Get Fit With PCA

MOHIT CHOITHWANI[1], SNEHA ALMEIDA[1], INDRANI ROY[1], AND PROF. DR. BERNHARD EGGER[2]

[1] *Project: Computational Visual Perception*
[2] *Project Suppervisor*

Compiled April 24, 2023

Physical exercises are being massively promoted as means to a healthy lifestyle. Although this statement might sound exaggerated, but as our lifestyle becomes more and more confined to screens, cubicles, and cabins, we need to turn to physical exercises to stay fit. While some physical exercises are easy, those generally fall into the category of easy or beginner-level of exercises. Intermediate and advanced exercise levels require monitoring body posture while performing exercises. If performed incorrectly, the exercise can do more harm than good, something as worse as an injury. A trainer is recommended in such cases. However, not everyone can have access to a trainer. So, we propose an automated application that takes video footage of the exercise being performed and scores the "correctness" of the exercise. This system currently works for one exercise, namely, the dumbbell shoulder press.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

Fig 1 provides an overview of the steps the application takes from a user's perspective. These steps have been explained below:

- Takes video of the exercise performed by the user as input
- Analyzes every frame of the video
- Marks each frame as correct or incorrect
- Scores the "correctness" of the exercise numerically

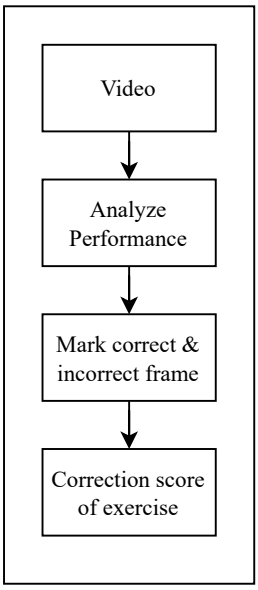

Fig. 1. System Overview

2. METHODOLOGY

While the overview in the introduction is quite concise, it also lacks technical and methodical details. The block diagram gives a detailed view of the system with the required technical and methodical details so as to enable detailed understanding of the system. As described in the block diagram in figure 2, the system can be divided into 3 major modules, namely training, testing and Evaluation. Each of these modules has been briefly explained in the subsections that follow:

A. Training

In this module, we train the PCA model. The input to this module is a set of videos where the correct exercise is performed. This module outputs a PCA model which has been trained to identify the dumb-

