CE-012

GTTM に基づくメロディレンダリング法
Melody-rendering Method Based on GTTM

浜中 雅俊[1)]
Masatoshi Hamanaka

1 はじめに

我々は GTTM のタイムスパン木に対応するメロディを生成するメロディレンダリング法を提案する．GTTM は，1983 年に Leardahl と Jackendoff により提案され，タイムスパン木は各枝が各音符に接続された 2 分木で，枝は構造的に重要な音符に接続されたものほど根の近くに接続される [1]．タイムスパン木分析器を計算機上に実装する試みが行われてきたが分析エラーが多いという問題があった [2, 3]．一方，深層学習に基づくタイムスパン木分析手法では高い精度を実現した [4]．その手法では，タイムスパンン木分析に自動翻訳の枠組みを導入した点が優れており，本稿で提案するレンダリング法でもその手法を参考にしている．しかし，分析エンジンとして自動翻訳を用いる手法は，場合によっては想定外の翻訳結果が出力される可能性があり，その場合に分析が途中で中断されてしまうという問題があった．そこで我々は，学習データをエンコードする際にどの枝からレンダリングされたかを明示することで想定外の結果が出にくくなるような工夫を行った．

タイムスパン木の特長は，音符を簡約する機能である．簡約操作では，タイムスパン木の構造的に重要な音符の枝を残し，装飾的な音符の枝を取り外す．図 1 のタイムスパン木は，メロディ A を GTTM に基づき分析した結果得られたものである．タイムスパン木のレベル B より下にある枝の音符を省略するとメロディ B のようになる．さらに，レベル C より下にある枝の音符を省略するとメロディ C のようになる．

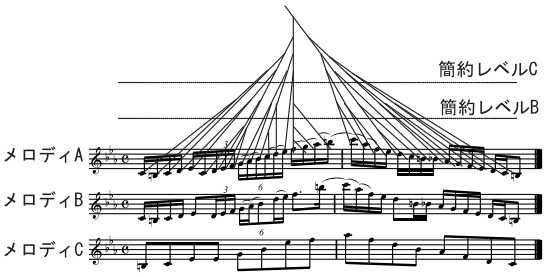


図 1 タイムスパン簡約

このメロディ簡約を応用した手法として，メロディモーフィング法が提案されている [5, 6]（図 2）．メロディモーフィング法は，2 つのメロディのタイムスパン木をそれぞれ簡約した後に join（統合）操作することで，2 つの入力メロディの構造の中間にあるような構造を持ったメロディを生成する手法である．しかし，メロディモーフィング法には大きな問題がある，それは，そもそも 2 つの入力メロディのタイムスパン木の構造が類似していないと join の操作ができない点である．タイムスパン木の構造が類似した曲を探索するのは容易ではない．

1) 理化学研究所 革新知能統合研究センター (AIP)

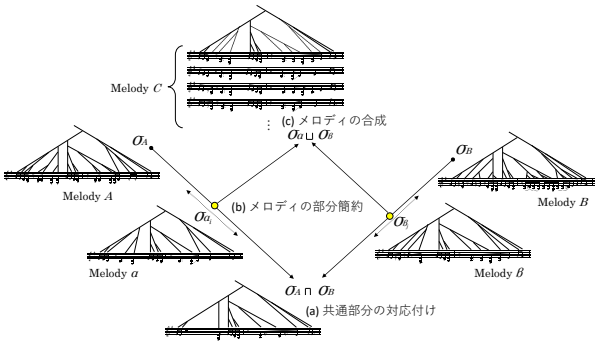


図 2 メロディモーフィングアルゴリズム

これに対し我々は，構造が等しいメロディをレンダリングによって直接生成することを可能にする．具体的には，木の根元から順に逐次的に枝を追加しながら，タイムスパン木の構造に対応したメロディを生成する．メロディレンダリング法には以下の 4 つの特徴がある．

逐次レンダリング: タイムスパン木の構造に対応するメロディを一度に獲得することは難しい．そこで我々は，1 音符ずつの逐次レンダリングを繰り返すことでタイムスパン木全体のレンダリングを実現する．

枝の優先度: 逐次レンダリングを学習するためのデータセットを作成するには，タイムスパン木の枝の優先度を定義する必要がある．我々は，GTTM に基づいて定義された最大タイムスパン長 [2] を優先度として使用する．

エンコード: 楽譜をテキストにエンコードすることで，自動翻訳の枠組みで逐次レンダリングが学習可能となる．これにより，あたかも文章の指定位置に単語を追加するように，メロディの指定位置に音符をレンダリングすることが可能となる．

タイムスパン木行列: これまでタイムスパン木は XML 形式あるいは Json 形式でデータ化されていたため二分木の構造を読み出すためには複雑な再帰プログラムを組む必要があり，プログラム作成に手間がかかっていた [7]．レンダリングに必要な音高，音価，タイムスパン木の構造，枝の優先度の情報を行列で表現することで扱いを容易にした．

GTTM データベースにある 30 のメロディのタイムスパン木について，メロディレンダリングを行う実験をしたところ，すべてでレンダリングが実行できた．本稿では以下 2 節でメロディレンダリング法について説明し，3 節ではレンダリングの実装について述べる．4 節では関連研究について説明し，5 節では実験結果について述べる．そして 6 節では，まとめと今後の課題について述べる．

2　メロディレンダリング法

我々は自動翻訳ネットワークの一つである Transformer モデル [8] に簡約の逆過程であるレンダリングの学習を試みる（図 3）.

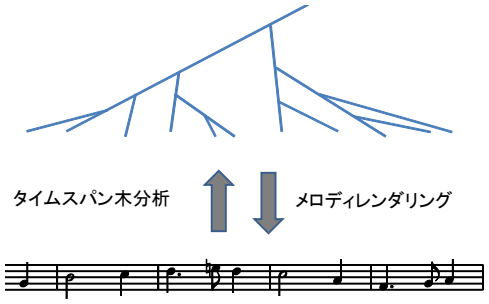


図 3　タイムスパン木分析とメロディレンダリング

2.1　逐次レンダリング

GTTM データベースには 300 件のメロディのタイムスパン木の正解データがある [7]．以前我々は，300 件のデータを用いて深層学習でタイムスパン分析を直接求めることを試みたが，深層学習を行うにはデータ件数が少なすぎて困難であった．そこで，分析の最小単位である，音符を 1 つずつ簡約する過程すなわち逐次簡約を学習対象としたところ学習が可能であった [4]．逐次簡約を行うと図 4 のようにデータ件数が増加する．たとえば，深層ネットワークが 4 音のメロディとそのタイムスパン木の関係を直接学習する場合，データ件数は 1 つだけである．一方，1 つの音符を減らすプロセスを 1 つのデータセットと考えると，データセットの数は 3 つになる．すなわち，4 つの音符で構成されるメロディのタイムスパン木は，4 音から 3 音，3 音から 2 音，2 音から 1 音を推定し，その結果を組み合わせることで構築できる (図 4)．逐次レンダリングは，逐次簡約の入出力を逆にしたものである．

