

ダンスゲーム譜面の特性分析とクラスタリングに基づく特徴的な譜面の自動生成

辻野 雄大^{1,a)} 山西 良典^{2,b)} 山下 洋一^{2,c)} 井本 桂右^{2,d)}

概要：ダンスゲーム譜面は、操作を要求される頻度、リズム構成の複雑さなど、複数の要素によって特徴づけられる。これらの要素によって、ダンスゲーム譜面の難しさおよび「面白さ」に影響する特性が形成される。この特性を客観的かつ多次元で表現する特徴量を提案する。提案した特徴量に基づいて k-means 法によるクラスタリングを実施し、特性が類似した譜面のクラスタを得る。クラスタ毎に楽曲と譜面の関係を深層学習させることによって、クラスタの特性を備えた譜面の自動生成が可能となり、多様な「面白さ」に対応した特徴的な譜面の提供を実現した。

キーワード：ダンスゲームの自動生成, ダンスゲームの分析, Procedural Content Generation

Generation of Characteristic Charts Based on Analyzing and Clustering Charts of Dance Games

YUDAI TSUJINO^{1,a)} RYOSUKE YAMANISHI^{2,b)} YOICHI YAMASHITA^{2,c)} KEISUKE IMOTO^{2,d)}

Abstract: Multiple factors, e.g. the frequency of action and the complexity of rhythm, may characterize the charts on dance games. Those factors affect the difficulty level and fun of the charts. This paper designs objective and multi-dimensional features concerning such factors. As conducting the k-means clustering with the designed features, the characteristically similar charts structure the clusters. The relationships between songs and charts are leaned for each cluster by the deep neural network. It was confirmed that our approach realized to automatically generate the charts with the characteristics corresponding to the learning cluster. The characteristic charts for each of diverse “fun” could be provided.

Keywords: Automatic generation of dance games, analysis of dance games, Procedural Content Generation

1. はじめに

今日のビデオゲーム市場で、大きなシェアを占めているジャンルに「音楽ゲーム」が存在する。音楽ゲームの代表例として、*Dance Dance Revolution*[®] (DDR) などのダンスゲームが挙げられる。多くの音楽ゲームでは、画面に

表示される指示符に沿って、プレイヤーが何らかの動作を行い、その動作のリズムに対する成否を競う。ダンスゲームでは、指示符は「ステップ」と呼ばれており、ある楽曲に対して用意されたステップの配置パターンは「譜面」と呼ばれている。図 1 に、ダンスゲーム画面の例を示す。

ダンスゲームのプレイヤーの中には、何年も継続して遊ぶ上級者もいれば、1 度だけ遊ぶような初心者もいる。幅広い実力のプレイヤーが面白いと感じるゲームとするためには、簡単な内容から始まり、少しずつ難しくなっていくように難易度バランスを構成することが重要である [1]。多くのダンスゲームでは、難易度の異なるたくさんの譜面を用意

¹ 立命館大学大学院情報理工学研究科

² 立命館大学情報理工学部

a) is0221rs@ed.ritsumei.ac.jp

b) ryama@media.ritsumei.ac.jp

c) yyama@is.ritsumei.ac.jp

d) k-imoto@fc.ritsumei.ac.jp

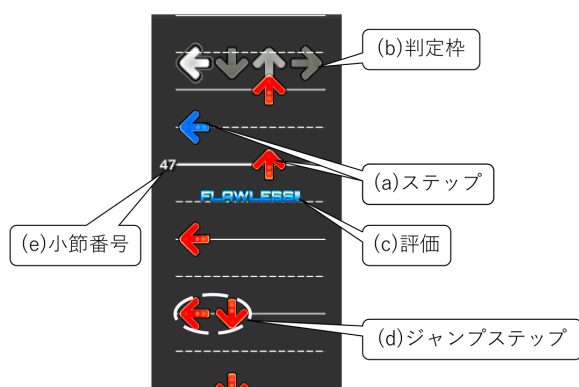


図 1 ダンスゲーム *Stepmania* におけるゲーム画面の例。流れる楽曲に合わせて (a) ステップが画面下部から上部へ移動し、(b) 判定棒とステップが重なったタイミングでプレイヤーが対応するパネルを踏むという動作を行う。(c) 評価は動作を正しいリズムで行えたかどうかを表す。ステップの中には、同一タイミングで 2 箇所以上のパネルを踏むことを指示する (d) ジャンプステップも存在する。(e) 小節番号は、その時刻が楽曲先頭から数えて何小節目にあたるかを示す。ステップの色は、そのステップが属する拍階層 (2.1.3 節に述べる) によって変化する。

