Lesson 2 Color Tracking

1. Working Principle

Recognize the color and process it with Lab color space. Firstly, convert RGB color space to LAB and then perform binaryzation, dilation and erosion and other operations to obtain the outline of the target color. Then frame the contour of the color to complete color recognition.

Then process height of robotic arm after recognition. The coordinates (x,y,z) of center point of image take as the set value and the currently obtained coordinates are used as input value to update pid.

Then, calculate on the basis the feedback of image position. Finally, the coordinate value will change linearly through the change of the position, so as to achieve the effect of tracking.

The source code of program is located in:
/home/ubuntu/armpi_pro/src/color_tracking/scripts/color_tracking_node.py

```
def run(msg):
    global x_dis, y_dis, z_dis

    center_x = msg.center_x
    center_y = msg.center_y
    radius = msg.data

    with lock:
        if __isRunning and target_color != 'None':
            if radius > 20:
                # x-axis tracking
                x_pid.SetPoint = img_w / 2.0  # Set
                x_pid.update(center_x)        # Current
                dx = x_pid.output             # Output
                x_dis += int(dx)
                # limiting
                x_dis = 200 if x_dis < 200 else x_dis
                x_dis = 800 if x_dis > 800 else x_dis

                # The y-axis positve direction tracking
                if radius - RADIUS < -3:
                    y1_pid.SetPoint = RADIUS # Set
                    y1_pid.update(radius)    # Current
                    dy = y1_pid.output       # Output

                # The y-axis negative direction tracking
                elif radius - RADIUS > 5:
                    y2_pid.SetPoint = RADIUS # Set
                    y2_pid.update(radius)    # Current
                    dy = y2_pid.output       # Output
                else:
                    dy = 0
                y_dis += dy
                y_dis = 0.12 if y_dis < 0.12 else y_dis
                y_dis = 0.25 if y_dis > 0.25 else y_dis
```

2. Operation Steps

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

2.1 Enter Game

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

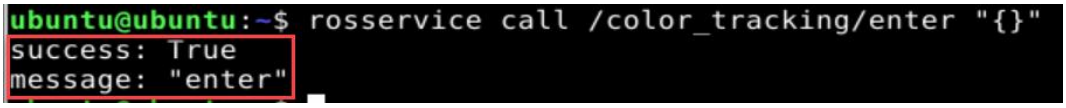
2) Click 

and select 

in pop-up interface to open the terminal.

3) Enter command “rosservice call /color_tracking/enter "{}"” and press “Enter”

to enter this game. After entering, the prompt will be printed, as the figure shown below:



2.2 Start image transmission

2.2.1 Start with browser

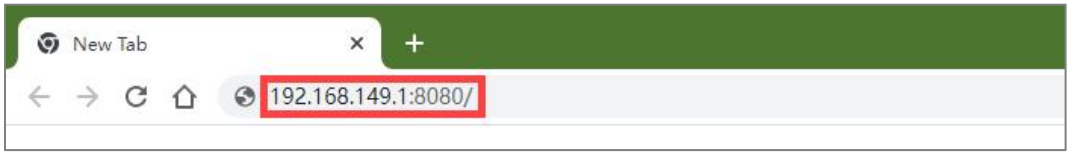
To avoid consuming too much running memory of Raspberry Pi. It is recommended to use an external browser to open the transmitted image.

The specific steps are as follows:

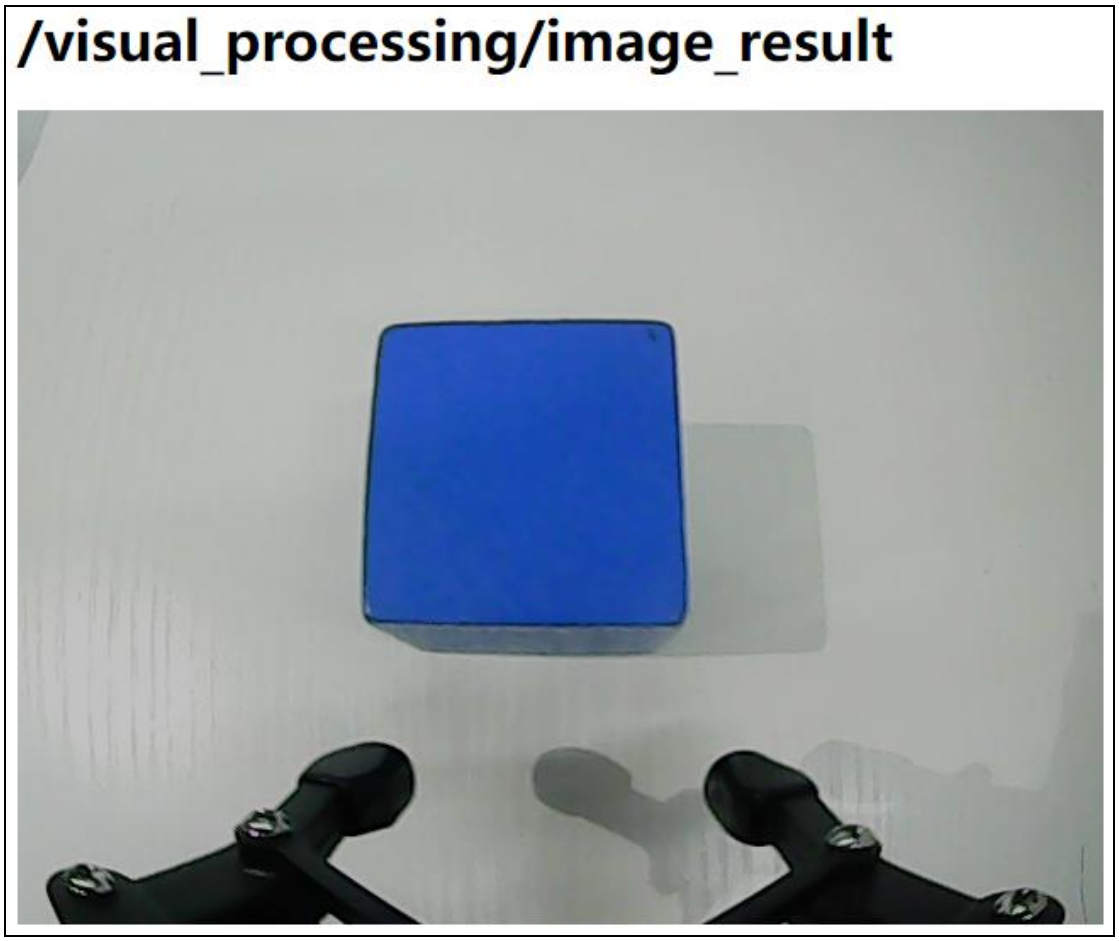
1) Select a browser. Take Google Chrome as example.

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/” for example, “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.



