Program Analysis for Color Recognition

1. File Path

The program file is stored in:

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

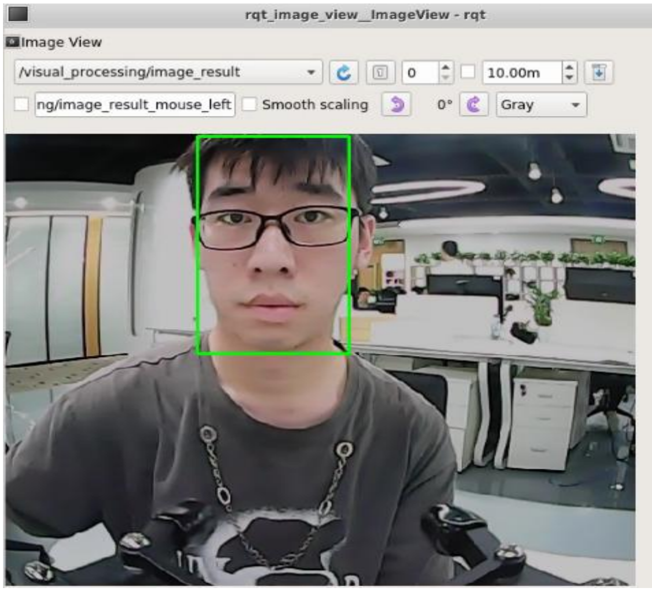
(Image processing)

/home/ubuntu/armpi_pro/src/face_detect/scripts/face_detect_node.py

(Action feedback）

2. Program Performance

After the program is initiated, the robot arm moves back and forth to search for a human face. Within the rqt tool, when the face is detected, the face target will be outlined. At this point, the gripper for the robotic arm will move left and right before performing an open-close action.


rqt_image_view__ImageView - rqt
Image View
/visual_processing/image_result
0
10.00m
ng/image_result_mouse_left
Smooth scaling
0°
Gray

3. Program Analysis

Note: please back up the initial program before making any modifications. It is prohibited editing the source code files directly to prevent making changes in an incorrect manner that could lead to robot malfunctions, rendering them irreparable.

3.1 Import Parameter Module

Imported Module	Function
import sys	The sys module of Python is imported to access to system-related functionalities and variables.
import cv2	The OpenCV library of Python is imported to perform image processing and computer vision-related functions.
import time	The time module of Python is imported to perform time-related functionalities, such as delay operations.
import math	The math module of Python is imported to perform mathematical operations and functions.
import rospy	The Python library rosy is imported for communication and interaction with ROS.
import numpy as np	The NumPy library is imported and is renamed as np for performing array and matrix operations.

from armpi_pro import Misc	The Misc module is imported from arm_pi_pro package to handle the recognized rectangular data.
from armpi_pro import apriltag	The apriltag module is imported from arm_pi_pro package to perform Apriltag recognition and processing.
from threading import RLock, Timer	The “RLock” class and “Timer” class is imported from the threading module of Python for thread-related operations.
from std_srvs.srv import *	All service message types are imported from the std_srvs in ROS for defining and using standard service messages.
from std_msgs.msg import *	All message types are imported form the std_msgs package in ROS for defining and using standard messages.
from sensor_msgs.msg import Image	The image message type is imported from the sensor_msgs packages for processing image data.
from visual_processing.msg import Result	The Result message type is imported from the visual_processing package for the message of image processing results.
from visual_processing.srv import SetParam	The SetParam service type is imported from the visual_processing packages for using customs service related to parameter

	settings.
from sensor.msg import Led	The Led message type is imported form the sensor.msg module for controlling or representing the LED status on a sensor.
from chassis_control.msg import *	All message types are imported from the chassis_control.msg module, which indicated that all message types defined in this module is imported to perform the chassis control.
from visual_patrol.srv import SetTarget	The SetTarget service type is imported from the visual_patrol.srv module is used to set a target for line following.
from hiwonder_servo_msgs.msg import MultiRawIdPosDur	The MultiRawIdPosDur message type is imported from the hiwonder_servo_msgs.msg module for controlling servos.
from armpi_pro import PID	The PID class is imported from the armpi_pro module to perform PID algorithm.
from armpi_pro import bus_servo_control	The bus_servo_control module is imported from the armpi_pro module, including the functions and methods related to the servo control.
from kinematics import ik_transform	The ik_transform function is imported from

	the kinematics module to perform conversion of inverse kinematics.

