ROS2 Sensors

Learning Outcomes:

This exercise will help students to get familiar with the Gazebo physics simulation environment, how to simulate different sensors in Gazebo and learn their working principles, integrate Gazebo simulations with the ROS interface, see the effects of sensor parameters on the outputs and learn to utilize visualization tools in ROS.

Grading

- Grade 3: successful demos + could not answer all questions
- Grade 4: successful demos + could answer main questions
- Grade 5: successful demos (including complete task 5) + could answer main questions

Demonstration instructions

Section	Demonstration Notes
1	Nothing to demo. Satisfactory answers to all questions
2	Successful demo of task 2.2. Satisfactory answers to all questions
3	Nothing to demo. Satisfactory answers to all the questions
4	Demo of tasks. Satisfactory answers to all the questions
5	Demo of tasks

1. Quick introduction to Gazebo Simulation Environment

Robotic simulation environments enable interacting with sensors, robots and mechatronic systems in a virtual environment. This is useful for testing, and for faster deployment of robotic systems with minimal errors. Gazebo is a physics simulator first introduced as Gazebo-classic which reached its end of life with version 11. The new Gazebo Sim has more improved features over Gazebo-classic.

There are multiple releases of Gazebo Sim (also known as Ignition Gazebo). It is important to know which version you are working with because of different release features and certain ROS packages require a specific version of Ignition Gazebo to function as intended. This information is usually mentioned in the readme section of the git repository of associated ROS packages.

For this tutorial we are using Ignition Gazebo version Fortress (6.11.0). There are multiple ways to install the package in an Ubuntu environment. Two of the main installation methods are Binary installation and Installation from source. Binary installation is a quick and ready-to-use method, while Installation from source downloads the source code and builds the package in your PC. Installation from source is recommended when developing a package on top of existing developments. Although you do not need to know much in depth on the above methods, it is recommended to check the documentation for different version combinations, especially if not using Ubuntu 22.04 Jammy.

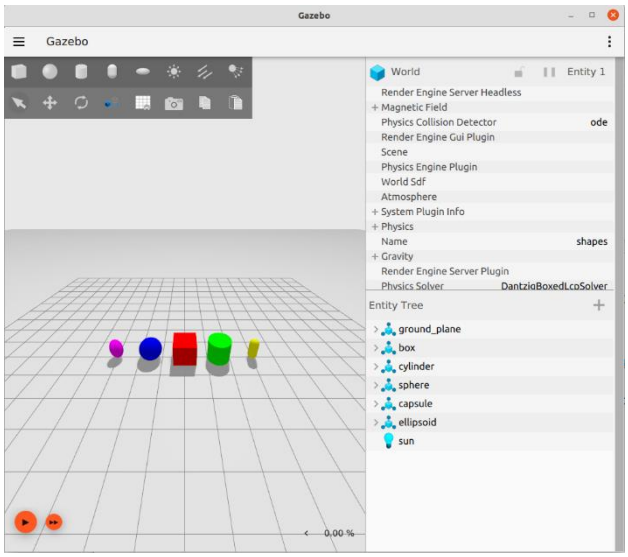
1.1 Installing Gazebo Simulation Environment

Follow the instructions from the link below and configure the Gazebo Fortress environment on your PC.

https://gazebosim.org/docs/fortress/install_ubuntu

Ignition gazebo GUI should open when you run the command below in your Ubuntu terminal if you have configured everything and installation is successful.

user:~$ ign gazebo shapes.sdf



1.2 Get familiar with Ignition Gazebo simulator

Implement instructions from each of the tutorials below. You do not have to implement anything on your own. But try out every guided step and understand all the fundamental concepts described in each section.

- Understanding the Graphical User interface (GUI). GUI functionalities are crucial to interact with the Gazebo interface. https://gazebosim.org/docs/fortress/gui

- Manipulating models: Interact with the models. https://gazebosim.org/docs/fortress/manipulating_models

- Model insertion from Fuel: (The online model database for Ignition Gazebo). https://gazebosim.org/docs/fortress/fuel_insert

- Building robot – Understanding how to create a Gazebo world using SDF format and including a custom-built robot. https://gazebosim.org/docs/fortress/building_robot
  For more information on SDF format follow the links below.
  https://gazebosim.org/docs/fortress/sdf_worlds | http://sdformat.org/

- Moving the robot: Introduction to plugins, topics and messages. https://gazebosim.org/docs/fortress/moving_robot

2. Simulating Sensors

2.1 Get Familiar with sensors

Ignition Gazebo can simulate the functionality of different sensors with the help of plugins. These plugins are developed by manufacturers/researchers and are available in the default installation itself. Try the example of simulating different types of sensors.
https://gazebosim.org/docs/fortress/sensors

Question: What are the different attributes of the Lidar sensor and what difference do they make when set to different values?

