Vol.2017-MUS-116 No.15
2017/8/25

音響的に自然なつなぎ目の発見による楽曲ループ検出

安井 拓未[†1,a)]　中村 篤祥[1,†1,b)]　田中 章[1,†1,c)]　工藤 峰一[1,†1,d)]

概要：音響的に類似している部分が周期的に連続して現れることを用いて，楽曲 PCM ファイルからループを検出する手法を提案する．ゲーム音楽用 MIDI ファイルを WAVE 形式に変換したファイルを用いた実験によれば，再生時間の 1/7 の時間で，222 曲中 199 曲で聴覚的に違和感のないつなぎ目で繋げてループさせることに成功した．

Yasui Takumi[†1,a)]　Nakamura Atsuyoshi[1,†1,b)]　Tanaka Akira[1,†1,c)]　Kudo Mineichi[1,†1,d)]

1. はじめに

ゲーム用 BGM はその目的上同一シーン中で音楽を途切れさせないようなループ構造を持っており，ループ状に再生することが基本的に行われている．一般的に同じゲームでは同じ BGM が流れるが，人により音楽の好みが異なるため，その人の好みの曲を BGM に使うことができれば，よりゲームを楽しむことができるであろう．MIDI ファイルを使った電子音での再生では，ループ再生は容易であるが，その人の好みの音楽は WAVE 形式などの PCM ファイルでしか手に入らないことがほとんどである．そのような PCM ファイルからループを検出して自然に繋がる位置を見つけることができれば，ゲーム用 BGM として用いることが可能となる．

本研究では，音響的に類似している部分が周期的に連続して現れることを用いて，楽曲 PCM ファイルからループを検出する手法を提案する．提案手法では，ビートトラッキングシステムを利用してビート位置を検出して，ビート位置のみの部分列に対して周期的な類似部分の検出を行うことにより，高速にループ候補を検出する．また，つなぎ目候補点近辺の，波形が負から正に変わるゼロクロスサンプル点を最終候補にすることにより，クリックノイズが生じない，自然なつなぎ目発見を実現する．ゲーム音楽用 MIDI ファイルを WAVE 形式に変換したファイルを用いた実験によれば，222 曲中 199 曲で聴覚的に違和感のないループのつなぎ目を，平均的に楽曲再生長の約 1/7 の時間で発見することに成功した．

関連研究

従来の研究として，Ong&Streich は音楽特徴として，ピッチクラス分布特徴を利用して，ループ区間をパターン認識を応用して発見し，発声ノート上の点上での類似性によってループの地点を発見する方法で，著者らのデータセット上において 85%の精度でよいループが発見できたとしている [1].

2. 楽曲 PCM ループ発見問題

2.1 問題設定

与えられた楽曲 PCM データに対し，曲の構成上の最も大きな基本周期の繰り返し位置を自動的に検出する問題を考える．具体的には以下のような問題を考える．入力として，サンプリングレート f_s でサンプリングされた楽曲 PCM データ $X = (x_1, \ldots, x_{N_X})$ が与えられる．ある $a > 1$ に対し，周期 p の繰り返しが時点 i から時点 $i+(a-1)p-1$ まで続くとき，この部分を周期 p のループ回数 a のループと呼ぶ．曲の構成上の最も大きな基本周期 p のループが時点 i から始まるループ回数 a のループとすれば，周期 p の繰り返しにおいて対応する時点のペア $(k, k+p)$ $(i \le k \le i+(a-1)p-1)$ を 1 つ見つける問題 (**楽曲 PCM ループ発見問題**) を扱う．ただし，ループ回数 a は 1 より大きな値 $a_{\min}$ 以上のループのみ対象とする．

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，北海道大学
Presently with Hokkaido Uniersity
[a)] yasu1@main.ist.hokudai.ac.jp
[b)] atsu@main.ist.hokudai.ac.jp
[c)] takira@main.ist.hokudai.ac.jp
[d)] mine@main.ist.hokudai.ac.jp

2.2 ビート検出

楽曲 PCM データを扱う場合，サンプリング周波数は CD では 44.1kHz が用いられ，23 秒を超える楽曲 X では長さ N_X は 100 万を超えてしまう．長いデータに対して，楽曲 PCM ループ発見問題の解を直接求めるのは計算量が大きく，実用的ではない．そこで，$X = (x_1, \ldots, X_{N_X})$ のビートの位置のリスト $X_B = (X_{b_1}, \ldots, X_{b_{N_B}})$ を，ビートトラキングシステムを用いて求め，X の部分列である X_B に対して楽曲 PCM ループ発見問題の解を求める方法を考える．ビート位置は繰り返しの対応時点としても適切であるため，繰り返しの検出の精度向上の効果も期待できる．ビートトラッキングシステムには，リアルタイム性を重視したオンライン型のシステム (例:IBT[2]) と，ビート検出精度を重視したオフライン型のシステム (例:Essentia[3]) が存在し，検出速度と検出精度はトレードオフの関係があるので，目的に応じたシステムを選択する必要がある．

