Voice Controlled Autopilot

1. Functional Description

By interacting with the voice array module on NAVROBO, you can use voice to turn on or off the NAVROBO red/blue/green/green line patrol function.

2. Start

Note: The [SWB] mid-range of the aircraft model remote control has the [emergency stop] function for this gameplay

- To start control, you need to first turn the SWB button to the upper gear position (control command mode) to release the remote control

2.1. Function package path

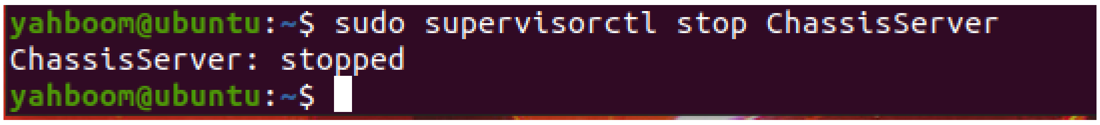
```
/home/yahboom/wukong-robot/voice_car_followline.py
```

2.2. Start

Turn off the self-start chassis service

This is because the self-start chassis occupies the camera, which needs to be released here.

```
sudo supervisorctl stop ChassisServer
```

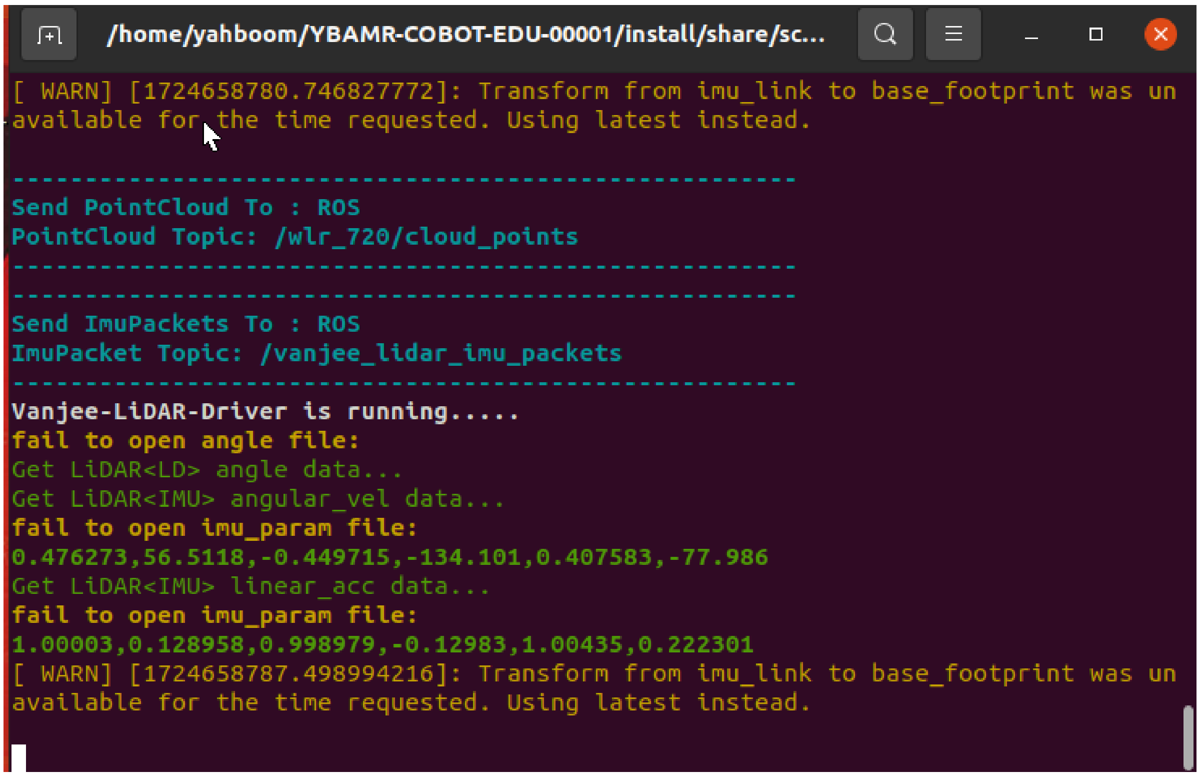


```
yahboom@ubuntu:~$ sudo supervisorctl stop ChassisServer
ChassisServer: stopped
yahboom@ubuntu:~$
```

Manually start chassis + radar service

```
roslaunch scout_bringup navrobot_base.launch
```