3.2 Program Logic

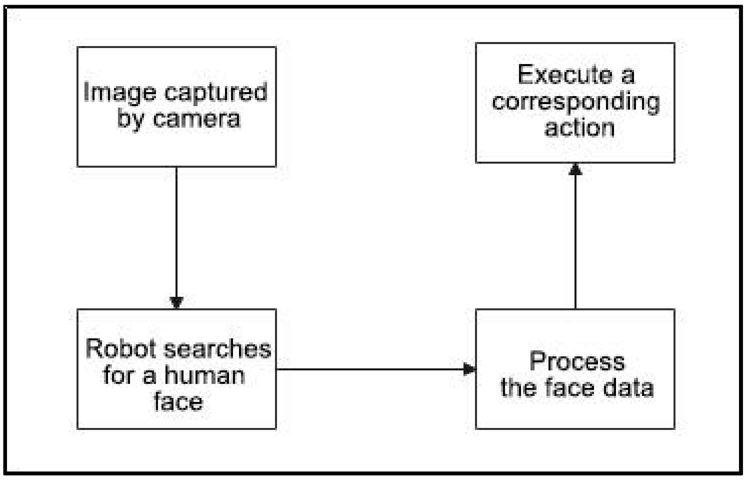
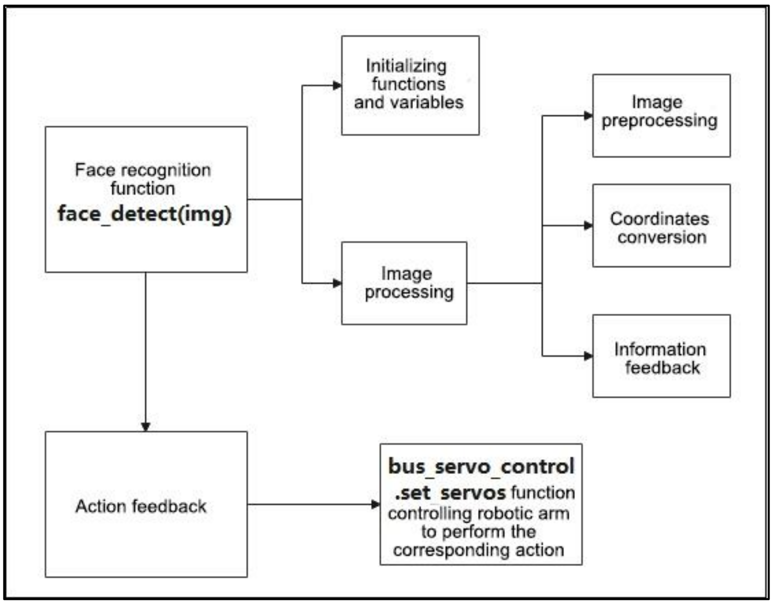


Image captured by camera
Robot searches for a human face
Process the face data
Execute a corresponding action

Obtaining the image information through the camera, and then control the robotic arm to move left and right to search for a human face. Process the obtained data of the human face and frame the human face on live feed image. Lastly, the robotic arm will first move left and right, and then perform open-close action.

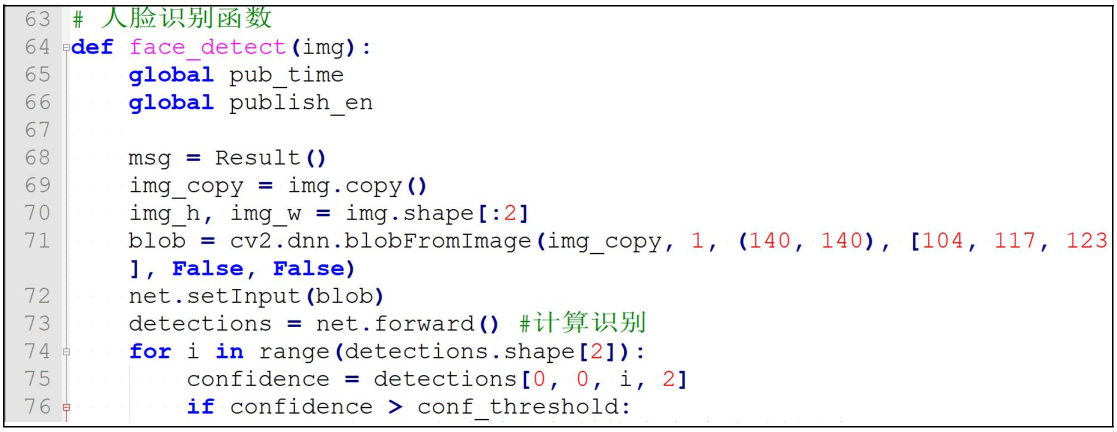
3.3 Code Analysis


Face recognition function face_detect(img)
Initializing functions and variables
Image processing
Image preprocessing
Coordinates conversion
Information feedback
Action feedback
bus_servo_control .set_servos function controlling robotic arm to perform the corresponding action

Seen from the flow diagram, the program is mainly used for facial recognition and action feedback. From the above flow diagram, the program is mainly used for color recognition and servo control. The following content is analyzed based on the above flow diagram.

