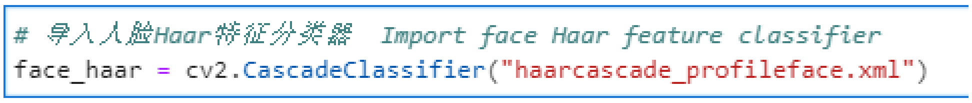
Face tracking

1. Introduction

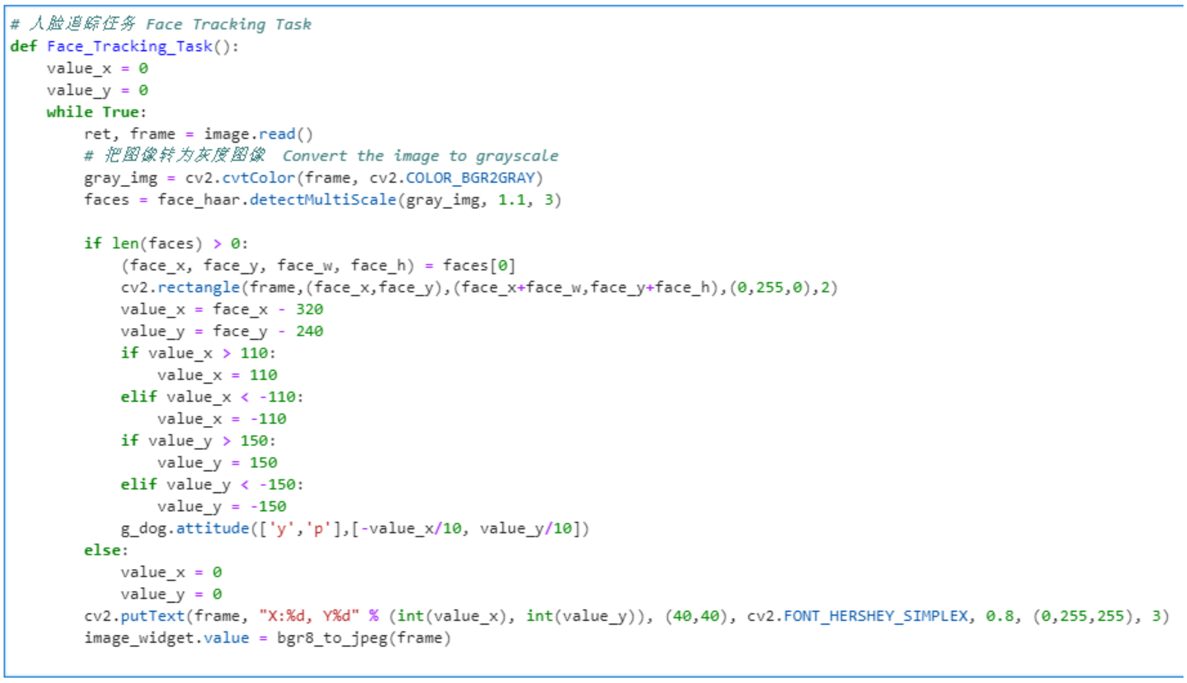
This course mainly uses the camera of Raspberry Pi to obtain the images of the camera. The opencv library imports the Haar feature classifier of the face to analyze the position of the face in the image, frame the face, adjust the body posture of the robot dog, and move with the image within the detection range.

2. Code analysis

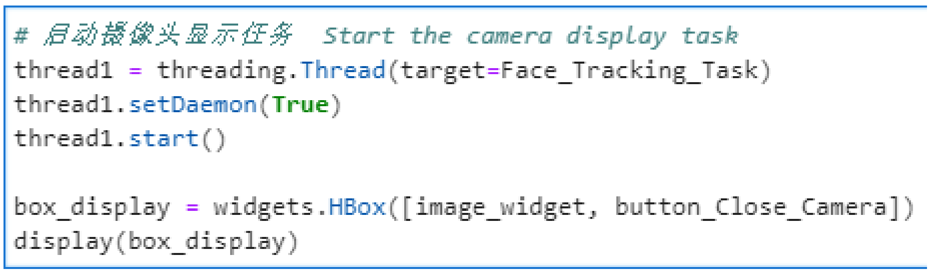
Importing face Haar feature classifier


```
# 导入人脸Haar特征分类器  Import face Haar feature classifier
face_haar = cv2.CascadeClassifier("haarcascade_profileface.xml")
```

Face detection task: firstly, the camera image is grayed, and then the image is sent to Haar face detection feature classifier for processing. The face is detected and the data of the face position is output. Then, a box is drawn for the face in the image, and then the posture of the robot dog is adjusted according to the position of the box to achieve the effect of moving with the image.


```
# 人脸追踪任务 Face Tracking Task
def Face_Tracking_Task():
    value_x = 0
    value_y = 0
    while True:
        ret, frame = image.read()
        # 把图像转为灰度图像  Convert the image to grayscale
        gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_haar.detectMultiScale(gray_img, 1.1, 3)

        if len(faces) > 0:
            (face_x, face_y, face_w, face_h) = faces[0]
            cv2.rectangle(frame,(face_x,face_y),(face_x+face_w,face_y+face_h),(0,255,0),2)
            value_x = face_x - 320
            value_y = face_y - 240
            if value_x > 110:
                value_x = 110
            elif value_x < -110:
                value_x = -110
            if value_y > 150:
                value_y = 150
            elif value_y < -150:
                value_y = -150
            g_dog.attitude(['y','p'],[-value_x/10, value_y/10])
        else:
            value_x = 0
            value_y = 0
        cv2.putText(frame, "X:%d, Y%d" % (int(value_x), int(value_y)), (40,40), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        image_widget.value = bgr8_to_jpeg(frame)
```

Start a daemon thread, run the camera identification task, and display the camera image.



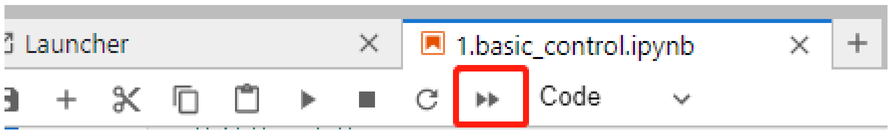
```
# 启动摄像头显示任务  Start the camera display task
thread1 = threading.Thread(target=Face_Tracking_Task)
thread1.setDaemon(True)
thread1.start()

box_display = widgets.HBox([image_widget, button_Close_Camera])
display(box_display)
```

1.3 Steps

Open the jupyterLab client and find following code path

```
DOGZILLA/Samples/3_AI_Visual/5.face_tracking.ipynb
```

Click the following icon to run all cells, and then pull to the bottom to see the generated controls.



Please place the robot dog in a pure color environment of the camera image, otherwise it may cause false recognition due to color interference.

The left side shows the camera image. Move the face on the camera, the camera will draw a box for the face, and the robot dog will change its posture as the face moves.

Finally, click close_camera button turns off the camera.