# A Directable Performance Rendering System: Itopul

Mitsuyo Hashida
School of Science & Technology
Kwansei Gakuin University
Sanda, 669-1337 JAPAN
hashida@kwansei.ac.jp

Yosuke Ito*
School of Science & Technology
Kwansei Gakuin University
Sanda, 669-1337 JAPAN

Haruhiro Katayose
School of Science & Technology
Kwansei Gakuin University
Sanda, 669-1337 JAPAN
katayose@kwansei.ac.jp

## ABSTRACT

One of the advantages of case-based systems is that they can generate expressions even if the user doesn't know how the system applies expression rules. However, the systems cannot avoid the problem of data sparseness and do not permit a user to improve the expression of a certain part of a melody directly. After discussing the functions required for user-oriented interface for performance rendering systems, this paper proposes a directable case-based performance rendering system, called Itopul. Itopul is characterized by 1) a combination of the phrasing model and the pulse model, 2) the use of a hierarchical music structure for avoiding from the data sparseness problem, 3) visualization of the processing progress, and 4) music structures directly modifiable by the user.

## Keywords

Performance Rendering, User Interface, Case-based Approach

## 1. INTRODUCTION

Music production has been increasing with the popularity of the web 2.0 services. Performance rendering is seen as a way to meet the needs of the new forms of production that can be supported with these developments. A performance rendering system expresses a certain music piece by changing its dynamics, rhythm, and tempo as a human virtuoso would play a piece expressively.

Researches that ushered in performance rendering systems date back to the 1980's [2]. Since the 1990's, approaches involving music recognition theories such as the generative theory of tonal music [7] and implication-realization model [8], learning systems, and example-based reasoning [6, 10] have been proposed. In addition, a hearing competition for system-rendered performances called *Rencon*[1] has been held since 2002 [5]. Moreover, a lot of commercial software for desktop music and digital audio workstations has been published.

Automated performance rendering systems are classified into rules-based and case-based. Many commercial music software systems adopt rules-based approach, which enable a user to generate expressions relatively easily. However, these systems have a problem in that their expression rules are limited.

Meanwhile, one of the advantages of case-based systems is that they can generate expressions even if the user doesn't know how the system applies expression rules. However, the systems cannot avoid the problem of data sparseness. Furthermore, the rule-based and case-based systems have a common problem that the design of the interface for user-oriented performance rendering.

We discuss the functions required for such a user-oriented interface and propose a directable case-based system for performance rendering.

## 2. INTERFACE OF A CASE-BASED PERFORMANCE RENDERING SYSTEM

One of the good points of a case-based performance rendering system is that it enables an inexperienced listener to give a music piece an expression with little manual operation. However, a system that automates every performance rendering function may make a listener feel that it is inconvenient. An interface that has enough directability to reflect a user's operation directly and immediately would solve these problems. To devise one, the following four supports are needed.

### 2.1 Supporting Searches of Referred Cases

A case-based performance rendering system must have a lot of pieces to use as cases to give an expression. It requires a large-scale database and a search engine. Furthermore, the pieces in the database need to be annotated to improve the usability of the system's search.

As an approach to annotation, Suzuki et al. use a variable for the playing situation in their system "Kagurame [9]". They try to deal with the data-sparseness problem by analyzing the similarities of the extracted melody candidates to the target melody and by using them as weights to transfer the features of the expression.

### 2.2 Enabling the User to Choose and Control the Rendering Procedure

The usual function of a case-based performance rendering system is automation of every rendering procedure. This causes the activity to be hidden from the user.

Some rule-based rendering systems such as SuperConductor [1], Finale[2] and jPop-E [4] enable a user to indicate the region of the score to apply rules and the values of the rule parameters. Such functions increase the variety of expressions of the target piece. To control expression, case-based systems need an interface framework to enable a user to

*Currently, Hewlett-Packard Development Company, L.P.
[1]http://www.renconmusic.org/
[2]http://www.cameo.co.jp/finale/