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



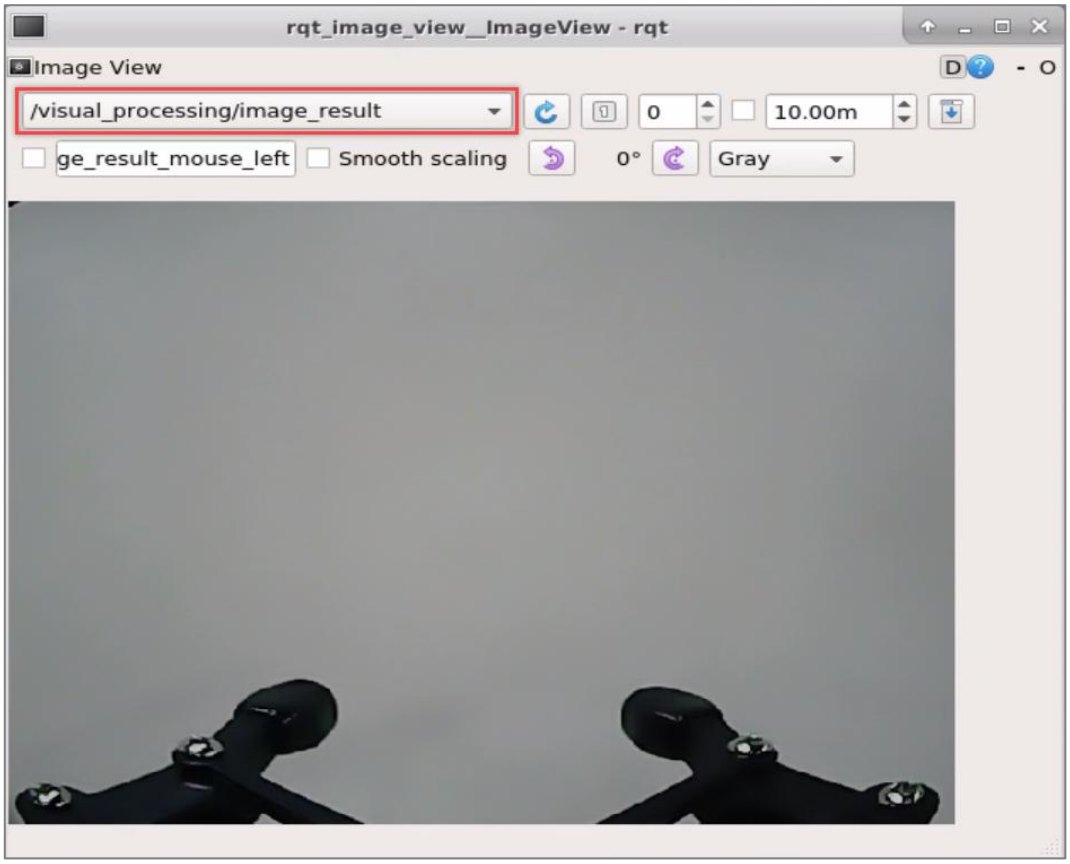
2.2.2 Start with rqt

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.

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

3) Click the red box as the figure shown below, select “/visual_processing/image_result” for the topic of color tracking and remain other settings unchanged.



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

2.3 Start Game

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

4) After starting the game, select the target color. Take blue as example. Enter command “rosservice call /color_tracking/set_target "data: 'blue'"”.

Note: If want to change to green or red, you can fill in green or red in "data: ' ' (The entered command should be case sensitive).

```
ubuntu@ubuntu:~$ rosservice call /color_tracking/set_target "data: 'blue'"
success: True
message: "set_target"
ubuntu@ubuntu:~$
```

2.4 Stop and Exit

1) If want to stop the game, enter command “rosservice call /visual_patrol/set_running "data: false"”. After stopping, you can refer to the content of “2.3 Start Game” to change the tracking color to green or red.

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

2) If want to exit the game, enter command “rosservice call /color_tracking/exit "{}"” to exit.

