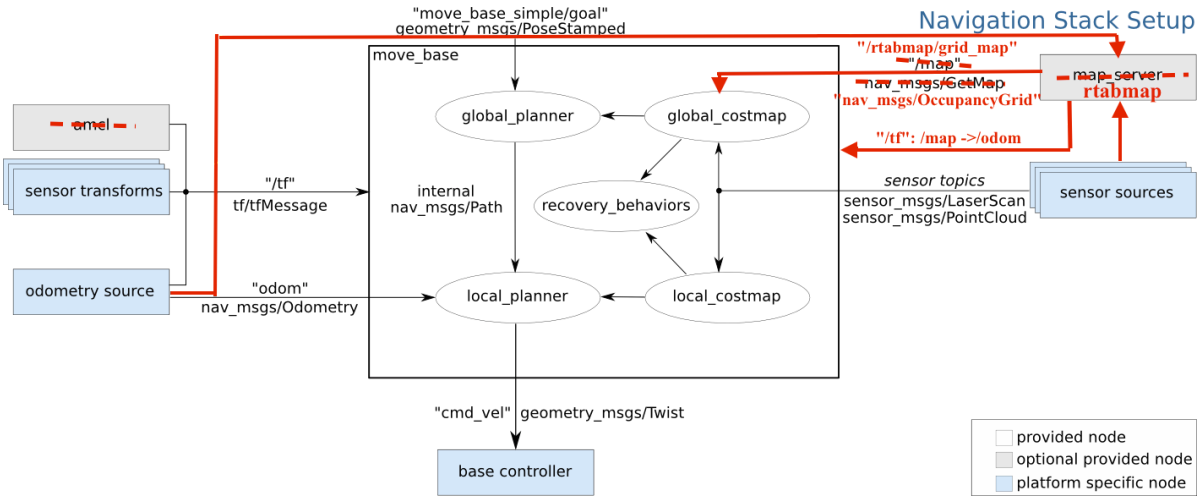
7 RTAB-Map 3D mapping navigation

wiki: http://wiki.ros.org/rtabmap_ros

7.1 Introduction

This package is a ROS functional package for RTAB Map, an RGB-D SLAM method based on a global loop closure detector with real-time constraints. This package can be used to generate 3D point clouds of the environment and create 2D occupancy grid maps for navigation.



As can be seen from the above figure, there is no need for Monte Carlo positioning amcl, and RTAB Map has its own positioning function; if it is used, it will cause repeated positioning and positioning failure. When using RTAB Map to navigate the core framework, the initialized map is provided by RTAB Map, not map_server.

7.2 Mapping use

Note: When building a map, the slower the speed, the better the effect(note that if the rotation speed is slower), the effect will be poor if the speed is too fast.

According to different models, you only need to set the purchased model in [.bashrc], X1(ordinary four-wheel drive) X3(Mike wheel) X3plus(Mike wheel mechanical arm) R2(Ackerman differential) and so on. Section takes X3 as an example

```
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to Jetson Orin-Docker/05, Enter the robot's docker container
~/run_docker.sh
#Multiple ros commands require multiple terminals to enter the same docker container for execution, please refer to Jetson Orin-Docker/05, Section 5.8 tutorial
```

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameter and modify the corresponding model

```
export  ROBOT_TYPE=X3    # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

7.2.1 Start

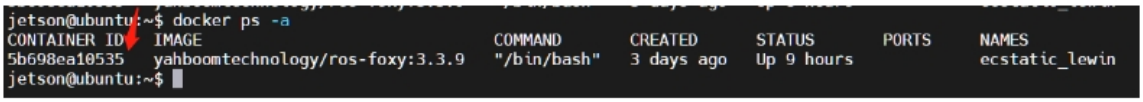
Start the underlying driver command(robot side)

```
roslaunch  yahboomcar_nav  laser_astrapro_bringup.launch
```

Command to start mapping or navigation(robot side)

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```



2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```

After successfully entering the container, you can open countless terminals to enter the container.

```
roslaunch  yahboomcar_nav  yahboomcar_rtabmap.launch  use_rviz:=False
```

- use_rviz parameter: whether to open rviz.

Start Visualization(Virtual Machine)

