Lesson 3 Intelligent Transport

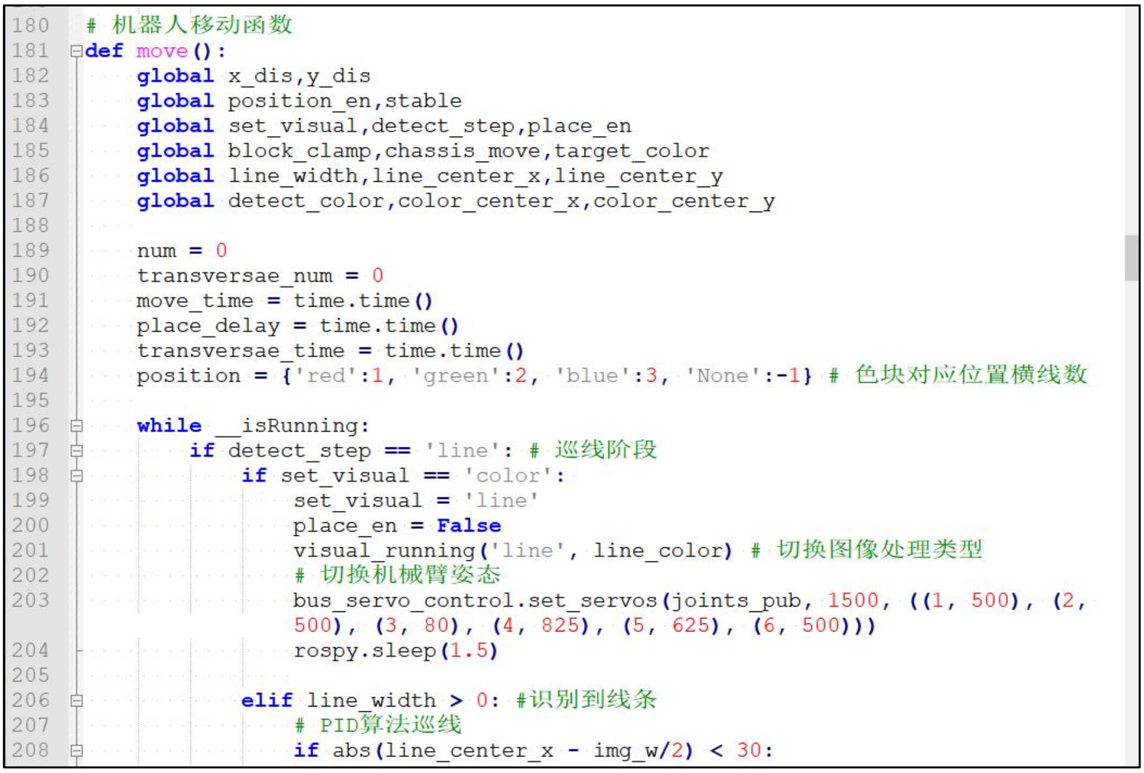
1. Working Principle

Recognize the block color first and grip it. Then read the corresponding position of the recognized color and detect the line color. After recognizing yellow line, the car will follow the line.

In the process of moving, the car will keep detecting the yellow line. When recognizing the numbers of line corresponding the colored block, the car will move to the corresponding position of color sorting. At this time, the robotic arm will put down the block and enter the next round of recognition.

The source code of program is located in:

/home/ubuntu/armpi_pro/src/intelligent_transport/scripts/intelligent_transport_node.py

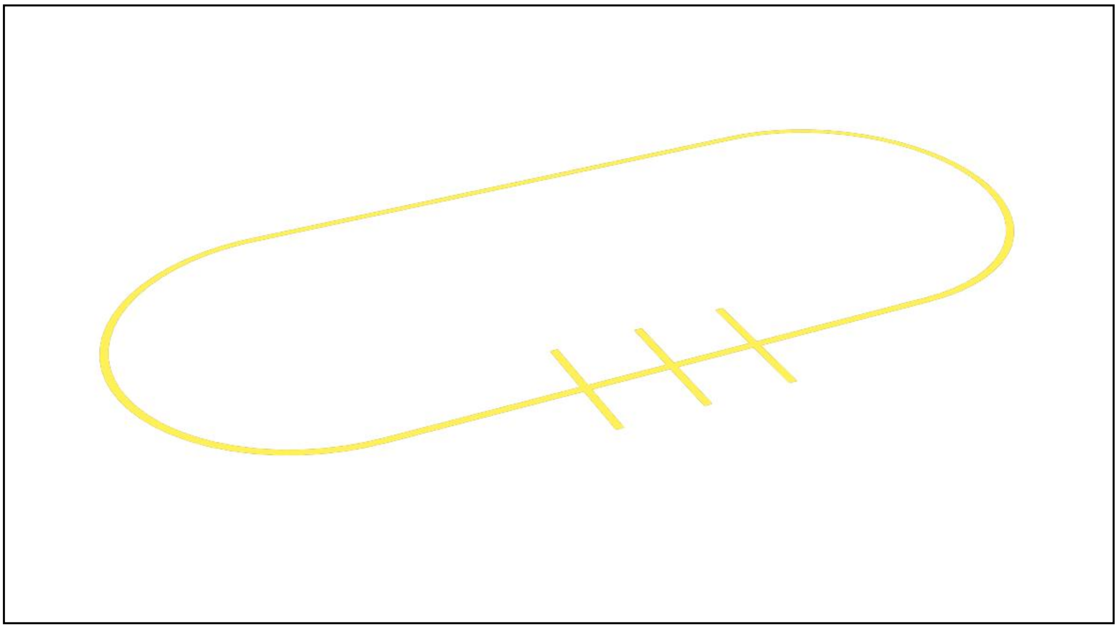


```
# 机器人移动函数
def move():
    global x_dis,y_dis
    global position_en,stable
    global set_visual,detect_step,place_en
    global block_clamp,chassis_move,target_color
    global line_width,line_center_x,line_center_y
    global detect_color,color_center_x,color_center_y
    
    num = 0
    transversae_num = 0
    move_time = time.time()
    place_delay = time.time()
    transversae_time = time.time()
    position = {'red':1, 'green':2, 'blue':3, 'None':-1} # 色块对应位置横线数
    
    while __isRunning:
        if detect_step == 'line': # 巡线阶段
            if set_visual == 'color':
                set_visual = 'line'
                place_en = False
                visual_running('line', line_color) # 切换图像处理类型
                # 切换机械臂姿态
                bus_servo_control.set_servos(joints_pub, 1500, ((1, 500), (2, 500), (3, 80), (4, 825), (5, 625), (6, 500)))
                rospy.sleep(1.5)
                
            elif line_width > 0: #识别到线条
                # PID算法巡线
                if abs(line_center_x - img_w/2) < 30:
```

2. Operation Steps

2.1 Tool Preparation

Use tape to make a map for line following, as the figure shown below:



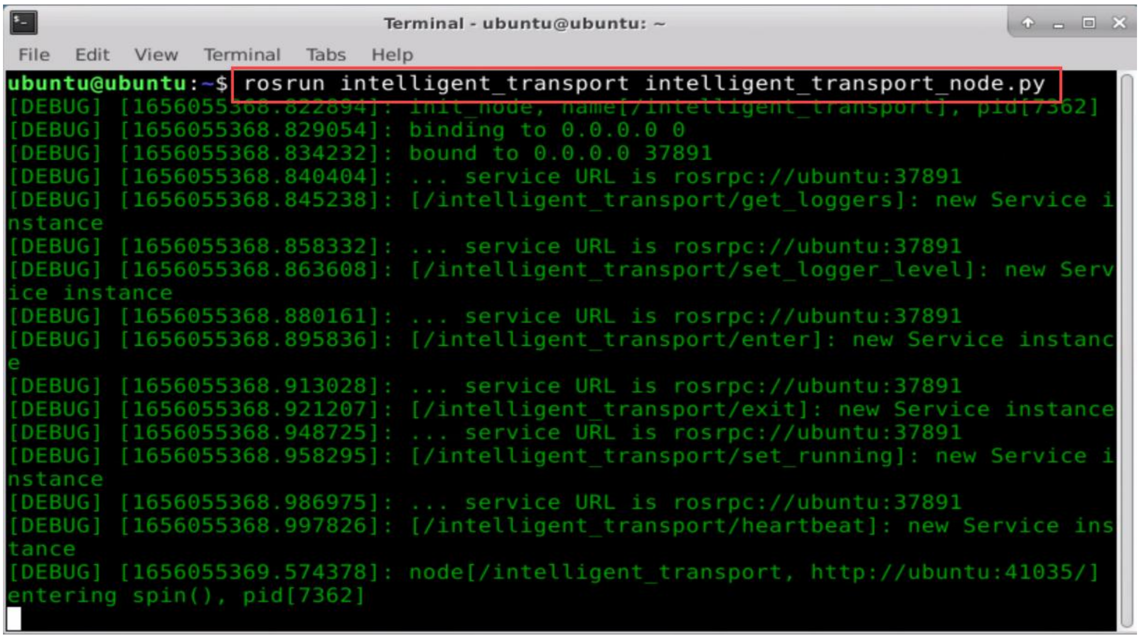
2.2 Enter Game

It should be case sensitive when entering command and the “Tab” key can be used to complete the keywords.

1) Turn on ArmPi Pro and connect to the system desktop via No Machine.

2) Click and select in pop-up interface to open the terminal.

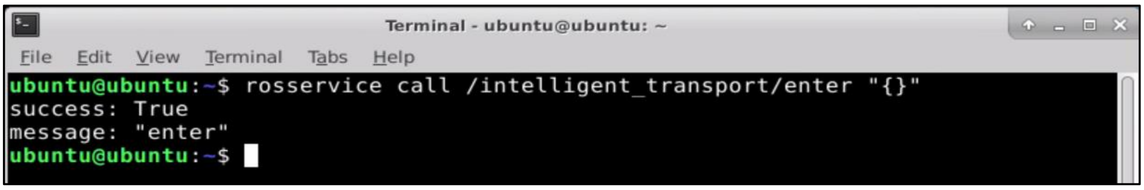
3) Enter command “**rosrun intelligent_transport intelligent_transport_node.py**” and press “Enter” to enter this game. After entering, the prompt will be printed, as the figure shown below



Terminal - ubuntu@ubuntu: ~
File Edit View Terminal Tabs Help

```
ubuntu@ubuntu:~$ rosrun intelligent_transport intelligent_transport_node.py
[DEBUG] [1656055368.822894]: init_node, name[/intelligent_transport], pid[7362]
[DEBUG] [1656055368.829054]: binding to 0.0.0.0 0
[DEBUG] [1656055368.834232]: bound to 0.0.0.0 37891
[DEBUG] [1656055368.840404]: ... service URL is rosrpc://ubuntu:37891
[DEBUG] [1656055368.845238]: [/intelligent_transport/get_loggers]: new Service i
nstance
[DEBUG] [1656055368.858332]: ... service URL is rosrpc://ubuntu:37891
[DEBUG] [1656055368.863608]: [/intelligent_transport/set_logger_level]: new Serv
ice instance
[DEBUG] [1656055368.880161]: ... service URL is rosrpc://ubuntu:37891
[DEBUG] [1656055368.895836]: [/intelligent_transport/enter]: new Service instanc
e
[DEBUG] [1656055368.913028]: ... service URL is rosrpc://ubuntu:37891
[DEBUG] [1656055368.921207]: [/intelligent_transport/exit]: new Service instance
[DEBUG] [1656055368.948725]: ... service URL is rosrpc://ubuntu:37891
[DEBUG] [1656055368.958295]: [/intelligent_transport/set_running]: new Service i
nstance
[DEBUG] [1656055368.986975]: ... service URL is rosrpc://ubuntu:37891
[DEBUG] [1656055368.997826]: [/intelligent_transport/heartbeat]: new Service ins
tance
[DEBUG] [1656055369.574378]: node[/intelligent_transport, http://ubuntu:41035/]
entering spin(), pid[7362]
```

4) Do not close the opened terminal and open a new terminal. Then enter command "**rosservice call /intelligent_transport/enter "{}"**" to enter and press "Enter" to enter this game. After entering, the terminal will print the prompt as the figure shown below:



Terminal - ubuntu@ubuntu: ~
File Edit View Terminal Tabs Help

```
ubuntu@ubuntu:~$ rosservice call /intelligent_transport/enter "{}"
success: True
message: "enter"
ubuntu@ubuntu:~$
```

2.3 Start image transmission

2.3.1 Start with browser

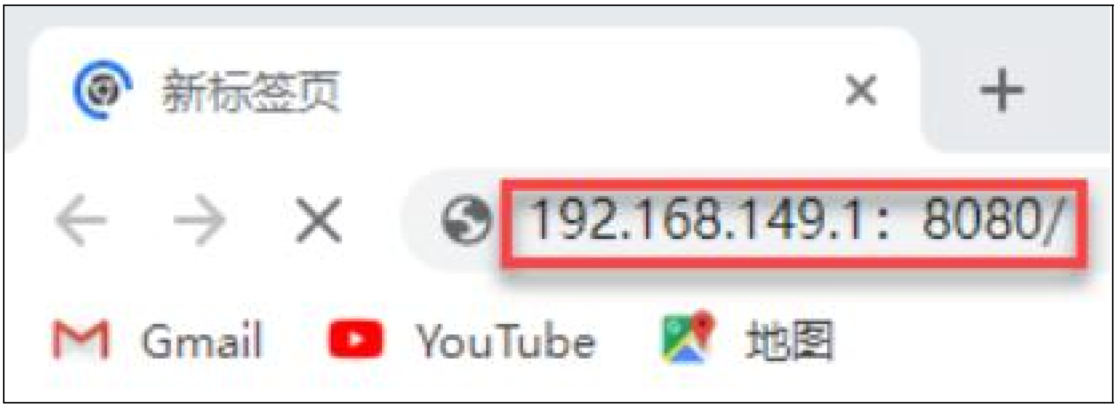
To avoid consuming too much running memory of Raspberry Pi. It is recommended to use an external browser to start image transmission.

