Sean Hayden

12/8/2023

Seizure Detection with Mediapipe

The purpose of this project is to detect a seizure with mediapipe. This is done with the values for the landmarks that are outputted by the mediapipe system which are for the head and shoulders. With the output values, we then use the data to calculate the speed and displacement of the points to detect movement, which can be used to determine whether a seizure is occurring, and for how long.

The first step is to get the feed from the user’s camera, which is done with the cv2 python library. This takes the input from the camera as an image, which is stored in the frame list for processing. When the frame is captured, the time that the frame was taken is stored to measure the interval between frames, which is in turn used to calculate the fps of the feed. The frames are then processed with mediapipe, which outputs the data as a series of coordinates. The camera output is run in parallel with the processing, so a delay is added to the camera feed to account for this time.

To get the image with the landmarks drawn on it, the raw output is parsed into a more user-friendly form. The raw output of mediapipe has a list of landmarks and normalized landmarks. For this program, the normalized landmark is used as the image has a constant width and height. If the image were to be rescaled during the process, then non-normalized landmarks might be better suited. With these landmarks, the raw data has a presence value, which represents how well that landmark is matched with the image. For example, if the legs are not captured within the image and are off screen, the corresponding landmarks will have a low presence value. Using this, only landmarks that have sufficient value are drawn on the image. From here the x and y values are used to draw the present landmarks onto the image, and connecting lines are A screenshot of a computer screen

Description automatically generateddrawn between the connected landmarks.

*The coordinate output of the mediapipe program*

With the image processing function, the coordinates are returned with the image, which is then used to calculate the movement of the points. The output from the function is a large string with all of the x-y-z coordinates of all of the landmarks. With the string, the first step is to separate the list into a list with each entry being an individual landmark, which is done with the split function for ‘landmark’. This makes the string for each landmark like ‘ { x: 0.0… y:0.0… z:0.0… }’. From here we take the index for each of the points that we care about and store them separately. The landmarks that are important are the left and right shoulder, which are 11 and 12 respectively, and landmarks 0 – 10 for the face, specifically 0 for the nose, 2 and 5 for the left and right eye, and 9 and 10 for the left and right mouth. The y coordinates of the shoulders as well as the x values for the head are used for the calculations. To extract these values, we again split the string with either ‘x:’ or ‘y:’ and take the second string in the list. This leaves us with the numerical value that we care about, and the remainder of the string. Finally to narrow this down to the value, split the string with the first space, and convert the first string into a float.

*Mapping of all landmarks on the body*

We store the coordinate value of each of the landmarks over 11 frames, giving us 10 samples of displacement and speed data. To find the maximum displacement of each of the landmarks, we take the difference between the minimum and maximum value. The purpose of getting the displacement of the points is to differentiate random jitter from actual movement. This is necessary because there is some inaccuracy in the calculation of the landmarks, which can differ from frame to frame. However, when there is no actual movement, this jitter from inaccuracy only happens to a certain degree, leaving little to no net movement over a period. To get the speed, we sum up the distance of each landmark between each consecutive frame, then convert this to a speed. To convert this, first the total distance is divided by the sample size to get the average distance between each frame. Then, a time elapsed over the sample duration is estimated with the inverse of frames per second. Finally, the average distance is divided by the time elapsed to give us the speed of each point over the duration. Simplified into one equation, the formula is as follows:

(distance1 + distance 2 + … + distance 10) / (sample size \* fps-1)

For the next frame after the first sample, the oldest value is discarded, and the newest coordinate value is stored in its place. This keeps a running average distance and displacement, which at any given frame represents the most recent 11 frames.

With these values, we can detect rapid movement of shoulders and head. This, however, is not sufficient to detect a seizure, as normal movement can provide a spike in speed and displacement. So, the next step is to detect rapid movement over a prolonged period of time. If the speed and displacement of the shoulders or head exceed a threshold (speed threshold is .5, displacement threshold is .03), a fixed value is added to a respective warning level variable, otherwise a value is subtracted from it. If the warning level is above 80, then a warning is displayed for the shoulders or head. With this, prolonged rapid movements of the head and shoulders can be distinguished between normal movements that can occur in basic activities.

If both the head and shoulder values are above 80, then a seizure is considered to have started. The time that this happened is stored, and the seconds elapsed since the seizure started is displayed for the entire duration. Once at least one of the warning values falls below 80, the end time is stored, and the duration is set. Until a seizure starts again, the duration of the most recent seizure is displayed. A new seizure beginning will then display the duration in the same manner as before.