**程序报告**

学号：2110508 姓名：杨冰雪

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

====================================================================

特征脸算法（Eigenfaces Algorithm）是一种用于人脸识别和人脸表征的技术，它基于主成分分析（PCA）的思想。特征脸算法通过将人脸图像投影到一个高维特征空间中，并在该空间中分析人脸图像的主要变化方向，从而实现人脸识别和表征。

1. **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**）

====================================================================

本实验采用特征脸算法进行人脸识别。

①算法先根据测试数据求出平均脸

②再计算训练数据中心每张脸与平均脸的差异，求出差异矩阵的特征值和特征向量

③提取前k个特征向量，计算出k张特征脸，然后利用k个特征脸对测试人脸进行重建

1. **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）

====================================================================

def eigen\_train(trainset, k=20):

"""

训练特征脸（eigenface）算法的实现

:param trainset: 使用 get\_images 函数得到的处理好的人脸数据训练集

:param K: 希望提取的主特征数

:return: 训练数据的平均脸, 特征脸向量, 中心化训练数据

"""

avg\_img = np.sum(trainset, axis=0)

# 获得平均人脸

avg\_img = avg\_img/trainset.shape[0]

# 获得差值矩阵

differential\_matrix = trainset - avg\_img

norm\_img = differential\_matrix

differential\_matrix\_t = differential\_matrix.T

covariance\_matrix = np.mat(differential\_matrix) \* np.mat(differential\_matrix\_t)

eigenvalue, featurevector = np.linalg.eig(covariance\_matrix)

# 确保k的取值不会超限

k\_min = min(featurevector.shape[1], k)

feature\_vector = np.mat(differential\_matrix\_t) \* np.mat(featurevector[:,:k\_min])

feature = np.mat(differential\_matrix\_t) \* np.mat(differential\_matrix) \* np.mat(feature\_vector)

feature = np.array(feature).T

# 返回：平均人脸、特征人脸、中心化人脸

return avg\_img, feature, norm\_img

====================================================================

def rep\_face(image, avg\_img, eigenface\_vects, numComponents = 0):

"""

用特征脸（eigenface）算法对输入数据进行投影映射，得到使用特征脸向量表示的数据

:param image: 输入数据

:param avg\_img: 训练集的平均人脸数据

:param eigenface\_vects: 特征脸向量

:param numComponents: 选用的特征脸数量

:return: 输入数据的特征向量表示, 最终使用的特征脸数量

"""

# 计算偏差矩阵

diff\_image = np.array(image) - np.array(avg\_img)

# 选取特定个特征脸数量

eigenface\_vects=np.array(eigenface\_vects[0:numComponents, :])

# 进行归一化

eigenface\_vect = normalize(eigenface\_vects, norm = 'l2')

# 将输入图像投影到特征空间中

representation=np.matrix(diff\_image)\*np.matrix(eigenface\_vect.transpose())

numEigenFaces = numComponents

# 返回：输入数据的特征向量表示, 特征脸使用数量

return representation, numEigenFaces

====================================================================

def recFace(representations, avg\_img, eigenVectors, numComponents, sz=(112,92)):

"""

利用特征人脸重建原始人脸

:param representations: 表征数据

:param avg\_img: 训练集的平均人脸数据

:param eigenface\_vects: 特征脸向量

:param numComponents: 选用的特征脸数量

:param sz: 原始图片大小

:return: 重建人脸, str 使用的特征人脸数量

"""

# 选取特定数量的特征脸向量

eigenVectors=np.array(eigenVectors[0:numComponents,:])

# 进行归一化

eigenface\_vect = normalize(eigenVectors, norm = 'l2')

# 计算特征空间基向量的内积

C=np.matrix(eigenface\_vect.transpose())\*np.matrix(eigenface\_vect)

C\_inversion = np.linalg.inv(np.matrix(C))

face = np.matrix(representations) \* np.matrix(eigenface\_vect)\* np.matrix(C\_inversion)

# 得到最终的差值图像

face = np.array(face) + np.array(avg\_img)

# 将图像调整为指定的大小

face = letterbox\_image(face, sz)

# 返回: 重建人脸, str 使用的特征人脸数量

return face, 'numEigenFaces\_{}'.format(numComponents)

1. **实验结果**

（实验结果，必填）

====================================================================



1. **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**）

====================================================================

成功达到实验预期结果，但实验也存在一些改进之处，可以基于ORL人脸库添加一些自己搜集到的人脸图像，从而形成一个更大的人脸库，更有利于模型的训练。