6. Voice control multi-point navigation

The operating environment and software and hardware reference configuration are as follows:

- Reference model: ROSMASTER X3
- Robot hardware configuration: Arm series main control, Silan A1 lidar, AstraPro Plus depth camera
- Robot system: Ubuntu (no version required) + docker (version 20.10.21 and above)
- PC virtual machine: Ubuntu（22.04）＋ ROS2（Humble）
- Usage scenario: Use on a relatively clean 2D plane

6.1. Function description

By interacting with the voice recognition module on ROSMASTER, voice control can be used to realize multi-point navigation in the established map;

Note: Before using the functions in this section, please first learn to use the [Lidar Series Courses ----- Mapping Navigation Function] module;

6.2. Configure navigation points

Raspberry Pi PI5 master needs to enter the docker container first, Orin master does not need to enter,

Enter the docker container (for steps, please refer to [docker course chapter ----- 5. Enter the robot's docker container]),

1. and execute the following commands on each terminal:

```
ros2 launch yahboomcar_nav laser_bringup_launch.py
```

2. Open the virtual machine, configure multi-machine communication, and then execute the following command to display the rviz node

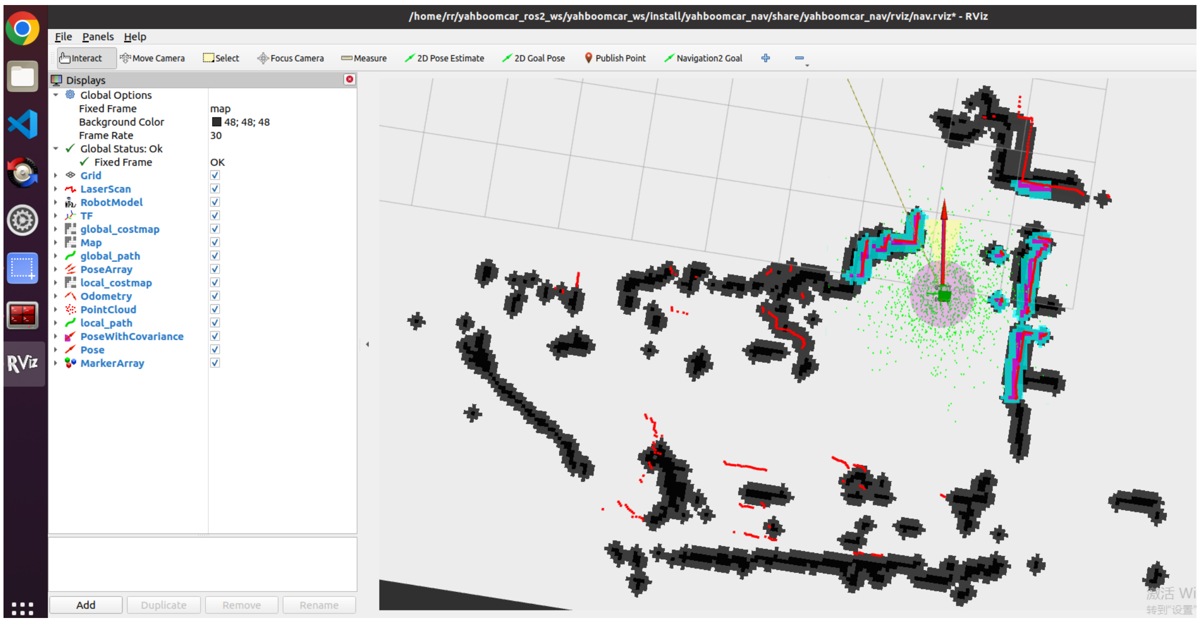
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

3. Execute navigation node in docker container

```
ros2 launch yahboomcar_nav navigation_teb_launch.py
```

4. At this time, click [2D Pose Estimate] in the rviz interface of the virtual machine, then compare the pose of the car and mark an initial pose for the car on the map;

The display after marking is as follows:



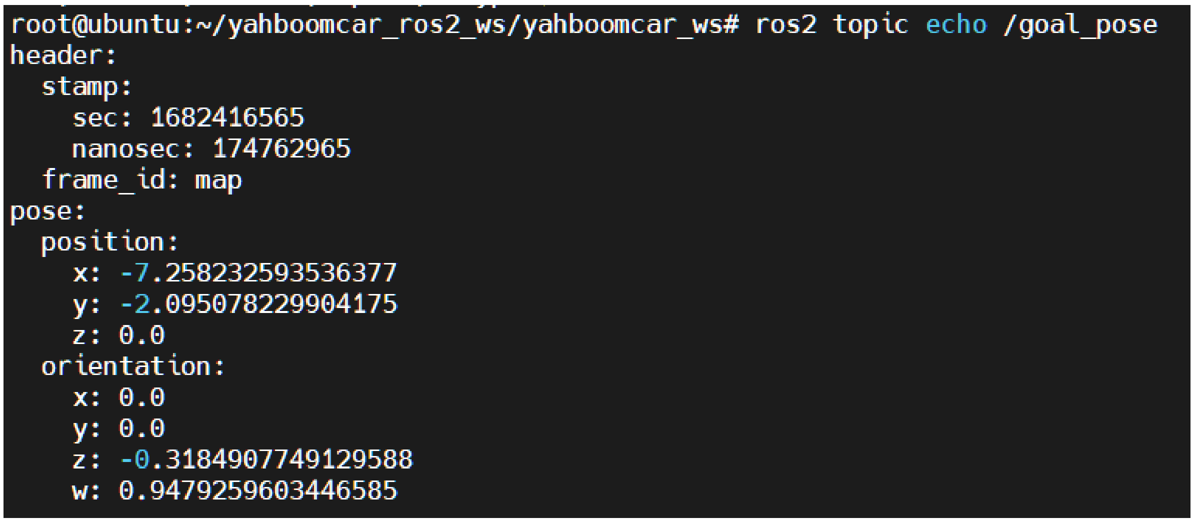
```
Global Options
  Fixed Frame          map
  Background Color     48; 48; 48
  Frame Rate           30
Global Status: Ok
  Fixed Frame          OK
```

5. Compare the overlap between the radar scanning point and the obstacle, and set the initial pose of the car multiple times until the radar scanning point and the obstacle roughly overlap;

6. Open another terminal to enter the docker container and execute

```
ros2 topic echo /goal_pose # Monitor /goal_pose topic
```

7. Click [2D Goal Pose] to set the first navigation target point. At this time, the car starts to navigate, and the monitored topic data will be received in step 6:

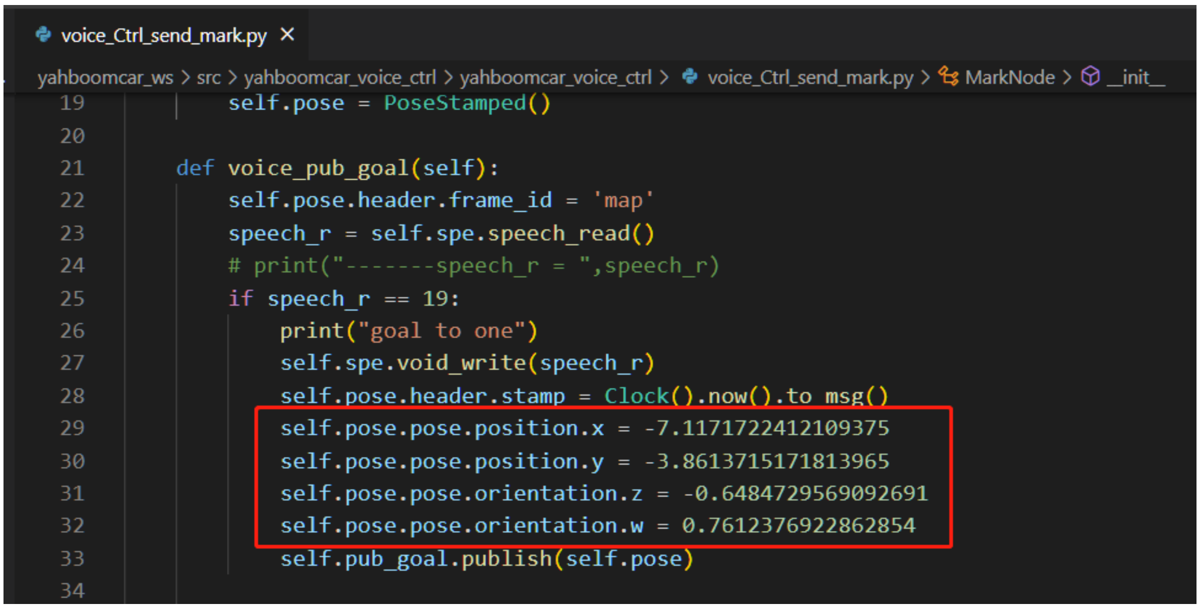


```
root@ubuntu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 topic echo /goal_pose
header:
  stamp:
    sec: 1682416565
    nanosec: 174762965
  frame_id: map
pose:
  position:
    x: -7.258232593536377
    y: -2.095078229904175
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: -0.3184907749129588
    w: 0.9479259603446585
```

8. Open the code at the following location:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctr
l/voice_Ctrl_send_mark.py
```

Modify the pose of the first navigation point to the one printed in step 7:


voice_Ctrl_send_mark.py
yahboomcar_ws > src > yahboomcar_voice_ctrl > yahboomcar_voice_ctrl > voice_Ctrl_send_mark.py > MarkNode > _init_

