

# 河南科技大学

## 课 程 项 目 报 告

课程名称 量化工程设计 2

题 目 人脸识别

学 院 机电工程学院

班 级 新工科 201

学生姓名 牛子贤

指导老师 彭建军 何社阳

日 期 2022 年 6 月 10 日

## 基于 python 的人脸识别实现

### 摘 要

目前人脸识别方法的研究方向主要有两个:其一是基于整体的研究方法,它主要是考虑了模式的整体属性,包括特征脸方法、模板匹配方法、弹性图匹配方法、隐马尔可夫模型方法以及神经网络方法等;其二是基于特征分析的方法,也就是将人脸基准点的相对比率和其它描述人脸脸部特征的形状参数或类别参数等一起构成识别特征向量。这种基于整体脸的识别不仅保留了人脸主要特征部件之间的拓扑关系,而且也保留了各部件本身的信息。基于特征分析的识别是通过提取出局部轮廓信息及灰度信息来设计具体识别算法。两种方式的人脸识别方法各有优点,本课程设计基于特征脸方法以及局部二值模式方法初步实现了人脸识别的功能,并比较了两种方法的优缺点。

**关 键 词:** 特征脸, 主成分分析, 人脸识别, 局部二值模式

## 目 录

前 言 .....	3
第 1 章 特征脸方法实现人脸识别 .....	4
§ 1.1 特征脸方法原理 .....	4
§ 1.1.1 特征脸算法解释 .....	5
§ 1.2 特征脸方法实验过程 .....	7
§ 1.2.1 程序代码实现 .....	7
§ 1.2.2 UI 的加入 .....	13
1、 UI 展示 .....	13
2、 采集数据模块 .....	13
3、 训练与识别结果 .....	14
第 2 章 局部二值模式方法原理及实验过程 .....	16
§ 2.1 局部二值模式解释 .....	16
§2.2 实验过程 .....	20
1. 采集数据 .....	20
2. 训练数据 .....	21
3. 人脸识别 .....	23
4. UI 导入和程序调用 .....	24
5. 效果展示 .....	25
第 3 章 总结 .....	27
1. 人脸识别发展与趋势 .....	27
2. 项目总结 .....	28
参考文献及网站 .....	29
致谢 .....	30

## 前 言

作为人工智能的一个重要应用，人脸机器自动识别是一项极具挑战性的难题。它在理论和应用中的潜在价值一直激励着科研人员的不懈努力。人脸检测和识别在公安部门犯罪搜索、安全部门动态监控、银行密码系统等许多领域有广泛的应用。与指纹、视网膜、虹膜、掌纹等其他人体生物特征识别方法相比，人脸识别具有直接、友好、使用者无心理障碍等特点。PCA 方法最早被 Turk 和 Pentland 用于人脸识别的研究，取得了不错的效果。由于其在降维和特征提取方面的有效性，在人脸识别领域得到了广泛的应用。PCA 方法的基本原理是：利用 K-L 变换抽取人脸的主要成分，构成特征脸空间，识别时将测试图像投影到此空间，得到一组投影系数，通过与各个人脸图像比较进行识别。局部二值模式算法将检测到的人脸分为小单元，并将其与模型中的对应单元进行比较，对每个区域的匹配值产生一个直方图。由于这种方法的灵活性，局部二值模式算法是唯一允许模型样本人脸和检测到的人脸在形状、大小上可以不同的人脸识别算法。

## 第 1 章 特征脸方法实现人脸识别

### §1.1 特征脸方法原理

特征脸方法是 90 年代初期由 Turk 和 Pentland 提出的目前最流行的算法之一，具有简单有效的特点，也称为基于主成分分析 (principal component analysis, 简称 PCA) 的人脸识别方法。

实际上，特征脸反映了隐含在人脸样本集合内部的信息和人脸的结构关系。将眼睛、面颊、下颌的样本集协方差矩阵的特征向量称为特征眼、特征颌和特征唇，统称特征子脸。特征子脸在相应的图像空间中生成子空间，称为子脸空间。计算出测试图像窗口在子脸空间的投影距离，若窗口图像满足阈值比较条件，则判断其为人脸。

基于特征分析的方法，也就是将人脸基准点的相对比率和其它描述人脸脸部特征的形状参数或类别参数等一起构成识别特征向量，这种基于整体脸的识别不仅保留了人脸部件之间的拓扑关系，而且也保留了各部件本身的信息，而基于部件的识别则是通过提取出局部轮廓信息及灰度信息来设计具体识别算法。现在 Eigenface (PCA) 算法已经与经典的模板匹配算法一起成为测试人脸识别系统性能的基准算法；而自 1991 年特征脸技术诞生以来，研究者对其进行了各种各样的实验和理论分析，FERET' 96 测试结果也表明，改进的特征脸算法是主流的人脸识别技术，也是具有最好性能的识别方法之一。

Turk 和 Pentland 提出特征脸的方法，它根据一组人脸训练图像构造主元子空间，由于主元具有脸的形状，也称为特征脸，识别时将测试图像投影到主元子空间上，得到一组投影系数，和各个已知人的人脸图像比较进行识别。Pentland 等报告了相当好的结果，在 200 个人的 3000 幅图像中得到 95% 的正确识别率，在 FERET 数据库上对 150 幅正面人脸象只有一个误识别。但系统在进行特征脸方法之前需要作大量预处理工作如归一化等。

在传统特征脸的基础上，研究者注意到特征值大的特征向量（即特征脸）并不一定是分类性能好的方向，据此发展了多种特征（子空间）选择方法，如 Peng 的双子空间方法、Weng 的线性歧义分析方法、Belhumeur 的 FisherFace 方法等。事实上，特征脸方法是一种显式主元分析人脸建模，一些线性自联想、线性

压缩型 BP 网则为隐式的主元分析方法,它们都是把人脸表示为一些向量的加权和,这些向量是训练集叉积阵的主特征向量,Valentin 对此作了详细讨论。总之,特征脸方法是一种简单、快速、实用的基于变换系数特征的算法,但由于它在本质上依赖于训练集和测试集图像的灰度相关性,而且要求测试图像与训练集比较像,所以它与有着很大的局限性。

### §1.1.1 特征脸算法解释

特征子脸技术的基本思想是:从统计的观点,寻找人脸图像分布的基本元素,即人脸图像样本集协方差矩阵的特征向量,以此近似地表征人脸图像。这些特征向量称为特征脸(Eigenface)。

一组特征脸可以通过在一大组描述不同人脸的图像上进行主成分分析(PCA)获得。任意一张人脸图像都可以被认为是这些标准脸的组合。例如,一张人脸图像可能是特征脸 1 的 10%,加上特征脸 2 的 55%,在减去特征脸 3 的 3%。值得注意的是,它不需要太多的特征脸来获得大多数脸的近似组合。另外,由于人脸是通过一系列向量(每个特征脸一个比例值)而不是数字图像进行保存,可以节省很多存储空间。

特征脸算法理论实现过程如下:

- (1) 对图片进行预处理:将图片灰度化,调整到统一尺寸 138\*168。
- (2) 将图片转换为一个向量:经过灰度化处理的图片是一个矩阵,将这个矩阵中的每一行连到一起,则可以变为一个向量,将该向量转换为列向量。
- (3) 将数据集种的所有图片都转换为向量后,这些数据可以组成一个矩阵,在此基础上进行零均值化处理,就是将所有人脸在对应的维度求平均,得到一个平均脸(average face)向量,每一个人脸向量减去该向量,从而完成零均值化处理。
- (4) 将经过零均值化处理的图像向量组合在一起,可以得到一个矩阵。通过该矩阵可以得到 PCA 算法中的协方差矩阵。
- (5) 计算协方差矩阵的特征值和特征向量,每一个特征向量的维度与原始图像向量的维度是一致的,因此这些特征向量可以看成是一致的,因此这些特征向量就是所谓的特征脸。
- (6) 去计算测试图片和训练后的矩阵映射后的空间中的欧式距离,距离最近的就是要识别的人脸。

具体步骤如下：

1. 对收集的图片拼音转汉字重命名批处理，便于后续识别；
1. 遍历循环图片库，得到所有人名字列表；
2. 把图库中的图片重新排序，与名字列表一一对应；
3. 加载图片集，每张图片相当于一个列向量，生成一个矩阵  $A$  (23184x600)；
4. 计算每一行的均值，减去这个平均脸，均值化处理；
5. 计算  $A \cdot A^T$  的特征向量，由于计算量较大，先算  $A^T \cdot A$  的特征向量，再用  $A \cdot A^T$  的特征向量；
6. 把  $A$  映射到  $T$ ；
7. 计算计算测试图片和  $T$  每一列的欧式距离；
8. 用 harra 分类器框出人脸区域，并标注名字，完成识别。

