Lesson 2 Auto Shooting

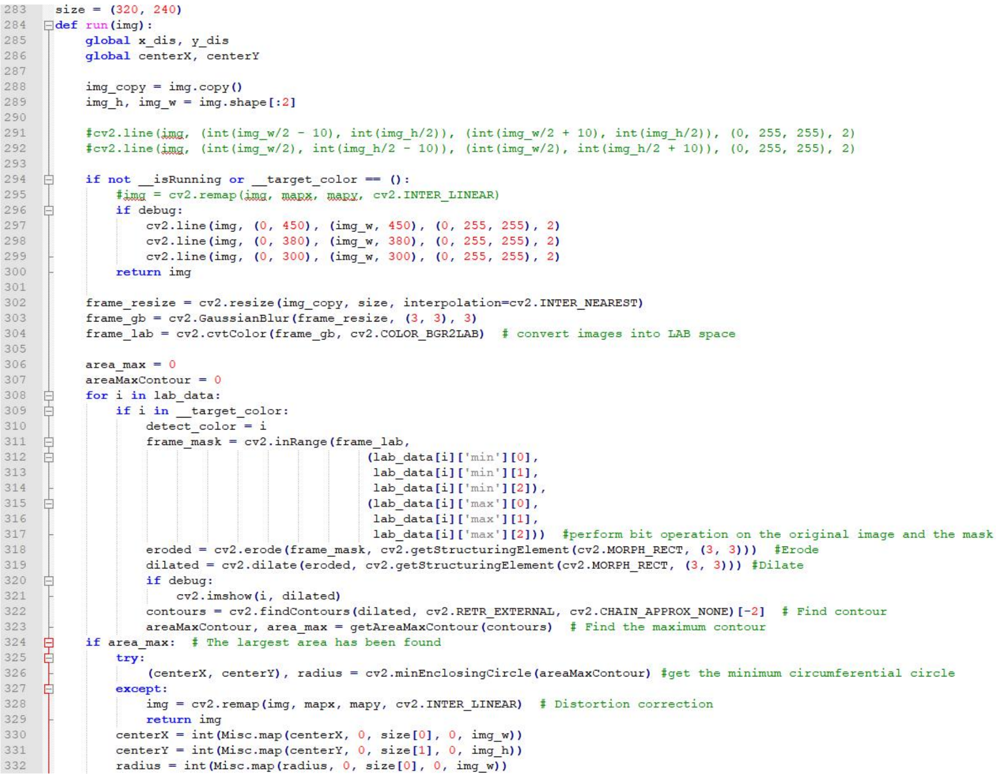
1. Program Logic

How do the TonyPi robot play football (soccer) penalty shot?

First, program TonyPi to recognize colors with Lab color space. Convert the RGB color space to Lab, image binarization, and then perform operations such as expansion and corrosion to obtain an outline containing only the target color. Use circles to frame the color outline to realize object color recognition.

Secondly, judge whether the object is in the central position after receiving the image feedback. If yes, call TonyPi to move forward to the target until it reaches the set range, and then execute the kicking action; otherwise, the robot will move left or right to the center of the target first.

The source code of the program is located in ： /home/pi/TonyPi/Functions/KickBall.py



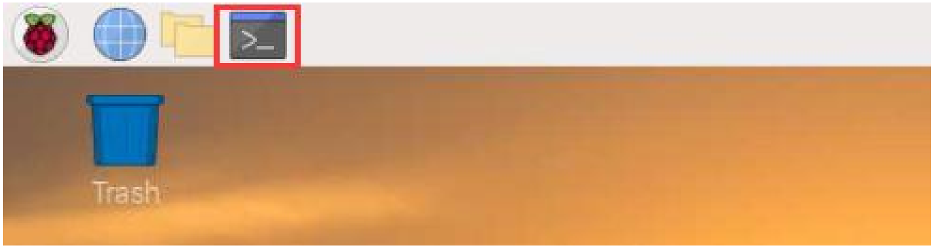
```
size = (320, 240)
def run(img):
    global x_dis, y_dis
    global centerX, centerY

    img_copy = img.copy()
    img_h, img_w = img.shape[:2]

    #cv2.line(img, (int(img_w/2 - 10), int(img_h/2)), (int(img_w/2 + 10), int(img_h/2)), (0, 255, 255), 2)
    #cv2.line(img, (int(img_w/2), int(img_h/2 - 10)), (int(img_w/2), int(img_h/2 + 10)), (0, 255, 255), 2)

    if not __isRunning or __target_color == ():
        #img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
        if debug:
            cv2.line(img, (0, 450), (img_w, 450), (0, 255, 255), 2)
            cv2.line(img, (0, 380), (img_w, 380), (0, 255, 255), 2)
            cv2.line(img, (0, 300), (img_w, 300), (0, 255, 255), 2)
        return img

    frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)
    frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
    frame_lab = cv2.cvtColor(frame_gb, cv2.COLOR_BGR2LAB)  # convert images into LAB space

    area_max = 0
    areaMaxContour = 0
    for i in lab_data:
        if i in __target_color:
            detect_color = i
            frame_mask = cv2.inRange(frame_lab,
                                         (lab_data[i]['min'][0],
                                          lab_data[i]['min'][1],
                                          lab_data[i]['min'][2]),
                                         (lab_data[i]['max'][0],
                                          lab_data[i]['max'][1],
                                          lab_data[i]['max'][2]))  #perform bit operation on the original image and the mask
            eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)))  #Erode
            dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #Dilate
            if debug:
                cv2.imshow(i, dilated)
            contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]  # Find contour
            areaMaxContour, area_max = getAreaMaxContour(contours)  # Find the maximum contour
    if area_max:  # The largest area has been found
        try:
            (centerX, centerY), radius = cv2.minEnclosingCircle(areaMaxContour) #get the minimum circumferential circle
        except:
            img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)  # Distortion correction
            return img
        centerX = int(Misc.map(centerX, 0, size[0], 0, img_w))
        centerY = int(Misc.map(centerY, 0, size[1], 0, img_h))
        radius = int(Misc.map(radius, 0, size[0], 0, img_w))
```

