CARLA-AUTOWARE BRIDGE

[Detailed documentation]

SM Hari Prasanth (sm.hariprasanth@gmail.com)
August 31, 2023

Table of Contents

Within this technical document, you will find comprehensive instructions pertaining to Carla, the carla-ros-bridge, and the carla-autoware-bridge. The content presented encapsulates the insights, expertise, and efforts accumulated over the course of my summer work. While the information provided is of a general nature, it's important to note that the software's installation locations are predicated upon the specific workstation in consideration.

1. System requirements:

The following requirements are followed in the current simulation implementation:

- Ubuntu 18.04
- CUDA 9.1
- ROS Melodic
- Unreal Engine 4.26
- Carla 0.9.13
- Carla-ros-bridge 0.9.13
- Autoware.ai openplanner.1.13

2. System Installations

Follow the system installation from scratch in the order for installing the Carla-autoware simulation for minimal hassles.

2.1 Ubuntu installation:

Follow the instructions provided at the link to install Ubuntu 18.04 on an external hard drive [Link]

Here are some important points to consider:

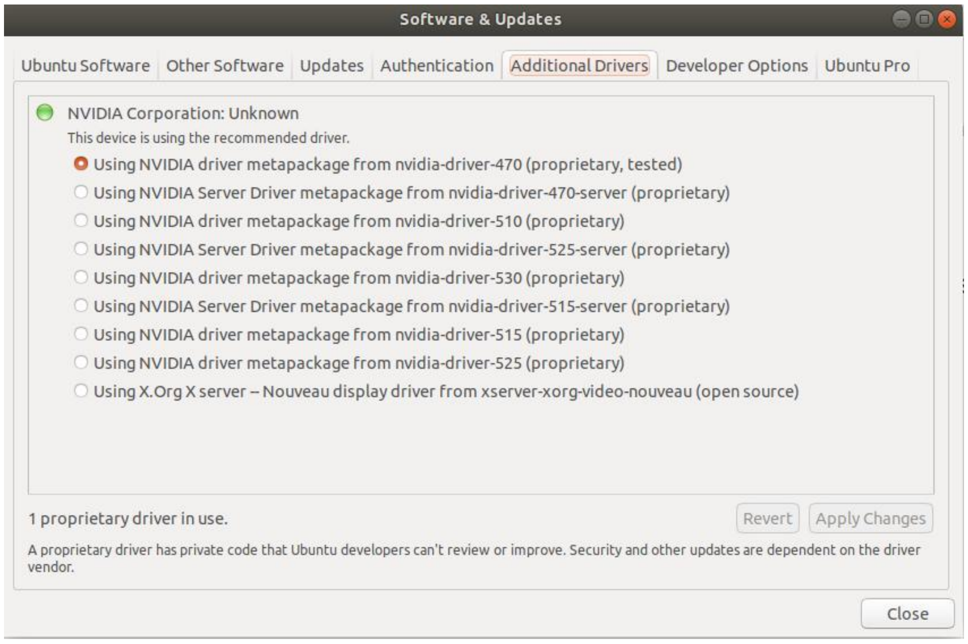
1. While proceeding with the installation, ensure that you choose an unoccupied or available hard drive.
2. If you are installing onto a distinct disk, such as a new SSD in our context, make sure to designate the available space under a new device label like "/dev/sda/".

Note: In certain cases, the new drive might not be visible in the list. If this occurs, adjust the connection type to AHCI from Intel Rapid Storage Technology in the BIOS settings.

BIOS > Storage > SATA Operation > AHCI (Shift from RAID On)

2.2 Nvidia driver installation:

If the graphics card isn't recognized within Ubuntu following a dual-boot setup – indicated by unexpected behavior of the "*nvidia-smi*" command – it's recommended to update the driver. To do so, navigate to "Software & Updates" and access the "Additional Drivers" section (as depicted in the image below). Confirm that the latest proprietary and tested driver is installed.



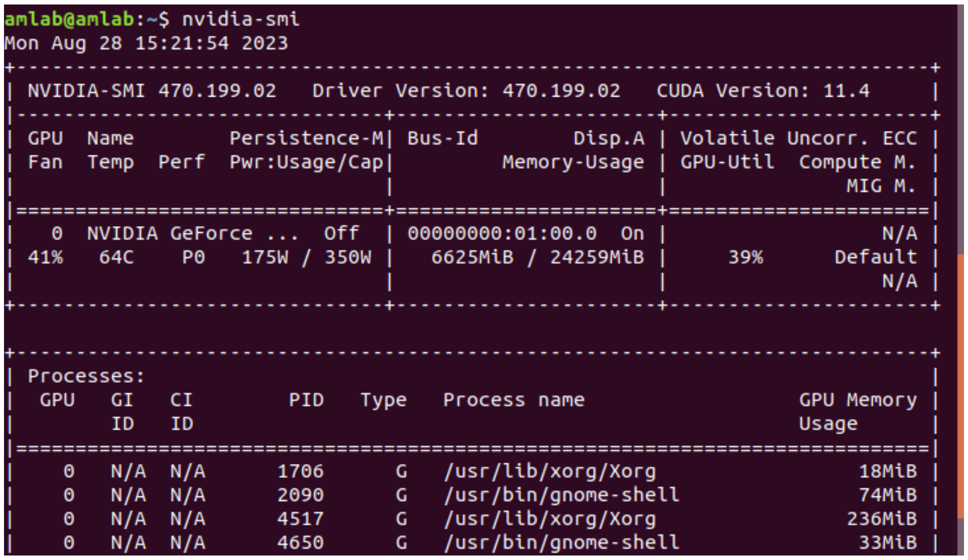
NVIDIA Corporation: Unknown
This device is using the recommended driver.

[✓ Using NVIDIA driver metapackage from nvidia-driver-470 (proprietary, tested)]
Using NVIDIA Server Driver metapackage from nvidia-driver-470-server (proprietary)
Using NVIDIA driver metapackage from nvidia-driver-510 (proprietary)
Using NVIDIA Server Driver metapackage from nvidia-driver-525-server (proprietary)
Using NVIDIA driver metapackage from nvidia-driver-530 (proprietary)
Using NVIDIA Server Driver metapackage from nvidia-driver-515-server (proprietary)
Using NVIDIA driver metapackage from nvidia-driver-515 (proprietary)
Using NVIDIA driver metapackage from nvidia-driver-525 (proprietary)
Using X.Org X server – Nouveau display driver from xserver-xorg-video-nouveau (open source)

1 proprietary driver in use.

A proprietary driver has private code that Ubuntu developers can't review or improve. Security and other updates are dependent on the driver vendor.

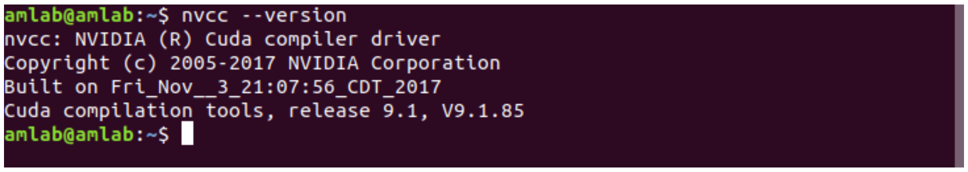
After this, you will observe the resultant output upon executing the "*nvidia-smi*" command:



```
amlab@amlab:~$ nvidia-smi
Mon Aug 28 15:21:54 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.199.02   Driver Version: 470.199.02   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0  On |                  N/A |
| 41%   64C    P0   175W / 350W |   6625MiB / 24259MiB |     39%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1706      G   /usr/lib/xorg/Xorg                 18MiB |
|    0   N/A  N/A      2090      G   /usr/bin/gnome-shell               74MiB |
|    0   N/A  N/A      4517      G   /usr/lib/xorg/Xorg                236MiB |
|    0   N/A  N/A      4650      G   /usr/bin/gnome-shell               33MiB |
```

IMPORTANT: Note that the CUDA version mentioned here is not the actual CUDA version installed but the maximum CUDA version supported by the current graphics card driver. To know the current CUDA version, execute the *"nvcc –version"* command, and you can see similar to the following output.



```
amlab@amlab:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri_Nov__3_21:07:56_CDT_2017
Cuda compilation tools, release 9.1, V9.1.85
amlab@amlab:~$ 
```

2.3 ROS Melodic installation

Follow the instructions provided at the link to install ROS melodic. [Link]

Once ROS is installed, source the ROS environment in the ***"~/.bashrc"*** file by executing the following commands on a terminal.

