PosePilot

A Power Lifter’s Form Analyzer

Chau Le (1668938)

Individual assignment
Minor Data-Driven Decision Making

Table of Contents

Introduction

With the current rise in people's interest in fitness and weightlifting, there is a lack of accessible tools for beginners to maintain a proper form in training. Some existing applications offer this, however with low accuracy and low level of details in the feedback. Other options, such as professional personal trainers or physical therapists, are more costly and less accessible to the public, especially to beginners. Therefore, by utilizing the current advancement in machine learning, this project develops a solution to this problem. It introduces PosePilot, an exercise form analyzer that utilizes MediaPose (a pose estimation framework) to analyze exercise forms based on the joint angles of the person performing the exercise and give detailed feedback accordingly.

Background of the Project

Nowadays, more and more young people are putting an increasing emphasis on fitness, one of the most common and accessible way to do this is by going to the gym (Naglazas, 2023). As people spend more time at the gym, most gym-goers are exposed to and focus more on the "big three" of weightlifting, squat, bench press, and deadlift (Top 3 Powerlifting Exercises: Squat, Bench Press & Deadlift | Flex Blog, n.d.). Undoubtedly, the importance of maintaining a proper form when performing these exercises cannot overstated. Performing these weightlifting exercises with poor technique and form is one of the primary cause for undue strain on the joints, ligaments, and muscles of the athlete, which will significantly increase the risk of injury (Bukhary et al., 2023). Traditionally, weightlifting form analysis requires trainings from professionals, such as weight lift personal trainers or physical therapists, which can be costly and inaccessible for the public. However, with the current fast-paced development of machine learning technologies, this process can be attainable without the oversight form professionals, making it available for a broader range of audiences.

This project addresses this gap by utilizing MediaPipe, an open-source framework developed by Google, to analyze the form for the three main weightlifting exercises mentioned above. This is a pose estimation framework, provided with functions to detect and track the different joints on the human body, which this project leverages to calculate the important joint angles and provide feedback accordingly. This is then combined with a graphical user interface (GUI) to produce a beginner-friendly solution for new gym-goers without the need for professionals.

Current situation

As the attention for fitness rises, there also comes multiple applications and tools to support the public's fitness journey. Some examples are MyFitnessPal (a diet and nutrition tracking application), Apple Fitness+ (an application that offers workout classes organized by professional trainers), Peloton (an application with interactive workout classes) and VimeoMove (an application that tracks body movement and gives feedback using body movement tracking). However, there are very limited applications that have the tools to analyze and give feedback on weightlifting form. If there are, such application (e.g. VimeoMove mentioned above) often have low accuracy and limited feedback details in form analysis, given that most of them rely on basic video analysis and movement sensors.

Goal

The project aims to develop an automatic form analyzer as an easily accessible solution for weightlifting beginners to improve their form in the main three exercises and reduce their risk of injury. This application aims to perform the following tasks:

- Process video inputs from users.
- Perform pose estimation in real time.
- Classify the type of exercise being performed in the video inputs (i.e. squat, deadlift or bench press).
- Provide comprehensive feedback accordingly.
- Create an intuitive, user-friendly graphical user interface (GUI) for seamless video uploads, view angle selection, and feedback.

Literature review

The danger of performing weightlifting exercises with bad form

When starting the weightlifting journey, many people make the mistake of putting too much priority on the weight on the bar and underestimate the importance of maintaining a proper form. Doing this can lead to serious issues, some of which are:

- When performing squats, two common mistakes are rounding the back and letting the knees cave inward. The prior can put excessive strain on the spinal discs and eventually lead to back injuries like disc herniation, and the latter can put stress on the knee joints, which in the long term can cause tears, sprains, or tendinitis (Austin and Mann, 2022).
- When performing deadlifts, rounding the back is also a common mistake and can potentially cause disc herniation and other back injuries (Кириченко, 2023).
- When performing bench presses, a common mistake is flaring the elbows out too wide, which potentially leads to shoulder impingement over time, or rotator cuff tears (Austin and Mann, 2022).