2. Operation Steps

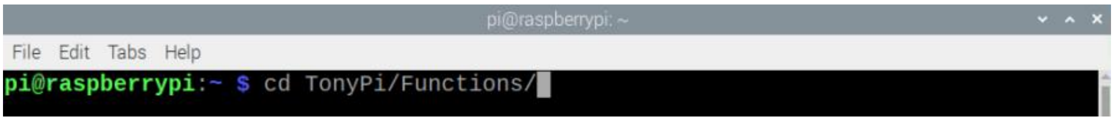
Pay attention to the text format in the input of instructions.

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

2) Click or press "Ctrl+Alt+T" to enter the LX terminal.

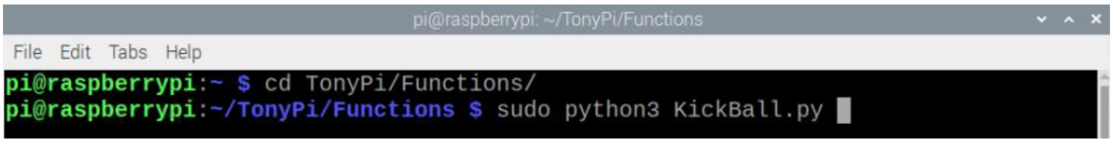


3) Enter "cd TonyPi/Functions/" command, and then press "Enter" to come to the category of games programmings.



```
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

4) Enter "sudo python3 KickBall.py", then press "Enter" to start the game.



```
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo python3 KickBall.py
```

5) 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.

3. Project Outcome

Please use the robot and ball on smooth hard surface.

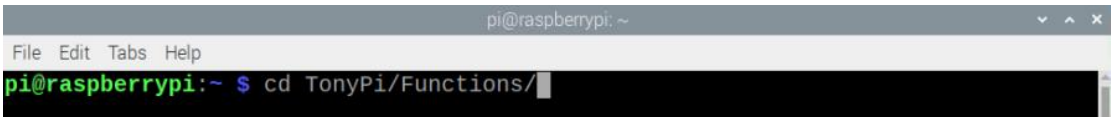
Place the red ball in front of the TonyPi. After recognition, the robot will adjust its position to close the ball and kick it forward.

4. Function Extension

4.1 Modify Program Default Recognition Color

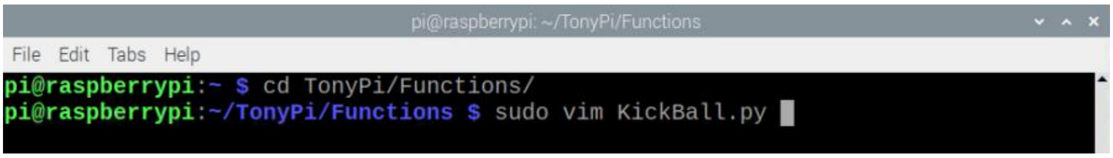
Red, green and blue are the built-in colors in “Auto Shooting” program and red is the default color. In the following steps, we’re going to modify the recognized color as green.

Step1: Enter command “cd TonyPi/Functions/” to the directory where the game program is located.



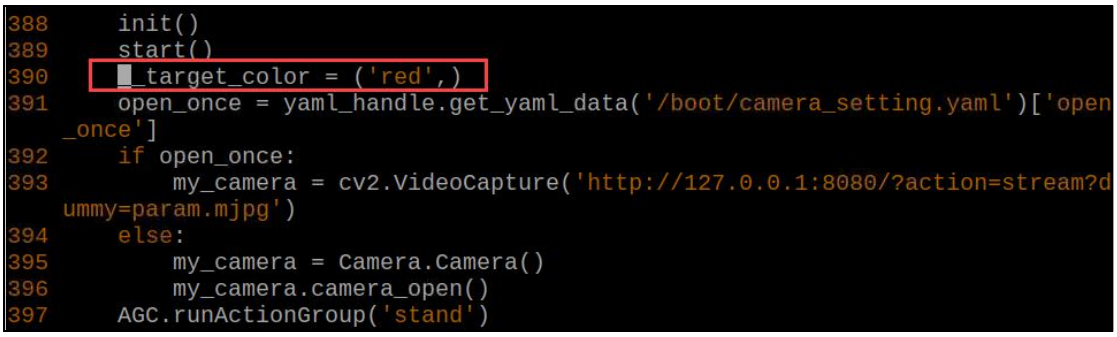
```
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

