Color threshold adjustment color patch 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. When using color-based gameplay for the first time, it is best to calibrate it. Each color-related AI gameplay has its own [HSV calibration.ipynb] file under the [scripts] folder.

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

/root/dofbot_ws/src/dofbot_color_identify/scripts/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 parameters of color in this model are: hue (H), saturation (S), and lightness (V).

H: 0 — 180

S: 0 — 255

V: 0 — 255

- HSV parameter table:

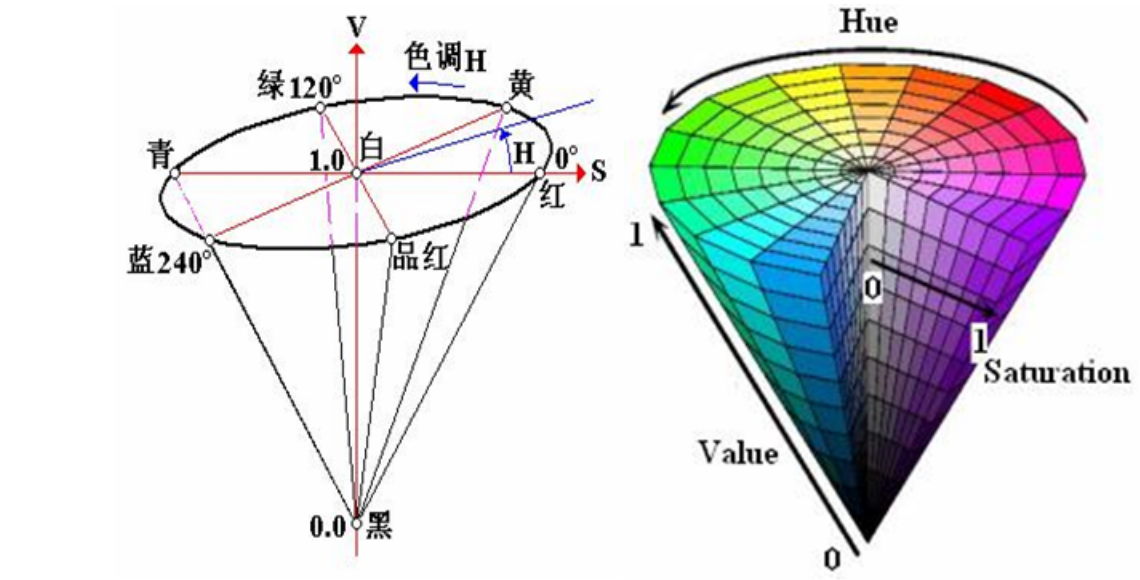
	black	**gray**	**white**	**red**	**orange**	**yellow**	**green**	**cyan**	**blue**	**purple**	
H_min	0	0	0	0	156	11	26	35	78	100	125
H_max	180	180	180	10	180	25	34	77	99	124	155
S_min	0	0	0	43	43	43	43	43	43	43	
S_max	255	43	30	255	255	255	255	255	255	255	
V_min	0	0	0	46	46	46	46	46	46	46	
V_max	46	220	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 represented by an angle, with a value ranging from 0° to 360°, starting from red and counting in counterclockwise direction. 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: 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. 120 degrees apart. Complimentary colors are 180 degrees apart. A color can be thought of as the result of mixing a certain spectral color with white. The greater the proportion of spectral colors, the closer the color is to spectral colors, and the higher the saturation of the color. The saturation is high and the color is deep and vivid. The white light component of the spectral color is 0, and the saturation reaches the highest level. Usually the value range is 0% ~ 100%. The larger the value, the more saturated the color.

(3) Brightness V. Brightness represents the brightness of a color. For light source color, the brightness value is related to the brightness of the luminous body; for object color, this value is related to the transmittance or reflectance of the object. Usually the value range is 0% (black) to 100% (white). One thing to note: there is no direct relationship between it and light intensity. The three-dimensional representation of the HSV model evolves from the RGB cube. If you imagine looking from the white vertices of the RGB along the diagonal of the cube to the black vertices, you can see the hexagonal shape of the cube. The hexagonal borders represent color, the horizontal axis represents purity, and lightness is measured along the vertical axis.



2. Code design

- Import header files

```
import threading
import cv2 as cv
from time import sleep
from dofbot_config import *
import ipywidgets as widgets
from IPython.display import display
```

- Create an instance and initialize parameters

```
#Create and update HSV instance
update_hsv = update_hsv()
#Initialize num parameter
num=0
#Initialization mode
model = "General"
#Initialize HSV name
HSV_name=None
#Initialize HSV value
color_hsv = {"red" : ((0, 143, 163), (11, 255, 255)),

         "green" : ((55, 80, 66), (78, 255, 255)),

         "blue" : ((110, 100, 121), (117, 255, 255)),

         "yellow": ((26, 100, 91), (32, 255, 255))}
```

```
# HSV parameter path (jupyter lab and ros function packages need to use absolute paths)
HSV_path="/root/dofbot_ws/src/dofbot_color_identify/scripts/HSV_config.txt"
# Read the HSV configuration file and update the HSV value
try: read_HSV(HSV_path,color_hsv)
except Exception: print("No HSV_config file!!!")
```

- Create controls

```
button_layout = widgets.Layout(width='320px', height='55px', align_self='center')
output = widgets.Output()
# Enter color update mode
HSV_update_red= widgets.Button(description='HSV_update_red',
button_style='success',layout=button_layout)
...
#Adjust slider

H_min_slider=widgets.IntSlider(description='H_min
:',value=0,min=0,max=255,step=1, orientation='horizontal')
...
# quit
exit_button=widgets.Button(description='Exit',button_style='danger',layout=button
_layout)
```

- Color update callback

```
def update_red_Callback(value):
     pass

def update_green_Callback(value):
     pass

def update_blue_Callback(value):
     pass

def update_yellow_Callback(value):
     pass

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 control

```
def write_file_Callback(value):
    pass

def Color_Binary_Callback(value):
    pass

def exit_button_Callback(value):
    pass

HSV_write_file.on_click(write_file_Callback)
Color_Binary.on_click(Color_Binary_Callback)
exit_button.on_click(exit_button_Callback)
```

```
For detailed code, see /root/dofbot_ws/src/dofbot_color_identify/scripts/HSV calibration.ipynb
```

- Interface examples

The upper middle part of the screen displays which color is selected. The six slide bars on the upper right side correspond to the six HSV values. Slide the slide bars to adjust the HSV threshold of each color in real time.

3. Operation process

(1). After starting all code blocks, the interface as 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 outline, so the object can be recognized normally only when it is completely within the range of the camera), slide the slider on the upper right to adjust the green color When adjusting the HSV threshold, pay attention to multi-directional adjustment and adjust it in different visual field environments until objects can be clearly identified in complex environments without being interfered by other objects [similar to other colors].

(3). The [Color/Binary] button switches between color images and binary images. It is only effective when the color is selected. After switching, only the selected color binary image is displayed, which makes it easier for us to debug.

(4). [HSV_write_file] button, click this button after debugging the HSV values of all colors. Save all the debugged parameters in the format of [.txt] in the same path of the code, and start automatically reading the file next time parameter.

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