3.Voice control color block transport

3.1 Function description

Voice control robot recognizes the color block and picks it up, then transports it to the coordinate point marked on the map, puts down the color block, and then automatically returns to the far point for the next recognition.

3.2 Steps

3.2.1 ROS package path

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

3.2.2 Start

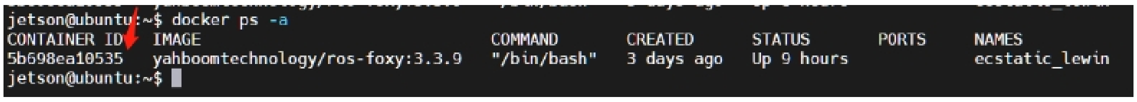
Note: Multi-machine communication needs to be implemented between rosmaster X3Plus and the virtual machine, you can view the "6.Linux operating system\6.4 Multi-machine communication configuration tutorial for configuration

```
#rosmaster X3Plus  running
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to 07.Docker-orin/05,
Enter the robot's docker container
~/run_docker.sh
roslaunch yahboomcar_voice_ctrl voice_transport_base.launch
```

<Open another terminal and enter the same docker container

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

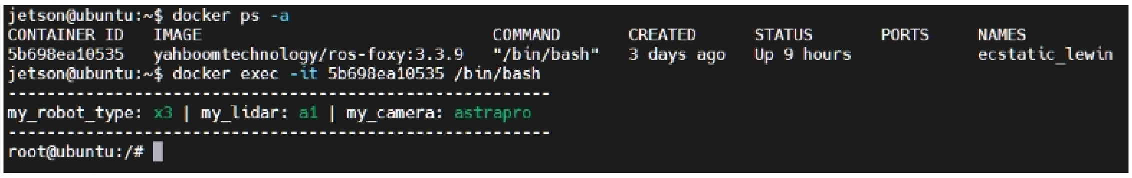
```
docker ps -a
```



```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS      PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9     "/bin/bash"   3 days ago   Up 9 hours            ecstatic_lewin
jetson@ubuntu:~$
```

2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```



```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS      PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9     "/bin/bash"   3 days ago   Up 9 hours            ecstatic_lewin
jetson@ubuntu:~$ docker exec -it 5b698ea10535 /bin/bash
-------------------------------------------------------
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
-------------------------------------------------------
root@ubuntu:/#
```