Step2: Enter command “sudo vim KickBall.py” to go into the game program through vi editor.



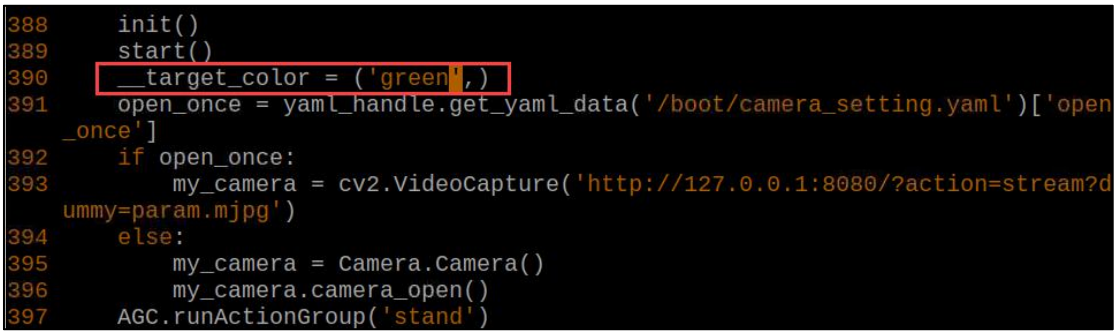
```
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo vim KickBall.py
```

Step3: Input “390” and press “shfit+g” to the line for modification.



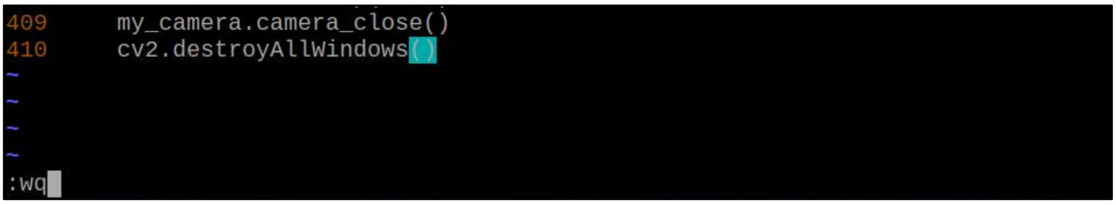
```
388     init()
389     start()
390     _target_color = ('red',)
391     open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open
    _once']
392     if open_once:
393         my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?d
    ummy=param.mjpg')
394     else:
395         my_camera = Camera.Camera()
396         my_camera.camera_open()
397     AGC.runActionGroup('stand')
```

Step4: Press “i” to enter the editing mode, then modify red in _target_color = ‘(red’,) to green. (if you want to recognize blue, please revise to “blue”)



```
    init()
    start()
    __target_color = ('green',)
    open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open_once']
    if open_once:
        my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?dummy=param.mjpg')
    else:
        my_camera = Camera.Camera()
        my_camera.camera_open()
    AGC.runActionGroup('stand')
```

Step5: Press "Esc" to enter last line command mode. Input ":wq" to save the file and exit the editor.

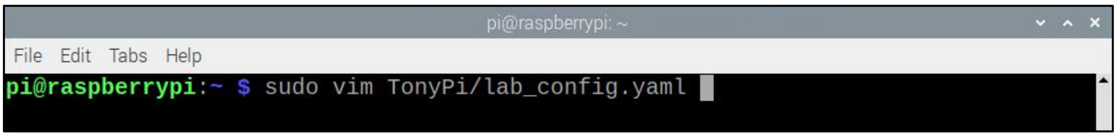


```
    my_camera.camera_close()
    cv2.destroyAllWindows()
~
~
~
~
:wq
```

