Face Recognition

1．算法描述

Principal Components Analysis (PCA)

定义：

N个向量xi，每个向量有d个元素，用列向量表示，每个xi都代表一个训练集图片。N就是训练集图片数量，40个人，每个人7张训练集图片，N=7\*40=280。d是每张图片的像素数量，d= 92x112。

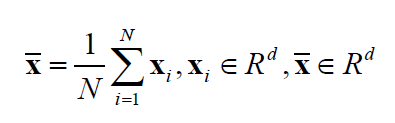
xiRd

我们的目标是，将原向量从d维空间投影到k维空间，称为降维，我们使用PCA降维。

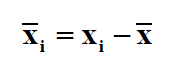
PCA算法的步骤

训练阶段。

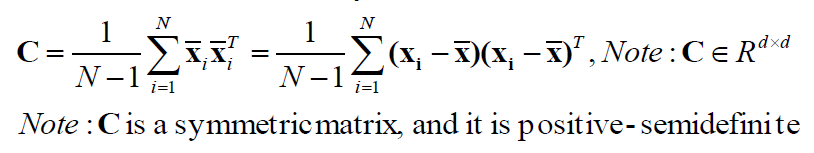
1. 计算表示训练集图片像素的列向量的均值，就是计算出xi的均值。



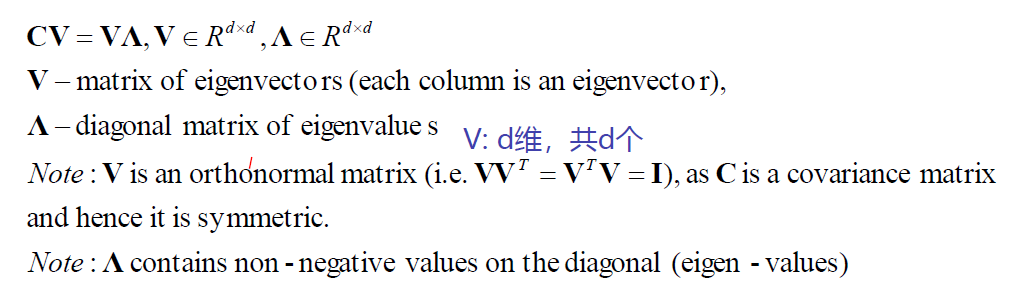
1. 每个列向量xi都减去均值。



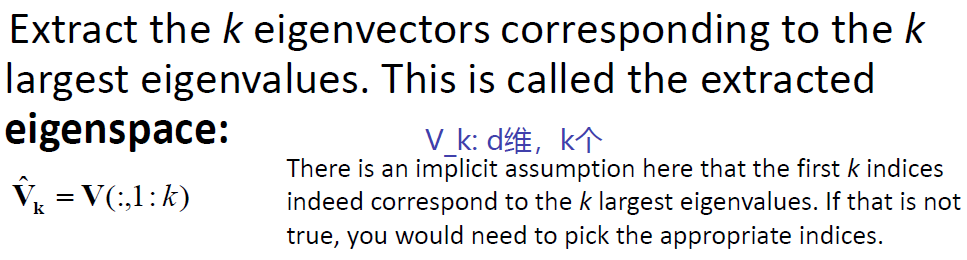
1. 计算步骤2得到的向量的协方差矩阵C。我们求的是协方差的无偏估计，由于减去了均值，故除以(N-1)。



