Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master's Thesis

Towards automated segmentation of repetitive music recordings

submitted by
Zhe Zuo

submitted
August 1, 2011

Supervisor / Advisor
Priv.-Doz. Dr. Meinard Müller

Reviewers
Priv.-Doz. Dr. Meinard Müller
Prof. Dr. Michael Clausen

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum / Date) (Unterschrift / Signature)

Acknowledgements

First of all, I would like to express my greatest gratitude to my supervisor Priv.-Doz. Dr. Meinard Müller, whom offers me this so interesting topic and gave me a lot of advices and sources. It is him who teaches me how to do researches. Without his enlightening instruction and patience, I could not have completed my thesis.

Next, greatly thanks to Peter Grosche. He always helped me on all kinds of problems patiently and gave valuable supports even when he was busy. It is him who help me to be familiar with the MATLAB and the sources.

I would like to express my great gratitude to many of my friends who support me during writing of my thesis. Thomas Prätzlich, Jonathan Driedger have patiently helped to proofread my thesis. I am grateful to them for giving me the detailed and very fast feedback. I would like to specially thank to Nanzhu Jiang whose experiences helped me a lot. And also thanks to her for proofreading my thesis patiently and carefully.

Last but not least, I want to thank my parents for their unlimited support and encouragement. Finally, I cannot thank enough my wife Yuxin Gao. Thanks to her for encouraging and accompanying me during my work.

Abstract

The principle of repetition is of central importance in music and the segmentation of music recordings into recurrent patterns constitutes an important task in the field of music information retrieval. In this thesis, we consider the special case of folk songs, which often consist of a large number of stanzas. Here, a tune is repeated over and over again with changing lyrics and possibly variations in melody and intonation. To deal with such variations, we present and compare two different segmentation procedures. In the first procedure, we assume the availability of a reference stanza given in form of a symbolic score representation. This reference stanza is then locally compared with the audio recording by means of a local variant of dynamic time warping to derive the segmentation results. In the second procedure, we introduce a reference-free segmentation approach which does not require any additional information. Here, the idea is to automatically derive the recording's most representative passage, which then plays the role of the reference. Our experiments show that our reference-free procedure yields segmentation results similar to the reference-based procedure. As a further contribution of this thesis, we show how the segmentation results can be used for music visualization and navigation applications. In particular, we describe several novel functionalities, which have been implemented as plug-ins for the SyncPlayer framework.

Contents

Chapter 1

Introduction

1.1 Repetitions in Music

The principle of *repetition* plays a particularly important role in music [13]. Recurrent patterns, which may be of rhythmic, harmonic, or melodic nature, evoke in the listener the feeling of familiarity and understanding of the music (see [4, 10]). Many kinds of music, for example pop music, contain literal and direct repetitions, making the music more accessible. But some other kinds of music like classical music contain varied and transformed repetitions [1].

As an example, we have a look at the well-known classical music *Für Elise* (WoO 59[1]), which is composed by *Ludwig van Beethoven*. This piece of music has the musical form $A_1BA_2CA_3$ consisting of three repeating A-parts, a B-part, as well as a C-part. The musical parts are indicated by capital letters such as X, where all repetitions of X are enumerated as X_1, X_2, an so on [14]. In this example, A-part has been repeated another two times. It is considered to be the most significant part. In fact, people often associate this music with the A-part. That is the effect of repetitions.

1.2 Music Segmentation Task

The general goal of music structure analysis is to divide an audio recording into temporal segments corresponding to musical parts and then to group these segments into musically meaningful categories. Considering the importance of repetitions in music, extracting the repetitive structure from a given audio recording, which often closely correlates to the musical form of the underlying piece of music, is a central subtask in music structure analysis.

In this thesis, we consider the special case that the given piece of music basically consists of the repetition of a single part, like it can be observed in many folk songs. Folk songs are considered as a typical kind of repetitive music. Common people sing folk songs, which

[1] WoO is a list of all the compositions of *Ludwig van Beethoven* that were not originally published with an opus number, or survived only as fragments [23]

have been passed down by oral tradition, during work or social activities [21]. Folk songs are very important resources for studying the culture and history. They often consist of a large number of stanzas. Here, a tune is repeated over and over again with changing lyrics and possibly variations in melody and intonation. For example, the Dutch folk song, named OGL49313, has a structure of $A_1A_2A_3A_4A_5$. It consists of five repeating A-parts.

1.3 Challenges of Segmentations

Even though the segmentation for our special case seems to be an easy task, there are a number of challenges one has to face when working with real audio recordings.

First of all, the notion of repetition is a rather ill-defined one. Actually, repeating parts are never performed the same way but may differ significantly in dynamics, instrumentation, execution of note groups, modulation, articulation, and tempo. For example, when dealing with field recordings of folk songs, non-professional singers often deviate significantly from the expected pitches and have serious problems with the intonation. Even worse, their voices often fluctuate by several semitones downwards or upwards across the various stanzas of the same recording [15].

Furthermore, there may be significant relative tempo differences between repeating parts [16]. One also often has to deal with changes in instrumentation, for example, the melody instrument may change throughout the recording or additional accompanying instruments may join at a certain point playing secondary voices or ornamental elements.

It is our goal to introduce automatic algorithms for audio structure analysis that can find the repetitive structure even in the presence of large variations in these parameters.

1.4 Folk Song Collection

In this thesis, the experimental dataset consists of 47 Dutch folk song recordings. They are named as *Onder de groene linde (OGL)* and *Nederlandse Liederenbank (NLB)* with additional number indices.

These folk songs basically have the musical form $A_1A_2...A_K$ consisting of K repeating A-parts. They are performed by elderly non-professional singers in a poor recording condition such that some of them are in poor quality. For example, sometimes singers forgot what to sing, they stopped to talk or jumped to the next section. Even worse, there are some loud noises, like bird singing, in the background during the recording.

1.5 SyncPlayer

The segmentation results can be used for music visualization and navigation applications. We extend the *SyncPlayer* which is an advanced audio player for multimodal presentation of high quality audio and associated music-related data [11]. It has been extended with

some useful plug-ins such as the *Audio Switcher* and the *Audio Structure*. People can get more information of the music by using the SyncPlayer. For example, chord information or structure information.

Based on the SyncPlayer and its plug-ins, the *Interpretation Switcher*, a new plug-ins of the SyncPlayer, is developed. With some extra extensions, we integrate some functions which make it more convenient and powerful. It is a useful tool to represent *music information retrieval (MIR)* results. For example, with extension of user interaction, one can correct the segmentation result in the application directly.

1.6 Contributions

In this thesis, we introduce a reference-based segmentation procedure which is used for segmenting the repetitive music recordings with a reference. Then, based on the reference-based segmentation, we developed a reference-free segmentation approach which does not require any additional information. Our experiments show that our reference-free procedure yields segmentation results similar to the reference-based procedure. Further more, we describe several novel functionalities, which have been implemented as plug-ins for the SyncPlayer framework. With this implementations, one can visually represent the segmentation results. It can be used for some other music information retrieval results, such as the chord recognition.

1.7 Organization of Thesis

The organization of this thesis is as follows.

- In Chapter 2, *feature extraction* is introduced. It is the transformation of a music signal into a feature representation.
- In Chapter 3, we introduce *Dynamic Time Warping (DTW)*, which is the basic technique to compute the distance between two time-dependent sequences. The distance function based on it is used in our music segmentation method throughout this thesis.
- In Chapter 4, we introduce a *reference-based segmentation* procedure, which can automatically segment the repetitive music recordings with a manually generated MIDI reference.
- In Chapter 5, we develop a *reference-free segmentation* procedure to segment audio recordings. It is based on the reference-based segmentation procedure, which automatically derive the recording's most representative passage to replace the MIDI reference. The result of our experiments is represented and discussed in detail.
- In Chapter 6, we describe the *SyncPlayer* framework which is an advanced audio player. People can use it to represent MIR research results.

- In Chapter 7, we develop the *Interpretation Switcher* based on the SyncPlayer framework. And we enriched the functionality of the Interpretation Switcher.
- In Chapter 8, we give a java documentation of the Interpretation Switcher.
- In Chapter 9, we give a summary of the thesis and discuss future work.

Chapter 2

Feature Extraction

In MIR, feature extraction is commonly used. It is the transformation of a music signal into a feature representation. With a suitable feature representation, some musical uninteresting aspects for certain researches can be excluded, e.g., the loudness information. In order to segment audio recordings, we need to transform the original data into a suitable representation.

In this chapter three kinds of features are introduced. Pitch feature is a basic technique. From pitch features, we build chroma feature. Then from chroma feature we describe the CENS feature which is finally used for music segmentation in this thesis.

2.1 Pitch Features

Pitch is a perceptual attribute which allows the ordering of sounds on a frequency-related scale extending from low to high [6]. Pitch feature are a technique to decompose an audio signal into frequency bands. We can transfer musical notes into the corresponding MIDI pitch p. A note corresponds to a frequency band but not single frequency value. In order to define the frequency bands, we first define the center frequency of a note by the following equation [14]:

$$f(p) = 2^{\frac{p-69}{12}} \cdot 440 \qquad p \in [1:120] \tag{2.1}$$

The center frequency $f(p)$ denotes the frequency which is in the center of the frequency band. We can calculate the center frequency of pitch No.69, which denote note A4. The center frequency of it equals to 440 Hz. With this formula we can also get $f(57) = 220$, which is note A3. It shows that the center frequency of A3 is half of A4. Actually the center frequency of a note is always half of the center frequency of the note that is 12 higher than it. Using the center frequency we can calculate the frequency bands of the note. The bandwidth w is defined as follows [14]:

$$w = f(p)/Q \tag{2.2}$$

Here, Q is a factor which denotes the ratio of the center frequency to the bandwidth. As an example, $w - 440/25 = 17.6Hz$ with Q = 25. So the left boundary of A4 is $440 - 17.6 = 431.2Hz$ and the right boundary of A4 is $440 + 17.6 = 457.6Hz$. We can find that the bandwidth for different notes is not the same since it depends on the center frequency. The higher the pitch of note is , the wider the bandwidth is.

Until now, we talked about continuous time signals. The signal, which we are working on, is discrete time signals. The sampling rate is an important parameter. For instance, if the input signal is discrete and sampled with a rate of 4410Hz. The center frequency of note A4 becomes $440/4410 \approx 0.0998$. Then the bandwidth $w = 0.0998/25 \approx 0.004$. The left boundary is 0.0978 and the right boundary is 0.1018.

Then we can establish a filter bank with the frequency band information of the notes.

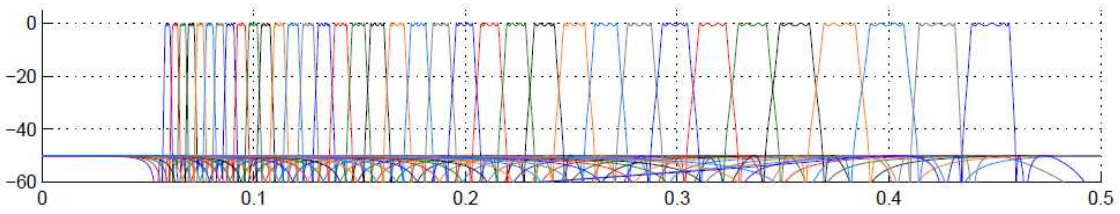


Figure 2.1. Pitch filter bank for $p \in [60 : 95]$ with sampling rate 4410Hz [14].

Figure 2.1 shows a pitch filter bank. With a suitable pitch filter bank, we can extract the pitch subband corresponding to a certain note. In order to use the information, We need a measurement function. The local energy is used here.

2.2 Local Energy(STMSP)

In the last section, we discussed how to decompose a music signal into different frequency bands corresponding to the pitch information. But this is not enough, We need a method to measure the value inside a frequency band. Local energy is a good choice.

Energy is a natural parameter of music signals. When we play or sing a note, it is a process that we produce energy. A new note will always lead to an increase in the signal's energy [5], especially for instruments for which the sound of a note is loudest immediately after it is played. For example, the piano and the drum. Beating a drum results in a sudden energy increase. These energy changes occur in the pitch bands corresponding to the fundamental frequents and harmonics of the respective note. According to these theories, we can calculate the local energy of each frequency band to detect the candidates for note onsets (see [18]).

Short-Time Mean-Square Power(STMSP) is used to calculate the local energy for the pitch

subbands. The formula of the local energy is defined as follows [14]:

$$E_x(n) = \sum_{k \in [n - \lfloor \frac{w}{2} \rfloor : n + \lfloor \frac{w}{2} \rfloor]} |x(k)|^2 \tag{2.3}$$

$E_x(n)$ is the local energy of x at the point n where $n \in \mathbb{Z}$. And window length $w \in \mathbb{N}$ is with a fixed size and x is a subband signal, In order to improve the efficiency, instead of computing the local energy for all samples, we evaluate every d samples where $d \in \mathbb{N}$. The factor d is called window step and usually equals to the half of the window length. For example we choose window step $d = 50$ for $w = 101$. In this case the local energy calculation has 50 percent overlap between two neighbors. We can control the overlap area by changing the window step. After having computed the local energy, we apply logarithmical decibel scale which makes the result clearly, since the STMSP values have indistinct differences.

Figure 2.2 shows an example of the STMSP for each specified pitch. The colors denotes the STMSP values according to the logarithmical decibel scale. The dark colors indicate that the value at the corresponding time point and pitch is very low. Comparing to the score of this audio piece, the STMSP values are very high at the onsets of notes and the energy changes only occur on the corresponding pitch and the harmonics. With calculating the local energy and representing the result on the diagram, we can roughly find out which notes are played and when they are played.

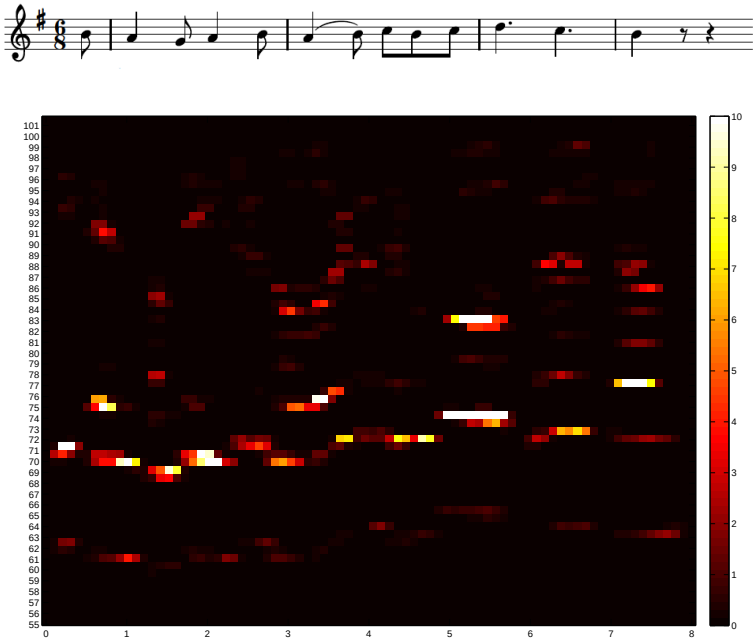


Figure 2.2. Pitch plot for folk song OGL49313. The columns to time measured in samples with a time resolution of 0.1s

2.3 Chroma Features

In Figure 2.2, we can find the energy exists not only at the corresponding frequency band, but also the harmonic frequency bands. These energies interfere our experiment result. As the example in the figure, although there are many noises, we can still find some harmonics. The energy changes might be misconceived as a played note. We want to exclude these noises during our experiment.

Shepard reported two distinct attributes of pitch, tone height and chroma, in order to reduce the complexity of the pitch representation [20]. He found that the human auditory system perceives the pitch represented as a helix was perceived better than as a one-dimensional line. The idea is the pitch can be separated into tone height and chroma (see [2]). Figure 2.3 shows a chromatic circle which has been annotated with chroma names are on the bottom. As they are introduced in last section, the center frequency of every 12 notes increase to be twice higher. To be consistency with it, the chromatic circle are divided into 12 parts, which denote 12 chroma. The chroma set is $\{C, C^\sharp, D, ..., B\}$. For each chroma, there is a pitch class, which is a set of all pitches sharing the same chroma. For instance, the pitch class of chroma D is $\{..., D0, D1, D2, D3, ...\}$. Chroma features can account for the close octave relationship in both melody and harmony [2].

One can also use higher-dimensional versions of chroma features that refine the chroma scale corresponding to certain fractions of semitones [8]. This allows for handling tuning differences that are fractions of semitones. For example, considering one-half or one-third semitone steps, one obtains 24-dimensional or 36-dimensional chroma features.

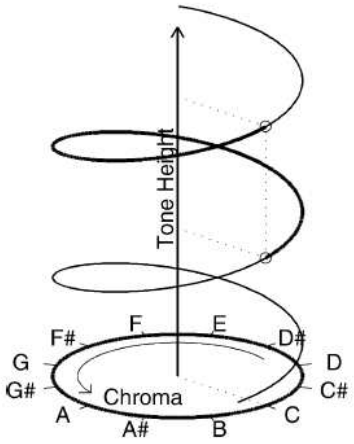


Figure 2.3. Illustration of Shepard's helix of pitch perception. The vertical dimension is tone height, while the angular dimension is chroma [12].

With the basic information of chroma and STMSP, we can use chroma feature to represent the audio recording with the following steps:

- 1. Decomposing an audio signal into 88 pitch subbands.

- 2. Calculating STMSP value for each pitch subband.
- 3. Adding up all STMSPs which belong to the same pitch class.

Then we can generate a chromagram for the audio recording as the Figure 2.4. As the color scale in the right part, blue means the STMSP value is very low while red means the STMSP value is very high. For each STMSP window, there is a 12-dimensional vector $v = (v(1), v(2), ..., v(12)) \in \mathbb{R}^{12}$. Here, $v(1)$ corresponds to chroma C, $v(2)$ to chroma $C^\sharp$, and so on.

In order to absorb differences in the sound intensity or dynamics, we normalized the chroma representation. The vector v will be replaced by $v = \|v\|_1$. And $\|v\|_1 := \sum_{i=1}^{12} |v(i)|$ denotes the $l^1 - norm$ of v. Sometimes random energy distributions exist in the music signal, for example in passages of silence at the beginning and ending of music. We replace the chroma vector v by the uniform distribution in case $v = \|v\|_1$ falls below a certain threshold. Note that there is a light blue area at around 30s in Figure 2.4. That is a part where the singer forgot what to sing next so that she stopped and kept silence for a while.

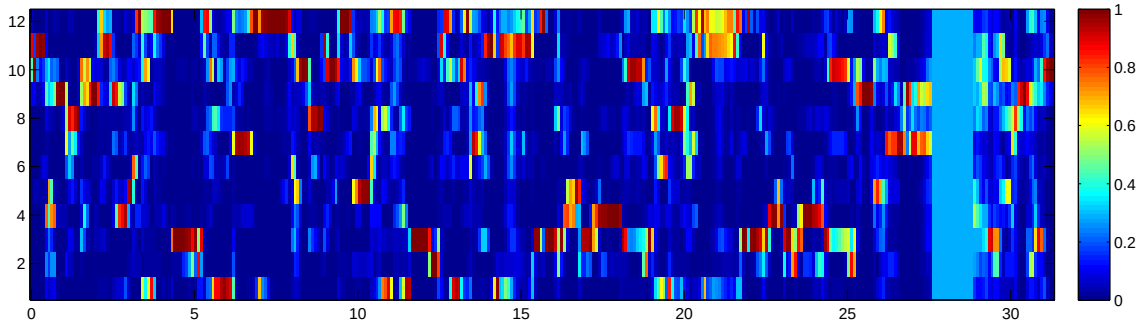


Figure 2.4. Chroma representation of the first stanza of folk song OGL49313

2.4 CENS Features

By transferring the audio signal into chroma feature, the information contains in the audio becomes easier to analyze. But we can still reduce some noise information and make it even clearer. The chroma energy normalized statistics (CENS) [17] has been developed by Mueller, Kurth and Clausen. CENS feature is helpful for our further study like audio matching.

The CENS feature is based on the chroma feature. So we do some further operation on chroma feature. We quantify the chroma vector v from [0,1] to 0,1,2,3,4 following the equation [14]:

$$\tau(a) = \begin{cases} 0 & for \quad 0 \leq a < 0.05, \\ 1 & for \quad 0.05 \leq a < 0.1, \\ 2 & for \quad 0.1 \leq a < 0.2 \\ 3 & for \quad 0.2 \leq a < 0.4, \\ 4 & for \quad 0.4 \leq a < 1 \end{cases} \tag{2.4}$$

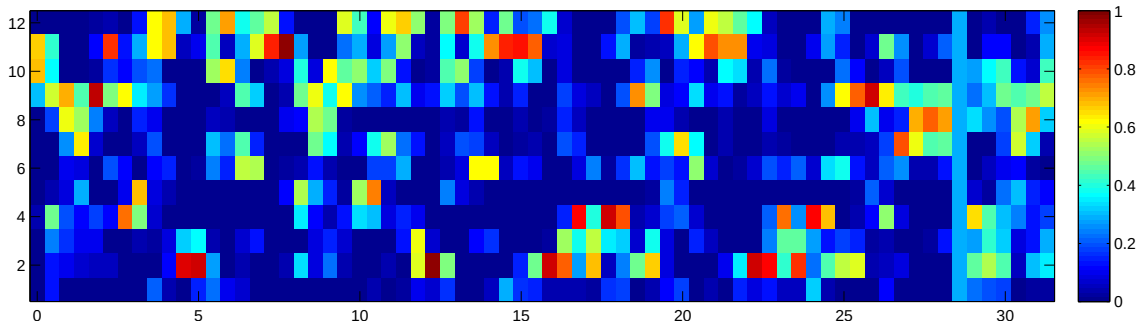


Figure 2.5. CENS(11,5) representation of the first stanza of folk song OGL49313

With this function, we compute $\tau(v_n) := (\tau(v_n(1)), ..., \tau(v_n(12)))$. After that the chroma energy distribution vector has been classified into several values. We have already reduced the variance of the vectors and we can do even further.

Window length w and downsampling rate d are defined for $CENS(d, w)$. We use a hann window with window length w for smoothing first and then we downsample the result. Sometimes there are some local errors inside the audio recording which will interfere our final result. By smoothing the vectors, these local errors can be reduced. But the problem is that the data accuracy will decrease at the same time. Therefore, to choose a suitable window length is very important and it depends on the data collection. The purpose for downsampling is straightforward. It can improve the efficiency of the program. For example, for a feature with sampling rate of 10Hz, we downsampling it with $d = 5$. Then the sampling rate will reduce to 2Hz which means two CENS vector per second. This is only 20 percent of the original data set.

Comparing the Figure 2.5 and the Figure 2.4, we can easily see the difference between the CENS feature representation and the chroma representation for the same audio recording. The CENS feature representation contains less information which makes it more clearly. According to the downsampling rate, the silence part at around 30s in the recording, which was made by mistake, becomes smaller in CENS feature representation. It reduces the influence of this part comparing to chroma representation.

Chapter 3

Dynamic Time Warping

Dynamic Time Warping (DTW) is a well-known technique to find an optimal alignment between two given (time-dependant) sequences under certain restrictions [14]. In this thesis, a distance function, which is based on DTW, is developed for comparing the reference with the audio recording.

We introduce the algorithm of the classical DTW in Section 3.1. Then the subsequence DTW which has been used for building up the distance function, is described in Section 3.2.

