Computational Music Theory and its Applications to Expressive Performance and Composition

Masatoshi Hamanaka[1], Keiji Hirata[2]
and Satoshi Tojo[3],

[1] 1-1-1, Tenodai, Tsukuba, Ibaraki, Japan, hamanaka@iit.tsukuba.ac.jp,
[2] 2-4, Hikaridai, Seikacho, Keihanna Science City, Kyoto, Japan, hirata@brl.ntt.co.jp,
[3] 1-1, Asahidai, Nomi, Ishikawa, Japan, Tojo@jaist.ac.jp

Abstract. This chapter describes a music analysis system based on a generative theory of tonal music (GTTM). Musical theory provides methodologies for analyzing and transcribing musical knowledge, experiences, and skills from a musician's way of thinking. Our concern is whether the concepts necessary for music analysis are sufficiently externalized in musical theory. Given its ability to formalize musical knowledge, we consider the GTTM to be the most promising theory among the many that have been proposed because it captures the aspects of musical phenomena based on the Gestalt in the music and follows relatively rigid rules. It also describes music expectation and melody morphing methods that use the analysis results from our music analysis system. The music expectation method predicts the next notes needed to assist musical novices in playing improvisations. The melody morphing method generates an intermediate melody between two melodies in a systematic order in accordance with a certain numerical measure.

Keywords: computational music theory, generative theory of tonal music (GTTM), melody expectation, melody morphing, tonal pitch space (TPS), time-span tree.

1 Introduction

Our goal is to create a system that will enable a musical novice to manipulate a piece of music, which is an ambiguous and subjective media, according to his or her intentions. We believe that it is important to make it possible to create a musical system for musical novices that can:

1) easily manipulate a piece of music, and
2) mirror the user's intentions.

Note that the higher the abstraction level of the objects for manipulating the music is, the more difficult it becomes to reflect the user's intentions. For example, it is difficult for musical novices to manipulate music with commercial music sequence software that only operates on the surface structure of music, that is, the pitch and note-on timing of each note. On the other hand, Garageband [1] can create a piece of music though simple manipulations, i.e., by just concatenating pre-stored phrases. However, when we want arrange a portion of a melody in a phrase, we have to manipulate the surface structure of the music, and a musical novice would find it difficult for the software to mirror his or her intentions in such a case.

Music theory, in particular, focuses on a piece written on a score, gives us the methodology for analyzing and understanding a piece, and explains deep structure, musical knowledge, experiences, and skills in a comprehensive way. We believe that implementing music theory on a computer would bring great benefits because it could provide a theoretical basis for developing a support system for musical activities. For instance, such advantages include performance rendering [2-4] and music summarization for practical use when displaying the results of a music information retrieval system [5].

We have developed a system of music analysis called ATTA [6, 7] and FATTA [8] based on a music theory called a generative theory of tonal music (GTTM) [9]. ATTA and FATTA can generate a time-span tree as the result of a GTTM analysis. The GTTM consistently represents multiple aspects of music in a single framework. This feature is important when the musical system is to assist a musical novice in manipulating musical structures. For instance, if we imagine a simple operation that splits a melody, the split operations may vary depending on the relevant musical structure. Therefore, it is preferred that the splitting position of the melody and that of the ornamented melody be identical. Unless a system of consistent operations in terms of melody, rhythm, and harmony is developed, the resulting splitting position may be different.

With regard to how the software should reflect the user's intentions, we take an example of composing a melody, where it is supposed that a user wants to arrange melody A by adding some musical nuances to it and he/she knows melody B has such nuances. If a user could use a system accepting a morphing command, issuing a simple command, for example, "add the nuance of melody B to melody A", can accurately convey a user's intention to a morphing system. As a result, the system generates multiple melodies approaching from melody A to B by little and little. The advantage of morphing is not only its simple and accurate transferability and manipulability but also its ease of understanding the relationship between inputs and outputs.

Previous music systems [10-12] have their own way of music analysis, from which it is difficult to acquire deeper musical structures, and thus these systems are difficult to manipulate according to the user's intention. On the other hand, Hirata [13] defined a representation method and primitive operation for polyphony, and this development indicates the potential for constructing a melody arranging algorithm.

We developed a melody morphing method in which we input monophonies A and B and then generate intermediate melodies between melodies A and B with a systematic order according to a certain numerical measure by configuring the parameters that determine the level of influence of the features of melodies A and B [14, 15]. Our system exploits melody morphing to make it possible for novices to create melodies reflecting their intensions.

Melody prediction is one of the most difficult problems in musical information retrieval because composers and players may or may not create melodies that conform to our expectation. The development of a melody expectation method is thus important for building a system that supports musical novices because melody expectation is one of the most basic skills for a musician.

Our melody prediction method helps a novice construct a melody or play an improvisation by displaying candidates for the next notes. This method is designed to be used with a "expectation piano" (Fig. 1), which was developed to assist musical novices play improvisations. On the lid of this piano, there is a 32 × 25 full-color LED matrix that displays a piano roll view that scrolls down in time with the music.

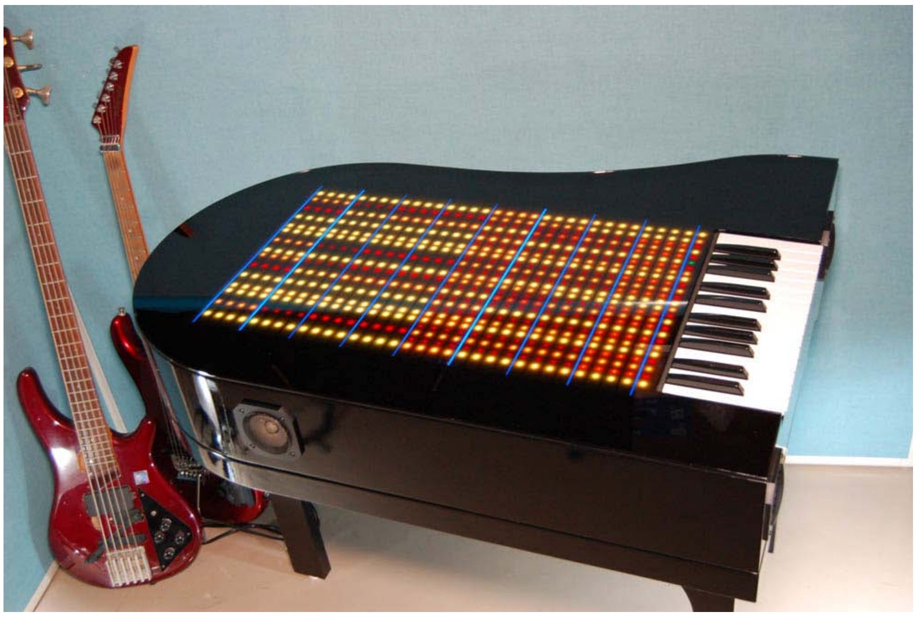


Fig.1. Expectation Piano.

We identified two key requirements for our melody expectation method to make it useful to musical novices playing an improvisation on the expectation piano.

1) Candidate notes are predicted and output even if the input melody is novel.
2) The output is appropriate from a musical point of view.

Two approaches were considered when developing this method: statistical learning and music theory. With the statistical learning approach, the predictions depend on the characteristics of the data used for learning: composer, genre, period, country, etc. [16, 17]. Moreover, predicting candidate notes for a novel melody is problematic because the system may not be able to find a similar melody in the learning data and thus may be unable to evaluate whether the notes are appropriate or not. With the music theory approach, the predictions do not depend on the characteristics of the data used for learning. It can thus be applied to novel melodies.

Unlike most previous prediction methods, which use statistical learning, our method evaluates the appropriateness of each candidate note from the view point of musical theory. Although many music theories have been proposed [18–21], GTTM is the most suitable for predicting notes in a melody because it can be used to represent the various aspects of music in a single framework. In particular, our prediction method uses the concept of melody stability based on the GTTM and the tonal pitch space (TPS) [22] to evaluate the appropriateness of the melody. It can thus predict the candidate next notes not only from the surface structure of the melody but also from the deeper structure of the melody acquired by GTTM and TPS analysis.

The organization of this chapter is as follows. In Section 2, we briefly describe music theory GTTM, discuss the problems and how to solve them when implementing GTTM, and propose exGTTM as extension of GTTM. In Section 3, we describe an interactive analyzer for GTTM, which enables seamless change of the automatic analysis process with an ATTA and the manual edit process with a GTTM manual editor. In Section 4, we present the melody expectation method that predicts the next notes is described for assisting musical novices to play improvisations. In Section 5, we explain the melody morphing method that generates an intermediate melody between a melody and another melody with a systematic order according to a certain numerical measure. In Section 6, we evaluate the performance of the implemented system. Finally in Section 7, we conclude with a summary and overview of future work.

2 Implementing GTTM on a computer

GTTM is a music theory for describing the insight of an “experienced listener,” and it consists of four sub-theories: grouping-structure analysis, metrical-structure analysis, time-span reduction, and prolongational reduction.

The grouping-structure analysis hierarchically divides a series of notes of a homophony into phrases or motives. It seems that a singer looks for breathing points when singing a long melody(Fig. 2). The metrical-structure analysis identifies strong and weak beats at each metrical level: quarter note, half note, whole note, two measures, and four measures. Essentially, it looks for timings at which a listener beats time with his/her hands to music or a conductor swings his/her baton. Time-span reduction distinguishes important parts of a melody from unimportant ones and yields a binary tree, called a time-span tree, so that each structurally important note belongs to a stem at every level (Fig 3). The prolongational reduction generates a tree structure representing subordinate relationships between chords by explicitly indicating harmonic retention and change.