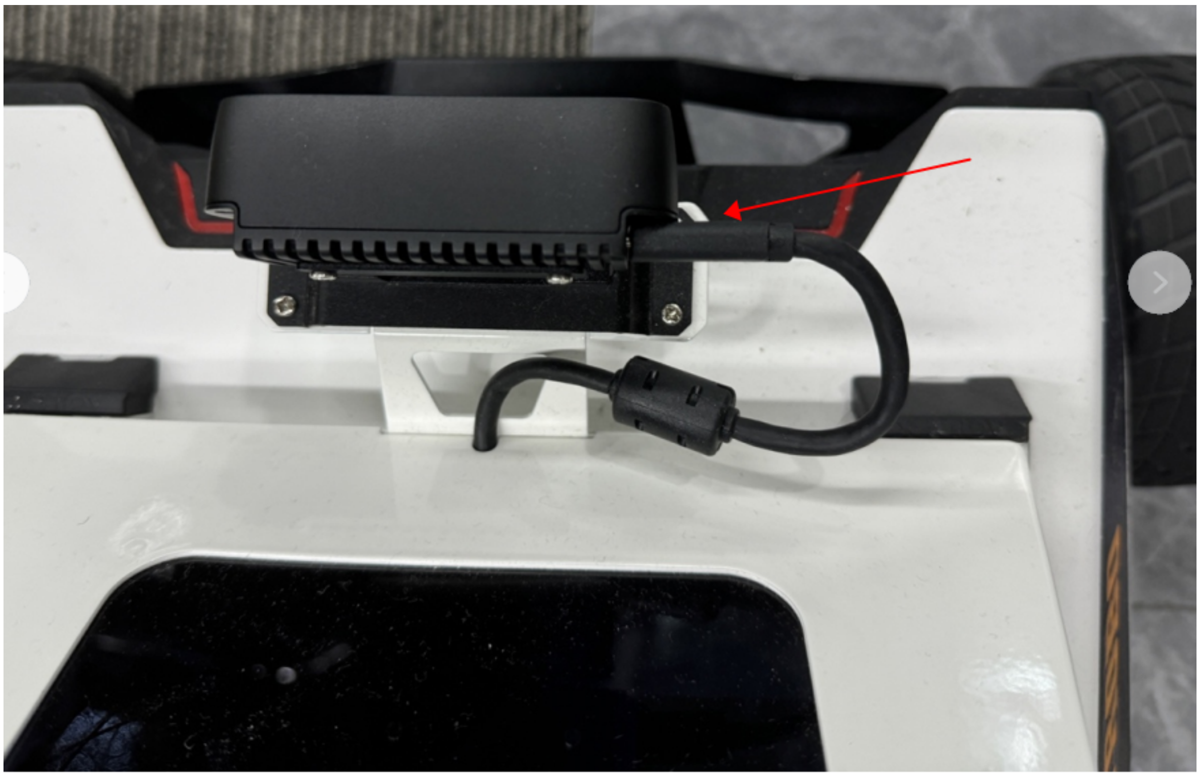
The appearance of imu and baselink conversion alarms will not affect normal use, because the time does not match.



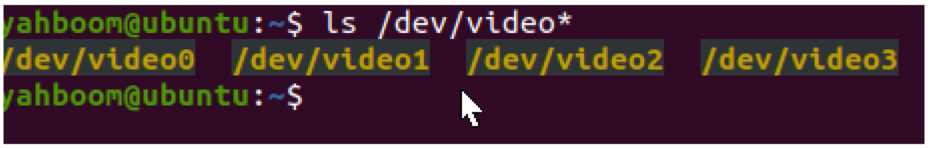
```
[ WARN] [1724658780.746827772]: Transform from imu_link to base_footprint was un
available for the time requested. Using latest instead.

------------------------------------------------------
Send PointCloud To : ROS
PointCloud Topic: /wlr_720/cloud_points
------------------------------------------------------
------------------------------------------------------
Send ImuPackets To : ROS
ImuPacket Topic: /vanjee_lidar_imu_packets
------------------------------------------------------
Vanjee-LiDAR-Driver is running.....
fail to open angle file:
Get LiDAR<LD> angle data...
Get LiDAR<IMU> angular_vel data...
fail to open imu_param file:
0.476273,56.5118,-0.449715,-134.101,0.407583,-77.986
Get LiDAR<IMU> linear_acc data...
fail to open imu_param file:
1.00003,0.128958,0.998979,-0.12983,1.00435,0.222301
[ WARN] [1724658787.498994216]: Transform from imu_link to base_footprint was un
available for the time requested. Using latest instead.
```

