

6. Voice control multi-point navigation

6. Voice control multi-point navigation

6.1, Functional description

6.2. Preparation

6.2.1 Bind the voice control device ports in the host computer.

6.2.2 Mounting the voice control device in the docker container

6.3. Configure the navigation point

6.4. Using voice multi-point navigation

6.5 Node resolution

6.5.1. Displaying a computational graph

6.5.2. Voice control node details

The operating environment and hardware and software 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 (version not required) + docker (version 20.10.21 and above)
- PC virtual machine: Ubuntu (20.04) + ROS2 (Foxy)
- Usage scenario: use on a relatively clean 2D plane

6.1, Functional description

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

Note: Before using the functions in this subsection, please learn to use the module [----- map building navigation function in LiDAR series course];

6.2. Preparation

This lesson requires the use of a voice control device, and the following preparations need to be made before running this lesson:

6.2.1 Bind the voice control device ports in the host computer.

Please refer to this section [1. Introduction to the module and the use of port binding].

6.2.2 Mounting the voice control device in the docker container

When you enter the docker container, you need to modify the script that enters the container to mount the voice control device:

Add this line to the [run_docker.sh] script:

```
--device=/dev/myspeech \           # 添加这行  
                                # Add this line
```

Other modifications as you see fit:

```
#!/bin/bash  
xhost +  
  
docker run -it \  
--net=host \  
--env="DISPLAY" \  
--env="QT_X11_NO_MITSHM=1" \  
-v /tmp/.X11-unix:/tmp/.X11-unix \  
-v /home/jetson/temp:/root/yahboomcar_ros2_ws/temp \  
-v /home/jetson/rosboard:/root/rosboard \  
-v /home/jetson/maps:/root/maps \  
-v /dev/bus/usb/001/010:/dev/bus/usb/001/010 \  
-v /dev/bus/usb/001/011:/dev/bus/usb/001/011 \  
--device=/dev/astradepth \  
--device=/dev/astrauvc \  
--device=/dev/video0 \  
--device=/dev/myserial \  
--device=/dev/rplidar \  
--device=/dev/myspeech \           # 添加这行  
                                # Add this line  
  
--device=/dev/input \  
-p 9090:9090 \  
-p 8888:8888 \  
yahboomtechnology/ros-foxy:3.5.4 /bin/bash
```

6.3. Configure the navigation point

1, enter the container, see [docker course in ----- 5, enter the docker container of the robot], sub-terminal execution of the following command:

```
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

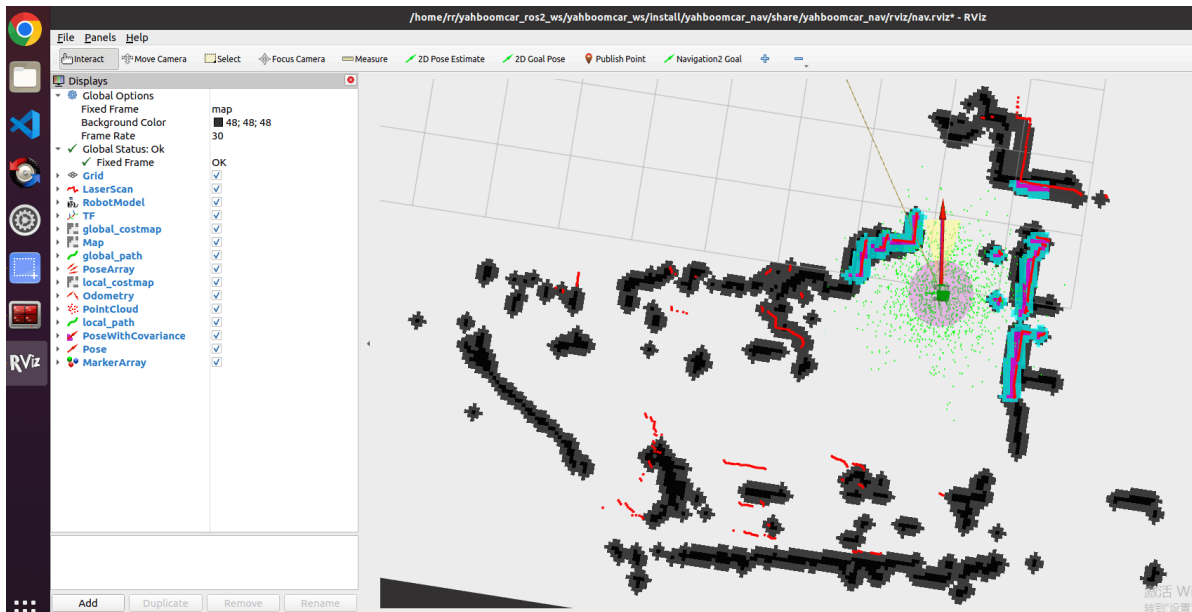
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

3. execution of navigation nodes in docker containers

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

4, this time in the virtual machine rviz interface click [2D Pose Estimate], and then compared to the position of the car in the map to the car mark an initial position;

After marking the display is as follows:



5. Compare the overlap between the radar scanning point and the obstacle, you can set the initial position for the cart several times until the radar scanning point and the obstacle roughly overlap;

6, open another terminal into the docker container, execute the

