CEE 678 CARLA Lab 4: Set Up RSU and Use Customized Map

Prerequisites:

1. Success installation and running of CARLA. (version 0.9.12 is used in this tutorial)
2. A Python IDE to edit the Python scripts.

Load the CARLA Server and Client:

Server side:

1. Open the Command Prompt window, type or copy and paste the two commands and press Enter:
   ```
   >> cd C:/{Your_Directory}/CarlaSimulator
   >> CarlaUE4.exe
   ```
   Since running CARLA requires strong graphical computing power, we recommend you run CARLA with low graphics quality to speed up the simulation:
   ```
   >> CarlaUE4.exe -quality-level=Low
   ```

Client side:

1. Open a **new** CMD window, type the following commands to run the example Python client:
   ```
   >> cd C:/{Your_Directory}/PythonAPI/examples
   >> python manual_control.py
   ```

2. To run the customized code, you can type the following commands in the same way:
   ```
   >> cd C:/{Your_Directory}/PythonAPI/examples
   >> python rsu.py
   ```

Procedures to Set Up the RSU:

Launch the server and follow steps in the example code `rsu.py.` Feel free to change the parameters.

1. Initialize the blueprint library and the spawn points
   ```
   blueprint_library = world.get_blueprint_library()
   spawn_points = world.get_map().get_spawn_points()
   ```

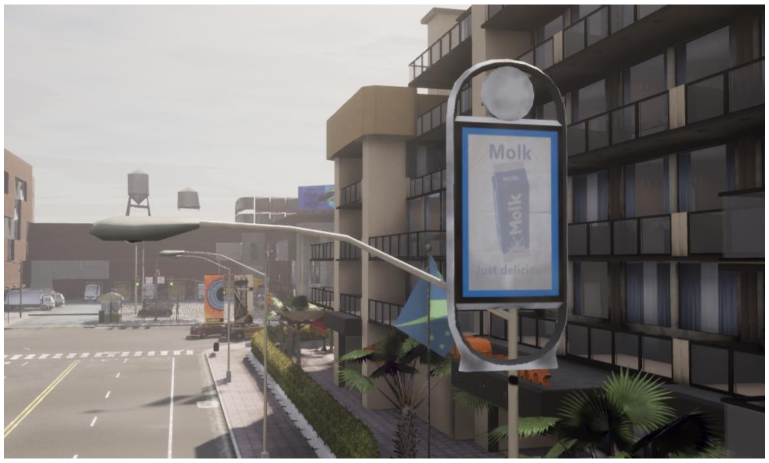
2. Iterate and generate NPC vehicle with a total number of **NPC_VEH_NUM**.
   ```
   # Generate NPC vehicles
   for i in range(NPC_VEH_NUM):
       # Choose random blueprint and choose the i-th default spawn points
       vehicle_bp_i = random.choice(blueprint_library.filter('vehicle.*.*'))
       spawn_point_i = spawn_points[i]

       # Spawn the actor
       vehicle_i = world.try_spawn_actor(vehicle_bp_i, spawn_point_i)
   ```

```
    # Append to the actor_list
    if vehicle_i != None:
        actor_list.append(vehicle_i)
        vehicle_list.append(vehicle_i)
    print('%d vehicles are generated' % len(actor_list))

    # Set autopilot for each vehicle
    for vehicle_i in actor_list:
         vehicle_i.set_autopilot(True)
```

3. Spawn the RSU at a selected location.

```
rsu_bp_1 = blueprint_library.find('static.prop.streetsign')
spawn_point_rsu1 = carla.Transform(carla.Location(x=-57, y=61.22, z=6.5),
carla.Rotation(pitch=0.000000, yaw=0.000000, roll=0.000000))
rsu1 = world.spawn_actor(rsu_bp_1, spawn_point_rsu1)
actor_list.append(rsu1)
```

Right now, we use a street sign to represent the RSU device installed on the street light.



