

Turtlebot3 机器人的激光SLAM与导航实验

Turtlebot3 机器人的激光SLAM与导航实验

- 一、实验概览
- 二、课前准备
 - 2.1 操作系统
 - 2.2 环境准备
 - 2.3 代码准备
 - 2.4 硬件准备
- 三、课上网络配置
- 四、测试一：PC端和机器人端通讯
- 五、测试二：在笔记本电脑上使用键盘控制Turtlebot3机器人运动
- 六、实验一：同时定位与建图（SLAM）
- 七、实验二：自主导航
- 八、拓展部分-编程给定导航目标点
 - 8.1 基本要求
 - 8.2 进阶

一、实验概览

- 体验在真实机器人上运行的ROS操作系统，控制机器人运动并获取激光传感器信息；
- 了解并体验ROS的多机协作功能；
- 亲手控制机器人在室内进行基于小型激光雷达的SLAM实验，建立局部室内环境的二维占有地图；
- 使用上一步生成的地图，人为设定机器人当前位置和目的点，进行机器人的自主导航。

二、课前准备

2.1 操作系统

进行实验前，请确认自己的电脑上安装了 Ubuntu 16.04 操作系统和 Kinetic 版本的ROS操作系统。本实验讲义是在同学们都使用 Kinetic 版本ROS的假设下撰写的。

注意

- Turtlebot3 支持的最低ROS版本版本为 Kinetic，低于此版本的ROS可能无法正常进行下面的实验；
- 理论上Turtlebot3 也能够支持 Lunar 和 Melodic 版本的ROS，但是目前还没有测试过，如果有同学使用这些版本ROS在进行实验的过程中出现问题的话，实验课上恐怕难以给予帮助。

2.2 环境准备

完成 Turtlebot3 及有关依赖包的安装，注意复制后不要直接在命令行中粘贴，先粘贴到文本编辑器中，删除每行末尾的换行符后再复制粘贴到终端中：

```
$ sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-interactive-markers
```

建议课前就完成该过程。

2.3 代码准备

从 Github 上下载本次实验用到的源代码，或从附带的源码包中复制代码文件到ROS工作空间下亦可。假设同学们的工作空间在 `~/catkin_ws/` 下：

```
$ cd ~/catkin_ws/src/  
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git  
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
```

将源码放置到了工作空间之后，编译：

```
$ cd ~/catkin_ws && catkin_make  
$ source ~/catkin_ws/devel/setup.bash
```

注意确认在 `catkin_make` 阶段没有产生任何报错。

2.4 硬件准备

每小组最好携带一套键盘鼠标，实验过程中需要在 Turtlebot3 的树莓派上进行一些操作。

三、课上网络配置

本次实验需要使用Turtlebot3机器人和同学们的笔记本协同工作，为了让两者能够共同工作，需要设置一下网络。下面简称笔记本电脑为**PC端**，Turtlebot3机器人为**机器人端**，操作之前请注意是在哪一端进行操作。

1. **PC端**：首先将笔记本电脑连接到某个无线网络，例如校园网、自己带的无线路由器、或者是手机的Wifi热点：
 - 如果是连接到了校园网，请登录IP网关，否则无线接入情况下无法访问校园内的资源（例如下一步即将使用到的机器人）
 - 如果是使用手机Wifi热点，请确保开启的热点频段为"2.4GHz"而不是"5GHz"，因为下一步使用的机器人携带的树莓派在中国连接5GHz的Wifi还是有些问题；
 - 如果是自己携带的路由器，请注意依旧需要将路由器连接到校园网，否则接下来的机器人时间同步过程无法进行。

确认连接成功后执行下列命令来获取PC端的IP地址：

```
$ ifconfig
```

现在一般电脑上都有多块网卡或者虚拟网卡，每块网卡都具有自己的IP。请注意使用的网卡是无线网卡，对于其中一块网卡的信息中：

```
...
enp3s0    Link encap:Ethernet  HWaddr 40:8d:5c:79:a2:ad
          inet addr:192.168.10.239  Bcast:192.168.10.255  Mask:255.255.255.0
          ...
```

`inet addr` 后面的 `192.168.10.239` 就是实验需要PC端的IP地址。处于不同网络环境下的的电脑的IP不同，下面的一些先验知识可以帮助大家快速判断哪个是这一步中我们需要的IP：

- 对于校园网，IP地址一般形式为 `219.216.XXX.XXX` 或者是 `172.17.XXX.XXX`
- 对于手机Wifi热点或者是自己的路由器，IP地址一般形式为 `192.168.XXX.XXX`
- IP地址 `127.0.0.1` 指代本机，如果执行 `ifconfig` 后只能够找到这个IP地址，说明没有成功连接到任何网络

下面的实验章节中将使用 `IP_OF_PC` 来指代得到的这个IP地址。

2. **PC端**：打开 `~/.bashrc` 文件，这里使用系统默认安装的编辑器 `gedit`，也可以使用你喜欢的编辑器打开：

```
$ gedit ~/.bashrc
```

