卒業研究報告書

題目

バスケットボール技術向上に向けた画像認識の研究と分析

指導教員

石水　隆　講師

報告者

21–1–037–0267

今泉　太貴

近畿大学理工学部情報学科

令和 7 年 2 月 2 日提出

概要

現在のスポーツ界では様々な戦術が誕生し, 競技者の技術向上が様々な側面で進展している. この発展に大きく貢献している技術の一つとして「画像認識」が挙げられる. 画像認識の先行研究の例としてはバスケットボールの試合において個人の試合中の運動特徴を画像から抽出し, 機械学習により選手の動作を分類, 認識しシュートの動作を客観的に分析してその人に合うシュートフォームを考察するといったものがある [1], [2].

本研究では, 初心者のバスケットボールのシュート技術向上を目的として, 画像認識技術を活用した分析を行った. また, 画像認識においてどのような画像データが有益な情報を提供するかについての考察を行った. 初心者のシュートフォームを解析し, シュート成功時と失敗時それぞれのデータを活用して個々のプレーヤーに適した指導を行うとともに, 経験者のデータとの比較により具体的な改善点を明確化することで, 個々のプレーヤーに適した技術の向上を図ることが可能である. さらに, 経験者のフォームを理想的フォームとして設定し, 差異を数値化することで, 初心者の練習意欲向上にも寄与する可能性を示した.

目次

1 **序論** 4

1.1 研究背景 4
1.2 本研究の目的 4
1.3 先行研究との位置付け 4
1.4 本報告書の構成 4

2 **本研究で使用した言語, ライブラリ** 5

2.1 Python 5
2.2 OpenCV 5
2.3 Mediapipe 5
2.4 ランドマーク 5
2.5 Math 6
2.6 Matplotlib 6

3 **研究内容** 6

3.1 システム構成図 6
3.2 データの収集 8
3.3 動画フレームの取得と保存 10
3.4 姿勢推定, それ基づく散布図の作成 11
3.5 姿勢推定 12
3.6 関節の角度の算出, グラフ化 15

4 **考察** 18

4.1 初心者の改善点 19

5 **結論と今後の課題** 22

付録 A ソースプログラム 25

1 序論

1.1 研究背景

スポーツ競技全般において, それぞれの競技には長年の経験からできた練習法や最適なフォームが存在する. バスケットボールもその例外ではなく, 得点をいかに効率的に取るかを競うゲームであるバスケットボールにおいては, 正確なシュート能力が必要となる. しかし正確なシュートをすることは困難であり, 初心者においてはハードルが高いのは明らかである. 近年は動画サイトの普及により, 動画や画像から経験者と初心者の差異は大まかに認識されるようになったが, 具体的な差異や改善点を見出すのは難しいのが現状である. 筆者自身もバスケットボールを始めた際にプロの動画を参考に技術向上を図ったが, 初心者の段階で模倣するのには限界があり, その差異を縮める練習法を見つけられず, 挫折した経験がある.

1.2 本研究の目的

本研究では, 初心者と経験者のシュート動作の差異を画像認識技術を用いて数値的に解析する手法を提案することを目的としている. この手法により, 指導者は動作の改善点をより具体的に把握することが可能になり, 指導内容の精度向上が期待できる. また, プレイヤー自身も改善点を明確に知ることで, 練習の効率化が図られる. また本研究では経験者のデータを取得し, 経験者のフォームを理想的フォームと仮定した上で, 理想的フォームとの差異を可視化することにより, プレイヤーに対して具体的な目標を提示できるとともに, その差異が縮まる過程を数値で視覚的に可能となる. これによるプレイヤーの練習に対するモチベーションの向上も期待される.

1.3 先行研究との位置付け

画像認識技術を用いた先行研究においても, シュート時における姿勢解析を行なっており, 個人のシュートフォームとシュート成功確率の関連性をより明確にすることを今後の課題としている.

Zaima は動画の中からポーズ動作の特徴抽出に画像認識を使用しており, 動作認識を行うことで, パフォーマンス向上に向けた研究を行っている [1]. また,Rong はそれぞれの画像から運動特徴を抽出し, 機械学習を行い選手の動作を分類, 認識しシュートの動作を客観的に分析して優れたシュート動作を考察している [2].

本研究では, 初心者と経験者とのシュート動作の比較を通じて初心者の技術向上に着目した上で, 指導の効率化を図る. 初心者特有のシュート動作がシュート成功率にどのような影響を及ぼすかを数値的に明確にすることを目的にする. これにより, 初心者の技術向上における新しい知見を提供し, バスケットボールの指導と練習の質を高めることが期待される.

1.4 本報告書の構成

本報告書の構成は以下の通りである. まず第 2 章で本研究で使用したツールやライブラリーについて示し, 第 3 章では本研究のデータ収集から角度の計算, 散布図の作成までの過程及び結果を示し, 第 4 章で結果に対する考察を述べ, 第 5 章で結論及び今後の課題について示す.

2　本研究で使用した言語, ライブラリ

本章では本研究で使用した言語やライブラリの詳細について示す.

2.1　Python

本研究では VScode 環境を用いて, 一般的なプログラミング言語である Python を使用した.Python は専門的なライブラリが豊富であるため, データ処理や機械学習, グラフ作成を効率的に行うことができる.

2.2　OpenCV

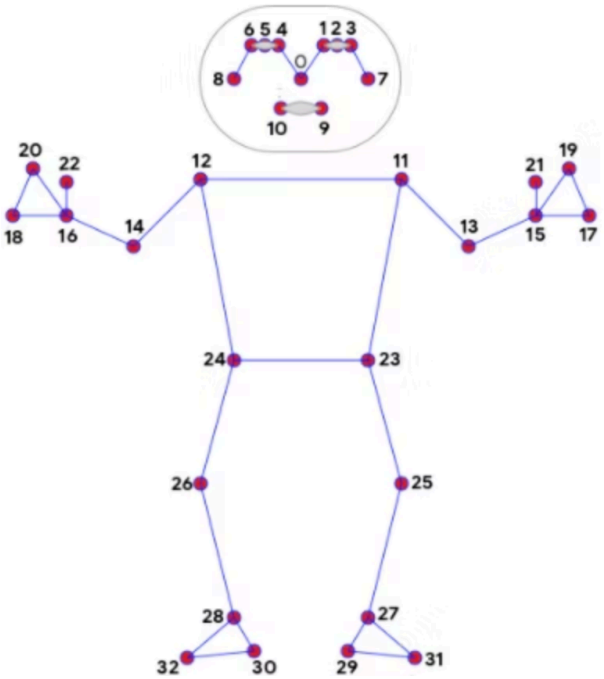
本研究では画像処理のために OpenCV [11] を利用した.OpenCV は, 画像解析や動画処理を効率的に行うための多くの機能を提供しているオープンソースのライブラリである.

2.3　Mediapipe

本研究では姿勢推定を行うために Mediapipe [12] を利用した.Mediapipe は Google が提供するオープンソースのライブラリで, 機械学習を活用したメディア処理のためのツールである. 特に短期のメディア解析を行うのに有用である. 今回のようなフリースローの瞬間的な動作の姿勢推定には非常に効果的である.

2.4　ランドマーク

姿勢推定を行う上で,Mediapipe モデルは, 人体における 33 個のランドマークを提供する. これにより, 姿勢推定で得られたランドマーク番号を元に部位を識別し, それぞれの部位の x 座標,y 座標, データの精度を得る. ランドマーク番号におけるそれぞれの部位の情報を図 1に示す.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

図 1　ランドマーク

2.5　Math

姿勢推定で得た部位の座標から,「肘、肩、手首」に着目して, ベクトル計算を行い角度を求める際に Math [13] を用いた.Math は様々な数学的な関数や定数を提供するモジュールである. 三角関数や対数関数といった数学的な処理を容易に行える.

2.6　Matplotlib

実際に算出したデータを視覚的に表現するために Matplotlib [14] を用いた.Matplotlib は, 折れ線グラフや散布図, ヒストグラムといった様々なグラフを描画するのに適していることから, 今回のような膨大な数値データを可視化する手段としては最適である.

3　研究内容

本章では本研究のデータ収集から角度を求める計算までの過程を示す.

3.1　システム構成図

本研究を行うにあたって, あらかじめ想定していた手法とその流れをまとめたものを以下の図 2に示す. 図 2では, 収集した動画をフレーム化する過程, 静止画像とフレーム画像をどのように解析して散布図を得たかの過程, そして最終的に導出した出力結果の内容についてを示す.

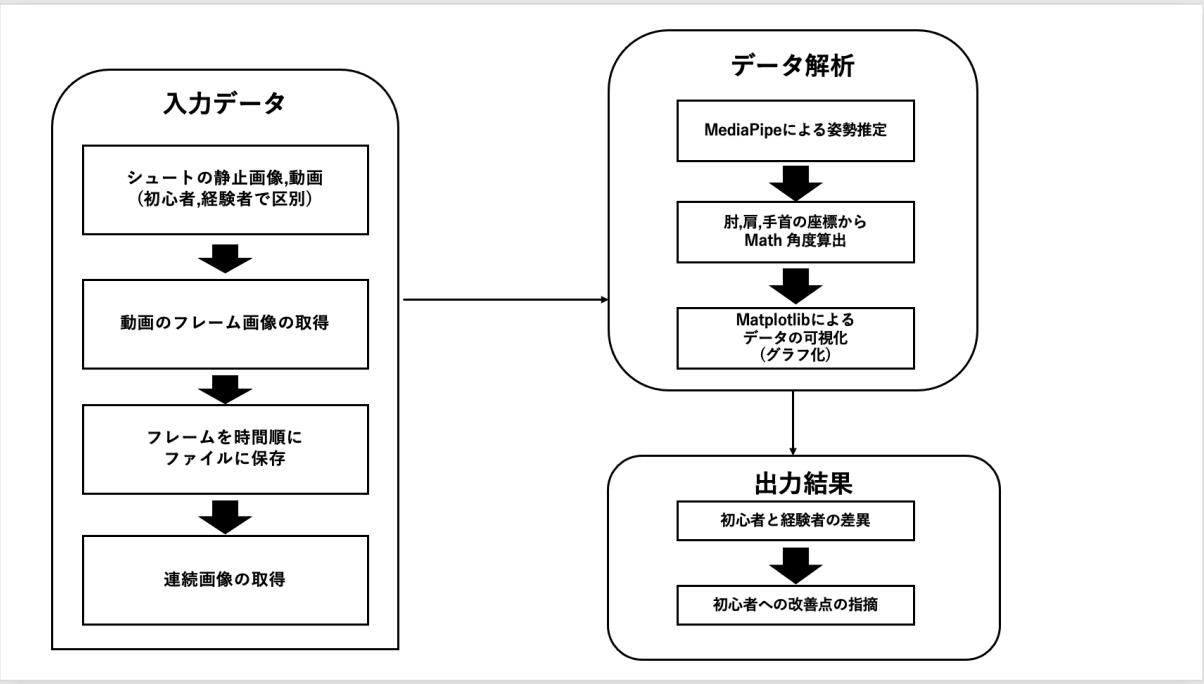

入力データ
シュートの静止画像,動画
（初心者,経験者で区別）
動画のフレーム画像の取得
フレームを時間順に
ファイルに保存
連続画像の取得
データ解析
MediaPipeによる姿勢推定
肘,肩,手首の座標から
Math 角度算出
Matplotlibによる
データの可視化
（グラフ化）
出力結果
初心者と経験者の差異
初心者への改善点の指摘

