

6.Control multi-point navigation

6.Control multi-point navigation

- 6.1. Function description
- 6.2. Preparation
 - 6.2.1. Bind the voice control device port in the host machine
 - 6.2.2. Mount voice control device to docker container
 - 6.2.3. Configuration before use
- 6.3. Configure navigation points
- 6.4. Use voice multi-point navigation
- 6.5. Node analysis
 - 6.5.1. Display calculation graph
 - 6.5.2. Voice control node details

6.1. Function description

Through the speech recognition module, multi-point navigation can be realized through voice control in the established map;

6.2. Preparation

This course requires the use of voice control equipment. Before running this course, you need to make the following preparations:

6.2.1. Bind the voice control device port in the host machine

Please refer to this chapter [1. Module Introduction and Port Binding Usage] for operation

6.2.2. Mount voice control device to docker container

When entering the docker container, you need to modify the script to enter the container and mount the voice control device in it:

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

```
--device=/dev/myspeech \           # add this line
```

Others can be modified according to your own situation:

```
#!/bin/bash
xhost +

docker run -it \
--net=host \
--env="DISPLAY" \
--env="QT_X11_NO_MITSHM=1" \
```

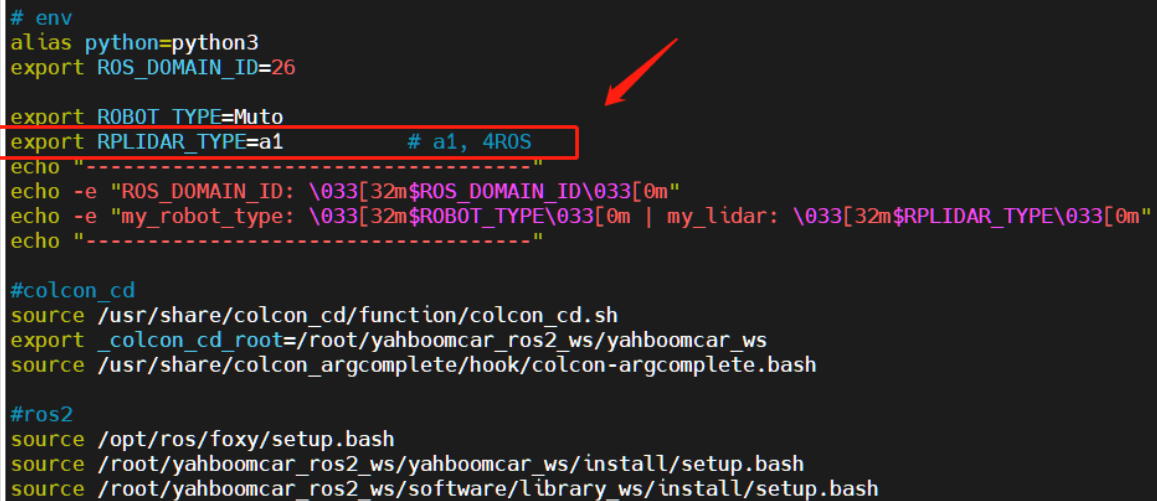
```
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /home/jetson/temp:/root/yahboomcar_ros2_ws/temp \
-v /home/jetson/rosboard:/root/rosboard \
-v /home/jetson/maps:/root/maps \
-v /dev/bus/usb/001/010:/dev/bus/usb/001/010 \
-v /dev/bus/usb/001/011:/dev/bus/usb/001/011 \
--device=/dev/astradepth \
--device=/dev/astrauvc \
--device=/dev/video0 \
--device=/dev/myserial \
--device=/dev/rplidar \
--device=/dev/myspeech \
--device=/dev/input \
-p 9090:9090 \
-p 8888:8888 \
yahboomtechnology/ros-foxy-muto:3.0.0 /bin/bash
```

6.2.3. 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 lidar 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 lidar type.

