DEIM Forum 2012 F6-4

# 頻出直列エピソードマイニング手法を用いた音楽データからの繰り返し部分抽出

藤川　純平†　　喜田　拓也†

† 北海道大学 大学院情報科学研究科　〒 060–0814 北海道札幌市北区北 14 条西 9 丁目
E-mail: †{f-jumpei,kida}@ist.hokudai.ac.jp

**あらまし**　本研究では，信号処理的なアプローチの代わりに，離散的な知識発見手法によるアプローチを用いて，音楽信号から繰り返し部分の抽出をする方法について述べる．この提案手法は二つの手順からなる．一つ目は，音楽信号から音高情報を表す離散的なイベントの列へと変換すること，二つ目は，イベントの列より頻出なパターンをマイニングすることである．前者は，楽曲の拍間隔毎のクロマベクトルを計算することにより行われ，後者は，すべての頻出エピソードパターンを列挙することにより行われる．提案手法が繰り返し部分を表すパターンを抽出することを確認するために，RWC Music Database の楽曲を用いた予備実験を行った．

**キーワード**　データマイニングアルゴリズム，直列エピソードパターン，RWC 音楽データベース

# Extracting refrained phrases from music recordings using a frequent serial episode pattern mining method

Jumpei FUJIKAWA† and Takuya KIDA†

† Graduate School of Information Science and Technology, Hokkaido University
14–9 Kitaku, Sapporo, Hokkaido, 060–0814 Japan
E-mail: †{f-jumpei,kida}@ist.hokudai.ac.jp

**Abstract**　In this paper, we discuss a method for extracting refrained phrases from a music signal by a discrete knowledge discovery processing approach instead of a signal processing approach. The proposed method consists of two processes: translating a music signal into a sequence of events that represent pitch information, and then mining the frequent patterns from the event sequences. The former is performed by computing chroma vectors at every beat interval, and the latter is performed by enumerating the frequent episode patterns. We carried out a preliminary experiment on some pieces in the RWC music databases to examine if the extracted patterns represent the refrained phrases.

**Key words**　data mining algorithm, serial episode pattern, RWC music database

## 1. はじめに

本研究は，実世界の音楽音響信号に対して，人間が理解するようにして音楽を特徴づけられるシステムを目指して，楽曲検索の基礎技術について取り組んでいる．本稿では，特に，楽曲を特徴づける大局的な音楽的要素である繰り返し部分 (refrain) の抽出について議論する．refrain とは，楽曲全体の構造の中で，繰り返し現れる主題の部分である．この refrain を抽出することは，多くの応用システムにとって有益である．たとえば，多数の楽曲からの高度な検索や，refrain 部分のプレビュー提示による高速なブラウジングを可能とする [1].

楽曲の類似部分抽出に対するこれまでの研究アプローチは，信号処理的なものが主である．すなわち，音声信号の類似性に着目して，連続して波形が似ている部分を類似した楽曲部分と考える．より，進んだ考え方としては，音声信号から音高をとらえた特徴ベクトルへと変換し，その特徴ベクトル列上での類似尺度を定義することで，楽曲の類似部分を検索する方式が考案されている [2]〜[4]. これまで，楽曲中に含まれる多少のテンポのずれや転調などに対応するために，様々な工夫がこの方式の上に組み上げられている．既に実用的なレベルにまで達成しているシステムも存在するが，複数種類の楽器音や歌声を含む複雑な混合音を対象にしているため，広範囲な楽曲の種類に精度良く対応できるシステムを構築することは容易ではない．

一方で，楽曲理解のための記号処理的なアプローチも存在す

る [5]. 例えば，手元に楽譜データがあり，信号ではなく音符の列として楽曲が与えられている場合に，その上で類似部分を検索したり抽出したりする手法である．すなわち，楽譜を記号の列（文字列）とみなして処理を行う．このアプローチにおいては，楽曲中の各パートは分離して与えられると仮定していることが多い．例えば，MIDI データのように，主旋律や，ベース，ドラムなどの各パートが別々に入力として与えられる．この場合，リズムの理解はドラムパートを用い，和声解析には主旋律を用いるといったように，解析者にとって非常に都合のいい状況であるため，高い精度の処理を行うことができる．もちろん，このような「きれいな」データを入力として考えることのできる状況であれば，記号処理的なアプローチは精度的にも計算量的にもリーズナブルな選択である．しかし，手元に楽曲の音響信号データしか無い場合，それを「きれいな」パート情報に変換しなくてはならず，それは完璧に行うことは困難である．音響信号と楽譜データとのマッピングを行う研究も提案されているが，やはり楽譜データが必要となる [6]〜[8].

