二〇二三～二〇二四学年 第二学期

信息科学与工程学院

数字图像处理
实验报告书

班　　级：_____

学　　号：_____

姓　　名：Henry　Lee

指导教师：_____

二〇二四年六月

实验 1　MATLAB 图像的点运算

一、实验目的：

1. 了解点运算的定义及其作用的性质；
2. 熟练掌握灰度的线性变换；
3. 理解和掌握直方图及直方图均衡化的原理和应用。

二、实验要求：

1. 启动 MATLAB 程序，对图像文件分别进行灰度线性变换、直方图、直方图均衡化操作；
2. 记录和整理实验报告。

三、实验内容：

1. 具体实验过程：

（1）启动 MATLAB 程序，导入图像文件；

（2）计算并绘制原始图像的直方图，可以自己设计一个计算直方图的函数。也可调用 Matlab 自带的 imhist()函数；

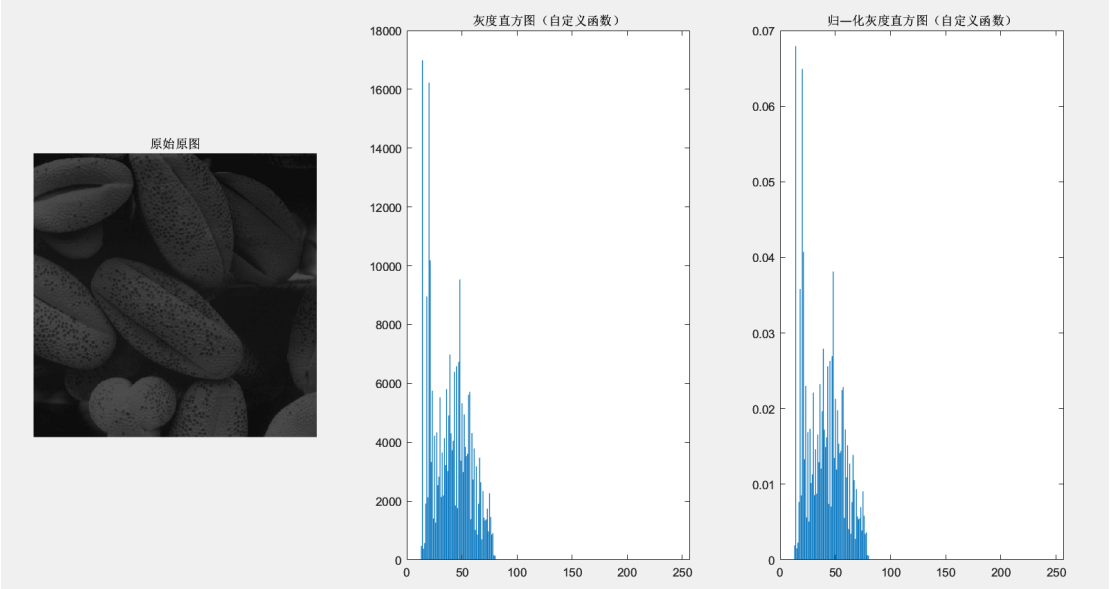
（3）对图像进行灰度转换，并绘制变换后的直方图；

（4）实现直方图均衡化算法，可调用 Matlab 自带的 histeq()函数；

（5）上述步骤封装到一个循环中，实现对实验所给的三幅图像同时处理。

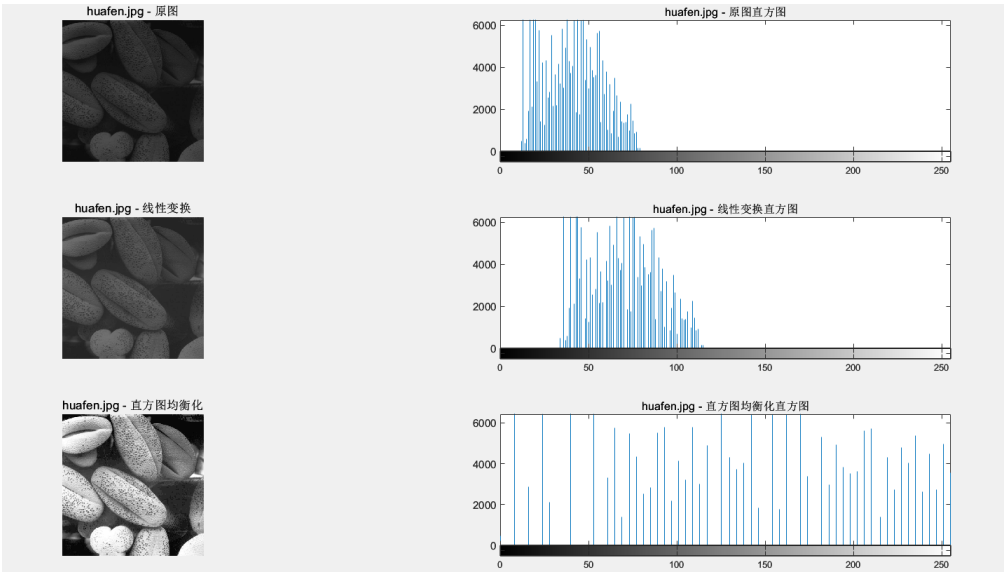
2. 实验的原始图像和结果图像

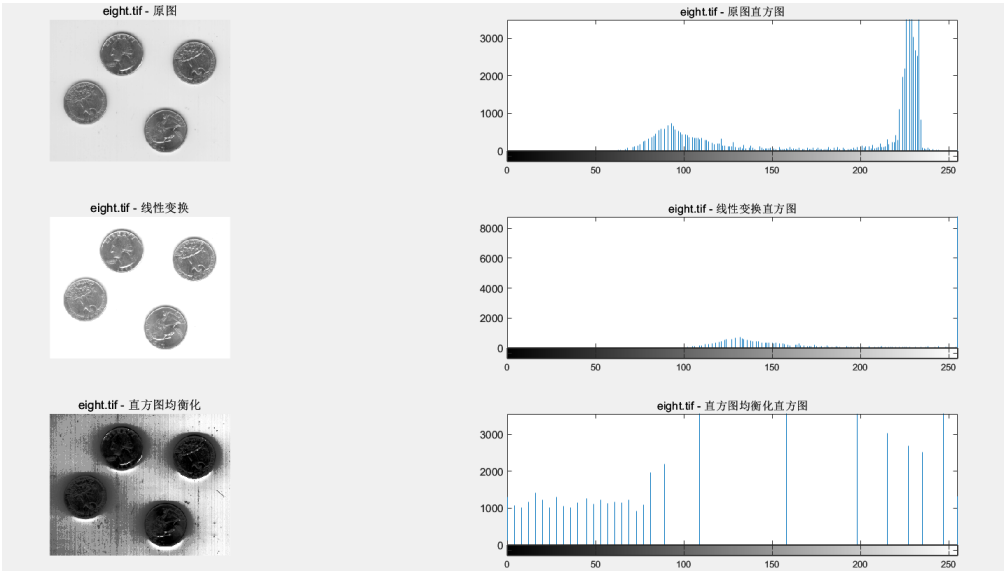
（1）自己写的直方图绘制函数 MyImhist()的效果：

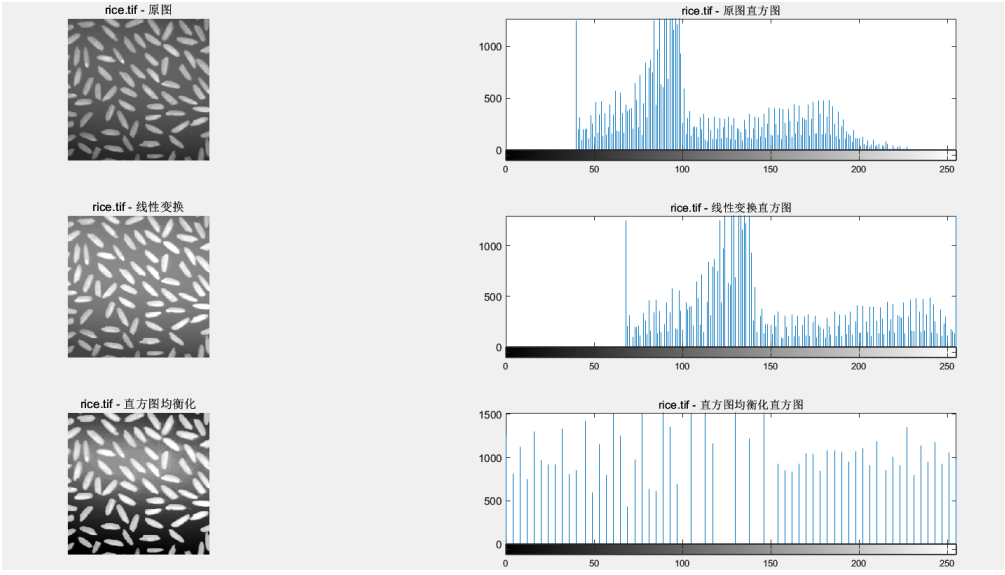


（2）三幅图像的灰度线性变换、直方图、直方图均衡化效果：

（使用 Matlab 自带的 imhist()、histeq()等函数）







本次实验相关代码：

编辑器 - D:\Code\DIP\MyImhist.m

Test_1.m | MyImhist.m

```m
function MyImhist(img)
% 自己写的直方图绘制函数

% 如果是彩色图像，转换为灰度图像
if size(img, 3) == 3
   img = rgb2gray(img);
end

[m,n] = size(img);
N=zeros(1,256);
for i=1:m
    for j=1:n
        k=img(i,j);
        N(k+1)=N(k+1)+1;
    end
end
sum=m*n;
p=N/sum;    %统计频率
figure;
subplot(1,3,1),imshow(img), title('原始原图');
subplot(1,3,2),bar(N);title('灰度直方图（自定义函数）');
subplot(1,3,3),bar(p),title('归一化灰度直方图（自定义函数）');
end
```

编辑器 - D:\Code\DIP\Test_1.m

Test_1.m

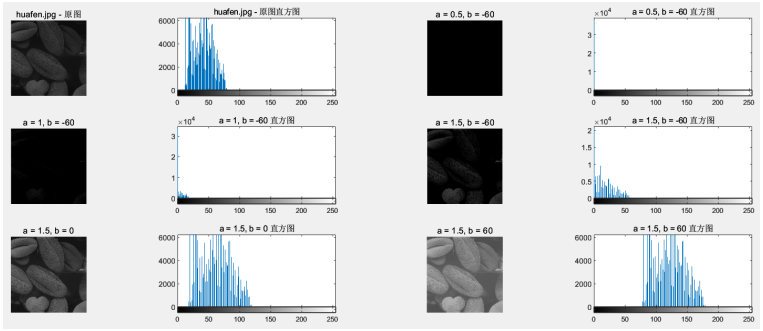
```m
% 读取图像
img1 = imread('images\huafen.jpg');
img2 = imread('images\eight.tif');
img3 = imread('images\rice.tif');

% 图像列表
