**深 圳 大 学 实 验 报 告**

|  |
| --- |
| **课程名称 计算机导论**  **项目名称 PCA与人脸识别**  **学 院 计算机与软件学院**  **专 业 软件工程**  **指导教师 赖志辉**  **报 告 人 赵美玲 学号 2023155025**  **实验时间 2022.11.3-2022.11.6** |

**教务处制**

# 实验目的与要求

1系统阐述主成分分析（PCA）的理论。

2.给出PCA人脸识别的算法。

3.写出程序实现（不限语言）。

4.不少于2个数据集上比较PCA提完特征的识别率与KNN在原始数据上的识别率（训练样本数分别是前3，前4，前5，前6都要做，即列表比较）；然后再随机选2，3，4，5，6个训练样本，重复10次实验，列表比较平均识别率与标准差。

5.可示化。展示前10个投影形成的图片；展示用前10个、前20、前50个投影进行重构原图生成的图片与误差。

6．每个数据库，把3个类的数据投影到2维、3维空间（即用PCA的前2，3个投影）用不同的标记显示出来，看完后总结如何设计算法才能更好识别各个人脸，总结一下。

# 二、实验内容与方法

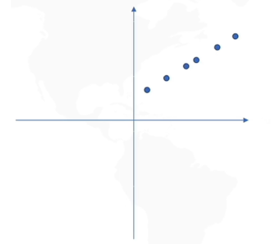
上网学习PCA，matlab，以及其他人的文章。

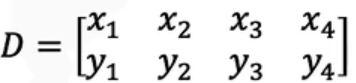
# 三、实验步骤与过程

1.系统阐述PCA的理论

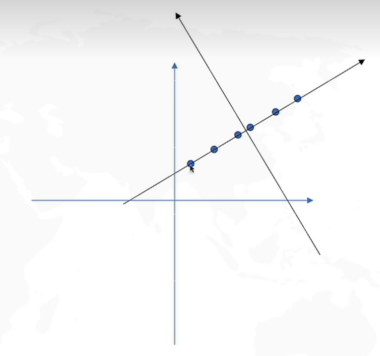
PCA的目的是简单化数据的同时能够保证简单化后的数据仍然符合人类运用的需求，使得后续所需要处理的数据计算量变小，从而提高PCA所属的程序运行效率，并且仍能符合人类的需求。

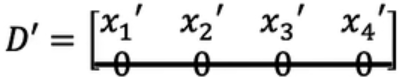
首先我们来考虑一个特殊情况。



此时我们储存的信息是： 

当我们转变坐标轴以后。就会出现以下情况。



那么此时我们需要保存的信息就会变化成

我们可以发现需要保存的数据变少了，而且二维的数据也变成了一维的数据，那么这个时候不仅运算量小了，而且还能够保留数据。这就是PCA的第一步，找坐标系。

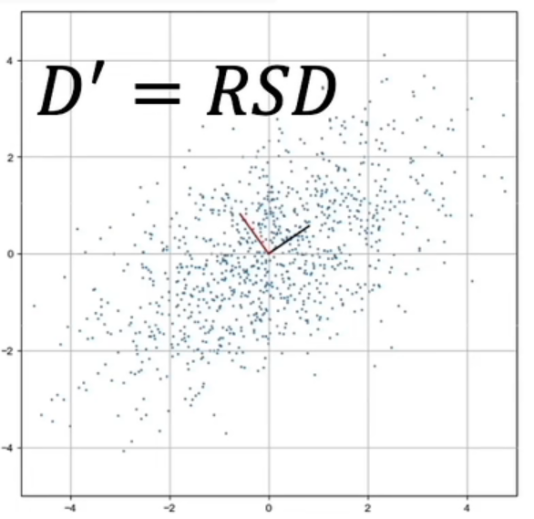
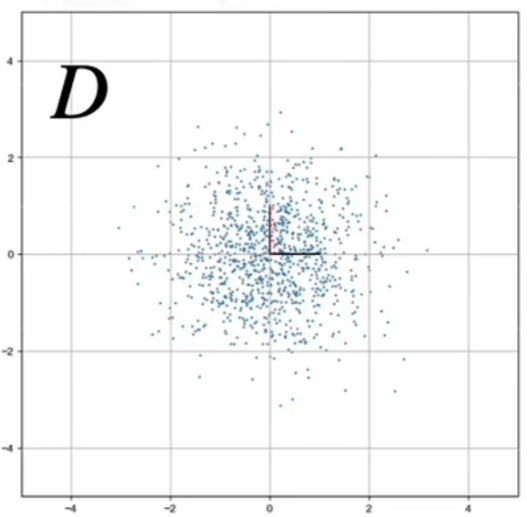
但是更多的时候数据并非是纯粹的线性排布，那么为了仍然降低提高效率的目的，我们就需要找到一种更加普适方法去运用PCA.。

