3.HSV value test

3.1. Find the appropriate HSV value by adjusting the slider

Source code path:

/home/pi/project_demo/07.AI_Visual_Recognition/02.Color_Recog/03_color_recognition+hsv.ipynb

```
#导入Raspbot驱动库及相关库 Import the Raspbot library and related libraries
from Raspbot_Lib import Raspbot
import sys
sys.path.append('/home/pi/software/oled_yahboom/')
from yahboom_oled import *
# 创建Raspbot对象 bot Create the Raspbot object bot
bot = Raspbot()
# 创建oled对象 Create an oled object
oled = Yahboom_OLED(debug=False)
#bgr8转jpeg格式 bgr8 to jpeg format
import enum
import cv2

def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

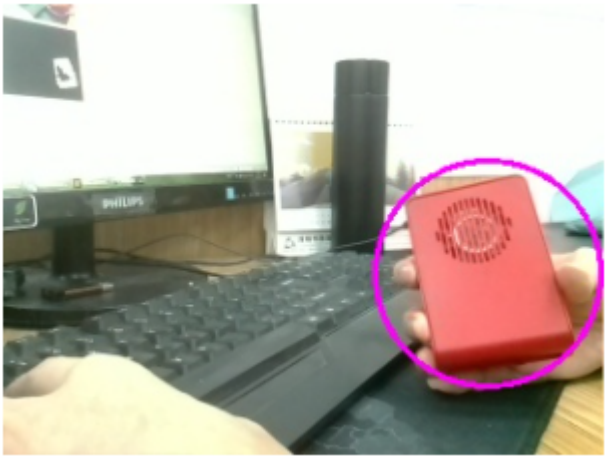
```
#显示摄像头组件 Display camera components
import cv2
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
import time
# 线程功能操作库 Thread function operation library
import threading
import inspect
import ctypes
```

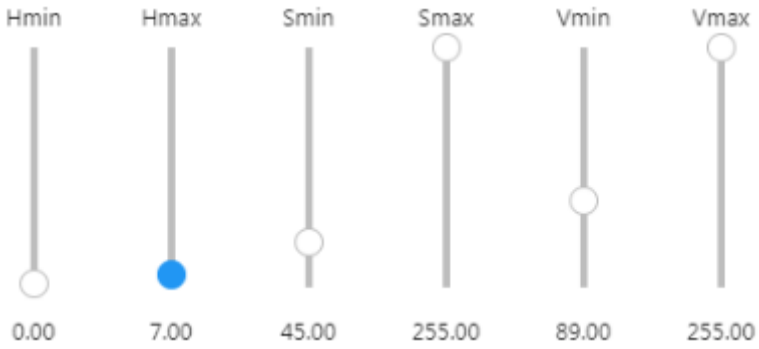
```
def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    tid = ctypes.c_long(tid)
    if not inspect.isclass(exctype):
        exctype = type(exctype)
    res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,
ctypes.py_object(exctype))
    if res == 0:
        raise ValueError("invalid thread id")
    elif res != 1:
```

```
        # """if it returns a number greater than one, you're in trouble, 如果它返回一个大于 1 的数字，你就有麻烦了，
        # and you should call it again with exc=NULL to revert the effect 你应该再次调用它并设置 exc=NULL 来恢复效果"""
        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)

def stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```
image_widget = widgets.Image(format='jpeg', width=300, height=300)
image_widget1 = widgets.Image(format='jpeg', width=300, height=300)
```

```
image = cv2.VideoCapture(0) #打开摄像头/dev/video0 Open the camera /dev/video0

image.set(3, 320)
image.set(4, 240)
image.set(5, 30)  #设置帧率 Setting the frame rate
# image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
# image.set(cv2.CAP_PROP_BRIGHTNESS, 62) #设置亮度 -64 - 64  0.0 Set Brightness -64 - 64 0.0
# image.set(cv2.CAP_PROP_CONTRAST, 63) #设置对比度 -64 - 64  2.0 Set Contrast -64 - 64 2.0
# image.set(cv2.CAP_PROP_EXPOSURE, 4800) #设置曝光值 1.0 - 5000  156.0 Set the exposure value 1.0 - 5000 156.0

ret, frame = image.read()
image_widget.value = bgr8_to_jpeg(frame)

# from picamera2 import Picamera2, Preview
# import libcamera
# picam2 = Picamera2()
# camera_config = picam2.create_preview_configuration(main={"format":'RGB888',"size":(320,240)})
# camera_config["transform"] = libcamera.Transform(hflip=1, vflip=1)
# picam2.configure(camera_config)
# picam2.start()
# frame = picam2.capture_array()

image_widget.value = bgr8_to_jpeg(frame)
display(image_widget)
display(image_widget1)
```





It can be seen that by adjusting the H (hue) S (saturation) V (brightness) values in the default HSV range of red to find a more appropriate HSV value, the recognition accuracy is higher than the previous default HSV value. This is very important in subsequent color tracking. If the color recognition is inaccurate, it always recognizes errors or recognizes other irrelevant things, then the tracking will definitely be a problem.

3.2. Use good HSV values to find button selection color recognition

