Name : Mohammad Ausaf
Contact Number : +91 7007388740
E-Mail : ausaf9911@gmail.com
LinkedIn : https://www.linkedin.com/in/mohammad-ausaf/

Computer vision

Instruction Doc

Operating System : Windows 11

IDE used : PyCharm

Language Used : Python 3.9.13

Libraries Used:

- OpenCV2
- Numpy
- From Datetime, used datetime and timedelta
- From PyScenedetect used content detector
- Used Media pipe Pose classification model as advised

NOTE : You can install PySceneDetect package within PyCharm with name of “scenedetect”, or if you want to pip install look here.

LOGICAL Flow: (Import all the Libraries)

For handling fluctuations:

1) Using scenedetector to grab location of cutscenes
2) Store this cutscene frames in a list (say delete) (starting from the location and till the (current frames + fps) is reached), that is (25) frames for each cut scenes, amounting for 1 second of the video. (fps may vary depending on the video).
3) Keep a count of current frame, increasing as soon as it enters the detection algorithm.

4) While the capture device is opened, check if the current frame belongs to deleted frames, if no, continue with detection other wise skip.

For detection:

1) While the capture device is opened, perform landmarks detection with media pipe pose detection
2) Get the coordinates of the detected landmarks
3) Calculate angle between the required landmarks
4) Initialize the stage of the knee at the start based on the starting angle
5) Check if the angle is greater than 140 degrees, if yes keep the stage as ‘straight knee’
6) If the angle is less than 140, change the knee stage to ‘bent knee’ and start the timer.
7) Check if the timer is exhausted (8 seconds), if not show the timer of the window
8) If the timer is greater than 8, increase the rep count and keep the stage as ‘bent knee’ until the angle becomes greater than 140.
9) If the timer < 8 and the angle becomes greater than 140, increase the show feedback to the window “ keep your knee bent”
10) Repeat the algorithm for every frame
11) If at any point of time “q” key is pressed, break out of the loop, and close the windows, while releasing the capture device.

Result:

As the Pose detection algorithm has 3 complexities, the counts vary accordingly, note that the complexity increment does increases detection accuracy, however, higher complexity is very RAM intensive, so frame rates will take a hit, the default detection complexity = 1, does work fine. For pose detection the min detection and tracking confidence values are 0.8 and 0.5 respectively.

The number of successful reps counts for all the complexity levels are:

1) For model complexity = 1 (default), rep counts are 12
2) For model complexity = 0 (lower accuracy), rep counts are 13
3) For model complexity = 2(higher accuracy), rep counts are 14

Further improvements possible:

1) The detection of fluctuations can be further improved by tuning parameters of ContentDetector(), especially the *threshold value*.
2) The detection of landmarks can be further improved by tuning of parameter, like, *detection confidence, tracking confidence, and model complexity.*