机器人操作系统

机电工程学院

2025 · 秋

课程安排

01 第一章 ROS概述与环境搭建

02 第二章 ROS通信机制

03 第三章 ROS架构与运行管理

04 第四章 ROS常用组件

05 第五章 机器人建模与仿真

06 第六章 ROS进阶功能

第　五　章

机器人建模与仿真

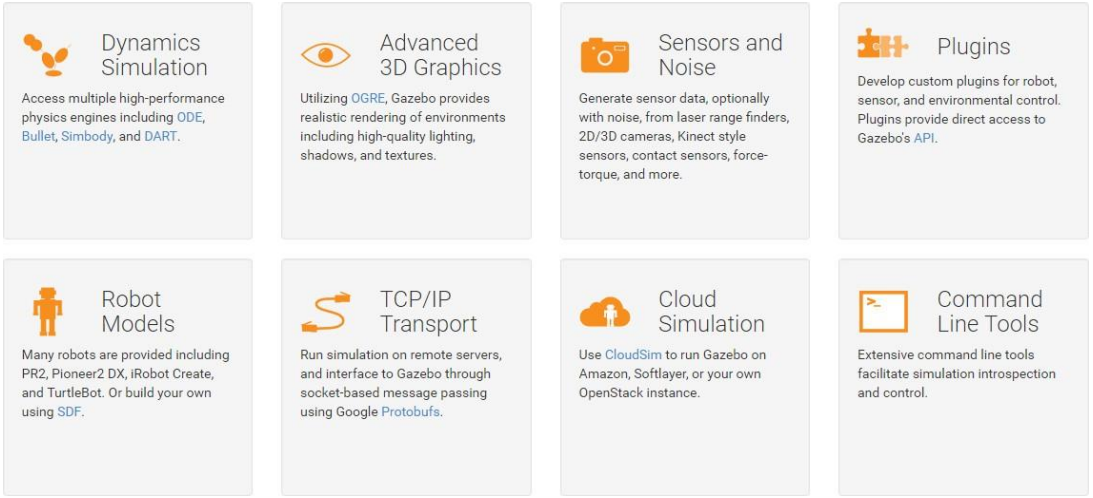
目录

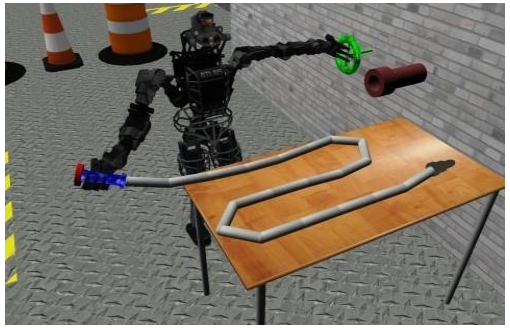
CONTENTS

Gazebo

Gazebo是一款功能强大的三维物理仿真平台：

- 具备强大的物理引擎
- 高质量的图形渲染
- 方便的编程与图形接口
- 开源免费

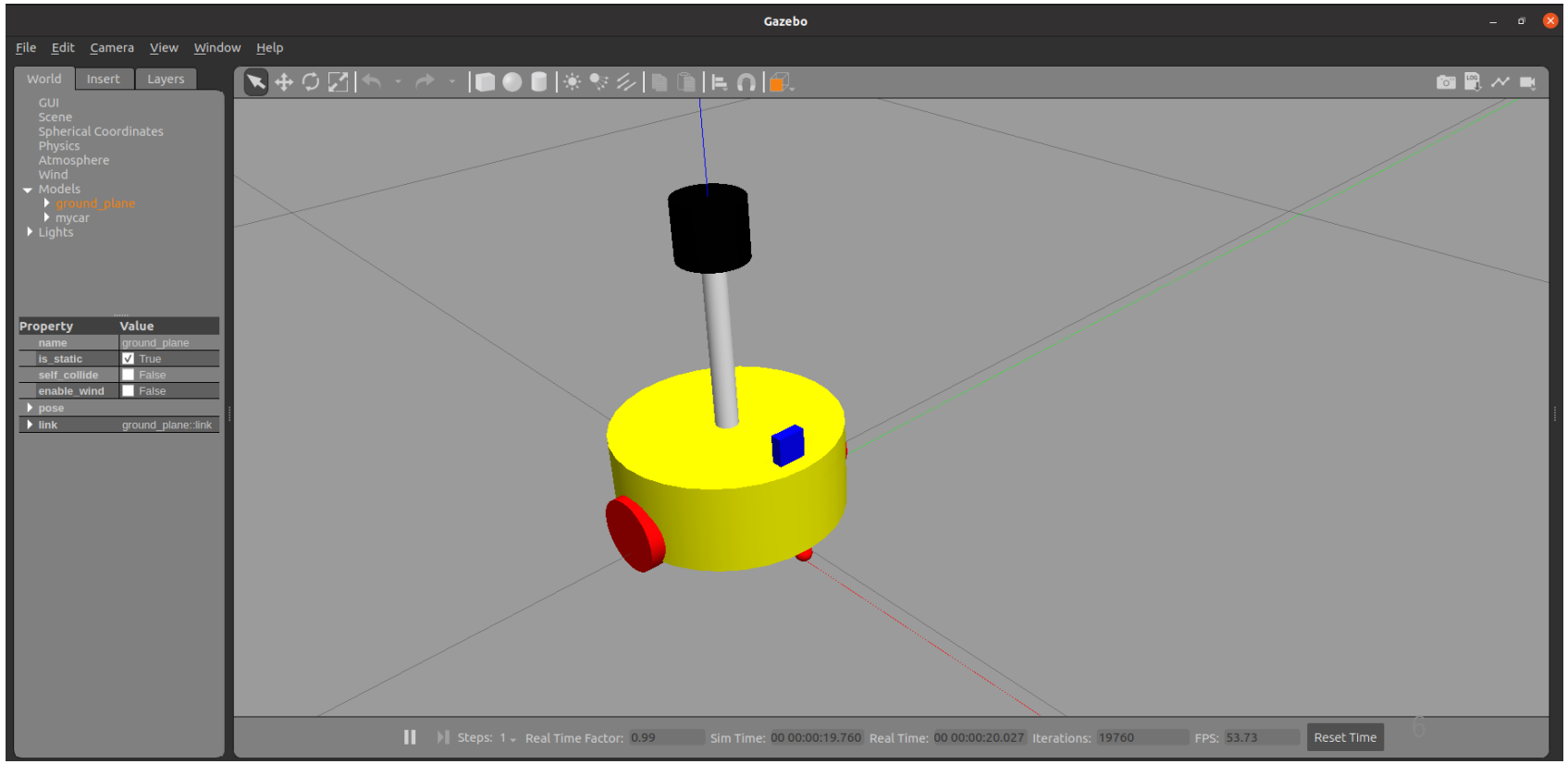




Gazebo

Gazebo（用于搭建仿真环境）：

Gazebo是一款3D动态模拟器，用于显示机器人模型并创建仿真环境,能够在复杂的室内和室外环境中准确有效地模拟机器人。与游戏引擎提供高保真度的视觉模拟类似，Gazebo提供高保真度的物理模拟，其提供一整套传感器模型，以及对用户和程序非常友好的交互方式。



运行使用命令:

