Chapter16: Raspberry pi face tracking

！ Note: When running the program of this course, there must be a desktop for displaying pictures. It is recommended that you use VNC to log in to the system so that the pictures can be displayed.

In the previous course, we have learned how realize face recognition, alarming and the color tracking with servo. In this lesson, we will combine several techniques to complete a face tracking program with servo control.
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.

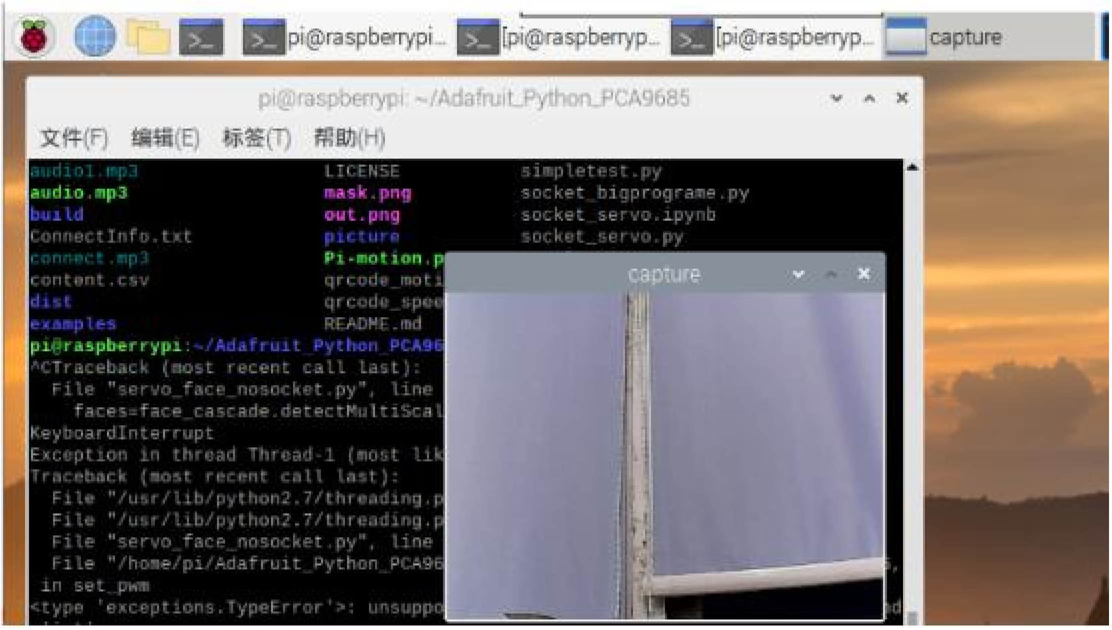
The camera PTZ is controlled by two servos. The PID technology makes the movement of the servos stable, and the masks in the color tracking, morphological processing and other commands are replaced to the recognition of the face.xml file.

The source code of the program is located at:
/home/pi/Adafruit_Python_PCA9685/servo_face_nosocket.py

Please use the following command line to run the program:

```
python3 servo_face_nosocket.py
```

The result is as shown in the figure below.

