Programming Assignment 1

应之未 521030910376

2023/2/23

一、实验内容和过程概述

1. 1）通过 imread 函数读取图片"baboon.bmp"至矩阵 A 中，再创建 5*5 的全 0.04 矩阵 B 作为点扩散函数。

   2）考虑到卷积后的图像为 516*516，而我们希望的得到的图像为 512*512，因此创建一个数组 rect = [3, 3, 511, 511]来标记需要截取的图片位置和大小。

   3）采用 conv2 函数对 A 和 B 进行卷积得到 C，将 C 通过 rect 数组进行截取之后可以得到"blurred.bmp"。

2. 通过 awgn([matrix], snr)函数对 C 添加高斯噪声，调整 snr 的取值可以得到模糊处理后分别加入 30dB，20dB，10dB 高斯噪声后的图片。

3. 1）采用 deconvwnr([matrix], PSF, 0)进行直接逆滤波。

   2）采用 deconvwnr([matrix], PSF, nsr)（其中 nsr = 1/snr）进行 Wiener 滤波。

二、实验结论

1. 从结果上看，图像与点扩散函数卷积后可以明显发现图像变模糊，从理论上讲，相当于把一个数据点的数据均摊到周围 25 个数据点，每个数据点都是周围 25 个数据点数据的平均，理论推导与实验结果相一致。

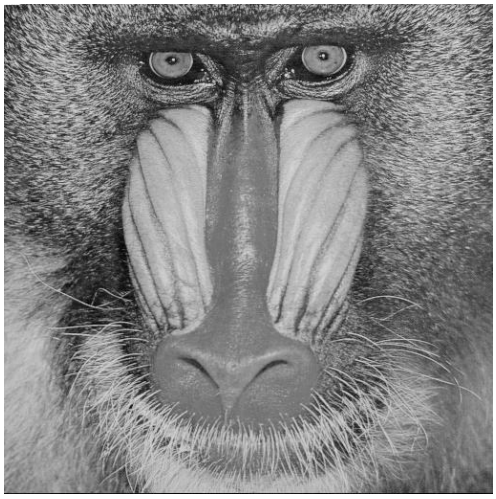
2. • 若对含有高斯噪声的图像进行直接逆滤波，根据上课的推导，会导致计算过程中出现 a/b(b->0)的情况，使得最后产生的数据非常极端，在图像上呈现的状态即为黑白相间的方块。

3. 在模糊化和叠加噪声后，图像的部分信息丢失，即使经过逆滤波后也难以恢复成原图的效果。

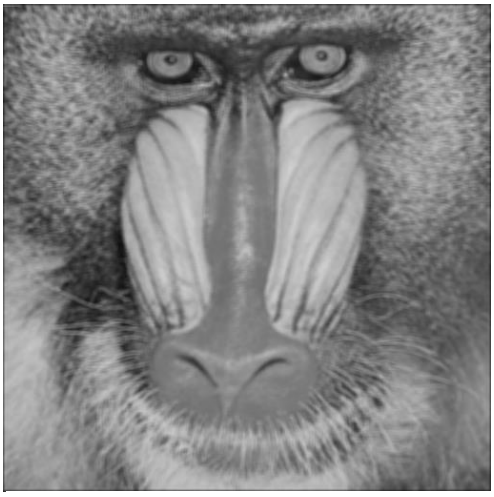
三、实验代码

```
A = imread("baboon.bmp");
B = [ 0.04 0.04 0.04 0.04 0.04 ;
      0.04 0.04 0.04 0.04 0.04 ;
      0.04 0.04 0.04 0.04 0.04 ;
      0.04 0.04 0.04 0.04 0.04 ;
      0.04 0.04 0.04 0.04 0.04 ]; % 卷积核（点扩散函数）
rect = [3,3,511,511]; % imcrop截取函数的参数，用以将卷积得到的516*516图像截取成512*512图
像
C = conv2(A,B);
C1 = uint8(C);
Cout = imcrop(C1, rect);
imwrite(Cout,'blurred.bmp');

% 加入高斯噪声
D1 = awgn(C,30,"measured");
D11 = uint8(D1);
D1out = imcrop(D11, rect);
imwrite(D1out,'blurred and noised in 30 dB.bmp'); % 加入30dB高斯噪声

D2 = awgn(C,20,"measured");
D21 = uint8(D2);
D2out = imcrop(D21, rect);
imwrite(D2out,'blurred and noised in 20 dB.bmp'); % 加入20dB高斯噪声

D3 = awgn(C,10,"measured");
D31 = uint8(D3);
D3out = imcrop(D31, rect);
imwrite(D3out,'blurred and noised in 10 dB.bmp'); % 加入10dB高斯噪声

%直接逆滤波
W0 = deconvwnr(C,B,0);
W01 = uint8(W0);
W0out = imcrop(W01, rect);
imwrite(W0out,'directly recoverd.bmp'); % 对仅卷积点扩散函数的图像直接逆滤波复原

W1 = deconvwnr(D1,B,0);
W11 = uint8(W1);
W1out = imcrop(W11, rect);
imwrite(W1out,'directly 30dB recoverd.bmp'); % 对添加30dB高斯噪声的图像直接逆滤波复原

W2 = deconvwnr(D2,B,0);
W21 = uint8(W2);
W2out = imcrop(W21, rect);
imwrite(W2out,'directly 20dB recoverd.bmp'); % 对添加20dB高斯噪声的图像直接逆滤波复原

W3 = deconvwnr(D3,B,0);
W31 = uint8(W3);
W3out = imcrop(W31, rect);
imwrite(W3out,'directly 10dB recoverd.bmp'); % 对添加10dB高斯噪声的图像直接逆滤波复原

%Wiener滤波
X1 = deconvwnr(D1,B,1/30);
X11 = uint8(X1);
X1out = imcrop(X11, rect);
imwrite(X1out,'Wiener 30dB recoverd.bmp'); % 对添加30dB高斯噪声的图像Wiener滤波复原

X2 = deconvwnr(D2,B,1/20);
X21 = uint8(X2);
X2out = imcrop(X21, rect);
imwrite(X2out,'Wiener 20dB recoverd.bmp'); % 对添加20dB高斯噪声的图像Wiener滤波复原

X3 = deconvwnr(D3,B,1/10);
X31 = uint8(X3);
X3out = imcrop(X31, rect);
imwrite(X3out,'Wiener 10dB recoverd.bmp'); % 对添加10dB高斯噪声的图像Wiener滤波复原
```

