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.

4.2 start

4.2.1 ROS package path

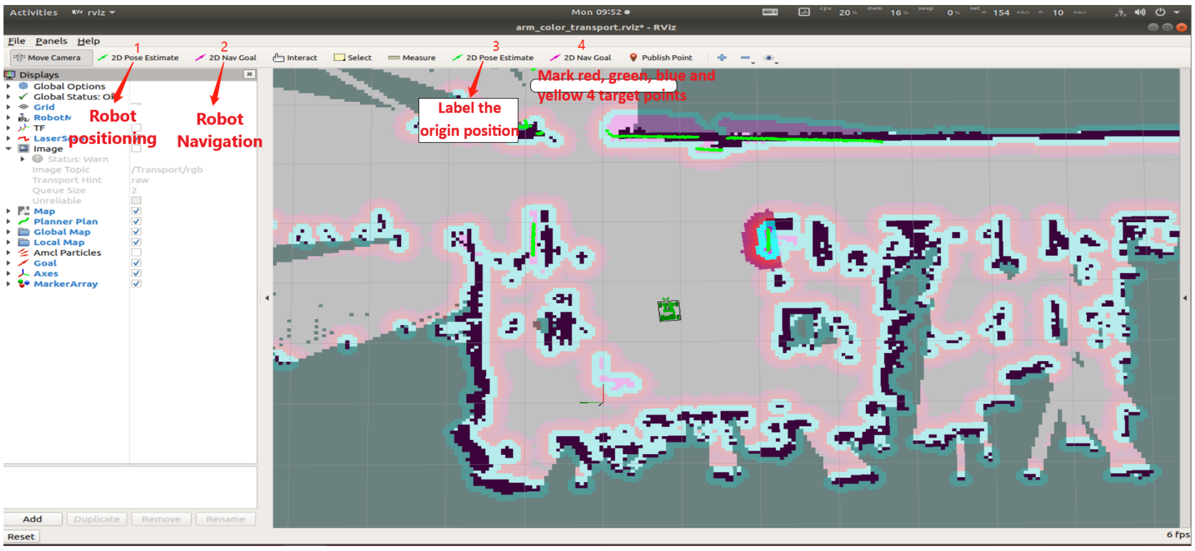
```
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/
```

4.2.2 start

Note: Multi-machine communication needs to be implemented between rosmaster X3Plus and the virtual machine , you can view the " **06.Linux operating system\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 is shown in the following figure,