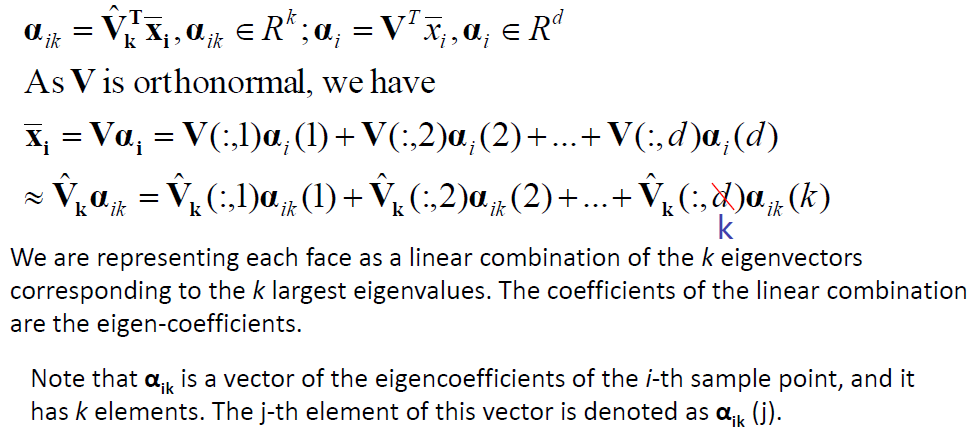
1. 求协方差矩阵C的特征向量。V是特征向量矩阵，每一列都是一个特征向量。



1. 根据协方差矩阵C的特征值的大小，提取最大的k个特征值对应的k个特征向量，称为eigenspace。



1. 将投影到eigenspace，对于每个，得到一个向量αik，αik有k个元素，每个元素称为eigen-coefficients。我们将表示成k个特征值最大的特征向量的线性组合，线性组合的系数就是eigen-coefficients。



1. 保存每个测试集图像的eigen-coefficient和此图像对应的人的身份到数据库中。保存到数据库中。

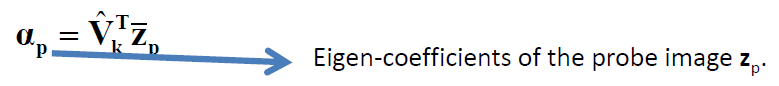
测试阶段。

测试阶段，以列向量的形式给你一个测试图像Zp，Zp有d个元素。（Zp和xi的形式一样）。

1. Zp减去训练阶段步骤1得到的均值。



1. 将投影到eigen-space，得到eigen-coefficients

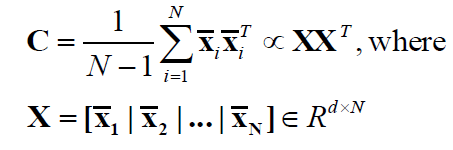


1. 将αp和αik对比，计算αp和αik的欧几里得距离（平方距离），采用2范数最小匹配，当欧几里得距离最小时，得到αik，αik对应的人，就是人脸识别匹配到的人。

PCA算法的优化

当N远小于d的时候，训练集图片数量远小于每张图片的像素数量时，有如下的优化方法。

考虑协方差矩阵C。



考虑矩阵，大小为N\*N，而不考虑矩阵，大小为d\*d。

的特征向量有如下形式。



等式两边左乘X，得XXT(Xw) = λ(Xw)，将Xw看做一个整体，就是Ax=λx的形式。

我们就得到了XXT的特征向量Xw，就是协方差矩阵C的特征向量。

这种方法计算协方差矩阵C的特征向量的时间复杂度，等于

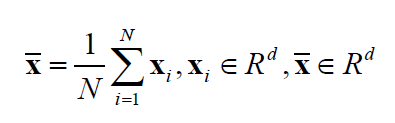
计算矩阵XTX的特征向量的时间复杂度O(N3)+对于每个w，计算Xw的时间复杂度O(N\*dN)

等于O(N3+dN2)，远远小于原来的时间复杂度O(d3)。

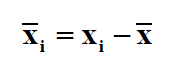
优化后的算法步骤

训练阶段

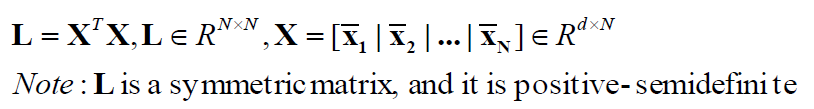
1. 计算表示训练集图片像素的列向量的均值，就是计算出xi的均值。



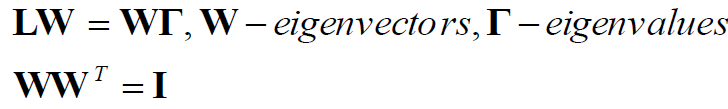
1. 每个列向量xi都减去均值。



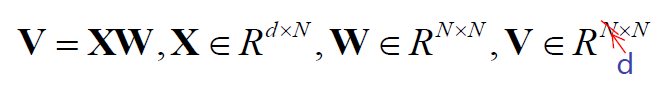
1. 计算矩阵L。



1. 计算L的特征向量矩阵W。



1. 根据W得到协方差矩阵C的特征向量。



1. 将V的列向量归一化。
2. 根据协方差矩阵C的特征值的大小，提取最大的k个特征值对应的k个特征向量，称为eigenspace。
3. 将投影到eigenspace，对于每个，得到一个向量αik，αik有k个元素，每个元素称为eigen-coefficients。我们将表示成k个特征值最大的特征向量的线性组合，线性组合的系数就是eigen-coefficients。
4. 保存每个测试集图像的eigen-coefficient和此图像对应的人的身份到数据库中。保存到数据库中。

