# Carla0.9.10部署文档

## Carla环境部署与安装

本次部署的版本是编译版本的Carla，这种方式的Carla需要安装UE4，然后进行编译后才能使用，由于UE4的存在，该方式的Carla对电脑的配置要求较高，不过这种方式的Carla可操作性强，可以根据自己的需要创建地图环境，重新编译自己想要的环境和车辆等。除了部署编译版本的carla也对ros-bridge的进行部署。ros-bridge是carla和ros进行数据通信的接口，通过ros-bridge，可以开发出和实车相同的接口方便以后在实车验证算法，

### 安装显卡驱动

在安装之前，首先需要检查显卡驱动是否安装成功，如果安装成功，请确认是否安装了cuda和cudnn。如果都安装成功，可以跳过此步骤。

安装显卡驱动

|  |
| --- |
| # 安装推荐版本的驱动  $ sudo ubuntu-drivers autoinstall  # 也可以手动安装其他驱动  $ sudo apt install nvidia-465  # 查看是否安装成功  $ nvidia-smi |

安装cuda和cudnn

|  |
| --- |
| # 下载cuda的run文件  # cuda安装  $ wget https://developer.download.nvidia.com/compute/cuda/11.3.0/local\_installers/cuda\_11.3.0\_465.19.01\_linux.run  # 下载后运行.sh文件  $ sudo sh cuda\_11.3.0\_465.19.01\_linux.run  # 运行后出现节目，安装的过程中，切忌不要选驱动安装。  # 在~/.bashrc添加环境变量  $ echo "export PATH=/usr/local/cuda/bin:$PATH" >> ~/.bashrc  $ echo "export LD\_LIBRARY\_PATH=/usr/local/cuda/lib64:$LD\_LIBRARY\_PATH " >> ~/.bashrc  # 更新以下，使其生效  $ source ~/.bashrc  # 查看是否安装完成  $ nvcc -V |

|  |
| --- |
| # cudnn安装  # 下载连接：https://developer.nvidia.com/cudnn  $ cd Downloads/ && tar zxvf cudnn-11.3-linux-x64-v8.2.1.32.tgz  $ sudo cp cuda/include/cudnn.h /usr/local/cuda-11.3/include  $ sudo cp cuda/lib64/libcudnn\* /usr/local/cuda-11.3/lib64  $ sudo chmod a+r /usr/local/cuda-11.3/include/cudnn.h  $ sudo chmod a+r /usr/local/cuda-11.3/lib64/libcudnn\* |

### 安装conda环境

推荐安装miniconda也可以安装anaconda，二选一即可

|  |
| --- |
| # 下载连接:https://docs.conda.io/en/latest/miniconda.html  $ bash Miniconda3-latest-Linux-x86\_64.sh |

### Carla编译版本的部署

Carla编译版本的安装，除了需要安装显卡驱动外。安装UE4之前，首先确定carla版本和UE4的版本对应。本文安装的版本为carla0.9.10，对应的UE4版本为4.24，确定版本之后，开始安装。首先要安装相关环境和依赖，安装完成后开始安装UE4.24，之后安装carla进行编译运行。参考官方连接：

<https://carla.readthedocs.io/en/0.9.10/build_linux/>

#### 安装UE4.24

首先根据官方给的资料安装相关环境，官方推荐的是18.04系统，按照官方文档提供的安装方法有些问题，本文在Ubuntu18.04和20.04均成功实现了安装，分别在两台科研机上面都进行了部署。下面是Ubuntu20.04系统的配置方法

