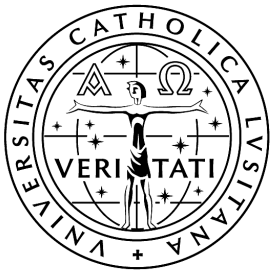


UNIVERSIDADE CATÓLICA PORTUGUESA

The Development of Corpus-Based Computer Assisted Composition Program and its Application for Instrumental Music Composition

Tese apresentada à Universidade Católica Portuguesa

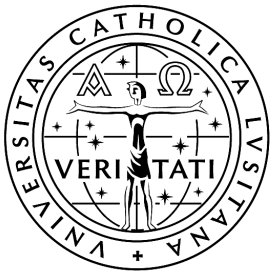
para obtenção do grau de Doutor em (*Ramo Cientifico e Especialidade)*

por

Keitaro Takahashi

ESCOLA DAS ARTES

(December 2017)



UNIVERSIDADE CATÓLICA PORTUGUESA

The Development of Corpus-Based Computer Assisted Composition Program and its Application for Instrumental Music Composition

Tese apresentada à Universidade Católica Portuguesa

para obtenção do grau de Doutor em (*Ramo Cientifico e Especialidade)*

Por Keitaro Takahashi

Sob orientação de Erik Oña

ESCOLA DAS ARTES

(December 2017)

Contents

Abstract

In the last 20 years, we have seen the nourishing environment for the development of music software using a corpus of audio data expanding significantly, namely that synthesis techniques producing electronic sounds, and supportive tools for creative activities are the driving forces to the growth. Some software produces a sequence of sounds by means of synthesizing a chunk of source audio data retrieved from an audio database according to a rule. Since the matching of sources is processed according to their descriptive features extracted by FFT analysis, the quality of the result is significantly influenced by the outcomes of the Audio Analysis, Segmentation, and Decomposition. Also, the synthesis process often requires a considerable amount of sample data and this can become an obstacle to establish easy, inexpensive, and user-friendly applications on various kinds of devices. Therefore, it is crucial to consider how to treat the data and construct an efficient database for the synthesis. We aim to apply corpus-based synthesis techniques to develop a Computer Assisted Composition program, and to investigate the actual application of the program on ensemble pieces. The goal of this research is to apply the program to the instrumental music composition, refine its function, and search new avenues for innovative compositional method.

keywords

Computer Assisted Composition, Concatenative Sound Synthesis, Corpus-based Synthesis, Classification, Decomposition, Labeling, Music Information Retrieval, Segmentation, Self-Organizing Map, Spectral Analysis

Part I
General Introduction

1 Background

The progress of technology has influenced the creation of new art while the creation of new art has demanded the innovation of technology. As the word art also means technology, technological innovation has always stimulated the creation of new art and has also changed life styles. Particularly the invention of the personal computer has hugely expanded the possibilities for creative activities. As the computer technology developed rapidly and extensively, it took over complicated tasks without utilizing a significant number of the labor force and fully equipped environment. We have obtained a great benefit from it to accomplish large-scale works, which were impossible a century ago. Recent breakthroughs in technology for big data and artificial intelligence have allowed the computer to imitate and become a resource of our creativity creating a new artistic works without any interventions by human beings. These results can already be seen in brand-new paintings of Rembrandt, new pieces of Bach and Chopin, or unique hallucination artworks using a corpus of data stored on the internet. Nowadays, art and computer technology are inseparable in terms of advancing the quality of creations.

The creation of music also has received significant benefit from the computer technology not only as a tool but also as a important factor to provoke our creative ideas. This has happened in many areas such as electronic music (use of synthesizers), algorithmic composition (applying mathematical theories), Desktop Music (DTM) (use of sequencers or audio editing programs), and automatic composition by artificial intelligence with big data. One could even say that the influence of computer technology in music is presently remarkable.

The center of our research is the development of a music software which is capable of producing music materials and supporting instrumental and electronic music composition. The software is aimed to produce any kinds of sound sources and to generate creative ideas for users. The research covers the following areas, audio analysis, data classification, music information retrieval, synthesis, and music composition. Although it is extremely meaningful to study all topics at a deep level, our purpose is neither to produce precise and correct synthesis result nor to prove the mathematics problems to

discover a formula. Our goal is to build a system which inspires composers to form their artistic ideas and to support to realize them. Therefore, whenever we investigate any algorithms, we will advance them from the aesthetic point of view as much as possible. This task is particularly inevitable when applying these technologies to the instrumental music composition because the software needs to be flexible to satisfy diverse ideas and the output is expected to be playable by musicians. This discussion needs to be done for diverse topics over various research areas. Therefore, we will avoid describing concrete issues and our remedies here, but will collect them in the corresponding sections.

Before we start discussing our topic, we will preliminarily give a short survey of the history of the relationship between music and technology. It is essential to know how they were interactively developed in order to design appropriate softwares. The technology we will introduce is not only for computers but also for various devices and instruments. The survey is written referring to the following books [1], [2], [3].

1.1 Music and Technology

The development of music is closely related to that of technology since long time ago. The manufacturing technology led to the invention of new instruments with rich and varied expression, which encouraged people to create new music styles. It is interesting to mention that the development of construction technology also influenced music. For instance, it could be argued that large architectures, such as in a church, inspired people strong motivation to manufacture new instruments with excellent acoustics. One can easily speculate that space and vibration in a large architectural space should have accelerated the development of the organ not only with better acoustic quality, but also with more powerful dynamics. This, in order to allow that an instrument placed in the church room, could be appreciated very well and easily by the listeners. At the same time, the number of sound colors increased by using different techniques of producing sounds with various different pipes. It can be imagined that the divine atmosphere of a church and an organ with much resonance, a wide range of amplitudes and a great variety of different sound-colors must have impressed people and changed their perceptions of music.

Printing technology also influenced the situation of music. In the 16th century, printed scores became available and as a result, instruments designed for both use of

accompaniment and solo performances, such as Lute, Clavichord, and Cembalo, became popular. Scores of suites for lute, written in tablature, were published during 1507 to 1536 by printers or musicians such as Ottaviano Petrucci (1466 - 1539) and Francesco da Milano (1497 - 1543). The first printed score for keyboard instruments, Frottole in tabulate per sonare organi were published by Andrea Antico in the beginning of 16th century, in Rome. These pieces were originally created for voices and arranged for instrument solo for the publications. Since then, instruments that were originally played for accompaniment became played independently of the voice. This development definitely changed the function of those instruments and influenced how the music was played. It is possible to say that these changes stimulated the creation of a repertoire for solo instrument and ensemble and simultaneously, it encouraged the improvement of those instruments. Many instruments were significantly developed during the Renaissance and Baroque musical periods, at the end of the 16th century. Register of Recorders became wider while the sound became clearer. New strings instruments were invented, such as the Violin, which compared to the instruments of the family of the Gamba has more stable timbre and pitch, allows for more flexible changes in dynamics, and has a sharper sound. In this period too, a great number of organ pieces were also composed due in part to several important improvements such as the construction of larger organs, which enabled to produce louder sound and the invention of the stop function, which improved the range of registers and the variation of timbres. It should be noted that the innovations of media and the improvements of instruments led to create wide variety in music and eventually made it penetrate more deeply into society.

One of the most unequivocal relationships between developments of technology and music is the influence that the innovation in the manufacturing technology of keyboard instruments had on music composition. In the music composed for Clavichord or Cembalo, dynamics or articulation marks were not written and it could be assumed that the musical expressions allowed by changes in dynamics were scarcely used. Early keyboard instruments had no function to control dynamics and thus could not make loud and soft sounds as modern instruments do. In the beginning of the 18th century, Bartolomeo Cristofori di Francesco developed the Gravicembalo col piano forte, an instrument with a function controlling change in dynamics. After the invention of this early pianoforte, composers began to employ musical expressions using changing dynamics in their music. For example, Haydn began to use a few dynamics and articulation marks in piano

sonata No.42 Hob.XVI. 27, composed in 1770s. It is also possible to find noticeable changes in piano sonatas by Ludwig van Beethoven following the development of the pianoforte. In earlier periods, Beethoven composed sonatas by using an instrument whose register was 5 octaves (F1-f3). We can see how the limitation of narrow register affected his composition of the piano sonata No.17 composed in 1801/1802. According to History of Western Music published by Tokyo Shoseki Co. (2004), the second theme in the exposition of the first movement has an ascending unison motion on the upper voice and inner voice as shown in the figure 1. The theme is repeated in different key in the recapitulation. It is noted that the upper voice does not rise as the inner voice does but stays in d3 because the highest note of his piano forte was f3, otherwise the upper voice should rise until b3 in figure 2. In 1803, Beethoven obtained a new pianoforte with wider register 5 octaves and a perfect 5th (F1-c4) manufactured by Sebastian Erard (1752 - 1831). With this pianoforte, he composed the piano sonata Op.57 with a wide register in which he created heavier and more powerful sound as shown in the figure 3. As mentioned above, the development of technology has improved music instruments, what has enabled an extended musical representation.

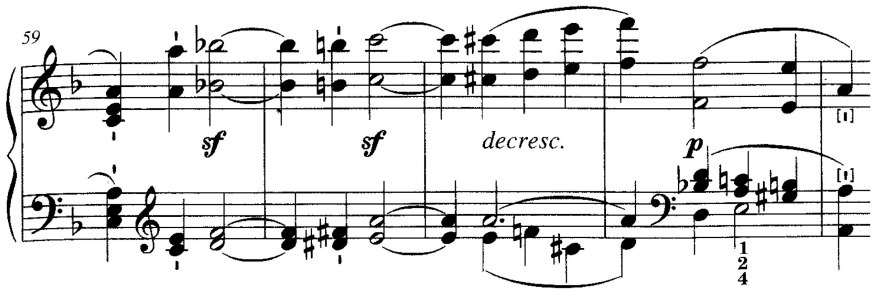


Figure 1: L.V.Beethoven Piano Sonata No.17 in 1801/1802

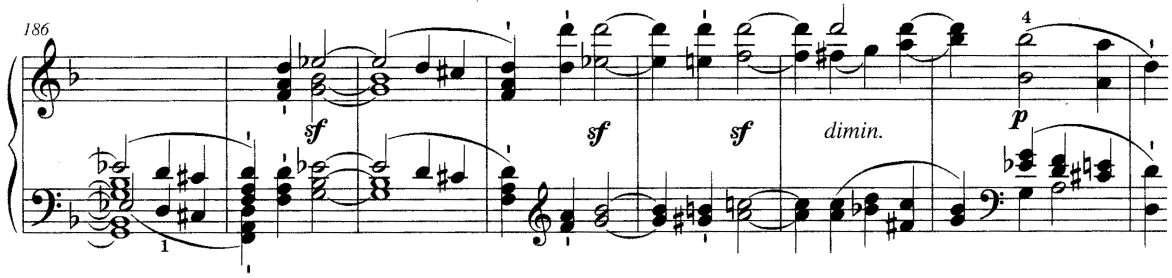


Figure 2: L.V.Beethoven Piano Sonata No.17 in 1801/1802

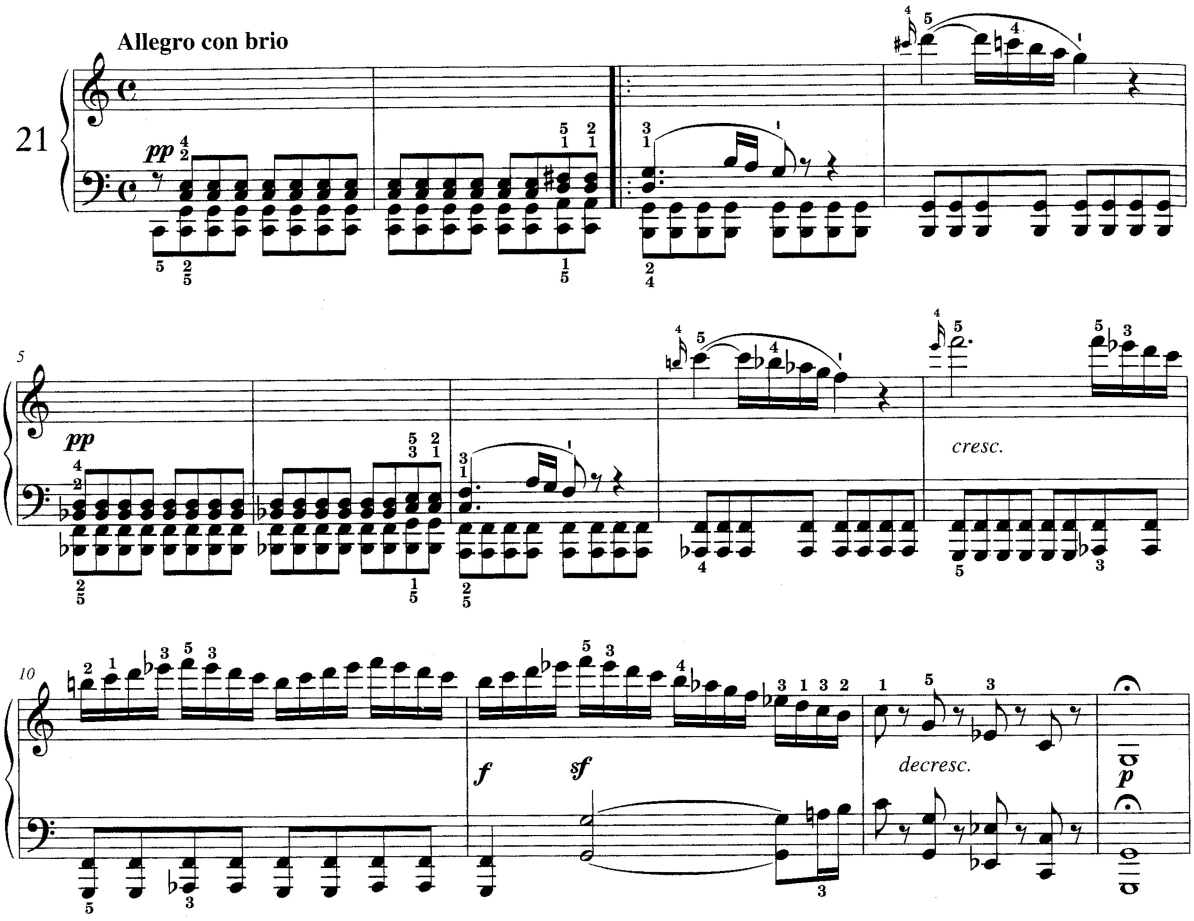


Figure 3: L.V.Beethoven Piano Sonata NO.23 in 1806

Besides the development of the instruments, non-instrumental sounds have been also employed in music over the past century due to the development of the recording and radio systems. The first person that proposed the use non-instrumental sounds,

such as environmental noises, in music was Luigi Russolo (1883 - 1947) who was an Italian futurist painter and composer. His manifesto The Art of Noises (Larte dei rumori) was made in 1913. Russolo went on to create noise-generating instruments called Intona rumori with which he performed noise music in public. Subsequently, Edgard Varése (1883 - 1965) composed Ionisation (1929 - 1931), which is one of the first pieces written using only percussion instruments. In this piece, Varése created a new style which employed not only the sense of pitch or rhythm but also spatial position, the characteristic timbre of percussion instruments, timbre associations created by means of unconventional orchestration concepts, and the effective use of silence. This new music style was truly original in those days. In the 1940s, Pierre Schaeffer(1910 - 1995) and Pierre Henry(1927 - 2017) invented Musique concréte, a form of the electronic music, which was made by means of cutting, looping, filtering, reversing, and changing the speed of recorded sounds such as human voices, animal calls, variable environment sounds, and instrumental sounds. They created musical motifs, gestures, rhythms, or acoustics-spaces by using those recorded sounds and assembling them in a sort of collage-art.

Electronishe Musik, a purely electronic music, employing electronic sound synthesis and processes such as electronic filtering, frequency modulation, ring modulation, et cetera, was developed at the studio of WDR in Cologne, Germany, around 1950. From 1953, the composer Karlheinz Stockhausen(1928 - 2007) began to be engaged in electronic music at the WDR studio and created one of the earlyst electronic music pieces Studien I/II in 1954. In another early work of electronic music, Gesang der Jüglinge (1955 - 1956), Stockhausen employed both electronic and recorded sounds such as a boys voice. After this piece, the differences between Elektronishe Musik and Musique concréte gradually faded away. Around that time, in 1947, Dennis Gabor(1900 - 1979) developed the theory of Granular Synthesis from his research on how human beings communicate and hear. Dennis Gabor studied a method to treat every type of analog sound by quantum physics and constructed a sprocketed optical recording system with which he made some practical experiments in time compression and expansion, and pitch shifting. Granular synthesis is a technique to modify time-stretching, pitch-shifting and spatial organization of grains, which are audio data divided into short fragments of 1-100ms, and synthesize sounds by assembling those modified grains. Each grain is played individually or successively in order to create a soundscape or sound texture

consisting of a mass of granular sounds.

Iannis Xenakis was the first composer who employed Granular synthesis technique in his musical work for chamber instruments such as Analogigue A-B for string orchestra and tape. He created granular sounds using analog tone generators and tape splicing. The composition method using such a grain of sounds are called as Micromontage Composition[69].

The rise of the computer greatly expanded the possibility of applying these synthesis techniques and electronic sounds on composition. The computer treats a sound as the digital signals which can be recorded, played, manipulated, and synthesized and thus, most of the tasks that used to be done by hand, such as, cutting and connecting tapes and changing the rolling speed of the player became digitalized. The computers in early era were hardly usable by composers because they were too large and expensive. Later, with the innovation of the transistor, they became significantly compact. After the appearance of the personal computers, the development of the electronic music has been hugely accelerated.

1.2 Computer Technology and Music

Computers and the process of digitalization both are technologies that have, in the last 50 years, entered in a productive feedback loop with artistic endeavors. In the case of music, composers have been interested in applying computer technology to the creation of new music from the early days of the computer. The research areas that usually profits from computers are, among others, those requiring the processing of great amount of data. This generates at the same time the need for research of algorithms for data processing. Composers are thereby able to investigate various new possibilities in their compositional process with the support of computers, and this expands the horizons of compositional activity. Vice versa, developments in computer technology are stimulated by the demands that music creation makes upon it. To composer employing the support of computer s is identified under the name of Computer Assisted Composition or Computer Aided Composition[1]. In this area, computers are used not only to simulate compositional ideas, but composers also apply various algorithms in

[1]Computer Assisted Composition/Computer Aided Composition : OpenMusic[85] developed by IRCAM is one of the most famous software developed for the use of Computer Assisted Composition. OpenMusic is a Visual Programming Language which has a Graphic User Interface and creates programs by connecting each functional module together. Another software which has a similar programming environment is Max/MSP[86] developed by Cycling74 which has also a signal-processing and an image processing programming environments. A real-time audio synthesis software, SuperColider[87] developed by James McCartney has been employed in varied algorithmic composition.

order to build new compositional systems more easily. For instance, algorithms such as theory of progression, probability, signal processing, are possibly applied to create musical series, fragments, structures, and timbres. Iannis Xenakis is widely known for his use of algorithms in the composition of music. He used computers to implement, among others, stochastic process to generate musical materials.

Later, the introduction of the personal computer and development of its performance allowed a diversification in the forms of Computer Assisted Composition. Especially recent progress of the personal computer with parallel processing technology by multi-core processors[2], speedy data communication, and rapid storage devices like Solid State Drives, are making a remarkable improvement to provide higher performance for the processing of great size of data. Due to such innovations in technology, the field of Computer Assisted Composition has been recently granted the possibility of development of programs demanding the process of increasing data set sizes, while at the same time allowing for more direct and intuitive interactions with the user.

Our study of the development of Computer Assisted Composition program consists of three main topics, that are, Audio analysis, Classification, and Synthesis. In this thesis, these three topics are separated in distinct sections and are discussed independently. Although the topics eventually focus on a common purpose, we will discuss the issue and solutions of individual topics. In the subsequent section, we will suggest the application of the program in music composition where we will discuss about the technology and aesthetic issues.

2 Motivation

In this thesis, we will describe a program that supports music creation. The development of our Computer Assisted Composition program demands an integration between several theories and technologies from different fields such as signal processing, signal analysis, neural networks, database construction, information retrieval, perceptual psychology, etc. Therefore, a comprehensive study covering those research fields is significantly required. Furthermore, there are many problems and issues that need to be addressed in order to employ the technology in instrumental music composition. In order to discuss these issues, we will present how this research embodies some of the

[2]There are several frameworks for parallel processing such as OpenMP frameworks for multi-processing of multi-core CPU, CUDA for GPU processing, and OpenCL frameworks which has both functions.

authors compositional ideas, as well as some of the solutions we have developed for applying the technology in my chamber music composition works. The program has been evolving by means of experimenting with it on the composition of concrete pieces, and not on mere exercises. This means that the two activities, programming and composing, have been closely intertwined. While the program might bring new ideas for the musical composition, the composition processes require in turn new functions or improvements of the program. Vice versa, original compositional ideas, might have suggested new ways of using the program or of further developing it. We will also describe some issues and problems that we confronted while using the program, and point towards some possible solutions.

As the computer plays a larger role and produces more concrete and practical results in music composition, it might be criticized that it is the computer that creates music and not the composer. However, computer programs are just processes generally designed to rapidly and efficiently perform a large number of tasks that normally would demand a gigantic amount of time and endeavor if done by hand. For instance, a tremendous amount of calculations are often required for applying mathematics theories to the creation of music. The composition of serial music required composers to create many different forms from each tone row, such as retrograde, inversion, and translation. It is noteworthy that the use of computer calculates these algorithms at higher speed, with more efficiency and with greater accuracy than handiwork. Composers also do not have to dictate their music to scores and do not need to simulate ensemble pieces by playing the piano or asking many musicians to play instruments if the computer can generate their idea on a score and sound data. On the positive side, it could be argued that the use of the computer often makes it possible to avoid personal habits or tendencies of music creation and frequently can bring a new musical point of view. What we especially expect the benefit of using the computer to be that the computer can bring us a kind of objective sight in order to create a musical idea. Even in the case that the computer shows concrete and practical results to composers, a piece finally depends on how the composer develops these musical materials or fragments. The important thing is that the musical materials produced by computer programs can widen the musical view of the composer, and offer a chance to create the new musical materials such as sound texture, timbres, rhythms, and gestures. However, the computer itself does not compose music automatically.

3 Structure of this thesis

As our research topic covers a diverse area of research fields, we decided to discuss them in distinct sections independently. Their introductions, related works, and discussions will be given in each section instead of presenting a comprehensive survey in this preceding section.

This thesis consists of six sections as follows.

Audio Analysis Various audio analysis algorithms which extract the descriptive feature of the audio data are described. Some advanced algorithms adjusting the characteristics of the audio data to be familiar with our perception are also discussed with some examples.

Audio Segmentation/Decomposition The Segmentation and Decomposition section can be seen as a part of the Audio Analysis section, but here, we focus on various algorithms for audio separation in time sequences and polyphonic structures. Some features extracted by the spectral analysis are applied.

Classification The Classification or Clustering contains the theories and mathematical models which organize a tremendous amount of data for an efficient Music Information Retrieval including some tips dealing with the audio segments and fragments separated in the previous section.

Synthesis We discuss the various synthesis techniques employed in our program. Here, we will also present the survey of the related research and the synthesis techniques to our topic including the Concatenative Sound Synthesis which is one of the most fundamental techniques of our synthesis program. We will present basic to advanced algorithms specified for producing particular sound results.

GUI The GUI (Graphic User Interface) section introduces the GUI of our synthesis program developed in three different platforms, Max, Web application, and Cocoa application for MacOS.

Application In this section, we will introduce the actual application of our Computer Assisted Composition program through its use in two of the author's works composed using the results of our research.

Part II
Audio Analysis

4 Introduction

In this section, we introduce various algorithms for audio analysis, namely, FFT-based analysis, feature extraction, audio segmentation, and audio decomposition, which will be used throughout this thesis. Our aim here is to present a list of related algorithms and mathematical models. We also summarize our experiences with improvements to the algorithms, but will provide a more elaborate discussion with our conceptional ideas of the Computer Assisted Composition in other sections such as the Classification and Synthesis. Therefore, we will compile this section as a sort of dictionary for digital signal processing(DSP) methods for the entire thesis.

Audio analysis is the primary topic of our study. After we collect a corpus of sounds, we first need to analyze them in order to extract a set of features representing the characteristics of the audio signals. These results influence significantly the audio identification, classification, and eventually, the synthesis phases. Since there is no omnipotent algorithm to extract a set of features, we will discuss both strengths and weaknesses of diverse algorithms, and simultaneously, we will propose some possible improvements and applications for our audio analysis program.

There are various descriptive features representing the characteristics of audio signals. For instance, the magnitude feature represents a dynamics of a sound, fundamental pitch represents a frequency of a note, and centroid represents a brightness of a timbre. Since each feature describes only one aspect of a sound, we need to combine several descriptive features in order to identify the characteristics of the audio signals. The combinations need to be carefully considered depending on the type of sound and our purpose. For instance, a piano sound can sufficiently be represented by a combination of magnitude, centroid, and pitch, because it consists of a harmonic sound structure and fits in a chromatic pitch system. In contrast, a percussive sound such as cymbal or bell, comprise of inharmonic sound structure and has less sense of a pitch and thus, it can be represented by magnitude, centroid, and noisiness. It is significant to investigate what descriptive features can represent what kind of property of a sound and it is important to find the effective combinations. It is also significant to analyze

time structures and polyphonic constituents of the audio signals in order to reveal the sound properties.

All algorithms and graphs presented in this section were experimented and produced by the programs we have developed in the present study. We acknowledge that we have used the Discrete Fourier Transform(DFT) function provided by vDSP Acceleration frameworks[3] and fft1d function by T. Oura and H. Kawashima[4] which is the most primary function in the audio analysis. Besides DFT function, all other algorithms were developed in C and Swift by ourselves. It is known that the DSP libraries such as Matlab[5], numpy(Python)[6], and Julia[7] enable us to develop complex FFT-based analysis program without primal coding tasks and even without deep expertise for the DSP analysis. However, our goal of the study includes developing the original analysis program and adjust and implement their specific properties for our synthesis program which is aimed to support the chamber music composition. Therefore, we did not use the above-mentioned isolated analysis packages, but tackled diverse mathematical problems and made the source codes from a primal level.

4.1 High and Low level criteria

There are two levels in the analysis criteria of audio data; the low-level provided by computer algorithms and the high-level provided by human evaluation. The low-level analysis is a sort of signal analysis to which is possible to give regular criteria that are in a sense objective that is, not influenced by human evaluations. These analysis values are one of the most significant criteria for the computer to appropriately organize a vast amount of data. However, they are not always suited to human perceptions. The use of high-level criteria can interpolate the sense of human perceptions into this drawback of the low-level analysis. Examples of high-level criteria are: how the sound is created (idiophones, membranophones, chordophones, etc.), the material of the instrument (wood, metal, paper, etc.), or personal impression of the generated sound[1]

Another kind of high-level criteria includes emotional expressions such as happiness, sadness, anger, solemnity, tenderness, fear, etc. These have also been implemented in

[3] `https://developer.apple.com/documentation/accelerate/vdsp`
[4] `http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html`
[5] `https://www.mathworks.com/products/matlab.html`
[6] `http://www.numpy.org`
[7] `https://julialang.org`
[1] In the case of CHARTR [?], speech synthesis program, human evaluations are applied to detect the speech rhythms and intonations.

the development of computer programs to control some parameters or musical events while the music is playing [2]. The sound evaluations using emotional expressions have great possibilities to get analysis information which is very close to human perceptions. On the other hand, the sense of human expression differs depending on persons, nationalities, genders, cultures, races, and so on. Some of the high-level criteria such as the ways of generating sounds, rhythms or intonations have relatively universal validity. However, the use of human expressions still presents many unsolved issues. We employ only low-level criteria. The use of high-level criteria is part of our future research plans.

5 Fourier Transform Analysis

In the first step of the audio analysis, we transform the given audio signal to the frequency domain by means of the Fourier transform. The Fourier Transform is the most primal algorithm for the audio analysis we will discuss, and in fact, it is employed in a fundamental process of the most of our analysis algorithms. The most basic purpose of the analysis is to represent the given signal by a sum of weighted sinusoidal functions which are called spectral components. The resulting decomposition unfolds a snapshot of the density of spectral components whose temporal variation is averaged over time. This frequency density is also called a spectrum. The spectrum becomes a key to describe the various characteristics of the sound as well as to reveal its structures. Here, we will introduce variable Fourier Transform functions and the representation ways of the resultant spectra.

5.1 Discrete Fourier Transform(DFT)

Let kth Fourier Coefficient $X(k)$ consisting of *real* and *imaginary* values, and then it is calculated by Discrete Fourier Transform (DFT) with the following formula.

$$X(k) = \sum_{n=0}^{N-1} x(n) exp(-2\pi ikn/N) \tag{1}$$

where $x(n)$ denotes a sequence of wave data ($0 \leq n < N$) and N denotes FFT frame size. With regard to these parameters, the Inverse FFT is given by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) exp(2\pi ikn/N) \tag{2}$$

We describe the DFT function with the symbol $\mathcal{F}$ and then 1 is represented by the following,

$$X(k) = \mathcal{F}[x(n)] \quad (0 \leq n < N) \tag{3}$$

The inverse Fourier Transform, the formula 2 is represented by following,

$$x(n) = i\mathcal{F}[X(k)] \quad (0 \leq k < N) \tag{4}$$

Where $\mathcal{F}[]$ is the DFT and $i\mathcal{F}[]$ is the iDFT function respectively.

5.2 Short-Time Fourier Transform(STFT)

The Fourier Transform yields spectral information of the signal that is averaged over the entire time domain. While the signal represents the information across time, the Fourier transform represents the information across frequency, and the time domain information becomes hidden. When the given signal is perfectly periodic like a sine-wave, the hidden time information is not problematic because the signal does not change through time. However, the actual signal usually varies in time. To address this issue, Dennis Gabor introduced the short-time Fourier transform (STFT) in 1946, which performs a small section of the signal instead of the entire signal. The small section of the FFT frame is the processing unit of the STFT, and its size is called as the FFT frame size or window size. Because a sequence of the signal is segmented arbitrarily into the frame size, the edges of the segments are usually nonzero. These signal discontinuities produce some errors in the transform. Therefore, the original signal is multiplied by the window function to yield a windowed signal whose edges fade out into zero.

The figures (4, 5, 6) illustrate the process of the windowing. The figure 4 is the Hanning window which is calculated by the formula 6, and the figure 5 illustrates the original signal whose edges are nonzero. The figure 6 is the windowed signal. As seen in the figure 6, some materials near the edges are missing in the windowed signal. To compensate for this, the STFT is performed with the overlap method described in the section 5.7 (Overlap Method).

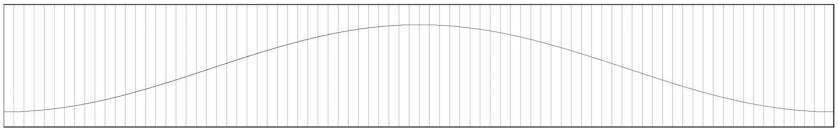


Figure 4: Hanning window function

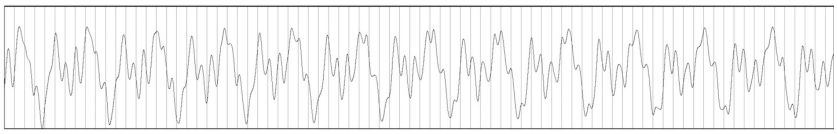


Figure 5: Original signal

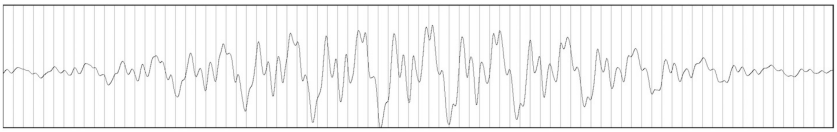


Figure 6: Windowed signal

The windowed short-Time Fourier Transform is defined as following. Let kth Fourier Coefficient $\chi(m,k)(0 \leq k < N/2)$, where N denotes window size, for the mth time frame and then

$$\chi(m,k) := \sum_{n=0}^{N-1} x(n+mH)w(n)exp(-2\pi ikn/N) \tag{5}$$

where n is $(0 \leq n < N)$, k is $(0 \leq k < N)$ and $w(n)$ denotes window function (in this case, we use the hanning window function.). mth frame has $K+1$ size spectral vector. $H \in \mathbb{N}$ denotes hop size. The hanning window function $w(n)$ is calculated as follows.

$$w(n) = 0.5(1 - cos(\frac{2\pi n}{N-1})) \tag{6}$$

5.3 Discrete Cosine Transform(DCT)

Transforms with cosine and sine functions are also useful for the signal analysis. Here we focus on the cosine transform which is more relevant to our research topic. The discrete cosine transform (DCT), firstly introduced by Ahmed, Nataraja, and Rao[7],

uses cosine waves in order to transofrm the signal into the frequency domain by means of summing the signal and cosine functions at different frequencies. While the DFT uses both real and imaginary numbers, the DCT uses an only real number and extract the periodicity of the data series. As the DCT compresses the significant information of the signal in its lower coefficients, it is applied to data compression areas such as in audio (e.g., MP3) and images (e.g., JPEG).

There are several version of the DCT formula, and the most common DCT function is the one-dimensional transform of length N as follows.

$$C(k) = \frac{2}{N} \sum_{x=0}^{N-1} x(n) cos\left[\frac{(2x+1)k\pi}{2N}\right] \quad (0 < k < N) \tag{7}$$

where $C(k)$ is the kth DCT coefficient and $x(n)$ is the targeted signal.

As the DCT compresses the periodic information of the given signal in the lower order cepstrum coefficients which describe the overall spectral shape. In contrast, the pitch and detailed spectral structure are represented in higher coefficients[18]. The zeroth coefficient is usually discarded, as it is a function of the channel gain[18].

5.4 Linear Power Spectrum

The linear power spectrum of the given signal represents a distribution of power into frequency components. In this representation, both power and frequency components consist of linear numbers, and which means, they are not scaled or mapped by specific mathematical models. The STFT formula 5 yields complex value r and i, and the linear power spectrum μ for each mth frame is represented in the formula 8 where k is the index number of FFT bin in the linear frequency domain.

$$\mu_k = \sqrt{r_k^2 + i_k^2} \quad (0 \leq k < N/2) \tag{8}$$

Where $N/2$ is the Nyquist frequency. The linear power spectrum yields graphs shown in figure 7 and 9, where three sin waves consisting of 440Hz, 880Hz, and 1760Hz and a piano sound in C3 are plotted respectively.

5.5 Log Power Spectrum

The log power spectrum is log-scaled linear power spectrum which represents the magnitude of each frequency component by the following formula.

$$log(\mu_k) = 20log_{10}(\sqrt{r_k^2 + i_k^2}) \quad (0 \leq k < N/2) \tag{9}$$

According to the Weber-Fechner law, some of the human senses operate in a logarithmic fashion. Some measurements, especially the magnitudes with a large range of quantities are represented by the logarithmic scale such as earthquake, loudness, and pitch. In music, the logarithmic scale is considered as a criterion which is closer to the human auditory system than the linear one. It is also useful to extract the subtle structure of the spectrum. In fact, the log-scaled power spectrum is more often used in speech recognition to investigate the formant and to detect vowels and consonants(see sections 5.9 and 6.8).

Figures 7 and 8 show the linear power spectrum and the log power spectrum for the signal consisting of three sin waves; 440Hz, 880Hz, and 1760Hz. The figure 9 and 10 are for a piano in C3. The linear power spectrum has only a few peaks prominently in the lower frequency bands. Other minor peaks, especially in the higher frequency bands, are very weak and the form of the spectrum is spiky. In contrast, the log power spectrum has more peaks in entire frequency bands, and this enables to represent the more fine form of the spectrum which characterizes the sound.

Three sinwaves 440Hz, 880Hz, 1760Hz.

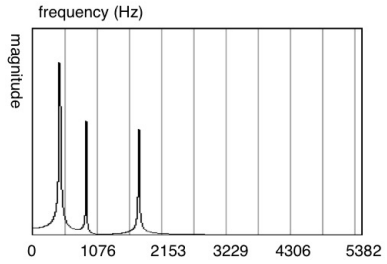


Figure 7: Linear-Power spectrum

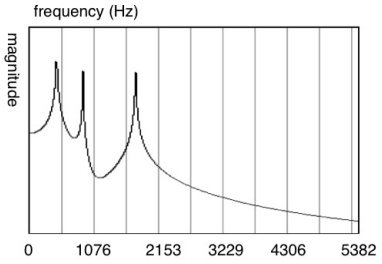


Figure 8: Log-Power Spectrum

Piano sound in C3.

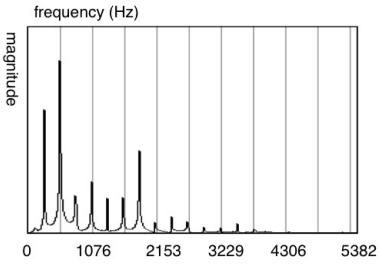


Figure 9: Linear Power spectrum

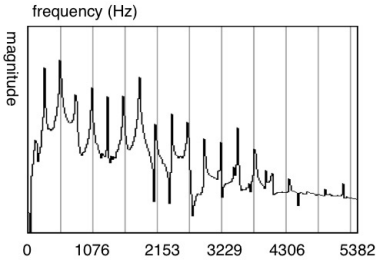


Figure 10: Log Power Spectrum

5.6 Time-Frequency Localization

The FFT analysis represents the signal in two domains; time and frequency. While it can produce adequate results for deterministic and stationary signals, it can not sufficiently represent time-varying signals or non-stationary signals. To realize the time localization, the STFT uses a time-window function to segment the signal into chunks" of small frames, and it operates the transform on those individual frames. As the FFT frame is the smallest unit which segments the audio data, this frame size directly affects the resolution of the time localization. The smaller size of FFT frame can follow smaller changes of audio data through time. However, it has lower resolution in the frequency localization and especially, a profound deficiency in the low-frequency band. On the other hand, the longer size produces better frequency localization, but it is not capable of tracking detailed events occurring in time. In our study, we use the FFT frame sizes from 512 samples (ca. 11.6ms. at a sample rate of 44.1kHz) to 16384 samples (ca.185.8ms) depending on the analysis methods and the targeted sound qualities.

The figure 11 illustrates the relationship between time and frequency localization. The vertical line represents frequency where NF denotes the Nyquist Frequency. The horizontal line represents time in samples. Two bars illustrate the time-frequency resolutions of the STFT analysis with different frame sizes where the left bar is a frame size of 512 samples, and the right bar is a frame size of 4096 samples. The vertical lattice and gradation colors represent the resolution in the frequency localization.

While a certain frame size is imperative in order to obtain the sufficient frequency resolution, it is equally important to track quick sound changes through time. To tackle this contradictory problem, we employ overlap method where each FFT frames

are shifted across the signal by a fraction of the frame size (called as hop size) or another solution would be to use the Multi-resolution FFT described in the section 5.8.

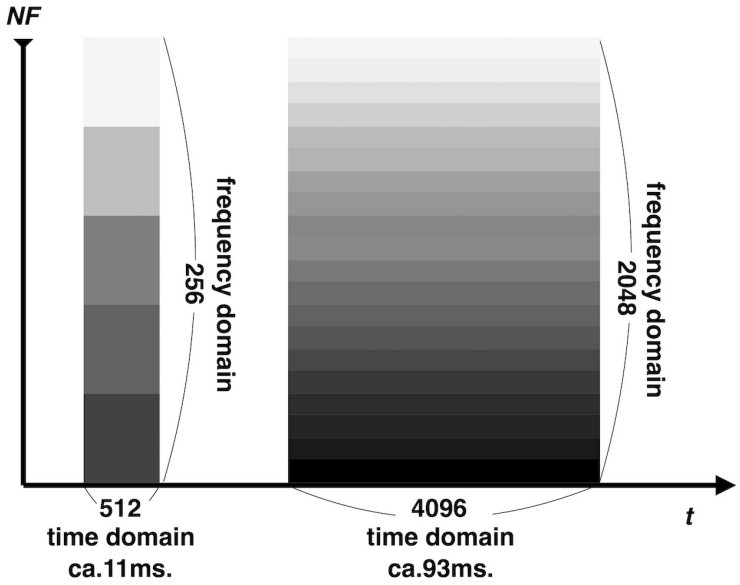


Figure 11: The relationship between time and frequency localization.

5.7 Overlap Method

Figure 12 shows two different analysis methods, the first method just divides an audio data into chunks of FFT frame size, which can be called Normal method, and it makes a chain of FFT frames. The overlap method also divides an audio data into FFT frame size but twice. While the lower chain of Overlap method is the same to the Normal method, the upper chain has been shifted a fraction of half FFT frame size. In the figure 13, the first frame of both Normal method and Overlap method have a short musical event with high dynamics in the first half. However, only one set of analysis values are generated for each frame, which shows an average property of musical events happening in the frame, and therefore it is not possible to represent the change of a fast musical event in one single frame. On the other hand, the first shifted frame in the Overlap method does not cover the short musical event but covers most of the quiet part following it, which holds the later half of the first frame. In this case, the half shifted frame in the Overlap method can interpolate the short musical event by the combination of the first frame. Now, it is possible to detect the exact place of the first short event by using two FFT frames; the first and the first shifted. In this method,

each unit can keep its longer length what allows for a more natural synthesis result. In my program, the overlap method is used. Instead of overlapping two frames as in the previous example, four overlaps are made, the shifting fraction is a quarter of the FFT frame size, this naturally increases, even more, the resolution in time of the analysis. As a general rule, it is necessary to have more overlaps of FFT frames when a longer FFT frame size is used in order to detect faster changes of a sound. The caveat is that higher overlap numbers bring more expensive processing to computers.

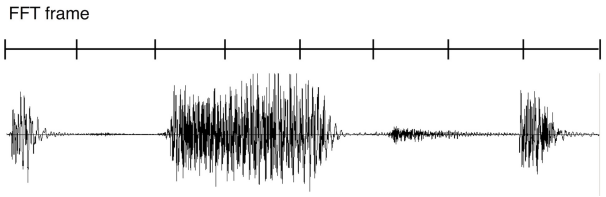


Figure 12: Normal method

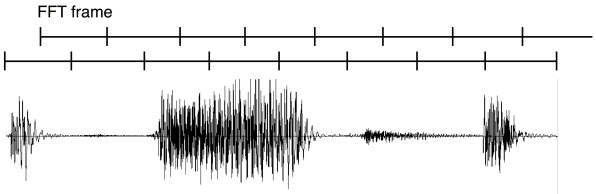


Figure 13: Overlap method

5.8 Multi-resolution FFT analysis

The combination of the STFT and the overlap method can realize better results for representing both frequency and time localizations. Our experiments and practices show that these methods are especially useful when calculating a sum or deviation of the entire spectrum such as magnitude and centroid. However, they are still insufficient for extracting peaks or estimating a periodicity of the peaks, such as fundamental pitch or spectral peaks estimations.

Another solution would be the Multi-resolution FFT analysis which properly selects various frame sizes depending on the frequency components. Since it analyses a particular frequency component using the most appropriate frame sizes, the time and frequency localizations are not consistent throughout an analysis and as a result, it can extract better frequency information from the signal. Due to the employment of the

inconsistent frame sizes, this method is not suitable to produce a spectrum.

We employed the Multi-resolution FFT in the fundamental pitch($f0$) estimation. It operates the analysis with a particular size of frame to estimate a transient f_0. When the accuracy of the f_0 is suspect, it repeats the process using a different frame size which is expected to produce a better result. The detailed process is discussed in the section 6.7 (Multi-resolution Pitch estimation).

5.9 Cepstrum Analysis

Cepstrum analysis, which was firstly introduced by Bogert et. al.[15], is a form of spectral analysis by the Fourier transform of the log magnitude spectrum of the input waveform. The name of Cepstrum is an analogy of the name of "spectrum" where the first four letters are reversed. Similarly, other related words are also derived from the same way such as, quefrency from frequency, rahmonic from harmonic, etc. This is one of the most useful methods to investigate the spectral constitution by the concept of homomorphic (i.e., linear in a generalized sense) mappings between algebraic groups and vector spaces [14]. The Cepstrum is formed by taking the FFT or iFFT of the log magnitude spectrum of a signal. The reason of why FFT and iFFT are interchangeable is because one will just give you a reversed version of the other and thus, both are equally valid for yielding the Cepstrum.

The Cepstrum $c(\tau)$ is defined by the following formula.

$$c(\tau) := i\mathcal{F}[log(|\mathcal{F}[x(n)]|^2)] \tag{10}$$

where $\mathcal{F}$ denotes DFT function and $x(n)$ is the signal in the time domain. $|\mathcal{F}[x(n)]|^2$ denotes the power spectrum of the signal. While the FFT is given in term of frequency, the Cepstrum is given in term of the quefrency which represents pitch lag. The quefrency is represented by the characteristics τ and the magnitude of $c(\tau)$ which is called the gamnitude. The cepstral coefficients $c(\tau)$ describe the periodicity of the spectrum of $\mathcal{F}[x(n)]^2$. When the spectrum consists of multiple regular peaks, it is found in the cepstrum as the number of the coefficient where the peak occurs.

Now we give an example of the Cepstrum analysis which shows how the Cepstrum coefficient is produced depending on the signal.

The six figures (figure 14 - 19) illustrate both spectra and Cepstra of an audio data Cresc_A4.wav which contains a clarinet sound played crescendo in A4 pitch. The analysis is operated within the FFT frame size of 2048 samples, the overlap number of two. The first three figures illustrate the analysis results of the 10th frame(at the position of ca. 107 to 128ms.) and the second three are of the 100th frame(ca.1076 to 1097ms.) respectively. The clarinet sound starts from soft and pure sound qualities and then gradually transitions to loud and slightly distorted sound.

The figure 14 illustrates the power spectrum of the 10th frame, and it yields fewer peaks than the power spectrum of the 100th frame shown in the figure 17. The distinction between the 10th and the 100th frames are more obvious when we see the log-magnitude power spectrums shown in the figure 15 and 18. Compared with the 10th frame, the 100th has more peaks and those of which are standing at the particular interval, in other words, periodically. The distinction of the spectrums is remarkable in the form of the Cepstrums shown in the figure 16 and 19. While the 10th frame have indistinct peaks, the 100th frame have shaper and more prominent peaks, which are marked by the arrows. The result indicates that the 100th frame has a bigger number of the regular harmonic structure than the 10th frame.

In the Cepstrum, when the signal has regularly-spaced frequency partials, it appears as peaks. However, the signals such as pure tone and inharmonic sound do not provide any clear peaks.

The result of the 10th frame

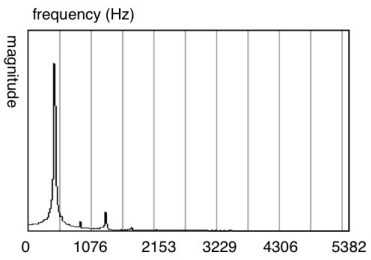


Figure 14: Power spectrum

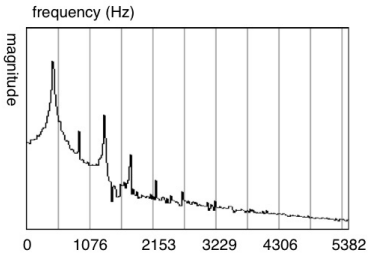


Figure 15: Log-scaled power spectrum

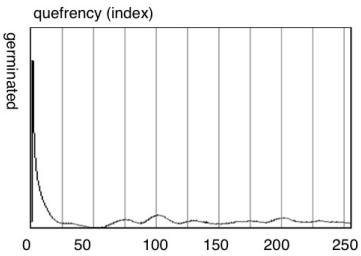


Figure 16: Cepstrum

The result of the 100th frame

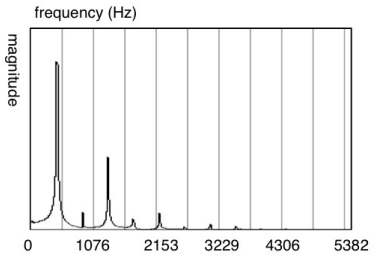


Figure 17: Power spectrum

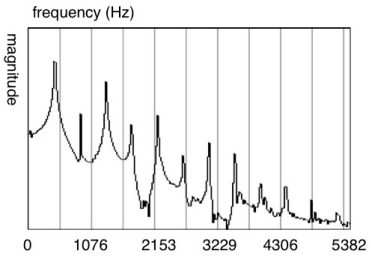


Figure 18: Log-scaled power spectrum

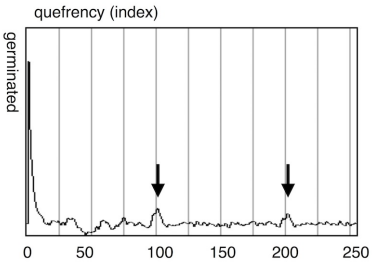


Figure 19: Cepstrum

5.9.1 Liftering

The Liftering is the linear filtering of the log spectrum which separates the various convolutional components, such as separating the vocal tract filter response from the periodic excitation spectrum [14]. The liftering performs similar effects to the Adaptive or Linear predictive coding(LPC) introduced by B.S. Atal in 1967 and the liftering is employed to extract the formants, intensity, and fundamental frequency components from the speech signal [16]. The algorithms are employed to realize the speech compression and speech synthesis for telephone systems.

By applying a low-pass lifter to the Cepstrum, the spectrum envelope which is the slowly varying curve of the spectrum is extracted. The spectral envelope represents the resonance structure of the vocal tract which defines the vowel of the voice. This property of the liftering is also applied to extract the musical instrument formants which represent the character of the instrumental timbre. Therefore, the liftering is also employed for musical instrument identification within the Mel-Frequency Cepstral coefficient (MFCCs) [17] described in the section 6.8.

The figure 20 illustrates the log power spectrum of a sound of a human voice a and the figure 21 illustrates its Cepstrum. The Cepstrum is liftered into two different low quefrency components; the one has the first 50th components(figure 22) and the second has the first 100th components(figure 23). The results of the inverse liftered Cepstra, which are in the spectral domain, are illustrated in figure 24 and figure 25. The figures show that the fewer quefrency components yield a smoother curve of the spectrum which means the vocal tract is more roughly separated from the periodic excitation spectrum. A bigger number of quefrency components can yield more complex and

detailed spectrum curve which is more similar to the original log power spectrum.

Spectrum and Cepstrum of a voice sepaking "a".

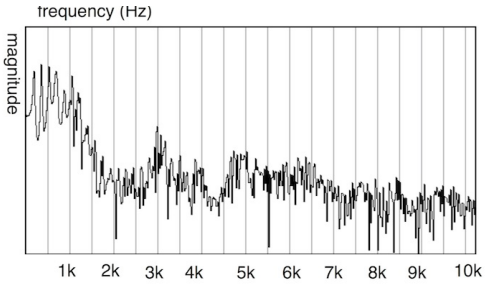


Figure 20: Log Power Spectrum

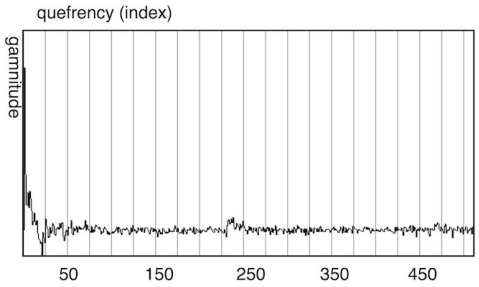


Figure 21: Cepstrum

Liftered Cepstrums

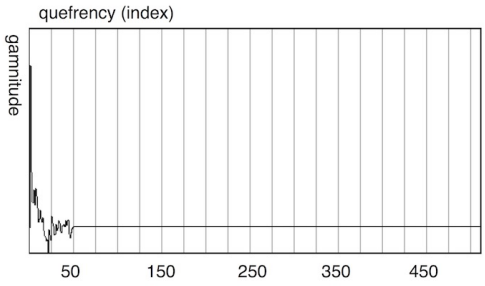


Figure 22: First 50 components

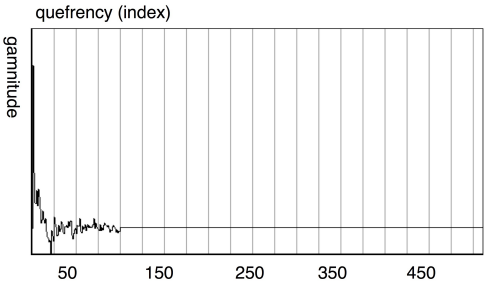


Figure 23: First 100 components

Spectrum envelope

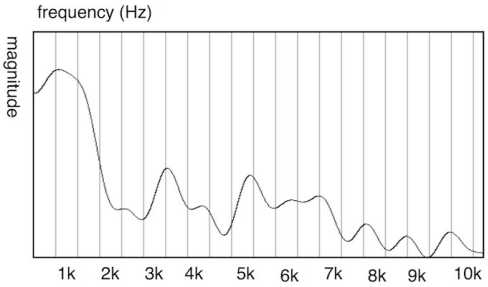


Figure 24: First 50 components

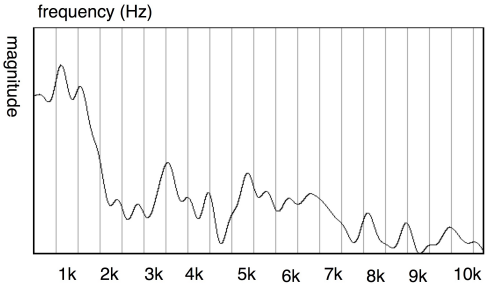


Figure 25: First 100 components

5.10 Log-Frequency Power Spectrum

Until here, we have introduced the spectral analysis with various scaled magnitude; the linear and log-scaled magnitude.

Now we focus on scaling the frequency domain and discuss the linear-frequency and the log-frequency power spectrum. The DFT yields the linear-frequency power spectrum illustrating distributions of the spectral peaks in a constant frequency domain; in Herz. As the spectral coefficients are not mapped efficiently to musical frequencies, which are the ratio intervals called pitches, the frequency resolution is not considered equally by the human perception.

We have already introduced the multi-resolution FFT analysis which realizes the multiple frequency resolutions but the algorithm is not suitable to yield a log-spectrum. The log-frequency power spectrum is, in contrast, an algorithm to yield a spectrum whose frequency resolution is constant in pitches. The nice feature of the log-frequency power spectrum is its increasing time resolution towards higher frequencies and which resembles the situation in our auditory system [11] as seen in the figure 26. Furthermore, the relative positions of the fundamental frequency and the n-th harmonic frequency remain constant no matter how the fundamental frequency fluctuates and are an overall parallel shift depending on the degree of fluctuation. This characteristic is useful for analyzing the harmonic constitutions between different pitches and for extracting the fundamental pitches discussed in the Specmurt analysis, section 5.11.

The log-frequency power spectrum is calculated by the wavelet transform or the Constant Q Transform. Here we would like to introduce the Constant Q Transform.

