Pose Estimation for Fitness Exercise Analysis

AI Lab: Computer Vision and NLP

STUDENTS:

Di Gennaro Veronica (1984033)

Guarnizo Orjuela Daniel Yezid (1993238)

PROFESSOR:

Pannone Daniele

Sapienza Università di Roma

Department of Computer Science

ABSTRACT

This paper presents a computer vision-based approach for analysing fitness exercises, specifically focusing on the squat, deadlift and bench press movements.

We leverage the MediaPipe framework to obtain landmark positions from video data and employ Scikit-learn to train models for the primary task of distinguishing between phases of an exercise repetition. These models serve as the foundation for achieving the main objectives of our system: repetition counting and exercise quality assessment.

1. INTRODUCTION

Pose Estimation is a computer vision task that involves detecting the position and orientation of an object or, as it pertains to our situation, of a person, namely Human Pose Estimation. For the latter, this is done by predicting the location of specific keypoints. These may include various joints in the human body, such as the shoulders, wrists, elbows, and other specific body parts.

Nowadays, pose estimation has become a ubiquitous feature in a wide variety of applications. One typical example is physical fitness tracking and performance analysis, and our team has decided to focus precisely on this. Physical fitness is undoubtedly crucial for maintaining a healthy lifestyle, but effective exercise execution is even more fundamental for achieving fitness goals while minimizing the risk of injury. Our project implements a computer vision-based solution for analyzing three basic fitness exercises, the squat, the deadlift, and the bench press using pose estimation techniques through the MediaPipe framework and machine learning algorithms.

1.1. MEDIAPIPE FRAMEWORK

MediaPipe is an open-source framework developed by Google that comes with pre-trained machine-learning models for a range of tasks, including hand tracking and the one in our interest, pose estimation. One of the standout features of this framework is its emphasis on real-time processing making it ideal for applications that require immediate feedback or interaction.

An important component of this framework is the MediaPipe Pose Landmarker, which can process input images or video frames and use a series of models to detect the presence of bodies within the frame and to track a complete mapping of the pose on screen. The model then outputs an estimate of 33 3-dimensional body landmarks, each of which is assigned a score corresponding to the model's confidence in the accuracy of the landmark's position.

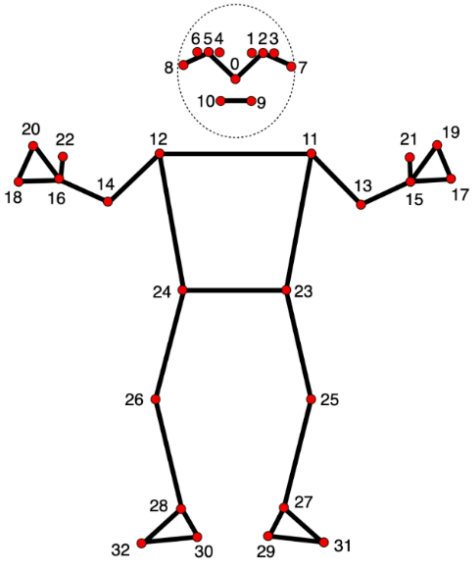

Figure 1: Definition of landmarks in MediaPipe Pose [1]

1.2. FUNCTIONAL OVERVIEW

The main idea for this project was to build an algorithm that could distinguish the different phases of an exercise execution so as to correctly count the repetitions and give some real-time suggestions to improve the form of the execution where needed.

Let’s take deadlifts as our example. Once we access our video camera through python using the OpenCV library, we can start working out doing our deadlifts. As we move in front of the camera, the information about our pose is extracted and used to categorize movements and consequentially display on screen (with the use of OpenCV) whatever we need. All of this is done in real time.

Figure 2 displays the screen during execution.

