

Lesson 5 Color Tracking

1. Program Logic

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, process the rotation and the x and y coordinates of the center point of the image are used as the set value. Input the current acquired x and y coordinates to update the pid.

Thirdly, calculate according to the image position feedback, and program the robot to achieve the function of color tracking.

The source code of the program is located in:

/home/pi/TonyPi/Functions/ColorTrack.py

```

17 #color tracking
18
19 debug = False
20
21 if sys.version_info.major == 2:
22     print('Please run this program with python3!')
23     sys.exit(0)
24
25 range_rgb = {
26     'red': (0, 0, 255),
27     'blue': (255, 0, 0),
28     'green': (0, 255, 0),
29     'black': (0, 0, 0),
30     'white': (255, 255, 255),
31 }
32
33 __target_color = ('red',)
34 # set be detected color
35 def setTargetColor(target_color):
36     global __target_color
37     __target_color = target_color
38     return (True, ())
39
40 # find the maximum area contour
41 # the parameter is a list of contours to be compared
42 def getAreaMaxContour(contours):
43     contour_area_temp = 0
44     contour_area_max = 0
45     area_max_contour = None
46
47     for c in contours: # traversal all the contours
48         contour_area_temp = math.fabs(cv2.contourArea(c)) # calculate the contour area
49         if contour_area_temp > contour_area_max:
50             contour_area_max = contour_area_temp
51             if contour_area_temp > 10: # only when the area is greater than 300, the contour of the maximum area is effective to filter interference
52                 area_max_contour = c
53
54     return area_max_contour, contour_area_max # return the maximum area contour
55

```

2. Operation Steps

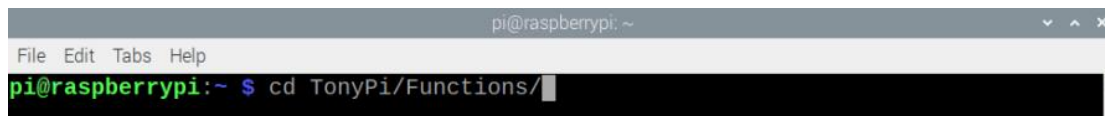
i Pay attention to the text format in the input of instructions.

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

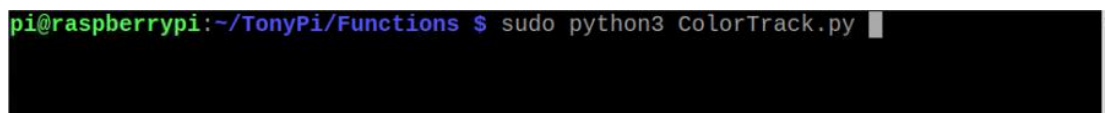
2) Click  or press “Ctrl+Alt+T” to enter the LX terminal.



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



4) Enter “sudo python3 ColorTrack.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

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

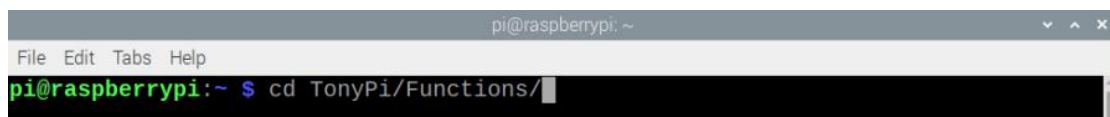
Start the color tracking game and place the red ball in front of the TonyPi to move slowly. TonyPi will follow the movement of the color ball.

4. Function Extension

4.1 Modify Default Tracking Color

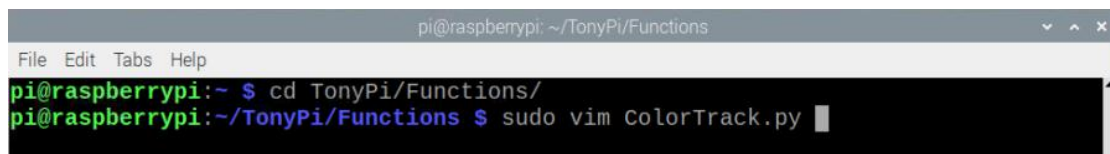
Black, red and green are the built-in colors in the color tracking program and red is the default color. In the following steps, we're going to modify the tracking color as green.