The proper squat, deadlift, and bench press form based on joint angles

Squat

In a squat, the hip crease should be below the top of the knees, which means the hip flexion angle should be greater than 90 degrees. Additionally, the thighs should be at least parallel to the floor, meaning the knee flexion angle needs to be over 90 degrees. These angles provide multiple benefits, such as maintaining a full range of motion, optimal engagement of the gluteal muscles, hamstrings, and adductors, reducing shear forces, and improving leverages (Newby et al., 2014). Lastly, when squatting, the torso should remain uptight with an acceptable forward lean from 30 to 50 degrees from vertical. This ensures a proper spinal alignment and even weight distribution across the joints (hips, knees, and ankles), reduces excessive strain on the lifter's lower back, and optimizes the leverage and force production (Branni, 2018; Fekete et al., 2011).

Deadlift

When performing a deadlift, the hip-knee-ankle should be over 160 degrees. This wide hip angle ensures the hips descend low enough to maintain a flat back and vertical flat back. Angles less than 160 degrees can compromise the leverages and shift the weight excessively onto the lifter's lower back. These impacts can overall increase the risk of injury and reduce the effectiveness of the deadlift (Newby et al., 2014). Additionally, throughout the deadlift, the lifter's knees should remain approximately hip-width apart. The inward collapse of the knees, so called knee calgus, can put excessive strain on the knee joints and compromise the stability of the lifter (Sutthiprapa et al., 2017).

Bench press

During a bench press, the lifter's elbows should form an angle around 90 degrees. This allows for optimal leverage along with optimal force production from the pectorals and triceps. Additionally, it also helps remain retracted and depressed shoulders, which helps reduce excessive shoulder strain. A 90-degree shoulder angle also ensures that the bar travels in a straight vertical line (Huang et al., 2014; Fioranelli and Lee, 2008). One additional requirement for a proper bench press form is to maintain a fixed distance between the shoulders throughout the movement (Chowdhury et al., 2023).

MediaPipe and other pose estimation techniques

MediaPipe Pose

MediaPipe is an open source framework developed by Google for the purpose of processing multimedia. Within this framework, there is MediaPipe Pose, a specific solution developed to perform accurate and efficient high-fidelity human body pose estimation and tracking on video input. This solution utilize the BlazePose research model to infer 33 3D human body landmarks from RGP video frames while also inferring a background segmentation mask in real time. It works by a two-step detector-tracker pipeline, starting with the person/pose region-of-interest (ROI) detection and proceeds to the prediction of the pose landmarks and segmentation mask within said ROI (Google-Ai-Edge, n.d.)

Other pose estimation techniques

Other commonly used powerful tools for real-time human pose estimation include TensorFlow Pose and OpenPose. TensorFlow Pose is a framework within the TensorFlow machine learning library. It identifies key body joints by deploying deep learning techniques like concolutional neural networks (CNNs). However, this framework offers a lower-level interface compared to MediaPipe Pose due to its need for manual configuration and customization (CodeTrade, 2023). On the other hand, OpenPose uses the Caffe deep learning framework for neural network implementation to identify human body, foot, hand, and facial key points. Developed in Carnegie Mellon University, it can detect key points on single persons in 3D, multiple persons in 2D using a calibration toolkit (Beyond Poses: Openpose Vs Mediapipe for Dynamic Vision, n.d.)

K-means Clustering

K-mean clustering is an unsupervised machine learning algorithm, used to split a dataset into a K number of clusters, with K being a predefined value. The algorithm starts with K initial cluster centers and iterates between assigning data points to the nearest cluster center and computing the cluster centers as the means of the data points in the according cluster (Bishop and Nasrabadi, 2006).

Main and sub-research questions

Main research question

How to utilize a pose estimation algorithm (MediaPipe) to analyze weightlifting exercises form and give comprehensive, insightful feedback to the users?

Sub-research questions

- How to effectively capture and analyze joint coordinates from video inputs?
- How to accurately identify the type of exercise (squat or deadlift or bench press) using joint coordinate data from the video input?
- What are the main factors to decide proper or improper form to perform each exercise?
- How to generate detailed, insightful feedback?
- How to create an intuitive, user-friendly graphical user interface (GUI)?

