

11.RTAB-Map 3D mapping navigation

11.RTAB-Map 3D mapping navigation

1. Introduction
2. Configuration before use
3. Map construction
4. Navigation
 - 4.1. Start the camera
 - 4.2. Start rviz to display the map [Start on the virtual machine side]
 - 4.3. Display rtabmap_viz [Virtual machine side startup]
 - 4.4. Start navigation node
 - 4.5. Single-point navigation
 - 4.6. Multi-point navigation
5. Node analysis
 - 5.1. Display calculation graph
 - 5.2. Details of rtabmap navigation related nodes
 - 5.3. TF transformation

rtabmap official website: <http://introlab.github.io/rtabmap/>

rtabmap ros-foxy: <https://github.com/introlab/rtabmap/tree/foxy-devel>

1. Introduction

rtabmap is a RGB-D image-based SLAM method that uses a bag-of-words-based global Bayesian loop closure detector to build maps in real-time in large-scale environments.

The features of rtabmap include:

- It can use a handheld RGBD camera for 6 degrees of freedom RGB-D mapping, or use a robot equipped with lidar for 3 degrees of freedom (2D laser) or 6 degrees of freedom (3D laser) mapping.
- It uses a memory management mechanism that divides the map into working memory (WM), short-term memory (STM) and long-term memory (LTM) to limit the number of anchor points used for closed-loop detection and graph optimization to ensure real-time performance
- It uses a discrete Bayesian filter to estimate the probability of forming a closed loop and selects which anchor points to transfer or retrieve based on weights and time.
- It uses an incremental dictionary creation method instead of pre-trained dictionaries, so that it can adapt to different environments.
- It can run under ROS or as a standalone library and application.

2. Configuration before use

Note: Since the Muto series robots are equipped with multiple radar devices, the factory system has been configured with routines for multiple devices. However, since the product cannot be automatically recognized, the radar model needs to be manually set.

After entering the container: Make the following modifications according to the lidar type:

```
root@ubuntu:/# cd
root@ubuntu:~# vim .bashrc
```



```
# env
alias python=python3
export ROS_DOMAIN_ID=26

export ROBOT_TYPE=Muto
export RPLIDAR_TYPE=a1      # a1, 4ROS
echo "-----"
echo -e "ROS_DOMAIN_ID: \033[32m$ROS_DOMAIN_ID\033[0m"
echo -e "my_robot_type: \033[32m$ROBOT_TYPE\033[0m | my_lidar: \033[32m$RPLIDAR_TYPE\033[0m"
echo "-----"

#colcon_cd
source /usr/share/colcon_cd/function/colcon_cd.sh
export _colcon_cd_root=/root/yahboomcar_ros2_ws/yahboomcar_ws
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash

#ros2
source /opt/ros/foxy/setup.bash
source /root/yahboomcar_ros2_ws/yahboomcar_ws/install/setup.bash
source /root/yahboomcar_ros2_ws/software/library_ws/install/setup.bash
```

After the modification is completed, save and exit vim, and then execute:

```
root@jetson-desktop:~# source .bashrc
-----
ROS_DOMAIN_ID: 26
my_robot_type: Muto | my_lidar: a1
-----
root@jetson-desktop:~#
```

You can see the current modified radar type.

Note: The depth camera needs to be connected directly to the Muto main control, not to the USB-HUB. This can improve the transmission efficiency of the depth camera and improve the mapping and navigation effects. Otherwise, mapping and navigation may become impossible

3. Map construction

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

First, the port binding operation needs to be done on the host machine [that is, Muto's jetson] [see the port binding tutorial chapter]. The three devices of radar, serial port and camera are mainly used here;

Then check whether the radar and serial port device are in the port binding state: on the host machine [that is, the jetson of the car], refer to the following command to execute the check. The successful binding is as follows:

```
jetson@ubuntu:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx 1 root root 7 Apr 21 14:52 myserial → ttyUSB0
lrwxrwxrwx 1 root root 7 Apr 21 14:52 rplidar → ttyUSB1
crwxrwxrwx 1 root dialout 188, 0 Apr 21 14:52 ttyUSB0
crwxrwxrwx 1 root dialout 188, 1 Apr 21 14:52 ttyUSB1
```

Check whether the camera is in port binding state

```
jetson@jetson-desktop:~$ ll /dev/astra*
lrwxrwxrwx 1 root root 15 6月 21 14:27 /dev/astradepth → bus/usb/001/007
lrwxrwxrwx 1 root root 15 6月 21 14:27 /dev/astrauvc → bus/usb/001/008
```

If it shows that the lidar, serial port, and camera devices are not bound, you can plug and unplug the USB cable to check again.

Enter the docker container and execute the following launch file in a terminal:

1. Start Astra camera

```
ros2 launch astra_camera astro_pro_plus.launch.xml
```

2. Start mapping

```
ros2 launch yahboomcar_nav map_rtabmap_launch.py
```

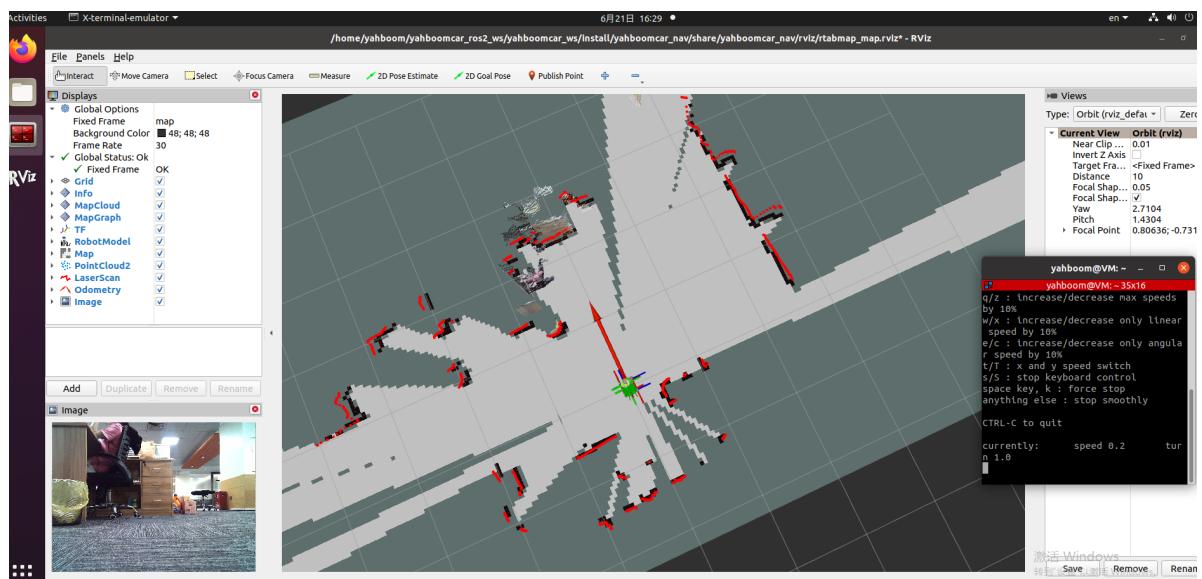
3. Start the lidar odometer

```
ros2 launch rf2o_laser_odometry rf2o_laser_odometry.launch.py
```

4. Start rviz to display the map. It is recommended to perform this step in [Virtual Machine].

Multi-machine communication needs to be configured in the virtual machine.

