Vol.2017-MUS-114 No.24
2017/2/27

Strummer：インタラクティブなギターコード練習システム

有賀 竣哉[1,a)]　後藤 真孝[2,b)]　矢谷 浩司[1,c)]

概要：楽器の演奏は多くの人が望むスキルであり，様々な練習方法や教材が考案されているが，その種類の多さゆえ初心者は何を練習すればよいのか迷いがちである．さらに独学では自分の演奏の可否を判断して修正することが難しく，これらはモチベーション低下につながる問題となっている．本稿では，データドリブンでインタラクティブなギターコード練習システム「Strummer」を提案する．Strummer システムの開発にあたり，コードラベルが付与された 727 曲分の楽曲データを解析し，コードの出現率と難易度を考慮した曲の重要性の尺度を新たに定義した．これをシステム中で用いることで，ユーザは簡単かつ重要な楽曲から練習が可能になる．さらに，本システムは音響信号解析によってユーザが正しいコードを弾いたのかを認識してフィードバックする機能を有しており，インタラクティブかつ段階的な練習を提供する．実験では 5 人のギター初心者に計 5 時間ずつ本システムを用いてギター練習をしてもらい，インタビューやアンケートを実施し，本システムの評価を行った．

Strummer: An Interactive System for Guitar Chord Practice

SHUNYA ARIGA[1,a)]　MASATAKA GOTO[2,b)]　KOJI YATANI[1,c)]

1. はじめに

楽器の演奏は複雑な手の動きを必要とするため，習得には継続的な練習が不可欠である．しかしながら楽器の演奏は，その複雑性や環境などの要因から多くの壁が存在しており，初心者が挫折しやすい．特に，ギターやピアノなどの和音楽器を用いて楽曲を演奏する際には，コード (和声) の演奏が重要である．こうしたコードの練習は単調になりやすいので，学習者にとってつまらなく感じてしまうことがある．

楽器演奏初心者に向けた演奏支援システムや，練習支援システムは数多く提案されてきている．また，オンライン上の学習素材や学習支援サービスも多数存在する．例えば，Youtube 上などにある楽曲の音源を指定するとコード譜が生成される chordify [*1]や，ギターのオンラインレッスンを提供する GuitarTricks [*2]，Yousician [*3]などがある．しかし，楽器練習に向けたこうしたコンテンツは大量に存在するため，初心者は何を練習すればいいのか迷ってしまいやすい．例えば，初心者は練習する曲の難易度を見積もることが難しいため，どの曲が簡単であるか分からない場合がある．初心者の適切な練習のためには，適切な指標に基づいて練習コンテンツの重要性を判断したうえで，ユーザが練習しやすいようなチュートリアルを提示することが重要である．

そこで本稿では，データドリブンでインタラクティブなギターコード練習システム「Strummer」を提案する．Strummer は，与えられた楽曲データの中からコードの難易度・出現頻度・音価などの特徴を元に，練習コンテンツとしての楽曲の重要性を計算する．これをシステム中で用いることで，ユーザは簡単かつ重要な楽曲から練習が可能になる．Strummer が提供する練習はユーザの好みで選ぶこ

[1] 東京大学 大学院工学系研究科
Interactive Intelligent Systems Laboratory,
Graduate School of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
[2] 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
[a)] ariga@iis-lab.org
[b)] m.goto@aist.go.jp
[c)] koji@iis-lab.org

[*1] https://chordify.net/
[*2] https://www.guitartricks.com/
[*3] https://get.yousician.com/

とができ，練習した曲をもとにより多くの曲が弾けるようなアルゴリズムになっている．ユーザが段階を踏んで曲全体を練習できるように，システムは 6 つのステージに細分化したチュートリアルを楽曲データをもとに自動で作成する．さらに，本システムは音響信号解析によってユーザがコードを正しく弾けたのかを認識してフィードバックする機能を有しており，ユーザはインタラクティブな練習が可能である．

