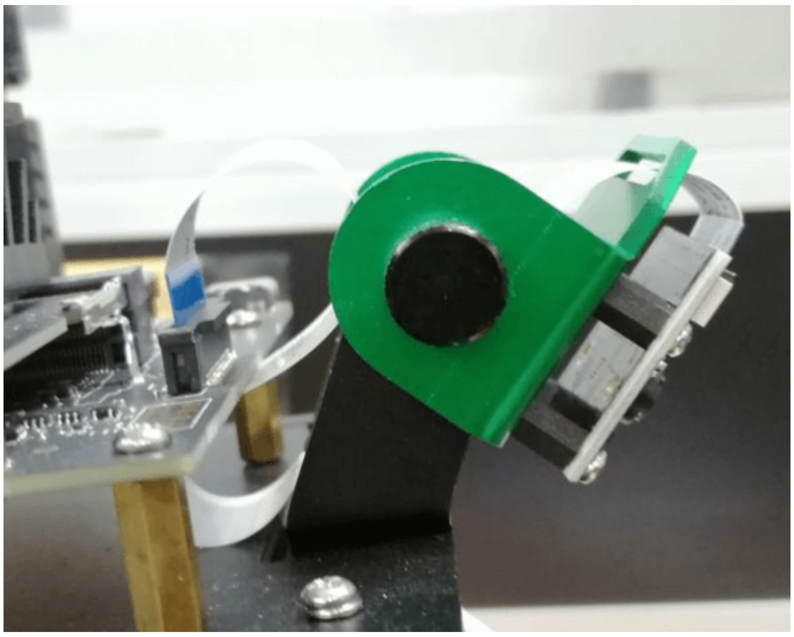
10.3 、Autopilot

To use the camera in this chapter, you need to manually twist the camera to face the line, otherwise it will not be able to drive automatically.



10.3.1、 Introduction

In this section, we realize autonomous driving through color recognition.

Jetbotmini robot autopilot, with the function of real-time regulation of HSV, control the color of the car detection and recognition, by adjusting the high and low thresholds of HSV, filter out the interference color, so that the square can be recognized ideally in a complex environment. If the color picking effect is not ideal, you need to move the car to a different environment to calibrate it, so as to be able to recognize the color we need in a complex environment.

The threshold setting was introduced in 4.2.6, you can also refer to the preset colors in section 6.6 for setting, or refer to the following table.

