

令和6年度卒業研究報告書

座位姿勢をサポートするアプリの作成

情報技術科 浅沼颯斗 角地湧成 上村武之 高橋遼

指導教員 ソソラ

目次

第 1 章 研究概要.....	3
1.1 研究背景と目的.....	3
1.2 関連研究.....	4
第 2 章 アプリの概要.....	5
2.1 アプリの概要.....	5
2.2 開発環境.....	6
2.3 MediaPipe について.....	7
2.4 よい姿勢とは.....	12
第 3 章 姿勢検出について	13
3.1 角度の検出方法について.....	13
3.2 首の角度の検出方法	16
3.3 腰の角度の検出方法	18
第 4 章 姿勢の評価(点数化)について.....	20
4.1 首の角度の評価.....	20
4.2 腰の角度の評価.....	21
4.3 総合評価	24
4.4 姿勢をサポートするグッズにおける実験結果	26
第 5 章 実装したアプリについて.....	28
5.1 アプリの構成図について(システムの流れ)	29
5.2 アプリの機能について	30
5.3 画面遷移について	31
第 6 章 振り返り機能の実装について	35

6.1 特徴.....	35
6.2 実装の詳細.....	37
6.3 使用したライブラリについて.....	38
第 7 章 おわりに.....	41
参考文献.....	42
付録.....	43
付録 A 使用したライブラリ	43
付録 B ソースコード一覧.....	46
付録 C 開発環境セットアップマニュアル	54

第1章 研究概要

1.1 研究背景と目的

近年、デスクワークや座り仕事が増加しており、これに伴って腰痛や肩こりなどの症状に悩む人が増えている。長時間同じ姿勢でいることや不適切な姿勢が身体に負担をかけ、健康に悪影響を与えることが多い。この問題を解決するため、座った姿勢の改善をサポートするアプリケーションが求められている。

本研究では、授業で MediaPipe を活用した経験をもとに、姿勢改善に役立つシステムを開発することを目指した。授業の中で MediaPipe を使ってリアルタイムで身体の部位を検出する技術に触れ、その精度や適用範囲に感銘を受けた。そのため、この技術を活かして、座った姿勢を解析し、ユーザにフィードバックを提供するシステムを構築できると考えた。

本研究の目的は、座った姿勢をリアルタイムで解析し、ユーザに姿勢の改善を促すアプリケーションを開発することにある。授業で学んだ MediaPipe を活用し、ユーザの姿勢を正確に評価し、即時にフィードバックを提供することを目指す。さらに、このアプリケーションは、ユーザが姿勢改善をサポートするためのグッズ（例えば、姿勢矯正クッションやストレッチャーなど）を選択する際の手助けになることを期待している。ユーザがどのようなサポートが必要かを認識し、適切なグッズを選ぶための一つの指標として活用できるようにすることも、このアプリケーションの重要な目的の一つである。

1.2 関連研究

近年、デスクワークや座り仕事に従事するユーザの姿勢改善を目的とした研究が進められている。これらの研究では、センサや機械学習を活用してユーザの姿勢をリアルタイムで評価し、適切なフィードバックを提供することが試みられている。

センサを用いた姿勢評価 [1]: ウェアラブルデバイスやモーションキャプチャ技術を用いた姿勢評価が注目されている。例えば、圧力センサを内蔵したマットを用いて座位姿勢の左右の偏りを識別する研究がある。また、センサを内蔵した椅子を設計・開発し、オフィスワーカーの姿勢を業務の妨げなく継続的に測定するシステムも提案されている [2]。

機械学習を活用した姿勢推定 [3]: 機械学習を用いた姿勢分類・評価の研究も進められている。特に、Google の「PoseNet」はディープラーニングを活用し、人体のキーポイント（関節位置）を検出してリアルタイムの姿勢推定を可能にする技術として知られている。この技術はカメラのみで動作し、専用センサが不要という利点があるが、学習データによっては特定の姿勢に対する認識精度が低くなる課題も指摘されている。

そこで、本研究では MediaPipe を使用することに決めた。

第 2 章 アプリの概要

2.1 アプリの概要

Flask を用いて姿勢改善をサポートする web アプリを実装した(図 2.1 参照). ユーザを真横から撮影できる位置にカメラを設置する. 姿勢を測定開始する場合は、アプリの「測定開始」メニューを選択すると、MediaPipe により検出された首や腰などの位置データから、首や背筋が垂直に対してどれだけ傾いているかを計算し、その状態を数値化して画面上に表示される。ユーザは画面に映っている自分の姿勢を確認しながら、自分にとって最も良い姿勢を見つける。保存ボタンを押すと、現在の状態が csv ファイルとして保存される。また、アプリではいつどのような状態だったかを確認でき、過去 10 個分のデータを振り返ることが可能である。

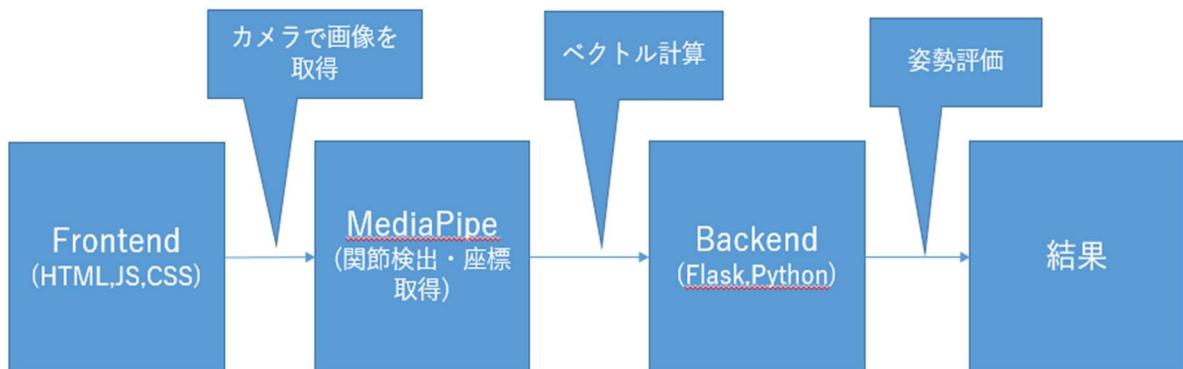


図 2.1 姿勢測定の流れ

2.2 開発環境

本研究では、座位姿勢改善アプリの開発にあたり、以下の環境を構築した（詳しく付録 A 参照）。

ハードウェア：

（1） PC : Windows 10

（2） カメラ：外付けの USB カメラ

ソフトウェア：

（3） 開発言語：python3.12.6 HTML JS CSS

（4） フレームワーク・ライブラリ

- MediaPipe（姿勢推定）
- OpenCV（画像処理）
- NumPy（データ処理）
- Flask（Web アプリケーション）
- Csv（データ保存・管理）
- Datetime（日付・時間管理）
- Tkinter（GUI 構築）

（5） 開発環境：VS Code

（6） 実行環境：ローカル PC

本システムは、Google の MediaPipe を用いて姿勢推定を行い、Web カメラの映像からユーザー

の座位姿勢をリアルタイムで解析する。MediaPipe は高精度かつ軽量なモデルを提供し、特別なハードウェアを必要とせずに動作するため、手軽に姿勢評価を行える点が利点である。また、取得したデータは csv 形式で保存し、datetime ライブラリを利用してデータの管理を行っている。さらに、Tkinter を使用して振り返り機能の GUI を構築し、ユーザが過去の姿勢データや撮影した画像を簡単に確認できるようにした。Flask を導入することで、Web インターフェースを活用したデータ表示や比較機能も実装しており、利便性の向上を図っている。

2.3 MediaPipe について

MediaPipe は、Google が開発したオープンソースのライブメディア処理ソリューション。特に、動画や画像処理を対象とした機械学習(ML)を効率的に実装するためのライブラリとして、多くの分野で活用されている。

以下の図に MediaPipe で取得できる位置情報(ランドマーク)を示す。

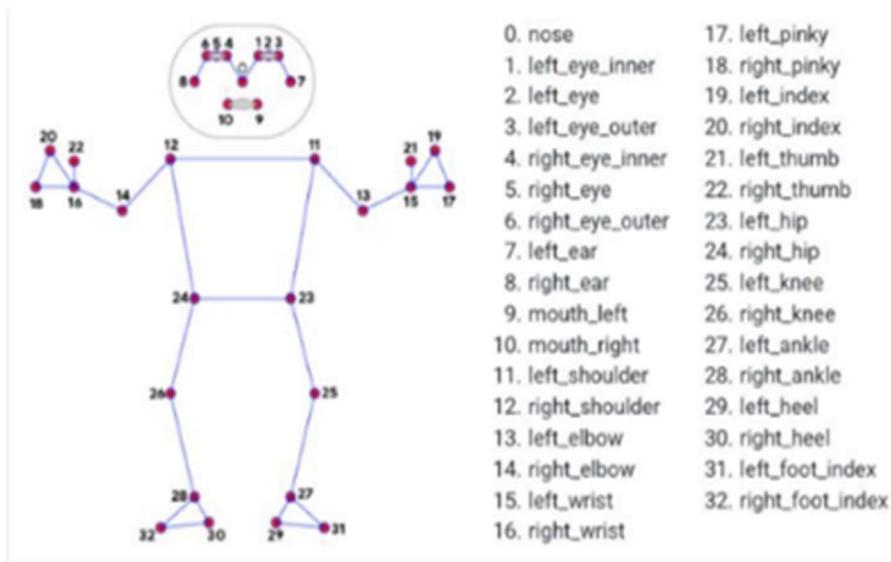


図2. MediaPipe のランドマーク

MediaPipe を利用した姿勢を検出するコード

以下に,MediaPipe を用いた姿勢検出のコードを示す.

このコードでは,OpenCV を用いてカメラ映像を取得し,MediaPipe の Pose モジュールを利用して姿勢を検出している.検出された関節情報はフレーム上に描画され,リアルタイムでの姿勢解析が可能となる.本研究では,リアルタイムでの姿勢解析を目的とし,計算負荷が軽く,導入が容易な MediaPipe を採用した. OpenPose と比較して,処理速度とリアルタイム性に優れており,ユーザの負担を少なくしながら姿勢改善をサポートできる点が利点である.

```
import cv2
import mediapipe as mp

mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
mp_drawing = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # 映像を左右反転
    frame = cv2.flip(frame, 1)

    # BGRをRGBに変換
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = pose.process(rgb_frame)

    if result.pose_landmarks:
        mp_drawing.draw_landmarks(frame, result.pose_landmarks, mp_pose.POSE_CONNECTIONS)

    cv2.imshow('Pose Detection (Mirrored)', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

図 2-1 ランドマーク検出のコード

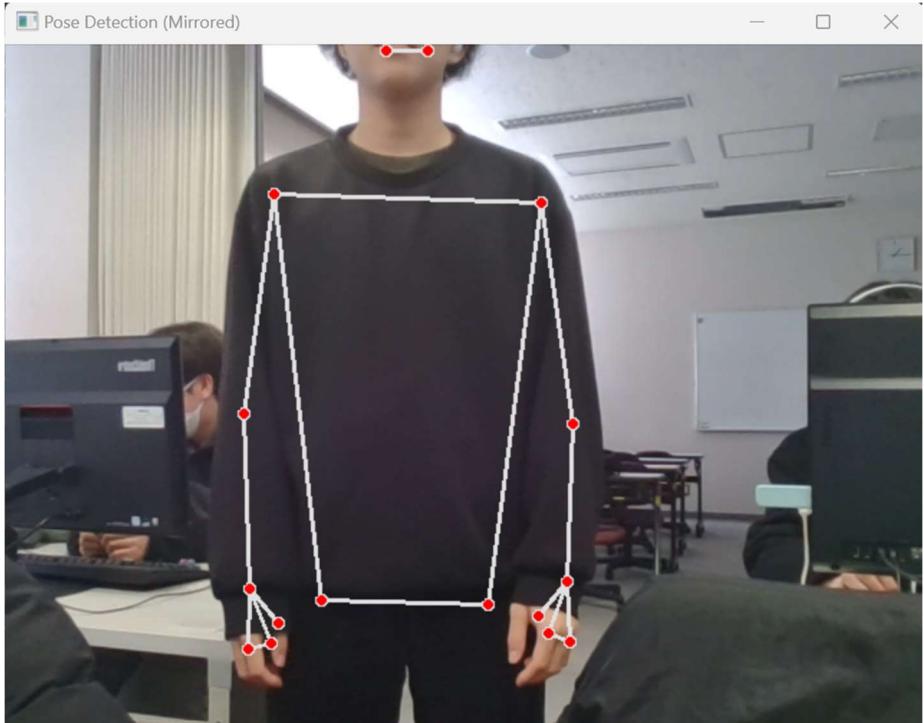


図 2-2 実行結果

本研究は以下の基本的な姿勢検出プログラムを基準として角度検出等の機能を追加することによって作成する。

ライブラリのインポート

