

Lesson 4 Line Follow

1. Program Logic

Line tracking is common in robot competitions which is implemented by two-channel or four-channel line-tracking sensors. However, TonyPi only need the vision module to recognize the line color, process by image algorithms, to realize the line follow.

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 feedback of the line position in the image, and program the robot to follow the line to achieve the function of intelligent line tracking.

The source code of the program is located in:

/home/pi/TonyPi/Functions/VisualPatrol.py

```

152 # Divide the image into three parts: upper, middle and lower so that the processing speed will be faster and more accurate
153 for r in roi:
154     roi_h = roi_h_list[n]
155     n += 1
156     blobs = frame_gb[r[0]:r[1], r[2]:r[3]]
157     frame_lab = cv2.cvtColor(blobs, cv2.COLOR_BGR2LAB) # convert images to LAB space
158
159     area_max = 0
160     areaMaxContour = 0
161     for i in lab_data:
162         if i in __target_color:
163             detect_color = i
164             frame_mask = cv2.inRange(frame_lab,
165                                     (lab_data[i]['min'][0],
166                                      lab_data[i]['min'][1],
167                                      lab_data[i]['min'][2]),
168                                     (lab_data[i]['max'][0],
169                                      lab_data[i]['max'][1],
170                                      lab_data[i]['max'][2])) # perform bit operation on the original image and mask
171             eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #erode
172             dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #dilate
173             dilated[:, 0:160] = 0
174             dilated[:, 480:640] = 0
175             cnts = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_L1)[-2]#find all contour
176             cnt_large, area = getAreaMaxContour(cnts)#Find the contour with maximum area
177             if cnt_large is not None:#If the contour is not empty
178                 rect = cv2.minAreaRect(cnt_large)#the minimum enclosing rectangle
179                 box = np.int0(cv2.boxPoints(rect))#the four vertices of the smallest enclosing rectangle
180                 for i in range(4):
181                     box[i, 1] = box[i, 1] + (n - 1)*roi_h + roi[0][0]
182                     box[i, 1] = int(Misc.map(box[i, 1], 0, size[1], 0, img_h))
183                 for i in range(4):
184                     box[i, 0] = int(Misc.map(box[i, 0], 0, size[0], 0, img_w))
185
186                 cv2.drawContours(img, [box], -1, (0,0,255,255), 2)#Draw the rectangle consisting of four points
187
188                 #Get the diagonal points of the rectangle
189                 pt1_x, pt1_y = box[0, 0], box[0, 1]
190                 pt3_x, pt3_y = box[2, 0], box[2, 1]
191                 center_x, center_y = (pt1_x + pt3_x) / 2, (pt1_y + pt3_y) / 2#center point
192                 cv2.circle(img, (int(center_x), int(center_y)), 5, (0,0,255), -1)#draw the center point
193
194                 center.append([center_x, center_y])
195                 #Sum the upper, middle and lower three center points according to different weights
196                 centroid_x_sum += center_x * r[4]
197                 weight_sum += r[4]

```

2. Operation Steps

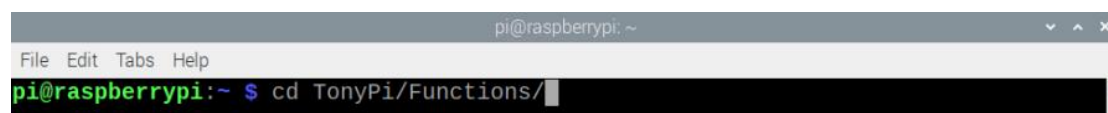
i Pay attention to the text format in the input of instructions.

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

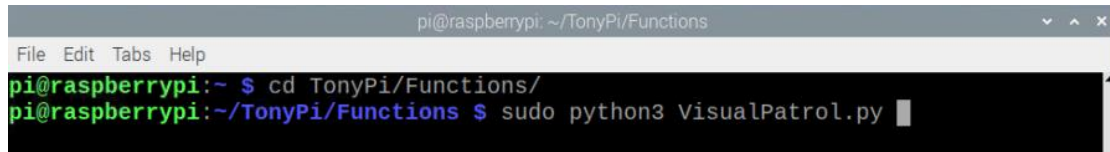
2) Click  or press “Ctrl+Alt+T” to enter the LX terminal.