図 2　システム構成図

3.2　データの収集

本研究を行うにあたって, バスケットボール初心者である被験者 A と, 経験者である被験者 B の協力のもとフリースローの静止画像, 動画を撮影した. 初心者, 経験者共に「静止画像 15 枚, 動画 15 本」のそれぞれ合計 30 回のシュートを行いデータを収集した. 撮影の流れから保存までの流れを示したアクティビティ図を図 3に示す. 具体的には, 撮影方法および撮影で得たデータをどのようなディレクトリー構造で保存したかの過程を示す. 図 3に示したように, 撮影は異なる画角である 0 度,45 度,90 度で行い, 視点の違いが解析結果にどのような影響を及ぼすかの考察も目的としている. それぞれの画角で撮影した写真の例を図 4〜6に示す.

図 3　撮影から保存までの流れ

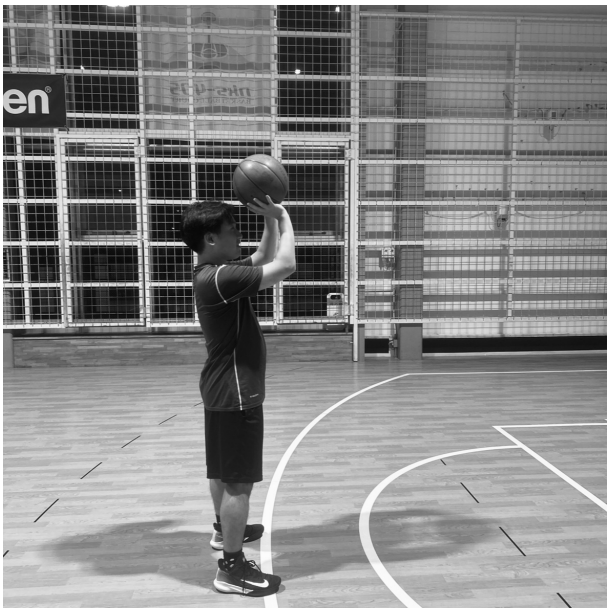


図 4　0 度画角の初心者

図 5　45 度画角の初心者

図 6　90 度画角の初心者

3.3　動画フレームの取得と保存

前節で述べた通り, 本研究では動画と静止画像を用いる. しかし, 動画のままだと姿勢推定を行う上で次の点で問題が生じるため動画をフレーム化する. 一つは姿勢推定は画像単位での入力が必要なるため, 動画形式のままだと処理が難しくなる点である. また処理をより効率化する点でも動画をフレーム化することは有効的である. これは, 動画をフレーム化することにより特定の時間やフレームの姿勢データを得ることができ, 必要な部分だけを解析できるためである.

動画をフレーム化した後フレームをバラバラに保存するのではなく, 撮影した時間を元に順番に保存することで姿勢推定で得た数値を確認する際に, どのフレームと対応しているかを即座に把握できる. これは初心者の指導の際の画像提示を容易にする目的もある. また本研究は初心者と経験者の両者の「手首, 肘, 肩」を中心とした関節角度を算出する. そのため撮影時間順にフレームを整理することでそれぞれの計算を行う上で再利用性が向上し, 結果データの照合にかかる手間を大幅に削減できるといった利点を持つ. 動画フレームの取得から保存までの一連の流れを示したアクティビティ図を図 7に示す. 図 7では撮影した動画をどの尺度でフレーム化するか, 得たフレーム画像をどのように保存しているかを示す.

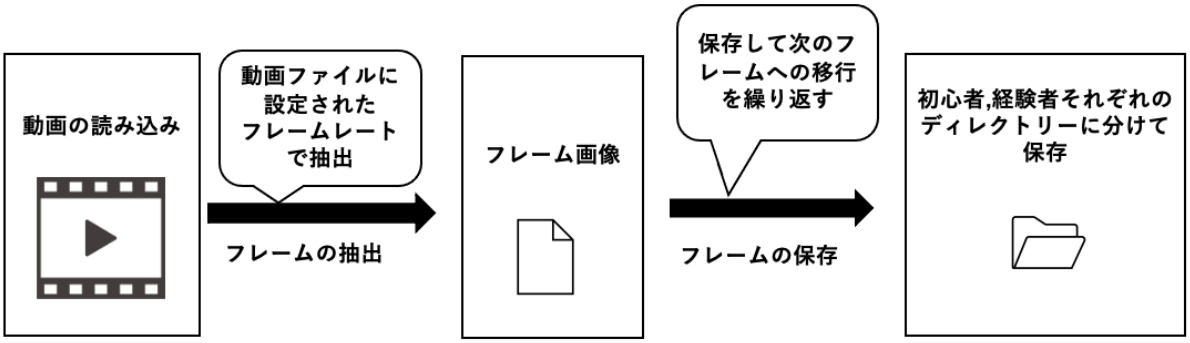


図 7　動画フレーム化と保存処理のアクティビティ図

3.4　姿勢推定, それ基づく散布図の作成

姿勢推定から散布図の作成までの一連の流れを示したアクティビティ図を図 8に示す. 図 8では, フレーム画像と静止画像の姿勢推定を行い, 得られる CSV ファイルの値と,Matplotlib で作成した散布図に示される情報について説明している. 以降の説では全体的なプロセスを示した後, 各ステップにおける詳しい内容を示す.

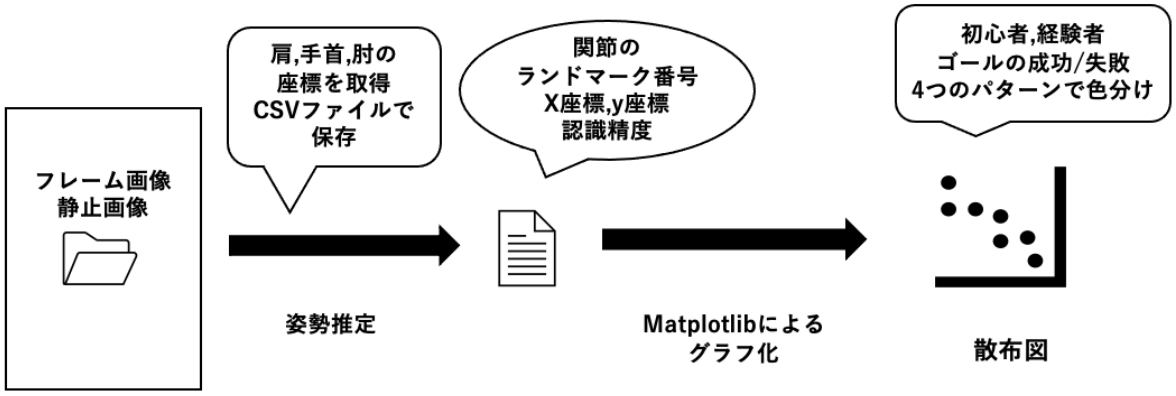


図 8　散布図作成のプロセス

3.5　姿勢推定

本研究では MediaPipe を用いて静止画像における姿勢推定を行なった. この処理は, 各フレーム画像にも適用可能なので省略し, ここでは静止画像に限定して説明する.

静止画像の姿勢推定

以下に静止画像の姿勢推定を行った Python コードを示す.

```
import cv2
import mediapipe as mp
import os
import pandas as pd

# MediaPipe の準備
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5)
mp_drawing = mp.solutions.drawing_utils

# 姿勢設定をして保存
def process_images(image_dir, result_dir):
    # 結果ディレクトリを作成
    os.makedirs(result_dir, exist_ok=True)
```

```
15      print(f"結果を保存するディレクトリ: {result_dir}")
16
17      # 静止画像を処理
18      images = [f for f in os.listdir(image_dir) if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
19
20      for image_name in images:
21          image_path = os.path.join(image_dir, image_name)
22          img = cv2.imread(image_path)
23
24          if img is None:
25              print(f"画像を読み込めませんでした: {image_path}")
26              continue
27
28          # BGR 画像を RGB に変換
29          img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
30          results = pose.process(img_rgb)
31
32          # 姿勢データを保存するリスト
33          pose_data = []
34
35          # 推定結果からランドマークデータを取得
36          if results.pose_landmarks:
37              for idx_landmark, landmark in enumerate(results.pose_landmarks.landmark):
38                  pose_data.append({
39                      "landmark": idx_landmark,#ランドマーク番号
40                      "x": landmark.x,#x 座標
41                      "y": landmark.y,#y 座標
42                      "z": landmark.z,#z 座標
43                      "visibility": landmark.visibility#認識精度
44                  })
45
46              # データフレームに変換
47              df = pd.DataFrame(pose_data)
48
49              # CSV ファイルとして保存
50              csv_filename = os.path.splitext(image_name)[0] + ".csv"
51              csv_path = os.path.join(result_dir, csv_filename)
52              df.to_csv(csv_path, index=False)
53              print(f"姿勢データを保存しました: {csv_path}")
54
```

```
55            else:
56                print(f"姿勢データが取得できませんでした: {image_path}")
57
58    # 初心者と経験者の静止画像に対する姿勢推定
59    zumi_image_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/zumi/images"
60    zumi_result_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/zumi/z-result"
61
62    kana_image_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/kana/images"
63    kana_result_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/kana/k-result"
64
65    process_images(zumi_image_dir, zumi_result_dir)
66    process_images(kana_image_dir, kana_result_dir)
67
68    # リソース解放
69    pose.close()
```

以下に Python コードの各部位の処理の内容について説明する.