2. 関連研究

教育学的に言えば，学習者は楽器に触れはじめる時から指導者のもとで直接教わるべきであり，楽器の持ち方や弦の押さえ方など基本的な事項から教わることが望ましい [3]．しかしながら指導者を探して楽器を教わるのはコストが高く，気軽に行うのは難しい．ギター演奏などの楽器演奏の習得には，学習者のモチベーションを保ち，自己学習を推進することも重要である [6]．

2.1 楽器練習支援

楽器演奏の習得には少なからぬ練習と努力が必要であるため，コンピュータを用いてユーザの楽器練習をサポートする研究がさかんに行われてきている．ピアノ初心者に向けた総合的な演奏支援システムである Piano Tutor [2] を始めとして様々な支援システムが提案されてきた．近年はウェアラブルデバイスを用いた手法，ならびに拡張現実 (Augmented Reality, AR) の技術を用いたシステムが多く見られる．AR を用いた演奏支援の研究には，ギター演奏において弦の押さえ方を AR 技術により直感的に提示するシステムを提案した Motokawa らの研究 [4] などがある．そのほかにも，五線譜の理解を助けるピアノ学習支援システムを提案した Takegawa らの研究 [7] や，ウェアラブルセンサーを用いて学習者のバイオリンの演奏姿勢を検知し，3D モデリングした演奏者のリプレイ動画を見せてフィードバックを与える Ng らの研究 [5] など，多様な楽器に対応したシステムが提案されている．

2.2 ゲームシステム

ユーザの練習をモチベートするような商用のゲームも存在しており，我々が Strummer をデザインする際の参考となっている．Guitar Hero[*4] はギターを簡略化したギター型デバイスを採用しており，ユーザは画面に流れてくるノーツに従ってボタンを押す．Rocksmith[*5] は Guitar Hero と似たゲームシステムであるが，ユーザが実際のエレクトリック・ギターを接続可能なゲームである．Yousician はピアノ，ギター，ウクレレの 3 種の和音楽器に向けたオンライントレーニングシステムである．このシステムは段階的な練習を提供し，さらにユーザの演奏を即座に解析してフィードバックを提供する．Yousician は我々の Strummer システムに最も近いシステムであるが，我々はギターコード演奏向けにデータドリブンな手法を取り込むことで，楽曲セット中から練習しやすく重要な曲を選んで練習ができる点で異なっている．

[*4] https://www.guitarhero.com/
[*5] http://rocksmith.ubi.com/rocksmith/

3. 使用するデータセット

3.1 コードデータ

本研究では，ポピュラー音楽のコード進行のデータセットとして Billboard データセット [1] を用いている．Billboard データセットは，アメリカのポピュラー音楽のヒットチャートである Billboard の Hot 100 チャート[*6] に過去のある期間にランクインした曲から選んだ 890 曲分のデータを含んでおり，タイトル・アーティストなどの基本情報と，A メロ・B メロ・サビのような音楽の構造的情報，ならびにコード名が記されている．本データセットには楽曲のジャンル情報が無かったため，我々は Wikipedia の情報をもとに楽曲のジャンルを得た．

本データセットは詳細で正確な記述がなされている一方で，一般的にコード譜などに使われている表記方法ではなく，ユーザに提示するには不適である．そこで，Strummer は データセット中のコードを簡略化してユーザに提示する．アドナインスコードを除く四和音以上のテンションコードを四和音に簡略化し，かつ表記が複雑すぎるコードはマイナーもしくはメジャーに簡略化した．簡略化の結果，全ての楽曲に含まれるコードが [maj, min, dim, aug, maj7, min7, 7, hdim7, minmaj7, maj6, min6, add9, sus4, 5] の 14 種類に絞られた．そのほか，曲の拍子は全て四分の四と仮定した．さらに，1 小節に含まれるコードを最大 2 つまでとし，それ以上のコードを含む小節は 3 つ目以降のコードを省略した．

