令和元年度　特別研究報告書

ベース初心者の熟達度に応じたベースフレーズの演奏難易度評価

龍谷大学　理工学部　情報メディア学科

T160361　上田 優太

指導教員　三好 力　教授

・内容概要

ベース初心者が独学で練習をする際、自身の熟達度に応じた課題曲を自身で選定することは困難であるといえる。そこで本研究では独学での練習支援を目的に、ベース初心者の熟達度に応じたベースフレーズの難易度予測式を構築した。フレーズの難易度評価において、楽譜的特徴量と演奏者の熟達度の両方を考慮することが重要であるが、本稿では後者に重点を置いた。

まず、演奏者の熟達度を考慮するためにベースフレーズに対する熟練度の評価を行った。その際、フレーズの正解音源と奏者の演奏音源との一致率を算出し、その値を用いた。音源の比較を容易にするため、二つの音源を MIDI(Musical Instrument Digital Interface)データとして取得した。正解音源はあらかじめ打ち込みで作成し、演奏音源は MIDI コンバータを用いて MIDI 変換すると同時に演奏動画を録画し、その動画像情報を用いて変換の誤差補正を行った。

そして、ベースの教則本より選出した演奏の基本的な 10 フレーズに対し、7 人のベース経験者に演奏を行ってもらい、演奏者の熟達度と主観的に判断したフレーズの難易度の 2 つのデータをそれぞれ収集した。これらのデータを用いて、難易度予測式を重回帰分析で構築した。目的変数には演奏者が主観的に判断したフレーズの演奏難易度、説明変数には従来の演奏難易度評価に用いられた楽譜的特徴量とベースフレーズに対する熟練度を用いた。結果として、構築した予測式の相関係数 R は 0.54、寄与率R^2は 0.29 であった。予測式の評価は Leave-one-out-cross-validiation で行った。また、従来手法の素性のみで重回帰式を構築し比較を行った。その結果、重回帰式の素性にフレーズに対する熟練度を加えることで相関係数が向上したため、予測式の素性にフレーズに対する熟練度を加えることが有用であると示唆された。

目次

第1章 はじめに

近年、インターネットの普及により楽曲、楽譜の取得が容易になり、楽器などの演奏機材を低価格で揃えることができるようになってきている。また、楽器の練習をする際、既存の楽曲を忠実に再現することが多い[1]。このように、既存の楽曲を用いて独学で練習する環境の敷居が低くなったといえる。その際、演奏者の熟達度に応じた演奏難易度の楽曲を選択することが重要となる。しかし、特に初心者による独学練習においては、演奏者の熟達度に応じた楽曲の選択を自身で行うことは困難であり、膨大な労力がかかる。

この問題への対策として、演奏難易度を自動的に測定する評価手法[2,3]が提案されている。これらは、2.1 で述べるように、楽曲の楽譜的特徴に基づく評価手法である。

しかし、これらの評価手法は演奏者の熟達度に応じたものではないため、独学練習における課題曲の演奏難易度を測定することには問題がある。

そこで、本研究では、ベース初心者の独学練習の効率化、および、目標レベルの到達に至るまでの最適な課題曲を推薦することを目的に、楽曲におけるベースフレーズの難易度予測式の構築を行った。

第2章 従来研究

2.1 フレーズおよび楽曲の難易度評価

齋藤ら[2]は楽譜に含まれる統計量を用いてギターフレーズの難易度評価を行っている。実際に用いた統計量は BPM、ノーツ密度（休符でなく、発音時間だけをフレーズ全体の時間で割ったもの）などのフレーズに 1 つしかない量と、それ以外のノーツ長、フレット番号、弦番号、指板上二音間距離、二音間時間差などである。ここで BPM などのフレーズに 1 つしかない量以外は、各音符のリストに対して平均、中央値、分散、標準偏差を取得している。例えばフレットを 1、2、4、5 の順に押さえるフレーズのフレット番号の平均は 3 と計算するということを意味する。

実験では、ギター経験者 10 名に対し、9 個のギターフレーズを演奏してもらい、演奏後にフレーズの難易度に関するアンケートを行っている。アンケートは難易度という人間の感情的な数値と楽譜的統計量を結びつけるため、SD 法を用いて 10 項目のアンケートを行っている。アンケートの質問項目は以下の 10 項目である。

(1)このフレーズは難しかったですか? (2)リズムは弾きづらかったですか?
(3)BPM は速かったですか? (4)右手は難しかったですか?
(5)フィンガリングは大変でしたか? (6)音はきれいに出ていましたか?
(7)左手がこんがらがる (8)このフレーズはよく弾く感じの運指だ
(9)基本的なスケールを使用している (10)ぱっと見てフレーズの仕組みが理解できる

(1)の質問は全体的な難易度を表すもので、(2)~(4)は音価、(5)~(7)は運指技術に関して、(8)~(10)はフレーズの頻出度の影響を想定した質問である。

実際に用意されたフレーズはロック・ポップスのバンド曲から被験者が「易しい」「適性」「難しい」と感じるようにそれぞれ 3 フレーズずつ、合計 9 フレーズである。10 名の被験者がそれぞれ 4~8 小節程度のフレーズを実際にギター演奏した上で上記の質問項目に回答している。

そして、楽譜的統計量を説明変数 X、SD 法を用いたアンケートによる人間が付与したフレーズの難易度を目的変数 y として、回帰式(1)をもとに重回帰分析を行っている(ただし、a:偏回帰係数、b:誤差)。

$$y=aX+b \tag{1}$$

結果として、ギターの難易度に関するアンケートを実施により、難易度が音価と運指の 2 軸に大別されることが確認されている。また、楽譜的統計量の観点からは、BPM の他に、音価の軸に関してはノーツ長の分散、運指の軸に関しては弦やフレットの番号の分散が難易度に大きく影響していると結論付けている。

藤井ら[3]は SMF(standard MIDI file)のピアノの楽曲データの難易度判定を行

った。まず、既存の SMF を読み込み、各イベントを解析し、個々の音符長、音符の個数などの情報を得る。次に、そこで得た情報を元に音長・音高・テンポ・指くぐりの 4 点に対して難易度判定を行い、楽曲全体の難易度評価を行っている。

音長に対する難易度を判定では、音符の種類毎に難易度値 Dc を定めている(表 1)。

表 1.音符の種類毎の難易度値 Dc

音符の種類	難易度値Dc
2分音符以上の長さ	2
4分音符	4
8分音符	6
16分音符	8
上記に当てはまらないもの	9

判定方法として、対象のトラック内の全ての音長を調べ、最も多く使われる音長を L とおき、その長さの音符の持つ難易度値 Dc を難易度値 Dp とした。複数の音長が同数で最もよく使われていた場合は、それらの音符の難易度値の平均を Dp としている。ここで、Dp をそのまま音長における難易度値 D1 としたとき、長さ L 以外の音符の持つ難易度値が反映されないため、音符の長さが L でない音符の長さを 1 つずつ L と比較していき、L よりも短い音符の分だけ Dp を大きく、また長い音符の分だけ Dp を小さくして対象トラック内の全ての音符の難易度値を反映させ、最終的な値を D1 としている。

長さが L でない音符の総数を N とおくと、1 回あたりの比較で増加または減少され得る値の上限値A_{max}、S_{max}は式(2)、(3)で求めている。

$$A_{max} = (10 - D_p)/N \tag{2}$$
$$S_{max} = (D_p - 1)/N \tag{3}$$

次に、L よりも音長の短い音符の総数をN_s、n 番目の音符が持つ難易度値D_sをD_{sn}とし、同様に L よりも音長の長い音符の総数をN_L、n 番目の音符の持つ難易度値D_LをD_{Ln}とすると、合計加算値 A と合計減算値 S は各々式(4)、(5)で求めている。また、A と S を用いて、最終的な難易度値D_1を式(6)で求めている。

$$A = \sum_{n=1}^{N_s}((D_{sn} - D_P)/D_P)A_{max} \tag{4}$$
$$A = \sum_{n=1}^{N_L}((D_P - D_{Ln})/D_P)S_{max} \tag{5}$$
$$D_1 = D_P + A - S \tag{6}$$

音高に関する難易度判定では、音間の高低差を表す隣り合う音符間の差分の大きさ毎に難易度値D_Hを定めている(表 2)。判定方法としては、隣り合う音符の差分が 0 でない箇所全てに対して音高の差分を求め、その大きさに該当するD_Hを足し合わせた平均値を音間の高低差における難易度値D_2としている(図 1)。