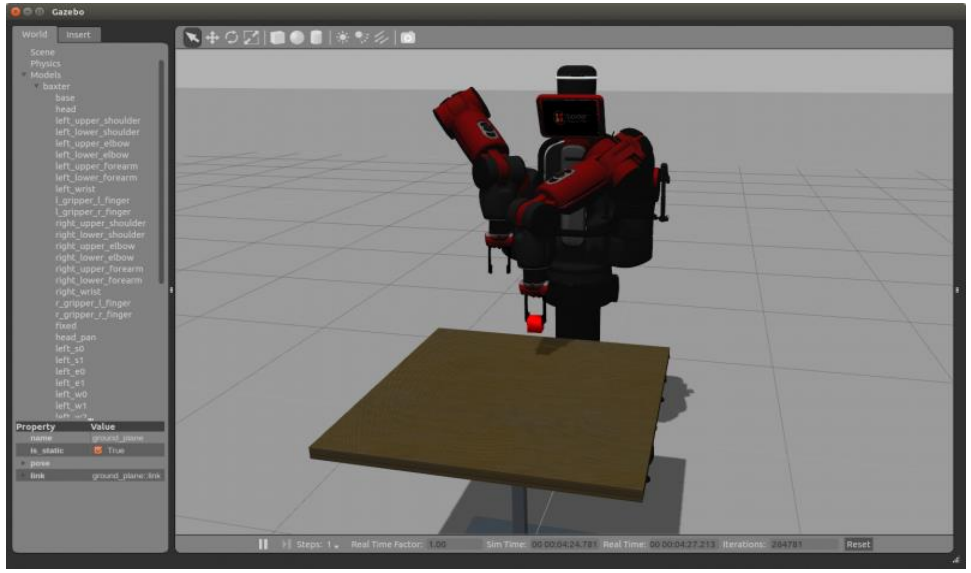
$ gazebo

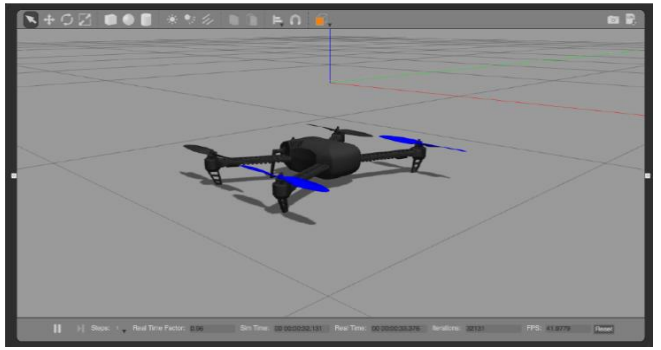
或

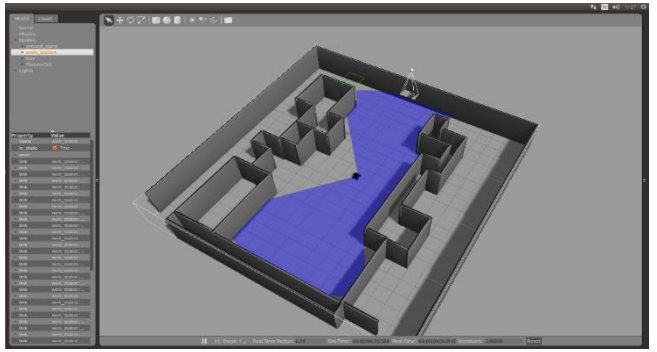
$ rosrun gazebo_ros gazebo

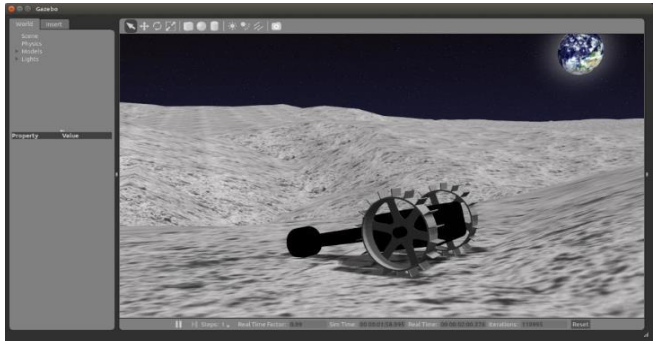
Gazebo的典型应用场景包括：

- 测试机器人算法
- 机器人的设计
- 现实情景下的回溯测试





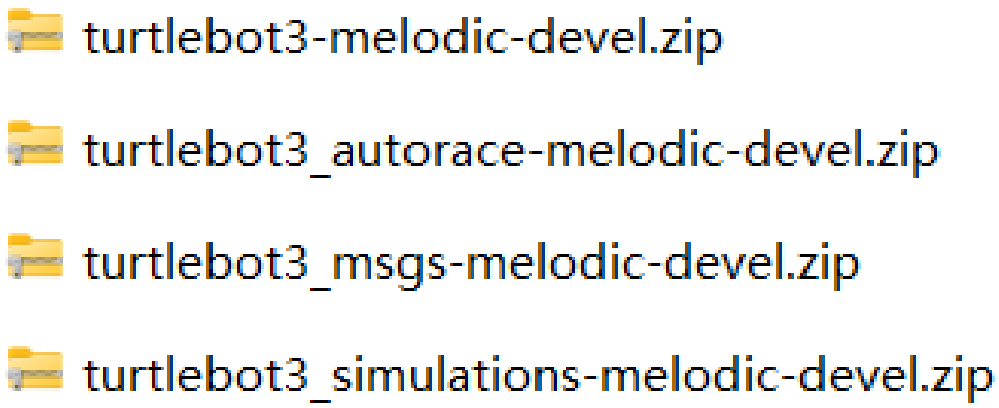




Turtlebot3 仿真

Turtlebot3安装

- 无法完成gitclone
- 下载code，需要匹配自己的ROS版本



- Setting up Turtlebot3 — TechX2019 Robotics Course Documentation documentation
- 将上述4个压缩文件拷贝至 yourworkspace/src，并进行提取操作
- 在终端操作 cd ~/yourworkspace/
  【yourworkspace是你自己的工作空间名】

Turtlebot3安装

这条命令的作用是：

1. 从工作空间的 `src` 目录开始扫描 ROS 包。
2. 安装这些包的所有外部依赖项（例如，第三方库和工具），忽略已经存在于 `src` 目录中的包。
3. 自动确认所有安装操作，无需用户确认。

- 在终端操作 rosdep install --from-paths src -i -y

  - `rosdep install`：是 ROS 中用于安装依赖项的命令，`rosdep` 是一个用于管理依赖关系的工具，它根据你所使用的 ROS 包的描述信息（例如 `package.xml` 文件），自动查找并安装缺少的依赖。
  - `--from-paths src`：指定 `src` 目录作为依赖查找的起始路径，`src` 目录通常是你工作空间的源代码目录。ROS 会从这里开始扫描所有的包，并根据每个包的描述文件（`package.xml`）来解析需要的依赖项。
  - `-i` 或 `--ignore-src`：这个选项告诉 `rosdep` 忽略 `src` 目录中的包，因为这些包已经在工作空间内。它只会安装外部的依赖项。
  - `-y` 或 `--yes`：这个选项表示自动确认安装，不需要手动干预。它默认回答"是"来确认安装所有需要的依赖。

Turtlebot3安装依赖项常见问题

- 网络连接问题
  - 你可以尝试使用 ifconfig 或 ip a 命令查看网络接口状态，确保网络接口已正确启用并分配 IP 地址。
  - 如果网络接口激活，可以尝试使用sudo ifconfig ens33 up 或者sudo ip link set ens33 up 【ens33需要替换成你自己的】
  - 如果你的网络接口已经启用但没有分配 IP 地址，可以尝试使用sudo dhclient ens33 该命令会请求 DHCP 服务器分配一个 IP 地址。
  - 启用网络接口并成功获取 IP 地址后，你可以测试网络连接，看看是否能够成功访问外部网站，比如，可以尝试使用ping www.baidu.com

Turtlebot3安装依赖项常见问题

- 软件包缺失问题
  - 尝试使用sudo rosdep update命令，更新本地的 rosdep 数据库。

    1. `rosdep update` 会访问 ROS 相关的网络源，下载并更新可用的 ROS 依赖数据。
    2. 更新完成后，系统就能基于最新的依赖信息执行后续操作，例如安装缺失的依赖。

  - 手动安装依赖项

    比如，sudo apt-get install ros-melodic-map-server ros-melodic-amcl ros-melodic-rosserial-python ros-melodic-joint-state-publisher-gui

Turtlebot3安装

- 在终端操作
  - catkin_make
  - source ~/ yourworkspace/devel/setup.bash
  - rospack profile

```
rospack profile
```

    - 这个命令更新 ROS 包索引。当你在工作空间中添加了新的包时，执行此命令可以确保 ROS 识别并正确访问这些包。它会重新构建必要的环境文件，帮助 ROS 定位到新增的包。

