Project Gazebo robot movement and enviorement

STEP 1: Create Workspace Folders and Clone the Packages

Here, we need to mention that the installation process presented below is tested in ROS Noetic distribution. The first step is to create workspace folders and clone the packages from an online GitHub repository. First, open a Linux terminal window and type

```
mkdir -p ~/test_turtlebot/src
cd ~/test_turtlebot/src
```

These two code lines will create a workspace folder called “test_turtlebot” and inside of that folder a subfolder calles “src”. You can change the name of the workspace folder as you wish. However, the name of the subfolder “src” should not be changed. The second command will change the current folder to the subfolder “src”

Then, in the currently active subfolder “src”, we clone (copy from) the remote repository the following TurtleBot 3 packages:

```
git clone https://github.com/ROBOTIS-GIT/turtlebot3.git

git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
```

```
git clone
https://github.com/ROBOTIS-GIT/turtlebot3_simulations
.git
```

After executing these commands, the folder structure of the remote repository will be copied to the local “src” folder. Next, we change the active folder

```
cd ~/test_turtlebot
```

Then, we make the packages and workspace by typing

```
catkin_make
```

In order to be able to use the created packages, we need to run the source command:

```
source ~/test_turtlebot/devel/setup.bash
```

By executing this command, we sourced “setup.bash” file which contains the configuration parameters of our packages.

STEP 2: Simulate TurtleBot Robot in Gazebo

As you can read on this webpage, the Turtle Bot 3 package comes with 2 robot options: Burger and Waffle Pi. We use Burger in our simulations. Consequently, in the same terminal window that is used to install the packages, we type

```
export TURTLEBOT3_MODEL=burger
```

Next, let us run the Gazebo simulation. We do that by typing:

```
roslaunch turtlebot3_gazebo
turtlebot3_empty_world.launch
```

This command will create an empty map in Gazebo, with the CAD model of TurtleBot 3. However, we cannot move the robot in this environment. To be able to move the robot, we need to open a new terminal window, and to run a teleop node that will enable us to control the robot by using the keyboard keys. After opening a new terminal (while keeping the old terminal open), type

```
source ~/test_turtlebot/devel/setup.bash
```

This command will source the workspace. Then, we need to select the robot by typing

```
export TURTLEBOT3_MODEL=burger
```

Finally, we can run the teleop node by typing

```
roslaunch turtlebot3_teleop
turtlebot3_teleop_key.launch
```

As explained in the video tutorial, we control the robot motion by pressing the keyboard keys “a,s,d,x” and “w”. This will enable us to control the robot. Now, go back to the simulation environment, and try to control the robot. You still will not be able to control the robot. To control the robot, the terminal which is used to start the teleop node has to be active. So keep this terminal active, and try to move the robot (you can do that by simply moving the terminal to the edge of the screen and by clicking on the terminal and keeping the mouse pointer in the area of the terminal). You will be able to move the robot.

Besides an empty Gazebo simulation world. We can use two other worlds that can be initiated like this (make sure that you first closed the original

Gazebo simulation, and make sure that you abort the current Gazebo simulation by pressing CTRL+C in the original terminal):

```
1  roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

and

```
1  roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

The last Gazebo world represents a house which is a very important world for testing nagivation and SLAM algorithms.