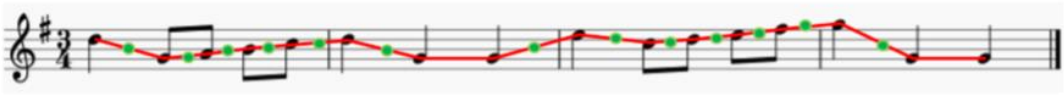

図 1.音高の差分の抽出

表 2.難易度値D_H難易度値D_r

差分	難易度値DH	テンポ(BPM)	難易度値Dr
~3	1	~79	2
4~8	3	80~119	3
9~12	6	120~159	4
13~	8	160~199	6
		200~	10

表 3.指くぐりの頻度毎の修正係数O_p

1小節あたりの指くぐりの回数	修正係数Op
~0.20	1.00
0.21~0.50	1.05
0.51~0.80	1.10
0.81~1.00	1.15
1.01~	1.20

差分が 0 でない箇所の総数をN_a、n 箇所目の差分の大きさに該当する難易度値D_HをD_{Hn}とし、音間の高低差における難易度値D_2を式(7)で表す。

$$D_2 = \sum_{n=1}^{N_a}(D_{Hn})/N_a \tag{7}$$

テンポに関する難易度判定では、楽曲のテンポ毎に難易度値D_r定め(表 2)、該当するテンポの難易度値を楽曲のテンポにおける難易度値D_rとしている。

楽曲全体の難易度判定には、音符が存在する全てのトラックと楽曲のテンポの判定結果を用いている。対象トラック i の難易度値D_iを式(8)で求めている。

$$D_i = (D_{i,1} + D_{i,2})/2 \tag{8}$$

ここで、$i = \{M, A\}$であり、各々メロディ・パート、伴奏パートを示している。

その後、対象トラック内で発生した指くぐりの頻度による難易度の修正を行う(指くぐりとは、一定数以上の音符によって音程が上昇あるいは下降するフレーズを演奏する際に用いられる技法である)。対象トラック内で 1 小節あたりに発生する指くぐりの回数を求め、その値に該当する修正係数O_p(表 3)とD_iとの乗算を行い、D_iの値を増加させる。

楽曲全体の難易度D_Oは、メロディ・パートの難易度値D_M、伴奏パートの難易度値D_M、楽曲のテンポから判定された難易度値D_Tおよび指くぐりによる修正係数O_pの 4 項目から決定される。これらの項目について、どの難易度をどの程度重視するかは任意で決められるものとする。各重みをW_M、W_A、W_Tとおき、D_Oを式(9)で求めている。

$$D_O = O_p(W_M D_M + W_A D_A) + W_T D_T \tag{9}$$

実験結果として、音符の長さと音間の高低差のみを判定要素としたフレーズの難易度判定と、簡単な楽曲全体の難易度判定の 2 点を行い、良好な結果を得ている。

2.2 カメラを用いた運指情報の取得

Chutisant ら[4]はステレオカメラを用いて、ギターで弾いた和音を認識する手法を提案している。システムの構成としては、演奏者の各指に異なるカラーマーカを貼り付け、入力画像から各指の押し位置のパターンをリアルタイムで認識している。ここでカラーマーカは、カメラ画像から得られた RGB 値を輝度変化に強い HSV 値に変換し、単純ベイズ分類器を用いて行うことで、著しい照明変化および動的背景においても認識することを可能にしている。さらに、ARtag を基準に用いて、弦を押さえる範囲をギターが動いていても、認識させている。また、ステレオカメラで三角測量法を利用することによって、ギターの弦が押されたときに指の 3D 位を認識している。さらに、複雑なコード分類を効率的に処理するために PCA(主成分分析)の利点を利用している。結果として、この提案システムはリアルタイムにおいて正確にギターコードを認識できている。図 2 は PC 画面上に認識されたギターコードが表示される様子である。

澤ら[5]はウッドベース効率的な演奏学習を目的に、カメラベースのシンプルな画像処理と、ウッドベースの演奏特性をもとに定義したルールを組み合わせることで、実時間で高精度にウッドベースの演奏情報を取得するシステムを設計した。ここでの提案システムでは、指に貼り付けたカラーマーカをカメラで読み取り、ウッドベースの演奏特性から定義したルールを用いて検出結果を補正することで運指検出を行い、平均 87%の正答率で運指情報を取得している。ここにおいても同様に RGB 値から HSV 値に変換してカラーマ

ーカの色抽出を行っている。

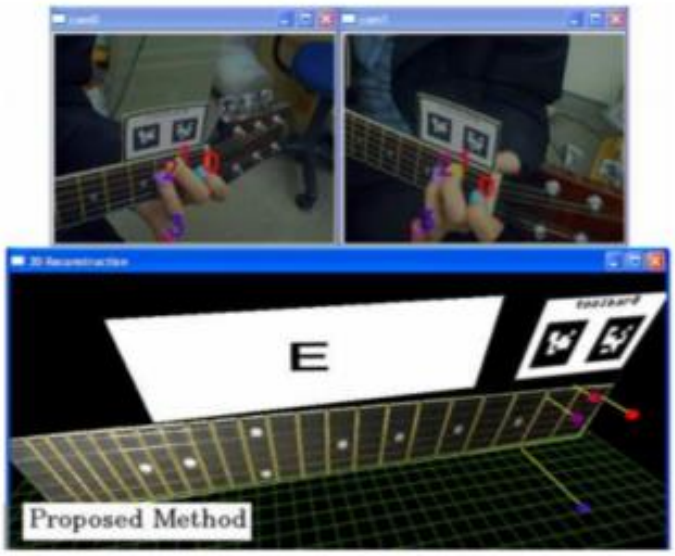

図 2. 従来研究によるギターコード認識結果の出力画面

2.3 演奏における熟達度の評価

岩見ら[6]はドラム演奏の基礎的な練習を支援するシステムを提案しており、MIDI ドラムから入力される演奏情報を解析し、奏者の演奏を様々な方法でリアルタイムに視覚化している。提案システムでは、ユーザが演奏したいドラム楽譜を入力あるいは選択し、MIDI ドラムを用いてその楽譜を指定テンポで演奏する。演奏は MIDI データとして入力される演奏に対して、欠落音の有無や打叩時刻のずれ、打叩強度を自動的に検出し、演奏を様々な方法でリアルタイムに視覚的にしている。演奏中の奏者は自身の演奏を視覚的に確認し、以降の演奏に反映させることができる。演奏が終了すると、その演奏全体が予め定められた評価項目に基づいて自動的に評価される。そこで、ドラムループ演奏における熟達度評価のために、欠落音に関する評価、打叩時刻ずれに関する評価、打叩強度に関する評価の順に、各評価項目に基づいて段階的に評価され、後に続く評価項目ほど高い演奏技術が要求される演奏評価スキームを定め、いくつかの評価項目によって簡易的に演奏を自動的に評価している。その際に、楽譜通りの演奏を評価することが前提となっており、演奏と楽譜間のマッチング処理を行っている(図 3)。ドラム演奏においては、音高情報をほとんど使用することができないため、打叩時刻と打叩強度情報のみを用い処理を行っている。提案されたマッチングの手法では入力されたすべての演奏は時間誤差が最も小さいと解釈される音に対応付けるという前提のもと、楽譜上のいずれかの音符に関連付けしている。これにより、楽譜上の単一の音符に対して、単一の演奏音が関連づけられた場合、それらをマッチさせている。また、楽譜上の単一の音

符に対して、複数の演奏音が関連付けられた場合、その中で打叩強度が最大であるものとマッチさせ、それ以外は挿入音であると見なす、関連付けられた演奏がない楽譜上の音符は欠落音として検出している。

上記のマッチング処理を行い、奏者の演奏からいくつかのパラメータを抽出し、各パラメータを抽出し、各パラメータに対して閾値処理をし、演奏における熟達度の評価を行っている。

