



SCHOOL OF COMPUTER SCIENCE ENGINEERING

Project Report

of

CSE3013 – Artificial Intelligence

Fall Semester 2022-23

AI BASED PERSONAL GYM TRAINER

Nithik S R
20BCE0199

Samridhi Hisaria
20BCE0792

Karthik Sunil
20BCE2144

Prasanna Kumar N
20BCE2121

TABLE OF CONTENTS:

PROBLEM STATEMENT:	3
INTRODUCTION:	3
A) Motivation:	3
B) Significance of the problem :	4
C) Scope and Applications:	4
LITERATURE SURVEY:	5
A) Related work happened so far:	7
Research Paper 1:	7
Research Paper 2:	8
Research Paper 3:	8
Research Paper 4:	9
B) Gaps Identified:	9
C) Drive to the present work:	10
IMPLEMENTATION:	11
1. Framework/Architecture/Flowchart:	11
2. Algorithm:	12
3. Complexity analysis:	13
4. Programs:	14
RESULT ANALYSIS:	37
Calculation of points using mediapipe:	37
FUTURE WORK:	40
REFERENCES:	41

PROBLEM STATEMENT:

An app that detects the user's exercise pose counts the specified exercise repetitions and gives a personalised, detailed analysis of how to improve the user's body posture and can also count a trainee's reps and give immediate feedback on the same.

It guides people who don't have access to the gym but are still willing to work out at home to maintain their physique and fitness and keep their body in good shape.

INTRODUCTION:

A) Motivation:

More than two years have passed since the beginning of the COVID pandemic, and people are still unable to leave their homes. As health and fitness are paramount considerations for everyone, it's no surprise that the gym is one of the most frequented public establishments. These unfortunate circumstances have put many people at risk for a wide range of health issues, including but not limited to: obesity, disturbed sleeping patterns, eye strain, mental stress, and lowered immunity.

Many people watch Youtube videos , wearable techs and personal gym trainers. But all of them have their disadvantages. While watching YouTube videos, no feedback is given. Most gyms have a wide variety of exercise equipment and also have trainers who guide us about the exercise and its correct posture. But the unavailability of the above equipment and trainers

can be an important reason that can stop us from doing exercise at home. So, this personal trainer can be useful.

B) Significance of the problem :

It will help people maintain their physique and to maintain their body in good shape. To help them perform the exercises correctly and prevent them from chronicle and immediate injuries. It provides user an effective and enriching workout regime.

C) Scope and Applications:

The counting of the number of repetitions of the exercises can help the user keep a track of the amount of work out performed by them. It monitors the daily exercise routine of the user and helps them remain healthy and fit.

LITERATURE SURVEY:

There are a number of fitness apps that can replace a personal trainer by showing you how to perform specific exercises. In contrast, our software makes use of computer vision to guide the user through the entire exercise process, from choosing the right move to performing the right form while standing to counting reps.

This app can be compared to a training partner due to its real-time posture analysis and nutritional recommendations. The app can be used as a smart trainer in gyms, reducing the need for human instructors if its features are expanded.

Insufficient physical activity is recognized as one of the leading risk factors for death worldwide and can lead to a variety of health issues, including cardiovascular diseases, diabetes, cancer, and mental health conditions. According to a recent Lancet Global Health report, in 2016, approximately 27.5 per cent, or more than a quarter of all adults worldwide, didn't get enough physical activity.

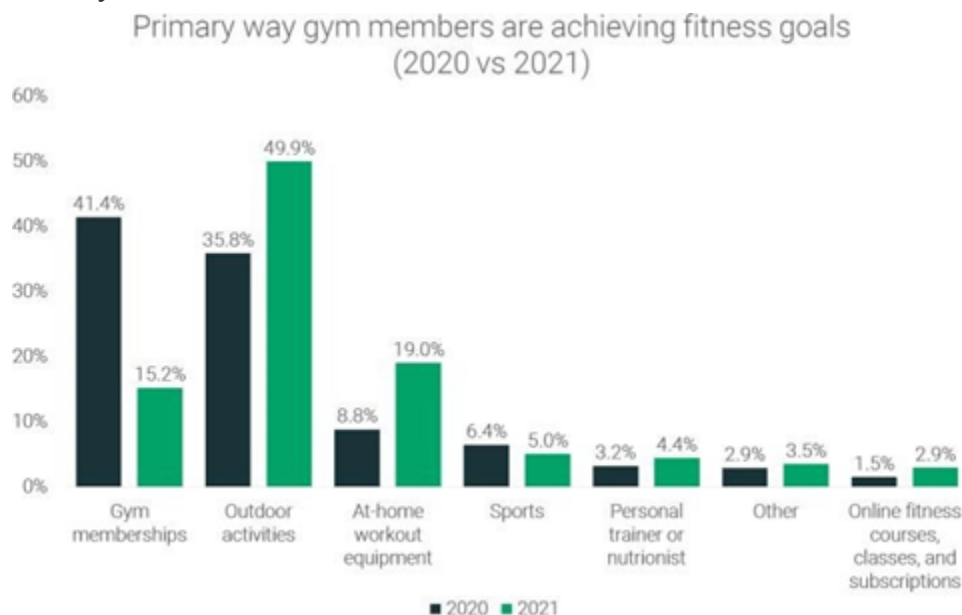
Lack of physical activity worldwide



Source: "Worldwide trends in insufficient physical activity from 2001 to 2016: a pooled analysis of 358 population-based surveys with 1·9 million participants,"
The Lancet, 2018

In recent years, people have been increasingly turning to technology to help them solve this problem, with a number of wellness, fitness, and nutrition apps and gadgets appearing on the market.

