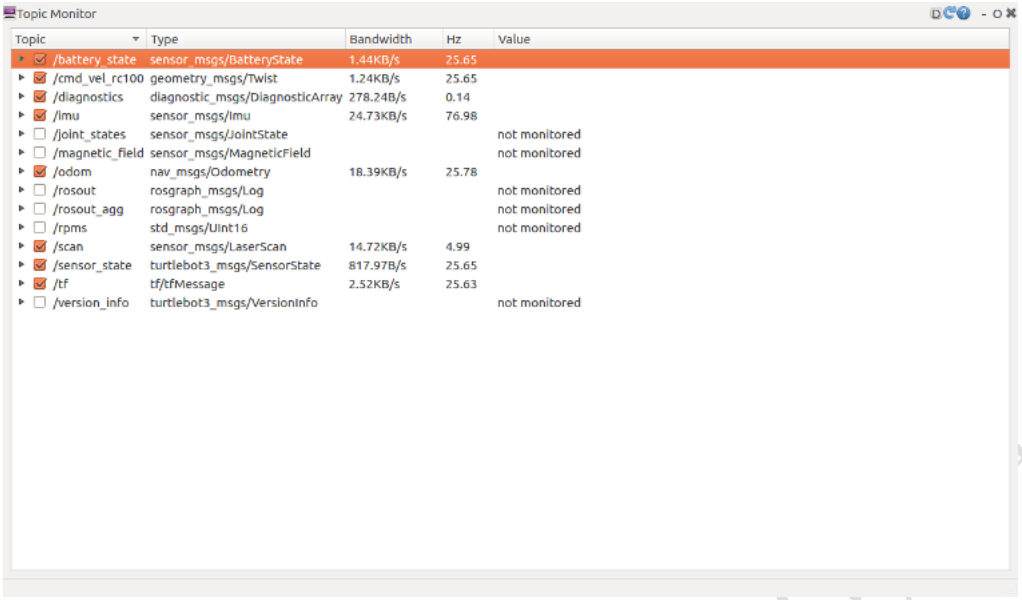
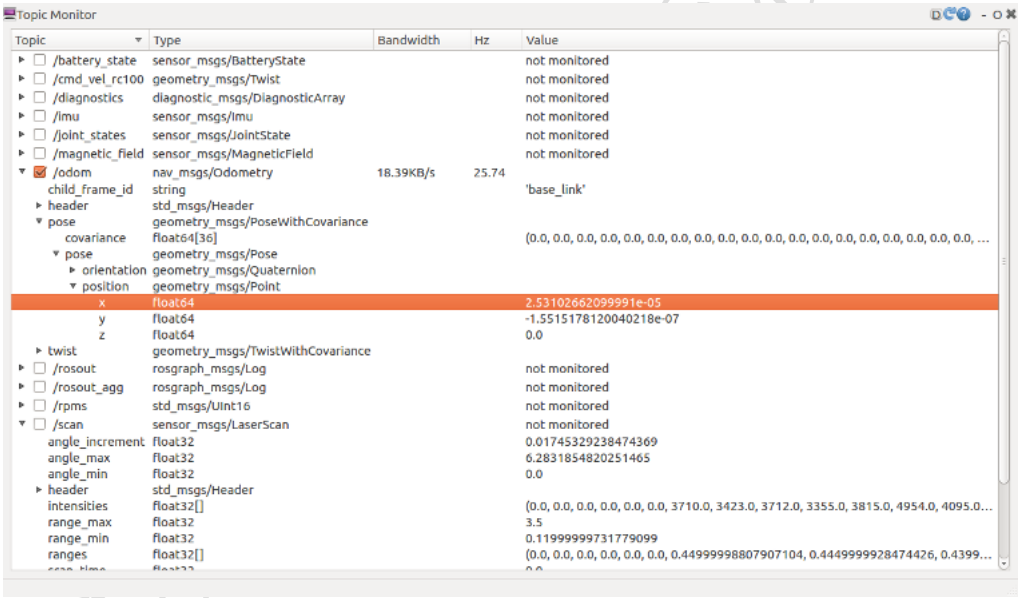


如果要查看更多详细主题消息，请单击▶每个复选框旁边的按钮。



此外，只要添加了主题，就可以通过 rqt 监视主题。

7. 地图构建 SLAM

同步定位与地图构建（SLAM 或 Simultaneous localization and mapping）是一种概念：希望机器人从未知环境的未知地点出发，在运动过程中通过重复观测到的地图特征（比如，墙角，柱子等）定位自身位置和姿态，再根据自身位置增量式的构建地图，从而达到同时定位和地图构建的目的。