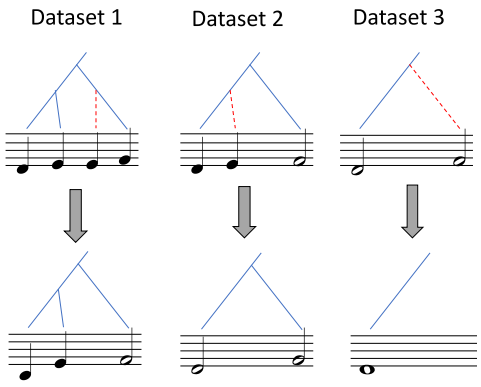


図 4　逐次簡約

2.2　枝の優先順序

タイムスパン簡約は，枝の先端の装飾的な音符を抽象化し，構造的に重要な音を残す．逐次簡約を計算機で実現するためには，枝の優先順序を全順序で決定する必要がある．しかし，GTTM 本では，タイムスパン簡約は数例しか掲載がなく，しかもその手順については詳細な説明がない [1].

Marsden らは，タイムスパン木上で直接接続していない 2 つの枝に接続されている音符イベントについて，イベントの優先度をイベントが接続している枝の最大タイムスパン長とすることを提案した [9].

Transformer モデルが学習する逐次簡約の順序は必ずしも音楽家の認識に近づける必要はなく，計算機上での実装が可能で，かつ，レンダリング前後の関係が学習可能となることが重要である．そのような方法として，我々はタイムスパン長に基づく優先度を提案する．

ヘッドは，タイムスパン木の最も根本にあたる枝で，その木の中で最も優先度が高い音符に接続されている．タイムスパン木は，複数の部分木のヘッドが接続していくことで構成されていると考えることができる．このとき，隣接する 2 つの部分木があったときに，それらが接続されると，部分木の各ヘッドのうちの 1 つが全体の木のヘッドとなる．そしてそのヘッドが，その木が占める時間の中で最も優先度が高くなる．たとえば，部分木が単音の音符だった場合には，その音価の長さが長いほど優先度が高くなる．

最大タイムスパン長: ある音符をヘッドとする部分木が占める最大の時間間隔を最大タイムスパン長と呼ぶ (図 5)．言い換えれば，ある音符の最大タイムスパン長は，その音符がヘッドとなる部分木の時間長と一致する．

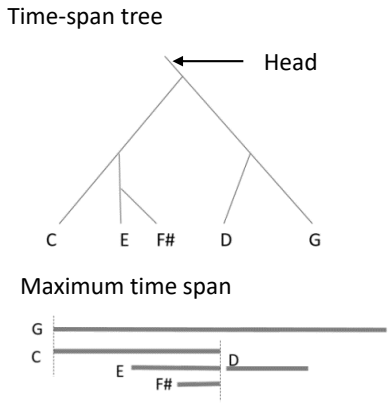


図 5　最大タイムスパン長

そして，タイムスパン木の各枝の優先順序は，タイムスパン簡約の第一段階として行われるタイムスパン分割における部分木の最大タイムスパン長により決定される．具体的には，以下のルールに従って決定される．

- 優先順序はタイムスパン木の上から最大タイムスパン長ごとに割り振られる
- 1 番上のレベルでは，ヘッドが優先される．
- 2 番目以降のレベルでは，枝が接続する幹の優先順位が上なほど，そこから枝分かれする枝の優先度が高くなる．

図 6 のようなタイムスパン木があったとき，第 1 のルールに従い，上から順に優先度が決定されていく．そして，第 2 のルールに従って，枝 1 がこのタイムスパン木で最も優先度の高い枝, 枝 2 が 2 番目に優先度の高い枝となる．このタイムスパン木では，2 番目のレベルが倍全音符レベルである．最大タイムスパン長が同じ場合

には，第 3 のルールに従い，各枝からヘッドに到達するノードが分岐したノードで，枝が親側すなわち優先度の高い枝に接続されたものを優先する．すなわち，枝 1 から分岐した枝が 3，枝 2 の幹から分岐した枝が 4 となる．以下同様に 16 分音符レベルまで優先順位が決まっていく．

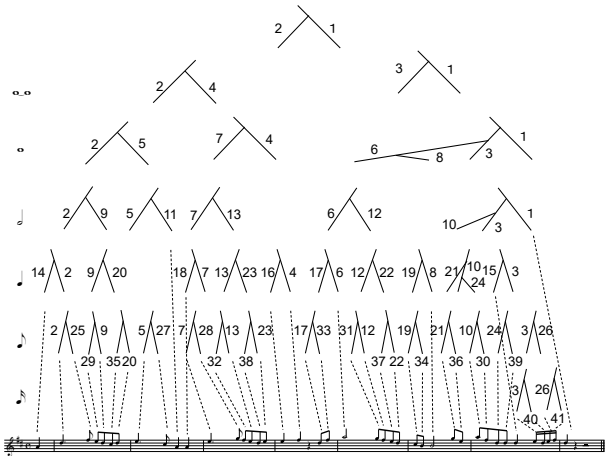


図 6　最大タイムスパン長に基づくタイムスパン木の枝の優先度

2.3　学習・評価データ

逐次レンダリングを学習し評価するためのデータセットは，タイムスパン木から優先度が低い 1 音に対して逐次簡約を行っていくことで作成される．すなわち，逐次簡約の前のデータを出力データ，簡約後のデータを入力データとして学習・評価データセットを作成する（図 7）．

300 曲からなる GTTM データベースのうち 270 曲から 7,362 件の逐次レンダリングの学習用データセットを生成し，残りの 30 曲から 849 件の逐次レンダリングの評価用データセットを生成した．

