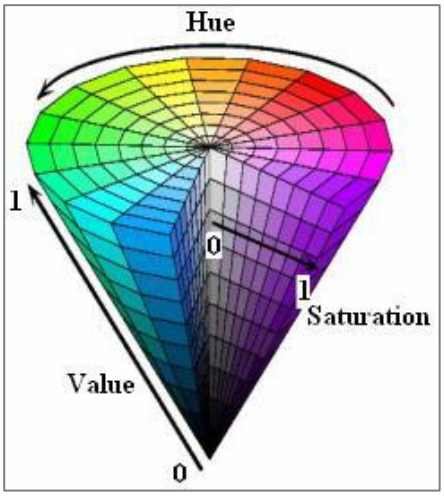
By adjusting the thresholds of HSV, the interference color is filtered out, so that the target color object can be identified in a complex environment.
Before we use the functions related to color recognition, we need to perform color calibration.

1. Introduction of HSV

The color parameters 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



2. About code

Path: dofbot_ws/src/dofbot_color_identify/scripts/HSV_calibration.ipynb

- Import header file

```
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, 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_ws/src/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!!!")
```

- Create control

```
# 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):
    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)
```

界面示例

屏幕中上方显示选择哪种颜色,右侧上方的六个滑动条分别对应着 HSV 的六个数值,滑动滑动条,实时调整每种颜色的 HSV 阈值,

3.操作流程

(1).启动完所有代码块后,在代码的最下方显示如图所示的界面,默认颜色选择为空,故不识别任何颜色.

(2).点击【HSV_update_green】按钮,便开始识别绿色的物体(注:此颜色识别,检测的是轮廓,故物体完全在摄像头范围内才可正常识别),滑动右上方的滑动条进行调节绿色的 HSV 阈值,调节时要注意多方位调节,在不同的视野环境下调节,直到复杂环境下能够清晰的识别出物体,不被其他物体所干扰为止【其他颜色类似】.

(3).【Color/Binary】按钮是彩色图和二值图的切换,只有在选择完颜色的情况下才有效.切换后只显示选择的颜色二值图像,更便我们调试.
(4).【HSV_write_file】按钮,调试完所有颜色的 HSV 值后,点击此按钮.将调试好的所有参数以【.txt】的格式保存在代码同路径下,下次启动自动读取文件的参数.
(5).【Exit】按钮,关闭摄像头,退出程序.