|  |
| --- |
| # 安装vulkan-sdk  $ wget -qO - http://packages.lunarg.com/lunarg-signing-key-pub.asc | sudo apt-key add -  $ sudo wget -qO /etc/apt/sources.list.d/lunarg-vulkan-bionic.list http://packages.lunarg.com/vulkan/lunarg-vulkan-bionic.list  $ sudo apt update  $ sudo apt install vulkan-sdk  # 配置端口  $ iptables -I INPUT -p tcp --dport 2000 -j ACCEPT  $ iptables -I INPUT -p tcp --dport 2001 -j ACCEPT  $ iptables-save  # 安装依赖  $ sudo apt-get update  $ sudo apt-get install wget software-properties-common  $ sudo add-apt-repository ppa:ubuntu-toolchain-r/test  $ wget -O - https://apt.llvm.org/llvm-snapshot.gpg.key|sudo apt-key add -  $ sudo apt-add-repository "deb http://apt.llvm.org/$(lsb\_release -c --short)/ llvm-toolchain-$(lsb\_release -c --short)-8 main"  # 安装相关依赖  $ sudo apt-get update  $ sudo apt-get install build-essential clang-8 lld-8 g++-7 cmake ninja-build libvulkan1 python3-dev python3-pip libpng-dev libtiff5-dev libjpeg-dev tzdata sed curl unzip autoconf libtool rsync libxml2-dev  # 配置clang版本  $ sudo update-alternatives --install /usr/bin/clang++ clang++ /usr/lib/llvm-8/bin/clang++ 180  $ sudo update-alternatives --install /usr/bin/clang clang /usr/lib/llvm-8/bin/clang 180 |

配置完成后即可开始下载相关依赖，切忌不可以download之后离线安装，一定要克隆下来安装。由于网络不稳定，服务器是国外的，克隆之前配置一下git

|  |
| --- |
| # 增加缓存  $ git config --global http.postBuffer 524288000  # 修改下载速度  $ git config --global http.lowSpeedLimit 0  $ git config --global http.lowSpeedTime 999999  # 修改完成后就可以开始下载了，期间一定要保证网络连接正常  $ git clone --depth=1 -b 4.24 https://github.com/EpicGames/UnrealEngine.git ~/UnrealEngine\_4.24  $ cd ~/UnrealEngine\_4.24  # 下载个文本格式的文件  $ wget https://carla-releases.s3.eu-west-3.amazonaws.com/Linux/UE\_Patch/430667-13636743-patch.txt ~/430667-13636743-patch.txt  patch --strip=4 < ~/430667-13636743-patch.txt  # 编译，编译过程比较缓慢，要保证网络稳定  $ ./Setup.sh && ./GenerateProjectFiles.sh && make  # 打开UE4.24，如果可以成功开启，证明安装成功。  $ cd ~/UnrealEngine\_4.24/Engine/Binaries/Linux && ./UE4Editor  # 系统环境添加变量  $ sudo gedit ~/.bashrc  # export UE4\_ROOT=~/UnrealEngine\_4.24  # 刷新生效  $ source ~/.bashrc |

#### 安装carla

安装之前确认是否安装python等环境，对于Ubuntu20.04已经不再支持python2故，安装依赖的时候不会安装python2相关依赖。如果已安装则通过克隆的方法进行carla下载，下载完成后通过命令切换到0.9.10版本的分支。

|  |
| --- |
| $ git clone https://github.com/carla-simulator/carla  $ cd carla  # 切换到0.9.10版本  $ git checkout -f c7b2076  # 获取相关资源，获取相关资源之前更改一下源，替换掉亚马逊的源  # 更改update.sh文件中的50行。CONTENT\_LINK=http://mirrors.sustech.edu.cn/carla/carla\_content/${CONTENT\_ID}.tar.gz  # 运行更新，获取相关资源  $ ./Update  # 编译，编译过程中出现的boost下载问题，可以通过下载后，放在carla文件夹下面的build文件夹中，重新开始make launch，其他的git问题，换个好点的网络，如果失败就重新输入命令进行安装，直至安装成功。  $ make launch  # 编译PythonAPI，通过ARGS命令确定python版本，3.8是Ubuntu20.04版本支持的python版本  $ make PythonAPI ARGS="--python-version=3.8"  # 生成执行文件  $ make package  # 运行方法，通过下面命令开启carla  $ make launch-only |

### 安装ros

安装ros-bridge之前需要安装ros，ros的安装较为简单，只要网络没有问题，就可以根据官网成功实现安装。Ubuntu20.04系统的ros是python3支持的名称为noetic。

|  |
| --- |
| # Ubuntu20.04安装ros参考连接  http://wiki.ros.org/cn/noetic/Installation/Ubuntu  # 设置sources.list  $ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'  # 设置密钥  $ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654  # 安装，ros桌面完整版  $ sudo apt update  $ sudo apt install ros-noetic-desktop-full# 初始化rosdep  $ sudo rosdep init  $ rosdep update  # 设置环境  $ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc  $ source ~/.bashrc  # 也可以使用小甲鱼写的工具，推荐使用，比较方便，根据命令提示操作即可  $ sudo apt-get install curl && curl http://fishros.com/tools/install/ros-noetic | bash  $ wget http://fishros.com/install -O fishros && bash fishros |

