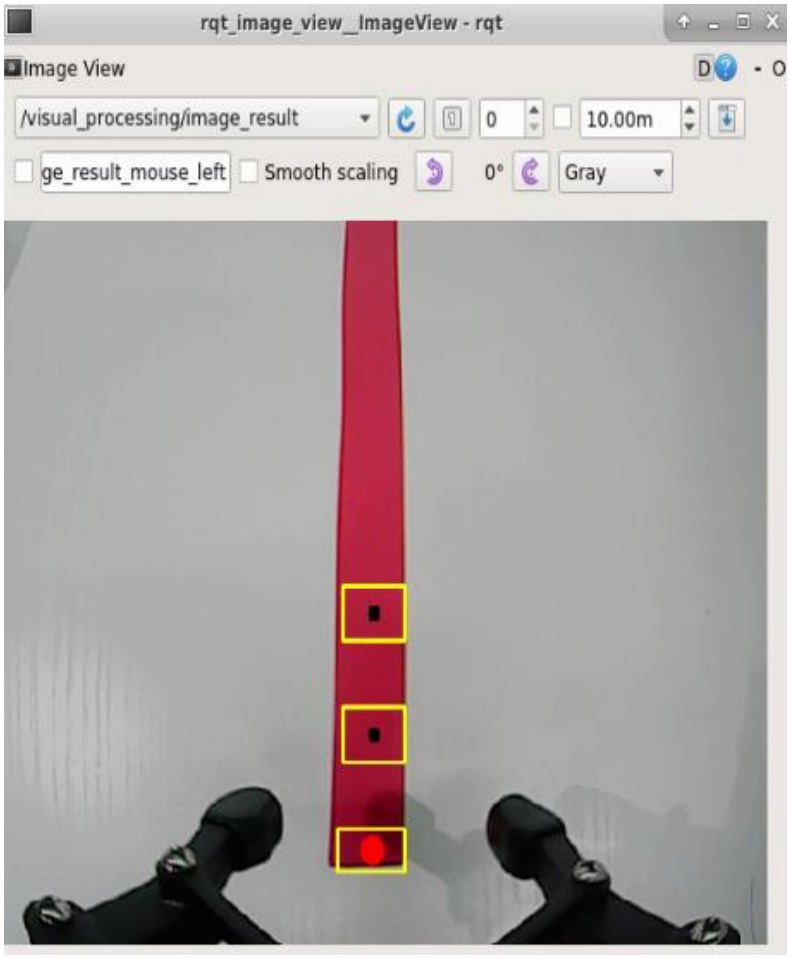
Program Analysis for Line Following

1. File Path

The program corresponding to this lesson is stored in: **/home/pi/MasterPi/Functions/VisualPatrol.py**

2. Performance

Red is the default recognition color. After the program is started, MasterPi will follow the red line.



3. Program Analysis

Note: Before modifying the program, it is necessary to back up the original file. Only after that, proceed with the modifications. Directly modifying the source code files is strictly prohibited to avoid any errors that could lead to the robot malfunctioning and becoming irreparable!!!

3.1 Import Parameter Module

Import module	Function
import sys	Importing Python sys module is used for getting access to the relevant system functionalities and variables.
import cv2	Importing OpenCV library is used for functionalities relayed to the image processing and computer vision.
import time	Importing Python time module is used for time-related functionalities,such as delay operations.
import math	Importing Python math function is used for mathematics operations and functions.
import HiwonderSDK.Board as Board	Importing board library is used for controlling sensor.
import numpy as np	Importing numpy library and renaming it as “np” for performing array and matrix operations

from HiwonderSDK.Misc as Misc	Importing Misc module is used for processing therectangular data identified.
import threading	Provide an environment for multi-thread running
import Camera	Importing Camera library for the use of camera.
PID	Import the PID class from the armpi-pro module. This is used to implement PID control algorithm.
from ArmIK.Transform import *	Used for functions related to the transformation of the robotic arm's pose.
from ArmIK.ArmMoveIK import *	Used for functions regarding to inverse kinematics solution and control.
import yaml_handle	Contain some functionalities or tools related to handling YAML format file.

