专业：工程力学
姓名：唐栎
学号：3190100827
日期：2021.12.10

浙江大学 实验报告

课程名称：计算机视觉 指导老师：宋明黎 成绩：
实验名称：Eigenface 人脸识别

一 实验目的和要求

1.1 实验目的

熟悉 OpenCV 的使用熟悉 Eigenface 人脸识别算法

1.2 实验要求

自己写代码实现 Eigenface 人脸识别的训练与识别过程

二 实验内容和原理

实验内容如实验要求所述

2.1 实验原理

实验采用人脸识别的特征脸（EigenFaces）方法，主要的原理为 PCA 降维

2.1.1 Principal Component Analysis(PCA)

PCA（Principal Component Analysis）是一种常见的数据分析方式，常用于高维数据的降维，可用于提取数据的主要特征分量。通过一系列的数学推导（从最大可分性出发），利用拉格朗日乘子法可以证明：PCA 本质上是对样本的协方差矩阵进行特征值分解。根据能量百分比，可以选取最大的 K 个特征向量组成特征子空间的一组基，将样本都投影到该子空间进行分类和识别

三 实验主要步骤与代码分析

3.1 训练

3.1.1 主函数

```
#main function
def main(argv):
    if not os.path.exists(argv[2]):
        os.makedirs(argv[2])
    path='./'+argv[3]+'/'#Get the path to the folder where the photos are stored
    energyRatio=float(argv[1])
    #filelist=os.listdir(path)#Get a list of photo file names
    img_size=(90,90)
    #read images
    ImageMatrix,count,Class=readImage(path,img_size)
    np.savetxt('./'+argv[2]+'/Class.csv', Class, delimiter = ',')
    #get meanFace
    mean_vector=meanFace(ImageMatrix,count)
```

```
np.savetxt('./'+argv[2]+'/mean.csv', mean_vector, delimiter = ',')
#compute the covarianceMatrix
matrix_prime,S=covarianceMatrix(ImageMatrix,mean_vector,count)
np.savetxt('./'+argv[2]+'/matrix_prime.csv', matrix_prime, delimiter = ',')
#compute the eigenvalues and eigenvectors
eigenvalues,eigenvectors = np.linalg.eig(S)
eigenvalues=eigenvalues.reshape(1,-1)
eigenvectors=np.dot(matrix_prime,eigenvectors)
#get transformation matrix
W=getPrincipal(eigenvectors,eigenvalues,energyRatio)
np.savetxt('./'+argv[2]+'/PCA.csv', W, delimiter = ',')
#show and write the mean image of the first 10 eigen faces
EigenFaces=W[:,0:10].sum(1)/10
EigenFaces=(EigenFaces.reshape(img_size))
EigenFaces=cv.normalize(EigenFaces,None,0,255,cv.NORM_MINMAX,cv.CV_8UC1)
cv.imwrite('EigenFaces.png',EigenFaces)
cv.imshow('first 10 eigen faces',EigenFaces)
cv.waitKey(0)
return 0
```

3.1.2 读入图像

```
def readImage(path,size):
width=size[0]
height=size[1]
Matrix_X=np.empty((height*width,400),np.uint8)
Class=np.empty((1,400),np.uint8)
#print(Matrix_X[:,0:1].shape)
count=0
for i in range(1,41,1):
    cur_path=path+str(i)+'/'
    filelist=os.listdir(cur_path)
    for item in filelist:
        if item.endswith('.png'):
            if item[0:2]=='10':
            #use the last img as test
                if i<10:
                    string='0'+str(i)
                else:
                    string=str(i)
                if not os.path.exists("testData"):
                    os.makedirs("testData")
                img=cv.imread(cur_path+'10.png',cv.IMREAD_GRAYSCALE)
                cv.imwrite('./testData/'+string+'_'+'10.png',img)
            else:
                #read a image
                img=cv.imread(cur_path+item,cv.IMREAD_GRAYSCALE)
                re_img=cv.resize(img,size,0,0,interpolation = cv.INTER_CUBIC)
                #histogram equalization
                equ_img = cv.equalizeHist(re_img)
                #cv.imshow('1',equ_img)
                final_img = cv.normalize(equ_img,None,0,255,cv.NORM_MINMAX,cv.CV_8UC1)#TODO chec
                #reshape the image to a vector
                vector_x=final_img.reshape((-1,1))#note -1 means let python calculate
                Matrix_X[:,count:count+1] =vector_x
                Class[0,count]=i
                count+=1
return Matrix_X[:,0:count],count,Class[0,0:count]
```

