Voice controlled mechanical dog patrolling the line

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. Color line, after receiving the instruction, the program recognizes the color, loads the processed image, and then presses the L2 key of the handle to start the program. The car will start moving along the route identified by the color. During the automatic driving process, it will stop when it encounters obstacles.

2. Program code reference path

After entering the shell terminal or docker container, the location of the source code of this function is:

```
#pi4
/home/pi/cartographer_ws2/src/voice_xgo_ctrl_run/voice_xgo_ctrl_run/voice_xgo_follow_line.py
#pi5
/root/yahboomcar_ws/src/voice_xgo_ctrl_run/voice_xgo_ctrl_run/voice_xgo_follow_line.py
```

3. Program startup

3.1. Start command

Connect the mechanical dog through vnc and enter in the terminal:

pi4 version steps:

```
#startchassis
sudo systemctl start YahboomStart.service
cd /home/pi/cartographer_ws2
source install/setup.bash
#Start the voice line patrol program
ros2 run voice_xgo_ctrl_run voice_follow_line
```

Reopen a terminal and enter:

```
cd /home/pi/cartographer_ws2
source install/setup.bash
#Start the handle control node
ros2 run yahboom_ctrl yahboom_joy
ros2 run joy joy_node
```

pi5 version steps:

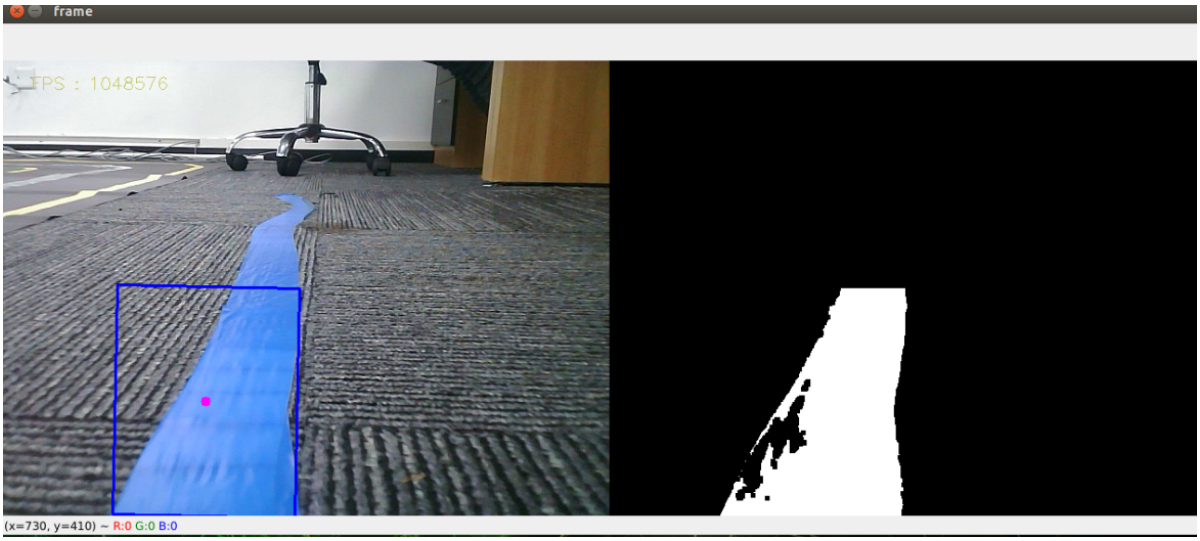
After entering the docker terminal, enter,

```
#startchassis
ros2 launch bringup Navigation_bringup.launch.py
#Start the voice line patrol program (need to enter the same docker terminal)
ros2 run voice_xgo_ctrl_run voice_follow_line
```

Reopen a terminal and enter:

```
#Start the handle control node
ros2 run yahboom_ctrl yahboom_joy
ros2 run joy joy_node
```

Pull down the car's camera so that it can see the line, and then wake up the module first ("Hello, Xiaoya"). After getting the reply, take the patrol blue line as an example, you can say " patrol the blue line" to it,



By default, the controller does not release speed messages. You need to press the start button before you can use the joystick to control it. In this routine, joystick control is not used by default.

