EE585 Mobile Robotics and Autonomous Navigation

[CARLA Introduction]

URBAN ROBOTICS LAB

Provided by Jinwoo Jeon and Minho Oh
Advised by Prof. Hyun Myung

Oct. 2023

KAIST

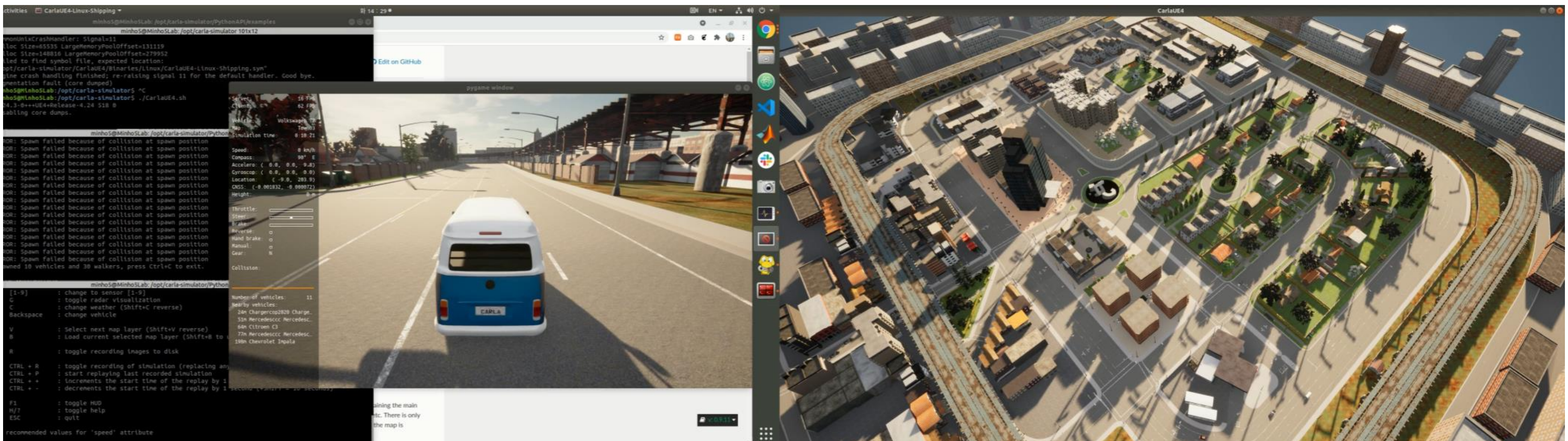
URBAN ROBOTICS LAB

Contents

- CARLA and CALRA-ROS Bridge
- Install Guideline
- Final Term Project

What is CARLA

- CARLA is an open-source autonomous driving simulator based on Unreal Engine and OpenDRIVE.
- CALRA Configurations
  - World & Clients
  - Actors & Blueprints
  - Maps & Navigations
  - Sensors and its data