The specific steps are as follows:

1) Select a browser. Take Google Chrome as example.

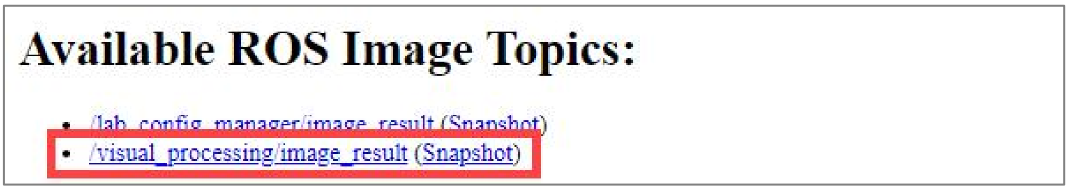
2) Then enter the default IP address "192.168.149.1:8080/" (Note: this IP address is the default IP address for direction connection mode). If it is LAN mode, please enter "Device IP address+：8080/" such as "192.168.149.1:8080/") If fail to open, you can try it several times or restart camera.

Note: If it is in LAN mode, the method to obtain device IP address can refer to "10.Advanced Lesson"/ 1.Network Configuration Lesson/ LAN Mode Connection.



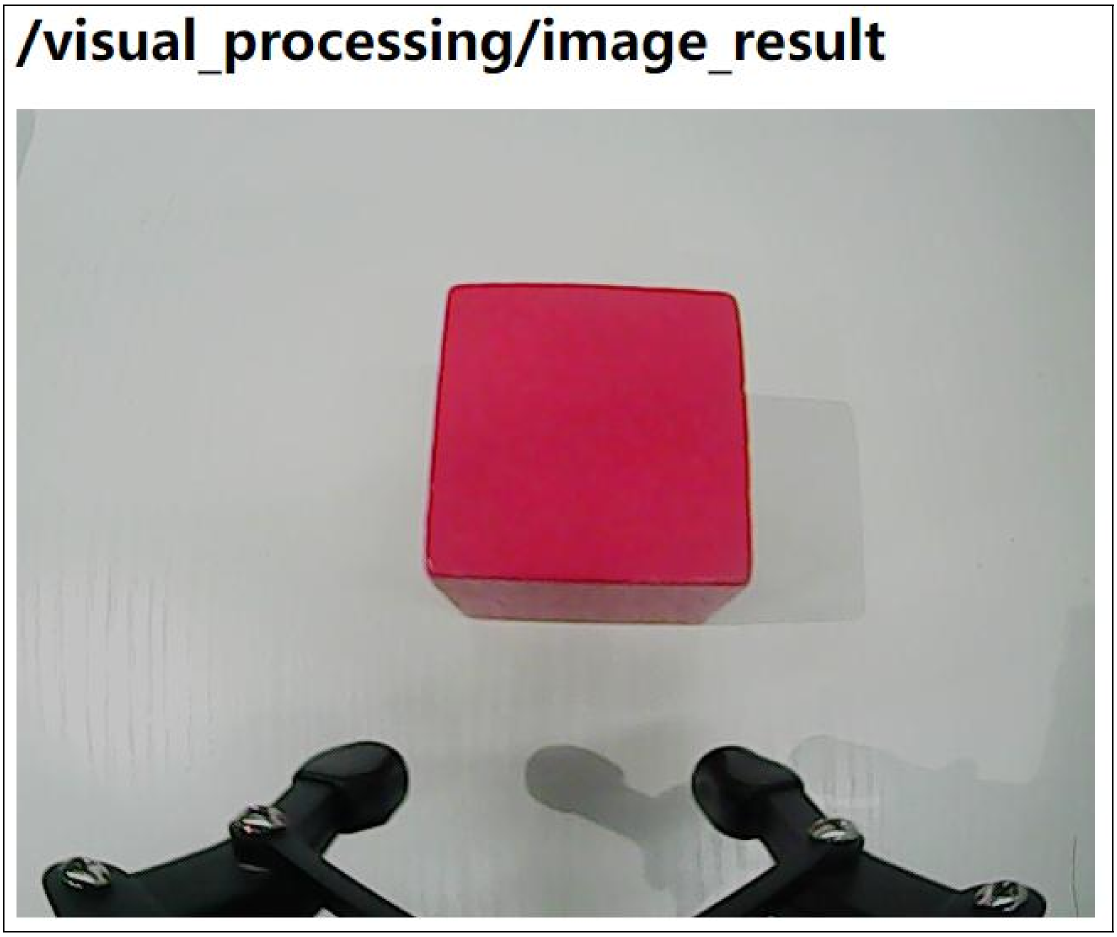
192.168.149.1： 8080/

3) Then, click the option shown in the following figure to open the display window of the transmitted image.



Available ROS Image Topics:

- /lab_config_manager/image_result (Snapshot)
- /visual_processing/image_result (Snapshot)

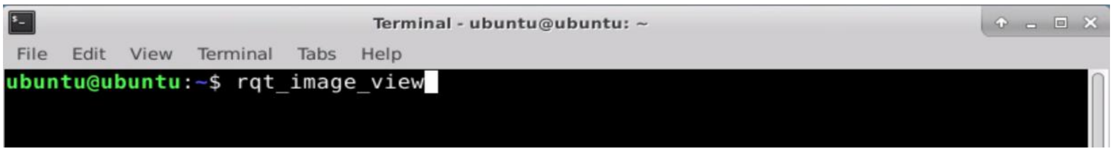


/visual_processing/image_result

2.3.2 Start with rqt

1) After completing the steps of "2.1 Enter Game" and do not exit the terminal, open a new terminal.