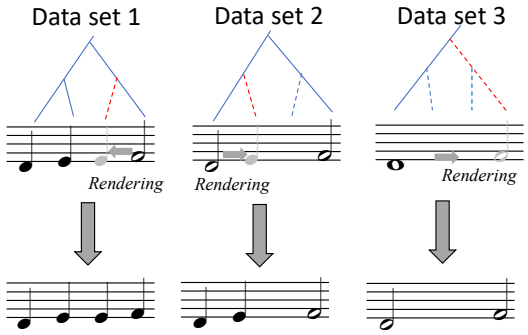


図 7　逐次レンダリング

2.4　データオーギュメンテーション

7,362 件の逐次レンダリングの学習データは，Transformer モデルを学習するのにはまだ十分でないためデータオーギュメンテーションを行う．メロディ中のすべての音符を半音ごとにシフトして学習データを 12 倍に拡張した．さらに，音価を 2 倍から 16 倍までの整数倍に拡張した．最終的に学習データは，1,432,704 (=7,362x12x16) 件となった．

2.5　エンコード

学習･評価データは, GTTM データベースの MusicXML および Time-spanXML から作成する．GTTM データベースのメロディはすべてモノフォニ（和音のない単旋律）であるため，本稿でのレンダリングもモノフォニに限定する．メロディ中の各音符は，音高を表す 1 文字と音価を表す数字を結合した文字列にエンコードする．

音高は，オクターブの違いを区別せず，12 種類で表現する．具体的には，MusicXML の Note タグの c, d, e, f, g, a, b のアルファベット 1 文字もしくは ♯（シャープ）を付けた 2 文字で表現する．♭（フラット）は用いず，たとえば A ♭の場合には G ♯に置き換える．

音価は MusicXML の Note タグの duration の値を 4 倍することでほとんどの音符の音価は整数になるが，データベースには 3 連符，5 連符，6 連符，7 連符がわずかに含まれるメロディがあるため，少数点以下を切り上げて整数にする．なおデータオーギュメンテーションにより音価を 2 倍から 16 倍までの整数倍に拡張しているため，たとえデータに 3 連符，5 連符，6 連符，7 連符などが含まれていても，拡張によって生成されたいずれかのデータでは切り上げをせずとも音価が整数となる．

簡約によって音符がなくなった位置には，プレースホルダを挿入する．逐次簡約の学習では 1 種類のプレースホルダしか用いなかったために，Transfomer モデルから想定外の出力が出た場合にデコードができないことがあった [4]．そこで本研究では，2 種類のプレースホルダ「l」または「r」を挿入する．「l」（左）は簡約によって左の音に吸収されたときに挿入され，「r」（右）は右の音に吸収されたときに挿入される．これにより，レンダリングする音符を明示的に示すことが可能となり Transfomer モデルが予想外の出力を出すことが非常に少なくなった．図 8 は学習データの一例である．

レンダリング前.　→　レンダリング後.

c14 c16 d30 **c26 l** c16 d20 c16. → c14 c16 d30 **c14 c12** c16 d20 c16 .

c14 c16 d30 c26 **r d36** c16. → c14 c16 d30 c26 **c16 d20** c16 .

c14 c16 d30 **r d62** c16. → c14 c16 d30 **c26 d36** c16 .

c30 l d30 R2 d62 c16. → **c14 c16** d30 d62 c16 .

c60 l d62 c16. → **c30 d30** d62 c16 .

c62 **r c78**. → c60 **d62 c16** .

r c138. → **c60 c78** .

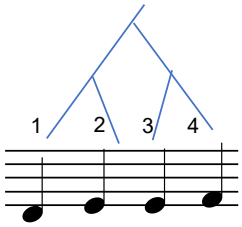
図 8　メロディレンダリングの学習データの一例

3　実装

逐次レンダリングを繰り返すことでタイムスパン木に対応したメロディが獲得される．

3.1　タイムスパン木の行列表現

タイムスパン木を行列で表現することで，メロディの簡約やレンダリングをプログラムとして実装することが容易となる．図 9 のマトリックスの 1 行目は音高と音価をエンコードした文字列, 2 番目は接続している枝の番号である．枝がヘッドで接続する親の枝がない場合の枝の番号は 0 とする．



$$\begin{bmatrix} \text{d8} & \text{e8} & \text{e8} & \text{f8} \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

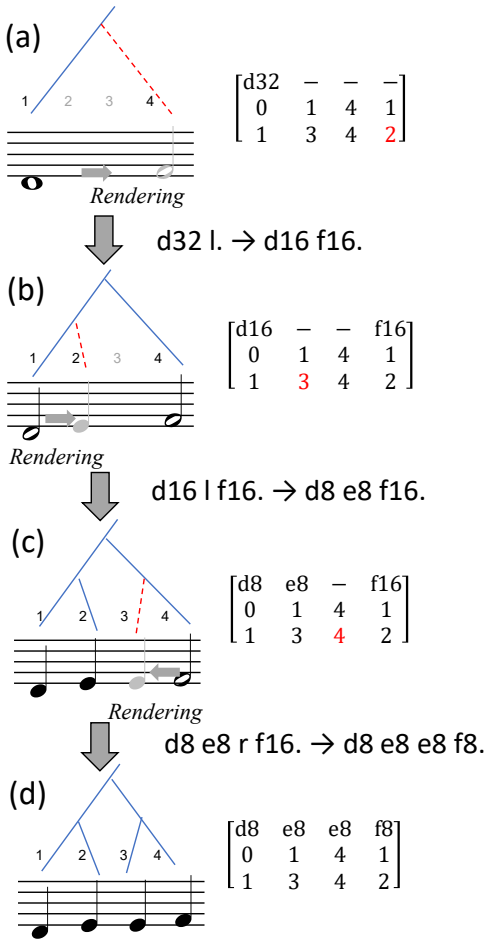
図 9　タイムスパン木の行列表現

図 9 では，2 番の枝も，4 番の枝も 1 番の枝に接続しているが，タイムスパン木の枝は交差しないと定義されているため，4 番が根に近い位置で 1 番に接続していることがわかる．簡約によりなくなった音符やレンダリングによって付加される予定の音符は，行列上で音高や音価の値が空欄となる．

行列の 3 行目は枝の優先度である．枝の優先度はタイムスパン木の構造と音価から求められるので冗長な情報であるが，本稿では説明のために明記する．

3.2　逐次レンダリング

逐次レンダリングの反復によりタイムスパン木に対応したメロディが生成される．図 10(a) のタイムスパン木行列の 2 行目と 3 行目は図 9 と同しである．



$$\begin{bmatrix} \text{d32} & - & - & - \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \text{d16} & - & - & \text{f16} \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \text{d8} & \text{e8} & - & \text{f16} \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \text{d8} & \text{e8} & \text{e8} & \text{f8} \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

