Color calibration

By adjusting the high and low thresholds of HSV, interfering colors are filtered out, so that the blocks can be ideally identified in complex environments. It is best to calibrate when using color-based gameplay for the first time.

The values after this color calibration will be saved in a file for easy reading and use by multiple programs. Since the color positioning, color tracking and snake-leading functions are in the front view, the HSV calibration values are different, so there are separate calibration files under the corresponding functions. Other routines that use the top view can use this program for calibration.

Here, the color calibration file in this path is used for calibration:

```
~/home/jetson/dofbot_pro/dofbot_basic_visual/scripts/02.HSV_Calibration.ipynb
```

1. Introduction to HSV

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 value (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	

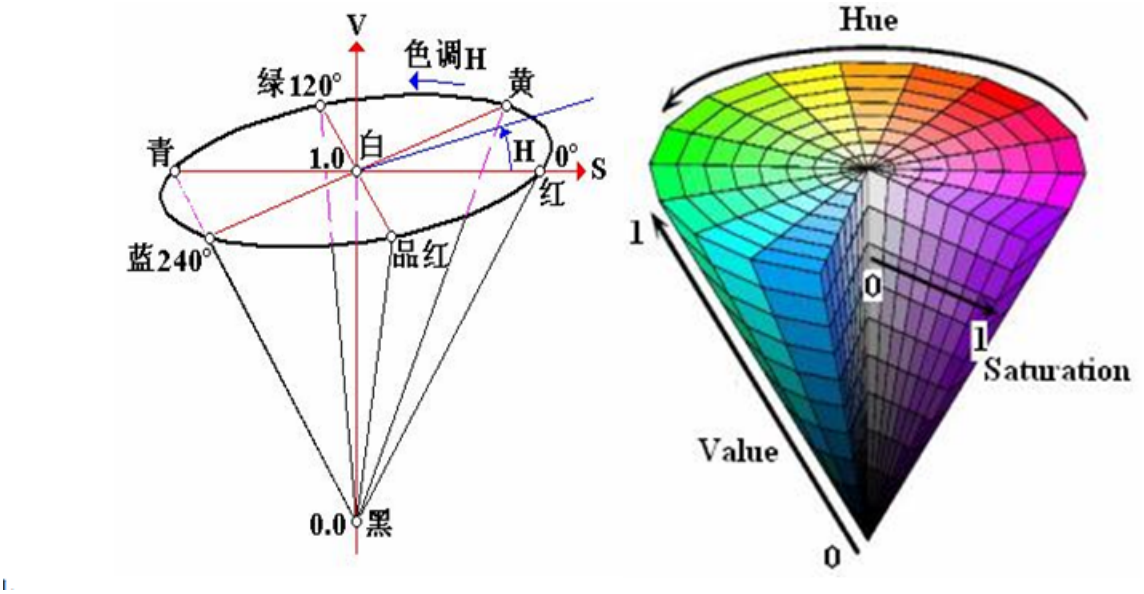
- HSV hexagonal pyramid

(1) Hue H. Represents color information, that is, the position of the spectral color. This parameter is expressed as an angle, ranging from 0° to 360°. Starting from red and counting counterclockwise, red is 0°, green is 120°, and blue is 240°. Their complementary colors are: yellow is 60°, cyan is 180°, and purple is 300°.

(2) Saturation S. Saturation S: It is expressed as the ratio between the purity of the selected color and the maximum purity of the color. When S=0, there is only grayscale. The complementary colors are 120 degrees apart. The complementary colors are 180 degrees apart. A color can be regarded as the result of mixing a certain spectral color with white. The larger the proportion of the spectral color, the closer the color is to the spectral color, and the higher the saturation of the color. The higher the saturation, the deeper and brighter the color. The white light component of

the spectral color is 0, and the saturation reaches the highest. The value range is usually 0% to 100%. The larger the value, the more saturated the color.

(3) Brightness V. Brightness indicates the brightness of the color. For light source color, the brightness value is related to the brightness of the light source; for object color, this value is related to the transmittance or reflectance of the object. The value range is usually 0% (black) to 100% (white). One thing to note is that there is no direct connection between it and light intensity. The three-dimensional representation of the HSV model evolved from the RGB cube. Imagine observing from the white vertex along the diagonal of the RGB cube to the black vertex, you can see the hexagonal shape of the cube. The hexagonal boundary represents the color, the horizontal axis represents the purity, and the brightness is measured along the vertical axis.