Each sub-theory of GTTM is described by two kinds of rules: a well-formedness rule prescribes the conditions and constraints that must always be satisfied; a preference rule prescribes which structure is preferable among the structures satisfying the well-formedness rules. Thus, depending on the situation, some preference rules hold and others do not.

Among the four sub-theories, this section presents the methodology for implementing the grouping-structure analysis, metrical-structure analysis, and time-span reduction [6-8, 23, 24]. Since the prolongational reduction is still evolving and controversial, we do not implement it at present.

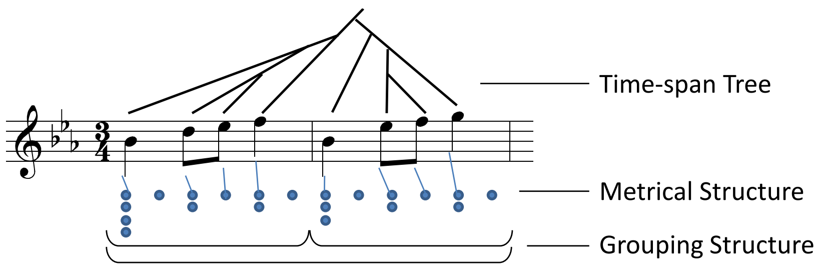


Fig. 2. Grouping structure, metrical structure, and time-span tree. Groups are graphically presented as several levels of arcs below a music staff. Strong beats are illustrated by dots in multiple levels below the music staff.

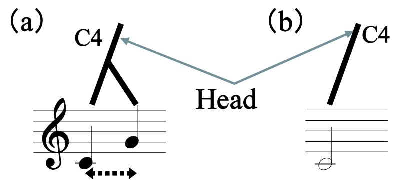


Fig. 3. Left side shows a melody and its corresponding time-span tree, where a single note, called a head (note C4 shown in right side), represents the time span denoted by <---> containing the two notes.

2.1 Problems in implementing GTTM

when we make GTTM rules operate on a computer, we have to make an ambiguous concept a firm definition and supplement the lack of concepts. Roughly speaking, these tasks correspond to parameterization and externalization, respectively. Here, we discuss the problems occurring in the process of parameterization and externalization.

Ambiguous concepts defining preference rules. GTTM uses some undefined words that can cause ambiguities in the analysis. For example, GTTM has rules for selecting structures for discovering similar melodies (called parallelism), but does not have the definition of similarity itself. To solve this problem, we attempted to formalize the criteria for deciding whether each rule is applicable or not.

Conflict between preference rules. Conflict between rules often occurs and results in ambiguities in the analysis because there is no strict order for applying the preference rules. Fig. 4 shows a simple example of a conflict between grouping preference rules (GPR). To solve this problem, we introduced adjustable parameters that enable us to control the strength of each rule.

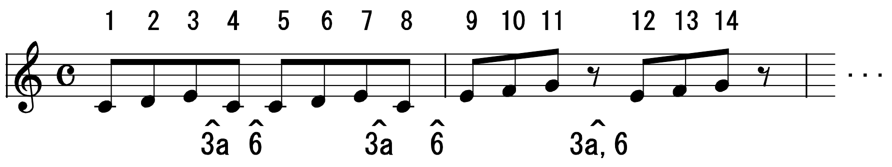


Fig. 4. GPR3a (register) is applied between notes 3 and 4 and GPR6 (parallelism) is applied between notes 4 and 5. A boundary cannot be perceived at both 3-4 and 4-5, because GPR1 (alternative form) strongly prefers that note 4, by itself, cannot form a group.

Lack of working algorithm. Knowledge represented in the rule form is in general declarative, which is advantageous in the sense that a knowledge programmer does not need to take into account an algorithm for reasoning. A system is required to perform automatic reasoning on the declaratively described knowledge. Unfortunately, GTTM has few descriptions of the reasoning and working algorithms needed to compute analysis results.

2.2 Solution: proposal of exGTTM

We extended the GTTM theory through full externalization and parameterization and devised a machine-executable extension of GTTM, exGTTM. The externalization includes introducing an algorithm for generating a hierarchical structure of the time-span tree in a mixed top-down/bottom-up manner. Such an algorithm has not previously been represented for GTTM. The parameterization includes a parameter for controlling the priorities of rules to avoid conflicts among them as well as parameters for controlling the shape of the hierarchical time-span tree. Although it has been suggested that such parameters are required in GTTM, they were not explicitly presented.

Here, we distinguish two kinds of ambiguity in music analysis: one involves the musical understanding by humans, and the other concerns the representation of a music theory. The former kind of ambiguity derives from the ambiguity of the music itself. For the latter type of ambiguity, related to GTTM, either no concept for mechanization has been presented, or it has only been presented in an implicit way. Therefore, due to the former kind of ambiguity, we assume there is more than one correct result. We avoid the latter kind of ambiguity as much as possible by performing full externalization and parameterization.

Full externalization and parameterization. The significance of full externalization and parameterization is twofold: precise controllability and coverage of the manual results. Whenever we find a correct result that exGTTM cannot generate, we introduce new parameters and give them appropriate values so that it can generate the correct result. In this way, we repeatedly externalize and introduce new parameters until we can obtain all of the results that people consider correct. In total, we introduced 15 parameters for grouping-structure analysis(Table 1), 18 for metrical-structure analysis(Table 2), and 13 for time-span reduction (Table 3).

We appropriately supply lacking parameters and make implicit parameters explicit[1]. The parameters introduced by exGTTM are categorized into identified, implied, and unaware.

A parameter in the first category is identified in GTTM but it is not assigned concrete values. Hence, we valuate such a parameter. For example, since the resulting value of the GPR2a application, D_{GPR2a}, is binary, if the rule holds, D_{GPR2a} makes 1, and 0 if it does not hold. On the other hand, since GPR6 holds indefinitely, the resulting value of GPR6, D_{GPR6}, varies continuously between 0 and 1.

[1] In the chapter, the word "parameter" is used not only for parameters used in controlling a system externally but also for internal variables (intermediated variables) connecting submodules.

A parameter of the second category is implied in GTTM. Hence, we make it explicit. For example, to resolve the preference rule conflict, we introduce parameters to express the priority for each preference rule ($S^{\mathrm{GPR\ R}}$, $S^{\mathrm{MPR\ R}}$, and $S^{\mathrm{TSRPR\ R}}$ in Table 1, 2 and 3). Since each preference rule has its own priority, all of the priority patterns are realized. This is an example of full-parameterization.

For the third category, we need to complement parameters that are not recognized in the original theory, since some of them may nearly lack any musicological meaning. For example, GPR6 in exGTTM needs to add parameters for controlling the properties of parallel segments, including the weights for pitch-oriented matching or timing-oriented matching.

We add a comment to the domain of intermediate variables, denoted as *D* and *B*. The domain of all the intermediate variables is constrained within the range of 0 to 1, and for this purpose, these variables are normalized at every computing stage. Thanks to this property, exGTTM can flexibly combine any intermediate variables (and possibly parameters) and cascade as many weighted-mean calculations as needed. This facilitates precise controllability.

Table 1. Fifteen adjustable parameters for grouping-structure analyzer.

Parameters	Description
$S_{GPR\ R}$ $(0 \leqq S_{GPR\ R} \leqq 1)$	Strength of each grouping preference rule. The larger the value is, the stronger the rule acts. $R \in \{$ 2a, 2b, 3a, 3b, 3c, 3d, 4, 5, and 6 $\}$
σ $(0 \leqq \sigma \leqq 0.1)$	Standard deviation of a Gaussian distribution, the average of which is the boundary by GPR5. The larger the value is, the wider its skirt becomes.
W_m $(0 \leqq W_m \leqq 1)$	Balance between temporal similarity of attack points and that of pitch difference in GPR6. The larger the value is, the more the system estimates the pitch difference.
W_l $(0 \leqq W_l \leqq 1)$	Weight for the length of parallel phrases. The larger the values is, the more the length of parallel phrases is prioritized in GPR6.
W_s $(0 \leqq W_s \leqq 1)$	Balance determining whether the note *i* becomes the ending note of a group or the beginning note of the following group in GPR6. The larger the value is, the more the note tends to be the ending note.
T_{GPR4} $(0 \leqq T_{GPR4} \leqq 1)$	Threshold at which the effects of GPR2,3 are considered to be salient in GPR4. The smaller the value is, the more probably GPR4 is applied.
T^{low} $(0 \leqq T^{low} \leqq 1)$	Threshold in the lower-level boundary. The smaller the value is, the more salient the boundary becomes.

Table 2. Eighteen adjustable parameters for metrical-structure analyzer.

Parameters	Description
$S_{MPR\,R}(0 \leqq S_{MPR\,R} \leqq 1)$	Strength of each metrical preference rule. The larger the value is, the stronger the rule acts. $R \in$ {1,2,3,4,5a, 5b, 5c, 5d, 5e, and 10)
W_m $(0 \leqq W_m \leqq 1)$	Balance between temporal similarity of attack points and that of pitch difference in MPR1. The larger the value is, the more the system estimates the pitch difference.
W_l $(0 \leqq W_l \leqq 1)$	Weight for the length of parallel phrases. The larger the value is, the more the length of parallel phrases is prioritized in MPR1.
W_s $(0 \leqq W_s \leqq 1)$	Balance determining whether the note *i* becomes the ending note of a group or the beginning note of the following group in MPR1. The larger the value is, the more the note tends to be the ending note.
$T_{MPR\,R}(0 \leqq T_{MPR\,R} \leqq 1)$	Value of the threshold that decides whether each rule is applicable. $R \in$ { 4, 5a, 5b, 5c }

Table 3. Thirteen adjustable parameters for time-span tree analyzer.