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

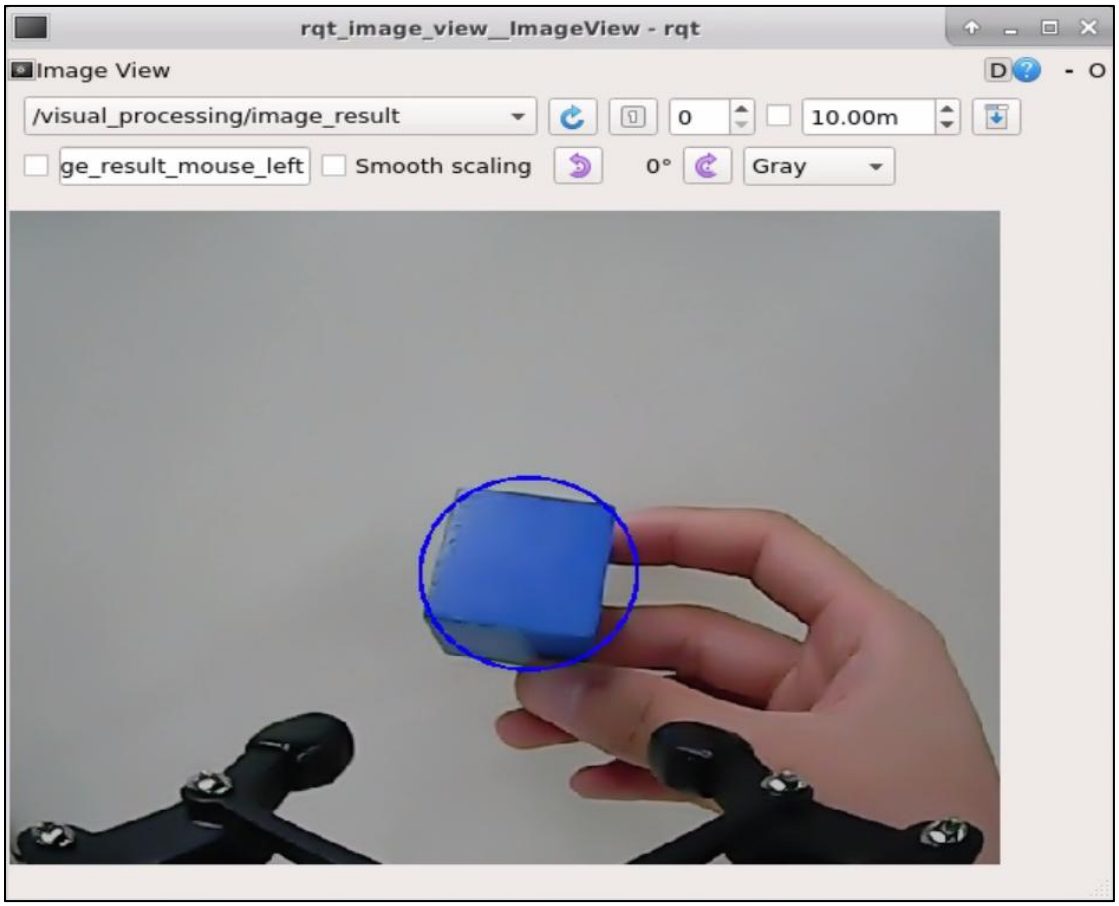
Note: Before exiting the game, it will keep running when Raspberry Pi is powered on. To avoid consume too much running memory of Raspberry Pi, you need to exit the game first according to the operation steps above before performing other AI vision games.

---

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

3. Project Outcome

After starting game, place the blue block within the detected range of camera. The target color will be framed in rqt tool after recognition. At this time, move the block slowly. Then the robotic arm will move with the target color.