Methodology

Necessary libraries and dependencies

The application uses the following libraries and frameworks:

- OpenCV for capturing, processing, and visualizing the video inputs.
- MediaPipe (MediaPipe Pose) for detecting key landmarks on the human body from the video inputs.
- NumPy for computing joint angles and data manipulation.
- Tkinter for creating the graphical user interface (GUI).
- Scikit-learn for K-means clustering to classify exercise types.

Video processing

A function is defined to process the video input. After the upload, the video undergoes frame extraction and resizing to reduce processing time and resource usage. The video is then converted from BGP to RGB as a requirement for MediaPipe.

Joint angles detection and calculation

MediaPipe Pose is deployed to detect body landmarks on the human body across the frames from extracted from the video input. With each landmark (joints) detected, their coordinates are extracted (as x, y, z) and appended into a list. Lines are then drawn to connect these landmarks using MediaPipe's drawing utilities. The processing of the frames from the video input is displayed in a window called "Video Processing", which was defined so that users can stop by pressing the 'q' key.

A function is defined to calculate the joint angles, using the coordinates x, y, z above as the input, naming them a, b, and c. The angle formed at point b by the lines connecting a to b and b to c is calculated. This was done using the arctangent of the differences in t and x coordinates, which returns the angle in radians, and is then converted to degrees. The angle is then normalized to ensure it is within 0 and 180 degrees.

Joint angles analysis for each exercises

For all exercises, the analysis function initializes empty lists to store feedback for different sets of joints angles before extracting the coordinates of the corresponding joints. This set of angles also change depending on the different view angle. Each exercises are looked at from two different group of view angles. After processing the frames, the function for analyzing all the exercises consolidate the feedback by selecting the most common observations for each joints.

Squat

The squat analysis function gives feedback for the hip flexion, knee flexion, ankle dorsiflexion, and torso angle using the coordinates of the hip, knee, ankle, shoulder, and foot. These joints give the following angles: hip-knee-shoulder, hip-knee-ankle, hip-knee-foot, and a point below the shoulder to the shoulder and hip. For the front-facing views, the function calculates the distance between the knees to check if the lifter's knees are caving inwards. For all the views except front view, the function considers if the hip flexion (hip-knee) is lower than 90 degrees, the knee flexion (knee angle) is higher than 90 degrees, the ankle dorsiflexion is between 25 and 30 degrees, and if the torso is up right (the angle of a point below the shoulder to the shoulder and hip is between 30 and 50 degrees).

Deadlift

For the front-facing views, the deadlift analysis function considers the distance between the knees to check if the lifter's knees are caving inwards. And for the rest of the views, it checks if the hip-knee angle is over 160 degrees.

Bench press

For the front-facing views, the bench press analysis function considers the distance between the left and shoulder to check if the lifter's shoulders are fixed. And for the rest of the views, it checks if the shoulder-elbow angle is over 90 degrees.

Exercise types classification using K-means clustering

The exercise classification function works by analyzing the joint coordinates extracted from the video frames. For each frame, it calculates the angle between the shoulder, elbow, and wrist, along with the angle between the hip, knee, and ankle before storing them in a list, which is then converted into a NumPy array. This array is then used to train a K-means clustering model with K=3, corresponding to the three types of exercises, squat, deadlift, and bench. The function then prints the counts for the occurrences of each exercise, and returns the exercise with the highest count.

Video analysis and feedback generation

The video analysis function processes the input video, analyzes the identified exercise, and gives feedback accordingly. If no exercise is recognized, it returns an error indicating an unknown exercise.

Graphical User Interface (GUI) setup

The GUI includes a video file selection, view angle definitions and selection, and feedback display. The layout for this GUI consists of the main window "Exercise Form Analyzer" with a light pink background (personal preference), which was achieved through multiple custom styles for the frame, labels, buttons, entry fields, and combo boxes. This main window contains a bold title label at the top, a label with definitions for all six view angles (front, side, 45-degree front-right, 45-degree front-left, 45-degree behind-left, 45-degree behind-right), entry fields, buttons for selecting a video, and a dropdown menu to select the view angle. At the bottom, the "Analyze" button triggers the video analysis, and after the analysis finishes, the feedback is displayed in labels at the bottom of the window.