3.1 Classical DTW

First we define two music pieces, which could be considered as two time-dependent sequences, X and Y. Here, $X := (x_1, x_2, ..., x_N)$ with $n \in \mathbb{N}$ while $Y := (y_1, y_2, ..., y_M)$ with $M \in \mathbb{N}$. These two sequences are transferred into feature representation. The local cost measure $c(x, y)$ denotes the distance between x and y. Generally speaking, a low value of $c(x, y)$ means high similarity between x and y. By calculating the cost for each pair of elements of sequences X and Y, a two-dimensioned cost matrix $C \in \mathbb{R}^{N \times M}$ can be established. It is defined as $C(N, M) := c(x_n, y_M)$. For example, we define a local cost $c(x, y) = |x - y|, x, y \in \mathbb{R}$. Let $X = (1, 5, 2, 2, 3)$ and $Y = (1, 2, 5, 2)$. We can get the cost matrix as the Figure 3.1.

3	2	1	2	1
2	1	0	3	0
2	1	0	3	0
5	4	3	0	3
1	0	1	4	1
	1	2	5	2

Figure 3.1. An example of a cost matrix

Based on the cost matrix, one can find the $(N, M) - warping$ path of the two sequences.

A warping path is a sequence $p = (p_1, ..., p_L)$ with $p_l = (n_l, m_l) \in [1 : N] \times [1 : M]$ for $l \in [1 : L]$ satisfying the following three conditions [14]:

- 1. *Boundary condition:* $p_1 = (1, 1)$ and $p_L = (N, M)$.
- 2. *Monotonicity condition:* $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$.
- 3. *Step size condition:* $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L - 1]$

These three basic rules restrict the warping path selection. A warping path should not contain backward parts, the starting point should correspond to the first elements of the two sequences, and the ending point should correspond to the last element of two sequences.

The total cost $c_p(X, Y)$ of a warping path p between X and Y is defined as [14]:

$$c_p(X, Y) := \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}) \tag{3.1}$$

The distance $DTW(X, Y)$ is the minimal total cost for all $(N, M)-warping$ path between sequences X and Y. Now we can compute the $DTW(X, Y)$ for the example mentioned in Figure 3.1. The result shows in Figure 3.2. And the total cost $c_p(X, Y) = 2$. The gray area shows the optimal warping path which leads to minimal total cost.

3	2	1	2	1
2	1	0	3	0
2	1	0	3	0
5	4	3	0	3
1	0	1	4	1
	1	2	5	2

Figure 3.2. An example of DTW

However, it is very slow to compute all the possible paths and select the one with the minimal total cost. In order to find the optimal warping path based on a cost matrix efficiently, we can use an accumulated cost matrix. With an accumulated cost matrix, we transfer the computation from the global total cost into the local total cost. The prefix sequences of X and Y is used for computing the DTW value. We define $X(1 : n) := (x_1, ..., x_n)$ for $n \in [1 : N]$ and $Y(1 : m) := (x_1, ..., x_m)$ for $m \in [1 : M]$. The accumulated cost is defined as following [14]:

$$D(n, m) := DTW(X(1 : n), Y(1 : m)) \tag{3.2}$$

By the equation, the accumulated cost $D(n, m)$ is actually the DTW distance between $X(1 : n)$ and $Y(1 : m)$. Using an accumulated cost matrix, one can compute $D(n, m)$

from the pervious computation, which makes the cost of DTW be very low. $D(n,m)$ can be calculated by the formula [14]:

$$D(n,m) = \begin{cases} c(x_1,y_1) & for \quad n=1, m=1, \\ D(n-1,m)+c(x_n,y_m) & for \quad n\neq 1, m=1, \\ D(n,m-1)+c(x_n,y_m) & for \quad n=1, m\neq 1, \\ min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\}+c(x_n,y_m) & else \end{cases} \tag{3.3}$$

The accumulated cost matrix shows the minimal total cost directly, since $D(N,M)$ is the one we are looking for. The only task is to extract the path which leads to this minimal total cost. This is done via backtracking. Using the pervious example again, the Figure 3.3 shows the accumulated cost matrix. The green arrows show the backtracking path and the red ones show the optimal warping path. $p^* = p_1, ..., p_L$ denotes the optimal warping path.

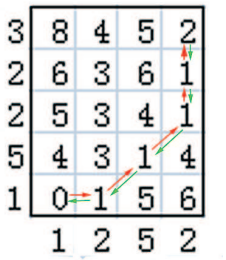


Figure 3.3. An example of accumulated cost matrix and the optimal warping path.

DTW is flexible, one can change some parameters of it such as the step size and local weights (see [19]). Using the variation of the step size as an example. The normal step size $\Sigma = \{(1,0),(0,1),(1,1)\}$ can be replaced by $\Sigma = \{(1,2),(2,1),(1,1)\}$. The accumulated cost matrix D can be computed with $min\{D(n-1,m-1), D(n-1,m-2), D(n-2,m-1)\}+c(x_n,y_m)$ for $n \in [2:N]$ and $m \in [2:N]$. And we set $D(0,0) := 0$, $D(1,1) := c(x_1,y_1)$, $D(n,0) := \infty$ for $n \in [1:N]$, $D(n,1) := \infty$ for $n \in [2,N]$, $D(0,m) := \infty$) for $m \in [1:M]$ and $D(1,m) := \infty$ for $m \in [2:M]$. Using this step size, some of the elements in X and Y might be omitted and do not cause any cost. And the lengths N and M should only differ at most by a factor of two which also depends on the lengths of them. Otherwise, one can not find a warping path or the total cost would be infinity.

3.2 Subsequence DTW

The normal DTW and variations of it are comparing two sequences from the starting point to the ending point. But sometimes one sequence could be much longer than the other. For example, one sequence is a stanza of music and another one is the whole piece of the music. Computing DTW distance between these sequences is meaningless in this case. Therefore, we focus on finding a optimal subsequence, which has the highest similarity

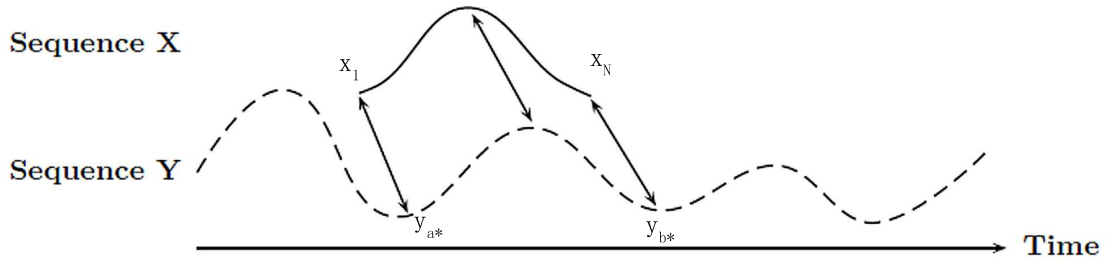


Figure 3.4. Optimal time alignment of the sequence X with a subsequence of Y [14].

to the short sequence, within the longer sequence. Subsequence DTW is used for the segmentation strategies which will be introduced in the following chapters.

Let $X := (x_1, x_2, ..., x_N)$ and $Y := (y_1, y_2, ..., y_M)$ where M is much bigger than N. As Figure 3.4 shows, the goal is to find a subsequence $Y(a^* : b^*) := (y_{a^*}, y_{a^*+1}, ..., y_{b^*})$ where $1 \leq a^* \leq b^* \leq M$ [14]. It is clear that the strict rules of $p(1,1)$ and $p(N,M)$ must be the beginning and the ending of warping path has been broken. Instead, the beginning point y_{a^*} and ending point y_{b^*} are not fixed. Our task is not finding the optimal warping path but finding the optimal subsequence. Here, the optimal subsequence is the one that leads to the minimal DTW distance comparing to sequence X.

The accumulated cost matrix D should be modified in order to fit the new task. Instead of using $D(1,m) := \sum_{k=1}^{m} c(x_1, y_k)$, $D(1,m) := c(x_1, y_m)$ is defined. The others stay the same. One can determine b^* as following [14]:

$$b^* := \underset{b \in [1:M]}{argmin} D(N, b) \tag{3.4}$$

After having computed the b^*, one can apply the inverse optimal warping path to determine a^*. We start with $p_L = (N, b*)$, and find the maximal a^* where $a^* \in [1 : b^*]$.

A distance function can be developed with respect to the DTW distance by the equation [14]:

$$\Delta : [1 : M] \to \mathbb{R}, \qquad \Delta(b) := D(N, b) \tag{3.5}$$

The distance function specifies the minimal distance between a certain point of a sequence and the reference sequence.

Chapter 4

Reference-based Segmentation

In this chapter, we introduce a procedure, developed by Müller, Grosche and Wiering, for segmenting the folk song recording that consists of several repeating stanzas (see [15]). The MIDI reference is used for segmenting the music, which has been generated by some experts manually. Therefore, we call this procedure as reference-based segmentation in this thesis.

We begin with introducing the reference-based segmentation procedure in Section 4.1. After that, in Section 4.2 we represent the result of the experiment.

4.1 Procedure

The reference-based segmentation procedure is as follows. Firstly, applying feature extraction to transform the MIDI reference and the audio recording into chroma representation. Secondly, locally comparing the reference with the audio recording be means of a suitable distance function, we can generate a matching curve. Thirdly, we use a greedy strategy to deriving the segmentation from local minima of the distance function. Finally, we apply some evaluation functions to evaluate the segmentation results.

4.1.1 Feature Extraction

In Müller, Grosche and Wiering's paper, the MIDI references are used for segmenting recordings. In the folk song dataset, the MIDI references are a standard stanza of the audio recording. In feature extraction step, both of the MIDI reference and the audio recording are transformed into a same mid-level representation. Here, the CENS feature representation which is summarized in Section 2.4 has been used.

4.1.2 Matching Method

A distance function is used for computing the distance between MIDI reference and subsequences of audio recordings (see [15]). Let $X = (X(1), X(2), ..., X(K))$ to be the sequence

of MIDI reference and $Y = (Y(1), Y(2), ..., Y(L))$ to be the sequence of the audio recording. The distance function is defined as $\Delta := \Delta_{X,Y} : [1 : L] \to \mathbb{R} \cup \infty$ with respect to X and Y using a variant of DTW which has been introduced in Chapter 3:

$$\Delta(l) := \frac{1}{K} \underset{a \in [1:l]}{min} (DTW(X, Y(a : l))) \tag{4.1}$$

$Y(a : l)$ is a subsequence of Y which is from a to l, $1 \leq a \leq l \leq L$. Here, the $DTW(X, Y(a : l))$ refers to the DTW distance between X and $Y(a : l)$ with a step size $\Sigma = \{(1, 2), (2, 1), (1, 1)\}$. Then normalize the minimal DTW value. As the equation defined, $\Delta(l)$ refers to the smallest distance between X and $Y(a : l)$, and it can be easily recovered within the DTW computation procedure. $\Delta(l)$ is used for representing distance value of the point l. A low $\Delta(l)$ value means we can find a $Y(a : l)$ which has high similarity to X.

Until now, the local minima of the matching curve refers to the ending point of a segment. In order to make it clear, we modify the algorithm to find the starting point of a segment. The idea is before computing the matching curve, we left-right flip the sequences of audio recording and MIDI reference. With this operation, sequences X and Y are converted to be $X = (X(K), X(K - 1), ..., X(1))$ be the sequence of MIDI reference and $Y = (Y(L), Y(L-1), ..., Y(1))$. We locally compare the flipped reference with the flipped audio recording by the original distance function. Using those flipped sequences, we can derive a backward-ordered matching curve. The local minima in the matching curve indicates the ending point of the segment for the backward-ordered sequences of the audio recording. Then, we left-right flipping the matching curve to get a forward-ordered matching curve. With this operation, the local minima refers to the starting point of the segment.

A matching curve of folk song OGL49313 is shown in Figure 4.1. It is computed by reference-based segmentation. In the figure, the horizontal time axis scaled as seconds and the vertical cost axis scaled from 0 to 1. The horizontal black lines refer to the segments which are derived by the segmentation procedure. The green vertical lines indicate starting points of segments, which have been annotated manually. As it is introduced, lower $\Delta(l)$ value corresponds to higher similarity. Comparing the matching curve and the ground truth annotation, we can find that reference-based segmentation preforms pretty well for folk song OGL49313. The local minima matches the ground truth annotation. Considering the length of MIDI reference, the subsequences at the end of recordings are too short to compare with MIDI reference. Therefore, the tail of the matching curve goes up to 1, which means that part is not comparable with the reference.

4.1.3 Segmentation Method

Greedy segmentation strategy is used for finding the segmentation points. Firstly, we set the point $l \in [1 : L]$, which corresponds to the global minima value in the matching curve, to be the start of a segment. Meanwhile, the ending point e_l is derived. The interval $S := [l : e_l]$ constitutes the first segment. Secondly, to avoid large overlaps between the various segments to be computed, we exclude a $[L_l{:}R_l] \subset [1 : L]$ around the index l from further consideration. In this thesis we set $L_l := max(1, l - \frac{3}{5}K)$ and $R_l := \min(\mathrm{L}, l + \frac{3}{5}\mathrm{K})$,

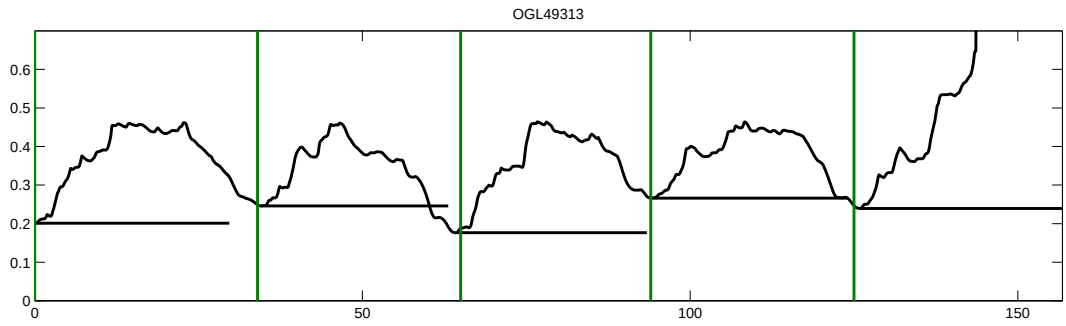


Figure 4.1. Matching curve of folk song OGL49313.

thus K denotes the length of the segment. We set $\Delta(m) = \infty$ for m $\in [L_l : R_l]$. And then repeat the same procedure to determine the next segment. The procedure is repeated until all values of the modified Δ lie above a quality threshold $\tau > 0$. Let N denote the number of resulting segments, then S_1,S_2,...,S_N constitutes the segmentation result. Finally segments are ascending sorted respect to the starting time of them.

4.1.4 Enhancement Strategies

Considering the problems described in the Section 1.3. The tempo changes have been solved by applying the distance function. But the intonation difference still influences the segmentation result. One method to solve it is to compute the distance not only with the original chromagram, but also with the shifted chromagram. A chroma vector contains 12 different pitches. We can circularly shift the energy distribution from one pitch to the upper pitch. For example the original energy located in the note C, and then we shift it from C to $C^\sharp, D, ..., B$ one by one (see [9]). By shifting reference chromagram, we can extend the reference chromagram into 12 versions. Then each of the reference chromagram is used for comparing with the original audio chromagram, which can compute 12 matching curves. For every l, we pick up the minima Δ value among 12 versions. A new matching curve will be computed. In Figure 4.2, the second chromagram is one step shifted from the original one. With this method, the intonation problem can be solved. Even when intonation differences between two repetition parts exist, we can still get low Δ values.

One can improve the algorithm even further. Due to the problems of our data collection introduced in Section 1.4. The voices of the singer often fluctuate by several semitones downwards or upwards across the various stanzas of the same recording. To reduce the influence of this condition, we can apply 24 shifts. That is to shift by half a semitone. Then 24 distance values are computed and the one with minimal value will be selected. Also we can apply 36 shifts depends on the data collection.

Even further, to deal with folk songs which are monophonic music, we can only picking spectral components that correspond to the fundamental frequency [3]. Firstly, we estimate the fundamental frequency for each frame. Then we determine the MIDI pitch which

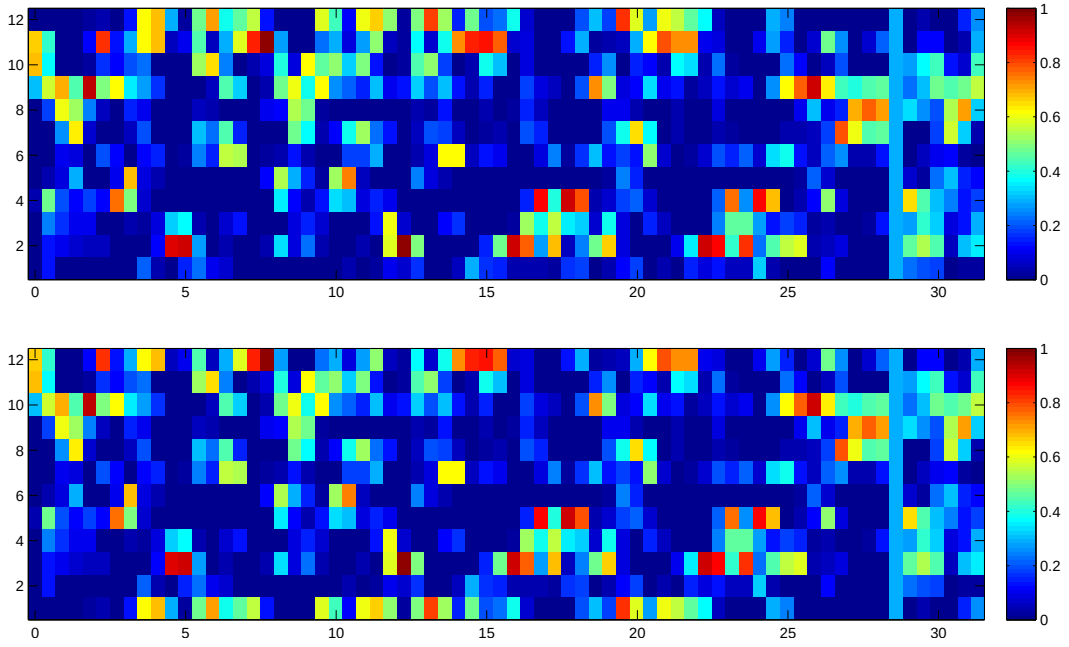


Figure 4.2. CENS chromagram of the first stanza of OGL49313. **Top:** the original CENS chromagram; **Bottom:** one of the shifted version.

center frequency is closest to the fundamental frequency. Secondly, we only assign energy to the pitch subband that corresponds to the determined MIDI pitch, when decomposing the recording into pitch. Finally, we compute the CENS feature representation as before, see Section 2.4. Then we can get a cleaned chromagram named F0-enhanced chromagram, see Figure 4.3

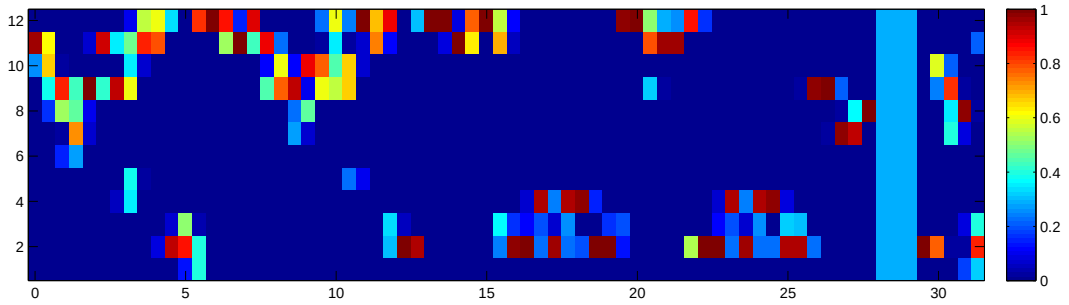


Figure 4.3. F0-enhanced chromagram of the first stanza of OGL49313.

4.1.5 Evaluation Method

Two evaluation functions are used for evaluating segmentation results, one is the alpha-beta-based evaluation and the other is the PR-based evaluation.

Firstly, alpha-beta-based evaluation contains α, β and γ. α is defined to be the average over the cost of the starting point of ground truth segments. A low α value denotes high similarity between the reference and one subsequence of the audio recording. β is defined to be the average over all values of the matching curve. And $\gamma = \alpha/\beta$, which shows how well the desired minima are separated from possible irrelevant minima.

Secondly, PR-based evaluation contains Precision, Recall and F-measure (see [22]). Precision is a measure of the ability of a system to present only relevant items while recall is a measure of the ability of a system to present all relevant items. Here, we only consider the starting points of segments. We define a computed segment is *true positive*, if it coincidences with a ground truth segment up to a small tolerance δ which is measured in seconds. Otherwise, the computed segment is referred to a *false positive*. To be noticed, if there are more than one segment are coincidences with a same ground truth segment in the range of δ, only one of them is considered as a true positive and the others are considered as false positive. Further more, the ground truth segment is a *false negative*, if there is not any computed segment coincidences with it in the range of δ. Precision P is defined as $P = tp/(tp + fp)$, and recall R is defined as $R = tp/(tp + fn)$. With precision and recall, one can obtains the F-measure $F = 2 \cdot P \cdot R/(P + R)$.

4.2 Experiment

In this part, we implement the algorithm, introduced in [15]. The dataset is the same as the one used in the paper. Finally, we derive a segmentation result which is used as a reference for the following experimentations (see Section 5.2)

The experiment result is shown in Table 4.1. The basic experimental parameters are set to be the same as the paper. In this thesis, the CENS(9,1) feature is used. Segmentation precision tolerance $\delta = 2$, which leads to a 4s tolerance range. The quality threshold $\tau = 0.4$. 24 shifts introduced in Section 4.1.4 is applied.

The evaluation result is closed to that in [15]. The average F0-measure in the paper is 0.900 while the average F0-measure for our experiment is 0.904.

Strategy	P	R	F	α	β	γ
Reference-based segmentation CENS(9,1)	0.915	0.899	0.904	0.265	0.389	0.674

Table 4.1. Performance of evaluation result for reference-based segmentation using CENS(9,1) feature.

Chapter 5

Reference-free Segmentation

In the last chapter, reference-based segmentation is introduced. It is efficient but has some limitations. The MIDI reference is used for the segmentation procedure. Without MIDI reference, the segmentation method could not be implied. However, MIDI reference has to be created manually by some experts. Due to this disadvantage, we want to find a segmentation algorithm which can segment music automatically without any additional reference.

Reference-free segmentation is developed. The idea is that, instead of using MIDI reference, we derive the most representative passage of the recording and use it as a reference for segmentation. With this method, we can not only segment the recording but also find the most representative passage of the recording.

In Section 5.1, we describe the mechanism of reference-free segmentation procedure. Then we introduce some experiment results of reference-free segmentation, and show how do we select the mid-level feature representation and the fitness measure. The causes of some problems which occur in the segmentation results are analyzed in Section 5.2. At last, we make a conclusion of the reference-free segmentation.

5.1 Procedure

The procedure of reference-free segmentation is similarly to reference-based segmentation except for the optimal reference selection step. The optimal reference selection step is to find the most representative passage of the recording. The passage plays a role of the MIDI reference. After that step, we apply reference-based segmentation to segment music using this optimal reference instead of the MIDI reference.

5.1.1 Audio Reference

Here, we call the reference in reference-free segmentation as audio reference to distinguish it from the MIDI reference.

Different from the MIDI reference, the audio reference is a part of the audio recording, which can be defined as $R = Y[s : t]$ where $Y = (Y(1), Y(2), ..., Y(L))$ denotes the sequence of the audio recording. we define $R = (R(1), R(2)..., R(K))$ with $R(1) = Y(s)$, $R(2) = Y(s+1)$,...,$R(K) = Y(t)$. The features $R(k), k \in [1 : K]$ and $Y(l), l \in [1 : L]$, are normalized 12-dimensional vectors.

