数字图像处理第六次作业

姓名：张璞
班级：自动化 64
学号：2160700034
提交日期：2019 年 4 月 2 日

1. **在测试图像上产生高斯噪声 lena 图-需能指定均值和方差；并用滤波器（自选）恢复图像**

1) 问题分析

a) 高斯噪声:

所谓高斯噪声是指它的概率密度函数服从高斯分布（即正态分布）的一类噪声。一个高斯随机变量 z 的 PDF 可表示为：

$$P（z）= \frac{1}{\sqrt{2\pi\sigma}}\exp\left[\frac{(z-u)^2}{2\sigma^2}\right]$$

其中，z 代表灰度，u 是 z 的均值，是 z 的标准差。高斯噪声的灰度值多集中在均值附近。

b) 图像退化：

图像退化过程被建模为一个退化函数和一个加性噪声项，对一幅输入图像 f(x,y)进行处理，产生一副退化后的图像 g(x,y)。给定 g(x,y)和关于退化函数 H 的一些知识以及关于加性噪声项 η(x,y)的一些知识后，图像复原的目的就是获得原始图像的一个估计。如果 H 是一个线性的、位置不变的过程，那么空间域中的退化图像可由下式给出：

$$g(x,y)=h(x,y)★f(x,y)+\eta(x,y)$$

其中，h(x,y)是退化函数的空间表示；符号'★"表示空间卷积。等价的频率域表示为，

$$G(u,v)=H(u,v)F(u,v)+N(u,v)$$

当一副图像存在的唯一退化是噪声时，上式可表示为，

$$G(u,v)= F(u,v)+N(u,v)$$

当仅存在加性噪声的情况下，可以选择空间滤波的方法，例如算术均值滤波器、集合均值滤波器等。

c) 算术均值滤波器：

算术均值滤波器是最简单的滤波器。令 Sxy 表示中心在点(x,y)处，大小为 m×n 的矩形子图像窗口的一组坐标。算术均值滤波器在 Sxy 定义的区域中计算被污染的图像 g(x,y)的平均值。在点(x,y)处复原图像的值：

$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t)$$

这个操作可以使用大小为 m×n 的一个空间滤波器来实现，其所有系数均为其值的 1/mn。均值滤波器平滑一幅图像中的局部变化，虽然模糊了结果，但降低了噪声。

d) 几何均值滤波器：

几何均值滤波器复原一副图像由如下的表达式给出，

$$\hat{f}(x,y)=\left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{\frac{1}{mn}}$$

其中，每个复原的像素由子图像窗口中像素的乘积的$\frac{1}{mn}$次幂给出，几何均值滤波器实现的平滑可与算术均值滤波器的相比，但这种处理丢失的细节更少。

2) 实验过程

利用 matlab 的函数 imnoise 给图像添加不同均值与方差的高斯噪声，之后利用 fspecial 函数构造均值滤波器，最后使用 imfilter 函数进行滤波。

3) Matlab 函数说明

a) g=imnoise(f，‘gaussian’，m，var)

将均值为 m，方差为 var 的高斯噪声加到图像 f 上。m 的默认值是 0，var 默认值是 0.01。

b) g = imfilter(f, w, filtering_mode, boundary_options, size_options)

f 为输入图像，w 为滤波掩模，g 为滤波后图像。filtering_mode 用于指定在滤波过程中是使用“相关”还是“卷积”。boundary_options 用于处理边界充零问题，边界的大小由滤波器的大小确定。

c) h = fspecial(type，para)

type 指定算子的类型，para 指定相应的参数。

4) 实验结果

5)　结果分析

当高斯噪声均值不变为 0 时，随着方差增加，图像噪声越严重；当高斯噪声方差不变时，均值会影响到整个图像的灰度值，使整个图像变亮。与理论上均值和方差对图像的影响一致。