3.2 コード難易度のスコア付け

本研究ではギターのコード演奏に向けた練習を提供するため，それぞれのコードを押さえる難易度，および次のコードを弾くために指を移動させる際の難易度を考慮する必要がある．本稿では前者を「**押弦難易度**」，後者を「**遷移難易度**」と定義する．

3.2.1 押弦難易度

我々はまず，2 名の評価者によって，システムで用いるコードを全て含んだ 180 種のコードについて押弦難易度のスコア付けを行った．スコア付けの基準は単純にコードの押さえ方のみを考慮することとし，5 段階のリッカート尺度（1:最も簡単, 5:最も難しい）を用いた．押弦難易度のスコア付け結果を表 1 の混同行列に示す．2 者のスコアは 52.2%が一致

[*6] http://www.billboard.com/charts/hot-100

しており，重み付け κ 係数は $\kappa = .81(95\%CI : [.7534, .8537])$ であるので，ほぼ一致していることが分かる．システムで用いるコードの難易度はこのスコアの単純平均とした．

3.2.2 遷移難易度

コードの組み合わせは非常に種類が多く，全てを人手でスコア付けするのは現実的ではない．そこで我々は遷移難易度のモデルを作成することとした．まず，コード自体の押弦難易度の決定時と同じように，同じ 2 名の評価者によってコードの遷移難易度のスコア付けを行った．対象とするコードの組み合わせは，データセット中で頻出する 100 組に加えて，ランダムに選んだ 30 組を合計した 130 組とした．スコア付けの基準は前後の指の移動の難易度のみを考慮することとし，前述の 5 段階のリッカート尺度を用いた．遷移難易度のスコア付け結果を表 1 の混同行列に示す．2 者のスコアは 60.0%が一致しており，重み付け κ 係数は $\kappa = .76(95\%CI : [.6615, .8359])$ であり，程よく一致していることが分かる．

我々はさらに，上記で得た遷移難易度のスコアを元に線形回帰を行い，全てのコードの組み合わせの遷移難易度を求めるモデルを作成した．まず 2 つの独立な変数より回帰を始め，AIC を基準に最適なモデルを設計した．結果として，以下に述べる独立な変数と，括弧内に示すその係数，および切片の値 (0.790) を得た．これらの変数は全て 99%の水準で有意であった．なお，以下で述べるコードの特徴は全て左手におけるものである．

- *diff_1* (0.310): 遷移先のコードの押弦難易度．
- *fig_num* (0.211): 中指・薬指・小指のうち，使う指の本数の増減を表す変数で，[-2, 2] の整数値をとりうる．人差し指は大半のコードに使われているためここでは考慮していない．
- $\log(move\_base + 1)$ (0.336): 左手の移動距離を表す変数に自然対数をとったもの．左手の位置は，バレーコードの場合はバレーするポジション番号を，そうでない場合は 0 となる．遷移前後の左手の位置の差の絶対値が *move_base* となる．
- *move_figR* (0.233): 薬指の相対移動距離．上述の左手の位置から見た薬指の相対位置を考え，前後のコードで薬指の相対位置のユークリッド距離を求めたものがこの変数である．なお，コードを押さえるのにその指を使わない場合，相対位置は 0 となる．
- *move_figL* (0.242): 小指の相対移動距離．薬指の相対移動距離である *move_figR* と同様に定義される．
- *barre_1* (0.481): 遷移先がバレーコードであれば 1，そうでなければ 0 となる変数．
- *slide* (-1.401): 両者ともバレーコードであれば 1，そうでなければ 0 となる変数．

設計した遷移難易度モデルは決定係数 $R^2 = .786$ を示した．このモデルで得られた出力を [1.0, 5.0] の値域に制限したものを最終的に遷移難易度とした．

表 1: コードの押弦難易度のスコア付けにおける混同行列

		評価者 B				
	スコア	1	2	3	4	5
評価者 A	1	15	20	1	1	0
	2	0	9	3	1	0
	3	0	6	33	13	0
	4	0	0	17	25	1
	5	0	0	1	22	12