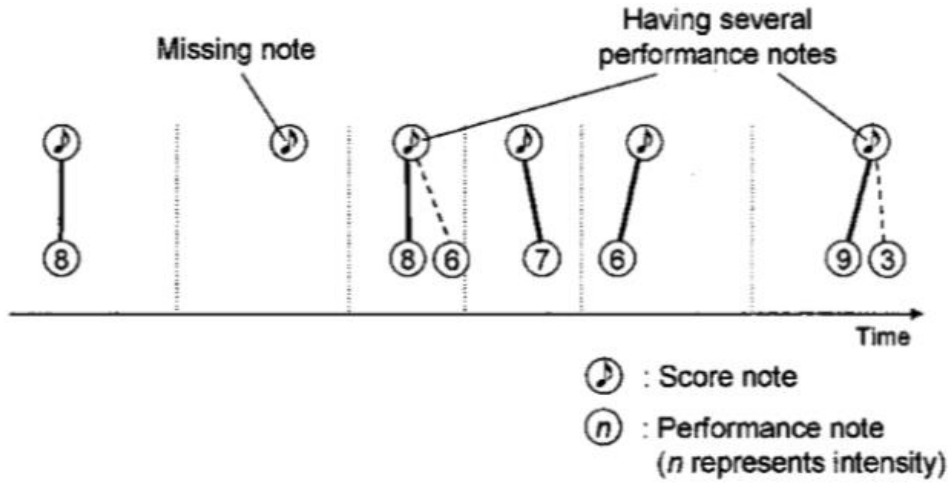

図 3.演奏–楽譜マッチングアルゴリズム

第3章 提案手法

従来の難易度評価[2,3]においては楽譜的な特徴量のみを用いており、実際に演奏する対象者の熟達度を考慮していない。このとき、楽譜的な特徴量を用いて定めた楽曲の難易度は熟練者や初心者であろうと同等の難易度値になってしまう。すなわち、演奏難易度というのは演奏者の熟達度によって異なるという点が考慮できていないといえる。

そこで、本研究では従来手法で用いられた楽譜的特徴量に加えて、演奏者の熟達度を考慮した演奏難易度予測式(以下、提案式 1)の構築を行う。提案式 1 の構築には重回帰分析を用いた。演奏者の熟達度としては、実際に奏者にベースフレーズの演奏を行ってもらい、奏者ごとに演奏の評価値を定め、その値を用いた。

3.1 フレーズの演奏音源と主観評価の取得

演奏者の熟達度を算出するために演奏の録音を行うが、録音したアナログ音源情報から、演奏評価のための特徴量抽出や解析が困難であった。そこで、演奏評価を容易に進めるために演奏情報を MIDI データとして取得した。その際、リアルタイムでの MIDI 変換に少々、誤入力や遅延が見られたため、誤入力の補正には録画した演奏の動画像データから運指情報を取得し用い、遅延の補正には録音した演奏のアナログ音源情報を用いた。動画像データから運指情報を取得する際、左手の人差し指から小指にかけて爪先にカラーマーカを装着し色抽出を行った。色抽出において、左手が動作している場合、輝度変化などが問題で抽出ができないことがあったため、動画像データを RGB 値から輝度変化に強い HSV 値に変換することで対応を図る。また、演奏者には演奏したフレーズに対し、主観的な難易度評価を 5 段階で行ってもらい、その評価値を提案式 1 の構築に用いた。

3.2 演奏者の熟達度評価

演奏者の熟達度の評価は、ドラムとベースが音楽において比較的に同様の役割を果たす[6]ことから、岩見ら[7]が提案したドラム演奏における熟達度評価を参考に行った。あらかじめフレーズの正解音源を MIDI データとして作成し、補正した演奏音源との一致率を算出し、その平均値を熟達度とした。

本研究ではベース演奏における熟達度評価のためのスキームを提案した。そのスキームを図 3 に示す。ここで、このスキームにおける欠落音の数、音高差の大きさ、発現時刻のずれの大きさの割合区分は表 4 に示す。提案するスキームでは、演奏を図 4 に示すように、欠落音に関する評価、音高に関する評価、撥弦時刻のずれに関する評価の順に、各評価項目に基づいて段階的に評価されていく。このスキームにおいては、後に続く評価項目ほど高い演奏技術が要求されるように設計されている。

このスキームをもとに演奏のマッチング処理を行う。まず、入力されたすべての演奏は時間誤差が最も小さいと解釈される音に対応付けるという手法にしたがっ

て楽譜上のいずれかの音符に関連付けられる。これにより、楽譜上の単一の音符に対して、単一の演奏音が関連付けられた場合、それらをマッチさせる。また、楽譜上の単一の音符に対して、複数の演奏音が関連付けられた場合、演奏時刻と音高の情報が最も楽譜上の音符に近いものとマッチングさせ、それ以外は余剰な演奏音と見なす。関連付けられた演奏がない楽譜上の音符は欠落音として検出される。そして、図 4 の演奏評価スキームをもとに、上記のマッチング処理を行い、演奏者の演奏からいくつかのパラメータを抽出し、各パラメータに対して閾値処理を行うことで演奏評価を行う。この演奏評価は演奏者の各フレーズに対して行い、それぞれのフレーズに対する熟練度の平均値をその演奏者の熟達度とする。

表 4.演奏に対する熟練度評価項目の区分の詳細一覧

	非常に多い/大きい	多い/大きい	少ない/小さい
欠落音の数	40%以上	39~10%	9%以下
音高差の大きさ			
撥弦時刻のずれの大きさ			

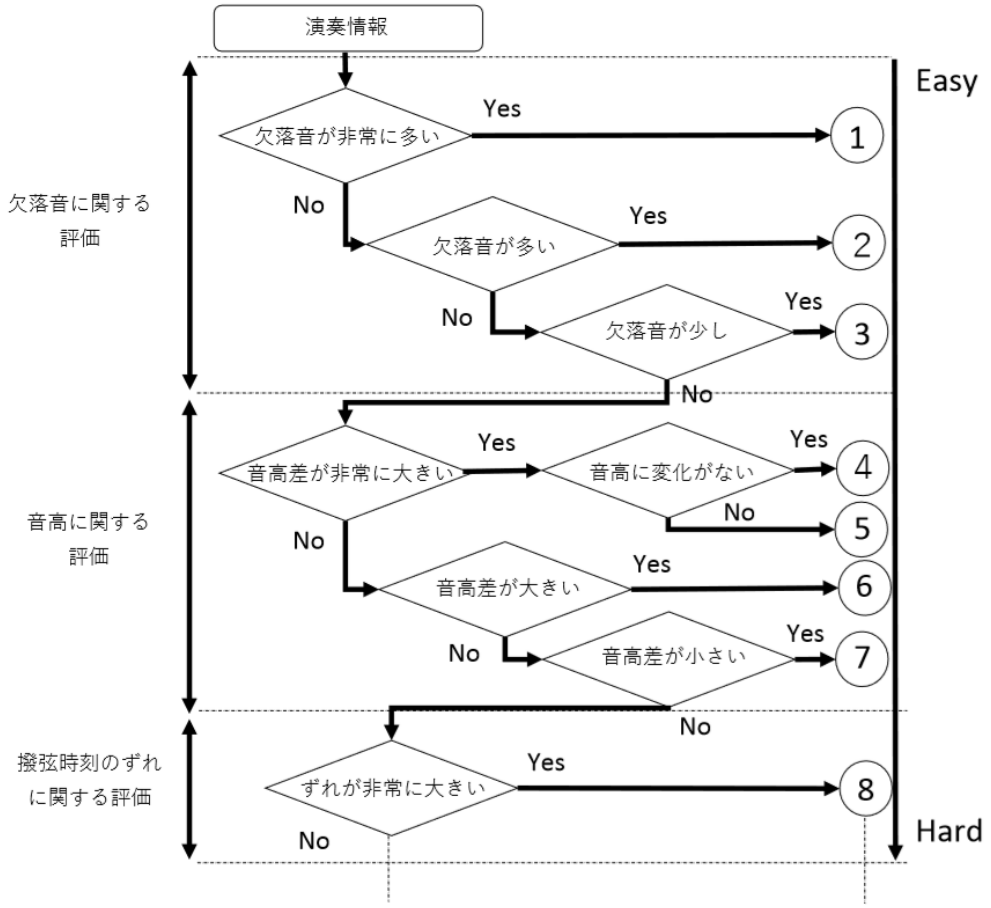

図 4.ベース演奏における熟達度評価スキーム

3.3 予測式の構築

目的変数には 4.1 で述べる 70 件の演奏者が付与した主観的なフレーズに対する 5 段階の演奏難易度を用いた。また、説明変数には従来の予測手法[2,3]で用いられた素性をすべて用いた。さらに、演奏者の熟達度を考慮するために 4.3 で算出した値である演奏者のフレーズに対する熟練度も加えて提案式 1 を提案し、式(10)に示す。

$$Y = \sum_{i=1}^{7} a_i X_i \tag{10}$$

・ X_1:BPM、X_2:ノーツ密度、X_3:ノーツ長(分散)、 X_4:フレット番号(分散)、X_5:弦番号(分散)、X_6:特殊奏法の割合、X_7:フレーズに対する熟練度(平均値)、a_1～ a_7：偏回帰係数