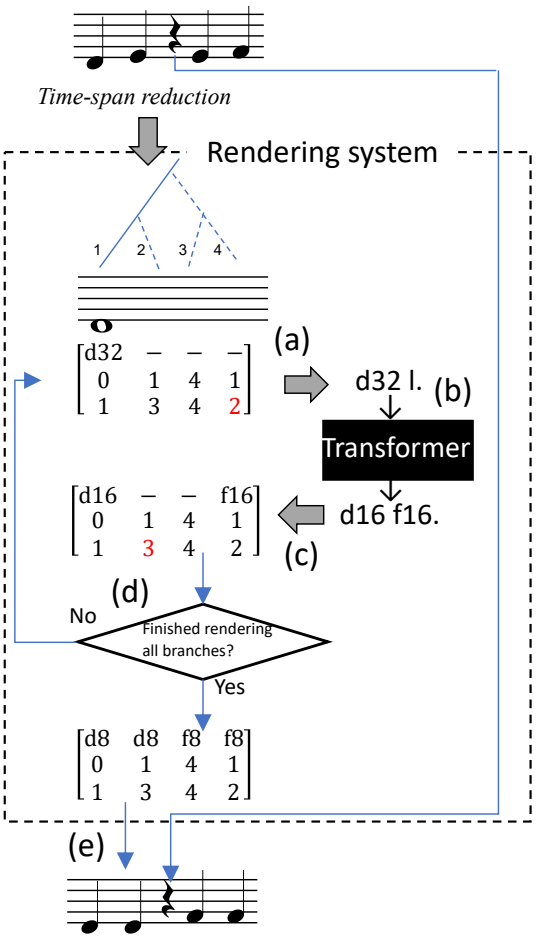
図 10　メロディレンダリング

タイムスパン木のレンダリングの開始時には，音符は 1 つで優先度の最も高いヘッドの枝に接続されている．次のステップでは，2 番目に優先度が高い 4 番の枝に音符がレンダリングされる．1 番の枝は，4 番の枝の左側なので，Transformer モデルへの入力は，”d32 l.” となる．レンダリングされる音符の音高と音価は Transformer モデルの出力によって決まる． 仮に，Transformer モデルの出力が”d16 f16”であれば，タイムスパン木行列の第 1 行は，”d16 - - f16”となる．そして 3 番目に優先度の高い 2 番の枝がレンダリングされ，最後に最も優先度の低い 3 番の枝がレンダリングされる．

Transformer モデルは学習していない入力に対しては想定外の出力を出す可能性がある．場合によっては，レンダリングをそれ以上進めることが困難になることもある．そのような場合には，一旦レンダリングした音をすべて外して初期状態に戻り，ヘッドに接続された音符の音価に 1 から 16 までランダムで選んだ整数をかけた値を新たな音価として再度レンダリングプロセスを開始する．

3.3　レンダリング手順

システムの入力は，1 音まで簡約されたタイムスパン木行列と，休符の位置と長さである．レンダリングは以下の手順で行われる（図 11）．



$$\begin{bmatrix} \text{d32} & - & - & - \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \text{d16} & - & - & \text{f16} \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \text{d8} & \text{d8} & \text{f8} & \text{f8} \\ 0 & 1 & 4 & 1 \\ 1 & 3 & 4 & 2 \end{bmatrix}$$

図 11　メロディレンダリング手法の概要

(a)タイムスパン木行列から Transformer モデルに入力する文字列を生成する．

(b)Transformer モデルによって出力文字列が生成される．
(c)出力文字列をタイムスパン木行列に代入する．
(d)レンダリングする枝が残っていれば (a) に戻る．
(e)休符を元の位置に戻し，終了する．

GTTM では，タイムスパン木における休符の定義があいまいである [1]．しかし，最大タイムスパンの定義では，休符はタイムスパンに含まれないとされている [2]．そこでレンダリングシステムでは，休符を除いてレンダリングした後に元の位置に休符を挿入する．

4 関連研究

本節では，AI に基づくメロディ生成と GTTM に基づくメロディ生成について関連する研究について述べる．また，メロディレンダリングの逆過程である GTTM 分析の関連研究についても述べる．

4.1 AI に基づくメロディ生成

Flow Machines は，マルコフ連鎖をベースとしてメロディとコード進行をモデル化しており，テンポやコード進行などが記述されたスタイルパレットをユーザが作成あるいはプリセットを選択すると，コード進行に合ったメロディをインタラクティブに生成可能であった [10]．

Magenta プロジェクトでは，様々なツールを公開しているが，たとえば Music VAE はメロディ，ベース，ドラムの 2 小節のループメロディが生成可能であった．Improv RNN では入力したコード進行に沿ったメロディ生成可能であった [11]．

Jukebox では，ユーザによるジャンル，アーティスト，タイミング，歌詞などの条件の入力に対する数分の長さの歌詞（歌声）を含む波形データである音楽音響信号の生成を実現した [12]．

言語モデルに基づき音楽を生成する MuiscLM では， Google が開発してきた SoundStream，w2v-BERT，AudioLM，MuLan を組み合わせて，入力文章 (プロンプト) から数分の音楽音響信号の出力を可能とした [13]．

これらの AI に基づくメロディ生成システムでは，それ以前と比べて飛躍的に品質が向上したが，音楽的な完成度はまだ十分ではなく手直しが必要である．[10, 11, 12, 13]．

ここで，音楽 AI の一貫性と操作性について考える．一般に，学習ベースのアプローチを使用して作成された音楽は，局所的には一貫性があるが，大局的な構造が欠けていると指摘されてきた [14]．たとえば，MusicLM で適切なプロンプトをあたえれば比較的大きなレベルの構造も生成可能だが，1 回で目的とする音楽を生成することは困難で，ユーザの意図を反映するためにプロンプトの試行錯誤が必要である．制作における試行錯誤では，ユーザが自分の意図を反映した適切な操作ができること，すなわち操作性が高いことが重要である．一貫性と操作性の高い手法として，我々は音楽理論に基づいた分析手法や作曲手法を計算機上に実装する計算論的音楽理論について研究してきた．

4.2 GTTM に基づくメロディ生成

数多くの音楽理論が提唱されてきているが，我々は Fred Lerdahl と Ray Jackendoff が 1983 年に提唱した GTTM に注目し，GTTM 分析器の構築を行うとともに GTTM に基づくメロディモーフィング法を研究してきた．メロディモーフィングの基本的なアイディアは，1 節で述べた通りである．そのアルゴリズムを実装するためには以下の 2 つの問題がある．

