3. Voice control Autopilot

- 3. Voice control Autopilot
  - 3.1. Description
  - 3.2. Steps
    - 3.2.1. function package path
    - 3.2.2. Start
    - 3.2.3. Regulated HSV value
  - 3.3. Code analysis
  - 3.4. Voice module communication protocol

Raspberry Pi PI5 master controller needs to check this step, Orin master controller does not need it

Before running this program, it is necessary to bind the port number of the voice board and the port number of the ROS extension board on the host computer; When entering the docker, you need to mount the voice board to recognize the voice board in the docker.

3.1. Description

Voice control robot open and close tracking red/blue/green/yellow function, and the R2 key on the handle can stop/start this function at any time.

3.2. Steps

3.2.1. function package path

Raspberry Pi PI5 master control needs to enter the docker container first, Orin master control does not need to enter,

The location of the function source code is located

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voic
e_ctrl/Voice_Ctrl_follow_line_a1_R2.py
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voic
e_ctrl/Voice_Ctrl_follow_line_4ROS_R2.py
```

The structure of A1 radar between S2 radar is the same and can be shared

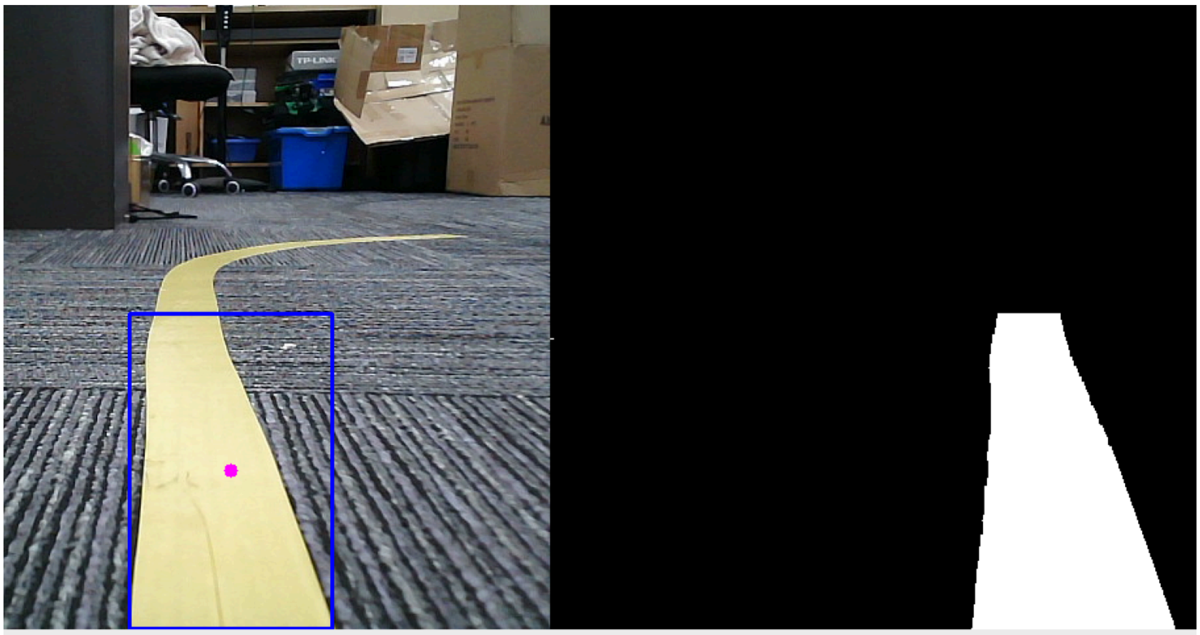
3.2.2. Start

Enter the docker, according to the actual vehicle type and radar type, terminal input

```
#Start the voice patrol program A1 radar
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_follow_line_a1_R2
#Start the voice patrol program 4ROS2 radar
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_follow_line_4ROS_R2
#Start the handle control node
ros2 run yahboomcar_ctrl yahboom_joy_R2
ros2 run joy joy_node
#Start A1 radar
ros2 launch sllidar_ros2 sllidar_launch.py
#Start 4ROS radar
ros2 launch ydlidar_ros2_driver ydlidar_launch.py
#Start the car chassis
ros2 run yahboomcar_bringup Ackman_driver_R2
```

*** (Take tracking red for example) ***

1. Put ROSMASTER on the yellow line, adjust the camera position, and push the camera down.
2. After the program is run, we can say "Hi Yahboom" to wake up the voice module, until it replies "Hi, i'm here", indicating that the module has been woken up.
3. We can say "Tracking the yellow line" , it will reply "OK, I will track the yellow line".
4. Next, we press the R2 key on handle, then ROSMASTER starts track the yellow line. If you don't use handle, you can also start ROSMASTER by inputting the following command through the terminal.



If you want to cancel this color tracking function, say "Stop the tracking function", it replies "OK, tracking mode is closed". ROSMASTER will cancel this function.

When the robot is moving, you can pause the robot by pressing the R2 key on handle again.

3.2.3. Regulated HSV value

Note: The HSV value can be modified according to the actual situation. Since the camera is more sensitive to light, it may appear that the HSV value here may be different, and the line tracking effect is not very good. The user can adjust the maximum and minimum values of HSV with dynamic parameters, modify the max and min values of the calibrated color HSV into the above code, and use the calibrated values after restarting the code.

- Autopilot program and dynamic parameter adjuster

```
ros2 run yahboomcar_linefollow follow_line_a1_R2.py
ros2 run rqt_reconfigure rqt_reconfigure
```

- Press the "r" key on your keyboard to enter color selection mode, Frame an area where you need to patrol, click the blank screen of the reconfigure_GUI interface, HSV value will be change, change them from <Voice_Ctrl_follow_line_a1_X3.py > self.hsv_range, be careful not to mix up the colors.
- finally, Modified <Voice_Ctrl_follow_line_a1_X3.py code>, You need to go back to the yahboomcar_ws directory, Compile with **colcon build** and **source install/setup.bash**

3.3. Code analysis

The speed is calculated by the center coordinates of the processed image. There is a voice module, so we only need to load the corresponding HSV value according to the command.

Core code:

```
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)
    #Here begins receiving voice commands as well as issuing commands 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")
    #red HSV
    self.hsv_range = [(0, 84, 131), (180, 253, 255)]
    .....
#The following part is to pass the HSV value into the image processing, return
self.circle, pass in self.execute function and 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()
```

3.4. Voice module communication protocol

Voice command	Speech Recognition Module Results	Voice broadcast content
Stop the tracking function	22	OK, tracking mode is closed
Start track red line	23	OK, I will track the red line

Voice command	Speech Recognition Module Results	Voice broadcast content
Start track green line	24	OK, I will track the green line
Start track blue line	25	OK, I will track the blue line
Start track yellow line	26	OK, I will track the yellow line