Lesson 1 Line Following

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 line and its center point.

Finally, after identifying red line, ArmPi Pro will follow the line.

The source code of program is located in: **/home/ubuntu/armpi_pro/src/visual_patrol/scripts/visual_patrol_node.py**

```
def exit_func(msg):
    global lock
    global result_sub
    global __isRunning
    global result_sub_st
    
    rospy.loginfo("exit visual patrol")
    with lock:
        rospy.ServiceProxy('/visual_processing/exit', Trigger)()
        __isRunning = False
        reset()
        set_velocity.publish(0, 90, 0)
        try:
            if result_sub_st:
                result_sub_st = False
                heartbeat_timer.cancel()
                if result_sub is not None:
                    result_sub.unregister()
        except BaseException as e:
            rospy.loginfo('%s', e)
            
    return [True, 'exit']
```

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 /visual_patrol/enter "{}"” and press “Enter” to enter this game. After entering, the prompt will be printed, as the figure shown below:

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

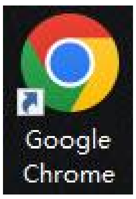
2.2 Start image transmission

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 start image transmission.

The specific steps are as follows:

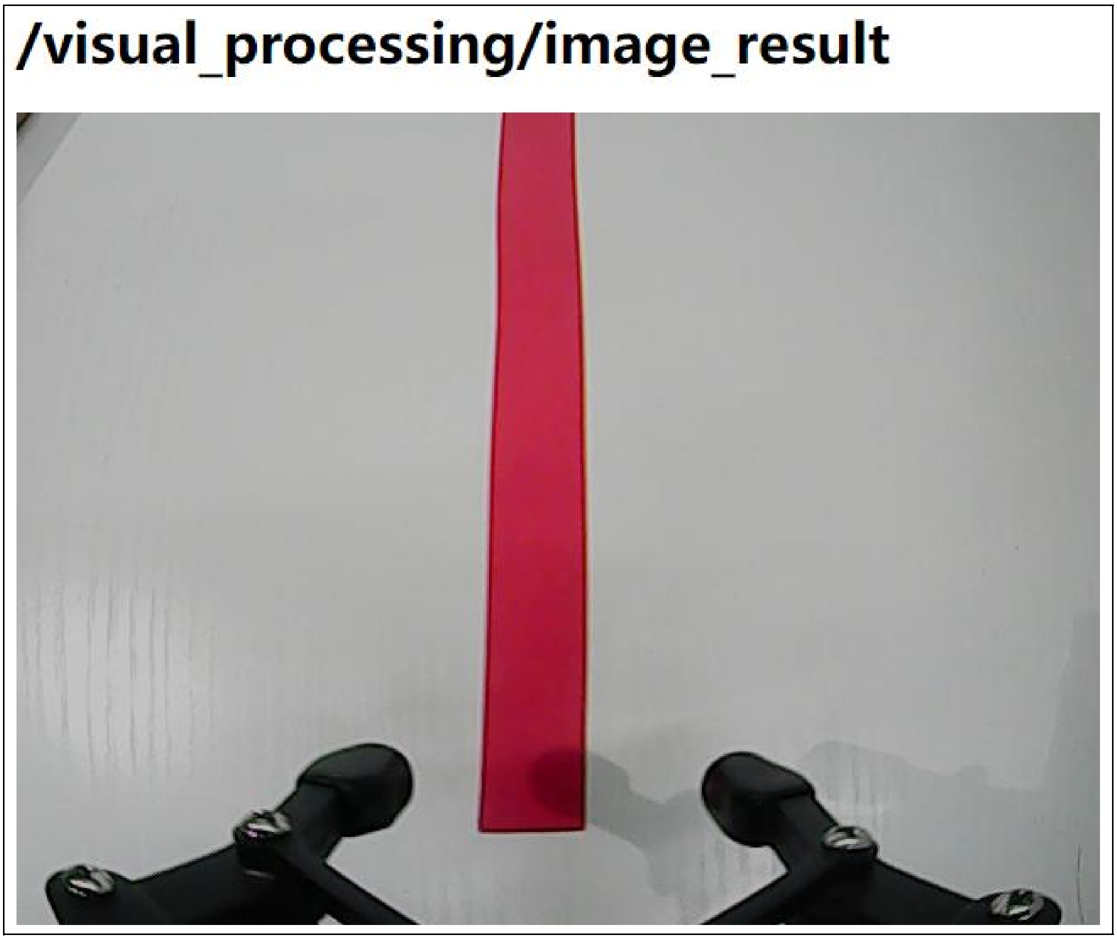
1) Select a browser. Take Google Chrome as example.



