建图导航基本原理

在 ROS 机器人中，进行构建地图需要使用到的 1 个 ROS 功能包：

- gmapping：根据激光数据（或者深度数据模拟的激光数据）构建地图。

gmapping 是一个基于 2D 激光雷达使用 RBPF（Rao-Blackwellized Particle Filters）算法完成二维栅格地图构建的 SLAM 算法。

- 优点：gmapping 可以实时构建室内环境地图，在小场景中计算量少，且地图精度较高，对激光雷达扫描频率要求较低。
- 缺点：随着环境的增大，构建地图所需的内存和计算量就会变得巨大，所以 gmapping 不适合大场景构图。一个直观的感受是，对于 200x200 米的范围，如果栅格分辨率是 5cm，每个栅格占用一个字节内存，那么每个粒子携带的地图都要 16M 的内存，如果是 100 粒子就是 1.6G 内存。

gmapping 需要订阅机器人坐标变换话题/tf、里程计/odom 和激光雷达扫描数据话题/scan，将会发布栅格地图话题/map。

在 ROS 机器人中，进行自主导航需要使用到的 2 个 ROS 功能包：

- move_base：根据参照的消息进行路径规划，使移动机器人到达指定的位置；
- amcl：根据已经有的地图进行定位。

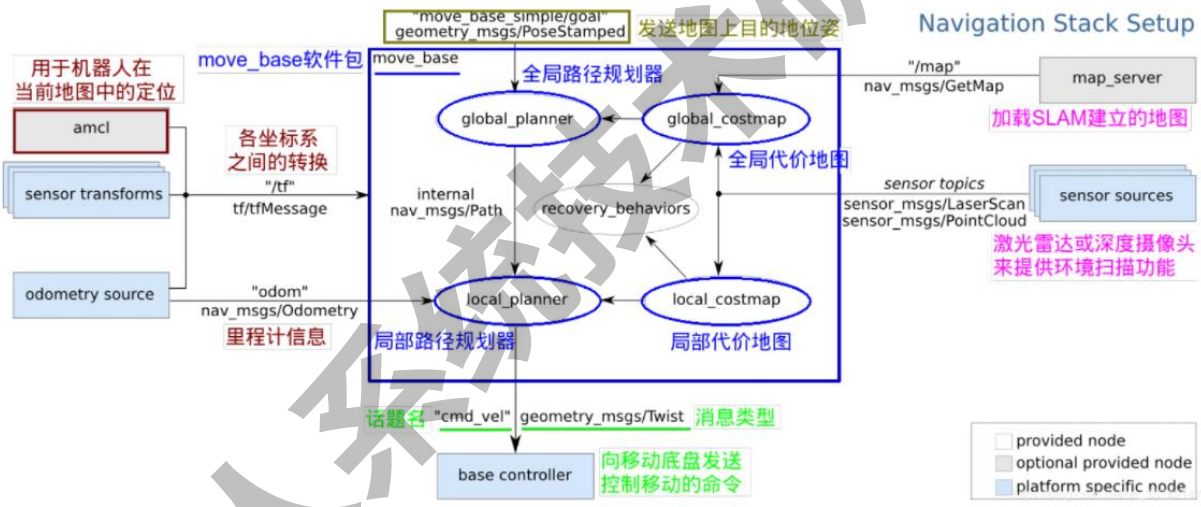

图 1：move_base/amcl 导航定位框架

在导航框架图中可以看到，move_base 提供了 ROS 导航的配置、运行、交互接口，它主要包括两个部分：

- 全局路径规划（global planner）：根据给定的目标位置进行总体路径的规划。在 ROS 的导航中，首先会通过全局路径规划，计算出机器人到目标位置的全局路线。这一功能 navfn 这个包实现的。参考说明：http://wiki.ros.org/global_planner。navfn 通过 Dijkstra 或 A*最优路径的算法，计算 costmap 上的最小花费路径，作为机器人的全局路线。
- 本地实时规划（local planner）：根据附近的障碍物进行躲避路线规划。本地的实时规划是利用 base_local_planner 包实现的。该包使用 Trajectory Rollout 和 Dynamic Window Approaches 算法计算机器人每个周期内应该行驶的速度和角度（dx, dy, dtheta velocities）。参考说明：http://wiki.ros.org/dwa_local_planner。base_local_planner 这个包通过地图数据，通过算法搜索到达目标的多条路经，利用一些评价标准（是否会撞击障碍物，所需要的时间等等）选取最优的路径，并且计算所需要的实时速度和角度。

其中，Trajectory Rollout 和 Dynamic Window Approaches 算法的主要思路如下：

采样机器人当前的状态（dx, dy, dtheta）；针对每个采样的速度，计算机器人以该速度行驶一段时间后的状态，得出一条行驶的路线；利用一些评价标准为多条路线打分；根据打分，选择最优路径；重复上面过程。

