Simulating Robots

1. Importance of Simulation

Simulation is an essential part of Robotics Automation. They provide a versatile platform to quickly test your code for bugs and performance and try out new ideas. While robots are becoming more accessible all the time, it is still not at the stage where you can test your code directly on them. In fact, running code in simulations will always be more convenient. For a more thorough introduction on simulators and the robots you can find in them, go through **Chapter 6** of Morgan Quigley.

2. Gazebo

Gazebo is the most popular simulator for robotics development. It can simulate robots in a 3D environment and can be fully integrated into ROS integrated with Gazebo using the `gazebo_ros` ROS package. You can interface your robots in the simulation using ROS and control them using ROS messages and services.

2.1 Installation

Gazebo is automatically installed when you install ROS. To make sure you have all the ROS packages necessary for running Gazebo simulations are installed

```
sudo apt-get install ros-melodic-gazebo-*
```

2.1 Getting Started

You can launch the Gazebo GUI simulator window by just running the command `gazebo` in the terminal. You can follow this tutorial to get used to the gazebo GUI and make some models.

2.1 Integration with ROS

Gazebo has tight integration with ROS to make simulation as convenient as possible. For example you can publish a velocity to the cmd_vel topic and this can cause a robot in Gazebo to move.

3. Turtlebot

Turtlebot is a differential drive robot which is completely open source. This means everything from its design to its code is available online. You can read more about turtlebot here. The latest version of turtlebot is Turtlebot 3 and comes in 3 variants : Turtlebot3 Burger, Turtlebot3 Waffle and Turtlebot3 Waffle Pi.

While Turtlebot is a real life robot, it has been modeled to be used in simulation as well. In fact, it has become the standard platform used to simulate ground robots with ROS due to its simplicity and ease of use. After spawning it in gazebo, all you need to do to make it move is publish to the `cmd_vel` topic through ROS.

3.1 Installation

Use the command :

```
sudo apt install ros-melodic-turtlebot3-*
```

to install the required packages. After installation, make sure to add the following to your `~/.bashrc` file -

```
export TURTLEBOT3_MODEL=burger
```

This defines which model of turtlebot to use for simulation. You can use `waffle` or `waffle_pi` instead of `burger`.

A brief description of some these packages which we'll be using in this course is given below:

- turtlebot3_bringup : Used to launch actual turtlebot. Not used in simulations.
- turtlebot3_gazebo : Contains all the files necessary for simulating the turtlebot in Gazebo.
- turtlebot3_msgs : Contains all the message and service definitions used by the turtlebot
- turtlebot3_description : Provides complete 3D model of turtlebot, different environments (worlds) for simulation and visualisation. You can see all the urdf files in the urdf folder of this Package

- turtlebot3_teleop : Provides launch files and nodes for teleoperation of turtlebot using different input devices (like keyboard, joysticks and even video game controllers!)

- turtlebot3_navigation : Contains the roslaunch scripts for starting autonomous navigation

- turtlebot3_slam : Contains the roslaunch scripts for starting SLAM (Simultaneous Localization and Mapping) in turtlebot.

3.2 Controlling Turtlebot inside Gazebo

The best way to see turtlebot in action is to use teleop - a ROS node which lets you publish to cmd_vel using your keyboard. For this you will need to run the following commands in separate terminal windows -

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

The first command launches gazebo and spawns Turtlebot3 in a field of obstacles. The second runs the teleop node which lets you drive turtlebot using arrow keys. Check out this link for more.

Let's examine the ROS topics initialised by these commands. Open another terminal window and run

```
rostopic list
```

This will list out all of the active topics. Among them you should see `cmd_vel`. To see more information about it, run

```
rostopic info cmd_vel
```

You should see the message type for the topic being `Twist` as well as who is publishing and subscribing to the topic. To see the message being published, you can run -

```
rostopic echo cmd_vel
```

This will print out the message continuously into the terminal. To get more info on the message type `Twist` you can run -

```
rosmsg info Twist
```

This will tell you that `Twist` is part of `geometry_msgs` (an inbuilt set of message types you can learn about here) as well as its structure.

All four of the above commands come in very handy when getting to know a new system of ROS nodes or even debugging your own, so do get familiarized with them. You can learn more about them in the last section of Chapter 20 in Morgan Quigley. While we examined the `cmd_vel` topic, you can go ahead and try it on any of the topics you see with `rostopic list`

From the above exploration we saw that the `turtlebot3_teleop_keyboard` node publishes a message of type `Twist` on the topic `cmd_vel`. This topic is subscribed to by the `gazebo` node which moves the turtlebot inside the simulator accordingly. If you are unsure about any of the terms we just used, you should go over the ROS basics document again.

The two main topics you will be using for interacting with turtle bot are `cmd_vel` and `odom`

3.2.2 cmd_vel

All you have to do to make the turtlebot move in the simulator is write a node that publishes appropriate `Twist` messages on `cmd_vel`. For a tutorial, check out this video (Note that they use an online ROS development environment but you can do the same thing on any platform)

3.2.2 odom

To get the location of turtlebot in the environment you will be using `odom`. Again, to see more information about the topic, run -

```
rostopic info /odom
```

You will see that `odom` carries a message of type `Odometry` which is part of the `nav_msgs` package. Like `geometry_msgs` this is also a set of messages. You can learn more about it here. You can explore Odometry using `rosmsgs info`