2.3 音響的類似性

MIDI データや楽譜データと異なり，繰り返しにおいて対応するする時点のペア $(k, k+p)$ に対して，その時点の値が完全に一致 $(x_k = x_{k+p})$ するとは限らない．そこで楽曲 PCM データ X 内の 2 時点 i, j の音響的な近さを測る指標 $D_X(i,j)$ が指定された値 $\epsilon > 0$ より小さいとき，x_i と x_j は近似的に等しいと言い $x_i \overset{\varepsilon}{\simeq} x_j$ と表記する．本稿では $D_X(i,j)$ として以下のものを用いる．

$$D_X(i,j) = \frac{\|F_M(X,i) - F_M(X,j)\|}{\|F_M(X,i)\| + \|F_M(X,j)\|} \tag{1}$$

ただし，$F_M(X,k)$ を，楽曲データ X の k 番目からの M サンプル $x_k, x_{k+1}, \ldots, x_{k+M-1}$ を高速フーリエ変換した結果の $(M/2)+1$ 個の周波数成分におけるパワースペクトルとし，$\|\cdot\|_1$ を L1 ノルムと定義する．分母は小さいベクトル間の方が小さくなる傾向を補正するものである．実験では $M = 8192$ を用いた．

2.4 周期的部分列

楽曲 PCM データ $X = x_1 \cdots x_{N_X}$ において，$x_i \overset{\varepsilon}{\simeq} x_{i+p}$ $(i = j, \ldots, j+\ell-1)$ を満たすとき $x_j \cdots x_{j+\ell-1}$ は X の**周期 (period)**p の**周期的 (periodic)** 部分列であると言い，ℓ を周期的部分列 $x_j \cdots x_{j+\ell-1}$ の長さと定義する．本稿では，以下の問題を解くアルゴリズムを利用して楽曲 PCM ループ発見問題に有効な方式を提案する．

問題 1 (最長周期的部分列問題) 楽曲 PCM データ $X = x_1 \cdots x_{N_X}$ に対し，長さが最大の周期的部分列を 1 つ求めよ．

2.5 音響的に自然なつなぎ目

本稿では，音響的類似性に着目し，類似な部分が周期的に現れることを検出してループを見つける方法を考えるが，実際に見つかった時点を繋げて再生する場合には，つなぎ目が自然に聴こえるような注意が必要である．

人間の聴覚が音波信号の絶対位相を認識することは難しいとされるが，不適切な位置で繋ぐと，波形の急激な変化が出力され，クリックノイズとして認識されるという問題がある．これに対処するため位相の連続性について考慮し，X の波形が負から正へと変わる瞬間 (ゼロクロスサンプル点) の集合 $S = \{k \mid x_{k-1} < 0 < x_k\}$ のみを，最終的なつなぎ目 (楽曲 PCM ループ発見問題の解) の候補とする方法をとる．これによって得られた解の 2 時点を繋げた場合のクリックノイズ発生を抑制することができる．

3. 提案手法

以下の手順で，与えられた楽曲 PCM データ $X = (x_1, \ldots, x_{N_X})$ に対し，楽曲 PCM ループ発見問題の解の候補を見つける方法を提案する．

Step 1:データ X にビートトラッキングを適用し，得られたビート時点リストを $B = (b_1, \ldots, b_{N_B})$ とし，X の対応する部分列を $X_B = (x_{b_1}, \ldots, x_{b_{N_B}})$ とする．

Step 2:X_B に対する最長周期的部分列問題を解き，解の開始位置を j とし，その周期を p とする．

Step 3:以下の式を満たす (m^*, n^*) を楽曲 PCM ループ発見問題の解の候補とする．

$$(m^*, n^*) = \underset{m \in Z_j, n \in Z_{j+p}}{\arg\min} \|F_{256}(X,m) - F_{256}(X,n)\| \tag{2}$$