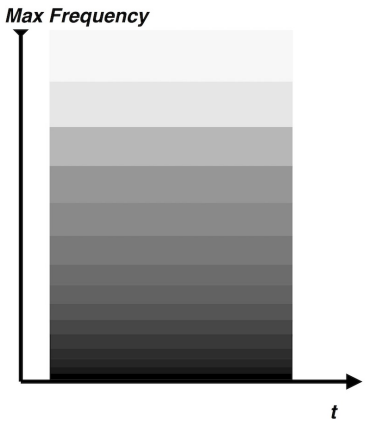


Figure 26: The relation ship between time and frequency localization in Log-frequency spectrum. The vertical lattice and gradation colors represent the resolution in frequency localization.

5.10.1 Constant Q Transform

The Constant Q Transform was introduced by J. C. Brown in 1991. It is a bank of filters which uses a constant ratio of center frequency to resolution. In the Constant Q Transform, the frequencies sampled by the DFT should be exponentially spaced and, when the pitch resolution is quarter tone spacing, this gives a variable resolution of $(2^{\frac{1}{24}} - 1) \simeq 0.029$ times frequency [10]. According to Judith C. Brown and B. Blankertz [10] [11], the frequency f_k is defined by the following formula,

$$f_k = f_0 \cdot 2^{\frac{k}{b}} \quad (k = 0, ...) \tag{11}$$

where b denotes the number of filters per octave. When the resolution per octave is defined as 24, this means quarter-tone spacing of the equal tempered scale, the frequency of the kth spectral component is calculated by the following formula.

$$f_k = (2^{\frac{k}{24}}) f_{min} \tag{12}$$

where f will vary from appropriately chosen (minimal center frequency) to below the Nyquist frequency which corresponds to the lowest frequency. Now, the bandwidth of the filter for kth cq(constant Q)-bin, which corresponds to the FFT bins in the DFT, is,

$$\Delta_k^{cq} = f_{k+1} - f_k = f_k(2^{\frac{1}{24}} - 1) \tag{13}$$

and this yields the constant ratio of frequency to resolution Q as following,

$$Q = \frac{f_k}{\Delta_k^{cq}} = (2^{\frac{1}{24}} - 1)^{-1} \tag{14}$$

With regard to these parameters, the length of the window in samples at frequency f_k is defined as following formula,

$$N(k) = Q(\frac{f_s}{f_k}) \tag{15}$$

where f_s denotes the sampling rate. The number of cq-bin $K \in \mathbb{Z}$ is defined with the pitch resolution $b = 24$ as following,

$$K := b \cdot log2(\frac{f_{max}}{f_{min}}) \tag{16}$$

With regard to these parameters, and with the frequency range parameter (f_{min}, f_{max} = 56Hz, 7604Hz), the bandwidth filterbank is illustrated as shown in the figure 27.

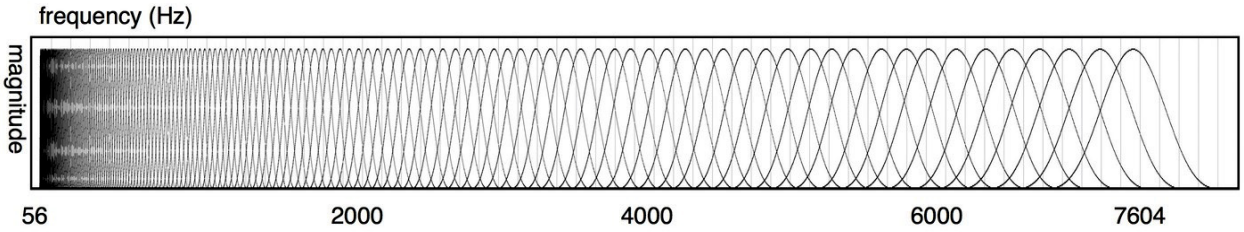


Figure 27: Constant Q Filterbank

Now the constant Q transform for kth spectral component is,

$$X(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} W(k,n)x(n)exp(-2\pi Qin/N(k)) \tag{17}$$

where the digital frequency of the kth component is $2\pi Q/N(k)$ and $W(k,n)$ denotes window function. The calculation of the Constant Q Transform defined by the formula 17 is very expensive and time-consuming. Judith Brown and Miller Puckette devised an efficient algorithm of transforming a DFT into a Constant Q Transform using the sparse matrix multiplication in the spectral domain in 1992[12] .

According to the B. Blankertz [11], the constant Q transform formula 17 is transformed to the matrix multiplication row vector x as following,

$$x^{cq} = x \cdot T^* \tag{18}$$

where $(T \in N \cdot K)$ and T^* the complex conjugate of the temporal kernel $T = (T_{nk} \;\; n < N, k < K)$

$$T_{nk} := \begin{cases} \frac{1}{N_k} w N_k[n] e^{2\pi Qin/N_k} & if\ n < N_k \\ 0 & otherwise \end{cases} \tag{19}$$

eliminating small enough components of $S := \mathcal{F}[T]$ yields a sparse matrix.

From Parseval's equation,

$$\sum_{n=0}^{N-1} x[n]y[n]^* = \frac{1}{N} \sum_{n=0}^{N-1} x^{ft}[n]y^{ft}[n]^* \tag{20}$$

where x^{ft} and y^{ft} are the discrete Fourier transform $x[n]$ and $y[n]$, and $y^{ft}[n]^*$ is the complex conjugate of $y^{ft}[n]$[11], the Constant Q Transform formula 17 can be written as following,

$$x^{cq}[k] = \frac{1}{N} \sum_{n<N} x^{ft}[n] S^*_{nk} \tag{21}$$

Now rewrite the formula 21 in matrix notation

$$x^{cq} = \frac{1}{N} x^{ft} \cdot S^* \tag{22}$$

As derived above, S is sparse matrix and therefore, $x^{ft} \cdot S^*$ eliminates much calculation from $x \cdot T^*$ in the formula 18.

In the log frequency spectrum, the frequency resolution changes towards higher frequencies which realize the constant pitch resolution. The remarkable difference between the linear frequency spectrum and the log frequency spectrum is the distance between fundamental frequency and the n number of harmonic frequencies. The nth harmonic of the linear frequency spectrum located integral number multiples of the fundamental frequency and thus the distance changes exponentially. In contrast, the nth harmonic of the log frequency spectrum located $log2$, $log3$ $logn$ away from the fundamental frequency and the distance changes linearly. The figure 28 and 29 illustrates the log frequency spectrum produced by the Constant Q Transform. The Δa, Δb, Δc represents the distance between neighboring harmonics in the piano sound in C3 and similarly, Δa‘, Δb‘, Δc‘ in the piano sound in C4. As seen in the figure 28 and 29, the piano sounds in C3 and C4 have similar harmonic structures and therefore, the corresponding harmonic distances in different pitches are $\Delta a \simeq \Delta a$‘, $\Delta b \simeq \Delta b$‘, $\Delta c \simeq \Delta c$‘. It indicates that the harmonic structure is changed as the parallel shift depending on the fundamental frequency.

Log Frequency Spectrum

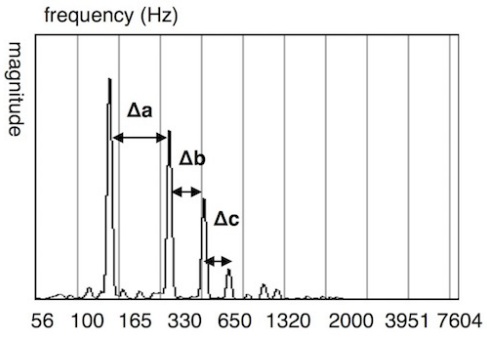


Figure 28: Spectrum of Piano sound in C3

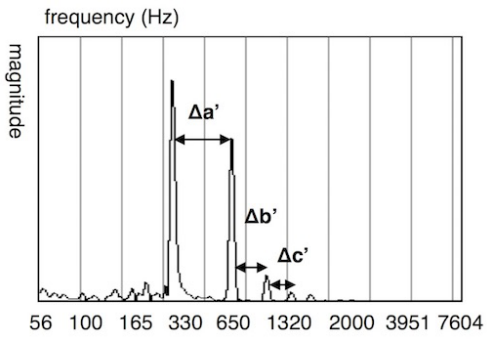


Figure 29: Spectrum of Piano sound in C4

We have done our own implementation of this efficient algorithm in C in order to produce the log-frequency spectrum and this is used for the specmurt analysis discussed in the next section.

The inverse Constant Q Transform is also studied by D. Fitzgerald et al., however, the resultant signals are not as well practical as the one produced by the Inverse Fourier Transform. He concluded that the inverse Constant Q Transform with his algorithms are only valid for signals containing only pitched instruments, and the signals containing broadband noise such as drum sounds will not be inverted correctly[13]. In our study, we have not implemented the inverse Constant Q transform because we target various kinds of sounds including percussive sounds in our analysis process.

5.11 Specmurt Analysis

The Specmurt, which was first introduced by Sagayama et. al. in 2003[20], is aimed at estimating the fundamental frequency(f_0) distribution of polyphonic music signals under the assumption that all tones in the polyphonic music have relatively common harmonic structure patterns. The name Specmurt was derived from the name of Cepstrum. While the Cepstrum is the inverse Fourier transform of a log-scaled power spectrum with the linear frequency, the specmurt is the inverse Fourier transform of a linear power spectrum with log-scaled frequency [20].

As discussed in the section 5.10 Log Frequency Power Spectrum, the relative location of f_0 and harmonic frequencies are constant no matter how f_0 changes and the harmonics are shifted in parallel depending on the modification. Since musical instruments have regular sound quality in the same register, they have similar harmonic structure pattern

as seen in the piano sound examples in the figure 28 and 29. Under proper conditions, an instrumental sound with different pitches can be represented by the convolution of fundamental frequencies and a common harmonic structure pattern. Conversely, the f_0 distribution is calculated by the deconvolution of spectrum and common harmonic structure pattern.

The key to the successful analysis is the definition of a common harmonic pattern which influences the resultant pitch distribution significantly. However, all constituent sounds of a polyphonic sound do not necessarily have a common harmonic structure in real polyphonic music, and furthermore, the structures might be varying over time. To address this issue, Sagayama suggests an optimization algorithm in which calculating the best compromise common harmonic pattern to minimize the amplitudes of subharmonics after deconvolution in the specmurt domain [20]. Hence the specmurt is usually operated with an optimization algorithm, but we found that the employment of the erroneous harmonic/overtone pattern has a significant potential to decompose a targeted spectrum into the particular harmonic or overtone structures. We exploit this possibility to develop an algorithm which decomposes a corpus of audio data into specific sound characteristics. Here, we only introduce the basic algorithm of the specmurt analysis and discuss the further applications in the section 11 Decomposition.

According to Sagayama, the specmurt is defined as following. Firstly, a harmonic structure pattern is represented by $h(x)$, where x is a function of log-frequency and $x = 0$ represents a fundamental frequency, $x = 1, 2...n$ represents 1st, 2ed nth harmonics respectively, and $h(0) = 1$. The log frequency power spectrum $u(x)$ consisting of multi pitch signal is represented by the convolution of $h(x)$ and fundamental pitch components $v(x)$

$$v(x) = h(x) * u(x) \tag{23}$$

Since we already have a log frequency power spectrum $tildeu(x)$, the fundamental pitch components $v(x)$ are calculated by the deconvolution of harmonic structure pattern $h(x)$ and $tildeu(x)$,

$$u(x) = h(x)^{-1} * \tilde{v}(x) \tag{24}$$

The convolution becomes multiplication in the frequency domain by Fourier trans-

form, the formula can be written with $U(y)$, $H(y)$, $\tilde{V}(y)$ that are the inverse Fourier-transformed function of $u(x)$, $h(x)$ and $v(x)$ as following,

$$U(y) = \frac{\tilde{V}(y)}{H(y)} \tag{25}$$

where y denotes log-Fourier transformed log-frequency. Now $u(x)$ can be obtained by the Fourier transform of $U(y)$

$$u(x) = \mathcal{F}[U(y)] \tag{26}$$

The figure 30, 31 and 32 illustrate a targeted log frequency spectrum representing piano C major harmony (C3-E3-G3), harmonic pattern and the resultant specmurt respectively. We assume that the harmonic peaks of the piano sound are located integral number multiples $(1, 2, 3, ..., n)$, and the magnitudes are in inverse proportion $(1, 1/2, 1/3, ..., 1/n)$ as seen in the figure 31. With the harmonic pattern, the specmurt declines the target harmonic components and retains firm peaks representing the fundamental pitches C3, E3 and G3. The result of the specmurt analysis is significantly affected by the initially defined harmonic pattern, and therefore, when the pattern is not the adequate to the original harmonic structure of the target, the specmurt has significant irregular peaks which are regarded as noise. The weak peaks seen in the entire specmurt arise from this problem. Real-world sounds including musical instruments consist of many irregular qualities. Since the harmonic structure is slightly different depending on the register, it is impossible to estimate an exclusive harmonic pattern satisfying all constituting notes. To address this problem, we use a noise reduction and optimization processes in order to produce a better result which is discussed in the section of Audio Decomposition.

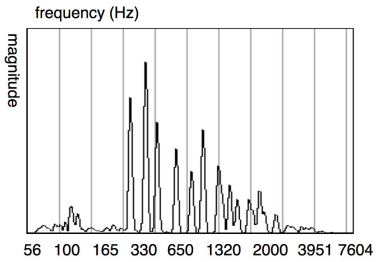


Figure 30: Target Spectrum

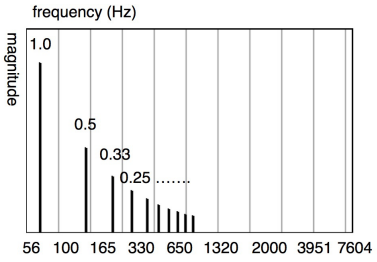


Figure 31: Common Harmonic pattern

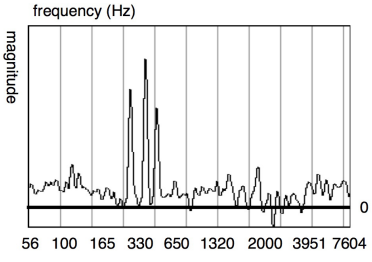


Figure 32: Result

6 Feature Extraction

The features or descriptors are a set of values which describes various qualities of a unit of a signal such as loudness, pitch, and noisiness. Some high dimensional coefficients such as MFCCs represent the shape of a spectrum which is capable of describing various aspects of timbre. Various features are calculated by means of spectral analysis in which a time-varying signal is transformed to a static frequency domain. Therefore, the features are calculated per each FFT frame and the time variant characteristics are averaged over a frame. We exploit these features to identify and label characteristics of signals. The time variant information is interpolated by the delta features between adjacent frames. A sequence of features is a sort of very rough discrete temporal representation. Besides employing an overlap method for the temporal representation, we also exploit the delta and delta-delta features between neighboring frames or differentiate coefficient in a sequence of frames which represent the trajectories of the features over time. Here, diverse methods of feature extractions and characteristics representations will be introduced.

6.1 Linear Power Spectrum

The Linear Power Spectrum is the most basic feature which is employed to calculate various advanced features. The formula 5 yields complex value r and i. The following formula yields power spectrum μ for each mth frame where k denotes FFT bin index.

$$\mu_k = \sqrt{r_k^2 + i_k^2} \quad (0 \leq k < N/2) \tag{1}$$

6.2 Spectral Magnitude

With the formula 1, the magnitude of one FFT frame can be written as following.

$$M_m = \sum_{k=0}^{N/2} \mu_k \tag{2}$$

From our experiments, the descriptors showing a sum of linear power spectrum such as the spectral magnitude is relatively less influenced by the frequency localization problem. It is because of that, that descriptor does not describe the property of the frequency information but the magnitude of the entire spectrum.

6.3 Spectral Centroid

The Spectral Centroid represents the static spectral distributions and has a connection to the brightness of the signal, which is obtained by calculating a center of spectral gravity. John M. Grey[24] concluded that the centroid is a better numeric representation for most static spectral distributions than the median of a spectrum. If the value of the centroid indicates high frequency, it means the sound data consisting of mainly higher frequencies. The sound character tends to become more metallic when it has more high-frequency components. The frequency of each FFT bin is represented as F_n, then the centroid of the mth frame is represented as C_m, and then the equation of centroid is written as following,

$$C_m = \frac{\sum_{k=0}^{N/2} (F_k \cdot \mu_k)}{M_m} \tag{3}$$

However, in the particular cases, the centroid is not able to describe an accurate sound brightness measure by itself. The issue and a solution are discussed in the next section.

6.4 Spectral Spread

It is not possible to get an accurate sound brightness measure by using only the spectral centroid. It is because the centroid represents only the center of spectral gravity but not the spectral distribution. For instance, figures 1 and 2 show spectrums of different sound data. Figure 1 illustrates a big energy on the center, and other places have small energy. On the other hand, 2 illustrates big energies on both sides, and the center has

small energy. Although both sound data have totally different characters, the position of the spectral gravity centers is the same. Therefore, it is not possible to obtain the appropriate measure of brightness by using only the centroid.

The spectral spread represents the distribution of the FFT bins in the spectrum. It is calculated by multiplying the deviations between the magnitude of each FFT bin, which value needs to be more than 0, and the spectral centroid previously obtained with the formula 3. Thus, FFT bins having large magnitude and being far from the centroid add big values to the sum, and vice versa. For example, the value of the spectral spread for the figure 1 is much smaller than the figure 2. This means the combination of the spectral centroid and the spectral spread represent more accurate spectral distribution in various cases.

The spectral spread S is described as follows.

Two cases for the spectal centroid.

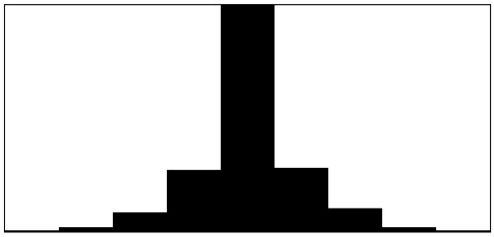


Figure 1:

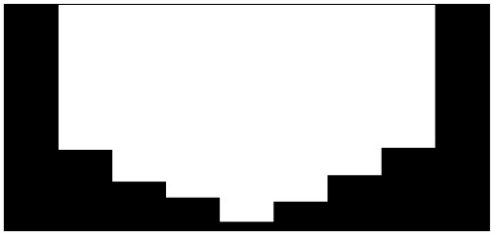


Figure 2:

In the following formula 4, the spectral spread is described with S

$$S_m = \frac{\sum_{k=0}^{N/2}(|F_k - C_m| \cdot \mu_k)}{M_m} \tag{4}$$

where F_k denotes the frequency of each FFT bin calculated by the following formula.

$$F_k = \frac{SamplingRate}{N}k \tag{5}$$

6.5 Spectral Flatness

The spectral flatness is a criterion of the flatness of a spectrum. It represents a sort of noisiness. For a given spectral magnitude, if the magnitudes of all FFT bins in a frame are the same, the spectral flatness of the frame has a value of 1. For the same

given spectral magnitude, if there are differences in magnitude among FFT bins, then the value of spectral flatness gets smaller as those differences get bigger. The value of the spectral flatness is smallest for a pure tone, such as sine wave, and largest for white noise. The first step to obtaining the spectral flatness, shown in formula 6, provides a product of each FFT bin magnitude. The computing of square roots is operated as the same number of times N which denotes FFT window size so that the result always stays the same value when the magnitude of each FFT bin is the same, and the result either increases or decreases when the magnitude of each FFT bin is different.

$$P_m = \sqrt[N/2]{\prod_{k=0}^{N/2}(\mu_k)} \tag{6}$$

6.6 Fundamental Pitch

In this section, we discuss the fundamental pitch f_0 extraction, also referred as pitch estimation/detection algorithms for monophonic music signal. The multi pitch estimation for polyphonic music signal is discussed in the section IV. Simply saying, the fundamental pitch corresponds to the periodicity or repetition rate of a waveform. In the spectrum, the fundamental pitch indicates the lowest frequency of the constituent of a periodic waveform. While young people, pure tones with frequencies between 20Hz and 20kHz are audible, only sounds with repetition rates between about 30Hz and 5kHz elicit a pitch percept that can be called musical and is strong enough to carry melody as the ranges of most of the standard orchestral instruments are between 27.5Hz and 4186Hz [28].

6.6.1 Perception of Pitch

The above discussion may mislead that the fundamental pitch is an extracted peak which is the lowest and, in many cases, the loudest in a spectrum. However, we have complicated phenomena in our auditory system, and hence, we need to consider a proper fundamental pitch estimation method which follows human perception. When we deal with a fundamental pitch, there are two aspects to be considered; the notion of frequency and the perception of pitch[25]. The former is perceived in the signal which is the isolated sinusoid without overtones or with very few overtones. The latter is in the signal from Real-world sounds such as music sounds consisting of harmonic

structures or rich overtones. In the spectral analysis, it is easier to estimate the accurate fundamental pitch from sin-like waveforms or synthesized waves consisting of isolated sinusoid than the sounds which have a rich spectrum. The rich spectrum, however, reinforces the sensation of a pitch more than the sinusoid waves. The sensation is related to the partials of the sound, and the more harmonically related the partials of a tone are, the more distinct the perception of pitch[25].

The human auditory system must then be considered. One of the most interesting controversies is about how the resolved and unresolved harmonics influence the perception of pitch. The resolved harmonics indicates the low-frequency harmonics which are exclusively represented by the single auditory filter and that means, the lower harmonics are separated out in the cochlea. The resolved harmonics has strong excitation pattern and those of which are clearly observed in the first few harmonics. The unresolved harmonics indicates the high-frequency harmonics which elicit complex waveforms that reflect the interaction between other harmonics. The unresolved harmonics are not separated out in the cochlea. The threshold between resolved and unresolved harmonics is not clear and depends on many factors such as the frequency of the fundamental pitch[27][29].

For the resolved harmonics, no single auditory filter has unambiguous information about the fundamental frequency of the signal[25], and a few resolved harmonics may be heard as separated pitches especially by the trained people such as musicians. Most studies suggest that this phenomenon is possible for harmonics up to number five to eight and recent research suggests ten harmonics[30]. On the other side, unresolved harmonics are influenced more easily by phase distortions produced by room acoustics, and they can change the waveform of complex tones. In fact, many experiments have shown that the low-numbered resolved harmonics elicit a much more salient, robust, and accurate pitch than do high-numbered unresolved harmonics[26]. The Moore et al also found that the ability to hear a harmonic out from a complex, or simply detect that it was mistuned, was better for resolved than for unresolved harmonics, although the absolute frequency of harmonic was also found to be important [29]. While the resolved harmonics elicit the robust perception of fundamental pitch candidates, the unresolved harmonics enable to estimate an exclusive fundamental pitch in a complex tone.

6.6.2 Pitch estimation algorithms

There are some pitch detection algorithms such as zero-crossing rate, Cepstrum, and Autocorrelation. The zero-crossing rate is one of the most simple algorithms which estimates the fundamental pitch by the number of times the signal crosses the zero per unit time. The algorithm gives good results for sinusoid waveforms but provides easily erroneous results when the improper zero-crossing caused by modulation effects by high-frequency components. Therefore, the zero-crossing algorithm needs to operate with an initial low-pass filter in order to remove high-frequency components. The filter cut-off frequency should be carefully chosen depending on the quality of the targeted sound since high-frequency components affect our pitch sensation as discussed in the previous section. The zero-crossing rate algorithm is susceptible to noise and fluctuations in instantaneous frequency.

In our case, the targeted sounds to be analyzed are diverse. The sounds may be music sounds, speech sounds, environment sounds or oscillators and they may have monophonic or polyphonic structures. It is also important to detect whether the signal has periodic wave structure or not. For instance, some idiophonic sounds or pink noise are not meaningfuly described a pitch but rather suitable for spectral centroid discussed in the section 6.3. Therefore, we focus on searching the periodicity of a spectrum itself which is formulated by the harmonic or overtone structures. When the targeted sounds have sufficient sensation of a pitch, it should have harmonic or overtone series which are integral number multiples of the fundamental pitch. Otherwise, with a small series, we may hear it as multiple pitches as discussed in the section 6.6.1, and in contrast, with a large series, we may hear it as very complex polyphonic sound, percussive sound, or noise. We employed Cepstrum(section 5.9) and Autocorrelation algorithms. When the target consists of multiple complex tones and is considered as a polyphonic sound, we decompose it into multiple voices as discussed in the section IV, Decomposition.

6.6.3 Pitch Estimation by Autocorrelation

The Autocorrelation is a mathematical representation of the similarity between a time series data and a delayed version of itself over time. The correlation is the result of a measure of similarity between two signals calculated by a function of a time-lag between the beginnings of the two-time series. The Autocorrelation is the correlation

between the same time series, but one is shifted over successive time intervals. The Autocorrelation is also known as the sliding inner product[6]. When the time lag is zero, the result of the Autocorrelation gives significant high peak which shows the exact similarity and then the similarity decreases as the time lag increases. When the signal has a regular periodic wave over the entire series, then the Autocorrelation gives some peaks at the particular time lag index. The highest peak, except around zero indices, is expected to indicate the fundamental pitch.

The advantage of this algorithm is its better performance in more general sound qualities and its immunity to noise[32][34] compared with the Cepstrum. Cheveigne and Kawahara developed the YIN system based on the Autocorrelation to prevent errors by using several features[33].

For signal $f(\omega)$, Autocorrelation function $r_t(\tau)$ is defined as following formula.

$$r_t(\tau) = \int_{-\infty}^{\infty} f(\omega)\overline{f}(\omega - \tau)d\omega \quad (-\infty < \tau < \infty) \tag{7}$$

Where τ is a lag from the starting time t. Consequently, the above formula is represented in the discrete domain as following.

Autocorrelation function (ACF)

For signal x, the first is infinitive discrete function and the second is finite discrete function.

$$r_t(\tau) = \sum_{j=-\infty}^{\infty} x_j x_{j+\tau} \tag{8}$$

$$r_t(\tau) = \sum_{j=t}^{t+W-1} x_j x_{j+\tau} \tag{9}$$

Where τ is a lag from the starting time t, where W is the initial window size. Alan de Chaveigne and Hideki Kawahara found the advanced formula, where the correlation is calculated by square distance instead of convolution[33], which is named YIN as follows,

Square difference function (SDF)

$$d_t(\tau) = \sum_{j=t}^{t+W-1} (x_j - x_{j+\tau})^2 \tag{10}$$

Norbert Wiener and Aleksandr Khinchin proved and formulated the Wiener-Khinchin theorem where the Autocorrelation is calculated by the inverse Fourier transform of the power spectrum. This formula realizes an efficient calculation of the Autocorrelation.

Wiener-Khinchin theorem

$$\int_{-\infty}^{\infty} e^{j\tau\omega} f(\omega) d\omega \quad (-\infty < \tau < \infty) \tag{11}$$

and then, the above formula is represented in the discrete domain as follows.

$$r_t = i\mathcal{F}[\mathcal{F}[m_i]], \quad (0 \leq i < I) \tag{12}$$

Where $\mathcal{F}[]$ is the forward DFT function and $i\mathcal{F}[]$ is the inverse DFT function. The result of the Autocorrelation shows multiple peaks, and the first peak represents the similarity of time-lag zero which is the magnitude of the entire data series. The fundamental peak is found on the coefficient index of the highest peak f except the time-lag zero. The index number is translated to Hz by the following formula.

$$note = Samplingrate/f \tag{13}$$

6.6.4 Pitch Estimation by Cepstrum

The Cepstrum is another option to estimate the fundamental pitch. The Cepstrum has an advantage over the Autocorrelation for speech analysis[32]. The Autocorrelation function convolves the vocal source and tract and produces multiple broad peaks standing over several indices. Due to the formants components, the Autocorrelation provides unnecessary peaks in the result. In contrast, the Cepstrum separates the formants and frequency components by filtering effect, and thus, the obtained peaks are spiky and not modulated by the vocal sources. It is easier to extract a particular index of the peak in the Cepstrum domain which realizes accurate fundamental pitch estimation[32][34], and this effect appear remarkably in speech signals.

Figures 3 and 4 illustrate the comparison between the Autocorrelation and Cepstrum. The arrows indicate the peak which corresponds to the fundamental pitch. We can observe rougher fluctuations in the entire figure of the Autocorrelation and flatter peaks in the Cepstrum. In the enlarged view, figure 5 and 6, the Cepstrum has more spiky peak than the Autocorrelation. The narrower peak can point more accurate and undeniably frequency which possibly indicates a more precise fundamental pitch.

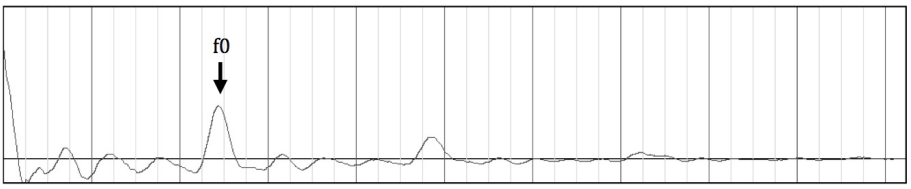


Figure 3: Autocorrelation

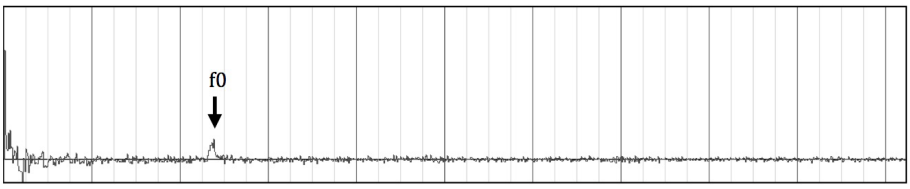


Figure 4: Cepstrum

Enlarged view of the figure 3 and the figure 4. The arrows show the range of peaks.

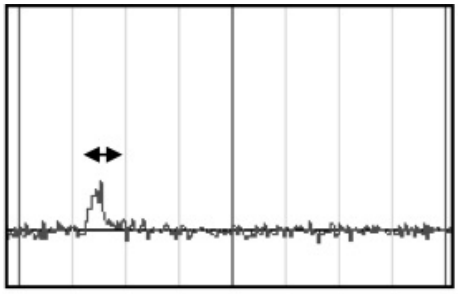


Figure 5: Cepstrum Function

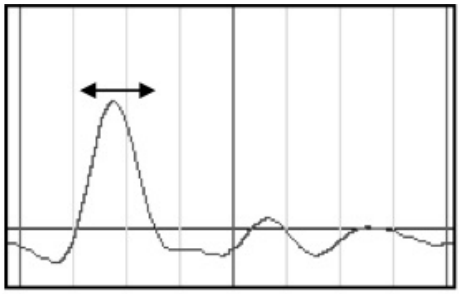


Figure 6: Autocorrelation Function

6.7 Multi-resolution Pitch estimation

Both the Cepstrum and Autocorrelation have different strengths and weaknesses depending on the quality of the targeted sound. However, they have the same issue in the simultaneous representation of the frequency and time localizations. The Cepstrum is calculated in a unit of the FFT frame size, and the Autocorrelation is done in a window size or within the Wiener-Khinchin theorem, in the FFT frame size. With a short unit size, they yield a lower frequency resolution, and the accuracy of the pitch estimation decreases particularly in the low-frequency band. While a bigger frame size improves the frequency resolution, it reduces the quality of the time localization significantly. To address this problem, we employed the multi-resolution FFT(in the section5.8) to complement the qualities of both the frequency and time localization. The figures 7 to

11 are the waveform of an octave progression from A-1(ca.55Hz) to A5(3,520Hz) played by the piano tuned in an equal temperament (A=440Hz). The black lines illustrate the sequence of the estimated fundamental pitches calculated by the Autocorrelation with different FFT frame sizes from 1024 to 16384samples. The shorter FFT frame sizes yield more incorrect result particularly in the lower frequency bands. The worst error is observed in the FFT frame size of 1024 samples shown in the figure 7. In contrast, the longer FFT frame sizes yield better results, but they involve worse time localizations.

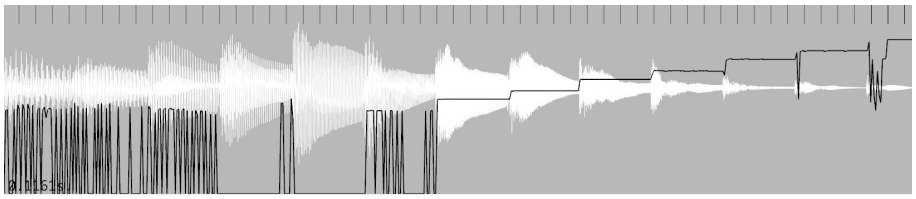


Figure 7: Autocorrelation FFT size = 1024

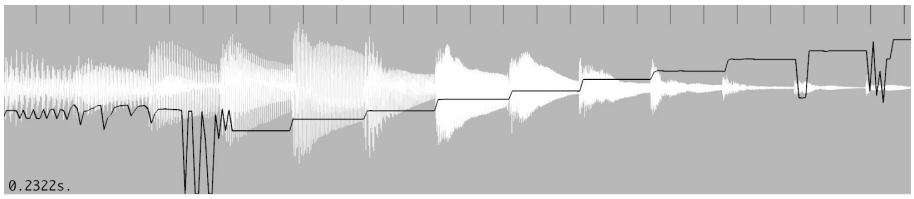


Figure 8: Autocorrelation FFT size = 2048

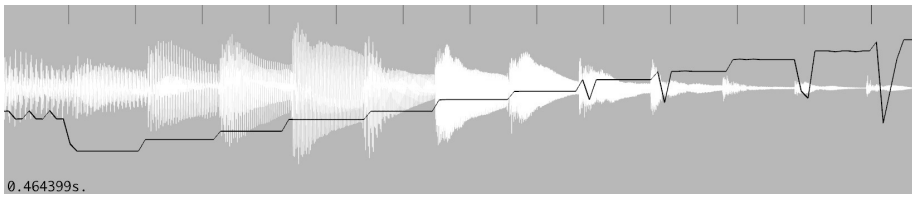


Figure 9: Autocorrelation FFT size = 4096

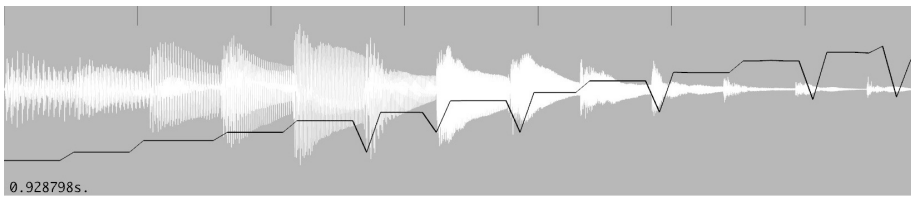


Figure 10: Autocorrelation FFT size = 8192

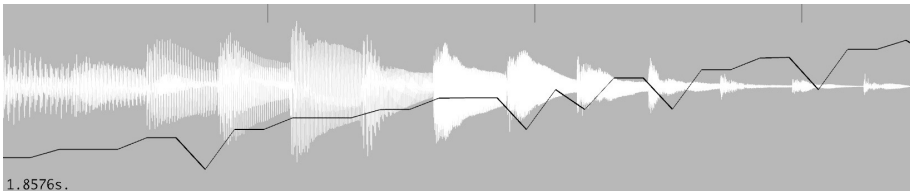


Figure 11: Autocorrelation FFT size = 16384

We developed our pitch estimation system under the assumption that the longer frame size can estimate a fundamental pitch more accurately and it can be employed as a guideline to select the best suitable shorter frame sizes. We made a rule which constrains the selection of a frame size in a particular frequency ranges in order to realize an efficient pitch estimation for the given signal as shown in the table 6.7. At first, the system estimates a rough frequency with the longest frame size. Subsequently, it evaluates the result whether the size is adequate or not. If the frequency is beyond the threshold, the estimation is operated again with one smaller frame size in order to gain the resolution in the time domain as shown in the figure 12. The process is repeated until the best efficient frame size is selected. This system is also able to detect the sound events occurring for a short moment. At last, the system produces a sequence of fundamental pitches estimated by different frame sizes. Figure 12 shows the process of our pitch estimation system and figure 13 illustrates the result. We can observe better fundamental pitch representation in both frequency and time localizations. Some errors occur nearby pitch changing points due to the hammer noise.

Table 1: Relationship between FFT frame size and estimated frequency

FFT frame size	Threshold frequency (Hz)	MIDI
16384	$0 \sim 30.87$	$0 \sim 23$
8192	$30.87 \sim 41.20$	$23 \sim 28$
4096	$41.20 \sim 61.74$	$28 \sim 35$
2048	$61.74 \sim 110.0$	$35 \sim 45$
1024	$110.0 \sim$	$45 \sim$

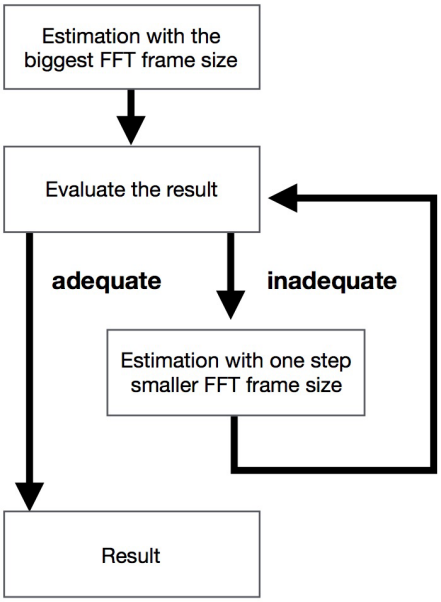


Figure 12: Multi-Resolution Pitch Estimation System

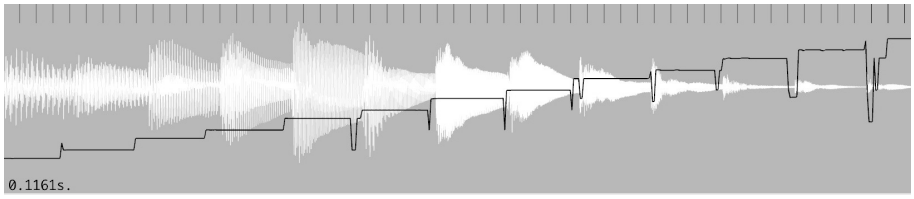


Figure 13: Pitch estimation by the multi-resolution system

6.8 Mel-Frequency Cepstral coefficients

The Mel-frequency Cepstral Coefficients (MFCCs) are rough representation of the shape of the spectral envelope, and they capture timbral properties of the sound. The primal idea of the MFCCs is introduced by Bridle and Brown in 1974 and is developed by Mermelstein in 1976[35]. The rough spectral envelope suggests the primal characteristics of the timbre and which is represented by a small number of low quefrency components in the cepstral domain. A small number of coefficients is easily manageable which is able to reduce computations significantly. It is one of the most commonly used features in the area of speech analysis, automatic speech recognition, and speech synthesis[36], and is also applied for musical instrument identification systems[17].

The calculation process of the MFCCs consists of several phases. The first step is to transform the given audio signal to the frequency domain using STFT(formula 5.

Secondly, the spectrum is transformed to the Mel-frequency spectrum by means of N number of triangular band-pass filters whose center frequencies are calculated by Mel-scale as following formula.

$$Mel(f) = 2595log_{10}(1 + \frac{f}{700}) \tag{14}$$

Where f denotes the linear frequency domain. The generated filter banks with the channel number = 27 are illustrated in the figure 14.

The spectrum obtained in the first step is multiplied by the Mel-filter bank and summed the values at each channel which yields the Mel frequency coefficients. The Mel-frequency spectrum $Xmel_i(n)$ where i denotes frame index, n is sample index, $h(n)$ is window (i.e. hanning window in our case), and $x_i(n)$ is the signal, is calculated by the following formula.

$$Xmel_i(n) = \sum_{n=1}^{N} x_i(n)h(n)e^{-j2\pi kn/N} \tag{15}$$

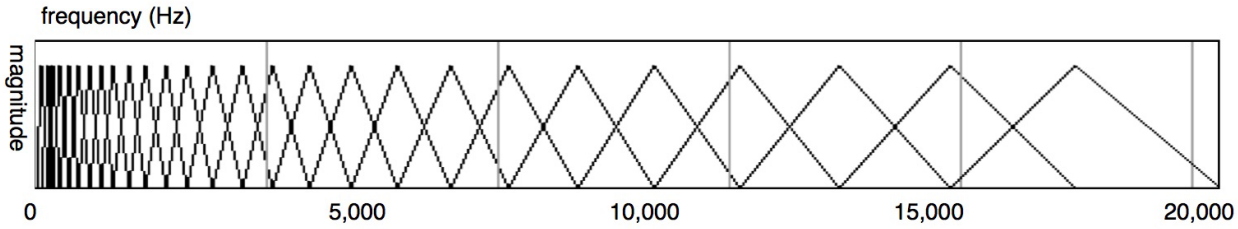


Figure 14: Mel-frequency filter bank

Finally, the Mel frequency coefficients are scaled to logarithmic power, and then, the log-power Mel frequency coefficients are transposed to the cepstrum domain by means of cosine transform (see section 5.3) which yields the Mel-frequency cepstrum coefficients(MFCCs) as follows.

$$MFCCs_i = \sqrt{\frac{2}{L}} \sum_{l=1}^{L} logm(l)cos\left[\left(l - \frac{1}{2}\right)\frac{i\pi}{L}\right] \quad (L = 27) \tag{16}$$

Where L denotes the number of a Mel-filter bank.

The above formula produces L number of MFCCs representing the spectrum envelope. We extract the first twelve MFCCs which are the most determinant coefficients

of the envelope. A larger number of MFCCs represents a more detailed envelope, and less represent a rougher one. The mechanism of this algorithm has been already described in the section 5.9.1 the Liftering. In any cases, it is important to be aware that the lower MFCCs have more significant information of the timbral characters than the higher ones.

In order to reveal sufficiently the spectral envelope, the high-frequency components of the original spectrum are slightly emphasized in order to regulate the spectrum envelope and compensate for the fast decay of sound components. The filter is defined as follows.

$$x'(i) = x(i) - p \cdot x(i-1) \quad (p = 0.97)(0 \leq i < N) \tag{17}$$

The figure 15 illustrates the whole process of the MFCCs calculation.

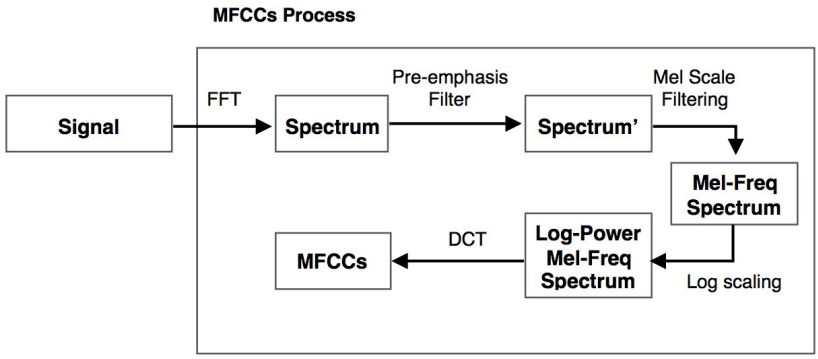


Figure 15: MFCCs process

6.9 Deltas and Delta-Deltas

The delta-feature or differential feature represents the range of how the features change between adjacent frames. While the spectral analysis reveals a snapshot of the signal varying in time, the delta-features can interpolate the discrete features over time. Let a sequence of optimal features be $a(n)$, the delta feature is calculated as follows.

$$\Delta a(n) = |a(n) - a(n+1)| \quad (0 \leq n < N) \tag{18}$$

Delta of the delta features or acceleration features are calculated in the same way but using $\Delta a(n)$ instead of $a(n)$.

6.9.1 Delta MFCCs

The transitional properties of the MFCCs are calculated by summing all delta coefficients as follows,

$$\Delta MFCCs(n) = \frac{\sum_{l=1}^{L} |a_l(n) - a_l(n-1)|}{L} \quad (0 < n < N) \tag{19}$$

and the delta-delta MFCCs is also calculated by the same way using $\Delta MFCCs(n)$.

6.10 Discussion

We have just introduced various statistical spectral analysis methods. The section 5, which is the first half of this part, was concentrated on describing the DFT functions and the spectrum representations. These algorithms are the fundamentals of our advanced audio analysis programs which will be discussed in the further sections. We have discussed the representation method of both frequency and time localization throughout this section. In the spectral analysis, these two properties contradict each other, and thus, it is significant to consider carefully selecting the best efficient algorithms according to a given sound quality.

In section 6, a more practical audio analysis method, the feature extraction has been discussed with various applications of the DFT functions. Some features are scaled or interpolated by different mathematical models in order to adapt their properties to more familiar formats to the human perception. As the extraction process operates on the given signals in a unit of the FFT frame size, the characteristics of the signals are averaged over a short time. Therefore, they are represented in a discrete manner, that is, the descriptive representation is done for the individually separate and distinct frame. We proposed some algorithms such as overlap method and delta-features in order to interpolate the statistic features between adjacent frames and to represent the time-varying spectral components. Nevertheless, the time-varying representation is still limited, and this is a significant theme for developing advanced algorithms in audio segmentation and decomposition which will be discussed in the next sections.

The extracted features are employed in a concatenation in order to describe the signal transition in time. For instance, the series of MFCCs are assembled in a Markov Chain or Hidden Markov Chain in order to identify words or any meaningful sentences in the speech recognition[35]. As an individual phoneme is separated into a frame

or a few successive frames, its concatenation can represent a segment of speech. For another example, N. Campbell et al. developed a speech re-sequencing synthesis system by concatenating a chunk of short segmented recordings of human speeches in order to produce naturally synthesized voices[4]. In the next section, we will handle the descriptive features as a data series and explore the algorithms which reveal the sound structures.

Part III
Audio Segmentation

7 Introduction

In this section, we will describe segmentation methods for audio signals in time. The STFT itself already operates the segmentation task as it divides a given signal into a short audio unit of FFT frame size. However, this segmentation is done expediently in order to realize efficient computation and the segments are often too short to represent audibly meaningful units. Real-world sounds usually comprise various temporal structures. For instance, musical fragments consist of melodies, gestures, and rhythmical patterns. The environment sounds also comprise time variable components. While a human being has the ability to distinguish easily the individual sound events occurring through time, it is quite difficult to build computational models to realize this process. The goal of segmentation in the audio analysis is to find the boundaries of individual sound events or patterns and to extract the isolated sound sources. A sound event or music event often shares its boundaries with the neighboring events or silence, and thus, the resonance of their sounds is often complicatedly mixed, and are inseparable by the algorithms introduced in this section. This problem overlaps the issue of the audio decomposition which will be discussed later, and here, we concentrate on describing the segmentation of the time sequences, not polyphonic structures.

In our research, sound event extraction is one of the primary subjects. The determination of sound events instead of chunks of arbitrarily segmented units is beneficial to obtain for chamber music composition. The segments divided by the STFT involve abrupt discontinuity because they split on the middle of sound events. In contrast, the longer and isolated sound units have less sense of the discontinuity and it can avoid producing noisy and mosaic-like synthesis results. This process is expected to synthesize audibly natural and practical musical sources and also simplifies the audio representations. Moreover, it is capable to operate analysis, labeling, and manipulations phases more efficiently than processing the original signals. We estimate the boundaries of musical events according to a criterion representing a musical change which is calculated by the descriptive features of the signals. This is called as the novelty-based segmentation. The audio novelty can be measured by various features. For example, magnitude

indicates the boundary of sound changes in dynamics, MFCCs are good indicators for changes in timbre or instrumentation, and chroma-based features such as MIDI pitch are for changes in melody or harmony[6]. In the novelty-based segmentation, as we employ the STFT in the process, extracting boundaries of the events is equal to finding the frames showing significant audio novelty. While we use the same FFT frame size over an operation for reasons of expediency, we adjust the position of boundary using the Overlap method, or multi-resolution FFT analysis.

The FFT frame size significantly affects the quality of the result, and it needs to be defined carefully depending on the property of a targeted sound and the synthesis purpose. For instance, in the speech analysis/synthesis, the FFT frame size of 1024 to 2048 samples corresponding to the length ca. 20 to 30 ms. are used in order to separate individual phonemes.[36]. In music segmentation, such a very short unit length is sufficient for detecting musical elements such as attack or decay from an envelope of the sound magnitude. Longer frame size results in the longer segments with rougher boundaries and may avoid producing short mosaic-like sounds. However, the sensitivity to the novelty is so small that it increases a risk of missing some segmentation points.

Although novelty-based segmentation is a powerful algorithm, it has a problem when multiple segmentation points are extracted for a brief time in a musical event, as is typically observed in a percussive sound. The resultant segments are abruptly separated in the middle of an events, what is heard as unnatural and mosaic-like sounds. To address this problem, we have also implemented a segmentation rule which prevents to separate audio signals into too short units and preserves isolated sounds, gestures, and silence. However, we admit that it is extremely difficult to develop an exclusive algorithm which produces satisfactory results for all kinds of sounds.

Here, we will discuss the variation of the novelty-based segmentation by applying variable feature vectors to calculate the audio novelty. The self-similarity matrix is also a useful algorithm to extract structural properties, and represent them as a novelty function.

8 Delta-features based Segmentation

We first describe the segmentation method using the Delta-features. The Delta-features, which is introduced in the section 6.9, represents a range of change between adjacent

frames. An intense peak of a the Delta-feature is expected to indicate a novelty of the sound. Figure 16 illustrates the waveform of a drum loop sound (`Drum_Loop1.aif`), the magnitude with a black line, and the delta-magnitude with a red line respectively. The FFT analysis is operated in the frame size of 1024 samples, overlap number of 2. The drum loop has clear attacks on all events which are represented by intense peaks of the delta-magnitude. These peaks show the significant dynamics change over time. However, it can be seen that the delta-magnitude feature does not effectively represent the pitch or timbral changes played in slur or legato as seen in figure 17.

Figure 17 illustrates the waveform of a short musical fragment played by a flute (`flute-fragment.aif`). A blue line represents the delta-MFCCs(introduced in the section 6.9.1) and the arrows indicate the significant peaks. The first two arrows indicate the place where the same note is repeated with a soft articulation, and the last two arrows indicate the smooth pitch changes played in legato. While we can observe remarkable peaks in the blue line of the delta-MFCCs, the red line of the delta-magnitude shows much lower peaks so that some of them are easily overwhelmed by surrounding fluctuating noise. Similar things can be happened when analyzing Violin legato sounds, human voices, and glissando like sounds. The delta-MFCCs, in contrast, can represent the timbral changes and thus, they are useful to detect more diverse sound changes. In fact, the MFCCs and the delta-MFCCs features are applied for the speech segmentation in order to separate and identify faint phonemes[36].

In our program, we use vast amounts of audio data to be analyzed in order to construct a database, and it is not realistic to manually assess whether each segmentation task is done properly or not. The only solution we find in this moment is to employ a robust and versatile algorithm not one which is greatly precise but specialized for a particular case. It is also considerable to combine multiple delta-features, such as the delta-centroid and delta-pitch. The delta-centroid is especially useful to measure the timbral differential which can be employed in order to assess the precision of the segmentation result operated with the delta-magnitude.

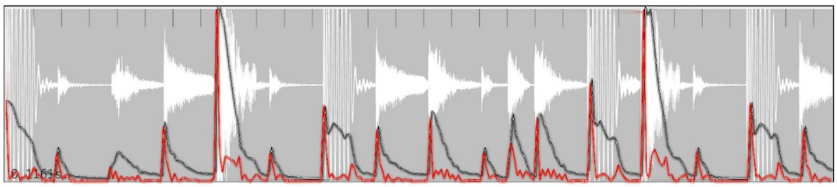


Figure 16: Drum Loop : Magnitude and Delta-Magnitude

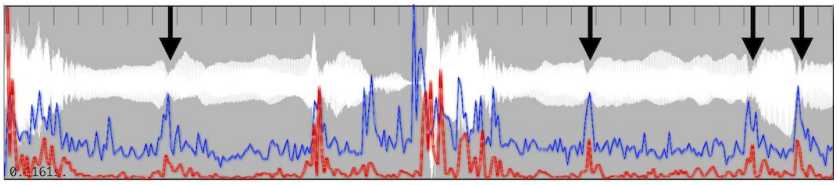


Figure 17: Flute : Delta-Magnitude and Delta-MFCCs

8.1 Segmentation algorithm

Now we describe our segmentation algorithm. We calculated the delta-MFCCs with the FFT frame size of 1024 and overlap number of 2 and then, operate the standardization process. The standardization reduces the variance of analysis data between different sounds and enables to create a coherent model for the segmentation. For instance, drum loop and violin solo yield the sets of data which are significantly different, i.e. different maximum values, medians, and deviations. The standardization can scale a set of values in a particular converged range by using median and deviation. See the further description about the standardization in the section of standardization. Subsequently, we developed the Two Threshold algorithm in order to separate the given audio signals along with a set of the standardized Delta-features.

The Two Threshold algorithm uses two different thresholds; the first threshold is used for extracting a boundary of a segment and the second constrain the frequency of the boundary extractions. A boundary is determined when the Delta-feature goes beyond the first threshold as shown in figure 18. The first threshold in a red line is usually defined larger than the second threshold in a blue line as shown in figure 19. Only the first threshold results in detecting the multiple fluctuating peaks standing close to each other as shown in figure 18. This occurs because the quality of an attack sound often changes dramatically in a short time which appears as the successive intense peaks of the Delta-features. These peaks could be recklessly regarded as boundaries

between new sound events, but actually, the extracted boundaries are interrupting musically meaningful units. The second threshold is therefore employed to prevent the first threshold from taking a new boundary of a sound event in a short time interval until the Delta-features go down below the second threshold. After the condition of the second threshold is satisfied, the first threshold again starts searching for a new boundary. Figure 19 illustrates how the second threshold -blue line- works.

Figure 20 and 21 illustrates the segmentation results operated by the Two Threshold algorithm. The segmentation is done more rigidly with smaller threshold values, and more roughly with larger values. We can observe the different sensitivity of the segmentation around the beginning of sound events. This algorithm is particularly beneficial when analyzing a drum loop which has explosive attack sound as we can see in figure 22.

Comparison between single and double thresholds segmentation algorithms.

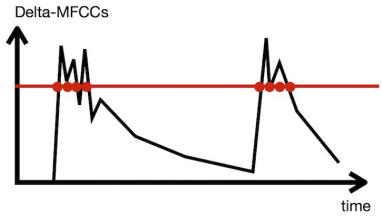


Figure 18: Single Threshold

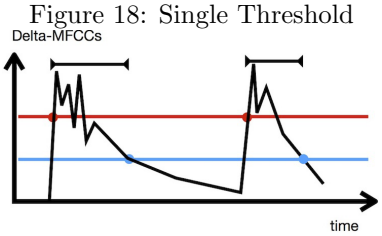


Figure 19: Double Thresholds

The red line represents the first threshold which is larger than the blue line representing the second threshold. The red or blue dots illustrate locations of the segmentation detected by the corresponding thresholds. Inside the horizontal bars indicating the ranges between the detected dots of the first and second thresholds, the algorithm stops taking the next audio novelty and thus no segments are detected.

The red line represents the first threshold which is larger than the blue line representing the second threshold. The red or blue dots illustrate locations of the segmentation detected by the corresponding thresholds.