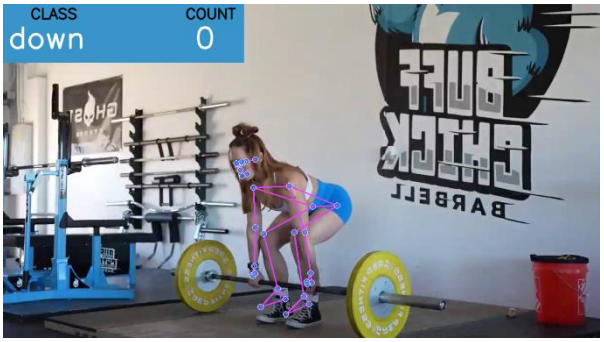

Figure 2

2. METHODOLOGY

2.1. DATA COLLECTION AND PRE-PROCESSING

We first started by building a labeled dataset ourselves. We captured video footage of us individually performing squats, deadlifts, and bench presses in the most correct way possible. To use this data for subsequent analysis, we needed it to be in the form of relative positions of body parts in each phase of the exercise execution.

Continuing with the previous example of the deadlift, we aimed to distinguish the body pose at the top from the one at the bottom of the exercise repetition. To accomplish this the MediaPipe framework was employed to accurately extract the positions of landmarks throughout both the "UP" and "DOWN" phases of the exercise.

With a few lines of code, we were able to organize this data in the form of a CSV file. The first column of this file contains the classes, "UP" and "DOWN", while the rest of the columns denote the positions of the landmarks in these phases. (Figure 3)

We then repeated the same procedure to obtain data for the quality assessment of the exercise execution.

For the sake of consistency, we take up again the deadlift example. We wanted our model to distinguish between 4 possible classes of incorrect posture during a deadlift repetition: DOWN_LOW (hips too low), DOWN_ROLL (excessive downward bending of the back), UP_BACK (leaning too far backward), UP_ROLL (rounding of the back after completing the lift).

After extracting the coordinates for these new body postures, the resulting data was appended to the previously mentioned CSV file.

We must specify that, upon completion of the data collection process, we had with a specific CSV file for each exercise and not a unique one due to the varying classes present among them.

	A	B	C	D	E	F	G	H	I	
1	class	x1	y1	z1	v1	x2	y2	z2	v2	x3
2	up	0.5333298	0.094314(	-0.174967	0.9999268	0.5376046	0.0848772	-0.153137	0.9998688	0.
3	up	0.5504945	0.105592(	-0.107166	0.9999706	0.5552634	0.1009077	-0.089068	0.9999433	0.
4	up	0.5504187	0.108258:	-0.104524	0.9999723	0.5551168	0.1024703	-0.085323	0.9999462	0.
5	up	0.5223667	0.123939:	-0.122131	0.9999896	0.5276777	0.1163325	-0.104159	0.9999776	0.
6	up	0.5337255	0.115172:	-0.128488	0.9999846	0.5389376	0.1110304	-0.109424	0.9999656	0.
7	down	0.5392485	0.2994395	-0.178422	0.9999701	0.5439674	0.2922919	-0.157572	0.9999328	0.
8	down	0.5404546	0.3622912	-0.155521	0.9999061	0.5457814	0.3525052	-0.136552	0.9998414	0.
9	down	0.5404545	0.3620835	-0.152223	0.9998815	0.5457834	0.3520965	-0.133605	0.9998085	0.
10	down	0.5403847	0.3639316	-0.158573	0.9998598	0.5457512	0.3536970	-0.141978	0.9997782	0.
11	down	0.5401666	0.3514882	-0.213817	0.9998744	0.5455397	0.3415545	-0.196062	0.9997876	0.
12	up	0.5372525	0.1143211	-0.120907	0.9999635	0.5428015	0.1116017	-0.101847	0.9999202	0.
13	up	0.5358964	0.1133873	-0.131658	0.9999846	0.5410238	0.1084233	-0.112860	0.9999654	0.
14	down	0.5386455	0.3564726	-0.181217	0.9999255	0.5443902	0.3482435	-0.161758	0.9998596	0.
15	down	0.5390615	0.3513775	-0.182531	0.9999012	0.5450534	0.3443967	-0.165958	0.9998185	0.