Results Analysis

The "PosePilot: A Power Lifter's Form Analyzer" project is a tool used to analyze exercise forms from video input using machine learning techniques. Utilizing MediaPipe Pose for pose estimation, the final result shows high accuracy in detecting body landmarks in all three main powerlifting exercises while successfully classifying exercises with high reliability. Most importantly for the users, this tool provides relevant and comprehensive feedback for all exercises, squat, deadlift, and bench press, which aligns with professional guidelines from literature. The feedback highlights common form issues, making it valuable for powerlifting beginners without requiring costly solutions or supervision from professionals. The graphical user interface (GUI) is also user-friendly and intuitive with specific definitions of view angles for users to select, allowing seamless selection and feedback viewing.

Recommendations

For users

- To achieve the most accurate results, users should select a video input of them performing the exercise with multiple reps instead of one singular rep. This helps with better exercise classification since this is performed using K-means clustering, which works better with more accuracy when having more data points. The algorithm can also average out noises and anomalies in the data more effectively with more reps, leading to more reliable and consistent results in classifying exercises.
- Users need to ensure their videos are recorded at one of the angles available in the application - front, side, 45-degree front-right, 45-degree front-left, 45-degree behind-left, and 45-degree behind-right. This ensures the accuracy of pose detection and feedback.
- Users are recommended to perform this analysis regularly and from multiple view angles to effectively identify and correct their forms, reducing the risk of injury and improve the effectiveness of their exercises.

For future improvements

- **Expand exercise library:** Allow users to include additional exercises, such as pull-ups, chin-ups, hip thrust, bicep curls, or overhead press. This will make PosePilot more versatile and reach a broader range of users, not limited within beginners in powerlifting.
- **Automate view angle selection:** Develop an algorithm that automates the process of view angle selection to simplify the user experience.

Reflection

This project has definitely been a new learning experience for me. Building a Python-based app, even though on very low levels, was still a challenging exercise for someone with little to no coding experience (apart from the half semester with the minor Data-Driven Decision Making). During the project, I stumbled upon multiple problems, one of which was that the initial pose estimation algorithm (OpenPose) did not work out the way I expected. After multiple attempts to consult teachers (Witek), classmates, YouTube videos, I decided to switch to another framework, MediaPipe Pose, that turned out to work much more smoothly. In the modeling phase, I also spent a lot of time figuring out how to deal with multiple view angles, which phases/movements of the exercises to analyze, how to classify the exercises, and how to style the GUI effectively. The process was quite confusing and frustrating at first but eventually, as I got a hang of it, it turned out to be a very interesting learning experience. Now I know how to process a video input on Python, how to use a pose estimation algorithm, how to effectively utilize K-means clustering in data modeling, and many other skills. However, I cannot deny that this project still has various spaces for improvement. It can benefit from enhancements such as the ones mentioned above in the Recommendation section, which will further improve user engagement and experience.

Reference list

Naglazas, M. (2023, May 28). Gen gym: Why the young are leading the fitness revolution. *WAtoday*. https://www.watoday.com.au/national/western-australia/gen-gym-why-the-young-are-leading-the-fitness-revolution-20230524-p5db18.html

Top 3 powerlifting exercises: squat, bench press & deadlift | *Flex Blog*. (n.d.). Flex Fitness Equipment. https://www.flexequipment.com.au/blog/top-3-powerlifting-exercises-squat-bench-press-deadlift/

Bukhary, H. A., Basha, N. A., Dobel, A. A., Alsufyani, R. M., Alotaibi, R. A., Almadani, S. H., ... & Almadani, S. (2023). Prevalence and pattern of injuries across the weight-training sports. *Cureus*, *15*(11).

Austin, D. , & Mann, B. (2022). Powerlifting: The complete guide to technique, training, and competition. Champaign: Human Kinetics. Retrieved June 11, 2024, from http://dx.doi.org/10.5040/9781718225473

Кириченко, Т. Г. (2023). General physical fitness of powerlifters as a basis for effective strength development. Науковий часопис Національного педагогічного університету імені МП Драгоманова. Серія 15. Науково-педагогічні проблеми фізичної культури (фізична культура і спорт), (12 (172)), 100-107.