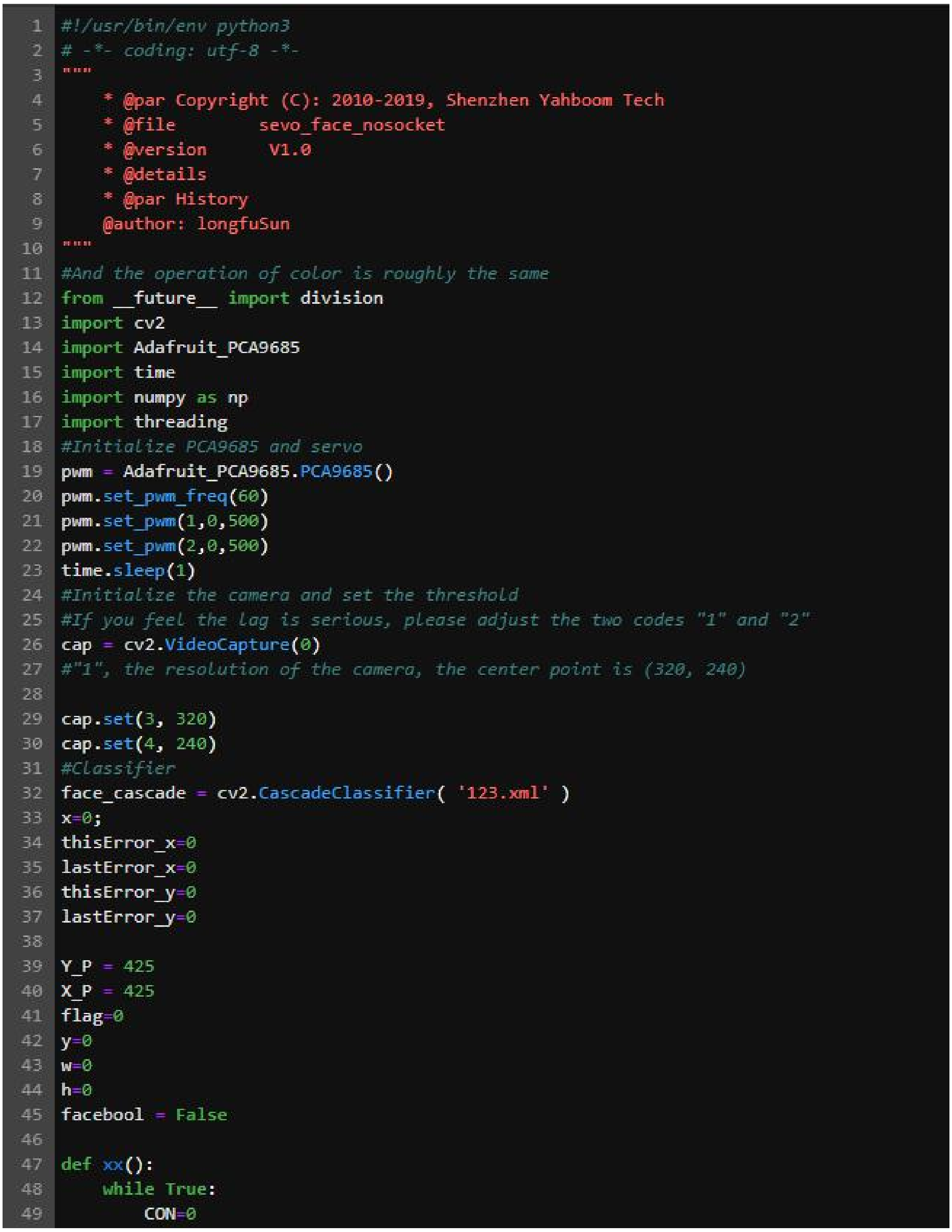


pi@raspberrypi...
[pi@raspberryp...
[pi@raspberryp...
capture
pi@raspberrypi: ~/Adafruit_Python_PCA9685
文件(F) 编辑(E) 标签(T) 帮助(H)
capture

