ゴルフスイング練習支援のための マルチモーダル動作アラインメント

山本 和彦[1,a)]　田嶋 良平[1,b)]　澤 雅樹[1,c)]　佐藤 貴博[1,d)]　伊藤 栄美[1,e)]

概要：ゴルフのスイング練習では, 自分のスイングの様子を撮影し, 上級者のスイング動画と見比べることで自分のフォームを修正していくことが有効な練習方法とされている. しかしながら, 各々のスイングは振りの速度や緩急のタイミングが大きく異なっており, それぞれどのタイミングのフォーム同士を比較すれば参考になり得るのか初心者には判断が困難であるという問題がある. また, ひとつひとつのスイングをこまめに撮影して編集するのは大きな労力がかかる. そこで本稿では, 異なる複数のスイング動画を動作アラインメントすることによって, それぞれどことどこのタイミングが対応しているか比較できるようユーザに提示する手法を提案する. 本システムでは, あらかじめ教師データに対してスイングとして重要なポイントをアノテーションしておくことで, 要所となるフォームを異なる動画間で簡単に比較することができる. また, 画像と音の両面からマルチモーダル解析することによって, ひとつの動画に複数スイングが収録されていたり, スイング以外の余分な動作が含まれる撮りっぱなしの動画からも精度良く個々のスイングだけを抜き出して解析することができる. これによって, ユーザは動画をスイングごとに分けて撮影したり, 後から編集する手間から解放され, 効率的に練習に集中することができるようになる.

1. はじめに

ゴルフのプレイヤーがスイングを上達していくためには, 自分のスイングの様子を撮影し, 上級者や自分の過去のスイング動画と見比べることで自分のフォームを修正していくことは有効な練習方法である, とされている. しかしながら, 各々のスイングは振りの速度や緩急のタイミングが大きく異なっており, 自分の動画と比較動画のそれぞれどのタイミングのフォーム同士を比較すれば参考になり得るのか, 初心者には判断が困難であるという問題がある. また, ひとつひとつのスイングごとに分けて撮影したり, あるいは練習中に長い時間撮りっぱなしにした動画を後から編集してそれぞれのスイング部分だけを抜き出して見比べることは大きな労力がかかり, スイング練習そのものに集中できなくなってしまうという問題もある.

そこで本稿では, 異なる複数のスイング動画を動作アラインメントすることによって, 長時間の様々な行動が含まれる動画から個々のスイング部分だけを抜き出し, かつ動画間でそれぞれどことどこのタイミング同士が対応しているかをユーザに提示することができる手法を提案する. 動作アラインメントとは, 与えられたユーザ動画 (ターゲット動画) 内のどのフレームが比較動画 (リファレンス動画) 内の一連のスイング動作中のどこの過程に対応するかを推定してフレーム間の対応関係を計算することである (図 1). これは, Subsequence Dynamic Time Warping 法 (SDTW) [1] を安定してゴルフスイングの動作アラインメントができるよう拡張し, さらに動画像と音の両面からのマルチモーダル解析をおこなうことによって可能となった.

本システムでは, あらかじめ教師データとなるスイングのリファレンス動画に対してインパクト点 (クラブがボールに当たる時点) などのスイングの特徴的なポイントをアノテーションしておくことによって, 異なるスイング動画間で要所となるフォーム同士を簡単に比較することが可能となる. また, 動画像と音の両面からマルチモーダル解析することによって, ひとつの動画に複数スイングが収録されている場合やスイング以外の余計な行動が含まれる動画からも精度良くそれぞれのスイング部分だけを抜き出して解析することができる (図 2). これによって, ユーザは動画を検索したり編集したりして適切な比較位置を探す手間や個々のスイングごとに分けて撮影する労力から解放され, 効率的に練習に集中することができるようになる.

本稿ではまず, 提案手法によって精度良くロバストにユーザが撮影した動画の中から個々のスイングの検出とアライ

[1] ヤマハ株式会社
[a)] yamo_o@acm.org
[b)] tajima.ryohei@music.yamaha.com
[c)] masaki.sawa@music.yamaha.com
[d)] takahiro.sato@music.yamaha.com
[e)] emi.ito@music.yamaha.com

© 2023 Information Processing Society of Japan

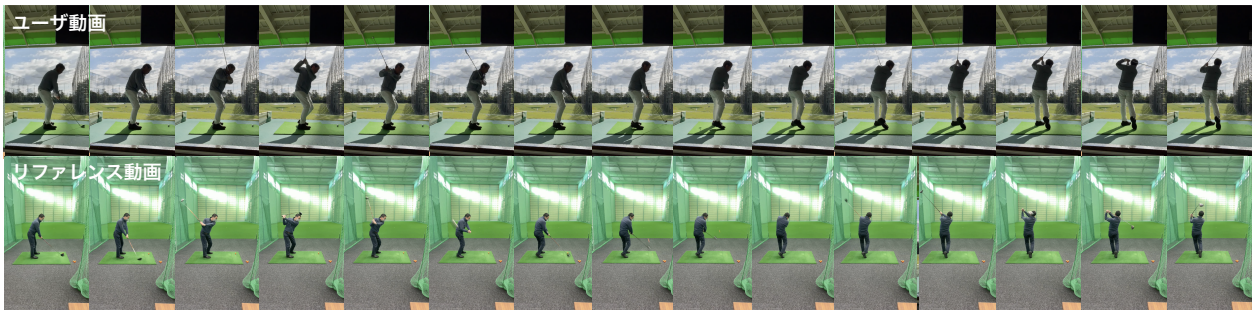


図 1 ターゲット動画となるユーザが入力したスイング動画 (上段) に対して, リファレンス動画となる比較対象のスイング動画 (下段) との間で動作アラインメントをおこない, フレーム間対応を出力する.

ンメントを同時におこなうことができるようになることを定量的に評価する. その後, iOS 上のモバイルアプリケーションとして実装, ユーザ評価することによって, スイング練習動画の撮影と視聴が非常に効率的におこなうことができるようになることを示す.

2. 関連研究

動画内の人間の特定の動作を認識する研究は数多くある. 近年では Deep Neural Network (DNN) ベースの手法が主流であり, 2D Convolutional Neural Network (CNN) と Long-short Term Memory (LSTM) 等の Recurrent Neural Network (RNN) を組み合わせるもの [2][3][4][5], 3D の CNN に複数フレームを入力して認識するもの [6][7], Visual Transformer を利用するもの [8], 動画から一旦骨格情報を抽出してそれを入力として Graph Convolutional Network (GCN) で認識をおこなうもの [9][10][11][12], などがある. さらに本稿と同様に, 視覚情報だけでなく, 音の情報も利用してより柔軟な認識をおこなう手法も提案されている [13][14][15]. これらの手法は実際にスポーツの動作認識にも採用されている [16]. しかしながらこれら DNN ベースの動作認識手法は, 1. 大量の学習データを必要とする, 2. 動作の種別を認識するのみであり, 一連の動作のどの部分がどのフレームに対応しているのかをアラインメントするためには認識後に別の手法を用意しなくてはいけない, という問題がある.