Mapping the world

Overview

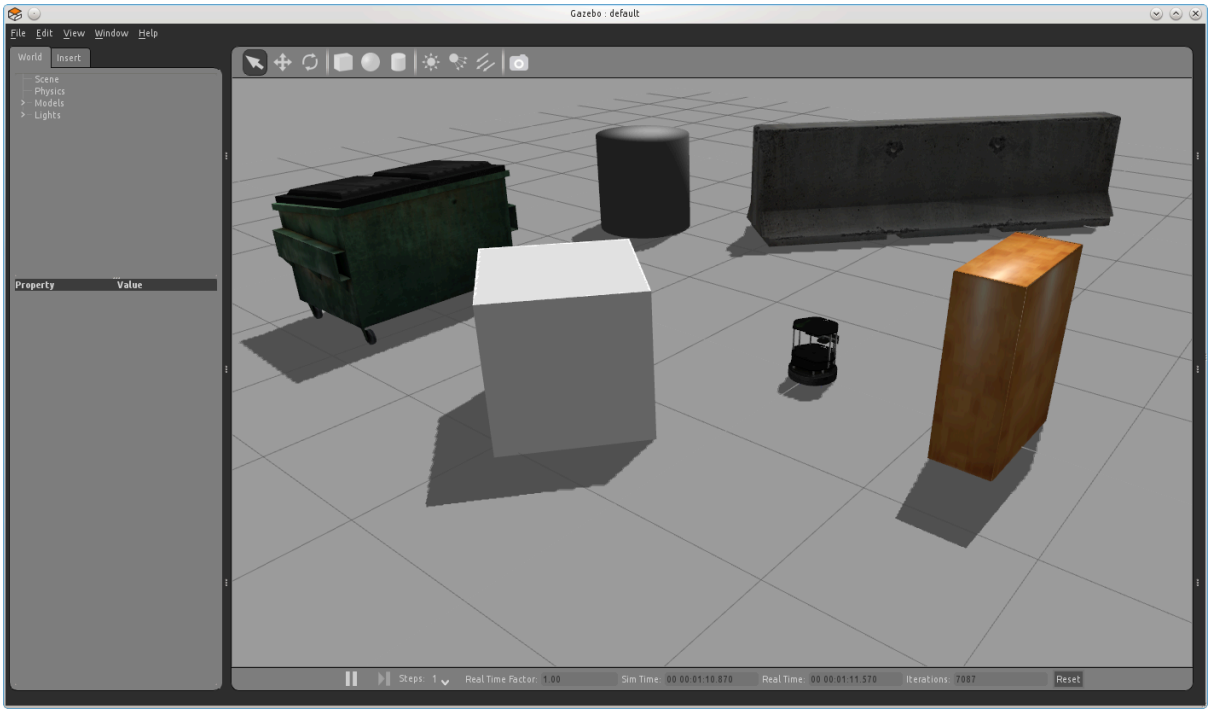
In this tutorial we will create a map of a virtual world and then use it to navigate around. Basically we will reproduce build a map and navigate a map TurtleBot tutorials in a simulated world. So if you have doubts about using the navigation stack with TurtleBot those tutorials will probably help you.

Make a map

First, bring up the TurtleBot simulation as described in the Gazebo Bringup Guide, but this time we will load a slightly funnier world:

```
$ source /opt/ros/indigo/setup.bash
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

The playground world with a TurtleBot2 looks like this:



Alternatively you can use another existing world file like this:

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
world_file:=worlds/willowgarage.world
```

Tip: of course you can also make your own world! With Gazebo it's not difficult, but is beyond the scope of this tutorial.

You can customize your simulated TurtleBot by setting TURTLEBOT_XXX environment variables; for example:

```
$ export TURTLEBOT_BASE=create
$ export TURTLEBOT_STACKS=circles
$ export TURTLEBOT_3D_SENSOR=asus_xtion_pro
$ roslaunch turtlebot_gazebo turtlebot_playground.launch
```

will simulate a TurtleBot 1 with an Asus Xtion Pro camera.