images = {img1, img2, img3};
image_names = {'huafen.jpg', 'eight.tif', 'rice.tif'};

% 循环处理
for i = 1:length(images)
    img = images{i};

    % 如果是彩色图像，转换为灰度图像
    if size(img, 3) == 3
        img = rgb2gray(img);
    end

    % 灰度线性变换
    % 线性变换 y = a*x + b
    a = 1.5; % 增加对比度
    b = 20;  % 提高亮度
    gray_img = a * double(img) + b;
    gray_img = uint8(gray_img);

    % 计算并显示直方图
    figure;
    subplot(3, 2, 1);
    imshow(img);
    title([image_names{i} ' - 原图']);

    subplot(3, 2, 2);
    imhist(img);
    title([image_names{i} ' - 原图直方图']);

    % 显示灰度线性变换
    subplot(3, 2, 3);
    imshow(gray_img);
    title([image_names{i} ' - 线性变换']);

    subplot(3, 2, 4);
    imhist(gray_img);
    title([image_names{i} ' - 线性变换直方图']);

    % 直方图均衡化
    equalized_img = histeq(img);

    subplot(3, 2, 5);
    imshow(equalized_img);
    title([image_names{i} ' - 直方图均衡化']);

    subplot(3, 2, 6);
    imhist(equalized_img);
    title([image_names{i} ' - 直方图均衡化直方图']);
end
```

四、思考题

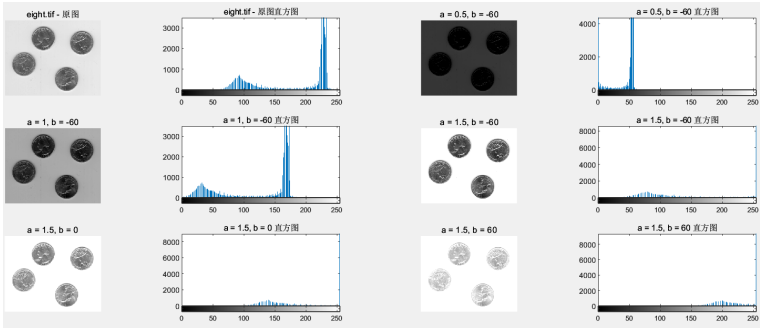
1. 设定不同的斜率值和截距，显示效果会怎样？

答：对实验所给的 huafen.jpg，eight.tif 这两幅图像，作如下设置：

将斜率 a 分别设置为 1，1.5，0.5；截距 b 设置为-60，0，60



huafen.jpg 不同斜率和截距值效果



eight.tif 不同斜率和截距值效果

总结：　当 a > 1 时，图像会变亮，因为灰度值增大；

　　当 a < 1 时，图像会变暗，因为灰度值减小；

　　当 a = 1 时，图像的亮度不变，但是 b 的值会改变图像的对比度，如果 b 增大，图像会变亮；如果 b 减小，图像会变暗。

2. 直方图均衡化是什么意思？它的主要用途是什么？

答：直方图均衡化是一种图像处理技术，主要用于调整图像的对比度。它的原理是将原始图像的灰度直方图（即每个灰度级别的像素数量）重新分布，使得所有灰度级别的像素数量均匀分布，从而增强图像的对比度。