### 安装ros-bridge

|  |
| --- |
| # 进入carla文件夹  $ cd ~/carla  # 创建carla-ros-bridge文件夹  $ mkdir -p carla-ros-bridge/catkin\_ws/src  $ cd carla-ros-bridge/catkin\_ws/src  # 下载ros\_bridge  $ git clone --recurse-submodules https://github.com/carla-simulator/ros-bridge.git  # 查看当前分支，切换分支  $ git checkout -f 0700f2d  # 更新rosdep安装编译环境  $ sudo rosdep init  $ rosdep update  # 进入catkin\_ws文件夹下  $ rosdep install --from-path src --ignore-src -r  # 进入carla\_ros\_scenario\_runner文件夹下，将其CMakeLists.txt文件的catkin\_install\_python注释以下，不然就会报无法导入ApplicationStatus的问题，这是由于ros编译过程中在devel/lib文件夹下生成了carla\_ros\_scenario\_runner不可用的包。  # 开始编译  $ catkin\_make  # 注意：编译报-- Could NOT find PY\_em (missing: PY\_EM)错的话，就退出conda环境 |

### 安装scenario\_runner

|  |
| --- |
| # 进入carla文件夹  $ cd ~/carla  # 下载scenario\_runner  $ git clone --single-branch https://github.com/carla-simulator/scenario\_runner.git  $ git checkout -f 0ff0ebe |

## 基于ROS接口的深度强化学习

强化学习通过ros-bridge和Carla仿真环境进行通信，中间的ros层主要是用来往环境中放置车辆，订阅算法发布出来的控制命令，同时将车辆的状态实时发送给算法。通过scenario\_runner模块可以规划输出车辆的轨迹，通过计算车辆的相对位置来进行全局路径规划。同时根据轨迹进行环境的搭建，搭建环岛，AEB，车道保持，换道，ACC场景。

carla

ros\_bridge

强化学习

实车环境

ros对接

图 1 强化学习通过ros-bridge和仿真环境连接

### 软件组成

软件部分程序写了两个版本，一个是carla\_gym\_xxx程序文件，后面的xxx是对应场景的名字，该版本需要启动2-3个程序文件，这些程序文件都是运行强化学习所需要的，包括front\_vehicle.py用来和carla-ros-bridge进行数据通信，发布下一个节点的位置。TS\_ppo.py是强化学习对应的代码文件；另外一个是carla\_gym\_v2程序文件，里面有个lane\_keep.py文件，该文件更改起点和终点就可以进入对应的环境，TS\_ppo.py会调用lane\_keep.py文件，不需要再单独启动，v2版本的写法更接近于carla0.9.6版本的强化学习文件。

### 环境配置

|  |
| --- |
| # ros环境配置  # 创建环境  $ conda create -n carla\_ros python=3.8  # 激活环境  $ conda activate carla\_ros  # 进入ros\_bridge文件下安装相关包  $ cd ~/carla/carla-ros-bridge/catkin\_ws/src/ros-bridge  $ pip install -r requirements.txt  # 进入scenario\_runner文件夹中  $ cd ~/carla/scenario\_runner  $ pip install -r requirements.txt  # 安装好上述包之后安装下面几个包  $ pip install pyyaml  $ pip install rospkg  $ pip install pycryptodomex  $ pip install gnupg  # 强化学习环境配置  # 创建环境  $ conda create -n ts\_rl python=3.8  # 激活环境  $ conda activate ts\_rl  # 安装好上述包之后安装下面几个包  $ pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch\_stable.html  $ pip install tianshou  $ pip install pyyaml  $ pip install rospkg |

