3.1.8 Face tracking

In this course, we will complete face tracking, which will use PID.
Code path:
/home/pi/Yahboom_Project/2.AI_Visual_course/08.Face_tracking/Face_tracking.ipynb

```
#bgr8 to jpeg format
import enum
import cv2

def bgr8_to_jpeg(value, quality=75):
        return bytes(cv2.imencode('.jpg', value)[1])
```

## Import related packages and create camera instances

```
import cv2
import ipywidgets.widgets as widgets
import threading
import time
import sys

image_widget = widgets.Image(format='jpeg', width=320, height=240)
display(image_widget)
```

# Add PID slider to adjust PID value

```
import ipywidgets as widgets

XServo_P = widgets.FloatSlider(
value=1.1,
min=0,
max=10.0,
step=0.1,
description='XServo-P:',
disabled=False,
continuous_update=False,
orientation='horizontal',
readout=True,
readout_format='.1f',
)

XServo_I = widgets.FloatSlider(
value=0.2,
min=0,
max=10.0,
step=0.1,
description='XServo-I:',
disabled=False,
continuous_update=False,
orientation='horizontal',
readout=True,
readout_format='.1f',
)

XServo_D = widgets.FloatSlider(
value=0.8,
min=0,
max=10.0,
step=0.1,
description='XServer-D:',
disabled=False,
continuous_update=False,
orientation='horizontal',
readout=True,
readout_format='.1f',
)

YServo_P = widgets.FloatSlider(
value=0.8,
min=0,
max=10.0,
step=0.1,
description='YServo-P:',
disabled=False,
continuous_update=False,
orientation='horizontal',
readout=True,
readout_format='.1f',
)

YServo_I = widgets.FloatSlider(
value=0.2,
min=0,
max=10.0,
step=0.1,
description='YServo-I:',
disabled=False,
continuous_update=False,
orientation='horizontal',
readout=True,
readout_format='.1f',
)
```

```
YServo_D = widgets.FloatSlider(
value=0.8,
min=0,
max=10.0,
step=0.1,
description='YServer-D:',
disabled=False,
continuous_update=False,
orientation='horizontal',
readout=True,
readout_format='.1f',
)
display(XServo_P, XServo_I, XServo_D, YServo_P, YServo_I, YServo_D)
```

## Create related control variables

```
global face_x, face_y, face_w, face_h
face_x = face_y = face_w = face_h = 0
global target_valuex
target_valuex = 2048
global target_valuey
target_valuey = 2048
```

## Create a PID control instance

```
import PID

# xservo_pid = PID.PositionalPID(1.1, 0.2, 0.8)
# yservo_pid = PID.PositionalPID(0.8, 0.2, 0.8)

xservo_pid = PID.PositionalPID(XServo_P.value, XServo_I.value, XServo_D.value)
yservo_pid = PID.PositionalPID(YServo_P.value, YServo_I.value, YServo_D.value)
```

# Import Raspblock drive board function

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# define servo pin
ServoPin = 11  #S2
ServoPinB = 9  #S3
```

```
# Set the servo pin to output mode
def init():
    GPIO.setup(ServoPin, GPIO.OUT)
    GPIO.setup(ServoPinB, GPIO.OUT)
```

```
#Define a pulse function, used to simulate the pwm value
#Time base pulse is 20ms, the high level part of the pulse is controlled from 0 to 180 degrees in 0.5-2.5ms
```

```
def servo_pulse(myangleA, myangleB):
    pulsewidth = myangleA
    GPIO.output(ServoPin, GPIO.HIGH)
    time.sleep(pulsewidth/1000000.0)
    GPIO.output(ServoPin, GPIO.LOW)
    time.sleep(20.0/1000-pulsewidth/1000000.0)

    pulsewidthB = myangleB
    GPIO.output(ServoPinB, GPIO.HIGH)
    time.sleep(pulsewidthB/1000000.0)
    GPIO.output(ServoPinB, GPIO.LOW)
    time.sleep(20.0/1000-pulsewidthB/1000000.0)
```

```
#According to the steering gear pulse control range is 500-2500usec
def Servo_control(angle_1, angle_2):
    init()
    if angle_1 < 500:
        angle_1 = 500
    elif angle_1 > 2500:
        angle_1 = 2500

    if angle_2 < 500:
        angle_2 = 500
    elif angle_2 > 2500:
        angle_2 = 2500
    servo_pulse(angle_1, angle_2)
```

## Load the "Haar"" cascade classifier

