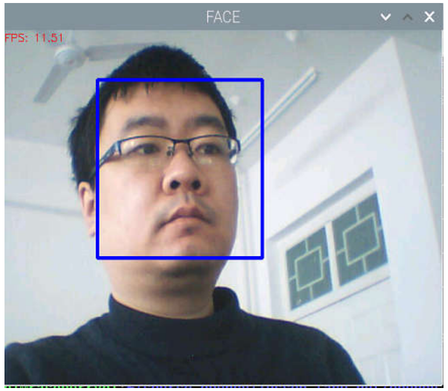
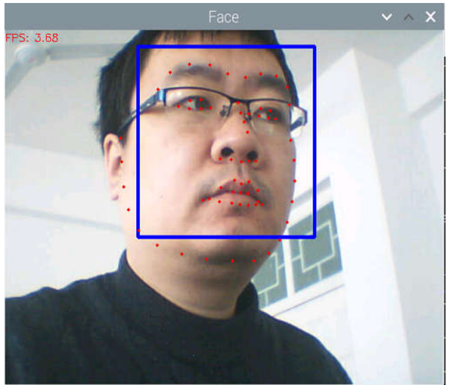
人工智能学院

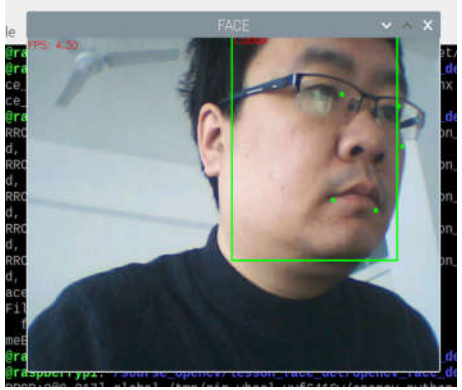
鲁東大學 LUDONG UNIVERSITY

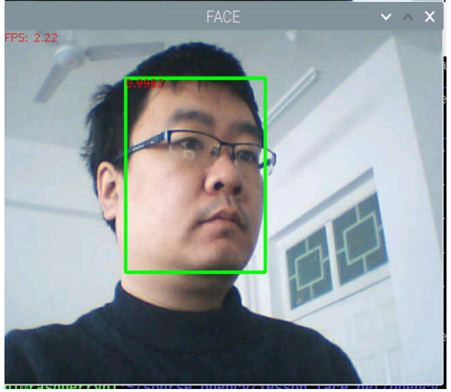
智能系统与控制

树莓派：OpenCV 人脸检测









于泓
鲁东大学
信息与电气工程学院

2022.2.15

人脸识别指识别并理解一张脸。人脸识别是一项热门的计算机技术研究领域，它属于生物特征识别技术，是对生物体（一般特指人）本身的生物特征来区分生物体个体。人脸识别技术在安防监控、身份认证等众多领域都有着重要的作用。人脸识别技术发展历史悠久并且有着很多成熟的算法和落地的应用。

人脸检测作为人脸识别系统的前置工作在人脸识别系统中有着重要的作用

(1) OpenCV 中采用的是基于 Haar 特征的级联分类器（cascade classifiers）实现的人脸检测能。

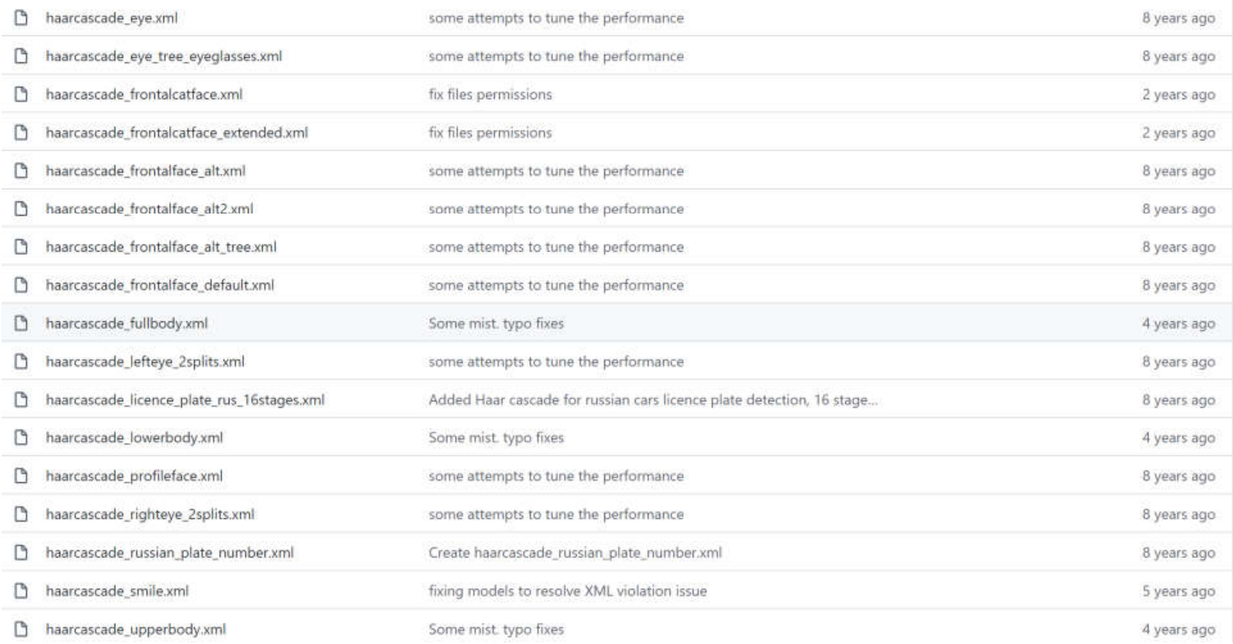
(2) OpenCV 中利用tensorflow训练的SSD（目标检测器）实现人脸检测。