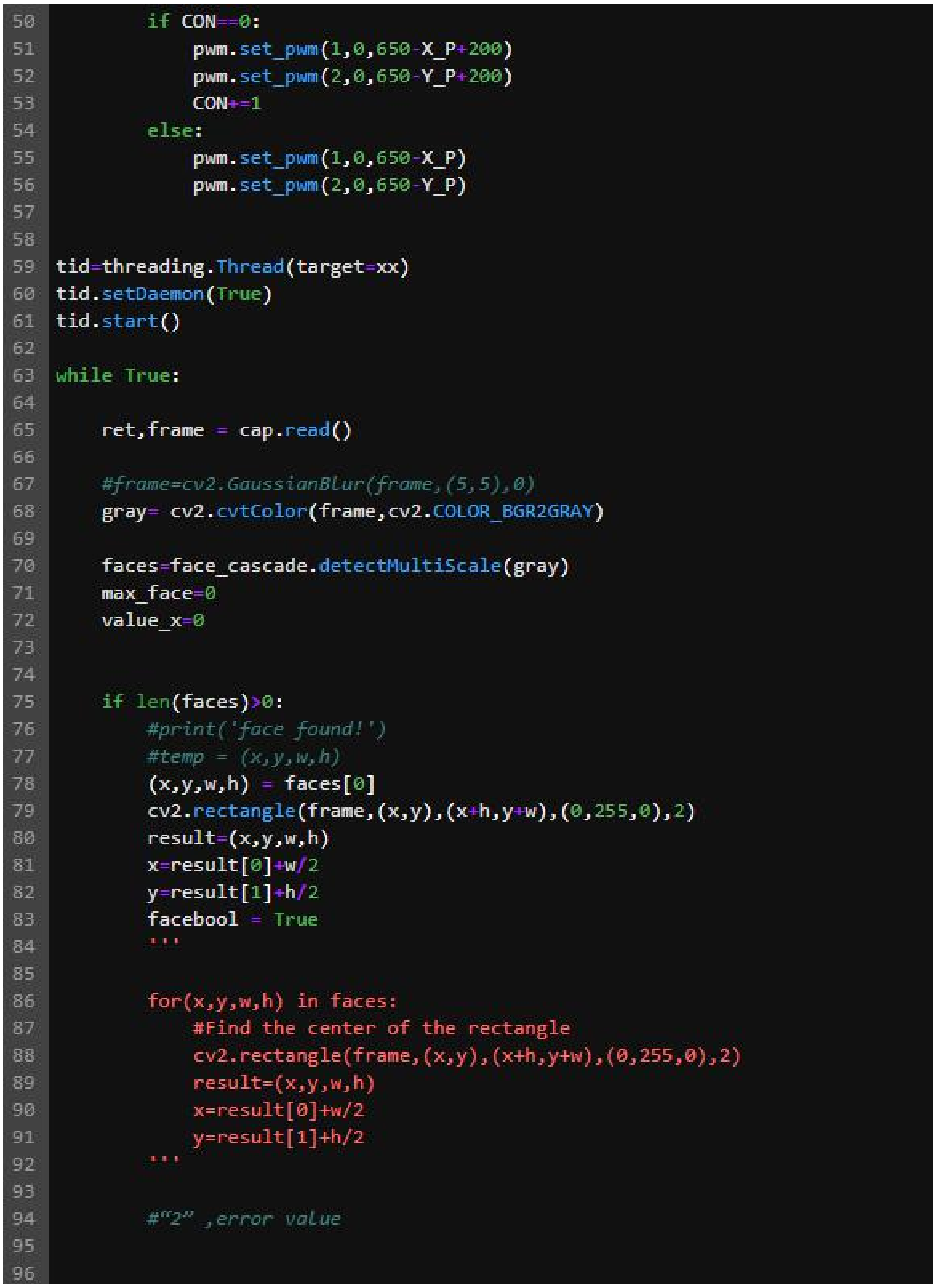
```
audio1.mp3                 LICENSE             simpletest.py
audio.mp3                  mask.png            socket_bigprograme.py
build                      out.png             socket_servo.ipynb
ConnectInfo.txt            picture             socket_servo.py
connect.mp3                Pi-motion.p
content.csv                qrcode_moti
dist                       qrcode_spee
examples                   README.md
pi@raspberrypi:~/Adafruit_Python_PCA96
^CTraceback (most recent call last):
  File "servo_face_nosocket.py", line
    faces=face_cascade.detectMultiScal
KeyboardInterrupt
Exception in thread Thread-1 (most lik
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.p
  File "/usr/lib/python2.7/threading.p
  File "servo_face_nosocket.py", line
  File "/home/pi/Adafruit_Python_PCA96
 in set_pwm
<type 'exceptions.TypeError'>: unsuppo
```

Finally, we can press Ctrl+C to end the process and close the output.

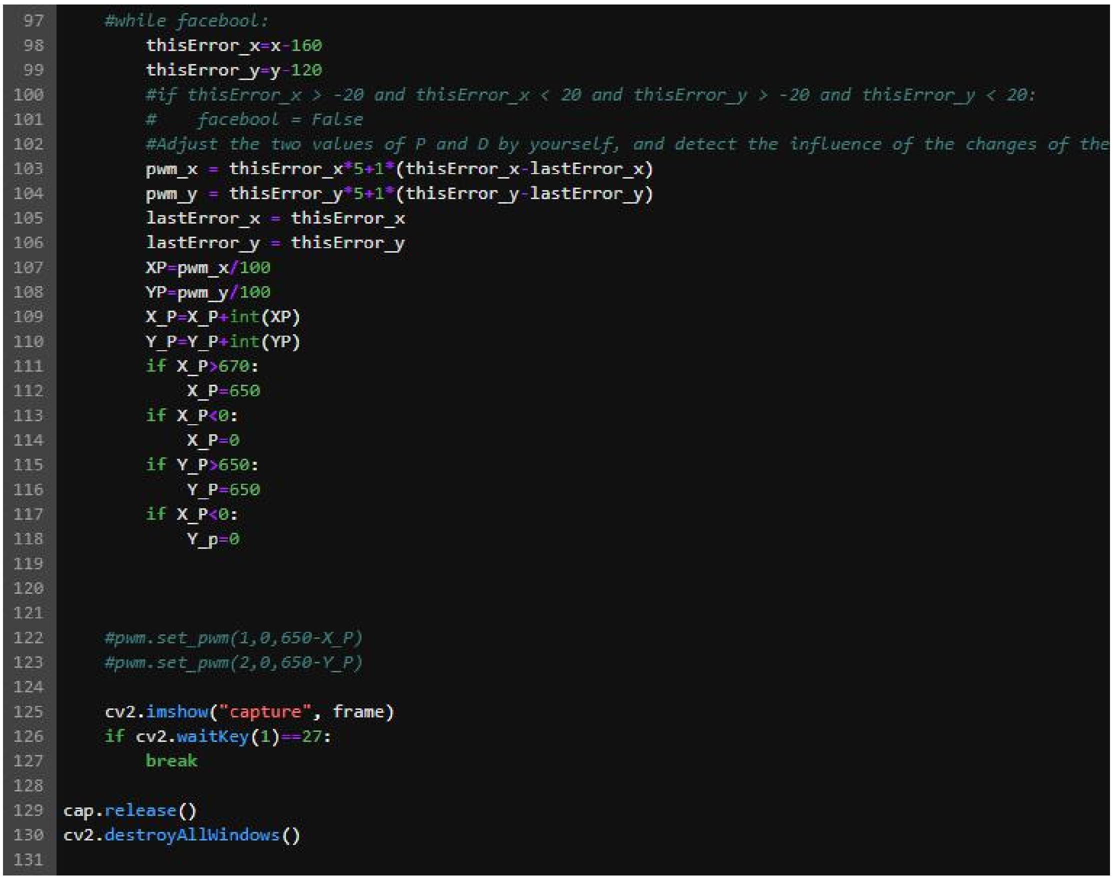
Code as shown below.