4.2 Add Recognized Color

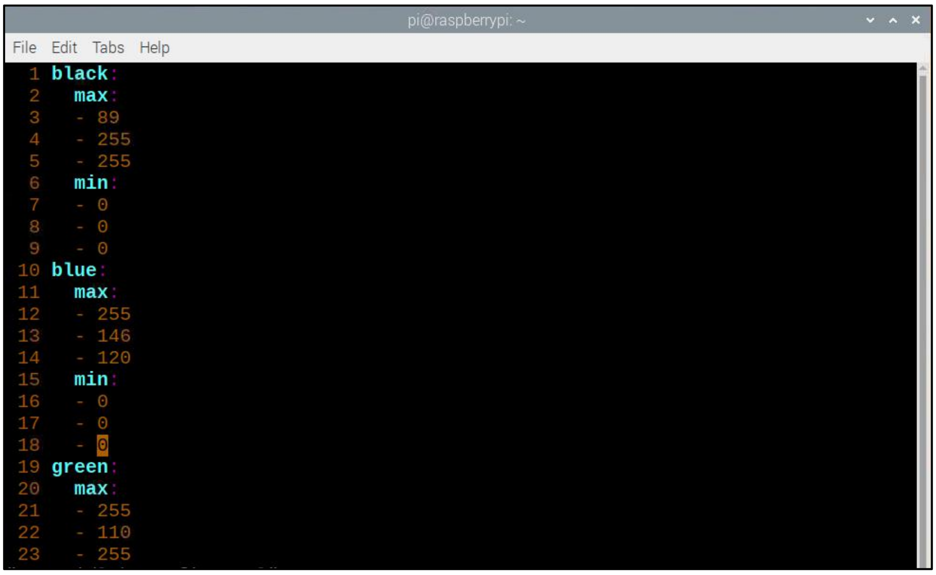
In addition to the built-in recognized colors, you can set other recognized colors in the programming. Take orange as example:

1) Open VNC, input command "**sudo vim TonyPi/lab_config.yaml**" to open Lab color setting document.



```
pi@raspberrypi:~ $ sudo vim TonyPi/lab_config.yaml
```

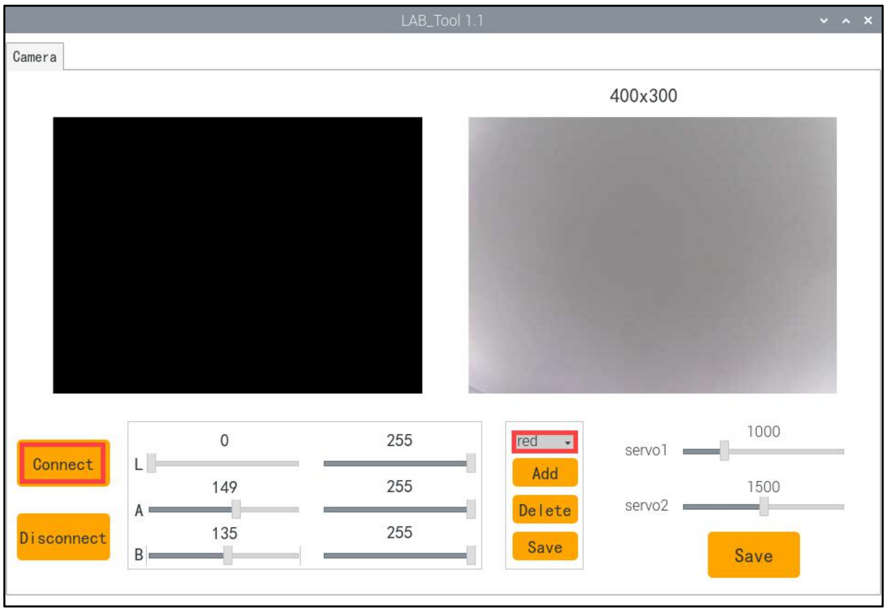
It is recommended to use screenshot to record the initial value.



```
black:
  max:
  - 89
  - 255
  - 255
  min:
  - 0
  - 0
  - 0
blue:
  max:
  - 255
  - 146
  - 120
  min:
  - 0
  - 0
  - 0
green:
  max:
  - 255
  - 110
  - 255
```

2) Click the debugging tool icon in the system desktop. Choose “Run” in the pop-up window.