6.3. Configure navigation points

1. Enter the container and execute the following command in a terminal:

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

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

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

3. Start the lidar odometer in the docker container

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

4. Execute navigation node in docker container

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

#The above method loads the yahboomcar map by default. If you need to load other maps, please use the following method to start:

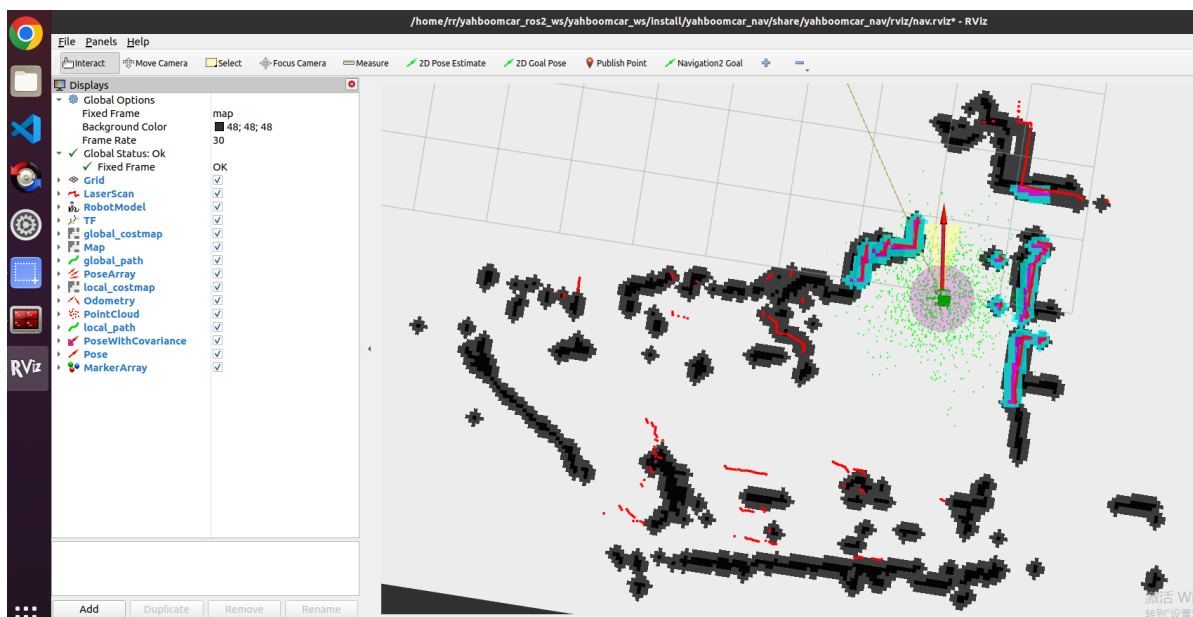
```
ros2 launch yahboomcar_nav navigation_teb_launch.py
```

```
map:=/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/test.yaml
```

#The path of the map above is the path of the map that you want to load during navigation. test.yaml is the name of the map, which means that the test map is loaded during navigation. If it is another name, it should be changed accordingly.

5. 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:



6. Compare the overlap between the lidar scanning point and the obstacle, and set the initial pose of the car multiple times until the lidar scanning point and the obstacle roughly

coincide;

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

```
ros2 topic echo /goal_pose # monitor /goal_pose topic
```

8. 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
```

9. Open the code at the following location:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctrl/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 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
```

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

6.4. Use voice multi-point navigation

1. Enter the container and execute the following command in a 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. Start the lidar odometer in the docker container

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

4. Execute navigation node in docker container

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

#The above method loads the yahboomcar map by default. If you need to load other maps, please use the following method to start:

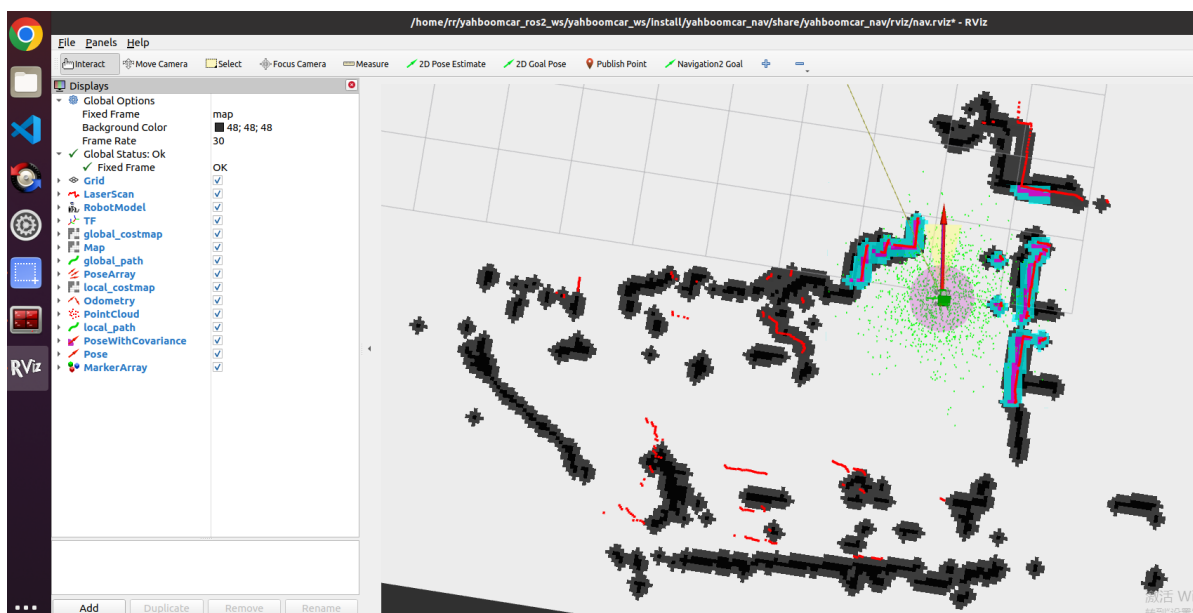
```
ros2 launch yahboomcar_nav navigation_teb_launch.py
```

```
map:=/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/test.yaml
```

#The path of the map above is the path of the map that you want to load during navigation. test.yaml is the name of the map, which means that the test map is loaded during navigation. If it is another name, it should be changed accordingly.

5. 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:



6. Compare the overlap between the lidar scanning point and the obstacle, and set the initial pose of the car multiple times until the lidar scanning point and the obstacle roughly coincide;
7. 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
```

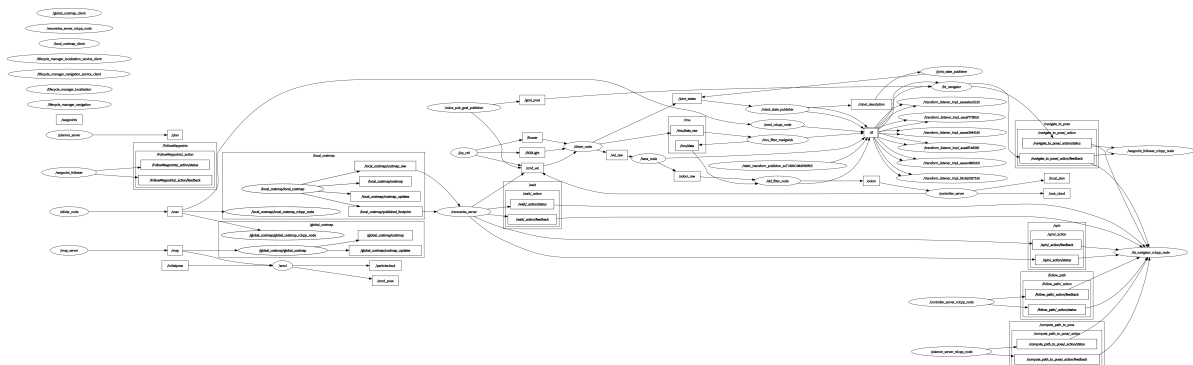
8. Say "Hello, yahboom" to the voice module on the car to wake up the voice module. After hearing the voice module's feedback "at", continue to say "Navigate to No. 1"; The voice module will feedback and broadcast "Okay, going to No. 1", and at the same time, the car will start to navigate to No. 1. Navigation in other locations can be used in the same way. Please refer to the following table for voice control function words :

Function words	Speech recognition module results	Voice broadcast content
Go to the point A	19	OK, I'm going to the point A.
Go to the point B	20	OK, I'm going to the point B.
Go to the point C	21	OK, I'm going to the point C.
Go to the point D	32	OK, I'm going to the point D.
back to original	33	OK, I'm return back.

6.5. Node analysis

6.5.1. Display calculation 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:
```