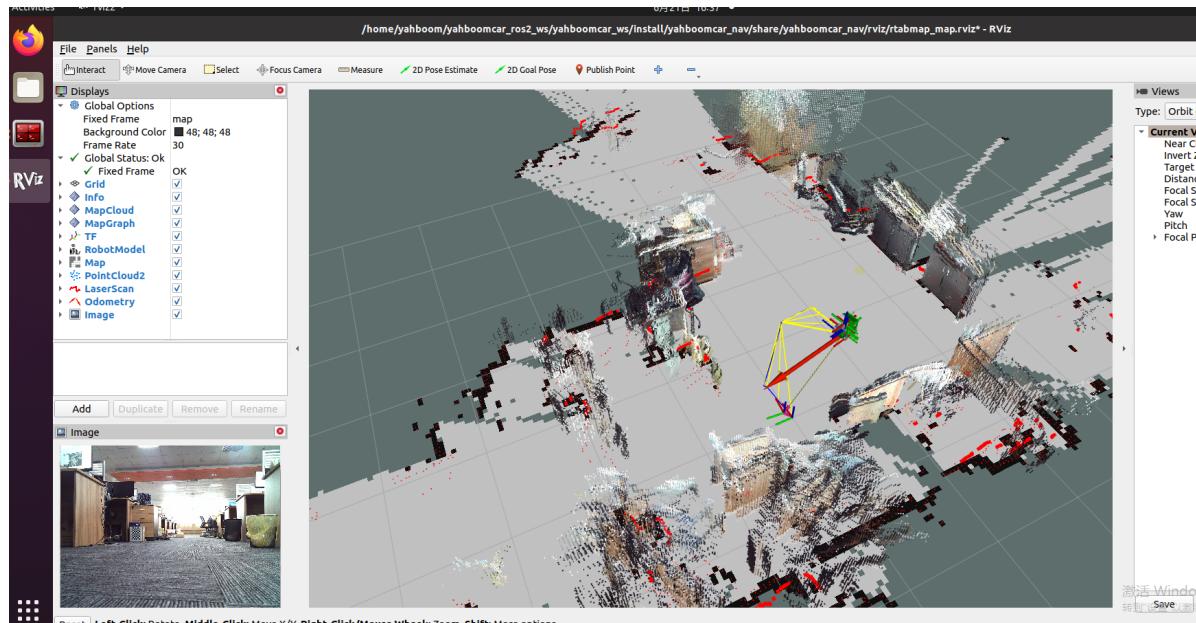
```
ros2 launch yahboomcar_nav display_rtabmap_map.launch.py
```



5. Start the keyboard control node. It is recommended to perform this step in [Virtual Machine].
Multi-machine communication needs to be configured in the virtual machine.

Or use the remote control [Slowly Move Muto] to start mapping until a complete map is created

```
ros2 run yahboomcar_ctrl yahboom_keyboard  
Or  
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```



6. Map saving

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

4. Navigation

Enter the docker container and execute in a terminal:

4.1. Start the camera

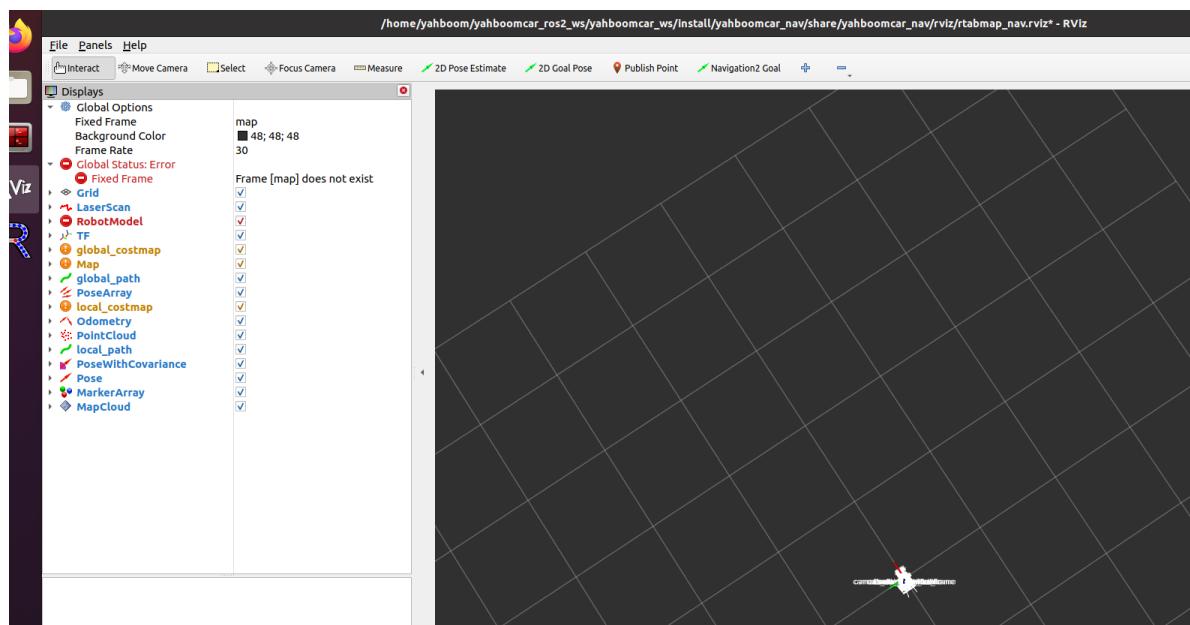
```
ros2 launch astra_camera astro_pro_plus.launch.xml
```

4.2. Start rviz to display the map [Start on the virtual machine side]

Multi-machine communication is configured in the Ubuntu virtual machine and docker container. It is recommended to start it in the virtual machine in this step: to maintain time synchronization and reduce resource consumption, because if you use vnc, it is very dependent on the network and may cause navigation failure.

[Note that you must first start the node that displays the map, and then start the navigation node in step 3. This is because the navigation2 terminal map topic is only published once. If you start the navigation node first and then start the rviz display, you may not be able to subscribe to the map topic published only once, resulting in the map not being displayed]

