Color Calibration

For Orin board users, directly open a web page and enter the IP address:8888 to access jupyter-lab and run directly. For Jetson-Nano board users, you need to first enter the docker container, then enter the following command in docker:

```
cd
jupyter-lab --allow-root
```

Then open a web page and enter the IP address:9999 to access jupyter-lab and run the following program.

By adjusting the HSV high and low thresholds, you can filter out interfering colors, allowing blocks to be ideally recognized in complex environments. When using color-based functions for the first time, it's best to calibrate first.

The calibrated color values will be saved to a file for easy reading and use by multiple programs. Color positioning, color tracking, and snake-leading functions use front-view perspective, so HSV calibration values are different, thus corresponding functions have separate calibration files. Other routines using top-down perspective can use this program for calibration.

Here is the color calibration file path for calibration:

```
#Jetson-Nano users need to enter the docker container to view
~/dofbot_pro/dofbot_basic_visual/scripts/02.HSV_Calibration.ipynb
```

1. HSV Introduction

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

H: 0 — 180

S: 0 — 255

V: 0 — 255

- HSV Parameter Table:

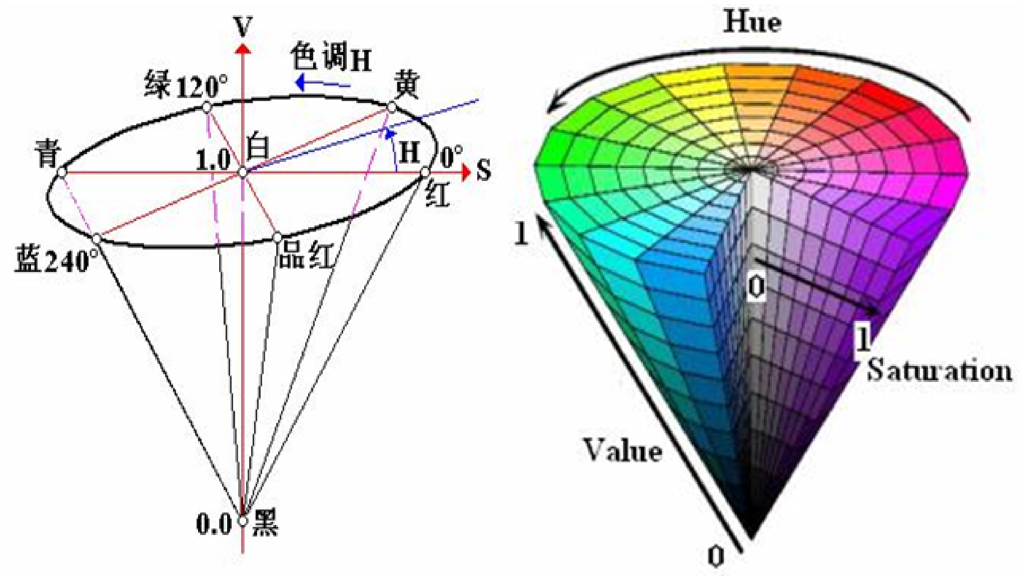
	Black	**Gray**	**White**	**Red**	**Orange**	**Yellow**	**Green**	**Cyan**	**Blue**	**Purple**	
H_min	0	0	0	0	156	11	26	35	78	100	125
H_max	180	180	180	10	180	25	34	77	99	124	155
S_min	0	0	0	43	43	43	43	43	43	43	
S_max	255	43	30	255	255	255	255	255	255	255	
V_min	0	0	0	46	46	46	46	46	46	46	
V_max	46	220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

- HSV Hexagonal Pyramid

(1) Hue H. Represents color information, i.e., the position of the spectral color. This parameter is represented by an angle measure, ranging from 0° to 360°, calculated counterclockwise starting from red. Red is 0°, green is 120°, blue is 240°. Their complementary colors are: yellow is 60°, cyan is 180°, purple is 300°.

(2) Saturation S. Saturation S: represents the ratio between the purity of the selected color and the maximum purity of that color. When S=0, there is only grayscale. Complementary colors are 120° apart. Complementary colors differ by 180°. A color can be seen as the result of mixing a certain spectral color with white. The larger the proportion of spectral color, the closer the color is to the spectral color, and the higher the saturation. High saturation means the color is deep and vivid. The white light component of spectral colors is 0, reaching maximum saturation. The usual range is 0% to 100%, with larger values indicating more saturated colors.