测试阶段不变。

2. 代码

generate\_random\_sequence\_to\_define\_training\_and\_test\_images.m文件，里面有

generate\_random\_sequence\_to\_define\_training\_and\_test\_images函数。

对于每个人的10张图像，随机选择7张用来训练，另外3张用于测试。

生成1-10的不重复的随机数列，前7个数作为训练图像，后3个数作为测试图像。一共生成40个这样的数列。

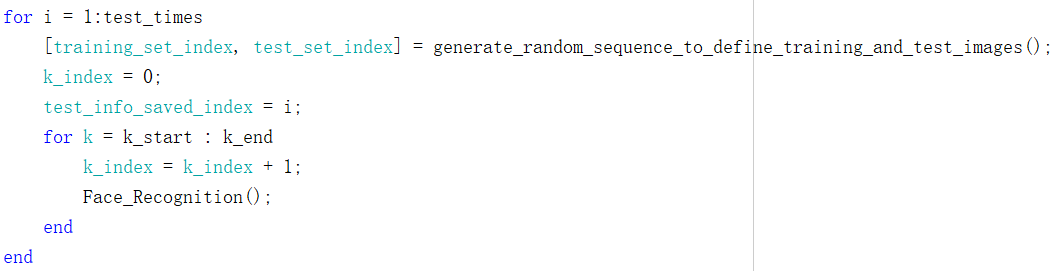
training\_set\_index是40\*7的矩阵。第i行表示第i个人的7个训练图像的索引。

例如，第5行是1 3 4 5 8 9 10，则第5个人的训练图像是图1 3 4 5 8 9 10。

test\_set\_index是40\*3的矩阵。第i行表示第i个人的3个测试图像的索引。

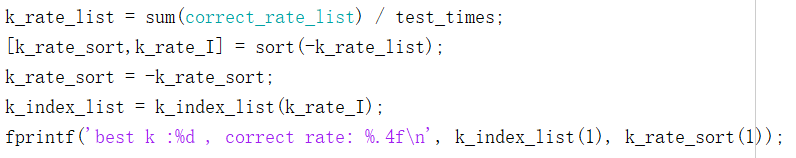


Face\_Recognition\_main.m文件是主文件。



总共测试test\_times轮。每一轮先生成训练集和测试集索引，然后对于[k\_start,k\_end] 区间内的每个k值都进行测试。

第i轮，提取的特征向量数量为k时，正确率保存在correct\_rate\_list(i,k)中。



sum(correct\_rate\_list) / test\_times得到，提取的特征向量数量为k时，test\_times次测试的平均正确率。排序后得到正确率最高的k。

correct\_rate\_list.xlsx保存了correct\_rate\_list的一个样例。

Face\_Recognition.m文件，实现了PCA优化后的算法。代码较长，只说关键步骤。

function [pic, pic\_identity, avg\_training\_pic] =

read\_training\_images\_and\_deduct\_mean\_from\_images()