3.1.3 获得平均脸

```
def meanFace(matrix,count):
mean=np.zeros((matrix.shape[0],1))
mean=matrix.sum(1).reshape(-1,1)
mean=mean/count
return mean
```

3.1.4 求协方差矩阵的特征值和特征向量

由于协方差矩阵在本问题中过于庞大（10000*10000），其特征值分解变得极为困难，因此这里采用一个小技巧

假设样本（已经减去均值）为

$$X = \{x_1, x_2, \ldots, x_n\} \tag{1}$$

其中

$$x_i \in R^d \tag{2}$$

样本的协方差矩阵为

$$S = \frac{1}{n-1} X X^T \tag{3}$$

为求得协方差矩阵的特征值和特征向量，转而求以下矩阵的特征值和特征向量

$$S' = \frac{1}{n-1} X^T X \tag{4}$$

S' 的特征向量和特征值

$$\frac{1}{n-1} X^T X v_i = \frac{1}{n-1} \lambda_i v_i \tag{5}$$

两边左乘 X

$$X X^T (X v_i) = \lambda_i (X v_i) \tag{6}$$

因此，S' 的特征值与 S 的特征值相同，S' 的特征向量左乘 X 即可得到 S 的特征向量以下的函数求 S'

```
#compute the covarianceMatrix
def covarianceMatrix(matrix,mean,count):
    matrix_prime=matrix-mean
    S=np.dot((matrix_prime.T),matrix_prime)/count
return matrix_prime,S
```

在主函数的对应部分：

```
#compute the  fake_covarianceMatrix
matrix_prime,S=covarianceMatrix(ImageMatrix,mean_vector,count)
np.savetxt('./'+argv[2]+'/matrix_prime.csv', matrix_prime, delimiter = ',')
#compute the eigenvalues and eigenvectors
eigenvalues,eigenvectors = np.linalg.eig(S)
eigenvalues=eigenvalues.reshape(1,-1)
eigenvectors=np.dot(matrix_prime,eigenvectors)
```

3.1.5 求转换矩阵

转换矩阵的求解：

```
def getPrincipal(eigenvectors,eigenvalues,energyRatio):
eigenvalues=eigenvalues.reshape(1,-1)
#sort and get index
sorted_index=np.argsort(-eigenvalues)
totalSum=np.sum(eigenvalues)
thisSum=0
k=0# number of principal components selected

for i in range(0,sorted_index.shape[1]):
    index=sorted_index[0,i]
```

```
        thisSum+=eigenvalues[0,index]
        k+=1
        #find the least k
        if thisSum/totalSum >= energyRatio:
            break
    #select k components
    sorted_index=sorted_index[0,0:k].reshape(1,-1)
    W=np.empty((eigenvectors.shape[0],k))
    j=0
    for i in range(0,sorted_index.shape[1]):
        index=sorted_index[0,i]
        #normalize the eigenvectors
        W[:,j:j+1]=eigenvectors[:,index:index+1]/np.sqrt(np.sum(eigenvectors[:,index:index+1]**2))
        j+=1
    #return the transformation matrix
    return W
```

在主函数的对应部分：

```
    #get transformation matrix
    W=getPrincipal(eigenvectors,eigenvalues,energyRatio)
    np.savetxt('./'+argv[2]+'/PCA.csv', W, delimiter = ',')
```