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



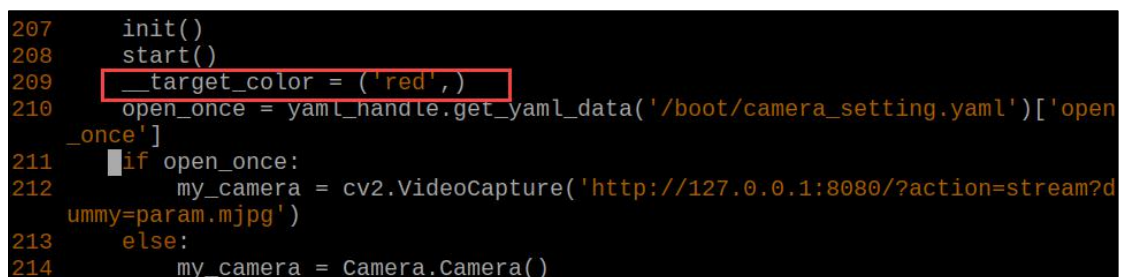
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

Step2: Enter command “sudo vim ColorTrack.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 ColorTrack.py
```

Step3: Input “209” and press “shfit+g” to the line for modification.



```
207     init()  
208     start()  
209     __target_color = ('red',)  
210     open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open  
_once']  
211     if open_once:  
212         my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d  
ummy=param.mjpg')  
213     else:  
214         my_camera = Camera.Camera()
```

Step4: Press “i” to enter the editing mode, then modify red in `__target_color = ('red')` to green. (if you want to recognize blue, please revise to “blue”)

```
207     init()
208     start()
209     __target_color = ('green',)
210     open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open
    _once']
211     if open_once:
212         my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d
        ummy=param.mjpg')
213     else:
214         my_camera = Camera.Camera()
215         my_camera.camera_open()
216         AGC.runActionGroup('stand')
```

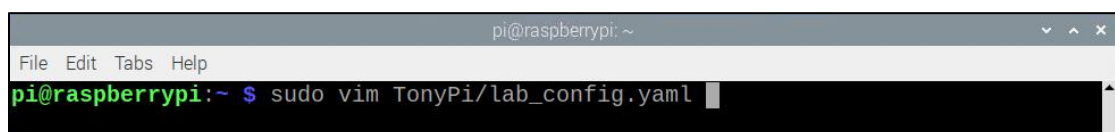
Step5: Press “Esc” to enter last line command mode. Input “:wq” to save the file and exit the editor.