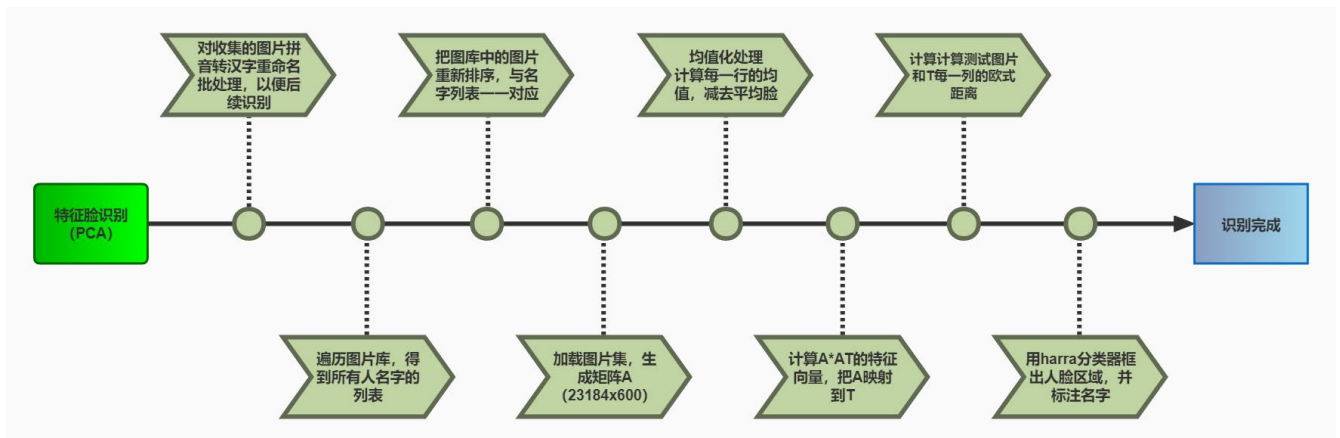


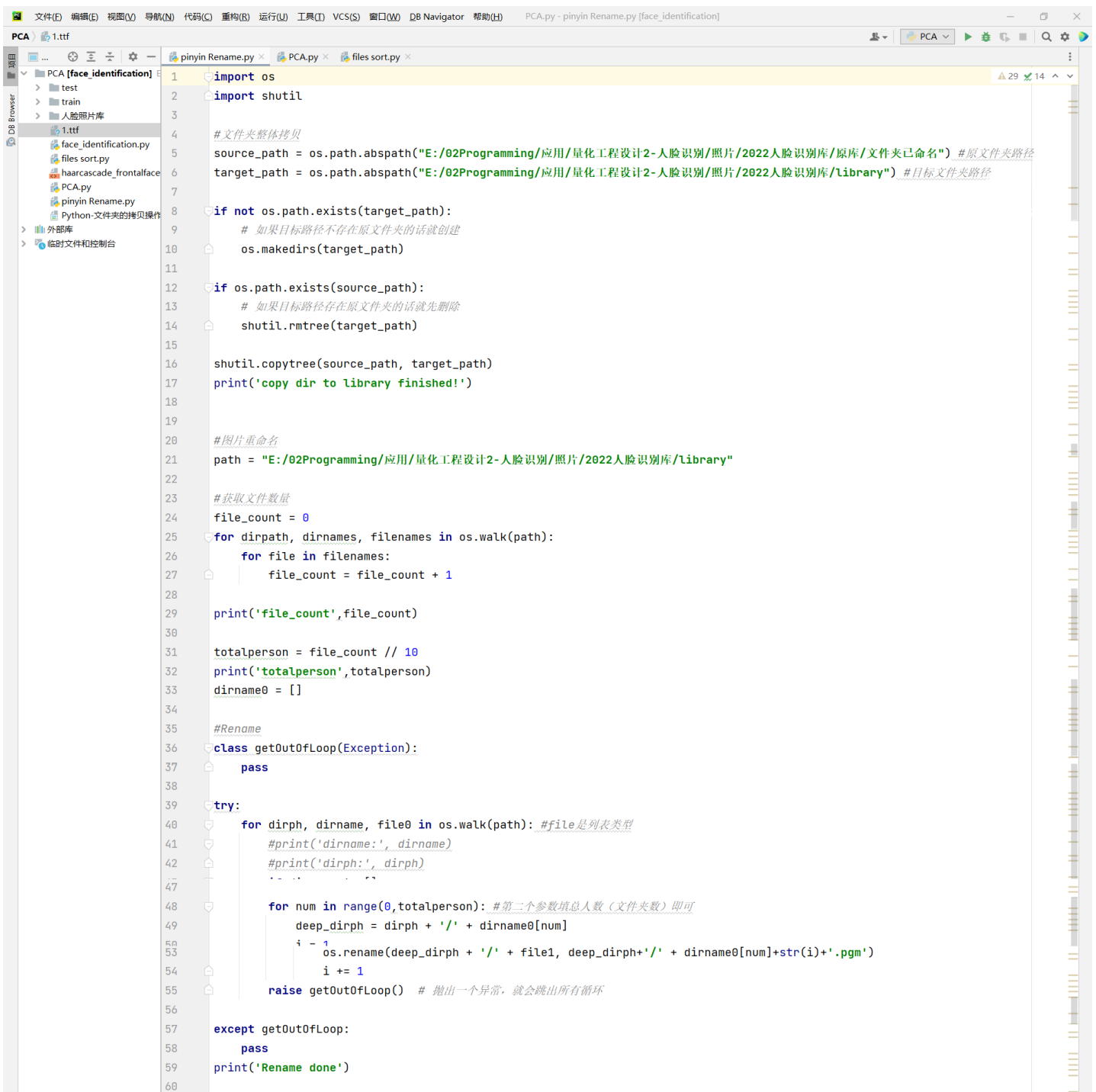
图 1 PCA 算法

## §1.2 特征脸方法实验过程

### § 1.2.1 程序代码实现

#### 1. 重命名 Rename

通过 Rename 将以拼音命名的图片重新以汉字命名，批处理提高效率。



```
1 import os
2 import shutil
3
4 # 文件夹整体拷贝
5 source_path = os.path.abspath("E:/02Programming/应用/量化工程设计2-人脸识别/照片/2022人脸识别库/原库/文件夹已命名") # 原文件夹路径
6 target_path = os.path.abspath("E:/02Programming/应用/量化工程设计2-人脸识别/照片/2022人脸识别库/Library") # 目标文件夹路径
7
8 if not os.path.exists(target_path):
9     # 如果目标路径不存在原文件夹的话就创建
10     os.makedirs(target_path)
11
12 if os.path.exists(source_path):
13     # 如果目标路径存在原文件夹的话就先删除
14     shutil.rmtree(target_path)
15
16 shutil.copytree(source_path, target_path)
17 print('copy dir to Library finished!')
18
19
20 # 图片重命名
21 path = "E:/02Programming/应用/量化工程设计2-人脸识别/照片/2022人脸识别库/Library"
22
23 # 获取文件数量
24 file_count = 0
25 for dirpath, dirnames, filenames in os.walk(path):
26     for file in filenames:
27         file_count = file_count + 1
28
29 print('file_count', file_count)
30
31 totalperson = file_count // 10
32 print('totalperson', totalperson)
33 dirname0 = []
34
35 #Rename
36 class getOutOfLoop(Exception):
37     pass
38
39 try:
40     for dirph, dirname, file0 in os.walk(path): # file 是列表类型
41         # print('dirname:', dirname)
42         # print('dirph:', dirph)
43         ...
44
45         for num in range(0, totalperson): # 第二个参数填总人数 (文件夹数) 即可
46             deep_dirph = dirph + '/' + dirname0[num]
47             i = 1
48             os.rename(deep_dirph + '/' + file1, deep_dirph + '/' + dirname0[num] + str(i) + '.pgm')
49             i += 1
50         raise getOutOfLoop() # 抛出一个异常, 就会跳出所有循环
51
52 except getOutOfLoop:
53     pass
54
55 print('Rename done')
```

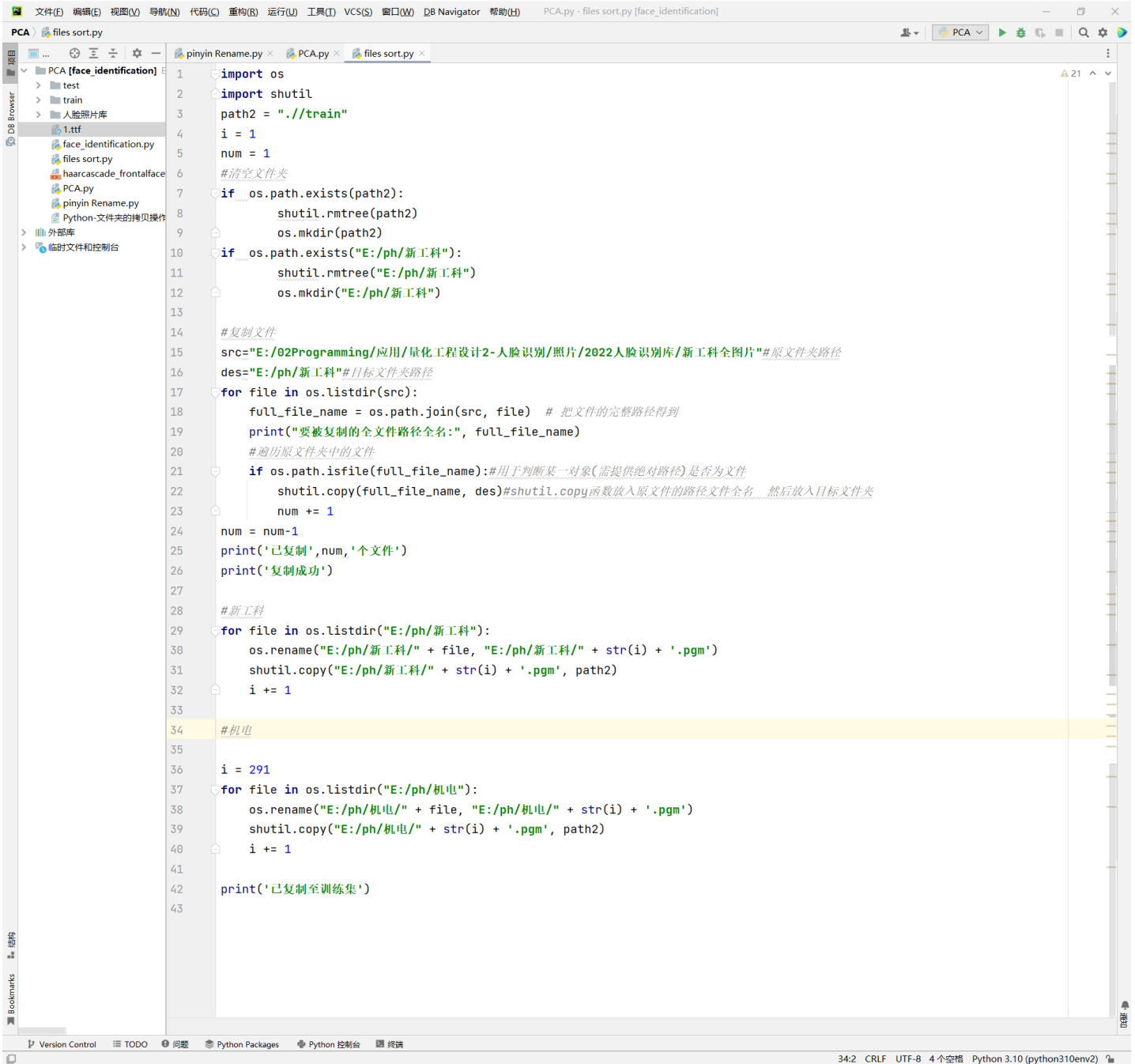


## 2、获取名字列表

```
# 获取名字列表
i = 0
for dirph, dirnames, file in os.walk(library): #file是列表类型
    for files in os.listdir(dirph): #files是独立的文件名
        if files.endswith(".pgm"):
            i += 1
            if dirph.endswith("电照片"):
                # print(files[:3], ': ', files)
                if i % 10 == 0:
                    name.append(files[:3])
            else:
                if dirph[-3:].startswith('\\'):
                    # print(dirph[-2:], ': ', files)
                    if i % 10 == 0:
                        name.append(dirph[-2:])
                else:
                    # print(dirph[-3:], ': ', files)
                    if i % 10 == 0:
                        name.append(dirph[-3:])
print(name) # 打印名字列表
```