SLAM 技术是 TurtleBot3 的著名功能。下面介绍如何通过远程操作创建地图。首先启动

TurtleBot3。

首先测试雷达是否正常

把雷达连接到树莓派上 [TurtleBot3 SBC]给 LiDAR 连接到 ttyUSB0 的权限

$ sudo chmod a+rw /dev/ttyUSB0

运行

$ roslaunch turtlebot3_bringup turtlebot3_lidar.launch

然后新打开一个终端，运行

$ rostopic echo /scan

可以看到激光雷达采集到的数据信息。

运行 SLAM 操作

[Remote PC]运行 roscore

$ roscore

[TurtleBot3 SBC]给 LiDAR 连接到 ttyUSB0 的权限

$ sudo chmod a+rw /dev/ttyUSB0

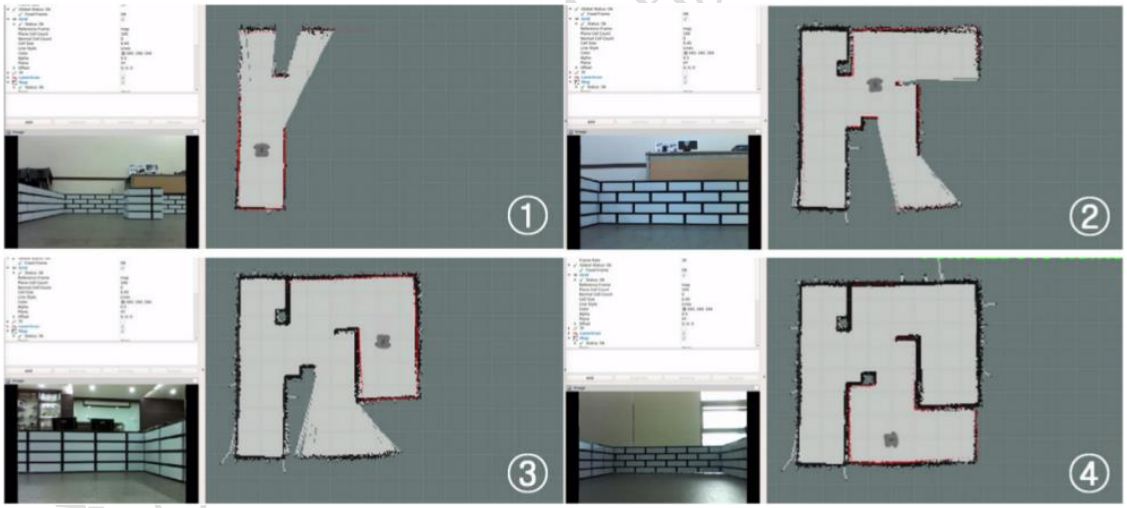
[TurtleBot3 SBC]启动 launch 文件

$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

[Remote PC]打开终端，然后运行 SLAM 启动文件

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping



运行远程操作节点

[Remote PC]打开新终端并运行远程操作节点。以下命令允许用户控制机器人手动执行 SLAM 操作。重要的是避免剧烈运动，例如过快地改变速度或过快地旋转。使用机器人制作地图时，机器人应扫描要测量的环境的每个角落。

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

将地图保存到文件

[Remote PC] 打开终端，然后运行地图保存节点

$ rosrun map_server map_saver -f ~/map

调优指南

该调优指南为您提供了一些配置重要参数的提示。如果要根据您的环境更改性能，此技

巧可能对您有所帮助并节省了时间。

maxUrange

- `turtlebot3_slam/launch/turtlebot3_gmapping.launch`

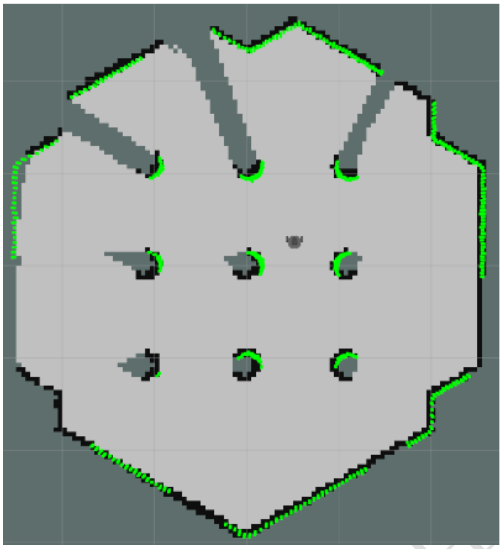
该参数设置了激光雷达传感器的最大可用范围。

map_update_interval

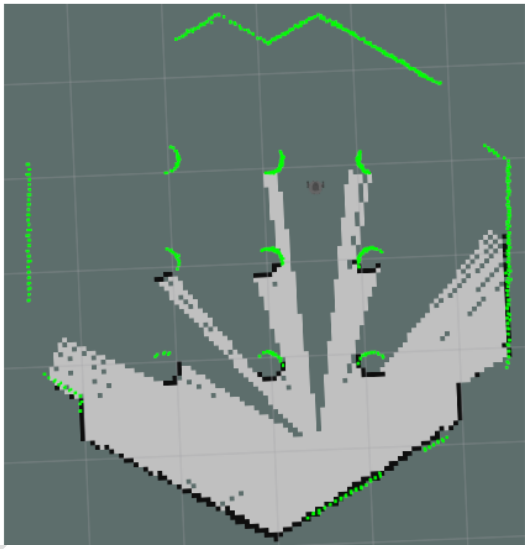
- `turtlebot3_slam/launch/turtlebot3_gmapping.launch`

两次更新之间的时间间隔（以秒为单位）。如果将其设置为较低，则地图将更频繁地更新。但这需要更大的计算量。设置此参数取决于您的环境。

map update interval = 2.0



map update interval = 20.0



minimumScore

- `turtlebot3_slam/launch/turtlebot3_gmapping.launch`

考虑扫描匹配结果的最低分数。该参数可以避免跳跃姿势估计。如果此设置正确，您可以观看以下信息。

```
Average Scan Matching Score=278.965
neff= 100
Registering Scans:Done
update frame 6
update ld=2.95935e-05 ad=0.000302522
Laser Pose= -0.0320253 -5.36882e-06 -3.14142
```

如果此设置太高，则可以观看以下警告。

```
Scan Matching Failed, using odometry. Likelihood=0
lp:-0.0306155 5.75314e-06 -3.14151
op:-0.0306156 5.90277e-06 -3.14151
```

linearUpdate

- `turtlebot3_slam/launch/turtlebot3_gmapping.launch`

机器人平移时，每次都会进行扫描过程。

angularUpdate

- `turtlebot3_slam/launch/turtlebot3_gmapping.launch`

机器人旋转时，每次都会进行扫描。将其设置为小于 linearUpdate 更好。

参考文献

- Gmapping ROS WIKI, Github
- Cartographer ROS WIKI, Github
- Hector ROS WIKI, Github
- Karto ROS WIKI, Github
- frontier_exploration ROS WIKI, Github