することで、多くのプレイヤーが面白いと感じるよう配慮されている。譜面の難易度に影響する要素は複数あるとされている。その事例として、複数のプレイヤーのゲームログを比較したとき、ゲーム製作者によって同程度の難易度であるとラベル付けされた譜面の中で、あるプレイヤーにとって最も良いパフォーマンスを発揮できる譜面が、ほかのあるプレイヤーにとっては全くできない譜面であったりする。もしも譜面の難易度に影響する要素が 1 つだけであれば、どのプレイヤーも同じ難易度の譜面であれば同じパフォーマンスを発揮できるはずである。したがって、譜面は複数の要素によって特徴付けられ、同程度の難易度とされている譜面であってもそれぞれ異なる特性を持っていると言える。この譜面の特性は、難易度に影響するのみではなく、譜面の「面白さ」にも大きく影響する。同程度の難易度の譜面であっても、どの譜面をより面白いと感じるかはプレイヤーによって異なる。このことから、プレイヤーは譜面の特性を「難易度」という単一の基準のみではなく、「面白さ」という異なる観点でも評価していると言える。

ゲームの難易度や「面白さ」に影響する要素を調査した研究はいくつか存在している。Wehbe らは、シンプルなアクションゲームに対して、画面スクロールの速さやジャンプ動作の複雑さなどのパラメータを変化させることで難易度が変化するかを調査している [2]。Spiel らは、パズルゲーム *TETRIS* を題材にして、自明に難易度に影響するゲームの速度ではなく、ピースの選択アルゴリズムを変化させることで難易度が変化するかを調査している [3]。しかしながら、音楽ゲームの譜面の特徴、難易度と「面白さ」に影響する要素について調査した研究は、従来存在していなかった。従来の音楽ゲームに関わる主流の研究は、人類学的な研

究 [4] であるか、譜面の自動生成に関わる研究であった。音楽ゲームにおける譜面など、ゲーム内のコンテンツを自動生成する試みは Procedural Content Generation (PCG) と呼ばれ [5]、さまざまなジャンルのビデオゲームを対象に行われている。一例として、Pedersen らは、アクションゲーム *Super Mario Bros*[®] を題材に、ゲーム内の状況からプレイヤーの行動をモデル化し、それに合わせたマップを生成する手法を提案している [6]。Donahue らの *Dance Dance Convolution (DDC)* [7] は、ダンスゲームにおける PCG の代表的な研究である。彼らの研究では、データセット内の譜面全てを用いて Long Short Term Memory (LSTM) モデルを学習させることで、譜面の自動生成を実現していた。しかし、前述の通り譜面データセットの中には様々な特性を持った譜面が混在している。このような多様性のあるデータを同一のモデルに学習させると、それぞれの特性が平均化されてしまう。その結果、どのような楽曲を入力しても同じような譜面が生成されてしまう。このような生成結果では、多様な「面白さ」の感性に対して、充分に対応できていないとも言えない。また、譜面データセット内には、他の譜面とは大きく異なる極端な性質を持つものも存在する。例として、特定の区間にステップが集中して配置されている譜面、楽曲途中でテンポが大きく変化する楽曲に対して用意された譜面などが挙げられる。このような譜面はモデル学習時にはノイズとなり、学習する中でこれらの性質も含めて平均化しようとする。その結果、特殊なステップ配置が突如現れるような、中途半端で一貫性のない譜面が生成されてしまう。これらの問題を解決するためには、極端な性質を持つ譜面を除外し、譜面を特性ごとに分類した上で学習することが有効であろう。特定の特性を持つ譜面のみをモデルに学習させることによって、意図した特性を持つ譜面が生成されることが期待される。これにより、多様な「面白さ」の感性に対応した特徴的な譜面をプレイヤーに提供可能になると考えた。

本稿では、譜面の難易度と「面白さ」に影響する要素として考えられる特徴を挙げ、各特徴に基づいてゲーム譜面をクラスタリングする。このとき、極端な性質を持つ譜面を除外した上で、譜面をクラスタリングすることで、難しさや「面白さ」の傾向を把握する。そして、クラスごとに譜面生成モデルを学習させることによって、そのクラスタの特性を備えた特徴的な譜面の自動生成を実現する。

2. 環境設定

本稿では、先行研究 [7], [8], [9] にならって、ITG データセットを使用する *1。このデータセットには、オーブ

*1 “In The Groove” <http://east.stepmania-online.com/In%20The%20Groove%201.zip> および “In The Groove 2” <http://east.stepmania-online.com/In%20The%20Groove%202.zip> [最終閲覧日: 2019 年 7 月 31 日]

表 1 データセットの情報 ([7] を参考に作成)

データセット名	ITG
製作者人数	8
パックの数	2
楽曲数	133
譜面数	652
公開年	2004~2005

ソースのダンスゲーム *Stepmania* *2で遊べるように作られた 133 曲分の譜面データが含まれている。1 曲につき 4~5 種類の譜面が収録されており、難易度が低い順に Beginner, Easy, Medium, Hard, Challenge というラベルが付与されている。表 1 にデータセットの情報を示す。