```
228         time.sleep(0.01)
229         my_camera.camera_close()
230         cv2.destroyAllWindows()
~
~
:wq
```

4.2 Add Tracking Color

In addition to the built-in recognized colors, you can set other tracking 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.



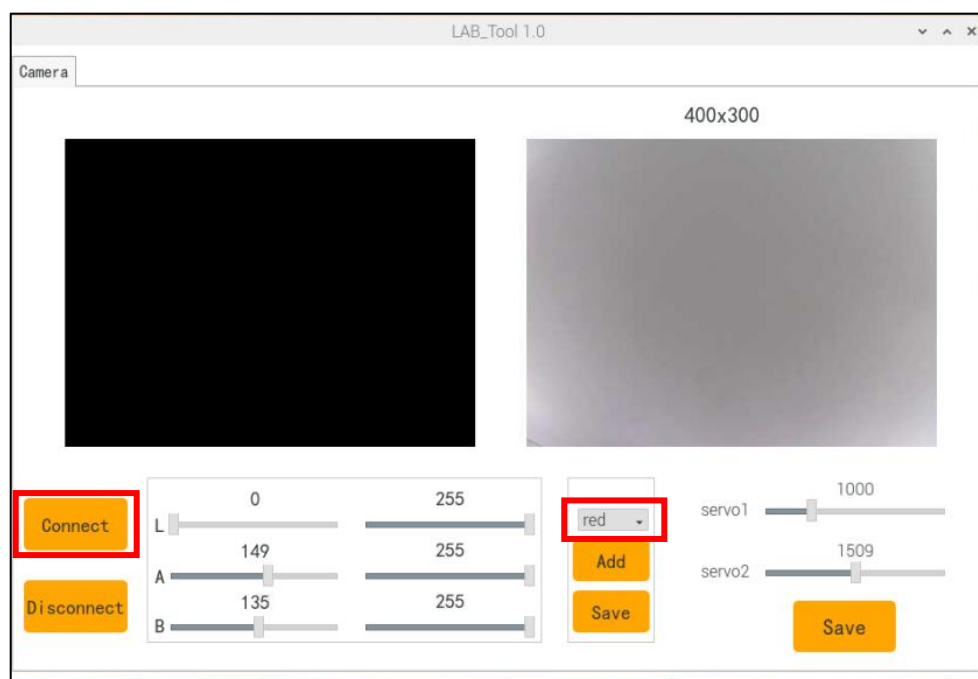
It is recommended to use screenshot to record the initial value.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
1 black:  
2 max:  
3 - 89  
4 - 255  
5 - 255  
6 min:  
7 - 0  
8 - 0  
9 - 0  
10 blue:  
11 max:  
12 - 255  
13 - 146  
14 - 120
```

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

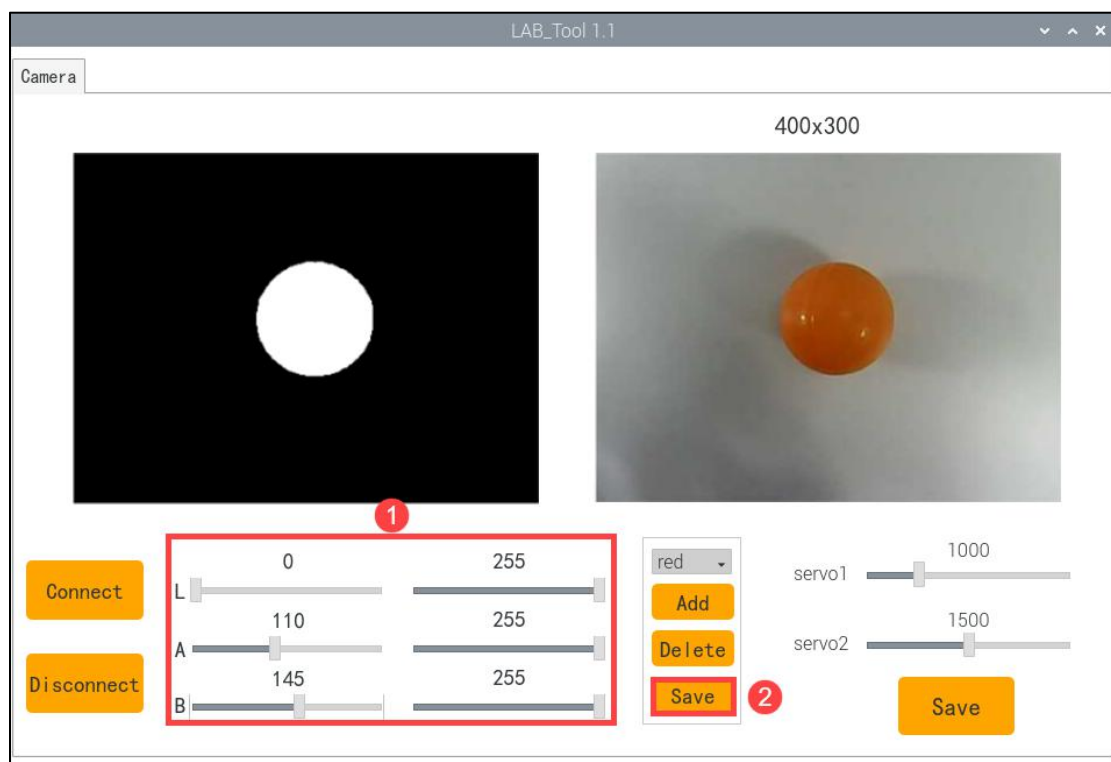


3) Click “Connect” button in the lower left hand. When the interface display the camera returned image, the connection is successful. Select "red" in the right box first.