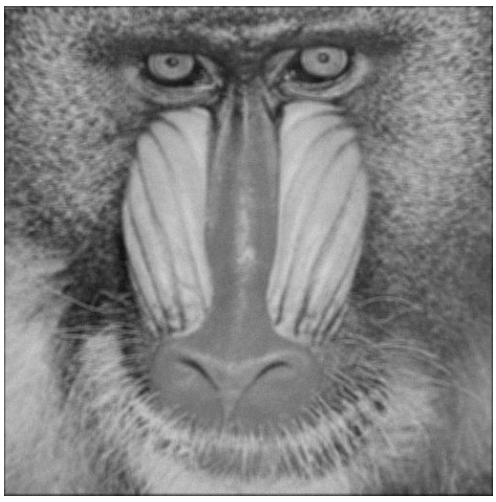
四、生成图像



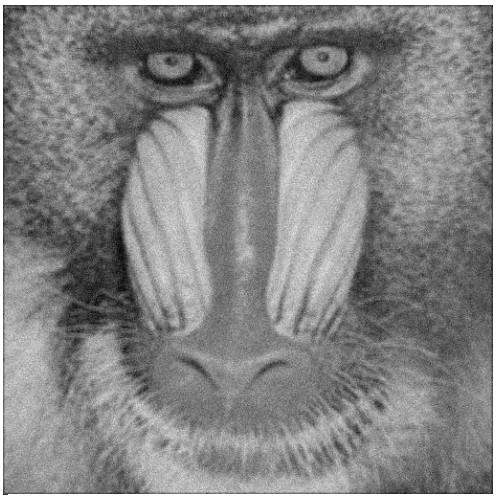
baboon.bmp



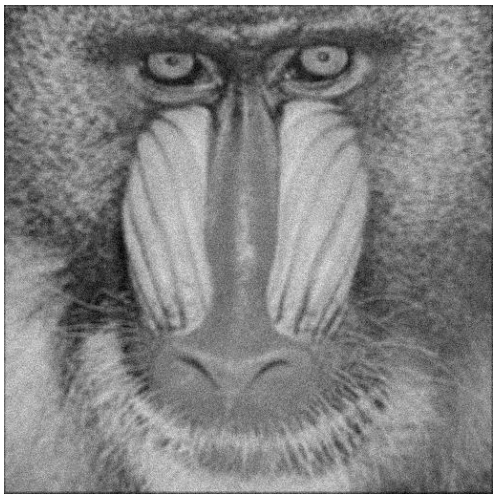
blurred.bmp



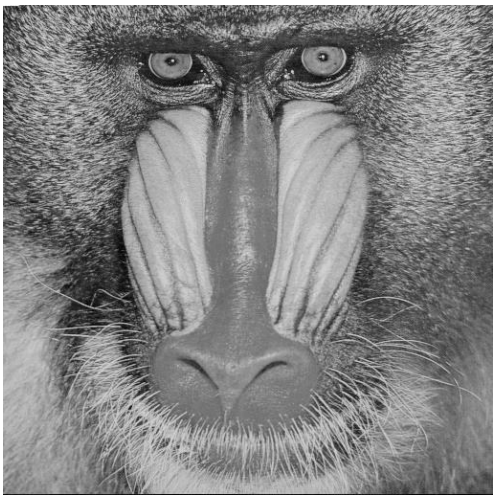
blurred and noised in 30dB.bmp



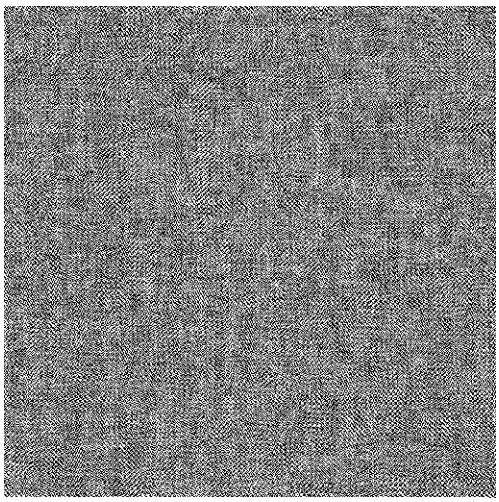
blurred and noised in 20dB.bmp



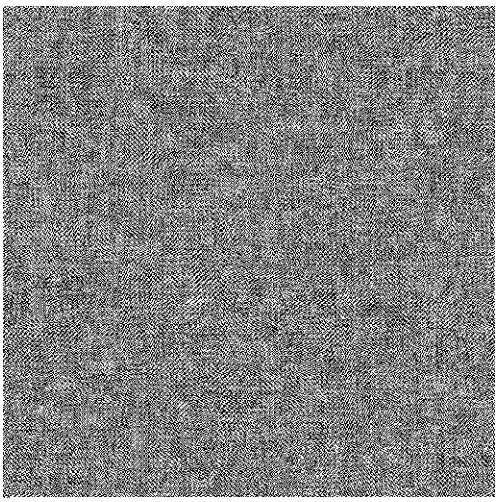
blurred and noised in 10dB.bmp