```
ros2 topic echo /goal_pose # 监听 /goal_pose话题  
# Listening to /goal_pose topics
```

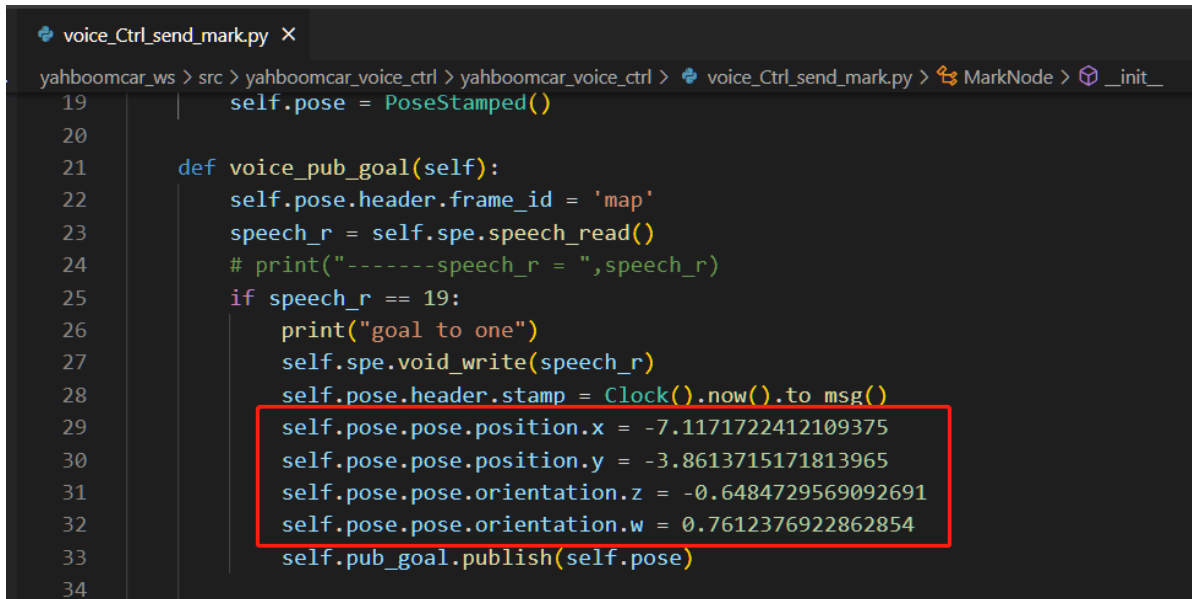
7, click [2D Goal Pose], set the first navigation target point, this time the cart began to navigate, while 6 steps in the topic data will be received listening:

```
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 in the following location:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctrl/voice_Ctrl_send_mark.py
```

Modify the positional attitude of the first navigation point to that printed in step 7:



