

# Colorscore - Visualization and Condensation of Structure of Classical Music

Aki Hayashi\*  
Ochanomizu University

Takayuki Itoh†  
Ochanomizu University

Masaki Matsubara‡  
Keio University

## 1 INTRODUCTION

It is not always easy to quickly understand musical structure of orchestral scores for classical music works, because the works contain many staves of instruments. This poster presents "Colorscore", a technique for visualization and condensation of musical scores. Colorscore supports two requirements for composers, arrangers and players: overview and arrangement. Colorscore divides each track of the score into note-blocks, and then determines their roles. Colorscore then displays all the note-blocks in one display space to provide the overview, so that novice people can quickly understand the musical structures. In addition, Colorscore supports vertical condensation which reduces the number of displayed tracks, and horizontal condensation which reduces the display space. It is especially useful as hints to rearrange music to smaller bands.

## 2 TECHNICAL COMPONENTS OF COLORSCORE

### 2.1 Analysis of musical structure

Colorscore supports SMF (Standard MIDI file) as input data. It requires note information of MIDI, including pitch, strength, duration and timing. Colorscore firstly read the SMF file, and divides each track of the score into note-blocks. Then, Colorscore determines roles of note-blocks by matching their notes to patterns given by users.

#### 2.1.1 Providing the pattern to decide the role

First of all, Colorscore requires the patterns used to determine the roles of the note-blocks. In this paper, "pattern" means a short set of notes which consist of just one track in MIDI format. Currently we assume that patterns are given by expert users. Colorscore determines whether each block plays melodies or accompaniments, as a "role." As regard to melodies, Colorscore supposes to input basic phrases of several main melodies. At the same time, Colorscore supposes to input only typical rhythms for the accompanying phrases such as harmonic or bass accompaniments: it does not analyze transition of intervals for accompaniments. We think the accompaniments are often characterized by repeated rhythm rather than by transition of interval, and therefore we designed Colorscore to input patterns of accompaniments as rhythms.

#### 2.1.2 Generating the initial note-blocks

While consuming user-given patterns, Colorscore generates rough note-blocks, called "initial note-blocks." Colorscore generates the initial note-blocks by the following procedure:

1. Treat a track as a single block.
2. Divide a block at a whole note rest.
3. Repeat 2. for all blocks until all whole note rests are eliminated from the blocks.

\*e-mail: aki@itolab.is.ocha.ac.jp

†e-mail: itot@is.ocha.ac.jp

‡e-mail: masaki@nak.ics.keio.ac.jp

4. Repeat 2. and 3. for all tracks.

#### 2.1.3 Pattern-matching of the blocks with patterns

Then, Colorscore matches each initial note-block (see Section 2.1.2) to user-given patterns (see Section 2.1.1). In this step, Colorscore calculates distances between the patterns and each note-block, and chooses the pattern closest to the note-block. It determines that the note-block has the role which is the same as the chosen pattern, if the distance between the note-block and the pattern is smaller than the predefined threshold.

To calculate the distance between the  $i$ -th pattern and the  $j$ -th note-block, our implementation applies the following distance  $D(i, j)$ :

$$D(i, j) = w_1 D_{RA}(i, j) + w_2 D_{MA}(i, j) \quad (1)$$

Here,  $w_1$  and  $w_2$  denote constant weights,  $D_{RA}(i, j)$  is a cosine of timing which features the rhythm, and  $D_{MA}(i, j)$  is a cosine of transition of the notes which features the melody.  $D_{RA}(i, j)$  corresponds to the cosine of RA vectors between the  $i$ -th pattern and the  $j$ -th note-block. Here, a RA vector is an  $n$ -dimensional vector denoting the timing of note-on events of note-blocks or patterns.  $D_{MA}(i, j)$  corresponds to the cosine of MA vectors between the  $i$ -th pattern and the  $j$ -th note-block. Here, a MA vector is an  $(n-1)$ -dimensional vector denoting the pitch transition of note-on events of note-blocks or patterns.

If the length of a note-block is longer than that of a pattern, it may often happen that the note-block partially matches to the pattern. Considering such situation, Colorscore first extracts parts of the note-block, where the lengths of the parts are equal to the pattern, and calculate  $D(i, j)$  applying each part. If one of the parts matches to the pattern, Colorscore divides the note-blocks into two or three note-blocks, where one of them corresponds to the part matched to the pattern. Then, Colorscore applies the same pattern-matching process to the remaining note-blocks.

Colorscore applies the above-mentioned process to every note-block of every track. Here, Colorscore determines that a part of the  $j$ -th note-block matches to the  $i$ -th pattern, if the  $D(i, j)$  is smaller than a predefined threshold  $D_0$ , where  $D_0$  is a function of  $n$ . On the other hand, Colorscore treats note-blocks which do not match to any patterns as decoration note-blocks, and draws them in gray.

The procedure to assign roles to the note-blocks is as follows:

1. Calculate RA and MA vectors of the  $i$ -th pattern.
2. Calculate RA and MA vectors of the  $j$ -th note-block.
3. Extract a part of the note-block, where the length of the part is equal to the  $i$ -th pattern, and then calculate  $D(i, j)$  between the part and the  $i$ -th pattern.
4. If  $D(i, j)$  is smaller than  $D_0$ :
  - Divide the  $j$ -th note-block if necessary.
  - Assign the role of the  $i$ -th pattern to the matched note-block.
5. Repeat 3. and 4. for all possible parts of the  $j$ -th note-block.
6. Repeat 2. to 5. for all note-blocks.
7. Repeat 1. to 6. for all patterns.