著者のうち 1 名の知見に基づき、譜面データから得る特徴量を設定した。特徴量は、大きく分けて「譜面特徴量」と「楽曲特徴量」に分類される。譜面特徴量は、同じ曲に対して用意された譜面であったとしても、得られる値は譜面ごとで異なる。一方で、楽曲特徴量は、同じ曲に対して用意された譜面であれば、異なる譜面であっても同一の値が得られる。いくつかの特徴量は Donahue らの論文 [7] 中でも触れられている。特徴量の計算には、譜面中の最初のステップから最後のステップまでの部分を用いた。

2.1 譜面特徴量

2.1.1 1 秒あたりステップ数

図 2 に、ある譜面の一部とそこから得る 1 秒あたりのステップ数の例を示す。譜面を 1 秒ごとで切り分け、各部分に含まれるステップの数を数えることで、以下の特徴量を得る。

n_l 最もステップ数が少ない部分のステップ数。

n_m 最もステップ数が多い部分のステップ数。

n_μ 譜面中の全ての部分のステップ数の平均。

n_{σ^2} 譜面中の全ての部分のステップ数の分散。

2.1.2 ジャンプステップ

同一のタイミングに 2 つ以上の矢印が存在するステップがある。これはプレイヤーに指定されたすべてのパネルを両足で踏むことを要求している。このようなステップを「ジャンプステップ」と定義する。ある譜面中のジャンプステップの回数を集計し、以下の特徴量を得る。

j_r = ジャンプステップの回数/総ステップ数。

j_d = ジャンプステップの回数/譜面の長さ [秒]。

2.1.3 拍階層割合

我々の先行研究 [8], [9] にもとづいて、拍階層表現を用いる。小節を n 等分 (ただし $n \geq 4$) することで表現出来る時刻集合を「 n 分層」と定義する。あるステップが存在する時刻が属する層のうち最下層 (すなわち n が最小) の

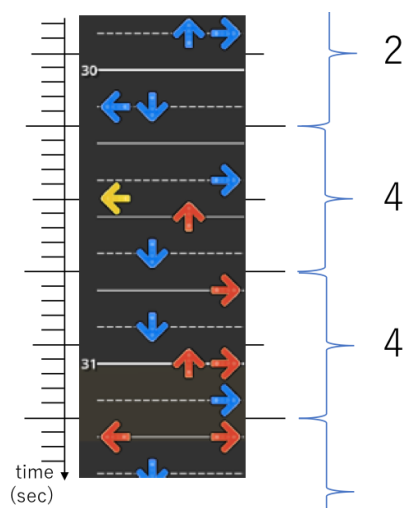


図 2 1 秒あたりステップ数の集計例

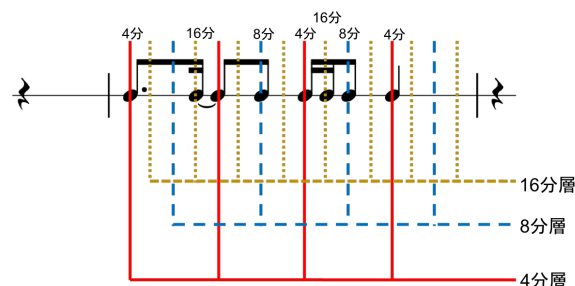


図 3 拍階層の概念。音符の上に書かれた「4 分」「8 分」が各音符の拍階層表現である。4 分層に属する音符は、8 分層や 16 分層にも属するが、その中で最下層である 4 分層に存在するものとする。

ものが l 分層であるとき、そのステップは l 分層に存在するものとし、「 l 分 (のステップ)」と表現する。*Stepmania* においては、ステップが属する拍階層によって、図 1 のように異なる色でステップが表示される。図 3 に、実際の楽器の楽譜と拍階層の概念の対応付けを示す。ダンスゲームにおける「ステップに合わせてパネルを踏む」という動作は、楽器の演奏における「音符に合わせて発音を開始する」という動作に対応する。拍階層表現は、動作を行う時刻が楽曲聴取上で知覚しやすいタイミングであるかどうかを表現している。ステップが上位の階層に属しているほど、そのステップは複雑なリズムで構成されており、知覚しにくいと言える。反対に、下位の階層に属しているステップが連続しているとき、それらのステップは知覚しやすいリズムであると言える。

譜面中の各ステップがどの拍階層に属するかを集計し、以下の特徴量を得る。

l_4 = 4 分層のステップ数/総ステップ数。

l_8 = 8 分層のステップ数/総ステップ数。

l_{12} = 12 分層のステップ数/総ステップ数。

l_{16} = 16 分層のステップ数/総ステップ数。

*2 News - Stepmania, <https://www.stepmania.com>, [最終閲覧日: 2019 年 7 月 31 日]

表 2 出現頻度上位 3 種類の 3-gram

Step 1	Step 2	Step 3	出現頻度	出現譜面数
↑	↓	→	4,457	600
↓	↑	↓	4,238	553
←	↑	↓	4,216	589

l_{24} = 24 分層のステップ数/総ステップ数.

l_{32} = 32 分層のステップ数/総ステップ数.

$l_{oth} = 1 - \sum_{q \in \{4,8,12,16,24,32\}} l_q$.

