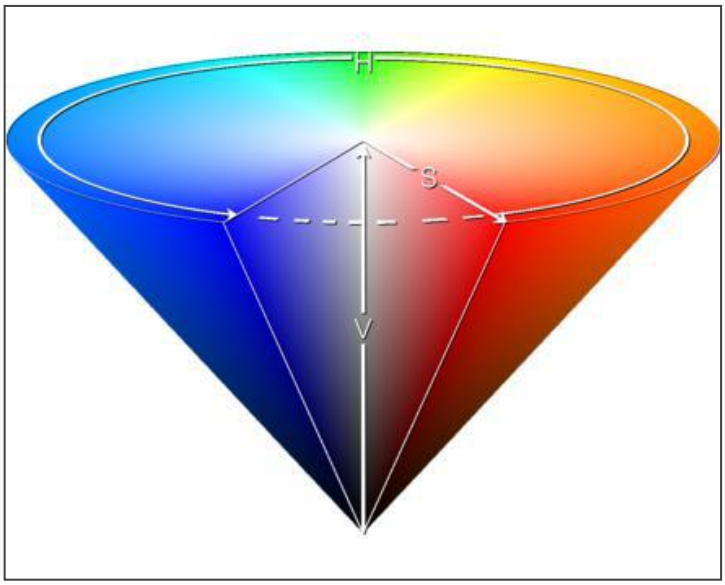
Color recognition

1. Introduction to HSV color space

HSV(Hue, Saturation, Brightness Value) is a color space created based on the intuitive characteristics of color, also be called hexagonal cone model.



【HSV color space model】

2. HSV color space model(gray BGR HSV)

In OpenCV, we often use only two color space conversion methods, namely BGR-> Gray and BGR-> HSV.

!Note: Gray and HSV cannot be converted to each other.

Color space conversion function: cv2.cvtColor (input_image, flag)

BGR-> Gray: flag is cv2.COLOR_BGR2GRAY

BGR-> HSV: flag is cv2.COLOR_BGR2HSV

Value range of HSV color space in OpenCV:

H--[0, 179] S --[0, 255] V--[0, 255]

	black	gray	white	red		orange	yellow	green	verdant	blue	purple
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

【Range of commonly used colors】

Code path:

/home/pi/Yahboom_Project/Raspbot/3.AI Vision course/02.Color recognition

```
import enum
import cv2

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

```
import cv2
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
import time

import threading
import inspect
import ctypes

image_widget = widgets.Image(format='jpeg', width=320, height=240)
display(image_widget)
```

```
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,
        # and you should call it again with exc=NULL to revert the effect"""
        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)

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

```
image = cv2.VideoCapture(0)
image.set(3, 320)
image.set(4, 240)
image.set(5, 30)
image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
image.set(cv2.CAP_PROP_BRIGHTNESS, 62)
image.set(cv2.CAP_PROP_CONTRAST, 63)
image.set(cv2.CAP_PROP_EXPOSURE, 4800)
ret, frame = image.read()
image_widget.value = bgr8_to_jpeg(frame)
```

```
import numpy as np
```

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

```
def Color_Recongnize():
    t_start = time.time()
    fps = 0
    while True:
        ret, frame = image.read()
        frame = cv2.resize(frame, (320, 240))
        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,(5,5),0)
        cnts =
cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
        if len(cnts) > 0:
            cnt = max (cnts, key = cv2.contourArea)
            (color_x,color_y),color_radius = cv2.minEnclosingCircle(cnt)
            if color_radius > 10:
                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)
        image_widget.value = bgr8_to_jpeg(frame)
        # print(g_mode)
```

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

```
stop_thread(thread1)
image.release()
#After using, we need to release the object, otherwise it will be occupied and cannot be used in th
e next program
```


FPS 31