Program Analysis

1. File Path

The program file is stored in:

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

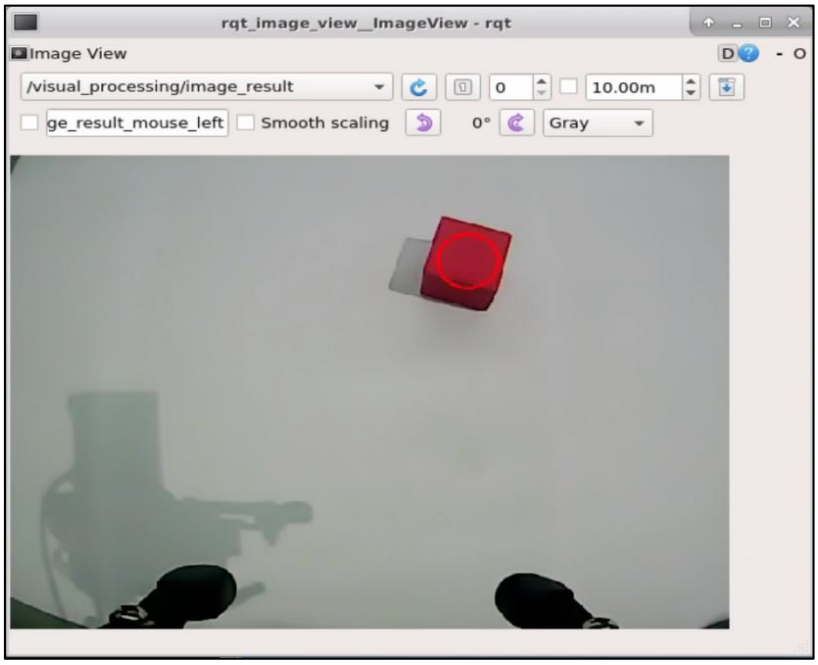
(image analysis）

/home/ubuntu/armpi_pro/src/intelligent_grasp/scripts/intelligent_grasp_node.py

(picking control)

2. Program Performance

After the game starts, the robotic arm will rotate back and forth to search for target. The target will be marked with a bounding box once it is detected. At this pint, robotic arm will slowly move towards the target, and then grip and place it at a specific position.



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

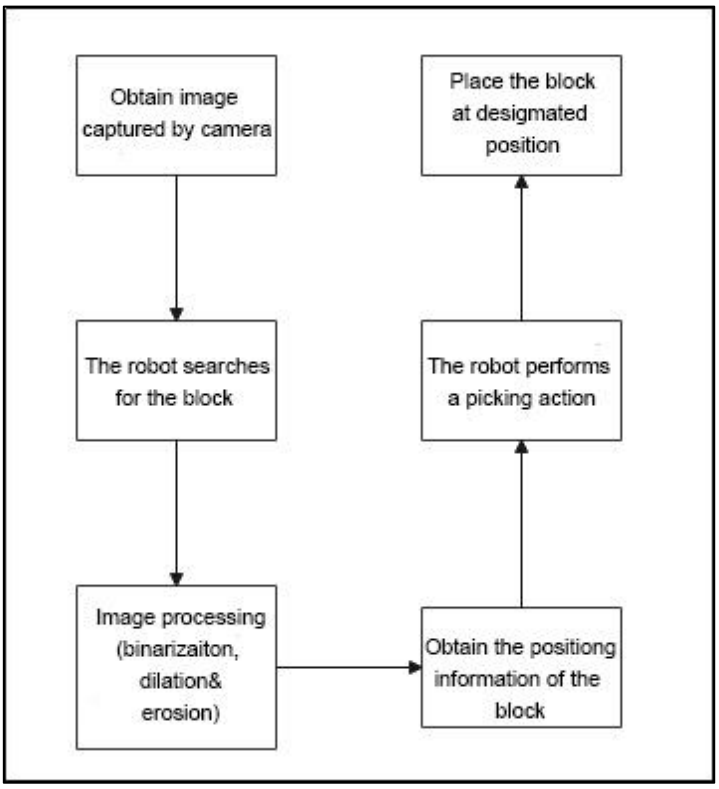
Imported Module	Function
import sys	The sys module of Python is imported to access to system-related functionalities and variables.
import cv2	The OpenCV library of Python is imported to perform image processing and computer vision-related functions.
import time	The time module of Python is imported 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.

