4.Visual tracking autopilot

4.Visual tracking autopilot
- 4.1、Introduction
- 4.2、Steps
  - 4.2.1、Start up
  - 4.2.2、Identify
  - 4.2.3、Color calibration
  - 4.2.4、Autopilot

4.1、Introduction

The Transbot robot HD camera Autopilot can recognize multiple colors at any time, and autonomously storing the currently recognized colors.
According to the color of the detection and recognition, the function of real-time obstacle avoidance can also be realized in the process of Autopilot .

The color tracking of the Transbot robot can also realize the function of real-time HSV regulation. By adjusting the high and low thresholds of HSV, the interfering colors can be filtered out, so that the square can be identified ideally in a complex environment. If the color picking effect is not ideal At this time, we need to move the car to a different environment to calibrate it, so that we can recognize the color we need in a complex environment.

- **HSV**

H: 0 — 180

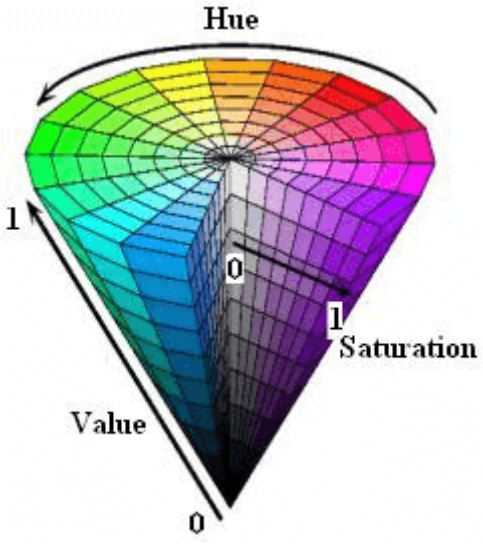
S: 0 — 255

V: 0 — 255

Part of the red is classified as the purple range here:

	black	gray	white	red		orange	yellow	green	verdant	blue	purple
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

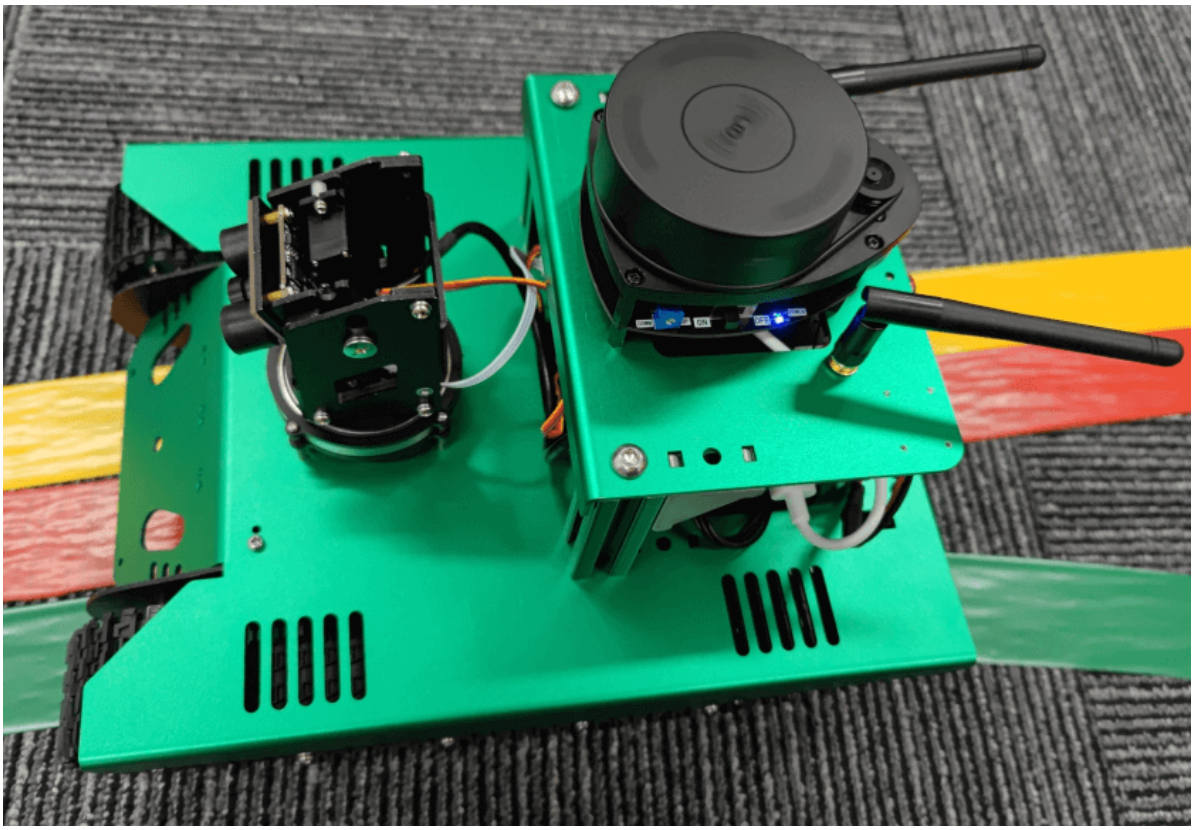
- **HSV**
  - **Lightness V**
  - **Saturation S**
  - **Hue H**