One observable difference between using MIDI reference and audio reference can be found in the matching curve. In Figure 5.1, the red curve is the matching curve of NLB70328 by applying reference-free segmentation with CENS(11,5) feature. In this example, Δ value at 51 seconds equals to zero. That is exactly the starting point of the audio reference (R = Y[51:67]) which is derived by the algorithm of optimal reference selection (see Section 5.1.2). The Δ value of that point equals to zero because an audio reference and its corresponding part in the audio recording are exactly the same.

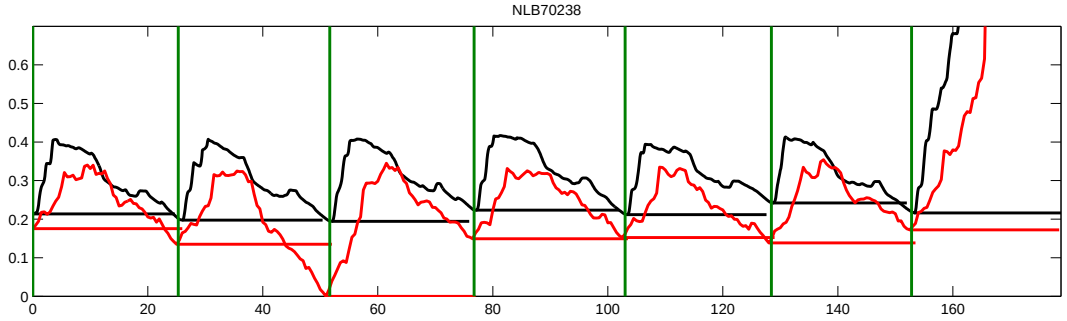


Figure 5.1. Segmentation result of NLB70238 **Black** Reference-based segmentation with CENS(9,1); **Red** Reference-free segmentation with CENS(11,5).

5.1.2 Optimal Reference Selection

Optimal reference selection is the step to select the most suitable reference for segmentation. We developed a fitness measure to evaluate segmentation results which are computed by using different candidate references. We develop a new evaluation function because the old ones, the alpha-beta-based evaluation and PR-based evaluation, are not suitable for reference-free segmentation any more. These two evaluation functions are based on the ground truth information which has to be created manually by some experts as well as the MIDI reference. But we would like to develop a segmentation function which can segment recordings without any manually created resources. The new fitness measure do not need any additional information. Before introducing this new evaluation function, we need to define the basic criteria of a good audio reference.

Optimal Audio Reference Criteria

The optimal audio reference, which is the most representative passage of the recording, should satisfy the following criteria.

1.) Overlaps between segments, which are computed by an audio reference, can not be greater than a certain tolerance τ.

2.) The union of the segments should cover as much of the audio recording as possible.

The more the union of the segments covers, the better the audio reference represents the audio recording.

3.) The audio reference should be as short as possible. We are looking for the shortest audio reference which can represent the whole recording well. A large reference could also represent the recording but is useless for segmentation. For example, an audio recording with an $A_1A_2A_3A_4$ structure. The first half of it can be used as an audio reference. The segmentation result is that the recording is divided into A_1A_2 and A_3A_4. In this case, the audio recording is not completely segmented. The best segmentation result should contain four segments, which are A_1, A_2, A_3, and A_4.

4.) The average cost of the local minimal points in the matching curve should be as small as possible. A low minimal value in the matching curve denotes high similarity between the audio reference and the corresponding part of the music.

Fitness Measure

According to the audio reference criteria, we define a fitness measure ϕ, to score the performance of audio references.

We segment the audio recording Y with an audio reference R. The length of the recording is L. We define hit $H_k, k \in [1:K]$ to be segments in segmentation result. K is the number of hits. H_k^{start} denotes the starting point of H_k while H_k^{end} denotes the ending point of H_k. And $H_k = [H_k^{start} : H_k^{end}]$. The fitness score of reference R is $\phi(R)$ where $\phi(R) \in \mathbb{R}^+$.

Firstly, we detect the overlap condition. A small overlap tolerance τ is set. As Equation 5.1 shows, the overlap between two segments can not be bigger than τ, otherwise the fitness measure procedure of the reference will be stopped and $\phi(R) = 0$. Here, zero is the lowest fitness score which means that the corresponding audio reference is dropped.

$$|H_k^{end} \cap H_{k+1}^{start}| < \tau \qquad k \in (1 : K-1) \tag{5.1}$$

Secondly, after the first step, we calculate coverage rate $\rho(R)$ if $\phi(R) \neq 0$. As the following equation shows, the coverage range of a segmentation result is calculated by summing up the length of every segment, then minus the overlap range. And $\rho(R)$ equals to the coverage range divided by L which is the length of the audio recording.

$$\rho(R) := (|\cup_{k=1:K} H_k| - |\cup_{k=1:K-1} H_k^{end} \cap H_{k+1}^{start}|)/L \tag{5.2}$$

Finally, α_Q, β_Q and γ_Q are used for evaluating the matching curve. It is developed from the alpha-beta-based evaluation function. α_Q is defined to be the average over the cost which is the value at the starting points of every segment. β_Q is the average of all maximum cost between two segmentation points. When calculating β_Q, the tail of the matching curve, where Δ value equals to 1, is excluded. And $\gamma_Q = \alpha_Q/\beta_Q$. Since $0 \leq \alpha_Q \leq \beta_Q \leq 1$, $\gamma_Q \geq 1$. For example, we segment OGL49313 using a reference and the matching curve is shown in Figure 5.2. In this example, α_Q is the average cost of all green arrows pointed points while β_Q is the average cost of all yellow arrows pointed points. The limitation of this evaluation function is the starting points of every segment may be incorrect. And α_Q is based on these points so that a low α_Q do not mean the segmentation result must be correct. It can only be used with other measures.

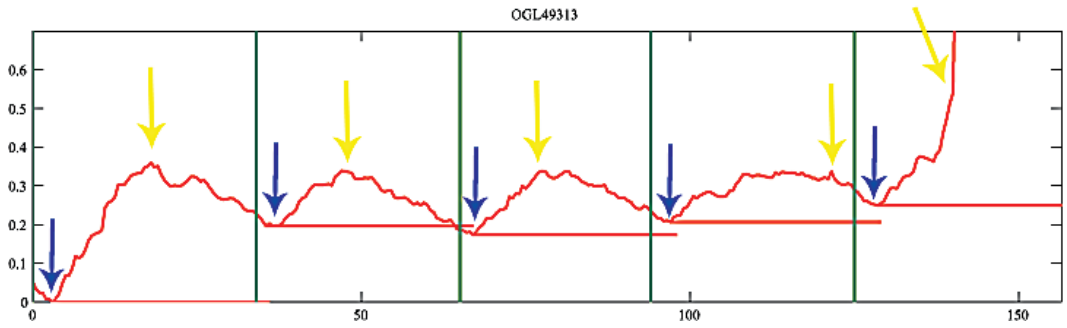


Figure 5.2. An matching curve of OGL49313.

We develop two kinds of fitness measure to compute the fitness score. The first one is GammaQ-based measure. The fitness score of the reference R is $\phi(R) = (\rho(R) * \beta_Q)/\alpha_Q$. The idea of this measure is the cover rate of segmentation results should be as large as possible and β_Q/α_Q should be as large as possible as well. Here, β_Q/α_Q indicates the difference between the local minima and the local maxima of the matching curve. This parameter is meaningful since its value will be small when over-segmented problem occurs. However, this fitness measure do not work well according to the limitation of α_Q, which will be introduced in Section 5.2.2 with some experiments. The second fitness measure is developed to exclude the limitation of α_Q. It is named as BetaQ-based measure and computed by $\phi(R) = \rho(R) * \beta_Q$. We drop α_Q and only use β_Q. This fitness measure works because the combination of $\rho(R)$ and β_Q can still get a meaningful fitness score. First, we assume the qualities of recordings are good. We used one reference for segmenting the recording. If the segmentation result contains over-segmented problem, the coverage rate of it increases while the β_Q value of it decreases sharply. Meanwhile, if the segmentation result contains under-segmented problem, the β_Q value increases while the cover rate decreases sharply. Therefore, if a reference R leads to a under-segmented or a over-segmented result, the fitness score of it will be low.

Candidate Audio References

Brute force algorithm is used for setting up a candidate group of audio references. The idea is to extract the possible references and use them for segmenting the recording. These references are set to be candidate audio references. Considering the runtime cost, we limit the range of candidate audio references, for example the length of the reference. We note the candidate audio references as $R_1, R_2, ..., R_M$. The corresponding fitness scores of them are $\phi(R_1), \phi(R_2), ..., \phi(R_M)$. If $\phi(R_m)$, where $m \in [1 : M]$, is the maxima of this fitness score set, reference R_m is considered as the most representative passage of the audio recording. Then R_m is used as the optimal audio reference for segmenting the audio recording.

5.2 Experiment

Our experiment is based on the experimental dataset, which is used for reference-based segmentation. It contains 47 Dutch folk songs. Considering the condition of it, we limit the

length of candidate references is every 0.5 second from 6s to 40s and the offset of candidate reference is every 1s from beginning to end of the recording. In the experiment, the basic parameters are similar to that in reference-based segmentation. Segmentation precision tolerance is 2s, which leads to a 4s tolerance range. 24-shifts algorithms introduced in Section 4.1.4 is applied. And the global threshold for segmentation is 0.35, and the overlap tolerance is 1s.

Our experiment is mainly focusing on two parts. The first part is to prove the feasibility of reference-free segmentation, the second part is improving the segmentation result.

5.2.1 Feature selection

The feature CENS(9,1) has been used in reference-based segmentation. The feature rate keeps the same as original one since the downsampling rate is 1. But CENS(9,1) feature is not suitable for reference-free segmentation considering the runtime cost. Reference-free segmentation could be considered as running reference-based segmentation thousand times. Therefore, the runtime cost of reference-free segmentation would increase extremely.

Using OGL49313 as an example, we transform it into CENS(9,1) representation and the feature rate equals to 10Hz. The total number of features is 1570. With reference-based segmentation procedure, we only need to locally compare the MIDI reference with the audio recording once. The runtime cost of them is acceptable. But the condition changes for reference-free segmentation. In this example, around five thousand references are picked out. As the mechanism of our optimal reference selection, the segmentation process need to be repeated around five thousand times. Therefore, the runtime cost raises up extremely.

One method of reducing the runtime cost is to increase the downsampling rate. But a higher downsampling rate means a lower resolution, which influences the quality of segmentation result. Because of that we try to find a suitable downsampling rate which could balance the time cost and the quality of the final result. After a series of experiments, CENS(11,5) feature is used.

We compare the segmentation result of reference-based segmentation using CENS(11,5) feature with using CENS(9,1) feature with four different strategies. Additionally, the reference-based segmentation result is used for comparison. Here, we define the ground truth segments as $G_1 = Y[G_1^s, G_1^t], G_2 = Y[G_2^s, G_2^t], ..., G_M = Y[G_M^s, G_M^t]$.

Before we discuss the experiment results, we introduce the table which shows the evaluation results. See Table 5.1, $P, R, F, \alpha, \beta, \gamma, \alpha_Q, \beta_Q, \gamma_Q$ have been introduced in last two chapters. And the last two columns, Hits and ρ, are represented as well. Hits denote the number of segments in the segmentation result. For example the average Hits values of reference-based segmentation (CENS(9,1)) and reference-free segmentation (CENS(11,5)) are 10.38 and 12.26 in this table. This means using the reference-free segmentation (CENS(11,5)) can segment recordings into more segments than using reference-based segmentation (CENS(9,1)). ρ is the coverage rate of segmentation result. We can see the difference of the average coverage rate among different segmentation strategies.

First Stanza Strategy

In this strategy, we use the ground truth information and select the first stanza of each recording as a reference for both CENS(9,1) feature and CENS(11,5) feature, which means $R = G_1$. Table 5.1 shows the average result of applying this strategy. Comparing the result of segmenting recordings using CENS(9,1) feature and CENS(11,5) feature, the differences between the average precision, the average recall and the average F-measure are not more than 0.03. And the result for reference-free segmentation in this strategy is close to reference-based segmentation.

Strategy	P	R	F	α	β	γ	α_Q	β_Q	γ_Q	Hits	ρ
Reference-based CENS(9,1)	0.915	0.899	0.904	0.265	0.389	0.674	0.262	0.471	0.553	9.91	0.899
Reference-free CENS(9,1)	0.853	0.928	0.880	0.169	0.295	0.571	0.172	0.375	0.463	12.26	0.947
Reference-free CENS(11,5)	0.835	0.898	0.857	0.163	0.275	0.594	0.165	0.342	0.488	11.98	0.950

Table 5.1. Average result comparison applying first stanza strategy.

Best Stanza Strategy

In this strategy, we try out all the stanzas noted by ground truth for both CENS(9,1) feature and CENS(11,5) feature, which means $R = G_1, G_2, ..., G_M$. Then we evaluate segmentation results with the fitness measure. The stanza with the highest fitness score is selected as the best stanza for segmentation. According to the evaluation values, the results of segmenting recordings using CENS(9,1) feature and CENS(11,5) feature are pretty good. Both of the average F-measures are above 95 percent, see Table 5.2. The segmentation result is improved a lot comparing to the first stanza strategy. That is because the first stanza is often not the most representative one, singer is still not confident and large deviations exist in the first stanza. Note that, the segmentation results of reference-free segmentation with best stanza strategy are even better than the reference-based segmentation, which also prove the feasibility of reference-free segmentation.

Strategy	P	R	F	α	β	γ	α_Q	β_Q	γ_Q	Hits	ρ
Reference-based CENS(9,1)	0.915	0.899	0.904	0.265	0.389	0.674	0.262	0.471	0.553	9.91	0.899
Reference-free CENS(9,1)	0.951	0.986	0.964	0.139	0.310	0.458	0.169	0.394	0.434	10.62	0.960
Reference-free CENS(11,5)	0.939	0.979	0.954	0.133	0.288	0.475	0.163	0.365	0.451	10.79	0.957

Table 5.2. Average evaluation result comparison applying best stanza strategy.

CENS(9,1) Feature-based Strategy

In this strategy, we use brute force algorithm and CENS(9,1) feature for reference-free segmentation. An optimal reference R is derived when segmenting the audio recording. After that, using this optimal reference R, we apply reference-free segmentation with CENS(11,5) feature. Since the runtime cost is pretty large. We pick up two recordings, OGL49313 and OGL25010. BetaQ-based measure is used as the fitness measure.

Figure 5.3 shows the segmentation result. An offset problem occurs since the selected references shifts from the ground truth segments. Therefore, the PR-based evaluation results are bad, although the structure of segmentation is correct. Applying the reference with CENS(11,5) feature, the result is nearly the same.

Free Strategy

In this strategy, we apply brute force algorithm for both CENS(9,1) and CENS(11,5)

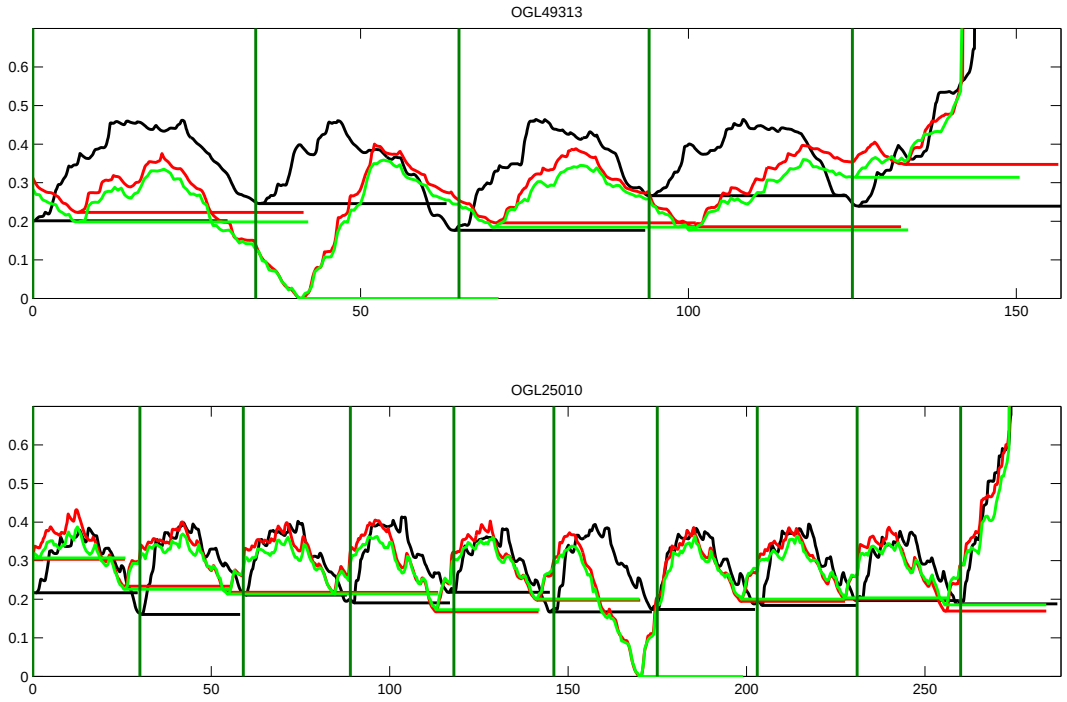


Figure 5.3. Segmentation result of OGL49313 (Top) and OGL25010 (Bottom). **Black** Reference-based segmentation with CENS(9,1); **Red** Reference-free segmentation with CENS(9,1); **Green** Reference-free segmentation with CENS(11,5).

feature. In Figure 5.4 we can find that different audio references are selected by observing the zero value point. For OGL49313, the offset problem still exists and the result is not good either. But the precision raises up to 0.800 when using CENS(11,5) feature. The fitness measure is the same as CENS(9,1) feature-based strategy.

Strategy	P	R	F	α	β	γ	α_Q	β_Q	γ_Q	Hits	ρ
OGL49313 CENS(9,1)	0.000	0.000	0.000	0.237	0.290	0.818	0.190	0.428	0.445	5	0.949
OGL49313 CENS(11,5)	0.400	0.400	0.400	0.169	0.282	0.600	0.165	0.383	0.431	5	0.978
OGL25010 CENS(9,1)	0.100	0.100	0.100	0.248	0.306	0.809	0.190	0.421	0.452	10	0.985
OGL25010 CENS(11,5)	0.800	0.800	0.800	0.197	0.314	0.628	0.196	0.394	0.498	10	0.971

Table 5.3. Segmentation result comparison of free strategy.

According to the comparison results for these strategies, we can conclude that the result of reference-free segmentation using CENS(11,5) feature is nearly the same as using CENS(9,1) feature. Meanwhile, the time cost when using CENS(11,5) feature is much less than using CENS(9,1) feature. Using strategy 4 as an example, the total runtime is about 14.7 hours when we segment those two recordings with CENS(9,1) feature. This is far too much. But running the reference-free segmentation procedure in the same environment, the total runtime for using CENS(11,5) feature is only about 0.6 hour, which is about 25 times faster than using CENS(9,1) feature. In fact the final runtime for segmenting the whole dataset, which contains 47 folk songs, is about 12.7 hours.

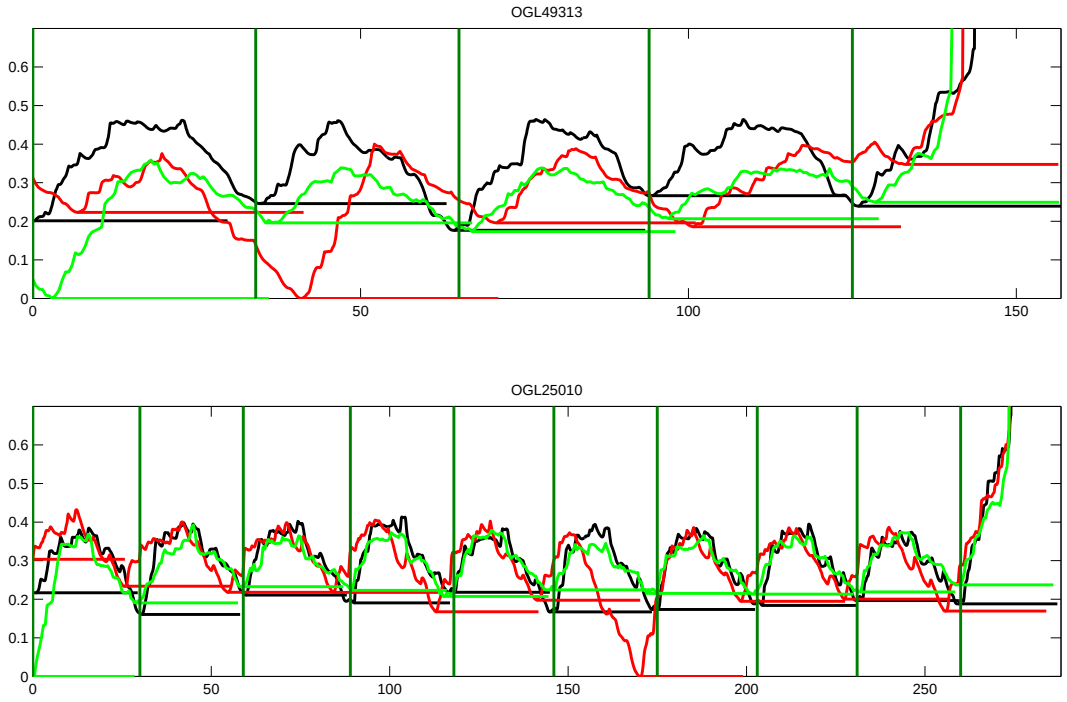


Figure 5.4. Segmentation result of OGL49313(Top) and OGL25010(Bottom). **Black** Reference-based segmentation with CENS(9,1); **Red** Reference-free segmentation with CENS(9,1); **Green** Reference-free segmentation with CENS(11,5).

We also do experiment with CENS(41,11), but the accuracy of the result decreases a lot. The average F0-measure for the 47 folk songs is 0.548. Considering the quality of results and the time cost, finally we use CENS(11,5) feature for reference-free segmentation.

5.2.2 Fitness Measure Selection

We define two fitness measures in Section 5.1.2, the BetaQ-based measure and the GammaQ-based measure. Comparing to the BetaQ-based measure, the GammaQ-based measure includes one more parameter which is α_Q. But the performance of it is not as good as the BetaQ-based measure. Table 5.4 shows the comparison of average results between these measures.

Many of segmentation results, using GammaQ-based measure, have under-segmented problem. The problem is caused by the limitation of α_Q. Different from using MIDI reference, the matching curve, which is computed by using audio reference, contains a zero point. It is the starting point of the audio reference which is also the global minima of the matching curve. Therefore, this point must be considered as a segment point and included in α_Q. And this zero value decreases α_Q especially when the number of segmentation is small. Since we define α_Q value is the lower the better, a long references, which lead to less segments, can get a better fitness score. Therefore, many recordings are seg-

mented with under-segmented problem when using reference-free segmentation with the GammaQ-based measure.

In order to solve this problem, we exclude the zero value when calculating α_Q. A modified GammaQ-based measure is calculated by using this modified α_Q. The result is improved extremely. The under-segmented problem, which is caused by the zero value, is solved. But the limitation of α_Q is still exist.

Considering the overall performances, we drop α_Q, and choose the BetaQ-based measure as the fitness measure.

Strategy	P	R	F	α	β	γ	α_Q	β_Q	γ_Q	Hits	ρ
BetaQ-based measure	0.713	0.758	0.709	0.176	0.271	0.656	0.157	0.359	0.448	13.17	0.970
GammaQ-based measure	0.515	0.489	0.470	0.191	0.237	0.840	0.124	0.355	0.364	11.30	0.903
modified GammaQ-based measure	0.695	0.729	0.693	0.160	0.253	0.648	0.161	0.347	0.469	12.87	0.953

Table 5.4. Average evaluation result of reference-free segmentation with BetaQ-based measure feature, reference-free segmentation with GammaQ-based measure, and reference-free segmentation with modified GammaQ-based measure (The α_Q for this fitness measure is the modified one).

