4. Voice control robotic arm grip objects

4.1 Function description

Through the interaction with the voice module, the rosmaster can be navigated to point A/B/C/D/origin position, any of the five positions. After reaching the destination, the robotic arm on the rosmaster can also be used to grab the handed object, and then Then command it to navigate to point A/B/C/D/origin position, any of the five positions, and put the object down, waiting for the next command.

Voice control allows rosmaster to go to any of the five positions of A/B/C/D/origin point. After reaching the destination, you can also let the robotic arm grab the object, and then let rosmaster go to any of the five positions of A/B/C/D/origin point, and put the object down, waiting for the next command.

4.2 Steps

4.2.1 ROS package path

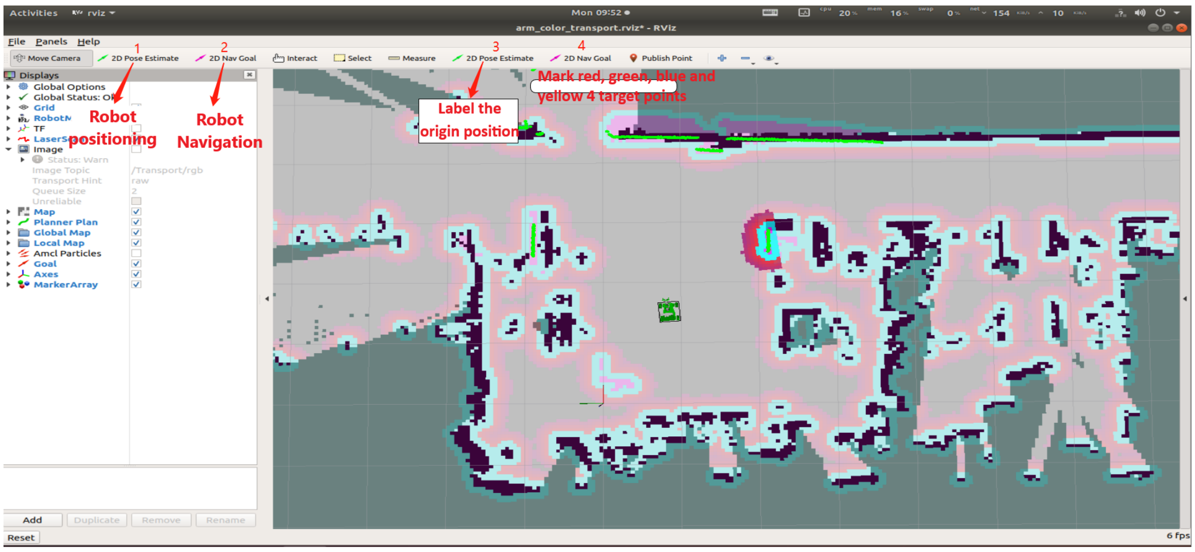
```
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/
```

4.2.2 start

Note: Multi-machine communication needs to be implemented between rosmaster X3Plus and the virtual machine, you can view the "6.Linux operating system\6.4 Multi-machine communication configuration tutorial for configuration.

```
#rosmaster X3Plus running
roslaunch yahboomcar_voice_ctrl voice_transport_base.launch
python3
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/scripts/voice_ctrl_takethings.py
# virtual machine running
roslaunch arm_color_transport transport_rviz.launch
```

Each tool annotation on the virtual machine rviz, as shown below.



- Step 1

In rviz, use the Tool-1 to calibrate and position the rosmaster.