本稿では，音響信号を対象に，記号処理的なアプローチで，曲の refrain を抽出する手法について議論する．このアプローチでは，まず音響信号を記号列に変換する必要がある．我々は，音響信号から，各パートを完全に分離することをあきらめ，音響信号を chroma vector 列に変換したものから，単純に強度の強い音の列を並べることで楽曲を記号列へと変換する [9]．このように生成された記号列は，すべてのパートが混合した「汚い」データである．このような汚いデータから，繰り返し現れる不連続な記号の列（エピソードパターン）を抽出することで，楽曲の refrain 部分を抽出する．抽出されたパターンは，もはや refrain な単一のメロディラインを表すものではなく，ベースやリズム音も含むようなごちゃごちゃしたものとなる．しかしながら，このアプローチは，ある意味，人間が楽曲を把握するやり方に近いとも考えられる．なぜなら，私たちは，曲の主旋律だけでなく，ベースやドラム音も混ざった風に曲を口ずさむことができるからである．我々はまた，予備的な実験により，このようにして抽出したパターンが，楽曲の refrain 部分を良く表していることを確認した．

以下，まず 2. 節でエピソードパターンと頻出エピソードマイニングについて述べる．次に 3. 節で我々が提案する手法について解説し，4. 節にて行った実験の説明を行う．最後に 5. 節にてまとめを述べる．

## 2. 頻出エピソードマイニング

この節では，エピソードパターンの定義について紹介する，また頻出するエピソードパターンのマイニング問題についても述べる．我々は，ここでは各単位時間において一つのみのイベントが起きているものとする．一般的なケースについては [10] を参照されたい．

a） 基本的な用語と表記について

$\Sigma$ を有限のアルファベットとする．$\Sigma^+$ を $\Sigma$ 上の空文字列を含まない文字列の集合と定義する．任意の要素 $e \in \Sigma$ を**イベント**と呼び，要素の列 $S \in \Sigma^+$ を**イベント列**と呼ぶ．$S[i]$ を $S$ の $i$ 番目のイベントと定義し，$S[i..j]$ を $S$ の $i$ 番目から $j$ 番目までの連続したイベントの列と定義する．ここで，$i$ と $j$ は正の数であり，かつ $i < j$ であるとする．また，$S$ からいくつかのイベントを取り除くことによって得られた列が $\alpha \in \Sigma^+$ に対応するとき，$\alpha \in \Sigma^+$ を $S$ の**部分列**と呼ぶ．

長さ $n$ のイベント列 $S = S[1..n]$ について考える．$win$ を $1 \leqq win \leqq n$ を満たす整数とする．また，任意の $0 \leqq i \leqq n - win + 1$ における部分文字列 $W = S[i..i + win - 1]$ を**窓**と呼び，そのサイズを**窓幅**と呼ぶ．

$\mathcal{W}(S, win) = \{W_i = S[i..i+win-1] : i = 1, \ldots, n-win+1\}$ を，窓幅が $win$ である全ての窓の集合とする．ここで $\mathcal{W}(S, win)$ に含まれる窓の数は $n - win + 1$ である．

$S$ 上の**区間**とは，2 つの 上の 位置 $[i, j] \in \mathbf{N}^2$ $(i \leqq j)$ であり，$S$ の位置の集合 $\{i, i+1, \ldots, j\} \subset \{1, \ldots, n\}$ を表す．2 つの区間 $I = [i, j]$ と $I' = [i', j']$ について，もし $i' \leqq i$ かつ $j \leqq j'$ であるとき，$I$ **は** $I'$ **に包含される**といい，$I \subseteqq I'$ と書く．また，$I \subseteqq I'$ かつ $I' \not\subseteqq I$ であるとき，$I \subset I'$ と定義する．

b） 直列エピソードと頻出エピソードマイニング問題

**直列エピソードパターン**(以降，単に**エピソード**もしくは**パターン**と呼ぶ) は $S$ 中の窓にはめ込むことの出来るイベント列である．形式的に，直列エピソードは窓の中に出現する部分列 $\alpha = a_1 \ldots a_m$ である．ここで $m$ を $\alpha$ の大きさとし，$m = |\alpha|$ と定義する．

