Lesson 2 Target 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 takes 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/object_tracking/scripts/object_tracking_node.py

```
def reset():
    global target_color
    global arm_x, arm_y

    with lock:
        arm_x = Arm_X
        arm_y = Arm_Y
        x_pid.clear()
        y_pid.clear()
        arm_x_pid.clear()
        arm_y_pid.clear()
        off_rgb()
        target_color = 'None'

# app initialization call
def init():
    rospy.loginfo("object tracking Init")
    initMove()
    reset()

# image process result callback function
def run(msg):
    global lock
    global move
    global arm_x, arm_y

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

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 Applications and select Terminal Emulator in pop-up interface to open the terminal.

3) Enter command "**rosrun object_tracking object_tracking_node.py**" and press "Enter" to run the program of target tracking.

```
ubuntu@ubuntu:~$ rosrun object_tracking object_tracking_node.py
[DEBUG] [1656589583.407065]: init_node, name[/object_tracking], pid[22602]
[DEBUG] [1656589583.412227]: binding to 0.0.0.0 0
[DEBUG] [1656589583.417123]: bound to 0.0.0.0 33069
[DEBUG] [1656589583.422855]: ... service URL is rosrpc://ubuntu:33069
[DEBUG] [1656589583.427693]: [/object_tracking/get_loggers]: new Service instanc
e
```

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

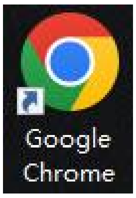
```
ubuntu@ubuntu:~$ rosservice call /object_tracking/enter "{}"
success: True
message: "enter"
```

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.

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.

/visual_processing/image_result

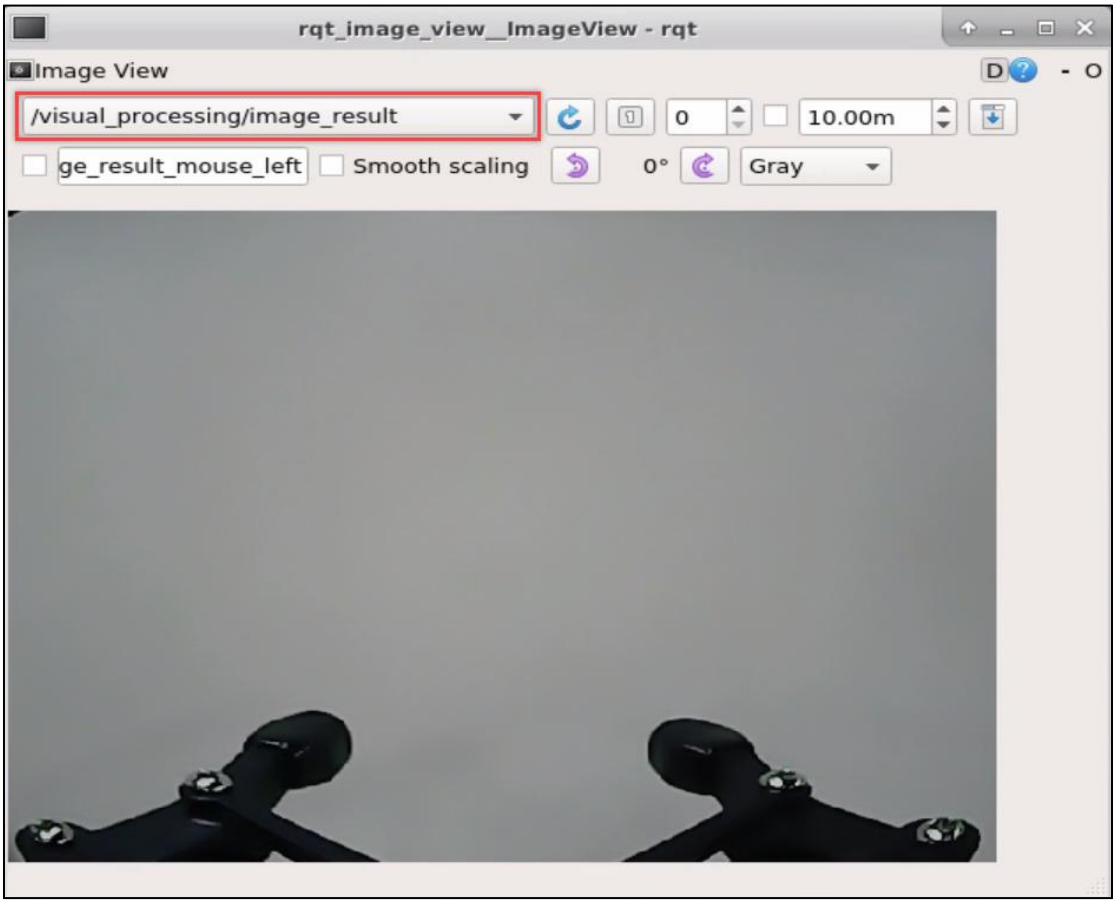
2.2.2 Start with rqt

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 line following 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

Now, enter the terminal according to the steps in “2.1 Enter Game” and input command “**rosservice call /object_tracking/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 /object_tracking/set_running "data: true"
success: True
message: "set_running"
ubuntu@ubuntu:~$
```