```
roslaunch  yahboomcar_nav  view_rtabmap.launch
```

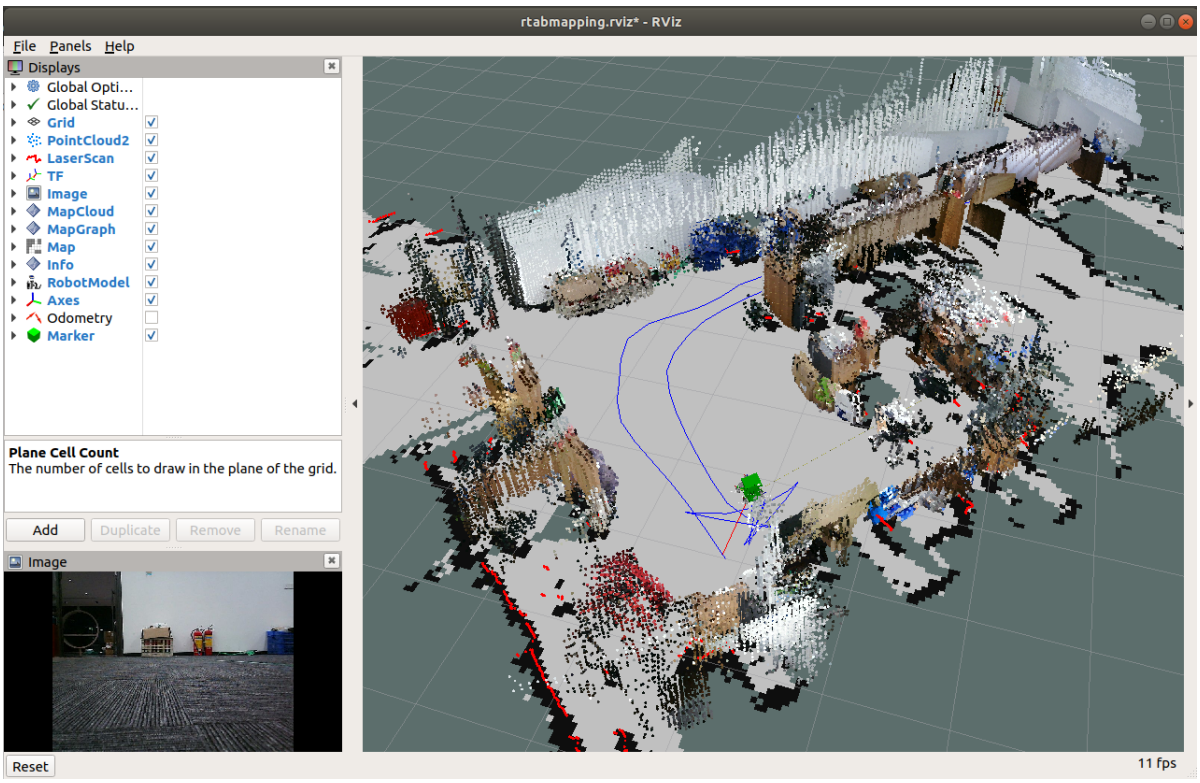
Keyboard control node(virtual machine)

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py  # system integration
roslaunch yahboomcar_ctrl yahboom_keyboard.launch      # yahboomcar custom
```

7.2.2 Mapping

After starting up according to the above method, choose any method to control the map(handle control is recommended); the slower the speed when building the map, the better the effect(especially the angular speed); the robot fills the area to be built, and the map is closed as much as possible.

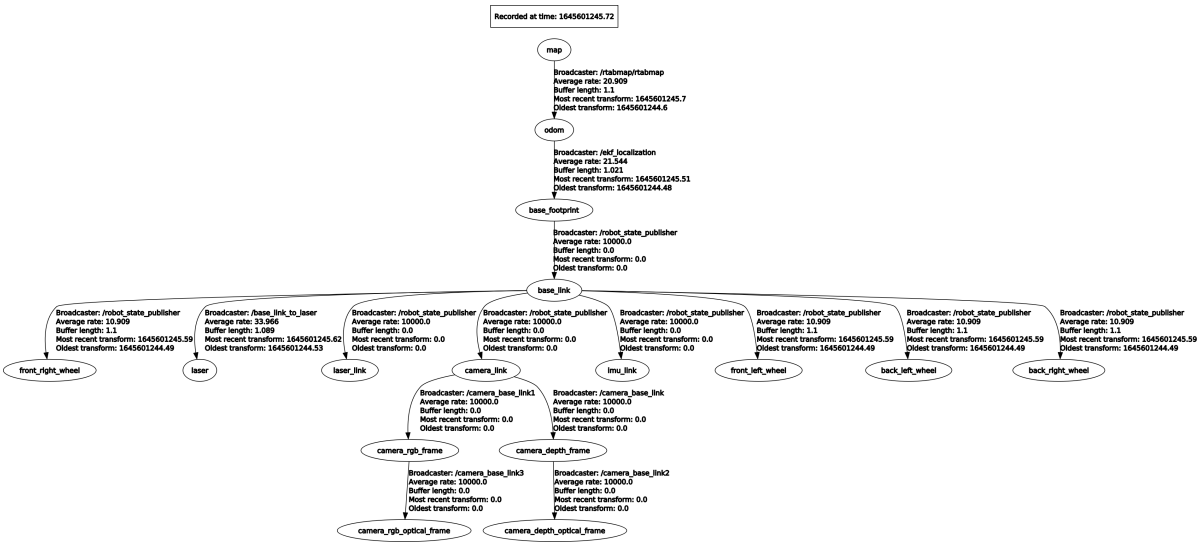
- handle control
- keyboard control



When the map is completed, directly [ctrl+c] to exit the map node, the system will automatically save the map. The default map save path [~/.ros/rtabmap.db].

View tf tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



Node view

```
rqt_graph
```

As can be seen from the above figure, the information that the [rtabmap] node needs to subscribe: radar data, camera data, tf data.

7.3 Navigation and obstacle avoidance

Note: [R2] of the remote control handle has the function of canceling the target point.

