Chapter15: Color tracking (table tennis) with servo

In the previous course, we have learned how uses the PCA9685 and Raspberry Pi to control the servo and the color recognition. In this lesson, we will try to combine these two technologies.

The demand is that the Raspberry Pi camera will identify yellow table tennis in motion, the effect is to draw a circle on the contour of the table tennis in real time, at the same time, Raspberry Pi controls two servos to enable the camera to up, down, left and right directions.

The user move the table tennis up,down, left and right within the camera shooting range, and the camera can follow the table tennis.

We need to introduce the PID algorithm, which is an algorithm widely used in process control.

We will use the incremental PID algorithm, according to the actual needs, using PD for control, (use P (proportional) and D (differential) in the PID).

First, we need to connect the servo, BST_AI board and the Raspberry Pi, connect the servo that is control left and right movement of the camera (X axis) to the S1 of the PCA9685, and connect the servo that is control for the up and down movement of the servo (Y axis) to S2 of the PCA9685.

About code:

```
cap.set(3, 640)
cap.set(4, 480)
```

Set the resolution of the camera to 640*480 and initialize the servo.

```
pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(60)
pwm.set_pwm(1,0,400)
pwm.set_pwm(2,0,400)
```

The midpoint of the video window is (320, 240). When the small ball is recognized and the center of the drawn circle is found, we have actually found the center point of the table tennis, and set this point to (x, y). Then the distance between the table tennis ball and the center of the camera is (x-320, y-240).

Define two parameters (thisError_x=x-320, thisError_y=y-240), set the value of P in the PID to 3, set the value of D in the PID to 1, can obtain error values pwm_x and pwm_y, and the distance values thisError_x and thisError_y are recursive to lastError_x and lastError-y.

```
pwm_x = thisError_x*3+1*(thisError_x-lastError_x)
pwm_y = thisError_y*3+1*(thisError_y-lastError_y)
```

The third parameter pulse value of set_pwm() can only be an integer when controlling the servo,so we have to do the following code:

```
YP=pwm_y/100
Y_P=Y_P+int(YP)
pwm.set_pwm(0,0,Y_P)
```

We need to protect the servo, set the threshold for the rotation of the servo:

```
if X_P>670:
        X_P=650
if X_P<0:
        X_P=0
```

Note: In actual use, the direction of the x-axis movement of the ball on the servo is opposite to the actual direction used.

For example, the user moves table tennis to the left side of the camera, but the ball in the video frame actually moves to the right. So when dealing with x-axis servos, we should convert them to:

Pwm.set_pwm(1,0,650-X_P)

Users can modify P and D in the PID algorithm to get the best results.

The source code of the program is located at:

/home/pi/Adafruit_Python_PCA9685/servo_ball_nosocket.py

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file        sevo_ball_nosocket
    * @version      V1.0
    * @details
    * @par History
    @author: longfuSun
"""
from __future__ import division
import cv2
import Adafruit_PCA9685

import time
import numpy as np
import threading

#Initialize PCA9685 and servo
pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(60)
pwm.set_pwm(1,0,320)
pwm.set_pwm(2,0,240)
time.sleep(1)
#Initialize the camera and set the threshold
#If you think it is stuck, please adjust the "1" and "2" two codes
cap = cv2.VideoCapture(0)
#"1",Camera resolution, center point is (320, 240)
cap.set(3, 320)
cap.set(4, 240)
yellow_lower=np.array([0,43,46])
yellow_upper=np.array([10,255,255])
#4 variables for each degree of freedom
x=0;
thisError_x=500       #current error value
lastError_x=100       #Last error value
thisError_y=500
lastError_y=100
Y_P=425
X_P = 425
flag=0
```

```
y=0
def xx(X_P,Y_P):
    pwm.set_pwm(1,0,650-X_P)
    pwm.set_pwm(2,0,650-Y_P)
while True:
    ret,frame = cap.read()

    frame=cv2.GaussianBlur(frame,(5,5),0)
    hsv= cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
    #Mask and morphological operations to find the outline of the ball
    mask=cv2.inRange(hsv,yellow_lower,yellow_upper)
    mask=cv2.erode(mask,None,iterations=2)
    mask=cv2.dilate(mask,None,iterations=2)
    mask=cv2.GaussianBlur(mask,(3,3),0)
    res=cv2.bitwise_and(frame,frame,mask=mask)

    cnts=cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
    #When the ball is found
    if len(cnts)>0:
        #print('face found!')
        cnt=max(cnts,key=cv2.contourArea)
        (x,y),radius=cv2.minEnclosingCircle(cnt)
        radiusInt=int(int(radius)*1.5)
        #cv2.putText(frame,'佩奇',(int(x)-radiusInt,int(y)-radiusInt),cv2.FONT_HERSHEY_SIMPLEX,1.2,(255,255,255),2)
        cv2.rectangle(frame,(int(x)-radiusInt,int(y)-radiusInt),(int(x)+radiusInt,int(y)+radiusInt),(255,0,255),2)
        #"2",error value
        thisError_x=x-160
        thisError_y=y-120
        #PID
        pwm_x = thisError_x*3+1*(thisError_x-lastError_x)
        pwm_y = thisError_y*3+1*(thisError_y-lastError_y)

        lastError_x = thisError_x
        lastError_y = thisError_y
        #Adafruit_PCA9685 input integer
        XP=pwm_x/100
        YP=pwm_y/100
        X_P=X_P+int(XP)
        Y_P=Y_P+int(YP)
        #Limit the steering pulse of the servo to a safe range
        if X_P>670:
            X_P=650
        if X_P<0:
            X_P=0
        if Y_P>650:
            Y_P=650
        if X_P<0:
            Y_p=0
        print('x',x,X_P);
    tid=threading.Thread(target=xx,args=(X_P,Y_P,))
    tid.setDaemon(True)
    tid.start()
    #The servo rotate and cannot use function: pwm.set_pwm(1,0,X_P/Y_P)
    #pwm.set_pwm(1,0,650-X_P)
    #pwm.set_pwm(2,0,650-Y_P)

    cv2.imshow("capture", frame)
    if cv2.waitKey(1)==119:
        break

cap.release()
cv2.destroyAllWindows()
```