```
import cv2
import mediapipe as mp
```

- (1) cv2: OpenCV のライブラリでカメラ映像の取得や画像処理を行う。
- (2) MediaPipe as mp: Google が提供する MediaPipe ライブラリで姿勢推定を行うために使用する。

MediaPipe の設定

```
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
mp_drawing = mp.solutions.drawing_utils
```

- (3) mp_pose = mp.solutions.pose
→ MediaPipe の Pose モジュールを使用するための設定.
- (4) pose = mp_pose.Pose()
→ 姿勢推定を行うためのインスタンスを作成.
- (5) mp_drawing = mp.solutions.drawing_utils
→ MediaPipe のランドマーク(関節)を画像上に描画するためのツール.

カメラの起動と映像の取得

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
```

- (6) cv2.VideoCapture(0)
→ パソコンのカメラ(ID 0)を起動する.
- (7) cap.isOpened()
→ カメラが正常に開かれているか確認.
- (8) ret, frame = cap.read()
→ frame に 1 フレーム分の画像を取得する.
- (9) if not ret: break
→ 映像が取得できない場合,ループを終了.

カメラ映像の左右反転

```
# 映像を左右反転
frame = cv2.flip(frame, 1)
```

- (10) cv2.flip(frame, 1)
→ 画像を 左右反転(ミラーリング) する.左右を反転させることで映像が直感的に見やすくなる.

姿勢推定の実行

```
if result.pose_landmarks:  
    mp_drawing.draw_landmarks(frame, result.pose_landmarks, mp_pose.POSE_CONNECTIONS)
```

(11) pose.process(rgb_frame) を呼び出し, MediaPipe が 姿勢を解析する.

姿勢のランドマーク描画

```
cv2.imshow('Pose Detection (Mirrored)', frame)  
if cv2.waitKey(1) & 0xFF == ord('q'):  
    break
```

(12) cv2.imshow('Pose Detection (Mirrored)', frame)

→ 処理した映像を ウィンドウに表示 する.

(13) cv2.waitKey(1) & 0xFF == ord('q')

→ 「q」キーが押されるとループを終了し, プログラムが終了する.

2.3.2 終了処理

```
cap.release()  
cv2.destroyAllWindows()
```

(1) cap.release()

→ カメラを解放 し, 使用終了.

(2) cv2.destroyAllWindows()

→ 開いている すべてのウィンドウを閉じる.

2.4 よい姿勢とは

本研究では、以下の図に示した通りにより座位姿勢を以下の条件を満たすように定義した。



図 2-3 いい姿勢の基準

よい座位姿勢の条件:

- 足の裏全体が地面にしっかりと接している状態
- 背筋が垂直方向に伸びていて腰の角度が 90°
- 図 2-3 では青色で表示した肩から耳(顔の中心)までのベクトルに対して閾値以内の角度を保っていること

閾値として、首の角度は 0° から 20° 、腰の角度は 90° としました。

第3章 姿勢検出について

3.1 角度の検出方法について

本研究では Python の MediaPipe を用いて、landmark から体の座標を取得する手法で算出し、取得した位置情報を基に各関節間の角度を以下の式を用いて算出した。

$$\vec{a} = (a_x, a_y), \vec{b} = (b_x, b_y) \text{ のなす角を } \vartheta \text{ とすると}$$

$$\cos\vartheta = \frac{a_x b_x + a_y b_y}{\sqrt{a_x^2 + a_y^2} \sqrt{b_x^2 + b_y^2}}$$

図 3-1 ベクトルのなす角度を求める式

この計算では、なす角を利用してことで、関節の動きや姿勢の変化を数値化し、詳細な分析を行うことが可能である。図 3-2 には実装したベクトルの間の角度を計算する関数を示した。

```
# ベクトル間の角度を計算する関数
def calculate_angle(v1, v2):
    dot_product = np.dot(v1, v2)
    magnitude_v1 = np.linalg.norm(v1)
    magnitude_v2 = np.linalg.norm(v2)
    cosine_angle = dot_product / (magnitude_v1 * magnitude_v2)
    angle = np.arccos(np.clip(cosine_angle, -1.0, 1.0)) # 角度の範囲を[-1,1]に制限
    return np.degrees(angle)
```

図 3-1 ベクトル間の角度を計算する関数

記した部分で $\cos \theta$ を計算する。その後、逆三角関数を用いて、 $\cos \theta$ を角度に変換する。

以下に、アプリの測定前準備について説明する。

カメラ設置: カメラを三脚で固定し、ユーザの全身を撮影できるようにユーザの真横に設置する(図 3-3 の右)。

画面配置: ユーザは図 3-3 の左に示した通り、画面に向かって座る。ミラーで自分の横姿を確認するように、カメラプレビュー画面を見ながら姿勢を確認する。

位置調整: カメラプレビュー画面でユーザの全身がしっかりと映っていることを確認し、必要に応じてカメラの位置や角度を調整する。

姿勢確認: ユーザはカメラプレビュー画面を見ながら、自分の姿勢を確認し、必要に応じて調整する。

これらの準備が整ったら、アプリの測定ボタンを押して測定を開始することができる。



図 3-2 姿勢を測定する様子

首と腰の角度を検出するには、以下の図に示した青い丸で囲まれた位置情報(座標)を取得するのである。

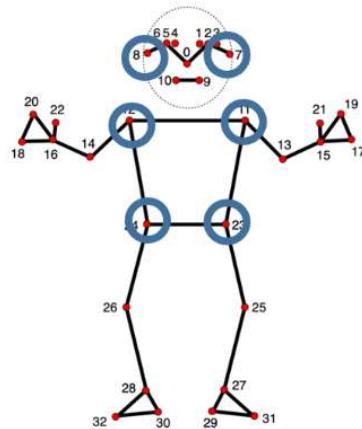


図 3-3 取得する landmark の座標

顔の中心座標を求めるには、取得した座標をもとに耳、肩、腰のそれぞれの中点を求める（図 3-4）。これらの中点同士をつないだ結果を図 3-5 に示した。

```
ear_mid = (
    (ear_left_coords[0] + ear_right_coords[0]) // 2,
    (ear_left_coords[1] + ear_right_coords[1]) // 2,
)
shoulder_mid = (
    (shoulder_left_coords[0] + shoulder_right_coords[0]) // 2,
    (shoulder_left_coords[1] + shoulder_right_coords[1]) // 2,
)
hip_mid = (
    (hip_left_coords[0] + hip_right_coords[0]) // 2,
    (hip_left_coords[1] + hip_right_coords[1]) // 2,
)
knee_mid = [
    (knee_left_coords[0] + knee_right_coords[0]) // 2,
    (knee_left_coords[1] + knee_right_coords[1]) // 2,
```

図 3-4 中点を求めるプログラム

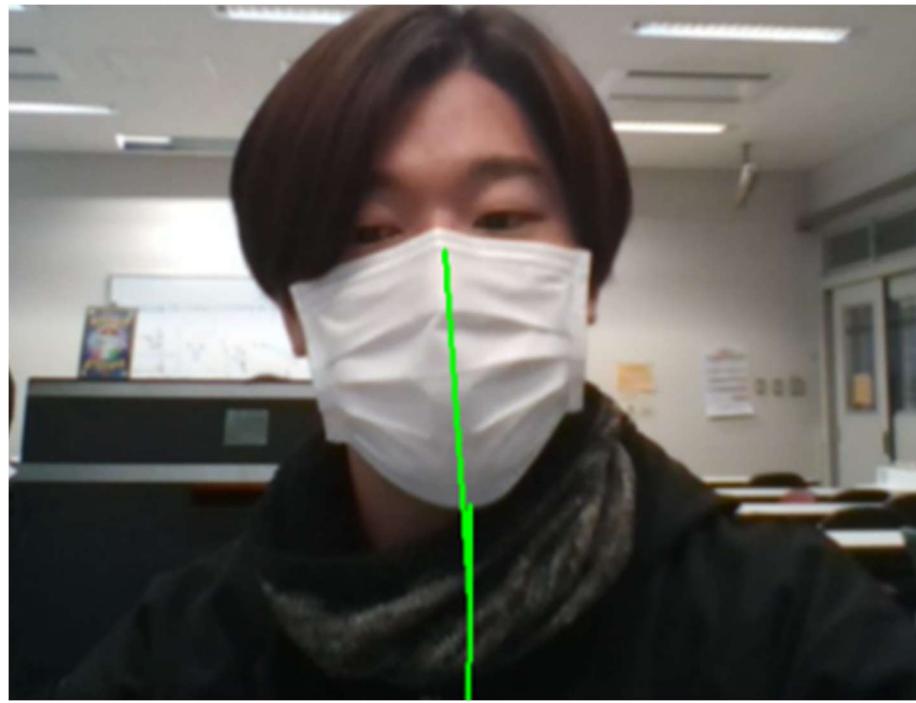


図 3-5 中点を表示した様子

3.2 首の角度の検出方法

首の角度を計算するためには、まず図に示した黄色枠で囲まれた部分の 2 つのベクトルを求める必要がある。最初に、赤色で表示されている肩から顔の中心へのベクトルを計算する。このベクトルは、顔の中心(鼻の中心)の座標から肩の中心の座標を引くことで得られる。次に、緑色で示した垂直方向(Y 軸方向)の単位ベクトルを用いるのである。

これら 2 つのベクトルを用いて、ベクトル間の角度を計算する関数(図 3-2 参照)に渡し、その結果得られた角度(NeckAngle)を画面の右側に表示する。図 3-6 に示されている通り、首の角度は 20.32° でした。さらに首の角度を求めた検証結果を図 3-7 に表示した。