Press the L2 button of the handle twice. When the code below appears on the interface, the mechanical dog starts to act.

```
pi@yahboom:~/cartographer_ws2$ ros2 run voice_xgo_ctrl_run voice_follow_line
import finish
cv_edition:  4.6.0
Speech Serial Opened! Baudrate=115200
start it
0
Cancel color_follow_line
25
bule follow line
2
2
0
Cancel color_follow_line
25
bule follow line
按下了 std_msgs.msg.Bool(data=True)
按下了 std_msgs.msg.Bool(data=False)
color_radius: 29 Joy_active: False
0.18
color_radius: 29 Joy_active: False
```

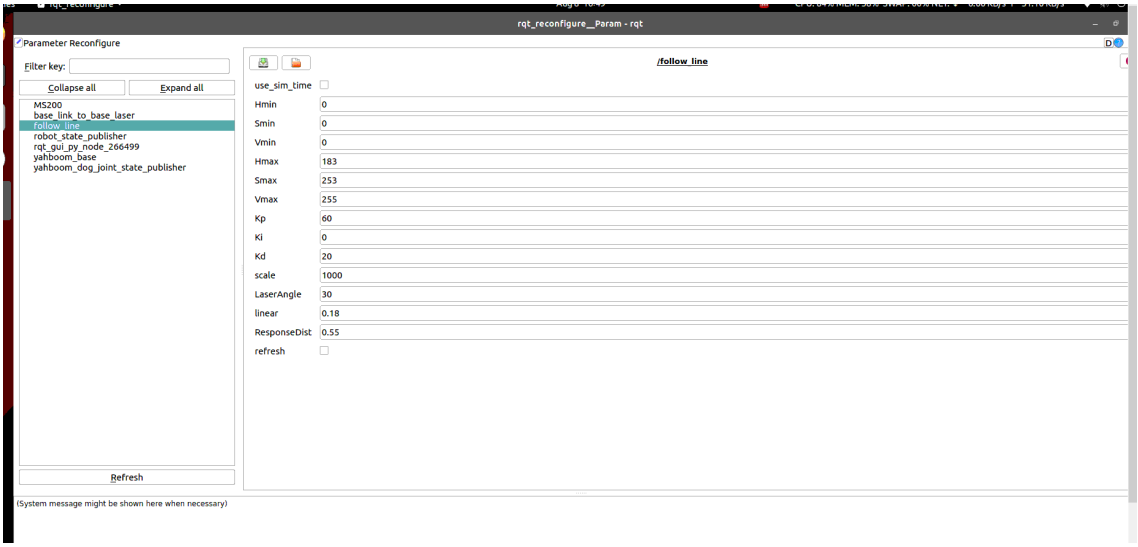
3.2. Adjust HSV value

The HSV value loaded in the program is not used in 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 it needs to be recalibrated. The value of hsv is calibrated as follows:

- Run line tracking program and dynamic parameter adjuster,

```
#PI4
cd cartographer_ws2
source install/setup.bash
ros2 run voice_xgo_ctrl_run follow_line
ros2 run rqt_reconfigure rqt_reconfigure
#PI5 (need to enter the docker container terminal)
ros2 run voice_xgo_ctrl_run follow_line
ros2 run rqt_reconfigure rqt_reconfigure
```

- Click the mouse to the camera interface, press the r key on the keyboard to enter the color selection mode, frame an area where the line needs to be followed, and then click on the blank interface of the reconfigure_GUI interface, you will find the HSV value inside If changes occur, map these values to self.hsv_range in voice_xgo_follow_line.py and modify them. Be careful not to mix up the colors.



- Finally, after modifying the voice_xgo_follow_line.py code, you need to return to the cartographer_ws2 directory, use **colcon build** to compile and **source install/setup.bash**.

4. Core code

Here, the speed is calculated from the center coordinates of the processed image, and the hsv value is loaded through voice. 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)
#Here we start receiving voice commands, 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")
#redHSV
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 self.execute function 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 words	Speech recognition module results
Turn off line patrol	22
Patrol the red line	23
Patrol Green Line	24
Patrol the blue line	25
Yellow Line Patrol	26