修改或者添加ROS主节点的配置：

```
# for waffle
export ROS_MASTER_URI=http://IP_OF_PC:11311
export ROS_HOSTNAME=IP_OF_PC
export TURTLEBOT3_MODEL=waffle_pi
```

其中 `IP_OF_PC` 就是在上一步中查询得到的PC端的IP地址。

完成后记得**保存**，然后在当前窗口执行：

```
$ source ~/.bashrc
```

来使得设置生效。

3. **机器人端**：

注意在启动机器人之前就要将HDMI连接线插到树莓派上，否则启动后再连接有一定概率会出现无显示信号的现象。启动后登录系统，用户名和密码默认都是 `attb3`。进入桌面后连接网络，注意要和笔记本连接到同一个网络中。

加载出桌面以后按 `Ctrl+Alt+T` 打开终端，执行以下几条命令，确认当前Turtlebot机器人上已经正确完成环境配置：

```
$ source /opt/ros/kinetic/setup.bash
$ cd ~/catkin_ws && catkin_make
$ source ~/catkin_ws/devel/setup.bash
$ rosrun turtlebot3_bringup create_udev_rules
```

接下来需要进行时间同步工作。首先安装需要的组件：

```
$ sudo apt-get install chrony
$ sudo apt-get install ntpdate
```

然后执行下面的命令来同步时间：

```
$ sudo ntpdate ntp.ubuntu.com
```

同步完成后可以输入下面的命令查看当前日期和时间是否正确：

```
$ date
```

4. **机器人端**：下面在Turtlebot上进行网络配置。执行 `ifconfig` 获取Turtlebot的IP地址：

```
$ ifconfig
```

下文将使用 `IP_OF_TURTLEBOT3` 来表示Turtlebot3机器人的IP地址。修改 `~/.bashrc`，注意 Turtlebot3 机器人上的树莓派安装的是MATE版Ubuntu，默认安装的编辑器是 `pluma`：

```
$ pluma ~/.bashrc
```

修改或添加配置：

```
export ROS_MASTER_URI=http://IP_OF_PC:11311
export ROS_HOSTNAME=IP_OF_TURTLEBOT3
```