Many new fitness apps, gadgets, and wearables are being launched in the market and creating all the buzz. A recent report from Research n Reports revealed that the global worth of fitness technology was \$17.9 billion in 2019 and it is expected to grow to \$62.1 billion by the year 2025. You may be unaware, but Artificial Intelligence (AI) is assimilating deeper into our lifestyles. The fitness industry is going through a major transformation as IoT and AI are bringing innovations in fitness product offerings. Reports and Data research firm predicted that by the year 2027, the annual revenues of the fitness app market will be \$14.64 billion, with around 100.2 million active fitness users by 2024.



The primary way gym members plan to stay fit in 2021 is now outdoor activities like running, hiking, walking, cycling, etc. Having increased from 35.82% in at the beginning of 2020 to 49.92% in 2021 (a 39.4% increase).

What are gym members favouring:

- 1.4x more gym members are opting for home fitness options like at-home workout equipment (18.99%) and **online fitness courses**, classes, and subscriptions (2.93%) than gym memberships in 2021.

At least, according to a recent survey of 2,000 Americans.

72% of us are finding it easier to maintain our fitness routines now, when we can't go to the gym, than pre-Coronavirus. Almost half are using fitness apps for the first time, and 56% of people actually don't plan to buy back into their gym memberships after the current health crisis. And a staggering 80% of men are exercising more now without access to their gyms than before Covid-19, according to data from Freeletics, an AI-based fitness app with 47 million users in over 160 countries.

A) Related work happened so far:

Research Paper 1:

The purpose of this study was to create a bottom-up strategy for the task of user posture prediction and real-time segmentation utilising photos from a multi-person solution and an efficient one-shot method.

So, the idea they proposed method involved CNN, which is a convolutional neural network, and trained it to detect and classify the key points and accordingly give accurate results by studying the relative displacements and, hence, by clustering or identifying the group of different key points and studying the pose instances.

The model achieved a COCO accuracy of 0.665 using single-scale inference and 0.687 using multi-level inference. Applying the technique of component-based modelling. The training of real-time segmentation activity is dependent on the level-keypoint structure.

Research Paper 2:

The purpose of this study was to develop BlazePose, a lightweight convolutional neural network architecture for mobile devices capable of predicting human posture. During inference on a Pixel 2 smartphone, the network generates 33 body key points for a single individual at a frame rate of more than 30. This is perfect for use in real-time applications like health monitoring and speech recognition.

Their two most effective contributions are a fresh strategy for tracking people's posture and a simple neural network for predicting people's poses in different situations with minimal resources. Heat maps and regression are used in both methods to zero in on the pinpoints. Using BlazePose, which employs convolutional neural networks (CNNs) and a dataset of up to 25,000 images showing different body extremities, they developed a method to estimate the posture with high accuracy.

This model runs in super real-time on a mobile GPU and in near real-time on a mobile CPU. The provided algorithm for a 33-keypoint topology works well with BlazeFace and BlazePalm.

Research Paper 3:

This paper proposes an effective solution to the difficulty of detecting poses in a real-time setting when there are many people present. The model is trained to identify the user's input points and then classifies images according to their shared features.

In terms of accuracy and performance, this bottom-up method is top-notch, and it doesn't care how many people are in the frame.

Research Paper 4:

Through the use of a deep neural network, the findings of this article sought to pinpoint the exact locations of the dots. DNN-based estimators were presented in this method.

This allowed an efficiency in the precision to predict pose. The total efficiency increases by using this approach.

B) Gaps Identified:

Our primary goal in this project was to pinpoint a deep neural network to analyse the points. The disadvantage of OpenPose is it doesn't return any data about the depth. and also needs high computing power.

This strategy involves estimators based on DNNs being given. As a result, we were able to make more accurate predictions of poses with greater efficiency.

When compared to the previous methods we've reviewed, this one achieves 8.5% better mAP on the dataset that includes 288 frame images. The method achieves better real-time accuracy and precision. In subsequent levels of training, the earlier solutions were redefined.

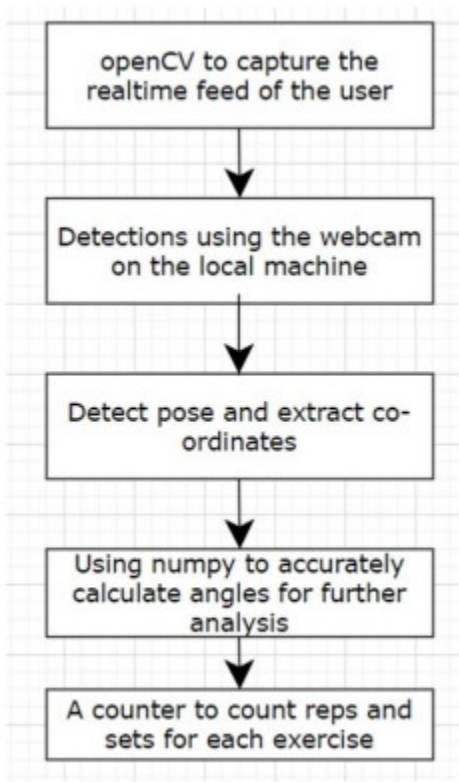
Adding it all up, we find that this method is more efficient and can save time and effort.

C) Drive to the present work:

This project focuses on supporting people to perform their workout regimens at their homes in a simple and efficient manner. Users would be able to perform their routines without using gyms. This eliminates the requirement for the assistance of a trainer or specialist in determining the proper technique for doing workouts and correcting them. This application can be run on any laptop equipped with a webcam.