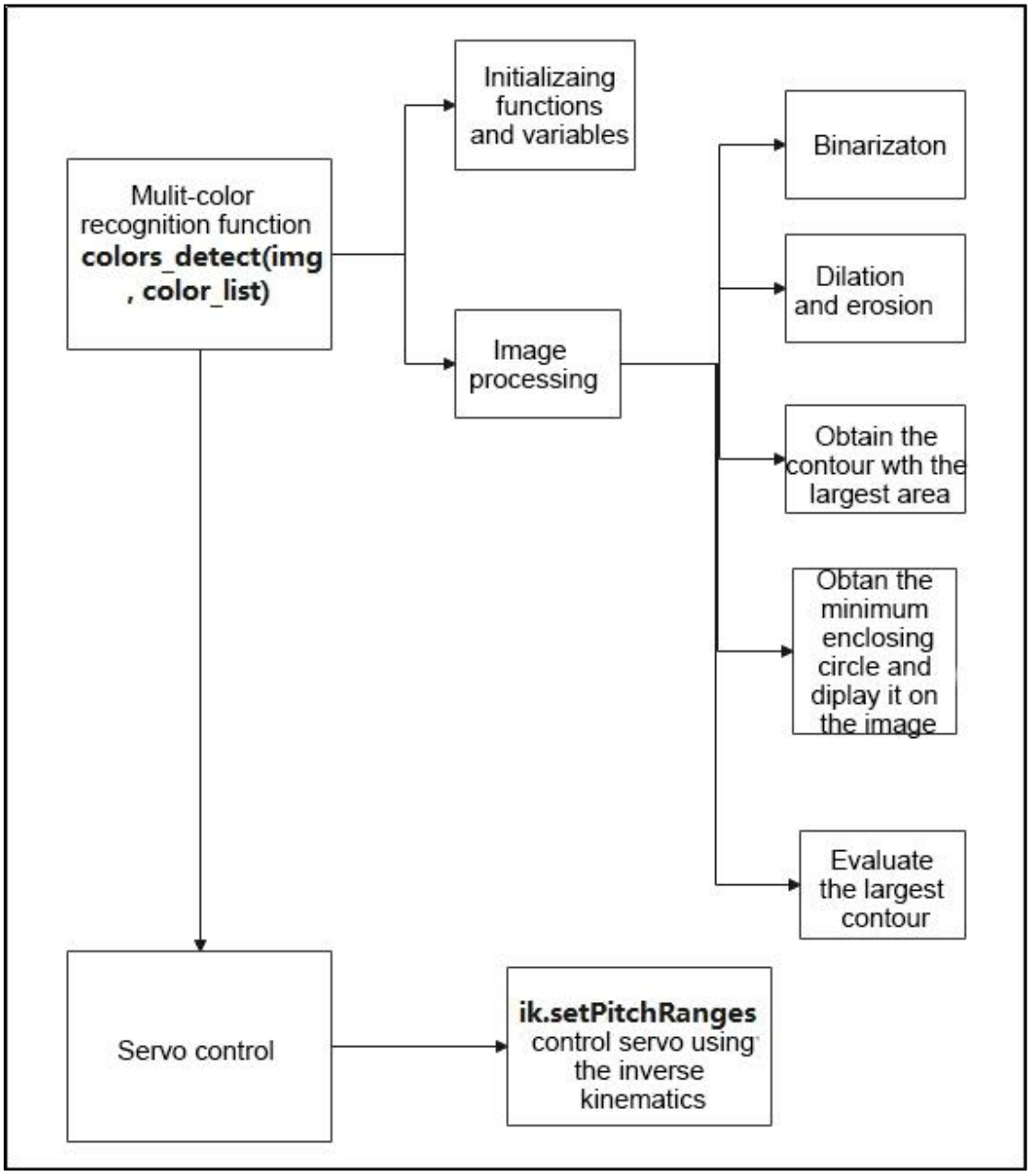
3.2 Program Logic



The image information is obtained through camera,

By using the camera to capture the visual information, the robotic arm moves back and forth to locate a colored block. When the block is recognized, the detected image will be processed to obtain the position information of the colored block. Then, control the robotic arm to grip the block and place it at a designated position.