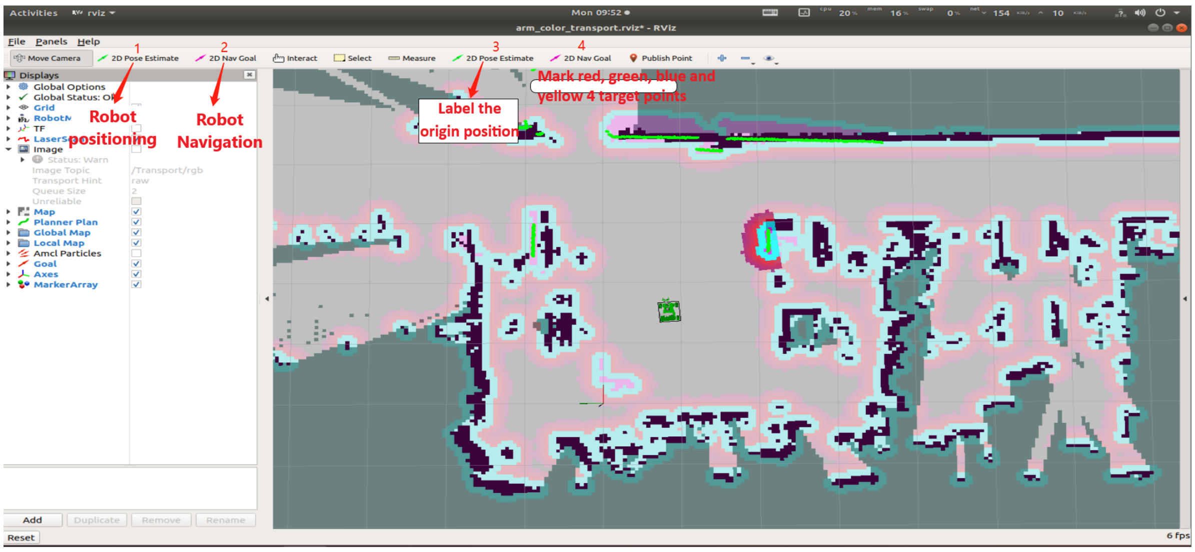
After successfully entering the container, you can open countless terminals to enter the container.

```
#rosmaster X3Plus  running
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, as shown below.


1
2
3
4
2D Pose Estimate
2D Nav Goal
Robot positioning
Robot Navigation
Label the origin position
Mark red, green, blue and yellow 4 target points

- step 1

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

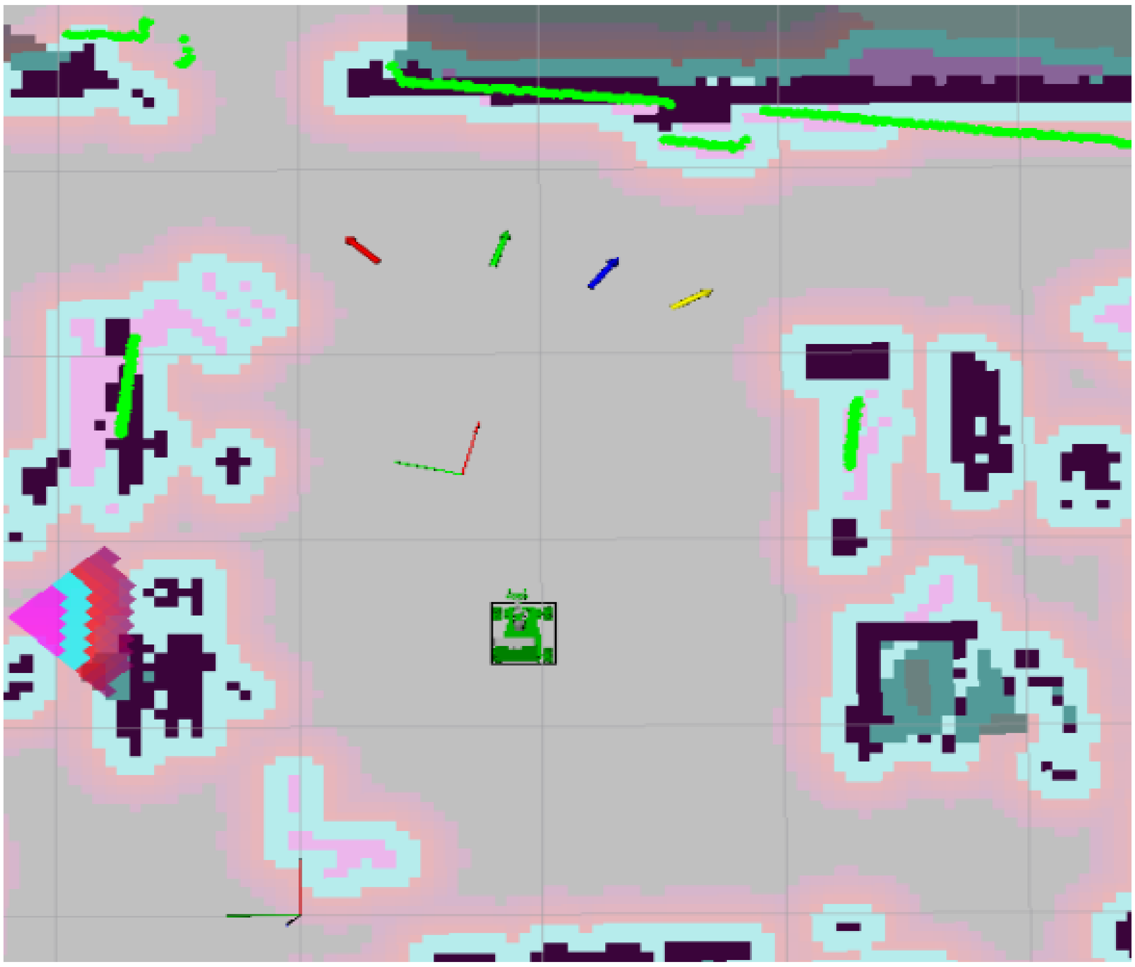
- Step 2

  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.



- Step 4

  1) We need to say "Hi, Yahboom", the voice module will reply "Hi, i'm here"

  2) Then we say "Sort color block", and then place the color block on the robotic arm.
   The camera is about 5-7cm in front of the camera. The camera recognizes that the color block is, and the buzzer will sound once.
  3) The voice module replies "This is yellow/green/blue/red".
  4) We hand the color block to the gripper, the buzzer will whistle once, and then the gripper will tighten the color block;
  5) The car will automatically navigate to the position of the corresponding color, put down the color block, and the voice module will reply "It has" been in its place".
  6) The trolley will return to the origin.

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 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
  ```

