实验一 图像复原MATLAB编程实验

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 李雨梦 | 学号 | 231101128 | 专业 | 电子信息 |
| 班级 | 电子231 | 教师 | 张福如 | 评分 |  |

第一部分 目的与要求

1.1 实验目的

（1）掌握基于傅里叶变换的图像复原(逆滤波复原)方法,了解其原理和应用场景。通过实现基于傅里叶变换的图像复原算法,加深对该方法的理解。

（2）学习基于维纳滤波的图像复原技术,探索该方法在不同噪声环境下的性能。通过编程实现维纳滤波算法,分析其优缺点。

（3）比较不同图像复原方法的性能,探讨各种方法的适用场景,为实际应用提供参考。

1.2 实验内容

（1）设计基于傅里叶变换的图像复原(逆滤波复原)滤波器，对不同类型的退化图像(如模糊、噪声等)进行复原，并分析复原结果。

（2）编写维纳滤波算法,实现图像复原，在不同噪声条件下测试算法性能。

1.3 编程涉及的部分Matlab函数

（1）%：注释符号，在其后的同一行中的内容是注释，不会被执行。

（2）clc：清除命令窗口中的内容。

（3）clear all：清除工作空间的所有变量、函数和MEX文件。

（4）close all：关闭所有的Figure窗口。

（5）imread('path\_filename')：读入指定路径及文件名为path\_filename的图像。

（6）[m, n, color]=size(I)：获取图像矩阵I的行数h和列数w及颜色数color。当color值等于3时，说明图像I是彩色图像。缺省color的形式为[m, n]=size(I)。

（7）double(x)：将其它类型的数据x转换为双精度型数据。在MATLAB的矩阵运算中，要求所有的运算变量为double型(双精度型)。

（8）abs (x)：求x的绝对值。

（9）subplot(m, n, p)：将多幅图像输出到一个平面（figure）上。其中，m表示将图像排成m行，n表示将图像排成n列，也即整个figure中有n幅图像排成一行，一共m行。p表示图所在的位置，p=1表示从左到右从上到下的第一个位置。

（10）imshow(I,[low high])：显示灰度图像 I，并以二元素向量[low high]指定显示范围。显示方法是将值 low（以及任何小于 low 的值）显示为黑色；将值 high（以及任何大于 high 的值）显示为白色。缺省[low high]的imshow(I)为显示灰度图像。

（11）title(' heading')：给输出显示的图像加标题，单引号内的内容为图像的标题内容。

（12）fft2, ifft2：进行二维快速傅里叶变换和逆变换。

（13）fftshift, ifftshift：将傅里叶频谱的零频率分量移动到频谱中心。

（14）deconvwnr函数：使用维纳滤波器对图像进行去模糊，I表示输入图像，具有以下几种调用形式：

J=deconvwnr(I,PSF): PSF为点扩散函数矩阵；

J=deconvwnr(I,PSF,NSR): NSR为信噪比，默认为0。

（15）imnoise：向图像添加噪声。

1.4 实验要求

（1）简述实验内容原理。

（2）编写完成实验内容的Matalb程序。

（3）通过Matlab环境中的程序截图、按实验内容要求对原图像和实验结果图像截图，说明实验完成情况。

（4）对实验过程的相关问题及实验工作进行总结，最好有实验收获和体会等部分内容。

第二部分 实验报告

根据实验内容及相关要求，我完成了基于傅里叶变换和维纳滤波的图像复原的matlab程序编程及验证，现将实验中的相关情况汇总如下。

2.1a 基于傅里叶变换的图像复原(逆滤波复原)的原理及编程思路  
  


(1)对退化图像g(x, y)作二维离散傅立叶变换，得到G(u, v)；

(2)计算系统点扩散函数h(x, y)的二维傅立叶变换，得到H(u, v);

(3)逆滤波计算F(u,v);