### 3、图片顺序重排

将所有图片整合到一起，以 1~300 排序，以满足代码模块识别要求。



```

1  import os
2  import shutil
3  path2 = "./train"
4  i = 1
5  num = 1
6  #清空文件夹
7  if os.path.exists(path2):
8      shutil.rmtree(path2)
9      os.mkdir(path2)
10 if os.path.exists("E:/ph/新工科"):
11     shutil.rmtree("E:/ph/新工科")
12     os.mkdir("E:/ph/新工科")
13
14 #复制文件
15 src="E:/02Programming/应用/量化工程设计2-人脸识别/照片/2022人脸识别库/新工科全图片"#原文件夹路径
16 des="E:/ph/新工科"#目标文件夹路径
17 for file in os.listdir(src):
18     full_file_name = os.path.join(src, file) # 把文件的完整路径得到
19     print("要被复制的全文件路径全名:", full_file_name)
20     #遍历原文件夹中的文件
21     if os.path.isfile(full_file_name):#用于判断某一对象(需提供绝对路径)是否为文件
22         shutil.copy(full_file_name, des)#shutil.copy函数放入原文件的路径文件全名 然后放入目标文件夹
23         num += 1
24     num = num-1
25     print('已复制',num,'个文件')
26     print('复制成功')
27
28 #新工科
29 for file in os.listdir("E:/ph/新工科"):
30     os.rename("E:/ph/新工科/" + file, "E:/ph/新工科/" + str(i) + '.pgm')
31     shutil.copy("E:/ph/新工科/" + str(i) + '.pgm', path2)
32     i += 1
33
34 #机电
35
36 i = 291
37 for file in os.listdir("E:/ph/机电"):
38     os.rename("E:/ph/机电/" + file, "E:/ph/机电/" + str(i) + '.pgm')
39     shutil.copy("E:/ph/机电/" + str(i) + '.pgm', path2)
40     i += 1
41
42 print('已复制至训练集')
43

```

## 4、PCA 实现

```
pinyin Rename.py × PCA.py × files sort.py ×
27 # 训练图片
28 def loaddatas(Path):
29     A = []
30     number = len(os.listdir(Path))
31     for i in range(1, number + 1):
32         img = cv2.imread(Path + '\\ ' + str(i) + '.pgm', cv2.IMREAD_GRAYSCALE)
33         img = cv2.resize(img, (138,168))
34         img = img.ravel() # ravel函数是 numpy 的函数: 将多维数组中的元素变成一个一维数组
35         A.append(img)
36     A = np.array(A)
37     return A.T
38
39
40 # PCA实现
41 def PCA(X, num):
42     mean = X.mean(axis=1) # 对每行求均值
43     mean = mean.reshape(mean.shape[0], 1)
44     A = X - mean # 零均值化处理
45     # 求协方差矩阵的特征向量, 由于样本维度远远大于样本数目, 所以不直接求协方差矩阵
46     M = np.dot(A.T, A)
47     # AT*A的特征值和特征向量
48     evalue, evector = np.linalg.eig(M)
49     # A*AT的特征值和特征向量
50     evector = np.mat(np.dot(A, evector))
51     evalueindex = np.argsort(evalue)
52     evalueindex = evalueindex[::-1]
53     evalueindex = evalueindex[:num] # 取前num个特征
54     real_evector = evector[:, evalueindex]
55     return real_evector, mean, A
56
```

## 5、欧氏距离计算

```
def predict(testpath, real_evector, A, mean):
    testing1 = cv2.imdecode(np.fromfile(testpath, dtype=np.uint8), cv2.IMREAD_UNCHANGED) # 读取中文路径的
    testing1 = cv2.resize(testing1, (138,168))
    testing = cv2.imdecode(np.fromfile(testpath, dtype=np.uint8), cv2.IMREAD_UNCHANGED) # 读取中文路径的
    testing = cv2.resize(testing, (138,168))
    testing = np.reshape(testing, (-1, 1))
    testing = np.array(testing)
    difftesting = testing - mean
    testing = np.dot(real_evector.T, difftesting)
    dataimg = np.dot(real_evector.T, A)
    distancelist = []
    # 遍历每一个特征（每一列），差值最小的一组数据即为最近，就是相对应的人脸
    for i in range(0, real_evector.shape[1]):
        data = dataimg[:, i]
        temp = np.linalg.norm(testing - data) # 欧式距离计算
        distancelist.append(temp)

    index = np.argmin(distancelist)
    res = int(index)/10
    cv_show('test', testing1)
    return res, testing1
```

## 6、框出人脸区域，标注姓名，完成识别

```
def showname(xmlpath, imag, res):
    faceCascade = cv2.CascadeClassifier(xmlpath)
    faceCascade.load(xmlpath)
    faces = faceCascade.detectMultiScale(imag, scaleFactor=1.15, minNeighbors=3, minSize=(3, 3))
    #print("res", res)
    #print("int_res", int(res))
    testname = name[int(res)]
    print(str(testname))
    # 逐个标注人脸
    for (x, y, w, h) in faces:
        cv2.rectangle(imag, (x, y), (x + w, y + h), (255, 255, 0), 2)
        cv2ImgAddText(imag, testname, x, y, textSize=16)

#主程序
DATA = loaddatas(path2) # 加载图片训练
real_evector, mean, A = PCA(DATA, 290) # 得出特征向量
for fil in os.listdir(testpath):
    res, testing = predict(testpath + '\\ ' + fil, real_evector, A, mean) # 预测识别图片名字
    showname(pathf, testing, res)
```