2.1.4 ステップ向き 3-gram の $tf-idf$

連続する 3 ステップの向きの組み合わせについて出現頻度を集計し、3-gram とする. 表 2 に、出現頻度上位 3 種類の 3-gram を示す. ITG データセット内には、6,910 種類の 3-gram が存在し、出現頻度の総和は 176,279 であった. このうち、総出現頻度の 90% を占める出現頻度上位 979 種類の 3-gram を、分析の対象とした. 以下、979 種類の 3-gram の集合を G とし、 $g \in G$ とする.

それぞれの $g \in G$ に対して、ある 1 譜面中の出現頻度 tf_g と、データセット内で出現している譜面の数 df_g から、数式 (1) を用いて $tf-idf$ 値を計算し、 s_g とする.

$$s_g = \frac{tf_g}{\sum_{k \in G} tf_k} \times \log_2 \frac{m}{df_g}. \quad (1)$$

数式 (1) で、 m はデータセット内の譜面の数 652 である. s_g は、 g がその譜面にどの程度固有であるか、すなわちどの程度特徴付けているかを示す.

2.2 楽曲特徴量

2.2.1 テンポ

ITG データセットに含まれる譜面データには、楽曲のテンポ値の情報が記録されている. テンポ値の単位には BPM (Beats Per Minute) が用いられている. 楽曲途中でテンポが変化する場合、全ての変化点のタイミングと変化後のテンポ値が記録されている.

Stepmania においては、テンポが速いほど画面上のステップの移動速度が早くなり、プレイヤーに素早い判断を要求することになるため、一般的にテンポが速い曲の方が難しいとされている. また、楽曲途中でテンポの変化は、通常は等速運動をするステップの移動速度の変化を引き起こし、プレイヤーにとって予測しづらいギミックとして作用する.

以下に、テンポに関して取得する特徴量を挙げる. 単位は BPM である. t_μ を除き、小数第一位を四捨五入し、整数値として扱う.

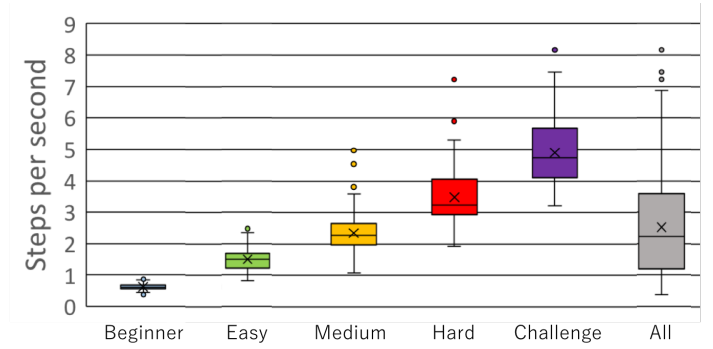
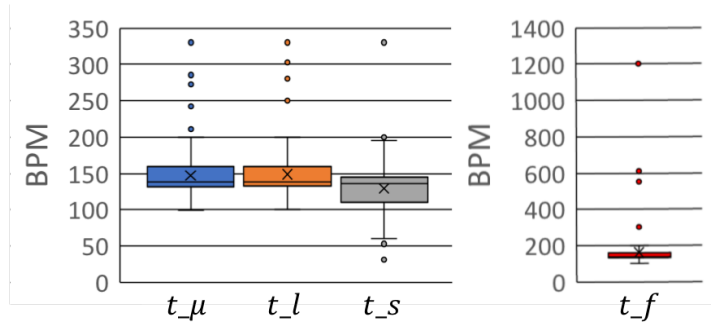
t_l 譜面中で最も長い時間を占めるテンポ値.

t_s 譜面中で最も遅い瞬間のテンポ値.

t_f 譜面中で最も速い瞬間のテンポ値.

t_μ = 総拍数/楽曲の長さ [秒] $\times$ 60.

さらに、テンポの変化に関する以下の特徴量を得る.

図 4 n_μ の分布を示す箱ひげ図.図 5 t_μ, t_l, t_s, t_m の分布を示す箱ひげ図.

$t_r = t_f/t_s$.

t_n テンポ変化が記録されている回数.

t_t 変化前と変化後のテンポの比が 2^m ($m \in \mathbb{Z}$ and $m \neq 0$) であるテンポ変化が記録されている回数. ステップの移動速度変化のみを目的としたテンポ変化の回数とみなすことができる.

2.2.2 譜面停止ギミック

ITG データセット内のいくつかの楽曲には、指定された時間の間ステップの移動を一時停止する「譜面停止ギミック」が設定されている. これは、楽曲途中のテンポ変化と同様、ステップの移動速度に変化を与えることでプレイヤーを騙すギミックとして設定されている. 譜面停止ギミックに関する特徴量は、以下の 1 つである.

p_n 譜面停止ギミックが設定されている回数.

