EE267 – Lab 0: Access CARLA

Name: *Rishi Patel(rpate248) and Darshil Shukla(dshuk003)*

Environment Used: Option 2 – BCOE GPU Machines

1. Objective

The goal of this lab is to set up and verify the CARLA simulator environment on the BCOE GPU machines.
By the end of the lab, I should:

- Be able to connect to the CARLA environment through TurboVNC
- Run the CARLA server and client successfully
- Understand the CARLA API basics (traffic manager, sensor data)
- Implement visual perception components like bounding boxes and instance segmentation

2. Environment Setup (BCOE GPU Server)

2.1 System Information

- **Access method:** TurboVNC
- **Conda environment name:** carla
- **CARLA version:** 0.9.15

3. Part 1 – Running CARLA Server

Procedure:

- Started the CARLA server using:

```
bash CarlaUE4.sh --RenderOffScreen
```

- Confirmed CARLA server loaded the default map (Town01) and displayed logs such as:

```
LogCarlaServer: Waiting for the client to connect...
```

4. Part 2 – Drive a Car Using Carla Client

4.1 Manual Control

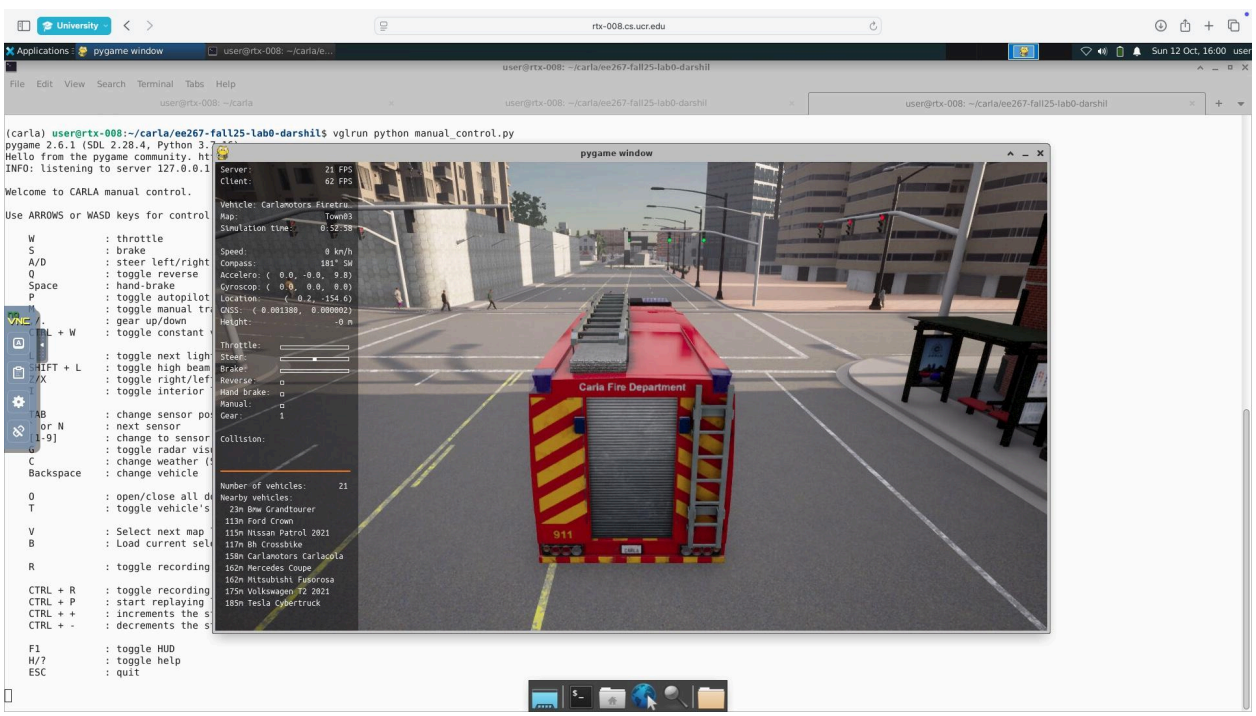
- Activated the `carla` environment:

```
conda activate carla
```

- Launched manual control client:

```
vglrun python manual_control.py
```

- Controlled vehicle movement using keyboard inputs (W, A, S, D).