If video6 is not recognized, you can plug and unplug the USB port of the depth camera at this time,

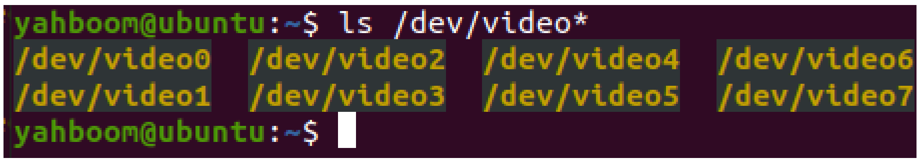


```
ls /dev/video*
```



```
yahboom@ubuntu:~$ ls /dev/video*
/dev/video0   /dev/video1   /dev/video2   /dev/video3
yahboom@ubuntu:~$
```

After plugging and unplugging, the following videos are displayed, which means the normal state.



```
yahboom@ubuntu:~$ ls /dev/video*
/dev/video0  /dev/video2  /dev/video4  /dev/video6
/dev/video1  /dev/video3  /dev/video5  /dev/video7
yahboom@ubuntu:~$ 
```

Open another terminal and enter the command to start voice control.

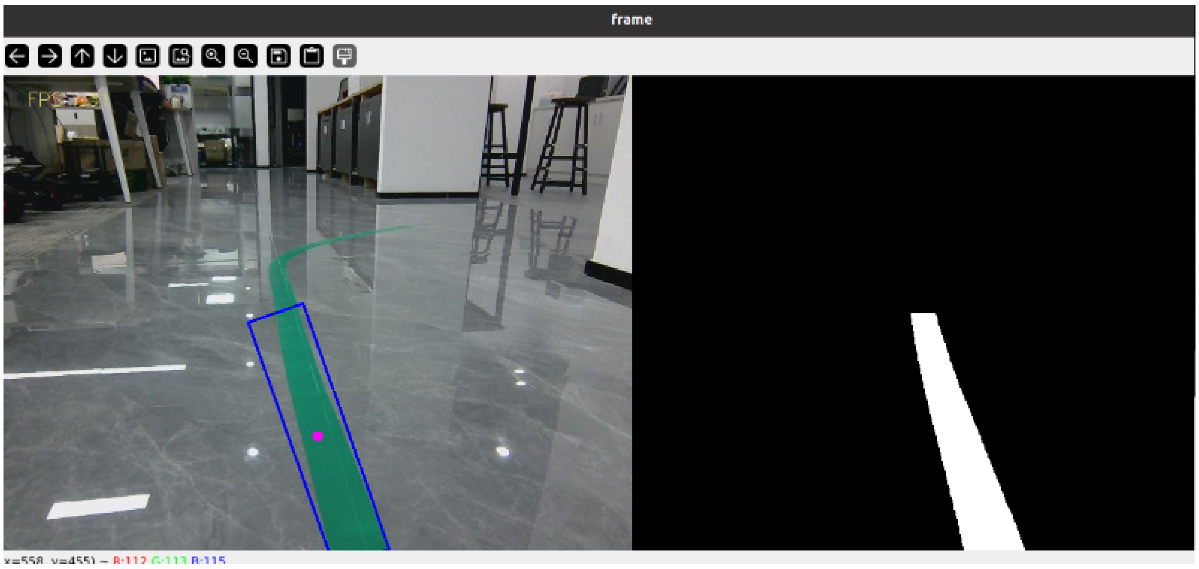
```
cd wukong-robot/
python3 voice_car_followline.py
```

- The car has the function of front close-range collision avoidance. It is necessary to ensure that the radar is started normally. If you run `rostopic echo /scan`

and the print is empty and the data cannot be obtained, the startup is abnormal. Please restart the radar service command.

- If there is no response to the call, you can restart the voice control command.

Take the green line patrol as an example, stop NAVROBO on the green line, adjust the camera position, and bend the camera down. After the program starts, call NAVROBO "Hello, Xiaoya" to wake up the module. When it broadcasts "Yes", it means the module is awakened. Then you can say "patrol the green line" to it. You can observe the value printed in the terminal. NAVROBO will broadcast "OK, the green line patrol function has been turned on."



