Carla and ROS

20.04 LTS — CARLA 0.9.10.1 — ROS Noetic

William R. Ebenezaraj
IIT Kharagpur
william@iitkgp.ac.in

Contents

Introduction

About the simulator

CARLA is an open-source self-driving simulator which has grown over the years to be known to represent accurate physics and breathtaking visuals. It comes with a Python API which allows for a quick and easy interface with "actors" (what participants in the simulation environment are called). While the graphics may *seem* to be overly fancy at first, this level of visual accuracy is *vital* for the cameras and lidars to function as they would in the real world.

Why 0.9.10?

CARLA 0.9.10.1 is chosen for present work as it is the latest version to support the carla-ros-bridge and still remain affordable for GPUs with 4GB graphics memory. At the time of writing this, a carla server is running on my machine (GTX 1650 4GB) and nvidia-smi shows a 100% GPU utilization, so it still demands a lot of resources.

Installation

Python3.7 is needed. Also install pygame, cv2, and numpy (do a `pip install` or python will not like it), these are vital packages.

CARLA 0.9.10.1

It is recommended to go for the package installation rather than the debian installation. That way, the installation stays where you can access it easily (and not in some /usr folder). This can be done by getting the Ubuntu release () from CARLA 0.9.10.1 Release on GitHub. Get the link to CARLA_0.9.10.1.tar.gz and run `wget` ⟨link-to-CARLA_0.9.10.1.tar.gz⟩ .

CARLA ROS Bridge

1. Run `wget` https://github.com/carla-simulator/ros-bridge/archive/refs/tags/0.9.10.1.tar.gz , preferably in home, or wherever you are used to keeping your catkin workspaces.
2. Run `mkdir carla-ros-bridge` followed by `tar -xvzf ros-bridge-0.9.10.1.tar.gz -C carla-ros-bridge` .
3. `cd carla-ros-bridge` and rename the folder `ros-bridge-0.9.10.1` to `ros-bridge` .
4. Run `cd carla-ros-bridge`
5. Run `cd ros-bridge && git submodule update --init`

6. Run `mkdir -P ../catkin_ws/src`
7. Run `cd ../catkin_ws/src`
8. Run `ln -s ../../ros-bridge`. This creates a symbolic link to the ros-bridge packages.
9. Run `source /opt/ros/noetic/setup.bash`

The last few steps (to be run inside `catkin_ws`):

1. `rosdep update && rosdep install --from-paths src --ignore-src -r`
2. `catkin_make`

`catkin_make` will probably throw a myriad errors. Most can be solved by trying to build it in a conda environment with python3.7 running (apparently, CARLA supports only 2.7 and 3.7).

Running

This section describes how to run CARLA after a successful installation and build.

Running the CARLA server

Run `./CarlaUE4.sh` inside the carla installation folder to run the Carla server. You should be able to see a window with a view of the default town.

Using the bridge

After sourcing the catkin_ws (`. ~/carla-ros-bridge/catkin_ws/devel/setup.bash`) and the Carla server running, execute

`roslaunch carla_ros_bridgecarla_ros_bridge_with_example_ego_vehicle.launch`

You should be able to see the manual control window (pygame) installed.
If you get errors like `no module named 'carla'`, run something like shown below (replace < uname >, of course, and modify the directory if needed):

Listing 1: ImportError? Try this.

```
export PYTHONPATH=$PYTHONPATH:/home/<uname>/CARLA_0.9.10.1
    /PythonAPI/carla/dist/carla-0.9.10-py3.7-linux-x86_64.egg
```

Creating a package to run your nodes

Create a package using the following command after replacing `test` with the desired package name.

Listing 2: A typical package creation command

```
catkin_create_pkg test std_msgs rospy roscpp carla_msgs
        ackermann_msgs geometry_msgs visualization_msgs
            derived_object_msgs sensor_msgs
```

Customizing things to your liking

If you tried running the , you would have noticed that it spawns the ego vehicle at a random location and with a random type. This is because the actor filter set in the launch file allows for that, and the spawn location is blank.

Specifying the ego vehicle

Open a copy of that file (will be inside carla-ros-bridge/catkin_ws/src/ros-bridge/carla_ros_bridge/launch). Modify the following line to specify the ego vehicle type. You can find a list of blueprint arguments here.

Listing 3: Argument that fixes Ego Vehicle type

```
<arg name="vehicle_filter" default='vehicle.mercedes-benz.coupe'/>
```

Fixing the spawn point

Listing 4: Argument that fixes Ego Vehicle type

```
<arg name="spawn_point" default="-88.30,-21.53,0,0,0,270"/>
```

You may also modify other files (non-example files) to do things more elegantly. However, manual control mode not actually operating in manual mode is a quick way to get a camera following the ego vehicle.

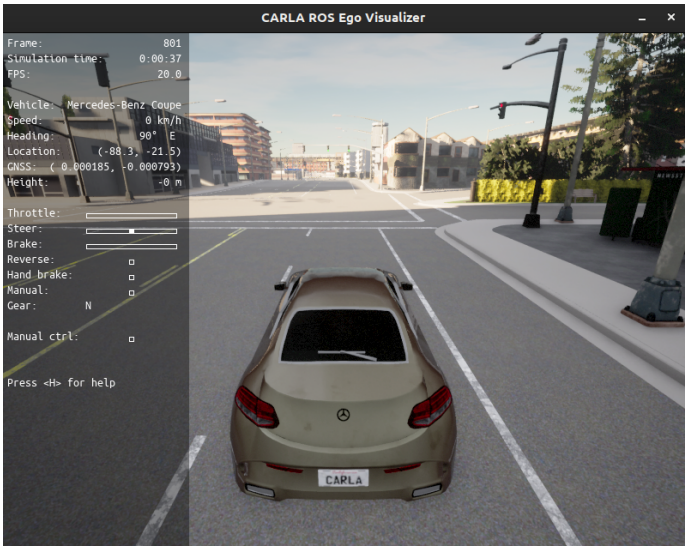

Figure 1: Manual control window modified to be the ego visualizer