Lesson 3 Color Recognition

1. Program Logic

With camera vision module, TonyPi can recognize different colors visually.

First, program TonyPi to recognize colors with Lab color space. Convert the RGB color space to Lab, image binarization, and then perform operations such as expansion and corrosion to obtain an outline containing only the target color. Use circles to frame the color outline to realize object color recognition.

Secondly, judge the recognized color. If the sett color is detected the head servo will be turned up and down, otherwise it will be turned left and right.

The source code of the program is located in：/home/pi/TonyPi/Functions/ColorDetect.py

```
    if action_finish:
        for i in lab_data:
            if i != 'black' and i != 'white':
                frame_mask = cv2.inRange(frame_lab,
                                         (lab_data[i]['min'][0],
                                          lab_data[i]['min'][1],
                                          lab_data[i]['min'][2]),
                                         (lab_data[i]['max'][0],
                                          lab_data[i]['max'][1],
                                          lab_data[i]['max'][2]))  #Perform bit operations on the original image and mask
                eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)))  #erode
                dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #dilate
                if debug:
                    cv2.imshow(i, dilated)
                contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]  #Find the contour
                areaMaxContour, area_max = getAreaMaxContour(contours)  #the maximum area has been found
                if areaMaxContour is not None:
                    if area_max > max_area:#Find the maximum area
                        max_area = area_max
                        color_area_max = i
                        areaMaxContour_max = areaMaxContour
        if max_area > 200:  # The maximum area has been found
            ((centerX, centerY), radius) = cv2.minEnclosingCircle(areaMaxContour_max)  # Get the minimum circumferential circle
            centerX = int(Misc.map(centerX, 0, size[0], 0, img_w))
            centerY = int(Misc.map(centerY, 0, size[1], 0, img_h))
            radius = int(Misc.map(radius, 0, size[0], 0, img_w))
            cv2.circle(img, (centerX, centerY), radius, range_rgb[color_area_max], 2)# draw circle

            if color_area_max == 'red':  # Red is the color with the maximum area
                color = 1
            elif color_area_max == 'green':  # Green is the color with the maximum area
                color = 2
            elif color_area_max == 'blue':  # Blue is the color with the maximum area
                color = 3
            else:
                color = 0
            color_list.append(color)

            if len(color_list) == 3:  # Multiple judgment
                # take average
                color = int(round(np.mean(np.array(color_list))))
                color_list = []
                if color == 1:
                    detect_color = 'red'
                    draw_color = range_rgb["red"]
                elif color == 2:
                    detect_color = 'green'
                    draw_color = range_rgb["green"]
                elif color == 3:
                    detect_color = 'blue'
                    draw_color = range_rgb["blue"]
```

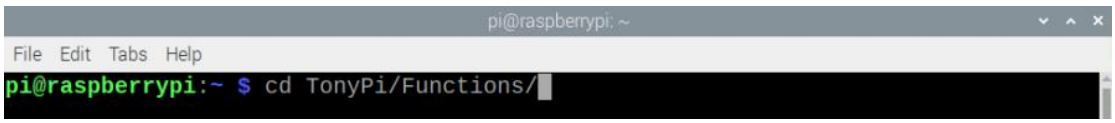
2. Operation Steps

Pay attention to the text format in the input of instructions.

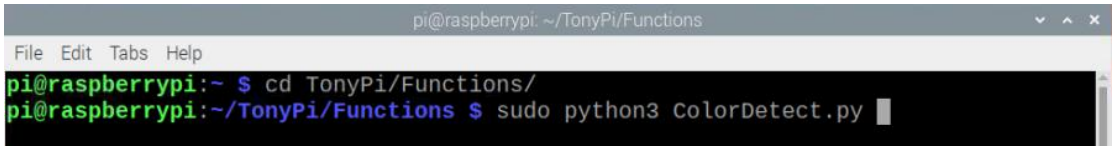
1) Turn on robot and connect to Raspberry Pi desktop with VNC.

2) Click  or press “Ctrl+Alt+T” to enter the LX terminal.