用 TianbotMini 机器人构建地图

Gmapping 建图算法+Trajectory 路径规划：

```
tianbot@ros2go:~$ roslaunch tianbot_mini slam.launch
```

在 ROS2GO 随身系统上依次运行以下 ROS 指令：

- 打开终端输入以下命令：运行迷你机器人驱动
  roslaunch tianbot_mini bringup.launch
- 打开新终端输入以下命令：运行雷达驱动
  roslaunch tianbot_mini lidar.launch
- 打开新终端输入以下命令：运行 SLAM
  roslaunch tianbot_mini slam.launch

指令运行结束后，出现下图所示的 RVIZ 可视化窗口，即可使用 Tianbot Mini 进行 SLAM 任务。

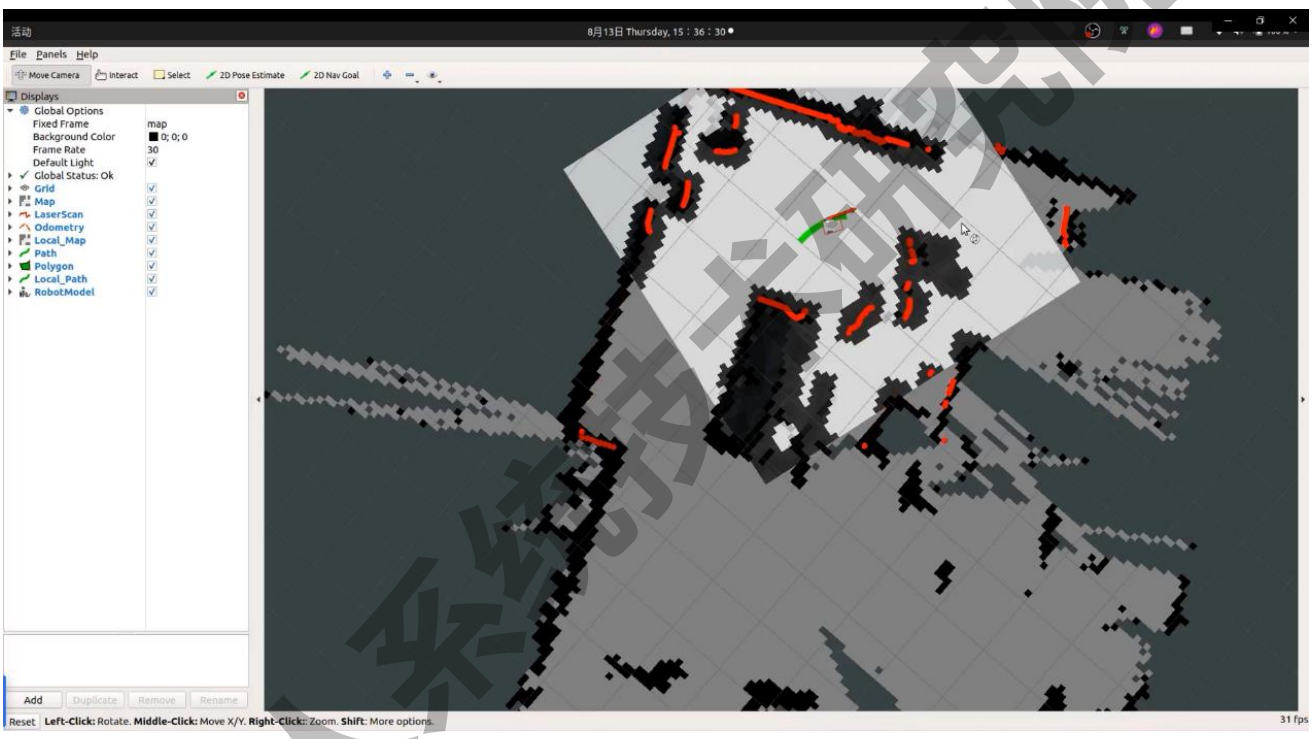

图 2：通过 gmapping 进行 SLAM 建图

建图过程也可以通过键盘遥控完成。也可以使用 RVIZ 可视化工具中的“2D Nav Goal”工具，在地图中指定目标地点，然后 TianbotMini 机器人会进行自主导航到目标地点的同时，通过 gmapping 算法构建未知区域的 2D 栅格地图。