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 /object_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 /object_tracking/set_running "data: false"**". After stopping, you can refer to the content of "2.3 Start Game" to change other tracking colors.

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

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

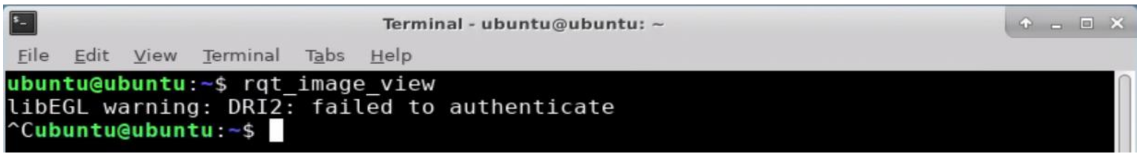
```
ubuntu@ubuntu:~$ rosservice call /object_tracking/exit "{}"
success: True
message: "exit"
```

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.

```
[INFO] [1656053310.803753]: data: "blue"
[DEBUG] [1656053310.917365]: connecting to ubuntu 38919
[INFO] [1656053337.062093]: stop running object tracking
[DEBUG] [1656053337.102660]: connecting to ubuntu 38919
[INFO] [1656053352.044230]: exit object tracking
[DEBUG] [1656053352.056126]: connecting to ubuntu 38919
[INFO] [1656053352.145915]: 'NoneType' object has no attribute 'cancel'
^C[DEBUG] [1656053365.322010]: connecting to ubuntu 38919
[DEBUG] [1656053365.340156]: TCPServer[37005] shutting down
[INFO] [1656053365.877228]: shutdown
```

