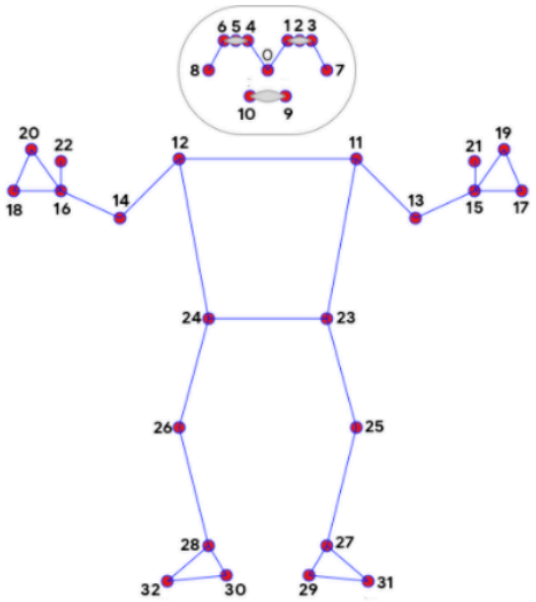
Exercise Counter using MediaPipe Human Pose Estimation and OpenCV

Mizba 20BCE1004

I. Introduction

MediaPipe

MediaPipe is an open-source machine learning framework developed by Google. It provides a suite of pre-built solutions for computer vision and machine learning tasks one of them being human pose detection. MediaPipe's pose detection solution uses a deep learning model trained on a large dataset of human pose examples to estimate the positions of key points on a human body. These key points, also known as landmarks. It detects 33 landmarks:



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

The landmarks are detected using a convolutional neural network (CNN) that has been trained on a large dataset of labelled images and videos of human poses. The CNN takes as input an image or video frame and outputs a set of confidence scores indicating the likelihood that each landmark is present at a particular location. These confidence scores are then used to estimate the final positions of the landmarks.

Usage: The input (an image or video stream) is fed to the Human Pose Estimation Model and the positions of the key points are displayed in real-time. This can detect human poses, track movements, and recognize specific gestures making it useful for applications such as fitness tracking, gaming, and virtual try-on. MediaPipe's combination of accuracy, speed, ease of use, and customizability makes it a high-performance and powerful tool.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It can be used to capture video feed from a variety of sources, such as webcams, USB cameras, and IP cameras. It provides a simple and easy-to-use interface for accessing video streams and capturing frames in real-time and a wide range of tools and algorithms for image and video processing, object detection and tracking for instance.

Exercise Counter

The project uses the Human Pose Estimation Model from MediaPipe to detect landmarks from a live camera feed. OpenCV is used for capturing the feed. Angles between the relevant joints are calculated for two exercises namely - donkey kicks and weight lifts. Base conditions and logic are built with respect the calculated angles to decide if the movement detected corresponds to the respective exercise. Angles formed by the key points are displayed in real time and a counter display is used to show the number of times the exercise has been correctly performed. OpenCV is also used for these visualisations.

II. Methodology

- Importing relevant packages and initializing MediaPipe instances
- Initializing counter and stage variables. Counter keeps count of the number of times exercise has been done correctly. Each exercise has 2 stages - task has been completed once if both stages have been reached once each.
- Capturing live camera feed using OpenCV and creating a loop of frames for which
    1. Convert BGR to RGB format (recoloring)
    2. Make detections
    3. Converting RGB to BGR (recoloring)
    4. Extract the landmarks (as per given 33 key points)
    5. Get coordinates of relevant landmarks
    6. Calculate angles between the landmarks
    7. Build logic to detect task completion
    8. Visuals on live feed: Display the counter box and value; Display the angle values in real time
    9. render the detections and display the same on live feed
    10. Close camera by hitting q

Note: The same procedure is repeated for the both exercises with relevant key points and logic.

III. Angle Calculations

Donkey Kick

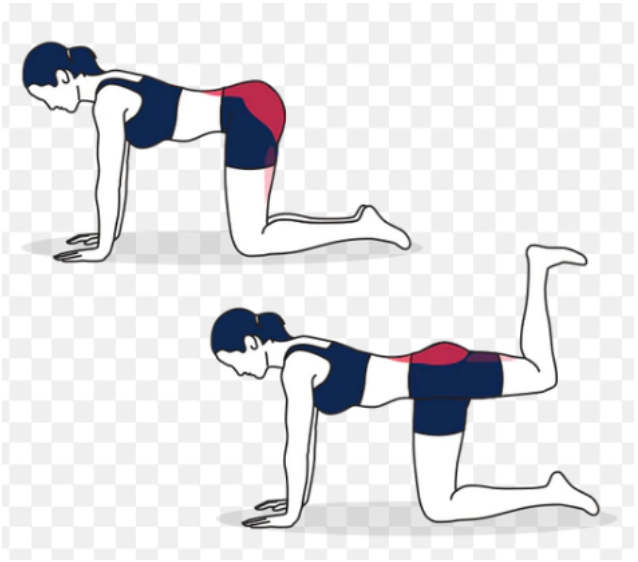


Figure 1: Donkey Kick stages (down and up)

Base conditions

Upper body:

- shoulder-elbow-wrist should form 180° (straight): keypoints - 11 13 15
- hip-shoulder-elbow should form 90° (perpendicular): keypoints - 23 11 13

Lower body:

- hip-knee-ankle should form 90° (perpendicular): keypoints - 23 25 27

Stages

- **Down:** shoulder-hip-knee form 90°: keypoints - 11 23 25
- **Up:** shoulder-hip-knee form 90°: keypoints - 11 23 25

Code Snippet of Logic

```
angle1 = calculate_angle(shoulder, elbow, wrist)
angle2 = calculate_angle(hip, shoulder, elbow)
angle3 = calculate_angle(hip, knee, ankle)
angle4 = calculate_angle(shoulder, hip, knee)
```

```
while (angle1 > 170 and angle1 < 190) and (angle2 > 70 and angle2 < 100) and (angle3 > 70 and angle3 < 100):
    if angle3 >70 and angle3 < 100:
        stage = "down"
    if angle3 > 170 and angle3 < 190 and stage=='down':
        stage = "up"
        counter+=1
        print(counter)
```

Note: A range/threshold has been provided as exact angles may not be detected at all times.

Weight Lift

Figure 2: Weight Lift (down and up)

Stages

- **Down:** shoulder-elbow-wrist: keypoints - 11 13 15 **and** hip-shoulder-elbow : keypoints - 23 11 13 should each be close to 0°
- **Up:** shoulder-elbow-wrist: 11 13 15 **and** hip-shoulder-elbow: 23 11 13 should each be close to 0°

Code Snippet of Logic

```
angle1 = calculate_angle(shoulder, elbow, wrist)
angle2 = calculate_angle(hip, shoulder, elbow)
```

```
if (angle1 > 0 and angle1 < 10) and (angle2 > 0 and angle2 < 10):
    stage = "down"
if (angle1 > 170 and angle1 < 190) and (angle2 > 170 and angle2 < 190) and stage=='down':
    stage = "up"
    counter+=1
    print(counter)
```

IV. Output Snippets