3.2 Program Logic

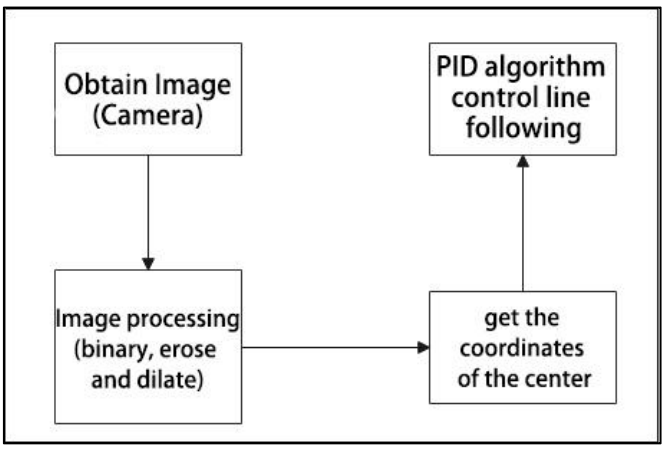
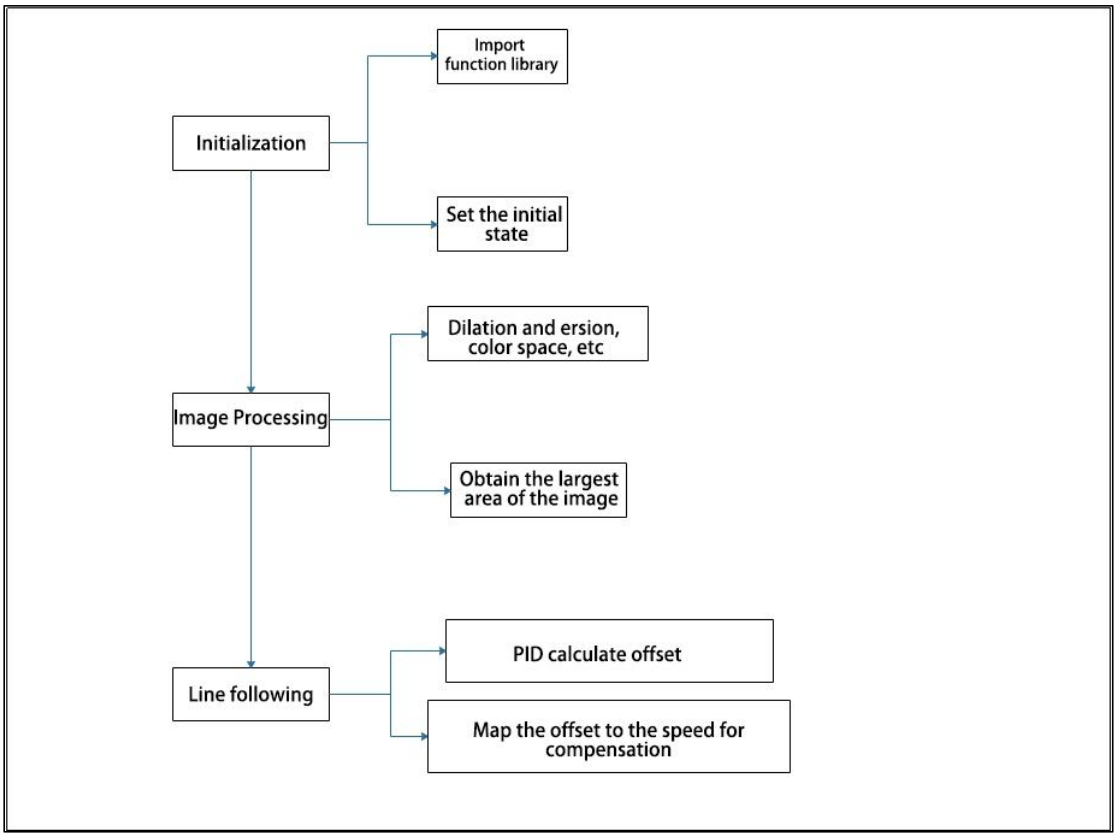


Image information is obtained through the camera, then perform image processing. This includes binarizing the image to reduce interference and make it smoother. Then obtain the contour with the largest area and the minimum enclosing circle of the target, and calculate the central coordinates of the target. Lastly, PID algorithm is used to control the chassis according to the centre coordinates.

3.3 Program Logic and Corresponding Code Analysis



From the above flow diagram, it is primarily used for initialization, image processing and line following. The following content are edited based on this program flow diagram.