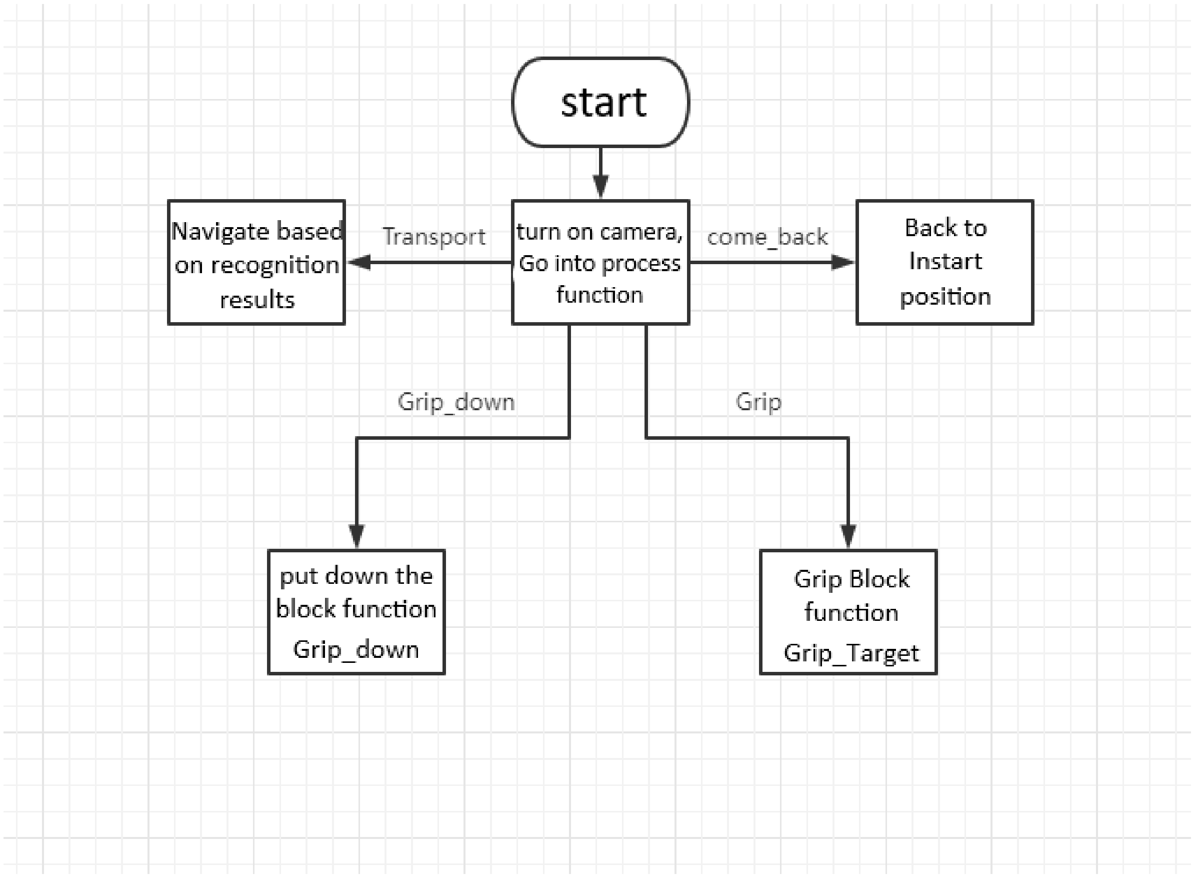
  3) important function

  ```
  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 is entered to execute the program.

3.3 Flow chart


start
turn on camera, Go into process function
Transport
Navigate based on recognition results
come_back
Back to Instart position
Grip_down
Grip
put down the block function Grip_down
Grip Block function Grip_Target