此函数实现功能：读入训练集图片，求出像素均值，每个图片像素值都减去均值

定义：

m\*n是图片的像素。

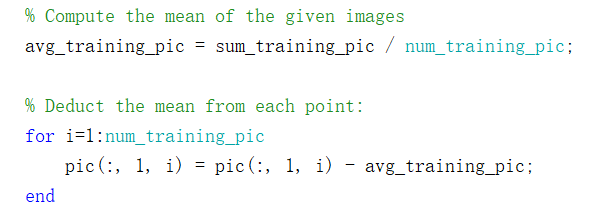
pic用来存放所有训练集图片 m\*n\*k

k是训练集图片数量，一个特定的k0表示m\*n大小的一张图片。

pic(:, i) 是列向量xi。

pic\_identity(i) 是第i个图片的身份信息。

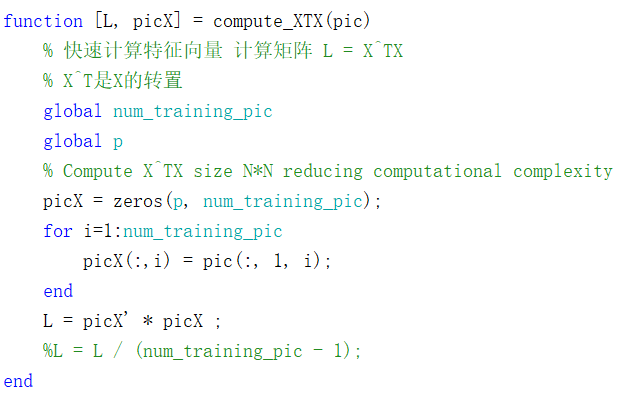
sum\_training\_pic 训练集图片的像素值之和。



function [L, picX] = compute\_XTX(pic)

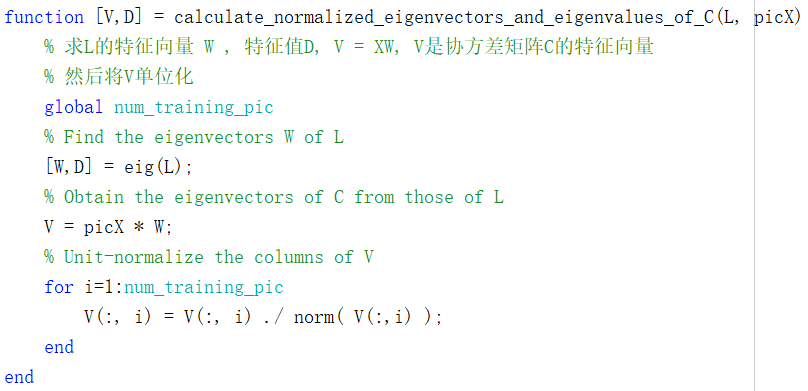
此函数实现功能：快速计算特征向量 计算矩阵 L = XTX。

定义：picX就是X。



function [V,D] = calculate\_normalized\_eigenvectors\_and\_eigenvalues\_of\_C(L, picX)

此函数实现功能：求L的特征向量 W , 特征值D, 根据W求协方差矩阵C的特征向量V，V = XW。然后将V单位化。

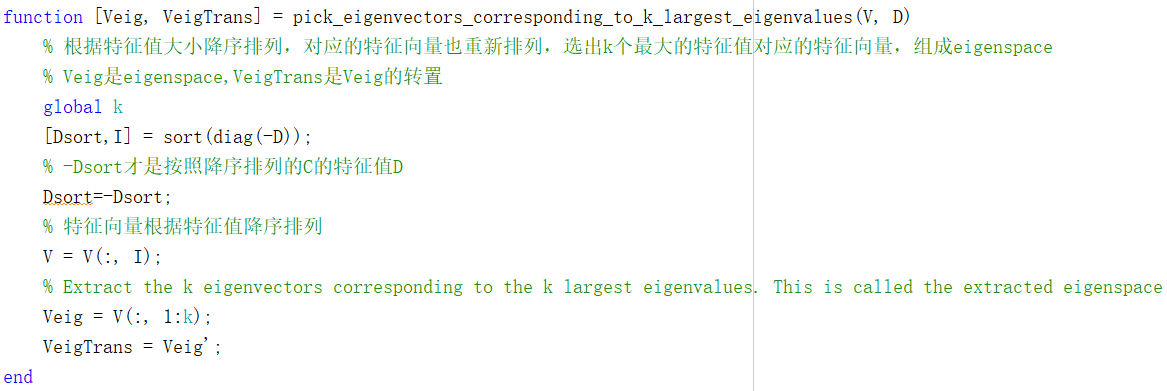


function [Veig, VeigTrans] =

pick\_eigenvectors\_corresponding\_to\_k\_largest\_eigenvalues(V, D)

