## **1. 概述**

ExplorerNode 是一个基于 ROS 2 的节点，用于实现机器人自主探索和地图构建，结合 SLAM（通过 /map 话题获取 OccupancyGrid）和 Nav2 导航栈（通过 NavigateToPose 动作）。核心功能是通过前沿（frontier）检测算法，引导机器人探索未知区域并动态更新地图。

## 要求

ROS 2 (Humble/Foxy)

Python 3

TurtleBot3 packages installed

Nav2 installed and configured for your robot

SLAM Toolbox installed

numpy Python library

**仿真启动流程**

ros2 launch turtlebot3\_gazebo [turtlebot3\_world.launch.py](5kop52)

ros2 launch nav2\_bringup navigation\_launch.py use\_sim\_time:=True

ros2 launch slam\_toolbox [online\_async\_launch.py](rpk7r8)

ros2 launch nav2\_bringup [rviz\_launch.py](36oe1j)

ros2 run custom\_explorer explorer

## **2. 处理流程**

以下是 ExplorerNode 的完整处理流程，聚焦于 explore 函数的周期性执行逻辑。

### **2.1 总体流程**

1. **初始化**：

创建节点，初始化日志、订阅器、动作客户端和定时器。

设置空地图数据（self.map\_data）、机器人位置占位符（self.robot\_position= (0, 0)）、已访问前沿集合（self.visited\_frontiers）。

1. **地图数据接收**：

通过 /map话题订阅 OccupancyGrid消息，调用 map\_callback更新 self.map\_data。

**周期性识别定位：**

探索过程中通过定时器周期性识别camera内容，判断是否发现物块。

若没有发现，退出继续探索；

若发现物块，定位，坐标给nav2导航；

判断是否达到目标点：

若达到目标点，关闭识别定时器；

将action传给servo\_node，进行抓取；

小车进入等待状态，并获取反馈和抓取结果：

若抓取成功，周期性检查抓取状态，nav2到放置位置；  
 若抓取失败，重新抓取。

若没有达到目标点，通过定时器周期更新物块坐标进行导航。

1. **周期性探索（每 5秒触发 explore）**：

检查地图数据是否可用。

将地图转换为二维 NumPy数组。

检测前沿（自由格子旁有未知格子的区域）。

选择最近的未访问前沿。

将前沿格子坐标转换为世界坐标。

通过 Nav2发送导航目标。

等待导航结果，重复上述步骤直到无前沿。

1. **导航执行**：
   1. 使用 navigate\_to函数构造 PoseStamped目标，发送到 Nav2的 NavigateToPose动作服务器。
   2. 通过回调函数（goal\_response\_callback和 navigation\_complete\_callback）处理目标接受和导航结果。
2. **终止条件**：
   1. 当无前沿可探索时，记录“探索完成”日志（未实现具体的终止逻辑）。

### **2.2 详细流程（explore 函数）**

以下是 explore 函数的步骤分解：

1. **检查地图数据**：

输入：self.map\_data（OccupancyGrid类型）。

输出：若为空，记录警告并退出；否则继续。

1. **地图转换**：

输入：self.map\_data.data（一维占用值数组）、self.map\_data.info.height/width。

输出：二维 NumPy数组 map\_array（值：-1未知，0自由，100占用）。

1. **前沿检测**：

输入：map\_array。

输出：前沿坐标列表frontiers（每个元素为(row,col)）。

调用：self.find\_frontiers(map\_array)。

1. **前沿选择**：

输入：frontiers、机器人位置self.robot\_position、已访问集合self.visited\_frontiers。

输出：最近的未访问前沿chosen\_frontier（(row,col) 或 None）。

调用：self.choose\_frontier(frontiers)。

1. **坐标转换**：

输入：chosen\_frontier、地图分辨率self.map\_data.info.resolution、地图原点self.map\_data.info.origin.position.x/y。

输出：世界坐标 (goal\_x,goal\_y)。

1. **导航命令**：

输入：(goal\_x,goal\_y)。

输出：发送 NavigateToPose动作目标。

调用：self.navigate\_to(goal\_x,goal\_y)。

## **3. 接口定义**

### **3.1 输入接口**

1. **/map话题**：

**类型**：nav\_msgs.msg.OccupancyGrid

**描述**：SLAM算法生成的地图，包含：

* + 1. data：一维数组（-1未知，0自由，100占用）。
    2. info：地图元数据（resolution、width、height、origin）。

**订阅器**：self.map\_sub（回调：map\_callback）。

**依赖**：SLAM系统正常运行，发布动态更新的地图。

### **3.2 输出接口**

1. **navigate\_to\_pose动作**：

**类型**：nav2\_msgs.action.NavigateToPose

**描述**：发送导航目标，包含PoseStamped（目标位置x,y 和朝向）。

**动作客户端**：self.nav\_to\_pose\_client。

**依赖**：Nav2栈正常运行，动作服务器可用。

**回调**：

goal\_response\_callback：处理目标是否被接受。

navigation\_complete\_callback：处理导航结果。

### **3.3 内部接口**

1. **前沿检测（find\_frontiers）**：

**输入**：map\_array（二维NumPy数组）。

**输出**：frontiers（列表，元素为(row,col)）。

**逻辑**：遍历地图，寻找自由格子（0）旁有未知格子（-1）的单元。

1. **前沿选择（choose\_frontier）**：

**输入**：frontiers（前沿列表）、self.robot\_position（机器人位置）、self.visited\_frontiers（已访问集合）。

**输出**：chosen\_frontier（(row,col) 或 None）。

**逻辑**：计算机器人到每个前沿的欧几里得距离，选择最近的未访问前沿。

1. **导航命令（navigate\_to）**：

**输入**：(x,y)（世界坐标）。

**输出**：发送NavigateToPose动作目标。

**逻辑**：构造 PoseStamped，通过动作客户端发送目标。

**可能存在的迁移问题：**

**机器人位置更新：**

代码中 self.robot\_position硬编码为 (0,0)，需订阅定位数据（如 /amcl\_pose或 /odom）。

**地图话题适配：**

确认 /map话题的发布者（SLAM节点）与新机器人兼容。

如果话题名不同，修改订阅器

**导航接口调整：**

验证 NavigateToPose动作服务器可用性。

如果新机器人使用不同动作或话题（如/move\_base），需重写navigate\_to函数