(3) OpenCV 中利用YuNet（密集采样的CNN）实现人脸检测及关键点检测。

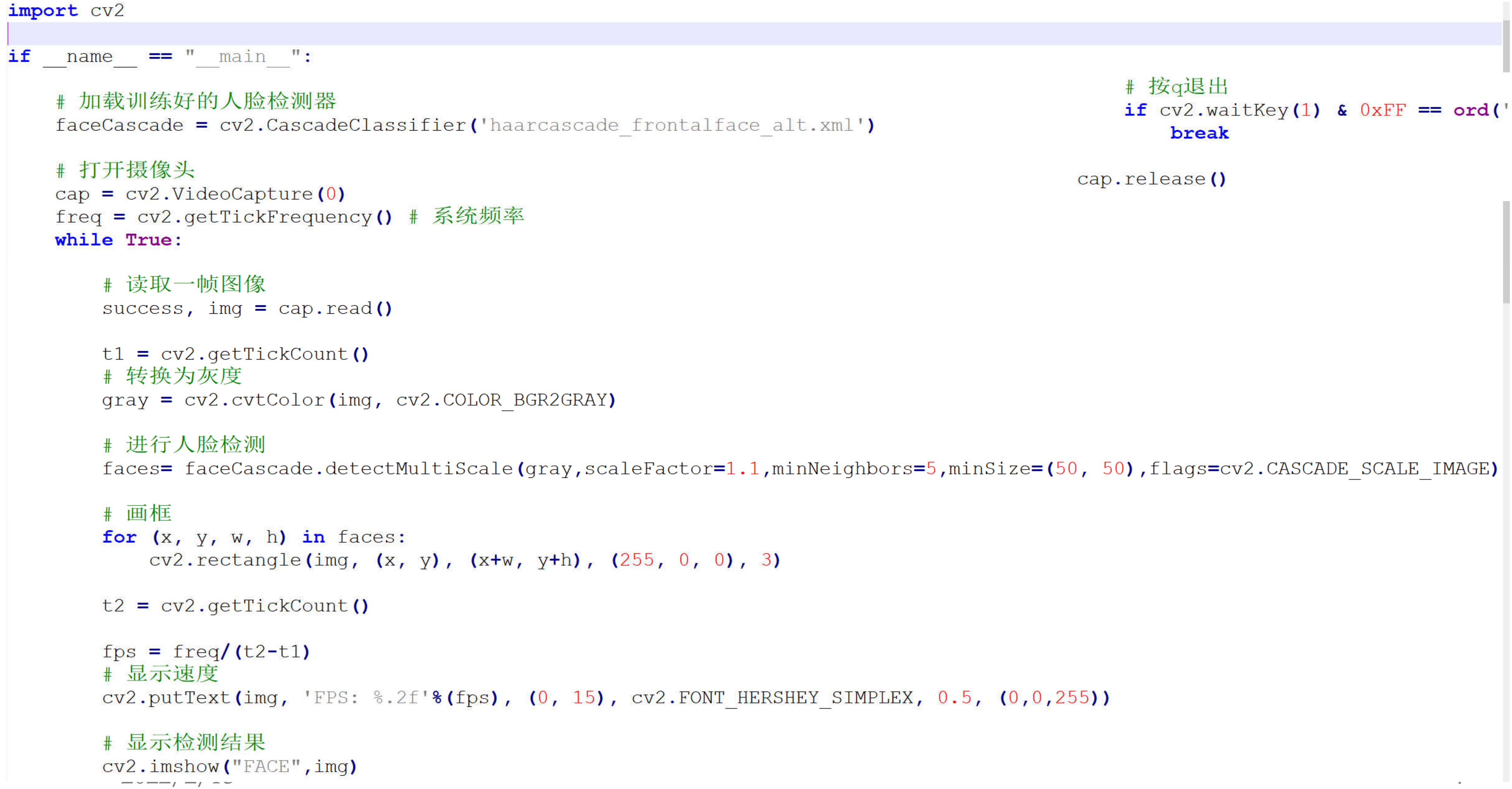
(4) 结合OpenCV 中利用Dlib实现人脸检测及关键点检测。

在 OpenCV 中采用的是基于 Haar 特征的级联分类器（cascade classifiers）实现的人脸检测功能。该算法于 2001 年由 Paul Viola 与 Michael Jones 在论文”Rapid Object Detection using a Boosted Cascade of Simple Features” 中首次提出。该算法通过将大量简单的基于 Haar 特征二分类器进行级联实现实现人脸/非人脸的检测。在 OpenCV 中提供了很多已经训练好的人脸检测器供我们直接下载使用

