3D 导航功能使用与讲解

1. 功能简介

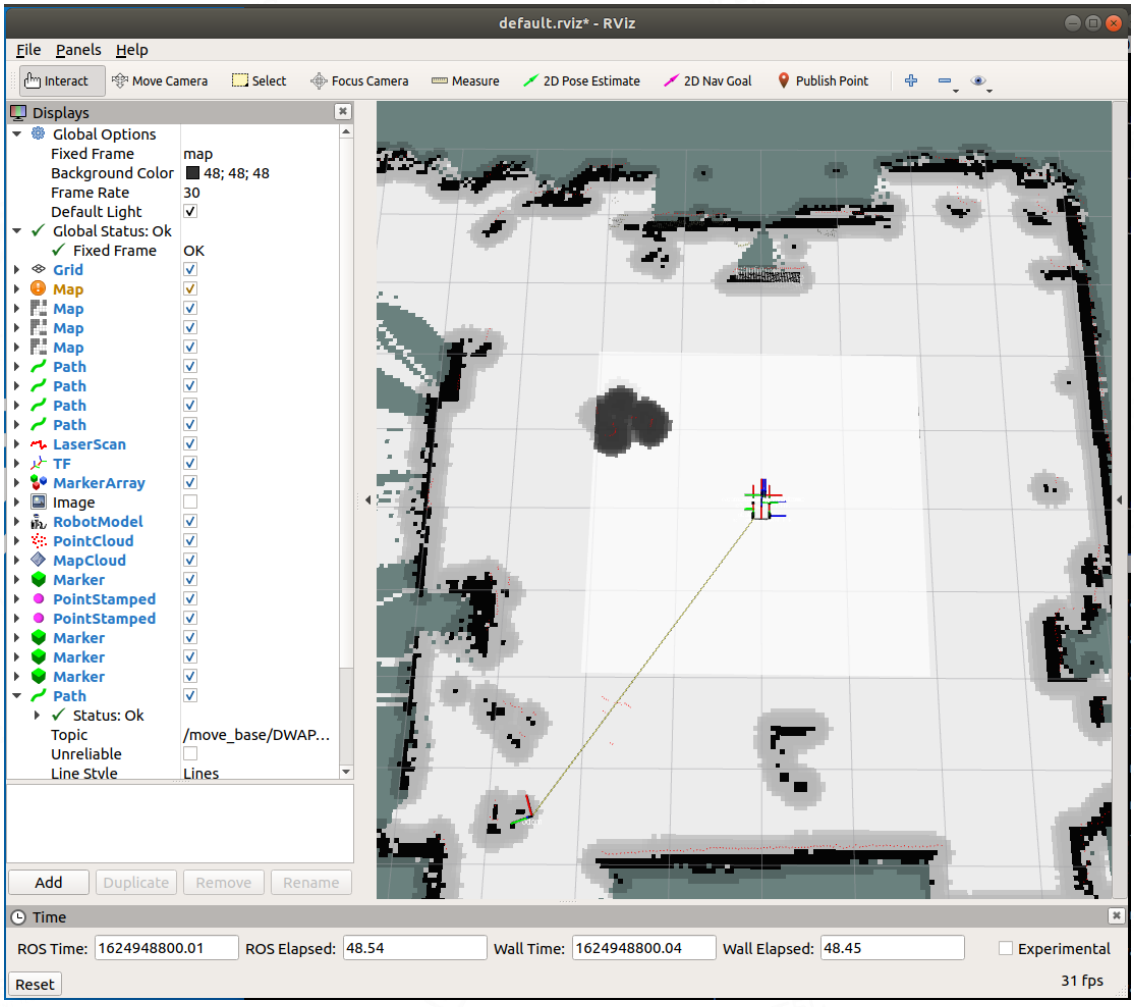
3D 导航功能主要是与建立的三维地图进行比对，小车确定自己在三维空间中的位置从而实现 3D 定位，常见的方法有基于特征的定位和全局定位。小车知道了起始点和目标点的位置，就可以使用 3D 地图中的信息进行路径规划。