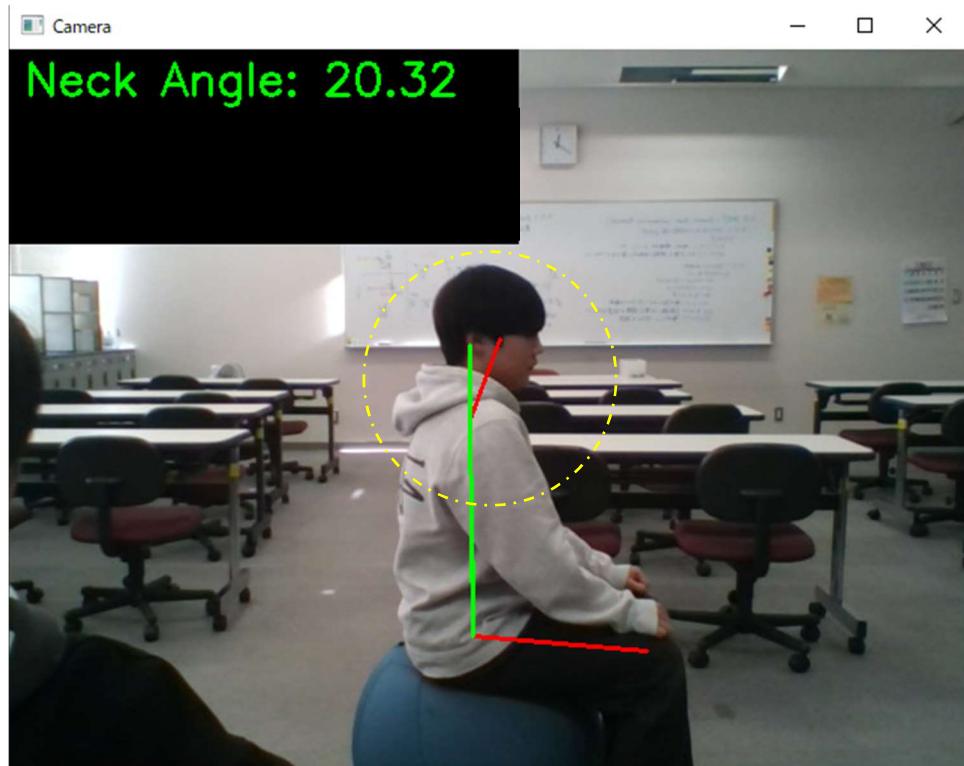


図 3-6 首の角度(Neck Angle 20.32)の検出している検証の様子

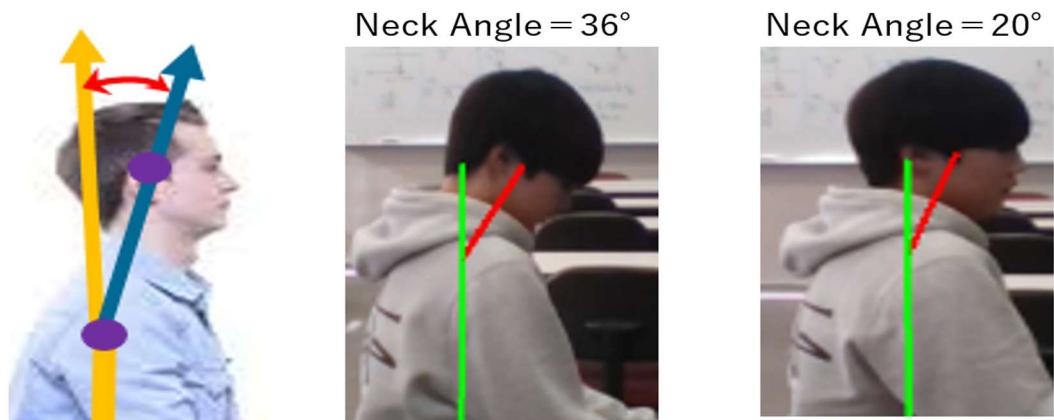


図 3-7 首の角度(Neck Angle)の検出した様子

3.3 腰の角度の検出方法

3.2 と同様に腰の角度を計算するためには、まず図に示した黄色枠で囲まれた部分の 2 つのベクトルを求める必要がある。最初に、赤色で表示されている腰から膝の中心へのベクトルを計算する。次に、緑色で示した垂直方向(Y 軸方向)の単位ベクトルを用いるのである。

これら 2 つのベクトルを用いて、ベクトル間の角度を計算する関数(図 3-2 参照)に渡し、その結果得られた角度(Waist Angle)を画面の右側に表示する。図 3-8 に示されている通り、腰の角度は 80° であった。さらに、椅子の高さによって得られた腰の角度の検証結果を図 3-9 に示す。

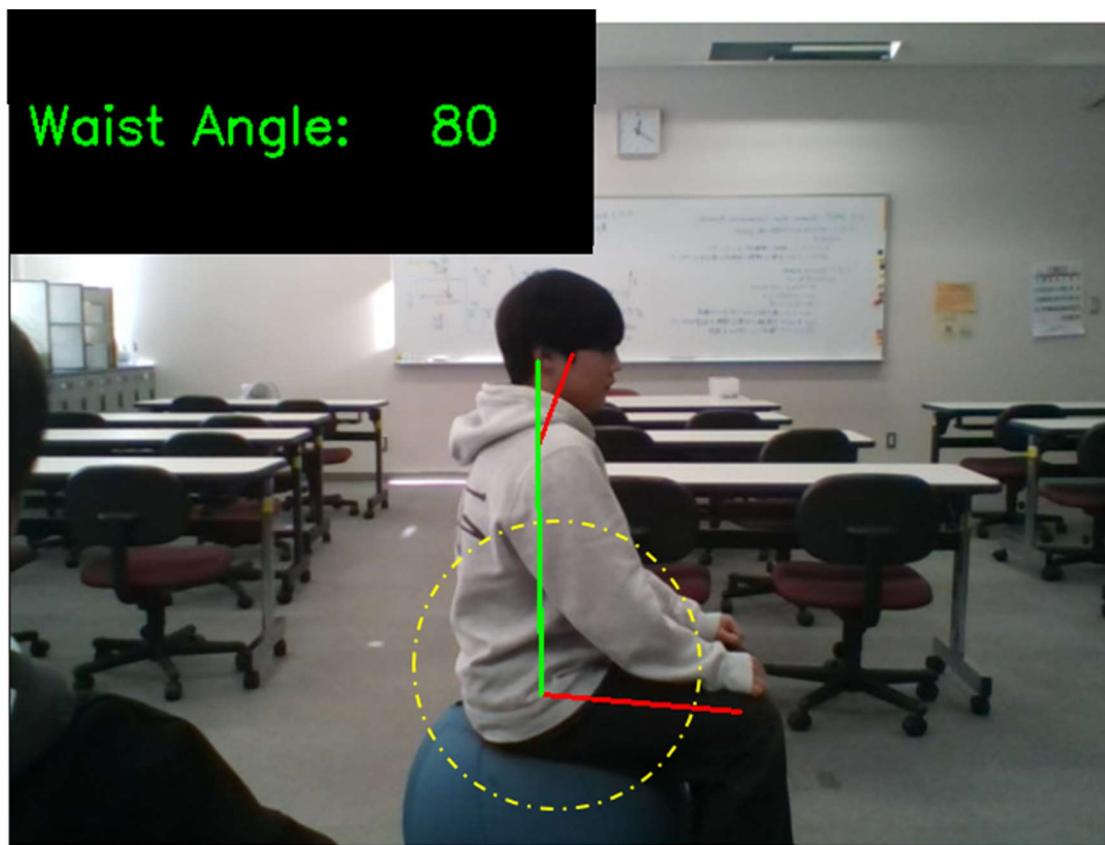


図 3-8 腰の角度(Waist Angle 80) の検出様子

低い	中間	高い
 Angle: 77.75??	 Angle: 89.60??	 Angle: 108.58??
77°	89°	108°

図 3-9 腰の角度の検出様子

背筋が伸びていることを確認するために、図で示すように、腰から肩までのベクトルと Y 軸方向の単位ベクトルがほぼ同じ方向に向かっている場合は緑線で表示し、そうでない場合は赤線で表示した。

第4章 座姿勢の評価(点数化)について

座姿勢の評価は、以下の順序で行われる。まず、検出した首の角度を評価し、次に腰の角度を評価し、最後に総合的な評価を行う。首の角度の評価においては、基準値(20°)を設定し、この基準値を超えた分を 1° あたり1点として減点する。一方、腰の角度の評価においては、椅子の高さに関係なく背筋が正しく伸びているかを確認し、垂直(Y軸)とのずれを計算して、その分を 1° あたり1点として減点する。最終的に、100点からこの2つの減点をそれぞれ引いて座姿勢を評価する。以下に詳しく説明する。

4.1 首の角度の評価

- ・ **首の角度:** 第3章で説明した通りに首の角度を求める(図3-1参照)。
- ・ **基準値の設定:** 本研究では基準値を 20° とする。参考文献[7]のひげ鍼灸整骨院のWebサイトおよび2023年大阪電気通信大学の「集中力を維持するための座位姿勢測定システム」[8]を基に、首の角度は両肩の中間点から垂直上向く黄色のベクトルと、両肩の中間点から耳の中間へ向かう緑色のベクトルのなす角度の範囲を 20° (図4-1の赤で示した範囲)として設定した。



図 4-1 首の角度を求めるベクトルとそのなす角度の範囲

- ・ **点数の算出:**評価方法としては、100点満点で基準値 20° を超えた場合、 1° のずれごとに1点減点する方式を採用した。角度が 20° 未満の場合は減点されない(図4-2参照)。例えば、以下の図では、Neck Angleが 36° の場合は16点減点され、 20° の場合は減点されない。

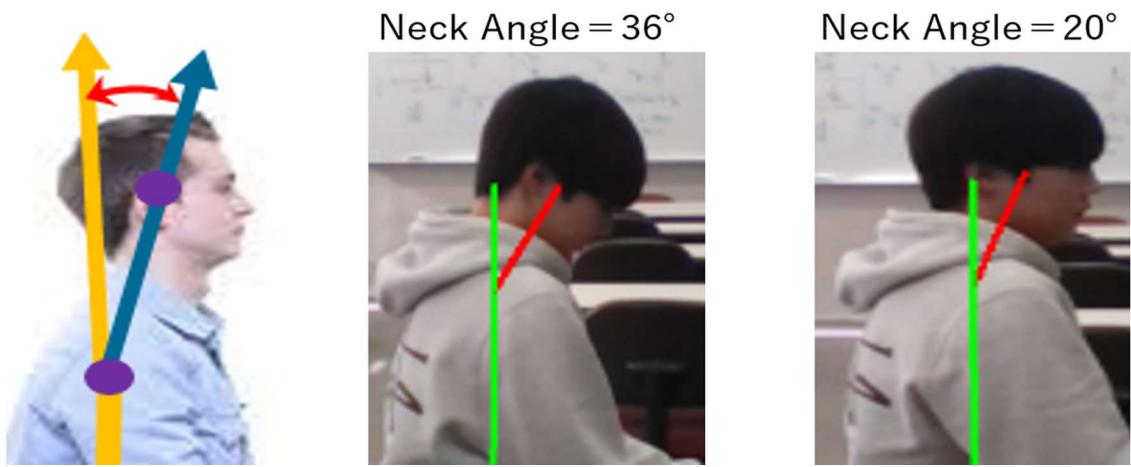


図 4-2 首の角度の検出様子

4.2 腰の角度の評価

- ・ **腰の角度:** 第3章で説明した通りに腰の角度を求める(図3-8参照)。



図 4-3 腰の角度を検出するベクトルとなす角度の範囲

- ・ **腰の角度の基準値:**

くまのみ整骨院・整体院グループの Web サイト[9]を参考にして、図 4-3 に示した両腰の中間点から垂直上向きの黄色のベクトルと、両腰の中間点から両肩の中間点へ向かう緑色のベクトルとのなす角度を 90° とした。これにより、椅子の高さによって腰の角度が変化しても影響を受けないものとなっている。

- ・ **評価方法:**

100 点満点で基準値 90° を超えた場合、 1° のずれごとに 1 点減点される減点方式を採用した。

以下の図 4-4 では背筋が伸びているため、黄色で囲まれている部分に注目すると、減点される点数が 0 であることが表示されている。そして、腰の角度は 90.31° であったため、基準値 90° を超えた分の減点が画面の左上に表示されている。