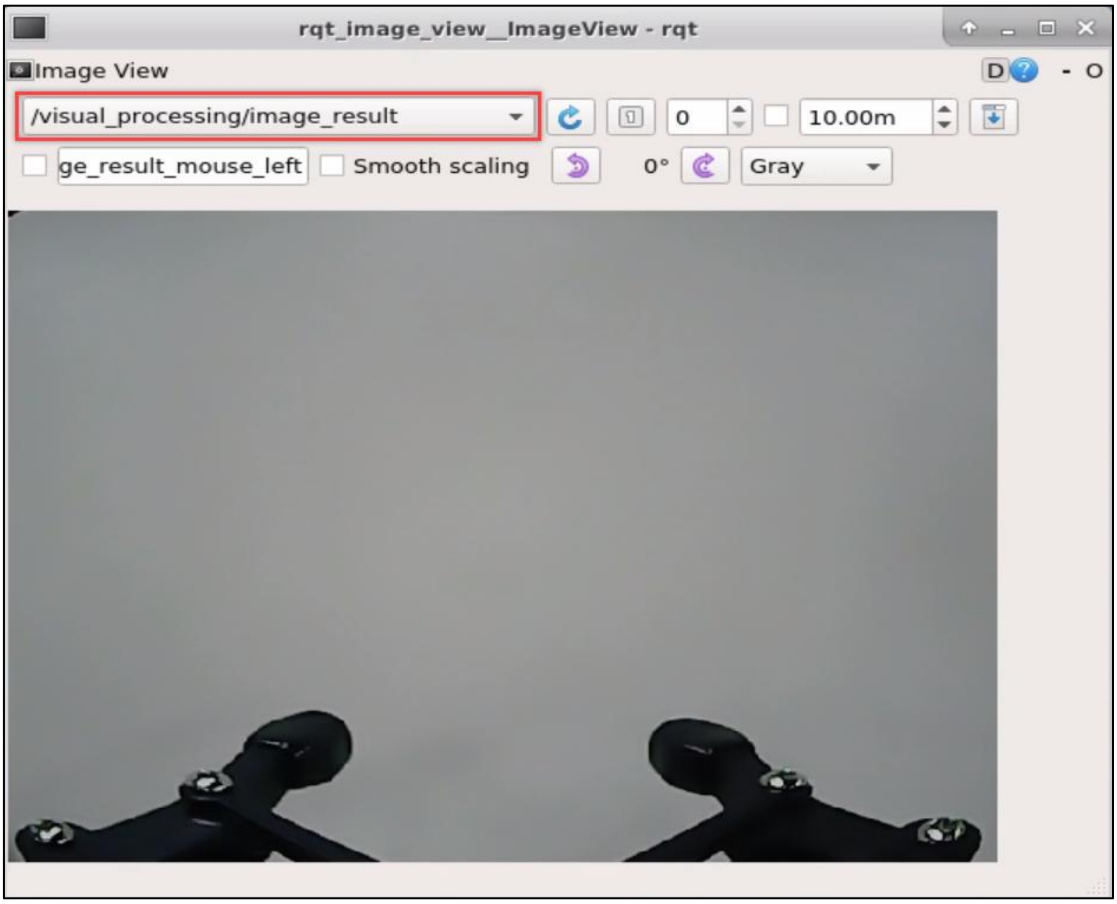
2) Enter command "rqt_image_view" and press "Enter" to open rqt.



Terminal - ubuntu@ubuntu: ~
File Edit View Terminal Tabs Help

```
ubuntu@ubuntu:~$ rqt_image_view
```

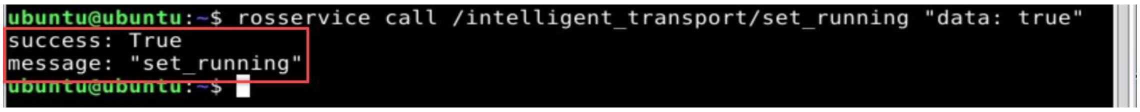
3) Click the red box as the figure shown below, select "/visual_processing/image_result" for the topic of line following and remain other settings unchanged, as the figure shown below:


rqt_image_view__ImageView - rqt
Image View
D
- O
/visual_processing/image_result
0
10.00m
ge_result_mouse_left
Smooth scaling
0°
Gray

Note: After opening image, the topic option must be selected. Otherwise, after starting game, the recognition process can not be displayed normally.

2.4 Start Game

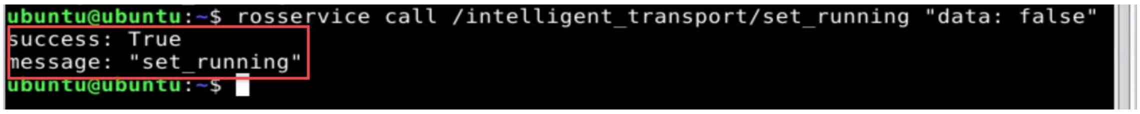
Now, enter the terminal according to the steps in “2.1 Enter Game” and input command “**rosservice call /intelligent_transport/set_running "data: true"**”. Then if the prompt shown in the following red box appears, which means game has been started successfully.



```
ubuntu@ubuntu:~$ rosservice call /intelligent_transport/set_running "data: true"
success: True
message: "set_running"
ubuntu@ubuntu:~$
```

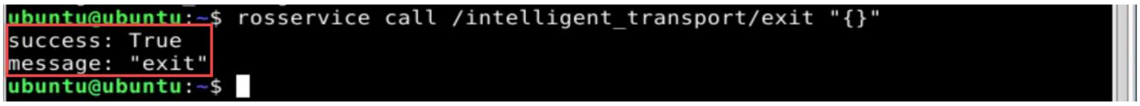
2.5 Stop and Exit

1) If want to stop the game, enter command “**rosservice call /intelligent_transport/set_running "data: false"**”.



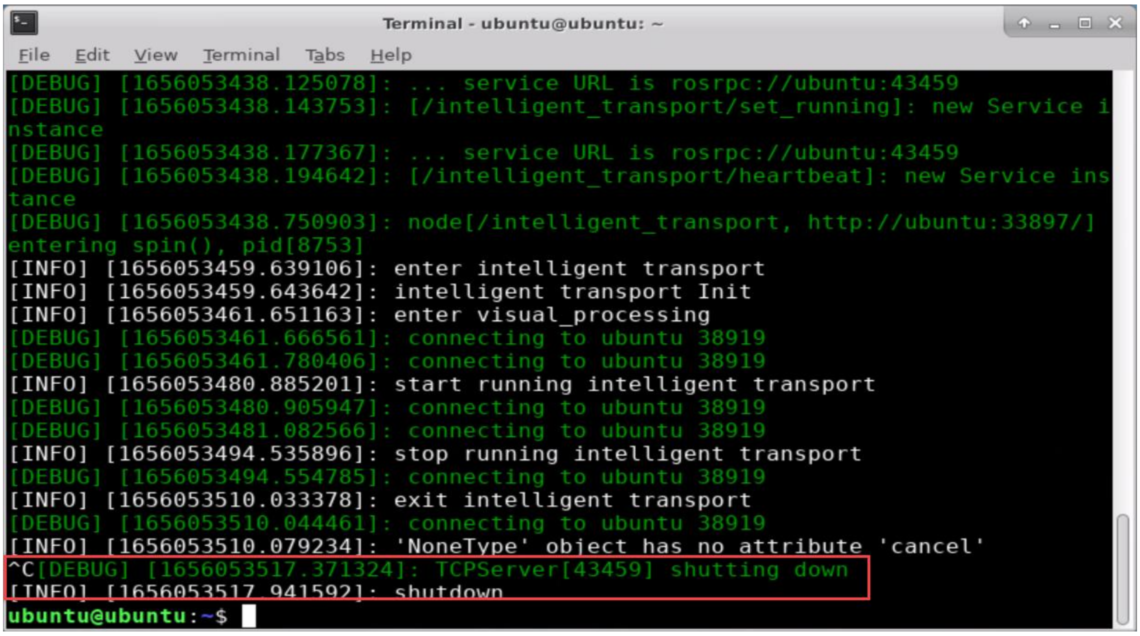
```
ubuntu@ubuntu:~$ rosservice call /intelligent_transport/set_running "data: false"
success: True
message: "set_running"
ubuntu@ubuntu:~$ 
```

2) If want to exit the game, enter command "**rosservice call /intelligent_transport/exit "{}"**" to exit.



```
ubuntu@ubuntu:~$ rosservice call /intelligent_transport/exit "{}"
success: True
message: "exit"
ubuntu@ubuntu:~$ 
```

3) To avoid consume too much running memory of Raspberry Pi, after exiting the game and returning to the terminal of running game programmings, press "Ctrl+C" to exit the program. If fail to exit, please keep trying several times.

