3、Voice controlled automatic driving for car line inspection

This course needs to be combined with the hardware of the Rosmaster-X3 car, and only code analysis will be done here. Firstly, let's take a look at the built-in voice commands,

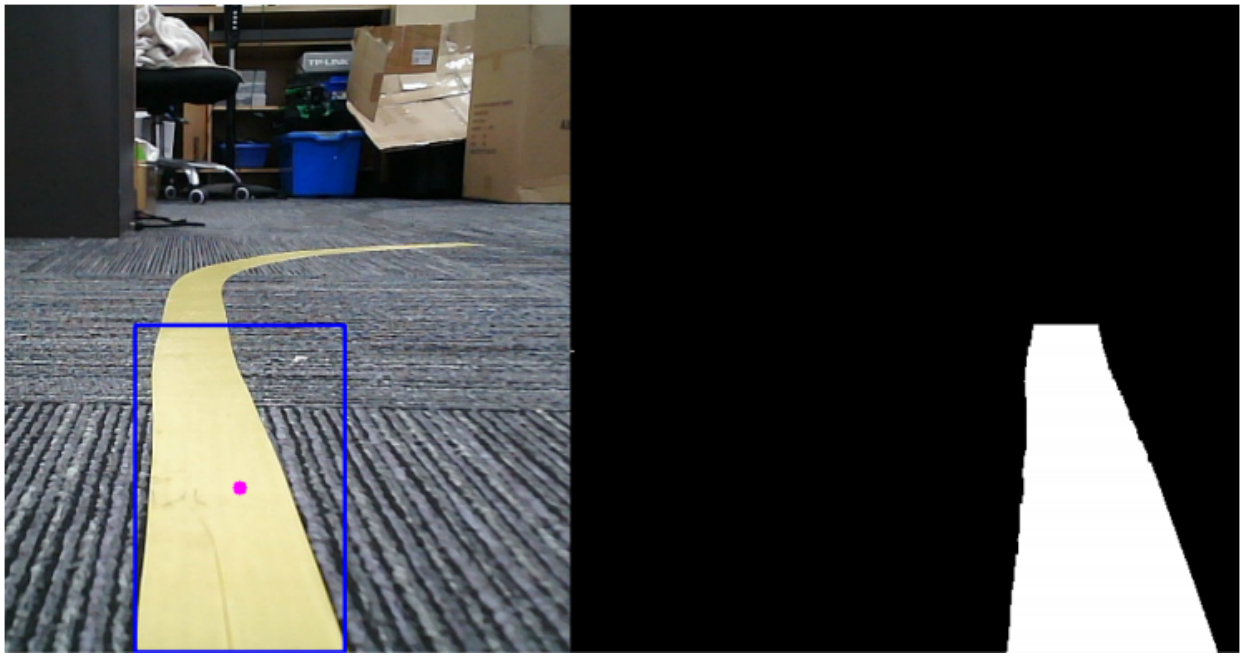
Directive word	Speech recognition results
Close tracking mode	22
track red line	23
track green line	24
track blue line	25
track yellow line	26

1、Enter the startup program

Terminal input,

```
#Start the trolley chassis
roslaunch yahboomcar_voice_ctrl voice_ctrl_followline.launch
#Enable line patrol function
python3 ~/driver_ws/src/yahboomcar_voice_ctrl/scripts/voice_Ctrl_follow_line.py
```

Taking the yellow line patrol as an example, place ROSMASTER on the yellow line, adjust the camera position, and break the camera down. After the program starts, call ROSMASTER "Hi Yahboom" to wake up the module. When it broadcasts "Hi , I'm here.", it means the module is awakened. Next, you can say "track yellow line" to ROSMASTER, and ROSMASTER will broadcast "OK, I will track the yellow line".



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

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

If you want to cancel the line patrol function, say 'Close tracking mode' to ROSMASTER, and ROSMASTER will stop. The voice will broadcast 'OK, tracking mode is closed'.

2、Core code

code path：~/driver_ws/src/yahboomcar_voice_ctrl/scripts/voice_Ctrl_follow_line.py

```
#Importing speech recognition libraries and creating speech recognition objects
from Speech_Lib import Speech
self.spe = Speech()
#Modify the hsv based on the content read from speech recognition_ The value of
range (process function), 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)
#Based on the obtained value of self.circle, it is passed into execute as an actual
parameter for data processing, determining whether to avoid obstacles, and finally
publishing speed topic data

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)
```