```
import cv2
face_haar = cv2.CascadeClassifier('haarcascade_profileface.xml')
```

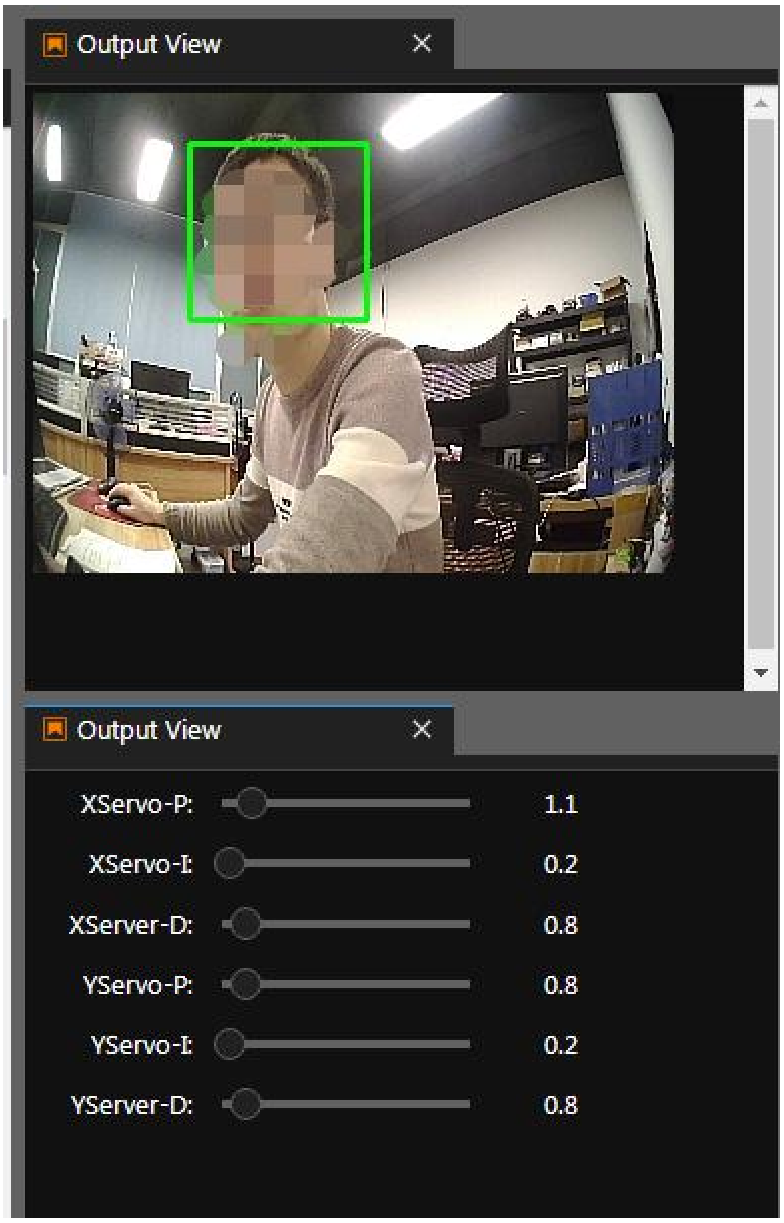
# Open camera

```
image = cv2.VideoCapture(0)
image.set(3,320)
image.set(4,240)
image.set(5, 120)    #Set frame rate
image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
image.set(cv2.CAP_PROP_BRIGHTNESS, 40) #set brightness -64 - 64    0.0
image.set(cv2.CAP_PROP_CONTRAST, 50)      #set contrast -64 - 64    2.0
image.set(cv2.CAP_PROP_EXPOSURE, 156)    #set exposure 1.0 - 5000    156.0
```

## The main process of the head movement

```
while 1:
        ret, frame = image.read()
        try:
                image_widget.value = bgr8_to_jpeg(frame)
        except:
                continue
        # Convert the image to black-white image
        gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_haar.detectMultiScale(gray_img, 1.1, 3)

        xservo_pid = PID.PositionalPID(XServo_P.value, XServo_I.value, XServo_D.value)
        yservo_pid = PID.PositionalPID(YServo_P.value, YServo_I.value, YServo_D.value)

        if len(faces) > 0:
            (face_x, face_y, face_w, face_h) = faces[0]

#cv2.rectangle(frame,(face_x+10,face_y),(face_x+face_w-10,face_y+face_h+20),(0,25
5,0),2)

cv2.rectangle(frame,(face_x,face_y),(face_x+face_w,face_y+face_h),(0,255,0),2)
            try:
                image_widget.value = bgr8_to_jpeg(frame)
            except:
                continue

            #Proportion-Integration-Differentiation Arithmetic
            # Input X axis direction parameter PID control input
            xservo_pid.SystemOutput = face_x + face_w/2
            xservo_pid.SetStepSignal(150)
            xservo_pid.SetInertiaTime(0.01, 0.1)
            target_valuex = int(1500 + xservo_pid.SystemOutput)

            # Input Y axis direction parameter PID control input
            yservo_pid.SystemOutput = face_y + face_h/2
            yservo_pid.SetStepSignal(120)
            yservo_pid.SetInertiaTime(0.01, 0.1)
            target_valuey = int(1500 - yservo_pid.SystemOutput)

            # Rotate the camera pan to the PID adjustment and calibration position
            robot.Servo_control(target_valuex,target_valuey)
```

After run above program, we can see following picture, the face in the camera screen will be detected.



XServo-P: 1.1

XServo-I: 0.2

XServer-D: 0.8

YServo-P: 0.8

YServo-I: 0.2

YServer-D: 0.8