8. 自主导航和路径规划

本节介绍如何利用 Turtlebot3 进行导航，导航使机器人进入期望的位置。

导航操作：

[Remote PC]启动导航文件

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml

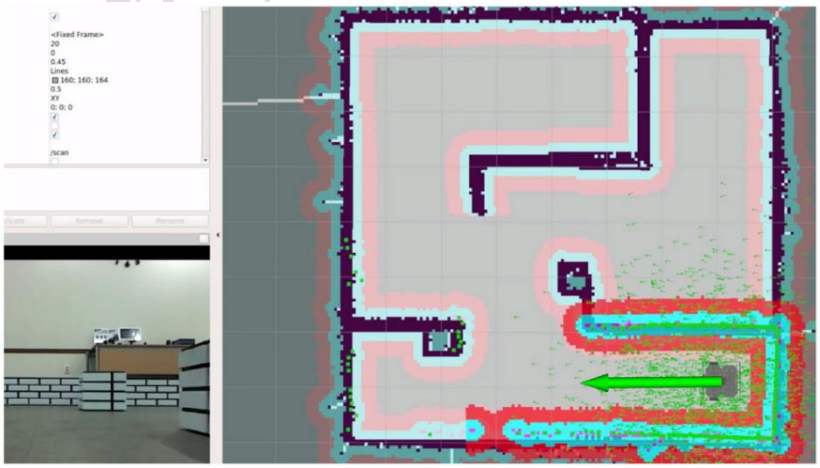
注意：地址请输入本机地图保存位置

提示：当您运行上述命令时，还将执行可视化工具 RViz。如果要单独运行 RViz，请使用以下命令。

$ rviz -d `rospack find turtlebot3_navigation`/rviz/turtlebot3_navigation.rviz

[Remote PC] 在开始导航之前，TurtleBot3 需要估计它的位置和姿势。

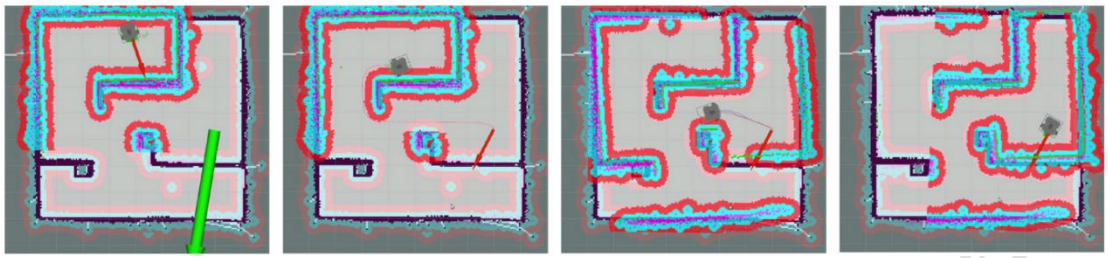
点击 2D Pose Estimate 按钮，通过单击并拖动地图上的方向来设置地图上的大致位置。如果没有定位好，请重复调整 2D Pose Estimate



[Remote PC]一切准备就绪后，让我们尝试通过导航 GUI 进行 move 命令。如果按 2D Nav GoalRViz 菜单中的，则会出现一个非常大的绿色箭头。该绿色箭头是可以指定机器人目的地的标记。箭头的根是机器人的 x 和 y 位置，箭头所指的 theta 方向是机器人的方向。在机器人将要移动的位置单击此箭头，然后将其拖动以设置方向，如下所示。TurtleBot3 发送目标位置：

先点击 2D Nav Goal 按钮，然后点击地图上你想要的 TurtleBot 驱动和拖动方向 TurtleBot 应该指向地方。

机器人将根据地图创建一条路径，以避开障碍物。然后，机器人沿着路径移动。此时，即使突然检测到障碍物，机器人也会避开障碍物而移动到目标点。



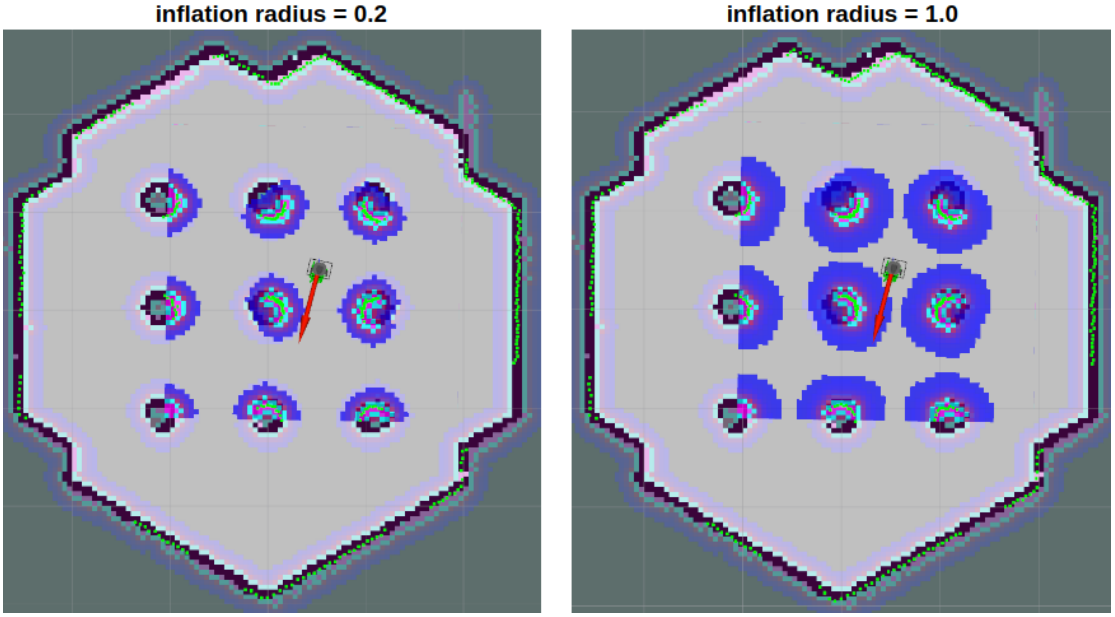
如果无法创建目标位置的路径，则设置目标位置可能会失败。如果希望在到达目标位置之前停止机器人，请将 TurtleBot3 的当前位置设置为目标位置。

调优指南

该调优指南为您提供了一些配置重要参数的提示。如果要根据您的环境更改性能，此技巧可能对您有所帮助并节省了时间。

通货膨胀半径

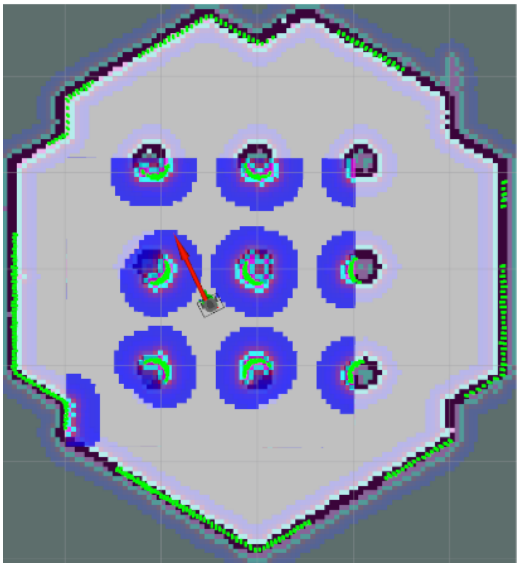
- `turtlebot3_navigation/param/costmap_common_param_$(model).yaml`
- 该参数使膨胀区域脱离障碍物。应规划路径，以使其不跨越该区域。将其设置为大于机器人半径是安全的。



