Sound Separation of Harmonic Rhythmic Signals through Principal Component Analysis

Arlow Emmanuel Hergara 13523161[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523161@std.stei.itb.ac.id, [2]arlow5761@gmail.com

***Abstract*—Automatic Music Transcription is one of the most challenging fields of Artificial Intelligence. While it has been studied thoroughly for years, the performance of modern methods still fall short of being satisfiable due to the issues of handling multi pitch and multi instrument audio. A method of separating sounds in an audio signal can be used to improve the accuracy of Automatic Music Transcription techniques. An interesting way to tackle the separation of sounds is through the use of Principal Component Analysis. This paper attempts to achieve sound separation by using PCA.**

***Keywords*—Automatic Music Transcription, Sound Separation, Principal Component Analysis**

I. INTRODUCTION

Automatic music transcription remains one of the most challenging tasks in the field of artificial intelligence and machine learning, and it continues to be an area of active research. Despite significant advancements, current methods still face considerable limitations, particularly in their ability to transcribe polyphonic music (music where multiple notes are played simultaneously). While a variety of techniques have been employed to address this problem, ranging from salience functions to modern deep learning models such as transformers, no solution has yet emerged that can fully satisfy the transcription needs of complex music without human intervention.

The core challenge of automatic music transcription lies in its ability to accurately process polyphonic audio, especially in cases where multiple instruments are involved. In polyphonic music, each note can have distinct characteristics based on the instrument producing it, its pitch, timbre, and dynamics. The overlap of notes from various instruments in a dense mix can make it incredibly difficult for existing transcription methods to separate and identify individual musical elements. As the number of simultaneous notes increases, the complexity of the transcription task grows exponentially, and current approaches often fail to provide results that are both accurate and interpretable. This becomes even more problematic when attempting to transcribe complex musical compositions or performances featuring multiple instruments playing in harmony or counterpoint.

One promising direction for improving transcription accuracy is the use of machine learning techniques. However, even with these advancements, most methods are highly dependent on the specific instruments or genres on which the model has been trained. For example, a model trained on piano music might perform well for piano pieces, but it may struggle when attempting to transcribe other instruments such as strings or brass. This limitation underscores the lack of a generalizable, universal transcription model that can handle the full spectrum of instruments and musical styles. As of now, the development of such a universal model remains an open challenge in the field.

This paper presents an attempt to address some of the shortcomings of current automatic music transcription methods by exploring an alternative approach for sound separation, specifically focused on strictly harmonic music. The proposed method leverages Principal Component Analysis (PCA), a statistical technique commonly used for dimensionality reduction and feature extraction, to decompose a musical track into its constituent notes and their associated characteristics. While not providing full music transcription, the goal is to enable a more efficient and effective means of transcription that is not bound to any single instrument.

PCA has long been used for feature extraction in other domains, such as image processing, where it helps to capture the most important features of visual data while discarding less significant information. In the context of music transcription, PCA holds the potential to extract meaningful features from the complex audio signal, allowing for the identification of the fundamental musical elements like pitch and timbre. This approach draws inspiration from both the traditional method of Non-Negative Matrix Factorization (NMF) and the way PCA is employed in image and signal processing. While NMF has been used in music transcription to separate mixed audio signals into constituent parts, the application of PCA offers an opportunity to explore new ways of modeling the relationships between musical features in a more compact and computationally efficient manner.

The ultimate goal of this paper is to contribute to the development of a more robust and scalable automatic music transcription method that can handle the challenges of polyphony and multiple instruments. By utilizing PCA

to extract a set of harmonically significant features from the music, it is hoped to lay the groundwork for a transcription system that is both versatile and applicable across different musical genres and instrumentation. In doing so, this research aims to move closer to the creation of a universal music transcription system that can be applied to any piece of music, regardless of its complexity or instrumentation.

II. SPECTROGRAM CREATION

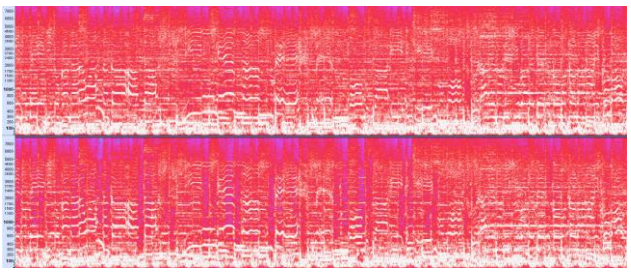

Fig. 1 Example of a Spectrogram in Audacity

The first step to achieve sound separation of music is to decompose the complex audio signal that makes up the signal into a combination of simpler signals at various points of time of the original signal. Typically, sine wave signals of varying frequencies are used to decompose the original signal as it has been proven that all signals can be represented as a combination of sine waves with different frequencies (the number of sine waves within the combination does not have to be finite). Decomposing the original signal as such at various time intervals makes it easier to analyze the frequencies present at a given time period.