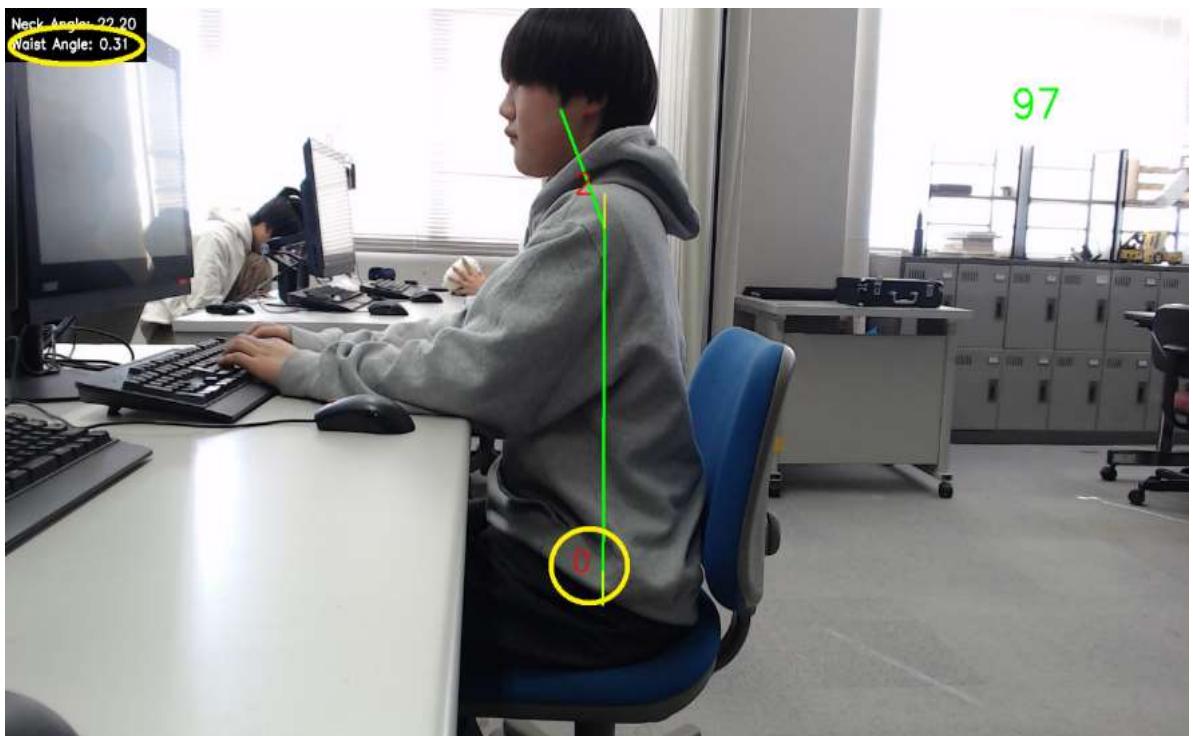


図 4-4 検証の様子

4.3 総合評価

総合点数の算出には、首の点数と腰の点数の合計を 100 から引く減点方式を採用している。具体的な計算式は以下の通りである。

$$\text{点数} = 100 - (\text{首の基準値からのずれ} + \text{腰の基準値からのずれ})$$

点数は、首および腰の角度それぞれ小数点以下を四捨五入して計算するため、点数に 1 点の誤差が生じることがある。

例えば、図 4-5 の実行例では、首の基準値からのずれが 3° 、腰の基準値からのずれが 3° の場合、総合点数は以下のように計算される。

$$\text{総合点数} = 100 - (3^\circ + 3^\circ) = 94 \text{ 点}$$

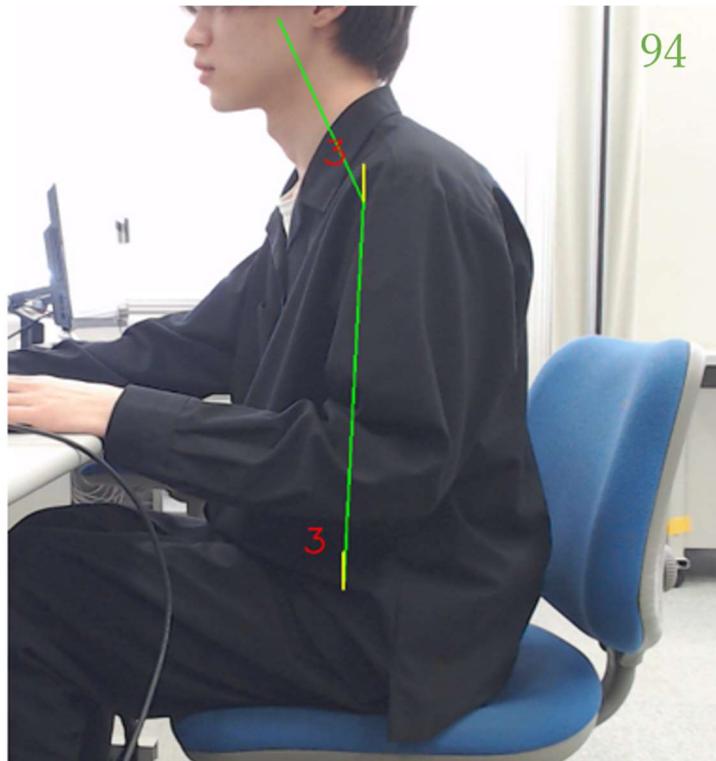


図 4-5 実験結果①(総合点数 94)

図 4-6 の場合、首の角度が 47° であり、基準値から 27° のズレが画面に表示されている。

$$100 - (\text{首の基準値からのずれ } 27^{\circ} + \text{腰の基準値からのずれ } 14^{\circ}) = 59 \text{ 点}$$

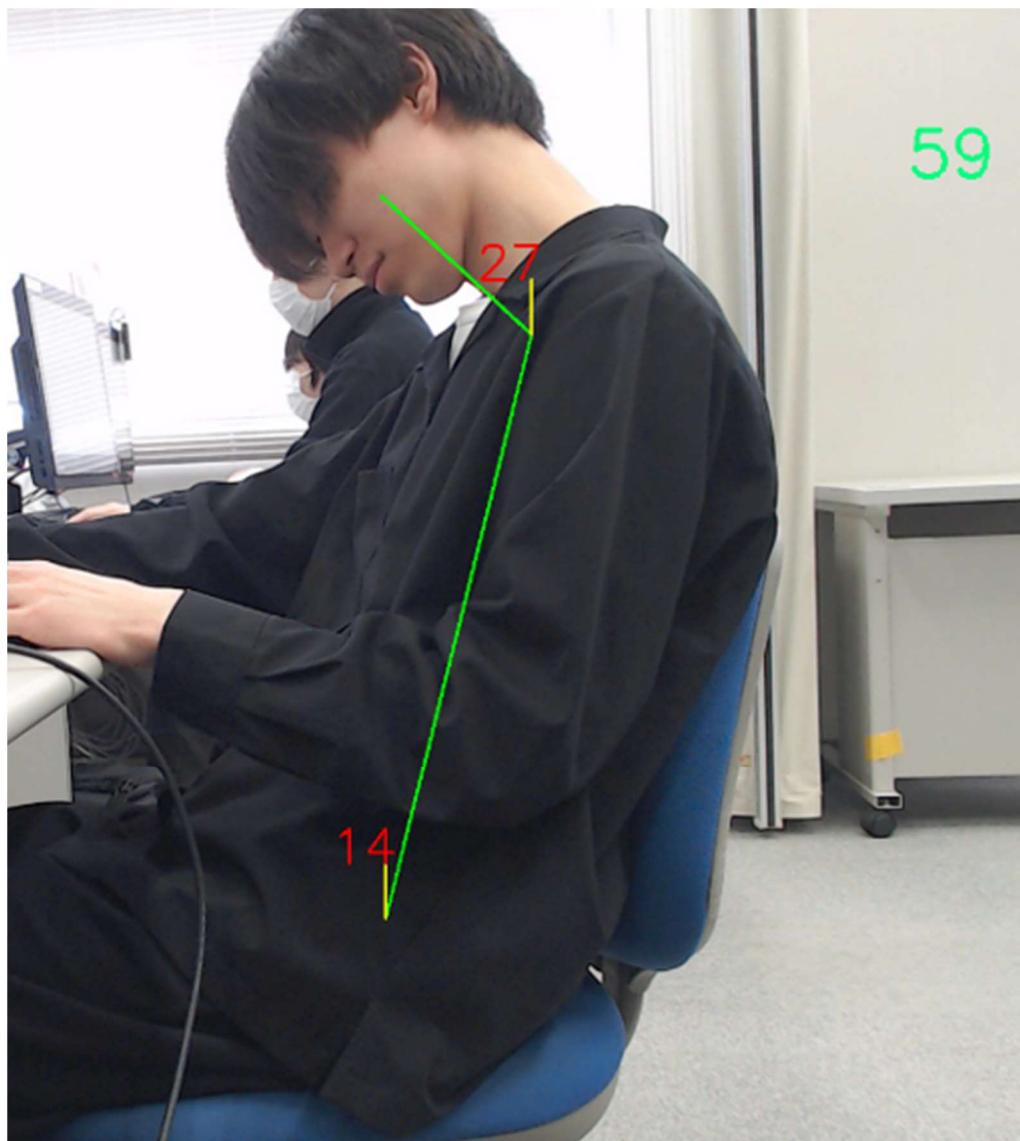


図 4-6 実験結果② (総合点数 59)

例えば、図 4-7 の場合は総合点数 97 点である。しかし、ユーザが厚い服やフードのある服を着ている場合、取得する座標の精度が低下する傾向が見られた。

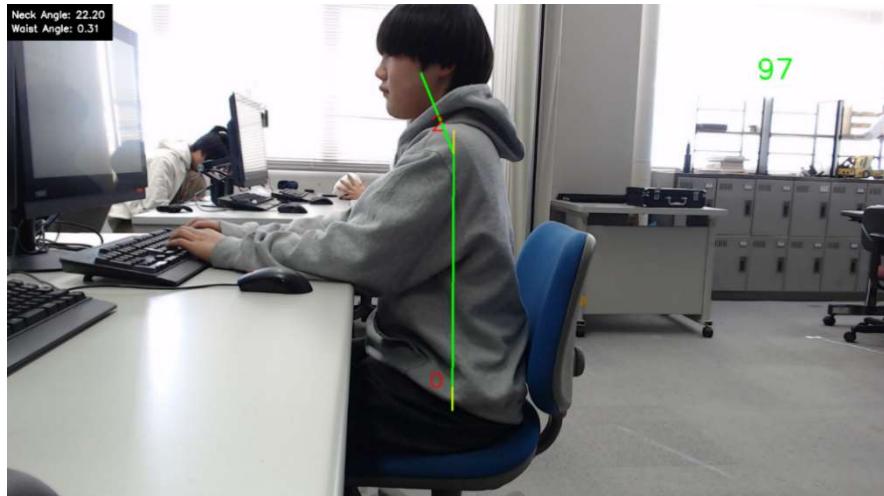


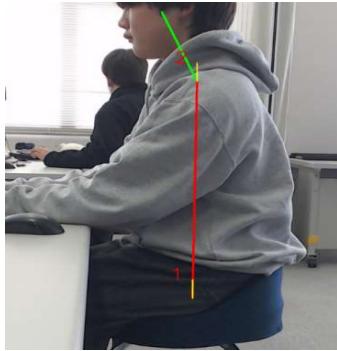
図 4-7 実験結果③の様子（総合点数 97）

4.4 姿勢をサポートするグッズにおける実験結果

本アプリを使用することで、ユーザ自身に最も適したグッズを選ぶきっかけになることを目指して、座姿勢をサポートする椅子用クッションやバランスボールなどのグッズに関する検証実験を行った。その結果を以下の表にまとめた。バランスボールを使用した場合、99 点という非常に高い評価が得られた。一方、姿勢サポート椅子を使用した場合、腰の座標を正確に読み取るのが難しかったものの、総合評価は 90 点であった。