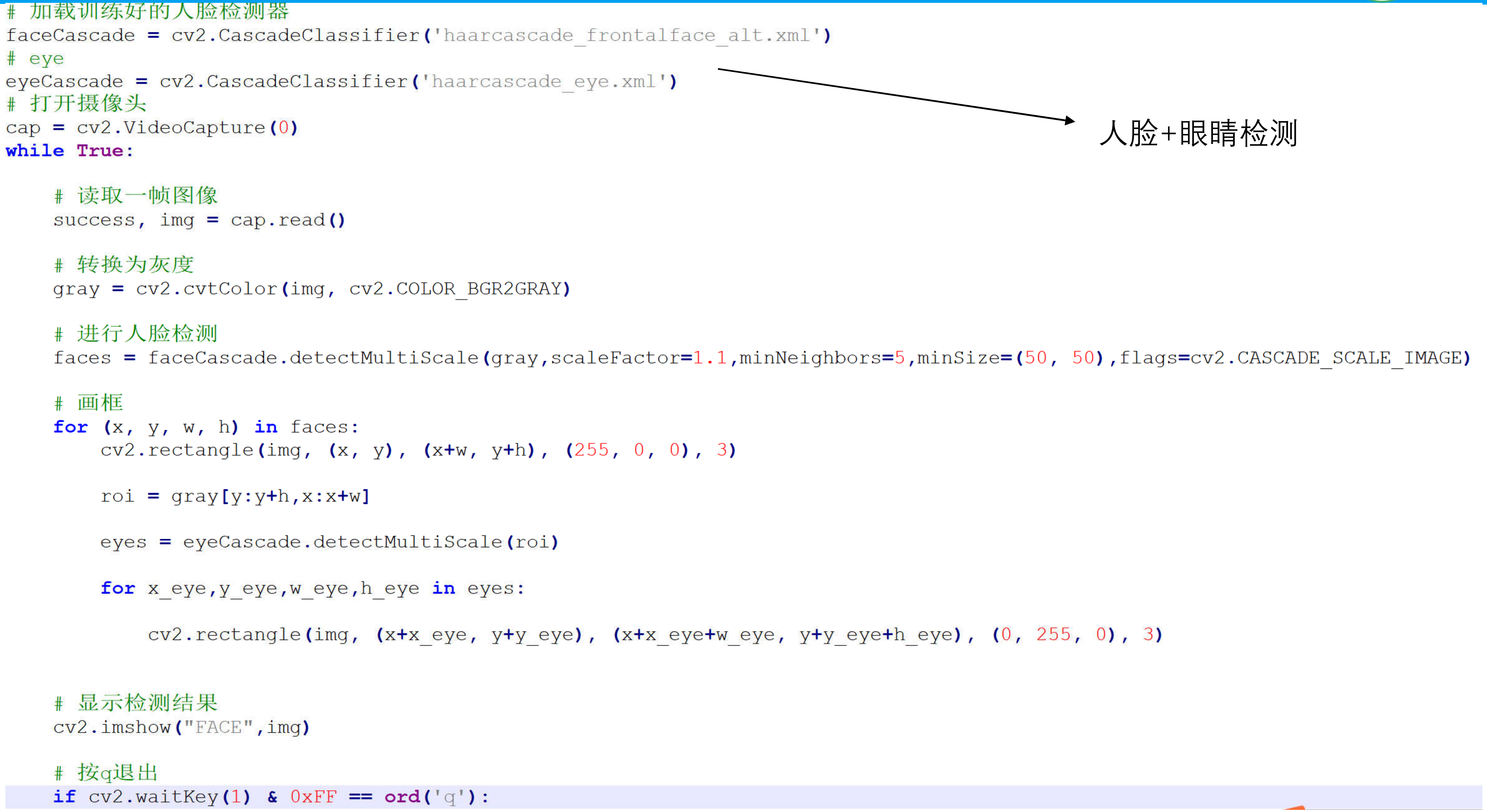
https://github.com/opencv/opencv/tree/master/data/haarcascades