Start the underlying driver command(robot side)

```
roslaunch yahboomcar_nav laser_astrapro_bringup.launch
```

Command to start mapping or navigation(robot side)

```
roslaunch yahboomcar_nav yahboomcar_rtabmap_nav.launch use_rviz:=False
```

- use_rviz parameter: whether to open rviz.

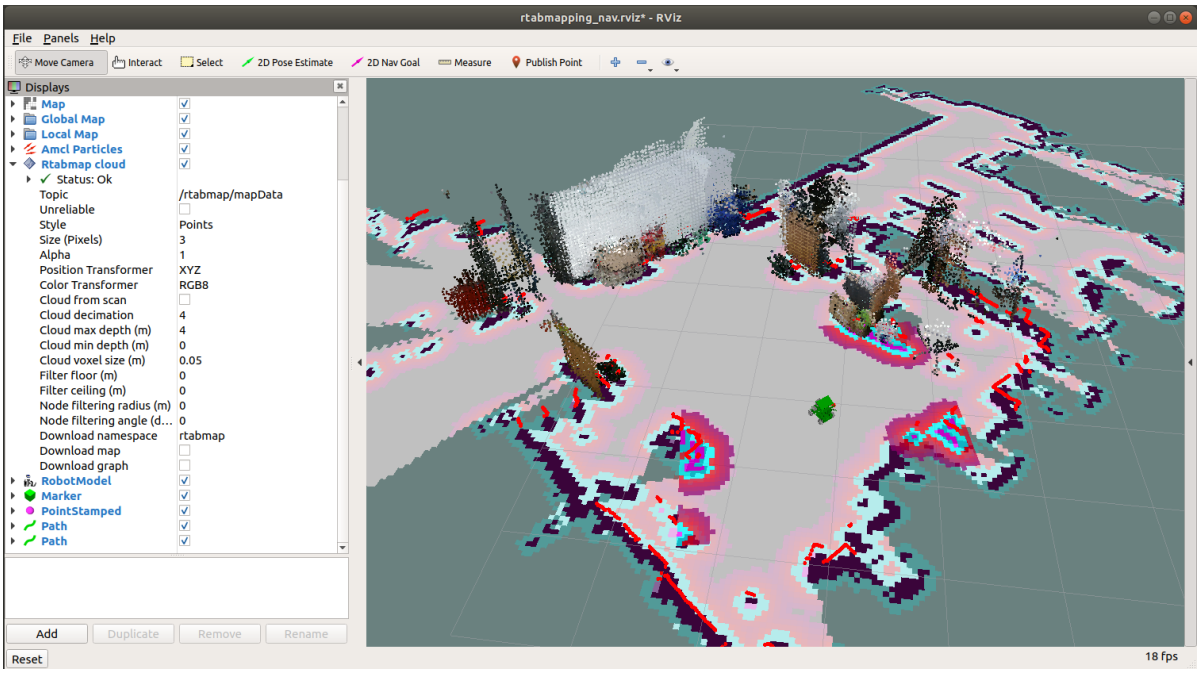
Start Visualization(Virtual Machine)

```
roslaunch yahboomcar_nav view_rtabmap_nav.launch
```

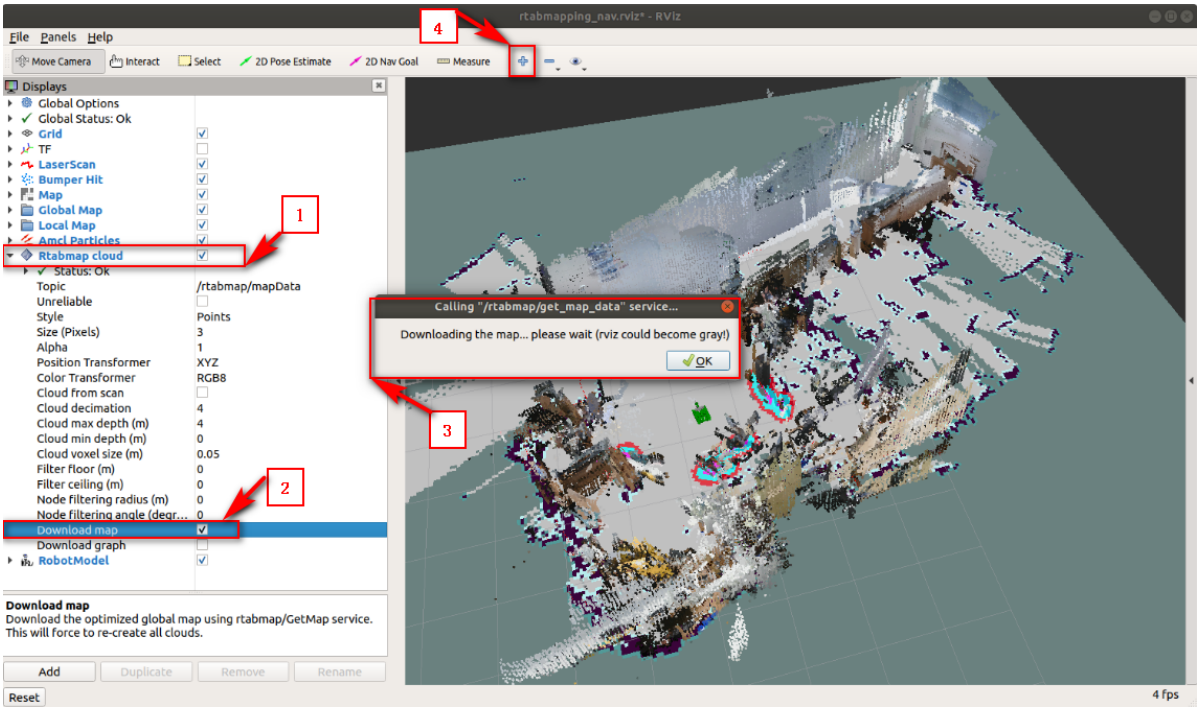
Keyboard control node(virtual machine)