此函数实现功能：根据特征值大小降序排列，对应的特征向量也重新排列，选出k个最大的特征值对应的特征向量，组成eigenspace

定义：Veig是eigenspace,VeigTrans是Veig的转置（方便计算）



function coefficients\_eig = project\_image\_onto\_eigenspace(VeigTrans, pic)

此函数实现功能：将投影到eigenspace。

定义：coefficients\_eig 是 eigen-coefficients。列向量i表示投影到eigenspace的结果αik。



function [result, correct\_rate\_info] =

test\_phase(pic\_identity, VeigTrans, coefficients\_eig, avg\_training\_pic)

此函数实现功能：

测试阶段，先将测试图像从n\*m矩阵变为p\*1列向量Zp p=n\*m

Zp减去训练阶段求出的图像各个坐标的平均像素值

映射Zp到eigenspace,求出Zp的eigen-coefficients

求Zp的eigen-coefficients和训练集图片的eigen-coefficients的

欧几里得距离的平方jp

jp最小值对应的训练集图片就是匹配图片。

定义：

num\_test\_face 提供图片的测试者数量

num\_testpic\_per\_face 每个测试者的测试图片数量

success\_match 成功匹配的数量 fail\_match 错误匹配的数量

sum\_diff 和 difference 是测试图像和训练图像的欧几里得距离的平方

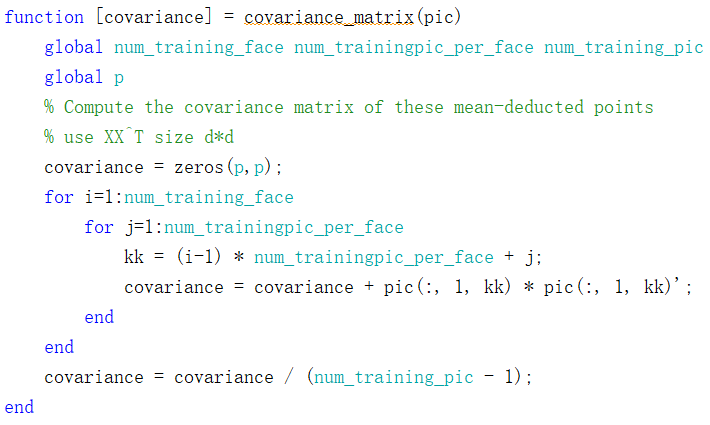
test\_index 测试图像下标

下面两个函数实现的是优化前的方法。

function [covariance] = covariance\_matrix(pic)

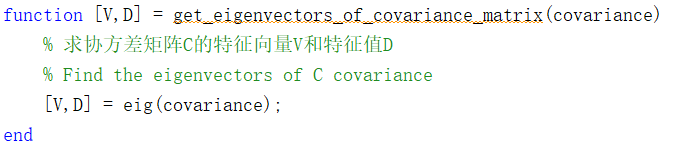
此函数实现功能：求协方差矩阵C

定义：covariance是协方差矩阵。



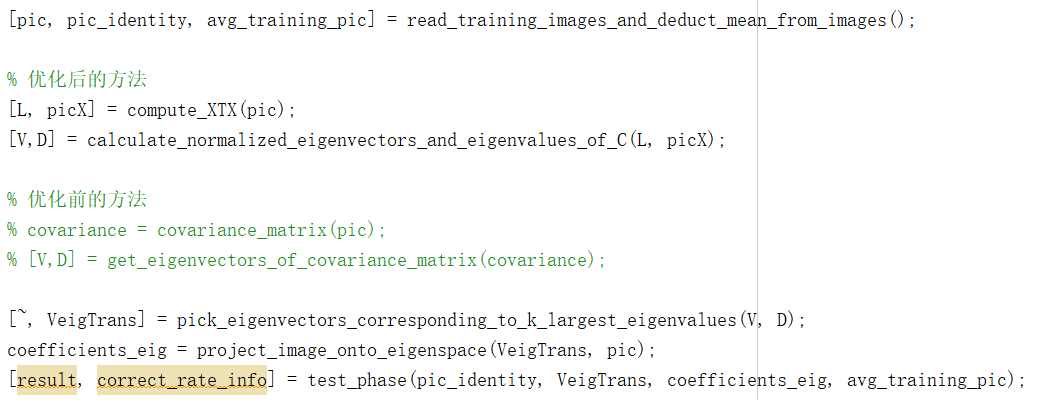
function [V,D] = get\_eigenvectors\_of\_covariance\_matrix(covariance)

