Home Work 3
Gazebo

Robotics 2019 - ShanghaiTech University

1 Overview

The goal of this homework is to learn how to use the Gazebo http://gazebosim.org/ robot simulator and learn about robotics and ROS at the same time. You will create your own simulated world, you will create a robot model with a laser and a camera and you will use ROS and rviz to visualize the simulated data. Additionally you will use gmapping and move_base for mapping and simple autonomy.

While some hints and tutorials are given in this pdf, you should also take a look in piazza - potentially we will put some more hints there.

2 Academic Honesty

Please everybody do the homework completely by yourself. You are doing this to learn about robotics and ROS. Do not ask other students for help. Ask in piazza! If the problem persists go to the office hour of the TA's (see on piazza) and/ or ask Prof. Schwertfeger - we will help you!

3 Task 1: Create your own simulated world (25 %)

Attached to this homework you will find a 'How to Build Your Own Gazebo World' section.

You should create your own world to simulate a robot in. It can be indoors, in which case it needs to have at least 2 different rooms. Or it can be outdoors, where there must be at least some narrow passages as well as a bigger place surrounded by some obstacles. You are of course free to model the STAR Center or the ShanghaiTech campus, for example.

In any case, the world must be closed: Starting inside the world, the robot should not be able to leave the world. So there should be walls or other obstacles all around.

4 Task 2: Create a simple robot model (25%)

In the Gazebo you can also create your own robots. The robot to be simulated should be a Clearpath Jackal robot. Find the model online! The robot has to have one camera facing forward and one Velodyne HDL-32E laser scanner. The robot should move properly and the sensors should deliver data in the namespace 'camera' and 'velodyne'.

Reduce the number of samples per beam of the laser to 219 to get a good performance. Appended to this homework is an section named 'Velodyne Simulator' with some more hints.

Also make sure you create the according UDRF to be able to use the robot in ROS and visualize it in RVIZ (including the models).

Create a launchfile called 'gazebo.launch' to start your world with your robot.

5 Task 3: Autonomy! (30%)

Now you have a robot in a simulator in ROS. Time to give it some autonomy! First we need some mapping. Gmapping is a very well known 2D SLAM package available in ROS. See the appendix 'Robot

Navigation in Gazebo Simulator' for some more information. Use it with your velodyne to crate a 2D map.

Now launch ROS navigation and move_base to enable autonomy. When using rviz to set a goal point, the robot should drive to it, avoiding obstacles along the way.

Create a launchfile 'autonomy.launch' that will launch both gmapping and all of the needed ROS navigation nodes.

6 Task 4: Visualize the data in rviz (10%)

Use rviz to visualize the data. The following data should be displayed:

- the grid
- the robot model
- the camera data in an imageview
- the laser data in the 3D view
- the 2D map in the 3D view
- the tf tree

Save the configuration of rviz. Create a launchfile 'rviz.launch' to start rviz with this configuration file.

7 Task 5: Submission (10%)

Your submission consists of all the files needed to run all four tasks and a single launchfile launching them all. Everything has to be committed to a folder named 'HW3' to git and pushed to the gitlab server.

You should create individual ROS packages under the 'HW3' folder for each task (one or more per task).

The following files are important:

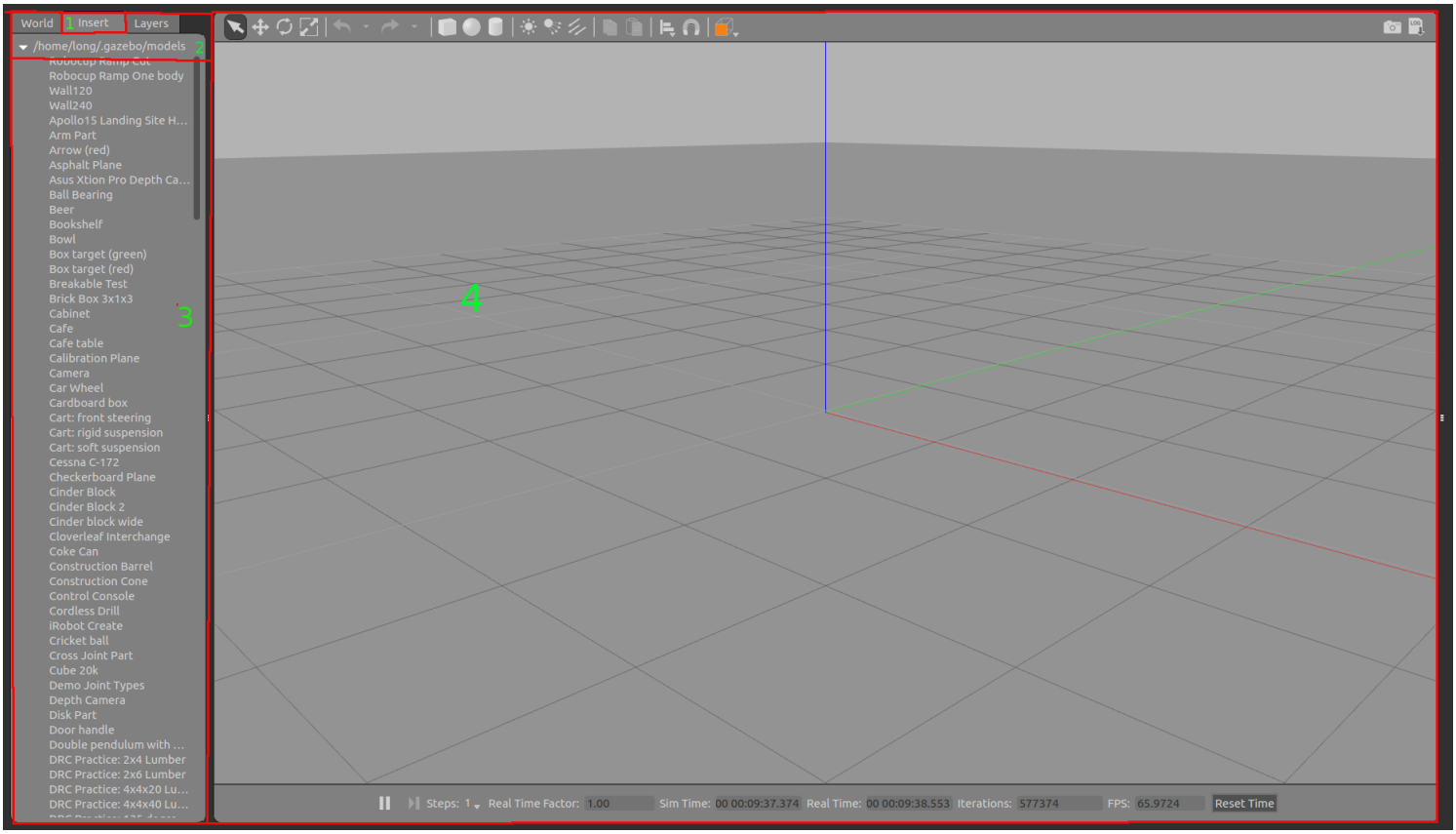
- In the root of the 'HW3' folder there should be a file called 'start.launch' which should launch gazebo, 'autonomy.launch' and rviz by including the above mentioned launchfiles. We will use this to start and test your homework!
- The files for task 3.
- The robot model from task 2.
- The world from task 1.