2.2 Segmentation camera example

Now you have all the fundamental knowledge to simulate sensors in Gazebo. Let's simulate a segmentation camera and generate a dataset. The example utilizes existing plugins, and you only have to include them in your Gazebo world. Do not worry If you are unfamiliar with the terms instance segmentation and semantic segmentation. You will understand the two terms from the output when you run the world.
https://staging.gazebosim.org/api/sensors/7.0/segmentationcamera_igngazebo.html

Follow the above example and replicate the following tasks:

1. Either build your own world file from scratch or use the given segmentation_scene.sdf file.
2. Set up the segmentation camera model and plugin. You may choose any model file to represent the camera.
3. Be creative and include several model files to be included in your dataset and label them.
4. Run the example and save data
5. Visualize with Python script (optional)

Note: The last step on Processing the segmentation sensor via ign-transport is not required.

Run the command below and explore the topics being published over ignition transport.

$ ign gazebo segmentation_scene.sdf

6. Add a Realsense depth camera to your world. Use available model files from Fuel. Capture RGB and Depth images from the depth camera

Questions

1. Explain what information is being published over each of the ignition topics.
2. What are the message types of them?
3. Do you see the same list of topics in ros2 topics list? If the answer is no, then why?

3. ROS Integration with Gazebo

So far, you familiarized yourself with the capabilities of Ignition Gazebo. Let's try some exciting examples from this open-source repository which integrates ROS and Ignition Gazebo.

https://github.com/gazebosim/ros_gz/tree/humble

The installation instructions are specified here. After the installation is complete you can run each of the 'ros2 launch' commands to run the sequence of demos from here. Try each of the demo examples and interact with rviz visualizations.

On a separate terminal observe the rostopics being published while the above commands run.

Questions:

1. What are the different topics being published, what information do they publish? What are the message types?
2. The rostopics were not being published in the example in section 2.2. However, the rostopics are being published over the ROS master in this implementation in section 3. What is the reason for this difference between the two implementations?

4. ros_ignition_bridge

The communication between ROS and Ignition transport is bridged using a library called ros_ignition_bridge. This serves as a tool which translates messages from Ignition transport to ROS messages. You already installed the library while installing dependencies in a previous section so there is nothing to install in this section. You can read more information about the library here. Let's look at how the command works

Task

1. Run your previous example with segmentation camera with Ignition.
2. On a 2nd terminal run the command below.

```
$ ros2 run ros_gz_bridge parameter_bridge
        /panoptic/camera_info@sensor_msgs/msg/CameraInfo@ignition.msgs.CameraInfo
```

3. On a 3rd terminal run the command to list rostopics. Now you can see the translated rostopic /panoptic/camera_info being published over ROS.
4. Try to bridge more messages from 3 more different message types and observe them being published over ROS.

Note: The ign_ros_bridge only supports certain types of messages to be converted to ROS messages. Have a look at the supported types.

5. Implement your own task

You gained all the fundamental knowledge on ROS CLI commands and ROS client libraries from the tutorials you followed in Exercise 1. Exercise 2 taught you the fundamentals of Gazebo simulation and ROS integration of Gazebo. This is a task to demonstrate that you have understood all the concepts and can present them in a creative way.

Task:

Each subtask advances with the understanding of the core concepts learned so far and their connection. The more subtasks you implement, the higher the better.

1. Implement your own Gazebo simulation with sensors and other plugins for a simple mechatronic/robotic application. You have the freedom to choose your own application. The segmentation camera and dataset generation you implemented in section 2 is a good example. But try to be creative and implement your own application.
2. Bridge the topics from ignition_transport to ROS.
3. Write a Python/C++ script to subscribe/publish to ROS topics and add functionality to your application.
4. Create a ROS package and build your application as a ROS package, so that you can launch the package using ‘ros2 launch’ command as in Section 3.

Arrange a demo, contact: Eetu (eetu.airaksinen@tuni.fi)