

自主建图功能使用与讲解

1. 功能简介

自主建图功能又称为 RRT 建图功能，顾名思义就是使用 RRT 算法来进行自主建图。RRT（Rapidly-Exploring Random Trees）即快速随机扩展树，它是一种单一查询算法，一边抽样一边建图，通过不断随机探索来实现遍历。RRT 算法概率完备，可用于进行路径规划与地图边界探索，但因为其随机性比较强，规划的路径并不一定是最优解，在我们的源码中 RRT 算法主要是用于进行边界的探索，通过将地图边界点作为目标点来实现小车自主导航探索，同时开启建图节点，实现自主建图。关于 RRT 算法的详细介绍这里不作解说。

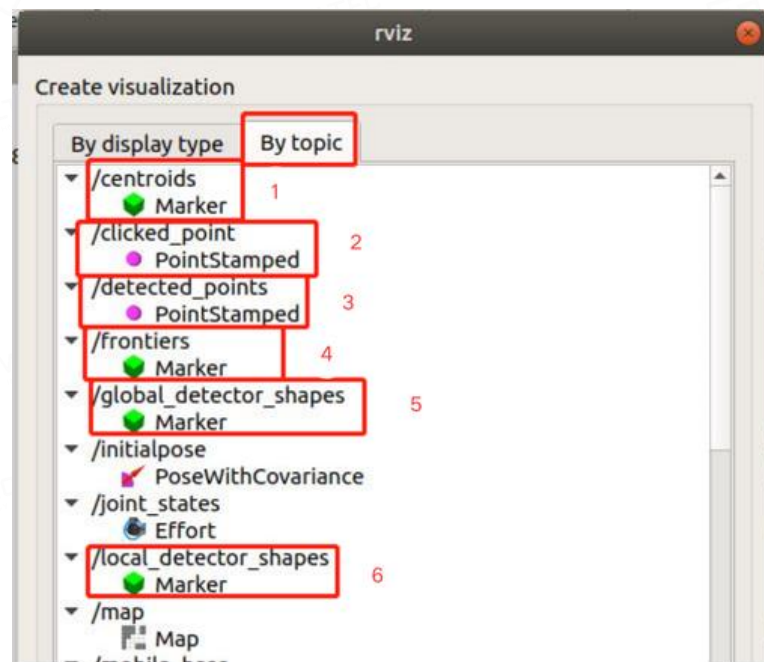
2. 使用方法

- 1) 打开终端远程登录后输入启动自主建图功能的 launch 文件指令

```
roslaunch turn_on_wheeltec_robot rrt_slam.launch
```

- 2) 虚拟机小车端打开 rviz 可视化工具并进行配置

打开 rviz 后，首先要进行 rviz 的配置，在原有的基础配置上，点击 rviz 界面左下角的 ADD，选择 By topic，添加图中框选的两个话题，（我们所提供的虚拟机镜像都是已经配置好的，用户无需重复配置）



