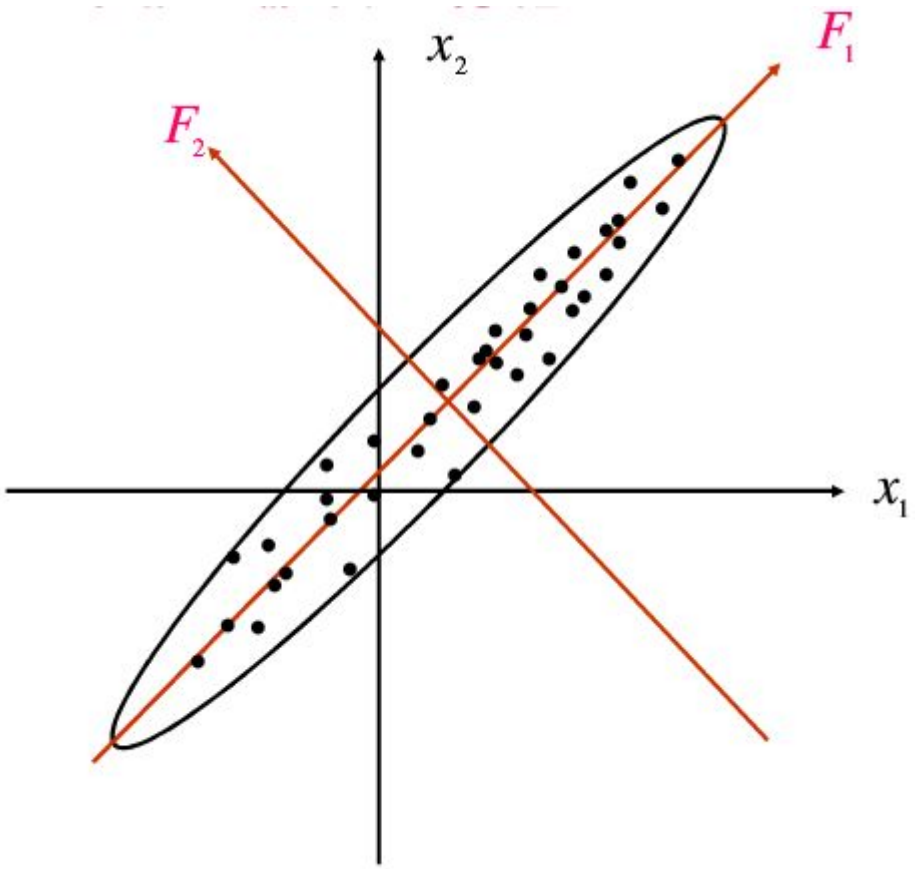
一、PCA压缩

用两种PCA方法对人脸图像进行压缩，分别给出压缩比为50%，60%，70%，80%，90%，95%时，SNR是多少。更进一步，对两种PCA的求法进行比较分析

1. 基本原理

1.1理论原理

对于给定的向量数据集，一般情况下，向量分量与分量之间存在高度的相关性，这表现在线性空间中，就是数据集大多数都分布在某个子空间附近，因此，对原始数据集的坐标轴进行旋转，得到新的坐标轴，在新的坐标轴中，数据的分布使得某些分量值更接近在0附近，从而其他维的信息可以作为该数据的估计。而这里的变换矩阵，从理论上来说，可以通过求协方差矩阵特征向量来获得，对于新坐标系中某些维的省略，构成了PCA压缩的原理，这也是PCA算法一种几何解释。如下图所示，向量$\mathbf{F_1},\mathbf{F_2}$就是这些数据的两个主元。



PCA本质上是一个基替换的过程，假设元数据$\mathbf{x}^p \in R^n$，变换后的数据$\mathbf{y}^p \in R^m$。其目的是找到m个基，$m < n$。使得投影后的数据误差最小。

因此，记新的一组正交基为：$\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_n$。则

$$\mathbf{x}^p = \sum_{k=1}^{n} a_k^p \mathbf{u}_k$$

a_i^p为$\mathbf{u}_i$上的投影系数。取其中的m个基：

$$\tilde{\mathbf{x}}^p = \sum_{i=1}^{m} a_k^p \mathbf{u}_i$$

则目标即是：

$$\min E = \frac{1}{2}\sum_{p=1}^{N}(\mathbf{x}^p - \tilde{\mathbf{x}})^T(\mathbf{x}^p - \tilde{\mathbf{x}}) =$$