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py  # system integration
roslaunch yahboomcar_ctrl yahboom_keyboard.launch      # yahboomcar custom
```

When the navigation mode is turned on, the system automatically loads the 2D grid map, and cannot directly load the 3D map, it needs to be loaded manually.



Load the 3D map(1, 2, 3), 4 is to add the rviz debugging tool.



At this point, you can manually add [MarkerArray] to facilitate multi-point navigation and observation, and adjust [rviz] display parameters as needed, such as the size of lidar points.

7.3.1 Single point navigation

- Use the [2D Pose Estimate] of the [rviz] tool to set the initial pose until the position of the car in the simulation is consistent with the position of the actual car.
- Click the [2D Nav Goal] of the [rviz] tool, and then select the target point on the map where there are no obstacles, release the mouse to start the navigation, only one target point can be selected, and it will stop when it arrives.

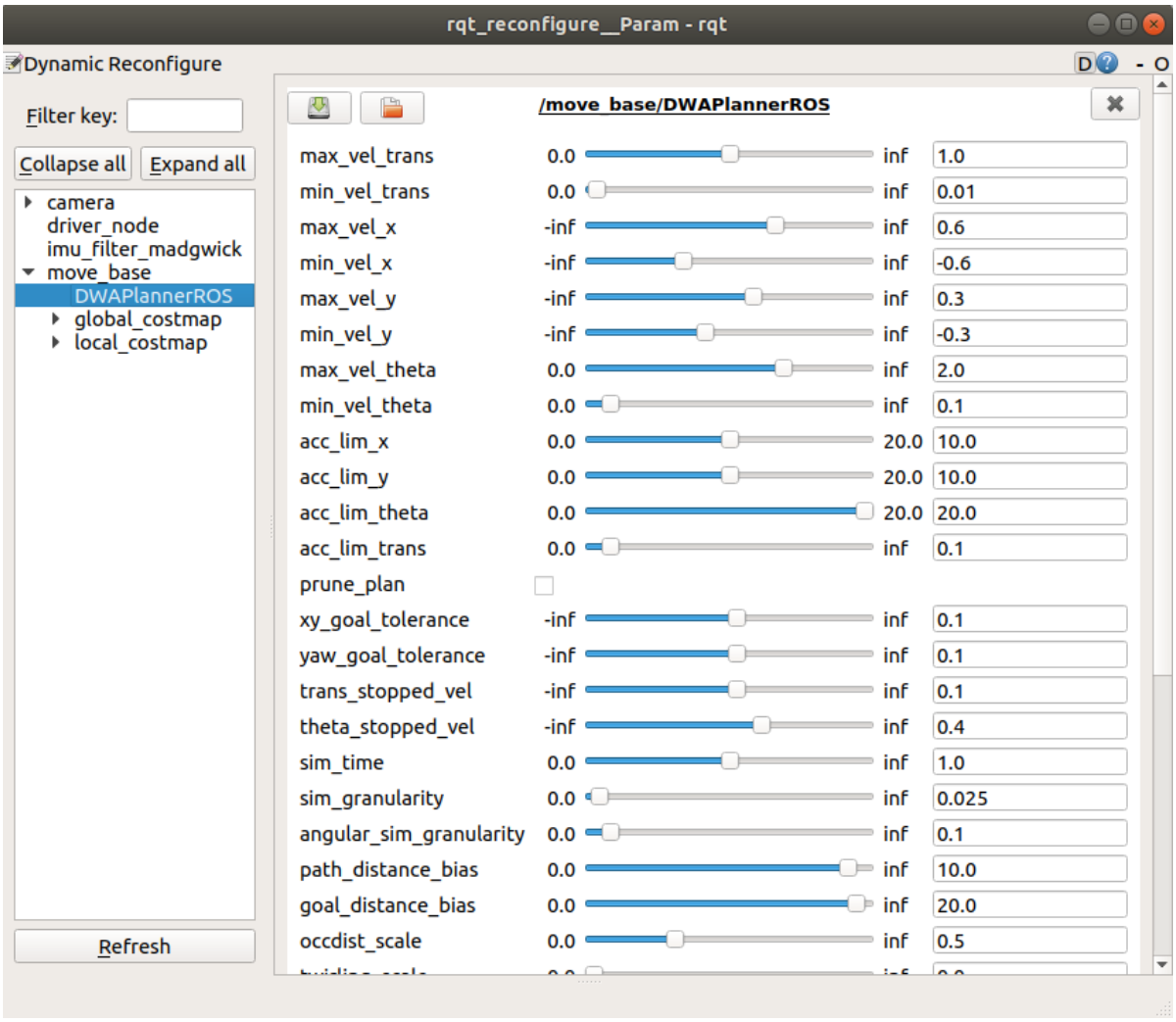
7.3.2 Multi-point navigation

- Same as the first step of single-point navigation, first set the initial pose of the car.
- Click the [Publish Point] of the [rviz] tool, and then select the target point on the map where there are no obstacles, release the mouse to start the navigation, you can click the [Publish Point] again, and then select the point, the robot will point and point cruising in between.
- When using the [2D Pose Estimate] tool of the [rviz] tool to set the initial pose of the car, the multi-point navigation function is automatically canceled.

7.3.3 Parameter configuration

After starting the navigation function, open the dynamic parameter adjustment tool, adjust it according to your own needs, and observe the motion state of the robot until the effect is optimal, record the current parameters, and modify them to the corresponding dwa_local_planner_params.yaml file under the yahboomcar_nav function package.