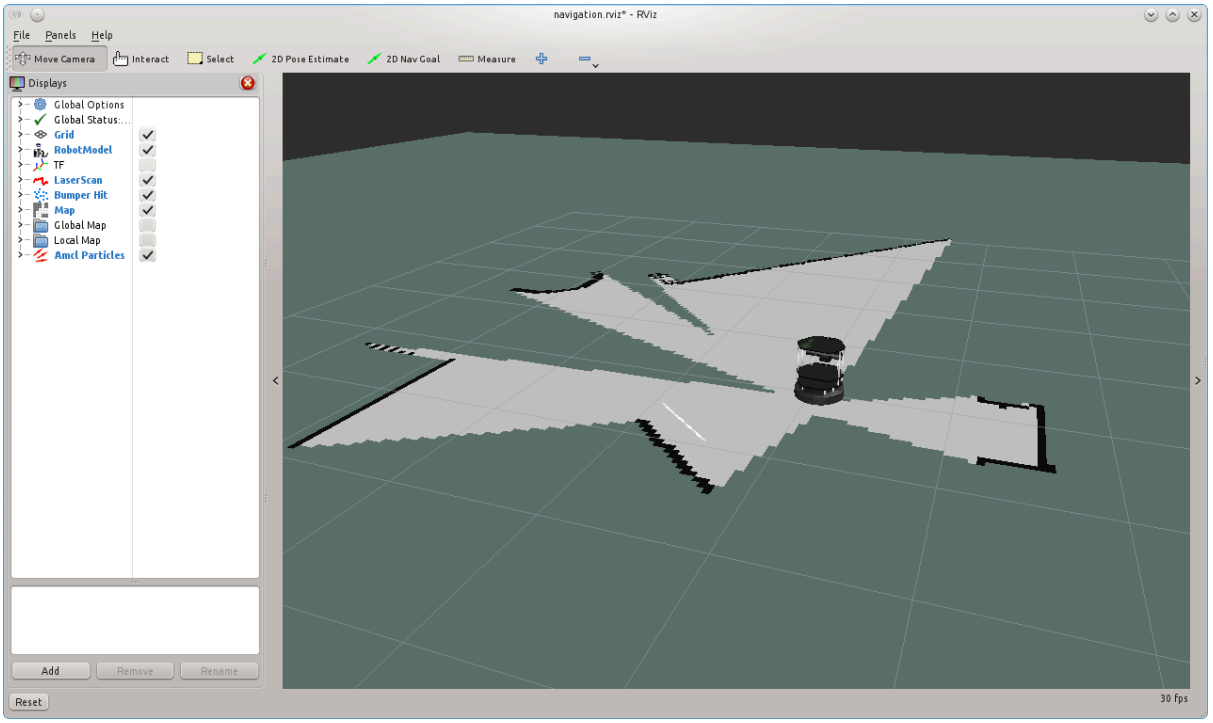
To start map building, type:

```
roslaunch turtlebot_gazebo gmapping_demo.launch
```

Use RViz to visualize the map building process:

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Then, as explained in the previous tutorial, use your favorite teleoperation tool to drive the TurtleBot around the world, until you get satisfied with your map. This capture shows the mapping process after turning 360 degrees.



Finally, save the map to disk:

```
$ rosrun map_server map_saver -f <your map name>
```

Navigate the playground

To be sure that all will run as expected, kill all you launched in the previous section and repeat all steps except the map building. Instead, type:

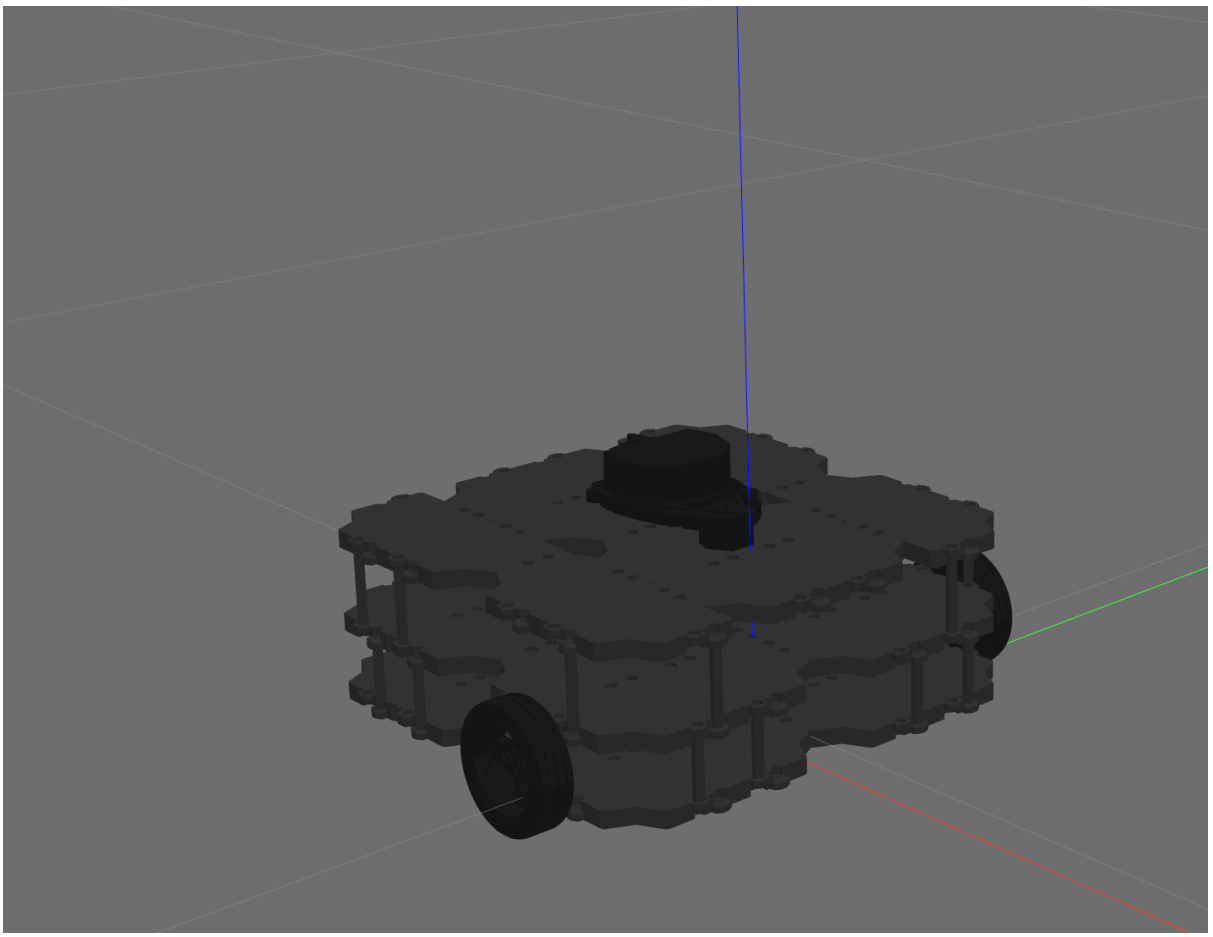
```
roslaunch turtlebot_gazebo amcl_demo.launch map_file:=<full path to your map YAML file>
```

Or if you prefer to use an already created map, just omit the map_file argument.

Now you can send the robot anywhere in the playground with RViz, same way as explained on navigate a map tutorial.

Gazebo World

The Gazebo Simulator



We will use the Gazebo simulator for this course. It offers a number of powerful abilities is quite customizable.

Starting the Simulator

---

1. Start `rscore`.

   Go to the terminal and run the following command.