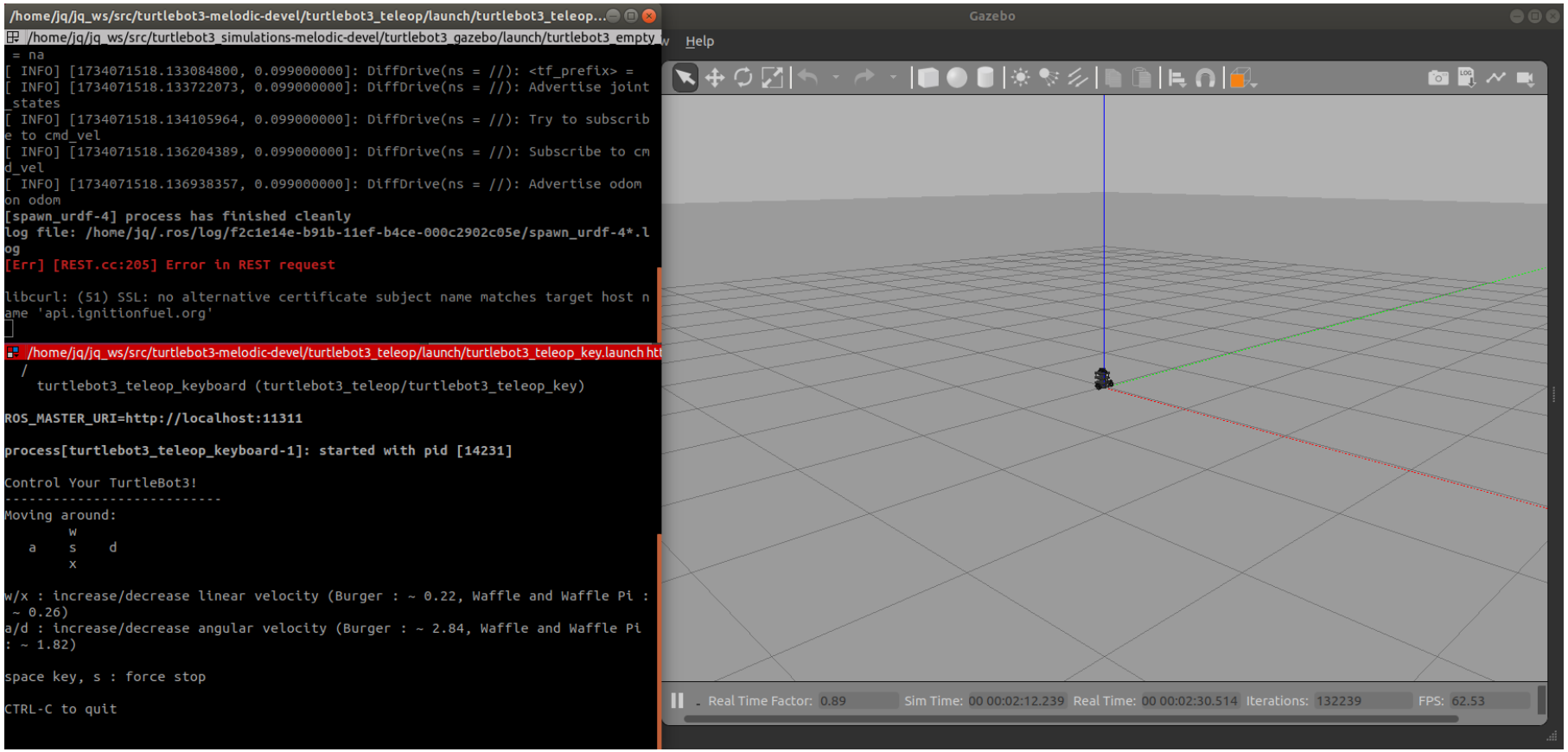
关闭3D加速

- echo "export SVGA_VGPU10=0" >> ~/.bashrc
  - `echo "export SVGA_VGPU10=0"`：这条命令会打印出 `export SVGA_VGPU10=0` 字符串，`export` 用于设置环境变量，使其在当前的 shell 会话中有效。
  - `>> ~/.bashrc`：将上述输出追加到当前用户的 `~/.bashrc` 文件中。`~/.bashrc` 是 Bash shell 的配置文件，每次打开新的终端时，它会自动被执行。通过向该文件追加内容，可以确保在每次打开终端时设置的环境变量都会被加载。
- source ~ /.bashrc
  - `source ~/.bashrc`：这条命令会重新加载 `~/.bashrc` 文件，立即使其中的更改生效。也就是说，在你修改了 `~/.bashrc` 文件后，执行这条命令可以让你无需重新打开终端窗口，就能应用新添加的环境变量。

Turtlebot3示例

用键盘控制小车运动

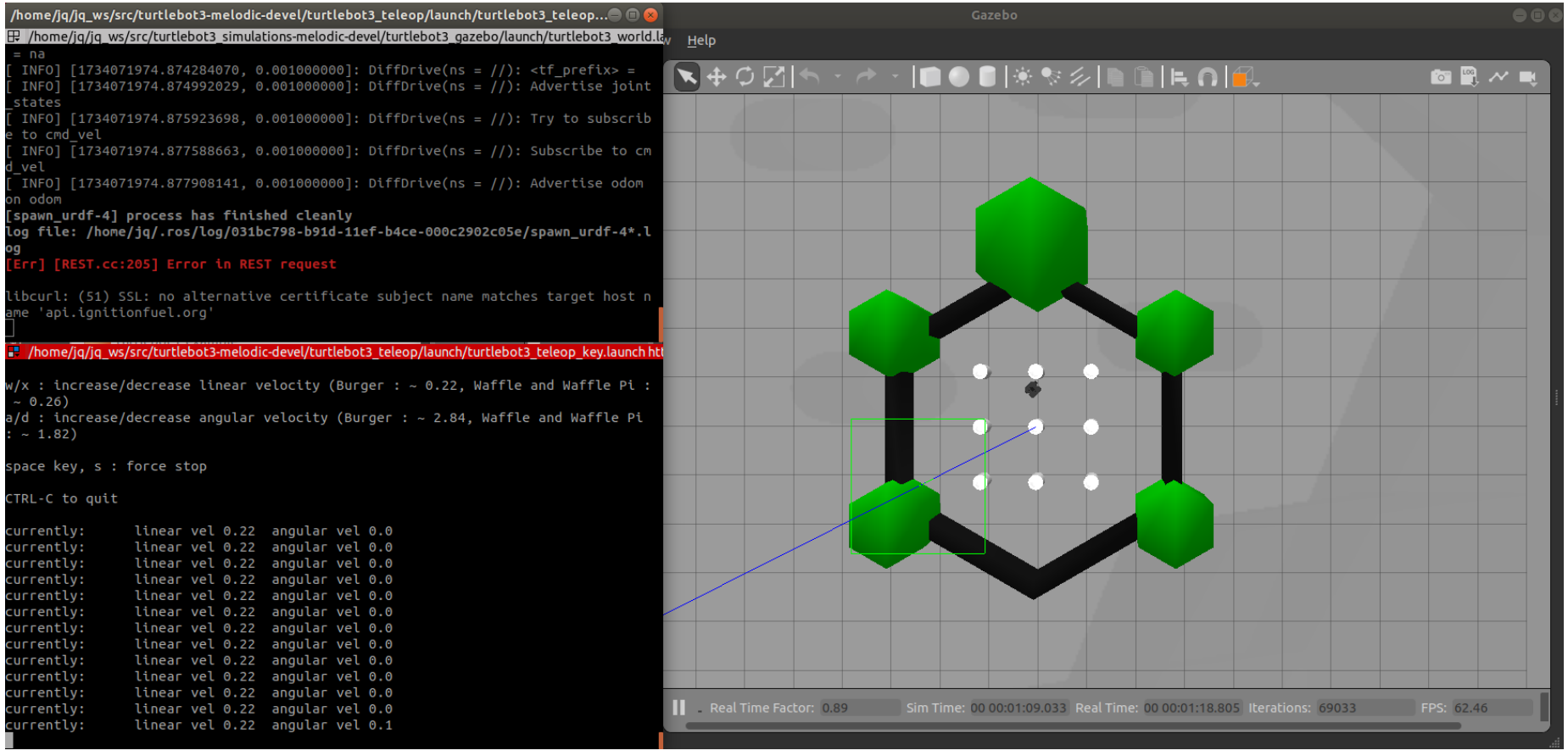
- export TURTLEBOT3_MODEL=burger
- roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
- roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch



Turtlebot3示例

用键盘控制小车运动

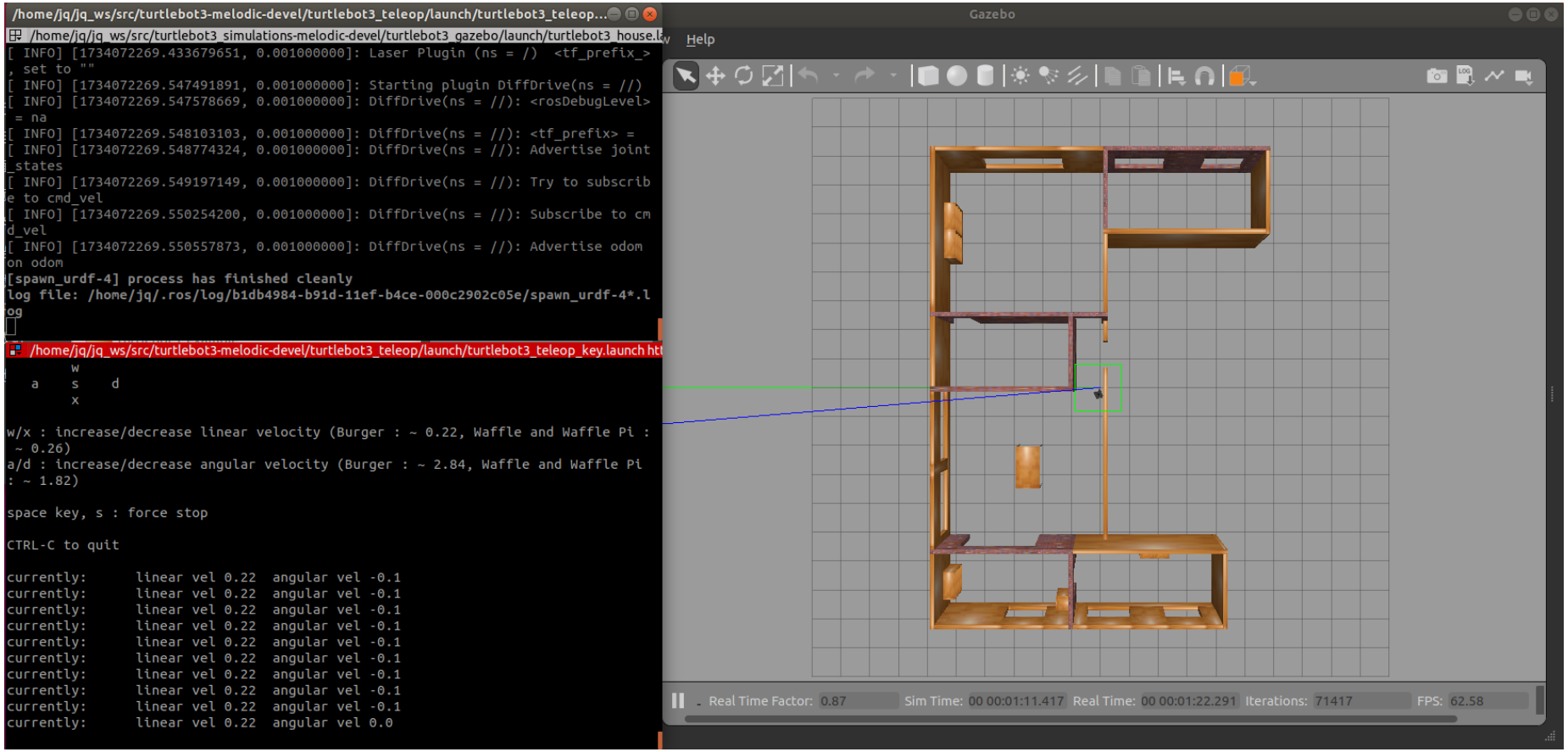
- export TURTLEBOT3_MODEL=waffle
- roslaunch turtlebot3_gazebo turtlebot3_world.launch
- roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch



Turtlebot3示例

用键盘控制小车运动

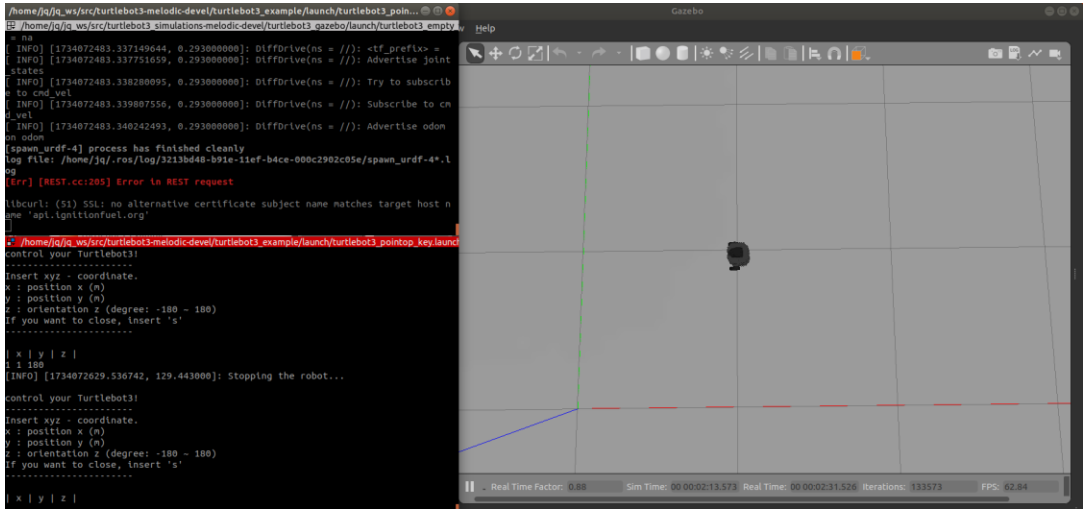
- export TURTLEBOT3_MODEL=waffle_pi
- roslaunch turtlebot3_gazebo turtlebot3_house.launch
- roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch



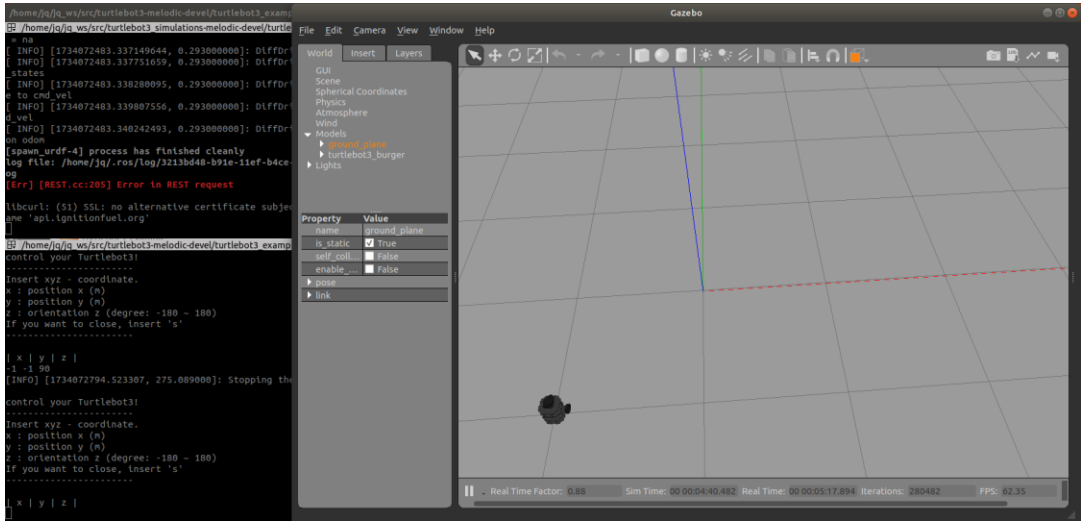
Turtlebot3示例

输入参数控制小车运动
Point operations

- export TURTLEBOT3_MODEL=burger
- roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
- roslaunch turtlebot3_example turtlebot3_pointop_key.launch