第4章　実験

4.1　フレーズの演奏の録音および主観評価の取得

本研究で提案する演奏音源取得システムの構成を図 5 に示す。このシステムでは、ピックアップから演奏情報(音高、弦番号)、PC カメラからは AR マーカを基準に指板領域(ネック表側の弦を指で押さえつける面)と奏者の左手の人差し指から小指にかけて各指の爪に装着したカラーマーカの動画像データの 2 つの情報を得る。ここで、カラーマーカは各指、色が異なるもの用意し、演奏に支障をきたさない薄さのシールを用いた。そして、PC 上で取得した動画像データから運指を特定し、演奏情報に補正を加える。こうして補正した演奏情報をフレーズの演奏音源とする。なお、演奏情報を MIDI データとして取得するために、ベースの各弦の音源を出力できる Roland 社の外付けピックアップ GK-3B をベース本体に装着し、各弦の出力を MIDI データに変換するために、MIDI コンバータとして Roland 社の GR-20 を用いた。そのため、弦ごとに独立して処理を行うことで弦番号と音高を取得できる。また、弦番号とは、ベースがもつ 4 本の弦の中で最も太い(音高が低い)弦から順に 4 から 1 まで番号付けした弦の番号のことである。また、PC 上で演奏した MIDI データの処理を行うために steinberg 社のオーディオ・インターフェース UR22mkⅡを用いて、PC に演奏情報を出力する。さらに PC で演奏情報の録音・補正編集などを行うために、DAW ソフトの reaper を用いた。DAW ソフトの reaper を用いて補正編集を行う際、MIDI データをもとに、動画像データから算出した運指情報と照らし合わせて余分な音の削除、出力されなかった音の追加などを行う。

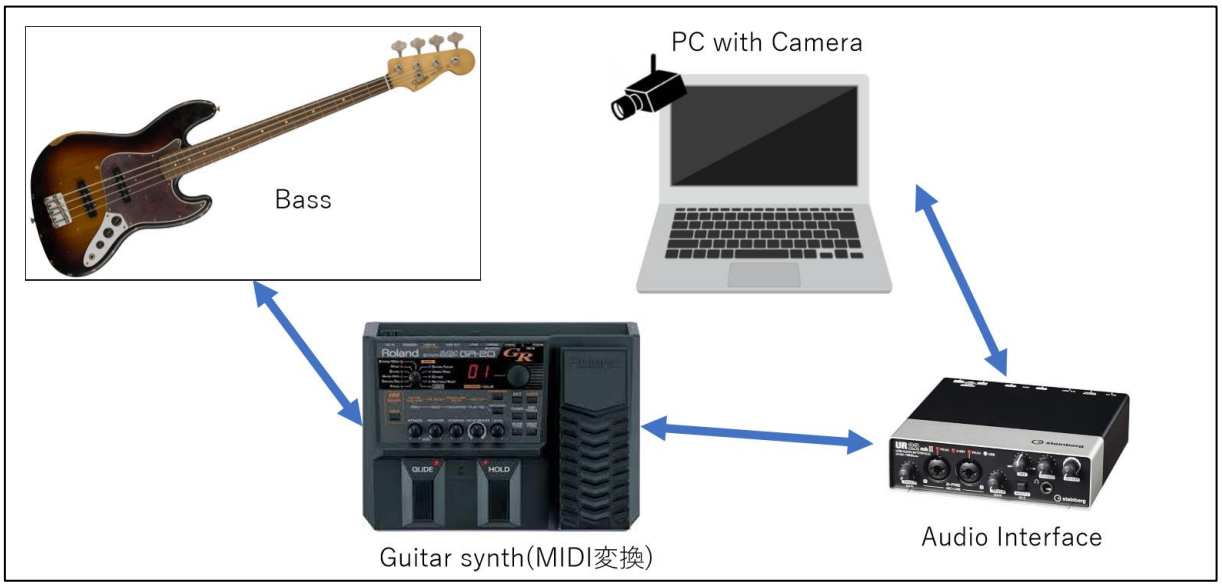

図 5.　演奏音源取得システムの構成

次に、演奏音源取得の流れを図 6 に示す。また、本研究で使用したエレクトリック・ベースと外付けピックアップの位置を図 7 に示す。外付けピックアップは図 6(右)のようにベース本体に固定し、AR マーカを図 7(左)の位置に貼り付けた。

演奏情報取得の流れは前処理として外付けピックアップの出力から各弦の MIDI データ、同時に PC カメラの前に AR マーカ及び指板領域が映る位置で演奏の録画を行い動画像データの 2 つのデータを取得し、動画像データによる MIDI 情報の補正は後処理で行った。

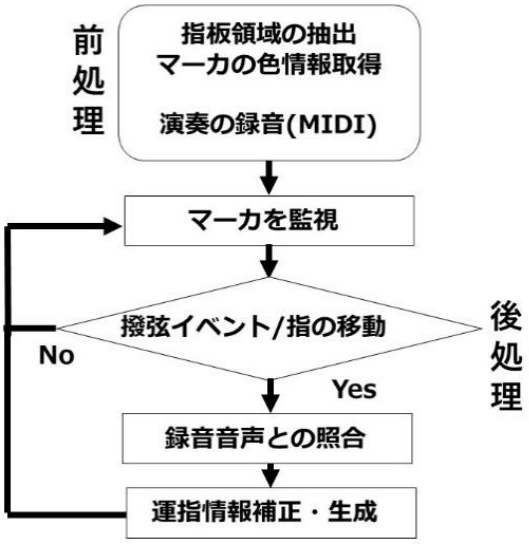

図 6. 演奏音源取得処理のフローチャート

そして、演奏情報補正の後処理として、前処理で取得した動画像データをもとに、各指のカラーマーカ(図 8)と指板領域の相対的位置から抑弦位置を算出し運指情報の取得を行った。ここで、指板領域の認識・抽出は AR マーカを基準に ARtoolkit ライブラリを用いて、動画像データ上における xy 座標をあらかじめ設定している。AR マーカの認識判定は図 9 に示すように、認識されたとき、抽出する指板領域のフレットが青く表示される。また、各指につけた色の異なる 4 色のカラーマーカの色抽出をそれぞれ行い、そのときの各色(各指)が動画像データ上における位置情報を xy 座標として取得する。ここで、動画像データは RGB 値を輝度変化に強い HSV 値に変換し、用意する 4 色のカラーマーカの色の HSV 値をあらかじめ取得し設定しておくことでカラーマーカの色抽出を行う。こうして、取得した各指のカラーマーカの位置情報とあらかじめ設定しておいた指板領域の位置情報を照らし合わせ、相対的に抑弦位置を算出し、運指情報の取得をした。そして、この運指情報と MIDI データの演奏情報の 2 つを照らし合わせ、異なる点があれば補正を行った。なお、AR マーカおよび指先につけたカラーマーカの画像処理には、Windows8.1 上で Vsual C++.NET2019 と Intel 社の OpenCV ライブラリ、ARToolKit ライブリを用いて行った。

また、この補正とは別にライン出力より取得した、アナログ音源の情報を用いて

の補正も行う。MIDI データが出力される際、実際の演奏タイミングより少し遅れが見られた。この出力の遅延を考慮するために、変換前のアナログ音源の発音タイミングを基準に遅延のずれを補正した。このようにして MIDI データの誤出力、遅延などの補正を行い、より正確な MIDI データとして編集し取得を行った。

このように、70 件(教則本より抜粋した 10 フレーズ×ベース経験者 7 人)のデータ取得を行い、フレーズ演奏はライン出力と MIDI 出力で同時に録音した。また、演奏者は楽譜を見て、正解音源を聴きながら演奏を行い、演奏を終えるたびにフレーズの演奏難易度を主観的に判断した。演奏難易度は 1:簡単、2:少し簡単、3:普通、4:少し難しい、5:難しいの 5 段階である。