haarcascade_eye.xml	some attempts to tune the performance	8 years ago
haarcascade_eye_tree_eyeglasses.xml	some attempts to tune the performance	8 years ago
haarcascade_frontalcatface.xml	fix files permissions	2 years ago
haarcascade_frontalcatface_extended.xml	fix files permissions	2 years ago
haarcascade_frontalface_alt.xml	some attempts to tune the performance	8 years ago
haarcascade_frontalface_alt2.xml	some attempts to tune the performance	8 years ago
haarcascade_frontalface_alt_tree.xml	some attempts to tune the performance	8 years ago
haarcascade_frontalface_default.xml	some attempts to tune the performance	8 years ago
haarcascade_fullbody.xml	Some mist. typo fixes	4 years ago
haarcascade_lefteye_2splits.xml	some attempts to tune the performance	8 years ago
haarcascade_licence_plate_rus_16stages.xml	Added Haar cascade for russian cars licence plate detection, 16 stage...	8 years ago
haarcascade_lowerbody.xml	Some mist. typo fixes	4 years ago
haarcascade_profileface.xml	some attempts to tune the performance	8 years ago
haarcascade_righteye_2splits.xml	some attempts to tune the performance	8 years ago
haarcascade_russian_plate_number.xml	Create haarcascade_russian_plate_number.xml	8 years ago
haarcascade_smile.xml	fixing models to resolve XML violation issue	5 years ago
haarcascade_upperbody.xml	Some mist. typo fixes	4 years ago



```
import cv2

if __name__ == "__main__":

    # 加载训练好的人脸检测器
    faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

    # 打开摄像头
    cap = cv2.VideoCapture(0)
    freq = cv2.getTickFrequency() # 系统频率
    while True:

        # 读取一帧图像
        success, img = cap.read()

        t1 = cv2.getTickCount()
        # 转换为灰度
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # 进行人脸检测
        faces= faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(50, 50),flags=cv2.CASCADE_SCALE_IMAGE)

        # 画框
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 3)

        t2 = cv2.getTickCount()

        fps = freq/(t2-t1)
        # 显示速度
        cv2.putText(img, 'FPS: %.2f'%(fps), (0, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255))

        # 显示检测结果
        cv2.imshow("FACE",img)

        # 按q退出
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
```



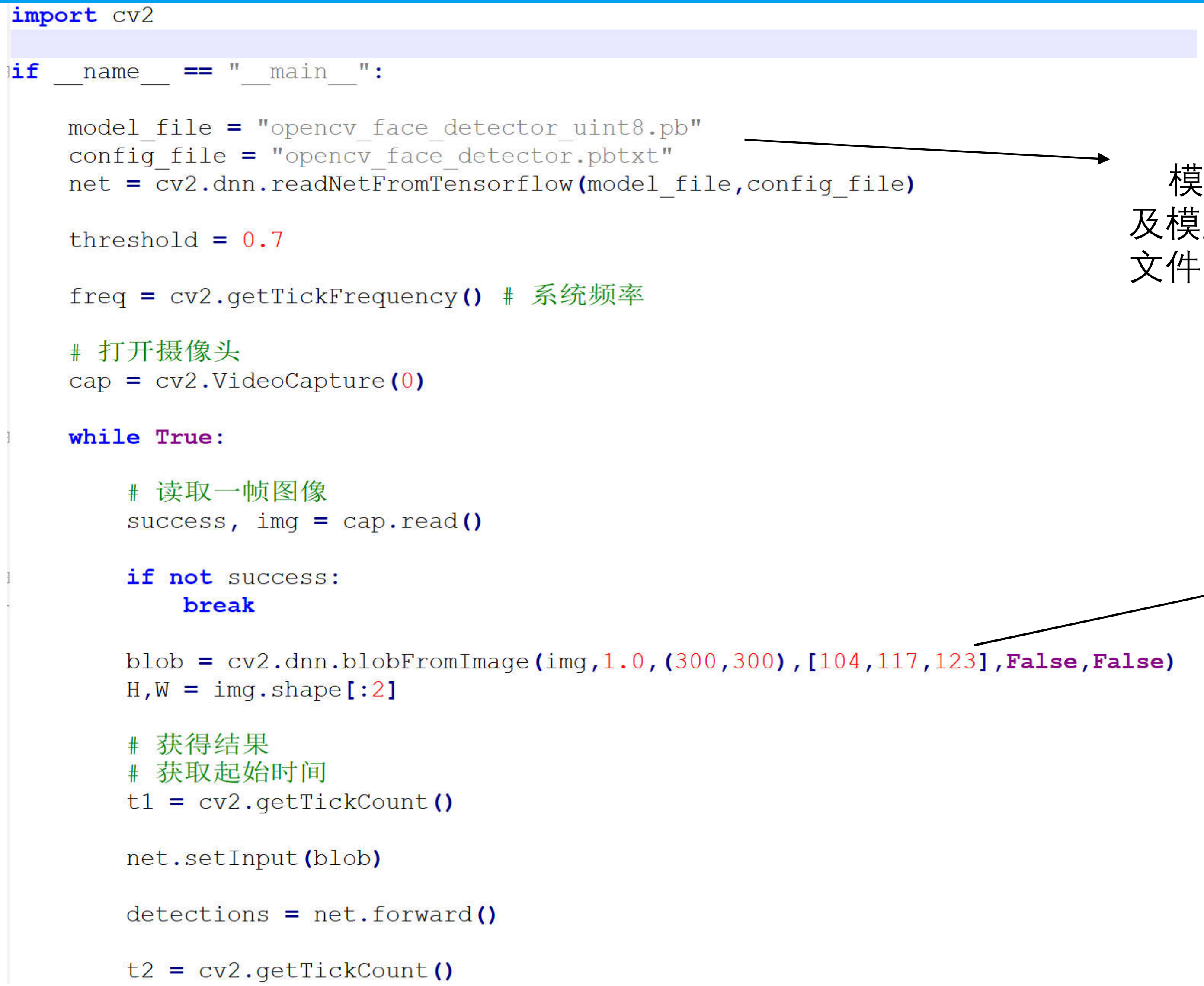
```
# 加载训练好的人脸检测器
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
# eye
eyeCascade = cv2.CascadeClassifier('haarcascade_eye.xml')
# 打开摄像头
cap = cv2.VideoCapture(0)
while True:

    # 读取一帧图像
    success, img = cap.read()

    # 转换为灰度
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # 进行人脸检测
    faces = faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(50, 50),flags=cv2.CASCADE_SCALE_IMAGE)

    # 画框
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 3)

        roi = gray[y:y+h,x:x+w]

        eyes = eyeCascade.detectMultiScale(roi)

        for x_eye,y_eye,w_eye,h_eye in eyes:

            cv2.rectangle(img, (x+x_eye, y+y_eye), (x+x_eye+w_eye, y+y_eye+h_eye), (0, 255, 0), 3)


    # 显示检测结果
    cv2.imshow("FACE",img)

    # 按q退出
    if cv2.waitKey(1) & 0xFF == ord('q'):
```