総合評価(椅子) = 98 点



総合評価(椅子用クッション)は 97 点



総合評価(姿勢サポート椅子)は 90 点

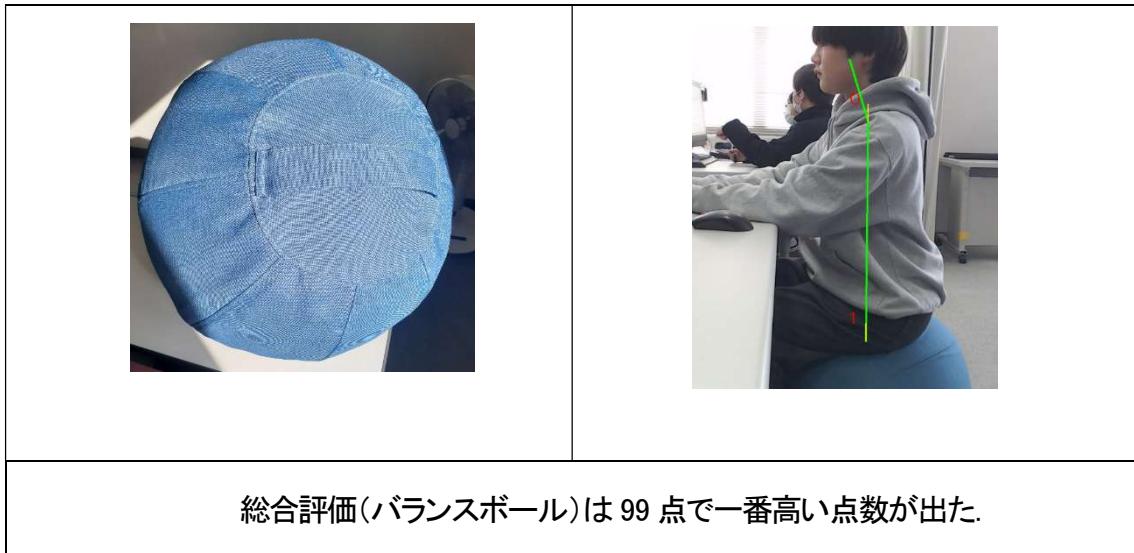


図 4-8 姿勢をサポートするグッズにおける実験結果

第 5 章 実装したアプリについて

5.1 アプリの構成図について(システムの流れ)

アプリの構成図について(システムの流れ)を以下の図に示す。

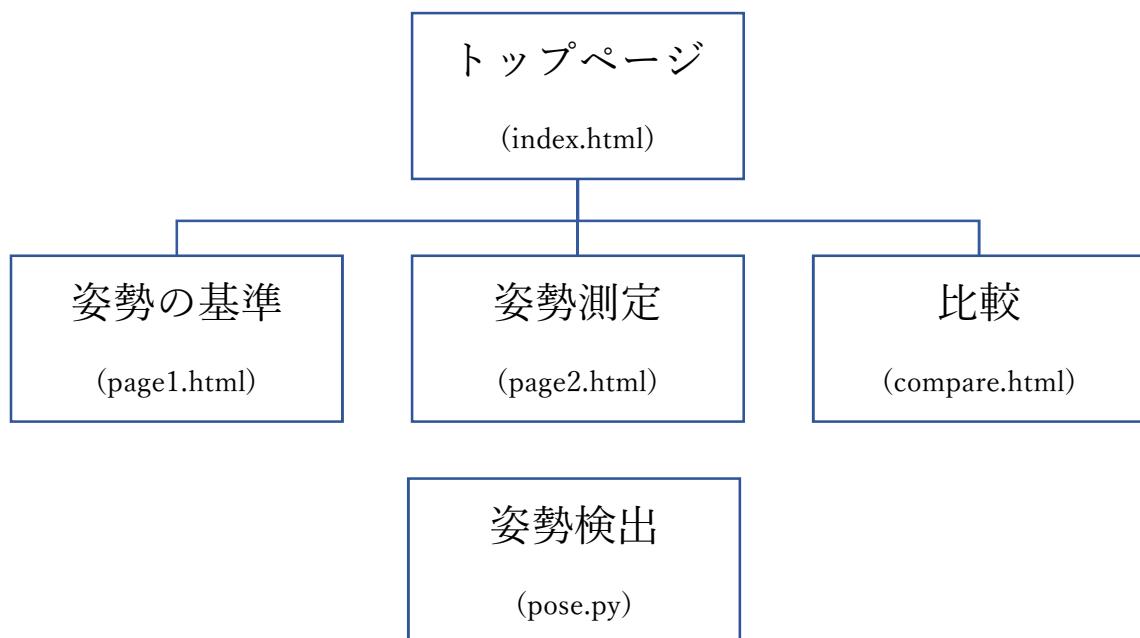


図 5-1 アプリの構成図

本システムでは、以下の手順で姿勢の測定および比較を行う。

姿勢測定の実行: ユーザはトップページ(index.html)から姿勢測定ページ(page2.html)へ移動し、「姿勢をチェックする」ボタンを押す。これにより,pose.py が実行され,カメラが起動し,ユーザーの姿勢をリアルタイムで解析する。MediaPipe を用いて関節の座標データを取得し,姿勢の点数や角度を算出する。

測定データの比較: ユーザはトップページ(index.html)から比較ページ(compare.html)へ移動し、「比較する」ボタンを押す。これにより,csv_op.py が実行され,過去に取得した姿勢データ(関節角度の平均や画像データなど)を閲覧し,現在の姿勢との比較が可能となる。

5.2 アプリの機能について

本システムでは、ユーザが自身の姿勢を測定し、過去のデータと比較するための以下の機能を提供する。

① 姿勢測定機能

- ・ ユーザは「姿勢をチェックする」ボタンを押すと Python スクリプトが実行され、姿勢を測定できる。
- ・ MediaPipe を用いた座標の取得により、姿勢の点数や角度を算出し、フィードバックを提供する。

② 姿勢データの保存機能

- ・ 測定した姿勢データ（角度・点数・画像など）を保存し、後で比較できるようにする。
- ・ 保存されたデータは CSV 形式で管理される。

③ 姿勢比較機能

- ・ ユーザは「比較する」ボタンを押すことで、過去の測定データと現在の姿勢を比較できる。
- ・ 過去の角度の平均値や画像を確認し、どのように姿勢が変化したかを分析できる。

5.3 画面遷移について

(1) TOP ページ

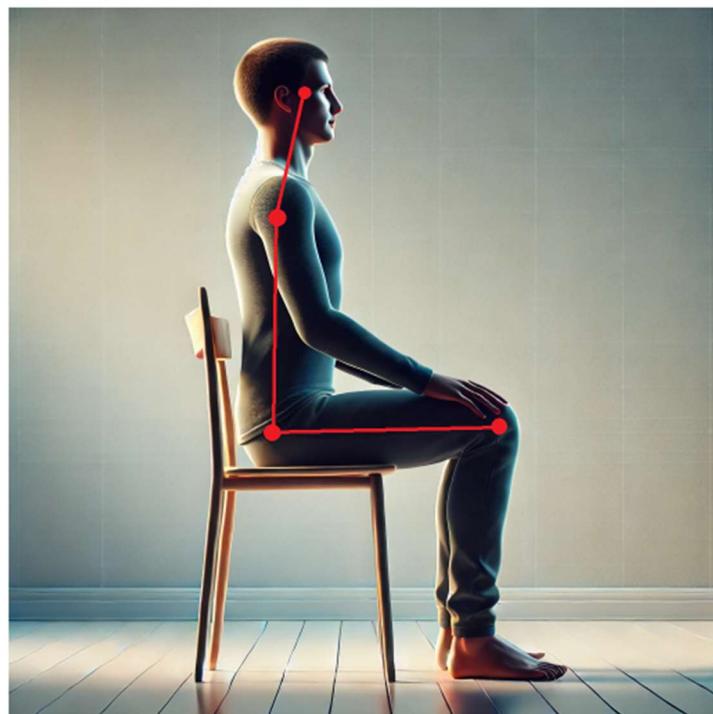


図 5-2 ページ遷移画面

図 5.2 に示すボタンのいずれかをクリックすることで、各ページへ移動し、姿勢測定や比較を行うことができる。

(2) 基準説明ページ

理想的な座り姿勢の角度



この画像は理想的な座り姿勢を示しています。腰、首の角度に注意し、背筋をまっすぐ保つことが重要です。

**腰の角度
($\angle BCD$)** 理想的な角度: 90 ~110度 腰と背中がこの範囲にあると、腰への負担が軽減され、背筋をまっすぐ保つことができます。椎間板への圧力もこの範囲で最も少くなります。

**首の角度
($\angle ABC$)** 理想的な角度: 0~20 度の前傾 頭がわずかに前に傾いている状態が理想です。首や肩への負担が軽減されます。頭が前に出過ぎると、首に過度な負担がかかります。

図 5-3 基準確認ページ

TOP ページから図 5.3 の「正しい姿勢の基準」ページへ移動することで腰、肩の理想の角度を確認することができる。

(3) 測定ページ

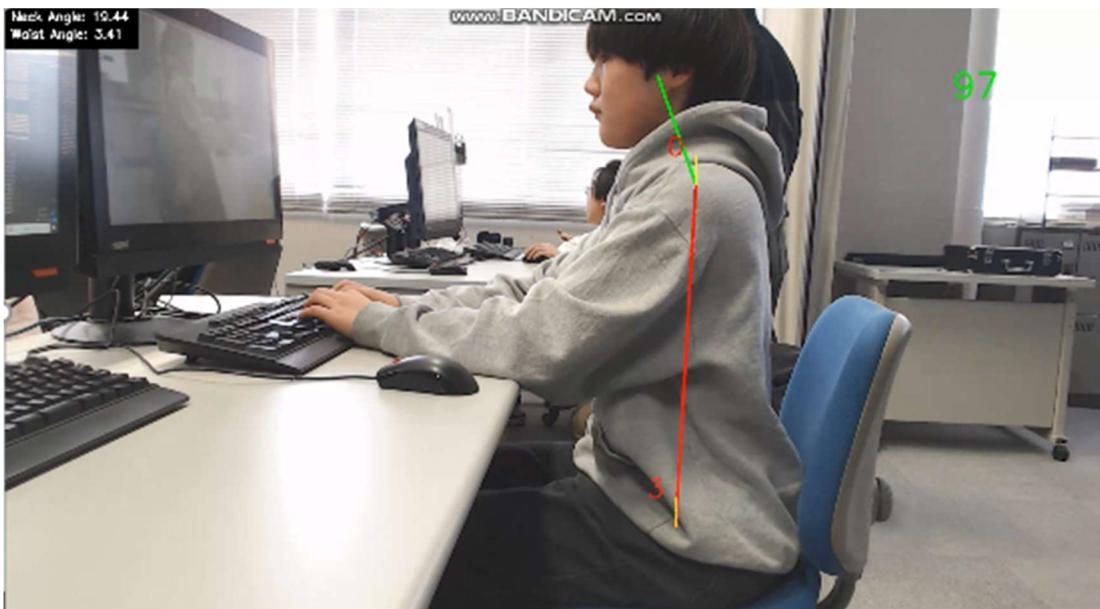


図 5-4 測定ページ

「姿勢をチェックする」ボタンをクリックすると、図 5.4 のような画面が表示される。

Python スクリプトが実行され、姿勢の測定が開始される。測定結果として、自分の姿勢の点数や腰・首の角度を画面上で確認できる。

(4) 確認ページ



図 5-5 確認ページ

図 5.5 では、首や腰の角度の平均値や点数の平均が表示され、測定時の画像も一緒に閲覧できる。

また、「戻る」ボタンや「次へ」ボタンを使って、過去 10 件までのデータを遡り、比較できる。第 6 章に詳しく説明する。

第6章 振り返り機能の実装について

6.1 特徴

- ・ **CSV および画像の管理:** プログラムは CSV および画像ファイルをディレクトリ内のファイル数が増えすぎないように管理する。ファイルの最大数は 10 に設定している。
- ・ **固定画像の表示:** アプリケーションは左側に固定画像として 2 つの画像を表示する。これらの画像は指定されたパス (`./img/graph.jpg` および `./img/sesuji.jpg`) から読み込んでいる。
- ・ **動的な CSV および画像の表示:** プログラムは動的に CSV ファイルと対応する画像を読み込み、表示する。各 CSV ファイルは Notebook ウィジェットの別々のタブに表示する。CSV ファイルは Treeview ウィジェットに表示され、適切なヘッダーと列が設定されている。画像は CSV データの下に表示され、指定されたサイズにリサイズされる。
- ・ **ナビゲーションボタン:** アプリケーションは異なる CSV ファイルのタブ間を簡単に移動するためのナビゲーションボタンを提供する。

