6. Voice control multi-point navigation

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

- Reference model：ROSMASTER X3
- Robot hardware configuration: Arm series master, Slan A1 LiDAR, AstraPro Plus depth camera
- Robot system: Ubuntu (version not required) + docker (version 20.10.21 and above)
- PC VM: Ubuntu (20.04) + ROS2 (Foxy)
- Usage scenario: Used on a relatively clean 2D plane

6.1. Description

On the established map, voice control ROSMASTER to navigate to point1, point2, point3. The R2 key on the handle can stop/start this function at any time.

6.2. Steps

6.2.1. Configuration navigation point

1. Enter the docker，run the following command on the termina

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

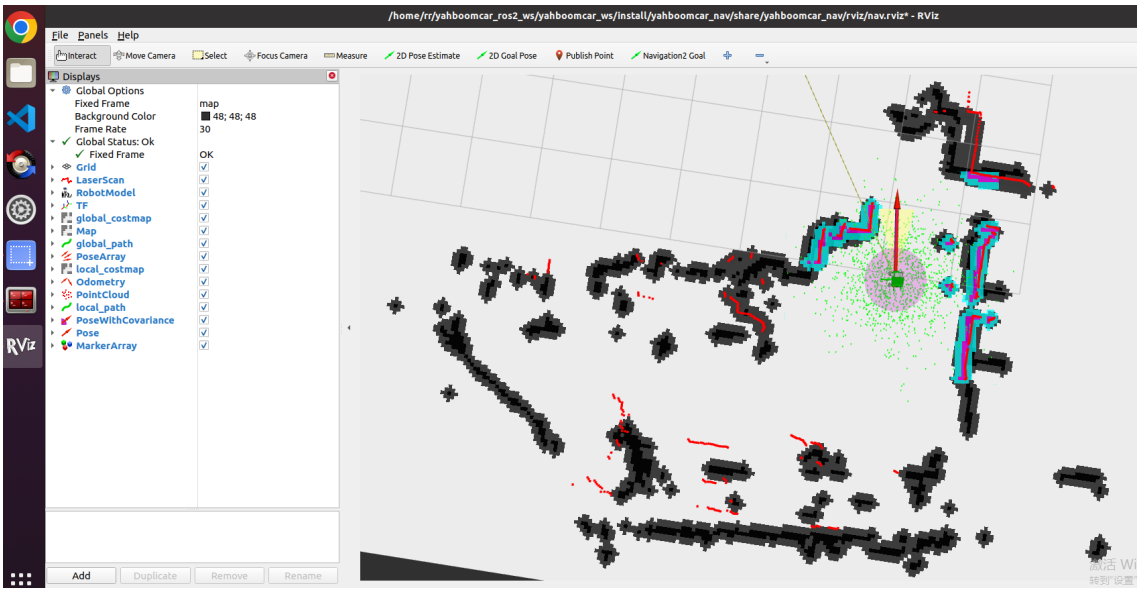
2. Open VM, configure multiple machine communication, then run the following command to display the RIVZ node

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

3. Execute navigation nodes in docker

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

4. Click [2D Pose Estimate] on the rviz screen of the VM, compare the posture of the robot to mark the initial posture of the robot on the map



5. Compare the overlap between radar scan points and obstacles, the initial posture of the robot can be set several times until the radar scanning point and the obstacle roughly coincide

6. Open another terminal into the docker, then run the following command

```
ros2 topic echo /goal_pose   # display /goal_pose topic
```

7. Click [2D Goal Pose], set the first navigation target point, at which point the car begins to navigate. In addition, the topic data is 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 in the following position

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

   Change 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. Modify the pose of the other 4 navigation points in the same way