其主要用途有：

（1）增加图像的全局对比度：通过调整图像的直方图，使得图像的灰度级分布更加均匀，从而增加图像的整体对比度，使图像更加清晰可见。

（2）改善视觉效果：通过加大图像的反差，改善图像的视觉效果，使得图像中的细节更加突出，有利于图像的分析和识别。

（3）增强局部对比度：直方图均衡化不仅增加了图像的全局对比度，还能有效地增强局部对比度，尤其是在图像的有用数据对比度接近的情况下。

实验 2　空间域图像增强

一、实验目的：

1. 掌握基本的空域图像增强方法，观察图像增强的效果，加深理解；
2. 了解空域平滑模板的特性及其对不同噪声的影响；
3. 了解空域锐化模板的特性及其对边缘的影响。

二、实验原理：

1. 图像增强是指按特定的需要突出一幅图像中的集些信息，同时消弱或去除某些不需要的信息的处理方法。其主要自的是处理后的图像对某些特定的应用比原来的图像更加有效。

2. 均值（中值）滤波是指在图像上，对得处理的像素给定一个模板，该模板包括了其周围的邻近像索。将模板中的全体像素的均值（中值）来代替原来像素值的方法。

3. 空域锐化是一种图像处理技术，用于提高图像的清晰度和细节。它主要通过增强图像中高频成分来实现，这些高频成分通常与边缘和细节有关。

三、实验内容：

具体实验过程如下：

1. 空域平滑
(1) 读入原图像 lena.bmp 并显示；
(2) 对原图像分别添加高斯噪声和椒盐噪声，并显示加噪图像；
(3) 采用均值滤波进行去噪处理，并显示去噪图像；
(4) 采用中值滤波进行去噪处理，并显示去噪图像。

2. 空域锐化
(1) 读入原图像 lena.bmp 并显示;
(2) 分别采用 Roberts 算子、Prewitt 算子、Sobel 算子计算图像的梯度；
(3) 采用 Laplacian 增强算子对图像进行增强，并显示增强结果；
(4) 比较各种锐化算子对图像边缘的增强效果。

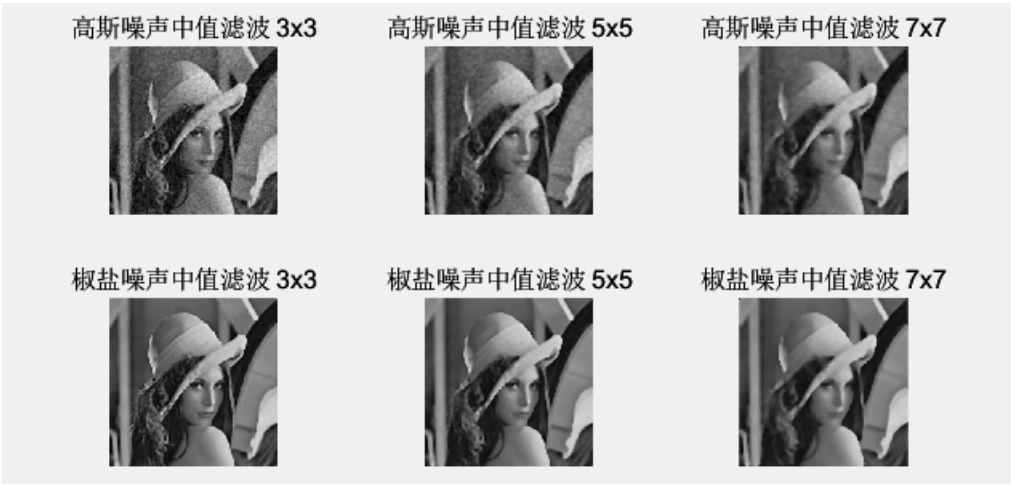
3. 空间域高斯低通滤波
(1) 读入原图像 lena.bmp 并显示；
(2) 对原图像分别添加高斯噪声，并显示加噪图像；
(3) 用空间域高低低通滤波器（5*5 以及 15*15 尺寸）进行去噪处理，并显示去噪图像；

实验的原始图像和结果图像：

1. 空域平滑

均值滤波结果如下：

中值滤波结果如下：



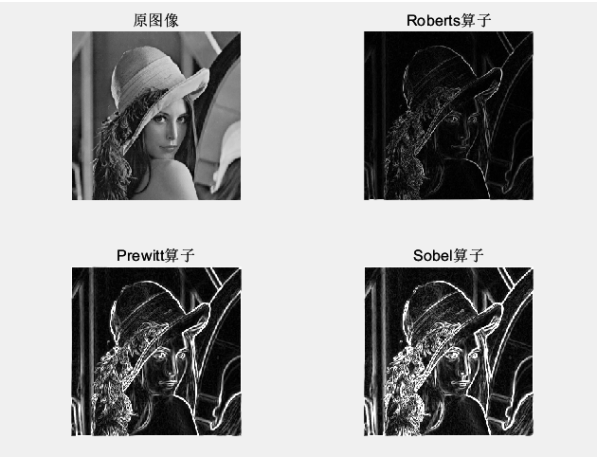
实验结果分析：

（1）由上图可知，中值滤波对椒盐噪声的去除效果非常好，因为它不受极端值的影响。它可以保留图像的边缘和细节；

（2）其次，对于本次试验，均值滤波和中值滤波对高斯噪声效果差不多，但均值滤波对图像细节有一定的损害，因为它假设窗口中的像素都是均匀的，所以滤波效果稍微差一点。

（3）此外，滤波器的窗口大小也是一个重要的参数。较小的窗口适用于去除细小的噪声，但可能会丧失图像细节。较大的窗口可以更好地去除大面积的噪声，但可能会模糊图像。

2. 空域锐化



实验结果分析：

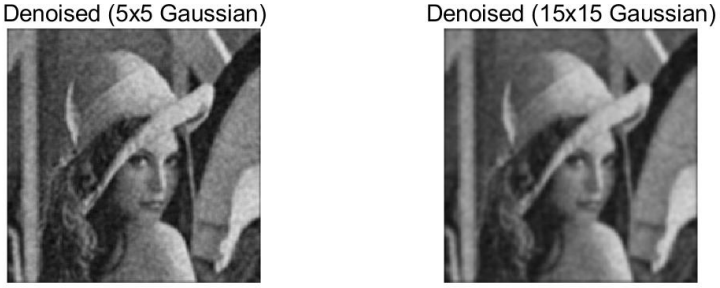
（1）Robert 算子对细节反映较敏感，但提取到的信息量较少，中心位置不明显，图像处理后的边缘不是很平滑。Roberts 算子采用对角线方向相邻两像素之差近似梯度幅值检测边缘。检测水平和垂直边缘的效果好于斜向边缘，定位精度高，对噪声敏感。

（2）Sobel 算子主要用作边缘检测，在技术上，它是一离散性差分算子，用来运算图像亮度函数的灰度之近似值。Sobel 算子引入了平均因素，增强了最近的像素影响，噪声抑制效果比 Prewitt 算子要好。

（3）Prewitt 算子利用像素点上下、左右邻点灰度差，在边缘处达到极值检测边缘。对噪声具有平滑作用，定位精度不够高。Prewitt 算子对噪声相对不敏感，有抑制噪声作用，但操作简便，计算方便快捷。

（4）Laplace 算子是一种各向同性算子，二阶微分算子，在只关心边缘的位置而不考虑其周围的象素灰度差值时比较合适。Laplace 算子对孤立象素的响应要比对边缘或线的响应要更强烈，因此只适用于无噪声图象。