以下の図 6.1 には比較ボタンクリックした後の部分を示す。

```

1   # CSVと画像の最大保存数を制限する関数
2   def limit_files(self, directory, max_files=10, extensions=[]):
3       """ 指定フォルダ内のファイルを max_filesまでに制限し、超えた分は古いものから削除 """
4       files = sorted([f for f in os.listdir(directory) if any(f.endswith(ext) for ext in extensions)],
5                      key=lambda x: os.path.getmtime(os.path.join(directory, x)))  # 古い順にソート
6
7       while len(files) > max_files:
8           oldest_file = files.pop(0)  # 一番古いファイルを取得
9           os.remove(os.path.join(directory, oldest_file))  # 削除
10
11  # 画像表示メソッド
12  def display_image(self, img_path, label, size):
13      """ 指定した画像をラベルに表示 """
14      try:
15          img = Image.open(img_path)
16          img.thumbnail(size, Image.LANCZOS)
17          img_tk = ImageTk.PhotoImage(img)
18          label.config(image=img_tk)
19          label.image = img_tk  # 参照を保持
20      except Exception as e:
21          print(f"画像の読み込みに失敗しました: {img_path}, エラー: {e}")
22
23
24  frame = ttk.Frame(self.notebook)
25  self.notebook.add(frame, text=file_name)
26  self.tabs.append(frame)
27  tree = ttk.Treeview(frame, show='headings')
28  tree.pack(fill=tk.BOTH)
29  file_path = os.path.join(csv_directory, file_name)
30  with open(file_path, newline='', encoding='utf-8') as csv_file:
31      reader = csv.reader(csv_file)

31  reader = csv.reader(csv_file)
32  headers = next(reader)
33  tree["columns"] = headers
34  for header in headers:
35      tree.heading(header, text=header)
36      tree.column(header, width=100, anchor="center")
37
38  for row in reader:
39      tree.insert("", "end", values=row)
40
41  if i < len(self.image_files):
42      image_path = os.path.join(image_directory, self.image_files[i])
43      img = Image.open(image_path)
44      img.thumbnail((650, 650))  # 比率を維持してサイズ変更
45
46      img_tk = ImageTk.PhotoImage(img)
47      label = tk.Label(frame, image=img_tk)
48      label.image = img_tk
49      label.pack(pady=3)
50
51  # 前後のタブに移動するボタン
52  def next_tab(self):
53      """次のタブへ移動"""
54      self.current_tab = (self.current_tab + 1) % len(self.tabs)
55      self.notebook.select(self.current_tab)
56
57  def previous_tab(self):
58      """前のタブへ移動"""
59      self.current_tab = (self.current_tab - 1) % len(self.tabs)
60      self.notebook.select(self.current_tab)

```

図 6-1 振り返り機能で使用した関数

6.2 実装の詳細

実装した機能を以下の表にまとめた。

	CSV および画像ファイルの制限:	(参照図 6-1 limit_files 関数)
1	※ limit_files メソッドは、指定されたディレクトリ内のファイル数を最大 10 に制限する。ファイル数がこの制限を超えた場合、最も古いファイルが削除される。	
2	定画像の読み込みと表示:	(参照 図 6-2 display_image 関数)
	※ display_image メソッドは、指定されたパスから画像を読み込み、表示する。画像は thumbnail メソッドを使用してアスペクト比を維持しながらリサイズされる。	
3	Notebook ウィジェットによる CSV ファイルの表示	(参照 図 6-1 24 行目～)
	※ プログラムは Notebook ウィジェットを使用して、各 CSV ファイルのタブを作成する。各タブには CSV データを表示する Treeview ウィジェットが含まれている。	
4	CSV データの読み込みと表示	(参照 図 6-1 30 行目～)
	※ CSV ファイルは csv モジュールを使用して読み込み、データは Treeview ウィジェットに表示される。ヘッダーは列見出しとして設定され、各データ行が Treeview に挿入される。	
5	対応する画像の表示	(参照 図 6-1 41 行目～)
	※ 各 CSV ファイルに対応する画像は、各タブ内の CSV データの下に表示される。画像は指定されたサイズにリサイズされるため、thumbnail メソッドを使用する。	
6	タブ間のナビゲーション:	(参照 図 6-1 next_tab、 previous_tab)
	※ next_tab および previous_tab メソッドを使用して、ユーザは異なる CSV ファイルのタブ間を移動することができる。	

6.3 使用したライブラリについて

Tkinter は Python で GUI(グラフィカルユーザインターフェース)を作成するための標準ライブラリである。以下のような場面で使用される。

デスクトップアプリケーションの開発: Tkinter は、ウィンドウ、ボタン、テキストボックス、ラベルなどの GUI 要素を作成し、ユーザーとコンピュータのやり取りを可能にする。例えば、簡単なテキストエディタや画像ビューアなどのデスクトップアプリケーションを作成する際に使用される。

データ表示: Tkinter は、データを視覚的に表示するためのツールとしても使用される。例えば、CSV ファイルのデータを表形式で表示、グラフや画像を表示するアプリケーションを作成する際に役立つ。以下は、Tkinter が使用されている部分であり、GUI 要素の作成と配置が行われている。GUI の設計およびユーザインターフェースの操作はすべて Tkinter を通じて行われている。

```
2 # (1) Tkinterモジュールのインポート:
3 import tkinter as tk
4 from tkinter import ttk
5
6 # (2) メインウィンドウの作成:
7 root = tk.Tk() # メインウィンドウ作成(この時点では空)
8
9 # (3) ウィンドウタイトルの設定:
10 self.root.title("CSV & Image Viewer") # ウィンドウのタイトル設定
11
12 # (4) フレームの作成と配置:
13 self.left_frame = ttk.Frame(root)
14 self.left_frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)
15
16 # (5) ラベルの作成と配置:
17 self.img_label1 = tk.Label(self.left_frame)
18 self.img_label1.pack(pady=0)
19
20 self.img_label2 = tk.Label(self.left_frame)
21 self.img_label2.pack(pady=50)
22
23 # (6) Notebookウィジェット(タブ用)の作成と配置:
24 self.notebook = ttk.Notebook(root)
25 self.notebook.pack(fill=tk.BOTH, expand=False)
```

```

26
27 # (7) Treeviewウィジェットの作成と配置(CSVデータ表示エリア):
28 tree = ttk.Treeview(frame, show='headings')
29 tree.pack(fill=tk.BOTH)
30
31 vsb = ttk.Scrollbar(frame, orient="vertical", command=tree.yview)
32 vsb.pack(side=tk.RIGHT, fill="y")
33 tree.configure(yscrollcommand=vsb.set)
34
35 hsb = ttk.Scrollbar(frame, orient="horizontal", command=tree.xview)
36 hsb.pack(side=tk.BOTTOM, fill="x")
37 tree.configure(xscrollcommand=hsb.set)
38
39 # (8) ボタンの作成と配置:
40 self.prev_button = ttk.Button(button_frame, text="戻る", command=self.previous_tab)
41 self.prev_button.pack(side=tk.LEFT, padx=10)
42
43 self.next_button = ttk.Button(button_frame, text="次へ", command=self.next_tab)
44 self.next_button.pack(side=tk.RIGHT, padx=10)
45

```

図 6-2 Tkinterを通じて行われている GUI 要素に関する基本的な部分の説明

```

1 # (1) Pillowモジュールのインポート:
2 from PIL import Image, ImageTk
3
4 # (2) 画像の読み込みとリサイズ(サムネイル)の作成:(display_imageメソッド内)
5 def display_image(self, img_path, label, size):
6     try:
7         img = Image.open(img_path)
8         img.thumbnail(size, Image.LANCZOS)
9         img_tk = ImageTk.PhotoImage(img)
10        label.config(image=img_tk)
11        label.image = img_tk # 参照を保持
12    except Exception as e:
13        print(f"画像の読み込みに失敗しました: {img_path}, エラー: {e}")
14
15 # (3) 対応する画像の読み込みと表示: (各タブ内のCSVデータの下に画像を表示する部分)
16
17 if i < len(self.image_files):
18     image_path = os.path.join(image_directory, self.image_files[i])
19     img = Image.open(image_path)
20     img.thumbnail((650, 650)) # 比率を維持してサイズ変更
21
22     img_tk = ImageTk.PhotoImage(img)
23     label = tk.Label(frame, image=img_tk)
24     label.image = img_tk
25     label.pack(pady=3)
26

```

図 6-3 Pillowによる画像処理の部分

PIL (Python Imaging Library) またはその後継である Pillow は、画像処理に使用されるライブラリである。このプログラムでは、主に画像の読み込み、リサイズ、表示に使用されている。以下の 図 6.3 で示す通り、PIL (Pillow) は、画像を読み込み、リサイズし、Tkinter のラベルに表示するために使用される。画像のリサイズには thumbnail メソッドが使用され、アスペクト比を維持しながら画像サイズを変更する。また、ImageTk.PhotoImage を使用して、Tkinter ラベルに画像を表示するためのオブジェクトが作成される。

第7章 終わりに

本研究では,Python の環境構築からプログラム開発,アプリ作成までを一貫して行った.研究を通じて,MediaPipe や OpenCV を活用した姿勢解析の仕組みを理解することができた. MediaPipe を用いて,耳,肩,腰の座標を取得し,座位姿勢の改善をサポートするアプリを開発した. 本アプリでは,測定時の角度平均や点数のデータ,画像を保存し,過去のデータと比較できるようにした.

このアプリを活用することで,デスクワーク時の姿勢改善に役立てるとともに,適切なクッションなどのサポートアイテムを見つけるきっかけになることを期待する.

参考文献

- [1] chrome-extension://efaidnbmnnibpcajpcglclefindmkaj/https://www.jstage.jst.go.jp/article/jmel/2023/0/2023_38/_pdf?utm_source=chatgpt.com (姿勢測定マット)
- [2] https://jglobal.jst.go.jp/detail?JGLOBAL_ID=202002282111238458&utm_source=chatgpt.com (センサ埋め込み椅子)
- [3] chrome-extension://efaidnbmnnibpcajpcglclefindmkaj/https://www.jstage.jst.go.jp/article/jje/58/Supplement/58_2G4-01/_pdf?utm_source=chatgpt.com (スクワット運動支援)
- [4] https://www.ergoassist.ca/ergonomic-tips/rule-of-90-90-90 (いい姿勢の基準)
- [5] https://learnopencv.com/building-a-body-posture-analysis-system-using-MediaPipe/
Building a Poor Body Posture Detection & Alert System Using MediaPipe Body Tracking
- [6] MediaPipe を用いたトレーニングの効果をフィードバックするアプリの作成
日野翼, 武蔵治樹 (2023)
- [7] 柏とらひげ鍼灸整骨院 <https://tora-hige.net/symptom/straight-neck>
- [8] 大阪電気通信大学 集中力を維持するための座位姿勢測定システム(2023)
- [9] 正しい座り方のコツ！骨盤を立てて身体の負担を軽減しよう
<https://kumanomi-seikotu.com/blog/3333/>

付録

使用したライブラリ

OpenCV

- 概要

画像処理やコンピュータービジョンの分野で広く使用されるライブラリであり,画像の読み込み,加工,解析が可能.