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/”, 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

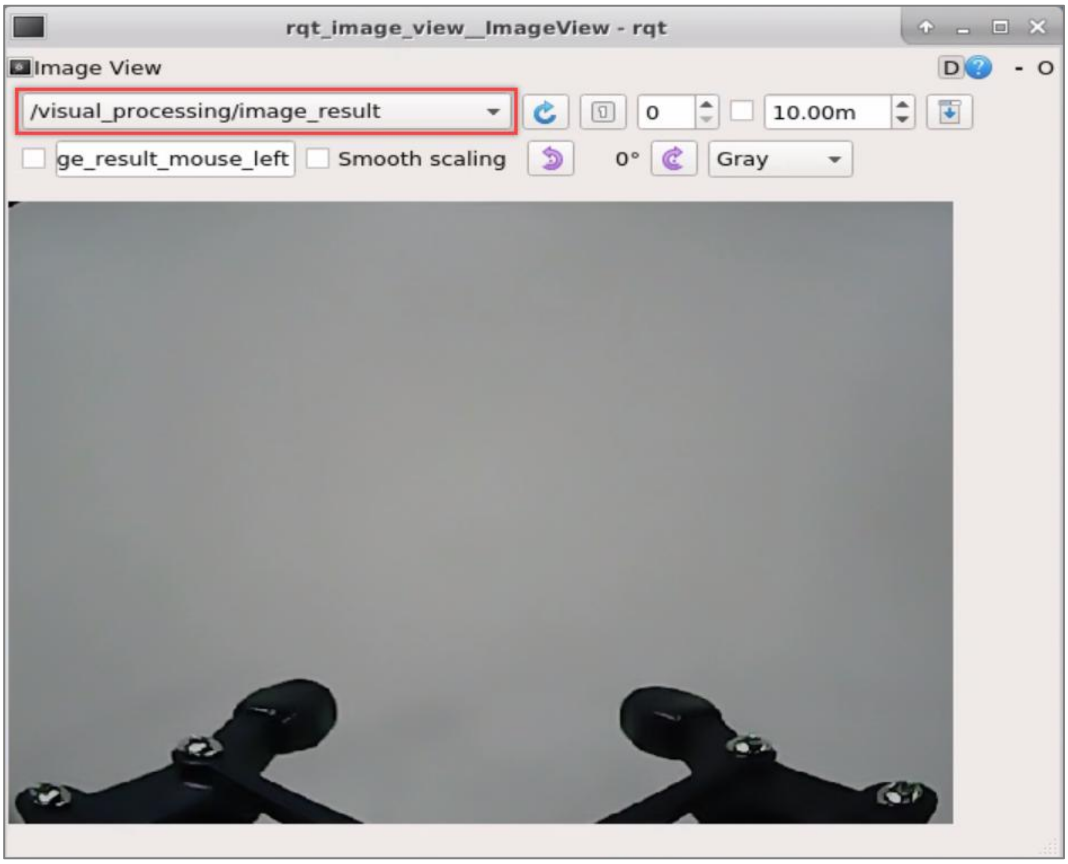
1) After completing the steps of "2.1 Enter Game" and do not exit the terminal,