分别使用算术均值滤波器和几何均值滤波器对加噪图像进行恢复。两种方法

在一定程度上都可以降低噪声，但几何均值滤波器并未像算术均值滤波器那样使图像变得模糊。

2. **在测试图像 lena 图加入椒盐噪声（椒和盐噪声密度均是 0.1）；用学过的滤波器恢复图像；在使用反谐波分析 Q 大于 0 和小于 0 的作用；**

1) 问题分析

椒盐噪声的概率密度函数由下式给出，

$$\mathrm{p(z)}=\begin{cases} p_a\,, & z=a \\ p_b, & z=b \\ 1-p_a-p_b, 其它 \end{cases}$$

如果$b>a$，则灰度级b在图像中显示为一个亮点；反之，灰度级a在图像中显示为一个暗点。如果p_a和p_b两者均不可能为零，尤其是它们近似相等时，则脉冲噪声值将类似于在图像上随机分布的胡椒和盐粒微粒。

逆谐波均值滤波器基于如下的表达式产生一副复原的图像，

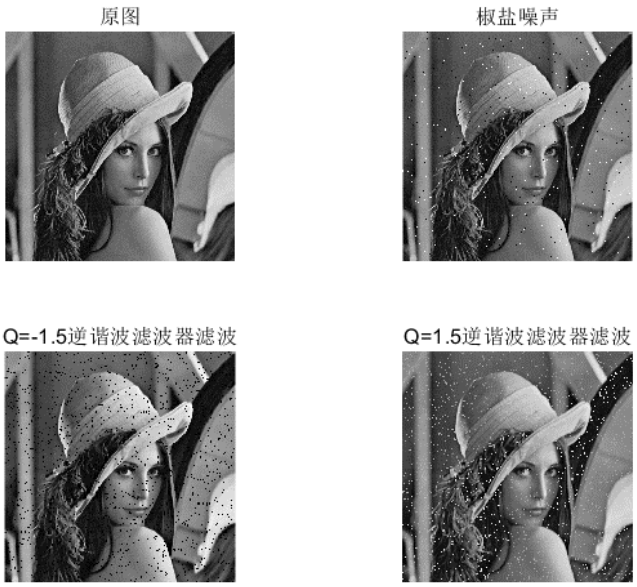
$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

其中，Q 为滤波器的阶数。这种滤波器适合减少或在实际中消除椒盐噪声的影响。当 Q 为正时，该滤波器消除胡椒噪声；当 Q 为负时，该滤波器消除盐粒噪声。但它不能同时去除两种噪声。

2) Matlab 函数说明

g=imnoise(f，‘salt & pepper’，d)给图像 f 添加椒盐噪声，其中 d 是噪声密度（即包含噪声值的图像区域的百分比）。因此，大约有 d*numel（f）个像素受到污染，默认的噪声密度为 0.05。

3) 实验结果



4) 结果分析

分别用 Q=1.5 和 Q=-1.5 的逆谐波滤波器对添加了噪声密度为 0.1 的 lena 图像进行滤波，两种滤波器都有很好的去除噪声的效果。正阶滤波器除了使暗区稍微有些淡化和模糊之外，都使背景变得更为清晰，而负阶正好相反。

3. 推导维纳滤波器

(a) 实现模糊滤波器如方程 Eq. (5.6-11).

图像的退化模型为：

$$x(\mathrm{n}_1,\mathrm{n}_2)=b(\mathrm{n}_1,\mathrm{n}_2)*s(\mathrm{n}_1,\mathrm{n}_2)+\mathrm{w}(\mathrm{n}_1,\mathrm{n}_2) \tag{1}$$

其中，s(n1,n2)为原始图像，b(n1,n2)为退化函数，w(n1,n2)为噪声函数，x(n1,n2)为退化的图像。并假设 s 与 w 不相关，w 为 0 均值的平稳随机过程。

