Program Analysis for Color Recognition

1. File Path

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

2. Performance

After the program is started, robot will recognize color and perform different action based on the color recognized, as shown in the below table:

Object color	Buzzer	RGB light	Action	Content printed in frame tool
red	make a single sound	red	“nod”	red
green	make a single sound	green	“shake head”	green
blue	make a single sound	blue	“shake head”	blue

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.
import signal	Used for receiving and processing signals

3.2 Program Logic

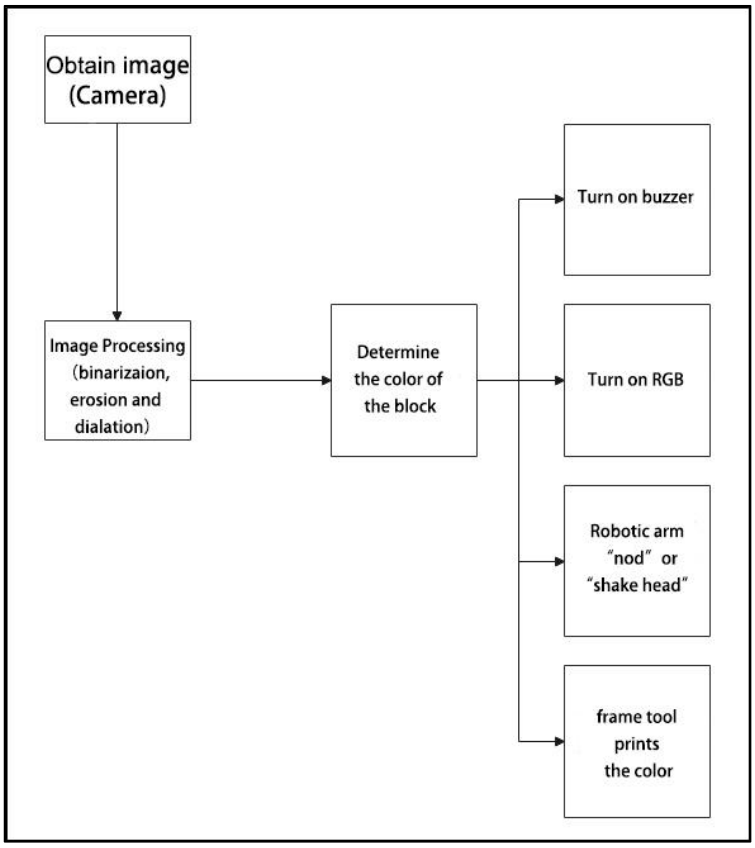
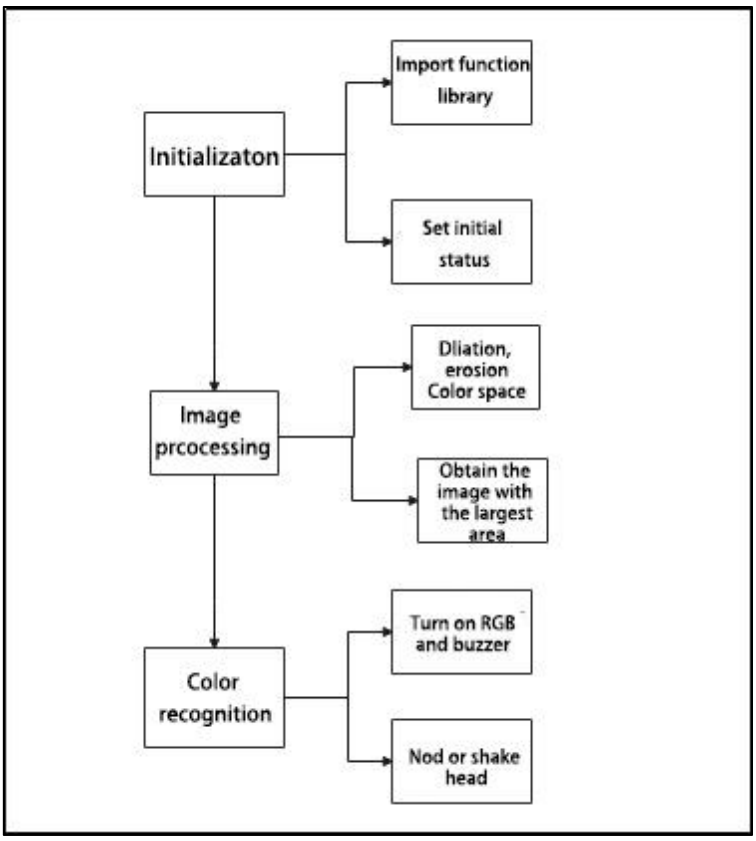


Image information is obtained through the camera, then perform image processing. This includes binarizing the image to reduce interference and make it smoother. Additionally, the image undergoes erosion and dilation processes, then the largest area contour of the target is obtained. Afterward, determine the object color and provide a corresponding feedback.

3.3 Program Logic and Corresponding Code Analysis



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

3.3.1Image Processing

◆ **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”.

```
#!/usr/bin/python3
# coding=utf8
import sys
sys.path.append('/home/pi/MasterPi/')
import cv2
import time
import threading
import yaml_handle
from ArmIK.Transform import *
from ArmIK.ArmMoveIK import *
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, 1500, 800)
    AK.setPitchRangeMoving((0, 7, 11), -60, -90, 0, 1500)
    MotorStop()
```

◆ **Image Pre-processing**

Resize and apply Gaussian blur to the image.

```
        frame_resize = cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST)
        frame_gb = cv2.GaussianBlur(frame_resize, (3, 3), 3)
```

cv2.resize(img_copy, size, interpolation=cv2.INTER_NEAREST) is used to resize the image.