我々の目的のためには, 動画内での特定の動作を検出するだけではなく, その動作のアラインメントをおこなわなくてはならない. Taein らは, 動画内の人間の動作を教師データの動作にアラインメントするための自己教師あり学習を提案した [17]. 彼らの手法では, 動画から推定された人間の骨格情報をデータ拡張して対比学習させることで, フレーム間対応を考慮した潜在変数を出力できるようにする. しかしながら, 我々の問題においては, 目的の動作以外の動作も含み, かつ目的の動作も複数収録されている動画の場合にこの手法はそのまま適用できないという問題がある. また, 教師データとのフレーム間対応が確率分布で得られたとしても, もっとも高い確率のフレーム同士を順に繋げるだけでは時間が前後してしまう可能性がある. よってそれを一連の動作として繋げるためにマッチングの後処理が必要となるがこれは自明ではない. 我々の提案手法は, これらの問題を一連の動作の抜き出しとアラインメントを同時にすることによって解決している.

ゴルフスイングの動作解析に特化した練習支援アプリケーションとして, GolfVision [18] がある. GolfVision では正面, または背後から撮影したスイング動画からアドレス位置やインパクト位置など各要点となるフレーム位置を検出することができる. 技術的な詳細は残念ながら公開されていないが, 筆者らが検証したところ, 複数スイングや余計な動作を含む撮りっぱなしの動画から対象者のスイングだけを抜き出すことは困難であった. また, 素振りと実際の打球を分けて扱うことはできない. 本稿ではこうした問題を解決する.

3. スイングのマルチモーダル解析

提案手法は事前学習と実行時の 2 段階から成り立つ. 事前学習時にはまず, 集められた N 本の背後から撮影された単発スイングのリファレンス動画を学習データとしてシステムに入力する. このリファレンス動画にはそれぞれスイングフォームの要所となるフレーム位置のアノテーションが含まれている. ここで本稿では, 要所となる位置として, アドレス (スイング開始), トップ (クラブを振りあげた時点), インパクト (クラブがボールに当たる瞬間), フィニッシュ (ボールを打ち終わってクラブを振りあげた時点) の 4 点と定義する. システムは入力された学習動画において, それぞれの骨格情報を抽出 (§3.1) してすべての動画間の距離 (§3.2) を計算することで, 階層バイナリクラスタツリー (*Dendrogram*) を形成する (§3.4). これはあるターゲット動画が与えられたときに比較すべきリファレンス動画を高速に検索するための検索木となる.

実行時, ユーザが解析対象となるスイング動画 (ターゲット動画) を入力すると, システムはまず動作解析のための骨格検出をおこなう (§3.1). 次に得られた骨格情報を検索

© 2023 Information Processing Society of Japan

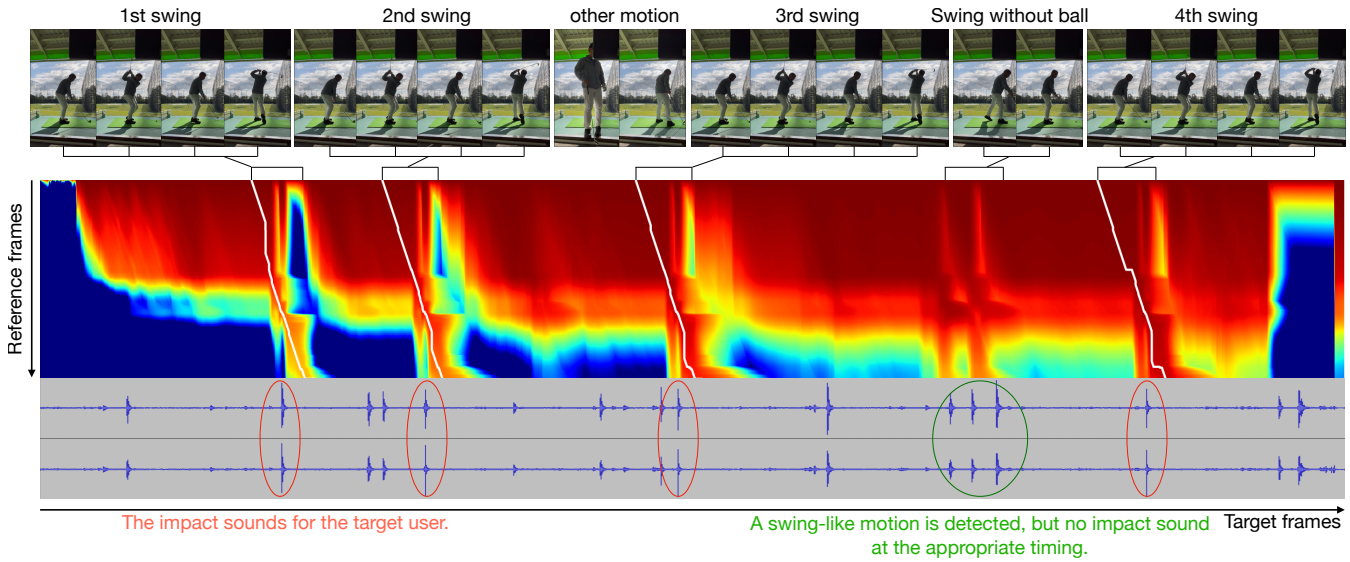


図 2 画像処理による骨格情報抽出と音 (下段) の解析を併用したゴルフスイングのマルチモーダル解析．中段に累積された SDTW コストの逆数を可視化している (縦軸:下に向かってリファレンス動画の時間, 横軸: 右に向かってターゲット動画の時間)．中段内の白線はスイングのリファレンス-ターゲット動画間のアラインメントされたフレーム対応を示す．このターゲット動画には 4 本の実スイングのほか, 素振りや余計な動作も含まれているが, 正確に実スイング部分だけを検出することができている (上段)．

木に入力し, 比較するリファレンス動画を検索する (§3.4)．次にターゲット動画の骨格情報とリファレンス動画の骨格情報を入力として Subsequence-Dynamic Time Warping 法 (SDTW) [1] の拡張でターゲット動画にいくつスイングらしい動作 (仮スイング) が含まれているか を検出する (§3.2)．検出されたそれぞれの仮スイングに対して, さらに Dynamic Time Warping 法 (DTW) [19][20][21] を適用し, 細かいアラインメントをおこなう (§3.3)．ここでリファレンス動画においてアノテーションされたインパクト位置に対してアラインメントされたターゲット動画のインパクトフレーム近傍の音を解析し, オンセット検出することで詳細なインパクト位置を計算する (§3.3.1)．このとき, 適切なインパクト音が検出されなかった場合, 仮スイングは棄却され, 残った仮スイングが実スイングとなる．最後に計算されたターゲット動画のインパクト位置から実スイングのフレーム全体のアラインメントをスムージングすることで最終的なそれぞれのスイングアラインメントを得る (§3.3.2)．ここで, 複数のターゲット動画間のスイングアラインメントを取りたい場合は, 一旦それぞれのターゲット動画においてリファレンス動画とのアラインメントを計算したのち, それぞれのリファレンス動画間のアラインメントからターゲット動画間のアラインメントを算出することで対応できる．