### 功能模块设计

carla\_gym\_xxx

├── build

├── devel

├── log

├── QiDong.md

└── src

其中build和devel是编译生成的文件，log里面是训练的权重文件，src里面是程序文件。src

├── carla\_msgs

│   ├── CMakeLists.txt

│   ├── LICENSE

│   ├── msg

│   ├── package.xml

│   └── README.md

├── carla\_waypoint\_types

│   ├── CMakeLists.txt

│   ├── msg

│   ├── package.xml

│   └── srv

├── CMakeLists.txt -> /opt/ros/noetic/share/catkin/cmake/toplevel.cmake

└── gym\_carla

├── CMakeLists.txt

├── include

├── launch

├── package.xml

└── src

其中gym\_carla是和强化学习有关的文件，src文件中是主要编写的代码文件，文件内容如下，注意是gym\_carla里面的src。

src

├── carla\_ppo.py

├── envs

│   ├── acc\_ros\_carla\_env.py

│   ├── followed\_vehicle.py

│   ├── front\_vehicle.py

│   ├── gym\_data\_trans.py

│   ├── \_\_init\_\_.py

│   └── \_\_pycache\_\_

├── \_\_init\_\_.py

├── main.py

├── ppo\_acc.py

├── sac\_acc.py

└── TS\_ppo.py

其中，envs是对应的环境文件，该文件里面有场景相关的程序，例如上面的front\_vehicle.py就是对应的acc前车的控制文件，TS\_ppo.py是强化学习文件。

上面的程序文件例子是v1版本的，v2版本和v1版本的差距不大，只不过是v2版本的启动TS\_ppo.py文件后，对应的其他数据文件也同时启动。

### 案例整体方案

深度强化学习搭建的场景包括环岛、AEB、车道保持、ACC和换道五个场景。为了满足环岛场景下的车辆控制，本文选择Carla的Town03地图进行场景搭建，该地图下面具有环岛、上下坡、丁字路口和长直线等城市场景，方便选取合适的路径进行场景搭建。通过carla-ros-bridge进行数据连接，启动相关的roslaunch文件后会放置一辆车，同时会发布出来车辆的轨迹。本文根据车辆的轨迹坐标和车辆的相对位置和carla-ros-bridge进行数据通信完成场景搭建。首先设置车辆的起点和终点位置，carla-ros-bridge的轨迹规划模块会根据车辆的起点和终点发布出来一条轨迹。本文通过获得车辆与轨迹之间的相对位置作为车辆的状态，发布车辆的控制命令，实现车辆的车道保持场景的搭建。

### 基本功能要求

1. 车辆的环岛场景的车道保持

通过设置车辆在环岛附近的初始位置，规划路线，实现环岛通行。

1. 遇到前方车辆进行紧急制动

初始车辆固定位置，前方放置障碍物车辆，判断前方是否有车辆，如果有车辆，则进行紧急制动，否则车辆继续行驶。

1. 车辆保持在固定的车道行驶

随机初始车辆位置，实现车辆在一段路程中保持中间车道行驶，不偏离车道。

1. 跟随前方车辆行驶

在固定的一段路径下，车辆前方放置被跟踪车辆，被跟踪车辆以恒定的速度向前行驶，被控车辆以恒定距离跟踪前方车辆

### 整体设计

**环岛场景**

在地图中选取环岛的起点{105.5,3.4,0}，前两个数据是车辆相对于Carla地图坐标系的x,y的位置，后面数据是车辆的偏航角。设置车辆在环岛的终点位置{-6.6, -59.5, 0}。初始位置放置车辆后，通过carla-ros-bridge规划一条通向终点的轨迹，然后根据车辆当前位置计算车辆相对于轨迹的状态数据。强化学习算法根据车辆距离轨迹的位置，车辆的偏航角度，车辆的速度，车辆前方相同道路是否有其他车辆作为算法的状态输入，根据当前状态输出控制命令，控制车辆在环岛场景下行驶。

**AEB场景**

在地图中选取长度为80米左右的路段，设置车辆的起点为{41.5,-7.3,0}，前两个数据是车辆相对于Carla地图坐标系的x,y的位置，后面是车辆的偏航角，不设置车辆的终点。AEB场景下除了放置被控车辆外，还需放置障碍车辆，障碍车辆放置的位置为{150,-8,0}，设置障碍车辆的速度为0，刹车为1。初始位置放置车辆后，通过carla-ros-bridge规划一条直行的轨迹，然后根据车辆当前位置计算车辆相对于轨迹的状态数据。强化学习算法根据车辆距离轨迹的位置，车辆的偏航角度，车辆的速度，车辆前方相同道路是否有其他车辆作为算法的输入，输出控制命令，当前方遇到障碍车辆时，进行紧急停车。

