

3. Control Muto tracking

Before running this program, you need to bind the port number of the voice board and the port number of the ROS expansion board on the host machine. You can refer to the previous chapter for binding;

When entering the docker container, you need to mount the voice board to recognize the voice board in the docker container.

1. Program function description

After the program is started, say "Hello, Xiaoya" to the module, and the module's reply "in" means waking up the voice board, and then you can say to it any one of red line patrol, green line patrol, blue line patrol, or yellow line patrol. After receiving the instruction, the program recognizes the color, loads the processed image, and then presses the R2 key of the handle to start the program. Muto will start moving along the route identified by the color. During the autonomous driving process, if it encounters an obstacle, the buzzer will sound and the vehicle will stop.

2. Program code reference path

After entering the docker container, the source code of this function is located at

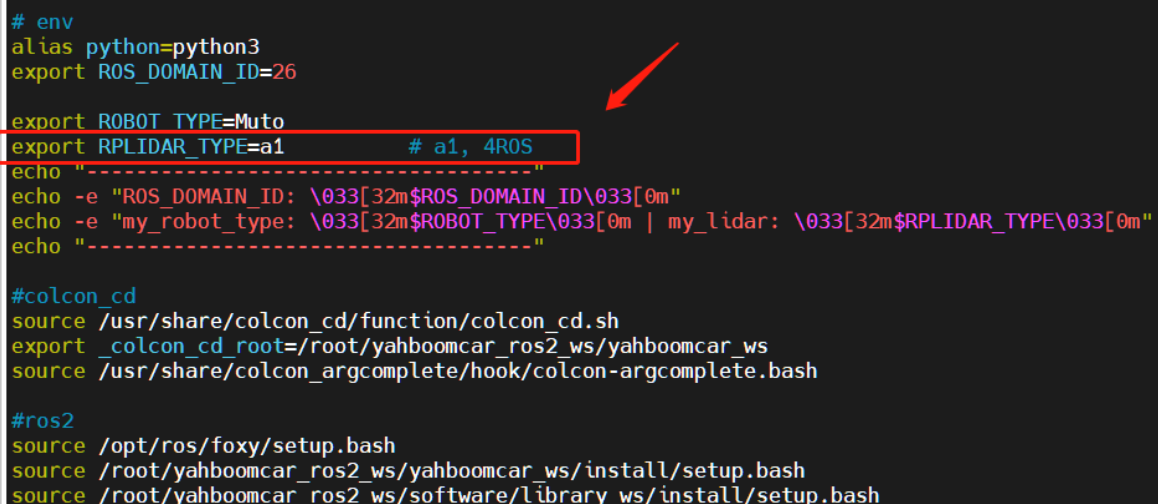
```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/launch/voice_ctrl_follow_line_launch.py
```

3. Configuration before use

Note: Since the Muto series robots are equipped with multiple radar devices, the factory system has been configured with routines for multiple devices. However, since the product cannot be automatically recognized, the radar model needs to be manually set.

After entering the container: Make the following modifications according to the lidar type:

```
root@ubuntu:/# cd
root@ubuntu:~# vim .bashrc
```



```
# env
alias python=python3
export ROS_DOMAIN_ID=26

export ROBOT_TYPE=Muto
export RPLIDAR_TYPE=a1 # a1, 4ROS
echo "-----"
echo -e "ROS_DOMAIN_ID: \033[32m$ROS_DOMAIN_ID\033[0m"
echo -e "my_robot_type: \033[32m$ROBOT_TYPE\033[0m | my_lidar: \033[32m$RPLIDAR_TYPE\033[0m"
echo "-----"

#colcon cd
source /usr/share/colcon_cd/function/colcon_cd.sh
export _colcon_cd_root=/root/yahboomcar_ros2_ws/yahboomcar_ws
source /usr/share/colcon_argcomplete/colcon-argcomplete.bash

#ros2
source /opt/ros/foxy/setup.bash
source /root/yahboomcar_ros2_ws/yahboomcar_ws/install/setup.bash
source /root/yahboomcar_ros2_ws/software/library_ws/install/setup.bash
```

After the modification is completed, save and exit vim, and then execute:

```
root@jetson-desktop:~# source .bashrc
-----
ROS_DOMAIN_ID: 26
my_robot_type: Muto | my_lidar: a1
-----
root@jetson-desktop:~#
```

You can see the current modified lidar type.