2. 使用方法

1) 建立并保存 3D 地图后，就可以运行 3D 导航节点了，打开终端输入运行 launch 文件开启 3D 导航节点

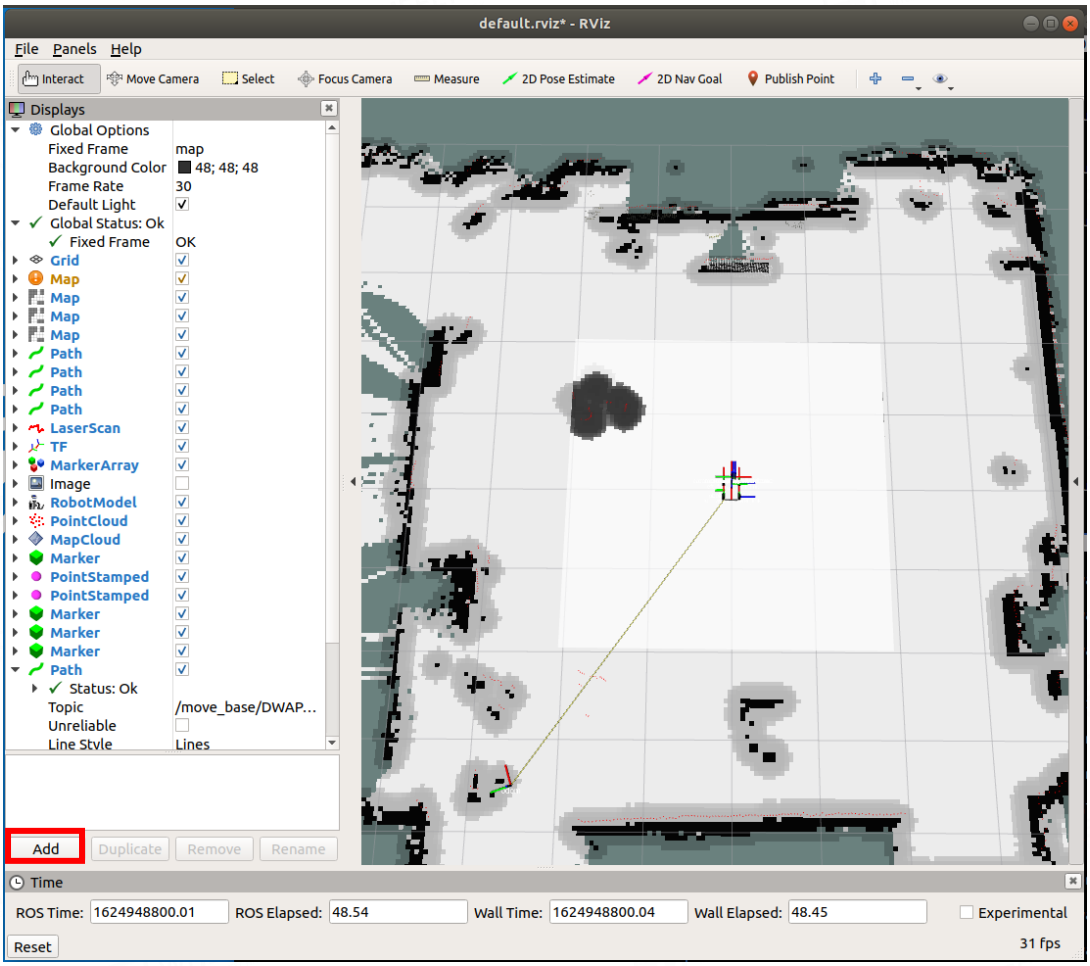
```
roslaunch turn_on_wheeltec_robot 3d_navigation.launch
```

2) 待 3D 导航节点完全开启后再运行虚拟机小车端的 rviz，在 3D 建图步骤中所配置好的 rviz 基础上，此时我们就可以在 rviz 中看到刚刚所建的地图