rviz 配置

这六个话题分别的含义是：

centroids: 滤波后的有效边界点

clicked_points: 随机树的范围点和起点

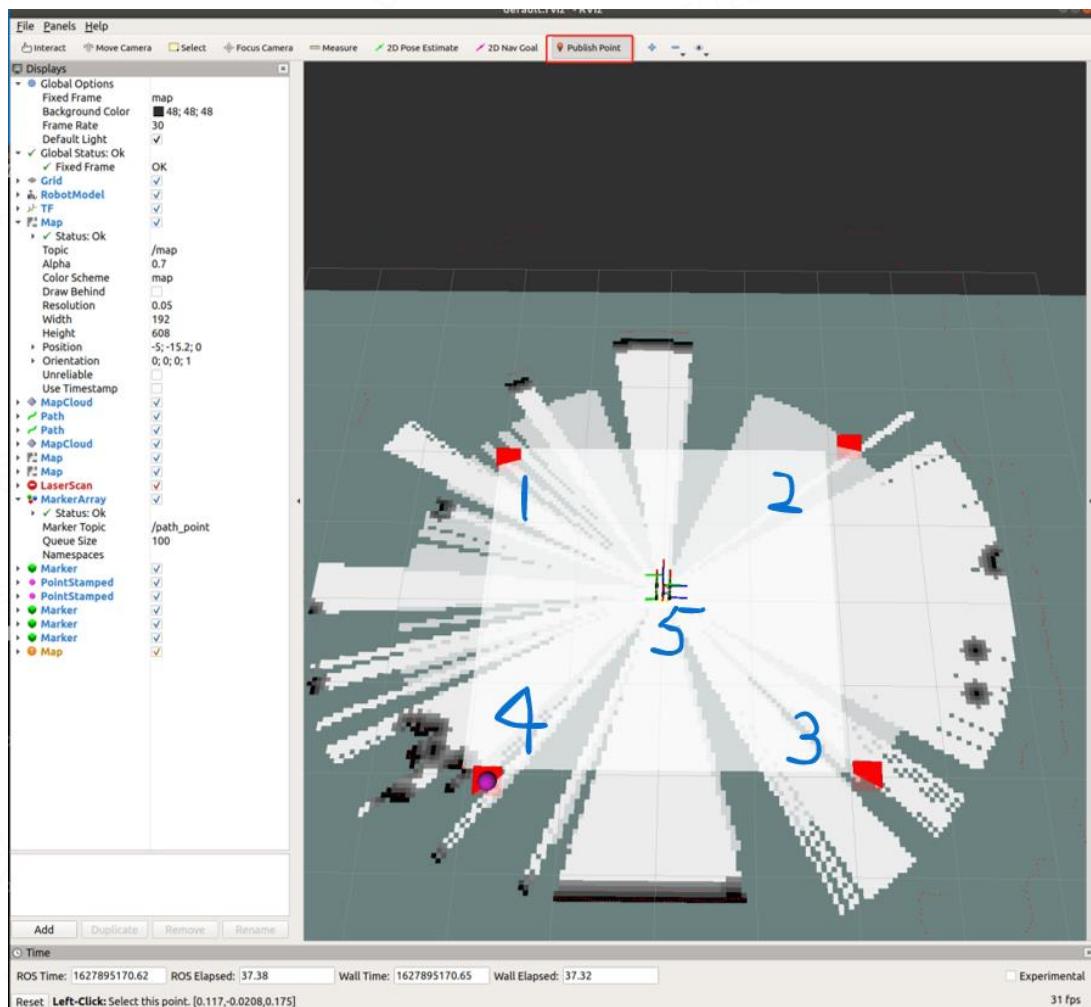
detected_points: 检测到的边界点

frontiers: 滤波器接收到的边界点

global_detector_shapes: 全局树

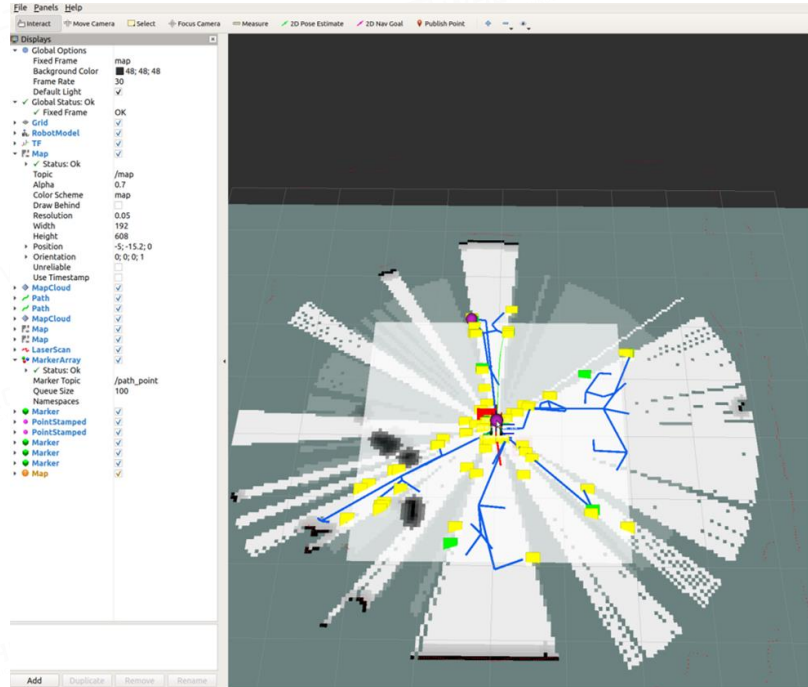
local_detector_shapes: 本地树

添加完话题后，就可以开始操作了。首先使用 rviz 中的 publish point 工具顺时针或逆时针定下四个点，这四个点所包围的区域就是所定下的建图区域，注意各点按顺序连线后需要包围成一个区域，因此要以顺时针或逆时针顺序来定下范围点。定好四个点后，用户还需要定下第五个点也就是最后一个点，最后一个点是随机树的起点，一般选择定在小车车身上，当雷达扫描范围有部分选择屏蔽时，需将起点定在距离小车较近处、雷达可扫描范围内。