人脸+眼睛检测

利用目标检测算法SSD实现人脸识别

（基于深度神经网络的方法）

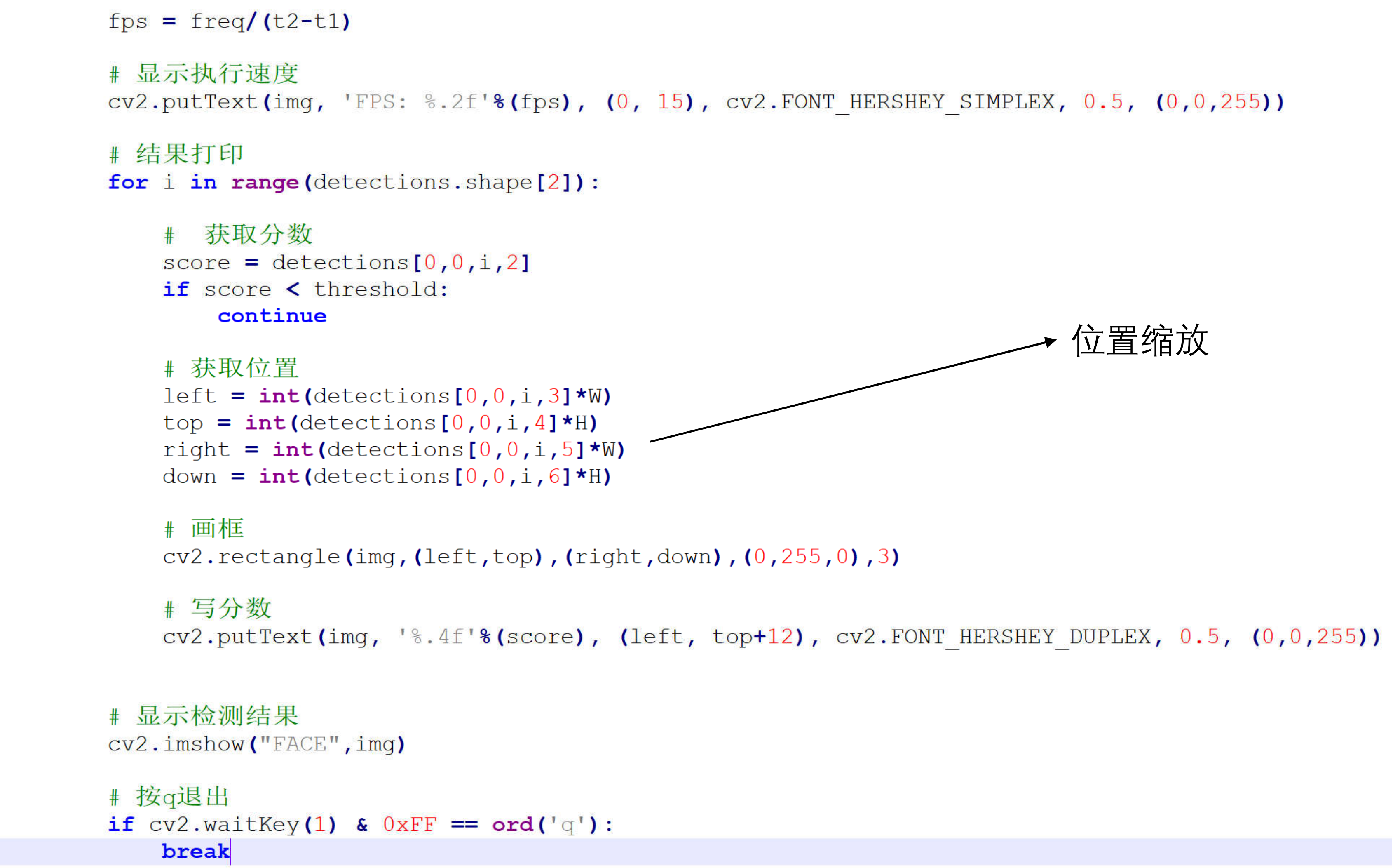
预测人脸边框



```python
import cv2

if __name__ == "__main__":

    model_file = "opencv_face_detector_uint8.pb"
    config_file = "opencv_face_detector.pbtxt"
    net = cv2.dnn.readNetFromTensorflow(model_file,config_file)

    threshold = 0.7

    freq = cv2.getTickFrequency() # 系统频率

    # 打开摄像头
    cap = cv2.VideoCapture(0)

    while True:

        # 读取一帧图像
        success, img = cap.read()

        if not success:
            break

        blob = cv2.dnn.blobFromImage(img,1.0,(300,300),[104,117,123],False,False)
        H,W = img.shape[:2]

        # 获得结果
        # 获取起始时间
        t1 = cv2.getTickCount()

        net.setInput(blob)

        detections = net.forward()

        t2 = cv2.getTickCount()
```