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 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 /visual_patrol/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 /visual_patrol/set_running "data: true"
success: True
message: "set_running"
```

1) After starting the game, select the line color. Take following red line as example. Enter command “**rosservice call /visual_patrol/set_target "data: 'red'"**”.

Note: If want to change the target line from red to green or blue. You can replace red in "data: 'red '" with green or blue. (The entered command should be case sensitive)

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

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 line color and start following again.

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

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

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

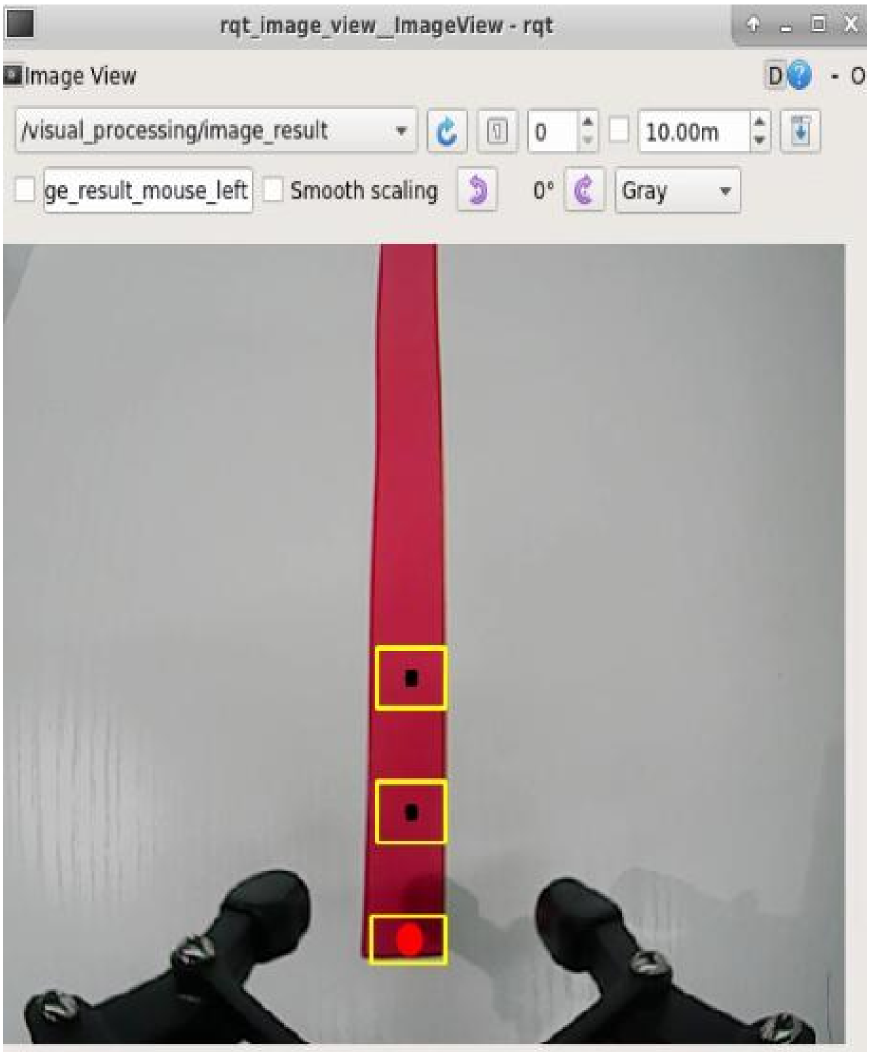
Note: Before exiting the game, it will keep running when Raspberry Pi is powered on. To avoid consume too much running memory of Raspberry Pi, you need to exit the game first according to the operation steps above before performing other AI vision games.

---

3) If want to 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

Stick the tape on a flat surface and put ArmPi Pro on the red line. After starting game, robot will follow the red line.

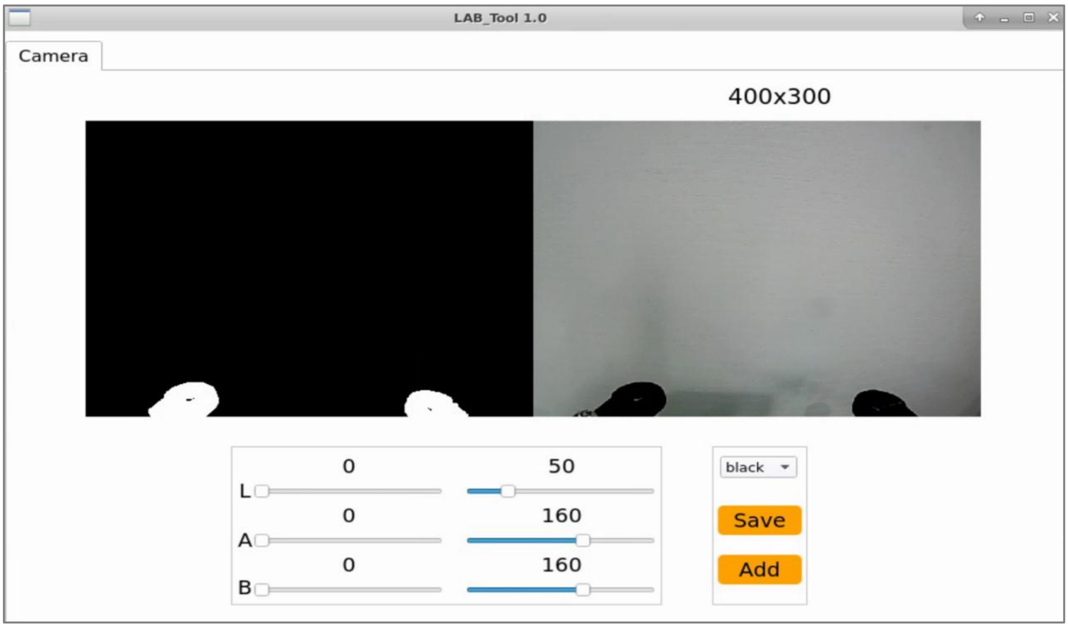


4. Function Extension

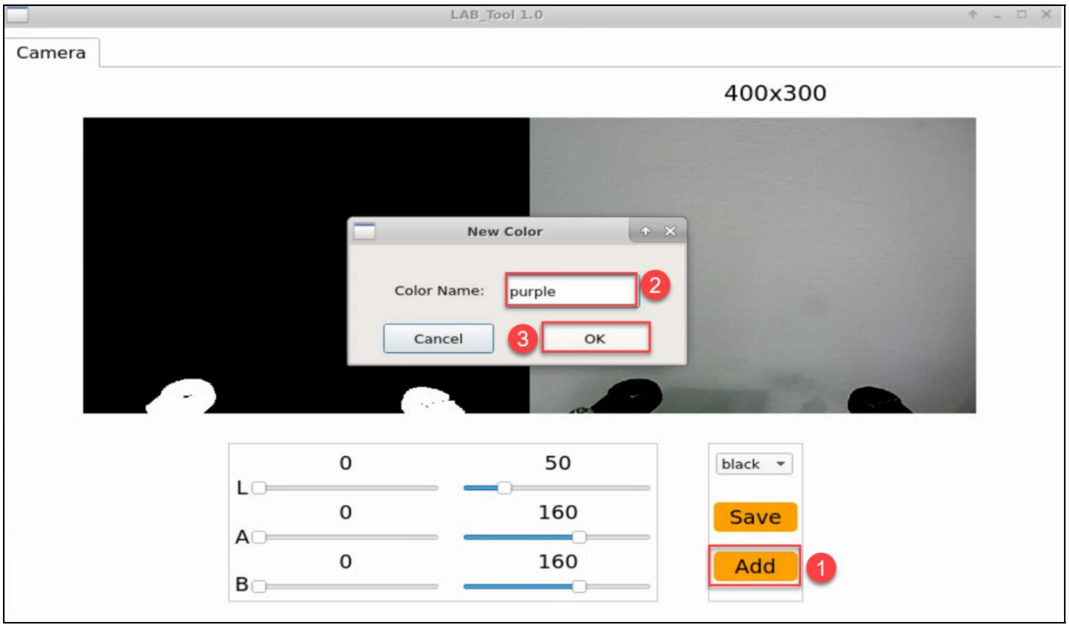
4.1 Add New Recognition Color

In addition to the built-in colors (red and white), we can add other colors for line following. For example, add purple as a new recognition color. The operation step are as follow:

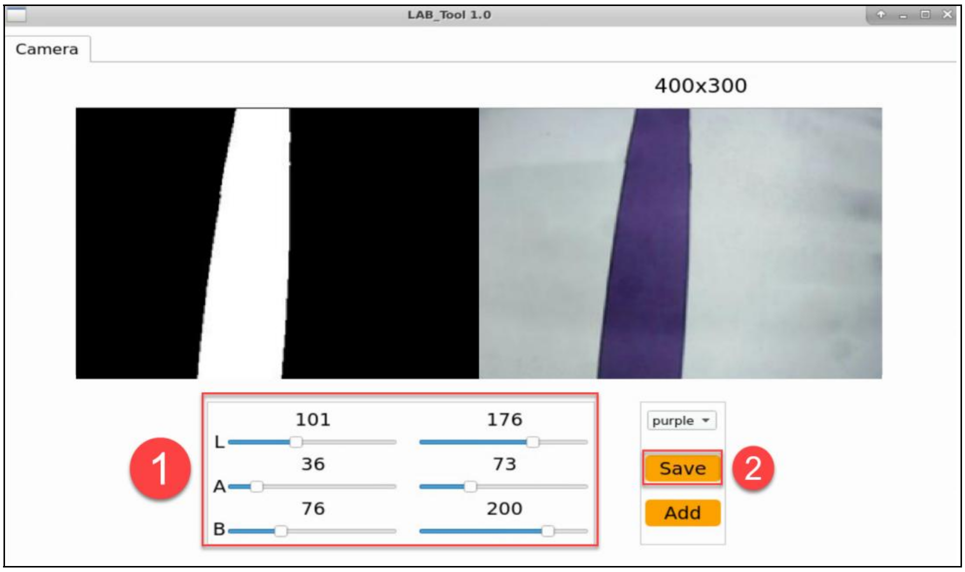