```
#转向PID输出值 Steering PID output value
Hmin = widgets.FloatSlider(description='Hmin', min=0, max=180, step=1,
orientation='Vertical')
Hmax = widgets.FloatSlider(description='Hmax', min=0, max=179, step=1,
orientation='Vertical')
Smin = widgets.FloatSlider(description='Smin', min=0, max=255, step=1,
orientation='Vertical')
Smax = widgets.FloatSlider(description='Smax', min=0, max=255, step=1,
orientation='Vertical')
Vmin = widgets.FloatSlider(description='Vmin', min=0, max=255, step=1,
orientation='Vertical')
Vmax = widgets.FloatSlider(description='Vmax', min=0, max=255, step=1,
orientation='Vertical')
# create a horizontal box container to place the sliders next to eachother
slider_container = widgets.HBox([Hmin,Hmax,Smin,Smax,Vmin,Vmax])
# display the container in this cell's output
display(slider_container)
import numpy as np
```

```
color_hsv  = {"red"   : ((0, 70, 72), (7, 255, 255)),
              "green" : ((54, 109, 78), (77, 255, 255)),
              "blue"  : ((92, 100, 62), (121, 251, 255)),
              "yellow": ((26, 100, 91), (32, 255, 255))}
```

```
color = "red"
Hmin.value=color_hsv[color][0][0]
Smin.value=color_hsv[color][0][1]
Vmin.value=color_hsv[color][0][2]
Hmax.value=color_hsv[color][1][0]
Smax.value=color_hsv[color][1][1]
Vmax.value=color_hsv[color][1][2]
```

```
def TEST():
    while True:
        ret, frame = image.read() #USB摄像头 USB Camera
        # frame = picam2.capture_array() #CSI摄像头 CSI Camera
        color_lower =
np.array([int(Hmin.value),int(Smin.value),int(Vmin.value)])
        color_upper = np.array([int(Hmax.value), int(Smax.value),
int(Vmax.value)])
        #frame = cv2.resize(frame, (400, 400))
        frame_ = cv2.GaussianBlur(frame,(5,5),0)
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv,color_lower,color_upper)
        mask = cv2.erode(mask,None,iterations=2)
        mask = cv2.dilate(mask,None,iterations=2)
        mask = cv2.GaussianBlur(mask,(3,3),0)
        cnts =
cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
        if 1: # 按钮切换开关 Push button switch
            if len(cnts) > 0:
                cnt = max (cnts, key = cv2.contourArea)
                (color_x,color_y),color_radius = cv2.minEnclosingCircle(cnt)
```

```
                if color_radius > 10:
                # 将检测到的颜色用原形线圈标记出来 Mark the detected color with a
prototype circle
                    cv2.circle(frame,
(int(color_x),int(color_y)),int(color_radius),(255,0,255),2)
                    # Proportion-Integration-Differentiation
        # 实时传回图像数据进行显示 Real-time image data transmission for display
        image_widget.value = bgr8_to_jpeg(frame)
        image_widget1.value = bgr8_to_jpeg(mask)
        # print(g_mode)
```

```
thread1 = threading.Thread(target=TEST)
thread1.daemon=True
thread1.start()
```

```
stop_thread(thread1)
```

Because of the thread opened above, you need to close the previous thread before you can run the following program.

```
global g_mode
g_mode = 0
```

```
import numpy as np
global color_lower
color_lower = np.array([0, 43, 46])
global color_upper
color_upper = np.array([10, 255, 255])
```

```
Redbutton = widgets.Button(
    value=False,
    description='Red',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Greenbutton = widgets.Button(
    value=False,
    description='Green',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Bluebutton = widgets.Button(
    value=False,
    description='Blue',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Yellowbutton = widgets.Button(
    value=False,
```

```
    description='Yellow',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Orangebutton = widgets.Button(
    value=False,
    description='Orange',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Closebutton = widgets.Button(
    value=False,
    description='OFF',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
output = widgets.Output()
display(Redbutton, Greenbutton, Bluebutton, Yellowbutton, Orangebutton,
Closebutton, output)

def ALL_Uncheck():
    Redbutton.icon = 'uncheck'
    Greenbutton.icon = 'uncheck'
    Bluebutton.icon = 'uncheck'
    Yellowbutton.icon = 'uncheck'
    Orangebutton.icon = 'uncheck'

def on_Redbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower = np.array([0, 43, 46])
    #color_upper = np.array([10, 255, 255])
    color_lower = np.array([0,43,89])
    color_upper = np.array([7, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,0)#红色 red
        oled.clear()
        oled.add_line("color: red", 2)
        oled.refresh()
        print("RedButton clicked.")