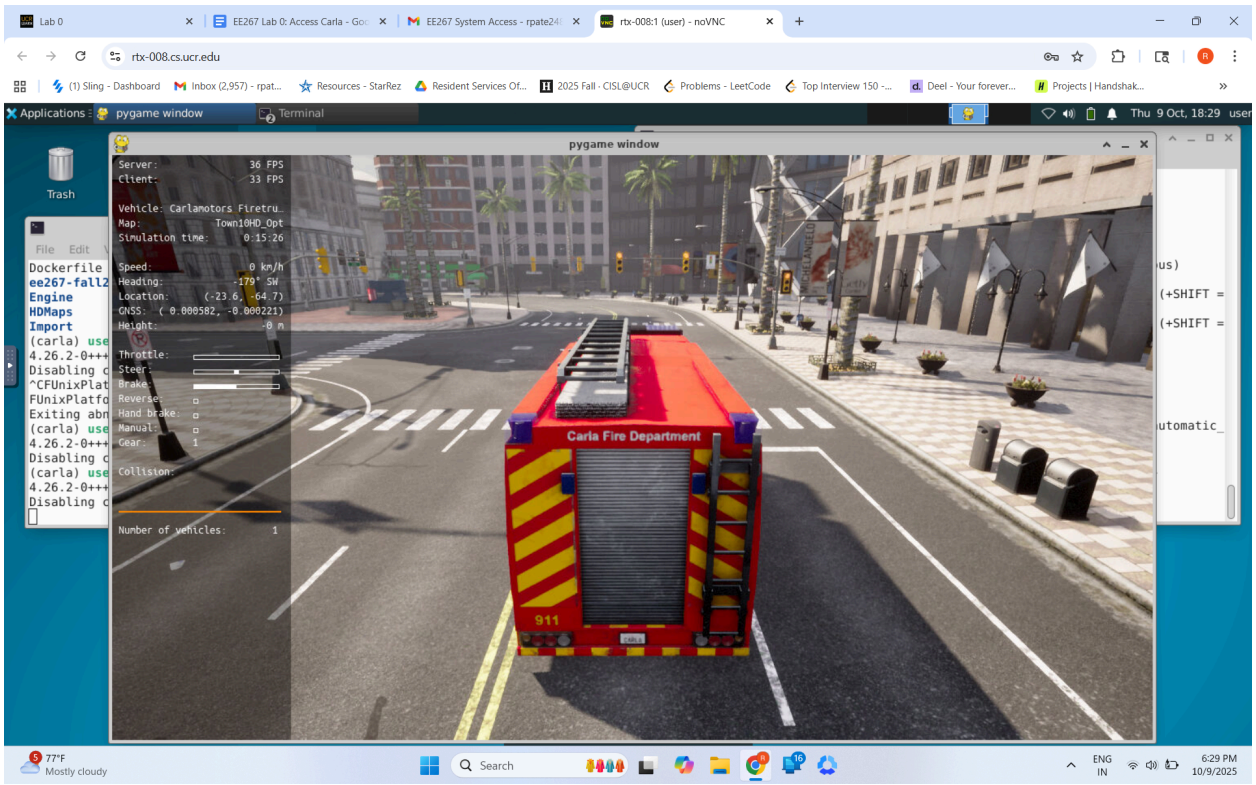


4.2 Automatic Control

- Ran the automatic control client:

```
vglrun python automatic_control.py
```

- The vehicle automatically followed a navigation route.



Observations:

- Pygame window responsive via VNC.
- Autopilot path planning performed smoothly.

5. Part 3 – Using Carla API (Traffic Manager)

Implementation Steps:

- Connected to CARLA client:

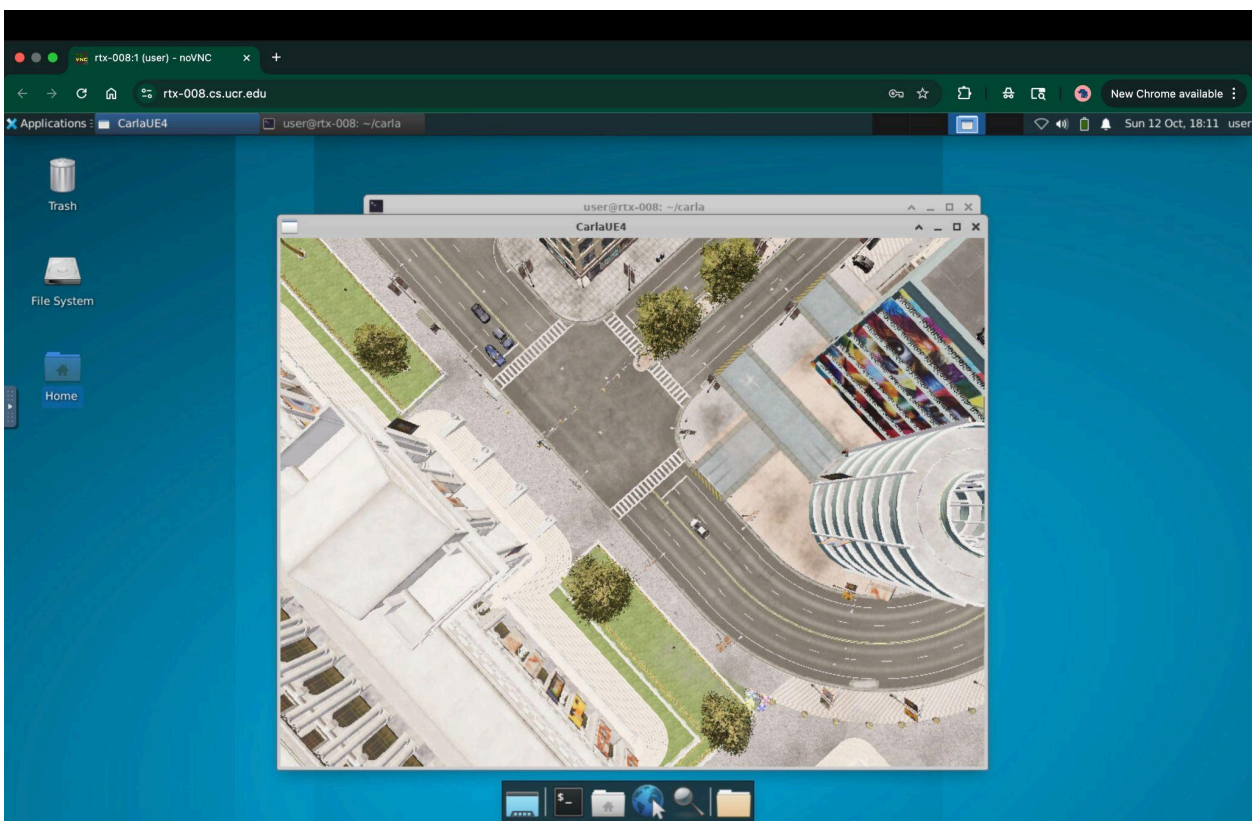
```
client = carla.Client('localhost', 2000)

world = client.get_world()

tm = client.get_trafficmanager()
```