ただし，$Z_k = \{i \mid b_j < i < b_j + 1, x_{i-1} < 0 < x_i\}$ とする．

Step 1 は 2.2 節で説明したように，適切に候補を絞ってループの検出精度をあげる目的の他，Step 2 で解く最長周期的部分列問題の入力を短くし，計算時間を短縮する効果もある．Step 3 は 2.5 節で説明したように，Step 2 で最長周期的部分列問題を解くことにより見つかった解 (b_j, b_{j+p}) を，自然に繋がるように微調整する役割をもつ．

補注 1 繰り返しの対応する部分がすべて近似的に等しいと判定されると仮定する．残念ながら，最長周期的部分列問題の解が基本周期 p が最大のループになるとは限らない．例えば，重ならない 2 つの周期的部分列で短い周期の部分列の方が長さが長いということはありうる．しかし，基本周期が最大のループは，ループ回数 a がある程度 1 よりおおきければ，長さが長い周期的部分列になるので，楽曲 PCM データにおいて最長周期的部分列になる可能性が高いと考える．ループ回数 a が多い場合には，整数倍の周期の周期的部分列にもなるが，周期が短い方が長さが長くなるため，基本周期の整数倍周期の周期的部分列は最長周期的部分列問題の解とはならない．

Algorithm 1 最長周期的部分列問題のアルゴリズム

Algorithm MaxPeriodicSubseq(x)
Input: $X = x_1 \cdots x_n$ ▷ 楽曲 PCM データ (の部分列)
Output: (p^*, b^*, ℓ^*) ▷ 最長周期的部分列の (周期, 開始位置, 長さ)

```
p* ← 0
for p = 1 to n − p* do
  b ← 1
  for i = 1 to n − p do
    if x_i ≁^ε x_{i+p} then
      ℓ ← i − b
      if ℓ > ℓ* then
        (p*, b*, ℓ*) ← (p, b, ℓ)
      end if
      b ← i + 1
    end if
  end for
  ℓ ← n − p* + 1 − b
  if ℓ > ℓ* then
    (p*, b*, ℓ*) ← (p, b, ℓ)
  end if
end for
return(p*, b*, ℓ*)
```

3.1 最長周期的部分列問題のアルゴリズム

Algorithm 1 に，長さ n の楽曲 PCM データ X に対して最長周期的部分列を見つける素朴なアルゴリズム MaxPeriodicSubseq を示す．最悪の場合，各々の周期 p に対して，各々の時点 i の値 x_i を p だけずらした時点の値 x_{i+p} が近似的に等しいかをすべての異なる時点のペアに対して行うので，$n(n-1)/2$ の定数倍の計算時間で行うことができる．

4. 実験

4.1 データ

ループのある MIDI ファイルを WAVE ファイルに変換したものを使用する．PC 用同人ゲーム FF3.5 ver0.9922(`http://ff35.exblog.jp`) 内に含まれる MIDI ファイルのうち，実際に楽曲途中にループ利用がされている 222 曲 (表 1) を対象とした．対象のファイルはサンプリング周波数が 44.1KHz の WAVE ファイルへと変換したものを使用した．サンプリング周波数より平均的に 635 万程度の長さの楽曲 PCM データとなっている．

表 1 実験に用いた MIDI ファイル (FF3.5ver0.992 ループ有)

楽曲数	222
平均楽曲長 [秒]	144

4.2 評価方法

繰り返し単位が最大の繰り返し部分に対応するの WAVE ファイル X における周期的部分列の (周期, 開始位置, 長さ)$= (p, b, l)$ を，対応する元の MIDI ファイルから計算する．楽曲 PCM ループ発見問題の解の候補 (m^*, n^*) に対し，$p = n^* - m^*, b \leq m^*$ 及び $n^* < b + l$ を満たす場合に正解と判断し，正解率で評価する．また，アプリケーションによっては $n^* - m^*$ が p の整数倍の場合が検出されても問題ないものもあるのでその場合の正解率も評価する．

4.3 計算機環境

計算機環境は，以下のものを利用した．

表 2 計算機環境

CPU	Core i7-6700 @ 3.4GHz
メモリ	DDR3-1600 8GB
OS	Windows 10 Pro (1703)
コンパイラ	Visual Studio 2017 Enterprise C++

4.4 結果

表 3 は，オフライン型の Essentia[3] とオンライン型の IBT[2] という 2 つのビートトラッキングシステムをビート検出に用いて提案法を実行した場合の正解率と計算時間を示したものである．参考のため，正しいビート時点リストを提案手法に与えた場合の正解率をビート位置が「既知」の場合として表に載せている．

