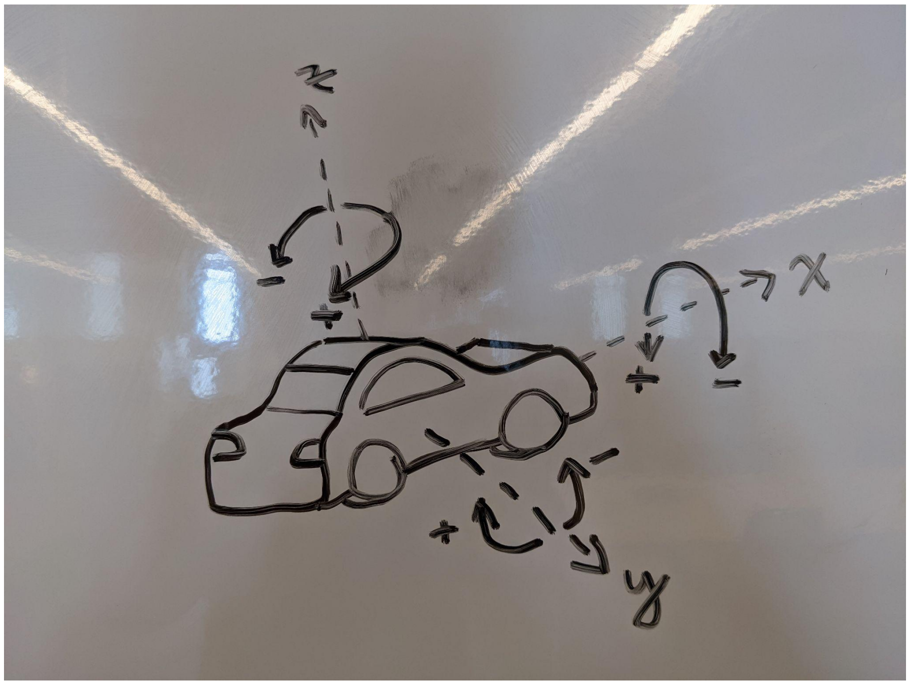
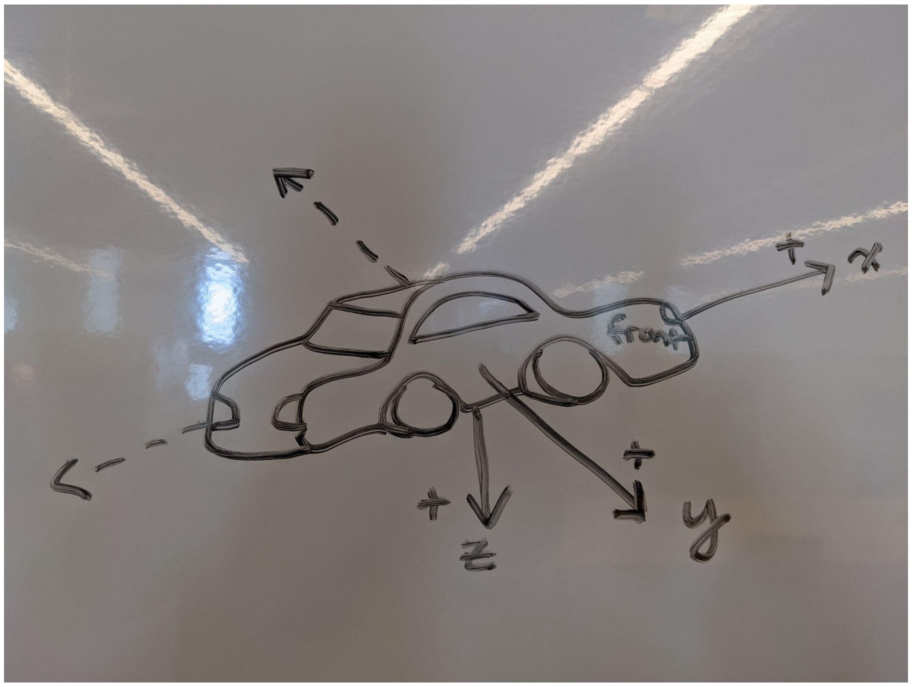
RIDWAN HOSSAIN
100747897