To decompose the audio signal as described previously, a transformation known as the Short-Time Fourier Transform (STFT) is employed. More precisely, this paper uses the discrete-time version of the STFT to decompose a discretely sampled audio signal. The discrete-time STFT is a variation of the Discrete Fourier Transform (DFT) which itself is a version of the Fourier Transform (FT) that works on discretely sampled signals.

As a brief explanation, the Fourier Transform (FT) is a transformation that turns a wave function (mapping time to amplitude) into a function of frequency contribution (mapping the frequencies of contributing sine waves to their amplitude and phase shift). Fourier Transform (FT) traditionally takes in continuous wave signals as an input and produces continuous frequencies as an output. The following is the equation representing the Fourier Transform (FT) of an integrable wave function f with ξ as a value representing a specific frequency.

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x}\, dx \quad (1)$$

The result of this equation is a complex number which represents both the amplitude and phase shift of a contributing sine wave with a frequency that corresponds with ξ. The magnitude of the complex number represents the amplitude of the sine wave, while the imaginary part represents its phase shift.

As the audio signals used to store recordings of music are typically not continuous but rather discretely sampled points of the actual audio waves, the traditional Fourier Transform (FT) cannot be used. Instead, a variation of it know as the Discrete Fourier Transform (DFT) is employed. As the name suggests, the DFT is a variation of the Fourier Transform (FT) that works for discretely sampled points of the wave function that are spaced equally apart in time. As the input of the DFT is a finite set of discrete points, its output is also a finite set of discrete frequency contributions ranging from 0 to the audio signal's Nyquist frequency which is half of the sampling rate of the signal. The DFT can be represented as the following equation with k as the wavelength of a sine wave, N as the number of samples and x as the set of samples.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi\frac{k}{N}n} \quad (2)$$

Equation (2) represents X_k as the result of applying DFT on a set of samples x for a sine wave with a frequency corresponding to the wavelength k in terms of samples. While (2) describes the general equation for the DFT, most implementations of DFT uses a class of algorithms known as Fast Fourier Transforms (FFT) to compute the result of the DFT using a more efficient method.

Another issue that needs to be addressed in both the Fourier Transform (FT) and Discrete Fourier Transform (DFT) is the lack of time information in the result. Both FT and DFT are able to identify what frequencies are present in a signal, but neither are able to locate where those frequencies exist within the signal. This is especially problematic in the case of musical analysis as the frequencies representing notes in a musical recording change frequently and the specific location of each frequency is important for representing the audio signal properly. This issue is solved by applying the discrete-time version of the Short-Time Fourier Transform (STFT) on the signal instead of FT or DFT.

Generally, the STFT is an application of FT or DFT with a windowing function that limits the FT or DFT to only a portion of the signal. By sliding this window over the audio signal, it is possible to get the frequency information at different points of time of the original audio signal. The general equation for STFT can be shown as the product of the FT (for continuous signals) or DFT (for discrete samples) and the windowing function. The equation for discrete-time STFT which is the STFT used in this paper is as the following.

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x_n w(n-m) e^{-i2\pi\omega n} \quad (3)$$

Equation (3) represents the discrete-time STFT as the function X with m representing a point of time in the signal and ω representing a value corresponding to a specific frequency. Equation (3) also includes w as the windowing function used to localize the signal to a specific point in

time. While the mathematical representation of the STFT may be a bit complicated, in practice, the STFT only involves splitting the audio signal into multiple chunks, applying the windowing function on every chunk, and applying FT or DFT on each chunk separately.

The discrete-time STFT is applied to the audio signal, creating a representation of the signal in terms of time and frequency. Afterwards, a spectrogram of the audio signal is generated using the results of the discrete-time STFT. The spectrogram is generated by squaring the magnitude of each frequency contribution of the STFT. Doing so eliminates the phase shift information which is not needed for the purposes of this paper.

III. PRINCIPAL COMPONENT ANALYSIS

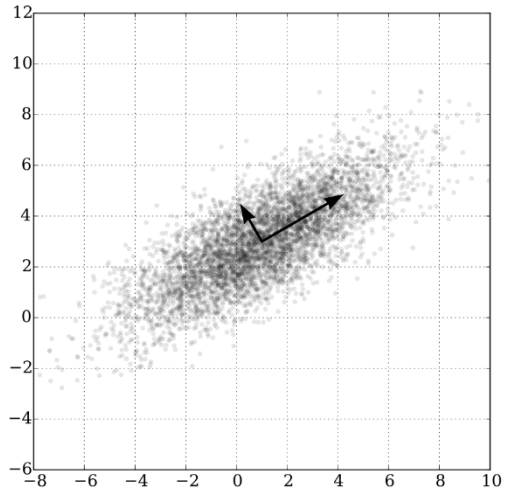

Fig. 2 PCA of a multivariate Gaussian distribution (https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg)