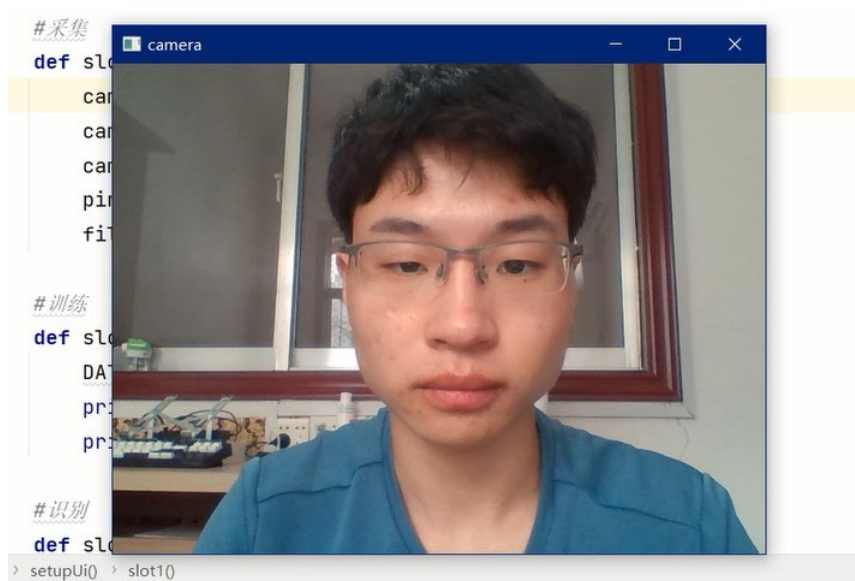
## § 1.2.2 UI 的加入

### 1、 UI 展示

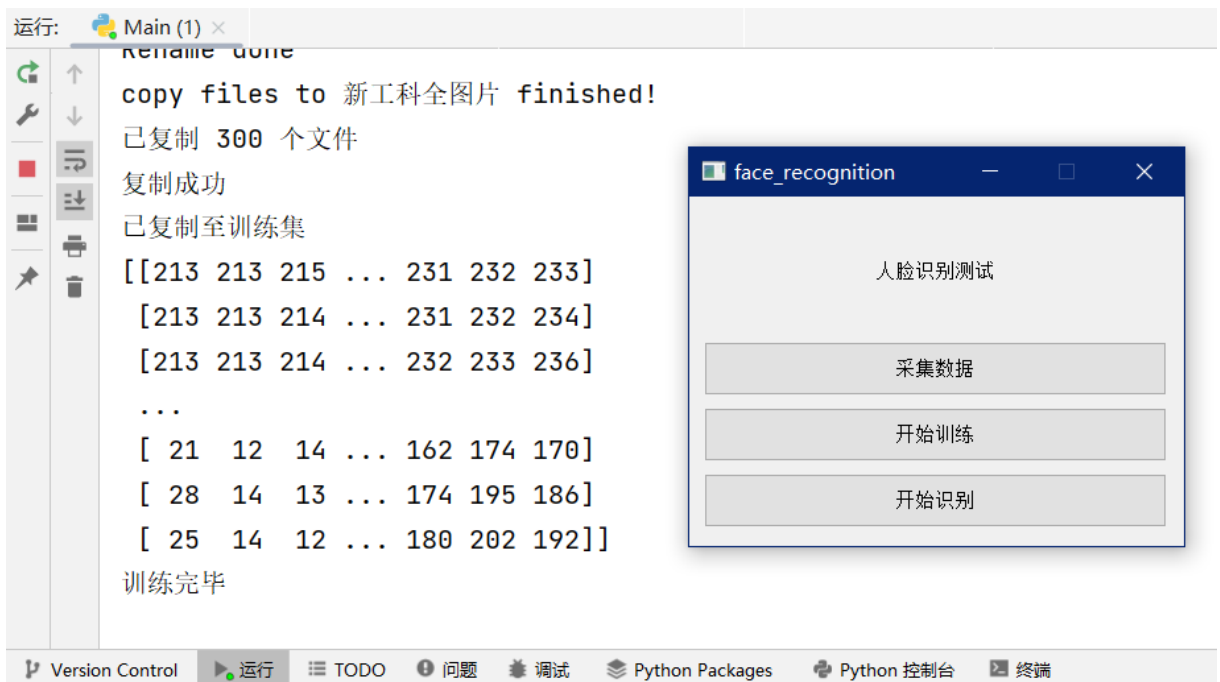


### 2、 采集数据模块

调用电脑摄像头，按空格可拍照保存并导入训练集和识别库，采集十张图片自动退出。



### 3、 训练与识别结果





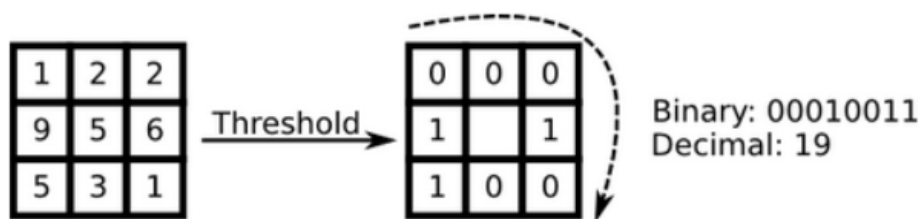
可以看到，不管是新采集的图像还是库中人脸图像，代码都能很好地识别。



## 第 2 章 局部二值模式方法原理及实验过程

### §2.1 局部二值模式解释

将一副人脸图像用局部二值模式来转化，每个像素点可以用如图所示二进制数来描述。



图像中每个采样点的值用下述方程来描述：

$$LBP_{(P,R)} = \sum_{i=0}^{P-1} s(x_i - x_c) * 2^i, (i = 0, 1, \dots, P-1)$$

$$s(u) = \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases}$$

特征脸算法从整体出发，实现了用人脸表示人脸，有效地降低了运算量。但对检测人脸约束要求较高，受环境、光照等外部条件影响较大。局部二值模式方法从像素值出发，充分利用了照片中的信息，可以在不同环境下完成人脸识别，但运算量较大。

下面我们采用与 opencv 相结合的方法来实现 LBPH 人脸识别。

OpenCV 环境带有的内置人脸识别器有以下 3 种：

### (1) EigenFaces 人脸识别器：

这个人脸识别器是特征脸识别器，在 Python 中调用方式为 `cv2.face.createEigenFaceRecognizer()`，这种识别器就相当于我们量化工程设计 2 课堂上讲解的特征脸、特征向量，以及 PCA 主成分分析法，这个面部识别器的原理是，并不是脸上所有部分都很重要，他会对面部中一些独特的特征进行捕捉，他关注的只是最大变化的区域，也就是差异最大的地方、最能将一个人和其他人区分开来的地方。EigenFaces 特征脸的识别器，能够捕获到人脸之间的最大变化，利用这一点就可以帮助我们区分不同的人脸，这就是特征脸识别器的工作原理。

EigenFaces 人脸识别器会将所有人的训练图像作为一个大的整体，也就是将所有人的照片作为一个整体，然后去提取重要的和有用的成分，丢弃其余的成分，提取出来的重要的和有用的成分，即为我们在 PCA 中提到的主成分。

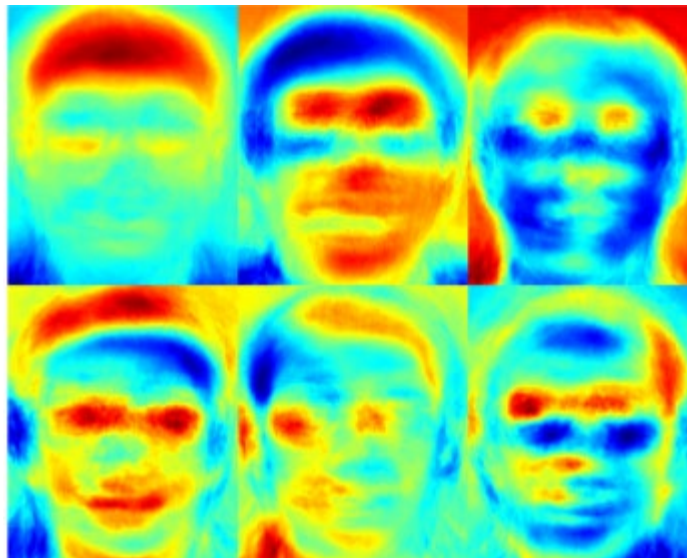


图 2 特征脸显示图像（即 EigenFaces）

他会记录下来哪个主成分对应哪个人，这就是在完成训练的过程，当下一次输入新的图像的时候，他就会把这个图像的信息去和训练期间存储的信息库进行对比，找到匹配最好的组件，然后返回最佳匹配的标签（人的名字）。

## (2) FisherFaces 人脸识别器:

该人脸识别器是对第一种人脸识别器的改进,它可以同时查看所有人训练集照片中的脸部特征,并且从这些所有的脸部数据里面找到主要的组成部分。然后通过从所有的人脸中捕获到的主要组成部分,不把注意力集中放在区分一个人和另一个人的特征上,而是集中代表整个训练数据中所有人的所有脸部特征。但是第 1、2 种识别器都有一个缺点,就是光照对其影响很大,因为他们会把光照作为一个重要的特征,也就是把光照作为一种主成分,那么就需要用到第 3 种人脸识别器来进行识别,也就是本任务人脸识别的核心:LBPH。

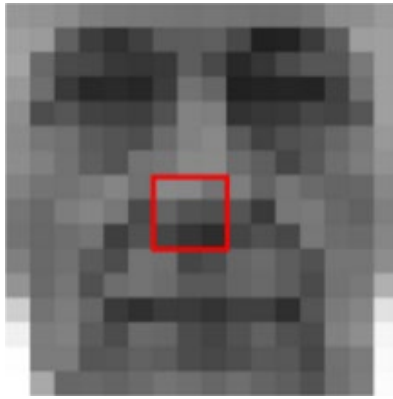


图 3 FisherFaces 方法提取到的人脸信息 (来源于官方训练库)

