Program Analysis for Line Following

1. File Path

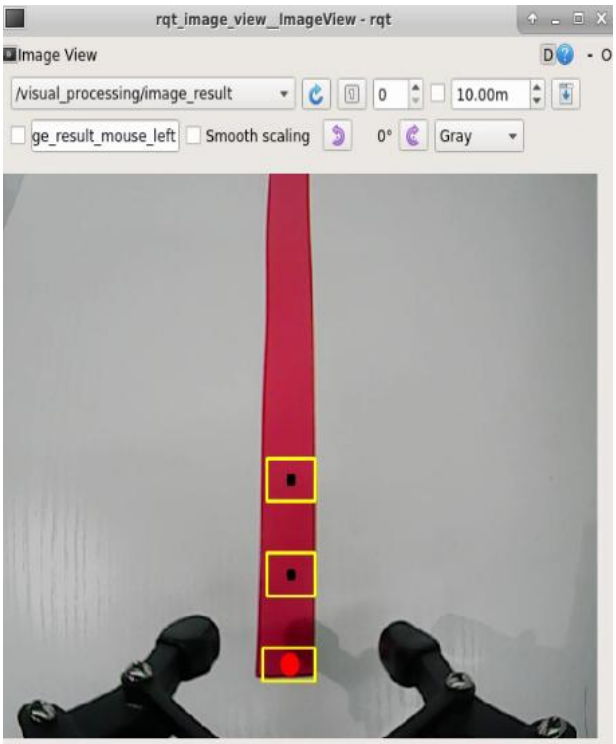
The program file is stored in:

/home/ubuntu/armpi_pro/src/visual_processing/scripts/visual_processing_node.py (Image processing)

/home/ubuntu/armpi_pro/src/visual_patrol/scripts/visual_patrol_node.py (Line following)

2. Program Performance

Through processing the image captured by the camera, the robot will move along the red line.



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

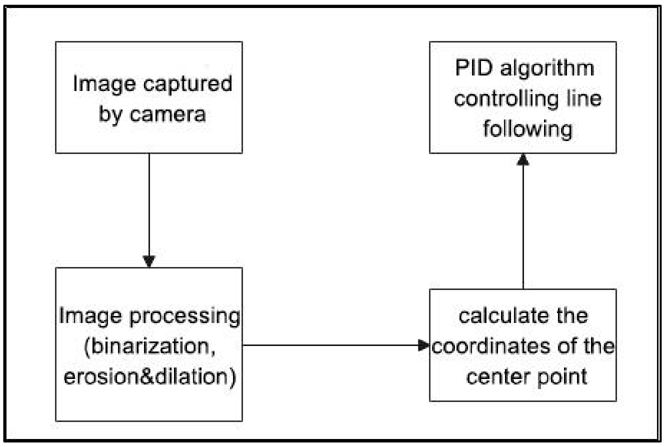
Imported Module	Function
import sys	The sys module of Python is imported to access to system-related functionalities and variables.
import cv2	The OpenCV library of Python is imported to perform image processing and computer vision-related functions.
import time	The time module of Python is imported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