3.1 特徴量

与えられた動画から動作特徴量と音響特徴量の 2 種類を抽出する．動作特徴量としては各フレーム t における全身の各関節 33 点 $P_t \in \mathbb{R}^{33}$ を全フレーム T にわたって求める (図 3)．この 2 次元画像からの姿勢推定には Google MediaPipe [22] を利用した．MediaPipe では関節座標が腰の位置を原点としたワールド座標系と画像座標系の 2 通りが出力されるが, 本稿ではワールド座標系を採用する．これはカメラが手持ちで撮影された場合カメラの揺れによる影響を抑えるためである．また, スイング動作では主に手の位置が大きく移動するため, 手首から先の関節には 1 以上の重み (実験では 10.0 を使用) をあらかじめ掛けて増幅した．さらに動作速度を考慮するために前後フレーム間での差分 $V_t = P_t - P_{t-1} \in \mathbb{R}^{33}$ を加えて計 66 次元を動作特徴量とした．

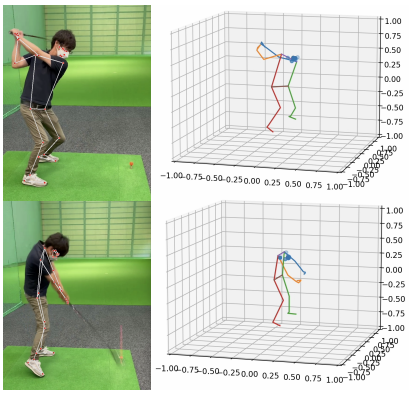


図 3 スイング動画からの骨格抽出 (ワールド座標系)．

ゴルフスイングデータのほとんどは右利きのものであったが, 中にはレフティ (左利き) のスイングも含まれる．我々の実験では, 動画内で一番高く手を上げている瞬間に,

© 2023 Information Processing Society of Japan

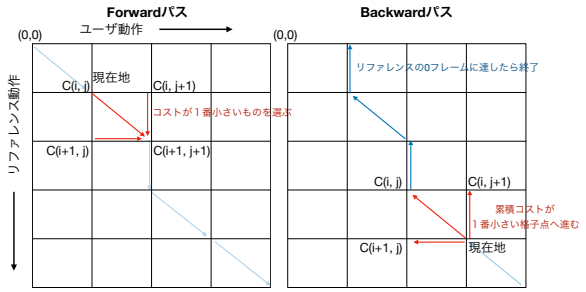


図 4 SDTW では forward パス (左) 時には時間順方向にコストを累積させていき, backward パスでは終端から逆方向に最も累積された SDTW コスト (尤度) が小さいパスを辿ることでアラインメントをおこなう.

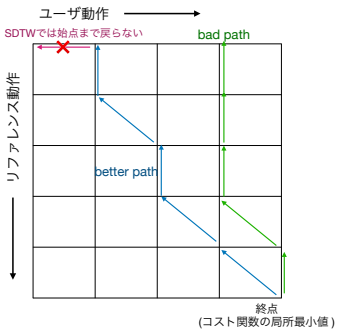


図 5 動作アラインメントの問題では片方の動画のフレームだけが一方的に進んでしまう短絡的なパス (緑) より, どちらの動画フレームもバランス良く進んで距離が長いパス (青) を辿るほうが望ましい .

画像座標系において両足の重心が画面の左右どちらに寄っているかによって, その動画がレフティかどうかをあらかじめ判別し, もしレフティ動画ならば世界座標系の x 軸を反転させることで対応した.

一方, 音響特徴量としては, サンプリング周波数 44.1kHz/16bit で収録された音を, 窓幅 128 サンプル, ホップサイズ 64 サンプルで FFT し, 各フレームでのパワースペクトル $\in \mathbb{R}^{64}$ を計算した.

3.2 長時間の動画からの単発スイングの検出と分割

複数スイングや, 除外したい素振りや他の動作を含むターゲット動画から, 単発の打球有りスイングそれぞれの部分だけに分割するためにはまず, 得られた骨格時系列データを任意のリファレンス動画に対して動作アラインメントする. この動作アラインメントを安定しておこなうために我々は SDTW の拡張を導入する. SDTW は DTW の拡張で, DTW がターゲット系列とリファレンス系列が 1 対 1 に対応し, かつ入力された両者の始点と終点は既に揃っていることを仮定しているのに対し, ターゲット系列に複数のリファレンス系列が含まれていて, かつ途中の始点から途中の終点までの部分的なアラインメントにも対応できるようにした手法である [1].

長さ S フレームのリファレンス動画と長さ L フレームのターゲット動画が与えられたとする. SDTW では動的計画法と同様に時間方向に尤度を積算していく forward ステップと最尤パスを逆向きに辿る backward ステップから成り立つ (図 4). 我々の forward ステップにおいては, リファレンスのフレーム $i+1, i \in \mathbb{N}_{[0,S)}$, ターゲットのフレーム $j+1, j \in \mathbb{N}_{[0,L)}$ における尤度 (SDTW コスト) は,

$$\begin{aligned}\mathbb{C}_{i+1,\,j+1} &= \mathbb{Y}_{i+1,\,j+1} \\ &+ \min([\mathbb{C}_{i,\,j},\ \mathbb{C}_{i+1,\,j},\ \mathbb{C}_{i,\,j+1}]) \\ &+ \alpha \cdot |\mathbb{D}^r_{i+1,\,j+1}/T^r - \mathbb{D}^t_{i+1,\,j+1}/T^t| \quad (1)\end{aligned}$$

のように更新される. ここで, $\mathbb{Y}_{i,\,j} = |P^r_i - P^t_j|_2 + \beta|V^r_i - V^t_j|_2$, $\beta = 1.0$, はリファレンス動画 i フレーム目とターゲット動画 j フレーム目の動作特徴量間の距離である. α は正則化重みである (実験では $\alpha = 10.0$ を使用). この Forward ステップですべての格子点における尤度を求めたのち, backward ステップで終点とする位置から時間を逆向きに尤度の最も低い格子点を辿っていくことでターゲット-レファレンス間のアライメントをおこなうことができる.