3.3.1 Image Processing

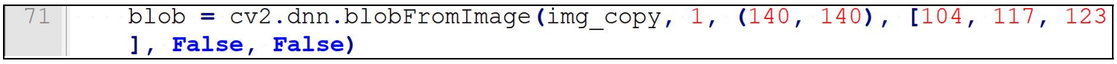
Initializing functions and variables



```
# 人脸识别函数
def face_detect(img):
    global pub_time
    global publish_en
    
    msg = Result()
    img_copy = img.copy()
    img_h, img_w = img.shape[:2]
    blob = cv2.dnn.blobFromImage(img_copy, 1, (140, 140), [104, 117, 123
    ], False, False)
    net.setInput(blob)
    detections = net.forward() #计算识别
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > conf_threshold:
```

Image Pre-processing

Using the cv2.dnn.blobFromImage() function from cv2 library to perform pre-processing on image.



```
    blob = cv2.dnn.blobFromImage(img_copy, 1, (140, 140), [104, 117, 123
    ], False, False)
```

The first parameter “**img_copy**” represents the input image.

The second parameter "**1**" is the scale factor for the image after mean subtraction is performed.

The third parameter "**(140, 140)**" represents the spatial dimensions of the output image, with the values denoting a width (w) of 150 and a height (h) of 150.

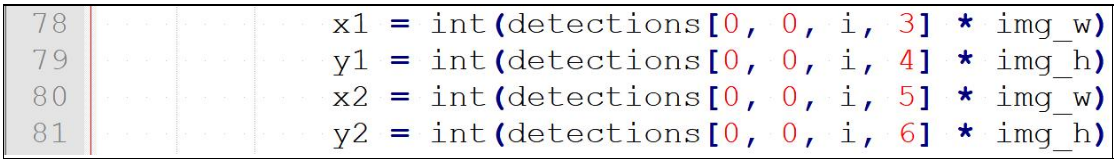
The fourth parameter "[104, 117, 123]" signifies the values subtracted from each channel.

In OpenCV, the channel order is B, G, R. Here, the values imply subtracting 104 from the B channel, 117 from the G channel, and 123 from the R channel. The fifth parameter "False" determines whether to swap the R and B channels. By default, it is set to "False," meaning no swapping of R and B channels. If the mean subtraction order is assumed to be R, G, B, then R and B channels need to be swapped, which would require setting this parameter to "True."

The sixth parameter "False" decides whether to crop the image. By default, it is set to "False," implying no image cropping. The image's size is adjusted directly, while preserving the aspect ratio. If set to "True," the image is first scaled proportionally, and then cropped from its center according to the dimensions specified in parameter three.

Coordinates Conversion

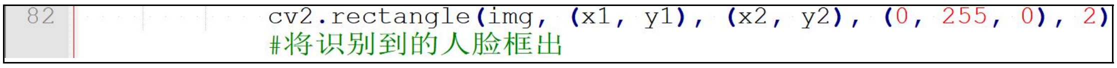
During the preprocessing process, the image undergoes scaling, resulting in mismatched coordinates for the detected faces and the actual scene. Therefore, after completing image preprocessing, it is necessary to perform coordinate transformation.



```
            x1 = int(detections[0, 0, i, 3] * img_w)
            y1 = int(detections[0, 0, i, 4] * img_h)
            x2 = int(detections[0, 0, i, 5] * img_w)
            y2 = int(detections[0, 0, i, 6] * img_h)
```

Information Feedback

By using the rectangle() function from the cv2 library, the faces within the returned image are highlighted with rectangular bounding boxes.



```
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            #将识别到的人脸框出
```

The parameters within the function parentheses are as follows:

The first parameter "img" represents the input image.

The second parameter "(x1, y1)" denotes the starting coordinates of the rectangle.

The third parameter "(x2, y2)" indicates the ending coordinates of the rectangle.