(4)计算F(u,v)的逆傅立叶变换，求得f(u,v)。

2.1b 编写的Matlab源程序  
 %% 图像复原实验 - 逆滤波和维纳滤波

% 实验环境: MATLAB 2013

%% 清理工作区

clc; % 清除命令窗口

clear all; % 清除所有变量

close all; % 关闭所有图形窗口

%% 第一部分: 图像加载和预处理

% 读取原始图像

original\_img = imread('lena.jpg'); % 请确保图像文件在MATLAB路径中

if size(original\_img, 3) == 3 % 如果是彩色图像则转为灰度

original\_img = rgb2gray(original\_img);

end

original\_img = double(original\_img); % 转换为double类型

% 显示原始图像

figure(1);

subplot(2,3,1);

imshow(original\_img, []);

title('原始图像');

%% 第二部分: 图像退化（模拟模糊和噪声）

% 创建运动模糊的点扩散函数(PSF)

PSF = fspecial('motion', 15, 45); % 长度为15像素，角度为45度的运动模糊

% 应用模糊

blurred\_img = imfilter(original\_img, PSF, 'conv', 'circular');

% 添加高斯噪声

noise\_mean = 0;

noise\_var = 0.001;

blurred\_noisy\_img = imnoise(blurred\_img/255, 'gaussian', noise\_mean, noise\_var);

blurred\_noisy\_img = blurred\_noisy\_img \* 255; % 将图像重新缩放到0-255范围

% 显示退化后的图像

subplot(2,3,2);

imshow(blurred\_img, []);

title('模糊图像');

subplot(2,3,3);

imshow(blurred\_noisy\_img, []);

title('模糊+噪声图像');

%% 第三部分: 逆滤波复原

% 计算傅里叶变换

G = fft2(blurred\_img); % 模糊图像的傅里叶变换

H = fft2(PSF, size(G,1), size(G,2)); % PSF的傅里叶变换，扩展到与图像相同大小

% 逆滤波

F\_hat = G ./ H; % 逆滤波公式

% 计算逆傅里叶变换

inverse\_filtered\_img = real(ifft2(F\_hat));

% 显示逆滤波结果

subplot(2,3,4);

imshow(inverse\_filtered\_img, []);

title('逆滤波复原结果(无噪声)');

%% 第四部分: 维纳滤波复原

% 计算噪声功率和信号功率

signal\_var = var(original\_img(:));

noise\_power = noise\_var \* 255^2; % 由于我们之前将图像缩放到0-255

% 使用MATLAB内置的维纳滤波函数

wiener\_filtered\_img = deconvwnr(blurred\_noisy\_img, PSF, noise\_power/signal\_var);

% 显示维纳滤波结果

subplot(2,3,5);

imshow(wiener\_filtered\_img, []);

title('维纳滤波复原结果');

%% 第五部分: 性能比较

% 计算并显示均方误差(MSE)

mse\_inverse = sum(sum((original\_img - inverse\_filtered\_img).^2)) / numel(original\_img);

mse\_wiener = sum(sum((original\_img - wiener\_filtered\_img).^2)) / numel(original\_img);

fprintf('逆滤波复原的MSE: %.2f\n', mse\_inverse);

fprintf('维纳滤波复原的MSE: %.2f\n', mse\_wiener);

% 显示所有结果对比

figure(2);

subplot(2,2,1);

imshow(original\_img, []);

title('原始图像');

subplot(2,2,2);

imshow(blurred\_noisy\_img, []);

title('退化图像(模糊+噪声)');

subplot(2,2,3);

imshow(inverse\_filtered\_img, []);