Parameters	Description
$S_{TSRPR\,R}(0 \leqq S_{TSRPR\,R} \leqq 1)$	Strength of each rule. The larger the value is, the stronger the rule acts. $R \in$ {1, 2, 3, 4, 5a, 5b,5c, 5d, 5e, and 10}
W_m $(0 \leqq W_m \leqq 1)$	The balance between the temporal similarity of attack points and that of the pitch difference in TSRPR4. The larger the value is, the more the system estimates the pitch difference.
W_l $(0 \leqq W_l \leqq 1)$	The weight for the length of parallel phrases. The larger the values is, the more the length of parallel phrases is estimated in TSRPR4.
W_s $(0 \leqq W_s \leqq 1)$	The balance determines whether the note i becomes the ending note of a group or the beginning note of the following group in TSRPR4. The larger the value is, the more the note tends to be the ending note.

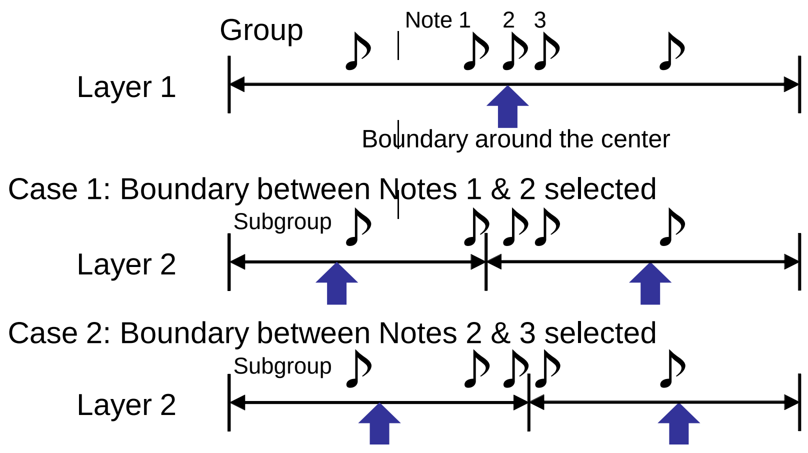

Group
Note 1
2
3
Layer 1
Boundary around the center
Case 1: Boundary between Notes 1 & 2 selected
Subgroup
Layer 2
Case 2: Boundary between Notes 2 & 3 selected
Subgroup
Layer 2

Fig. 5. In computing GPR5, the determined boundary position influences the identifications of remote boundaries in lower layers, and we have to take into account up-to-date global information every time. That is, a global constraint is inevitably dynamic.

2.3 FATTA: Fully Automatic Time-span Tree Analyzer

We implemented a time-span tree analyzer, called automatic time-span analyzer (ATTA), based on exGTTM. Although ATTA can automatically acquire a time-span tree, because the parameters are manually controlled, it takes too much time to find a set of optimal parameters. Therefore, we developed a method for automatically estimating the optimal parameters [8].

Two rules in GTTM [9] are not implemented in ATTA: GPR7 and TSRPR5.

GPR7 (time-span and prolongational stability): prefer a grouping structure that results in a more stable time-span and/or prolongational reductions.

TSRPR5 (metrical stability) In choosing the head of time-span *T*, prefer a choice that results in a more stable choice of metrical structure.

These rules require that information from later processes, such as time-span/prolongational reductions, be sent back to earlier processes.

To estimate the optimal parameters automatically, we evaluate the structural stability of the analysis results derived by ATTA. We use GPR7 and TSRPR5 to calculate the level of stability. Fig. 6 shows the process flow of the FATTA, which consist of the ATTA and a loop by the GPR7 and TSRPR5.

Implementation of GPR7 with Tonal Pitch Space. GPR7 is applied to the loop between the time-span/prolongational reduction and grouping structure analysis. This rule leads to a preference for a grouping structure that results in a more stable time-span and/or prolongational reductions. The holding level of GPR7, which varies continuously between 0 and 1, is defined as

$$D_{GPR7} = \frac{\sum_i \text{distance}(p(i), s(i)) \times \text{size}(i)^2}{\sum_i \text{size}(i)^2}. \quad (1)$$

where *i* indicates the head of the time-span, which has primary and secondary branches, denoted by *p*(*i*) and *s*(*i*), respectively. Distance (x, y) indicates the distance between notes x and y in the tonality of the piece, which are defined using Lerdahl's tonal pitch space [2]. We normalized the distance from 0 to 1. The size(*i*) indicates the length of the time-span with head i. When calculating D_{GPR7}, we use the square of size(*i*) for the weighting for empirical reasons.

In the tonal pitch space, the distance between chord x = $C_1/\mathbf{R_1}$ and chord y = $C_2/\mathbf{R_2}$ is defined as follows:

$$\delta(x \to y) = i + j + k\ ,. \quad (2)$$

where *i* is region distance, *j* is chord distance, and *k* is basic space difference. The region distance is the smallest number of steps along the regional circle of fifth between $\mathbf{R_1}$ and $\mathbf{R_2}$. The chord distance is the smallest number of steps along the chordal circle of fifth between the roots of C_1 and C_2 within each region. The basic space distance is a specially weighted to define each chord and region. Note that pitch class only has a meaning in terms the elements of the sets that define chords and regions, and chords are always understood as functioning within some region.

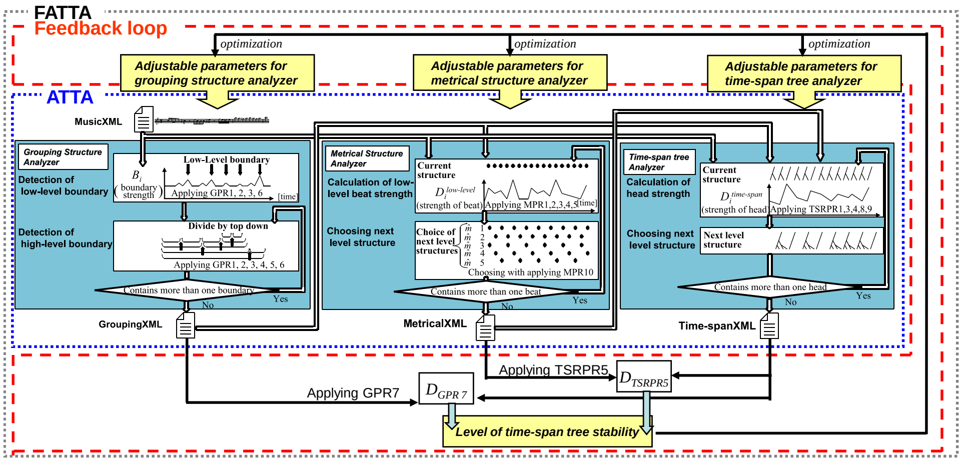
Implementation of TSRPR5. TSRPR5 is applied to the loop between the time-span reduction and metrical structure analyzer and results in a more stable metrical structure when choosing the head of a time-span. The holding level of TSRPR5, which varies continuously between 0 and 1, is defined as

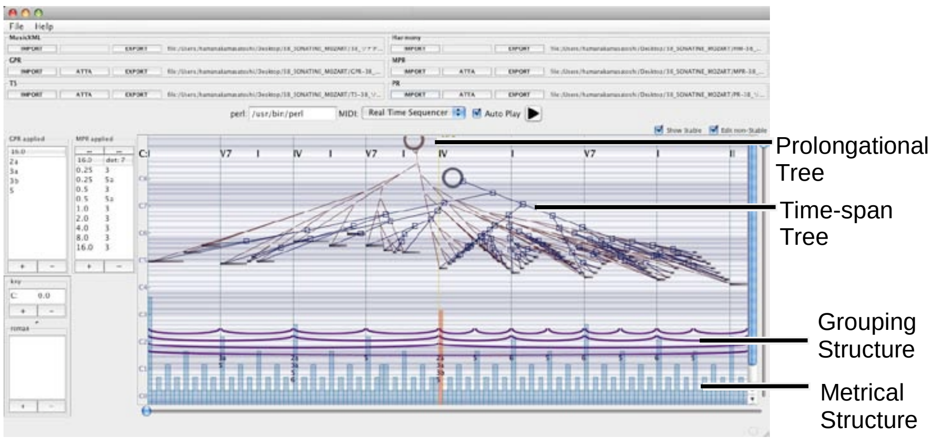
$$D_{TSRPR5} = \frac{1}{\sum_i \mathrm{size}(i)^2} \times \sum_i \begin{cases} \mathrm{size}(i)^2 & dot(p(i)) \geq dot(s(i)) \\ 0 & dot(p(i)) < dot(s(i)) \end{cases}, \qquad (\mathbf{3})$$

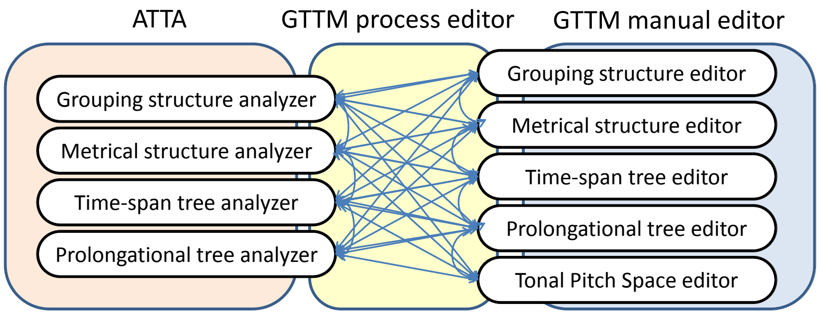
where $dot(x)$ indicates the number of metrical dots of note x.