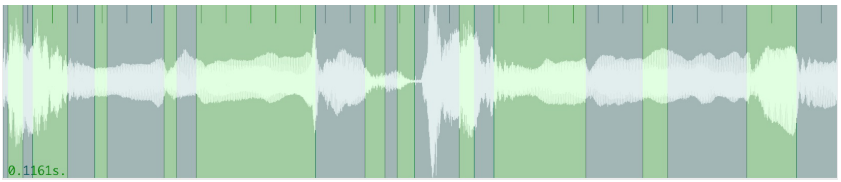


Figure 20: threshold1 = 0.3, threshold2 = 0.01

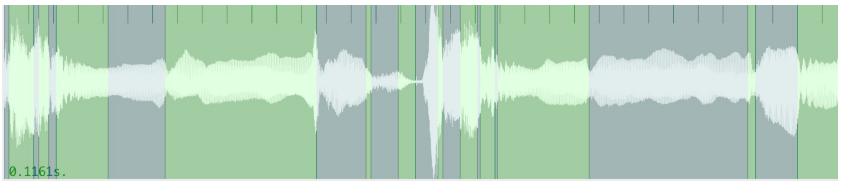


Figure 21: threshold1 = 1.0, threshold2 = 0.8

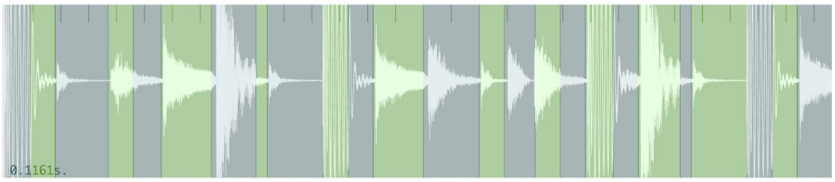


Figure 22: Drum Loop : Segmentation by using the Delta-MFCCs

8.2 Discussion

We have discussed the novelty based segmentation algorithm by using the Delta-features. While the Delta-feature detects the sensitive and detailed audio novelty, they often show the transient novelty which is not related to the sound structure or pattern. Therefore the above-discussed algorithm can be easily influenced by random factors. Since the audio segments are strongly related to the entire sound structure, it is significant to consider the structural elements as well. We will now discuss the Self-Similarity Matrix algorithm to address this issue.

9 Self Similarity Matrix

In many cases, detecting the repetition, homogeneity, and novelty of audio data can provide a clue for tracking musically meaningful structural elements. As discussed in the Autocorrelation section, the correlation of a feature sequence itself yields a result

which reveals its entire structure. Similarly, the self-similarity matrix(SSM) illustrates the self-similarity of a feature sequence in a 2-dimensional matrix by means of comparing each frame of a feature sequence with all other frames of the sequence. Foote Introduced the way to calculate the novelty function from the SSM which automatically locates a point of audio novelty by analyzing local self-similarity[37].

The SSM is capable of revealing a particular aspect of music structures according to the employed feature sequence. For instance, the magnitude feature shows the dynamics pattern, the MFCCs shows the timbral pattern and so on. The resultant 2-dimensional matrix has a lattice-like pattern which separates homogeneous regions of the data series. A boundary of each lattice describes different level of sound novelties depending on the FFT frame size which determines the resolution of the resultant SSM. While the shorter frame size represents individual note, short melody, and timbral patterns, the length of seconds frame size identifies the regional patterns or boundaries between musical segments[6].

The feature vectors, which are extracted in the spectral analysis, are embedded in a 2-dimensional representation by measuring the distance between every pair of frames in the original sequence. According to Foote(2000 [37]), for a sequence $(v_0, v_1, \ldots, v_i)$, the similarity D between feature vectors v_i and v_j, where i and j donates frame indices($i \in \mathcal{R}$ and $j \in \mathcal{R}$), is calculated by the cosine distance as follows.

$$D(i,j) \equiv cos\theta = \frac{v_i \cdot v_j}{\|v_i\|^2 \|v_j\|^2} \tag{20}$$

The result of $D = 0$ means total dissimilarity, and $D = 1$ means complete similarity which always occurs when $(i = j)$. This two-dimensional representation, matrix S is calculated for all frame combinations, where the time frames i and j are corresponding to the coordinate i and j in the matrix. A sequence of the feature vectors is then transformed into a 2-dimensional image, with the pixel size of the width i and the height j respectively as shown in figure 23. A level of the similarity is visualized in the gradation between black and white colors, and darker color represents higher similarity and vice versa. We can obtain different SSM depending on the feature type and which appears as a block-like structure patterns of the (dis)similarity of the feature. Figures 23, 24, and 25 are the various SSM of a Drum Loop sound calculated by the three feature types of magnitude, delta-MFCCs, and Chroma features of MIDI pitch estimated by

the multi-resolution Autocorrelation algorithm. All results involve a black diagonal line which shows the complete similarity on the place of $(i = j)$.

The SSMs calculated by the Magnitude and MFCCs feature vectors reveal the structure of the given signals in which, the boundaries of the block-like structures are corresponding to the segmentation results by means of the previously discussed Delta-MFCCs based novelty algorithm as shown in figure 29. On the other hand, the chroma-based SSM illustrates tremendous amount of small blocks that are rather observed as noise. It is presumed that the Drum loop sound has much more complicated frequency variance over time and has produced such an intricate pattern. We also experienced with the spectral centroid, and the resultant SSM is as well noisy as the chroma-based SSM. In order to test our hypothesis, we subsequently analyzed a flute fragment(`flute_fragment.aif`) which has more harmonic sound characters than the Drum loop. As we can hear, the fragment consists of a short phrase which starts with a fast descending gesture which reaches to a long sustained note. This short phrase is repeated twice in the fragment. This fragment yields unique patterns in the resultant SSMs in various feature vectors and all of them indicate the repeating structure. The Chroma-based SSM illustrates the symmetrical structure as well as the boundaries between the fast and sustained phrases (see the figure 30). The flute sound has harmonic structure, and the fragment consists of the twelve note system in the equal temperament, and therefore, we obtained more meaningful chroma-based SSM than the one generated from the Drum loop.

Self Similarity Matrices from a Drum Loop sound(`Drum_loop1.aif`).

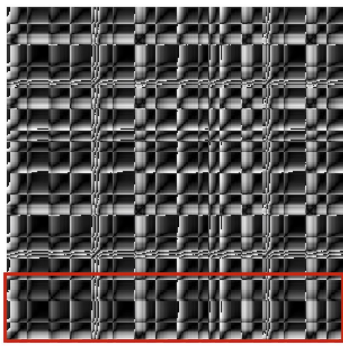


Figure 23: Magnitude

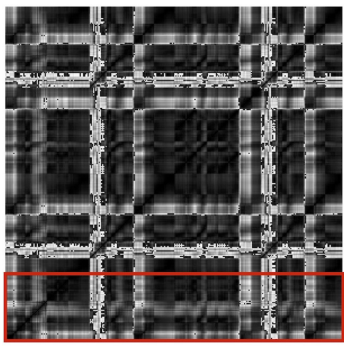


Figure 24: Delta-MFCCs

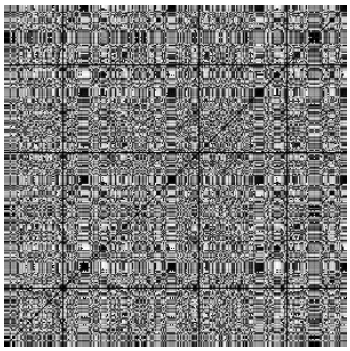


Figure 25: Chroma (MIDI Pitch)

Self Similarity Matrices from a Flute fragment sound(`flute_fragment.aif`).

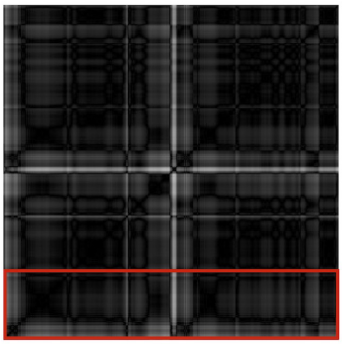


Figure 26: Magnitude

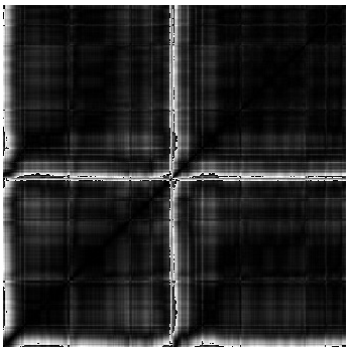


Figure 27: Delta-MFCCs

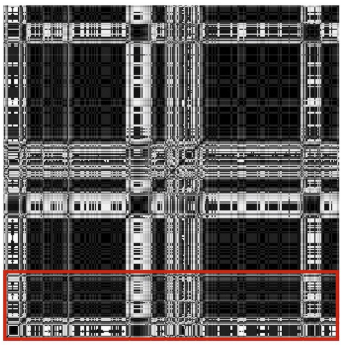


Figure 28: Chroma (MIDI Pitch)

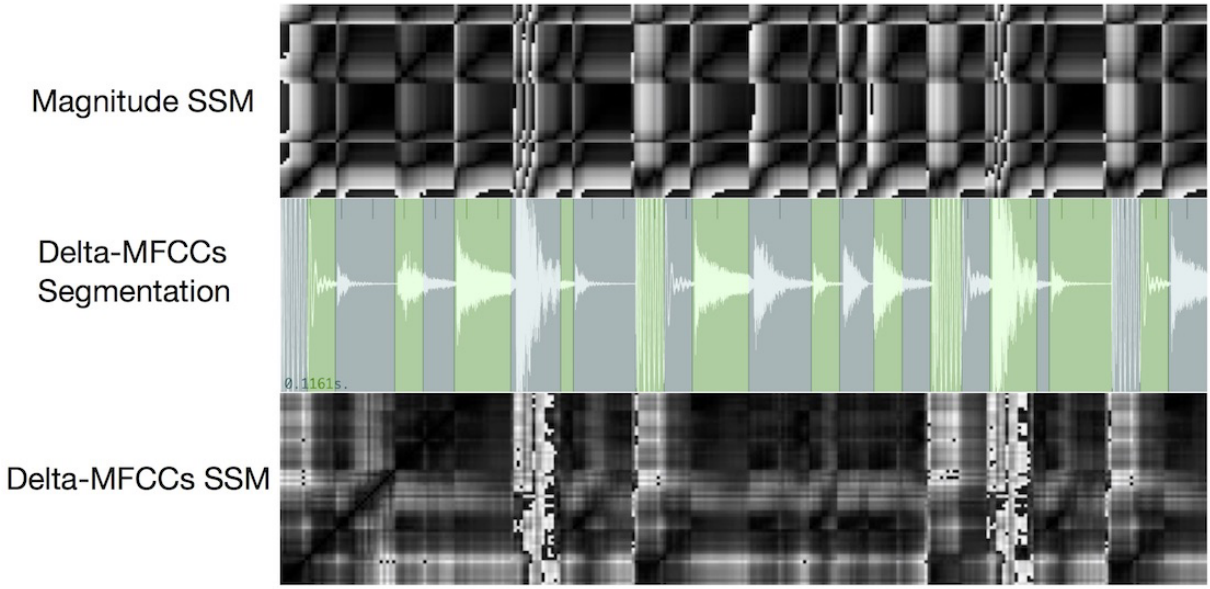


Figure 29: Comparison between the segmentation results by the SSM and the previously discussed Delta-features of Drum loop sound.

The top and bottom figures are the enlarged version of the parts of the Magnitude and Chroma-based SSMs contained by red squares in the figure 23 and 25. The middle figure is the segmentation result by the Delta-MFCCs based algorithm discussed in the previous section.

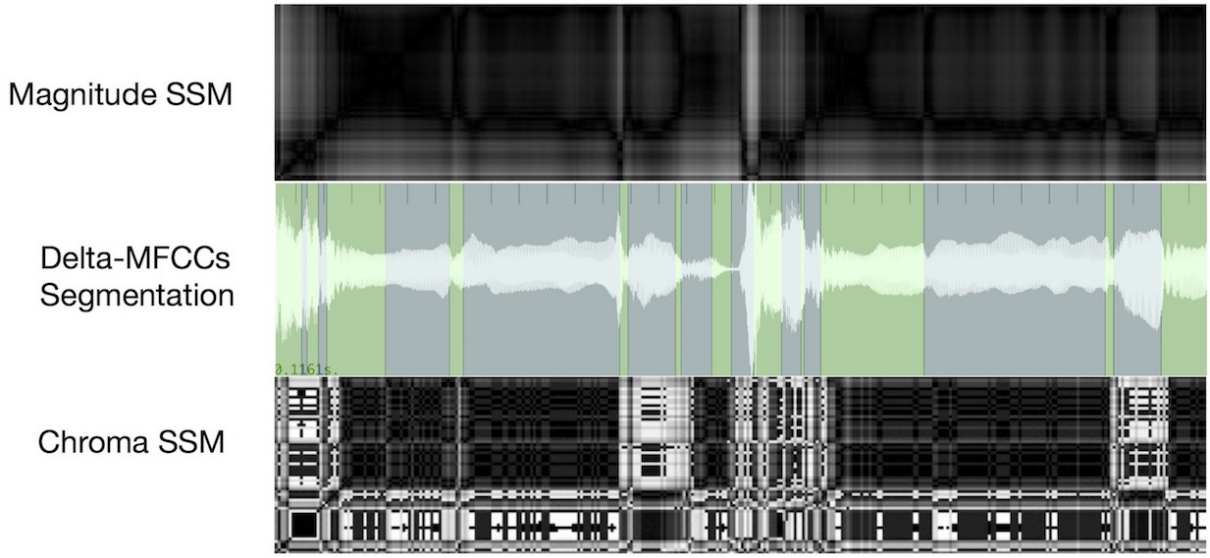


Figure 30: Comparison between the segmentation results by the SSM and the previously discussed Delta-features of Flute fragment sound.

The top and bottom figures are the enlarged version of the parts of the Magnitude and Chroma-based SSMs encompassed by red squares in the figure 26 and 28. The middle figure is the segmentation result by the Delta-MFCCs based algorithm discussed in the previous section. We can observe that the location of extracted boundaries are corresponding each other among three algorithms.

9.1 Novelty Function

As discussed above, the two-dimensional matrix S reveals block-like structures in the case that a sequence of the employed feature contains structural patterns in entire audio signals. However, the obtained matrix consists of two homogeneous structures, which appears in both horizontal and vertical directions and which makes it complicated to extract the Novelty function representing the self-similarity over the matrix. The primal idea in novelty detection is to identify the boundary between two homogeneous but contrasting segments by correlating a checkerboard-like kernel function along the main diagonal of the SSM[6]. The kernel function with the size $L = 2$ which measures the cross-similarity of two regions is defined as follows[38].

$$C = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{21}$$

The first term of the kernel matrix measures the self-similarity and the second term measures the cross similarity between two regions. The inner product of C with the 2x2 area of the matrix S, whose center is shifted along with the diagonal center of

the matrix S where the $(i = j)$, gives a measure of the cross-similarity of the matrix S. A larger size of the kernel matrix measures the novelty over a greater number of frames[38] which is capable of detecting major sections and conversely, a smaller detects more detailed sections. The larger kernels can be constructed by the Kronecker product of C as follows,

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \tag{22}$$

The above shown matrix is for the case of even number $L = 2M$, and M. Müller suggested odd size of the kernel matrix, $L = 2(M+1)$ for more flexible kernel construction[6] which is defined as follows

$$C = sgn(k) \cdot sgn(l) \tag{23}$$

where $-M \leq k, l \leq M$, and the center column and line is zero-padded. Eventually, this formula yields the following matrix.

$$C = \begin{bmatrix} 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix} \tag{24}$$

The edge of kernels can be smoothed using a window function which tapper towards zero at edges. We use a radially-symmetric Gaussian function G defined by follows

$$G(s, t) = exp(-\xi^2(s^2 + t^2)) \tag{25}$$

where $\xi(\xi > 0)$ controls the degree of tapering(Figure 31). Then, the kernel matrix C is tapered by the Gaussian function as following

$$C_{Gauss}(s, t) = G(s, t) \cdot C(s, t) \tag{26}$$

Finally, we calculate a correlation between the windowed kernel matrix G_{Gauss} and the given SSM S. The correlation of each frames in the index $N(i)$ is calculated by

accumulating the product of C_{Gauss} and the corresponding area of the matrix S, where the C_{Gauss} is shifted along with the diagonal line of the S.

$$N(i) = \sum_{s=-L/2}^{L/2} \sum_{t=-L/2}^{L/2} C(s,t)S(i+s, i+t) \tag{27}$$

where $-L/2 < s, t < L/2$. For the concise calculation, the width of the kernel is defined as L whose center is on $(0, 0)$. In our Pseudo code shown below, the zero-padding is operated when computing the edge of the matrix S in order to avoid calculating undefined values. Below is of our implementation written in Swift.

Algorithm 1 Computing correlation of S and C

```
for i = 0 to I Step 1 do
    for s = 0 to L Step 1 do
        for t = 0 to L Step 1 do
            x = (i − L/2) + s
            y = (i − L/2) + t
            if 0 <= x < I then
                if 0 <= y < I then
                    N[i]+ = C_Gauss[s][t] · S[x][y]
                end if
            end if
        end for
    end for
end for
```

As we can see in both formulas and codes, the size of a kernel matrix has a significant impact on the resultant novelty function $N(i)$. Figure 32 and Figure 33 illustrates the filtering results of the matrix S with different size of the Gaussian windows (figure 31). Figure 34 to 36 illustrates the novelty function calculated by different kernel sizes. The kernel of the smaller size yields more sensitive peaks which have good time-localization. In contrast, the kernel of the larger size, which captures a larger number of adjacent frames, yields more smoothed and averaged fluctuations of the sound. While a small kernel is better suitable for detecting a short time scale, such as musical gestures or events, a large one is better for extracting boundaries and transitions between coarse structural sections[6].

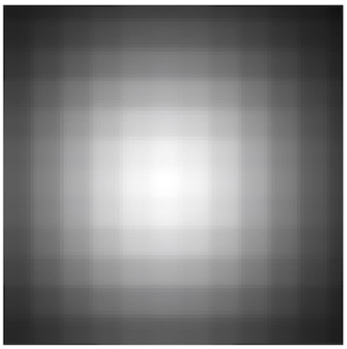


Figure 31: Gaussian window

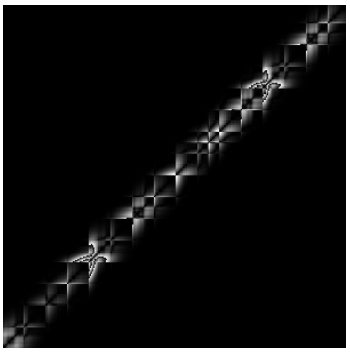


Figure 32: small kernel matrix

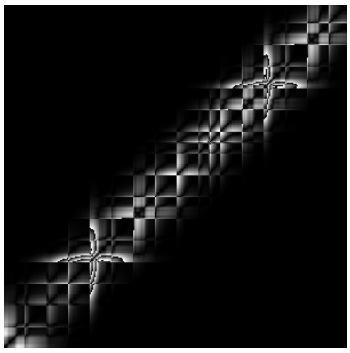


Figure 33: Large kernel matrix

Comparing the resultant novelty functions calculated with different kernel size and gaussian parameter ξ.

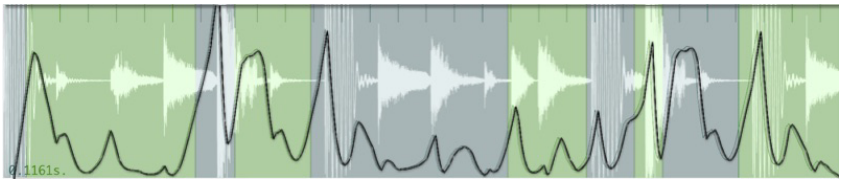


Figure 34: kernel size = 32, $\xi = 0.05$

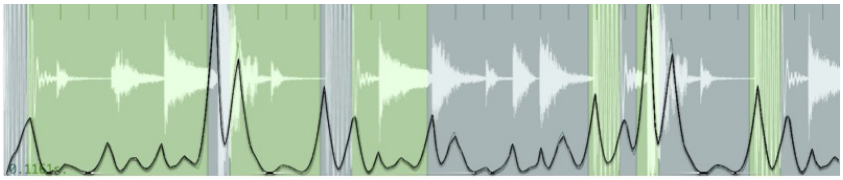


Figure 35: kernel size = 15, $\xi = 0.1$

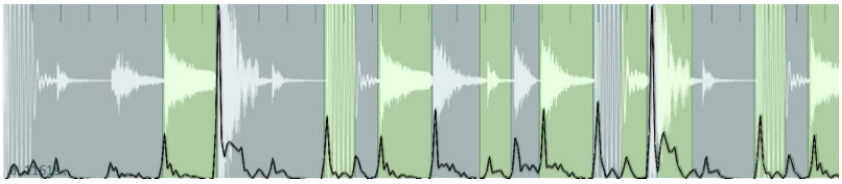


Figure 36: kernel size = 4, $\xi = 0.3$

9.2 Discussion

The novelty function derived from the delta-features is employed to extract the boundaries of the sound changes, which is a criterion of separating individual sound events.

The Delta-features based segmentation method is satisfactory for sensitive sound changes and is particularly effective to extract one short event or independent tone from melodies or rhythmic patterns. However, the boundaries are determined without any regard for the musical structure or pattern, and sometimes the resultant segments are separated haphazardly. This weakness is typically seen when processing a complex sound event such as sweeping musical gestures and fluttering sound effects. Although the delta-feature based segmentation is good at splitting adjacent sound sources, it is not capable of grouping a sequence of frames to preserve an independent sound event.

On the other hand, the novelty function derived from the SSM represents a structural pattern of the sound, and thus, it is more immune to the transient fluctuation of the sound property than the one based on the Delta-features. Figures 37 and 38 show the segmentation results of a short fragment played by a recorder. The fragment comprises the combination of flutter and pizzicato sounds. These figures show the difference in qualities between the Delta-feature and the SSM based segmentation. As the Delta-features detect detailed and sensitive sound changes, some very fast successive sounds are separated into distinct segments. The reason for the irregular segmentation is because the individual fluttering sound is much shorter than the FFT frame size and the properties of successive attack sounds are averaged in a unit. In contrast, the SSM based segmentation clearly separate individual sound structures, that is, a flutter, two pizzicati, and again, a flutter. Although the SSM based novelty function has less sensitivity for faint sound changes, it produces more efficient results for such sound examples than that from the Delta-features.

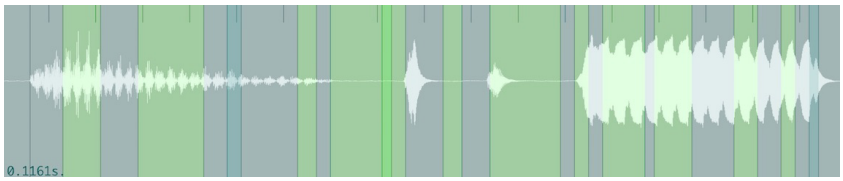


Figure 37: Segmentation result by the Delta-features

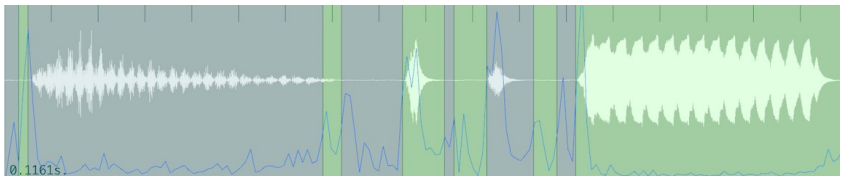


Figure 38: Segmentation result by the Self-Similarity Matrix

Upon consideration of the strengths and weaknesses of the two above-mentioned methods, it leaded us to consider dividing the segmentation method into two phases. In the first phase, the SSM method separates major sound events according to the structural pattern of the entire signal, which is the pre-segmentation. The resultant segments have better-isolated sound within a musically meaningful unit, but they may consist of multiple sound events and which is still complicated to reveal the time-varying components. Therefore, the Delta-feature method operates the second phase to separate the obtained major segments into more detailed sound components. In this thesis, we only use either the Delta-feature or the SMM based method in the analysis process and did not combine them. The further study of this collaborative segmentation method will be our next research interest.

Part IV
Audio Decomposition

10 Introduction

The audio signal often consists of different sound sources. For instance, a polyphonic music is comprised of multiple voices and rhythmic patterns played by various musical instruments together, and environmental sounds consist of multiple layers. The individual sources constituting the polyphonic structure may also consist of sound components of harmonic-inharmonic sound, variable timbre, percussive sound, and background noise. Once sound sources are mixed in the audio signal, the conventional spectral analysis is incapable of extracting the characteristics of individual components. The spectral analysis produces an exclusive feature, which describes an averaged characteristics of the assembled sound sources and in which, the original properties of each sources are hidden. This representation manner of a polyphonic sound is not appropriate for our system, particularly in the Music Information Retrieval(MIR) and Synthesis. When the targeted sound has multiple independent layers, it will be more difficult to search sufficiently matching sound units representing an entire structure of the target from a database. If the characteristic of the retrieved sound unit does not match adequately to the characteristic of the corresponding segments of the target, the quality of the synthesis result is reduced. Accordingly, it is more important to decompose a polyphonic sound into simpler and more isolated units before retrieving the best matching units of sound from a database. In this manner, segments of a polyphonic sound can be reconstructed with multiple sources.

In our synthesis system, we operate the temporal audio segmentation task for both the targeted audio signals and the source signals stored in a database in order to separate the complex sound structures into simple and readily usable units. However, we only operate the audio decomposition task for the targeted signals because the decomposition algorithms we will discuss in this section are not capable of reversing the decomposed components to the audible signals very well. The algorithm can reveal the polyphonic structure of the targeted sound and extract some features representing the characteristics of the decomposed components. However, the algorithms can only reproduce the audio signals of significantly low quality, which directly influences the

quality of the synthesis result. In our system, the targeted sound only provides a plan of the reconstruction structure, and the signal itself does not appear in the synthesis result. On the other hand, the sources retrieved from a database are practically synthesized, and thus they are present in the result. Therefore, we decided not to apply the audio decomposition for the sources. The study of a better reproduction of the decomposed audio signals is our future subject.

In order to decompose the polyphonic sound, we need to find boundaries of individual sound components. However, due to the complexity of the sounds, the boundaries are ambiguous, and they have different meanings depending on which levels we decompose the targeted sound. For instance, estimating a multiple fundamental frequencies from harmony or chord, and separating different sound components such as percussion-like sound, harmonic components, and background noises are another problems. Furthermore, the sound components are independently variable through time with or without correlations and they are complicatedly overlapping each other. A human being can distinguish such complicated sounds from each other, and well-trained people can even discriminate the different sound sources in their auditory system. However, it is an arduous task for a computer and thus, the audio decomposition is one of the most complicated subjects in the digital signal processing.

We apply decomposition task not only for separating the spectral information in the frequency domain, but also segmenting a sequence of individual components varying in time. In this section, we will first discuss the specmurt analysis for the multiple fundamental pitches estimation, which has already been introduced in the section 5.11, but here, we focus on its further applications. As the specmurt analysis employs a harmonic structure pattern, this algorithm is suitable for a polyphonic sound played by chamber instruments which have regular harmonic structures. However, we employ various overtone structure patterns instead of a single harmonic pattern in order to decompose a spectrum into diverse components. Secondly, we discuss the Non-negative Matrix Factorization which is a mathematical model factorizing a two-dimensional non-negative matrix into two different matrices. We apply this algorithm for decomposing a sequence of spectrums, which is the spectrogram, into its static spectra and variable magnitudes. This algorithm operates two tasks simultaneously, the spectrum decomposition and the temporal segmentation in a stochastic optimization manner.

11 Specmurt analysis based multi pitch estimation

As we discussed in the section 5.11, the specmurt analysis operates the deconvolution of the original spectrum by an overtone pattern[8], and as a result, it produces the fundamental pitch distributions. The idea of this algorithm is based on the assumption that the pattern of the sound source is constant and the overtone depends on the sound sources and the fundamental frequency. We have already shown some examples of the primary use of the specmurt analysis for polyphonic sound using the simple integral multiple overtone patterns. In the resultant spectra[9], which are shown in the section 5.11, the overtones of the original signal are successfully suppressed and the expected fundamental pitches remain. However, the resultant spectrum still has considerable amount of fluctuating peaks which are unwanted components for the multiple pitch estimation. These noise components derive from the dissimilarity between the model pattern and the analyzed overtone structure.

The cause of this issue is seen in the initially defined overtone pattern. It is rarely possible to define the exact pattern for a given sound source because each sound sources consisting a polyphonic sound has slightly different overtone structures depending on its fundamental frequency. Therefore, the specmurt analysis requires the best-compromised overtone pattern which satisfies the structures of all constituent components in order to produce the sufficiently satisfactory. However, it is not realistic to manually adjust the pattern by trial and error, and thus, Saito et al. suggested an optimization algorithm to find the best optimal overtone pattern to produce the best-compromised results[19]. This algorithm operates an adaptive estimation of the common overtone structure for each frame of the given signals which maximizes the resolution between significant fundamentals and other unnecessary components in specmurt analysis through interactive nonlinear mapping[19].

As described in section 5.11, the log-frequency spectrums of the pitch distribution $u(x)$, the original spectrum $v(x)$, and the overtone pattern $h(x)$ are represented in the specmurt domain $U(y)$, $V(y)$, and $H(y)$. The $U(y)$ is obtained by dividing $V(y)$ by $H(y)$.

[8] Sagayama et al. call the harmonic structure pattern but we here call it as the overtone pattern because we use versatile sound sources consisting of both harmonic and inharmonic structures.

[9] The resultant spectra are obtained by the inverse specmurt of the deconvolution result

$$U(y) = \frac{V(y)}{H(y)} \tag{28}$$

The resultant pitch distribution $U(y)$ is then transformed to the spectrum domain $\overline{u}(x)$ by the DFT.

The $h(x)$ is the manually defined overtone pattern. The adaptive estimation algorithm generates quasi-optimal $h(x)$ through iterations.

In the first step of the optimization process, we suppress the fluctuating erroneous peaks in the obtained pitch distribution $\overline{u}(x)$ by means of the nonlinear mapping which is based on the sigmoid function as follows.

$$\overline{u}(x) = \frac{u(x)}{1 + exp\{-\alpha(\frac{u(x)}{u_{max}} - \beta)\}} \tag{29}$$

where $\frac{u(x)}{u_{max}}$ is the normalization of $u(x)$ into a range $0 \leq (x) \leq 1$. This nonlinear mapping function suppresses the values around β and does not change values significantly larger than β. The inclination of a suppression curve is constrained by *alpha* as shown in the Figure 39. Through the nonlinear mapping process, the weak fluctuating peaks in the pitch distribution $u(x)$ are suppressed according to their magnitude and the thresholding function. As a result, the updated $\overline{u}(x)$ has less unwanted peaks and preserves the expected fundamental pitches.

Sigmoid Function

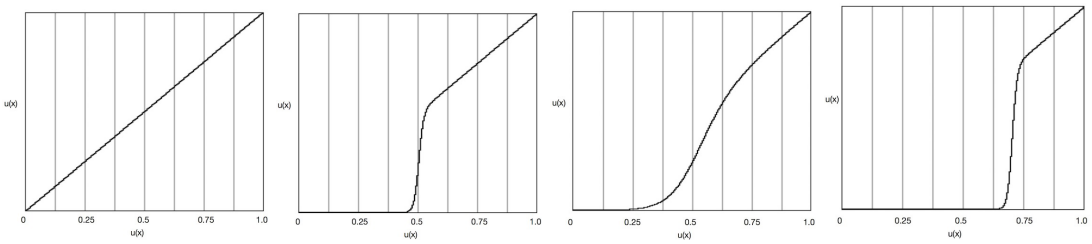


Figure 39: α=100, β = 100

Figure 40: α=100, β = 0.5

Figure 41: α=15, β = 0.5

Figure 42: α=100, β = 0.7

11.1 Iterative estimation algorithm

According to Saito, the iterative estimation algorithm operates as the following process [21]. First calculating $u(x)$ from $v(x)$ with an initially defined overtone pattern $h(x)$ as described above and then, obtaining $\overline{u}(x)$ by applying a nonlinear mapping.

Subsequently, finding a better optimal overtone pattern $\overline{h}(x)$ and update $h(x)$ with it. Eventually go back to the first step, and repeat this process with the new $h(x)$ until we obtain the best optimized $\overline{h}(x)$. The goal of this algorithm is to find a set of the adequate amplitude ratios corresponding to the n number of initially defined overtones. For the calculation, we define an updating function with a parameter θ which is the amplitude ratio of the nth overtone relative to fundamental pitch.

With regard to these parameters, the $\overline{h}(x)$ is defined as follows.

$$\overline{h}(x,\theta) = \sum_{n=1}^{N} \theta_n \delta(x - \Omega_n) \quad (1 \leq n \leq N : N \in \mathbb{N}) \tag{30}$$

where Ω_n is the x-coordinate of the nth harmonic overtone on the log-frequency scale which is calculated by $\Omega_n = logn \cdot 1200/r$. r represents the log-frequency resolution of the constantQ transform, and is calculated by $r = cent/bin$. N is a number of overtones and $\Omega_1 = 0$, $\theta_1 = 1$ because the first frequency corresponds to the fundamental frequency located on the first bin in the overtone pattern and has a magnitude value of 1.

Now, we calculate new overtone pattern by optimizing the parameter θ. As we assume $v(x) = h(x) * u(x)$, the temporal $\overline{v}(x)$ which is calculated by the updated $\overline{h}(x)$ is described as $\overline{v}(x) = \overline{h}(x,\theta) * \overline{u}(x)$. We want to have a set of θ which minimize the integral square error $\Delta = \{v(x) - \overline{v}(x,\theta)\}^2$. The square distance of all overtone is then described as follows.

$$E(\theta) = \sum_{i=0}^{I-1} \{v(x_i) - \overline{h}(x_i,\theta) * \overline{u}(x_i)\}^2 \tag{31}$$

where I donates the number of log-frequency samples. We will estimate the parameter θ by minimizing $E(\theta)$.

Since we only focus on the parameter θ, the quasi-optimal solution can be obtained by partial differential equations by the parameter θ.

$$\frac{\partial E(\theta)}{\partial \theta_n} = 0 \quad (2 \leq n \leq N) \tag{32}$$

This cost function is only valid for the range of the overtone which is the higher than 2ed harmonics which is therefore, represented by $n : ((2 \leq n \leq N))$. We can re-write the above equation as the following matrix equations:

$$\begin{bmatrix} a_{1,1} & \dots & a_{1,n} & \dots & a_{1,N} \\ \vdots & & \vdots & & \vdots \\ a_{n,1} & \dots & a_{n,n} & \dots & a_{n,N} \\ \vdots & & \vdots & & \vdots \\ a_{N,1} & \dots & a_{N,n} & \dots & a_{N,N} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \\ \vdots \\ \theta_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \\ \vdots \\ b_N \end{bmatrix} \tag{33}$$

The calculation of each cell is described as follows.

$$a_{j,k} = \sum_{i=0}^{I-1} u(x_i - \Omega_j) u(x_i - \Omega_k) \tag{34}$$

$$b_j = \sum_{i=0}^{I-1} \{\overline{v}(x_i) - u(x_i)\} u(x_i - \Omega_j) \tag{35}$$

The matrix θ is then obtained by the convolution of the matrix a by the inverse of the matrix b. Below, we show the simplified C source code written by us which updates the overtone pattern according to the above algorithm.

```
1
2
3   int size = 10; // overtone number
4   int N = 56; // log-frequency resolution (bin number of the ConstantQ Transform)
5   int overtone_x[size]; // x-coordinate of n-th overtone
6   double delta[size]; // delta v-u
7
8   //calculating the matrix a
9     for(j=0;j<N;j++){
10        for(k=0;k<N;k++){
11            for(i=0;i<size;i++){
12                index_j = i - overtone_x[j];
13                index_k = i - overtone_x[k];
14                if(index_j>=0 && index_k >=0){
15                    a[j*N + k] += u[index_j]*u[index_k];
16                }
17            }
18        }
19    }
20
21    //calculating the first term of the matrix b
22    for(i=0;i<size;i++){
23        delta[i] = (v[i]-u_sigmoid[i]);
24    }
25
26    //calculating the matrix b
27    for(j=0;j<N;j++){
28        for(i=0;i<size;i++){
29            index_j = i - bLog[j];
30            if(index_j >= 0){
31                b[j] += delta[i]*u[index_j];
32            }
33        }
34    }
35
36    //inverse matrix b
37    b = m_inv(N, N, b);
38
```

```
39      //updating the overtone pattern.
40      long mc = N;
41      long nc = 1;
42      double amplitude[N]; // buffer for the updated overtone pattern
43      memset( amplitude, 0, N*sizeof(double) );
44      //inner product of the matrix a and the inversed matrix b
45      m_mul(N, N, N, 1, &mc,&nc, N,a, b, amplitude);
46
47      amplitude[0] = 1.0;// the first overtone equal to the fundamental pitch
```

Listing 1: Updating overtone pattern in C code

11.2 Result

Figure 44 shows the progress of the iterative estimation. Each line has the original spectrum, the initially defined overtone pattern, deconvolve spectrum, and the nonlinear mapping result. The second to fourth lines show the iterative estimation process; the updated overtone pattern, deconvolve spectrum, and a nonlinear mapping result. We used piano tetrad of C-major C3-E3-G3-C4(`C3E3G3C4.wav`), and 10 number of integral multiples overtone pattern whose amplitude ratios are $(1.0, 1/2, 1/3, \ldots, 1/10)$. The first deconvolution result, which is shown in the first line, involves many strong unwanted peaks on the entire spectrum due to the dissimilarity of the overtone structure. These unwanted peaks are not sufficiently suppressed because their amplitudes are still higher than the threshold of the mapping function. The first optimal result, which is in the second line, shows optimized overtone pattern which produces slightly better deconvolution result and a better mapping result. The updating weight of the overtone pattern is calculated by the calculation of the square distance $\{v(x) - \overline{v}(x,\theta)\}^2 = \{v(x) - \overline{h}(x,\theta) * \overline{u}(x)\}^2$ where the $\overline{u}(x)$ is the suppressed pitch distribution by the nonlinear mapping. Therefore, the parameters of α and β for the sigmoid function indirectly constrain the updating weight.

The overtone pattern is modified step by step according to the update rule, and it improves the deconvolution result. The third optimal result shows an adequate fundamental pitch distribution which has four high peaks corresponding to the given tetrad. However, the 5th optimal result shows only three peaks corresponding to C3-E3-G3 and C4 is missing which is an octave higher than the lowest pitch C3. As we can see the overtone pattern in the 5th optimal result, the second overtone is the highest of all, and which corresponds to the same frequency of C4. Therefore, the optimization process regarded C4 as a strong second overtone of C3, and suppressed it throughout the iterative estimation process. This problem can be solved by giving a

softer suppression weight for the nonlinear mapping or controlling the number of the iterative estimation. However, when the suppression weight is too soft, the unwanted peaks are not sufficiently suppressed, and the optimization has never been convergent to the expected result. Similarly, the number of iteration is also important to obtain a reliable result. Saito investigated the relationship between iteration times and updating weight, and he found that the weight decreases logarithmically and becomes very small at around third to fourth iterations, and then, it is almost vanishingly small at fifth iteration[21].

The iterative estimation is a useful algorithm for estimating the multiple fundamental pitch distributions of polyphonic music. The specmurt analysis resolves the given spectrum into a constituent spectral component and its fundamental frequencies. However, the result is significantly influenced by the initially defined overtone pattern. The algorithm optimizes the amplitude ratio of each overtone, but it does not control the number of overtones and their x-coordinates in the frequency domain. The approximate overtone pattern of the targeted audio signals needs to be known in order to obtain precise results. Furthermore, the components comprising the polyphonic sound need to have a common overtone pattern. Therefore, the specmurt analysis is valid only when separating a given audio signals into voices of the same sound qualities.

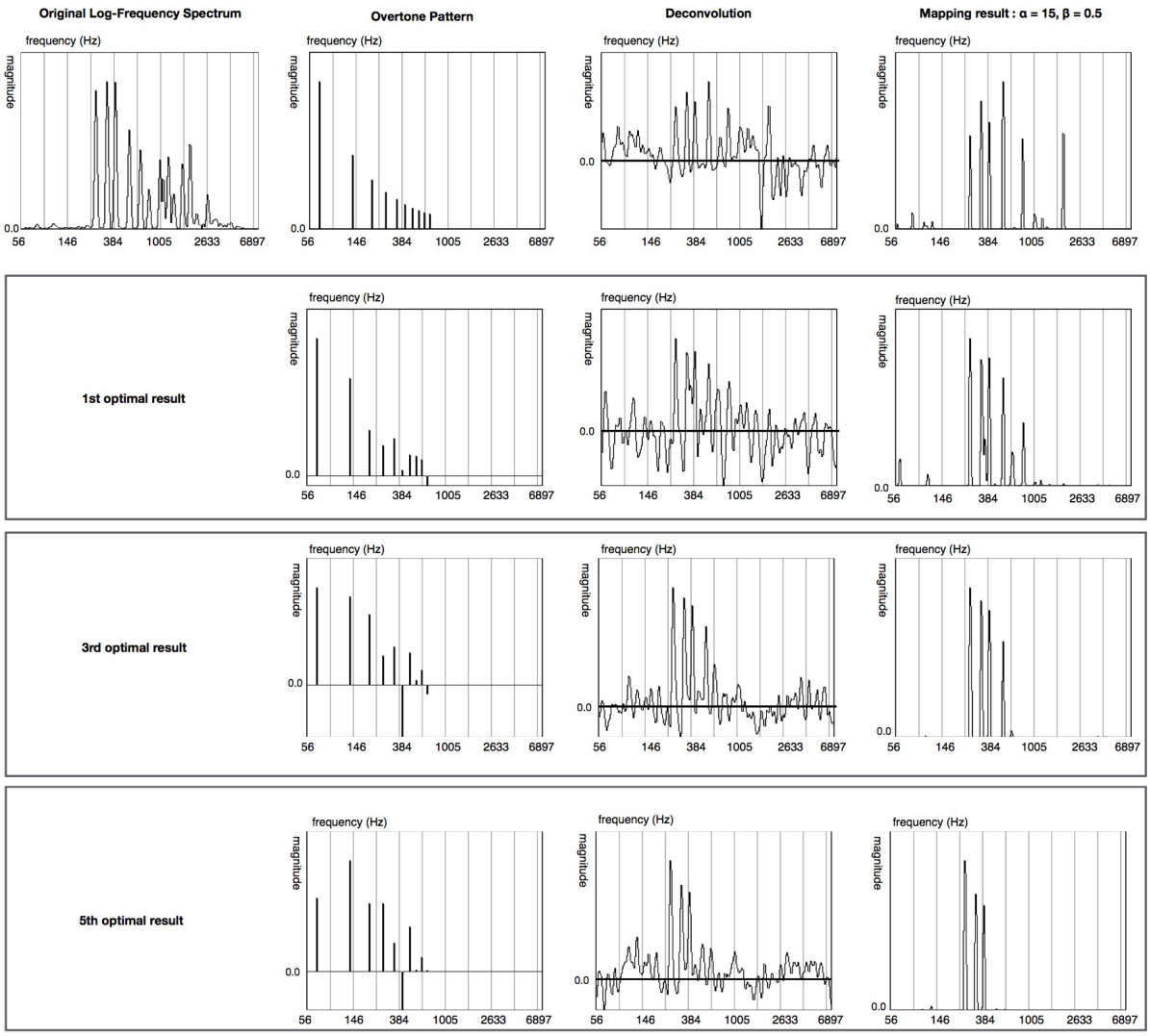


Figure 43: The result of the specmurt analysis with the iterative estimating algorithm.

11.3 The spectrum decomposition using freely defined overtone pattern

We found another possibility to apply the specmurt analysis to the audio decomposition. As already described, the concept of the specmurt analysis is based on the assumption that the all tones in a polyphonic sound have a common overtone pattern, and its spectrum can be described as a sum of linearly stretched common overtone structures alongside the loge-scaled frequency[21]. Conversely, this assumption suggests that the targeted spectrum can be resolved into any combinations of the fundamental frequencies and optionally defined overtone patterns. Now, our aim is not to estimate a proper fundamental pitch distribution from the given spectrums but to decompose the spec-

trums into a set of particular sound characteristics, such as clarinet, bell, Formant, and piano by defining variable harmonic or overtone patterns. In this idea, the specmurt analysis results in a set of notes or a chord of the indicated overtone pattern.

The figure 44 and 45 illustrate the specmurt analysis using the odd multiple overtone pattern(frequency ratio: $1, 3, 5, \ldots, 2n-1$, amplitude ratio $1, 1/3, 1/5, \ldots, 1/(2n-1)$) and the bell overtone pattern respectively (see figure 46). The odd-multiple overtone structure models upon the clarinet harmonic pattern, and the bell overtone structure is taken from the FFT analysis result by Jonathan Harvey. The notes of the overtone patterns are shown in the figure 46. We again used a piano tetrad C3-E3-G3-C4 as the targeted polyphonic sound and operated five-time iterative estimation process in order to optimize the initially defined overtone pattern. The unsuitable overtone pattern possibly yields a frequency distribution which satisfies the condition of $\overline{u}(x) \leftarrow \arg\min_{\overline{u}(x)} D(x)$.

With the odd-multiple overtone, a larger number of the fundamental frequencies are observed than the integral multiples because the fewer overtone peaks in the pattern can deconvolve less corresponding peaks in the original spectrum. Since the odd-multiples do not comprise the octave relatives, the second overtones of all fundamental frequencies are not suppressed through the deconvolution. As a result, the frequency distribution is relatively close to the original spectrum. On the other hand, we obtained a contrasting result with the bell overtone pattern which comprises the irregular frequency ratio. The resultant fundamental frequencies are disordered compared with those of the integer- and odd-multiple overtone patterns. A significant dissimilarity between the original and bell overtone patterns are observed in the high-frequency band, which provokes the small suppression rate in the high registers of the fundamental frequencies.

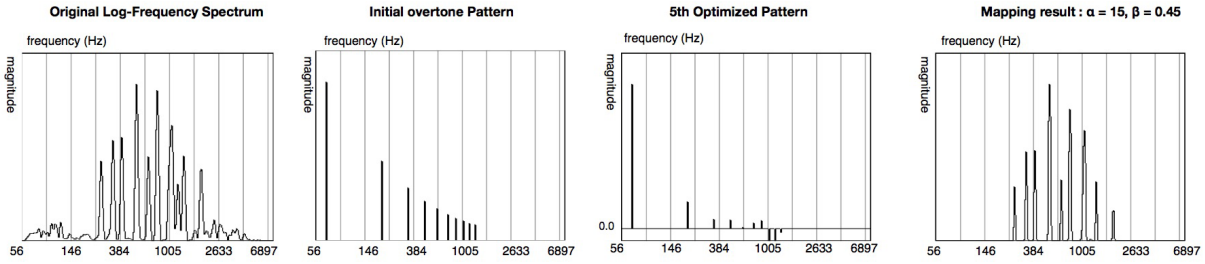


Figure 44: The specmurt analysis using the odd multiple overtone pattern.

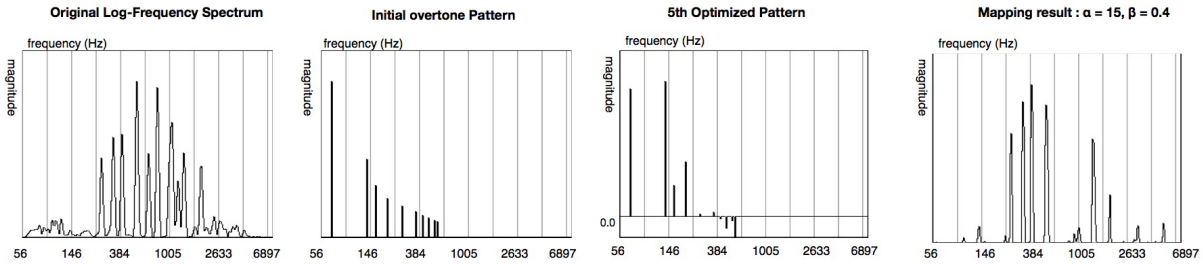


Figure 45: The specmurt analysis using the bell overtone pattern.

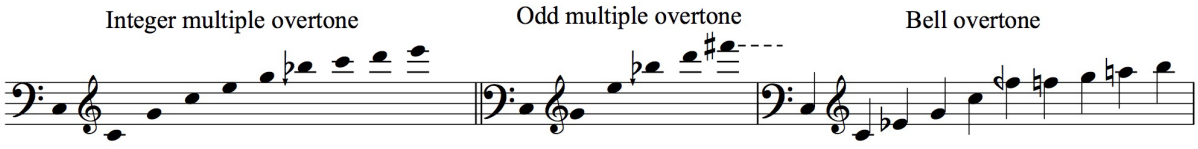


Figure 46: The overtone pattern.

11.4 Experimental evaluation

As we can see in the results, the specmurt analysis using an optionally defined overtone pattern seems to have properly decomposed the spectral structure into several voices. The resultant frequency distribution gives sufficient quality in appearance. However, because we intentionally apply the erroneous operations of the specmurt analysis by using unfitting overtone patterns, its result is not always reliable. In fact, more fluctuating peaks are obtained in the resultant deconvolution due to the dissimilarity problem than the one using the appropriate overtone pattern. We can suspect that the resultant fundamental frequency distribution is only sufficient due to the suppression of the nonlinear mapping. From this point of view, we assessed the analysis result by measuring the correlation between the result and the original spectrum in order to inspect the quality of the decomposition. The correlation is calculated by measuring the distance D between the original spectrum and the reversed spectrum obtained by the convolution of the resultant fundamental frequencies by the optimized overtone pattern. The distance D is a sum of the delta magnitude of each corresponding coefficients between two spectra, which represents its dissimilarity. Let the reversed spectrum in the x the frame $\overline{v}(x) = \overline{h}(x) * \overline{u}(x)$, where $\overline{h}(x)$ donates the optimized overtone pattern and $\overline{u}(x)$ donates the resultant pitch distribution, hence D is calculated as follows,

$$D(x) = \sum_{i=0}^{i=I-1} |v_i(x) - \overline{v}_i(x)| \quad (0 \leq i < I) \tag{36}$$

where i donates the FFT index in the log-frequency domain. We determine that the result is reliable when the $D(x)$ is small enough.

Figures 47 and 48 illustrate the reversed spectrums in the different optimization levels, and its corresponding distances D. The original spectrum is shown in figure 44. The successful iterative estimation produces a reverse spectrum which is apparently similar to the original, and indeed, it has a smaller distance. Table 11.4 shows how the distances change throughout the iterative estimation process in the three overtone patterns.

Table 2: The changes of the distances D throughout the iterative estimation using three overtone patterns.

Iteration	D of integer multiple	D of odd multiple	D of bell
1	0.322415	0.295136	0.303621
2	0.209612	0.289617	0.202987
3	0.197715	0.230458	0.216729
4	0.208607	0.252578	0.195380
5	0.220370	0.264129	0.192255

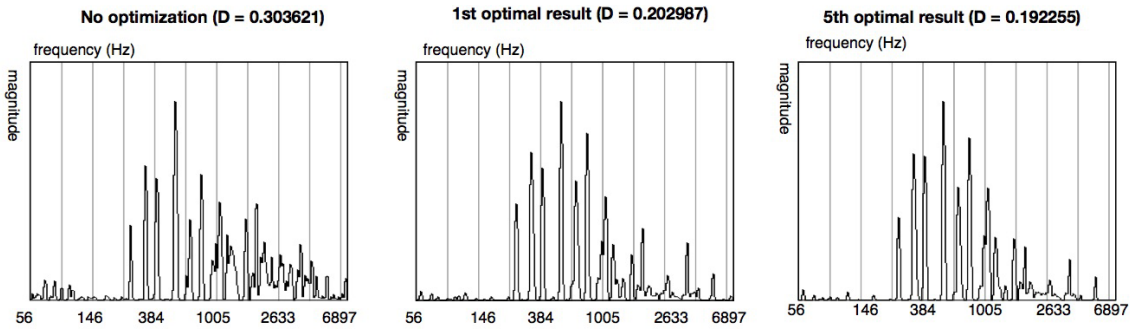


Figure 47: The reversed result of the bell overtone pattern in the 15th frame.

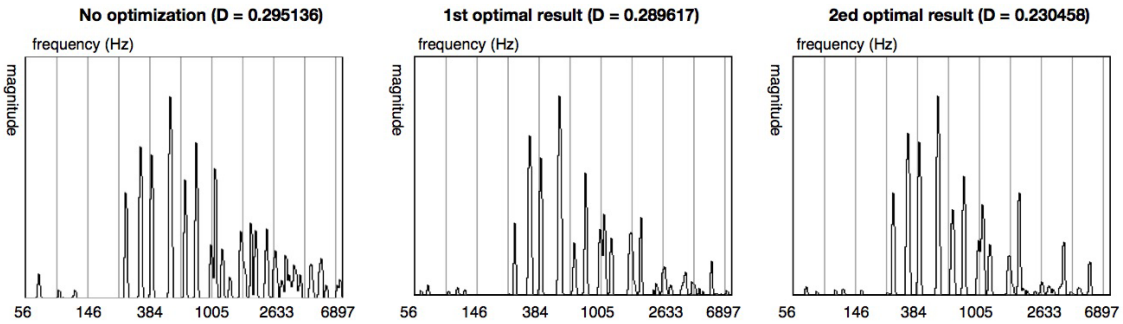


Figure 48: The reversed result of the odd overtone pattern in the 15th frame.

The dissimilarity usually gets smaller as the iteration advances, but some exceptions are observed depending on the progress rate of the optimization. For instance, in the cases of the integral and odd-multiple overtone, the optimization resulted in the minimum distances at the third iteration but the distances again increase from the fourth iterations. One reason would be that a parameter of an extremely rigid threshold for the nonlinear mapping can cause superfluous suppressions on the peaks of fundamental frequencies and which produces erroneous optimizations. Another reason to be noted is that the fundamental frequency is regarded as one of a constituent overtone of its lower fundamentals, which is why it is suppressed in the iterative process as discussed in the section 11.1. When the suppressed fundamentals are lower than the threshold of the non-linear mapping, they are even more suppressed and eventually disappear from the resultant pitch distribution. As a result, the distance between the original and the reverse spectrums increase more than the previous estimation.

To avoid the erroneous optimization, we constrain the iteration process according to the optimization progress. As we have already introduced the evaluation carried out by Saito [21], the iterative estimation has significant update weight in the first few iterations, and then it vanishes abruptly around the fifth iteration. We define the condition of the iteration where the estimation repeats when $D_\theta(x) < D_{\theta+1}(x)$, where the θ donates the iterative time. The best optimized overtone pattern is then extracted by interrupting the iterative estimation process when $D_\theta(x) < D_{\theta+1}(x)$, where the θ donates the iterative time.

11.5 Discussion

We have introduced the advanced specmurt analysis using the iterative overtone estimation algorithm. Our study also suggests that the analysis using various overtone patterns is capable of separating the given spectrum into a particular sound quality. However, further study is still required to represent a time varying structure between frames. A crucial issue of the specmurt analysis is that the fundamental frequency distribution is estimated from an individual spectrum, which has no necessary correlations with the adjacent frames. The iterative estimation algorithm is operated for each frame independently, and the overtone pattern is not optimized for the whole sound. Therefore, the specmurt analysis does not represent a sequence of polyphonic structure varying in time.