```
rosrun rqt_reconfigure rqt_reconfigure
```



Looking at the yahboomcar_navigation.launch file, you can see that the navigation parameters are modified in the move_base.launch file under the yahboomcar_rtabmap_nav function package.

```
< launch >
    <!-- Whether to open rviz || Whether to open rviz -->
    < arg  name = "use_rviz"  default = "false" />
    <!-- MarkerArray node-->
    < node  name = 'send_mark'  pkg = "yahboomcar_nav"  type = "send_mark.py" />
    <!-- Mobile APP Node-->
    < include  file = "$(find yahboomcar_nav)/launch/library/app.launch" />
    <!-- Navigation core component move_base -->
    < include  file = "$(find yahboomcar_nav)/launch/library/move_base.launch" />
    <!-- rtabmap navigation -->
    < include  file = "$(find yahboomcar_nav)/launch/library/rtabmap_nav.launch"
/>
    <!-- RVIZ -->
    < include  file = "$(find
yahboomcar_nav)/launch/view/view_rtabmap_nav.launch"  if = "$(arg use_rviz)" />
</ launch >
```

Find the move_base.launch file, open the sample file as follows, you can modify and replace it according to your needs; at this time, the [DWA planner] is selected, and the [DWA] file is loaded.

```
< launch >
    < arg  name = "robot_type"  value = "$(env ROBOT_TYPE)"  doc = "robot_type
[X1,X3,X3plus,R2,X7]" />
```

```
    <!-- Arguments -->
    < arg  name = "move_forward_only"  default = "false" />
    <!-- move_base -->
    < node  pkg = "move_base"  type = "move_base"  respawn = "false"  name =
"move_base"  output = "screen" >
        < rosparam  file = "$(find
yahboomcar_nav)/param/common/global_costmap_params.yaml"  command = "load" />
        < rosparam  file = "$(find
yahboomcar_nav)/param/common/local_costmap_params.yaml"  command = "load" />
        < rosparam  file = "$(find
yahboomcar_nav)/param/common/move_base_params.yaml"  command = "load" />
        < rosparam  file = "$(find
yahboomcar_nav)/param/common/costmap_common_params_$(arg robot_type).yaml"
command = "load"
            ns = "global_costmap" />
        < rosparam  file = "$(find
yahboomcar_nav)/param/common/costmap_common_params_$(arg robot_type).yaml"
command = "load"
            ns = "local_costmap" />
        < rosparam  file = "$(find
yahboomcar_nav)/param/common/dwa_local_planner_params_$(arg robot_type).yaml"
command = "load" />
        < param  name = "base_local_planner"  type = "string"  value =
"dwa_local_planner/DWAPlannerROS"  if = "$(eval arg('robot_type') == 'X3')" />
        <!-- <param name="base_local_planner" type="string"
value="teb_local_planner/TebLocalPlannerROS"/>-->
        < param  name = "DWAPlannerROS/min_vel_x"  value = "0.0"  if = "$(arg
move_forward_only)" />
        < remap  from = "cmd_vel"  to = "cmd_vel" />
        < remap  from = "odom"  to = "odom" />
    </ node >
</ launch >
```

Note: When using the DWA planner, the difference between the omnidirectional car and the differential car is whether the speed in the Y direction is 0. There are clear comments in it, which can be modified according to the actual situation.

Enter the dwa_local_planner_params.yaml file under the yahboomcar_nav function package, some parameters are as follows:

```
DWAPlannerROS :
  # Robot Configuration Parameters
  # The absolute value of the maximum linear velocity in the x direction, unit:
m/s
  # The maximum y velocity for the robot in m/s
  max_vel_x :  0.6
  # The absolute value of the minimum linear velocity in the x direction, a
negative number means that it can be moved back, unit: m/s
  # The minimum x velocity for the robot in m/s, negative for backwards motion.
  min_vel_x : - 0.6
  # The absolute value of the maximum linear velocity in the y direction, unit:
m/s. Differential robot is 0
  # The maximum y velocity for the robot in m/s
  max_vel_y : 0.3
```