模型
及模型描述
文件

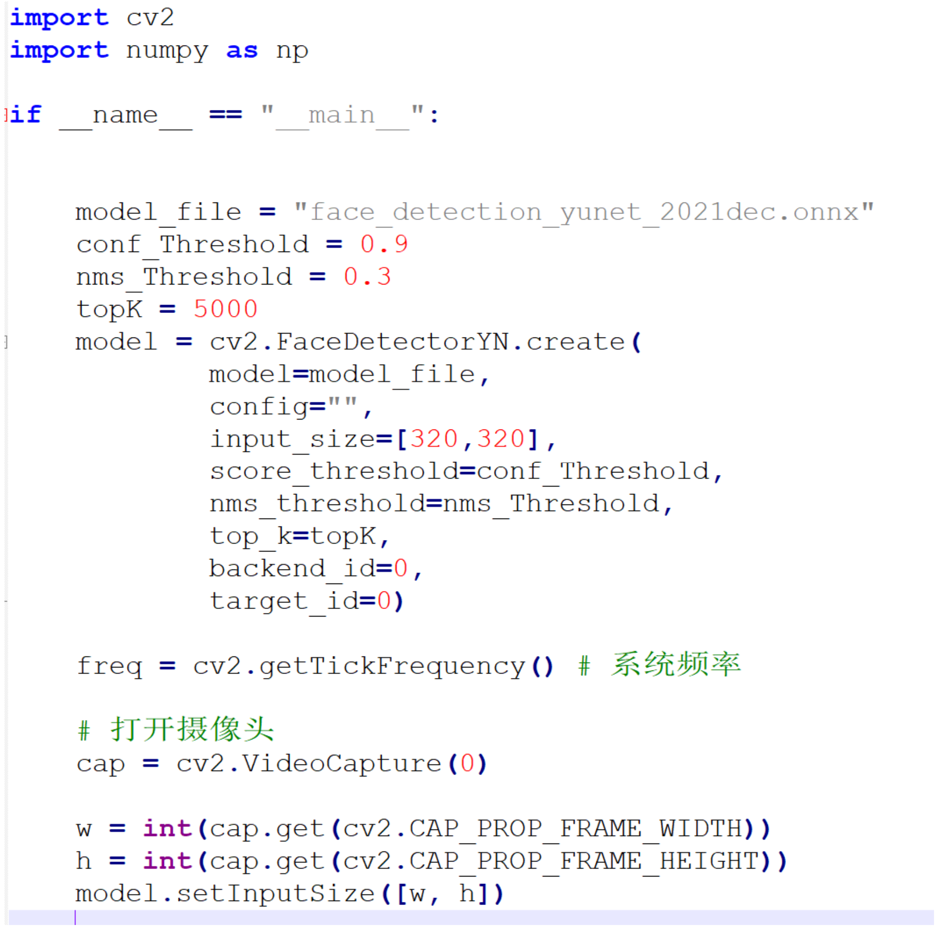
图像预处理



```
fps = freq/(t2-t1)

# 显示执行速度
cv2.putText(img, 'FPS: %.2f'%(fps), (0, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255))

# 结果打印
for i in range(detections.shape[2]):

    #  获取分数
    score = detections[0,0,i,2]
    if score < threshold:
        continue

    # 获取位置
    left = int(detections[0,0,i,3]*W)
    top = int(detections[0,0,i,4]*H)
    right = int(detections[0,0,i,5]*W)
    down = int(detections[0,0,i,6]*H)

    # 画框
    cv2.rectangle(img,(left,top),(right,down),(0,255,0),3)

    # 写分数
    cv2.putText(img, '%.4f'%(score), (left, top+12), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,0,255))


# 显示检测结果
cv2.imshow("FACE",img)

# 按q退出
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

位置缩放

YuNet　　密集采样　　**预测像素点到边框的相对位置**