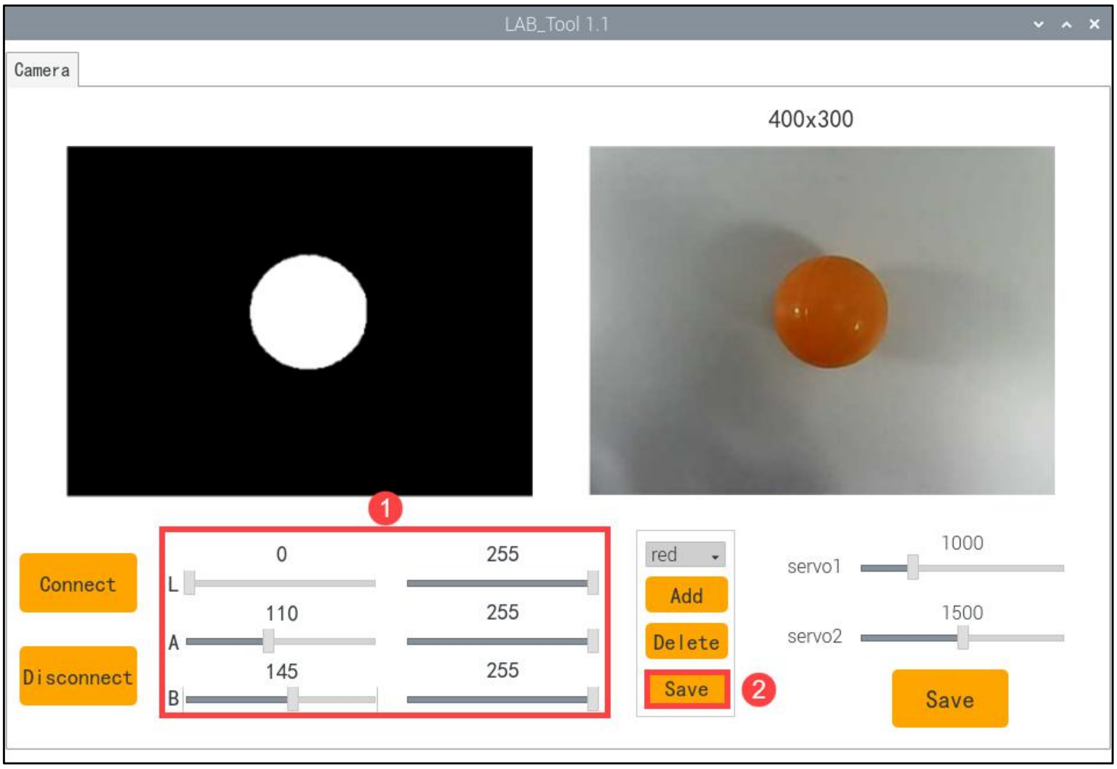
3) Click “Connect” button in the lower left hand. When the interface display the camera returned image, the connection is successful. Select "red" in the right box first.



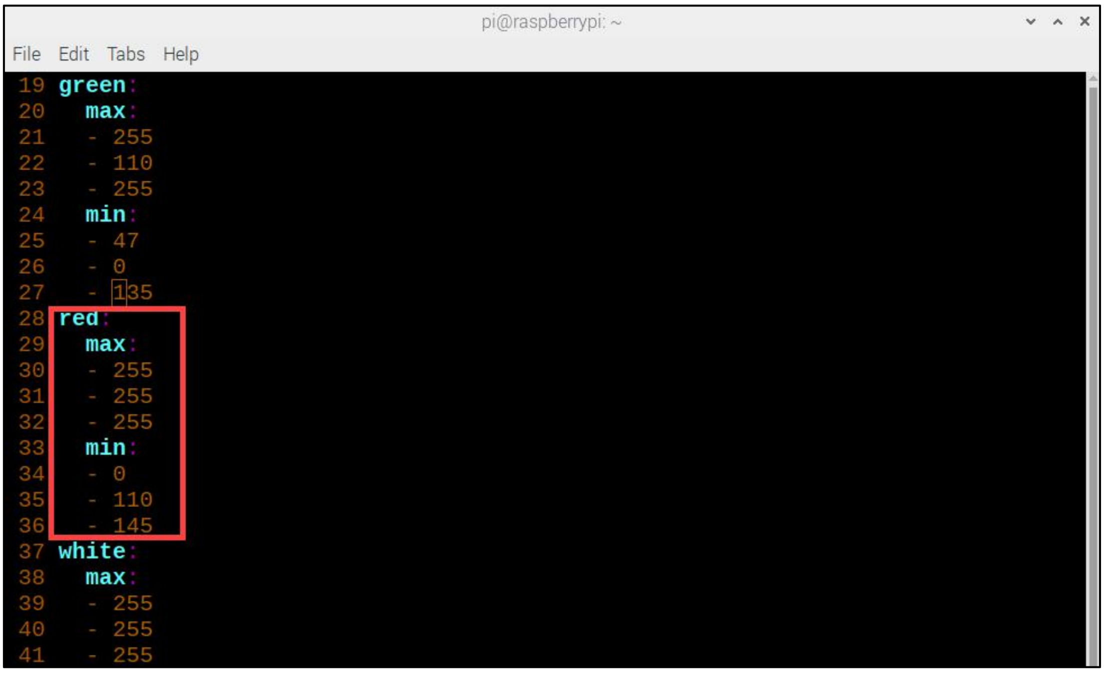
L 0 255
A 149 255
B 135 255
servo1 1000
servo2 1500

4) Drag the corresponding sliders of L, A, and B until the color area to be recognized in the left screen becomes white and other areas become black.