```
  # The absolute value of the minimum linear velocity in the y direction, unit:
m/s. Differential robot is 0
  # The minimum y velocity for the robot in m/s
  min_vel_y : - 0.3
 …
  # The limit acceleration of the robot in the x direction, the unit is
meters/sec^2
  # The x acceleration limit of the robot in meters/sec^2
  acc_lim_x :  10.0
  # The limit acceleration of the robot in the y direction, it is 0 for the
differential robot
  # The y acceleration limit of the robot in meters/sec^2
  acc_lim_y : 10.0
... ..
```

Other parameter files can be opened, combined with annotations and courseware, and modified according to their own needs.

7.4 node rtabmap

This is the master node for this package. It is a wrapper around the RTAB mapping core library. Here, the map is incrementally built and optimized when loop closures are detected. The online output of the node is the local map, which contains the most recent data added to the map. The default location of the RTAB map database is [.ros/rtabmap.db], and the workspace is also set to [.ros].

7.4.1 Subscription topic

name	type	Parse
odom	nav_msgs/Odometry	Odometer. Required if parameter subscribe_depth or subscribe_stereo is true; and odom_frame_id is not set.
rgb/image	sensor_msgs/Image	RGB/monocular image.
rgb/camera_info	sensor_msgs/CameraInfo	RGB camera parameters.
depth/image	sensor_msgs/Image	depth image.
scan	sensor_msgs/LaserScan	Single line laser.
scan_cloud	sensor_msgs/PointCloud2	Laser scan point cloud stream.
left/image_rect	sensor_msgs/Image	Left eye corrected image.
left/camera_info	sensor_msgs/CameraInfo	Left eye camera parameters.
right/image_rect	sensor_msgs/Image	Right eye corrected image.
right/camera_info	sensor_msgs/CameraInfo	Right eye camera parameters.
goal	geometry_msgs/PoseStamped	Plan a path to this goal using the current online map.

name	type	Parse
rgbd_image	rtabmap_ros/RGBDImage	RGB-D sync image, only if subscribe_rgbd is true.

7.4.2 Post topic

name	type	Parse
info	rtabmap_ros/Info	rtabmap information.
mapData	rtabmap_ros/MapData	Graph and latest node data for rtabmap.
mapGraph	rtabmap_ros/MapGraph	rtabmap graphics
grid_map	nav_msgs/OccupancyGrid	Maps generated by laser scanning occupy the grid.
proj_map	nav_msgs/OccupancyGrid	Deprecated, use /grid_map instead of Grid/FromDepth=true
cloud_map	sensor_msgs/PointCloud2	A 3D point cloud generated from a local grid.
cloud_obstacles	sensor_msgs/PointCloud2	Generate a 3D point cloud of obstacles from a local mesh.
cloud_ground	sensor_msgs/PointCloud2	A 3D ground point cloud generated from a local grid.
scan_map	sensor_msgs/PointCloud2	2D scans or 3D point clouds generated from 3D scans.
labels	visualization_msgs/MarkerArray	Convenience method for displaying graph labels in RVIZ.
global_path	nav_msgs/Path	The planned pose of the planned global path. Published only once per planned path.
local_path	nav_msgs/Path	Plan future local poses corresponding to the global path. Posted every time the map is updated.
goal_reached	std_msgs/Bool	A plan status message whether the goal was successfully achieved.
goal_out	geometry_msgs/PoseStamped	Plan the current metric target sent from rtabmap's topology planner. For example, you can connect to move_base via move_base_simple/goal.

name	type	Parse
octomap_full	octomap_msgs/Octomap	Get octomap. Available only when rtabmap_ros is built with octomap.
octomap_binary	octomap_msgs/Octomap	Get octomap. Available only when rtabmap_ros is built with octomap.
octomap_occupied_space	sensor_msgs/PointCloud2	octomap The point cloud of the occupied space(obstacles and ground). Available only when rtabmap_ros is built with octomap.
octomap_obstacles	sensor_msgs/PointCloud2	A point cloud of obstacles on the octomap. Available only when rtabmap_ros is built with octomap.
octomap_ground	sensor_msgs/PointCloud2	octomap's point cloud. Available only when rtabmap_ros is built with octomap.
octomap_empty_space	sensor_msgs/PointCloud2	Blank point cloud for octomap. Available only when rtabmap_ros is built with octomap.