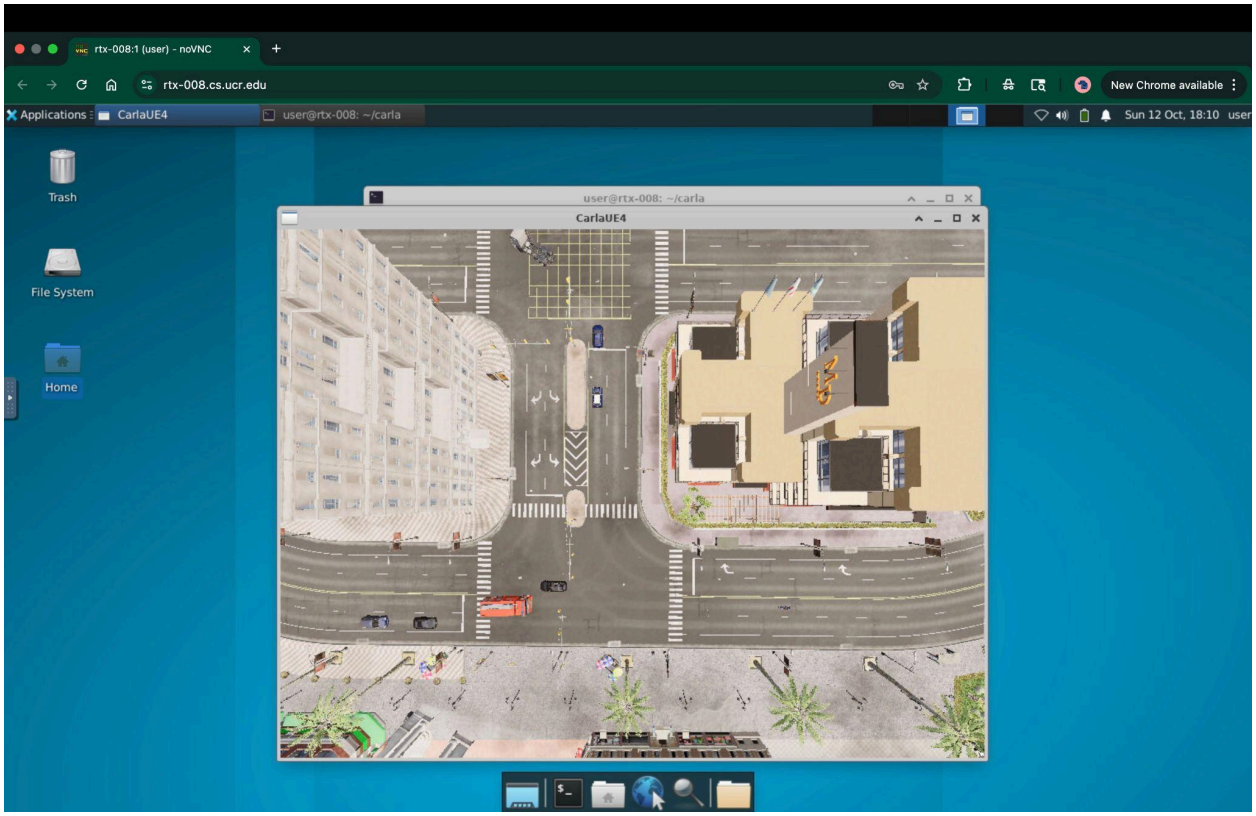
1. Spawned random vehicles and pedestrians:

```
blueprints = world.get_blueprint_library().filter('vehicle.*')
```

2. Enabled autopilot for background vehicles:

```
vehicle.set_autopilot(True, tm.get_port())
```





Observations:

- The traffic manager ensures collision avoidance and lane discipline.
- Useful for generating training and testing data with realistic traffic flow.