4.2、Steps

Note: The [R2] of the handle remote controller can [Pause/Open] for all functions of robot car

Place the robot at the starting position so that the depth camera is facing down. as shown below.



4.2.1、Start up

Start the robot drive, lidar and HD camera

```
roslaunch transbot_nav laser_bringup.launch
```

Method 1

Start up HD camera

```
roslaunch usb_cam usb_cam-test.launch
```

Start autopilot control

```
roslaunch transbot_linefollow follow_line.launch VideoSwitch:=False
```

In the same local area network, this method realizes remote control.

For example: Raspberry Pi start up usb_cam-test.launch， the virtual machine can start follow_line.launch

Method 2

Note:【q】key to exit.

```
roslaunch transbot_linefollow follow_line.launch VideoSwitch:=true
```

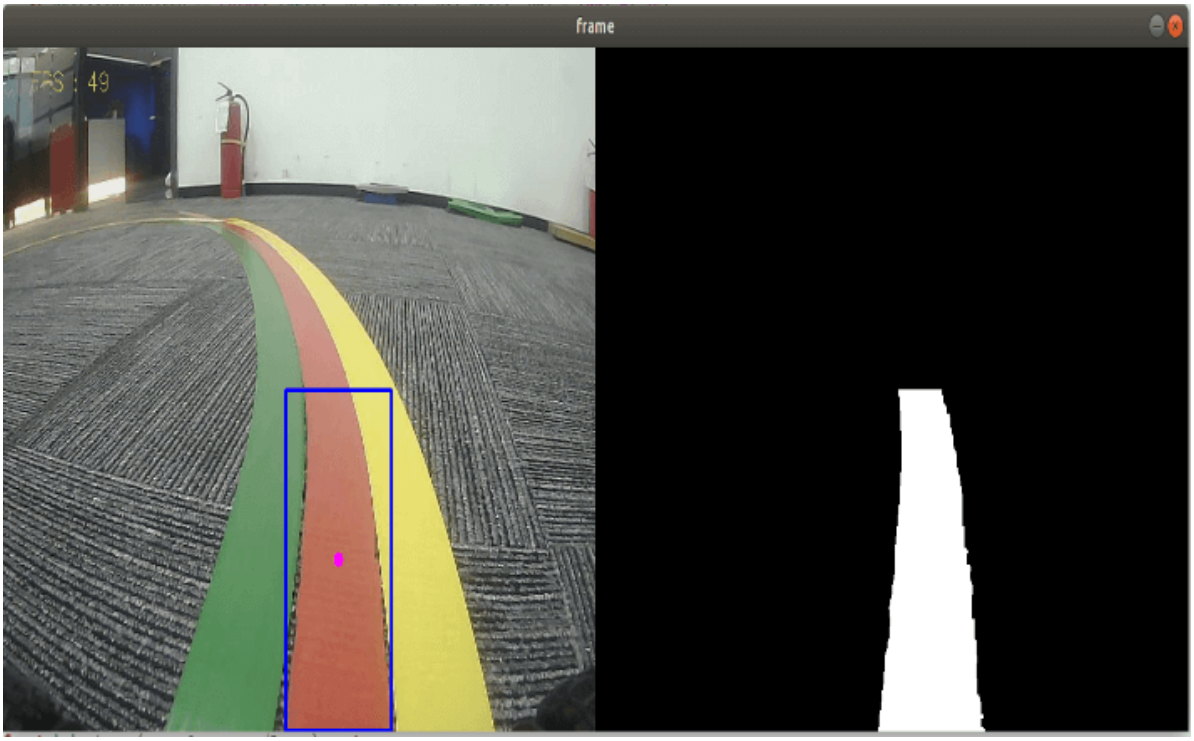
This method can only be activated in the master control that the camera is connected.

