4. Voice control automatic driving

4.1. Function description

By interacting with the voice recognition module on ROSMASTER, you can turn on or off ROSMASTER's red/blue/green/yellow line patrol function by voice. The R2 key on the handle can cancel/turn on this function at any time.

4.2. Start up

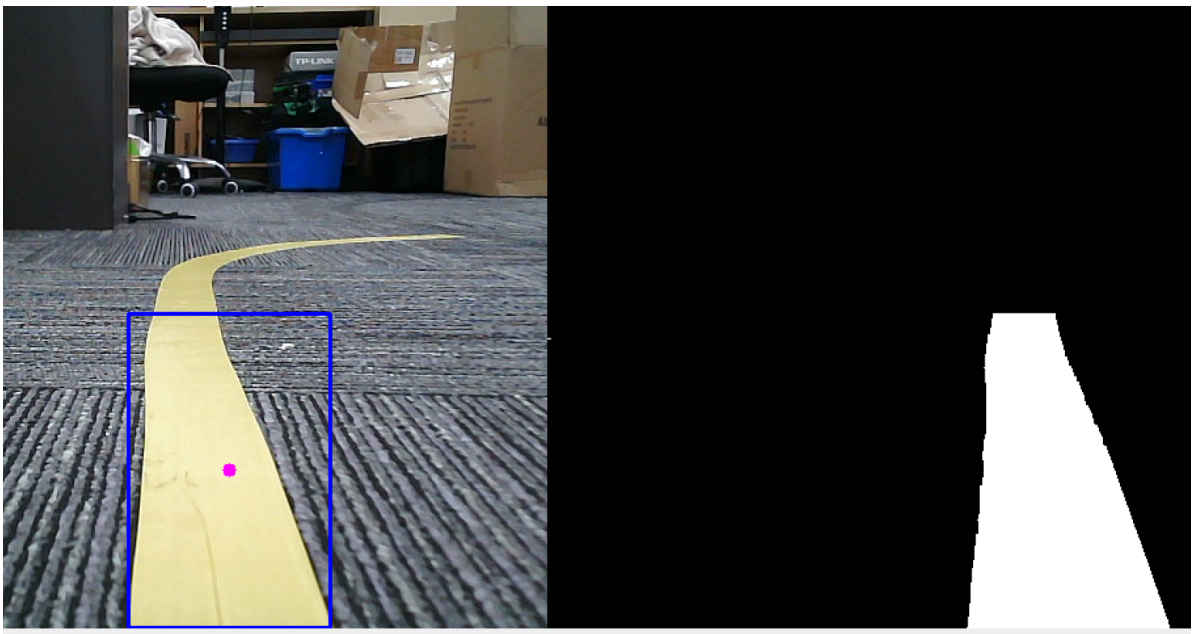
4.2.1. Function package path

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

4.2.2. Start up

```
roslaunch yahboomcar_voice_ctrl voice_ctrl_followline.launch    #Enable chassis
control
python3
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/scripts/voice_Ctrl_follow_line.py
#Turn on line tracking function
```

Take the yellow line patrol as an example, put ROSMASTER on the yellow line. After the program is started, call ROSMASTER "Hello, Xiaoya" to wake up the module. When it broadcasts "in", it means waking up the module. Next, you can say "Patrol Yellow Line" to it, and ROSMASTER will broadcast "Okay, it's on." Yellow line patrol function".



Then, we release the handle's control of ROSMASTER by pressing the R2 key of the handle, and ROSMASTER begins to patrol the yellow line. If there is no remote control, you can also enter the following command through the terminal,