2. Code design

- Import header files

```
#!/usr/bin/env python
# coding: utf-8
import threading
import cv2 as cv
import ipywidgets as widgets
from IPython.display import display
from dofbot_utils.dofbot_config import *
```

- Create instances and initialize parameters

```
# Create and update HSV instances
update_hsv = update_hsv()
# Initialize the num parameter
num=0
# Initialization mode
model = "General"
# Initialize HSV name
HSV_name=None
# Initialize HSV value
color_hsv = {"red" : ((0, 43, 46), (10, 255, 255)),
"green" : ((35, 43, 46), (77, 255, 255)),
"blue" : ((100, 43, 46), (124, 255, 255)),
```

```
"yellow": ((26, 43, 46), (34, 255, 255))}
# Set random colors Set random colors
color = [[random.randint(0, 255) for _ in range(3)] for _ in range(255)]
# HSV parameter path HSV parameter path
HSV_path="/home/jetson/dofbot_pro/dofbot_color_identify/scripts/HSV_config.txt"
# Read HSV configuration file, update HSV value Read HSV configuration file,
update HSV value
try: read_HSV(HSV_path,color_hsv)
except Exception: print("Read HSV_config Error!!!")
```

- Initialize the position of the robot arm

```
XYT_path="/home/jetson/dofbot_pro/dofbot_color_identify/scripts/XYT_config.txt"
try: xy, threshold = read_XYT(XYT_path)
except Exception: print("Read XYT_config Error !!!")

import Arm_Lib
Arm = Arm_Lib.Arm_Device()
joints_0 = [xy[0], xy[1], 0, 0, 90, 30]
Arm.Arm_serial_servo_write6_array(joints_0, 1000)
```

- Create controls

```
# 创建布局 Create layout
button_layout      = widgets.Layout(width='260px', height='40px',
align_self='center')
# 输出部件 Output part
output = widgets.Output()
# 进入颜色更新模式 Enter color update mode
HSV_update_red     = widgets.Button(description='HSV_update_red',
 button_style='success', layout=button_layout)
HSV_update_green   = widgets.Button(description='HSV_update_green',
 button_style='success', layout=button_layout)
HSV_update_blue    = widgets.Button(description='HSV_update_blue',
button_style='success', layout=button_layout)
HSV_update_yellow  = widgets.Button(description='HSV_update_yellow',
button_style='success', layout=button_layout)
HSV_write_file     = widgets.Button(description='HSV_write_file',
 button_style='primary', layout=button_layout)
Color_Binary       = widgets.Button(description='Color/Binary',
 button_style='info',    layout=button_layout)
# 调整滑杆 Adjust the slider
H_min_slider       = widgets.IntSlider(description='H_min :', value=0, min=0,
max=255, step=1, orientation='horizontal')
S_min_slider       = widgets.IntSlider(description='S_min :', value=43, min=0,
max=255, step=1, orientation='horizontal')
V_min_slider       = widgets.IntSlider(description='V_min :', value=46, min=0,
max=255, step=1, orientation='horizontal')
H_max_slider       = widgets.IntSlider(description='H_max :', value=10, min=0,
max=255, step=1, orientation='horizontal')
S_max_slider       = widgets.IntSlider(description='S_max :', value=255, min=0,
max=255, step=1, orientation='horizontal')
V_max_slider       = widgets.IntSlider(description='V_max :', value=255, min=0,
max=255, step=1, orientation='horizontal')
```

```
# 退出按钮 Exit button
exit_button = widgets.Button(description='Exit', button_style='danger',
layout=button_layout)
# 图像控件 Image control
imgbox = widgets.Image(format='jpg', height=480, width=640,
layout=widgets.Layout(align_self='center'))
# 调试按钮布局 Debug button layout
HSV_slider = widgets.VBox(
    [H_min_slider, S_min_slider, V_min_slider, H_max_slider, S_max_slider,
V_max_slider, HSV_update_red,
     HSV_update_green, HSV_update_blue, HSV_update_yellow,
Color_Binary,HSV_write_file, exit_button],
    layout=widgets.Layout(align_self='center'))
# 整体布局 overall layout
controls_box = widgets.HBox([imgbox,HSV_slider],
layout=widgets.Layout(align_self='center'))
```