3. 空间域高斯低通滤波



本次实验相关代码如下：

编辑器 - D:\Code\DIP\Test_2.m

Test_2.m

```m
%1
% 读取并显示原图像
img = imread('images\lena.BMP');
figure;
subplot(3, 3, 1);
imshow(img);
title('原图像');

% 添加高斯噪声并显示
img_gaussian = imnoise(img, 'gaussian', 0, 0.01);
subplot(3, 3, 2);
imshow(img_gaussian);
title('高斯噪声图像');

% 添加椒盐噪声并显示
img_salt_pepper = imnoise(img, 'salt & pepper', 0.02);
subplot(3, 3, 3);
imshow(img_salt_pepper);
title('椒盐噪声图像');

% 设置均值滤波模板尺寸
filter_sizes = [3, 5, 7];

% 对高斯噪声图像进行均值滤波并显示
for i = 1:length(filter_sizes)
    filter_size = filter_sizes(i);
    h = fspecial('average', filter_size);
    img_gaussian_filtered = imfilter(img_gaussian, h);
    subplot(3, 3, 3+i);
    imshow(img_gaussian_filtered);
    title(['高斯噪声均值滤波 ', num2str(filter_size), 'x', num2str(filter_size)]);
end

% 对椒盐噪声图像进行均值滤波并显示
for i = 1:length(filter_sizes)
    filter_size = filter_sizes(i);
    h = fspecial('average', filter_size);
    img_salt_pepper_filtered = imfilter(img_salt_pepper, h);
    subplot(3, 3, 6+i);
    imshow(img_salt_pepper_filtered);
    title(['椒盐噪声均值滤波 ', num2str(filter_size), 'x', num2str(filter_size)]);
end

% 中值滤波处理并显示
figure;
subplot(3, 3, 1);
imshow(img);
title('原图像');

% 对高斯噪声图像进行中值滤波并显示
for i = 1:length(filter_sizes)
    filter_size = filter_sizes(i);
    img_gaussian_med_filtered = medfilt2(img_gaussian, [filter_size filter_size]);
    subplot(3, 3, 3+i);
    imshow(img_gaussian_med_filtered);
    title(['高斯噪声中值滤波 ', num2str(filter_size), 'x', num2str(filter_size)]);
end

% 对椒盐噪声图像进行中值滤波并显示
for i = 1:length(filter_sizes)
    filter_size = filter_sizes(i);
    img_salt_pepper_med_filtered = medfilt2(img_salt_pepper, [filter_size filter_size]);
    subplot(3, 3, 6+i);
    imshow(img_salt_pepper_med_filtered);
    title(['椒盐噪声中值滤波 ', num2str(filter_size), 'x', num2str(filter_size)]);
end
```

```m
%2
% 读取并显示原图像
img = imread('images\lena.BMP');
img_gray = im2gray(img); % 直接使用 im2gray 转换为灰度图像
figure;
subplot(3, 3, 1);
imshow(img_gray);
title('原图像');
```

```
% Roberts算子
roberts_x = [1 0; 0 -1];
roberts_y = [0 1; -1 0];
img_roberts_x = imfilter(double(img_gray), roberts_x);
img_roberts_y = imfilter(double(img_gray), roberts_y);
img_roberts = sqrt(img_roberts_x.^2 + img_roberts_y.^2);
subplot(3, 3, 2);
imshow(uint8(img_roberts));
title('Roberts算子');

% Prewitt算子
prewitt_x = [-1 0 1; -1 0 1; -1 0 1];
prewitt_y = [-1 -1 -1; 0 0 0; 1 1 1];
img_prewitt_x = imfilter(double(img_gray), prewitt_x);
img_prewitt_y = imfilter(double(img_gray), prewitt_y);
img_prewitt = sqrt(img_prewitt_x.^2 + img_prewitt_y.^2);
subplot(3, 3, 3);
imshow(uint8(img_prewitt));
title('Prewitt算子');

% Sobel算子
sobel_x = [-1 0 1; -2 0 2; -1 0 1];
sobel_y = [-1 -2 -1; 0 0 0; 1 2 1];
img_sobel_x = imfilter(double(img_gray), sobel_x);
img_sobel_y = imfilter(double(img_gray), sobel_y);
img_sobel = sqrt(img_sobel_x.^2 + img_sobel_y.^2);
subplot(3, 3, 4);
imshow(uint8(img_sobel));
title('Sobel算子');

% Laplacian算子
laplacian_filter = [0 -1 0; -1 4 -1; 0 -1 0];
img_laplacian = imfilter(double(img_gray), laplacian_filter, 'replicate');
subplot(3, 3, 5);
imshow(uint8(img_laplacian));
title('Laplacian算子');

% Laplacian增强
alpha = 0.5;
img_enhanced = double(img_gray) - alpha * img_laplacian;
subplot(3, 3, 6);
imshow(uint8(img_enhanced));
title('Laplacian增强');

% Laplacian增强算子
laplacian_enhanced_filter = [1 -2 1; -2 4 -2; 1 -2 1];
img_laplacian_enhanced = imfilter(double(img_gray), laplacian_enhanced_filter, 'replicate');
subplot(3, 3, 7);
imshow(uint8(img_laplacian_enhanced));
title('Laplacian增强算子');
```

```
%3
% 读取原图像
img = imread('images\lena.BMP');

% 显示原图像
figure;
subplot(2, 2, 1);
imshow(img);
title('原图像');

% 添加高斯噪声
noisy_img = imnoise(img, 'gaussian', 0, 0.01);

% 显示加噪图像
subplot(2, 2, 2);
imshow(noisy_img);
title('加噪图像');

% 空间域高斯低通滤波器尺寸
filter_size_1 = 5;
filter_size_2 = 15;

% 应用高斯低通滤波器进行去噪处理
denoised_img_1 = imgaussfilt(noisy_img, filter_size_1);
denoised_img_2 = imgaussfilt(noisy_img, filter_size_2);

% 显示去噪图像
subplot(2, 2, 3);
imshow(denoised_img_1);
title('5x5高斯低通滤波去噪图像');

% 显示去噪图像
subplot(2, 2, 4);
imshow(denoised_img_2);
title('15x15高斯低通滤波去噪图像');
```

实验 3　图像的傅立叶变换

一、实验目的：

1. 解图像变换的意义和手段；
2. 熟悉傅立叶变换的基本性质；
3. 熟练掌握 FFT 变换方法及应用：
4. 通过实验了解二维频谱的分布特点；
5. 通过本实验掌握利用 MATLAB 编程实现数字图像的傅立叶变换;
6. 评价人眼对图像幅频特性和相频特性的敏感度。