The next step for sound separation is to identify the base notes that make up all the different sounds present at any time in the audio signal. This is done by representing the spectrogram information as a matrix where each row is the spectrogram data at a specific time point and applying Principal Component Analysis (PCA) on the matrix. Doing so projects the original matrix into a space where each point is represented as a combination of principal components that maximize the variance of each data point. By eliminating principal components with little to no variance, it is also possible to filter out noise from the audio signal, resulting in a cleaner representation of the signal.

Principal Component Analysis (PCA) is a method to represent a set of data by its principal components. Principal components are defined as a set of orthonormal vectors that make up axes such that data points projected to these axes have maximum variance compared to all other axes. PCA commonly follows three steps, those steps being data centering, covariance matrix calculation, and Singular Value Decomposition (SVD).

Data centering is the process of positioning the data such that the average for all data points is located at the origin. This is done to ensure that principal components of PCA point at the direction of maximum variance when data points are distributed more towards a certain direction. In practice, data centering is a simple process which only involves subtracting each data point with the average of the dataset. This process can be represented by the following equation where p is a data point, p' is the centered data point, and N is the number of data points.

$$p' = p - \frac{1}{N}\sum_{i=0}^{N-1} p_i \quad (4)$$

Covariance matrix calculation is a step where the covariance matrix of the centered data is calculated. A covariance matrix is a square matrix that encodes the covariance between all pairs of data points in a dataset. Keeping in mind that the data has already been centered, the covariance between two variables x and y can be calculated using the following equation.

$$cov_{x,y} = \frac{\sum x_i y_i}{N} \quad (5)$$

In equation (5), $cov_{x,y}$ represents the covariance between variables x and y in a dataset that's already centered with N representing the number of data points. For the spectrogram matrix, the covariance matrix represents the covariance between each column representing a specific frequency. Such a covariance matrix can be calculated using the following equation.

$$C = \frac{1}{N} X^T X \quad (6)$$

In equation (6), C represents the covariance matrix, while X represents the spectrogram matrix and N represents the number of data points. The covariance matrix has a few interesting properties, such as being symmetric and the diagonal elements being the variance of a column (its covariance with itself).

The final part of PCA is the application of Singular Value Decomposition (SVD) on the covariance matrix. Singular Value Decomposition is a method to decompose a matrix into three other matrices representing a rotation, scaling, and another rotation. The purpose of using SVD in PCA is to obtain a transformation from the original space to the principal component space. SVD decomposes an mxn matrix A into an mxm matrix U, an mxn matrix Σ, and an nxn transposed matrix V such that:

$$A = U\Sigma V^T \quad (6)$$

In equation (6), the matrix Σ is a diagonal matrix of the singular values of A sorted in descending order. The singular values of A can be calculated by finding the square root of the eigenvalues of the matrix A^TA. The matrix V is a matrix of the *right singular vectors* of A which are the eigenvectors of the matrix A^TA divided by their length arranged by column. Similarly, the matrix U is a matrix of

the *left singular vectors* of A which are the eigenvectors of the matrix AA^T divided by their length arranged by column. Both matrices U and V are orthonormal matrices so when there aren't enough eigenvectors to fill the matrices, a normal vector that is orthonormal to all other vectors in the matrix is chosen to fill in the gaps.

After applying PCA on the spectrogram matrix, it is hoped that the principal components of the dataset points in the direction of different base notes that make up all different sounds present within the audio signal.

IV. IMPLEMENTATION

A program was created to try and implement the approach previously explained to separate sounds to verify the viability of the method. The implementation of the approach was created using Python 3.12.4 with the scipy, numpy, and matplotlib packages. In order to run, this program requires two things: a WAV file containing music and the tempo (BPM) of the music.

The program starts by opening the WAV file and extracting the all the samples contained within it. It also extracts the sampling frequency used to generate the samples. If the audio is found to be in stereo, then the program flattens the audio signal by taking the average of the two tracks, turning the audio signal into mono. This is done so that the entire audio track can be represented within one spectrogram.

After getting the sample points, the program calculates the size of the window that should be used when creating the spectrogram. The window size is calculated by the tempo of the music that was specified. The window size is configured in such a way so that the size of the window is equal to the length of a beat of the music. That size is calculated by dividing the tempo in bpm by sixty to get the tempo in bps, then calculating the duration of a beat which is 1 / bps and then calculating the number of samples a window should be by multiplying the duration of a beat by the audio signal's sampling frequency. The window size is defined as such so that each data point in the spectrogram correlates to a specific beat in the track.

