Turning A Duet Into Two Solos

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Hsiao-Fang Lin

Annandale-on-Hudson, New York
May, 2013

Abstract

Is it just a statistics problem?

How can a listener tell which instruments are playing together without reading the musical score? Can a computer algorithm separate different instrument voices automatically? Blind signal separation tries to separate instrumental voices based on statistical regularities in the signal. This project focuses on recordings of two different instruments simultaneously playing the same notes. Independent components analysis, one form of blind signal separation, was used to separate signals collected by two independent microphones. The experimental results show that voices combined by adding signals mathematically are separated better than voices combined acoustically and collected by two independent microphones. Instrument timbres also affect the ability to separate voices. Statistics can separate some artificially constructed signals, but separating recorded signals is much more difficult.

Contents

List of Figures

Dedication

秋風拂檻氣蕭森，
兀坐紅窗思轉深，
烏鳥尚然知反哺，
吾生卻愧不如禽。

謹獻給我最親愛的父母親。

Acknowledgments

This research project would not have been possible without the support of many people.

First and foremost, I would like to express my deep gratitude to Professor Sven Anderson, my research advisor, for his patient guidance of this research work. He never tires about answering my questions, no matter what my question is. Without him, I could never finish this project. I would also like to thank Professor Robert McGrail, Feifan Zheng, and Chi-Hui Yen for their spiritual support. They are always there when I need them no matter what. My grateful thanks are also extended to my dear suit-mates , they make me feel I'm at my hometown, Taiwan, everyday. As an international student, I could never be more thankful for what they do.

I would also like to extend my thanks to all the Bard conservatory of Music members who help me to record the data for my project and support me whenever I have the difficult time between music and computer science.

Finally, I wish to thank everyone who shows up in my life at Bard College and all my friends in Taiwan who always cheer me up in my 5 years study aboard.

1
Introduction

1.1 Motivation

Purchasing the digital music become popular nowadays. If you want to know which instruments are playing but you do not have the music score and cannot distinguish all of the instruments by ear, how will you do it? The good of this research is to help people recognize different instruments in instrumental music. For example, if you listen to a symphony piece with more than 15 instruments playing at the same time, how can you tell which instruments are playing? Blind Signal Separation is explored as a solution to this problem.

Instrumental music separation is important for musicians, especially for orchestra players. When a musician listens to music and tries to learn their own part with music score, it is easier for the musician if they can listen to their own part without other instruments. Therefore, if we can successfully separate the instrumental music to individual instrument parts, it benefits the musicians.

Music separation has two parts. One is human voice separation and the other one

is instrumental voice separation. Separation of human voice from instruments has been done well so far. REPET (REpeating Pattern Extraction Technique) is the most popular method for separating the human voice from instrumental music. It focuses on modeling the accompaniment instead of the vocals. REPET starts from the observation that many popular music recordings can be understood as a repeating musical background, over which a voice signal is superimposed that does not exhibit any immediate repeating structure. Based on this observation, a model for the background signal can be computed, provided its period is correctly estimated. This technique proved to be highly effective for excerpts with a relatively stable repeating background (for example, 10 second verse). For longer musical excerpts, however, the musical background is likely to vary over time (for example, verse followed by chorus), limiting the length of excerpt that REPET can be applied to. We can't use the same technique to separate instrumental music as explained below.

Usually, an orchestra music contains more than 10 different instruments and often they play together harmonically or switch the melody line. Hence, we cannot use REPET to isolate one of the instruments from rest of the orchestra sound.

The character of the sound is spread across a large frequency range, with higher peaks at some frequencies, and lower levels at other frequencies. Combined, all of this together is the total"energy" of the sound. From a different perspective, the total energy of a sound is spread across a range of frequencies. Taking the guitar example above, the energy of the sound is almost entirely in the 80Hz - 5,000Hz range. There is an excellent examination of musical instrument spectra above 20,000Hz by James Boyk [1] that explains more about the energy of various instruments. In particular, the article contains a good number of diagrams illustrating peaks at different harmonics.

1.2 Background

Gavelin, Klomp, Priddle, and UddenfeldtBl (2004) [5] present a real-time implementation of blind source separation of two sources from two unknown mixtures. The algorithm used is a version of the degenerate unmixing estimations technique (DUET) algorithm and has been implemented in Matlab. They have tested the implementation on both artificial and real mixtures. The artificial mixtures could be separated almost perfectly, whereas only some of the "real recordings" produced a noticeable separation, others did not separate at all. They separated voice mixtures, but BSS works with any near W-disjoint orthogonal signals. This report also contains a brief theoretical description of Bayesian ICA for static linear mixtures and the use of a feedback Infomax separation network for convolved mixtures.

Antti Eronen (2003) [3] describe a system for the recognition of musical instruments from isolated notes or drum samples. The author first describes a baseline system that uses melfrequency cepstral coefficients and their first derivatives as features, and continuous-density hidden Markov models (HMMs). Two improvements are proposed to increase the performance of this baseline system. First, transforming the features to a base with maximal statistical independence using independent component analysis can give an improvement of 9 percentage points in recognition accuracy. Secondly, discriminative training is shown to further improve the recognition accuracy of the system. The evaluation material consists of 5895 isolated notes of Western orchestral instruments, and 1798 drum hits.

Uhle, Dittmar, and Sporer (2003) [12] propose a method for the separation of drum tracks from polyphonic music. It consists of an Independent Component Analysis and a subsequent partitioning of the derived components into subspaces containing the

percussive and harmonic sustained instruments. With the proposed method, different samples of popular music have been analyzed. The results show sufficient separation of drum tracks and non-drum tracks for subsequent metadata extraction. Informal listening tests prove a moderate audio quality of the resulting audio signals.

Teddy and Lai(2004) [11] restrict the single track recordings to instrumental music. They attempt to construct separate streams of data each consisting of the sound of a single instrument. A model-based approach is used. The architecture of the system is based on a cerebellar-based (CMAC) fuzzy neural network.

Lewis, Zhang, and Ras (2006) [8] analyze how musically trained human listeners overcome resonance, noise, and overlapping signals to identify and isolate what instruments are playing and then what pitch each instrument is playing. They identify the dominant instrument from a base signal uses temporal features proposed by Wieczorkowska in addition to the standard 11 MPEG7 features as a part of the instrument and pitch recognition system. After retrieving a semantical match for that dominant instrument from the database, it creates a resulting foreign set of features to form a new synthetic basen signal which no longer bears the previously extracted dominant sound.

Pohle, Knees, Schedl, Gerhard, and Widmer (2006) [10] use Independent Component Analysis (ICA) to decompose an audio signal into individual parts that appear maximally independent from each other. In the paper, they present one basic algorithm to use these components for similarity computations, and evaluate a number of modications to it with respect to genre classication accuracy.

Favaro, Lewis, and Schlesinger (2011) [4] focus on to separate the music recording contains 4 different instruments by using Independent component analysis. They make four different instruments play unison, various rhythmic pattern, and also a simple piece. Although ICA performed well in the time domain on our artificially created instantaneous mixtures, the algorithm's performance degraded rapidly when propagation delays were

introduced. The recovered signals from the real world recordings were less isolated than the observed signals. To account for these propagation delays we subsequently focused our efforts on separation in the frequency domain. In order to focus on the frequency domain, they apply the Fourier transform to the source signal before they apply ICA.

Liutkus, Rafii, Badeau, Pardo, and Richard (2012) [9] overcome the limitation of REPET and generalize REPET to permit the processing of complete musical tracks. The limitation of PEPET is that while effective on individual sections of a song, REPET does not allow for variations in the background (e.g., verse vs. chorus), and is thus limited to short excerpts only. The proposed algorithm tracks the period of the repeating structure and computes local estimates of the background pattern. Separation is performed by soft time-frequency masking, based on the deviation between the current observation and the estimated background pattern. Evaluation on a dataset of 14 complete tracks shows that this method can perform at least as well as a recent competitive music/voice separation method, while being computationally efficient.

As with many fields today, the processing of instrumental musical information is inundated with huge masses of data, much of which is redundant. An important algorithm for separating audio signals is Independent Component Analysis. We first present simple signal sine wave and sawtooth wave mixed separation and then secondly, we present the pseudo-mixed signal separation. Pseudo-mixed signal separation means we take two solo instrument recordings and then mix them on the computer. Last, we present the real polyphonic music recording separation is mixed by the recorder.

2
Blind Signal Separation

2.1 Blind Signal Separation

Blind Signal Separation (BSS) is the isolation of individual source signals from a mixture of those signals. The Cocktail party problem is one of the most famous problem for BSS. Assume you are in a cocktail party and you are listening to your friend to talk. However, you are more interested in other people's conversation on the other side of the room. Hence, you are trying to focus on the conversation you want to listen to. This is the cocktail party problem: to pick out one thread of speech from the babble of two or more people. For example, M microphones might record M signals from the microphones from different location, Figure 2.1.1.

We assume that the source signals are independent; knowing the value of one of them does not give any information about the others. Blind Signal Separation is "blind" because: 1) we don't know the original signals, and 2) we don't know how they are mixed. There are many fields where BSS is useful. For example, separation of radio signals in telecommunication, separation of brain waves from medical sensors or demixing stereo

recordings. The methods for BSS are not tied to any specific type of signals, but in my senior project mixed instrumental audio signals are used in the experiments.

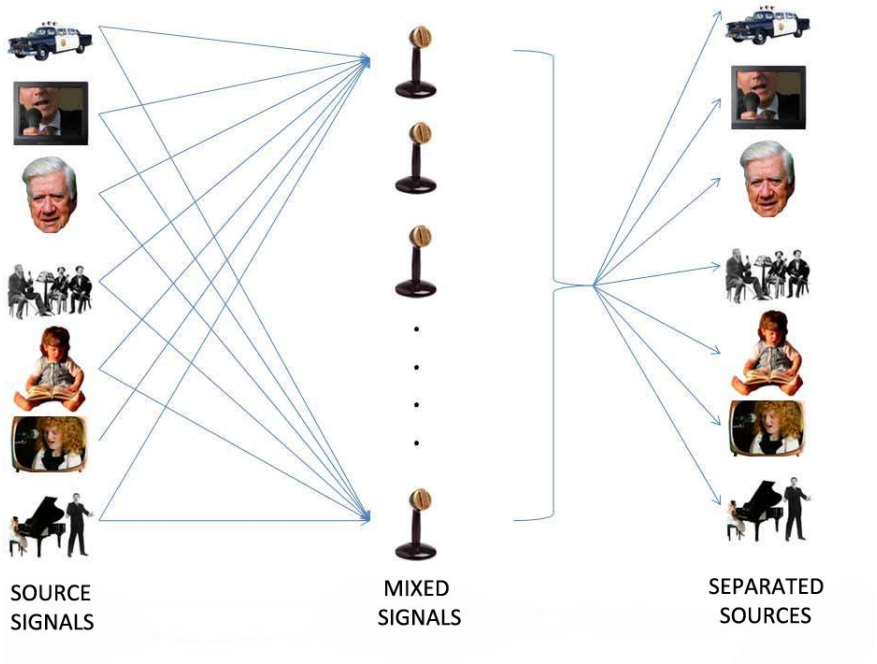

Figure 2.1.1. Mixed signals come from outputs of different sensors. Every sensor/microphone gets different signal from different source signal. The signal strength recording by the microphone depends on how far the microphone is from the source signal.