- 使用用途

カメラから取得した画像データの処理に使用し,ユーザの姿勢をリアルタイムでキャプチャする際に活用.

MediaPipe

- 概要

Google が開発した機械学習ベースのライブラリであり,手や顔,姿勢の検出などをリアルタイムで行うことができる.

- 使用用途

ユーザの姿勢を解析し,関節の座標データを取得するために使用した.これにより,姿勢の変化を数値的に把握することが可能となった.

Numpy

- 概要

高速な数値計算を可能にする Python ライブライアリであり,配列や行列演算を効率的に処理できる.

- 使用用途

MediaPipe で取得した関節の座標データをもとに,ベクトル計算や角度の算出を行うために使用した.これにより,姿勢の評価を数値的に解析することが可能となった.

Flask

- 概要

軽量な Web アプリケーションフレームワークであり,バックエンドの開発に適している.

- 使用用途

フロントエンドとバックエンドの連携を行い,ユーザから取得した画像データの処理,姿勢評価の計算,そして結果の表示を行う Web アプリケーションの構築に使用した.

Csv

- 概要

カンマ区切りのデータを扱うための標準ライブラリであり,データの保存や読み込みに利用される.

- 使用用途

取得した関節の座標データや姿勢評価結果を記録し,データ分析や結果の保存を行うために使用した.

Datetime

- 概要

日時情報を扱うための Python の標準ライブラリであり,現在時刻の取得や日時フォーマットの変換が可能である.

- 使用用途

姿勢評価の記録において,データにタイムスタンプを付与し,解析や管理を容易にするために使用した.

ソースコード一覧

● App.py

```
1  from flask import Flask, render_template, request, jsonify, redirect, url_for
2  import subprocess
3
4  app = Flask(__name__)
5
6  @app.route('/')
7  def index():
8      return render_template('index.html')
9
10 @app.route('/page1')
11 def page1():
12     return render_template('page1.html')
13
14 @app.route('/page2')
15 def page2():
16     return render_template('page2.html')
17
18 @app.route('/run_pose')
19 def run_pose():
20     subprocess.run(['python', 'scripts/pose.py']) # pose.pyの実行
21     return redirect(url_for('page2'))
22
23 @app.route('/compare')
24 def compare():
25     return render_template('compare.html')
26
27 @app.route('/run_compare')
28 def run_compare():
29     subprocess.run(['python', 'scripts/compare.py']) # compare.pyの実行
30     return redirect(url_for('compare'))
31
32 @app.route('/page4')
33 def page4():
34     return render_template('page4.html')
35
36
37 @app.route('/run_script', methods=['POST'])
38 def run_script():
39     try:
40         # 実行したいPythonファイルを指定
41         result = subprocess.run(['python', 'scripts/csv_op.py'], capture_output=True, text=True)
42         return jsonify({'output': result.stdout, 'error': result.stderr})
43     except Exception as e:
44         return jsonify({'error': str(e)})
45
46 if __name__ == '__main__':
47     app.run(debug=True)
```

● Csv_op.py

```
1 import tkinter as tk
2 from tkinter import ttk
3 import csv
4 import os
5 from PIL import Image, ImageTk # 画像表示用
6
7 #-----Tkinterの設定-----
8 class CSVImageViewer:
9     def __init__(self, root, csv_directory, image_directory):
10         self.root = root #メインウインドウ
11         self.root.title("CSV & Image Viewer") #ウインドウのタイトル設定
12
13         # csvと画像のファイルの個数の管理(ここでは最大で10個として古いものから削除)
14         self.limit_files(csv_directory, max_files=10, extensions=[".csv"])
15         self.limit_files(image_directory, max_files=10, extensions=[".png", ".jpg", ".jpeg"])
16
17         # 左側に固定する画像の指定
18         self.side_image1 = "./img/graph.jpg"
19         self.side_image2 = "./img/sesuji.jpg"
20
21         # CSVと画像のディレクトリ
22         self.csv_directory = csv_directory
23         self.image_directory = image_directory
24
25         # CSV & 画像ファイルを取得(上限を適用済み)
26
27         # csvディレクトリ内のcsvファイルを日時順に並べ替える
28         self.csv_files = sorted([f for f in os.listdir(csv_directory) if f.endswith(".csv")])
29         # photoディレクトリ内の画像ファイルを日時順に並べ替える
30         self.image_files = sorted([f for f in os.listdir(image_directory) if f.lower().endswith((".png", ".jpg", ".jpeg"))])
31
32         # スタイル設定
33         style = ttk.Style()
34         # フォント、フォントサイズの設定(数字部分で変更可)
35         style.configure("Treeview", font=("Arial", 36))
36         style.configure("Treeview.Heading", font=("Arial", 16, "bold"))
37
38         # 現在のタブインデックス
39         self.current_tab = 0
40
41         # 固定されている左側の画像
42         self.left_frame = ttk.Frame(root)
43         self.left_frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)
44
45         # 画像ラベル作成
46         self.img_label1 = tk.Label(self.left_frame)
47         self.img_label1.pack(pady=0)
48
49         self.img_label2 = tk.Label(self.left_frame)
50         self.img_label2.pack(pady=50)
51
52         # 固定画像を表示
53         self.display_image(self.side_image1, self.img_label1, (350, 350))
54         self.display_image(self.side_image2, self.img_label2, (350, 350))
55
56         # Notebookウィジット(タブ用)
57         self.notebook = ttk.Notebook(root)
58         self.notebook.pack(fill=tk.BOTH, expand=False)
59
60         # 各CSVごとにタブを作成
61         self.tabs = []
```

```

62      for i, file_name in enumerate(self.csv_files):
63          frame = ttk.Frame(self.notebook)
64          self.notebook.add(frame, text=file_name)
65          self.tabs.append(frame)
66
67      # CSVデータ表示エリア
68      tree = ttk.Treeview(frame, show='headings')
69      tree.pack(fill=tk.BOTH)
70
71      vsb = ttk.Scrollbar(frame, orient="vertical", command=tree.yview)
72      vsb.pack(side=tk.RIGHT, fill="y")
73      tree.configure(yscrollcommand=vsb.set)
74
75      hsb = ttk.Scrollbar(frame, orient="horizontal", command=tree.xview)
76      hsb.pack(side=tk.BOTTOM, fill="x")
77      tree.configure(xscrollcommand=hsb.set)
78
79      # CSVファイルを読み込み
80      file_path = os.path.join(csv_directory, file_name)
81      with open(file_path, newline='', encoding='utf-8') as csv_file:
82          reader = csv.reader(csv_file)
83          headers = next(reader) # ヘッダーを取得
84
85          tree["columns"] = headers
86          for header in headers:
87              tree.heading(header, text=header)
88              tree.column(header, width=100, anchor="center")
89
90          for row in reader:
91              tree.insert("", "end", values=row)
92
93      # 画像を表示
94      if i < len(self.image_files):
95          image_path = os.path.join(image_directory, self.image_files[i])
96          img = Image.open(image_path)
97          img.thumbnail((650, 650)) # 比率を維持してサイズ変更
98
99          img_tk = ImageTk.PhotoImage(img)
100         label = tk.Label(frame, image=img_tk)
101         label.image = img_tk
102         label.pack(pady=3)
103
104     # ボタンエリア
105     button_frame = ttk.Frame(root)
106     button_frame.pack(fill=tk.X, pady=0)
107
108     self.prev_button = ttk.Button(button_frame, text="戻る", command=self.previous_tab)
109     self.prev_button.pack(side=tk.LEFT, padx=10)
110
111     self.next_button = ttk.Button(button_frame, text="次へ", command=self.next_tab)
112     self.next_button.pack(side=tk.RIGHT, padx=10)
113
114     # CSVと画像の最大保存数を制限する関数
115     def limit_files(self, directory, max_files=10, extensions=[]):
116         """ 指定フォルダ内のファイルを max_filesまでに制限し、超えた分は古いものから削除 """
117         files = sorted([f for f in os.listdir(directory) if any(f.endswith(ext) for ext in extensions)],
118                       key=lambda x: os.path.getmtime(os.path.join(directory, x))) # 古い順ソート
119
120         while len(files) > max_files:
121             oldest_file = files.pop(0) # 一番古いファイルを取得
122             os.remove(os.path.join(directory, oldest_file)) # 削除

```

```
123
124
125     # 画像表示メソッド
126     def display_image(self, img_path, label, size):
127         """ 指定した画像をラベルに表示 """
128         try:
129             img = Image.open(img_path)
130             img.thumbnail(size, Image.LANCZOS)
131             img_tk = ImageTk.PhotoImage(img)
132             label.config(image=img_tk)
133             label.image = img_tk # 参照を保持
134         except Exception as e:
135             print(f"画像の読み込みに失敗しました: {img_path}, エラー: {e}")
136
137     # 前後のタブに移動するボタン
138     def next_tab(self):
139         """次のタブへ移動"""
140         self.current_tab = (self.current_tab + 1) % len(self.tabs)
141         self.notebook.select(self.current_tab)
142
143     def previous_tab(self):
144         """前のタブへ移動"""
145         self.current_tab = (self.current_tab - 1) % len(self.tabs)
146         self.notebook.select(self.current_tab)
147
148     # メイン処理
149     csv_directory = "./csv"      # CSVデータのフォルダ
150     image_directory = "./photo"  # 画像データのフォルダ
151
152     root = tk.Tk() #メインウィンドウ作成(この時点では空)
153     app = CSVImageViewer(root, csv_directory, image_directory)
154     root.mainloop()
```