窓 $W_i = W[i..i + win - 1]$ について，正の整数のシーケンス $i \leqq \phi(1) \leqq \cdots \leqq \phi(m) \leqq i + win - 1$ が存在し，任意の $k = 1, \ldots, m$ について $a_k = S[\phi(k)]$ ならば，エピソード $\alpha$ は窓 $W_i$ に出現するという．

直列エピソード $\alpha$，イベント列 $S$，窓幅 $win$ が与えられ，$S$ に含まれる $\alpha$ の出現回数を

$$fr(\alpha, S, win) = |\{\, W \in \mathcal{W}(S, win) : \alpha \text{ occurs in } W \,\}|.$$

と定義する．正の整数 $\sigma > 0$ を考える．もし $fr(\alpha, S, win) \geqq \sigma$ であるとき，$\alpha$ は**頻出**であるといい，$\sigma$ を**最小頻度しきい値**と呼ぶ．ここでの問題は，$S$ に含まれる全ての頻出な直列エピソードを列挙することである．$S$ と $win$ と $\sigma$ における頻出エピソードの集合を**頻出エピソード集合**と呼び，$\mathcal{F}(S, win, \sigma)$ と書く．

［定義 1］ イベント列 $S$，窓幅 $win$，頻出度のしきい値 $\sigma$ が与えられたとき，**頻出エピソードマイニング問題**とは全ての頻出な直列エピソード $\mathcal{F}(S, win, \sigma)$ を列挙することである．

**例:** イベント列 $S = ABABBCAD$，窓幅 $win = 4$，最小頻度しきい値 $\sigma = 2$ が与えられたとき，$\alpha = ABB$ は 2 つの最小出現区間 $[1, 4], [3, 5]$ を持ち，窓 $[1, 4]$ と $[2, 5]$ と $[3, 6]$ で出現している，一方 $[4, 7]$ と $[5, 8]$ では出現していない．これより，$\alpha$ の頻度は $fr = 3$ となる．

## 3. 提 案 手 法

本節では，音響信号データ（audio music signal）を離散化

し，頻出エピソードマイニングを用いて楽曲中で繰り返し出現するパターンを抽出する具体的方法について説明する．以下は，提案手法の大まかな手順である．

（1） まず，音響信号に対して，ビートトラッキングを用いて拍節間隔を計算する．そして，得られた拍節間隔毎に音響信号のクロマベクトルを計算し，12 次元の特徴ベクトル列へと変換する．

（2） 1で得られたクロマベクトル列から，音符列に相当する離散的なイベント列へと変換する．

（3） 上記のイベント列を，各音のオンセット情報のみを残したイベント列へと整形する．

（4） 得られたイベント列に対して頻出エピソードマイニングを行い，頻出エピソードを列挙する．

（5） 列挙されたエピソードにスコア付けを行う．このときスコアとしては，なるべく長いエピソードで，かつ頻出すぎないものを高くスコア付けする．

このようにして得られたパターンは，楽曲中で繰り返されるフレーズを含んだものであると考えられる．以下では，上述した各処理の詳細について述べる．

a） 音響信号から拍間隔のクロマベクトルを抽出する

音響信号から音のイベント系列に変換する前段階として，クロマベクトル列へ変換を行う．音響信号からのクロマベクトルへの変換は，楽曲理解のためのオーソドックスな手法である．例えば 1/20 秒毎のような短い間隔ごとにフーリエ変換を行うことで，時間的に等間隔なクロマベクトル列を計算することが多い．その場合，テンポの揺れに対してロバストなマッチングを行うために，DTW 法に類する近似マッチングが施される．Ellis ら [11] は，より強力にテンポ変化に対応するために，ビートトラッキングを前処理として行い，得られた拍単位でクロマベクトルを求める手法を提案している．それによりメロディラインを認識しやすくなるため，カバー曲のような，かなりテンポが異なる楽曲でのマッチング精度が向上することが示されている．我々の方法では，Ellis らの手法を模倣し，ビートトラッキングによって得られた拍間隔でクロマベクトルを計算する．ビートトラッキングについては，今回, [11] のプログラムを借用した．当然ながら，ビートトラッキングに失敗するような楽曲に対しては極端に精度が下がる危険性は存在する．リアルタイムに音響信号のビートトラッキングを行うことは難しい問題であるが，静的な波形データに対してならばかなり精度良くビートトラッキングを行う技術が既に確立されている．