- VideoSwitch parameter: whether to use the camera function package to start; for example: start usb_cam-test.launch, this parameter must be set to True; otherwise, it is False

Set the parameters according to your needs, and you can also modify the launch file directly, so you don't need to attach parameters when you start.

4.2.2、Identify

After startup, the system defaults to [Target Detection Mode], as shown in the figure below.



Keyboard key control:

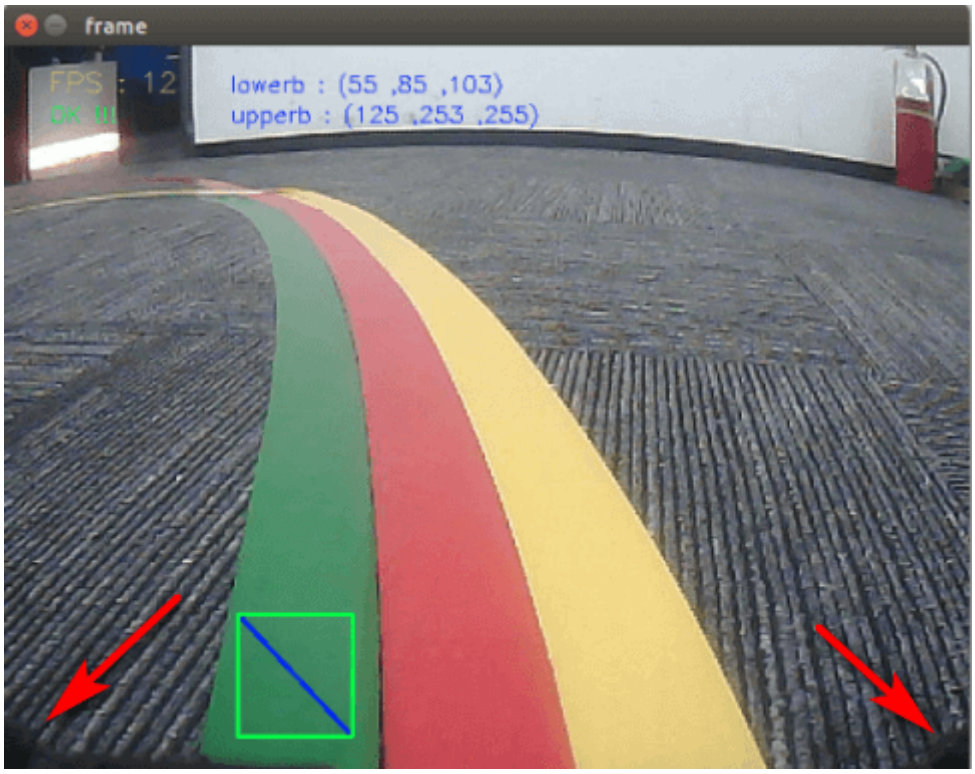
【r】: Color selection mode, the mouse can be used to select the area of the color to be recognized (cannot exceed the area range).

【i】: Target detection mode. Color map on the left (Color), binary map on the right (Binary).

【q】： Exit the program.

【Space key】: Start autopilot

In the color selection mode, the mouse can be used to select the area of the color to be recognized (cannot exceed the area range), as shown below, release it to start recognition.