```
voice_Ctrl_send_mark.py X
yahboomcar_ws > src > yahboomcar_voice_ctrl > yahboomcar_voice_ctrl > voice_Ctrl_send_mark.py > MarkNode > _init_
19     self.pose = PoseStamped()
20
21     def voice_pub_goal(self):
22         self.pose.header.frame_id = 'map'
23         speech_r = self.spe.speech_read()
24         # print("-----speech_r = ", speech_r)
25         if speech_r == 19:
26             print("goal to one")
27             self.spe.void_write(speech_r)
28             self.pose.header.stamp = Clock().now().to_msg()
29             self.pose.pose.position.x = -7.1171722412109375
30             self.pose.pose.position.y = -3.8613715171813965
31             self.pose.pose.orientation.z = -0.6484729569092691
32             self.pose.pose.orientation.w = 0.7612376922862854
33             self.pub_goal.publish(self.pose)
34
```

9. Modify the position of the other 4 navigation points in the same way.

6.4. Using voice multi-point navigation

1. Enter the container, see [5. Enter the docker container of the robot], and execute the following command in a sub-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

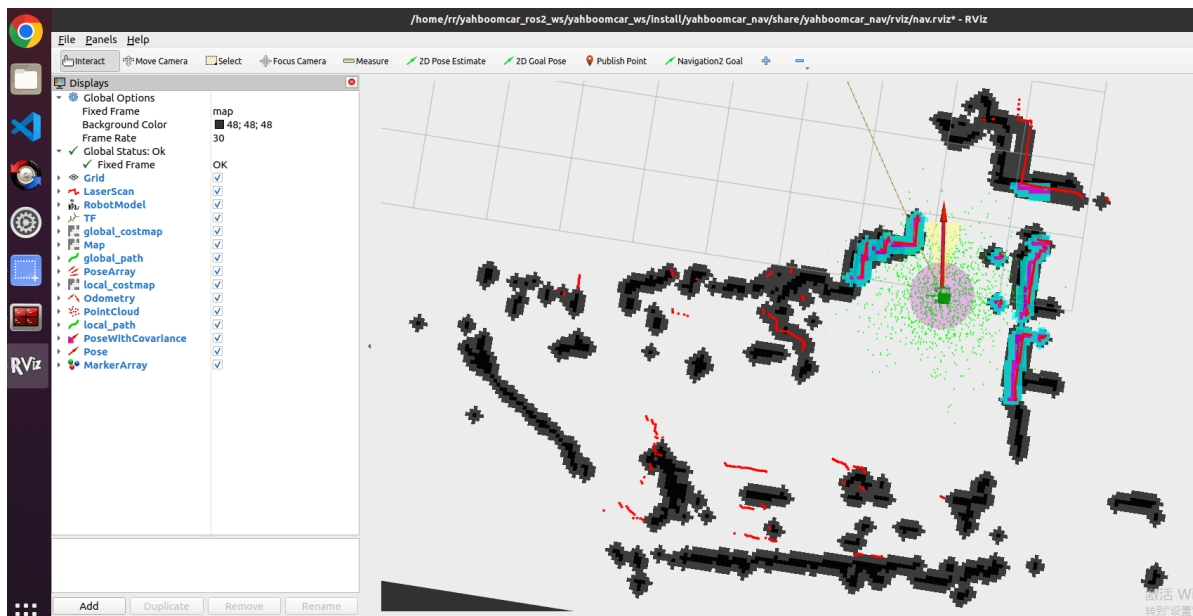
```
ros2 launch yahboomcar_nav display_nav_launch.py
```

3. execution of navigation nodes in docker containers

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

4, this time in the virtual machine rviz interface click [2D Pose Estimate], and then compared to the position of the car in the map to the car mark an initial position;

After marking the display is as follows:



5. Compare the overlap between the radar scanning point and the obstacle, you can set the initial position for the cart several times until the radar scanning point and the obstacle roughly overlap;

6, open another terminal into the docker container, the execution of the opening of the voice control navigation node

```
ros2 run yahboomcar_voice_ctrl voice_ctrl_send_mark
```

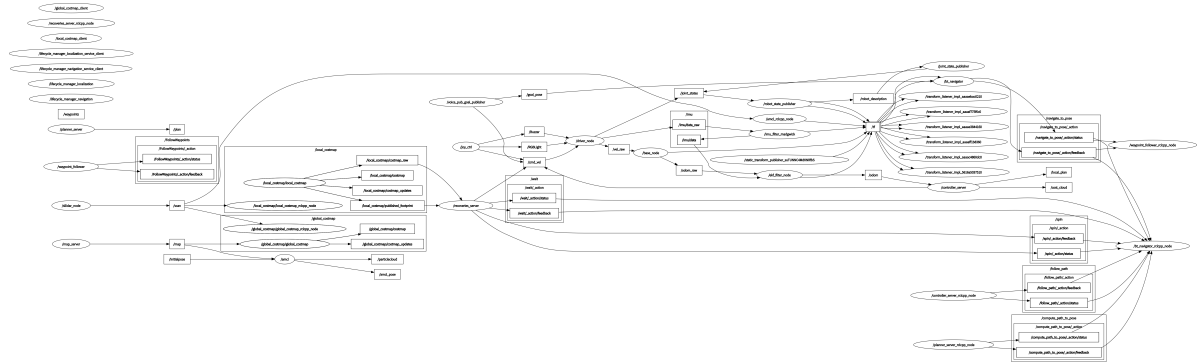
7, to the voice module on the car said "Hello, Xiaoya" wake up the voice module, hear the voice module feedback broadcast "in the", continue to say "navigation to the first position"; voice module will feedback broadcast "good, is going to the first position", at the same time the car began to navigate to the first position. After hearing the feedback from the voice module "yes", continue to say "navigate to position 1"; the voice module will announce "OK, going to position 1", and at the same time, the cart will start to navigate to position 1. Other position navigation can be used in the same way. Refer to the following table for voice control function words:

function word	Speech Recognition Module Result	Contents of the voice announcement
Navigate to position one.	19	Okay. Going to one.
Navigate to position two.	20	Okay. Going to two.
Navigate to position three.	21	Okay. Going to three.
Navigate to position four.	32	Okay. Going to four.
return to square one	33	Okay, it's coming back around.

6.5 Node resolution

6.5.1. Displaying a computational graph

rqt_graph



6.5.2. 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:
```