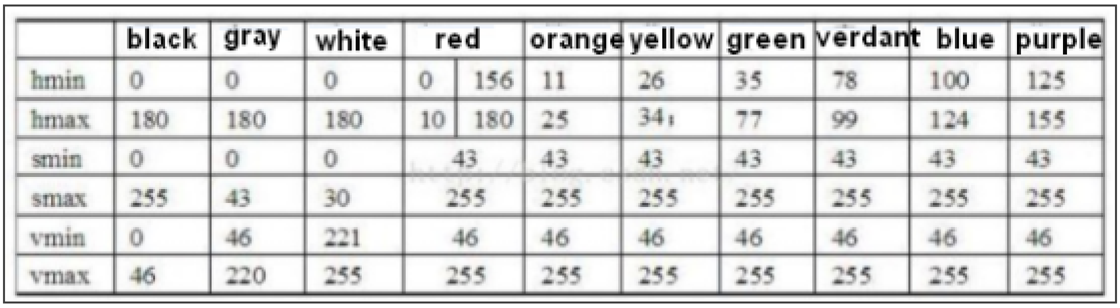
10.2、 Color follow

10.2.1、 Introduction

In the previous section, we used OpenCV to achieve color recognition. Next, we combined color recognition and ROS robot to achieve color following. The basic principle of implementation is similar to that in Section 6.5.

The Jetbotmini robot color follower has the function of real-time regulation of HSV, controlling the color of the car following detection and recognition. By adjusting the high and low thresholds of HSV, the interference color is filtered out, so that the square can be ideally recognized in a complex environment. If the effect in the color picking is not ideal, you need to move the car to a different environment for calibration at this time, so that we can 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 6.5 for setting, or refer to the following table.



	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

10.2.2、 Steps

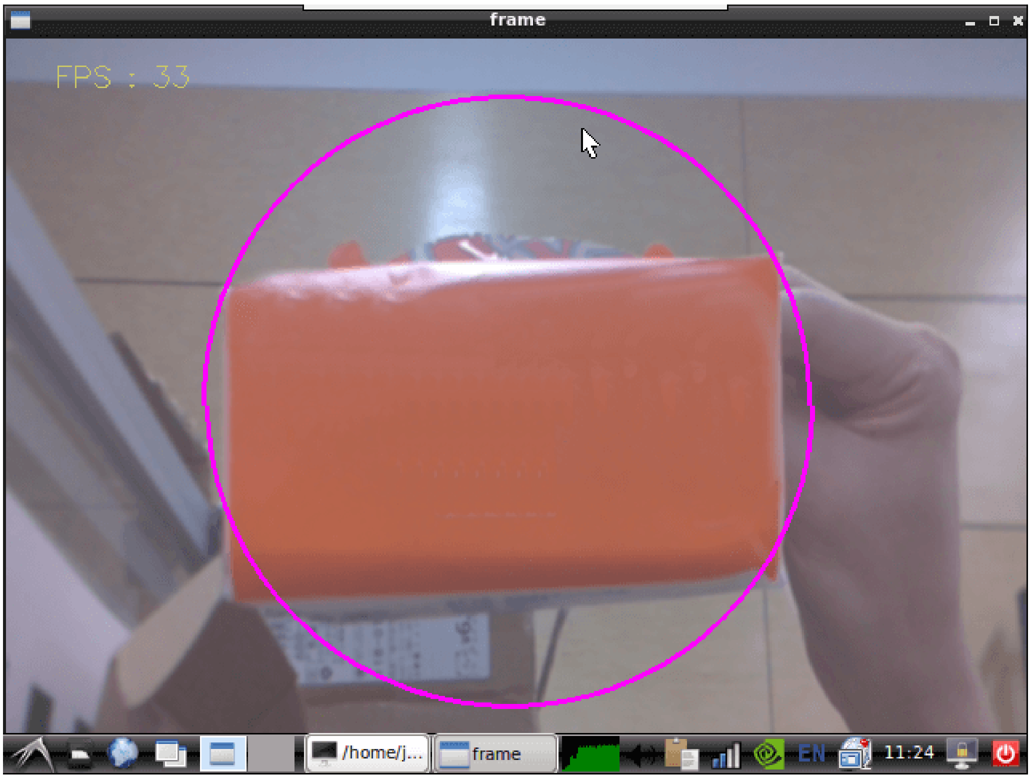
1. start up

Start color follow control.

```
roslaunch jetbot_ros color_tracker.launch
```

2. Identify

Use the preset color for recognition after startup.