```
ros2 launch yahboomcar_nav display_rtabmap_nav.launch.py
```

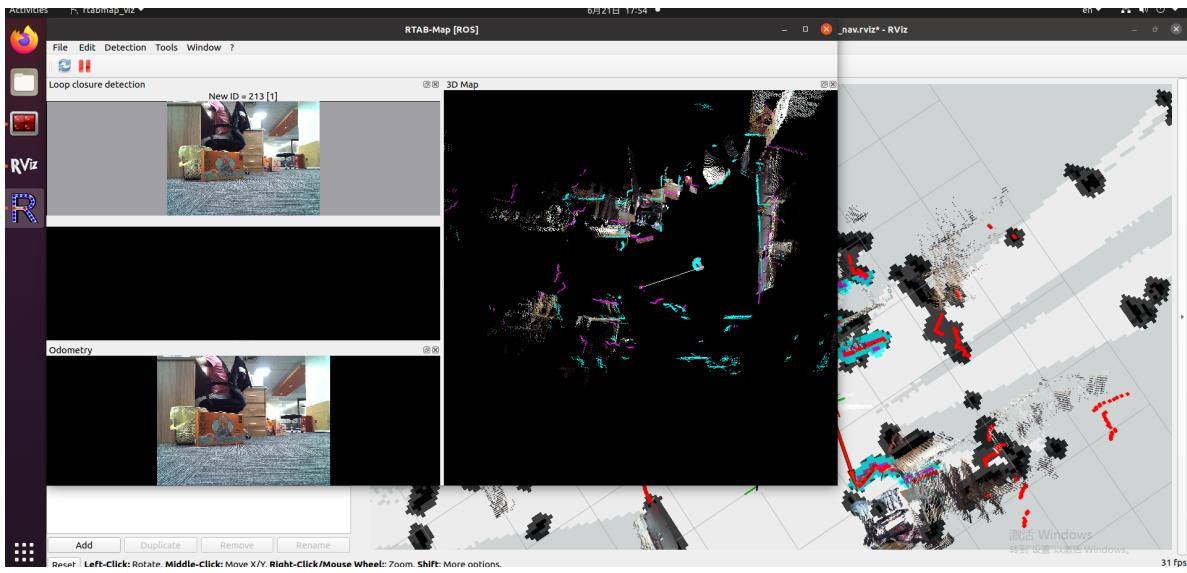


4.3. Display rtabmap_viz [Virtual machine side startup]

rtabmap_viz is the visual interface of RTAB-Map. It is an encapsulation of the RTAB-Map GUI graphics library. It is similar to rviz but has options for RTAB-Map. It can subscribe to different topics, such as odom, rgb/image, depth/image, scan, etc., to display the SLAM process and results, and at the same time load the 3D map into rviz. It is recommended to start this step in a virtual machine.

Note that the map will not be displayed after this step is started. You need to wait for the next navigation node to be started before it will be displayed.

```
ros2 launch yahboomcar_nav rtabmap_viz.launch.py
```



4.4. Start navigation node

Navigation can be divided into single-point navigation and multi-point navigation, which will be introduced below.

Enter the docker container and execute in a terminal:

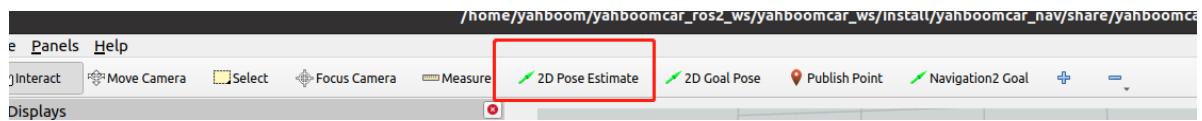
1. Start the chassis and load the map

```
ros2 launch yahboomcar_nav rtabmap_bringup_launch.py
```

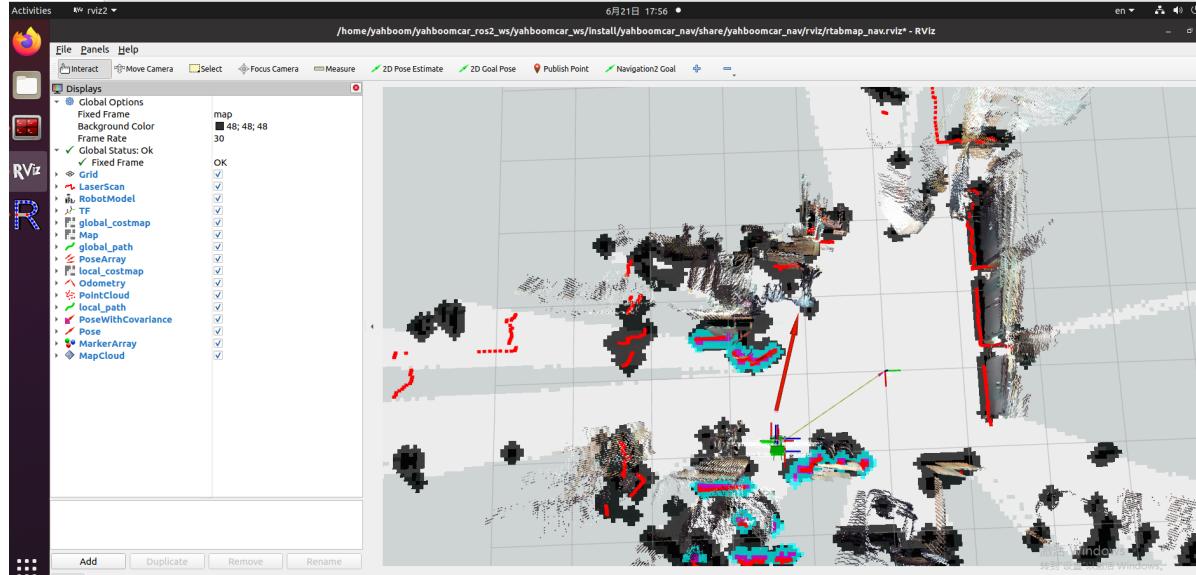
2. Start the navigation node