二、实验原理：

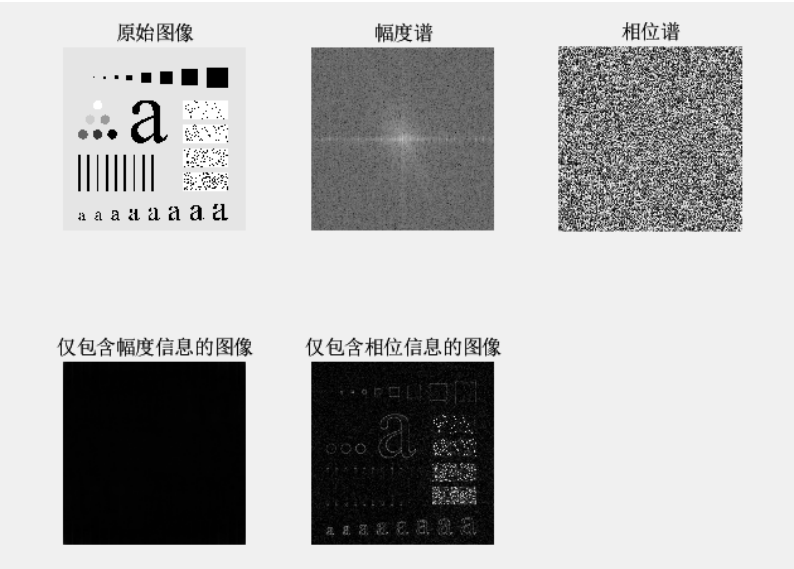
傅里叶变换是线性系统分析的一个有利工具，它能够定量地分析诸如数字化系统、采样点、电子放大器、卷积滤波器、噪音和显示点等的作用。

三、实验内容：

1. 具体实验过程：

启动 MATLAB 程序，对图像文件分别进行二维 Fourier 变换，将其幅度谱进行搬移，在图像中心显示，并评价人眼对图像幅频特性和相频特性的敏感度。

2. 实验的原始图像和结果图像：



实验结果分析：

观察上图，可知幅频特性主要涵盖了图像颜色的分布，相频特性主要刻画了图像的边界信息。人眼对图像的相频特性更加敏感，看相频特性能够大概地知道图像内容。

四、思考题

1. 傅里叶变换有哪些重要的性质？

答：重要的性质主要有：线性性质（Linearity）、平移性质（Shift）、对称性质（Symmetry），以及卷积性质（Convolution）等

2. 图像的二维频谱在显示和处理时应注意什么？

答：（1）进行傅里叶变换的图像应该是灰度图形，原 RGB 彩色图像无法进行相应变换；

（2）注意使用 fftshift 函数将频谱的零频分量移至频谱的中心。此时，频谱的中心对应于低频成分，而边缘对应于高频成分。图像的大部分能量通常集中在低频区域，这是因为图像的大部分信息（如亮度、颜色等）都在低频区域；

（3）频谱的可视化：由于频谱的幅度范围可能非常大，直接显示可能会丢失很多细节。因此，通常会对频谱进行对数变换以改善可视化效果。

本次实验相关代码：

编辑器 - D:\Code\DIP\Test_3.m

Test_3.m

```
% 读取图像
img = imread('images\test.tif');

% 进行二维Fourier变换
fft_img = fft2(double(img));
fft_img_shifted = fftshift(fft_img);

% 计算幅度谱
magnitude_spectrum = abs(fft_img_shifted);

% 显示幅度谱
figure;
subplot(2, 3, 1);
imshow(img, []);
title('原始图像');

subplot(2, 3, 2);
imshow(log(1 + magnitude_spectrum), []);
title('幅度谱');

% 计算相位谱
phase_spectrum = angle(fft_img_shifted);

% 显示相位谱
subplot(2, 3, 3);
imshow(phase_spectrum, []);
title('相位谱');

% 重构仅包含幅度信息的图像
magnitude_only = ifft2(ifftshift(magnitude_spectrum));
magnitude_only_img = abs(magnitude_only);

subplot(2, 3, 4);
imshow(magnitude_only_img, []);

title('仅包含幅度信息的图像');

% 重构仅包含相位信息的图像
phase_only = ifft2(ifftshift(exp(1i * phase_spectrum)));
phase_only_img = abs(phase_only);

subplot(2, 3, 5);
imshow(phase_only_img, []);
title('仅包含相位信息的图像');

% 对相位更敏感
```

实验 4　图像增强——频域滤波

一、实验目的：

1. 掌握怎样利用傅立叶变换进行频域滤波；
2. 掌握频域滤波的概念及方法；
3. 熟练掌握频域空间的各类滤波器：
4. 利用 MATLAB 程序进行频域滤波。

二、实验原理：

频域滤波分为低通滤波和高通滤波两类，对应的滤波器分别为低通滤波器和高通滤波器。频域低通过滤的基本思想：

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

$F(u,v)$是需要钝化图像的傅立叶变换形式，$H(u,v)$是选取的一个低通过滤器变换函数，$G(u,v)$是通过$H(u,v)$减少$F(u,v)$的高频部分来得到的结果，运用傅立叶逆变换得到钝化后的图像。

理想地通滤波器(ILPF)具有传递函数：

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

其中，D_0为指定的非负数，$D(u,v)$为(u,v)到滤波器的中心的距离。$D(u,v) = D_0$的点的轨迹为一个圆。

n阶巴特沃兹低通滤波器(BLPF)(在距离原点D_0处出现截至频率)的传递函数为:

$$H(u,v) = \frac{1}{1+[\frac{D(u,v)}{D_0}]^{2n}}$$

与理想地通滤波器不同的是，巴特沃兹低通滤波器的传递函数并不是在D_0处突然不连续。

高斯低通滤波器(GLPF)的传递函数为：

$$H(u,v) = \exp\left(D^2(u,v)/2\sigma^2\right)$$

其中，σ^2为标准差。相应的高通滤波器也包括：理想高通滤波器、n阶巴特沃兹高通滤波器、高斯高通滤波器。通过使用如下的简单关系，可以获得相应高通滤波器的传递函数：$H_{hp} = 1 - H_{lp}(u,v)$