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



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



```
pi@raspberrypi: ~/TonyPi/Functions
File Edit Tabs Help
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo python3 VisualPatrol.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 color is black. If you want to change to white or red, please refer to “4.1Modify Program Default Recognition Color”.

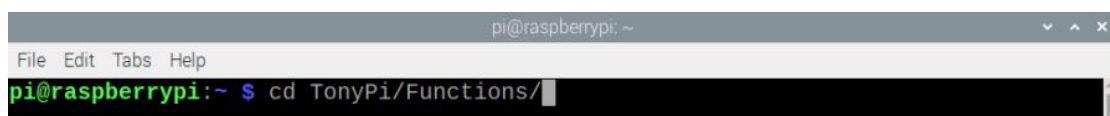
Lay the black tape and then place the robot on the line. TonyPi will move along the black track.

4. Function Extension

4.1 Modify Default Tracking Color

Black, red and white are the built-in colors in the line follow program and black is the default color. In the following steps, we’re going to modify the tracking color as white.

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 VisualPatrol.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 VisualPatrol.py
```

Step3: Input "222" and press "shfit+g" to the line for modification.

```
220     init()
221     start()
222     __target_color = ('black',)
223     open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open
    _once']
224     if open_once:
225         my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d
ummy=param.mjpg')
226     else:
227         my_camera = Camera.Camera()
228         my_camera.camera_open()
229         AGC.runActionGroup('stand')
```

Step4: Press "i" to enter the editing mode, then modify black in __target_color = ('black') to white. (if you want to recognize red, please revise to "red")

```
220     init()
221     start()
222     __target_color = ('white',)
223     open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open
    _once']
224     if open_once:
225         my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d
ummy=param.mjpg')
226     else:
227         my_camera = Camera.Camera()
228         my_camera.camera_open()
229         AGC.runActionGroup('stand')
```

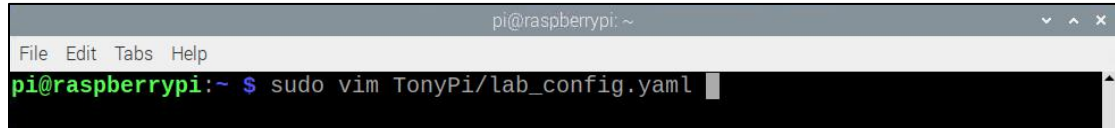
Step5: Press "Esc" to enter last line command mode. Input ":wq" to save the file and exit the editor.

```
241         time.sleep(0.01)
242         my_camera.camera_close()
243         cv2.destroyAllWindows()
:wq
```

4.2 Add Tracking Color

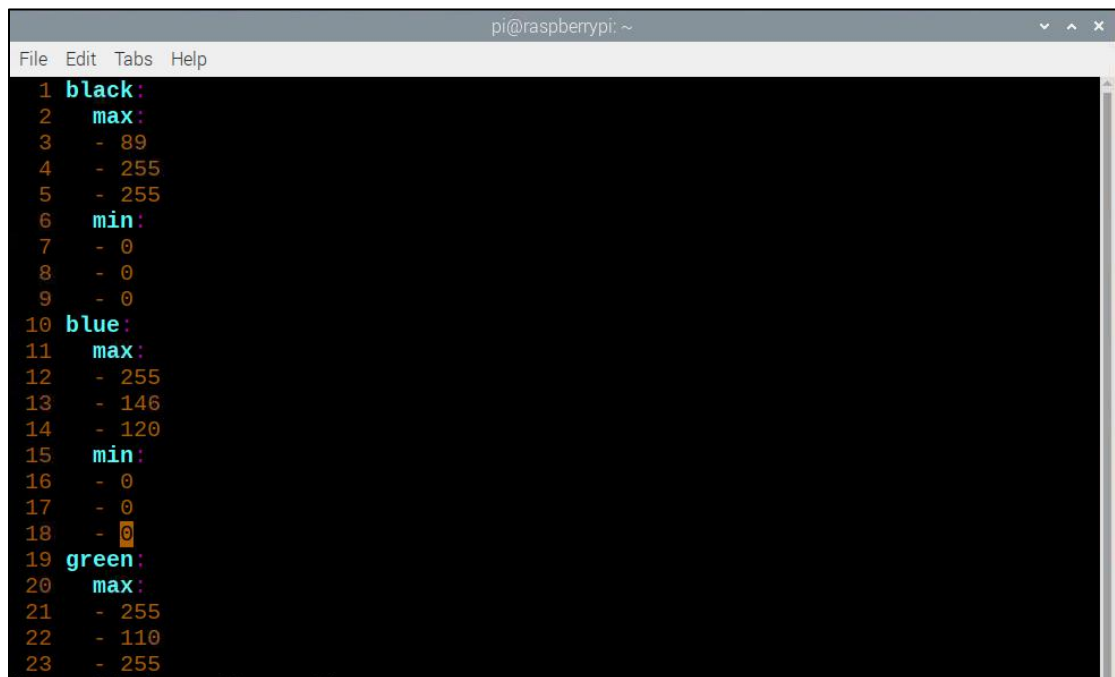
In addition to the built-in recognized colors, you can set other tracking colors in the programming. Take blue as example:

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



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo vim TonyPi/lab_config.yaml
```

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

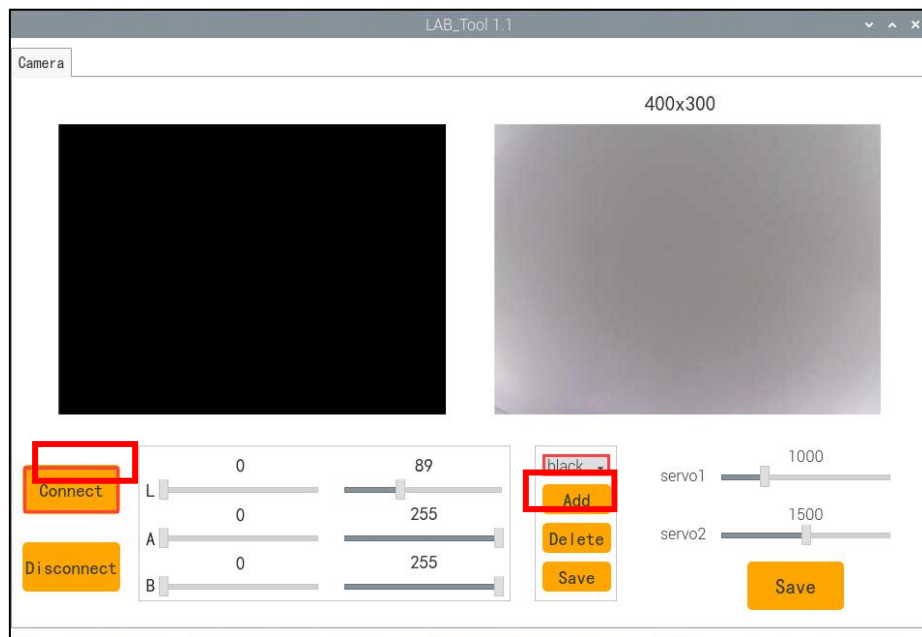