记得**保存**，然后使得配置生效：

```
source ~/.bashrc
```

四、测试一：PC端和机器人端通讯

目的是为了测试上述网络配置过程正确，为后面的实验进行做好准备。

1. **PC端**：启动 `roscore`：

```
$ roscore
```

输出信息应该和之前正常启动 `rsocore` 没有太大的区别。

2. **机器人端**：启动一系列的机器人基本驱动程序：

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

输出示例：

```

/home/attb3/catkin_ws/src/turtlebot3/turtlebot3_bringup/launch/turtlebot3_robot.launch http://192.168.43.156:11311
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)
/home/attb3/catkin_ws/src/turtlebot3/turtlebot3_bringup/launch/turtlebot3_robot.launch http://192.168.43.156:11311  attb3@attb3-desktop: ~
PARAMETERS
* /roslaunch: klnetic
* /rosversion: 1.12.13
* /turtlebot3_core/serial: 115200
* /turtlebot3_core/port: /dev/ttyACM0
* /turtlebot3_core/serial_prefix:
* /turtlebot3_lds/frame_id: base_scan
* /turtlebot3_lds/port: /dev/ttyUSB0

NODES
/
  turtlebot3_core (roscpp/serial_node.py)
  turtlebot3_diagnostics (turtlebot3_bringup/turtlebot3_diagnostics)
  turtlebot3_lds (hls_lfcd_lds_driver/hls_laser_publisher)

ROS_MASTER_URI=http://192.168.43.156:11311

process[turtlebot3_core-1]: started with pid [3412]
process[turtlebot3_lds-2]: started with pid [3413]
process[turtlebot3_diagnostics-3]: started with pid [3414]
[INFO] [1561031660.504679]: ROS Serial Python Node
[INFO] [1561031660.581732]: Connecting to /dev/ttyACM0 at 115200 baud
[INFO] [1561031662.815615]: Note: publish buffer size is 1024 bytes
[INFO] [1561031662.817031]: Setup publisher on sensor_state [turtlebot3_msgs/SensorState]
[INFO] [1561031662.830276]: Setup publisher on firmware_version [turtlebot3_msgs/FirmwareVersionInfo]
[INFO] [1561031662.986063]: Setup publisher on imu [sensor_msgs/Imu]
[INFO] [1561031663.011248]: Setup publisher on cmd_vel [geometry_msgs/Twist]
[INFO] [1561031663.071280]: Setup publisher on odom [nav_msgs/Odometry]
[INFO] [1561031663.093984]: Setup publisher on joint_states [sensor_msgs/JointState]
[INFO] [1561031663.120956]: Setup publisher on battery_state [sensor_msgs/BatteryState]
[INFO] [1561031663.141256]: Setup publisher on magnetic_field [sensor_msgs/MagneticField]
[INFO] [1561031665.980876]: Setup publisher on tf [tf/TFMessage]
[INFO] [1561031666.026240]: Note: subscribe buffer size is 1024 bytes
[INFO] [1561031666.027935]: Setup subscriber on cmd_vel [geometry_msgs/Twist]
[INFO] [1561031666.061853]: Setup subscriber on sound [turtlebot3_msgs/Sound]
[INFO] [1561031666.099866]: Setup subscriber on motor_power [std_msgs/Bool]
[INFO] [1561031666.132947]: Setup subscriber on reset [std_msgs/Empty]
[INFO] [1561031666.153414]: Setup TF on Odometry [odom]
[INFO] [1561031666.157457]: Setup TF on imu [imu_link]
[INFO] [1561031666.161443]: Setup TF on MagneticField [mag_link]
[INFO] [1561031666.164765]: Setup TF on JointState [base_link]
[INFO] [1561031666.185106]: -----
[INFO] [1561031666.186645]: Connected to OpenCR board!
[INFO] [1561031666.192893]: This core(v1.2.2) is compatible with T3 Burger
[INFO] [1561031666.196942]: -----
[INFO] [1561031666.202872]: Start Calibration of Gyro
[INFO] [1561031668.716726]: Calibration End

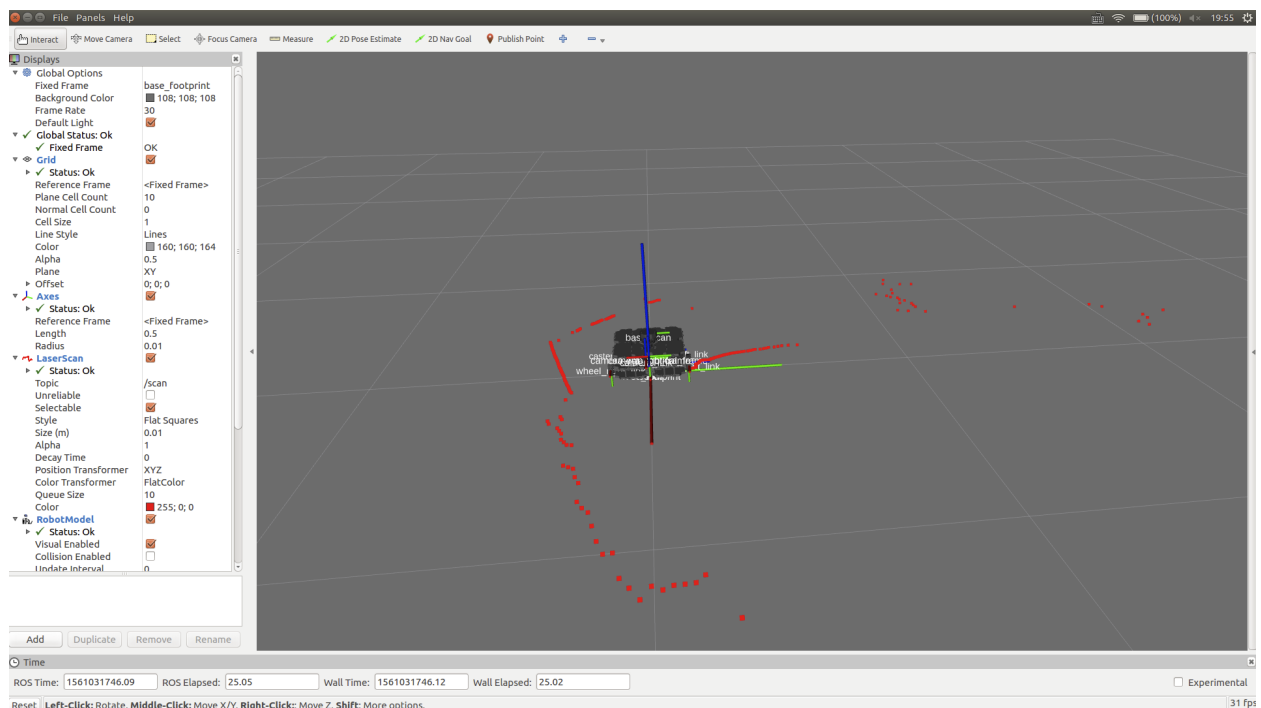
```

正常情况下应该没有任何的错误和警告。另在最后提示进行 Start Calibration of Gyro 期间，请保持机器人水平静止放置。当提示 Calibration End 之后，初始化阶段完成，可以进行下一步操作。

3. PC端：启动 rviz 查看机器人激光雷达数据：

```
$ roslaunch turtlebot3_bringup turtlebot3_model.launch
```

正常运行示例：



如果前面的时间同步不成功，可能会造成这里的机器人模型加载异常，或者是激光传感器数据错误。

4. 如果上述过程中均没有问题，就说明通讯测试是成功的。为节省时间，请不要关闭机器人端的终端，并请关闭PC端的 rviz。此时也可以将鼠标、键盘、显示器连接线从机器人底盘上的树莓派上拔下来了，并将机器人放在地面上，为下面的实验做准备。