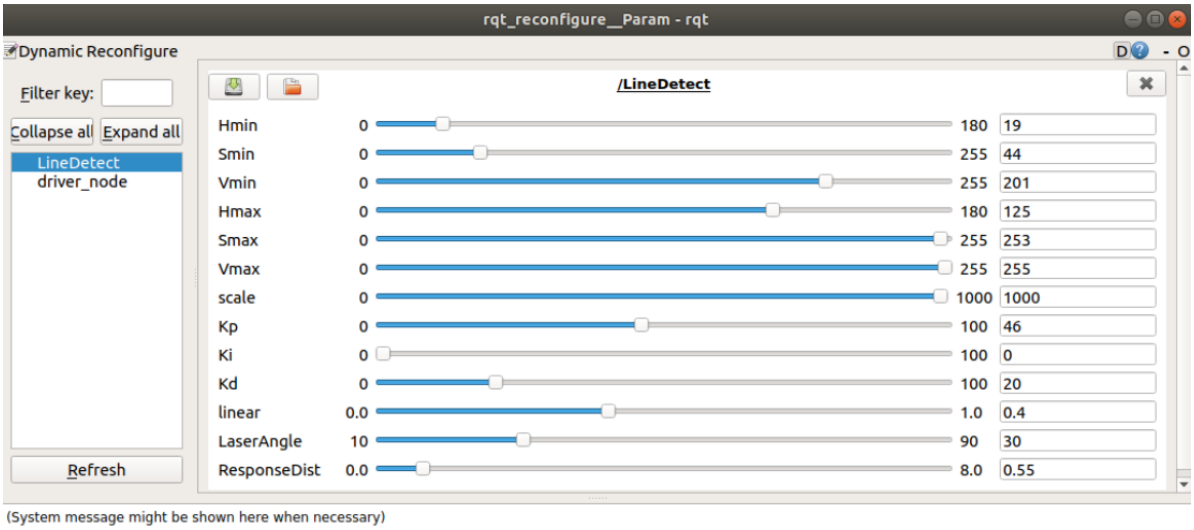
```
rostopic pub /JoyState std_msgs/Bool False
```

If you want to cancel the line patrol function, say "Turn off line patrol" to ROSMASTER. ROSMASTER will stop and the voice will broadcast "OK, line patrol function has been turned off".

4.2.3. Dynamic parameter adjustment

```
rosrun rqt_reconfigure rqt_reconfigure
```

Open the dynamic adjustment parameters, select the LineDetect column, and then adjust the parameters inside. Manually modify the adjusted parameters to voice_Ctrl_follow_line.py, and restart the program to use the adjusted parameters.



4.3. Core code analysis:

4.3.1. Import the speech recognition library and create speech recognition objects

```
from Speech_Lib import Speech
self.spe = Speech()
```

4.3.2. Modify the value of hsv_range (process function) based on the content read through speech recognition, and then obtain the value of circle

```
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")
    self.hsv_range =  [(0, 106, 175), (180, 255, 255)]

elif self.command_result == 24 :
    self.model = "color_follow_line"
    print("green follow line")
```

```
    self.hsv_range =  [(55, 105, 136), (95, 255, 255)]

elif self.command_result == 25 :
    self.model = "color_follow_line"
    print("bule follow line")
    self.hsv_range =  [(55, 134, 218), (125, 253, 255)]

elif self.command_result == 26 :
    self.model = "color_follow_line"
    print("yellow follow line")
    self.hsv_range =  [(17, 55, 187), (81, 255, 255)]
rgb_img, binary, self.circle = self.color.line_follow(rgb_img, self.hsv_range)
```

