

第十七课 Gmapping 建图算法

1. 初步介绍 Gmapping

对于建图，SLAM, 也称为 CML (Concurrent Mapping and Localization), 我们从名字就可以得知，其包含机器人的定位与地图的构建两部分，或者说并发建图与定位。对于这个问题的模型，就是如果将一个机器人放入未知环境中的未知位置，是否有办法让机器人一边移动一边逐步描绘出此环境完全的地图（完全的地图是指不受障碍行进到房间可进入的每个角落，也就是熟知地图的障碍点）。定位与建图相辅相成、互相影响。

首先，构建地图要知道机器人的精确位姿，精确定位又需要给定的地图做参考。先讲已知精确位姿（坐标和朝向）的地图创建，机器人位置已知，通过激光雷达扫描到环境特征，即障碍物距离。可通过机器人坐标和朝向以及障碍物距离计算出障碍物的坐标，采用 bresenham 直线段扫描算法，障碍物所处的栅格标注为 `occupy`，机器人所处的栅格与障碍物所处的栅格之间画直线，直线所到之处都为 `free`。当然每个栅格并不是简单的非 0 即 1，栅格的占据可用概率表示，若某一个栅格在激光束 `a` 扫描下标识为 `occupy`，在激光束 `b` 扫描下也标识为 `occupy`，那该栅格的占据概率就变大，反之，则变小。这样，机器人所处的环境就可以通过二维栅格地图来表征。

2. Gmapping 功能包实现的前提

1>. 提供的有两个 `tf`，一个是 `base_footprint` 与 `laser` 之间的 `tf`，即机器人底盘和激光雷达之间的变换；我们在 `Huanyu_robot_start.launch` 文件的开头已经通过静态 `tf` 变换提供了。一个是 `base_footprint` 与 `odom_combined` 之间的 `tf`，即底盘和里程计原点之间的坐标变换。`odom_combined` 可以理解为里程计原点所在的坐标系。

2>. `/scan` :激光雷达数据，类型为 `sensor_msgs/LaserScan`。

3. Gmapping 功能包参数说明

```
<launch>
  <arg name="scan_topic" default="scan" /> //激光的 topic 相对应
  <arg name="base_frame" default="base_footprint"/> //机器人的坐标系
  <arg name="odom_frame" default="odom_combined"/> //机器人原点

  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
    <param name="base_frame" value="$(arg base_frame)"/>
    <param name="odom_frame" value="$(arg odom_frame)"/>
    <param name="map_update_interval" value="0.01"/> //地图更新的一个间隔，两次 scanmatch 的间隔，地图更新也受 scanmach 的影响，如果 scanmatch 没有成功的话，是不会更新地图的
    <param name="maxUrange" value="4.0"/> //set maxUrange < maximum range of the real sensor
    <param name="maxRange" value="5.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="3"/>
    <param name="lstep" value="0.05"/> optimize 机器人移动的初始值（距离）
```

```

<param name="astep" value="0.05"/> //optimize 机器人移动的初始值（角度）
<param name="iterations" value="5"/> //icp 的迭代次数
<param name="lsigma" value="0.075"/>
<param name="ogain" value="3.0"/>
<param name="lskip" value="0"/> //为 0,表示所有的激光都处理，尽可能为零，如果计算压力过大，
可以改成 1
<param name="minimumScore" value="30"/> //很重要，判断 scanmatch 是否成功的阈值，过高的话会
使 scanmatch 失败，从而影响地图更新速率
<param name="srr" value="0.01"/> //以下四个参数是运动模型的噪声参数
<param name="srt" value="0.02"/>
<param name="str" value="0.01"/>
<param name="stt" value="0.02"/>
<param name="linearUpdate" value="0.05"/> //机器人移动 linearUpdate 距离，进行 scanmatch
<param name="angularUpdate" value="0.0436"/> //机器人选装 angularUpdate 角度，进行 scanmatch
<param name="temporalUpdate" value="-1.0"/>
<param name="resampleThreshold" value="0.5"/>
<param name="particles" value="8"/> //很重要，粒子个数

<param name="xmin" value="-1.0"/> //map 初始化的大小
<param name="ymin" value="-1.0"/>
<param name="xmax" value="1.0"/>
<param name="ymax" value="1.0"/>

<param name="delta" value="0.05"/>
<param name="llsamplerange" value="0.01"/>
<param name="llsamplestep" value="0.01"/>
<param name="lasamplerange" value="0.005"/>
<param name="lasamplestep" value="0.005"/>
<remap from="scan" to="$(arg scan_topic)"/>
</node>