Branni, M. G. (2018). *Experimental protocol to develop and validate a 3D patient-specific Finite Element Model of human knee joint after Total Knee Arthroplasty: Squat analysis* (Doctoral dissertation, Politecnico di Torino).

Fekete, G., Csizmadia, B. M., Wahab, M. A., & De Baets, P. (2011). Analytical and computational estimation of patellofemoral forces in the knee under squatting and isometric motion. *International Journal of Sustainable Construction and Design*, *2*(2).

Newby, N., Caldwell, E., Sibonga, J., & Ploutz-Snyder, L. (2014, January). ISS Squat and Deadlift Kinematics on the Advanced Resistive Exercise Device. In *2015 Human Research Program Investigator''s Workshop* (No. JSC-CN-32155).

Sutthiprapa, S., Vanijja, V., & Likitwon, T. (2017). The deadlift form analysis system using Microsoft Kinect. *Procedia computer science*, *111*, 174-182.

Huang, Y. P., Chou, Y. L., Chen, F. C., Wang, R. T., Huang, M. J., & Chou, P. P. (2014). Elbow joint fatigue and bench-press training. *Journal of athletic training*, *49*(3), 317–321. https://doi.org/10.4085/1062-6050-49.1.05

Fioranelli, D., & Lee, C. M. (2008). The influence of bar diameter on neuromuscular strength and activation: inferences from an isometric unilateral bench press. *Journal of strength and conditioning research*, *22*(3), 661–666. https://doi.org/10.1519/JSC.0b013e31816a5775

Chowdhury, R., Chung, M., Gnagey, D., & Forsyth, J. (2023, April). GymSense: Exploration in Measuring Bench Press Form Using Sensor Technology. In *2023 Systems and Information Engineering Design Symposium (SIEDS)* (pp. 131-136). IEEE.

Google-Ai-Edge. (n.d.). *mediapipe/docs/solutions/pose.md at master · google-ai-edge/mediapipe*. GitHub. https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/pose.md

CodeTrade. (2023, November). MediaPipe vs TensorFlow: Battle of the human pose estimation giants. CodeTrade. https://www.codetrade.io/blog/mediapipe-vs-tensorflow-battle-of-the-human-pose-estimation-giants/

Beyond Poses: Openpose vs Mediapipe for Dynamic Vision. (n.d.). Saiwa. https://saiwa.ai/blog/openpose-vs-mediapipe/

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.

Appendix

Appendix A

Link to Github repo: https://github.com/orangecapybara/individualassignment

Appendix B

Link to presentation video: https://videotoetsing.han.nl/P2G/Player/Player.aspx?id=bgPHEe

Appendix C

The code to the project:

```python
import cv2
import mediapipe as mp
import numpy as np
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
from tkinter import ttk, font
from sklearn.cluster import KMeans

#-------INITIALIZE MEDIAPIPE POSE

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


#-------DEFINE VIDEO PROCESSING FUNCTION

def process_video(video_path, frame_skip=5, resize_factor=0.5):
    print(f"Starting video processing for {video_path}")
    cap = cv2.VideoCapture(video_path)
    joint_coordinates = []
    frame_count = 0

    if not cap.isOpened():
        print("Error: Could not open video.")
        return joint_coordinates

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

        frame_count += 1
        if frame_count % frame_skip != 0:
            continue

        frame = cv2.resize(frame, (0, 0), fx=resize_factor, fy=resize_factor)
        image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = pose.process(image_rgb)

        if results.pose_landmarks:
            frame_landmarks = [(lm.x, lm.y, lm.z) for lm in results.pose_landmarks.landmark]
            joint_coordinates.append(frame_landmarks)
            # Draw landmarks on the frame
            mp_drawing.draw_landmarks(frame, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)

        # Display the frame
        cv2.imshow('Video Processing', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

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

```python

#-------DEFINE ANGLE CALCULATION FUNCTION

def calculate_angle(a, b, c):
    a, b, c = np.array(a), np.array(b), np.array(c)
    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 = 360.0 - angle
    return angle