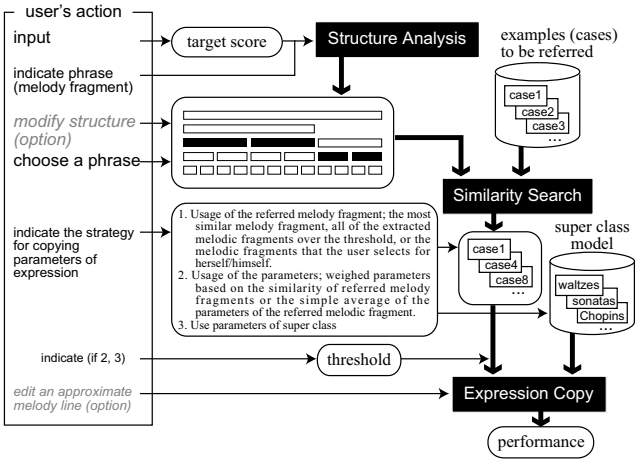


**Figure 1: Overview of Itopul**

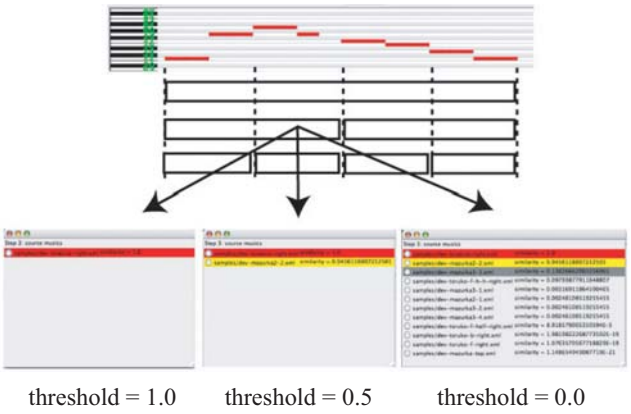


**Figure 2: Examples extracted by threshold. The user can listen to and measure the referred melody fragments on the list.**

indicate the criterion of the similarity search and the expression transfer procedures.

## 2.3 Showing the User the Situation and Progression of Each Procedure

As related above, by not showing the result of each rendering procedure, the present case-based systems don't provide enough feedback to a user. In jPop-E [4], the function to show a user the resulting expression of the target phrases and the rule parameters works in real-time. This function is useful for design support and education. The interfaces of case-based systems must also show the progress of each procedure.

## 2.4 Seamless Function to Modify the Rendered Expression

The present performance rendering systems by academic researchers use automatic rendering and do not support subsequent editing of the details of the expression. These systems have superior automatic rendering processing compared with commercial DTM systems, whereas the DTM systems comparatively have superior quality and productivity in their expressions, which a user elaborated with using them, than the automatic systems. The automatic systems should have editing functions like those of a DTM system and seamless combination of expression interfaces.

# 3. DESIGN OF DIRECTABLE CASE-BASED PERFORMANCE RENDERING SYSTEM

We developed a case-based performance rendering system *Itopul* based on the considerations stated above, as shown in Figure 1. Itopul works on Java environment.

Itopul is characterized by 1) a combination of the phrasing model and the pulse model, 2) the use of a hierarchical music structure for avoiding from the data sparseness problem, 3) visualization of the processing progress, and 4) music structures directly modifiable by the user. In this section, we describe the design of Itopul by focusing on the search support of reference cases and the choice and control of processing by the user.

## 3.1 Architecture of the Phrasing Model and the Pulse Model

The fundamental function of the Itopul is to copy the characteristics of a performance expression in existing music samples. There are two possible methods to do this; one is to copy tempo and dynamics of a sequential melody line directly, and the other is to analyze (decompose) and compose hierarchically each tempo and dynamics of the melody structure. Itopul takes the latter approach to make much of expression of each phrase. This approach requires a hierarchical description of the performance expression.

A popular method to analyze music structure hierarchically is Clynes' pulse model, based on which *SuperConductor* [1] is implemented. The pulse model is focusing on metrical structure (strong beat, weak beat) analysis. Clynes analyzed the characteristics of music expression by composer, based on the idea that characteristics appear in the balance of tempo, and the dynamics in each metrical structure. The pulse model is very useful, but it is difficult to apply phrasing expression and SuperConductor dose not support transferring the features of a certain expression.

