Process Book

Overview and Motivation

The motivation of the *OpenPose Visualizer Project* is to build a visualization tool for constructing a proof-of-concept for designing an autonomous system that monitors the movements of epilepsy patients and uses machine learning algorithms to predict the onset of a potential epilepsy episode. We speculate that there are certain *movement patterns* that are likely to be indicators of a potential epilepsy episode and thus we should be able to train a neural network to recognize such patterns. Using *OpenPose*, we are able to identify and record the positions of various keypoints on a human body. However, it is hard for us to analyse and evaluate the validity of these data. Therefore, the OpenPose Visualizer aims to represent these data in a way that is intuitive and effective to the human eyes, thus making it possible for us to get a better understanding of the data.

Questions

We want the visualization to help us gain a better understanding of the following questions:

1. Are the data generated by *OpenPose* accurate enough to reconstruct the movement trace?
2. If so, are there any movement patterns distinct enough for the human eyes to recognize during an epilepsy episode?
3. If so, can we use these patterns as indicators of a potential epilepsy episode?

Related Work

1. Understanding Patients' Behavior: Vision-based Analysis of Seizure Disorders https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8629065
2. Convolutional neural networks for real-time epileptic seizure detection https://www.tandfonline.com/doi/full/10.1080/21681163.2016.1141062
3. D3 brush-and-zoom https://bl.ocks.org/mbostock/34f08d5e11952a80609169b7917d4172

Data

A detailed overview of the data generated by the *OpenPose* can be found here. The data used in this project were collected from a 2-min video from the Epilepsy Monitoring Unit at the Vanderbilt Medical Center. I wrote a few python scripts to calculate the Euclidean distance of the same keypoint position between every two consecutive frames for each keypoint in the original JSON file and combine the results into one single JSON file. It is formatted as [d1, c1_1, c2_1, d2, c1_2, c2_2, d3, c1_3, c2_3, ...], d being the distance, and c1 being the confidence level of the point in the first frame and c2 being the confidence level of the point in the second frame. Given the timeline and data validity (*OpenPose* did not recognize the patient's face for most of the frames due to the quality of the video), the *OpenPose Visualizer* will only be using the data from pose_keypoints_2d, hand_left_keypoints_2d, and hand_right_keypoints_2d.

Exploratory Data Analysis

Initially, I took all the data points in the pose_keypoint_2d and used a line chart where the x-axis represents the frame number and y-axis represents the Euclidean distance. It did not look quite good since there were about 3600 data points and the scale got stretched out by a few outliers so majority of the points were just clustered around the bottom of the chart. This initial design made me realize that the visualization to have more control over the selection of data points. Therefore I decided to add the filter bar and the brush-and-zoom features.

Design Evolution

11/05/2019

The basic layout of the visualization will be three *line charts* with *coordinated views*. The *y-axis* of each line chart will correspond to the *Euclidean distance* and the *x-axis* will correspond to the *frame number*. The user should be able to *zoom in and out* around a certain frame and *select* a group of keypoints to *highlight*. Furthermore, if possible, the user should be able to add more plots by clicking the *Add* button and choosing a specific keypoint. Finished the basic layout structure of the visualization. Added the pose plot. Continuing working on the dropdown menus and the filter bar. Right now everything is hard coded and I can see that this will result in a lot of redundant and ugly code in the future so that needs to be fixed soon.

11/10/2019

The *add-a-plot* feature was canceled at this time unfortunately due to the timeline of the project. I also decided to use scatter plots instead of line plots since the data are discrete. However, I decided to add a *filter bar* to the visualization since the initial visualization did not look good due to the fact that a few outlier data points stretched out the scale and make the rest of the data points really hard to see. The newly added filter bar has a *draggable* line that represents a *threshold* confidence value. Only data points whose overall confidence value is above this threshold will be plotted. The plots will be *updated and rescaled* after the user updates the threshold confidence value. Finished implementing the *filter bar* and the *dropdown menus*. Added *coordinated views* so that whenever the user selects a new threshold confidence level, all three plots will be updated and rescaled accordingly. Rewrote most of the code responsible for rendering the plots to make it reusable and robust enough to support potential addition of visualization features. Added some css for the texts and the dropdown menus.