	黑	灰	白	红		橙	黃	绿	青	蓝	紫
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

10.3.2、Steps

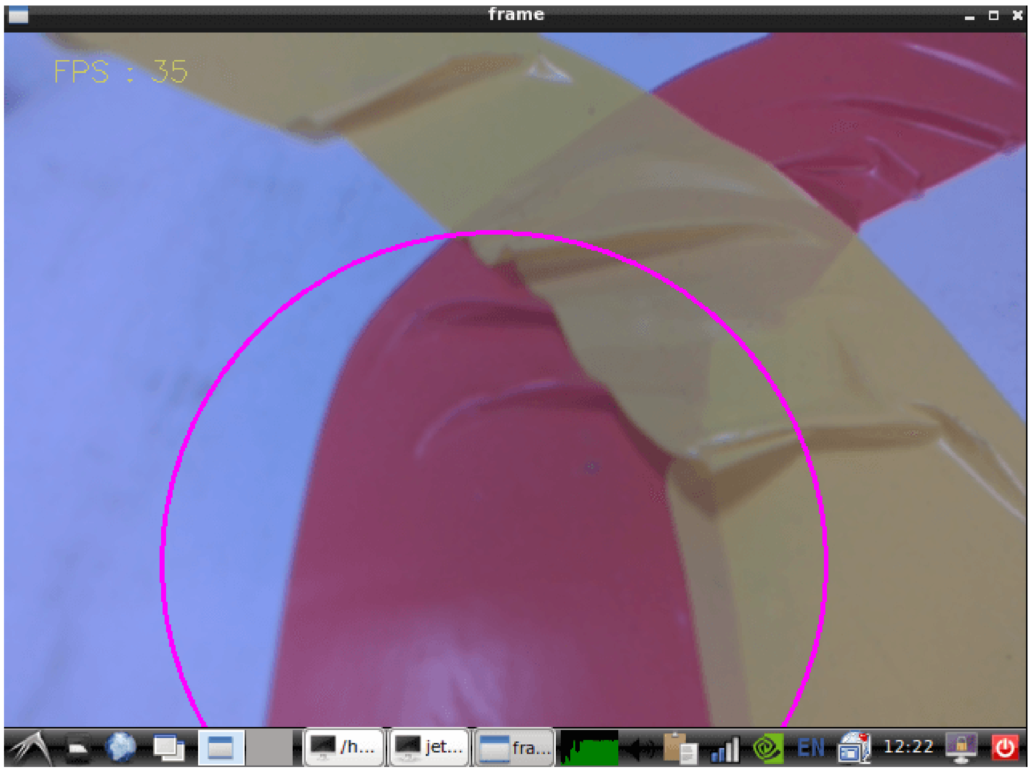
1. start up

Activate autopilot control

```
roslaunch jetbot_ros color_line.launch
```

2. Identify

Use the preset color for recognition after startup



Keyboard key control introduction：

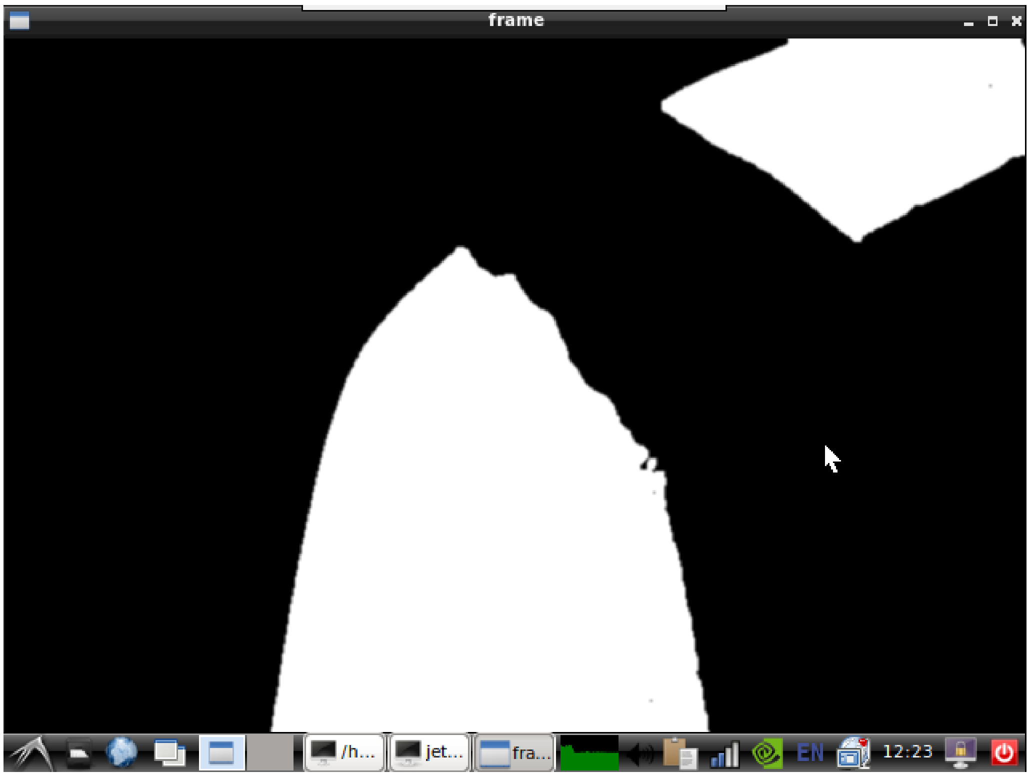
【s】： Start the color to follow.

【i】： Switch to Binary mode.

【Space key】： Switch to Color mode.

【q】： exit the program.