There are some studies applying the specmurt analysis for the temporal music representation such as automatic transcription system. For instance, Sagayama et al. used a piece of music played by a Piano or Guitar solo and they simply translated the resultant frequency distribution to a piano-roll representation[19]. Azuma et al. employed a Hidden Markov Chain in order to treat the extracted frequencies as a melody or voice part[22]. In all cases, the resultant fundamental frequencies are translated to MIDI or similar format and represented in the classical notation system and are not utilized as spectral information of the decomposed sound components. Since their study aims to capture the fundamental pitches, the process is relatively simple compare with our needs.

For the temporal representation, we have already suggested various techniques which interpolate descriptive features varying in time, but they do not always bring a satisfactory result in the decomposition. The polyphonic audio signals can have more complicated structures where, each constituent voice varies individually over time, and it is very difficult to track each of them independently. We operated the specmurt analysis under the assumption that the overtone pattern remained constant throughout the different fundamental pitches, and now we assume that the pattern is also constant in time sequence. Therefore, the polyphonic structure is decomposed into multiple sets of an overtone pattern and its variate amplitude in time. This idea led us to atoner mathematic model which is the Non-Negative Matrix Factorization discussed in the next section.

12 Non-Negative Matrix Factorization

After a discussion of the decomposition of both frequency and temporal information, we will now introduce the Non-Negative Matrix Factorization(NMF) which is a useful decomposition technique applying unsupervised statistical analysis. In order to decompose the spectral structures into multiple vectors and their magnitude over time, we need to handle the Spectrogram instead of an individual spectrum. The spectrogram is a representation of a sequence of spectra produced by the STFT, which illustrates frequency information and time variance in a 2D image as shown in figure 49. The vertical line represents the frequencies, the horizontal line represents the frame index in time, and the colors illustrate the magnitude of the corresponding frequency. The red and the blue color indicate the high dynamics and the low dynamics. While a spectrum represents a snapshot of the audio signal, the spectrogram illustrates an entire state of it.

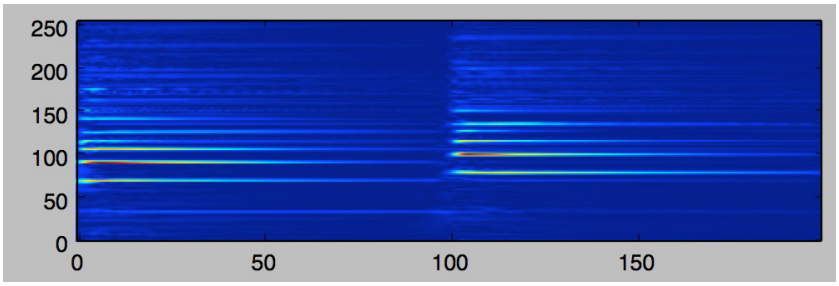


Figure 49: Spectrogram of Piano sound playing C major and c minor successively.

The NMF represents a multivariate data as a product of two specific properties, which is applied for decomposing a spectrogram into a product of multiple spectrum profiles and their magnitudes varying over time. While the spectrum profile is static over the entire signals, the magnitude represents its temporal information. A set of the static spectrum and its time-variant magnitude information is called as a vector, and the NMF operates a factorization of the targeted spectrogram into a particular number of vectors.

The basic idea of this factorization algorithm was introduced by P. Paatero in 1994[43], and the advanced version was developed by D. Lee and H.Seung in 1999[41]. Since the optimization algorithm was suggested by Lee et al. in 2001[42], and since then, this has actively been explored for further application[44]. According to D. Lee

and H.Seung[41], the NMF is defined to solve the following problem. We consider a non-negative matrix V to find non-negative matrix factors W and H that are defined as follows.

$$V \approx WH \tag{37}$$

V is a set of multivariate n-dimensional data vectors of the size $n \times m$, where the m donates the number of samples in the data set. Each vector is placed in the columns of an $n \times m$ in the matrix. This matrix is approximately factorized into two representative matrices, W the size of $n \times r$, and H the size of $r \times m$. The parameter r, which is referred to as the vector number of the factorization, is usually smaller than m or n and thus, the matrix sizes W and H are smaller than the original matrix V. Therefore, the product WH is a compressed version of the original data matrix.

Since only the original matrix V is definite, the two matrices W and H are found through the unsupervised statistical optimization process using a cost function. At the beginning of the process, W and H are initialized with non-negative random numbers. The goal of this algorithm is to find a pair of W, H which minimizes the cost function, for example, when we use a Euclidean distance, we find a pair which satisfies the following condition.

$$find\ (W, H)\ minimize \|V - WH\|^2 \tag{38}$$

The above formula qualifies the Euclidean distance between V and WH. Lee et al. also suggested qualifying the divergence of V from WH, which reduces to the Kullback-Leibler divergence[42]. H. Kameoka mentioned that the Itakura-saito distance is also beneficial for the NMF[44]. Their discussions suggest that the cost function is a significant factor for the statistic optimization process. Although it is very interesting to investigate more deeply about the differences of the results depending on the cost function, we here employed only the most simple cost function, which is Euclidean distance.

12.1 Non-Negative Matrix Factorization for music deomposition

We have described the basic idea of the NMF and now, we will discuss more details of the NMF applied for music decomposition according to the algorithm introduced by H.

Kameyama 2012[44].

As described above, we consider a set of multivariate data vectors $Y \in \mathbb{R}_{\geq 0}^{K \times N}$, and now, we determine Y as a spectrogram of the audio signal, where K donates frequency index, which is a FFT frame size, and N donates the time, which is a frame number. With the factorized vector number R, the spectrum vectors are represented as $H \in \mathbb{R}_{\geq 0}^{N \times R}$, and the magnitude vectors as $U \in \mathbb{R}_{\geq 0}^{M \times R}$. The problem to be solved is to find a pair of factorized vectors U and H from Y and therefore, represented as follows.

$$Y \approx UH \tag{39}$$

As previously described, three cost functions are introduced; Euclidean distance, I-divergence, and Itakura-saito distance, that are defined with $y, x \in \mathbb{R}$as follows.

$$\mathcal{D}_{EU}(y|x) = (y - x)^2 \tag{40}$$

$$\mathcal{D}_{KL}(y|x) = ylog\frac{y}{x} - y + x \tag{41}$$

$$\mathcal{D}_{IS}(y|x) = \frac{y}{x} - log\frac{y}{x} - 1 \tag{42}$$

All functions become 0 when $x = y$ and get lager when bigger distance between x and y is observed. The I-divergence and Itakura-saito distance are not symmetric and have greater increment when $x < y$. The distance between matrices $Y(y_{k,n})$ and $U(u_{m,n}) \cdot H(h_{k,m})$ are calculated by summing the distances of all sample data.

$$\mathcal{D}(HU|Y) = \sum_{k,n} \mathcal{D}(y_{k,n}|\sum_{m} h_{k,m}u_{m,n}) \tag{43}$$

Here, we employ Euclidean distance which is the most simple cost function.

$$\mathcal{D}_{EU}(Y|HU) = \sum_{k,n} \|y_{k,n} - \sum_{m} h_{k,m}u_{m,n}\|^2 \tag{44}$$

$$\tag{45}$$

and when we define

$$x_{k,n} = \sum_{m} h_{k,m}u_{m,n} \tag{46}$$

The formula 44 is represented as follows.

$$\mathcal{D}_{EU}(H,U) = \sum_{k,n}(-2y_{k,n}x_{k,n} + x_{k,n}^2) \tag{47}$$

The update rules of H, U is then described as follows.

$$H,U = \arg\min_{H,U} \mathcal{D}(HU|Y) \quad (h_{k,m}, u_{m,n} \geq 0) \tag{48}$$

Since only Y is known, and two matrices H, U are unknown. This mathematic model is usually called the optimization problem and the problem to be solved is to find the best optimized pair which approximately represent the original spectrogram Y.

An Auxiliary function is employed for the optimization. The function uses representative function $\mathcal{Q}$ which is iteratively updated to minimize the cost function and it is the generalized version of the EM algorithm [45]. Kameoka introduced Jensens Inequality which is defined as follows.

$$g(\sum_i \lambda_i z_i) \leq \sum_i \lambda_i g(z_i) \quad (1 \leq i \leq I) \tag{49}$$

When g is a quadratic equation of a downward convex, and the weight coefficients λ satisfies $\sum_i \lambda_i = 1$, the inequality shows that a downward convex parabola $g(\sum_i \lambda_i z_i)$ is always smaller than the parabola $\lambda_i g(z_i)$. Since the cost function also yields a downward convex, the Jensens Inequality is valid for finding the variables which minimize the cost function.

In the case of the Euclidean distance shown in the formula 47, we only need to minimize the item of x^2 and thus, the Jensons Inequality can be written with the parameter $\lambda_{k,m,n}$ as follows.

$$(\sum_m \lambda_{k,m,n} x_{k,n})^2 \leq \sum_m \lambda_{k,m,n} x_{k,n}^2 \tag{50}$$

$$\sum_m \lambda_{k,m,n}^2 x_{k,n}^2 \leq \sum_m \lambda_{k,m,n} x_{k,n}^2 \tag{51}$$

$$x_{k,n}^2 \leq \sum_m \frac{\lambda_{k,m,n} x_{k,n}^2}{\lambda_{k,m,n}^2} = \sum_m \lambda_{k,m,n} (\frac{x_{k,n}}{\lambda_{k,m,n}})^2 \tag{52}$$

as we defined $x_{k,n} = \sum_m h_{k,m} u_{m,n}$, the Auxiliary function is eventually defined as follows.

$$x_{k,n}^2 \leq \sum_m \lambda_{k,m,n} (\frac{h_{k,m} u_{m,n}}{\lambda_{k,m,n}})^2 \quad (\lambda > 0, \sum_m \lambda_{k,m,n} = 1, \lambda_{k,m,n} = \frac{h_{k,m} u_{m,n}}{x_{k,n}}) \tag{53}$$

where $\lambda_{k,m,n} = \frac{h_{k,m} u_{m,n}}{x_{k,n}}$ is a normalization of the $h_{k,m} u_{m,n}$.

Finally, we operate the iterative optimization to find the matrixes H, U, λ which minimize the cost function. The best approximate H, U is calculated by repeated iteration in the following order with the above described Auxiliary function $\mathcal{G}_{EU}$.

$$\lambda \leftarrow \arg\min_{\lambda} \mathcal{G}_{EU}(H, U, \lambda) \tag{54}$$

$$H \leftarrow \arg\min_{H} \mathcal{G}_{EU}(H, U, \lambda) \tag{55}$$

$$U \leftarrow \arg\min_{U} \mathcal{G}_{EU}(H, U, \lambda) \tag{56}$$

Lee et al. found a multiplicative update rules which are good compromise between speed and case of implementation for solving the above formulas.

The update rules of the Euclidean distance $\|Y - HU\|$.

$$H_{k,m} = H_{k,m} \frac{[U^T Y]_{k,m}}{[U U^T H]_{k,m}} \tag{57}$$

$$U_{m,n} = U_{m,n} \frac{[H^T Y]_{m,n}}{[U H H^T]_{m,n}} \tag{58}$$

The update rules of the I-divergence distance $\mathcal{D}(Y|HU)$.

$$H_{k,m} = H_{k,m} \frac{\sum_n U_{m,n} Y_{n,k} / (UH)_{n,k}}{\sum_i U_{m,i}} \tag{59}$$

$$U_{m,n} = U_{m,n} \frac{\sum_n H_{m,k} Y_{n,m} / (UH)_{n,k}}{\sum_j H_{k,j}} \tag{60}$$

We give a parameter which constrains a number of factorizing vectors. The vector number significantly influences the quality of factorization result. For instance, with a vector number of two, the NMF factories the targeted spectrogram into two sets of a spectrum profile and its magnitude. Since the spectrum is static in a vector, the time varying frequency information is significantly reduced. In contrast, a large number of vector number is capable of interpolating more detailed variation of the frequency information.

12.2 Experiments

We will discuss three different examples of various sound qualities and structures. For the first experiment, we analyzed a short music fragment played by piano. It consists of four notes of C4, D4, E4, and F4, and which comprises a simple monophonic melody whose score is presented in figure 50 and it is audible in (`doremi.wav`). Although the music structure is monophonic, the resonances of the neighboring notes are overlapped to each other, which means that the fragment has a polyphonic sound structure. We considered decomposing the targeted audio signals into four vectors as the fragment consist of four notes. The NMF has been operated under the condition of the FFT size of 1024 and the overlap number of 2. Figure 50 illustrates the resultant NMF where the signal is decomposed into four sets of the spectral profile and its temporal information. The left column shows the reversed signals of the factorized results and the right column shows their corresponding spectral profiles. The reversed signals are reproduced by the convolution of the spectrum profile and its magnitude information in time. Since the spectrum profiles are static in a vector, the reproduced audio signals preserve the same sound quality and they are only variable in their magnitudes. The audio data of the factorized vectors are audible in (`doremi_1.wav`) to (`doremi_4.wav`).

The signals corresponding to the four notes are sufficiently separated into individual vectors, and noteworthy, the overlapped resonances, which are the polyphonic components, are also separated together. The piano sound has very similar spectrum structure typically in the same registers as we can see in the right column of the figure. These spectra are however, regarded as different profiles even a half step changes in the linear-frequency domain and therefore, the different notes are classified into separate vectors. The vector1 undertakes note of the pitch D3, and the vector2 of E3, the vector3 of F3, and the vector4 of C3. In this example, we focus on separating each note including its resonance, and we were able to obtain an adequate result. In this example, the simple monophonic music sound is used and the number of voices are know. We will use more complex sound samples in the next experiments.

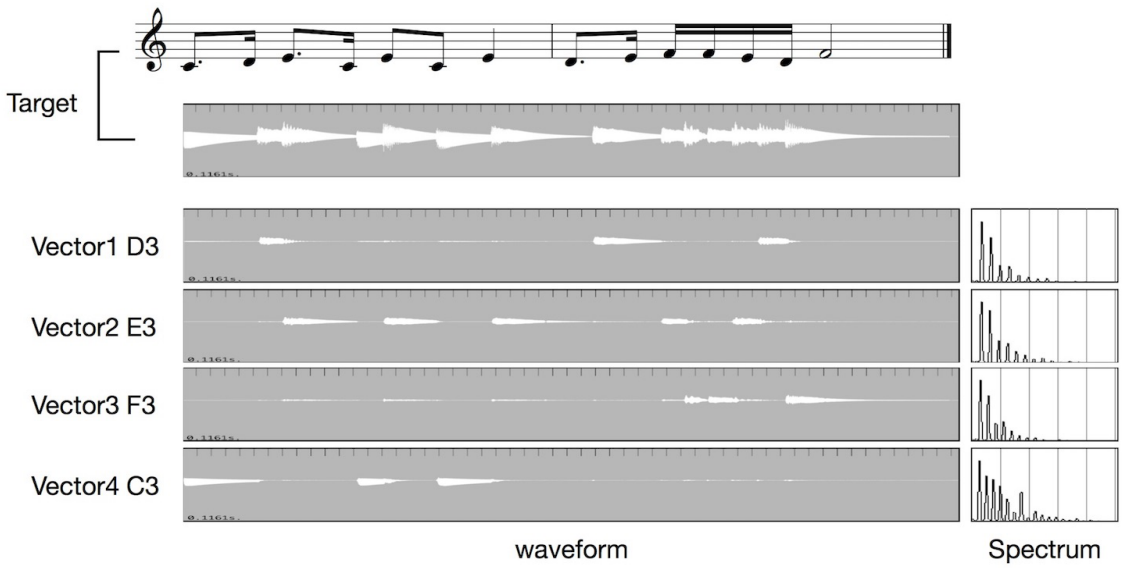


Figure 50: The NMF result of a piano fragment.

The targeted sound file is `doremi.wav` *and the factorized vectors are audible in* `doremi_1.wav` , `doremi_2.wav`, `doremi_3.wav`, *and* `doremi_4.wav` *where the index numbers represent the corresponding vector indices.*

We have tested more complex sound samples and this time, we rather focused on decomposing different sound qualities and separating individual notes. The first is a drum loop which we have already used in the several analysis examples. The process is operated under the condition of the vector number of 4, the FFT frame size of 1024, and the overlap number of 2. Figure 51 shows the target waveform, and the factorized results. The decomposition result is audible in (`drumLoop_1.wav`) to (`drumLoop_4.wav`). The magnitude information is also illustrated on the corresponding waveforms by black lines. The factorized audio signals can be segmented in time by means of delta-Magnitude feature. In this example, the different sound qualities are factorized into individual vectors, for instance, the very high frequency and low-frequency components, metallic-like sound, and snare-like sound are separated. The decomposition is concentrated on around the bass drum sounds. Indeed, the three vectors represent the separated sound components around bass drums and only one vector is for another component. It is assumed that the various percussions such as hi-hat and bells are played together on the bass drum and therefore, the sound has wide range frequency and is easily decomposed into multiple vectors.

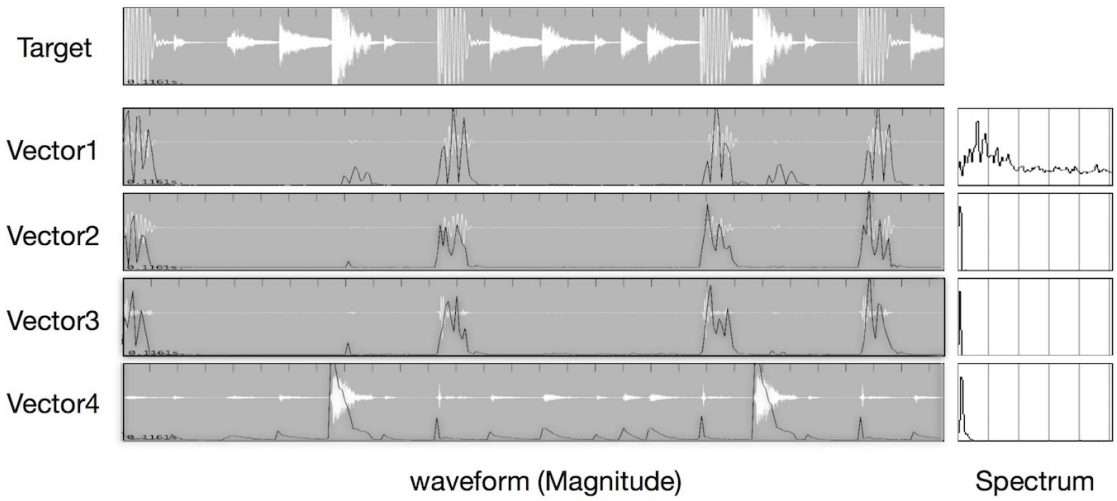


Figure 51: The NMF result of a drum loop.

The targeted sound file is `Drum_Loop1.aif` *and the factorized vectors are audible in* `drumLoop_1.wav` , `drumLoop_2.wav`, `drumLoop_3.wav`, *and* `drumLoop_4.wav` *where the index numbers represent the corresponding vector indices.*

In the next example, we used a flute fragment of (`flute-fragment.aif`). The FFT analysis is operated under the same condition to the previous test, but this time, the NMF decomposed the targeted signals into six vectors. The result is shown in figure 52 and is audible in (`flute_1.wav`) to (`flute_6.wav`). The flute fragment consists of two contrasting sections; a fast descending gesture involves with strong articulations and a sustained and stable tone, this phrase is played twice. The descending gesture is decomposed more finely into four vectors than the sustained tone into two vectors. It is presumed that the fast descending section has more pitch variations which yields different spectrum envelope than the sustained tone. Therefore, each sound event is separated into its respective vectors. We evaluated the quality of the decomposition by mixing all resultant vectors into an audio data of (`flute_NMF_Mix.wav`). The mixed vectors represent sufficiently the original audio data in its timbre, dynamics, and pitch information. With a small number of vectors, the mix of the resultant vectors represent the rough characteristics of the original signals, and the quality of the descending gesture is significantly reduced as we can hear in (`flute_NMF_2Vector_Mix.wav`). This decomposition process can be applied for the synthesis process in which each vector is independently reconstructed by sound sources retrieved from a database, and subsequently, all synthesis results are mixed. Although an insufficient vector number does not

produce an ideal decomposition result, it gives some degree of freedom for controlling the audio representation.

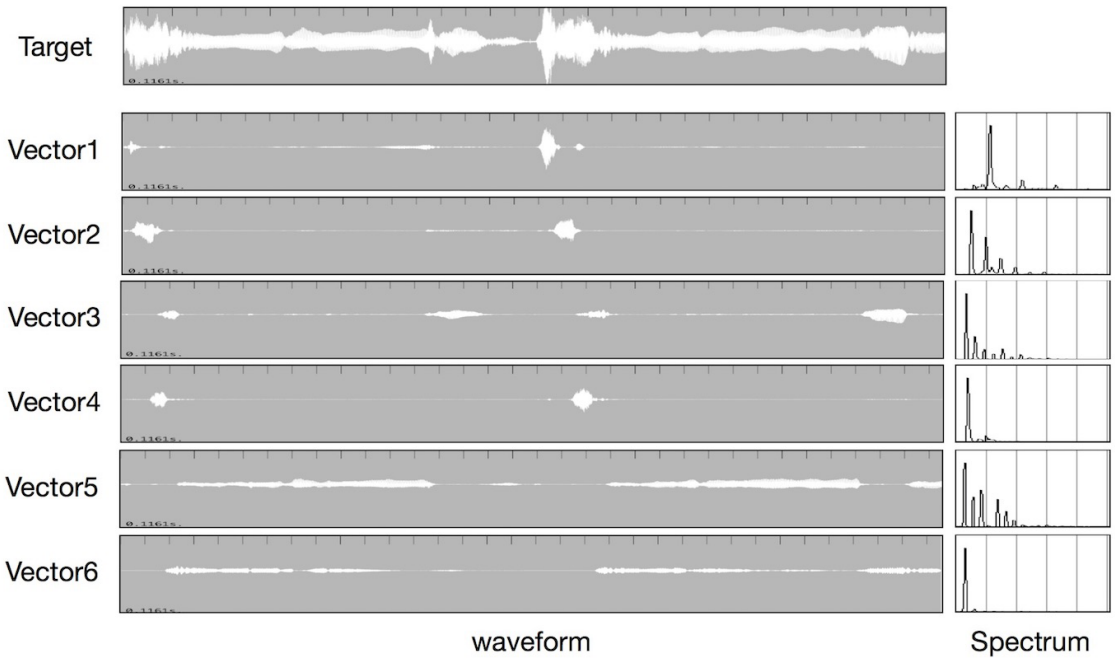


Figure 52: The NMF result of a flute fragment.

The targeted sound file is `flute-fragment.aif` *and the factorized vectors are audible in* `flute_1.wav` , `flute_2.wav`, `flute_3.wav`, `flute_4.wav`, `flute_5.wav`, *and* `flute_6.wav` *where the index numbers represent the corresponding vector indices.*

12.3 Discussion

The NMF is one of the most important algorithms for our synthesis program in order to represent the polyphonic structure of a sound. The NMF we have introduced here operates unsupervised statistical optimization which decomposes audio signals mechanically into the best-compromised combinations of the frequency and temporal information. Due to the unsupervised optimization, the result is significantly influenced by the initially defined random numbers. Due to that, this approach may cause unintended results because the algorithm is based on a mathematical analysis without using any perceptual models[49].

There is also the supervised-based NMF. For instance, Nakashika et al. employed a common spectral envelop of a particular instruments[49]. Nakashika collected variable isolated sound data of a particular instrument, for instance, piano, played in different pitches individually. The collected sound data are analyzed in order to extract their

spectral envelopes and those of which are trained in the learning process in order to produce a probability distribution of the spectral envelope. The distribution is utilized for initializing vector matrices instead of using random numbers, and the most appropriate envelope is selected according to the targeted sound quality. M. N. Schmidt developed a speech separation algorithm using a training data of a speech corpus[48]. These speech data are analyzed into their phonemes and are employed to separate speech signals into their components. These methods are expected to produce better results as the vectors initialized with a particular spectral models, and which is somehow similar to what we have discussed in the specmurt analysis using the optionally defined overtone patterns.

The log-scaled frequency spectrogram is also interesting approach especially for tracking the pitch variant musical gestures such as glissando and chromatic movement. The log-frequency based NMF algorithm is studied by E. Benetos and S. Dixon[47] where the pitch templates of an instrument is prepared for initializing the vectors and the templates are shifted in order to adopt the pitch of the targeted sound components.

The conventional NMF algorithm requires a vector number as a parameter, and we need to estimate the correct number of sources consisting the targeted audio signals in order to obtain precise decomposition results, which is not ideal for a practical system. Hoffman et al. and Kameoka et al. developed the Bayesian Nonparametric Matrix Factorization, which is operated for blind source separation where the number of sources is unknown. Since these methods employ iterative estimation methods in order to estimate the vector number, it demands very expensive computation. Although we consider that the vector number is one of the most significant parameters which enable us to control decomposition process creatively for the subsequent synthesis process, the automatic source number estimation is also beneficial to improve the quality of the decomposition result and to enhance the usability of the system.

As we have already mentioned, audio decomposition is one of the most difficult and complicated subjects in signal processing. There are various mathematical models which can be applied for the decomposition methods, but further study is required to realize a practical and versatile system. Indeed, it is important to develop a mathematical model which produces physically precise result, however we rather focus on developing a program which provokes us to create a new music sources. The decomposition task is significantly useful for our synthesis process. Because the targeted sound often consists of complex sound components, and it is difficult to find an adequate matching sound

source from a database in order to reconstruct it. The decomposition makes it much easier to do so because the decomposed components are more simple and isolated.

Future work includes more robust and versatile decomposition algorithms which cover variable creative purposes. The combination of the NMF and specmurt analysis are also to be considered. Once the spectrum is transformed into the specmurt domain, only the spectral structure is considered because the pitch information is hidden. Therefore, it is capable of grouping the same sound qualities into the same vector regardless its fundamental frequency. It is an advanced version of the specmurt analysis which represents also the time-varying components. It is also beneficial to pre-segment the audio signals by means of previously discussed segmentation algorithms before operating the NMF based decomposition. Since the NMF utilizes statistical analysis, the targeted audio signals are better segmented into more isolated units in order to reduce the complexity of the sound quality, and which makes it easier to find the best spectral profiles.

Part V
Classification

13 Introduction

The classification is a data mining technique which automatically assigns an individual unit of audio data to a number of categories or classes according to its descriptive features extracted by the FFT-based analysis. This process aims to extract common characteristics from a data set and sort them in a straightforward and concise format. The classification is one of the most essential operations to increase the efficiency of Pattern matching and the Music Information Retrieval (MIR).

Firstly, we define several related terms to avoid confusions and misunderstanding as some have almost the same meanings but slightly different concepts in some particular contexts. For instance, three terms, Pattern matching, finding the Best Matching Unit (BMU), and the MIR are the almost equivalent in our synthesis process. These words indicate tasks to search a unit or a unit sequence, which has the most similar structure or characters to that of a target, in a data set. Accordingly, we describe the pattern matching as a process to find the BMU retrieved from a database, which corresponds to the MIR. The BMU is selected by calculating dissimilarity between a candidate unit and a corresponding target unit. Similarly, the classification operates a categorization, and its consequence is designated as classification model, feature space or learning map. The classification model indicates simply the result of this process and other equivalent words, the feature space represents a classified features by audio analysis which is often used in the audio analysis or electronic music, and the learning map focuses on the machine learning or neural network.

The classification enhances the efficiency of pattern matching by building a model or learning space of characteristic patterns or structures of a data set and sorting data in a particular order. This process avoids searching superfluous area of a data set, and as a result, it reduces a computation time significantly. Also the classification model can illustrate a distribution of a data set, and it gives a weight of each member of datum which represents a degree of the presence of its data property. This aspect of the model becomes a criterion for evaluating whether a datum belongs to a group of majority or minority of a set. When it belongs to the majority group, we may have more candidates

to be matched to the target unit and have a higher expectation to obtain a good results than the one assigned to a minority group. Therefore, this criterion is appropriate to measure the adequateness of the matching between the selected target and database.

The classification model and the above-introduced criterion are also available to compare the patterns of units sequences. When the successive units are sorted in the same or similar categories, the categorization result suggests that no significant audio novelties occurred in the sequence. On the other hand, when units are assigned to diverse categories, we can expect a wide variety of sound events in the sequence. A chain of categories itself becomes a criterion to identify characteristics of a sequence of units which is known as the Markov Chain. We can find the Best Matching Sequence by calculating the distances between chains of categories to which each constituent units belong.

There are various ways to operate the classification to build a learning space. We will here discuss the Self-Organizing Map (SOM) which is a neural network employing unsupervised learning introduced by Kohonen in 1981. Although the SOM is one of a primitive neural network under existing algorithms, it has been actively studied and extended for long years due to its straightforward structure and easy implementability. There are many studies proving its usability and efficiency particularly for building a small scale model[54]. The SOM we have developed is capable of operating the data retrieval task on the main memory of the computer, which is called In-Memory Database or Memory resident database system, and it realizes fast database management. The speedy data retrieval is one of the most significant issue for our synthesis system, since it requires quick and easy access to the large amount of data.

The SOM based classification takes three steps; firstly, we operate a learning process in order to build a classification model or a learning map of a data set. The model illustrates the typical characteristic patterns of units in a set and their topological mapping. Secondly, each unit of sources is categorized into a model according to its descriptive features. A result obtained by the categorization illustrates a distribution of the source units and gives a relative weight to each unit which represents a (dis)similarity to its assigned category. Finally, we apply the model for associating the characteristic pattern of a target with sources. Throughout the process, the target units are labeled category index and the weights which represents a distance between the unit and its categorized characteristics pattern of a data set.

The Labeling re-identifies the characteristics of target units by lower dimensional criteria which are the coordinates of the learning space as shown in figure 53. The original features are preserved and are employed to evaluate the detail characters. Since the low dimensional criteria are relative to the character of the data set, the accuracy of the labeling becomes rough or rigid depending on the distance between a target and a source data set. For instance, the Labeling result is more reliable when a target has similar characteristics to a mass of the sources in a data set, and vice versa. In any case, the criteria are specialized for the sources but not for the target and thus, the labeling method is not the best efficient way to realize theoretically precise identification to the target, but it is valid for our reconstruction process. This is to be discussed profoundly later in the Labeling section.

The property of the classification is significantly influenced by the employed features and in other words, the selected features determine how the database is categorized. For instance, the magnitude feature is employed to categorize a database into distinct dynamics and the Mel-Frequency Cepstrum Coefficients (MFCCs)[10] is for timbres. When we combine multiple features, the database is categorized into complex characteristics. The smaller number of features tends to produce higher reliable classification model, and in contrast, those of the larger number reduces its reliability due to the curse of multidimensionality as we can see in the poor generalization[11]. According to the Neural network Design[59], for the network to be able to generalize, it should have fewer features than there are data points in the training set (a concept advocated by Ockhams Razor). For our case, a more versatile network is desirable for our synthesis, and the design of the highly generalized network is the highly prioritized task.

14 Feature Vectors and Feature Space

As discussed in the Audio Analysis section, we employ several descriptive features in order to identify the characteristics of audio data. In the classification, we represent a set of features as a vector in order to simplify the further processes. The n-dimensional features are represented by a vector X as follows,

[10] MFCCs discussed in the Audio Analysis section.

[11] The generalization problem is also related to the number of training data. More features demand more training data to have a well-trained network and otherwise, the resultant network can not adequately represent the training set and thus, it is less reliable.

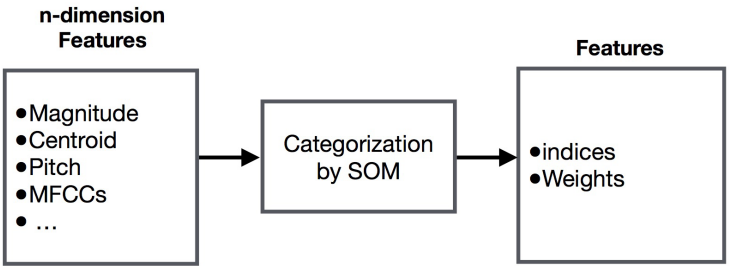


Figure 53: Reduction in criteria by categorization.

The original features consists of several criteria (left). The categorization process reduces the number of criteria into two or three features (right).

$$X = (x_1, x_2, \ldots, x_n)^t \tag{61}$$

where t means transposition of the matrix[60]. Considering a n-dimensional feature space which shows a distribution of the feature vectors, each vector is located at a certain position according to the member values, and it is represented by a dot in the space. A distance between two vectors corresponds to a distance between two dots, and it can be calculated by Euclidean distance or inner product. Figure 54 illustrates a n-dimensional feature space and two distinct vectors X and X'. A dotted line with D shows the distance between them.

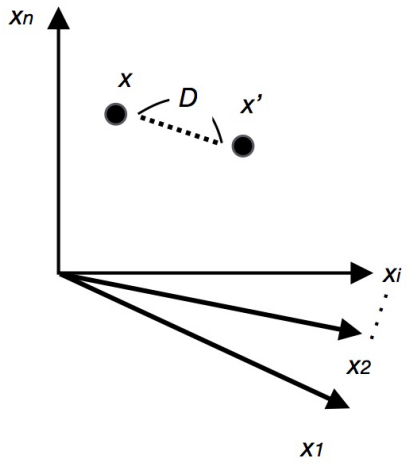


Figure 54: n-dimensional feature space

The vector representation is useful to apply to the descriptive features because of

its capability of handling a data chain. In the case of the audio analysis, each member of a vector corresponds to distinct features such as Magnitude, Centroid, Pitch, and MFCCs. The number of features and their combinations vary depending on the purpose, e.g., what characteristics to describe. The feature space represents the distribution of the audio data according to the employed features, and thus, the distance between two vectors on the space represents the (dis)similarity of two audio units.

There are two problems to be addressed regarding the application of the feature space in the audio analysis.

The first problem is that the features are not described in the same measurement unit but are represented in various scales which have different ranges of values[61]. For instance, the linear magnitude is represented by a linear scale of 16bit data which is variable between 0 to 65,536. The centroid and pitch are represented by frequency (Hz) between 0 to ca. 22kHz at the sampling rate of 44.1kHz. The ranges of values are not coherent among distinct features, and thus, the distance of each feature is not compatible and has different property. Furthermore, the human auditory system does not perceive sounds in the linear scale but the exponential scales such as decibels and pitches which is generally represented as MIDI-pitch numbers. The different scales vary the balance of a feature space, and as a consequence, they change the distances between vectors. While analyzing the features, we cannot assume that all of them have the same weight in the perception. This problem prevents building a reliable pattern model illustrating the static relationships between vectors.

Rescaling all features to a straightforward value range is one of the easiest solutions to organize a static feature space. For example, Normalization (section 16.1) is capable of simplifying values and typically, it rescales any values between 0 to 1. This method is beneficial when rescaling an exclusive feature or sets of values represented in the same measurement unit. However, when applying it to feature vectors, it can change the original weights among distinct features because the normalization ignores the distribution of entire values in a feature space, and it homogenizes them regardless of their actual weights. In contrast, the Standardization rescales all member features in a coherent scale while preserving their wights in a space which will be introduced in the section 16.3. One smart solution focuses on calculating the distance between two vectors without rescaling values (section 16.2). This way, we can ignore the distribution weights of features because the distance is represented by a ratio between two vectors

as long as representing them in the linear scale.

Another problem is considered when a dataset has a strong bias in its features. For instance, when a dataset consists of audio data which have very low magnitudes and in contrast, a rich variety of pitches, the resultant space has a large distribution in pitch and a small one in magnitude. However, this distribution bias does not tell us why it occurs e.g., whether it comes from the difference between measurement scales or merely from the characteristics of the dataset. In the former case, we need to adjust the weights to the appropriate balance, but the latter, we should evaluate as it is. The two features; dynamics and brightness, are relatively perceptible by our audible system which enable us to estimate the regular value ranges. Therefore, we can conclude that the result shows low magnitude trend and can give some parameters to adjust the distribution balance manually. However, when handling abstract features such as spectral flatness and MFCCs, it is hardly possible to estimate how each feature take values under particular conditions. Especially, the MFCCs are multidimensional in themselves, since each MF-band could be considered as a component of a vector, and each coefficient may take different value range. The lower coefficients tend to take much larger values than those of higher because they hold more crucial properties determining the spectral envelope. However, we cannot define what extent they are significant compared to others.

Some studies related to the pattern matching of pictures and speeches apply the standardization in order to adjust the balance of significance among distant features according to deviations of the features[60]. Before discussing these rescaling algorithms, we will show the basic algorithm of the Self-Organizing Map (SOM) and how the feature vectors are employed there.

15 Self-Organizing Map

Self-Organizing Map (SOM) or Kohonen network is an artificial neural network which performs classification where It clusters the corpus data into distinct groups. Sine the SOM is introduced by Kohonen in 1981[50], various advanced algorithms have been studied[54]. The algorithm is inspired by the structure of the human cerebral cortex which manages recognition and perception. The SOM operates unsupervised learning to represent the topological properties of the input multi-dimensional data series into

two or three-dimensional feature maps after a training process.

The SOM has two phases; the first is the learning of the map using the training dataset and the second is the prediction in which the new data are given their location on the converged map according to their vectors. Through the entire process, the SOM performs in the basic unit of a neuron or node which is characterized by a vector consisting of the features equivalent to that of the input dataset. A vector of a node is adjusted throughout the learning process, and conclusively it represents one of a characteristic pattern of a training dataset. Therefore, the output map illustrates a list of the statistic pattern of the dataset.

Now we will shortly describe some steps of the learning. Initially, the output map is filled up with random numbers, snd subsequently, a training dataset is selected from the input dataset and presented on the map[12]. The node which has the most similar vector is called the Best Matching Unit (BMU), and the updating weight is propagated around the BMU towards neighboring neurons. The bigger weight of modification is given to the nodes close to the BMU and vice versa. There are some other parameters such as the update radius which determines the area propagating the update weights, and the update ratio constrains the extent of the modification. This learning process is repeated many times, and as a result, the nodes of a map gradually get closer to the statistic pattern of the training dataset and thus, the updating weight declines. The iteration ends when the weight becomes sufficiently small.

After the learning process, we obtain a converged map which represents relative distance in the space of the training dataset. Units of whole input dataset are categorized on the map where each unit is given a pair of x-y coordinates on the map, and therefore, they are clustered according to their vectors. Units with similar characteristics are positioned in the same or close place on the map.

Figure 55 shows the categorization process using the SOM. We first divide a database into two groups of data; the training data and others. The training data is separated when using a large size of database and otherwise, all input data is used for the training. When we inspect the quality of the learning map, we operate a test and then, we also obtain the third group which is the test data. The training data and the test data need to be separated in order to obtain a precise test result.

[12]The input dataset is divided into a training and test datasets in order to inspect the learning quality which will be discussed in the further section.

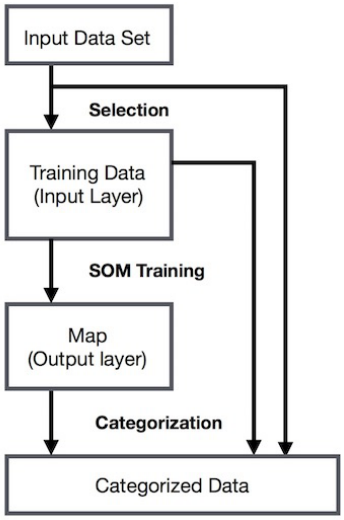


Figure 55: The categorization process using the Self-Organizing Map

The input data set (database) is divided into training data and others, or all input data are used as training data. The training data, which corresponds to the input layer, is trained and results in a map corresponding to the output layer. Eventually, both the training data and others are categorized according to the obtained map, and hence we obtain the categorized data set.

15.1 Kohonen's learning law

We will describe the basic algorithm of the Kohonen's self-organizing map according to [53].

Let one-dimensional network of the Kohonen self-organizing map $(w_1, w_2, \ldots, w_m)$, $m \in \mathbb{R}$ and an input vector space $v_1, v_2, \ldots, v_n$.

We define an initial update radius r which constrains how furthest nodes the update weights propagate from the BMU, a learning constant η, and a neighborhood function ϕ. The neighborhood function $\phi(i, k)$, where i is the index of the input vector, and k is the index of the BMU, represents the strength of the coupling between unit i and k during the training process.

We select an input vector ξ from the input space and find the BMU w_i from the map in which the distance between w_i and ξ is minimal and thus, $i \leftarrow \arg\min_i |w_i - \xi|$. The each weight vector of the map w is updated using the neighborhood function and the update rule as follows.

$$w_i = w_i + \eta\phi(i, k)(\xi - w_i) \quad (1 \leq i \leq m) \tag{62}$$

This process is repeated until the number of iterations has been reached to the initially defined iteration time. Figure 56 illustrates an initialized map and the trained map where each node has 8-bit data which is 0 to 255 values (0 is black, and 255 is white) as a feature. After an iterative learning, random grayscale colors are organized according to its color information, and the result appears as a gradation which interpolates white and black colors.

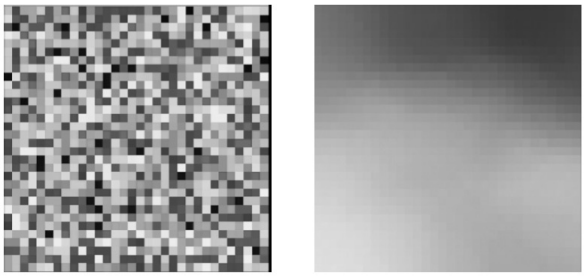


Figure 56: The result of the SOM.

An initialized map (left) with random values between 0 to 255 (8 bit) and a trained map (right) where the random grayscale is organized in the order from dark to bright.

We give a demonstration of the SOM using simple pictures of various shapes as shown in figure 57. The picture consists of approx. 320x320pixels and each pixel has 8bit data size (0 to 256) which represented a grayscale color. In order to extract feature vectors, the pictures are compressed into 16x16 pixels, and each of them has 256 data series. We utilized a two-dimensional SOM with 64 number of nodes which is eight nodes per side, and each of them has 256 values. The program operates an unsupervised learning thereby organizing the pictures in distinct regions in which similar shapes are adjacent to each other. Figure 58 shows the resultant map visualizing its nodes as grayscale pictures. We can observe topological relationships and their shape gradations between shapes. The horizontal line of the map is represented as $(x_1, x_2, \ldots, x_8)$ and the vertical line is $(y_1, y_2, \ldots, y_8)$ and thus, a node is identified as (x_i, y_j).

15.2 Categorization

After the map is trained, each picture is stored in the node which has the closest feature vector, and the node is called BMU for the picture. The BMU is found by calculating the distance between the vector weights of the input picture and the vector weights of each node. This distance is calculated in an assumed Euclidean Space in which

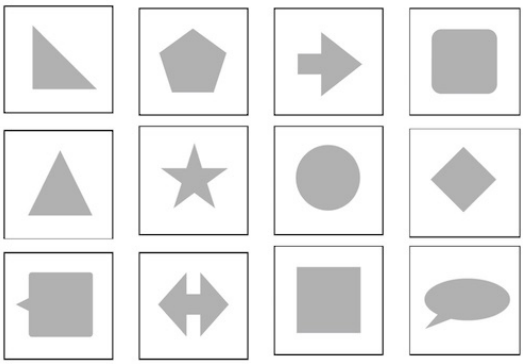


Figure 57: Pictures of the Twelve Shapes

it is possible to map or calculate multi-dimensional information into a single vector value based on the Pythagorean theorem. For instance, a distance of two dimensional coordinates (x_1, y_1) and (x_1, y_2) is calculated by the following equation,

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{63}$$

With the above equation, the distance between between n-dimensional coordinates $f(t) = (x_1, y_1, z_1, \dots), g(t) = (x_2, y_2, z_2, \dots)$ is calculated by follows,

$$d_n = \sqrt{\sum_{1}^{n} (f(t) - g(t))^2} \tag{64}$$

The node whose distance d_n to the input vector is smallest becomes the BMU. Through this process, similar pictures are stored in the same or in adjacent nodes. For instance, categorizing some new input pictures which do not belong to the trading data, may be assigned to nodes which are not sufficiently similar but the nearest on the map. This matching method finding the most similar unit is called as Nearest Neighbor algorithm (NN algorithm). The quality of the categorization is significantly reduced when using a deficit number of training data, or when training data have poor variations. A larger number of training data is capable of build more relatable learning map, and vice versa[61].

The table 1 shows the result of the categorization and counts of the input data each node has. The 18,333 numbers of randomly generated pictures are used, and the average of counts in each node is 286.

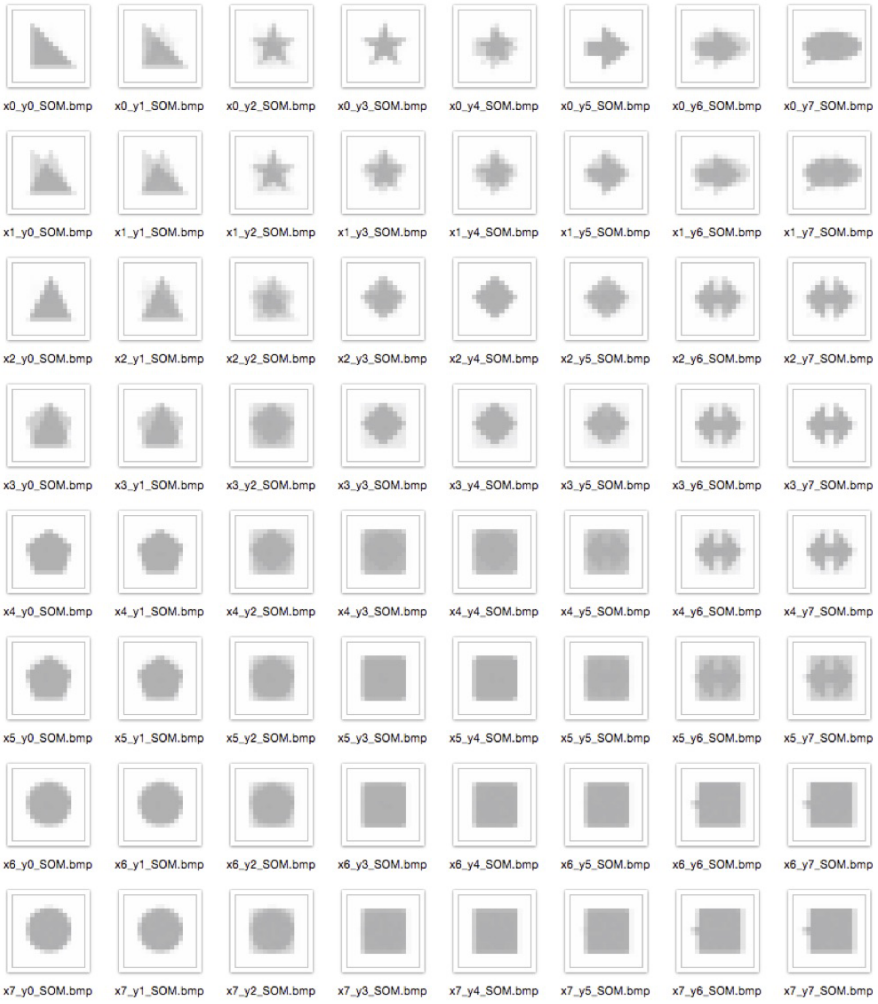


Figure 58: Trained map consists of 8x8 nodes

Figure shows the trained map of the shape pictures shown in figure 57. We can see how adjacent nodes are related to the original pictures. Between the nodes which illustrate nearly the original pictures, there are some other nodes which interpolate these topological characters appearing as gradations.

Table 3: Data distribution.

(x_i, y_j)	1	2	3	4	5	6	7	8
1	200	142	87	175	216	156	478	503
2	15	327	156	289	109	78	407	69
3	482	667	262	731	414	561	191	328
4	288	436	250	385	449	191	331	121
5	281	720	347	80	86	408	222	557
6	119	269	1335	194	111	199	95	117
7	160	449	122	279	70	132	118	254
8	321	96	585	360	136	25	310	282

The table shows the distribution of the input data in the categorization result. The vertical line represents x-coordinate of the map, and the horizontal line represents that of y-coordinate. The number represents the number of data assigned to the corresponding node.

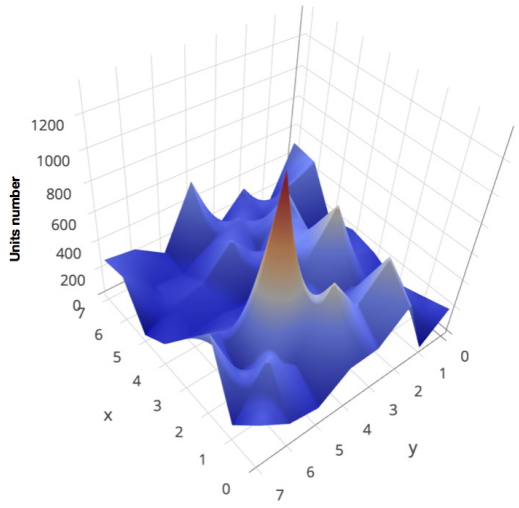


Figure 59: Data distribution in 3D graph

15.3 Self-Organizing Map for Audio data

In the previous example using picture data, data of a small area of a picture is stored in a node. In the case of audio data, a sound file is divided into multiple smaller units which are then individually stored in different coordinates of the map corresponding to their characteristics. The calculation method for measuring the distance between and unit and a node is also different. In the case of the picture, the Euclidean distance was calculated with subtraction. All feature vectors of picture data represent the same property in 8bit data. In the case of audio data, however, each criteria represents totally different perceptual aspects of sound data as discussed in the feature vector

section. There are several methods to solve this problem as we will introduce in the next sections.

16 Scaling

16.1 Normalization

Normalization is a way to adjust values measured on different scales to a common scale. A feature vector which identifies a single frame, consists of a set of features and each of them is measured in different units. The normalization transforms these values into equally compatible format whose range is between 0 to 1. The most simple normalization is defined as follows.

Considering a set of feature $x = (x_1, x_2, \ldots, x_i)$, where i denotes a frame index, the normalization of each value is calculated by the following formula.

$$z(i) = \frac{x(i) - min(x)}{max(x) - min(x)} \tag{65}$$

The normalization scales a data set in simple and concise representation and it is particularly beneficial when building a model for an exclusive feature set. indeed, the normalization is applied for some analysis algorithms we have already discussed. For instance, the Specmurt analysis employs a sigmoid function which is the nonlinear mapping function and where the normalized input values are employed in order to assess the suppressed weights efficiently throughout the iterative estimation. The Non-Negative Matrix Factorization also uses normalized parameters when calculating the Auxiliary function in the optimization process.

While the normalization is useful method to simplify some processes, it is problematic when comparing multiple feature sets. The reason is that the scale is operated to a data set without considering its variance. The variance is one of the most significant criteria to know the character of the data set. When the variance is small, the data members are close together and they are similar to each other and vice versa. This principle is not related to the measurement unit but to the statistic character of the data set.

For instance, defining two feature sets, $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$, and assuming that one has small variance and other has large variance. In the normalization, both feature sets are scaled equally from 0 to 1 which means, it enlarges ones weights and reduces others. Let normalized vectors $A' = (a'_1, a'_2, \ldots, a'_n)$ and

$B' = (b'_1, b'_2, \ldots, b'_n)$. The distance between two vectors $v_1 = (a'_i, b'_i)^t$ and $v_2 = (a'_j, b'_j)^t$ can be calculated by Euclidean distance as $D = \sqrt{(a'_i - a'_j)^2 + (b'_i - b'_j)^2}$. Although the members of A' and B' are assumed to be compatible with each other, they are actually scaled arbitrarily from 0 to 1 and thus, the original weights are lost. The obtained distance does not represent the appropriate result. The error is worse when the variations of two data sets are more different, and smaller when they are closer.

16.2 Distance measurement

The above discussed problem can be solved by applying some methods of distance measurement. One solution will be calculating the Dissimilarity ratio between two vectors. The dissimilarity ratio is calculated by dividing corresponding features as follows.

Let the original feature set $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$, the dissimilarity ratio between two vectors $v_1 = (a_i, b_i)^t$ and $v_2 = (a_j, b_j)^t$ is

$$D_{ratio} = \frac{min(a_i, a_j)}{max(a_i, a_j)} + \frac{min(b_i, b_j)}{max(b_i, b_j)} \quad (min(a_i, a_j) > 0), (min(b_i, b_j) > 0) \tag{66}$$

When both of corresponding members are 0, then the result is 1 which means the two vectors are the same and when the result gets close to 0, the difference between two vectors is large. One restriction of using this method is that it requires non-negative inputed values.

Cosine Distance or Similarity is also the one which measures distance between distinct vectors. It is also called as dot product. We have already introduced this method when calculating the Self-Similarity Matrix. The Cosine Distance is calculated by the following formula.

$$D_{cos} = cos(\theta) = \frac{A \cdot B}{\|A\|^2 \|B\|^2} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n}} B_i^2} \tag{67}$$

The resultant distance of -1 meaning exactly opposite correlation, a distance 1 meaning exactly the same, and a distance of 0 meaning total decorrelation.

16.3 Standardization

Let a set of N number of values $x : N \in \mathbb{R}$, its mean which we denote $\bar{x}$ is

$$\bar{x} = \frac{\sum_{n=1}^{N} x(n)}{N} \tag{68}$$

The variance of a set of values which is represented as σ^2 is defined as

$$\sigma^2 = \frac{\sum_{n=1}^{N} (x(n) - \bar{x})^2}{N} \tag{69}$$

A root of the variance gives a Standard Deviation σ which is a measure of how the member of the data set spread out as follows.

$$\sigma = \frac{\sqrt{\sum_{n=1}^{N} (x(n) - \bar{x})^2}}{N} \tag{70}$$

Eventually, each member of the data set is standardized by the following formula.

$$z(n) = \frac{x(n) - \bar{x}}{\sigma} \tag{71}$$

The standardized data set Z has its mean of 0 and distribution of 1 which is known as the standard normal distribution. The standard deviation lets us know the character of a data set since the variance suggests the datas spread size compared to the mean value. The variance is bigger when the data set has wider variations where the member data have bigger distances to each other. On the other hand, when less variation occurs, they have smaller variances and hence smaller distances among them. The standardized values indicate measures of the dispersion from their mean and we can predict from the weights of each values whether it belongs to the majority or not. For instance, we applied standardization in order to build the universal segmentation model which detects audio novelties as discussed in the Segmentation section. In the segmentation algorithm, it is more favorable to detect the exclusive peaks to avoid noisy peaks.