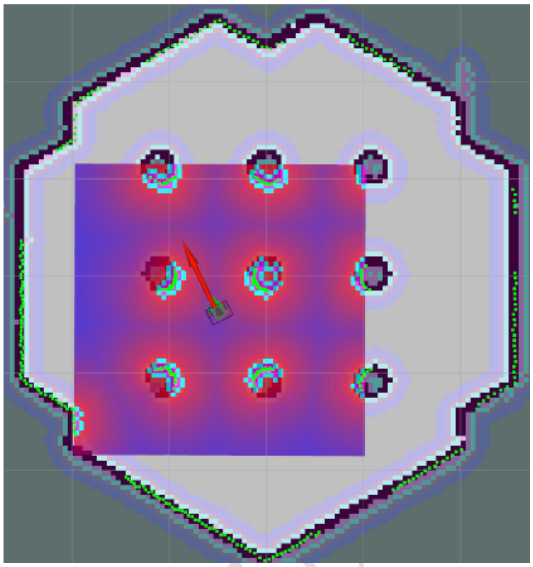
cost_scaling_factor

- `turtlebot3_navigation/param/costmap_common_param_$(model).yaml`
- 该系数乘以成本值。因为是倒数，所以增加了此参数，降低了成本。

cost scaling factor = 20.0



cost scaling factor = 2.0



- 机器人的最佳路径是穿过障碍物之间的中心。将此因子设置得较小，以便远离障碍物。

max_vel_x

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 该因子设置为平移速度的最大值。

min_vel_x

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 该因子设置为平移速度的最小值。如果将此值设置为负，则机器人可以向后移动。

max_trans_vel

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 最大平移速度的实际值。机器人不能比这快。

min_trans_vel

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 最小平移速度的实际值。机器人不能比这慢。

max_rot_vel

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 最大转速的实际值。机器人不能比这快。

min_rot_vel

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 最小转速的实际值。机器人不能比这慢。

acc_lim_x

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 平移加速度极限的实际值。

acc_lim_theta

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 旋转加速度极限的实际值。

xy_goal_tolerance

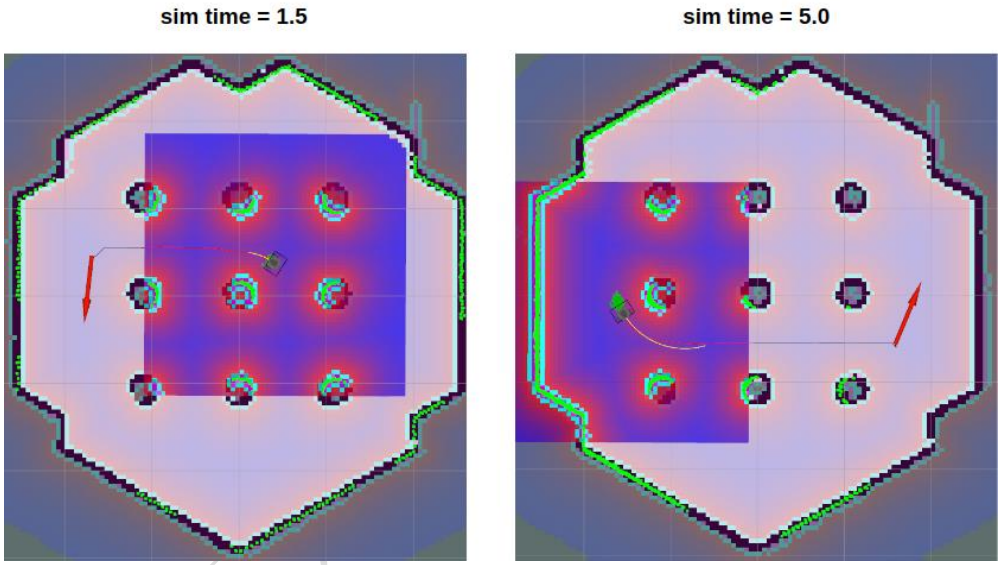
- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 机器人达到目标姿势时允许的 x，y 距离。

偏航目标

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 机器人达到目标姿势时允许的偏航角。

sim_time

- `turtlebot3_navigation/param/dwa_local_planner_params_$(model).yaml`
- 该因子以秒为单位设置正向仿真。值太低意味着无法通过狭窄区域，而值太高则不允许快速旋转。您可以在下图中看到黄线长度的差异。



参考文献

- 基本导航调整指南（ROS Wiki）
- 郑凯宇的 ROS 导航调整指南

9. Turtlebot3 ROS 仿真

Gazebo 是一个自主机器人 3D 仿真环境。 它可以与 ROS 配套用于完整的机器人仿真，也可以单独使用。接下来介绍 Turtlebot3 在 Gazeboz 中的应用。

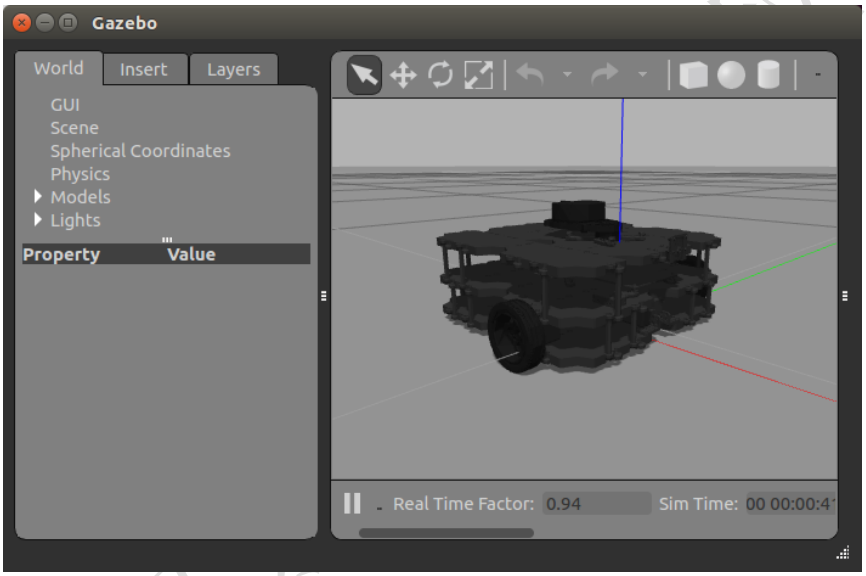
9.1 安装测试

下载 turtlebot3 的仿真包，并编译：

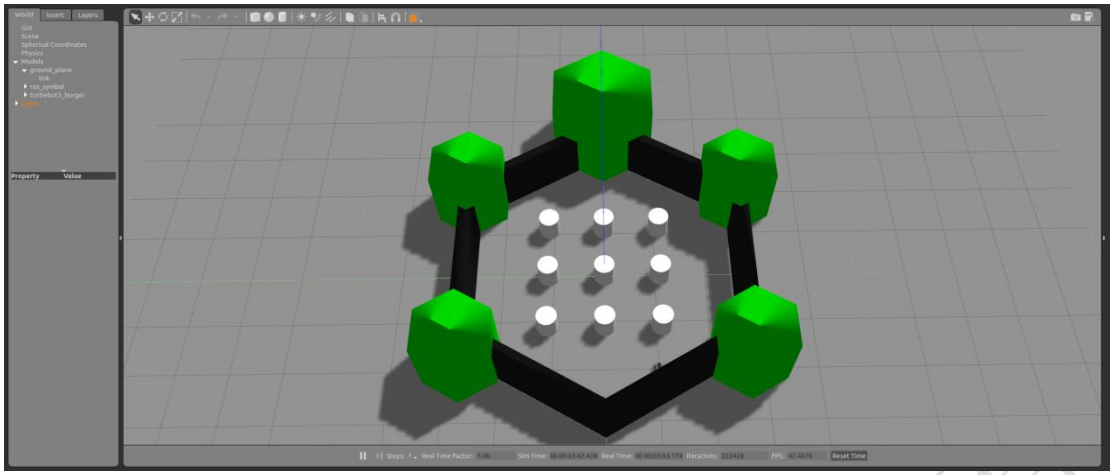
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
试运行：
$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_fake turtlebot3_fake.launch
远程控制移动:
$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