表 2: コードの遷移難易度のスコア付けにおける混同行列

		評価者 B				
	スコア	1	2	3	4	5
評価者 A	1	18	11	4	1	0
	2	1	19	8	1	0
	3	0	9	21	8	0
	4	0	0	6	19	1
	5	0	0	0	3	1

4. 優先度付けアルゴリズム

本研究では学習者にとってのコードや楽曲の学習の有益性を示す指標を作成し，システムに組み込んだ．

4.1 Chord Primariness

コードの重要性を示す指標の Chord Primariness (*CP*) を，コードの出現率，および難易度を考慮した以下の式 (1) で定義した．

$$CP_{c_i} = -\sqrt{w_f \left(\frac{1}{f_{c_i}}\right)^2 + w_d {d_{c_i}}^2} \tag{1}$$

ただし，

$$f_{c_i} = \frac{N_{c_i}}{\sum_C N_{c_i}} \tag{2}$$

であり，f_{c_i} はあるコード c_i が与えられた楽曲集合中に出現する頻度を表す．さらに，d_{c_i} はコード c_i の押弦難易度を表す．w_f，w_d はそれぞれ出現頻度と難易度についての重み係数であり，これらを調整することで *CP* を自由に変更することが可能である．本システムではそれぞれ $w_f = 0.1$，$w_d = 10$ を採用した．

4.2 Song Primariness

上述の *CP* を用いて，曲の重要性の指標 Song Primariness (*SP*) を以下のように定義する．

$$SP = \sum_C CP_{c_i} + \sum_{T_f} TD + \sum_{T_h} TD \tag{3}$$

ただし，*TD* は遷移難易度 (Transition Difficulty) を表し，*C* は曲中に含まれる全コードを表す．そして T_f は遷移先のコードの音価が全音符であるコード遷移を，T_h は遷移先のコードの音価が二分音符であるコード遷移のうち，遷移難

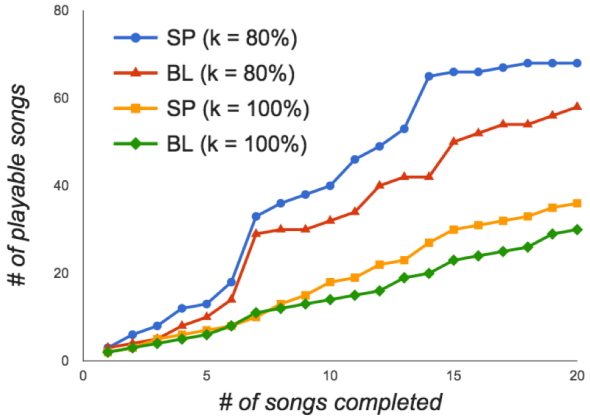

図 1: ポップスのジャンルの中で *k*%-playable の曲数を *SP* と *BL*(*TCD*) で比較したプロット

易度が全体の平均値を上回るコード遷移を表している．

この *SP* を用いて，コードの難易度，出現率，曲に含まれるコードの数などの複数の要素から，楽曲の練習における重要性を推し測ることができる．本システムでは *SP* に基づいて楽曲を順序付けし，ユーザは *SP* が高い順に楽曲を学習していくこととなる．

4.3 アルゴリズムの評価

SP の有用性を示すため，曲中に含まれるコードの押弦難易度の総計である *TCD* (Total Chord Difficulty) を定義した．*SP* と *TCD* の比較のため，我々はユーザがまだ練習していない未知の曲に対して “*k*%-playable” という指標を定義した．ユーザが既に練習してきたコードで，ある未知の曲中のコードのうち *k*%のコードを弾けるようになったという指標である．例えば $k = 100\%$ の曲は，まだ練習していない未知の曲であるにもかかわらず，曲中のコードは全て練習済みであるという曲である．少ない数の楽曲の練習で “*k*%-playable” である曲が多いならば，そのアルゴリズムが効率的な練習を提供可能であるといえる．