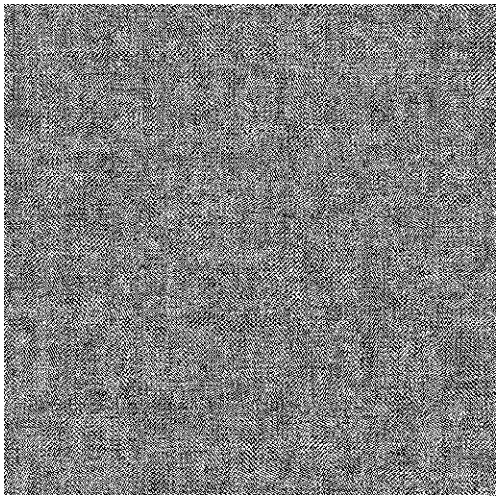
directedly recovered.bmp



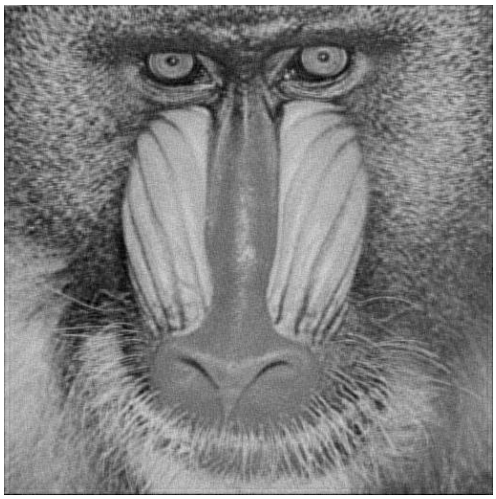
directedly 30dB recovered.bmp



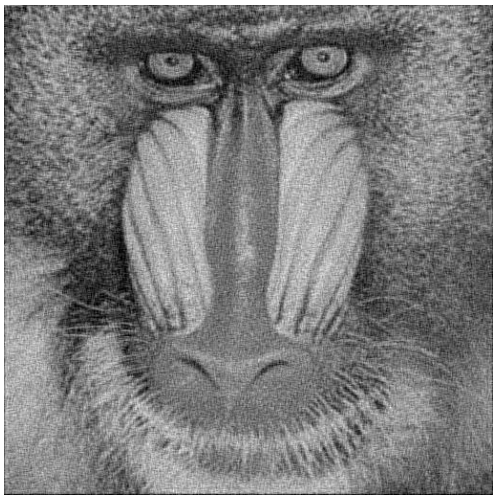
directedly 20dB recovered.bmp



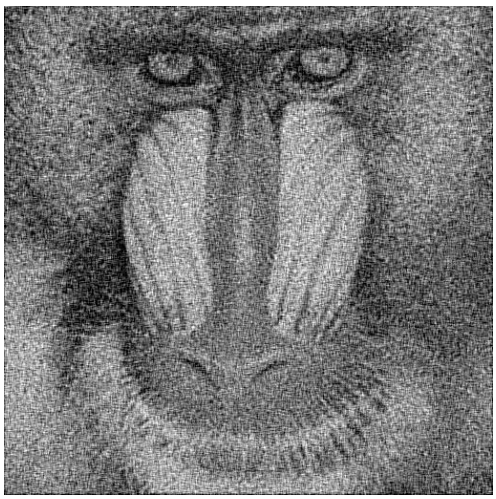
directedly 10dB recovered.bmp



Wiener 30dB recovered.bmp



Wiener 20dB recovered.bmp



Wiener 10dB recovered.bmp