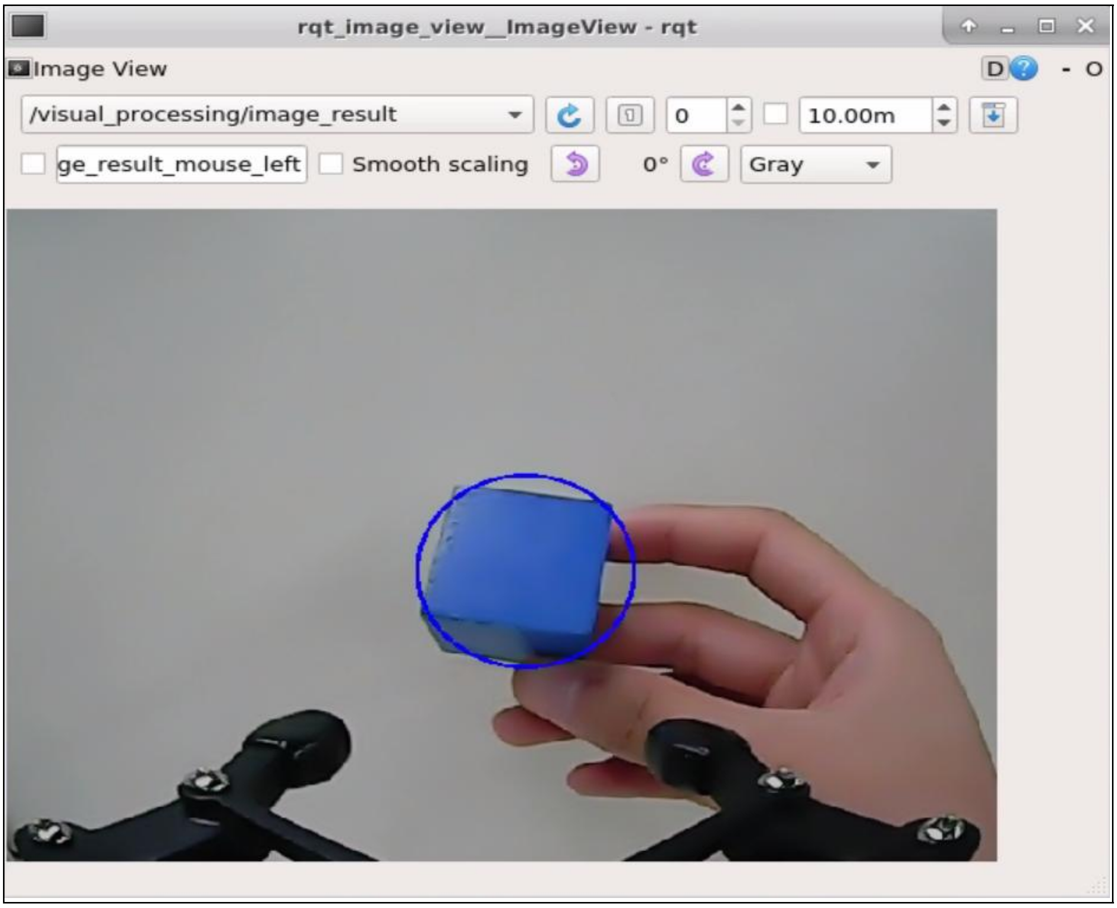
Note: Before exiting the game, it will keep running when Raspberry Pi is powered on. To avoid consume too much running memory of Raspberry Pi, you need to exit the game first according to the operation steps above before performing other AI vision games.

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.



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 rotate to the direction of the block and the car will move to the block.