```
ros2 launch yahboomcar_nav navigation_rtabmap_launch.py
```

3. Click [2D Pose Estimate] on rviz, then compare Muto's pose and mark an initial pose for Muto on the map;



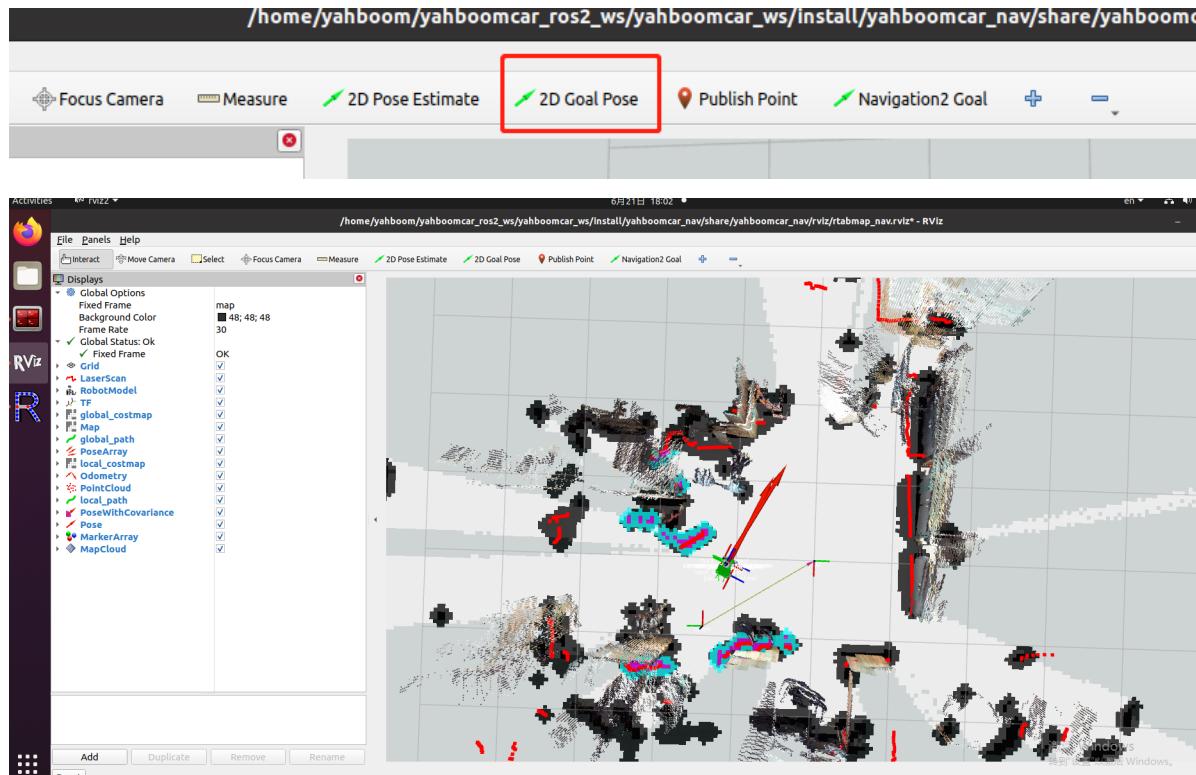
The display after marking is as follows:



4. Compare the overlap between the lidar scanning point and the obstacle, and set the initial pose for Muto multiple times until the radar scanning point and the obstacle roughly overlap;

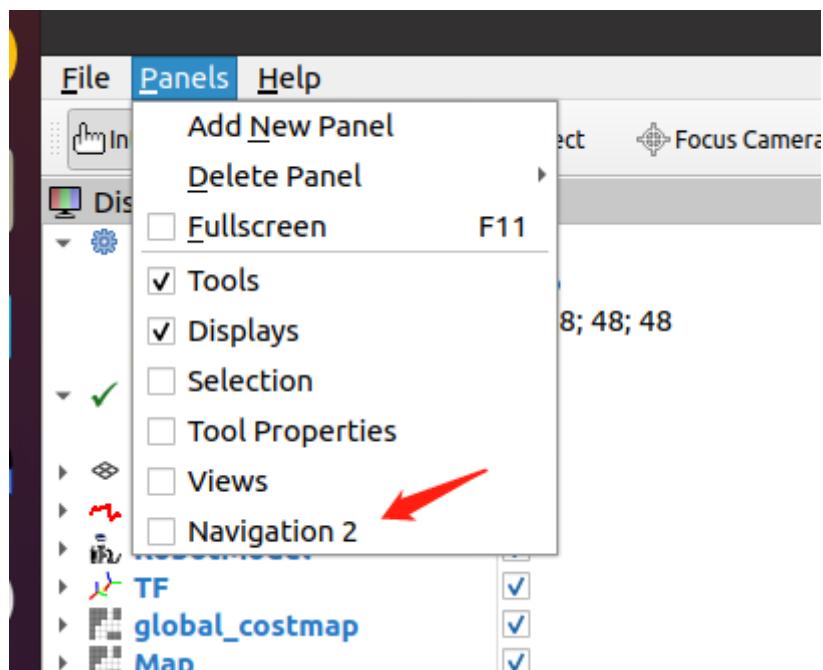
4.5. Single-point navigation

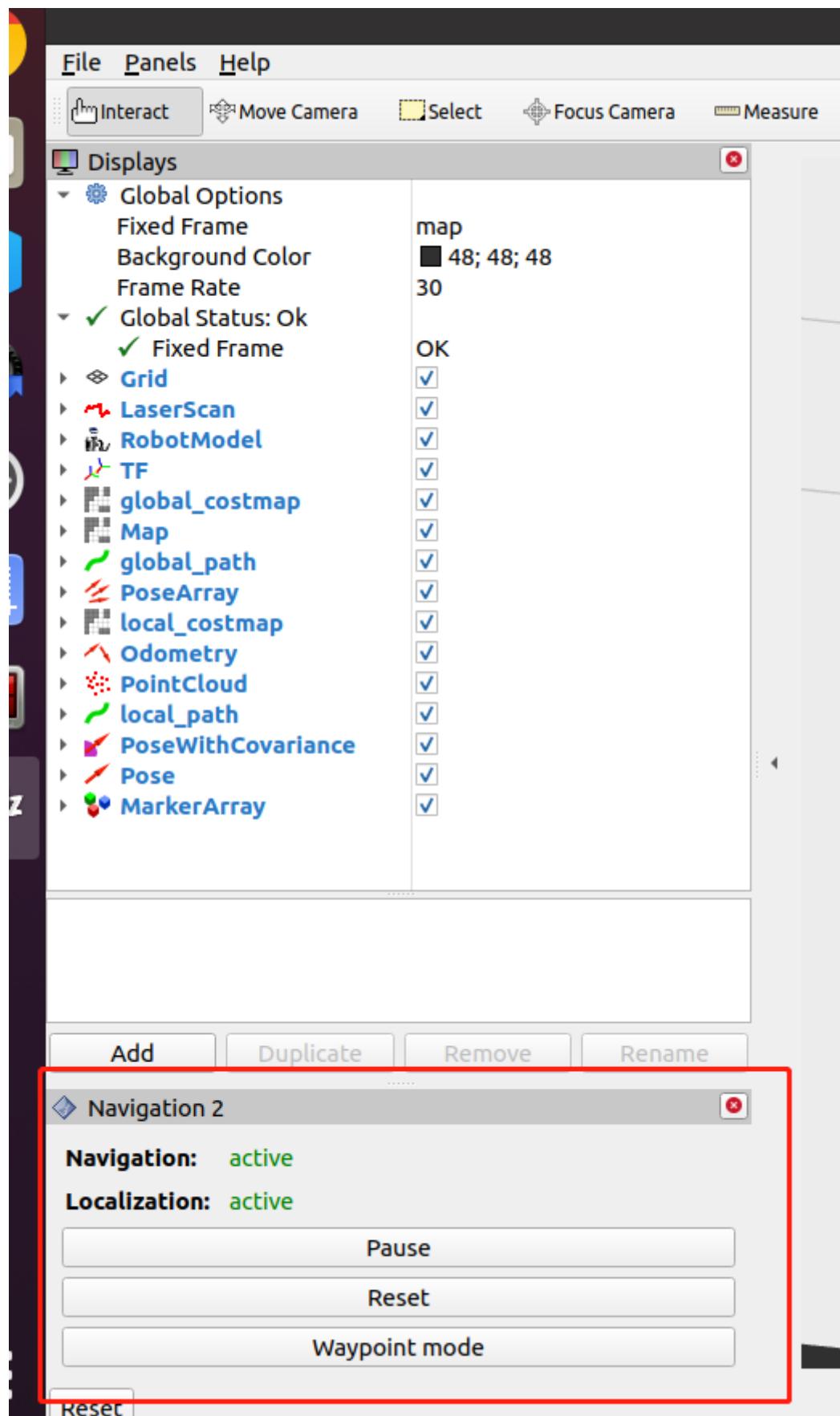
After the initial pose is set, you can click [2D Goal Pose] to set a navigation target point, and Muto will start single-point navigation;



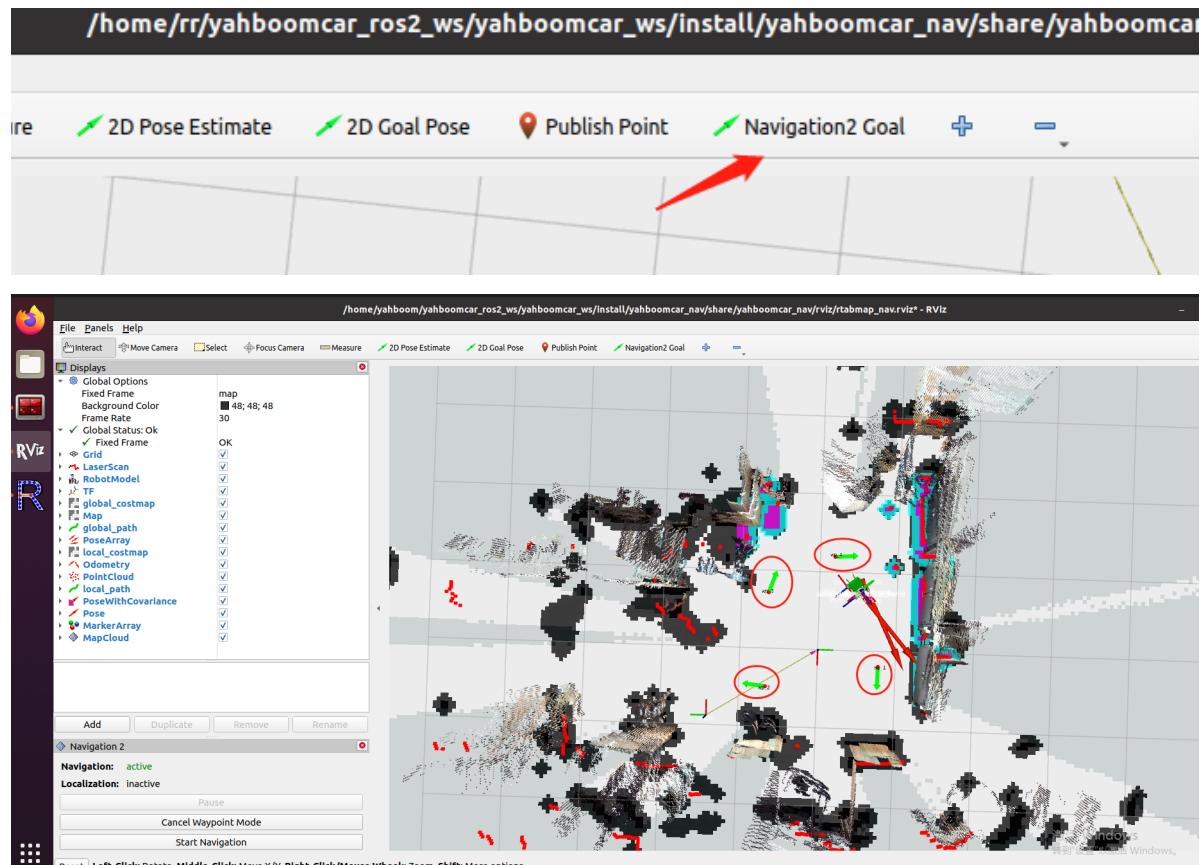
4.6. Multi-point navigation