4. Set the view of spectator to the RSU.

```
spectator = world.get_spectator()
rsu_transform = rsu1.get_transform()
spectator.set_transform(carla.Transform(rsu_transform.location,
carla.Rotation(pitch=-35)))
```

5. Set construction cones. You may customize the location of the cones.

```
cone_list = []
cone_bp = blueprint_library.find('static.prop.trafficcone01')
spawn_point_cone1 = carla.Transform(carla.Location(x=-53, y=58, z=0))
spawn_point_cone2 = carla.Transform(carla.Location(x=-51, y=58, z=0))
spawn_point_cone3 = carla.Transform(carla.Location(x=-51, y=61, z=0))
spawn_point_cone4 = carla.Transform(carla.Location(x=-51, y=64, z=0))
spawn_point_cone5 = carla.Transform(carla.Location(x=-53, y=64, z=0))
cone_1 = world.spawn_actor(cone_bp, spawn_point_cone1)
cone_2 = world.spawn_actor(cone_bp, spawn_point_cone2)
cone_3 = world.spawn_actor(cone_bp, spawn_point_cone3)
cone_4 = world.spawn_actor(cone_bp, spawn_point_cone4)
cone_5 = world.spawn_actor(cone_bp, spawn_point_cone5)
```

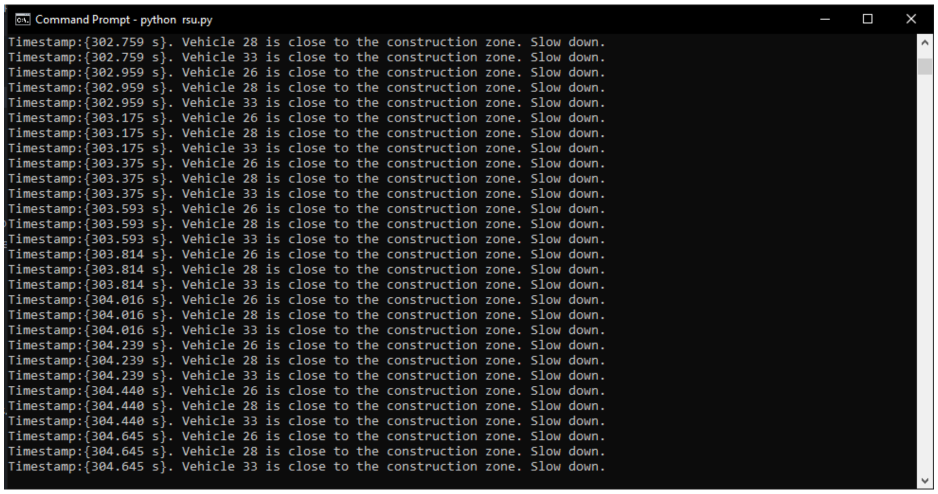
```
cone_list.append(cone_1)
cone_list.append(cone_2)
cone_list.append(cone_3)
cone_list.append(cone_4)
cone_list.append(cone_5)
```

We will see five cones installed in this area.

6. Use a while loop to check the distance between the vehicles and the construction zone.

```
while True:
        world_snapshot = world.get_snapshot()
        timestamp = world_snapshot.timestamp.elapsed_seconds

        # Set the detection range
        detection_range = 10

        # Calculate the distance between each vehicle and each cone
        for i in range(len(vehicle_list)):
            vehicle = vehicle_list[i]
            v_x = vehicle.get_location().x
            v_y = vehicle.get_location().y

            for j in range(len(cone_list)):
                cone = cone_list[j]
                c_x = cone.get_location().x
                c_y = cone.get_location().y

                # Calculate the Euclidean distance
                dist = np.sqrt(np.square(v_x-c_x) + np.square(v_y-c_y))

                # If the vehicle is in the range, assume RSU could send this
message to the vehicle.
                if dist < detection_range:
                    print("Timestamp:{%.3f s}. Vehicle %d is close to the
construction zone. Slow down." % (timestamp, i+1))
                    break
        time.sleep(0.2)
```

The terminal will print out all the messages that a vehicle is close to the construction zone.



```
Timestamp:{302.759 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{302.759 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{302.959 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{302.959 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{302.959 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{303.175 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{303.175 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{303.175 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{303.375 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{303.375 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{303.375 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{303.593 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{303.593 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{303.593 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{303.814 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{303.814 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{303.814 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{304.016 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{304.016 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{304.016 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{304.239 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{304.239 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{304.239 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{304.440 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{304.440 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{304.440 s}. Vehicle 33 is close to the construction zone. Slow down.
Timestamp:{304.645 s}. Vehicle 26 is close to the construction zone. Slow down.
Timestamp:{304.645 s}. Vehicle 28 is close to the construction zone. Slow down.
Timestamp:{304.645 s}. Vehicle 33 is close to the construction zone. Slow down.
```

Task:

- Create a scenario that simulates a slow-down process when Connected and Automated Vehicles (CAVs) receive information from Roadside Units (RSUs). The scenario should showcase the impact of information received from RSUs on CAVs' behavior, such as their speed, acceleration, and lane changes.
- Record a video and submit it with the code in a ZIP file.