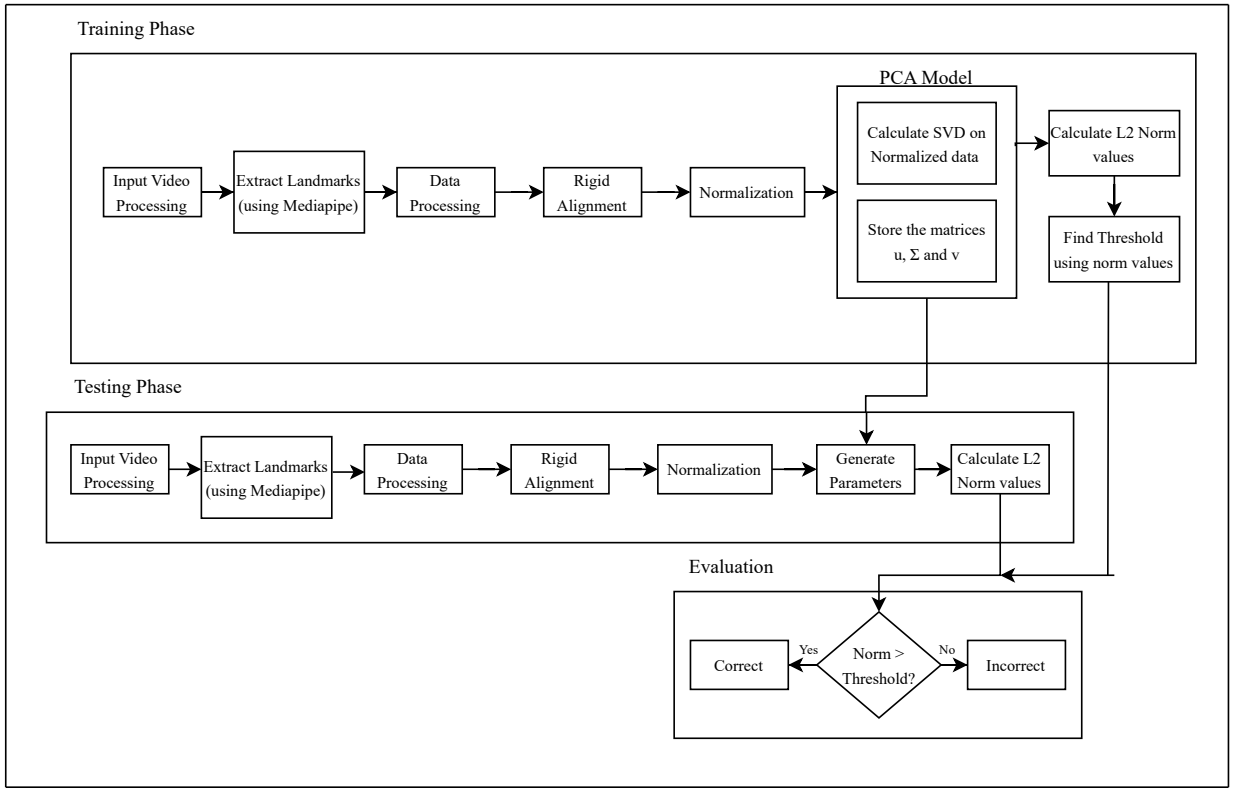

Fig. 2. Block Diagram

bell shoulder press exercise. This module involves the following steps:

1. Input Video Processing
   In this step, frames are extracted from the training videos to enable further processing of the video (or frames).

2. Landmark Extraction
   For every frame extracted from the video in the previous step, 2D landmarks are calculated. The MediaPipe library [1] is used for calculating these landmarks. As shown in Fig 4, it marks 33 landmarks on a normal human body. However, we only take 23 out of those, which are relevant for our application. Landmarks relevant for our application have been plotted on a frame in Fig 3.

3. Data Processing
   The landmarks calculated in the previous step are stored in a CSV file. Since we are considering 23 landmarks, we have 46 coordinates (x and y coordinates for each landmark). The landmarks of each frame are represented by a row in the CSV file. So there are 46 columns (2 columns

Fig. 3. Normalized mean landmarks plotted on reference frame

for each landmark) and as many rows as the number of frames.

4. Rigid Alignment
   In this step, we use the frame shown in Fig 3 as a reference frame and align landmarks in all the other frames according to the reference frame. For this alignment, the "Kabsch Umeyama" algorithm [2] is used.
   Rigid alignment is the process of transforming a set of points from one coordinate space to another while trying to preserve the shape and orientation of the points.
   Rigid alignment transforms the sample from one domain into another, this transformation tries to find the optimal translation, rotation, and scaling operation such that the orientation and shape of the sample point aligns with the reference point

   In the application, we used rigid alignment to align the frames of the video so that they are all the frames in the same orientation and position. This is important because it ensures that the landmarks extracted from each frame are consistent and can be compared accurately. For this we use Kabsch Umeyama algorithm to ensure that the landmarks extracted from each frame are aligned consistently, allowing the model to accurately determine whether the user is performing the exercise correctly or not.

5. Normalization
   After performing a rigid alignment on landmarks of all frames, the mean is calculated for each landmark. Fig 3 shows the mean set of landmarks plotted on the reference frame. Later, the mean is subtracted from landmarks of all frames and the data is now "mean-free" or "normalized".