しかしながらここで問題となるのは, ターゲット系列とリファレンス系列の対応を計算したとき, 極端に片方だけ進んでしまう短絡的なパスもまた尤度が低くなってしまうことが多いということである. 動作アラインメントの観点からはターゲットもリファレンスもできる限り均等にバランス良く長く進むパスを選択したほうが滑らかなスイングが検出できると期待される (図 5). このできる限り長くバランス良く進むパスのほうが望ましいという条件を拘束として導入したものが式 1 の右辺第 3 項目であり,

$$\mathbb{D}^r_{i+1,\,j+1},\ \mathbb{D}^t_{i+1,\,j+1} = \begin{cases} \mathbb{D}^r_{i,\,j+1} + 1,\ \mathbb{D}^t_{i,\,j+1} & (a) \\ \mathbb{D}^r_{i+1,\,j},\ \mathbb{D}^t_{i+1,\,j} + 1 & (b) \\ \mathbb{D}^r_{i,\,j} + 1,\ \mathbb{D}^t_{i,\,j} + 1 & (c) \end{cases} \quad (2)$$

と定義する. ここで (a), (b), (c) はそれぞれ $(i, j+1)$, $(i+1, j)$, (i, j) から $(i+1, j+1)$ へ進んだ場合を示す. この forward パスで累積されて各格子点 $\in \mathbb{R}^{S \times L}$ に格納される累積された尤度を SDTW コストと呼ぶ.

ここで計算された SDTW コストをもとにターゲット動画からそれぞれの単発の仮スイングの部分を抜き出す. まず, リファレンス動画でアノテーションされたフィニッシュ位置から 0.5 秒後をスイング終端と定義する. この終端位置に対応するターゲット動画の SDTW コストの逆数の時間遷移を図 6 に示す. この図におけるピーク位置 (赤と緑の円) はターゲット動画内でユーザがスイングのような動作をして始点から終端まで振り切った位置に相当する. つまり, ピークの数はそのままターゲット動画に含まれるスイングらしい動作の数となる. よってそれぞれのピーク検出を終端位置として, そこからの最尤パスを backward ステップで求めることによって, それぞれ個別のスイングのアライメント (フレーム間対応) を得ることができる.

© 2023 Information Processing Society of Japan

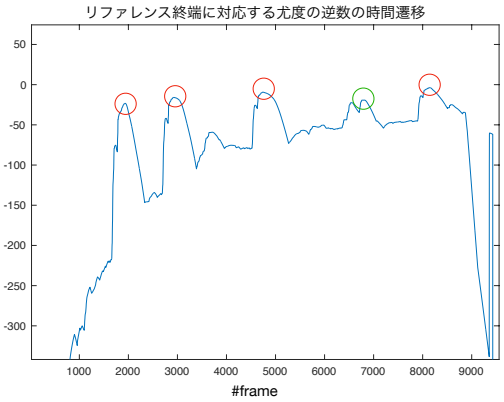


図 6　リファレンスの終端に対応する SDTW コストの逆数 (縦軸) のターゲット時間遷移 (横軸). このピークがスイングらしい動作の終端に対応する. 赤丸のピーク: 実際のスイング終端 フレーム. 緑色のピーク: 素振りの終端フレーム.

Algorithm 1 Finding the terminated frames.

```
function FindPeaks(X)
    peaks ← ∅
    i ← 1
    while i < len(X) − 1 do
        if X[i − 1] < X[i] then
            j ← i + 1
            while j < len(X) do
                if X[j] ≠ X[i] then
                    break
                end if
                j ← j + 1
            end while
            p1 ← j − 1
            while X[j] < X[i] do
                j ← j + 1
                if j ≥ len(X) then
                    break
                end if
            end while
            p2 ← j − 1
            peaks ← peaks ∪ {0.5 · (p1 + p2)}
            i ← j
        end if
        i ← i + 1
    end while
    return peaks
end function
```

Algorithm.1 にピーク検出のアルゴリズムを示す. ピーク位置のリストを作成したのち, 不適切なピークをフィルタリングする. これにはまず, ピークのリストを SDTW コスト関数の逆数が降順になるようにソートし, 閾値以下のものを削除する. 次にリファレンススイングの長さと比較して近接し過ぎているピーク (実験では 0.8 倍以下) をリストの末尾から削除していく. ここで残ったピークの数はターゲット動画に含まれるスイングらしい動作の数となる. よって最終的に残ったピーク位置がそれぞれ仮スイングの末端として採用される.

3.3　単発スイング同士のアラインメント

撮りっぱなしの長いターゲット動画から仮スイングの部分を切り出したのち, それぞれ単発仮スイングごとのリファレンス動画に対するアラインメントをおこなう. このアラインメントにも §3.2 と同様に SDTW を利用する. DTW を使用しない理由は, 終端から原点までのアラインメントが必ずしも必要無いからである. ここでアラインメントされたそれぞれの単発仮スイングは, スイング以外のものを含んでいたり, 球を設置せずに打った動作だけの素振りなどの余計なものも含まれている. また, スイング動作は非常に高速であるため, インパクト位置ではとくに計算されるフレーム間のずれが大きくなる傾向がある. そこで, 本稿では, インパクト音の検出を併用することで, より精度良く, 目的の本スイングだけを検出する手法を提案する.

3.3.1　インパクト音検出

スイングのインパクト位置では, 金属質のかなり大きな衝撃音が発生する. このインパクト音を利用して, 動作情報のみから得られたそれぞれの仮単発スイングのおおよそのインパクト位置をより正確な位置へ修正することを考える. まず, 推定されたおおよそのインパクト位置から前後 0.15 秒区間の音データ (パワースペクトル) においてオンセット検出をおこなう. 屋外での撮影などでは風の音なども同時に収音されてしまうことがあるため, オンセット検出の前にハイパスフィルタをスペクトル上でかけるとノイズの影響を軽減させることができる. ここでオンセットが検出されない, またはオンセット付近のパワーが閾値より低い場合にはこの仮単発スイングはスイングではない, または素振りである, と判断して棄却する. 閾値以上のオンセットが検出された場合はインパクト位置をその位置へ矯正する.

3.3.2　スムージング

インパクト音検出によってより正確なインパクト位置を算出した後, 次にアラインメントをこのインパクト位置に合わせ込んでスイング全体を辻褄が合うようにスムージングする. まず, リファレンス動画に付随するアノテーションから各点に対応するフレームとして, ターゲット動画におけるスイングのアドレス, トップ, インパクト, フィニッシュ各点を求める. このとき, インパクト位置は §3.3.1 で計算した位置を採用し, もし更新されたインパクト位置が元の位置より前方ならばアドレスとトップ位置をその差分だけ前に, 逆に後方ならばフィニッシュ位置は後ろにずらす. この各点に対して切り出した単発スイング部分を, 動画開始フレームからアドレス, アドレスからトップ, トップからインパクト, インパクトからフィニッシュ, フィニッシュから動画終端フレームまでを図 7 のように小さな局所ブロックへ分割し, それぞれのブロックごとに DTW でアラインメントをおこなう. 各ブロックはそれぞれ始点 (図 7 における左上) のコストが 0 として初期化され, かつ各制

© 2023 Information Processing Society of Japan

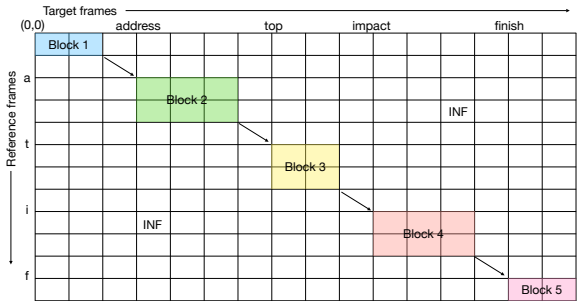


図 7 音の解析によってより正確なインパクト位置が算出されると, システムはそのインパクト位置に合わせてスイングのアラインメント全体をスムージングする. 各要所では必ずリファレンス-ターゲット両方が 1 フレームずつ進んで到達するとし, 区切られたブロック内でそれぞれ DTW によるアラインメントをおこなう.