枝の優先度が未定義: メロディの部分簡約において，音符を抽象化する順番が定義されていないため簡約のプロセスを自動化することが困難である．

時間的に重複する音符の発生: 2 つのタイムスパン木を join で結合すると時間的に重複するが音高が異なる音符が発生する．この場合，生成された複数の音符の中から手動で音符を選択する作業をする必要があり，モーフィング手法を完全に自動化することが困難である．

我々は上記の 2 つの問題に対し 2 つの方法で取り組んできた．第 1 の方法では，論理的に一貫したモーフィング実装を提供することを目指した．もともとの join と meet の定義は，枝の構成が大部分重なる木の対に対してのみ適用可能であった．その定義を修正しタイムスパン木の枝の分岐を三分木となることを許容するように拡張を行った [15]．

第 2 の方法では，タイムスパン木の枝の優先順序を定義することで，部分簡約によって抽象化される音符の順序を決定し，かつ，join によって時間的に重複する音符から選択する音符の順序を決定した．これによって，2 つのメロディのタイムスパン木 σ_A と σ_B があったとき，それぞれについて抽象化される音符の数が決まると，モーフィング結果のメロディ C が一意に得られるようになった [4, 16]．

上記のように，メロディモーフィング法は複数の方法が存在していることから，メロディレンダリング法も複数の方法が存在する可能性があり，今回はそのうち 1 つの方法を実装したと考えている．

4.3 構造が類似したメロディの利用

モーフィングは，入力メロディ A,B と生成された C がいずれも構造が類似している特長がある．構造が類似していると，メロディの一部を別のものに置き換えても，全体の構造は大きく変化せず違和感が少ない．その特長を利用したシステムがメロディスロットマシンである [17, 18]（図 12）．ダイヤル型のインタフェースを操作するとメロディの一部を別のバリエーションに切り替えることができる．

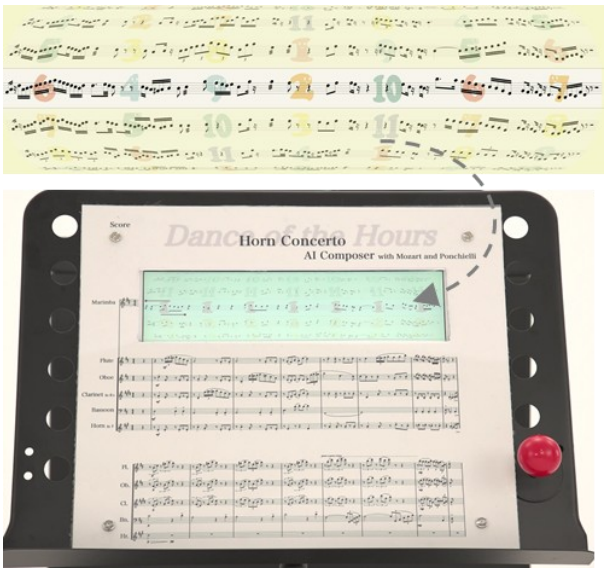


図 12 メロディスロットマシン

研究展示ではあたかも目の前に小さな演奏者がいるかのように感じられるようホログラムディスプレイ（ペッパーズゴーストホログラム）を使って演奏者を正面からだけでなく左右からも見ることを可能とした．（図 13）．

図 13　ペッパーズゴーストホログラム

装置の制作時のレコーディングは残響音が非常に少ないレコーディングスタジオで行った．残響が長い場合には，メロディをセグメントに分割する際に，前のセグメントの演奏の残響音のみが次のセグメントに入りこみやすく，バリエーションを切り替えた場合に不自然となる可能性がある．しかし，再生時に残響が短いままであると不自然であるため，音が出力されるタイミングでプラグインエフェクトを使って残響音を付加した．その際，各方向ごとにリバーブのかけぐあいや音量バランスの調整を行い，ホログラム上で奏者が見える場所から音が発生しているかのようにチューニングした．たとえば，装置に向かって右側からマリンバ奏者を見ると，手前側が低音の鍵盤で奥が高音の鍵盤になるので，そのように聞こえるようにプラグインエフェクトのパラメーターを調整した．図 14 は，方向ごとのスピーカーを設置した様子である．

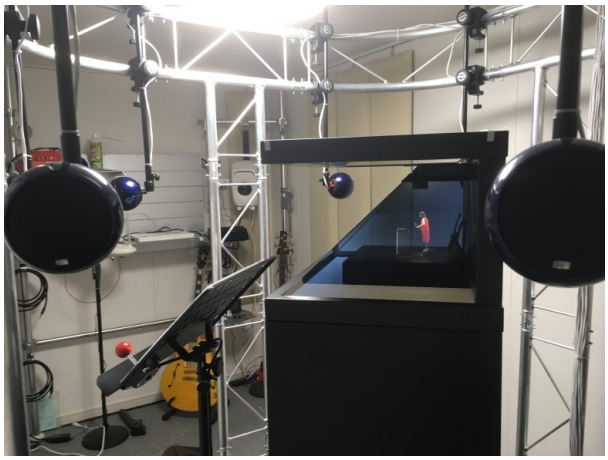


図 14　方向ごとのスピーカーの設置

展示会用のメロディスロットマシンは，大型のペッパーズゴーストホログラム装置に加え，スピーカーを釣るトラスなど大掛かりな準備が必要であった．またシステムとしても iPad2 台と Mac mini を MIDI ケーブルによる通信で連携した複雑な構成であった．そこで，小型の組み立て式のペッパーズゴーストホログラムを自作し，2 台の iPad のワイヤレス通信のみで動作可能なポータブル版のメロディスロットマシンを構築した（図 15）[19].

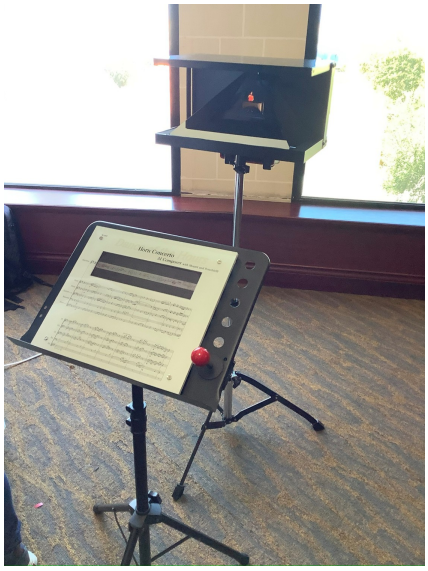


図 15　ポータブル版のメロディスロットマシン

メロディスロットマシン HD は，ポータブル版で 2 台の iPad で動作させていた機能を 1 台の iPad で動くよう改良を行い構築したものである（図 16）[20].