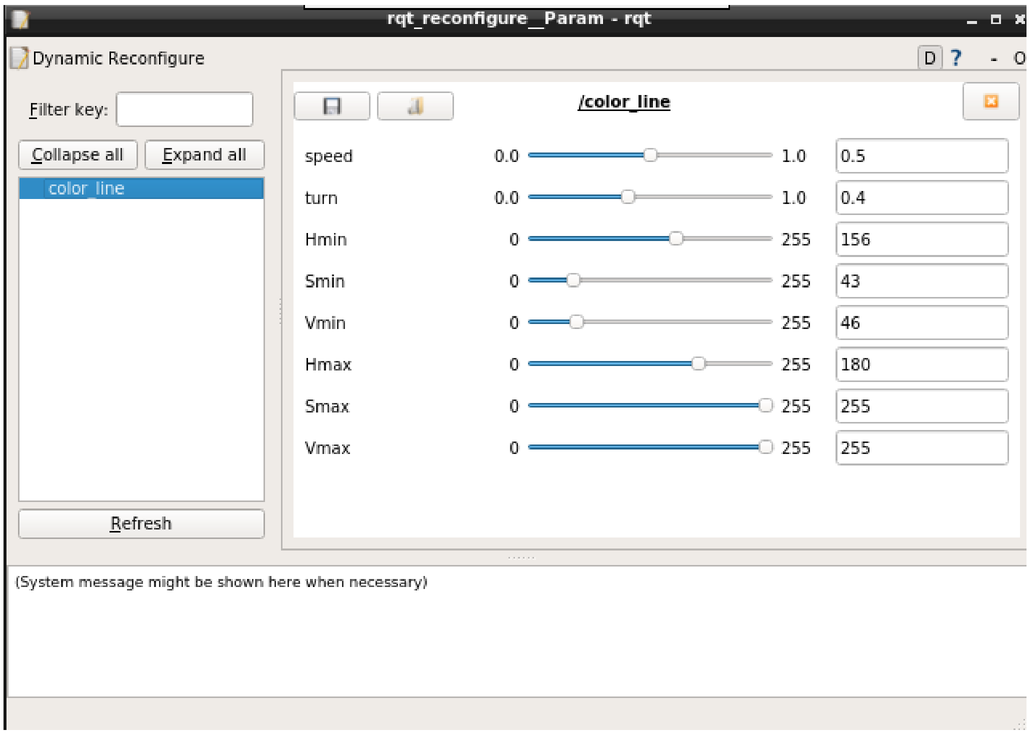
Press the key【i】to switch to Binary mode, the effect is as follows:



3. Color calibration

Dynamic parameter tuning

```
rosrun rqt_reconfigure rqt_reconfigure
```



```
speed   0.0  1.0   0.5
turn    0.0  1.0   0.4
Hmin    0    255   156
Smin    0    255   43
Vmin    0    255   46
Hmax    0    255   180
Smax    0    255   255
Vmax    0    255   255
```

(System message might be shown here when necessary)