4. Function Extension

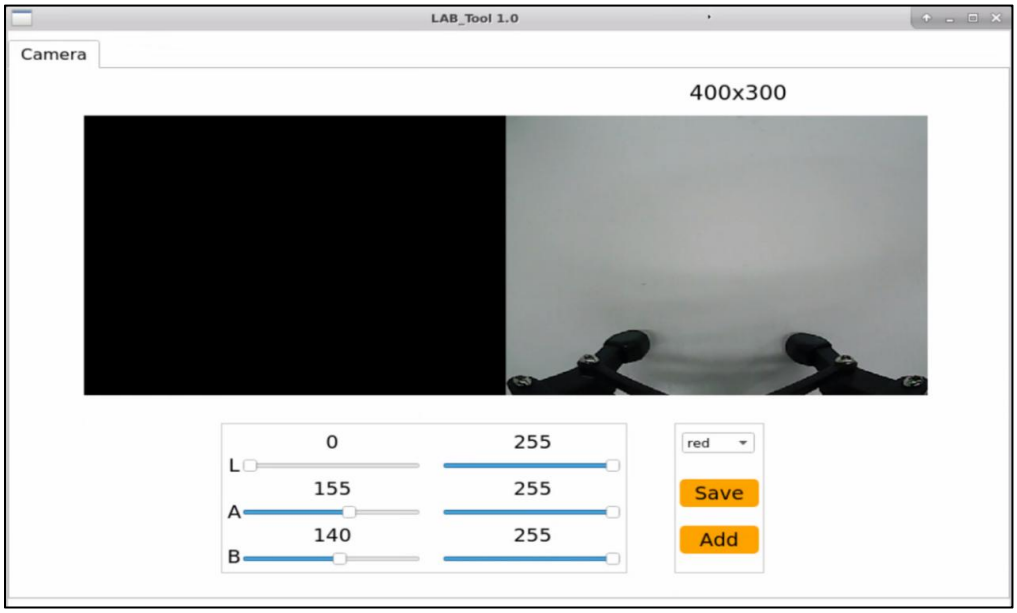
4.1 Add New Recognition Color

Color tracking has three built-in color red, green and blue. In addition to the built-in colors, we can add other recognition colors. For example, add pink as a new recognizable color. The operation steps are as follow:

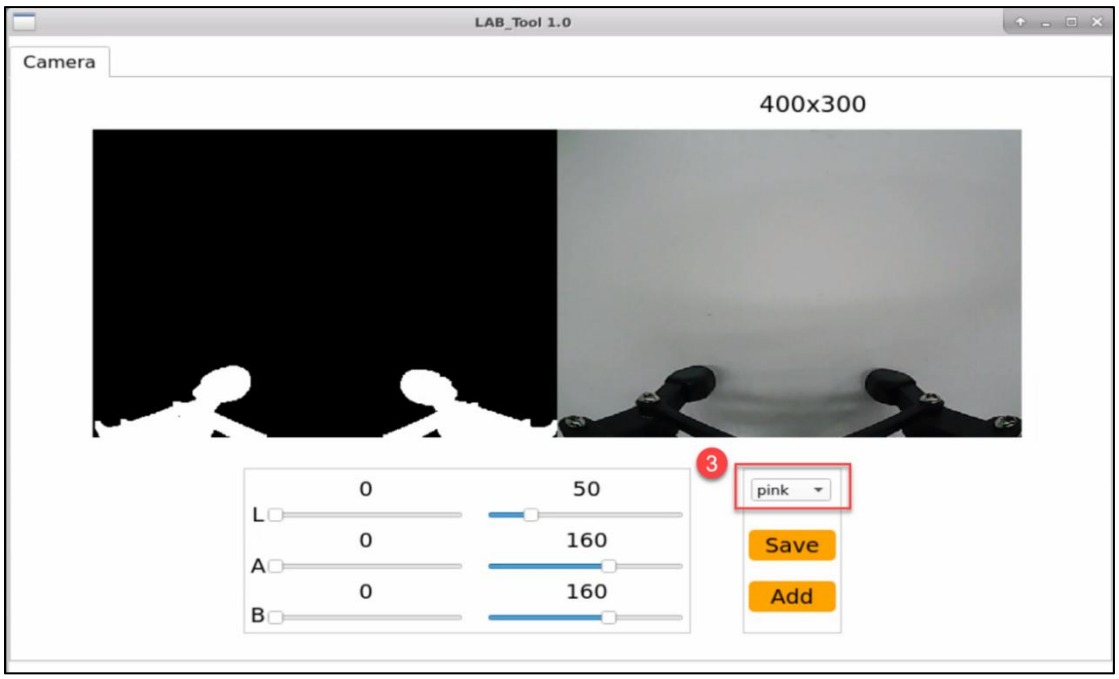
1) Open the terminal, enter command “python3 lab_config/main.py” and press “Enter” to open the tool for color threshold adjustment. If no transmitted image appears in the pop-up interface, it means the camera fails to connect and needs to be checked whether the wire is connected.

```
ubuntu@ubuntu:~$ python3 lab_config/main.py
libEGL warning: DRI2: failed to authenticate
```

