CEE 678 CARLA Lab 5: Visualization

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_visualize.py
   ```

Procedures to Visualize Locations in the Map:

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

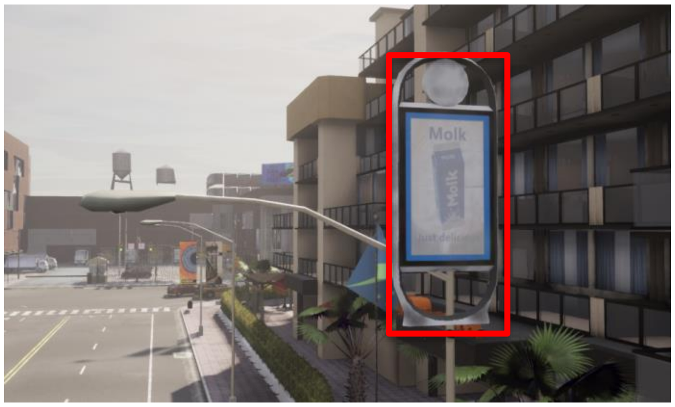
1. Multiple 0-1 toggles are used. You may turn the spawning function and visualization functions off by set 0 to the parameters or turn the functions on by setting 1 to them.
   ```
   SPAWN_NPC_VEHICLES = 1
   SPAWM_CONSTRUCTION_CONES = 1
   SPAWN_CONSTRUCTION_VEHICLES = 1
   VISUALIZE_CONSTRUCTION_ZONE = 1
   VISUALIZE_DETECTION_AREA = 1
   VISUALIZE_COMMUNICATION= 1
   ```

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

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 streetlight.



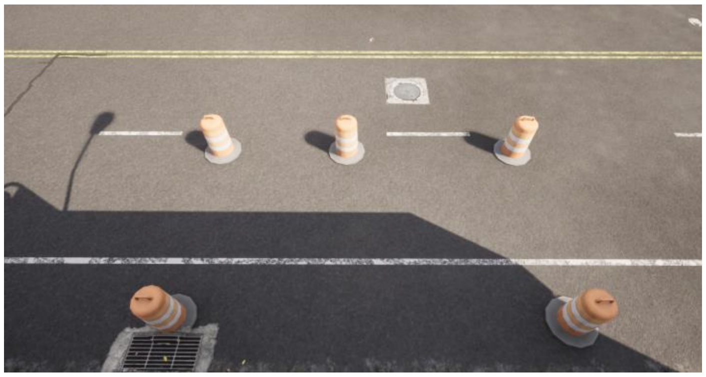
4. Set the view of spectator to the RSU.

```
spectator = world.get_spectator()
rsu_transform = rsu1.get_transform()
# View 1:
spectator.set_transform(carla.Transform(rsu_transform.location,
carla.Rotation(pitch=-35)))
# View 2:
spectator.set_transform(carla.Transform(carla.Location(x=-57,y=61,z=20),
carla.Rotation(pitch=-63)))
# View 3:
spectator.set_transform(carla.Transform(carla.Location(x=-37,y=61,z=20),
carla.Rotation(pitch=-60,yaw=180)))
```

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

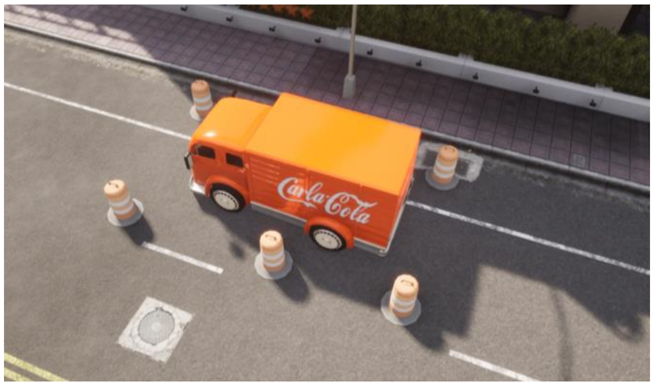
```
# Spawn construction cones
if SPAWM_CONSTRUCTION_CONES == 1:
    cone_list = []
    cone_bp = blueprint_library.find('static.prop.trafficcone01')
    spawn_point_cone1 = carla.Transform(carla.Location(x=-55, y=58, z=0))
    spawn_point_cone2 = carla.Transform(carla.Location(x=-51, y=58, z=0))
    spawn_point_cone3 = carla.Transform(carla.Location(x=-51, y=61, z=0))
    spawn_point_cone4 = carla.Transform(carla.Location(x=-51, y=65, z=0))
    spawn_point_cone5 = carla.Transform(carla.Location(x=-55, y=65, 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.