IMPLEMENTATION:

1. Framework/Architecture/Flowchart:

Flowchart:	Framework/Architecture:
 <pre>graph TD; A[openCV to capture the realtime feed of the user] --> B[Detections using the webcam on the local machine]; B --> C[Detect pose and extract co-ordinates]; C --> D[Using numpy to accurately calculate angles for further analysis]; D --> E[A counter to count reps and sets for each exercise];</pre>	<ul style="list-style-type: none">• All detections done in real-time.• Make detections using the device's webcam on the local machine.• Extracting the co-ordinates of the joint from the detected pose.• Calculating angles between the detected points using libraries.• A counter which will help keep track of number of repetitions and the number of sets.• Setting up media-pipe for Python which includes all the models required for the different pose detections• OpenCV to be able to capture the user's movements in real-time to be used for further analysis.• Numpy (another python library) to calculate angles between the joints for a further analysis.

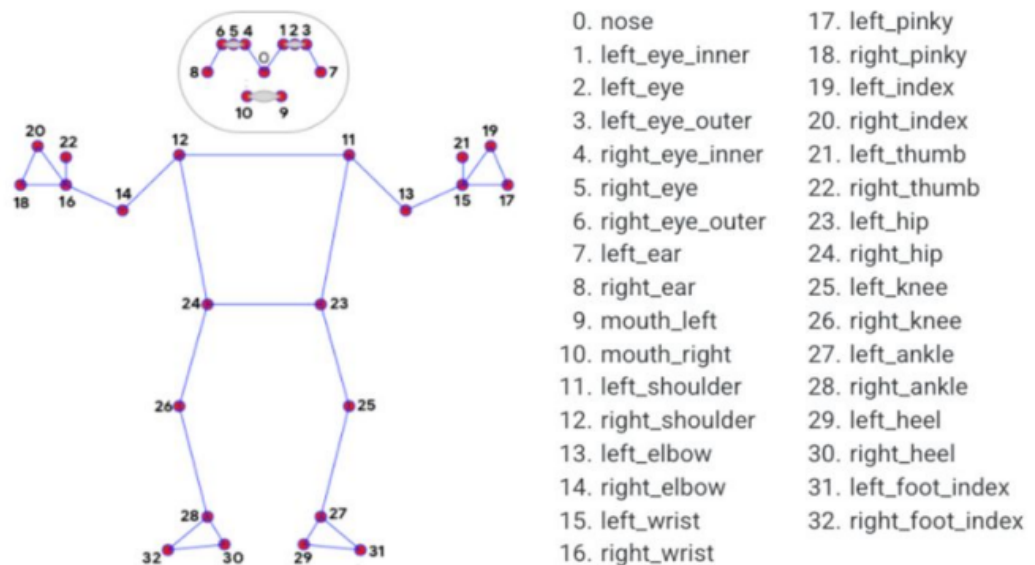
2. Algorithm:

- We have used Jupyter notebook(python language) and different libraries such as Open CV and MediaPipe which is a library using ML algorithms along with different numerical and algorithms.
- The MediaPipe pose estimation tool uses a 33 key points approach wherein it detects the key points and accordingly uses and studying the data set estimates the pose. It tracks The pose from the real-time camera frame or RGB video by using the blaze pose tool that has a Machine Learning approach in pose detection.
- This approach uses a double step tracker machine learning pipeline which is efficient in media pipe solutions. Using the tracker locates the region of interest of the activity or posture in the real-time video. It then predicts the key points in the region of interest using the real-time video frame as an input. But the point to be noted is that the tracker is invoked only during the start or whenever the model is unable to detect the body key points in the frame.
- We have created a module named PoseModule.js and defined various functions in it and imported this module to our main project file aiTrainer.js to utilize these functions. We are basically first detecting the landmark positions on the body in the video with the help of MediaPipe[9]. Then the angle between the points is calculated and a range is determined. This range

can be demonstrated by a 0-100 % efficiency bar on the output video frame. We also calculate the number of repetitions of the exercise and display the count in the output video.

- In the output following data is displayed: fps rate, counter for repetitions, landmark points, the angle between landmark points and status bar.

3. Complexity analysis:



Pose estimator has the ability to detect the pose and count the repetitions along with the posture guide. Personalised calorie counter depending upon the exercises performed. Diet planners exhibit different diets depending upon the health conditions and calorie intake.

A platform to display different health insurances and policies provided by the Indian government along with the benefits and

eligibility criteria. Display different exercise routines according to the health conditions and focus majorly on being fit and weight loss.

4. Programs:

```
!pip install mediapipe
```

```
import cv2
```

```
import mediapipe as mp
```

```
import numpy as np
```

```
mp_drawing = mp.solutions.drawing_utils
```

```
mp_pose = mp.solutions.pose
```

```
#the following block is for capturing the video in real time and for detection  
of points using mediapipe and displays
```

```
#the coordinates of the points
```

```
cap = cv2.VideoCapture(0)
```

```
with mp_pose.Pose(min_detection_confidence=0.5,  
min_tracking_confidence=0.5) as pose:
```

```
while cap.isOpened():
```

```
    ret, frame = cap.read()
```

```
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
image.flags.writeable = False
```

```
results = pose.process(image)
```

```
image.flags.writeable = True
```

```
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

```
try:
```

```
    landmarks = results.pose_landmarks.landmark
```

```
    print(landmarks)
```

```
except:
```

```
    pass
```

```
    mp_drawing.draw_landmarks(image, results.pose_landmarks,  
mp_pose.POSE_CONNECTIONS,
```

```
        mp_drawing.DrawingSpec(color=(245,117,66),  
thickness=2, circle_radius=2),
```