5.2.3 Result and Problem Conclusion

The result of reference-free segmentation is not as good as reference-based segmentation. Here, the BetaQ-based measure is used as the fitness measure. Table 5.5 shows the result. The F-measure result of reference-free segmentation is lower than that of reference-based segmentation. The decreasing F-measure is caused by optimal reference selection. To summarize the results, there are mainly three kinds of problems: over-segmented Problem, under-segmented problem and offset problem. In this section, the old version α_Q is used, which includes the zero point.

Strategy	P	R	F	α	β	γ	α_Q	β_Q	γ_Q	Hits	ρ
Reference-based CENS(9,1)	0.915	0.899	0.904	0.265	0.389	0.674	0.262	0.471	0.553	9.91	0.899
Reference-free CENS(11,5)	0.713	0.758	0.709	0.176	0.271	0.656	0.157	0.359	0.448	13.17	0.970

Table 5.5. Comparison of reference-based segmentation using CENS(9,1) feature and reference-free segmentation using CENS(11,5) feature with BetaQ measure.

5.2.3.1 Problems

Table 5.6 represents recordings which have been segmented wrongly. The segmentation result of 17 recordings are not good. We listen to the music to find the reason why these problems occurs.

Under-segmented Problem

NLB74028 is a song that is hard to sing. The tempo of it is very fast. Because of that, the old lady sing the song full of intonation mistakes. And she starts talking at the end of recording. But these problems do not influence the segmentation result. The cause of the under-segmented problem is the length of the audio reference. Comparing the maximum

Offset Problem	Over-segmented Problem	Under-segmented Problem
OGL49313	OGL27517	NLB74028
OGL27516	NLB72395	NLB75059
OGL25011	NLB75068	
NLB70134	NLB76271	
NLB72355		
NLB73993		
NLB74437		
NLB75167		
NLB75325		
NLB76426		

Table 5.6. Conclusion of recordings with bad segmentation result.

length of reference(40s), the recording length, which is only 50s. The reference, which length equals 39s, are used. This reference lead to only one hit, but it could get a very high β_Q. One possible improvement to deal with the problem is to limit the maximum length of candidate reference no more than half of the length of the recording.

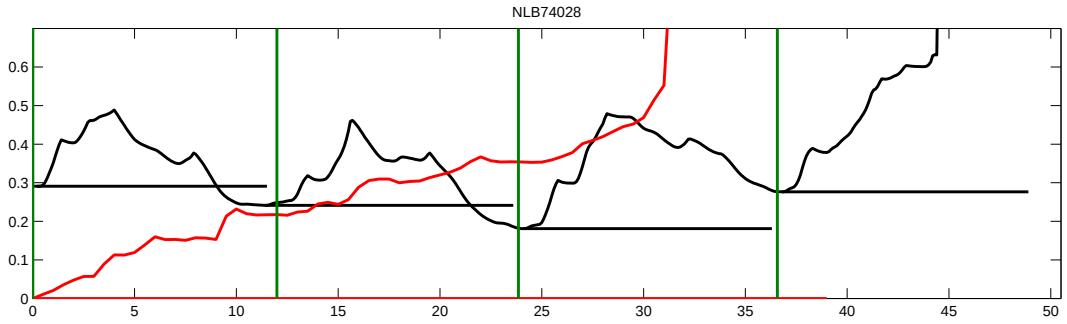


Figure 5.5. Segmentation result of NLB74028. **Red:** Reference-free segmentation with BetaQ measure; **Green:** Reference-based segmentation

After limiting the maximum reference length to be the half length of the recording, the segmentation result, which is shown in Figure 5.6, contains 2 hits. The F-measure raises up from 0.4 to 0.7, although the under-segmented problem is still exist. Considering the length of the recordings in our dataset, the limitation of the maximum reference length only affects the recording. The improved segmentation results can be find in Appendix.

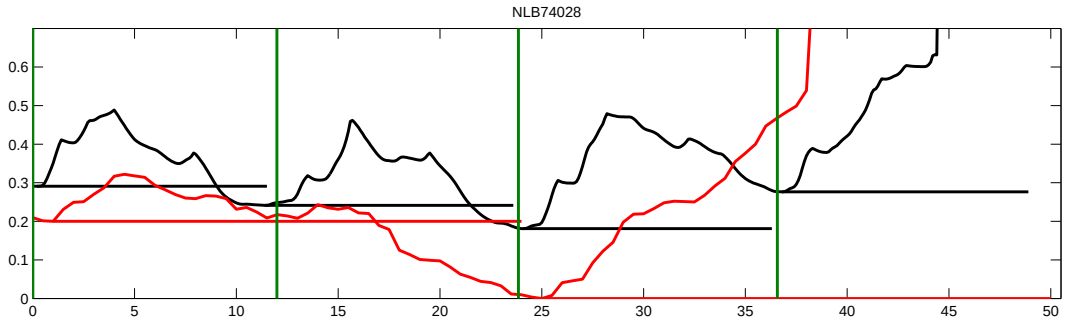


Figure 5.6. Segmentation result of NLB74028 with the maximum candidate reference length limitation

NLB75059 is another recording with under-segmented problem. In the recording, singer forgets to sing the last phrase of the last stanza, which makes the length of the last stanza shorter than the others. As Figure 5.7 shows, even for the best stanza strategy, the segmentation result is not correct. There is a large overlap between the last two segments. As our rules, the fitness score of the segmentation result is zero, when large overlap exists. In this example, using one of the first six stanzas as the audio reference would lead to a segmentation result with overlap. Therefore, we can not get a good segmentation result for this recording. And using a long reference could get better β_Q so that under-segmented problem occurs.

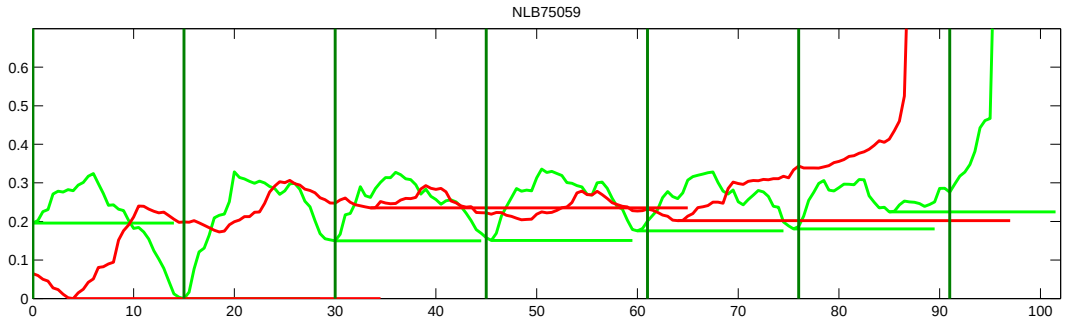


Figure 5.7. Segmentation result of NLB75059 with different reference-free segmentation strategies. **Red:** Reference-free segmentation with BetaQ measure; **Green:** Reference-free segmentation using best stanza strategy.

Over-segmented Problem

OGL27517 is a typical example that small repetitions are contained in stanzas. The score of one standard stanza of OGL27517 is indicated in Figure 5.8. We can find that the last two lines in this stanza are nearly the same. And the differences between the first line and these two lines are not so big either. When this kind of repetitions exist in the standard stanza, these repetitions could also be considered as a stanza. It is a hard task to decide which repetition is correct automatically. And as our criteria of optimal reference selection, the reference should be as small as possible. In our experiment, the third line of the first stanza is selected as the reference. The results is shown in Figure 5.9. The red matching curve also shows the relations among three phase in one stanza.

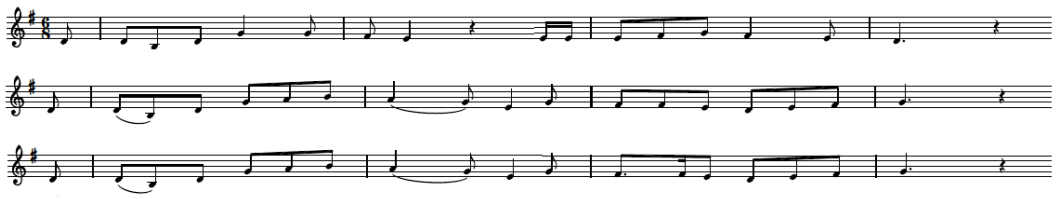


Figure 5.8. Score representation of one stanza of folk song OGL27517.

The problem for NLB72395 is that the recording is messed up. There are considerable differences between stanzas. For example, the old lady often misses several sentences and jumps to the following part. And there are some hesitations in her voice during the

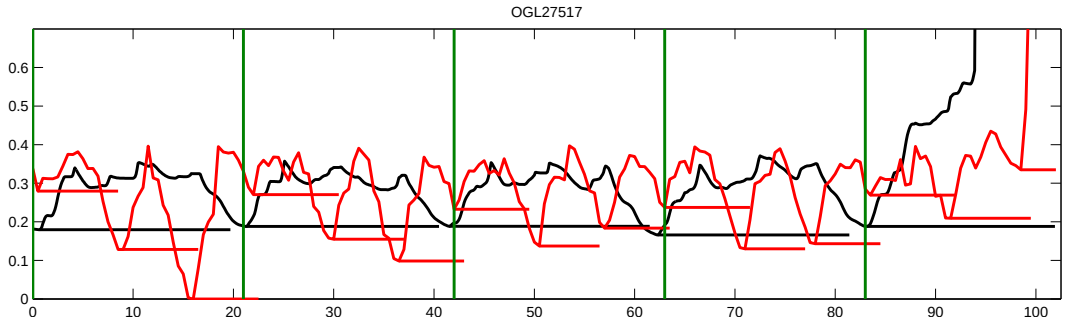


Figure 5.9. Segmentation result of OGL27517. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation.

recording. Considering the condition of this recording, it is very hard to segment it. Even we use the best stanza strategy, the result is bad. In Figure 5.10, black line indicates result of reference-based segmentation and the curve is very noisy. Similar to the best stanza strategy, the reference-based segmentation do not get a acceptable result either.

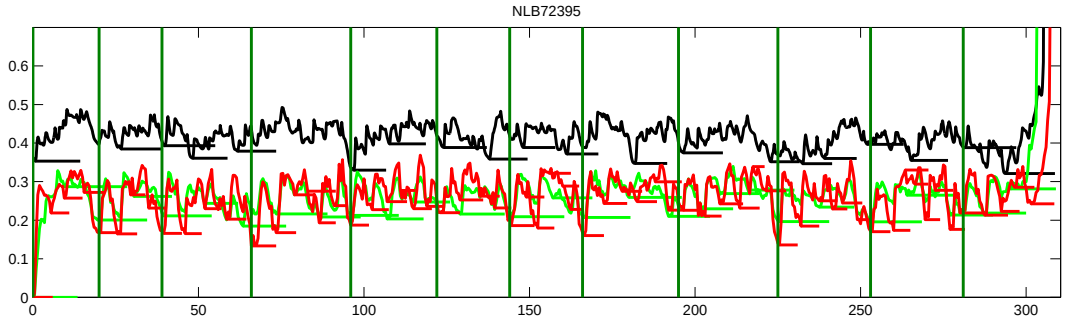


Figure 5.10. Segmentation result of NLB72395. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure; **Green:** Reference-free segmentation using best stanza strategy.

The segmentation result of NLB75068 reflects the shortage of the reference-free segmentation. In Figure 5.11, we can find the last part of the recording is no long the repetitions of any stanza in the front part. In fact the singing part is only from 0s to 155s, and the singer begins to talk in the rest part. It is a very big challenge to reference-free segmentation. Unfortunately, it fails to extract the optimal reference. The red matching curve shows that the distance between two dissimilar parts of recording could be small when they are short enough. And the segmentation coverage rate could be pretty good when a short reference is used. The coverage rate in this example is 94.8%.

NLB76271 is a very complex example. Firstly, the structure of the stanza is complex. There are some small repetitions in one stanza. Secondly, the recording quality is the worst one in our dataset. A bird keeps on singing loudly during entire recording. And the singer talks from 57s to 68s and she missed many parts of the song. Figure 5.12 shows the result. The length of the selected reference is 6s, which is similar to the last example.

In the last two examples, we find that a fixed threshold for segmenting is not flexible

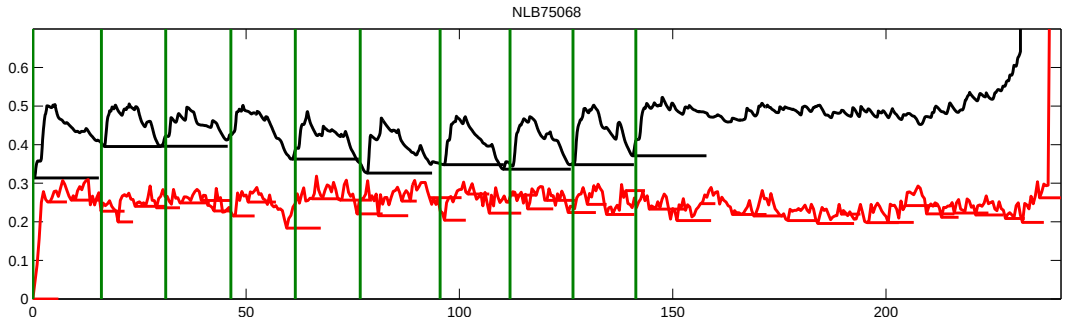


Figure 5.11. Segmentation result of NLB75068. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure.

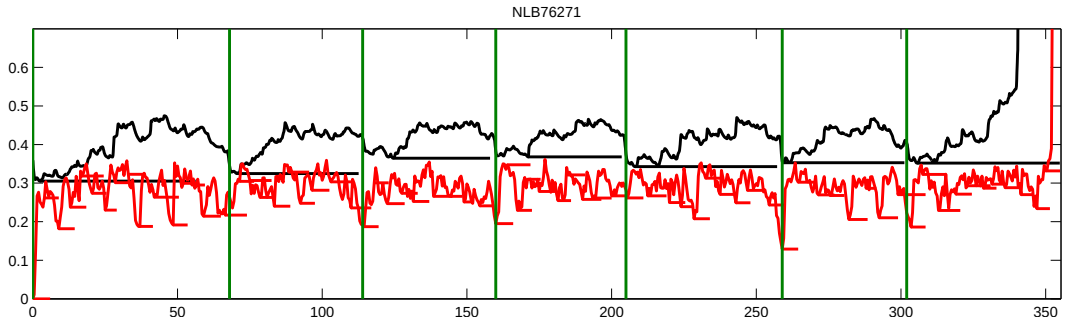


Figure 5.12. Segmentation result of NLB76271. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure.

enough. Although we can set it by estimating the average of the most common condition, sometimes exception happens like this example. There are some hits even on the top of matching curve, which messes up the result completely. So we develop a flexible threshold which is the average cost of segmentation result. And the value equals to 0 and 1 is excluded. Using this flexible threshold could not improve our result significantly, but it could avoid selecting some meaningless references, especially for the very short ones. Because the coverage rate of the result which is calculated with a short reference is very low in these examples. But the experiment result is still not acceptable. As Figure 5.13 shows, although the matching curves looks better, the segmentation result is still incorrect. The talking part in NLB75068 contains several segments wrongly, and both of NLB75068 and NLB76271 are segmented with offset problem.

Offset Problem

Offset problem is the main problem in segmentation results. This problem occurs in 10 recordings. According to the causes of offset problem, these recordings are classified into three groups. The first group contains OGL25011, NLB72355 and NLB75325, the second group contains NLB74437, NLB75158, NLB75167, and the third group contains the rest.

Firstly, recordings in the first group have a common character. All of them contain one special stanza. This special stanza is not the same as the standard stanza. In this case, segmentation result using the best stanza as a reference contains a large overlap

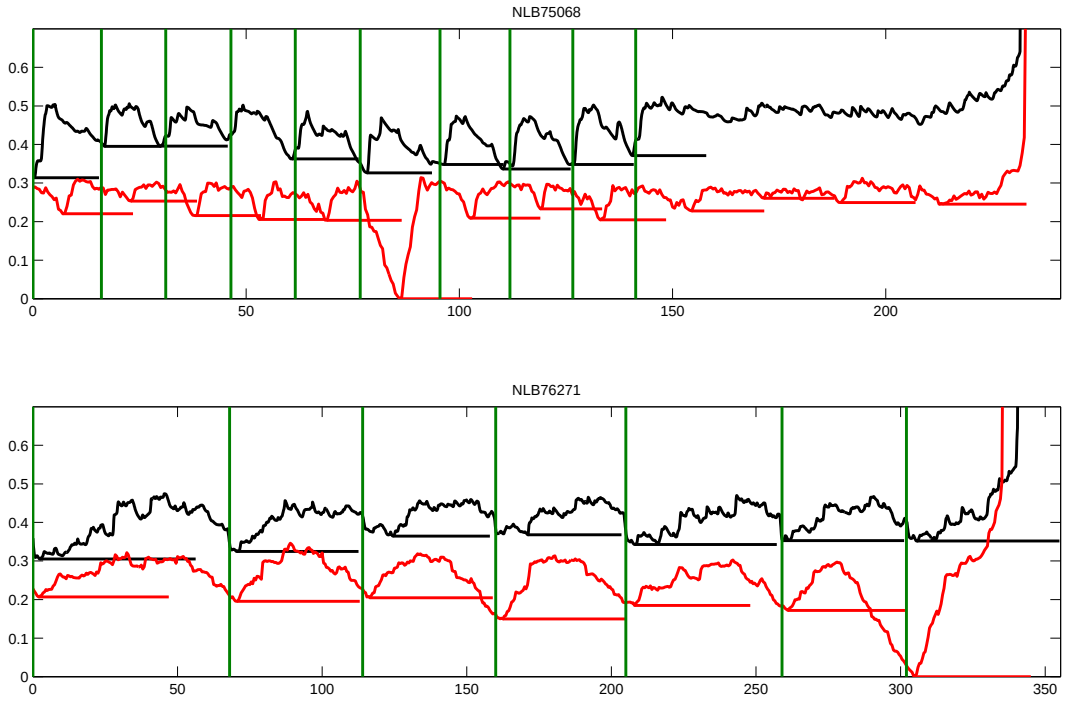


Figure 5.13. Segmentation result of NLB75068(Top) and NLB76271(Bottom) using flexible threshold. **Black** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure.

area. Considering the overlap rule, the fitness score of the reference is zero. Therefore, a reference with offset problem is derived as the optimal reference.

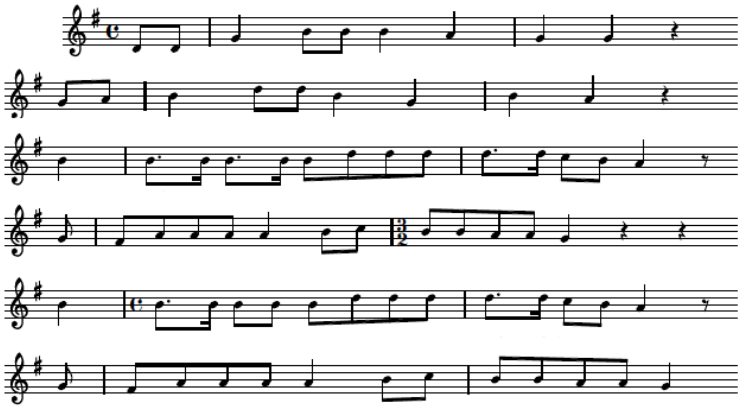


Figure 5.14. Score representation for the standard stanza of OGL25011.

In the last stanza of OGL25011, one third part of standard stanza are missing. As Figure 5.14 shows, the structure of the stanza could be considered as $A_1A_2B_1B_2$. In last stanza, singer only sings B_1B_2 part. The segmentation result could be found in Figure 5.15. The results of both Reference-based segmentation and reference-free segmen-

tation with best stanza strategy contain overlaps. Therefore, the best stanza is dropped when segmenting this recordings by reference-free segmentation with BetaQ measure. The reference with a wrong offset is selected. The precision of segmentation drops to 0 while the coverage rate is 93.8 percent.

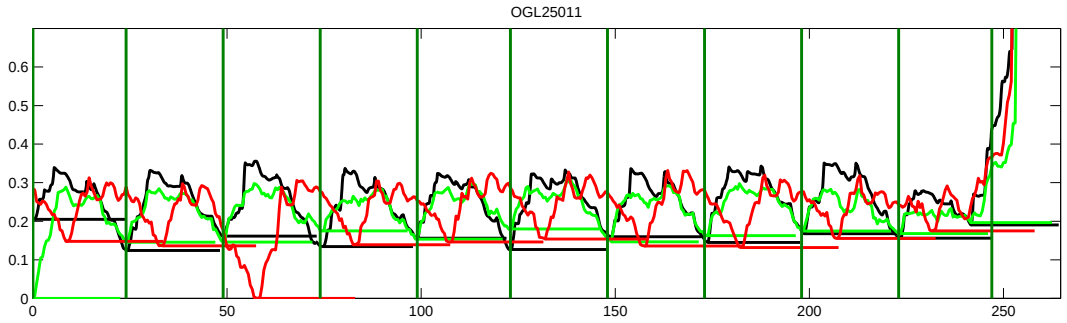


Figure 5.15. Segmentation result of OGL25011. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure; **Green:** Reference-free segmentation using best stanza strategy.

The first stanzas of NLB72355 and NLB75325 are special. Figure 5.16 represents the score of the first stanza and rest stanza of NLB72355. The length of first stanza is three forth of the standard stanza. The condition for NLB75325 is nearly the same, but even more complex. Figure 5.17 represents the segmentation result. It is the same as the condition of OGL2011.

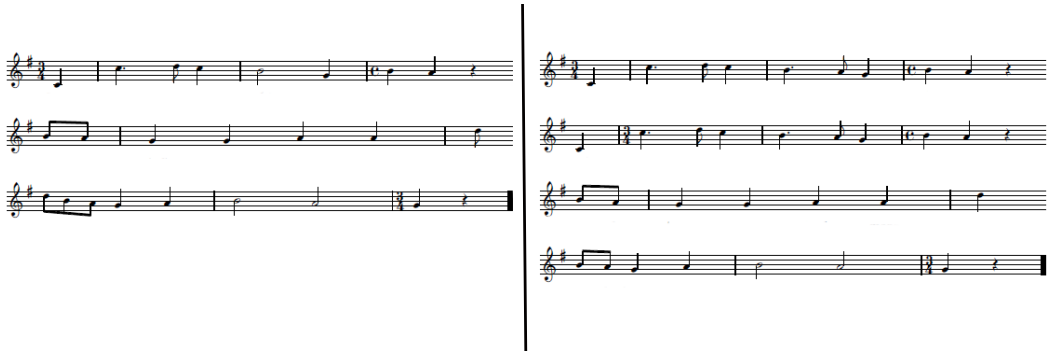


Figure 5.16. Score representation of two different kinds of stanzas of NLB72355. **Left:** First stanza; **Right:** Standard stanza.