五、测试二：在笔记本电脑上使用键盘控制Turtlebot3机器人运动

现在进一步测试能否用电脑控制机器人运动，为下一步实验做准备。

0. 如果没有退出测试一中机器人端的终端的话，可以直接进行后面的步骤；否则请先在**PC端**执行：

```
$ roscore
```

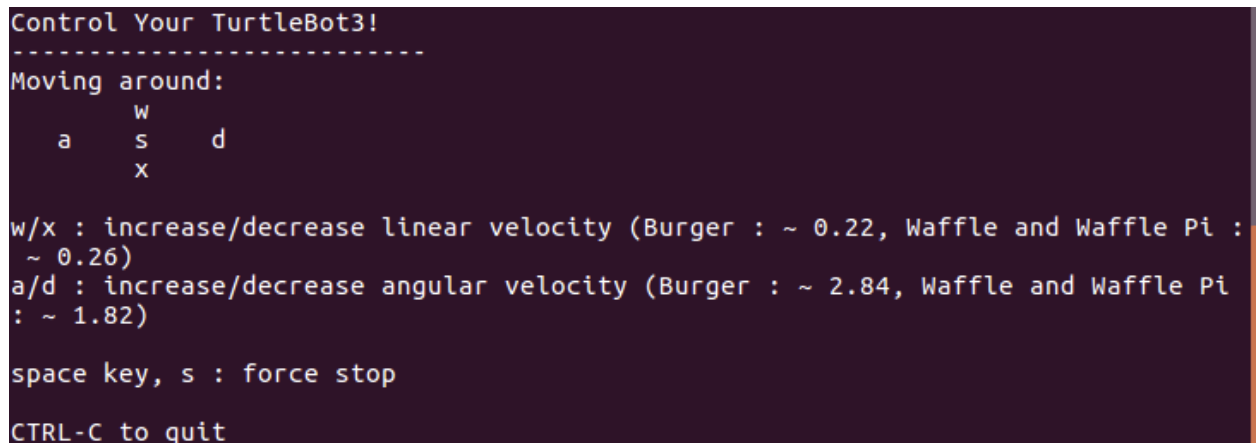
然后在**机器人端**执行命令：

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

1. **PC端**：执行键盘控制节点：

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

正常输出：



```
Control Your TurtleBot3!
-----
Moving around:
   w      a      s      d      x
   ↑      ↙      ↓      ↘      ↻

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit
```