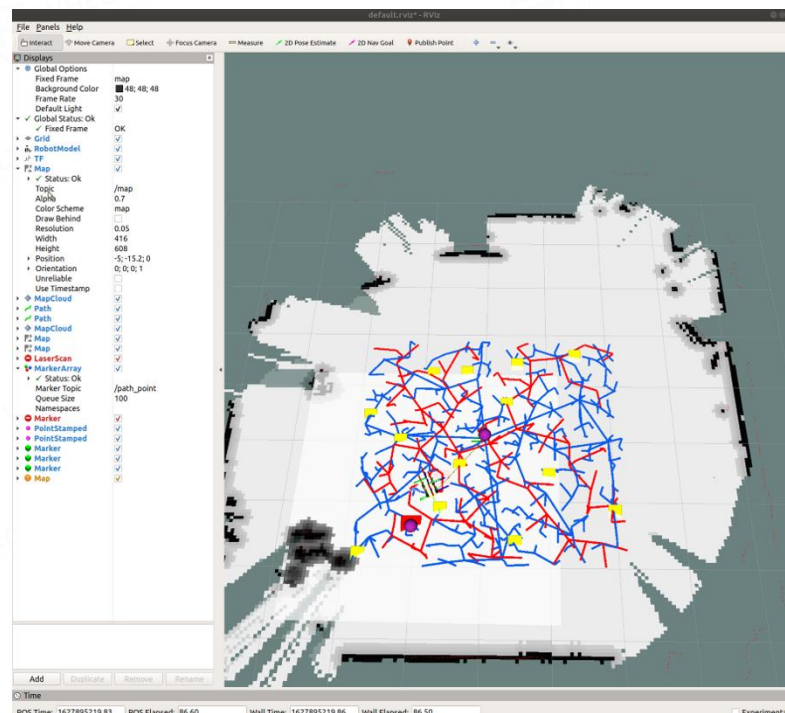
选择建图范围点

定下最后一个点后，随机树开始生长，小车开始进行导航来对地图边界进行探索，建图过程中蓝色树状为全局树，红色树状为本地树，黄色点为滤波器接收到的边界点，绿色点为边界点。