2.2 Independent Component Analysis

Independent Component Analysis (ICA) is one algorithm that implements a limited form of Blind Signal Separation. Assume source signals are mutually statistically in-

dependent and non-Gaussian. We restrict our attention to signals that are linearly mixed. Assume we have a set of source signals S and a mixing matrix, A, contains constant factors. The mixed signal, X, is the sum of the product of each source signal and a factor that might represent the distance to each sensor.

$$X = AS \tag{2.2.1}$$

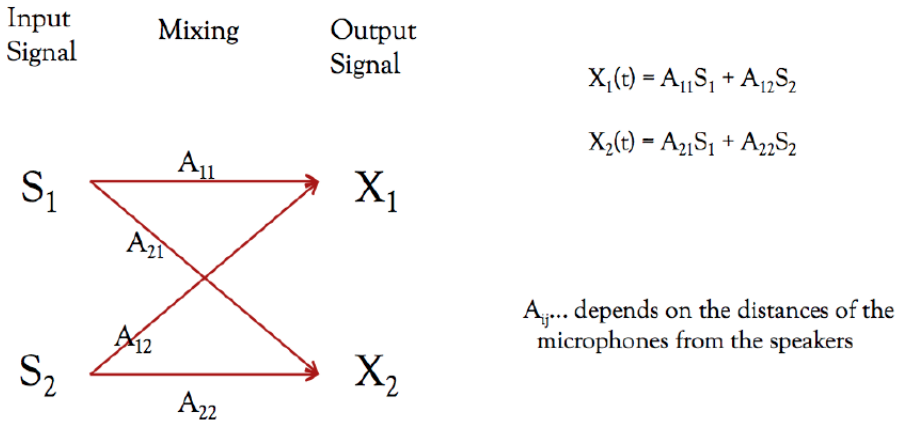

Figure 2.2.1. $X = AS$. X is the mixed signal from the source signal S and the mixing matrix A.

The Independent Components, $S(t)$, are unknown input signals, and the Mixing matrix A is also unknown. Our task is to estimate A and S by using only the observable output signals $X(t)$. We also know that 1) S_i are statistically independent, and 2) they have non-gaussian distributions. In probability theory, to say that two events are statistically independent means that the occurrence of one does not affect the probability of the other.

Formally,

$$P(y_1, y_2) = P(y_1)P(y_2). \tag{2.2.2}$$

Consider the case of two different random variables s_1 and s_2. The random variable s_1 is independent of s_2 , if the information about the value of s_1 does not provide any information about the value of s_2 , and vice versa. Here s_1 and s_2 could be random signals originating from two different physical process that are not related to each other. For N different instruments at different location played by different players, the source signals should be independent.

2.3 Why Non-Gaussian?

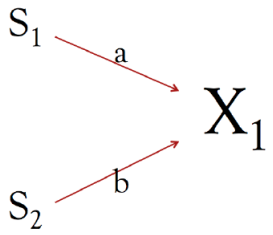

Figure 2.3.1. The sum of less Gaussian inputs will get more Gaussian output

According to the Central Limit Theorem the distribution of a sum of independent signals with arbitrary distributions tends toward a Gaussian distribution under certain conditions. The sum of two independent signals usually has a distribution that is closer to Gaussian than distribution of the two original signals.

Conversely, if two signals have Gaussian distributions, Figure 2.3.3, their density is symmetric: no matter how we rotate it, it will always be the same. We want to focus on the most important part in the mixed signal, which means the most frequent data we get in a mixed signal. The number of appeared times is the density of the mixed signal. Hence, if our source signal S is Gaussian, then we still cannot reduce less appeared data from the mixed signal,Figure2.3.2. If S_1 and S_2 are non-Gaussian, then we can mix S_1 and S_2 by multiply the mixing matrix A then we can get the a non symmetric mixed signal X, Figure2.3.3 However, if only one independent component is gaussian, the estimation is still possible (Hyvarinen and Oja 1999).

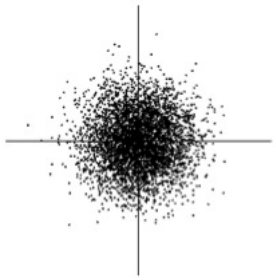

Figure 2.3.2. When both sources signals are Gaussian. Horizontal axis: S_1; vertical axis: S_2

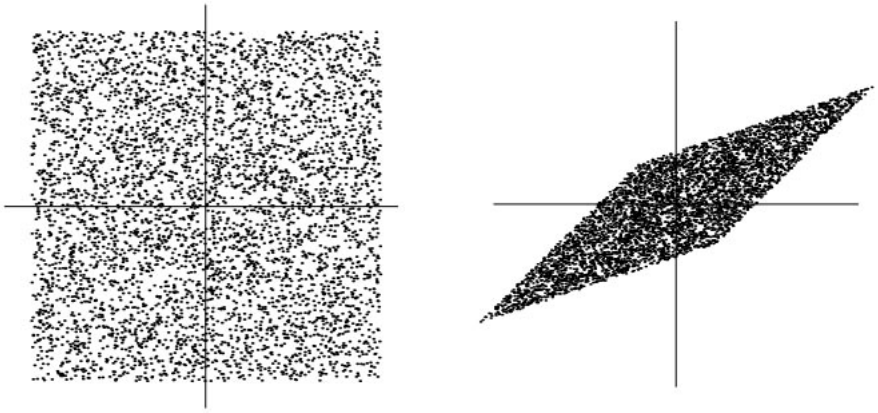

Figure 2.3.3. Left: The joint distribution of the independent components S_1 and S_2. Horizontal axis: S_1; vertical axis: S_2; Right: the joint distribution of mixed signals X_1 and X_2. Horizontal axis: X_1; vertical axis: X_2

2.4 Fast ICA

Fast ICA is one implementation of Independent Component Analysis. It is based on finding the inverse of mixing matrix A. We will refer to the this as matrix W. Since the mixed signals X equal to the mixing matrix A times the source signals S, we can obtain the unknown source signal from by

$$S = WX. \tag{2.4.1}$$

2.5 What's negentropy?

Negentropy is a way to measure of nongaussianity. It is based on information-theoretic quantity of entropy, a basic concept of information theory. The entropy of a random variable can be explained as the information obtained by observing the variables. For example, if the variable is more random, then its entropy is larger.

Entropy H is defined for a discrete random variable Y as

$$H(Y) = -\sum_i P(Y = a_i) log\ P(Y = a_i) \tag{2.5.1}$$

where the a_i are the possible values of Y. The differential entropy is the generalization of entropy for continuous valued random variables and vectors. The differential entropy H of a random vector y with density $f(y)$ is defined as

$$H(y) = -\int f(y) log\ f(y) dy. \tag{2.5.2}$$

A fundamental result of information theory is that *a Gaussian variable has the largest entropy among all random variables of equal variance* (Cover and Thomas,1991). Therefore, the entropy can be used as a measure of nongaussianity. Nongaussianity is always nonnegative and is zero for a gaussian variable. To obtain the measure of the nongaussianity, we modify the differential entropy in order to get the nongaussianity, J, this is called negentropy. Negentropy J is defined as follows

$$J(y) = H(y_{gaussian}) - H(y) \tag{2.5.3}$$

where $y_{gaussian}$ is a Gaussian random variable of the same covariance matrix as y. A covariance is an unstandardized version of correlation. To compute any correlation, we

divide the covariance by the standard deviation of both variables to remove units of measurement. Hence a covariance is a correlation measured in the units of the original variables. Because the covariance is in the original units of the variables, variables on scales with bigger numbers, and with wider distributions, will necessarily have bigger covariances. A covariance matrix is a symmetric matrix whose element in the i, j position is the covariance between the i^{th} and j^{th} elements of a vector of random variables.

From the above equation, we know that $J(y)$ is always nonnegative, and it is zero if y is a Gaussian distribution. However, the estimation of negentropy is difficult. Therefore, some approximation methods of negentropy have to be used. In FastICA, we use the approximation of negentropy is

$$J(y) \propto [E\{G(y)\} - E\{G(v)\}]^2 \tag{2.5.4}$$

for practically any non-quadratic function G. In FastICA, we choose non-quadratic function G as follows

$$G(u) = \frac{1}{4}\, u^4, g(u) = u^3 \tag{2.5.5}$$

where $1 \leq a_1 \leq 2$ is some suitable constant [6, 7].

2.6 FastICA for one unit

FastICA contains two parts. First, we need to find one signal. We find one of the independent components, the source signals by calculating a weight vector, **w**, that the neuron is able to update by a learning rule. FastICA is a learning rule that finds one of the direction of the mixed signal X. We use the approximation of negentropy $J(w^T x)$ given

in Eq.(2.5.4) to measure nongaussianity then use an unit weight vector **w** such that one of the projection $w^T x$ maximizes nongaussianity. FastICA is based on a fixed-point iteration scheme for finding a maximum of the nongaussianity of $w^T x$, as measured in Eq.(2.5.4). Because we want to find the maximum nongaussianity of $w^T x$, we take the derivative of the non-quadratic function G used in Eg.(2.5.5), denoted as g:

$$g_1(\, u\,) = tanh(\, a_1\, u\,) \tag{2.6.1}$$

where $1 \leq a_1 \leq 2$ is some suitable constant, often taken as $a_1 = 1$. The basic form of the FastICA algorithm is:

For one unit:

Step 1: Choose an initial weight vector w

Step 2: Let $w^+ = E\{\, X\, g(W^T\, X)\,\} \,-\, E\{\, g^{'}(W^T\, X)\,\}\, W$

Step 3: Let $w = \frac{\mathrm{w}^+}{\|w^+\|}$

Step 4: If not converged, go back to step 2.

First, we choose a random weight vector **w**. Then we try to find the maximum nongaussianity of $w^T x$ by using the derivative of the nonquadratic function G, as derived above . After we find the maximum nongaussianty, we normalize the updated wight vector w^+. If W end converge, the algorithm terminates; otherwise it returns to Step 2.

2.7 FastICA for multi units

When we find the maximum of nongaussianity for one signal via FastICA, we now find the rest of the weight vectors $w_1 \ldots w_n$ by running the one unit algorithm again and again. In order to prevent different vectors from converging to the same maximum, we must decorrelate the outputs $w_1^T x, \ldots, w_n^T x$ after every iteration. Symmetric decorrelation is one of the implementations to decorrelate weight vectors $w_1 \ldots w_n$, in which no vectors are "privileged" over others:

$$W = (WW^T)^{-}1/_2 W \tag{2.7.1}$$

where **W** is the matrix of all the weight vectors w we find from one unit FastICA, $(w_1, \ldots, w_n)^T$. The inverse square root $(WW^T)^{-}1/_2 W$ is obtained from the eigenvalue decomposition of WW^T as follows:

Step 1, Normalize the **W**:

Let $W = \frac{W}{\sqrt{\|WW^T\|}}$

Step 2, Repeat this step until convergence:

Let $W = \frac{3}{2}W - \frac{1}{2}WW^TW$

$< Proof >$ [7]

Denote by W_+ the result of applying once the iteration in Step 2 on W. Let $WW^T = EDE^T$ be the egenvalue decomposition of WW^T. Then we have

$$W_+W_+^T = \frac{9}{4}EDE^T - \frac{3}{2}ED^2E^T + \frac{1}{4}ED^3D^T \tag{2.7.2}$$

$$= E(\frac{9}{4}D - \frac{3}{2}D^2 + \frac{1}{4}D^3)E^T \tag{2.7.3}$$

Note that due to the normalization, i.e division of W by $\sqrt{\|WW^T\|}$, all the eigenvalues of WW^T are in the interval [0,1]. Now, according to Equ. (2.7.2), for every eigenvalue of WW^T, say λ_i, $W_+W_+^T$ has a corresponding eigenvalue $h(\lambda_i)$ where $h(.)$ is defined as:

$$h(\lambda) = \frac{9}{4}\lambda - \frac{3}{2}\lambda^2 + \frac{1}{4}\lambda^3 \tag{2.7.4}$$

Thus, after k iterations, the eigenvalues of WW^T are obtained as $h(h(h(\ldots h(lambda_i))))$, where h is applied k times on the $lambda_i$, which are the eigenvalues of WW^T for the original matrix before the iterations. Now, we have always $h(lambda) > \lambda$ for $0 < \lambda < 1$. It is therefore clear that all the eigenvalues of WW^T converge to 1, which means that $WW^T \to I$, Q.E.D. Moreover, it is not difficult to see that the convergence is quadratic.