b） イベント列に変換する

クロマベクトルは 12 次元の実数値ベクトルであるため，そのままではエピソードマイニングアルゴリズムを適用できない．(図 1) 得られた拍時間毎のクロマベクトルから，音のイベントの列へと変換する必要性がある．ここでは，クロマベクトルの最大要素をその時間における代表的なイベントとして採用する．すなわち，各拍時間において，12 半音階成分の中で最も大きい成分の値を 1 とし，それ以外の要素を 0 とする．この手法で

| time | A | A# | B | C | C# | D | D# | E | F | F# | G | G# |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.00023 | 0.00580 | 0.00087 | 0.01232 | 0.00005 | 0.00454 | 0.08793 | 0.00335 | 0.00012 | 0.00168 | 0.02933 | 0.00305 |
| 2 | 0.00072 | 0.00929 | 0.00038 | 0.00872 | 0.00052 | 0.00253 | 0.08039 | 0.00114 | 0.00011 | 0.00067 | 0.01624 | 0.00380 |
| 3 | 0.00636 | 0.00338 | 0.00019 | 0.10357 | 0.00055 | 0.00253 | 0.03384 | 0.00056 | 0.00054 | 0.00066 | 0.01569 | 0.00046 |
| 4 | 0.00013 | 0.00321 | 0.00002 | 0.05812 | 0.00037 | 0.00227 | 0.02513 | 0.00013 | 0.00007 | 0.00022 | 0.01508 | 0.00117 |
| 5 | 0.00011 | 0.00366 | 0.00004 | 0.02786 | 0.00006 | 0.00088 | 0.08541 | 0.00011 | 0.00016 | 0.00016 | 0.01616 | 0.00061 |
| 6 | 0.00024 | 0.00348 | 0.00017 | 0.01649 | 0.00002 | 0.00091 | 0.06744 | 0.00020 | 0.00009 | 0.00032 | 0.01167 | 0.00136 |
| 7 | 0.03491 | 0.00185 | 0.00010 | 0.00654 | 0.00002 | 0.00060 | 0.02932 | 0.00008 | 0.06098 | 0.00018 | 0.00519 | 0.00017 |
| 8 | 0.00141 | 0.00159 | 0.00010 | 0.00570 | 0.00002 | 0.00062 | 0.01785 | 0.00016 | 0.03894 | 0.00009 | 0.00821 | 0.00127 |
| 9 | 0.01010 | 0.00737 | 0.00018 | 0.00856 | 0.00140 | 0.10952 | 0.00380 | 0.00025 | 0.04350 | 0.00315 | 0.00092 | 0.00119 |
| 10 | 0.00800 | 0.01547 | 0.00017 | 0.00341 | 0.00019 | 0.06139 | 0.00094 | 0.00001 | 0.03147 | 0.00104 | 0.00145 | 0.00082 |
| 11 | 0.03832 | 0.00109 | 0.00011 | 0.00404 | 0.00001 | 0.09515 | 0.00008 | 0.00008 | 0.02172 | 0.00025 | 0.00050 | 0.00020 |
| 12 | 0.00440 | 0.00632 | 0.00005 | 0.00084 | 0.00004 | 0.07427 | 0.00019 | 0.00030 | 0.01622 | 0.00038 | 0.00094 | 0.00083 |
| 13 | 0.00473 | 0.00537 | 0.00006 | 0.00232 | 0.00008 | 0.04612 | 0.00003 | 0.00001 | 0.02315 | 0.00027 | 0.00040 | 0.00012 |
| 14 | 0.00465 | 0.00768 | 0.00004 | 0.00188 | 0.00004 | 0.02190 | 0.00043 | 0.00005 | 0.01749 | 0.00105 | 0.00078 | 0.00103 |
| 15 | 0.04294 | 0.00016 | 0.00027 | 0.00120 | 0.00004 | 0.01512 | 0.00002 | 0.00017 | 0.00954 | 0.00003 | 0.07414 | 0.00019 |
| 16 | 0.00342 | 0.00086 | 0.00038 | 0.00207 | 0.00056 | 0.07436 | 0.00049 | 0.00010 | 0.01004 | 0.00020 | 0.02901 | 0.00079 |
| 17 | 0.00047 | 0.00339 | 0.00013 | 0.18023 | 0.00007 | 0.03418 | 0.02470 | 0.00560 | 0.00046 | 0.00146 | 0.01275 | 0.00712 |
| 18 | 0.00067 | 0.00348 | 0.00003 | 0.13137 | 0.00000 | 0.00821 | 0.02892 | 0.00088 | 0.00016 | 0.00039 | 0.00543 | 0.00435 |
| 19 | 0.06254 | 0.00205 | 0.00041 | 0.08137 | 0.00004 | 0.00572 | 0.01801 | 0.00030 | 0.00057 | 0.00020 | 0.00614 | 0.00031 |
| 20 | 0.00188 | 0.00181 | 0.00003 | 0.06221 | 0.00010 | 0.00229 | 0.02063 | 0.00024 | 0.00008 | 0.00012 | 0.00298 | 0.00521 |

