RIDWAN HOSSAIN
100747897

2022/06/09

- Noticed there was an error where a certain section of the code was commented out but was not supposed to be. Reinserted that code into the simulator, and attempted to push, however, github desktop was having authentication issues and would not allow for a push.
- Looked into the issue online, and found that the best course of action would be to utilize a Personal Access Token opposed to using the normal login.
- After some tinkering, I was able to update global git configuration and save a Personal Access Token, allowing me to push the change from the CLI.
- One small issue now is that for every commit, I need to re enter the username and Personal Access Token, which becomes very tedious. Looking into resolutions for this matter.
- Used the command **git config credential.helper store** to store authentication data so committing changes would be much easier.
- **Note:** The new streamlined process for pushing changes to the repository is to commit using the github desktop application to easily view changes, and then push the changes using the CLI.
- To make sure no new issues arose with the system, ran all tests and ran CAN attacks to make sure everything is still working as usual.
- After running all tests and checking everything, all subsystems and functionality is working properly. However, one small issue was discovered, one of the tests left an extra vehicle actor spawned within the simulator. Need to find which test it is and resolve the issues.
- Found that the test that was leaving a duplicate vehicle was **Hand_Brake_Test**. After extensive modifications, it was not found as to why this one test was behaving differently from all the other tests, particularly when this test was built the same as the others. As this was not a critical issue, this problem will be deferred to be solved later as it is more important to continue the work that was being done yesterday.
- **The first piece of data I will work to export to a csv is the server and client fps.**
- Was able to access the server and client fps and export them to a global scope within ‘testbed.py’. The next step is to set up a **pandas** dataframe, add the data to the dataframe, and export the dataframe for viewing.
- Needed to choose a sampling rate for collecting the data, as obtaining the data for every clock cycle of the cpu would be overkill and result in datasets that are far too large.

Decided to try a sampling rate of **120 samples per minute** or 1 sample every half of a second.

- Was able to insert the server and client fps readings into a dataframe at 120 sampler per minute using the pandas **append** function. The next step is to export this dataframe to a csv once the user enters the keys to stop recording.

- I implemented a logging mechanism for the datasets such that the user can continuously log data without needing to change filenames, the mechanism already checks for existing filenames and automatically produces a new one.

- Now that a pipeline for obtaining and sampling the data, recording the data, generating a dataset, and exporting that dataset has been constructed, the next step is to obtain the rest of the remaining data that is to be used in the datasets.

- **Notes from brief meeting with supervisor (IMPORTANT):**
    - He said he'd reach out to the previous capstone team to check if they have any existing Jenkins framework related files

    - He said for the demo with the company members, they would be interested in logging data from autonomous driving functions, like I'm driving around in manual mode, its collecting data (including obstacle and pedestrian detection), and then I switch to autonomous mode, and it continues to collect that data (it can have a column in the data that says whether autonomous functions are active or not). **This is the main work to be done.**

    - He said I don't have to worry about developing the autonomous driving functions from scratch, he said **Joelma** has those ready to go, so I should email her to get those. My job would be to log the data and capture it.

- Was able to add the vehicle speed and timestamp to the logs.

- In order to have timestamps be coherent with a video being paired with the data, in the future I would like to launch a screen recording when the simulator begins recording data. This source may be helpful: https://www.geeksforgeeks.org/create-a-screen-recorder-using-python/

- Added compass heading, accelerometer data, gyroscope data, and location data to the logs. The remaining telemetry to be implemented into the logging pipeline are:

    - GNSS data

    - Height data (can be graphed) (not useful on flat maps)

    - Throttle data (can be graphed)

    - Steering data (can be graphed)

    - Brake data (can be graphed)

- Collision data (Can be graphed)
- Line breakage data