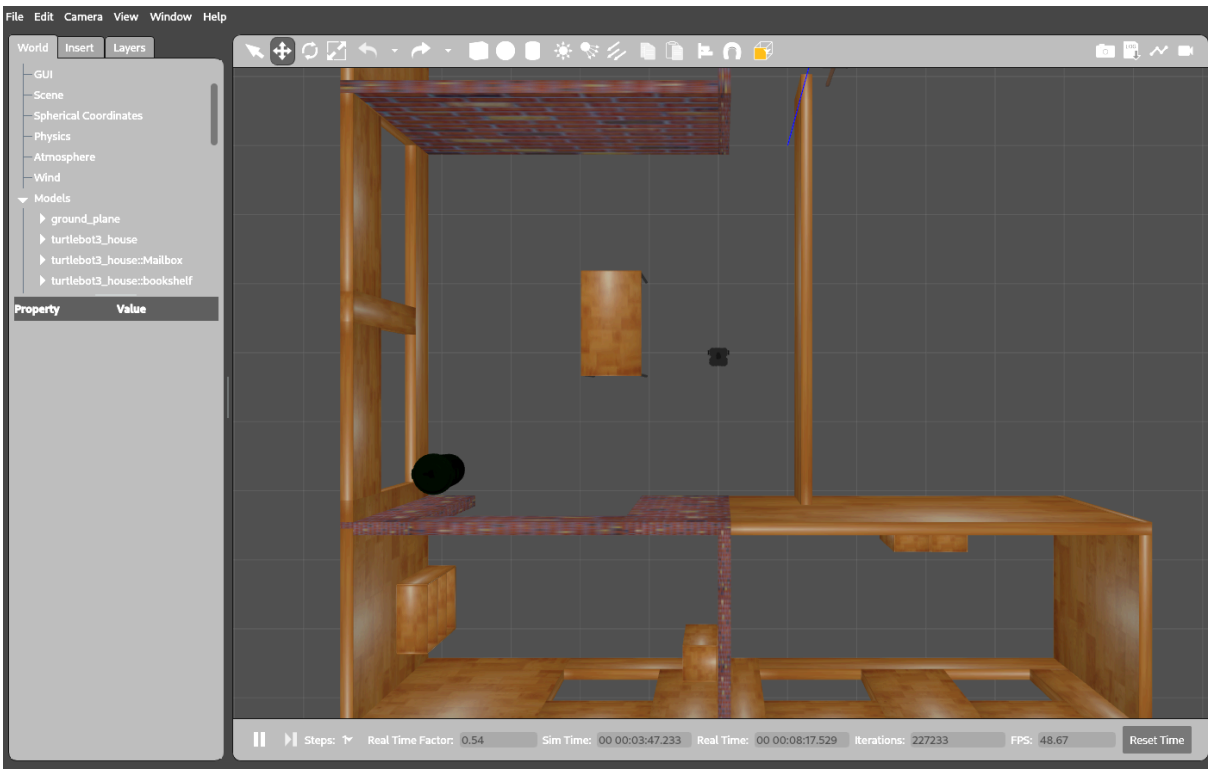
2. `$ roscore`
3. You should see an output that looks like the following.

```
      ... logging to
~/.ros/log/9cf88ce4-b14d-11df-8a75-00251148e8cf/roslaunch-machine_
name-13039.log
      Checking log directory for disk usage. This may take awhile.
      Press Ctrl-C to interrupt
      Done checking log file disk usage. Usage is <1GB.

      started roslaunch server http://machine_name:33919/
```

```
      ros_comm version 1.15.8

      SUMMARY
      ======

      PARAMETERS
       * /rosdistro: noetic
       * /rosversion: 1.15.8

      NODES

      auto-starting new master
      process[master]: started with pid [13054]
      ROS_MASTER_URI=http://machine_name:11311/

      setting /run_id to 9cf88ce4-b14d-11df-8a75-00251148e8cf
      process[rosout-1]: started with pid [13067]
      started core service [/rosout]
```

4.
5.

6. Lunch Gazebo.
   You can launch the robot in many simulated worlds. To see a list of worlds, type in
7. `$ roslaunch turtlebot3_gazebo turtlebot3_`
8. and double tap tab, which should offer you a list of possible simulated worlds. For instance, if we use the `turtlebot3_gazebo_house.launch` option, you will some terminal output and a visualization like the following will pop up.



If you want to create your own world, you can put your robot in an empty world and then follow instructions for populating your own world TODO.