- Color update callback

```
def update_red_Callback(value):
    # 点击红色按钮,回调函数 Click the red button, call back function
    global HSV_name
    HSV_name = "red"
    H_min_slider.value=color_hsv[HSV_name][0][0]
    S_min_slider.value=color_hsv[HSV_name][0][1]
    V_min_slider.value=color_hsv[HSV_name][0][2]
    H_max_slider.value=color_hsv[HSV_name][1][0]
    S_max_slider.value=color_hsv[HSV_name][1][1]
    V_max_slider.value=color_hsv[HSV_name][1][2]
def update_green_Callback(value):
    # 点击绿色按钮,回调函数 Click the green button to call back function
    global HSV_name
    HSV_name = "green"
    H_min_slider.value=color_hsv[HSV_name][0][0]
    S_min_slider.value=color_hsv[HSV_name][0][1]
    V_min_slider.value=color_hsv[HSV_name][0][2]
    H_max_slider.value=color_hsv[HSV_name][1][0]
    S_max_slider.value=color_hsv[HSV_name][1][1]
    V_max_slider.value=color_hsv[HSV_name][1][2]
def update_blue_Callback(value):
    # 点击蓝色按钮,回调函数 Click the blue button to call back function
    global HSV_name
    HSV_name = "blue"
    H_min_slider.value=color_hsv[HSV_name][0][0]
    S_min_slider.value=color_hsv[HSV_name][0][1]
    V_min_slider.value=color_hsv[HSV_name][0][2]
    H_max_slider.value=color_hsv[HSV_name][1][0]
    S_max_slider.value=color_hsv[HSV_name][1][1]
    V_max_slider.value=color_hsv[HSV_name][1][2]
def update_yellow_Callback(value):
    # 点击黄色按钮,回调函数 Click the yellow button to call back function
    global HSV_name
    HSV_name = "yellow"
    H_min_slider.value=color_hsv[HSV_name][0][0]
    S_min_slider.value=color_hsv[HSV_name][0][1]
```

```
    V_min_slider.value=color_hsv[HSV_name][0][2]
    H_max_slider.value=color_hsv[HSV_name][1][0]
    S_max_slider.value=color_hsv[HSV_name][1][1]
    V_max_slider.value=color_hsv[HSV_name][1][2]
# 点击按钮 Click the button
HSV_update_red.on_click(update_red_Callback)
HSV_update_green.on_click(update_green_Callback)
HSV_update_blue.on_click(update_blue_Callback)
HSV_update_yellow.on_click(update_yellow_Callback)
```

- Mode switch controls

```
# 模式设置 Mode setting
def write_file_Callback(value):
    global model
    model = 'Write_file'
def Color_Binary_Callback(value):
    global model,num
    if num%2==0: model="Binary"
    if num%2==1: model="General"
    num+=1
def exit_button_Callback(value):
    global model
    model = 'Exit'
    with output: print(model)
# 点击按钮 Click the button
HSV_write_file.on_click(write_file_Callback)
Color_Binary.on_click(Color_Binary_Callback)
exit_button.on_click(exit_button_Callback)
```

- Interface Example

The upper part of the screen shows which color is selected. The six sliders on the upper right side correspond to the six HSV values. Slide the slider to adjust the HSV threshold of each color in real time.

3. Operation process

(1). After all code blocks are started, the interface shown in the figure is displayed at the bottom of the code. The default color selection is empty, so no color is recognized.

(2). Click the [HSV_update_green] button to start identifying green objects (Note: This color recognition detects the contour, so the object can only be recognized normally when it is completely within the camera range). Slide the slider on the upper right to adjust the green HSV threshold. Pay attention to multi-directional adjustment when adjusting, and adjust in different field of view environments until the object can be clearly identified in a complex environment without being disturbed by other objects. [Other colors are similar].

(3). [Color/Binary] button is the switch between color image and binary image, which is only valid after the color is selected. After switching, only the selected color binary image is displayed, which is more convenient for us to debug.

(4). [HSV_write_file] button, after debugging the HSV values of all colors, click this button. Save all debugged parameters in the format of [.txt] in the path pointed to by the [HSV_path] variable, and automatically read the file parameters next time it is started.

(5). [Exit] button, turn off the camera and exit the program.