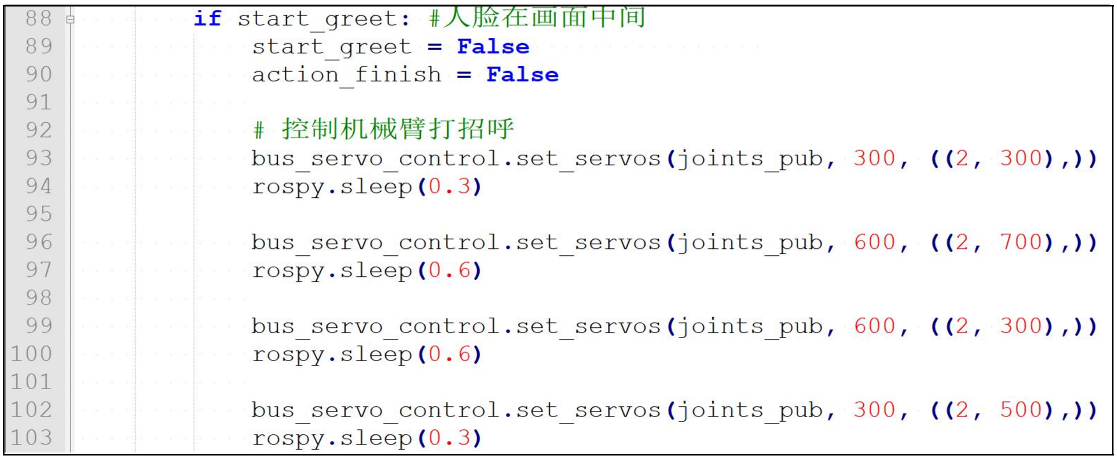
The fourth parameter "(0, 255, 0)" represents the color of the rectangle's outline, using the BGR order; in this case, it's green.

The fifth parameter "2" is the width of the rectangle's outline.

A value of "-1" means that the rectangle will be filled with the color specified in parameter four.

3.3.2 Action Feedback

When a human face is detected, ArmPi Pro is controlled to execute a corresponding action by invoking the bus_servo_control.set_servos() function from the hiwonder_servo_msgs.msg library.



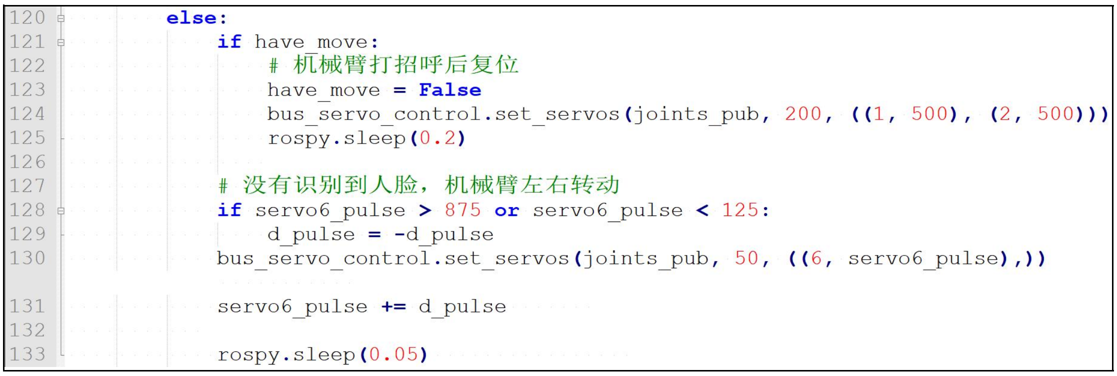
```
if start_greet: #人脸在画面中间
    start_greet = False
    action_finish = False

    # 控制机械臂打招呼
    bus_servo_control.set_servos(joints_pub, 300, ((2, 300),))
    rospy.sleep(0.3)

    bus_servo_control.set_servos(joints_pub, 600, ((2, 700),))
    rospy.sleep(0.6)

    bus_servo_control.set_servos(joints_pub, 600, ((2, 300),))
    rospy.sleep(0.6)

    bus_servo_control.set_servos(joints_pub, 300, ((2, 500),))
    rospy.sleep(0.3)
```



```
else:
    if have_move:
        # 机械臂打招呼后复位
        have_move = False
        bus_servo_control.set_servos(joints_pub, 200, ((1, 500), (2, 500)))
        rospy.sleep(0.2)

    # 没有识别到人脸，机械臂左右转动
    if servo6_pulse > 875 or servo6_pulse < 125:
        d_pulse = -d_pulse
    bus_servo_control.set_servos(joints_pub, 50, ((6, servo6_pulse),))

    servo6_pulse += d_pulse

    rospy.sleep(0.05)
```

Taking the code example "bus_servo_control.set_servos(joints_pub, 300, ((2, 300),))" as reference, the meanings of the parameters within the parentheses are as follows:

The first parameter "(joints_pub)" is for publishing servo control messages. The second parameter, "300," represents the runtime duration.

The third parameter, "((2, 300),)," consists of tuples where:

"2" is the servo motor number.

"300" is the angle of the servo motor.