```
pi@raspberrypi: ~  
File Edit Tabs Help  
1 black:  
2   max:  
3     - 89  
4     - 255  
5     - 255  
6   min:  
7     - 0  
8     - 0  
9     - 0  
10 blue:  
11   max:  
12     - 255  
13     - 146  
14     - 120  
15   min:  
16     - 0  
17     - 0  
18     - 0  
19 green:  
20   max:  
21     - 255  
22     - 110  
23     - 255
```

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

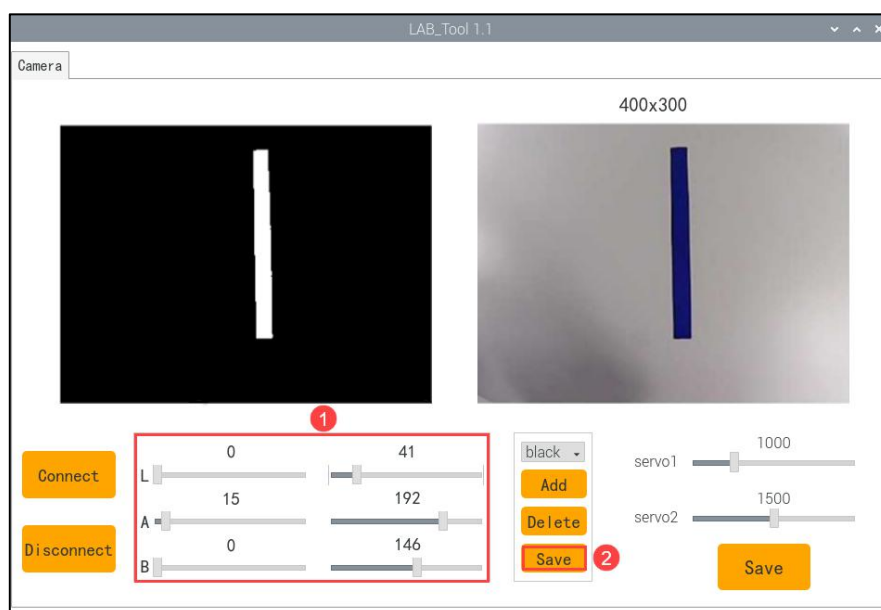


3) Click “Connect” button in the lower left hand. When the interface display the camera returned image, the connection is successful. Select "black" in the right box first.



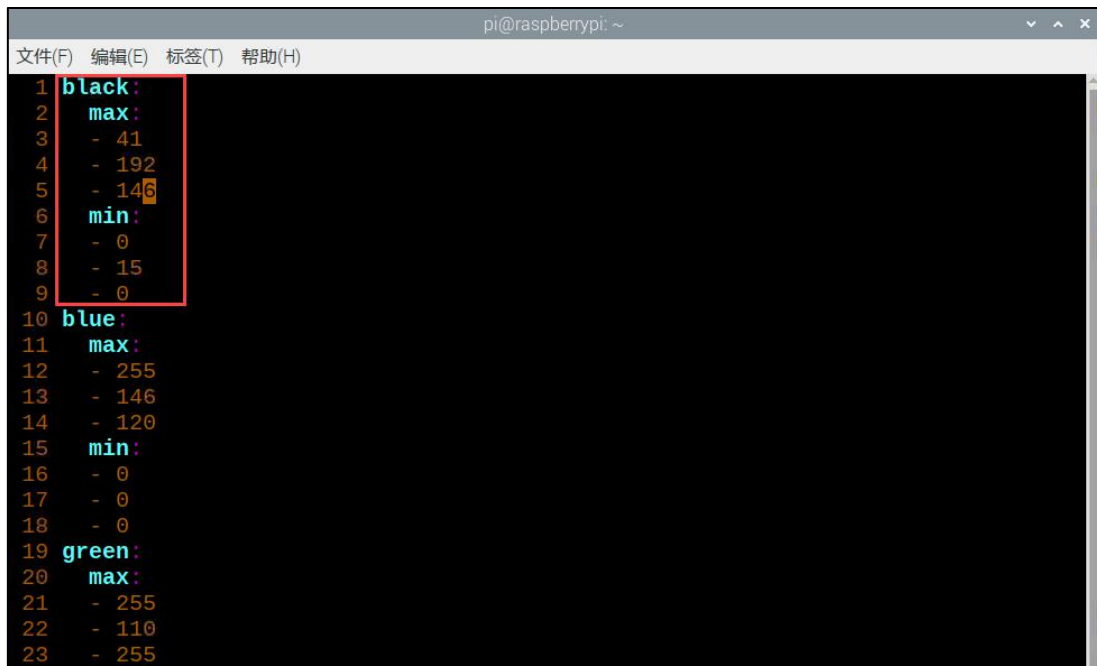
4) Drag the corresponding sliders of L, A, and B until the color area to be recognized in the left screen becomes white and other areas become black.

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.

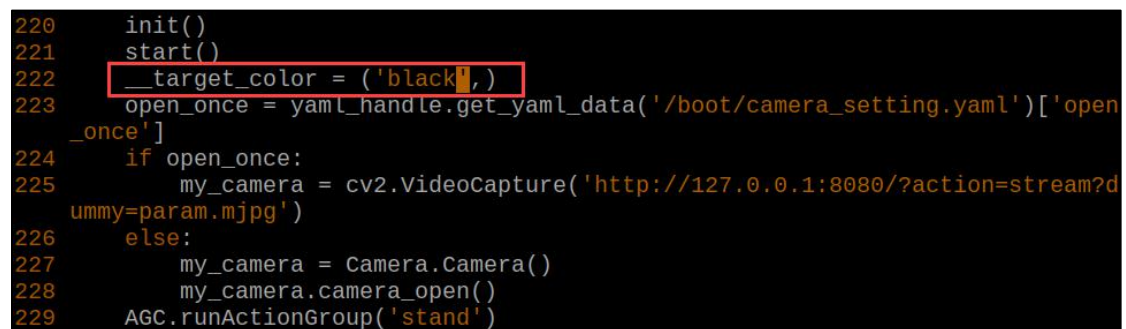
5) After the modification is completed, check whether the modified data was successfully written in. Enter the command again “`sudo vim TonyPi/lab_config.yaml`” to check the color setting parameters.