Secondly, the qualities of recordings in the second group are bad. NLB74437 is sang by three old ladies. In fact, segmenting a recoding sang by one or more persons is not different. But the problem in this recording is that one lady always sings the song with both an octave below the melody and a different tempo from others. This special case influences the segmentation result. In another example, the singer of NLB75158 forgets what to sing and makes mistakes frequently, which makes her singing full of hesitations. NLB75167 is the audio recording with the worst quality, a female voice is in background towards end. Singer omits several parts during her singing and she sang the last stanza with a considerable different version of melody. The matching curves is shown in Figure 5.18. Even for reference-based segmentation, the result is not good. The average value of the

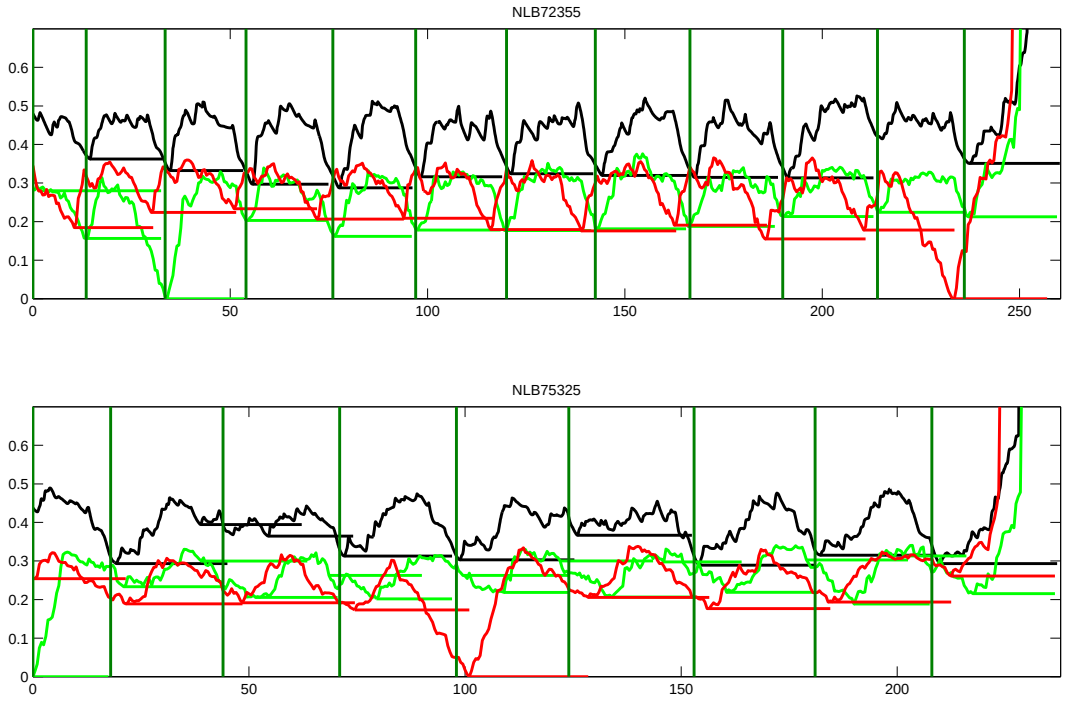


Figure 5.17. Segmentation result of NLB72355(Top) and NLB75325(Bottom) using flexible threshold. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure; **Green:** Reference-free segmentation using best stanza strategy.

black curve is high, which shows low similarity between the audio recording and the MIDI reference. And the shape of the curve is very special, which do not contain the distinct local minima like other examples.

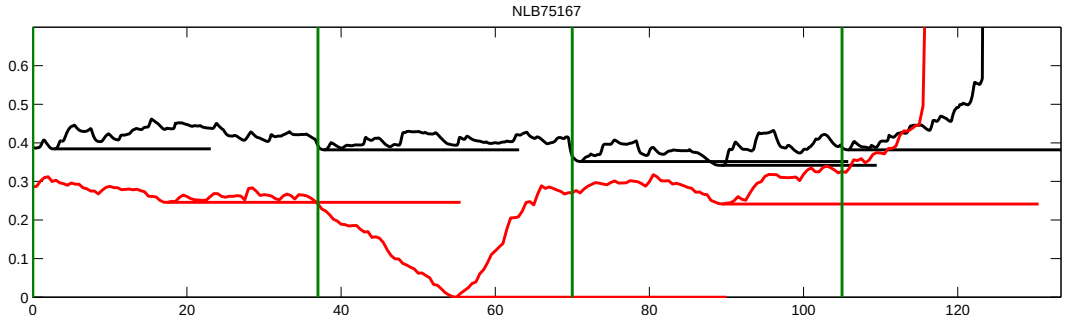


Figure 5.18. Segmentation result of NLB75167. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure.

Thirdly, recordings in the last group do not contain any quality problem, except for OGL49313. The singer of OGL49313 forgets what to sing at the beginning of the second stanza. She talks for several seconds and than continues to sing. According to the similar cause of offset problem like others, this recording belongs to this group.

One possible cause of the offset problem in these recordings is the short breaks between two stanzas. Singer often breathes between two stanzas. These breaks are not included in the segmentation parts, when we segment recordings using the MIDI reference which do not contain these breaks. But using reference-free segmentation, these breaks could be included into the selected audio reference. One possible way to include these breaks is shifting the reference offset a little bit. This kind of reference is starting from the last several notes of the previous stanza and the break between two stanza is included. Due to this different aspect between MIDI reference and audio reference, the average coverage rate of reference-based segmentations is smaller than that of reference-free segmentations.

Figure 5.19 indicates the result of reference-free segmentation for NLB76426. In this figure, we can find that offset problem occurs when we segment NLB76426 with reference-free segmentation. And the table shows that the segmentation is failed. The precision of reference-free segmentation in this example is closing to zero. What interests us is that the coverage rate measure, which is shown in the last row, is pretty good. It is about 0.100 higher that the result of reference-based segmentation.

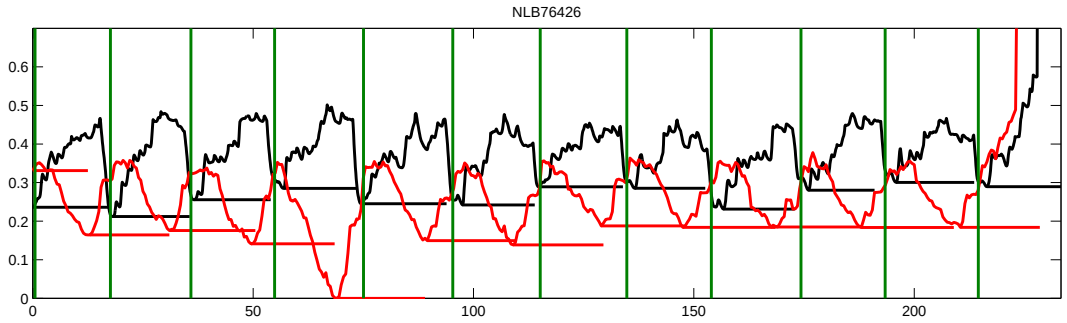


Figure 5.19. Segmentation result of NLB76426. **Black:** Reference-based segmentation; **Red:** Reference-free segmentation with BetaQ measure.

5.2.3.2 Conclusion

The problem for reference-free segmentation can be concluded as two types, the problem of the recording and the problem of the algorithm.

The problem of the recording is the bad quality of recordings. Unlike reference-based segmentation using a standard MIDI reference, reference-free segmentation needs to extract references from recordings. The optimal reference selection is sensitive to the quality of the audio recording. That is because the mistakes and noises in the recording are considered as a part of the song as well. Therefore, it is more sensitive to quality of recordings than reference-based segmentation. A bad quality of recordings always leads to a bad result of optimal reference selection.

The problem of the algorithm is how we measure a good reference. Using OGL25011 as an example, the reference has to be shifted in order to avoid overlap occurs, so that the best reference is dropped. The coverage rate parameter is very important. However, it could cause the offset problem as well.

Chapter 6

SyncPlayer Framework

The SyncPlayer system is an advanced audio player for multimodal presentation of high quality audio and associated music-related data [11]. The basic framework of the SyncPlayer is introduced in this chapter.

The basic framework of the SyncPlayer is introduced in Section 6.1. We show two plug-ins of the SyncPlayer, *the Audio Switcher* and *the Audio Structure* in Section 6.2 and Section 6.3.

6.1 SyncPlayer Overview

The SyncPlayer is a client-server based software [7]. It serves as a tool for MIR. The framework of the SyncPlayer is shown in Figure 6.1.

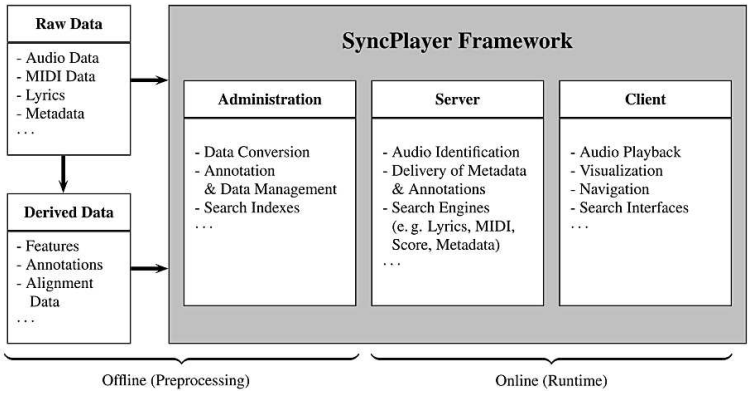


Figure 6.1. Overview of the SyncPlayer Framework [7].

The basic function of the SyncPlayer is a standard audio player which supports MP3 and WAV files. It is extended by some plug-ins such as the Audio Switcher, the MultiStructure Visualizer and the SymbolicQuery Plugin. The input data is treated as two types. One is

raw data and the other is derived data. Raw data refers to data which represents music, like audio, MIDI or MusicXML files. It is also used to generate derived data. The derived data could be features, annotations and so on. Using annotations as an example, one can annotate a piece of music with different kinds of annotations such as structure and chord information. With certain plug-ins, we represent the derived data in different ways. Additionally, a database is built to index and store the raw data and derived data. One can access the data on the SyncPlayer server.

6.2 Audio Switcher

The Audio Switcher is one of the available plug-ins for the SyncPlayer. It allows user to switch among several interpretations for the same piece. Figure 6.2 shows a instance of the Audio Switcher.

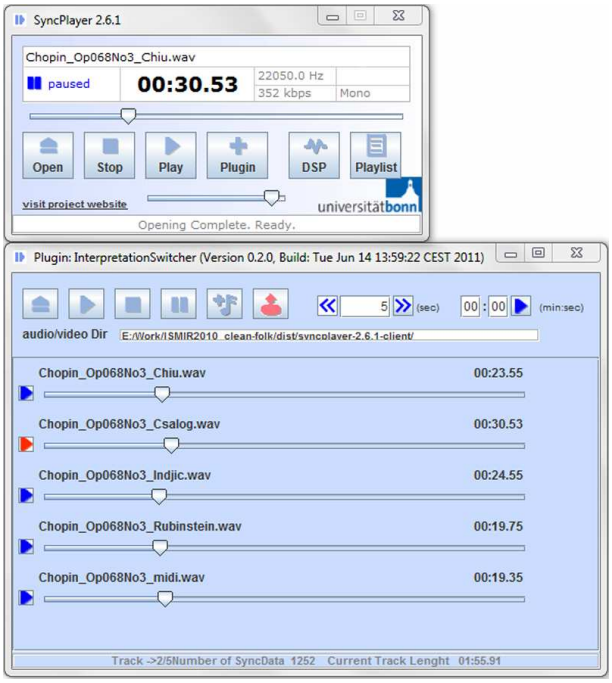


Figure 6.2. The SyncPlayer with Audio Switcher plug-in.

The understanding of music and style of interpretation varies among different performers. Therefore, for a single piece of music, there exist many different interpretations from various performers and instruments. By using Audio Switcher, users can listen to, compare and switch between these interpretations smoothly.

Figure 6.2 shows five synchronized interpretations of Chopin Mazurka Op.068 No.3 playing in the Audio Switcher. The duration of these interpretations are certainly not the same. However, in our design, the sliders which contain related time information are all stretched to the same length for visualization purpose. Therefore, the slider knob which indicates the corresponding time position of each interpretation is running differently.

The red arrow on the left side indicates the currently playing recording. It is considered as a reference interpretation. Whenever the user wants, the playing recording can be changed by clicking the arrows on the left side. The jump function of the slider stays the same as a normal slider. One can choose any position within any of the recordings by clicking a position of the respective slider. With this operation, the clicked recording is set as a reference and the playing time is converted the synchronized time.

6.3 Audio Structure

The Audio Structure is also one of the plug-ins for the SyncPlayer. The repetitive segments which reveals important structure information is displayed in this plug-in. It is an visualization of musical structure which helps users to understand and manage the audio information.

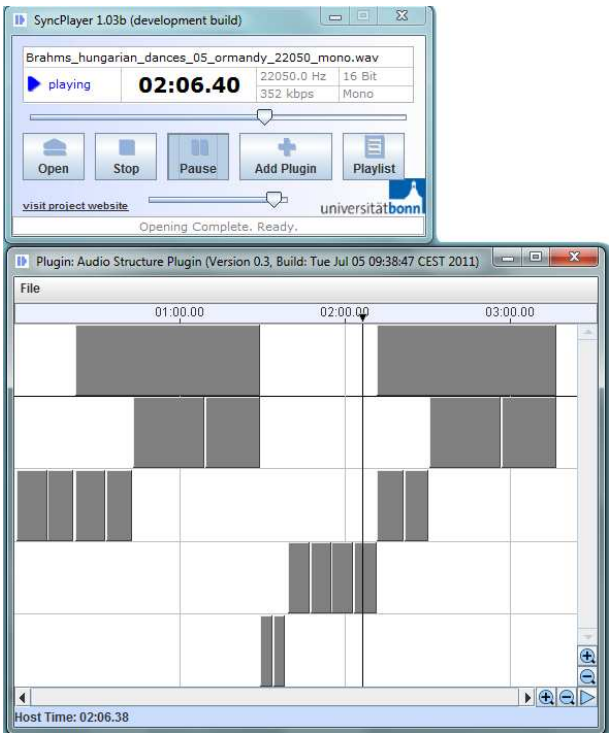


Figure 6.3. The SyncPlayer with Audio Structure plug-in.

Figure 6.3 shows Hungarian Dance No. 5 composed by Brahms. In the plug-in component, 18 gray blocks are classified into 5 groups and the blocks inside a group are repetitions of each other. In the visualization area, blocks of the same group are placed in the same line. In the first line, there are two large blocks which represents the repeating two parts of the recording. Additionally, blocks in the second line are actually subpart of those blocks in the first line. This shows that in some large repetitions, small repetitions may exist. Furthermore, the length of blocks in the same line is not always the same. That is because of the tempo difference between two repetitive part.

Chapter 7

SyncPlayer Extension

The Interpretation Switcher is considered as a plug-in of the SyncPlayer. It is based on the Audio Switcher and the Audio Structure. Some of extensions of the Interpretations Switcher, which is the extensions of the of the SyncPlayer as well, are developed.

Firstly, We introduce the basic functionalities of the Interpretation Switcher in Section 7.1. Secondly, three extensions of the Interpretation, is represented in the following sections.

7.1 Interpretation Switcher

Two plug-ins for the SyncPlayer, namely the Audio Switcher and the Audio Structure, were introduced in Chapter 6. They are helpful for music analysis. But the user can not use them at the same time. The Interpretation Switcher is developed based on the ideas of these two plug-ins. The user interface of the Interpretation Switcher, which has been changed completely from the previous plug-in style, is indicated by Figure 7.1.

As the figure shows, except from the changes of the user interface, an annotation field is added for each interpretation. Every annotation field contains several annotations. These annotations may carry different musical information, such as structure segmentation of a recording or chord information of a recording. Besides, they can be easily changed by the user.

The annotation field is a good place for visualizing the MIR results. It is also useful to show the relations among several interpretations. We use Chopin Mazurka Op.068 No.3 as example. In each annotation field, there are several annotations with previously specified colors. In this example, one color denotes one chord type. With the help of these annotations, one can get a rough idea about the selected recordings. As one can clearly sees from the annotations in the figure, the chord distributions of different interpretations share some common properties like the progression or duration of the chord, but it is also obvious that at some points strong differences exits.

Some basic functions are modified in this version. The jump function is extended by clicking annotation labels. The player plays the music from the beginning of the annotation

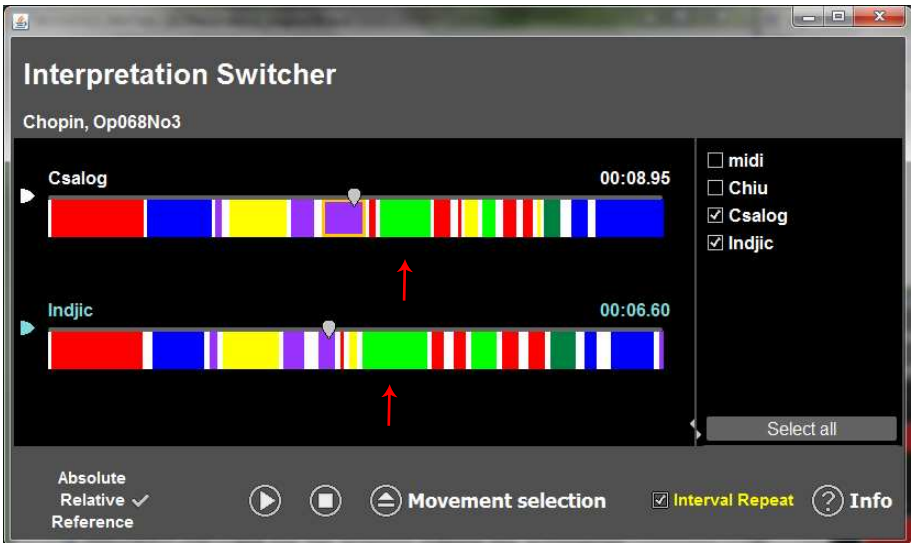


Figure 7.1. New user interface of Interpretation Switcher extended with annotation field.

which the user left clicked. At the same time, the current playing recording will switch to the corresponding recording at the same time. An Interval Repeat button is added at the bottom of this application. One can turn on the repeat model by activating the Interval Repeat button. The border of the selected annotation will be painted by yellow and the annotation range of the recording will by played repeatedly.

Different from traditional audio players, the Interpretation Switcher does not load a audio file directly. Instead, one should open a *Sync File*. The Sync File, which is a text file, contains the information of which audio files need to be loaded, the length of the audio files, and the synchronization information. Figure 7.2 shows the Sync File which has been used for the example shown in Figure 7.1. The Sync File can be separated into two parts. The first part contains some basic information and the second part, *called Sync Table*, is the synchronization data. Line 3, 4, 5 are the number of interpretations. Line 6 is the number of rows of the Sync Table. The information of first interpretation is stored from line 9 to line 12. Filename and the duration are two important parameters. The filename determines which audio file and annotation file will be opened. The duration determines the length of the corresponding interpretation. And the information of the other three interpretations are set in the following part. In this example, the Sync Table starts from line 35. Each column corresponds to one interpretation. The synchronization information is contained in the table. For example, see line 46, the content of the second interpretation at the time 0.1s corresponds to that of the first interpretation at the time 1.0s.

After opening the Sync File, the related audio file and *Label File* are loaded at the same time. In the Label File, two kinds of information are included. The first part contains the annotation information and the second contains the color classes. See Figure 7.3, The number of annotations is determined in line 1, which is 21 in this example, such that the following 21 lines are the boundaries and color types for 21 annotations. See line 2, the first two numbers determine the boundary of an annotation, and the last number is the

```
1  InterpretationSwitcher Text File
2  FileVersion: 2
3  numInterpretations: 4
4  numAudioInterpretations: 4
5  numVideoInterpretations: 0
6  numRows: 1252
7
8  //Interpretation Infos:
9  interpretation_type: audio
10 dir_rel_with_ending_backslash:
11 filename: Chopin_Op068No3_Chiu.wav
12 duration_sec: 96.46712
13
14 interpretation_type: audio
15 dir_rel_with_ending_backslash:
16 filename: Chopin_Op068No3_Csalog.wav
17 duration_sec: 115.91215
18
   • • •
34 //Synchronization Data:
35 //format: one row per feature frame
36 00000.00000 00000.00000 00000.00000 00000.00000
37 00000.10000 00000.00000 00000.09995 00000.10000
38 00000.20000 00000.00000 00000.09995 00000.20000
39 00000.30000 00000.00000 00000.09995 00000.30000
40 00000.40000 00000.00000 00000.09995 00000.40000
41 00000.50000 00000.00000 00000.09995 00000.50000
42 00000.60000 00000.00000 00000.09995 00000.60000
43 00000.70000 00000.00000 00000.09995 00000.70000
44 00000.80000 00000.00000 00000.09995 00000.80000
45 00000.90000 00000.10000 00000.09995 00000.90000
46 00001.00000 00000.20000 00000.10000 00001.00000
47 00001.10000 00000.30000 00000.19995 00001.09995
   • • •
```

Figure 7.2. An example of the Sync File of Chopin Mazurka Op.068 No.3.

color ID which indicates the color of the annotation which will be defined in the following part. Note that, if the color number is not defined in the color class, the corresponding color would be set to a default color. From line 24, the color classes are set. For example in line 25, 1 indicate the ID of one color and [0,0,255] is the corresponding RGB value.

The main advantages of Interpretation Switcher are as follows. Firstly, the functionalities of the Audio Switcher and the Audio Structure are integrated. Secondly, the meaning of annotations are extended. It can not only representing the repetitive structure but also some other musical characters. Additionally, comparisons among interpretations can also be done by observing and comparing annotations. Meanwhile, with the improvement of the jumping function and the extension of the repeating function, user can locate and analyze some key points of recordings easily.

7.2 Extension of Timeline Modes

We have some further developments of the Interpretation Switcher. Two new modes for the timeline slider is implemented. These timeline modes are named as *relative mode*, *absolute mode*, and *reference mode*.

```
1  NumIntervals: 21
2  0.100000 2.800000 6
3  2.900000 4.800000 1
4  4.900000 5.100000 22
5  5.200000 5.200000 6
6  5.300000 7.000000 15
7  7.100000 7.800000 22
8  8.000000 9.300000 22
9  9.400000 9.600000 6
10 9.700000 11.200000 11
   • • •
24 NumColorClass: 6
25 1: [0,0,255]
26 6: [255,0,0]
27 11: [0,255,0]
28 15: [255,255,0]
29 22: [154,50,255]
30 8: [0,128,64]
```

Figure 7.3. An example of the Label File of Chopin Mazurka Op.068 No.3 performed by Csalog

Relative mode

The relative mode is what we have used until now. All the timelines are linearly stretched to yield the same length. Figure 7.1 shows an example of the relative mode. Annotation fields are linearly stretched too. It can be considered as a normalization process. The length of each annotation is no longer absolute, it is stretched with the annotation field. Therefore, we can not compare the length of annotations which are not in the same interpretation. For instance, the first green annotations, which are noted by red arrows, in the interpretation played by Csalog is smaller than the one played by Indjic. But their actual durations are nearly the same, because the duration of the whole interpretation performed by Csalog is longer than the one performed by Injic. The limitation for the relative mode is that one can not infer the actual duration of each recording and annotations by the length of them directly.

Absolute mode

The absolute mode leaves the timeline length corresponding to the actual duration for each interpretation. All the sliders keep a uniform scale which is depends on the size of the application window. In this mode, one can find out the duration differences between recordings directly. Figure 7.4 is an example of the Interpretation Switcher in the absolute mode. The sliders' lengths for the interpretations played by Csalog and Indjic are different, which shows difference in total duration between them. Indjic played this piece much faster than Csalog. And the length of the two green annotations mentioned in the relative mode is nearly the same.