2022/06/15

- **Will continue from yesterday and complete the following items:**
    - Keep adding to the scripts to visualize the remaining data left to be represented.
    - Produce more data for other maps and use the scripts to visualize some of the data.
    - Produce an informal script for the demo on Tuesday.
    - Continue to make test cases for the additional functions I created (CAN messages and CAN attacks if possible).
- Will add **collision data** to the graphing script next.
- Discovered that many different collisions of different intensities are recorded for the same recording time. This is most likely a result of Carla's impact physics model. This results in bar charts that have strange values that end up stacking on top of one another without any order.
- To resolve this issue, I decided to order all the collision data based on their intensity, such that the highest value recorded for a specific time would be last (ascending order). After doing so, pandas' built-in **drop_duplicates** function allowed me to drop all duplicate rows based on the Rec_time (recording time), while keeping the last row with that Rec_time. This results in the highest intensity collision for each time being recorded.
- After performing the above steps, the collision data was represented with much better coherency.
- The next step is to implement **gyroscope data** into the graphing script
- Each gyroscope reading is technically a string and a tuple, so I **need to clean up the data** and convert each string to a proper numeric list. To do that, I removed all white spaces, all brackets, and split the tuple by a delimiter. After splitting the data item, I converted each coordinate of the gyroscope reading into a float value.
- The plan is to plot the gyroscope data into **3 lines** on a graph, each line corresponding to the **angular velocity** (radians/second) in the **x**, **y**, and **z** planes.
- As the Carla documentation did not provide a detailed description of how exactly the gyroscope readings are measured, I had to manually test vehicles and interpret the data. I went to a map containing hills and quickly changing elevations (**Town06**) in order to perform such actions as take jumps, climb and descend hills, and roll the vehicle. From the testing, the following rough diagram was drawn:



- The x, y, and z axis are labeled based on their position in the gyroscope readings. It is read as **(x, y, z)**. The following characteristics were observed which resulted in the above diagram:
    - Turning the vehicle sharply right resulted in a positive z value and vice versa
    - Causing the vehicle to roll clockwise relative to the above x axis resulted in a negative x value and vice versa
    - When going on jumps and climbing hills, as the vehicle lands and the nose is pushed upwards, the y axis reads a negative value. When the vehicle leaves the end of the platform during the jump, the nose slightly dips down, resulting in a positive y value.
- Now that I have made sense of the gyroscope readings, this information can be used to help any team that decides to use this data later on.
- Afterwards, I was able to plot each individual axis as a separate line on the graph for gyroscope readings. The next step is to implement **accelerometer data** to the graphing script.
- As the reading of the accelerometer data is formatted the same as the gyroscope data, the same steps are required to clean up the data.
- Once again, the Carla documentation lacks a detailed description of how exactly the accelerometer readings are measured, other than they read the **linear acceleration along an axis** in **meters/second^2**.
- Once again, manual testing was performed and the following rough graph resulted:



- The following characteristics were observed which resulted in the graph above:
    - When accelerating forward, the reading on the x axis is positive and vice versa. This can also be interpreted as acceleration being felt, pushing backwards.
    - When turning right, the reading on the y axis is positive and vice versa. This can also be interpreted as acceleration being felt, pushing to the left.
    - The z axis reading is nearly always positive **9.8**, with slight changes when the vehicle moves sporadically. This shows that the downwards acceleration resulting from the earth's gravity is a positive 9.8, thus, acceleration pushing downwards is positive.
- Now that I have made sense of the accelerometer readings, this information can be used to help any team that decides to use this data later on.
- Afterwards, the accelerometer was successfully graphed in the same manner as the gyroscope data.
- **A crucial realization was made**. It was discovered that the collision data, obstacle data, and lane invasion data do not carry the autopilot flag with them, which would provide useful information when testing autonomous vehicle algorithms.
- I could feasibly merge the data with the vehicle telemetry data to achieve this, but the reason I separated these CSVs in the first place was because collisions, lane invasions, and obstacle detections can possibly occur outside the sampling rate of the vehicle telemetry.

- To resolve this issue, the data logging pipeline was revised to include the autopilot flag in all CSVs, and I recorded another run of data.
- Now that the autopilot flag is included in every CSV, I will now work on implementing the Autopilot flag into the existing Server & Client performance (to see how much the autopilot functions affect the system performance) and Collision Graphs.
- Implementing the Autopilot flag into the existing graphs turned out to be a MASSIVE headache, not because of the data itself, but because of how hover text and labels work in plotly. This took lots of time and research to make it look nice.
- The last items left to graph are Obstacle detection and lane invasion data.