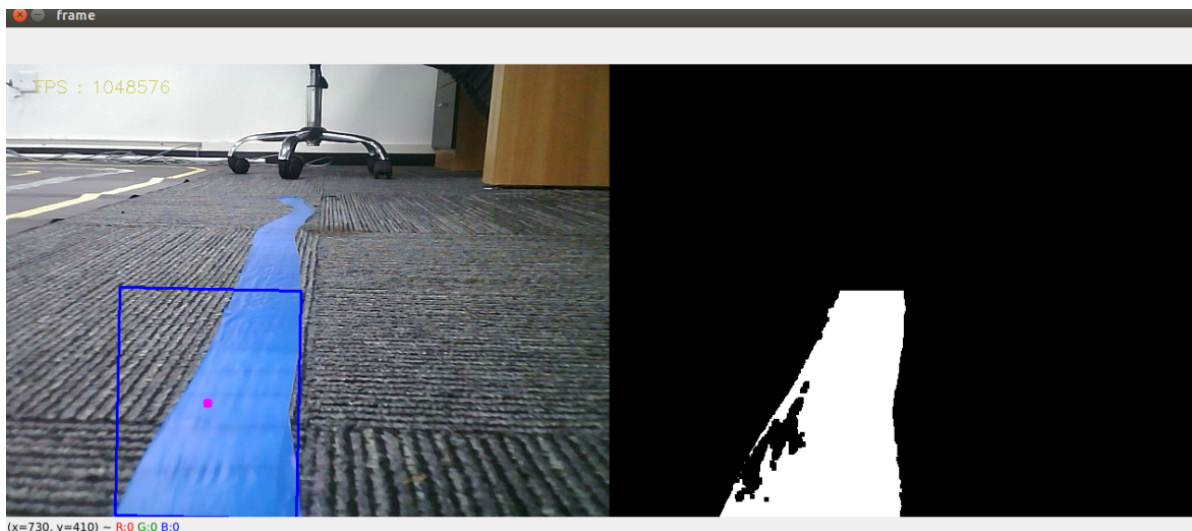
4. Program startup

4.1. Start command

After entering the docker container, enter in the terminal,

```
ros2 launch yahboomcar_voice_ctrl voice_ctrl_follow_line_launch.py
```

Pull down Muto's camera so that it can see the line, and then wake up the module first ("Hello, Yahboom"). After getting the reply, you can use the blue line to say "check blue line" to it.



Press the R2 button on the handle to start line following.

4.2. Adjust HSV value

The hsv value loaded in the program is not used for all scenes. As we all know, the effect of light on image processing will be quite large. Therefore, if the image processing effect of the loaded HSV value is not good, then the HSV value needs to be recalibrated. The calibration method is as follows:

- Run the line tracking program and dynamic parameter adjuster,

```
ros2 run rqt_reconfigure rqt_reconfigure
```

- Press the r key on the keyboard to enter the color selection mode, and frame an area where the line needs to be followed. Then, click on the blank interface of the reconfigure_GUI interface, and you will find that the HSV value inside changes. Correspond these values to self.hsv_range in Voice_Ctrl_follow_line_a1.py and modify them. Be careful not to mix up the colors.
- Finally, after modifying the Voice_Ctrl_follow_line_a1.py code, you need to return to the yahboomcar_ws directory, use **colcon build** to compile and **source install/setup.bash**.

5.Core code

The principle of line following here is the same as that of the previous "10.AI vision course-7.Visual line following", which is to calculate the speed based on the center coordinates of the processed image. The only extra part is to load the hsv value through voice. Previously, you needed to manually select the color of the line with the mouse. With the voice module, we only need to load the corresponding hsv value according to the command. The core content is as follows:

```
def process(self, rgb_img, action):

    binary = []
    rgb_img = cv.resize(rgb_img, (640, 480))

    if self.img_flip == True: rgb_img = cv.flip(rgb_img, 1)
    #Start receiving voice commands here, issuing instructions and loading hsv
    values.
    self.command_result = self.spe.speech_read()
    self.spe.void_write(self.command_result)
    if self.command_result == 23:
        self.model = "color_follow_line"
        print("red follow line")
        #红色HSV
        self.hsv_range = [(0, 84, 131), (180, 253, 255)]
        .....
    #The following part is to pass the value of hsv in, image processing, get a value
    of self.circle, and finally pass in the function of self.execute to calculate the
    speed
    if self.model == "color_follow_line":
        rgb_img, binary, self.circle = self.color.line_follow(rgb_img,
        self.hsv_range)
        if len(self.circle) != 0:
            threading.Thread(target=self.execute, args=(self.circle[0],
            self.circle[2])).start()
```

The instruction words in this course correspond to the following:

command word	Speech recognition module results
Close tracking mode	22
track red line	23
track green line	24
track blue line	25
track blue line	26