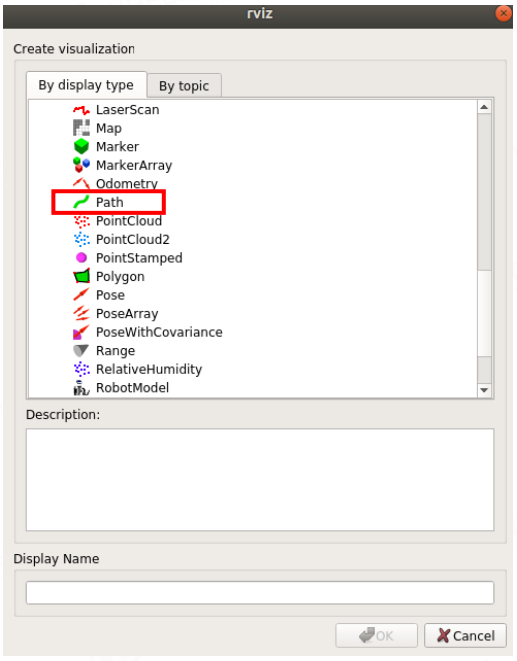


3D 导航节点运行时的 rviz 显示界面

除此之外，为了显示导航过程中小车的路径规划和运动轨迹，我们还需要在此 rviz 配置基础上添加两个“Path”选项，点击左下角“Add”按钮

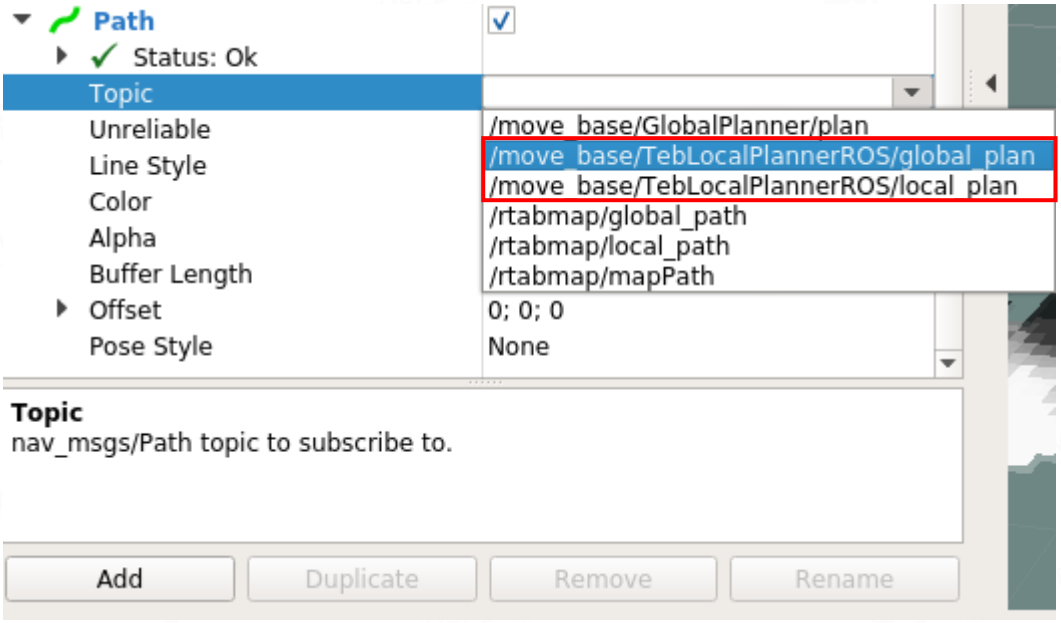


添加“Path”选项步骤一

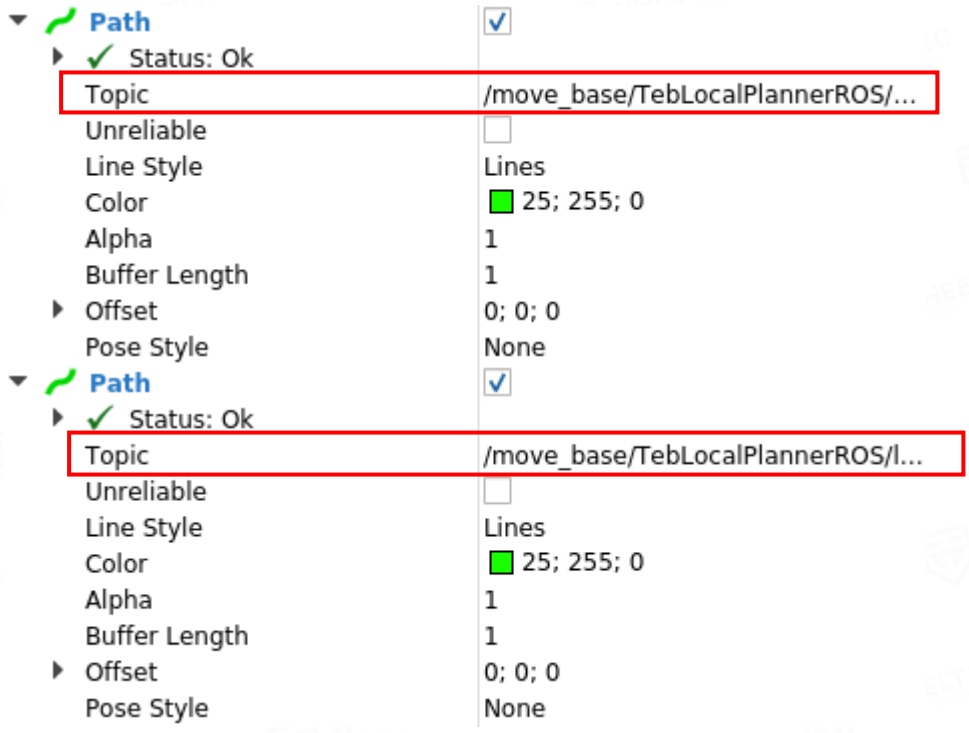


添加“Path”选项步骤二

“Path”选项中的 Topic 选择如图所示，分别代表着 Teb 算法的全局路径信息和局部路径信息。



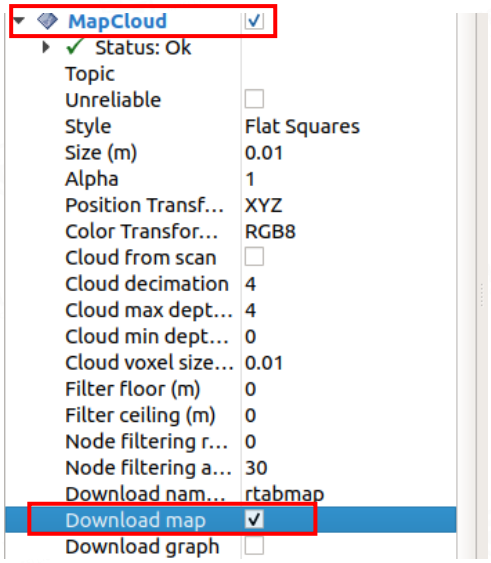
“Path”的话题选择



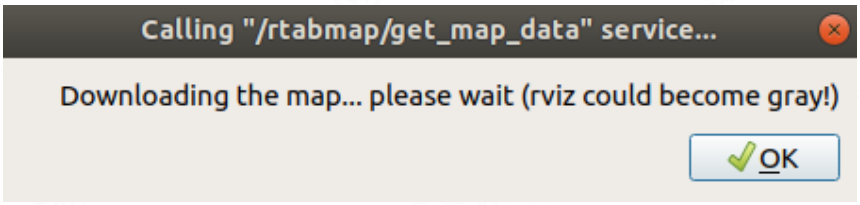
“Path”话题选择

3D 导航所用的是 3D 建图时建立的 2D 图层，而 3D 导航时的点云图像是导航时摄像头所接收到的实时信息，也就是实时画面，在小车进行导航的过程中，会不断的在地图上显示最新的点云信息，如果想显示之前进行 3D 建图时