图像的复原模型为：

$$\hat{s}(\mathrm{n}_1,\mathrm{n}_2)=\mathrm{h}(\mathrm{n}_1,\mathrm{n}_2)*x(\mathrm{n}_1,\mathrm{n}_2)=\sum_{l_1}\sum_{l_2}h(\mathrm{l}_1,\mathrm{l}_2)\times\mathrm{x}(\mathrm{n}_1-\mathrm{l}_1,\mathrm{n}_2-\mathrm{l}_2) \tag{2}$$

其中，$\hat{s}(\mathrm{n}_1,\mathrm{n}_2)$为恢复的图像，$\mathrm{h}(\mathrm{n}_1,\mathrm{n}_2)$为恢复滤波器。

误差度量为：

$$e^2=E\{(s(\mathrm{n}_1,\mathrm{n}_2)-\hat{s}(\mathrm{n}_1,\mathrm{n}_2))^2\} \tag{3}$$

基于正交性原理，若要求误差最小，则必有下式成立：

$$E\{\mathrm{e}(\mathrm{n}_1,\mathrm{n}_2)\times x^*(\mathrm{m}_1,\mathrm{m}_2)\}=0 \tag{4}$$

将（3）式带入（4）式有：

$$E\{\mathrm{s}(\mathrm{n}_1,\mathrm{n}_2)\times x^*(\mathrm{m}_1,\mathrm{m}_2)\}=E\{\hat{\mathrm{s}}(\mathrm{n}_1,\mathrm{n}_2)\times x^*(\mathrm{m}_1,\mathrm{m}_2)\} \tag{5}$$

即,

$$\begin{aligned}
&R_{sx}(n_1-m_1,n_2-m_2)\\
&=E\{\sum_{l_1}\sum_{l_2}h(l_1,l_2)\times x(n_1-l_1,n_2-l_2)\times x^*(m_1,m_2)\}\\
&=\sum_{l_1}\sum_{l_2}h(l_1,l_2)\times E\{x(n_1-l_1,n_2-l_2)\times x^*(m_1,m_2)\}\\
&=\sum_{l_1}\sum_{l_2}h(l_1,l_2)\times R_{xx}(n_1-l_1-m_1,n_2-l_2-m_2)\\
&=h(n_1-m_1,n_2-m_2)*R_{xx}(n_1-l_1-m_1,n_2-l_2-m_2)
\end{aligned} \tag{6}$$

换元得：

$$R_{sx}(\mathrm{n}_1,\mathrm{n}_2)=h(\mathrm{n}_1,\mathrm{n}_2)*R_{xx}(\mathrm{n}_1,\mathrm{n}_2) \tag{7}$$

等式两端同时取傅里叶变换得：

$$P_{sx}(\mathrm{w}_1,\mathrm{w}_2)=H(\mathrm{w}_1,\mathrm{w}_2)\times P_x(\mathrm{w}_1,\mathrm{w}_2) \tag{8}$$

即

$$H(\mathrm{w}_1,\mathrm{w}_2)=\frac{P_{sx}(\mathrm{w}_1,\mathrm{w}_2)}{P_x(\mathrm{w}_1,\mathrm{w}_2)} \tag{9}$$

公式（8）中，

$$\begin{aligned}
&R_{sx}(n_1,n_2)\\
&=\mathrm{E}\{\mathrm{s}(n_1+k_1,n_2+k_2)\times x^*(k_1,k_2)\}\\
&=\mathrm{E}\{\mathrm{s}(n_1+k_1,n_2+k_2)\times(\sum_{l_1}\sum_{l_2}b(l_1,l_2)\times s(k_1-l_1,k_2-l_2)+w(k_1,k_2))^*\}\\
&=\sum_{l_1}\sum_{l_2}b^*(l_1,l_2)\times E\{\mathrm{s}(n_1+k_1,n_2+k_2)\times \mathrm{s}^*(n_1+k_1,n_2+k_2)+\mathrm{s}(n_1+k_1,n_2+k_2)\times w^*(k_1,k_2)\}\\
&=\sum_{l_1}\sum_{l_2}b^*(l_1,l_2)\times R_s(n_1+l_1,n_2+l_2)\\
&=b^*(-n_1,-n_2)*R_s(n_1,n_2)
\end{aligned} \tag{10}$$