6. Part 4 – Bounding Boxes

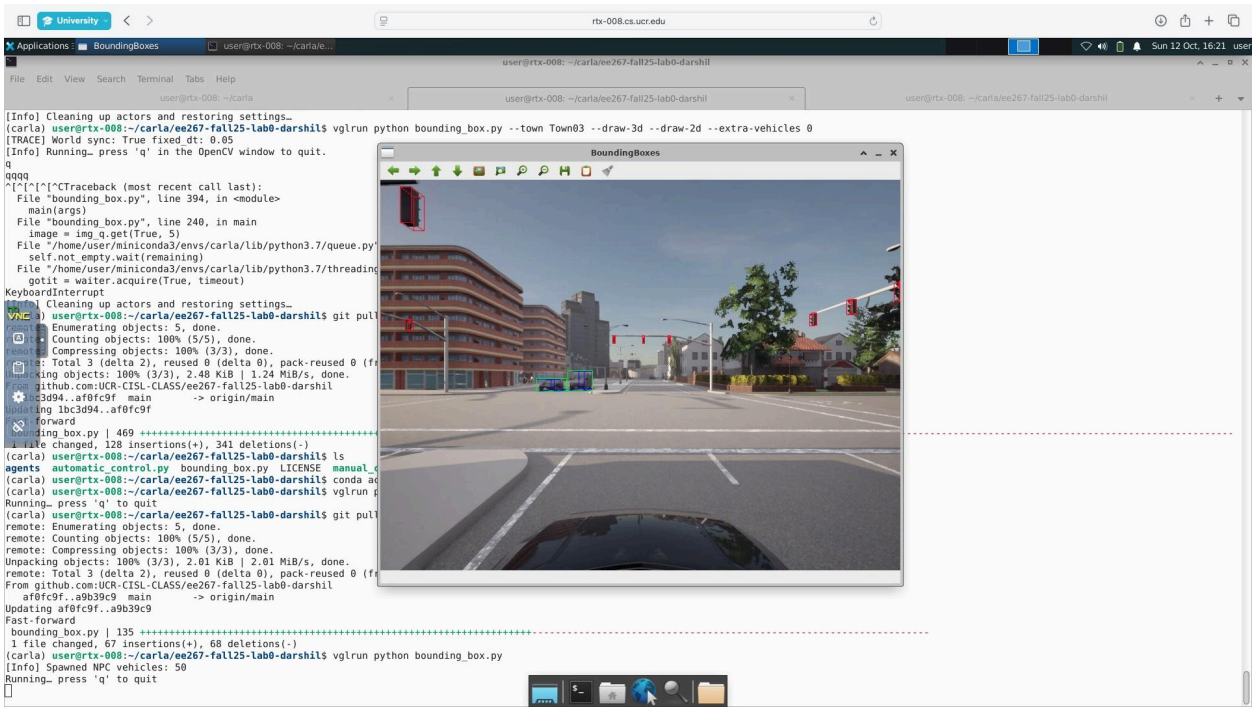
Implementation Details:

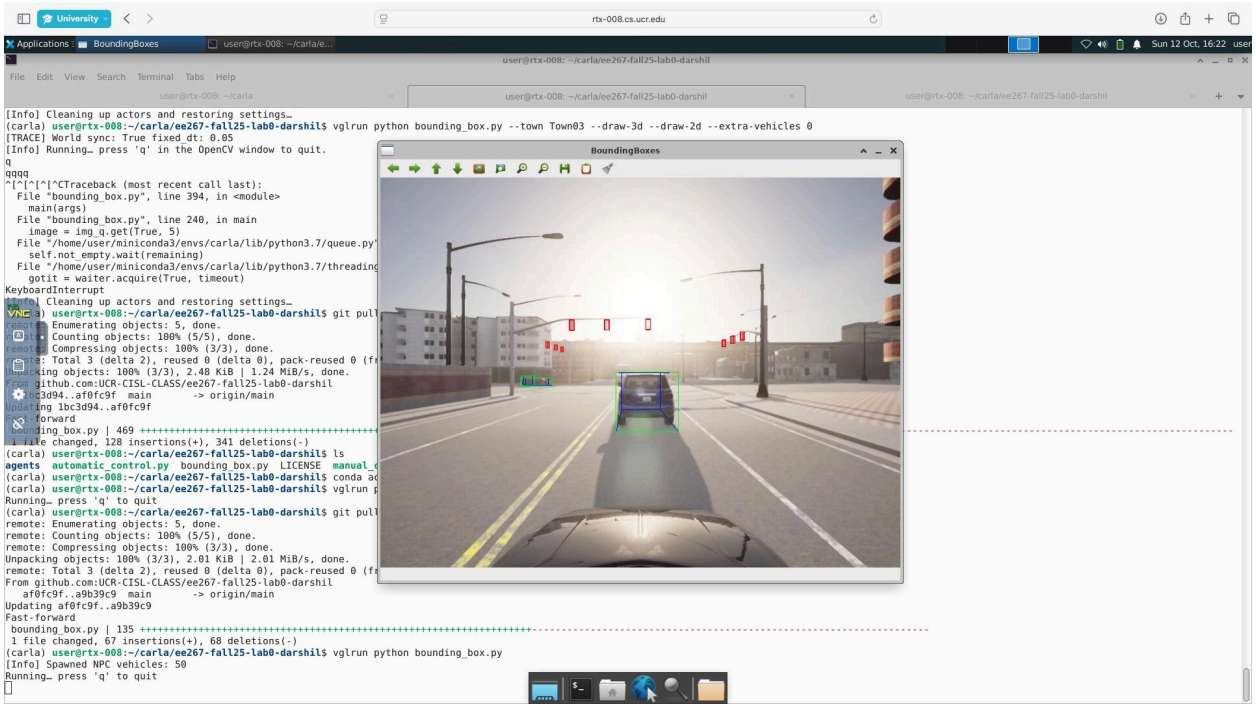
- Implemented geometric transformations:
    - 3D → 2D projection using the camera's intrinsic matrix.
    - Used CARLA-provided vehicle bounding box data.