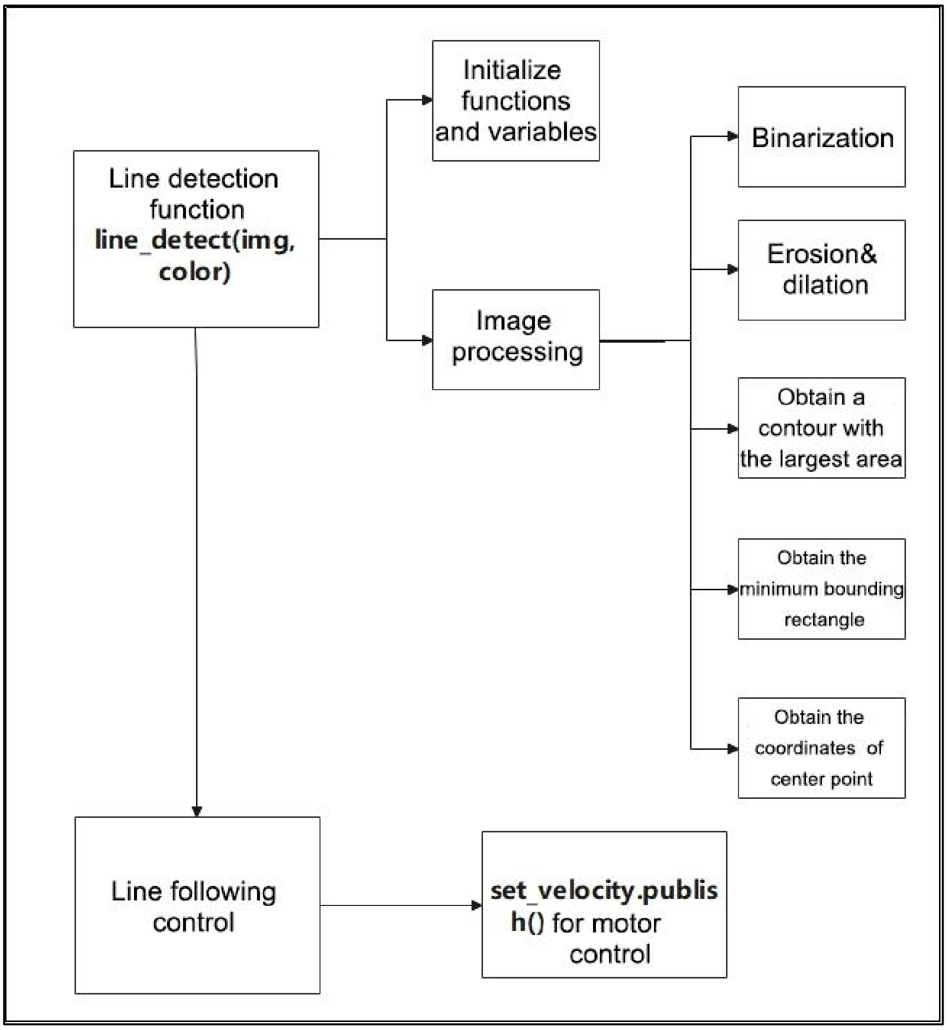
3.2 Program Logic



Obtaining the image information through the camera, and then perform image processing, e.i, binarization. To reduce interference and create smoother images, erosion and dilation processes are applied. Then, the largest area contour and the minimum enclosing rectangle of the target are extracted.

This allows us to calculate the central coordinates of the target. Finally, using the PID algorithm, the robot's base is controlled for line-following movement based on these central coordinates.

3.3 Code Analysis



From the above flow diagram, the program is mainly used for line recognition functions and line-following control. The following content is analyzed based on the above flow diagram.

3.3.1 Image Processing

Initializing functions and variables

```
# 线条识别函数
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)
    #将图像分割成上中下三个部分，这样处理速度会更快，更精确
    for r in roi:
```

Binarization

Using the inRange() function from the cv2 library to perform binarization on 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 create smoother images, erosion and dilation processes are applied

```
            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, (3, 3)))”. 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.

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]  # 找出轮廓
```

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)  # 找出最大轮廓
        if area_max > 50:  # 有找到最大面积
```

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)  #最小外接矩形
            box = np.int0(cv2.boxPoints(rect))        #最小外接矩形的四个顶点
            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)
            #画出四个点组成的矩形
            #获取矩形的对角点
            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) #中心点
            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) #画出中心点
```

3.3.2 Line Following Control

After performing the image processing, control the motors on ArmPi Pro through invoking the set_velocity.publis() function.

```
            # PID算法巡线
            if abs(center_x - img_w/2) < 20:
                center_x = img_w/2
            x_pid.SetPoint = img_w/2        # 设定
            x_pid.update(center_x)          # 当前
            dx = round(x_pid.output, 2)     # 输出
            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 for motor control. Here use an example of the code “set_velocity.publish(100, 90, dx)”:

The first parameter "100" represents the linear velocity, indicating the speed of the motor in millimeters per second. The range is "-100 to 100". When the value is negative, the motor rotates in the opposite direction.

The second parameter "90" represents the heading angle, indicating the direction of movement for the vehicle in degrees. The range is "0 to 360". Where 90 degrees corresponds to forward, 270 degrees to backward, 0 degrees to right, and 180 degrees to left.

The third parameter "dx" represents the yaw angular velocity, indicating the rate of yaw change for the vehicle. It is measured in 5 degrees per second and is constrained in the program to the range of "-0.8 to 0.8". A positive value corresponds to clockwise rotation,