1 行目から 4 行目はそれぞれ必要なモジュールのインポート行っている.

7 行目から 9 行目で姿勢推定のための MediaPipe の準備, 具体的には Pose で静止画像ように姿勢推定モデルを初期化し静止画像向けに設定する.

12 行目から 15 行目は, 姿勢推定を行いその結果を保存する関数であり, ディレクトリー内の画像に対して姿勢推定を行いそれを csv ファイルとして指定したディレクトリーに保存する.

18 行目では画像をフィルタリングをしている.os.listdir(image_dir) で image_dir の中にあるすべてのファイルとフォルダーの名前をリストとして取得し,

拡張子が jpg,jpeg,png のデータのみを新しいリストに追加する. これはフォルダーの中に誤って PDF など違うデータが含まれてエラーが出るのを防ぐためである.

20 行目から 26 行目では,OpenCV で画像を読み込み, 読み込めなかった場合はエラーメッセージを表示する.

29,30 行目は BGR 画像を RGB に変換し, 姿勢推定モデルに画像を渡して姿勢推定を行う.

36 行目から 44 行目は, 姿勢推定から得られるランドマークデータと座標, 姿勢推定の精度をそれぞれ取得している.

47 行目から 56 行目は, 前で得たランドマークデータをデータフレームに変換し, 画像ファイルの拡張子を削除し CSV ファイルを生成し結果を保存する. この時, ランドマークデータが検出されない場合はエラーメッセージを表示する.

59 行目から 66 行目は,CSV ファイルをそれぞれの結果として保存するディレクトリーを設定する.

最後に 69 行目で,pose.close() で MediaPide の Pose オブジェクトで使用されるリソースを解放している. これは, 今回のように大量の画像の処理を行っている際に, リソースを適切に管理しないとメモリーを無駄に使用してしまい, メモリー不足を引き起こすのを防ぐためである.

姿勢推定を行う Python コードで得られた CSV ファイルの結果の一部を図 9に示す. 図 9の実行結果には,「関節の種類を示すランドマーク番号,x 座標,y 座標,z 座標, 認識精度」の数値が表示されている.

data > kana > k-result > kana1.csv

```
landmark,x,y,z,visibility
0,0.41536062955856323,0.36398255825042725,-0.1609107255935669,0.9999043941497803
1,0.40958356857299805,0.35137656331062317,-0.129765585064888,0.9998886585235596
2,0.409542441368103,0.35119834542274475,-0.12990102171897888,0.9998577833175659
3,0.40943780541419983,0.3509775698184967,-0.1300121545791626,0.9998999834060669
4,0.403238445520401,0.3519473075866699,-0.17927953600883484,0.9998853206634521
5,0.3972623348236084,0.35219550132751465,-0.1790749430656433,0.9998689889907837
6,0.3903176188468933,0.35260719060897827,-0.17905209958553314,0.9998810291290283
7,0.38136956095695496,0.3583838939666748,0.029006872326135635,0.9998989105224609
8,0.35311710834503174,0.36338913440704346,-0.1850041002035141,0.9999213218688965
9,0.4123314619064331,0.37866321206092834,-0.09782838821411133,0.9998784065246582
10,0.4032936990261078,0.3811378479003906,-0.15996310114860535,0.9998888969421387
11,0.38228991627693176,0.4501480758190155,0.1869550496339798,0.9992651343345642
12,0.3052079677581787,0.4487665295600891,-0.3181544840335846,0.9993273019790649
13,0.505439043045044,0.4631367325782776,0.043689485639333725,0.660961925983429
14,0.43010246753692627,0.48794007301330566,-0.5311225056648254,0.9937258362770081
15,0.5762302875518799,0.3852938711643219,-0.27086690068244934,0.7398136854171753
16,0.497711181640625,0.38331103324890137,-0.522930920124054,0.9849901795387268
17,0.5911957025527954,0.3594511449337056,-0.33215567469596863,0.6628091335296631
18,0.5124570727348328,0.361433207988739,-0.5743971467018127,0.9668394923210144
19,0.5813354253768921,0.3526425063610077,-0.3187950551509857,0.6479862928390503
20,0.5037053823471069,0.3553864359855652,-0.5639765858650208,0.9614397287368774
21,0.5707638263702393,0.3642243444919586,-0.28064364194869995,0.6463853716850281
22,0.4987671971321106,0.3627859652042389,-0.5190509557723999,0.9469985365867615
23,0.3520505130290985,0.6683339476585388,0.16603314876556396,0.9991222023963928
24,0.3131842017173767,0.6689750552177429,-0.16588450968265533,0.998747706413269
25,0.4083675146102905,0.8127021789550781,0.16639302670955658,0.3986564874649048
26,0.4554925262928009,0.8118994832038879,-0.21044151484966278,0.9702082872390747
27,0.326286643743515,0.9671315550804138,0.2691705822944641,0.489254891872406
28,0.3344268500804901,0.9633468389511108,-0.11895138025283813,0.9166555404663086
29,0.2895999550819397,0.984765887260437,0.27004945278167725,0.5215973258018494
30,0.29419708251953125,0.9796828031539917,-0.11424374580383301,0.8562695980072021
31,0.39380306005477905,1.002445936203003,0.18330153822898865,0.5788055658340454
32,0.3938528895378113,1.0019729137420654,-0.206144317984581,0.8868423700332642
```

図 9　姿勢推定を行う Python コードの出力例

3.6　関節の角度の算出, グラフ化

本節では, 姿勢推定で実際に求めた関節部分の座標を用いて,「肩, 手首, 肘」に注目して初心者, 経験者のそれぞれの角度をベクトル計算を用いて求める. その後 Matplotlib を用いて散布図にして可視化する. 今回は角度のばらつき具合などから, 初心者と経験者のシュート動作の差異を可視化することを目的としているため, 散布図が有効と判断した. 以下に実際の角度の算出方法を示すとともに,「肩, 手首, 肘」の実際の散布図を示す.

角度の算出方法

関節の角度 θ は以下のベクトル計算を用いて求める．各関節の座標を $A(x_A, y_A)$，$B(x_B, y_B)$，$C(x_C, y_C)$ とすると，以下の式により計算することができる．

$$\vec{BA} = (x_A - x_B, y_A - y_B), \quad \vec{BC} = (x_C - x_B, y_C - y_B)$$

$$\theta = \cos^{-1}\left(\frac{\vec{BA} \cdot \vec{BC}}{\|\vec{BA}\|\|\vec{BC}\|}\right)$$

ここで，$\vec{BA} \cdot \vec{BC}$ はベクトルの内積，$\|\vec{BA}\|$ および $\|\vec{BC}\|$ はそれぞれのベクトルの大きさを示している．今回はボールの位置などの詳細な 3D 情報を極力無視し, 視覚的なわかりやすさを重視した. またそれぞれ異なる角度で撮影された写真を用いてるため,2D での姿勢推定を行なうことで計算負荷を軽減し、結果の解釈が容易になっている．

散布図の結果

初心者と経験者の「左肩, 左手首, 左肘」及び「右肩, 右手首, 右肘」に基づいて算出されたベクトル角度のばらつき具合を示す散布図を図 10〜15に示す. 本散布図は初心者と経験者のシュート動作における関節角度の特徴を視覚的に比較し, その差異やそれぞれの傾向を評価することを示している. よって「初心者, 経験者, ゴールの成功/失敗」の四つで色分けを行っている. 具体的には, 図中の赤い点は経験者がゴールを失敗した時, 青い点は初心者がゴールを失敗した時, 緑の点は初心者がゴールを成功した時, 紫の点は経験者がゴールを成功した時を表した散布図である．