9. `$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch`

Using the Gazebo GUI

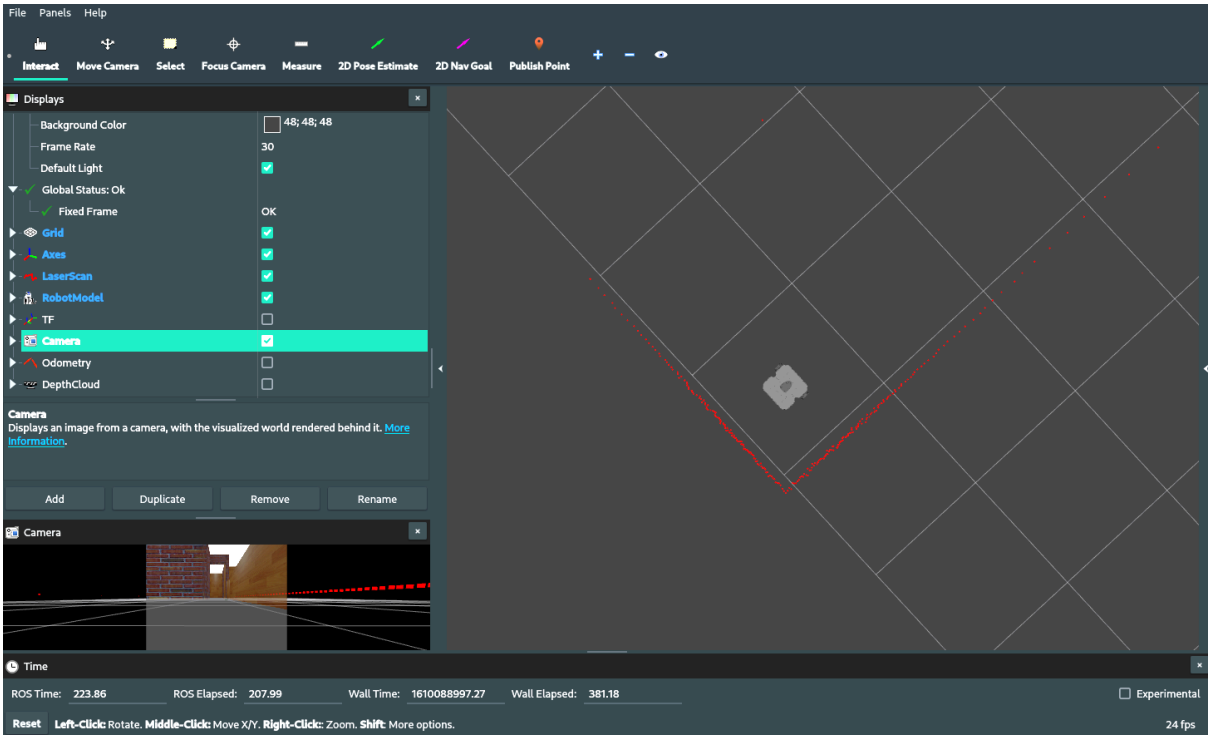
The Gazebo website has a guide on using the Gazebo graphical interface.

Using RViz with the Simulator

To launch an instance of RViz, run the following command.

```
$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

This command takes care of preparing the setup for you, so the RViz instance would be ready for you from the beginning. If all goes well, you should have a window that looks like the following.



RViz vs Gazebo

While both RViz and Gazebo are graphical user interfaces that seem somewhat similar, their nature is quite different. Gazebo is a simulator. It places robots in a virtual world and allows users to give it commands and generates simulated sensor data. This sensor data is not very human friendly (e.g. some numbers and binaries). RViz simply visualizes this generated sensor data so that human users can understand what their robot *observes* in human terms.

Topics

Note that like any other ROS node, the robot simulator publishes and subscribe to topics. In this case, the simulator publishes and subscribes to the same topics that real robot does.
The following documentation gives high level details of a few topics. To explore more use the tools discussed in ROS Resources.

`cmd_vel`

When you publish to this topic, you'll set the robot's velocity. The `linear.x` direction sets forward velocity and `angular.z` sets angular velocity.

`scan`

The robot's LiDAR publishes measurements to this topic. LiDAR is used for measuring distance via lasers. The laser scan message consists of a number of attributes,

```
$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

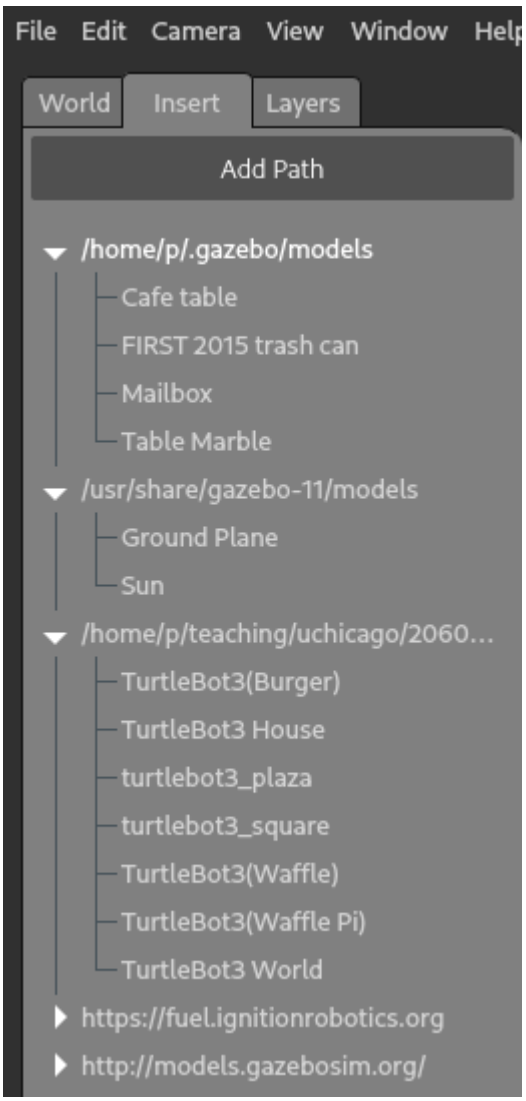
For initial purposes, the most important one is ranges, a list of 360 numbers where each number corresponds to the distance to the closest obstacle from the LiDAR at various angles. Each measurement is 1 degree apart. The first entry in the array corresponds to what is in front of the robot.