図 1 クロマベクトルの列

は，メロディーラインが存在しないような間奏の部分や，楽曲中で休符になっている箇所でもイベントが生起しているように変換されてしまう．つまり，楽曲中の音量が大きい箇所で 2 つの音が同じくらいのレベルで鳴っていた場合，片方は 0 として切り落とされてしまうが，楽曲中の音量が小さい場所では，先ほど切り落とされた片方の音よりもレベルが低い音が採用されうる．また，各時点において 1 音しか採用されないため(注1)和音を取りこぼすうえ，主題の旋律を構成する音すら取りこぼす可能性がある．このような理由から，イベント列を求める手法として，この手法はベストではないかもしれない．しかし，いま我々は，正確な主旋律やベースを抽出することを目的とはしていない．すなわち，refrain を含むエピソードが抽出できればよく，そのエピソード自体が主旋律やベースを完全に表している必要はない．実際，4 節で示すように，この手法でも一定の成果が認められる．イベント列への変換の仕方としては，全体の音量レベルを考慮したしきい値を定めて，それ以上を 1，それ以下を 0 とする方法も考えられる．

c） オンセット情報を抽出する

次に，連続して鳴っている音，つまり，2 つ以上の連続した拍時間上で 1 となっている成分が存在する場合，先頭の拍時間を代表として 1 のまま残し，それ以降の部分を 0 とする．これは，長い音に対してそのオンセットのみを残して，それ以降をカットするという操作に相当する．このような変換を行う理由は，頻出エピソードマイニングにおいては，長い音を一つの持続音として取り扱うことができずに拍時間毎に生起する連続イベントとして処理してしまうためである．この処理を行わずに頻出エピソードマイニングを行うと，長い音に対応する連続したイベント列に対応する価値の低いエピソードが大量に検出されてしまい，本来発見したい特徴的な refrain が埋もれて発見しにくくなってしまう．音の開始時点だけをイベントとして取り扱うことで，検出される頻出エピソードの数を格段に絞り込むことができる．当然，この変換を行うことによって，本来は持続音ではなく拍間隔で連続していた発音を誤って取り除いてしまうことになる．しかしながら，上でも述べたが，我々が今回の求めたいものは refrain の完全なメロディラインではない．refrain を含む特徴的なパターンが抽出できればよいので，多少

(注1)：いくつかの成分の値が互いに等しい場合には複数のイベントが残されるが，成分が全く等しい値になることは極めてまれである

| time | A | A# | B | C | C# | D | D# | E | F | F# | G | G# |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

図 2　イベント列
各行が一つのイベントを表している．縦方向が時間である．1 が立っている箇所に対応する音がイベントとして生起している．

のリズムやメロディーライン情報の欠落を許容する．

以上の手順から，音響音楽信号から，最終的に図 2 のような離散的なイベントの列が得られる．

図 2 の例では，イベント列は，D#,-,C,-,D#,-,F,-,D,-,-,-,-,-,-,G,D,C,-,-,-,... となっている．ここで “-” はイベントが存在していない部分である．

d） 頻出エピソードマイニングを行い，頻出エピソードを列挙する

得られたイベント列に対して，頻出エピソードマイニングを行う．この処理には，エピソードの最小出現頻度と最大出現頻度，窓幅を設定してエピソードの発見を行う．ごく僅かしか楽曲中に出現しないイベント列は refrain ではないが，一方であまりに大量に出現するエピソードも，ドラムパターンの一部のような短く価値の低いパターンであるため，我々が欲するものではない．そのため，最小出現頻度だけではなく，最大出現頻度によっても検出するエピソードの足きりを行っている．節 2. で述べた頻出エピソードマイニングのアルゴリズムでは，あるエピソードに対してそれを含む窓の数でエピソードの頻度をカウントするため，楽曲中に 2・3 か所しか出現しないエピソードであってもその頻度は数倍に膨れ上がることに注意する．ここで設定する窓幅は，発見したい refrain の最大の長さに等しい．そのため，窓幅があまり短いとエピソードが refrain のごく短い部分しかとらえられない．逆に窓幅が長すぎると，頻出なエピソードが増えすぎるため，計算時間が多くかかる．窓幅としては，1〜3 小節分ぐらいが適切であると考えられる．