3) Enter “cd TonyPi/Functions/” command, and then press “Enter” to come to the category of games programmings.



4) Enter “sudo python3 ColorDetect.py”, then press “Enter” to start the game.



5) If you want to exit the game programming, press “Ctrl+C” in the LX terminal interface. If the exit fails, please try it few more times.

3. Project Outcome

The default color is red. If you want to change to blue or green, please refer to “4.1Modify Program Default Recognition Color”.

Place the red ball in front of TonyPi’s camera and TonyPi will nod its head when red ball is recognized. It will shake head is the green and blue balls are detected.

4. Function Extension

4.1 Modify Default Recognition Color

Red, green and blue are the built-in colors in the color recognition program and the red is the default color. In the following steps, we're going to modify the recognized color as green.

Step1: Enter command "cd TonyPi/Functions/" to the directory where the game program is located.

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

Step2: Enter command "**sudo vim ColorDetect.py**" to go into the game program through vi editor.

```
pi@raspberrypi: ~/TonyPi/Functions
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo vim ColorDetect.py
```

Step3: Input "114" and press "shfit+g" to the line for modification.

```
110             return
111         if __isRunning:
112             if detect_color != 'None':
113                 action_finish = False
114                 if detect_color == 'red':
115                     Board.setPWMServoPulse(1, 1800, 200)
116                     time.sleep(0.2)
117                     Board.setPWMServoPulse(1, 1200, 200)
118                     time.sleep(0.2)
119                     Board.setPWMServoPulse(1, 1800, 200)
120                     time.sleep(0.2)
121                     Board.setPWMServoPulse(1, 1200, 200)
122                     time.sleep(0.2)
123                     Board.setPWMServoPulse(1, 1500, 100)
124                     time.sleep(0.1)
125                     detect_color = 'None'
126                     draw_color = range_rgb["black"]
127                     time.sleep(1)
128                 elif detect_color == 'green' or detect_color == 'blue':
129                     Board.setPWMServoPulse(2, 1800, 200)
```

Step4: Press "i" to enter the editing mode, then modify red in if detect_color == 'red' to green. And enter "red" instead of "green" in 128 line （elif detect_color== 'green' or detect_color == 'blue'）

Same modify method if you want to recognize blue.

```
111         if __isRunning:
112             if detect_color != 'None':
113                 action_finish = False
114                 if detect_color == 'green':
115                     Board.setPWMServoPulse(1, 1800, 200)
116                     time.sleep(0.2)
117                     Board.setPWMServoPulse(1, 1200, 200)
118                     time.sleep(0.2)
119                     Board.setPWMServoPulse(1, 1800, 200)
120                     time.sleep(0.2)
121                     Board.setPWMServoPulse(1, 1200, 200)
122                     time.sleep(0.2)
123                     Board.setPWMServoPulse(1, 1500, 100)
124                     time.sleep(0.1)
125                     detect_color = 'None'
126                     draw_color = range_rgb["black"]
127                     time.sleep(1)
128                 elif detect_color == 'red' or detect_color == 'blue':
129                     Board.setPWMServoPulse(2, 1800, 200)
130                     time.sleep(0.2)
131                     Board.setPWMServoPulse(2, 1200, 200)
132                     time.sleep(0.2)
133                     Board.setPWMServoPulse(2, 1800, 200)
-- 插入 --                                                    128,42        43%
```

Step5: Press “Esc” to enter last line command mode. Input “:wq” to save the file and exit the editor.

```
274         else:
275             time.sleep(0.01)
276     my_camera.camera_close()
277     cv2.destroyAllWindows()
~
~
:wq
```

