Project Report: Using ROS2-CARLA Bridge to Integrate LiDAR into CARLA

Author: Ruifeng Zhang | **Unity ID:** rzhang38

Webpage: https://invimirage.github.io/CSC791Project/

1. Achieved Goals

1. Integrate LiDAR device with ROS2

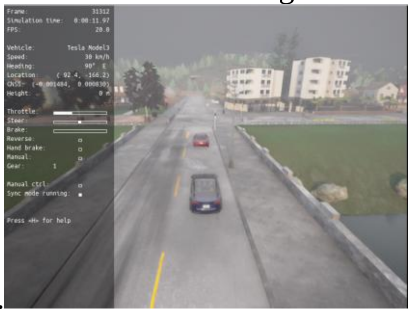
- Used usbipd to attach the LiDAR device to WSL subsystem via USB.
- Used rplidar_ros.view_rplidar_a1_launch to publish the scanning results.
- Implemented node to read scanning data.
- **Test case:** with LiDAR device linked to device, run
  ros2 launch rplidar_ros view_rplidar_a1_launch.py serial_port:=/dev/ttyUSB* serial_baudrate:=115200
  where ttyUSB* is decided via ls commands
- **Results:** A sample of scanning results is in scan_sample.txt

2. Use ROS2-Carla bridge to stream LiDAR data for Carla Agent

- Implemented a ROS2 node to read LiDAR data and parse it to detect objects.
- The node also publishes the results to ROS-Carla bridge agent via topic **/detected/obstacle_relative_position**
- **Test case:** with LiDAR running (check ros2 topic echo /scan), launch node with ros2 launch lidar_to_carla lidar_bridge_launch.py, then run: **ros2 topic echo /detected/obstacle_relative_position**. Place some obstacle in front of the LiDAR and then the obstacle relative position information would be shown**.**

3. Visualize the detected obstacle dynamically in Carla Simulator

- Built pipeline to connect the local ROS-Carla bridge to remote Carla server, and launch the local ROS agent to perform autonomous driving in the simulator.
- Based on ROS-Carla bridge's ad_agent, developed my agent functions to read the obstacle information from LiDAR.
- My agent also spawns an invisible car (at some elsewhere on the map) in the beginning. When the obstacle appears, it immediately moves the car to the obstacle's position. When it disappears, it moves the car back to be invisible again.

- My agent also updates the position of the obstacle if it continues to exist, also, the bearing, distance from the driving vehicle, and the relative position changes real-time with the object moving in the real world.
- **Test case:** with the previous modules running, launch remote Carla server (same as previous homework). Then use **ssh -i path_to_ssh_pub_key -L 2000:cxx:2000 -L 2001:cxx:2001** to bind the local port with remote server. Then run: **ros2 launch carla_ad_demo carla_ad_demo.launch.py.** It automatically links to local host and thus forwards to remote Carla server. The response frames from the remote simulator will be displayed by rviz2, and with WSLg installed it automatically shows up on local PC. It will display a running autonomous driving car, then place some metal obstacle in front of the LiDAR device (note: must be at the front since I only detect objects within 90 degrees of FOV), adjust the position to be not too far or too close, then the obstacle vehicle would appear in front of the driving vehicle. Here is a screenshot and more details are in the slides.

```
Frame:                    31312
Simulation time:    0:00:11.97
FPS:                       20.0

Vehicle:          Tesla Model3
Speed:                 30 km/h
Heading:                 90° E
Location:       ( 92.4, -166.2)
GNSS:  (-0.001484,  0.000830)
Height:                    0 m

Throttle:
Steer:
Brake:
Reverse:
Hand brake:
Manual:
Gear:        1

Manual ctrl:
Sync mode running:

Press <H> for help
```

2. Open Problems

1. Instable & slow network connection

- Although I tried to minimize communication, like just sending the position of obstacle instead of all LiDAR cloud data, using remote Carla server still is unstable. There were many network failures on both ends. Using local Carla, to me, would definitely be more favorable.
- **Delayed response.** Another problem caused by this is that the change of real-world object's position would be immediately reflected by the simulator. The FPS I got was pretty low and one primary cause is that Carla simulator will have to send images back to my PC to display.
- **Solution:**
  - Use a local Carla Server
  - Faster network and faster server machines
  - Make Carla directly output the simulation animation or use techniques like frame compression to make it faster.

2. LiDAR detection not so accurate

- Current LiDAR could only detect one object steadily, and it requires the object face directly towards the LiDAR. The device only provides 128 ranges, so at FOV=90 there are only 32 data points, which are two few for more complicated tasks like object classification. Also, the current LiDAR just output 2-D results without heigh information.
- **Solution:**
    - Use more advanced devices, then incorporate with better algorithms. However, more computation workload likely requires finer optimization.
    - Use a combination of LiDAR and RGB cameras.

3. Replacing the virtual LiDAR in Carla

- One of the original goals was to use the physical LiDAR as a substitute of the virtual LiDAR in Carla, but it’s not feasible since I don’t have the same simulation environment in real life. So I could only scale up the distance and simulate the obstacle, but it’s definitely more fun to have a real simulation environment**.**
- **Solution:**
    - Not very sure how to achieve this level of simulation.

4. Wrong place to put the obstacle

- Because the LiDAR doesn’t have knowledge of the Carla world, it may put objects in wrong places. For instance, if you move the object too close up the LiDAR, there could be collision between cars.
- **Solution:**
    - Implement a two-way communication: the LiDAR processing node should also know the environment information. But if a physical simulation (problem 3) could be established, using object detection algorithm should serve the same purpose.

5. More advanced functions

- Because of the aforementioned limitations, more advanced functions, like planning a route a round an obstacle, automatic cruise control, etc, are hard to implement.
- **Solution:**
    - Covered before.