11/18/2019

Since the visualization uses dropdown menus for the selection of the keypoints, a drawback with this approach is that the user does not know which keypoint to select at first. *Dr. Bergers* suggested to visually communicate the distribution of each keypoint, rather than a single number, e.g. mean and variance, where the latter would seem important for understanding what body parts are rapidly changing. However, the problem with this method is that even if the variance of a keypoint is high, it could still have *two* possibilities: 1. The position of the keypoint was changing rapidly; 2. *OpenPose* did not know what it was doing and generated a lot of garbage values with low confidence. Since the user would not want to look at keypoints with low accuracy, I thought it would be better to encode the overall confidence level of each keypoint. Although I did not come up with a way to effectively communicate this to the user before the selection (mostly because there are *over 60 keypoints* and it is hard to fit them all in a single web page), I did manage to *color-code* each data point based on its own confidence value using a diverging color scale. Therefore, after choosing a keypoint, the user will at least be able to decide if it contains valuable information for a more detailed look. Finished implementing the *brush-and-zoom* and *color-coded* each data point based on its overall confidence level. Added *reset view* button that zooms out the plots to its original view while *retaining* the threshold confidence level. Added *color legends* and *axis* for the filter bar. Added *labels* to each plot.

Implementation

Effective use of visual channels:

1. Color

Using a threshold scale, data points with different range of overall confidence level will be colored accordingly. Furthermore, since the threshold was implemented with a diverging color range, data points with low confidence level will be easily distinguished from those with high confidence level. Therefore, at a first glance, the user will receive constructive visual feedbacks on the validity of the data of the keypoint selected.

2. Spatial position

By mapping each data point to a specific position on a 2D plane, the user will be able to see the relationship between the movement trace and the frame number, making the process of observing potential patterns more straightforward. Furthermore, it also makes it easier for the user to observe the range of the data points, i.e. the maximum and minimum value over a specific amount of frames.

3. Interaction

a. Drag and filter

By dragging the threshold line, the user can easily filter out data whose confidence level is below the selected threshold value. This will allow the visualization to become more effective since it will eliminate outliers that stretch out the scale and reconstruct a more accurate representation of the remaining data points.

b. Brush and zoom

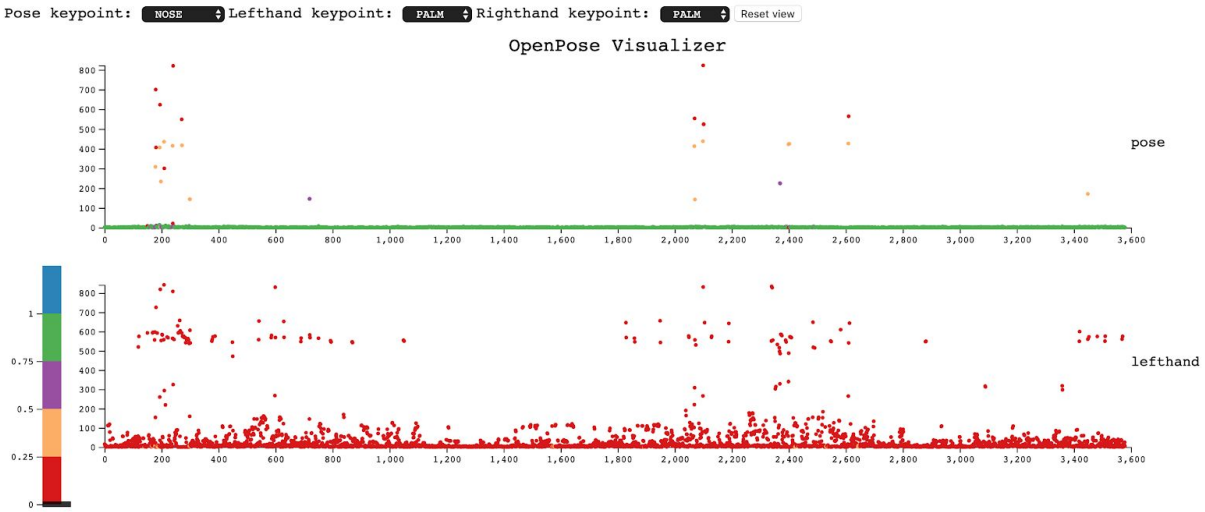
Since there are about 3600 frames, it is hard to see a specific area of the distribution of the data points. Therefore, having a way to interactively modify the view is necessary to obtain a more detailed information on the data. By brushing along the x-axis, the user will be able to zoom in on the plot around the selected frames and the plot will be rescaled accordingly.