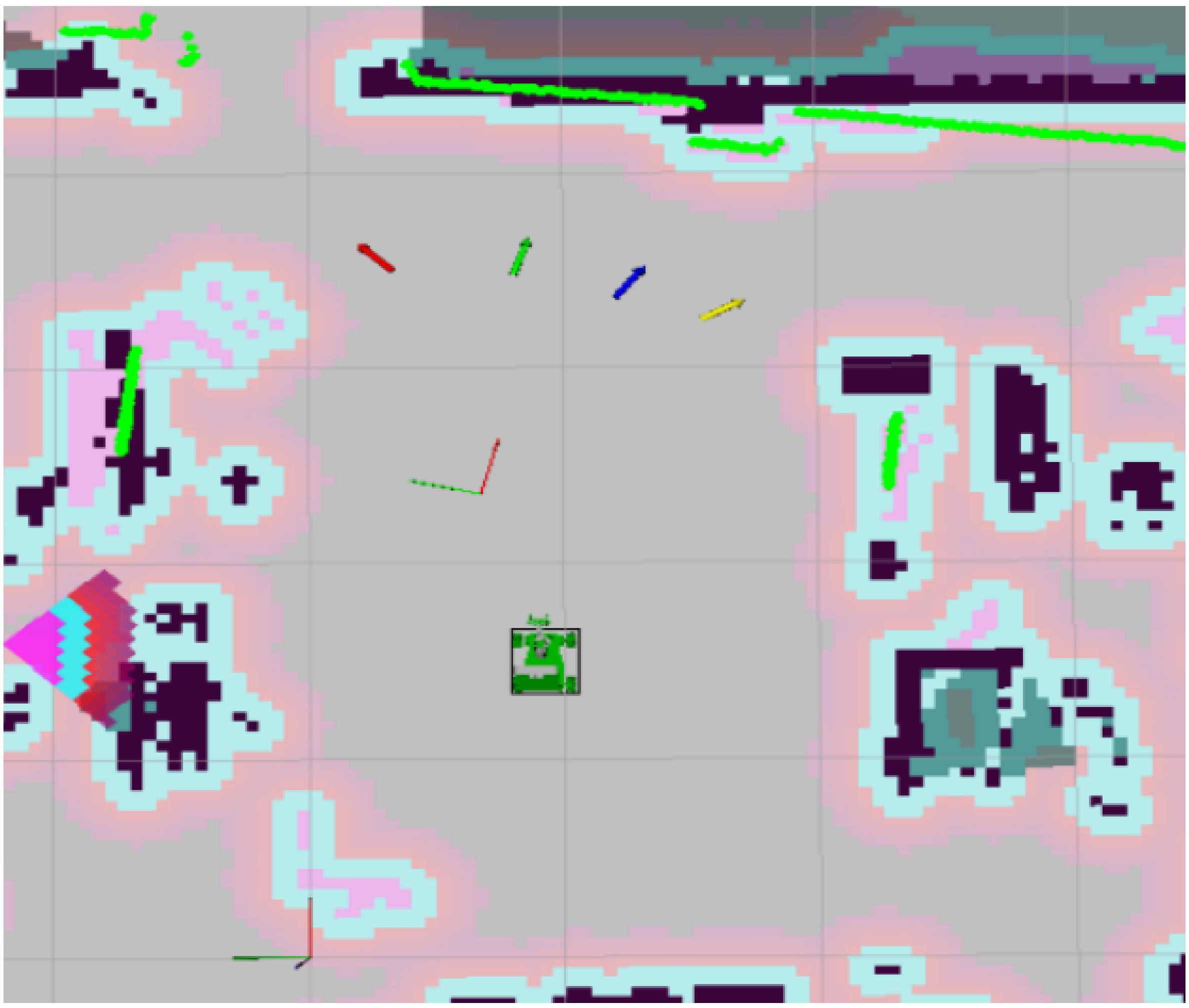
- Step 2

  In rviz, use Tool-3 to label the origin position for rosmaster. (This origin position is the position that the car will automatically return to after putting down the color blocks).

- Step 3

  In rviz, use Tool-4 to mark the red, green, blue and yellow destinations on the map in turn. After the rosmaster's robotic arm grips the color block, it will navigate to the destination corresponding to the color block.

  Note: We need four points to be marked in a row, and the distance between the front and rear cannot be too far apart, otherwise, when ROSMASTER plans the path, it may hit other color blocks.



Note: **point A** corresponds to **red** marker point; **point B** corresponds to **green** marker point; **point C** corresponds to **blue** marker point; **point D** corresponds to **yellow** marker point;

- Step 4 (Go to point A and then take the item and place it in point B)

  1. We need to say "Hi, Yahboom", the voice module will reply "Hi, i'm here"
  2. Then we can say "Go to the point A", the voice module will say "OK, I'm going to the point A " and robot will autonomously go to point A.
  3. After reaching the destination, the buzzer will whistle once. We can say "Clip the block", after the module receives our command, the buzzer will whistle once, indicating that the command is received.
  4. Then, we need give color bloc, to the gripper, the buzzer will whistle once, the gripper will pick up the color block, and voice module will reply "OK,let me clip them".

