

Lesson 3 Motion Tracking

1. Principle

First, program TonyPi Pro 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, compare the recognized color to select the object with the largest contour area as the target. Finally, program TonyPi Pro to track in real-time.

The source code of the program is located in:

/home/pi/TonyPi/Functions/Follow.py

```

216     x_dis = servo2 - 400 if x_dis < servo2 - 400 else x_dis
217     x_dis = servo2 + 400 if x_dis > servo2 + 400 else x_dis
218
219     y_pid.SetPoint = img_h/2
220     y_pid.update(centerY)
221     dy = int(y_pid.output)
222     use_time = round(max(use_time, abs(dy*0.00025)), 5)
223     y_dis += dy
224
225     y_dis = servo1 if y_dis < servo1 else y_dis
226     y_dis = 2000 if y_dis > 2000 else y_dis
227
228     Board.setPWMServoPulse(1, y_dis, use_time*1000)
229     Board.setPWMServoPulse(2, x_dis, use_time*1000)
230     time.sleep(use_time)
231 else:
232     centerX, centerY = -1, -1
233
234     #img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR) # Distortion correction
235
236     return img
237
238 if __name__ == '__main__':
239     init()
240     start()
241     __target_color = ('red',)
242     my_camera = Camera.Camera()
243     my_camera.camera_open()
244     AGC.runActionGroup('stand')
245     while True:
246         img = my_camera.frame
247         if img is not None:
248             frame = img.copy()
249             Frame = run(frame)
250             cv2.imshow('Frame', Frame)
251             key = cv2.waitKey(1)
252             if key == 27:
253                 break
254         else:
255             time.sleep(0.01)
256     my_camera.camera_close()
257     cv2.destroyAllWindows()
258

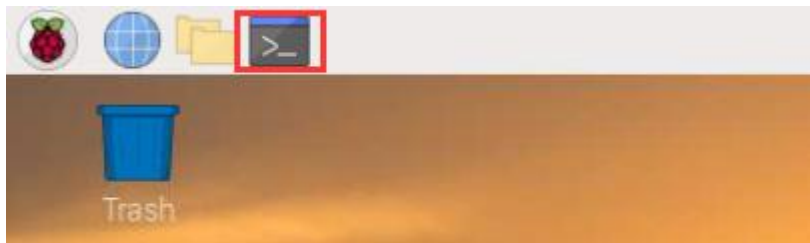
```

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.

```
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

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

```
pi@raspberrypi:~/TonyPi/Functions $ python3 Follow.py
```

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

Start the motion tracking game, place the red block in front of the TonyPi Pro and move it slowly. TonyPi Pro will follow the movement of the block.

4. Function Extension

4.1 Modify Default Tracking Color

Black, red and green are the built-in colors in the motion 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:~ $ cd TonyPi/Functions/
```

Step2: Enter command “sudo vim Follow.py” to go into the game program through vi editor.

```
pi@raspberrypi:~/TonyPi/Functions $ sudo vim Follow.py
```

Step3: Input “241” and press “shfit+g” to the line for modification.

```
238 if __name__ == '__main__':
239     init()
240     start()
241     __target_color = ('red',)
242     my_camera = Camera.Camera()
243     my_camera.camera_open()
244     AGC.runActionGroup('stand')
245     while True:
246         img = my_camera.frame
247         if img is not None:
248             frame = img.copy()
249             Frame = run(frame)
250             cv2.imshow('Frame', Frame)
251             key = cv2.waitKey(1)
252             if key == 27:
```

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”)

```
238 if __name__ == '__main__':
239     init()
240     start()
241     _target_color = ('green',)
242     my_camera = Camera.Camera()
243     my_camera.camera_open()
244     AGC.runActionGroup('stand')
245     while True:
246         img = my_camera.frame
247         if img is not None:
248             frame = img.copy()
249             Frame = run(frame)
250             cv2.imshow('Frame', Frame)
251             key = cv2.waitKey(1)
252             if key == 27:
```