```
        self.pose = PoseStamped()

    def voice_pub_goal(self):
        self.pose.header.frame_id = 'map'
        speech_r = self.spe.speech_read()
        # print("-------speech_r = ",speech_r)
        if speech_r == 19:
            print("goal to one")
            self.spe.void_write(speech_r)
            self.pose.header.stamp = Clock().now().to_msg()
            self.pose.pose.position.x = -7.1171722412109375
            self.pose.pose.position.y = -3.8613715171813965
            self.pose.pose.orientation.z = -0.6484729569092691
            self.pose.pose.orientation.w = 0.7612376922862854
            self.pub_goal.publish(self.pose)
```

9. Use the same method to modify the poses of the other four navigation points.

After modification, **cd ~/yahboomcar_ros2_ws/yahboomcar_ws**

Compile, **colcon build --packages-select yahboomcar_voice_ctrl**

Update environment, **souce install/setup.bash**

6.3. Use voice multi-point navigation

Raspberry Pi PI5 master needs to enter the docker container first, Orin master does not need to enter,

Enter the docker container (for steps, please refer to [docker course chapter ----- 5. Enter the robot's docker container]),

1. and execute the following commands on the terminal:

```
ros2 launch yahboomcar_nav laser_bringup_launch.py
```

2. Open the virtual machine, configure multi-machine communication, and then execute the following command to display the rviz node

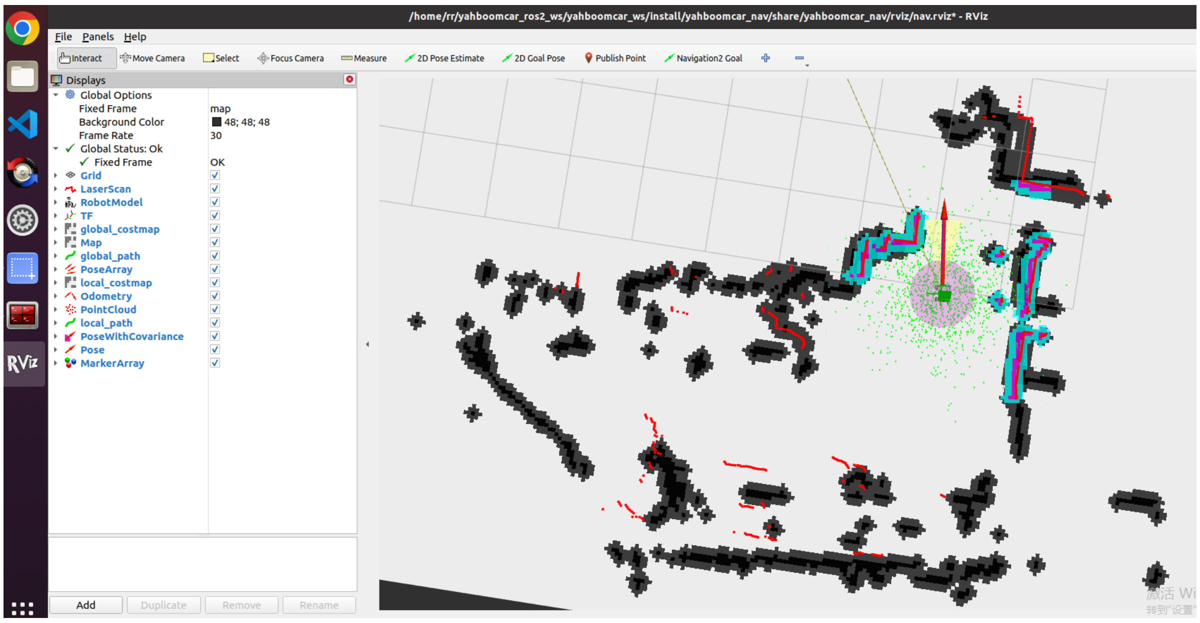
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

3. Execute navigation node in docker container