Reference mode

In this mode, the sliders have the same length again. A reference recording is selected and can be changed by clicking the check box under the play button. As it shows in Figure 7.5,

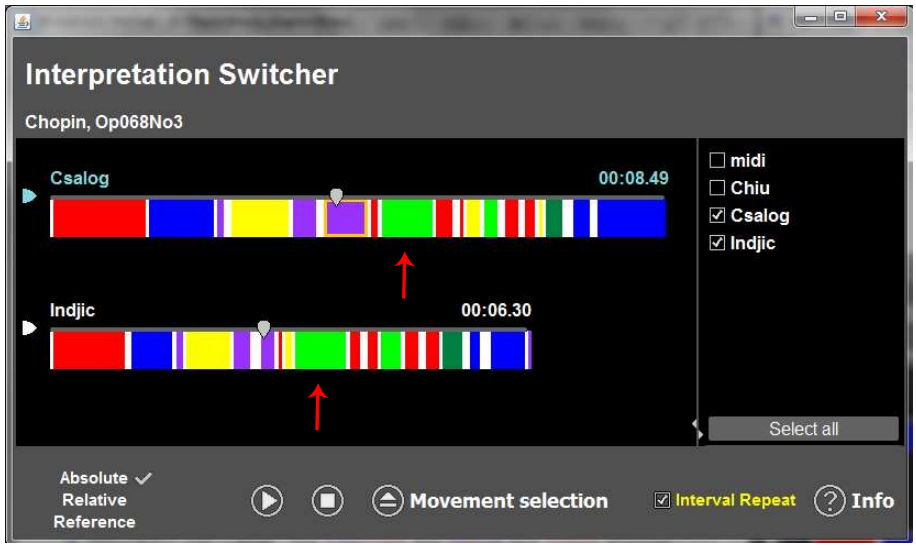


Figure 7.4. Interpretation Switcher in absolute mode.

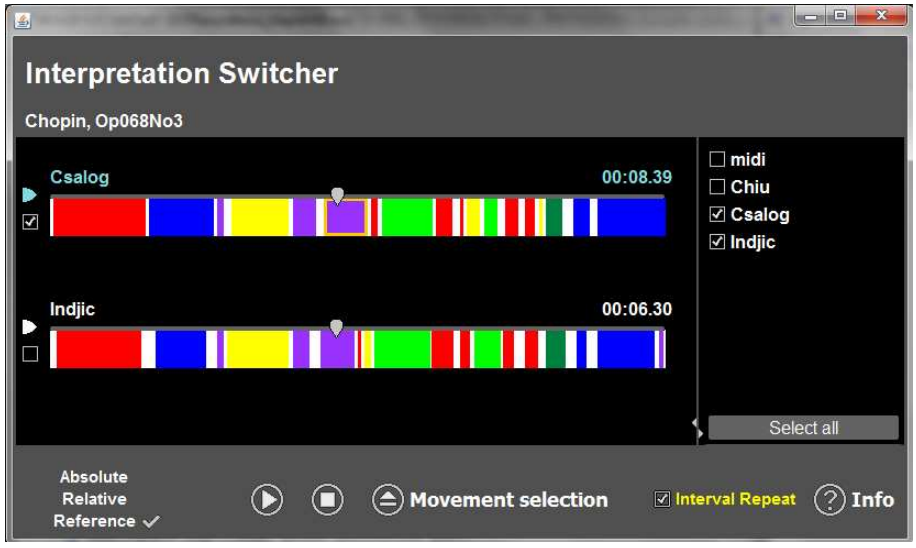


Figure 7.5. Interpretation Switcher in reference mode.

the interpretation performed by Csalog is selected as the reference. The annotations of it keeps the original form. We can find that the distribution of annotations is the same as it is in absolute mode (see Figure 7.4). But the sliders and annotations of other interpretations are temporally warped to run synchronously to the reference recording. In this mode, the sliders will move at the same speed for all the interpretations. The relation between the playing recording and the reference can be observed by the moving speed of the slider. If current playing recording is the reference one, the slider will move with a constant speed. But if we play another recording, the slider will move at nonconstant speed. A faster

speed than normal one mean that performer of the current playing recording played faster than who played the reference recording.

It is important to notice that if we change the reference recording in reference mode, the display of annotations will be changed and the scale of sliders will be adapted to the reference one. The changes are shown in Figure 7.6.

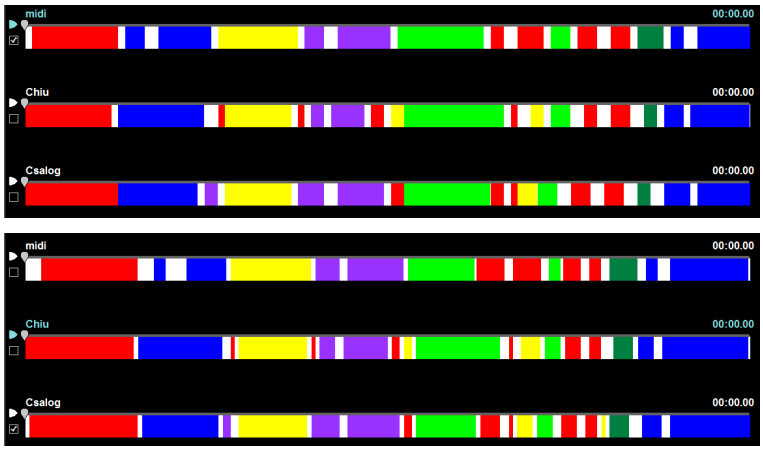


Figure 7.6. Comparison of selecting different reference recordings in reference mode. **Top:** Indjic's interpretation is selected. **Bottom:** Csalog's interpretation is selected.

7.3 Image Mode

The previous extensions focus on representing the relations between different interpretations. But the image mode changes the function of Interpretation Switcher completely. Instead of switching among different interpretations of the same piece of music, we only focus on one interpretation and build an additional annotation field to representing some special images. These images is helpful to represent the details of music. They could be chromagram, matching curve, self-similarity matrix or any images which needed by users.

The advantage of image mode is that audio player, annotation information and visualizing information are integrated together. Firstly, user can not only play a recording but also get specified information of the recording from the image. For example, by adding the chromagram image, one can know the chroma information of the played notes when listening to the recording. Secondly, it helps users to understand the image. For instance, the shift problem occurred when we applying reference-free segmentation. Using image mode of Interpretation Switcher, one can find possible causes of the problem more conveniently than using traditional audio player. Thirdly, users are free to put any related image in the image area. Several examples are introduced in the following part.

In Figure 7.7, folk song OGL49313 is represented by Interpretation Switcher in image mode. The image in annotation field is the chromagram of OGL49313. And it has been warped to be the same length as the slider so that they have a same time scale. The second

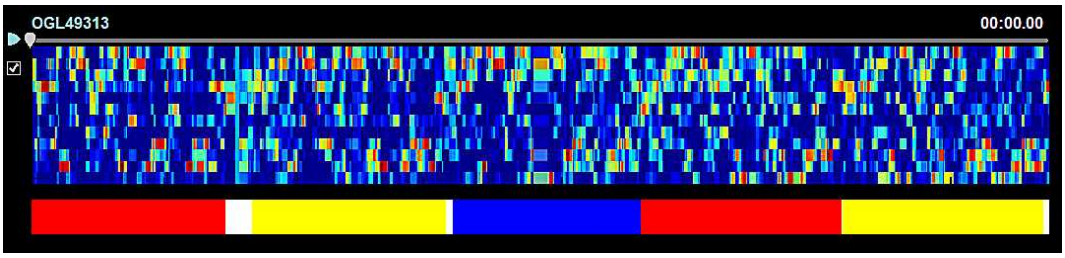


Figure 7.7. An example of Interpretation Switcher (Image mode) representing the chromagram of OGL49313

annotation field shows the segmentation information for this recording, and annotations on it are colored to make them more clearly. The slider will jump to the corresponding position when users left click the image or slider.

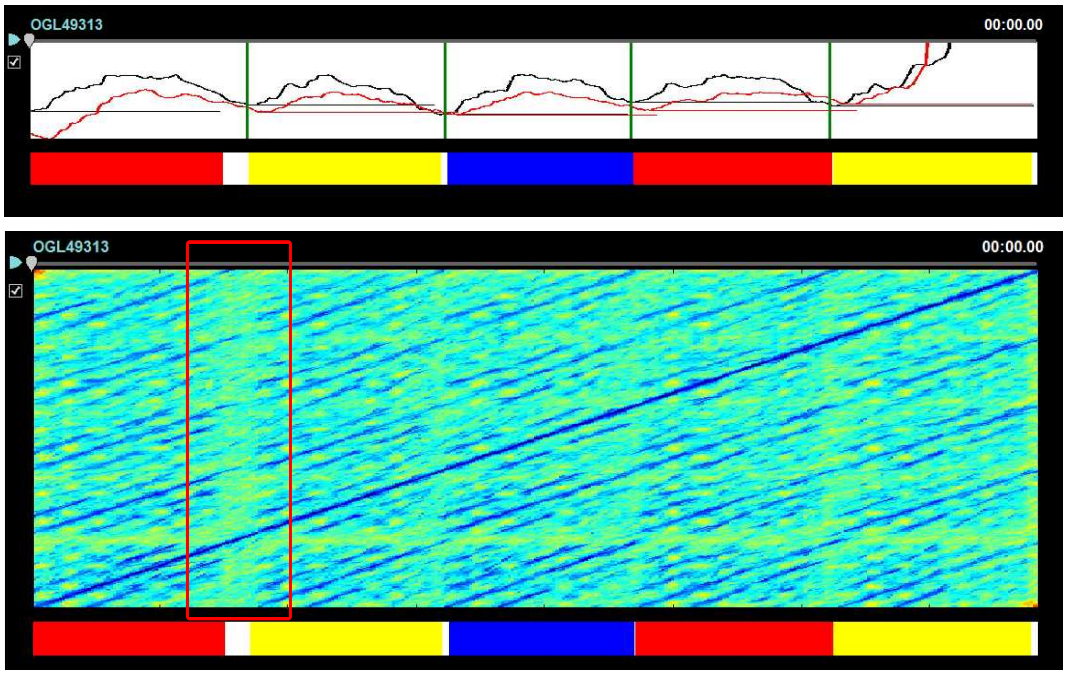


Figure 7.8. An example of Interpretation Switcher (Image mode). **Top:** representing the matching curve of OGL49313; **Bottom:** representing the smoothed self-similarity matrix of OGL49313

Figure 7.8 shows two different examples. The chromagram image is replaced by matching curve image and self-similarity matrix image.

In matching curve image, the black line is computed by reference-based segmentation with CENS(9,1) feature and the red is computed by reference-free segmentation with CENS(11,5) feature, which are introduced in the pervious chapter. The precision, recall and F0-measure of the reference-free segmentation result are all equals to 0.4. The result has an offset problem which has been introduced in Section 5.2.3.1. We can use interpretation switcher to help us find out the cause of the problem. By clicking the image, the

corresponding part is played immediately. With the help of the visualized user interface, it is easier to target the key point. For this example, we can click the green vertical lines, which denote the ground truth segmentation points. Listening to the beginning part of each segmentation, the reason why shift problem occurs in this recording can be find out. That is because the singer forgot what to sing at the end of the first stanza, which makes the recording messed up at that part. According to this mistake, the reference which shifted 3 second are selected since it get a better fitness score (see Section **??**). It is very easy to jump to the segmentation point and play from the corresponding time by using Interpretation Switcher. However, working with a traditional audio player, users have to know the time of each segmentation point and then click the slider to find those points. This example shows one of the usage of the image mode. It is helpful for researchers to represent and analyze their results.

The self-similarity matrix image is another example. In the image, the lighter blue denotes lower similarity while the darker blue denotes higher similarity. There is a light area, noted by red box, at the beginning of second stanza where the singer forgot what to sing. It could be confused when user looks at the result without listening to the recording. With the help of Interpretation Switcher, users can jump to and play that interesting part easily.

Figure 7.9. An example of Interpretation Switcher (Image mode) representing the score of OGL49313's first stanza

Figure 7.9 shows another possible application of image mode. The image is the score of OGL49313's first stanza, which could be considered as a standard stanza of the recording. It has been repeated 5 times in this folksong. We extract the first stanza from the whole piece of recording and use image mode to show its score information.

7.4 User Interaction Extension

Until now the annotations are loaded from the annotation files. In old versions, one has to modify the original annotation files if the annotations in the user interface need to be modified. Here in this version, the user interaction extension helps user to operate annotations directly and easily in Interpretation Switcher. Especially in image mode, it is easier to correct mistakes in annotations according to the image. For example, one can modify or create annotations according to the ground truth line in the matching curve image.

Here, we name these color-coded annotations in the annotation field as markers. The meaning of markers may vary for different recordings, which is set manually.

The user interaction function includes modification, deletion and insertion. User can use mouse and keyboard for operation. There are several rules for makers to limit the modification operation. Firstly, the length of markers should be positive, which also

means the starting position should be on the left of the ending position. Secondly, overlap between two markers is not allow in this case. Respect to these rules, we can limit the available modification area of a marker. Let $T_{S_n}, n \in [1 : N]$ denote the starting time of the nth marker while $T_{E_n}, n \in \mathbb{N}$ denote the ending time of the nth marker. L is the length of the recording. The modifications must satisfy the following rule: $0 \leq T_{S_1} < T_{E_1} < T_{S_2} < T_{E_2} <, ..., < T_{S_N} < T_{E_N} \leq L$.

7.4.1 Mouse Operation

User can use mouse to modify and insert the markers. The left button of mouse is used for jump function in normal music player. In order to avoid the confliction, right button of mouse is selected to operate the markers in our design.

Modification

The modification of a marker could be the starting position modification, the ending position modification and the whole marker position modification. With the first two kind of operation, the length of the marker is changed. But moving the whole marker, the length of the marker stays the same.

As Figure 7.10 shows, we modified the starting position of the second marker colored by yellow. And to modify the ending position is similar. The operation procedure is as following:

- 1. Move the cursor onto the left(right) edge of the marker. The cursor changes to the resize style when it reach a certain position.
- 2. Press the right button of mouse and drag the marker. The border of the marker will be colored by light green to indicate the marker is being modified.
- 3. Release the mouse and the new starting(ending) position will be set.

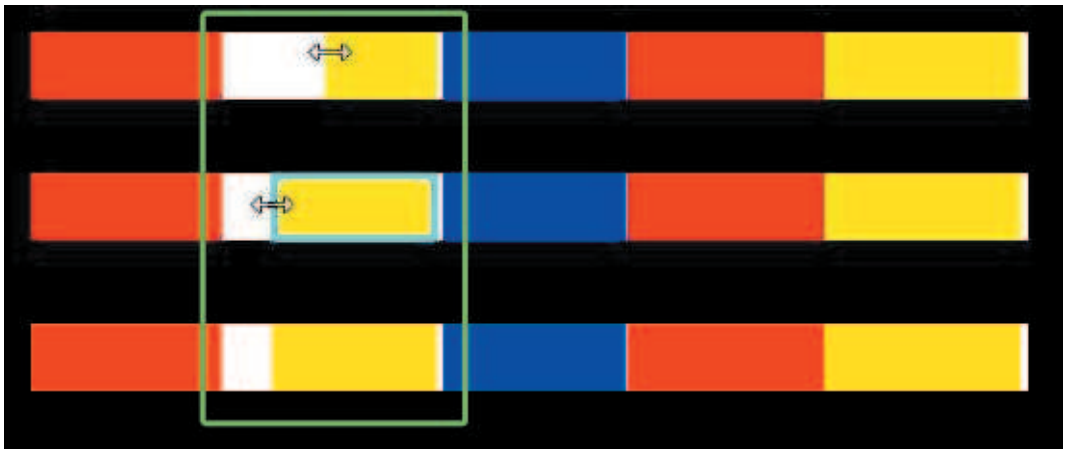


Figure 7.10. The operation procedure of changing the starting position of a marker with mouse

One can also move the maker without changing the length. Figure 7.11 shows the procedure.

- 1. Move the cursor onto the marker (not the edges). The cursor changes to the move style when it reach the right position.
- 2. Press the right button of mouse and drag the marker. The border of the marker will be colored by light green to show the marker is being modified.
- 3. Release the mouse and the new maker position will be set.

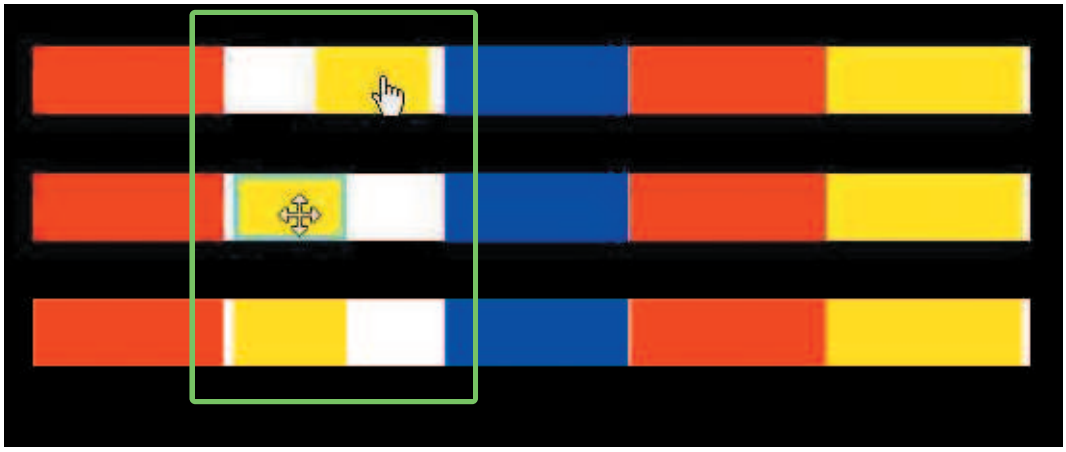


Figure 7.11. The operation procedure of moving a marker with mouse

Insert

User can also insert a new marker. The color of it is set as the default color, the color could be changed manually. The procedure is as following:

- 1. Find a starting position which has not been marked, and move cursor to the position.
- 2. Right press mouse and drag the cursor to the ending position which should not cross any marker.
- 3. Release the mouse and a new maker will be generated.

7.4.2 Keyboard Operation

The keyboard operation consists of deletion and modification of markers. The marker, which would be operated, should be activated by right clicking the mouse. And the activated marker's border will change to be light green as Figure 7.12 shows. The marker can be deactivated by right clicking it again or activating another marker.

Modification

Sometimes modifying markers with mouse is not precise enough since it is hard to control the moving distance of mouse. But with the help of keyboard, one can change the start time of a marker from 4.5s to 4.6s by typing the keyboard once. And the modification step size could be set manually, it could be 0.1s, 1s or even 10s. Therefore, with the help of keyboard, one can adjust markers in a finer level. The modification function applies to the starting position, the ending position and the whole marker position. The introduction of using keyboard to modify markers is as following:

- **Starting position:** Press *Ctrl + left (right) arrow*.
- **Ending position:** Press *Alt + left (right) arrow*.
- **Whole marker position:** Press *left (right) arrow*.

Delete

By typing the delete button, user can delete the activated marker.

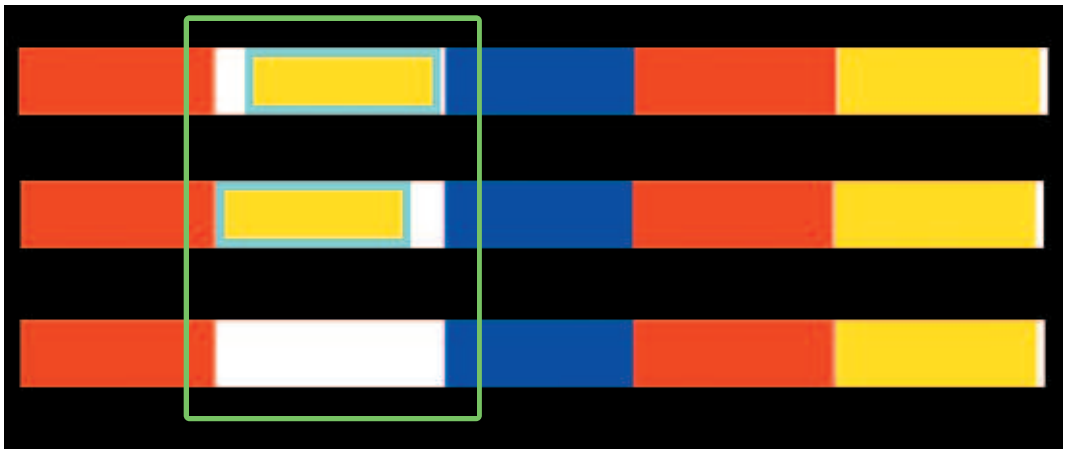


Figure 7.12. Example of moving a marker and deleting a marker with keyboard. **Top:** Marker activated; **Middle:** Marker moved; **Bottom:** Marker deleted.

Color Changing

Except of modifying the position of markers, one can use keyboard for changing the color of markers. A color library is established by reading files. By typing *Shift + left (right) arrow*, the color will be switched to the previous(next) one.

Chapter 8

Interpretation Switcher Documentation

In this chapter, we represent the Java documentation of some main classes of the Interpretation Switcher. As a plug-in of the SyncPlayer, the basic functions of the Interpretation Switcher is called from the SyncPlayer, for example, *play()*, *stop()*, *pause()*, and so on.

Class InterpretationSwitcherAppletWrapper

A wrapper class used to start the applet directly from the IDE.

main

```
Main function of this class. Instantiation of InterpretationSwitcherAppletFramer
```

Class InterpretationSwitcherAppletFramer

This provides an applet viewer such that the applet can be run directly from the IDE. It simulates the behavior of a web browser in terms of calling the applet's methods init(), start(), stop() and destroy().

InterpretationSwitcherAppletFramer

```
Creates an instance of interpretationSwitcherApplet.

Parameters:
            appletname - text of appletname
            width - initial width of applet
            height - initial height of applet
```

Class InterpretationSwitcherApplet

This applet uses the interpretation switcher plugin for syncplayer to provide an user interface for the Interpretation Switcher.

init

```
Initializes Interpretation Swticher.
```

start

```
start() is called immediately after init() and whenever the applet needs
to be restarted after a call of stop().
```

run

```
The run loop of the GUI updates the internal time counter.

Specified by:
        run in interface java.lang.Runnable
```

stop

```
This method is used to stop the applet.
It is allegedly called whenever the applet loses focus.
```

destroy

```
destroy() is called when the application quits.
```

initComponents

```
Initializes GUI all the components are initialized in the function.
```

initSlider

```
Initializes a slider.
Sets orientation, initial & maximum value, name (used to decide
which slider was clicked later on), and changes listener,.

Parameters:
            index - index of interpretation which the slider corresponding to
            verticalOffset - vertical position of the slider
            verticalSpacing - vertical space between two sliders
Returns:
            check box
```

initMarkerLabel

```
    Initializes the markerlabel.
    The action performances of mouse and keyboard are added.

    Parameters:
                beginTime - the beginning time of the marker
                endTime - the end time of the marker
                label - the color index of the marker
                songIndex - the index of the song which marker belongs to
                MarkerNumber - the index of the marker.
    Returns:
                MarkerLabel
```

initCheckBox

```
    Initializes a checkbox.
    Set title, position, bounds, initial state, action listener.

    Parameters:
                title - title of the check box
                index - index of the interpretation which the checkbox corresponding to
    Returns:
                check box
```

alignComponents

```
    Aligns the various components according to window/applet dimensions.
```

alignBasicComponents

```
    Aligns components which are displayed in playBackPanel.
    This function is called by alignComponents().
    Components would be aligned when loading works, selecting different
    displaying interpretations.
```

alignPlayBackComponets

```
    Aligns components which are displayed in playBackPanel.
    This function is called by alignComponents().
    Components would be aligned when loading works, selecting different
    displaying interpretations.
```

alignMarkerLabel

```
    Aligns marker labels.
    The function is called in normal mode.
```

alignMarkerLabelSeg

Aligns marker labels.
The function is called in seg mode.

updatePlaybackPanel

Dynamically updates elements for the interpretations depending on user selection.

updateSelectionPanel

Updates interpretations to selection panel.

updateInfoPanel

Displays meta info about currently active interpretation.

parameter:
index - index of currently active interpretation.

updateSlider

Updates slider positions and time displayed for interpretations.

setMarkerLabelBoundary

Sets the possible modification boundaries for each marker label.
The function is called when annotations are initialized.