e） スコア付けを行い，スコアに従って並び替えをする

頻出エピソードマイニングによって検出されたエピソードのうち，ある程度以上の頻度で検出されたものは，それ自身の一部であるより短いエピソードも同時に検出されうる．そのような短いエピソードは冗長であり，重要なエピソードは，出現箇所が等しいもののなかで最長のものである．また，我々が欲するエピソードは，できるだけ長いものである．よって，短く大量に検出されるエピソードを振るい落とし，頻度が大きすぎず，できるだけ長いエピソードを絞り込む必要がある．

検出されたエピソードの集合を $\Pi$ とする．$\Pi$ 全体に対するエピソード $\mathrm{P} \in \Pi$ の相対的な頻度 $f_{\mathrm{P}}$ を，

$$\hat{f}_P = \frac{f_P}{\sum_{P \in \Pi} f_P}, \tag{1}$$

と定める．ここで，$f_P$ は，エピソード $P$ の頻出エピソードマイニングの結果による頻度である．このとき，エピソード $P$ は，情報量 $-\log_2 \hat{f}_{\mathrm{P}}$ を持っていると考えることができる．そこで，各エピソード $P$ のスコア $S(\mathrm{P})$ を以下のように定義する．

$$S(P) = l_P \times (-\log_2 \hat{f}_P), \tag{2}$$

ここで，$l_P$ は，エピソード $P$ の長さである．この式 2 で求めたスコアを元に，各エピソードの順位付けを行いソートを行う．その結果得られた上位のエピソードは，楽曲の繰り返し部分を表す特徴を含んでいるエピソードであると考えられる．

## 4. 実　　験

本節では，予備的な実験により，3. 節で述べた手法によって得られるエピソードが，実際に楽曲中の繰り返し部分を捉えていることを示す．以下に評価実験の方法について述べる．

まず，提案手法によって楽曲中から頻出エピソードを抽出する．アルゴリズムは [12] を用いた．スコア上位のエピソードに対して，楽曲中にそれが出現している範囲を特定し，各時点においていくつのエピソードが重複するかをプロットした．頻出エピソードマイニングアルゴリズムの性質から，楽曲中の refrain 部分には，複数のエピソードが山のように重なって出現すると考えられる．その結果得られたグラフが示す山の部分を聞き取り，実際に楽曲の繰り返し部分であるかどうかを調べた．

また，naive な手法として，頻出な $n$-gram を同様に抽出して比較を行った．ここで扱う $n$-gram とは，曲中に連続する $n$ 個の音（イベント）の連なりである．我々の行った離散化によると，ある時間においてはイベントが生起していない場合も存在するが，そうした無音部分は無視して，連続した $n$ 音を $n$-gram とした．例えば，CD-E-DC に含まれる 3-gram は，CDE,DED,EDC の 3 種である．ここで，-はイベントが無い部分を示す．

この実験では，RWC Music Database [13] のポピュラー楽曲より 10 曲を用いた．それぞれの楽曲は，モノラルの wav データ形式を用いており，楽曲の時間は最も長いもので 5：01 である．ビートトラッキングを行い，得られた拍間隔毎にクロマベクトルを抽出するにあたって，Ellis らによって提供されている Matlab のプログラムを用いた．(注2) ビートトラッキングの結果，得られた拍間隔はいずれも 8 分間隔であった．よって，頻出エピソードマイニングに渡すパラメータは，最小検出頻度 16，最大検出頻度 32，窓幅 16 とした．提案手法で得られたエピソードの上位 10 個を用いてプロットした．

結果，10 曲中 7 曲について refrain 部分が抽出できていることを確認した．図 3 は，RWC Music Database の RWC-MDB-P-2001 No.2 の楽曲における上位 10 個のエピソードである．ま