1 1 180



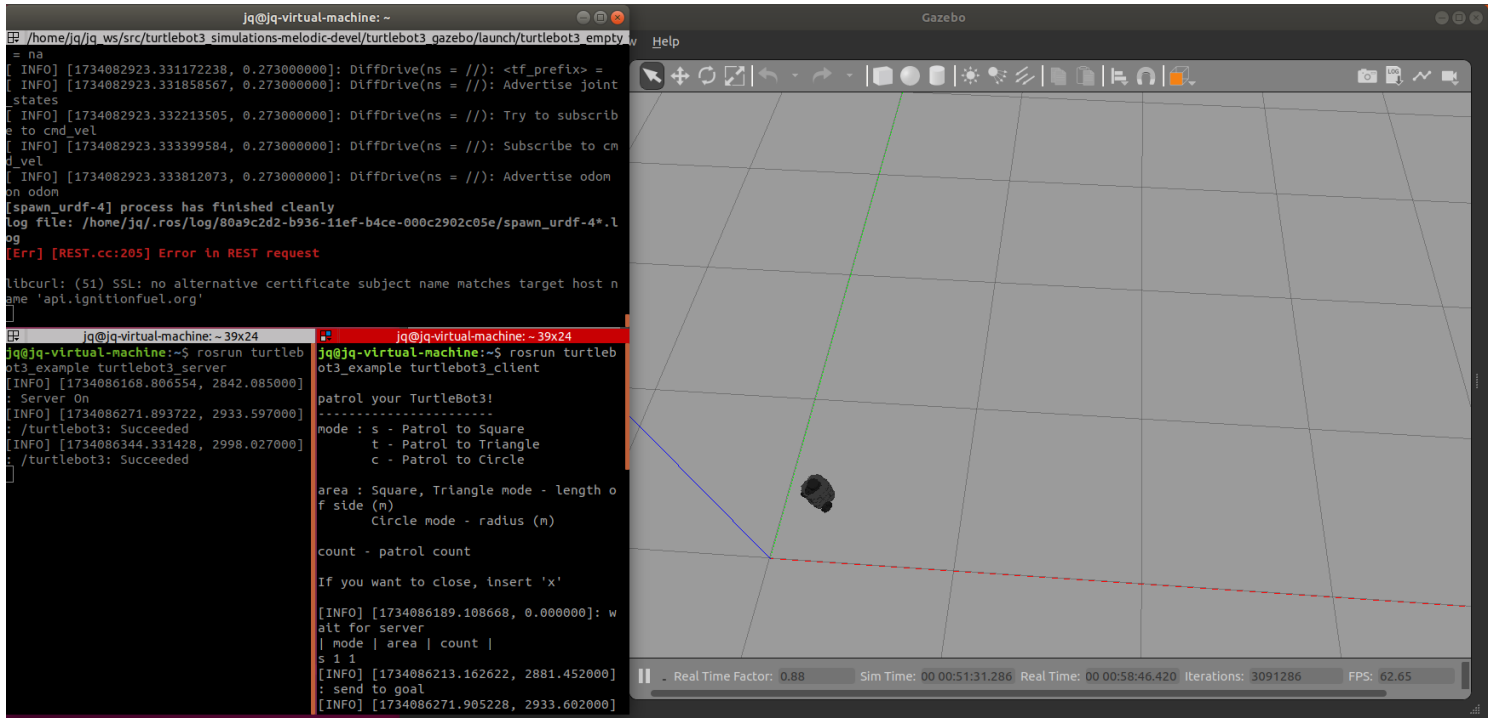
-1 -1 90

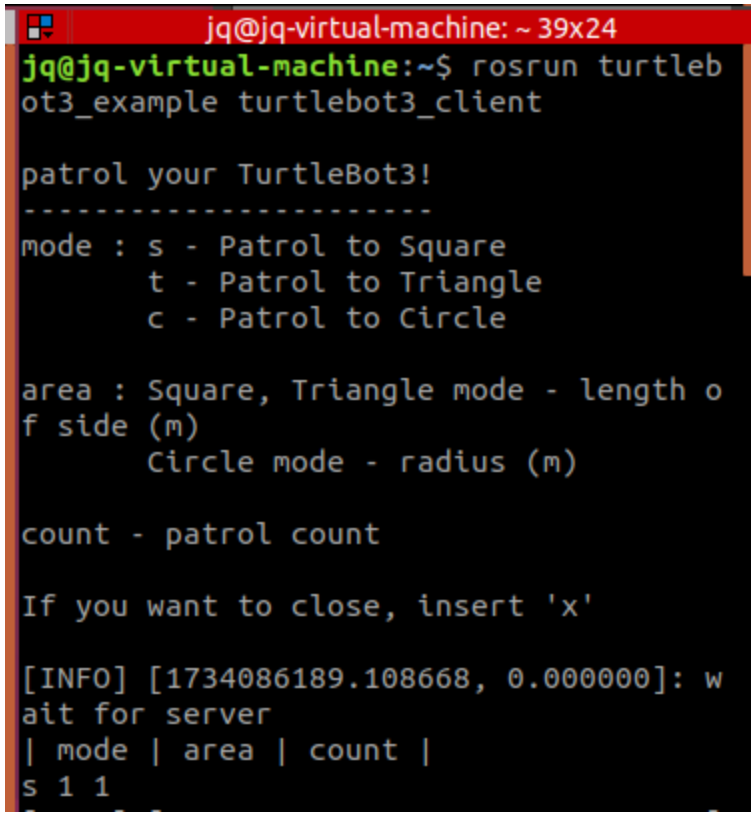
Turtlebot3示例

输入参数控制小车运动

Patrols

- export TURTLEBOT3_MODEL=burger
- roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
- rosrun turtlebot3_example turtlebot3_server
- rosrun turtlebot3_example turtlebot3_client





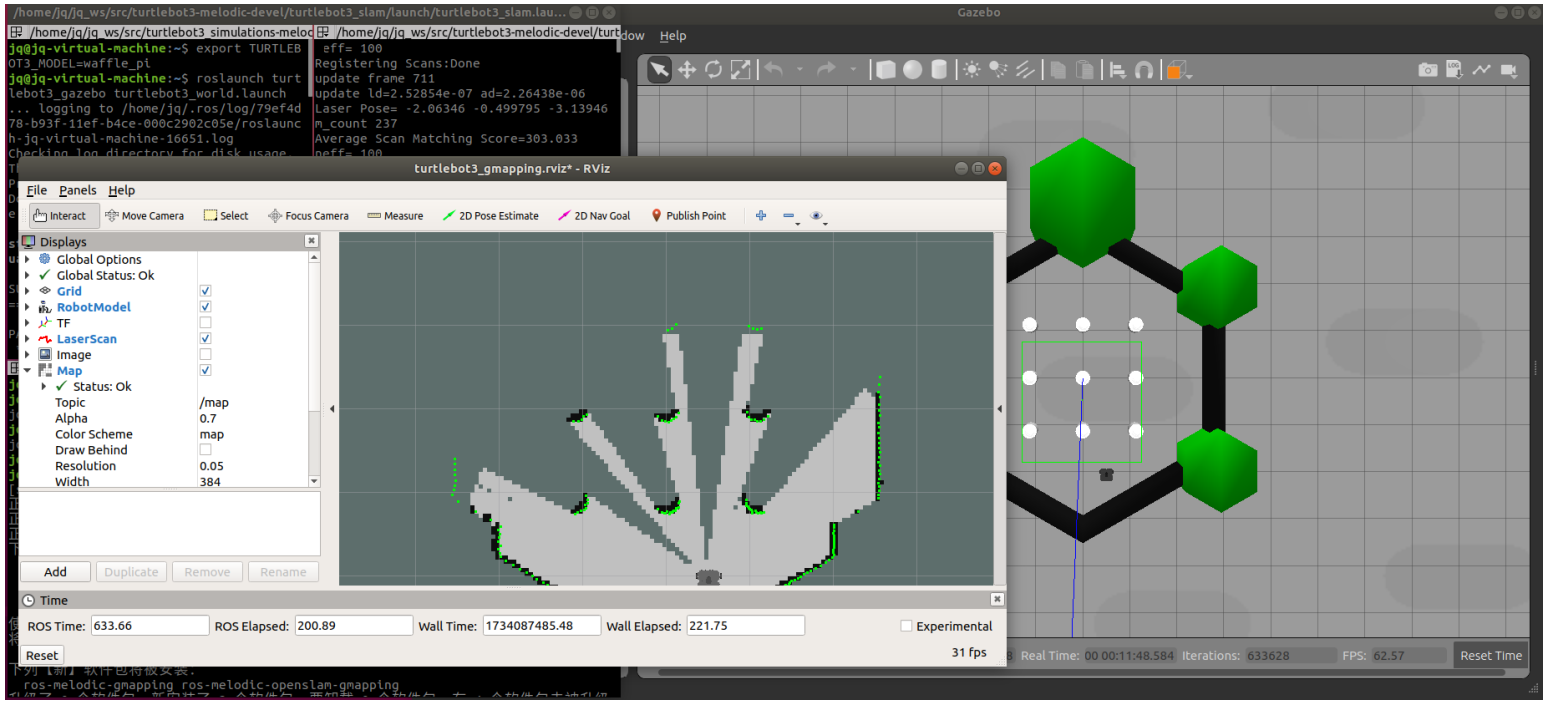
Turtlebot3示例 SLAM

- export TURTLEBOT3_MODEL=waffle_pi
- roslaunch turtlebot3_gazebo turtlebot3_world.launch

终端1

- export TURTLEBOT3_MODEL=waffle_pi
- roslaunch turtlebot3_slam turtlebot3_slam.launch

终端2



ERROR: cannot launch node of type [gmapping/slam_gmapping]: gmapping

1. 检查 gmapping 包是否已安装：

```
rospack list | grep gmapping
```

2. 如果没有显示 gmapping 包：

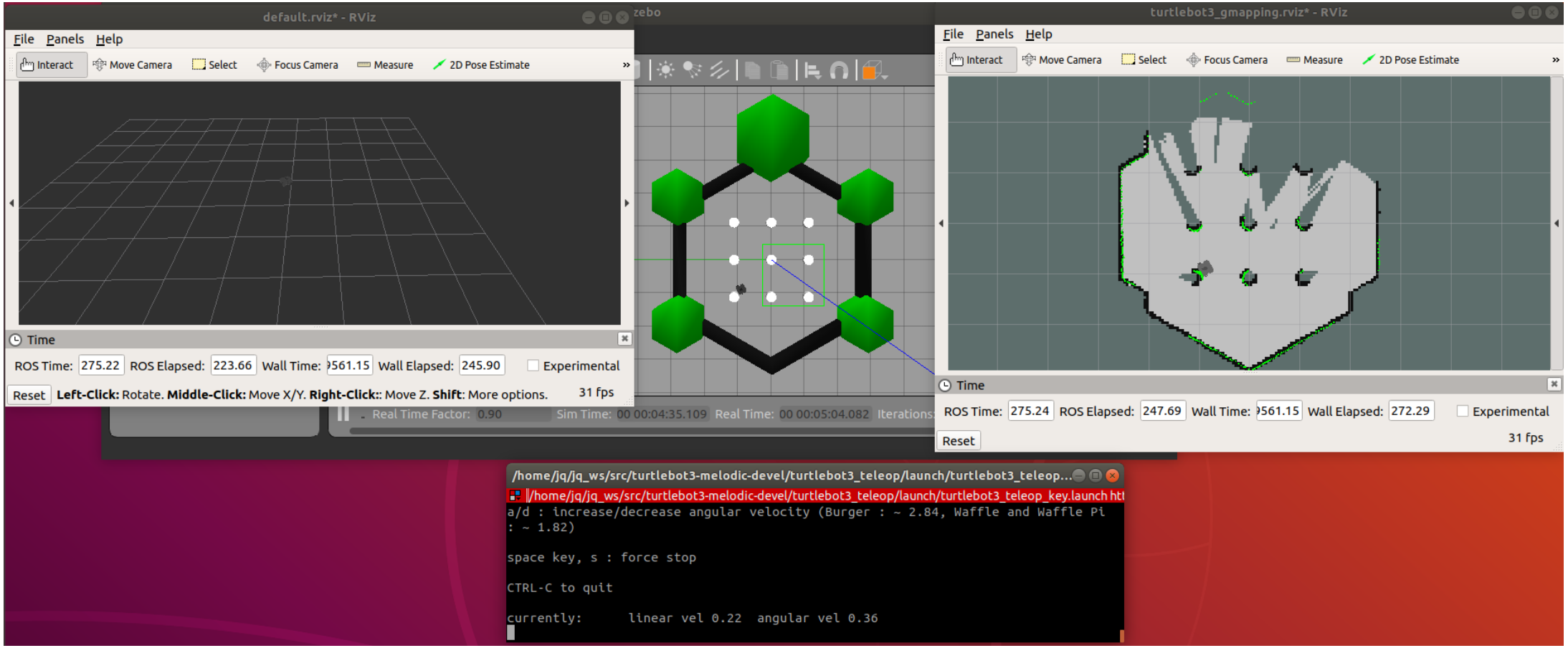
```
sudo apt-get install ros-melodic-gmapping
```