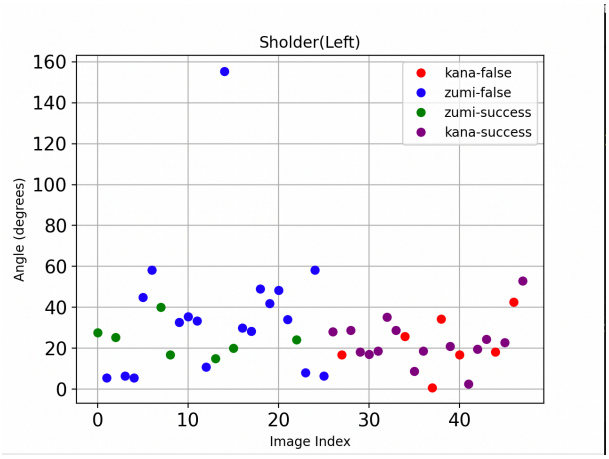


図 10　左肩の散布図

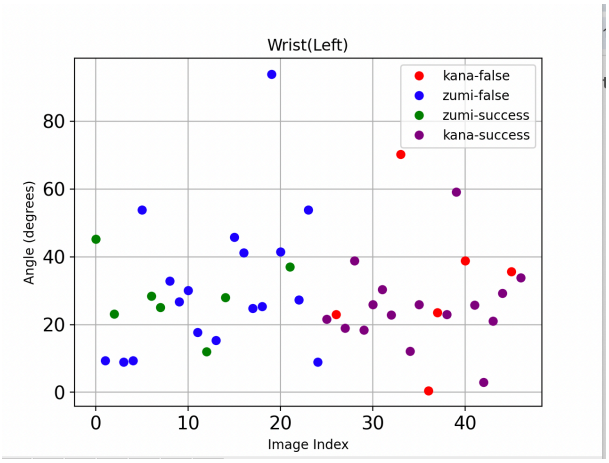


図 11　左手首の散布図

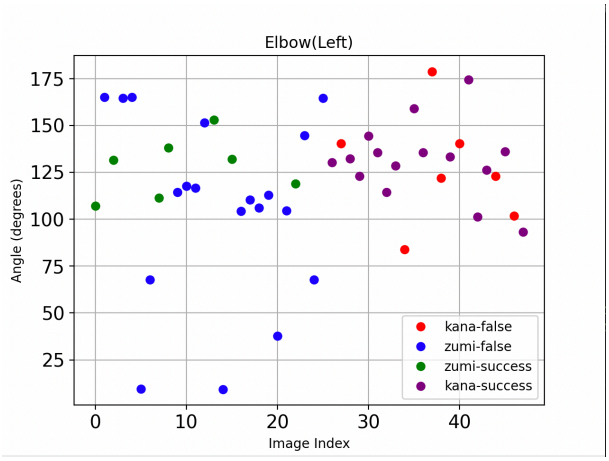


図 12　左肘の散布図

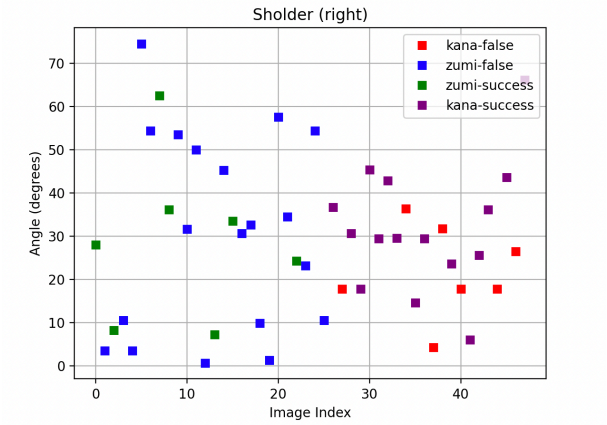


図 13　右肩の散布図

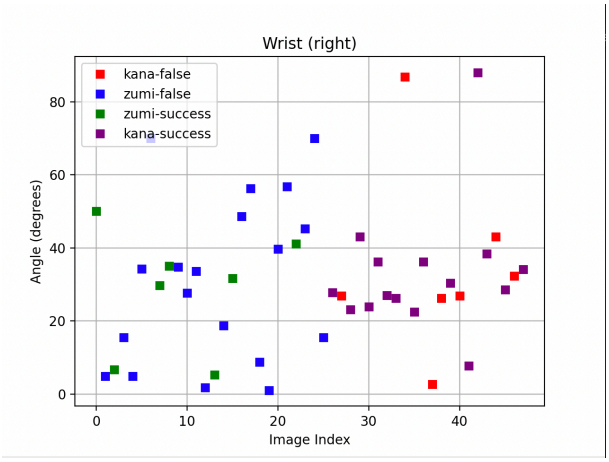


図 14　右手首の散布図

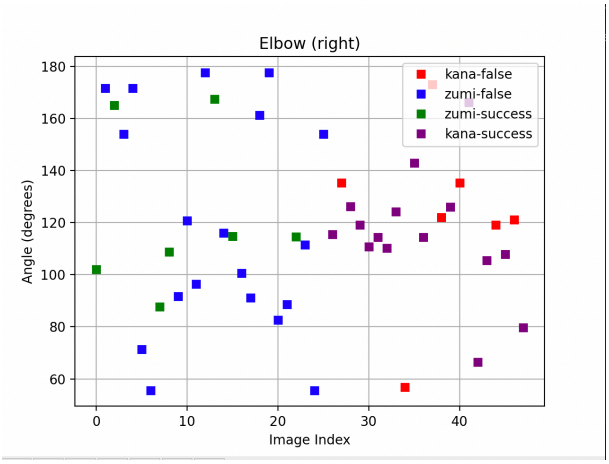


図 15　右肘の散布図

4　考察

本章では，本研究における結果をもとに考察を述べる．特に，「肩，肘，手首」の角度に着目した散布図の分析を通して，初心者と経験者のシュート動作における特徴的な差異について示す．散布図の結果を踏まえると, 経験者のシュート動作はゴール成功時はもちろん, それぞれのシュートにおいて関節の角度のばらつき具合が少なく, 非常に安定していることがわかる.「肩, 手首, 肘」の角度が一定の範囲内に収まっており, これはフリースローの状況下で安定したフォームの再現性が高いことがわかる. 一方で, 初心者のシュート動作は関節の角度のばらつきが広い範囲で分散しており, フォームが不安定であることが示唆される. この原因は, シュート動作の際におけるフォームの不安定さや体の鍛え具合による未熟さが影響していることが考えられる. その結果, 安定したフォームの再現性が低く, シュート成功率が低い原因であると推測される. この差異は特に図 11と図 12の散布図から読み取りやすい.

4.1 初心者の改善点

散布図の結果から, 左肘と左手首に着目した際に初心者と経験者の間で顕著な差異が見られた. この比較結果を基に, シュート動作の安定化を図るための安定したフォームを提案する. 具体的には, 初心者と経験者の「角度の平均値, 最小値, 最大値, 標準偏差, そしてゴール成功率とゴールが決まった際のそれぞれの角度の平均値」を用いて, 成功率向上を図るフォームを導き出すことを目的としている. 散布図に示した関節部分における初心者と経験者それぞれの値を表 1〜6に示す.

表 1　左肩の角度の比較

	平均値	最大値	最小値	標準偏差	成功率	成功した際の平均値
初心者	32.7°	155°	5.50°	30.2	0.32	25.6°
経験者	22.8°	52.9°	1.78°	11.8	0.73	22.6°

表 2　左手首の角度の比較

	平均値	最大値	最小値	標準偏差	成功率	成功した際の平均値
初心者	30.6°	94.0°	8.94°	18.5	0.32	28.2°
経験者	27.4°	70.2°	0.50°	15.2	0.73	23.2°

表 3　左肘の角度の比較

	平均値	最大値	最小値	標準偏差	成功率	成功した際の平均値
初心者	113°	165°	9.33°	42.8	0.32	126°
経験者	130°	179°	84.0°	22.6	0.73	132°

表 4　右肩の角度の比較

	平均値	最大値	最小値	標準偏差	成功率	成功した際の平均値
初心者	28.3°	62.5°	0.71°	19.9	0.32	28.9°
経験者	28.6°	66.1°	4.26°	14.0	0.73	31.0°

表 5　右手首の角度の比較

	平均値	最大値	最小値	標準偏差	成功率	成功した際の平均値
初心者	30.1°	70.0°	1.03°	21.4	0.32	28.4°
経験者	33.5°	88.0°	2.63°	19.9	0.73	32.5°

表 6　右肘の角度の比較

	平均値	最大値	最小値	標準偏差	成功率	成功した際の平均値
初心者	122°	178°	55.6°	38.2	0.32	123°
経験者	118°	173°	56.8°	26.8	0.73	117°

表 1〜6の標準偏差の値からわかるように, 初心者の角度の値は経験者の値に比べて分散しており, 表 1の左肩においては約 3 倍, 表 3の左肘においては約 2 倍ほど分散していることから, 安定さに大きな差があることが読み取れる. 初心者における改善点はまず左肘の角度をより大きくすることである. シュート時に腕の力をより効率的にボールに伝えることができ, シュートの安定性や精度が向上する. そのために以下の二点を意識して練習を行うよう指導する. 一つは肘の位置を今の状態より上げることである. 具体的には手のひらが額にくるぐらいの高さを意識することによって肘の位置を修正する. 二つ目は肘を伸ばすこと意識を持つことである. これは曲がっている状態だと力が入りすぎて, シュートが安定しないことが考えられるため, 二の腕が地面と平行になるように意識することでフォームを改善する. この調整は, 初心者が安定した動作を身につける基礎となる. そして左肩と左手首の改善点に関しては, 成功時の値において特段の修正点はないが, 散布図から角度のばらつきが多いことがわかる. 特に左肩に関しては, 一度だけ角度が 150 度以上となっており, 腕が非常に高い位置にあることがわかる. このことから, フォームが安定していないことが分かる. そしてこれは左肘の位置の不安定さが原因の一つであると考えられる. よって左肘の角度を改善することにより, 左手首と左肩も連動してシュート動作が安定し, シュートフォーム全体の精度が向上することが考えられる. また「右肩，右肘，右手首」に関しても同様に散布図からも多少のばらつき具合も視覚的に確認でき, 表 4〜6の標準偏差の値からも初心者のばらつき具合が大きいことがわかる. 特に右肘においては左肘同様にばらつき具合の差が大きいことから, 経験者と初心者のシュート動作の安定さに肘が大きく関係していることが推測される. 両者の利き手である右手に関して, 全体の平均角度と成功した際の平均角度の数値にそこまで差異がないことから, 初心者であっても右手は左手に比べて安定性があると考えられる。よって右手に関する指導法としては, 反復練習によって安定性をさらに高めることが有効であると考えられる. これにより、右肩や右手首を含めた動作全体が連動し、シュートフォーム全体の精度が向上することが期待される. 上記の肘の位置を意識した結果として, 練習過程で一時的にではあるが, 初心者の修正前のシュート成功率が 0.32 だったのに対して, 修正後は 0.47 に向上した結果も得られた. 以下に修正前と修正後を比較するための図 16と図 17を示す.

図 16　修正前

図 17　修正後

5　結論と今後の課題

本研究では, 経験者と初心者のシュートフォームを比較することで, シュート動作における安定さの差異を明らかにし, 求めた関節部位の角度データを用いて初心者のシュートフォームに対する改善点と指導法を提案することができた. しかし今回の撮影方法では, 下半身の動作, 具体的には膝や足首といった関節部分を含めた体全体の分析を十分にできなかった. そのため全身の動作を考慮した改善案を提案することができなかった. 今後の課題としては, さらに異なる角度や深度からの撮影を行い, 下半身を含めた姿勢推定が必要である. これにより全身を用いたシュートフォームの改善点を明らかにし, より正確な改善点と指導法の提案が可能になると考えられる.

謝辞

本研究を行うにあたり, 撮影の協力をしてくれた友人に感謝すると共に, 本研究の研究概要や中間報告書, 卒業研究報告書の推敲, および研究に対して客観的なご意見を頂き, 終始多大なご指導を賜った, 石水隆講師に深く感謝いたします. また本研究での撮影方法や研究の手法をご教授していただいた越智准教授にも深く感謝いたします.

参考文献

[1] Lu Zaima, Algorithm of Basketball Posture Motion Feature Extraction Based on Image Processing Technology, IEEE Conference Proceedings, Vol.2022, No.FAIML, pp.219–223, 2022.

[2] Ji Rong, Research on Basketball Shooting Action Based on Image Feature Extraction and Machine Learning, IEEE Access, Vol.8, pp.138743–138751, 2020.

[3] Esp-v2, ”【MediaPipe】Holistic のランドマークの具体的な場所、座標取得・保存まとめ【Python】”, https://qiita.com/Esp-v2/items/9c671ad29a263ce3675e, 2022.