Figures 61 and 62 show the distribution of the feature vectors of a drum loop (`Drum_Loop1.aif`) and a flute fragment (`flute-fragment.aif`). Each dot represents a vector of a frame which consists of two features; Magnitude and Centroid. The thicker lines indicate the averages of corresponding features. As shown in the original data distribution, the vectors of the drum loop sound are distributed over broader frequency bands as it consists of the low sounds produced by bass drum and the high sounds by hi-hats as well as the broader range of dynamics compared with the flute fragment sound. The difference between the characteristics of two sounds is apparent in their

original data distributions. Note that the measurements of both Magnitude and Centroid are defined in order to capture all vectors, but they are not scaled to represent any relationships between two features. Although we can see the Magnitude vectors spreading out in twice broader than that of the Centroid in the flute fragment, it does not describe that the Magnitude is more variable than the Centroid. Similarly, the normalized data distributions, which are shown in the middle line in figure, are the result of mechanically rescaling all features from 0 to 1 and thus, they have the same distance in appearance. However, this method loses the actual proportions of the Magnitude to Centroid, and therefore, the normalized values are not adequate to represent the real distances of vectors.

In contrast, the standardization rescales a data set according to how the members are far from its mean and therefore, it is capable of preserving the actual proportion of distinct features[60][61]. Observing the original data distribution, the Centroid of the drum loop equally spreads out over broader frequency bands, while the majority of the Magnitude concentrate on lower dynamics area hence it has a strong bias. The mean of the Centroid is located at the position close to the middle of entire data, but that of the Magnitude is located at a much lower position. Since the vectors in the higher dynamics area are sparse whose number ratio corresponds to only 2% of the population, we can assume that these vectors are categorized in the irregular members. The normalization tends to represent the majority dataset with smaller distances as the irregular members farther from the mean. This aspect loses the actual weights of distance between data. On the other hand, the standardized data distribution shows a higher range in the Magnitude than in the Centroid since the Magnitude has broader distribution. Therefore, it preserves the actual proportions in both the majority and the minority groups and as well as those of the two features. A degree of how a feature spreads can be measured by the Coefficient of variation (CV) which is calculated by dividing standard deviation by mean[62]. Since both parameters, standard deviation and mean are calculated in the same measurement unit, the obtained CV shows the same values regardless of their value ranges. Therefore, we can know how the values actually spread out between distinct features.

Figure 60 illustrates the typical case of how irregular data gives a detrimental effect on the majority data set in the normalization. Figure shows two data sets of the original and the normalized data; one is without any irregular data (left), and another is with a

few irregular data (right). The arrows indicate the distribution ranges of the majority data sets. While the original data sets have absolute distances between data no matter how they contain the irregular datasets, the normalized one has a rescaled distance where the majority datasets are significantly compressed in small areas when having some irregulars. This example shows the case of only one feature having irregular data, but imagine several features having unpredictable number of irregulars. The obtained feature space gives unnecessarily large weights on the distances between the irregulars and the majorities, and it cannot represent adequate ranges in the majority of the data.

As shown in figures 61 and 62, the CVs of the drum loop sound are the Magnitude CV = 1.1313472, and the Centroid CV = 0.3161678 which show the high variations of the Magnitude and the low of the Centroid in their distributions. Indeed, many vectors take much higher values in the Magnitude than the average, and in contrast, those of the Centroid concentrate around the average. The normalization modifies the actual weights between features. While the Centroid values preserve relatively the same weights to the original, the majority of the Magnitude is significantly compressed in much lower weights. Therefore, the distance of the Magnitude between vectors are now underestimated, and it brings a bias in the feature space. In the standardized data distribution (left), both features are now rescaled according to their distributions and the feature space is coordinated properly as only majorities fit into an area enclosed by a regular square. Note that the graph takes now the same value ranges in its vertical and horizontal axis which are minus three to six.

Comparing with the drum loop example, the features of the flute fragment spread out similarly which are a consequence of the moderate CVs. In that case, the normalized data and standardized data does not differ dramatically. That is the reason why the flute sound does not have high variations in its timbre and dynamics in the phrase. The CV feature is an useful criterion to assess the bias of a feature space and therefore, this feature informs us of the characteristics of the dataset.

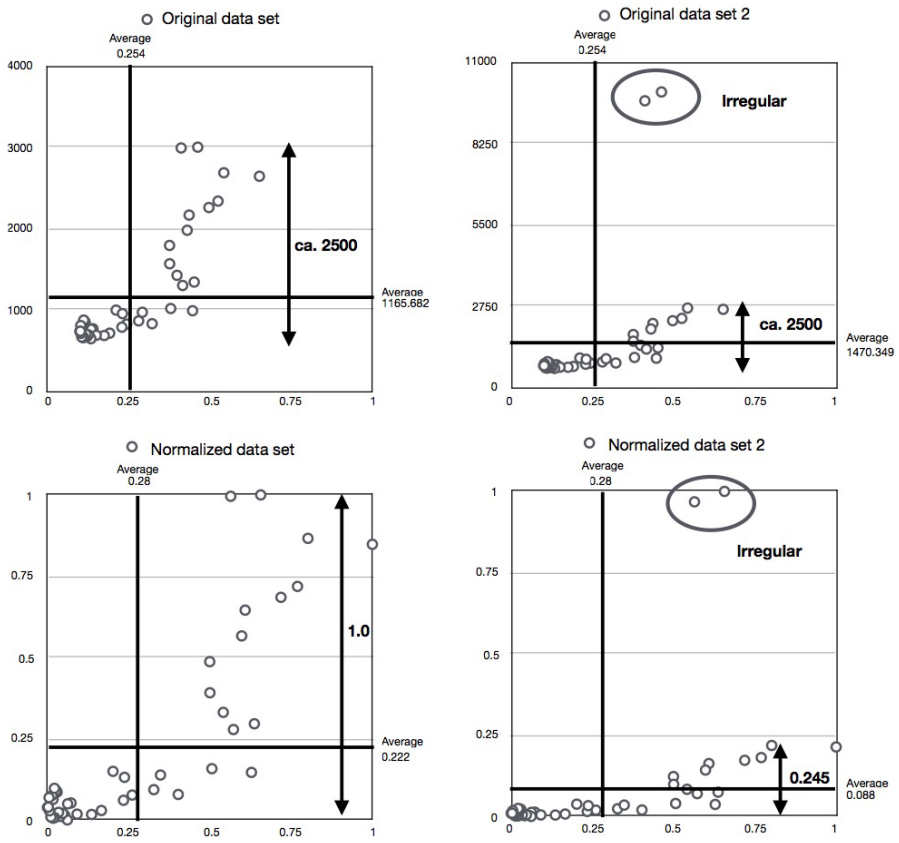


Figure 60: The detrimental effect of the irregular data.

The data distributions of the original datasets (above) and the normalized datasets (bottom). The sets without irregular data (left) and the one with irregular data (right). The circles with a thicker line mark the irregular datasets. The distributions of the majorities are indicated by arrows with the values which represent its ranges in the scales.

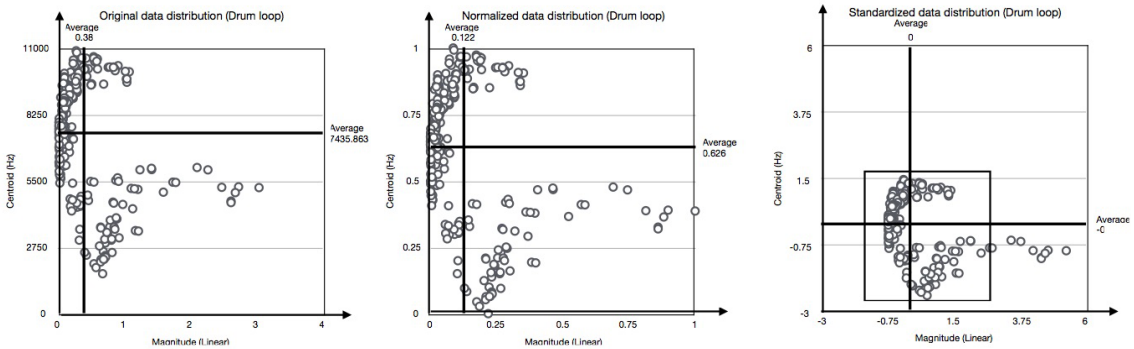


Figure 61: Drum loop sound : Comparison between the original (left), normalized (middle), and standardized data distribution (right) of two features; Magnitude and Centroid.The coefficient of Variation; Magnitude = 1.313472 and Centroid = 0.3161678

The analysis condition is the FFT frame size of 1024, hopsize of 512. The drum loop audio data has 302 frames. The graphs illustrate the distribution of frames in two features; the Magnitude (horizontal line) and Centroid (vertical line), and their vector is shown as a dot on the feature space. The top figures illustrate the original data distribution, and the middle illustrates the normalized data, the bottom illustrates the standardized data respectively.

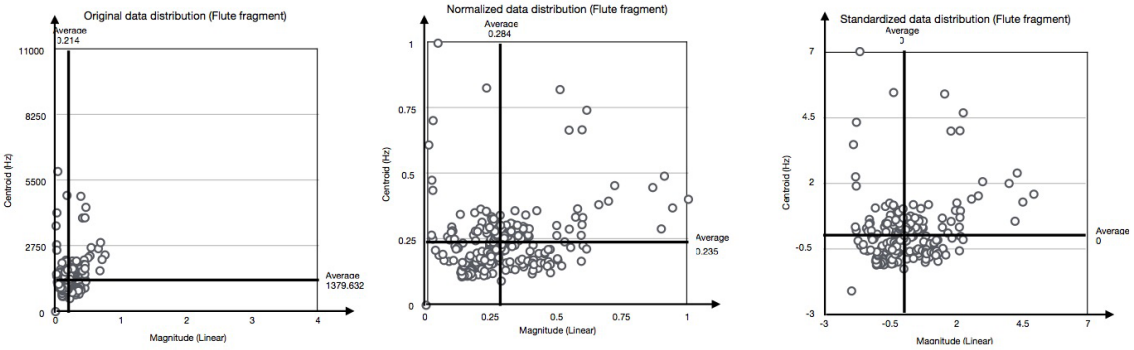


Figure 62: Flute fragment : Comparison between the original (left), normalized (middle), and standardized data distribution (right) of two features; Magnitude and Centroid. The coefficient of Variation; Magnitude = 0.507432 and Centroid = 0.466657

The analysis condition is the FFT frame size of 1024, hopsize of 2. The flute fragment audio data has 343 frames.

16.4 Categorization process

16.4.1 Self-Organizing Map with MFCCs

We give a demonstration of building a classification model by using the Self-Organizing Map. Here, we will use MFCCs as the feature vector.

In the audio analysis, we extracted 27 MF-bands and employed the first thirteen

MFCCs. The first coefficient is discarded as it represents the entire power of the spectrum and therefore, the 2nd to 13th coefficients are ultimately preserved. We defined the number of coefficients in an adequate compromise between a good representation of the spectrum envelope and a decline of the impact of the multi-dimensionality problems for the feature vector. Although it is meaningful to employ other features such as Magnitude, Centroid, etc., we focus on only MFCCs in order to simplify the process.

We have prepared three kinds of databases for the demonstration as follows. The piano database which focuses on similar sound qualities but a vast range of register. The percussion database containing various percussion sounds with or without a sense of pitch. The Violoncello database containing the Arco, Pizzicato, and various extended techniques, which focuses on similar sound qualities with rich variations. Secondly, we operate the spectral analysis by using FFT under the condition of a frame size of 1024 and the hopsize of the FFT half size. The detailed properties of the database and analysis are shown in the table 16.4.1.

Table 4: Database Property

Database	Files	Data size	FFT frames	Segments	FFT size	Overlap
Violoncello	2418	2.93GB	1411500	190759	1024	2
Percussion	689	0.45GB	183395	19697	1024	2
Piano	260	1.5GB	690870	91933	1024	2

The table shows the contents of a database, the number of files, its data size, the number of the obtained FFT frames, segments obtained by the segmentation algorithms using Delta-MFCCs, and the analysis FFT size, and overlap number respectively.

After operating the analysis, each MFCC is standardized according to its deviation and mean which are then utilized to standardize new input units or target units in order to unify the measurement and preserve the compatibility between target and sources. The input units are rescaled regardless of their deviations but by those of the sources.

Figure 63 illustrates the structures of the classification and the MIR in the synthesis process. The standardized source dataset is divided into three sets; a training dataset, a test dataset, and others. All datasets are eventually categorized into the model after the training process is complete and the learning test is done. The features of the target dataset is standardized according to the deviation and mean calculated from those of the source dataset and then, the target dataset is examined in the pattern matching process. The retrieved BMU is employed in the synthesis process, and finally, we obtain

a synthesis result.

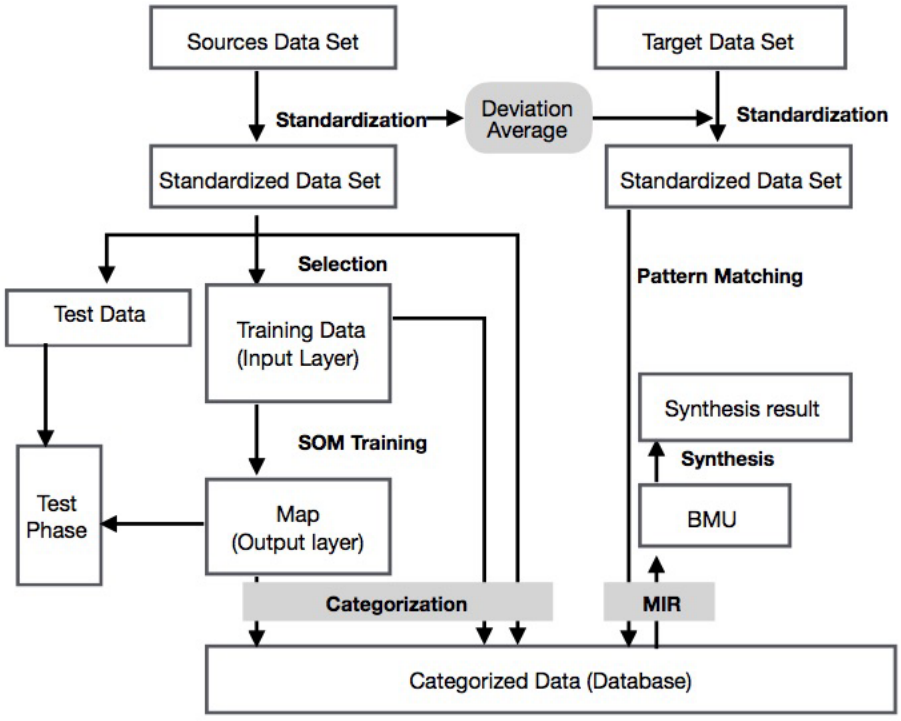


Figure 63: The Categorization and MIR processes in our system

Figure 64 to 68 illustrate the resultant SOM of the three database introduced in the table 2. The vector of each node, which consists of twelve MFCCs, is visualized by a black line and a white number represent a count of assigned units. We employed larger map sizes when classifying a larger size of database and vice versa. As shown in figures, the neighboring nodes have similar MFCCs patterns and the farthest have different one.

The counts of the assigned units are projected on 3D graphs as shown in figure 65, 67, and 69. The units are not assigned evenly over the map, but a considerable number of units are concentrated in a few nodes. This bias implies the character of the database or otherwise, the map is too small to represent another variation of characteristics patterns.

Another aspect of the bias arises from silence in audio data. As the SOM represent the topological relationships between distinct patterns, the map has only one or a few nodes which represent silence. The audio data often contain empty or quiet parts at the beginning and end of the data or between each musical event, and therefore, a plenty of

empty units are assigned to a small area of the map, and as a result, it yields a strong bias in the distribution of the assigned units. In order to prove this hypothesis, we tested the SOM training with a Magnitude threshold which is used to assess the input units whether it contains signals or silence.

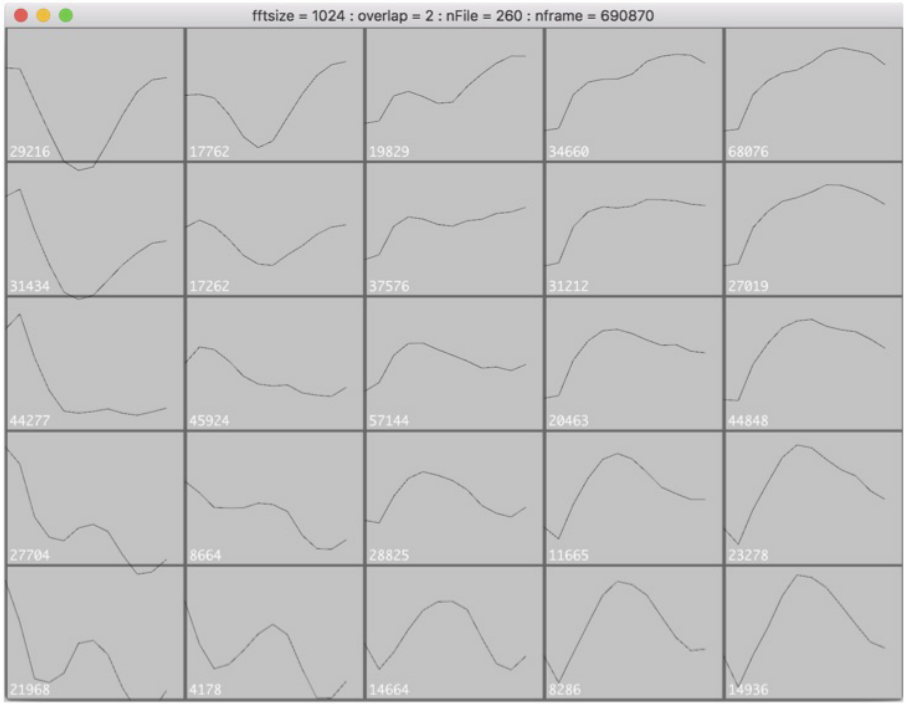


Figure 64: The SOM (5x5 size) of the Piano database.

The black lines represent the eleven MFCCs of each node, and the numbers show the units categorized in the node. The MFCCs patterns of the piano database have gentle curves and are relatively similar each other.

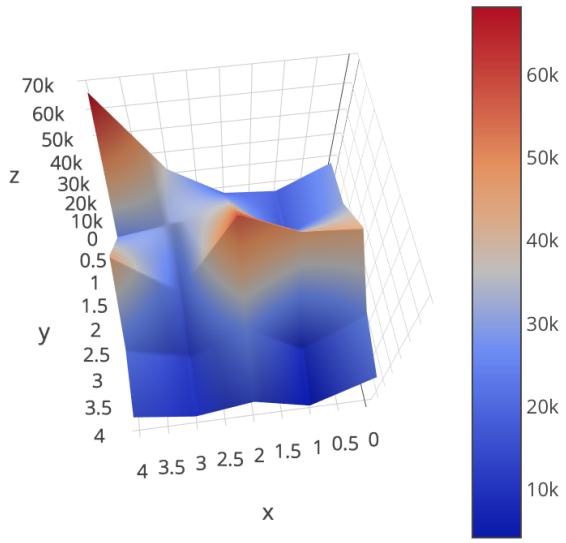


Figure 65: The distribution map of the categorization result of the Piano database.

The z-axis represents the count of units assigned to the corresponding node located on the (x,y) coordinate of the map, which is also illustrated by color, where the red color represents a higher number, and the blue color represents a lower number of units. Note that the angle of the graph is appropriately rotated in order to show all undulation of the entire graph.

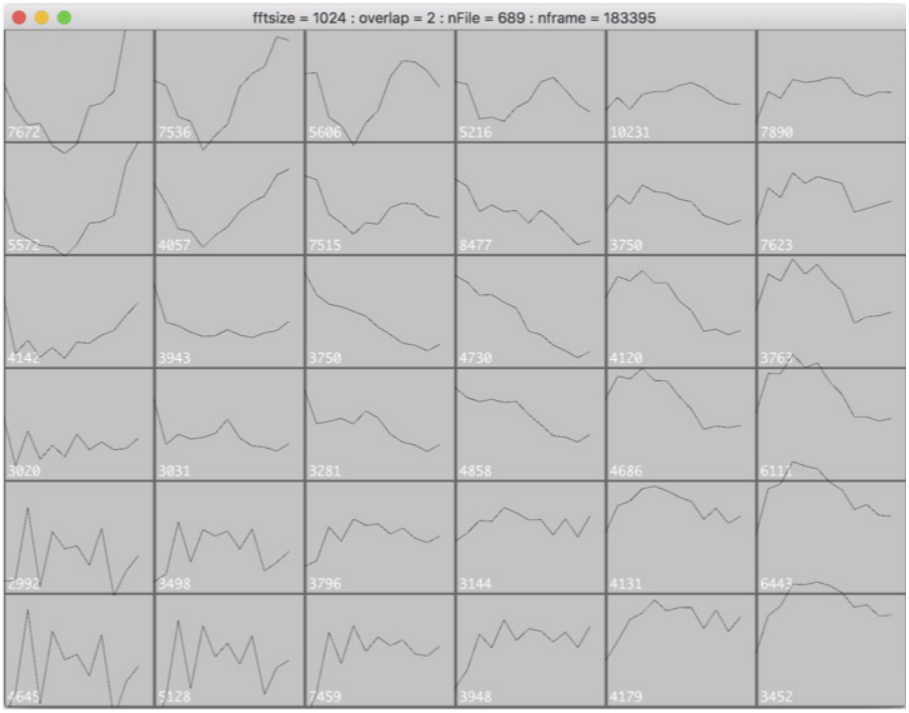


Figure 66: The SOM (6x6 size) of the Percussion database.

The black lines represent the eleven MFCCs of each node, and the numbers show the units categorized in the node. The MFCCs patterns of the percussion database are diverse, spiky to gentle, and flat to rough curves due to their variable sound characters.

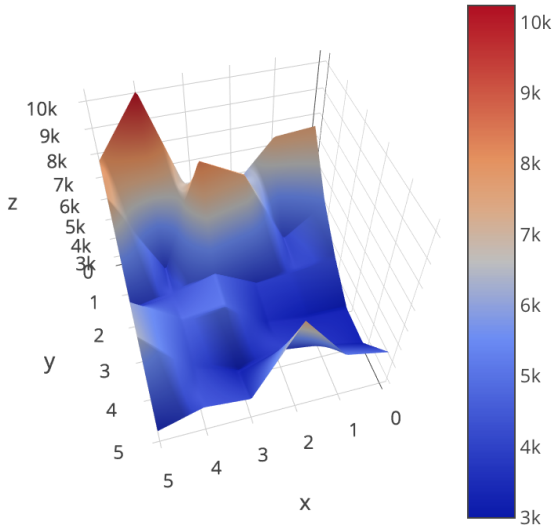


Figure 67: The distribution map of the categorization result of the Percussion database.

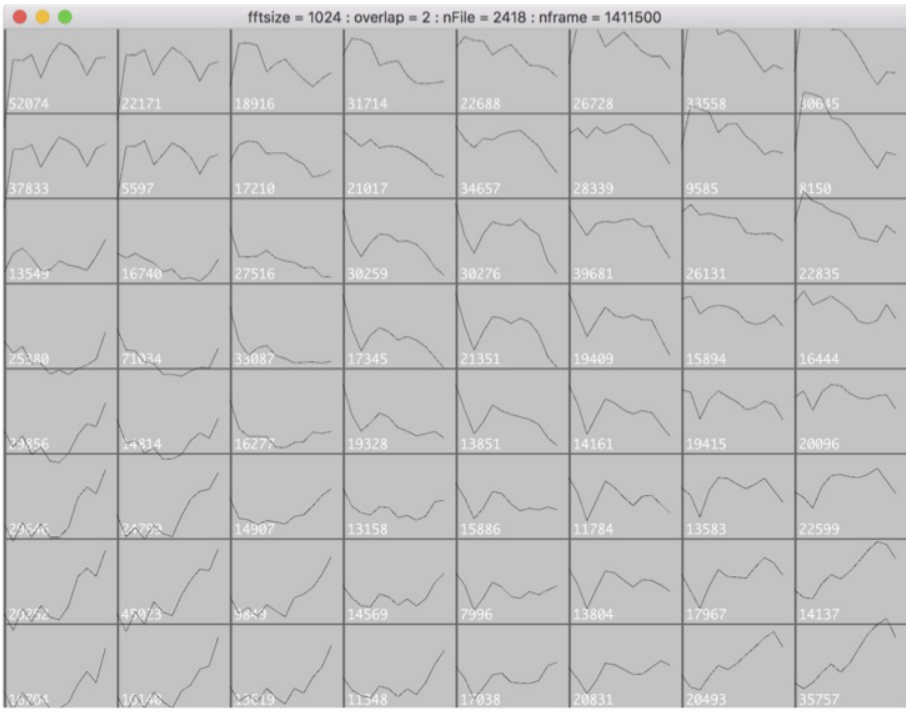


Figure 68: The SOM (8x8 size) of the Violoncello database.

Due to its large database size, we made a large map size (8x8). The Violoncello database is comprised of a large size of similar sounds and their variations, and also large number of unique sounds.

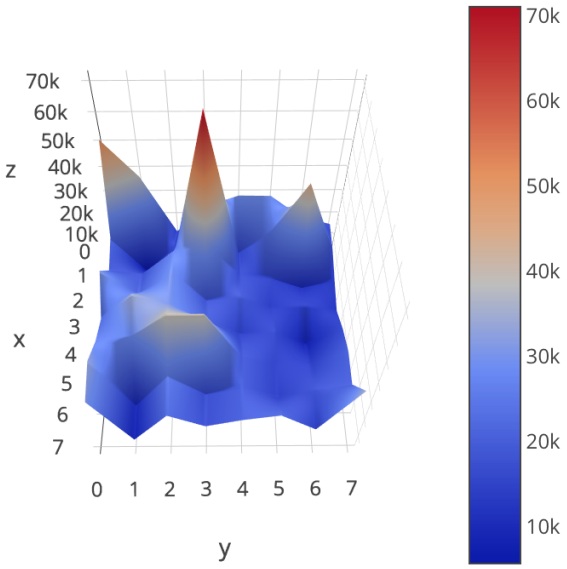


Figure 69: The distribution map of the categorization result of the Violoncello database.

16.4.2 Short digression on single peak distribution

Figure 70 shows two 3D graphs representing the categorization results operated under different Magnitude thresholds. The left case excluded ca. 5.15% of units whose Magnitudes are under 0.0001, and the right case excluded ca. 53.5% of units whose Magnitudes are under 0.01. While the left map has a peak in which plenty a number of units are assigned, the right one has multiple moderate peaks which proves that the units disperse in a broad area of the map. This result suggests that the most of the silence or quiet units tend to concentrate on a few nodes, and others spread out in the entire map. We can easily explain what is happening throughout the learning process. The silence or sufficiently quiet units have similar vectors which are always presented on the same BMU on the map through the iterations. Since both learning rate and radius become smaller as progressing the learning, the vector representing silence converges to a narrow area of the large map, or an exclusive node in a small map. The silence units do not have many variations, and thus, they are categorized to the same place.

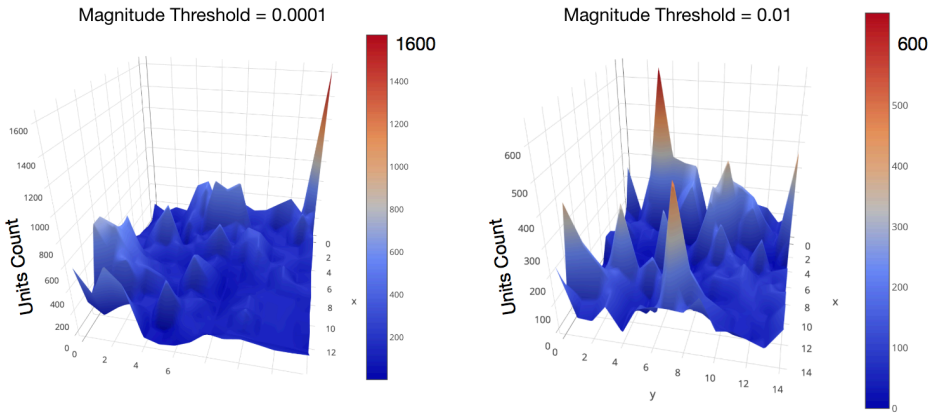


Figure 70: Comparison of the distributions of assigned units between Magnitude threshold 0.0001 and 0.01

The categorization result with the Magnitude Threshold of 0.0001 (right) and with 0.01 (left). While the right graph has an exclusive peak which indicates around 1600 units, the peaks of the left one disperse over the entire map. Note that the maximum units count is now around 600 which is approximately one third to that of the right.

16.5 Parameters of the SOM

As previously discussed, we give some parameters in order to constrain the learning process of the SOM. There are four crucial parameters which influence the quality of

learning result, which are the map size, learning radius, initial learning rate, and learning iteration number. Appropriate values should be given depending on the properties of the database.

For instance, the learning rate and iteration affect the rate of learning progression, a high learning rate accelerates the learning progress but it may result in a rough or erroneous map. The negative side of the high learning rate would be improved by giving more learning iterations. However, the problem is still significant. Since the learning rate slowly decreases as progresses the learning, the units selected at the beginning give larger impact on the initial map and thus, they are superior to the laters. This phenomenon is called a winner-takes-all competition[59]. Therefore, we need to balance the learning rate and iterative number appropriately in order to avoid this adverse effect on the unsupervised iterative learning.

The trained map is expected to cover whole characteristics patterns of the units constituting the database, and the ideal categorization lets the units distribute evenly over the map. However, in a practical situation, it is hardly possible to obtain such a categorization result due to variable factors. For instance, we can consider a case when the units of a database can be divided into major groups in their characteristics patterns, and the two are very different to each other. The resultant map attempts to represent their major patterns as well as a considerable number of complemental patterns which represent the transitions between them. However, there are only a few units that match to the complemental nodes, and as a result, the assigned units are dense in some nodes and are sparse in others. This consideration suggests that the bias against the distribution of the assigned units over the map is inevitable. However, we can improve it by adjusting the appropriate parameters of the SOM and removing some silent units as previously discussed.

16.6 The Test using the Cross-Validation method

In order to adjust the parameters, we need to reveal the process of learning and assess the quality of the resultant map. We operate a test against the trained map to evaluate its accuracy and generalization levels. We have employed the Cross-Validation (CV) method which has been developed as a remedy to prevent a learning space from progressing an overfitting[58]. The result of the CV describes the adequateness of the trained map for the same dataset from the input. We have also extracted some features

throughout the test process, which represents the distribution of the units categorized in the map, the distance between the inputs and map, and the generalization level of the trained map.

In the CV, we first separate an input set into a test and training. The test set is organized by picking up a unit every two to several continuous units from the input. The continuous units correspond to a sequence of a musical event, and thus, the neighboring units may have similar feature vectors. The interval of each test unit should be defined according to the hopsize[13] used in the audio analysis because the ideal test set should have different but similar set of vectors to those of the training. Therefore, the small hopsize requires more interval to extract the test units in order to avoid equaling the two sets.

Figure 71 illustrates the process of the CV where it picks up a test unit every two inputs.

Figure 71: Cross Validation test algorithm.

This figure illustrates dividing the database into two sets which are the training data and the test data. The Cross Validation chose the test data every one to several data and thus, it can make a separated data set which can cover the characteristics of the entire data set.

Since the CV is a method to divide test and training data, we need to analyze the learning result and extract some features representing the quality in order to know the adequateness of the parameters.

We investigated the distribution of the assigned units over a map in order to assess how the resultant map covers the characteristic pattern of the training set. Furthermore, we process the test dataset consisting of another sound units which has no relations with the training set in order to measure the generalization level towards unpredicted

[13]The hopsize constrains the extent of shifting the FFT frame on the time axis, and thus, the small hopsize will produce units that contain more common signals in its neighboring units, and vice versa.

sound type. The test dataset covers various sound characteristics patterns, from noisy to timbral tone, human voice, instrumental sound, percussive to resonated tone, and so on.

16.7 Distribution Coefficient

In order to know the distribution of the units, we simply counted the number of units assigned to each node and calculated the deviation. For instance, the table 1, which is shown in the Categorization section, illustrates the list of units corresponding to each node. Firstly, we calculate a mean count M which is shown as following formula.

$$M = \frac{TotalUnitsCount}{MapSize} \tag{72}$$

The distribution coefficient T represents the deviation between the mean count M and the actual count of each node $x_{i,j}$ where i, j represents a pair of indices of a node as follows.

$$T = \frac{(\sum_1^i \sum_1^j \sqrt{(M - x_{i,j})^2})/MapSize}{M} \tag{73}$$

When the units concentrate on a small area of the map, T gets higher value, and in contrast, it gets close to 0 when they spread evenly over a map. For instance, when a total 1000 units are categorized in a map whose size is 10x10, then M is $1000/100 = 10$ which means, each node should have ten units in average. If all nodes get 10 units as a result of the categorization, then the T becomes 0 as seen in the equation $\sqrt{(10 - (x_{i,j} = 10))^2}$, and otherwise, T takes a larger value as higher deviations.

We also calculated the average distance and the deviation of units for a node. The average is calculated by summing all distances between each unit and its BMU. The deviation represents how the units vary in a node. Both features represent the categorization quality of each node, and we unified them by calculating means of two features in order to describe the quality of the map.

17 Test Analysis Results and Discussion

We will focus on the Percussion database for the test because it is expected to have various sound qualities of all. Figures 75, 76, 75, and 76 illustrate the test results of the Map size, Learning Radius, Iteration, and Learning Radius respectively. Please note

that the horizontal axes of the iteration and the radius are described in the logarithmic scales. For each test, we determined six features consisting of two main groups which are the self-test and the other-test. The self-test uses the test dataset extracted from the input according to the CV method, and the other-test uses another dataset consisting various sounds. Figure 72 instructs how the features are identified by distinct marks, and the Self-test and the Other-test are identified black or white colors.

First of all, we describe our interpretation of the features as follows; the combination of the high distance and deviation suggest that most of the units do not fit their BMUs. The low distance but the high deviation indicate that some units fit their BMUs very well, but some are significantly far. The high distance but the low deviation suggest that the most units equally have considerable distances to their BMUs. The low distance and deviation suggest that the most units fit their BMUs sufficiently.

The distribution coefficient is also one of the most significant features to evaluate the categorization quality. A low coefficient shows that the units are evenly distributed over the map, which is expected to be an ideal map, and the high coefficient shows that some units concentrate on a small area of the map which implies a high learning bias. This feature should be carefully assessed by referring to the map size because the small map size can easily yield a low coefficient. Therefore, the low coefficient does not always indicate a satisfactory result.

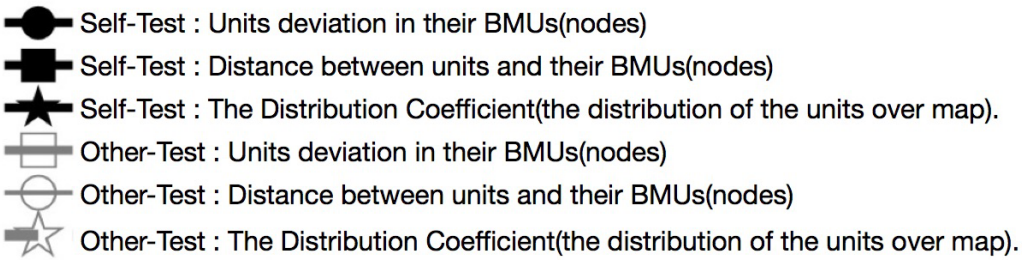


Figure 72: Test result instruction

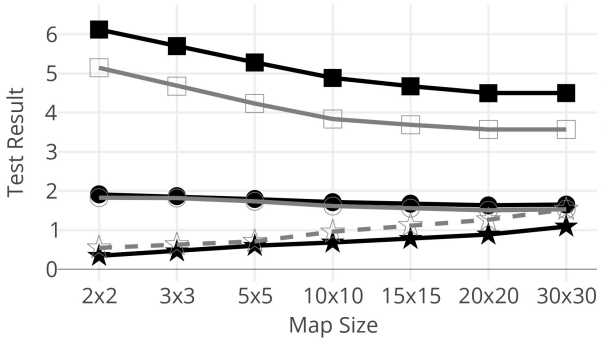


Figure 73: Test results for Map size (Percussion)

Initial Learning Rate = 0.1, Radius = Map size, Iteration = optional

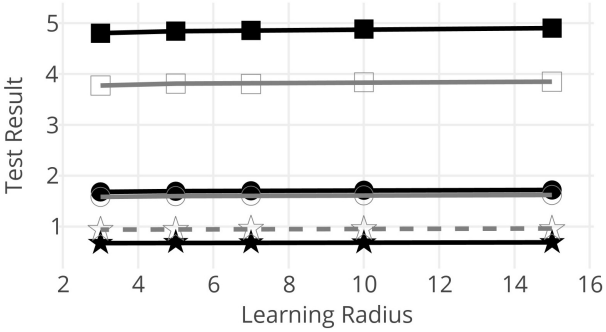


Figure 74: Test results for Learning Radius (Percussion)

Initial Learning Rate = 0.1, Map size = 10x10, Iteration = 2000

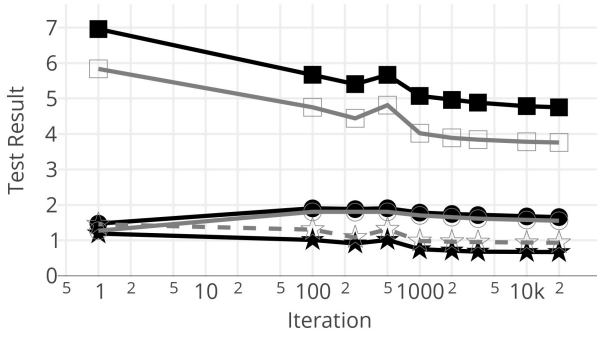


Figure 75: Test results for Iteration (Percussion)

Initial Learning Rate = 0.1, Radius = Map size = 10x10, Iteration = optional

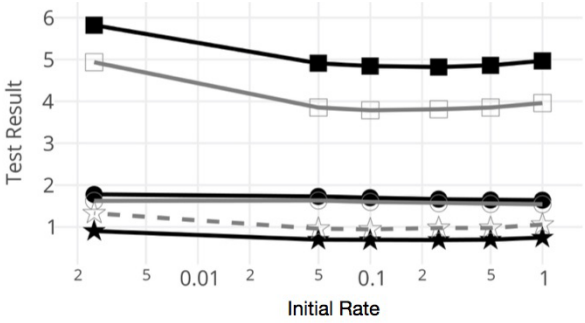


Figure 76: Test results for Initial Learning Rate (Percussion)

Map Radius = Map size = 10x10, Iteration = 5000. The numbers on the horizontal axis

Figure 75 illustrates that the larger map sizes have lower distances and deviations in both self and other-test. The decrement is exponential and then becomes moderate for sizes bigger than 20x20. In contrast, the distribution coefficients increase significantly in the larger map sizes. For instance, the self-distribution coefficient takes about 3.43 at the 2x2 map size, 0.59 at the 5x5, 0.77 at 15x15, and 1.09 at the 30x30. The result of the 30x30 map size gets about twice larger than that of the 5x5. We have assumed the case for the increment as a result of an excessive subdivision of the characteristics pattern on the map. In the case of the Percussion database, the sound qualities are so various that they have significant contrasts in their characters. Since a map attempts

to represent all statistic patterns and their transitions within the restricted number of nodes, the extreme patterns are moderated, and they are adjusted to be closer to the averaged characteristics of the entire dataset. While the average units fit sufficiently the pattern of a map, the unique units, which have contrastive patterns to the average, are not assigned to the satisfactory BMU, but they concentrate on the nodes which are the boundary of another patterns. The typical example is silent units, which tend to concentrate on a node which has the lowest Magnitude, and therefore, their influences on the distribution coefficients are crucial. As evidence for our hypothesis, the test for the map size of the piano database has a flat self-distribution coefficient whereases it has an ascending other-distribution coefficient. The piano database consists of a tremendous number of isolated piano samples that capturing whole piano resonances until they reach the silence so that all sounds have similar sound structures which vary smoothly through time. For this kind of dataset, it is easier to build a statistic models that fit the set, and thus, the test result yields better self-distance and deviation results than those of the other-test. The supremacy of the self-dataset is getting higher in a larger map size, and ultimately, it is recognized as an overfitting because the resultant map fits very well for the self-dataset and gets far for the others. In the percussion database, on the other hand, the overfitting problem seems not to occur due to its variety of sound qualities.

According to the above discussion, we have concluded that the appropriate map size stands around the boundary where both distances and deviations get flattened for the self and other-test, and simultaneously the distribution coefficients are moderate. The satisfactory result is seen at the 10x10 or 15x15 sizes in the percussion database test result. For the piano database, it corresponds to the 10x10 map size.

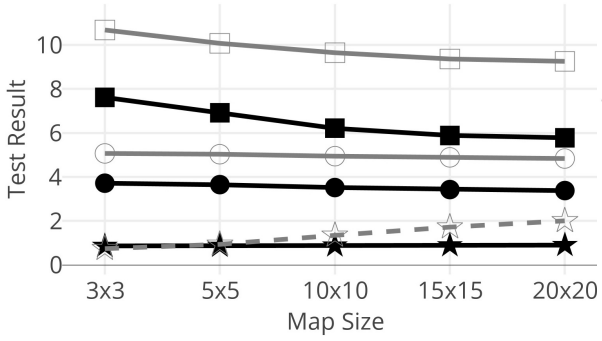


Figure 77: Test results for Map size (Piano)

Initial Learning Rate = 0.1, Radius = Map size, Iteration = 3000

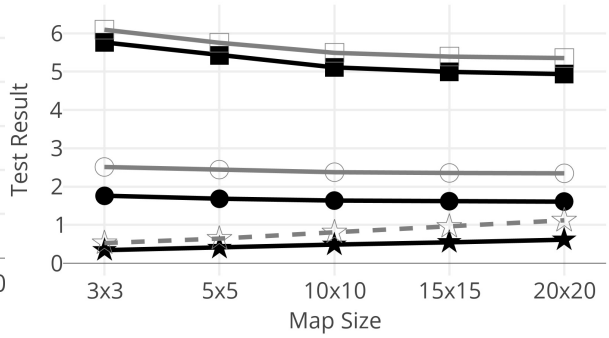


Figure 78: Test results for Map size (Violoncello)

Initial Learning Rate = 0.1, Radius = Map size, Iteration = 3000

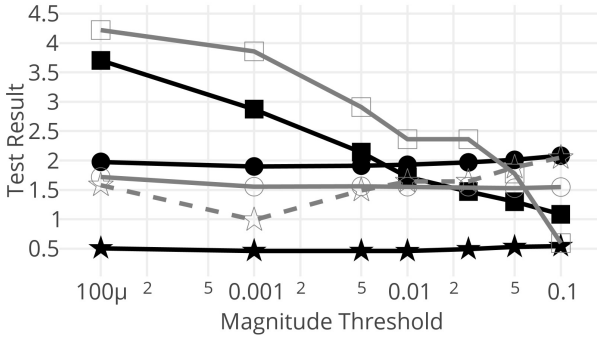


Figure 79: Magnitude Threshold Test

Initial Learning Rate = 0.1, Radius = 10, Map size = 10x10, Iteration = 3000

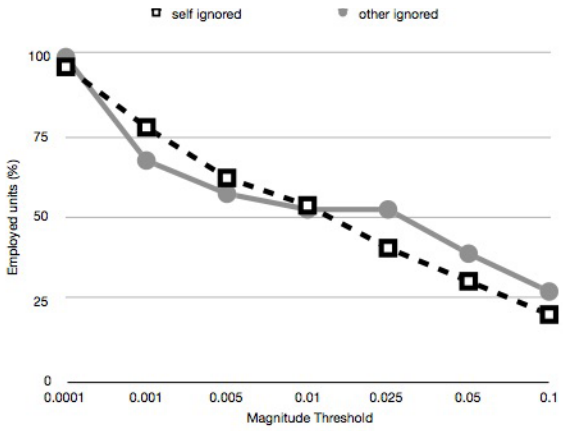


Figure 80: Magnitude Threshold Test : The employed data rate

Initial Learning Rate = 0.1, Radius = 10, Map size = 10x10, Iteration = 3000

In most cases, the silent or extremely quiet units make a negative impact on the distribution of the units because they tend to concentrate on a small area of the map regardless of the map size as discussed in the section 16.4.2. Thats the reason why we employed the Magnitude threshold to ignore these units from the test process. While the strict threshold can show us the actual distribution of the meaningful units, it gives us a risk to discard too many quest units which may be a majority of the input dataset. For the previous test, we have employed the threshold value of 0.001 which is

the moderate value where the 20% to 30% of units are determined as a silent or enough quiet units, and they are excluded from the learning test.

Figures 79 and 80 illustrate the learning test results for the different Magnitude Thresholds and the employed units rate calculated by (100 - (removed silent or quiet units rate))% respectively. This test was operated not by the CV method but by using the whole input and other-dataset. Since the higher Magnitude threshold removes a large number of units, we need to use as many units as possible to obtain more accurate results. The threshold made a significant impact on the units removal and the quality of the learning results. While the deviations do not change actively, the distances of both self and other dataset decreases significantly as giving a higher threshold. The self-distribution coefficient stays flat but the other-distribution increases moderately. Eventually, more than 80% of the input units are removed when the Magnitude threshold is 0.1.

We need to consider the crucial impact of the noise on the MFCCs features under the low magnitude. When the audio signal is weak, the envelope of the spectral is influenced significantly by the background noises, and as a result, it can not represent the accurate timbral characteristics of the target. We can assume that this problem declines the quality of the resultant map featuring the MFCCs. Araki who is a Japanese researcher developing the speech recognition system described the difficulty of the noise canceling when extracting the accurate MFCCs [61].

18 Two steps Self-Organizing Map

We will shortly introduce the two steps self-organizing map which consists of two categorization phases. In this method, we have built two maps by using different vectors of descriptive features. The first map, which is called a parent map, has superior features such as MFCCs. Each node of the parent map has its own inferior SOM called a child map, which is trained by means of another combination of the features such as Magnitude, Centroid, and Pitches. In this case, the units are firstly categorized into different timbral characters, and subsequently, the similar units are again categorized according to their Magnitude, Centroid, or Pitch into smaller categories. Since the parent map determines the major categorization of the units, this process can prioritize the features whereases the conventional SOM equally employs all features. Generally, we utilize

larger size of SOM for the parent map and smaller for the child map because the number of units gets exponentially smaller throughout the categorization. The advantage of this method is to avoid making complicated neural network due to a large number of features and is expected to enhance the generalization level. Furthermore, it is easy to understand how the units are categorized in each SOM because each categorization phase is focused on particular features.

19 Regression analysis

Until here, we have discussed the classification method using the SOM for units. The units hold their features extracted by FFT based spectral analysis. They are mechanically separated into FFT frame size and do not correspond to any musical event. In reality, we have implemented segments of musical events that are separated by our algorithm discussed in the Segmentation section. The length of a units sequence is different depending on the contained musical event, and it gains a number of features as it gets longer in order to represent the time-varying information. The concise features are ideal to build a straightforward and reliable classification model, and thus, we need to reduce the dimension of features before processing the learning.

We have operated regression analysis to extract a new feature which represents a statistic tendency of the variation of features through time. For instance, the structure of an isolated piano sound forms a decrescendo gesture in its magnitude, and it is represented as a chain of descending features. Assume a chain as a linear function $f(x)$, we can estimate its inclination by means of differential calculus which yields a Differential Coefficient (DC) representing a statistic tendency of the function. Figure 81 illustrates the model of regression analysis, where some number of features distributed on a space are represented by dots, and a line shows their statistic tendency which is estimated by the analysis. Figure 82 illustrates an application example of the analysis of the audio signal in which, we employ the first frame in order to define the beginning feature of the segment and then interpolate its time-varying gradient by the DC. In this method, we can reduce the number of features to two.

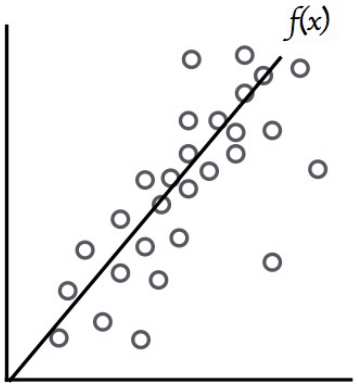


Figure 81: The regression analysis model

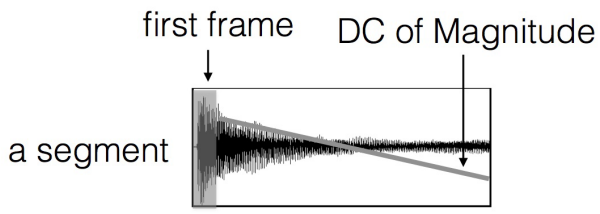


Figure 82: The regression analysis for a audio segment

We calculated the regression analysis by using the least squares method first introduced by Adrien-Marie Legendre in 1805. As we only need to estimate the gradient of the time-varying features, and thus, it is simply obtained by following formula.

$$a = \frac{n\sum_{k=1}^{n} x_k y_k - \sum_{k=1}^{n} x_k \sum_{k=1}^{n} y_k}{n\sum_{k=1}^{n} x_k^2 - (\sum_{k=1}^{n} x_k)^2} \tag{74}$$

In the case of Magnitude, we give a ramp function which yields (1,2,3,. . . ,n) to $x(k)$ and the Magnitude features to $y(k)$.

A weak point of the regression analysis is that, when the features fluctuate throughout the segment, regression analysis does not represent the moves but merely yields a flat gradient. This problem will be improved by subdividing the audio signals into smaller chunks of segments, or otherwise, by employing the deviations representing the distances between the member units and the resultant gradient line in order to measure the fluctuation bands.

20 Discussion

In this section, we have discussed the audio classification method by using the SOM. The SOM produces a map representing the statistic characteristics patterns of the source dataset, and it is utilized for the pattern matching between sources and target audio data. In MIR, it is beneficial to use a categorized database to improve the performance of our data retrieval task significantly because it only needs to search particular patterns to find the BMU instead of searching the entire dataset. Since we use a small database,

whose sizes are at maximum several GB, the concise and straightforward classification model is adequate for building our retrieval system. In such a simple model, sufficient generalization levels are maintainable[59].

When using a tremendous size of a database, such as TB of data, these characteristics are not efficiently represented by a map. In this case, the idea of the two steps SOM is applicable. When an enormous amount of units concentrate on a node of the map, the node should have a stepwise map which represents variations of its characteristics pattern. While the previously discussed two-step SOM utilizes different features in a parent and child SOMs to prioritize the characteristics, this version of the SOM applies the same features in both maps to subdivide the patterns and refine the categorization. This SOM may improve the concentration problem of the units categorization.

As the conventional 2D SOM forms a grid structure, the edge or corner of the map does not have neighbor relationships on their one or two sides. These nodes on the boundaries of the map have fewer chances to be updated due to their fewer neighbors. This is called border effect which interrupts the topological mapping and it causes a less accurate result. There are some studies in which developing another shape of SOM to preserve the continuity of the topological characteristics by permuting the edge of the map. Oyabu et al. have developed a spherical SOM based on the latitude and longitude grid system. Each node is shaped in a square as the conventional SOM is, and thus, it is easier to represent the position of the node and the distance between nodes. However, the constituent squares positioned around the equator are much larger than those of around the polar regions, and thus, the distances between nodes are exponentially significantly changed depending on the position on the sphere. Therefore, Wu et. at. suggested a geodesic SOM [55] [56] whose node is shaped in a triangle and the sphere consists of the tessellation, and therefore, the small size of the geodesic SOM becomes a regular octahedron, icosahedron, and so on. The geodesic SOM is more time-consuming than the conventional SOM, due to its shapes and map size. The map size is given by the equation $(10 * f^2 + 2)$ where f denotes the frequency number, which can easily become much larger than the simple SOM (its size corresponds to (f^2)) we used in our system.

Deep learning is a novel neural network algorithm which is overtaking other network models, and it is catching a great deal of attention in the wide area of the engineering. While a conventional neural network requires features, which are extracted by the

FFT analysis in our case, to build a statistic patterns in the learning process, the deep learning does not utilize this way, but it automatically estimates the features which may be the best efficient to identify the input data characteristics. The neurons construct multi-layer neural networks, where each node is complicatedly connected to each other to represent a particular characteristics pattern. An interesting aspect of deep learning is that the features and the patterns automatically extracted throughout the learning process are unknown to us. This anonymity of the features is one of a problem in the deep learning because we can not assess how and why the network gives us an output. There are some studies to interpret a deep neural network model and to explain its predictions[57]. The further study of the deep learning is one of our most significant subjects in the future.

Part VI
Synthesis

21 Introduction

In this section, we will discuss the various synthesis techniques which are the core of our Computer Assisted Composition (CAC) Program. Our program is designed to synthesize all kinds of audio data provided as a target by using a database of freely definable source audio data. Multiple source audio data are assembled according to the characteristics of the target extracted by the FFT based spectral analysis, and therefore, it is possible to simulate or orchestrate the target with different source data. The program can produce musical examples containing sound textures, gestures, and instrumentations by outputting an audio file.

Our synthesis technique originates from the Concatenative Sound Synthesis (CSS) technique which was formerly studied in speech synthesis[69] and then advanced by Diemo Schwarz[65]. The synthesis allows for the production of sound sequences by means of chaining multiple segmented sound data together[]. It requires a large database of source sounds segmented into units with the units selection algorithm that finds the best matched sounds to the segments of the target. The strength of the CSS technique for instrumental music composition is the versatility of the synthesis results. The source audio data can be flexibly collected according to users purposes whereby the target sound is possibly reconstructed with any kinds of audio data . This means that, the program does not restrict the genre or style of music once users make their own database. Another advantage is that the algorithm is expected to produce natural and practical synthesis result because it uses actual audio data and the quality of the audio data is not declined through the synthesis process. Our idea is to extend the synthesis technique to realize the advanced musical representations to be applied to the instrumental music composition, which can be achieved by focusing on the following two main subjects.

The first is to synthesize a target using musically meaningful segments instead of mechanically segmented FFT frames. This way, there should be a great probability of enhancing the quality of the synthesis result which are reproducible by musicians. This also implies the representation of longer sequences of the target units, where the

pattern matching is operated for the musical events not for the FFT frames.

The second is the polyphonic reconstruction of a target structure. For instrumental music composition, it is desired that the program is designed to generate polyphonic synthesis results that are accomplished by combining different source data in order to enhance the results similarity to the target data. This function enables users to simulate not only single similar sound data, but also to simulate the combinations of different instruments or playing techniques. Our synthesis program estimates the best combination of the units or segments from database according to the decomposition results carried out by the Non-Negative Matrix Factorization (NMF) and the Specmurt analysis.

We will show the algorithms of segments and polyphonic representations individually and subsequently, demonstrating the synthesis process integrating both of them. We will also discuss a Graphic User Interface (GUI) which can precisely control various parameters aiming to restrict the synthesis result and an ad-hoc score which illustrates the structure of the synthesis result to be used for the analysis of the result or the transcription. The synthesis programs and GUIs have been implemented on the different platforms in order to investigate the availability of the system in different environments and also to find the advantages and disadvantages of the platforms. We have built three programs for MaxMSP, Web application, and Cocoa Application for MacOS.

22 The Concatenative Sound Synthesis

The CSS uses the similar basic method of granular synthesis : longer sounds are produced by the superposition and synchronization of short grains (a sort of sound-quanta). The basic methods of time-stretching and pitch-shifting developed for granular synthesis are also employed in many concatenative synthesis program which can be seen in the Adaptive CSS[69].