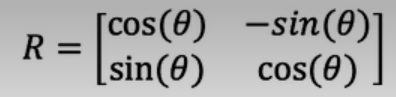
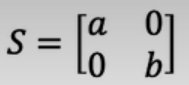
那么我们就要换坐标系。先确定坐标的原点（为什么要确定坐标的原点，假如说我的数据很密集离我本来的原点很远。那么我很难找到很适合的数据组的），原点将定在所有数据求平均值的中心，然后是坐标轴的方向，我们将我们最想要的数据组（最想要的数据组即损失的数据最小，但是却最能体现原来数据特征的数据组）所投影的轴称为主成分一，另一个相对的我们最不想要的称为主成分二。这个时候我们的数据就成功的，按照我们的想法降维了。实际上PCA就可以抽象为一个找坐标的计算。

1. PCA人脸识别的算法

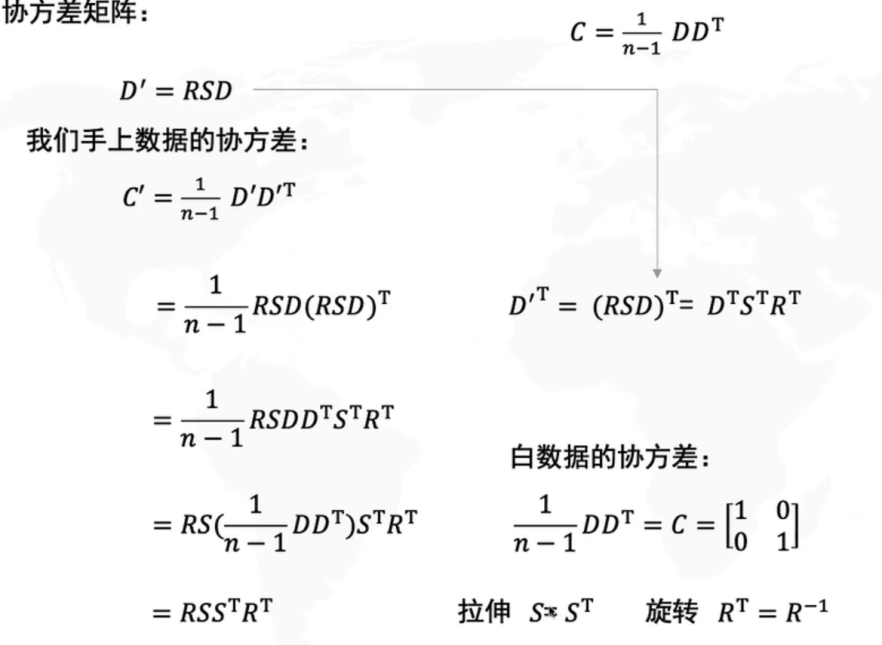
那怎么实现坐标系的变换呢？原点好找，那个数据我们拿到手了就可以计算中心的值了。而坐标轴方向怎么计算呢？我们就可以运用协方差计算，方差最大的即使我们最想要的数据组，方差最小的即使我们不想要的数据组。

那么怎么求得坐标轴方向？

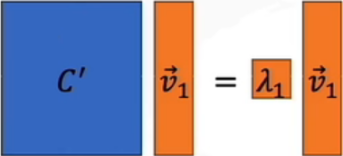
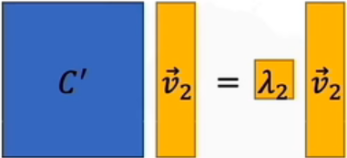