Algorithm 2 Constructing the dendrogram.

```
1: function BuildDendrogram(D)
2:     nodes ← ∅
3:     for all e ∈ [0, N) do
4:         nodes ← nodes ∪ {e}
5:     end for
6:     root ← null
7:     updated ← true
8:     while updated do
9:         updated ← false
10:        min ← INFINITY
11:        pair ← {0, 0}
12:        for j in N do
13:            if not node[j] then
14:                continue
15:            end if
16:            for all i ∈ [0, j) do
17:                if not nodes[i] then
18:                    continue
19:                end if
20:                d ← D[nodes[j]][nodes[i]]
21:                if d < min then
22:                    min ← d
23:                    pair[0] ← j
24:                    pair[1] ← i
25:                    updated ← true
26:                end if
27:            end for
28:        end for
29:        if updated then
30:            newNode ← node[pair[0]] ∪ node[pair[1]]
31:            node[pair[0]] ← newNode
32:            nodes[pair[1]] ← null
33:            root ← node[pair[0]]
34:        end if
35:    end while
36:    return root
37: end function
```

御点へは 1 つ前のフレームから必ずターゲット, リファレンス両方 1 フレームずつ進んで到達するよう拘束をかける. 前者は forward パス時, 後者は backward 時にブロックごとに計算を分けることでおこなうことができる.

3.4 適切なリファレンス動画の検索

ユーザがターゲット動画を入力すると, システムはまず比較するのに適切なリファレンス動画を検索する. ここで, 適切なリファレンス動画とは SDTW でアラインメントしたときの終端での SDTW コストができる限り小さくなるような学習動画である. しかしながら, 学習動画データベースを全検索するのは時間がかかるため実用的ではない. そこで, あらかじめ事前学習時にリファレンス動画間の関係を解析しておくことで, 高速に適切なリファレンス動画を検索できるようにする. 具体的には, 総リファレンスデータ間の距離を使って検索木を作成しておき, 実行時には検索木の枝をたどって検索することで高速に適切なリファレンス動画を求める.

N 本の学習動画が入力されたとき, システムはまず, 全通りの動画間の DTW 距離を計算して距離行列 $\mathbf{D} \in \mathbb{R}^{N \times N}$ に格納する. この距離行列 $\mathbf{D}$ を入力として階層バイナリクラスタツリー ($Dendrogram$) を構築する. 木構築の疑似コードを Algorithm.2 に示す. まずすべての動画インデックスをノードとしてリストに格納する. つぎにもっとも近いノード同士を結合していき, 最終的にひとつのノードになるまで繰り返す. 複数の動画インデックスを含むノードでは, 内部に含まれる他の動画間 DTW 距離の和が最小のものをそのノード n の代表動画インデックス $e_n = \arg\min_i \sum_{i \neq j} \mathbf{D}_{i,j,},\ i, j \in \mathbb{I}_n$, とする. ここで, $\mathbb{I}_n$ はそのノード n 以下に含まれるリファレンス動画の集合である. 異なる 2 つのノード間 $n,\ m$ の DTW 距離にはこの代表動画同士の DTW 距離 $\mathbf{D}_{e_n, e_m}$ を採用する. Algorithm.2 の 30 行目では, $node[pair[0]]$ を左の子ノード, $node[pair[1]]$ を右の子ノードとしてもつ新しいノード $newNode$ が生成され, 31 行目で $node[pair[0]]$ を置き換えることを示している. これを可視化したものが図 8: 左である. この図では 145 本すべての学習動画に対してどの動画とどの動画がどれくらい近く, かつ似ている動画がどのようにまとまってクラスタを形成していくか, を表している.

検索時には, 入力されたターゲット動画に対して, 検索木のルート (図 8: 左の一番上のクラスタ) から順に左右の子ノードの代表動画と DTW 距離を比較し, 近いほうの枝を辿る, ということを, そのノード以下の子の数が閾値 $K = 8$ になるまで再帰的に繰り返していく. ここで最終的に辿りついたノード以下の子ノードのリファレンス動画のうち, もっともターゲット動画と DTW 距離が近い動画が検索結果となる. ここで検索された結果は最適 (もっともリファレンス動画のデータベースの中でターゲット動画と DTW 距離が小さい) であるという保証はないが, 近似的に上位の良いリファレンス動画となることが期待される.

この検証をおこなうため, リファレンス動画の単発スイング 145 本, 総収録時間 9.07 分すべてをターゲット動画のクエリとして検索をおこなった. この場合は当然ながら入

© 2023 Information Processing Society of Japan

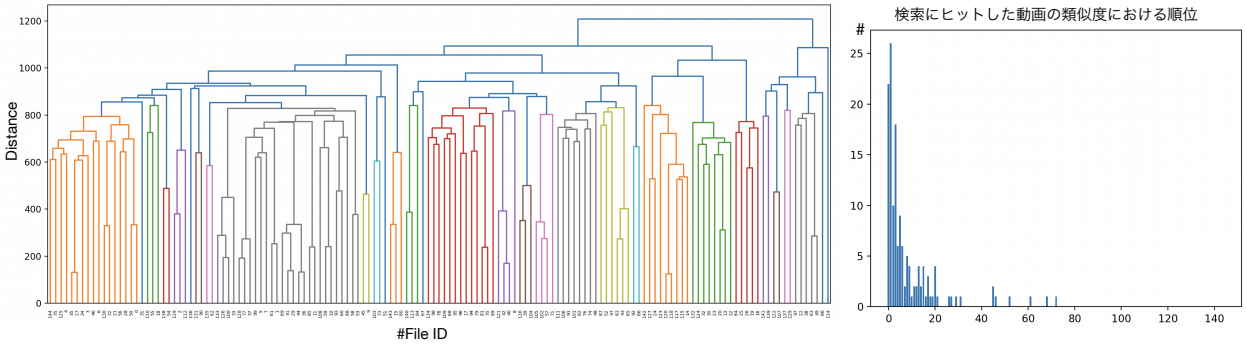


図 8 左: 階層バイナリクラスタツリーの可視化. それぞれの学習データ (リファレンス) 同士がどのくらいの距離でどのようにまとめられているかを示す. 一番上のまとまりがルートとなる. 右: 学習データそれぞれをクエリとして検索にかけたとき, ヒットしたデータが実際に全データの中で何番目にクエリに近いものだったかをヒストグラムとして示す.