## 2.2 Visualization of note-blocks

Colourscore visualizes the result of note-block generation and role determination. Figure 1(Left) shows the result of the analysis drawn on a traditional musical score, and Figure 1(Right) shows the result of visualization by Colourscore. It vertically draws the tracks, and horizontally draws the blocks in a track. It assigns colors to the note-blocks based on their roles. Our implementation assigns high-saturation colors to melodies, and low-saturation colors to accompaniments.

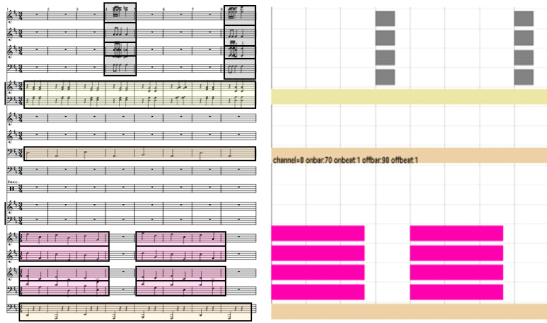


Figure 1: (Left) Result of note-block generation and role determination. (Right) Result of visualization.

## 2.3 Vertical and horizontal condensation

### 2.3.1 Vertical condensation

Vertical condensation reduces the number of tracks to be drawn by the following two steps. The first step removes decoration note-blocks, then removes tracks which have no note-blocks to be drawn, and finally reduces the tracks by moving and packing remaining note-blocks. The second step removes more note-blocks so that only the note-blocks especially similar to the given patterns remain in the visualization results. Our implementation remains only note-blocks whose  $D(i, j)$  values are smaller than a predefined threshold  $D_1$  ( $D_1 < D_0$ ). Consequently, the second step eliminates harmonic melodies, and remains main themes. Again, this step reduces the tracks by moving and packing remaining note-blocks. The above process is repeated until the tracks are reduced to the user-specified number. We think this functionality is especially useful while arranging orchestra or other large-scale music into smaller organization such as piano solo or chamber ensemble.

### 2.3.2 Horizontal condensation

Colourscore saves the display spaces based on change of roles by horizontal condensation. It shrinks bars if no note-blocks end or change their roles, while it keeps other bars longer. It shrinks bars if no note-blocks change their roles at that time. On the other hand, it keeps bars longer if new note-blocks start at that time.

## 3 RESULT

We implemented the musical structure analysis module (see Section 2.1) in GCC 3.4, and visualization and condensation module (see Section 2.2 and 2.3) in Java JDK 1.6.0. We executed it on Lenovo ThinkPad T500 with Windows XP SP3.

Figure 2 shows an example of visualization result of whole MIDI data which contains 16 tracks. Colourscore represents the musical structure in a single display space, even though a miniature score of "Valse des fleurs" we bought occupies 33 pages. Many traditional classical musical works have two themes, and forms musical structures while repeating and varying the two themes. Also, they may contain several additional melodies delivered from the themes.

Considering such composition techniques, we prepared five melody patterns to visualize the music. We also prepared typical Waltz patterns for harmonic and bass accompaniments.

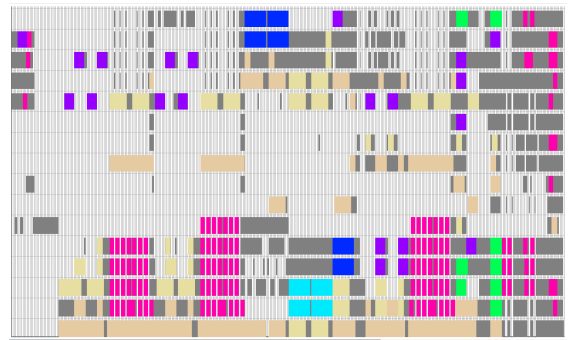


Figure 2: Visualization result of "Valse des fleurs" by Tchaikovsky

Figure 3(Left) shows a part of the visualization result shown in Figure 8, corresponding to 314 to 328 bars. Figure 3(Right) shows a result of the second step of vertical condensation. It reduces the number of tracks from 16 to 9, and note-blocks from 26 to 9. Figure 3(Left) shows that two melodies drawn in green and purple are played at the same time. We can see such orchestration techniques especially in the end (Coda) of the music. In this case, one of the melodies drawn in green is a leading melody, and the other drawn in purple is a variation and refrain of a previously played melody. The vertical condensation remained the leading melody, while removing the refrain melody. Actually, we may decide to play only the leading melody if we play the music by piano. We think this functionality is useful to arrange the music to smaller number of players.

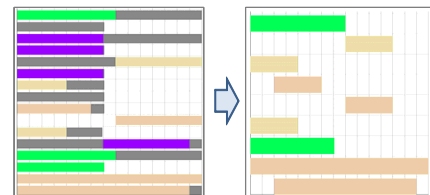


Figure 3: Vertical condensation.

Figure 4(Upper) shows a part of the visualization result in Figure 2, corresponding to 211 to 250 bars. Figure 4(Lower) shows a result of horizontal condensation. It reduces the width of the visualization space as approximately 60% of the original width. However, it does not shrink the timings when roles of note-blocks change: for example, it keeps the length of short note-blocks surrounded by circles (A) and (B) after the horizontal condensation, indicated as (A') and (B') in Figure 4(Lower). This functionality realizes to draw longer music in a display space, while preventing us to miss changes of roles from the visualization result.

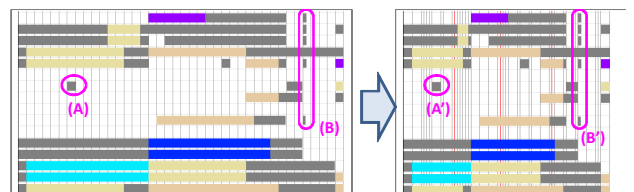


Figure 4: Horizontal condensation.