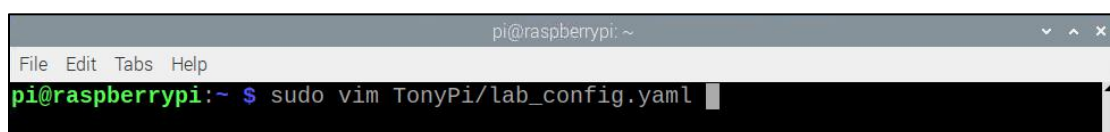
Step5: Press “Esc” to enter last line command mode. Input “:wq” to save the file and exit the editor.

```
245     while True:
246         img = my_camera.frame
247         if img is not None:
248             frame = img.copy()
249             Frame = run(frame)
250             cv2.imshow('Frame', Frame)
251             key = cv2.waitKey(1)
252             if key == 27:
253                 break
254         else:
255             time.sleep(0.01)
: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.



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

```
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
15  min:
16  - 0
17  - 0
18  - 0
19 green:
20  max:
21  - 255
22  - 110
23  - 255
```

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

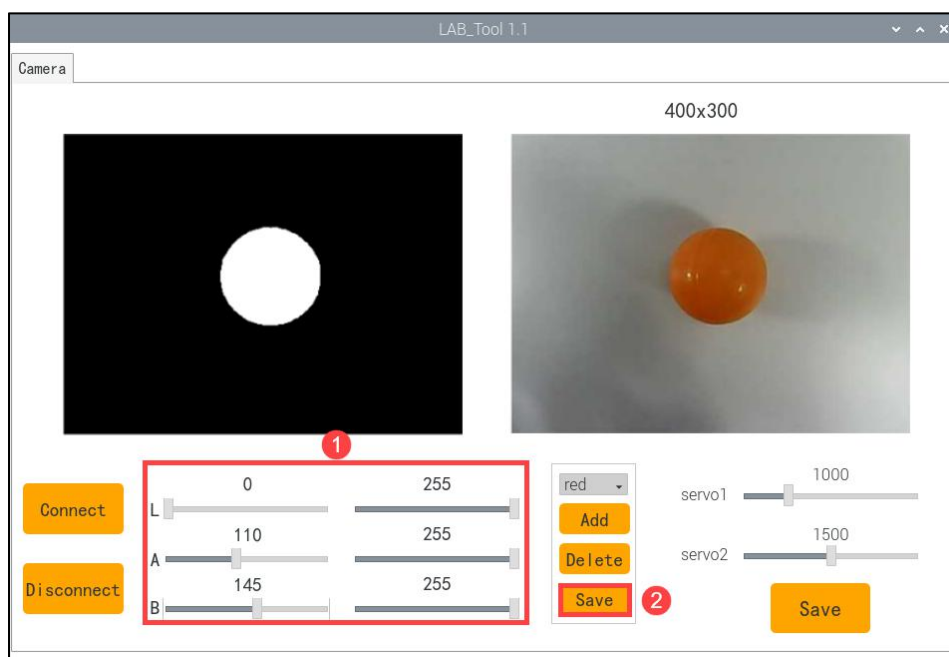


3) Click “Connect” button in the lower left hand. When the interface display the camera returned image, the connection is successful. Select "red" in the right box first.



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.

```
1 servo1 = 1000
2 servo2 = 1509
3 color_range = {
4   'red': [(0, 110, 145), (255, 255, 255)],
5   'green': [(70, 0, 0), (255, 117, 255)],
6   'blue': [(0, 0, 0), (255, 255, 115)],
7   'black': [(0, 0, 0), (41, 255, 136)],
8   'white': [(193, 0, 0), (255, 250, 255)],
9 }
```


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) 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 Instructions

5.1 Color Detection Parameter

In motion tracking program, the detected object color is red.