```
import cv2
import numpy as np

if __name__ == "__main__":


    model_file = "face_detection_yunet_2021dec.onnx"
    conf_Threshold = 0.9
    nms_Threshold = 0.3
    topK = 5000
    model = cv2.FaceDetectorYN.create(
            model=model_file,
            config="",
            input_size=[320,320],
            score_threshold=conf_Threshold,
            nms_threshold=nms_Threshold,
            top_k=topK,
            backend_id=0,
            target_id=0)

    freq = cv2.getTickFrequency() # 系统频率

    # 打开摄像头
    cap = cv2.VideoCapture(0)

    w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    model.setInputSize([w, h])
```

```
# 打开摄像头
cap = cv2.VideoCapture(0)

w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
model.setInputSize([w, h])

while True:

    # 读取一帧图像
    success, img = cap.read()

    if not success:
        break

    # 获得结果
    t1 = cv2.getTickCount()

    faces = model.detect(img)
    results = faces[1]

    t2 = cv2.getTickCount()

    fps = freq/(t2-t1)

  # 显示速度
    cv2.putText(img, 'FPS: %.2f'%(fps), (0, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255))
```

```
# 绘图
for det in (results if results is not None else []):

    # 获得检测区域
    x,y,w,h = det[0:4].astype(np.int32)
    cv2.rectangle(img, (x, y), (x+w, y+h), (0,255,0), 2)

    #  获得分数
    score = det[-1]
    cv2.putText(img, '%.4f'%(score), (x, y+12), cv2.FONT_HERSHEY_DUPLEX, 0.5, (0,0,255))

    # 显示 关键点

    landmarks = det[4:14].astype(np.int32).reshape((5,2))

    for idx, landmark in enumerate(landmarks):
        cv2.circle(img, landmark, 2, (0,255,0), 2)


 # 显示检测结果
cv2.imshow("FACE",img)

# 按q退出
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

基于Dlib的人脸检测

Dlib 本身是一个用 C++ 编写的机器学习和信号处理相关的开源工具包，里面包含了很多人工智能相关理论的实现方，人脸检测与识别是其中的一个子功能。利用其提供的预训练模型，除了可以检测人脸位置外，还可以对人脸中的 68 个标志点（landmarks）进行检测。