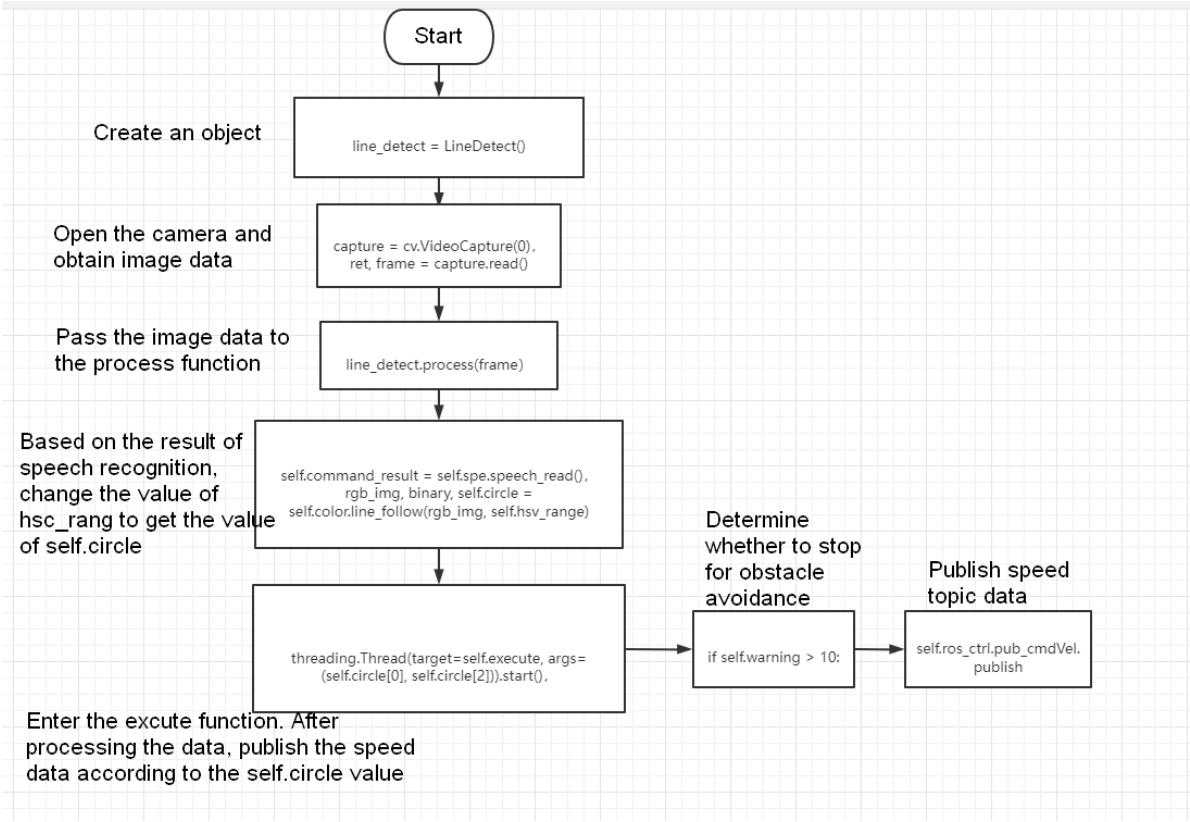
Note: The value of hsv here can be modified according to the actual situation. Since the camera is sensitive to light, the hsv value here may be different, and the line following effect is not very good. Users can adjust the maximum and minimum values of HSV with dynamic parameters, modify the max and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

4.3.3. Publish speed to chassis (execute function)

```
if color_radius == 0: self.ros_ctrl.pub_cmdVel.publish(Twist())
    else:
        twist = Twist()
        b = Bool()
        [z_Pid, _] = self.PID_controller.update([(point_x - 320)/16, 0])
        if self.img_flip == True: twist.angular.z = +z_Pid
        else: twist.angular.z = -z_Pid
        twist.linear.x = self.linear
        if self.warning > 10:
            rospy.loginfo("Obstacles ahead !!!")
            self.ros_ctrl.pub_cmdVel.publish(Twist())
            self.Buzzer_state = True
            b.data = True
            self.pub_Buzzer.publish(b)
        else:
            if self.Buzzer_state == True:
                b.data = False
                for i in range(3): self.pub_Buzzer.publish(b)
                self.Buzzer_state = False
            self.ros_ctrl.pub_cmdVel.publish(twist)
```

According to the value of self.circle obtained, it is passed into execute as an actual parameter, data is processed, whether to avoid obstacles is judged, and finally the speed topic data is released.

4.3.4. Flowchart



For complete code, please refer to:

```
~/yahboomcar/src/yahboomcar_voice_ctrl/scripts/voice_Ctrl_follow_line.py
```

4.3.5. Function module communication table

Function words	Voice recognition module results	Voice broadcast content
Turn off line tracking function	22	OK, line tracking function is turned off
Turn on the red line patrol function	23	OK, the red line patrol function has been turned on
Turn on the green line patrol function	24	OK, the green line patrol function has been turned on
Turn on the blue line patrol function	25	OK, the blue line patrol function has been turned on
Turn on yellow line patrol function	26	OK, the yellow line patrol function has been turned on