```
255 if __name__ == '__main__':  
256     init()  
257     start()  
258     target_color = ('red',)  
259     open_once = yaml_handle.get_yaml_data("/boot/camera_setting.yaml")["open_once"]  
260     if open_once:
```

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:

```
178 frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)  
179 frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)  
180 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:

```
187 frame_mask = cv2.inRange(frame_lab,
188                             (lab_data[i]['min'][0],
189                             lab_data[i]['min'][1],
190                             lab_data[i]['min'][2]),
191                             (lab_data[i]['max'][0],
192                             lab_data[i]['max'][1],
193                             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 process 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 .

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

```
130 for c in contours: # 遍历所有轮廓
131     contour_area_temp = math.fabs(cv2.contourArea(c)) # 计算轮廓面积
132     if contour_area_temp > contour_area_max:
133         contour_area_max = contour_area_temp
134         if contour_area_temp >= 100: # 只有在面积大于设定值时，最大面积的轮廓才是有效的，以过滤干扰
135             area_max_contour = c
136
137     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 robot recognizes the colored object, draw its contour through `cv2.drawContours()` function, as the figure shown below:

```

200 box = np.int0(cv2.boxPoints(rect))#最小外接矩形的四个顶点
201 for j in range(4):
202     box[j, 0] = int(Misc.map(box[j, 0], 0, size[0], 0, img_w))
203     box[j, 1] = int(Misc.map(box[j, 1], 0, size[1], 0, img_h))
204
205 cv2.drawContours(img, [box], -1, (0,255,255), 2)#画出四个点组成的矩形
206 #获取矩形的对角点

```

The first parameter “img” is the input image.

The second parameter “[box]” is the contour itself, which is list in Python.

The third parameter “-1” is the index of contour, where the value represents all contours in the drawn contour list.

The fourth parameter “(0, 255, 255)” is the contour color and the order of parameters is B, G, R. The color here is yellow.

The fifth parameter “2” is the width of contour. If it is “-1”, which means that the contour is filled with specified color.

- 2) After the robot recognizes object, `cv2.circle()` function is used to draw the center point of the object in the returned image, as the figure shown below:

```

207 ptime_start_x, ptime_start_y = box[0, 0], box[0, 1]
208 pt3_x, pt3_y = box[2, 0], box[2, 1]
209 radius = abs(ptime_start_x - pt3_x)
210 centerX, centerY = int((ptime_start_x + pt3_x) / 2), int((ptime_start_y + pt3_y) / 2)#中心点
211 cv2.circle(img, (centerX, centerY), 5, (0, 255, 255), -1)#画出中心点
212

```

The first parameter “img” is the input image. Here is the image of the recognized object.

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.

The fourth parameter "(0, 255, 255)" is the color of drawn circle and its order is B,G and then R. Here is yellow.

The fifth parameter "-1" is to fill the circle with the color in parameter 4. If it is a number, it means the line width of the drawn circle.

5.3 Execute Action Parameter

When the red object is recognized, control servo 1 and servo 2 to make the camera follow the movement of the red color, as the figure shown below:

```
234 x_dis = servo2 + 400 if x_dis > servo2 + 400 else x_dis
235
236 y_pid.SetPoint = img_h/2
237 y_pid.update(centerY)
238 dy = int(y_pid.output)
239 use_time = round(max(use_time, abs(dy*0.00025)), 5)
240 y_dis += dy
241
242 y_dis = servo1 if y_dis < servo1 else y_dis
243 y_dis = 2000 if y_dis > 2000 else y_dis
244
245 Board.setPWMServoPulse(1, y_dis, use_time*1000)
246 Board.setPWMServoPulse(2, x_dis, use_time*1000)
247 time.sleep(use_time)
248 else:
249 centerX, centerY = -1, -1
```

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

The first parameter "1" is the ID number of controlled servo.

The second parameter "y_dis" indicates the pulse width.

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

When the red ball is recognized, the robot will call action group in folder
“/home/pi/TonyPi/ActionGroups” , which controls robot to follow the movement
of the red object, as the figure shown below:

```
141 #执行动作组
142 def move():
143
144     while True:
145         if __isRunning:
146             if centerX >= 0:
147                 if centerX - CENTER_X > 100 or x_dis - servo2 < -80: # 不在中心, 根据方向让机器人转向一步
148                     AGC.runActionGroup('turn_right_small_step')
149                 elif centerX - CENTER_X < -100 or x_dis - servo2 > 80:
150                     AGC.runActionGroup('turn_left_small_step')
151                 elif 100 > circle_radius > 0:
152                     AGC.runActionGroup('go_forward')
153                 elif 180 < circle_radius:
154                     AGC.runActionGroup('back_fast')
155             else:
156                 time.sleep(0.01)
157         else:
158             time.sleep(0.01)
```