cd ~/jq_ws
catkin_make
source devel/setup.bash

Turtlebot3示例 SLAM

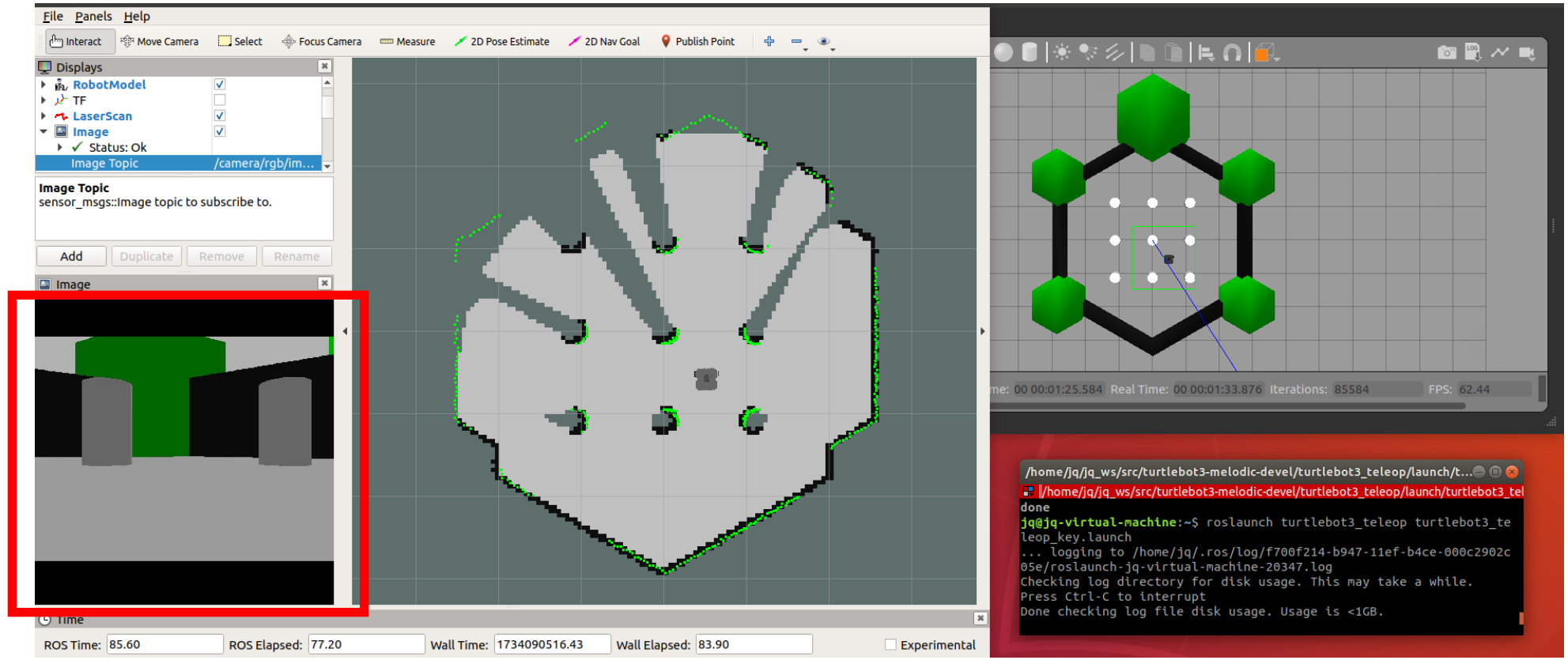
终端3
- rosrun rviz rviz -d "rospack find turtlebot3_sl" /rviz/turtlebot3_slam.rviz

终端4
- roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch



Turtlebot3示例 SLAM

- 在 RViz 左侧的 `Displays` 区域，找到 `Image` 显示项（如果没有，请点击 `Add` 添加一个 `Image` 类型的显示项）。
- 在 `Image` 的配置选项中，将 `Topic` 设置为 `/camera/rgb/image_raw`。



Turtlebot3示例 Navigation TurtleBot3

终端1

- export TURTLEBOT3_MODEL=waffle_pi
- roslaunch turtlebot3_gazebo turtlebot3_world.launch

终端2

- export TURTLEBOT3_MODEL=waffle_pi
- export MAP_FILE=/home/jq/jq_ws/src/turtlebot3-melodic-devel/turtlebot3_navigation/maps/map.yaml
- roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$MAP_FILE

终端3

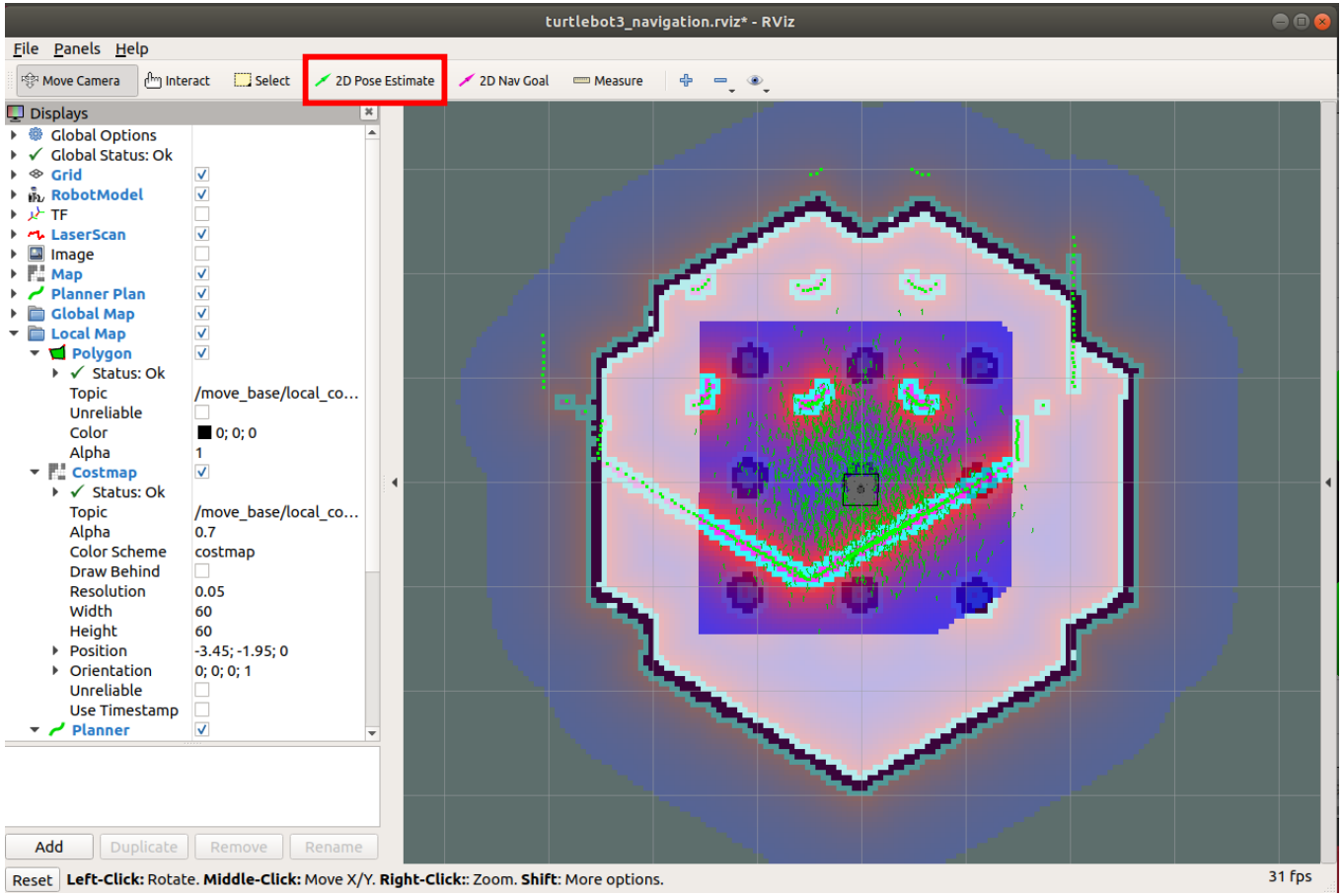
- rosrun rviz rviz -d "rospack find turtlebot3_navigation" /rviz/turtlebot3_nav.rviz