## (3) LBPH 人脸识别器:

由于实际情况中不能时时刻刻保证良好的光照条件, LBPH 人脸识别器就是克服这个缺点的一种改进。LBPH 即 Local Binary Patterns Histograms, 为局部二值模式直方图, 这种方法不去查看整个图像, 而是去查找图像的局部特征, 找出图像的局部结构, 并且通过比较每个像素与其相邻的像素来完成人脸的训练。

取一个 3x3 的窗口, 每移动一个图像(图像每个局部), 将中心的像素与相邻像素进行比较。强度值小于或等于中心像素的邻域用 1 表示, 其它邻域用 0 表示。以顺时针的顺序读取 3x3 窗口下的 0/1 值, 得到一个像 11100011 这样的二进制模式, 这个模式在图像的特定区域是局部的。在整个图像上这样做, 得到一个局部二进制模式的列表。



137	135	115
99	82	79
70	54	45

得到局部二进制模式列表之后，将二进制的信息转化成十进制，将这些值都放进同一张直方图里。通过这种方法得到训练集中每个人脸图像的直方图表示，并存储起来，同时算法将直方图和每个人的标签信息（姓名）进行绑定。



图 4

图为一亮一暗光照下同一个人的局部人脸信息，很明显看到这种方式可以忽略光照对人脸的影响。（图片中的局部人脸来自于 OpenCV 官方库中的人脸训练集，用于前期测试）

## §2.2 实验过程

### 1. 采集数据

定义摄像头读取函数，调用系统摄像头，得到每帧图像并显示，加入 if 函数实现按键保存和退出。

```
#摄像头
def camera(name):

    path = ("./data/jm/")
    # if os.path.exists(path):
    #     shutil.rmtree(path) # os.mkdir(path)

    cap=cv2.VideoCapture(0)
    num = 1
    while(cap.isOpened()):#检测是否在开启状态
        ret_flag,Vshow = cap.read()#得到每帧图像
        cv2.imshow("Capture_Test",Vshow)#显示图像
        k = cv2.waitKey(1) & 0xFF#按键判断
        if k == ord('s'):#保存
            cv2.imwrite("./data/jm/"+str(num)+". " + name + ".jpg", Vshow)
            #cv2.imencode('.jpg', Vshow)[1].tofile(path+str(num)+". " + name + ".jpg")

            print("success to save"+str(num)+".jpg")
            print("-----")
            num += 1
        elif k == ord(' '):#退出
            break

    #释放摄像头
    cap.release()

    #释放内存
```



## 2. 训练数据

### 1) 导入模块

先把需要的库进行导入。在本项目中，OS 库用于处理文件路径及信息；cv2 即 OpenCV 的库，用于图像处理和人脸识别器调用；PIL 是优秀的图像处理框架，用于完成图像归档和图像处理两方面功能需求；numpy 是用于进行数学计算，在本项目中用于矩阵运算。

```
import os
import cv2
from PIL import Image
import numpy as np
```

### 2) 获取图像数组和 id 标签数组和姓名

加载人脸识别分类器检测人脸，，获取人脸特征，获取图片 id 和姓名。

```
def getImageAndLabels(path):
    facesSamples=[]
    ids=[]
    imagePath=[os.path.join(path,f) for f in os.listdir(path)]
    #检测人脸
    face_detector = cv2.CascadeClassifier('D:/opencv/sources/data/haarcascades/haarcascade_frontalface_alt2.xml')
    #打印数组imagePaths
    print('数据排列: ',imagePaths)
    #遍历列表中的图片
    for imagePath in imagePath:
        #打开图片,黑白化
        PIL_img=Image.open(imagePath).convert('L')
        #将图像转换为数组,以黑白深浅
        # PIL_img = cv2.resize(PIL_img, dsize=(400, 400))
        img_numpy=np.array(PIL_img,'uint8')
        #获取图片人脸特征
        faces = face_detector.detectMultiScale(img_numpy)
        #获取每张图片的id和姓名
        id = int(os.path.split(imagePath)[1].split('.')[0])
        #预防无面容照片
        for x,y,w,h in faces:
            ids.append(id)
            facesSamples.append(img_numpy[y:y+h,x:x+w])
    #打印脸部特征和id
    #print('fs:', facesSamples)
```

```
print('id:', id)
#print('fs:', facesSamples[id])
print('fs:', facesSamples)
#print('脸部例子: ', facesSamples[0])
#print('身份信息: ', ids[0])
return facesSamples, ids
```

调用 LBPH 人脸识别器，训练后将训练文件保存到 yml 格式文件，方便后续识别直接调用训练数据。

```
if __name__ == '__main__':
    # 图片路径
    path = './data/jm/'
    # 获取图像数组和id 标签数组和姓名
    faces, ids = getImageAndLabels(path)
    # 获取训练对象
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    # recognizer.train(faces, names) # np.array(ids)
    recognizer.train(faces, np.array(ids))
    # 保存文件
    recognizer.write('trainer/trainer0.yml')
    # save_to_file('names.txt', names)
```

### 3. 人脸识别

检测人脸，得到相似度评分，根据置信评分标注相应姓名或“unknown”，置信评分越大越不可信。

```
#准备识别的图片
def face_detect_demo(img):
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)#转换为灰度
    face_detector=cv2.CascadeClassifier('D:/opencv/sources/data/haarcascades/haarcascade_frontalface_alt2.xml')
    face=face_detector.detectMultiScale(gray)
    #face=face_detector.detectMultiScale(gray)
    for x,y,w,h in face:
        cv2.rectangle(img,(x,y),(x+w,y+h),color=(0,0,255),thickness=2)
        #cv2.circle(img,center=(x+w//2,y+h//2),radius=w//2,color=(0,255,0),thickness=1)
        # 人脸识别
        label, confidence = recognizer.predict(gray[y:y + h, x:x + w])
        print('标签id:',label,'置信评分: ', confidence)#越大越不可信
        if confidence > 50:
            cv2.putText(img, 'unkonw', (x + 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 1)
        else:
            if label < len(names):
                cv2.putText(img,names[label-1], (x + 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 1)
                #cv2.putText(img, names[ids], x+10, y-10, textSize=16)
    cv2.imshow('result',img)
    #print('bug:',ids)
```



## 4. UI 导入和程序调用

### 1. 导入程序，调用 Pyqt5

```
from PyQt5 import QtCore, QtWidgets
import sys
import capture1
import train2
import recognition3
```

