3.Voice control color block transport

3.1 Function description

Through the interaction with the voice module, the camera can recognize the color of the building block, and after accurate identification, it can be clamped and autonomously navigate to the point marked on the map, put down the building block, and finally autonomously navigate back to the start positio for the next recognition.

3.2 start

3.2.1 ROS package path

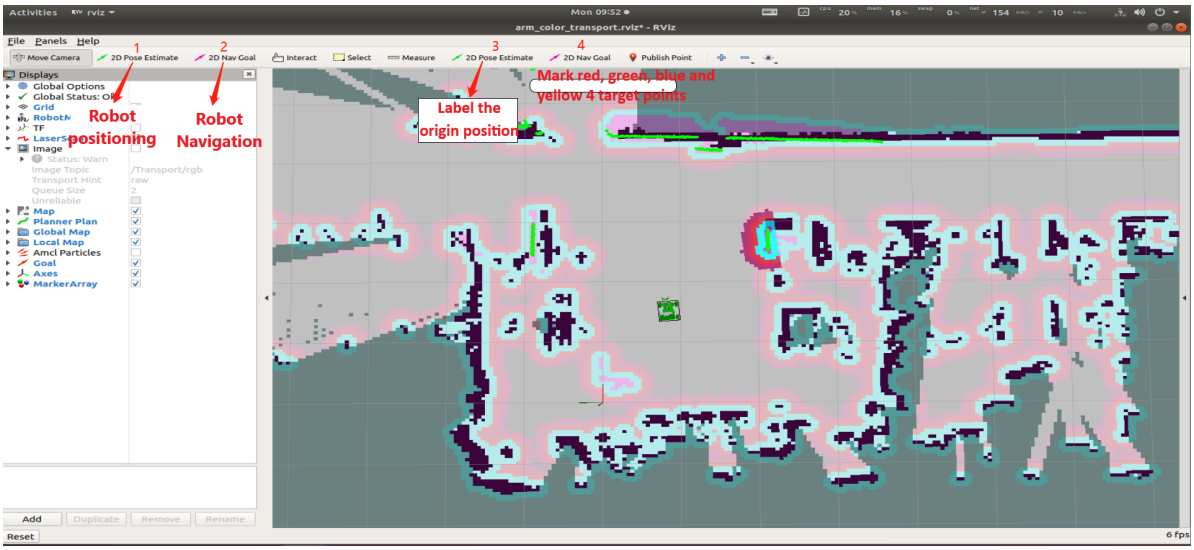
```
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/
```

3.2.2 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_color_transport.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

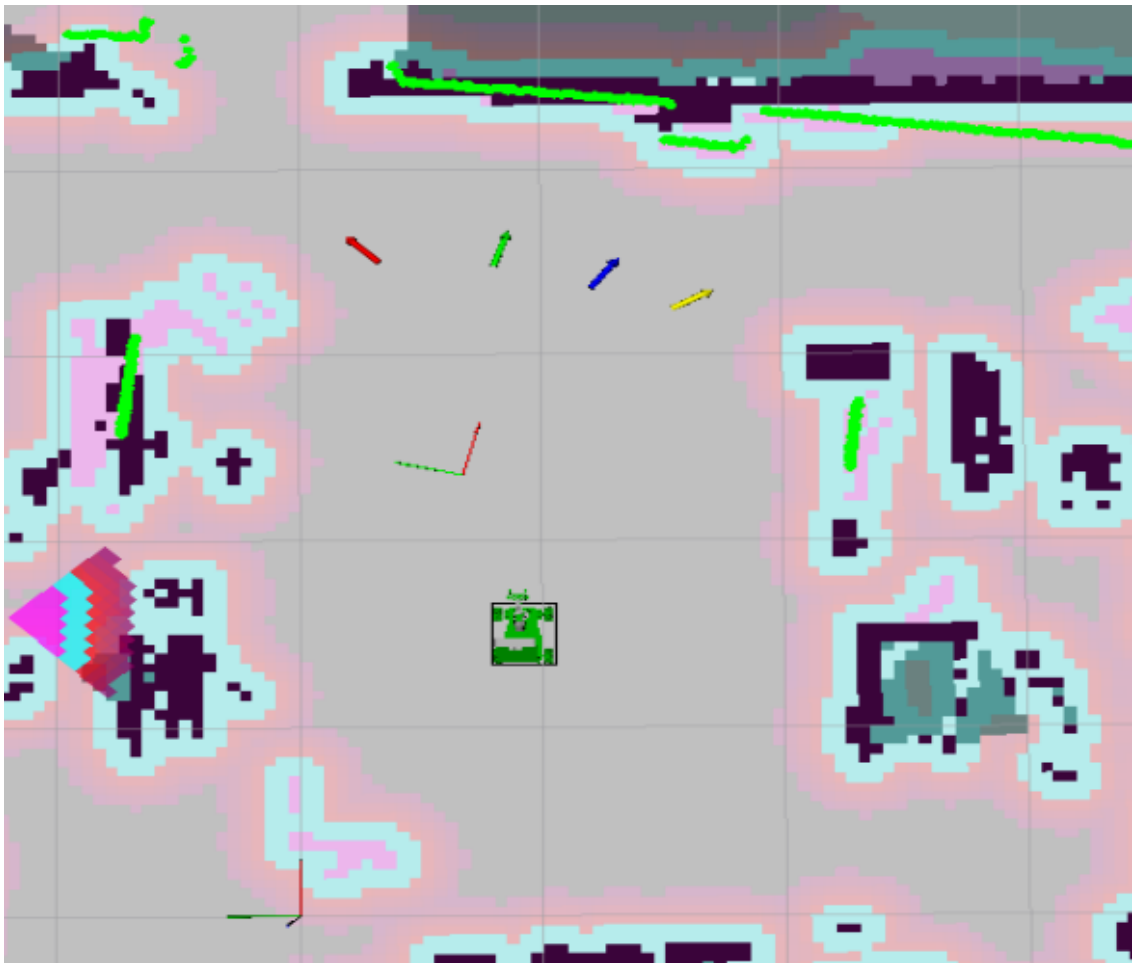
  In rviz, use the 1 tool to calibrate and position the rosmaster

