Color tracking

1. Introduction

This course mainly uses the camera of Raspberry Pi to obtain the image of the camera, and analyzes the image through the opencv library to frame the part of the image that conforms to the HSV value of the color you choose, and then combines the robot dog to adjust the body posture to control the camera activity, so as to track the color.

2. Code analysis

Create multiple buttons to switch colors. If you need to display Chinese, please set G_ ENABLE_ CHINESE = True.

```
# 中文开关，默认为英文 Chinese switch. The default value is English
g_ENABLE_CHINESE = False

Name_widgets = {
    'Close': ("Close", "关闭"),
    'Red': ("Red", "红色"),
    'Green': ("Green", "绿色"),
    'Blue': ("Blue", "蓝色"),
    'Yellow': ("Yellow", "黄色"),
    'Close_Camera': ("Close_Camera", "关闭摄像头")
}
```

```
# 按键按下事件处理   Key press event processing
def on_button_clicked(b):
    global color_lower, color_upper, g_mode
    global g_action
    ALL_Uncheck()
    b.icon = 'check'
    with output:
        print("Button clicked:", b.description)
    if b.description == Name_widgets['Close'][g_ENABLE_CHINESE]:
        g_dog.action(0xff)
        g_mode = 0
        b.icon = 'uncheck'
    elif b.description == Name_widgets['Red'][g_ENABLE_CHINESE]:
        color_lower = np.array([0, 43, 46])
        color_upper = np.array([10, 255, 255])
        g_dog.action(0xff)
        g_mode = 1
    elif b.description == Name_widgets['Green'][g_ENABLE_CHINESE]:
        color_lower = np.array([35, 43, 46])
        color_upper = np.array([77, 255, 255])
        g_dog.action(0xff)
        g_mode = 1
    elif b.description == Name_widgets['Blue'][g_ENABLE_CHINESE]:
        color_lower=np.array([100, 43, 46])
        color_upper = np.array([124, 255, 255])
        g_dog.action(0xff)
        g_mode = 1
    elif b.description == Name_widgets['Yellow'][g_ENABLE_CHINESE]:
        color_lower = np.array([26, 43, 46])
        color_upper = np.array([34, 255, 255])
        g_dog.action(0xff)
        g_mode = 1
```

Color tracking task processing: draw a box for the identified HSV color image, and adjust the body posture of the robot dog to track the identified color.

```
# 颜色追踪任务  Color tracking task
def Color_Tracking_Task():
    global color_lower, color_upper, g_mode
    t_start = time.time()
    fps = 0
    color_x = 0
    color_y = 0
    color_radius = 0
    while True:
        ret, frame = image.read()
        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  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:
                    # 将检测到的颜色标记出来  Mark the detected color
                    cv2.circle(frame,(int(color_x),int(color_y)),int(color_radius),(255,0,255),2)
                    value_x = color_x - 320
                    value_y = color_y - 240
                    if value_x > 110:
                        value_x = 110
                    elif value_x < -110:
                        value_x = -110
                    if value_y > 150:
                        value_y = 150
                    elif value_y < -150:
                        value_y = -150
                    g_dog.attitude(['y','p'],[-value_x/10, value_y/10])
            else:
                color_x = 0
                color_y = 0
            cv2.putText(frame, "X:%d, Y%d" % (int(color_x), int(color_y)), (40,40), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
            t_start = time.time()
            fps = 0
        else:
            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
        image_widget.value = bgr8_to_jpeg(frame)
```

Start a daemon thread, run the camera identification task, and display the camera image.

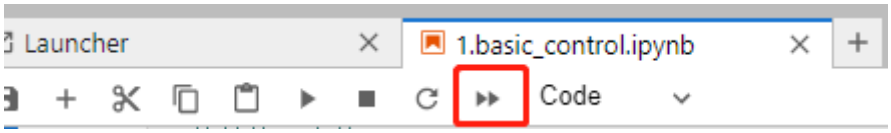
```
# 启动摄像头显示任务  Start the camera display task
thread1 = threading.Thread(target=Color_Tracking_Task)
thread1.setDaemon(True)
thread1.start()

output = widgets.Output()
box_btn = widgets.VBox([Redbutton, Greenbutton, Bluebutton, Yellowbutton, Closebutton, button_Close_Camera])
box_display = widgets.HBox([image_widget, box_btn, output])
display(box_display)
```

1.3 Steps

Open the jupyterLab client and find following code path

```
DOGZILLA/Samples/3_AI_Visual/2.color_tracking.ipynb
```

Click the following icon to run all cells, and then pull to the bottom to see the generated controls.



Please place the robot dog in the pure color environment of the camera image, otherwise it may interfere with the tracking color of the robot dog due to the wrong identification caused by color interference.

The left side shows the camera screen. First select the color to be tracked according to the button on the right, then place the color in front of the camera, move the color within the range of the robot dog's body and the camera, and you can see the robot dog wriggling with the color.

Finally, click close_camera button turns off the camera.