After getting the sample points and determining the window size, a spectrogram of the audio signal is generated and shown to the user. The spectrogram displays the contribution of frequencies at different times in terms of decibels (dB). The conversion from the usual spectrogram value to decibels is done by taking the base 10 logarithm of the value. The conversion to decibels is done to improve the color contrast of the spectrogram. The displayed spectrogram itself is just a way to verify that the audio had loaded in properly.

After displaying the spectrogram, the program proceeds to do PCA on the spectrogram data by performing data centering on the spectrogram matrix and putting it through SVD. The covariance matrix calculation part of PCA has been skipped because it has been found that the SVD alone is enough to generate the principal components. After the program has finished calculating the principal components, its values are outputted to a text file since the result doesn't fit in the terminal.

V. RESULTS

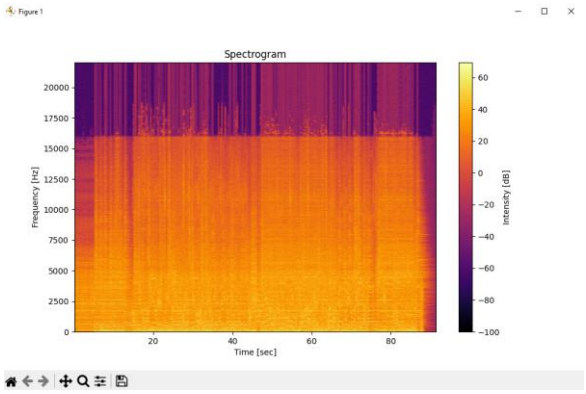

Fig. 3 Spectrogram Result of The Program

The program was supplied with a cut version of the song 醜い生き物 (read Minikui Ikimono) by CHiCO with HoneyWorks. The program described in the implementation section of this paper was able to successfully generate a spectrogram and perform PCA on the spectrogram matrix. It was able to display the spectrogram and the resulting principal components.

Fig. 4 Principal Component Result of The Program

VI. DISCUSSION

The generation of principal components was successful just as the paper predicted. But some problems occurred when trying to use the principal components for sound separation.

First, the ability for principal components to have negative elements does not match how sound waves behave. The appearance of negative elements within the principal components implies the existence of a negative frequency. Of course, such a thing does not exist in this world which means the principal components themselves do not represent the base notes of the music.

Second, the data centering process of PCA makes it so that the principal components do not originate from the origin of the data. In the case of a spectrogram, the origin point represents complete silence. All sounds that are in the audio signal should be created by adding base notes of differing scales to the origin. As the principal components themselves do not originate from the origin, it cannot be said that they are the true base notes of the music.

Third, the base notes of a music do not strictly have to be orthogonal to each other. Orthonormality is a key

feature of principal components, but not that of base notes. As such, not all the base notes might be captured by the principal components. That is if the principal components manage to capture any base notes in the first place considering the two other problems with the PCA method.

From the three problems discovered with this method, it has become apparent that applying the raw PCA method on the spectrogram matrix of a musical recording is not viable for sound separation, even when restricting the music given to strictly be composed of harmonic instruments.

Non-Negative Matrix Decomposition serves to be a better method of automatic music transcription and sound separation as it inherently limits the decomposition to matrices with non-negative values. Such a decomposition is more fit to decompose music as a signal composed of the addition of many simpler signals.

VII. Conclusion

The use of Principal Component Analysis (PCA) for the purpose of sound separation in an effort to expand Automatic Music Transcription (AMT) is at first an interesting thought. But as shown in this paper, the PCA method is not suited for handling and decomposing strictly positive data. A Non-Negative Matrix Decomposition approach is more suited for the problem of sound separation and Automatic Music Transcription without the need of specific instrumental data but more research is still needed in order to achieve satisfiable results.

VIII. Acknowledgment

The author of this paper would like to acknowledge the contributions of his professor for teaching them the basic concepts needed to get a grasp of the topics covered in this paper. The author would also like to acknowledge the contribution of the assistants of the subject for which this paper was made for as they have introduced the author to the topics of this paper. The author would like to acknowledge the contribution of their friends as well, as they have pushed the author into finishing this paper despite certain difficulties. Lastly, the author would also like to acknowledge the contribution of their parents as they have supported the author through all and any situations and have made it possible for the author to write this paper.

References

[1] B. Emmanouil, D. Simon, D. Zhiyao, and E. Sebastian, “Automatic Music Transcription: An Overview” in *IEEE Signal Processing Magazine*, 1st ed. Vol. 36, New York: IEEE, 2019, pp. 20-30. Accessed at 31-12-2024 from: https://ieeexplore.ieee.org/document/8588423

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

Arlow Emmanuel Hergara
13523161