図 7.エレクトリック・ベース(左)、外付けピックアップ(右)

図 8.各指にカラーマーカをつけた様子

図 9.AR マーカの認識判定および指板領域の抽出結果出力画面

4.2 録音データの補正

演奏は編集の容易である MIDI データで扱うが、録音の際に多少の遅延、誤入力などがみられたため、動画像データとライン出力を用いて補正を行った。動画像データは PC カメラを用いて、ベース本体に貼り付けた AR マーカと演奏者の左手の人差し指から小指にかけて指先につけたカラーマーカをよく映るように録音と同時に録画した。録画した動画像データは後処理で AR マーカから指板領域、指先のカラーマーカから運指を解析し抑えていると予測される指の位置を算出した。この情報をもとに MIDI データを編集することで補正を行った。また、ライン出力で録音した音源をもとに MIDI データの遅延(発音時刻のずれ)の補正を行った。図 10(上部)に示すように、アナログ音源の発音開始時刻を s1、MIDI 音源の発音開始時刻を s2 としたとき、MIDI 音源の遅延時間(発音時刻のずれ)=$s2 - s1$と表せる。
したがって、MIDI 音源の遅延は、MIDI 音源の発音開始時刻を s2 から s1 に補正する(s2= s1)ことで考慮できる(MIDI 音源の遅延時間(発音時刻のずれ)=$s1 - s1$=0)。

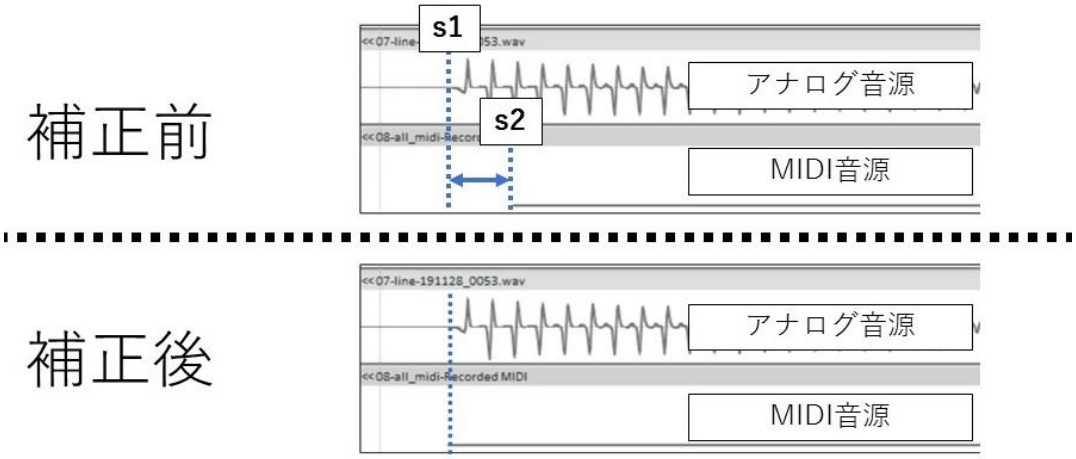

図 10.MIDI 音源の遅延補正結果

4.3 演奏者の熟達度

演奏者の熟達度を考慮するために演奏フレーズの正解音源(MIDI データ)と 4.2 で補正した MIDI データのマッチング処理を音高に関する評価、欠落音に関する評価、撥弦時刻のずれに関する評価の 3 つを評価項目とし、演奏者の演奏から該当するパラメータを抽出し、各パラメータに閾値処理を行いフレーズに対する熟練度を算出した。同様にその演奏者が演奏したすべてのフレーズに対して値を算出し、その平均値を演奏者の熟達度とした。7 人の演奏者(A,B,C,D,E,F,G)の熟達度を上記のマッチング処理を行い求めた結果を表 5 に示す。また、演奏者ごとの演奏難易度

分布を図 11 に示す。図 11 より、演奏者の熟達度が高くなるにつれ、演奏難易度が「簡単」と判断している件数の割合が多くなり、演奏者の熟達度が低くなるにつれ「難しい」と判断している件数の割合が多くなっていることから、実験で収集した演奏難易度は妥当であったと考えられる。

表 5. 7 人の演奏者の熟達度

演奏者	A	B	C	D	E	F	G
熟達度	4.92	5.02	5.82	6.25	7.29	7.52	8.56

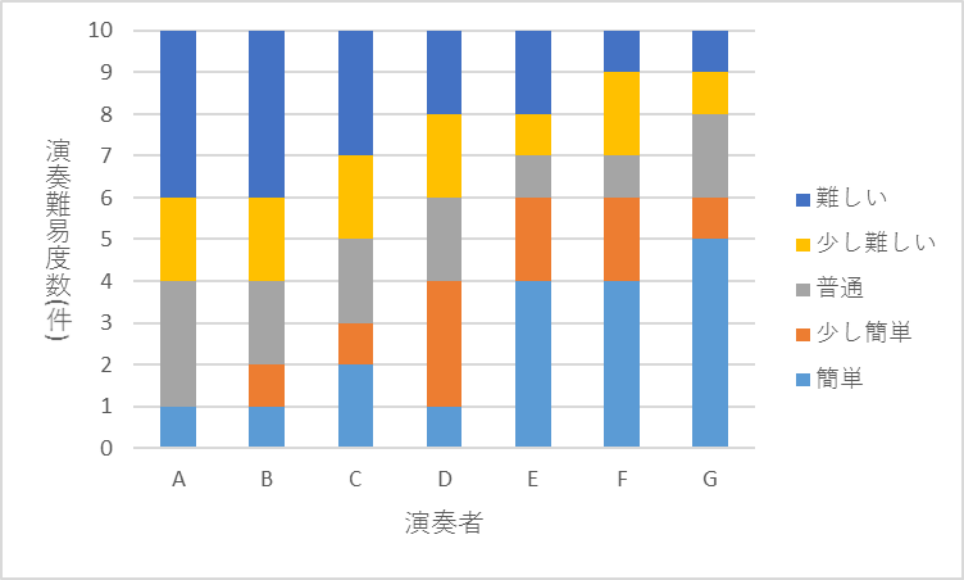

図 11.演奏難易度の分布

4.4 予測式構築結果

演奏者の熟達度に応じた演奏難易度予測式(提案式 1)を構築するために、重回帰分析における目的変数と説明変数よして次のデータを用いた。目的変数には 4.1 で述べた 70 件の演奏難易度を用いた。説明変数には、演奏者の熟達度として 4.3 で求めた値と従来の難易度評価[2,3]で使用されている楽譜的特徴量すべて用いた。

構築された提案式 1 の偏回帰係数と検定結果を表 6 に示す。説明変数中に多重共線性は見られなかった。提案式 1 の相関係数 R は 0.54、寄与率R^2は 0.29 であった。表 6 の標準化係数より、フレーズに対する熟練度が最も予測値に影響を与えていることがわかった。これにより、演奏者の熟達度を考慮した演奏難易度測定のためにフレーズに対する熟練度を素性に加える有効性が示唆された。

表 6.提案式 1 の偏回帰係数と検定結果

	偏回帰係数	標準化係数	t 検定値	p値
BPM	-0.0068	-0.0310	-0.3315	0.74
ノーツ密度	0.1190	0.0166	0.1619	0.87
ノーツ長:分散	0.0000	-0.1131	-0.7257	0.48
フレット番号:分散	-0.0363	-0.0628	-0.6509	0.52
弦番号:分散	0.6059	0.1400	0.6438	0.53
特集奏法の割合	-2.4858	-0.1349	-1.1870	0.25
演奏者の熟達度	-5.1799	-0.8521	-8.9454	<0.01
切片	6.7081	0.0000	4.0068	<0.01

