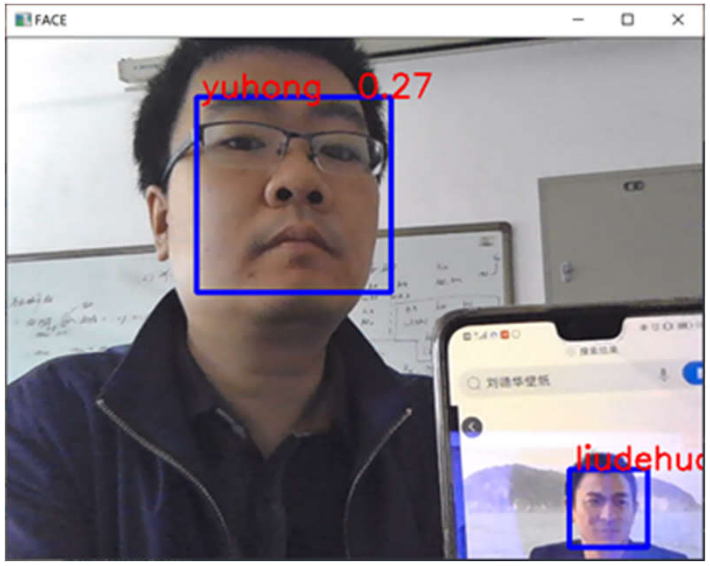
Python编程与人工智能实践



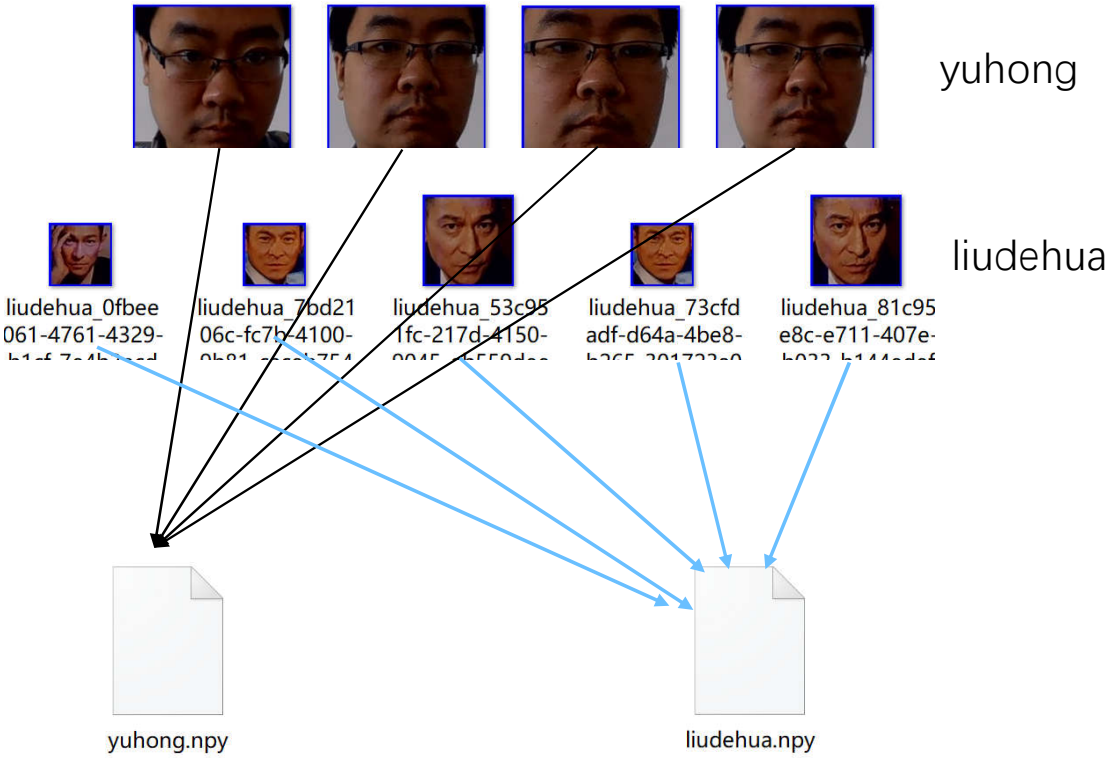
应用篇：基于Dlib的人脸识别

于泓
鲁东大学
信息与电气工程学院
2021.4.11

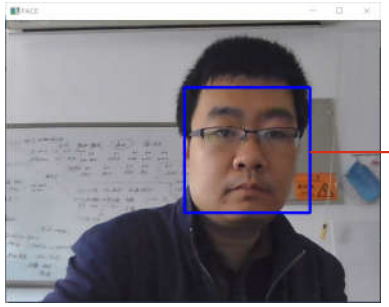
人脸识别的一般流程

(1) 人脸收集

(2) 人脸模型训练
（将每张图像转换为一个特征矢量）



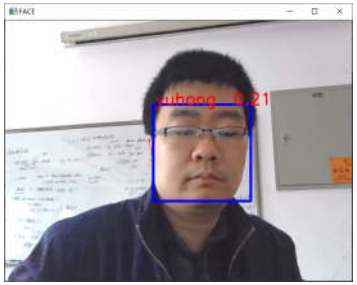
(3) 人脸检测，并提取人脸特征

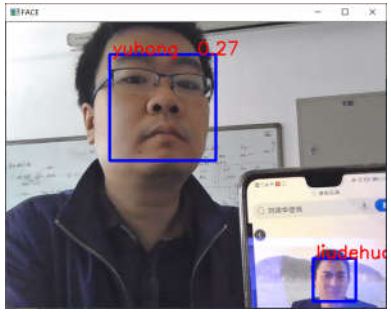


face_vector

计算相似度，找到最相似的模型

(4) 将face_vector与保存的
人脸模型进行比较
输出最相似的模型的ID





人脸收集 face_collect.py

```
import cv2
import os
import numpy as np
import uuid
import dlib
# 获取最大的人脸
def getMax_face(face_rects):

    if len(face_rects)==0:
        return 0,0
    face_areas =[ ]
    for rect in face_rects:
        area = (rect.bottom()-rect.top())*(rect.right()-rect.left())
        face_areas.append(area)
    index = np.argmax(face_areas)
    return face_areas[index],face_rects[index]
```

pip install uuid

人脸收集时只提取最大的人脸

```
def gen_face_name(str_face_id):
    return str_face_id+'_'+str(uuid.uuid4())+'.jpg'
```

为收集的人脸图片
起一个不重复的随机的名字

```
if __name__ == "__main__":

    #创建人脸文件保存文件夹
    os.makedirs('faces',exist_ok='True')

    # 创建人脸检测器
    det_face = dlib.get_frontal_face_detector()
    # 打开摄像头
    cap = cv2.VideoCapture(0)

    # 人脸ID
    str_face_id = ""

    while True:

        # 判断人脸id 是否为空，空的话创建face_id
        if str_face_id.strip()=="":
            str_face_id = input('Enter your face ID:')

            path_face = os.path.join('faces',str_face_id)
            os.makedirs(path_face,exist_ok='True')
```

在根目录下创建一个“faces”文件夹用来保存人脸数据

构建人脸检测器

循环开始时输入人脸ID并在faces目录下创建一个人脸ID的目录

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

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

# 检测人脸区域
face_rects = det_face(gray, 0)

# 得到最大的人脸区域
max_area,max_rect = getMax_face(face_rects)

# 画框
if max_area>10:
    cv2.rectangle(img, (max_rect.left(),max_rect.top()),
                       (max_rect.right(),max_rect.bottom()),
                       (255, 0, ), 3)
# 显示检测结果
cv2.imshow("FACE",img)
```

进行人脸检测

从检测到的区域中选取最大的一个作为人脸数据进行保存