The first parameter “img_copy” is the image to be resized.

The second parameter “size” is the target size.

The third parameter “interpolation” is the interpolation method used to determine the pixel interpolation algorithm used during resizing.

cv2.GaussianBlur(frame_resize, (3, 3), 3) is used to apply Gaussian blur to the image.

The first parameter “frame_resize” is the image to be processed with Gaussian blur.

The second parameter “(3,3)” is the size of the Gaussian kernel, wherr both of the width and height of the kernel are 3.

The third parameter “3” is the standard deviation of the Gaussian kernel, which controls the level of blurring. Larger value will result in a stronger blur effect.

- **Color Space Conversion**

Convert a BGR image to a LAB image.

```
248         frame_lab = cv2.cvtColor(frame_gb, cv2.COLOR_BGR2LAB)  # 将图像转换到LAB空间
```

- **Binarization Processing**

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

```
256             frame_mask = cv2.inRange(frame_lab,
257                                      (lab_data[i]['min'][0],
258                                       lab_data[i]['min'][1],
259                                       lab_data[i]['min'][2]),
260                                      (lab_data[i]['max'][0],
261                                       lab_data[i]['max'][1],
262                                       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 “lab_data[i]['min'][0]” is the lower limit of threshold.

The third parameter “lab_data[i]['max'][0]” is the upper limit of threshold.

- **Open and Close Operations**

```
263             opened = cv2.morphologyEx(frame_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))  # 开运算
264             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 the largest area contour**

After the above image processing is finished, it is required to obtain the contour the recognized target, which involves the findContours() function in the cv2 library.

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

Here will use an example of the 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 retrieval mode for contours.

The third parameter “**cv2.CHAIN_APPROX_NONE)[-2]**” specifies the contour approximation method.

It is necessary to set a minimum value to locate the contour with the largest area among all the obtained contours. Only when the area is greater than this value, the target contour is considered valid to avoid interference.

```
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
```

◆ **Evaluate the contour with the largest area**

```
314             if color_area_max == 'red':  #红色最大
315                 msg.data = 1
316             elif color_area_max == 'green':  #绿色最大
317                 msg.data = 2
318             elif color_area_max == 'blue':  #蓝色最大
319                 msg.data = 3
```

◆ **Multiple Evaluation**

Make multiple evaluations and take the average value to determine the recognized color.

```
287             if len(color_list) == 3:  # 多次判断
288                 # 取平均值
289                 color = int(round(np.mean(np.array(color_list))))
290                 color_list = []
291                 start_pick_up = True
292                 if color == 1:
293                     detect_color = 'red'
294                     draw_color = range_rgb["red"]
295                 elif color == 2:
296                     detect_color = 'green'
297                     draw_color = range_rgb["green"]
298                 elif color == 3:
299                     detect_color = 'blue'
300                     draw_color = range_rgb["blue"]
301                 else:
302                     detect_color = 'None'
303                     draw_color = range_rgb["black"]
304         else:
305             if not start_pick_up:
306                 draw_color = (0, 0, 0)
307                 detect_color = "None"
```

3.3.2 Color Recognition

◆ **Turn on RGB light and Buzzer**

```
169             if detect_color != 'None' and start_pick_up:  # 检测到色块
170
171                 set_rgb(detect_color) # 设置扩展板上的彩灯与检测到的颜色一样
172                 setBuzzer(0.1)      # 设置蜂鸣器响0.1秒
```

set_rgb(detect_color): The set_rgb() function is called to set the RGB on expansion board to the detected color.

setBuzzer(0.1): The setBuzzer() function is called to activate the buzzer for a duration of 0.1 seconds. It is used to control the audio effect and the duration of the buzzer.

Therefore, you can control the color lights on the expansion board based on the detected color and provide feedback through the sound emitted by the buzzer.

- **Control Servo**

```
174                 if detect_color == 'red' :  # 检测到红色,点头
175                     for i in range(0,3):
176                         Board.setPWMServoPulse(3, 800, 200)
177                         time.sleep(0.2)
178                         Board.setPWMServoPulse(3, 600, 200)
179                         time.sleep(0.2)
180                         if not __isRunning:
181                             continue
```

The Board.setPWMServoPulse function is used to control servo. The meaning of the parameters in parentheses is as follow:

The first parameter “3” is the ID number of the servo.

The second parameter “800” is the pulse width and its range is 500-2500.

The third parameter “200” is the duration of the action.

- **Back to the initial position**

```
198                 AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)  # 回到初始位置
199                 time.sleep(0.5)
200                 detect_color = 'None'
201                 start_pick_up = False
202                 set_rgb(detect_color)
```

The AK.setPitchRangeMoving function is used for inverse kinematics control of the robot’s pose. The parameters inside the parentheses have the following meanings:

The first parameter "(0, 6, 18)" represents the initial position coordinates.

The second parameter "0" is the given pitch angle.

The third parameter "-90" is the lower limit of the pitch angle range.

The fourth parameter "90" is the upper limit of the pitch angle range.

The fifth parameter "500" is the duration of the motion in milliseconds.