- Step 2

  In rviz, use 3 tools to label the origin position for rosmaster, the origin 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.



- Step 4

  Say to rosmaster `“Hi Yahboom”`, after waking up the voice module, then say to it `“Sort color block”`, and then place the color block on the camera on the robotic arm, about 5-7cm, after the camera recognizes the color block, it will make a "di" sound, and the recognition result will be broadcast; put The building block is handed to the gripper, and after a "di" sound, the gripper will clamp the building block; then it will navigate to the position of the corresponding color, put down the building block, and finally return to the marked origin position.

3.2.3 launch file analysis

voice_transport_base.launch

```
    <!--  Load map -->
    <node name="map_server" pkg="map_server" type="map_server" args="$(find
yahboomcar_nav)/maps/$(arg map).yaml"/>
    <include file="$(find yahboomcar_nav)/launch/laser_bringup.launch"/>
    <!-- Adaptive Monte Carlo Localization -->
    <include file="$(find yahboomcar_nav)/launch/library/amcl.launch"/>
    <!-- Navigation Core Components move_base -->
    <include file="$(find yahboomcar_nav)/launch/library/move_base.launch"/>
    <!-- web_video_server -->
    <node pkg="web_video_server" type="web_video_server" name="web_video_server"
output="screen"/>
    <node pkg="arm_color_transport" type="DrawMarker.py" name="draw_marker"
required="true" output="screen"/>
```

3.2.4 Core code voice_color_transport.py

- Code path

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

- Core code analysis

  1) import the corresponding library file

  ```
  from transport_common import ROSNav # Navigation library
  from Speech_Lib import Speech #Speech Recognition Library
  ```

  2) important status

  ```
  self.model = "Grip": When "Sort color block" is recognized, the init state
  changes to the ready to grip state

  self.model = "Grip_Target": grip state, after the grip state is executed, it
  becomes the Transport state

  self.model = "Transport": the state of transport color block, it will become
  Grip_down state after reaching the destination

  self.model = "Grip_down": Put down the block state, after the action is
  completed, it will enter the comeback function, and then the state will
  become come_back

  self.model = "come_back":Return to the state of origin position, after
  returning, reset all data and state
  ```

  3) important function

  ```
  get_color: identify colors, and increase the accuracy of color recognition by
  adjusting HSV
  Grip_Target: Grip Block function
  comeback: Back to origin position function
  Grip_down: Put down the block function
  buzzer_loop: Prompt, buzzer function
  ```

- Program Description

  In the main function, after the camera is turned on to obtain the camera data, the image data is passed to the process function, and then the mode status is judged in the process function, and then the relative function execution program is entered.

3.3 program flow chart