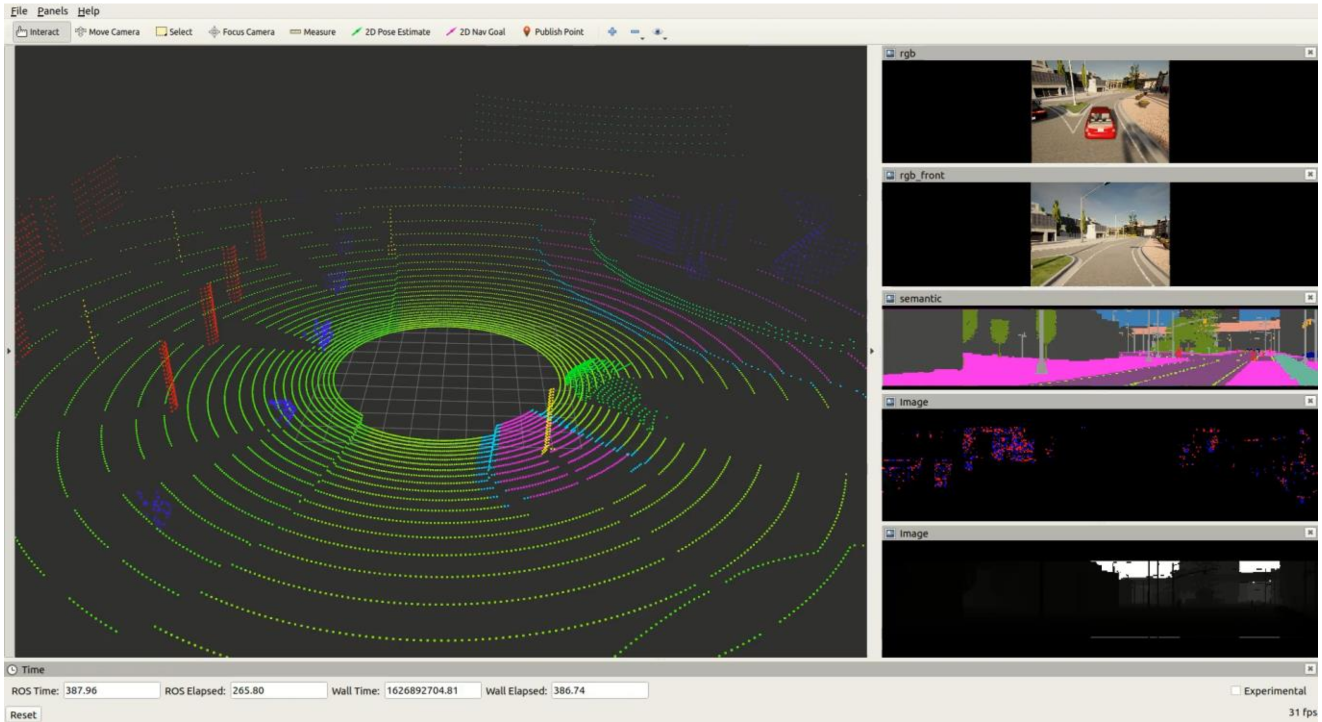


Configuration (1)

- World and Clients are two of the fundamentals of CARLA
  - World: an object representing the simulation environment
  - Clients: a module that the users run to request info. or make changes in the simulation
- Actors and Blueprints
  - Actors: anything that plays a role in the simulation
    - Vehicles, pedestrians, sensors, spectator, traffic signs, and lights
  - Blueprints: specific configurations of Actors.
- Maps and Navigation
  - Maps: the object representing the simulated world
    - Roads, lanes, and junctions are associated with the waypoint class
    - Traffic signs and lights are accessible as carla.Landmark
  - Navigation: managed via the waypoint API