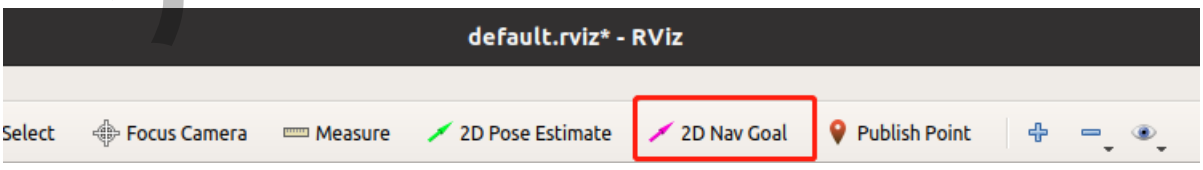

图 3：通过 2D Nav Goal 指定目标点

手动指定目标地点：

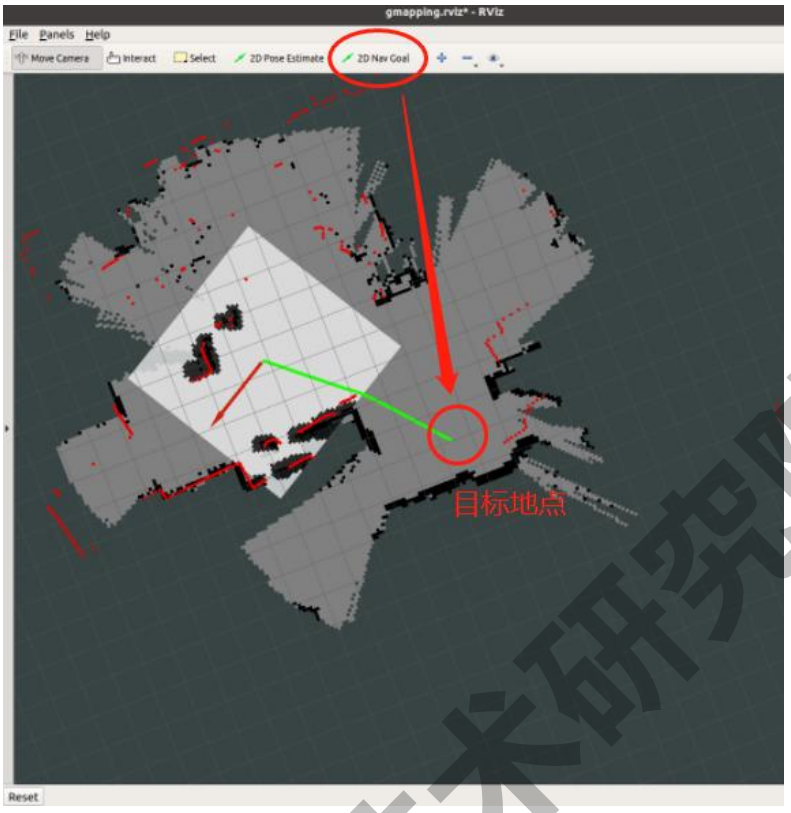

图 4：指定目标点

TianbotMini 机器人开始自主导航，并构建未知区域地图：

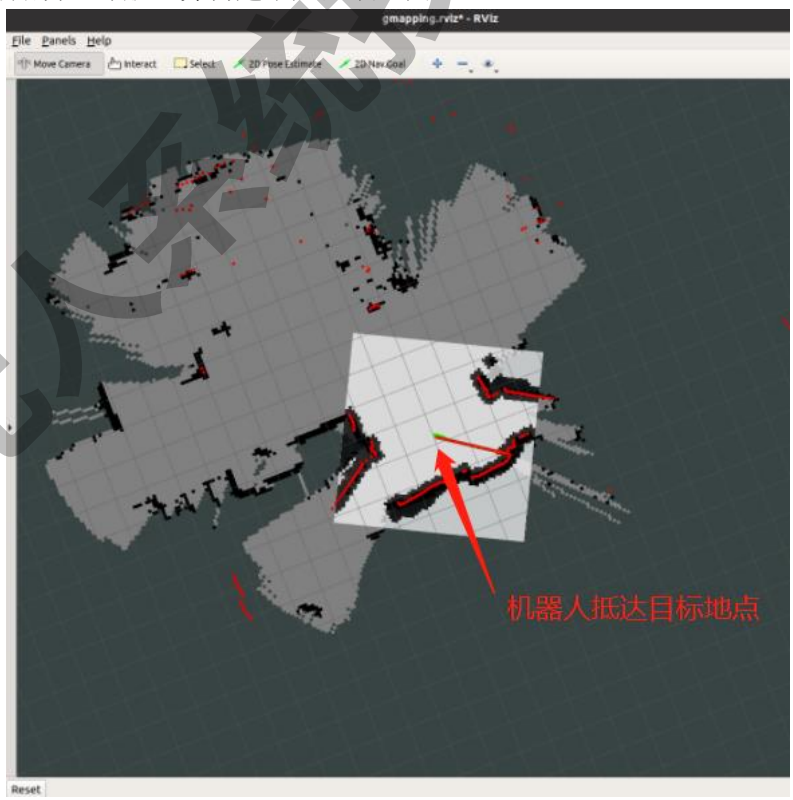

图 5：TianbotMini 行驶至目标点并建图

TianbotMini 机器人抵达目标地点，此时可继续使用 2D Nav Goal 工具指定新的目标地点。

在建图过程中记录目标点位置

打开终端，订阅 movebase_simple_goal:

rostopic echo /movebase_simple_goal

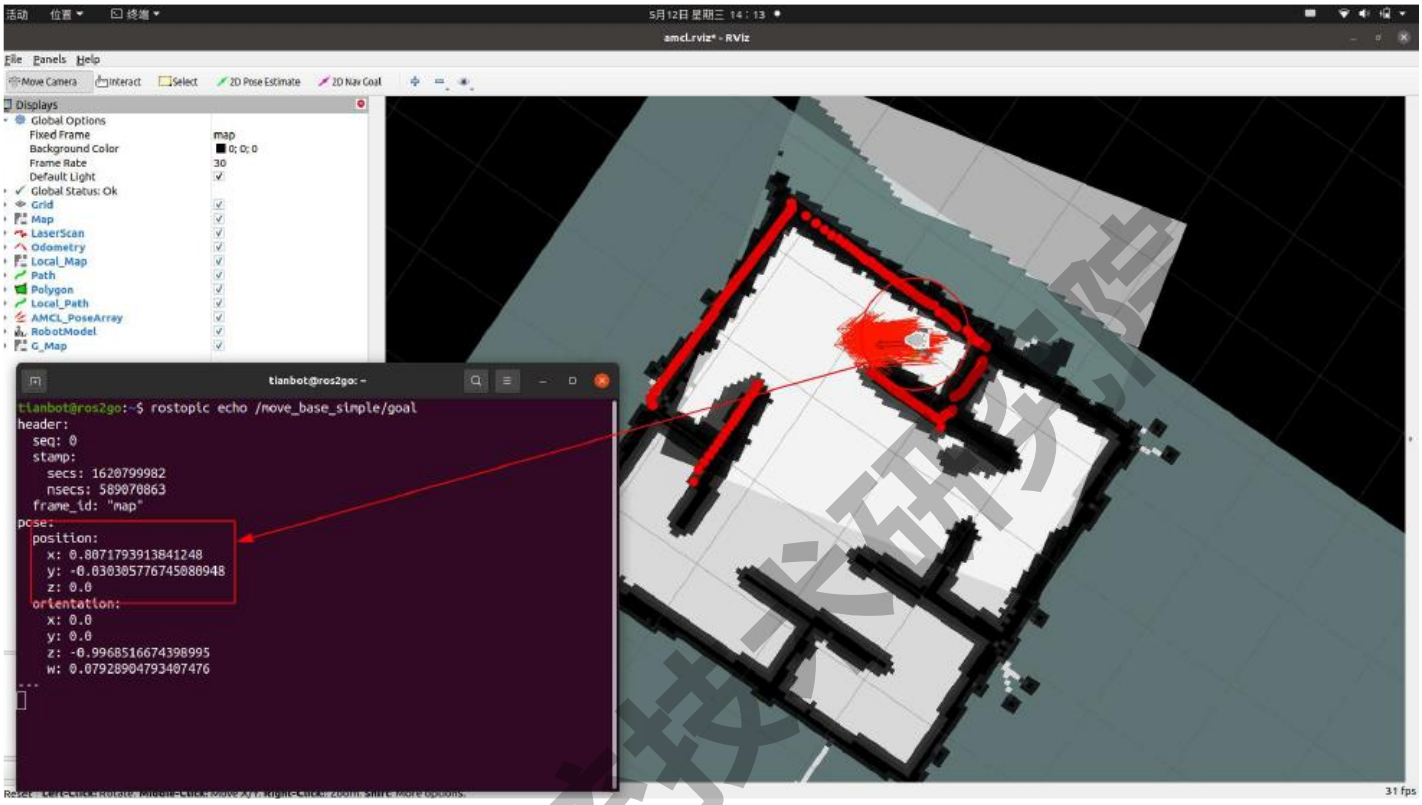

图 6：目标点位置

在 RVIZ 中点击目标位置，并记录 move_base_simple_goal 话题中的位置信息，该信息将用于之后的导航。

保存地图（map_save.launch）：

```
tianbot@ros2go:~$ roslaunch tianbot_mini map_save.launch
```

