Program Analysis for Object Tracking

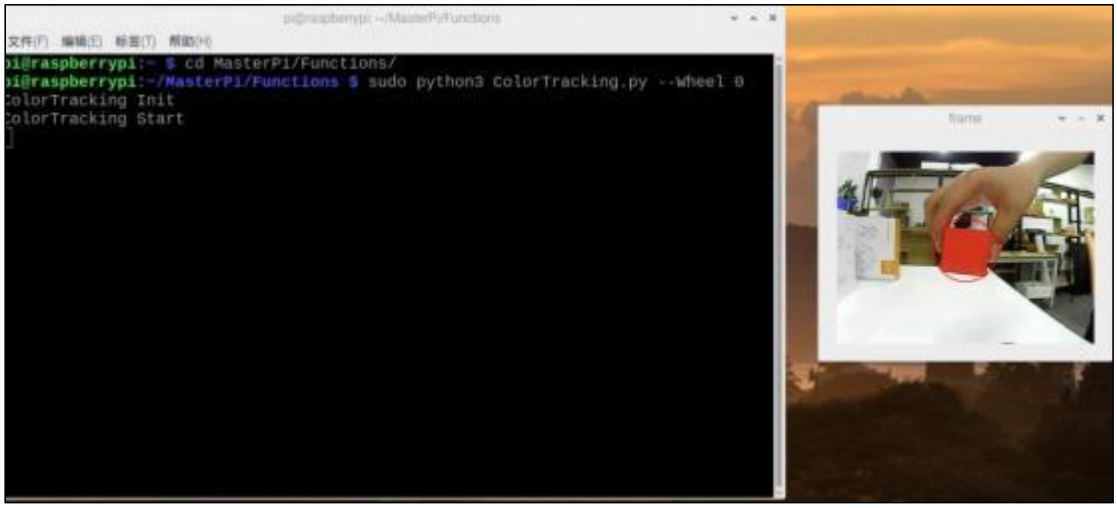
1. File Path

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

2. Performance

The default tracking color is red.

Mode	Outcome
Robotic arm tracking	Robotic arm moves with the colored block, while the car body keeps stationary.
Car tracking	Car moved with the colored block, while the robotic arm keeps stationary.



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

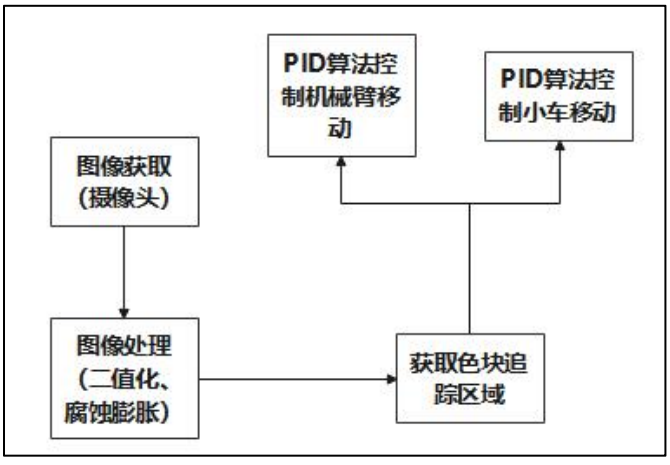
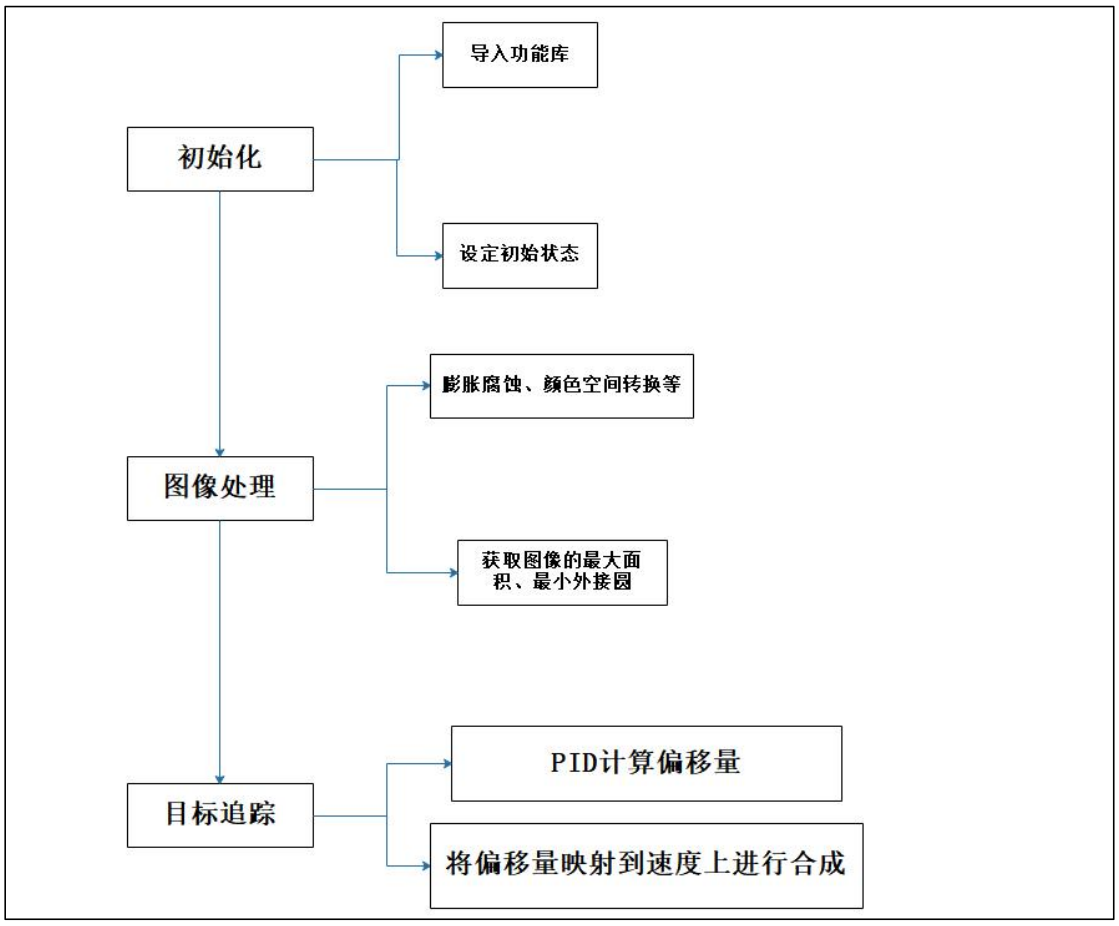