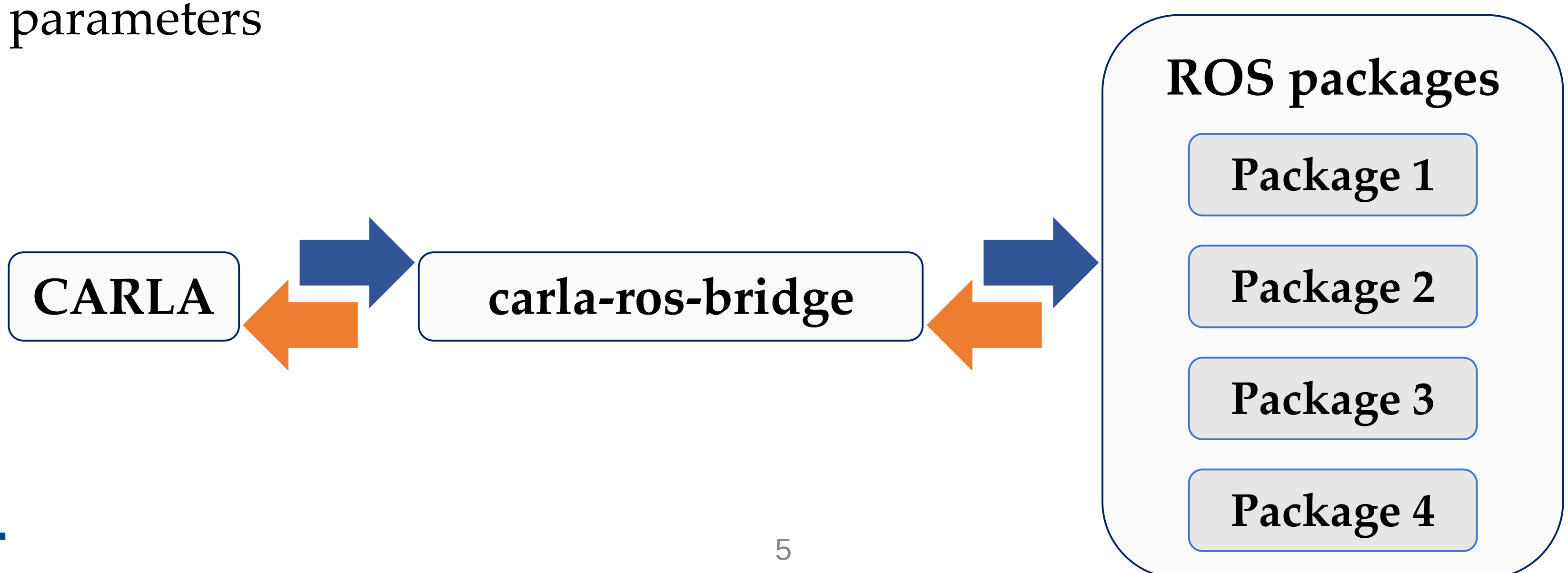
Configuration (2)

- Available sensors and its data
  - Camera with RGB, semantic labels, event informantion, and depth
  - GNSS (GPS) sensor
  - IMU sensor
  - LiDAR and semantic LiDAR raycast
  - Radar
  - Collision and lane invasion detectors
  - …



CARLA-ROS Bridge

- CARLA-ROS bridge
  - To obtain sensor data via ROS topics
    - LiDAR, semantic lidar, cameras, GNSS, Radar, IMU
  - To obtain object data via ROS topics
    - Transforms, traffic light status, collision, lane invasion
  - To control autonomous agents via ROS topics or RVIZ
    - Steer, throttle, brake
  - To control CARLA simulation by playing and pausing, and set simulation parameters



Install Guidelines

TAs Notion page

: https://minho5oh.notion.site/EE585-Mobile-Robotics-and-Autonomous-Navigation-Guideline-for-CARLA-simulation-2c7feebafc81482cbb30f4b42080fe37

CARLA documents

: https://minho5oh.notion.site/EE585-Mobile-Robotics-and-Autonomous-Navigation-Guideline-for-CARLA-simulation-2c7feebafc81482cbb30f4b42080fe37

CARLA ROS documents

: https://carla.readthedocs.io/projects/ros-bridge/en/latest/

Requirements

- Requirements
  - Ubuntu 18.04
    - These guidelines are based on Ubuntu 18.04 and ROS 1 (ver. melodic)
  - NVIDIA Graphics Driver
  - Docker
  - Nvidia-docker 2

Install docker

- Docker

```
# Uninstall old version if you need
$ sudo apt-get remove docker docker-engine docker.io containerd runc

# Set up the repository
# Update the apt package index and install packages to allow apt to use a repository over HTTPS:
$ sudo apt-get update
$ sudo apt-get install \
        apt-transport-https \
        ca-certificates \
        curl \
        gnupg \
        lsb-release

# Add Docker's official GPG key:
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
# Use the following command to set up the stable repository.
$ echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker Engine
# 1. Update the apt package index, and install the latest version of Docker Engine and containerd:
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io

# 2. Verify that Docker Engine is installed correctly by running the hello-world image.
$ sudo docker run hello-world
```