定下随机树起点后，随机树开始生长

3) 完成探索后，小车将会停止运动，此时开始进行计数，若 20 秒内小车都保持静止，则认为自主建图已经完成，启动保存地图节点。

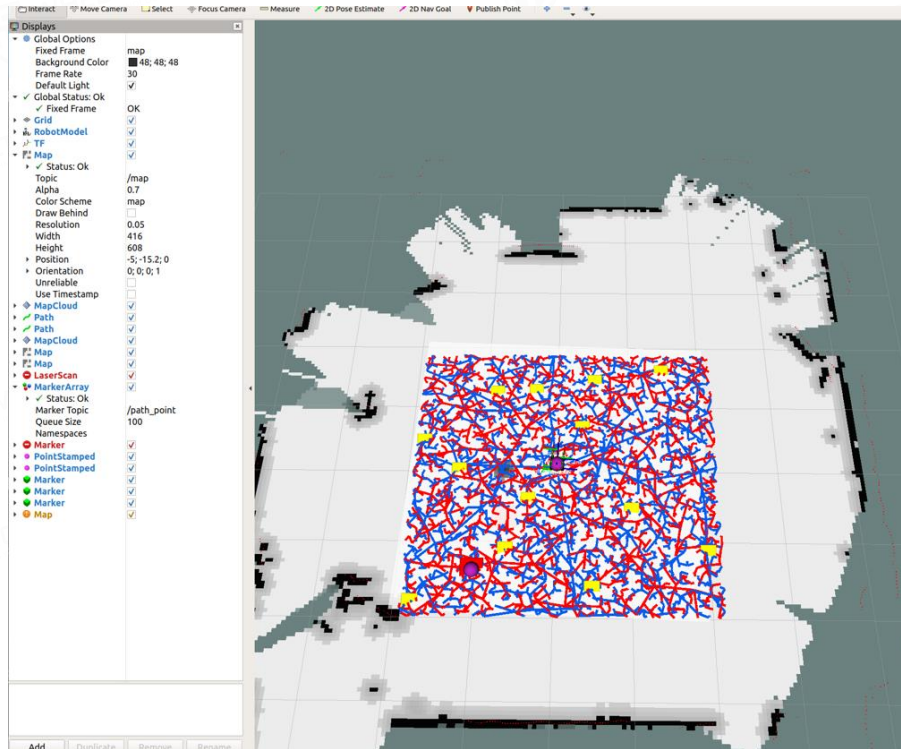


小车停止运动，开始计时

启动保存地图节点：

```
roslaunch turn_on_wheeltec_robot map_saver.launch
```

保存地图后，小车会等待几秒钟，然后又重新运动回自主建图起始位置，至此，自主建图就完成了。



小车回到建图起点，自主建图流程结束

3. 注意事项

① 范围点选择问题

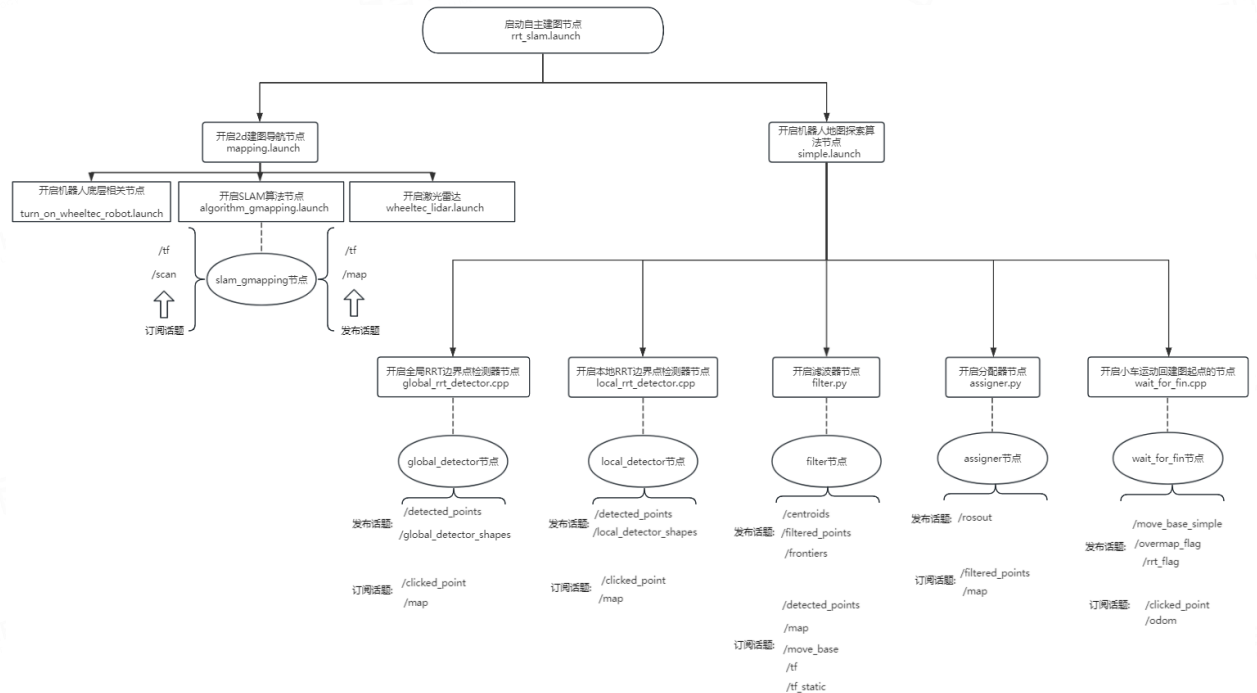
一定要以顺时针或者逆时针的顺序来定下范围点，如果没有按照顺时针或逆时针顺序选取范围点，则所选范围点无法围成闭合图形，无法启动自主建图。如果无法选择范围点，有可能是 rviz 没有配置好或者多机通信没有配置好，rviz 配置中六个话题都需要显示出来才能看到效果。

② 地图保存问题

注意，若保存地图节点运行窗口还未弹出就退出程序，则所建地图无法被保存，同时因为存在一定的延时，所以保存地图与回到原点也比我们所定下的默认时间稍有延迟，具体视各主控运行情况而定，部分主控需要耐心等待，在等待的二十秒中，如果小车发现新的仍需进行探索的边界点，则计数清零，小车再次进行探索，重新静止后再从零开始计数。