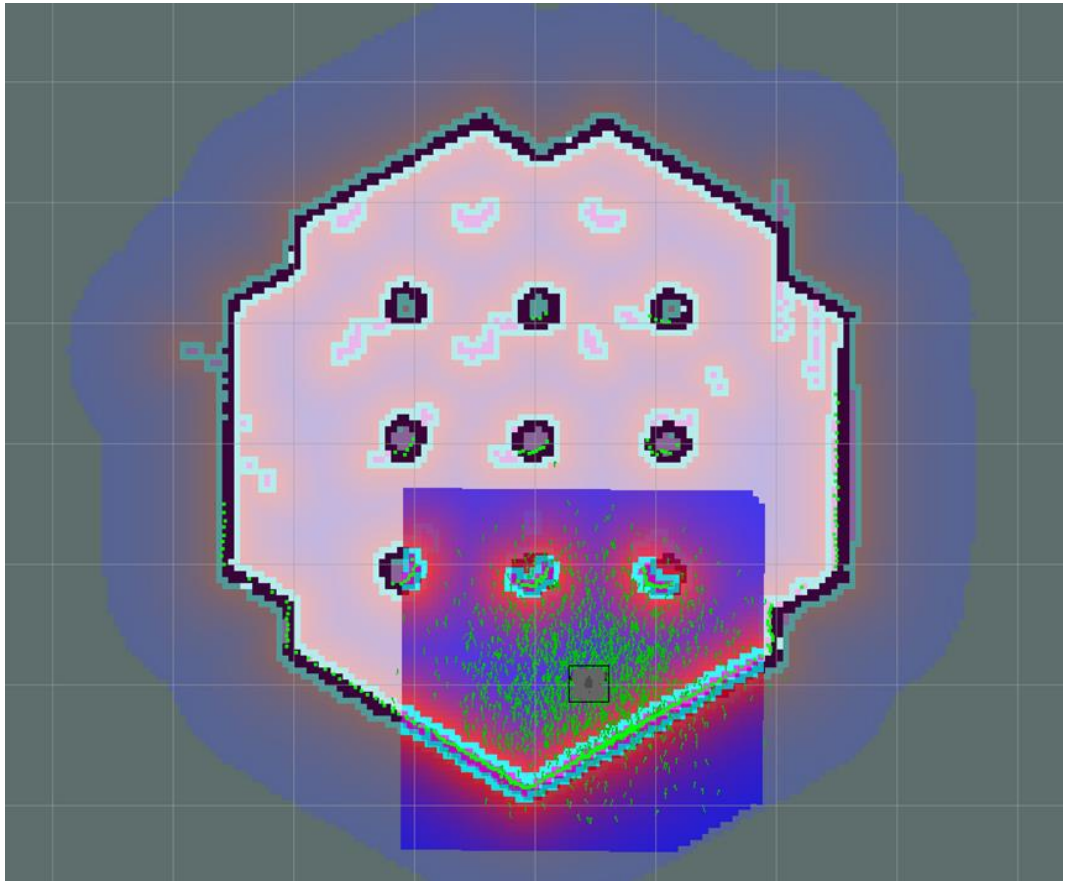
Turtlebot3示例 Navigation

在运行导航功能之前，必须执行初始位姿估计。这一过程会初始化 AMCL 参数，这些参数在导航中至关重要。TurtleBot3 必须在地图上正确定位，并且 LDS（激光测距传感器）数据需要与显示的地图精确对齐。

1. 在 RViz 菜单中点击 **2D Pose Estimate** 按钮。
2. 在地图上点击实际机器人所在的位置，并拖动绿色的大箭头，使其指向机器人正面所朝的方向。
3. 重复步骤 1 和步骤 2，直到 LDS 传感器数据与保存的地图重合。

Turtlebot3示例 Navigation





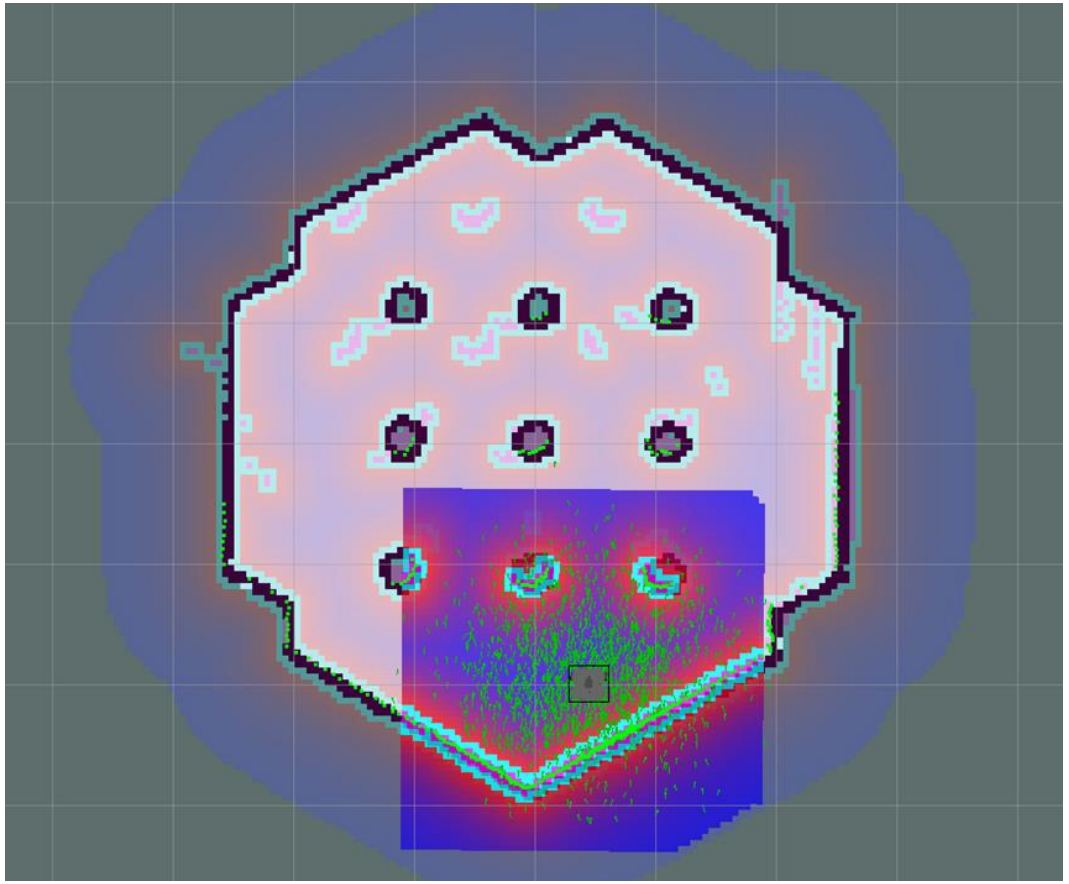
Turtlebot3示例 Navigation

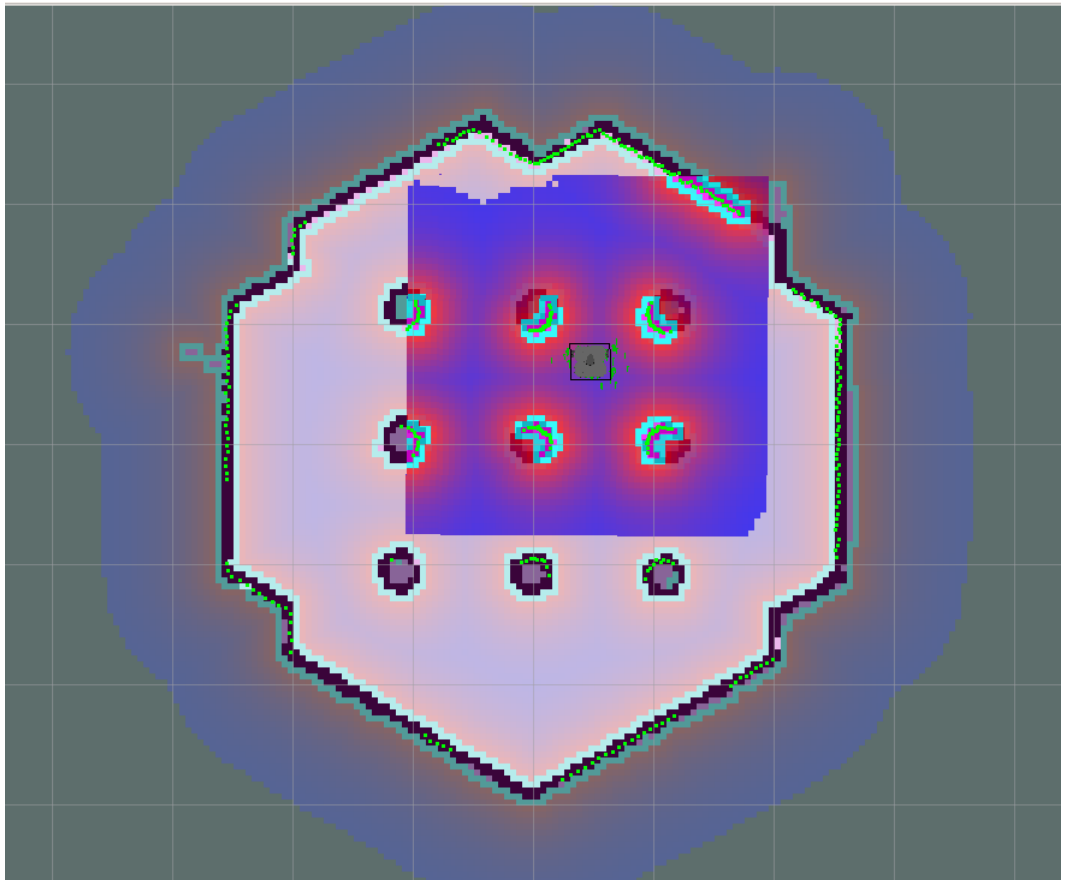
- 启动键盘遥控节点以精准地在地图上定位机器人

  终端4 `roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch`

- 操作机器人前后移动几次，以收集周围环境信息，并在地图上缩小对 TurtleBot3 位置的估计范围。地图上会以小的绿色箭头显示机器人的位置。
- 通过在遥控节点的终端中按下 Ctrl + C 来终止键盘遥控节点，以避免在导航期间多个节点发布不同的 cmd_vel 值（速度指令）。