6. Principal Component Analysis (PCA)
   Normalized data obtained in the previous step is used to perform PCA [3]. Following steps are involved in PCA:

   - Calculate SVD on normalized data
     Singular value decomposition (SVD) is performed on the normalized data.
   - Store the matrices u, Σ, v
     The matrices obtained after performing SVD are stored for later use(for testing).

7. Find Threshold
   u, Σ, v matrices are used to calculate a metric that can meaningfully differentiate between correct and incorrect frames of the exercise. The metric that we calculate is "Norm Values". Norm values are calculated by considering a specific number of parameters generated by providing test data. Fig 6 shows Norm values for correct and incorrect frames.

   In the app, we used the numpy library to calculate the L2 norm [4] of the first 20 parameters. Numpy provides a convenient way to calculate the L2 norm of a vector using the linalg.norm()[5] function.

   We find a threshold for these norm values. This threshold helps to distinguish between correct and incorrect frames.

   The threshold that we have calculated is 15.0. At this threshold, the algorithm described below gives an accuracy of 96 percent as shown in Fig 5.

   We use the following algorithm to find the threshold:

Algorithm 1. Algorithm for Threshold Calculation

Require: *list_of_parameters*: a list of numerical values representing the parameters *dataframe_of_labels*: a dataframe of binary labels for correct and wrong frames
Ensure: *accuracy*: an empty list that will store accuracy scores for each norm value

```
minimum ← min(list_of_parameters)
maximum ← max(list_of_parameters)
accuracy ← []
for i in round(minimum, maximum) do
    predicted ← []
    for j in list_of_parameters do
        if i > j then
            append 1 to predicted
        else
            append 0 to predicted
    calculate accuracy score for the i^th norm value
    append accuracy score to accuracy
return the norm value which has the highest accuracy
```

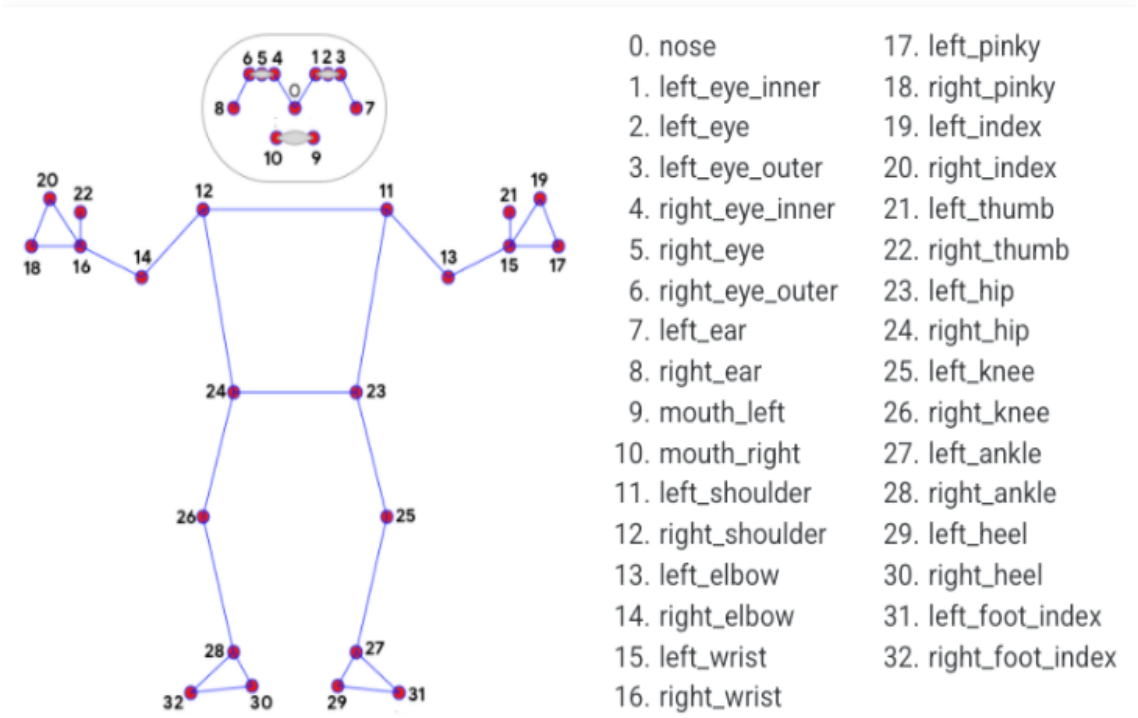

Fig. 4. Landmarks Detected by Mediapipe Library[1]