力を同じ動画が検索結果として返されるのが正解となる. 全検索した場合, 一致率 145/145 となり当然すべて正しい結果となったが時間は 2.77 分かかった. 一方, 検索木上で検索した場合は 8.7 秒で完了した. この差は学習データが増加するほど顕著になる. またこのとき, 検索結果がリファレンスデータベースの中で何番目にターゲット動画との DTW 距離が小さいものであったかをヒストグラムで示す (図 8: 右). つまり, この図において 0 番目にカウントされているものは全く同じリファレンス動画がヒットしたことを示す. 結果は, 全体の 63% のクエリが上位 5 位までに, 全体の 77% のクエリがが上位 10 位までに, 全体の 90% のクエリが上位 23 位までにヒットしており, 十分近似的に良い検索結果が得られた. これによって現実的な速度で適切な検索がおこなうことができていると言える. この検索木の近似精度はデータを増加させることでさらに改善することができる. ここで, 明らかに最適ではない手法を採用している点としてノードの代表動画の選別方法が挙げられるが, これについては今後さらなる検討が必要である.

4. 実験

4.1 データ収集

単発のスイングを背後から撮影した動画を学習データとして収集した。機材は Apple iPhone Pro 13 を使い, 120fps のスロー撮影をおこなった. また同時に音をサンプリング周波数 44.1kHz/16bit で収録した. スイングは中級者からプロまで約 100 名がおこない, 総計 145 本, 総収録時間 9.07 分となった. この 145 本の動画に対して手動でアドレス, トップ, インパクト, フィニッシュの各フレーム位置をアノテーションした. ここで, インパクト位置以外は本来の定義が曖昧であるが, 我々の実験では, アドレスは静止状態からクラブ先端が動き始めるタイミング, トップは手を振り上げてボールを打つために折り返すタイミング, フィニッシュは打った後もっとも手が高く上がるタイミング, とした.

一方, 評価用の動画としては単発だけではなく, ひとつの撮影に素振りや余計な動作も含み, かつ複数のスイングが含まれるものも用意した. 撮影は屋内/屋外の練習場およびフィールドで 120fps のスロー撮影と 30fps の通常撮影でおこない, カメラスタンドを使ったものと手持ちの撮影 (手ブレも入る) の両方が混在している. 評価用動画も学習データと同様に手動でアドレス, トップ, インパクト, フィニッシュのフレーム位置をアノテーションした. 評価動画は総計 52 本, 時間の合計は 15.47 分であり, 含まれるスイング数は 75 本となった.

4.2 アラインメント精度

動作アラインメントの精度を検証するため, 評価動画すべてにおいて解析をおこない, アノテーション位置からの時間誤差 (秒) を測定した. 機材は Apple iPhone Pro 14 を使用した. 解析は提案手法に加えて比較対象として, 音の解析を併用しない場合, 式 1 の右辺第 3 項の正則化項を導入しない場合の音解析併用あり/なしの 4 パターンでおこなった. アドレス, トップ, インパクト, フィニッシュ各位置でのずれ (秒) を計測したときの分布を図 9 に, 各アノテーション位置での誤差の平均, 中央値, 分散を計測した結果を図 10 に示す. 誤差の平均値, 中央値に関しては小さいほど精度が良く, かつ分散が小さいほどロバストな解析ができていることを示す.

提案手法において, すべての評価データに対する総計算時間は 34.7 分を要した. さらに, そのうち 50.8 秒が実際のアラインメントの計算時間であり, 残り 97.6%の時間が骨格検出に占められていたことになる. スイングの検出率は, まず, 提案手法では 73/75 本検出, スイングではない動作の誤検出は 2 本であった. 音の解析を併用しなかった場合は, 72/75 本検出で誤検出は 7 本であった. 次に正則化項を落とした状態で音解析併用ありの場合は, 69/75 本検出, 3 本の誤検出, 音解析併用なしの場合は, 70/75 本検出, 7 本の誤検出であった.

© 2023 Information Processing Society of Japan

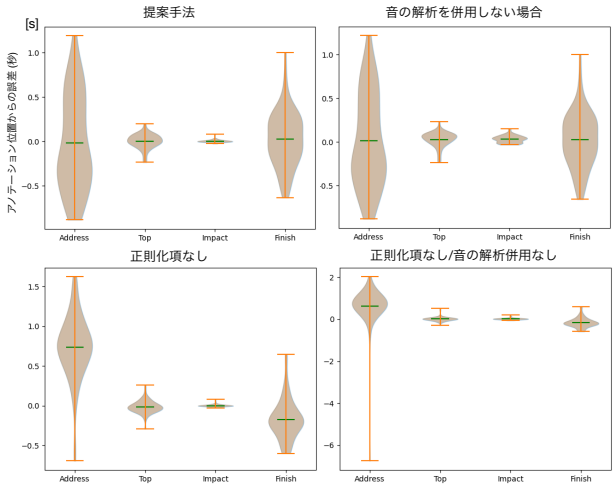


図 9　アラインメント精度比較. 評価動画におけるアノテーションフレームからのずれの分布 (秒).

平均値	提案手法	音なし	正則化なし	正則化/音なし
アドレス	0.461475	0.45372	0.771655	0.849077
トップ	0.0619167	0.0719832	0.058708	0.0664834
インパクト	0.0100407	0.0417696	0.0100434	0.0331777
フィニッシュ	0.24796	0.250969	0.251963	0.234093

中央値	提案手法	音なし	正則化なし	正則化/音なし
アドレス	0.628813	0.641313	0.812811	3.37135
トップ	0.117366	0.117411	0.145833	0.25831
インパクト	0.041635	0.0750701	0.041635	0.108228
フィニッシュ	0.500028	0.500028	0.327482	0.306939

分散	提案手法	音なし	正則化なし	正則化/音なし
アドレス	0.0679616	0.0726853	0.109203	0.633413
トップ	0.00299434	0.00291555	0.00316304	0.0073279
インパクト	0.000144944	0.00091117	0.000167288	0.00118329
フィニッシュ	0.0352703	0.0360038	0.0270975	0.0238111

図 10　アラインメント精度比較. 評価動画におけるアノテーションフレームからの誤差 (秒) の平均値, 中央値, 分散.

このように音の解析を併用しない場合は, スイングではないが似ている動作を弁別することができず, 誤検出が増えてしまうことがわかる. また音も考慮していることによりインパクト位置の精度が平均値, 中央値両方において改善されており, かつ分散値が低くなっているため, 非常に安定して解析ができていることを示している (図 10). アドレスとフィニッシュの推定位置については, かなり分散が大きく分布してしまっている (図 9). これは, 正確なアドレスとフィニッシュ位置というもの自体がかなり曖昧であり, 人が手動で付けたアノテーション自体に大きなばらつきがあることが主な原因である. また, SDTW では一般的に始点での精度がもっとも悪くなる傾向があることも原因のひとつであると考えられる.