2.7.1 Preprocessing for ICA algorithm

First, we need to center X, which is the mixed signal. When centering is applied, all the data in each mixed signal are slid vertically so that their average is zero. After we centering the mixed signal X, all the mixed signals have unit norm. To center the mixed signals, we subtract X's mean vector

$$m = E\{\, x \,\} \tag{2.7.5}$$

to make X become a zero-mean variable. Then X has to be whitened.

Whitening X means to decorrelate its components and make their variances

direct to the same unity. We can use the covariance matrix of $\widetilde{x}$ to examine if X is whitened. If the expectation value of $\widetilde{x}$'s covariance matrix is identity, then X is whitened:

$$E\{\,\widetilde{x}\,\widetilde{x}^{T}\} = I \tag{2.7.6}$$

We use the eigenvalue decomposition of the covariance matrix to whiten the X.

Eigenvectors and eigenvalues provide the eigenvalue decomposition (EVD) of a matrix which analyzes the structure of this matrix. EVD is used to find the maximum (or minimum) of functions involving these matrices. For example, the whitening X step is obtained from the eigenvalue decomposition of a covariance matrix of X and gives the least square estimate of the original data matrix.

2.7.2 Whitening X

One of the whitening methods is to sue the eigenvalue decomposition of the covariance matrix $E\{\,x\,x^{T}\,\} = \mathbf{EDE}^{T}$, where $\mathbf{E}$ is the orthogonal matrix of eigenvectors of $E\{\,xx^{T}\,\}$ and $\mathbf{D}$ is the diagonal matrix of its eigenvalues. Now, whitening can be done by

$$\widetilde{x} = \mathbf{E}\mathbf{D}^{\frac{-1}{2}}\mathbf{E}^{T}\,\mathbf{X} \tag{2.7.7}$$

The *Figure* 2.7.1 indicates the whitened mixed signal X from *Figure* 2.3.3. The square defining the distribution is now clearly a rotated version of the original square in *Figure* 2.3.3.

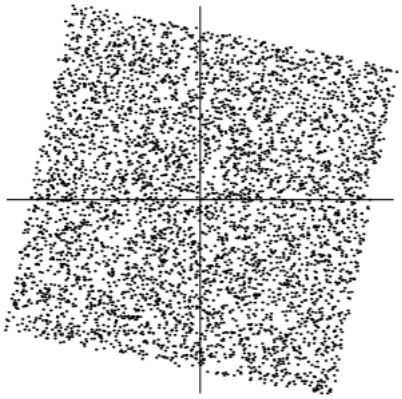

Figure 2.7.1. The joint distribution of whitened mixed signal X. Horizontal axis: X_1; vertical axis: X_2

3

Using FastICA to Separate Audio Signals

3.1 Algorithm FastICA with known signals

To test FastICA algorithm, I implemented the algorithm in Java. First, I used two known simple source signals S_1 and S_2 to create my mixed signal X by randomly pick a mixing matrix A. One of the source signals is sine wave and the other source signal is a sawtooth wave as shown in Figure 3.1.1 below:

```
double[] sineWave = new double[n];
double[] sawtoothWave = new double[n];
double x = 0.0;
for(int i = 0; i < n.length; i++){
        sinWave[i] = Math.sin(x);
        sawtoothWave[i] = 2*Math.abs(2*((x-0.25) - Math.floor(x-0.25))-1);
        x += (4*Math.PI)/n;
```

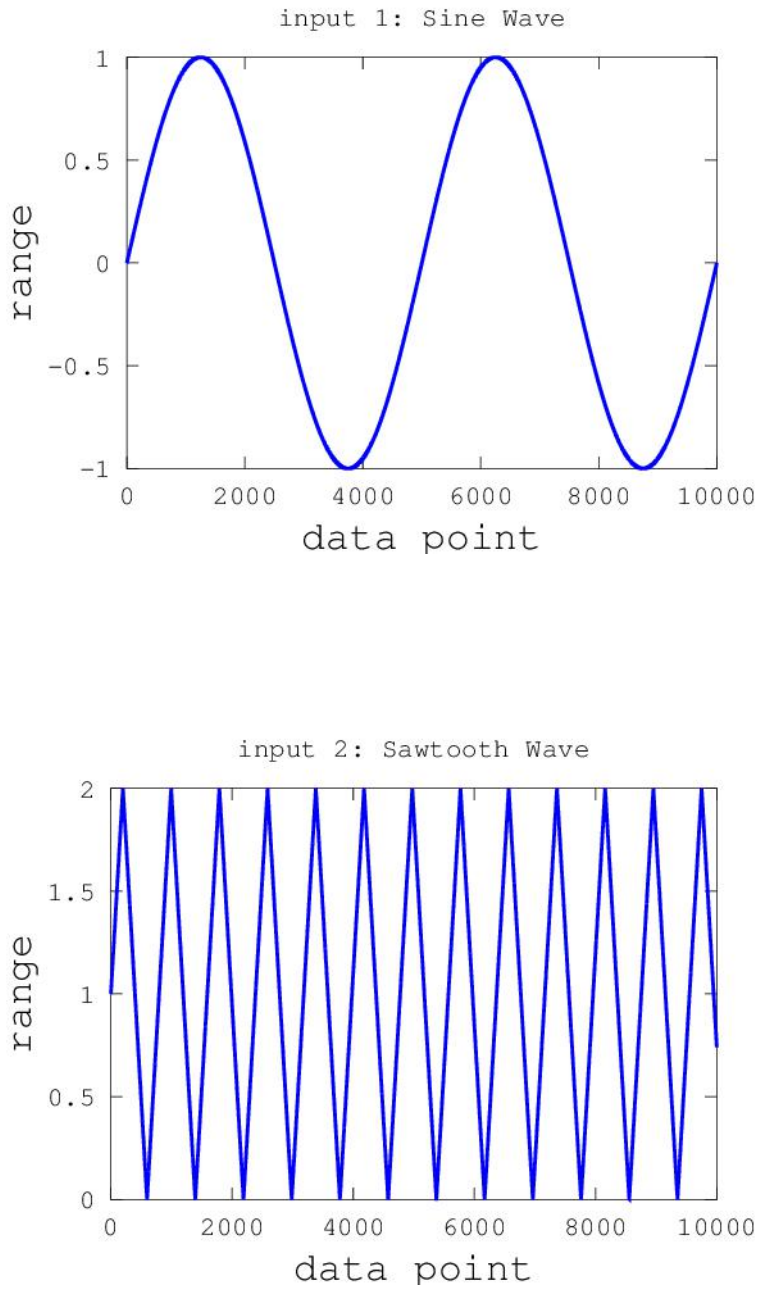

Figure 3.1.1. Two input signals. Top: sine wave; Bottom: sawtooth wave

The mixed signal $X = AS$, where $A = \{\{0.3, 0.9\}, \{1.27, 0.95\}\}$:

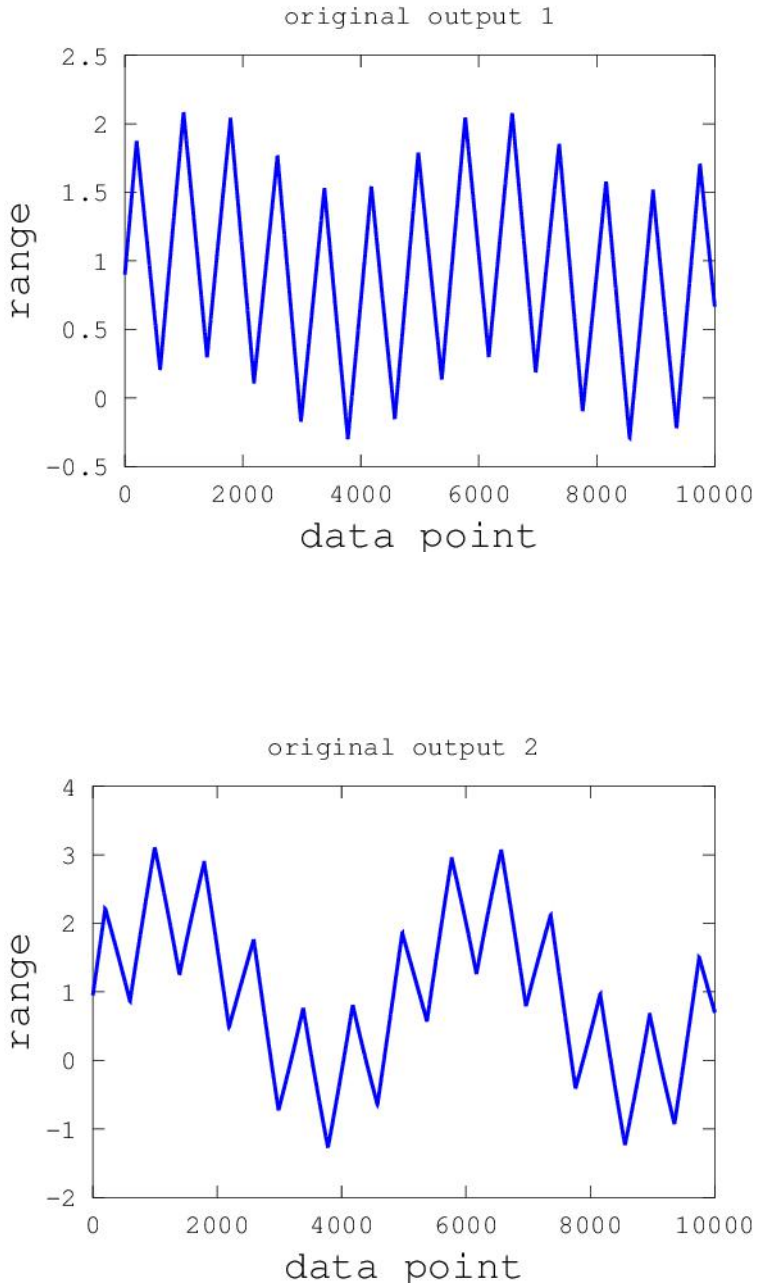

Figure 3.1.2. Top: mixed signal X_1; Bottom: mixed signal X_2

Before using the FastICA algorithm, I center the source signal as shown in Figure 3.1.3:

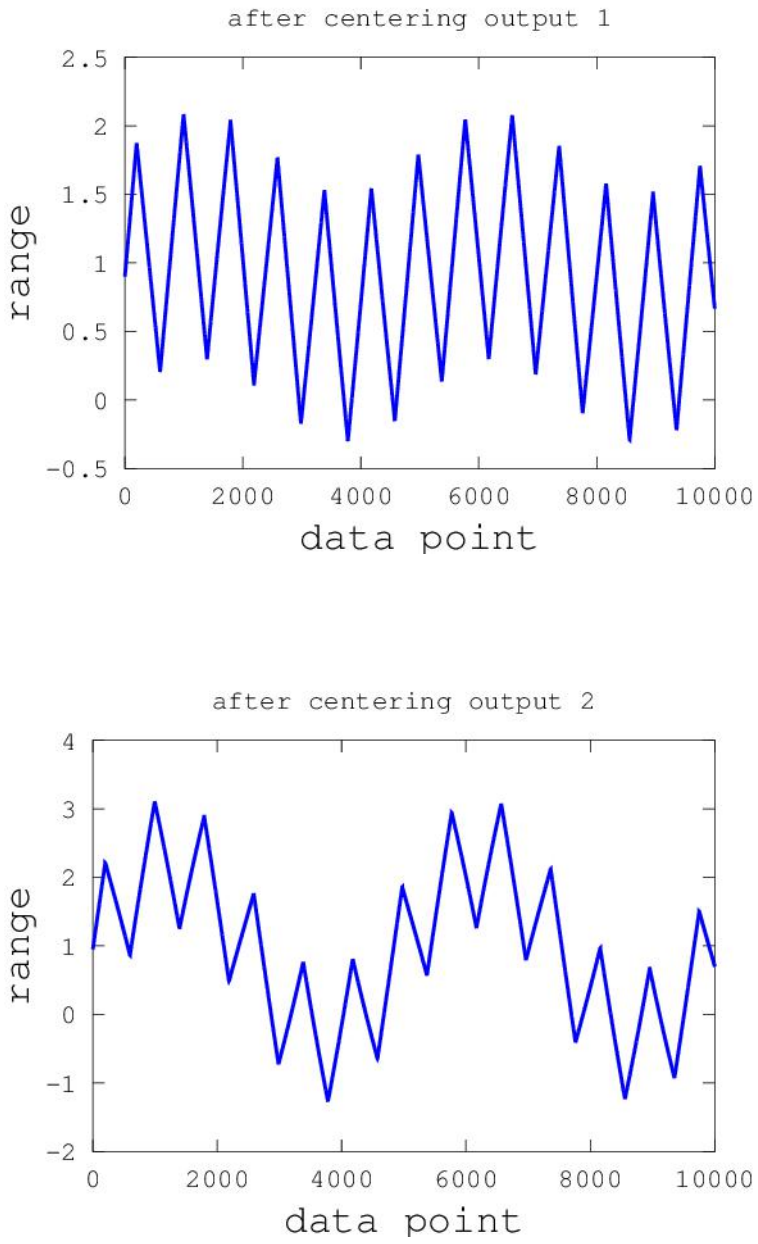

Figure 3.1.3. Top: after centering X_1; Bottom: after centering X_2

After centering the mixed signal, we need to whiten the X. The following Figure is a histogram of the data before and after whitening