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc

$ source ~/.bashrc
```

2.4 Unreal engine and Carla Installation

Follow the instructions provided at the link to build the Unreal engine and Carla on Linux from source. [Link]. Ensure that you have navigated to the appropriate page for installing Carla version 0.9.13. You can confirm this by locating the version number in the bottom-right corner of the page.

While following the initial step of part 2 installation, which involves cloning the Carla repository using Git, it's essential to ensure that you perform this cloning process within the "Documents" repository, as per our current setup. Even though you're adhering to the instructions specific to version 0.9.13 installation, keep in mind that Git cloning defaults to the latest version. Consequently, you'll need to manually switch to the specified branch of the intended version. To do so, follow these commands:

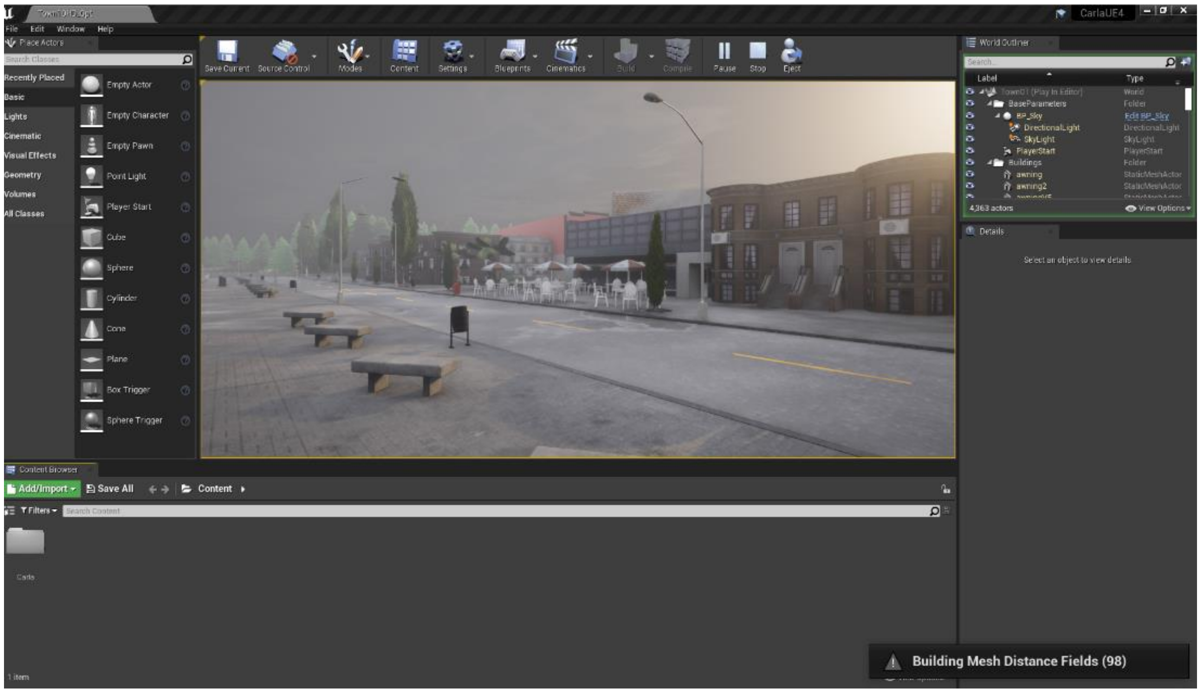
```
$ cd ~/Documents/carla/

$ git checkout 0.9.13

$ git status
```

When compiling the Python API for Carla, it is recommended to use the ".egg" approach over the ".whl" files.

Note: When initializing a map for the first time, the loading process might require approximately 2 to 3 hours to complete the entire map. Notifications will be displayed in the lower-left corner (refer to the image below). Although these notifications are not constantly visible, they can be monitored in the terminal where the Carla instance was launched. This process is not continuous, allowing you to close the map at any point and subsequently reopen it, with the updates resuming from where they were previously left off.



Important: If the simulation is running at a very low FPS rate, go to `Edit` -> `Editor preferences` -> `Performance` in the Unreal Engine editor and disable `Use less CPU when in background`.

Upon successfully building Carla from the source, integrate the subsequent environment variables into the "~/.bashrc" file:

```
export CARLA_ROOT=/home/amlab/Documents/carla

export PYTHONPATH=$PYTHONPATH:$CARLA_ROOT/PythonAPI/carla/dist/carla-0.9.13-py2.7-linux-x86_64.egg:$CARLA_ROOT/PythonAPI/carla
```

Note: The usage of the python2.7 egg file is applied here. Although other versions of python egg files can be utilized, they might lead to potential errors that require environment adjustments. The forthcoming instructions are predicated on the python2.7 egg file.

2.5 Carla-Autoware bridge installation

Retrieve the **carla_autoware_bridge** zip file from this [location] and extract it into the ~/Documents directory. This will establish the subsequent folder structure: home -> Documents -> carla_autoware_bridge -> Autoware.ai.openplanner & carla-autoware.

The initial step involves compiling the Autoware AI openplanner. Source the build and proceed to compile carla-autoware. To accomplish this, execute the commands within the same terminal session:

```
$ cd ~/Documents/carla_autoware_bridge/autoware.ai.openplanner/

$ AUTOWARE_COMPILE_WITH_CUDA=1 catkin_make -DCMAKE_BUILD_TYPE=Release -
DCMAKE_C_COMPILER=/usr/bin/gcc-6

$ source devel/setup.bash

$ cd ~/Documents/carla_autoware_bridge/carla-autoware/catkin_ws/

$ catkin_init_workspace src/

$ catkin_make
```

Retrieve the **carla_maps** and **kmlmaps_from_oxdr** zip file from the [location1], [location2] and extract it into the ~/Documents directory. This will result in the following directory arrangement: home -> Documents -> carla_maps & kmlmaps_from_oxdr.

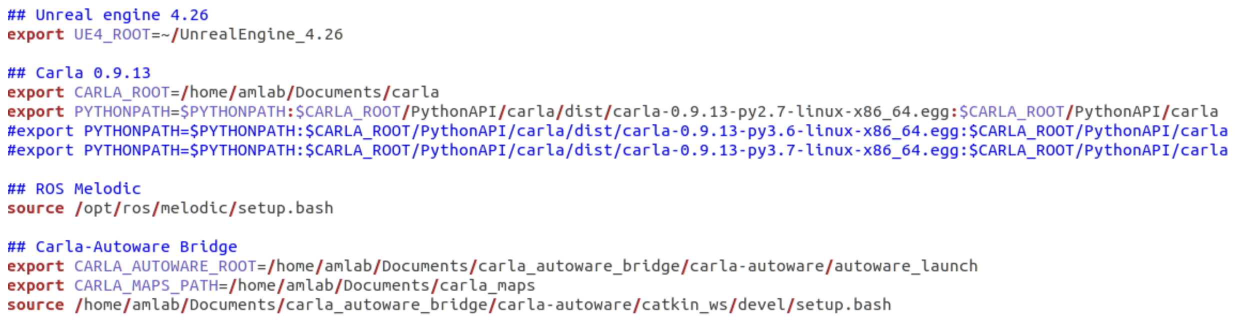
Once you have successfully followed all the outlined steps, proceed to implement the subsequent modifications in your "~/.bashrc" file:

```
export  CARLA_AUTOWARE_ROOT=/home/amlab/Documents/carla_autoware_bridge/carla-
autoware/autoware_launch

export CARLA_MAPS_PATH=/home/amlab/Documents/carla_maps

source /home/amlab/Documents/carla_autoware_bridge/carla-
autoware/catkin_ws/devel/setup.bash
```

At this point, the setup is primed for execution and exploration. The final configuration of your "~/.bashrc" file should resemble the following:



```
## Unreal engine 4.26
export UE4_ROOT=~/UnrealEngine_4.26

## Carla 0.9.13
export CARLA_ROOT=/home/amlab/Documents/carla
export PYTHONPATH=$PYTHONPATH:$CARLA_ROOT/PythonAPI/carla/dist/carla-0.9.13-py2.7-linux-x86_64.egg:$CARLA_ROOT/PythonAPI/carla
#export PYTHONPATH=$PYTHONPATH:$CARLA_ROOT/PythonAPI/carla/dist/carla-0.9.13-py3.6-linux-x86_64.egg:$CARLA_ROOT/PythonAPI/carla
#export PYTHONPATH=$PYTHONPATH:$CARLA_ROOT/PythonAPI/carla/dist/carla-0.9.13-py3.7-linux-x86_64.egg:$CARLA_ROOT/PythonAPI/carla

## ROS Melodic
source /opt/ros/melodic/setup.bash

## Carla-Autoware Bridge
export CARLA_AUTOWARE_ROOT=/home/amlab/Documents/carla_autoware_bridge/carla-autoware/autoware_launch
export CARLA_MAPS_PATH=/home/amlab/Documents/carla_maps
source /home/amlab/Documents/carla_autoware_bridge/carla-autoware/catkin_ws/devel/setup.bash
```

3. Usage guidelines and Illustrations

3.1 Unreal engine

Note: It is unnecessary to initiate Unreal Engine separately if your intention is to launch the Carla simulator. The initiation of the Carla simulator will automatically trigger the launch of the Unreal Engine.

If you wish to independently launch the Unreal Engine, you can do so by executing the following command in a terminal.

```
$ cd ~/UnrealEngine_4.26/Engine/Binaries/Linux/

$ ./UE4Editor
```

3.2 Carla

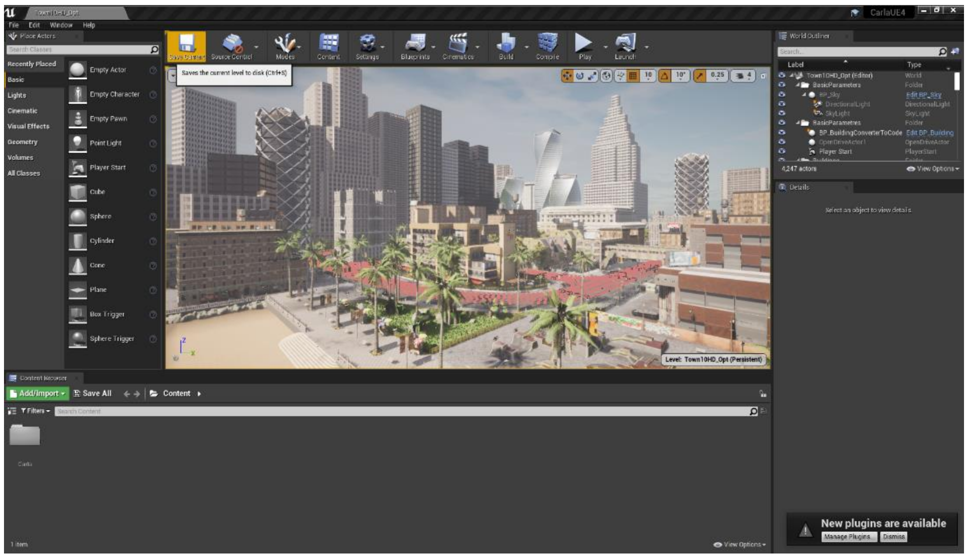
Open a new terminal and execute the following commands to launch Carla simulator along with Unreal engine:

```
$ cd ~/Documents/carla/

$ make launch
```

Terminating this terminal will close the Carla simulator altogether.

Hit the "Play" button on the top panel of Unreal engine to start the Carla server. Note that this will always launch Town10_Opt by default.


Saves the current level to disk (Ctrl+S)

Select an object to view details

New plugins are available

If you wish to change into a new town map, open a new terminal and run the following commands:

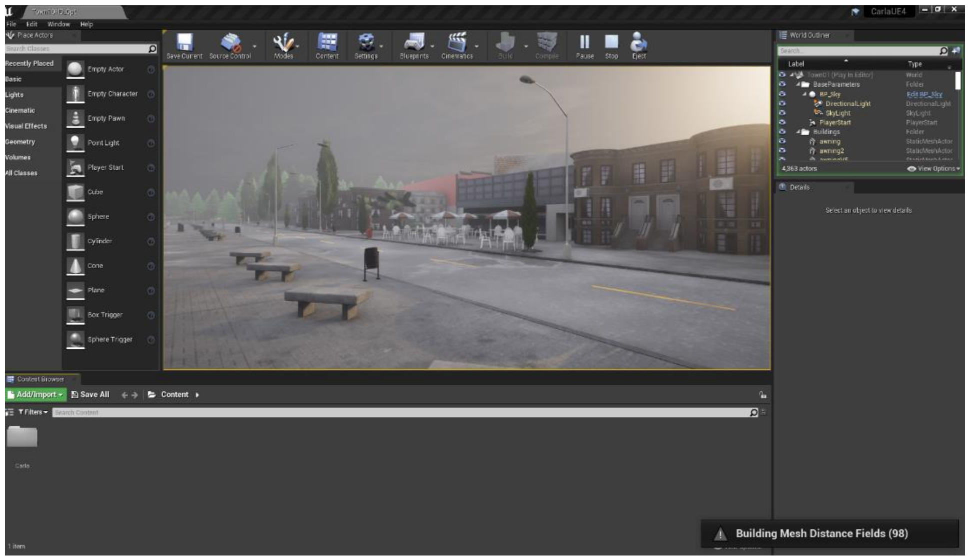
```
$ cd ~/Documents/carla/PythonAPI/util/

$ python config.py –map <town-name>
```

The available town maps are stored in this location:

~/Documents/carla/Unreal/CarlaUE4/Content/Carla/Maps

When launching a new town map for the first time, the loading process may take several hours to complete. It's essential to wait until loading is finished to avoid potential simulator crashes. Notably, the loading is not an ongoing operation, allowing you the flexibility to close the process at any point. Upon subsequent openings of the map, the building process will recommence from the point where it was previously interrupted. In the image below you can see the "Building Mesh distance fields" message appears in the lower-right corner. This popup window might not seen always but you can observe the updates in the Carla terminal.

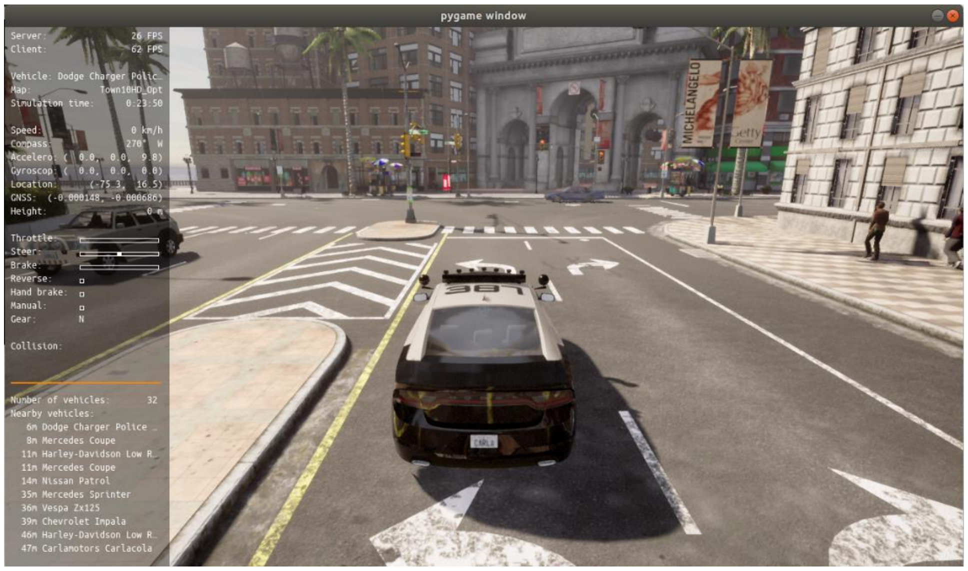


To introduce a vehicle and manually navigate it through the city, perform the following actions in a new terminal:

```
$ cd ~/Documents/carla/PythonAPI/examples/

$ python manual_control.py
```

This will spawn a random vehicle at a random place within the town. A new PyGame window will open, as illustrated below:



```
Server:                26 FPS
Client:                62 FPS

Vehicle: Dodge Charger Polic…
Map:             Town10HD_Opt
Simulation time:      0:23:50

Speed:                 0 km/h
Compass:              270°  W
Accelero: (  0.0,  0.0,  9.8)
Gyroscop: (  0.0,  0.0,  0.0)
Location:        (-75.3, 16.5)
GNSS:  (-0.000148, -0.000686)
Height:                   0 m

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

Collision:

Number of vehicles:       32
Nearby vehicles:
   6m Dodge Charger Police …
   8m Mercedes Coupe
  11m Harley-Davidson Low R…
  11m Mercedes Coupe
  14m Nissan Patrol
  35m Mercedes Sprinter
  36m Vespa Zx125
  39m Chevrolet Impala
  46m Harley-Davidson Low R…
  47m Carlamotors Carlacola
```

You can drive it with the arrow keys or WASD keys. Press **H** for more help. Press **P** to check Carla's Autopilot.

For more advanced control over vehicle selection and spawn points, you can pass additional parameters to the terminal command. Use the "***python manual_control.py --help***" command to explore available options.

To generate traffic and introduce additional actors to the simulator, execute the following commands in a new terminal:

```
$ cd ~/Documents/carla/PythonAPI/examples/

$ python generate_traffic.py
```

This will create 30 vehicles and 10 walkers across the town, all operating in autopilot mode. Terminating this terminal session using Ctrl+C will remove all spawned actors.

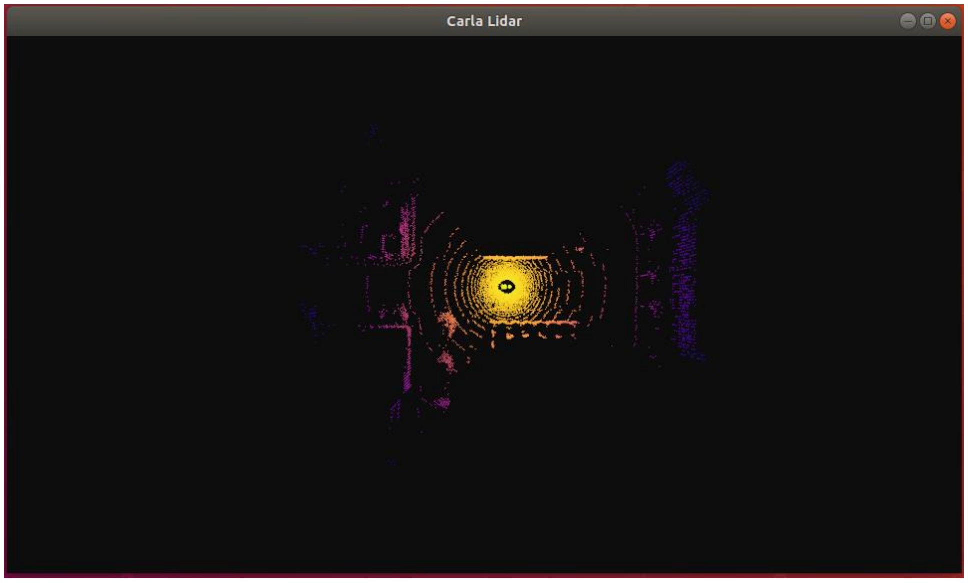
To directly visualize the open3d lidar output from Carla, execute the following commands in a new terminal:

```
$ cd ~/Documents/carla/PythonAPI/examples/

$ python3 open3d_lidar.py
```

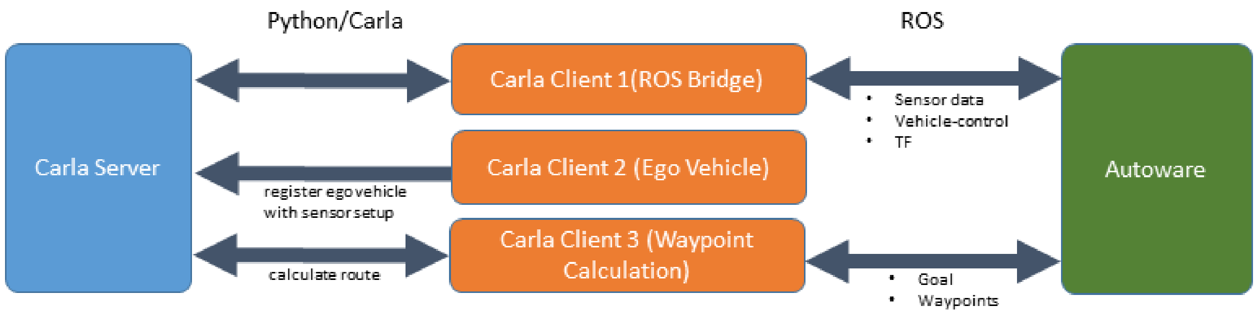
Note: Open3d requires Python3, so we need to change the egg file in the **"~/.bashrc"** file to python 3.6 or 3.7 and then execute the command. After this, change it back to 2.7.

This will spawn a vehicle within the Carla town and set it in a loop. A new screen will appear displaying lidar point clouds, as shown in the image below. It's important to note that no user control is possible in this context.



3.3 Carla-Autoware bridge

The carla-autoware bridge consists of Autoware.ai and carla-ros bridge. The design can be seen in the following image: [Reference]


Python/Carla
ROS
Carla Server
Carla Client 1(ROS Bridge)
Carla Client 2 (Ego Vehicle)
Carla Client 3 (Waypoint Calculation)
Autoware
register ego vehicle with sensor setup
calculate route
• Sensor data
• Vehicle-control
• TF
• Goal
• Waypoints

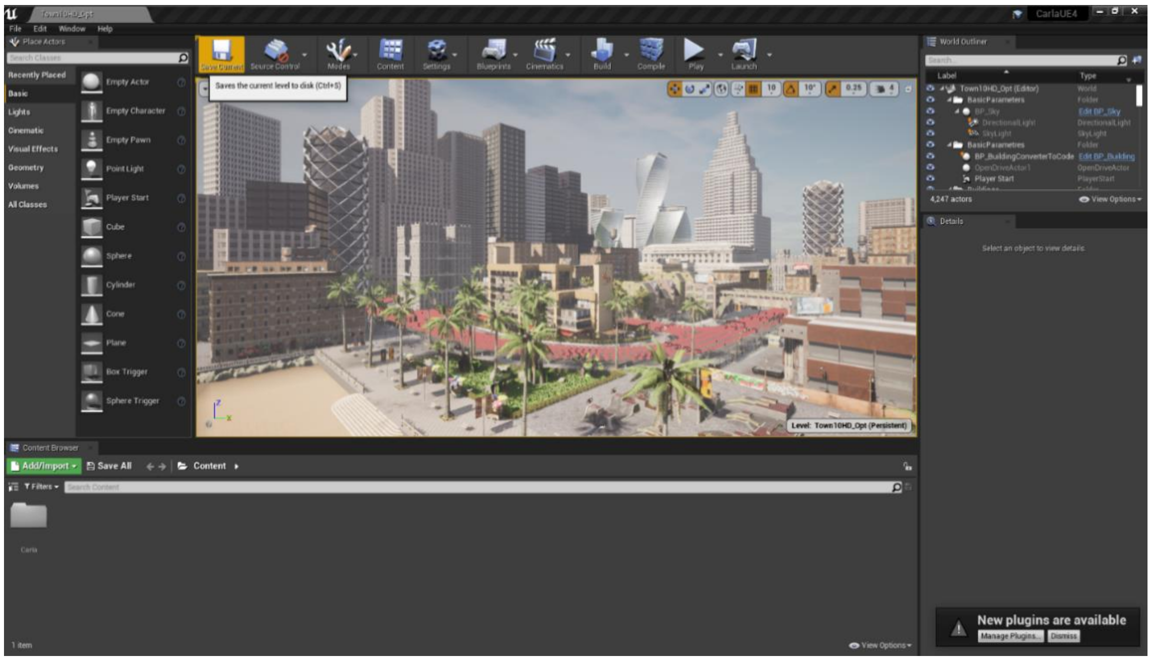
To establish the carla-autoware bridge follow the steps:

1. To initiate the Carla simulator, launch a new terminal and input the following command:

```
$ cd ~/Documents/carla

$ make launch
```

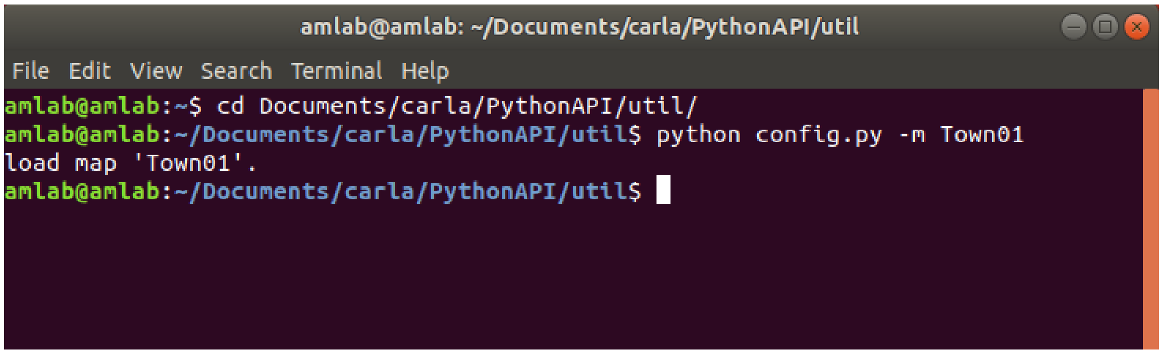
2. After the simulator has completely launched, an interface will be displayed. To run the simulator, click the play button within the Unreal Engine interface.



3. Note that the launcher always starts with Town 10 from the simulator. To switch to Town 01, use the subsequent command:

```
$ cd ~/Documents/carla/PythonAPI/utils/

$ python config.py --map Town01
```



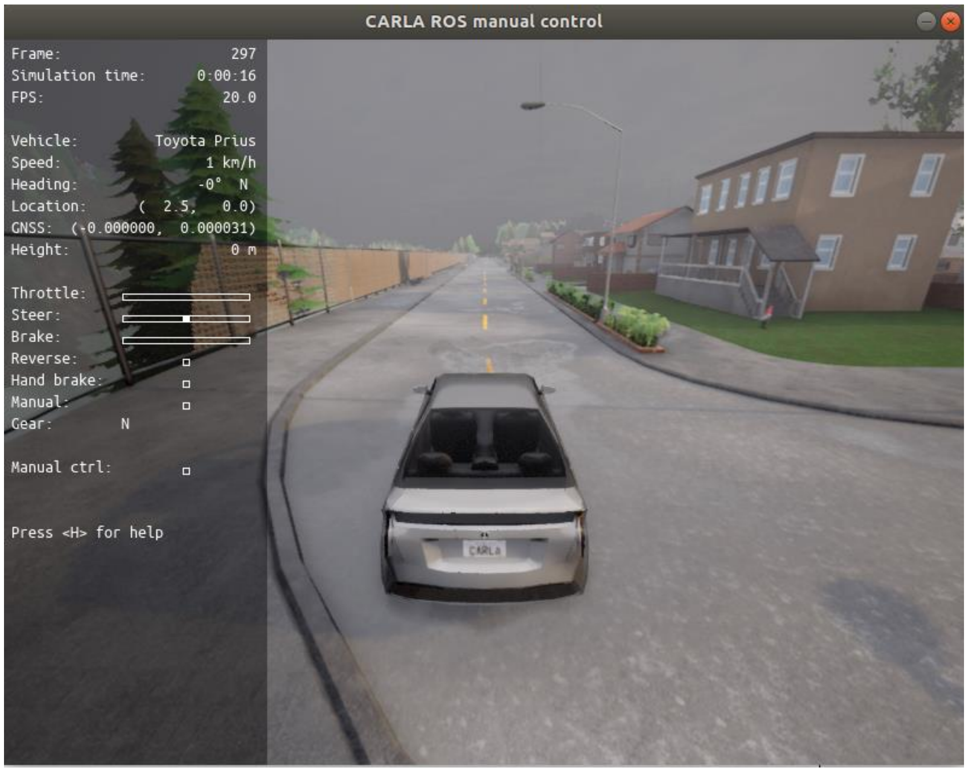
```
amlab@amlab: ~/Documents/carla/PythonAPI/util
File Edit View Search Terminal Help
amlab@amlab:~$ cd Documents/carla/PythonAPI/util/
amlab@amlab:~/Documents/carla/PythonAPI/util$ python config.py -m Town01
load map 'Town01'.
amlab@amlab:~/Documents/carla/PythonAPI/util$
```

4. Next, launch the carla-autoware bridge. For that, open a new terminal and enter the following command:

```
$ cd ~/Documents/carla_autoware_bridge/carla-autoware/autoware_launch/

$ roslaunch devel.launch
```

This will load the following PyGame window:



```
Frame:                  297
Simulation time:    0:00:16
FPS:                   20.0

Vehicle:        Toyota Prius
Speed:               1 km/h
Heading:             -0°  N
Location:       (  2.5,   0.0)
GNSS:  (-0.000000,  0.000031)
Height:                 0 m

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

Manual ctrl:


Press <H> for help
```

Upon execution, this command will initialize a bridge connecting ROS and Carla, as well as another bridge linking Carla with Autoware through ROS. Use the ***"rostopic list"*** command to access the available ROS topics.

(Note: The launch file can be executed directly without the need to modify the Carla map to Town01. The launch file is designed to handle the map change itself. However, on rare occasions, an error might occur and lead to termination after altering the map. If this occurs, simply rerun the launch file to resolve the issue.)

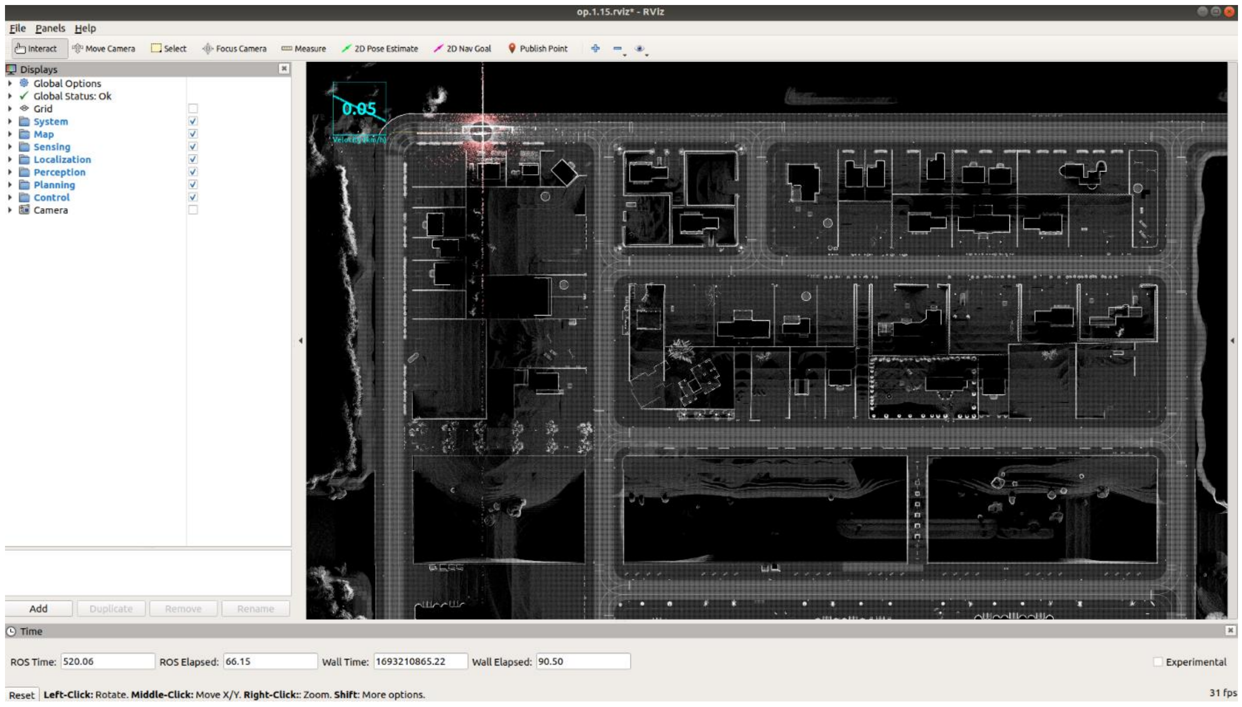
Important: Press **B** to turn off the control commands from Autoware. It will override the control, and you can drive it manually using arrow keys or WASD keys. Press **B** again for Autoware to take control. To check Carla's autopilot, known as "Traffic Manager", press **B** first, then press **P**.

5. Run the following command to open the Rviz:

```
$ cd ~/Documents/carla_autoware_bridge/autoware.ai.openplanner/

$ rosrun rviz rviz -d op.1.15.rviz
```

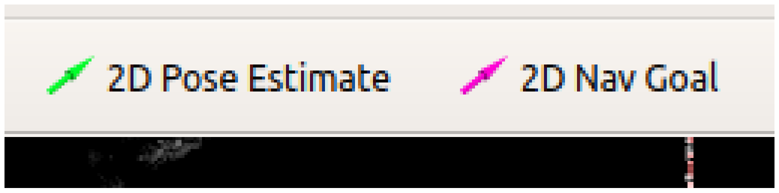
This command will launch Rviz, displaying the car's localization within the town map.



ROS Time: 520.06 ROS Elapsed: 66.15 Wall Time: 1693210865.22 Wall Elapsed: 90.50

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift:** More options.

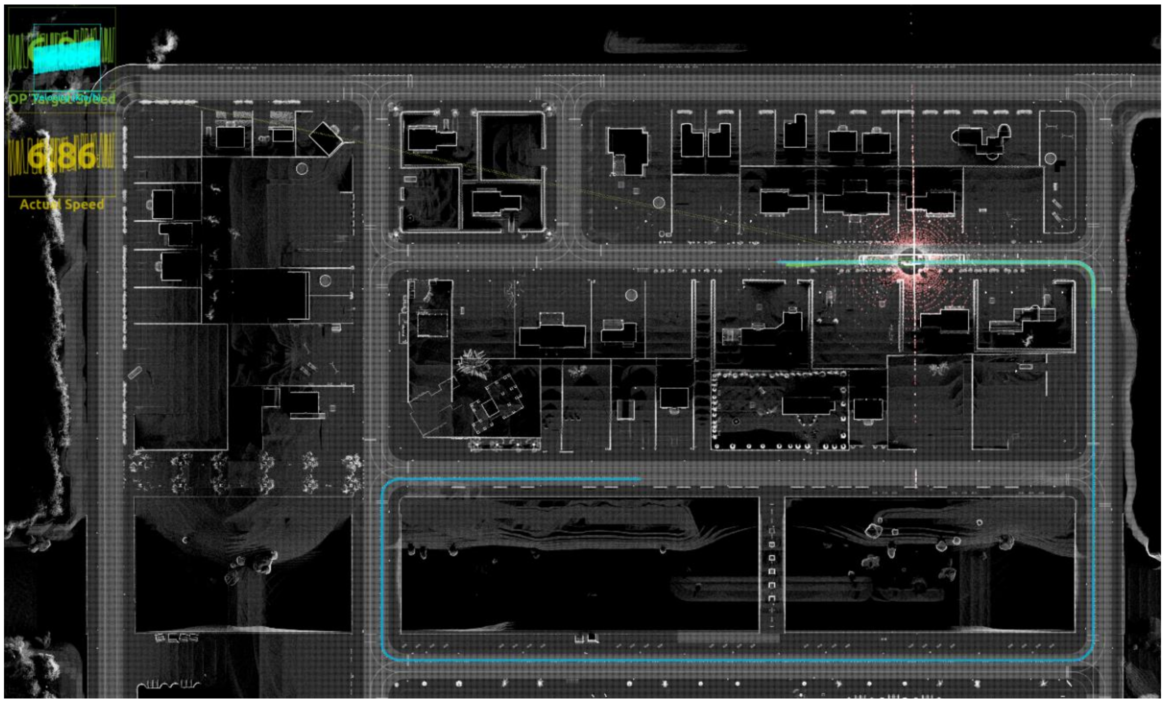
While using Rviz, you'll encounter the following options:



2D Pose Estimate: This option permits you to alter the car's spawn point. By selecting this option and clicking on a location within the road area of the map, the car will respawn at that point.

2D Nav Goal: Use this option to designate a goal point. By selecting this option and clicking on a spot within the road area generates a path from the current location to the chosen goal, prompting the car to navigate until it reaches the goal point.

Upon selecting a goal pose, Rviz will illustrate the generated path, as depicted in the next image.



Note: If you decide to close the Rviz window, a prompt will appear to save any changes. Opt to close without saving unless you've made alterations to the Rviz configuration and wish to retain them.

3.4 Current progress

The vehicle has the capability to spawn at any location within the map, establish its position through localization, generate a path towards the designated goal pose from Rviz, and ultimately navigate to that goal. Nonetheless, certain PID tuning is essential within the control segment (you can locate the relevant parameters within the "***high_level_controller_direct.launch***" file situated in the carla-autoware/ Autoware_launch directory). It's worth noting that there might be slight steering oscillations, and occasional instances where the vehicle may not come to a complete stop despite reaching the intended goal position.

4. Insights about the Carla-Autoware bridge

The initiation of the Carla-Autoware bridge takes place upon executing the ***devel.launch*** launch file. At a higher level, the initialization of parameters such as host, port, and town are necessary. If you alter the town from *Town01* to a different one, remember to update the kml file path accordingly. This configuration should function seamlessly, provided the specified town name exists, aligns with the correct point cloud file within the ***carla_maps*** repository, and is associated with a corresponding kml file under ***kmlmaps_from_oxdr*** repository.

It is recommended to refrain from altering the role name from "***ego_vehicle***" since this name is embedded in the ROS topics related to sensor data. Modifying the role name necessitates changes in multiple locations.

Additionally, you can define the spawn point here; if no collision is detected, the vehicle will spawn at this designated location. In the event of a collision, a warning will be issued, and the vehicle will spawn at a random point.

This launch file triggers the subsequent nodes and scripts in the same sequential order:

- Carla-autoware bridge with manual control
  - Carla manual control
  - Carla ros bridge
  - Carla ego vehicle
  - Carla points map loader
  - Tf remapping
  - Carla fake control
- Autoware
  - Ground filter
  - Localization
  - Vel pose connector
  - Object detection tracking
  - Global planner
  - Common params
  - Local planner
  - Traffic light detector
  - High level direct controller

4.1 Carla-autoware bridge with manual control

This will initiate a series of nodes, each corresponding to the Carla simulator's functionalities. These nodes establish a ROS bridge that facilitates bidirectional data exchange via ROS messages and topics. In the "***carla_autoware_bridge.launch***" file, a vehicle filter is available to facilitate the selection of a specific vehicle. Another crucial argument in this launch file is the sensor definition file, necessitating the provision of a path to a JSON file. Within this JSON file, sensors to be spawned, along with their respective locations, are specified. It's important to bear in mind that each sensor is associated with a unique identifier; changing this identifier requires adjustments in other sections (such as the "***tf.launch***" file) for proper functioning.

1. **Carla manual control:** Analogous to the "manual_control.py" script in Carla, this function leverages ROS messages for all control operations, granting access to sensor data through ROS topics.
2. **Carla ROS bridge:** Serving as a pivotal script, this function orchestrates the communication between Carla's PythonAPI and ROS messages and topics.
3. **Carla ego vehicle:** This node is responsible for spawning a designated vehicle at the specified spawn point, equipped with the sensors outlined in the provided JSON file.
4. **Carla points map loader:** This node facilitates the loading of the KML map file, generating various objects contained within the map, including nodes, junctions, and traffic sign data.
5. **TF remapping:** This node ensures alignment within the transformation tree for the map, vehicle, and all sensors, ensuring coherent data representation.
6. **Carla fake control:** Functioning as an intermediary, this node receives control commands from "op_direct_controller" and subsequently translates them into Carla vehicle commands. Note: If the intention is to utilize Ackermann control, this node should be substituted with the Ackermann control functionality.

4.2 Autoware

This sequence will initialize a succession of nodes originating from Autoware.ai (version 1.13). Assuming all prior steps have proceeded without issues (including map creation, vehicle spawning with sensors, and establishment of the transformation tree), this phase involves interfacing with the corresponding ROS topics and executing the Autoware nodes.

1. **Ground filter:** This node undertakes the task of filtering lidar data, segregating it into ground points and non-ground points.
2. **Localization:** Employing the NDT matching algorithm, this node is responsible for determining the current position of the ego vehicle within the map.
3. **Vel pose connector:** By renaming the ROS topics related to velocity and pose, this node ensures uniformity in their representation.
4. **Object detection tracking:** Utilizing the kf_contour algorithm, this node detects and tracks objects through the integration of lidar and camera data.
5. **Global planner:** Generating a comprehensive path plan from the current location to the goal point (provided by Rviz), this node serves as the global planning component.
6. **Common params:** This launch file consolidates most parameters utilized across all Autoware nodes, providing a centralized repository for these settings.
7. **Local planner:** With a focus on the immediate vicinity, this node generates a local path plan leading from the current point to the subsequent intermediate goal point.
8. **High level direct controller:** Ensuring adherence to the designated path and its maintenance, this node governs the vehicle's high-level control to achieve desired trajectory tracking.

5. Troubleshooting and Debugging

5.1 Unreal Engine

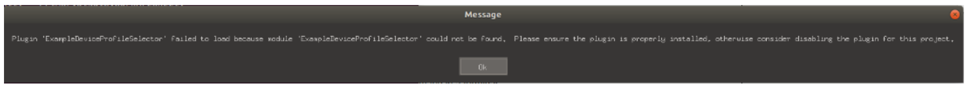
1. Unreal engine freezes at 95 % loading after the launching

Solution: Disable steamVR [Link]

How: Go to the file location: home/amlab/UnrealEngine_4.26/Engine/Plugins/Runtime/Steam -> SteamVR.uplugin -> "EnabledByDefault": ~~true~~ false

2. ExampleProfileDeviceSelector not found error.

While launching UE Editor alone, the following error pops up:



```
Message
Plugin 'ExampleDeviceProfileSelector' failed to load because module 'ExampleDeviceProfileSelector' could not be found.  Please ensure the plugin is properly installed, otherwise consider disabling the plugin for this project.
```

Solution: make the Unreal engine build again. Run the following commands and launch the Unreal engine again. It will take only few minutes. [Link]

```
$ cd ~/UnrealEngine_4.26

$ ./Setup.sh && ./GenerateProjectFiles.sh && make
```

5.2 Carla

1. Simulator is not running. / Simulator timeout.

Solution: Make sure the “Play” button in the simulator is enabled. If so, try to change into a different map. Still, if it is not working, close the terminal and rerun it again.

2. No module named ‘Carla’

Solution: Make sure the correct egg file is mentioned in the “~/.bashrc” file. If still the error exists, run the command “**make LibCarla**” inside the Carla root repository and relaunch the Carla simulator.

5.3 Carla-Autoware bridge

1. Error during catkin_make: wrong GNU version

Solution: It is important to have gcc-6 compiler installed and pass it as an argument during catkin_make.

2. PyGame window opens but vehicle didn’t spawn

Solution: Make sure the role_name is ego_vehicle. If you change it to something else, you need to change it in more than one places.

6. Assure Mapping tools: To generate KML map

The current setup of Carla-autoware requires map files in kml format for purposes such as visualization in Rviz, path planning, and other features. Consequently, there is a requirement to convert the existing Carla map towns into kml format. To achieve this, the installation of Assure Mapping Tools is essential. The repository for this tool can be cloned from the [Link], and currently, it has been cloned and set up in the *~/Documents* directory. This tool facilitates the conversion of maps from both oxdr and pcd formats into kml format. In our specific use case, the conversion from oxdr to kml is necessary to access more extensive data.

It's important to note that the Docker environment lacks access to external files unless explicitly specified. Thus, it's vital to provide the path to the folder containing the oxdr map files via the terminal. In our instance, the oxdr files are stored in this repository:

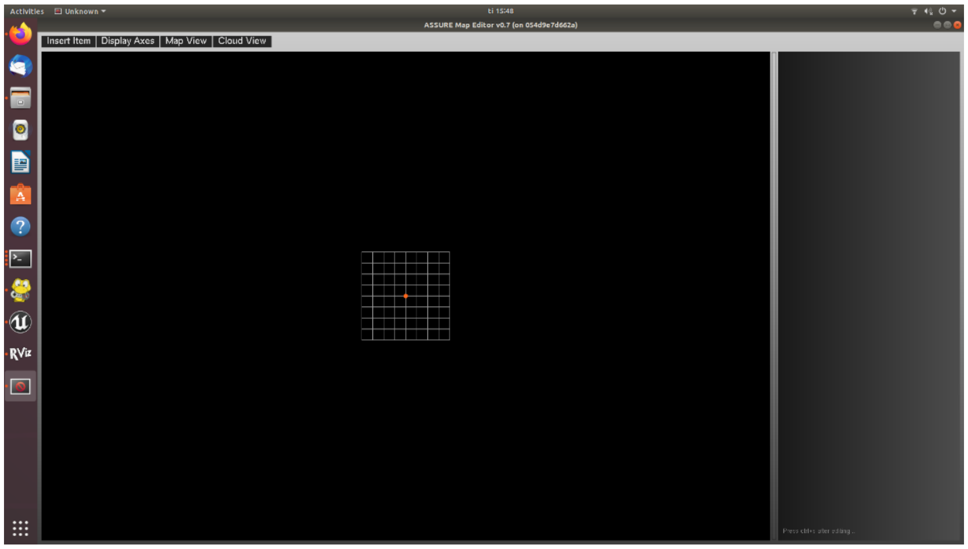
~/Documents/carla/Unreal/CarlaUE4/Content/Carla/Maps/OpenDrive

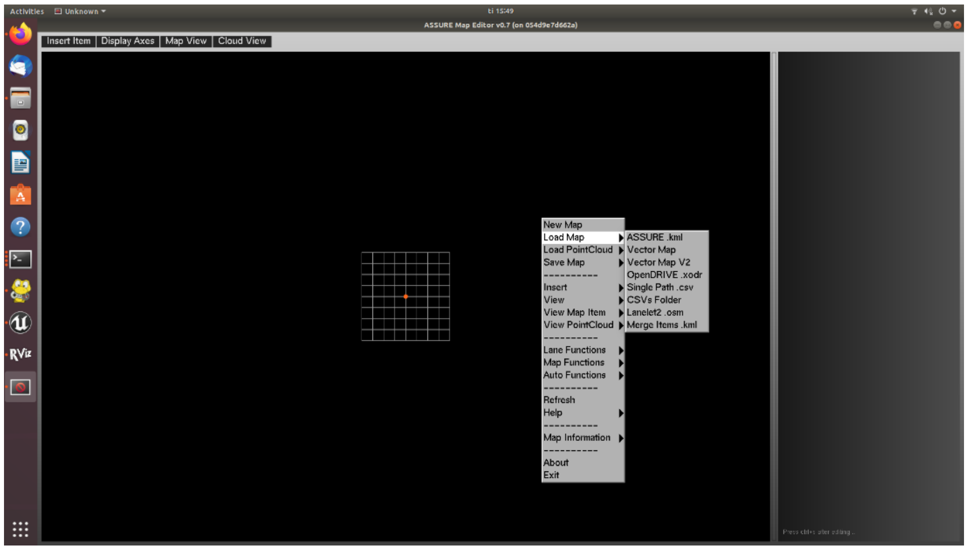
To launch the Assure Mapping Tools, execute the following command:

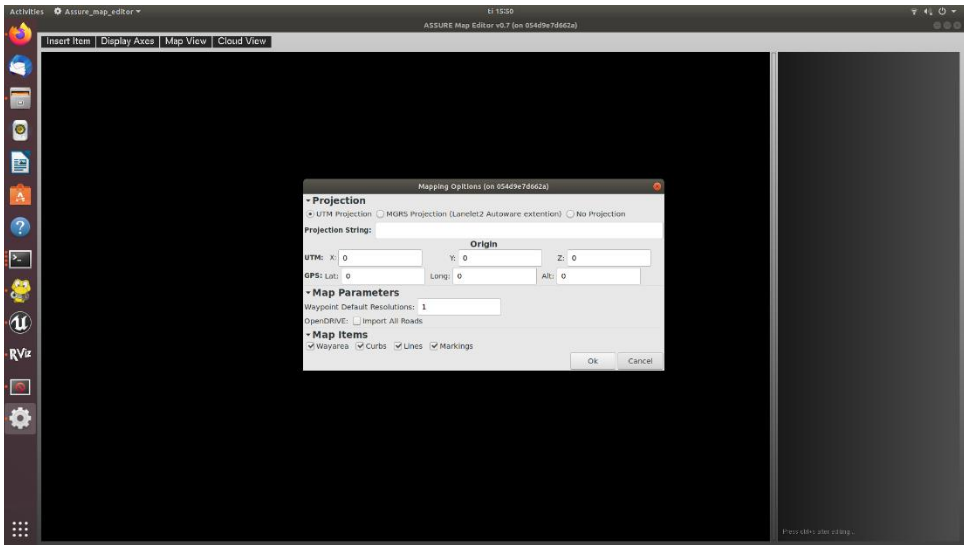
```
$ cd ~/Documents/assuremappingtools/