octomap_grid	nav_msgs/OccupancyGrid	Project the octomap into a 2D occupancy raster map. Available only when rtabmap_ros is built with octomap.

7.4.3 Services

name	type	Parse
get_map	rtabmap_ros/GetMap	Call this service to get a standard 2D occupancy grid.
get_map_data	rtabmap_ros/GetMap	Call this service to get map data.
publish_map	rtabmap_ros/PublishMap	Call this service to publish map data.
list_labels	rtabmap_ros/ListLabels	Get the current label of the graph.
update_parameters	std_srvs/Empty	The node will be updated with the current parameters of the rosparam server.

name	type	Parse
reset	std_srvs/Empty	Delete the map.
pause	std_srvs/Empty	Pause building.
resume	std_srvs/Empty	Restoring the map.
trigger_new_map	std_srvs/Empty	A new map will start.
backup	std_srvs/Empty	Backup the database to "database_path.back"(default ~/.ros/rtabmap.db.back).
set_mode_localization	std_srvs/Empty	Set location-only mode.
set_mode_mapping	std_srvs/Empty	Set the mapping mode.
set_label	rtabmap_ros/SetLabel	Set the label to the latest node or the specified node.
set_goal	rtabmap_ros/SetGoal	Plan to set topology goals.
octomap_full	octomap_msgs/GetOctomap	Get octomap. Only available if rtabmap_ros was built with octomap
octomap_binary	octomap_msgs/GetOctomap	Get octomap. Only available if rtabmap_ros was built with octomap

7.4.4 Parameters

name	type	Defaults	Parse
subscribe_depth	bool	true	Subscribe to depth images
subscribe_scan	bool	false	Subscribe to lidar data
subscribe_scan_cloud	bool	false	Subscribe to Laser 3D Point Cloud
subscribe_stereo	bool	false	Subscribe to binocular images
subscribe_rgbd	bool	false	Subscribe to the rgbd_image topic
frame_id	string	base_link	Frame attached to mobile base.
map_frame_id	string	map	The coordinate system attached to the map.
odom_frame_id	string	' '	The coordinate system attached to the odometer.

name	type	Defaults	Parse
odom_tf_linear_variance	double	0.001	When using odom_frame_id, the first 3 values of the diagonal of the 6x6 covariance matrix are set to this value.
odom_tf_angular_variance	double	0.001	When using odom_frame_id, the last 3 values of the diagonal of the 6x6 covariance matrix are set to this value
queue_size	int	10	The size of the message queue for each synchronization topic.
publish_tf	bool	true	Publish TF from /map to /odom.
tf_delay	double	0.05	
tf_prefix	string	' '	The prefix to add to the generated tf.
wait_for_transform	bool	true	The wait for the transform when the tf transform is still unavailable(the maximum wait time for a transform is seconds).
wait_for_transform_duration	double	0.1	wait_for_transform wait time.
config_path	string	' '	Path to a configuration file containing RTAB mapping parameters. Parameters set in the startup file will override those in the configuration file.
database_path	string	.ros/rtabmap.db	Path to the rtabmap database.
gen_scan	bool	false	Generate a laser scan from a depth image(using the middle horizontal line of the depth image). Not generated if subscribe_scan or subscribe_scan_cloud is true.

name	type	Defaults	Parse
gen_scan_max_depth	double	4.0	The maximum depth of the generated laser scan.
approx_sync	bool	false	Use approximate time synchronization of incoming messages. If false, note that the odometry input must have the exact same timestamp as the input image
rgbd_cameras	int	1	The number of RGB-D cameras to use(when subscribe_rgbd is true). Currently, up to 4 cameras can be synchronized simultaneously.
use_action_for_goal	bool	false	Use actionlib to send metrics targets to move_base.
odom_sensor_sync	bool	false	For each node added to the graph, adjust the image and scan pose relative to the odometry pose.
gen_depth	bool	false	A depth image is generated from a scanned cloud projection to an RGB camera, taking into account the displacement of the RGB camera from the odometry and lidar frames.
gen_depth_decimation	int	1	Reduce the image size of the received camera information(creates a smaller depth image)