TCD をベースライン (*Baseline*,*BL*) とし，ポップスのジャンルの曲を *SP* と *TCD* で優先度順にソートし，学習者がその順に 1 曲ずつ 20 曲目まで練習していった場合の “*k*%-playable” の曲数をプロットしたものが図 1 である．これを見ると *SP* における “*k*%-playable” が $k = 80\%$，$k = 100\%$ のどちらの場合でも *BL* を上回っていることがわかる．これは，*SP* で優先順位を付けた曲を練習した方が，より広範囲の未知の曲をカバーできることを示している．Strummer はこの *SP* を用いて楽曲セット中の楽曲を優先度付けし，優先度が高い楽曲から練習することをユーザに促す．

5. Strummer システム

5.1 インターフェース

曲一覧

メイン画面は図 2(a) のように，曲を *SP* が高い順に並べている．ユーザは *SP* が高い曲から練習することとなる．メイン画面からそれぞれの曲の構成コードや進捗状況が見れるようになっている．

レッスン

図 2(a) から各曲に進むと，図 2(b) のようにレッスンの一覧が表示される．各曲は A メロ，B メロ，サビなどのセクションが繰り返される構造を持っている．Strummer はセクションの繰り返しを省いた練習を提供しており，1 つのセクションにつき 1 つのレッスンが自動生成される．

ステージ

各レッスンを選択すると図 2(c) のようにステージの一覧画面となる．全てのレッスンは後述する 6 つのステージで構成されており，ユーザは段階的に練習を行うことができる．

Listen ステージ

図 2(d) が示す Listen ステージでは，そのセクション全体のお手本を聴くことができる．ユーザはまず全体のコード進行を聴くことによって全体像を掴むことができる．

Chord ステージ

このステージでは各セクションに出現するコードを 1 つずつ練習することができる．図 2(e) のように，4 章で述べた *CP* の降順に，選んだセクション中に出現するコードが並んでいる．それぞれのコード練習画面に進むと図 2(f) のような画面になる．画面中央にダイアグラム (コードの押さえ方を示す図) が表示され，ユーザはこのダイアグラムを見ながらコードを押さえる．ユーザがスタートボタンをクリックすると，システムはコード解析モジュールを作動させ，ユーザがコードを正しく弾けているのかを判断するようになる．正しく弾けると図 2(g) の左下部のように OK ラベルが増え，失敗すると減るようになっており，OK ラベルが 3 つ貯まると次のコードに進める．

コードがうまく弾けず失敗が続いた場合，追加の練習が提供される．この追加練習は図 2(g) のように，押さえる弦の数を少ない本数から徐々に増やしていくようになっており，ユーザは簡略化された押さえ方から徐々に押さえる指を増やして練習できる．それでもうまく弾けなかった場合は，以後のステージではコードが簡略化して表示されるようになり，弾けなかった弦についての音解析が不問となり，代わりにシステムが当該の弦の音を補完して鳴らすようになる．後続のステージで徐々に長い区間を弾くことになるので，もしユーザが特定のコードが弾けないのに何度も弾くよう指示されたらストレスとなってモチベーション低下につながる可能性がある．この機能は以上の問題を考慮して設定した．例として，図 2(h) では *Bm* コードの一部が省略されて表示されている．