3. 特徴量に基づく譜面の分析と考察

ITG データセット内の全ての譜面について、2 節にあげた特徴量を取得し、分析を行った. 本稿では、特に興味深い結果を得られた特徴量について分析結果を報告する.

図 4 に、 n_μ の集計結果を表す箱ひげ図を示す. Beginner, Easy, Medium, Hard, Challenge はそれぞれの難易度ラベルがつけられた 133 譜面 (Challenge のみ 120 譜面) の集計結果、All はデータセット内の全 652 譜面についての集計結果を示す. また、図 5 に、データセット内の 133 楽曲に対する t_μ, t_l, t_s, t_m の集計結果を表す箱ひげ図

を示す。両方の図において、 Q_1, Q_3 をそれぞれ第1四分位数、第3四分位数として、 $Q_1 - 1.5(Q_3 - Q_1)$ を下回る値、および $Q_3 + 1.5(Q_3 - Q_1)$ を上回る値を外れ値として示している。

3.1 1秒あたりのステップ数平均

図4より、難易度が上がるにしたがって、平均ステップ数が多い範囲に分布しており、ステップ数は難易度が高いほど多くなる傾向がわかる。また、難易度が上がるにしたがって、四分位範囲 $Q_3 - Q_1$ が広がっている。このことから、簡単な譜面はどの楽曲でも似たような頻度でステップが現れる一方、難しい譜面では楽曲によってステップ頻度に差があると言える。さらに、Medium 難易度と Hard 難易度には、それぞれ1つ上の難易度の第3四分位数よりも高い値を示す譜面が存在が外れ値として存在する。これらの譜面は、少なくとも n_{μ} の観点においては、該当の難易度としては不適切に難しい譜面であるとみなすことができる。

3.2 テンポ

図5中の t_{μ}, t_L より、ITG データセットの楽曲のうち半分が 130~160BPM の範囲に該当することがわかった。一方で、例外的に速いテンポの楽曲も存在する。“*Pandemonium*” は楽曲を通して 330BPM であるため、 $t_s = t_f = 330$ という値を示した。この値は、特に t_s において極端な外れ値となっている。“*Robotix*” は $t_L = 150$ だが、楽曲中盤に徐々に加速し $t_f = 1,200$ を記録したのちに減速するという構成になっており、 t_f において他の楽曲と大きくかけ離れた値が観測されている。以上から、ITG データセットには 140BPM 前後のテンポの楽曲が多く収録されている一方で、200BPM を超えるような高速テンポの楽曲もわずかに存在していることが確認された。

4. クラスタリングに基づく譜面の自動生成

2節で設定した特徴量を用いることで、譜面の特性を数値で表現することを実現した。特徴量の傾向が近い譜面同士をクラスタリングすることによって、データセット内の譜面を特性に基づいて分類し、共通の特性を持つ譜面の集合を得ることができる。クラスタごとに譜面生成モデルを学習させることによって、そのクラスタの特性を反映した特徴的な譜面の自動生成を実現する。

4.1 前処理：外れ値を持つ譜面の検出

3節より、ある特徴量において他の譜面とかけ離れた値を持つ譜面がデータセット内に存在することが確認された。このような譜面は、類似した性質を持つ他の譜面がほぼ存在しないノイズデータとして働いてしまう。ノイズデータの存在は、クラスタリングにおいて少数のデータのみで1

つのクラスタを占めてしまうという問題を引き起こす。また、譜面生成モデルがノイズデータを学習すると、生成される譜面にノイズデータの性質が中途半端に反映されてしまう。

これらの問題に対処するため、外れ値を持つ譜面をあらかじめデータセットから除外する。データセット内の全 652 譜面を母集団として、各特徴量に対して増山の棄却検定 ($p = 0.01$) [10] を実施し、外れ値を検出した。ただし、特徴量の次元数が 979 と非常に大きい s_g については、主成分分析 (PCA) を実施し、寄与率が高い上位 7 つの主成分にのみ棄却検定を実施した。652 譜面のうち、少なくとも 1 つの特徴量が外れ値と見なされた 205 譜面を、データセットから除外した。

4.2 クラスタリング

クラスタリングを行うにあたって、2節にあげた特徴量を用いて各譜面を表す特徴ベクトルを生成する。譜面ごとに、以下の 4 種類の特徴ベクトルを生成する。

$$\mathbf{n} = (n_L, n_M, n_{\mu}, n_{\sigma^2}, j_r, j_d),$$

$$\mathbf{l} = (l_4, l_8, l_{12}, l_{16}, l_{24}, l_{32}, l_{oth}),$$

$$\mathbf{s} = (s_g | g \in G),$$

$$\mathbf{t} = (t_{\mu}, t_L, t_s, t_f, t_r, t_n, t_t, p_n),$$

ただし、 $\mathbf{s}$ は 979 次元のベクトルとなる。

それぞれの特徴ベクトルに対して PCA を実施した。 $\mathbf{n}, \mathbf{l}, \mathbf{t}$ については、全ての主成分を利用し、それぞれ同一の次元数をもつベクトル $\mathbf{pc}_{\mathbf{n}}, \mathbf{pc}_{\mathbf{l}}, \mathbf{pc}_{\mathbf{t}}$ に変換した。 $\mathbf{s}$ は次元数が非常に大きいため、累積寄与率が 90% を超えるように寄与率が高い主成分から順に選択し、208 次元のベクトル $\mathbf{pc}_{\mathbf{s}}$ に変換した。