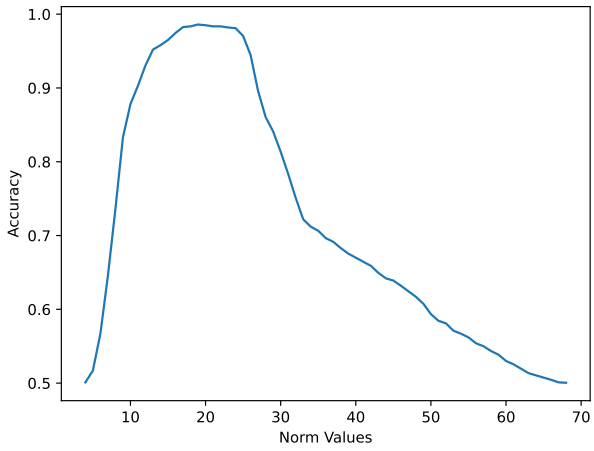

Fig. 5. Calculation of Threshold

B. Testing

In the testing phase, we give a test video as input and the model examines whether the exercise being performed is correct or incorrect. The first five steps (from Video processing to normalization are the same as training and hence, have not been elaborated here separately). The next steps in the testing phase which are different from training are as follows:

- Generate Parameters
  We generate parameters for each test frame using Σ and the u matrix. Maximum information is contained in the initial parameters. Hence, we will be using the first 20 parameters of every frame to calculate its norm value which is described below.
- Calculate L2 Norm values
  Norm values are calculated in the same way as in training.

C. Evaluation

In the evaluation phase, we evaluate if the frame is correct or incorrect, based on the threshold.

Value > Threshold: Incorrect Frame
Value <= Threshold: Correct Frame

3. ADVANTAGES

- Data requirement to train the model is very low.
- Simplicity of the model

4. LIMITATIONS

The system can be said to have the following limitations:

- Works for one exercise

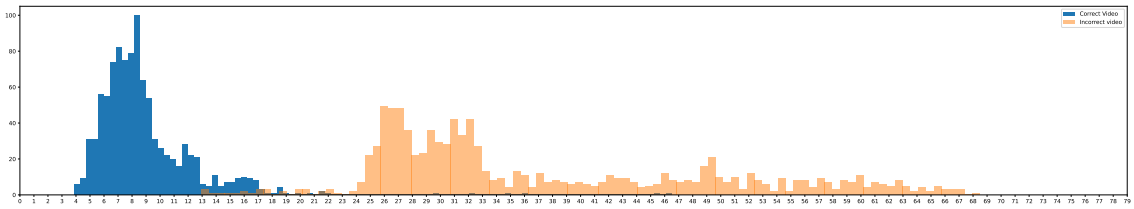

Fig. 6. Norm Values for Correct and Incorrect Frames

- Processes only frontal bodies

5. FUTURE WORK

Here are some points that hint at the possible directions for future work:

- Incorporate multiple exercises
- Support for non-frontal videos
- Real-time processing of incoming video stream
- Interactive GUI
- Portability (mobile phone application)

REFERENCES

1. "Mediapipe library," https://google.github.io/mediapipe/solutions/pose.html.
2. "kabsch umeyama," https://www.digitalocean.com/community/tutorials/norm-of-vector-python.
3. "Principle component analysis," http://gravis.cs.unibas.ch/smiley.
4. "L2 norm," https://www.digitalocean.com/community/tutorials/norm-of-vector-python.
5. "Python's library to calculate norm," https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html.