[4] 191-036-0222 大薗伶治,”動画解析によるバスケットボールのシュート動作とボールの軌跡計測手法”, 知能情報ソフトウェア研究室

[5] 佐々木剛,”フォーム解析やバスケロボに活かされる AI！ 最新技術とバスケットボールの融合”, https://news.yahoo.co.jp/articles/cc3396e29efa9ae058cd5c8ea5ba0bdee27f86e3

[6] 石崎 繁利, 松村 礼, 大熊 啓太, 大原 啓吾, 黒田 悠太, 西山 周大, 春名 桂, 北村 知徳, 山本 長紀, ”画像認識技術を用いたバスケットボールのための表示装置の提案”, 日本機械学会 関西支部第 94 期定時総会講演会, セッション ID: 810, DOI: 10.1299/jsmekansai.2019.94.810, 2019 年 3 月 11 日–3 月 12 日, 主催: 一般社団法人 日本機械学会.

[7] 石垣　翔太, 安倍恵一, ”2 次元骨格情報を用いたバスケットシュートの姿勢解析評価”, 2022 年 3 月 3 日, 情報処理学会 84 回全国大会

[8] 浦西　友樹,”Opencv と Python による機械学習プログラミング, 樹,” Opencv と Python による機械学習プログラミング” 日本バーチャルリアリティ学会誌,vol.27,pp41-pp42 (2022)

[9] 松原 孝, 猪木原孝二, 川上雅之, 浮田 剛, 荒木郁夫, 荒木直彦,”バスケットボールゲームのシュートについてシュート角度による考察” 岡山理科大学紀要,vol. 26,pp379 pp380(1990)

[10] 西松　均, 久保山 潤一, 山内 俊明, 関 靖夫, ”バスケットボール試合におけるボール認識法の検討”, FIT2004（第 3 回情報科学技術フォーラム）

[11] OpenCV, ”Open Source Computer Vision Library”, https://opencv.org

[12] Google, ”MediaPipe: Cross-platform framework for building multimodal applied machine learning pipelines”, https://mediapipe.dev

[13] Wolfram MathWorld, ”The Web’s Most Extensive Mathematics Resource”, https://mathworld.wolfram.com

[14] Hunter, J. D., ”Matplotlib: A 2D Graphics Environment”, Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007, https://matplotlib.org.

付録 A　ソースプログラム

以下に本研究で作成したプログラムのソースファイルを示す. また静止画像の姿勢推定を行ったプログラムは 3 章で示したため省略する.

sub.py

ディレクトリーを作成するコード

```
import os

# 作成したディレクトリ内容
root_dir = "/Users/ozumi/Documents/workspace/卒論 Project"
#ディレクトリー構造
directories=[
    "data/zumi/videos",#初心者の動画,
    "data/zumi/frame",#初心者のフレーム画像
    "data/zumi/images",#初心者の静止画像
    "data/zumi/z-result",#初心者の姿勢測定の結果
    "data/kana/videos",#経験者の動画
    "data/kana/frames",#経験者のフレーム画像
    "data/kana/images",#経験者の静止画像
    "data/kana/k-result",#経験者の姿勢測定結果
    "code",
    "main-result"
]
for directory in directories:
    path = os.path.join(root_dir, directory)
    try:
        os.makedirs(path, exist_ok=True)
        print(f"ディレクトリを作成: {path}")
    except Exception as e:
        print(f"エラーが発生: {e}")

print("すべてのディレクトリが作成されました！")
```

toFrame.py

動画からフレームを抽出して保存するコード

```
1  import os
2  import cv2
3
4  def extract_frames(video_folder,output_folder,frame_rate=1):
5      #得た動画をフレームに変換し，それぞれのフレームを入れるディレクトリーに入れる
6      #video_folder: 動画が保存されているフォルダー
7      #output_folder: 変換したフレームを保存するフォルダー
8      #frame_rate: フレーム抽出間隔
9      os.makedirs(output_folder,exist_ok=True)
10
11     for video_file in os.listdir(video_folder):
12         video_path=os.path.join(video_folder,video_file)
13         if not video_file.endswith((".mp4",".avi",".mov")):
14             print(f"スキップ:{video_file}は動画ファイルではありません")
15             continue
16         cap = cv2.VideoCapture(video_path)
17         if not cap.isOpened():
18             print(f"動画を開けない:{video_path}")
19             continue
20         fps = int(cap.get(cv2.CAP_PROP_FPS))
21         interval = int(fps * frame_rate)
22         frame_count=0
23         success,frame=cap.read()
24
25         print(f"処理中:{video_path}")
26         while success:#ループ開始
27             if frame_count%interval ==0:
28                 frame_filename=os.path.join(output_folder,f"{os.path.splitext(video_file)[0]}_fra
29                 cv2.imwrite(frame_filename,frame)
30                 print(f"保存:{frame_filename}")
31             success,frame=cap.read()#フレームの取得
32             frame_count += 1
33
34         cap.release()
35         print(f"フレーム抽出完了:{video_file}")
36        #メイン処理
37 if __name__=="__main__":
```

```
38      root_dir="/Users/ozumi/Documents/workspace/卒論 Project"
39
40      zumi_videos_folder=os.path.join(root_dir,"data/zumi/videos")#初心者の動画
41      zumi_frames_folder=os.path.join(root_dir,"data/zumi/frames")#初心者のフレーム画像
42      kana_videos_folder=os.path.join(root_dir,"data/kana/videos")#経験者の動画
43      kana_frames_folder=os.path.join(root_dir,"data/kana/frames")#経験者のフレーム画像
44      #初心者の動画をフレームに変換
45      extract_frames(zumi_videos_folder,zumi_frames_folder,frame_rate=1)
46      #経験者の動画をフレームに変換
47      extract_frames(kana_videos_folder,kana_frames_folder,frame_rate=1)
48
```

es-frame.py

フレームの姿勢推定を行うコード

```
1   import os
2   import cv2
3   import mediapipe as mp
4   import pandas as pd
5
6   # MediaPipe の準備
7   mp_pose = mp.solutions.pose
8   pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5)
9   mp_drawing = mp.solutions.drawing_utils
10
11  # フレーム画像の姿勢推定を行い結果を保存する関数
12  def process_frames(frames_dir, result_dir):
13      # 結果ディレクトリを作成
14      os.makedirs(result_dir, exist_ok=True)
15      print(f"結果を保存するディレクトリ: {result_dir}")
16
17      # フレーム画像ファイルを取得
18      frames = [f for f in os.listdir(frames_dir) if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
19
20      for idx, frame_name in enumerate(frames, 1):
21          frame_path = os.path.join(frames_dir, frame_name)
22          img = cv2.imread(frame_path)
23      #画像読み込めなかった場合にエラー文を出す
24          if img is None:
```

```
25              print(f"画像を読み込めませんでした: {frame_path}")
26              continue
27
28          # BGR 画像を RGB に変換
29          img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
30          results = pose.process(img_rgb)
31
32          # 姿勢データを保存するリスト
33          pose_data = []
34
35          # 推定結果からランドマークデータを取得
36          if results.pose_landmarks:
37              for idx_landmark, landmark in enumerate(results.pose_landmarks.landmark):
38                  pose_data.append({
39                      "landmark": idx_landmark,#ランドマーク番号
40                      "x": landmark.x,#x 座標
41                      "y": landmark.y,#y 座標
42                      "z": landmark.z,#z 座標
43                      "visibility": landmark.visibility#認識精度
44                  })
45
46              # データフレームに変換
47              df = pd.DataFrame(pose_data)
48
49              # CSV ファイルとして保存
50              csv_filename = os.path.splitext(frame_name)[0] + ".csv"
51              csv_path = os.path.join(result_dir, csv_filename)
52              df.to_csv(csv_path, index=False)
53              print(f"[{idx}/{len(frames)}] 姿勢データを保存しました: {csv_path}")
54
55          else:
56              print(f"姿勢データが取得できませんでした: {frame_path}")
57
58      print(f"すべてのフレーム画像の処理が完了しました。\n")
59
60  # ディレクトリのパス
61  zumi_frames_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/zumi/frames"
62  zumi_result_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/zumi/z-result"
63
64  kana_frames_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/kana/frames"
```

```
kana_result_dir = "/Users/ozumi/Documents/workspace/卒論 Project/data/kana/k-result"

# 初心者と経験者のフレーム画像を処理
process_frames(zumi_frames_dir, zumi_result_dir)
process_frames(kana_frames_dir, kana_result_dir)

# リソース解放
pose.close()
```