How to Build Your Own Gazebo World

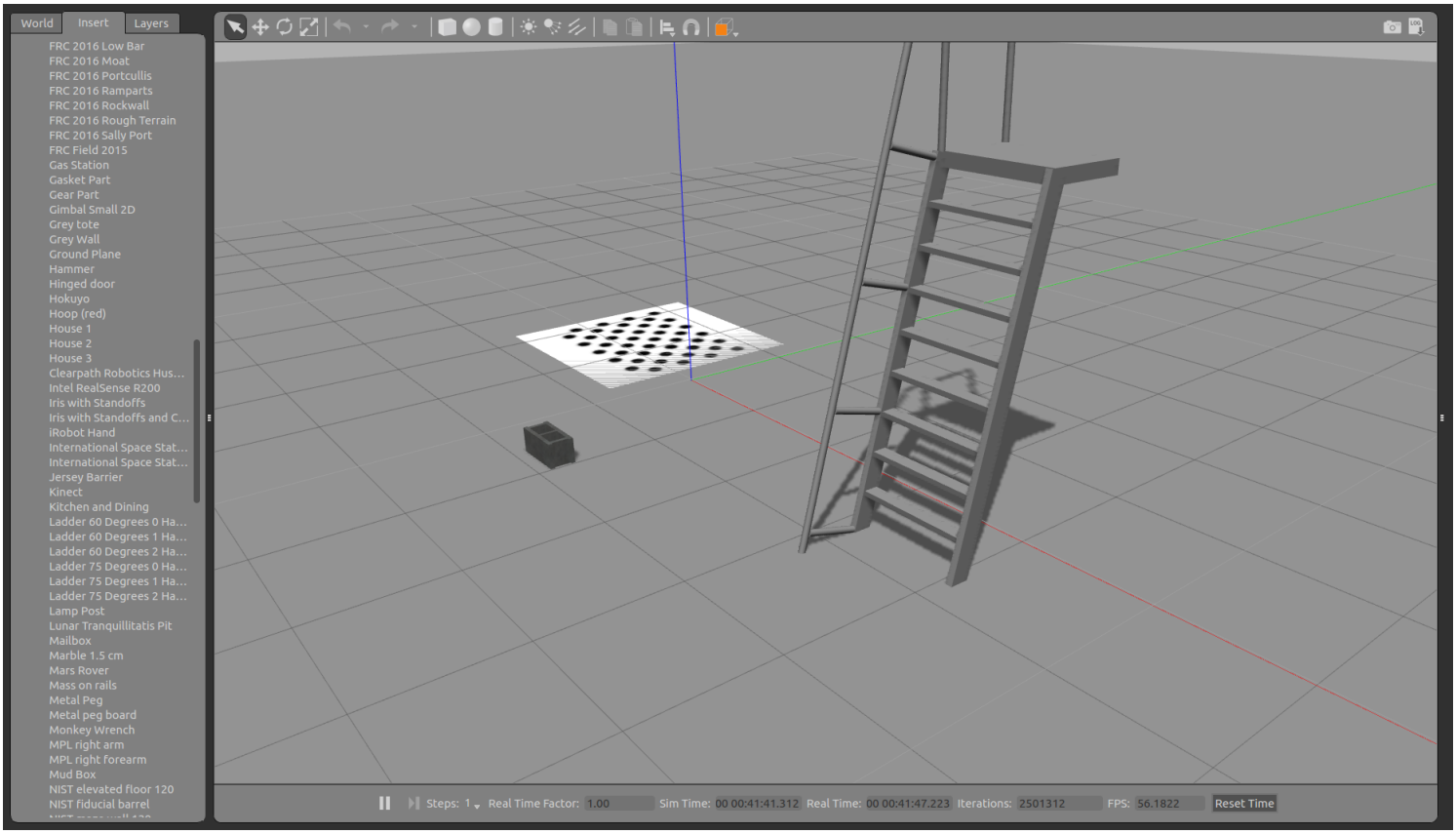
Installation Gazebo

Usually if you install ros desktop-full install. The gazebo will installed automatically, otherwise, you can either download from gezebosim or `sudo apt-get install gazebo7` will install for you. In order to use in ROS, you also need install ros related stuff by `sudo apt-get install ros-kinetic-gazebo-*` PS. Here is a tutorial for Using Gazebo plugins with ROS. It mainly show how to run simulation in gazebo and use ros to control robot and get sensor data.