メロディスロットマシンの構築により，モーフィングで生成した構造が類似したメロディに利用価値があることが確認された．しかし，モーフィングでは入力する 2 つのメロディの構造が類似している必要があり，そのようなメロディ対を見つけることは容易ではなかった．そこで，構造に対応するメロディを直接生成できる方法としてレンダリングの実現方法を検討してきた．

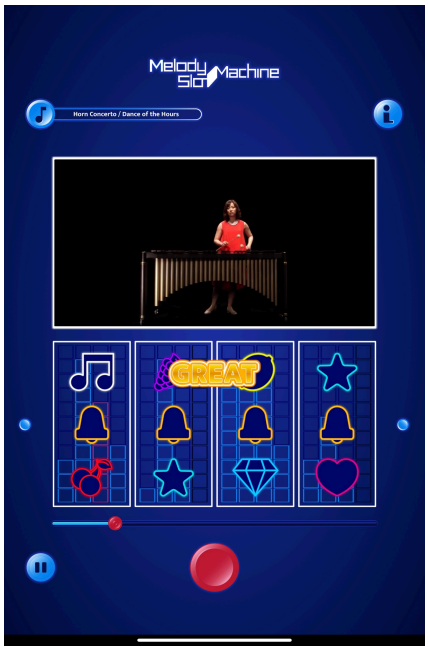


図 16　メロディスロットマシン HD

4.4 GTTM に基づく分析

メロディレンダリングは，GTTM 分析の逆過程であり，GTTM 分析の精度が高まったことがレンダリングの実現のきっかけとなった．GTTM 分析は，グルーピング構造，拍節構造，タイムスパン木の順に分析を進めていく．GTTM のルールをナイーブに実装したルールベースに基づく分析器 ATTA は，人手によるパラメータ調整が必要であり，分析性能も低かった [21, 22, 23, 24, 25, 26]. 人手によるパラメータ調整を不要とするよう改良されたシステム FATTA では，自動化は達成できたが分析性能はさらに低くなっていた [27, 28]. これらの結果は，GTTM のルールは非常に数が多く，人手の場合でも自動の場合でも各ルールの優先度などのパラメータを適切に調整することが困難なことを示している．

グルーピング構造分析に決定木を導入した分析器 σGTTM-I では，分析を行う複数の決定木の中から分析者が適切なものを選択できれば性能が向上したが，適切でない決定木を選択した場合には性能が低下した [29, 30]．決定木を学習する曲をクラスタリングしてクラスタごとに決定木を学習させた σGTTM-II では，σGTTM-I に比べてさらに分析性能が向上したが，分析者が適切な決定木を選択する点に変わりなく，適切でない決定木を選べば性能が低下した [31, 32].

タイムスパン木の分析器に確率文脈自由文法を導入した σGTTM-III 等では，タイムスパン木分析の自動化を実現したが，分析性能は十分ではなかった [3, 33, 34, 35, 36]．タイムスパン木分析器の性能を向上させることが難しい理由として，タイムスパン木が二分木であるために組み合わせの数が非常に多い点と，木の根元に近い枝で分析誤りがあるとそのエラーが伝播して大きく精度が低下する点があげられる．また，曲中の局所的な部分に関するルールと大局的な部分に関するルールが競合すると，それを解消することが難しいという問題もある．たとえば，局所的なルールであるタイムスパン簡約選好ルール 1 (TSRPR1: Time-span reduction preference rule) では，小節の先頭の音が強拍の場合，小節内で最も構造的に重要な音すなわちヘッドになりやすい．すると，ルールベースの手法でも確率的な手法でも小節の頭がヘッドになりやすいようにパラメータや確率の重みが増加することになる．その重みは曲の全体に対して決まるので，たとえば曲の最終小節であっても同様に，小節の先頭の強拍の音符がヘッドになりやすい．一方，大局的なルールであるタイムスパン簡約選好ルール 7（TSRPR7）では，曲の終止音がたとえ小節の先頭でなくても曲の中で最も構造的に重要な音となるが，TSRPR1 の重みが大きいと終止音ではなく小節の先頭の音をヘッドとするような誤推定が発生しやすい．

上記のように当初は曲の局所的な部分を見てボトムアップに全体の構造を生成していく GTTM 分析器を構築してきたが，そのような手法では性能を飛躍的に向上させることは困難であった．そこで我々は GTTM 分析に深層学習を導入することを考えた．深層学習であれば曲の全体の情報をネットワークの入力に入れることが可能であるため，局所的な情報と大局的な情報の両方を扱うことが可能である．音符列を深層ネットワークの入力に展開した deepGTTM-I, II, III では，グルーピング構造と拍節構造の分析性能が飛躍的に向上した [37, 38, 39, 40, 41, 42]．タイムスパン木分析では，一音ずつ簡約する逐次簡約を自動翻訳器で学習することで飛躍的に性能が向上した [4].

タイムスパン木分析における逐次簡約とその逆過程である逐次レンダリングは，学習時の入力と出力を入れ替えた関係にあり，類似した問題であると言える．しかし，逐次簡約では，データオーギュメンテーションなしで 1 つのプレースホルダを使って学習が可能であったのに対し，逐次レンダリングではデータオーギュメンテーションしたデータを使い，かつ，2 つのプレースホルダを用意しなければ学習することが困難であった．これは，メロディから 1 音減少させる逐次簡約に比べて，メロディに新たに 1 音追加する逐次レンダリングのほうが問題の難易度が高かったためだと考えられる．

5 実験結果

逐次レンダリングを Transformer モデルで学習可能であるか実験により確認する．また，音符を取り外したタイムスパン木に対して，学習済みの Transfomer モデルを使って逐次レンダリングを繰り返すことでレンダリング可能であるか確認する．Transformer モデルは OpenNMT-py toolkit (ver. 2.0.0rc2) [43] で実装した．

5.1 逐次レンダリングの学習

270 曲のタイムスパン木データからデータオーギュメンションによって作成された 1,432,704 件の逐次レンダリングの学習データで Transformer モデルを学習させた．その結果，20,000 Epoch の学習により評価データでの精度が 0.99 まで向上した．一方，データオーギュメンテーションをしていない，7362 件の逐次レンダリングデータで Transfomer モデルを学習したところ，評価データでの精度は 0.89 であった．データオーギュメンテーションによりモデルの性能が向上したことが確認された．なお，Transformer モデルのハイパーパラメータをランダムに変化しても推定精度には大きな変化がなかったため，デフォルトのパラメータを使用した．