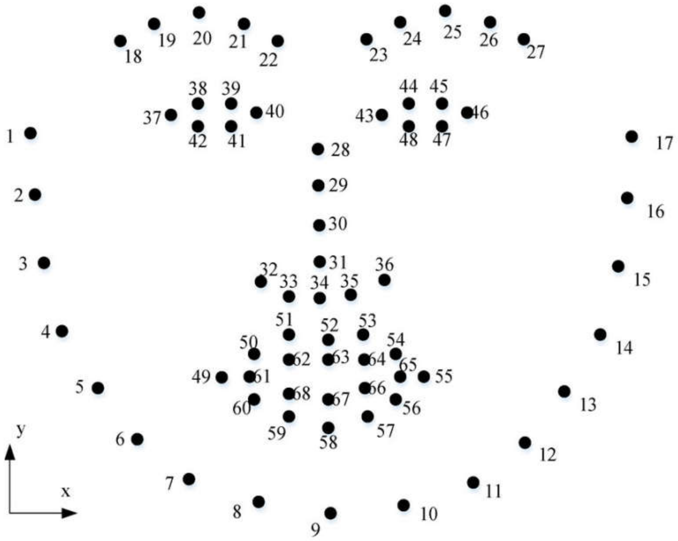


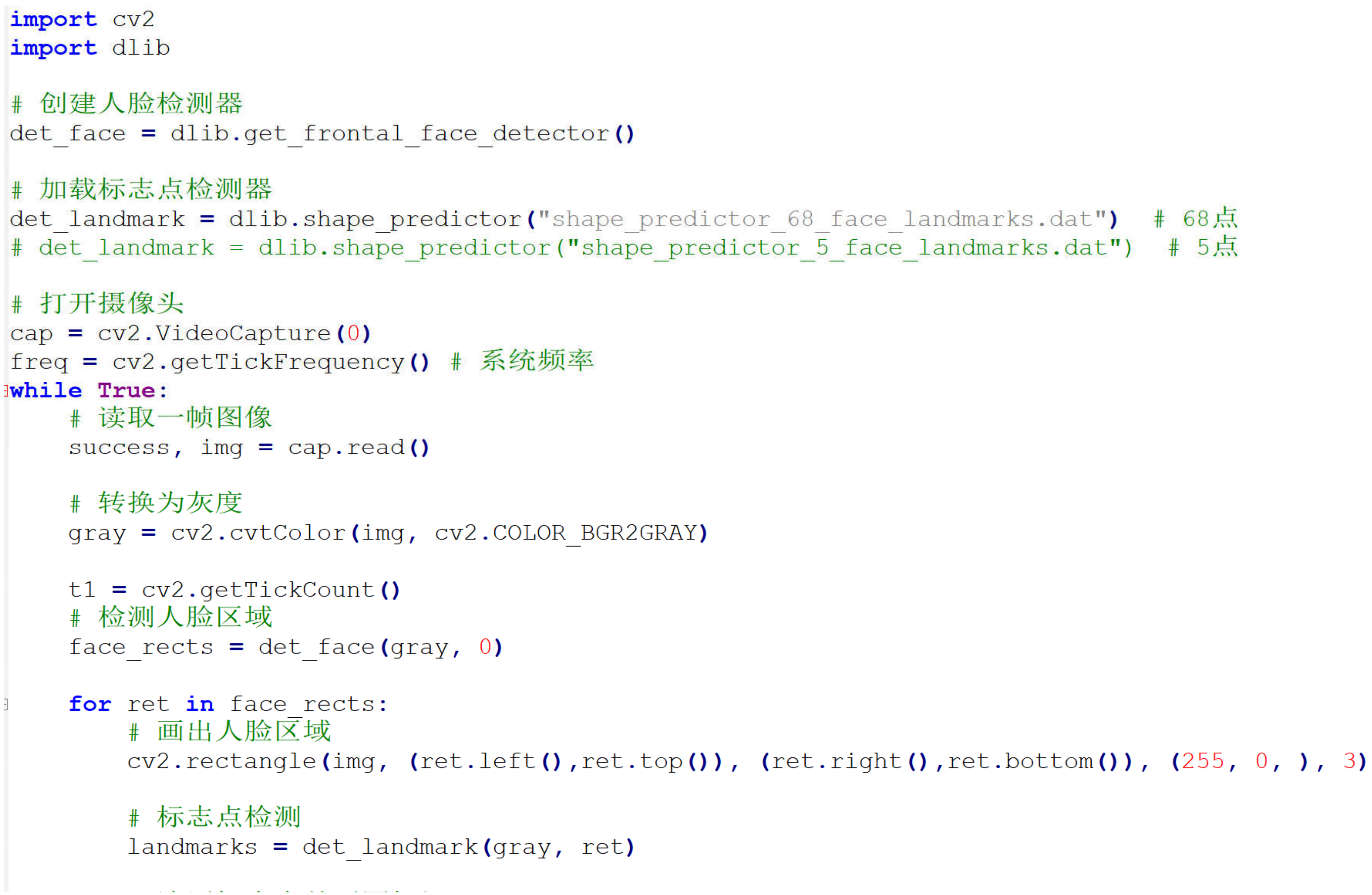
图 2.13: 人脸的 68 个标志点

Dlib的安装

```
pip3  install scipy
pip3  install scikit-image
pip3  install dlib
```

进行模型下载:

https://github.com/davisking/dlib-models



```
import cv2
import dlib

# 创建人脸检测器
det_face = dlib.get_frontal_face_detector()

# 加载标志点检测器
det_landmark = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")  # 68点
# det_landmark = dlib.shape_predictor("shape_predictor_5_face_landmarks.dat")  # 5点

# 打开摄像头
cap = cv2.VideoCapture(0)
freq = cv2.getTickFrequency() # 系统频率
while True:
    # 读取一帧图像
    success, img = cap.read()

    # 转换为灰度
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    t1 = cv2.getTickCount()
    # 检测人脸区域
    face_rects = det_face(gray, 0)

    for ret in face_rects:
        # 画出人脸区域
        cv2.rectangle(img, (ret.left(),ret.top()), (ret.right(),ret.bottom()), (255, 0, ), 3)

        # 标志点检测
        landmarks = det_landmark(gray, ret)
    # 遍历标志点并画圆标记
    for part in landmarks.parts():
        pt = (part.x,part.y)
        cv2.circle(img, pt, 2, (0,0,255),-1)


t2 = cv2.getTickCount()

fps = freq/(t2-t1)
# 显示速度
cv2.putText(img, 'FPS: %.2f'%(fps), (0, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255))

# 显示检测结果
cv2.imshow("Face",img)

# 按q退出
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```