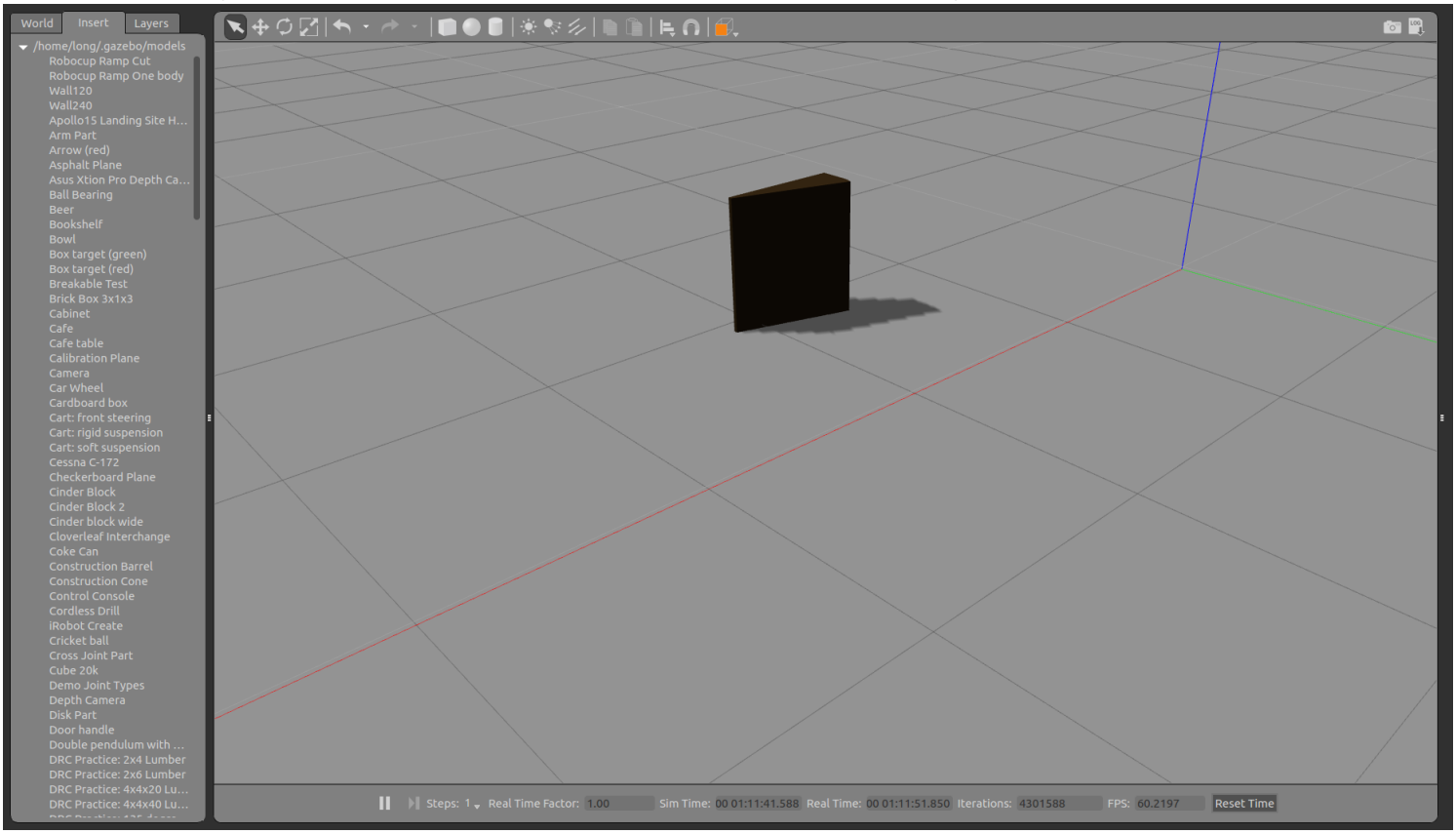
Model



The tab **Insert**(green 1) list all available model can be used for build your world. Green 2 show your local model folder. If it is empty, you can download all standard model files from gazebo_models. If you indeed have some model under the local folder, you can the list(green 3) of all model. You can left click any model and left click in visualization window(green 4) to build your world as you wish. Following is a example.



If you need any model the gazebo model don't present anyone similar. You also can diy any model you want. Following is a diyed ramp.



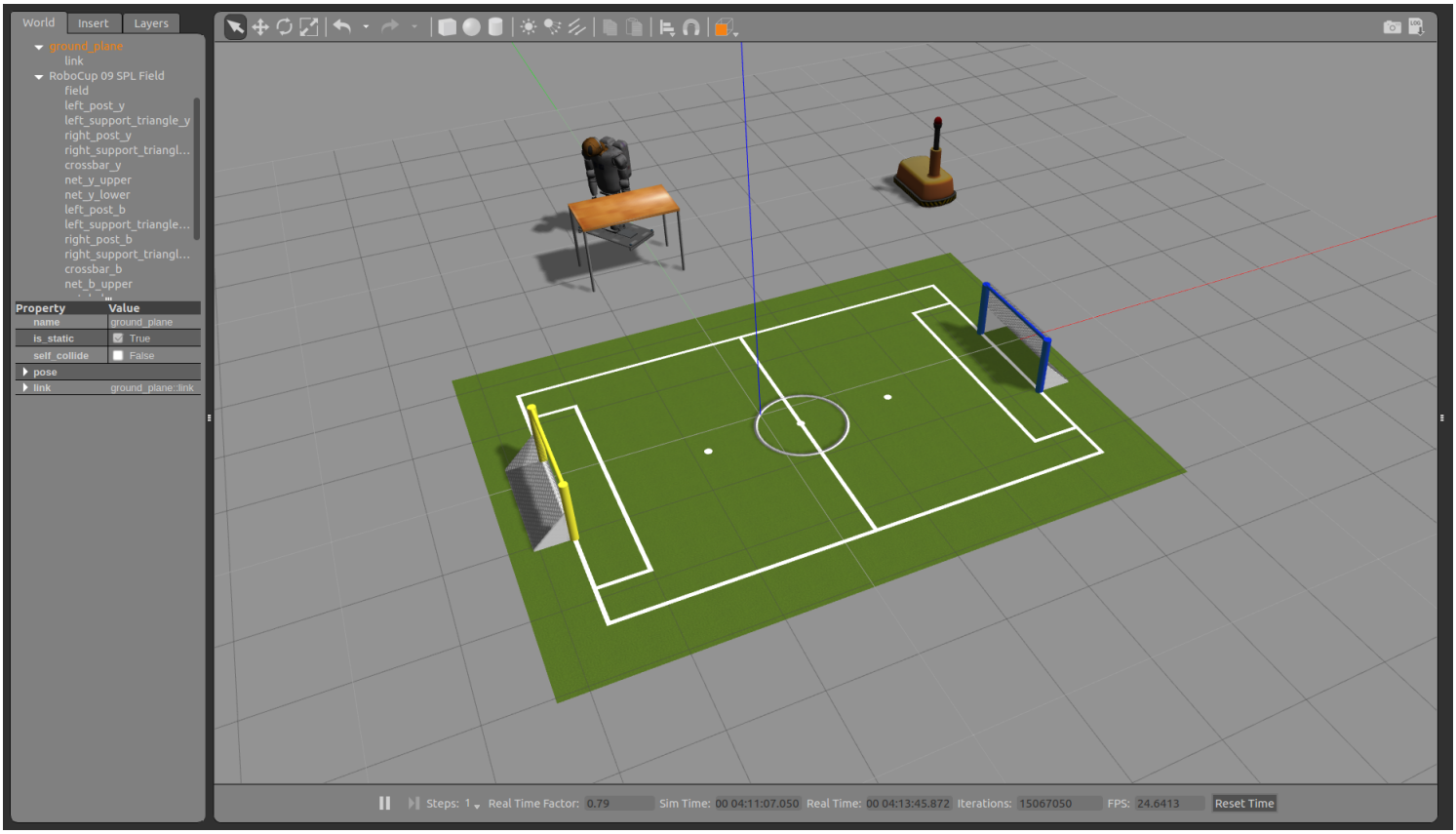
Firstly you need prepare the model mesh file both *.stl* and *.dae* extension file are OK. The *.sdf* file is model description file you need create. The synatax is similar to **urdf** and the *.config* file is used for gazebo description and specify the related *.sdf* file.

```
<?xml version="1.0" ?> <sdf version="1.5"> <model name="robocup_ramp_cut"> <static>true</static> <link name="robocup_ramp_cut_link"> <po
```

```
<?xml version="1.0"?>

<model>
  <name>Robocup Ramp Cut</name>
  <version>1.0</version>
  <sdf version="1.6">model.sdf</sdf>

    <author>
    <name>Xiaoling Long</name>
    <email>longxl@mail</email>
  </author>

  <description>
    A wooden wall ramp cut corner robocup rescue arena.
  </description>
</model>
```