Then, we turn the remote control SWB button to **upper gear**, release the control of NAVROBO, press **space bar**, and NAVROBO starts to patrol the green line. If there is no movement, you can enter the following command through the terminal,

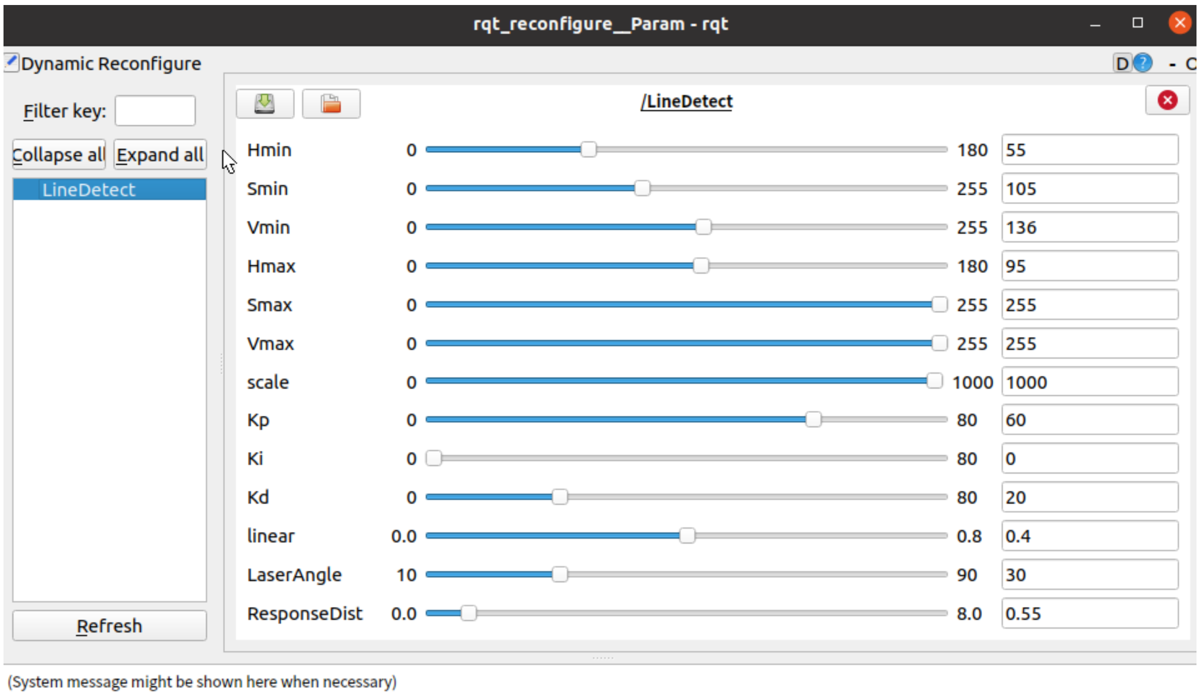
```
rostopic pub /JoyState std_msgs/Bool False
```

If you want to cancel the line patrol function or want to change the color of the line patrol, you can first turn the remote control SWB button to **mid-range**, then say "Xiao Ya" to NAVROBO, wake up the module, and then say "stop line patrol", NAVROBO stops, and the voice will broadcast "OK, the line patrol function has been stopped".

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. After adjusting the parameters, manually modify them to voice_car_follow.py, and restart the program to use the adjusted parameters.



Hmin	0	180	55
Smin	0	255	105
Vmin	0	255	136
Hmax	0	180	95
Smax	0	255	255
Vmax	0	255	255
scale	0	1000	1000
Kp	0	80	60
Ki	0	80	0
Kd	0	80	20
linear	0.0	0.8	0.4
LaserAngle	10	90	30
ResponseDist	0.0	8.0	0.55

(System message might be shown here when necessary)

2.4. Voice array communication table

Voice recognition content	Speech Array Module Results	Voice broadcast content
Tracking the red line	23	OK, I will track the red line
Tracking the green line	24	OK, I will track the green line
Tracking the blue line	25	OK, I will track the blue line
Tracking the yellow line	26	OK, I will track the yellow line

3, Core code analysis

3.1, Create a voice module object and get the return data

```
self.spe = Speech()
self.command_result = self.spe.speech_read()
```