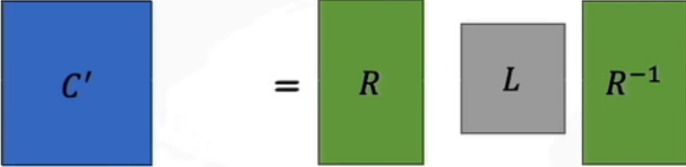


这里有白数据D，S起拉伸，和R起的是旋转角度的作用，D’即使我们一般情况下拥有的数据。



这个证明过程系统性的推到了D’的协方差矩阵C’的与R和S矩阵的关系。



第一张及第二张图片在定义特征向量，和特征值的同时，也证明了特征向量即使R的每一行列向量，特征值即是S的a的平方和b的平方。同时我们还要注意到S的另一条对角线除了a，b所在的对角线其他元素全为零，那么这意味着我们只需要求出除主对角线外协方差为零时的协方差矩阵即可。而此时对角线的值即使我们要的特征值。

运用到人脸识别上我们还需要将前几个特征向量组成变换矩阵W,并然后降维操作就是用W乘以数据矩阵，得到了低维度的数据。重构就是用变换后的数据乘以W的转置，得到之前的原始数据。

1. PCA人脸识别的算法实现（代码后有注释）

reshaped\_faces=[];%%创建一个矩阵

for i=1:99%循环多少个人

for j=1:20%每人多少个样本

if(i<10)%输入图片自动转化为矩阵

a=imread(strcat('C:\Users\27419\Desktop\pca\AR\_Gray\_50by40\AR00',num2str(i),'-',num2str(j),'.tif'));%num2str（）数组将数字转化为字符，构建图像地址

else

a=imread(strcat('C:\Users\27419\Desktop\pca\AR\_Gray\_50by40\AR0',num2str(i),'-',num2str(j),'.tif'));

end%结束选择循环

b=reshape(a,2000,1);%创建数组将每一张照片对应矩阵拉成列，2000为像素

b=double(b);%增大存储量

reshaped\_faces=[reshaped\_faces,b];%将循环中的b全部输入新矩阵中，每一列为一张图片

end

end

test=[];%定义一个测试数据组

train=[];%定义一个训练数据组

for i=0:99

test=[test 10\*i+1:10\*i+6];%提取每一张人脸前六张作为测试

train=[train 10\*i+7:10\*i+14];%提取每一张人脸的后14张作为训练

end

train\_data=reshaped\_faces(:,train);%将每张人脸的train数列合并为一个矩阵，以便开始降维

test\_data=reshaped\_faces(:,test);

%%求均值和中心化减去平均脸

mean\_face=mean(train\_data,2);%求训练脸的平均mean函数

centered\_face=(train\_data-mean\_face);%每一张脸均减去平均脸

%协方差特征值等准备进行降维（直接用cov函数）

cov\_matrix=centered\_face\*centered\_face';%求协方差

[eigen\_vectors,dianogol\_matrix]=eig(cov\_matrix);%特征向量转化为对角矩阵，eig函数的用法第一个为特征向量，第二个为特征值（以矩阵方式返回）

eigen\_values=diag(dianogol\_matrix);%从对角阵获取特征值diag函数获取矩阵对角元素

[sorted\_eigen\_values,index]=sort(eigen\_values,'descend');%sort函数 按降序对特征值进行排序（按列排序）

sorted\_eigen\_vectors=eigen\_vectors(:,index);%排序后特征值对应的特征向量

all\_eigen\_faces=sorted\_eigen\_vectors;%获取所有特征脸的特征向量

重构（数据降维，还原图片矩阵）

single\_face = centered\_face(:,1);%取出第一个人脸进行重构

index = 1;%索引1

for dimensionality=20:20:160%以20维度开始向160逐渐增加

eigen\_faces = all\_eigen\_faces(:,1:dimensionality);

rebuild\_face = eigen\_faces \* (eigen\_faces' \* single\_face) + mean\_face;%在不同维度下重构人脸

subplot(2, 4, index); %两行四列，subplot函数创建坐标区

index = index + 1;%依次增大索引，不同照片

fig = show\_face(rebuild\_face);%展示照片

title(sprintf("dimensionality=%d", dimensionality));%明晰该照片的维度

if (dimensionality == 160)

waitfor(fig);%阻止程序的运行

end%if选择函数的结束

end%循环结束，第一张照片的测试完成

人脸识别

index = 1;