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 blue, you can put the blue line 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.



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.

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

```

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 tracking color can be set as red according to the “4.1 Modify Default Recognition Color” in this text.

```

207   init()
208   start()
209   __target_color = ('red',)
210   open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open
  _once']
211   if open_once:
212     my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d
  ummy=param.mjpg')
213   else:
214     my_camera = Camera.Camera()
215     my_camera.camera_open()
216   AGC.runActionGroup('stand')

```

8) Starting the game again, TonyPi will follow the movement of target object. If you want to add other colors as tracking color, please operate as the above steps.

5. Program Parameter Instruction

5.1 Color Detection Parameter

In this program, the color of the detected ball is red.

```
207 init()
208 start()
209 __target_color = ('red',)
210 my_camera = Camera.Camera()
211 my_camera.camera_open()
212 AGC.runActionGroup('stand')
```

The parameters mainly 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:

```
136 frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)
137 frame_gb = cv2.GaussianBlur(frame_resize, (5, 5), 5)
138 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 “**(5, 5)**” is 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 “**5**” 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:


```

145 frame_mask = cv2.inRange(frame_lab,
146                             (lab_data[i]['min'][0],
147                             lab_data[i]['min'][1],
148                             lab_data[i]['min'][2]),
149                             (lab_data[i]['max'][0],
150                             lab_data[i]['max'][1],
151                             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 it is rectangle.

The second parameter “(3, 3)” is the size of rectangle. Here it is 3×3 .

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

```

47 for c in contours: # 遍历所有轮廓
48     contour_area_temp = math.fabs(cv2.contourArea(c)) # 计算轮廓面积
49     if contour_area_temp > contour_area_max:
50         contour_area_max = contour_area_temp
51         if contour_area_temp > 10: # 只有在面积大于10时，最大面积的轮廓才是有效的，以过滤干扰
52             area_max_contour = c
53
54 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:

```

161 centerY = int(Misc.map(centerY, 0, size[1], 0, img_h))
162 radius = int(Misc.map(radius, 0, size[0], 0, img_w))
163 cv2.circle(img, (int(centerX), int(centerY)), int(radius), range_rgb[detect_color], 2)
164

```

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

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

The third parameter “int(radius)” 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.

5.3 Execute Action Parameter

After recognizing the red ball, control servo 1 and servo 2 to make the robot move with the ball, as the figure shown below:

```

165 use_time = 0
166 x_pid.SetPoint = img_w/2 #设定
167 x_pid.update(centerX) #当前
168 dx = int(x_pid.output)
169 use_time = abs(dx*0.00025)
170 x_dis += dx #输出
171
172 x_dis = 500 if x_dis < 500 else x_dis
173 x_dis = 2500 if x_dis > 2500 else x_dis
174
175 y_pid.SetPoint = img_h/2
176 y_pid.update(centerY)
177 dy = int(y_pid.output)
178 use_time = round(max(use_time, abs(dy*0.00025)), 5)
179 y_dis += dy
180
181 y_dis = 1000 if y_dis < 1000 else y_dis
182 y_dis = 2000 if y_dis > 2000 else y_dis
183
184 if not debug:
185     Board.setPWMServoPulse(1, y_dis, use_time*1000)
186     Board.setPWMServoPulse(2, x_dis, use_time*1000)
187     time.sleep(use_time)

```

Take code "Board.setPWMServoPulse(1, y_dis, use_time*1000)" as example:

The first parameter "1" represents ID1 servo.

The second parameter "y_dis" is the pulse width.

The third parameter "use_time*1000" is the servo running time and its unit is ms.