```
        mp_drawing.DrawingSpec(color=(245,66,230),  
thickness=2, circle_radius=2)
```

```
)
```

```
cv2.imshow('Mediapipe Feed', image)
```

```
if cv2.waitKey(10) & 0xFF == ord('q'):
```

```
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

```
len(landmarks)
```

```
for lndmrk in mp_pose.PoseLandmark:
```

```
    print(lndmrk)
```

```
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].visibility
```

```
landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
```

```
landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
```

```
def calculate_angle(a,b,c):
```

```
    a = np.array(a) # First
```

```
    b = np.array(b) # Mid
```

```
    c = np.array(c) # End
```



```
radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
```

```
angle = np.abs(radians*180.0/np.pi)
```

```
if angle >180.0:
```

```
    angle = 360-angle
```

```
return angle
```

```
shoulder =
```

```
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[  
mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
```

```
elbow =
```

```
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_  
pose.PoseLandmark.LEFT_ELBOW.value].y]
```

```
wrist =
```

```
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_p  
ose.PoseLandmark.LEFT_WRIST.value].y]
```

```
shoulder, elbow, wrist
```

```
calculate_angle(shoulder, elbow, wrist)
```

```
tuple(np.multiply(elbow, [640, 480]).astype(int))
```

```
#to calculate and estimate the angle between the points
```

```
cap = cv2.VideoCapture(0)

## Setup mediapipe instance

with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:

    while cap.isOpened():

        ret, frame = cap.read()

        # Recolor image to RGB

        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        image.flags.writeable = False

        # Make detection

        results = pose.process(image)

        # Recolor back to BGR

        image.flags.writeable = True

        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Extract landmarks

        try:

            landmarks = results.pose_landmarks.landmark

            # Get coordinates
```

```
    shoulder =  
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[  
mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
```

```
    elbow =  
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_  
pose.PoseLandmark.LEFT_ELBOW.value].y]
```

```
    wrist =  
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_p  
ose.PoseLandmark.LEFT_WRIST.value].y]
```

```
# Calculate angle
```

```
angle = calculate_angle(shoulder, elbow, wrist)
```

```
# Visualize angle
```

```
cv2.putText(image, str(angle),  
            tuple(np.multiply(elbow, [640, 480]).astype(int)),  
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,  
cv2.LINE_AA  
            )
```

```
except:
```

```
    pass
```

```
# Render detections
```

```
    mp_drawing.draw_landmarks(image, results.pose_landmarks,  
mp_pose.POSE_CONNECTIONS,
```

```
        mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),

        mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)

    )
```

```
cv2.imshow('Mediapipe Feed', image)
```

```
if cv2.waitKey(10) & 0xFF == ord('q'):
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

```
#code to count and display the number of reps and print them
```

```
cap = cv2.VideoCapture(0)
```

```
# Curl counter variables
```

```
counter = 0
```

```
stage = None
```

```
## Setup mediapipe instance
```

```
with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:
```

```
    while cap.isOpened():
```

```
ret, frame = cap.read()
```

```
# Recolor image to RGB
```

```
image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
image.flags.writeable = False
```

```
# Make detection
```

```
results = pose.process(image)
```

```
# Recolor back to BGR
```

```
image.flags.writeable = True
```

```
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

```
# Extract landmarks
```

```
try:
```

```
    landmarks = results.pose_landmarks.landmark
```

```
    # Get coordinates
```

```
    shoulder =
```

```
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x, landmarks[  
mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
```

```
    elbow =
```

```
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x, landmarks[mp_  
pose.PoseLandmark.LEFT_ELBOW.value].y]
```

```
wrist =  
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_p  
ose.PoseLandmark.LEFT_WRIST.value].y]
```

```
# Calculate angle
```

```
angle = calculate_angle(shoulder, elbow, wrist)
```

```
# Visualize angle
```

```
cv2.putText(image, str(angle),  
             tuple(np.multiply(elbow, [640, 480]).astype(int)),  
             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,  
cv2.LINE_AA  
             )
```

```
# Curl counter logic
```

```
if angle > 160:
```

```
    stage = "down"
```

```
if angle < 30 and stage == 'down':
```

```
    stage="up"
```

```
    counter +=1
```

```
    print(counter)
```

```
except:
```

```
    pass
```

```
# Render curl counter
```

```
# Setup status box
```

```
cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)
```

```
# Rep data
```

```
cv2.putText(image, 'REPS', (15,12),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
```

```
cv2.putText(image, str(counter),
```

```
(10,60),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)
```

```
# Stage data
```

```
cv2.putText(image, 'STAGE', (65,12),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
```

```
cv2.putText(image, stage,
```

```
(60,60),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)
```

```
# Render detections
```

```
mp_drawing.draw_landmarks(image, results.pose_landmarks,  
mp_pose.POSE_CONNECTIONS,
```

```
mp_drawing.DrawingSpec(color=(245,117,66),  
thickness=2, circle_radius=2),
```

```
        mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)
    )
```

```
cv2.imshow('Mediapipe Feed', image)
```

```
if cv2.waitKey(10) & 0xFF == ord('q'):
```

```
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

```
shoulder1 =
```

```
[landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,landmarks[
mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y]
```

```
elbow1 =
```

```
[landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,landmarks[mp
_pose.PoseLandmark.RIGHT_ELBOW.value].y]
```

```
wrist1 =
```

```
[landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].x,landmarks[mp_
pose.PoseLandmark.RIGHT_WRIST.value].y]
```

```
#code to count and display the number of reps and print them
```

```
cap = cv2.VideoCapture(0)
```

```
# Curl counter variables
```



```
counter = 0
```

```
stage = None
```

```
## Setup mediapipe instance
```

```
with mp_pose.Pose(min_detection_confidence=0.5,  
min_tracking_confidence=0.5) as pose:
```