c. Multiple and coordinated views

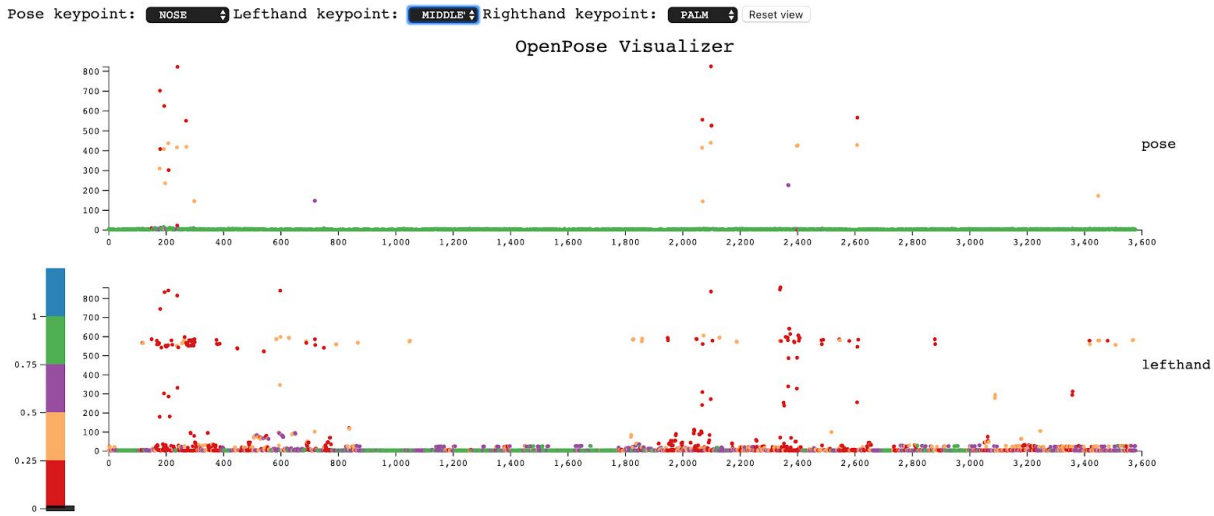
By having a coordinated view between the three plots, it is easier for the user to observe the relationship between different body parts.

Reference images:

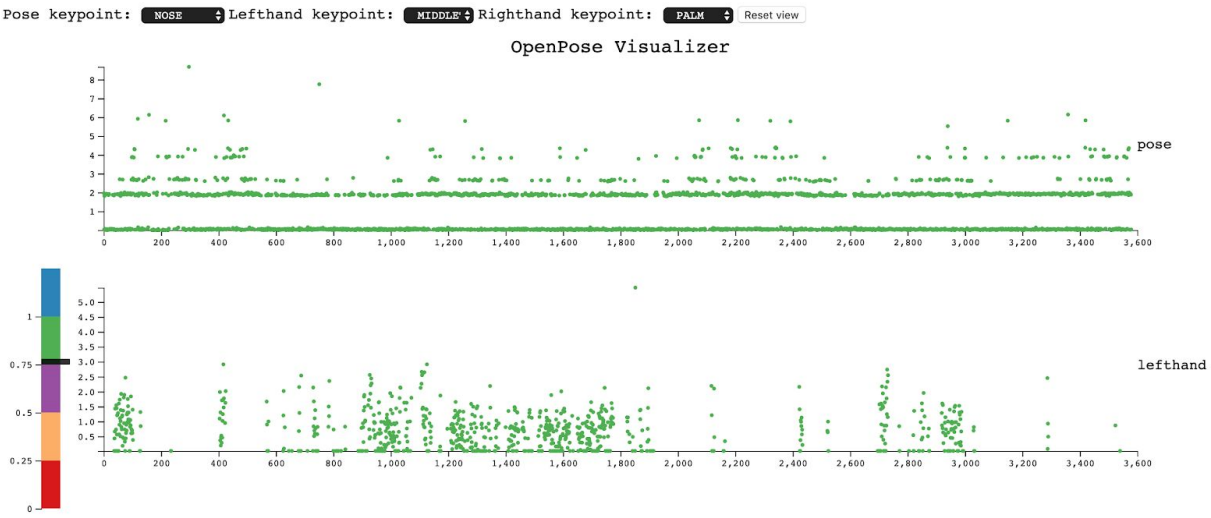
1. Visual feedback . When the user selects palm as the keypoint for the lefthand plot, the majority of the data points are red, meaning that they are low in confidence level. Therefore the user knows to select a different keypoint to plot.



2. Diverging color scale . Data points with high confidence levels have a very different hue than those with low confidence levels. Therefore the user can easily distinguish between them. As seen below, when the user chooses the tip of the middle finger as the keypoint for the lefthand plot, there are more points with higher confidence levels.