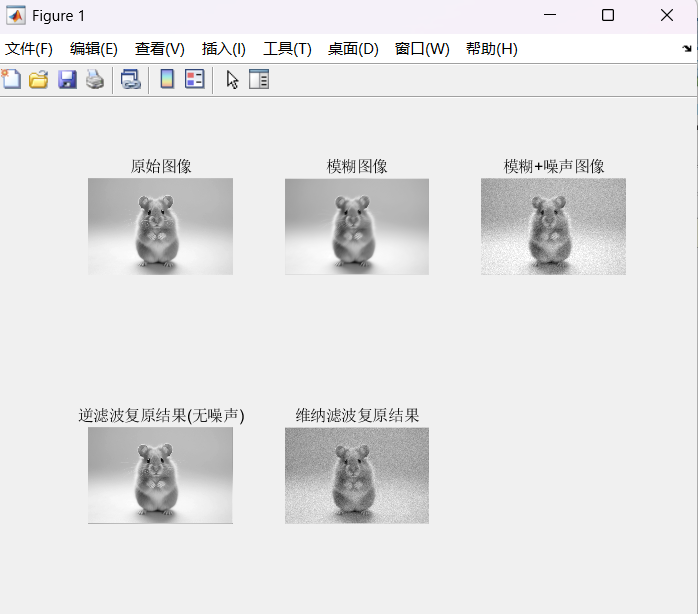
title(['逆滤波结果, MSE=', num2str(mse\_inverse, '%.2f')]);

subplot(2,2,4);

imshow(wiener\_filtered\_img, []);

title(['维纳滤波结果, MSE=', num2str(mse\_wiener, '%.2f')]);

2.1c 程序运行结果截屏



2.2a 基于维纳滤波的图像复原的原理及编程思路  
Wienner滤波恢复的思想是在假设图象信号可近似地看成平稳随机过程的前提下，按照使恢复的图象与原图象的均方差最小原则来恢复图象。  
  
（1）计算图象g(x,y)的二维离散付里叶变换得到G(u,v)；

（2）计算点扩散函数的二维离散付里叶变换；

（3）估算图象的功率谱密度和噪声谱密度；

（4）计算图象的估计值；

（5）逆变换，得到恢复后的图象。

2.2b 编写的Matlab源程序  
 %% 图像复原实验 - 逆滤波和维纳滤波

% 实验环境: MATLAB 2013

%% 清理工作区

clc; % 清除命令窗口

clear all; % 清除所有变量

close all; % 关闭所有图形窗口

%% 第一部分: 图像加载和预处理

% 读取原始图像

original\_img = imread('lena.jpg'); % 请确保图像文件在MATLAB路径中

if size(original\_img, 3) == 3 % 如果是彩色图像则转为灰度

original\_img = rgb2gray(original\_img);

end

original\_img = double(original\_img); % 转换为double类型

% 显示原始图像

figure(1);

subplot(2,3,1);

imshow(original\_img, []);

title('原始图像');

%% 第二部分: 图像退化（模拟模糊和噪声）

% 创建运动模糊的点扩散函数(PSF)

PSF = fspecial('motion', 15, 45); % 长度为15像素，角度为45度的运动模糊

% 应用模糊

blurred\_img = imfilter(original\_img, PSF, 'conv', 'circular');

% 添加高斯噪声

noise\_mean = 0;

noise\_var = 0.001;

blurred\_noisy\_img = imnoise(blurred\_img/255, 'gaussian', noise\_mean, noise\_var);

blurred\_noisy\_img = blurred\_noisy\_img \* 255; % 将图像重新缩放到0-255范围

% 显示退化后的图像

subplot(2,3,2);

imshow(blurred\_img, []);

title('模糊图像');

subplot(2,3,3);

imshow(blurred\_noisy\_img, []);

title('模糊+噪声图像');

%% 第三部分: 逆滤波复原

% 计算傅里叶变换

G = fft2(blurred\_img); % 模糊图像的傅里叶变换

H = fft2(PSF, size(G,1), size(G,2)); % PSF的傅里叶变换，扩展到与图像相同大小

% 逆滤波

F\_hat = G ./ H; % 逆滤波公式

% 计算逆傅里叶变换

inverse\_filtered\_img = real(ifft2(F\_hat));

% 显示逆滤波结果

subplot(2,3,4);

imshow(inverse\_filtered\_img, []);

title('逆滤波复原结果(无噪声)');