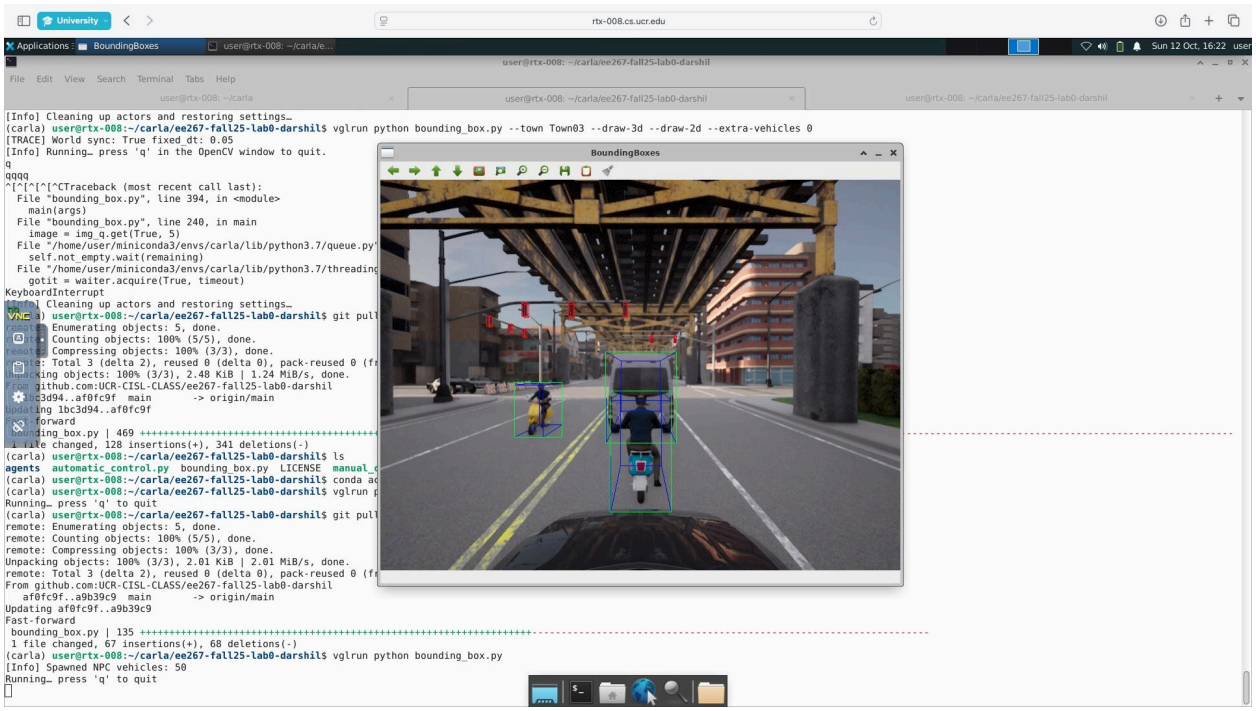
- Implemented helper functions:

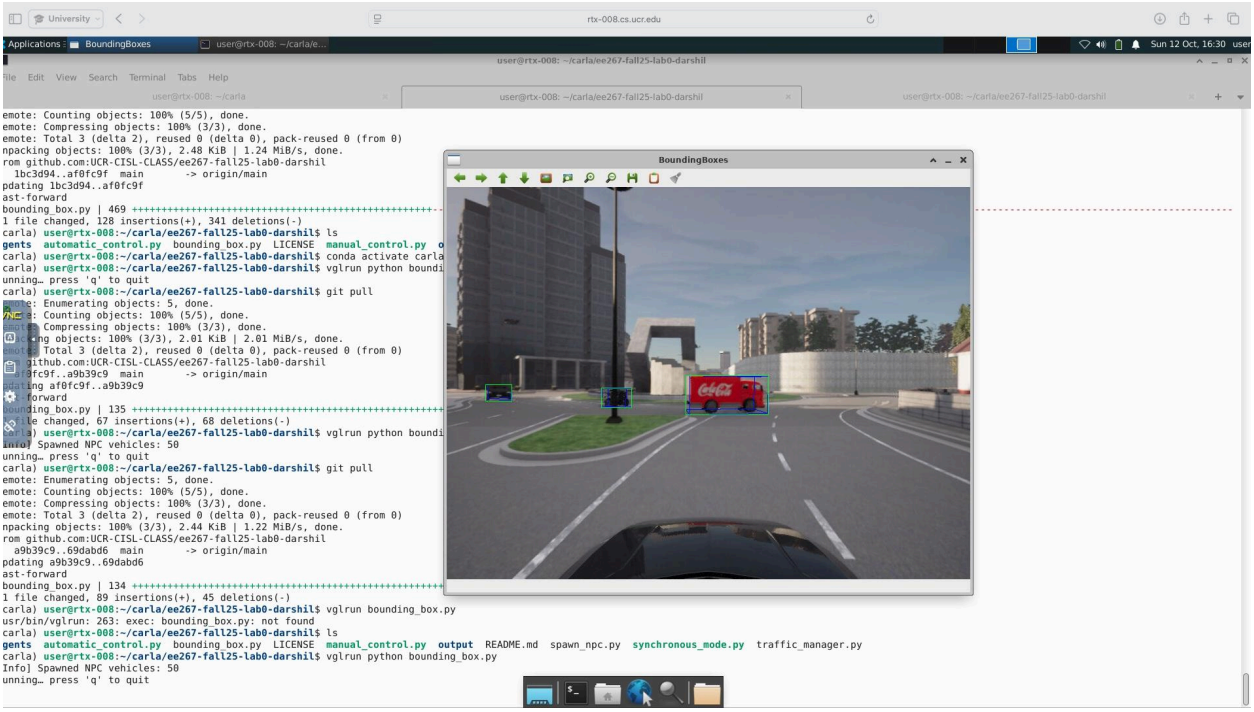
- `build_projection_matrix()`
- `get_image_point()`

- Drew bounding boxes around detected vehicles on RGB images.
- Exported bounding boxes in **Pascal VOC format**.




rtx-008.cs.ucr.edu
Applications : BoundingBoxes
Sun 12 Oct, 16:22
user@rtx-008: ~/carla/ee267-fall25-lab0-darshil
File Edit View Search Terminal Tabs Help
[Info] Cleaning up actors and restoring settings…
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ vglrun python bounding_box.py --town Town03 --draw-3d --draw-2d --extra-vehicles 0
[TRACE] World sync: True fixed_dt: 0.05
[Info] Running… press 'q' in the OpenCV window to quit.
q
qqqq
^[^[^[^[^CTraceback (most recent call last):
File "bounding_box.py", line 394, in <module>
main(args)
File "bounding_box.py", line 240, in main
image = img_q.get(True, 5)
File "/home/user/miniconda3/envs/carla/lib/python3.7/queue.py"
self.not_empty.wait(remaining)
File "/home/user/miniconda3/envs/carla/lib/python3.7/threading
gotit = waiter.acquire(True, timeout)
KeyboardInterrupt
[Info] Cleaning up actors and restoring settings…
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
Unpacking objects: 100% (3/3), 2.48 KiB | 1.24 MiB/s, done.
From github.com:UCR-CISL-CLASS/ee267-fall25-lab0-darshil
Updating 1bc3d94..af0fc9f
Fast-forward
bounding_box.py | 469
1 file changed, 128 insertions(+), 341 deletions(-)
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ ls
agents automatic_control.py bounding_box.py LICENSE manual_
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ conda ac
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ vglrun p
Running… press 'q' to quit
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
Unpacking objects: 100% (3/3), 2.01 KiB | 2.01 MiB/s, done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (fr
From github.com:UCR-CISL-CLASS/ee267-fall25-lab0-darshil
af0fc9f..a9b39c9 main -> origin/main
Updating af0fc9f..a9b39c9
Fast-forward
bounding_box.py | 135
1 file changed, 67 insertions(+), 68 deletions(-)
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ vglrun python bounding_box.py
[Info] Spawned NPC vehicles: 50
Running… press 'q' to quit
BoundingBoxes