Install Nvidia-docker2

- Nvidia-docker 2

```
# Set Repository
$ curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add -
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
$ curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list
$ sudo apt-get update

# Set Repository Key
$ curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add -

# Uninstall the nvidia-docker 1.0
$ sudo docker volume ls -q -f driver=nvidia-docker | xargs -r -I{} -n1 docker ps -q -a -f volume={} | xargs -r docker rm -f
$ sudo apt-get purge nvidia-docker

# Install nvidia-docker 2.0
$ sudo apt-get install nvidia-docker2
$ sudo pkill -SIGHUP dockerd
```

KAIST

URL
URBAN ROBOTICS LAB

Set CARLA in docker (1)

- Preparation
  - Get the docker file for carla-ros-bridge using the following command.

```
$ git clone https://gitlab.com/Minho5/carla_ros_bridge
```

  - Download the "CARLA_0.9.10.1.tar.gz" and "AdditionalMaps_0.9.10.1.tar.gz" from https://github.com/carla-simulator/carla/releases/tag/0.9.10.1.
  - Put the CARLA simulator in the same directory as the downloaded Dockerfile as following example:

```
~/carla_ros_bridge
|- AdditionalMaps_0.9.10.1.tar.gz
|- CARLA_0.9.10.1.tar.gz
|- Dockerfile.melodic
|- launch_container.sh
|- LICENSE
|- README.md
```

Set CARLA in docker (2)

- Preparation
  - Build the Docker image by using the following command:

    ```
    $ sudo docker build -t carla:0.9.10 -f Dockerfile.melodic .
    ```

  - Create a docker container and launch it with the following command:

    ```
    $ sudo ./launch_container.sh
    ```

  - Once you've executed the above command, you should see the following output in your terminal:
    *** If you get the error message such as "time out(?)" like following example, just ignore it.

```
euigon@euigon-jung:~/v0.9.10$ sudo ./launch_conta
iner.sh
xauth:  timeout in locking authority file /tmp/.d
ocker.xauth
carla_melodic@euigon-jung:~$ ls
CARLA_0.9.10.1  check_container.sh
catkin_ws       scenario_runner
carla_melodic@euigon-jung:~$
```

Set ROS package for term project

- Preparation
  - Remove "ros-bridge" folder in ~/catkin_ws/src and git clone a repository

```
$ cd ~/catkin_ws/src
$ rm -rf ros-bridge
$ git clone https://gitlab.com/Minho5/ee585_carla_project.git
```

  - Make

```
$ cd ~/catkin_ws
$ catkin_make
```

Run Demo. (1)