3.2 识别

3.2.1 主函数及其调用

主函数：

```
    def main(argv,filename):
    img_size=(90,90)
    model_path='./'+argv[2]+'/'
    mean,model_matrix,matrix_prime,Class=inputModel(model_path)
    #read test img
    test_img=cv.imread('./'+sys.argv[1]+'/'+filename,cv.IMREAD_GRAYSCALE)
    test_img=cv.resize(test_img,img_size,0,0,interpolation = cv.INTER_CUBIC)
    test_vector=getImgVector(test_img)
    #Projecting the test image into the PCA subspace
    test_projected=PCA_project(mean,model_matrix,test_vector)
    #project training img
    train_projectedM=np.dot(model_matrix.T,matrix_prime)

    #findNearest
    minClass,bestImgV=findNearest(test_projected,Class,train_projectedM)
    best_img_reconstruct=PCA_reconstruct(mean,model_matrix,bestImgV)
    best_img=(best_img_reconstruct.reshape(img_size))
    best_img=cv.normalize(best_img,None,0,255,cv.NORM_MINMAX,cv.CV_8UC1)
    #show the best img
    cv.namedWindow('Best match',cv.WINDOW_AUTOSIZE)
    cv.imshow('Best match',best_img)
    #show the result
    result_img=test_img
    font = cv.FONT_HERSHEY_PLAIN
    #put the recognition result on the result_img
    cv.putText(result_img,str(int(minClass)),(50,70), font, 2,(0,0,255),2,cv.LINE_AA)
    cv.namedWindow('recognition result',cv.WINDOW_AUTOSIZE)
    if not os.path.exists("testResult"):
        os.makedirs("testResult")
    cv.imwrite('./testResult'+'/'+filename,result_img)
    cv.imshow('recognition result',result_img)
    cv.waitKey(0)
```

```
    #return the flag to show if the result is correct
    if int(filename[0:2])==int(minClass):
        return 1
    else:
        return 0
```

主函数的调用：

```
    if __name__ == "__main__":
    testpath='./'+sys.argv[1]+'/'
    filelist=os.listdir(testpath)
    length=len(filelist)
    lst=[]
    for filename in filelist:
        lst.append(main(sys.argv,filename))
    #print the accuracy
    print(sum(lst)/length)
    #Command line input:
    #argv[0] for the file name, e.g. mytest.py
    #argv[1] for the test folder
    #argv[2] for the model file folder
    # python mytest.py testData modelData
```

3.2.2 读入模型文件

inputModel 函数

```
    def inputModel(model_path):
    #input mean.csv
    try:
        with open(model_path+'mean.csv', "rb") as f:
            mean= np.loadtxt(f, delimiter = ",", skiprows = 0)
    except IOError:
        return None
    #input PCA.csv
    try:
        with open(model_path+'PCA.csv', "rb") as f:
            model_matrix= np.loadtxt(f, delimiter = ",", skiprows = 0)
    except IOError:
        return None
    #input matrix_prime.csv
    try:
        with open(model_path+'matrix_prime.csv', "rb") as f:
            matrix_prime= np.loadtxt(f, delimiter = ",", skiprows = 0)
    except IOError:
        return None
    #input Class.csv
    try:
        with open(model_path+'Class.csv', "rb") as f:
            Class= np.loadtxt(f, delimiter = ",", skiprows = 0)
    except IOError:
        return None
    #if all read,return them
    return mean,model_matrix,matrix_prime,Class
```

在主函数中对应的部分：

```
    model_path='./'+argv[2]+'/'
    mean,model_matrix,matrix_prime,Class=inputModel(model_path)
```

3.2.3 读入待识别的图像并投影到特征子空间

getImgVector 函数将图像预处理并拉平为向量：

```
def getImgVector(img):
equ_img = cv.equalizeHist(img)
final_img = cv.normalize(equ_img,None,0,255,cv.NORM_MINMAX,cv.CV_8UC1)
vector_x=final_img.reshape((-1,1))
return vector_x
```

PCA-project 函数将测试图像投影至 PCA 子空间

```
def PCA_project(mean,model_matrix,x):
mean=mean.reshape(-1,1)
y=np.dot(model_matrix.T,x-mean)
return y
```

在主函数中对应的部分：

```
#read test img
test_img=cv.imread('./'+sys.argv[1]+'/'+filename,cv.IMREAD_GRAYSCALE)
test_img=cv.resize(test_img,img_size,0,0,interpolation = cv.INTER_CUBIC)
test_vector=getImgVector(test_img)
#Projecting the test image into the PCA subspace
test_projected=PCA_project(mean,model_matrix,test_vector)
```