Essentia と IBT では，Essentia の方が 5 倍近い計算時間を要するが，その分提案法に適用した場合の正解率はよく，83.8%(整数倍可の場合は 86.5%) という正解率が得られた．ビート位置が既知の場合でも 86.5%(整数倍可の場合は 91.0%) であるので，提案手法に用いるビートトラッキングシステムとして精度は十分であると考えられる．IBT では，ビートトラッキングの結果が非常に不安定であり，ビート検出の失敗が，求めたい周期の繰り返し検出の失敗に繋がったと考えられる．

Essentia も IBT も楽曲 X の長さ N_X に対して $O(N_X)$ で計算が終わることが確認できた．よって全体の計算時間は $O(N_X + {N_B}^2)$ と考えられる．

5. 考察

本研究でループ検出に失敗した例について調べたところ，3 つのパターンが存在した．

第一に，検出したいループのループ回数が 2 回未満であ

表 3 提案法の正解率と計算時間 [秒](elapsed time) : 計算時間は「平均 (標準偏差)」で表記されている．

ビート位置	Essentia で検出	IBT で検出	既知
正解率	83.8%	75.7%	86.5%
正解数/222	186	168	192
正解率 (整数倍可)	89.6%	84.7%	91.0%
正解数/222	199	188	202
Step1 計算時間	15.9(7.34)	3.24(1.58)	-
Step2 計算時間	1.29(1.59)	1.17(1.44)	1.40(2.74)
Step3 計算時間	2.73(4.17)	2.86(3.07)	2.85(3.69)
全体計算時間	19.96(9.61)	8.74(8.37)	4.80(5.27)

る場合である．本アルゴリズムでは最長周期的部分列を求める際に最も長く一致する区間を返すため，本アルゴリズムでは失敗する場合がある．

第二に，ビートトラッキングの結果が本来の BPM の 2/3 倍など，一定して誤った検出をした場合である．この場合にビートトラッキングの地点が 1 回目と 2 回目でループ上の同じ時点を指していない場合に失敗する．表 5 の失敗パターン 2 の例では，およそ 2/3 倍の BPM を定常的に取ってしまった例で，本システムが検出したループは基本周期の 3 倍となった例である．

第三に，ビートトラッキングが不安定な場合である．ビートトラッキングの結果が 1 回目のループと 2 回目のループで異なり，音楽的に一致する部分が前後にずれた結果，検出できなかったものである．表 5 の失敗パターン 3 の例では，大部分は本来の 2 倍速の BPM を検出しているが，検出された BPM が不安定のため，標準偏差が大きくなっている．失敗パターン 3 の例の検出 BPM の最大値は 167.13，最小値は 107.72 である．

以上のパターン別の失敗数 (不正解の場合分け) の結果を表 4 にまとめた．

表 4 パターン別の失敗数

ビート位置	Essentia で検出	IBT で検出	既知
失敗パターン 1	30	32	30
失敗パターン 2	4	2	0
失敗パターン 3	2	20	0

表 5 パターン別の失敗の一例と BPM の関係

	失敗パターン 2 の例	失敗パターン 3 の例
正解 BPM	(161.0005, 0)	(70.00, 0)
Essentia BPM 結果	(107.20, 1.61)	(139.62, 11.39)

(平均, 標準偏差) で表記

正しいビート位置を与えた場合の失敗例は第一パターンのみであり，ビートトラッキングシステムでの失敗例では 3 パターン全てが起こった．また，Essentia ビートトラッキングよりも IBT ビートトラッキングは不安定な結果を出力しやすいため，第三パターンの失敗が増えていた．

6. まとめ

楽曲 PCM ファイルから周期的な類似を検出することによりループを発見する方法を提案し，MIDI ファイルを WAVE 形式に変換した楽曲 PCM ファイルで高い正解率が得られることを確認した．今後は，オーケストラ等のアコースティックサンドに本手法を適用し，有効性を検証し，既存法との比較を行う予定である．

参考文献

[1] Ong, B. S. and Streich, S.: Music loop extraction from digital audio signals, *2008 IEEE International Conference on Multimedia and Expo*, pp. 681–684 (online), DOI: 10.1109/ICME.2008.4607526 (2008).
[2] Oliveira, J., Gouyon, F., Martins, L. G. and Reis, L. P.: IBT: A Real-Time Tempo and Beat Tracking System, *International Society for Music Information Retrieval Conference*, pp. 291–296 (2010).
[3] Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J. R. and Serra, X.: Essentia: An Audio Analysis Library for Music Information Retrieval, *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013*, pp. 493–498 (2013).