5.2 タイムスパン木のレンダリング

学習に使っていない 30 曲のタイムスパン木について，それぞれ 1 音になるまでリダクションした後，レンダリングを行った．その結果，すべてのタイムスパン木でレンダリングが可能であった．30 曲中 28 曲では，レンダリングプロセスは 1 回で終了したが，残りの 2 曲ではプロセスを複数回おこなったあとレンダリングを完了した．それらの 2 曲にはいずれも 3 連符が含まれていた．

6 まとめ

本稿では音楽理論 GTTM に基づくメロディレンダリング法を提案した．本稿の貢献は以下の 2 点である．

レンダリングの概念の提案: タイムスパン木への操作としては，簡約，meet, join, モーフィングなどがこれまでも知られていたが，我々はレンダリングという新たな概念を提案した．タイムスパン木に基づくメロディ操作の中で，我々はレンダリングが最も自由度が高いと考えている．たとえば，レンダリングを行うには音符が簡約されたタイムスパン木が必要であるが，それは手作業で作ることもできるし，ランダムに発生させることもできる．またメロディ中で変更を加えたい音符のみ一度簡約した後に再度レンダリングすることもできるし，タイムスパン木に新たな枝を追加してその枝に音符をレンダリングす

ることもできる．

レンダリングの実装: レンダリングが実現したのは Transfomer モデルの強力な学習能力に依存する部分が大きい．しかし，逐次レンダリング, エンコード, タイムスパン木行列のアイディアもシステムを実装する上で欠かすことができない．逐次レンダリングとエンコードによって，極めて少ない正解データから Transformer モデルに適した学習データを生成することを可能にした．これまで，タイムスパン木の簡約の途中あるいはレンダリングの途中のデータが扱われることはなかったが，我々はタイムスパン木行列がそれを表現する上で適切な形式だと考える．

実験によりメロディレンダリングが実現可能なことが明らかになった．その結果，以下のような遂行すべき今後の課題が生じた．

メロディの評価: レンダリング法を作曲支援システムなどで利用する前に，手法で生成されたメロディに対して音楽的な評価が必要である．本稿では 30 のタイムスパン木のレンダリングを行った．より多くのタイムスパン木のレンダリングを行い音楽家による評価実験を行っていく．

様々なメロディの生成: Transfomer モデルは同じ入力に対して基本的には同じ出力となる．レンダリング法を作曲を支援するアプリケーションで使うことを考えると，様々なメロディが出力されるほうが望ましい場合がある．たとえば，変分オートエンコーダ（VAE: Variational Autoencoder) のように潜在変数を持ったモデルで学習することで，出力されるメロディに変化をもたせることが可能となろう．あるいは，今回の実験で学習に使った 300 件のタイムスパン木はクラッシック曲によるものであったが，ポップスやジャズなどのタイムスパン木を学習したネットワークを構築することで，様々なバリエーションを持ったメロディが出力されるようにしていきたい．

レンダリングメロディの利用: メロディスロットマシンは，メロディモーフィング法の特長を活かしたシステムであった．レンダリングメロディの特長を明らかにするために，レンダリングメロディを利用したシステムを構築していく予定である．また，アプリなどでダウンロード可能とすることで，レンダリング法を普及させていきたい．

作曲支援インタフェースの構築: メロディレンダリング手法が多くの人が利用可能にするために作曲支援インタフェースの構築を開始しており，今後公開する予定である．インタフェースでは，タイムスパン木を用いて一部の音符を簡約してから再度レンダリングしたり，タイムスパン木に枝を追加してその枝にレンダリングすることを可能とする（図 17）．

本稿ではレンダリングを実現可能にすること優先したため，議論が十分でない点が残されている．たとえば，休符をタイムスパンに含めるかについて，今回は実装が容易になるよう休符をタイムスパンに含まず，レンダリングプロセス後に同じ位置に戻す方法を選択した．休符も含めてエンコードすることも可能であり，さらなる検討を続けていく．

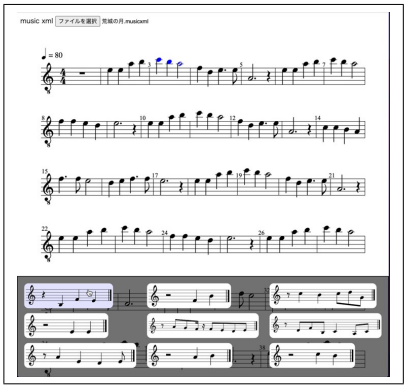


図 17　作曲支援インタフェース

謝辞

本研究の一部は JSPS 科研費 JP21H03572 の助成を受けたものです．

参考文献

[1] Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. The MIT Press, Cambridge. MA, 1983.

[2] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementation of melodic morphing based on generative theory of tonal music. *Journal of New Music Research*, Vol. 51, No. 1, pp. 86–102, 2023.

[3] 中村栄太, 浜中雅俊, 平田圭二, 吉井和佳. GTTM に基づくメロディ音符列の確率的木構造モデル. 人工知能学会全国大会論文集, 3G4-OS-15b-4, 2016.

[4] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Time-span tree leveled by duration of time-span. In *Proceedings of the 15th International Symposium on Computer Music Multidisciplinary Research* (CMMR2021), Vol. 1, pp. 155–164, 2021.

[5] 浜中雅俊, 平田圭二, 東条敏. タイムスパン木に基づくメロディモーフィング法. 情報処理学会 音楽情報科学研究会研究報告 2008-MUS-74, No. 12, pp. 107–112, 2008.

[6] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Melody morphing method based on GTTM. In *Proceedings of the 2008 International Computer Music Conference* (ICMC2008), pp. 155–158, 2008.

[7] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Musical structural analysis database based on GTTM. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR2014)*, pp. 325–330, 2014.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 2017.

[9] Alan Marsden, Satoshi Tojo, and Keiji Hirata. No longer 'somewhat arbitrary': Calculating salience in GTTM-style reduction. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, DLfM '18, pp. 26–33, New York, NY, USA, 2018. Association for Computing Machinery.

[10] Alexandre Papadopoulos, Pierre Roy, and Francois Pachet. Assisted lead sheet composition using flowcomposer. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming*, Vol. 9892, pp. 769–785, 2016.

[11] Claire Kayacik, Sherol Chen, Signe Noerly, Jess Holbrook, Adam Roberts, and Douglas Eck. Identifying the intersections: User experience + research scientist collaboration in a generative machine learning interface. In *CHI EA '19: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–8, 2019.

[12] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A genera-

tive model for music. In *arXiv:* https://arxiv.org/abs/2005.00341, 2020.

[13] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text. In *arXiv:* https://arxiv.org/abs/2301.11325, 2023.