```
while cap.isOpened():
```

```
    ret, frame = cap.read()
```

```
    # Recolor image to RGB
```

```
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
    image.flags.writeable = False
```

```
    # Make detection
```

```
    results = pose.process(image)
```

```
    # Recolor back to BGR
```

```
    image.flags.writeable = True
```

```
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

```
    # Extract landmarks
```

```
    try:
```

```
        landmarks = results.pose_landmarks.landmark
```

```
# Get coordinates

shoulder1 =
[landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,landmarks[
mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y]

elbow1 =
[landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,landmarks[mp
_pose.PoseLandmark.RIGHT_ELBOW.value].y]

wrist1 =
[landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].x,landmarks[mp_
pose.PoseLandmark.RIGHT_WRIST.value].y]
```

```
# Calculate angle
```

```
angle1 = calculate_angle(shoulder1, elbow1, wrist1)
```

```
# Visualize angle
```

```
cv2.putText(image, str(angle1),
            tuple(np.multiply(elbow1, [640, 480]).astype(int)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
            )
```

```
# Curl counter logic
```

```
if angle1 > 160:
```

```
    stage = "down"
```

```
if angle1 < 30 and stage == 'down':
```

```

        stage="up"

        counter +=1

        print(counter)

    except:

        pass

    # Render curl counter

    # Setup status box

    cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)


    # Rep data

    cv2.putText(image, 'REPS', (15,12),

                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

    cv2.putText(image, str(counter),

                (10,60),

                cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)


    # Stage data

    cv2.putText(image, 'STAGE', (65,12),

                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

    cv2.putText(image, stage,

                (60,60),

                cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

```

```

        # Render detections

        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,

                                mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),

                                mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)

                                )

cv2.imshow('Mediapipe Feed', image)

if cv2.waitKey(10) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

#code to count and display the number of reps and print them
#left wrist exercise
cap = cv2.VideoCapture(0)

# Curl counter variables
counter = 0

```

stage = None

Setup mediapipe instance

*with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:*

while cap.isOpened():

ret, frame = cap.read()

Recolor image to RGB

image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

image.flags.writeable = False

Make detection

results = pose.process(image)

Recolor back to BGR

image.flags.writeable = True

image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

Extract landmarks

try:

landmarks = results.pose_landmarks.landmark

```
index1 =  
[landmarks[mp_pose.PoseLandmark.LEFT_INDEX.value].x,landmarks[mp_p  
ose.PoseLandmark.LEFT_INDEX.value].y]
```

```
elbow1 =  
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_  
pose.PoseLandmark.LEFT_ELBOW.value].y]
```

```
wrist1 =  
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_p  
ose.PoseLandmark.LEFT_WRIST.value].y]
```

```
# Calculate angle
```

```
angle = calculate_angle(elbow1, index1, wrist1)
```

```
# Visualize angle
```

```
cv2.putText(image, str(angle),  
             tuple(np.multiply(wrist1, [640, 480]).astype(int)),  
             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,  
cv2.LINE_AA  
            )
```

```
# Curl counter logic
```

```
if angle > 40:
```

```
    stage = "up"
```

```
if angle < 30 and stage == 'up':
```

```
    stage="down"
```

```
    counter +=1
```

```
    print(counter)
```

except:

pass

Render curl counter

Setup status box

cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)

Rep data

cv2.putText(image, 'REPS', (15,12),

cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

cv2.putText(image, str(counter),

(10,60),

cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

Stage data

cv2.putText(image, 'STAGE', (65,12),

cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

cv2.putText(image, stage,

(60,60),

cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

Render detections

```
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,

                                mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),

                                mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)

                                )
```

```
cv2.imshow('Mediapipe Feed', image)
```

```
if cv2.waitKey(10) & 0xFF == ord('q'):
```

```
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

```
#code to count and display the number of reps and print them
```

```
#right wrist exercise
```

```
cap = cv2.VideoCapture(0)
```

```
# Curl counter variables
```

```
counter = 0
```

```
stage = None
```

```
## Setup mediapipe instance
```


*with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:*

while cap.isOpened():

ret, frame = cap.read()

Recolor image to RGB

image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

image.flags.writeable = False

Make detection

results = pose.process(image)

Recolor back to BGR

image.flags.writeable = True

image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

Extract landmarks

try:

landmarks = results.pose_landmarks.landmark

index2 =

*[landmarks[mp_pose.PoseLandmark.RIGHT_INDEX.value].x,landmarks[mp_
pose.PoseLandmark.RIGHT_INDEX.value].y]*

elbow2 =

*[landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,landmarks[mp_
_pose.PoseLandmark.RIGHT_ELBOW.value].y]*

```
wrist2 =  
[landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].x,landmarks[mp_  
pose.PoseLandmark.RIGHT_WRIST.value].y]
```

```
# Calculate angle
```

```
angle2 = calculate_angle(elbow2, index2, wrist2)
```

```
# Visualize angle
```

```
cv2.putText(image, str(angle2),  
             tuple(np.multiply(wrist2, [640, 480]).astype(int)),  
             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,  
             cv2.LINE_AA  
             )
```