Transition ステージ

図 2(i) のように，コードの遷移についての練習が Chord ステージと同様に提供される．

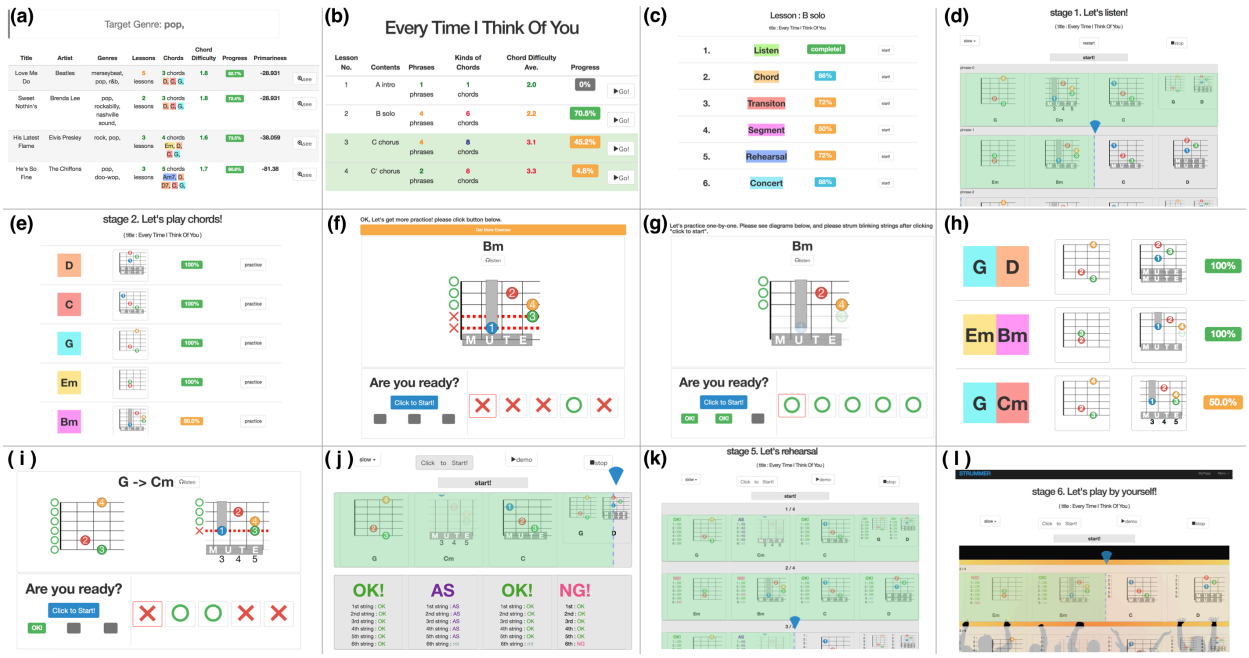

図 2: Strummer の画面．(a) 練習曲は *SP* 降順に並んでいる．(b) 各曲がいくつかのセクションに分かれている．(c) Strummer は 6 段階のチュートリアルを提供する．(d) Listen ステージ：手本を聴いて曲の概観を掴む．(e, f) Chord ステージ：コードは *CP* 降順に並んでおり，1 つずつ練習する．(g) コードがうまく弾けなかった場合，1 弦ずつ加算的に練習する（この例では *Bm* の上 3 弦を部分的に練習している）．(h) ユーザが何弦まで弾けたのかをシステムが記録し，後続ステージでは弾けなかった弦の音解析を行わない．この例でユーザは上 3 弦までしか弾けなかった為，元の形から省略された *Bm* になっている．(i) Transition ステージ：コードの遷移を練習する．(j) Segment ステージ：セグメントをシステムの示すタイミングに合わせて弾く．(k) Rehearsal ステージ：セクション全体を練習する．(l) Concert ステージ：ゲームのような画面で曲全体を練習する．

Segment ステージ

セグメントはセクションに含まれる小節を 4 小節ごとに区切ったものとして定義した．Segment ステージでは，図 2(j) のように，システムが示すタイミングに合わせてセグメント中のコードを演奏する．

Rehearsal ステージ

図 2(k) が示す Rehearsal ステージでは，セクションに含まれるフレーズをつなげて練習することができる．

Concert ステージ

最後にユーザは Concert ステージに進む．本ステージは図 2(l) のように，ゲームのようなインターフェースになっている．本ステージは，同じ曲に含まれる他のセクションのフレーズを全てつなげて練習する．例えば A メロ，B メロ，サビという構成の曲で A メロの練習は既に終えており，今サビの練習をしているならば，A メロとサビのフレーズを全てつなげて提示する．つまり，全てのレッスンを練習すれば Concert ステージで楽曲全体の演奏練習ができる．ユーザが 60%以上のコードを正しく弾けたと判断されるとステージクリアとなり，1 つのレッスンが終了する．

