\eta^2}{2} (\sum_{j=1}^{n} u_j(1) - 1)^2 \\ &\quad + \mu \sum_{j=1}^{n} \parallel u_j \parallel_1 . \end{aligned} \tag{3.1.15}$$

Corresponding to (3.1.13),

$$A = \begin{bmatrix} & & & & & & \\ X_1(1) & X_1(2) & \cdots & \cdots & X_1(m-2) & X_1(m-1) & \eta \\ & X_1(1) & \cdots & \cdots & X_1(m-3) & X_1(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_1(1) & \cdots & X_1(m-d) & 0 \\ X_2(1) & X_2(2) & \cdots & \cdots & X_2(m-2) & X_2(m-1) & \eta \\ & X_2(1) & \cdots & \cdots & X_2(m-3) & X_2(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_2(1) & \cdots & X_2(m-d) & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n(1) & X_n(2) & \cdots & \cdots & X_n(m-2) & X_n(m-1) & \eta \\ & X_n(1) & \cdots & \cdots & X_n(m-3) & X_n(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_n(1) & \cdots & X_n(m-d) & 0 \end{bmatrix}^T .$$

$$
\begin{aligned}
f &= [0 \quad 0 \quad \cdots \quad 0 \quad \eta]^T, \\
X_1 &= [X_1(1) \quad X_1(2) \quad \cdots \quad X_1(m-2) \quad X_1(m-1)]^T, \\
X_2 &= [X_2(1) \quad X_2(2) \quad \cdots \quad X_2(m-2) \quad X_2(m-1)]^T, \\
&\vdots \\
X_n &= [X_n(1) \quad X_n(2) \quad \cdots \quad X_n(m-2) \quad X_n(m-1)]^T,
\end{aligned}
$$

where $m-1$ represents the length of the audio signal; f is in size $m \times 1$; d represents the length of the cancellation kernel; $\{X_k\}_{k=1}^n$ are n different mixtures.

3.1.2 Proximity Operator

As previously mentioned in 2.4.3, the proximity operator has been used in image processing. We use it to estimate cancellation kernels for BSS. In general, we consider the following model

$$arg\min_u \{\frac{1}{2} \parallel u - x \parallel_2^2 + (\varphi \circ B)(u)\} \tag{3.1.16}$$

Compared with the ROF model (2.4.9), specifically used in this case (2.4.38)

$$arg\min_u \{\frac{1}{2} \parallel u - x \parallel_2^2 + \mu |u|_{TV} : u \in \mathbb{R}^d\},$$

clearly, one just needs to set up $\varphi(x) = \mu \parallel x \parallel_1$. Then $|\cdot|_{TV}$ can be rewritten as

$$|\cdot|_{TV} = \parallel \cdot \parallel_1 \circ B. \tag{3.1.17}$$

Next, the problem becomes,

$$arg\min_{u}\{\frac{1}{2}\parallel u-x\parallel_2^2+\mu\parallel Bu\parallel_1\} \tag{3.1.18}$$

3.1.2.1 Algorithm of proximity operator

Followed by (3.1.18), recall the algorithm mentioned in Section 2.4.3.4.

Given x; $\lambda < \frac{1}{4\sin^2\frac{(N-1)\pi}{2N}}$, $\mu > 0$, $\kappa \in [0,1)$

Initialize: $v^0 = 0$

For $n = 0, 1, \ldots$

$v^{n+1} := \kappa v^n + (1-\kappa)(I - prox_{\frac{\mu}{\lambda}\|\cdot\|_1})(Bx + (I-\lambda BB^T)v^n)$

End

Write the output of v^n from the above loop as v^∞ and compute

$prox_{\varphi\circ B}x = x - \lambda B^T v^\infty$.

By solving v^∞ and replacing the corresponding term in the last step, we have the following algorithm.

Given x, $\lambda < \frac{1}{4}\sin^{-2}(\frac{(N-1)\pi}{2N})$, $\mu > 0$, $\kappa \in [0,1)$

Initialize $u^0 = 0$

While $\parallel u^{k+1} - u^k \parallel_2 > \epsilon$

$u^{k+1} = \kappa u^k + (1-\kappa)(I - prox_{\frac{\mu}{\lambda}\|\cdot\|_1})(Bx + (1-\lambda BB^T)u^k)$

end While

$u = (\varepsilon I + B^T B)^{-1}B^T x - u^{k+1}$.

where $prox_{\frac{\mu}{\lambda}\|\cdot\|_1}x = (prox_{\frac{\mu}{\lambda}|\cdot|_1}x(1), prox_{\frac{\mu}{\lambda}|\cdot|_1}x(2), \cdots, prox_{\frac{\mu}{\lambda}|\cdot|_1}x(d))^T$, with $prox_{\frac{\mu}{\lambda}|\cdot|_1}x = \max\{|x| - \frac{\mu}{\lambda}, 0\}sign(x)$, and εI is a regularization term.

3.1.2.2 Application for BSS of Audio Signal Processing

In this part we change the algorithm in the form we can use to solve cancellation kernels. Since the model from previous ROF model is (3.1.3), and by changing the parameters, we should have

$$arg\min_{w}\{\frac{1}{2}\parallel w-f\parallel_2^2+\mu\parallel w\parallel_1\}. \tag{3.1.19}$$

If we assume $w=Au$, there is an equivalent problem

$$arg\min_{u}\{\frac{1}{2}\parallel Au-f\parallel_2^2+\mu\parallel Au\parallel_1\}. \tag{3.1.20}$$

Then we can transfer it to a similar problem,

$$arg\min_{u}\{\frac{1}{2}\parallel u-(A^TA)^{-1}A^Tf\parallel_2^2+\mu\parallel (A^TA)^{-1}A^TAu\parallel_1\},$$

i.e.,

$$arg\min_{u}\{\frac{1}{2}\parallel u-(A^TA)^{-1}A^Tf\parallel_2^2+\mu\parallel u\parallel_1\}, \tag{3.1.21}$$

where $B=I$.

By applying the algorithm in Section 3.1.2.1,

Initialize $u^0=0$

While $\parallel u^{k+1}-u^k\parallel_2>\epsilon$

$u^{k+1}=\kappa u^k+(1-\kappa)prox_{\frac{\mu}{\lambda}\|\cdot\|_1}((\varepsilon I+(A^TA))^{-1}A^Tf+(1-\lambda)u^k)$

end While

$u=(\varepsilon I+(A^TA))^{-1}A^Tf-u^{k+1}$.

where $prox_{\frac{\mu}{\lambda}\|\cdot\|_1}x=(prox_{\frac{\mu}{\lambda}|\cdot|_1}x(1),prox_{\frac{\mu}{\lambda}|\cdot|_1}x(2),\cdots,prox_{\frac{\mu}{\lambda}|\cdot|_1}x(d))^T$, with $prox_{\frac{\mu}{\lambda}|\cdot|_1}x=$

$\max\{|x| - \frac{\mu}{\lambda}, 0\} sign(x)$, εI is a regularization term.

The $prox_{\frac{\mu}{\lambda}\|\cdot\|_1} x$ function is a soft threshhold function which is the same as previously mentioned in split Bregman algorithm 3.1.1.1. The step

$$u = (\varepsilon I + (A^T A))^{-1} A^T f - u^{k+1} \tag{3.1.22}$$

also brings in error from a regularization term. At least this step is at the same level as (3.1.7) in split Bregman. We can compare this with the the SVD method applied in the later part.

3.1.2.3 Two Sources and Two Mixtures case

When we consider the determined two sources and two mixtures case, the matrix A is the same as in Section 3.1.1.2.

$$A = \begin{bmatrix} X_1(1) & X_1(2) & \cdots & \cdots & X_1(m-2) & X_1(m-1) & \eta \\ & X_1(1) & \cdots & \cdots & X_1(m-3) & X_1(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_1(1) & \cdots & X_1(m-d) & 0 \\ X_2(1) & X_2(2) & \cdots & \cdots & X_2(m-2) & X_2(m-1) & \eta \\ & X_2(1) & \cdots & \cdots & X_2(m-3) & X_2(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_2(1) & \cdots & X_2(m-d) & 0 \end{bmatrix}^T,$$

$$
\begin{aligned}
f &= [0 \quad 0 \quad \cdots \quad 0 \quad \eta]^T, \\
X_1 &= [X_1(1) \quad X_1(2) \quad \cdots \quad X_1(m-2) \quad X_1(m-1)]^T, \\
X_2 &= [X_2(1) \quad X_2(2) \quad \cdots \quad X_2(m-2) \quad X_2(m-1)]^T,
\end{aligned}
$$

where $m-1$ represents the length of the audio signal; f is in size $m \times 1$; d represents the length of the cancellation kernel; X_1 and X_2 are two mixtures.

3.1.2.4 Three Sources and Three Mixtures case

As three sources and three mixtures case, we have the same matrix A, and other corresponding parameters as in Section 3.1.1.3.

3.1.2.5 n Sources and n Mixtures case

In general, for determined n sources and n mixtures case, the matrix A and other parameters are defined as the same as in Section 3.1.1.4.

3.2 Quadratic Programming

3.2.1 Algorithm for Sparseness

The algorithm for the learning process is listed as followed. Assume the foreground sources $S_1, \ldots, S_L$ are inactive (silent) when $t \in [a+N, b]$.

Take $[s_1, e_1]$ and $[s_2, e_2]$ from $[a+N, b]$, $[s_1, e_1] \bigcap [s_2, e_2] = \emptyset$.

for $j = 1 : Q_1$

Randomly choose $[s_{1,j}, e_{1,j}] \subset [s_1, e_1]$ and $[s_{2,j}, e_{2,j}] \subset [s_2, e_2]$;

while $q_{nnz} \geqslant C + n_{sp}$ and $n_{sp} \geqslant 1$

(a) Solve $\{b_k\}$ on $[s_{1,j}, e_{1,j}]$;

(b) Find its error by equation (2.4.12) on $[s_{2,j}, e_{2,j}]$;

(c) Force $n_{sp} = 0.1 \times q_{nnz}$ smallest elements of $\{\mid b_k \mid\}$ be 0,

q_{nnz} = number of nonzero elements in $\{b_k\}$, $q_{nnz} = 0.9 q_{nnz}$;

end

Record positions of nonzero elements in $\{b_k\}$ with minimum

error;

end

Get the frequency of all the positions in $\{b_k\}$ with nonzero values, V_{freq}.