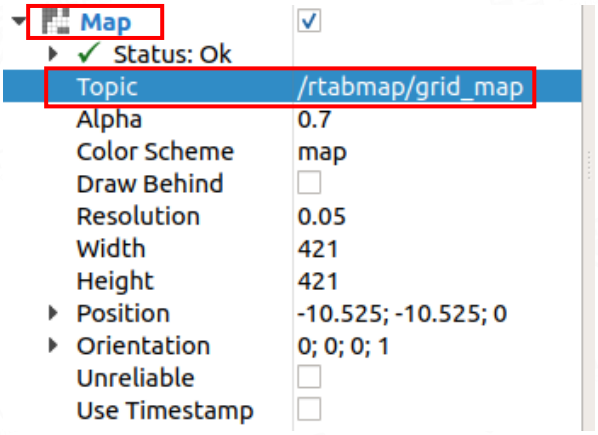
所存储的摄像头点云信息，可以在 rviz 左侧找到“MapCloud”选项并展开，点击“Download map”，此时会弹出提示框提示正在加载，需要耐心等待几分钟，加载完毕就可以在 rviz 中看到之前的摄像头点云信息了，注意此时该点云信息对导航功能完全没有影响，不起任何作用，仅用于显示点云图像。其中一个“Map”选项中的 Topic 话题选择为“/rtabmap/grid_map”用来显示之前建好的地图。



勾选上“Download map”