1) Open the terminal, enter command “python3 lab_config/main.py” and press “Enter” to open the tool for color threshold adjustment. If no transmitted image appears in the pop-up interface, it means the camera fails to connect and needs to be checked whether the cable is connected.



2) After the camera is connected completely, click “Add” in the lower right color to name the new color.



3) The right side is real-time transmitted image and the right side is the color to be collected. Point the camera at the purple tape and then drag the following six slider bars until the purple area becomes white and other areas become black. The threshold can be adjusted according the actual situation. Then click “Save” to save data.



4) According to the content of “2.Operation Steps” to start line following

Put ArmPi Pro in front of the purple line and then it will follow the purple line. If want to add other colors as new recognizable color, you can refer to the operation steps of “4.1 Add New Recognition 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 line_detect(img, color):
    global pub_time
    global publish_en
    global color_range_list

    if color == 'None':
        return img

    n = 0
    line_width = 0
    msg = Result()
    area_max = 0
    weight_sum = 0
    centroid_x_sum = 0
    area_max_contour = 0
    img_copy = img.copy()
    img_h, img_w = img.shape[:2]
    frame_resize = cv2.resize(img_copy, size_s, interpolation=cv2.INTER_NEAREST)
    #frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
    #Divide the image into upper, middle and lower parts, so the processing speed will be faster and more accurate
    for r in roi:
        roi_h = roi_h_list[n]
        n += 1
        blobs = frame_resize[r[0]:r[1], r[2]:r[3]]
        frame_lab = cv2.cvtColor(blobs, 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
```