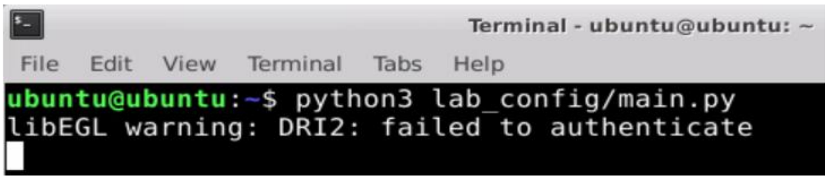


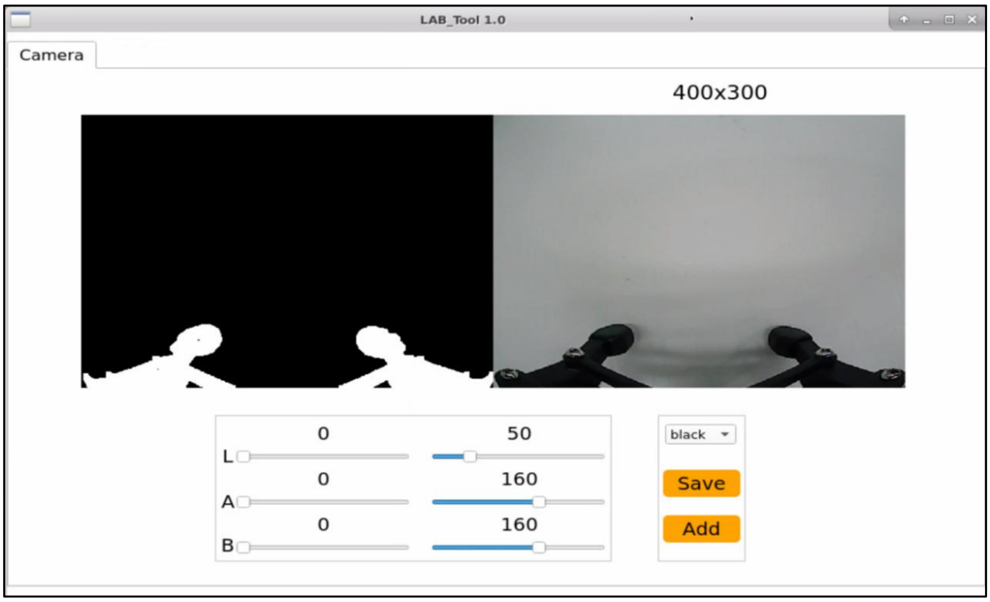
4. Function Extension

4.1 Add New Recognition Color

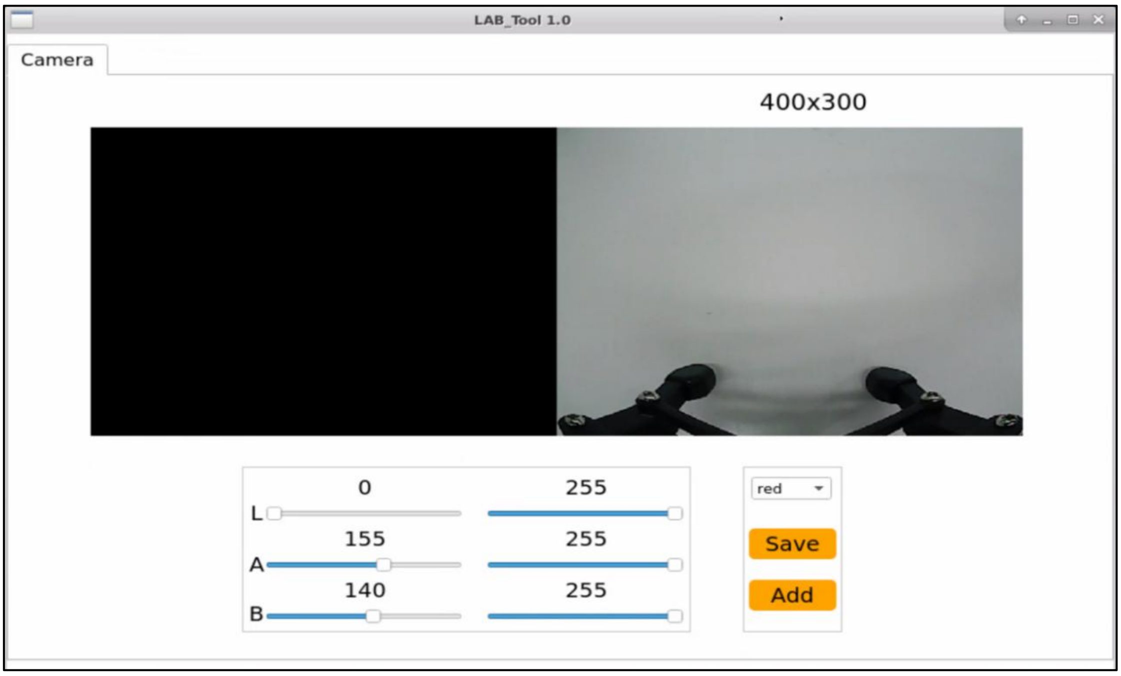
Target tracking has three built-in color red, green and blue. In addition to the built-in colors, we can add other recognition colors. For example, add pink as a new recognizable color. The operation steps are as follow:

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.