Also, to guarantee that the NPC vehicles in the autopilot mode could recognize the zone, we set a construction vehicle inside the construction zone.

```
# Spawn construction vehicles
if SPAWN_CONSTRUCTION_VEHICLES == 1:
    veh_construction_bp = blueprint_library.find('vehicle.carlamotors.carlacola')
    spawn_point_veh_con = carla.Transform(carla.Location(x=-53, y=61.2, z=1),
    carla.Rotation(yaw=90))
    veh_construction = world.spawn_actor(veh_construction_bp, spawn_point_veh_con)
    actor_list.append(veh_construction)
```

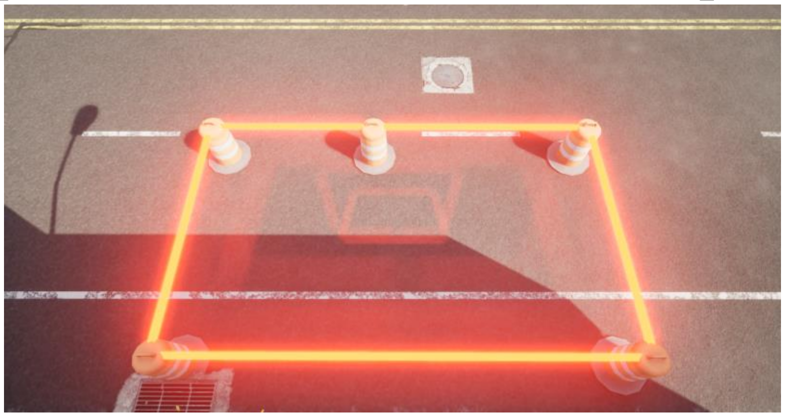


6. To avoid collision, we will spawn the NPC vehicles after the RSU and the construction cones. Iterate and generate NPC vehicle with a total number of **NPC_VEH_NUM**.

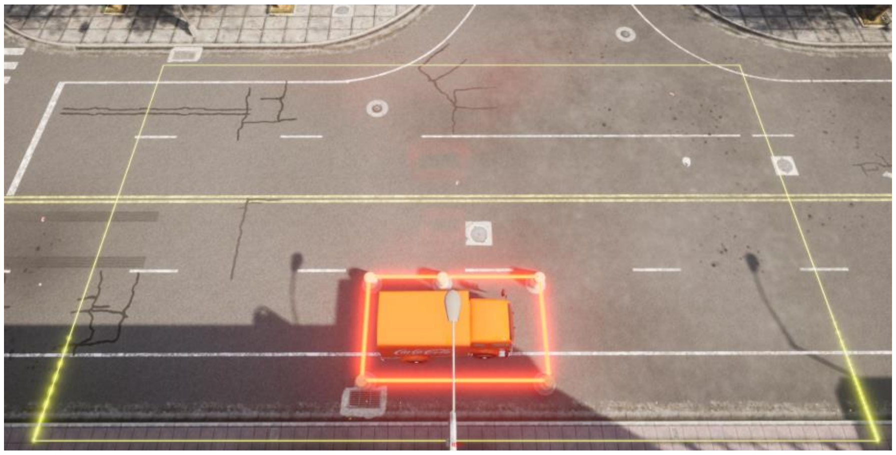
```
# Generate NPC vehicles
if SPAWN_NPC_VEHICLES == 1:
    for i in range(NPC_VEH_NUM):
        # Choose random blueprint and choose the i-th default spawn points
        vehicle_bp_i = random.choice(blueprint_library.filter('vehicle.*.*'))
        spawn_point_i = spawn_points[i]

        # 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)
```

7. Use a while loop to visualize the construction zone, detection area (service area), and the communiction and check the location of the vehicles. Before the while loop, initialize the iteration time `dt`, the maximum iteration number `max_iteration`. The while loop will break once it reaches the maximum iteration number.

```
dt = 0.2
max_iteration = 200
max_time = max_iteration * dt
iter = 0
while True:
    if iter >= max_iteration:
        break
    world_snapshot = world.get_snapshot()
    timestamp = world_snapshot.timestamp.elapsed_seconds

    if iter == 1:
        # Visulize static information:
        # Construction zone
        h = carla.Location(x=0, y=0, z=1)
        world.debug.draw_string(location=rsu1.get_location(), text='RSU',
    draw_shadow=True, color=carla.Color(255, 255, 0), life_time=max_time)
        world.debug.draw_line(begin=cone_1.get_location()+h,
    end=cone_2.get_location()+h, color=carla.Color(255, 0, 0), life_time=max_time)
        world.debug.draw_line(begin=cone_2.get_location()+h,
    end=cone_3.get_location()+h, color=carla.Color(255, 0, 0), life_time=max_time)
        world.debug.draw_line(begin=cone_3.get_location()+h,
    end=cone_4.get_location()+h, color=carla.Color(255, 0, 0), life_time=max_time)
        world.debug.draw_line(begin=cone_4.get_location()+h,
    end=cone_5.get_location()+h, color=carla.Color(255, 0, 0), life_time=max_time)
        world.debug.draw_line(begin=cone_5.get_location()+h,
    end=cone_1.get_location()+h, color=carla.Color(255, 0, 0), life_time=max_time)
```