left-e.py

左肘の角度を計算し散布図を作成するコード

```
import pandas as pd
import math
import matplotlib.pyplot as plt
import os

# 角度計算の関数
def calculate_angle(p1, p2, p3):
    """
    3 点の位置から角度を計算
    p1, p2, p3: (x, y) の座標
    """
    v1 = (p1[0] - p2[0], p1[1] - p2[1])  # ベクトル 1
    v2 = (p3[0] - p2[0], p3[1] - p2[1])  # ベクトル 2

    dot_product = v1[0] * v2[0] + v1[1] * v2[1]  # 内積
    magnitude_v1 = math.sqrt(v1[0]**2 + v1[1]**2)  # ベクトル 1 の大きさ
    magnitude_v2 = math.sqrt(v2[0]**2 + v2[1]**2)  # ベクトル 2 の大きさ

    if magnitude_v1 == 0 or magnitude_v2 == 0:
        return 0  # ゼロ除算を防ぐ

    cos_theta = dot_product / (magnitude_v1 * magnitude_v2)  # コサイン値
    angle = math.degrees(math.acos(max(min(cos_theta, 1.0), -1.0)))  # 角度に変換

    return angle

# k-result と z-result の CSV ファイルパスを取得
```

```
root_dir = "/Users/ozumi/Documents/workspace/卒論 Project"
csv_file_paths = []
labels = []  # ラベルリスト（色分け用）

# k-result と z-result ディレクトリ内の CSV ファイルを取得
for result_dir in ['data/zumi/z-result', 'data/kana/k-result']:
    for filename in os.listdir(os.path.join(root_dir, result_dir)):
        if filename.endswith(".csv"):
            csv_file_paths.append(os.path.join(root_dir, result_dir, filename))
            labels.append(result_dir)  # ディレクトリ名をラベルとして追加

# 左肘の角度を保存するリスト
angles_left_elbow= []

specific_color_map = {
    'data/zumi/z-result': 'blue',  # ディレクトリごとの色（青）初心者
    'data/kana/k-result': 'red',   # ディレクトリごとの色（赤）経験者
}
specific_csvs = ['zf5-g.csv', 'zf18-g.csv', 'zf19-g.csv', 'zumi5-g.csv','zumi7-g.csv','zumi-g.csv
心者のシュート成功の csv
specific_color = 'green'  # 特定の色（ここでは緑色）

specific2_csvs = ['k1.csv','k2.csv','k3.csv','k5.csv','kana9.csv','kana12.csv','kana16.csv','kana
                  ,'kana16_f0.csv','kana18_f0.csv','kana18_f29.csv','kana19_f0.csv','kana20_f0.cs
験者のシュート成功の csv
specific2_color = 'purple'  # 特定の色（ここでは紫色）



# 各 CSV ファイルを処理
for csv_file_path, label in zip(csv_file_paths, labels):
    try:
        # CSV を読み込む
        df = pd.read_csv(csv_file_path)

        # 肩、肘、手首の位置を抽出
        left_shoulder = (df.loc[df['landmark'] == 11, 'x'].values[0],
                         df.loc[df['landmark'] == 11, 'y'].values[0])
        left_elbow = (df.loc[df['landmark'] == 13, 'x'].values[0],
                      df.loc[df['landmark'] == 13, 'y'].values[0])
```

```
66          left_wrist = (df.loc[df['landmark'] == 15, 'x'].values[0],
67                        df.loc[df['landmark'] == 15, 'y'].values[0])
68
69          # 左肘角度の計算
70          angle = calculate_angle(left_wrist,left_elbow,left_shoulder)
71          angles_left_elbow.append(angle)
72
73          print(f"{csv_file_path} の肩、肘、手首の角度: {angle}度")
74
75      except Exception as e:
76          print(f"{csv_file_path} の処理中にエラーが発生しました: {e}")
77
78  # 色分けのためのカラーマップ
79  color_map = {'data/zumi/z-result': 'blue', 'data/kana/k-result': 'red'}
80
81  # グラフ化（左肘の角度をプロット）
82  for idx, (left_elbow_angle, csv_file_path) in enumerate(zip(angles_left_elbow, csv_file_paths)):
83      # `data/zumi/z-result` 内の特定の CSV には緑色を割り当て
84      if os.path.basename(csv_file_path) in specific_csvs:
85          file_color = specific_color  # 緑色
86      # `data/kana/k-result` 内の特定の CSV には紫色を割り当て
87      elif os.path.basename(csv_file_path) in specific2_csvs:
88          file_color = specific2_color  # 紫色
89      else:
90          file_color = specific_color_map.get(labels[idx], 'red')  # デフォルトは赤色
91
92      plt.plot(idx, left_elbow_angle, marker='o', color=file_color)
93  #ラベルの設定
94  plt.plot([], [], 'o', color='red', label='kana-false')#経験者のシュート失敗
95  plt.plot([], [], 'o', color='blue', label='zumi-false')#初心者のシュート失敗
96  plt.plot([], [], 'o', color='green', label='zumi-success')#初心者のシュート成功
97  plt.plot([], [], 'o', color='purple', label='kana-success')#経験者のシュート成功
98  plt.title("Elbow(Left)")
99  plt.xlabel("Image Index")
100 plt.ylabel("Angle (degrees)")
101 # 目盛りのフォントサイズを変更
102 plt.legend()
103 plt.tick_params(axis='x', labelsize=14)  # x 軸の目盛りサイズを 14 に設定
104 plt.tick_params(axis='y', labelsize=14)  # y 軸の目盛りサイズを 14 に設定
105 plt.grid(True)
```

```
106  plt.show()
107
```

left-h.py

左手首の角度を計算し散布図を作成するコード

```
1   import pandas as pd
2   import math
3   import matplotlib.pyplot as plt
4   import os
5
6   # 角度計算の関数
7   def calculate_angle(p1, p2, p3):
8       v1 = (p1[0] - p2[0], p1[1] - p2[1])  # ベクトル1
9       v2 = (p3[0] - p2[0], p3[1] - p2[1])  # ベクトル2
10
11      dot_product = v1[0] * v2[0] + v1[1] * v2[1]  # 内積
12      magnitude_v1 = math.sqrt(v1[0]**2 + v1[1]**2)  # ベクトル1の大きさ
13      magnitude_v2 = math.sqrt(v2[0]**2 + v2[1]**2)  # ベクトル2の大きさ
14
15      if magnitude_v1 == 0 or magnitude_v2 == 0:
16          return 0  # ゼロ除算を防ぐ
17
18      cos_theta = dot_product / (magnitude_v1 * magnitude_v2)  # コサイン値
19      angle = math.degrees(math.acos(max(min(cos_theta, 1.0), -1.0)))  # 角度に変換
20
21      return angle
22
23  # k-resultとz-resultのCSVファイルパスを取得
24  root_dir = "/Users/ozumi/Documents/workspace/卒論Project"
25  csv_file_paths = []
26  labels = []  # ラベルリスト（色分け用）
27
28  # k-resultとz-resultディレクトリ内のCSVファイルを取得
29  for result_dir in ['data/zumi/z-result', 'data/kana/k-result']:
30      for filename in os.listdir(os.path.join(root_dir, result_dir)):
31          if filename.endswith(".csv"):
32              csv_file_paths.append(os.path.join(root_dir, result_dir, filename))
33              labels.append(result_dir)  # ディレクトリ名をラベルとして追加
```

```
34
35  # 左手首角度を保存するリスト
36  angles_left_wrist = []
37
38  # ‘data/zumi/z-result‘ 内の 4 つの CSV ファイルに対して特定の色を設定
39  specific_color_map = {
40      'data/zumi/z-result': 'blue',  # ディレクトリごとの色（青）初心者
41      'data/kana/k-result': 'red',   # ディレクトリごとの色（赤）経験者
42  }
43
44  specific_csvs = ['zf5-g.csv', 'zf18-g.csv', 'zf19-g.csv', 'zumi5-g.csv','zumi7-g.csv','zumi-g.csv
  心者のシュート成功の csv
45  specific_color = 'green'  # 特定の色（ここでは緑色）
46
47  specific2_csvs = ['kana1.csv','kana2.csv','kana3.csv','kana5.csv','kana9.csv','kana12.csv','kana1
48                    ,'kana16_f0.csv','kana18_f0.csv','kana18_f29.csv','kana19_f0.csv','kana20_f0.cs
  験者のシュート成功の csv
49  specific2_color = 'purple'  # 特定の色（ここでは紫色）
50
51  # 各 CSV ファイルを処理
52  for csv_file_path, label in zip(csv_file_paths, labels):
53      try:
54          # CSV を読み込む
55          df = pd.read_csv(csv_file_path)
56
57          # 左肩、左肘、左手首の位置を抽出
58          left_shoulder = (df.loc[df['landmark'] == 11, 'x'].values[0],
59                           df.loc[df['landmark'] == 11, 'y'].values[0])
60          left_elbow = (df.loc[df['landmark'] == 13, 'x'].values[0],
61                        df.loc[df['landmark'] == 13, 'y'].values[0])
62          left_wrist = (df.loc[df['landmark'] == 15, 'x'].values[0],
63                        df.loc[df['landmark'] == 15, 'y'].values[0])
64
65          # 左手首の角度の計算
66          left_wrist_angle = calculate_angle(left_shoulder,left_wrist,left_elbow)
67          angles_left_wrist.append(left_wrist_angle)
68
69          print(f"{csv_file_path} の左肩、肘、手首の角度: {left_wrist_angle}度")
70
71      except Exception as e:
```

```
72          print(f"{csv_file_path} の処理中にエラーが発生しました: {e}")
73
74  # グラフ化（左手首の角度をプロット）
75  for idx, (left_wrist_angle, csv_file_path) in enumerate(zip(angles_left_wrist, csv_file_paths)):
76      # ‘data/zumi/z-result‘ 内の特定の CSV には緑色を割り当て
77      if os.path.basename(csv_file_path) in specific_csvs:
78          file_color = specific_color  # 緑色
79      # ‘data/kana/k-result‘ 内の特定の CSV には紫色を割り当て
80      elif os.path.basename(csv_file_path) in specific2_csvs:
81          file_color = specific2_color  # 紫色
82      else:
83          file_color = specific_color_map.get(labels[idx], 'red')  # デフォルトは赤色
84
85      plt.plot(idx, left_wrist_angle, marker='o', color=file_color)
86
87  # ラベルの設定
88  plt.plot([], [], 'o', color='red', label='kana-false')#経験者のシュート失敗
89  plt.plot([], [], 'o', color='blue', label='zumi-false')##初心者のシュート失敗
90  plt.plot([], [], 'o', color='green', label='zumi-success')#初心者のシュート成功
91  plt.plot([], [], 'o', color='purple', label='kana-success')#経験者のシュート成功
92
93  plt.title("Wrist(Left)")
94  plt.xlabel("Image Index")
95  plt.ylabel("Angle (degrees)")
96  plt.legend()
97  plt.tick_params(axis='x', labelsize=14)  # x 軸の目盛りサイズを 14 に設定
98  plt.tick_params(axis='y', labelsize=14)
99  plt.grid(True)
100 plt.show()
101
```