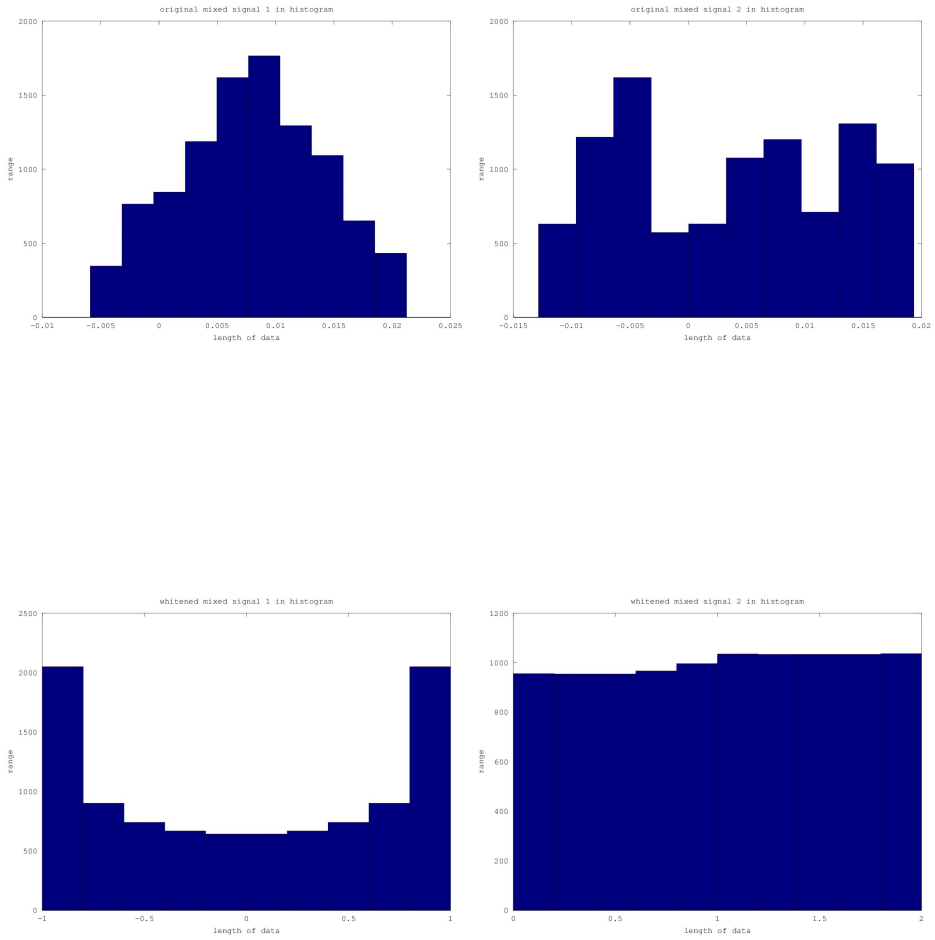

Figure 3.1.4. Top: before whitening X. Left: X_1, Right: X_2; Bottom: after whitening X. Left: X_1, Right: X_2. X-axis: range, Y-axis: data point

From the Figure3.1.4, we see after the whitening step, the X_1 and X_2 are flatter because the whitening step removes any correlations in the data. Now, implement the FastICA algorithm:

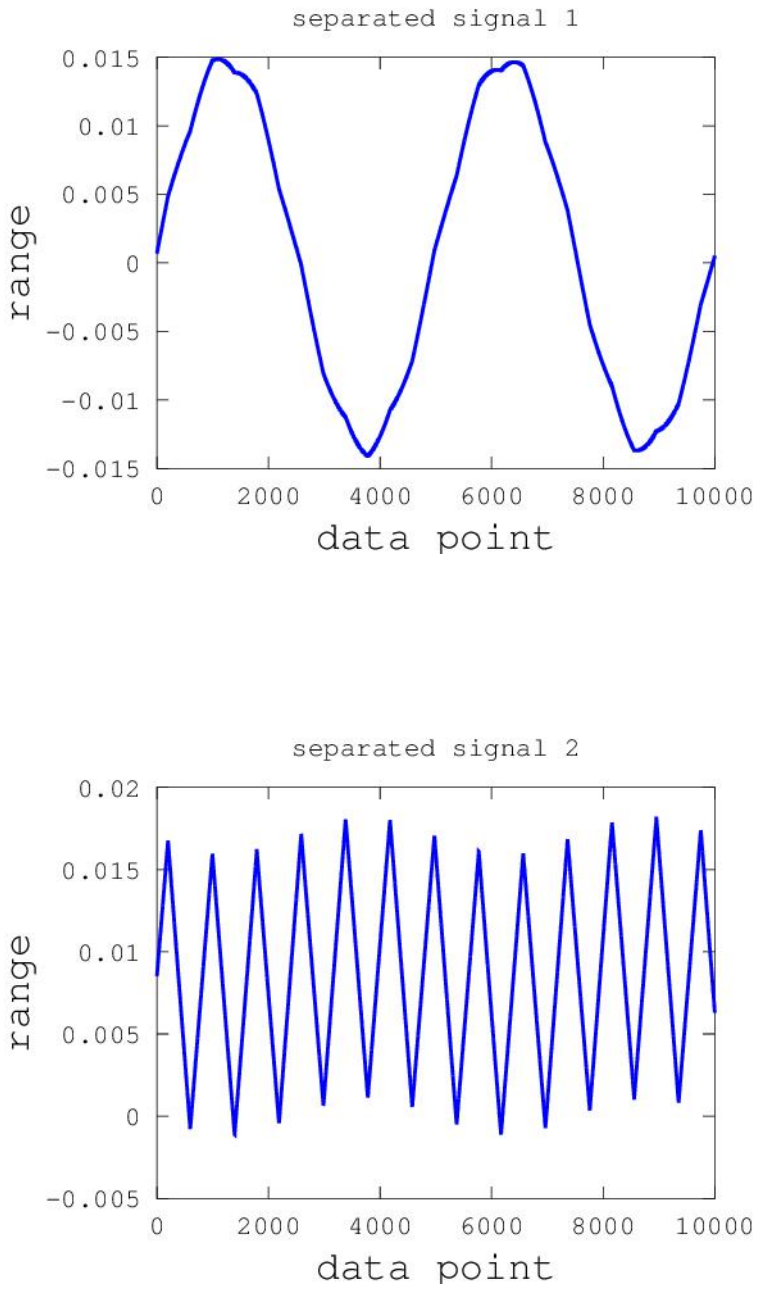

Figure 3.1.5. Top: separated signal S_0' from X_0 ; Bottom: separated signal S_1' from X_1

As shown in the Figure 3.1.5, the result we get after we implement the FastICA algorithm is scaled.

3.2 Algorithm FastICA with known audio signals

The previous section shows the FastICA algorithm works well for two simple input signals. However, how about the FastICA for music signals? Hence I ask nine conservatory students who play different instruments to record exact same scale and same loudness individually. Each played the scale at the same frequency. The instruments include violin, piccolo, flute, oboe, clarinet, bassoon, trumpet, trombone, and piano. These nine instruments include every way to make the sound. For example, piano is chordophone and percussion kind; violin is string; flute and piccolo are aerophone or reedless wind; oboe and bassoon are double reed wind; clarinet is single reed wind; trumpet and trombone are brass. They also have different ranges. Since they are in the different range, the players agree to play C_4 major scale. The frequency of C_4 is 261.6 HZ. I used the Zoom H4n digital recorder to record every instruments. I setup the amplify from Zoom H4n so the loudness of each recordings is the same. I made the recording in the recording studio to make sure it is a quiet environment. The players stand 3.7m far from the recording machin, Figure3.2.1. The recording is in 16 bits mono 44100 HZ. The average length of these recordings is 10 seconds; however, I only take 0.4 seconds from each recording. Because when the player plays the scale, the note does not resonant well until the player plays the seventh note in the ascending scale. From the waveform, the seventh note, B_4, is the most resonant note

in the scale for all the players. (*see Figure* 3.2.2)

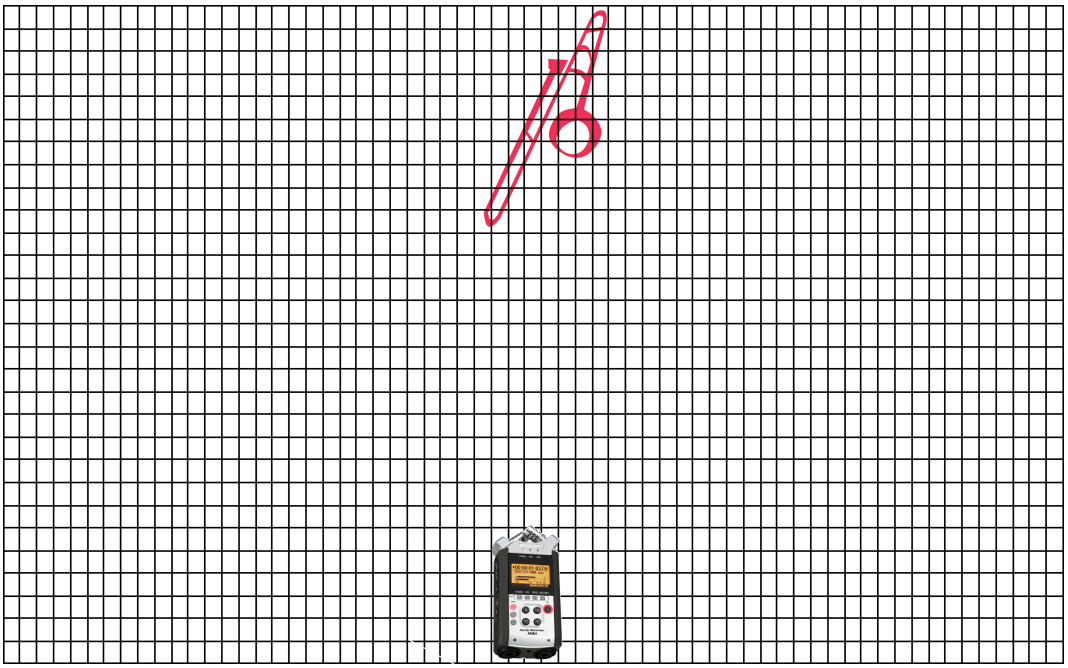

Figure 3.2.1. Different instruments' waveform. Left: the beginning of the scale, note C_4 waveform; Right: note B_4 waveform