gen_depth_fill_holes_size	int	0	Fill empty pixels to that size. Interpolate values from adjacent depth values. 0 means disabled.
gen_depth_fill_iterations	double	0.1	Maximum depth error(m) to interpolate.
gen_depth_fill_holes_error	int	1	The number of iterations to fill blanks.

name	type	Defaults	Parse
map_filter_radius	double	0.0	Loads data for only one node within the filter radius(use the latest data) up to the filter angle(map filter angle).
map_filter_angle	double	30.0	The angle to use when filtering nodes before creating the map. Reference map_filter_radius
map_cleanup	bool	true	If you are not subscribed to any map cloud maps, raster maps, or project maps, clear the corresponding data.
latch	bool	true	If true, the last message published on the map topic will be saved.
map_always_update	bool	true	Always update occupancy raster map
map_empty_ray_tracing	bool	true	Perform ray tracing to fill the unknown space of invalid 2D scan rays(assuming invalid rays to be infinite). Only used if map_always_update is also true.

7.4.5 tf conversion

what is needed:

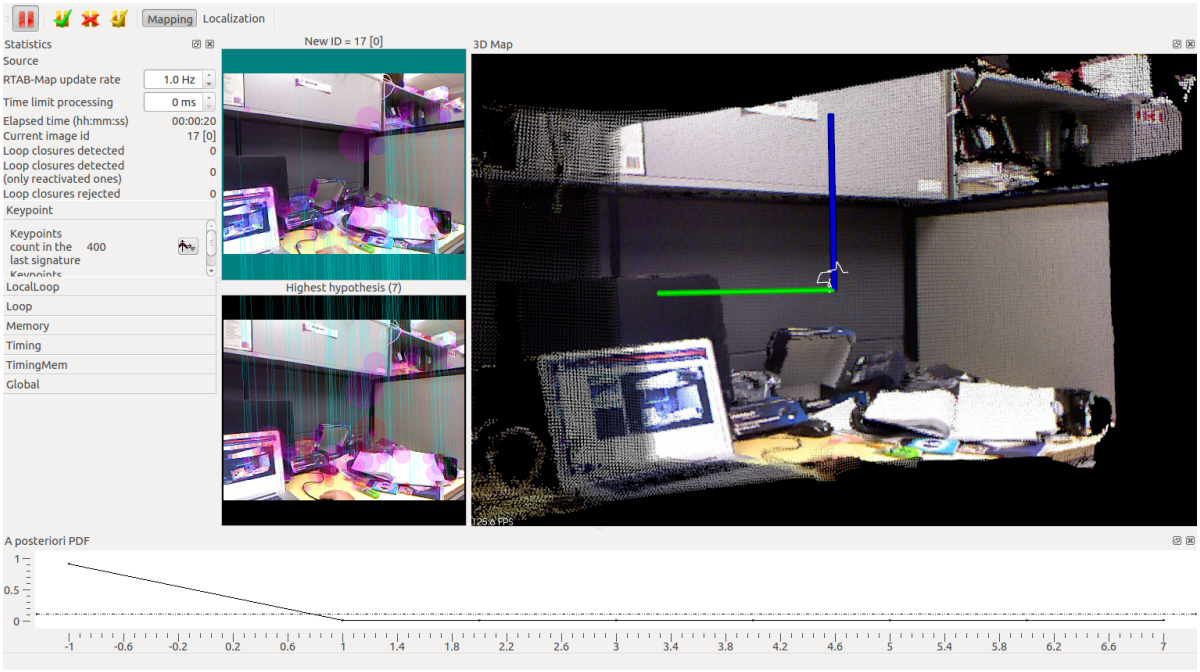
- base_link → sensor
- odom → base_link

which provided:

map → odom

7.5 node rtabmapviz

This node starts the visualization interface of RTAB-Map. It is a wrapper for RTAB-MapGUI library. It serves the same purpose as rviz, but with RTAB-Map specific options.



7.5.1. Subscription topic

name	type	Parse
odom	nav_msgs/Odometry	Odometer. Required if parameter subscribe_depth or subscribe_stereo is true and odom_frame_id is not set.
rgb/image	sensor_msgs/Image	RGB/monocular image. If the parameter subscribe_stereo is true, this option is not required(use left/image_rect instead).
rgb/camera_info	sensor_msgs/CameraInfo	RGB camera metadata. If the parameter subscribe_stereo is true, this option is not required(use left/camera_info instead).
depth/image	sensor_msgs/Image	Register the depth image. Required if parameter subscribe_depth is true.
scan	sensor_msgs/LaserScan	Laser scan stream. Required if parameter subscribe_scan is true.
scan_cloud	sensor_msgs/PointCloud2	Laser scan stream. Required if parameter subscribe_scan_cloud is true.
left/image_rect	sensor_msgs/Image	Left eye corrected image. Required if parameter subscribe_stereo is true.
left/camera_info	sensor_msgs/CameraInfo	Left eye camera parameters. Required if parameter subscribe_stereo is true.
right/image_rect	sensor_msgs/Image	Right corrected image. Required if parameter subscribe_stereo is true.

name	type	Parse
right/camera_info	sensor_msgs/CameraInfo	Right eye camera parameters. Required if parameter subscribe_stereo is true.
odom_info	rtabmap_ros/OdomInfo	Required if parameter subscribe_odom_info is true.
info	rtabmap_ros/Info	Statistics for rtabmap.
mapData	rtabmap_ros/MapData	rtabmap's graph and latest node data.
rgbd_image	rtabmap_ros/RGBDImage	RGB-D sync image, only if subscribe_rgbd is true.

7.5.2 parameter configuration

name	type	Defaults	Parse
subscribe_depth	bool	false	Subscribe to depth images
subscribe_scan	bool	false	Subscribe to lidar data
subscribe_scan_cloud	bool	false	Subscribe to Laser Scan Point Cloud.
subscribe_stereo	bool	false	Subscribe to binocular images.
subscribe_odom_info	bool	false	Subscribe to odometer information messages.
subscribe_rgbd	bool	false	Subscribe to the rgbd_image topic.
frame_id	string	base_link	The coordinate system attached to the mobile base.
odom_frame_id	string	' '	The coordinate system of the odometer. If empty, rtabmapviz will subscribe to the odom topic for odometers. If set, get odometer from tf.
tf_prefix	string	' '	The prefix to add to the generated tf.
wait_for_transform	bool	false	When the tf transform is still unavailable, wait for the transform(up to 1 second).
queue_size	int	10	Message queue size per synchronization topic.
rgbd_cameras	int	1	The number of RGB-D cameras to use(when subscribe_rgbd is true). Currently, up to 4 cameras can be synchronized simultaneously.

7.5.3 the required tf conversion

- base_link → sensor coordinate system
- odom → base_link
- map → odom