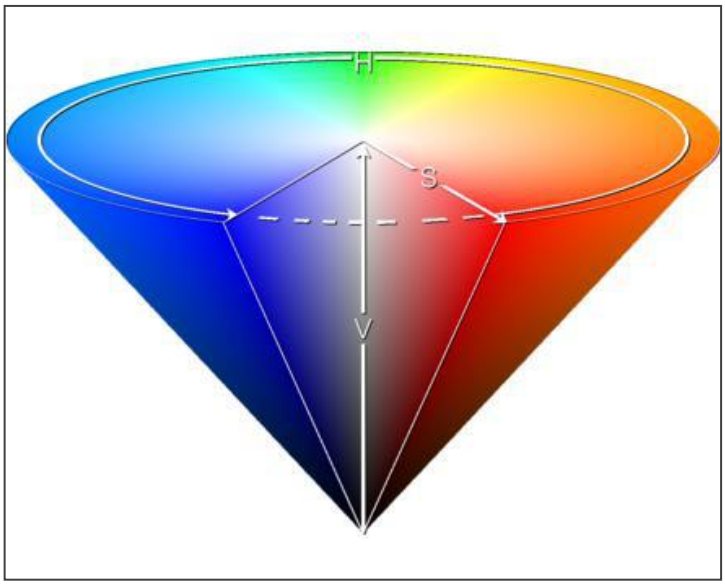
3.1.2 Color recognition

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/2.AI_Visual_course/02.Color_recognition.ipynb

```
#bgr8 to jpeg format
import enum
import cv2

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

```
# Camera component display
import traitlets
import ipywidgets.widgets as widgets
import time
# Thread function operation library
import threading
import inspect
import ctypes

origin_widget = widgets.Image(format='jpeg', width=320, height=240)
mask_widget = widgets.Image(format='jpeg',width=320, height=240)
result_widget = widgets.Image(format='jpeg',width=320, height=240)

# create a horizontal box container to place the image widget next to eachother
image_container = widgets.HBox([origin_widget, mask_widget, result_widget])

display(image_container)
```

```
# Thread related functions
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)
```

```
# Main process function
import cv2
import numpy as np
import ipywidgets.widgets as widgets
```

```
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
cap.set(5, 120)    # Set the frame rate
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
# image.set(cv2.CAP_PROP_BRIGHTNESS, 40) # set brightness -64 - 64    0.0
# image.set(cv2.CAP_PROP_CONTRAST, 50)    #set contrast -64 - 64    2.0
# image.set(cv2.CAP_PROP_EXPOSURE, 156)    #set exposure1.0 - 5000    156.0

# The default selection is red, if you want to recognize other colors, please close the code in the red section, and open the code section of the other section
# Red section
color_lower = np.array([0, 43, 46])
color_upper = np.array([10, 255, 255])

# Green section
# color_lower = np.array([35, 43, 46])
# color_upper = np.array([77, 255, 255])

# Blue section
# color_lower=np.array([100, 43, 46])
# color_upper = np.array([124, 255, 255])

# Yellow section
# color_lower = np.array([26, 43, 46])
# color_upper = np.array([34, 255, 255])

# Orange section
# color_lower = np.array([11, 43, 46])
# color_upper = np.array([25, 255, 255])

def Color_Recongnize():
    while(1):
        # get a frame and show Get the video frame and convert it to HSV format, use cvtColor () to convert the BGR format to HSV format, the parameter is cv2.COLOR_BGR2HSV.
        ret, frame = cap.read()
        #cv2.imshow('Capture', frame)
        origin_widget.value = bgr8_to_jpeg(frame)

        # change to hsv model
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        # Get mask
```

```
            mask = cv2.inRange(hsv, color_lower, color_upper)
            #cv2.imshow('Mask', mask)
            mask_widget.value = bgr8_to_jpeg(mask)

            # Detect blue
            res = cv2.bitwise_and(frame, frame, mask=mask)
            #cv2.imshow('Result', res)
            result_widget.value = bgr8_to_jpeg(res)

            #         if cv2.waitKey(1) & 0xFF == ord('q'):
            #                 break
            time.sleep(0.01)
        cap.release()
        #cv2.destroyAllWindows()
```

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

```
#When we need to end the entire program process, we need to run the code
stop_thread(thread1)
```

This is the effect of performing red recognition, as shown below.
If you want to recognize other color, you can close the code in the red section, and open the code section of the other section.