```

4. 在 HiBot 上完成准备工作

首先打开 HiBot 的电源开关，等待大概 1 分钟后，Ubuntu 主机连接树莓派开放的 AP（名称：HiBot-xxx，密码：12345678）。连接成功后，打开新的终端通过 ssh 连接树莓派，（密码：huike）

```
huanyu@ubuntu:~$ ssh huike@192.168.12.1
```

连接成功后，通过 nfs 挂载机器人的网络文件盘，方便代码查看和修改。

```
huanyu@ubuntu:~$ sudo mount -t nfs 192.168.12.1:/home/huike/robot_ws/ /mnt
```

在此之前请查看 ubuntu 主机和树莓派的连接相关课程，确保 Ubuntu 主机和树莓派的 ROS 网络是通畅的。

5. 在 HiBot 上利用 Gmapping 创建地图

1>. 在现实环境中选取一个合适的点，将机器人放置在选取的点上，开始运行机器人的启动节点。在 ssh 连接的窗口内运行以下 launch 文件：

```
huike@huike-desktop:~$ roslaunch huanyu_robot_start Huanyu_robot_start.launch
```

```
[ INFO] [1558887700.436580691]: [ZHOUXUEWEI] Serial Port opened
[ INFO] [1558887700.446627714]: output frame: odom_combined
[ INFO] [1558887700.447076506]: base frame: base_footprint
[ INFO] [1558887702.458841292]: Initializing Odom sensor
[ INFO] [1558887702.460137774]: Initializing Imu sensor
[ INFO] [1558887702.508447096]: Odom sensor activated
[ INFO] [1558887702.509350462]: Imu sensor activated
[ INFO] [1558887702.529683981]: Kalman filter initialized with odom measurement
```

2>. 打开机器人键盘控制节点，方便在建图时移动机器人。在新的终端内 ssh 连接到树莓派，然后运行以下 launch。按键盘控制机器人时，确保鼠标光标在

keyboard_teleop.launch 运行的窗口内。

```
huike@huike-desktop:~$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

```
Moving around:
u i o
j k l
m , .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly
```

3>. 运行 Gmapping 建图包。在新的终端内 ssh 连接到树莓派，然后运行以下 launch。

```
huike@huike-desktop:~$ roslaunch huanyu_robot_start gmapping_slam.launch
```

```
[ INFO] [1558887710.681366385]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= 0.05 0 -0.00872675
m_count 0
Registering First Scan
```

4>. 等到以上三个 launch 运行正常后，在 Ubuntu 主机上打开新的终端，运行 rviz，在数据窗口内查看地图和机器人姿态数据。rviz 中的消息和话题选择如下：即可看到地图、激光点、机器人的 tf 位姿数据。



5>. 开始移动机器人，尽量走一个闭合的曲线，当地图创建已经满足需要时，打开新的终端通过 ssh 连接到树莓派，进入到 robot_ws/src/huanyu_robot_start/map/ 目录下，开始利用 map_server 节点保存地图，rosrun map_server map_saver -f map

```
huike@huike-desktop:~$ cd robot_ws/src/huanyu_robot_start/map/
huike@huike-desktop:~/robot_ws/src/huanyu_robot_start/map$ rosrun map_server map_saver -f map
[ INFO] [1558888628.203850831]: Waiting for the map
[ INFO] [1558888628.406276356]: Received a 544 X 480 map @ 0.050 m/pix
[ INFO] [1558888628.406630827]: Writing map occupancy data to map.pgm
[ INFO] [1558888628.564130134]: Writing map occupancy data to map.yaml
[ INFO] [1558888628.565180893]: Done

huike@huike-desktop:~/robot_ws/src/huanyu_robot_start/map$ ls
map.pgm map.yaml
huike@huike-desktop:~/robot_ws/src/huanyu_robot_start/map$
```

保存成功后我们可以在 `map` 目录下看到两个文件了。

6. 总结

本节课程我们了解了 `Gmapping` 建图算法的基本原理，同时树立了 `Gmapping` 功能包运行时需要的话题数据，最后在 `HiBot` 上成功创建了一个全新环境中的地图。