

Lesson 2 Intelligent Transport

1. Principle

In the era of Industry 4.0, robots are widely used in the field of intelligent logistics, which can achieve high-efficiency management and improve the service. We are going to learn how TonyPi Pro robot realizes the function of AI Transport in this section.

The first stage is recognition. Program TonyPi Pro to search the recognized object on the map through walking and head rotation.

When a recognizable color appears in the visual range, TonyPi Pro starts to process the object color recognition. Convert the image to Lab, image binarization, and then perform operations such as expansion and corrosion to obtain an outline containing only the target color.

The second stage is transport. According to the processing of the image feedback information, TonyPi Pro will judge the distance of the items when multiple items appear. And then move the items according to the distance.

Set the AprilTag according to the corresponding color as the sign of the transport destination. Program the TonyPi Pro to scan on the map and detect whether is the target tag and then execute the actions. If the target tag is found, TonyPi Pro will place the item directly.

If the tag is not the target, TonyPi Pro will determine the location of the target tag based on the scanned tag, and then turn to the target until the target tag is scanned.

[The source code of the program is located in:](#)

`/home/pi/TonyPi/Functions/Transport.py`

```

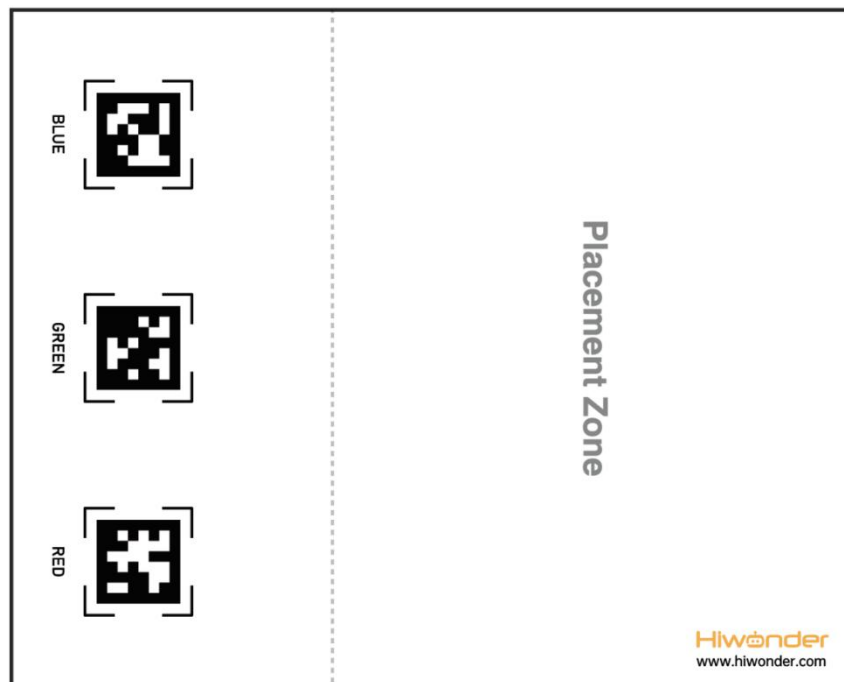
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406

else:
    step = 2
elif step == 2: # Close to object
    if 330 < object_center_y:
        AGC.runActionGroup(back, lock_serveos=lock_serveos)
    if find_box:
        if object_center_x - CENTER_X > 150: # Out of the center, turn the robot one step according to the direction
            AGC.runActionGroup(right_move_large, lock_serveos=lock_serveos)
        elif object_center_x - CENTER_X < -150:
            AGC.runActionGroup(left_move_large, lock_serveos=lock_serveos)
        elif -10 > object_angle > -45:
            AGC.runActionGroup(turn_left, lock_serveos=lock_serveos)
        elif -80 < object_angle <= -45:
            AGC.runActionGroup(turn_right, lock_serveos=lock_serveos)
        elif object_center_x - CENTER_X > 40: # Out of the center, turn the robot one step according to the direction
            AGC.runActionGroup(right_move_large, lock_serveos=lock_serveos)
        elif object_center_x - CENTER_X < -40:
            AGC.runActionGroup(left_move_large, lock_serveos=lock_serveos)
        else:
            step = 3
    else:
        if object_center_x - CENTER_X > 150: # Out of the center, turn the robot one step according to the direction
            AGC.runActionGroup(right_move_large, lock_serveos=lock_serveos)
        elif object_center_x - CENTER_X < -150:
            AGC.runActionGroup(left_move_large, lock_serveos=lock_serveos)
        elif object_angle < -5:
            AGC.runActionGroup(turn_left, lock_serveos=lock_serveos)
        elif 5 < object_angle:
            AGC.runActionGroup(turn_right, lock_serveos=lock_serveos)
        elif object_center_x - CENTER_X > 40: # Out of the center, turn the robot one step according to the direction
            AGC.runActionGroup(right_move_large, lock_serveos=lock_serveos)
        elif object_center_x - CENTER_X < -40:
            AGC.runActionGroup(left_move_large, lock_serveos=lock_serveos)
        else:
            step = 3
elif step == 3:
    if 340 < object_center_y:
        AGC.runActionGroup(back, lock_serveos=lock_serveos)
    elif 0 < object_center_y <= 250:
        AGC.runActionGroup(go_forward, lock_serveos=lock_serveos)
    elif object_center_x - CENTER_X >= 40: # Out of the center, turn the robot one step according to the direction
        AGC.runActionGroup(right_move_large, lock_serveos=lock_serveos)
    elif object_center_x - CENTER_X <= -40:
        AGC.runActionGroup(left_move_large, lock_serveos=lock_serveos)
    elif 20 <= object_center_x - CENTER_X < 40: # Out of the center, turn the robot one step according to the direction
        AGC.runActionGroup(right_move, lock_serveos=lock_serveos)
    elif -40 < object_center_x - CENTER_X < -20:
        AGC.runActionGroup(left_move, lock_serveos=lock_serveos)
    else:
        step = 4