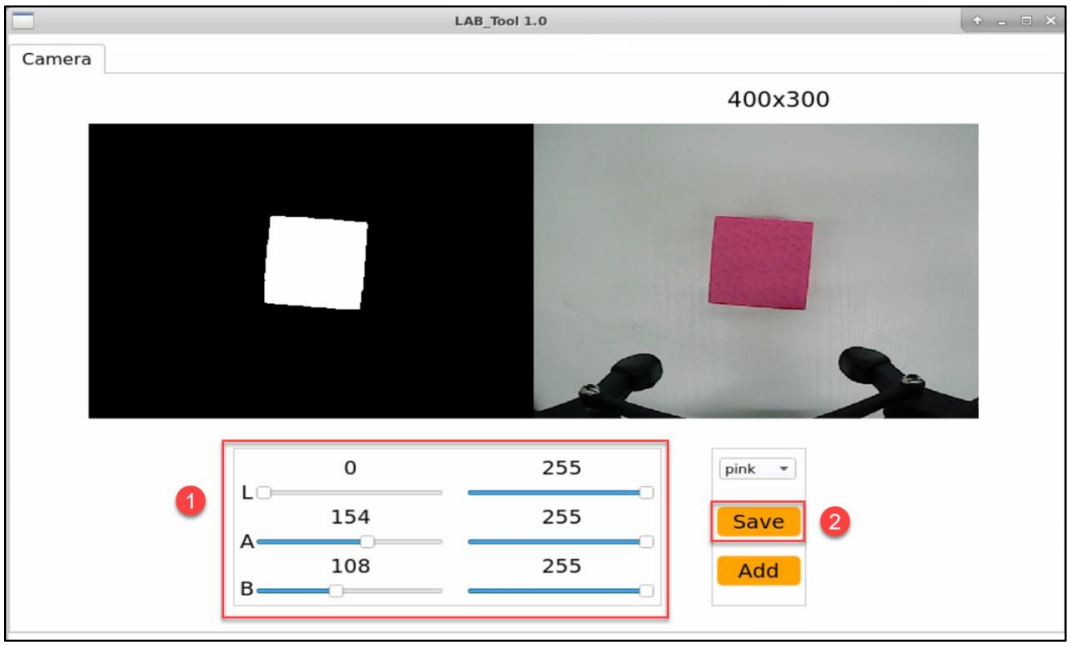
2) After the camera is connected completely, we can see that the right side is real-time transmitted image and the right side is the color to be collected. Then click “Add” in the lower right color to name the new color.



3) Fill in the name of added color and click “Ok”. The color will be updated to “pink” in the color options bar in the lower right corner.

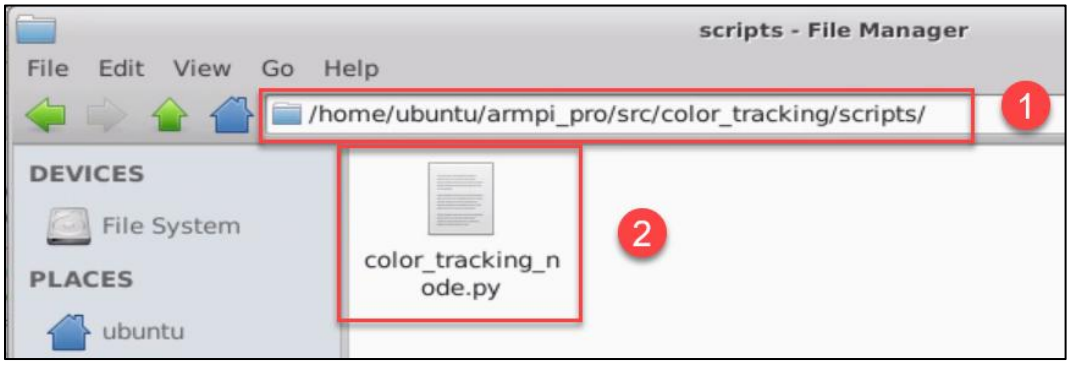


4) Point the camera at the pink object. Then drag the following six slider bars until the pink area becomes white and other areas become black and click “Save” to save data.