5.1.1 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']))
            # 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.

```
            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.

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.

```
        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 > 50:  # there is the biggest contour
```

5.1.4 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.

```
rect = cv2.minAreaRect(area_max_contour)  # The minimum enclosing rectangle
box = np.int0(cv2.boxPoints(rect))       # The four vertices of the smallest enclosing rectangle
for i in range(4):
    box[i, 1] = box[i, 1] + (n - 1)*roi_h + roi[0][0]
    box[i][0] = int(Misc.map(box[i][0], 0, size_s[0], 0, img_w))
    box[i][1] = int(Misc.map(box[i][1], 0, size_s[1], 0, img_h))
cv2.drawContours(img, [box], -1, (0, 255, 255), 2)  #draw a rectangle composed by four points
#Get the opposite corners of a rectangle
pt1_x, pt1_y = box[0, 0], box[0, 1]
pt2_x, pt2_y = box[1, 0], box[1, 1]
pt3_x, pt3_y = box[2, 0], box[2, 1]
center_x = int((pt1_x + pt3_x) / 2) #center point
center_y = int((pt1_y + pt3_y) / 2)
line_width = int(abs(pt1_x - pt2_x))
cv2.circle(img, (center_x, center_y), 5, (0,0,0), -1) # draw center point
```

5.2 Control Motor

After processing image, control the motor of ArmPi Pro by calling set_velocity.publis() function.

```
if abs(center_x - img_w/2) < 20:
    center_x = img_w/2
x_pid.SetPoint = img_w/2      # set
x_pid.update(center_x)        # current
dx = round(x_pid.output, 2)   # output
dx = 0.8 if dx > 0.8 else dx
dx = -0.8 if dx < -0.8 else dx
set_velocity.publish(100, 90, dx)
```

set_velocity.publish() function is used to control motor. Take code “set_velocity.publish(100, 90, dx)” as example:

The first parameter “100” is the linear velocity.

The second parameter “90” is the direction angle.

The third parameter “dx” is the yaw rate.