- Step 1

  In rviz, use the 1 tool to calibrate and position the rosmaster

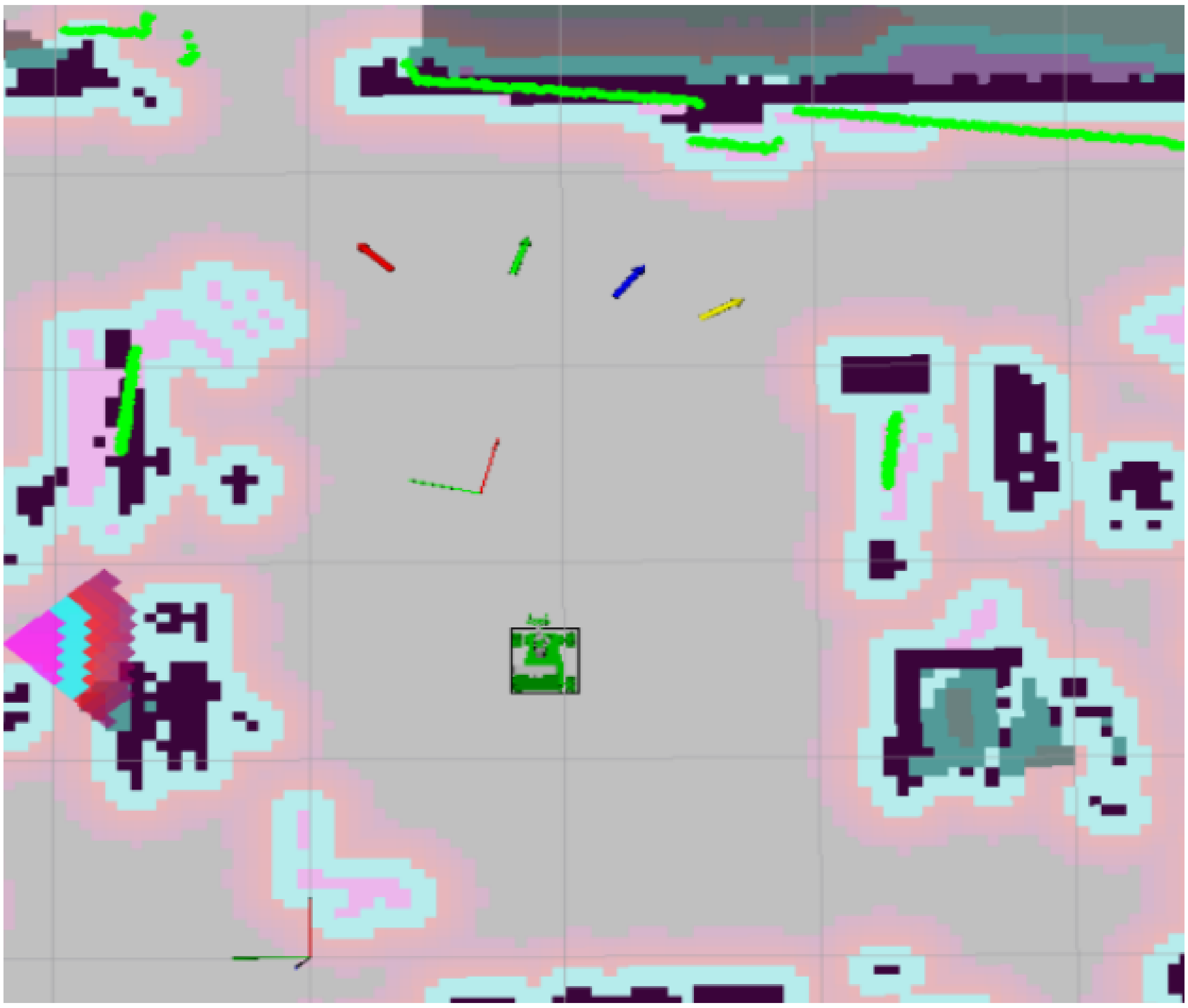
- Step 2

  In rviz, use 3 tools to label the start position for rosmaster, the start position is the position that the car will automatically return to after putting down the color blocks

- Step 3

  In rviz, use 4 tools 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.

  Here it is best to put four The points are marked in a row,and the distance between the front and rear cannot be too far apart. Otherwise, when planning the path, you may hit the 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 (take point A first and then take the item and place it in point B as an example)

  Say "`Hi Yahboom`" to the rosmaster, wake up the voice module, then say "`Go to the point A`" to it, after moving back to adjust the position, the voice module will say "`OK, I'm going to the point A`" and autonomously navigate to point A; after reaching the destination, the buzzer will sound "di", and then to it, "`Clip the block`"; after the module hears it, the buzzer will sound "di" One sound to indicate that the command is received, then hand the object to the gripper, after a second "di" the gripper will grab the object and will say "`I grabbed it just now`"; then, say to it "`Go to the point B`", after the car adjusts its posture backwards, it sends out "`Go to the point B`" and autonomously navigates to point B; when it reaches point B, the buzzer will make a "di" sound, indicating that Reach the destination, then grab down, put the thing down and say "`It has been in its place`".

4.2.3、 Core 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
  ```

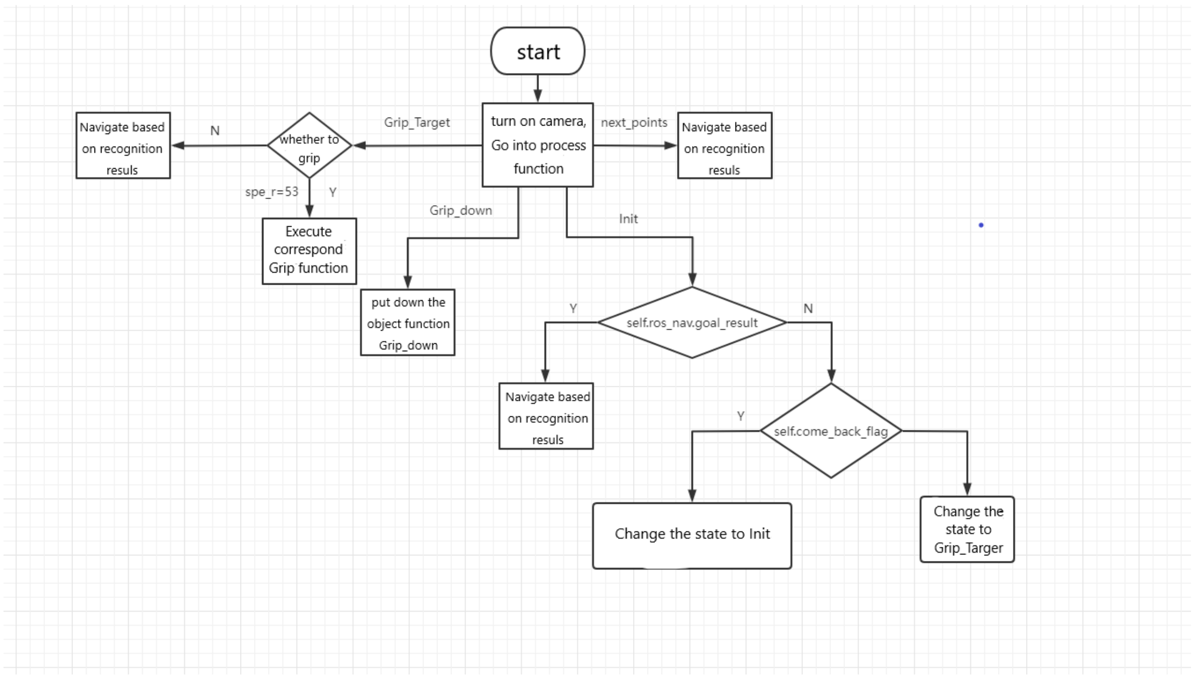
  3）important functions

  ```
  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, in the process function, judge the mode state and the value of the flag, and then execute the corresponding function or program.

4.3 program flow chart


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