Parameters:
pos - index of interpretation which marker label corresponding to

refreshMarkerLabelBoundary

Resets the marker label boundary when markers are modified.

Parameters:
pos - index of interpretation which marker label corresponding to
index - index of the modified marker

stateChanged

Reacts to position changes in the sliders.

Specified by:
stateChanged in interface javax.swing.event.ChangeListener

Parameters:
e - event

switchInterpretation

Updates the GUI to show the active interpretation

Parameters:
index - actived interpretation

deactiveAllMarkers

Deactives all markers.

syncPlayerEvent

Event listener for the syncplayer, updates GUI according to various events

Specified by:
syncPlayerEvent in interface syncplayer.client.SyncPlayerEventListener

Parameters:
event - SyncPlayerEvent

selectAll

Sets all interpretations to be displayed.

deselectAll

Reset the GUI when stop button is pressed or a new work is loaded

reset

Reset the GUI when stop button is pressed or a new work is loaded

getMaxDuration

Gets the maximum duration all over the selected interpretations.

Returns:
maximum duration

getTextWidth

Calculates the length of a string in a given font.

```
    Parameters:
                text - text content
                font - font type of text
    Returns:
                length of the text
```

getTextHeight

```
    Calculates the height of a string in a given font.

    Parameters:
                text - text content
                font - font type of text
    Returns:
                height of the text
```

timeToString

```
    Converts milliseconds to a string with the format MM:SS.MS.

    Parameters:
                time - time in double format
    Returns:
                time in MM:SS.MS format
```

getScaleOfSlider

```
    Gets the scale of sliders. Relates to the maximum duration and
    the width of playbackScrollPane

    Returns:
                scale of sliders
```

Class InterpretationSwitcher

The class of the Module of InterpretationSwitcher.

addSyncPlayerEventListener

```
    Sets the external syncPlayerEventListener

    Parameters:
                syncPlayerEventListener
```

openWorks

Opens basic config file Config file containing info about the works to load, including titles and paths.

Parameters:
codeBase - path of work config file
Returns:
true if work config file is loaded, otherwise false

openInterpretation

Opens interpretation files. CSV file contains config information of works.

Parameters:
workIndex - index of work
subWorkIndex - index of movement
progressBar - progress bar of loading files

openInterpretationSeg

Opens interpretation files. openInterpretationSeg is used when InterpretationSwitcher is in segMode. In segMode, csv file and color file are not required.

Parameters:
isFile - loaded sync file whose path information is used

openFile

Opens audio files.

Parameters:
file - loaded sync file whose path information is used
startPlaying - flag of whether play the loaded audio file or not
Returns:
true if successfully loads audio file, otherwise false.

openAudioContainer

Opens the sepecified AudioContainer.

Parameters:
ac - AudioContainer used for opening the audio stream
startPlaying - flag of whether play the loaded audio file or not

closeFile

Closes the loaded audio file

play

Calls the audioPlay.play() function and plays the loaded file

pause

Calls the audioPlay.pause() function and pauses the loaded file

stop

Calls the audioPlay.stop() function and stops the loaded file

reset

Calls the audioPlaygetAudioContainer().reset() function and resets the loaded file

skip

Jumps to desired position in interpretation and possibly switch the active interpretation.

Parameters:
position - desired postilion
index - index of selected interpretation
activeIndex - current playing interpretation

getStatus

Gets the status of the audioPlayer.

Returns:
status of the audioPlayer

getWorks

Gets the loaded files.

Returns:
loaded files

getStretchFactor

```
    Gets stretch factor for audioPlayer.

    Returns:
                stretch factor for audioPlayer
```

getTime

```
    Gets time of current playing interpretation.

    Returns:
                time of current playing interpretation
```

getLength

```
    Gets length of current playing interpretation.

    Parameters:
                index - index of the interpretation
    Returns:
                length of the interpretation
```

getTimeCorrespondingToTimeInActiveInterpretation

```
    Gets the synchronized time of the interpretation, which is corresponding to
    the current playing interpretation. The synchronized time is set in sync file.

    Parameters:
                time_ms - currant playing time
                songIndex - interpretation which is needed to be synchronized
                activeIndex - currant playing interpretation
    Returns:
                synchronized time
```

getTimeCorrespondingToTimeInAnyInterpretation

```
    Gets the synchronized time of the interpretation, which is corresponding to
    another interpretation. The synchronized time is set in sync file.

    Parameters:
                time_ms - current playing time
                questionSongIndex - index of the interpretation which is needed
                                    to be synchronized
                anySongIndex - index of the reference interpretation
    Returns:
                synchronized time
```

getLongTitles

Gets long titles of interpretations.

Returns:
long titles of interpretations

getShortTitles

Gets short titles of interpretations.

Returns:
long titles of interpretations

getMetaInfo

Gets meta information of interpretations The meta information has two versions, English version and German version.

Returns:
meta information

getInterpretationFiles

Gets files of interpretations.

Returns:
files of interpretations

getTitleOfWork

Gets titles of interpretations Titles are in two versions,
English and German versions.

Returns:
titles of interpretations

getTitleOfWorkGerman

Gets titles(German version) of interpretations.

Returns:
titles(German version) of interpretations

getTitleOfWorkEnglish

Gets titles(English version) of interpretations.

```
    Returns:
                titles(English version) of interpretations
```

setEncoding

```
    Sets encoding type.

    Parameters:
                encoding -  encoding type
```

Class Interpretation

A class of interpretation. Holds all basic information for each interpretation.

Interpretation

```
    Creates a new instance of Interpretation

    Parameters:
                fileName - text of wave file name
                longTitle - text of long title of the interpretation
                shortTitle - text of short title of the interpretation
                infoTextEnglish - text information of the interpretation
                                  in English
                infoTextGerman -  text information of the interpretation
                                  in German
```

Class InterpretationSwitcherTextFile

Reads files of type AudioSwitcherTextFile. Reads the sync file. Sync file contains some basic information of the audio recording, such as duration, interpretation number, and so on.

InterpretationSwitcherTextFile

```
    Creates a new instance of InterpretationSwitcherTextFile
```

openFile

```
    Opens interpretationSwitcherTextFile

    Parameters:
                file - sync file
                encoding - encoding type
    Returns:
                interpretations' information in sync file
```

saveAs

```
    Saves as InterpretationSwitcherTextFile

    Parameters:
                file - the config file
                encoding - the encoding type
```

save

```
    Renews InterpretationSwitcherTextFile

    Parameters:
                encoding - encoding type
```

writeRemoteAudioFileIDs

```
    Writes remote audiofileIDs

    Parameters:
                remote_splf - SyncPlayListFile
```

getInterpretations

```
    Gets interpretations

    Returns:
                interpretations
```

setInterpretations

```
    Sets interpretations

    Parameters:
                interpretations
```

Class AudioContainer

This class extends the SyncPlayer AudioContainer to manipulate its behavior. When the user opens an audio file, the program searches for an existing meta information file to get the metainformation of the file. If it does not exists it creates one.

AudioContainer

```
    Creates a new instance of AudioContainer
```

Class TitleLabel

The label shows the main title of some component, such as the title of work, the title of help panel.

TitleLabel

```
    Creates a new instance of TitleLabel

    Parameters:
                title - title text
                font - font of title
```

Class TitleLabel

The label shows the current displayed interpretation label on the top of each slider. Each interpretation has one TempLabel.

TempLabel

```
    Creates a new instance of TempLabel

    Parameters:
                title - text shown in the label
                index - index of the song which the label belongs to
                horizontalOffset - horizontal offset of the label
                width - width of the label
                verticalOffset - vertical offset of the label
                verticalSpacing - vertical space between two labels
```

updateLabel

```
    Updates label text font

    Parameters:
                font - font style of the label
                maxWidth - maximum width of the label
                color - color of the text
```

getTextWidth

```
    Gets the width of the label

    Parameters:
                text - text of the label
                font - font of the label
```

```
    Returns:
                width of the label
```

Class MarkerLabel

The panel contains annotation information. Each interpretation could have several annotations.

MarkerLabel

```
    Creates a new instance of marker.

    Parameters:
                start - starting position of marker
                end - ending position of marker
                label - color label of marker
                index - song index of marker belongs to
```

isSelected

```
    Tests if the marker is selected.

    Returns:
                true if the marker is selected, otherwise false
```

isActiveFlag

```
    Tests if the marker is actived

    Returns:
                true if the marker is active by right click, otherwise false
```

setSelected

```
    Sets the marker to be selected

    Parameters:
                selected - flag of whether marker label is selected
```

setSongIndex

```
    Sets song index.

    Parameters:
                songIndex - song index of the marker belongs to.
```

setSongLabel

Sets the color label of the maker.

Parameters:
label - color number

setMaxS_Left

Sets the left boundary of the marker.

Parameters:
Left - minimal value of the starting position in second

setMaxE_Right

Sets the right boundary of the marker.

Parameters:
Left - maximum value of the end position in second

setActiveFlag

Sets the maker state.

Parameters:
activeFlag - true if activated, otherwise false

setBoundary

Sets the marker boundaries.

Parameters:
maxLeft - left boundary
maxRight - right boundary

setStart_pos

Gets the start position of marker

Parameters:
start_pos - starting position of marker

setEnd_pos

Sets the end position of marker.

Parameters:
end_pos - ending position of marker

getSongIndex

Gets the song index.

Returns:
song index of the marker belongs to.

getSongLabel

Gets the song color label.

Returns:
song label of the marker belongs to.

getEnd_pos

Gets the end position of marker.

Returns:
ending position of marker

getStart_pos

Gets the start position of marker

Returns:
starting position of marker

getMarkerLength

Gets the length of marker

Returns:
length of marker

getMaxS_Left

Gets the left boundary of marker

Returns:
left boundary of marker

getMaxE_Righ

```
Gets the right boundary of maker

Returns:
            left boundary of marker
```

Class LabelFile

Label file contains the information of annotations and color class. The information of an annotation consists the starting time, the ending time and the label of color. Color type is stored in a hash table. The hash code is corresponding to the label of annotation.

LabelFile

```
Creates a new instance of labelFile A label file contains the information
of annotations and color type.

Parameters:
            file - the labelfile
            encoding - the encoding type
```

Class IconButton

Extension of JButton. Buttons, which have specific icons are belongs to this class, such as PlayPauseButton, StopButton. The iconButton has some specific interactions for example the color changes when mouse moves on. The interaction of the button is set in synth.xml.

IconButton

```
Creates a new instance of IconButton

Parameters:
            image - icon of the button
            buttonToolTip - tool tip of the button
            buttonText - text of the button
```

Class ColorFile

Color file sets the color class for annotations.

ColorFile

```
Creates a new instance of ColorFile.
```

```
    Parameters:
                file - color file name
                encoding - file encoding type.

openChordLabelFiles

    Opens chord label files.

    Parameters:
                labelFile - the label information
                labelArray - hashtable of each color
    Returns:
                color label of each marker
```

Class LogoPanel

The panel with an Image as a background The image will not be resized. It is placed at the bottom, centered.

LogoPanel

```
    Creates a new instance of LogoPanel

    Parameters:
                logoURL - url of logo image
```

LogoPanel

```
    Creates a new instance of LogoPanel

    Parameters:
                image - logo image
                posX - x-axis position of image
                posY - y-axis position of image
```

paintComponent

```
    Paints the logo image.

    Overrides:
                paintComponent in class javax.swing.JComponent

    Parameters:
                g - graphics component
```

getBgPositions

```
    Gets the logo position

    Returns:
                logo position
```

Class TempButton

The button which is the current activated interpretation switch button places on the left of each slider. Each interpretation has one TempButton.

TempButton

```
    Creates a new instance of TempButton.

    Parameters:
                title - text shown in the button
                index - index of the song which the button belongs to
                horizontalOffset - horizontal offset of the button
                width - width of the button
                verticalOffset - vertical offset of the button
                verticalSpacing - vertical space between two buttons
                icon - icon of the button
                string - name part of the action command of the button
                i - index part of the action command of the button
                j - index j in action command of the button
```

Class TempAnnotationPanel

The panel holds annotations for each interpretation. Each interpretation has one TempAnnotationPanel.

```
TempAnnotationPanel

    Creates a new instance of TempAnnotationPanel

    Parameters:
                title - text shown in the panel
                index - index of the song which the panel belongs to
                horizontalOffset - horizontal offset of the panel
                width - width of the panel
                verticalOffset - vertical offset of the panel
                verticalSpacing - vertical space between two panels
                height - height of the panel
                color - color of the panel
                string - text part of the panel name
                pos - index part of the panel name
```

Chapter 9

Conclusion

This thesis could be separate into two main part. In the first part, we introduce the automatic repetitive music segmentation procedure. The reference-based segmentation [15] is developed by Mueller, Grosche and Wiering. We experiment on the Dutch folk song dataset, which is used in [15]), and get a closed result. This result is used as a baseline for our following experiment. Based on reference-based segmentation, we develop reference-free segmentation which can segment music without any additional reference. The precision of the reference-free segmentation result decreases since the optimal reference step is sensitive to the quality and the structure of the music. And the conditions of actual recordings are complex. Therefore, automatically selected reference is not as precise as the manually generated reference. However, using the reference-free segmentation can segment most of recordings and derive the most representative passage of the recording. In the second part of this thesis we introduce an application to visualize and navigate the segmentation results. In particular, we describe several novel functionalities, which have been implemented as plug-ins for the SyncPlayer framework.

Our main contributions can be summarized as follows. Firstly, we develop an automatic segmentation procedure which can segment repetitive music without additional reference file. Secondly, we extend the SyncPlayer with some useful functions.

As future work, one could improve he efficiency and accuracy of the reference-free segmentation. For example, using single loop instead of double loop in optimal reference selection step to reduce the time cost. Even further, self-similarity matrix could be used for segmenting music also. For the SyncPlayer, more extensions could be developed considering different usages. For example developing a zoom in function for image mode, which helps to show the image clearly.

Appendix A

Reference-free Segmentation Result

In this chapter, the reference-free segmentation result is represented for every recording.

Table of Segmentation Evaluation Results

In Table A.1, the evaluation results of some segmentation strategies introduced in this thesis are represented.

Strategy 1: reference-based segmentation (CENS(9,1)).

Strategy 2: reference-free segmentation with first stanza strategy (CENS(11,5)).

Strategy 3: reference-free segmentation with best stanza strategy (CENS(11,5)).

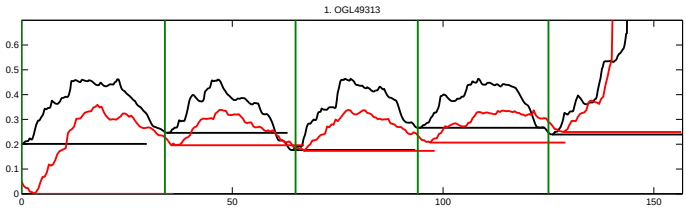
Strategy 4: reference-free segmentation (CENS(11,5)).

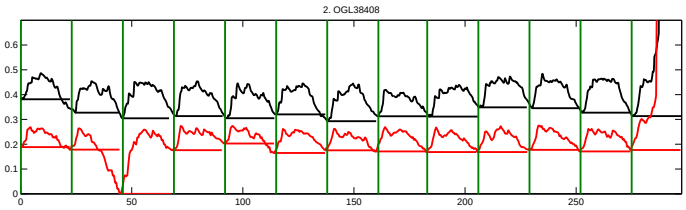
Figures of Reference-free Segmentation Result

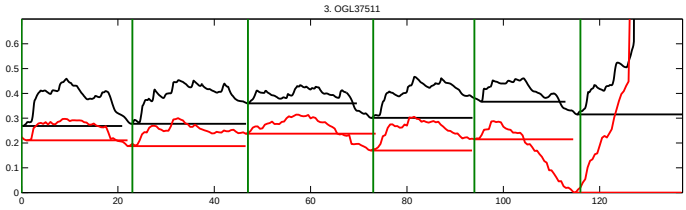
The figures of the segmentation results for every audio recording are represented. These segmentation results are derived by reference-free segmentation, which correspond to the Strategy 4 in Table A.1.

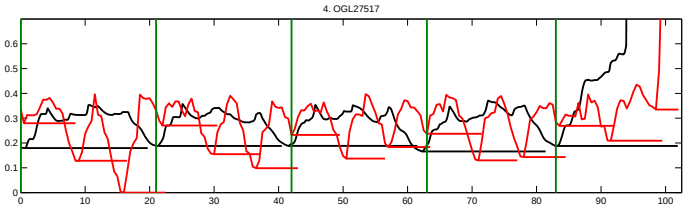
Table A.1. Summary

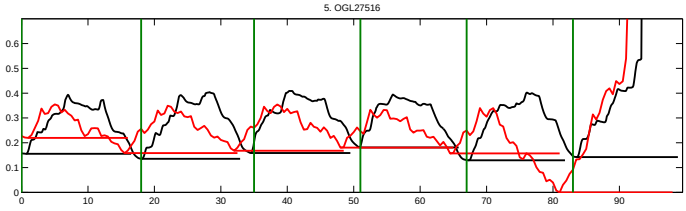
No.	Name	Stanza	Strategy1			Strategy2			Strategy3			Strategy4		
			P	R	F	P	R	F	P	R	F	P	R	F
1	OGL49313	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.400	0.400	0.400
2	OGL38408	13	1.000	1.000	1.000	0.923	0.923	0.923	1.000	1.000	1.000	1.000	1.000	1.000
3	OGL37511	6	1.000	1.000	1.000	0.833	0.833	0.833	1.000	1.000	1.000	1.000	1.000	1.000
4	OGL27517	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.333	1.000	0.500
5	OGL27516	6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.333	0.333	0.333
6	OGL27515	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
7	OGL26805	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
8	OGL25011	11	0.909	0.909	0.909	0.909	0.909	0.909	0.909	0.909	0.909	0.000	0.000	0.000
9	OGL25010	10	1.000	1.000	1.000	0.800	0.800	0.800	1.000	1.000	1.000	0.800	0.800	0.800
10	OGL25009	34	1.000	1.000	1.000	1.000	1.000	1.000	0.971	1.000	0.986	0.853	0.853	0.853
11	OGL19101	24	1.000	1.000	1.000	0.923	1.000	0.960	1.000	1.000	1.000	1.000	1.000	1.000
12	NLB70134	10	0.900	0.900	0.900	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.500	0.500
13	NLB70238	7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
14	NLB71227	14	1.000	1.000	1.000	0.857	0.857	0.857	1.000	1.000	1.000	1.000	1.000	1.000
15	NLB72355	12	1.000	0.833	0.909	0.087	0.167	0.114	1.000	1.000	1.000	0.000	0.000	0.000
16	NLB72395	12	0.300	0.500	0.375	0.333	0.750	0.462	0.393	0.917	0.550	0.188	1.000	0.316
17	NLB72886	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
18	NLB73374	10	0.889	0.800	0.842	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
19	NLB73993	14	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.143	0.143	0.143
20	NLB74028	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.667
21	NLB74234	9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
22	NLB74328	7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
23	NLB74336	12	1.000	1.000	1.000	0.478	0.917	0.629	0.800	1.000	0.889	1.000	1.000	1.000
24	NLB74437	8	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.500	0.500
25	NLB74603	27	0.900	1.000	0.947	0.964	1.000	0.982	0.964	1.000	0.982	1.000	1.000	1.000
26	NLB74613	17	0.875	0.824	0.848	0.293	0.706	0.414	1.000	1.000	1.000	0.938	0.882	0.909
27	NLB74649	8	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
28	NLB74754	8	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
29	NLB75059	7	0.833	0.714	0.769	0.750	0.857	0.800	0.857	0.857	0.857	0.000	0.000	0.000
30	NLB75063	7	0.857	0.857	0.857	0.857	0.857	0.857	1.000	1.000	1.000	0.857	0.857	0.857
31	NLB75068	10	1.000	0.900	0.947	0.471	0.800	0.593	0.400	0.800	0.533	0.149	0.700	0.246
32	NLB75073	5	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
33	NLB75079	6	0.667	0.667	0.667	0.667	0.667	0.667	1.000	1.000	1.000	1.000	1.000	1.000
34	NLB75158	4	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.500	0.500
35	NLB75167	4	0.600	0.750	0.667	0.750	0.750	0.750	1.000	1.000	1.000	0.000	0.000	0.000
36	NLB75174	8	1.000	0.750	0.857	0.500	1.000	0.667	1.000	1.000	1.000	1.000	1.000	1.000
37	NLB75249	10	1.000	1.000	1.000	0.263	0.500	0.345	1.000	1.000	1.000	1.000	1.000	1.000
38	NLB75325	9	0.778	0.778	0.778	0.188	0.333	0.240	0.375	0.667	0.480	0.111	0.111	0.111
39	NLB75431	10	0.900	0.900	0.900	0.900	0.900	0.900	1.000	1.000	1.000	1.000	1.000	1.000
40	NLB75612	7	1.000	0.571	0.727	1.000	1.000	1.000	0.875	1.000	0.933	1.000	1.000	1.000
41	NLB75616	12	1.000	1.000	1.000	1.000	1.000	1.000	0.923	1.000	0.960	1.000	1.000	1.000
42	NLB75619	9	1.000	1.000	1.000	0.818	1.000	0.900	0.818	1.000	0.900	1.000	1.000	1.000
43	NLB75831	14	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
44	NLB76271	7	0.143	0.143	0.143	0.857	0.857	0.857	0.857	0.857	0.857	0.096	1.000	0.175
45	NLB76426	12	0.667	0.667	0.667	0.833	0.833	0.833	1.000	1.000	1.000	0.083	0.083	0.083
46	NLB76495	7	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.714	0.714	0.714
47	NLB76632	10	0.800	0.800	0.800	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	summary	9.9	0.915	0.899	0.904	0.835	0.898	0.857	0.939	0.979	0.954	0.713	0.763	0.715