```

2. Operation Instruction


1) The function of this section should be operated on the provided map. The right side is the items placement zone and the left side is the receiving space.




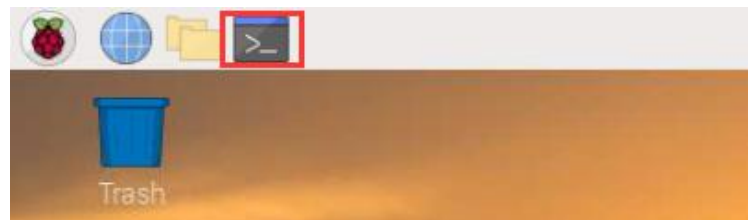
2) Place the map on the smooth floor. Place the TonyPi Pro and color blocks in the placement zone.

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

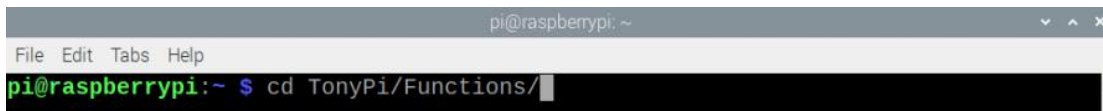
3. Operation Steps

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

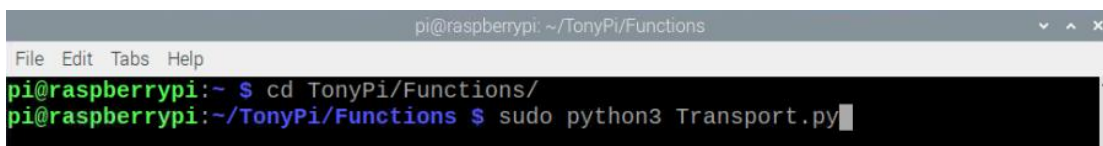
1) Click  or press “Ctrl+Alt+T” to enter the LX terminal.



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



3) Enter “sudo python3 Transport.py”, then press “Enter” to start the game.



4) 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.

4. Project Outcome

Place TonyPi Pro and blocks on the placement zone and start AI transport.

It will move the blocks to the corresponding AprilTag in order according to the distance.

5. Program Parameter Instruction

5.1 The parameter of target color and preset position

In this game, set to transport red, green and blue objects to corresponding tag position, as the figure shown below:

```
93 head_turn = 'left_right'
94 color_list = ['red', 'green', 'blue']
95 color_center_x, color_center_y = -1, -1
```

```
54 # 颜色对应的tag编号
55 color_tag = {'red': 1,
56              'green': 2,
57              'blue': 3
58              }
59
```