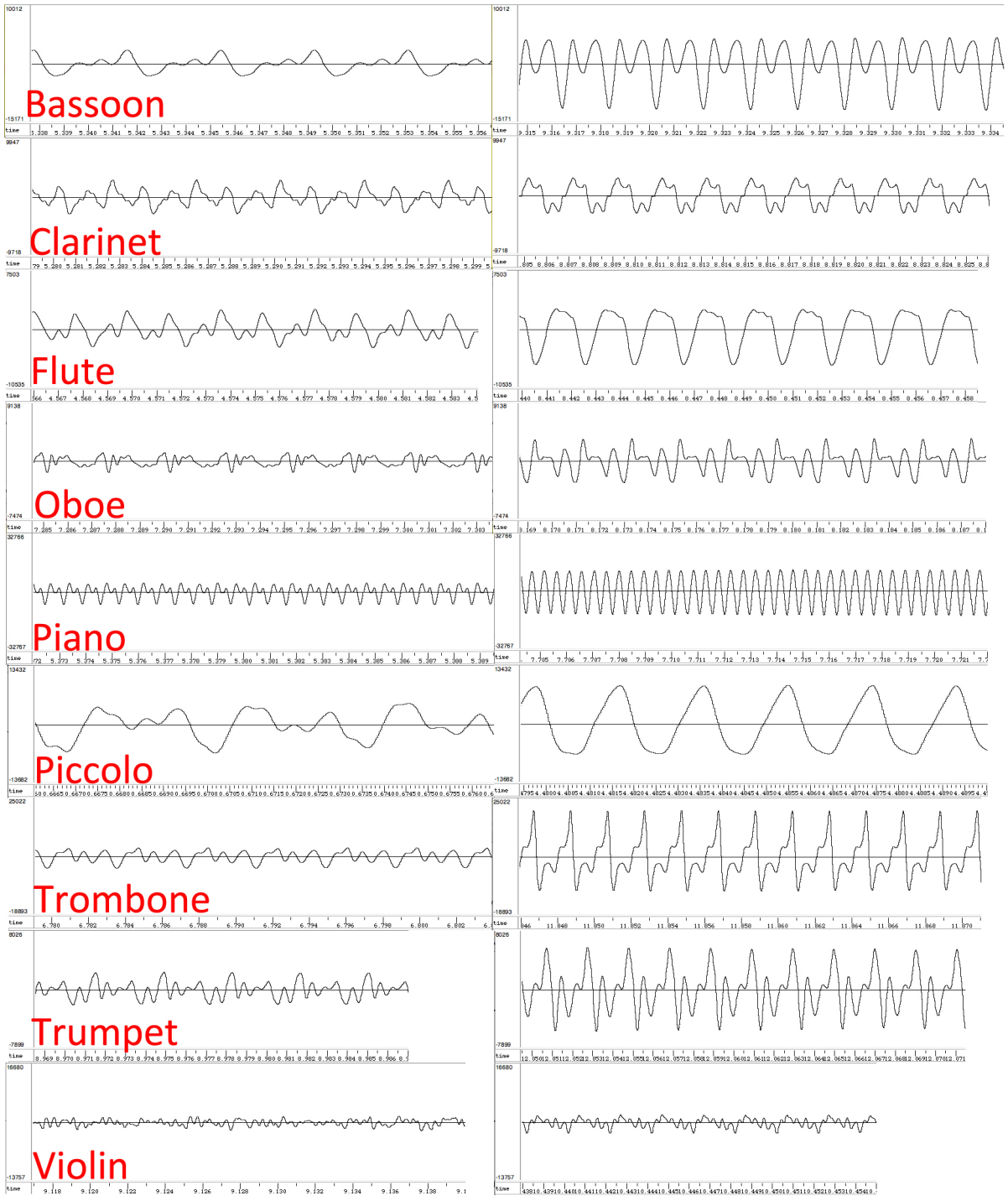

Figure 3.2.2. Different instruments' waveform. Left: the beginning of the scale, note C_4 waveform; Right: note B_4 waveform

I mix two different signals from two different instruments. The mixed matrix A is the same as the previous section: $A = \{\{0.3, 0.9\}, \{1.27, 0.95\}\}$. The result is shown in the table below:

	Bassoon	Clarinet	Flute	Oboe	Piano	Piccolo	Trombone	Trumpet	Violin
Bassoon		0.56225069 0.38240927	0.49255259 0.36304294	0.43566694 0.38989867	0.42339928 0.28319065	0.47607280 0.33427260	0.45697147 0.33085251	0.45027705 0.32530712	0.42768173 0.30138163
Clarinet	0.56225069 0.38229823		0.39227697 0.38606046	0.58800912 0.41250350	0.44912219 0.30721346	0.49002715 0.35444061	0.43070317 0.34925090	0.43221916 0.34137944	0.48974351 0.32543452
Flute	0.49255259 0.36156833	0.39227697 0.38118985		0.55762926 0.39319032	0.41429656 0.28459007	0.46211159 0.33859637	0.41069771 0.32741322	0.43424836 0.3204107	0.35203534 0.29981187
Oboe	0.43566694 0.38863351	0.58800912 0.40979633	0.55762926 0.38841273		0.46097804 0.31057272	0.50305365 0.36747314	0.46978365 0.35596733	0.4855748 0.34794519	0.46803008 0.32856464
Piano	0.42339928 0.28316421	0.44912219 0.30918009	0.41429656 0.28411808	0.46097804 0.31088691		0.38159978 0.25603137	0.36176341 0.25446523	0.40598705 0.24293161	0.32184394 0.22285127
Piccolo	0.47607280 0.33441906	0.49002715 0.35457105	0.46211159 0.33276332	0.50305365 0.36113499	0.38159978 0.25805831		0.41141141 0.30064227	0.4251696 0.29477632	0.39129331 0.27454579
Trombone	0.45697147 0.33366346	0.43070317 0.34956525	0.41069771 0.32651238	0.46978365 0.35554642	0.36176341 0.25311256	0.41141141 0.30321044		0.42828258 0.28727138	0.35794933 0.2687578
Trumpet	0.45027705 0.32068915	0.43221916 0.34387609	0.43424836 0.31978973	0.4855748 0.34986638	0.40598705 0.24316481	0.4251696 0.29694553	0.42828258 0.28730105		0.3806754 0.26329719
Violin	0.42768173 0.3022166	0.48974351 0.32227069	0.35203534 0.29954629	0.46803008 0.32856834	0.32184394 0.22279472	0.39129331 0.27366407	0.35794933 0.27245789	0.3806754 0.26193254	

Figure 3.2.3. Table of difference. Blue/top number: the difference of waveform between two instruments; Black/bottom number: the difference between the separated signal S' and the source signal S.

For the difference between two instruments, I subtract one data from the other one then divided by the data length. The length of two instruments are the same. For the difference between the original signal and separated signal, I first find how mach the separated signal is scaled, say constant d. Then I multiply d and separated music in order

to scale the separated signal back.

From the table above we can see the best result of separated signal S' is from the mixed signal X of violin and Piano. Violin and Piano are the closest waveforms in these nine different instruments. Hence, it is the best result. On the other hand, oboe and clarinet are the most different waveforms, mainly they are least well separated. The worst result means the separated signal S' is not closed to the source signal S. The figures below are shown the piano, violin, clarinet, and oboe's waveforms:

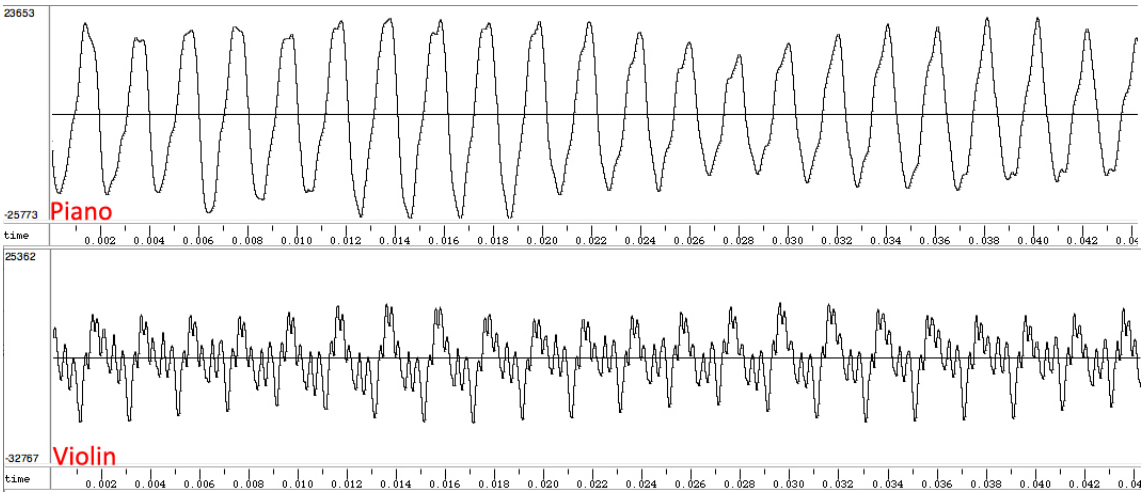

Figure 3.2.4. piano and violin's waveforms

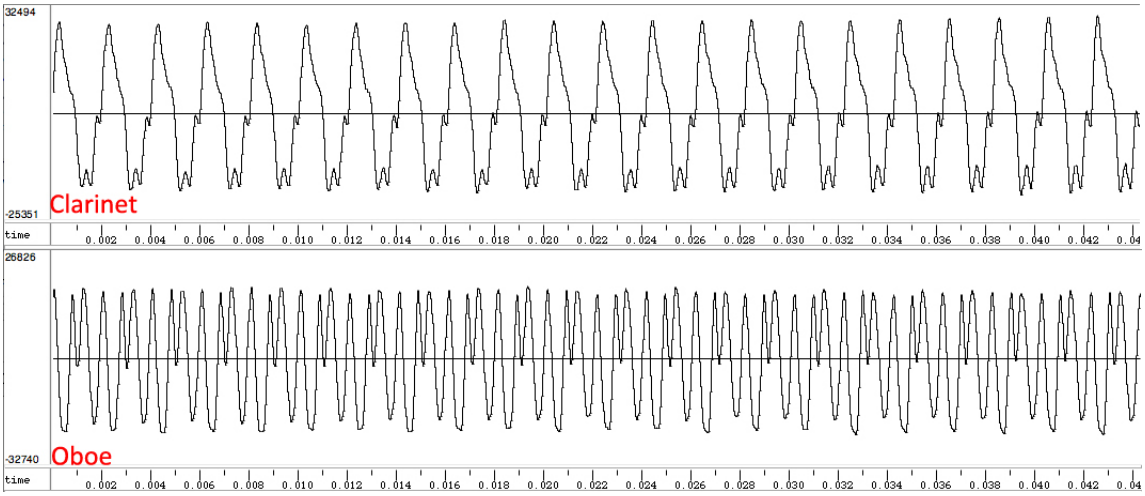

Figure 3.2.5. clarinet and oboe's waveforms

Since, mixed signal X from piano and violin can get the best result, I increase my data length to one second from scale to make the mixed signal X. The difference between one second piano waveform and one second violin waveform is 0.18864145 and the difference between the source signals S_1 and S_2 and separated signals S_1' and S_2' is 0.09339069. the result is shown below:

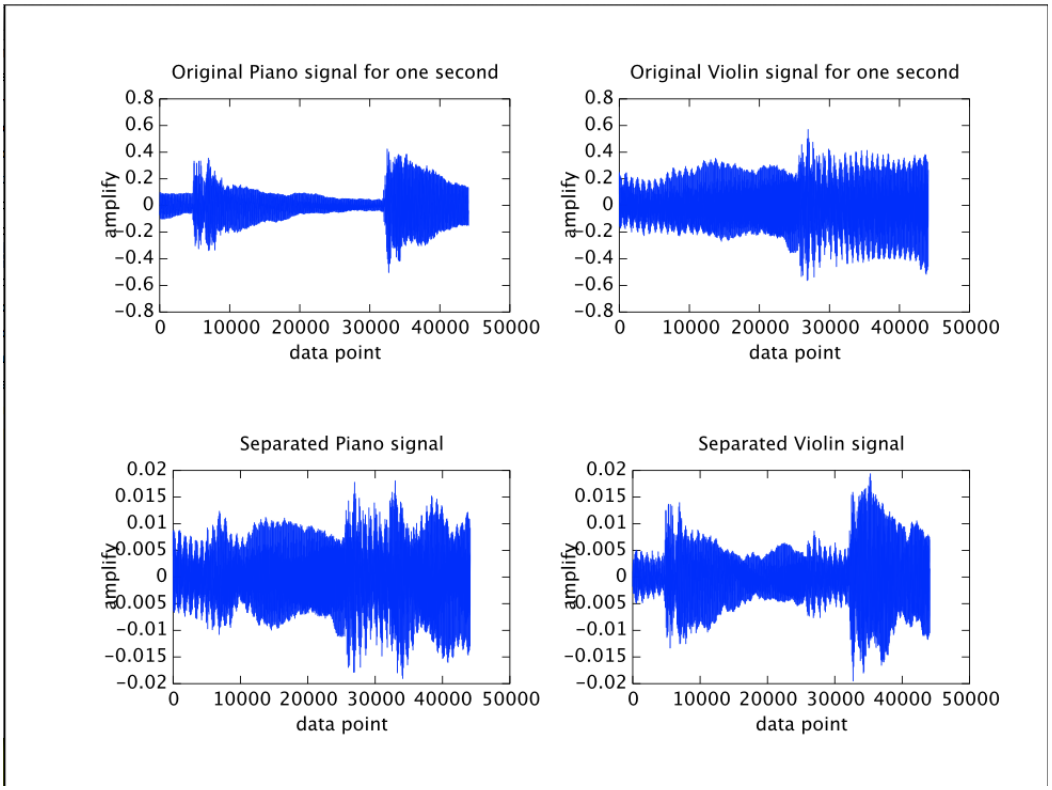

Figure 3.2.6. Top row: source signal S; bottom row: separated signal S'. Difference of S and S': 0.09339069.

I explored different data length and changing the mixing matrix A. The data length are 2, 5, and 10 seconds; the mixing matrix $A = \{\{0.3, 0.9\}\{1.27, 0.95\}, \{\{3, 9\}\{12.7, 9.5\},$ and $\{\{30, 90\}\{127, 95\}$. The table below shows the difference between the original signals and the separated signals:

	0.4 seconds	1 second	2 seconds	5 seconds	10 seconds
Diff of S1 & S2	0.32184394	0.18864145	0.18702481	0.17308345	0.15809225
A= {{0.3, 0.9}, {1.27, 0.95}}	0.22289472	0.12163239	0.12268604	0.15512958	0.10431122
A= {{3, 9}, {12.7, 9.5}}	0.22284694	0.09499229	0.12232556	0.11482994	0.10430606
A= {{30, 90}, {127, 95}}	0.22281135	0.09346783	0.12163433	0.11467226	0.10406017

Figure 3.2.7. the relation between the mixing matrix A and data length

From the table above, we notice that longer data length, the source signal S_1 and S_2 are more similar. And the difference between the source signal S and the separated signal S' is smaller when the mixing matrix are bigger. However, if we listen to the separated signal S', we notice that even the difference is small, both of the separated signals S'_1 and S'_2 still contain the sound of both instruments: recovery of the source signals is not 100%.

3.3 Algorithm FastICA with unknown source signals

I next tested FastICA with source signals mixed by using special arrangement of 2 instruments of 2 microphones. The microphone linearly mixed two instruments sound. I recorded two different instruments together by using two Zoom H4n recording machine. I set up the microphone exactly the same for two recorders. These two recorder are facing

to the players the same way and the distances between one recorder and one player are the same, see figure 3.3.2.

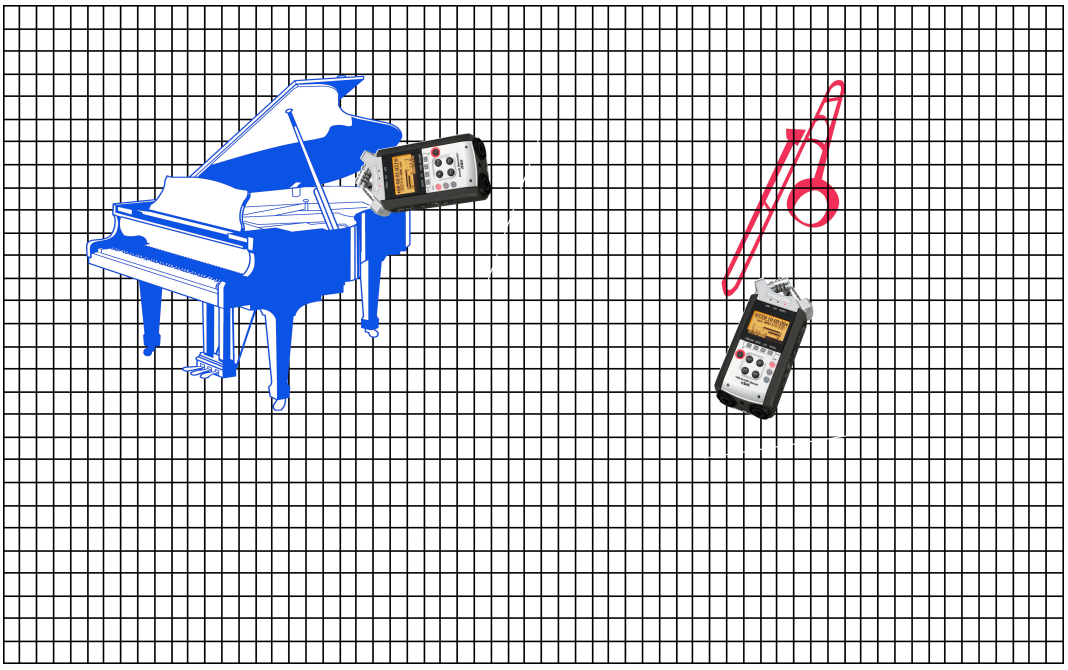

Figure 3.3.1. the set up for two instruments recording. Piano with other instruments. One recorder is 85cm far from piano and the recorder angle is 30 degree. Another recorder is 75 cm far from other instruments. Two recorders are 150 cm far from each other.

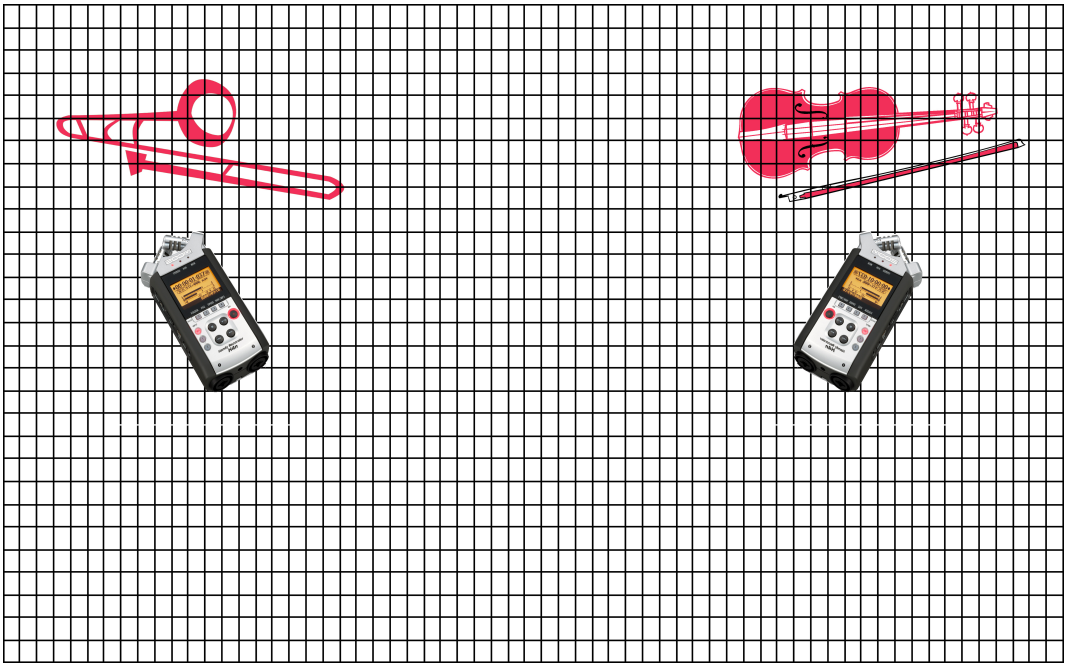

Figure 3.3.2. the set up for two instruments recording. Both of two instruments are 80cm from from the recorder. Two instruments are facing to each other. Two recorders are 230 cm far from each other.

I combined different pairs of instruments. Because the timbre of the instrument affects the result. If I chose two instruments that have similar timbre, which means they have similar waveform, the result is not as good as two instruments with different timbre. For example, flute and piccolo have similar sine-wave-like waveform as in Figure 3.2.2. From the Figure3.2.3, we know that piano's waveform and violin's waveform are more similar than other instruments. Also, the result from the previous section, we know that the pseudo mixed signal of piano and violin gives the best separated signal result. Hence, the figure below shows the real-time mixed signals of piano and violin and the separated piano and violin signals.

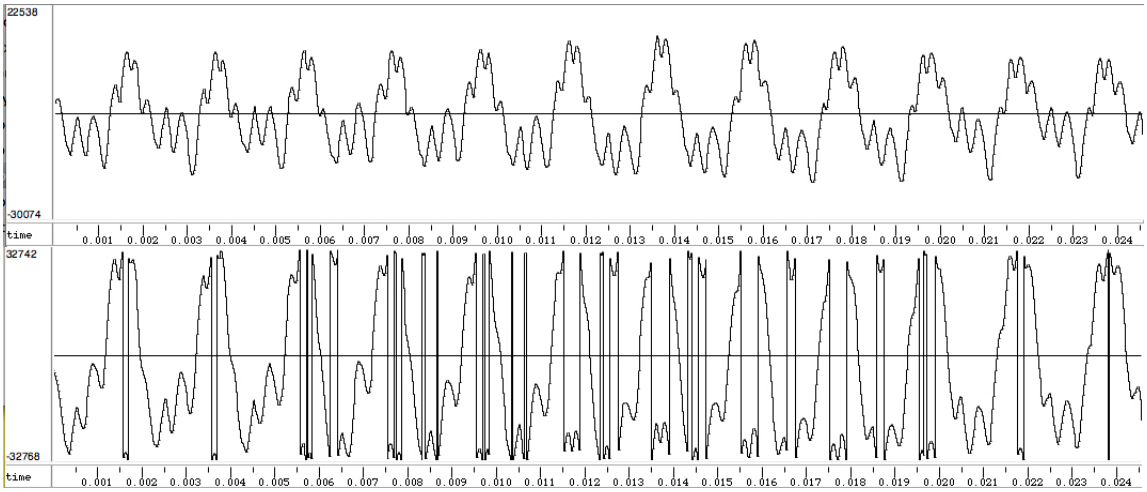

Figure 3.3.3. The mixed signal X_1 and X_2 of piano and violin, where mixing matrix $A =$ 0.3, 0.91.27, 0.95. X axis: time; Y axis: amplitude