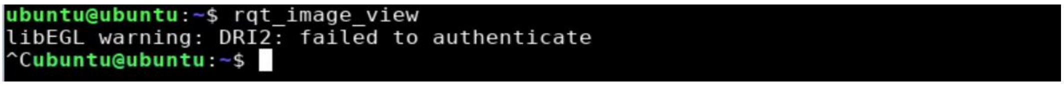


Terminal - ubuntu@ubuntu: ~
File Edit View Terminal Tabs Help

```
[DEBUG] [1656053438.125078]: ... service URL is rosrpc://ubuntu:43459
[DEBUG] [1656053438.143753]: [/intelligent_transport/set_running]: new Service i
nstance
[DEBUG] [1656053438.177367]: ... service URL is rosrpc://ubuntu:43459
[DEBUG] [1656053438.194642]: [/intelligent_transport/heartbeat]: new Service ins
tance
[DEBUG] [1656053438.750903]: node[/intelligent_transport, http://ubuntu:33897/]
entering spin(), pid[8753]
[INFO] [1656053459.639106]: enter intelligent transport
[INFO] [1656053459.643642]: intelligent transport Init
[INFO] [1656053461.651163]: enter visual_processing
[DEBUG] [1656053461.666561]: connecting to ubuntu 38919
[DEBUG] [1656053461.780406]: connecting to ubuntu 38919
[INFO] [1656053480.885201]: start running intelligent transport
[DEBUG] [1656053480.905947]: connecting to ubuntu 38919
[DEBUG] [1656053481.082566]: connecting to ubuntu 38919
[INFO] [1656053494.535896]: stop running intelligent transport
[DEBUG] [1656053494.554785]: connecting to ubuntu 38919
[INFO] [1656053510.033378]: exit intelligent transport
[DEBUG] [1656053510.044461]: connecting to ubuntu 38919
[INFO] [1656053510.079234]: 'NoneType' object has no attribute 'cancel'
^C[DEBUG] [1656053517.371324]: TCPServer[43459] shutting down
[INFO] [1656053517.941592]: shutdown
ubuntu@ubuntu:~$ 
```

Note: Before exiting the game, it will keep running when Raspberry Pi is powered on. To avoid consume too much running memory of Raspberry Pi, you need to exit the game first according to the operation steps above before performing other AI vision games.

4) If want to close the image transmission, press "Ctrl+C" to return and open the terminal of rqt. If fail to exit, please keep trying several times.



```
ubuntu@ubuntu:~$ rqt_image_view
libEGL warning: DRI2: failed to authenticate
^Cubuntu@ubuntu:~$ 
```

3. Project Outcome

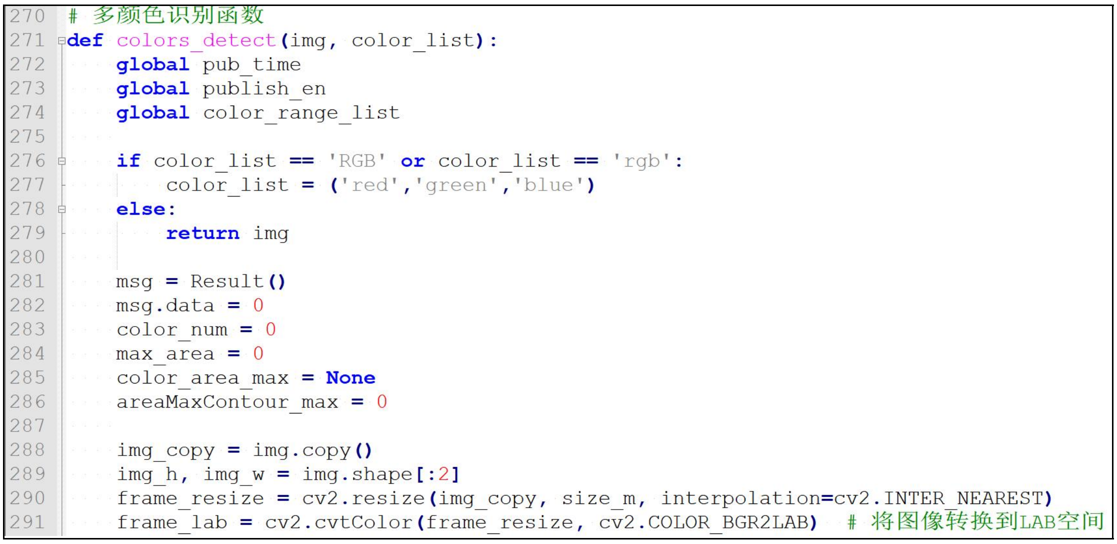
After starting the game, hold the block within the detected range of camera. When the block is recognized by ArmPi Pro, it will grip it and keep following line. Then place the block with different colors to the corresponding position.

4. Program Parameter Instruction

4.1 Image Process

The source code of image process program is located in:

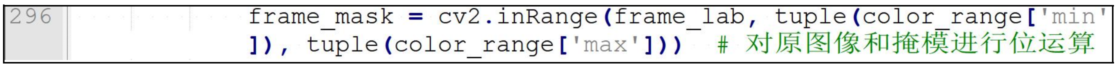
/home/ubuntu/armpi_pro/src/visual_processing/scripts/visual_processing_node.py



```
# 多颜色识别函数
def colors_detect(img, color_list):
    global pub_time
    global publish_en
    global color_range_list
    
    if color_list == 'RGB' or color_list == 'rgb':
        color_list = ('red','green','blue')
    else:
        return img
    
    msg = Result()
    msg.data = 0
    color_num = 0
    max_area = 0
    color_area_max = None
    areaMaxContour_max = 0
    
    img_copy = img.copy()
    img_h, img_w = img.shape[:2]
    frame_resize = cv2.resize(img_copy, size_m, interpolation=cv2.INTER_NEAREST)
    frame_lab = cv2.cvtColor(frame_resize, cv2.COLOR_BGR2LAB)  # 将图像转换到LAB空间
```

4.1.1 Binarization

Use the inRange () function in the cv2 library to binarize the image.



```
            frame_mask = cv2.inRange(frame_lab, tuple(color_range['min']), tuple(color_range['max']))  # 对原图像和掩模进行位运算
```

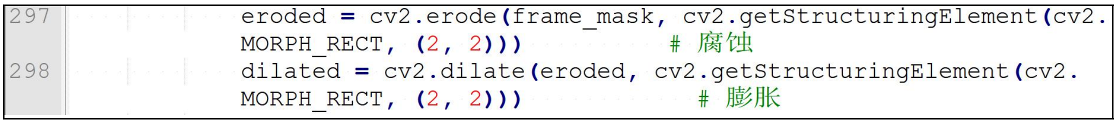
The first parameter “frame_lab” is the input image.

The second parameter “**tuple(color_range['min'])**” is the lower limit of threshold.

The third parameter “**tuple(color_range['max'])**” is the upper lower of threshold.