In Itopul, the performance parameters are expressed as the shape of linear line fragments given to each phrase divided into two subphrases. This approach can deal with both the metrical structure and the phrase expression within the same framework.

## 3.2 Hierarchical Performance to Avoid Sparseness Problem and Suggesting the Rendering Process to the User

We think Itopul should as faithfully as possible copy the characteristics of the performance examples that the user indicates. However, these examples perhaps can't always apply to all melody fragments of the target score. Some examples might have less similarity, so is it appropriate to apply such rare examples to the target score? Itopul provides a function to listen to the generated performance while showing extracted performances which have more similarity than the threshold a user indicates (see Figure 2).

The user can choose the strategy for copying the parameters of expression (1. usage of the referred melody fragment; the most similar melody fragment, all of the extracted melodic fragments over the threshold, or the melodic fragments that the user selects for him/her. 2. Usage of the parameters; weighed parameters based on the similarity of referred melody fragments, or the simple average of the parameters of the referred melodic fragment). If no melody fragments are extracted for a certain threshold, no expression might be given to the melody. In this case, Itopul notifies the user that it couldn't find melodies and then asks if the use wants it to apply a general (average) expression

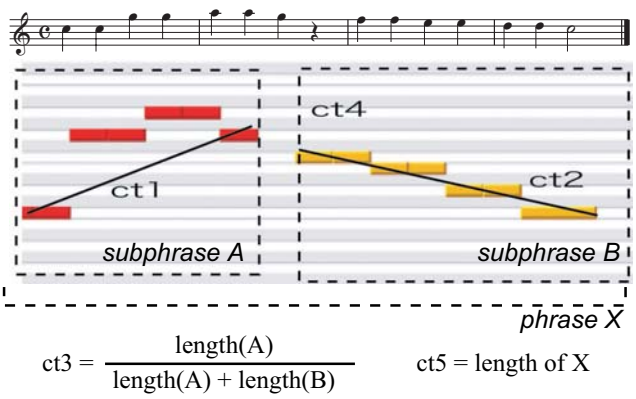


**Figure 3: Parameters of a melody fragment**

model matching the style of the target score.

## 3.3 User-oriented Hierarchical Music Structure Analysis

To copy the features of a melody fragment to the target music, we need to describe the fragment hierarchically and to consider the balance of automatic and manual procedures. In Itopul, a user first indicates the area of a melody fragment (phrase) as the user analyzes the fundamental phrase structure. Then, the system analyzes the upper and lower structure based on the phrase structure automatically. The upper structures are analyzed by using *ATTA* [3]. The lower structures are analyzed in terms of the metrical structure, as each melody forms a binary tree; long duration and rest notes are considered.

If the user isn't satisfied with the analyzed structure, Itopul provides an editing function to modify it manually.

# 4. COPYING THE PERFORMANCE EXPRESSION

Itopul decomposes a target and referred melody into multi-layered melody fragments (groups) as shown in section 3.3. Melody fragments are used as the units of the similarity calculation and also as the unit of expression feature copying. The expression of a note of the target is calculated by multiplying every parameter obtained from the extracted examples; the shape is similar to each hierarchical melody fragment that contains the note.

## 4.1 Similarity between Melody fragments

The similarity between the target and referred melody segments is calculated using the melodic contour and the surface rhythm. First, the Itopul calculates the similarity between the target and referred melody segments in the same layer. Then, the similarity is re-calculated as it considers the similarity of the lower layer.

### *4.1.1 Melodic contour*

There are five basic parameters of melodic contour ($ct_1$, ... , $ct_5$) in Figure 3. $ct_1$ and $ct_2$ are inclinations of two line segments, and $ct_3$ is the ratio of the length of the former line segment to the latter. $ct_4$ is the pitch difference between the last note of the former line segment and the first note the latter line segment. $ct_5$ is the length of the whole melody fragment. Parameters ($ct_1$, ... , $ct_4$) are automatically calculated using least squares fitting. If the user regards the boundary suggested by the automatic fitting to be unsatisfactory, he/she can manually edit the position by using the GUI.

### *4.1.2 Rhythm*

Itopul describes the surface rhythm as a vector. The vector element of the onset (the unit is the shortest note) is 1 and of the non-onset is 0. For example, the score of Figure 3 is described as $\{1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0\}$.