left-s.py

左肩の角度を計算し散布図を作成するコード

```
1  import pandas as pd
2  import math
3  import matplotlib.pyplot as plt
4  import os
5
```

```
# 角度計算の関数
def calculate_angle(p1, p2, p3):
    """
    3 点の位置から角度を計算
    p1, p2, p3: (x, y) の座標
    """
    v1 = (p1[0] - p2[0], p1[1] - p2[1])  # ベクトル 1
    v2 = (p3[0] - p2[0], p3[1] - p2[1])  # ベクトル 2

    dot_product = v1[0] * v2[0] + v1[1] * v2[1]  # 内積
    magnitude_v1 = math.sqrt(v1[0]**2 + v1[1]**2)  # ベクトル 1 の大きさ
    magnitude_v2 = math.sqrt(v2[0]**2 + v2[1]**2)  # ベクトル 2 の大きさ

    if magnitude_v1 == 0 or magnitude_v2 == 0:
        return 0  # ゼロ除算を防ぐ

    cos_theta = dot_product / (magnitude_v1 * magnitude_v2)  # コサイン値
    angle = math.degrees(math.acos(max(min(cos_theta, 1.0), -1.0)))  # 角度に変換

    return angle

# k-result と z-result の CSV ファイルパスを取得
root_dir = "/Users/ozumi/Documents/workspace/卒論 Project"
csv_file_paths = []
labels = []  # ラベルリスト（色分け用）

# k-result と z-result ディレクトリ内の CSV ファイルを取得
for result_dir in ['data/zumi/z-result', 'data/kana/k-result']:
    for filename in os.listdir(os.path.join(root_dir, result_dir)):
        if filename.endswith(".csv"):
            csv_file_paths.append(os.path.join(root_dir, result_dir, filename))
            labels.append(result_dir)  # ディレクトリ名をラベルとして追加

# 左肩の角度を保存するリスト
angles_left_sholder = []
# ‘data/zumi/z-result‘ 内の 4 つの CSV ファイルに対して特定の色を設定
specific_color_map = {
    'data/zumi/z-result': 'blue',  # ディレクトリごとの色（青）#初心者
    'data/kana/k-result': 'red',   # ディレクトリごとの色（赤）#経験者
}
```

```
46  specific_csvs = ['zf5-g.csv', 'zf18-g.csv', 'zf19-g.csv', 'zumi5-g.csv','zumi7-g.csv','zumi-g.csv
    心者のシュート成功の csv
47  specific_color = 'green'  # 特定の色（ここでは緑色）
48
49  specific2_csvs = ['kana1.csv','kana2.csv','kana3.csv','kana5.csv','kana9.csv','kana12.csv','kana1
50                    ,'kana16_f0.csv','kana18_f0.csv','kana18_f29.csv','kana19_f0.csv','kana20_f0.cs
    験者のシュート成功の csv
51  specific2_color = 'purple'  # 特定の色（ここでは紫色）
52
53
54
55  # 各 CSV ファイルを処理
56  for csv_file_path, label in zip(csv_file_paths, labels):
57      try:
58          # CSV を読み込む
59          df = pd.read_csv(csv_file_path)
60
61          # 左肩、左肘、左手首の位置を抽出
62          left_shoulder = (df.loc[df['landmark'] == 11, 'x'].values[0],
63                           df.loc[df['landmark'] == 11, 'y'].values[0])
64          left_elbow = (df.loc[df['landmark'] == 13, 'x'].values[0],
65                        df.loc[df['landmark'] == 13, 'y'].values[0])
66          left_wrist = (df.loc[df['landmark'] == 15, 'x'].values[0],
67                        df.loc[df['landmark'] == 15, 'y'].values[0])
68
69          # 左肩の角度の計算
70          left_sholder_angle = calculate_angle(left_elbow, left_shoulder,left_wrist,)
71          angles_left_sholder.append(left_sholder_angle)
72
73          print(f"{csv_file_path} の左肩、肘、手首の角度: {left_sholder_angle}度")
74
75      except Exception as e:
76          print(f"{csv_file_path} の処理中にエラーが発生しました: {e}")
77
78  # グラフ化（左肩の角度をプロット）
79  for idx, (left_sholder_angle, csv_file_path) in enumerate(zip(angles_left_sholder, csv_file_paths
80      # `data/zumi/z-result` 内の特定の CSV には緑色を割り当て
81      if os.path.basename(csv_file_path) in specific_csvs:
82          file_color = specific_color  # 緑色
83      # `data/kana/k-result` 内の特定の CSV には紫色を割り当て
```

```
84      elif os.path.basename(csv_file_path) in specific2_csvs:
85          file_color = specific2_color  # 紫色
86      else:
87          file_color = specific_color_map.get(labels[idx], 'red')  # デフォルトは赤色
88
89      plt.plot(idx, left_sholder_angle, marker='o', color=file_color)
90  #ラベルの設定
91  plt.plot([], [], 'o', color='red', label='kana-false')#経験者のシュート失敗
92  plt.plot([], [], 'o', color='blue', label='zumi-false')#初心者のシュート失敗
93  plt.plot([], [], 'o', color='green', label='zumi-success')#初心者のシュート成功
94  plt.plot([], [], 'o', color='purple', label='kana-success')#経験者のシュート成功
95
96  plt.title("Sholder(Left)")
97  plt.xlabel("Image Index")
98  plt.ylabel("Angle (degrees)")
99  plt.legend()
100 plt.tick_params(axis='x', labelsize=14)  # x 軸の目盛りサイズを 14 に設定
101 plt.tick_params(axis='y', labelsize=14)
102 plt.grid(True)
103 plt.show()
104
```

right-e.py

右肘の角度を計算し散布図を作成するコード

```
1   import pandas as pd
2   import math
3   import matplotlib.pyplot as plt
4   import os
5
6   # 角度計算の関数
7   def calculate_angle(p1, p2, p3):
8       v1 = (p1[0] - p2[0], p1[1] - p2[1])  # ベクトル 1
9       v2 = (p3[0] - p2[0], p3[1] - p2[1])  # ベクトル 2
10
11      dot_product = v1[0] * v2[0] + v1[1] * v2[1]  # 内積
12      magnitude_v1 = math.sqrt(v1[0]**2 + v1[1]**2)  # ベクトル 1 の大きさ
13      magnitude_v2 = math.sqrt(v2[0]**2 + v2[1]**2)  # ベクトル 2 の大きさ
14
```

```
15      if magnitude_v1 == 0 or magnitude_v2 == 0:
16          return 0  # ゼロ除算を防ぐ
17
18      cos_theta = dot_product / (magnitude_v1 * magnitude_v2)  # コサイン値
19      angle = math.degrees(math.acos(max(min(cos_theta, 1.0), -1.0)))  # 角度に変換
20
21      return angle
22
23  # k-result と z-result の CSV ファイルパスを取得
24  root_dir = "/Users/ozumi/Documents/workspace/卒論 Project"
25  csv_file_paths = []
26  labels = []  # ラベルリスト（色分け用）
27
28  # k-result と z-result ディレクトリ内の CSV ファイルを取得
29  for result_dir in ['data/zumi/z-result', 'data/kana/k-result']:
30      for filename in os.listdir(os.path.join(root_dir, result_dir)):
31          if filename.endswith(".csv"):
32              csv_file_paths.append(os.path.join(root_dir, result_dir, filename))
33              labels.append(result_dir)  # ディレクトリ名をラベルとして追加
34
35  # 右肘の角度を保存するリスト
36  angles_right_elbow = []
37
38  # `data/zumi/z-result` 内の 4 つの CSV ファイルに対して特定の色を設定
39  specific_color_map = {
40      'data/zumi/z-result': 'blue',  # ディレクトリごとの色（青）#初心者
41      'data/kana/k-result': 'red',   # ディレクトリごとの色（赤）#経験者
42  }
43
44  specific_csvs = ['zf5-g.csv', 'zf18-g.csv', 'zf19-g.csv', 'zumi5-g.csv','zumi7-g.csv','zumi-g.csv
    心者のシュート成功の csv
45  specific_color = 'green'  # 特定の色（ここでは緑色）
46
47  specific2_csvs = ['kana1.csv','kana2.csv','kana3.csv','kana5.csv','kana9.csv','kana12.csv','kana1
48                    ,'kana16_f0.csv','kana18_f0.csv','kana18_f29.csv','kana19_f0.csv','kana20_f0.cs
    験者のシュート成功の csv
49  specific2_color = 'purple'  # 特定の色（ここでは紫色）
50
51  # 各 CSV ファイルを処理
52  for csv_file_path, label in zip(csv_file_paths, labels):
```

```
53      try:
54          # CSV を読み込む
55          df = pd.read_csv(csv_file_path)
56
57          # 右肩、右肘、右手首の位置を抽出
58          right_shoulder = (df.loc[df['landmark'] == 12, 'x'].values[0],
59                            df.loc[df['landmark'] == 12, 'y'].values[0])
60          right_elbow = (df.loc[df['landmark'] == 14, 'x'].values[0],
61                         df.loc[df['landmark'] == 14, 'y'].values[0])
62          right_wrist = (df.loc[df['landmark'] == 16, 'x'].values[0],
63                         df.loc[df['landmark'] == 16, 'y'].values[0])
64
65          # 右肘の角度の計算
66          right_elbow_angle = calculate_angle(right_shoulder, right_elbow, right_wrist)
67          angles_right_elbow.append(right_elbow_angle)
68
69          print(f"{csv_file_path} の右肩、肘、手首の角度: {right_elbow_angle}度")
70
71      except Exception as e:
72          print(f"{csv_file_path} の処理中にエラーが発生しました: {e}")
73
74  # グラフ化（右肘の角度をプロット）
75  for idx, (right_elbow_angle, csv_file_path) in enumerate(zip(angles_right_elbow, csv_file_paths)):
76      # ‘data/zumi/z-result‘ 内の特定の CSV には緑色を割り当て
77      if os.path.basename(csv_file_path) in specific_csvs:
78          file_color = specific_color  # 緑色
79      # ‘data/kana/k-result‘ 内の特定の CSV には紫色を割り当て
80      elif os.path.basename(csv_file_path) in specific2_csvs:
81          file_color = specific2_color  # 紫色
82      else:
83          file_color = specific_color_map.get(labels[idx], 'red')  # デフォルトは赤色
84
85      plt.plot(idx, right_elbow_angle, marker='s', color=file_color)
86
87  #ラベルの設定
88  plt.plot([], [], 's', color='red', label='kana-false')#経験者のシュート失敗
89  plt.plot([], [], 's', color='blue', label='zumi-false')#初心者のシュート失敗
90  plt.plot([], [], 's', color='green', label='zumi-success')#初心者のシュート成功
91  plt.plot([], [], 's', color='purple', label='kana-success')#経験者のシュート成功
92  plt.title("Elbow (right)")
```

```
93  plt.xlabel("Image Index")
94  plt.ylabel("Angle (degrees)")
95  plt.legend()  # ラベルの表示
96  plt.grid(True)
97  plt.show()
98
```