Turtlebot3示例 Navigation

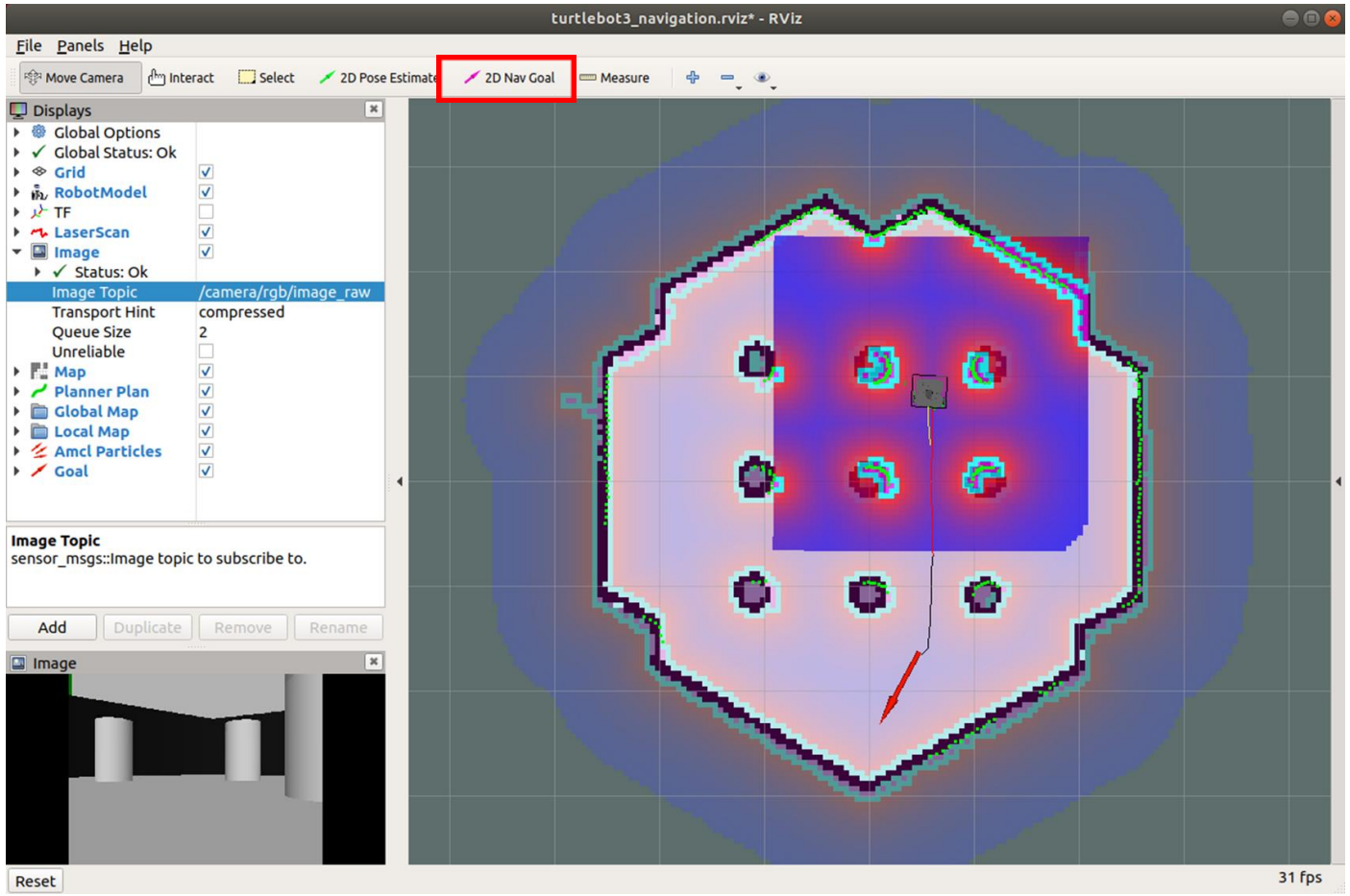




Turtlebot3示例 Navigation

- 点击 RViz 菜单中的 2D Nav Goal 按钮。

- 在地图上点击设置机器人的目标位置，并拖动紫色箭头指向机器人最终需要面向的方向。



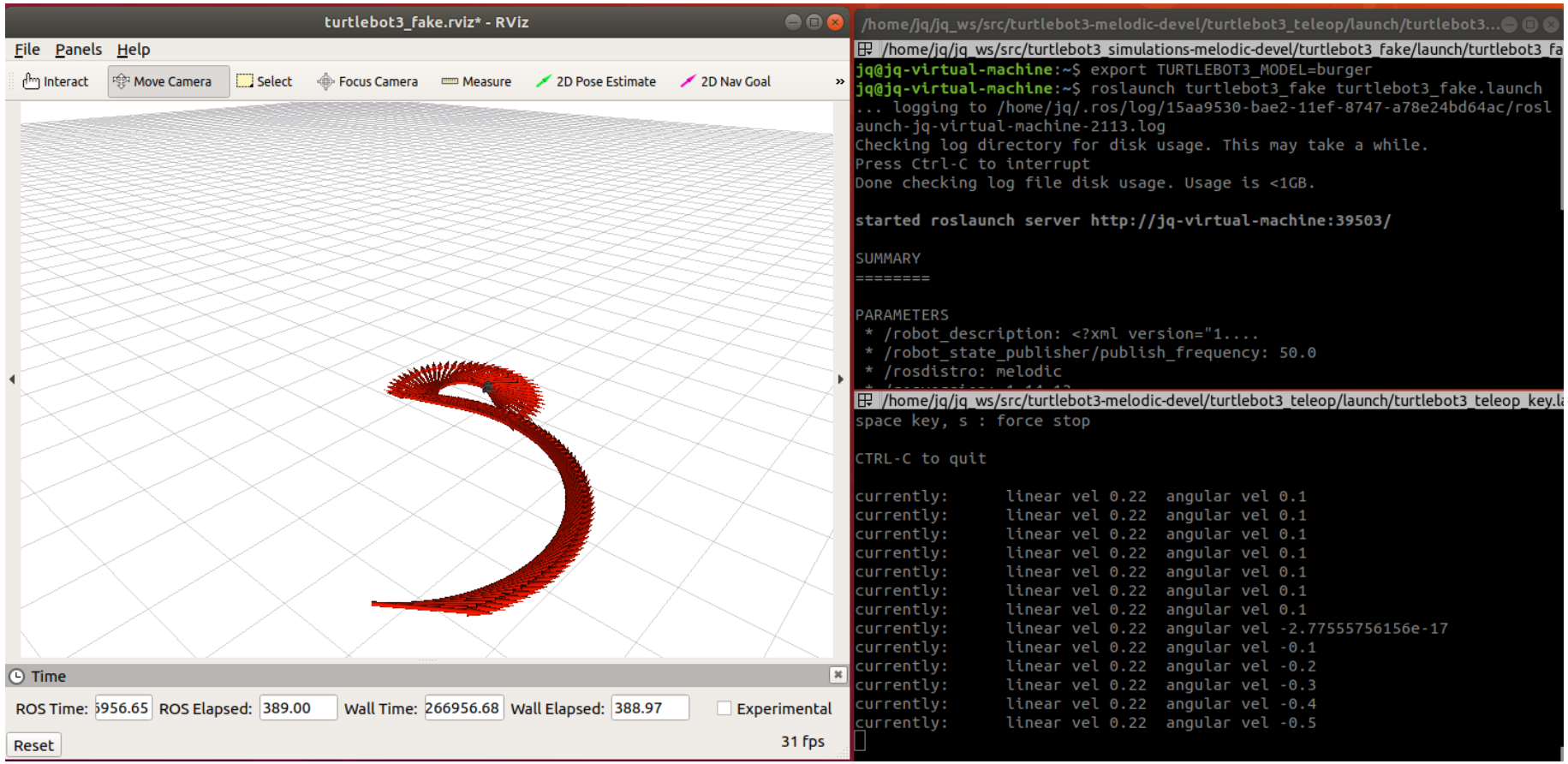
Turtlebot3示例 Fake Node

- export TURTLEBOT3_MODEL=burger
- roslaunch turtlebot3_fake turtlebot3_fake.launch

终端1

- export TURTLEBOT3_MODEL=burger
- roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

终端2



拓展案例

- GitHub - hikashi/multi-robot-rrt-exploration-melodic: A platform for executing RRT exploration in ROS Melodic and Ubuntu 18.04LTS