**车道保持场景**

在地图中随机选取可放置车辆的地点放置车辆位置为{-43.5,-207.6,0}，设置车辆的终点位置为{246.9,-50.2,0}。根据车辆当前位置规划车辆的轨迹，并根据车辆当前位置计算车辆相对于轨迹的状态数据。强化学习算法根据车辆距离轨迹的位置，车辆的偏航角度，车辆的速度，车辆前方相同道路是否有其他车辆作为算法的输入，通过算法输出控制命令，控制车辆进行车道保持。

**跟车场景**

在地图中选取一段长度为180米的路段，设置车辆的起点为{-50.5,-207.3,0}，前两个数据是车辆相对于Carla地图坐标系的x,y的位置，后面是车辆的偏航角，设置车辆的终点位置{180,-204,0}。初始位置放置车辆后，由carla-ros-bridge规划出来一条轨迹，然后根据车辆当前位置计算车辆相对于轨迹的状态数据。除了被控车辆外，还需要放置被跟随车辆，被跟随车辆放置在被控车辆前方15米左右位置，放置的位置为{30,207.2,0}，给予被跟随车辆一定的速度。强化算法根据车辆距离轨迹的位置，车辆的偏航角度，车辆的速度，车辆与前车的距离和偏角作为状态输入，输出控制命令，控制车辆以固定的距离跟随前车。

**换道场景**

在地图中选取长度为60米左右的路段，设置车辆的起点为{35.5,-7.3,0}，前两个数据是车辆相对于Carla地图坐标系的x,y的位置，后面是车辆的偏航角，设置车辆的终点{150.6,-5.0,0}。除了被控车辆外，还需要放置障碍车辆，障碍车辆放置的位置为{55.42,7.6,0}，设置车辆的速度为0，刹车为1。初始位置放置车辆后，由carla-ros-bridge模块规划出来一条轨迹，强化算法根据车辆距离轨迹的位置，车辆的偏航角度，车辆的速度，车辆前方相同道路是否有其他车辆作为算法的输入，输出控制命令，进行换道操作。

### 操作手册

1. 启动Carla仿真系统

进入/CARLA\_0.9.10文件夹下，右击鼠标新建终端，输入如下命令

|  |
| --- |
| # 启动无界面显示的carla仿真环境  $ DISPLAY= ./CarlaUE4.sh -opengl -carla-port=2000 |

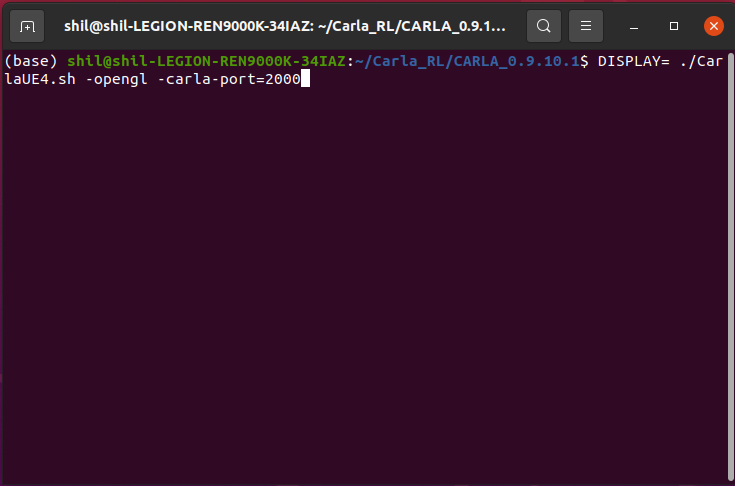


图 2 启动carla仿真环境

1. 启动carla-ros-bridge相关环境

进入～/carla/carla-ros-bridge/catkin\_ws/文件夹下，点击鼠标右键新建终端，输入以下命令

|  |
| --- |
| # 激活conda环境  $ conda activate carla\_ros  # 设置环境  $ export PYTHONPATH=$PYTHONPATH:/home/shil/carla/PythonAPI/carla/dist/carla-0.9.10-py3.8-linux-x86\_64.egg:/home/shil/carla/PythonAPI/carla  $ export SCENARIO\_RUNNER\_PATH=/home/shil/carla/scenario\_runner  $ source devel/setup.bash  $ roslaunch carla\_ad\_demo carla\_ad\_demo\_with\_scenario.launch |