(注2)：http://labrosa.ee.columbia.edu/projects/coversongs/

| ranking | episode patterns |
|---|---|
| 1 | (F)(G)(F)(D#)(C)(A#)(C)(D#) |
| 2 | (D#)(C)(A#)(G)(F)(D#)(F#)(D#) |
| 3 | (G)(D#)(C)(A#)(C)(G)(F)(D#) |
| 4 | (D#)(F)(G)(F)(D#)(C)(A#)(C) |
| 5 | (F)(D#)(C)(A#)(C)(D#)(F)(D) |
| 6 | (G)(D#)(C)(A#)(C)(D#)(F)(D) |
| 7 | (G)(F)(D#)(C)(A#)(C)(D#)(F) |
| 8 | (F)(F)(G)(D#)(C)(D#)(F) |
| 9 | (F)(D#)(D)(F)(D#)(D)(D#) |
| 10 | (F)(D#)(C)(D#)(F)(D#)(A#) |
| 11 | (F)(D#)(C)(G)(D#)(F)(D#) |

図 3　抽出した上位 10 個のエピソード

た，図 4 は，そのエピソードの重複度合いをプロットしたものである．見てのとおり，楽曲中の一部の狭い範囲にグラフの山が集中しており，実際にこの山が高く表れている範囲は，楽曲中で繰り返し表れている箇所を示していた．一方，図 5 は，頻出する 3-gram をパターンとして上位 10 個の重複をプロットしたものである．楽曲の全体にわたってパターンが分布しており，これから refrain な部分を見て取ることは困難である．

同様に，図 6 はそれぞれ，頻出する 5-gram，7-gram，9-gram をパターンとしたプロットである．5-gram は，比較的提案手法と似たような山が表れており，多少のズレは存在しているが上手くパターンを抽出できている．7-gram と 9-gram は，3-gram とは逆に少しの山しか表れておらず，その他に存在している繰り返し部分を抽出できていない．

図 9 は，提案手法ではうまく refrain 部分を抽出できなかった例である．図のグラフで，山が高くなっている部分は曲の前奏部分にて主題が演奏されている部分であった．しかし，その後の曲中における歌声を伴った主題部分を取りこぼしている．我々の手法では，信号を離散化する際に，各クロマベクトルの最大要素をその時間における代表的なイベントとして採用した．それにより，この例においては，同じ主題部分であっても歌声のある無しでかなり異なるイベント列へと変換されてしまっていた．その他にも，ドラムパターンの違いやベースの音量の違いなどの要因も考えられる．

## 5. おわりに

本稿では，音響信号を対象に，頻出エピソードマイニングに基づいて曲の refrain を抽出する手法について述べた．エピソードを用いることで，入力データの不均質さやテンポの揺れを吸収し，上手く refrain を抽出することができた．このように，音響信号データに対して記号処理的なアプローチを行う研究はあまり例がなく，エピソードを活用した本手法に関しては，我々が知る限り初めてのものである．この手法によって抽出されたエピソードは，楽曲の特徴的な部分を記号的に表現している．これらを用いることで，従来実現が困難であった，広い分野にまたがってアレンジされた楽曲の検索・抽出に応用できる

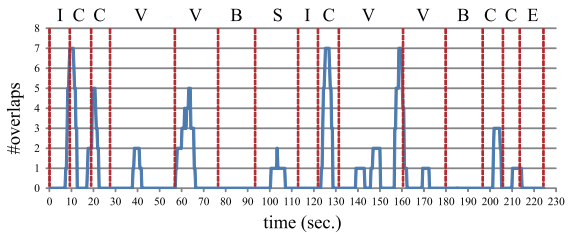


図 4　エピソードの重なりの数
ここの範囲に付けられたタグ I, C, V, B, S, E, はそれぞれ, introduction, chorus, verse, bridge, solo, ending である．