Optimization of adjustable parameters. The set of optimal ATTA parameters is obtained by maximizing the average of D_{GPR7} ($0 \leq D_{\mathrm{GPR7}} \leq 1$) and D_{TSRPR5} ($0 \leq D_{\mathrm{TSRPR5}} \leq 1$). The parameters and default values are $S_{\mathrm{rules}} = 0.5$, $T_{\mathrm{rules}} = 0.5$, $Ws = 0.5$, $Wr = 0.5$, $Wl = 0.5$, and $\sigma = 0.05$. Because there are 46 parameters, a great amount of time is needed to calculate all parameter combinations. To minimize the calculation time, we constructed an algorithm that does the following:

(1) Maximize average of D_{GPR7} and D_{TSRPR5} by changing a parameter from its minimum to its maximum value.
(2) Repeat (1) for all parameters.
(3) Iterate (1) and (2) as long as the average of D_{GPR7} and D_{TSRPR5} is higher than after the previous iteration.


FATTA
Feedback loop
optimization
Adjustable parameters for grouping structure analyzer
optimization
Adjustable parameters for metrical structure analyzer
optimization
Adjustable parameters for time-span tree analyzer
ATTA
MusicXML
Grouping Structure Analyzer
Detection of low-level boundary
Low-Level boundary
B_i (boundary strength)
Applying GPR1, 2, 3, 6
[time]
Detection of high-level boundary
Divide by top down
Applying GPR1, 2, 3, 4, 5, 6
Contains more than one boundary
Yes
No
GroupingXML
Metrical Structure Analyzer
Calculation of low-level beat strength
Current structure
$D_i^{low\text{-}level}$ (strength of beat)
Applying MPR1,2,3,4,5
[time]
Choosing next level structure
Choice of next level structures
$\hat{m}$ 1
$\hat{m}$ 2
$\hat{m}$ 3
$\hat{m}$ 4
$\hat{m}$ 5
Choosing with applying MPR10
Contains more than one beat
Yes
No
MetricalXML
Time-span tree Analyzer
Calculation of head strength
Current structure
$D_i^{time\text{-}span}$ (strength of head)
Applying TSRPR1,3,4,8,9
Choosing next level structure
Next level structure
Contains more than one head
Yes
No
Time-spanXML
Applying GPR7
D_{GPR7}
Applying TSRPR5
D_{TSRPR5}
Level of time-span tree stability
Fig. 6. Processing flow of fully automatic time-span tree analyzer (FATTA).




ATTA
GTTM process editor
GTTM manual editor
Grouping structure analyzer
Metrical structure analyzer
Time-span tree analyzer
Prolongational tree analyzer
Grouping structure editor
Metrical structure editor
Time-span tree editor
Prolongational tree editor
Tonal Pitch Space editor
Fig. 8. Overview of interactive GTTM analyzer.

The GTTM manual editor consists of grouping, metrical, time-span, prolongational, and Tonal Pitch Space editors. The Tonal Pitch Space [22] is a music theory for chord progression composed by Lerdhal, who is one of the authors of the GTTM. Although the GTTM includes rules that require the analysis results of chord progression, the ATTA uses such rules by adopting the results of the Tonal Pitch Space.

The analyzing process with the ATTA and GTTM manual editor is complicated, and sometimes a user is confused as to what he or she should do in the next process, as there are three analyzing processes in the ATTA and five editing processes in the GTTM manual editor. A user may iterate the ATTA and manual edit processes multiple times. To solve this problem, we propose a GTTM process editor, which presents candidates for the next process of analysis, and a user only needs to change the process, just by selecting the next process.

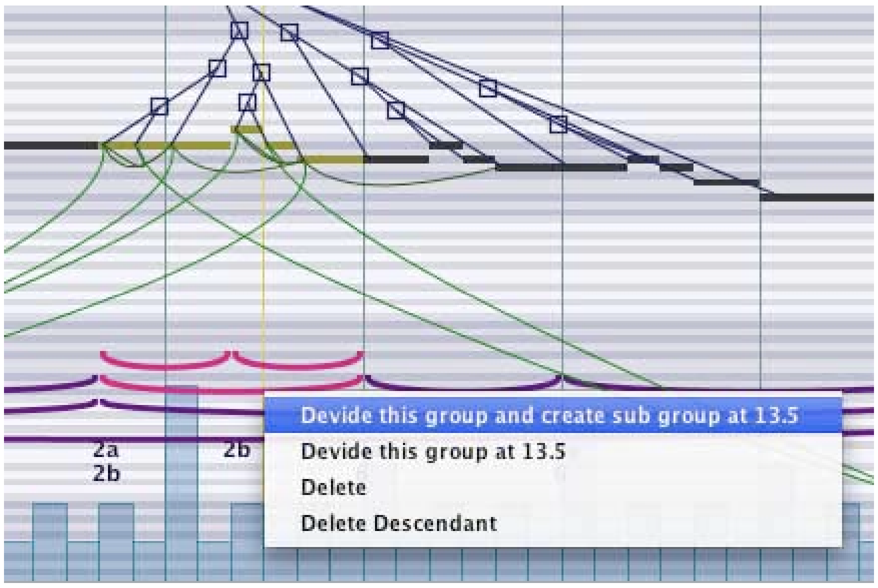
We use an XML format for all the input and output data structures of our interactive GTTM analyzer. Each analyzer and editor of our analyzer works independently, but they are integrated with the XML-based data structure.

3.1 GTTM Manual Editor

In some cases, the ATTA may produce a preferable result, which reflects the user's interpretation, but in other case it may not. When a user wants to change the analysis result according to his or her interpretation, he or she can use the GTTM manual editor. We describe the method for editing and constructing a musical structure of the GTTM using the GTTM manual editor.

Grouping structure editor. Fig.9 is a screenshot of our interactive GTTM analyzer in editing a grouping structure. The color of the target group and all its subgroups turn red after selection with a mouse. Then we can open a popup menu by right clicking the mouse. There are four operations in the popup menu, divide this group and create subgroup, divide this group, delete, and delete descendant.

To change a position of a grouping boundary, a user first delete the groups which adjoin the boundary then divide the upper level (global level) group and create new subgroups where he or she wants to create a boundary. By left clicking a grouping boundary, the user sees the rules that are applied to the boundary and he or she can add or delete these rules.



2a
2b
2b
Devide this group and create sub group at 13.5
Devide this group at 13.5
Delete
Delete Descendant

Fig. 9. Screenshot of grouping structure editor.

Metrical structure editor. Although the metrical structure analyzer in the ATTA performs fairly well [24], a user may want to slightly edit the metrical structure. In which case, he or she applies the metrical structure editor, and changes the strength level of a beat by dragging a bar up or down. At the same time, he or she sees the rules that are applied to the bar and can add or delete these rules.

While editing beat strength, a user may break hierarchical metrical structures. In other words, the results of the metrical structure editor sometimes do not hold for the metrical preference rules. This problem can be solved using the GTTM process editor, which we discuss in 3.2.

Time-span tree editor. In the time-span tree, each branch has a head represented by a square in the time-span tree editor, and a user can move the head by dragging another branch. Fig. 10 is a screenshot of dragging a head. The light blue branch is the former position, and the dark blue branch is the latter position. A user can select a type for each head by opening the popup menu among those four types, ordinary, fusion, transformation, and cadential retention.

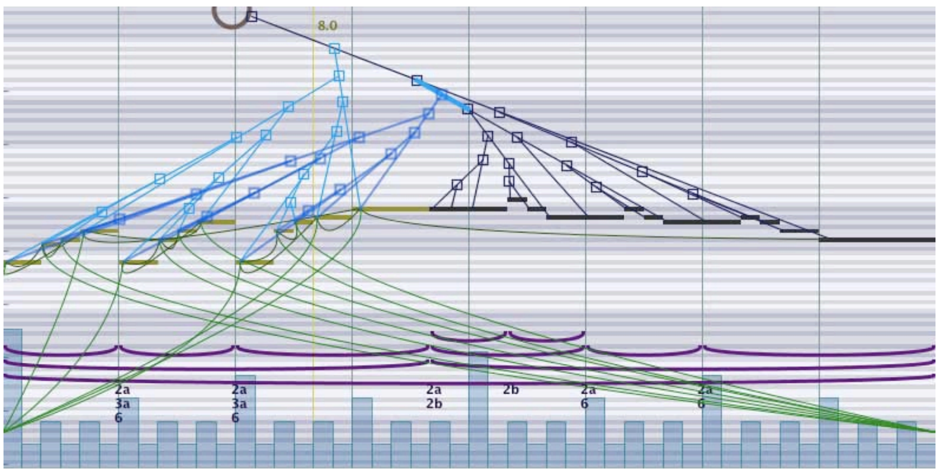


Fig. 10. Screenshot of when dragging head.

Prolongational tree editor. The process of the prolongational tree editor is the same as that for the time-span tree. The prolongational tree is constructed by reconnecting the heads based on the time-span tree. There are head connection constraints of the prolongational tree. When a head connection of a prolongational tree is ill -formed, the GTTM process editor automatically opens the popup menu and displays candidates for a solution.

Tonal Pitch Space editor. The reason we include a Tonal Pitch Space editor in our interactive GTTM analyzer is that the editor provides quantitative grounds for the prolongational tree to be hierarchical. Therefore, analyzing the Tonal Pitch Space with the prolongational tree improves analyzing performance.

3.2 GTTM Process Editor

In the GTTM, the analysis sequence proceeds from the grouping structure, secondly to the metrical structure, next to the time-span tree, and finally to the prolongational tree. However, the GTTM contains feedback links from higher- to lower-level structures. For example, grouping preference rule 7 (GPR7) (time-span and prolongational stability) prefers a grouping structure that results in a more stable time-span and/or prolongation reduction. Therefore, to analyze with feedback link rules, we need to perform several analyzing processes by trial and error. The GTTM process editor helps in this repetition by performing three functions, data inputting, history recording, and process controlling.