```
    # 读取按键键值
    key =  cv2.waitKey(1) & 0xFF

    #  按键"c" 进行人脸采集
    if key == ord('c'):
        key = 0
        # 保存人脸
        if max_area>10:
            # 截取人脸图像
            roi = img[max_rect.top():max_rect.bottom(),max_rect.left():max_rect.right()]
            # 生成文件名
            save_face_name = os.path.join('faces',str_face_id,gen_face_name(str_face_id))
            # 文件保存
            cv2.imwrite(save_face_name,roi)
            print('save_face',save_face_name)


    #  按键"x" 切换 人脸_id
    elif key == ord('x'):
        key = 0
        str_face_id = ""

    # 按键 "q" 退出
    elif key ==  ord('q'):
        break
cap.release()
```

按c进行人脸保存
保存最大人脸区域

按x进行人脸切换

人脸模型保存（人脸特征收集）部分：
使用了Dlib提供的人脸特征提取模型
dlib_face_recognition_resnet_model_v1.dat

下载地址：https://github.com/davisking/dlib-models

该模型是一个包含了 **29** 个卷积层的 **ResNet** 网络，其具体结构参照了经典的 ResNet-34 网络并进行了一些缩减，减少了层数，并将每一层的滤波器数目减半。整个网路训练用了 **7485** 个人的大约 **300** 万张人脸。这些人脸采集自 face scrub 数据库、VGG 数据库 以及从网上采集的数据，经过了精细数据清理去除了错误标签以及容易判断错误的数据。网络训练完成后，中间层的 **128** 维输出被用作人脸特征。

该模型的输入要求为 **150 × 150** 像素的图像，在实际应用中检测到的人脸大都不符合尺寸要求，因此需要进行缩放和剪裁。因此需要借助**人脸关键点检测**模型（5个关键点），辅助进行有效缩放

5关键点检测模型：shape_predictor_5_face_landmarks.dat

```
# 加载人脸特征提取器
facerec = dlib.face_recognition_model_v1("dlib_face_recognition_resnet_model_v1.dat")
# 加载人脸标志点检测器
sp = dlib.shape_predictor("shape_predictor_5_face_landmarks.dat")
# 用来记录所有模型信息
with open('model.scp','w',encoding='utf-8')as f:

    # 遍历faces文件夹
    base_path = 'faces'
    for face_id in os.listdir(base_path):

        face_dir = os.path.join(base_path,face_id)
        if os.path.isdir(face_dir):
            # 遍历 base_path/face_id 文件夹
            file_face_model = os.path.join(face_dir,face_id+'.npy')
            face_vectors = []
```

用来记录 人脸ID 与 人脸模型

```
for face_img in os.listdir(face_dir):

    if os.path.splitext(face_img)[-1]=='.jpg':
        # 读取人俩图像并转换为RGB
        img = cv2.imread(os.path.join(face_dir,face_img))

        img = cv2.cvtColor(img, cv2. COLOR_BGR2RGB)

        # 存放的是已经截取好的人脸图片，所以在整图内检测标志点
        img = np.array(img)
        h,w,_ = np.shape(img)
        rect = dlib.rectangle(0,0,w,h)
        # 辅助人脸定位
        shape = sp(img, rect)
        print("Generate face vector of ",face_img)
        # 获取128维人脸特征
        face_vector = facerec.compute_face_descriptor(img,shape)

        # 保存图像和人脸ID
        face_vectors.append(face_vector)

    if len(face_vectors)>0:
        np.save(file_face_model,face_vectors)
        f.write('%s %s\n'%(face_id,file_face_model))
```

获取5个关键点

获取128维的人脸特征

人脸模型保存，并写入model.scp文件

model.scp face_recognize_dlib.py EXP2-2.asm EXP2-3.asm

```
1 huangbo faces\huangbo\huangbo.npy
2 liudehua faces\liudehua\liudehua.npy
3 yuhong faces\yuhong\yuhong.npy
4
```