Y = [];

KNN

for k=1:6

for i=10:10:160

eigen\_faces = all\_eigen\_faces(:,1:i);

% 测试、训练数据降维

projected\_train\_data = eigen\_faces' \* (train\_data - mean\_face);

projected\_test\_data = eigen\_faces' \* (test\_data - mean\_face);

% 用于保存最小的k个值的矩阵

% 用于保存最小k个值对应的人标签的矩阵

minimun\_k\_values = zeros(k,1);

label\_of\_minimun\_k\_values = zeros(k,1);

% 测试脸的数量

test\_face\_number = size(projected\_test\_data, 2);

% 识别正确数量

correct\_predict\_number = 0;

% 遍历每一个待测试人脸

for each\_test\_face\_index = 1:test\_face\_number

each\_test\_face = projected\_test\_data(:,each\_test\_face\_index);

% 先把k个值填满，避免在迭代中反复判断

for each\_train\_face\_index = 1:k

minimun\_k\_values(each\_train\_face\_index,1) = norm(each\_test\_face - projected\_train\_data(:,each\_train\_face\_index));

label\_of\_minimun\_k\_values(each\_train\_face\_index,1) = floor((train\_data(1,each\_train\_face\_index) - 1) / 10) + 1;

end

% 找出k个值中最大值及其下标

[max\_value, index\_of\_max\_value] = max(minimun\_k\_values);

% 计算与剩余每一个已知人脸的距离

for each\_train\_face\_index = k+1:size(projected\_train\_data,2)

% 计算距离

distance = norm(each\_test\_face - projected\_train\_data(:,each\_train\_face\_index));

% 遇到更小的距离就更新距离和标签

if (distance < max\_value)

minimun\_k\_values(index\_of\_max\_value,1) = distance;

label\_of\_minimun\_k\_values(index\_of\_max\_value,1) = floor((train\_data(1,each\_train\_face\_index) - 1) / 10) + 1;

[max\_value, index\_of\_max\_value] = max(minimun\_k\_values);

end

end

% 最终得到距离最小的k个值以及对应的标签

% 取出出现次数最多的值，为预测的人脸标签

predict\_label = mode(label\_of\_minimun\_k\_values);

real\_label = floor((test\_data(1,each\_test\_face\_index) - 1) / 10)+1;

if (predict\_label == real\_label)

correct\_predict\_number = correct\_predict\_number + 1;

end

end

% 单次识别率

correct\_rate = correct\_predict\_number/test\_face\_number;

Y = [Y correct\_rate];

end

end

% 求不同k值不同维度下的人脸识别率及对k的平均识别率

Y = reshape(Y,k,16);

waitfor(waterfall(Y));

avg\_correct\_rate=mean(Y);

waitfor(plot(avg\_correct\_rate));

%内用函数定义

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 输入向量，显示脸

function fig = show\_face(vector)

fig = imshow(mat2gray(reshape(vector, [50, 40])));

end

% 显示矩阵中某些脸

function fig = show\_faces(eigen\_vectors)

count = 1;

index\_of\_image\_to\_show = [1,5,10,15,20,30,50,70,100,150];

for i=index\_of\_image\_to\_show

subplot(2,5,count);

fig = show\_face(eigen\_vectors(:, i));

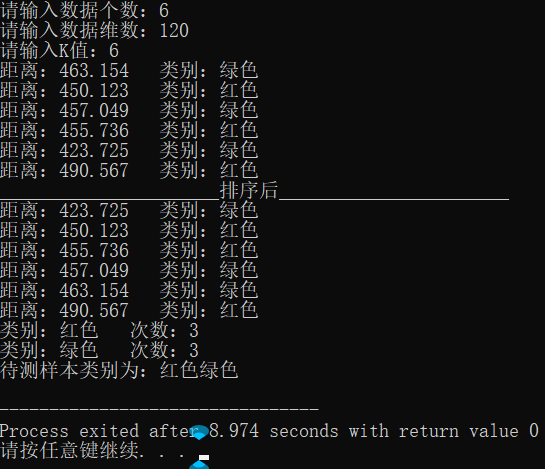
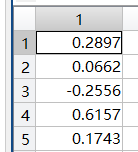
title(sprintf("i=%d", i));

count = count + 1;