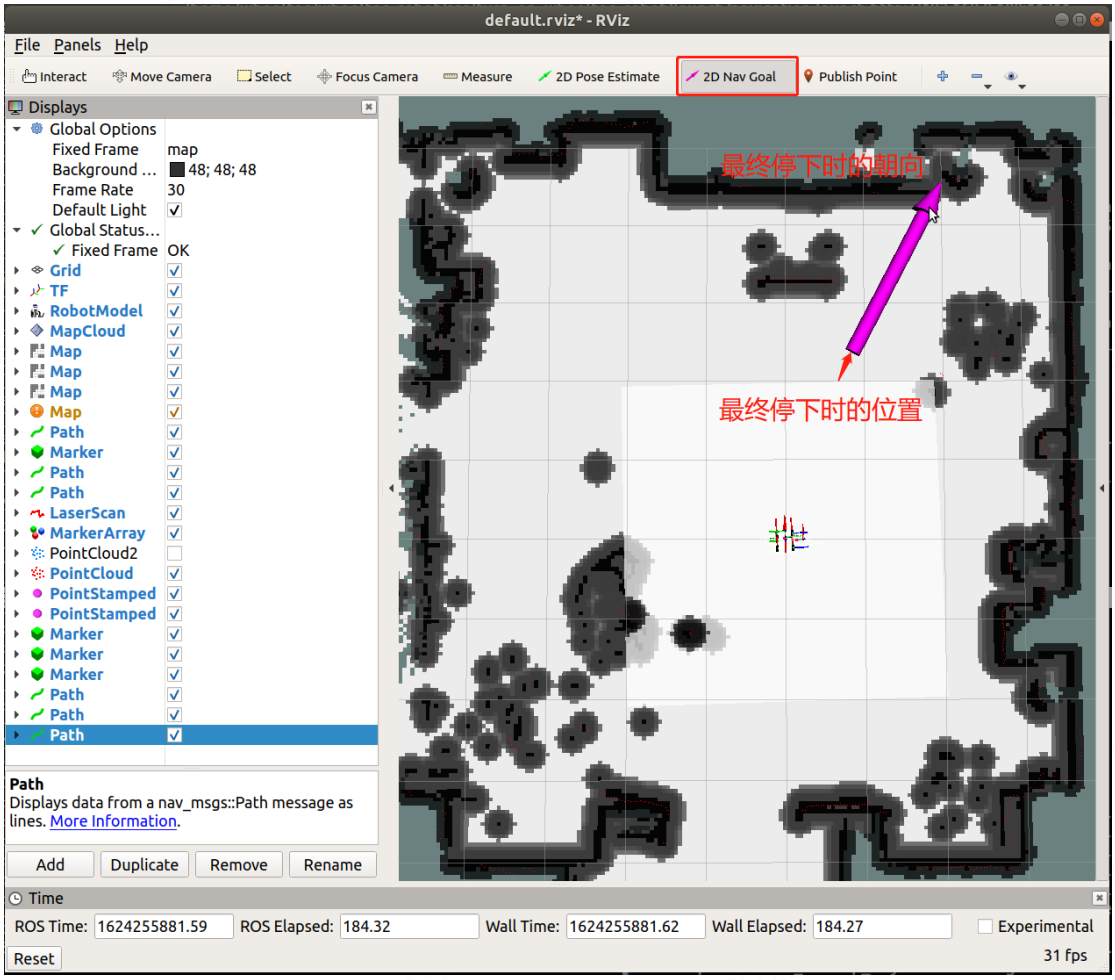
弹出提示框



“Map”话题选择

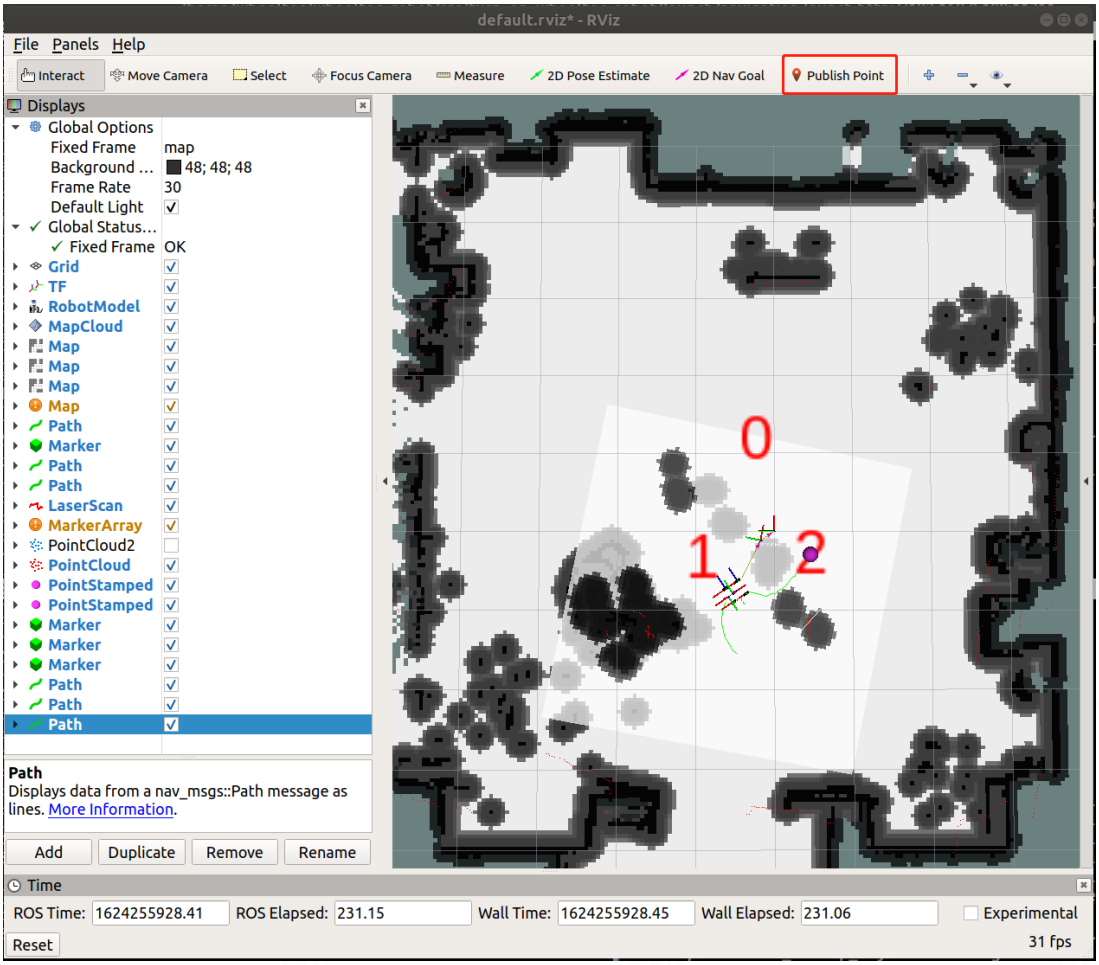
3) 单点导航与多点导航

3D 导航同样有两种方式，单点导航与多点导航，操作方法与 2D 导航大致相同，分别使用“2D Nav Goal”和“Publish Point”实现，在单点导航时，点击 rviz 上的 2D Nav Goal 按键，然后在地图中选定导航点与方向，单击鼠标确定导航目标点位置，并且不要松开，继续拖动选择导航目标方向，确定后松手，小车就会运动到所指定的目标点，在运行 rviz 的终端也会显示目标点的坐标。



3D 导航功能——单点导航

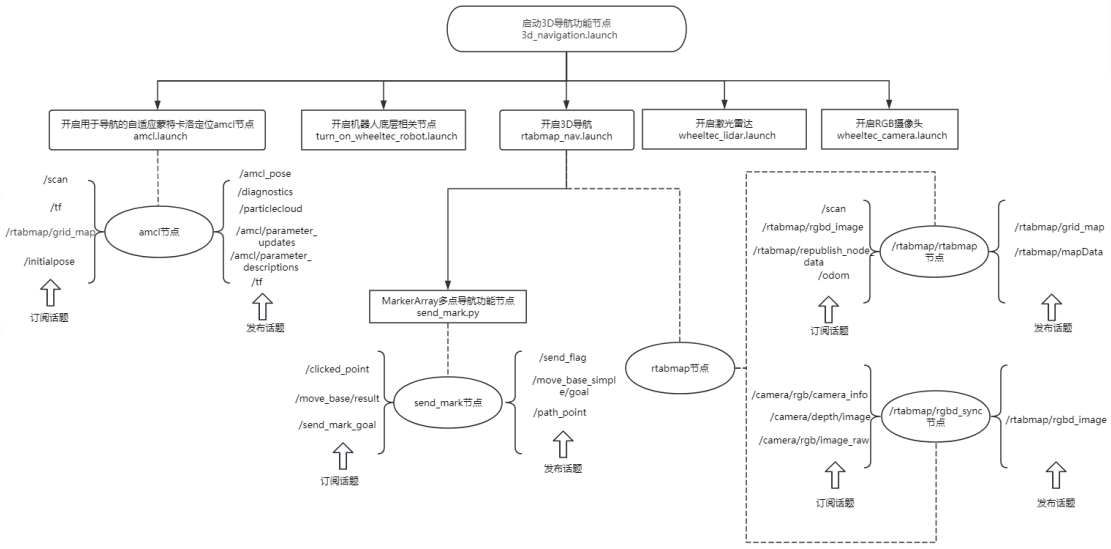
另一种是多点导航，点击 rviz 界面上方的 Publish Point 就可以设置用于多点导航的点了，当设置多个点时，小车会在这几个点之间往复运动，使用 Publish Point 工具进行多点导航时，所设置的导航点默认方向都是小车的初始车头方向。