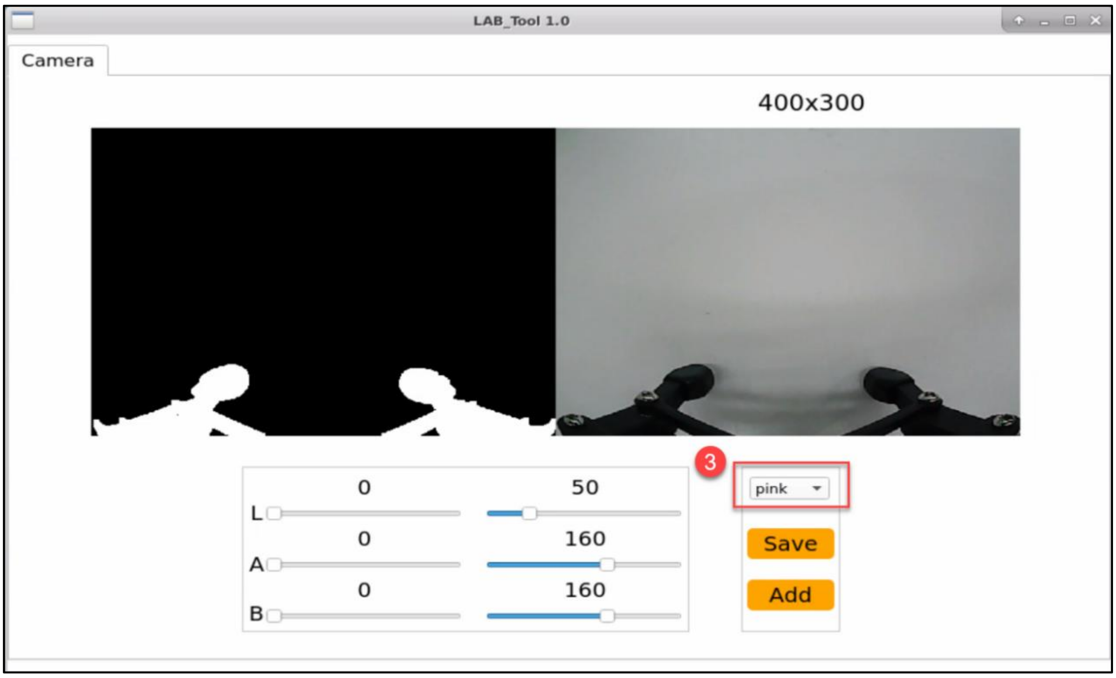




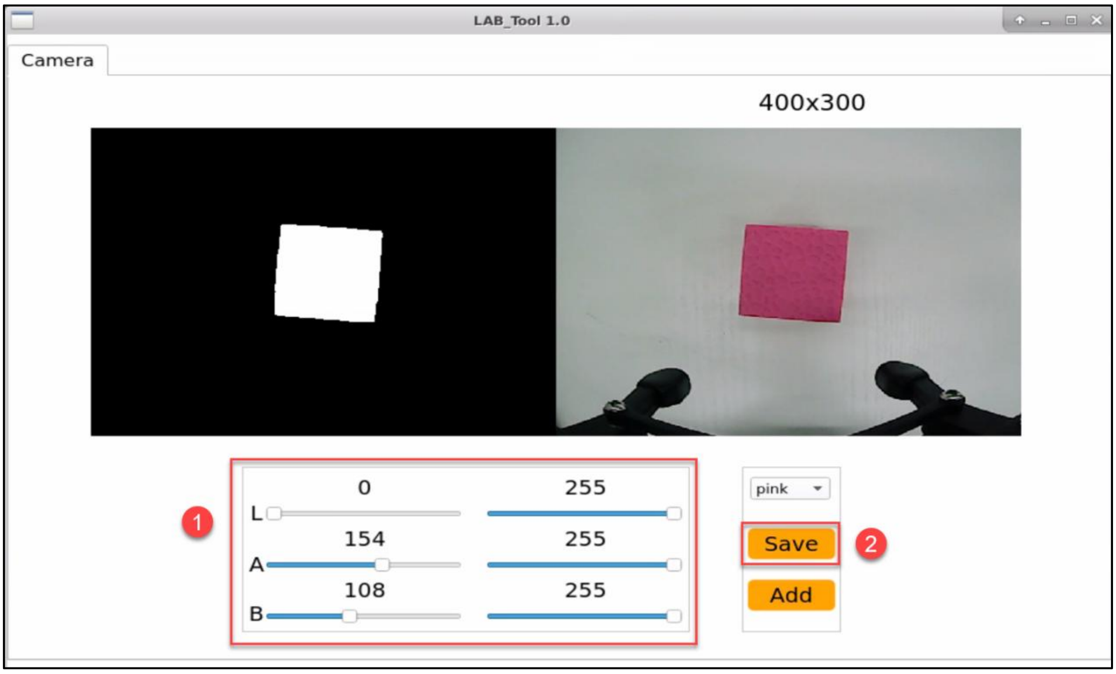
2) After the camera is connected completely, we can see that the right side is real-time transmitted image and the right side is the color to be collected. Then click “Add” in the lower right color to name the new color.