4. 自主建图功能程序讲解

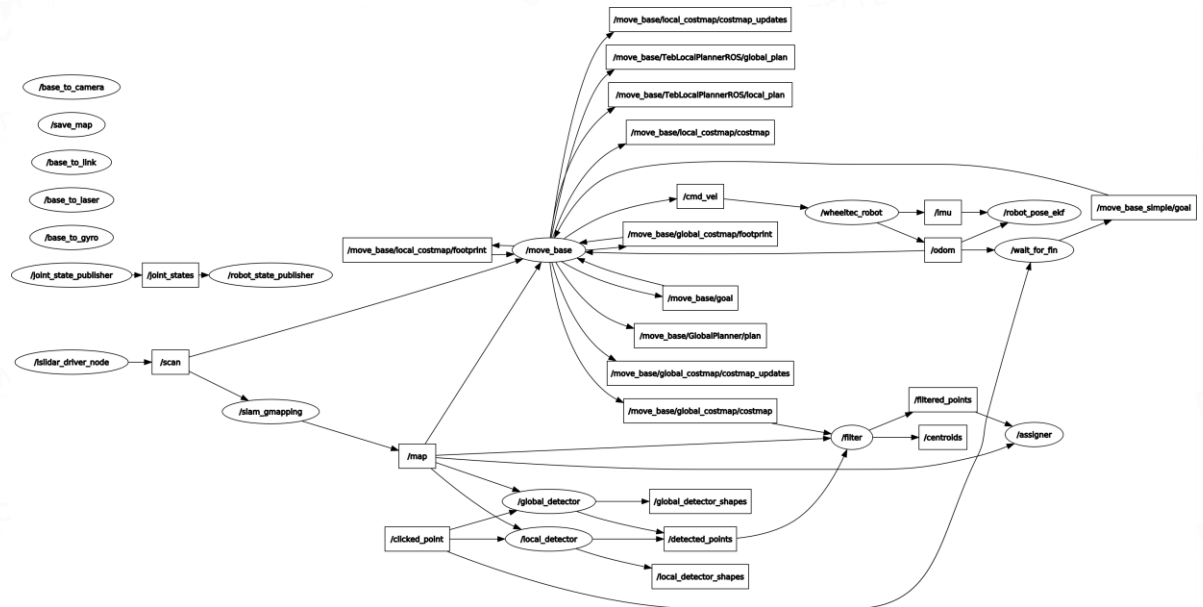
① 自主建图功能启动框架：



自主建图功能启动框架图

② 查看自主建图功能节点：

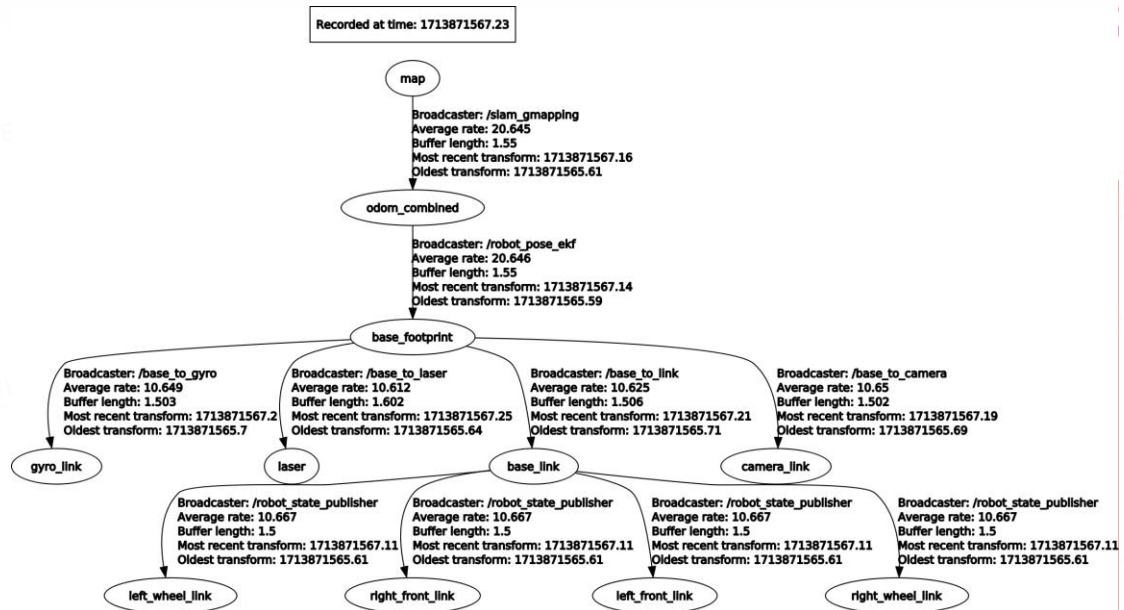
rqt_graph



自主建图功能节点关系图

③ 查看自主建图功能 TF 树：

roslaunch rqt_tf_tree rqt_tf_tree



自主建图功能 TF 树

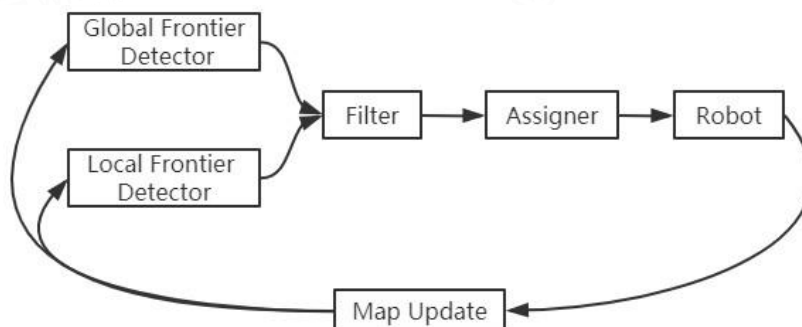
④ 自主建图 launch 文件解析:

使用 RRT 自主建图功能时，我们所运行的是 `rrt_slam.launch` 启动文件，该 launch 文件的重点在于所嵌套的另一个 launch 文件：`simple.launch`。

`simple.launch` 中调用了 `rrt_exploration` 功能包中的一些节点，`rrt_exploration` 是一个为移动机器人实现多机器人地图探索算法的 ROS 包，它基于 RRT 算法，使用占用网格作为地图表示。该包有 5 个不同的 ROS 节点：

- 全局 RRT 边界点检测器节点；
- 本地 RRT 边界点检测器节点；
- 基于 `opencv` 的边界检测器节点；
- 滤波器节点；
- 分配器节点。

节点类型可分为三种：用于在占用网格地图中检测边界点的节点，用于过滤被检测点的节点，以及用于将点分配给机器人的节点。



三种节点的关系图

这三类节点与机器人的关系：由检测边界点的节点检测到的边界点会给出过滤器节点，过滤器节点会滤掉部分重复无效的边界点，过滤后的边界点又给出了分配器，当多个机器人协同进行建图时，边界点会被分配器节点就近分配给对应的机器人进行探索，探索过程中，地图也会不断更新，不断检测新的边界点，最终循环检测探索从而完成建图。关于 `rrt_exploration` 功能包的详细介绍可以查看对用的 ROS wiki 页面：http://wiki.ros.org/rrt_exploration

在我们的 ROS 源码中，我们没有使用到 OpenCV 节点，使用了另外四个节点来完成地图的探索，各节点的作用已经在前文简要介绍，此处不再赘述，在完成建图后，我们还调用了一个 `wait_for_fin` 节点，这个节点的作用是在建图结束后运行保存地图的节点并使小车运动回建图起点，