$$E = \frac{N}{2}\sum_{i=m+1}^{n} \mathbf{u}_i^T \mathbf{C}_x \mathbf{u}_i$$

所以，当$\mathbf{u}_i$为协方差矩阵$\mathbf{C}_x$的特征向量时：

$$E = \frac{N}{2}\sum_{i=m+1}^{n} \mathbf{u}_i^T \lambda_i \mathbf{u}_i$$

$$E = \frac{N}{2}\sum_{i=m+1}^{n} \lambda_i (\mathbf{u}_i^T \mathbf{u}_i)$$

所以当取最大的m个特征值对应的特征向量时，E将最小。

1.2 算法分类

1.2.1 COVPCA

即用协方差矩阵的特征值来进行PCA，流程如下：

1. 向量数据集去均值$\mathbf{X} = \mathbf{X} - E\{\mathbf{X}\}$
2. 求协方差矩阵$Cov(\mathbf{X}) = E\{\mathbf{X}\mathbf{X}^T\}$
3. 求取协方差矩阵的特征值和特征向量。保留前m个最大特征值以及对应的特征向量$\mathbf{u}_{\max i}, i = 1, \ldots, m$
4. 得到$y_i^p = (\mathbf{x}^p)^T \mathbf{u}_{\max i}$

可以预想到，这种方法在协方差矩阵维数小时会很快，但是当协方差矩阵维数很大时，会出现维数灾难，大大增加计算量，使得非常耗时。当然，当样本数和样本长有一个较小时，可以用对样本矩阵先转置再求解的方法来处理，这样会大大减少运算量，因为两个协方差矩阵的特征值是一致的。

1.2.2 CCIPCA(Candid Covariance Free Incremental PCA)

在协方差PCA方法中，计算协方差是唯一在实际操作中比较困难的方法。由于我们对于统计量，可以采用步进的方法实现，当输入的样本数量足够多，且样本的统计特性比较稳定的时候，步进的方式得到的协方差可以趋于真实数据的协方差。在该思路的基础上，提出了CCFIPCA的算法。

首先是对于第一个主元的收敛。若在某次迭代后，$\mathbf{w}(t)$趋于对应最大特征值的特征向量的倍数，即$\mathbf{w}(t) = \boldsymbol{\lambda}\mathbf{u}_i$，其中$\boldsymbol{\lambda}$为特征值，则由特征值特征向量的定义，有：

$$\mathbf{w}(t) = \frac{1}{t}\sum_{i=1}^{t} \mathbf{x}^i \mathbf{x}^{iT} \frac{\mathbf{w}(t-1)}{\|\mathbf{w}(t-1)\|}$$

当$\mathbf{w}(t)$稳定时，由上式可以得到迭代公式：

$$\mathbf{w}(t) = \frac{t-1}{t}\mathbf{w}(t-1) + \frac{1}{t}\mathbf{x}(t)\mathbf{x}(t)^T \frac{\mathbf{w(t-1)}}{\|\mathbf{w(t-1)}\|}$$

所以当$\mathbf{w}(t)$收敛后，归一化即可得到最大特征值对应的特征向量，然后将原数据减去该方向上的投影，即：

$$\mathbf{x} = \mathbf{x} - \frac{\mathbf{x}^T\mathbf{w}}{\|\mathbf{w}\|} \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

之后对于去除第一主元的样本再用上述方法可以求取次大的特征值所对的特征向量。而实际操作中减去主分量的步骤可以穿插在每一步迭代中进行。

2. 仿真方法

2.1 压缩比和PCA主元数

压缩比又称为压缩率，PCA压缩的计算方法如下：

记主元个数为m，主元长度为l，样本个数为n，则，压缩比P为：

$$P = \frac{l \times m + m \times n}{n \times l}$$

也有将压缩比记为被压缩数据量与原数据量的比值，即为$1 - P$， 在这里，我采用上面的说法，出于完整性考虑，在列表时也会列出10%-90%的PSNR值，但出于篇幅考虑，仅放少部分图片。

所以主元数与其他参数的关系为：

$$m = \frac{n \times l \times P}{n + l}$$

在本例中，所有图像为：92×112pixel大小，所以可以取每个主元长度为10304。按人进行压缩时，则每次有10个样本，则（无法达到95%压缩率，所以不列出）：

压缩率	主元个数m	实际压缩率
10%	1	10.01%
20%	2	20.02%
30%	3	30.03%
40%	4	40.04%
50%	5	50.05%
60%	6	60.06%
70%	7	70.07%
80%	8	80.08%
90%	9	90.09%

当以每行像素为样本时，则主元长度为92，按人进行压缩，则每次有1120个样本，则：

压缩比	主元个数m	实际压缩率
5%	4	4.47%
10%	9	10.59%
20%	17	20.00%
30%	26	30.58%
40%	34	39.99%
50%	43	50.58%
60%	51	59.99%
70%	60	70.57%
80%	68	80.00%
90%	77	90.57%
95%	81	95.28%