基本操作：w-前进 x-后退 a-左转 d-右拐 s-刹车

为节省时间，完成当前测试后不需要关闭该终端。

六、实验一：同时定位与建图（SLAM）

0. 如果在测试二之后没有关闭任何终端，跳过此步；否则请先在**PC端**执行：

```
$ roscore
```

然后在**机器人端**执行：

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

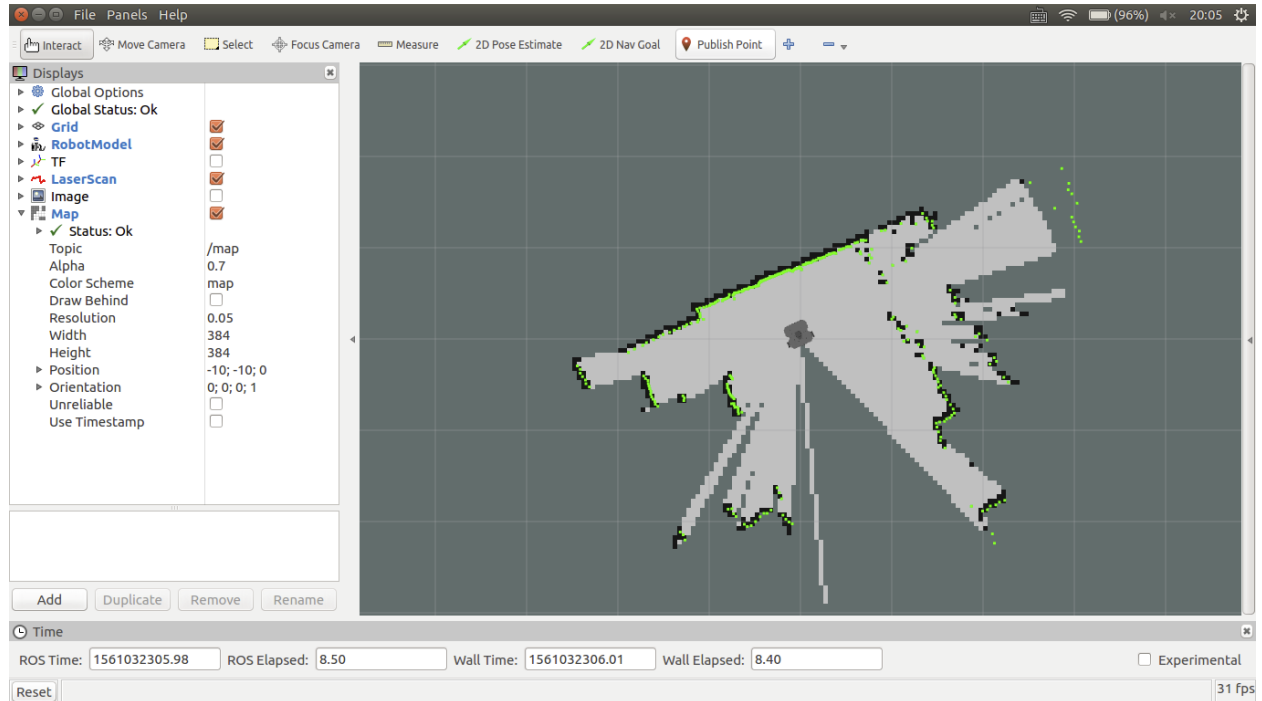
最后在**PC端**执行：

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

1. **PC端**：运行以下指令来启动SLAM建图程序，并且打开rviz查看实时效果：

```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

rviz 输出实例：

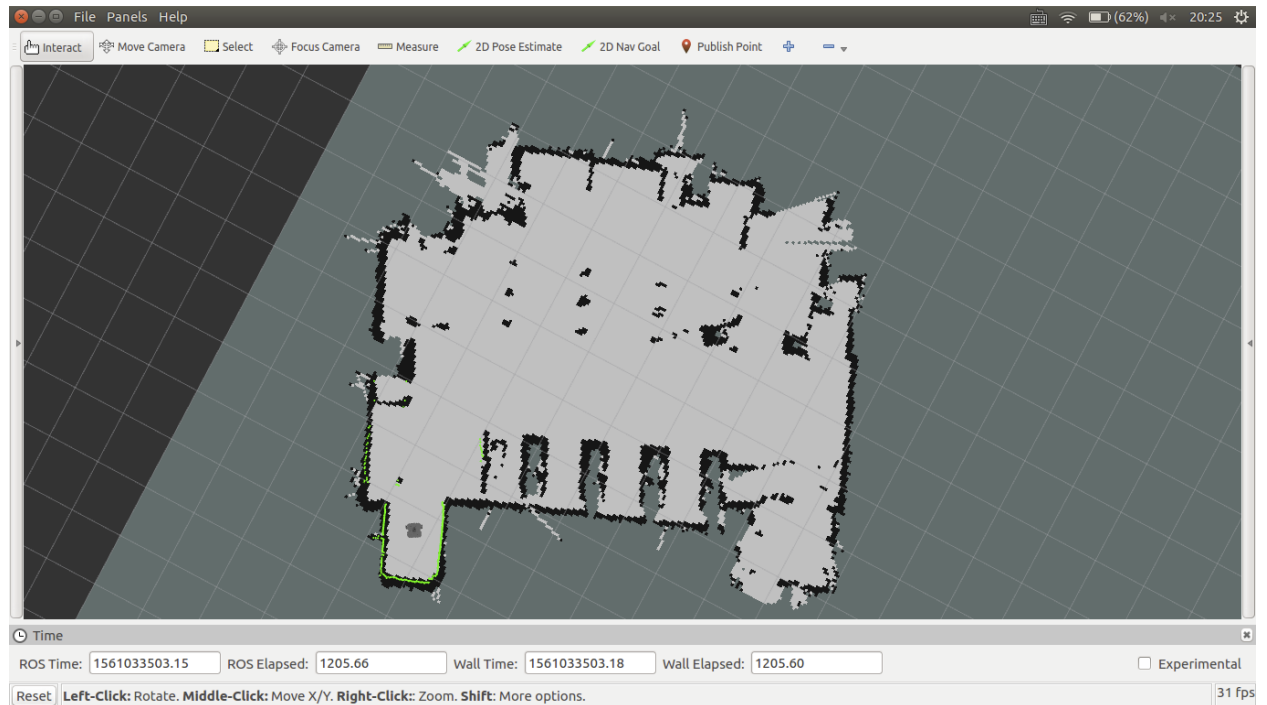


2. **PC端**：切换到运行“键盘控制”节点的终端，使用键盘控制Turtlebot小车运动、建图

3. **PC端**：地图建立完成之后，可以运行 `map_server` 来保存构建的地图：

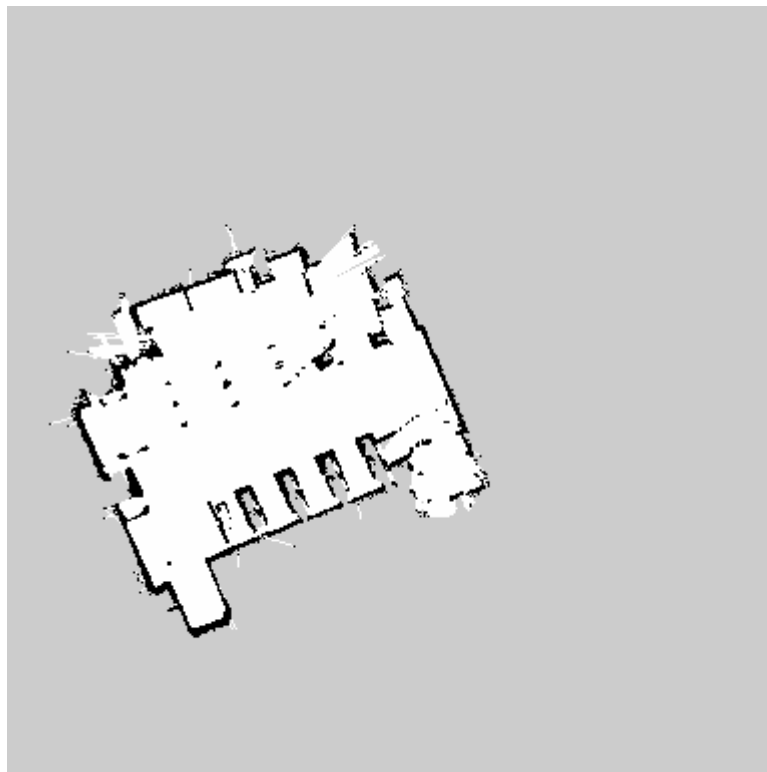
```
$ cd ~  