- Run CARLA

```
# Terminal 1.

$ cd CARLA_{$VERSION}
$./CarlaUE4.sh
```

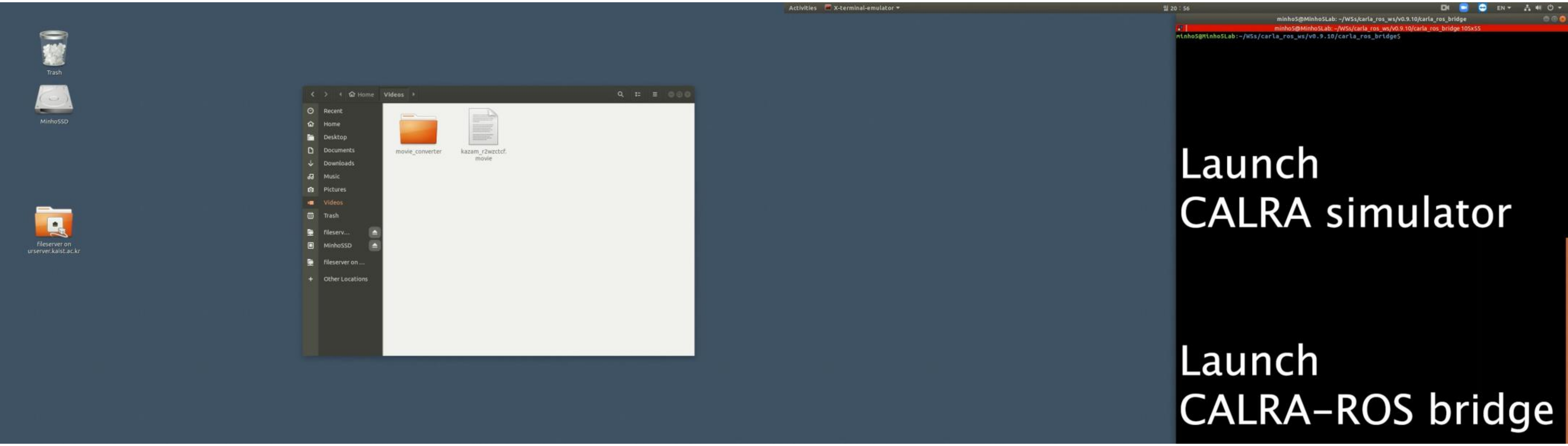
- Run CARLA-ROS-Bridge Demo

```
# Terminal 2.

# Run bridge with a random vehicle with manual control
# = carla_ros_bridge
# + carla_example_ego_vehicle.launch
# + carla carla_manual_control.launch
$ roslaunch carla_ros_bridge carla_ros_bridge_with_example_ego_vehicle.launch
```

Run Demo. (2)

- Launch CARLA simulator
- Launch CARLA-ROS bridge
- Launch Rviz for sensor data visualization



KAIST

URL
URBAN ROBOTICS LAB

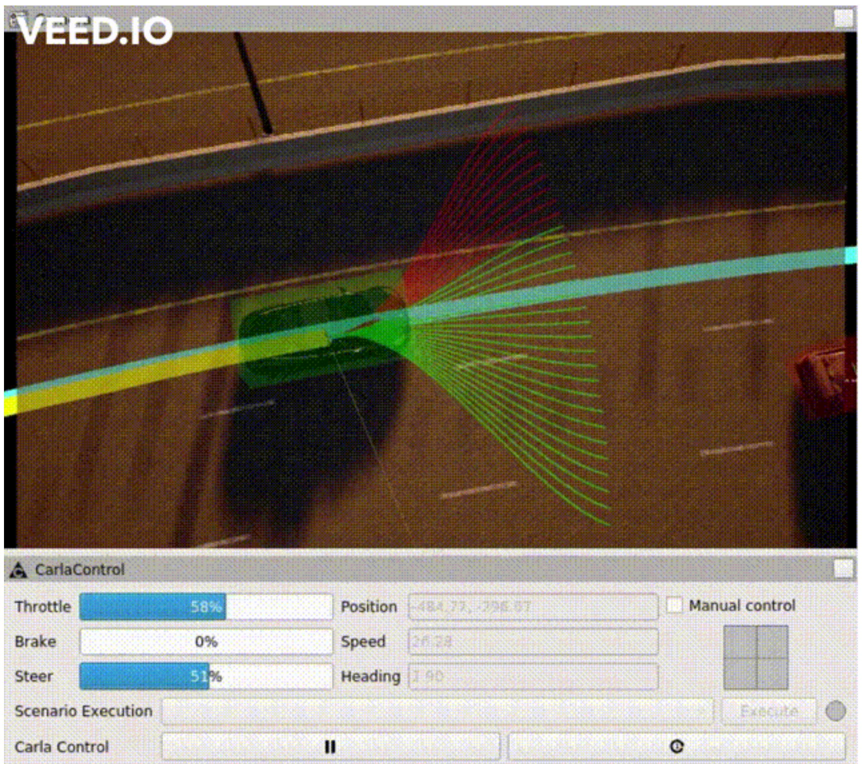
Final term project

Task 1. Path planning

Task 2. LiDAR / Visual SLAM

Task 1: Path planning

- Implement a local planning and reach the goal point through a given scenario.
  - Modify "{*$Workspace*}/src/ros-bridge/carla_ad_agent/src/**my_local_planner.py**"
  - Mission is to follow the start-to-goal global path



Task 1: Path planning

- Evaluation
  - Evaluate each person for each criterion.
  - Details for the scenario, score policy like "*Extra credit*", are described in the link.