5.2 システムの実装

図 3 にコード解析の流れの概要を示す．本システムは Ruby on Rails と JavaScript を用いて実装し，ウェブブラウザで利用できるようにしている．コード解析モジュールは Python の numpy，pyaudio ライブラリを使用して実装した．本モジュールは Fast Fourier Transform (FFT) によって和音の要素を解析している．サンプリングレートは 44.1kHz，窓関数はハニング窓を使用し，窓長は 8192 とした．各弦から鳴るはずの正解音高の周波数成分のピークが立っているのかを見ることで，各弦が正しく鳴っているのかを 1 つずつ判定し，HTTP 通信で結果をサーバに送信している．

6. 評価実験

6.1 実験設定

本システムの有効性を示すためにユーザテストを行った．このユーザテストの目的は，一定の練習期間ののち，ユーザが今まで練習したことがない初見の曲がどの程度弾けるようになっているのかを調べることである．本ユーザテストでは対象の曲をポップスのジャンルに絞り，*SP* 順にソートした楽曲を順次練習してもらった．最終的に，同ジャンルで初見の課題曲を短い練習時間でどの程度弾けるようになるのかを調べた．

ユーザテストは 5 日間行い，5 名が参加した．5 名ともギターに触れた経験のない初心者であった．1 日 1 時間本システムを用いてギターのコード演奏を練習してもらい，その後簡単なインタビューを行った．最終日の 5 日目は，最初の 30 分で通常通り練習し，その後の 15 分間で課題曲を予備練習した後に本番として演奏してもらった．筆者はそれを録音し，最後にインタビューを行った．課題曲はデータセットの中から Otis Redding の “*(SittingÓn) The Dock Of*

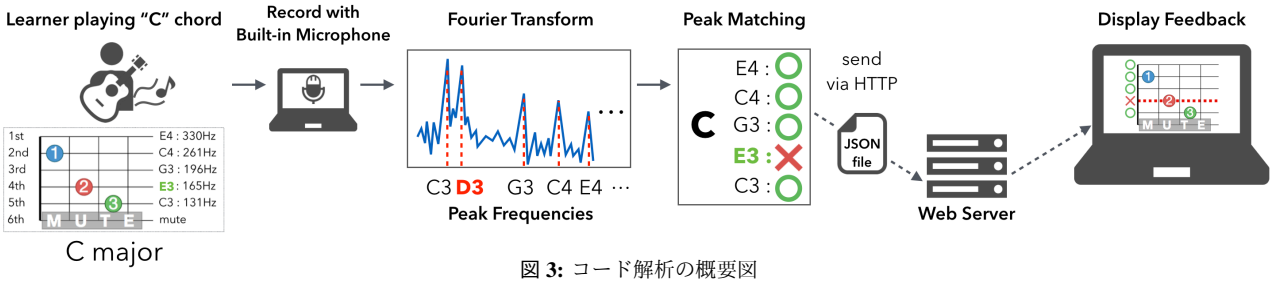

図 3: コード解析の概要図

表 3: 課題曲の演奏結果

	既出						初見		
コード	Em	D	C	E	G	F	A	B	合計
出現頻度	2	4	5	1	14	1	3	2	32
押弦難易度	1	1.5	2	2	2	3	2	3.5	2.13(平均)
P1	2	4	3	1	14	0	3	1	28
P2	1	4	3	0	12	1	0	0	21
P3	1	0	3	0	12	0	0	0	16
P4	2	4	5	1	14	0	3	0	29
P5	2	4	5	1	13	1	3	2	31
成功率 (%)	80.0	80.0	76.0	60.0	92.9	40.0	60.0	30.0	78.1

