ウォーミングアップと運動パフォーマンスの相関

OpenCVによるトラッキングデータの活用

神奈川県立光陵高校2年
亀井　奏汰

1、背景と目的

部活動の準備運動は怪我を防ぐことやパフォーマンスの向上に役立つ事は知っているが、自分が現在テニス部でやっている準備運動の内容に疑問を持っている。本当に効果があるのかを知りたいと思った。部活の大会などにおいて最大限にパフォーマンスを引き出すために、自分自身がどの程度の準備運動が必要なのかデータを使って調べたい。
トレーニングマシンなどがない高校部活で、ランニングは最も取り入れやすいウォーミングアップの1つである事から、走ることに焦点をあてて研究し、運動系の部活動をする高校生に役立つことを目指したい。

2、先行研究からの仮説

「ウォーミングアップにおける神経伝達速度の変化」の研究では心拍数110〜120回 / 分の運動強度で15分程の走行が、神経伝達速度が上昇する事が示されている。
この研究を踏まえ、心拍数110〜120回 /分の運動強度で適度な距離を走れば運動能力が上がると仮説をたて、自身の走った量と運動能力の関係のデータをとる実験をする。

3、実験方法

1、運動能力の指標として全ての運動の基礎であるジャンプ力を採用する。
2、ジャンプの前に一定の走行トレーニング　(100mごと)をする。
3、走行トレーニングはスマートウオッチを装着し、距離を測りながらその場で走る。
心拍数を110-120でキープする。
4、最初にトレーニングなしでのジャンプを録画し、次に100mごとに同じ条件でジャンプを録画。それをオールアウト(　疲労 max)まで繰り返す。
5、ジャンプ力は垂直最高到達点、飛び出す時の速度を指標とする。
6、ジャンプ力の測定法として python の opencv ライブラリのHaar Cascade顔認証より録画の中の認証された領域の中心座標を追って計算する。
7、2 - 4 の行程を同じ条件で異なる日に3回繰り返す。

4、予備実験と改善点

```python
# Haar cascadesを使用して顔を検出する
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# 検出された顔のリストを、縦方向の位置でソートする
sorted_faces = sorted(faces, key=lambda face: face[1])

# より上にある顔を選択する
if len(sorted_faces) > 0:
    selected_face = sorted_faces[0]

    # 選択された顔を矩形で囲む
    x, y, w, h = selected_face
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

予備実験では洋服の陰影が顔として誤認され、
領域が複数になってしまう事があった。
→確実に顔を認証するように、より上にある領域を選択するコードに書き換え、黒い服を着て誤認を少なくする様にした。

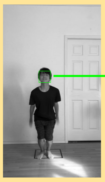
5、データの整理

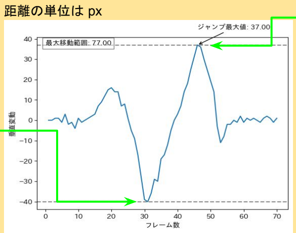
ジャンプの垂直変動と最高到達点を求める

1ジャンプの垂直変動をグラフ・数値化する

距離の単位は px

最低地点

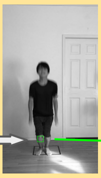


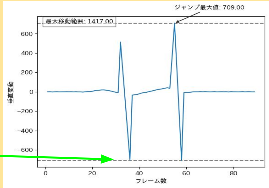


最高到達点

2正しく検出されなかったデータを省く

膝を
検出して
いる

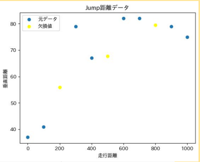




3正しくないデータの欠損を補完する

データに相関が見れたのでデータ補完は予測モデルを使う

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# 欠損値のないデータを抽出
non_missing_data = {
    'jump_numbers': [0, 100, 300, 400, 600, 700, 900, 1000],
    'jump_distances': [37, 41, 79, 67, 82, 82, 79, 75]
}

# ジャンプ回数とジャンプ距離を別々のリストに分割
jump_numbers = non_missing_data['jump_numbers']
jump_distances = non_missing_data['jump_distances']

# モデルの作成とトレーニング
model = LinearRegression()
model.fit([[num] for num in jump_numbers], jump_distances)

missing_jump_numbers = [200, 500, 800]  # 欠損している走行距離

# 欠損値の予測
predicted_jump_distances = model.predict([[num] for num in missing_jump_numbers])

# データフレーム作成
completed_data = pd.DataFrame({'Jump Number': jump_numbers + missing_jump_numbers,
                               'Jump Distance': jump_distances + list(predicted_jump_distances)})

# グラフ作成（散布図）
plt.scatter(jump_numbers, jump_distances, label='元データ')
plt.scatter(missing_jump_numbers, predicted_jump_distances, color='yellow', label='欠損値')
plt.ylabel('垂直距離')
plt.xlabel('走行距離')
plt.title('Jump距離データ')
plt.legend()
plt.show()

# 補完されたデータをデータフレームとして表示
print(completed_data)
```