以每列为样本时，主元长为112，按人进行压缩，则每次有920个样本，则：

压缩比	主元个数m	实际压缩率
5%	5	5.02%
10%	10	10.04%
20%	20	20.08%
30%	30	30.12%
40%	90	40.16%
50%	50	50.08%
60%	60	60.09%
70%	70	70.11%
80%	80	80.12%
90%	90	90.14%
95%	95	95.15%

以每幅图的同一个像素作为样本，主元长为10，每次有10304个样本，则：

压缩比	主元个数m	实际压缩率
10%	1	10.01%
20%	2	20.02%
30%	3	30.03%
40%	4	40.04%
50%	5	50.05%
60%	6	60.06%
70%	7	70.07%
80%	8	80.08%
90%	9	90.09%

3.2压缩图像评价标准：

3.2.1 峰值信噪比PSNR

一般来说，对于压缩最直观也最常用的便是肉眼评价，但是这一方法具有一定的主观性，并且，对于微小差异无法给出评价。在此，我们使用峰值信噪比（PSNR）和压缩比进行评价，其中，峰值信噪比定义为：

$$\mathrm{PSNR} = 10\log_{10}(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}})$$

MAX_I为图像点最大值，对于uint8的图像，$\mathrm{MAX}_I = 255$，MSE为均方误差：

$$\mathrm{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\|I(i,j) - \hat{I}(i,j)\|^2$$

I为原始图像，$\hat{I}$为压缩后的图像。

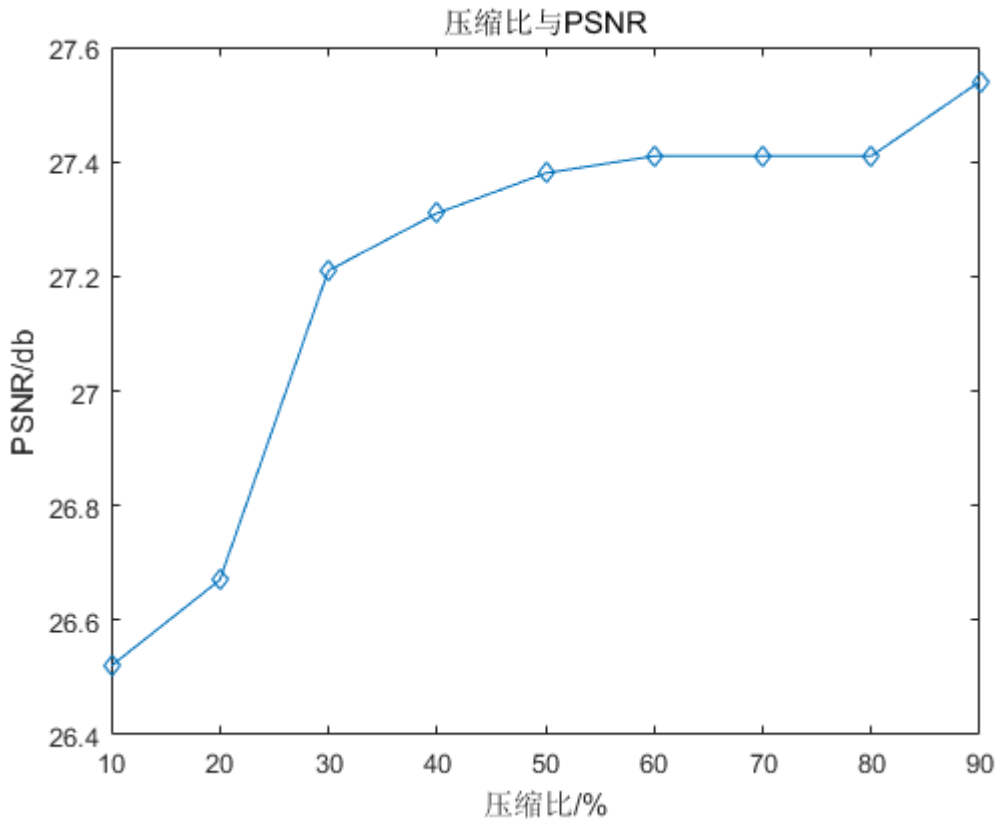
3. 压缩结果

以下结果均以s1作为样本进行PSNR的分析：

3.1 COVPCA

1. 按图像作为样本进行压缩：

压缩比	PSNR（db）
10%	26.52
20%	26.67
30%	27.21
40%	27.31
50%	27.38
60%	27.41
70%	27.41
80%	27.41
90%	27.54