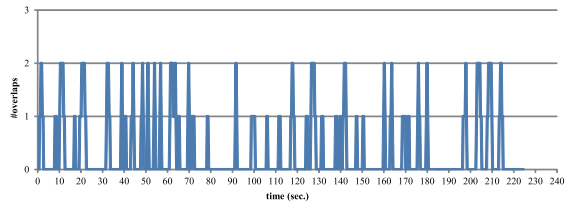


図 5　3-gram の場合

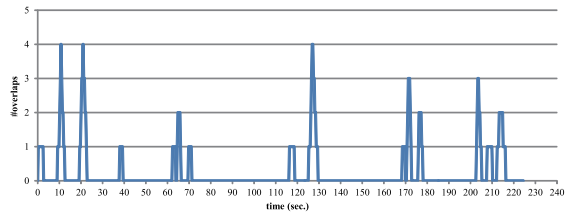


図 6　5-gram の場合

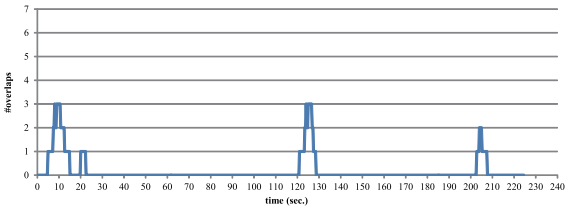


図 7　7-gram の場合

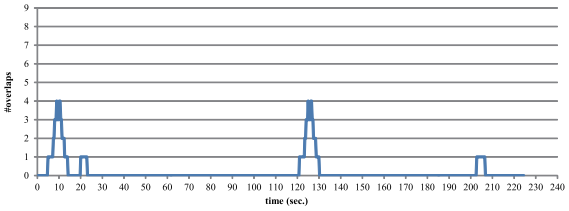


図 8　9-gram の場合

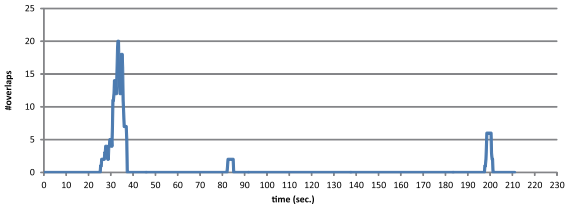


図 9　失敗した例

可能性がある．

また，我々は，提案手法が価値のあるパターンを抽出できているかどうかを確かめるため，予備的な前実験を行い，頻出エピソードマイニングを用いて抽出したパターンが，楽曲の refrain 部分を良く表していることを確認した．

今回，頻出エピソードの発見手法については，頻出直列エピソードを列挙するアルゴリズム [12] を用いたが，我々の目的からすると，飽和パターンを列挙することのできるアルゴリズムのほうが望ましい．また，直列エピソードだけでなく，和音を意識したエピソードパターンを用いることも考えられる．さらに今後の課題として，精度と再現率を求め, [14] のような既存手法との比較を行う必要がある．

## 文　献


[1] G. Masataka, “A chorus-section detecting method for musical audio signals,” in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 2003, pp. V–437–440.

[2] H. Nagano, K. Kashino, and H. Murase, “Fast music retrieval using polyphonic binary feature vectors,” in *2002 IEEE International Conference on Multimedia and Expo (ICME’02)*, vol. 1, August 2002, pp. 101–104.

[3] M. Casey and M. Slaney, “The importance of sequences in musical similarity,” in *2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP’06)*, vol. 5, May 2006, pp. V–5–V–8.

[4] T. Izumitani and K. Kashino, “A robust musical audio search method based on diagonal dynamic programming matching of self-similarity matrices,” in *9th International Conference on Music Information Retrieval (ISMIR’08)*, September 2008, pp. 609–613.

[5] A. Lubiw, “Pattern matching in polyphonic music as a weighted geometric translation problem,” in *5th International Conference on Music Information Retrieval (ISMIR’04)*, October 2004, pp. 289–296.

[6] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, “Evaluation of real-time audio-to-score alignment,” in *8th International Conference on Music Information Retrieval (ISMIR’07)*, September 2007, pp. 315–316.

[7] B. Niedermayer, “Towards audio to score alignment in the symbolic domain,” in *6th Sound and Music Computing Conference*, July 2009, pp. 77–82.

[8] C. Joder, S. Essid, and G. Richard, “An improved hierarchical approach for music-to-symbolic score alignment,” in *11th International Society for Music Information Retrieval Conference (ISMIR’10)*, August 2010, pp. 39–44.

[9] M. Bartsch and G. Wakefield, “To catch a chorus: using chroma-based representations for audio thumbnailing,” in *2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 2001, pp. 15–18.

[10] H. Mannila, H. Toivonen, and A. Inkeri Verkamo, “Discovery of frequent episodes in event sequences,” *Data Min. Knowl. Discov.*, vol. 1, pp. 259–289, January 1997.

[11] D. Ellis and G. Poliner, “Identifying ‘cover songs’ with chroma features and dynamic programming beat tracking,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, April 2007, pp. IV–1429–IV–1432.

[12] K. Takashi and H. Kouichi, “A simple characterization on serially constructible episodes,” in *12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2008.

[13] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “Rwc music database: Popular, classical and jazz music databases,” in *3rd International Conference on Music Information Retrieval (ISMIR)*, 2002.

[14] R. J. Weiss and J. P. Bello, “Identifying repeated patterns in music using sparse convolutive non-negative matrix factorization,” in *11th International Society for Music Information Retrieval Conference (ISMIR’10)*, August 2010, pp. 123–128.