程序报告

学号：　　姓名：

一、问题重述

（简单描述对问题的理解，从问题中抓住主干，必填）

==========

本问题主要分成三个部分：其一是训练特征脸算法的实现，生成特征脸映射矩阵，其二是将人脸向特征人脸中投影，得到特征人脸；其三是重建人脸。

二、设计思想

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

==========

根据上课时讲到的主成分分析算法，特征脸映射矩阵其实就是人脸矩阵的协方差矩阵的特征向量矩阵，而协方差矩阵经过中心化操作之后就是$\boldsymbol{\phi^T\phi}$，但是这个矩阵规模过大，这里使用奇异值分解的方法来减低计算量。根据 SVD 公式得到，先易得$\boldsymbol{\phi\phi^T}$的特征向量$\boldsymbol{U}$即左奇异值，然后乘以$\boldsymbol{\phi^T}$即得到右奇异值$\boldsymbol{V}$即是我们需要的特征人脸映射矩阵。

要得到特征人脸矩阵，只要将人脸矩阵乘以特征人脸映射矩阵即可。

要重建人脸，由于$\boldsymbol{U}$是正交矩阵，所以只需要将特征人脸矩阵再乘以$\boldsymbol{U^T}$即可。

三、代码内容

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

==========

首先是计算特征人脸映射矩阵的算法：

```
avg_img = np.mean(trainset, axis = 0)
norm_img = trainset - avg_img
w, v = np.linalg.eig(norm_img @ norm_img.T)
v = v[:, 0:k]
v = norm_img.T @ v
v_norm = np.linalg.norm(v, ord = 2, axis = 0, keepdims = True).repeat(v.shape[0], axis = 0)
v = v / v_norm
feature = v.T
```

笔者首先用 numpy 的 mean 方法得到平均人脸；然后将原图像减去平均人脸得到中心化之后的人脸 norm_img 也就是上面所述的ϕ；然后求得$\phi\phi^T$的特征向量，主要是调用 numpy 是 linalg.eig 函数得到；接下来取特征向量的前 k 个，也就得到左奇异矩阵U；然后乘以ϕ^T得到右奇异矩阵V，最后得到V二范数矩阵 v_norm，然后归一化得到V。笔者发现为使特征脸正常输出，feature 为V的转置。

==========

接下来是计算特征人脸的算法：

```
numEigenFaces = numComponents
representation = (image - avg_img) @ (eigenface_vects.T[:, 0:numEigenFaces])
```

这个算法比较简单，就是中心化 image，然后乘以特征映射矩阵的前 numEigenFaces 个特征

向量即可。

==========

接下来是重建人脸的算法：

```
face = representations @ (eigenVectors.T[:, 0: numComponents]).T
face = face + avg_img
face = face.reshape(sz)
```

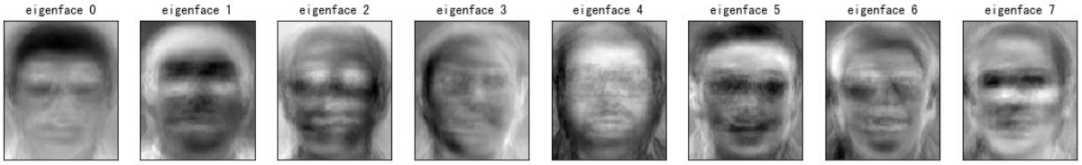
首先将特征人脸乘以映射矩阵前 numComponents 个特征向量的转置，即得到了原来空间中经过中心化的人脸，然后加上 avg_img 即可得到原图像，最后 reshape 成原图格式即可。

