4、Color recognition action group

4、Color recognition action group
4.1 Introduction to gameplay
4.2 Core content analysis
4.3 operation

4.1 Introduction to gameplay

This course focuses on identifying red, green, blue and yellow, and performing different actions based on these four colors.

4.2 Core content analysis

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.

```
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))}
```

Create a new image processing task, then analyze the image content to obtain relevant information and control the robot movement.

```
def task_processing():
    global g_stop_program, g_start_function
    t_start = time.time()
    m_fps = 0
    while g_camera.isOpened():
        if g_stop_program:
            break
        ret, frame = g_camera.read()
        frame, hsv_name = get_color(frame, Color_HSV)
        if g_start_function:
            robot_action(hsv_name)
        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)
```

Analyze images based on color HSV values.

```
def get_color(image, color_hsv):
    hsv_name = None
    # image = cv2.resize(image, (320, 240), )
    for key, value in color_hsv.items():
        color_contours, binary = Image_Processing(image, color_hsv[key])
        hsv_name = draw_contours(image, key, color_contours)
        if hsv_name is not None:
            break
    return image, hsv_name
```

Image processing and analysis to obtain the outline of the color block.

```
def Image_Processing(image, hsv_range):
        (lowerb, upperb) = hsv_range
        color_mask = image.copy()
        hsv_img = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        # 筛选出位于两个数组之间的元素。 Filters out the elements between two arrays.
        color = cv2.inRange(hsv_img, lowerb, upperb)
        # 设置非掩码检测部分全为黑色 Set the non-mask detection part to black
        color_mask[color == 0] = [0, 0, 0]
        # 将图像转为灰度图 Convert the image to grayscale
        gray_img = cv2.cvtColor(color_mask, cv2.COLOR_RGB2GRAY)
        # 获取不同形状的结构元素 Gets structure elements of different shapes
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
        dst_img = cv2.morphologyEx(gray_img, cv2.MORPH_CLOSE, kernel)
        # 图像二值化操作 Image binarization operation
        ret, binary = cv2.threshold(dst_img, 10, 255, cv2.THRESH_BINARY)
        # 获取轮廓点集(坐标) Get contour point (coordinates)
        version = cv2.__version__
        if int(version[0]) < 4:
            _, contours, heriachy = cv2.findContours(binary, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE) # opencv3
        else:
            contours, heriachy = cv2.findContours(binary, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)  # opencv4
        return contours, binary
```

Draw the outline of the color in the image, where area_threshold is the threshold that controls the color recognition, and the value can be modified according to the size of the recognized color object.

```
def draw_contours(image, hsv_name, contours, area_threshold=3000):
    for i, cnt in enumerate(contours):
        mm = cv2.moments(cnt)
        if mm['m00'] == 0:
            continue
        cx = mm['m10'] / mm['m00']
        cy = mm['m01'] / mm['m00']
        (x, y) = (np.int(cx), np.int(cy))
        # 计算轮廓的面积 Calculate the area of the contour
```

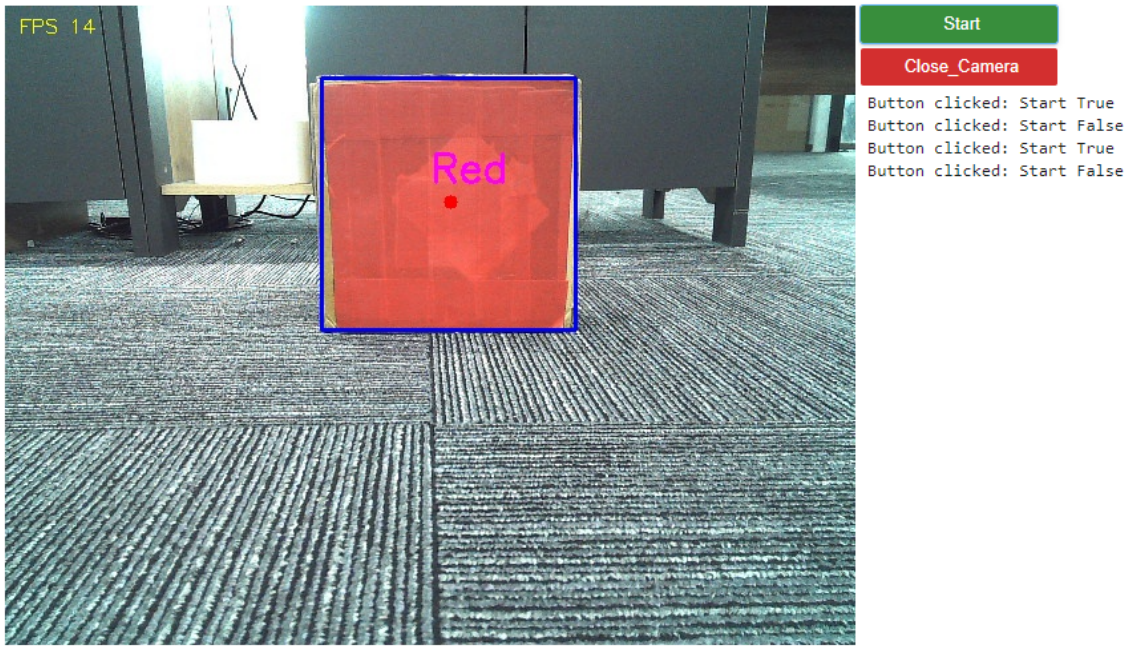
```
        area = cv2.contourArea(cnt)
        # print("area:", area)
        if area > area_threshold:
            # 绘制最小矩形轮廓和中心点 Draw the minimum rectangle outline and center
point
            cv2.circle(image, (x, y), 5, (0, 0, 255), -1)
            rect = cv2.minAreaRect(cnt)
            box = cv2.boxPoints(rect)
            box = np.int0(box)
            cv2.drawContours(image, [box], 0, (255, 0, 0), 2)
            cv2.putText(image, hsv_name, (int(x - 15), int(y - 15)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 255), 2)
            return hsv_name
    return None
```