此函数实现功能：求协方差矩阵C的特征向量V和特征值D



外层函数function [] = Face\_Recognition()

此函数实现功能：整个PCA优化后算法。如果想用优化前的方法，将优化后的方法后面两行加上注释，将优化前的方法后面两行的注释取消。



运行方法

打开Face\_Recognition\_main.m文件，F5运行。

可以调整的参数，test\_times测试轮数，测试k值的最小值k\_start，最大值k\_end。

效率对比。

k取50。

使用优化前的方法，完成一次测试花费的时间：319.7237s

由于优化前的方法时间复杂度较高，不重复测试求平均值。

其中，计算协方差矩阵C花费时间170.0099s。

计算协方差矩阵C的特征向量和特征值花费时间146.5489s。

说明优化前的方法，计算协方差矩阵C和计算C的特征向量、特征值花费大量时间。

使用优化后的方法，完成10次测试的总时间：10.9299s

平均一次测试花费的时间：1.0930s

完成每次测试花费的时间：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.2975 | 1.1279 | 1.1282 | 1.066 | 1.061 | 1.0551 | 1.0559 | 1.0793 | 1.0202 | 1.0388 |

可以看出，优化后的方法大大减少了时间复杂度，减少了花费的时间。

使用优化后的方法，完成10次测试的平均正确率：0.9775

每次测试的正确率：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.9583 | 0.9833 | 0.9667 | 0.9667 | 1 | 0.9833 | 0.975 | 0.975 | 0.9833 | 0.9833 |

可以看到，测试的正确率波动较大，训练集和测试集的选择对正确率的影响较大。甚至出现过正确率为1的情况。

选取最好的k，k=50最好，正确率是0.9658

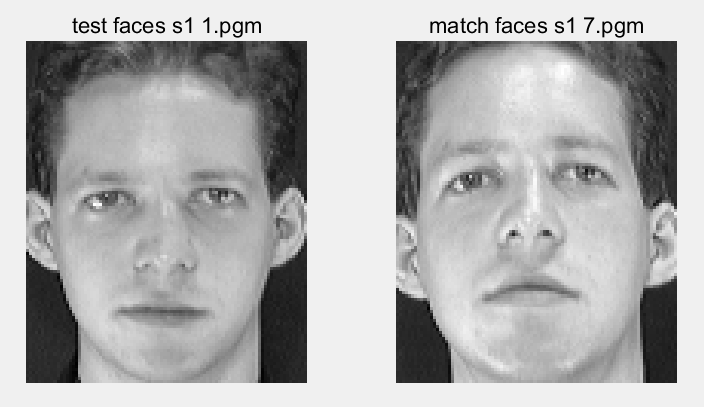
对于[50,100]之间的k，总共进行10轮测试，每轮测试，不同的k的训练集和测试集相同。

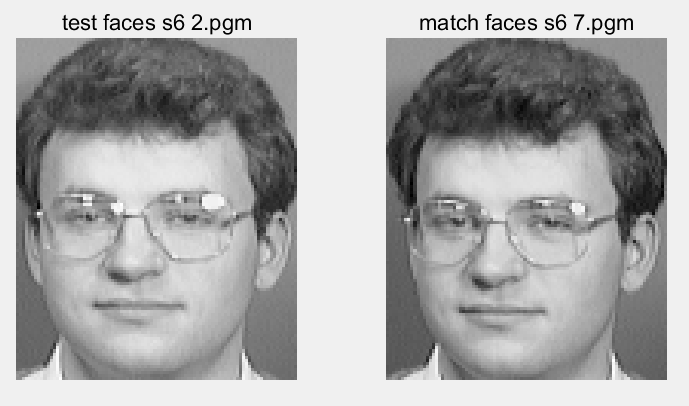
得到平均正确率，在correct\_rate\_list.xlsx文件中，由于结果太大，只展示[50,60]之间的k的平均正确率。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.9658 | 0.965 | 0.965 | 0.9633 | 0.9633 | 0.9633 | 0.9633 | 0.9633 | 0.9633 | 0.9641 | 0.9641 |