地图会被保存到本地如下路径（自动命名为 map.pgm | map.yaml）：

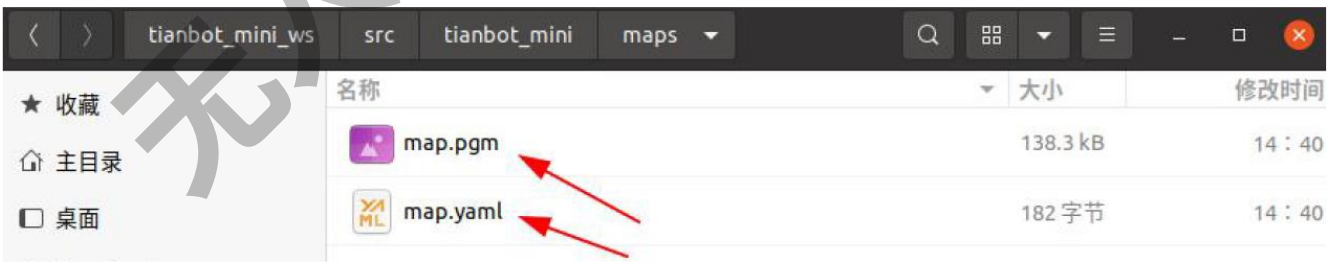

图 7：地图文件保存位置

在主目录下打开地图：在主目录找到该文件，双击打开，即可查看建好的地图。

参数调整（rqt、yaml）

启动动态调参工具：

```
tianbot@ros2go:~$ rosrun rqt_reconfigure rqt_reconfigure
```

启动后，可调整 movebase 中的具体参数：

膨胀层

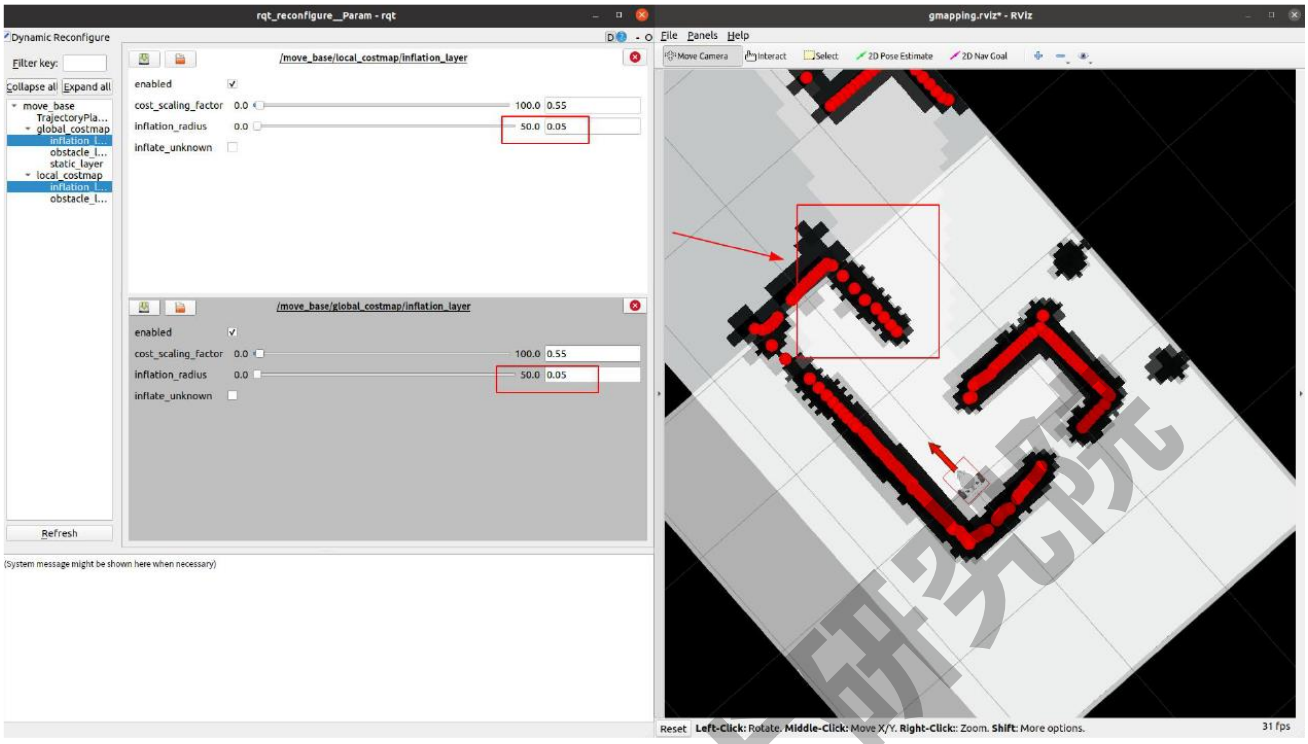

图 8：调节膨胀系数

规划器

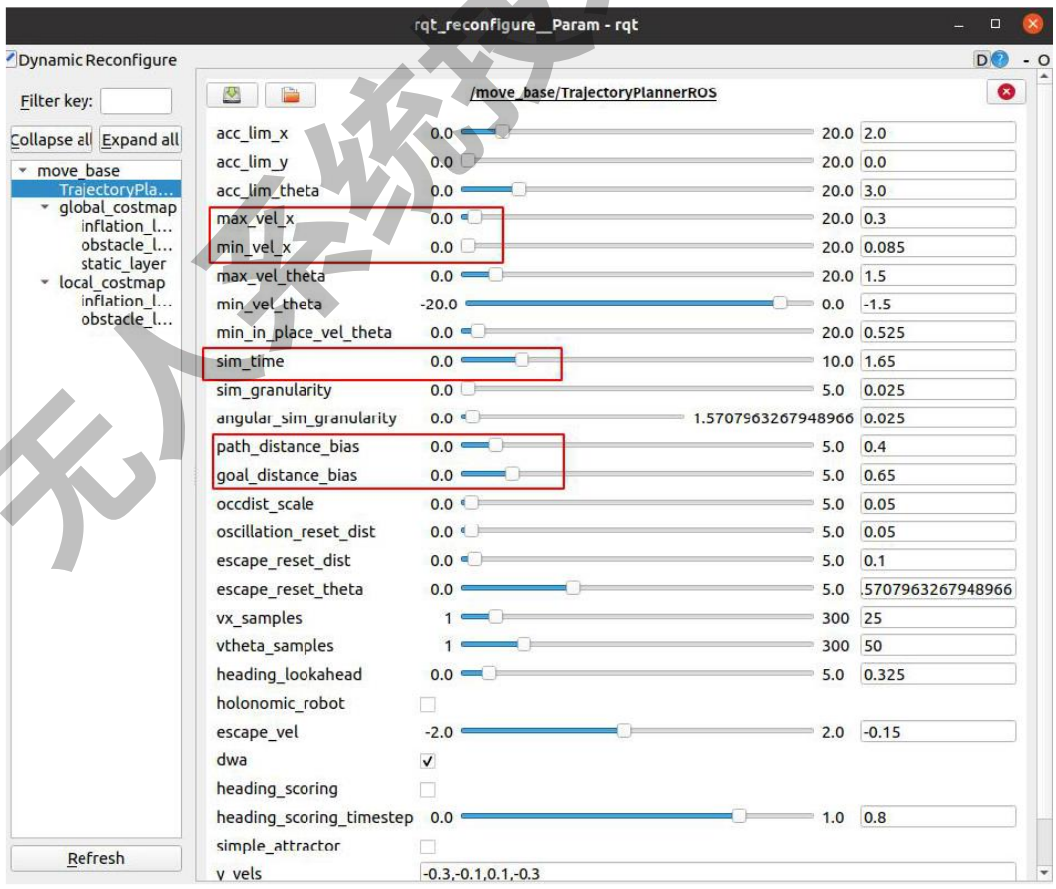

图 9：调节规划器参数

对应的本地参数文件（yaml）：

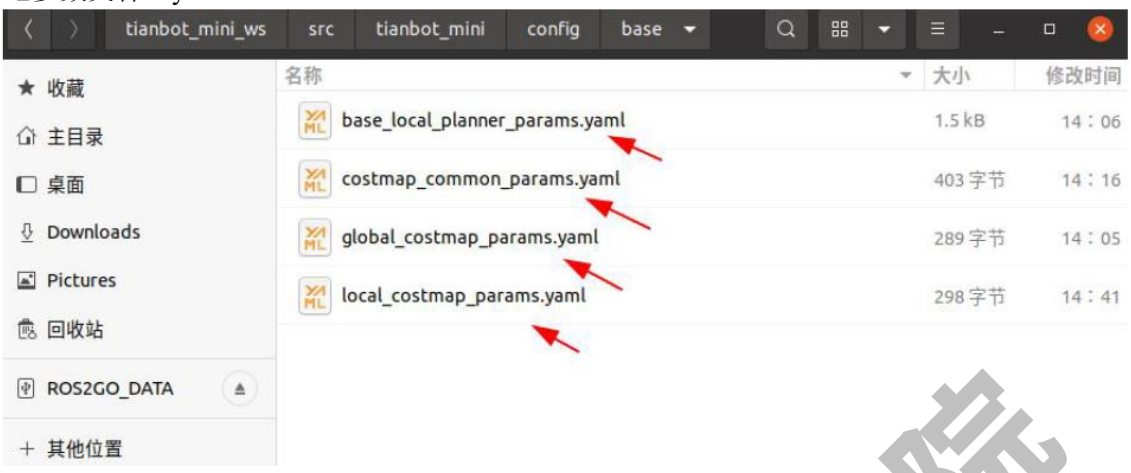

图 10：本地参数文件

每个配置文件中，标有注释作为参考：

```yaml
controller_frequency: 3.0 # movebase以 3Hz 的频率发布 运动控制命令(cmd_vel)
planner_frequency: 0.1 # movebase 以 0.1Hz 规划一次新路径

recovery_behavior_enabled: true # 启用 自救行为
clearing_rotation_allowed: true # 启用 自动清理障碍标记

TrajectoryPlannerROS:
  max_vel_x: 0.3 # 最大x轴线速度 运动速度 0.3m/s
  min_vel_x: 0.085 # 最小x轴线速度 运动速度 0.085m/s
  max_vel_y: 0.0  # zero for a differential drive robot
  min_vel_y: 0.0
  max_vel_theta: 1.5
  min_vel_theta: -1.5
  min_in_place_vel_theta: 0.525
  escape_vel: -0.15
  acc_lim_x: 2.0
  acc_lim_y: 0.0  # zero for a differential drive robot
  acc_lim_theta: 3.0

  holonomic_robot: false
  yaw_goal_tolerance: 0.75 # about -- degrees 目标位置的角度容忍度
  xy_goal_tolerance: 0.15  # 15 cm 目标位置的位置容忍度
  latch_xy_goal_tolerance: false
  path_distance_bias: 0.40
  goal_distance_bias: 0.65
  meter_scoring: true

  heading_lookahead: 0.325
  heading_scoring: false
  heading_scoring_timestep: 0.8
  occdist_scale: 0.05
  oscillation_reset_dist: 0.05
  publish_cost_grid_pc: false
  prune_plan: true

  sim_time: 1.65 # 预仿真时间，计算多长时间内的情况，影响连贯性
  sim_granularity: 0.025
  angular_sim_granularity: 0.025
  vx_samples: 25 # 线速度 采样点数量
  vy_samples: 0  # zero for a differential drive robot
  vtheta_samples: 50 # 角速度 采样点数量
  dwa: true
  simple_attractor: false
```