#-------DEFINE EXERCISE ANALYSIS FUNCTIONS

# Squat Analysis Fucntion

def analyze_squat(joint_coordinates, view_angle):
    hip_feedback = []
    knee_feedback = []
    ankle_feedback = []
    torso_feedback = []

    for frame in joint_coordinates:
        hip = frame[mp_pose.PoseLandmark.LEFT_HIP.value]
        knee = frame[mp_pose.PoseLandmark.LEFT_KNEE.value]
        ankle = frame[mp_pose.PoseLandmark.LEFT_ANKLE.value]
        shoulder = frame[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
        foot = frame[mp_pose.PoseLandmark.LEFT_FOOT_INDEX.value]

        if view_angle in ["side", "45-degree front-left", "45-degree front-right", "45-degree behind-left", "45-degree behind-right"]:
            hip_knee_angle = calculate_angle(hip, knee, shoulder)
            knee_angle = calculate_angle(hip, knee, ankle)
            ankle_angle = calculate_angle(knee, ankle, foot)
            torso_angle = calculate_angle((shoulder[0], shoulder[1] - 1), shoulder, hip)

            if hip_knee_angle < 90:
                hip_feedback.append("Hip flexion is adequate.")
            else:
                hip_feedback.append("Hip flexion is not adequate. Squat deeper.")

            if knee_angle >= 90:
                knee_feedback.append("Knee flexion is adequate.")
            else:
                knee_feedback.append("Knee flexion is not adequate. Squat deeper.")

            if 25 <= ankle_angle <= 30:
                ankle_feedback.append("Ankle dorsiflexion is within recommended range.")
            else:
                ankle_feedback.append("Ankle dorsiflexion is not within recommended range. Bend forward more.")

            if 30 <= torso_angle <= 50:
                torso_feedback.append("Torso angle is within recommended range.")
            else:
                torso_feedback.append("Torso angle is not within recommended range. Keep your torso upright.")

        if view_angle in ["front", "45-degree front-left", "45-degree front-right"]:
            knee_distance = np.linalg.norm(np.array(frame[mp_pose.PoseLandmark.LEFT_KNEE.value]) - np.array(frame[mp_pose.PoseLandmark.RIGHT_KNEE.value]))
            if knee_distance < 0.2:
                knee_feedback.append("Keep knees aligned with toes.")
            else:
                knee_feedback.append("Don't let your knees cave in.")

    # Consolidate feedback for each joint
    consolidated_feedback = {
        "Hip Flexion": max(set(hip_feedback), key=hip_feedback.count),
        "Knee Flexion": max(set(knee_feedback), key=knee_feedback.count),
        "Ankle Dorsiflexion": max(set(ankle_feedback), key=ankle_feedback.count),
        "Torso Angle": max(set(torso_feedback), key=torso_feedback.count)
    }

    return consolidated_feedback

```

```python

# Deadlift Analysis Fucntion

def analyze_deadlift(joint_coordinates, view_angle):
    bottom_phase_feedback = []
    for frame in joint_coordinates:
        hip, knee, ankle = frame[mp_pose.PoseLandmark.LEFT_HIP.value], frame[mp_pose.PoseLandmark.LEFT_KNEE.value], frame[mp_pose.PoseLandmark.LEFT_ANKLE.value]
        if view_angle in ["side", "45-degree front-left", "45-degree front-right", "45-degree behind-left", "45-degree behind-right"]:
            hip_knee_angle = calculate_angle(hip, knee, ankle)
            bottom_phase_feedback.append("Keep your back straight and bend at the hips." if hip_knee_angle < 160 else "Good form, keep it up!")
        elif view_angle in ["front", "45-degree front-left", "45-degree front-right"]:
            knee_distance = np.linalg.norm(np.array(frame[mp_pose.PoseLandmark.LEFT_KNEE.value]) - np.array(frame[mp_pose.PoseLandmark.RIGHT_KNEE.value]))
            bottom_phase_feedback.append("Keep knees aligned with toes." if knee_distance < 0.2 else "Don't let your knees cave in.")
    consolidated_feedback = {
        "Deadlift Feedback": max(set(bottom_phase_feedback), key=bottom_phase_feedback.count)
    }
    return consolidated_feedback