$ rosrun map_server map_saver -f mymap
```

示例：（实际试验中只需要有局部的室内地图即可）



注意

- 上述操作中是把地图保存到用户的 `Home` 目录下；如果想保存到别的目录下，需要 `cd 目标目录` 之后再运行 `map_server` 来保存地图
- 上述命令中得到的地图文件名是 `mymap.yaml` 和 `mymap.pgm` (你也可以修改上述命令中的 `mymap` 来自定义地图文件名)，两个文件在同一个目录下，其中 `*.pgm` 文件可以直接以图片打开，效果如下图所示：



- 由于实验室人多、走动、多台机器人相互干扰等原因，建图效果可能并不好

如果你决定进行下一步实验了，请在机器人停止运动后保存地图，**保存地图前最好使得机器人位于建图的初始位置处，同时保存完成地图后最好保持机器人当前的位置不变**，这样当进行下一个实验时，机器人将会默认处于地图中的原点，而这里的原点就是开始建图时机器人所处的位置。如果满足不了以上要求也没有关系，我们可以手动定位，但是效果在有些情况下可能会要差一些。

进行下一个实验之前，请确认地图已经保存，并在**PC端**退出SLAM建图和键盘控制的终端程序，但是不要关闭**PC端**的 `roscore`。

七、实验二：自主导航

0. 如果是从实验一进行到这一步，没有关闭**PC端**的 `roscore`，可以跳过此步。否则，请首先在**PC端**执行：