[14] Douglas Eck and Jürgen Schmidhuber. Learning the long-term structure of the blues. In José R. Dorronsoro, editor, *Artificial Neural Networks — ICANN 2002*, pp. 284–289, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[15] Keiji Hirata, Satoshi Tojo, and Masatoshi Hamanaka. Algebraic mozart by tree synthesis. In *Joint Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference* (ICMC2014 & SMC2014), pp. 991–997, 2014.

[16] 小林瑞季, 浜中雅俊. 新しい GTTM メロディモーフィング手法の提示：既存手法とマリンバ作品を経て. 情報処理学会 音楽情報科学研究会 研究報告 2019-MUS-123, No. 40, 5 pages, 2019.

[17] Masatoshi Hamanaka, Takayuki Nakatsuka, and Shigeo Morishima. Melody slot machine. In *ACM SIGGRAPH 2019 Emerging Technologies ET-245*, No. 2 pages, 2019.

[18] Masatoshi Hamanaka. Melody slot machine: A controllable holographic virtual performer. MM '19, New York, NY, USA, 2019. Association for Computing Machinery.

[19] Takayuki Nakatsuka, Masatoshi Hamanaka, and Shigeo Morishima. Audio-guided video interpolation via human pose features. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (VISIGRAPP 2020), pp. 27–35, 2020.

[20] 浜中雅俊. メロディスロットマシン hd. 情報処理学会 音楽情報科学研究会 研究報告 2023-MUS-137, No. 7 pages, 2023.

[21] 浜中雅俊, 平田圭二, 東条敏. 音楽理論 GTTM に基づくグルーピング構造獲得システム. 情報処理学会論文誌, Vol. 48, No. 1, pp. 284–299, 2007.

[22] Masatoshi Hamanaka, Keiji HIRATA, and Satoshi Tojo. Automatic generation of grouping structure based on the GTTM. In *Proceedings of the 2004 International Computer Music conference* (ICMC2004), pp. 141–144, 2004.

[23] 浜中雅俊, 平田圭二, 東条敏. GTTM に基づく楽曲構造分析の実装: グルーピング構造と拍節構造の獲得. 情報処理学会 音楽情報科学研究会 研究報告 2004-MUS-56-1, No. 84, pp. 1–8, 2004.

[24] Masatoshi Hamanaka, Keiji HIRATA, and Satoshi Tojo. Automatic generation of metrical structure based on GTTM. In *Proceedings of the 2005 International Computer Music conference* (ICMC2005), pp. 53–56, 2005.

[25] 浜中雅俊, 平田圭二, 東条敏. ATTA: exGTTM に基づく自動タイムスパン木獲得システム. 情報処理学会 音楽情報科学研究会 研究報告 2005-MUS-61-4, No. 82, pp. 19–26, 2005.

[26] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. ATTA: Automatic Time-Span Tree Analyzer Based on Extended GTTM. In *Proceedings of the 6th International Conference on Music Information Retrieval* (ISMIR2005), pp. 358–365, 2005.

[27] 浜中雅俊, 平田圭二, 東条敏. FATTA に基づくメロディ予測システム. 情報処理学会 音楽情報科学研究会 研究報告, No. 78, pp. 45–50, 2008.

[28] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. FATTA : Full automatic time-span tree analyzer. In *Proceedings of the 2007 International Computer Music Conference* (ICMC2007), Vol. 1, pp. 153–156, 2007.

[29] 三浦右士, 浜中雅俊, 平田圭二. 統計的学習に基づく音楽理論 σGTTM: 局所的グルーピング境界の検出. 情報処理学会 音楽情報科学研究会 研究報告 2008-MUS-76-14, No. 78, pp. 75–82, 2008.

[30] Yuji Miura, Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Use of decision tree to detect GTTM group boundaries. In *Proceedings of the 2009 International Computer Music Conference* (ICMC2009), pp. 125–128, 2009.

[31] 金森光平, 浜中雅俊, 星野准一. 類似楽曲の決定木学習に基づく音楽理論 GTTM のグルーピング構造検出システム. 電子通信学会論文誌, Vol. J100-D, pp. 129–139, 2017.

[32] Kouhei Kanamori, Masatoshi Hamanaka, and Junichi Hoshino. Method to detect GTTM local grouping boundaries based on clustering and statistical learning. In *Joint Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference* (ICMC2014 & SMC2014), 2014.

[33] 浜中雅俊. σGTTMIII の構築. 人工知能学会全国大会論文集, 2C5-OS-21b-1, 2015.

[34] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. σGTTM III: Learning-based time-span tree generator based on pcfg. In *Proceedings of The 11th International Symposium on Computer Music Multidisciplinary Research* (CMMR2015), Revised Selected Papers, Lecture Notes in Computer Science, Vol. 9617, pp. 387–404, 2016.

[35] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP2016), pp. 276–280, 2016.

[36] Ryan Groves. Automatic Melodic Reduction Using a Supervised Probabilistic Context-Free Grammar. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR2016)*, pp. 775–781. ISMIR, 2016.

[37] 浜中雅俊. deepGTTM-I: ディープラーニングに基づく局所的グルーピング境界分析器. 人工知能学会全国大会論文集, 3G4-OS-15b-3, 2016.

[38] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepGTTM-I: Local Boundaries Analyzer based on Deep Learning Technique. In *Proceedings of the 13th International Symposium on Computer Music Multidisciplinary Research* (CMMR2016), pp. 8–20, 2016.

[39] 浜中雅俊, 平田圭二, 東条敏. deepGTTM-II: ディープラーニングに基づく拍節構造分析器. 情報処理学会 音楽情報科学研究会研究報告 2016-MUS-112, 2016.

[40] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepGTTM-II: Automatic Generation of Metrical Structure Based on Deep Learning Technique. In *Proceedings of the 13th Sound and Music Conference* (SMC2016), pp. 221–249, 2016.

[41] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepGTTM-I&II: Local boundary and metrical structure analyzer based on deep learning technique. In *Proceedings of The 13th International Symposium on Computer Music Multidisciplinary Research* (CMMR2017), Revised Selected Papers, Lecture Notes in Computer Science, pp. 3–21, 2017.

[42] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepGTTM-III: Multi-task learning with grouping and metrical structures. In *Proceedings of The 13th International Symposium on Computer Music Multidisciplinary Research* (CMMR2018), Revised Selected Papers, Lecture Notes in Computer Science, pp. 3–21, 2018.

[43] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pp. 67–72, Vancouver, Canada, 2017. Association for Computational Linguistics.