Image information is obtained through the camera, then perform image processing. This includes binarizing the image to reduce interference and make it smoother.

Then obtain the contour with the largest area and the minimum enclosing circle of the target so as to get the region for tracking the colored object. Next, Next, apply the PID algorithm to make the robotic arm turn towards the colored object or make the car drive towards the direction of the colored object.

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 signal
import Camera
import argparse
import threading
import yaml_handle
from ArmIK.Transform import *
from ArmIK.ArmMoveIK import *
import HiwonderSDK.PID as PID
import HiwonderSDK.Misc as Misc
import HiwonderSDK.Board as Board
```

◆ 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, servo1, 800)
    AK.setPitchRangeMoving((0, 8, 18), 0,-90, 90, 1500)

# 设置蜂鸣器
def setBuzzer(timer):
    Board.setBuzzer(0)
    Board.setBuzzer(1)
    time.sleep(timer)
    Board.setBuzzer(0)

# 关闭电机
def MotorStop():
    Board.setMotor(1, 0)
    Board.setMotor(2, 0)
    Board.setMotor(3, 0)
    Board.setMotor(4, 0)
```

3.3.2 Image Processing

◆ Binarization Processing

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

```
223             frame_mask = cv2.inRange(frame_lab,
224                                      (lab_data[detect_color]['min'][0],
225                                       lab_data[detect_color]['min'][1],
226                                       lab_data[detect_color]['min'][2]),
227                                      (lab_data[detect_color]['max'][0],
228                                       lab_data[detect_color]['max'][1],
229                                       lab_data[detect_color]['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.

```
230             opened = cv2.morphologyEx(frame_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))  # 开运算
231             closed = cv2.morphologyEx(opened, cv2.MORPH_CLOSE, np.ones((3, 3), np.uint8))  # 闭运算
```

cv2.morphologyEx(frame_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8)) is an operation to perform opening on a binary image.

The first parameter “frame_mask” is the binary image on which morphological operation will be performed.

The second parameter “cv2.MORPH_OPEN” specifies the type of morphological operation, in this case, it is an opening operation.

The third parameter “np.ones((3, 3), np.uint8)” is the structuring element used in the morphological operation. It defines the shape and size of the operation. In this case , a 3x3 matrix filled with ones is used as the structuring element.

- **Obtain position information**

The getAreaMaxContourt() function in cv2 library is used to obtain the contour with the largest area, and the minimum enclosing circle is obtained through the minEnclosingCircle() function.