In our work, it turns out $Q_1 \geqslant 40$, since it does not work well in some cases if $Q_1 < 40$. For intervals $[s_{1,j}, e_{1,j}]$ and $[s_{2,j}, e_{2,j}]$, the length should be at least 1 second.

The algorithm for optimal cancelation kernels described as the second stage is as below.

Divide $[s_1, e_1]$ into Q_2 equal subintervals $I_1, \ldots, I_{Q_2}$;

$n_{V_{freq}}$ is the number of nonzero elements of V_{freq};

while $q \leqslant C$ and $q \leqslant n_{V_{freq}}$

(a) Take q highest frequency positions of $\{b_k\}$ by V_{freq} and

force the rest to be 0;

(b) Compute $\{b_k\}$ on $[s_1, e_1]$;

(c) Solve error on $I_1, \ldots, I_{Q_2}$ by (2.4.12) and their mean, Err_{mean};

(d) $q = 1.1q$;

end

Get $\{b_k\}$ with minimum Err_{mean};

By equation (1.2.6) and (2.3.7), foreground sources could be extracted.

Quadratic programming is a good tool to provide good estimate of cancellation kernels [61]. Our approach aims to bring sparseness to the cancellation kernels. It works theoretically which can be proved by the numerical experiments in the next chapter via the artificially mixed audios. However, for the real life recordings, it only works OK on two sources two mixtures case.

3.3 Singular Value Decomposition

Other than approximation methods mentioned above, we can also see the problem from another angle. We can solve the problem by direct method, such as singular value decomposition (SVD). Let us consider the following problem.

$$arg\min_{u}\{\frac{1}{2} \parallel Au - f \parallel_2^2\}$$

where $u = (u_1^T, u_2^T, \ldots, u_n^T)^T$, subject to $\sum_{j=1}^{n} u_j(1) = 1$.

3.3.1 Our Methodology

We indeed just need to solve the equation,

$$Au = f. \tag{3.3.23}$$

Since matrix A can rarely be a square matrix, in order to solve (3.3.23), we need to transform it to

$$A^T Au = A^T f. \tag{3.3.24}$$

As mentioned previously 2.4.62, A^TA is a symmetric positive definite matrix. Thus we apply SVD method to solve it.

$$VDV^T = A^TA. \tag{3.3.25}$$

Moreover, as a matrix A has its own SVD

$$A = U\Sigma V^T. \tag{3.3.26}$$

Matrix $D = \Sigma^T\Sigma$.

So

$$\begin{aligned} VDV^Tu &= A^Tf && (3.3.27)\\ \Longrightarrow u &= VD^{-1}V^TA^Tf && (3.3.28)\\ \Longrightarrow u &= VD^{-1}V^TV\Sigma U^Tf && (3.3.29)\\ \Longrightarrow u &= VD^{-1}\Sigma U^Tf. && (3.3.30) \end{aligned}$$

Theoretically (3.3.28) should be perfect for solving the problem (3.3.23). Unfortunately, the singular values of A^TA are not well distributed, which means the condition number of A^TA is very large. This results in a large error, sometimes may also make the system unsolvable.

As we can know from SVD, the diagonal elements of the matrix D are all the singular values of A^TA.

$$diag(D) = [\sigma_1, \sigma_2, \ldots, \sigma_n]^T, \tag{3.3.31}$$

in descending order. Similarly as previous in Section 3.1.2.2, we need to add a regularization

term to the diagonal of matrix D. Instead of using εI, we can adjust the diagonal of this relaxation matrix more specifically correspond to D.

Since the first k values of D, $\sigma_1, \sigma_2, \ldots, \sigma_k$ are the parts which will not cause a huge problem, starting from the $(k+1)$th item of the diagonal of D, we add a regularization term to each corresponding term in the diagonal of D. Then,

$$diag(D) := [\sigma_1, \sigma_2, \ldots, \sigma_k, \sigma_{k+1} + \varepsilon_{k+1}^2, \ldots, \sigma_n + \varepsilon_n^2]^T. \tag{3.3.32}$$

This will definitely give a more accurate computational result, compared with only adding a regularization term to all the items in the diagonal of D. The technique of adding a regularization term to all diagonal elements is equivalent as solving the inverse term $(\varepsilon I + A^T A)^{-1}$ of the regularization step (3.1.7) in split Bregman and the same part of (3.1.22) in proximity operator. It clearly indicates that split Bregman and proximity operator may take same amount of time to solve the inverse term.

This idea also inspires that we can use the special property of $D^{-1}\Sigma$ to reduce the computational time from above technique. Because $D = \Sigma^T \Sigma$, we can estimate that the diagonal elements of $D^{-1}\Sigma$ are around $\{\frac{1}{\sqrt{\sigma_k}}\}$ level. The following graph is a portion of the plot for $\{\frac{1}{\sigma_k}\}$.

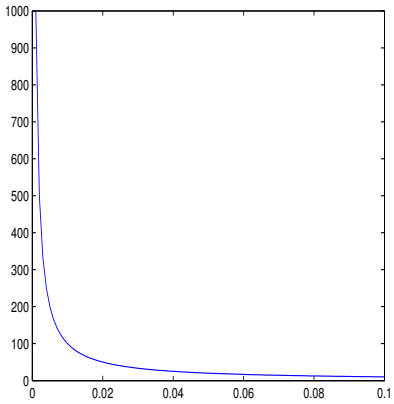

Figure 1: Plot of $\{\frac{1}{\sigma_k}\}$.

We can pre-select a positive number ϵ_0, then there exists a K such that $\sqrt{\sigma_{K+1}} \leqslant \epsilon_0 < \sqrt{\sigma_K}$. Then use the value of the tangent line to replace the left part of the plot.

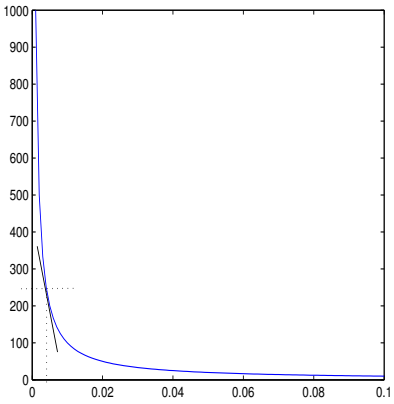

Figure 2: Plot of $\{\frac{1}{\sigma_k}\}$ with tangent line.

In our computation, we realize this technique by creating the tangent line to the plot of $\frac{1}{\sigma_k}$ at point $(\sigma_K, \frac{1}{\sigma_K})$. The corresponding slope at that point is $-\frac{1}{\sigma_K^2}$. By algebra, we can

find the equation of the line in point slope form

$$y = -\frac{1}{\sigma_K^2}x + \frac{2}{\sigma_K}. \tag{3.3.33}$$

Thus, we can replace the $\frac{1}{\sigma_j}$ element for $j \geqslant K+1$ in D^{-1} as $-\frac{1}{\sigma_K^2}\sigma_j + \frac{2}{\sigma_K}$, which also means we can replace σ_j with $\frac{1}{-\frac{1}{\sigma_K^2}\sigma_j + \frac{2}{\sigma_K}}$.

Since $\frac{1}{-\frac{1}{\sigma_K^2}\sigma_j + \frac{2}{\sigma_K}} = \frac{\sigma_K^2}{-\sigma_j + 2\sigma_K}$, and $\{\sigma_k\}_{k=1}^n$ decays dramatically fast, it can be derived as $\frac{\sigma_K^2}{-\sigma_j + 2\sigma_K} \approx \frac{\sigma_K}{2}$ for large j.

Based on our numerical examples, especially the real life recordings, the ratio $\frac{\sigma_k}{\sigma_1} \approx 10^{-4}$ when $150 \leqslant k \leqslant 200$ and the max delay is 230. We can simply replace σ_j with $\frac{\sigma_K}{2}$ when j is large enough. In our numerical results, we use $\sigma_j = \frac{\sigma_K}{2}$ when $j \geqslant \frac{5K}{4}$.

In general, we can redefine the diagonal elements in the matrix D

$$d_{j,j} = \begin{cases} \sigma_j & \text{if } j \leqslant K, \\ \frac{\sigma_K^2}{-\sigma_j + 2\sigma_K} & \text{if } K \leqslant j < \frac{5K}{4}, \\ \frac{\sigma_K}{2} & \text{otherwise,} \end{cases}$$

where $1 \leqslant j \leqslant n$.

3.3.2 Two Sources and Two Mixtures

In the determined two sources and two mixtures case, the matrix A and other corresponding parameters are the same as in Section 3.1.1.2.

$$A = \begin{bmatrix} X_1(1) & X_1(2) & \cdots & \cdots & X_1(m-2) & X_1(m-1) & \eta \\ & X_1(1) & \cdots & \cdots & X_1(m-3) & X_1(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_1(1) & \cdots & X_1(m-d) & 0 \\ X_2(1) & X_2(2) & \cdots & \cdots & X_2(m-2) & X_2(m-1) & \eta \\ & X_2(1) & \cdots & \cdots & X_2(m-3) & X_2(m-2) & 0 \\ & & \ddots & & & \vdots & \vdots \\ & & & X_2(1) & \cdots & X_2(m-d) & 0 \end{bmatrix}^T,$$

$$\begin{aligned} f &= [0 \quad 0 \quad \cdots \quad 0 \quad \eta]^T, \\ X_1 &= [X_1(1) \quad X_1(2) \quad \cdots \quad X_1(m-2) \quad X_1(m-1)]^T, \\ X_2 &= [X_2(1) \quad X_2(2) \quad \cdots \quad X_2(m-2) \quad X_2(m-1)]^T, \end{aligned}$$

where $m-1$ represents the length of the audio signal; f is in size $m \times 1$; d represents the length of the cancellation kernel; X_1 and X_2 are two mixtures.