3.2.4 找到与测试图像最相近的训练图像并获得类别

函数 findNearest：

```
def findNearest(test_projected,Class,train_projectedM):
Class=Class.reshape((-1,1))
minDist=float("inf")
#get the best match image and its class
for i in range(0,train_projectedM.shape[1]):
    #compute the distance between the test vector and the training vector
    Dist=Euclid_Distance(test_projected,train_projectedM[:,i:i+1])
    if Dist < minDist:
        minDist=Dist
        minClass=Class[i,0]
        bestImgV= train_projectedM[:,i:i+1]
return minClass,bestImgV
```

函数 EuclidDistance 计算欧几里得距离：

```
def Euclid_Distance(v1,v2): #compute the Euclid distance between 2 vectors
v1=v1.reshape(-1,1)
v2=v2.reshape(-1,1)
Dist=np.sqrt(np.sum((v1-v2)**2))
return Dist
```

在主函数中对应的部分：

```
#findNearest
minClass,bestImgV=findNearest(test_projected,Class,train_projectedM)
```

3.2.5 输出最相近图像和识别结果

在主函数中对应的部分：

```
#reconstruct the best image
best_img_reconstruct=PCA_reconstruct(mean,model_matrix,bestImgV)
best_img=(best_img_reconstruct.reshape(img_size))
best_img=cv.normalize(best_img,None,0,255,cv.NORM_MINMAX,cv.CV_8UC1)
```

```
#show the best img
cv.namedWindow('Best match',cv.WINDOW_AUTOSIZE)
cv.imshow('Best match',best_img)
#show the result
result_img=test_img
font = cv.FONT_HERSHEY_PLAIN
#put the recognition result on the result_img
cv.putText(result_img,str(int(minClass)),(50,70), font, 2,(0,0,255),2,cv.LINE_AA)
cv.namedWindow('recognition result',cv.WINDOW_AUTOSIZE)
if not os.path.exists("testResult"):
    os.makedirs("testResult")
cv.imwrite('./testResult'+'/'+filename,result_img)
cv.imshow('recognition result',result_img)
cv.waitKey(0)
#return the flag to show if the result is correct
if int(filename[0:2])==int(minClass):
    return 1
else:
    return 0
```

四　实验环境及运行方法

4.1　实验环境

Python 3.8.12 + OpenCV 4.5.4

4.2　运行方法

4.2.1　模型训练

在根目录打开 cmd 命令行，输入 python mytrain.py 0.95 modelData data

1. argv[0] for the file name, e.g. mytrain.py
2. argv[1] for the energyRatio
3. argv[2] for the model folder's name
4. argv[3] for the name of the data folder

4.2.2　人脸识别

在根目录打开 cmd 命令行，输入 python mytest.py testData modelData

1. argv[0] for the file name, e.g. mytest.py
2. argv[1] for the test folder
3. argv[2] for the model file folder

五　实验结果展示

5.1　训练过程

5.1.1　输入展示

输入的图像存放在当前目录下的“Data”文件夹中，Data 文件夹中有 40 个子文件夹，每个子文件夹中的 10 张图像属于同一个类别。假设每张人脸图像只有一张人脸，并且已经裁剪到相同的合适大小

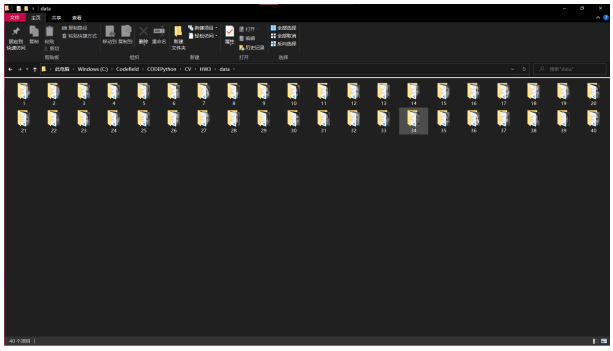

图 1: 训练的人脸库

5.1.2　输出展示

训练程序在当前目录下的“modelData”文件夹下，输出四个 csv 文件

1. Class.csv 存放每张训练图片的类别
2. matrix-prime.csv 存放每张训练图片的信息（每一列是一张图片对应的向量，并且已经减去了整体的平均脸）
3. mean.csv 存放的是平均脸对应的列向量
4. PCA.csv 存放的是 PCA 降维中的投影矩阵（每一个列向量就是一个归一化的特征向量）