Data inputting. Data inputting helps with the input of analysis results, which are prepared by another user or analyzer. For example, we do not have an automatic analyzer for the Tonal Pitch Space [22] in our interactive GTTM analyzer; however, attempts have been made to implement the Tonal Pitch Space, so we can use those results [25].

We can add new rules to the ATTA using data inputting. For example, grouping preference rule 6 (GPR6) is a rule for parallelism in a grouping structure; however, the GTTM does not define the decision criteria for construing whether two or more segments are parallel or not. Therefore, many implementations of GPR6 would be possible, although we propose only one of them. By adding a new rule to the ATTA, we can control a new adjustable parameter for the new rule, GPR6+, which is the new implementation of GPR6.

History recording. History recording records the operation of analysis, and a user can return to the previous phase of analysis. History recording enables the copying and pasting of several operations of analysis while editing parallel phrases.

In the GTTM, there are few descriptions of the reasoning and working algorithms needed to compute the analysis results, especially for the time-span and prolongational trees. By using history recording, we look forward to storing the analysis knowledge, which improves automatic analysis.

Process controlling. Process controlling enables seamless change of the analysis process by using the ATTA and the manual edit process by using the GTTM manual editor, representing candidates for the next process of analysis. The representation method differs depending on the number of candidates for the next process.

When there is only one candidate process, the process-controlling function automatically executes the process. For example, when a user edits the strongest beat in Fig. 11a in the 2nd level, the hierarchical metrical structure is broken because in level 3 of Fig. 11b there are three weak continuous beats, and the metrical well-formedness rule 2 (MWFW2) does not hold. MWFR2 requires that strong beats are spaced either two of three beats apart at each metrical level. The process editor automatically alternately produces strong and weak beats in level 3 (Fig. 11c). If there is a higher metrical structure than level 3, the metrical analyzer of the ATTA automatically analyzes after level 3 and constructs a hierarchical metrical structure reflecting the user's intention.

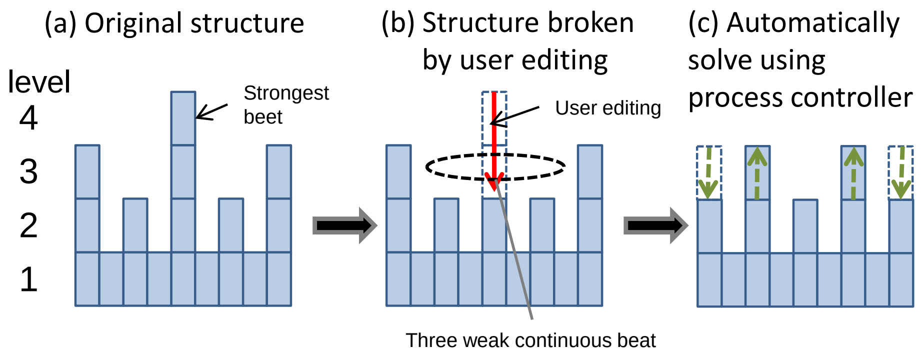

(a) Original structure
(b) Structure broken by user editing
(c) Automatically solve using process controller
level
4
3
2
1
Strongest beet
User editing
Three weak continuous beat

Fig. 11. Automatically correct broken metrical structure.

When there are a few candidates, the process controlling function automatically opens the popup menu and shows the candidates. For example, if there is a grouping structure, as shown Fig. 11a, and a user deletes a group at the upper left (Fig. 12b), the grouping structure of Fig. 12b is broken since grouping well-formedness rule 3 (GWFR3) does not hold. GWFR3 requires constraints that a group may contain smaller groups. To solve this problem, there are only two processes:

1) Delete all the groups at the same level of the deleted group (Fig. 12c).
2) Extend the grouping boundary of the left end of the right group of the deleted group to the left end of that deleted group (Fig. 12d).

The next process can be executed by selecting one of the two processes displayed in the popup menu.

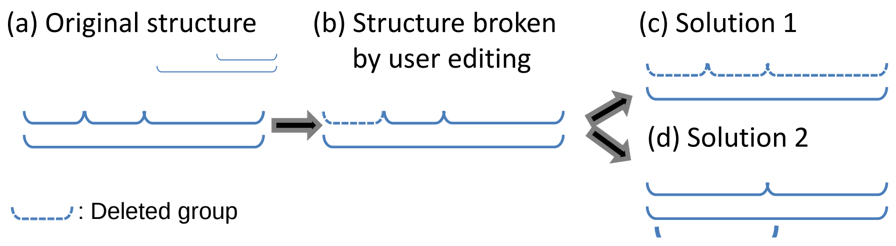


Fig. 12 Two types of solutions for broken grouping structure.

When there are many candidates, the process-controlling function selects and shows the top-ten candidates from the history recording. The candidates are ordered depending on the similarity of the history. For example, after editing a time-span tree with the time-span tree editor, executing a grouping analyzer or metrical analyzer in the ATTA is ranked in the upper levels because there are rules for feedback link such as GPR7 or metrical preference rule 9 (MPR9). GPR7 (time-span and prolongational stability) is a link from the time-span and prolongational trees to the grouping structure, and MPR9 (time-span interaction) is a link from the time-span tree to the metrical structure.

We have not implemented the original ATTA on GPR7 and MPR9. In this section, we omit the details of the implementation of these rules due to space limitations.

4 Melody Expectation

Our melody expectation method predicts candidate notes by using the level of stability of the time-span tree defined in FATTA. Our position is that we cannot always specify a single expected, following tone; and thus, we developed an expectation piano that simply suggests multiple candidates among stable pitch events with higher stability. The functions of FATTA are restricted as follows; FATTA only treats monophonic western tonal music. Thus, our expectation method can predict only monophonic musical structures of western tonal music.

4.1 Melody Expectation Method

The main advantage of our melody expectation method is that, the stability of a melody is calculated by analyzing the whole melody from the beginning note to the expected note, not from only the local melody(a few notes previous to a relevant note) ; previous melody expectation methods based on music theory (eg. Steve Larson's theory of musical forces [21]) derive the expected note from the local melody. Music tends to be more interesting when it does not match the listener's expectations, such as a delayed note, and this may result in tension and relaxation. A composer can deliberately construct such music, which can make it difficult to predict the next notes in the melody with accuracy. For example, an ornamentation note is often inserted before the expected note. In such cases, our method can predict candidate notes fairly well because FATTA can evaluate the stability of the entire structure of the time-span trees, which includes branches connected to essential notes and leaves connected to ornament notes.

Real-time Extension for FATTA. To be able to predict notes on the basis of GTTM, FATTA must run in real time. However, FATTA needs several minutes to finish the analysis, so running in real time is difficult. Therefore, we extended the algorithm to enable real-time operation. First, to speed up the iteration described in 2.2, we use the set of optimal parameter values for the last melody, which is one note shorter than the present melody, as the initial parameter set. Second, to reduce the time used by ATTA, we introduce a window for analysis. The size of the window is the longest group length within 16 measures of the present position. This length can be acquired through preprocessing using the grouping structure analyzer in ATTA. If there is no grouping boundary in 16 measures from the present position, we use 16 measures as the window size.

Calculation Level of Stability of Melodies by FATTA. Our method evaluates the appropriateness of each candidate melody to occur after the present one by calculating its stability. We use the average of D_{GPR7} and D_{TSRPR5} as the level of stability acquired by FATTA.

Fig. 13 shows the calculated level of stability from the primary note to the present note. The level of stability can be calculated after the third note because GTTM analysis needs at least four notes.

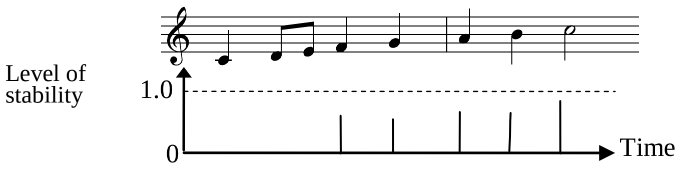


Fig. 13. Level of stability over time.

In the score, the primary note is a tonic of the region, and the tail note is also a tonic of the region. This means that the levels of stability indicate a large value at the tail of the melody and a smaller value in the middle of the melody. Therefore, the higher the level of stability, the relatively closer the tonic of the region. The region of the melody and chord progression were estimated in the implementation of GPR7 by applying the tonal pitch space.

4.2 Expectation Piano

Our expectation piano assists novices with musical improvisation by displaying the predicted notes on the piano lid. When the novice finds it hard to continue playing the melody, she/he can continue the improvisation by playing a note displayed on the lid, without impairing tonality.

Overview. The processing flow of the prediction piano is as follows. First, the MIDI signals for the music played on the piano are sent to a computer. The signals are quantized, and a MusicXML version of the performed melody is created. A learning-based quantization method [26] is used to eliminate the deviation in onset times, which are then aligned to the normalized positions. The predicted notes are acquired by inputting the MusicXML into FATTA. Finally, the predicted notes are displayed on the piano lid.

LED Scrolling Piano Roll. Our method evaluates the appropriateness of each candidate melody to occur after the present one by calculating its stability. We use the average of D_{GPR7} and D_{TSRPR5} as the level of stability acquired by FATTA. Fig. 13 shows the calculated level of stability from the primary note to the present note. The level of stability can be calculated after the third note because GTTM analysis needs at least four notes.

Calculation Level of Stability of Melodies by FATTA. The predicted notes are displayed in piano roll format within the range of view of the keyboard. The roll scrolls down at a constant speed. Below the piano lid, which is made of semitransparent acrylic resin, there is a 32 × 25 full-color LED matrix for displaying the scrolling piano roll. The 32 represents two measures when the resolution is a sixteenth note, and 25 is the number of keys on the keyboard. The color of each LED in the matrix is determined under the assumption that the onset of the next note will start at the corresponding position on the piano roll and by calculating the level of stability. When the level of stability is high, the LEDs show yellow, when it is low, they show black, and when it is neither, they show red. There is also a 32 × 20 blue LED matrix that displays the bar lines of the piano roll.

