Gesture Recognition and Pose Analysis using Mediapipe Pose Landmark Detection

Eduard Martín Jiménez[1] and Anna Pallarès López[1]

[1] DIBRIS, Università degli Studi di Genova

Abstract

In the following report a Python script is developed and presented. The MediaPipe Pose library has been used to analyze and compute 3D and 2D average directions for distinct body gestures. The script processes 3D trajectories of the selected landmark subsets, applying kinetic energy thresholds to identify different gestures. For each different gesture, the average direction of the movement is computed. Designed for applications ranging from rehabilitation to virtual environments, this tool provides flexibility in choosing subsets such as whole body, upper body, or specific limbs. In brief, this project is a valuable resource for motion analysis and human-computer interaction.

Introduction

Pose Detection, also known as Pose Estimation, is a widely used computer vision task that enables one to predict human poses in images or videos by localizing the key body joints (which from now on will be referred to as *landmarks*) such as elbows, shoulders, knees, between others. The MediaPipe library provides a robust solution capable of predicting thirty-three 3D landmarks on a human body (Figure. 1) from images or videos in real-time with high accuracy even on CPU. It utilizes a two-step machine learning pipeline, by using a detector that first localizes the body within the frame and then uses the pose landmarks detector to predict the landmarks within the region of interest.

The "MediaPipe Gesture Analyzer" project was developed as a possible solution to the problem statement proposed in the *Multimodal Systems* Course and has been powered by the increasing interest for gesture analysis in a variety of fields, including virtual reality, rehabilitation, and human-computer interaction. Creating a Python script with the MediaPipe Pose library and focusing on particular subsets of body landmarks is the main goal. MediaPipe Pose is used in the project's methodology to detect landmarks and then compute average directions in two and three dimensions on various planes. This introduction establishes the framework for an extensive examination of gesture analysis and provides the foundation for the sections that follow, which cover the entire methodology used, the results and its interpretation, and finally the conclusions.

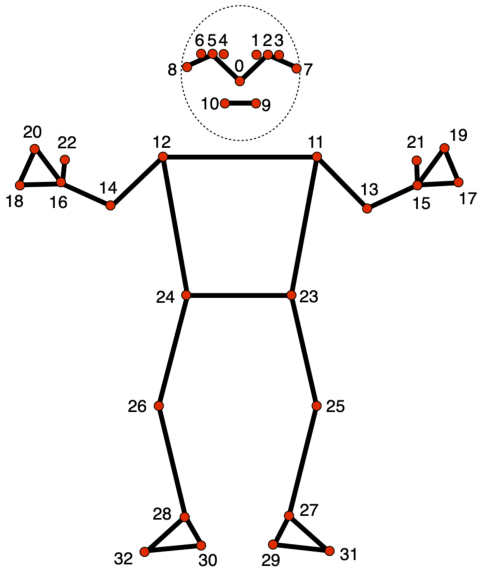

Figure 1: *The pose landmarker model tracks 33 body landmark locations, representing the approximate location of the following body parts. Source: Google MediaPipe,*

Methodology

In this section, a detailed description of the procedures and techniques followed to reach our goal are presented.

Data Collection

First of all, data collection has been achieved by using the open-source MediaPipe library developed by Google [1, 2]. In this sense, landmarks have been collected from a specific video we selected. At this point, we think it is important to remark that the algorithm has some limitations. In this particular case, as cleaner the clip is, the better the results are. Note that by clean, we refer to a video in which the human body is well-differentiated from the background and in which all the body parts are visible.

The first part of the algorithm makes it possible to integrate the MediaPipe algorithm on the selected video [3]. After that, the snippets we developed, use this collection of landmarks stored in three data frames -one for each space coordinate-, to compute the kinetic energy. To achieve this, the displacements (Eq. 3) were first calculated to obtain the velocity Eq. 2 and finally the energy of the movement using Eq.1 [4].

$$\text{ke}(t_i) = \frac{1}{2}\sum_{j=1}^{M_p} m_j \cdot (v^j(t_i))^2 \tag{1}$$

Where the velocity, v, and the displacements, d, are defined as follows in Eq. 2 and 3.

$$\mathbf{v^j}(\mathbf{t_i}) = \frac{\mathbf{d}^j(t_i)}{t_i} \tag{2}$$

$$\mathbf{d}^j(t_i) = \mathbf{p}^j(t_i) - \mathbf{p}^j(t_{i-1}) \tag{3}$$

An important remark to highlight from this calculation is that the approximate mass for each set of landmarks was retrieved from the Michigan Data provided in the professor's notes [4]. The percentages of the total body mass that we used were first normalized and grouped equally, by body parts. Finally, we obtained seven parts per unit from the total body mass. The different parts of the body and their mass according to each set of landmarks are summarized in table 1.

