10.1、Color recognition

10.1.1、Principle introduction

In section 4.2.6 HSV color space and color space conversion (RGB-HSV), we have already understood the HSV color gamut space. The principle of color recognition is to use the HSV color gamut space of different colors in each frame of image. Classify and mark it out, as in 6.3.2, we first pass

```
cv.cvtColor(image, cv.COLOR_BGR2HSV)
```

Convert RGB to hsv, construct the mask according to the threshold, and after the morphological processing of dilation and corrosion, the mask and the original picture are bitwise AND operated, after finding the color, draw a circle on the outline of the color for labeling, and the threshold The setting of is introduced in 4.2.6, you can also refer to the following table for setting.

	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

10.1.2 、Steps

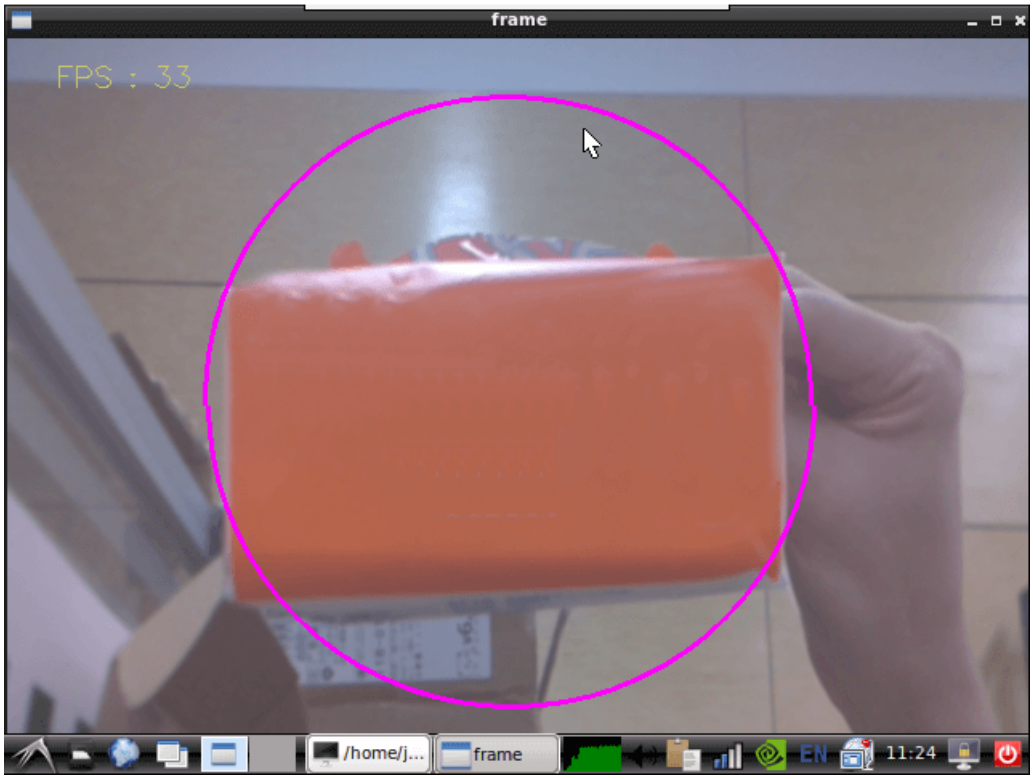
With the theoretical foundation, we will then use ROS to realize color recognition.

1. start up

```
roslaunch jetbot_ros color_recognition.launch
```

2. Identify

Use the preset color for recognition after startup.



Note: press【q】key to exit.

3.Source code analysis

- launch file

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

```
<launch>
    <!--Start the color recognition node-->
    <node pkg="jetbot_ros" type="color_recognition.py" name="color_recognition"
output="screen">
    </node>
</launch>
```

- python file

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

  Modify the following values to change the recognition color.

```
   color_lower = np.array([0,80,179], dtype=np.uint8)
   color_upper = np.array([10, 255, 255], dtype=np.uint8)
```

Identify the color and mark it by the following function.

```
def colorDisplay(image, font_path):
    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 > 10:
            # Mark the detected colors
            cv.circle(image,(int(color_x),int(color_y)),int(color_radius),
(255,0,255),2)
    return image
```