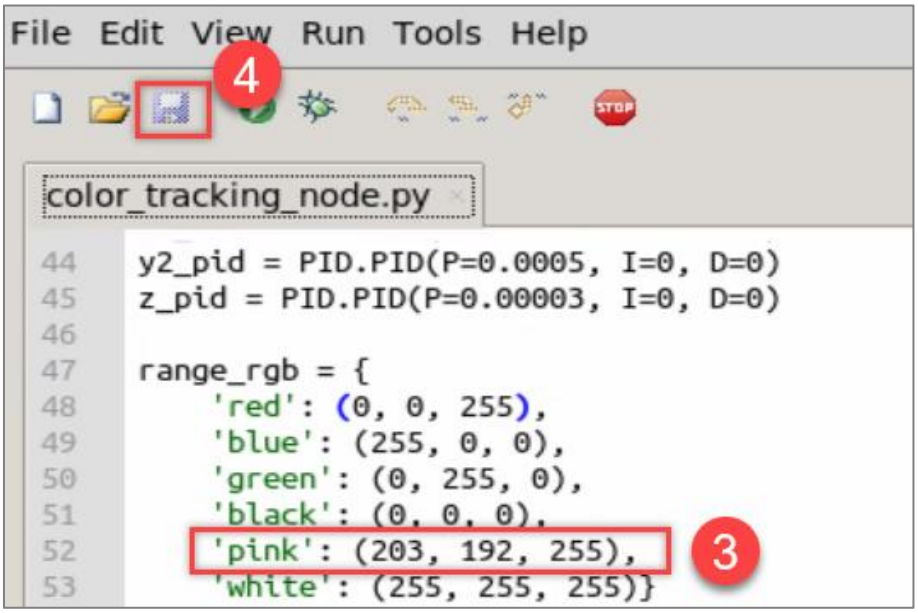


5) Find the source code of color tracking

/home/ubuntu/armpi_pro/src/color_tracking/scripts/ and double click to open.



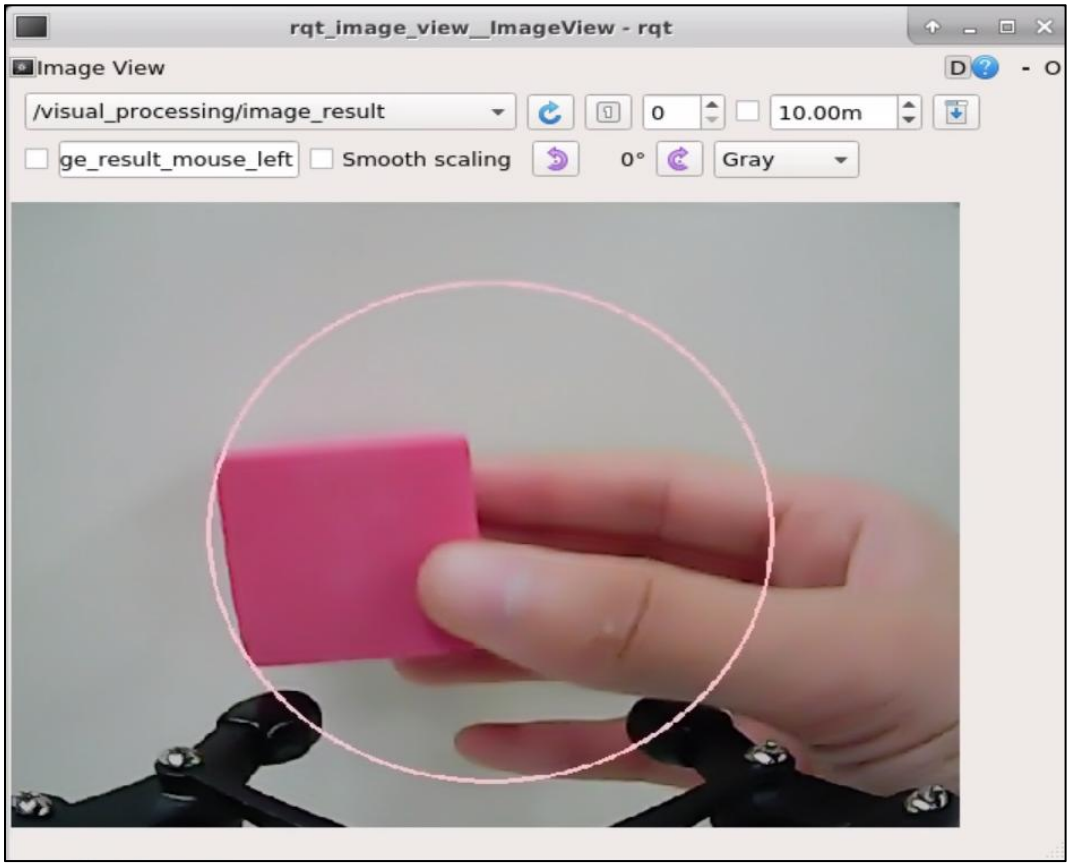
6) Then input the RGB value of pink "**'pink': (203, 192, 255),**" in source code and save it.



