



### 3.3 Main code

Code path: /home/yahboom/Dofbot/6.AI\_Visual/5.Color recognition.ipynb

The following code content needs to be executed according to the 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 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 hsv value of 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 frame
    cv2.rectangle(img, (280, 180), (360, 260), (0, 255, 0), 2)
    # Take out the H, S, and V values of each row and column in turn 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 values 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: Recognize 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) #Set frame rate
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'))

# Red is selected by default, and the program will automatically switch colors
# based on the color detected in the box.
# red interval
color_lower = np.array([0, 43, 46])
color_upper = np.array([10, 255, 255])

def Color_Recongnize():

    while(1):
        # get a frame and show Obtain video frames and convert them into HSV
        # format. Use cvtColor() to convert BGR format into HSV format. The parameter is
        # 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 Use the inRange() function and the upper and lower bounds of
the blue range in the HSV model to obtain the mask. The blue part of the original
video in the mask will be made white and the other parts black.
mask = cv2.inRange(hsv, color_lower, color_upper)
#cv2.imshow('Mask', mask)
mask_widget.value = bgr8_to_jpeg(mask)

# detect blue Perform a bitwise AND operation on the mask on the
original video frame, and the white in the mask will be replaced with the real
image:
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, you will see the camera component display.