4.1.2 Dilation and Erosion

To lower interference and make image smoother, the image needs to be dilated and eroded.



```
            eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2)))          # 腐蚀
            dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2)))            # 膨胀
```

erode() function is applied to erode image. Take code “**eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)))**” as example. The meaning of parameters in parentheses are as follow:

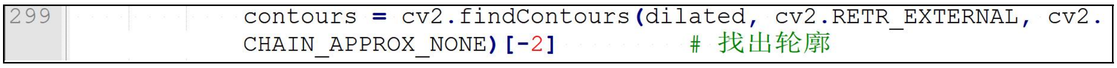
The first parameter “**frame_mask**” is the input image.

The second parameter “**cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2))**” is the structural elements and kernel that determines the nature of operation. The first parameter in parentheses is the shape of kernel and the second parameter is the size of kernel.

dilate() function is applied to dilate image. The meaning of parameters in parentheses is the same as the parameters of erode() function.

4.1.3 Obtain the contour of the maximum area

After processing the above image, obtain the contour of the recognition target. The findContours() function in cv2 library is involved in this process.



```
            contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]          # 找出轮廓
```

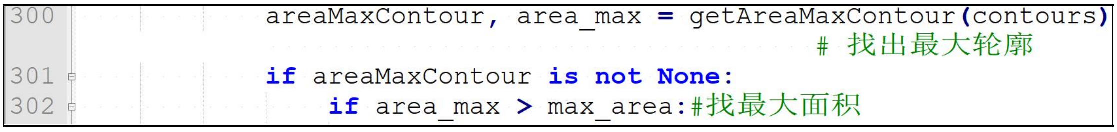
The erode() function is applied to erode. Take code “**contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]**” as example.

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

The second parameter “**cv2.RETR_EXTERNAL**” is the contour retrieval mode.

The third parameter “**cv2.CHAIN_APPROX_NONE)[-2]**” is the approximate method of contour.

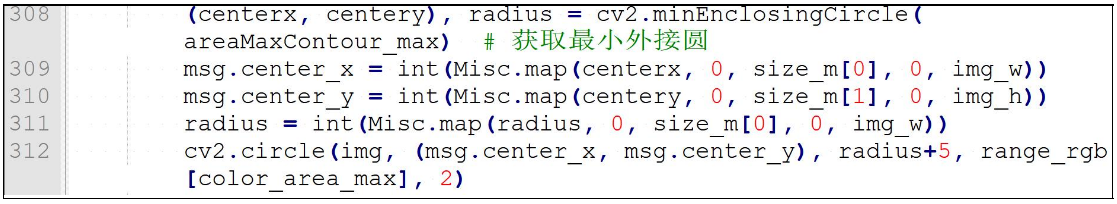
Find the maximum contour from the obtained contours. To avoid interference, set a minimum value. Only when the area is greater than this minimum value, the target contour will take effect. The minimum value here is “50”.



```
            areaMaxContour, area_max = getAreaMaxContour(contours)  # 找出最大轮廓
            if areaMaxContour is not None:
                if area_max > max_area:#找最大面积
```

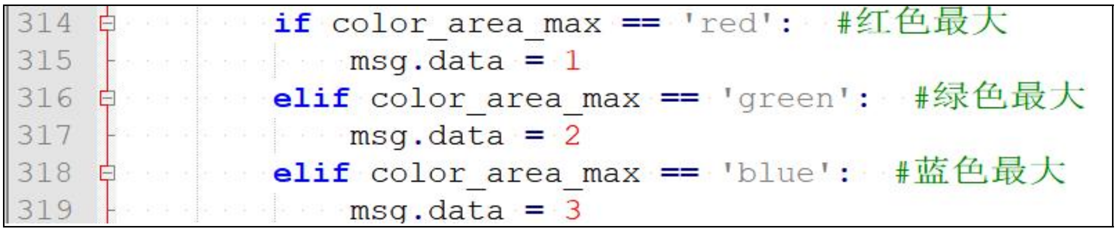
4.1.4 Obtain Position Information

The minAreaRect() function in cv2 library is used to obtain the minimum external rectangle of the target contour, and the coordinates of its four vertices are obtained through the boxPoints() function. Then, the coordinates of the center point of the rectangle can be calculated from the coordinates of the vertexes of the rectangle.



```
        (centerx, centery), radius = cv2.minEnclosingCircle(areaMaxContour_max)  # 获取最小外接圆
        msg.center_x = int(Misc.map(centerx, 0, size_m[0], 0, img_w))
        msg.center_y = int(Misc.map(centery, 0, size_m[1], 0, img_h))
        radius = int(Misc.map(radius, 0, size_m[0], 0, img_w))
        cv2.circle(img, (msg.center_x, msg.center_y), radius+5, range_rgb[color_area_max], 2)
```

Determine the color block with the largest area.

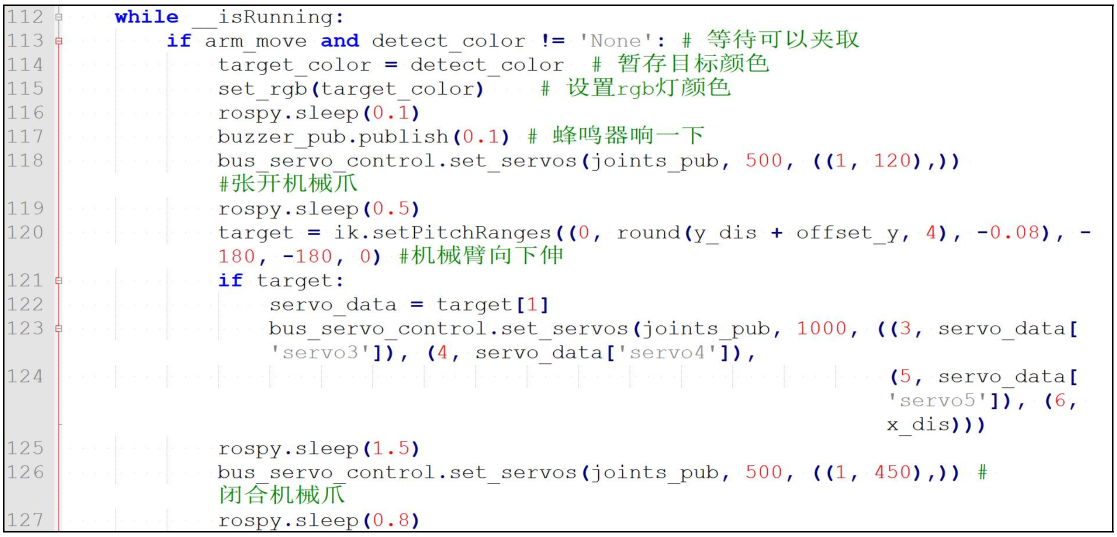


```
        if color_area_max == 'red':  #红色最大
            msg.data = 1
        elif color_area_max == 'green':  #绿色最大
            msg.data = 2
        elif color_area_max == 'blue':  #蓝色最大
            msg.data = 3
```