1. After the initial pose is set, you can click [Panels] in the upper left corner of rviz --- select [Navigation 2], and the [Navigation 2] panel will be displayed.

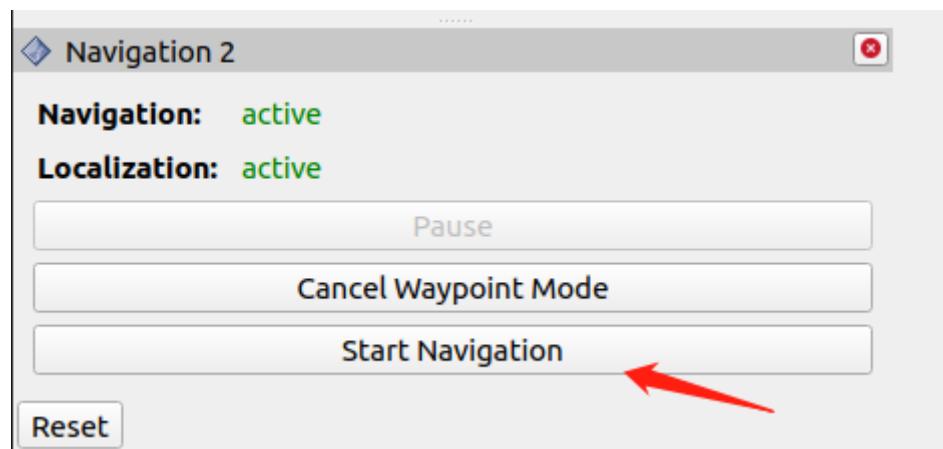




2. Click [Waypoint mode] in the picture above, then click [Navigation2 Goal] on rviz to mark a target point on the map. Click [Navigation2 Goal] again to mark the second goal point on the map, and the cycle continues , you can mark multiple target points at one time;



3. After marking multiple target points, click [Start Navigation] to start multi-point navigation.
After multi-point navigation is completed, Muto will stay at the pose of the last target point;



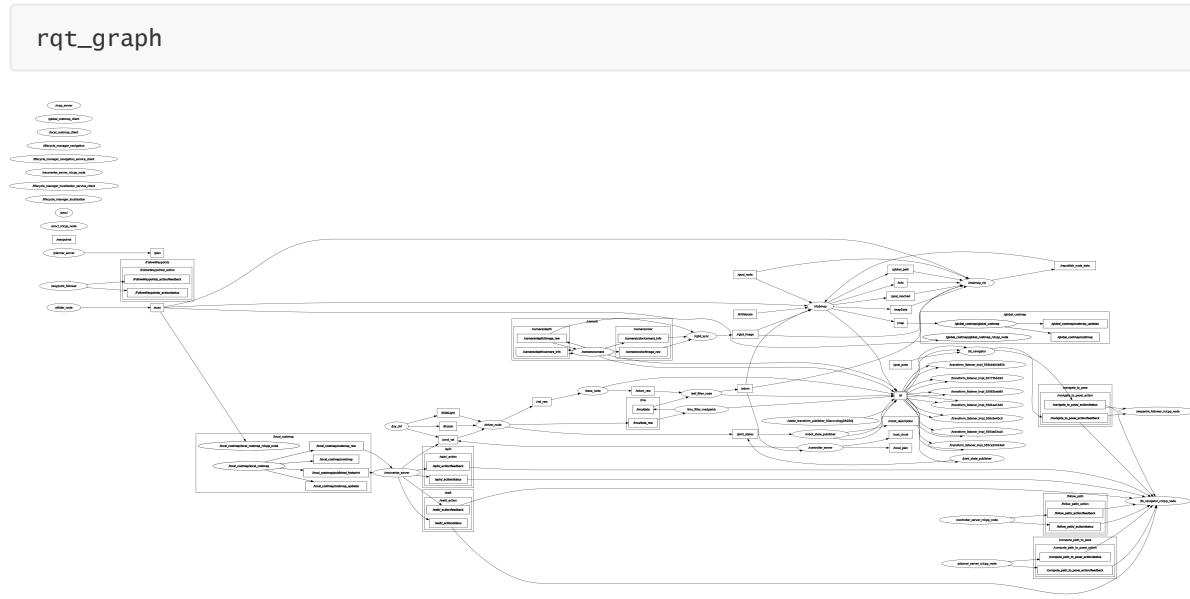
4. It may appear during the navigation process. This is due to the navigation2 itself in the ros-foxy version. The subsequent ros2 version has been fixed.