$\mathbf{pc}_{\mathbf{n}}, \mathbf{pc}_{\mathbf{l}}, \mathbf{pc}_{\mathbf{s}}, \mathbf{pc}_{\mathbf{t}}$ のうち任意数を結合することで、譜面ごとに 1 つのベクトルを生成する。本稿では、4 つ全てを結合した 229 次元のベクトルを用いてクラスタリングした結果について報告する。

4.1 節で外れ値が検出されなかった 447 譜面を対象に、 k -means 法によるクラスタリングを行った。クラスタ数 k を、4 から 12 の範囲で変更しながらクラスタリングを試行した。本稿では、 $k = 8$ の結果について報告する。

表3に、クラスタリング結果を示す。クラスタ 1, 2, 4, 6 は、ステップの頻度 n_{μ} とリズムの複雑さ $l_k (k \in \{4, 8, 16\})$ に差が見られた。クラスタ 3 は j_r と j_d が他に比べて高く、ジャンプステップが多い譜面が属した。クラスタ 5 は t_{μ} が高く、テンポが速い楽曲の譜面が属した。以上のよう、クラスタによってそれぞれ異なる特性を示すことが確認された。

外れ値特徴量を持つ譜面を除外したにも関わらず、クラスタ 7 とクラスタ 8 は 1 譜面のみが属するクラスタとなった。これらは、単一の特徴量で外れ値となることはないが、高い値を示す特徴量の組み合わせが他の譜面とは異なっ

表 3 クラスタリング結果

ID	譜面数	特徴量のクラス内平均値							クラスタの特性
		$n-\mu$	$j-r$	$j-d$	l_4	l_8	l_{16}	$t-\mu$	
1	133	3.512	7.8%	0.268	53.9%	34.0%	11.9%	140.265	ステップ数が多い.
2	122	1.229	5.8%	0.070	95.4%	4.5%	0.1%	134.592	ステップ数が少ない.
3	82	1.868	13.8%	0.256	86.6%	12.9%	0.5%	143.252	ジャンプステップが多い.
4	58	2.493	8.1%	0.200	74.0%	21.1%	4.7%	140.788	ステップ数が中程度.
5	26	2.397	6.2%	0.150	84.5%	14.9%	0.6%	185.993	テンポが速い.
6	24	0.758	5.8%	0.070	99.7%	0.3%	0.1%	135.177	4 分のステップの割合が高い.
7	1	4.374	11.2%	0.492	36.4%	30.7%	26.2%	107.004	“Zodiac” (Challenge) のみ.
8	1	2.293	1.3%	0.030	47.4%	40.8%	8.3%	127.994	“Hustle Beach” (Hard) のみ.

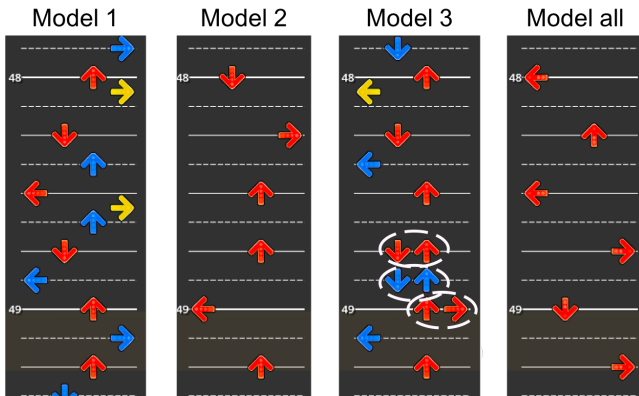


図 6 楽曲“Walking on Fire”を各モデルに入力し、出力された譜面の一部。

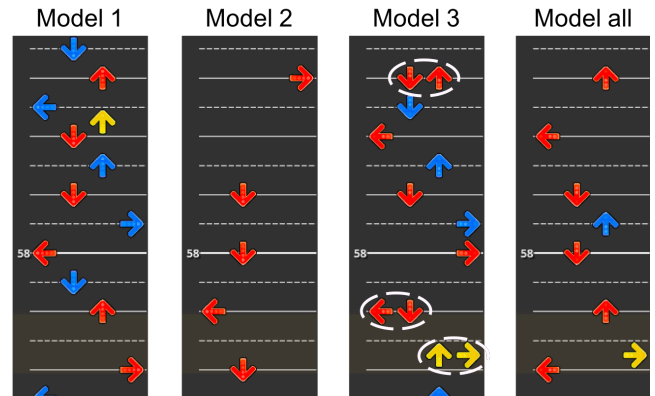


図 7 楽曲“Hillbilly Hardcore”を各モデルに入力し、出力された譜面の一部。

いるために起きた現象と考えられる。今後の課題として、Minimum Covariance Determinant [11] のような多変量データに対する異常検出手法を適用し、この問題を解決することがあげられる。