Kinetic Energy and Filtered Response

Once the kinetic energy is stored in the corresponding data frame, and before establishing a threshold to detect movements, a filter is applied to clean the response and obtain better results. To do so, we have used the *.rolling()* function from *pandas* library which allows to restrict the number of observations used for each calculation. A window size of 5 has been chosen, which is often a compromise between smoothing out the noise and retaining the responsiveness to actual changes in movement. Different window sizes were tested to prove that, from 2, where noise was still present from a value of 10, where too much data was lost. Therefore 5 is a good trade-off to accomplish a useful response. Moreover, we think that the choice of this method is appropriate due to its simplicity and effectiveness in comparison with other smoothing and filtering methods which may be more computationally expensive for the kind of work we want to achieve.

Next, a threshold was determined for the cleaned response to determine whether the person in the clip was in movement or not. This threshold was determined by computing the mean of the kinetic energy and subtracting two units, in other to adjust this threshold so that it has a valuable meaning for our purpose. Note that, this threshold may not be useful for other clips and would be a parameter to be adjusted. In Fig. 2 the kinetic energy over time is displayed, where four main movements can be clearly distinguished with the aforementioned threshold applied.

2D and 3D Average Directions

The following step was to obtain the 3D average direction and each of the 2D average directions for each gesture. To do so, two different functions were created.

2D average directions For the 2D projection calculation of the landmarks on the XY-plane, YZ-plane, and ZX-plane, the function "*plot 2D average direction*" was implemented.

The function in Python was designed to analyze and visualize the cumulative directional movement of a specific body part during a certain gesture. It achieves this by first validating the specified gesture and body part against the provided dictionaries. The function then selects relevant movement data from established time frames and calculates the total displacement in each of the X, Y, and Z dimensions. Using this data, it creates and displays vector plots on XY, XZ, and YZ planes, effectively illustrating the overall movement direction and magnitude for the given body part and gesture on different 2D planes.

To call the function, the user can select one of the body parts summarized below, and a gesture sequentially enumerated from 1 through 12.

Table 1: *Body parts mass represented in parts per unit and its corresponding set of landmarks.*

Body part	**Mass**	**Landmarks**
Head and neck	8.59	0-10
Shoulders and thorax	17.61	11-12
Arms and forearms	9.05	13-14
Hands	1.33	15-22
Abdominopelvic trunk	28.62	23-24
Thighs and shanks	31.71	25-26
Feet	3.10	27-32