The CSS technique produces a sound sequence by concatenating multiple short length audio units following a certain rule. The rule is different depending on the purposes: for instance, the speech synthesis utilizes texts which illustrate the combinations of vowels and consonants for generating artificial voices. Cambell et al. developed a generic speech synthesis system called CHARTR (1994) which reproduce the artificial human utterances by constructing the chains with previously recorded and analyzed

phonemes[4]. This system uses a database consisting of a great amount of recorded human speech data with which it synthesizes a concatenation of speech sounds. This system analyzes not only a configuration of phonemes such as combinations of vowel and consonants but also various intonations and speech speed in a human speech by combining both computer analysis and perceptional evaluation by humans. With the information obtained by computer analysis and human evaluation, CHATR generates an artificial speech sounds from a line of text by retrieving the most similar fragments from the recorded speech data. Although CHATR is implemented with a database of a tremendous amount of speech sound, it is so conceived that it can significantly reduce the computational effort and simultaneously can realize a more natural synthesis result than other programs. One reason for this is due to the fact that the speech sound database also includes intonation and tempo information. With this information, CHATR can retrieve optimal sound data to create more natural speech sound in line with the specific context. It is thus, concluded that the larger the size of the database, the higher the quality of speech that can be synthesized using CHATR algorithms. Some sampler softwares utilize MIDI data as a rule to produce appropriate sounds by combining attack and resonance samples recorded separately. These softwares load presets which contain the information of audio samples linked to the corresponding MIDI commands.

Schwarz has developed the data-driven concatenative sound synthesis (2003) which operates with a large database of source audio data segmented into units. In his approach, a sound file, which is called as target, is employed as a rule of reconstruction. The target is separated into units and each unit is analyzed to extract its features (which are called descriptors) identifying the characteristics. The synthesis result is assembled from other sounds based on a measure of similarity between the target and sources [63]. Similar synthesis program called *MATConcat* was also developed by Strum in 2004[64]. He applied MATLAB library to build the CSS system. The CSS technique was also applied for the real-time live performance tool called *CataRT* developed by Schwarz et al. in 2007[66] and has been updated frequently. In this software, a large amount of grains is placed in specific positions of an analysis space, called descriptor space, according to various characteristics of the sound. This software enables users to access directly similar sounds, and it easily generates a type of gradation effect between two different characteristics of granular sounds.

Strum introduced Adaptive Concatenative Sound Synthesis (ACSS) (2006) which is an advanced version of the CSS. While the CSS merely concatenate the units according to the descriptors of the target, the ACSS transforms the units to adapt the corresponding units of the target by means of frequency modulation or other digital audio effects[69]. Strum described that a one side of the ACSS is similar to the granular synthesis or multiple-wavetable synthesis, but it operates rather a sound scrambling or a sophisticated version of remixing. His approach targets to the Micromontage composition which is one way to create music by concatenating small chunks of audio data.

Further development for the CSS is seen in the AudioGuide by B. Hackbarth and Schwarz [67] in 2011 to 2013. The AudioGuide has a unique matching system where the source units in a database are normalized in order to facilitate similarity measurements before matching the target units. The normalization is operated according to the maximum and minimum or median and standard deviation. Since the target and source units are separately normalized by different criteria, their values are not anymore compatible each other, so that it is not for reproducing melody or harmony. However, the AudioGuide aims to represent the variability of the target units and eventually reproduces its sound transition or gesture efficiently even though using the limited sources in a database[68]. We also applied normalization or standardization as discussed in the Classification section, but our case utilizes the same criteria through the scaling of both target and source units, in order to realize the compatibility between distinct features. Therefore, our system has less probability to get an appropriate result when employing a database consisting of significantly limited sources. Hackbarth aims to realize both time-varying representation and polyphonic reconstruction by means of the subtractive spectral algorithm which will be discussed in the further section. The Concatenative Sound Texture Synthesis by Schwarz and Yeh in 2016[70] aims to produce sound textures by concatenating chunks of audio signals smoothly by applying macro event cross-fade algorithm which is similar to PSOLA (Pitch-Synchronous Overlap-Add) algorithms. This algorithm suggested a way to produce a sequence of natural sound timbral texture.

There are various applications employing similar techniques of the CSS.

Sonote beat re-mix synthesis software developed by YAMAHA [80] is also one of the interesting examples of the synthesis techniques. This software remixes a beat sound in a drum loop by extracting and replacing it with similar samples from other audio files.

Users can easily create their own database by adding optional audio files in the program, thus producing complicated combinations of different beats. Another example of the application of these techniques on a computer-aided orchestration program is *Orchidee* or *Orchids* developed by IRCAM[82]. Orchidee produces orchestration examples or makes suggestions to composers by reconstructing a one frame spectral structure of target sound data with different instruments. Orchidee is not able to represent the transition between different timbres in a target sound following the time axis, as it only simulates an acoustic structure of a short frame with selected instruments. Its advanced software Orchids employs temporal representations by tracking adequate peaks of spectrogram and transport them to pairs of pitch and energy envelope and compose isolated instrumental sounds according to them. This approach is rather related to the Spectralismic composition than Micromontage composition compared with the Strum case. *Synful* , a software synthesizer, generates instrument sounds by using its database which includes analyzed data of short phrases played by different instruments[81]. The database of Synful consists of analysis data of short phrases played by various instruments, and it contains how the property of instrumental sounds including attack, sustain, release, and these transitions, change depending on performance situations such as specific melody, dynamics, articulation, etc. Synful does not obtain a target data from sound data but from the conversion of inserted MIDI files which contains the information of instrumentation, concrete melody, gesture, pitch, velocity, rhythm, and so on. It receives musical information from an inserted MIDI file and reconstitutes with the closest units retrieved from the database. Thus, Synful can generate very natural and high-quality instrumental sounds from a MIDI file as long as it can find the adequate phrase pattern from the database.

Synful obtains a MIDI data as an input and includes the following information: instruments, melody, pitch, velocity, etc. The software then retrieves the most appropriate phrases from the database and synthesize them together.

As introduced above, the basic concept of the CSS is advanced with additional algorithms to apply to the different purposes in diverse research areas. Basically, the CSS is a synthesis technique covering the following three syntheses: Sample-based synthesis which operates with sampled audio data, an Example-based synthesis which operates according to various example data, and Corpus-based synthesis which runs with a prepared source database. For all techniques, the organization of the corpus

sound database is a common key to enhance the quality of the result, which should be done before the synthesis process.

23 Unit based Concatenative Sound Synthesis

Firstly, we will describe the most simple synthesis technique, which is the frame-based CSS. This technique only focuses on finding the BMU from the database without considering any relationships between adjacent units. The synthesis result is strongly related to the Music Mosaicing which is one of a Micromontage sound art.

23.1 Music Mosaicing

Music Mosaicing is one easy example that can produce a similar result of CSS. Research has been done on a Music Mosaic that can be generated with a creative intent by means of aggregating multiple small chunks of sound data. The term Music Mosaicing is identified by François Patchet as being analogous to similar applications in Image Mosaicing[71]. Image or Photo Mosaicing is a process in which a picture (or part of picture) is represented by using multiple trimmed down or segmented pictures, which are referred to as the units in a type of micro collage. There are several examples of Photo Mosaicing pictures (figure 83) on the web site of Robert Silvers, who created an algorithm for generating Photo Mosaicing with a computer[72]. As in Photo Mosaicing, Music Mosaicing creates a musical sound data by means of synthesizing multiple small segments of a possibly different audio data source. In CSS, some algorithms are employed in order to produce sequences of multiple segmented sound data for the purpose of generating longer musical transitions rather than generating mosaic sound.

Figure 83: Photo Mosaicing

Robert Silvers presents a Photo mosaicing example on his web site. There is an example using a famous paint Girl with a Pearl Earring by Johannes Vermeer. The picture is used as a target and segmented into small fragments, and they are analyzed by computer. The target picture (right picture) is assembled according to the analysis information with samples from a corpus of source pictures shown in the middle, and then, the generated result is as shown in the left.

23.2 Algorithm of the frame-based CSS

The frame-based CSS is intended to use FFT frame sizes from 512 samples (11.6ms. at a sample rate of 44.1kHz) to 8192 samples (185ms.), and the frame size coheres through one sound synthesis process to realize an efficient operation.

The process of the synthesis is executed in the following order: target audio data analysis, data retrieval, concatenative synthesis and production of the audio data. The input target data is segmented into units, in this case, into FFT frames, and analyzed to extract the descriptive features in the same way as the source data was treated. For each corresponding target units, the program retrieves a source unit called as the BMU, which has the most similar characteristics from the database. The neighboring target unit is also examined to retrieve the BMU from the source database, and the retrieved units are concatenated successively. The neighboring units have an overlap of a hopsize and the signals are convoluted with a cross-faded effect by a window function to avoid clip noise throughout the synthesis process (see the overlap method in the Audio Analysis Section).

Figure 84 illustrates the process of the basic CSS technique. The target audio data is segmented into units $(v_0, v_1,)$ where v represents its feature vector, and their BMUs

$(v_0, v_1,)$ are retrieved from the database. The retrieved units are then concatenated at the same position of their corresponding target units, and eventually, all units of the target is replaced by the sources. Figure 85 illustrates the comparison between the normal and overlap methods where the hopsize is half of the unit length.

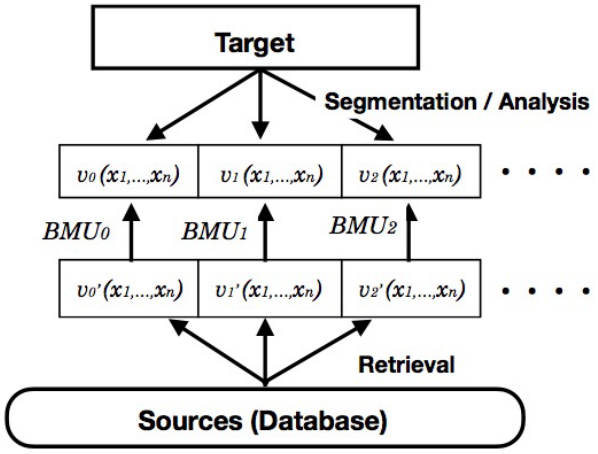


Figure 84: Basics of the Concatenative Sound Synthesis technique

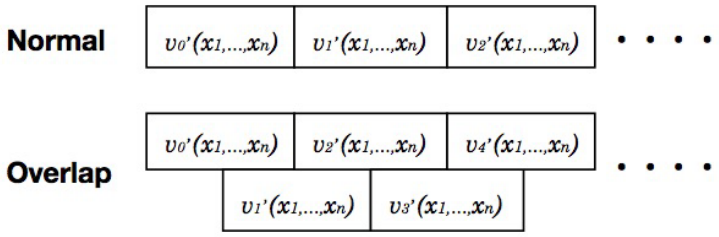


Figure 85: Comparison between normal and overlap method

23.3 Test of the frame-based CSS

The number of employed features significantly influence the quality of the synthesis results because they become criteria for the unit selection. For instance, when using only the MFCCs, the synthesis reproduces the timbral characteristics of a target, and it omits the Magnitude or Pitch information. An insufficient number of features may result in a reduced representation of the target. However, the employment of a larger number of features may cause erroneous unit selection in consequence of inefficient audio

identification. In order to tune the synthesis system, we need to know the associations between criteria and results. In this section, we will discuss the evaluation of the synthesis results in order to reveal the impact of features on the results.

In order to apply the CSS technique to creative purposes, the synthesis result needs to be both mathematically precise and perceptively adequate. For this reason, we examined the quality of the resultant sound from two different perspectives; computer-based analysis, and analysis by our perception. Yet, these two methods need to be carefully conducted as they are not always compatible with each other, and sometimes they come up with different conclusions. Particularly in the perceptive analysis, we need to observe the result knowing that the frame-based CSS is the most primitive and limited technique which is capable of reproducing a monophonic representation by using small chunks of audio data which is often heard as mosaic sounds.

We have tested the synthesis under several conditions using different descriptive features as shown in figure 86. We employed MFCCs, Magnitude, Centroid, Spread, Pitch, and their delta-features. From here, when we say Magnitude always means a set of the Magnitude and its delta-feature. The frequency and time structure of the results are visualized as the spectrograms and are also audible by the provided sound files. In the panel A[14], some regular overtones are observed on the metallic sound as pointed by the arrows, and intense peaks appear on the simultaneities of the snare, and bass drum sounds at around 1 and 3 seconds. The musical events consisting of sole instrumental sound has sparse spectrum, and in contrast, the mixed sounds have dense spectrums. Although these visual characteristics are vague in the synthesis results, we can still detect slight regularity in the panel B which utilized only MFCCs. Listening to the audio file of B[15], the original rhythmical pattern is hardly perceptible because the MFCCs does not effectively represent the dynamics and pitch information. In fact, the result of C[16] using both MFCCs and Magnitude presents clearer rhythmical structure which is also visible in the panel C. The representation of the pitch and brightness are improved in D[17] but it contains discordant sounds at the beginning, at around 1 sec, and at the end. These erroneous matchings are improved when adding MFCCs as heard in E [18] which employs all features.

[14] Drum_Loop1.aif
[15] audio/SynthesisResult/Unit/PercussionDatabase/B_MFCCs/DrumLoop_2048_1024.aif
[16] audio/SynthesisResult/Unit/PercussionDatabase/C_MFCCs_Mag/DrumLoop_2048_1024âif
[17] audio/SynthesisResult/Unit/PercussionDatabase/D_MagCentSprPitch/DrumLoop_2048_1024âif
[18] audio/SynthesisResult/Unit/PercussionDatabase/E_MFCCsMagCentSprPitch/DrumLoop_2048_1024âif

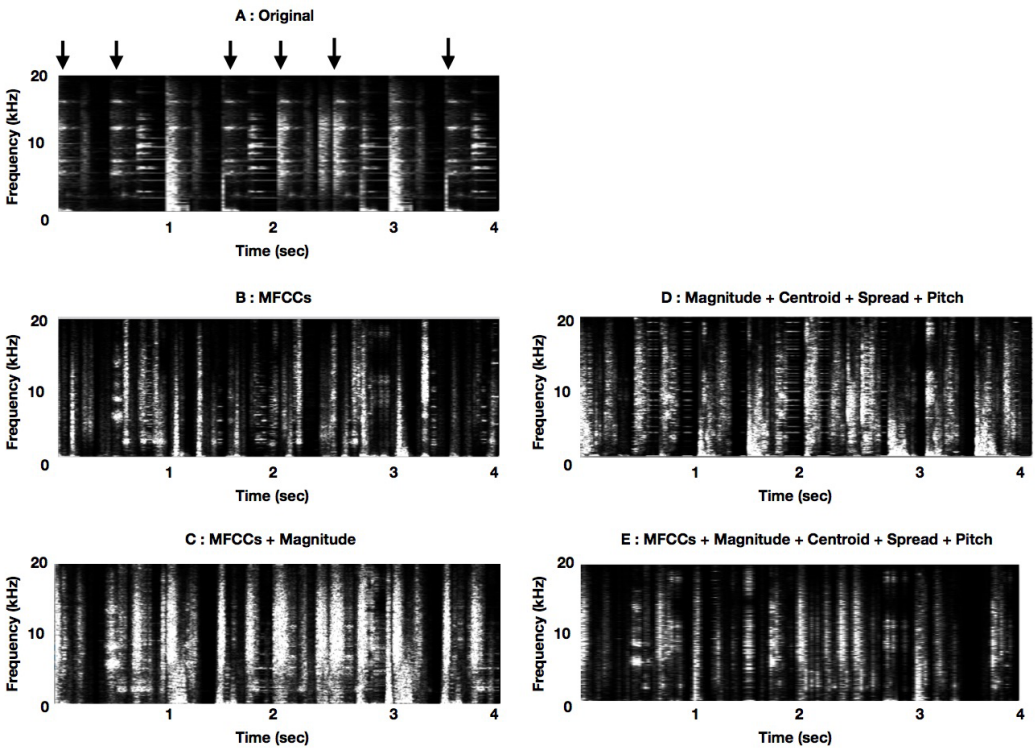


Figure 86: Spectrograms of the frame-based CSS results using Percussion Database

The panel A illustrates the spectrogram representing the frequency and time information of the target audio data and the B to E illustrate those of the synthesis results operated by using different descriptive features. The arrays shown in A indicate the metallic sounds which have some regular overtones. The analysis and synthesis are executed in the FFT frame size of 2048, the hopsize of 1024, and 10x10 SOM.

We discerned that the MFCCs enhance the low-frequency representations to a considerable extent by combining with other features. While the panel B contains a lot of erroneous events in the low-frequency bands which make the rhythmic structure unclear, panel C which is accompanied with Magnitude feature, significantly improved the distinction between low and high-frequency events (listen to the audio samples). The panel D, which does not employ the MFCCs, again contains unnecessary heavy sounds, and the panel E using all features realizes satisfactory source selection in which we observed metallic, snare, and bass drum sounds.

The similar tendency is observed in the result acquired by using another database consisting of various human voices with rhythmical accents as shown in figure 87. The B' [19] shows unclear rhythmical structure though it is improved by adding the Magnitude

[19] audio/SynthesisResult/Unit/DanceVoiceDatabase/B_MFCCs/DrumLoop_2048_1024.aif

as shown in the C'[20]. The rhythmical structure is sufficiently represented in the D'[21] and the E'[22] which has the most stable timbral character of all.

The synthesis results of the human voice database suggest other interesting aspects of the descriptive features. In most cases, the consonant segments play a significant role representing the percussive sounds, and the vowels give a sense of pitch. The vowel is seldom used independently but often accompanied by the consonants. Although the Magnitude is the most significant feature to determine the rhythmic patterns, other features are as important to enhance its quality by imitating the accents and intonations of the target. For instance, the unit selection of voice phenomena is strongly constrained by MFCCs because they are capable of identifying the consonants and vowels as applied in the speech analysis. The results of C' and E', which employed MFCCs, have closer timbral qualities to the target particularly in the high-frequency events such as hi-hat than the result D' which does not employ MFCCs. Especially in E', the hi-hat, snare, and bass drum sound colors are adequately imitated by voices, and thus, the rhythmic structure is apparently perceptible. Although D' have less similar timbral qualities, the selected units still remind us of the original sound characters due to its analogy of the pitch and brightness. These pitch fluctuations give us a sense of intonation and accent that also contribute to the construction of the rhythmical structure. This result suggests that these features also have significant impact on representing the target sound units for the human perception.

In contrast, the Violoncello database does not yield satisfactory results in any combination of the features due to the lack of matching sound. Figure 88 presents the resultant spectrograms produced by using different features. Compared with other databases, the quality of the high-frequency events are inferior to all others. Some hi-hat sounds are imitated by a col legno battuto, but they are not sufficiently present. Remarkably, B"[23] shows that it is too rough to find any associations with the target. The Magnitude feature contributes significantly resembling the rhythmic patterns as seen in C"[24], and frequency related features organized the low to middle-frequency events as shown in the D"[25] and E"[26].

[20] audio/SynthesisResult/Unit/DanceVoiceDatabase/C_MFCCs_Mag/DrumLoop_2048_1024.aif
[21] audio/SynthesisResult/Unit/DanceVoiceDatabase/D_MagCentSprPitch/DrumLoop_2048_1024.aif
[22] audio/SynthesisResult/Unit/DanceVoiceDatabase/E_MFCCsMagCentSprPitch/DrumLoop_2048_1024.aif
[23] audio/SynthesisResult/Unit/VioloncelloDatabase/B_MFCCs/DrumLoop_2048_1024.aif
[24] audio/SynthesisResult/Unit/VioloncelloDatabase/C_MFCCs_Mag/DrumLoop_2048_1024.aif
[25] audio/SynthesisResult/Unit/VioloncelloDatabase/D_MagCentSprPitch/DrumLoop_2048_1024.aif
[26] audio/SynthesisResult/Unit/VioloncelloDatabase/E_MFCCsMagCentSprPitch/DrumLoop_2048_1024.aif

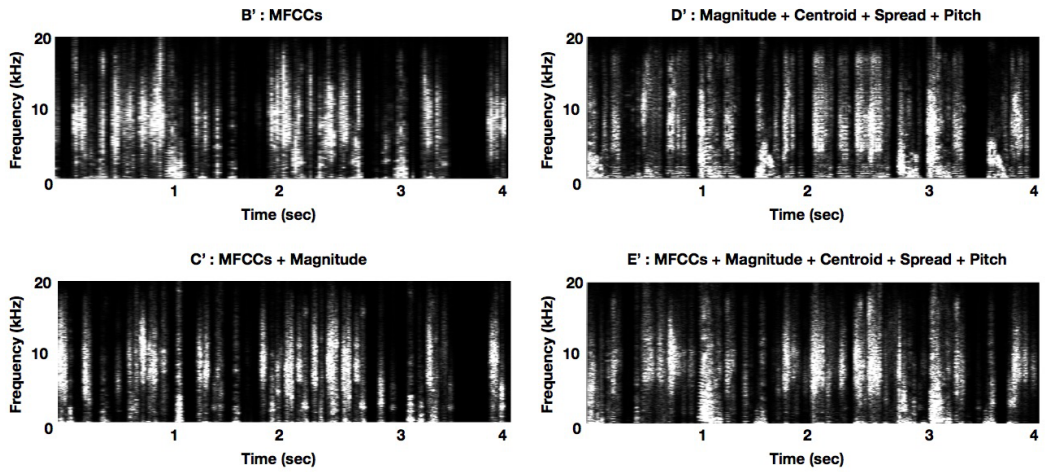


Figure 87: Spectrograms of the frame-based CSS results using Voice database

The panel B to E illustrate the spectrogram of the synthesis results operated by using different descriptive features. The analysis and synthesis are executed in the FFT frame size of 2048, the hopsize of 1024, and 10x10 SOM.

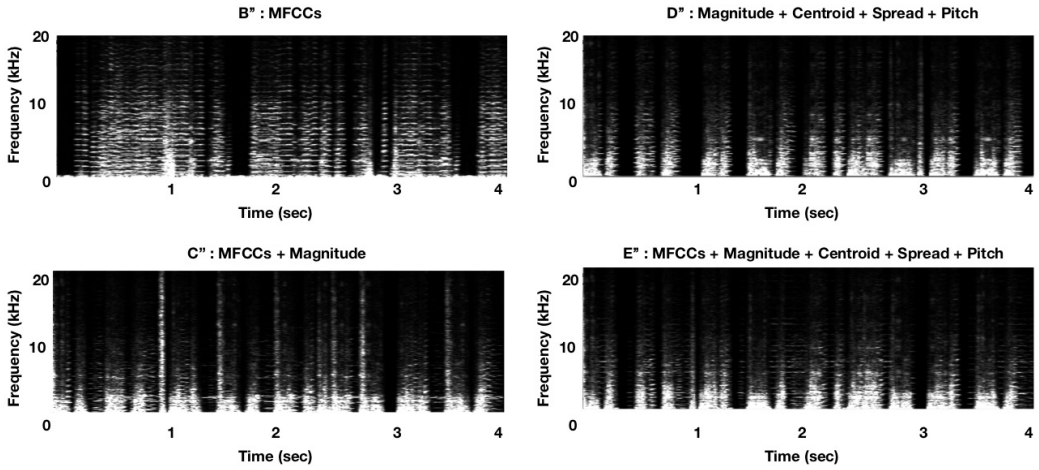


Figure 88: Spectrograms of the frame-based CSS results using Violoncello database

For further investigation of the synthesis results, we have processed the spectral analysis of the resultant audio data to extract the features which suggest us objective measurements of the quality. As our databases have limited samples, it is not realistic to expect all features to match rigidly. Therefore, we focus on observing the rough envelopes of the fluctuation through time instead.

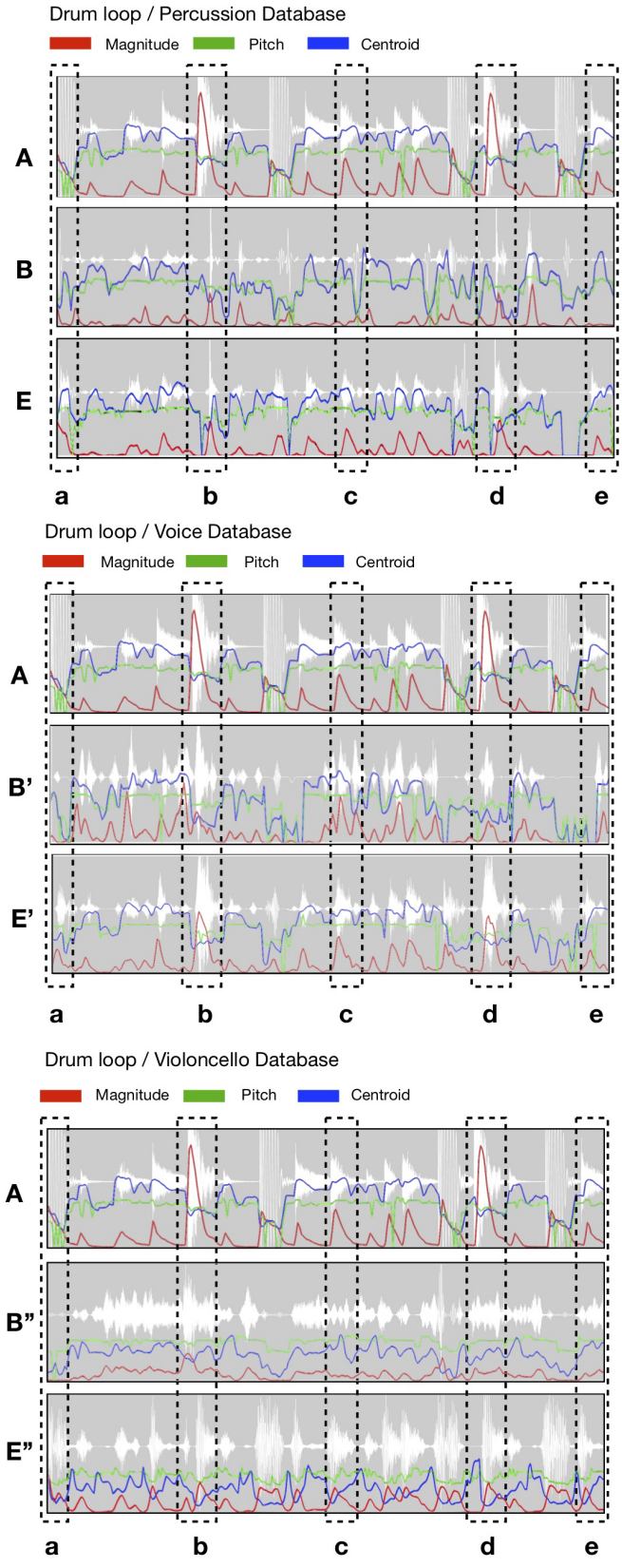


Figure 89: Analysis of the synthesis results

In figure 89, the panel A illustrates the analysis results of the target, the panel B, B', and B" are for the synthesis results acquired by using the MFCCs, and the panel E, E', and E" by the combination of the MFCCs, Magnitude, Centroid, Spread, and Pitch(MIDI). The capital letters identify the corresponding synthesis results shown in figure 86 and 85. The colored lines represent the descriptive features varying in time, and their remarkable variations are marked by dotted squares identified by small letters a to e , which indicate major sound events with high dynamics. We examined three databases consisting of samples from Percussion, Voice, and Violoncello samples.

The analysis suggests that the synthesis results of the Percussion and Voice databases successfully represent the characteristics of the target. The panels B and B' show proper features taking the similar trajectory to those of A, though some irregular fluctuations are still observed in Magnitude. The results suggest that the MFCCs are not sufficient to represent complete characteristics of the target and that the other additional features

are required for further improvement. The panel E and E' which employ more features illustrate more appropriate results compared with B and B'. Some peaks of the Magnitude witnessed around crucial sound events marked by *a* to *e* are adequately reproduced, and the fluctuations in the Pitch and Centroid are approximated too. These two databases are prospective to produce satisfactory results because the Percussion database contains the samples from the similar category of the target, and the Voice database has a wide variety of consonant sounds with various accents which adequately match percussive sounds. Further investigation of the feature transitions enclosed by dotted squares reveals the detailed quality of the representations. For instance, the feature transitions marked by *b* to *e* in the panel E' illustrates similar tracks to those of target. We can also audibly perceive the similarity by listening to the audio result[27].

On the other hand, the Violoncello database yields a lot of inappropriate fluctuations in all features. The quality of the panel B" is too rough to be recognized as producing any associations with the A. Although E" improves in Magnitude, hence better rhythmic structure, it does not yet represent sufficiently the character of the target. We have also tested this database with different target sound consisting of low-frequency events to check whether its poor representations are certainly caused by the lack of matching source units.

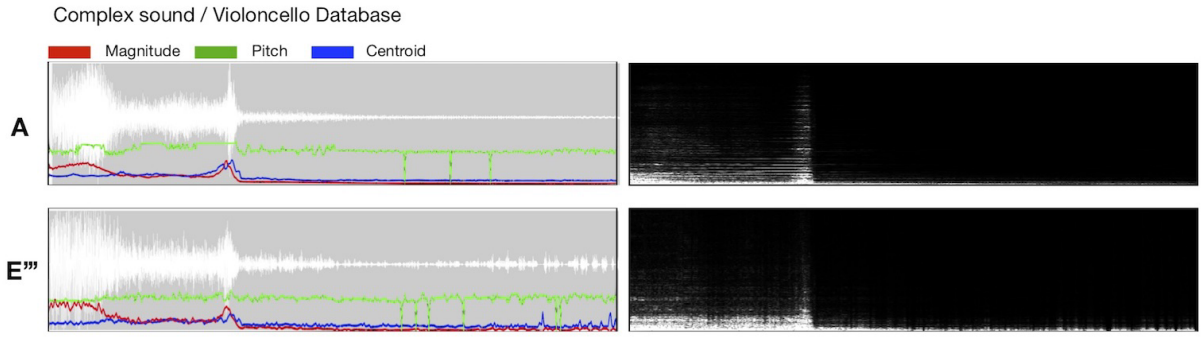


Figure 90: The analysis of the synthesis results using Voice database

The panel A illustrates the spectrogram of the target and E"' illustrates the synthesis results operated by using MFCCs, Magnitude, Centroid, Spread, Pitch(MIDI), and their delta-features. The analysis and synthesis are executed in the FFT frame size of 2048, the hopsize of 1024, and 10x10 SOM.

[27] audio/SynthesisResult/Unit/DanceVoiceDatabase/E_MFCCsMagCentSprPitch/DrumLoop_2048_1024.aif

Figure 90 illustrates the waveform and analysis results of the target sound `ComplexSound.wav` which has a sweep electronic sound gestures from the low to middle frequencies, and its synthesis result E"' with the Violoncello database. We detected significant similarity in the trajectories of all features between A and E"', and as well sufficient resemblance in the waveforms and spectrograms. However, the resultant sound[28] does not represent the target very well as proved by the analysis results. Since the result focuses on representing only the low-frequency event, it does not adequately reproduce high overtone gradually emphasized throughout the sweeping effect in the target.

Although the frequency information of the sweep effect is one of the most impressive factors from our perception, it was not considered as well serious by the computer analysis, which is why we observed an estrangement between two analysis methods. This problem is also overlapped with a polyphonic representation issue since the sweep sound can be divided into a low bass sound and a sweeping high frequency. We will go into details in later sections.

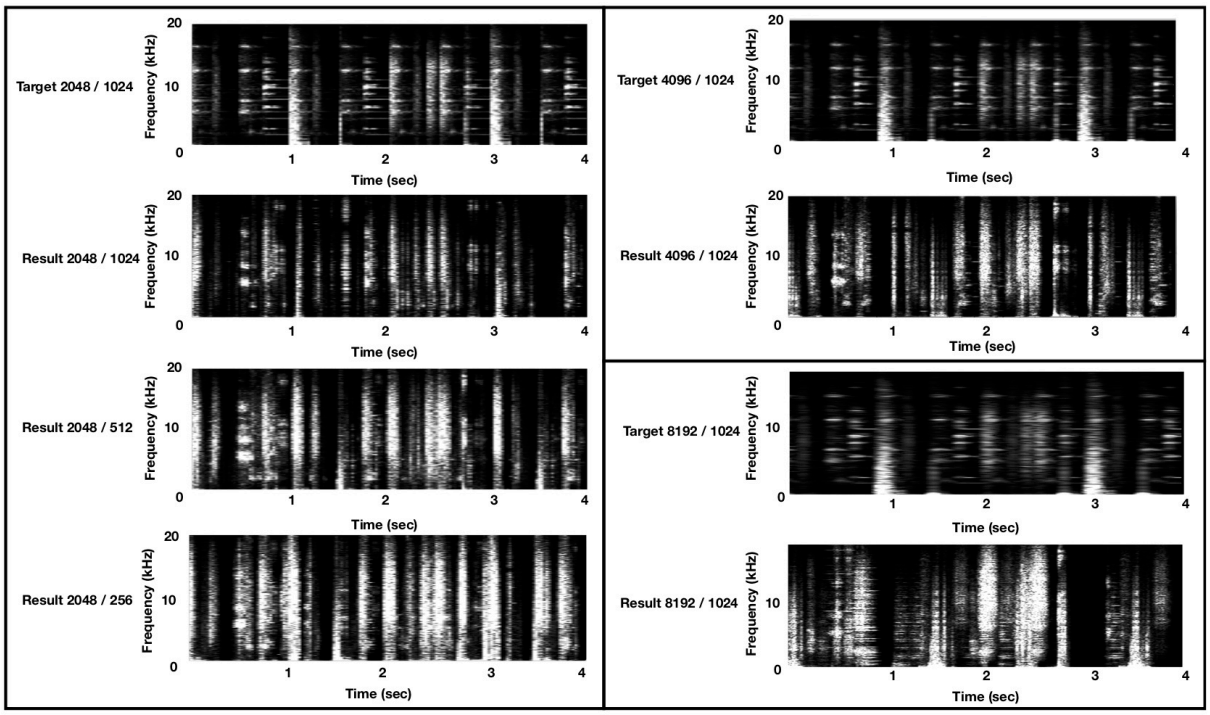


Figure 91: Spectrograms of the frame-based CSS results using Voice database

The spectrograms of the synthesis results by the FFT size of 2048 with different hopsize in left, and the FFT size of 4096 in the upper right, and the FFT size of 8192 in the bottom right.

[28] audio/SynthesisResult/Unit/VioloncelloDatabase/E_MFCCsMagCentSprPitch/ComplexSound_2048_1024.aif

23.4 The synthesis result with different FFT frame sizes

The FFT frame size and hopsize constrain the time-localization of the synthesis results. Since a larger FFT frame size holds more extended audio signals, it is capable of producing more natural synthesis results than that of the smaller one. However, it declines the time-localization of musical events, so that it is not suitable for a sound which has a detailed rhythmical structure. The smaller hopsize is more likely to improve the problem, but it does not deal with it at a fundamental level. We will present the association between these parameters and synthesis results.

Figure 91 presents the spectrograms of the target and synthesis results acquired by different FFT frame size and hopesize. With a small hopsize, the spectrum of each event got thicker and more dense because many signals are overlapped. The rhythmical structure is present and hence stable, but the character of each event is unified into full spectral sounds which cause a strong sense of mosaic" texture. While a bigger FFT frame size improved the timbral representations but results in an inaccurate rhythm, it is prospective to reproduce longer musical gestures as shown in `complexSound.wav`. The synthesis result[29] is produced by the FFT frame size of 8192 and hopsize of 2048, which is comprised of the combination of tremolo and strong bowing sounds. Compared with the synthesis results by the FFT frame size of 2048 and hopsize of 1024[30], it presents clearer characteristics of each unit. However, it is still perceptible as mosaic texture because the selected units are segmented mechanically into the FFT frame size, and there is no connection between the adjacent frames.

For further improvement, we need to advance the unit selection system to extract successive units from a source data instead of selecting individual unit which will be discussed in the next section.

24 Advanced algorithm for creating a concatenation

The process just described, that is, simply synthesizing segmented sound data retrieved from a database, is not useful to generate a practical synthesis result for a chamber music composition due to its mosaic textures. As mentioned earlier, the length of each unit is between 11 and 200ms and consequently barely reproducible with instruments, besides very short segments of audio data are heard as a clipping noise, and their sound

[29] audio/SynthesisResult/Unit/ViolonchelloDatabase/E_MFCCsMagCentSprPitch/ComplexSound_8192_2048.aif
[30] audio/SynthesisResult/Unit/ViolonchelloDatabase/E_MFCCsMagCentSprPitch/ComplexSound_2048_1024.aif

character is hardly recognizable. To tackle this issue, and in order to construct a proper concatenation, we developed an advanced algorithm to retrieve as many continuous successive units from the source data as possible. When the program finds the BMU for the nth unit of the target data to be the mth unit in the source data, it also refers to $(m+1)$th and $(m-1)$th units in the source data and $(n+1)$th and $(n-1)$th units in the target data in order to measure their similarities. If the $(m+1)$th or $(m-1)$th units and $(n+1)$th or $(n-1)$th are similar enough, the program employs $(m+1)$th or $(m-1)$th units together with the mth unit as one sequence in the creation of a concatenation. Subsequently, the program also refers $(m \pm 2)$th, $(m \pm 3)$th, and so on. As long as the adjacent units keep a certain level of similarity to the corresponding target units, the program continues to refer to the next or previous units, and as a result, it is possible to obtain a more natural sounding synthesis. Figure 92 illustrates the process of units retrieval and the matching process of the neighbor units.

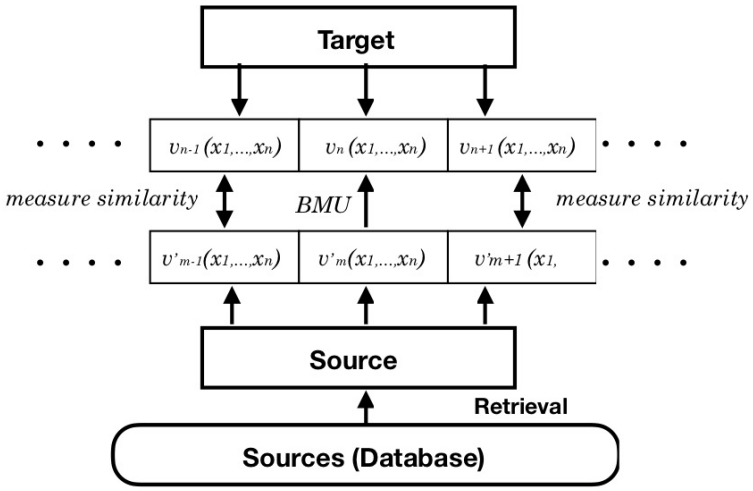


Figure 92: The neighbor units concatenation

The BMU, which is a unit of a selected source data, is retrieved from the database, and subsequently, the neighbor units of both the target and source are assessed whether they satisfy the condition. If they satisfy the condition, the source units are employed to create a concatenation, and the similarities of their neighbor units are again assessed. The concatenation tends to be longer when the target and the retrieved source consist of similar units.

24.1 k-NN method

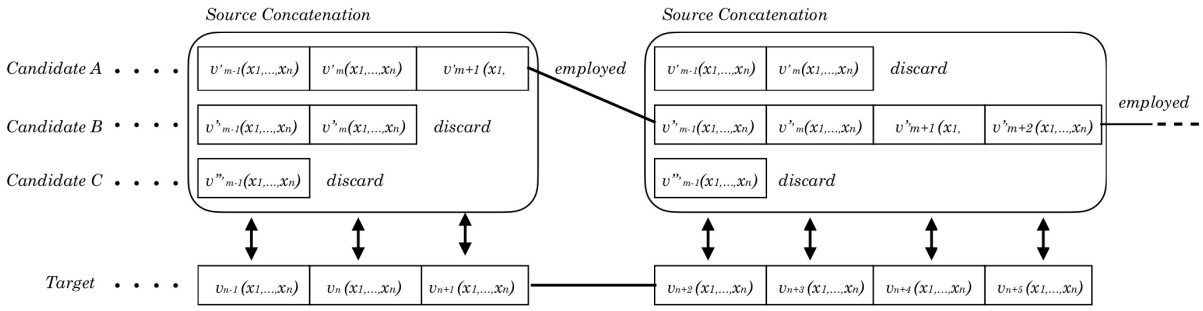


Figure 93: k-NN algorithm

Retrieving a sequence of successive units from the same source data does not only generate better recognizability of the source sound file in the synthesis result but also produces, in the sense of its playability by musical instruments, a more realistic musical idea. When the characteristics of a target file and multiple source files are much different, it is hardly possible to obtain a long sequence. This program evaluates several candidates for each target unit in order to increase the possibility of obtaining a longer sequence from the same source data. Firstly, k most similar units (one of them is the BMU) are retrieved as candidates from the database and a sequence of n-neighbors are assembled for each of them. Secondly, the program compares the k numbers of candidates and find out the candidate which has the longest n-sequence and employs it in the concatenative synthesis as shown in figure 93. This Nearest Neighbor algorithm is especially called as k-Nearest Neighbor (k-NN) algorithm. If two candidates having the longest sequence are obtained, the candidate which has the most similar vector of weights is selected.

24.2 Three Thresholds Algorithm

Throughout the process of k-NN algorithm, the program uses three thresholds in order to constrain the generation of sequences. Figure 94 illustrates how these thresholds function in the process. The vertical line represents the similarity between target and source units where the top shows the highest similarity and vice versa. The horizontal line represents the number of units in a sequence.

The first threshold constrains the BMU retrieval. If the program can not find a

BMU whose similarity is lower than the threshold, then it does not retrieve the BMU and leaves the frame empty. This threshold prevents to take a source unit whose characteristic is much further from the equivalent target unit, what would interfere with a good reproduction of the target file.

The second threshold constrains the length of sequences. It is not possible to expect to obtain a long sequence of successive units having high similarity to the target units if strict constraints are used. Therefore, the second threshold should be set softer than the first one. As long as the similarity does not become lower than the second threshold, the program continues to construct a sequence.

The third threshold is a trigger which decides whether or not the program should retrieve a new BMU. When the unit similarity is higher than the threshold, a new BMU, it just will not be retrieved, and the program just keeps on constructing the sequence. If the similarity gets lower than the threshold, the program tries to retrieve a new BMU from the database; if one is found, the program then re-starts the building of a new sequence. In the case the similarity remains higher than the second threshold but lower than the third threshold, the program retrieves another BMU and begins to construct another sequence. Therefore, multiple concatenations are simultaneously built and often overlapped to each other, and as a result, the program performs the synthesis polyphonically as shown in figure 94. The softer constrain of the second threshold produces more extended sequences what allows higher recognizability of the source and practicality for actual performance, however, the reproducibility of the target becomes compromised due to the lower similarity in each unit. It is therefore, significant to control these three thresholds appropriately according to the database and the intended purpose of the synthesis.

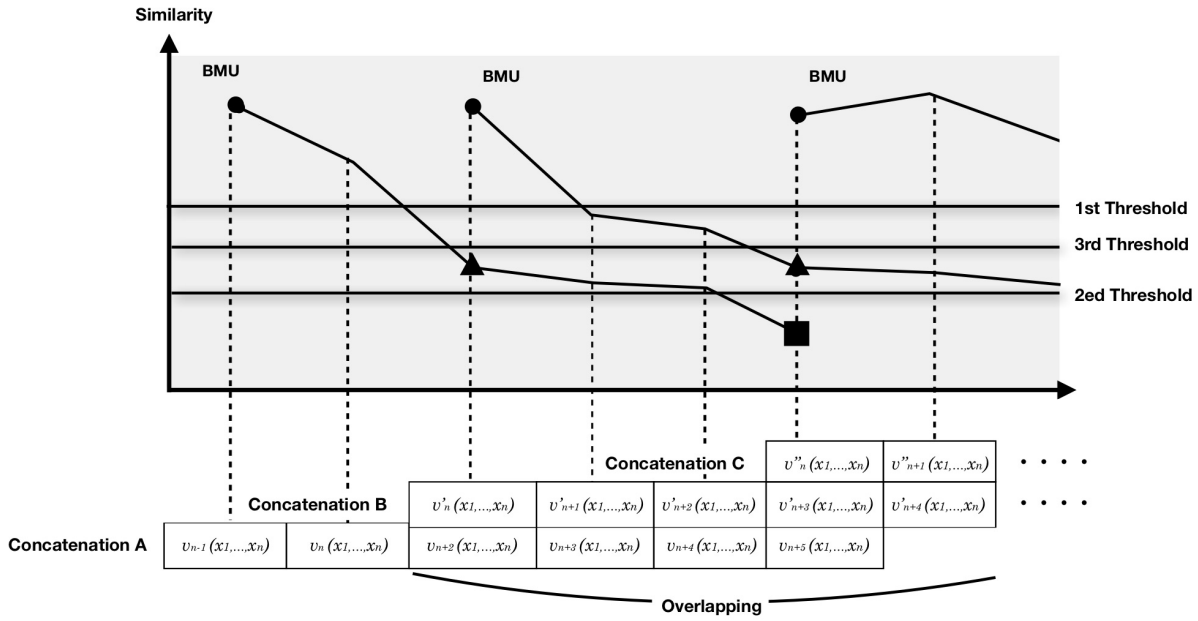


Figure 94: Three Thresholds algorithm

Knots of the poly lines are marked by circle, triangle, and square shapes which represent the BMU, a trigger of the new BMU retrieval, the end of the concatenation respectively.

24.3 The demonstration of our advanced CSS technique

We will introduce some synthesis results produced by our advanced CSS technique. The system allows changing weightings of the employed features so as to modify the criterion for the BMU retrieval and to constrain the construction of sequences by giving some parameters. For instance, in order to reproduce changes in dynamics in the target data more rigidly, we need to give more weightings to the Magnitude when calculating the distance between units. Should the pitch information of the target data be more important than other features, then giving more weightings to the Centroid and Pitch.

In the two threshold algorithm, the lower value of the first threshold restricts the BMU retrieval and the second threshold constrains the probability of generating a more extended sequence of units. As the impacts of the descriptive features were already discussed in the previous sections, we will here focus only on the threshold parameters which constrain the concatenation of the unit sequences. These criteria can be easily controlled by the Automation interface which will be described in the Graphic User Interface section.

We have used a sound file of `Drum_loop.aif` for the demonstration because it has

various characteristics which should make them easy to follow when re-synthesized. It includes clear dynamic changes, wide frequency range, clear rhythmic structure, and a combination of percussive attack and long resonance. Here, we have selected the Voice database which produced positive results in the previous test.

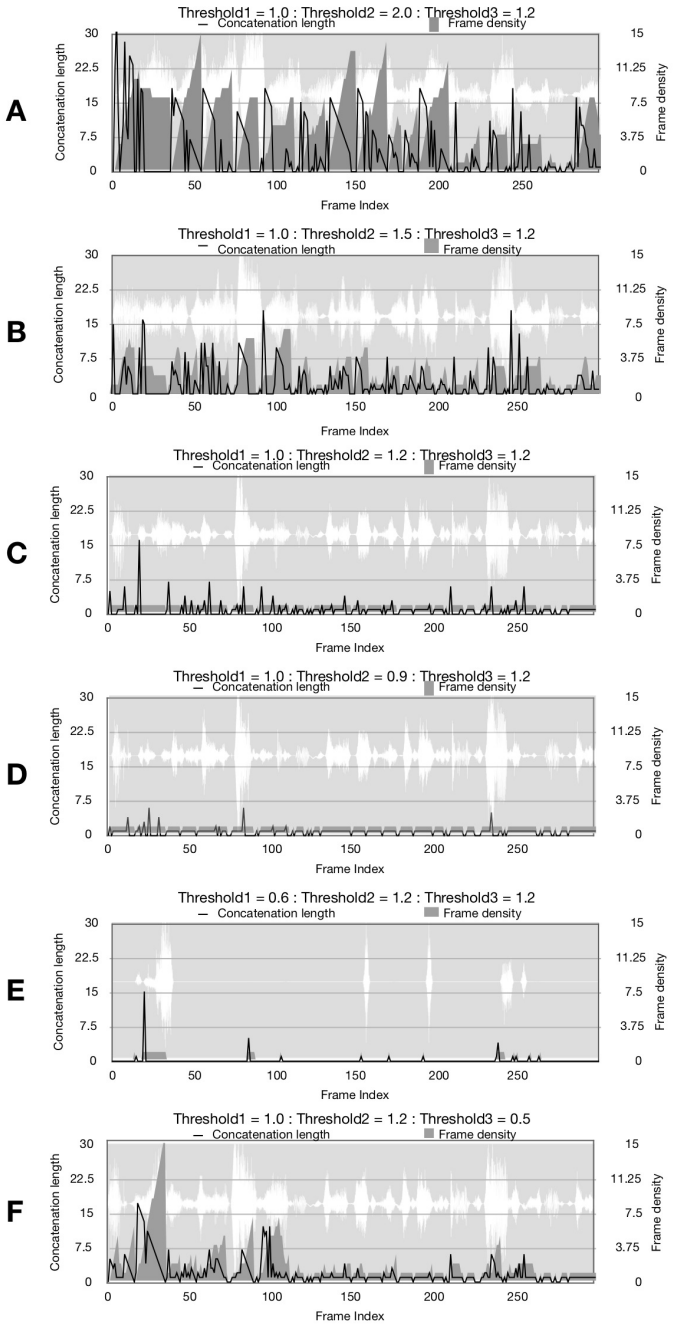


Figure 95: Analysis of the synthesis results

Figure 95 presents six panels which illustrate the test results acquired by different combinations of three thresholds. The list of values we have set for three thresholds in this test is described in the table 1.

The horizontal axis represents the frame index of the result which is corresponding to the waveforms attached to the background of the graphs. The left vertical axis for the polyline represents the length of unit concatenation constrained by the second threshold, and the right vertical axis for the solid sticks in dark gray color is for the frame density which shows a number of overlapped sequences on the corresponding frame. The sequence overlap occurs due to different timings of the start and end positions of the sequence controlled by the third threshold.

The synthesis result A[31] was acquired by a higher value of the second threshold which resulted in the extended unit sequences. We observed heavy simultaneities of var-

[31] audio/SynthesisResult/Unit/TwoThreshold/A.aif

ious voices for every corresponding target sound event. This effect blurred the original rhythmical structure and timbral characters. In the panel A, a polyline constantly stays in high values over ten to twenty frames which means that many successive frames became the starting points of generating the long sequence. In fact, the high frame density is detected over the entire panel which shows significant accumulations of sequences. We assessed that the second threshold in A is too soft to reproduce a satisfactory result.

Table 5: The list of the three thresholds

File name	Threshold1	Threshold2	Threshold3
A.aif	1.0	2.0	1.2
B.aif	1.0	1.5	1.2
C.aif	1.0	1.2	1.2
D.aif	1.0	0.9	1.2
E.aif	0.6	1.2	1.2
F.aif	1.0	1.2	0.5

The result B[32] was reproduced by the rigid second threshold in order to constrain the length of the sequence. We can hear tighter rhythms and more explicit contrasts between different sound events than A. This consequence gets more obvious as decreases the threshold.

The second threshold in C[33] reproduced an adequate result by constructing an appropriate length of sequences. The result D[34] made rhythms more rigidly, but it presented less clear timbral characters of the selected units as a consequence of the discontinuity of source sound. The panel D illustrates that most of all chosen units are merely synthesized without generating any sequences.

The panel E illustrates a result by a rigid value of the first threshold which discarded a considerable number of BMUs due to unsatisfied similarity to the corresponding unit of the target. As shown in both panel and audio sample[35], we observed sparse units and the reproduction failed obviously.

F is examined with a low value of the third threshold which provokes more frequent BMU retrieval and its sequence construction, whereas other thresholds are the same with C which produced a moderate length of unit sequences and their overlaps. Panel F shows longer sequences than those of C, and they are overlapped around the crucial sound events.

[32] audio/SynthesisResult/Unit/TwoThreshold/B.aif
[33] audio/SynthesisResult/Unit/TwoThreshold/C.aif
[34] audio/SynthesisResult/Unit/TwoThreshold/D.aif
[35] audio/SynthesisResult/Unit/TwoThreshold/E.aif

While the excess sequence accumulations produce a full spectral timbre which declines the contrast between distinct sound events, its adequate amount can contribute to gaining the quality of the timbral representation.

The quality of the unit sequence is strongly influenced by the characteristics of the selected BMU more than its adjacent units. When the corresponding unit of the target contains a dramatic change of the sound in a short period or consists of polyphonic sound events, its features tend to fluctuate significantly. Since the FFT analysis only reveals the averaged character of the short signal, it tends to give priority to a superior component. However, throughout the time sequence, the presence of the sound events often alternates with each others depending on the duration of their decay and sustain properties. As a result, the features of adjacent frames possibly correspond to the distinct sound events. For instance, in the drum loop example, we hear at least three different events simultaneously at the beginning, that are, the bass, snare, and hi-hat sounds and their domination change in short time. In the analysis result in figure 89, we observed that the snare and hi-hat sounds are taken over by the those of bass drum at the beginning, however in the subsequent frames, they are dominant in the analysis result even the resonance of bass drum is present. In this case, the third threshold should start constructing another sequence corresponding to the snare and hi-hat sounds once there are superior BMUs available, and the sequence of the bass drum should stay due to the soft second threshold. Therefore, the other sequences generated from the neighbor units can interpolate the character having been omitted by the previously constructed sequences.

25 Segment Based Concatenative Sound Synthesis

For another solution to enhance the recognizability of the selected source units, we have developed an advanced synthesis technique. The Segments or Sequence-based CSS (SCSS) processes a sequence of FFT frames as a minimum unit instead of individual frame retrieved from a database. The SCSS is considered prospective to produce more natural synthesis results, particularly its capability of avoiding the mosaic texture is what should be noted because it employs audio sequences which are expected to be separated into individual musical events. In the SCSS, it is more complicated to find the Best Matching Sequence (BMS) than the BMU because a sequence has more

information to be identified by features than a FFT frame. Since a sequence consists of multiple frames, the database will have fewer candidates compared with the one for the frame-based CSS and thus, the database for the segmented sources has less probability of finding a satisfactory MIR result.

When dealing with a sequence of frames as a minimum unit of synthesis, we need to describe its time-varying information. One of the easiest ways would be to apply a Markov Chain consisting of a set of features of each constituent component. However, a long sequence becomes a critical obstacle to establish an efficient classification model and the straightforward pattern matching system because it requires a tremendous number of features. As already suggested in the Classification section, operating a regression analysis extracting the differential coefficient (DC) can be a solution. The DC can represent a statistic transition of the features over a segmented period instead of using a chain of features as the Markov Chain does. We can significantly reduce the number of features by labelling a set of feature of its head frame and the gradients which represent how each feature varies over the segmented area. With this method, we can represent a sequence with at least as twice as many numbers of features that the frame-based CSS used for each unit regardless of its length. Figure 96 and 97 illustrate the BMS retrieval process by means of the Markov Chain and the DC respectively. Our SCSS technique also assesses the length of sequence between target and source candidates besides considering the similarity of their features, and it attempts to retrieve an adequate length of the sequence to the corresponding target sequence as well as a satisfactory feature similarity. The length of the source sequence is most likely to become shorter or longer than that of the target when it does not correspond, and it produces overlapping or empty frames in the result.