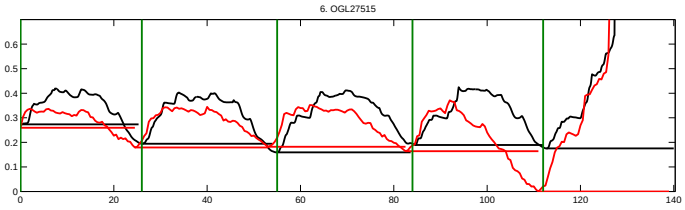

1. OGL49313

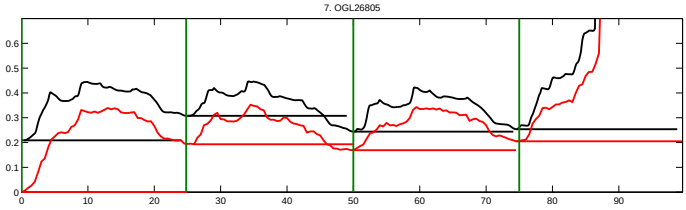

2. OGL38408

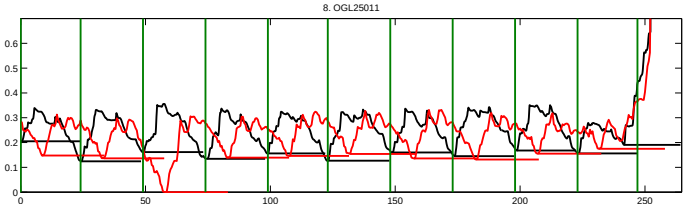

3. OGL37511

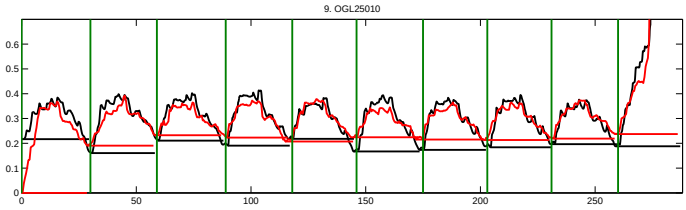

4. OGL27517

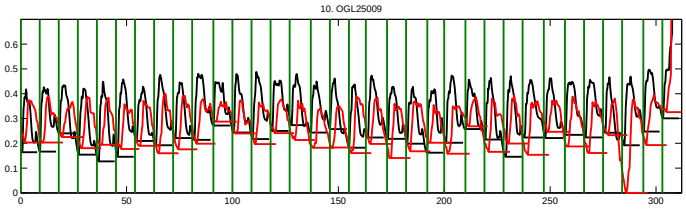

5. OGL27516

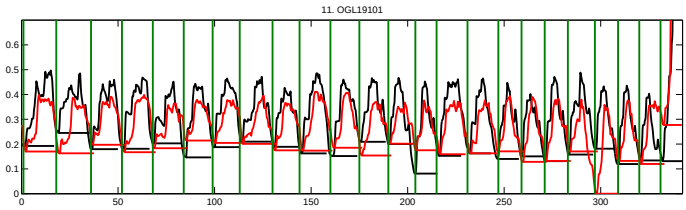

6. OGL27515

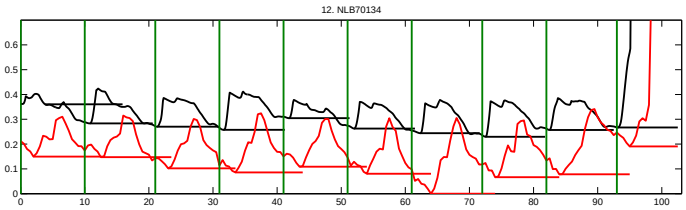

7. OGL26805

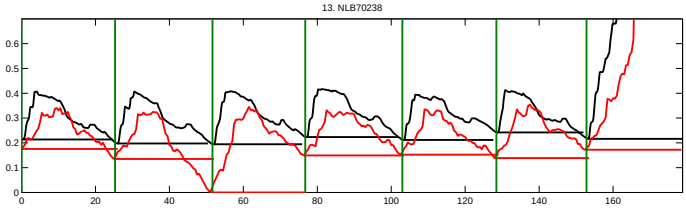

8. OGL25011

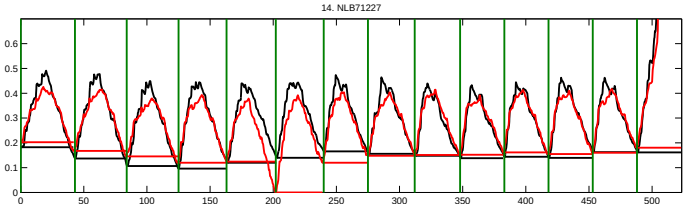

9. OGL25010

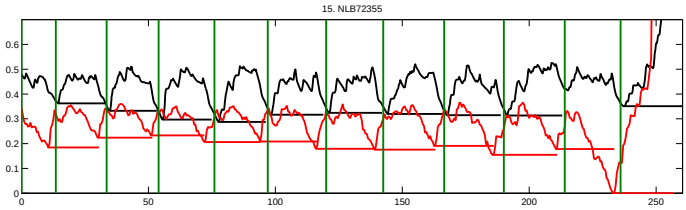

10. OGL25009

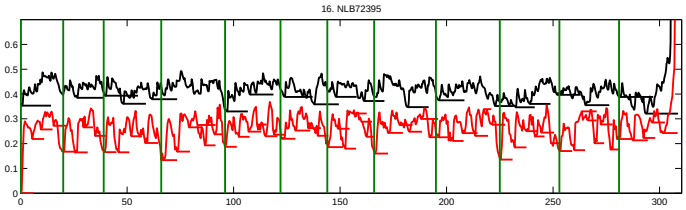

11. OGL19101

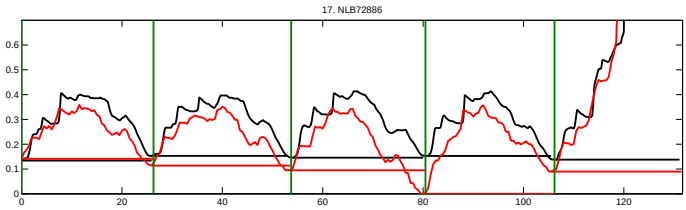

12. NLB70134

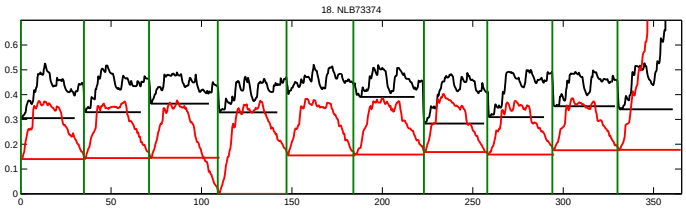

13. NLB70238

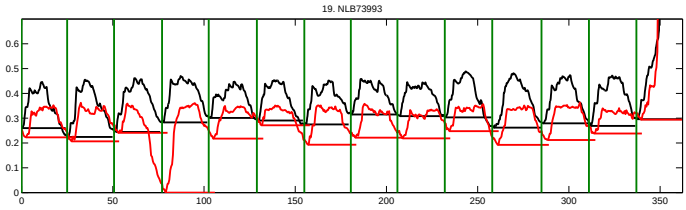

14. NLB71227

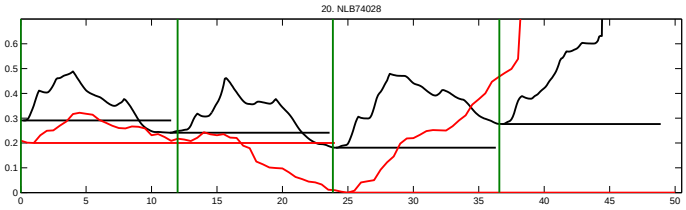

15. NLB72355

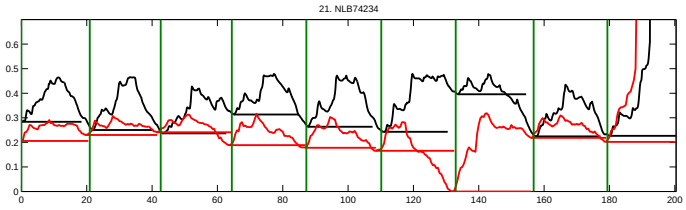

16. NLB72395

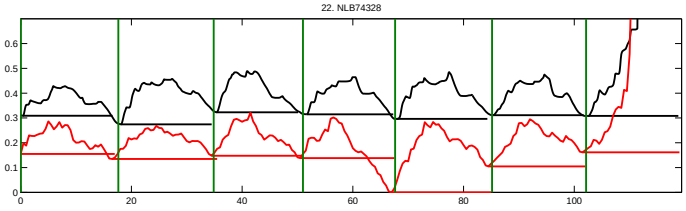

17. NLB72886

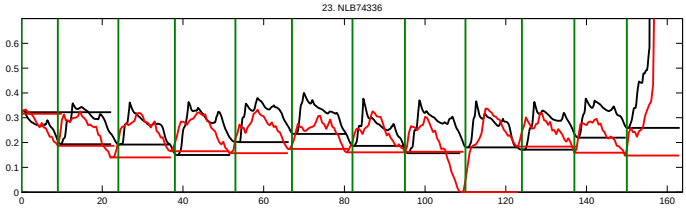

18. NLB73374

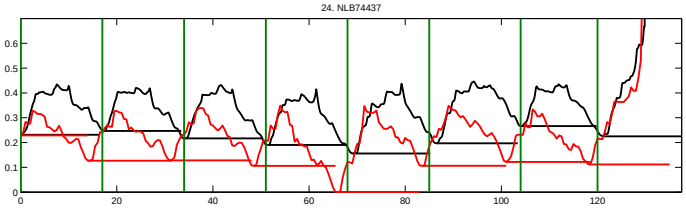

19. NLB73993

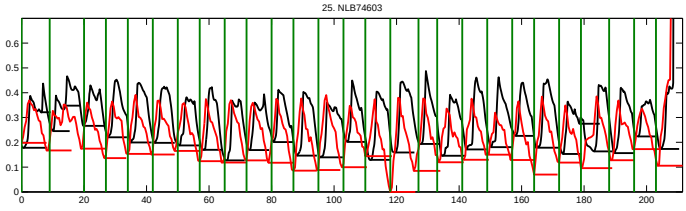

20. NLB74028

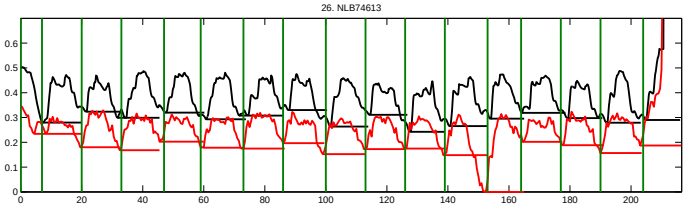

21. NLB74234

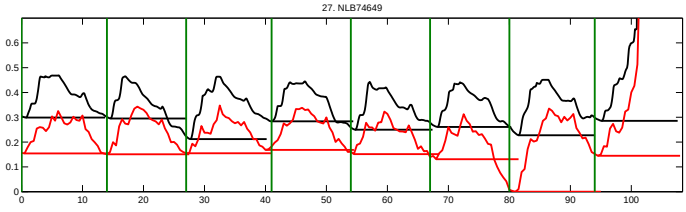

22. NLB74328

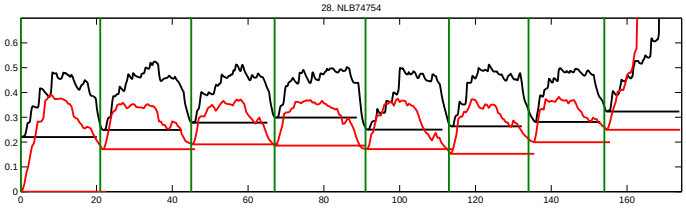

23. NLB74336

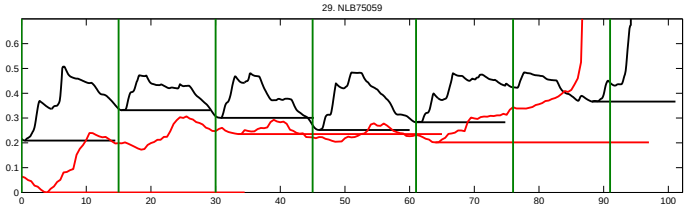

24. NLB74437

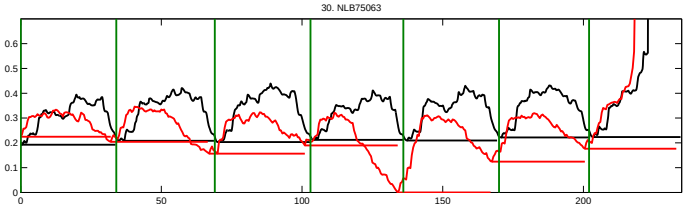

25. NLB74603

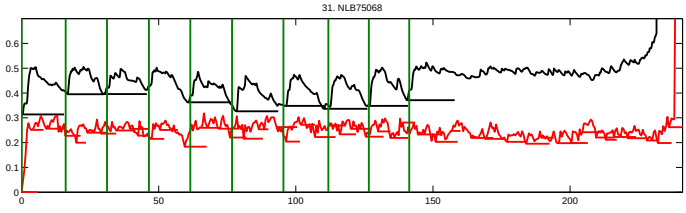

26. NLB74613

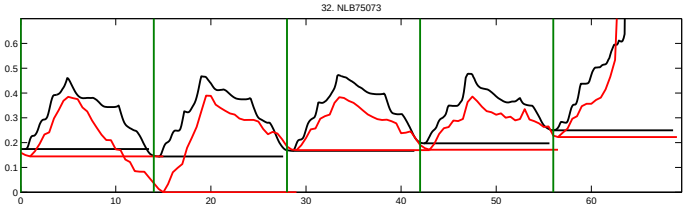

27. NLB74649

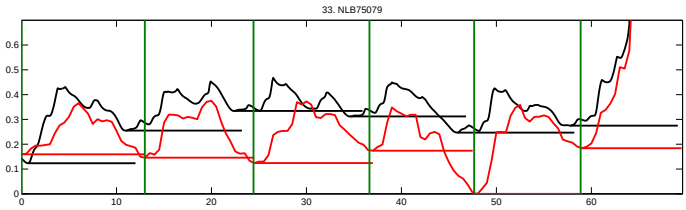

28. NLB74754

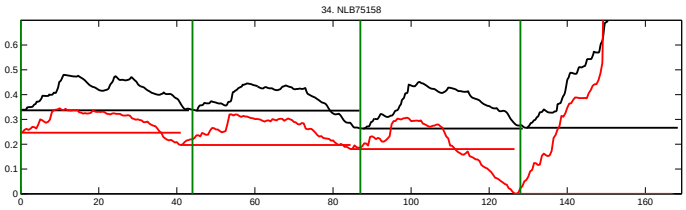

29. NLB75059

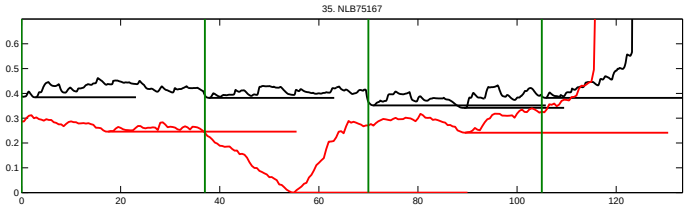

30. NLB75063

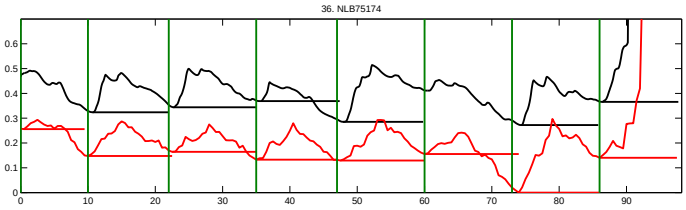

31. NLB75068

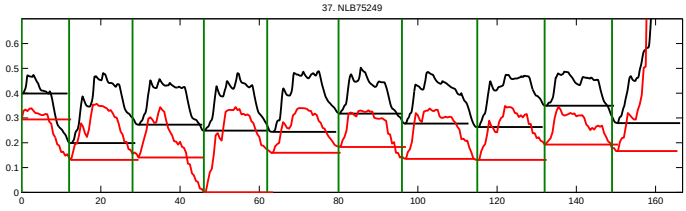

32. NLB75073

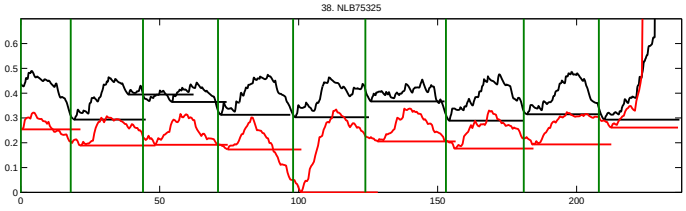

33. NLB75079

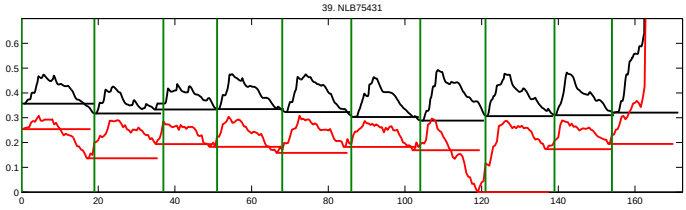

34. NLB75158

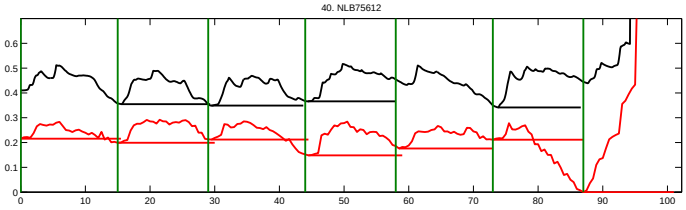

35. NLB75167

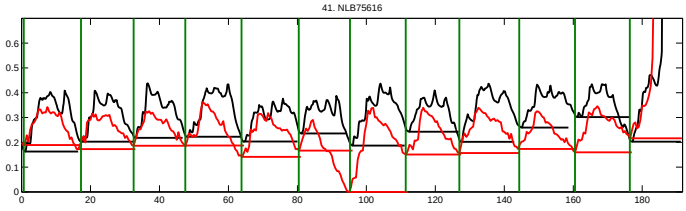

36. NLB75174

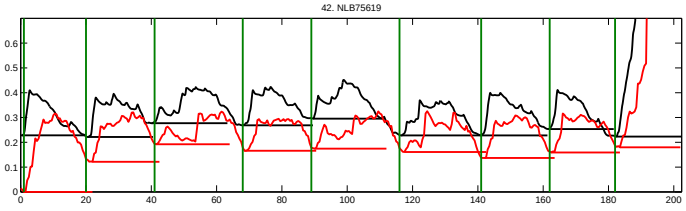

37. NLB75249

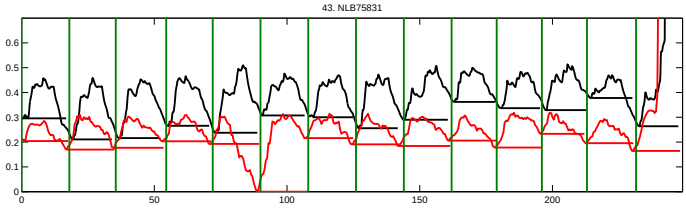

38. NLB75325

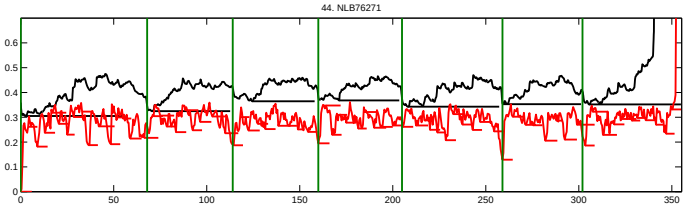

39. NLB75431

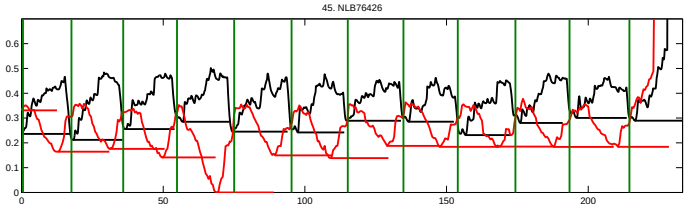

40. NLB75612

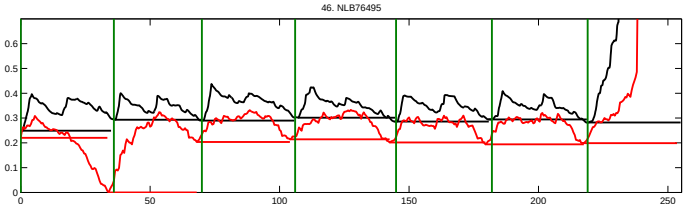

41. NLB75616

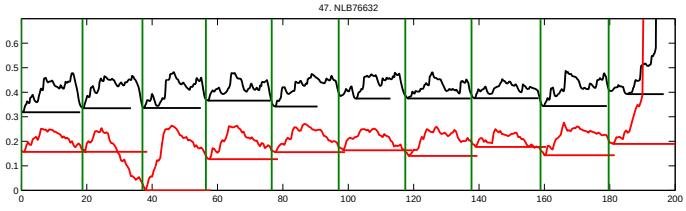

42. NLB75619


43. NLB75831


44. NLB76271


45. NLB76426


46. NLB76495


47. NLB76632

List of Figures

List of Tables

Bibliography

[1] Anthony Brandt, *How music makes sense*. Website, January 2008. http://cnx.org/content/m12953/latest/.

[2] M. A. Bartsch and G. H. Wakefield, *Audio thumbnailing of popular music using chroma-based representations*, IEEE Transactions on Multimedia, 7 (2005), pp. 96–104.

[3] A. de Cheveigné and H. Kawahara, *YIN, a fundamental frequency estimator for speech and music.*, Journal of the Acoustical Society of America (JASA), 111 (2002), pp. 1917–1930.

[4] P. Dorrell, *What is Music?: Solving a Scientific Mystery*, Lulu.com, 2005.

[5] C. Duxbury, J. P. Bello, M. Davies, and M. Sandler, *Complex Domain Onset Detection For Musical Signals*, Proc. of the 6th Int. Conference on Digital Audio Effects, September 2003.

[6] D. FitzGerald and J. Paulus, *Unpitched percussion transcription*, in Signal Processing Methods for Music Transcription, Springer, 2006, pp. 131–162.

[7] C. Fremerey, F. Kurth, M. Müller, and M. Clausen, *A demonstration of the SyncPlayer system*, in Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR), Vienna, Austria, Sept. 2007, pp. 131–132.

[8] E. Gómez, *Tonal Description of Music Audio Signals*, PhD thesis, UPF Barcelona, 2006.

[9] M. Goto, *A chorus-section detecting method for musical audio signals*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Hong Kong, China, 2003, pp. 437–440.

[10] D. J. Hargreaves, *The effects of repetition on liking for music*, Journal of Research in Music Education, 32 (1984), pp. 35–47.

[11] F. Kurth, M. Müller, D. Damm, C. Fremerey, A. Ribbrock, and M. Clausen, *SyncPlayer - an advanced system for multimodal music access*, in Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), London, UK, Nov. 2005, pp. 381–388.

[12] G. H. W. Mark A. Bartsch, *Audio thumbnailing of popular music using chroma-based representations*, IEEE Transactions on Multimedia, 7 (2005).

[13] R. Middleton, *Form*, in Key terms in popular music and culture, B. Horner and T. Swiss, eds., Wiley-Blackwell, Sept. 1999.

[14] M. Müller, *Information Retrieval for Music and Motion*, Springer Verlag, 2007.

[15] M. Müller, P. Grosche, and F. Wiering, *Robust segmentation and annotation of folk song recordings*, in Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR), Kobe, Japan, Oct. 2009, pp. 735–740.

[16] M. Müller and F. Kurth, *Towards structural analysis of audio recordings in the presence of musical variations*, EURASIP Journal on Advances in Signal Processing, 2007 (2007).

[17] M. Müller, F. Kurth, and M. Clausen, *Audio matching via chroma-based statistical features*, in Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), 2005, pp. 288–295.

[18] M. Müller, F. Kurth, and T. Röder, *Towards an efficient algorithm for automatic score-to-audio synchronization*, in Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain, Oct. 2004, pp. 365–372.

[19] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall Signal Processing Series, 1993.

[20] R. N. Shpard, *Circularity in judgments of relative pitch*, Journal of the Acoustical Society of America, 36 (1964), pp. 2346–2353.

[21] P. van Kranenburg, J. Garbers, A. Volk, F. Wiering, L. Grijp, and R. Veltkamp, *Towards integration of MIR and folk song research*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Vienna, AT, 2007, pp. 505–508.

[22] Wikipedia, *Precision and recall*. `http://en.wikipedia.org/wiki/Precision_and_recall`, Retrieved 06.06.2011.

[23] ———, *Woo (beethoven)*. `http://en.wikipedia.org/wiki/WoO`, Retrieved 15.07.2011.