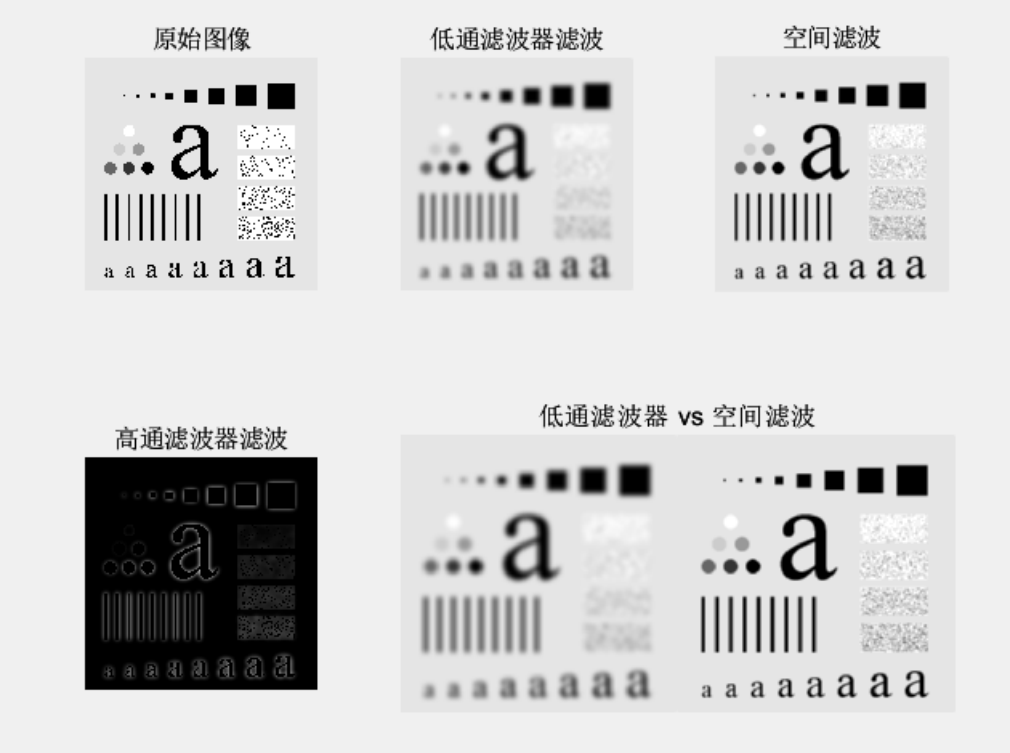
三、实验内容：

1. 具体实验过程：

利用 MATLAB 提供的低通滤波器实现图像信号的滤波运算，并与空间滤波进行比较；

利用 MATLAB 提供的高通滤波器对图像进行处理。

2. 实验的原始图像和结果图像：



结果分析：

可以看到，自己写的空间滤波程序与 matlab 自带函数处理效果有些许差别，原因在于滤波器设计时模板数值不同。显然，我们自己根据实际情况设计的空间滤波器的滤波效果均要优于系统自带的函数。

四、思考题

1. 结合实验，评价频域滤波有哪些优点？

答：(1) 可以利用频率成分和图像外表之间的对应关系。一些在空间域表述困难的增强任务，在频率域中变得非常普通。

(2) 滤波在频率域更为直观，它可以解释空间域滤波的某些性质。

(3) 给出一个问题，寻找某个滤波器解决该问题，频率域处理对于试验、迅速而全面地控制滤波器参数是一个理想工具。

(4) 一旦找到一个特殊应用的滤波器，通常在空间域采用硬件实现它。

(5) 数图像滤波技术要求求解复杂的微分方程，利用图像变换可以将这些微分方程转换成频域中的代数方程，大大简化了数学分析和求解。

(6) 在频域的增强是通过改变图像中不同频率分量来实现的。图像频谱给出图像全局的性质，所以频域增强不是对逐个像素进行的，从这点来讲它不像空域增强那么直接，但用频率分量来分析增强的原理却比较直观。

2. 在频域滤波过程中需要注意哪些事项？

答：(1) 在执行操作时，应确保灰度值介于 0 到 255 的整数范围内，因为原始图像灰度矩阵提供的往往是浮点数，所以需要进行数据类型的转换；

(2) 需注意矩阵的维度必须一致，否则无法进行操作；

(3) 频域滤波与理想滤波器之间存在差异，快速傅里叶变换（FFT）在处理时不可避免地会有频谱泄漏的现象。

本次实验相关代码：

编辑器 - D:\Code\DIP\Test_4.m

Test_4.m

```
% 读取图像
img = imread('images\test.tif');

% 显示原始图像
figure;
subplot(2, 3, 1);
imshow(img);
title('原始图像');

% 低通滤波
% 使用MATLAB提供的高斯低通滤波器
filtered_img_lowpass = imgaussfilt(img, 5); % 标准差为5的高斯滤波器
subplot(2, 3, 2);
imshow(filtered_img_lowpass);
title('低通滤波器滤波');

% 空间滤波
% 使用自定义的低通滤波器（例如均值滤波器）
h = fspecial('average', [5 5]); % 创建5x5的均值滤波器
filtered_img_spatial = imfilter(img, h, 'replicate');
subplot(2, 3, 3);
imshow(filtered_img_spatial);
title('空间滤波');

% 高通滤波
% 使用MATLAB提供的拉普拉斯滤波器
filtered_img_highpass = img - imgaussfilt(img, 5); % 高斯低通滤波后，原图像减去低通滤波后的图像
subplot(2, 3, 4);
imshow(filtered_img_highpass);
title('高通滤波器滤波');

% 显示滤波后的图像并进行比较
subplot(2, 3, [5, 6]);
imshowpair(filtered_img_lowpass, filtered_img_spatial, 'montage');
title('低通滤波器 vs 空间滤波');
```

实验 5　数字图像的几何变换

一、实验目的：

1. 熟悉 MATLAB 的操作和基本功能；

2. 理解和掌握图像平移、垂直镜像变换、水平镜像变换、缩放和旋转的原理和应用。

二、实验原理：

1. 初始坐标为$(x,\ y)$的点经过平移(x_0,y_0)，坐标变为(x',y')，两点之间的关系为：$\begin{cases}x'=x+x_0\\y'=y+y_0\end{cases}$，以矩阵形式表示为：

$$\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}1&0&x_0\\0&1&y_0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}$$

2. 图像的镜像变换是以图象垂直中轴线或水平中轴线交换图像的变换，分为垂直镜像变换和水平镜像变换，两者的矩阵形式分别为：

$$\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}-1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix},\ \begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}$$

3. 图像缩小和放大变换矩阵相同：

$$\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}S_x&0&0\\0&S_y&0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}$$

当$S_x\le 1,S_y\le 1$时，图像缩小；$S_x\ge 1,S_y\ge 1$时，图像放大。

4. 图像旋转定义为以图像中某一点为原点以逆时针或顺时针方向旋转一定角度。其变换矩阵为：

$$\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}cos\theta&sin\theta&0\\-sin\theta&cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}$$

该变换矩阵是绕坐标轴原点进行的，如果是绕一个指定点(a,b)旋转，则现要将坐标系平移到该点，进行旋转，然后再平移回到新的坐标原点。

三、实验内容：

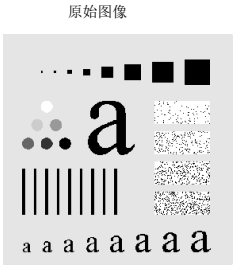
1. 具体实验过程：

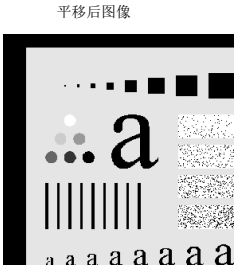
（1）启动 MATLAB 程序，导入图像文件；

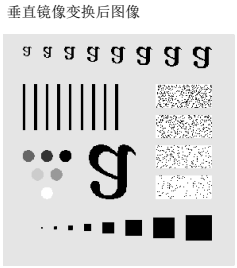
（2）对图像文件分别进行平移、垂直镜像变换、水平镜像变换、缩放和旋转操作，与实验箱运行结果进行比对；