3.3 Code Analysis



From the above diagram flow, the program is mainly used for multiple colors recognition function and servo control.

3.3.1 Image Processing

Initializing Functions and variables

```
# 多颜色识别函数
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)  # 将图像转换到LAB空间
```

Binarization

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

```
                frame_mask = cv2.inRange(frame_lab, tuple(color_range['min'
                ]), tuple(color_range['max']))  # 对原图像和掩模进行位运算
```

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.

Dilation and Erosion

To reduce interference and make a smooth image, it is necessary to perform dilation and erosion operations on the image.

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

erode() function is applied to erode image. Here uses an example of the 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.

Obtain the contour of the maximum area

After processing the above image, obtain the contour of the recognition target. The findContours() function from the cv2 library is involved in this process.

```
299                 contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.
                    CHAIN_APPROX_NONE)[-2]          # 找出轮廓
```

The erode() function is applied to erode. Here uses an example of code "contours = cv2.findContours(dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)[-2]".

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 approximation method for contours.

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.

```
300                 areaMaxContour, area_max = getAreaMaxContour(contours)
                                                    # 找出最大轮廓
301                 if areaMaxContour is not None:
302                     if area_max > max_area:#找最大面积
```

Obtain Position Information

The minAreaRect() function from the cv2 library is used to obtain the minimum enclosing circle and its center coordinates for the target contour. The circle() function is employed to display the circumferential in the feedback image.

```
            (centerx, centery), radius = cv2.minEnclosingCircle(
            areaMaxContour_max)  # 获取最小外接圆
            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))
            radius = int(Misc.map(radius, 0, size_m[0], 0, img_w))
            cv2.circle(img, (msg.center_x, msg.center_y), radius+5, range_rgb
            [color_area_max], 2)
```

Determine the color block with the largest area.

```
            if color_area_max == 'red':  #红色最大
                msg.data = 1
            elif color_area_max == 'green':  #绿色最大
                msg.data = 2
            elif color_area_max == 'blue':  #蓝色最大
                msg.data = 3
```

3.3.2 Gripping Control

The position of the target on x, y and z axes are obtained after processing image. Then get the target position calculated by inverse kinematics and grip it.

```
    while __isRunning:
        if arm_move and detect_color != 'None': # 等待可以夹取
            target_color = detect_color  # 暂存目标颜色
            set_rgb(target_color)     # 设置rgb灯颜色
            rospy.sleep(0.1)
            buzzer_pub.publish(0.1) # 蜂鸣器响一下
            bus_servo_control.set_servos(joints_pub, 500, ((1, 120),))
            #张开机械爪
            rospy.sleep(0.5)
            target = ik.setPitchRanges((0, round(y_dis + offset_y, 4), -0.08), -
            180, -180, 0) #机械臂向下伸
            if target:
                servo_data = target[1]
                bus_servo_control.set_servos(joints_pub, 1000, ((3, servo_data[
                'servo3']), (4, servo_data['servo4']),
                                                                (5, servo_data[
                                                                'servo5']), (6,
                                                                x_dis)))
            rospy.sleep(1.5)
            bus_servo_control.set_servos(joints_pub, 500, ((1, 450),)) #
            闭合机械爪
            rospy.sleep(0.8)
```

The inverse kinematics takes “**ik.setPitchRanges((0, round(y_dis + offset_y, 4), -0.08), -180, -180, 0)**” as example and the meaning of parameters in parentheses are as follow:

The first parameter is “**(0, round(y_dis + offset_y, 4)**”. “0” is the position of the target on x-axis. “**round(y_dis, 4)**” is the position of the target on y-axis. “**round(z_dis, 4)**” is the position of the target on z-axis.

The second parameter "**-180**" is the angle of x-axis.

The third parameter "**-180**" is the range of the pitch angle.

The fourth parameter "**0**" is the range of pitch angle.

The servo control uses an example of code "**bus_servo_control.set_servos(joints_pub, 20, ((3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5']), (6, x_dis)))**" and the meaning of parameters in parentheses is 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']**" and the rest of parameters are the servo angle.