公式（10）两端同时取傅里叶变换得：

$$P_{sx}(\mathrm{w}_1,\mathrm{w}_2)=B^*(\mathrm{w}_1,\mathrm{w}_2)\times P_s(\mathrm{w}_1,\mathrm{w}_2) \tag{11}$$

公式（8）中，

$$\begin{aligned}
&R_x(n_1,n_2)\\
&=E\{x(n_1+k_1,n_2+k_2)\times x^*(k_1,k_2)\}\\
&=\mathrm{E}\{(\sum_{l_1}\sum_{l_2}b(l_1,l_2)\times s(n_1+k_1-l_1,n_2+k_2-l_2)\\
&\qquad +w(n_1+k_1,n_2+k_2))\times(\sum_{m_1}\sum_{m_2}b(m_1,m_2)\times s(k_1-m_1,k_2-m_2)+w(k_1,k_2))^*\}\\
&=\sum_{m_1}\sum_{m_2}\sum_{l_1}\sum_{l_2}b(l_1,l_2)b^*(m_1,m_2)\times E\{\mathrm{s}(n_1+k_1-l_1,n_2+k_2-l_2)\times \mathrm{s}^*(k_1-m_1,k_2-m_2)+R_w(n_1,n_2)\\
&=\sum_{m_1}\sum_{m_2}b^*(m_1,m_2)\sum_{l_1}\sum_{l_2}b(l_1,l_2)R_s(n_1+m_1-l_1,n_2+m_2-l_2)+R_w(n_1,n_2)\\
&=\sum_{m_1}\sum_{m_2}b^*(m_1,m_2)\times(\mathrm{b}(n_1+m_1,n_2+m_2)*R_s(n_1+m_1,n_2+m_2))+R_w(n_1,n_2)\\
&=\ \mathrm{b}^*(-n_1,-n_2)*\mathrm{b}(n_1,n_2)*R_s(n_1,n_2)+R_w(n_1,n_2)
\end{aligned} \tag{12}$$

公式（12）两端同时取傅里叶变换，

$$P_x(\mathrm{w}_1,\mathrm{w}_2)=\left|B(\mathrm{w}_1,\mathrm{w}_2)\right|^2\times P_s(\mathrm{w}_1,\mathrm{w}_2)+P_w(\mathrm{w}_1,\mathrm{w}_2) \tag{13}$$

将（11）式和（13）式带入（8）式得，

$$\mathrm{H}(\mathrm{w}_1,\mathrm{w}_2)=\frac{B^*(\mathrm{w}_1,\mathrm{w}_2)\times P_s(\mathrm{w}_1,\mathrm{w}_2)}{\left|B(\mathrm{w}_1,\mathrm{w}_2)\right|^2\times P_s(\mathrm{w}_1,\mathrm{w}_2)+P_w(\mathrm{w}_1,\mathrm{w}_2)} \tag{14}$$

将符号化成与书中一致的表示，

$$W(u,v)=\frac{H^*(u,v)\times|F(u,v)|^2}{|H(u,v)|^2\times|N(u,v)|^2+|N(u,v)|^2}$$
$$=\frac{H^*(u,v)}{|H(u,v)|^2+\frac{|N(u,v)|^2}{|F(u,v)|^2}}$$