3D 导航功能——多点导航

3. 3D 导航功能程序讲解

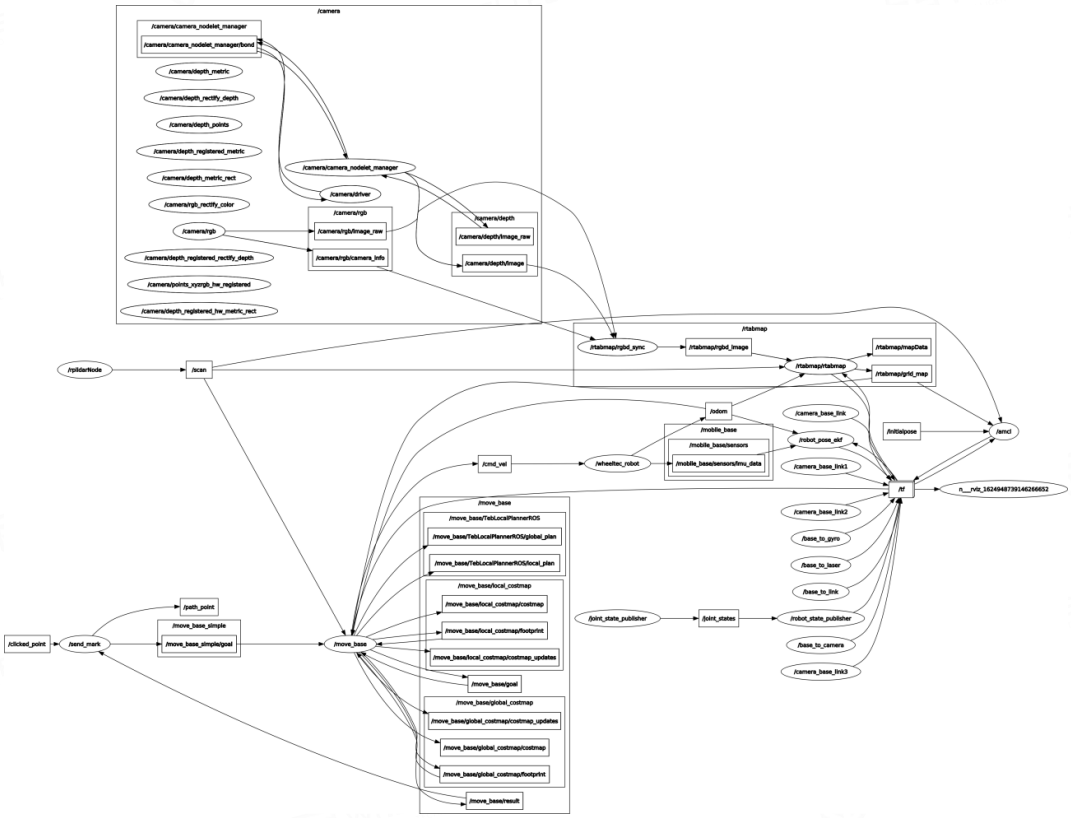
① 3D 导航功能启动框架：



3D 导航功能启动框架图

② 查看 3D 导航功能节点：

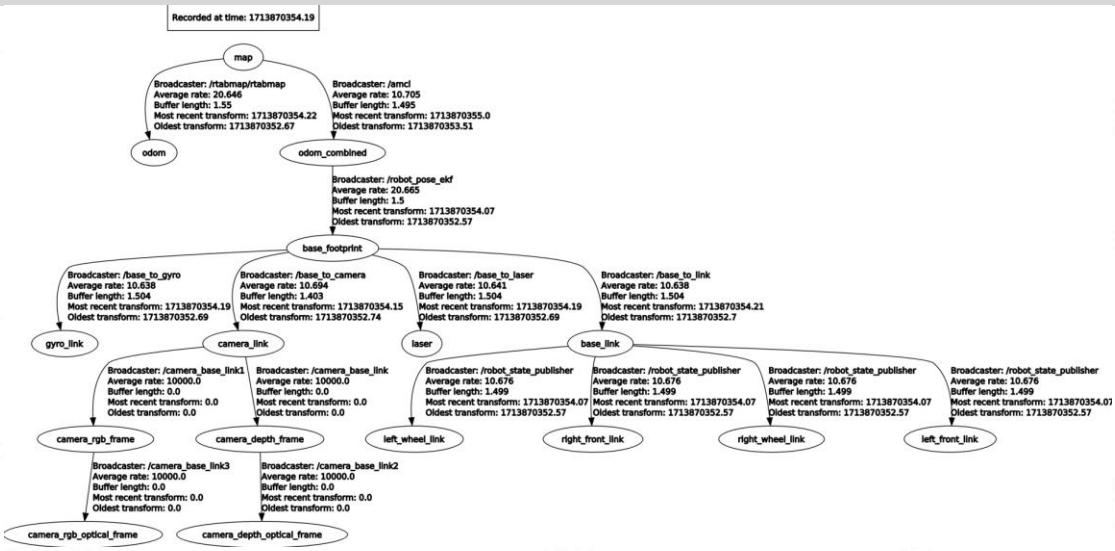
```
rqt_graph
```



3D 导航功能节点关系图

③ 查看 3D 导航功能 TF 树：

```
rosrun rqt_tf_tree rqt_tf_tree
```



3D 导航功能 TF 树

④ 3D 导航 launch 解析：

3D 导航节点的开启是靠启动 turn_on_wheeltec_robot 功能包下的

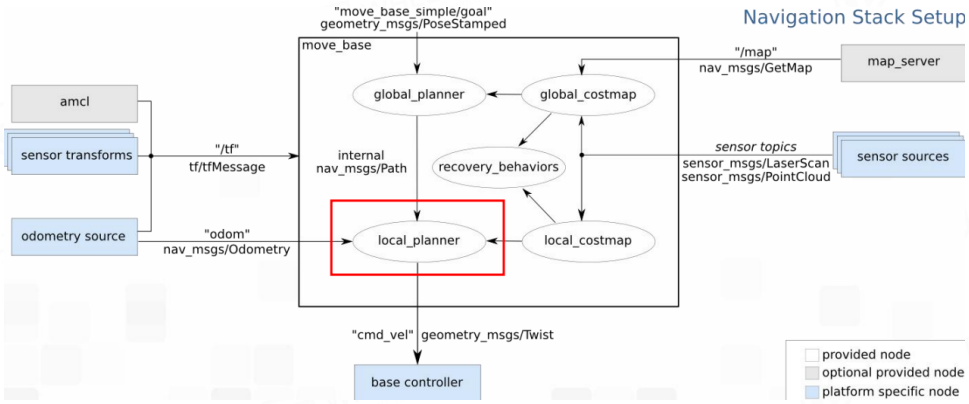
3d_navigation.launch 文件来实现的，launch 文件内容如下：

```
<launch>
 <!-- 开启机器人底层相关节点 同时开启导航功能 -->
  <include file="$(find
turn_on_wheeltec_robot)/launch/turn_on_wheeltec_robot.launch" >
    <arg name="navigation"  default="true"/>
  </include>

  <!-- turn on lidar 开启雷达  -->
  <include file="$(find turn_on_wheeltec_robot)/launch/wheeltec_lidar.launch"
/>
  <include file="$(find turn_on_wheeltec_robot)/launch/include/amcl.launch" >
   <arg name="use_map_topic" value="true"/>
   <arg name="map_topic" value="/rtabmap/grid_map"/>
  </include>

  <!-- 开启摄像头 -->
