Instruction for Preliminary Project 2

RC car keyboard control

1. Introduction

In the following projects, we will control an RC car in GAZEBO. You need to understand how keyboard inputs are passed through ROS communication and control the robot's movements. In this preliminary project 2, we spawn an RC car on GAZEBO and implement a keyboard control. ROS communication is used such as publisher and subscriber. It generates keyboard input topics and the RC car is controlled by sending linear velocity and steering angle which are calculated by keyboard input.

2. GAZEBO simulator

You already have a GAZEBO simulator. Since GAZEBO is installed when you install ROS melodic, you can easily start the program without external installation. This section explains the way to use a gazebo and communicate with a gazebo world.

2.1. Using roslaunch to Open World Models

The roslaunch tool is the standard method for starting ROS nodes and bringing up robots in ROS. To start an empty Gazebo world like the rosrun command in the previous tutorial, simply run this.

```
roslaunch gazebo_ros empty_world.launch
```

You can see the gazebo world like figure 1.

In your project file, there is a launch file to open a world model which contains the RC car.

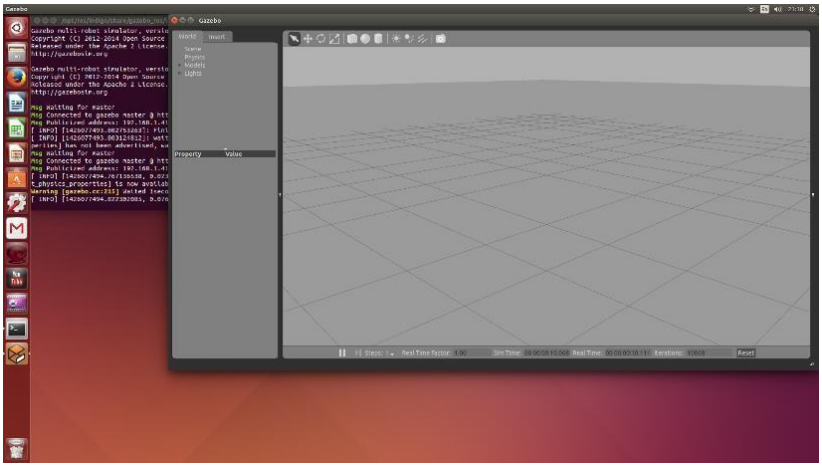

Figure 1. GAZEBO Empty world

3. Assignment

This assignment is making nodes to control RC cars by a keyboard. This will be helpful for your future project assignment. You have to download 'preproject2.zip' file and extract it at ~/catkin_ws/src/. This file contains an RC car model, and the model you will use is a fixed version of the MIT RC car model (https://mit-racecar.github.io/).

3.1. Install packages

You must install some packages to simulate RC cars in the GAZEBO world. You can download and install them by following commands.

```
sudo apt-get install ros-melodic-effort-controllers
sudo apt-get install ros-melodic-ackermann-msgs
sudo apt-get install ros-melodic-controller-manager
sudo apt-get install ros-melodic-joy
sudo apt-get install ros-melodic-ros-control
sudo apt-get install ros-melodic-ros-controllers
sudo apt-get install ros-melodic-gazebo-ros ros-melodic-gazebo-ros-pkgs
sudo apt-get install ros-melodic-gazebo-ros-control
```

Also, we need to install pip packages by following commands.

```
sudo apt-get install python-pip
pip install gym
pip install pynput==1.6.0
```

Some of these commands are needed to control the RC car model by a keyboard. After installation, compile your project.

```
cd /catkin_ws && catkin_make
```

3.2. ~/.bashrc Settings

Copy and paste following commands at the end line of ~/.bashrc. (NOT on terminal)
You can use vi, gedit, or echo.

```
source ~/catkin_ws/devel/setup.bash
export PYTHONPATH="/home/${USERNAME}/catkin_ws/src/IS_PROJECT/sim2real/scripts":${PYTHONPATH}
```

3.3. GAZEBO world

You can launch the GAZEBO world using the following command.

```
roslaunch sim2real base.launch
```

You will see a RC car and its sensing range. Check which node is running by using the rosnode command. After that, check which topic is made by the node using rostopic command.

```
rosnode list
rostopic list
```

4. Skeleton Code

- keyboard_controller.py, keyboard_signal.py, rccar_env.py in {...}/sim2real/scripts/project directory.
- You need to implement 'TODO' parts in provided codes.

4.1. keyboard_signal.py

w	Move straight forward	**a**	Turn left
s	Stop	**d**	Turn right
x	Slow Down	**r**	Reset
e	Terminate		

This file receives the keyboard inputs and sends the input controls through a topic named 'key_op'. Students must implement a code to publish appropriate inputs to self.pub. This control input will be sent to keyboard_controller.py through this topic. The inputs must control the race car in the manner as described in the table above. The keyboard inputs are received through the functions on_press(self, key), and on_release(self,key). You can execute this code by the following command.

```
rosrun sim2real keyboard_signal.py
```

4.2. keyboard_controller.py

The control inputs subscribed from 'key_op' are given to the race car environment which is then published to the race car. The subscribed data is received and processed through callback(self,msg) function. Once the msg is received, students must set self.omega and self.velocity values which are sent to the environment automatically. You can execute this code by the following command.

```
rosrun sim2real keyboard_controller.py
```

4.3 rccar_env.py

reset() is used to set the race car on the origin and step(angular velocity, linear velocity) function is used to publish the control inputs to the race car.

5. Summary

To summarize, what you should do is to complete TODO parts in keyboard_signal.py and keyboard_controller.py and execute codes below in separate terminals.

```
roslaunch sim2real base.launch
```

```
rosrun sim2real keyboard_signal.py
```

```
rosrun sim2real keyboard_controller.py
```

Make sure to activate the virtual environment where you have installed the required python packages in all terminal windows if you use any kind of virtual environment (venv, virtualenv, conda, etc.).

6. Submission Format

Compress your catkin_ws folder which includes all your project files. Then, upload it on ETL. The name of the compressed file should be "**IS_[Student ID]_Pre_Project2.tar.gz**". For example, if your ID number is 2023-12345, the file name should be 'IS_2023-12345_Pre_Project2.tar.gz'.

Due to: 2023.10.09 23:59 KST