Itopul can evaluate the similarity of the target melody fragment and the reference melody fragment even if their lengths are not the same. When the lengths are different, elements 0 will be added to the front or end of the shorter vector until the length becomes the same as the longer one. The cosine distance is exhaustively calculated. The system chooses the largest cosine-distance obtained by this procedure as the similarity of the surface rhythms $sim_r$.

### *4.1.3 Similarity measure*

The similarity of contours $sim_{ct}$ is calculated with the following equation (the weighted cosine distance):

$$sim_{ct} = \frac{w_{ct}(u_{ct} \cdot v_{ct})^t}{|w_{ct}||u_{ct}||v_{ct}|}$$

Where, $u_{ct}$ and $v_{ct}$, denote vectors which consist of the parameters described in section 4.1.1; and $w_{ct}$ are weight vector, each component of which is multiplied with $ct_1$, ..., $ct_5$.

The similarity between the target melody fragment $u$ and the referred melody fragment $v$, $sim\_sl_{u,v}$ is calculated as follows, using $sim_r$ and $sim_{ct}$, the weight parameter $w_{cr}$ the ratio of the lengths of the target and the referred melody fragment, and its weight $w_{ld}$:

$$sim\_sl_{u,v} = (w_{cr} sim_{ct} + (1 - w_{cr}) sim_r) \left\{ \frac{\min(len(u,v))}{\max(len(u,v))} \right\}^{w_{ld}}$$

$sim\_sl_{u,v}$ reflects the similarity of the same layers only. The final similarity between the target and referred melody fragment should reflect the similarity at the lower layers. The final similarity $sim_{u,v}$ is revised using the following equation recursively from the lower layer to the higher layer.

$$sim_{u,v} = 0.5 * sim\_sl_{u,v} + 0.25 * (sim\_sl_{u_1,v_1} + sim\_sl_{u_2,v_2})$$

Here, $\{u_1, u_2\}$, and $\{v_1, v_2\}$ are lower melody fragments of $u$ and $v$, respectively.

## 4.2 Getting the Expression Parameters

Itopul employs a model to deal with the pulse model and phrasing in the same architecture. Here, we describe a procedure to extract expression parameters regarding tempo and dynamics of the melody fragment of the source example.

**Step 1** Select the melody fragment (from a higher layer).

**Step 2** Fit two line segments (A, B) to the expression data. (Get the coefficients of two line segments (A, B) as the expression parameters from the reference value given by the average value of the expression data of the melodic fragment.)

**Step 3** Replace the expression data with the residue (subtract fitting at the step2), then go to step 1 with the melody fragment (lower layer).

This procedure is a modification of the phrase fitting with a quadratic equation proposed by Widmer et al. [10].

## 4.3 Calculation of target expression

The power of each note $I(N_i)$ is calculated using the expression parameter $I_{\text{ratio}}(N_i, level)$ of note $N_i$ at layer $l$:

$$I(N_i) = \prod_l I_{\text{ratio}}(N_i, l)$$

The duration of each note is calculated as the same manner as the power calculation. The difference is only the parameters of the note the position and the time-value of which are the same between the target and the referred melody fragment, when calculating the duration.

As for the timing control, tempo is controlled by sending the local score beat-time given by the following equation every infinitesimal score time $S_k$.

$$T(T_k) = \prod_l T_{\text{ratio}}\left(T_{\tilde{k},N_i,l}\right)$$

Where $N_i$ is the note that contains $T_k$ in its control, and $T_{\text{ratio}}\left(T_{\tilde{k},N_i,l}\right)$ is the score beat-time ratio at time $T_{\tilde{k}}$ of the referred melody fragment of the target melody fragment at level $l$. $I_{\text{ratio}}(N_i, l)$, and $T_{\text{ratio}}(T_{\tilde{k}}, N_i, l)$ are calculated by referring to the user's preferences (see section 3.2).

# 5. PERFORMANCE GENERATION AND DISCUSSION

## 5.1 Performance Generation

The input files of Itopul are the target score (MusicXML format), pairs of the score (MusicXML format) and performance data (DeviationInstanceXML format[3]) of the examples (cases) to be referred.