The Bay” を選んだ．この曲は *C*，*D*，*G* の頻出コードと *F*，*B* のバレーコード，および *A*，*B* の初見のコードが含まれており，課題曲として妥当であると考えられる．

6.2 実験結果

参加者 5 人全員が 5 日間のユーザテストを完遂した．最終日に行った課題曲の各参加者ごとの結果を表 3 に示す．筆者のうちの 1 人が実験参加者の演奏したコードが正しく弾けているのかを判定した．表 3 は各実験参加者がそれぞれの曲中に出てきたコードのうち，どれだけ正しく演奏することができたのかを示している．F や B のバレーコードを正しく演奏した実験参加者は少なかったが，全参加者が曲中の少なくとも半分以上，平均して 78.1%のコードを正しく演奏した．

また，インタビューによって Strummer システムの利点が明らかになった．実験参加者は段階的なチュートリアルに利点を感じたと報告している．P3 はチュートリアルの利点について以下のように述べている：

> なんかこう、段階踏んで *1* つずつどんどんクリアしていくような感じで。ほんともうモチベーションが結構出てきてよかったんじゃないかなあと思いました。 [P3]

Strummer の段階的練習の中で，コード認識モジュールが実験参加者が正しくコードを弾けているのか判断していたが，以下のような意見からこの機能は参加者にとって有用であったと言える．

> 正しい音が出てるのかどうかをチェックしてくれるというのはすごくいいなぁと思います。そうですね、基本バツが出たところは押さえが弱いんだろうなってことが分かりましたし。 [P2]

7. おわりに

楽器の演奏は多くの人が望むスキルである．近年多くのオンラインサービスなどが登場しており，初学者は何から練習すればよいのか迷いがちである．我々は，ユーザが選んだ楽曲セットの中で，重要かつ練習しやすい曲順を生成するアルゴリズムとギターコードの難易度を推定するモデルを作成した．我々が提案した Strummer は上記のアルゴリズムとモデルを活用し，ユーザの演奏を解析して即座にフィードバックすることで，データドリブンでインタラクティブなギターコード練習を可能にした．さらに，楽曲データから段階的なチュートリアルが自動生成され，ユーザは徐々にスキルアップを目指せる．評価実験では参加者に Strummer システムを用いて練習してもらい，参加した 5 人全員が実験を完遂し，Strummer システムに好意的意見を述べ，本システムの利点が明らかになった．

参考文献

[1] Burgoyne, J. A. et al.: An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis, *Proc. ISMIR*, pp. 633–638 (2011).
[2] Dannenberg, R. et al.: A computer-based multi-media tutor for beginning piano students, *Journal of New Music Research*, Vol. 19, No. 2–3, pp. 155–173 (1990).
[3] Glise, A.: *Classical Guitar Pedagogy: A Handbook for Teachers*, Mel Bay Publications, Inc. (1997).
[4] Motokawa, Y. and Saito, H.: Support System for Guitar Playing Using Augmented Reality Display, *Proc. IEEE ACM ISMAR*, pp. 243–244 (online), DOI: 10.1109/ISMAR.2006.297825 (2006).
[5] Ng, K. and Nesi, P.: i-Maestro : Technology-Enhanced Learning and Teaching for Music, *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME ’08, Genoa, Italy, pp. 225–228 (2008).
[6] Percival, G. et al.: Effective Use of Multimedia for Computer-assisted Musical Instrument Tutoring, *Proc. the international workshop on EMME*, pp. 67–76 (online), DOI: 10.1145/1290144.1290156 (2007).
[7] Takegawa, Y., Terada, T. and Tsukamoto, M.: A Piano Learning Support System considering rhythm, *Non-Cochlear Sound: Proceedings of the 38th International Computer Music Conference, ICMC 2012, Ljubljana, Slovenia, September 9-14, 2012*, (online), available from ⟨http://hdl.handle.net/2027/spo.bbp2372.2012.061⟩ (2012).