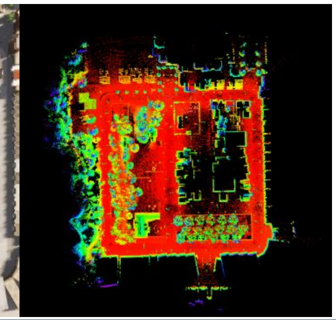
Team 1:

```
------Simulation Result------ [redacted]
Time: 63.6236450672 s
Distance travelled: 198.593741555 m
Mission progress: 31.2380952381%
Mission completed: False
# Collision per m: 0.0100708108138
# Invasion per m: 0.0704956756966
------------------------------
```

Team	Progress [%]	Goal reached	Run time [sec]	Collision per meter	Invasion per meter	Extra credit
1	31.238	False	63.624	0.010	0.070	0
2						
3						

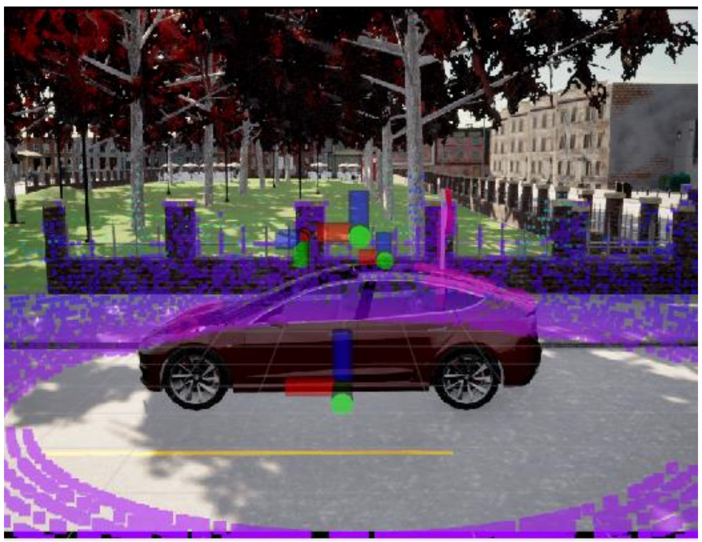
Task 2: SLAM on CARLA

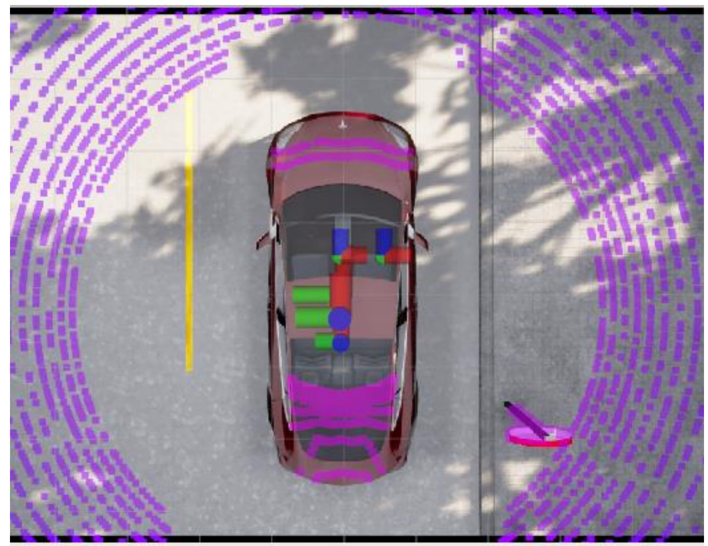
- Run open-source state-of-the-art SLAM on the given CARLA dataset.
  - LiDAR / Visual SLAM
- Improve the performance by modifying the existing module or adding your own SLAM.