3.3.3 Three Sources and Three Mixtures

In three sources and three mixtures case, the matrix A and other corresponding parameters are the same as in Section 3.1.1.3.

3.3.4 n Sources and n Mixtures

In general, for n sources and n mixtures case, the matrix A and other parameters are defined the same as in Section 3.1.1.4.

Chapter 4

Numerical Results

In this chapter, we provide several numerical experiments, most of them are from real life recordings. The artificial mixed audios are only used for quadratic programming to show that it works theoretically.

Based on the performance, we can conclude that our modified SVD is the best, and proximity operator is slightly better than split Bregman in computational time.

4.1 Numerical Example of Quadratic Programming

4.1.1 Numerical Examples of Two Sources and Two Mixtures

4.1.1.1 **Example** 1

In this example, the background music starts first. The desired foreground source is silent for a few seconds, and it is completely overwhelmed by the background music. Based on the work [61], we can require cancellation kernels to be non-negative, or only few of them being negative, in quadratic programming. In this way, we will not see some cancelation kernels with large magnitude in negative directions, which also make more sense, because sound sources are more reflected than absorbed in normal room condition. The mathematical theory behind this scenario definitely deserves further investigation, such as [65]. However, it is not a part we focuse on in this study.

In order to solve it, several parameters have taken the following values, the sample frequency $SF = 16000Hz$, the length of each cancellation kernel $d = 230$, the learning time interval $[14, 15]$, the percentage of non zero cancellation kernels $r = 15\%$. Moreover, 10% of all these non zero cancellation kernels are negative values to represent the absorptions. The figure below shows the two mixtures before separation and the normalized result after removal of the background music. In order to pick the important cancellation elements, the iterative selection has been run 80 times. The whole process takes 170 seconds.

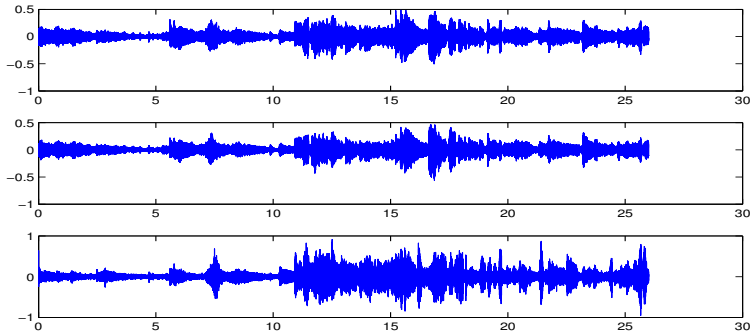

Figure 3: Recorded mixtures and result after removal of background music.

The following figure shows the value of the corresponding cancelation kernels.

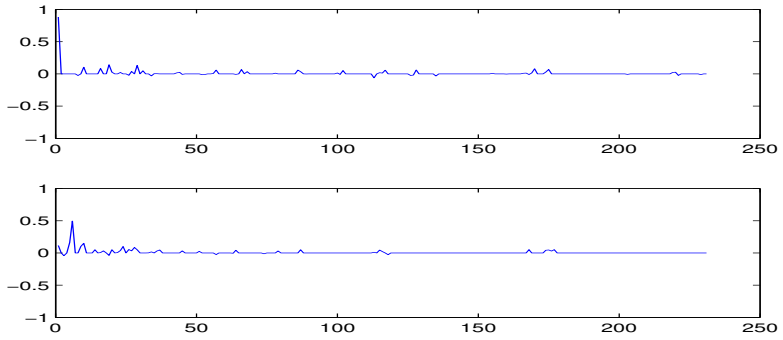

Figure 4: Cancelation kernels.

It looks like the result is not very good based on the third plot of the normalized result.

However, the truth is that the result is not as expected, even if we did separate the foreground source. The foreground source can be heard, and it is not overwhelmed by background music compared with original mixture. But there is still a defect that the background noise is a little loud. The reason is that the background music is so loud compared to the foreground source. If we need better result we have to use larger delay or longer learning time interval, which will sacrifice the computational time.

This can be shown in the spectrogram plot below by STFT and Hann window. The widow size is 512 with 500 overlaps. The first plot is the original mixture during the time interval $[15, 17]$, the other one is the result after removal the background during the same time interval. We can see from the figure that the spectrogram of the foreground source can be clearly seen from the second plot between the time interval $[16, 16.5]$, while it can be hardly seen in the first plot.

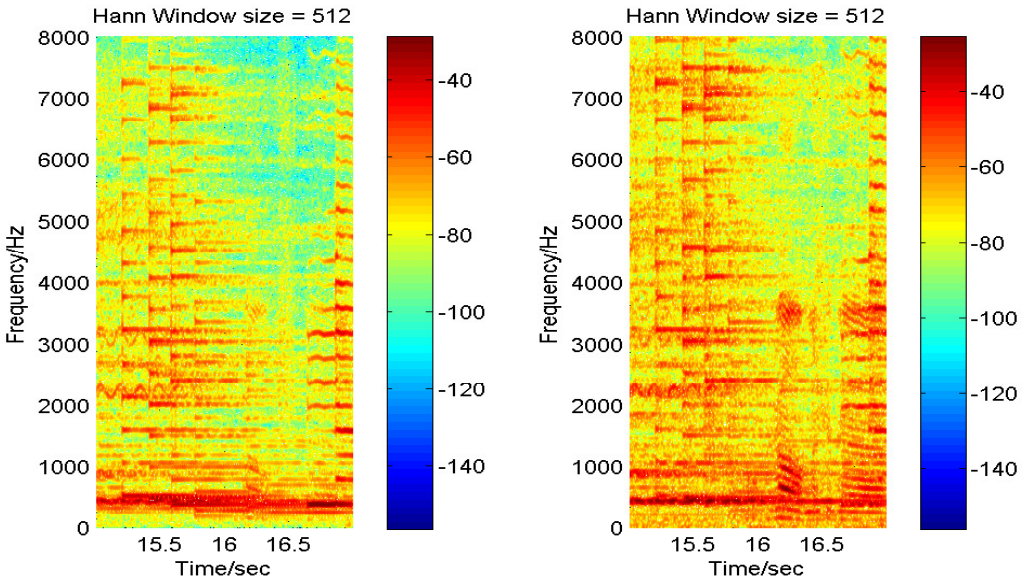

Figure 5: Spectrogram.

In order to evaluate the performance, a new quantity, relative error, is introduced, and

defined as,

$$\{\frac{2\times\sum_{a+N\leqslant t\leqslant b}[A^TF(t)-B^TG(t)]^2}{\sum_{a+N\leqslant t\leqslant b}[(A^TF(t))^2+(B^TG(t))^2]}\}^{\frac{1}{2}}.$$

The relative error of this example is 23.34%.

4.1.1.2 Example 2

In this example, the background music starts first. The desired foreground source is silent for a few seconds, and it is a little weaker than the background music.

Same values for the parameters are taken in the computation, the sample frequency $SF = 16000Hz$, the length of each cancellation kernel $d = 230$, the learning time interval $[14, 15]$, percentage of non zero cancellation kernels $r = 15\%$, and 10% of these cancellation kernels are negative. The iterative selection has been run 80 times.

The figure below shows the two mixtures before separation and result after removal of the background music. It takes 160 seconds to complete the computation.

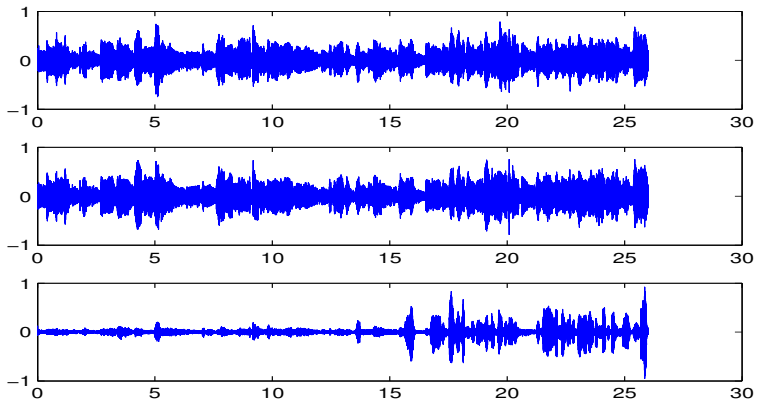

Figure 6: Recorded mixtures and result after removal of background music.

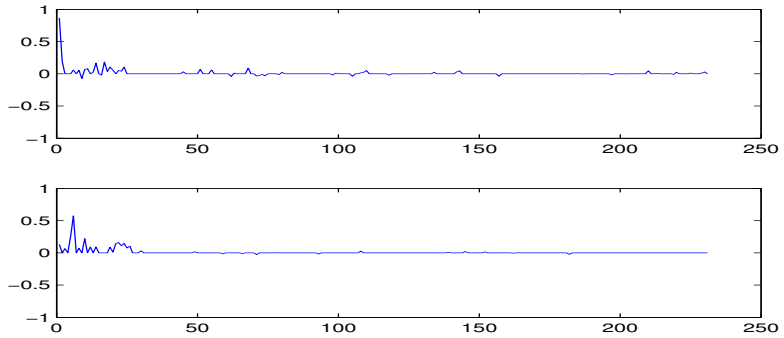

Figure 7: Cancelation kernels.