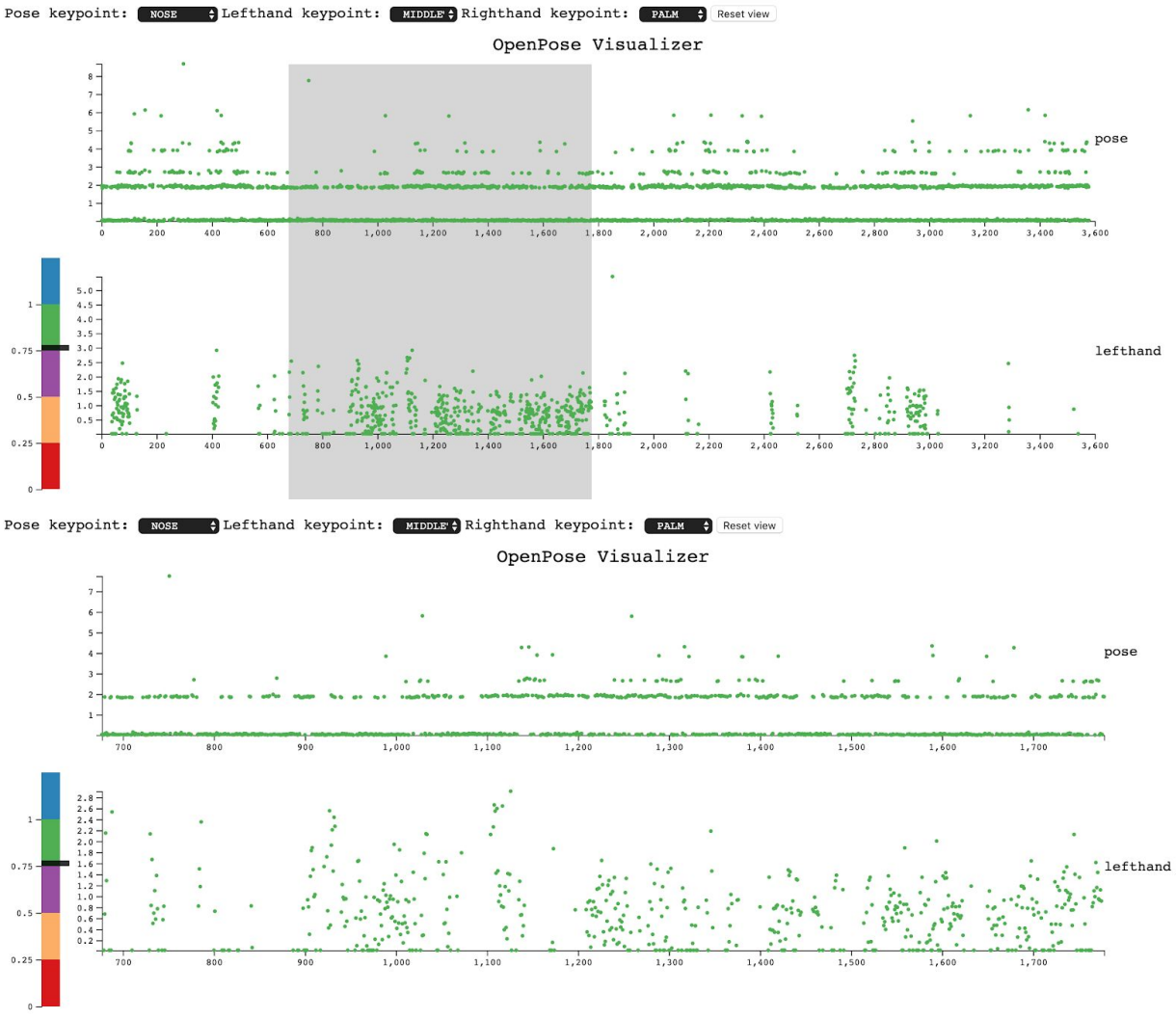


3. Filter data . By dragging the threshold line, the user will be able to filter out data points whose confidence level is below the threshold line and have a better view of the remaining points as the plots rescale.



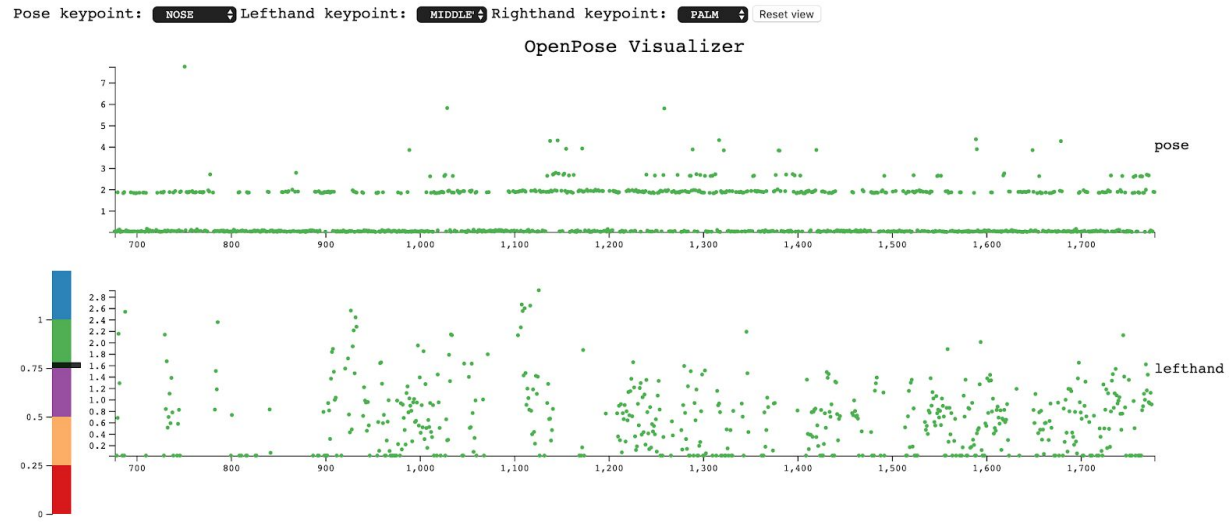
.

4. Brush and zoom . The user can brush along the x-axis to restrict the view around a certain number of frames to get a more detailed view.