(3) Value V. Value represents the brightness of the color. For light source colors, the value is related to the brightness of the emitter; for object colors, this value is related to the transmittance or reflectance of the object. The usual range is 0% (black) to 100% (white). One thing to note: there is no direct relationship between it and light intensity. The 3D representation of the HSV model evolves from the RGB cube. Imagine viewing from the white vertex to the black vertex along the diagonal of the RGB cube, and you can see the hexagonal appearance of the cube. The hexagonal boundary represents color, the horizontal axis represents purity, and value 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 Instance, Initialize Parameters

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

- Initialize Robotic Arm Position

```
#Orin board
XYT_path="/home/jetson/dofbot_pro/dofbot_color_identify/scripts/XYT_config.txt"
#Jetson-Nano board
XYT_path="/root/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

```
# Create layout
button_layout      = widgets.Layout(width='260px', height='40px',
align_self='center')
# Output part
output = widgets.Output()
# 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)
# 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')
# Exit button
exit_button = widgets.Button(description='Exit', button_style='danger',
layout=button_layout)
# Image control
imgbox = widgets.Image(format='jpg', height=480, width=640,
layout=widgets.Layout(align_self='center'))
# 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'))
# overall layout
controls_box = widgets.HBox([imgbox,HSV_slider],
layout=widgets.Layout(align_self='center'))
```

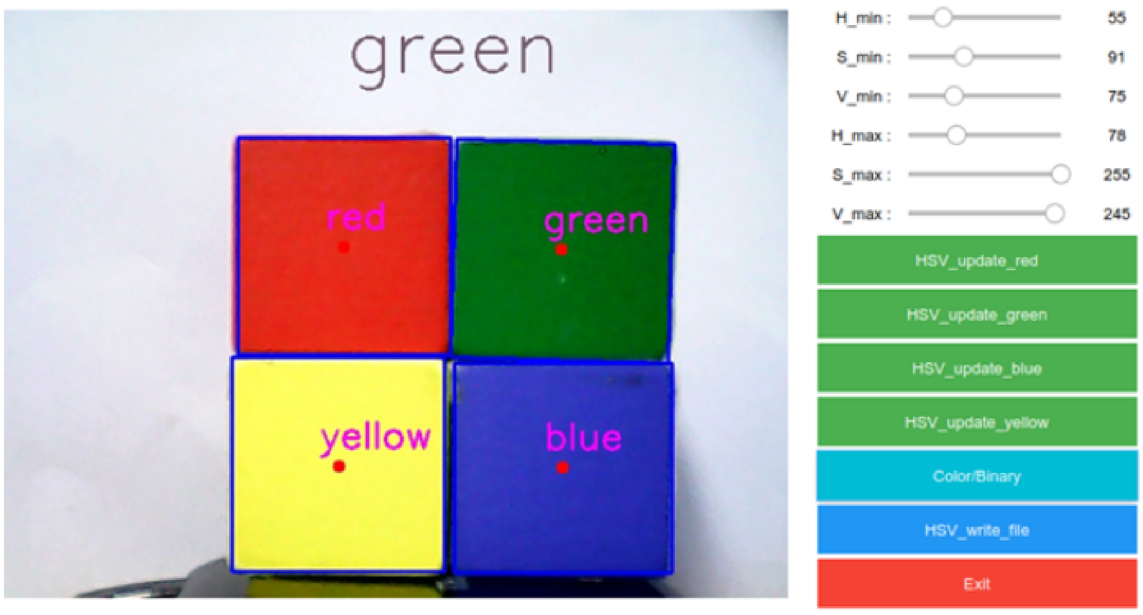
- Color Update Callback

```
def update_red_Callback(value):
    # Click the red button, callback 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):
    # Click the green button callback 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):
    # Click the blue button callback 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):
    # Click the yellow button callback 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]
# 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 Switching Controls

```
# 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)
# 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



H_min : 55
S_min : 91
V_min : 75
H_max : 78
S_max : 255
V_max : 245

The upper middle of the screen displays which color is selected, and the six sliders in the upper right correspond to the six HSV values. Slide the sliders to adjust the HSV threshold of each color in real time.

3. Operation Process

(1). After starting all code blocks, the interface shown in the figure will be displayed at the bottom of the code. By default, no color is selected, so no colors are recognized.

(2). Click the [HSV_update_green] button to start recognizing green objects (Note: this color recognition detects contours, so the object must be completely within the camera range for normal recognition). Slide the sliders in the upper right to adjust the green HSV threshold. When adjusting, pay attention to multi-directional adjustment and adjust in different viewing environments until the object can be clearly recognized in complex environments without interference from other objects [Other colors are similar].

(3). The [Color/Binary] button switches between color image and binary image, and is only effective after selecting a color. After switching, only the selected color binary image is displayed, making it more convenient for us to debug.

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

(5). The [Exit] button closes the camera and exits the program.