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file        sevo_face_nosocket
    * @version      V1.0
    * @details
    * @par History
    @author: longfuSun
"""
#And the operation of color is roughly the same
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(1,0,500)
pwm.set_pwm(2,0,500)
time.sleep(1)
#Initialize the camera and set the threshold
#If you feel the lag is serious, please adjust the two codes "1" and "2"
cap = cv2.VideoCapture(0)
#"1", the resolution of the camera, the center point is (320, 240)

cap.set(3, 320)
cap.set(4, 240)
#Classifier
face_cascade = cv2.CascadeClassifier( '123.xml' )
x=0;
thisError_x=0
lastError_x=0
thisError_y=0
lastError_y=0

Y_P = 425
X_P = 425
flag=0
y=0
w=0
h=0
facebool = False

def xx():
    while True:
        CON=0
```



```
        if CON==0:
            pwm.set_pwm(1,0,650-X_P+200)
            pwm.set_pwm(2,0,650-Y_P+200)
            CON+=1
        else:
            pwm.set_pwm(1,0,650-X_P)
            pwm.set_pwm(2,0,650-Y_P)


tid=threading.Thread(target=xx)
tid.setDaemon(True)
tid.start()

while True:

    ret,frame = cap.read()

    #frame=cv2.GaussianBlur(frame,(5,5),0)
    gray= cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    faces=face_cascade.detectMultiScale(gray)
    max_face=0
    value_x=0


    if len(faces)>0:
        #print('face found!')
        #temp = (x,y,w,h)
        (x,y,w,h) = faces[0]
        cv2.rectangle(frame,(x,y),(x+h,y+w),(0,255,0),2)
        result=(x,y,w,h)
        x=result[0]+w/2
        y=result[1]+h/2
        facebool = True
        '''

        for(x,y,w,h) in faces:
            #Find the center of the rectangle
            cv2.rectangle(frame,(x,y),(x+h,y+w),(0,255,0),2)
            result=(x,y,w,h)
            x=result[0]+w/2
            y=result[1]+h/2
        '''

        #“2” ,error value


```



```
    #while facebool:
        thisError_x=x-160
        thisError_y=y-120
        #if thisError_x > -20 and thisError_x < 20 and thisError_y > -20 and thisError_y < 20:
        #    facebool = False
        #Adjust the two values of P and D by yourself, and detect the influence of the changes of the
        pwm_x = thisError_x*5+1*(thisError_x-lastError_x)
        pwm_y = thisError_y*5+1*(thisError_y-lastError_y)
        lastError_x = thisError_x
        lastError_y = thisError_y
        XP=pwm_x/100
        YP=pwm_y/100
        X_P=X_P+int(XP)
        Y_P=Y_P+int(YP)
        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



    #pwm.set_pwm(1,0,650-X_P)
    #pwm.set_pwm(2,0,650-Y_P)

    cv2.imshow("capture", frame)
    if cv2.waitKey(1)==27:
        break

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