Color recognition action group

1. Introduction

This course mainly uses the camera of raspberry pie to obtain the image of the camera, and analyzes the image through the opencv library to frame some images that meet the HSV value of the color you choose. If the recognized color is within the detection range and the radius exceeds a certain threshold, the robot dog will take corresponding actions.

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")
}
```

Each time the color button is pressed, the global HSV data color is modified_ Lower and color_ Upper, and modify g according to the color value_ The action action can move the corresponding action when the corresponding color is recognized.

```
# 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
        g_action = 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
        g_action = 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
        g_action = 2
    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
        g_action = 3
    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
        g_action = 4
```

Color action task processing: draw the minimum radius circle for the identified HSV color image, and then transfer the information contained in the circle, such as the center coordinate and radius, to the control action function action_ Handle.

```
# 颜色动作任务  Color Action task
def Color_Action_Task():
    global color_lower, color_upper, g_mode
    global g_action
    t_start = time.time()
    fps = 0
    while True:
        ret, frame = image.read()
        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)
                    action_handle(color_x, color_y, color_radius, g_action)
        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)
```

action_ Handle function: according to the circle information and G_ The value of action to determine whether to execute the corresponding action.

```
# 控制动作 control action
def action_handle(color_x, color_y, color_r, action):
    if color_r > 60 and (220 <= color_x <= 420) and (140 <= color_y <= 340):
        if action == 1:
            g_dog.action(14) # 伸懒腰 Stretch
            time.sleep(.1)
        elif action == 2:
            g_dog.action(13)  # 招手 Wave_Hand
            time.sleep(.1)
        elif action == 3:
            g_dog.action(16)  # 左右摇摆 Swing
            time.sleep(.1)
        elif action == 4:
            g_dog.action(18) # 找食物 Foraging
            time.sleep(.1)
```

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

image.

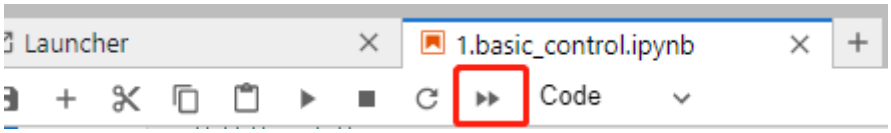
```
# 启动摄像头显示任务  Start the camera display task
thread1 = threading.Thread(target=Color_Action_Task)
thread1.setDaemon(True)
thread1.start()

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/3.color_action.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 a pure color environment of the camera image, otherwise it may be misidentified due to color interference and trigger the action by mistake.

The camera screen is displayed on the left. First select the color to be recognized according to the button on the right, and then place the color in front of the camera. After recognizing the selected color, the corresponding action will be taken.

Finally, click close_camera button turns off the camera.