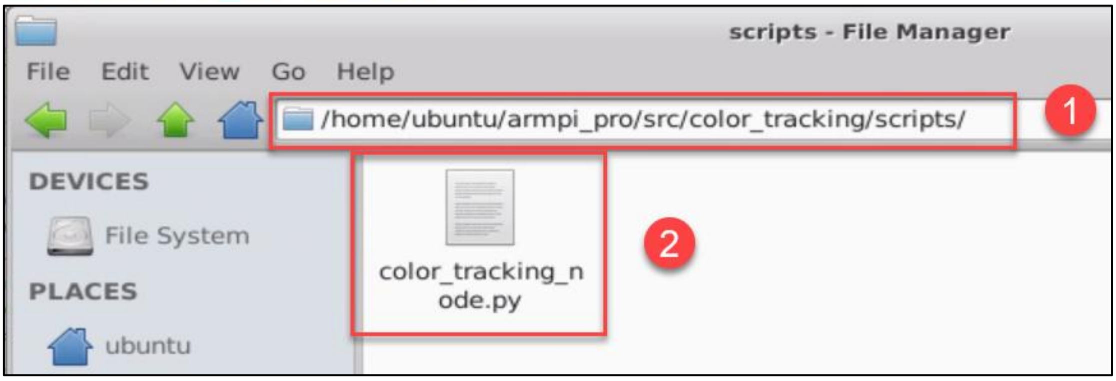
3) Fill in the name of added color and click “Ok”. The color will be updated to “pink” in the color options bar in the lower right corner.

