RIDWAN HOSSAIN
100747897

2022/06/13

- Will continue where I left off last time, which is looking into how to implement an obstacle sensor into the Carla bench.

- I found the following sources that may be helpful for implementing an obstacle detector:
    - https://carla.readthedocs.io/en/latest/ref_sensors/#obstacle-detector
    - https://github.com/carla-simulator/carla/issues/2863
    - https://carla.readthedocs.io/en/latest/tuto_G_retrieve_data/

- Was able to get the obstacle sensor initialized, attached to the player vehicle, and read telemetry data using the sources above. One small issue is that the range of the obstacle detection has not been optimized, and as such, the detector is reading everything within a large range of the vehicle as an obstacle. Need to do some manual testing to find an optimal value that would provide relatively accurate readings and warnings.

- Found that a roughly optimal distance was approximately 20 meters for detecting obstacles ahead of the vehicle, however, to perform better testing, I would like to add some more dynamic objects to the simulation world. To support this, I will look into how to add traffic and pedestrians to a Carla world. The following source may be useful: https://carla.readthedocs.io/en/latest/adv_traffic_manager/#creating-a-traffic-manager

- Was able to utilize the documentation linked above and the **generate_traffic.py** file in order to spawn in **30 vehicles** and **10 pedestrians** into the Carla world. The obstacle sensor is working as expected, and is able to detect objects such as traffic vehicles, pedestrians, poles, and so on, within the detection range. It was found that detection range was too short, since approaching a vehicle at speed may result at 20 meters being too short of a reaction distance. Changing this to 30 meters produced a better result.

- Now that obstacle detection has been implemented into the Carla bench, the next step is to log sampled data of the obstacle sensor into the data logging pipeline. It was also decided to have the obstacle detection data be exported to a separate csv as obstacles may be read outside the sampling rate.

- After adding obstacle detection data to the data logging pipeline, the next step is to look into how to implement autonomous driving functionality into the Carla bench. As **Joelma** had told me over email, there exists a file provided by the Carla development team, called **automatic_control.py** which spawns in a vehicle which will autonomously navigate to a randomly selected location on the map.

RIDWAN HOSSAIN
100747897

- My goal is to implement the functionality from automatic_control.py into the Carla bench (testbed.py) in such a manner that the user may push a button and the vehicle will select a random destination on the map and begin autonomously navigating to it. If the user pushes the button again or if the vehicle reaches its randomly chosen destination, it will switch back to the manual driving mode.

- I would like to map the right paddle on the steering wheel to toggling the autonomous driving functionality, as such, I need to alter the button mapping within testbed.py as well as the ADB_LaunchApp_Test test case as that test checks if the right paddle opens a specific application on the Android head unit.

- Updated ADB_LaunchApp_Test to only test the launching of the Google Maps application by simulating the left paddle on the steering wheel. This leaves the right paddle available to be used to enable the autonomous driving functions.

- After extensive work and multiple iterations of attempts and testing, I was unable to adapt the method automatic_control.py used for autonomous driving. It would not register the agent calls or apply those controls to the player vehicle. I will research other ways to implement autonomous driving functionality into the Carla simulator without using the methods used in automatic_control.py.

- After some searching, the following source was found that may help add autonomous vehicle functionality to the Carla bench:
  https://forum.carla.org/t/autopilot-in-sync-mode/649

- On the forum, it was noted that calling the **world.player.set_autopilot(True)** statement must be performed every loop of the game tick as the actions within the tick function sets autopilot to False.

- Implementing the above statement finally allowed autonomous driving to be enabled in the Carla simulator, however, having to execute this specific statement for every tick of the loop results in the client performance (fps) to drop to roughly 15 fps, about half of the usual performance. This is a very minor issue however, as the refresh rate is not as crucial when the vehicle is driving autonomously. Another upside is that opposed to the method used in automatic_control.py, this form of autopilot allows the vehicle to autonomously drive endlessly and not simply to a random checkpoint.

- An issue discovered was that having a set_autopilot statement to disable the autonomous driving running for every tick would reduce the refresh rate to a low number even when the user is manually controlling the car. This can easily be fixed using flags and counters to hold state information.

- The issue above was resolved and the performance issue no longer exists when in manual driving mode. As such, autonomous driving was now successfully implemented into the Carla bench and can easily be turned on and off using the right paddle on the steering wheel.

- The next step is to add whether the autopilot is enabled in the vehicle telemetry logs for the data logging pipeline, pair a screen recording function with the data logging, generate some sample data for a map, and develop some scripts to process and visualize the data which makes sense to be graphed. After this is done, produce more data for other maps and use the scripts to visualize some of the data, afterwards produce an informal script for the demo, and then continue to make test cases for the additional functions I created.