### 2. 定义 ui 按钮

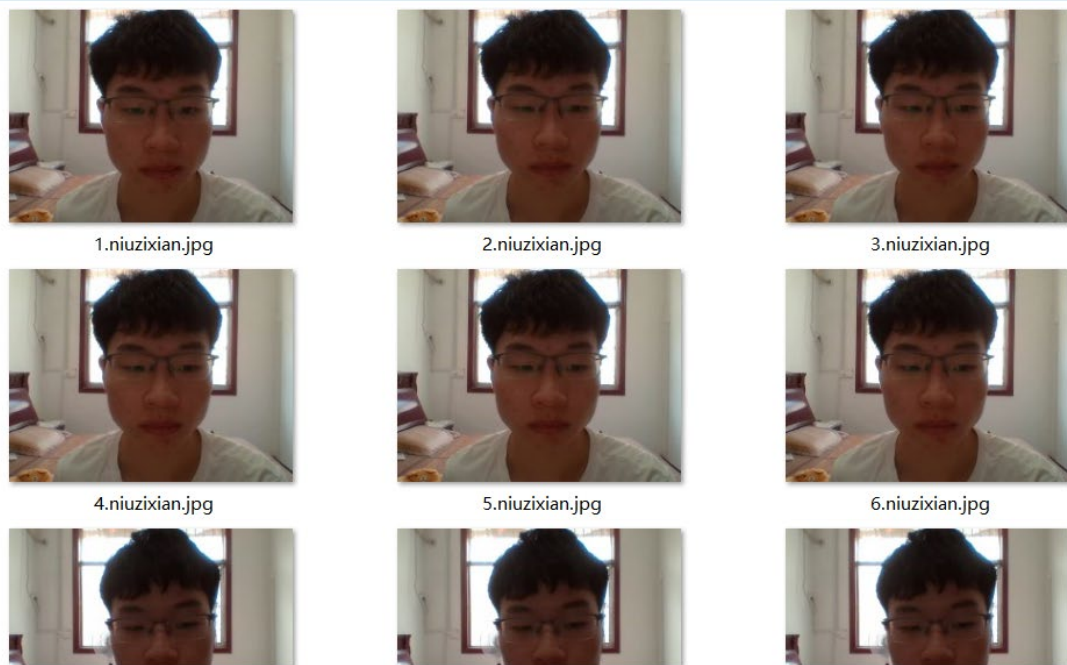
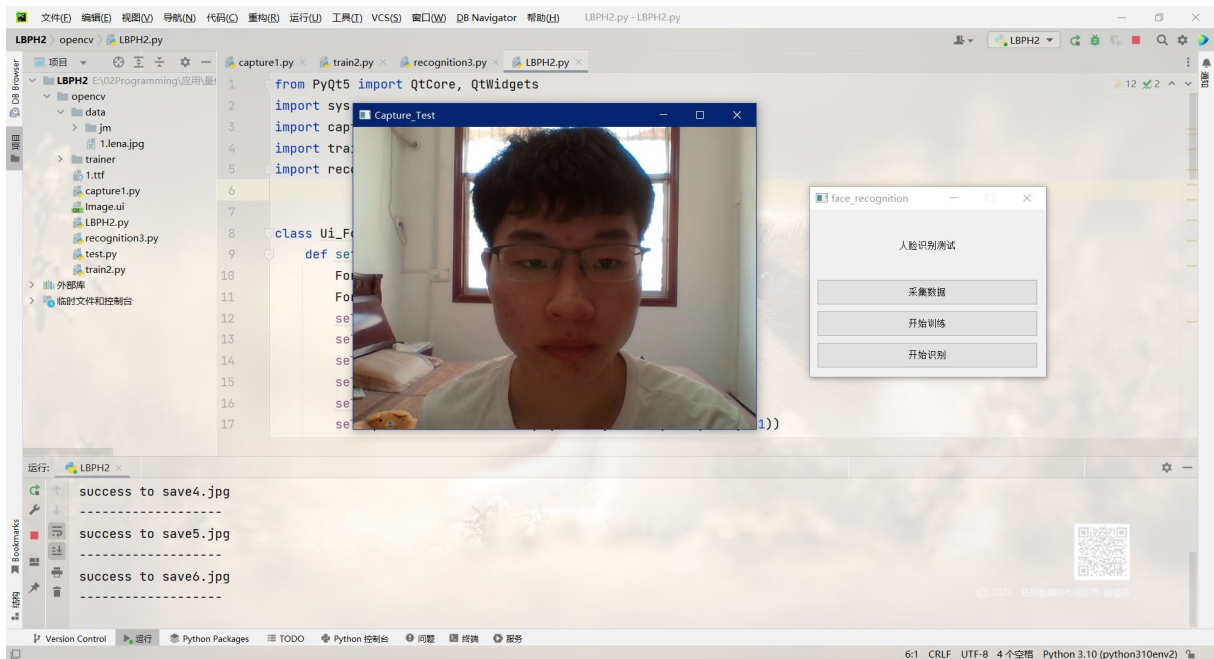
```
#采集
def slot1():
    capture1.camera("niuzixian")

#训练
def slot2():
    train2.train()
    print("训练完毕")

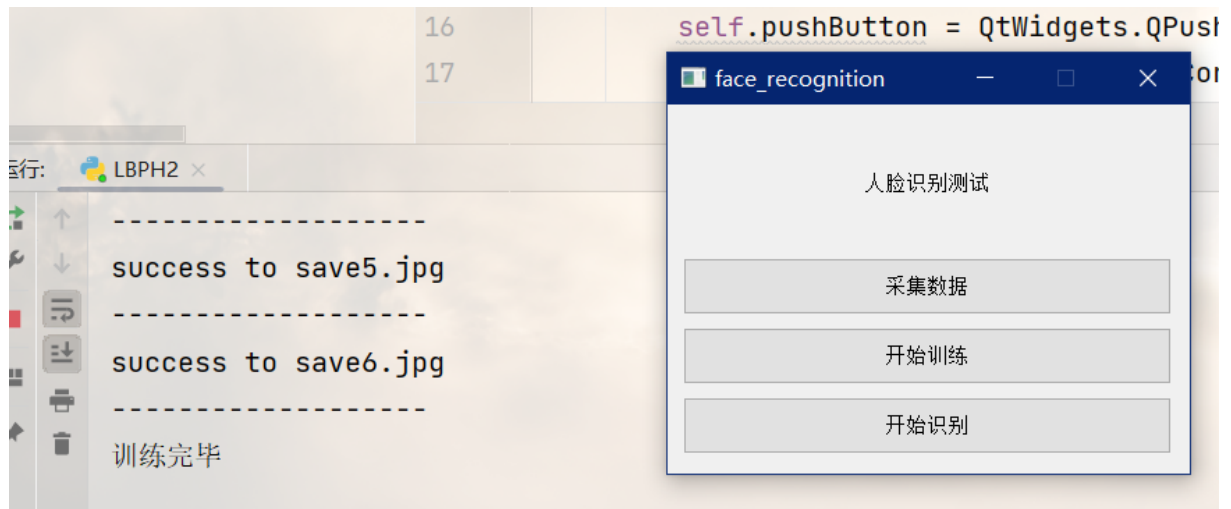
#识别
def slot3():
    recognition3.recognition()
```