$ ./run-docker -w
/home/amlab/Documents/carla/Unreal/CarlaUE4/Content/Carla/Maps/OpenDrive/
```

Once launched, you will be presented with the Assure map editor. By right-clicking within the editor, you can load a map by navigating through the options: ***Load Map -> OpenDRIVE .xodr -> (No modifications required) press OK***. In the file manager, browse to the "Maps" folder where you can access all the files from the location specified in the terminal command. By selecting an xodr file, you will be able to view the map. To proceed, right-click again and select ***Save Map -> ASSURE .kml***. This will open the file manager for saving. As a result of this action, a .kml file and a .kml.proj.dat file will be generated. Transfer these two files to a location of your choice, and subsequently provide the complete path to the roslaunch file to use them. It's important to save the file exclusively within the "Maps" folder to ensure it is retrievable from the same location specified in the terminal command. The following images provide visual guidance on this process:







Projection

[✓ UTM Projection] MGRS Projection (Lanelet2 Autoware extention) No Projection

Projection String:

Origin

UTM: X: 0 Y: 0 Z: 0

GPS: Lat: 0 Long: 0 Alt: 0

Map Parameters

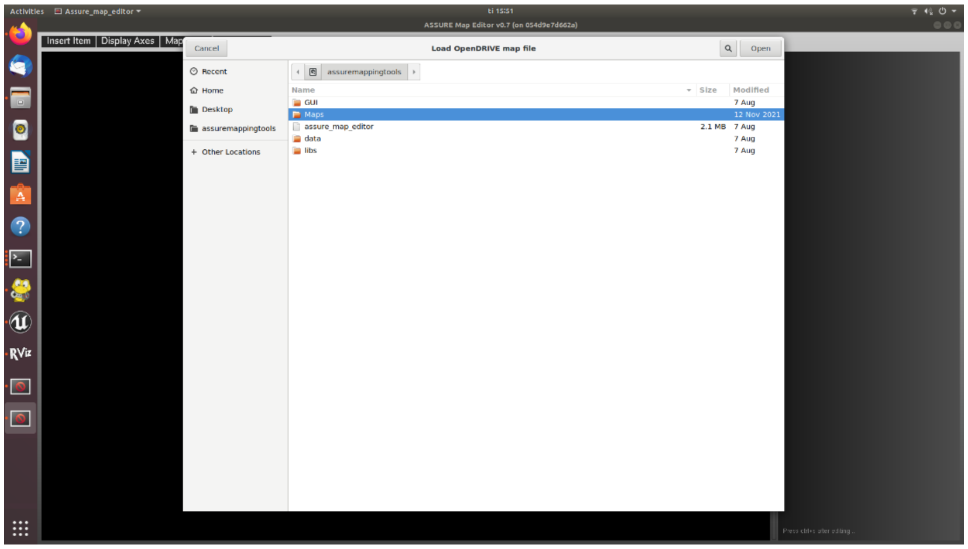
Waypoint Default Resolutions: 1

OpenDRIVE: Import All Roads

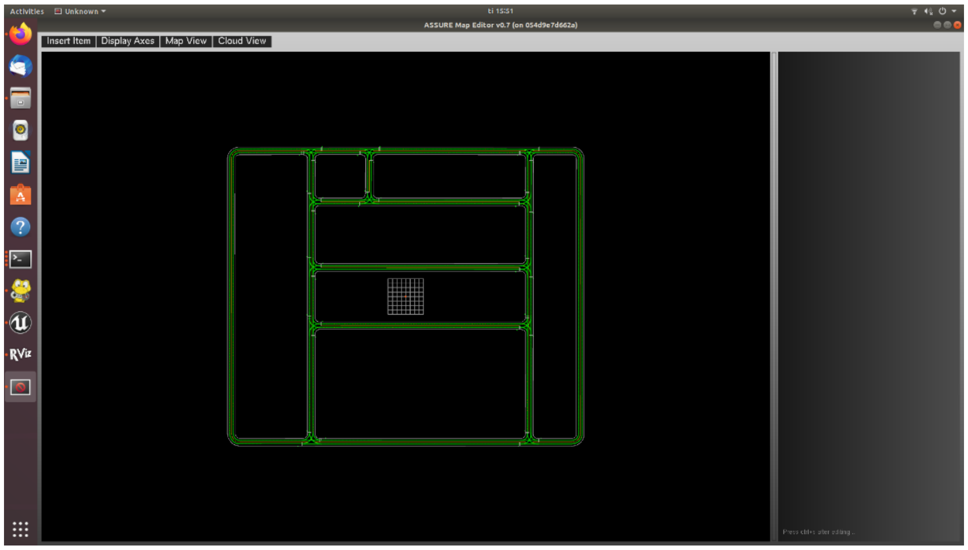
Map Items

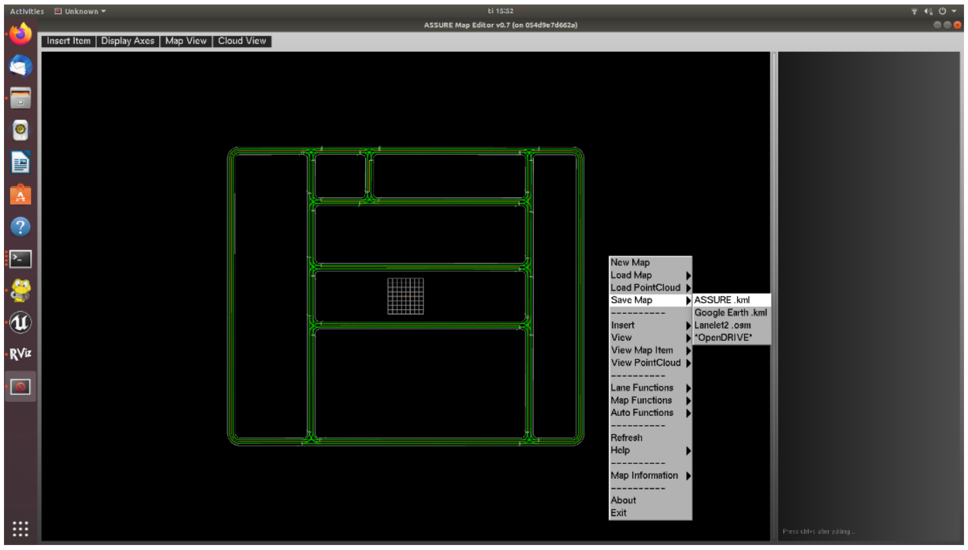
✓ Wayarea ✓ Curbs ✓ Lines ✓ Markings

Ok Cancel



Name	Size	Modified
GUI		7 Aug
Maps		12 Nov 2021
assure_map_editor	2.1 MB	7 Aug
data		7 Aug
libs		7 Aug


Activities
Unknown
ASSURE Map Editor v0.7 (on 054d9e7d662a)
Insert Item
Display Axes
Map View
Cloud View


Activities
Unknown
ASSURE Map Editor v0.7 (on 054d9e7d662a)
Insert Item
Display Axes
Map View
Cloud View
New Map
Load Map
Load PointCloud
Save Map
Insert
View
View Map Item
View PointCloud
Lane Functions
Map Functions
Auto Functions
Refresh
Help
Map Information
About
Exit
ASSURE .kml
Google Earth .kml
Lanelet2 .osm
OpenDRIVE