def on_Greenbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower = np.array([35, 43, 46])
    #color_upper = np.array([77, 255, 255])
    color_lower = np.array([54,104,64])
```

```
    color_upper = np.array([78, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,1)#绿色 green
        oled.clear()
        oled.add_line("color: green", 2)
        oled.refresh()
        print("GreenButton clicked.")

def on_Bluebutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower=np.array([100, 43, 46])
    #color_upper = np.array([124, 255, 255])
    color_lower = np.array([92,100,62])
    color_upper = np.array([121, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,2)#蓝色 blue
        oled.clear()
        oled.add_line("color: blue", 2)
        oled.refresh()
        print("Bluebutton clicked.")

def on_Yellowbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower = np.array([26, 43, 46])
    #color_upper = np.array([34, 255, 255])
    color_lower = np.array([26,100,91])
    color_upper = np.array([32, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,3)#黄色 yellow
        oled.clear()
        oled.add_line("color: yellow", 2)
        oled.refresh()
        print("Yellowbutton clicked.")

def on_Orangebutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    color_lower = np.array([11, 43, 46])
    color_upper = np.array([25, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_brightness_ALL(255, 48, 0)
        oled.clear()
        oled.add_line("color: orange", 2)
```

```
        oled.refresh()
        print("Orangebutton clicked.")

def on_Closebutton_clicked(b):
    global g_mode
    ALL_Uncheck()
    g_mode = 0
    with output:
        bot.Ctrl_WQ2812_ALL(0,0)
        oled.clear()
        oled.add_line("color: none", 2)
        oled.refresh()
        print("CloseButton clicked.")
display(image_widget)
display(image_widget1)
Redbutton.on_click(on_Redbutton_clicked)
Greenbutton.on_click(on_Greenbutton_clicked)
Bluebutton.on_click(on_Bluebutton_clicked)
Yellowbutton.on_click(on_Yellowbutton_clicked)
Orangebutton.on_click(on_Orangebutton_clicked)
Closebutton.on_click(on_Closebutton_clicked)
```

```
def Color_Recongnize():
    oled.init_oled_process() #初始化oled进程 Initialize oled process
    global color_lower, color_upper, g_mode
    t_start = time.time()
    fps = 0
    while True:
        ret, frame = image.read()#USB摄像头 USB Camera
        # frame = picam2.capture_array() #CSI摄像头 CSI Camera
        #frame = cv2.resize(frame, (400, 400))
        frame_ = cv2.GaussianBlur(frame,(5,5),0)
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv,color_lower,color_upper)
        mask = cv2.erode(mask,None,iterations=2)
        mask = cv2.dilate(mask,None,iterations=2)
        mask = cv2.GaussianBlur(mask,(3,3),0)
        cnts =
cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
        if g_mode == 1: # 按钮切换开关 Push button switch
            if len(cnts) > 0:
                cnt = max (cnts, key = cv2.contourArea)
                (color_x,color_y),color_radius = cv2.minEnclosingCircle(cnt)
                if color_radius > 10:
                # 将检测到的颜色用原形线圈标记出来 Mark the detected color with a
prototype circle
                    cv2.circle(frame,
(int(color_x),int(color_y)),int(color_radius),(255,0,255),2)
                    # Proportion-Integration-Differentiation
        fps = fps + 1
        mfps = fps / (time.time() - t_start)
        cv2.putText(frame, "FPS " + str(int(mfps)), (40,40),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        # 实时传回图像数据进行显示 Real-time image data transmission for display
        image_widget.value = bgr8_to_jpeg(frame)
```

```
        image_widget1.value = bgr8_to_jpeg(mask)
        # print(g_mode)
```

```
thread1 = threading.Thread(target=Color_Recongnize)
thread1.daemon=True
thread1.start()
```

```
stop_thread(thread1)
# 恢复屏幕基础数据显示 Restore basic data display on screen
os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
```

```
#usb摄像头 usb camera
#After using the object, remember to release the object, otherwise the object
module will be occupied by the next program, making it unusable
#使用完成对象记住释放掉对象，不然下一个程序使用这个对象模块会被占用，导致无法使用
image.release()
#CSI摄像头 CSI Camera
# picam2.stop()
# picam2.close()
```

After we debug HSV, we can do color recognition. When doing color recognition, we can see the color recognized by oled print. The RGB light bar shows the recognized color.

Advanced gameplay: The above are all single specific color recognition. You can try to recognize several colors and select them separately. The following provides examples for recognizing and selecting four colors: red, green, blue and yellow.

Source code path:
/home/pi/project_demo/07.AI_Visual_Recognition/02.Color_Rec/color_detection.ipynb