```
ros2 launch yahboomcar_nav navigation_teb_launch.py
```

4. At this time, click [2D Pose Estimate] in the rviz interface of the virtual machine, then compare the pose of the car and mark an initial pose for the car on the map;

The display after marking is as follows:



/home/rr/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/nav.rviz* - RViz

Global Options	
Fixed Frame	map
Background Color	48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
LaserScan	✓
RobotModel	✓
TF	✓
global_costmap	✓
Map	✓
global_path	✓
PoseArray	✓
local_costmap	✓
Odometry	✓
PointCloud	✓
local_path	✓
PoseWithCovariance	✓
Pose	✓
MarkerArray	✓

5. Compare the overlap between the radar scanning point and the obstacle, and set the initial pose of the car multiple times until the radar scanning point and the obstacle roughly overlap;

6. Open another terminal and enter the docker container, and execute to enable the voice control navigation node.

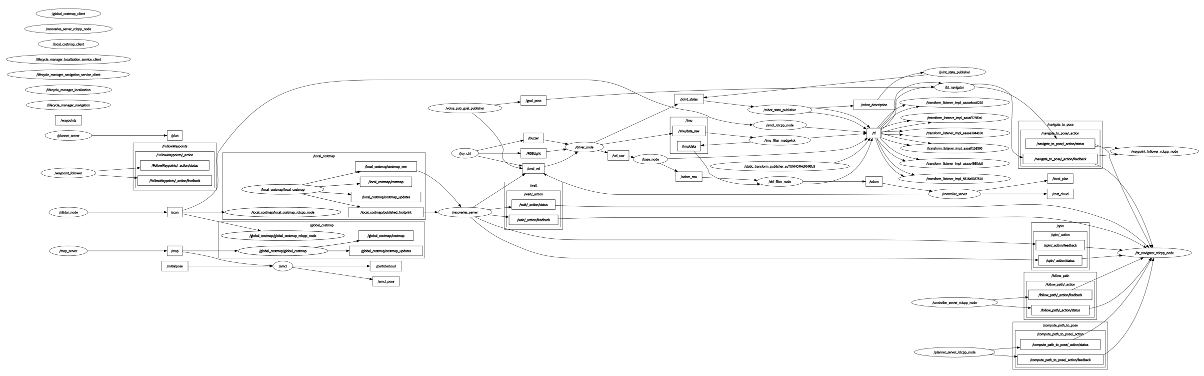
```
ros2 run yahboomcar_voice_ctrl voice_Ctrl_send_mark
```

7. Say "Hi Yahboom" to the voice module on the car. Wake up the voice module. After hearing the voice module feedback "at", continue to say "Navigate to No. 1"; the voice module will feedback "Okay" , "Going to position 1", and at the same time, the car starts to navigate to position 1. Navigation in other locations can be used in the same way.

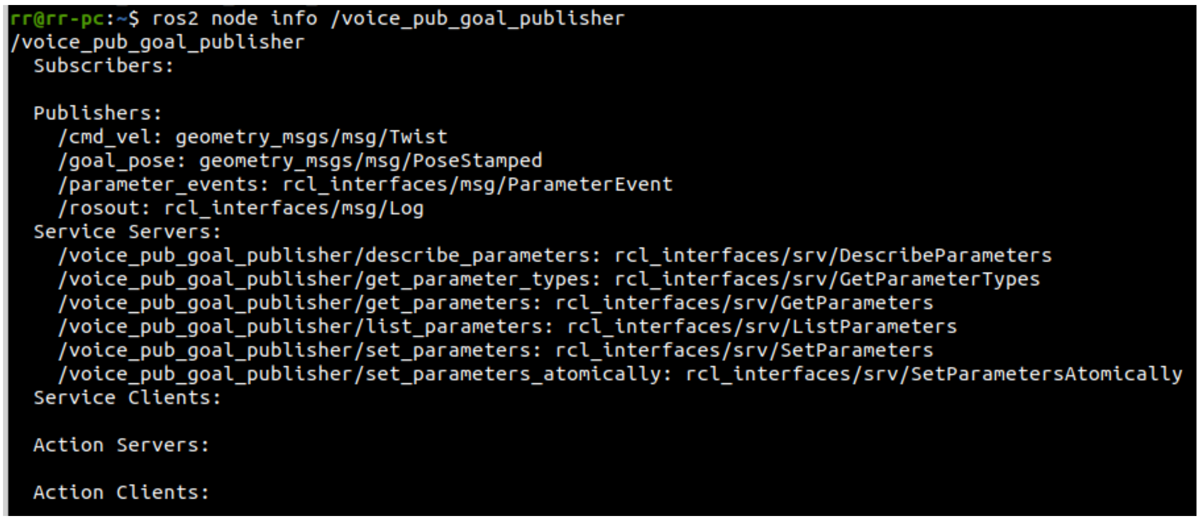
6.4. Node analysis

6.4.1. Display calculation graph

```
rqt_graph
```



6.4.2. Voice control node details



```
rr@rr-pc:~$ ros2 node info /voice_pub_goal_publisher
/voice_pub_goal_publisher
  Subscribers:

  Publishers:
    /cmd_vel: geometry_msgs/msg/Twist
    /goal_pose: geometry_msgs/msg/PoseStamped
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /voice_pub_goal_publisher/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /voice_pub_goal_publisher/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /voice_pub_goal_publisher/get_parameters: rcl_interfaces/srv/GetParameters
    /voice_pub_goal_publisher/list_parameters: rcl_interfaces/srv/ListParameters
    /voice_pub_goal_publisher/set_parameters: rcl_interfaces/srv/SetParameters
    /voice_pub_goal_publisher/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
```