The users of Itopul can generate expressive performances easily by using the following steps:

1. Suggest melodic fragments (boundaries) only at a certain layer of the target and the source music.
2. Adjust the threshold of the similar melody fragments search.
3. Select the strategy for copying expression parameters: (1. usage of the referred melody fragment; the most similar melody fragment, all of the extracted melodic fragments over the threshold, or the melodic fragments that the user selects for herself/himself. 2. Usage of the parameters; weighed parameters based on the similarity of referred melody fragments, or the simple average of the parameters of the referred melodic fragment)
4. Give directions on using parameters of expression of meta-class (In case similar melodic fragments are not extracted from the given source music.)

## 5.2 Discussion

### 5.2.1 User Interface

The following points are crucial for the realization of directability: 1) assistance in searching for examples, 2) users' operation and processing selection, 3) notification of the menu and processing status to the user by the GUI or sound device.

For 2), 3), we proposed a model to deal with the phrasing and pulse model on the same architecture and procedures for solving the sparseness problem. These proposals with some practical UI designs provide users with directability. As for 1), Itopul can analyze the melodic structure and obtain parametric features of melodic fragments. These functions are the bases to implement search engines on a large-scale music database. We still have to carry out a usability test.

[3]http://www.crestmuse.jp/cmx/

### 5.2.2 Musical ability of the system

Itopul allows users to generate expressions of metric structure and phrasing based on the results of a similar melodic fragment search. At present, the functions to copy expressions regarding tempo, dynamics and the duration of notes have been implemented. One of our future jobs will be to provide functions to deal with articulation of each note within chords.

The current version of Itopul is designed for monophony expressions. The expression of the accompaniment part is generated by simply copying the expression of the corresponding melody part. We should improve the system so that it can deal with polyphony. We are planning to transplant the functions from jPop-E [4] that we have been developing.

In addition, expression marks, explicitly described in scores, such as staccato or legato need to be handled.

# 6. CONCLUDING REMARKS

The main goal of the performance rendering systems that academic researchers have been developing is the realization of autonomous functions for performance generation. This paper discussed the requirements of performance rendering systems as a tool for human designers and introduced a case-based performance rendering system called Itopul. We believe that it is indispensable to evaluate musical competence as well as the usability of the performance rendering systems. We're going to bring Itopul to the ICMPC-Rencon[4] (an international event), then we will hand out this point.

# 7. REFERENCES

[1] M. Clynes. Superconductor: The global music interpretation and performance program. *http://www.superconductor.com/clynes/superc.htm*, 1998.

[2] L. Fryden and J. Sundberg. Performance rules for melodies. origin, functions, purposes. In *Proc. of ICMC*, pages 221–225, 1984.

[3] M. Hamanaka, , K. Hirata, and S. Tojo. *ATTA*: Automatic time-span tree analyzer based on extended *GTTM*. In *Proc. ISMIR*, pages 358–365, 2005.

[4] M. Hashida, N. Nagata, and H. Katayose. jpop-e: An assistant system for performance rendering of ensemble music. In *Proc. of NIME*, pages 313–316, 2007.

[5] R. Hiraga, M. Hashida, K. Hirata, H. Katayose, and K. Noike. Rencon: toward a new evaluation method for performance. In *Proc. of ICMC*, pages 357–360, 2002.

[6] O. Ishikawa, A. Aono, H. Katayose, and S. Inokuchi. Extraction of musical performance rules using a modified algorithm of multiple regression analysis. In *Proc. of ICMC*, pages 348–351, 2000.

[7] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[8] E. Narmour. *The Analysis And Cognition of Basic Melodic Structures*. the University of Chicago Press, 1977.

[9] T. Suzuki, T. Tokunaga, and H. Tanaka. A case based approach to the generation of musical expression. In *Proc. of IJCAI*, pages 642–648,, 1999.

[10] G. Widmer and A. Tobudic. Playing mozart by analogy: Learning phrase-level timing and dynamics strategies. In *Proc. of ICAD*, pages 28–35, 2002.

[4]http://www.renconmusic.org/icmpc2008/