rtx-008.cs.ucr.edu
Applications : BoundingBoxes
Sun 12 Oct, 16:22
user@rtx-008: ~/carla/ee267-fall25-lab0-darshil
File Edit View Search Terminal Tabs Help
[Info] Cleaning up actors and restoring settings…
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ vglrun python bounding_box.py --town Town03 --draw-3d --draw-2d --extra-vehicles 0
[TRACE] World sync: True fixed_dt: 0.05
[Info] Running… press 'q' in the OpenCV window to quit.
q
qqqq
^[^[^[^[^CTraceback (most recent call last):
File "bounding_box.py", line 394, in <module>
main(args)
File "bounding_box.py", line 240, in main
image = img_q.get(True, 5)
File "/home/user/miniconda3/envs/carla/lib/python3.7/queue.py"
self.not_empty.wait(remaining)
File "/home/user/miniconda3/envs/carla/lib/python3.7/threading
gotit = waiter.acquire(True, timeout)
KeyboardInterrupt
[Info] Cleaning up actors and restoring settings…
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
Unpacking objects: 100% (3/3), 2.48 KiB | 1.24 MiB/s, done.
From github.com:UCR-CISL-CLASS/ee267-fall25-lab0-darshil
Updating 1bc3d94..af0fc9f
Fast-forward
bounding_box.py | 469
1 file changed, 128 insertions(+), 341 deletions(-)
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ ls
agents automatic_control.py bounding_box.py LICENSE manual_
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ conda ac
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ vglrun p
Running… press 'q' to quit
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
Unpacking objects: 100% (3/3), 2.01 KiB | 2.01 MiB/s, done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (fr
From github.com:UCR-CISL-CLASS/ee267-fall25-lab0-darshil
af0fc9f..a9b39c9 main -> origin/main
Updating af0fc9f..a9b39c9
Fast-forward
bounding_box.py | 135
1 file changed, 67 insertions(+), 68 deletions(-)
(carla) user@rtx-008:~/carla/ee267-fall25-lab0-darshil$ vglrun python bounding_box.py
[Info] Spawned NPC vehicles: 50
Running… press 'q' to quit
BoundingBoxes



Observations:

- Bounding box accuracy depends on projection matrix calibration.
- Important for vision-based perception and dataset creation.

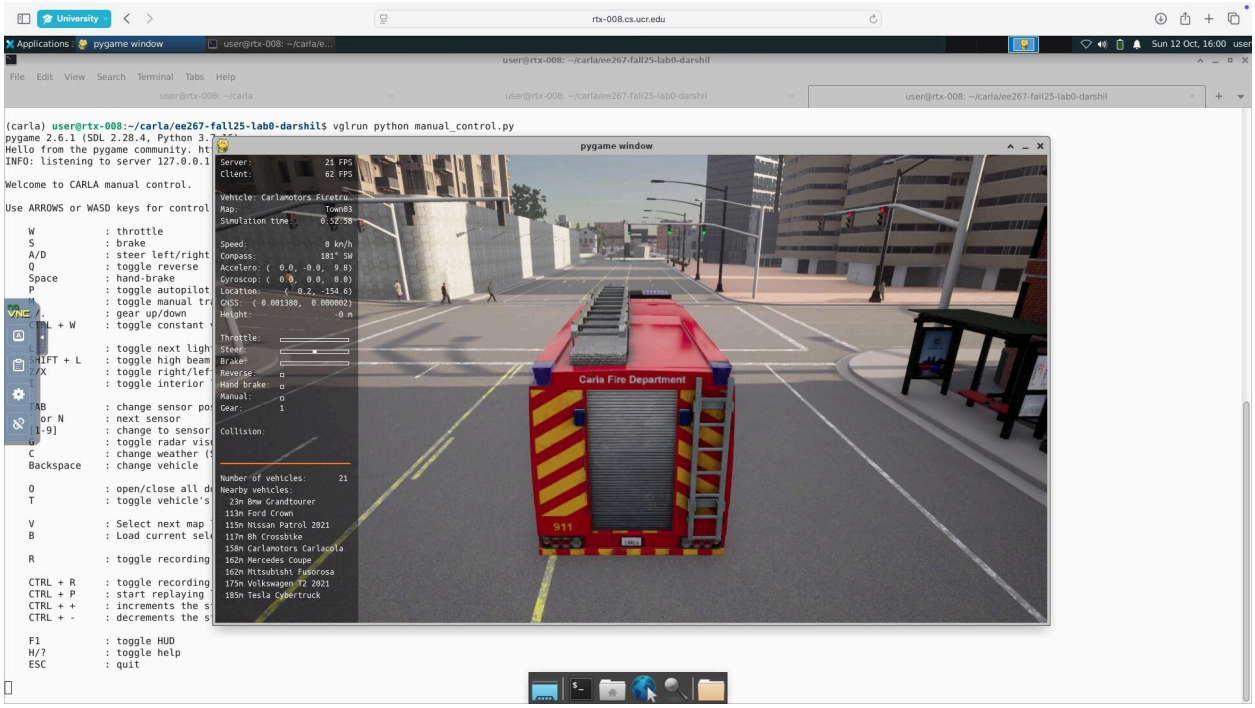
7. Part 5 – Instance Segmentation

Steps:

- Created instance segmentation camera blueprint:

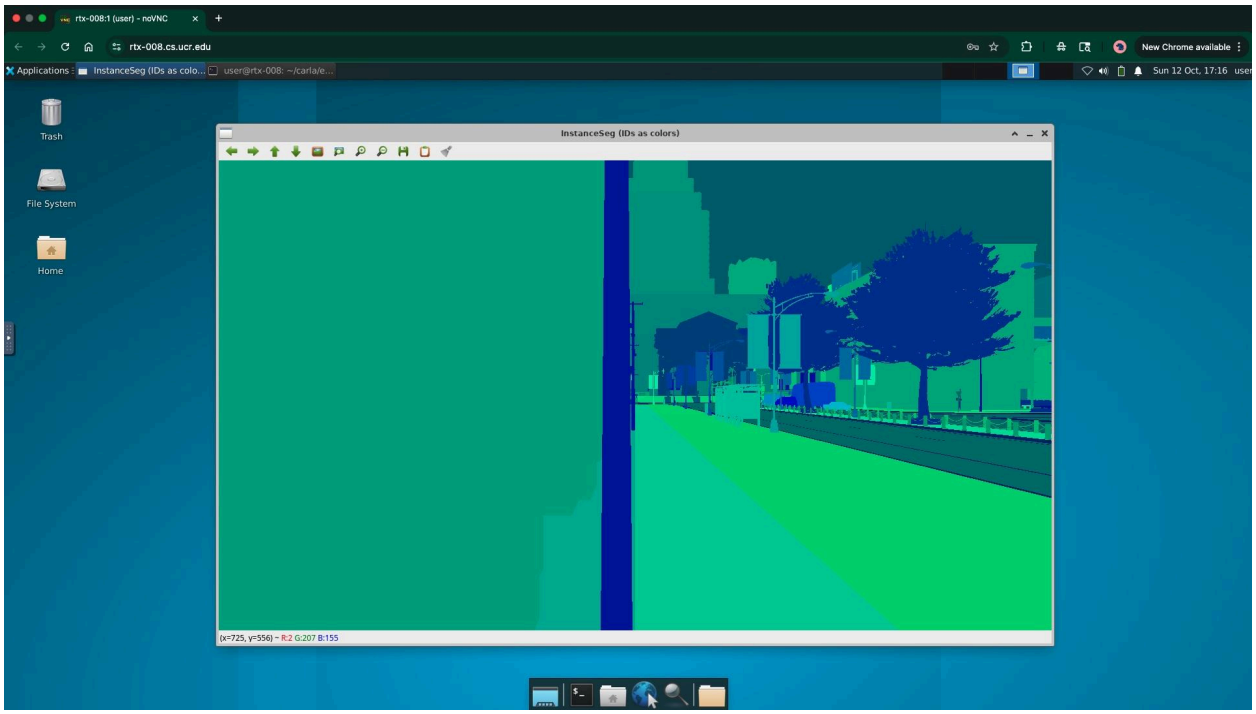
```
camera_bp = blueprint_library.find('sensor.camera.instance_segmentation')
```

1. Set up camera location and rotation near the ego vehicle.
2. Spawned multiple vehicles around the camera (radius = 80 m).



- Captured and saved the instance segmentation image:

  `instance_segmentation.png`



Analysis:

- Each object instance is represented by a unique ID encoded as a distinct RGB color.

- **Difference from Semantic Segmentation:**
    - Instance segmentation differs from semantic segmentation in that it not only classifies each pixel into a category but also distinguishes between individual objects within the same class. While semantic segmentation labels all pixels belonging to a category (e.g., all cars as “car”), instance segmentation assigns unique identifiers to each object instance, allowing the system to recognize and separate multiple cars, pedestrians, or other objects individually. In autonomous driving, this distinction is critical as it enables the vehicle to detect, track, and predict the movements of distinct road users, facilitating precise decision-making for tasks such as obstacle avoidance, lane changes, and safe navigation in complex traffic environments.

Applications:

- Object tracking

- Instance-aware obstacle detection

- Scene understanding for autonomous driving

8. Reflection

Challenges Faced:

- Occasional lag in TurboVNC rendering.

- Some vehicles spawned outside the field of view due to random spawn points.
- Random people were not spawned due to collision with an already present object at that place, so we retry.

Key Takeaways:

- Learned how to control the CARLA server-client interaction.

- Gained experience with CARLA's Traffic Manager and sensor APIs.

- Understood bounding box geometry and perception visualization.

- Differentiated between instance and semantic segmentation.

9. References

- CARLA Documentation: https://carla.readthedocs.io

- UCR CISL Carla Docker Repo: https://github.com/UCR-CISL/Carla-dockers

- EE267 Canvas Lab 0 Slides and Tutorials

10. Acknowledgements

We would like to thank our Professor and TA(s) for their hard work in designing, assigning as well as grading this Lab and providing their valued feedback.