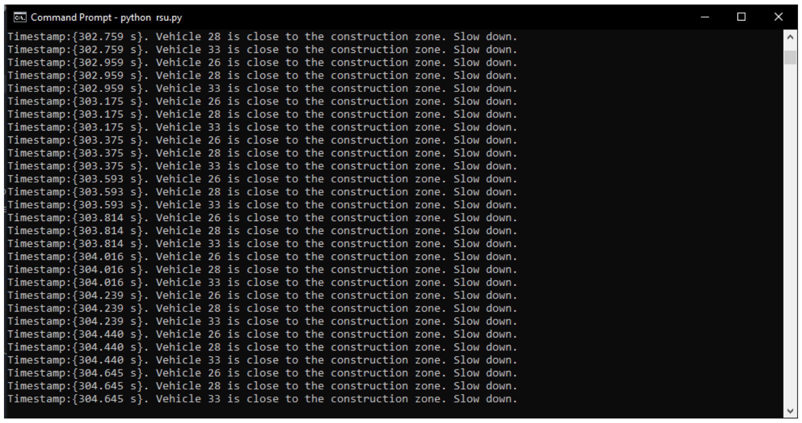
```
    # Detection area
    h_detect = 0.3
    corner_1 = carla.Location(x=rsu1.get_location().x,
y=cone_1.get_location().y) + carla.Location(x=0, y=-DETECTION_RANGE,
z=h_detect)
    corner_2 = cone_2.get_location() + carla.Location(x=DETECTION_RANGE, y=-
DETECTION_RANGE, z=h_detect)
    corner_3 = cone_4.get_location() + carla.Location(x=DETECTION_RANGE,
y=DETECTION_RANGE, z=h_detect)
    corner_4 = carla.Location(x=rsu1.get_location().x,
y=cone_5.get_location().y) + carla.Location(x=0, y=DETECTION_RANGE, z=h_detect)
    world.debug.draw_line(begin=corner_1, end=corner_2, thickness=0.03,
color=carla.Color(255, 255, 0), life_time=max_time)
    world.debug.draw_line(begin=corner_2, end=corner_3, thickness=0.03,
color=carla.Color(255, 255, 0), life_time=max_time)
    world.debug.draw_line(begin=corner_3, end=corner_4, thickness=0.03,
color=carla.Color(255, 255, 0), life_time=max_time)
    world.debug.draw_line(begin=corner_4, end=corner_1, thickness=0.03,
color=carla.Color(255, 255, 0), life_time=max_time)
```



```
# 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 distance between each vehicle and each cone
        dist_x = np.abs(v_x - c_x)
        dist_y = np.abs(v_y - c_y)

        # If the vehicle is within the detection area, assume RSU could send this
        message to the vehicle.
```

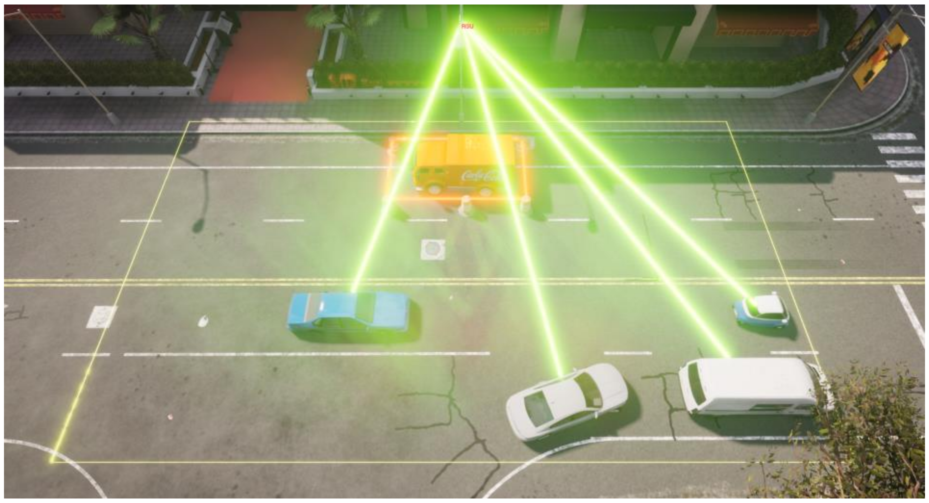
```
            if dist < detection_range:
                print("Timestamp:{%.3f s}. Vehicle %d is close to the construction
        zone. Slow down." % (timestamp, i+1))
                # Visualize the communication process
                world.debug.draw_line(begin=rsu1.get_location(),
                end=vehicle.get_location(), color=carla.Color(64, 255, 0),
                life_time=dt)
                break
    iter += 1
    time.sleep(dt)
```

The RSU will send messages to the vehicles in the detection area.



```
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.
```

The transmission of the data can be represented by the green line connecting the RSU and the vehicles.



Appendix:

For more details of drawing text, lines, points, arrows, and boxes in the world, please refer to:

- https://carla.readthedocs.io/en/latest/python_api/#carla.DebugHelper
- https://carla.readthedocs.io/en/latest/tuto_G_getting_started/#using-and-visualizing-map-spawn-points