（15）

故表达式由下式给出，

$$\hat{F}(u,v)=\left[\frac{H^*(u,v)\times S_f(u,v)}{|H(u,v)|^2\times S_f(u,v)+S_n(u,v)}\right]G(u,v)$$
$$=\left[\frac{H^*(u,v)}{|H(u,v)|^2+S_n(u,v)/S_f(u,v)}\right]G(u,v)$$
$$=\left[\frac{1}{H(u,v)}\times\frac{|H(u,v)|^2}{|H(u,v)|^2+S_n(u,v)/S_f(u,v)}\right]G(u,v)$$

（16）

(b) **模糊 lena 图像：45 度方向，T=1；**

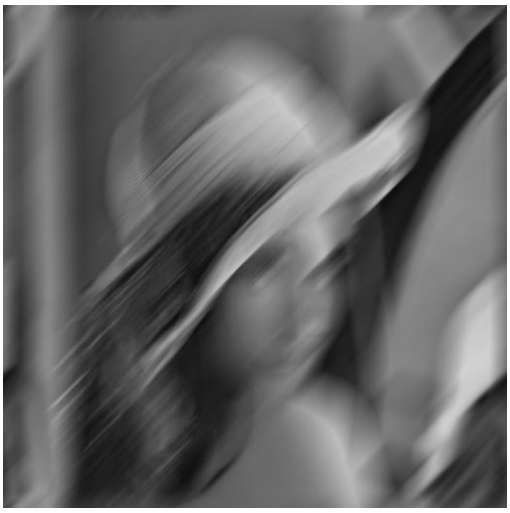
模糊滤波器的频域表达式为：

$$H(u,v)=\frac{T}{\pi(ua+vb)}\sin[\pi(ua+vb)]e^{-j\pi(ua+vb)}$$

故实现该滤波器，只需先将输入图像进行傅里叶变换并移至图像中心，之后将图像的傅里叶变换和模糊滤波器的傅里叶变换进行阵列相乘，将得到的结果经过傅里叶反变换返回到空间域即可实现该滤波器。

lena.bmp 原始图像　　lena 运动模糊的结果（a=0.1，b=0.1；T=1）

lena.bmp 原始图像　　lena 运动模糊的结果（调用 MATLAB 中的函数）



(c)再模糊的 lena 图像中增加高斯噪声，均值= 0 ，方差=10 pixels 以产生模糊图像；

lena 运动模糊的结果（a=0.1，b=0.1；T=1）添加高斯噪声的结果（均值为 0，方差为 0.01）

lena 运动模糊的结果（MATLAB 版）

添加高斯噪声的结果（MATLAB 版）

(d)**分别利用方程 Eq. (5.8-6)和(5.9-4)，恢复图像；并分析算法的优缺点.**

维纳滤波的结果：

lena 运动模糊+高斯噪声.bmp

维纳滤波结果（K=0.06）

维纳滤波结果（K=0.01）

维纳滤波结果（K=0.5）

lena 运动模糊+高斯噪声（MATLAB 版）

维纳滤波结果（MATLAB 版）

约束最小二乘方滤波的结果：

lena 运动模糊+高斯噪声（MATLAB 版）

约束最小二乘滤波结果（MATLAB 版）

首先分别通过自己编写的模糊函数和 MATLAB 中提供的 imfilter 和 fspecial 函数配合使用对图像 lena 进行了模糊滤波。发现套用书上的公式图像是斜向下 45 度运动模糊，而 MATLAB 中函数模糊的结果是斜向上 45 度运动模糊，不过这不是重点可以接受，模糊的基本效果还是一致的。之后调用 imnoise 函数对两幅图像加入高斯噪声，得到第二问的结果。

之后分别使用自己编写的函数和 MATLAB 中提供的 deconvwnr 函数进行维纳滤波。调用 MATLAB 中函数滤波后的图像得到了一定的改善，运动模糊的影响基本被消除，但噪声的影响仍然较大，导致图像质量下降；对于自己编写的维纳滤波函数，难点在于寻找令信噪比最大的 K 值，报告中显示了部分 K 值对应的滤波结果，其中 K=0.06，为信噪比最大时的滤波结果，从结果看，视觉上的效果并不是很理想，要想达到更好的效果可能需要寻找更加合适的 K 值。