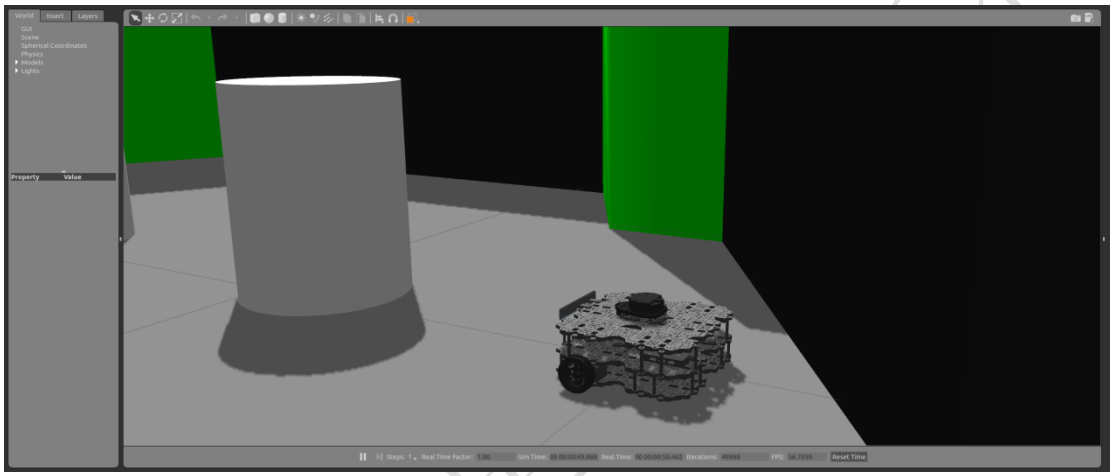
9.2 Gazebo 仿真

启动 Gazebo 仿真器：
$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch



仿真 SLAM
接下来使用 gazebo 仿真 Turtlebot3 SALM 地图构建：
开启仿真器：
$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch



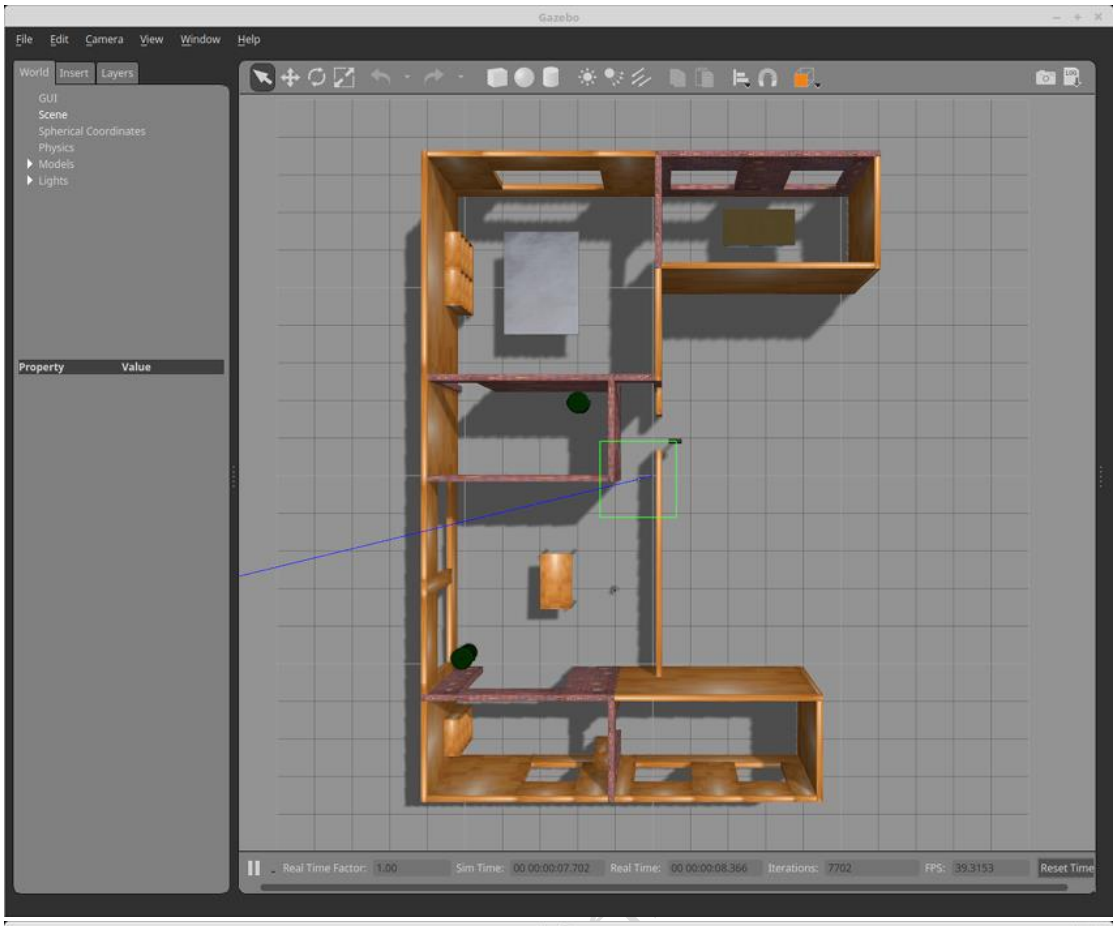


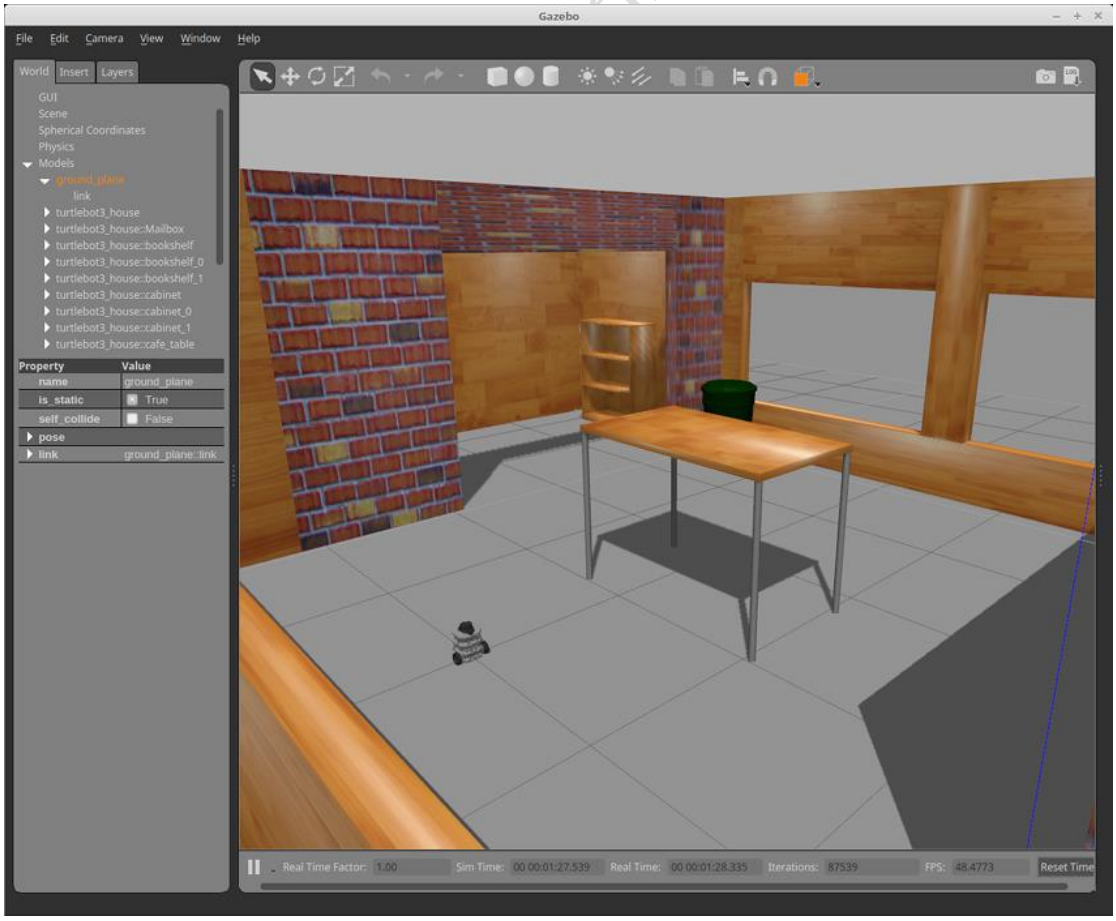
TurtleBot3 之家

`TurtleBot3 House`是用房屋图纸制成的地图。它适用于与更复杂的任务性能相关的测试。

```
$ export TURTLEBOT3_MODEL=${TB3_MODEL}
$ roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

注意：如果`TurtleBot3 House`是首次使用，则根据下载速度，下载地图文件将花费几分钟或更长时间。


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Models
Lights
Property Value
Real Time Factor: 1.00
Sim Time: 00 00:00:07.702
Real Time: 00 00:00:08.366
Iterations: 7702
FPS: 39.3153
Reset Time


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Models
ground_plane
link
turtlebot3_house
turtlebot3_house::Mailbox
turtlebot3_house::bookshelf
turtlebot3_house::bookshelf_0
turtlebot3_house::bookshelf_1
turtlebot3_house::cabinet
turtlebot3_house::cabinet_0
turtlebot3_house::cabinet_1
turtlebot3_house::cafe_table
Property Value
name ground_plane
is_static True
self_collide False
pose
link ground_plane::link
Real Time Factor: 1.00
Sim Time: 00 00:01:27.539
Real Time: 00 00:01:28.335
Iterations: 87539
FPS: 48.4773
Reset Time

驾驶 TurtleBot3

1）gazebo 上的遥控