	Jump Number	Jump Distance
0	0	37.000000
1	100	41.000000
2	300	79.000000
3	400	67.000000
4	600	82.000000
5	700	82.000000
6	900	79.000000
7	1000	75.000000
8	200	55.913043
9	500	67.750000
10	800	79.586957

正しいデータが得られなかった1回目の200/500/800mと3回目の200/600mの欠損を上記の方法によって補完

ジャンプの速度を求める

1ジャンプの瞬間速度を前進差分で求める

連続したデータの瞬間の速度を求める場合

$$v(\text{瞬間}) = \lim_{\Delta t \to 0} \frac{x(t+\Delta t)-x(t)}{\Delta t}$$

今回は30fpsで撮影なので1/30秒ごとのフレームで位置が変化していき、位置座標は不連続なデータである。ある時刻、i番目のデータを**x(i)**とすると次のデータは**x(i+1)**になるので、x(i)の時点の瞬間の速度は一つ先のデータの差分（前進差分）で求める。

不連続なデータx(i)の時点の速度を求める場合

$$v(i) = \frac{x(i+\Delta t)-x(i)}{\Delta t}$$

⇩

30fpsの動画でx(i)の時点の速度を求める場合

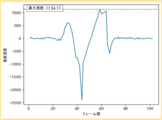
$$v(i) = \frac{x(i+1)-x(i)}{0.0333}$$

pythonの計算ライブラリnumpyを使い、配列内の前の位置からの変化と1フレーム値からjump動画の瞬間速度を計算

```python
# より上にある顔を選択する
if len(sorted_faces) > 0:
    selected_face = sorted_faces[0]

    # 選択された顔を矩形で囲む
    x, y, w, h = selected_face
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # 中心座標を計算する
    cx = x + w / 2
    cy = y + h / 2

    # 現在の時間を取得（1000で割ってミリ秒を秒に変換する）
    current_time = cap.get(cv2.CAP_PROP_POS_MSEC) / 1000.0

    # 現在の速度を計算する（前進差分）
    if prev_cy is not None and prev_time is not None:
        delta_cy = cy - prev_cy  # 前のフレームからの垂直位置の変化
        delta_time = current_time - prev_time  # 前のフレームからの時間の変化
        vy = delta_cy / delta_time # フレーム位置の変化を時間の変化で割り速度を出す
    else:
        vy = 0 # 初回フレームは前の位置データがないため計算しない
```

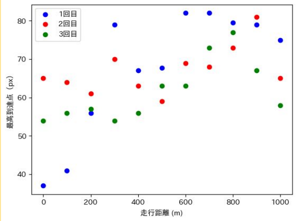
2瞬間速度の変化と最大値のデータを取得

45 f 辺りで沈み込む速度が最大値に、60 f で上向きの速度（ジャンプ速度）が最大値になっている事がわかる



6、実験結果

ジャンプの最高到達点と走行距離の関係

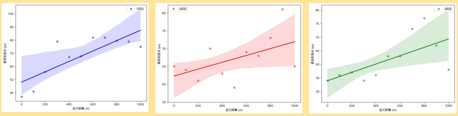


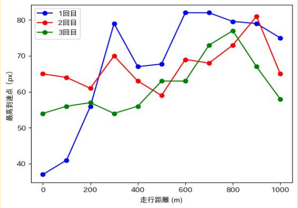
結果1

1回目〜3回目までの結果の散布図。
正の相関関係があると見られる。

結果2

各データの線形回帰と信頼区間。
後半に相関の信頼性が低い。

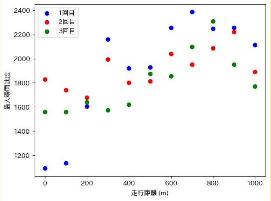




結果3

折れ線グラフにして変化を可視化。最高値の後は下がっていく事を確認。

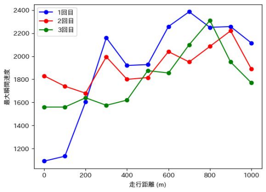
ジャンプの瞬間最高速度と走行距離の関係



結果4

結果1と同様に瞬間最高速度も正の相関が見られる。

結果5

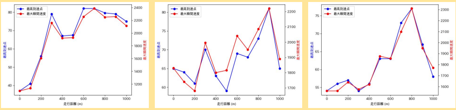


結果3と同様に瞬間最高速度も最高値の後は下がっていく。

最高到達点と最大瞬間速度の関係

結果6

この2つの関係も連動している。



7、考察・今後の課題

データより、ジャンプ力を最大限に引き出すためのウォーミングアップは心拍数を110-120 でキープし、800m程度の走行をするのが私自身では最適だと考えられる。
他の運動部員に当てはまるか、普段運動をしない人ではどうゆうデータが出るのか集めてみたい。

8、参考文献

ウォーミングアップにおける神経伝達速度の変化 内田大・岡本龍児　たかがジャンプされどジャンプ 抑木裕嗣