# Bench Press Analysis Fucntion

def analyze_bench_press(joint_coordinates, view_angle):
    bottom_phase_feedback = []
    for frame in joint_coordinates:
        shoulder, elbow, wrist = frame[mp_pose.PoseLandmark.LEFT_SHOULDER.value], frame[mp_pose.PoseLandmark.LEFT_ELBOW.value], frame[mp_pose.PoseLandmark.LEFT_WRIST.value]
        if view_angle in ["side", "45-degree front-left", "45-degree front-right", "45-degree behind-left", "45-degree behind-right"]:
            shoulder_elbow_angle = calculate_angle(shoulder, elbow, wrist)
            bottom_phase_feedback.append("Keep your elbows at a 90-degree angle." if shoulder_elbow_angle < 90 else "Good form, keep it up!")
        elif view_angle in ["front", "45-degree front-left", "45-degree front-right"]:
            shoulder_distance = np.linalg.norm(np.array(frame[mp_pose.PoseLandmark.LEFT_SHOULDER.value]) - np.array(frame[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]))
            bottom_phase_feedback.append("Keep shoulders stable and symmetrical." if shoulder_distance < 0.1 else "Don't let your shoulders shift.")
    consolidated_feedback = {
        "Bench press Feedback": max(set(bottom_phase_feedback), key=bottom_phase_feedback.count)
    }
    return consolidated_feedback

#-------DEFINE EXERCISE CLASSIFICATION FUNCTION

def classify_exercise(joint_coordinates):
    if not joint_coordinates:
        print("Error: No joint coordinates found.")
        return "unknown"

    angles = []
    for frame in joint_coordinates:
        shoulder, elbow, wrist = frame[mp_pose.PoseLandmark.LEFT_SHOULDER.value], frame[mp_pose.PoseLandmark.LEFT_ELBOW.value], frame[mp_pose.PoseLandmark.LEFT_WRIST.value]
        hip, knee, ankle = frame[mp_pose.PoseLandmark.LEFT_HIP.value], frame[mp_pose.PoseLandmark.LEFT_KNEE.value], frame[mp_pose.PoseLandmark.LEFT_ANKLE.value]
        shoulder_elbow_angle = calculate_angle(shoulder, elbow, wrist)
        hip_knee_angle = calculate_angle(hip, knee, ankle)
        angles.append([shoulder_elbow_angle, hip_knee_angle])

    angles = np.array(angles)
    kmeans = KMeans(n_clusters=3, random_state=0).fit(angles)
    labels = kmeans.labels_

    squat_count = np.sum(labels == 0)
    deadlift_count = np.sum(labels == 1)
    benchpress_count = np.sum(labels == 2)

    print(f"Squat count: {squat_count}, Deadlift count: {deadlift_count}, Benchpress count: {benchpress_count}")

    if squat_count > deadlift_count and squat_count > benchpress_count:
        return "squat"
    elif deadlift_count > squat_count and deadlift_count > benchpress_count:
        return "deadlift"
    else:
        return "benchpress"
```

```python
#-------DEFINE VIDEO ANALYSIS AND FEEDBACK FUNCTIONS

def analyze_video(video_path, view_angle):
    print(f"Analyzing video: {video_path}")
    joint_coordinates = process_video(video_path)
    if not joint_coordinates:
        return {"Error": ["Could not process video or no landmarks detected."]}

    exercise_type = classify_exercise(joint_coordinates)
    print(f"Detected exercise: {exercise_type}")

    if exercise_type == "squat":
        feedback = analyze_squat(joint_coordinates, view_angle)
    elif exercise_type == "deadlift":
        feedback = analyze_deadlift(joint_coordinates, view_angle)
    elif exercise_type == "benchpress":
        feedback = analyze_bench_press(joint_coordinates, view_angle)
    else:
        feedback = {"Error": ["Unknown exercise detected. Please try again with a different video."]}

    return feedback
```

```python
#-------GUI SETUP

def select_video():
    file_path = filedialog.askopenfilename(filetypes=[("All files", "*.*")])
    if file_path:
        video_path.set(file_path)

def process_selected_video():
    file_path = video_path.get()
    view_angle = view_angle_var.get()
    if not file_path:
        messagebox.showerror("Error", "Please select a video file.")
        return

    feedback = analyze_video(file_path, view_angle)
    feedback_text.set("\n".join([f"{key}: {value}" for key, value in feedback.items()]))

app = tk.Tk()
app.title("Exercise Form Analyzer")