Build your world

Finally you can feel free to build your own simulation world. The top of visualization window show some interactive tool such as *selection, rotation, scale*. You should save your world for next time experiment as *.world* file.

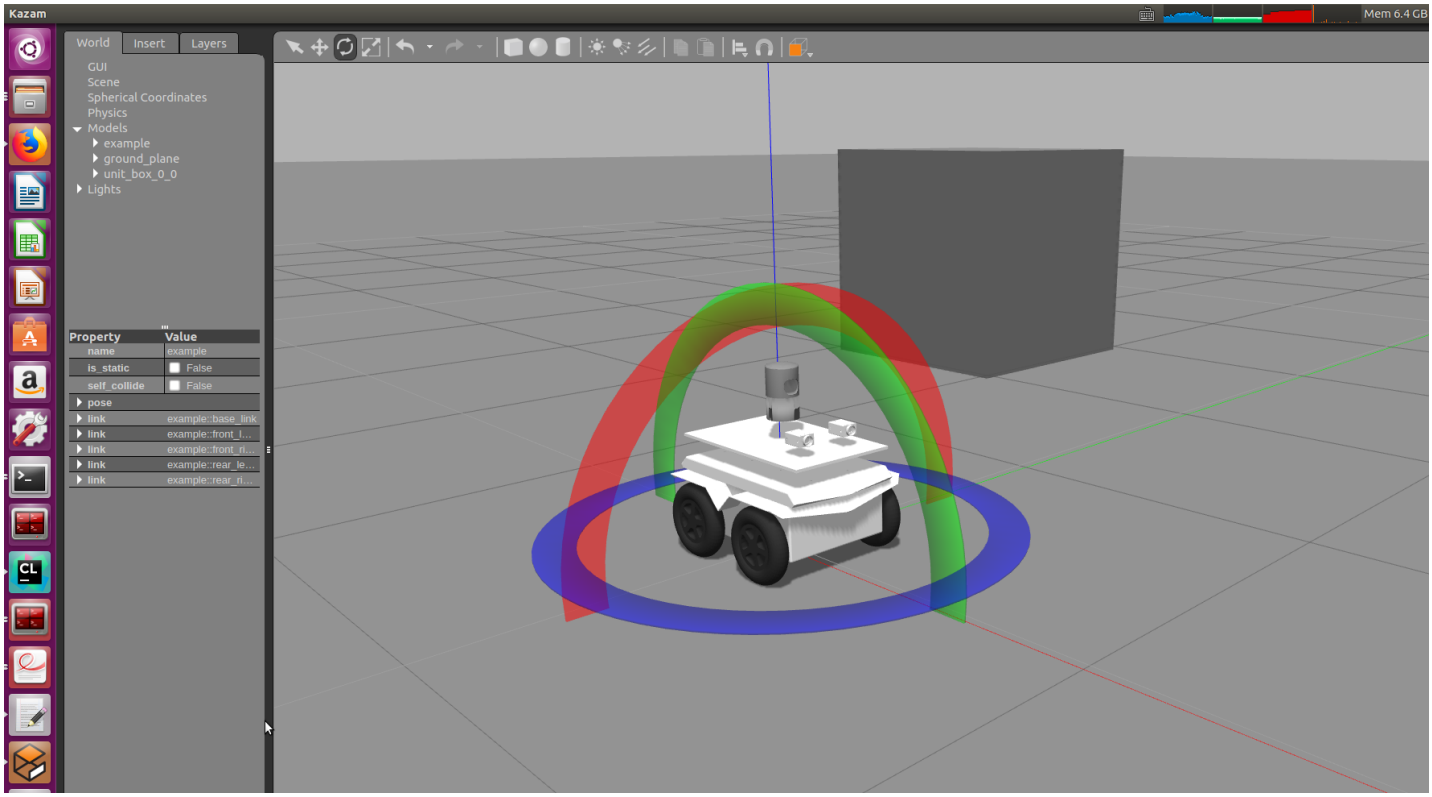


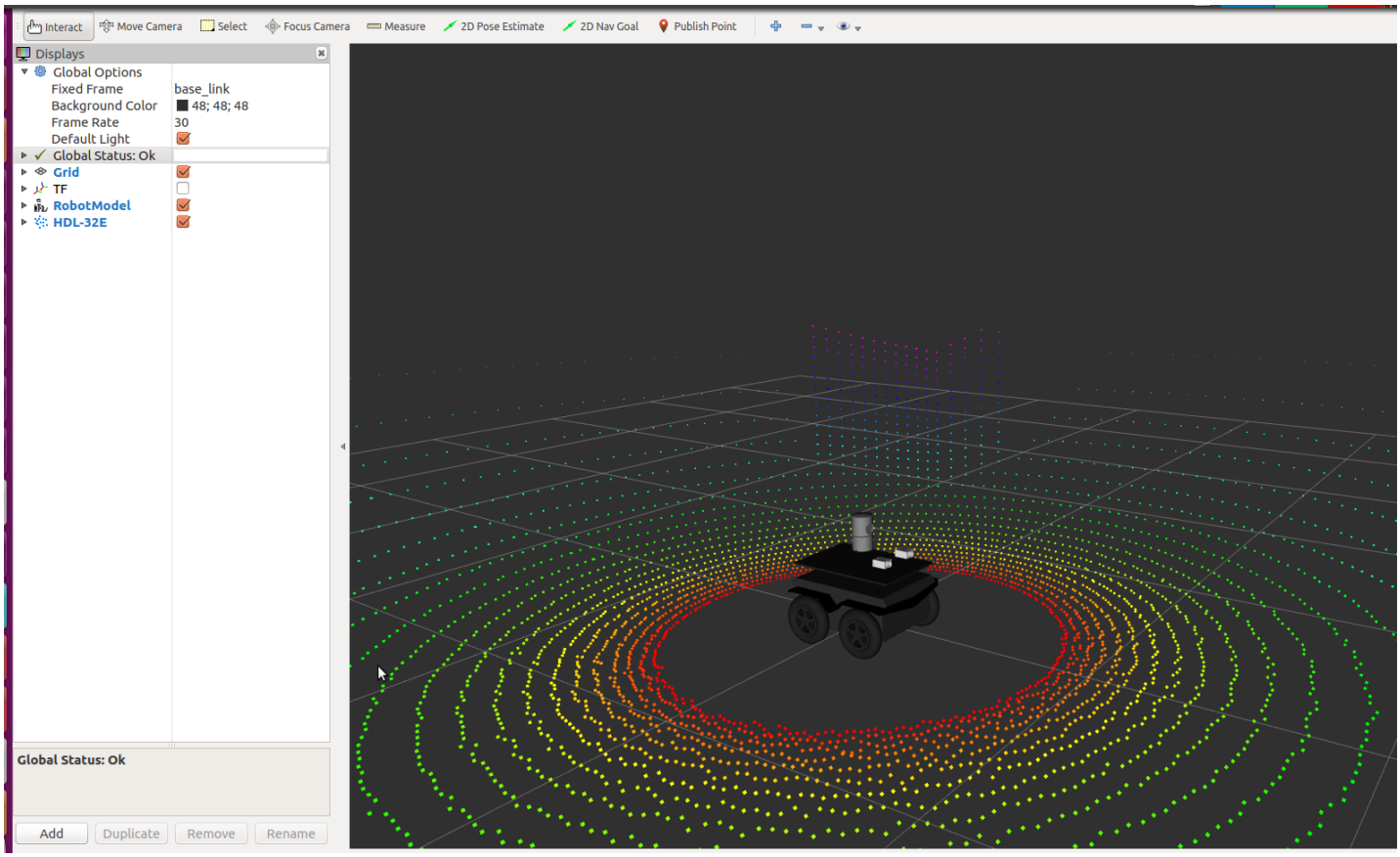
Velodyne Simulator

How to build URDF files, please see ros tutotials

`http://wiki.ros.org/urdf/Tutorials`

URDF description and Gazebo plugins to simulate Velodyne laser scanners





Features

- URDF with colored meshes
- Publishes PointCloud2
- Supported models:
  - HDL-32E

Parameters

- `*origin` URDF transform from parent link.
- `parent` URDF parent link name. Default `base_link`
- `name` URDF model name. Also used as tf frame_id for PointCloud2 output. Default `velodyne`
- `topic` PointCloud2 output topic name. Default `/velodyne_points`
- `hz` Update rate in hz. Default `10`
- `lasers` Number of vertical spinning lasers. Default `HDL-32E: 32`
- `samples` Nuber of horizontal rotating samples. Default `HDL-32E: 2187`
- `min_range` Minimum range value in meters. Default `0.9`
- `max_range` Maximum range value in meters. Default `130.0`
- `noise` Gausian noise value in meters. Default `0.008`
- `min_angle` Minimum horizontal angle in radians. Default `-3.14`
- `max_angle` Maximum horizontal angle in radians. Default `3.14`

- `min_intensity` The minimum intensity beneath which returns will be clipped. Can be used to remove low-intensity objects.

Known Issues

- At full sample resolution, Gazebo can take up to 60 seconds for the HDL-32E
- Gazebo cannot maintain 10Hz with large pointclouds
    - Solution: User can reduce number of points (samples) or frequency (hz) in the urdf parameters Gazebo crashes when updating HDL-32E sensors with default number of points. "Took over 1.0 seconds to update a sensor."
    - Solution: User can reduce number of points in urdf (same as above)