Keyboard key control introduction：

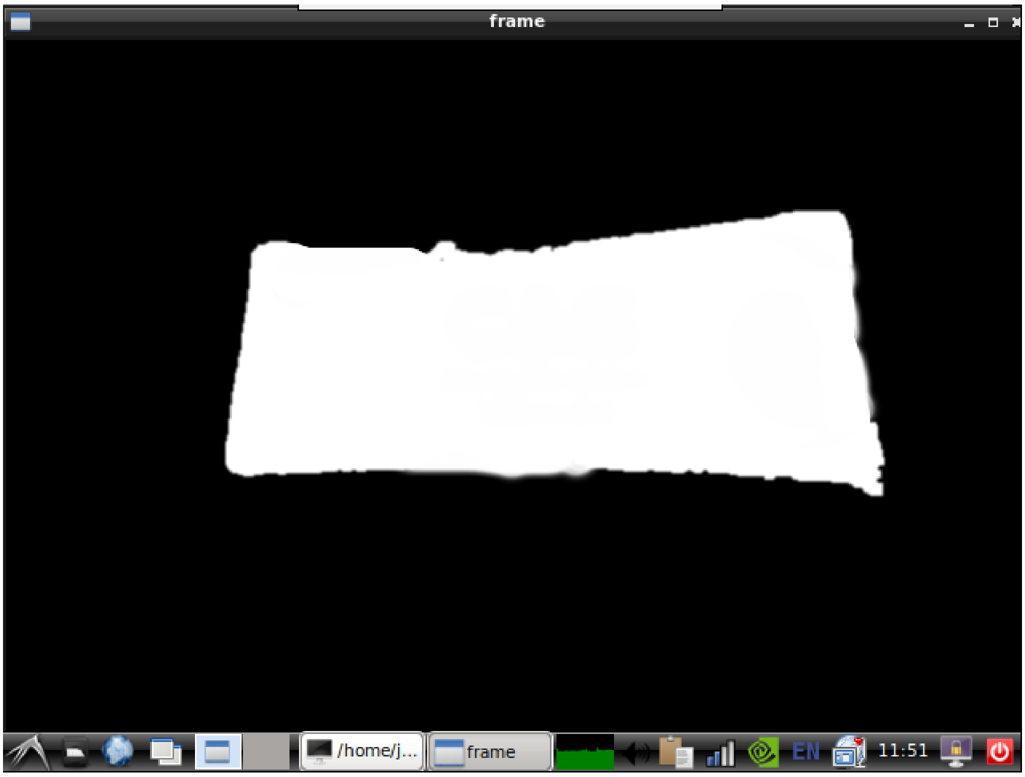
【s】： Start the color to follow.

【i】： Switch to Binary mode.

【Space key】： Switch to Color mode.

【q】： exit the program.

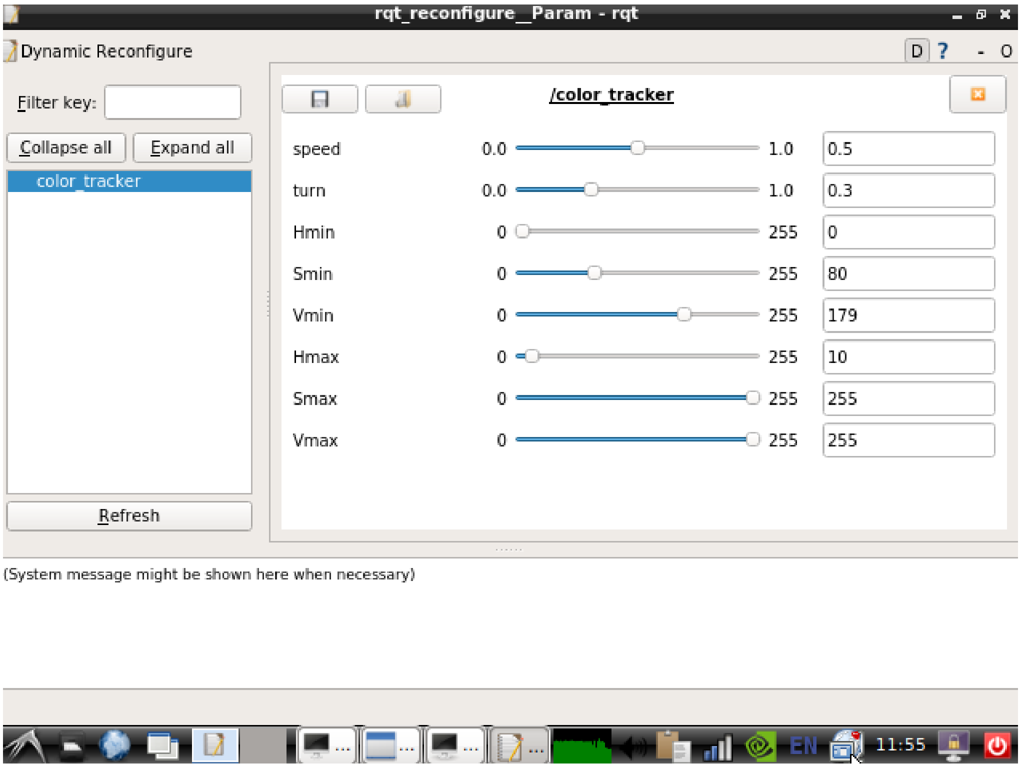
Press the key【i】to switch to Binary mode, the effect is as follows:



3. Color calibration

Dynamic parameter tuning.

```
rosrun rqt_reconfigure rqt_reconfigure
```



speed 0.0 1.0 0.5
turn 0.0 1.0 0.3
Hmin 0 255 0
Smin 0 255 80
Vmin 0 255 179
Hmax 0 255 10
Smax 0 255 255
Vmax 0 255 255

(System message might be shown here when necessary)

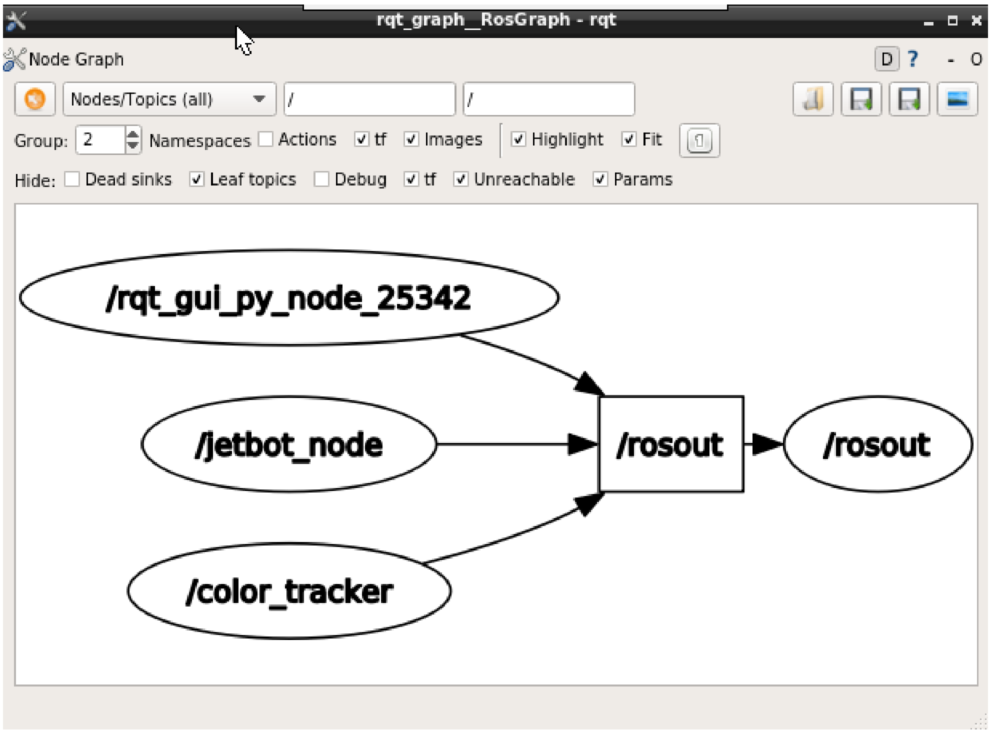
Select the [color_tracker] node, in the binary graph mode, adjust the HSV parameters, you can observe the recognized color changes, only need to adjust [Hmin], [Smin], [Vmin], [Hmax], these four The parameters can be well identified.

Note: The slider is always in the dragging state, and no data will 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 following. If the following 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
```



Nodes/Topics (all) / /

Group: 2 Namespaces ☐ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☐ Dead sinks ☑ Leaf topics ☐ Debug ☑ tf ☑ Unreachable ☑ Params

5. Source code analysis

- launch file

  file path：/home/jetson/workspace/catkin_ws/src/jetbot_ros/launch/color_tracker.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_tracker.py" name="color_tracker"
output="screen">
    </node>
</launch>
```

- python file

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

Recognize the color and convert the recognized 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:
                cv.circle(image,(int(color_x),int(color_y)),int(color_radius),
(255,0,255),2)
                if color_radius > 300:
                    self.Motor_srv(0,0)
                else:
                    center_x = (320 - color_x)/320
                    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
```

Read the real-time modified value of rqt through the following function.

```
    def dynamic_reconfigure_callback(self, config, level):
        print ("dynamic_reconfigure_callback!!!")
        self.speed_value = config['speed']
        self.turn_value = config['turn']
        self.Hmin = config['Hmin']
        self.Smin = config['Smin']
        self.Vmin = config['Vmin']
        self.Hmax = config['Hmax']
        self.Smax = config['Smax']
        self.Vmax = config['Vmax']
        return config
```