The corresponding spectrogram of time interval $[15, 17]$ when the foreground source started is also shown below by applying STFT and Hann window function. Same as the previous example, the widow size is 512 with 500 overlaps. The first plot is for the original mixture, and the other one is for the removal result. In the first one it can hardly tell the difference between the background music and the foreground source. In the second plot, it clearly shows the foreground source started between time interval $[15.5, 16]$, while the background music has been removed to a large extend. We can see this fact from the spectrogram of the foreground source in the second plot.

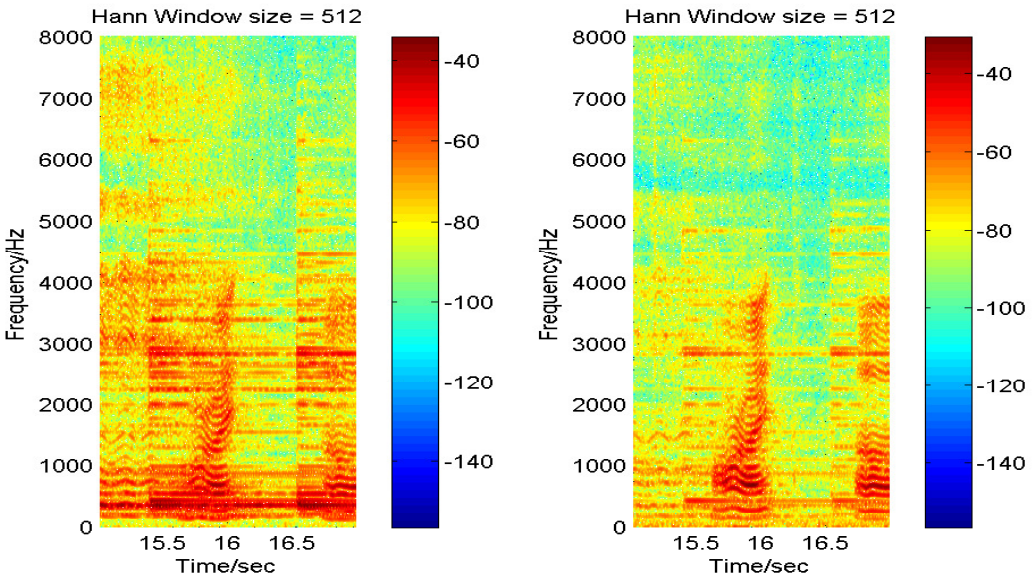

Figure 8: Spectrogram.

The relative error of this example is 22.48%.

4.1.1.3 Example 3

In this example, the audio file is artificially mixed, not from real recordings. There are two sources being used. One is a background music, and the other one is a human speech.

By randomly choosing the sample frequency $SF = 16000Hz$, the length of each cancellation kernel $d = 230$, the percentage of non zero cancellation kernels $r = 15\%$, and first element of the cancellation kernel non-zero, and 10% of these cancellation kernels are negative, it applies the convolution formula to mix the artificial recordings.

Similar as *Example* 1, the backgound music starts first. The desired foreground source is silent for a few seconds, and is not overwhelmed by the background music.

All the parameters are taken the same values for solving the system, while the learning time interval used is $[9, 10]$. The figure below shows the two mixtures before separation and

result after removal of the background music without any normalization.

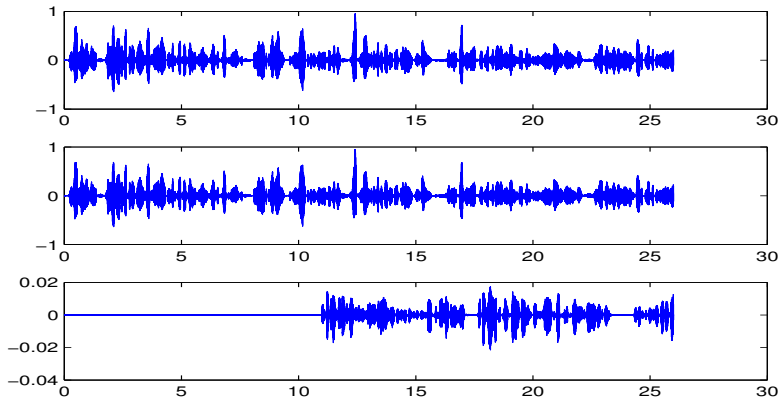

Figure 9: Artificial mixtures and result after the removal of background source.

Clearly, the background music has been successfully completely removed. It shows the fact that this methodology definitely works theoretically.

The following figure shows the value of the corresponding cancelation kernels.

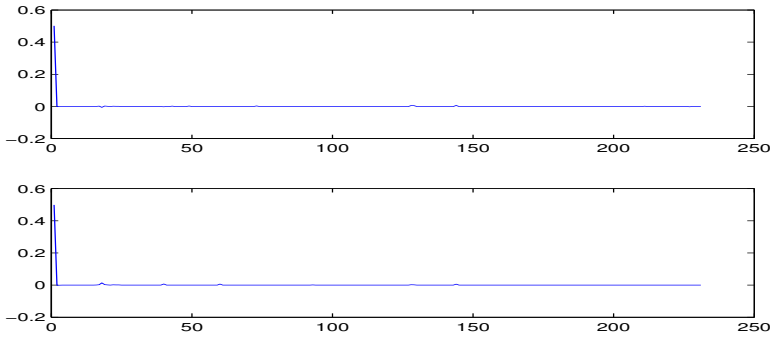

Figure 10: Cancelation kernels.

The cancellation kernels are not as ideal as expected. It only needs to pick a few elements from cancellation kernels to solve the system. This is not a surprise, since the artificial mixtures are simply formed by using the convolution notation. The optimization problem can be solved by other estimations.

The relative error in this example is less than 0.19%.

4.1.2 Numerical Examples of Three Sources and Three Mixtures

4.1.2.1 Example 4

In this example, three single channel speech signals are artificially mixed to make 3 different mixtures. In each of them, the third source S_3 is silent for a few seconds, and it is overwhelmed by other backgrounds.

The artificial mixing process is similar to previous example in 4.1.1.3. The same values of parameters are taken for producing mitures, the sample frequency $SF = 16000Hz$, the length of each cancellation kernel $d = 230$, percentage of non zero cancellation kernels $r = 15\%$, and first element of the cancellation kernel non-zero, and 10% of these cancellation kernels are negative.

The parameters for solving the system are the same as the mixing process, while the time interval $[9, 10]$.

The figure below shows that three mixtures before separation and result after the removal of two background sources.

g

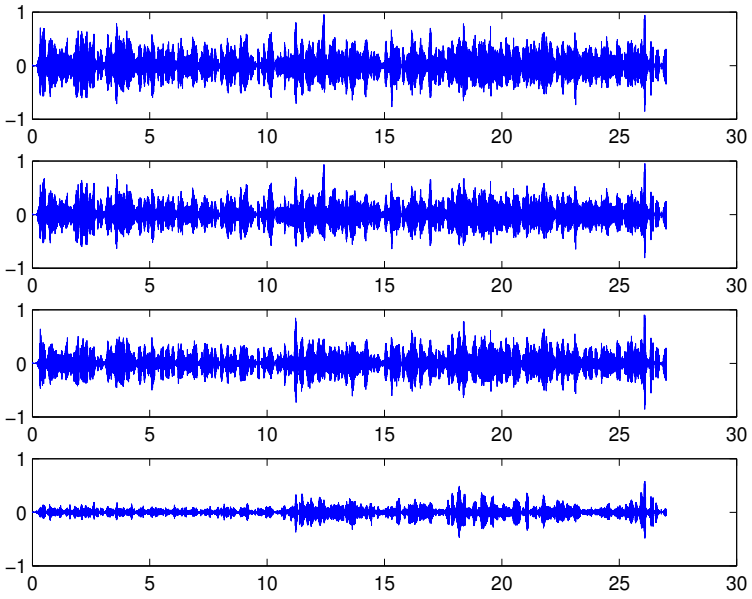

Figure 11: 3 Artificial mixtures and result after the removal of background source.

Based on the fourth plot, the marjority energy of the first two backgrounds is clearly removed, though there is still some small portion left.

The figure below shows the cancelation kernels.

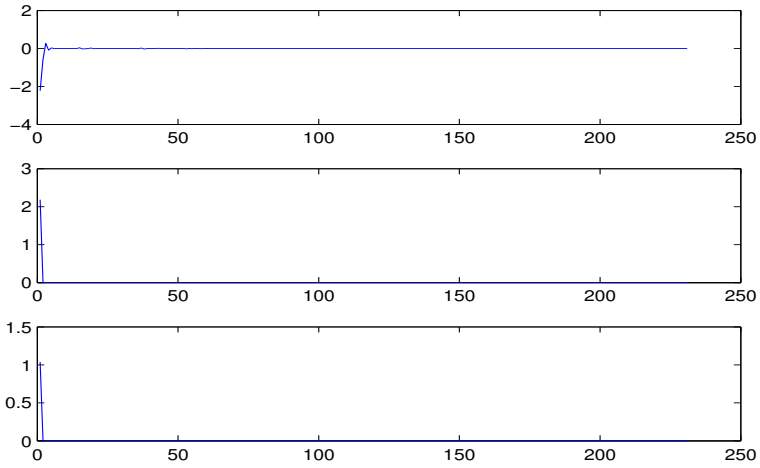

Figure 12: Cancelation kernels.

The cancellation kernels are not as ideal as expected. It only needs to pick a few elements from cancellation kernels to provide the solution. This is not a surprise, since the artificial mixtures are only simply formed by using the convolution notation. The optimization problem can be solved by other estimations.

The relative error is 3.3%. It takes 80 seconds to finish eliminating background signals.

4.2 Numerical Examples of SVD Method

In this part, we apply our modified SVD method developed in Section 3.3 to the same examples of real life recordings. In the following numerical examples, it will demonstrate that our modified SVD is a very fast algorithm and provides good results.

4.2.1 Two Sources Two Mixtures case

4.2.1.1 Example 5

By using the same audio recordings from Example 1 in section 4.1.1.1, we solve the problem by using SVD method.