```
pi@raspberrypi: ~  
文件(F) 编辑(E) 标签(T) 帮助(H)  
1 black:  
2   max:  
3     - 41  
4     - 192  
5     - 146  
6   min:  
7     - 0  
8     - 15  
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
```

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) Set the default tracking color as black according to the “4.1Modify Default Recognition Color” in below text.



```
220     init()  
221     start()  
222     __target_color = ('black',)  
223     open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open  
_once']  
224     if open_once:  
225         my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d  
ummy=param.mjpg')  
226     else:  
227         my_camera = Camera.Camera()  
228         my_camera.camera_open()  
229     AGC.runActionGroup('stand')
```

8) Starting the game again, TonyPi will track along the blue line. 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 detected line color is red.

```
220 init()
221 start()
222 __target_color = ('red',)
223 my_camera = Camera.Camera()
224 my_camera.camera_open()
225 AGC.runActionGroup('stand')
```

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

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

```
144 frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)
145 frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
146
147 centroid_x_sum = 0
148 weight_sum = 0
149 center_ = []
150 n = 0
151
152 #将图像分割成上中下三个部分，这样处理速度会更快，更精确
153 for r in roi:
154     roi_h = roi_h_list[n]
155     n += 1
156     blobs = frame_gb[r[0]:r[1], r[2]:r[3]]
157     frame_lab = cv2.cvtColor(blobs, 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:

```
164 frame_mask = cv2.inRange(frame_lab,
165 (lab_data[i]['min'][0],
166 lab_data[i]['min'][1],
167 lab_data[i]['min'][2]),
168 (lab_data[i]['max'][0],
169 lab_data[i]['max'][1],
170 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:

```
87 for c in contours: # 遍历所有轮廓
88     contour_area_temp = math.fabs(cv2.contourArea(c)) # 计算轮廓面积
89     if contour_area_temp > contour_area_max:
90         contour_area_max = contour_area_temp
91     if contour_area_temp >= 5: # 只有在面积大于300时，最大面积的轮廓才是有效的，以过滤干扰
92         area_max_contour = c
93
94 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.drawContours() function can be used to draw the contour of red line, as the figure show below:

```
180 for i in range(4):
181     box[i, 1] = box[i, 1] + (n - 1)*roi_h + roi[0][0]
182     box[i, 1] = int(Misc.map(box[i, 1], 0, size[1], 0, img_h))
183 for i in range(4):
184     box[i, 0] = int(Misc.map(box[i, 0], 0, size[0], 0, img_w))
185
186 cv2.drawContours(img, [box], -1, (0,0,255), 2)#画出四个点组成的矩形
187
```

The first parameter “img” is the input image.

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

The third parameter “-1” is the index of contour. The value here represents all the 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 red.

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

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

```
189 pt1_x, pt1_y = box[0, 0], box[0, 1]
190 pt3_x, pt3_y = box[2, 0], box[2, 1]
191 center_x, center_y = (pt1_x + pt3_x) / 2, (pt1_y + pt3_y) / 2#中心点
192 cv2.circle(img, (int(center_x), int(center_y)), 5, (0,0,255), -1)#画出中心点
193
```

The first parameter “img” is the input image. Here it 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 “5” 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 red.

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

After recognizing the red line, robot will call action group file in folder “/home/pi/TonyPi/ActionGroups”, and then move along the red line, as the figure shown below:

```
97 def move():
98     global line_centerx
99
100     while True:
101         if __isRunning:
102             if line_centerx != -1:
103                 if abs(line_centerx - img_centerx) <= 50:
104                     AGC.runActionGroup('go_forward')
105                 elif line_centerx - img_centerx > 50:
106                     AGC.runActionGroup('turn_right_small_step')
107                 elif line_centerx - img_centerx < -50:
108                     AGC.runActionGroup('turn_left_small_step')
109             else:
110                 time.sleep(0.01)
111         else:
112             time.sleep(0.01)
```