We can use the handle or mobile APP to adjust the gimbal servo, roughly to the position where the crawlers can be seen on both sides, as shown below.

The system has set the initial value of the servo by default, but due to the reason of the servo itself, the position of the servo of each car is slightly different.

After debugging, we can modify the initial position of the gimbal servo of the autopilot function, and we didn't need to adjust it again the next time.

Source code location: transbot_linefollow\scripts\follow_common.py

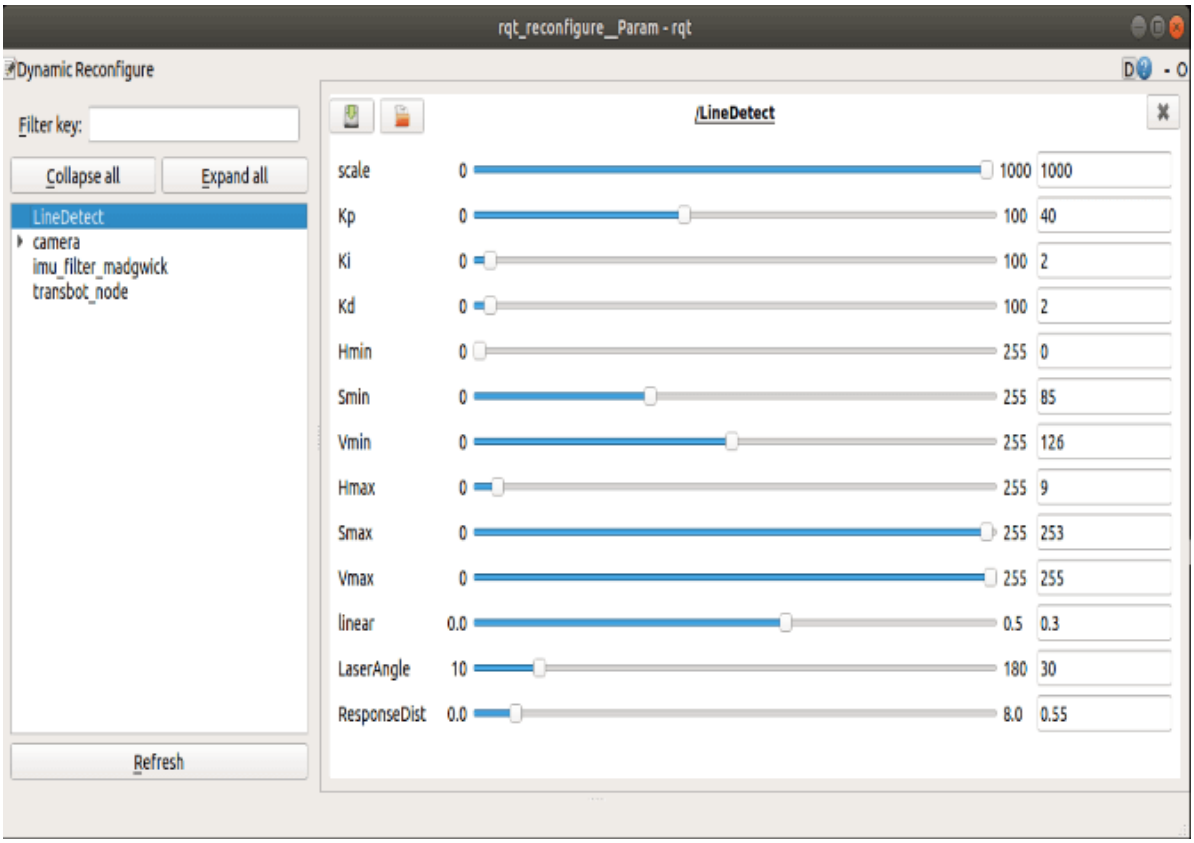
```
class ROSCtrl:
    def __init__(self):...

    def JoyStateCallback(self, msg):...

    def CamDevice_srv(self, value):...

    def Buzzer_srv(self, value):...

    def PWM_Reset(self):
        self.PWMServo_topic(1, 90)
        rospy.sleep(0.1)
        self.PWMServo_topic(2, 110)  # 115
        rospy.sleep(0.1)
        self.PWMServo_topic(1, 90)
```

4.2.3、Color calibration

Dynamic parameter debugging tool

```
rosrun rqt_reconfigure rqt_reconfigure
```

Set the mode to [Target Detection Mode] and start the dynamic parameter debugging tool.



