Carla simulator implementation notes

Wang Xiang

September 29, 2022

Contents

1 Installation

Official documentation is at: *carla.readthedocs.io*. Youtube official account is *CARLA Simulator*. The version that I installed is 0.9.13. The video on Youtube with title *CARLA 0.9.13* introduces new features of this version, including instance segmentation sensor that can generate instance segmentation ground truth mask. This sensor and semantic segmentation sensor may be useful in my project.

2 Sensors - Camera

2.1 Automatic data generation

Youtube video: *An in depth look at CARLA's sensors*. Github repositories:

- *github.com/Ozzyz/carla-data-export*,
- *github.com/jedeschaud/kitti_carla_simulator*,
- *github.com/jedeschaud/paris_carla_simulator*,

SensorsCamera.py mainly referred the Youtube video mentioned above. Achievement: 1) I can run *SensorsCamera.py* successfully and see the car in a scene on the road. 2) I can also run *CameraSensorsType.py* successfully and see images from 6 different sensors. The vehicle uses autopilot and drives normally on the road. 3) I can also run *LidarRadarSensors3d.py* successfully and see RGB images, lidar and radar point clouds simutaneously.

Problem 1: over exposure of camera sensor. Attempt 1: change the weather parameters. The original weather parameters are

```
cloudiness=20.0, precipitation=0.0, precipitation_deposits=0.0,
   wind_intensity=10.0, sun_azimuth_angle=300.0, sun_altitude_angle=45.0,
   fog_density=2.0, fog_distance=0.75, fog_falloff=0.1, wetness=0.0,
   scattering_intensity=1.0, mie_scattering_scale=0.03,
   rayleigh_scattering_scale=0.0331
```

I turned all parameters besides *sun_azimuth_angle*, *sun_altitude_angle* to 0. When I turned *sun_altitude_angle* $= -20.0$, the view of RGB camera images appears normal sometimes when the buildings cover the sky, it corresponds to night scenes. But the view isn't normal when the buildings don't cover the sky, even if I set *sun_altitude_angle* $= -60.0$. Conclusion: the problem probably lies on the sky. Similar issues on Github are:

- *github.com/carla-simulator/carla/issues/3417,   .../4438,   .../4527,*

Another approach: change camera settings. Default camera settings are stored in *CamerargbAttributes.txt*. A useful command is

```
camera_bp.set_attribute('enable_postprocess_effects', 'False')
```

After that, both day scenes and night scenes look better. But the sky still has pure white color. Bad news is that all attributes of *sensor.camera.rgb* are the same on both Windows and Ubuntu systems.

Initially, each time when I launch *CarlaUE4.sh*, the following warning appears, and I ran the second line's command. It changes the file $/proc/sys/dev/i915/perf\_stream\_paranoid$ from 1 to 0. I don't know if it has any further unpredicted effects. The *sysctl* command displays and modifies kernel parameters in the directory */proc/sys*.

```
MESA-INTEL: warning: Performance support disabled, consider sysctl dev.
   i915.perf_stream_paranoid=0
sudo sysctl dev.i915.perf_stream_paranoid=0
```

After comparing several fonts and styles of latex environment *lstlistings*, I found the following setting best meets my preference.

$$frame = single, \quad basicstyle = \backslash ttfamily \backslash small, \quad columns = fixed, \quad breaklines = true$$

I also tried to install *Carla-0.9.13* on Windows system. Prerequisite: DirectX runtime. Running dxwebsetup.exe isn't successfull, I installed it by downloading its full setup package. This time there's no bug on sky display. *Python 3.7* is required, note that its correct version is important, and I installed *3.7.9*. Run *manual_control.py* successfully on Windows system. Note that in order to import carla, I wrote two lines of code in *testcarlaimport.py* and copy it every time that I need to import carla. Other requirements are listed as follows:

```
pip install pygame numpy opencv-python==4.2.0.34 open3d matplotlib
```

Good news is that I don't need to fix *Carla*'s bug of sky and RGB camera on Ubuntu system. I can use *Carla-0.9.13* on Windows system instead. An unpredicted advantage is that it runs faster on Windows.

2.2 Manual control in simulator

```
conda activate carla
cd PythonAPI/examples
python manual_control.py
```

Notable required packages in *requirements.txt* are *pygame* and *open*3*d*, neither of the two packages are installed in system python, conda base and openmmlab environments. I can run manual_control.py successfully and change sensors by pressing $1-9$ keys. The sensors corresponding to these keys are listed as follows:

- 1: Camera RGB. Need to fix the bug of over exposure. Strange phenomenon is that the first second's scene after switching to RGB camera is slightly better.
- 2: Camera depth (raw). Displays normally. What is the relation between rendered color and depth?
- 3: Camera depth (grayscale). Displays normally.
- 4: Camera depth (logarithmic grayscale). Displays normally.
- 5: Camera semantic segmentation (raw).
- 6: Camera semantic segmentation (CityScapes Palette). Displays normally.
- 7: Camera instance segmentation (CityScapes Palette). Displays normally.
- 8: Camera instance segmentation (raw). Displays normally.
- 9: Lidar (ray-cast). Displays normally. Don't know what is the exact type of mounted lidar.