# Define a custom style for the frame
style = ttk.Style()
style.configure("LightPink.TFrame", background="#FFD1DC")  # Set the background color to light pink
style.configure("LightPink.TLabel", background="#FFD1DC", foreground="black")  # Set label background and text color
style.configure("LightPink.TButton", background="#FFD1DC", foreground="black")  # Set button background and text color
style.configure("LightPink.TEntry", fieldbackground="#FFD1DC", foreground="black")  # Set entry field background and text color
style.configure("LightPink.TCombobox", fieldbackground="#FFD1DC", foreground="black")  # Set combobox field background and text color

video_path = tk.StringVar()
feedback_text = tk.StringVar()
view_angle_var = tk.StringVar(value="side")

frame = ttk.Frame(app, padding="10", style="LightPink.TFrame")  # Apply the custom style to the frame
frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

# Title
title_label = ttk.Label(frame, text="Exercise Form Analyzer", style="Bold.TLabel", background="#FFD1DC")
title_label.grid(row=0, column=0, columnspan=3, sticky=tk.W)

# Configure font to make title bold or increase font size
title_font = font.Font(title_label, title_label.cget("font"))
title_font.configure(weight="bold", size=14)  # Adjust weight and size as needed
title_label.configure(font=title_font)

# View Angle Definitions
view_angle_definitions = """
View Angle Definitions:
- Front: Directly facing the person.
- Side: 90 degrees to the left or right of the person.
- 45-degree front-left: 45 degrees to the left of the front view.
- 45-degree front-right: 45 degrees to the right of the front view.
- 45-degree behind-left: 45 degrees to the left of the back view.
- 45-degree behind-right: 45 degrees to the right of the back view.
"""
# Create a label with light pink background for the view angle definitions
view_angle_definitions_label = ttk.Label(frame, text=view_angle_definitions, wraplength=500, justify="left", anchor="w", foreground="black", style="LightPink.TLabel")
view_angle_definitions_label.grid(row=1, column=0, columnspan=3, sticky=tk.W)
view_angle_definitions_label.configure(background="#FFD1DC")  # Set background color to light pink

# Select Video File
ttk.Label(frame, text="Select Video File:", style="LightPink.TLabel").grid(row=2, column=0, sticky=tk.W)
ttk.Entry(frame, textvariable=video_path, width=40, style="LightPink.TEntry").grid(row=2, column=1, sticky=tk.E)
ttk.Button(frame, text="Browse", command=select_video, style="LightPink.TButton").grid(row=2, column=2, sticky=tk.E)

# View angle selection
ttk.Label(frame, text="View Angle:", style="LightPink.TLabel").grid(row=3, column=0, sticky=tk.W)

# Create the OptionMenu widget for view angle selection
view_angle_menu = ttk.OptionMenu(frame, view_angle_var, "side", "side", "front", "45-degree front-left", "45-degree front-right", "45-degree behind-left", "45-degree behind-right")

# Configure the style for the dropdown menu options
style.configure("Dropdown.TMenubutton", foreground="black")

# Apply the custom style to each individual option in the dropdown menu
for child in view_angle_menu["menu"].winfo_children():
    style.configure(child, foreground="black")

# Grid the OptionMenu widget
view_angle_menu.grid(row=3, column=1, columnspan=2, sticky=tk.W)

# Analysis button
ttk.Button(frame, text="Analyze", command=process_selected_video, style="LightPink.TButton").grid(row=4, column=0, columnspan=3, pady=10)

# Feedback
ttk.Label(frame, text="Feedback:", style="LightPink.TLabel").grid(row=5, column=0, sticky=tk.W)
ttk.Label(frame, textvariable=feedback_text, wraplength=400, style="LightPink.TLabel").grid(row=6, column=0, columnspan=3, sticky=tk.W)

app.mainloop()
```