6.2.2. Use voice multi-point navigation

1. Enter the docker, run the following command on the terminal

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

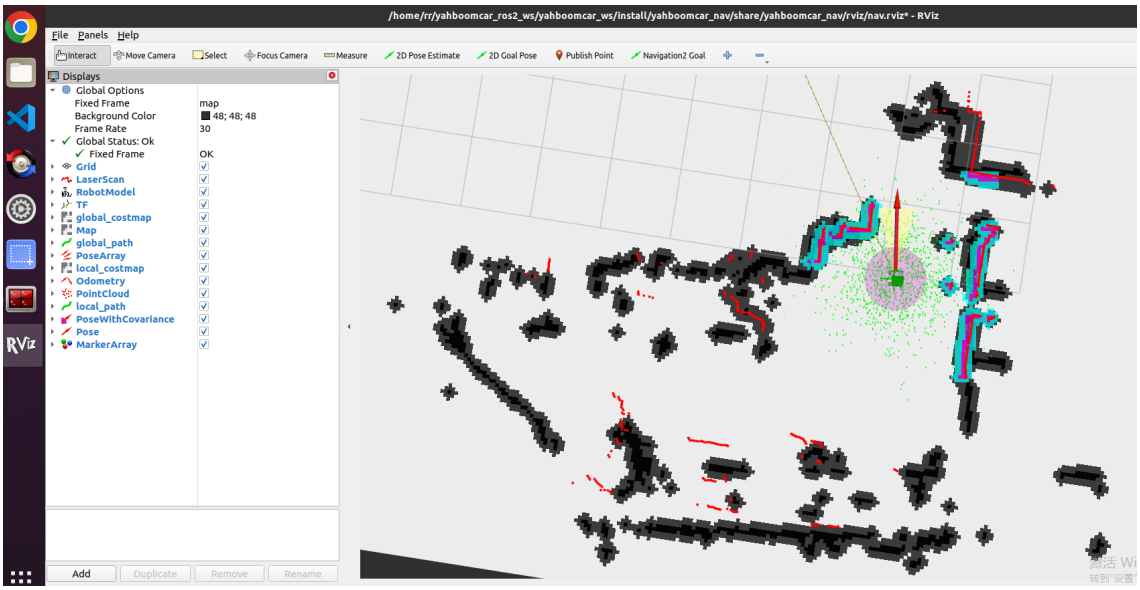
2. Open VM, configure multiple machine communication, then run the following command to display the RIVZ node

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

3. Execute navigation nodes in docker

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

4. Click [2D Pose Estimate] on the rviz screen of the VM, compare the posture of the robot to mark the initial posture of the robot on the map



5. Compare the overlap between radar scan points and obstacles, the initial posture of the robot can be set several times until the radar scanning point and the obstacle roughly coincide

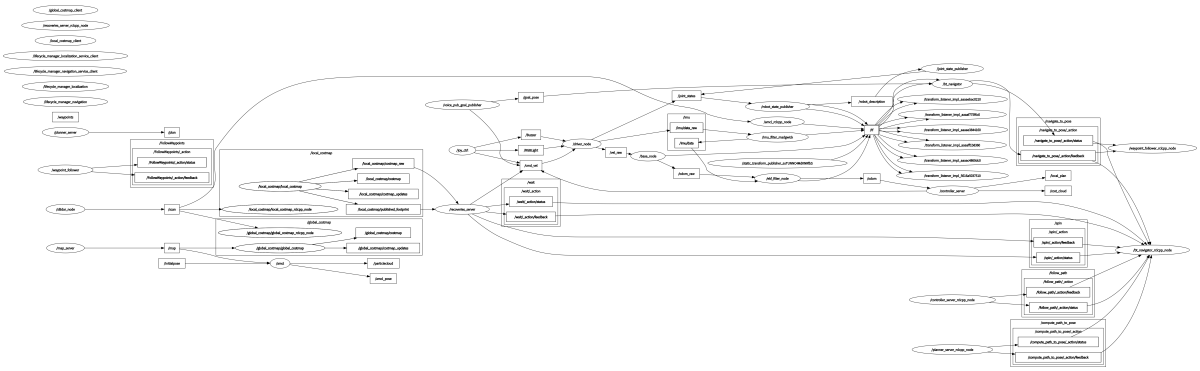
6. Open another terminal into the docker, then run the voice control navigation node

```
ros2 run yahboomcar_voice_ctrl voice_Ctrl_send_mark
```

7. After calibrating the initial pose in rviz, we can say "Hi Yahboom" to wake up the voice module, until it replies "Hi, i'm here", indicating that the module has been woken up.
8. We say "Go to the point A", it will reply "OK, I'm going to the point A."

6.2.3. Topic graph

```
rqt_graph
```



6.2.4. 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:
```

6.3. Voice module communication protocol

You need to configuration navigation point then use voice control

function word	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.

function word	Speech Recognition Module Results	Voice broadcast content
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.
Return to the original place	33	OK, I'm return back.