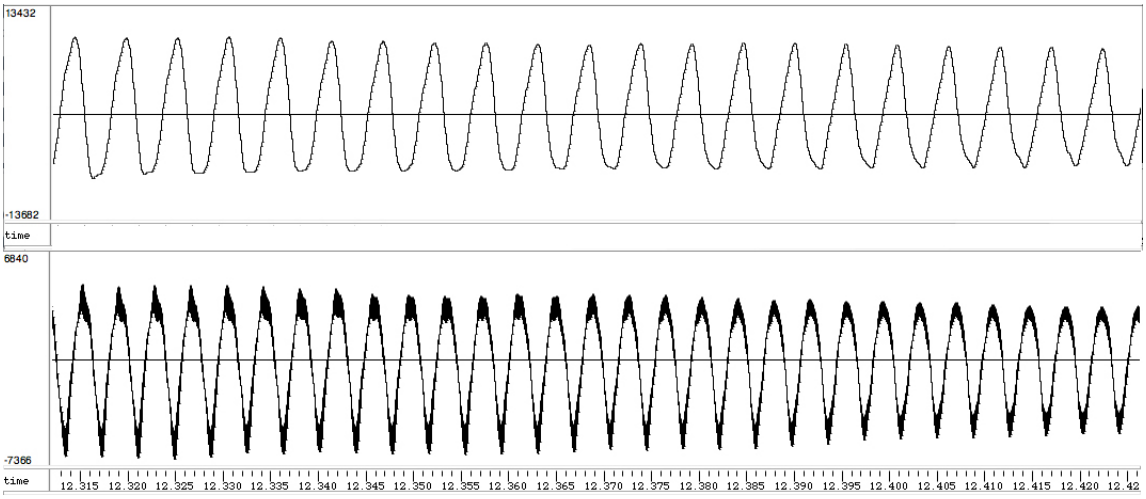

Figure 3.3.4. Comparison of original piano wave form and separated piano waveform from real-time mixed signal of piano and violin. Top: original piano waveform; Bottom: separated piano waveform. X axis: time, 2.5 second; Y axis: amplitude

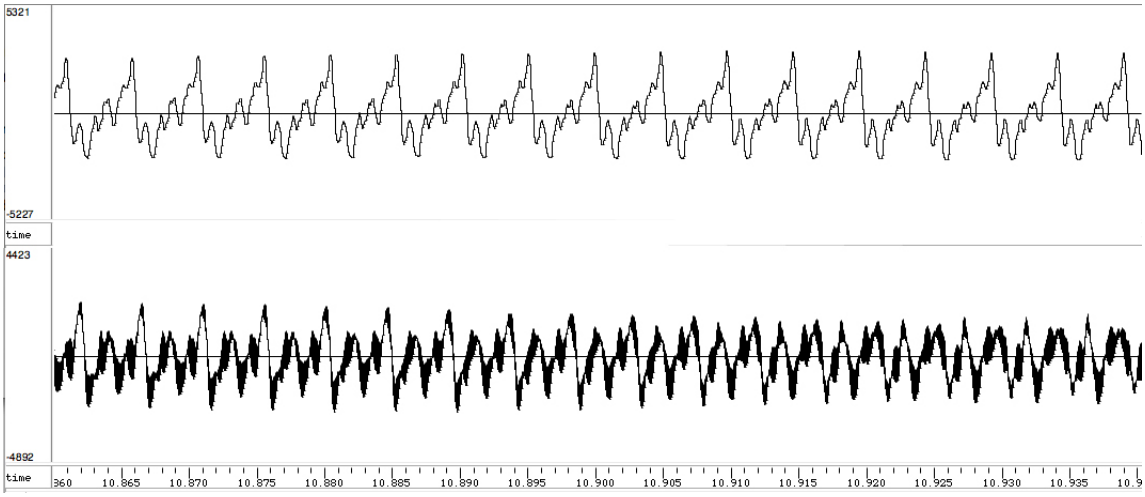

Figure 3.3.5. Comparison of original violin wave form and separated violin waveform from real-time mixed signal of piano and violin. Top: original violin waveform; Bottom: separated violin waveform. X axis: time, 2.5 second; Y axis: amplitude

The difference between the original piano waveform and the separated piano waveform from the real-time mixed signal is 11%, which means they are 89% similar to each other. The difference between the original violin waveform and the separated violin waveform from the real-time mixed signal is 31%, which means they are 69% similar to each other.

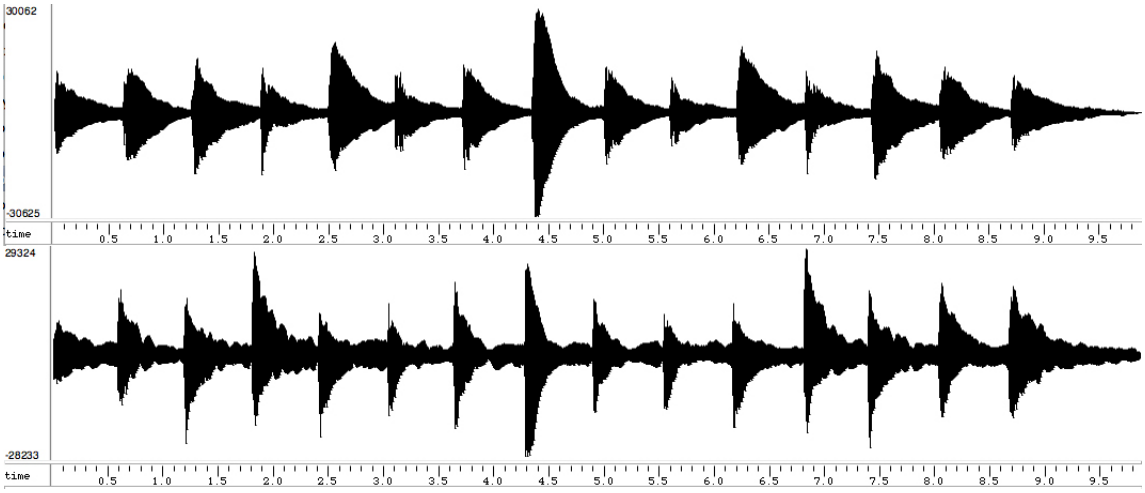

Figure 3.3.6. Comparison of original violin wave form and separated violin waveform from mixed signal of piano and violin. Top: original violin waveform; Bottom: separated violin waveform. X axis: time, 10 second (entire recording); Y axis: amplitude

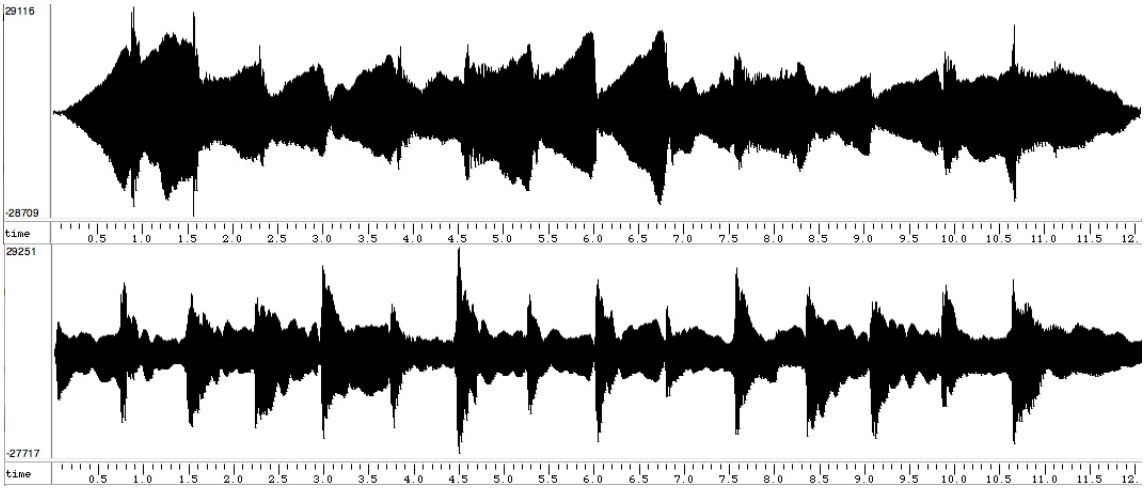

Figure 3.3.7. Comparison of original violin wave form and separated violin waveform from mixed signal of piano and violin. Top: original violin waveform; Bottom: separated violin waveform. X axis: time, 10 second (entire recording); Y axis: amplitude

However, form the figure 3.3.7 we see the waveforms of original violin and separated violin are very different. In fact, it's more closed original piano's waveform. Since piano's waveform and violin's waveform are similar, the difference between the separated violin waveform and original piano wave form is 14%, which means they are 86% similar to each other.

From the previous section, we know that clarinet and oboe have the most different waveform and also the worst separated signal result from sudo mixed signal. For real time mixed signal, the difference between the original clarinet and the separated clarinet waveform is 34% and the difference between the original oboe and the separated oboe waveform is 49%, see Figure3.3.8 to Figure3.3.11. Hence, the separation of real-time mixed signal is not as successful as separation of pseudo mixed signal.

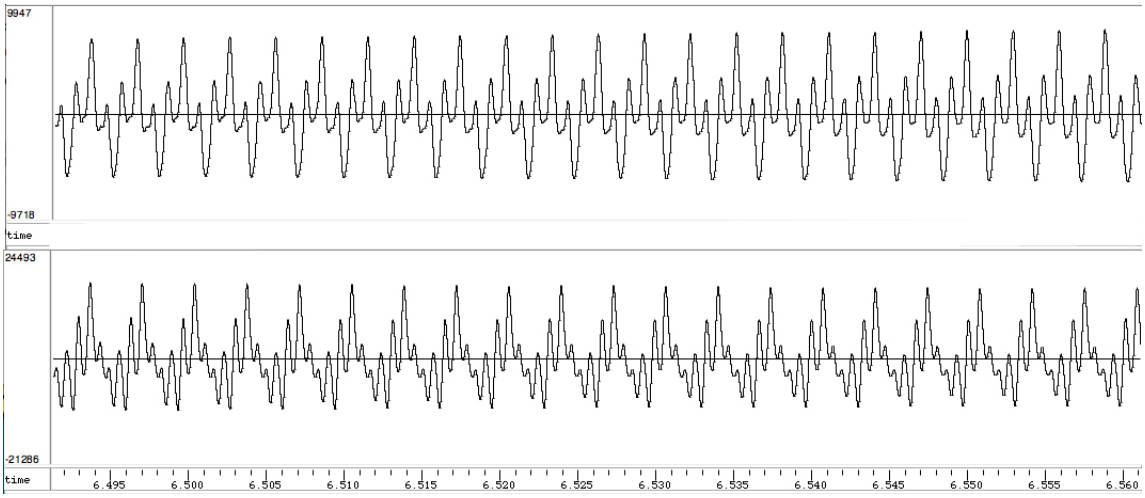

Figure 3.3.8. Comparison of original clarinet wave form and separated clarinet waveform from real-time mixed signal of clarinet and oboe. Top: original clarinet waveform; Bottom: separated clarinet waveform. X axis: time, 8 ms; Y axis: amplitude

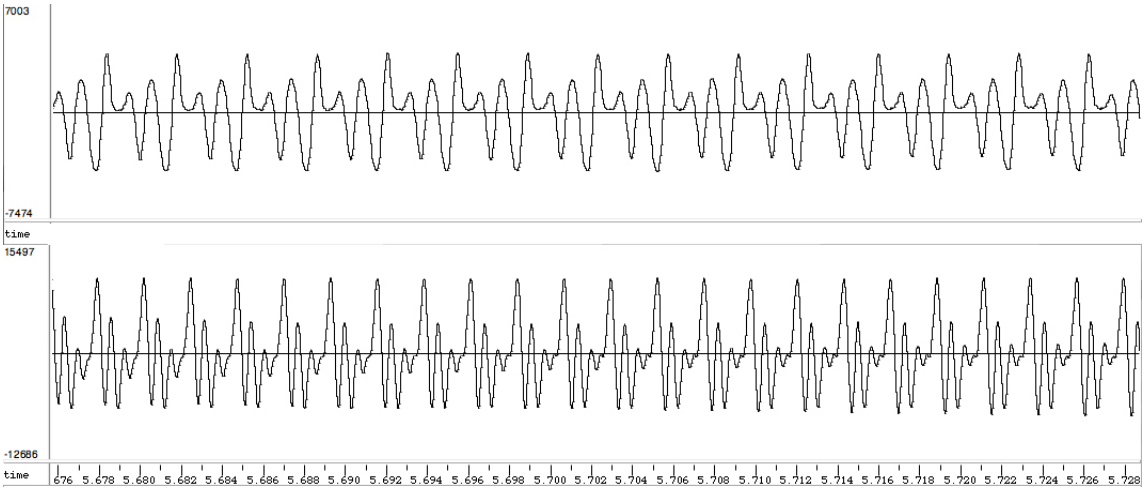

Figure 3.3.9. Comparison of original oboe wave form and separated oboe waveform from real-time mixed signal of clarinet and oboe. Top: original oboe waveform; Bottom: separated oboe waveform. X axis: time, 8 ms; Y axis: amplify