Construction. The piano is 953 mm long and 710 mm wide and resembles a grand piano. It contains a MIDI keyboard, the LED display, a microcomputer, a power supply, and four speakers. The LED display is 630 mm long and 390 mm wide. The colors of the LEDs are controlled by MAX6972 which is 16-output 12-bit pulse-width-modulation (PWM) LED drivers. There is a 5-mm gap between the LEDs and piano lid to protect the acrylic resin lid from the heat of the LEDs. A half-mirror sheet is bonded to the back side of the acrylic resin so that the lights of the LEDs show on the surface of the piano lid rather than 5 mm below it. The LED drivers are controlled using a microcomputer connected to the computer with a network cable. The computer sends the data for the LED colors by using the user datagram protocol.

5 Melody Morphing

Initial melody A, target nuance melody B, morphing result melody C , and the morphing method must meet the following conditions. 1 and 2 are conditions for melody C, 3 and 4 are conditions for the method.

1) The similarity between A and C is closer than that of A and B and the similarity between B and C is closer than that of A and B.
2) When B is the same as A, C will be the same as A.
3) The output of multiple melodies C depend on the parameters that decide the level of influence of the features of melodies A and B.
4) C will be a monophony if A and B are monophonies.

5.1 Using GTTM for Melody Morphing

Our melody morphing method uses time-span trees acquired by analysing the results of the GTTM. Fig. 14 is an example of abstracting a melody by using a time-span tree. In the figure, there is a time-span tree from melody D, which embodies the results of the GTTM analyses. In the time-span tree, the important notes are connected to a branch nearer the root of the tree. In contrast, the un-important notes are connected to the leaves of the tree. We can obtain an abstracted melody E by slicing the tree in the middle and omitting notes that are connected to branches under line E. In the same manner, if we slice the tree higher up at line F, we can get a more abstracted melody F. We can regard the abstraction of melody as a kind of melody morphing, because melody E is an intermediate melody between melody E and melody F.

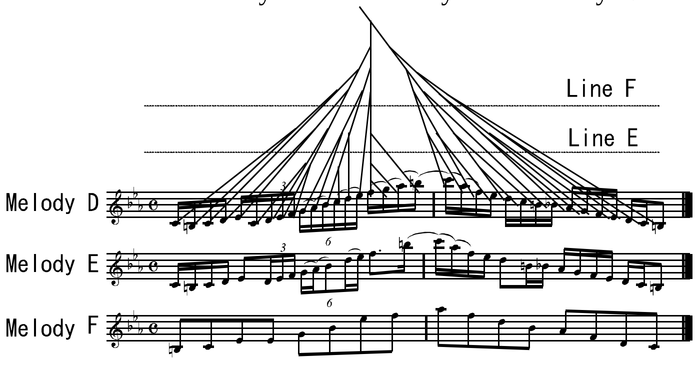


Fig. 14. Abstraction of melody.

In order to realize the melody morphing, we use the primitive operations of the subsumption relation (written as $\sqsubseteq$), meet (written as $\sqcap$) and join (written as $\sqcup$), as proposed in Hirata [13]. The subsumption relation represents the relation "an instantiated object" $\sqsubseteq$ "an abstract object" (Figure 15a). For example, the relationship among TD, TE and TF, which are the time-span trees (or reducted time-span trees) of melodies D, E and F in Figure 2, can be represented as follows:

$$TF \sqsubseteq TE \sqsubseteq TD \quad (\mathbf{4})$$

The meet operator extracts the largest common part or the most common information of the time-span trees of two melodies in a top-down manner (Figure 15b). The join operator joins two time-span trees in the top-down manner as long as the structures of two time-span trees are consistent (Figure 15c).

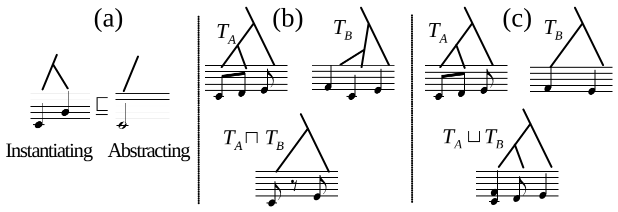


Fig. 15. Examples of subsumption $\sqsubseteq$ meet $\sqcap$ and join $\sqcup$.

5.2 Overview of Melody Morphing Method

The meaning of morphing is to change one image into another through a seamless transition. For example, a morphing method for a face picture can create intermediate pictures through the following operations.

1) Link characteristic points such as on the eyes, nose, etc, in the two pictures (Fig. 16a).
2) Rate the intensities of shape (position), colour, etc, in each picture.
3) Combine the pictures.

Similarly, our melody morphing method creates intermediate melodies with the following operations.

1) Link the most common information of the time-span trees of two melodies (Fig. 16b).
2) Abstract the notes of a melody in the difference branch of the time-span tree by using the melody divisional reduction method.
3) Combine both melodies.

The melody morphing method is illustrated in Fig. 17.

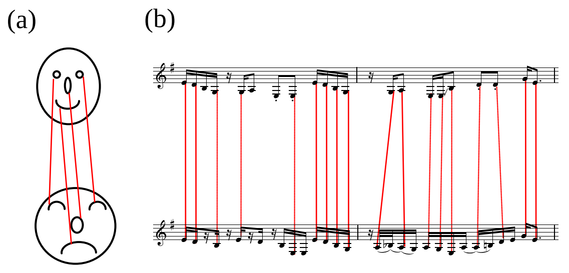


Fig. 16. Examples of linking two pictures/melodies.

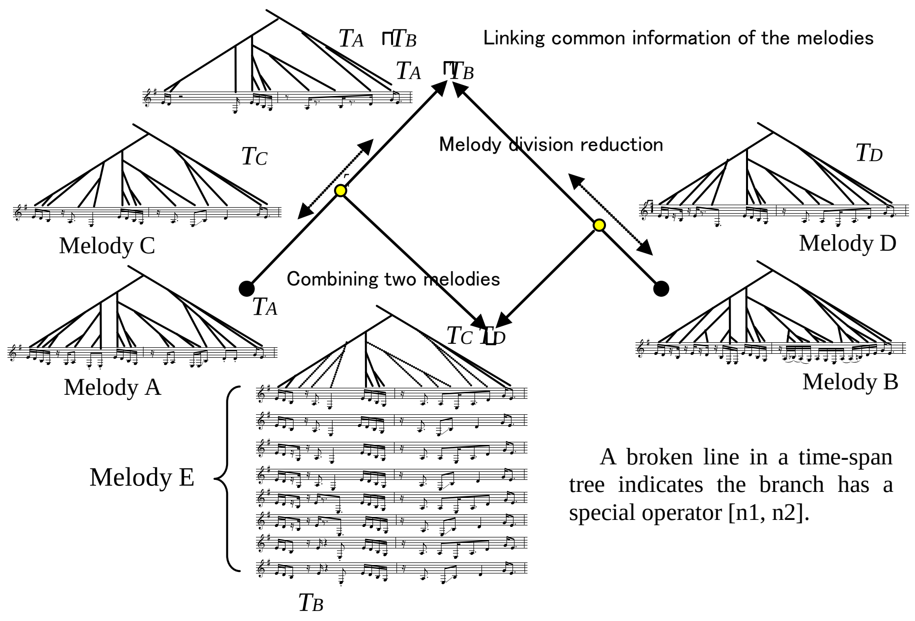

$T_A \sqcap T_B$
Linking common information of the melodies
$T_A \sqcap T_B$
T_C
Melody division reduction
T_D
Melody C
Melody D
Combining two melodies
T_A
$T_C \sqcup T_D$
Melody A
Melody B
Melody E
A broken line in a time-span tree indicates the branch has a special operator [n1, n2].
T_B

Fig. 17. Overview of melody morphing method.

5.3 Linking Common Information of the Melodies

By using the time-span trees T_A and T_B from melodies A and B, we can calculate the most common information $T_A \sqcap T_B$ which is not only the essential parts of melody A but also those of melody B. The meet operation $T_A \sqcap T_B$ are abstracted from T_A and T_B, and those discarded notes are regarded to be the *difference* information of T_A and T_B.

We use FATTA [8] to generate a time-span tree from a score automatically. We restrict the music structure to monophony, because FATTA only allows monophony input. From now on, we use a word 'melody' only as it pertains to monophony.

When calculating $T_A \sqcap T_B$ by extracting the largest common part of T_A and T_B in a top-down manner, the result may change depending on whether or not the octave notes such as C4 and C3 can be distinguished. If we discriminate octave notes, C4 $\sqcap$ C3 will be empty $\perp$. On the other hand, if we do not discriminate octave notes, the result is just C, which abstracts the octave information. We regard a note and the octave note to be different notes, because processing is difficult if the octave information is not defined.

5.4 Melody Division Reduction

We consider that there are features without the other melody in the difference information of T_A and T_B. Therefore, we need a method for smoothly increasing or decreasing the features. The melody divisional reduction method abstracts the notes of the melody in the difference branch of the time-span tree by applying the abstraction described in Section 5.1.

In the melody divisional reduction method, we can acquire melodies *Cm* (*m*=1,2,…,n) from T_A and $T_A \sqcap T_B$, by following algorithm. The subscript *m* of *Cm* indicates the number of notes in the difference information of the time-span trees that are included in T_{Cm} and not included in $T_A \sqcap T_B$.

Step 1: Decide the level of abstraction
A user decides the parameter *L* that determines the level of abstraction of the melody. *L* is from 1 to the number of notes in the difference information of the time-span trees that are included in T_A but not included in $T_A \sqcap T_B$.