## 5. 效果展示

### 1) 采集数据与保存



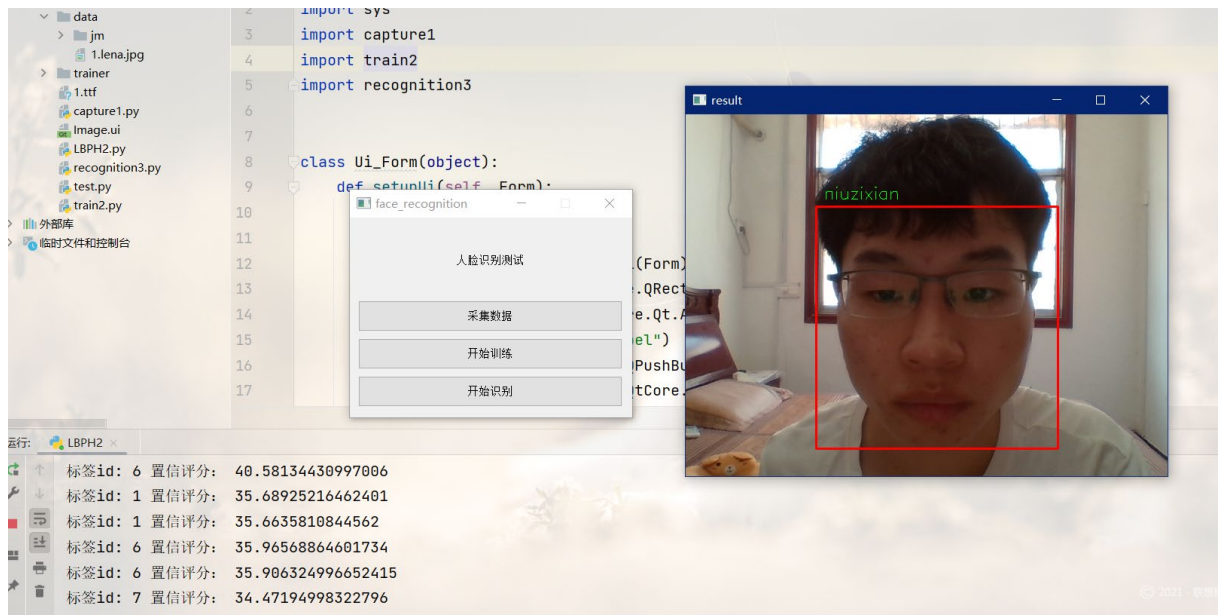
## 2) 训练与训练数据保存



量化工程设计2-人脸识别 > 代码 > 人脸识别牛子贤 > LBPH2 > opencv > trainer

名称	修改日期	类型	大小
trainer0.yml	2022/6/10 10:50	Yaml 源文件	3,211 KB

## 3) 人脸识别结果



## 第3章 总结

### 1. 人脸识别发展与趋势

人脸识别是基于人的脸部特征信息进行身份识别的一种生物识别技术。人脸识别技术渗透率不断提高，在给人们生活带来便利和安全的同时，将持续带动人脸识别行业发展。随着人脸识别技术的不断成熟，人脸识别技术逐渐被人们所熟知。同时，计算机、光学成像等相关技术的高速发展，人脸识别在各领域的应用不断拓展，人脸识别行业市场规模持续增长。数据显示，2020年中国人脸识别行业市场规模达到45亿元，预计2022年中国人脸识别市场规模将达到68亿元。近年来，国家及相关部门出台了一系列政策，鼓励人工智能和人脸识别行业的发展。在政策支持下，中国人工智能在国内的快速发展，科技巨头扎堆布局，越来越多的产业资本开始关注人脸识别。人脸识别技术逐渐在智慧城市、公共安全、轨道交通、政府治理及交通等行业的应用，为中国人脸识别行业奠定坚实的基础。

人脸识别技术存在三个类型的技术场景。第一种是基于移动终端的单向存储的人脸识别技术。由于单向存储和终端存储，因此泄露的可能性最低。第二种是基于“云端”生物介质的人脸识别技术。相对来说风险会高一些。不过，随着技术进步，云端人脸识别技术的安全性也会变得更高。第三种则是风险性比较大的，就是随机获取的人脸母版和静态人脸识别系统。

人脸识别测温系统产品已广泛研究应用于金融、司法、军队、公安、边检、政府、航天、电力、工厂、教育、医疗及众多企事业单位等领域。目前，人脸识别技术已经在多个领域落地，尤其是在安防和金融领域应用最广。海康威视、旷视等与公安部门、零售企业积极合作，通过自身技术颠覆性改变了传统刑侦和零售场景。下游应用市场不断拓展，推动3D识别、活体检测等关键技术引领人脸识别行业发展，实现人脸识别创新。未来，在人工智能等新一代信息技术加速商用，以及其他核心技术得到打破的情况下，人脸识别很有可能超过指纹识别，像二维码一样实现大规模普及。作为一种更为高效的身份验证和识别手段，人脸识别技术的相关价值还有很大的挖掘空间。

## 2. 项目总结

本文采用了两种方法对人脸识别进行了研究，分别是基于 PCA 的特征脸，基于 Opencv 的 LBPH。其主要内容涉及到人脸图像的预处理，两种算法的原理介绍以及最后进行实验验证。特征脸方法提供了一种可行的人脸识别的解决方法，这种方法相对简单，在背景受约束的条件下表现良好。然而，在人脸识别中，图像的姿态、表情、背景、装饰品和灯光等因素对识别效果都有很大影响，同时单一的识别算法识别率并不高，因此多种方法结合是今后着重改进的部分。

通过对不同方法的人脸识别的方法的探索和练习，初步学会了运用相关软件（python, pycharm, jupyter 等）来进行一定程度的编程，通过对不同 python 函数库的调用实践，提高了运用 python 等程序代码解决实际问题的能力。深入了解了人脸识别的几种方法和其操作步骤，并能自己还原人脸识别的实际过程。在这些过程中锻炼了自己阅读外国文献、操作相关软件的能力，初步了解到人脸识别技术的冰山一角，获益良多。

我与小组的其他三位成员在本课程中认真探索，深入学习，每个人都能够对人脸识别算法进行实践并取得成功，将课堂上学到的知识在项目中得到实践，既巩固了知识，又提高了运用知识解决实际问题的能力。在实践过程中，我们也遇到了一些困难，通过搜索博主的成功案例，小组成员之间互帮互助，最终取得了成功。

## 参考文献及网站

- [1] 俞王新. 计算机人脸检测与识别方法的研究. 2009.05
- [2] 黄菲菲. 基于 LBP 的人脸识别研究. 2005.05.27
- [3] Thrk and Pentland.Eigenfaees for Reecognition. [J]. Cognitive Neuroscience J.1991, 3(1):71 —86.
- [4]opencv 论坛 <https://forum.opencv.org/>
- [5] python 文档目录 <https://docs.python.org/zh-cn/3/contents.html>
- [6] T. Ojala, M. Pietikäinen, and D. Harwood (1996), “A Comparative Study of Texture Measures with Classification Based on Feature Distributions”, Pattern Recognition, vol. 29, pp. 51-59.
- [7] Ahonen, T., Hadid, A., and Pietikainen, M. Face Recognition with Local Binary Patterns. Computer Vision- ECCV 2004 (2004), 469–481.
- [8] Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang and Stan Z. Li. Learning Multi-scale Block Local Binary Patterns for Face Recognition. International Conference on Biometrics (ICB), 2007, pp. 828-837.
- [9] 基于 OpenCV 的人脸识别算法研究与实现 冯婧 , 顾梅花 , FENG Jing , GU Mei-hua - 《电脑知识与技术》 - 2020 年 14 期
- [10] 基于 OpenCV 的二维 PCA 人脸识别算法研究与实现 李德福 , 2017 - 桂林电子科技大学 : 控制科学与工程

## 致谢

感谢彭建军老师和何社阳老师在此课程项目中给予的指导，感谢众多博主对知识的无私分享，同时感谢小组成员的努力与帮助。自知仍有许多地方值得改进，在未来，我将继续学习，继续努力，不断丰富处理实际问题的经验，提高运用程序和所学知识解决实际问题的能力，为将来走入社会实现自我打好基础。