为了使用键盘控制 TurtleBot3，请在新的终端窗口中使用以下命令启动遥控功能。

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

2）避免碰撞

为了在 TurtleBot3 世界中自动驱动 TurtleBot3 ，请打开一个新的终端窗口，然后输入以下命令。

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

打开一个新的终端窗口，然后输入以下命令。

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

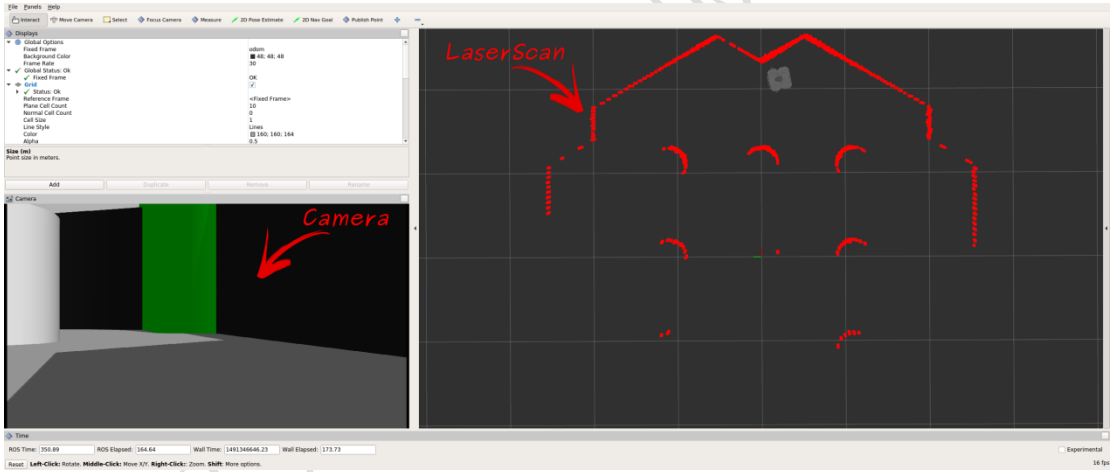
$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch

执行 RViz

在运行模拟时，RViz 可视化已发布的主题。您可以通过输入以下命令在新的终端窗口中启动 RViz。

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch



带有 TurtleBot3 的虚拟 SLAM

对于 Gazebo 中的虚拟 SLAM，您可以选择上面提到的各种环境和机器人模型，而不是运行实际的机器人，并且与 SLAM 相关的命令将使用 SLAM 部分中使用的 ROS 软件包。

虚拟 SLAM 执行程序

以下命令是使用 TurtleBot3 Waffle Pi 模型和 turtlebot3_world 环境的示例。

启动 Gazebo

$ export TURTLEBOT3_MODEL=waffle_pi

$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

启动 SLAM

$ export TURTLEBOT3_MODEL=waffle_pi

$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping

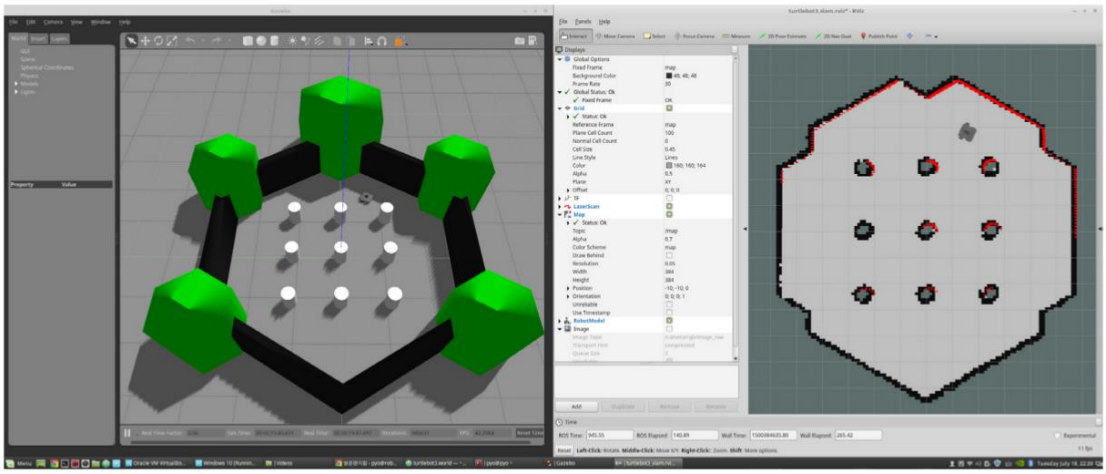
远程控制 TurtleBot3

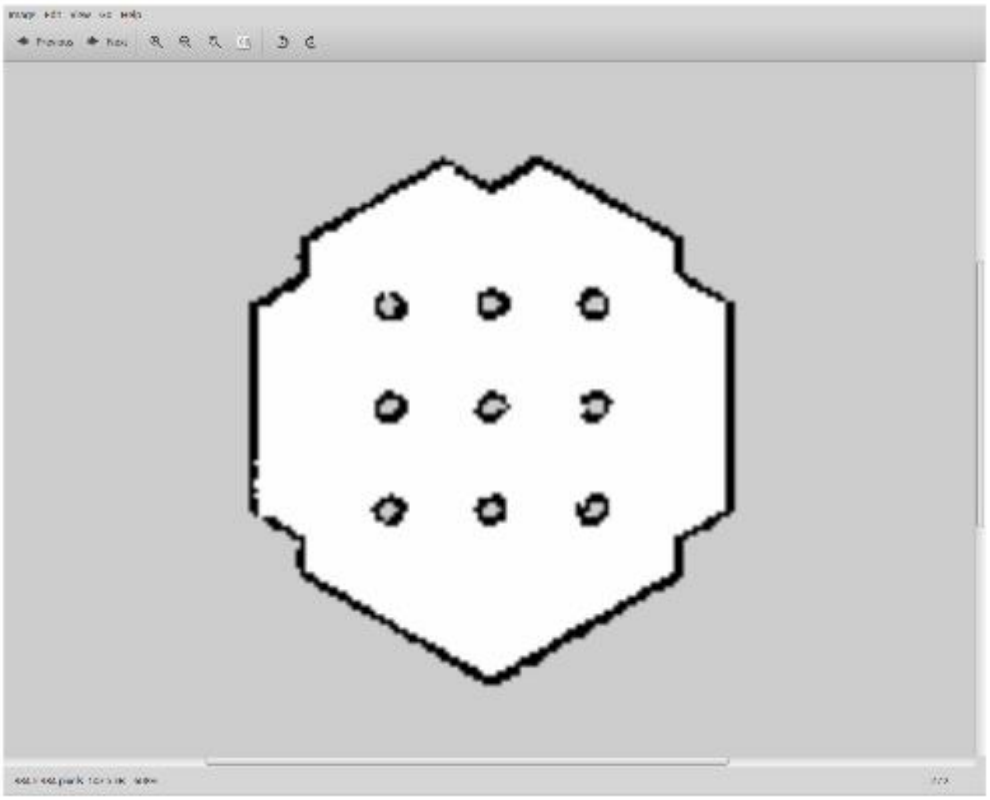
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

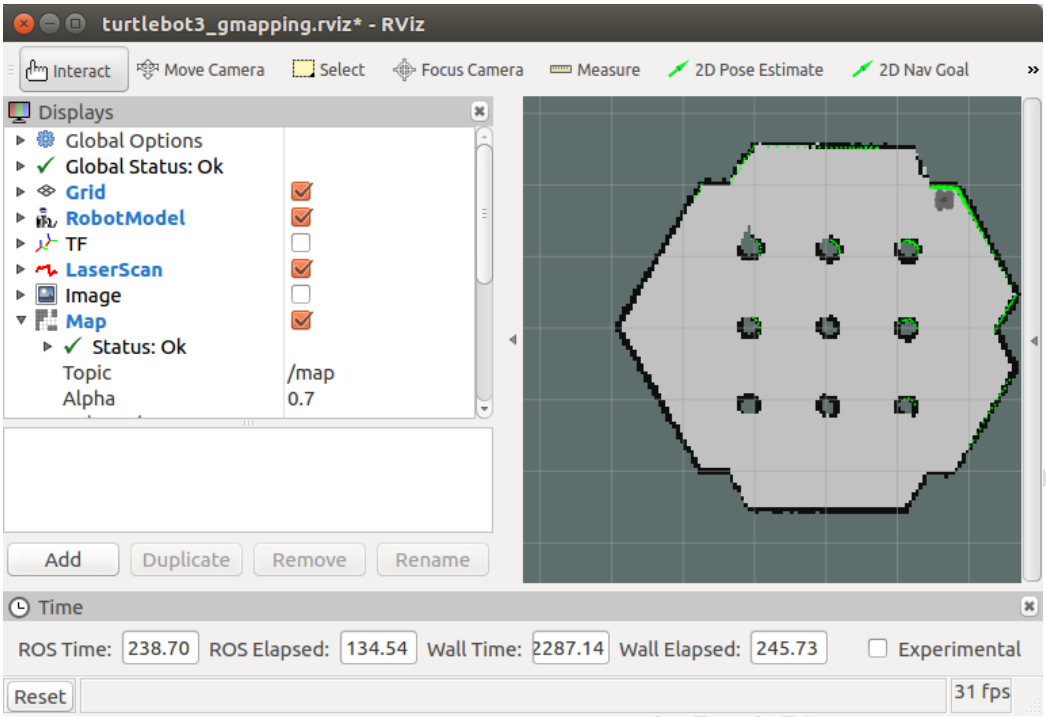