4.2 Function Realization

4.2.1Grip the block

The position of the target on x, y and z axes is obtained after processing image. Then get the target position calculated by inverse kinematics and grip it.



```
    while __isRunning:
        if arm_move and detect_color != 'None': # 等待可以夹取
            target_color = detect_color  # 暂存目标颜色
            set_rgb(target_color)    # 设置rgb灯颜色
            rospy.sleep(0.1)
            buzzer_pub.publish(0.1) # 蜂鸣器响一下
            bus_servo_control.set_servos(joints_pub, 500, ((1, 120),)) #张开机械爪
            rospy.sleep(0.5)
            target = ik.setPitchRanges((0, round(y_dis + offset_y, 4), -0.08), -180, -180, 0) #机械臂向下伸
            if target:
                servo_data = target[1]
                bus_servo_control.set_servos(joints_pub, 1000, ((3, servo_data['servo3']), (4, servo_data['servo4']),
                                                                                (5, servo_data['servo5']), (6, x_dis)))
            rospy.sleep(1.5)
            bus_servo_control.set_servos(joints_pub, 500, ((1, 450),)) # 闭合机械爪
            rospy.sleep(0.8)
```

The inverse kinematics takes "**ik.setPitchRanges((0, round(y_dis + offset_y, 4), -0.08), -180, -180, 0)**" as example and the meaning of parameters in parentheses are as follow:

The first parameter is "**(0, round(y_dis + offset_y, 4)**". "0" is the position of the target on x-axis. "**round(y_dis, 4)**" is the position of the target on y-axis. "**round(z_dis, 4)**" is the position of the target on z-axis.

The second parameter "-180" is the angle of x-axis.

The third parameter "-180" is the range of the pitch angle.

The fourth parameter "0" is the range of pitch angle.

The servo control takes the code "bus_servo_control.set_servos(joints_pub, 20, ((3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5']), (6, x_dis)))" as example and the meaning of parameters in parentheses is as follow:

The first parameter "joints_pub" is to publish the message of servo control node.

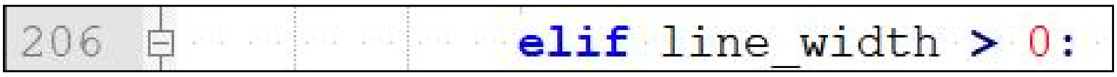
The second parameter "20" is the running time.

The third parameter is "**((3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5']), (6, x_dis)**". Among them, "3" is the servo number.

“**servo_data['servo3']**” and the rest of parameters are the servo angle.

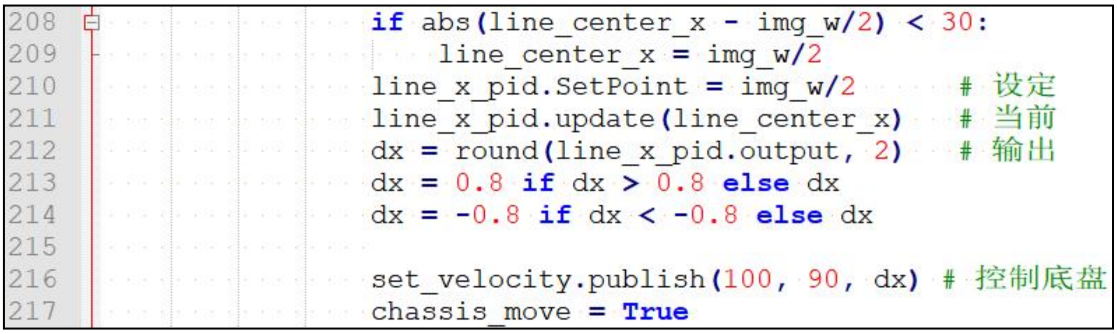
4.2.2 Follow line

After gripping the block, the car will follow the line. Firstly, judge if there is the line within the detected range. The code is shown in the following figure:



```
        elif line_width > 0:
```

Then the current x-coordinate of line subtracts the value of ideal center point. Get the yaw rate by PID mapping to adjust the speed of motor.



```
if abs(line_center_x - img_w/2) < 30:
    line_center_x = img_w/2
line_x_pid.SetPoint = img_w/2       # 设定
line_x_pid.update(line_center_x)    # 当前
dx = round(line_x_pid.output, 2)    # 输出
dx = 0.8 if dx > 0.8 else dx
dx = -0.8 if dx < -0.8 else dx

set_velocity.publish(100, 90, dx) # 控制底盘
chassis_move = True
```

Take the code “set_velocity.publish(100, 90, dx)” as example:

The first parameter “100” is the linear velocity.

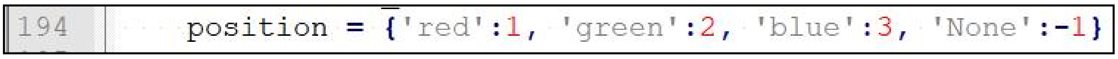
The second parameter “90” is the angular velocity.

The third parameter “dx” is the yaw rate. The larger the yaw rate, the faster the rotation speed of car.

4.2.3 Recognize the placement line

1) Determine the number of lines

In the process of identifying the color of block, we set the corresponding numbers of recognized lines for placement position of different blocks, as the figure shown below:

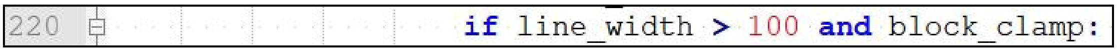


```
    position = {'red':1, 'green':2, 'blue':3, 'None':-1}
```

If the recognized color is red, the robot will run the the code for transporting the block to the first placement line when it identifies single line.

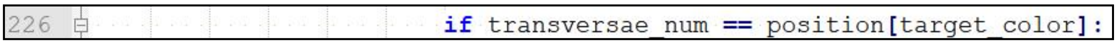
If the recognized color is green, the robot will transport the block to the second placement line when only two lines are recognized.

In the process of following line, the car will keep detecting the placement line. If the following condition is satisfied, which means the line is recognized.



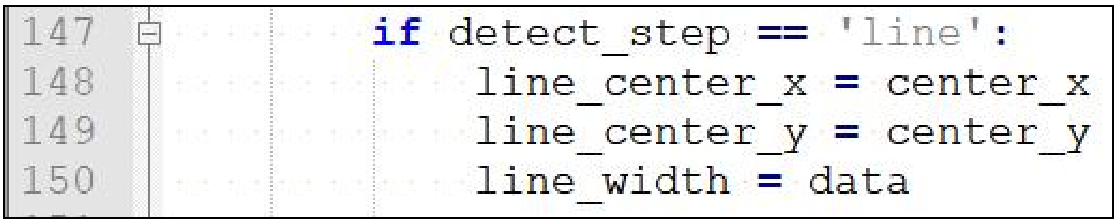
```
if line_width > 100 and block_clamp:
```

Then determine the position of placement line.



```
if transversae_num == position[target_color]:
```

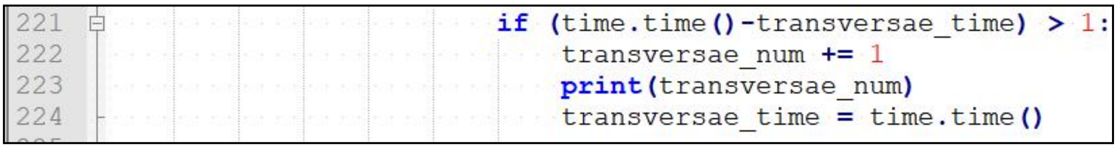
The width of the line is obtained by the following function.



```
if detect_step == 'line':
    line_center_x = center_x
    line_center_y = center_y
    line_width = data
```