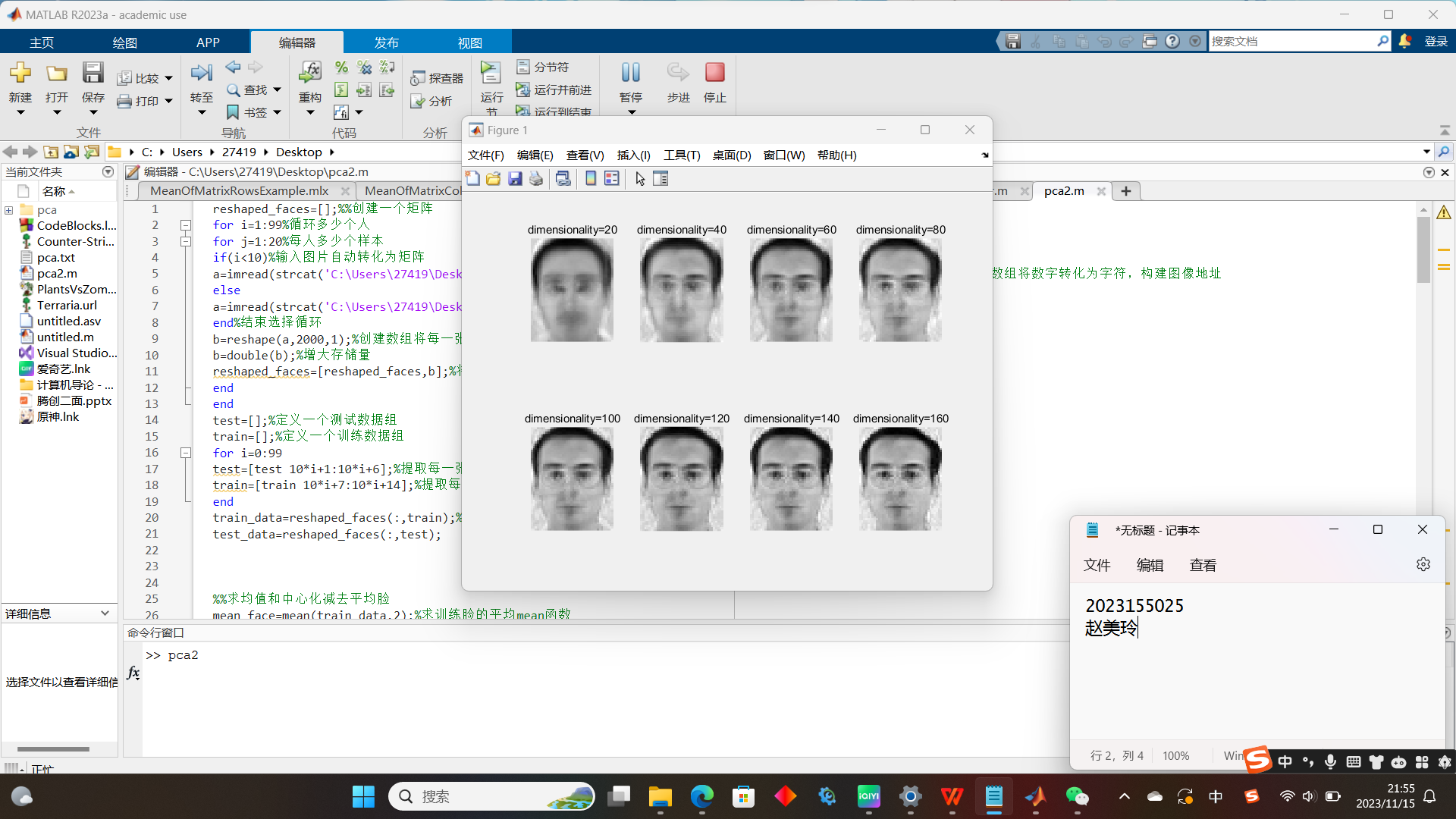
end

end

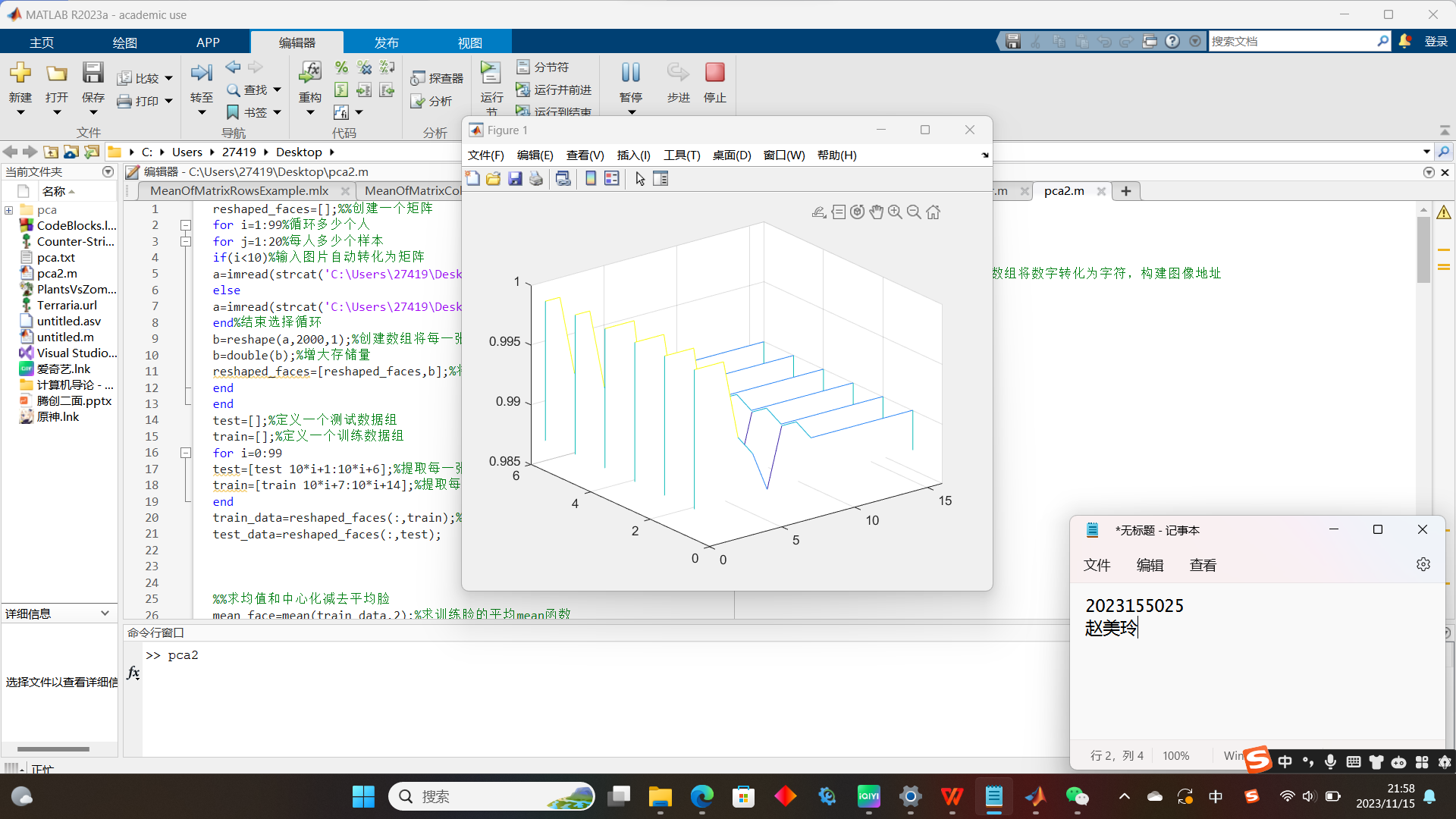
4.比较



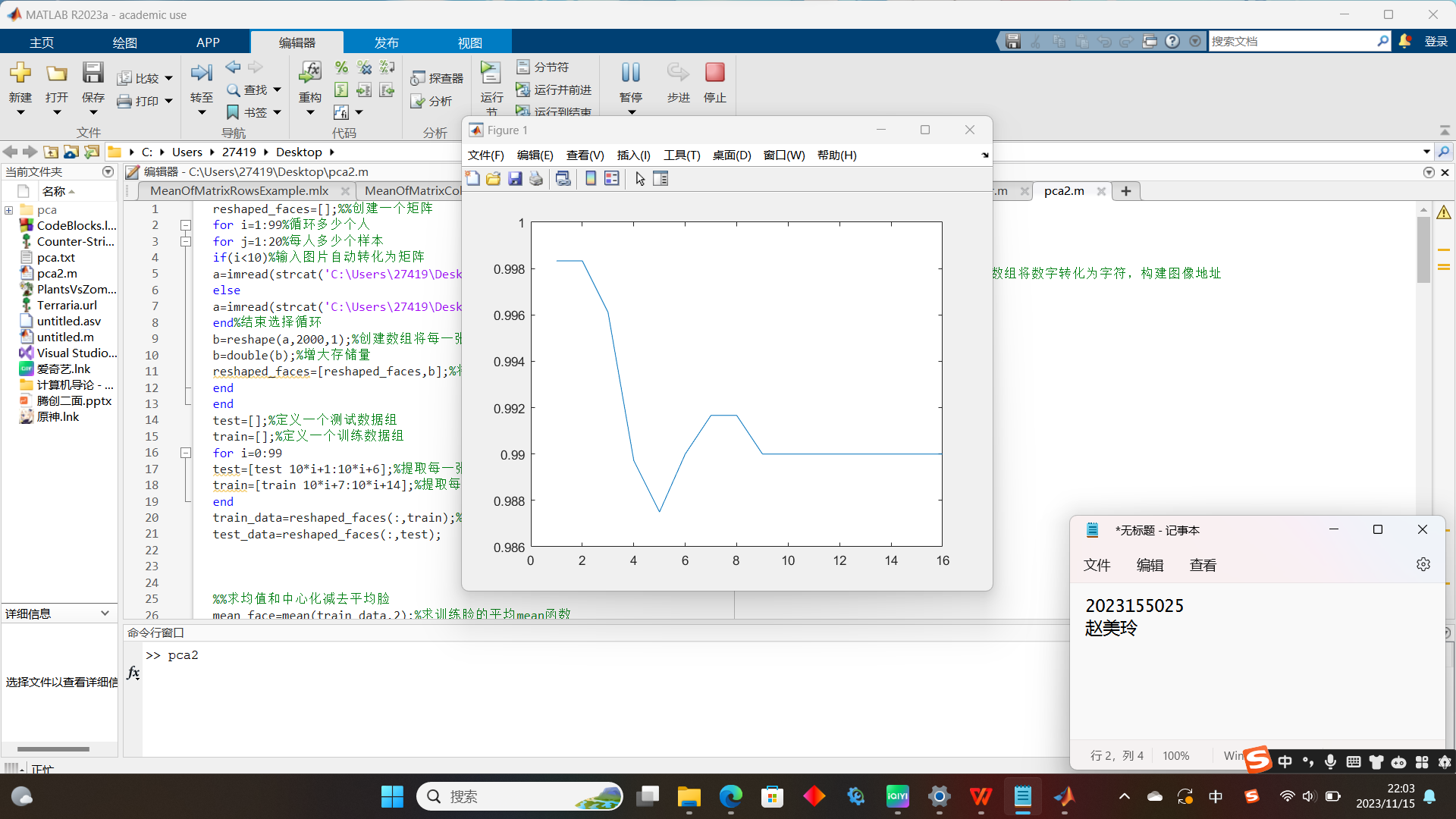
5.人脸重现。



6.三维显现。



三维显现



二维展示不同维度下的正确识别率

7.思考

很难，非常难，极其难。

所有的东西都是第一次接触，pca，matlab，knn等等算法都是，在网上找文章，每个人风格不同，常常这一篇看完看下一篇的时候发现二者没有关联，矩阵也需要自己去学，如何开头是最困难的，虽然理解pca的思路和大概实现方法，但是最后要将他通过代码实现还具有很大的鸿沟，只有通过对前人代码一点点的挖掘成为自己的东西，才能勉强实现对代码的理解，在此基础上稍作修改。中途遇到不懂的地方最好的办法就是单独拉出来运行，一切问题就会迎刃而解。

面对计算机这个东西还有很长的路要走，道阻且长，行则将至。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。