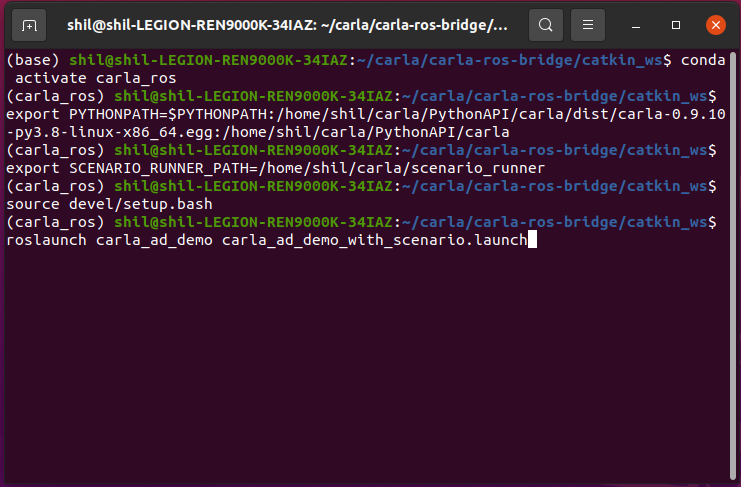


图 3 启动carla-ros-bridge

1. 启动强化学习环境

进入～/Carla\_RL/carla\_gym\_acc文件夹下，点击鼠标右键新建终端，输入以下命令

|  |
| --- |
| # 开启终端，进入Carla\_RL/carla\_gym\_xxx文件夹，发布路径信息  $ conda activate carla\_ros  # 刷新环境  $ source devel/setup.bash  # 启动frot\_vehicle.py  $ python src/gym\_carla/src/envs/front\_vehicle.py  其中aeb和acc和lanechange需要障碍车辆，可以启动对应的envs文件夹下面的xxx\_vehicle.py文件  # 开启终端，进入Carla\_RL/carla\_gym\_acc文件夹，放置障碍物车辆  $ conda activate carla\_ros  $ export PYTHONPATH=$PYTHONPATH:/home/shil/carla/PythonAPI/carla/dist/carla-0.9.10-py3.8-linux-x86\_64.egg:/home/shil/carla/PythonAPI/carla  $ python src/gym\_carla/src/envs/followed\_vehicle.py  # 开启终端，进入Carla\_RL/carla\_gym\_xxx文件夹，开始强化学习  $ conda activate ts\_rl  # 刷新环境  $ source devel/setup.bash  # 启动强化学习代码  $ python src/gym\_carla/src/TS\_ppo.py |