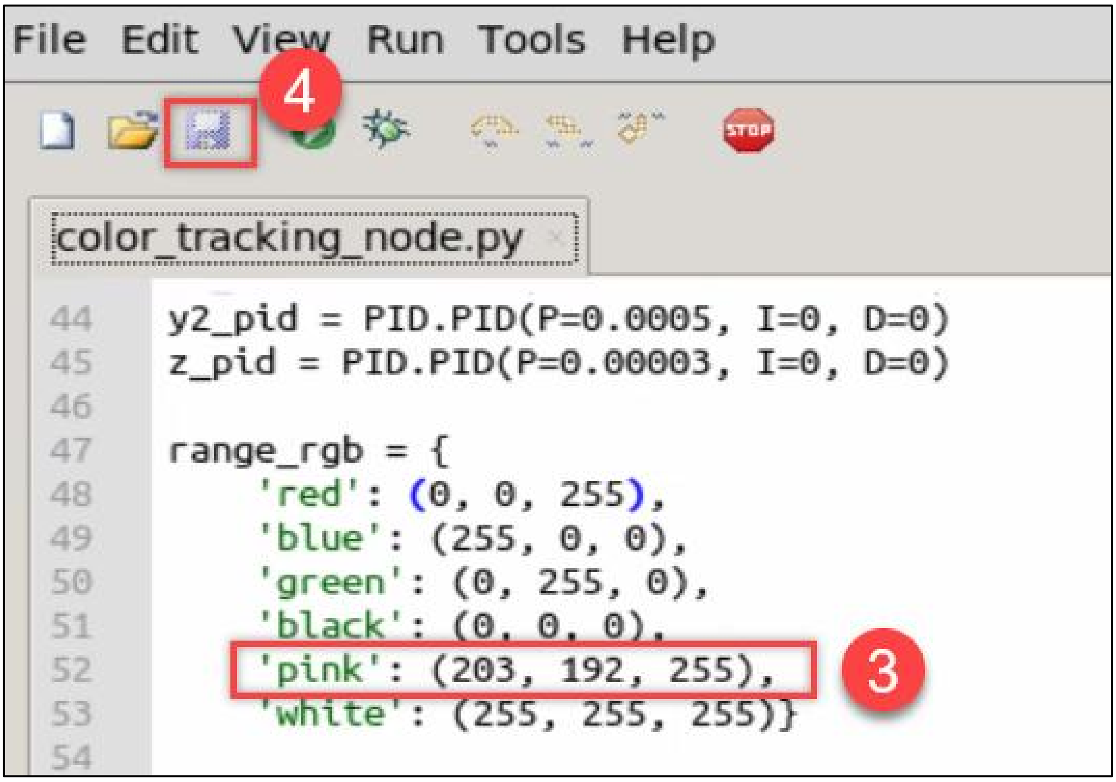


4) Point the camera at the pink object. Then drag the following six slider bars until the pink area becomes white and other areas become black and click "Save" to save data.



5) Find the source code of target tracking and double click to open. Then input the RGB value of pink in source code and save it.