Figure 3

2.2. MODEL TRAINING

At this point, we needed to train some models, one for each of the three exercises, that could accurately distinguish between the different classes based on landmark positions.

We opted to use the open-source library Scikit-learn for Python as it provides a comprehensive set of machine learning algorithms and because of the offered utilities for evaluating these models through various metrics.

After loading the CSV file into a Pandas DataFrame and splitting it into train and test sets, we were uncertain which of the wide range of classifier algorithms we should opt for, so we decided to take advantage of how Scikit-learn simplifies the process of experimenting with the different ML algorithms.

We started by importing and initializing the classes of some classifiers we were interested in:

```
LogisticRegression,
RidgeClassifier,
RandomForestClassifier,
GradientBoostingClassifier.
```

Then, we used a loop to iterate through a list of these models so as to train each of them on the common training data, make predictions on the same test data, and finally use some evaluation

```
lr 0.9986559139784946 0.9990310077519379 0.9985875706214689
rc 0.9986559139784946 0.9990310077519379 0.9985875706214689
rf 0.9986559139784946 0.9990310077519379 0.9985875706214689
gb 0.9959677419354839 0.9968178356118406 0.9962355012837536
```

Figure 4: Here are shown in order: Accuracy, Precision, and Recall for each of the four trained models for the deadlift exercise. We can notice how they all are surprisingly high, meaning we built a good dataset.

metrics to compare their performance. In detail, we used accuracy, precision, and recall. It was noteworthy that each one of the models performed exceptionally well, with scores that were very close to one another. However, we had to make a final decision and opted for the random forest classifier.

At this point, we had three (one for each exercise) pre-trained models crucial for recognizing different stages or poses during the fitness exercise.

2.3. REPETITION COUNTING and QUALITY ASSESSMENT

Now, we can finally start going through the real-time part.

Once the webcam feed is opened for real-time video input, video frames are captured in a loop, which enables the analysis of exercises as they are performed. Each captured video frame is then processed by converting it from the BGR color space to the RGB color space and adjusting its writeable flag to allow for processing.

Here, the MediaPipe Pose module comes in handy again to detect and track landmarks on the human body in the frame. In addition to being drawn on the video frame for visual representation, the landmarks are also extracted and organized into a Pandas DataFrame that is then used as input to the pre-trained classification model, specific to the exercise in execution.

The resultant output of the model is the predicted class. A probability associated with the prediction is also calculated. This particular information enabled us to complete the subsequent tasks reliably.

The code keeps track of the model prediction as the current stage, and it increments a repetition counter whenever it detects a transition from a “down” stage to an “up” stage, indicating the completion of a repetition. It is important to note that, for instance, in the case of a deadlift, DOWN_ROLL, DOWN_LOW, and DOWN are all different cases of a “down” stage, and for this reason, they are all considered the same for the repetition counting task.

However, when it comes to assessing the quality of the exercise, the pre-written suggestions displayed on the screen will vary depending on which one of the specific classes is predicted by the model.

3. CONCLUSION

In conclusion, the proposed system is able to identify exercise phases and effectively count repetitions. It can also distinguish between proper and improper execution of the exercises, demonstrating its potential to enhance fitness training and coaching. We have used MediaPipe’s Pose and leveraged the power of OpenCV to build a simple tool but we can further improve it by incorporating more advanced techniques, such as building a Human Action Recognition system modelling the temporal dependencies and relationships between frames system using a Recurrent Neural Networks (RNNs) and LSTM networks [2]. Another possible improvement could be expanding the scope to include additional exercises making it a more complete fitness tool.

REFERENCES:

[1] Y. Luo et al., "LSTM Pose Machines," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 5207-5215, doi: 10.1109/CVPR.2018.00546.

[2] Pose landmark detection guide | mediapipe . (n.d.). *Google*. Google. Retrieved September 1, 2023, from https://developers.google.com/mediapipe/solutions/vision/pose_landmarker