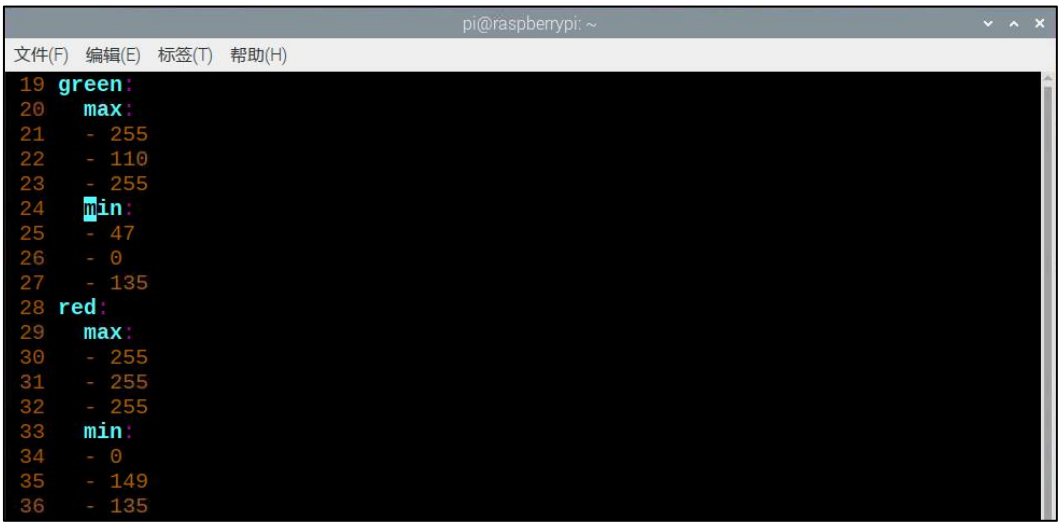
4.2 Add Recognized Color

In addition to the built-in recognized colors, you can set other recognized colors in the programming. Take orange as example:

1) Open VNC, input command “**sudo vim TonyPi/lab_config.yaml**” to open Lab color setting document.

```
pi@raspberrypi: ~
文件(F)  编辑(E)  标签(T)  帮助(H)
pi@raspberrypi:~ $ sudo vim TonyPi/lab_config.yaml
```

It is recommended to use screenshot to record the initial value.

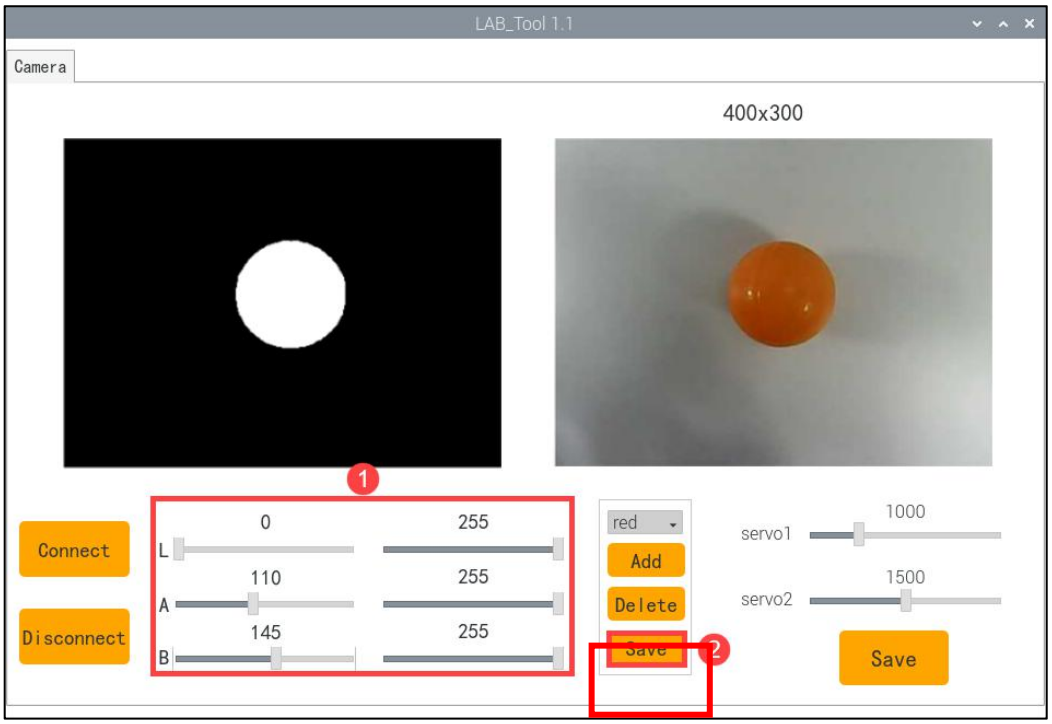


2) Click the debugging tool icon in the system desktop. Choose “Run” in the pop-up window.