7) Open the terminal and enter command "sudo systemctl restart start_node.service " to restart the game. (Wait for 1 minute to hear "Di" sound, then the game is restarted successfully)

8) Refer to the operation steps from 2.1 Enter game to 2.3 Start game to start color tracking.

9) Put pink object in front of the camera then slowly move the object. Arm Pi Pro will move with the pink object.



10) If want to add other colors as new recognizable color, you can refer to the operation steps of “4.1 Add New Color”.

5. Program Parameter Instruction

5.1 Image Process

The source code of image process program is located in:

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

```
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)  # convert image into LAB space

    for color in color_list:
        if color in color_range_list:
            color_range = color_range_list[color]
            frame_mask = cv2.inRange(frame_lab, tuple(color_range['min']), tuple(color_range['max']))  # Bitwise operation
            operates on the original image and mask.
            eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2)))        # erode
            dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2)))          # dilate
            contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]       # find contour
            areaMaxContour, area_max = getAreaMaxContour(contours)                                   # find the biggest contour
```

5.1.1 Binarization

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

```
246         frame_mask = cv2.inRange(frame_lab, tuple(color_range['min']), tuple(color_range['max']))
            # Bitwise operation operates on the original image and mask.
```

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.

5.1.2 Dilation and Erosion

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

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

erode() function is applied to erode image. Take code “**eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2)))**” 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.

5.1.3 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.

```
249     contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]
        # find contour
```

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.

```
250     area_max_contour, area_max = getAreaMaxContour(contours)    # find the biggest contour
251
252     if area_max > 200:  # find the biggest area
```

5.1.4 Obtain Position Information

The minAreaRect() function in cv2 library is used to obtain the minimum exterior 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(area_max_contour)  # Get the smallest circumcircle
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))
msg.data = int(Misc.map(radius, 0, size_m[0], 0, img_w))
cv2.circle(img, (msg.center_x, msg.center_y), msg.data+5, range_rgb[color], 2)
publish_en = True
```

5.2 Control Tracking

Take the X and Y coordinates of the center point of image as the set values, and tale the X and Y coordinates of the currently recognized target as the input values for inverse kinematic calculation to obtain the target position.

```
# z-axis tracking
z_pid.SetPoint = img_h / 2.0  # Set
z_pid.update(center_y)        # Current
dy = z_pid.output             # Output
z_dis += dy

z_dis = 0.22 if z_dis > 0.22 else z_dis
z_dis = 0.17 if z_dis < 0.17 else z_dis

target = ik.setPitchRanges((0, round(y_dis, 4), round(z_dis, 4)), -90, -85, -95) # 逆运动学求解
if target:
    # Publiash servo control node meassage to control the movement of robotic arm
    servo_data = target[1]
    bus_servo_control.set_servos(joints_pub, 20, (
        (3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5']), (6, x_dis)))
```

A the code of inverse kinematic calculation “**ik.setPitchRanges((0, round(y_dis, 4), round(z_dis, 4)), -90, -85, -95)**” as example. The meaning of parameters in parentheses are as follow:

The first parameter “0” is the value of x-axis, “**round(y_dis, 4)**” is the value of y-axis and “**round(z_dis, 4)**” is the value of z-axis.

The second parameter “-90” is the pitch angle.

The third parameter “-85” is the range of the pitch angle.

The fourth parameter “-95” is

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.The meaning of parameters in parentheses are 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']” is the servo angle and the remaining parameters are the same.