正則化項を導入することによる効果は, 終端から極端に短くアドレス位置まで到達しない最尤パスが求まってしまうことを避け, スイング動作の最初のフレームから終端のフレームまで全体をカバーできる長いパスを精度良くアラインメントできることにある. 正則化項が無い場合は, 短絡的な SDTW の最尤パスが計算されてしまいがちであり, 結果として終端フレームからバックトレースしたときにアドレス位置まで到達しない. このとき正しいインパクト位置付近を通過できない可能性が高く, 適切なインパクト音検出が働かないため仮スイングが実スイングとして採用されず棄却されてしまう. これが正則化項を落とすことによって検出率が減少してしまう原因となっている.

最後に既存技術と比較するため, GolfVision [18] にも評価用動画を取り込んだ. 結果, スイングとして検出されたのは 20/75 本のみであった. さらに, このうちインパクト位置が大きくずれている (トップとインパクト位置の中間やすでに打ち終わったあとになってしまっている) ものが 13 本あった (図 11: 右). このことからも提案手法の検出率とアラインメント精度は十分高いものであると考えられる.

4.3　ユーザスタディ

提案手法を iOS アプリケーションとして実装し, 実際に中級者から上級者の 32 人のゴルフプレイヤーに体験してもらった (図 11: 左). 被験者の内訳は 20 代が 7 人, 30 代が 2 人, 40 代が 7 人, 50 代が 12 人, 60 代が 4 人で男性 31 人, 女性 1 人であった. この iOS アプリには撮影機能が実装されており, スイングの様子を背後から撮影すると撮影終了とともに解析が走り, それぞれ個別のスイングがリストに表示される. リストに表示されたスイングは単独で要所となる各フォームを確認できるようになっているのに加え, 2 つの異なるスイングを選択することで要所となる 4 点のフォームを比較することもできる (図 12).

各被験者は 1, 2 回程度自分のスイングを撮影したのち, 解析結果を確認した. この実験中に解析に失敗したもの (スイングとして検出できなかったもの) は 1 回であった. 体験後, それぞれの被験者にはアンケートに答えてもらった. 質問項目は, 「Q1. 普段からスイングの様子を撮影しているか」, 「Q2. 撮影しているなら機材は何を使っているか」, 「Q3. アラインメント機能 (実際には AI スイング認識機能と呼称した) は実際に使いたいかどうか」, 最後に Q4. 自由記述の感想, である.

結果として, 被験者の 69%は実際に普段から自分のスイングの様子を撮影して練習をしており, その全員がスマートフォンで撮影をおこなっていた. さらに彼らは動画を見返すためにスイング位置を探したり, 後から編集する手間を感じていたことがわかった. そのため, 本アプリの機能は非常にポジティブに受け入れられており, 全員がアラインメント機能を練習で使いたいとした中, 14 人が「有料でも使いたい」, 18 人が「無料なら使いたい」と答えた. また, 異なるスイング間で精度良く要所となるフォームを見

© 2023 Information Processing Society of Japan

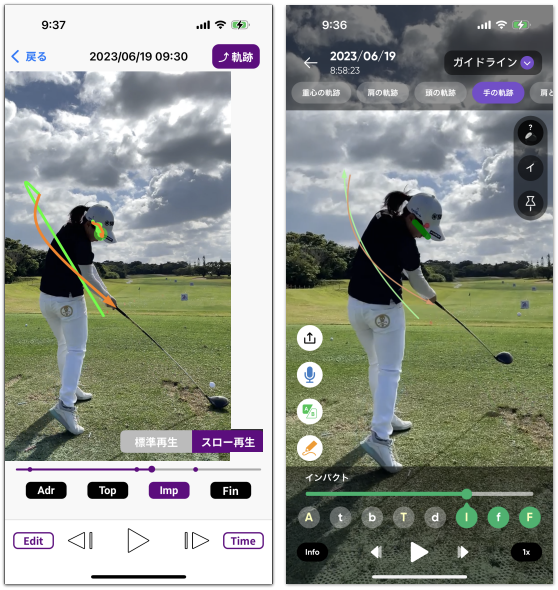


図 11　(左) iOS アプリケーションとしての実装と (右) GolfVision [18] で同じ動画を解析してインパクト位置を表示した結果．提案手法では正確にクラブがボールを打つ瞬間をとらえることができている．

比べることができるようになったため，インパクト時の下半身の使い方など，身体の使い方の違いが非常にわかりやすくなった，との意見も得ることができた．

2 つの異なるスイングの比較のユーザへの見せ方として大きく 2 通りに分けることができる．ひとつはそれぞれの動画のフレームレートは変えず，アドレス，トップ，インパクト，フィニッシュのいずれかひとつの要所のフレーム位置を揃えて表示する方法，もうひとつはそれぞれの再生フレームレートを可変にして，アラインメントされたすべてのフレーム同士を揃えて表示する方法である．前者はスイング時のクラブの振りの速度を確認するために有用であり，後者はフォームを比較するために有用であると期待される．本実験ではアプリケーションに前者を実装して被験者に体験してもらったが，後者の見せ方についても実際に練習で使いたいかどうかも追加でインタビューをおこなった．結果，後者においては今まで実現できたソフトウェア機能の前例が無かったため，体験者もそれが本当に有用であるかはある程度長い期間試用してみないとわからない，との意見が得られた．これに関しては今後も継続した調査が必要である．

4.4　議論

提案手法により非常に精度の高いスイング動画の検出とアラインメントが実現できたが，これをそのまま実環境で使用するにはまだいくつかの問題が残っている．まず，人が手動でスイングの要所位置をアノテーションしていることによる解釈のバラつきは大きな問題となる．とくに，アドレスとフィニッシュ位置については定義があいまいであり，人によってアノテーション位置が大きく変わってしまう．また，ゴルフクラブやボール自体の認識をしているわけではないことも問題である．例えばアドレス時に手はほとんど動かさずにクラブだけ揺らす動作から入るようなスイングの場合，実際にシステムが認識するのは手が動き始めてからであるため，少し遅いタイミングでアドレス位置が認識されてしまう．これに対し，実使用上は，スイングが開始されてしまった後にアドレスが設定されるよりも，動き出す前に動画を頭出しできるほうがユーザにとっては便利であると考えられるため，一律に前のフレームにシフトするほうが現実的である．一方，フィニッシュについては単発のスイングを抜き出した時に，完全にスイングが終了した時点までがアラインメント範囲としてカバーされることが期待されるので，推定された結果よりも一律に少し後ろにずらして提示したほうが好ましいと考えられる．

