Tampere University
Tampere University of Applied Sciences

Installation and usage of NeBoLab Gazebo ROS-simulator

Date: 12.2.2024

Introduction

This document covers the installation steps for the NeBoLab Gazebo simulator ROS package, as well as the basic usage of it. After following this document, you should be able to run your own controller inside a world of your own creation.

1. Installing the nebolab_gazebo_sim ROS2 package

In this chapter the installation of *nebolab_gazebo_sim* package will be explained step by step. After these steps you should be able to run the simulator on your own computer

The pre-requisities for this guide:

- ROS2 installed (tested with Humble & Foxy)
  https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html

If errors occur during package building, check for any missing dependencies and remember to source the workspace after running the build command!

$ colcon build --symlink-install

$ source install/setup.bash

First step is to create the ROS2 workspace. You can skip this step and continue to the next step if you already have a ROS2 workspace.

```
$ mkdir -p ~/ros2_ws/src
```

Clone the repository to the workspaces **src** folder:

```
$ cd ~/ros2_ws/src
$ git clone https://github.com/TUNI-IINES/nebolab_gazebo_sim.git
```

Install the dependencies of *nebolab_gazebo_sim* package:

```
$ cd ~/ros2_ws/
$ rosdep update
$ rosdep install --from-paths src --ignore-src -r -y
```

If dependencies were installed correctly we should be able to build the package:

```
$ colcon build --symlink-install
```

Note that after compiling, you **will** get two warning messages (at least in ROS Humble) about deprecation of install tools. This is completely normal and known behaviour of the ROS build system in some versions and can safely be ignored. Here is an example output of successful build:

```
...
 Summary: 1 package finished [1.45s]
   1 package had stderr output: nebolab_gazebo_sim
...
```

For the ROS system to find the packages, you have to ***source* the workspace every time after opening a new terminal and rebuilding**. This can be done with the following command:

```
$ source ~/ros2_ws/install/setup.bash
```

(optional) You can add this to your *.bashrc* file, so it will automatically run everytime you open a fresh terminal:

```
$ echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc
```

Before launching the simulator, you have to define which Turtlebot3 model you want to simulate. You can find all the options here. In this guide we will use the **burger** model :

```
export TURTLEBOT3_MODEL=burger
```

Now you should be able to launch the simulator with:

```
$ ros2 launch nebolab_gazebo_sim CCTA_setup.launch.py
```

2. Setting up and running the controller

If you want to set up your own controller, put your controller .py file in the location of:

```
~/ros2_ws/src/nebolab_gazebo_sim/nebolab_gazebo/sim/controller/
```

There is already one controller file "*CCTA_GoToGoal.py*" which can be used as an example. Most notable things are that your robots pose can be subscribed from "/tb3_0/posc" topic, and the output of your controller should be sent to "/cmd_vel" topic when using the simulator.

Once you have implemented your controller in the mentioned folder, add it to the *setup.py* file in the nebolab_gazebo_sim package folder, so that ROS launch process will find it later. The line 51 in that file can be used as an example.

Now you should be able to run the controller after rebuilding the package with:

```
$  colcon build --symlink-install & source ~/ros2_ws/install/setup.bash
$ ros2 run nebolab_gazebo_sim <your-controller-name-in-the-setup.py >
```

For example:

```
$ ros2 run nebolab_gazebo_sim CCTA_GoToGoal
```

3. Creating your own worlds for Gazebo

To create your own scenarios for the simulator, you have to create so called “world file” for the Gazebo. For this you have two options: Create the world in the Gazebo level editor, or create a .yml format file that describes the objects and convert that into a Gazebo world file.

There are three example scenarios in the nebolab_gazebo_sim package, two of which are already in the format of .world files (the empty_world.world and a example_room.world) and one example scenario in the .yml format (example_scenario.yml, which has to be converted into .world file).

3.1. Creating a world in Gazebo level editor

This subchapter guides you through how to create the ready-to-run .world file in the Gazebo editor. If you want to create the world in .yml file format, move to the next subchapter!