2) **Stop recognizing**

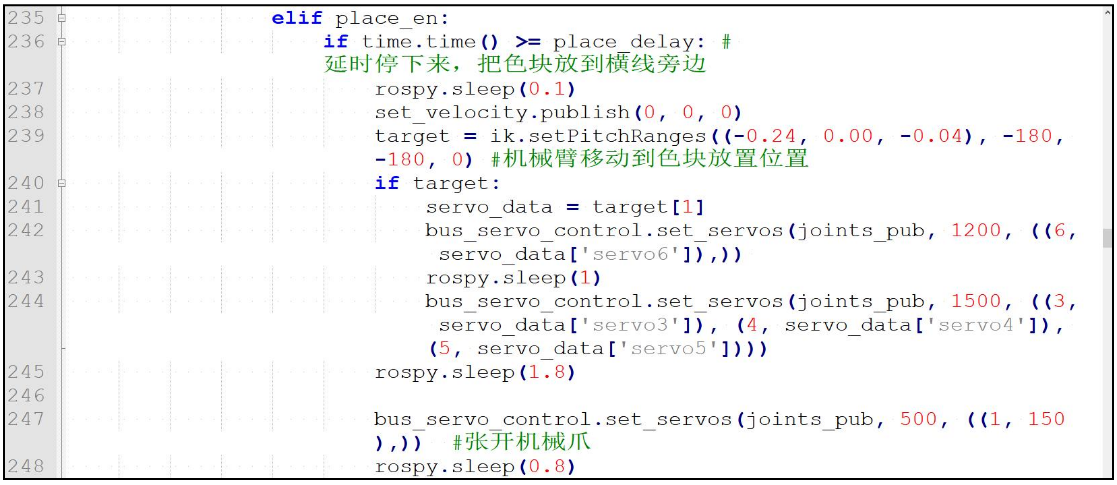
After all the lines are recognized completely, the recognition function will be stopped to prevent the interference from repeat recognition of the same placement line.



```
if (time.time()-transversae_time) > 1:
    transversae_num += 1
    print(transversae_num)
    transversae_time = time.time()
```

4.2.4 Place the block

When the numbers of recognized placement lines is equivalent to the numbers of placement lines corresponding to the placement position of the target block, the car will stop in the corresponding position and the robotic arm will be controlled to place the block to the corresponding position.



```
elif place_en:
    if time.time() >= place_delay: # 延时停下来，把色块放到横线旁边
        rospy.sleep(0.1)
        set_velocity.publish(0, 0, 0)
        target = ik.setPitchRanges((-0.24, 0.00, -0.04), -180, -180, 0) #机械臂移动到色块放置位置
        if target:
            servo_data = target[1]
            bus_servo_control.set_servos(joints_pub, 1200, ((6, servo_data['servo6']),))
            rospy.sleep(1)
            bus_servo_control.set_servos(joints_pub, 1500, ((3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5'])))
        rospy.sleep(1.8)

        bus_servo_control.set_servos(joints_pub, 500, ((1, 150),))  #张开机械爪
        rospy.sleep(0.8)
```

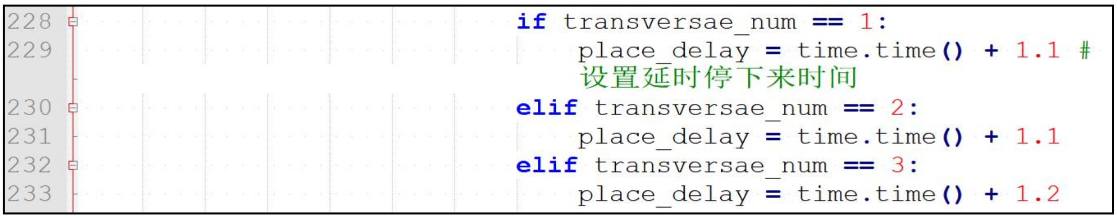
In the code shown in the figure above, the inverse kinematics is used to set the movement of robotic arm. Take code “**target = ik.setPitchRanges((-0.24, 0.00, -0.04), -180, -180, 0)**” as example:

The first parameter “**(-0.24, 0.00, -0.04)**” is the coordinate value (x,y,and z axes) of the end of robotic arm.

The second parameter “-180” is the pitch angle value of the end of robotic arm.

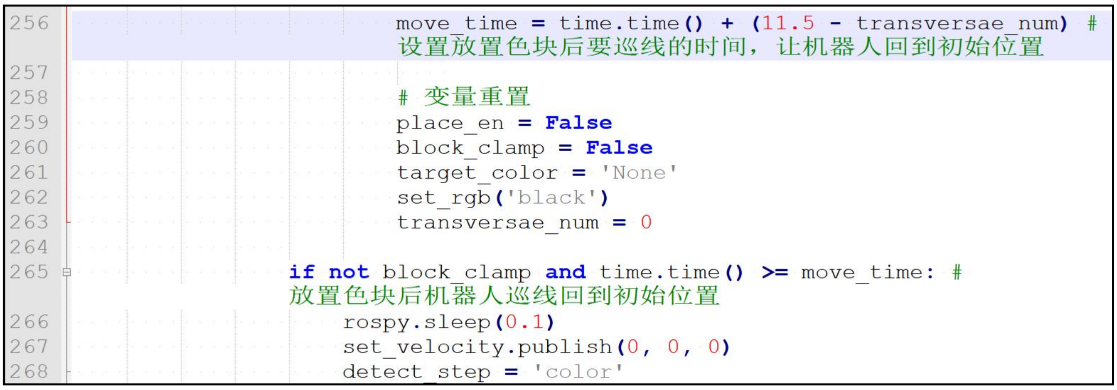
The third and fourth parameter “-180” and “0” is the range of the pitch angle.

Due the limitation of the detected range of camera, when the car has not moved to the corresponding position and the lines is no longer in the detected range. Therefore, it is necessary to add a delay, so that the car can keep moving when the line is not recognized,



```
if transversae_num == 1:
    place_delay = time.time() + 1.1 # 设置延时停下来时间
elif transversae_num == 2:
    place_delay = time.time() + 1.1
elif transversae_num == 3:
    place_delay = time.time() + 1.2
```

Then the car will continue following the line and return to the initial position, ans starting the next round of recognizing and sorting.



```
                        move_time = time.time() + (11.5 - transversae_num) # 设置放置色块后要巡线的时间，让机器人回到初始位置

                        # 变量重置
                        place_en = False
                        block_clamp = False
                        target_color = 'None'
                        set_rgb('black')
                        transversae_num = 0

                if not block_clamp and time.time() >= move_time: # 放置色块后机器人巡线回到初始位置
                    rospy.sleep(0.1)
                    set_velocity.publish(0, 0, 0)
                    detect_step = 'color'
```