Step 2: Abstraction of notes in the difference information
This step selects and abstracts a note which has the fewest number of dots in the difference information. The numbers of dots can be acquired from the GTTM analysis results [3]. If two or more notes have the fewest dots, we select the first one.

Step 3: Iteration
Iterate step 2 *L* times.

Subsumption relations hold as follows for the time-span trees T_{Cm} constructed with the above algorithm.

$$T_A \sqcap T_B \sqsubsetneq T_{Cn} \sqsubsetneq T_{Cn-1} \sqsubsetneq \ldots \sqsubsetneq T_{C2} \sqsubsetneq T_{C1} \sqsubsetneq T_A \tag{5}$$

In Fig. 17, there are 9 notes included in T_A but not included in $T_A \sqcap T_B$. Therefore, the value of n is 8, and we can acquire eight kinds of melodies *Cm* (*m*=1,2,…,n) between T_A and $T_A \sqcap T_B$. Hence, melody *Cm* is attenuates features that only have melody A without melody B (Fig. 18).

In the same way, we can acquire melody D from T_B and $T_A \sqcap T_B$ as follows.

$$T_A \sqcap T_B \sqsubsetneq T_D \sqsubsetneq T_B \tag{6}$$

5.5 Combine Two Melodies

We use the join operator to combine melodies C and D, which are results of the divisional reduction using time-span tree of melodies A and B. The simple join operator is not sufficient for combining T_C and T_D, because $T_C \sqcup T_D$ is not always a monophony if T_C and T_D are monophonies. In other words, the result of the operation has chords when the time-span structures are overrides and the pitches of the notes are different; therefore, the result violates condition 4 in section 3.

In order to solve this problem, we introduce a special operator [n1, n2], which indicates note n1 or note n2, as a result of n1 $\sqcup$ n2. Then, the result of $T_C \sqcup T_D$ is all combinations of monophonies made by the operators.

5.5 ShakeGuitar

The ShakeGuitar is demonstration system of melody morphing method for use with iPhone/iTouch that enables users to enjoy the simulated experience of guitar playing, even if they are musical novices (Fig. 19). Shaking the iPhone/iTouch with varying degrees of strength influences the level of guitar-melody difficulty, which smoothly changes in real-time from soft backing to heavy soloing by using melody morphing method. Users can identify the difficulty level visually as the color of the guitar body changes corresponding to the level.

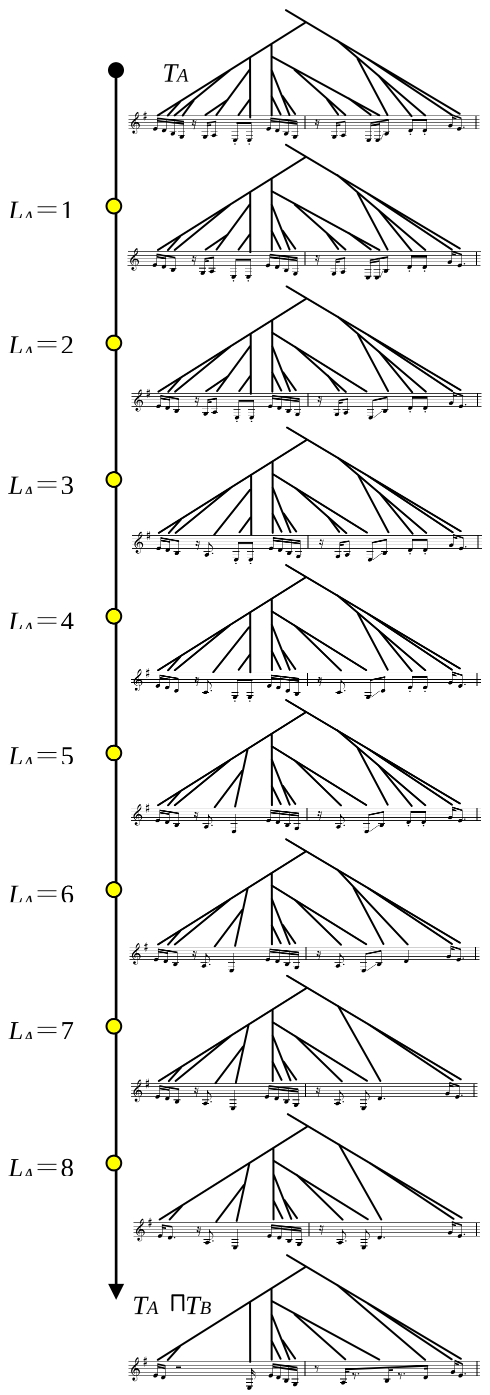


Fig. 19. Melody divisional reduction.

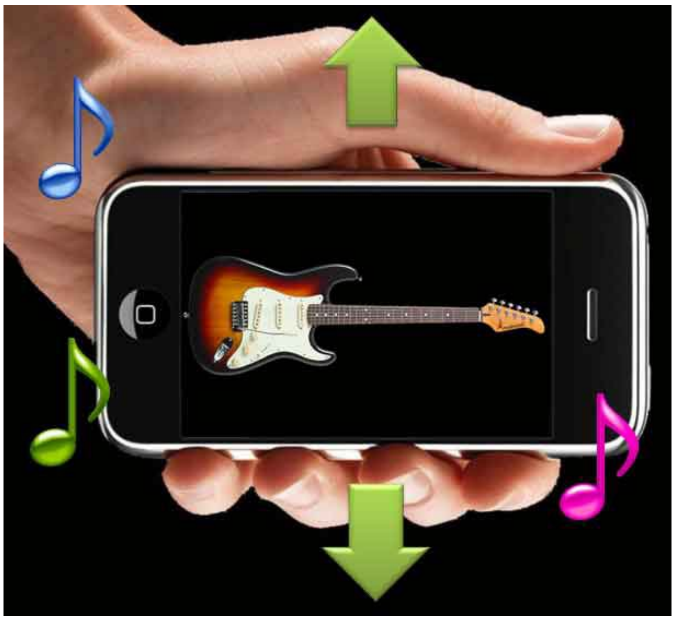


Fig. 19. ShakeGuitar.

6 Experimental Results

It is difficult to compare the performance of our system with those of previous systems, because the approaches taken are completely different. Our method, which is based on music theory, evaluates the appropriateness of the notes from a musical point of view. Therefore, we first quantitatively evaluate each step in our method and then describe an example result.

6.1 Evaluation of ATTA and FATTA

We evaluated the performance of the music analyzer using an F-measure, which is given by the weighted harmonic mean of Precision and Recall,

$$F_{measure} = 2 \times \frac{P \times R}{P + R} \tag{6}$$

This evaluation required us to prepare correct data of a grouping structure, metrical structure, and time-span tree. We collected a hundred pieces of 8-bar length, monophonic, classical music pieces, and asked musicology experts to manually analyze them with faithful regard to the GTTM, using the manual-edit mode of interactive GTTM analyzer to assist in editing the grouping structure, metrical structure, and time-span tree. Three other experts crosschecked these manually produced results.

To evaluate the baseline performance of our system, we used the following default parameters: S^{rules}=0.5, T^{rules}=0.5, *Ws*,=0.5 *Wr* =0.5, *Wl*=0.5, and σ=0.05.

In the current stage, the parameters are configured by humans, because the optimal values of the parameters depend on a piece of music. When a user changes the parameters, the hierarchical structures change as a result of the new analysis.

It took us an average of about 10 minutes per piece to find the plausible tuning for the set of parameters (Table 1,2 and 3). As a result of configuring the parameters, each F-measure of our analyzer outperformed the baseline (Table 4).

Table 4. F-measure of our analyzer outperformed the baseline.

Melodies	Grouping Structure Analyzer		Metrical Structure Analyzer		Time-Span Tree Analyzer	
	Baseline performance	Our method with Configured parameters	Baseline performance	Our method with Configured parameters	Baseline performance	Our method with Configured parameters
1. Moments musicaux	0.18	0.56	0.95	1.00	0.71	0.84
2. Wiegenlied	0.76	1.00	0.83	0.85	0.54	0.69
3. Traumerei	0.60	0.87	0.76	1.00	0.50	0.63
4. An die Freude	0.12	0.73	0.95	1.00	0.22	0.48
5. Barcarolle	0.04	0.54	0.72	0.79	0.24	0.60
	:	:	:	:	:	:
	:	:	:	:	:	:
Total (100 melodies)	0.46	0.77	0.84	0.90	0.44	0.60

After the set of parameters was optimized using FATTA, the average F-measure was 0.48, 0.89, and 0.49, respectively, for grouping, metrical, and time-span tree structures, all better than the baseline performance of ATTA.

6.2 Evaluation of Interactive GTTM Analyzer

We measured the operating time for acquiring the target analysis results of our interactive GTTM analyzer and compared it with that of the GTTM manual editor without an ATTA. For the target analysis, we used one hundred pieces from the three hundred pairs of scores and correct data of grouping structure, metrical structure, and time-span tree. We did not use the prolongational tree in this measurement since its analyzer is still under development. As a result, our interactive GTTM analyzer outperformed the GTTM manual editor without an ATTA (Table 5).

Table 5. Operation time of interactive GTTM analyzer and GTTM manual editor.

Melodies	Interactive GTTM analyzer	GTTM manual editor
1 Grande Valse Brillante	326 sec	624 sec
2. Moments Musicaux	541 sec	791 sec.
3. Turkish March	724 sec	1026 sec
4. Anitras Tanz	621 sec	915 sec.
5. Valse du Petit Chien	876sec.	1246 sec.
	:	:
Total (100 melodies)	575 sec.	891 sec.