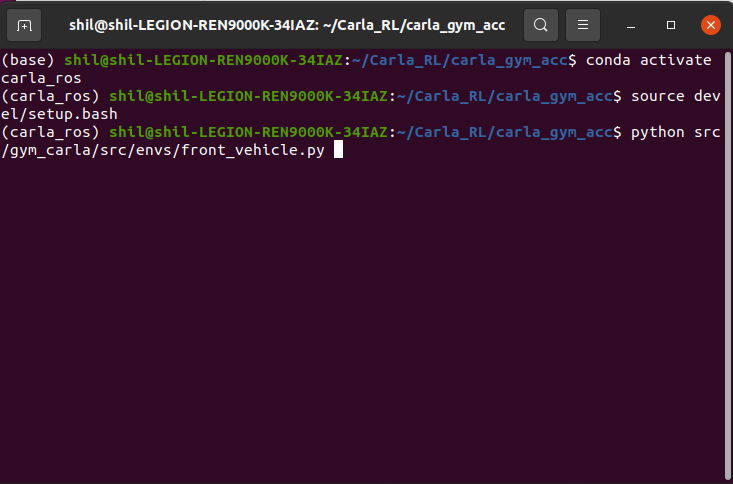


图 4 启动front\_vehicle

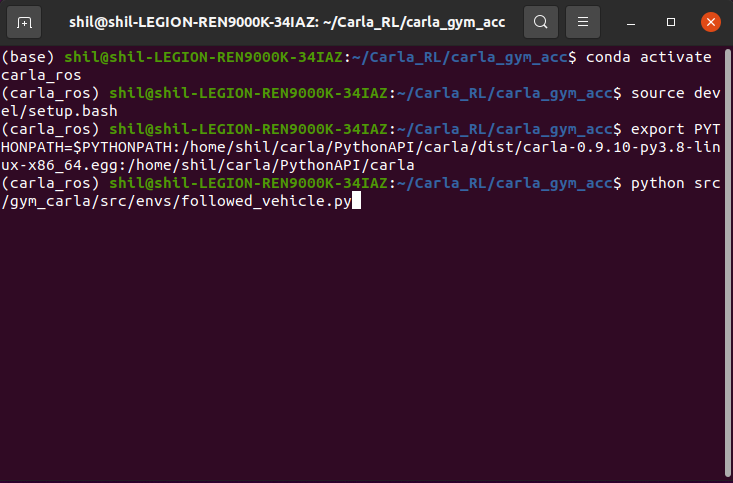


图 5 启动障碍车辆

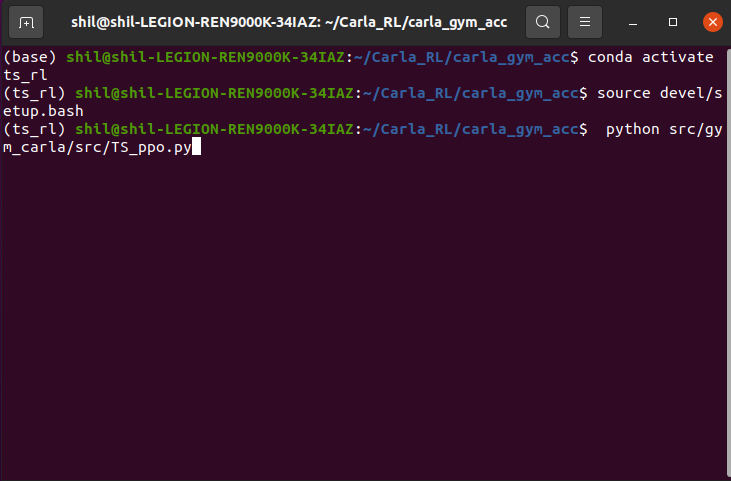


图 6 启动强化学习

### 案例实现结果展示（由于效果较差，未能较好完成，暂时没有能展示的结果）

#### 环岛

环岛场景下的主要难点是轨迹曲率较大，车辆横向控制的过程中需要不停的控制方向盘完成转弯动作。根据车辆的状态设计了车辆超出车道惩罚，车辆速度奖励，车辆横向加速惩罚等奖励项，完成了车辆在环岛场景下的驾驶。下面两张图是车辆控制过程中的截图，车辆在环岛转弯过程中摆动较大。分析原因是算法的奖励项设计的还不够完善，优化后在进行训练。

#### AEB

AEB场景下的难点为车辆的纵向控制，车辆在判断出现障碍物后，在合理的范围内进行刹车操作。根据车辆的状态设计了车辆超出车道惩罚，车辆速度奖励，车辆横向加速惩罚，碰撞惩罚等奖励项，完成了车辆遇见障碍物紧急停车。下面两张图是车辆控制过程中的截图。

#### 车道保持

车道保持场景下的难点为车辆的横纵向控制，车道保持在直线路段下行驶的较好，在十字路口和丁字路口这种曲率较大的左转弯和右转弯时车辆控制效果不是很理想，特别是右转弯，车辆很容易偏出车道。根据车辆的状态设计了车辆超出车道惩罚，车辆速度奖励，车辆横向加速惩罚，碰撞惩罚等奖励项，完成了车辆车道保持的功能。下面两张图是车辆控制过程中的截图。

#### 跟车

车道跟车场景下的难点为车辆的纵向控制，前方车辆的速度未知，保持固定的距离进行跟车，这个过程中很容易出现速度过快或者过慢的问题。根据车辆的状态设计了车辆超出车道惩罚，车辆速度奖励，车辆横向加速惩罚，碰撞惩罚，以及两车的距离奖励等奖励项，完成了车辆的跟车功能。下面两张图是车辆控制过程中的截图。

#### 换道

车道保持场景下的难点为车辆的横纵向控制，特别是在换道的过程中，车辆很容易偏出车道。根据车辆的状态设计了车辆超出车道惩罚，车辆速度奖励，车辆横向加速惩罚，碰撞惩罚等奖励项，完成了车辆跟车的功能。下面两张图是车辆控制过程中的截图。