四、实验结果

（实验结果，必填）

==========

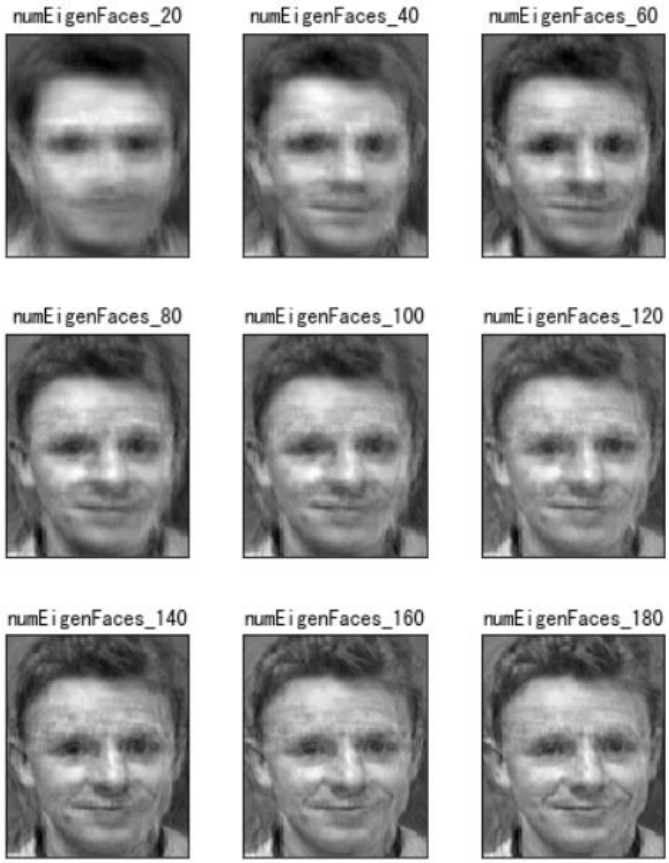
1. 前 8 个特征人脸如下

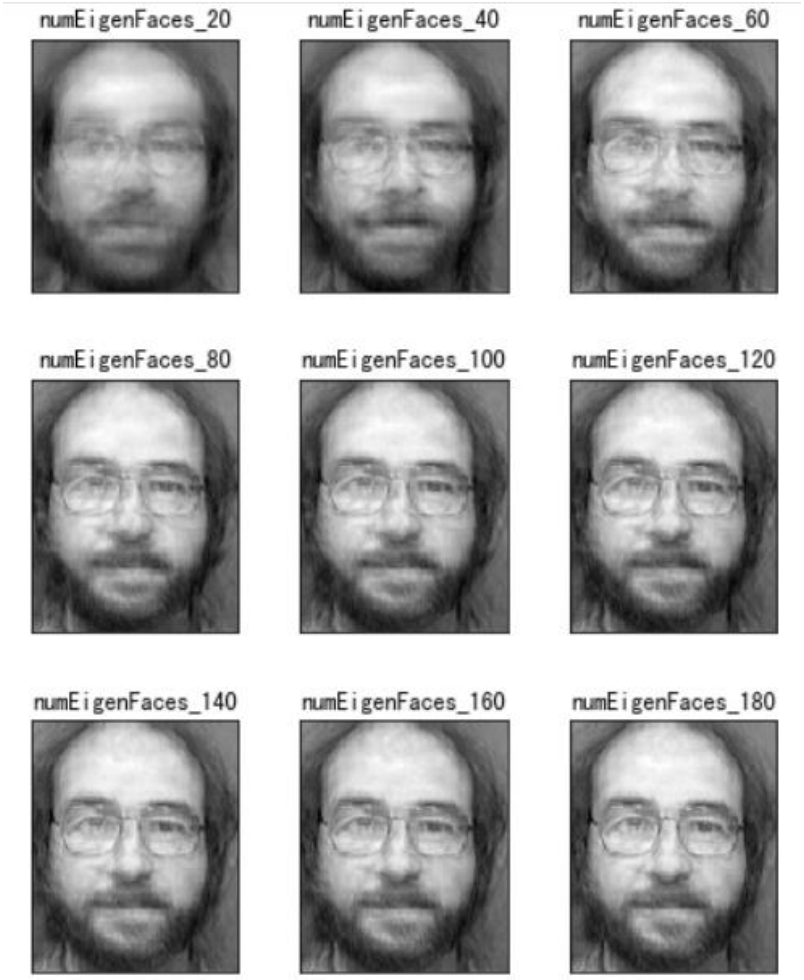


2. 人脸识别准确率

人脸识别准确率: 91.25%

3. 人脸重建效果





4. 最终结果

五、总结

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

==========

本次实验基本达到了预期目标，在做这个问题的时候出现了特征人脸看不出来的问题，后来发现是因为没有将特征人脸映射矩阵转置所致。