第5章　予測式の評価

Leave-one-out-cross-validiation により提案式 1 の評価を行った。また、各従来の予測手法で用いられる素性のみで、重回帰式を構築し提案式との比較を行った。フレーズに対する熟練度を加えた影響を調べるために、提案式 1 の素性からフレーズに対する熟練度を除外して提案式 2 を構築した。提案式 2 を式(11)に示す。

$$Y = \sum_{i=1}^{6} a_i X_i \tag{11}$$

また、演奏者は 3.1 での実験とは別に 10 フレーズ用意し、演奏は行わず試聴のみで 3.1 同様の演奏難易度を判断した。上記の演奏者が付与した演奏難易度と提案式および従来式とのスピアマン相関係数を比較し、その結果を表 7 に示す。表 7 より、藤井ら[3]が難易度評価に用いた素性のみで回帰式を構築したとき、相関係数 0.22 であった。また、齋藤ら[2]が難易度評価に用いた素性のみで回帰式を構築したときの相関係数は 0.28 であった。さらに、提案式 2 の相関係数が 0.31 であったことから、ベースフレーズの演奏難易度評価に従来手法[2,3]の素性をすべて用いることが有用であると考えられる。また、提案式 1 の相関係数が 0.54 であったことから、難易度評価の素性にフレーズに対する熟練度を用いない場合より、用いた場合のほうが演奏難易度との相関が高くなったことがわかる。このことから熟達度を考慮した演奏難易度測定のためにフレーズに対する熟練度を素性に加えることの有効性が示唆された。

表 7.提案式と従来の予測手法の比較

	素性							相関係数
	BPM	ノーツ密度	ノーツ長:分散	フレット番号:分散	弦番号：分散	特殊奏法の割合	フレーズに対する熟練度：平均値	
提案式 1	○	○	○	○	○	○	○	0.54
提案式 2	○	○	○	○	○	○		0.31
齋藤	○	○	○	○	○			0.28
藤井	○		○	○		○		0.22

第6章　おわりに

本研究では、演奏者の熟達度を考慮した演奏難易度評価を行った。予測式の構築には重回帰分析を用いて、目的変数には従来の演奏難易度評価に用いられた素性の組み合わせと演奏者の特徴としてフレーズに対する熟練度を用いた。構築した予測式の重相関係数 R は 0.54、寄与率R^2は 0.29 であった。予測式の評価は Leave-one-out-cross-validiation で行った。また、従来手法の素性のみで重回帰式を構築し比較を行った。その結果、重回帰式の素性にフレーズに対する熟練度を加えることで相関係数が 0.31 から 0.54 に向上したため、予測式の素性にフレーズに対する熟練度を加えることが有用であると示唆された。

今後は、フレーズだけでなく楽曲の難易度評価やそれらを用いた独学での練習者の熟達度に応じた課題曲を自動選定する練習支援システムの検討などを行っていく予定である。また、さらなる予測式の性能の向上のために、よりベース演奏に関連するような説明変数の検討など行っていく予定である。

謝辞

本研究を進めるにあたり多くのご指導、ご支援を頂きました三好　力教授に深く感謝致します。また、三好研究室の皆様には、研究に対する様々なアドバイスや相談に多大なご協力を頂いたことに深く感謝致します。

参考文献

[1]”教則本を利用したギターフレーズの難易度推定” 森田花野、小泉悠馬、伊藤克亘、第 75 回全国大会講演論文集、2013/03/06
[2]”ギター練習支援システムのためのフレーズ難易度評価”齋藤蕗生、島田広、田村直良、情報処理学会研究報告、2018/11/21
[3]”SMF 解析による楽曲の難易度判定” 藤井ほのか、齋藤康之、研究報告音楽情報科学(MUS)、2016/07/30
[4]”演奏ルールを用いたウッドベースのための実時間運指取得システムの設計と実装” 澤光映、竹川佳成、寺田努、塚本昌彦、コンピュータソフトウェア、2010/01/26
[5] ”Real-Time Guitar Chord Recognition SystemUsing Stereo Cameras for SupportingGuitarists” Chutisant Kerdvibulvech、Hideo Saito、In Proceeding of 10th International Workshop on Advanced Image Technology 2007 (IWAIT'07)、2007/01
[6]”MIDI 楽器を用いたドラム演奏練習支援システムの提案” 岩見直樹、三浦雅展、情報処理学会研究報告音楽情報科学 (MUS)、2007/10/11
[7]”ドラムとベースの共同演奏における演奏リズムの解析” 中黒大輔、山本知仁、 ヒューマンインタフェース学会論文誌、2009/02/25
[8]”ピアノ初心者のための独学支援システムの提案” 松原正樹、遠山紀子、斎藤博昭、情報処理学会研究報告、2011/02/15
[9]”運指認識技術を活用したピアノ演奏学習支援システムの構築” 竹川佳成、寺田努、塚本昌彦、情報処理学会論文誌、2006/02/23
[10]”習熟度を考慮した複数楽譜からのピアノ譜生成手法の提案” 藤田顕次、大野博之、稲積宏誠、情報処理学会研究報告、2008/09/22
[11]”ギター演奏者の習熟度に合わせた音響信号からのタブ譜自動生成” 矢澤一樹、糸山克寿、奥乃博、情報処理学会研究報告、2013/09/01
[12]”ギター演奏音からの難易度調整可能なタブ譜自動生成システム” 矢澤一樹、糸山克寿、奥乃博、情報処理学会第 76 回全国大会、2014/03/11

付録

・RGB から HSV への変換

```
void colorExtraction(cv::Mat* src, cv::Mat* dst,
	int code,
	int ch1Lower, int ch1Upper,
	int ch2Lower, int ch2Upper,
	int ch3Lower, int ch3Upper
	)
{
	cv::Mat colorImage;
	int lower[3];
	int upper[3];

	cv::Mat lut = cv::Mat(256, 1, CV_8UC3);

	cv::cvtColor(*src, colorImage, code);

	lower[0] = ch1Lower;
	lower[1] = ch2Lower;
	lower[2] = ch3Lower;

	upper[0] = ch1Upper;
	upper[1] = ch2Upper;
	upper[2] = ch3Upper;

	for (int i = 0; i < 256; i++){
		for (int k = 0; k < 3; k++){
			if (lower[k] <= upper[k]){
				if ((lower[k] <= i) && (i <= upper[k])){
					lut.data[i*lut.step+k] = 255;
				}else{
					lut.data[i*lut.step+k] = 0;
				}
			}else{
				if ((i <= upper[k]) || (lower[k] <= i)){
					lut.data[i*lut.step+k] = 255;
				}else{
					lut.data[i*lut.step+k] = 0;
				}
			}
		}
	}
```

・指先に付与したカラーマーカの色抽出

```
for (int y = 0; y < hsv_video.rows; y++) {
                                    for (int x = 0; x < hsv_video.cols; x++) {
                                    hue = hsv_video.at<Vec3b>(y, x)[0];
                                    sat = hsv_video.at<Vec3b>(y, x)[1];
                                    val = hsv_video.at<Vec3b>(y, x)[2];
// 居留地マップの検出
if ((hue < 35 && hue > 20) && sat > 127) {
yellow_img.at<uchar>(y, x) = 255;
                                    }
else {
yellow_img.at<uchar>(y, x) = 0;
                                }
                    }
}
//人差し指
for (int y = 0; y < hsv_video.rows; y++) {
for (int x = 0; x < hsv_video.cols; x++) {
hue = hsv_video.at<Vec3b>(y, x)[0];
sat = hsv_video.at<Vec3b>(y, x)[1];
val = hsv_video.at<Vec3b>(y, x)[2];
// 居留地マップの検出
if ((hue > 155 && hue < 165) && sat > 127) {
blue_img.at<uchar>(y, x) = 255;
                    }
else {
blue_img.at<uchar>(y, x) = 0;
                    }
}
}
//中指
for (int y = 0; y < hsv_video.rows; y++) {
for (int x = 0; x < hsv_video.cols; x++) {
hue = hsv_video.at<Vec3b>(y, x)[0];
sat = hsv_video.at<Vec3b>(y, x)[1];
val = hsv_video.at<Vec3b>(y, x)[2];
// 居留地マップの検出
if ((hue > 35 && hue < 45) && sat > 100) {
green_img.at<uchar>(y, x) = 255;
}
else {
green_img.at<uchar>(y, x) = 0;
                    }
}
}

//薬指
for (int y = 0; y < hsv_video.rows; y++) {
for (int x = 0; x < hsv_video.cols; x++) {
hue = hsv_video.at<Vec3b>(y, x)[0];
sat = hsv_video.at<Vec3b>(y, x)[1];
val = hsv_video.at<Vec3b>(y, x)[2];
// 居留地マップの検出
if ((177 > 35 && hue < 197) && sat > 100) {
green_img.at<uchar>(y, x) = 255;
}
else {
green_img.at<uchar>(y, x) = 0;
                    }
                }
            }
//小指
for (int y = 0; y < hsv_video.rows; y++) {
for (int x = 0; x < hsv_video.cols; x++) {
hue = hsv_video.at<Vec3b>(y, x)[0];
sat = hsv_video.at<Vec3b>(y, x)[1];
val = hsv_video.at<Vec3b>(y, x)[2];
// 居留地マップの検出
if ((hue > 105 && hue < 115) && sat > 100) {
green_img.at<uchar>(y, x) = 255;
}
else {
green_img.at<uchar>(y, x) = 0;
                    }
                }
            }
imshow("人差し指", red_img);
imshow("中指", yellow_img);
imshow("薬指", blue_img);
imshow("小指", green_img);
                    }
```