```
# Curl counter logic
```

```
if angle2 > 40:
```

```
    stage = "up"
```

```
if angle2 < 30 and stage == 'up':
```

```
    stage="down"
```

```
    counter +=1
```

```
    print(counter)
```

```
except:
```

```
    pass
```

```
# Render curl counter
```

Setup status box

cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)

Rep data

cv2.putText(image, 'REPS', (15,12),

cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

cv2.putText(image, str(counter),

(10,60),

cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

Stage data

cv2.putText(image, 'STAGE', (65,12),

cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

cv2.putText(image, stage,

(60,60),

cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

Render detections

*mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,*

*mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),*

*mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)*

)

cv2.imshow('Mediapipe Feed', image)

if cv2.waitKey(10) & 0xFF == ord('q'):

break

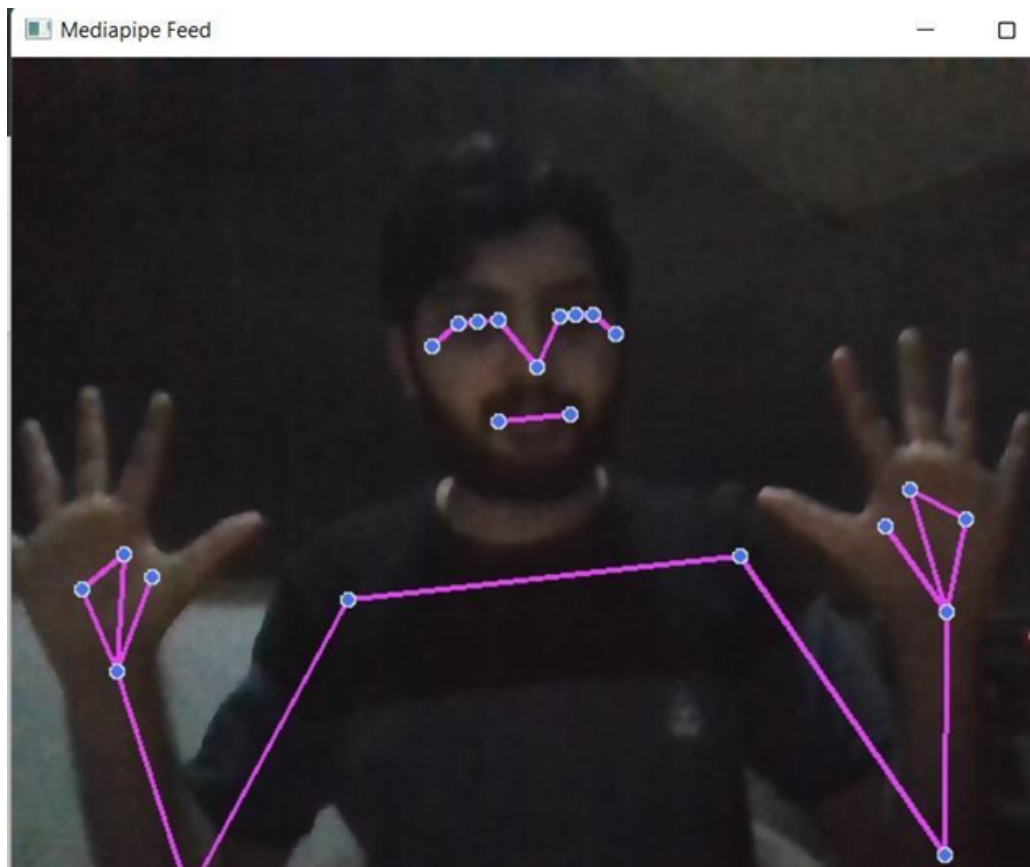
cap.release()

cv2.destroyAllWindows()