right-h.py

右手首の角度を計算し散布図を作成するコード

```
1   import pandas as pd
2   import math
3   import matplotlib.pyplot as plt
4   import os
5
6   # 角度計算の関数
7   def calculate_angle(p1, p2, p3):
8       v1 = (p1[0] - p2[0], p1[1] - p2[1])  # ベクトル1
9       v2 = (p3[0] - p2[0], p3[1] - p2[1])  # ベクトル2
10
11      dot_product = v1[0] * v2[0] + v1[1] * v2[1]  # 内積
12      magnitude_v1 = math.sqrt(v1[0]**2 + v1[1]**2)  # ベクトル1の大きさ
13      magnitude_v2 = math.sqrt(v2[0]**2 + v2[1]**2)  # ベクトル2の大きさ
14
15      if magnitude_v1 == 0 or magnitude_v2 == 0:
16          return 0  # ゼロ除算を防ぐ
17
18      cos_theta = dot_product / (magnitude_v1 * magnitude_v2)  # コサイン値
19      angle = math.degrees(math.acos(max(min(cos_theta, 1.0), -1.0)))  # 角度に変換
20
21      return angle
22
23  # k-resultとz-resultのCSVファイルパスを取得
24  root_dir = "/Users/ozumi/Documents/workspace/卒論Project"
25  csv_file_paths = []
26  labels = []  # ラベルリスト（色分け用）
27
28  # k-resultとz-resultディレクトリ内のCSVファイルを取得
29  for result_dir in ['data/zumi/z-result', 'data/kana/k-result']:
```

```
30      for filename in os.listdir(os.path.join(root_dir, result_dir)):
31          if filename.endswith(".csv"):
32              csv_file_paths.append(os.path.join(root_dir, result_dir, filename))
33              labels.append(result_dir)  # ディレクトリ名をラベルとして追加
34
35  # 右手首の角度を保存するリスト
36  angles_right_wrist= []
37
38  # ‘data/zumi/z-result‘ 内の 4 つの CSV ファイルに対して特定の色を設定
39  specific_color_map = {
40      'data/zumi/z-result': 'blue',  # ディレクトリごとの色（青）#初心者
41      'data/kana/k-result': 'red',   # ディレクトリごとの色（赤）#経験者
42  }
43
44  specific_csvs = ['zf5-g.csv', 'zf18-g.csv', 'zf19-g.csv', 'zumi5-g.csv','zumi7-g.csv','zumi-g.csv
    心者のシュート成功の csv
45  specific_color = 'green'  # 特定の色（ここでは緑色
46
47  specific2_csvs = ['kana1.csv','kana2.csv','kana3.csv','kana5.csv','kana9.csv','kana12.csv','kana1
48                    ,'kana16_f0.csv','kana18_f0.csv','kana18_f29.csv','kana19_f0.csv','kana20_f0.cs
    験者のシュート成功の csv
49  specific2_color = 'purple'  # 特定の色（ここでは紫色）
50
51  # 各 CSV ファイルを処理
52  for csv_file_path, label in zip(csv_file_paths, labels):
53      try:
54          # CSV を読み込む
55          df = pd.read_csv(csv_file_path)
56
57          # 右肩、右肘、右手首の位置を抽出
58          right_shoulder = (df.loc[df['landmark'] == 12, 'x'].values[0],
59                            df.loc[df['landmark'] == 12, 'y'].values[0])
60          right_elbow = (df.loc[df['landmark'] == 14, 'x'].values[0],
61                         df.loc[df['landmark'] == 14, 'y'].values[0])
62          right_wrist = (df.loc[df['landmark'] == 16, 'x'].values[0],
63                         df.loc[df['landmark'] == 16, 'y'].values[0])
64
65          # 右手首の角度の計算
66          right_wrist_angle = calculate_angle(right_shoulder,right_wrist,right_elbow)
67          angles_right_wrist.append(right_wrist_angle)
```

```
68
69          print(f"{csv_file_path} の右肩、肘、手首の角度: {right_wrist_angle}度")
70
71      except Exception as e:
72          print(f"{csv_file_path} の処理中にエラーが発生しました: {e}")
73
74  # グラフ化（右手首の角度をプロット）
75  for idx, (right_wrist_angle, csv_file_path) in enumerate(zip(angles_right_wrist, csv_file_paths)):
76      # ‘data/zumi/z-result‘ 内の特定の CSV には緑色を割り当て
77      if os.path.basename(csv_file_path) in specific_csvs:
78          file_color = specific_color  # 緑色
79      # ‘data/kana/k-result‘ 内の特定の CSV には紫色を割り当て
80      elif os.path.basename(csv_file_path) in specific2_csvs:
81          file_color = specific2_color  # 紫色
82      else:
83          file_color = specific_color_map.get(labels[idx], 'red')  # デフォルトは赤色
84
85      plt.plot(idx, right_wrist_angle, marker='s', color=file_color)
86
87  #ラベルの設定
88  plt.plot([], [], 's', color='red', label='kana-false')#経験者のシュート失敗
89  plt.plot([], [], 's', color='blue', label='zumi-false')#初心者のシュート失敗
90  plt.plot([], [], 's', color='green', label='zumi-success')#初心者のシュート成功
91  plt.plot([], [], 's', color='purple', label='kana-success')#経験者のシュート成功
92  plt.title("Wrist (right)")
93  plt.xlabel("Image Index")
94  plt.ylabel("Angle (degrees)")
95  plt.legend()  # ラベルの表示
96  plt.grid(True)
97  plt.show()
```

right-s.py

右肩の角度を計算し散布図を作成するコード

```
1  import pandas as pd
2  import math
3  import matplotlib.pyplot as plt
4  import os
5
```

```
6   # 角度計算の関数
7   def calculate_angle(p1, p2, p3):
8       v1 = (p1[0] - p2[0], p1[1] - p2[1])  # ベクトル 1
9       v2 = (p3[0] - p2[0], p3[1] - p2[1])  # ベクトル 2
10
11      dot_product = v1[0] * v2[0] + v1[1] * v2[1]  # 内積
12      magnitude_v1 = math.sqrt(v1[0]**2 + v1[1]**2)  # ベクトル 1 の大きさ
13      magnitude_v2 = math.sqrt(v2[0]**2 + v2[1]**2)  # ベクトル 2 の大きさ
14
15      if magnitude_v1 == 0 or magnitude_v2 == 0:
16          return 0  # ゼロ除算を防ぐ
17
18      cos_theta = dot_product / (magnitude_v1 * magnitude_v2)  # コサイン値
19      angle = math.degrees(math.acos(max(min(cos_theta, 1.0), -1.0)))  # 角度に変換
20
21      return angle
22
23  # k-result と z-result の CSV ファイルパスを取得
24  root_dir = "/Users/ozumi/Documents/workspace/卒論 Project"
25  csv_file_paths = []
26  labels = []  # ラベルリスト（色分け用）
27
28  # k-result と z-result ディレクトリ内の CSV ファイルを取得
29  for result_dir in ['data/zumi/z-result', 'data/kana/k-result']:
30      for filename in os.listdir(os.path.join(root_dir, result_dir)):
31          if filename.endswith(".csv"):
32              csv_file_paths.append(os.path.join(root_dir, result_dir, filename))
33              labels.append(result_dir)  # ディレクトリ名をラベルとして追加
34
35  # 右肩の角度を保存するリスト
36  angles_right_sholder = []
37
38  # ‘data/zumi/z-result‘ 内の 4 つの CSV ファイルに対して特定の色を設定
39  specific_color_map = {
40      'data/zumi/z-result': 'blue',  # ディレクトリごとの色（青）初心者
41      'data/kana/k-result': 'red',   # ディレクトリごとの色（赤）経験者
42  }
43
44  specific_csvs = ['zf5-g.csv', 'zf18-g.csv', 'zf19-g.csv', 'zumi5-g.csv','zumi7-g.csv','zumi-g.csv
```

心者のシュート成功の csv

```
specific_color = 'green'  # 特定の色（ここでは緑色）

specific2_csvs = ['kana1.csv','kana2.csv','kana3.csv','kana5.csv','kana9.csv','kana12.csv','kana1
                  ,'kana16_f0.csv','kana18_f0.csv','kana18_f29.csv','kana19_f0.csv','kana20_f0.cs
```
験者のシュート成功の csv
```
specific2_color = 'purple'  # 特定の色（ここでは紫色）

# 各 CSV ファイルを処理
for csv_file_path, label in zip(csv_file_paths, labels):
    try:
        # CSV を読み込む
        df = pd.read_csv(csv_file_path)

        # 右肩、右肘、右手首の位置を抽出
        right_shoulder = (df.loc[df['landmark'] == 12, 'x'].values[0],
                          df.loc[df['landmark'] == 12, 'y'].values[0])
        right_elbow = (df.loc[df['landmark'] == 14, 'x'].values[0],
                       df.loc[df['landmark'] == 14, 'y'].values[0])
        right_wrist = (df.loc[df['landmark'] == 16, 'x'].values[0],
                       df.loc[df['landmark'] == 16, 'y'].values[0])

        # 右肩の角度の計算
        right_sholder_angle = calculate_angle(right_elbow,right_shoulder,right_wrist)
        angles_right_sholder.append(right_sholder_angle)

        print(f"{csv_file_path} の右肩、肘、手首の角度: {right_sholder_angle}度")

    except Exception as e:
        print(f"{csv_file_path} の処理中にエラーが発生しました: {e}")

# グラフ化（右肩の角度をプロット）
for idx, (right_sholder_angle, csv_file_path) in enumerate(zip(angles_right_sholder, csv_file_pat
    # ‘data/zumi/z-result‘ 内の特定の CSV には緑色を割り当て
    if os.path.basename(csv_file_path) in specific_csvs:
        file_color = specific_color  # 緑色
    # ‘data/kana/k-result‘ 内の特定の CSV には紫色を割り当て
    elif os.path.basename(csv_file_path) in specific2_csvs:
        file_color = specific2_color  # 紫色
    else:
        file_color = specific_color_map.get(labels[idx], 'red')  # デフォルトは赤色
```

```
84
85      plt.plot(idx, right_sholder_angle, marker='s', color=file_color)
86
87  #ラベルの設定
88  plt.plot([], [],'s', color='red', label='kana-false')#経験者のシュート失敗
89  plt.plot([], [], 's', color='blue', label='zumi-false')#初心者のシュート失敗
90  plt.plot([], [], 's', color='green', label='zumi-success')#初心者のシュート成功
91  plt.plot([], [], 's', color='purple', label='kana-success')#経験者のシュート成功
92  plt.title("Sholder (right)")
93  plt.xlabel("Image Index")
94  plt.ylabel("Angle (degrees)")
95  plt.legend()  # ラベルの表示
96  plt.grid(True)
97  plt.show()
```