Example Gazebo Robot

```
roslaunch velodyne_description example.launch
```

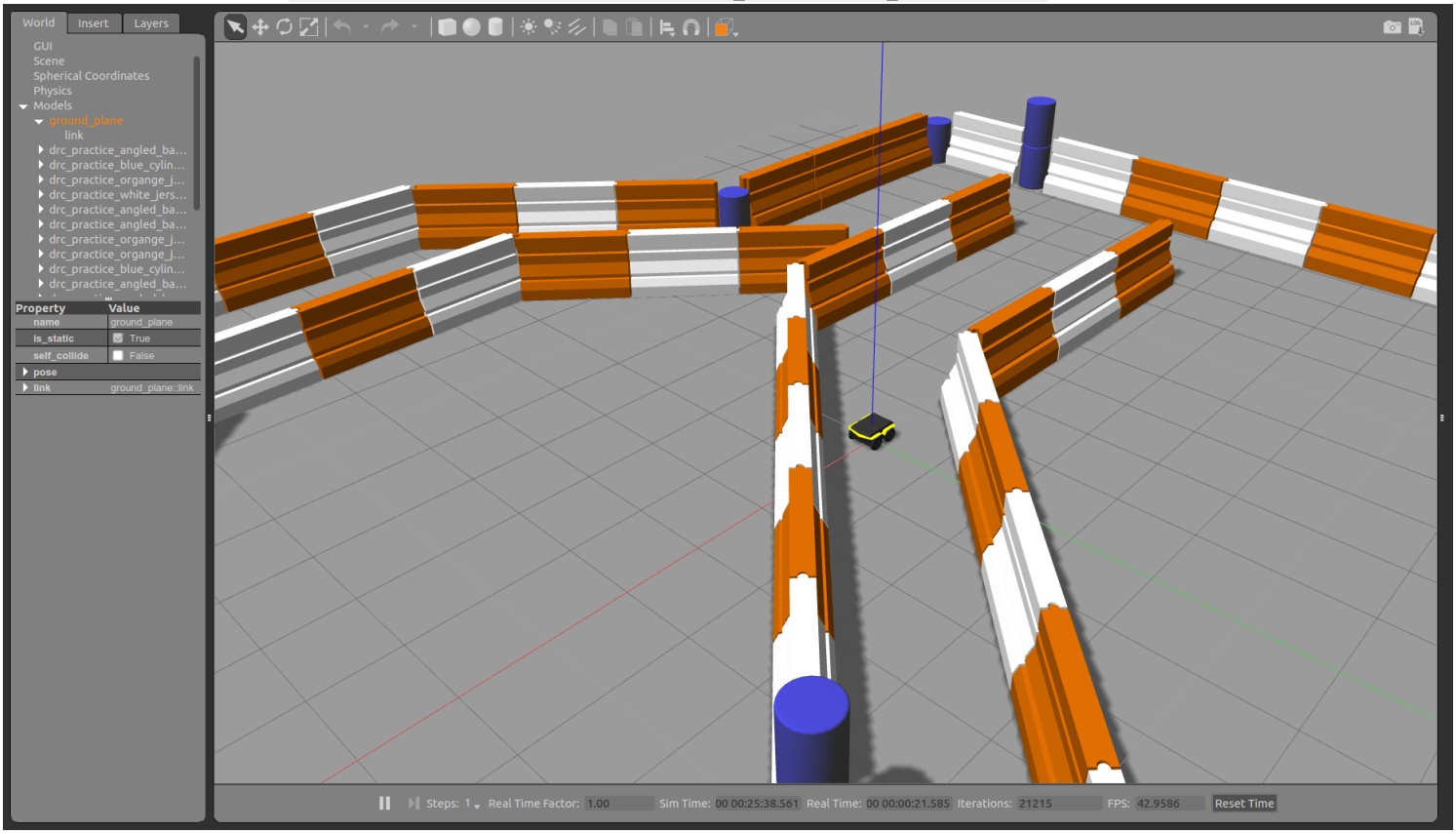
Robot Navigation in Gazebo Simulator

Robot Simulator with Velodyne Sensor

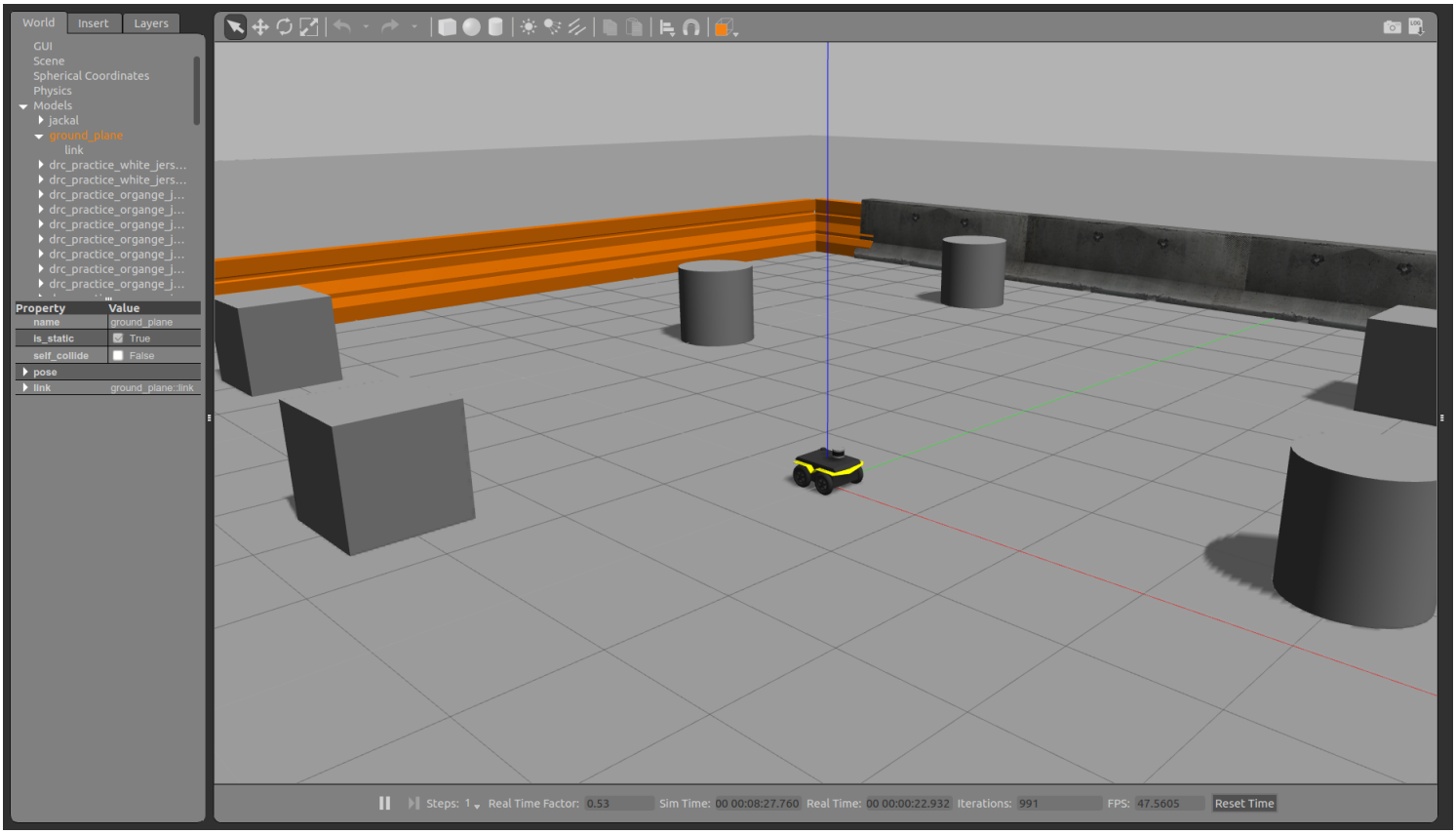
In this tutorial, we will use Jackal robot with a Velodyne VLP-16 sensor as example.