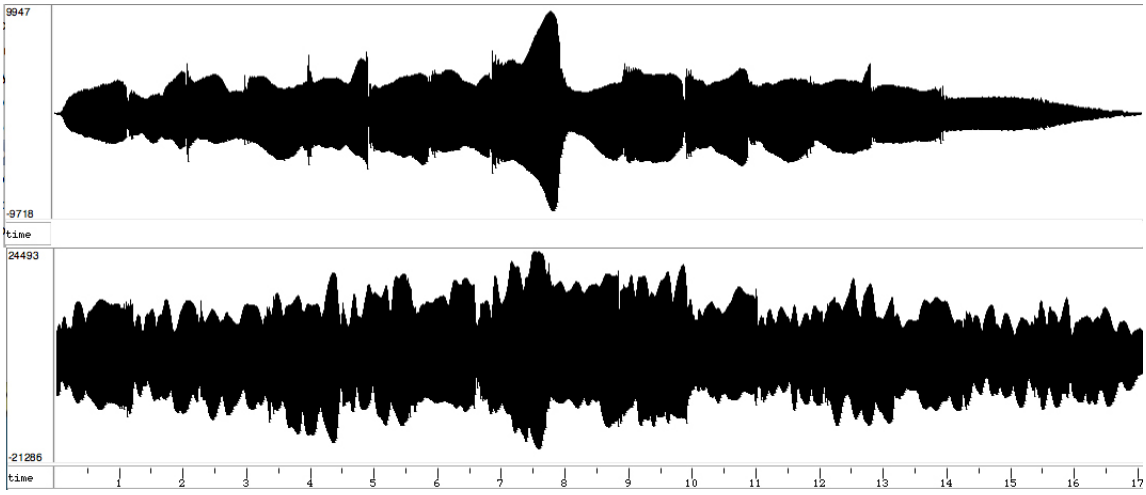

Figure 3.3.10. Comparison of original clarinet wave form and separated clarinet waveform from mixed signal of clarinet and oboe. Top: original clarinet waveform; Bottom: separated clarinet waveform. X axis: time, 17 second (entire recording); Y axis: amplify

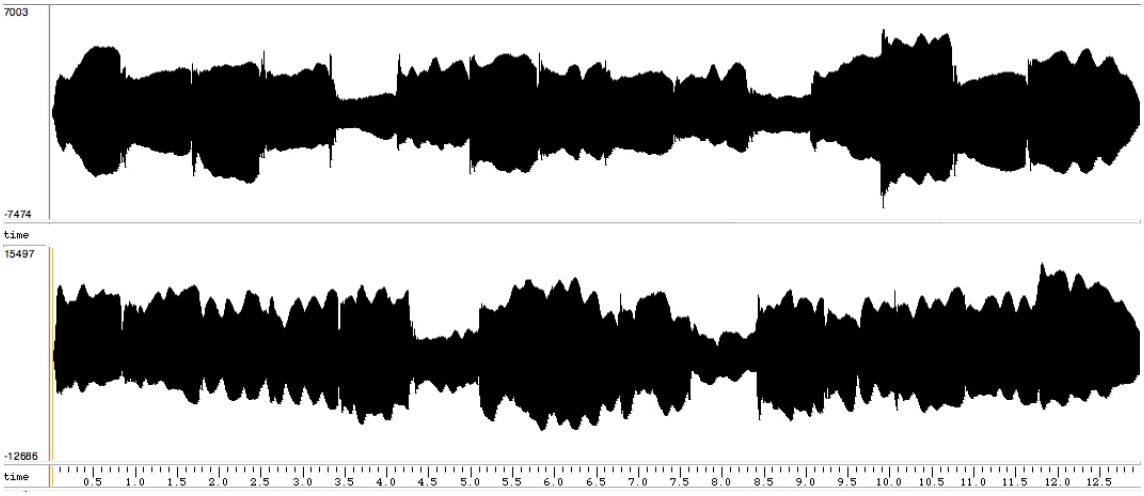

Figure 3.3.11. Comparison of original oboe wave form and separated oboe waveform from mixed signal of clarinet and oboe. Top: original oboe waveform; Bottom: separated oboe waveform. X axis: time, 13 second (entire recording); Y axis: amplify

For the real-time recording, unknown source signal, it is harder than mix the known source signals on the computer. The reason is because that real recording has more different resonants than mixed recording o the computer. When two instruments plays together, the resonant range of the instrument is different from signal instrument playing. Therefore, mixed known source signal is easier to separated than unknown source signal recording.

4

Future Work and Discussion

In this paper, I use only FastICA to separate instruments in shot-duration music recordings. Favaro, Lewis, and Schlesinger (2011) paper, uses FastICA to separate mixed signal. However, they use the Fourier Transform as their mixing matrix A, the equation as below:

$$X_{jk} = \sum_{k=1}^{n} a_{jk}\, exp(-i\, t_{jk} w) S_k \tag{4.0.1}$$

where X_j and S_k are the Fourier transforms of observed signal j and source signal k, respectively, and w_i is the frequency at sample i. Hence the propagation delay in the time domain becomes complex rotation in the frequency domain so the observed signals are now instantaneous mixtures of the source signal. They had success in isolating artificial mixed signals, especially the snare drum and the piano.

Pohle, Knees, Schedl, Gerhard, and Widmer (2006), also use ICA to separate a mixed signal. They also test different length of the music. First, they start from 0.075, 0.15 second, then 0.3 second, then 0.6 second. They use different songs from the ISMIR'04

Genre Classification Contest training set. This set consists of 724 tracks from the six genres classical, electronic, jazz/blue, metal/punk, rock/pop, and world. They also find the best result is from the length 0.075 seconds, which is the shortest period they test. It is the same as my result, the shorter duration is better isolated. The difference between their work and my work is that they use Principle Component Analysis before they implement the FastICA algorithm. PCA increases accuracy by about 4%. They also try to apply a Hamming window to the mixed signal before calculating the ICA. Nevertheless, this additional step does not lead to increased the accuracy in their experiments.

The main difference of my work from others is I only use FastICA to separate the mixed signal. My experimental result shows that shorter length and two instruments with similar waveforms give the best results. However, since I take the data from the middle of the scale, which is the most resonant part for the players, it might affect the performance of ICA. Because, for example, the piano is one kind of the percussion instrument, the strongest part of the note is in the beginning of the note, then its amplitude fades, Figure 4.0.1. But for the violin, because it is the string instrument, the note maintains its wave shape much longer. Usually, the beginning of the note for the violin is the weakest part of the note, Figure 4.0.2.

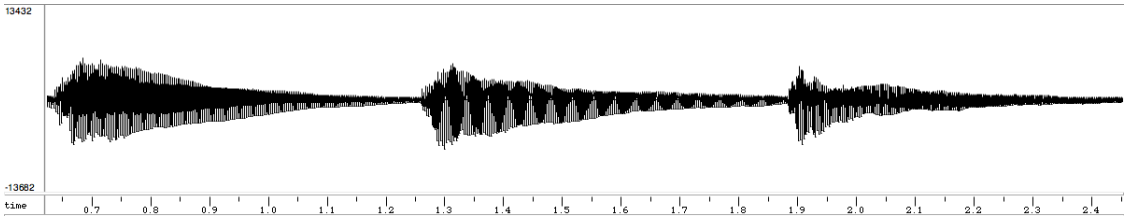

Figure 4.0.1. The waveform of the piano plays three notes. The strength of the note is decreasing, the weakest part is the end of the note, then the strength becomes the strongest part at the beginning of the next note. X axis: time, 2.5 second; Y axis: amplitude

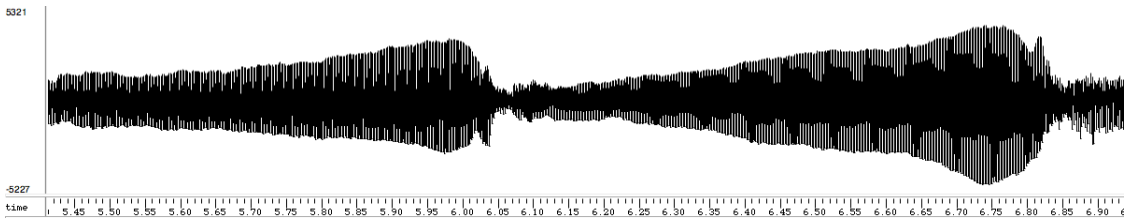

Figure 4.0.2. The waveform of the violin plays two notes. The strength of the note is increasing, the strongest part is the end of the note, then the strength becomes the weakest part at the beginning of the next note. X axis: time, 1.5 second; Y axis: amplitude

If I chose the beginning of every instrument's note for separation, the result might be different. Because the note is not steady and different instrument has different dynamic at the beginning of the note. From the figure 3.2.2, we see the difference between the beginning note in the scale and the middle note in the scale.

Since ICA algorithm cannot separate the mixed signal well if the source signals are similar, my future work will include using different music instead of using the scale. Because if it's a real music, different instruments play different melody. Hence, the per-

formance of ICA may give different result. Also, I will include finding the most effect preprocessing method before implement ICA algorithm to improve my result.

Bibliography

[1] James Boyk, *There's Life Above 20 Kilohertz: A Survey of Musical Instrument Spectra to 102.4 KHz* (2000).

[2] Paul Dawkins, *Linear Algebra* (2007), 305–331.

[3] Antti Eronen, *Musical Instrument Recognition Using ICA-Based Transform of Features And Discriminatively Trained HMMs*, 7th International Symposium on Signal Processing and its Applications (2003), 1–10.

[4] Alex Favaro, Aaron Lewis, and Garrett Schlesinger, *ICA for Musical Signal Separation* (2011), 1–4.

[5] Robert Gavelin, Harald Klomp, Clinton Priddle, and Mats Uddenfeldt, *Blind Source Seperation* (June, 2004), 1–18.

[6] Aapo Hyvarinen and Erkki Oja, *Independent Component Analysis: Algorithm and Applications*, Neural Networks (April 1999), 1–30.

[7] Aapo Hyvarinen, *Fast and Robust Fixed-Point Algorithm for Independent Component Analysis*, Neural Networks **10(3):626-634** (April 1999), 1–30.

[8] Rory Lewis, Xin Zhang, and Zbigniew Ras, *A Knowledge Discovery Model of Identifying Musical Pitches and Instrumentations in Polyphonic Sounds* (2006), 1–10.

[9] Antoine Liutkus, Zafar Rafu, Roland Badeau, Bryan Pardo, and Gael Richard, *Adaptive Filtering For Music/Voice Separation Exploiting The Repeating Musical Structure*, IEEE International Conference **21** (May, 2012), 53–56.

[10] Tim Pohle, Peter Knees, and Markus Schedl, *Independent Component Analysis for Music Similarity Computation* (2006), 1–6.

[11] Santiani Teddy and Edmund Lai, *Model-based Approach to Separating Instrumental Music from Single Track Recordings* (2004), 1–6.

[12] Christian Uhle, Christian Dittmar, and Thomas Sporer, *Extraction of Drum Tracks From Polyphonic Music Using Independent Subspace Analysis* (April 2003), 1–6.