  <include file="$(find turn_on_wheeltec_robot)/launch/wheeltec_camera.launch"
/>

  <!-- 开启 3d 导航 -->
  <include file="$(find
turn_on_wheeltec_robot)/launch/include/rtabmap_nav.launch" />
  <param name="move_base/global_costmap/static_layer/map_topic"   type="string"
value="/rtabmap/grid_map"/>
  <param name="move_base/local_costmap/static_layer/map_topic"    type="string"
value="/rtabmap/grid_map"/>

</launch>
```

首先是开启底层初始化节点，在初始化节点中需先设置导航算法，默认为 teb 算法。对于 teb 算法的解释，可参考“局部路径规划-TEB”的 PDF 文档。

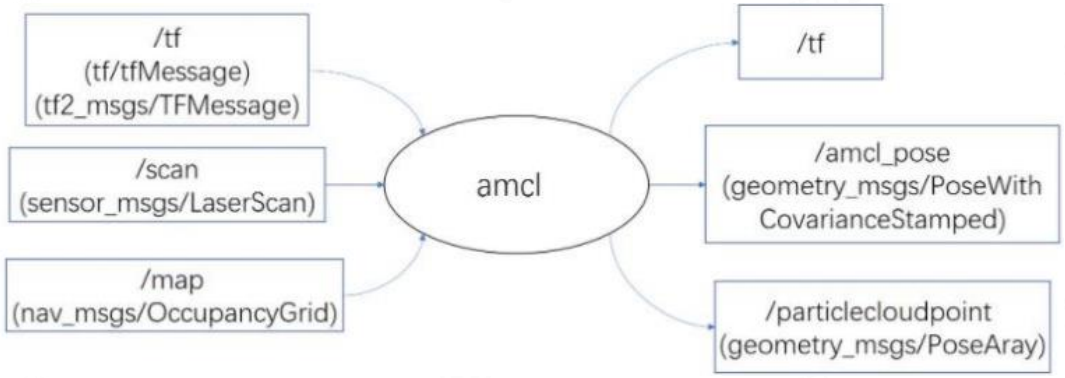
```
</launch>
<!--当开启导航功能时 启用导航算法选择-->
  <!--当开启 2D 或 3D 导航功能时-->

  <!-- 开启 teb_local_planner 导航算法 与 dwa 算法相比效果更佳-->
  <include file="$(find
turn_on_wheeltec_robot)/launch/include/teb_local_planner.launch" if="$(arg
navigation)">

  <!-- 开启 dwa_local_planner 导航算法-->
  <!-- <include file="$(find
turn_on_wheeltec_robot)/launch/include/dwa_local_planner.launch" if="$(arg
```

```
navigation)"> -->
    <arg name="car_mode" value="$(arg car_mode)"/>
  </include>
```