```
$ roscore
```

然后在**机器人端**执行：

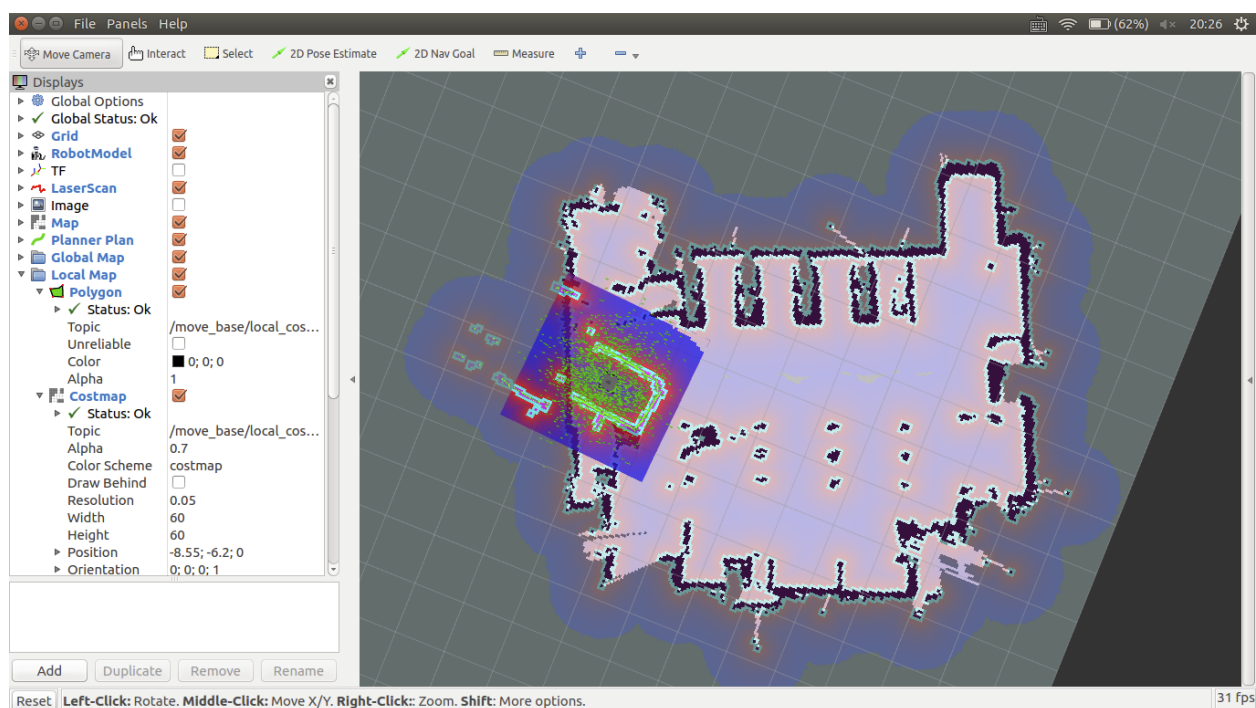
```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

