1、color recognition

1、color recognition
- 1.1 Introduction
- 1.2 Core content analysis
- 1.3 operation

1.1 Introduction

This course mainly uses the camera on the motherboard to obtain the camera picture, and analyzes the image through the OPENCV library. It can frame red, green, blue and yellow objects and display the names of the corresponding colors.

1.2 Core content analysis

Set the default angle of the servo gimbal. The default angle is S1=90 and S2=20. The default angle can be modified according to actual needs. Since the steering gear needs a certain angle to rotate, the range of def_s1 is [30, 150] and the range of def_s2 is [20, 95].

```
def_s1 = 90
def_s2 = 20
g_bot.Gimbal_1_2(def_s1, def_s2)
```

The HSV_Config file is a library file used to process color recognition of incoming images.

Color_HSV defines the HSV value of a color. Since the color of an object has color differences and light effects, if the recognition of a certain color is inaccurate, you can adjust the course based on the color HSV value. After adjusting for the best effect, record the data and update the HSV value of the corresponding color.

```
update_hsv = HSV_Config.update_hsv()

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))}
```

The program function of processing the camera screen is to pass the read camera image to the update_hsv object to process and output the colors existing in the current screen, frame the corresponding color and display the color name, and finally transmit it through the image control for display.

```
def task_processing():
    global g_stop_program, g_color_state
    t_start = time.time()
    m_fps = 0
    while g_camera.isOpened():
        if g_stop_program:
            break
        ret, frame = g_camera.read()
        frame, binary, color = update_hsv.get_contours(frame, Color_HSV)
        if color is not None and g_color_state == False:
            print("color:", color)
            g_color_state = True
            threading.Thread(target=color_action, args=(color,)).start()

        m_fps = m_fps + 1
        fps = m_fps / (time.time() - t_start)
        if (time.time() - t_start) >= 2:
            m_fps = fps
            t_start = time.time() - 1
        cv2.putText(frame, "FPS " + str(int(fps)), (10,20),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,255), 1)

        # 图像传输给显示组件 The image is transmitted to the display component
        image_widget.value = bgr8_to_jpeg(frame)
        # time.sleep(.01)
```

Depending on the color, the robot reacts differently. If red is detected, the robot's camera pan/tilt moves left and right. If green is detected, the robot's camera pan/tilt moves up and down. If blue is detected, the robot does a warm-up squatting motion. If yellow is detected, the robot's camera pan/tilt moves up and down. , the robot waved and said no action.

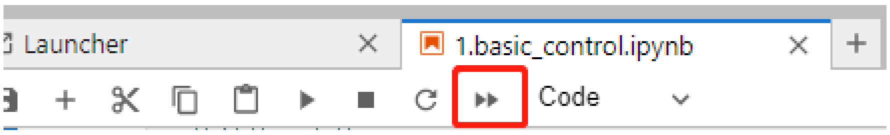
```
def color_action(color):
    global g_color_state
    if color == "Red":
        angle = 30
        g_bot.Gimbal_1_2(def_s1+angle, def_s2)
        time.sleep(.1)
        g_bot.Gimbal_1_2(def_s1-angle, def_s2)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1+angle, def_s2)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1-angle, def_s2)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1+angle, def_s2)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1-angle, def_s2)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1, def_s2)
        time.sleep(1)
    elif color == "Green":
        angle = 20
        g_bot.Gimbal_1_2(def_s1, def_s2+angle)
        time.sleep(.1)
        g_bot.Gimbal_1_2(def_s1, def_s2-angle)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1, def_s2+angle)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1, def_s2-angle)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1, def_s2+angle)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1, def_s2-angle)
        time.sleep(.2)
        g_bot.Gimbal_1_2(def_s1, 20)
        time.sleep(1)
    elif color == "Blue":
        g_bot.action(4) # warm up
        time.sleep(4)
    elif color == "Yellow":
        g_bot.action(6) # say no
        time.sleep(3)
    g_color_state = False
```

1.3 operation

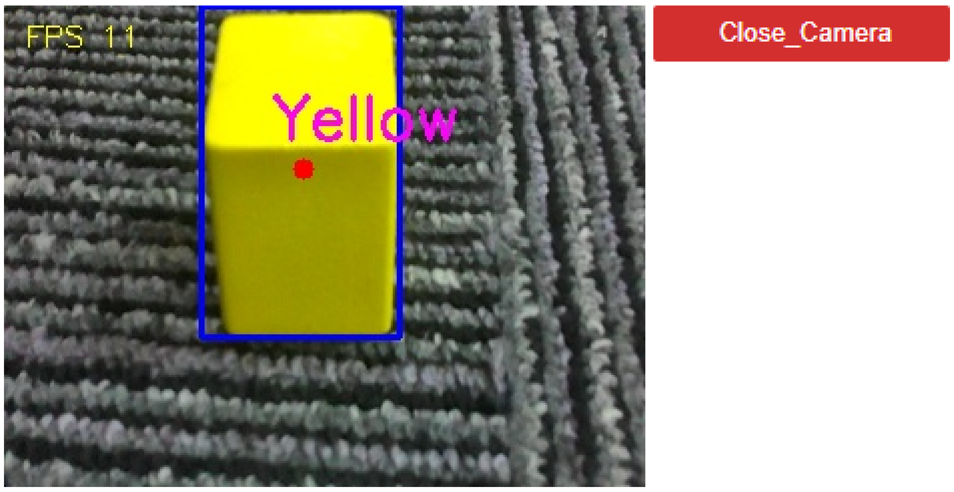
Open the jupyterLab client and find the code path:

```
muto/Samples/AI_Samples/01_color_recogniton/color_recognition.ipynb
```

Click to run all cells, and then scroll to the bottom to see the generated controls.



The left side shows the picture collected by the camera, and the Close_Camera on the right is used to close the camera and program process. Place a red, green, blue or yellow object in front of the camera, and the image will automatically frame the corresponding color block and display the name of the color. If red is detected, the robot's camera pan/tilt moves left and right. If green is detected, the robot's camera pan/tilt moves up and down. If blue is detected, the robot does a warm-up squatting motion. If yellow is detected, the robot's camera pan/tilt moves up and down. , the robot waved and said no action.



Finally click the Close_Camera button to close the camera.