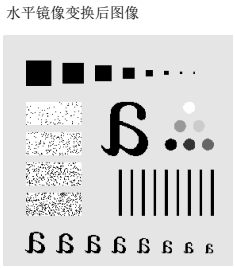
（3）记录和整理实验报告。

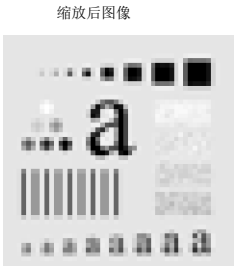
2. 实验的原始图像和结果图像：

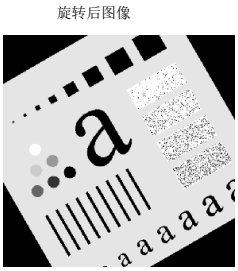








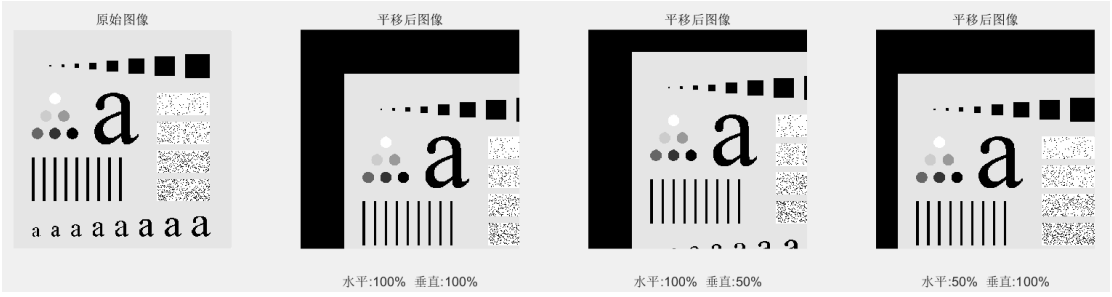




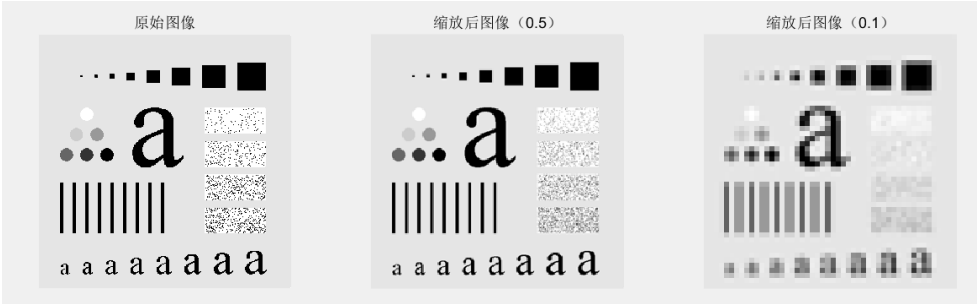
四、思考题

1. 改变水平和垂直的偏移量，观察显示？

答：如下图所示：



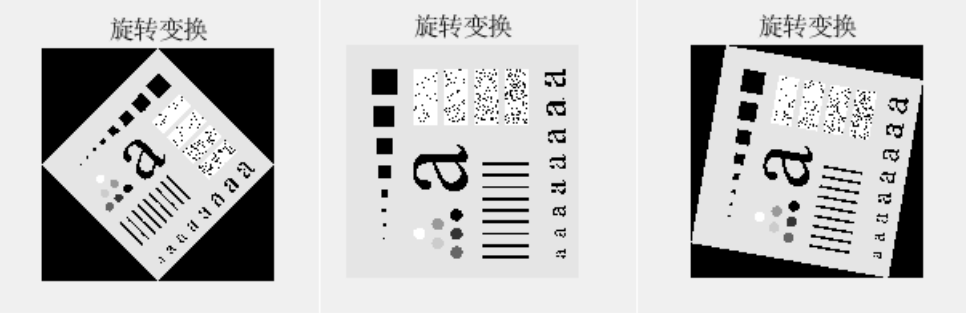
2. 改变缩放比例，看看效果如何？



答：改变缩放比例后，效果图片会越来越缩小。由于上图图片显示尺寸一致，故进行缩放过后，图片会越来越模糊。

3. 改变旋转角度，显示效果会怎么样？

答：如下图所示：



本次实验相关代码如下：

编辑器 - D:\Code\DIP\Test2_1.m

Test2_1.m

```
% 读取图像
img = imread('images\test.tif');
% 显示原始图像
figure;
subplot(2, 3, 1);
imshow(img);
title('原始图像');

% 平移操作
tx = 50; % 水平方向平移量
ty = 30; % 垂直方向平移量
translated_img = imtranslate(img, [tx, ty]);
subplot(2, 3, 2);
imshow(translated_img);
title('平移后图像');

% 垂直镜像变换
flipped_img_vertical = flipud(img);
subplot(2, 3, 3);
imshow(flipped_img_vertical);
title('垂直镜像变换后图像');

% 水平镜像变换
flipped_img_horizontal = fliplr(img);

% 显示水平镜像变换后的图像
subplot(2, 3, 4);
imshow(flipped_img_horizontal);
title('水平镜像变换后图像');

% 缩放操作
scale_factor = 0.5; % 缩放因子
scaled_img = imresize(img, scale_factor);

% 显示缩放后的图像
subplot(2, 3, 5);
imshow(scaled_img);
title('缩放后图像');

% 旋转操作
angle_degrees = 30; % 旋转角度（度）
rotated_img = imrotate(img, angle_degrees, 'bilinear', 'crop');

% 显示旋转后的图像
subplot(2, 3, 6);
imshow(rotated_img);
title('旋转后图像');
```

实验 6　形态学图像处理

一、实验目的：

1. 了解并掌握数学形态学运算基本方法（腐蚀、膨胀、开、闭运算）；
2. 熟练掌握二值图像中区域填充、边界提取等形态学的应用。

二、实验原理：

1. 形态学运算是数学形态学（Mathermatical Morphogy）集合论方法发展起来的图像处理方法，其基本思想是：用具有一定形态的结构元素去量度和提取图像中的对应形状，以达到图像分析和识别的目的。常见的形态学运算是：腐蚀（Erosion）和膨胀（Dilation）两种。

2.所需函数：

（1） strel 函数可以为各种常见形态学运算生成结构元素 se。

（2） 腐蚀：一种消除边界点，使边界向内部收缩的过程。

调用函数：I2 = imerode(I,se)

参数说明：

I 是原始图像，可以是二值或灰度图像（对应于灰度腐蚀）；

se 是由 strel 函数返回的自定义或预设的结构元素对象。

I2 为腐蚀后的输出图像。

（3） 膨胀：将与物体接触的所有背景点合并到该物体中，使边界向外部扩张的过程。

调用函数：I2 = imdilate(I,se)

参数说明：与腐蚀的参数说明类似。

（4）开运算和闭运算都是由腐蚀和膨胀复合而成，开运算是先腐蚀后膨胀，而闭运算是先膨胀后腐蚀。

开运算：调用函数：I2 = imopen(I,se)

闭运算：调用函数：I2 = imclose(I,se)