● Pose.py

```
1 import cv2
2 import mediapipe as mp
3 import numpy as np
4 import csv
5 import os
6 import datetime
7 import tkinter as tk
8
9 # グローバル関数の設定
10 now = datetime.datetime.now()
11 csv_dir = './csv'
12 photo_dir = './photo'
13 csv_file_name = f"{now.strftime('%Y-%m-%d_%H-%M')}.csv"
14 photo_file_name = f"{now.strftime('%Y-%m-%d_%H-%M')}.png"
15 csv_file_path = os.path.join(csv_dir, csv_file_name)
16 photo_file_path = os.path.join(photo_dir, photo_file_name)
17 # ディレクトリが存在しない場合は作成
18 os.makedirs('./csv', exist_ok=True)
19 os.makedirs('./photo', exist_ok=True)
20 avg_count = 0
21 sum_neck_angle = 0
22 sum_waist_angle = 0
23 sum_score = 0
24
25
26 # Mediapipe Poseの初期化
27 mp_pose = mp.solutions.pose
28 pose = mp_pose.Pose()
29
30 # ベクトル間の角度を計算する関数
31 def calculate_angle(v1, v2):
32     dot_product = np.dot(v1, v2)
33     magnitude_v1 = np.linalg.norm(v1)
34     magnitude_v2 = np.linalg.norm(v2)
35     cosine_angle = dot_product / (magnitude_v1 * magnitude_v2)
36     angle = np.arccos(np.clip(cosine_angle, -1.0, 1.0)) # 角度の範囲を[-1, 1]に制限
37     return np.degrees(angle)
38
39 def calculate_score(neck_angle, waist_angle):
40     # 絶対値90からの角度の差
41     neck_diff = abs(neck_angle)
42     neck_penalty = max(0, neck_diff - 20) # 20°を超えた部分だけ減点
43
44     # 腰角度の計算(この部分は必要に応じて調整)
45     waist_diff = abs(waist_angle - 90)
46     waist_penalty = max(0, 90 - waist_diff)
47
48     # スコアの計算
49     total_penalty = neck_penalty + waist_penalty
50     score = max(0, 100 - total_penalty) # スコアは最低0点
51
52     return neck_penalty, waist_penalty, score
53
54 # ファイル制限、削除機能の関数
55 def manage_files(directory, max_files = 10):
56     """
57     指定されたディレクトリ内のファイルが最大数を超えた場合、古いファイルを削除
58
59     parameters:
60         directory (str): 対象のディレクトリパス
61         max_files (int): 保持するファイルの最大数
```

```

62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
"""
#ディレクトリ内のファイル一覧を取得
files = [os.path.join(directory,f) for f in os.listdir(directory) if os.path.isfile(os.path.join(directory,f))]

#ファイルが最大数を超える場合に処理
if len(files) > max_files:
    #ファイルを作成日時順ソート
    files.sort(key = os.path.getctime)

    #超過分のファイルを削除
    for file_to_delete in files[:len(files) - max_files]:
        os.remove(file_to_delete)
        print(f"Deleted: {file_to_delete}")

# カメラの起動
cap = cv2.VideoCapture(0)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)

def is_vertical_n(p1 , p2):
    return abs(p1[0] - p2[0]) < 85 # 閾値を緩めて20に設定

def is_vertical_w(p1 , p2):
    return abs(p1[0] - p2[0]) < 10 # 閾値を緩めて1に設定

# 垂直な線を上向きに描画するための関数
def draw_vertical_line(frame, point, length = 50, color = (0, 255, 0)):
    start_point = (point[0], point[1] - length) # 上向き(50px)の位置
    cv2.line(frame, start_point, point, color,2) # 描画

# カメラが開いているときに動かす制御文
while cap.isOpened():
    ret,frame = cap.read()
    if not ret:
        break
    # カメラ映像を反転
    frame = cv2.flip(frame, 1)
    height, width, _ = frame.shape

    # RGBに変換
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = pose.process(rgb_frame)

    # 座標の初期化
    ear_mid = None
    shoulder_mid = None
    hip_mid = None
    knee_mid = None

    if results.pose_landmarks:
        frame[:int(height * 0.07), :int(width * 0.13)] = (0, 0, 0) # 左上の塗りつぶし
        frame[:int(height * 0.07), int(width * 0.90):] = (0, 0, 0) # 黒で塗りつぶし(高さpx, 幅px)

        # 耳、肩、腰、膝、くるぶしの座標を取得
        landmarks = results.pose_landmarks.landmark
        ear_left = landmarks[mp_pose.PoseLandmark.LEFT_EAR]
        ear_right = landmarks[mp_pose.PoseLandmark.RIGHT_EAR]
        shoulder_left = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER]

```

```

122 shoulder_right = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER]
123 hip_left = landmarks[mp_pose.PoseLandmark.LEFT_HIP]
124 hip_right = landmarks[mp_pose.PoseLandmark.RIGHT_HIP]
125 knee_left = landmarks[mp_pose.PoseLandmark.LEFT_KNEE]
126 knee_right = landmarks[mp_pose.PoseLandmark.RIGHT_KNEE]
127
128 # 画面座標に変換
129 def to_pixel_coords(landmark):
130     return int(landmark.x * width), int(landmark.y * height)
131
132 ear_left_coords = to_pixel_coords(ear_left)
133 ear_right_coords = to_pixel_coords(ear_right)
134 shoulder_left_coords = to_pixel_coords(shoulder_left)
135 shoulder_right_coords = to_pixel_coords(shoulder_right)
136 hip_left_coords = to_pixel_coords(hip_left)
137 hip_right_coords = to_pixel_coords(hip_right)
138 knee_left_coords = to_pixel_coords(knee_left)
139 knee_right_coords = to_pixel_coords(knee_right)
140
141 # 中間点を計算
142 ear_mid =
143     (ear_left_coords[0] + ear_right_coords[0]) // 2,
144     (ear_left_coords[1] + ear_right_coords[1]) // 2,
145 )
146 shoulder_mid =
147     (shoulder_left_coords[0] + shoulder_right_coords[0]) // 2,
148     (shoulder_left_coords[1] + shoulder_right_coords[1]) // 2,
149 )
150 hip_mid =
151     (hip_left_coords[0] + hip_right_coords[0]) // 2,
152     (hip_left_coords[1] + hip_right_coords[1]) // 2,
153 )
154 knee_mid =
155     (knee_left_coords[0] + knee_right_coords[0]) // 2,
156     (knee_left_coords[1] + knee_right_coords[1]) // 2,
157 )
158
159 # 首線(肩-耳中間)と胴体線(肩-腰)の描画
160 if is_vertical_n(ear_mid, shoulder_mid):
161     cv2.line(frame, ear_mid, shoulder_mid, (0, 255, 0), 3) # 緑色(垂直)
162 else:
163     cv2.line(frame, ear_mid, shoulder_mid, (0, 0, 255), 3) # 赤色(非垂直)
164 if is_vertical_w(shoulder_mid, hip_mid):
165     cv2.line(frame, shoulder_mid, hip_mid, (0, 255, 0), 3) # 緑色(垂直)
166 else:
167     cv2.line(frame, shoulder_mid, hip_mid, (0, 0, 255), 3) # 赤色(非垂直)
168
169 # 首と腰の中間点に垂直な50pxの線を描画
170 draw_vertical_line(frame, shoulder_mid, length=50, color=(0, 255, 255)) # 首
171 draw_vertical_line(frame, hip_mid, length=50, color=(0, 255, 255)) # 腰
172
173 # 首と腰の角度を計算
174 shoulder_to_ear = (shoulder_mid[0] - ear_mid[0], shoulder_mid[1] - ear_mid[1])
175 shoulder_to_hip = (hip_mid[0] - shoulder_mid[0], hip_mid[1] - shoulder_mid[1])
176
177 neck_angle = calculate_angle(shoulder_to_ear, (0, 1)) # 首の角度(70°以上)
178 waist_angle = calculate_angle(shoulder_to_hip, (0, 1)) # 腰の角度(垂直方向に対して)
179
180 an, bn, score = calculate_score(neck_angle, waist_angle)
181

```

```

162     else:
163         cv2.line(frame, ear_mid, shoulder_mid, (0, 0, 255), 3) # 赤色(非垂直)
164         if is_vertical_w(shoulder_mid, hip_mid):
165             cv2.line(frame, shoulder_mid, hip_mid, (0, 255, 0), 3) # 緑色(垂直)
166         else:
167             cv2.line(frame, shoulder_mid, hip_mid, (0, 0, 255), 3) # 赤色(非垂直)
168
169     # 首と腰の中間点に垂直な50pxの線を描画
170     draw_vertical_line(frame, shoulder_mid, length=50, color=(0, 255, 255)) # 首
171     draw_vertical_line(frame, hip_mid, length=50, color=(0, 255, 255)) # 腰
172
173     # 首と腰の角度を計算
174     shoulder_to_ear = (shoulder_mid[0] - ear_mid[0], shoulder_mid[1] - ear_mid[1])
175     shoulder_to_hip = (hip_mid[0] - shoulder_mid[0], hip_mid[1] - shoulder_mid[1])
176
177     neck_angle = calculate_angle(shoulder_to_ear, (0, 1)) # 首の角度(70°以上)
178     waist_angle = calculate_angle(shoulder_to_hip, (0, 1)) # 腰の角度(垂直方向に対して)
179
180     an, bn, score = calculate_score(neck_angle, waist_angle)
181
182     # anの表示位置を計算(肩の左斜め上50px)
183     an_display_coords = (shoulder_mid[0] - 50, shoulder_mid[1] - 50)
184     bn_display_coords = (hip_mid[0] - 50, hip_mid[1] - 50)
185
186     # scoreの表示位置を計算(画面右上10%*10%)
187     score_display_coords = (int(width * 0.93), int(height * 0.05))
188
189
190     # 累積和に現在のフレームの角度とスコアを加算
191     avg_count += 1
192     sum_neck_angle += abs(90 - neck_angle)
193     sum_waist_angle += abs(90 - waist_angle)
194     sum_score += score
195
196     # 平均値の計算(累積和を使用)
197     avg_neck_angle = sum_neck_angle / avg_count
198     avg_waist_angle = sum_waist_angle / avg_count
199     avg_score = sum_score / avg_count
200
201
202     # テキスト描画
203     cv2.putText(frame, f"Neck Angle: {neck_angle:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
204     cv2.putText(frame, f"Waist Angle: {waist_angle:.2f}", (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)
205     cv2.putText(frame, f"(an:.0f)", an_display_coords, cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
206
207     cv2.putText(frame, f"(bn:.0f)", bn_display_coords, cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
208     cv2.putText(frame, f"(score:.0f)", score_display_coords, cv2.FONT_HERSHEY_SIMPLEX, 2.0, (0, 255, 0) if score > 50 else (0, 0, 255), 3)
209
210     # フレームを表示
211     cv2.imshow("Camera", frame)
212
213     # 'q' キーで終了
214     if cv2.waitKey(1) & 0xFF == ord('q'):
215         #画像を保存
216         cv2.imwrite(photo_file_path,frame)
217         break
218
219     #リリース解放
220     cap.release()
221     cv2.destroyAllWindows()
222
223     #最終的な平均値を計算
224     avg_neck_angle = sum_neck_angle / avg_count
225     avg_waist_angle = sum_waist_angle / avg_count
226     avg_score = sum_score / avg_count
227
228     #小数点第二位以下を丸める
229     avg_neck_angle = round(avg_neck_angle,1)
230     avg_waist_angle = round(avg_waist_angle,1)
231     avg_score = round(avg_score,1)
232
233     # csvにデータを書き込む
234     csv_op_data =[['首角度平均', '腰角度平均', '点数平均'], [avg_neck_angle, avg_waist_angle, avg_score]]
235     with open(csv_file_path, mode='w', newline='', encoding='utf-8') as f:
236         writer = csv.writer(f)
237         writer.writerows(csv_op_data)
238
239     #ファイルの数を管理する関数を呼び出し(csv写入)
240     manage_files(csv_dir, max_files=10)
241     manage_files(photo_dir, max_files=10)

```

開発環境セットアップマニュアル

必要なソフトウェアのインストール

Visual Studio Code のインストール

- Visual Studio Code の公式サイトにアクセス
- OS に適したインストーラーをダウンロードしインストール

Python のインストール

- Python の公式サイトからバージョン 3.8 以上をダウンロード
- インストーラーを実行し“Add Python PATH”にチェックを入れインストール

必要なライブラリのインストール

アプリを動作させるために、以下のライブラリをインストール

```
PS C:\Users\y23501> pip install mediapipe opencv-python Flask numpy
```

ライブラリ	バージョン
MediaPipe	0.10.14
Opencv-python	4.10.0.84
Flask	3.1.0
Numpy	2.0.0

ディレクトリ構成

```
SitSmart/
├── app.py
├── csv/
├── img/
│   ├── graph.jpg
│   └── sesuji.jpg
├── photo/
├── scripts/
│   ├── csv.py
│   └── pose.py
└── static/
    ├── css/
    │   ├── style.css
    │   ├── style_compare.css
    │   ├── style_page1.css
    │   ├── style_page2.css
    ├── images/
    │   ├── alert.png
    │   ├── average.png
    │   ├── bad.png
    │   ├── image.jpg
    └── model.png
```

```
|   |   └── score.png  
|   |   └── sisei1.png  
|   └── js/  
|       └── compare.js  
|       └── slide.js  
└── templates/  
    └── compare.html  
    └── index.html  
    └── page1.html  
    └── page2.html
```