5.2 Find Target Object

5.2.1 Control Robot to Left or Right

When starting this game, adjust the robot to left or right to find the object to be transported, the specific operation is as follow:

```

489 if not __isRunning or stop_detect:
490     if step == 5:
491         object_center_x = 0
492     elif step == 6:
493         find_box = not find_box
494         object_center_x = -2
495         step = 1
496         stop_detect = False
497         #img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)

347 elif step == 1: # 左右调整, 保持在正中
348     x_dis = servo_data[servo2]
349     y_dis = servo_data[servo1]
350     turn = ""
351     haved_find_tag = False
352
353     if (object_center_x - CENTER_X) > 170 and object_center_y > 330:
354         AGC.runActionGroup(back, lock_servos=lock_servos)
355     elif object_center_x - CENTER_X > 80: # 不在中心, 根据方向让机器人转向一步
356         AGC.runActionGroup(turn_right, lock_servos=lock_servos)
357     elif object_center_x - CENTER_X < -80:
358         AGC.runActionGroup(turn_left, lock_servos=lock_servos)
359     elif 0 < object_center_y <= 250:
360         AGC.runActionGroup(go_forward, lock_servos=lock_servos)
361     else:
362         step = 2

```

5.2.1 Color Detection Parameter

When detect the target object, the detection purpose is achieved by detecting the color. The code is as follows:

```

500
501 color, color_center_x, color_center_y, color_angle = colorDetect(img) # 颜色检测, 返回颜色, 中心坐标, 角度
502
503 # 如果是搬运阶段
504 if find_box:
505     object_color, object_center_x, object_center_y, object_angle = color, color_center_x, color_center_y, color_angle

```

The parameters mainly involved in the process of detecting object color 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:

```

183 frame_resize = cv2.resize(img, size, interpolation=cv2.INTER_NEAREST)
184 frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
185 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)**” 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 “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:

```
191 frame_mask = cv2.inRange(frame_lab,  
192                           (lab_data[i][min][0],  
193                           lab_data[i][min][1],  
194                           lab_data[i][min][2]),  
195                           (lab_data[i][max][0],  
196                           lab_data[i][max][1],  
197                           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:


```

169 for c in contours: # 遍历所有轮廓
170     contour_area_temp = math.fabs(cv2.contourArea(c)) # 计算轮廓面积
171     if contour_area_temp > contour_area_max:
172         contour_area_max = contour_area_temp
173     if contour_area_temp >= 300: # 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
174         area_max_contour = c
175
176 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) When the robot recognizes the target object, cv2.drawContours() function can be used to draw the contour of object, as the figure show below:

```

207 box = np.int0(cv2.boxPoints(rect)) # 最小外接矩形的四个顶点
208 for j in range(4):
209     box[j, 0] = int(Misc.map(box[j, 0], 0, size[0], 0, img_w))
210     box[j, 1] = int(Misc.map(box[j, 1], 0, size[1], 0, img_h))
211
212 cv2.drawContours(img, [box], -1, (0,255,255), 2) # 画出四个点组成的矩形

```

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 yellow.

The fifth parameter “2” is the width of contour, and “-1” represents the contour is filled with specified color.

5) 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:

```

214 # 获取矩形的对角点
215 ptime_start_x, ptime_start_y = box[0, 0], box[0, 1]
216 pt3_x, pt3_y = box[2, 0], box[2, 1]
217 center_x, center_y = int((ptime_start_x + pt3_x) / 2), int((ptime_start_y + pt3_y) / 2) # 中心点
218 cv2.circle(img, (center_x, center_y), 5, (0, 255, 255), -1) # 画出中心点
219

```

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 it 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 Start Transport

After detecting the target object, the robot will start transporting. This process can be divided into several steps, which are close to the object, pick up the object, find the placement position, transport the object and put down the object.

5.3.1 Approach the object

Before transporting, control the robot to approach the object to be carried firstly, as the figure shown below:

```
503 # 如果是搬运阶段
504 if find_box:
505     object_color, object_center_x, object_center_y, object_angle = color, color_center_x, color_center_y, color_angle
506 else:

363 elif step == 2: # 接近物体
364     if 330 < object_center_y:
365         AGC.runActionGroup(back, lock_servos=lock_servos)
366     if find_box:
367         if object_center_x - CENTER_X > 150: # 不在中心, 根据方向让机器人转向一步
368             AGC.runActionGroup(right_move_large, lock_servos=lock_servos)
369         elif object_center_x - CENTER_X < -150:
370             AGC.runActionGroup(left_move_large, lock_servos=lock_servos)
371         elif -10 > object_angle > -45:
372             AGC.runActionGroup(turn_left, lock_servos=lock_servos)
373         elif -80 < object_angle <= -45:
```



```

410     step = 4
411 elif step == 4: #靠近物体
412     if 280 < object_center_y <= 340:
413         AGC.runActionGroup('go_forward_one_step', lock_servos=lock_servos)
414         time.sleep(0.2)
415     elif 0 <= object_center_y <= 280:
416         AGC.runActionGroup('go_forward', lock_servos=lock_servos)
417     else:
418         if object_center_y >= 370:
419             go_step = 2
420         else:
421             go_step = 3
422         if abs(object_center_x - CENTER_X) <= 40:
423             stop_detect = True
424             step = 5
425         else:
426             step = 3

```

5.3.2 Pick Up Object

After approaching the object, control the robot to pick it up, as the figure shown below:

```

427 elif step == 5: #拿起或者放下物体
428     if find_box:
429         AGC.runActionGroup('go_forward_one_step', times=2)
430         AGC.runActionGroup('stand', lock_servos=lock_servos)
431         AGC.runActionGroup('move_up')
432         lock_servos = LOCK_SERVOS
433         step = 6

```

5.3.3 Find Placement Position

Before transporting object, locate the placement position for the object by recognizing the tag, as the figure shown below:

```

506 else:
507     tag_data = apriltagDetect(img) # apriltag检测
508

```

The parameters mainly involved in this process are as follow:

- 1) After obtaining the information of the four corners of the tag, draw the contour of the tag through cv2.drawContours() function, as the figure shown below:

```
237 if len(detections) != 0:
238     for detection in detections:
239         corners = np.rint(detection.corners) # 获取四个角点
240         cv2.drawContours(img, [np.array(corners, np.int)], -1, (0, 255, 255), 2)
241
```

- 2) After recognizing the tag, draw the centre point of the tag in returned screen through cv2.circle() function, as the figure shown below:

```
245 object_center_x, object_center_y = int(detection.center[0]), int(detection.center[1]) # 中心点
246 cv2.circle(frame, (object_center_x, object_center_y), 5, (0, 255, 255), -1)
247
```

5.3.4 Transport Object

After picking up the object, transport it to the corresponding position, as the figure shown below:

```
509 if tag_data[color_tag[object_color] - 1][0] != -1: # 如果检测到目标arpiitag
510     object_center_x, object_center_y, object_angle = tag_data[color_tag[object_color] - 1]
```

```
427 elif step == 5: # 拿起或者放下物体
428     if find_box:
429         AGC.runActionGroup('go_forward_one_step', times=2)
430         AGC.runActionGroup('stand', lock_servos=lock_servos)
431         AGC.runActionGroup('move_up')
432         lock_servos = LOCK_SERVOS
433         step = 6
```

```
492 elif step == 6:
493     find_box = not find_box
494     object_center_x = -2
495     step = 1
496     stop_detect = False
497     #img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
498
499     return img
```

During transporting, if the target tag is not detected, the relative position is judged by other tags, as the figure shown below:

```

511 else: # 如果没有检测到目标apriltag, 就通过其他apriltag来判断相对位置
512     turn = getTurn(color_tag[object_color], tag_data)
513     if turn == 'None':
514         object_center_x, object_center_y, object_angle = -1, -1, 0
515     else: # 完全没有检测到apriltag
516         object_center_x, object_center_y, object_angle = -3, -1, 0
517

```

```

260 # 通过其他apriltag判断目标apriltag位置
261 # apriltag摆放位置: 红(tag36h11_1), 绿(tag36h11_2), 蓝(tag36h11_3)
262 def getTurn(tag_id, tag_data):
263     tag_1 = tag_data[0]
264     tag_2 = tag_data[1]
265     tag_3 = tag_data[2]
266
267     if tag_id == 1: # 目标apriltag为1
268         if tag_2[0] == -1: # 没有检测到apriltag 2
269             if tag_3[0] != -1: # 检测到apriltag 3, 则apriltag 1在apriltag 3左边, 所以左转
270                 return 'left'
271             else: # 检测到apriltag 2, 则则apriltag 1在apriltag 2左边, 所以左转
272                 return 'left'
273         elif tag_id == 2:
274             if tag_1[0] == -1:
275                 if tag_3[0] != -1:
276                     return 'left'
277             else:
278                 return 'right'
279         elif tag_id == 3:
280             if tag_1[0] == -1:
281                 if tag_2[0] != -1:
282                     return 'right'
283             else:
284                 return 'right'
285
286     return 'None'

```

5.3.5 Put Down Object

After getting to the placement position, put down the object, as the figure shown below:

```

434 else:
435     AGC.runActionGroup('go_forward_one_step', times=go_step, lock_servos=lock_servos)
436     AGC.runActionGroup('stand', lock_servos=lock_servos)
437     AGC.runActionGroup('put_down')
438     AGC.runActionGroup(back, times=5, with_stand=True)
439     color_list.remove(object_color)
440     if color_list == []:
441         color_list = ['red', 'green', 'blue']
442         lock_servos = ""
443         step = 6

```