```
[bt_navigator-7] [INFO] [1682061491.700042174] [bt_navigator]: Begin navigating from current location to (-6.95, -1.41)
[bt_navigator-7] [ERROR] [1682061491.727441591] [bt_navigator]: Action server failed while executing action callback: "send_goal failed"
[bt_navigator-7] [WARN] [1682061491.727594875] [bt_navigator]: [navigate_to_pose] [ActionServer] Aborting handle.
```

5. rviz may exit during the navigation process. This may be caused by insufficient resources.
You can turn off the display of rtabmap_viz on the virtual machine side after the navigation node is started to fully display the 3D map.

5. Node analysis

5.1. Display calculation graph



5.2. Details of rtabmap navigation related nodes

/rgbd_sync

```
root@jetson-desktop:~# ros2 node info /rgbd_sync
/rbgd_sync
Subscribers:
/camera/color/camera_info: sensor_msgs/msg/CameraInfo
/camera/color/image_raw: sensor_msgs/msg/Image
/camera/depth/image_raw: sensor_msgs/msg/Image
/parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rgbd_image: rtabmap_msgs/msg/RGBDImage
/rgbd_image/compressed: rtabmap_msgs/msg/RGBDImage
/rosout: rcl_interfaces/msg/Log
Service Servers:
/rgbd_sync/describe_parameters: rcl_interfaces/srv/DescribeParameters
/rgbd_sync/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/rgbd_sync/get_parameters: rcl_interfaces/srv/GetParameters
/rgbd_sync/list_parameters: rcl_interfaces/srv/ListParameters
/rgbd_sync/set_parameters: rcl_interfaces/srv/SetParameters
/rgbd_sync/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
Action Clients:
```

/rtabmap

```
root@jetson-desktop:~# ros2 node info /rtabmap
/rtabmap
Subscribers:
/global_pose: geometry_msgs/msg/PoseWithCovarianceStamped
/goal: geometry_msgs/msg/PoseStamped
/goal_node: rtabmap_msgs/msg/Goal
/gps/fix: sensor_msgs/msg/NavSatFix
 imu: sensor_msgs/msg/Imu
/initialpose: geometry_msgs/msg/PoseWithCovarianceStamped
/odom: nav_msgs/msg/Odometry
/parameter_events: rcl_interfaces/msg/ParameterEvent
/republish_node_data: std_msgs/msg/Int32MultiArray
/rgbd_image: rtabmap_msgs/msg/RGBDImage
/scan: sensor_msgs/msg/LaserScan
/user_data_async: rtabmap_msgs/msg/UserData
Publishers:
/cloud_ground: sensor_msgs/msg/PointCloud2
/cloud_map: sensor_msgs/msg/PointCloud2
/cloud_obstacles: sensor_msgs/msg/PointCloud2
/global_path: nav_msgs/msg/Path
/global_path_nodes: rtabmap_msgs/msg/Path
/goal_out: geometry_msgs/msg/PoseStamped
/goal_reached: std_msgs/msg/Bool
/grid_prob_map: nav_msgs/msg/OccupancyGrid
/info: rtabmap_msgs/msg/Info
/labels: visualization_msgs/msg/MarkerArray
/landmarks: geometry_msgs/msg/PoseArray
/local_grid_empty: sensor_msgs/msg/PointCloud2
/local_grid_ground: sensor_msgs/msg/PointCloud2
/local_grid_obstacle: sensor_msgs/msg/PointCloud2
/local_path: nav_msgs/msg/Path
/local_path_nodes: rtabmap_msgs/msg/Path
/localization_pose: geometry_msgs/msg/PoseWithCovarianceStamped
/map: nav_msgs/msg/OccupancyGrid
/mapData: rtabmap_msgs/msg/MapData
/mapGraph: rtabmap_msgs/msg/MapGraph
/mapodomCache: rtabmap_msgs/msg/MapGraph
/mapPath: nav_msgs/msg/Path
/octomap_binary: octomap_msgs/msg/Octomap
/octomap_empty_space: sensor_msgs/msg/PointCloud2
```