Task 2: SLAM on CARLA

- Test platform
  - Sensor configuration: Two cameras, one LiDAR, and one IMU.

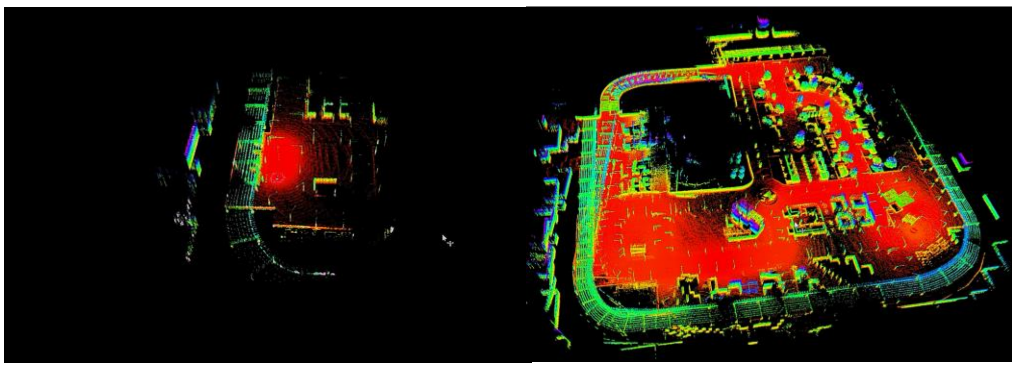




- Evaluation
  - State-of-the-art SLAM algorithm demonstration [100%]
  - Creativity + Performance improvement [extra 20%]

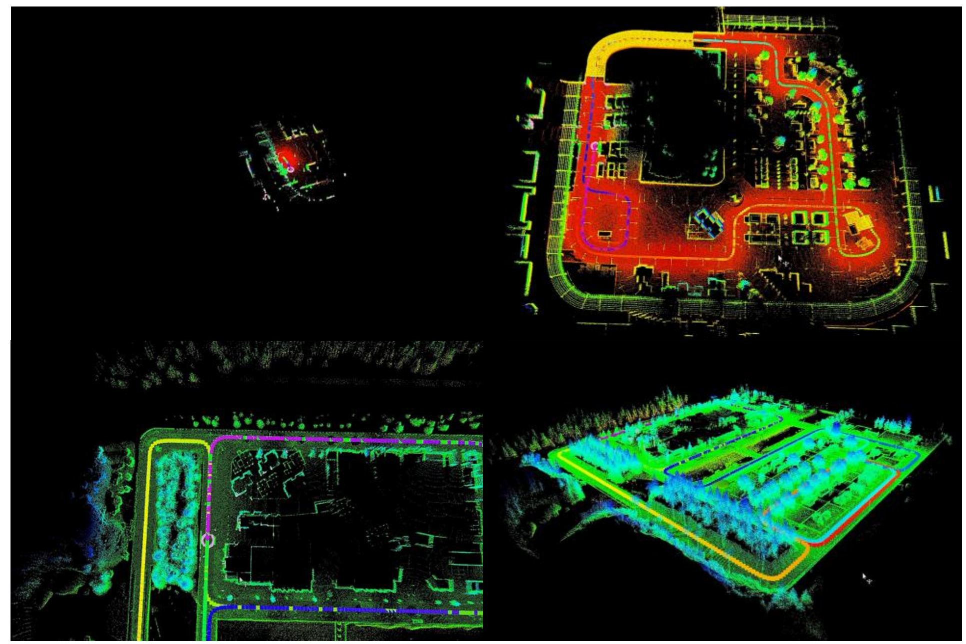
Task 2: Example Demo: LiDAR SLAM

- Run the original LiDAR SLAM
  - Baseline: LeGO-LOAM[1]



LeGO-LOAM result with poor loop closure result

- Improve LiDAR SLAM
  - Contribution:
    Improve the loop closure detection module and add the loop closure constraints in pose graph optimization to prevent the divergence.