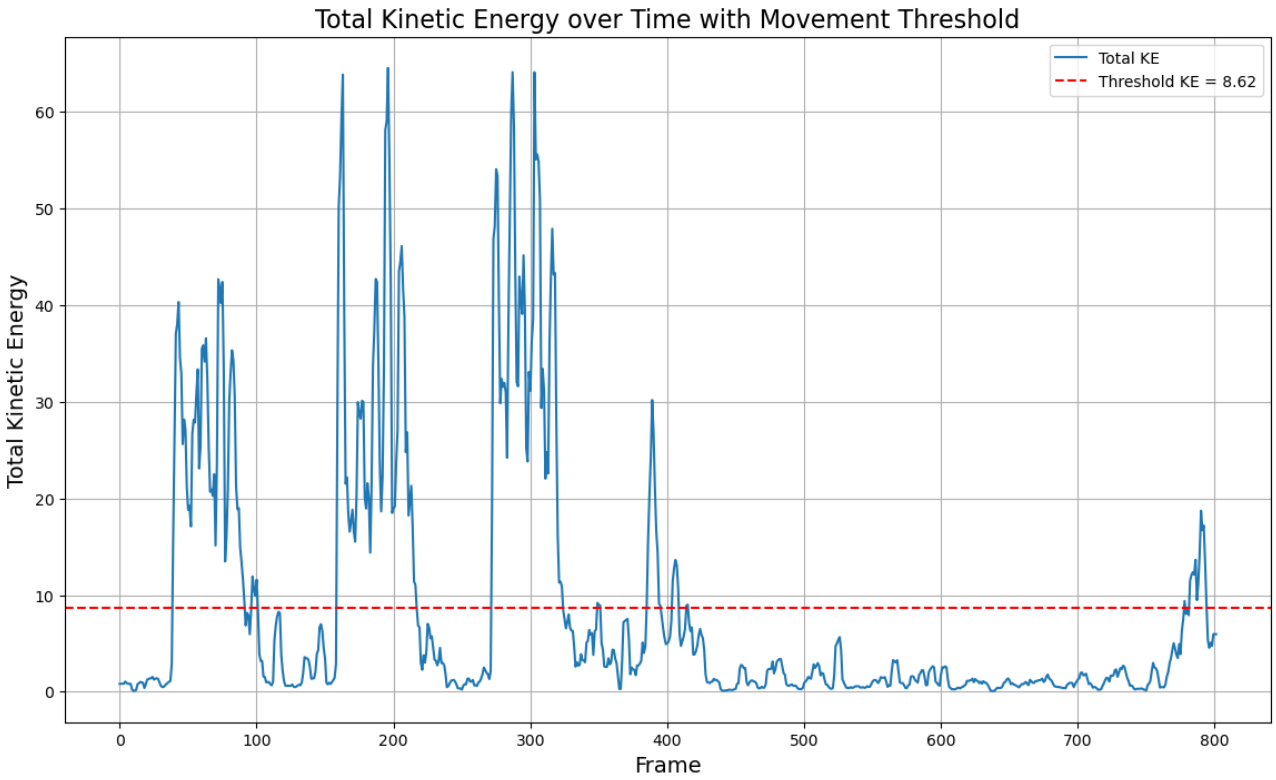

Figure 2: *Total kinetic energy over time with movement threshold.*

Table 2: *Average directions of right arm movements in different planes for units 1 and 2*

Body Part, Unit	Plane	Avg. Dir.	Avg. Dir.
Right Arm, Pause	XY	0.237	0.972
	XZ	0.184	-0.983
	YZ	0.609	-0.792
Right Arm, Movement	XY	0.997	-0.083
	XZ	0.247	-0.969
	YZ	-0.021	-0.999

- Face
- Right Arm
- Left Arm
- Right Leg
- Left Leg
- Upper Body
- Lower Body
- Whole Body

3D average directions On the other hand, to obtain the 3D average direction, another function was created, named "*compute 3d average direction*". This function calculates the three-dimensional average directional vector for a specified body part. It verifies the presence of the body part in a dictionary, extracts relevant data from the data frames representing X, Y, and Z coordinates, and computes the average directional vector in 3D space by determining mean motion vectors in each dimension. The output is the body part's name along with its average direction components in the X, Y, and Z axes.

Results

In this section, the results and the main findings and outcomes are discussed.

First of all, we can state that the kinetic energy of a body can be used to determine whether there is movement or not, by setting an appropriate threshold. However, it is important to mention that this limit must be set according to the characteristics of each video, which might be a limitation of the model. A further study could be done in this aspect to make the model even more effective.

The 2D trajectories were satisfactorily computed and displayed for each plane. An example of the results for the right arm during pause unit 1 and movement unit 2, is shown below.

To go further, a comparative analysis has been made, where we take the first clip window (static pose) and the first gesture (unit 2, a "plié"), and compare the 2D graph trajectories to see and interpret the results. These results can be plotted using the appropriate function, and the results are displayed below in Fig.3 and Fig. 3.

Table 3: *3D average directions for each body part in Movement unit 2*

Body part	Avg. X	Avg. Y	Avg. Z
Face	-0.018	-0.052	-0.998
Right Arm	0.247	-0.021	-0.969
Left Arm	-0.340	-0.053	-0.939
Right Leg	0.607	-0.571	0.552
Left Leg	0.026	-0.049	0.998
Upper Body	-0.034	-0.045	-0.998
Lower Body	0.248	-0.251	0.936
Whole Body	-0.014	-0.071	-0.997

For plane XY (horizontal plane, red row), we can see that when the body is paused, the direction of the right arm points to the top, according to the clip, while when there is movement and the arm is extended to the right, we can see that the vector changes its direction accordingly.

Similarly, the average 3D directions for each plane and each body part were obtained using the function "*compute 3d average direction*". Results are displayed in Table 3.

References

[1] Google. *MediaPipe Pose: Pose Landmarking*. Accessed: November 13th, 2023. 2023. URL: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker.