在保存的地图上定位导航

定位导航（amcl.launch）：

```
tianbot@ros2go:~$ roslaunch tianbot_mini amcl.launch
```

指定机器人在地图上的初始位置和姿态（最好将机器人放到它建图时的起点位置，再启动 amcl.launch）：

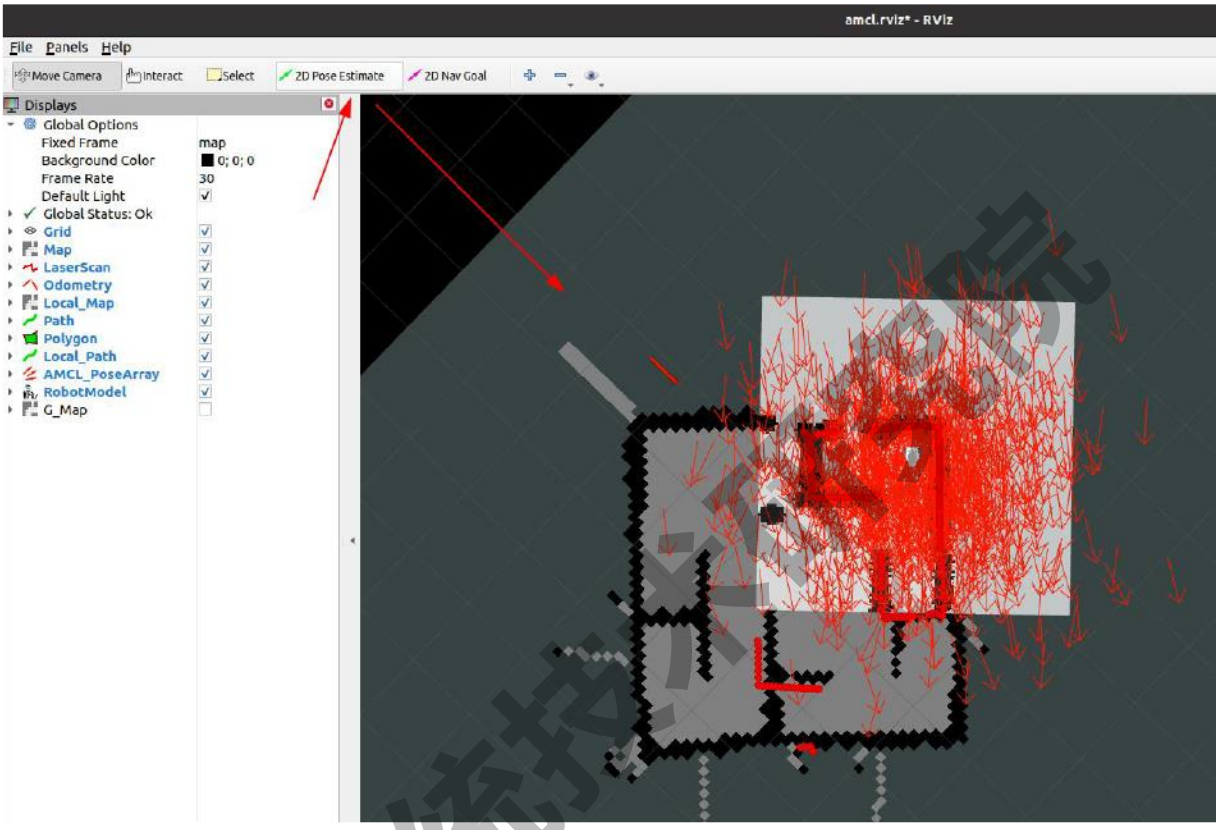

图 11：用 2D Pose Estimate 指定机器人初始位姿

手动发布之前保存的导航点：

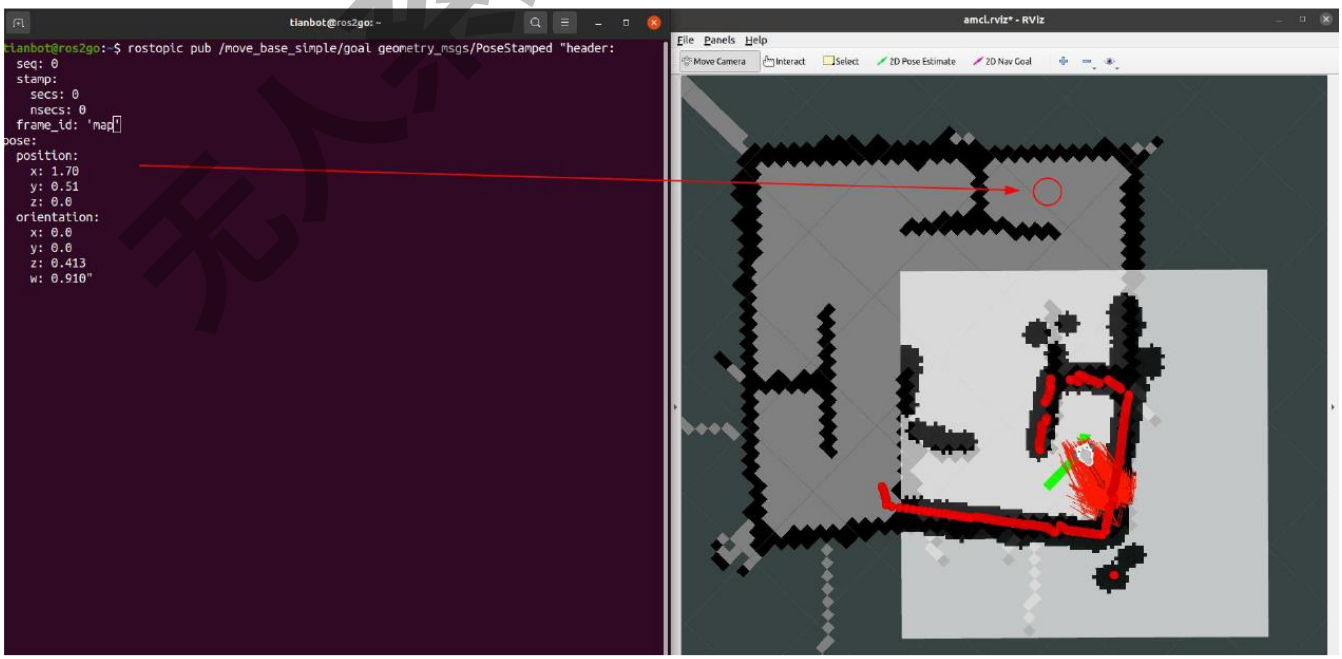

图 12：发布导航点

发布导航点命令如下（用 tab 进行补全，然后填写坐标和 frame_id）：

```
rostopic pub /move_base_simple/goal geometry_msgs/PoseStamped "header:
seq: 0
stamp:
  secs: 0
  nsecs: 0
frame_id: 'map'
pose:
  position:
    x: 1.70
    y: 0.51
    z: 0.0
orientation:
  x: 0.0
  y: 0.0
  z: 0.413
  w: 0.910"
```

以上几个下划线部分需要修改和填写，否则机器人将无法执行相应导航命令。

切换规划、导航算法（teb）

Gmapping 建图算法+Teb 路径规划：

```
tianbot@ros2go:~$ roslaunch tianbot_mini slam_teb.launch
```

Amcl 定位算法+Teb 路径规划：

```
tianbot@ros2go:~$ roslaunch tianbot_mini amcl_teb.launch
```