最后采用 MATLAAB 提供的 deconvreg 函数进行约束最小二乘方滤波。从滤波后的结果看，约束最小二乘方滤波得到了比维纳滤波更好的结果，尤其是对噪声的滤除。

参考文献

[1] 冈萨雷斯.数字图像处理（第三版）北京：电子工业出版社，2011

[2] 周品.MATLAB 数字图像处理 北京：清华大学出版社，2012

[3] 杨杰.数字图像处理及 MATLAB 实现 北京：电子工业出版社，2010

附录

1. 添加高斯噪声（算术均值滤波与几何均值滤波）

```
clc;clear;
I=imread('lena.bmp');
I=im2double(I);
h=figure(1);
subplot(2,2,1);
imshow(I,[]);%自动产生适当比例的显示图像
title('Ô-í¼');
%添加高斯噪声
subplot(2,2,2);
I_noise=double(imnoise(I,'gaussian',0,0.01));
imshow(I_noise,[]);title('¸ßË¹ÔëÉù(u=0£¬s^2=0.01)');
%均值滤波
subplot(2,2,3);
I_3=fspecial('average',[3,3]);
I_3=imfilter(I_noise,I_3);%(待处理矩阵，滤波器)
imshow(I_3,[]);title('3*3 滤波器');
subplot(2,2,4);
I_=exp(imfilter(log(I_noise),fspecial('average',3)));%算术均值滤波
imshow(I_,[]);title('3*3 几何均值滤波');
```

2. 添加椒盐噪声（逆谐波滤波）

```
clc;clear;
I=imread('lena.bmp');
I=im2double(I);
figure(2);
%显示原图
subplot(2,2,1);
imshow(I,[])；
title('Ô-í¼');
%添加椒盐信号
subplot(2,2,2);
I_noise=double(imnoise(I,'salt & pepper',0.01));%salt & pepper 注意中间的空格
imshow(I_noise,[]);title('椒盐噪声');
```

```
subplot(2,2,3);
Q=-1.5;
I_mean=imfilter(I_noise.^(Q+1),fspecial('average',3))./imfilter(I_noise.^Q,fspecial('averag
e',3));
imshow(I_mean,[]);title('Q=-1.5 逆谐波滤波器');
 subplot(2,2,4);
Q=1.5;
I_mean=imfilter(I_noise.^(Q+1),fspecial('average',3))./imfilter(I_noise.^Q,fspecial('averag
e',3));
imshow(I_mean,[]);title('Q=1.5 逆谐波滤波器');
```

3. 添加运动模糊的图像

```
I=imread('lena.bmp');
figure(1);
imshow(I);
title('lena.bm 原始图像');
imwrite(I,'lena 原始图像.bmp');
f=double(I);
F=fft2(f);
F=fftshift(F);
[M,N]=size(F);
a=0.1;b=0.1;T=1;
for u=1:M
    for v=1:N
        H(u,v)=(T/(pi*(u*a+v*b)))*sin(pi*(u*a+v*b))*exp(-sqrt(-1)*pi*(u*a+v*b));
      G(u,v)=H(u,v)*F(u,v);
    end
end
G=ifftshift(G);
g=ifft2(G);
g=256.*g./max(max(g));
g=uint8(real(g));
figure(2);
imshow(g);
title('运动模糊化 lena.bmp');
imwrite(g,'lena 运动模糊化结果.bmp');
```