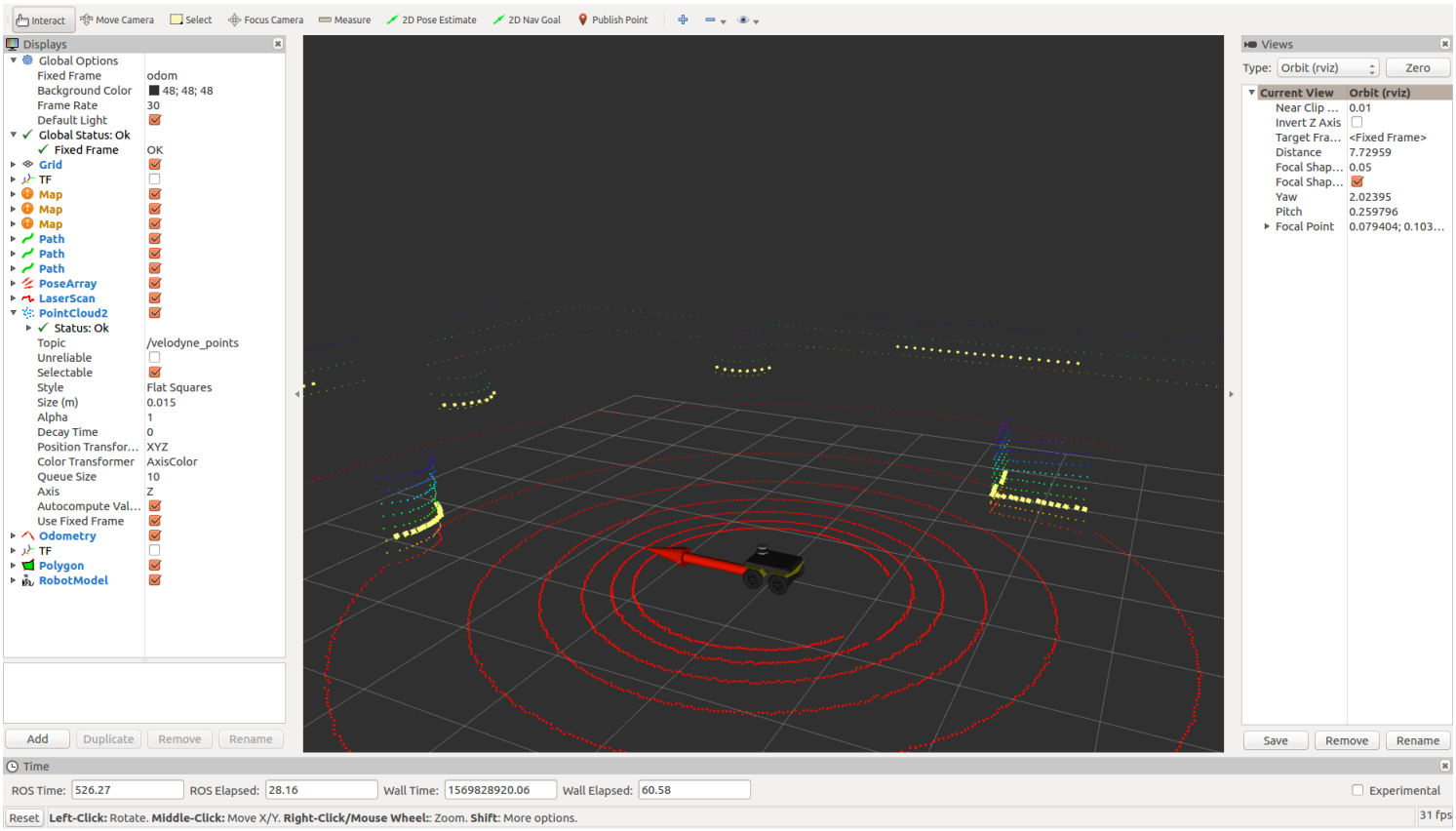
Firstly, you need make a catkin workspace e.g. hw3_ws `cd mkdir -p hw3_ws/src cd hw3_ws catkin_make`

Then, download the Jackal robot simulation packages to your workspace.
`cd hw3_ws/src sudo apt install ros-kinetic-jackal-* git clone https://github.com/jackal/jackal.git git clone https://github.com/jackal/j`
Last, launch the Gazebo simulator `source devel/setup.bash roslaunch jackal_gazebo jackal_world.launch` you will see:



Here, you need use the previous method to create your owns scenario and add the sensors to your robot, for example I will add a Velodyne VLP-16 sensors on the robot and put them in a customized simulation environment like below figures.





Here, we could see the visualization of pointclouds data in rviz. And we also transfer 3D pointclouds data to 2D laser data (yello line in rviz) by using `pointcloud_to_laserscan` , you can refer to the fellowing two packages http://wiki.ros.org/pointcloud*to*laserscan http://wiki.ros.org/velodyne_simulator

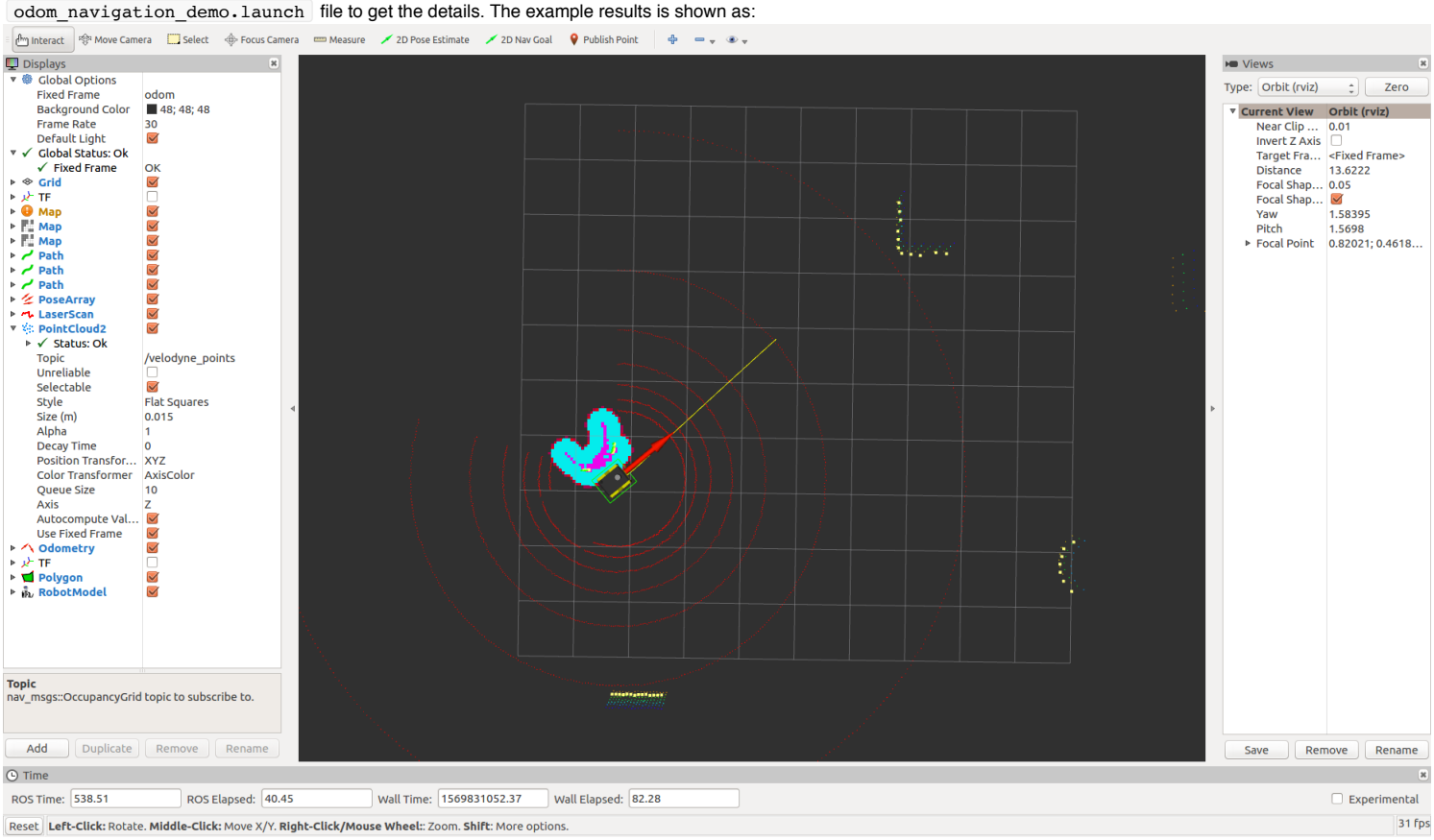
Up to here, you could use the simulate robot to finish a navigation task.