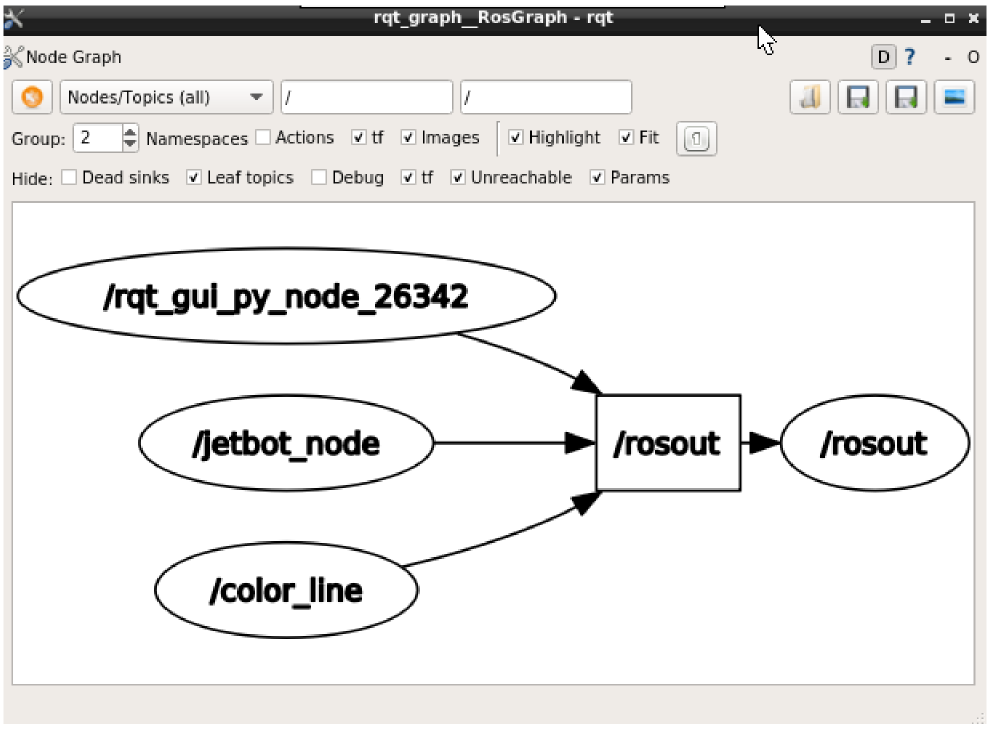
Select the [color_line] node, in the binary graph mode, adjust the HSV parameters, you can observe the recognized color changes, only need to adjust [Hmin], [Smin], [Vmin], [Hmax], these four The parameters can be well identified.

Note: The slider is always in the dragging state, and the data will not be transferred to the system. You can release it only after you release it; you can also select a row and then slide the mouse wheel.

After adjusting the color parameters, you can press [s] to start jetbotmini autopilot. If the effect is not good, you can adjust [speed] to adjust the speed of movement, and adjust [turn] to adjust the speed of turning

4. View node

```
rqt_graph
```


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/rqt_gui_py_node_26342
/jetbot_node
/color_line
/rosout
/rosout

5.Source code analysis

- launch file

  file path: /home/jetson/workspace/catkin_ws/src/jetbot_ros/launch/color_line.launch

```
<launch>
    <!--Start the bottom node of jetbotmini-->
    <node pkg="jetbot_ros" type="jetbotmini_driver.py" name="jetbot_node" 
required="true" output="screen">
    </node>
    <!--Start the color recognition node-->
    <node pkg="jetbot_ros" type="color_line.py" name="color_line" 
output="screen">
    </node>
</launch>
```

- python file

  file path: /home/jetson/workspace/catkin_ws/src/jetbot_ros/scripts/color_line.py

  Identify the color and convert the identified position to speed, and issue a service to control the motor.

```
    def colorDisplay(self, image, font_path):
        color_lower = np.array([self.Hmin, self.Smin, self.Vmin], 
dtype=np.uint8)
        color_upper = np.array([self.Hmax, self.Smax, self.Vmax], 
dtype=np.uint8)
```

```
        image=cv.GaussianBlur(image,(5,5),0)
        hsv = cv.cvtColor(image, cv.COLOR_BGR2HSV)
        hsv.astype(np.uint8)
        mask=cv.inRange(hsv,color_lower,color_upper)
        mask=cv.erode(mask,None,iterations=2)
        mask=cv.dilate(mask,None,iterations=2)
        mask=cv.GaussianBlur(mask,(3,3),0)

 cnts=cv.findContours(mask.copy(),cv.RETR_EXTERNAL,cv.CHAIN_APPROX_SIMPLE)[-2]
        if len(cnts)>0:
            cnt = max (cnts,key=cv.contourArea)
            (color_x,color_y),color_radius=cv.minEnclosingCircle(cnt)
            if color_radius > 30:
                # Mark the detected colors
                cv.circle(image,(int(color_x),int(color_y)),int(color_radius),
(255,0,255),2)
                center_x = (320 - color_x)/320
                # print(center_x)
                if mot_start == 1:
                    self.Motor_srv(
                        float(self.speed_value - self.turn_value * center_x),
                        float(self.speed_value + self.turn_value * center_x)
                    )
        else:
            self.Motor_srv(0,0)
        return image, mask
```