4．Wiener_filter.m(题目 2 维纳滤波器自编版)

```
I=imread('lena 运动模糊+高斯噪声.bmp');
figure(1);
imshow(I);
```

```
    title('lena 运动模糊+高斯噪声');
    imwrite(I,'lena 运动模糊+高斯噪声.bmp');
    g=double(I);
    G=fft2(g);
    G=fftshift(G);
    [M,N]=size(G);
    a=0.1;b=0.1;T=1;K=0.0259;
    for u=1:M
        for v=1:N   H(u,v)=(T/(pi*(u*a+v*b)))*sin(pi*(u*a+v*b))*exp(-
sqrt(-1)*pi*(u*a+v*b));
         F(u,v)=1/H(u,v)*(abs(H(u,v)))^2/((abs(H(u,v)))^2+K)*G(u,v);
        end
    end
    F=ifftshift(F);
    f=ifft2(F);
    f=256.*f./max(max(f));
    f=uint8(real(f));
    figure(2);
    imshow(f);
    title('维纳滤波的将结果');
    imwrite(f,'维纳滤波的结果(K=0.0259).bmp');
```

5．wiener_filter_k.m(题目 2 维纳滤波器寻找信噪比最大的 κ 值)

```
    I=imread('lena 运动模糊+高斯噪声.bmp');
    figure(1);
    imshow(I);
    title('lena 运动模糊+高斯噪声');
    imwrite(I,'lena 运动模糊+高斯噪声.bmp');
    g=double(I);
    G=fft2(g);
    G=fftshift(G);
    [M,N]=size(G);
    a=0.1;b=0.1;T=1;i=1;
    format long
    for k=0.01:0.01:0.11
        for u=1:M
            for v=1:N
H(u,v)=(T/(pi*(u*a+v*b)))*sin(pi*(u*a+v*b))*exp(-sqrt(-
1)*pi*(u*a+v*b));
F(u,v)=(1/H(u,v))*((abs(H(u,v)))^2/((abs(H(u,v)))^2+k))*G(u,v);
            end
        end
        F=ifftshift(F);
        f=ifft2(F);
        f=256.*f./max(max(f));
        f=uint8(real(f));
        figure;
```

```
    imshow(f);
    title('维纳滤波的结果');
    e=f-uint8(g);
    SNR(i)=sum(sum(g.^2))/sum(sum(e.^2));
    i=i+1;
end
idx=find(max(SNR))
```

6．wiener_filter_matlab.m(题目 2 维纳滤波器 MATLAB 版)

```
I=imread('lena.bmp');
H=fspecial('motion',50,45);
I1=imfilter(I,H,'circular','conv');
figure(1);
imshow(I1);
title('运动模糊后的 lena.bmp(角度为 45 度)');
imwrite(I1,'lena 运动模糊（调用 matlab 中的函数）.bmp');
I2=imnoise(I1,'gaussian',0,0.01);
figure(2)
imshow(I2);
title('加噪并模糊的 lena.bmp');
imwrite(I2,'lena 运动模糊+高斯噪声（调用 matlab 中的函数 0.bmp');
figure(3);
noise=imnoise(zeros(size(I)),'gaussian',0,0.01);
NSR=sum(noise(:).^2)/sum(im2double(I(:)).^2);
I3=deconvwnr(I2,H,NSR);
imshow(I3);
title('维纳滤波的结果');
imwrite(I3,'lena 维纳滤波的结果（调用 MATLAB 中的函数）.bmp');
```

7. yueshuzuixiaercheng_filter_matlab.m

```
I=imread('lena.bmp');
h=fspecial('motion',50,45);
I1=imfilter(I,h,'circular','conv');
I2=imnoise(I1,'gaussian',0,0.01);
figure(1);
imshow(I2);
title('lena 运动模糊+高斯噪声');
imwrite(I2,'lena 运动模糊+高斯噪声（MATLAB 版）.bmp');
V=0.0001;
NoisePower=V*prod(size(I));
[g,LAGRA]=deconvreg(I1,h,NoisePower);
figure(2);
imshow(g);
title('约束最小二乘滤波的结果（MATLAB 版）');
imwrite(g,'约束最小二乘滤波的结果（MATLAB 版）.bmp
```