3.3.1Initialization

- **Import function library**

You need to first import the function library during the initialization.Regarding the content imported, please refer to “3.1 Import Parameter Module”.

```
import sys
sys.path.append('/home/pi/MasterPi/')
import cv2
import time
import math
import signal
import Camera
import threading
import numpy as np
import yaml_handle
from ArmIK.Transform import *
from ArmIK.ArmMoveIK import *
import HiwonderSDK.Misc as Misc
import HiwonderSDK.Board as Board
from HiwonderSDK.PID import PID
```

◆ **Set the initial status**

After the initialization is complete, it is necessary to set the initial state. This includes configuring the color for line following, the initial position of the servo, the state of the motors and so on.

```
# 设置检测颜色
def setTargetColor(target_color):
    global __target_color

    print("COLOR", target_color)
    __target_color = target_color
    return (True, ())
```

```
# 初始位置
def initMove():

    Board.setPWMServoPulse(1, 1500, 800)
    AK.setPitchRangeMoving((0, 7, 11), -60, -90, 0, 1500)
    MotorStop()
```

3.3.2 Image Processing

◆ **Binarization Processing**

The inRange() function from cv2 library is used to perform image binarization processing.

```
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
```

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 limit of threshold.

◆ Dilation and Erosion Processing

It is necessary to apply open and close operations to image to reduce interference and make image smoother.

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

The erode() function is used to perform erosion on image. Take the example pf the code “**eroded = cv2.erode(frame_mask, cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)))**”. The meaning of the parameters in parentheses are as follow”

The second parameter “**cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))**” is the structuring element or kernel. The first parameter in parentheses is the shape of the kernel. The second parameter is the size of the kernel.

The dilate() function is used to perform dilation on image. The meaning of the parameters in parentheses are the same as that if the **erode()** function.

◆ Obtain position information

To obtain the minimum bounding rectangle of the target contour using the minAreaRect() function from the cv2 library, and then get the coordinates of its four vertices using the boxPoints() function. Afterwards, you can calculate the coordinates of the center point by using the vertex coordinates of the rectangle.

```
212                 rect = cv2.minAreaRect(cnt_large)#最小外接矩形
213                 box = np.int0(cv2.boxPoints(rect))#最小外接矩形的四个顶点
214                 for i in range(4):
215                     box[i, 1] = box[i, 1] + (n - 1)*roi_h + roi[0][0]
216                     box[i, 1] = int(Misc.map(box[i, 1], 0, size[1], 0, img_h))
217                 for i in range(4):
218                     box[i, 0] = int(Misc.map(box[i, 0], 0, size[0], 0, img_w))
219
220                 cv2.drawContours(img, [box], -1, (0,0,255,255), 2)#画出四个点组成的矩形
221
222                 #获取矩形的对角点
223                 pt1_x, pt1_y = box[0, 0], box[0, 1]
224                 pt3_x, pt3_y = box[2, 0], box[2, 1]
225                 center_x, center_y = (pt1_x + pt3_x) / 2, (pt1_y + pt3_y) / 2#中心点
226                 cv2.circle(img, (int(center_x), int(center_y)), 5, (0,0,255), -1)#画出中心点
227                 center .append([center_x, center_y])
```

3.3.2 Lind Following Control

After the image processing are complete, the Board.setMotor() function is called to control the movement of the motor on robot.

```
def move():
    global line_centerx

    i = 0
    while True:
        if __isRunning:
            if line_centerx != -1:

                num = (line_centerx - img_centerx)
                if abs(num) <= 5:  # 偏差比较小，不进行处理
                    pitch_pid.SetPoint = num
                else:
                    pitch_pid.SetPoint = 0
                pitch_pid.update(num)
                tmp = pitch_pid.output    # 获取PID输出值
                tmp = 100 if tmp > 100 else tmp
                tmp = -100 if tmp < -100 else tmp
                base_speed = Misc.map(tmp, -100, 100, -50, 50)  # 速度进行映射
                Board.setMotor(1, int(50-base_speed)) #设置马达速度
                Board.setMotor(2, int(50+base_speed))
                Board.setMotor(2, int(50+base_speed))
                Board.setMotor(3, int(50-base_speed))
                Board.setMotor(4, int(50+base_speed))

```

The Board.setMotor() function is used to control motor. Take an example of the code “**Board.setMotor(1, int(50-base_speed))**”, the meaning of the parameters in parentheses is as follow:

The first parameter “**1**” is the sequential number of the motor, representing motor 1.

The second parameter "**int(50-base_speed)**" is the speed, which represents the current movement speed of the robot plus or minus the speed of the PID compensation.