1. **PC端**：执行自动导航有关的节点：

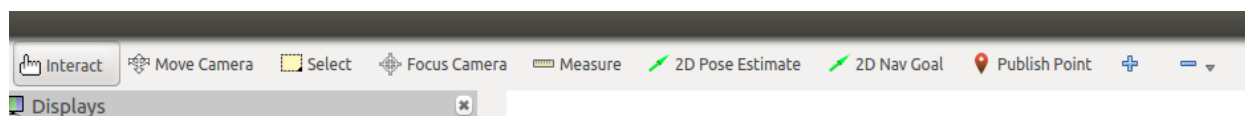
```
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch  
map_file:=$HOME/mymap.yaml
```

启动文件的参数中写的是地图 `yaml` 文件的**绝对路径**。如果自定义了地图存放位置或地图名称，请注意修改上述命令中最后 `map_file` 参数。

2. **PC端**：如果进行实验一的时候机器人的处于建图时的初始位置，那么在 `rviz` 中查看到的机器人位置应该和实际环境中的机器人位置相同；否则如果是像下图一样，机器人的初始位置明显不正确的时候，需要手动设定机器人的初始位置：

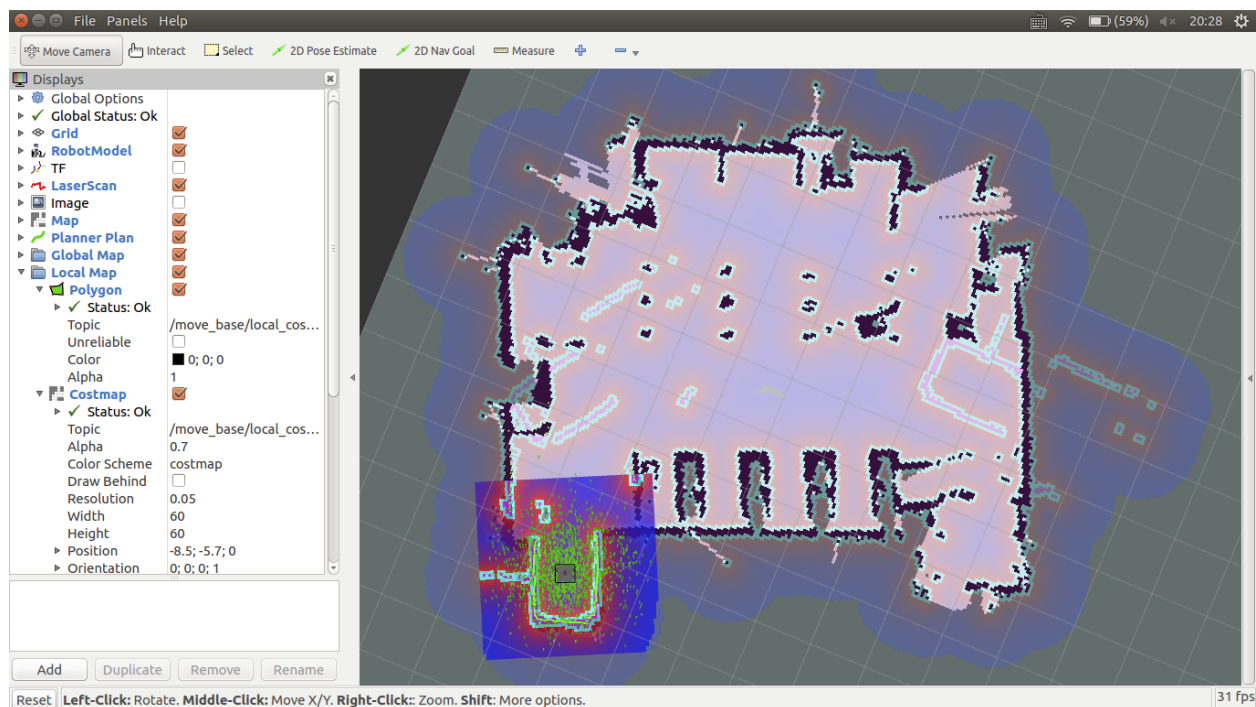


当需要手动设定机器人的初始位置时，请点击 `rviz` 窗口上方的 `2D Pose Estimate` 按钮：



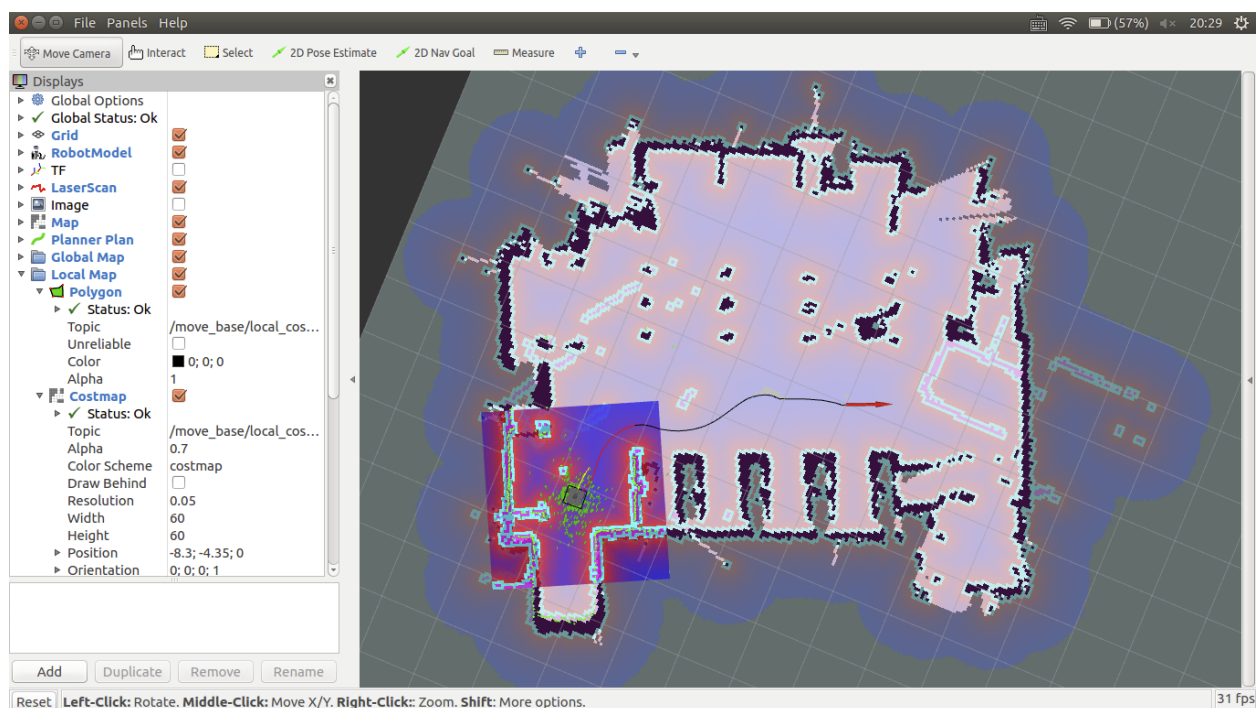
然后在地图中移动鼠标指针到机器人的实际位置附近，按住鼠标左键不放以设置其位置；然后向和机器人前方一致的方向拖动鼠标（可以看到有个绿色箭头跟随鼠标拖动方向）来设置机器人的朝向；最后松开鼠标完成设置。如果初始位置设置的不理想，可以多次设置。

机器人处于正确初始位置时的示例图：



3. **PC端**：点击 rviz 窗口上方的 2D Nav Goal 按钮，在地图中点击和拖动设置机器人的目标点，然后观察机器人是如何进行路径规划和运动的。

实际测试效果图：



如果最终机器人成功地到达的目的地，在终端上会输出有 Goal Reached 字样：

```
[ INFO] [1561034604.204086620]: Got new plan
[ INFO] [1561034604.404068577]: Got new plan
[ INFO] [1561034604.604007475]: Got new plan
[ INFO] [1561034604.804132227]: Got new plan
[ INFO] [1561034605.004108837]: Got new plan
[ INFO] [1561034605.204034836]: Got new plan
[ INFO] [1561034605.404051883]: Got new plan
[ INFO] [1561034605.604130729]: Got new plan
[ INFO] [1561034605.604244529]: Goal reached
```

八、拓展部分-编程给定导航目标点

此部分拓展实验部分属于加分项，做出来会有额外加分奖励。

在上面的自主导航的实验过程中，如要给定机器人的导航目标点，需要在rviz窗口中使用鼠标点击和拖动的方式给定。而在许多应用场合中，需要通过编程的方式来提供导航的目标点。

8.1 基本要求

1. 自己动手编写ROS节点，实现在程序中给定机器人导航的目标点；
2. 导航目标点的位置坐标和朝向角度要求在调用该节点程序时，可以从键盘输入；
3. 完成该项实验验收时，要求机器人给定的导航目标点需要在机器人的背后，即在导航动作执行的过程中，机器人需要有一个明显的转向过程。
4. 程序功能包的CMakeLists.txt文件及package.xml文件中所需要的依赖，可以参考教材第八章和第九章。

8.2 进阶

在8.1的基础上，编程实现当机器人到达目标点之后，可以接着以目标点为圆心，以指定半径进行圆周运动。其中半径需要在运行程序时的命令行参数中给定。