4.2.3、 Code voice_ctrl_takethings.py

- Code path

```
~/yahboomcar/src/yahboomcar_voice_ctrl/scripts
```

- Core code analysis

1） important mode status

```
self.model== "Init"：initialized state
self.model == "Grip_Target"：grip state
self.model == "next_points"：Navigate to the next target point state
self.model == "Grip_down"：Put down the object state
```

2）important flag flag bit

```
self.clip:has clipped the object, True means the object has been clipped,
Fasle means the object has not been clipped
self.come_back_flag: Whether to return to the origin position flag, True is
to go back, False is not to go back
self.ros_nav.goal_result:Whether to reach the navigation target point, 0 is
not reached, 3 is reached
```

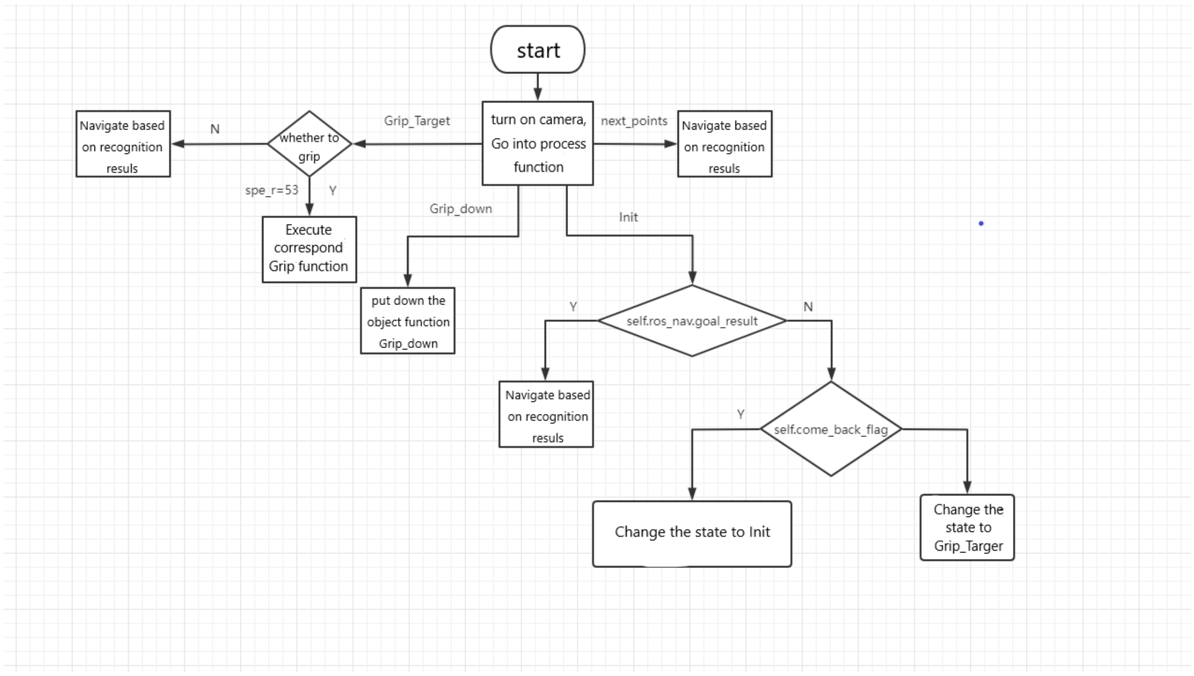
3）important functions

```
next_points：recognize voice, navigate to the next target point
comeback：return to origin position
Grip_down：Put down the object
Grip_Target：recognize voice, grip objects or navigate to the next target
point
```

- Program Description

  In the main function, directly enter the process function, judge the mode state and the value of the flag in the process function, and then execute the corresponding function or code.

4.3 Flow chart


start
turn on camera, Go into process function
Grip_Target
next_points
Navigate based on recognition resuls
whether to grip
N
spe_r=53
Y
Navigate based on recognition resuls
Execute correspond Grip function
Grip_down
Init
put down the object function Grip_down
Y
self.ros_nav.goal_result
N
Navigate based on recognition resuls
Y
self.come_back_flag
Change the state to Init
Change the state to Grip_Targer