For the corresponding parameters, sample frequency $SF = 16000Hz$, length of each cancellation kernel $d = 230$, the time interval $[14, 15]$, ϵ_0 is the value $\sqrt{\sigma_K}$ when $\frac{\sigma_{K+1}}{\sigma_1} \leqslant 10^{-4} \leqslant \frac{\sigma_K}{\sigma_1}$.

Then we have the following result as shown in the figure. It only takes 0.158 to solve the SVD process to the sigular value σ_K with relative error 5.1%.

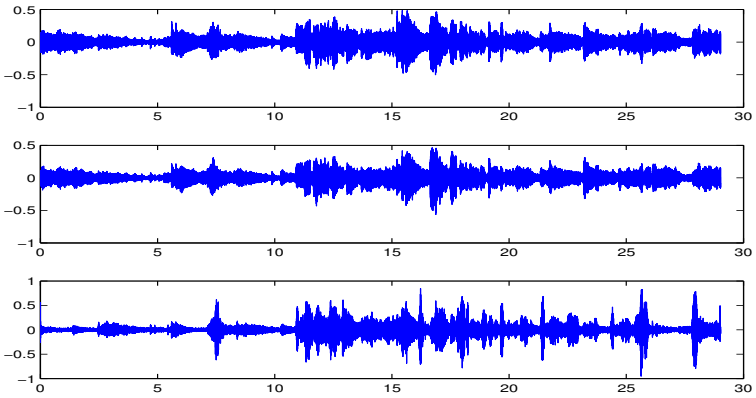

Figure 13: Recorded mixtures and results after the removal of background source.

The corresponding spectrogram of time interval $[15, 17]$ when the foreground source started is also shown below by applying STFT and Hann window function. Same as the previous example, the widow size is 512 with 500 overlaps.

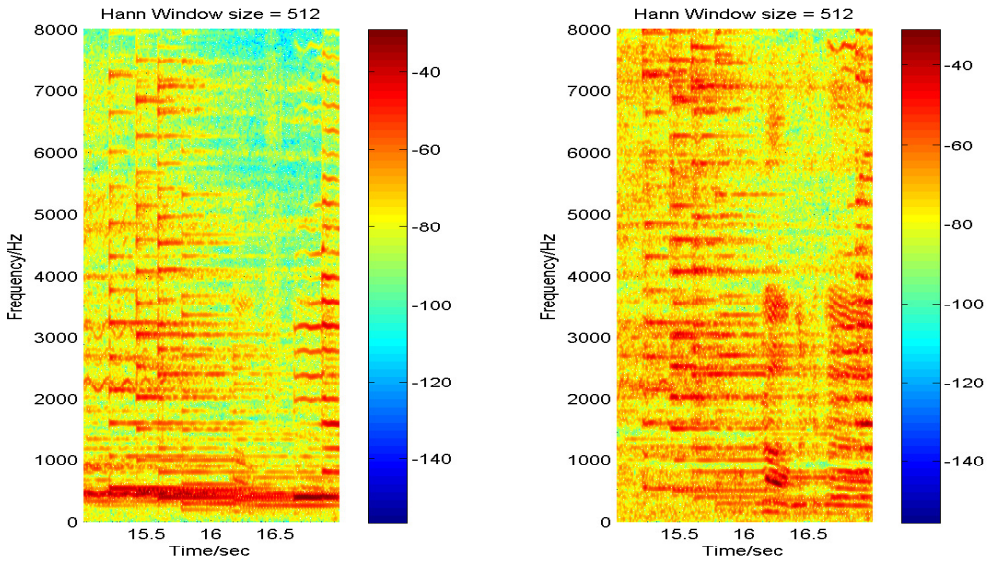

Figure 14: Spectrogram.

The first one is for the original mixture, while the second one is for the result. From the second plot, the spectrogram of the foreground source can be clearly seen between the time interval $[16, 16.5]$, while it can hardly seen in the first plot. Compared with Example 1, a lot low frequency energy has been removed compared with the second plot in Figure 5, especially for the frequencies between $[0, 500]$. So there is definitely improvement from Example 1 to Example 5.

4.2.1.2 Example 6

Using the same audio recordings from Example 2 in section 4.1.1.2, we solve the problem by using SVD method.

For the corresponding parameters, sample frequency $SF = 16000Hz$, length of each cancellation kernel $d = 230$, the time interval $[14, 15]$, ϵ_0 is the value $\sqrt{\sigma_K}$ when $\frac{\sigma_{K+1}}{\sigma_1} \leqslant 10^{-4} \leqslant \frac{\sigma_K}{\sigma_1}$.

Then we have the following result as shown in the figure. It takes 0.172 to solve the SVD process to the sigular value σ_K with relative error 6.02%

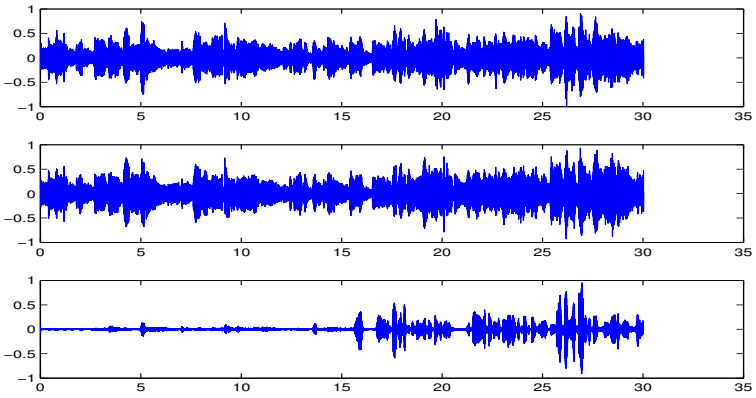

Figure 15: Recorded mixtures and results after the removal of background source.

The following part uses STFT and Hann window function to check the spectrogram of the mixture and result between time interval $[15, 17]$. Same as the previous example, the widow size is 512 with 500 overlaps.

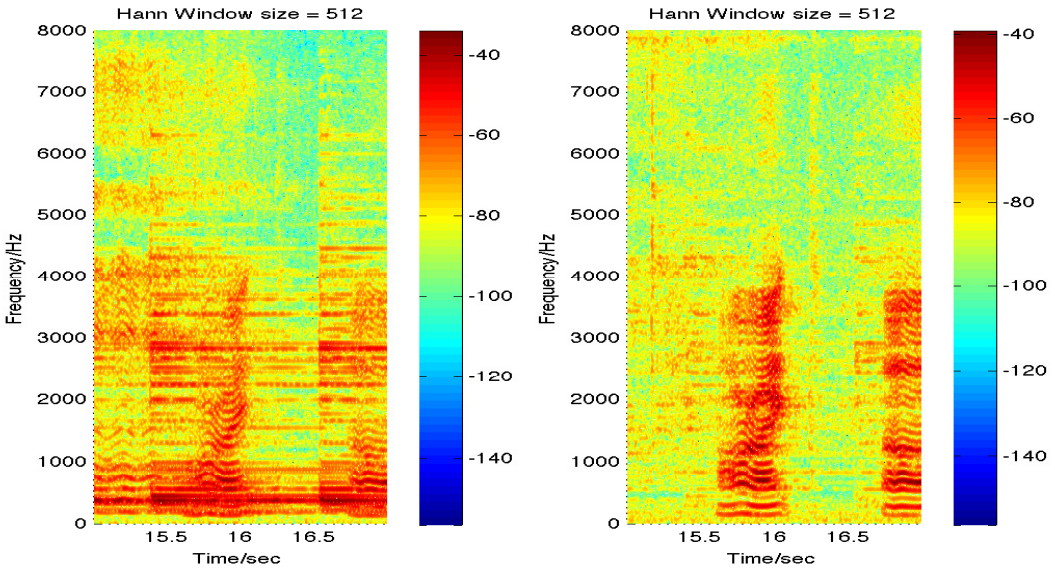

Figure 16: Spectrogram.

The first plot is for the original mixture, while the second one is for the result. From the second plot, the spectrogram of the foreground source can be clearly seen between the time interval $[15.5, 16]$, while the majority of the first source has been eliminated. Compared with Example 2, a lot low frequency energy has been removed compared with the second plot in Figure 8, especially for the frequencies between $[0, 500]$. So there is definitely improvement from Example 2 to Example 6.

4.2.2 Three Sources Three Mixtures

4.2.2.1 Example 7

In this example, the first source is music; the other two sources are speeches of two people. The second source starts a few seconds later than the first source. And the third source starts a few seconds later than the second one. We use our modified SVD method to remove first tow sources and keep the third source.

For the corresponding parameters, sample frequency $SF = 16000Hz$, length of each cancellation kernel $d = 230$, the time interval $[18, 19]$, ϵ_0 is the value $\sqrt{\sigma_K}$ when $\frac{\sigma_{K+1}}{\sigma_1} \leqslant 10^{-4} \leqslant \frac{\sigma_K}{\sigma_1}$.

The following figures are the original audio signals and result from the above method.