%% 第四部分: 维纳滤波复原

% 计算噪声功率和信号功率

signal\_var = var(original\_img(:));

noise\_power = noise\_var \* 255^2; % 由于我们之前将图像缩放到0-255

% 使用MATLAB内置的维纳滤波函数

wiener\_filtered\_img = deconvwnr(blurred\_noisy\_img, PSF, noise\_power/signal\_var);

% 显示维纳滤波结果

subplot(2,3,5);

imshow(wiener\_filtered\_img, []);

title('维纳滤波复原结果');

%% 第五部分: 性能比较

% 计算并显示均方误差(MSE)

mse\_inverse = sum(sum((original\_img - inverse\_filtered\_img).^2)) / numel(original\_img);

mse\_wiener = sum(sum((original\_img - wiener\_filtered\_img).^2)) / numel(original\_img);

fprintf('逆滤波复原的MSE: %.2f\n', mse\_inverse);

fprintf('维纳滤波复原的MSE: %.2f\n', mse\_wiener);

% 显示所有结果对比

figure(2);

subplot(2,2,1);

imshow(original\_img, []);

title('原始图像');

subplot(2,2,2);

imshow(blurred\_noisy\_img, []);

title('退化图像(模糊+噪声)');

subplot(2,2,3);

imshow(inverse\_filtered\_img, []);

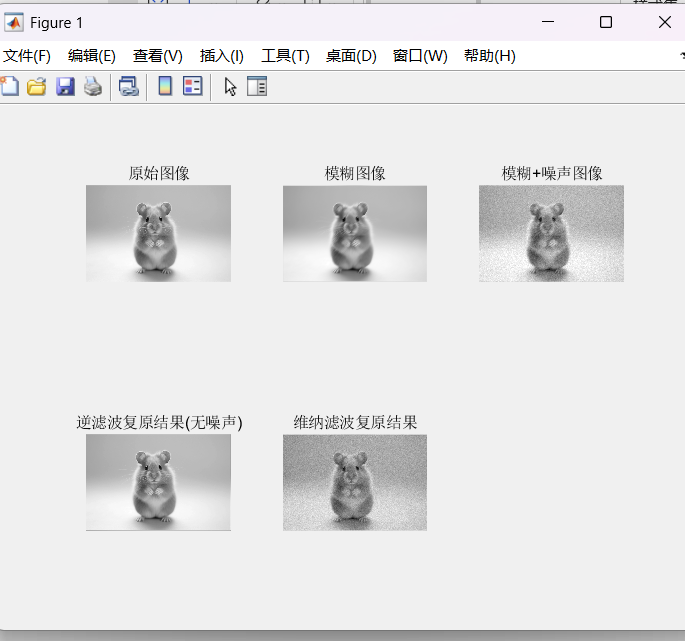
title(['逆滤波结果, MSE=', num2str(mse\_inverse, '%.2f')]);

subplot(2,2,4);

imshow(wiener\_filtered\_img, []);

title(['维纳滤波结果, MSE=', num2str(mse\_wiener, '%.2f')]);

2.2c 程序运行结果截屏



2.3 实验总结

逆滤波和维纳滤波是两种经典的图像复原方法，各有特点：逆滤波原理简单直接，通过在频域中对退化图像进行逆变换来恢复原始图像，在无噪声条件下效果理想且计算效率高，但对噪声极其敏感，容易放大高频噪声导致复原图像质量下降；维纳滤波则基于最小均方误差准则，通过引入信噪比参数在复原过程中同时考虑图像和噪声的统计特性，能有效抑制噪声干扰，在含噪声环境下表现更稳定，但需要估计噪声功率和信号功率，且计算复杂度略高。实验结果表明，对于无噪声的模糊图像，逆滤波可获得较好复原效果，而对于含噪声的退化图像，维纳滤波凭借其噪声抑制能力通常能获得更优的复原质量，体现了两种方法在不同场景下的适用性差异。