Point the camera at the color you want to recognize. For example, if you want to recognize orange, you can put the orange ball in the camera's field of view. Adjust the corresponding sliders of L, A, and B until the orange part of the left screen becomes white and other colors become black, and then click " Save" button to keep the modified data.



5) After the modification is completed, check whether the modified data was successfully written in. Enter the command again “**sudo vim TonyPi/lab_config.yaml**” to check the color setting parameters.

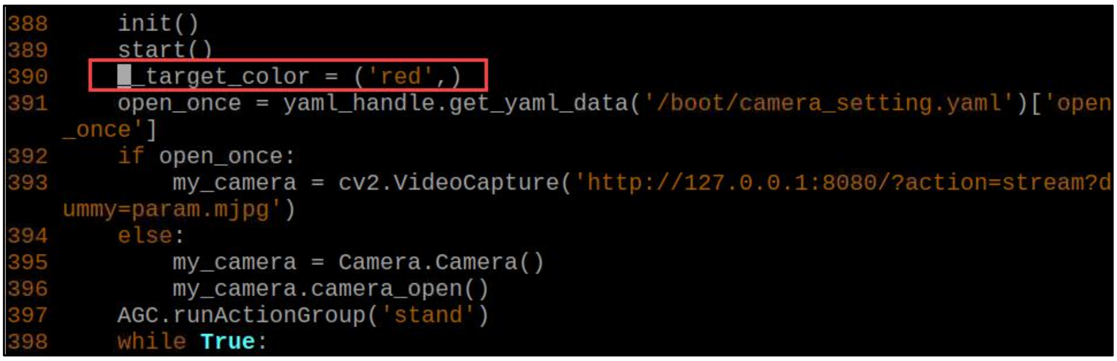


```
green:
  max:
  - 255
  - 110
  - 255
  min:
  - 47
  - 0
  - 135
red:
  max:
  - 255
  - 255
  - 255
  min:
  - 0
  - 110
  - 145
white:
  max:
  - 255
  - 255
  - 255
```

For the game's performance, it's recommended to use the LAB_Tool tool to modify the value back to the initial value after the modification is completed.

6) Check the data in red frame. If the edited value was written in the program, press "Esc" and enter ":wq" to save it and exit.

7) The default recognized color can be set as red according to the "4.1Modify Program Default Recognition Color" in this text.



```
    init()
    start()
    __target_color = ('red',)
    open_once = yaml_handle.get_yaml_data('/boot/camera_setting.yaml')['open_once']
    if open_once:
        my_camera = cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?dummy=param.mjpg')
    else:
        my_camera = Camera.Camera()
        my_camera.camera_open()
    AGC.runActionGroup('stand')
    while True:
```

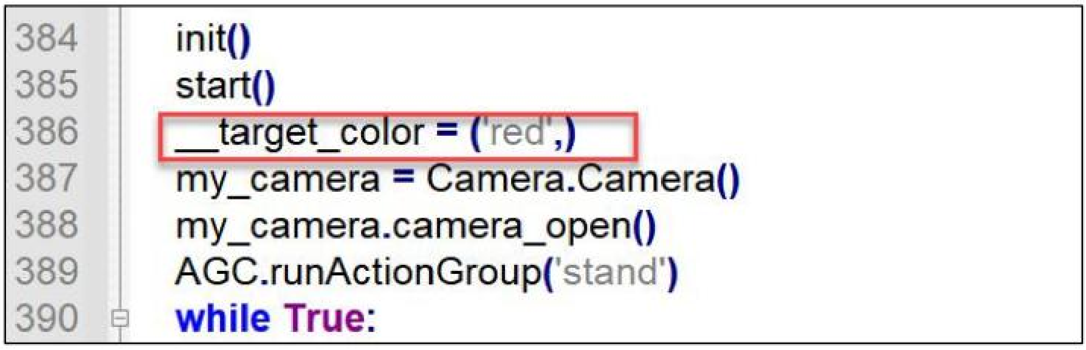
8) Start the game again and put the orange ball in front of the camera. TonyPi will adjust the position to kick the ball after recognizing.

9) If you want to add other colors as recognized color, please operate as the above steps.

5. Program Parameter Instruction

5.1 Color Detection Parameter

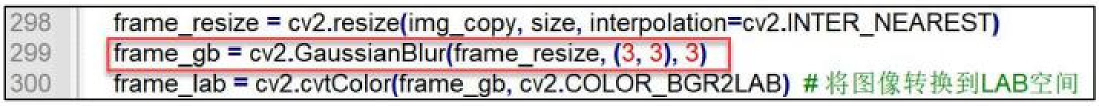
In this program, the default recognized color is red.



```
    init()
    start()
    __target_color = ('red',)
    my_camera = Camera.Camera()
    my_camera.camera_open()
    AGC.runActionGroup('stand')
    while True:
```

The parameters mainly involved in the process of detection 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:



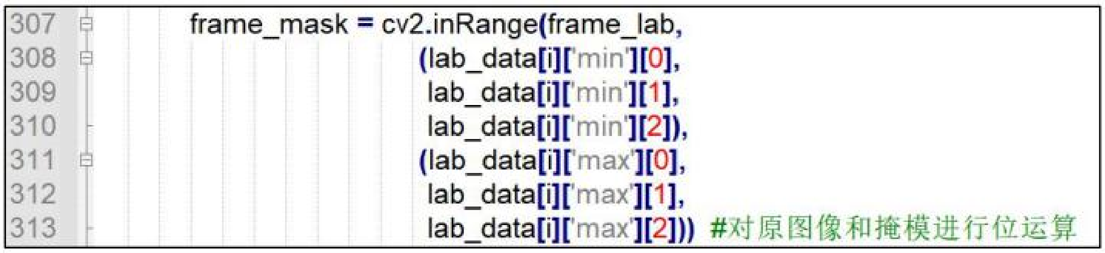
```
        frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)
        frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
        frame_lab = cv2.cvtColor(frame_gb, cv2.COLOR_BGR2LAB)  # 将图像转换到LAB空间
```

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.

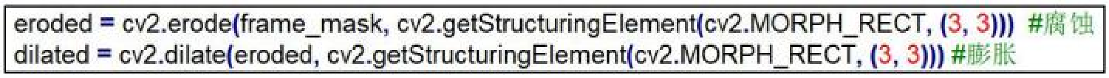
The third parameter “3” is the standard deviation of the Gaussian function along X direction, which is used in Gaussian filters to control the variation around the its mean value. When the data increases, the allowable variation range around the mean value increases, vice verse.

2) Binarize the input image by inRang function, as the figure shown below:



```
        frame_mask = cv2.inRange(frame_lab,
                                 (lab_data[i]['min'][0],
                                  lab_data[i]['min'][1],
                                  lab_data[i]['min'][2]),
                                 (lab_data[i]['max'][0],
                                  lab_data[i]['max'][1],
                                  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:



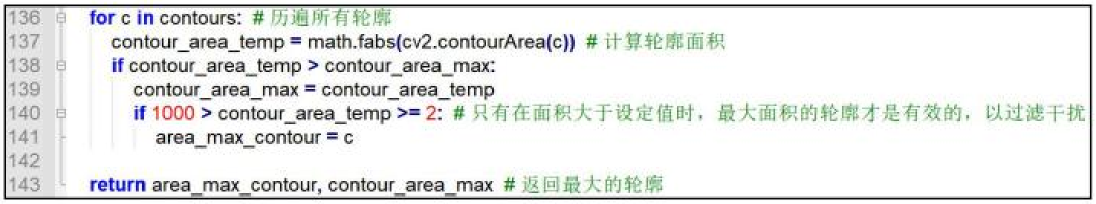
```
eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)))  #腐蚀
dilated = cv2.dilate(eroded, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))) #膨胀
```

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 is rectangle.

The second parameter "(3, 3)" is the size of rectangle. Here is 3×3.

4) Find the object with the biggest contour, as the figure shown below:



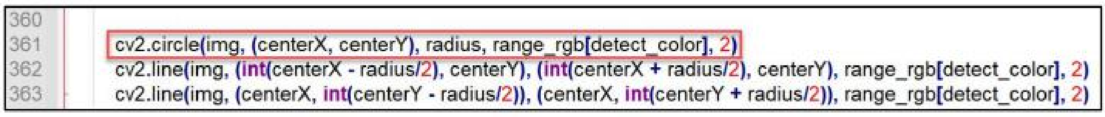
```
for c in contours: # 历遍所有轮廓
    contour_area_temp = math.fabs(cv2.contourArea(c)) # 计算轮廓面积
    if contour_area_temp > contour_area_max:
        contour_area_max = contour_area_temp
        if 1000 > contour_area_temp >= 2: # 只有在面积大于设定值时，最大面积的轮廓才是有效的，以过滤干扰
            area_max_contour = c

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.

5.2 Color Recognition Parameter

1) The control parameters involved in color recognition are as follow:

When the robot recognizes the red ball, cv2.circle() function can be used to draw a circle in the returned image to circle the ball, as the figure show below:

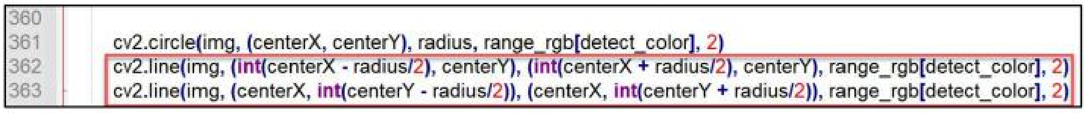


```

    cv2.circle(img, (centerX, centerY), radius, range_rgb[detect_color], 2)
    cv2.line(img, (int(centerX - radius/2), centerY), (int(centerX + radius/2), centerY), range_rgb[detect_color], 2)
    cv2.line(img, (centerX, int(centerY - radius/2)), (centerX, int(centerY + radius/2)), range_rgb[detect_color], 2)
```

The first parameter “img” is the input image. The parameter here is the image of the recognized red ball.