```
# 找出面积最大的轮廓
# 参数为要比较的轮廓的列表
def getAreaMaxContour(contours):
    contour_area_temp = 0
    contour_area_max = 0
    areaMaxContour = None
    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 contour_area_temp > 300:  # 只有在面积大于300时，最大面积的轮廓才是有效的，以过滤干扰
                areaMaxContour = c
    return areaMaxContour, contour_area_max  # 返回最大的轮廓
```

```
        if area_max > 1000:  # 有找到最大面积
            (center_x, center_y), radius = cv2.minEnclosingCircle(areaMaxContour)  # 获取最小外接圆
            center_x = int(Misc.map(center_x, 0, size[0], 0, img_w))
            center_y = int(Misc.map(center_y, 0, size[1], 0, img_h))
            radius = int(Misc.map(radius, 0, size[0], 0, img_w))
            if radius > 100:
                return img
```

3.3.2Target Tracking

After the image processing is finished, if you want to start robotic arm tracking, it will be implemented by calling the Board.setPWMServosPulse() function.

```
        else:
            if Motor_:
                MotorStop()
                Motor_ = False

            x_pid.SetPoint = img_w / 2.0  # 设定
            x_pid.update(center_x)  # 当前
            dx = x_pid.output
            x_dis += int(dx)  # 输出
            x_dis = 500 if x_dis < 500 else x_dis
            x_dis = 2500 if x_dis > 2500 else x_dis

            y_pid.SetPoint = img_h / 2.0  # 设定
            y_pid.update(center_y)  # 当前
            dy = y_pid.output
            y_dis += int(dy)  # 输出
            y_dis = 500 if y_dis < 500 else y_dis
            y_dis = 2500 if y_dis > 2500 else y_dis

            Board.setPWMServosPulse([20, 2, 3,int(y_dis), 6,int(x_dis)])
```

The Board.setPWMServosPulse() function is used for controlling servos. Taking the example code "Board.setPWMServosPulse([20, 2, 3, int(y_dis), 6, int(x_dis)])", the parameters inside the parentheses have the following meanings:

The first parameter "**20**" represents the running time, i.e., the duration of each servo movement during tracking.

The second parameter "**2**" represents the number of servos controlled when the robotic arm is tracking. In this case, it refers to 2 servos being controlled.

The third and fourth parameters "3, int(y_dis)" represent controlling servo number 3 to move in the y-direction.

The fifth and sixth parameters "6, int(x_dis)" represents controlling servo number 6 to move in the x-direction.

If the purpose is to control the robot's chassis for tracking, then the Board.setMotor() function is used instead.

```
if __isRunning:   # 检测是否开启玩法

    if enableWheel == True:   #  检测是否开启车身跟随;   enableWheel = True,为开启车身跟随
        Motor_ = True

        if abs(center_x - img_w/2.0) < 15: # 移动幅度比较小，则不需要动
            about_pid.SetPoint = center_x
        else:
            about_pid.SetPoint = img_w/2.0 # 设定
        about_pid.update(center_x) # 当前
        x_speed = -int(about_pid.output)  # 获取PID输出值
        x_speed = -100 if x_speed < -100 else x_speed
        x_speed = 100 if x_speed > 100 else x_speed

        if abs(center_y - img_h/2.0) < 10: # 移动幅度比较小，则不需要动
            go_pid.SetPoint = center_y
        else:
            go_pid.SetPoint = img_h/2.0
        go_pid.update(center_y)
        y_speed = int(go_pid.output)# 获取PID输出值
        y_speed = -100 if y_speed < -100 else y_speed
        y_speed = 100 if y_speed > 100 else y_speed

        speed_1 = int(y_speed + x_speed) # 速度合成
        speed_2 = int(y_speed - x_speed)
        speed_3 = int(y_speed - x_speed)
        speed_4 = int(y_speed + x_speed)


        Board.setMotor(1, speed_1)
        Board.setMotor(2, speed_2)
        Board.setMotor(3, speed_3)
        Board.setMotor(4, speed_4)
```

The function Board.setMotor() is used for motor control. Taking the example code "Board.setMotor(1, speed_1)", the parameters inside the parentheses have the following meanings:

The first parameter "1" represents the motor number, indicating that it is controlling motor number 1.

The second parameter "speed_1" represents the composite speed. This speed is calculated using the previous PID calculation, which takes into account the deviations in the X and Y directions from the target position. The PID algorithm calculates the appropriate speed to control the movement of the robot's motor.

In summary, Board.setMotor() is used to control the motor of the robot, and the second parameter "speed_1" is the calculated speed value based on the PID algorithm to move the robot in the desired direction.