5. 譜面の学習と自動生成

クラスタごとに、音楽データを入力、譜面データを出力とする LSTM モデルを学習させた。モデルの構造は、Donahue らの先行研究 DDC [7] と同一のものを用いた。

譜面数が比較的多い、クラスタ 1,2,3 を学習の対象とした。クラスタごとにランダムに、64 譜面を学習データ、9 譜面を検証データ、9 譜面をテストデータに振り分けた。クラスタ n を学習したモデルを、Model n と名付けた。また、比較対象として、上記 3 クラスタ全ての学習データ 192 譜面と検証データ 27 譜面を用いて学習したモデルを用意し、Model all とした。

図 6, 図 7, 図 8, 図 9 に、それぞれ楽曲“Walking on Fire” (クラスタ 1 テストデータ)、“Hillbilly Hardcore” (クラスタ 2 テストデータ)、“Bubble Dancer” (クラスタ 2 テストデータ)、“Twilight” (クラスタ 3 テストデータ) を入力音楽データとした際に、各モデルが生成した譜面の一部を示す。図中で、破線丸で囲ったステップはジャンプ

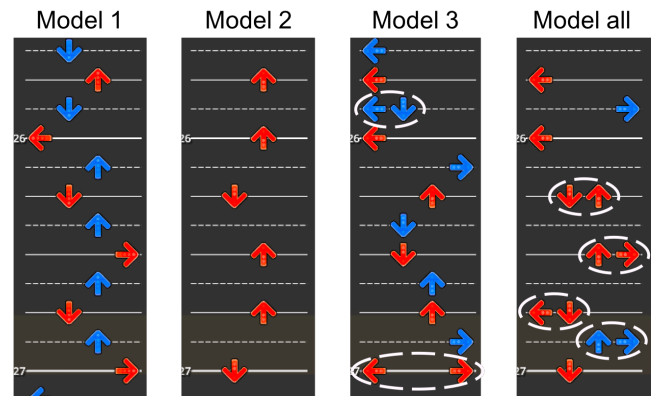


図 8 楽曲“Bubble Dancer”を各モデルに入力し、出力された譜面の一部。

ステップである。配置されたステップ数について、Model 2 が最も少なく、Model all, Model 3, Model 1 の順に多くなっていくことが確認される。また、Model 3 が生成した譜面にはジャンプステップが多く存在するが、Model 1 と Model 2 が生成した譜面にはジャンプステップが見られなかった。Model all が生成した譜面のうち、図 6, 図 7, 図 9 にはジャンプステップが見られなかったが、図 8 中にはいくつかのジャンプステップが確認された。以上から、Model 1,2,3 が生成した譜面は、4.2 節に記述したクラス

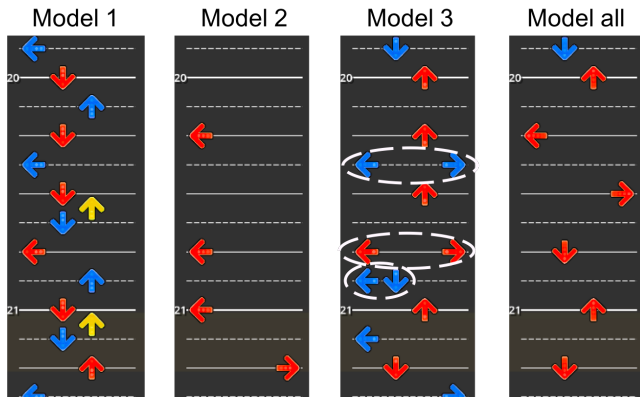


図 9 楽曲“Twilight”を各モデルに入力し、出力された譜面の一部.

表 4 各モデルの F -score

	Test 1	Test 2	Test 3
Model 1	0.678	0.364	0.474
Model 2	0.546	0.455	0.508
Model 3	0.676	0.401	0.506
Model all	0.662	0.444	0.517

タ 1,2,3 の特性をそれぞれ強く反映していると考えられる. その一方で, 3 つのクラスタ全てを学習した Model all の生成譜面は, わずかに特性が見られるものの, 各クラスタの特性が平均化され強くは反映されていないと考えられる.

以降の 5.1 節と 5.2 節では, 各モデルの生成譜面が対応するクラスタの特性をどの程度反映できているかについて, 各クラスタのテストデータを用いて客観評価実験を行った結果について述べる.

5.1 F -score による生成譜面の評価

それぞれのクラスタでテストデータに分類された楽曲を, 各モデルに入力することによって, 譜面を生成した. 生成された譜面でステップが配置された時刻と, 正解データにおいてステップが存在する時刻を比較し, F -score を計算した. 表 4 に結果を示す. ここで, Test n はクラスタ n から選ばれたテストデータを表す.