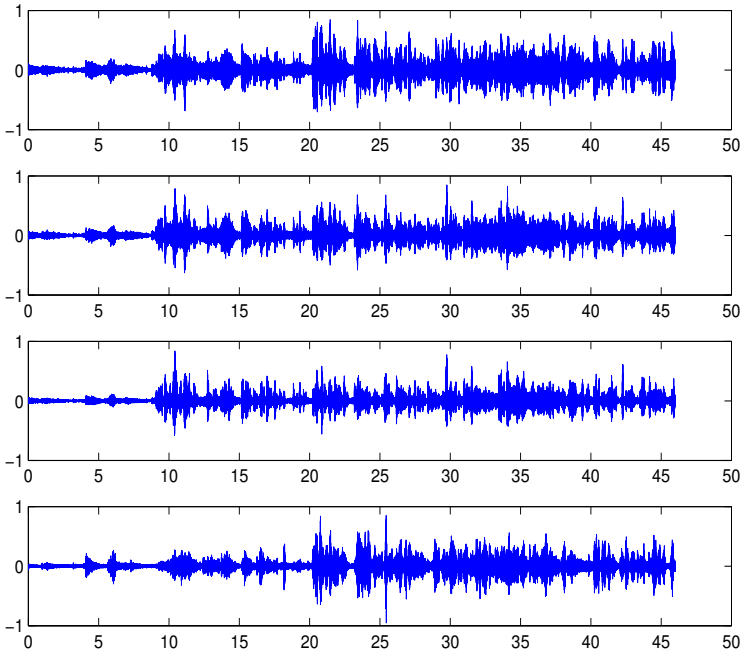

Figure 17: Recorded mixtures and results after the removal of background sources.

Clearly, the first two sources have been largely removed in the fourth plot. Since we normalized the result to the maximum of 0.95, it may look like that there is still a large portion of first two sources in the result. In fact, the result is very good. The relative error is 11.2%.

4.2.2.2 **Example** 8

In this example, all three sources are human speeches. The second source starts a few seconds later than the first one. And the third one starts a few seconds later than the second one. We use the SVD method to remove the first two sources and extract the third one.

For the corresponding parameters, sample frequency $SF = 16000Hz$, length of each cancellation kernel $d = 230$, the time interval $[14, 15]$, ϵ_0 is the value $\sqrt{\sigma_K}$ when $\frac{\sigma_{K+1}}{\sigma_1} \leqslant$

$10^{-4} \leqslant \frac{\sigma_K}{\sigma_1}$.

The following figures are the original audio signals and results from this method.

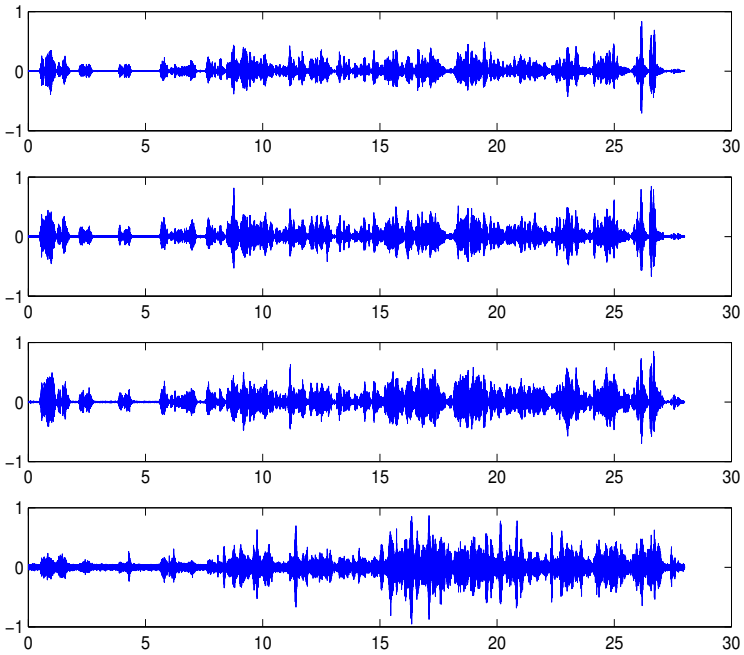

Figure 18: Recorded mixtures and results after the removal of background sources.

Based on above plot, the first two sources have been largely removed in the fourth plot. Since we normalized the result to the maximum of 0.95, it may look like that there is still a large portion of first two sources in the result. In fact, the result is very good. The relative error is 14.6%.

In general, our modified SVD provides very good results.

4.3 Numerical Examples of Split Bregman Iteration

In the following examples, we choose $\mu = \epsilon = 10^{-3}$, $\eta = 1$, $\lambda = 2\mu$ same as [40].

4.3.1 Two sources two mixtures case

4.3.1.1 **Example** 9

By using the same audio recordings from Example 1 in section 4.1.1.1, we have the following result as shown in the Figure 19.

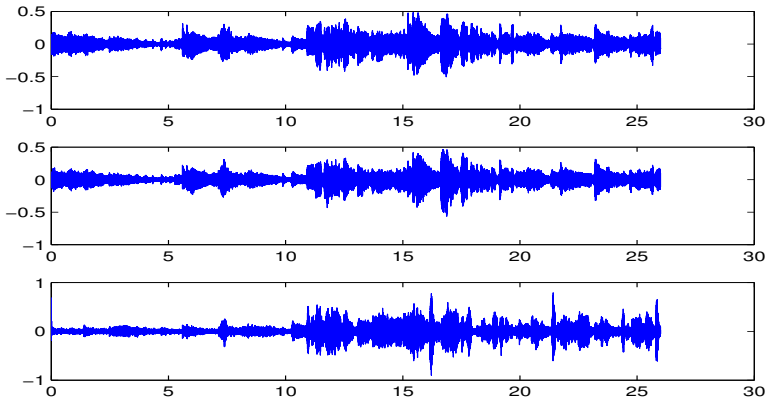

Figure 19: Recorded mixtures and result after the removal of background music.

The relative error for this example is 4.5%.

4.3.1.2 **Example** 10

By using the same audio recordings from Example 2 in section 4.1.1.2, we have the following result as shown in the Figure 20.

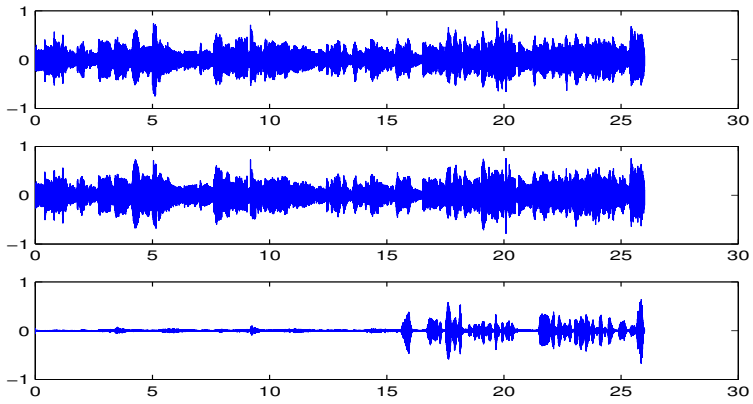

Figure 20: Recorded mixtures and result after the removal of background music.

The relative error for this example is 4.1%.

4.3.2 Three sources three mixtures

4.3.2.1 **Example** 11

In this example, we use the same audio files from Example 7 in section 4.2.2.1. There are two ways to remove the foreground sources. One is to use the split Bregman iteration directly for the mixtures.

The other is to remove the first source by two groups of different two mixtures like what we use for two sources two mixtures case. Then use the corresponding results to remove the second source.

The following plots in Figure 21 are the original audio signals and the results from the two ways mentioned above.

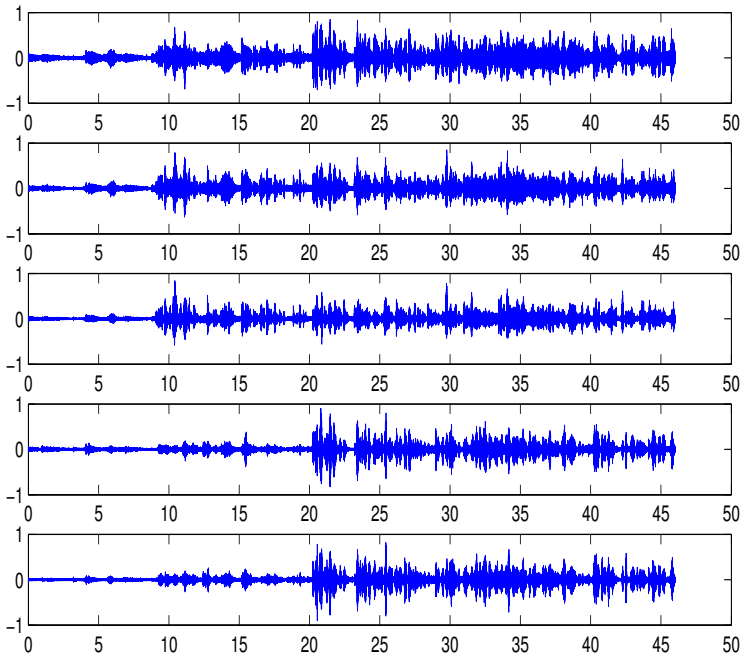

Figure 21: Recorded mixtures and results after the removal of background sources.

4.3.2.2 **Example** 12

In this example, we use the same audio files from Example 8 in section 4.2.2.2. We use the similar two ways listed in Example 11 in section 4.3.2.1 to solve the problem.

The following plots in Figure 22 are the original audio signals and results from the above two approaches mentioned in Example 11.

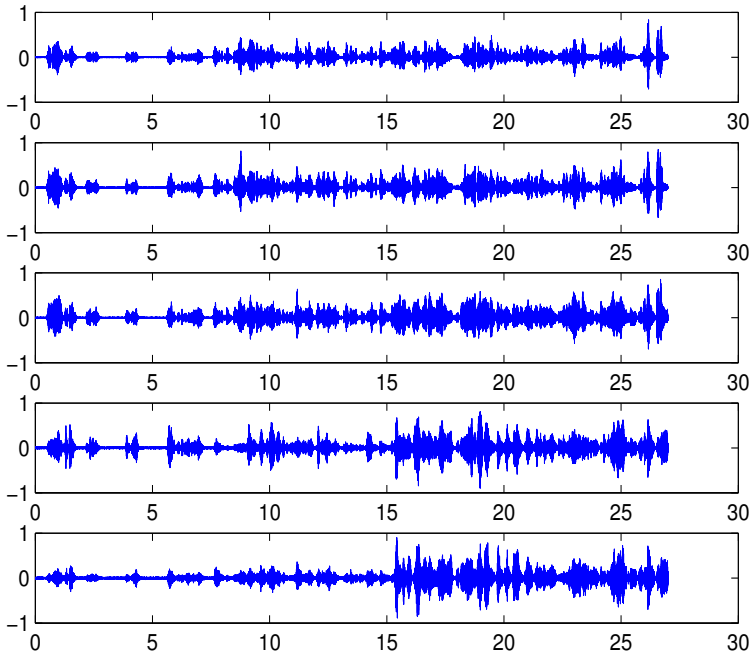

Figure 22: Recorded mixtures and results after the removal of background sources.