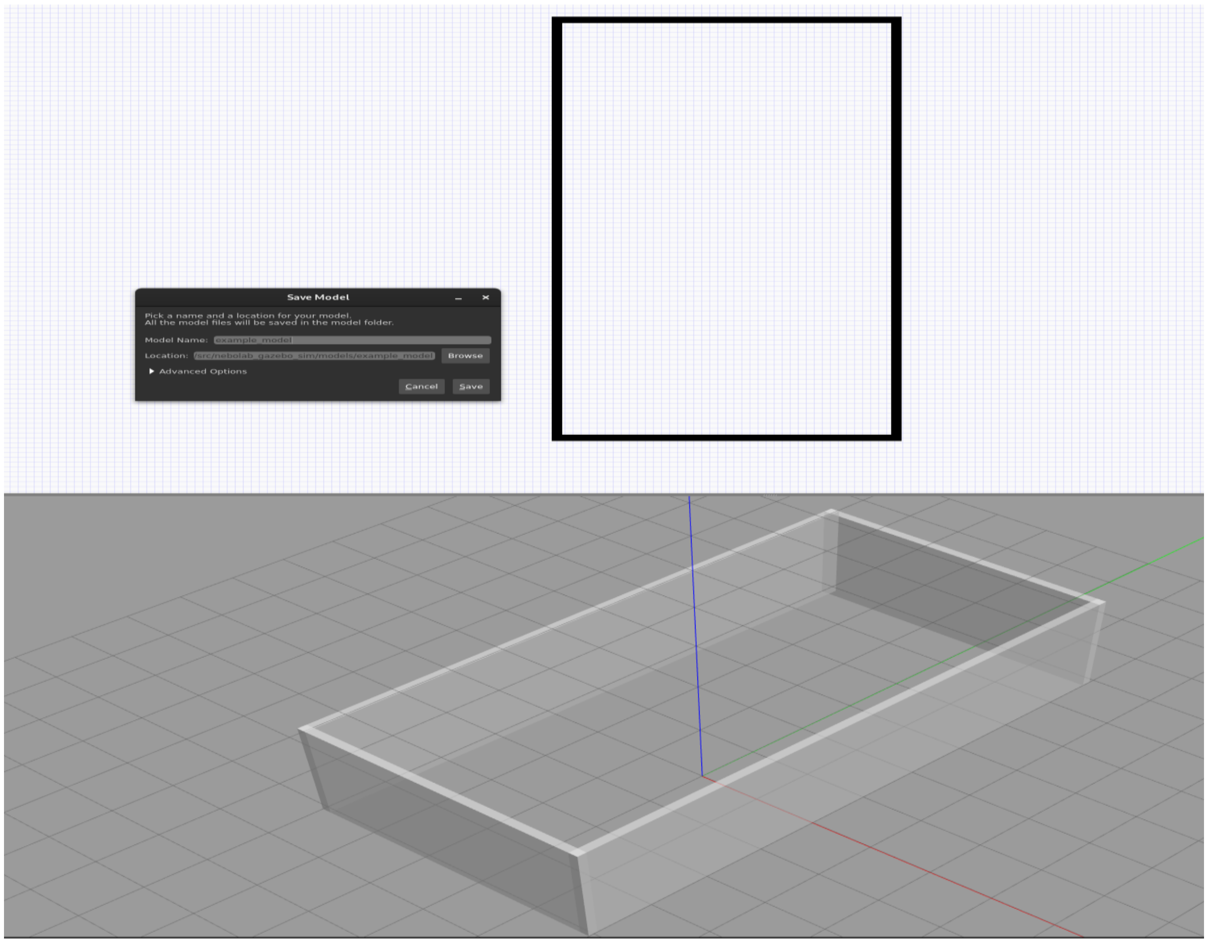
To start the world creation in Gazebo editor, open up a fresh standalone Gazebo, by running:

```
$ gazebo
```

Then press Ctrl+B (or from the upper bar press, Edit -> Building Editor). Now you should the Building Editor menu open. From here you can start creating the floorplan for your world. Just press the “Wall” icon on the left and start drawing walls to the floorplan. Once satisfied with your world, save it with Ctrl+Shift+S (or from the upper bar, File -> Save As). IMPORTANT: then save it to the location of

```
~/ros2_ws/src/nebolab_gazebo_sim/models/
```

The name can freely be chosen.



Save Model

Pick a name and a location for your model.
All the model files will be saved in the model folder.

Model Name: example_model

Location: /src/nebolab_gazebo_sim/models/example_model

Example of a simple model, and its save path.

After saving, exit the building editor (Ctrl+X or File -> Exit Building Editor), but do not close Gazebo just yet! Save the world by pressing Ctrl+Shift+S (File -> Save World As). IMPORTANT: The world save location must be folder:

```
~/ros2_ws/src/nebolab_gazebo_sim/worlds/
```

The name of the world can be chosen freely.

3.2. Creating a world with the .yml format

This subchapter guides you through the process of crerating a world in the .yml format. If you are satisfied with creating a world in Gazebo editor, you can safely skip this chapter and move on to the next!

Create a .yml file into the location of :

```
~/ros2_ws/src/nebolab_gazebo_sim/scenarios/your_scenario.yml
```

Check out the ***example_scenario.yml*** for the yml format.

Once you are done with editing the .yml file, convert it to a .world file by running:

```
$  colcon build --symlink-install & source ~/ros2_ws/install/setup.bash
```

```
$ ros2 run nebolab_gazebo_sim yaml_to_world --ros-args –p yaml_name:=your_scenario
```

Replace the “your_scenario” with the desired name **WITHOUT the .yml suffix in it.**

This command creates a **your_scenario.world** file in the share space of your workspace (and not in the package folder in src).

Then also notify Gazebo about the new location of models of your world by running:

```
$ export
GAZEBO_MODEL_PATH=~/ros2_ws/install/share/nebolab_gazebo_sim/models:$GAZEBO_MODEL_PA
TH
```

IMPORTANT! If you want to make changes to this world you have to remove the share space (as well as the build space) of your workspace, and then edit your .yml file, rebuild, convert and so on... This can be done by running the following commands:

```
$ rm -rf ~/ros2_ws/install ~/ros2_ws/build ~/ros2_ws/log
```

And then follow the same process as before in this chapter.

4. Launching your own world

To launch the newly created world in the simulator, you have to edit the CCTA_setup.launch.py file:

```
~/ros2_ws/src/nebolab_gazebo_sim/launch/CCTA_setup.launch.py
```

There is a marked section in that file, where you should change the world file name. (Edit the line 25):

```
"empty_world.world",
```

with the name of the .world file you just created (if you created it with .yaml format the naming should be ***yaml_name_without_suffix.world***). Then rebuild the workspace, and run the simulator again:

```
$ cd ~/ros2_ws/ && colcon build --symlink-install
$ ros2 launch nebolab_gazebo_sim CCTA_setup.launch.py
```

And you should have your newly created world appearing in the simulator.