The second parameter “**(centerX, centerY)**” is the coordinate of centre point of drawn circle. (determined according to the detected object)
The third parameter “radius” is the radius of drawn circle. (determined according to the detected object)
The fourth parameter “**range_rgb[detect_color]**” is the line color of drawn circle.
The fifth parameter “2” is the line width of the drawn circle.

2) By using the cv2.line function, draw 2 straight lines, draw a cross, and give a feedback of position information of red ball, as the figure shown below:



```

cv2.circle(img, (centerX, centerY), radius, range_rgb[detect_color], 2)
cv2.line(img, (int(centerX - radius/2), centerY), (int(centerX + radius/2), centerY), range_rgb[detect_color], 2)
cv2.line(img, (centerX, int(centerY - radius/2)), (centerX, int(centerY + radius/2)), range_rgb[detect_color], 2)
```

Take code“cv2.line(img, (int(centerX - radius/2), centerY), (int(centerX + radius/2), centerY), range_rgb[detect_color], 2)” as example.

The first parameter “img” is the specific label image for drawing the line;

The second parameter “(int(centerX - radius/2), centerY)” is the coordinates of the starting point of the straight line.

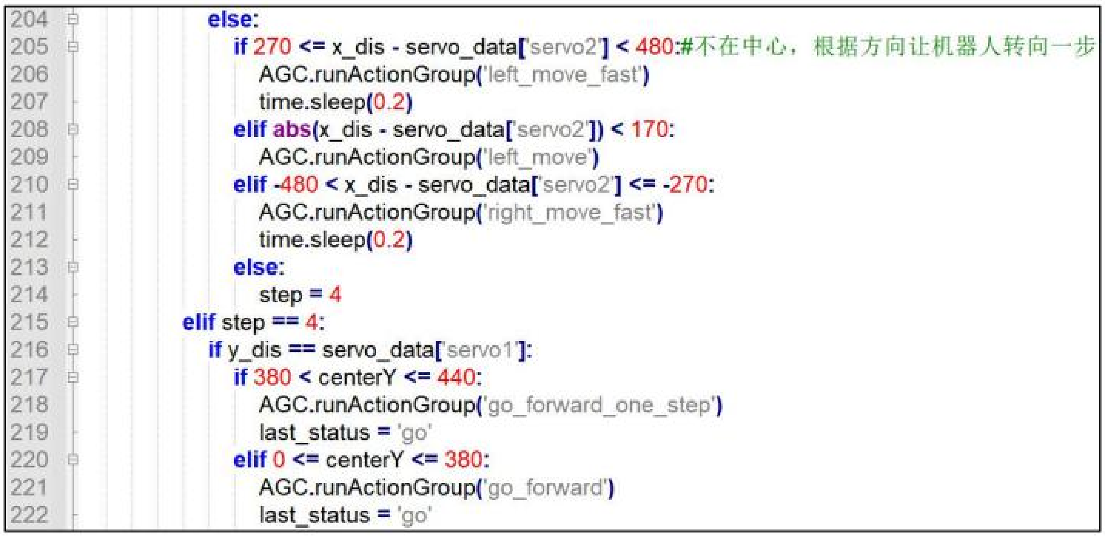
The third parameter “(int(centerX + radius/2), centerY)” is the coordinate of the ending point of the straight line.

The fourth parameter “range_rgb[detect_color]” represents the color of the straight line.

The fifth parameter “2” the width of the straight line.

5.3 Execute Action Parameter

1) After the red ball is recognized, control the movement of robot close to the ball to ensure that the robot can touch the ball when shooting, as the figure shown below:



```
        else:
            if 270 <= x_dis - servo_data['servo2'] < 480:#不在中心，根据方向让机器人转向一步
                AGC.runActionGroup('left_move_fast')
                time.sleep(0.2)
            elif abs(x_dis - servo_data['servo2']) < 170:
                AGC.runActionGroup('left_move')
            elif -480 < x_dis - servo_data['servo2'] <= -270:
                AGC.runActionGroup('right_move_fast')
                time.sleep(0.2)
            else:
                step = 4
    elif step == 4:
        if y_dis == servo_data['servo1']:
            if 380 < centerY <= 440:
                AGC.runActionGroup('go_forward_one_step')
                last_status = 'go'
            elif 0 <= centerY <= 380:
                AGC.runActionGroup('go_forward')
                last_status = 'go'
```

2) After approaching Judge which robot’s foot is closest to the ball, and then call the action group file to control the corresponding foot to shoot the ball.