4.4 Numerical Examples of Proximity Operator Method

In the following examples, we choose $\kappa = 0$ and $\lambda = \frac{1}{4}$ same as [37].

In general, the proximity operator is slightly better than split Bregman, but the computational time is slower than our modified SVD. It definitely deserves more investigations for more reasonis behind this phenomena.

4.4.1 Two sources two mixtures case

4.4.1.1 **Example** 13

By using the same audio recordings from Example 1 in section 4.1.1.1, we solve the problem by using proximity operator method. Then we have the following result as shown in the Figure 23.

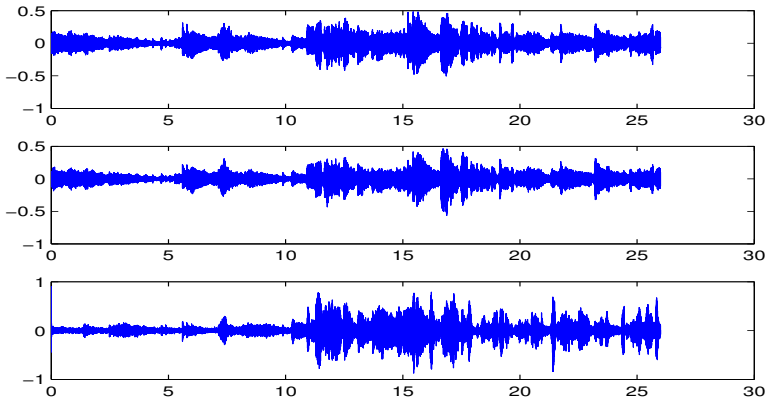

Figure 23: Recorded mixtures and results after the removal of background source.

The relative error for this example is 4.5%.

4.4.1.2 **Example** 14

Using the same audio recordings from Example 2 in section 4.1.1.2, we solve the problem by using proximity operator method. Then we have the following result as shown in the Figure 24.

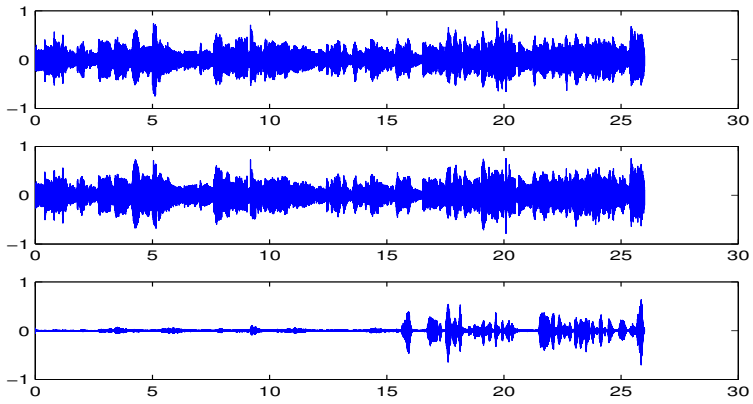

Figure 24: Recorded mixtures and results after the removal of background source.

The relative error for this example is 3.9%.

4.4.2 Three Sources Three Mixtures

4.4.2.1 **Example** 15

In this example, we use the same audio files from Example 7 in Section 4.2.2.1. There are two methods to remove the foreground sources. One is to use the proximity operator directly.

The other is to remove the first source by two groups of different two mixtures like what we use for two sources two mixtures case by proximity operator method. Then use the corresponding results to remove the second source.

The following plots in Figure 25 are the original audio signals and results from the above two ways.

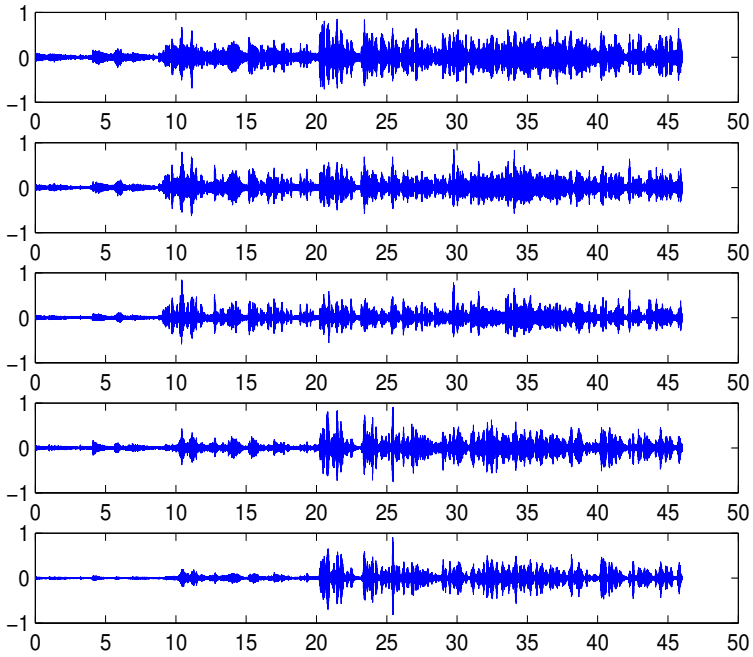

Figure 25: Recorded mixtures and results after removal of background sources.

4.4.2.2 **Example** 16

In this example example, we use the same audio files from Example 8 in Section 4.2.2.2. There are two methods to remove the foreground sources. One is to use the proximity operator directly on three mixutres to solve the problem. two groups of different two mixtures like what we use for two sources two mixtures case by proximity operator method. Then use the corresponding results to remove the second source.

The following plots in Figure 26 are the original audio signals and results from the above two ways.

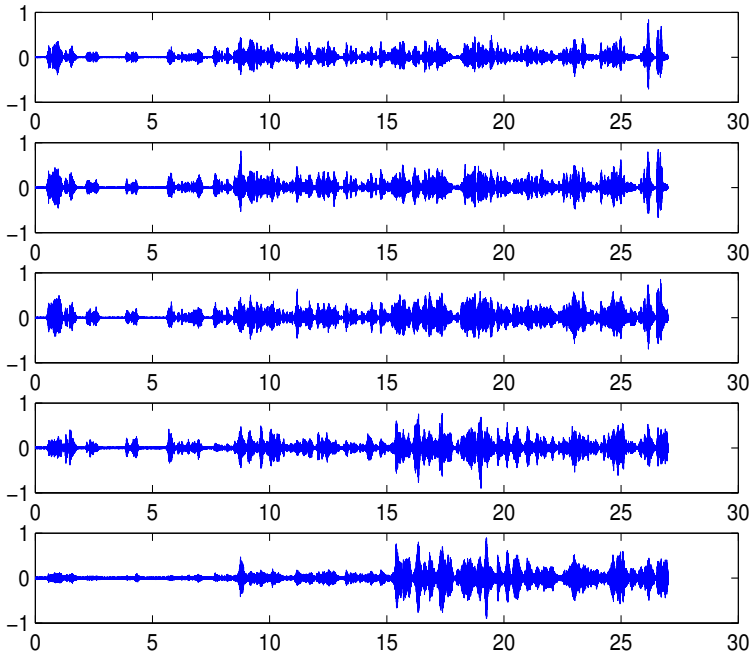

Figure 26: Recorded mixtures and results after the removal of background sources.

4.5 Summarization

We use a system with CPU Intel Core i7 − 4770 and 16 GB DDR3 800 MHz memory to run all the above numerical examples.

As listed above, there is a big performance difference between quadratic programming with sparseness and the modified SVD. Clearly, the performance of SVD method is better than the quadratic program. Moreover, based on the analysis of Section 3.3, the SVD method is at least at the same level of computaitonal time compared with proximity operator method and split Bregman.

In our numerical results, our modified SVD does have the best performance. It is fastest with the same level of relative error compared with proximity operator and split Bregman,

especially when we consider a small length of each cancellation kernel $d = 230$. Proximity operator method is slightly better than split Bregman iteration in terms computational time but they do have the same level of relative error.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Alliney. *A property of the minimum vectors of a regularizing functional defined by mean of the absolute norm*. IEEE Transactions on Signal Processing, 45 1997, pp. 913-917.

[2] S. I. Amari, A. Cichocki, and H. H. Yang. *A new learning algorithm for blind signal separation*. Advances in neural information processing systems (1996): 757-763.

[3] S. Araki, S. Makino, H. Sawada, and R. Mukai. *Reducing musical noise by a fine-shift overlap-add method applied to source separation using a time-frequency mask*. Proc. ICASSP, III, 81-84, 2005.

[4] J. Allen, and L. Rabiner. *A unified approach to short-time Fourier analysis and synthesis*. Proceedings of the IEEE, Vol. 65, NO. 11, Nov. 1977.

[5] L. Bregman. *The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex optimization*. USSR Computational Mathematics and Mathematical Physics, 1967, 7, pp. 200-217.

[6] A. Bell, and T. Sejnowski. *An information-maximization approach to blind separation and blind deconvolution*. Neural Computation, 1995, 7(6), pp. 1129-1159.

[7] R. Blackman, and J. Tukey. *The measurement of power spectra from the point of view of communications Engineering*. 1958, Dover Publication Inc., New York, pp. 170.

[8] J. Cai, S. Osher, and Z. Shen. *Linearized Bregman iterations for compressed sensing*. Mathematics of Computation 78.267 (2009): 1515-1536.