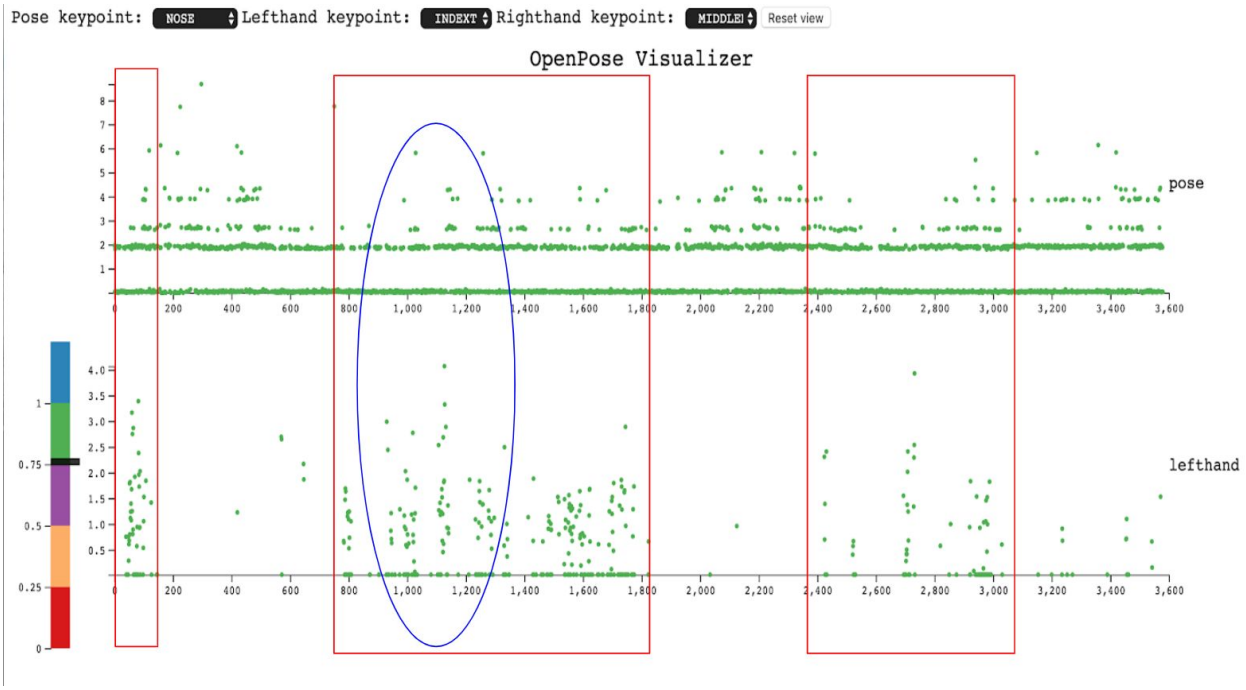


Analysis

I did discover some kind of pattern in the distribution of the data points. Below is the area where the epilepsy occurred in the video.



As seen in the visualization, the pose plot displays a rather stable movement of the keypoint (keypoint bounces between 0 and 2 for most of the frames which means it was barely moving). Whereas the distribution of the lefthand keypoint is rather chaotic. This could result from the fact that when the patient was having the epilepsy episode, his hands were twitching but the rest of his body was still. Zooming out the plots back to their original views, we can see that there are three areas that contain such pattern. If we can see such pattern occurring on other epilepsy patients, we should be able to train a neural network to recognize it.



Hence, to answer the questions outlined in the beginning:

1. We are able to reconstruct the movement trace based on the data for a few keypoints with high confidence levels. This really depends on the quality of the video.
2. Yes, at least for this particular video, we are able to see the pattern described above.
3. We are not sure since we do not have more videos available to check whether this particular pattern occurs in other EMU videos. Overall, the visualization did help us understand the data generated by *OpenPose* better. We saw some kind of patterns in this video but due to the lack of data, we do not know if this particular pattern is applicable to other EMU videos. However, we will be able to answer this question once there are more data available.

Improvements

1. There still isn't a way to communicate each keypoint's information before the user makes a selection.
2. For this project to go beyond the scope of this class, the web app should simply takes a uri to a folder containing the raw data generated by the OpenPose, run the processing scripts and display the visualization. Also it should be able to add more plots.