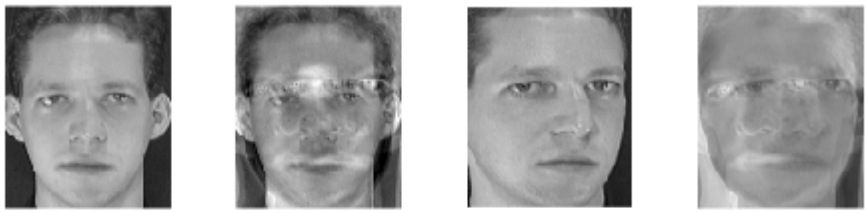


测试结果：（为了节约篇幅，只放出10%，30%，50%，70%，90%的效果，左侧是原图，右图是压缩后的图片）

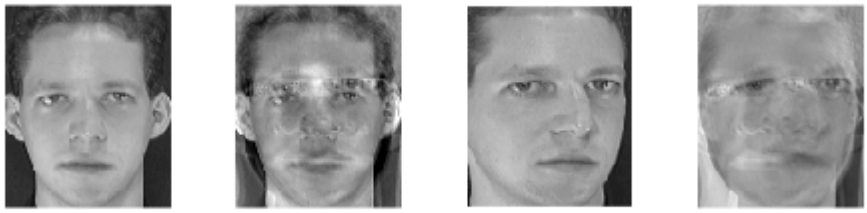
10%：

30%:

50%:



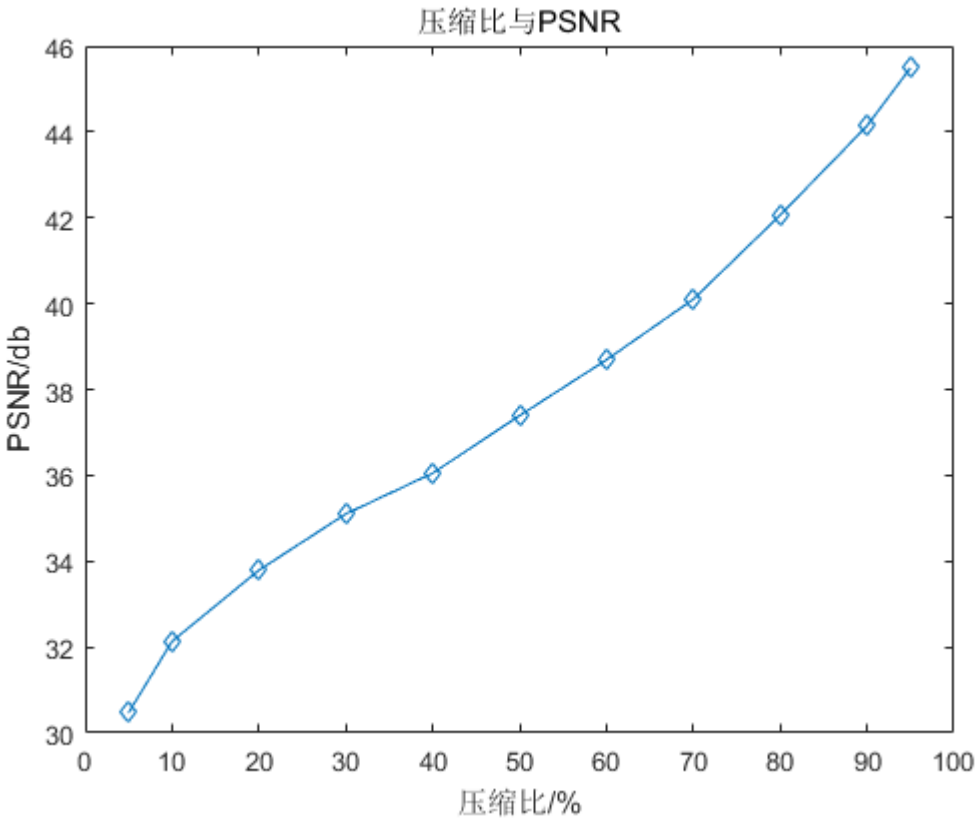
70%:



90%:

2. 按列作为样本进行压缩：

压缩比	**PSNR（db)**
5%	30.18
10%	32.25
20%	35.02
30%	37.57
40%	39.75
50%	41.79
60%	43.69
70%	45.74
80%	48.06
90%	50.87
95%	52.56


压缩比与PSNR
PSNR/db
压缩比/%
46
44
42
40
38
36
34
32
30
0
10
20
30
40
50
60
70
80
90
100

10%:

30%:

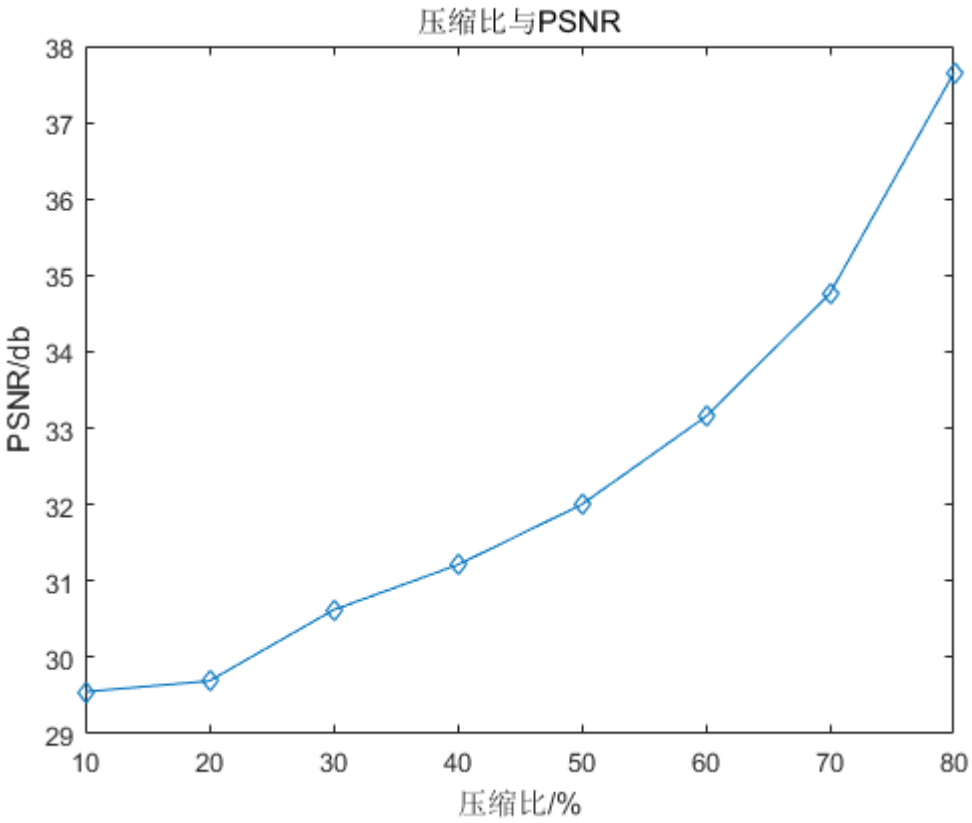
50%:

70%:

90%：

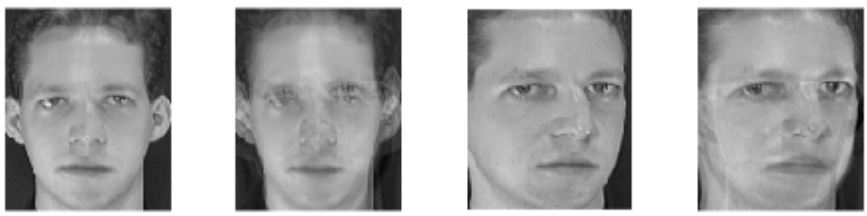
3. 按像素作为样本进行压缩：

压缩比	PSNR（db）
10%	29.55
20%	29.69
30%	30.62
40%	31.21
50%	32.00
60%	33.15
70%	34.76
80%	37.65
90%	Inf



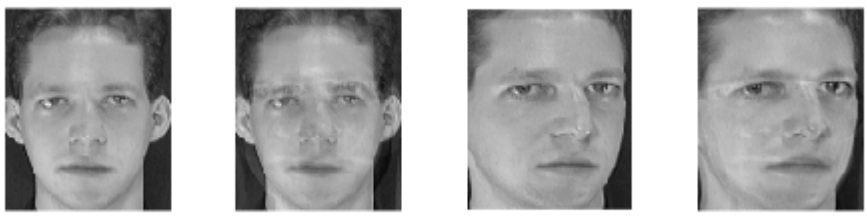
10%:

30%:



50%:

70%:

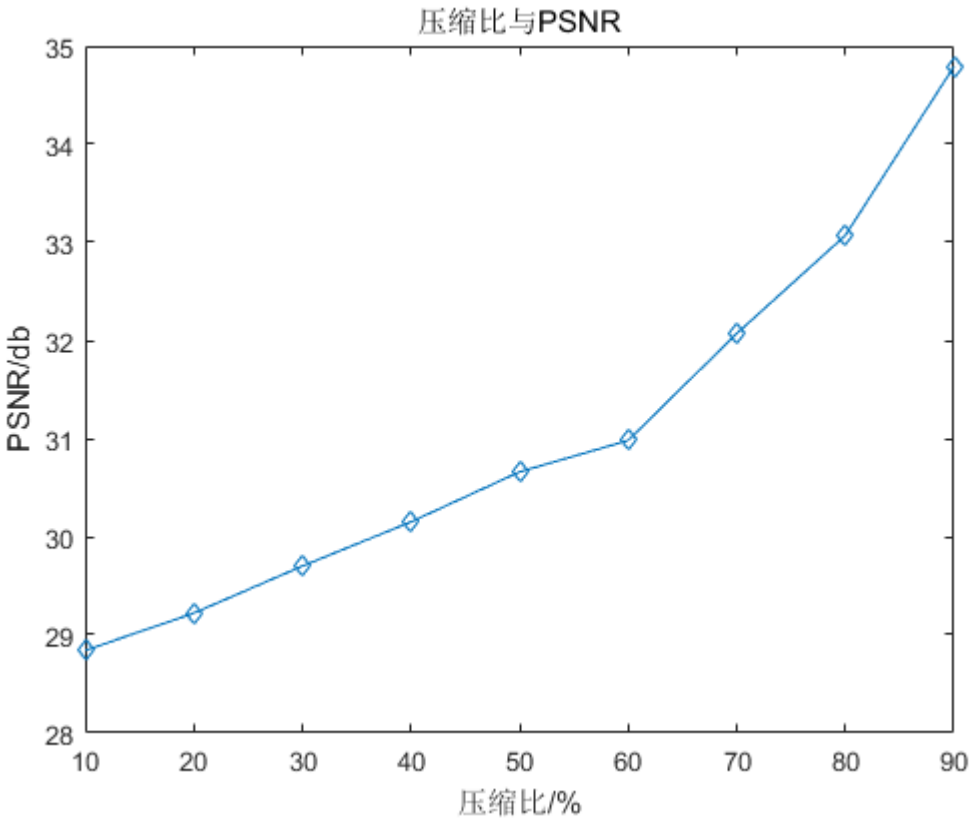


90%:

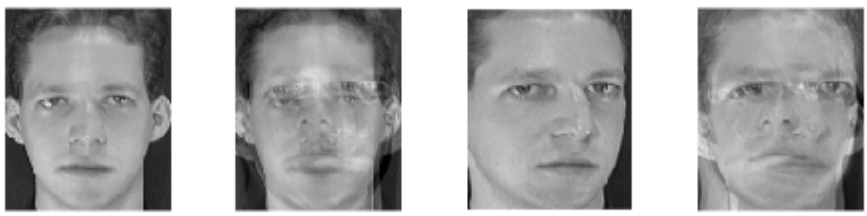
3.2 CCIPCA

1. 按图像作为样本进行压缩：

压缩比	PSNR（db）
10%	28.84
20%	29.22
30%	29.70
40%	30.15
50%	30.66
60%	30.98
70%	32.07
80%	33.07
90%	34.79