```
/octomap_empty_space: sensor_msgs/msg/PointCloud2
/octomap_full: octomap_msgs/msg/Octomap
/octomap_global_frontier_space: sensor_msgs/msg/PointCloud2
/octomap_grid: nav_msgs/msg/OccupancyGrid
/octomap_ground: sensor_msgs/msg/PointCloud2
/octomap_obstacles: sensor_msgs/msg/PointCloud2
/octomap_occupied_space: sensor_msgs/msg/PointCloud2
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/tf: tf2_msgs/msg/TFMessage
Service Servers:
/add_link: rtabmap_msgs/srv/AddLink
/backup: std_srvs/srv/Empty
/cancel_goal: std_srvs/srv/Empty
/cleanup_local_grids: rtabmap_msgs/srv/CleanupLocalGrids
/detect_more_loop_closures: rtabmap_msgs/srv/DetectMoreLoopClosures
/get_map: nav_msgs/srv/GetMap
/get_map_data: rtabmap_msgs/srv/GetMap
/get_map_data2: rtabmap_msgs/srv/GetMap2
/get_node_data: rtabmap_msgs/srv/GetNodeData
/get_nodes_in_radius: rtabmap_msgs/srv/GetNodesInRadius
/get_plan: nav_msgs/srv/GetPlan
/get_plan_nodes: rtabmap_msgs/srv/GetPlan
/get_prob_map: nav_msgs/srv/GetMap
/global_bundle_adjustment: rtabmap_msgs/srv/GlobalBundleAdjustment
/list_labels: rtabmap_msgs/srv/ListLabels
/load_database: rtabmap_msgs/srv/LoadDatabase
/log_debug: std_srvs/srv/Empty
/log_error: std_srvs/srv/Empty
/log_info: std_srvs/srv/Empty
/log_warning: std_srvs/srv/Empty
/octomap_binary: octomap_msgs/srv/GetOctomap
/octomap_full: octomap_msgs/srv/GetOctomap
/pause: std_srvs/srv/Empty
/publish_map: rtabmap_msgs/srv/PublishMap
/remove_label: rtabmap_msgs/srv/RemoveLabel
/reset: std_srvs/srv/Empty
/resume: std_srvs/srv/Empty
/rtabmap/describe_parameters: rcl_interfaces/srv/DescribeParameters
/rtabmap/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
```

```
/rtabmap/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/rtabmap/get_parameters: rcl_interfaces/srv/GetParameters
/rtabmap/list_parameters: rcl_interfaces/srv/ListParameters
/rtabmap/set_parameters: rcl_interfaces/srv/SetParameters
/rtabmap/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
/set_goal: rtabmap_msgs/srv/SetGoal
/set_label: rtabmap_msgs/srv/SetLabel
/set_mode_localization: std_srvs/srv/Empty
/set_mode_mapping: std_srvs/srv/Empty
/trigger_new_map: std_srvs/srv/Empty
/update_parameters: std_srvs/srv/Empty
Service Clients:
/rtabmap/describe_parameters: rcl_interfaces/srv/DescribeParameters
/rtabmap/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/rtabmap/get_parameters: rcl_interfaces/srv/GetParameters
/rtabmap/list_parameters: rcl_interfaces/srv/ListParameters
/rtabmap/set_parameters: rcl_interfaces/srv/SetParameters
/rtabmap/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Action Servers:
```

```
Action Clients:
```

```
/rtabmap_viz
```

```
root@jetson-desktop:~# ros2 node info /rtabmap_viz
/rtabmap_viz
Subscribers:
/global_path: nav_msgs/msg/Path
/goal_node: rtabmap_msgs/msg/Goal
/goal_reached: std_msgs/msg/Bool
/info: rtabmap_msgs/msg/Info
/mapData: rtabmap_msgs/msg/MapData
/odom: nav_msgs/msg/Odometry
/rgbd_image: rtabmap_msgs/msg/RGBDImage
/scan: sensor_msgs/msg/LaserScan
Publishers:
/republish_node_data: std_msgs/msg/Int32MultiArray
Service Servers:
Service Clients:
Action Servers:
Action Clients:
```

/bt_navigator

```
root@ubuntu:/# ros2 node info /bt_navigator
/bt_navigator
Subscribers:
/goal_pose: geometry_msgs/msg/PoseStamped
/parameter_events: rcl_interfaces/msg/ParameterEvent
/tf: tf2_msgs/msg/TFMessage
/tf_static: tf2_msgs/msg/TFMessage
Publishers:
/bt_navigator/transition_event: lifecycle_msgs/msg/TransitionEvent
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
Service Servers:
/bt_navigator/change_state: lifecycle_msgs/srv/ChangeState
/bt_navigator/describe_parameters: rcl_interfaces/srv/DescribeParameters
/bt_navigator/get_available_states: lifecycle_msgs/srv/GetAvailableStates
/bt_navigator/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
/bt_navigator/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/bt_navigator/get_parameters: rcl_interfaces/srv/GetParameters
/bt_navigator/get_state: lifecycle_msgs/srv/GetState
/bt_navigator/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
/bt_navigator/list_parameters: rcl_interfaces/srv/ListParameters
/bt_navigator/set_parameters: rcl_interfaces/srv/SetParameters
/bt_navigator/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
/navigate_to_pose: nav2_msgs/action/NavigateToPose
Action Clients:
```

/controller_server

```

root@ubuntu:/# ros2 node info /controller_server
/controller_server
Subscribers:
  /odom: nav_msgs/msg/Odometry
  /parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
  /cmd_vel: geometry_msgs/msg/Twist
  /controller_server/transition_event: lifecycle_msgs/msg/TransitionEvent
  /cost_cloud: sensor_msgs/msg/PointCloud
  /evaluation: dwb_msgs/msg/LocalPlanEvaluation
  /local_plan: nav_msgs/msg/Path
  /marker: visualization_msgs/msg/MarkerArray
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /received_global_plan: nav_msgs/msg/Path
  /rosout: rcl_interfaces/msg/Log
  /transformed_global_plan: nav_msgs/msg/Path
Service Servers:
  /controller_server/change_state: lifecycle_msgs/srv/ChangeState
  /controller_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /controller_server/get_available_states: lifecycle_msgs/srv/GetAvailableStates
  /controller_server/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
  /controller_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /controller_server/get_parameters: rcl_interfaces/srv/GetParameters
  /controller_server/get_state: lifecycle_msgs/srv/GetState
  /controller_server/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
  /controller_server/list_parameters: rcl_interfaces/srv/ListParameters
  /controller_server/set_parameters: rcl_interfaces/srv/SetParameters
  /controller_server/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
  /controller_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /controller_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /controller_server/get_parameters: rcl_interfaces/srv/GetParameters
  /controller_server/list_parameters: rcl_interfaces/srv/ListParameters
  /controller_server/set_parameters: rcl_interfaces/srv/SetParameters
  /controller_server/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Action Servers:
Action Clients:
```

/global_costmap/global_costmap