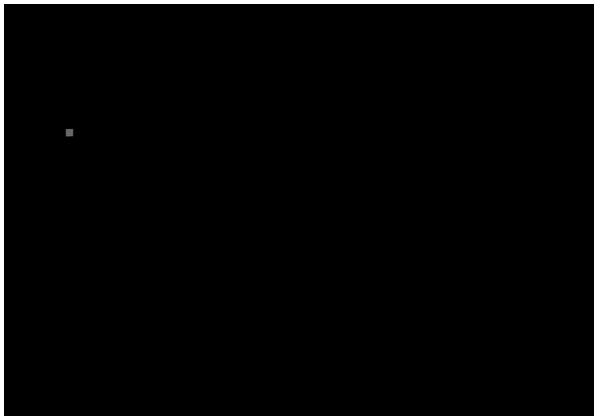


Student's improved SLAM result

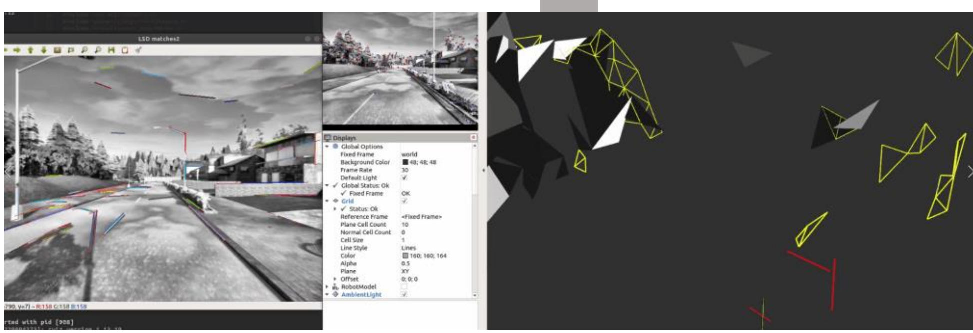
[1] Tixiao Shan, and Brendan Englot, „LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain", in *Proc. IEEE/RSJ IROS* 2018.

Task 2: Example Demo: Visual SLAM

- Run the original visual SLAM
  - Baseline: PL-VIO[2]



PL-VIO result



Constrained Delaunay triangulation (CDT)-based mesh generation

- Improve visual SLAM
  - Contribution:
    To make more dense visual map, a student employed ″mesh generatcion technique″; built from point and line features by using constrained Delaunay triangulation (CDT).

Student's improved SLAM result

[2] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, "PL-VIO: Tightly-coupled monocular visual–inertial odometry using point and line features," *Sensors*, vol. 18, no. 4, p. 1159, 2018.

Open source SLAM algorithms

- The following open-source algorithms are available on GitHub, and you are encouraged to utilize and develop these algorithms for your Term Project Task 2.

- LiDAR-based algorithms
  - LeGO-LOAM
    https://github.com/RobustFieldAutonomyLab/LeGO-LOAM
  - LIO-SAM
    https://github.com/TixiaoShan/LIO-SAM
  - Fast-LIO
    https://github.com/hku-mars/FAST_LIO
  - …

- Vision-based algorithms
  - VINS-Fusion
    https://github.com/HKUST-Aerial-Robotics/VINS-Fusion
  - ORB-SLAM
    https://github.com/UZ-SLAMLab/ORB_SLAM3
  - …

Submission

- Your file should include:
  - "ros-bridge" file inside "ee585_carla_project" (Task 1)
  - Two bag files (Task 2)
    - "origin_slam.bag": include the odometry result of original code and ground truth
    - "improved_slam.bag": include odometry result of your own improved SLAM and ground truth (only if you want to get an extra credit)
  - Readme.txt
    - Brief explanation of your modification and improvement for both Task 1 and Task 2
- Zip your files and name it to "ee585_carla_($ student_id).zip"
- Submit it via KLMS

Thank you for Listening