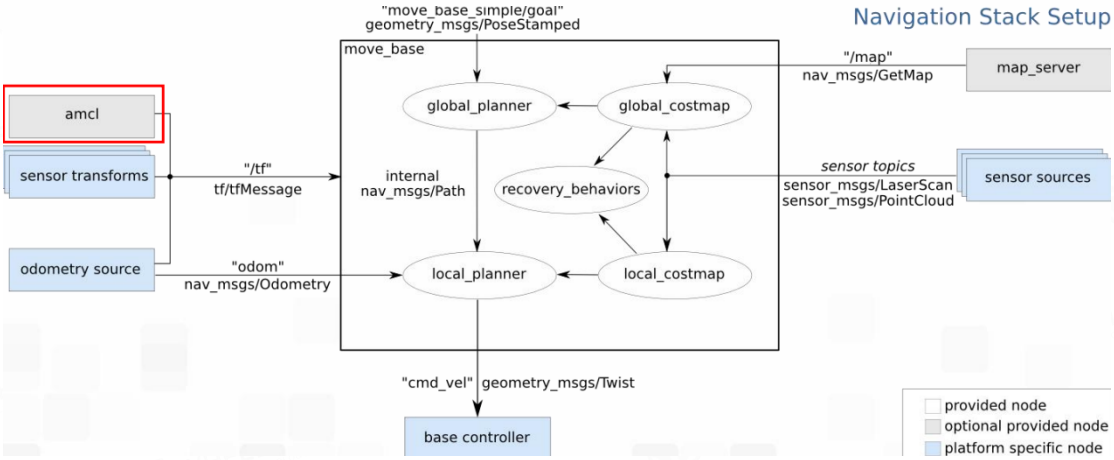


local_planner 节点关系图

接着是开启雷达节点、用于导航的蒙特卡洛定位以及摄像头。其中重要的有 amcl.launch 这个文件，开启用于导航的自适应蒙特卡洛定位 amcl 节点。amcl 功能包主要作用是补偿定位累计误差。订阅小车位置(TF 坐标 base_footprint)和雷达信息，计算小车在地图上的位置偏差，然后发布 map 到 odom_combined 的 TF 转换。amcl 根据雷达信息计算 base_footprint 在 map 上的位置误差，再通过在 map 和 base_footprint 中间添加一个 TF：odom_combined 来进行误差补偿。



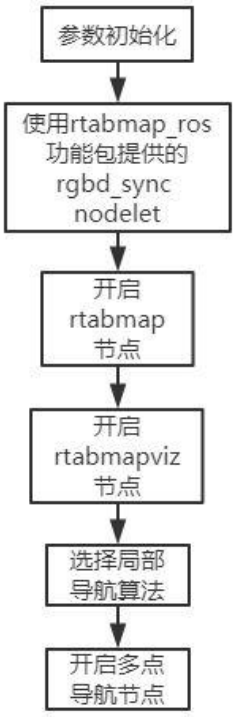
amcl 订阅和发布的话题



navigation 工作框架

最后是开启 3D 导航。前面的部分都是前文重复讲过的，接下来主要讲解最后开启 3D 导航的这一部分。

3D 导航的启动文件为 turn_on_wheeltec_robot 路径下的 rtabmap_nav.launch，



rtabmap.launch 源码流程图

首先是参数的初始化，下面这一段的意义是以 test.db 中存储的地图数据作为导航所用地图的数据，如果我们事先运行了 3D 建图，此时所用地图就是刚才所建的地图。

```
<arg name="localization" default="true"/>
<arg name="database_path" default="~/.ros/test.db"/>
```

```
<arg     if="$(arg localization)" name="rtabmap_args" default=""/>
<arg unless="$(arg localization)" name="rtabmap_args" default="-d"/>
```

接下来依旧是使用了 rtabmap 功能包提供的 nodelet 中的 rtabmap_ros/rgbd_sync，关于这个 nodelet 的使用等在 3D 建图程序解析中已经做过简要描述，rtabmap 节点同样也做过介绍了，此处不再做重复描述。

rtabmapviz 是 rtabmap_ros 功能包中的一个节点，该节点启动 RTAB-Map 的可视化界面，它是 RTAB-Map GUI 库的一个包装，具有与 rviz 相同的目的，但它还具有 RTAB-Map 的特定选项。在我们的 ROS 源码中默认设置不开启，如果想要使用 rtabmapviz 工具，可在 launch 文件开头处进行修改：

```
<!-- Arguments -->
  <arg name="model" default="waffle" doc="model type [burger, waffle,
waffle_pi]"/>
  <arg name="open_rviz" default="false"/>
  <arg name="rtabmapviz" default="false"/>
  <arg name="move_forward_only" default="false"/>
```

将参数 rtabmapviz 的默认值改为 true，则运行 3D 导航时 rtabmapviz 工具会自动启动。

```
<!-- visualization with rtabmapviz -->
    <node if="$(arg rtabmapviz)" pkg="rtabmap_ros" type="rtabmapviz"
name="rtabmapviz" args="-d $(find rtabmap_ros)/launch/config/rgbd_gui.ini"
output="screen">
      <param name="subscribe_scan"   type="bool"   value="true"/>
      <param name="subscribe_odom"   type="bool"   value="true"/>
      <param name="frame_id"         type="string" value="base_footprint"/>
      <param name="approx_sync"      type="bool"   value="true"/>

      <remap from="odom"            to="/odom"/>
      <remap from="scan"            to="/scan"/>
    </node>
```

参数-d 的意义是设置 GUI 界面参数的 RTAB-Map ini 文件，默认路径是 / .ros / rtabmapGUI.ini，这里我们设置文件路径为/rtabmap_ros /launch /config /rgbd_gui.ini，其他参数的设置可以参考 rtabmap_ros 的 ROS wiki 页面，这里不做描述。

多点导航节点与 2D 导航一样是调用了 turn_on_wheeltec_robot 路径下的 send_mark.py，这里不做描述。