`camera/rgb/image_raw`

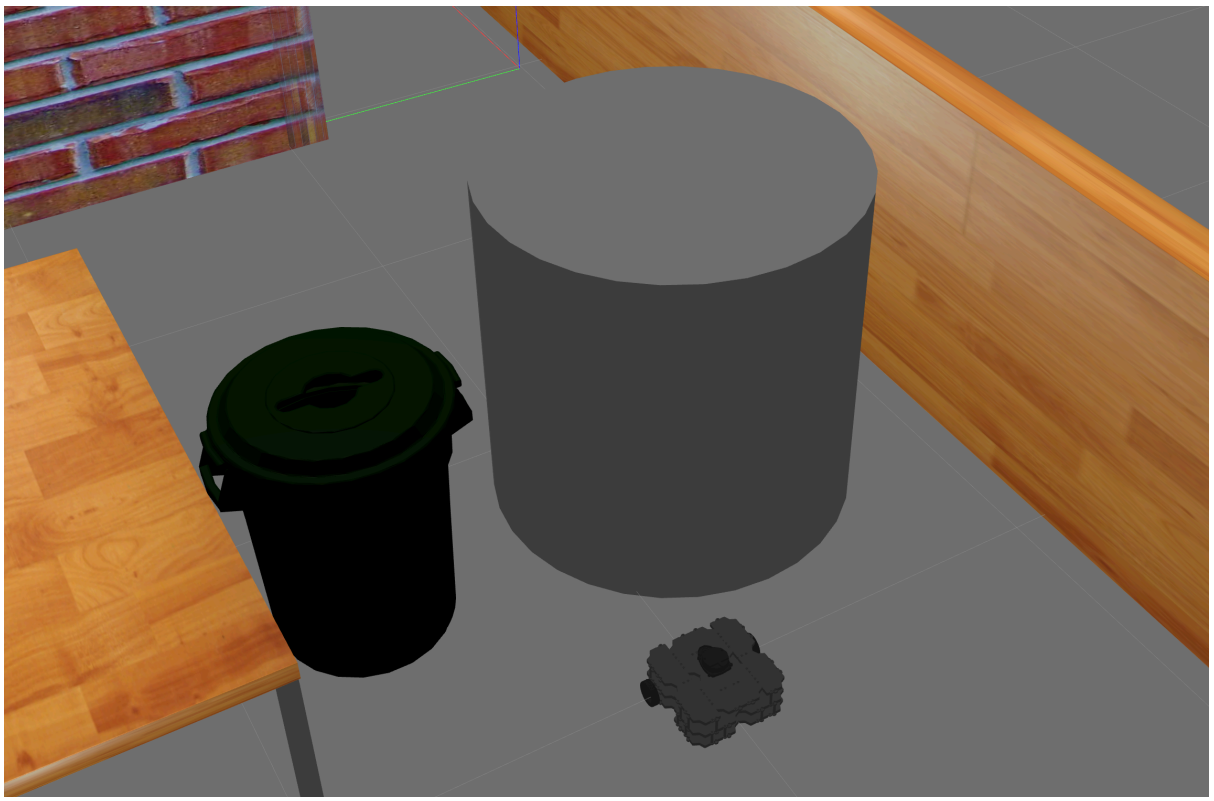
Publishes the RGB image seen by the robot's camera.