Select [LineDetect] node, generally only need to adjust [Hmin], [Smin], [Vmin], [Hmax], these four parameters can be well identified.
The slider is always in the dragging state, and no data will be transferred to the system. The data will actually be transferred to the system when you release it; you can also select a row and then slide the mouse wheel.

Parameter analysis:

【Kp】、【Ki】、【Kd】：PID control during the movement of the car.

【scale】：PID scaling.

【linear】：Robot car running speed; range [0, 0.45], unit: meter; set as required.

【LaserAngle】：Lidar effective angle; range [0, 180], unit: degree; set as required.

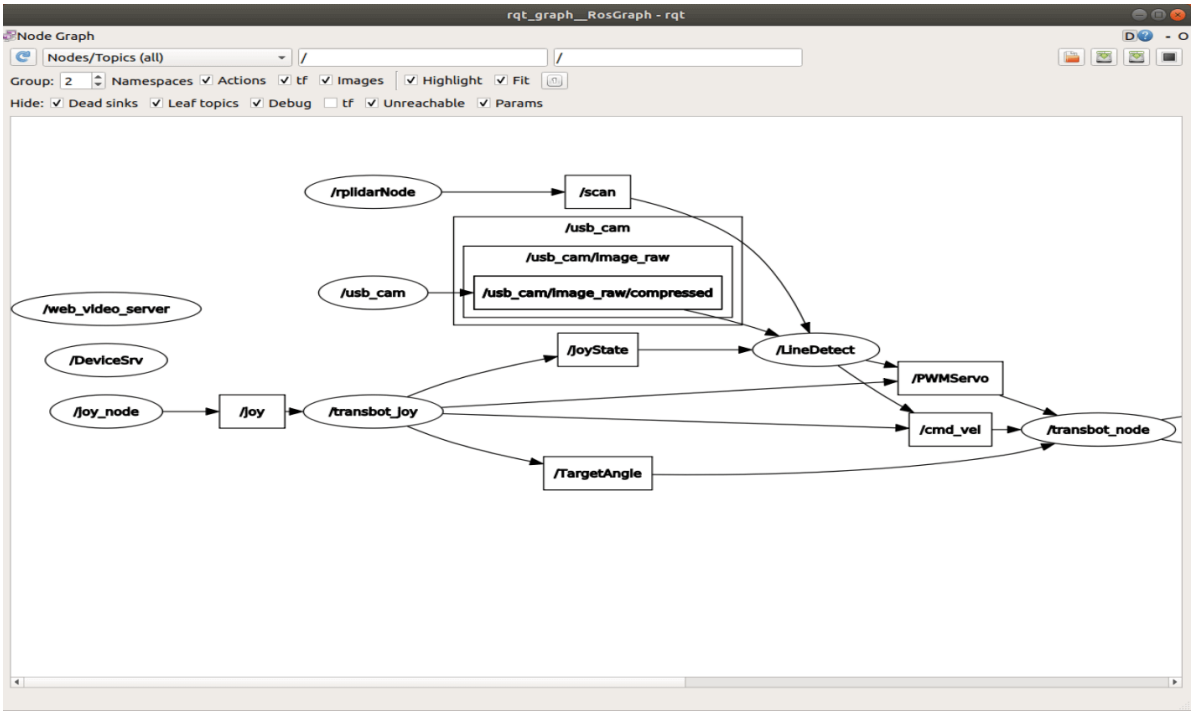
【ResponseDist】：Lidar response distance; range [0.15, 12], unit: meter; set as required.

4.2.4、Autopilot

After identifying is ok, click [Space key] on the keyboard to execute the Autopilot program.

- View node

```
rqt_graph
```



Node【LineDetect】

- Subscribe

  Lidar

  Image

  Handle

- Publish

  Car speed

  Gimbal servo