Color recognition

This part is mainly for the preparation of the following functional gameplay. Main steps:

- Convert the RGB image to be detected into an HSV image
- Define an object of Mat type: mask
- Define the upper and lower limits of the color

The upper limit is a Scalar object containing three values: hmin, smin, vmin, indicating the minimum value of the three elements of hsv;

The lower limit is also a Scalar object containing three values: hmax, smax, vmax, indicating the maximum value of the three elements of hsv.

- Use the inRange function to detect whether each pixel of the src image is between lowerb and upperb

If so, the pixel is set to 255 and saved in the mask image, otherwise it is 0.

3.1 Basic principles

The commonly used models in digital image processing are RGB (red, green, blue) model and HSV (hue, saturation, brightness). RGB is widely used in color monitors and color video cameras. Our usual pictures are generally RGB models. The HSV model is more in line with the way people describe and interpret colors. The color description of HSV is natural and very intuitive to people. Another reason for choosing the HSV model is that the RGB channel cannot well reflect the specific color information of the object. Compared with the RGB space, the HSV space can very intuitively express the brightness, hue, and brightness of the color, which is convenient for color comparison.

3.2, HSV model

HSV (Hue, Saturation, Value) is a color space created by A. R. Smith in 1978 based on the intuitive characteristics of color, also known as the Hexcone Model. The color parameters in this model are: hue (H), saturation (S), and brightness (V).

H: 0 — 180

S: 0 — 255

V: 0 — 255

- HSV parameter table:

	Black	Gray	White	Red	Orange	Yellow	Green	Cyan	Blue	Purple	
H_min	0	0	0	0	156	11	26	35	78	100	125
H_max	180	180	180	10	180	25	34	77	99	124	155
S_min	0	0	0	43	43	43	43	43	43	43	
S_max	255	43	30	255	255	255	255	255	255	255	
V_min	0	0	0	46	46	46	46	46	46	46	
V_max	46	220	255	255	255	255	255	255	255	255	

3.3. Main code

Code path:

```
~/jetcobot_ws/src/jetcobot_ai_basic/scripts/3.Color_recognition.ipynb
```

The following code content needs to be executed according to each actual step. It cannot be run all at once. Running the last unit will directly exit the thread.

```
#bgr8 to jpeg format
import enum
import cv2
def bgr8_to_jpeg(value, quality=75):
return bytes(cv2.imencode('.jpg', value)[1])
```

```
#Camera component display
import traitlets
import ipywidgets.widgets as widgets
import time
# Thread function operation library
import threading
import inspect
import ctypes

origin_widget = widgets.Image(format='jpeg', width=320, height=240)
mask_widget = widgets.Image(format='jpeg',width=320, height=240)
result_widget = widgets.Image(format='jpeg',width=320, height=240)

# Create a horizontal box container to place the image widgets next to each other
image_container = widgets.HBox([origin_widget, mask_widget, result_widget])
# image_container = widgets.Image(format='jpeg', width=600, height=500)
display(image_container)
```

Get the hsv value of the color

```
def get_color(img):
    H = []
    color_name={}
    img = cv2.resize(img, (640, 480), )
    # Convert color image to HSV
    HSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # Draw a rectangular box
    cv2.rectangle(img, (280, 180), (360, 260), (0, 255, 0), 2)
    # Take out the H, S, V values ••of each row and column and put them into the
container
    for i in range(280, 360):
    for j in range(180, 260): H.append(HSV[j, i][0])
    # Calculate the maximum and minimum of H, S, and V respectively
    H_min = min(H); H_max = max(H)
    # print(H_min, H_max)
    # judge color
    if H_min >= 0 and H_max <= 10 or H_min >= 156 and H_max <= 180:
color_name['name'] = 'red'
```

```
    elif H_min >= 26 and H_max <= 34: color_name['name'] = 'yellow'
    elif H_min >= 35 and H_max <= 78: color_name['name'] = 'green'
    elif H_min >= 100 and H_max <= 124: color_name['name'] = 'blue'
    return img, color_name
```

Main process: Identify red, green, blue and yellow colors.

```
import cv2
import numpy as np
import ipywidgets.widgets as widgets


cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
cap.set(5, 30)  #设置帧率
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))

color_lower = np.array([0, 43, 46])
color_upper = np.array([10, 255, 255])


def Color_Recongnize():

    while(1):
        # get a frame and show 获取视频帧并转成HSV格式，利用cvtColor()将BGR格式转成HSV格式，参数为cv2.COLOR_BGR2HSV。
        ret, frame = cap.read()
        frame, color_name = get_color(frame)
        if len(color_name)==1:
            global color_lower
            global color_upper

            if color_name['name'] == 'yellow':
                color_lower = np.array([26, 43, 46])
                color_upper = np.array([34, 255, 255])

            elif color_name['name'] == 'red':
                color_lower = np.array([0, 43, 46])
                color_upper = np.array([10, 255, 255])

            elif  color_name['name'] == 'green':
                color_lower = np.array([35, 43, 46])
                color_upper = np.array([77, 255, 255])

            elif color_name['name'] == 'blue':
                color_lower=np.array([100, 43, 46])
                color_upper = np.array([124, 255, 255])

        origin_widget.value = bgr8_to_jpeg(frame)
        #cv2.imshow('Capture', frame)

        # change to hsv model
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

```
        # get mask 利用inRange()函数和HSV模型中蓝色范围的上下界获取mask，mask中原视频中的蓝色部分会被弄成白色，其他部分黑色。
        mask = cv2.inRange(hsv, color_lower, color_upper)
        #cv2.imshow('Mask', mask)
        mask_widget.value = bgr8_to_jpeg(mask)


        # detect blue 将mask于原视频帧进行按位与操作，则会把mask中的白色用真实的图像替换：
        res = cv2.bitwise_and(frame, frame, mask=mask)
        #cv2.imshow('Result', res)
        result_widget.value = bgr8_to_jpeg(res)

        time.sleep(0.01)

    cap.release()
    #cv2.destroyAllWindows()
```

After the program block is run, the camera component is displayed.