Test 1 および Test 2 については, それぞれ対応する Model 1 と Model 2 が, 他のモデルよりも高い F -score を示した. 一方で, Test 3 に対して最も高い数値を示したのは, Model 3 ではなく Model all であった. このことから, Model 1 および Model 2 は, ステップを配置する時刻についての特徴をうまく捉えられていると言える. しかし, Model 3 については, ステップ配置時刻の特徴を捉えていない.

5.2 データセットと自動生成譜面の統計比較

各モデルに Test 1, Test 2, Test 3 の楽曲データを入力し, 計 24 譜面を生成した. 表 5 に, 生成された各譜面に対

表 5 各モデルが生成した譜面の $n_{-}\mu, j_{-}r, j_{-}d$ の平均値

	$n_{-}\mu$	$j_{-}r$	$j_{-}d$
Model 1	4.481	1.7%	0.067
Model 2	1.736	1.2%	0.020
Model 3	3.816	11.2%	0.423
Model all	2.998	6.7%	0.181

して $n_{-}\mu, j_{-}r, j_{-}d$ を計算し, 24 譜面の平均を求めた結果を示す. 表 5 と表 3 を比較すると, $n_{-}\mu$ について, Model 1 および Model 2 はそれぞれある程度クラスタ 1 およびクラスタ 2 に近い値を示したが, Model 3 はクラスタ 3 よりも明らかに高い値を示した. $j_{-}r$ と $j_{-}d$ について, Model 3 は他のモデルよりも高い値を示した. 以上より, Model 1 および Model 2 については, 対応するクラスタのステップ頻度についての特性を, ある程度学習できたと言える. Model 3 は, ジャンプステップが多いというクラスタ 3 の特性を反映できている一方で, ステップ頻度についてはうまく学習できなかったのではないかと考えられる.

6. おわりに

本稿では, ダンスゲーム譜面の難易度に影響する特徴量を設定し, 特徴量に基づいてクラスタリングを行った. 設定した特徴量は, 難易度によって確かに差があることを確認した. また, いくつかの特徴量で例外的に高い値を示す譜面が存在することを確認した. このような極端な譜面をデータセットから除外した上で, クラスタリングを実施した. クラスタリングの結果, ステップの頻度, リズムの複雑さ, ジャンプステップの頻度などの特性にもとづいて譜面を分類することができた. クラスタごとに譜面生成モデルを学習させることで, 各クラスタの特性を反映した特徴的な譜面が生成できた.

今後は, クラスタリングに用いる特徴量を精査するなどして, より譜面の特性を詳細に分析することを考えている. また, 入力楽曲やプレイヤーの嗜好に応じて, 適した特性の譜面を生成する方法についても検討する.

謝辞 実験の実施にあたり, 立命館大学情報理工学部の西原 陽子准教授の協力を得た. 記して謝意を示す.

参考文献

- [1] Hunnicke, R.: The case for dynamic difficulty adjustment in games, *Advances in Computer Entertainment Technology*, pp. 429–433 (2005).
- [2] Wehbe, R. R., Mekler, E. D., Schaekermann, M., Lank, E. and Nacke, L. E.: Testing Incremental Difficulty Design in Platformer Games, *Proc. of the 2017 CHI*, pp. 5109–5113 (2017).
- [3] Spiel, K., Bertel, S. and Kayali, F.: “Not another Z piece!” Adaptive Difficulty in TETRIS, *Proc. of the 2017 CHI*, pp. 5126–5131 (2017).

- [4] Behrenshausen, B. G.: Toward a (Kin)Aesthetic of Video Gaming: The Case of Dance Dance Revolution, *Games and Culture*, Vol. 2, No. 4, pp. 335–354 (2007).
- [5] Shaker, N., Togelius, J. and Nelson, M. J.: *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Springer (2016).
- [6] Pedersen, C., Togelius, J. and Yannakakis, G. N.: Modeling player experience for content creation, *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 1, pp. 54–67 (2010).
- [7] Donahue, C., Lipton, Z. C. and McAuley, J.: Dance Dance Convolution, *Proc. of ICML 2017*, pp. 1039–1048 (2017).
- [8] 辻野雄大, 山西良典, 西原陽子, 福本淳一: 時系列深層学習に基づく難易度間関係モデルを用いたダンスゲーム譜面難易度の自動調整, 情報処理学会論文誌, Vol. 59, No. 11, pp. 1953–1964 (2018).
- [9] Tsujino, Y. and Yamanishi, R.: Dance Dance Gradation: A Generation of Fine-Tuned Dance Charts, *Entertainment Computing – ICEC 2018*, pp. 175–187 (2018).
- [10] Kobayashi, K. and Pillai, K.: *A Handbook of Applied Statistics Pharmacology*, CRC Press (2012).
- [11] Rousseeuw, P. J. and Driessen, K. V.: A fast algorithm for the minimum covariance determinant estimator, *Technometrics*, Vol. 41, No. 3, pp. 212–223 (1999).