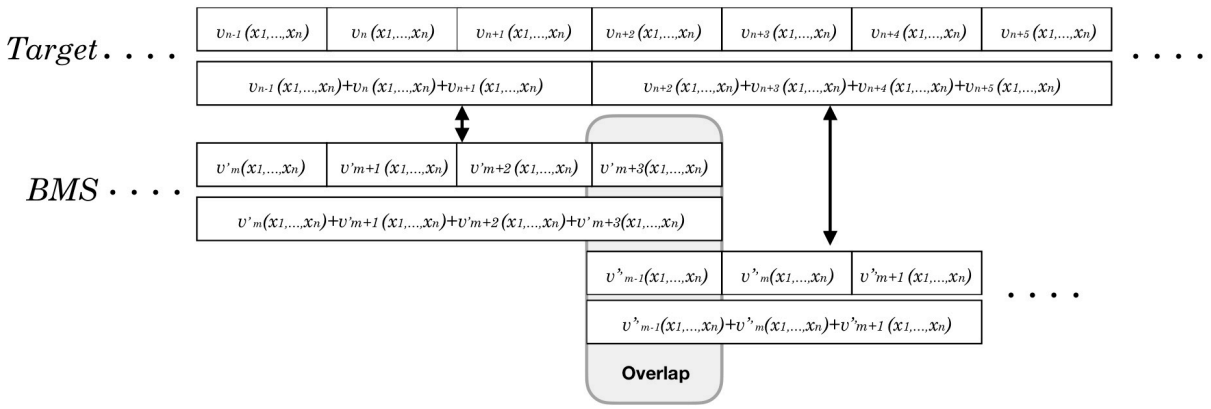


Figure 96: A SCSS using a sequence labelled by Markov Chain

The number of features of a sequence corresponds to the number of units multiplied by the number of features each unit has.

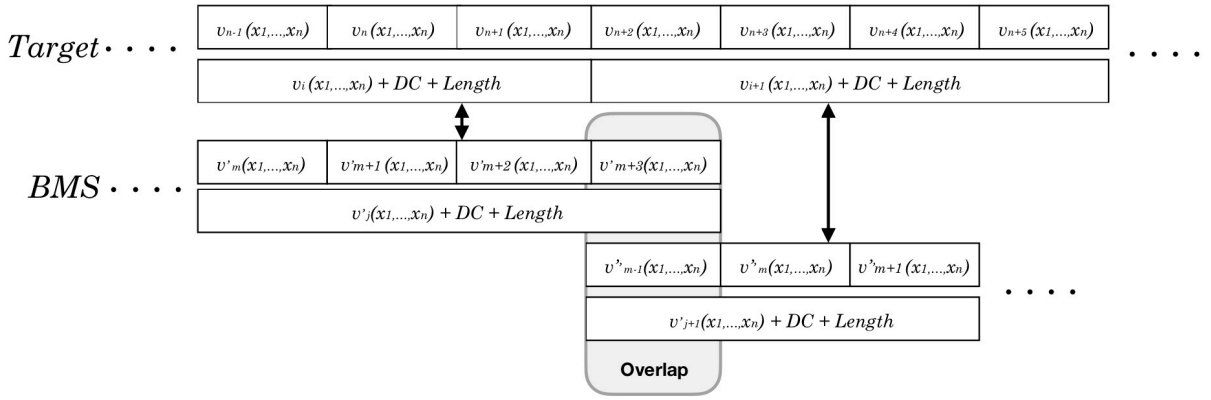


Figure 97: A SCSS using a sequence labelled by the Differential Coefficient (DC)

The number of features of a sequence corresponds to the number of features the head frame, the same number of DC, and the sequence length. The number of total features are significantly reduced for a longer sequence.

25.1 Overlapping Segmentation Algorithm

In this synthesis technique, the segmentation manner is the most significant key to reconstruct target segments because it determines the resolution of the synthesis in the time axis. The segmentation algorithm we have already discussed in the Segmentation section employs two thresholds which constrain the separation of each musical event from its adjacent events without any overlaps. In this algorithm, the beginning position of a new segment corresponds to the end position of the previous segment.

As discussed in our advanced CSS technique, the real world sounds contain multiple sound events overlapped to each other, and the beginning and end points of the events are seldom simultaneous. It is almost impossible to separate those of overlapped sound components and isolate every single event by the segmentation algorithm, and it may easily omit the events coinciding or at a slightly different timing to the detected one. Not to omit the intensive sound variations, we need to make the segmentation algorithm sensitive for small changes. However, a too sensitive algorithm may separate a sound into too small segments, which will be eventually similar to the frame-based CSS. Therefore, we need to design an algorithm which separates a sound into detail events as well as extracting them in long sequences.

In our advanced CSS algorithm, we have suggested the k-NN method and three thresholds algorithm which detect the onset of a segment independently on the off-set of its previous segment. Therefore, the unit concatenations are overlapped with their neighbor units which can interpolate missing characteristics of the target and thus, it is expected to reproduce a perceptively satisfactory result. Expecting a similar benefit, we have also applied the same idea to the sequence based CSS algorithm.

Our advanced segmentation algorithm is operated for the target only, and the source segmentation is done by the conventional algorithm discussed in the Segmentation section. The advanced algorithm has different criteria to detect the sound events and to define their length by referring different features. We defined three rules. The first constrains the sensitivity of detecting the onset of segments in which, a high value avoids detecting too frequent segment onsets and thus, it is immune to noise-like fluctuations. The second controls the detection of the segments cut-off. These two rules refer the delta-MFCCs. However, the delta-MFCCs is not adequate for detecting the segment cut-off when the sound decays into silence because it gives an enormous number of peaks due to its no immunity against noise. We have, therefore, employed the Magnitude to prevent detecting erroneous segmentation cut-off and to keep segments when a sound is sufficiently quiet because it does not produce a negative impacts on the synthesis result.

25.2 Magnitude Adjustment

Compared with the frame-based CSS, the SCSS is expected to realize less similar character to the target because it uses a longer sequence of frames without evaluating the

similarity of the individual frame. The Magnitude is a significant feature that declines the quality of the synthesis results among others, and thus, we have employed the Magnitude of the target to adjust the synthesis result. After producing the synthesis result, its Magnitude gets extracted by the spectral analysis, and the calculation of the dissimilarity ratio with that of the target is done. Let the Magnitude of the target M_n, and of the synthesis result M_n where the n represents the frame index, and the dissimilarity ratio $ratio_n$ is calculated by the following formula.

$$ratio_n = \frac{M_n}{M'_n} \tag{75}$$

The ratio is calculated for each FFT frame, and the same ratio is applied to a number of examples corresponding to the hopsize. The powers of a bunch of samples in synthesis result S_i where i is the sample index is then adapted as follows.

$$S'_i = S_i \cdot ratio_n \tag{76}$$

25.3 The Demonstration of the Sequence based CSS technique

Figure 98 represents the structures of the SCSS results on top and the analysis of the target, the SCSS results and the frame-based CSS of two targets; the complex gesture `complexSound.wav` and `Drum_loop1.aif` respectively.

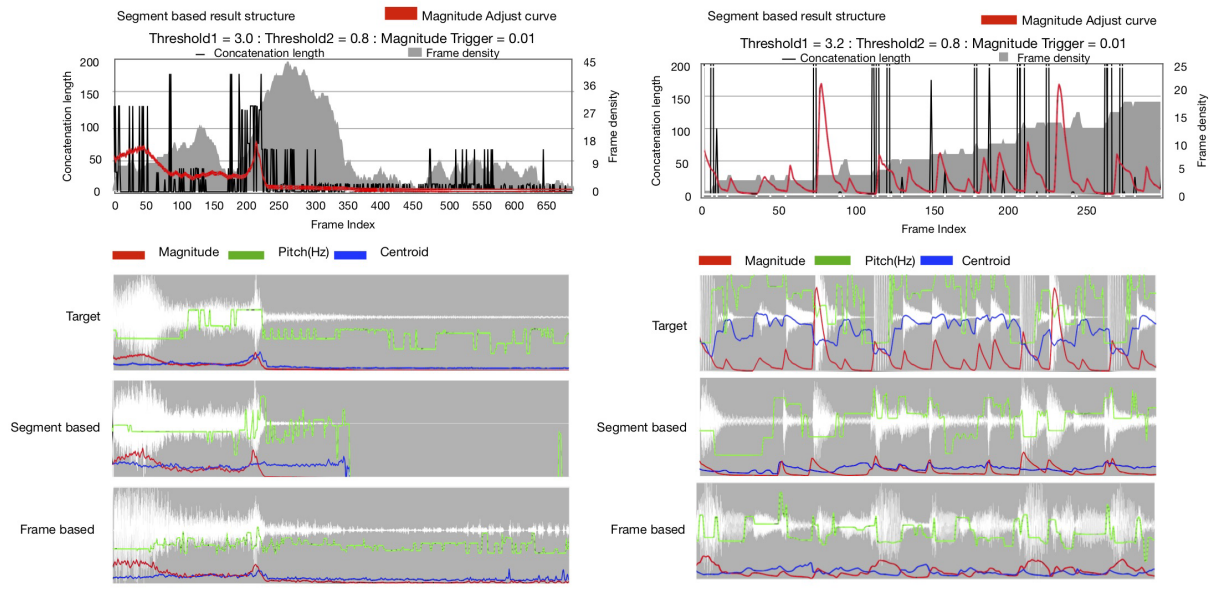


Figure 98: The SCSS results

In the segment-based result structure panels, the horizontal axis represents the frame index of the result. The left vertical axis for the polyline represents the length of unit concatenation constrained by the second threshold, and the right vertical axis for the solid sticks in dark gray color is for the frame density which shows a number of overlapped sequences on the corresponding frame.

Figure 98 represents the structures of the SCSS results(top), the analysis of the target, the SCSS results, and the advanced frame-based CSS of two targets; the complex gesture `complexSound.wav`(left) and `Drum_loop1.aif`(right) respectively. The left line shows the result using the Violoncello database and the right is for the Piano database. In both the frame-based and sequence-based CSS by the Violoncello database, we observed sufficient representations in the Magnitude and Centroid features and the moderate similarities in the Pitch. In the SCSS, some combinations of low and high pitch sounds are adequately reconstructed. The structure of the SCSS result(top) illustrates that the sufficiently long segments whose lengths are over 100 frames (the hopsize is 512 samples, and thus, a length of five frames corresponds to a length of double of the FFT frame size(2048 samples)) are successfully selected around crucial sound events, and the frame density increased as more segments selected. However, the high frame density does not directly show the intensive accumulation of sounds because some of the long sequences contain resonance in low dynamics which are not present in the result.

The audio result `ComplexSound\_2048\_512.aif`[36] shows consistent sound charac-

[36] audio/SynthesisResult/Segment/ViolonceloDatabase/ComplexSound_2048_512.aif

ters over an entire sound which presents tremolo and strong bowing in low and high pitches. Compared with the result by the frame-based CSS[37], the SCSS produced more natural reconstruction presenting clear characters of the sources.

Similarly in the results by the Piano database(right), the frame-based CSS[38] produced mosaic-like sounds whereas the SCSS[39] reconstructed the isolated piano sounds which can be easily reproducible by musicians. Remarkably, the SCSS did not discard the most of piano resonances declining to the silence due to the third segmentation rule referring the Magnitude feature. While the frame density increases over the entire result, the accumulation of the resonance is not perceptively significant, and it merely sounds like an effect of the sustain pedal.

26 Polyphonic Synthesis Techniques

The previously discussed CSS algorithms are based on the monophonic representation of the target, and they do not utilize the features derived from the decomposition analysis. Some algorithms resulted in polyphonic reconstruction but they are basically as a consequence of the sequence representation, and there is no theoretical evidence.

There are some studies to represent the polyphonic structure of the target as discussed in the section 22, but a prominent solution is not yet found. Our system follows the analysis result by the decomposition algorithms of the Specmurt and Non-Negative Matrix Factorization(NMF). However, these algorithms are not sufficient for all kinds of sound, and therefore, we need to adjust them with additional algorithms to a specific type of sound.

Here we will introduce three algorithms which are the Subtractive Spectral algorithm suggested by Hackbarth and the Specmurt based algorithm and the NMF based algorithm. We will only demonstrate the NMF based algorithm as others are our future works.

26.1 Subtractive Spectral Algorithm

The subtractive spectral algorithm uses Mel-frequency spectrum to represent the characteristics of a target and a corpus of sources[67]. It describes the time-varying information by chaining the spectrum, and some length of a chain is separated to extract a

[37] audio/SynthesisResult/Unit/ViolonchelloDatabase/E_MFCCsMagCentSprPitch/ComplexSound_2048_1024.aif
[38] audio/SynthesisResult/Unit/PianoDatabase/DrumLoop_2048_512.aif
[39] audio/SynthesisResult/Segment/PianoDatabase/DrumLoop_2048_512.aif

sound segment by the onset detection. In each of segment, the algorithm retrieves the BMU, and its time-varying Mel-spectra magnitudes are subtracted from the relevant region of the target's spectra. When the subtraction remains a considerable surplus of the targets Mel-spectra, the algorithm searches another BMU for the residual Mel-spectra. This process is repeated until the subtracted targets Mel-spectra is sufficiently suppressed. In this strategy, the first selected BMU has the most significant impact on the target representation as it matches the original targets Mel-spectra. On the other hand, the subsequent units are chosen to account for spectral energy which is not yet satisfactory represented and therefore, they tend to become softer as the energy of the Mel-spectra is reduced from the original one. This algorithm carries a complemental role to the frame-based or sequence based CSS technique as it adjusts the quality of the target representation by accumulating multiple segments polyphonically. This aspect of the algorithm is also similar to the winner-takes-all algorithm as the first retrieved BMU defines the quality of synthesis result dominantly. We can hear some examples generated by the Audio Guide software on Hackbarths web site[84].

Although the subtractive spectral algorithm is straightforward, it has a great perspective to enhance the quality of the frame-based CSS. However, this algorithm does not find the best combination of the source segments from a database. One solution would be to estimate the statistically best combination of the spectra and its time-varying amplitude and then operate the retrieval process for each decomposed component. Since the combination is calculated as a probability problem, it is expected to output a mathematically sophisticated decomposition result.

26.2 Specmurt analysis based Polyphonic CSS technique

The Specmurt analysis estimates multiple fundamental pitches by calculating a deconvolution between the spectrum and an optionally defined overtone pattern. The analysis is processed only for the target audio data in order to reveal its polyphonic structure, and then the BMU is retrieved for each decomposed component.

The Specmurt analysis estimates multiple fundamental pitches by calculating a deconvolution between the spectrum and an optionally defined overtone pattern. The analysis is processed only for the target audio data in order to reveal its polyphonic structure, and then the BMU is retrieved for each decomposed component.

Since the Specmurt operates is for an individual FFT frame, the resultant pitch

distribution needs to be dealt with a Markov chain so as to represent time-varying information. Some studies have suggested the transformation of the Specmurt result to MIDI number[22][23] which is an appropriate format to be chained. In the Specmurt analysis based polyphonic CSS, some sets of the MIDI pitch, overtone pattern, and the Magnitude become criteria to identify a target frame. When a target frame has more voices to be decomposed, then the number of set increases. Therefore, the number of features do not cohere throughout the entire target frames but varies depending on the character.

The extraction of the multi-pitches and the Magnitude from the target is also used in *Orchids* developed by IRCAM for automatic orchestration. We can see an enormous amount of notes in the figure 99 which illustrates the UI of the Orchids presenting the constituent partials of the targets spectrum information. These partials are utilized for retrieving the source segment from a database and reconstruct the target polyphonically.

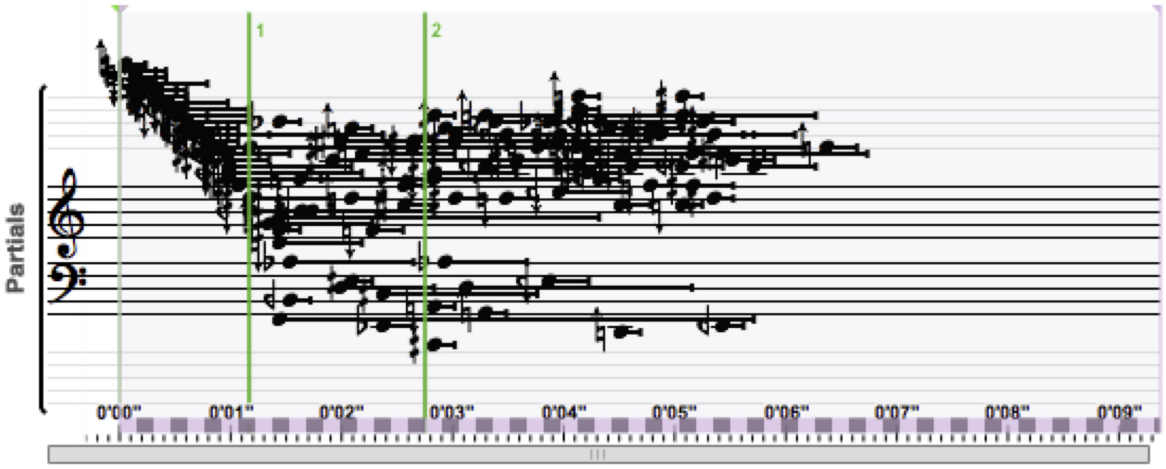


Figure 99: The UI of Orchids developed by IRCAM

The picture is quoted from the Orchids documentation[83].

Figure 100 illustrates the synthesis process using the Specmurt analysis. The analysis separates a frame of the target into multiple voices and yields the fundamental pitches and their magnitudes. Subsequently, the pitch distribution is separated into individual pitches and each pitch constituents a feature vector with its Magnitude and an overtone pattern. The analysis does not present any correlations between adjacent fundamental pitch distributions, and thus, the extracted voices are independent of each

other. We first need to associate the voices in order to create their sequences with an use of the Markov Chain. As discussed above, it is convenient to convert the pitches to MIDI number because it follows equal temperament which makes pitches easily grouped systematically. After the chain of feature vectors are made, we operate the same synthesis process as previously discussed. The synthesis results of each voice are mixed in the final process to realize a polyphonic representation.

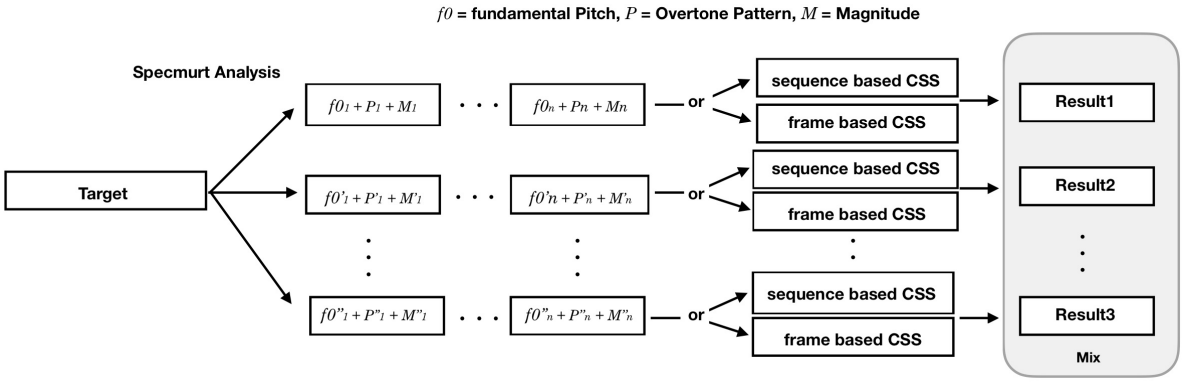


Figure 100: The process of the Specmurt analysis based polyphonic CSS technique

Each frame of the target is separated into multiple voices where, each of them is represented by the fundamental pitch, its Magnitude, and an overtone pattern. Some voices which have similar pitches are associated with adjacent frames to concatenate a sequence. The CSS or SCSS synthesis is operated for each concatenated sequence, and then all results are eventually mixed.

26.3 Non-Negative Matrix Factorization based Polyphonic CSS technique

The polyphonic CSS technique based on the NMF analysis follows a process similar to the one using the Specmurt analysis in the sense that both operate synthesis to individual voices independently. Unlike the Specmurt, the NMF can previously define a number of vectors to decompose, and it reproduces the resultant vectors as audio signals. Assuming that the vector signals are appropriate representations of the constituent components of the target signal, the conventional CSS technique can be directly applicable to vectors without extra algorithms. Figure 101 illustrates the process of the NMF based CSS technique where we operate the Sequence-based CSS technique for each vector. Since the vectors are already decomposed into smaller components, they can be separated more efficiently into the isolated sound events than the original target sound.

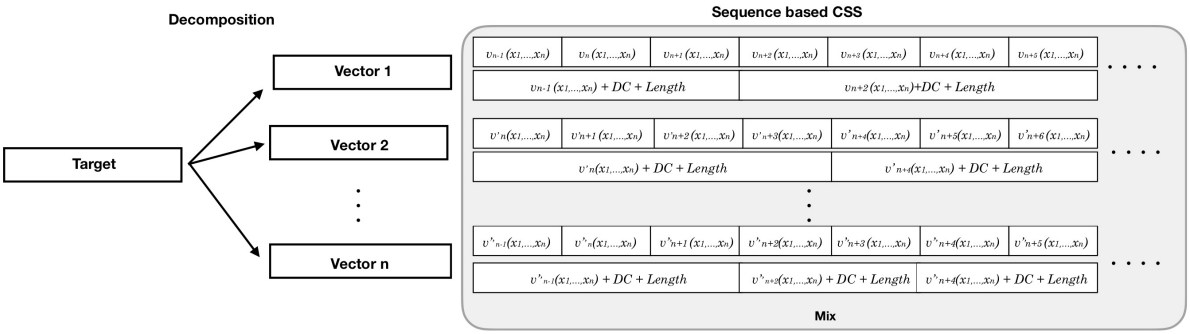


Figure 101: The process of the NMF based polyphonic CSS technique

Until here, it appears that the NMF based CSS can be easily implemented in our synthesis. However, in fact, the conventional NMF does not generate appropriate vectors to be applied to the CSS technique. One of the most significant problems is that the NMF re-synthesizes the vectors by a static spectral pattern and its time-varying amplitude calculated through the optimization process, and thus, the reproduced vectors lose their frequency information varying in time, but only show constituent statistic patterns satisfying the given probability problem. For this reason, the spectral analysis does not extract sufficient information from the reproduced vector signals, and thus, we do not obtain adequate features of the Centroid, Spread, Pitch, MFCCs, and their delta-features. Therefore, it is not so meaningful to process the conventional spectral analysis towards the vector signals, but we can only extract an exclusive feature from the static spectral pattern of vectors.

Furthermore, the optimization process often causes unintended results because the method is based on mathematical analysis, not perceptual coding. Therefore, the spectral pattern of the vectors is not necessarily practical to those of the real-world sounds, which becomes a significant obstacle to match the vectors and sources. The problem of this inconsistency of the optimized spectral patterns is discussed by Nakashima et al. [49]. In his approach, a statistic spectrum envelope pattern of the sources, which is called a probabilistic spectrum envelope (PSE), is previously trained before operating the NMF. Subsequently, the NMF performs the optimization of the vectors using the PSE instead of the conventional unsupervised optimization. Therefore, the target is decomposed to the vectors which have more likely similar spectral pattern to the sources that could adequately match each other.

There is another issue to be solved to obtain an adequate result. The conventional NMF algorithm is immune to small variations of the amplitudes, but it is very delicate to variations of frequency. When a sound sequence contains small pitch changes or smooth glissandi effect, the NMF intends to separate them into the distinct vectors regardless of their timbral character. This problem yields inefficient decomposition and, in consequence of producing an inappropriate synthesis result. The employment of the log-frequency spectrum is one possible solution as suggested in the Shift-invariant latent variable Model by Benetos and Dixon[47], and the Specmurt analysis. Implementing those of the advanced algorithms is yet to be done, and we need to address further study for more efficient and practical decomposition analysis, labeling method, and synthesis techniques in order to make the results satisfactory.

We will examine the synthesis by means of the present algorithms we have developed, and verify the above-discussed issues.

26.4 Demonstration of the NMF based SCSS technique

We will present two synthesis results using a Piano database and a Drum loop sound as a target decomposed into different numbers of vectors.

First, we separated the target into two vectors which are audible in two audio files, vector1[40], and vector2[41], and subsequently, we operated SCSS for each vector. The vector1 presents only the low and short bass drum sounds, and the vector2 represents hi-hat and snare sounds occurring in the high-frequency band. Remarkably, the vector2 contains also low-frequency components simultaneously with the high components. The SCSS result for vector1 consists of muted piano sounds which have fast decay, and the result for vector2 consists of various piano sounds from low to high registers. The mixed synthesis result is audible in the audio file[42]. In this result, the target was not sufficiently decomposed, and vector1 barely has signals, and vector2 has almost of all components of the original. We did not observe any significant differences from the conventional SCSS result. Therefore, the second test, we decomposed the target into four vectors; vector1[43], vector2[44], vector3[45], and vector4[46]. We could obtain the meaningful results

[40] audio/SynthesisResult/Polyphonic/test1/DrumLoopVector1.wav
[41] audio/SynthesisResult/Polyphonic/test1/DrumLoopVector2.wav
[42] audio/SynthesisResult/Polyphonic/test1/Mixed_2048_512.wav
[43] audio/SynthesisResult/Polyphonic/test2/VectorWave_1.wav
[44] audio/SynthesisResult/Polyphonic/test2/VectorWave_2.wav
[45] audio/SynthesisResult/Polyphonic/test2/VectorWave_3.wav
[46] audio/SynthesisResult/Polyphonic/test2/VectorWave_4.wav

only for the first[47], third[48], and the fourth[49] vectors, and no adequate sources were found for the second vector. Compared with the first test decomposing to the two vectors, we can hear more specific sound characters in each vector. The synthesis result for the first vector has low piano sounds corresponding to the bass drum, and the one for the second vector has high piano string scratching sounds for hi-hat, and high-register chords for the snare sounds. The result for the fourth vector represents some rhythms in the middle to low register. The mixed result is audible in the file[50].

When we used the percussion sounds as a target, we obtained an apparently satisfactory result, but when using harmonic or gestural sounds, we only obtained an erroneous result. Only when we use a percussion database, we could ostensibly obtain a slightly proper result. We here present a result using a short double stop fragment played by violin as a target[51], and a percussion database[52]. We concluded that these problems caused due to the previously discussed reasons.

27 Discussion

We have discussed how advanced CSS techniques were developed by combining with the additional algorithms. It is also noteworthy that we could add more varieties to the result by controlling the parameters for the creative purpose. While the simple frame-based CSS algorithm produced the mosaic-like sound result, the employment of longer sequences improved it more naturally and practically. However, the spectral analysis shows lower similarity as the length of a sequence gets longer and thus, the accuracy and the adequateness of the synthesis s result may contradict each other. The use of longer sequences is a significant factor to produce a practical result to be playable by musicians. Here, we have advanced our segmentation algorithm by referring the Magnitude feature, which let a sequence stay when it is sufficiently quiet. While this rule avoids the unnatural interruptions of the employed source sounds, it possibly generates excessive overlaps of the sources. Therefore, it is also important to control the redundant length of the sequence in order to avoid a huge accumulation of the sequence overlaps in the synthesis result.

[47] audio/SynthesisResult/Polyphonic/test2/DrumLoopVector1.wav
[48] audio/SynthesisResult/Polyphonic/test2/DrumLoopVector3.wav
[49] audio/SynthesisResult/Polyphonic/test2/DrumLoopVector4.wav
[50] audio/SynthesisResult/Polyphonic/test2/Mix_2048_512.wav
[51] audio/SynthesisResult/Polyphonic/test3/Violin_Fragment.wav
[52] audio/SynthesisResult/Polyphonic/test3/Mixed_2048_512.wav

We have observed polyphonic constructions in both the advanced frame-based and the sequence-based CSS techniques, but they have not been produced as a result of polyphonic representations but of the simultaneous sequences supplementary constructed in order to represent the omitted characteristics of the target.

We have also presented different polyphonic synthesis techniques. The NMF based CSS technique has a great potential to reconstruct the polyphonic structure of the target, it is yet off from practical use. The current algorithms are not sufficient to realize the versatile and effective polyphonic representations, and the further studies are awaited. There are indeed some issues and their potential solutions for the lack of the practicality which are worth examining, which will be the subject of our future work.

Part VII
Graphic User Interface

28 Introduction

A Graphic User Interface(GUI) is provided in order to enhance the user-friendliness and to obtain both flexible and rigid synthesis results. The GUI is designed for the following purposes: displaying the waveform of the target file and the synthesis result, automation of functions for controlling the balance (weighting) of the feature vectors, and automation functions for controlling the threefold parameters for the BMU retrieval and sequence concatenation.

Our GUIs have been developed on three different platforms, which are, Max in C, Web application written in HTML5, Javascript, and PHP, and the Cocoa application written in C and Swift. The Cocoa application version of the synthesis program is still under the development and therefore, we will just introduce it shortly.

29 Max

The first version of our synthesis system has been implemented on Max6 which has been developed by Cycling74[86]. The advantage of Max is that it provides a lot of efficient and robust UIs and bundles of programs such s waveform viewer, automation, and DSP functions which are applicable to our synthesis system.

As shown in figure 102, the GUI for max consists of a waveform of target data, a waveform of synthesis result, the first automation parameters for Magnitude, Centroid, Spread, Flatness, Pitch, and the second automation parameter for the threshold constraining the BMU retrieval, the second Threshold for constraining the sequence concatenation, and the third threshold is a trigger of the new sequence. The forth threshold defines the minimum length of the obtained sequence to be employed for the synthesis result. The Density sets the interval of a BMU, which prevents to retrieve a BMU in every frame. The Similarity changes the priority of the synthesis which decides to employ the BMU of a higher similarity or longer sequence.

29.1 Discussion

Besides its significant advantages, Max is not suitable to develop a heavy computing program which could analyze a tremendous number of audio data and implement a MIR system organizing a large size of the database. Therefore, we anyway had to develop another software which could operate the audio analysis and database constructions by means of multi-core processing which maximizes the performance of the computer. Concerning the compatibility between different environment, the Max is an adequate platform because it works on both Mac and Windows OS as long as the Max is installed. One obstacle will be that both developers and users need to purchase a full license of Max to modify the programs and save changes.

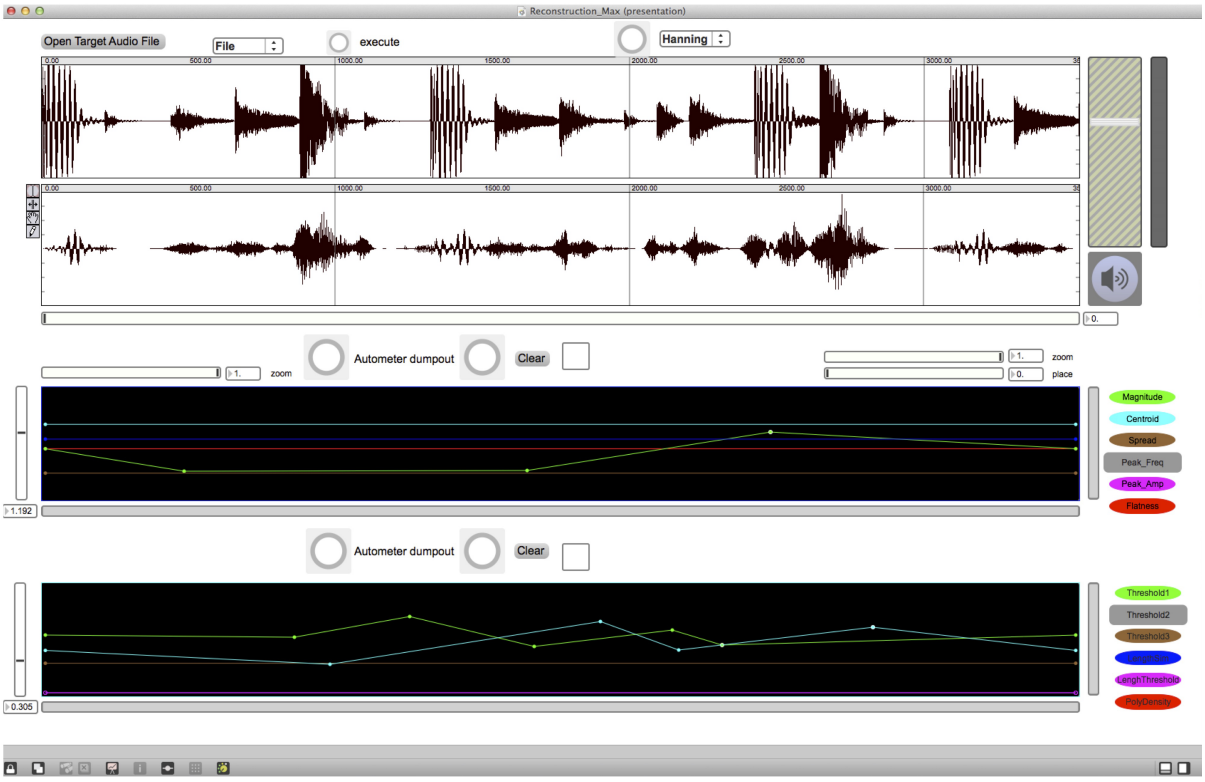


Figure 102: The GUI for Max6

29.2 Ad-hoc Score

In order to create the conditions that would allow the program to be used for chamber music composition, we designed this program with the following specifications: a

flexibility-modified database, the use of polyphonic synthesis, the use of multiple algorithms that produce the longest possible sequences in order to increase the chances of the sequences being used for the final product, the use of multiple parameters that allow the user to precisely control the synthesis results, a Graphic User Interface to improve the user-friendliness, and an ad-hoc score which enables composers to analyze the synthesis result and to translate it to a musical score.

It is particularly important to provide an ad-hoc score which represents the complicated structure synthesis result. The score is similar to a piano roll representation common in sequencers shown in figure 103. The horizontal line represents the time axis divided into cells, each reprinting a unit which is equal to the smallest segment of audio data. The vertical line represents the number of voices that indicate how each sequence is polyphonically generated and employed in the synthesis result. Each rectangle colored with alternative colors of dark and light greens which indicate a sequence constructed with successive units from the same source unit. When the rectangle objects are selected with the mouse cursor, a pop-up window displays information about the sequence shown in figure 104. This information includes the number of frames which represent the position of the source data, the beginning time and length which represent the position of the synthesis result, and the address of the source file. When a rectangle object is clicked, the corresponding audio data is played. It is also possible to select multiple objects and to play their audio data simultaneously.

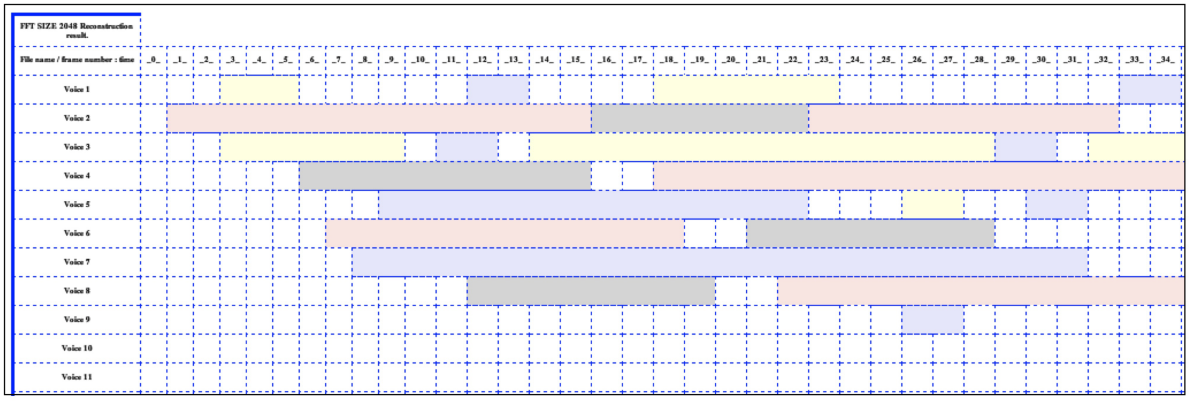


Figure 103: The ad-hoc score

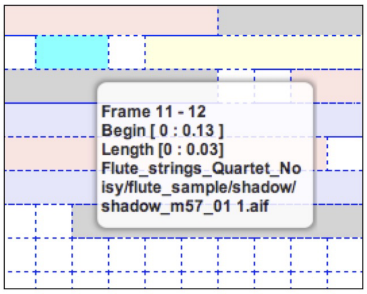


Figure 104: The ad-hoc score

30 Web Application

30.1 Introduction

Due to recent progress in building mass storage devices for computers, there has been an increase in the development of application software that uses large quantities of data. In particular, as large size databases are becoming more common, and some software requires such large quantities of data that the average personal computer lacks sufficient storage space for such programs. Our synthesis system also uses a large size database (minimum of several gigabytes), in order to yield a higher quality of synthesis result. In this section, we will discuss the design of this program to be used on various devices including smartphones or tablet computers, as we would like to make the program as user-friendly and practical as possible for computers. However, we must be aware of the risks of using such large quantities of data, as this could become an obstacle for distribution of the application software and restrict the audience.

Today, many services that employ large quantities of data are provided mainly through internet technology. Their databases are placed on a server, and the users can simply access and retrieve data they need. Here, we will discuss the possibilities presented by these kinds of software focusing in particular on Web applications for music. In consequence, we will report on the pros and cons of applying these applications to our synthesis system.

Our synthesis system can be primarily divided into two parts; a synthesis engine, and a database of source sound data and their analysis information. In the synthesis program, it has been found that the greater the size of a database, then the greater the versatility and the higher the quality of synthesis result. This is the most impor-

tant criteria because the synthesis program is designed to target Computer-Assisted Composition for instrumental music, and this requires natural and physically playable synthesis result. Furthermore, higher user-friendliness and easier usability are required (in contrast to purely electronic music) because the synthesis results are often translated to a traditional musical score. However, a large volume of data could require a high-capacity storage of the device, thus making installation much more complex. This factor makes the database more difficult to organize. The database is huge (several gigabytes) and is distributed through the internet. The size of this database limits the types of devices that can access it, as it is then limited to being downloaded on desktop or laptop computers, thus excluding mobile devices such as smartphones and tablet computers.

Our synthesis program has additional complicating factors, as its database is designed to be flexibly modified by the user. This is one of the most unique aspects of our synthesizer. Thereby it is assumed that the database is constructed not only using individual instrument samples., but also with short musical fragments such as extended techniques or musical gestures. While the fragment data could generate more natural synthesis results, this data can also easily require much greater storage space. In addition, as we intend to design the software to be used on mobile computers in order to enhance the user-friendliness for actual composition work, it is therefore our top priority to address this storage problem.

A Web application is an application software which runs on the web. It consists of two main parts; the Client-side, which indicates the device or software that receives service, and the server or Host-side, which indicates the device or software which offer service. A Web application uses a web browser as a client to execute any programs provided by the server. A Web application can be used as a mere extension of a web page, or it can be used as an advanced software.

The Client-side is developed by both markup and script languages such as HTML, Javascript, flash, etc. that help to create dynamic and animated web pages. Server-side programs are developed by using programming languages such as PHP, CGI, Ruby, etc. and database constructions by MySQL among others. In short, a Web application runs through the interactions between both sides. However, as a Web application runs on a browser, the programs are not entirely dependent on the Client-side devices or operation systems. This is one of the most unique qualities of a Web application. In

addition, any user with a browser and internet access can use the application without having to download a complicated installation on their devices.

The most familiar examples of the Web application that employ large databases are Google search, Facebook social network service, Youtube video sharing service and so on. All of these have large databases on their servers, and clients can receive services by retrieving only the necessary contents.

Here are the possible benefits and issues associated with implementing our synthesis program on a Web application.

Benefits :

- The Client does not need to install any program and only requires a browser to access the program/application. This solves the potential compatibility issues with the clients computer.
- Web applications require minimum disc space on the Client-side, thus enabling the use of mobile computers.
- Web applications can divide the computing process into Host and Client-sides, thus the server with its high spec computer and a higher efficiency can carry out high-performance computing.
- Web applications do not require the client to perform any special software upgrade procedure.

Issues :

- Web applications require an internet connection to provide the services, and the speed of internet access can significantly influence the performance of Web applications.
- Different browsers support APIs in slightly different ways, and thus it is important to develop programs that are sufficiently general and that are constantly updated.
- The copyright issues involved in uploading and distributing audio contents must be resolved.
- Servers are at risk of being hacked. It is therefore very important to maintain high security for the programs.

Most of these technical issues can be solved by making robust programs. It is also important to adequately consider all copyrights issues such as Digital Rights Management[73][53].

30.2 Development Environment

There are several programming languages and environments used in developing Web applications. Our synthesis program requires a comfortable GUI, audio processing functions which work on a browser, and proper database management.

Flash Player Adobe Flash player is one of the most popular platforms for Web applications. There are various web pages and Web applications developed by using Flash player on the internet. Flash can create dynamic and interactive animations and multimedia on a browser. However, due to the controversy between Apple and Adobe[95], Apple has stopped supporting the use of the Adobe Flash player on Apple mobile computers due to Adobes closed system, security issues, and high demand for device performance. Since Flash cannot be properly installed on Apple mobile computers, Adobe has also stopped developing it for later models scubas Android 4.1 and has been promoting its new programs which translate Flash web sites to HTML5. These developments have resulted in HTML5 becoming one of the most promising new platforms for developing Web applications for different devices.

HTML5/canvas HTML5 is the fifth major revision of the core language, and it is developed by the World Wide Web Consortium(W3C) and WHATWG. HTML5 subsumes HTML4 and provides enhanced functionality in new elements, attributes and APIs[91]. The language is able to describe clearer sentence constructions, simpler movie/sound implementations, and many other functions which were previously engaged by Adobe Flash player. The first plot type was distributed in 2004, and the full specifications were completed in 2014.

The canvas is one of the most important elements in HTML5, and this allows it to render 2D graphics and bitmap images on web pages. This canvas originated in an Apple Webkit component in 2004. This technology has been employed by Dashboard

[53] According to the [73], Digital Rights Management is an access-control technology used by manufacturers, publishers, and copy-right holders to limit the usage of digital devices or information. It describes the technology that prevents unauthorized distribution and usage of content, media, or devices.

applications and Safari browsers in Mac OSX. The canvas provides many opportunities to create GUI and animations on web browsers without using Flash player. Although HTML5 has rich functions to develop advanced programs, it does not handle APIs which would enable it to use complicated audio processing. This issue can be solved by using Web Audio API, which is described in the next section.

Web Audio API Web Audio API is high-level Javascript API developed by W3C for the purpose of processing and synthesizing audio data in Web applications. The first working draft was distributed on December 15, 2011, and the latest version was published on October 10, 2013, on the W3C website[96]. The API has two main goals: the first is to fully utilize the capabilities provided by modern game audio engines. The second is to manipulate and play sample data using some mixing, processing, and filtering tasks that are found in modern audio production/editing applications such as sequence software, Audacity, Logic Pro, ProTools, etc. Web Audio API has an advanced audio processing library which generates, manipulates, and plays diverse signals, obtained from oscillators or various kinds of audio files. The library includes multiple functions such as filters, compressors, delay, FFT, oscillators, etc.

30.3 Related Web Applications

Web Audio Editor The HTML5 Audio Editor[90], or audio tool web app, was developed with HTML5 and Web Audio API by Rainer Heinke. Parts of the audio editing functions were taken from the Audacity project[89]. With this audio editor, the user can simply drag-and-drop an audio file(WAVE, OGG, and MP3) into its interface, and the program will plot the waveform and spectrogram obtained by Web Audio API functions. The GUI is made with the HTML5 canvas element. A particular audio region can be selected by mouse click and manipulated in diverse ways such as changing volume, normalizing, silencing, and fading in/out. Following these manipulations, the audio data is rendered and exported as WAVE format data. This Web application demonstrates how Web Audio API can manipulate and play audio data in a flexible way.

Recorderology The Recorderology is a research project using recorder instruments which has been carried out by Mayer-Spohn and Author (Takahashi) since 2013. This

project aims to analyze and provide various sound samples and extended techniques of recorder instruments primarily for contemporary music. The research results are published on the projects Recorder Map web site[93] and in a paper[94]. One of the unique aspects of the project is that some of the contents are provided by using a Web application.

For example, figure 105 shows a type of sampler which presents various kinds of playing techniques played by eight different size recorders. This sample can play any possible combinations of the following sound characteristics for any size instruments, articulations, dynamics, air pressures for blowing, registers, bisbigliando, and extended techniques.

The user can select a particular instrument and playing technique in the menu above the sampler, and he or she can then play the sound result by clicking on a corresponding note and a dynamic.

This example shows how the Web application can vastly increase the scope of instrumentation research. This software has been designed so that users can listen to the different instruments, notes, and articulations very easily and so that they can intuitively compare any possible combinations of sound. This is particularly impossible using a normal web site or a paper-based media. As demonstrated in this example, maximizing user-friendliness is one of our top priorities for our synthesis program.

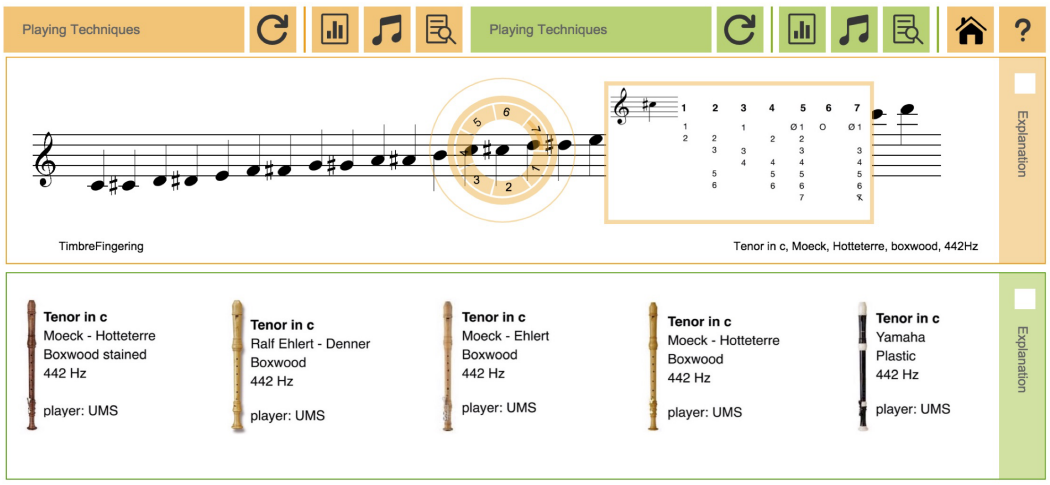


Figure 105: The GUI of the Recorderology

This figure illustrates the layout of the web-application recorderology on Google Chrome browser. In the main page, two players are installed in parallel on the same window, and the user is able to assign a set of samples of different playing techniques or instruments to each player in order to investigate the sensitive differences between them. The playing technique menus appear on the top of the player colored by orange and green. When the user selects one specific note, the program shows its possible variations of the playing technique which appears in a circle around the note, which increases the visual focus to the standard notation. Simultaneously, the user can navigate around it to access different options of its timbre. Figure and the description are quoted from the paper[94].

30.4 Implementing our synthesis program on a Web application

Figure 107 shows the structure of our synthesis program on the Web application. It has two sides; 1. the Host-side, which is the server side and consists of audio analysis engine, synthesis engine, and database, and 2. the Client-side, which consists of the users device and browser, in which GUIs such as waveform, automation functions, menus, and sequence for the synthesis are provided. The process of synthesis is divided into three steps. In the first step, a target audio file is uploaded from the Client to the Host-side, and the audio data is loaded into the AudioAnalyzer software which analyzes its characteristics. The analysis result is outputted as a CSV(Comma-Separated Values) file and stored in the Host disc storage. In the second step, the Host-side provides audio information(audio length, data size, etc.) to the Client. In turn, the Client generates the waveform of the audio data, the automation functions which are adopted to the obtained audio length, and multiple menus. The user can also modify parameters by using the automation function and multiple menus at this step. Three parameters are then sent to the Synthesis Engine program. In the third and final step, the Host-side

synthesis executes a synthesis process. The synthesis result is generated with multiple source audio data retrieved from the database located in the Host storage, and this result is then provided to Client. Following the completion of the synthesis process, the waveform of the synthesis result is plotted on the waveform display. The second and third steps can be repeated as necessary for the re-synthesis using different parameters.

In this system, each task is assigned to the most appropriate side. In particular, the Synthesis Engine, which is the main program and the most expensive process, is assigned to the Host-side. Therefore, low spec computers such as mobile computers can use this software just as effectively as desktop or laptop computers.

Figure 106 represent the GUI launched on a Google Chrome browser. The waveform display plots a target audio data in violet and the synthesis result in black. This representation of the waveforms enables the user to compare the target and the result visually. The four automation functions that control the synthesis result are placed directly on top of each other. Each function can be switched with another by selecting particular parameters using the HTML5 selection element which is opened in figure 106. The user can download the synthesis result as a WAV data and the analysis data as a CSV file.

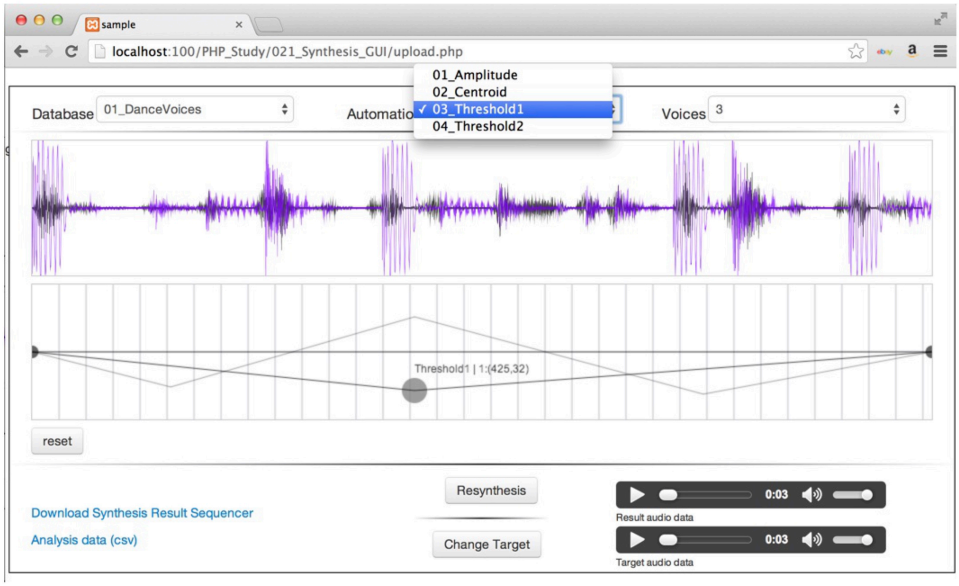


Figure 106: The GUI fo the Web application

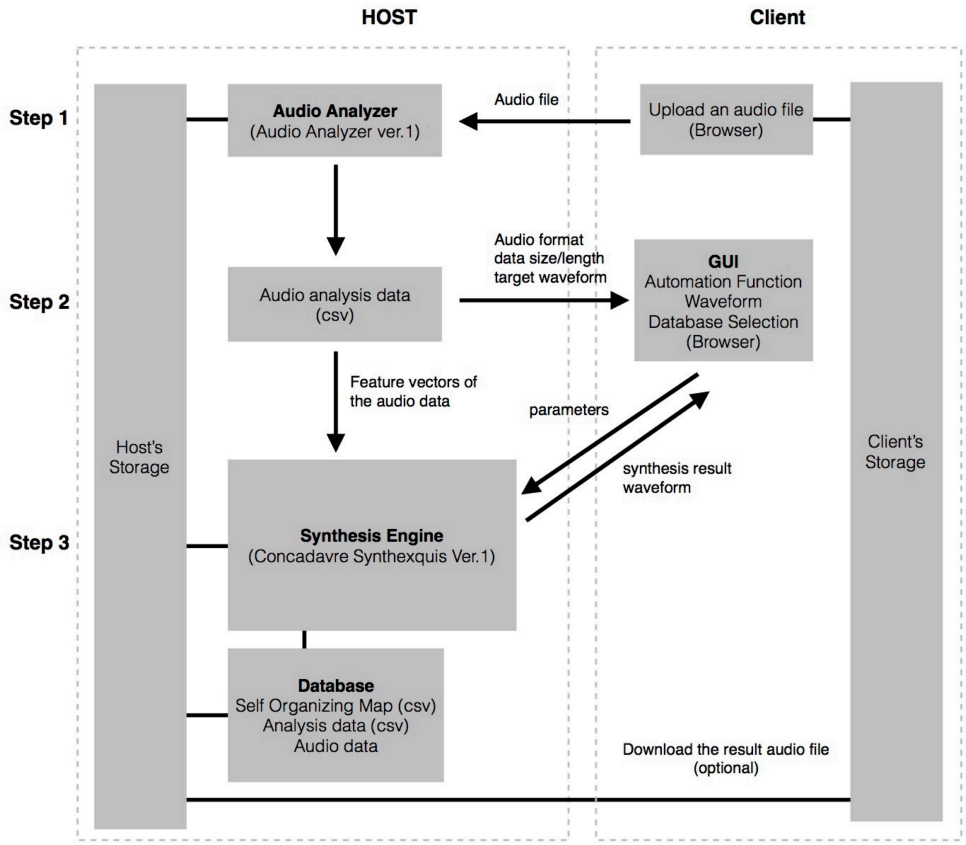


Figure 107: Synthesis process

The step 3 can be repeated multiple times until the satisfied synthesis result is obtained.

30.5 Sequencer

The Sequencer is the new version of an ad-hoc score, and its appearance is similar to a piano-roll. The previous version of the ad-hoc score was developed using HTML tags and Javascript exclusively, which did not yield good usability. This new version was developed using an HTML5 canvas element and Web Audio API that could create much better usability, better synchronization of playing multiple audio data and modifications of the synthesis result. Figure 108 presents the Sequencer, in which the vertical line represents the number of voices, and the horizontal line represents the time axis, and each black square represents a sequence which constitutes a synthesis result. The user can now listen to the individual or multiple sequences and obtain information such as source data name, sequence positions, length, etc. The synthesis result is also modified by dragging each sequence square in the time axis, and extending (should the sequence

audio data have more length) or shortening the sequence by dragging the edge or the sequence.

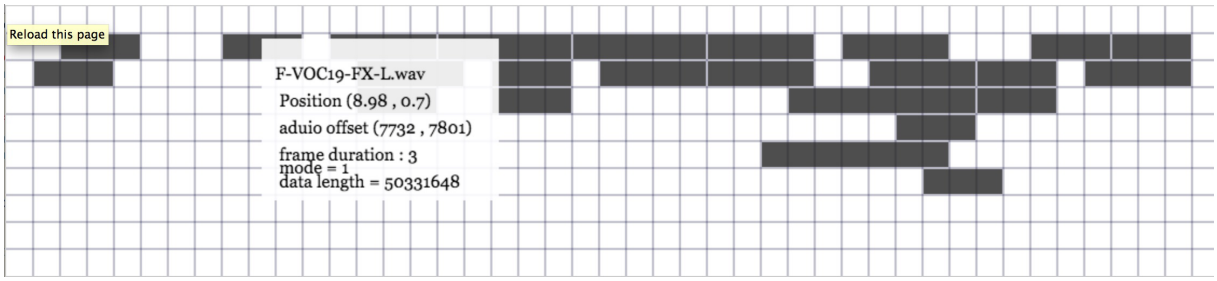


Figure 108: The sequencer on the Web Application

30.6 Discussion

In our research, we are still addressing some issues in the Web applications and in the software development. At the current stage, the software is not yet ready to be distributed or thoroughly tested. Some of the issues we are facing include the stability and security of the program, the volume of data communication between Client and Host, and database organization. For the stability and security issues, the most important factor is the Host-side security in order to prevent any fatal crashes.

