基于 ROS 系统的移动扫地机器人

一、选题背景

扫地机器人凭借着全自主地移动清扫功能，在近几年大受欢迎。

基于 ROS 系统的扫地机器人项目需要达成如下目标：（1）自主建图；（2）路径规划；（3）按照规划路径自主导航

二、方案论证(设计理念)

我们小组使用的是 Noetic 版本的 ROS 系统。首先我们确定了扫地机器人的运动方式和传感器。其外形大致为一扁平圆柱体，有两个驱动轮和两个牛眼轮，运动方式为两轮差速运动，所带传感器为 2D 激光雷达和电机里程计。

根据扫地机器人的自主建图需求，我们采用的是 explore_lite 算法。这是一种基于边界的探索，机器人会贪婪地探索环境直到找不到边界，而且该算法不会创建自己的成本图，效率相对较高，这十分适合封闭的室内环境和扫地机器人较有限的算力。

在扫地机器人得到全局地图后，还需要做全覆盖的路径规划，路径规划部分我们是基于当前点分别考虑下一点的坐标和方向，不断迭代得到整体路径。

最后对于扫地机器人的自主导航，我们通过设定一距离阈值，当扫地机器人进入该范围后便发布下一目标点，再通过全局地图、自身定位、路径规划和运动控制使机器人向目标点移动。

三、过程论述

3.1 结构建模

首先对于扫地机器人的建模我们使用 xacro 集成 urdf 并分模块编写的方式建立机器人模型，如图 3.1.1。

图 3.1.1 扫地机器人模型

之后对于环境的建模，我们直接在 Gazebo 中通过图形化界面建立了一个环境，如图 3.1.2。

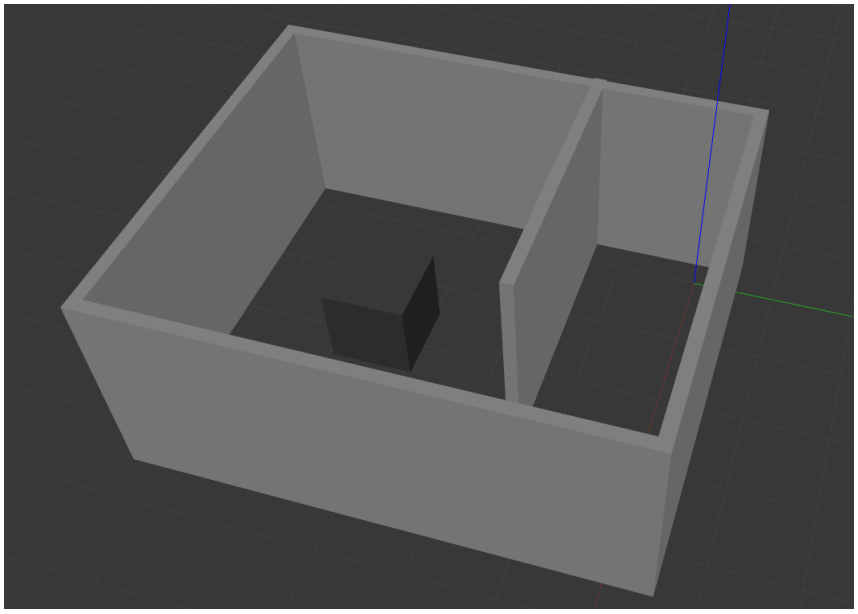

图 3.1.2 环境建模

3.2 自主建图

该部分我们直接使用的是 explore_lite 算法。其为基于边界的探索，当节点运行时，机器人会贪婪地探索环境直到找不到边界。与类似的包不同，explore_lite 不会创建自己的代价图,这使得配置更容易且更高效(资源更少)。节点只是订阅 nav_msgs/OccupancyGrid 消息。机器人移动的命令被发送到 move_base 节点。节点可以进行边界过滤，甚至可以在非膨胀地图上运行。目标黑名单允许处理机器人无法进入的地方。该算法运行时的节点间关系如图 3.2.1 所示。

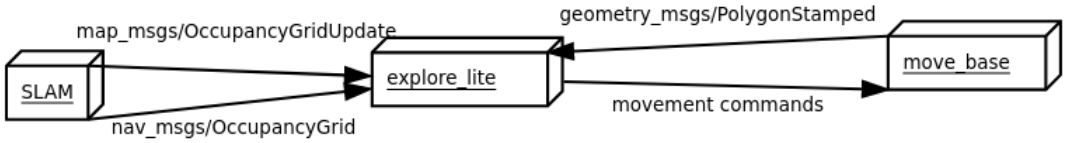

图 3.2.1 explore_lite 运行时节点关系图

3.3 路径规划及自主导航

由 3.2 我们能得到环境的栅格地图，但是栅格地图的分辨率是 5cm，而我们的扫地机器人的半径是 10cm，若直接规划路径稍显不合理。故我们考虑先按 3*3 的大小合并栅格，这样意味着新地图的分辨率是 15cm，即扫地机器人能扫过的宽度为 15cm，更加合理。在合并栅格时，考虑到栅格地图为 .pgm 格式，本质为一张图片，故我们使用 OpenCV 来进行合并和后续路径规划的任务。