Control the robot to perform action group movements based on the colors in the image. Among them, red corresponds to the stretching movement, green corresponds to the spinning movement, blue corresponds to the greeting movement, and yellow corresponds to the warm-up squatting movement.

```
def robot_action(color):
    global g_time_pass
    if g_time_pass >= 0:
        return
    if color is None:
        return
    elif color == "Red":
        print("Red Color")
        # 伸懒腰 stretch
        g_bot.action(1)
        g_time_pass = 10
    elif color == "Green":
        print("Green Color")
        # 原地转圈 circle around
        g_bot.action(5)
        g_time_pass = 10 * 5
    elif color == "Blue":
        print("Blue Color")
        # 打招呼 greet
        g_bot.action(2)
        g_time_pass = 10 * 1
    elif color == "Yellow":
        print("Yellow Color")
        # 热身蹲起 Warm-up
        g_bot.action(4)
        g_time_pass = 10 * 2.5
```

Every time an action is executed, there will be a detection timeout. Before the timeout reaches zero, no action instructions will be sent to avoid sending action instructions frequently.

```
def task_time():
    global g_time_pass
    while True:
        if g_time_pass > 0:
            g_time_pass = int(g_time_pass - 1)
            if g_time_pass == 0:
                g_bot.zero_reset()
                time.sleep(.5)
                g_time_pass = -1
        time.sleep(.1)
```

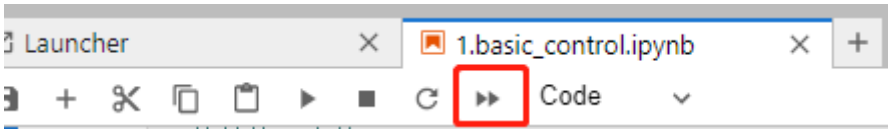


4.3 operation

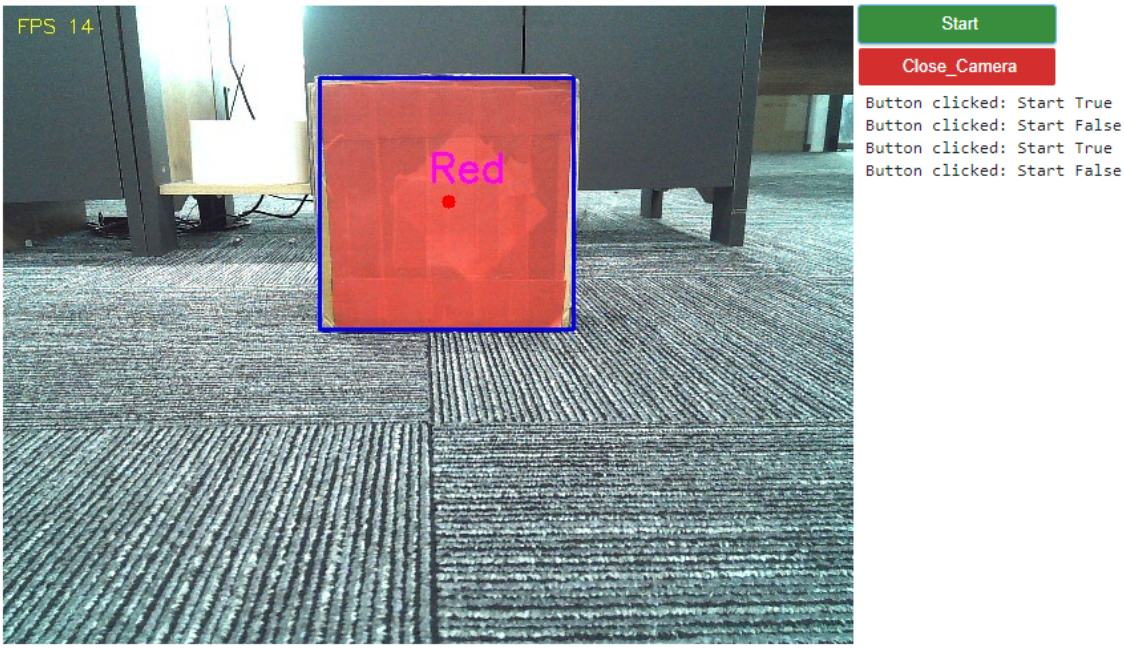
Open the jupyterLab client and find the code path:

```
muto/Samples/AI_Samples/04_color_action/color_action.ipynb
```

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



The camera recognizes red, green, blue and yellow, outlines the color, and displays the color name. At this time, whether the robot is moving or not, you need to click the Start button before the robot will make corresponding action responses based on the recognized colors. Click the Start button again to turn off robot motion.



Finally click the Close_Camera button to close the camera.