[2] Riddhi Kumari Singh. *Real-Time Pose Estimation from Video using MediaPipe and OpenCV in Python*. Accessed: November 13th, 2023. Apr. 2023. URL: https://medium.com/@riddhisi238/real-time-pose-estimation-from-video-using-mediapipe-and-opencv-in-python-20f9f19c77a6.

[3] Nick Nochnack. *MediaPipe Pose Estimation Tutorial*. https://github.com/nicknochnack/MediaPipePoseEstimation. 2023.

[4] Gualtiero Volpe. *Body Movement and Gesture*. Accessed: 2023-11-03. 2023. URL: https://2023.aulaweb.unige.it/pluginfile.php/137087/mod_resource/content/3/MS3_2023-2024_body_movement_c.pdf.

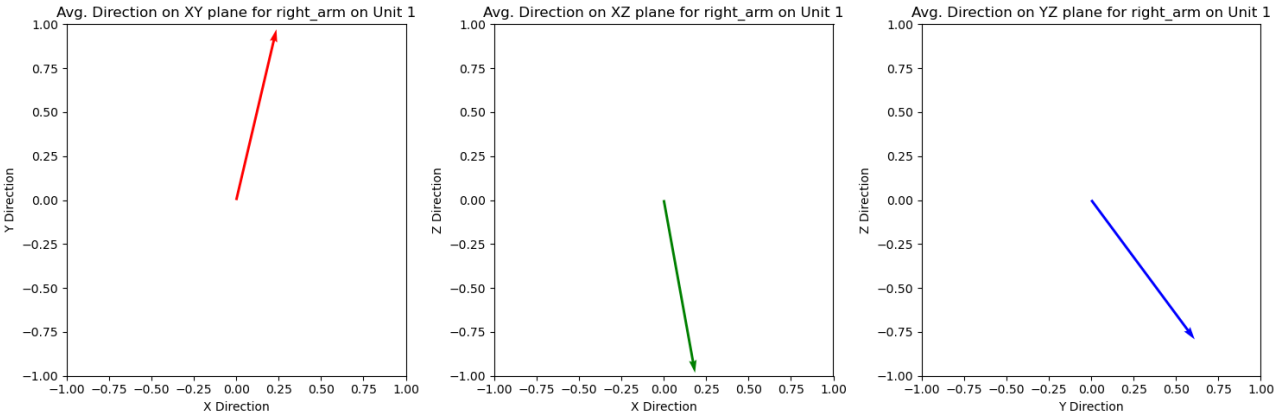

Figure 3: *Total 2D Average Direction plots right arm during pause.*

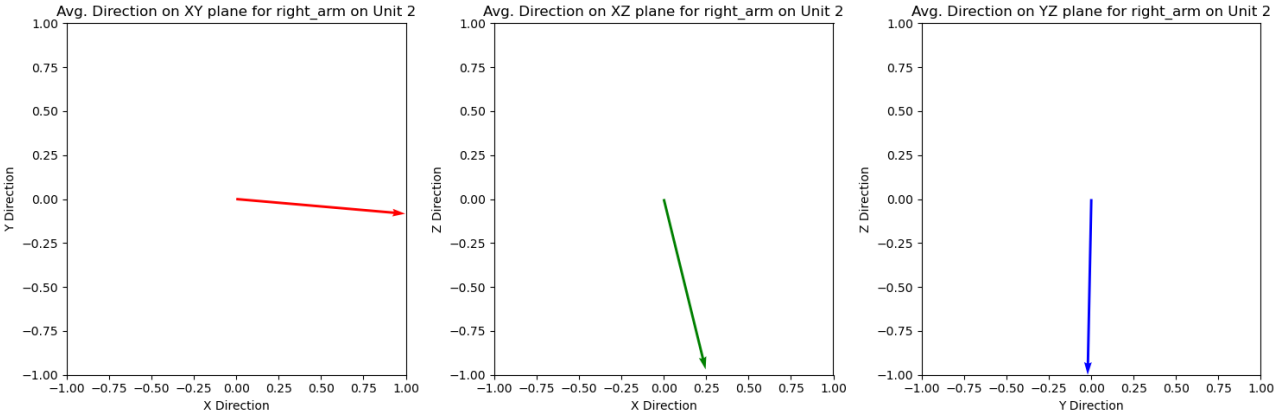

Figure 4: *Total 2D Average Direction plots right arm during movement unit.*