最好的k=50，正确率是 0.9658。

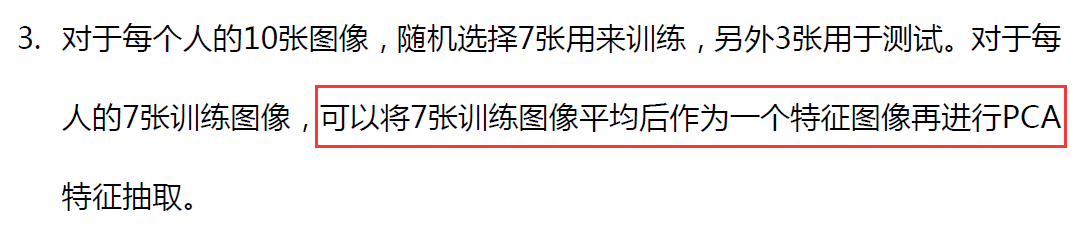
部分人脸识别结果（左图是测试图片，右图是匹配图片）



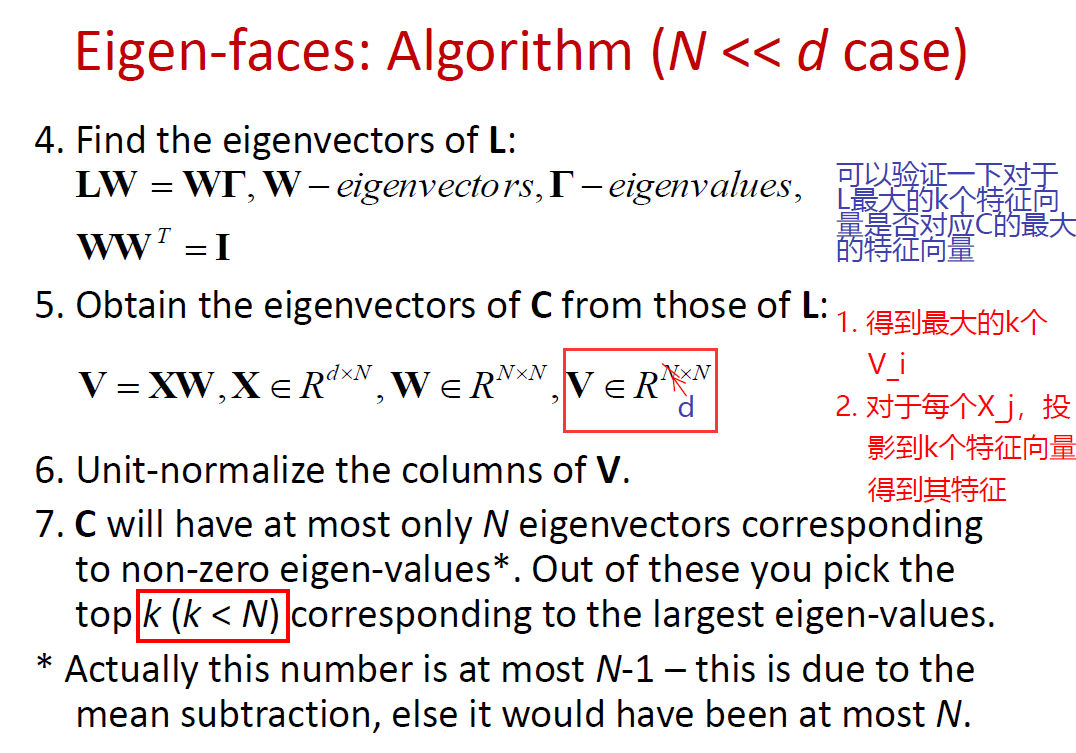


细节。

老师在pdf中提到了。



按照这种方法，以将7张训练图像平均后作为一个特征图像再进行PCA特征抽取，用于PCA特征抽取的图像数量N是40。



当N=40时，求出来的协方差矩阵C的特征向量矩阵大小是d\*N，我们得到的特征向量的数量是N=40个。

而老师在pdf中又提到，建议k取50-100，(k < N)，此时我们得到的特征向量只有40个，无法满足k取50-100， 感觉将7张训练图像平均后作为一个特征图像再进行PCA特征抽取，会忽略掉同一个人的不同训练图像之间区别，降低人脸识别准确率。所以我没有用这种方法。