保存地图

$ rosrun map_server map_saver -f ~/map

运行相关程序包并在虚拟空间中移动机器人并创建如下图所示的地图时，可以如下图所示创建地图。







我们可以在我们的 HOME 文件夹下找到刚刚构建的地图 map.pgm 和地图配置文件 map.yaml

现在我们已经构建好了地图，接下来，用已经构建好地图进行路径规划的仿真

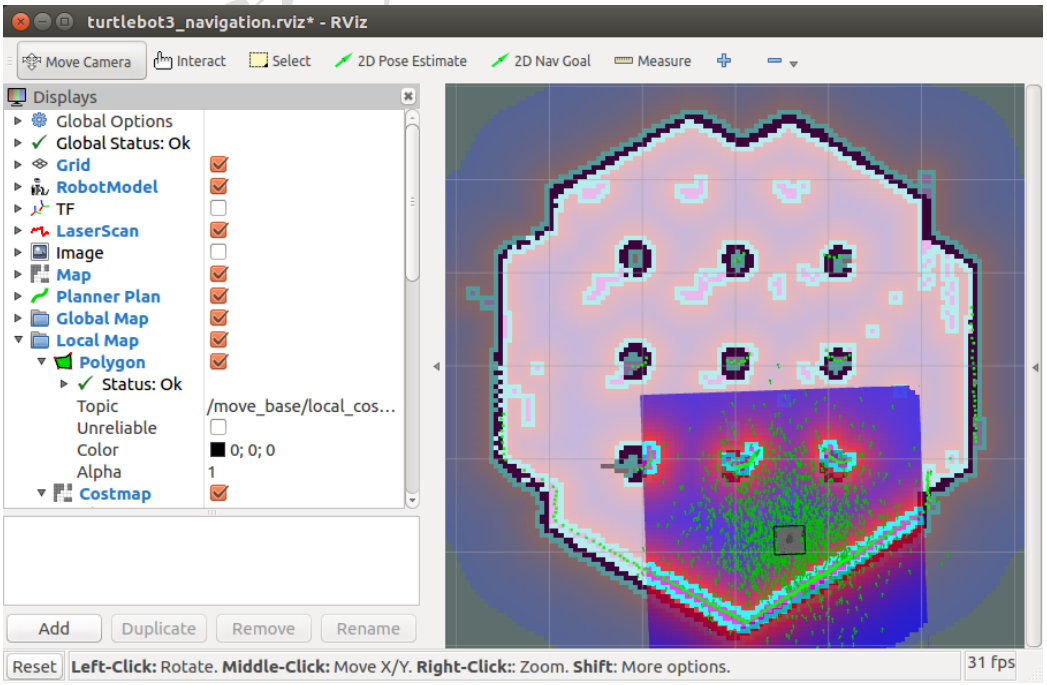
$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

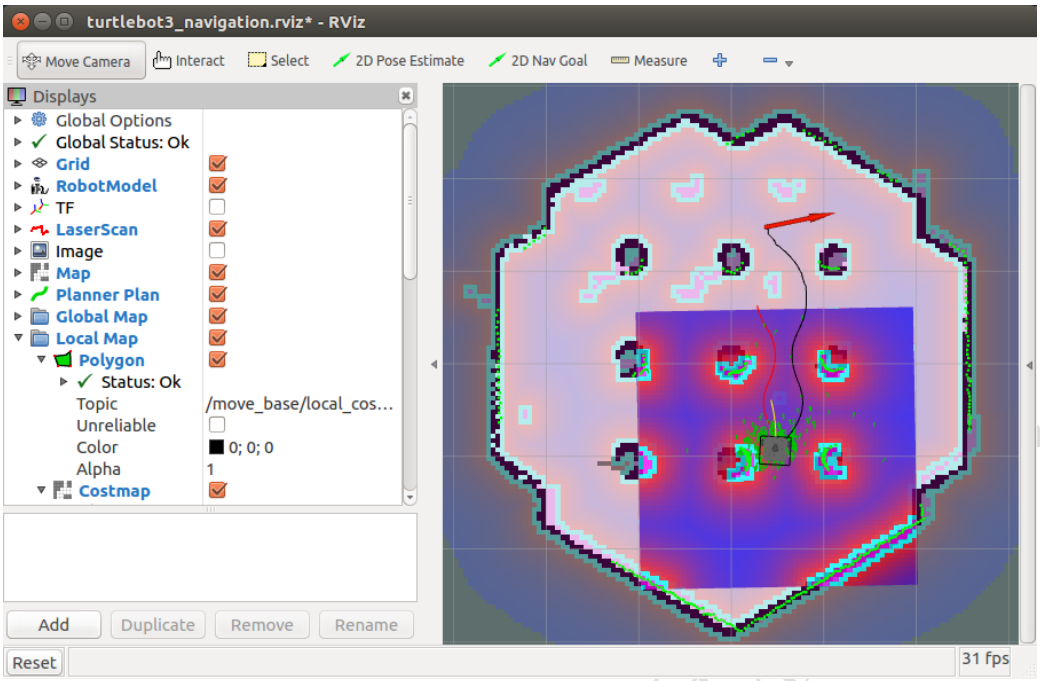
发布 navigation 节点文件。

$ export TURTLEBOT3_MODEL=${TB3_MODEL}

$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml



设定目标点：



如何使用 Gazebo 插件

1）安装 Gazebo7 的库

$ sudo apt-get install libgazebo7-dev

2）从 Github 下载源代码

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_gazebo_plugin

3）在.bashrc 文件中添加凉亭插件路径

$ nano ~/.bashrc

提示：turtlebot3_gazebo_plugin 路径=∽/ turtlebot3_gazebo_plugin

export GAZEBO_PLUGIN_PATH=$GAZEBO_PLUGIN_PATH:${turtlebot3_gazebo_plugin path}/build

export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:${turtlebot3_gazebo_plugin path}/models

4）建造

$ cd ${turtlebot3_gazebo_plugin path}

$ mkdir build

$ cd build

$ cmake ..

$ make

5）执行插件

$ cd ${turtlebot3_gazebo_plugin}

$ gazebo worlds/turtlebot3_${TB3_MODEL}.world