PS: you can refer to the official tutorials of Jackal Robot Simulation and Navigation: https://www.clearpathrobotics.com/assets/guides/jackal/index.html#

Navigation through move_base packages

In this stage, you need to learn how to use `move_base` package. A 2D navigation stack is provide a navigation framework in ROS and easy to extend your personal works.

Before the next example, you need to understand the fundamental knowledge about `move_base` : http://wiki.ros.org/navigation http://wiki.ros.org/navigation/Tutorials Here, we use the simulate robot in last stage to do 2D navigation. First, launch your simulator; Then, fellow the command to launch the `move_base` : `cd hw3_ws source devel/setup.bash roslaunch jackal_navigation odom_navigation_demo.launch` Here, maybe you need to modify the parameters in `jackal_navigation` package according for your simulator setting, e.g. change the sensor data type, or sensor frame id. You can open the `odom_navigation_demo.launch` file to get the details. The example results is shown as:



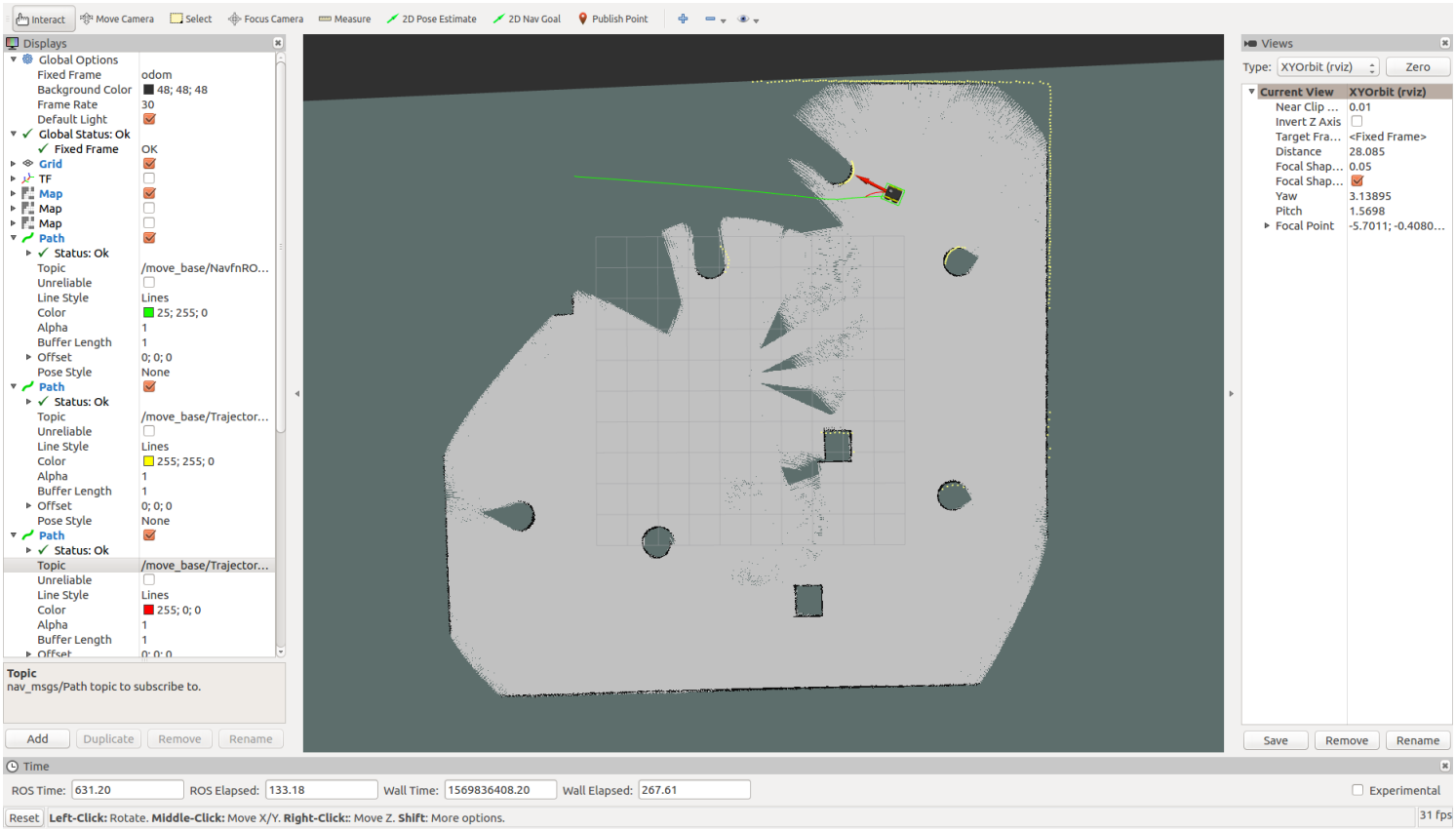
After launch the `move_base` node, you could use `2D Nav Goal` to set a target pose, and robot will generate a collision free path and move the target position.

SLAM with Gmapping package

It is provide a example about how to use the `gmapping` and `move_base` packages to generate a 2D occupancy grid map in the simulator.

Before, maybe you need install and understand the gmapping package: http://wiki.ros.org/slam_gmapping http://wiki.ros.org/gmapping

install command: `sudo apt install ros-kinetic-slam-gmapping sudo apt install ros-kinetic-gmapping` Here, we use same simulate robot in last stage to do 2D navigation. First, launch your simulator; Then, fellow the command to launch the `gmapping` and `move_base` : `cd hw3_ws source devel/setup.bash roslaunch jackal_navigation gmapping_demo.launch` Here, maybe you also need to modify some parameters in `jackal_navigation` package according for your simulator setting, e.g. change the sensor data type, or sensor frame id. You can open the `gmapping_demo.launch` file to get the details. The example results is shown as:



After generating the map, you could use `map_server` package to save the occupancy grid map. You can get the details from website: http://wiki.ros.org/map_server.