・AR マーカを基準とし指板領域の認識

```
void draw() {

  // camera capture
  if(cam.available() == false) {
    return;
  }
  cam.read();// capturing image

  // Start of AR process
  mm.detect(cam);// detecting marker
  mm.drawBackground(cam);// drawing captured image on background

  //drawing image
  blendMode(BLEND);
  imageMode(CENTER);
  scoreImgCl.draw();

  blendMode(ADD);
  noteImgCl.draw();

  blendMode(BLEND);
  imageMode(CORNER);
  blendMode(BLEND);

  // if marker[0] is not exist within the image
  if(mm.isExist(0) == false) {
    return;
  }

  // starting coordinate projection based on marker[0]
  mm.beginTransform(0);

    blendMode(ADD);
    imageMode(CENTER);
    hint(DISABLE_DEPTH_TEST);

    scale(-0.8*ajstZoom,-0.8*ajstZoom);
    rotateX(ajstRX);
    rotateY(ajstRY);
    rotateZ(ajstRZ);
    translate(300+ajstX, 100+ajstY, 20+ajstZ);
    fletImgCl.drawFlet();
    fletImgCl.drawCircles();
    fletImgCl.drawPickups();

    viewport = (PMatrix3D)getMatrix();

    translate(-(300+ajstX), -(100+ajstY), -(20+ajstZ));
    rotateZ(ajstRZ);
    rotateY(ajstRY);
    rotateX(ajstRX);
    scale(-1.25/ajstZoom,-1.25/ajstZoom);

    hint(ENABLE_DEPTH_TEST);
    imageMode(CORNER);
    blendMode(BLEND);
```

```
  mm.endTransform();
  // End of AR process
}

PImage createLight(float rPower, float gPower, float bPower) {
  int side = 200;
  float center = side / 2.0;
  PImage img = createImage(side, side, RGB);

  for (int y = 0; y < side; y++) {
    for (int x = 0; x < side; x++) {
      //float distance = sqrt(sq(center - x) + sq(center - y));
      float distance = (sq(center - x) + sq(center - y)) / 50.0;
      int r = int( (255 * rPower) / distance );
      int g = int( (255 * gPower) / distance );
      int b = int( (255 * bPower) / distance );
      img.pixels[x + y * side] = color(r, g, b);
    }
  }
  return img;
}

void mousePressed() {
  noteImgCl.newNote(0,0);
}

PImage createNoteImg(String path, float rPower, float gPower, float
bPower) {

  PImage noteImg = loadImage(path);
  for (int y = 0; y < noteImg.height; y++) {
    for (int x = 0; x < noteImg.width; x++) {
      color c = noteImg.pixels[x + y * noteImg.width];
      if(red(c) < 20){
        noteImg.pixels[x + y * noteImg.width] =
color(255*rPower,255*gPower,255*bPower);
      }else{
        noteImg.pixels[x + y * noteImg.width] = color(0, 0, 0);
      }
    }
  }
  noteImg.filter(BLUR, 8);

  return noteImg;
}

class ScoreImg {
  PImage scoreImg;
  PImage backImg;
  PImage numberImg0;
  int numberInd[] = new int[6];
  PImage numberImg[] = new PImage[120];
  int ix[] = new int[120];
  int iy[] = new int[120];
  int flag[] = new int[120];
  int backSizeX = 650;
  int backSizeY = 130;

  int baseSizeX = 600;
  int baseSizeY = 100;
  int offsetX = width/2 + baseSizeX/2;
  int offsetY = baseSizeY/2 + 100;
  float vx = -4;
  int fcSpeed = 100;
  ScoreImg() {
    init();
    newNumber(4,4);
  }
  void draw() {
    blendMode(ADD);
    imageMode(CENTER);
    image(backImg, offsetX, offsetY);

    blendMode(BLEND);
    imageMode(CENTER);
    image(scoreImg, offsetX, offsetY);

    for (int i = 0; i < 120; i++) {
      if(flag[i] == 1){
        ix[i] += vx;
        if(ix[i] < width/2+30)flag[i]=0;
        image(numberImg[i], ix[i], iy[i]);
      }
    }

  }
  void init(){
    scoreImg = createBase(0.1,0.1,1.0,0.7);
    backImg = createBack();
  }

  PImage createBack() {
    int sizeX = backSizeX;
    int sizeY = backSizeY;

    PImage img = createImage(sizeX, sizeY, ARGB);

    for (int y = 0; y < sizeY; y++) {
      for (int x = 0; x < sizeX; x++) {
        img.pixels[x + y * sizeX] = color(255,255,255,120);
      }
    }
    return img;
  }
  PImage createBase(float rPower, float gPower, float bPower, float aPower)
{
    int sizeX = baseSizeX;
    int sizeY = baseSizeY;

    PImage img = createImage(sizeX, sizeY, ARGB);

    for (int y = 0; y < sizeY; y++) {
      for (int x = 0; x < sizeX; x++) {
        if(y % int(sizeY/6) == int(sizeY/12)){
          img.pixels[x + (y-1) * sizeX] =
color(255*rPower,255*gPower,255*bPower, 255*aPower);
          img.pixels[x + y * sizeX] =
color(255*rPower,255*gPower,255*bPower, 255*aPower);
          //img.pixels[x + (y+1) * sizeX] =
color(255*rPower,255*gPower,255*bPower, 255*aPower);
        }
      }
    }

    PGraphics pg = createGraphics( sizeX, sizeY );

    pg.beginDraw();
    pg.image( img, 0, 0 );
    pg.noFill();
    pg.fill(255*rPower,255*gPower,255*bPower, 255*aPower);
    pg.textSize(36);
    pg.textAlign(CENTER);
    pg.text("T", 20, sizeY/2 + 13 - 27);
    pg.text("A", 20, sizeY/2 + 13 + 0);
    pg.text("B", 20, sizeY/2 + 13 + 27);
    pg.endDraw();

    img.pixels = pg.pixels;

    return img;
  }

  PImage createNumberImg(int n, float rPower, float gPower, float bPower,
float aPower) {
    int sizeX = 100;
    int sizeY = 100;

    PImage img = createImage(sizeX, sizeY, ARGB);

    PGraphics pg = createGraphics( sizeX, sizeY );

    pg.beginDraw();
    pg.image( img, 0, 0 );
    pg.noFill();
    pg.fill(255*rPower,255*gPower,255*bPower, 255*aPower);
    pg.textSize(20);
    pg.textAlign(CENTER);
    pg.text(str(n), sizeX/2, sizeY/2-10);
    pg.endDraw();

    img.pixels = pg.pixels;

    return img;
  }

  void newNumber(int string, int flet){

    int interval = 30;
    int j = numberInd[string-1] + 20*(string-1);
    int diff = offsetX + baseSizeX/2 - ix[j];
    if(diff < interval){
      for (int jj = 0; jj < 120; jj++) {
        ix[jj] -= interval - diff;
      }
    }

    numberInd[string-1] += 1;
    if(numberInd[string-1] >= 20)numberInd[string-1]=0;
    int i = numberInd[string-1] + 20*(string-1);

    numberImg[i] = createNumberImg(flet, 0.4,0.4,1.0,0.9);
    ix[i] = offsetX + baseSizeX/2;
    iy[i] = offsetY + int((string-2.5)*baseSizeY/6);
    flag[i] = 1;
  }
}


class NoteImg {
  int numberInd = 0;
  PImage noteImg[] = new PImage[100];
  int vx[] = new int[100];
  int vy[] = new int[100];
```

```
  int ix[] = new int[100];
  int iy[] = new int[100];
  int flag[] = new int[100];

  int scorePosX = width/2 + 600;
  int scorePosY = 150;

  NoteImg() {
    init();
  }
  void draw() {
    for (int i = 0; i < 100; i++) {
      if(flag[i] == 1){
        ix[i] += vx[i];
        iy[i] += vy[i];
        if(ix[i] > scorePosX-30)flag[i]=0;
        image(noteImg[i], ix[i], iy[i]);
      }
    }
  }
  void init(){