压缩比与PSNR
PSNR/db
压缩比/%
35
34
33
32
31
30
29
28
10
20
30
40
50
60
70
80
90

50%：

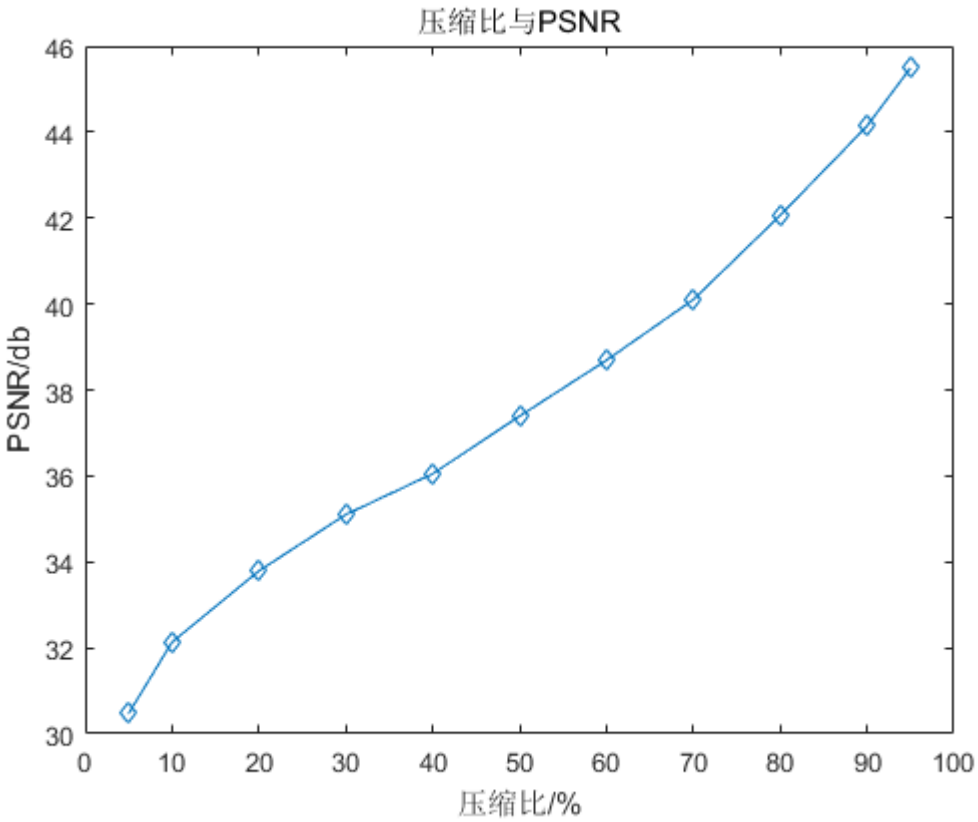


70%：

90%：

2. 按列作为样本进行压缩：

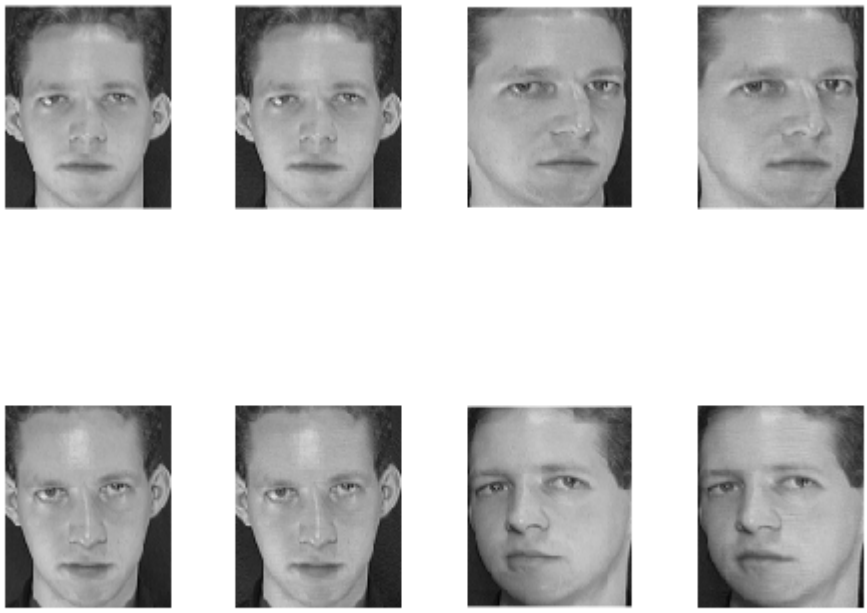
压缩比	PSNR（db）
5%	30.47
10%	32.13
20%	33.79
30%	35.10
40%	36.05
50%	37.39
60%	38.69
70%	40.10
80%	42.06
90%	44.14
95%	45.49