在合并栅格时应注意在原栅格地图中原点在左下角而在 Mat 类型中原点在左上角。同时要注意的是我们应设置图片类型为 CV_32F，以免影响后续算法精度。此外，我们规定若原 9 个栅格中都没有障碍，新栅格才设为没有障碍。

在得到新“栅格地图”后，我们选定机器人初始化的坐标所在栅格为起始栅

格，机器人初始化的方向为起始方向，并规定两个路径点间的角度只能为 0°，45°，90°，135°，180°，215°，270°或 315°。现在我们规定第 i 行第 j 列的栅格权值表达为 $V_{i,j}$，若该栅格无障碍，权值为 V_0；若该栅格有障碍，权值为 V_1，再规定该栅格角度为 $A_{i,j}$。之后我们基于当前栅格和方向进行下一栅格的角度决策，对于 0°，我们计算:

$$V_0 = V_{i,j+1} + C \cdot (1 - F(|A_{i,j} - 0°|)/180°)$$

对于 45°，我们计算：

$$V_{45} = V_{i-1,j+1} + C \cdot (1 - F(|A_{i,j} - 45°|)/180°) - K$$

对于 90°，我们计算：

$$V_{90} = V_{i-1,j} + C \cdot (1 - F(|A_{i,j} - 90°|)/180°)$$

对于 135°，我们计算：

$$V_{135} = V_{i-1,j-1} + C \cdot (1 - F(|A_{i,j} - 135°|)/180°) - K$$

对于 180°，我们计算：

$$V_{180} = V_{i,j-1} + C \cdot (1 - F(|A_{i,j} - 180°|)/180°)$$

对于 215°，我们计算：

$$V_{215} = V_{i+1,j-1} + C \cdot (1 - F(|A_{i,j} - 215°|)/180°) - K$$

对于 270°，我们计算：

$$V_{270} = V_{i+1,j} + C \cdot (1 - F(|A_{i,j} - 270°|)/180°)$$

对于 315°，我们计算：

$$V_{315} = V_{i+1,j+1} + C \cdot (1 - F(|A_{i,j} - 315°|)/180°) - K$$

其中我们取 V_0=50，V_1=-100000，C=50，K=200，且有：

$$F = \begin{cases} X, (0 \le X \le 180) \\ 360 - X, (180 < X \le 360) \end{cases}$$

通过以上计算，我们取计算结果最大的那个角度作为下一个栅格角度。

再考虑下一个点的坐标，我们取距离当前栅格最近的那个栅格作为下一栅格。这样如此往复，我们就能得到全局的扫地路径，一种示例如图 3.3.1 所示。

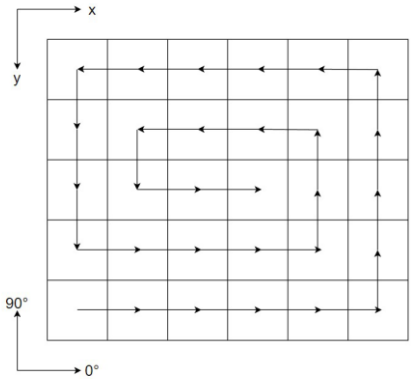

图 3.3.1 扫地路径的一种示例

得到了路径点后我们要按顺序发布出去，这里我们判断机器人在当前目标点一定范围内便发布下一目标点。

3.3 的节点拓扑如图 3.3.2 所示：

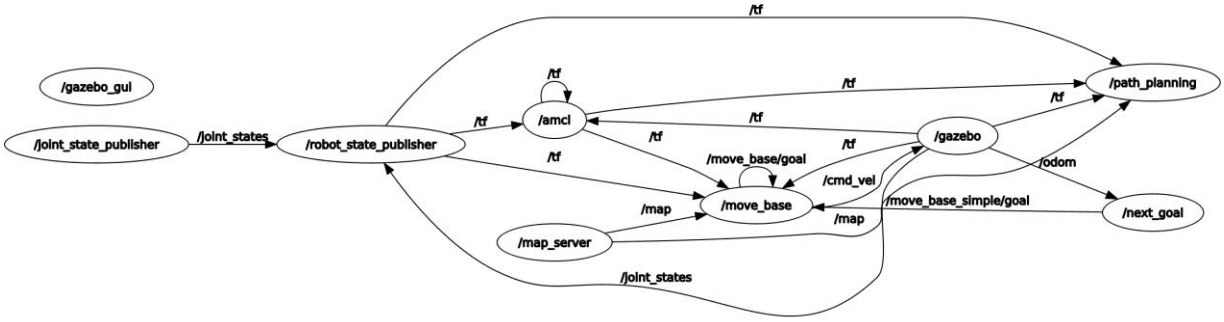

图 3.3.2 节点拓扑图

四、结果分析

通过上述过程，我们成功在 Gazebo 中进行了仿真，仿真效果见：https://www.bilibili.com/video/BV1Fe41117gR?vd_source=e67cc43f2e8443b722a5f50ef79db03e。

本扫地机器人代码见：https://github.com/scybd/Sweeping-Robot。

五、创新实践总结

在本次创新实践中，我接触并学习了 ROS、Gazebo、OpenCV 和 C++等知识，对 ROS 中的各功能包有了一定了解，对机器人的仿真有了更深刻的认识。