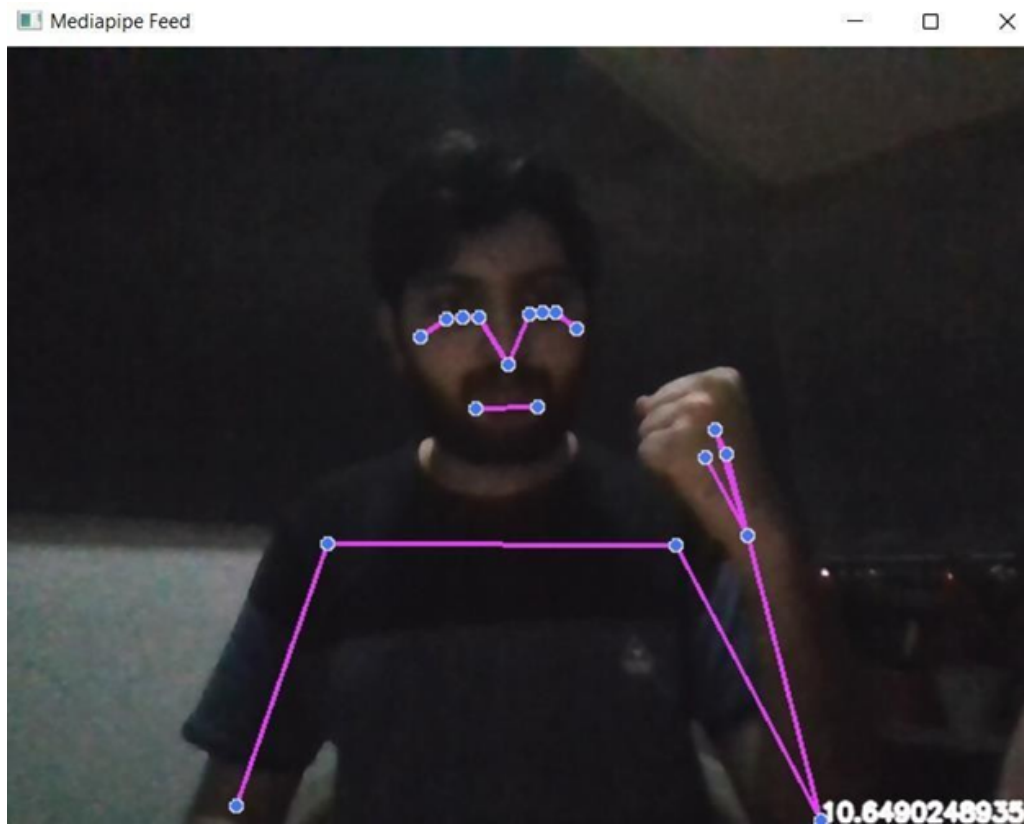
RESULT ANALYSIS:

No dataset was used by us in this project. However, we used the mediapipe library which provided us with pre trained models enabling us to calculate and estimate poses with ease.

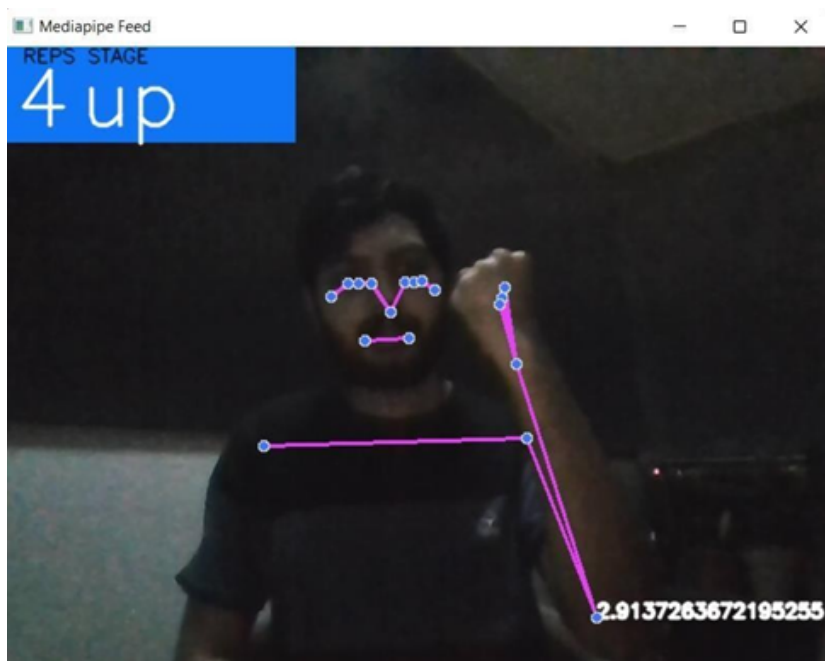
Calculation of points using mediapipe:



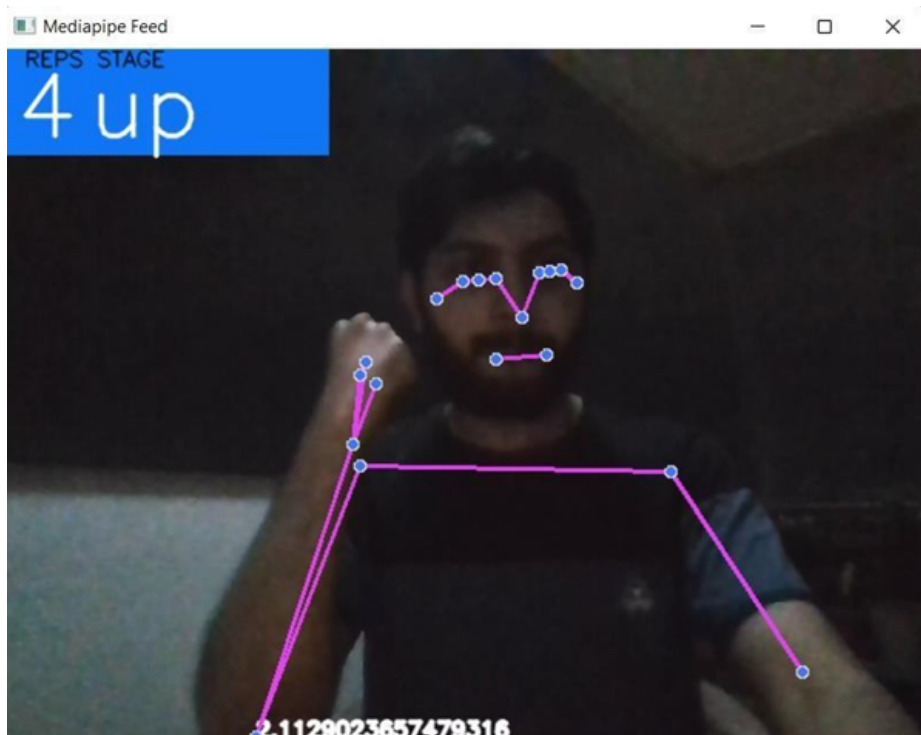
Calculation of angle between 3 random points



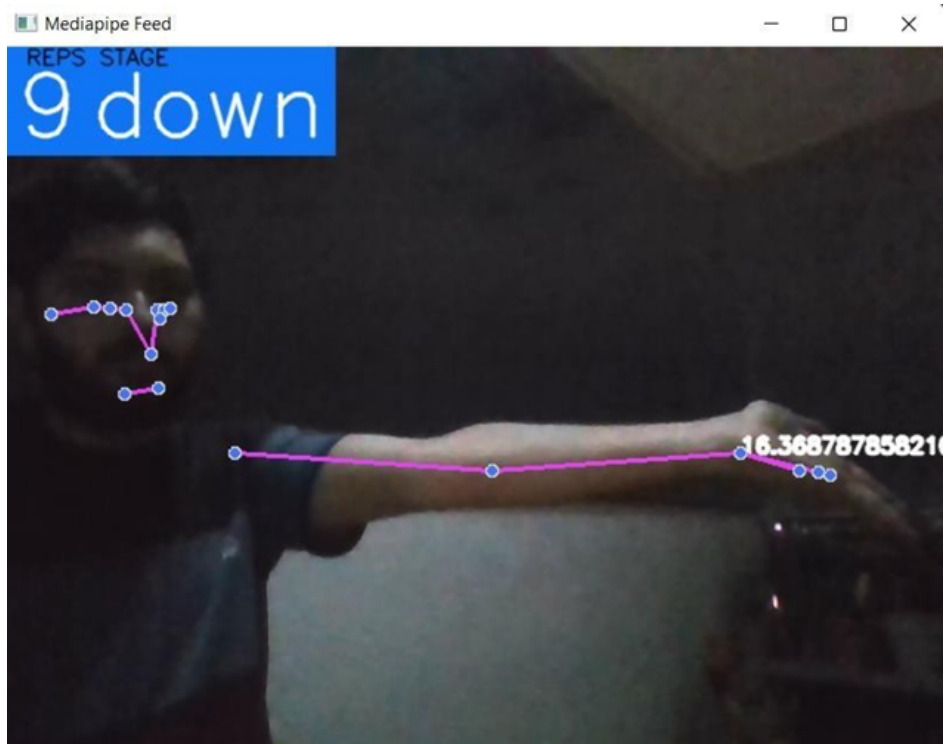
Left bicep curl



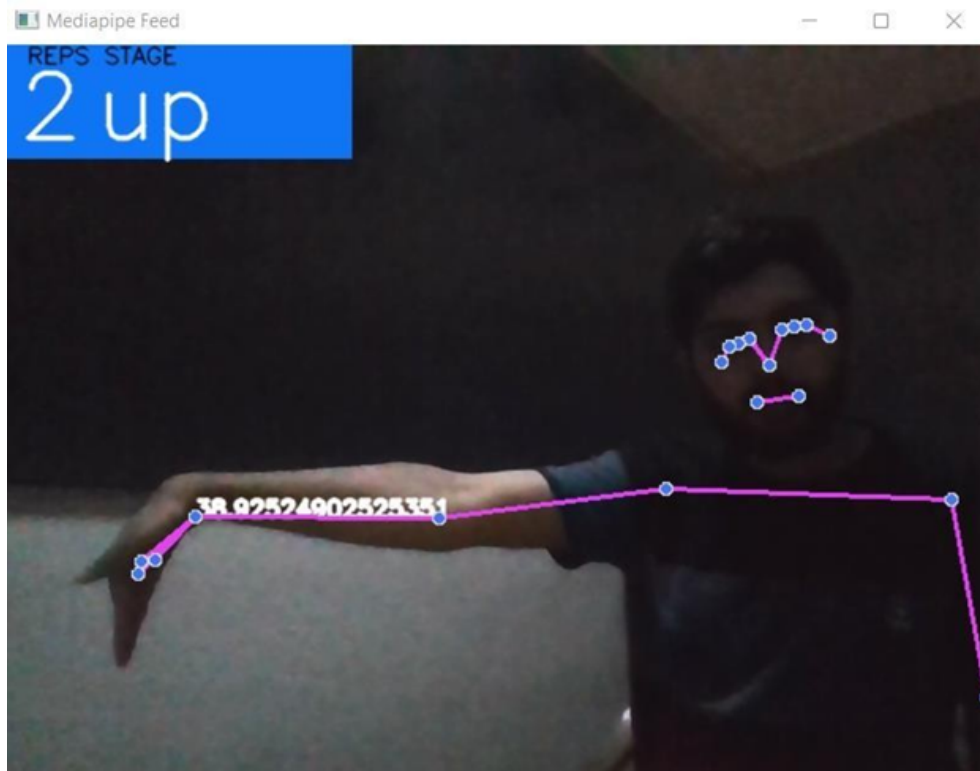
Right bicep curl



Left wrist rotation



Right wrist rotation



FUTURE WORK:

There is a lot of scope of development in this project. The project can be upgraded to support more exercises. A User interface can be added for easy navigation through the exercises. The data collected by the AI trainer can be saved and processed for the next sessions. Daily steps tracker can also be added. The trainer will suggest you workout plan and its intensity according to your body type and weight. This application can be developed into a complete android/ios application for ease of use.