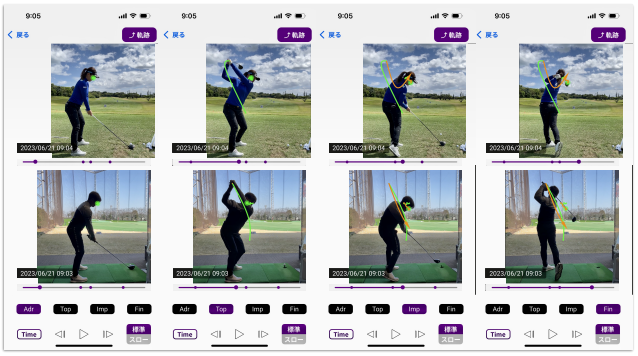


図 12　ユーザ評価のための iOS アプリケーション．ユーザが撮影した練習動画からそれぞれ単発のスイングを自動で抜き出し，2 つの異なるスイング間で要所となるフォームを比較できる．

提案手法では動作の解析と音の解析を併用することで，素振りなどのスイング以外の動作と本スイングを識別している．しかしながら，複数人が並んで練習しているような場所では，素振りのタイミングにたまたま他の人の打球音が重なって入ってくる場合がある．このような場合に提案手法ではボール有りの本スイングと誤検出してしまう．実際に §4.2 において誤検出された 2 本のスイングの原因は，この素振りに重なった他の人の打球音である．こうした問題を解決するためには，ゴルフボールやゴルフクラブ自体の認識をおこなったり，インパクト音以外の音の解析をおこなう必要があるが，これは今後の課題である．

§4.2 において本スイングが 2 本検出できなかった主な原因は，ゴルフプレイヤーが黒い服を着てかつ暗い練習場で撮影していたり，またはスカートなど身体の骨格がわかりにくい服を着ていたため，骨格検出の精度が落ちてしまったからであったと考えられる．また，撮影環境によっては，音がそのまま収録されず，オートゲインコントロールがかかってしまっており，この場合にインパクト音のオンセット検出がうまく動作しないこともあった．前者を解決するためには，骨格検出より前に画像の前処理，または撮影時

© 2023 Information Processing Society of Japan

のカメラパラメータの調整をする必要がある．また, 後者に関しては, アプリケーション自体に撮影機能を組み込み, 余計な信号処理をおこなうことがないような自然な撮影から解析への導線を確保することが現実的である．

5. まとめ

本稿では, ゴルフスイングの練習支援のため, 複数の異なるゴルフスイング動画での人間の動作のフレーム間対応をアラインメントして提示する手法を提案した．提案手法では, 動画像処理と音解析を組み合わせたマルチモーダル解析により, スイング以外の余計な動作も含む撮りっぱなしの練習動画から, 精度良く目的のスイングだけを抜き出してフレーム間対応を求めることができる．これによってユーザはスイングひとつひとつを分けて撮影したり, 練習動画を後から編集する手間から解放され, スイングの練習自体に集中することができる．実験では提案手法が既存手法を検出率およびアラインメント精度両面において大きく上回ることを定量評価で示し, さらにユーザスタディでは実際にスイング練習に有用であることが示された．

本稿で提案した手法はアノテーションの方法やハイパパラメータのデザインを変えることで野球やテニスのスイングやサッカーのシュートなどの動作にも適用できる可能性がある．今後はハイパパラメータを自動調整する手法を導入するなどすることによって他の種類の動作にも提案手法を広く適用していく．さらに, 今後は継続したアプリケーションの配布を通して広くユーザ調査をおこない, 練習の様子を撮影して上級者のフォームを見比べることが初心者がスイングを上達していくために本当に効率的な方法であるのか, を明らかにしていく．

参考文献

[1] Jens Barth, Patrick Kugler, Dominik Schuldhaus, Jurgen Winkler, Jochen Klucken, and Bjorn Eskofier, Subsequence dynamic time warping as a method for robust step segmentation using gyroscope signals of daily life activities, *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, Large-scale video classification with convolutional neural networks, *CVPR*, 2014.

[3] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, Temporal segment networks for action recognition in videos, *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[4] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, Beyond short snippets: Deep networks for video classification, *CVPR*, 2015.

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venu- gopalan, K. Saenko, and T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, *CVPR*, 2015.

[6] S. Ji, W. Xu, M. Yang, and K. Yu, 3d convolutional neural networks for human action recognition, *IEEE transactions on pattern analysis and machine intelligence*, 2012.

[7] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition, *IEEE/CVF international conference on computer vision*, 2019.

[8] A.Arnab, M.Dehghani, G.Heigold, C.Sun, M.Lucic, and C.Schmid, Vivit: A video vision transformer, *IEEE/CVF International Conference on Computer Vision*, 2021.

[9] H. Duan, Y. Zhao, K. Chen, D. Shao, D. Lin, and B. Dai, Revisiting skeleton-based action recognition, arXiv preprint arXiv:2104.13586, 2021.

[10] S. Yan, Y. Xiong, and D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, *AAAI*, 2018.

[11] Y. Chen, Z. Zhang, C. Yuan, B. Li, Y. Deng, and W. Hu, Channel- wise topology refinement graph convolution for skeleton-based action recognition, *IEEE/CVF International Conference on Computer Vision*, 2021.

[12] H. Duan, Y. Zhao, K. Chen, D. Shao, D. Lin, and B. Dai, Revisiting skeleton-based action recognition, arXiv preprint arXiv:2104.13586, 2021.

[13] Zhensheng Shi, Ju Liang, Qianqian Li, Haiyong Zheng, Zhaorui Gu, Junyu Dong, and Bing Zheng, Multi-Modal Multi-Action Video Recognition , *ICCV*, 2021.

[14] Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani, Listen to Look: Action Recognition by Previewing Audio, *CVPR*, 2020.

[15] Muhammad Bilal Shaikh, Douglas Chai, Syed Mohammed Shamsul Islam and Naveed Akhtar, MAiVAR: Multimodal Audio-Image and Video Action Recognizer, *IEEE International Conference on Visual Communications and Image Processing (VCIP)*, 2022.

[16] Fei Wu, Qingzhong Wang, Jiang Bian, Haoyi Xiong, Ning Ding, Feixiang Lu, Jun Cheng, and Dejing Dou, A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications, 10.48550/arXiv.2206.01038, 2022.

[17] Taein Kwon, Bugra Tekin, Siyu Tang and Marc Pollefeys, Context-Aware Sequence Alignment using 4D Skeletal Augmentation, *CVPR*, 2022.

[18] GolfVision, SKYROAD 株式会社, 2022

[19] Yingjiang Li, Jianhong Sun, and Rui Li, Human Action Recognition Based on Dynamic Time Warping and Movement Trajectory, *International Conference on Electrical Engineering and Informatics*, 2011.

[20] Mohshi Masnad, G.M. MukitHasan, Kazi Md. Iftekhar, and Md. Sadekur Rahman, Human Activity Recognition using DTW Algorithm, *IEEE Region 10 Symposium (TENSYMP)*, 2019.

[21] 金 宏潤, 土田修平, 寺田 努, 塚本昌彦, 空手の予備動作を小さくするための慣性センサを用いた練習支援システム, インタラクション, 2012.

[22] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg and Matthias Grundmann, MediaPipe: A Framework for Perceiving and Processing Reality, *CVPR*, 2019.

© 2023 Information Processing Society of Japan