Some of other keys act normally but some keys don't act normally, and they are listed as follows:

- Backspace: change vehicle. Acts normally. Drawback is that the response time is approximately 7 seconds after pressing the key, and it's a little bit long.
- M: toggle manual transmission. I have to switch to manual transmission to drive the car, I don't know how to drive in automatic transmission mode.
- ,: gear down; .: gear up. I can switch to gear $1-6$ or R. Pressing W is effective only when the gear is not N. It is weird that the gear number has no upper limit.
- W: throttle. When the gear is R, press W to drive backwards.
- O: open/close all doors of vehicle. Acts normally.
- R: toggle recording images to disk. Acts normally. I tested instance segmentation camera and RGB camera, the output images are .png files in subfolder _out/. Instance segmentation images have good quality, but RGB camera images still have over exposure. Frequency of exporting images is unknown, but the images are named using 8-digits integers, and I guess it's near 3Hz.

I can run manual_control.py in any directory besides PythonAPI/examples. In order to fix the over exposure bug, I decide to change the RGB camera configuration code in a copy of manual_control.py referring to *github.com/carla-simulator/carla/issues/3634*, but it turns out to be a failed attempt. Why?

Good news: while running *manual_control.py*, all keys act normally in Windows system.

2.3 Sensor types in Carla

Following its official account on Youtube, I wrote *CameraSensorsType.py* to explore different kinds of sensors in Carla. They are:

sensor.other.gnss, *sensor.lidar.ray_cast*, *sensor.camera.semantic_segmentation*, *sensor.other.radar*, *sensor.other.lane_invasion*, *sensor.camera.instance_segmentation*, *sensor.camera.rgb*, *sensor.other.rss*, *sensor.camera.optical_flow*, *sensor.lidar.ray_cast_semantic*, *sensor.other.obstacle*, *sensor.other.imu*, *sensor.camera.depth*, *sensor.camera.dvs*, *sensor.other.collision*,

Comments: 1) It has built in lidar semantic sensor, but I don't know how to use it currently.

2) DVS stands for *dynamic vision sensor*, also known as *event camera*, *neuromorphic camera* or *silicon retina*, is an imaging sensor that responds to local changes in brightness. Event cameras don't capture images using a shutter as conventional frame cameras do. Instead, each pixel inside an event camera operates independently and asynchronously, reporting changes in brightness as they occur, and staying silent otherwise.

3) What is an optical flow sensor? It is a vision sensor capable of measuring optical flow or visual motion and outputting a measurement based on optical flow. Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. It can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image.

3 Road marking segmentation dataset preparation

3.1 Exporting front view camera images

Currently the only known example way of exporting images is to press *R* in *manual_control.py*, so I may refer to this code. The second known example is *tutorial.py*. I modified the callback functions of *rgb, semantic, instance, depth* camera sensors in *CameraSensorsType.py*, and successfully exported the above *4* sensors' images.

3.2 Exporting BEV images

One possible way is to rotate the spectator. Reference:

- *medium.com: From semantic segmentation to semantic bird's eye view in the Carla simulator, by Maciek Dziubiński*;
- *Youtube: Semantic bird's view, by Acta Schola Automata Polonica*;
- *NEAT: Neural attention fields for end-to-end autonomous driving, by Andreas Geiger, ICCV 2021; code repository: github.com/autonomousvision/neat*;
- *carla.readthedocs.io/en/0.9.13/python_api/#carlatransform, .../#carla.Rotation*;

I'm mainly interested in the way how BEV semantic segmentation ground truth the is generated. If I can walk through this pipeline, I believe that generating BEV rgb camera images can be done. Key points are two classes: *carla.Transform* and *carla.Rotation*. *carla.Transform* consists of two instance varibles: location (*carla.Location* class) and rotation (*carla.Rotation* class). Notice that *Unreal Engine*'s coordinate system adopted left-handed convention. Correct configuration for BEV camera is *carla.Rotation(pitch=-90)*, unit of Euler angles is degree.

Problem 1 (Reinstall ros-kinetic-desktop-full for Lianzhong)**.**

I backtraced dependency relations through the following commands:

```
sudo apt install ros-kinetic-desktop-full
sudo apt install ros-kinetic-desktop
sudo apt install ros-kinetic-viz
sudo apt install ros-kinetic-rqt-common-plugins
sudo apt install ros-kinetic-rqt-image-view
sudo apt install ros-kinetic-opencv3
sudo apt install libvtk6-qt-dev
sudo apt install libvtk6-dev
sudo apt install libgdal-dev
```

The message of the last two dependencies, and solution to fix this bug are as follows:

```
sudo apt install libgeos-dev
sudo aptitude install ros-kinetic-desktop-full
The following packages have unmet dependencies:
 libgeos-dev : Depends: libgeos-c1v5 (= 3.5.0-1ubuntu2) but 3.5.1-3~
    xenial0 is to be installed
sudo apt install libgeos-c1v5=3.5.0-1ubuntu2
```

Now I know that *aptitude* command works better than *apt* sometimes. In this case, it reports the key dependent package with unmet version through backtracing automatically. Main reference: *https://blog.csdn.net/HelloJinYe/article/details/109104249*.