Future work may include adjusting the camera's height and width to capture additional exercises, or employing additional cameras to capture the body's position from a variety of perspectives for use as a template in subsequent exercises.

REFERENCES:

MediaPipe Hands: On-device Real-time Hand Tracking." F.Zhang, V.Bazarevsky, A.Vakunov, A.Tkachenka, G.Sung, C.L. Chang, M.Grundmann.

"Robust 3d hand pose estimation in single depth images: from single-view CNN to multi-view CNNs" by L.Ge, H.Liang, J.Yuan, and D.Thalmann. IEEE conference on computer vision and pattern recognition, 2016.

"Robust articulated-icp for real-time hand tracking" by A.Tagliasacchi, M.Schroder, A.Tkach, S.Bouaziz, M.Botsch, and M.Pauly. In Computer Graphics Forum, volume 34 Wiley Online Library, 2015.

"PersonLab: Person Pose Estimation & Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model" G.Papandreou, T.Zhu, L.-C.Chen, S.Gidaris, J.Tompson K.Murphy.

"Deep Learning-based Human Pose Estimation using OpenCV" By V Gupta.

"Feature pyramid networks for object detection" by T Yi Lin, P Dollar, R . Girshick, K He, 'B Hariharan, and S J Belongie. CoRR, abs/1612.03144, 2016.