3) Click “Connect” button in the lower left hand. When the interface display the camera returned image, the connection is successful. Select "red" in the right box first.

4) Drag the corresponding sliders of L, A, and B until the color area to be recognized in the left screen becomes white and other areas become black.

Point the camera at the color you want to recognize. For example, if you want to recognize orange, you can put the orange ball in the camera's field of view. Adjust the corresponding sliders of L, A, and B until the orange part of the left screen becomes white and other colors become black, and then click " Save" button to keep the modified data.



5) After the modification is completed, check whether the modified data was successfully written in. Enter the command again "**sudo vim TonyPi/lab_config.yaml**" to check the color setting parameters.

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
19 green:
20   max:
21   - 255
22   - 110
23   - 255
24   min:
25   - 47
26   - 0
27   - 135
28 red:
29   max:
30   - 255
31   - 255
32   - 255
33   min:
34   - 0
35   - 110
36   - 145
37 white:
38   max:
39   - 255
40   - 255
41   - 255
```

For the game's performance, it's recommended to use the LAB_Tool tool to modify the value back to the initial value after the modification is completed.

6) Check the data in red frame. If the edited value was written in the program, press "Esc" and enter ":wq" to save it and exit.

7) The default recognized color can be set as red according to the "4.1Modify Default Recognition Color" in this text.

```
pi@raspberrypi: ~/TonyPi/Functions
文件(F)  编辑(E)  标签(T)  帮助(H)
110                 return
111             if __isRunning:
112                 if detect_color != 'None':
113                     action_finish = False
114                     if detect_color == 'red':
115                         Board.setPWMServoPulse(1, 1800, 200)
116                         time.sleep(0.2)
117                         Board.setPWMServoPulse(1, 1200, 200)
118                         time.sleep(0.2)
119                         Board.setPWMServoPulse(1, 1800, 200)
120                         time.sleep(0.2)
121                         Board.setPWMServoPulse(1, 1200, 200)
122                         time.sleep(0.2)
123                         Board.setPWMServoPulse(1, 1500, 100)
124                         time.sleep(0.1)
125                         detect_color = 'None'
126                         draw_color = range_rgb["black"]
127                         time.sleep(1)
128                     elif detect_color == 'green' or detect_color == 'blue':
129                         Board.setPWMServoPulse(2, 1800, 200)
130                         time.sleep(0.2)
131                         Board.setPWMServoPulse(2, 1200, 200)
132                         time.sleep(0.2)
                                                        115,21        42%
```

8) Start the game again and put the orange ball in front of the camera. TonyPi will execute “nod” action after recognizing.

9) If you want to add other colors as recognized color, please operate as the above steps.

5. Program Parameter Instruction

5.1 Color Detection Parameter

The detected parameters involved in the process of detection are as follow:

1) Before converting the image into LAB space, GaussianBlur() function is used to perform Gaussian filtering to denoise image, as the figure shown below:

```
frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)
frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
frame_lab = cv2.cvtColor(frame_gb, cv2.COLOR_BGR2LAB)  # 将图像转换到LAB空间
```

The first parameter “**frame_resize**”is the input image.

The second parameter “**(3, 3)**” the size of Gaussian kernel. Larger kernels usually result in greater filtering, which makes the output image more blurred and also increase the computational complexity.

The third parameter “3” is the standard deviation of the Gaussian function along X direction, which is used in Gaussian filters to control the variation around the its mean value. When the data increases, the allowable variation range around the mean value increases, vice verse.

2) Binarize the input image by inRang function, as the figure shown below:

```
frame_mask = cv2.inRange(frame_lab,
                         (lab_data[i]['min'][0],
                          lab_data[i]['min'][1],
                          lab_data[i]['min'][2]),
                         (lab_data[i]['max'][0],
                          lab_data[i]['max'][1],
                          lab_data[i]['max'][2])) #对原图像和掩模进行位运算
```

3) To reduce interference to make the image smoother, it needs to be eroded and dilated, as the figure shown below:

```
eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #腐蚀
dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #膨胀
```

The getStructuringElement function is used in processing to generate structural elements in different shapes.

The first parameter "cv2.MORPH_RECT" is the kernel shape. Here is rectangle.

The second parameter "(3, 3)" is the size of rectangle. Here is 3×3.

4) Find the object with the biggest contour, as the figure shown below:

```
for c in contours: # 历遍所有轮廓
    contour_area_temp = math.fabs(cv2.contourArea(c)) # 计算轮廓面积
    if contour_area_temp > contour_area_max:
        contour_area_max = contour_area_temp
        if contour_area_temp > 50: # 只有在面积大于50时，最大面积的轮廓才是有效的，以过滤干扰
            area_max_contour = c

return area_max_contour, contour_area_max # 返回最大的轮廓
```

To avoid interference, the "if contour_area_temp > 100" instruction sets the contour with the largest area is valid only if the area is greater than 100.

5.2 Color Recognition Parameter

The control parameters involved in color recognition are as follow:

1) When the robot recognizes the red ball, cv2.circle() function can be used to draw a circle in the returned image to circle the ball, as the figure show below:

```
centerY = int(Misc.map(centerY, 0, size[1], 0, img_h))
radius = int(Misc.map(radius, 0, size[0], 0, img_w))
cv2.circle(img, (centerX, centerY), radius, range_rgb[color_area_max], 2)#画圆
```

The first parameter “img” is the input image. The parameter here is the image of the recognized ball.

The second parameter “(centerX, centerY)” is the coordinate of centre point of drawn circle. (determined according to the detected object)

The third parameter is the radius of drawn circle. (determined according to the detected object)

The fourth parameter “range_rgb[detect_color]” is the line color of drawn circle.

The fifth parameter “2” is the line width of the drawn circle.

2) After detecting the ball, add text on the returned screen to describe the color of the ball through cv2.putText() function,as the figure shown below:

```
cv2.putText(img, "Color: " + detect_color, (10, img.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.65, draw_color, 2)
return img
```

The first parameter “**img**” is the input image.

The second parameter “"Color: " + detect_color” is the added text.

The third parameter “(10, img.shape[0] - 10)” is the coordinate of the upper-left corner of the added text.

The fourth parameter “cv2.FONT_HERSHEY_SIMPLEX” is the font of the added text.

The fifth parameter “0.65” is the font size.

The sixth parameter “draw_color” is the font color.

The seventh parameter “2” is the font thickness.

5.3 Execute Action Parameter

1) After recognizing the red ball, servo 1 is controlled to make robot nod twice, and then return to the initial position, as the figure shown below:

```
if detect_color != 'None':
    action_finish = False
    if detect_color == 'red':
        Board.setPWMServoPulse(1, 1800, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(1, 1200, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(1, 1800, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(1, 1200, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(1, 1500, 100)
        time.sleep(0.1)
        detect_color = 'None'
        draw_color = range_rgb["black"]
        time.sleep(1)
```

Take code “Board.setPWMServoPulse(1, 1500, 100)” as example:

The first parameter “1” represents ID1 servo.

The second parameter “1500” is the pulse width and 1500 is to control servo return to the initial position.

The third parameter “100” is the servo running time, that is, the running time is 100ms.

2) After recognizing green or blue ball, servo 2 is controlled to make robot shake head twice, and then return to the initial position, as the figure shown below:

```
    elif detect_color == 'green' or detect_color == 'blue':
        Board.setPWMServoPulse(2, 1800, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(2, 1200, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(2, 1800, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(2, 1200, 200)
        time.sleep(0.2)
        Board.setPWMServoPulse(2, 1500, 100)
        time.sleep(0.1)
        detect_color = 'None'
        draw_color = range_rgb["black"]
        time.sleep(1)
```