此外，训练过程同时会将前 10 个特征脸平均成一张图像，然后显示出来。（并保存为“EigenFaces.png”）同时，由

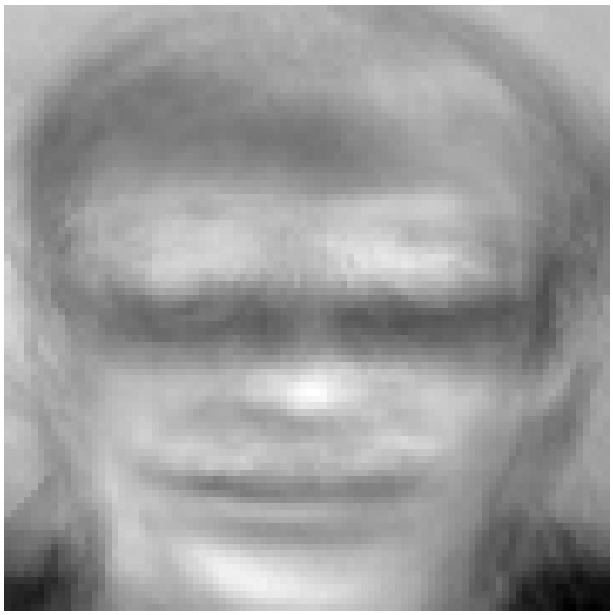

图 2: 前十张特征脸的平均脸

于每个类别的最后一张图片（编号为 10）会用作最终的测试集，在目录下会将这些测试图像保存到“testData”文件夹下

5.2 识别过程

5.2.1 输入展示

识别过程的输入即训练过程输出的“modelData”和“testData”两个文件夹中的所有文件

5.2.2 输出展示

输出的图像存放在当前目录下的“testResult”文件夹中，文件名和“testData”中的一一对应。同时这些输出图像都会标上程序识别得到的类别（数字），最终会打印出识别的准确率经测试发现识别的准确率为 0.9（40 张测试图片中识别正确了 36 张）

```
(CV) C:\Codefield\CODEPython\CV\HW3>python mytest.py testData
 modelData
0.9
```

图 3: 识别准确率

同时会显示识别结果和最相近的图像，下面以其中一张为例：

图 4: 识别结果

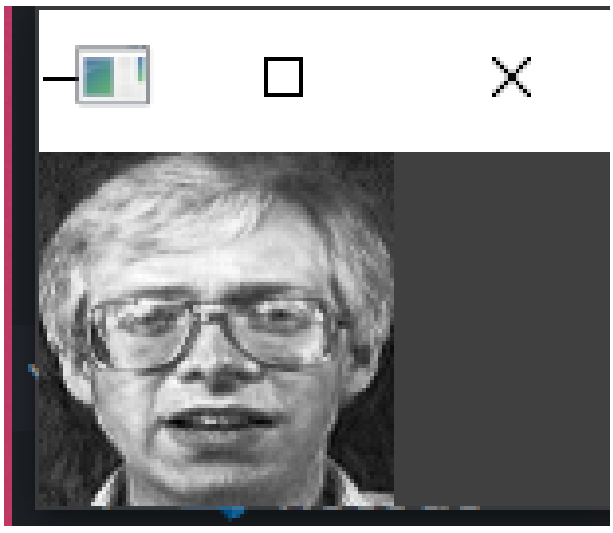

图 5: 最相近的图像

六　心得体会

通过本次的实验，我对主成分分析算法 (PCA)、图像处理的知识有了深入的理解和体会。动手搭建了一个人脸识别系统，但是本次实验也让我体会到了 EigenFace 人脸识别的巧妙。