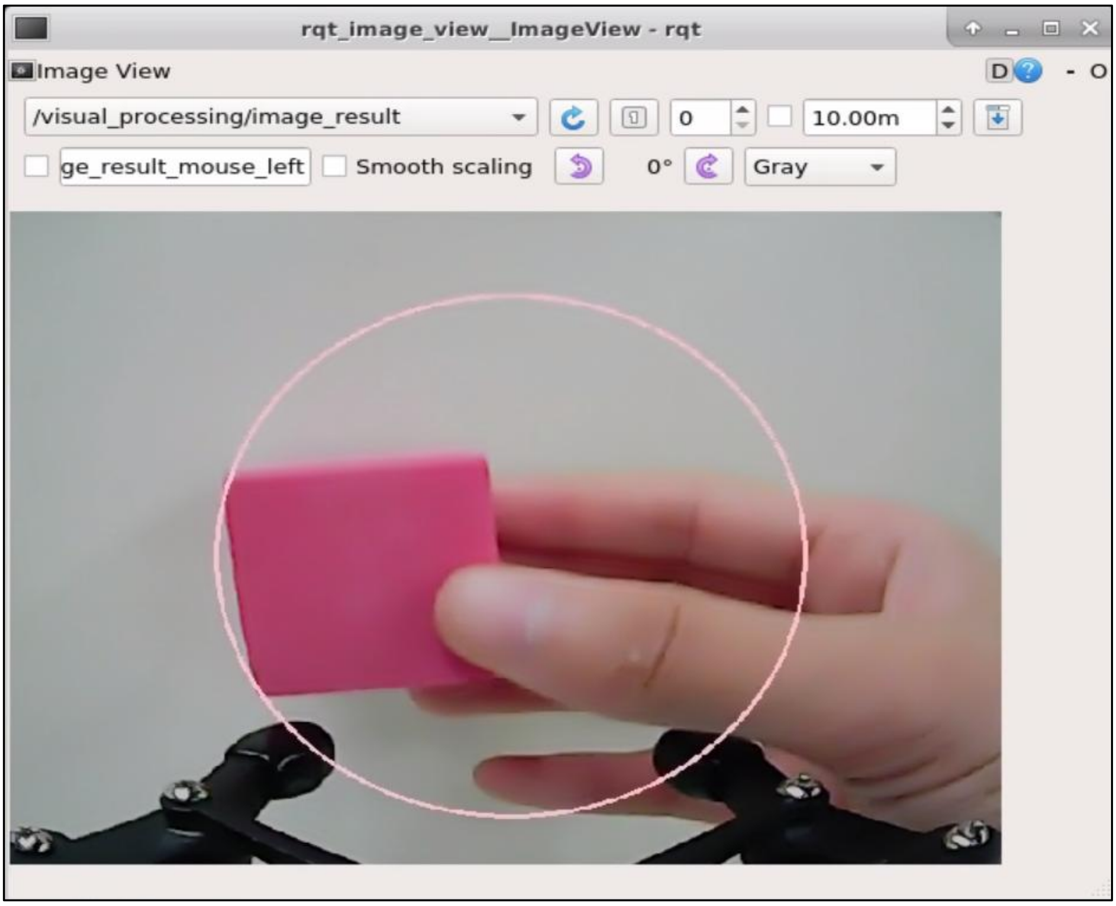




6) Open the terminal and enter command “sudo systemctl restart start_node.service ” to restart the game. (Wait for 1 minute to hear “Di” sound, then the game is restarted successfully)

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

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



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

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**

```python
# single oolor detection
def color_detect(img, color):
    global pub_time
    global publish_en
    global color_range_list

    if color == 'None':
        return img

    msg = Result()
    area_max = 0
    area_max_contour = 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

    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
        area_max_contour, area_max = getAreaMaxContour(contours)  # find the biggest contour

        if area_max > 200:  # find the biggest area
            (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

        if publish_en:
            if (time.time()-pub_time) >= 0.06:
                result_pub.publish(msg)  # publish result
                pub_time = time.time()
```

4.1.1 Binarization

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

```python
        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.

4.1.2 Dilation and Erosion

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

```python
        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
```

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, (3, 3))**" 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]
# 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.

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".

```
area_max_contour, area_max = getAreaMaxContour(contours)    # find the biggest contour

if area_max > 200:  # find the biggest area
```

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(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
```

4.2 Control Action

Control the servos by calling bus_servo_control.set_servos() function.

```
# robotic arm x-axis tracking
if abs(center_x - img_w/2.0) < 15:
    center_x = img_w/2.0
arm_x_pid.SetPoint = img_w/2.0  # set
arm_x_pid.update(center_x)      # current
arm_x += arm_x_pid.output       # output
arm_x = 200 if arm_x < 200 else arm_x
arm_x = 800 if arm_x > 800 else arm_x

# robotic arm y-axis tracking
if abs(center_y - img_h/2.0) < 15:
    center_y = img_h/2.0
arm_y_pid.SetPoint = img_h/2.0  # set
arm_y_pid.update(center_y)      # current
arm_y += arm_y_pid.output       # output
arm_y = 50 if arm_y < 50 else arm_y
arm_y = 300 if arm_y > 300 else arm_y

# robotic arm movement
bus_servo_control.set_servos(joints_pub, 20, ((3, arm_y), (6, arm_x)))
```

Servo control takes “bus_servo_control.set_servos(joints_pub, 20, ((3, arm_y), (6, arm_x)))” as example and the meaning of parameters in parentheses are as follow:

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

The second parameter “20” is the running time.

The third parameter is “((3, arm_y), (6, arm_x)”. “3” is the servo number, “arm_y']” is the servo angle.

```
            # chassis car x-axis tracking
            if abs(arm_x - Arm_X) < 5:
                arm_x = Arm_X
            x_pid.SetPoint = Arm_X       # set
            x_pid.update(arm_x)          # current
            dx = x_pid.output    # output
            dx = -200 if dx < -200 else dx
            dx = 200 if dx > 200 else dx

            # chassis car y-axis tracking
            if abs(arm_y - Arm_Y) < 5:
                arm_y = Arm_Y
            y_pid.SetPoint = Arm_Y   # set
            y_pid.update(arm_y)      # current
            dy = -y_pid.output    # output
            dy = -180 if dy < -180 else dy
            dy = 180 if dy > 180 else dy

            # chassis car movement
            set_translation.publish(dx,dy)
            move = True
```

Motor control takes the code “set_translation.publish(dx,dy)” as example and the meaning of parameters in parentheses are as follow:

The first parameter “dx” is the movement distance of car in x-axis.

The second parameter “dy” is the movement distance of car in y-axis.