The data communication issue would be partially addressed by relying on lossy compressed audio data such as MP3. It is also important to limit the maximum amount of data communication for one execution which can be accomplished by restricting the length of a target sound file. At present, our database does not use a public database management system, and this is the biggest obstacle to developing the efficient and robust database system that we are striving to build. Further research will be required in order to overcome this obstacle.

31 Cocoa Application

The development of the Cocoa application version of the synthesis program is one of our future works. The software will cover all UI functions implemented on the Max and Web applications and will improve the issues we have previously faced on in other platforms. Cocoa is a framework for developing software for MacOS which allows us to utilize various UIs and APIs called NSObjects provided by Apple. In

this program, we have employed two programming languages, Swift and C, depending on the tasks. For instance, expensive and time-consuming processes are written in C in order to optimize the processing, and the UI part is written by Swift which can communicate with NSObjects. Some of the GPU based APIs such as SceneKit and OpenGL, are utilized to accelerate the drawing of the waveforms, spectrograms, and graphs showing the analysis results. The parallel processing is applied to the tasks requiring the expensive computations like constructing a database and thus, this version of software requires the full performance of the computer.

Figure 109 illustrates the GUIs of the software currently under development. We still need to work on designing an efficient and user-friendly GUI which allows us to intuitively control the synthesis engine and have access to the database.

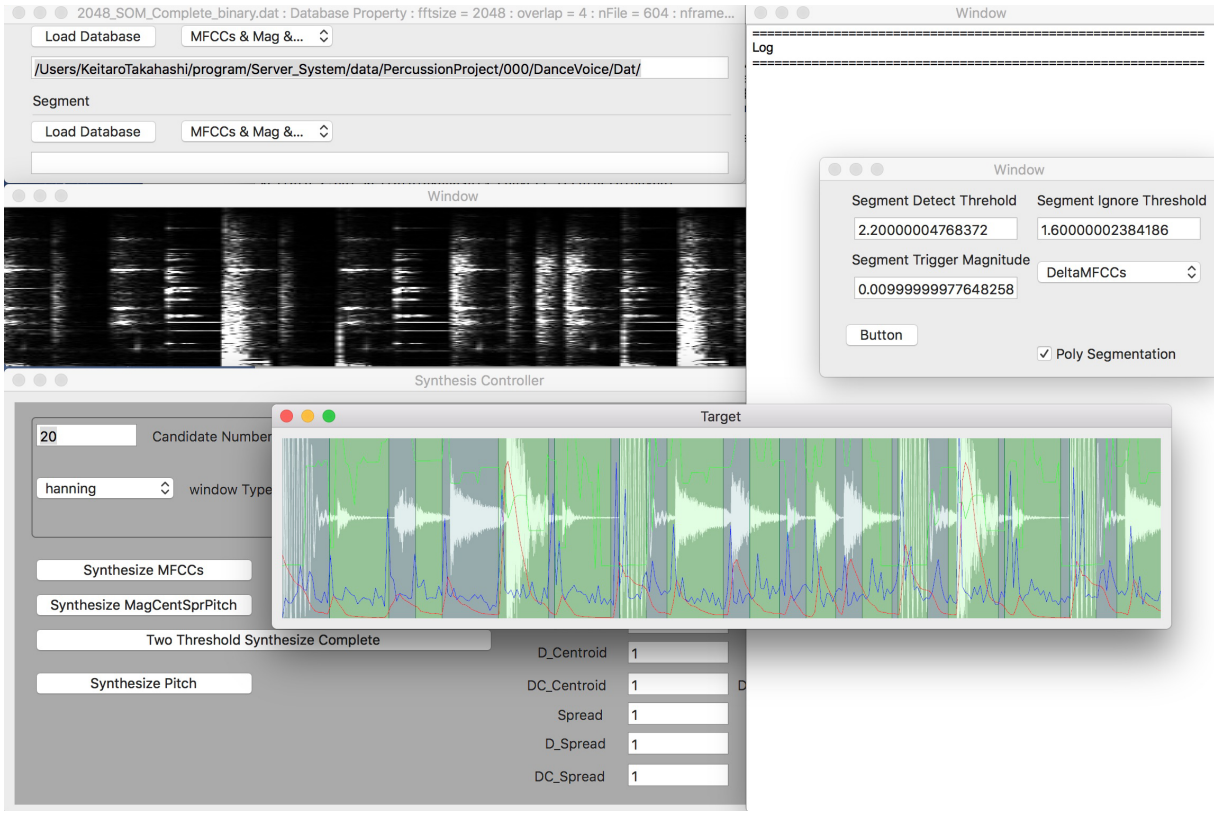


Figure 109: The GUI of the Cocoa Application

The UIs are merely designed for the test of synthesis, and thus, it is not yet sophisticated for the use in the creation.

Part VIII
Application to Composition

32 Introduction

In this section, we will describe the actual applications of our Computer Assisted Composition program on two different types of compositions: Bricolage for a percussion ensemble and Pentimento for a small ensemble consisting of Flute, Violin, Viola, and Violoncello. Although the synthesis program is designed for both electronic music and instrumental music composition, we focus on the ensemble piece which demands a lot of restrictions because we need to transcribe the synthesis result to the actual music played by musicians. We will suggest some applications for this program, including constructing a database for particular compositions, controlling the synthesis program, and compositional and instrumentation methods.

32.1 Short descriptions of the compositions

Bricolage The piece Bricolage was written for a percussion ensemble, consisting of six players, and live-electronics. This was the first piece I composed with the assistance of the program. Bricolage is a term used to describe an artistic genre, especially for architecture or craft works, in which a diverse range of materials whatever available, including various kinds of garbage or scrap, are employed. It is sometimes also denominated junk art. Swiss painter and sculptor, Jean Tinguely, is one of the most famous artists making different uses of Bricolage in his art. Tinguely assembled scrapped industrial materials such as gears, wheels, broken machines, and furniture, and used them to create sculptures or installation art. In the realm of music, there are some musics that employ the idea of Bricolage such as generating sounds by playing housewares or tools, sampling various kinds of sounds existing in the environment and creating with them a kind of electronic music. While composing this piece, I payed great attention and make a special effort to listen to environmental sounds existing around me, carefully observing them from a musical point of view, and investigating how these musical materials were possibly combined or concatenated in fragments or passages in the piece. The instrumentation consists of only percussion instruments, this was in order to apply dissonant and inharmonic sound qualities for the creation of complex timbre associations. The

employment of the program generated many unexpected musical ideas and highlighted new aspects of the sound by generating various combinations of instruments, gestures, and rhythms from banal sound materials. In their turn, they became triggers to find new musical ideas of my own. I used the program in order to create not only timbre associations, sound textures, and gestures, but also to create a compositional system, such as musical scales, with which I constructed motifs in the piece.

Pentimento Pentimento was composed for Quartet Flute, Violin, Viola, and Violoncello after Bricolage for percussion ensemble, I focused on writing a piece for instruments that have harmonic spectra as the next step. I selected a traditional ensemble formation in order to investigate new possibilities of this program thereby defining totally different conditions as I had for Bricolage. Pentimento is a term used in traditional painting. It defines the painting of a new picture on top of a different pre-existent one, or the tracing of a previous work. The technique is used as a compositional technique or for a special painting effect. It was not unusual in earlier times to re-use the canvas of a pre-existing picture. The previous picture, in whole or in part, was then sometimes incorporated into the new one. Also a pre- existing painting would be covered with base-color and, later, this base would be erased in some areas in order to let details of the previous painting raise to the surface of the new picture. The technique was employed both intentionally, to create new expressions, and sometimes unintentionally when a new picture was painted on an old canvas due to financial reasons, or just for study, etc. For my piece Pentimento, I constituted a database with fragments of a previous work of mine written for solo flute. I did this in order to generate new musical ideas based on the old work. This was my way of incorporating the concept of Pentimento into my own compositional process. For example, some re-synthesis results were decomposed into small musical materials. With those small elements, new musical events such as motifs, fragments, rhythms, etc. were generated by recombination. The re-synthesis potentially includes many kinds of musical elements and some of them are represented neither completely nor clearly, they are often hidden behind other musical aspects. The process of seeking those hidden musical elements and bringing them out to generate new musical elements made me conjure up the idea of pentimento.

33 Demonstrations

33.1 Database

The databases employed throughout the compositions of these pieces consist of two types of sound data: samples of individual instrumental sounds and recordings of short phrases or rhythmic patterns. Samples of the first type were created by recording the sound of an individual tone played on each instrument using different techniques, mallets, and so on. Samples of the second type were created by recording the performance of short musical gestures and extended techniques played by either individual instruments or sometimes combinations of different instruments. Additional samples were obtained from sound materials provided on the website of the University of Iowa Electronic Music Studio and from Prosamples PS41 Solo Strings sampling library produced and published by Crypton Co. These libraries provided a sufficient variety of sounds, techniques, and transpositions for our purposes; it is possible that this richness of sound material allowed for the high-quality reconstruction of target files.

33.2 Example 1: the extraction of musical motives and elements from the re-synthesis (Bricolage)

The first section of Bricolage was composed by using a type of scale system derived from a portion of the synthesis results, and a motif was then made using this scale. The re-synthesis was created with the target file shown in Bricolage_A.aif, and the sound was generated by rubbing a wooden snare stick on the surface of a wooden table. The synthesis result can be heard inBricolage_A_Result.aif.

There are three main steps in the synthesis process of Bricolage_A. The first step is to organize the parameters and the database by focusing on generating the specific sound texture constructed by the ascending and descending gestures produced by the gongs; this texture is then alternated with low sounds played by the timpani and the bass drums. In the second step, the synthesis result and its ad hoc score are translated into the rough sketch of the piece using the listed instruments. In this process, only some specific timbre associations and musical gestures are extracted (see figure 110 and figure 111). This step depends entirely on the composer's intentions. In the third step, the rough score is analyzed in order to reveal more concrete musical elements. This process requires a more traditional compositional technique. Figure 110 shows five

adjunct note pairs extracted from the rough score. They are defined as the minimum-unit musical element in this re-synthesis, and these are some of the most significant elements to construct further timbre associations and musical gestures.

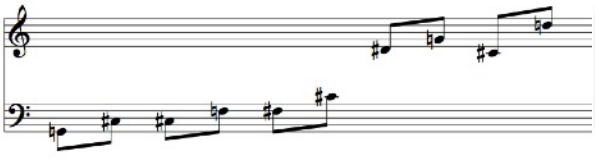


Figure 110: The minimum-unit musical element

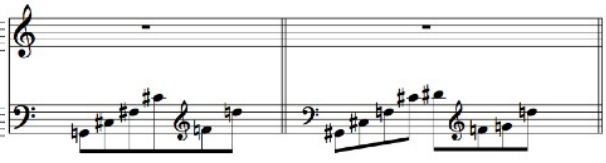


Figure 111: The minimum-unit musical element

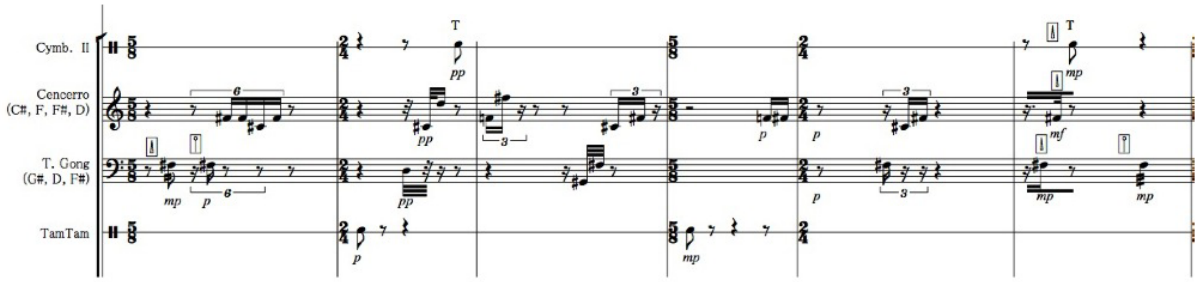


Figure 112: Bricolage, bar 38 to 43

These note pairs are played by different instruments and form arpeggios, as shown in figure 111. The first bar of figure 111 shows an arpeggio found in the first seven bars in the rough score, and the second bar shows an arpeggio found in the last six bars of the same score. Figure 112 shows a part of Bricolage in which the ascending motives are originally created by the synthesis results described above. In this short fragment, different note pairs and arpeggios are partially combined to develop the musical sequences.

Another important musical structure, which is heard as an alternation between low sounds and the ascending motives, is represented by the combinations of the bass drum the timpani, and the low register gongs. Figure 114 represents the alternating structure, in which the low register sounds are marked with circled black lines. There are arpeggio motives formed by the different low sounds, and these motives consist of note pair played by different instruments. The marimba plays a complete arpeggio (clearly marked with

square black line) throughout the fragments.

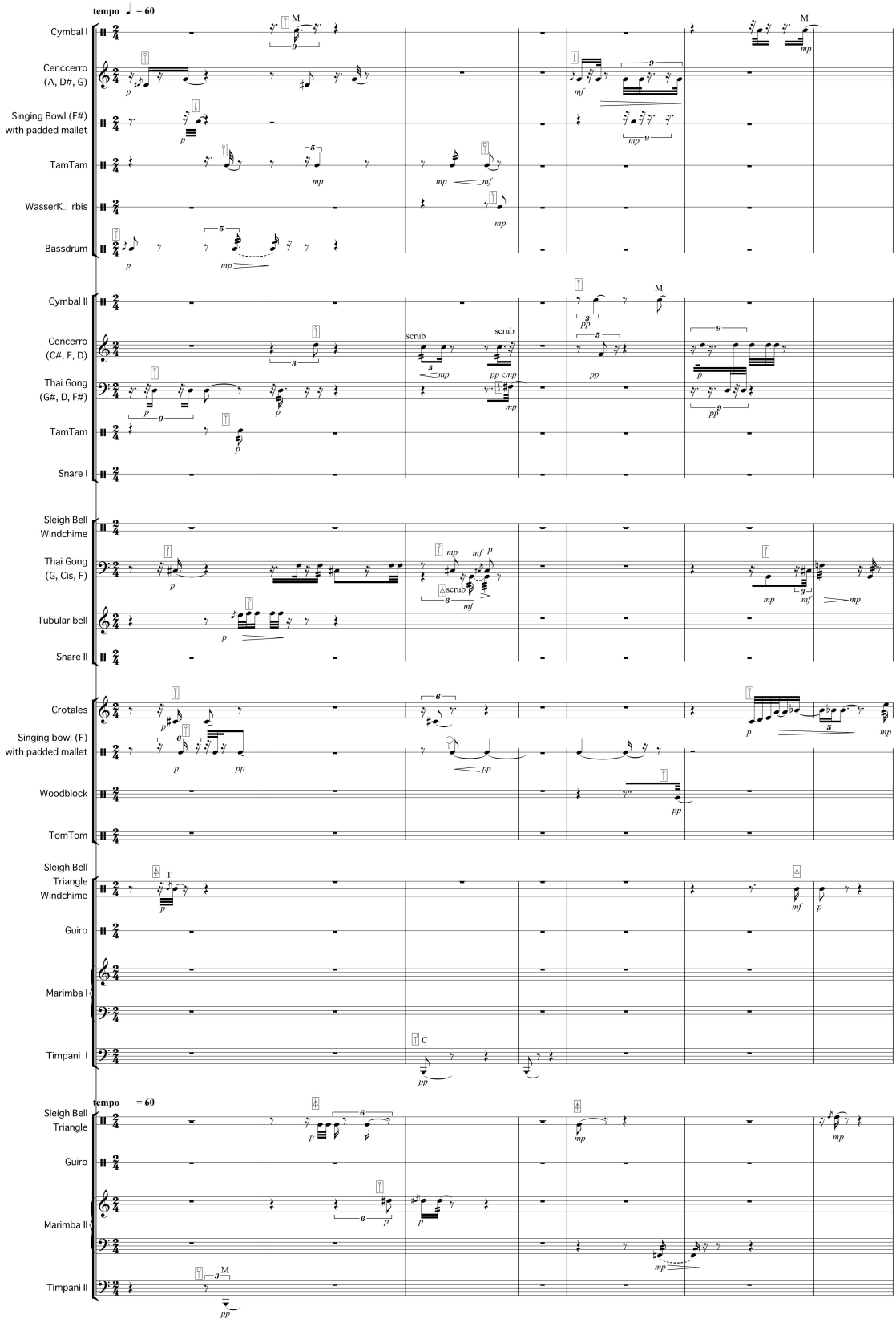


Figure 113: sketch of Bricolage

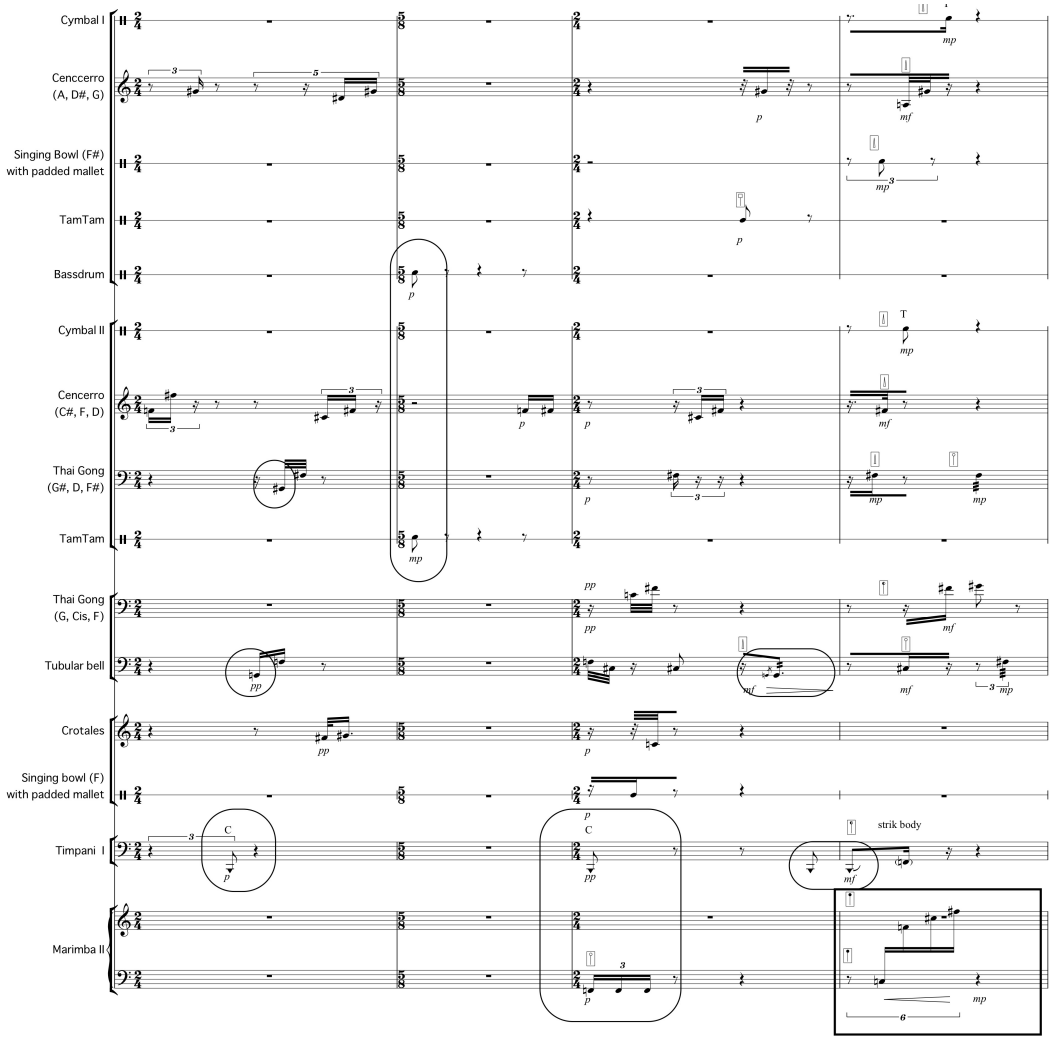


Figure 114: Alternating structure in Bricolage

33.3 Example 2: subtraction method (Bricolage)

Another way to apply the synthesis to this piece is by using the subtraction method. In figure 115 (Bricolage from bar 63), the musical gesture was produced based on a synthesis whose target file was created by tearing up a piece of paper. The target file is Bricolage_B.aif, the synthesis is Bricolage_B_Result.aif. In the final score, the granulated sounds are represented by tremolos in the cymbals and the snare instruments. This is done in order to reduce chaotic noises and to produce a more coherent timbre association. The density of sounds in the re-synthesis is reduced and the sound color is made more coherent by using less instruments and by employing the tremolo tech-

nique. This method can be said to be a subtraction method : this is a practical way to re-interpret the synthesis by filtering out its result in order to enhance the possibility of the true performance and thereby filtering out more significant musical gestures from the piece. This subtraction method is also used in Pentimento as described further examples.

33.4 Example 3: employment of musical fragments (Pentimento)

The following example shows one of the strongest findings of this research project. The database of Pentimento includes many fragments of flute sounds and consists in the combination of whistle tones, keyclicks with voice (figure 116), pitched sounds with sweeping breath gestures (117), a fast passage played with breath noise and tongue ram (figure 118) and fragments of cello harmonics (figure 119), among other sounds.

These fragments are mainly used in the re-synthesis of fluorescent and bulb lamp sounds, and these are the most significant fragments used to generate new musical ideas. In the re-synthesis of Pentimento_A (Pentimento_A.aif and Pentimento_A_Result.aif), the target file begins with the flickering noise of a fluorescent lamp followed by a buzzing sound. In the re-synthesis, the flickering noise is represented by a high register staccato in the violin, and the buzzing sound is represented by a sweeping breath noise in the flute in addition to a tremolo harmonic in the violoncello. Figure 119 shows the beginning of the piece which features a musical motif derived from the re-synthesis. In the next subsection (Example 4), other fragments are used in re-synthesis, thus creating variations of the synthesis result. Example 3 and example 4 show one of the most unique approaches used in this research project. This program not only makes it possible to find the best solution in relation to the full orchestra and to all instrumental techniques, but it also allows the different source and target materials to be re-combined in an ad hoc search for the most suitable solutions within the constraints of the material. This process is entirely dependent on the creative intentions of the composer.

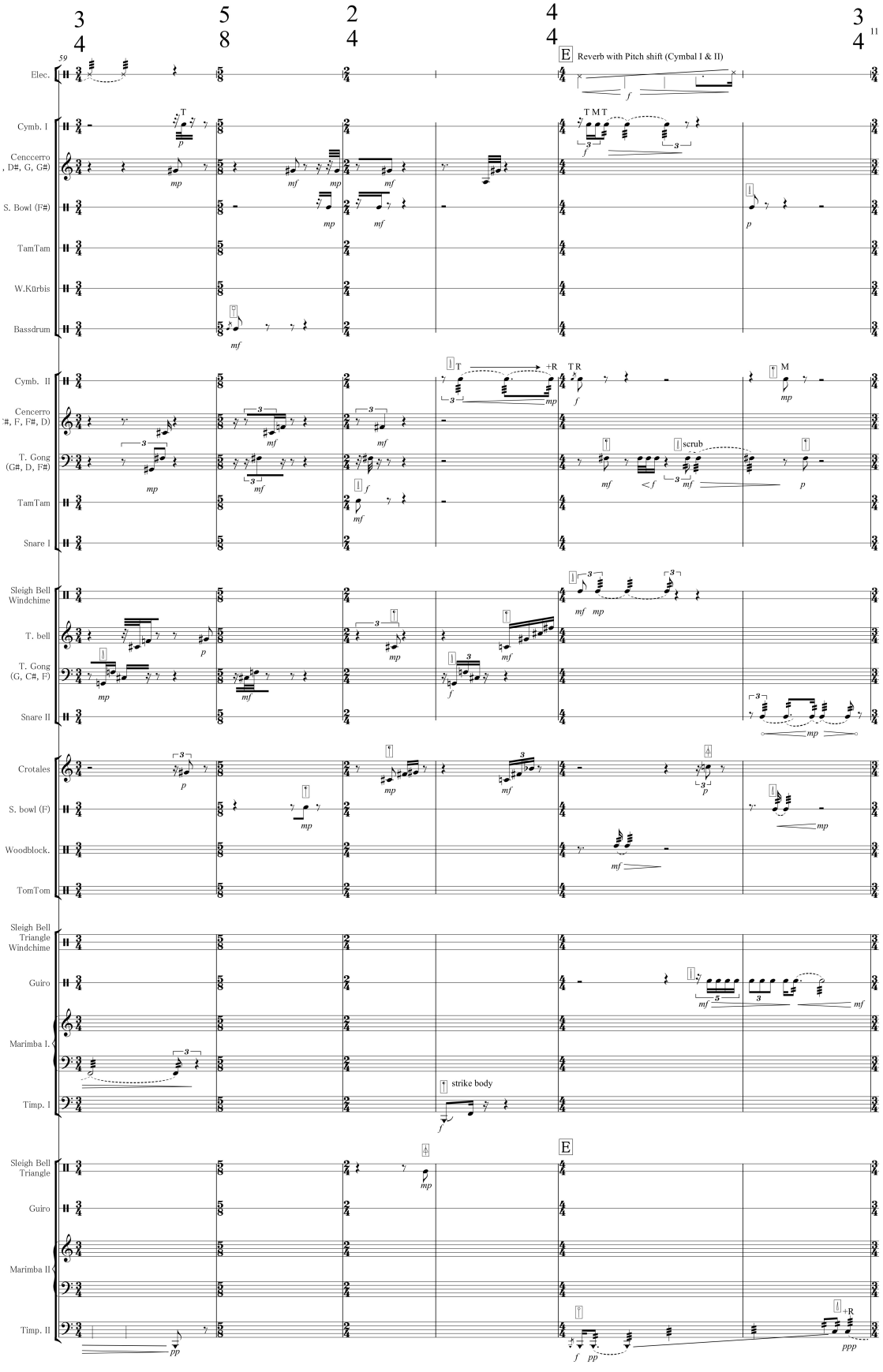


Figure 115: Bricolage from bar 63

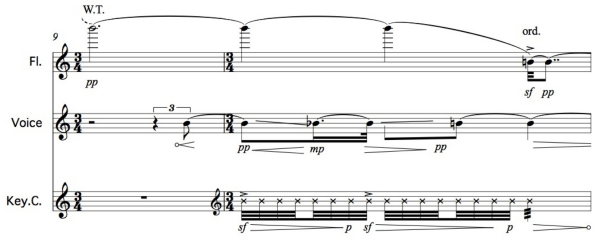


Figure 116: whistle tone and keyclicks with voice

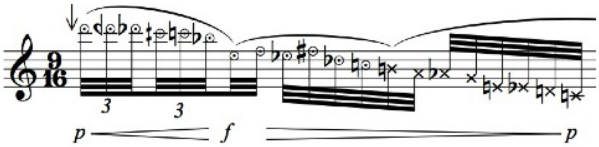


Figure 117: sweeping breath gestures

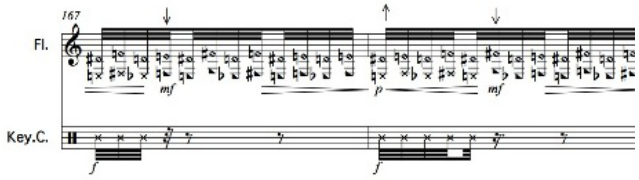


Figure 118: breath noise and tongue ram

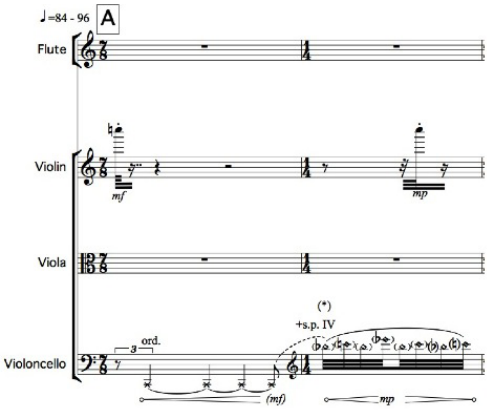


Figure 119: Pentimento, bar 1 and 2

33.5 Example 4: Variations of re-synthesis of the same target

The synthesis result of Pentimento_B can be perceived as an evolved version of Pentimento_A as it generates a more complex rhythm, sound texture, and also many granulated noises. It consists of additional diverse sound variations such as pizzicato, noisy string rubbing sounds generated by high pressure bowing, and whistle tones in the flute. Such a re-synthesis, which contains many varied sounds, is then modified by the subtraction method in order to extract specific musical materials and to employ them as fragments in the piece. In this piece, however, the re-synthesis is not simply adjusted to become a part of the piece. Instead, much smaller musical events or even individual notes are extracted from the result and are then utilized to build new musical events. For example, figures 10 and 11 show the score displaying the fragments in the piece derived from the combinations of the re-synthesis Pentimento_A and Pentimento_B.

While the musical fragments shown in figures 119 to 120 were all created based on the same kind of target sounds, the re-synthesis was produced with different parameters in order to develop the musical fragments.

Figure 122 shows the development process of the flickering noise motif originally derived from the re-synthesis Pentimento_A,Pentimento_B(Pentimento_B.aif : Pentimento_B_Result.aif),Pentimento_B2(Pentimento_B2.aif : Pentimento_B2_Result.aif), and-Pentimento_C (Pentimento_C.aif, Pentimento_C_Result.aif). The motif is gradually developed, thus creating more complex rhythms and timbres.

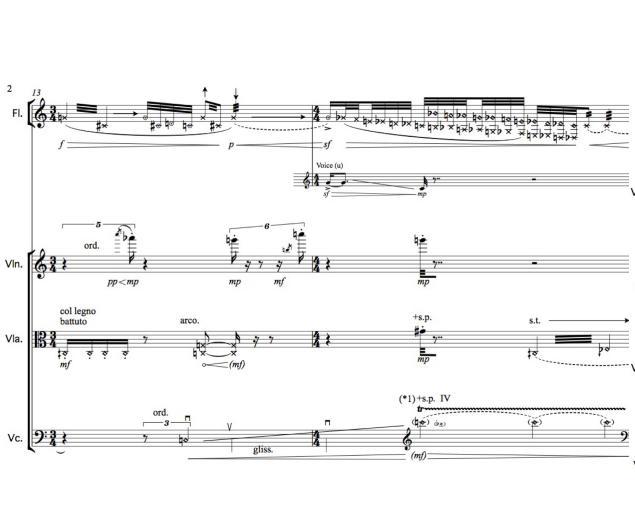


Figure 120: measure13 to measure15

Sweeping breath gestures by flute and harmonics by cello

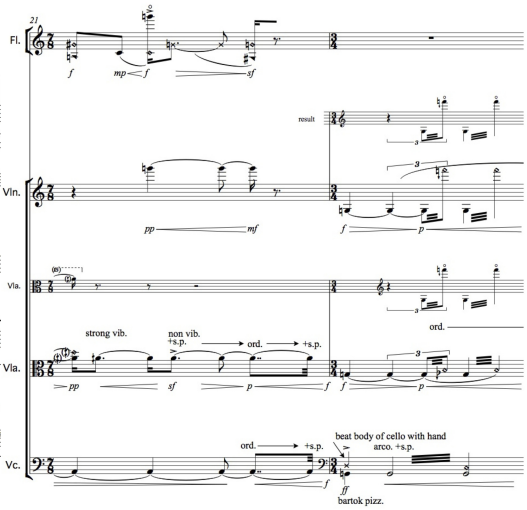


Figure 121: measures 21 and 22

The flickering noise is now also played by the flute and the harmonics are now played by the violin, viola, and viononcello

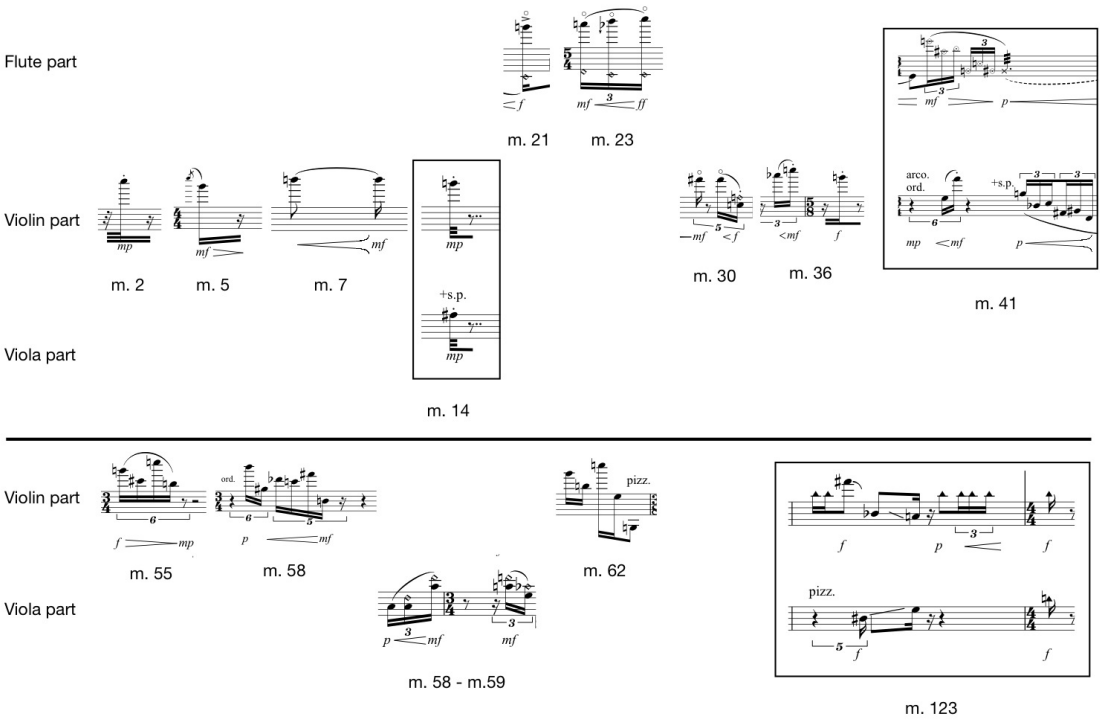


Figure 122: The development of the flickering noise motif

34 Discussion

Bricolage In Bricolage, specific rhythm, melody, and chords are not to be heard clearly, rather all musical events were created based on the idea of timbre associations and musical gestures were constructed with complex combinations of different instruments. These musical gestures generated by the program were often modified for a specific purpose and, at the same time, many unpredictable elements generated by the re-synthesis, such as granulated noises, sporadic sounds, et cetera. were also employed as accurately as possible. The character of these complex musical events was often abstract and heard as a sort of ornament found concrete elements, such as scales, motifs, et cetera. In Bricolage, all musical elements generated by the program enabled to produce a heterophonic result. The short fragments played by each instrument contain irregular rhythms, tremolo, and articulation effects, and it is the addition of those individual fragments which create a big modulation of the acoustic space, and the variations of the resulting musical gestures.

Pentimento The re-synthesis was decomposed into small musical events such as staccato, pizzicato, or a small gesture and reconstructed in slightly different ways in order to guide the musical progress. Music fragments created from the same sound idea, the same target file, or one or more re-synthesis, were often recapitulated with a development throughout the piece. Furthermore, each decomposed musical material was often independently developed and combined with further different materials in order to create a new fragment. The procedure could be so described: each re-synthesis becomes a sort of dot which represents a concrete musical idea created by specific combinations of musical materials. Multiple dots are then connected to form new fragments constructed by freely combining modified musical materials. Many concrete musical ideas were created by using the program, but the basic compositional idea was developed in a more traditional way. The composition examples were able to show that, due to the flexibility of the programs design, the program holds great potential for developing new musical materials.

Part IX
Conclusion

In this thesis, we have described different topics across the various research fields and eventually, we have integrated them to achieve our goal which is the development of a Computer Assisted Composition (CAC) program and its application to the instrumental music composition. The program is useful to create various musical ideas and the results have a great potential to provoke new musical ideas. We have only introduced the case of the instrumental music, but we see a lot of potential in this program of applying to other types of creation, such as electronic music, sound art, and installation. The advanced algorithms developed throughout the research can be also applicable to other research fields as the concatenative sound synthesis is employed in divers areas.

In the development phase, the potential of diverse algorithms was investigated and improved to be adapted to particular situations by combining with additional algorithms. In the Audio analysis, we investigated the various analysis methods, and attempted to define the most appropriate one to extract each descriptive features. These analysis methods are often advanced accompanied with novel algorithms we have developed, and when possible, their results are scaled to the perceptively adequate values. The employment of the high-level feature vectors is also an interesting factor to improve the re-synthesis especially because we gave a priority to produce a perceptively convincible result. The source separation algorithms, which are the Segmentation and Decomposition, were also discussed as a key of an advanced synthesis technique. Subsequently, we have looked closely the labeling and classification methods for an individual audio unit, and for a sequence or decomposed unit. In this process, a large number of data units are mapped into the lower-dimensional space called Self-Organizing Map which creates a straightforward and concise data retrieval system, and this system also contributes reducing the searching time significantly. Based on these fundamental research, we have developed a synthesis program. We presented various synthesis techniques and their synthesis results on different conditions, parameters, and creative purposes. The results were inspected by applying the spectral analysis and also the perceptional interpretation, which is one of a unique aspect of this research. We believe that it is significant to assess the quality of the result from both subjective and objective point of views because the design of the synthesis system is a core work to make our

CAC program efficient for the creative use. We found that there are still some issues to be solved to realize the versatile polyphonic representation particularly when using the pitched and harmonic target sound. All of those examinations were eventually tested in the actual instrumental music composition by the author. Although we are aware that the program needs to be tested by various composers, the first step in a larger project provided interesting and prospective results for the future work. We have also experimentally implemented the CAC program on three different platforms, Max, Web application, and Cocoa Application for macOS. Most of all, Internet technology is one of the most prospective platforms to broaden the range of possibility and to promote users to access the program. However, there are a lot of issues to be clear to realize the efficient and speedy system due to the data communication limits. The further development of the GUI is also the significant work in the future. The ideal GUI should enable users to access the synthesis system more intuitively, and easily control the result in more creative way as it is usually the case in sequencer software such as Logic Pro, Cubase, and Pro Tools.

The program is able to objectivize a musical idea by outputting audible and visible results of its analysis. It can possibly discover unpredictable aspects of a target sound, aspects which are otherwise hardly noticeable. One of the most profitable advantages of the program is that the database can be customized freely by the user. This can expand the application of this program for many purposes. We still have issues that the program enables to always produce reliable results for any kinds of sounds, however, the quality of the result is strongly influenced by the target and sources. Even so, the use of this program in its present stage of development has already significantly expanded and influenced the computer assisted composition and the traditional composition realms. The program users could often avoid personal habits or usual tendencies in the creation of music, and it has frequently brought a new musical point of view. It is like establishing a dialog with a partner who is quite savvy in the area of instrumentation. On the other hand, sometimes the re-synthesis produces results that are too strong, and from this, a form of prejudice for the creation of musical materials could arise. This can restrict ones imagination in the production of original ideas. It is therefore important not to persist in a blind employment of the re-synthesis, it is important to assume the results as a little part of a musical element that can be interpolated and developed with other ideas.

As discussed in the actual compositions, many compositional methods were used in order to employ the re-synthesis and, for instance, the re-synthesis often required some special compositional method, and in its turn, that new compositional method stimulated the re-synthesis of sound data in a different way. In our experience, this mural relationship between digital and analog processes can possibly change the musical point of view and help in the creation of new musical ideas. Needless to say, it is therefore important to seek out the target data and observe it from various perspectives in order to try and conceive what kinds of musical materials can be obtained from it, and those of which could be employed in our own music.

Finally, I believe that there is still some space for growth and development of the program in two directions: enhancing its quality and expanding its areas of application. The first is planned for relatively close future, and the goal is to distribute the program implemented on Cocoa application and apps for more practical devices such as smartphones and tablets. These portable devices would make the program available to divers types of composers. The second development would become a larger term project which demands the intersection of further fields research. In the next generation, the deep neural network is an essential technology which will take over many algorithms we discussed in this thesis. The deep learning is expected to realize the efficient identification system of the audio segments than ever. It can also be applied to the source separations system. I believe that this has so much potential for the application of the concatenative synthesis with a corpus audio data in many fields that restricting the destination, and to incorporate further fields of research will be absolutely meaningless. This, at the same time, is an essential part of the research on how could the program help to innovate in the creation of new music.

Part X
Bibliography

References

[1] Curtis Roads, "The Computer Music Tutorial", The MIT Press, Cambridge, Massachusetts, London, England, 1996

[2] Kubota K., "The music history", Tokyo Ongaku no tomo sha co., Tokyo Japan, 2002

[3] Takahashi H., Nakamura, T. "The history of the classical music", Tokyo publication Co. Japan, Tokyo Japan, 2004

[4] Black A. W.. and Paul T. CHATR : a generic speech synthesis system, 1994

[5] Roberto B., Anders F. "Emotion rendering in music : Range and characteristic values of seven musical variables", CORTEX A Journal Devoted to the Study of the Nervous System and Behavior Volume 47 October 2011 p1068 - 1081.

[6] Meinard M. "Fundamentals of Music Processing", Springer International Publishing Switzerland, 2015.

[7] N.Ahmed, T.Natarajan, and K.R.Rao, "Discrete cosine transform", IEEE Transactions on Computers, vol. C-32, pp.90-93, January 1974.

[8] David G. "Pitch Extraction and Fundamental Frequency: History and Current Techniques" Department of Computer Science Univeristy of Regina, CANADA, November, 2003

[9] Philip M., Geoff W., "A Smater Way to Find Pitch", Department of Computer Science University of Otago, 2005

[10] Brown J. C. , "Calculation of a constant Q spectral transform", J. Acoust. Soc. Am, 89(1):425-434, 1991.

[11] Benjamin B., "The Constant Q Transform", 1999.

[12] Brown J. C. , Puckette M. S. , "An efficient algorithm for the calculation of a constant Q transform", J. Acoust Soc. Am., Vol 92, No. 5 November 1992.

[13] FitzGerald D. , Granitch M. , Cychowski M. T., "Towards an Inverse Constant Q Transform", Dublin Institute of Technology, 120th Convension 2006 May, Paris France, 2006

[14] Oppenheim V. A., Shafer,W. R. "From Frequency to Quefrency: A History of the Cepstrum", IEEE Signal Processing Magnizine, 2004.

[15] Bogert P. B., Healy M.J.R. , and Tukey J.W., "The quefrency analysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-Cepstrum, and saphe cracking" in Time Series Analysis, M. Rosenblatt, Ed., 1963, ch. 15, pp.209-243.

[16] Atal S. B., "The History of Linear Prediction", IEEE Signal Processing Magnizine, 2006.

[17] Sturm L .B., Morvidone M. , Daudet L. , "Musical Instrument Identification using Multiscale Mel-Frequency Cepstral Coefficients", IEEE Signal Processing Magnizine, 2006.

[18] EronenA. , "Automatic Musical Instrument Recognition", Tampere University of Technology, Finland, April 2001.

[19] Saito S. , TakahashiK. , KameokaH. , NishimotoT., and SagayamaS. , "SPECMURT ANALYSIS OF MULTI-PITCH MUSIC SIGNALS WITH ADAPTIVE ESTIMATION OF COMMON HARMONIC STRUCTURE", ISMIR, 2005.

[20] Takahashi K. , Nishimoto T. , and Sagayama S., "Multi - Pitch Analysis Using Deconvolution of Log-Frequency Spectrum", 2003-MUS-53-13pp.6166 (2003).

[21] Saito S., Kameoka H. , Takahashi K. , Nishimoto T., and Sagayama S. "Specmurt Analysis of Polyphonic Music Signals", IEEE Transactions on Audio Speech and Language Processing 16, 2008.

[22] Azuma M. , Mitsuhashi W. "Automated Transcription for Polyphonic Piano Music with a focus on Harmonics in Log-Frequency Domain", IPSJ SIG Technical Report, Information Processing Society in Japan, 2011

[23] Kayeyama H., Saito S. Nishimoto T., Sagayama S., "Recursive Estimation of QUasi-Optimal Common Harmonic Structure Pattern for Specmurt Analysis: Piano-Roll Display Visualization and MIDI Conversion of Polyphonic Music Signal", IPSJ SIG Technical Report 2004-MUS-56 (7), 2004

[24] Grey M. J. and John W. Gordon, "Perceptual effects of spectral modifications on musical timbres", J. Acoust. Soc. Am., Vol.63, No.5, May 1978.

[25] Gerhard D., "Pitch Extraction and Fundamental Frequency: History and Current Techniques", Technical Report TR-CS 2003-06, June 2003.

[26] Oxenham J. A., "Pitch Perception", Jornal of Neurosceience 26 September 2012, 32(39).

[27] Trevor M. Shackleton and Robert P. Carlyon, "The role of resolved and unresolved harmonics in pitch perception and frequency modulation discrimination", J. Acoust Soc. Am. Vol. 95, No.6 June 1994.

[28] Oxenham A. J. , "The Perception of Musical Tones",The Psychology of Music Third edition p.1-p.33, 2013.

[29] Oxenham A. J., "Pitch perception",J Neurosci. 2012:32(39):1333513338, 2012.

[30] Banes P. , "Resolved and Unresolved Harmonics 341 Defining Resolvability", European Medical Alliance : http://www.europeanmedical.info/auditory-nerves/resolved-and-unresolved-harmonics-341-defining-resolvability.html, 2012.

[31] McLeod P. , Wyvill G. , "A SMATER WAY TO FIND PITCH", Proceedings of International Computer Music Conference, ICMC, 2005.

[32] Verteletskaya E. , Simak B. , "Performance Evaluation of Pitch Detection Algorithms", Ceske vysoke uceni technicke v Praze, FEL, 2009

[33] Alain de Cheveign e, Kawahara H.. "Yin, a fundamental frequency estimator for speech and music." Journal of the Acoustical Society of America, 111(4), 2002.

[34] Hess J. W., "Pitch Determination of Speech Signals" New York: Springer, 1993.

[35] Lutter M. , "Mel-Frequency Cepstral Coefficients" : http://recognize-speech.com/feature-extraction/mfcc : 2014

[36] Furui S. , " : We are creating a computer which has dialogue with a human being." Kadogawa Gakugei Public. Tokyo Japan, 2009

[37] Foote J. , "Automatic Audio Segmentation Using A Measure of Audio Novelty", FX Palo Alto Laboratory, Inc., Multimedia and Expo, 2000. ICME 2000. IEEE International Conference, July-August 2000.

[38] Robertson H. , "MIR tools: Self-similarity matrices and dynamic time warping", MUMT621 winter 2012, March 2012

[39] Finkelstein P. , "Music Segmentation Using Markov Chain Methods", March 2011

[40] Jothilakshmi S. , Palaniel S. , Ramalingam V. , "Unsupervised Speaker Segmentation using Autoassociative Neural Network", International Journal of Computer Applications (0975 - 8887) Volume 1 - No.7, 2010

[41] Lee DD. and Seung HS., "Learning the parts of objects by non-negative matrix factorization", Nature 401, pp.788-791, 1999.

[42] Lee DD. and Seung HS., "Algorithms for nonnegative matrix factorization", in Adv. NIPS, pp.556-562, 2000.

[43] Paatero P. and Tapper U. , "Positive matrix factorization : A non-negative factor model with optimal utilization of error estimates of data values", Environmetrics, vol.5 pp.111-126, 1994

[44] Kameoka H. , "The applciation of the Non-Negative Matrix Factorization in digital signal processing", The Acoustical Society of Japan, a bulletin Vo. 68-11,pp.559-565, 2012.

[45] Ono J. , "TFast Blind Source Separation with Auxiliary-function-based Independent Vector Analysis", The institute of electronics, information and communication engineers, IEICE Technical Report EA2013-5, 2013.

[46] Yamamoto R. , "Resolution of the NMF algorithm", `http://r9y9.github.io/blog/2013/07/27/nmf-euclid/`, 2013

[47] Benetos E. and Dixion S. , "A Shift-Invariant Latent Variable Model for Automatic Music Transcription" Computer Music Journal, 36(4) pp. 81-94, 2012

[48] Schmidt M. and Olsson R., "Single-Channel Speech Separation using Sparse Non-Negative Matrix Factorization", In International Conference on Spoken Language Processing (INTERSPEECH), 2006

[49] Nakashika T., Takiguchi T., Ariki Y. "NMF Matrix Generation Using Probabilistic Spectrum Envelope for Mixed Music Analysis", IPSJ SIG Technical Report Vol.2011-MUS-89 No.18, 2011

[50] Kohonen. T., Honkela, T., "Kohonen Network", Scholarpedia, http://www.scholarpedia.org/article/Kohonen_network, 2007

[51] Kohonen. T., "Self-organized formation of topologically correct feature maps", Biological Cybernetics, Springer-Verlog, 43:59-69, 1982

[52] Horowitz R., Alvarez L. "Self-organizing Neural Networks: convergence properties", IEEE International Conference Washington DC USA, 1996

[53] Rojas R., "Neural Networks A Systematic introduction", Springer Berlin, 1996

[54] Oja M. Kaski S., Kohonen T., "Bibliography of Self-Organizing Map (SOM) 1998-2001 Addendum", Neural Computing Surveys 3, 1-156, 2002

[55] Wu. Y., Takatsuka M., "The Geodesic Self-Organizing Map and Its Error Analysis", Computer Science 2005, Twenty-Eighth Australasian Computer Science, Australia, 2005

[56] Wu. Y., Takatsuka M., "Fast Spherical Self Organizing Map - Use of Indexed Geodesic Data Structure -", 5th Workshop On Self-Organizing Maps, Paris, 2005

[57] Montavon G., Samek W., Mueller K., "Methods for Interpreting and Understanding Deep Neural Networks", Berlin Germany, 2017

[58] Sarle W. S. "Stopped Training and Other Remedies for Overfitting", Proceedings of the 27th Symposium on the Interface, USA, 1995

[59] Hagan T. M., Demuth H. B., Beale M. H., Jesus D. O. "Neural Network Design 2ed Edition", USA, 2014

[60] Ishii K., Ueda S., Maeda E., Murase H, Takebu S., "", Ohmsha Public., Tokyo Japan, 2012

[61] Araki M. "The development of a Speech Recognition System", Morishita Public., Tokyo, Japan, 2013

[62] Everitt B., "The Cambridge Dictionary of Statistics", Cambridge, UK, New York, Cambridge University Press, 1998

[63] Schwarz D., "The Caterpillar System For Data-Driven Concatenative Sound Synthesis", Proc. of the 6th int. Conference on Digital Audio Effects (DAFx-3), London, UK, 2003.

[64] Strum L. B. "MATConcat : An Application for exploring concatenative sound synthesis using MATLAB", Proceedings ICMC 2004, 2004

[65] Schwarz D., "Concatenative Sound Synthesis: The Early Years",

[66] Schwarz D., Beller, G., Verbruggh, B., Britton, S., "Real-Time Corpus-Based Concatenative Synthesis With CATART", Expanded version 1.1 of submission to the 9 Int. Conference on Digital Audio Effects (DAFx-06), Montreal, Canada.

[67] Hackbarth B., Schwarz D., "AudioGuide: A Framework for Creative Exploration of Concatenative Sound Synthesis", 2011

[68] Hackbarth B., Schwarz D., "Composing Morphology: Concatenative Synthesis as an Intuitive Medium for Prescribing Sound in Time", Contemporary Music Review, Vol.32, No. 1, 49-59,2013

[69] Sturm L. B., "Adaptive Concatenative Sound Synthesis and Its Application to Micromontage Composition", Computer Music Journal, 30:4, pp. 46-66, Winter 2006, 2006

[70] Schwarz D., Robel A., Yeh C., LaBurthe A., "Concatenative Sound Texture Synthesis Methods and Evaluation", Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16), Brno, Czech Republic, September 5-9, 2016

[71] Zils A., Patchet F., "MUSICAL MOSAICING", Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01), Limerick, Ireland, December 6-8,2001

[72] Robert Silvers :

http://www.photomosaic.com

[73] EC-Council, "Investigating Networking Intrusions and Cybercrime", Ec-council press Series: Computer Forensics, Cengage Learning, 2009

[74] Apple Inc. "vDSP" Web `https://developer.apple.com/documentation/accelerate/vdsp`

[75] Apple Inc. "Accelerate Frameworks" Web `https://developer.apple.com/documentation/accelerate`

[76] T. Oura "General Purpose FFT (Fast Fourier/Cosine/Sine Transform) Package" Web http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html¿December 2016

[77] MathWorks "MATLAB" Web https://www.mathworks.com/products/matlab.html 1994 - 2017

[78] Num Focus "Numpy" Web http://www.numpy.org 2017

[79] Num Focus "Julia" Web https://julialang.org 2017

[80] Sonote http://www.y2lab.com/project/sonote/ 2017

[81] Synful http://www.synful.com/SynfulOrchestra.htm 2017

[82] Orchidee http://repmus.ircam.fr/orchidee 2017

[83] Orchids http://repmus.ircam.fr/_media/esling/orchids-documentation.pdf 2014

[84] AudioGuide http://www.benhackbarth.com/audioGuide/index.html 2017

[85] OpenMusic http://repmus.ircam.fr/openmusic/home 2015

[86] Max https://cycling74.com 2017

[87] SuperColider http://supercollider.github.io 2017

[88] Adobe won't support Flash on Android 4.1 https://www.engadget.com/2012/06/28/adobe-confirms-it-wont-support-flash-on-android-4-1/ 2012

[89] Audacity http://www.audacityteam.org 2017

[90] The HTML5 Audio Editor http://audioeditor.wikiaudio.org 2017

[91] The HTML5 differences from HTML4 https://www.w3.org/TR/html5-diff/ 2014

[92] Real-Time Audio Mosaicing http://imtr.ircam.fr/imtr/Real-Time_Audio_Mosaicing 2011

[93] Recorderology http://recorderology.com 2015

[94] Mayer-Spohn U., Takahashi K., "Recorderology - development of a web based instrumentation tool concerning recorder instruments" International Computer Music Conference 2016, Utrecht The Netherland, 2016

[95] Thoughts on Flash https://www.apple.com/hotnews/thoughts-on-flash/ 2010

[96] World Wide Web Consortium https://www.w3.org/TR/webaudio/ 2015

Acknowledgements

I would like to express my greatest appreciation to Professor Erik Oña for his continuous support throughout my Ph.D. studies from 2014 to 2017 and all the related research. He has constantly encouraged me to gain access to a wide variety of scientific and artistic topics.

I would also like to give special thanks to the Foundation Christoph Delz which awarded me a special grant for my both artistic and research activities in 2014.