```

root@ubuntu:/# ros2 node info /global_costmap/global_costmap
/global_costmap/global_costmap
Subscribers:
  /global_costmap/footprint: geometry_msgs/msg/Polygon
  /map: nav_msgs/msg/OccupancyGrid
  /parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
  /global_costmap/costmap: nav_msgs/msg/OccupancyGrid
  /global_costmap/costmap_raw: nav2_msgs/msg/Costmap
  /global_costmap/costmap_updates: map_msgs/msg/OccupancyGridUpdate
  /global_costmap/global_costmap/transition_event: lifecycle_msgs/msg/TransitionEvent
  /global_costmap/published_footprint: geometry_msgs/msg/PolygonStamped
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
Service Servers:
  /global_costmap/clear_around_global_costmap: nav2_msgs/srv/ClearCostmapAroundRobot
  /global_costmap/clear_entirely_global_costmap: nav2_msgs/srv/ClearEntireCostmap
  /global_costmap/clear_except_global_costmap: nav2_msgs/srv/ClearCostmapExceptRegion
  /global_costmap/get_costmap: nav2_msgs/srv/GetCostmap
  /global_costmap/global_costmap/change_state: lifecycle_msgs/srv/ChangeState
  /global_costmap/global_costmap/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /global_costmap/global_costmap/get_available_states: lifecycle_msgs/srv/GetAvailableStates
  /global_costmap/global_costmap/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
  /global_costmap/global_costmap/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /global_costmap/global_costmap/get_parameters: rcl_interfaces/srv/GetParameters
  /global_costmap/global_costmap/get_state: lifecycle_msgs/srv/GetState
  /global_costmap/global_costmap/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
  /global_costmap/global_costmap/list_parameters: rcl_interfaces/srv/ListParameters
  /global_costmap/global_costmap/set_parameters: rcl_interfaces/srv/SetParameters
  /global_costmap/global_costmap/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
Action Clients:
```

/local_costmap/local_costmap

```

root@ubuntu:/# ros2 node info /local_costmap/local_costmap
/local_costmap/local_costmap
Subscribers:
  /local_costmap/footprint: geometry_msgs/msg/Polygon
  /parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
  /local_costmap/costmap: nav_msgs/msg/OccupancyGrid
  /local_costmap/costmap_raw: nav2_msgs/msg/Costmap
  /local_costmap/costmap_updates: map_msgs/msg/OccupancyGridUpdate
  /local_costmap/local_costmap/transition_event: lifecycle_msgs/msg/TransitionEvent
  /local_costmap/published_footprint: geometry_msgs/msg/PolygonStamped
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /rosout: rcl_interfaces/msg/Log
Service Servers:
  /local_costmap/clear_around_local_costmap: nav2_msgs/srv/ClearCostmapAroundRobot
  /local_costmap/clear_entirely_local_costmap: nav2_msgs/srv/ClearEntireCostmap
  /local_costmap/clear_except_local_costmap: nav2_msgs/srv/ClearCostmapExceptRegion
  /local_costmap/get_costmap: nav2_msgs/srv/GetCostmap
  /local_costmap/local_costmap/change_state: lifecycle_msgs/srv/ChangeState
  /local_costmap/local_costmap/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /local_costmap/local_costmap/get_available_states: lifecycle_msgs/srv/GetAvailableStates
  /local_costmap/local_costmap/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
  /local_costmap/local_costmap/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /local_costmap/local_costmap/get_parameters: rcl_interfaces/srv/GetParameters
  /local_costmap/local_costmap/get_state: lifecycle_msgs/srv/GetState
  /local_costmap/local_costmap/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
  /local_costmap/local_costmap/list_parameters: rcl_interfaces/srv/ListParameters
  /local_costmap/local_costmap/set_parameters: rcl_interfaces/srv/SetParameters
  /local_costmap/local_costmap/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
Action Clients:

```

/planner_server

```

root@ubuntu:/# ros2 node info /planner_server
/planner_server
Subscribers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
  /parameter_events: rcl_interfaces/msg/ParameterEvent
  /plan: nav_msgs/msg/Path
  /planner_server/transition_event: lifecycle_msgs/msg/TransitionEvent
  /rosout: rcl_interfaces/msg/Log
Service Servers:
  /planner_server/change_state: lifecycle_msgs/srv/ChangeState
  /planner_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /planner_server/get_available_states: lifecycle_msgs/srv/GetAvailableStates
  /planner_server/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
  /planner_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /planner_server/get_parameters: rcl_interfaces/srv/GetParameters
  /planner_server/get_state: lifecycle_msgs/srv/GetState
  /planner_server/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
  /planner_server/list_parameters: rcl_interfaces/srv/ListParameters
  /planner_server/set_parameters: rcl_interfaces/srv/SetParameters
  /planner_server/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
  /planner_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
  /planner_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
  /planner_server/get_parameters: rcl_interfaces/srv/GetParameters
  /planner_server/list_parameters: rcl_interfaces/srv/ListParameters
  /planner_server/set_parameters: rcl_interfaces/srv/SetParameters
  /planner_server/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Action Servers:
Action Clients:

```

/recoveries_server

```

root@ubuntu:/# ros2 node info /recoveries_server
/recoveries_server
Subscribers:
/local_costmap/costmap_raw: nav2_msgs/msg/Costmap
/local_costmap/published_footprint: geometry_msgs/msg/PolygonStamped
/parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
/cmd_vel: geometry_msgs/msg/Twist
/parameter_events: rcl_interfaces/msg/ParameterEvent
/recoveries_server/transition_event: lifecycle_msgs/msg/TransitionEvent
/rosout: rcl_interfaces/msg/Log
Service Servers:
/recoveries_server/change_state: lifecycle_msgs/srv/ChangeState
/recoveries_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
/recoveries_server/get_available_states: lifecycle_msgs/srv/GetAvailableStates
/recoveries_server/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
/recoveries_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/recoveries_server/get_parameters: rcl_interfaces/srv/GetParameters
/recoveries_server/get_state: lifecycle_msgs/srv/GetState
/recoveries_server/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
/recoveries_server/list_parameters: rcl_interfaces/srv/ListParameters
/recoveries_server/set_parameters: rcl_interfaces/srv/SetParameters
/recoveries_server/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
/backup: nav2_msgs/action/BackUp
/spin: nav2_msgs/action/Spin
/wait: nav2_msgs/action/Wait
Action Clients:

```

/waypoint_follower

```

root@ubuntu:/# ros2 node info /waypoint_follower
/waypoint_follower
Subscribers:
/parameter_events: rcl_interfaces/msg/ParameterEvent
Publishers:
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/waypoint_follower/transition_event: lifecycle_msgs/msg/TransitionEvent
Service Servers:
/waypoint_follower/change_state: lifecycle_msgs/srv/ChangeState
/waypoint_follower/describe_parameters: rcl_interfaces/srv/DescribeParameters
/waypoint_follower/get_available_states: lifecycle_msgs/srv/GetAvailableStates
/waypoint_follower/get_available_transitions: lifecycle_msgs/srv/GetAvailableTransitions
/waypoint_follower/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
/waypoint_follower/get_parameters: rcl_interfaces/srv/GetParameters
/waypoint_follower/get_state: lifecycle_msgs/srv/GetState
/waypoint_follower/get_transition_graph: lifecycle_msgs/srv/GetAvailableTransitions
/waypoint_follower/list_parameters: rcl_interfaces/srv/ListParameters
/waypoint_follower/set_parameters: rcl_interfaces/srv/SetParameters
/waypoint_follower/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
Service Clients:
Action Servers:
/FollowWaypoints: nav2_msgs/action/FollowWaypoints
Action Clients:

```

5.3. TF transformation

```
ros2 run tf2_tools view_frames.py
```