```
//turn_on_wheeltec_robot——simple.launch:
<node pkg="rrt_exploration" type="wait_for_fin" name="wait_for_fin"
output="screen">
<param name="waiting_time" value="20"/>
</node>
```

在 launch 文件中，我们设定的默认等待时间为 20 秒，20 秒内若小车没有再次运动，就认为建图完成并启动地图保存节点，并在几秒后发布建图起点坐标数据，小车再运动回建图起点。

```
//rrt_exploration——wait_for_fin.cpp:
if(count == waiting_time)
{
    system("dbus-launch gnome-terminal -- roslaunch
turn_on_wheeltec_robot map_saver.launch");//保存地图节点
    system("dbus-launch gnome-terminal -- rosnode kill /assigner");
}

if(count == (waiting_time+5))
{
    back_pub.publish(origin);//发布建图起点坐标数据
    std_msgs::Int8 overmap_flag_msg;
    overmap_flag_msg.data = 1;
    overmap_flag_pub.publish(overmap_flag_msg);//建图完成标志位发布
    printf("a=%d", overmap_flag_msg.data);
    break;
}
```

调用 `rrt` 算法后，我们还需要开启建图节点来实现自主建图，同时传入参数值置为 `true` 的参数 `navigation`，同时开启建图与导航节点。

<!-- 开启建图的同时也开启导航 -->

```
<include file="$(find turn_on_wheeltec_robot)/launch/mapping.launch" >
```

```
  <arg name="navigation" value="true"/>
```

```
</include>
```