3.2. Modify the value of hsv_range (process function) according to the content of the voice recognition, and then get the value of circle

```
if self.command_result == 23 :
self.model = "color_follow_line"
print("red follow line")
self.hsv_range = [(0, 84, 131), (180, 253, 255)]
#self.PID_init()
self.spe.void_write(self.command_result)

self.dyn_update = True

elif self.command_result == 24 :
self.model = "color_follow_line"
#print("green follow line")
self.hsv_range = [(55, 105, 136), (95, 255, 255)]
#self.PID_init()
self.spe.void_write(self.command_result) self.dyn_update = True elif
self.command_result == 25 : self.model = "color_follow_line" print("bule follow
line") self.hsv_range = [(55, 134, 218), (125, 253, 255)] #self.PID_init()
self.spe .void_write(self.command_result) self.dyn_update = True elif
self.command_result == 26 : self.model = "color_follow_line" print("yellow follow
line") #self.hsv_range = [(17, 55, 187), (81, 255, 255)] self.hsv_range = [(18,
45, 144), (125, 253, 255)]
#self.PID_init()
self.spe.void_write(self.command_result)

self.dyn_update = True

elif self.command_result == 22 or self.command_result == 0 :
self.model = "Stop"
self.Track_state = 'stop'
self.ros_ctrl.Joy_active = True
self.ros_ctrl.pub_cmdVel.publish(Twist())
print("Cancel color_follow_line")
rgb_img, binary, self.circle = self.color.line_follow(rgb_img, self.hsv_range)
```

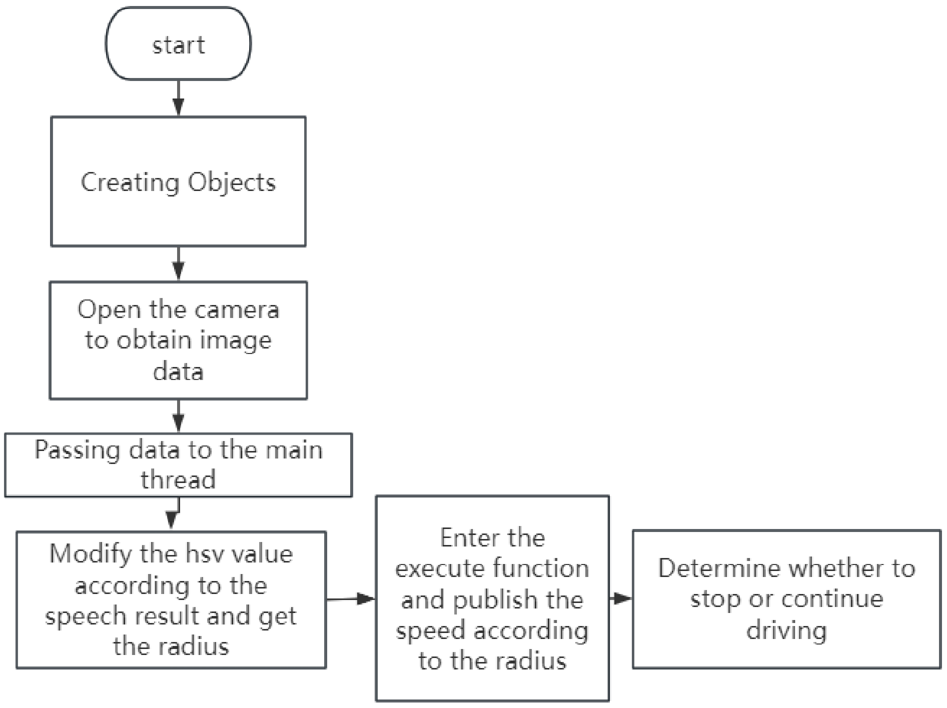
Note: The value of hsv here can be modified according to the actual situation. Since the camera is sensitive to light, the HSV value here may be different, and the line patrol effect may not be very good. Users can adjust the maximum and minimum values of HSV dynamically, modify the max and min values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

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])
            #rospy.loginfo("angular.z: {}".format(z_Pid))
            if self.img_flip == True: twist.angular.z = -z_Pid
            else: twist.angular.z = +z_Pid
            twist.linear.x = self.linear
            rospy.loginfo("point_x: {},linear: {:.2f}, angular.z: 
{:.2f}".format(point_x, twist.linear.x, twist.angular.z))
            #if(twist.angular.z>0.82 or twist.angular.z<-0.82):twist.angular.z = 
0
            if(twist.linear.x>0.82 or twist.linear.x<-0.82):twist.linear.x = 0
            if self.warning > 15:
                rospy.loginfo("Obstacles ahead !!!")
                self.ros_ctrl.pub_cmdVel.publish(Twist())
            else:
                self.ros_ctrl.pub_cmdVel.publish(twist)
```

According to the obtained value of self.circle, it is passed as the actual parameter to execute, data processing is carried out, obstacle avoidance is determined, and finally the speed topic data is published.

3.4、flow chart


start
Creating Objects
Open the camera to obtain image data
Passing data to the main thread
Modify the hsv value according to the speech result and get the radius
Enter the execute function and publish the speed according to the radius
Determine whether to stop or continue driving