    for (int i = 0; i < 100; i++) {
      String path;
      if(random(0.0, 1.0) < 0.5){
        path = "noteImg.jpg";
      }else{
        path = "noteImg2.jpg";
      }
      float rPower = random(0.2, 0.8);
      float gPower = random(0.2, 0.8);
      float bPower = random(0.2, 0.8);
      noteImg[i] = createNoteImg(path, rPower, gPower, bPower);
      noteImg[i].resize(noteImg[i].width/4, noteImg[i].height/4);
      flag[i] = 0;
    }

  }

  void newNote(int inx, int iny){
    numberInd += 1;
    if(numberInd >= 100)numberInd=0;

    ix[numberInd] = inx;
    iy[numberInd] = iny;
    vx[numberInd] = (scorePosX-inx)/20 ;
    vy[numberInd] = (scorePosY-iny)/20 ;

    flag[numberInd] = 1;
  }

  PImage createNoteImg(String path, float rPower, float gPower, float
bPower) {

    PImage noteImg = loadImage(path);

    for (int y = 0; y < noteImg.height; y++) {
      for (int x = 0; x < noteImg.width; x++) {
        color c = noteImg.pixels[x + y * noteImg.width];
        if(red(c) < 20){
          noteImg.pixels[x + y * noteImg.width] =
color(255*rPower,255*gPower,255*bPower);
        }else{
          noteImg.pixels[x + y * noteImg.width] = color(0, 0, 0);
        }
      }
    }
    noteImg.filter(BLUR, 8);

    return noteImg;
  }
}

class FletImg {
  PImage fletImg;
  PImage circleImg[] = new PImage[6];
  PImage PickupImg[] = new PImage[24];
  int sizeX = 1200;
  int sizeY = 120;
  int nf, nfys, nfye;
  float nfya = 0.15;
  int fletLengthX = 1050, fletLengthY = 100;
  int ix[] = new int[6];
  int iy[] = new int[6];
  int ix_p[] = new int[24];
  int iy_p[] = new int[24];
  int pickupFlag = 0;

  FletImg() {
    createFlet(0.0,0.0,1.0);
    createCircles(0.2,1.0,0.2);
    createPickups(0.2,1.0,0.2);
  }
  void drawFlet() {
    image(fletImg, 0, 0);
  }
  void drawCircles() {
    for (int i = 0; i < 6; i++) {
      image(circleImg[i], ix[i], iy[i]);
    }
  }
  void drawPickups() {
    if(pickupFlag == 0){
      return;
    }
    for (int i = 0; i < 24; i++) {
      image(PickupImg[i], ix_p[i], iy_p[i]);
    }
  }
  void changePosition(int string, float flet) {
    ix[string-1] = getPositionX(flet);
    iy[string-1] = getPositionY(string, flet);
  }
  void createCircles(float rPower, float gPower, float bPower){
    for (int i = 0; i < 6; i++) {
      circleImg[i] = createCircleImg(rPower, gPower, bPower);
      ix[i] = 10000;
      iy[i] = 10000;
    }
  }

  void createPickups(float rPower, float gPower, float bPower){
    for (int i = 0; i < 6; i++) {
      PickupImg[i] = createCircleImg(rPower, gPower, bPower);
      ix_p[i] = getPositionX(24.5);
      iy_p[i] = getPositionY(i+1, 24.5);
    }
    for (int i = 6; i < 12; i++) {
      PickupImg[i] = createCircleImg(rPower, gPower, bPower);
      ix_p[i] = getPositionX(31);
      iy_p[i] = getPositionY(i-5, 31);
    }
    for (int i = 12; i < 18; i++) {
      PickupImg[i] = createCircleImg(rPower, gPower, bPower);
      ix_p[i] = getPositionX(42);
      iy_p[i] = getPositionY(i-11, 42);
    }
    for (int i = 18; i < 24; i++) {
      PickupImg[i] = createCircleImg(rPower, gPower, bPower);
      ix_p[i] = getPositionX(50);
      iy_p[i] = getPositionY(i-17, 50);
    }
  }
  PImage createCircleImg(float rPower, float gPower, float bPower){
    int side = 200;
    float center = side / 2.0;

    PImage img = createImage(side, side, RGB);

    for (int y = 0; y < side; y++) {
      for (int x = 0; x < side; x++) {
        float distance = (sq(center - x) + sq(center - y)) / 50.0;
        int r = int( (255 * rPower) / distance );
        int g = int( (255 * gPower) / distance );
        int b = int( (255 * bPower) / distance );
        img.pixels[x + y * side] = color(r, g, b);
      }
    }
    return img;
  }

  void createFlet(float rPower, float gPower, float bPower){
    fletImg = createImage(sizeX, sizeY, RGB);

    for (int f = 0; f < 22; f++) {
      nf = int(pow(2,(12-float(f))/12)*fletLengthX/2);
      nfys = 20 + int(fletLengthY * (float(nf)/float(fletLengthX)*nfya));
      nfye = 20 + int(fletLengthY * (1 - float(nf)/float(fletLengthX)*nfya));
      for (int y = nfys; y < nfye; y++) {
        fletImg.pixels[nf + y * sizeX] =
color(255*rPower,255*gPower,255*bPower);
        for (int i = 0; i < 5; i++) {
          fletImg.pixels[nf+i + y * sizeX] =
color(255*rPower,255*gPower,255*bPower);
          fletImg.pixels[nf-i + y * sizeX] =
color(255*rPower,255*gPower,255*bPower);
        }
      }
    }

    fletImg.filter(BLUR, 3);

    for (int f = 0; f <= 22; f++) {
      nf = int(pow(2,(12-float(f))/12)*fletLengthX/2);
      nfys = 20 + int(fletLengthY * (float(nf)/float(fletLengthX)*nfya));
      nfye = 20 + int(fletLengthY * (1 - float(nf)/float(fletLengthX)*nfya));
      for (int y = nfys; y < nfye; y++) {
        fletImg.pixels[nf + y * sizeX] =
color(255*rPower,255*gPower,255*bPower);
        for (int i = 0; i < 3; i++) {
          fletImg.pixels[nf+i + y * sizeX] =
color(255*rPower,255*gPower,255*bPower);
          fletImg.pixels[nf-i + y * sizeX] =
color(255*rPower,255*gPower,255*bPower);
        }
```

```
        }
      }

      for (int f = 0; f <= 22; f++) {
        nf = int(pow(2,(12-float(f))/12)*fletLengthX/2);
        nfys = 15 + int(fletLengthY * (float(nf)/float(fletLengthX)*nfya));

drawText(fletImg,nf,nfys,str(f),17,int(255*rPower),int(255*gPower),int(255
*bPower));
      }
   }

   void changeColor() {
      createFlet(random(0.2, 0.8), random(0.2, 0.8), random(0.2, 0.8));
   }

   int getPositionX(float flet) {
      int x;
      x = int(pow(2,(12-(flet-0.5))/12)*fletLengthX/2) - sizeX/2;
      return x;
   }
   int getPositionY(int string, float flet) {
      int y;
      int nf = int(pow(2,(12-flet)/12)*fletLengthX/2);
      int nfy = 20 + int(fletLengthY * (float(nf)/float(fletLengthX)*nfya));
      int fletLengthYEach = int(fletLengthY * (1 -
float(nf)/float(fletLengthX)*nfya*2));
      y = nfy + int((6.5-float(string))/6 * fletLengthYEach) - sizeY/2;
      return y;
   }

   void drawText(PImage img_out, int x, int y, String txt,int size,int r,int
g,int b) {

      PGraphics pg = createGraphics( img_out.width, img_out.height );

      pg.beginDraw();
      pg.image( img_out, 0, 0 );
      pg.noFill();
      pg.fill(r, g, b);
      pg.textSize(size);
      pg.textAlign(CENTER);
      pg.text(txt, x, y);
      pg.endDraw();

      img_out.pixels = pg.pixels;
   }


}
```