50%：

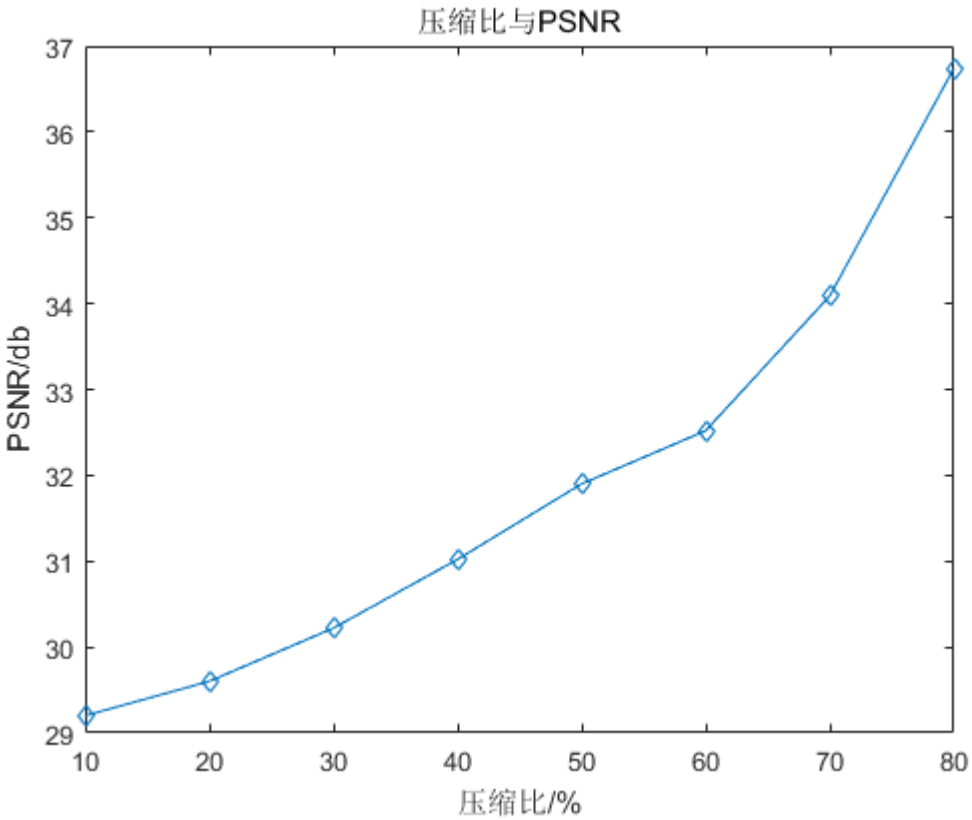
70%:

90%:



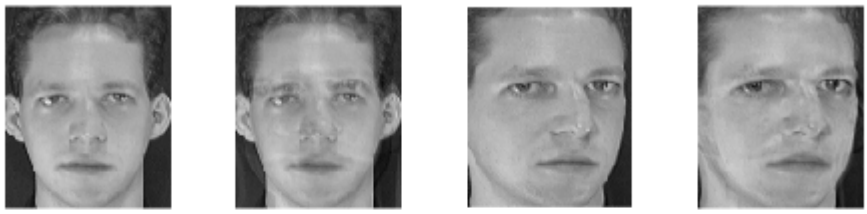
3. 按像素作为样本进行压缩：

压缩比	**PSNR（db)**
10%	29.20
20%	29.60
30%	30.22
40%	31.02
50%	31.90
60%	32.52
70%	34.09
80%	36.74
90%	Inf

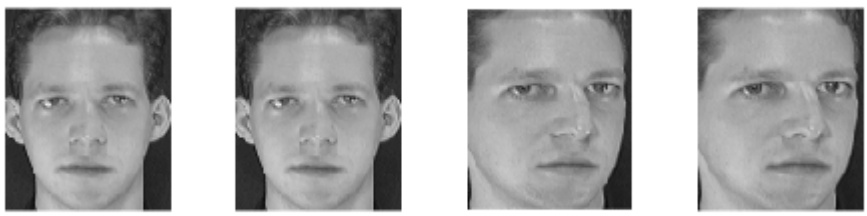

压缩比与PSNR
PSNR/db
压缩比/%
37
36
35
34
33
32
31
30
29
10
20
30
40
50
60
70
80

50%：

70%：



90%：



4. 总结

4.1运行时间

在运行过程中，当不分块直接PCA时，COVPCA运算非常慢，但CCIPCA运算比较快，这主要是由于CCIPCA影响运行时间主要是样本个数，而COVPCA影响运算时间的主要是主元长度。当不分块时，样本数量小，因此CCIPCA比COVPCA快。但当分块或按像素进行压缩时，COVPCA要远快于CCIPCA。这也是因为此时，样本数量变大而主元长度变小。

4.2 压缩效果

当不分块时，无论是从PSNR的数据还是从肉眼观察都可以看出，CCIPCA的效果要好于COVPCA，这可能是由于样本数较少，CCIPCA迭代并不完全收敛于最大特征向量处，反而会收敛于某几幅图相似图的叠加效果上，这反而使得压缩效果好于原来COVPCA。但是当用像素或用列作为主元时，COVPCA就好于CCIPCA。这是因为当取主元数足够多时，COVPCA才是最优的PCA结果，CCIPCA只是逼近COVPCA，这在理论上也是合理的。

而在压缩效果来说，用一幅图长度作为主元时，压缩率低时，可以看到恢复效果是很多人的脸的替加。随着压缩率上升，可以看到自己脸的轮廓变得清晰，但还是会有其他人脸的模轮廓。而用一列长度作为主元长度时，压缩率低时会有横条纹的效果。当用像素作为输入样本时，压缩率低时，图案会变得模糊。

同时，从PSNR可以看出，按列压缩取得的效果要好于另两者。这也说明了，用PCA压缩时，对主元长度或者说输入样本的选择很重要。