Populating a Simulated World

To populate a simulated world, use the "insert" menu (top left) in Gazebo. This menu will show a number of 3D models that can be inserted into Gazebo.



Simply click on one and drag them into your world. Below you can see the house environment populated with a cylinder and a trashcan.



For more on populating the world, refer the to Gazebo manual.

Saving The World

If you have altered a world, you can save it using *file->save world* from within Gazebo. You should save the world in the directory `catkin_ws/src/intro_robo/intro_robo_utils/worlds` as a file with the `.world` extension.

Loading the World

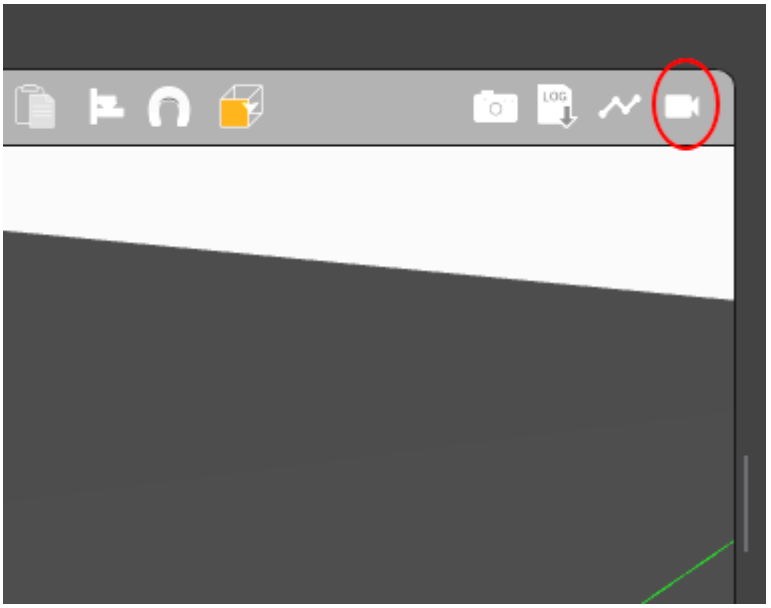
Assuming you performed the steps above, you should be able to reload your world after shutting down Gazebo via the following command. Do not include the .world extension.

```
$ roslaunch intro_robo_utils turtlebot3_custom_world.launch world:=name_of_the_world
```

Video Recording a Simulation

To record what is happening in your simulation, simply click on the video camera icon on the top right of your simulation. It should provide you with a

number of format options. For our purposes, select the mp4 format and proceed with recording your simulation. To stop the recording click again on the video camera icon. Gazebo should pop up a file manager window, ask you for a file name, and request an address for saving the file.



Once you have the mp4 video, inspect and ensure that it captures your intended behavior. In the next step, we are going to turn the video into a gif via the command line tool `ffmpeg`. Note that while the command itself is very long, you only need to replace the italics to fit it for your own purposes.

```
$ ffmpeg -i input_name.mp4 -vf
"eq=brightness=brightness,fps=frames_per_sec,scale=width:-1:flags=
lanczos,split[s0][s1];[s0]palettegen[p];[s1][p]paletteuse" -loop 0
output_name.gif
```

Here is a description of what each italic represents and the recommended values:

- *input_name*: The input file's name, i.e. what you recorded in the previous step.
- *output_name*: The name you would like your output file to have.
- *brightness*: For increasing or decreasing the brightness of your gif. Try values in the 0.01-0.1 range to brighten and negative values in the same range to darken your gif.
- *frames_per_sec*: For stating how many frames per second you would like your gif to have. Recommended values are 15-30.
- *width*: For scaling your gif's width. There is no need to set the height, the command takes care of that.

Let us know if you run into any issues with this command.

Shutting Down the Simulator

To shut down the simulator, go back to the terminal in step 2 and hit `Ctrl-C`.

Requirement:

Make a node that controls the robot in Gazebo to follow a wall.
Please choose a specific world and implement the package that will use the laser scan topic to can check the robot's surroundings.