[9] J. Cai, S. Osher, and Z. Shen. *Split Bregman methods and frame based image restoration*. Multiscale Model. Simul., Vol. 8(2), 2010, pp. 337-369.

[10] J.F. Cardoso. *Sources separation using higher order moments*. Proc. Internat. Conf. Acoust. Speech Signal Process., Glasgow, 1989, pp. 2109-2112.

[11] J.F.Cardoso, and A. Souloumiac. *Blind beamforming for non-Gaussian signals*. IEE Proceedings F (Radar and Signal Processing). Vol. 140. No. 6. 1993.

[12] A. Cichocki and S. Amari. *Adaptive blind signal and image processing: learning algorithms and applications*. Wiley, 2002.

[13] S. Choi, A. Cichocki, H. Park and S. Lee. *Blind scource separarion and independent component analysis: A review*. Neural Inform. Process. Lett. Rev., 6, 2005, pp. 1-57.

[14] P. Combettes. *Convexité et signal*. Proc. Congres de Mathématiques Appliquées et Industrielles SMAI. Vol. 1, 2001.

[15] P. Combettes, and V. Wajs. *Signal recovery by proximal forward-backward splitting*. Mutiscale Model. Simul., 2005, 4, pp. 1168-1200.

[16] P. Combettes, and J. Pesquet. *Proximal splitting methods in signal processing. Fixed-point algorithms for inverse problems in science and engineering*. Springer New York, $185 - 212$, 2011.

[17] P. Comon. *Independent component analysis, A new concept*. Signal Processing 36, 1994, pp. 287-314.

[18] P. Comon, and C. Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[19] T. Cover, and J. Thomas. *Elements of information theory*. New York: Wiley (1991).

[20] N. Delfosse, and P. Loubaton. Adaptive blind separation of independent sources: a deflation approach. Signal Processing, 45(1995), 5983.

[21] G. Giannakis, Y. Inouye and J.M. Mendel. *Cumulant based identification of multichannel moving average models*. IEEE Automat. Control, Vol. 34, July 1989, pp. 783-787.

[22] T. Goldstein, and S. Osher. *The split Bregman method for L_1 regularized problems*. SIAM Journal Imaging Sci., Vol. 2, April 2009, pp. 323-343.

[23] J. Herault, and C. Jutten. *Space or time adaptive signal processing by neural network models*. AIP Conf. Proc. 151, 206(1986).

[24] J. Hughes, D. Mao, D. Rockmore, Y. Wang, Q. Wu. *Empirical mode decomposition analysis for visual stylometry*. IEEE Trans Pattern Anal Mach Intell, Nov. 2012, 34(11), pp. 2147-57.

[25] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley, New York, 2001.

[26] A. Hyvärine, and E. Oja. *Independent Component Analysis: Algorithms and Applications*. Neural Networks, $13(4-5)$, 2000, pp. 411-430.

[27] Y. Inouye, and T. Matsui. *Cumulant based parameter estimation of linear systems*. Proc. Workshop Higher-Order Spectral Analysis, Vail, Colorado, June 1989, pp. 180-185.

[28] S. Johnson, and M. Frigo. *A modified split-radix FFT with fewer arithmetic operations*. Signal Processing, IEEE Transactions on 55.1 (2007): 111-119.

[29] C. Jutten, and J. Herault. *Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture*. Signal Processing 24, 1991, pp. 1-10.

[30] A. Jourjine, S. Rickard and Ö. Yilmaz. *Blind separation of disjoint orthogonal signals: demixing N sources from 2 mixtures*. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Jun 2000, pp. 2985-2988.

[31] D. D. Lee, and H. S. Seung. *Learning of the parts of objects by non-negative matrix factorization*. Nature, $401 : 788-791$, 1999.

[32] J. Liu, J. Xin, Y. Qi, and F. Zeng. *A time domain algorithm for blind separation of convolutive sound mixtures and L_1 constrained minimization of cross correlations*. Commun. Math. Sci., Vol. 7, No. 1, 2009, pp. 109-128.

[33] J. Liu, J. Xin, and Y. Qi. *A dynamic algorithm for blind separation of convolutive sound mixtures*. Neurocomputing 72.1 (2008): 521-532.

[34] J-J. Moreau. *Fonctions convexes duales et points proximaux dans un espace hilbertien*. CR Acad. Sci. Paris Sér. A Math 255: 2897-2899, 1962.

[35] J-J. Moreau. *Propriétés des applications "prox"*. CR Acad. Sci. Paris 256 (1963), pp. 1069-1071.

[36] J-J. Moreau. *Proximité et dualité dans un espace hilbertien*. Bulletin de la Société mathématique de France 93 (1965), pp. 273-299.

[37] C. Micchelli, L. Shen, and Y. Xu. *Proximity algorithms for image models: denoising*. Inverse Problems, IOP Science, 2011, Vol. 27.

[38] C. Micchelli, L. Shen, Y. Xu, and X. Zeng. *Proximity algorithms for the L1/TV image denoising model*. Adv. Comput. Math., 2013, 38(2), pp. 401-426.

[39] J. Liu, J. Xin, Y. Qi, and F. Zheng. *A time domain algorithm for blind separation of convolutive sound mixtures and L1 constrained minimization of cross correlations*. Commun. Math. Sci., 2009, Vol. 7, No. 1, pp. 1-247.

[40] W. Ma, M. Yu, J. Xin, and S. Osher. *Reducing musical noise in blind source separation by time-domain sparse filters and split Bregman method*. Interspeech 2010, pp. 402-405, Sept 26-30, 2010, Chiba, Japan.

[41] W. Ma, M. Yu, J. Xin, and S. Osher. *A convex model and L_1 minimization for musical noise reduction in blind source separation*. Commun. Math. Sci., 2012 Vol. 10, No. 1, pp. 223-238.

[42] D. Mao, Y. Wang, and Q. Wu. *A new approach for analyzing physiological time series*. Jul. 2010.

[43] J. Nadal, and N. Parga. *Non-linear neurons in the low noise limit: a factorial code maximizes information transfer*. Network, 5 (1994), 565581.

[44] A. Nuttall. *Some windows with very good sidelobe behavior*. IEEE Transaction on Acoustics, Speech, and Signal Processing, Vol. Assp 29, NO. 1, Feb, 1981.

[45] Z. Opial. *Weak convergence of the sequence of successive approximations for nonexpansive mappings*. Bull. Amer. Math. Soc., 1967, Vol. 73, Num. 4, pp. 591-597.

[46] V. Oppenheim. *Speech spectrograms using the fast Fourier transform*. IEEE Spectrum, vol. 7, Aug. 1970, pp. 57-62.

[47] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. *An iterative regularization method for total variation-based image restoration*. Multiscale Model. Simul., Vol. 4, N.O. 2, 2005.

[48] S. Osher, Y. Mao, B. Dong, W. Yin. *Fast linearized Bregman iteration for compressive sensing and sparse denoising*. arXiv preprint arXiv:1104.0262 (2011).

[49] S. Osher, and L. I. Rudin. *Feature oriented image enhancement using shock filters*. SIAM J. Num. Anal. 27(1990)919.

[50] A. Papoulis, *Probability, random variables, and stochastic processes (3rd ed.)*. New York: McGraw-Hill 1991.

[51] M. Pinsky. *Introduction to Fourier Analysis and Wavelets*. American Mathematical Society, 2002.

[52] L. Rudin. *Images, numerical analysis of singularities and shock filters*. Caltech, C.S. Dept. Report 1987 ♯ TR. 5250 : 87.

[53] L. Rudin, and S. Osher. *Total variation based image restoration with free local constraints*. Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference. Vol. 1. IEEE, 1994, pp.31-35.

[54] L. Rudin, S.Osher and E. Fatemi. *Nonlinear total variation based noise removal algorithms*. *Physica D*, 60(1992) pp. 259-268.

[55] S. Rickard, and Ö. Yilmaz. *On the w-disjoint orthogonality of speech*. Proceedings of the IEEE international conference on acoustics, speech, and signal processing, May 2002, pp. 529-532.

[56] D.Strong, and T. Chan. *Edge-preserving and scale-dependent properties of total variation regularization*. Inverse Problems, 19(2003), pp. $S165$-$S187$.

[57] F. Bach and M. Jordan. *Beyond independent components: trees and clusters*. Journal of Machine Learning Research, 4 : 1205-1233, 2003.

[58] L. Vese, and S. Osher. *Modeling testures with total variation minimization and oscillatory patterns in image processing*. Journal of scientific computing, 19. 1-3 (2003), pp. 553-572.

[59] M. Van Segbroeck, and H. Van hamme. *Robust speech recognition using cepstral domain missing data techniques and noisy masks*. IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, 2004.

[60] Y. Wang, Ö. Yilmaz, and Z. Zhou. *Phase aliasing correction for robust blind source separation using DUET*. Applied and Comput. Harm. Anal., Vol. 35, Issue 2, Sept. 2013, pp. 341-349

[61] Y. Wang, and Z. Zhou. *Source extraction in audio via background learning*. Inverse problems and imaging 7.1 (2013): 283.

[62] B. Yang. *A study of inverse short time fourier transform.* IEEE International Conference on Acoustics, Speech and Signal Processing, 2008.

[63] M. Yu, W. Ma, J. Xin, and S. Osher. *Convexity and fast speech extraction by split Bregman method.* Interspeech 2010, pp. 398-401, Sept 26-30, 2010, Chiba, Japan.

[64] M. Yu, W. Ma, J. Xin, and S. Osher. *Multi-channel l_1 regularized convex speech enhancement model and fast computation by the split Bregman method.* Audio, Speech and Language Processing, IEEE Transactions on 20(2), (2012): 661-675.

[65] Z. Zeng. *The numerical greatest common divisor of univariate polynomials.* Randomization, relaxation, and complexity in polynomial equation solving, Providence, RI. Contemp. Math. Amer. Math. Soc 556 (2011): 187-217.