6.3 Example of Melody Expectation

Fig. 19 shows the results for Haydn's *andante*. The graph below the musical staff indicates the level of melody stability, corresponding each above note. The number under each note indicates the level of stability for selecting a pitch of 25 possible ones. Stability of a pitch event is calculated from the partial melody between its beginning and the relevant event, and not by the sole event. As a result, the F# in measure 7 does not have the lowest stability, and that of C in measure 5 is lower than that of G in the previous measure, even in C major. Although this may contradict intuition, the calculated result is faithful to our algorithm. At the end of the 4th and 8th measures, the level of stability is high. This is because a dominant chord's note that wants resolve to a tonic chord's note occurs. In contrast, at the beginning of the 5th measure, the level of stability is relatively low. This is because a tonic chord's root note at the beginning of the 5th measure occurs, and various progressions can follow the root note. These results show that our prediction method works well from a musical point of view.

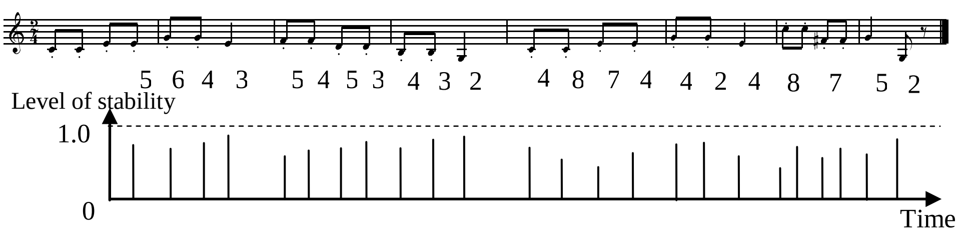


Fig. 19. Example of melody expectation.

6.1 Evaluation of Melody Morphing Method

We tried to determine whether the method can generate the extrapolative melody M from melody A and B which holds following expression.

$$\{R(A,M) < R(A,B)\ and\ R(B,M) < R(A,B)\} \tag{7}$$

where $R(X,\mathrm{Y})$ indicates the similarity between melodies X and Y. The meaning of the expression (7) is that a melody M is an interpolative melody of melodies A and B.

To measure the similarity between melodies X and Y, we used the following $R_N(X, Y)$, defined by Hirata [13], which indicates how much information is lacking from the two melodies as a result of the meet operation.

$$R_N(X,Y) = \frac{|meet(X,Y)|_N}{max(|X|_N, |Y|_N)} \tag{8}$$

where $|X|_N$ indicates the number of notes in melody X.

We use 10 pairs of sample melodies A and B, and as a result of confirmation, all the extrapolative melodies M from melodies A and B, holds the expression (7).

7 Conclusion

We developed a music analyzing system called ATTA and FATTA, which derives the time-span tree of GTTM. We constructed a music analyzer called the interactive GTTM analyzer, which derives the grouping structure, metrical structure, time-span tree, and prolongational tree of the GTTM. The analyzer also derives analysis results of chord progression based on the Tonal Pitch Space. We also proposed melody expectation method and melody morphing method, which uses the time-span tree of GTTM.

The following five points are the main results of this study.

Proposed extended GTTM. We propose an extended GTTM for computer im-plementation. The difficulty with computer imple-mentation of GTTM has been designated, however no radical solution has been proposed. We re-formalized the rules using a numerical expression with adjustable parameters, so that it can separate the definition and ambiguity from the material analyzed.

Implemented Music Analyzer on computer. We implemented an actual working system to ac-quire the hierarchical grouping structure, metrical structure, and time-span tree. The iteractive GTTM analyzer consists of an automatic GTTM analyzer, called an ATTA, a GTTM manual editor, and a GTTM process editor. By using the process editor, a user can seamlessly change the analysis process of the ATTA and that of the manual editor. The experimental results show that our interactive GTTM analyzer outperformed the GTTM manual editor without an ATTA.

Experimented with correct data. Our experimental results showed that, as a result of configuring the parameters, our music analyzer out-performed the baseline F-measure. We made a set of one hundred correct data, which is the greatest database of analyzed results of GTTM thus far. We plan to exhibit the database with the analyzer in the near future. Since we hope to contribute to the research of music analysis, we publicize our interactive GTTM analyzer and a dataset of three hundred pairs of a score and analysis results by musicologist on our website http://music.iit.tsukuba.ac.jp/hamanaka/gttm.htm, which is the largest database of analyzed results from the GTTM to date.

Proposed the Melody Expectation Method. We devised a melody expectation method that predicts candidate notes on the basis of the GTTM and the tonal pitch space (TPS). It is designed to be used with an expectation piano, which displays the predicted notes on its lid, thereby supporting musical novices in playing improvisations.

Proposed the Melody Morphing Method. We constructed a melody morphing method for generating interpolative melodies from two input melodies A and B by using melody divisional reduction, which smoothly decreases the difference information of those melodies. To generate, interpolative melodies, all we need to do is select two input melodies and configure the parameters for controlling the reduction or augmentation level of each melody. We plan to finish developing the interactive melody generator and evaluate whether the melody-morphing method is useful for generating melodies.

We plan to develop further systems, using time-span trees and the results of the music analyzer, for other musical tasks such as harmonizing, voicing, ad-lib, and so on, to indicate the effectiveness of implementing the GTTM to provide music knowledge.

References

1. Apple - GarageBand, http://www.apple.com/ilife/garageband/.
2. Todd, N. : A Model of Expressive Timing in Tonal Music, Musical Perception, Vol. 3, No. 1, pp. 33--58 (1985).
3. Widmer, G.: Understanding and Learning Musical Expression, In Proceeding of the 1983 International Computer Music Conference (ICMC1983), pp. 268--275, New York (1983).
4. Hirata, K. and Hiraga, R: Ha-Hi-Hun plays Chopin's Etude, In Working Notes of IJCAI-03 Workshop on Methods for Automatic Music Performance and their Applications in a Public Rendering Contest, pp. 72--73, Acapulco (2003).

5. Hirata, K. and Matsuda, S.: Interactive Music Summarization based on Generative Theory of Tonal Music, Journal of New Music Research (JNMR), Vol. 32, No. 2, pp. 165--177 (2003).
6. Hamanaka, M., Hirata, K., and Tojo, T.: Implementing "A Generating Theory of Tonal Music", Jornal of New Music Reserch (JNMR), Vol. 35, No. 4, pp. 249--277 (2007).
7. Hamanaka, M., Hirata, K., and Tojo, S.: ATTA: Automatic Time-span Tree Analyzer based on Extended GTTM, In Proceedings of the 6th International Conference on Music Information Retrieval Conference (ISMIR2005), pp. 358--365, London (2005).
8. Hamanaka, M., Hirata, K., and Tojo, S.: FATTA: Full Automatic Time-span Tree Analyzer, Proceedings of the 2007 International Computer Music conference (ICMC2007), Vol. 1, pp. 153--156, Copenhagen (2007).
9. Lerdahl, F., and Jackendoff, R. A Generative Theory of Tonal Music. Cambridge, Massachusetts: MIT Press (1983).
10. Balaban, M.: The Music Structures Approach to Knowledge Representation for Music Processing', Computer Music Journal, 30:2, pp. 96--111 (1996).
11. Cope, D.: Experiments in Musical Intelligence. A-R Editions, Inc. (1996).
12. Dannenberg, R.: Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis', Computer Music Journal, 21:3, pp. 50--60 (1997).
13. Hirata, K., and Aoyagi, T.: Computational Music Representation Based on the Generative Theory of Tonal Music and the Deductive Object-Oriented Database, Computer Music Journal, 27:3, pp. 73--89 (2003).
14. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Extrapolation in GTTM Approach, Proceedings of the 2009 International Computer Music Conference (ICMC2009), pp. 89--92, Montreal (2009).
15. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Morphing Method based on GTTM, Proceedings of the 2008 International Computer Music conference (ICMC2008), pp. 155--158, Belfast (2008).
16. Mozer, M.: Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-Scale Processing, Connection-Science, Vol. 6, No. 2-3, pp. 247--280 (1994).
17. Yen, G., Lucas, S., Fogel, G., Kendall, G., Salomon, R., Zhang, B., Coello, C., and Runarsson, T.: Evolving Musical Sequences with N-Gram Based Trainable Fitness Functions, Proceedings of the 2006 IEEE Congress on Evolutionary Computation, pp. 604--614 (2006).
18. Cooper, G., and Meyer, L. B.: The Rhythmic Structure of Music, The University of Chicago Press (1960).
19. Narmour, E.: The Analysis and Cognition of Basic Melodic Structure. The University of Chicago Press (1990).
20. Temperley, D.: The Cognition of Basic Musical Structures. MIT press, Cambridge (2001).
21. Larson, S.: Musical Forces and Melodic Expectations: Comparing Computer Models with Experimental Results, Music Perception, 21/4, pp. 457--498 (2004).
22. Lerdahl, F.: Tonal Pitch Space, Oxford University Press (2001).
23. Hamanaka, M., Hirata, K. and Tojo, S.: Automatic Generation of Grouping Structure based on the GTTM, In Proceeding of 2004 International Computer Music Conference (ICMC2004) pp. 141--144 (2004).
24. Hamanaka, M., Hirata, K. and Tojo, S.: Automatic Generation of Metrical Structure based on the GTTM, In Proceeding of 2005 International Computer Music Conference (ICMC2005), pp. 53--56 (2005).
25. Sakamoto, S., and Tojo, S.: Harmony Analysis of Music in Tonal Pitch Space, Information Processing Society of Japan SIG Technical Report, Vol. 2009 (2009) (in Japanese).
26. Hamanaka, M., Goto, M., Asoh, H., and Otsu, N.: 'A Learning-Based Quantization: Unsupervised Estimation of the Model Parameters, In Proceedings of 2003 International Computer Music Conference (ICMC2003), pp. 369--372, Singapore (2003).