区域填充：是在边界已知的情况下得到边界包围的整个区域的形态学技术。调用函数：BW2 = imfill(BW)，该函数用于填充图像区域或“空洞”。

三、实验内容：

1.具体实验过程如下：

（1）Matlab 读入实验图像；

（2）分别设置膨胀、腐蚀的结构元素；

（3）分别利用结构元素对二值图 text.png 进行膨胀、腐蚀操作；

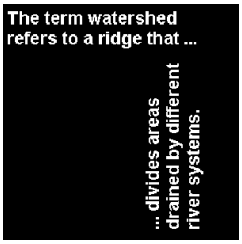
（4）对二值图 circles.png 分别进行开运算、闭运算操作；

（5）对二值图 coins.png 进行孔洞填充操作。

2.实验的原始图像和结果图像

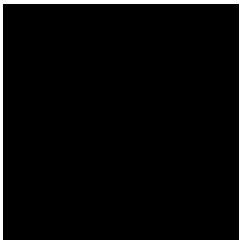
（1）对 text.png 进行膨胀、腐蚀：

原图像 text.png



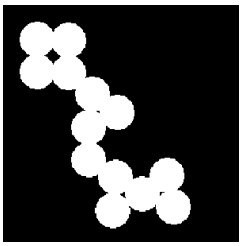
膨胀 text.png

腐蚀 text.png

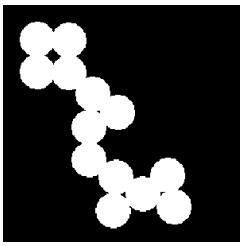


（2）对 circles.png 进行开、闭运算：

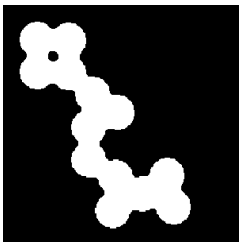
原始 circles.png



开运算 circles.png

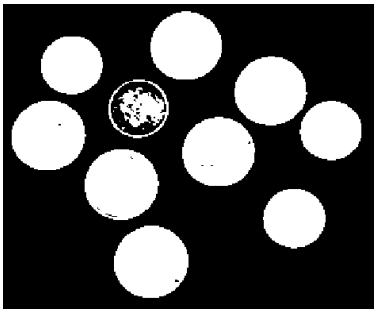


闭运算 circles.png

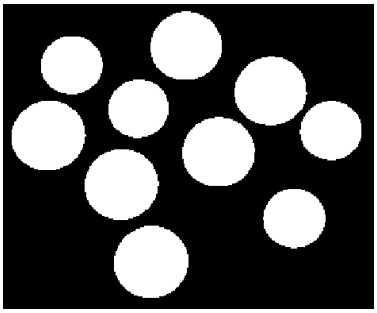


（3）对 coins.png 进行孔洞填充：

原始 coins.png



孔洞填充 coins.png



四、思考题

1. 结构元素的大小和形状对于形态学操作结果有何影响？

答：形态学结构元素的大小和形状决定了它在图像上的操作效果。
（1）大小影响：结构元素越大，其覆盖的像素范围越广，导致形态学操作（如腐蚀和膨胀）的效果更显著。大的结构元素可以更强烈地改变图像的边缘和形状。
（2）形状影响：结构元素的形状决定了操作的方向性和特征。不同形状的结构元素可以针对图像的特定方向进行操作，如水平线段主要影响水平方向，而圆形结构元素则提供无方向性的处理。

2. 区域填充和闭运算之间有何区别？

答：（1）区域填充：区域填充是一种形态学操作，它的目的是填充图像中的“空洞”或闭合区域。这种操作通常用于去除图像中的小孔或噪声。

（2）闭运算是形态学中的一种操作，它首先对图像进行膨胀操作，然后进行腐蚀操作。闭运算的目的是通过填充目标区域内部的细小空洞，将断开的邻近目标连接起来，同时在不明显改变物体面积和形状的情况下平滑其边界闭运算。但是，与区域填充不同，闭运算不仅会填充孔洞，还会改变图像的其他部分（例如，通过膨胀操作增加物体的大小）。

本次实验相关代码如下：

编辑器 - D:\Code\DIP\Test2_2.m

Test2_2.m

```
% 读取实验图像
textImage = imread('images/text.png');
circlesImage = imread('images/circles.png');
coinsImage = imread('images/coins.png');

% 设置膨胀、腐蚀的结构元素
se = strel('disk', 5); % 圆盘形结构元素，半径为5

% 对二值图text.png进行膨胀操作
textDilated = imdilate(textImage, se);

% 对二值图text.png进行腐蚀操作
textEroded = imerode(textImage, se);

% 对二值图circles.png进行开运算
circlesOpened = imopen(circlesImage, se);

% 对二值图circles.png进行闭运算
circlesClosed = imclose(circlesImage, se);

% 对二值图coins.png进行孔洞填充操作
coinsFilled = imfill(coinsImage, 'holes');

% 显示结果
figure;
subplot(1, 3, 1);
imshow(textImage);
title('原图像text.png');

subplot(1, 3, 2);
imshow(textDilated);
title('膨胀text.png');

subplot(1, 3, 3);
imshow(textEroded);
title('腐蚀text.png');

figure;
subplot(1, 3, 1);
imshow(circlesImage);
title('原始circles.png');

subplot(1, 3, 2);
imshow(circlesOpened);
title('开运算circles.png');

subplot(1, 3, 3);
imshow(circlesClosed);
title('闭运算circles.png');

figure;
subplot(1, 2, 1);
imshow(coinsImage);
title('原始coins.png.png');

subplot(1, 2, 2);
imshow(coinsFilled);
title('孔洞填充coins.png');

```

实验 7　图像分割

一、实验目的：

1. 了解图像分割的基本理论和算法；
2. 掌握用 MATLAB 语言进行图像边缘提取的方法；
3. 熟练掌握基于阈值的图像分割方法。

二、实验原理：

1. 图象分割是按照某些特性(如灰度级,频谱,颜色，纹理等)将图象划分成一些区域，在这些区域内其特性是相同的或者说是均匀的，两个相邻区域彼此特性则是不同的，其间存在着边缘或边界， 其中有基于形态学、区域、边界和阀值的分割方法。

2. 边缘检测可以大幅度地减少数据量，并且除去那些被认为不相关的信息，保留图像重要的结果属性。

（1）常用的边缘检测算子：Canny、LoG 等算子。

调用函数：BW = edge(I,method,threshold,sigma)

（2）阈值分割方法：是确定一个阈值，然后把每个像素点的灰度值和阈值相比较，根据比较的结果把该像素划分为两类——前景和背景，前景就是要寻找的目标。一般分为 3 步：I.确定阈值；II.将像素和阈值比较；III.把像素归类。

三、实验内容：

1. 具体实验过程：

（1）读入图像 circuit.tif，将其转换成灰度图，对该灰度图进行 Canny、LoG 等算子的边缘检测，观察检测结果图；

（2）使用全局阈值分割对灰度图像 coins_gray.png 进行灰度分割处理，显示分割结果图；

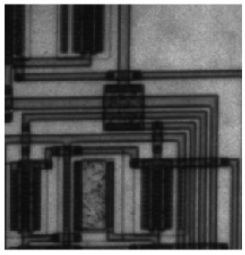
（3)使用自适应阈值分割对灰度图像 printedtext.png 进行灰度分割处理，显示分割结果图;并调节'ForegroundPolarity','Sensitivity'这两个参数使得前景比背景暗，并清晰显示文字；

（4）记录和整理实验报告。

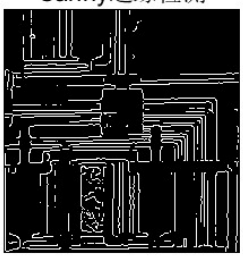
2. 实验的原始图像和结果图像

（1）边缘检测结果：

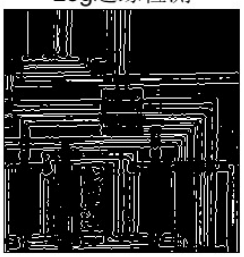
原Circuit.tif



Canny边缘检测



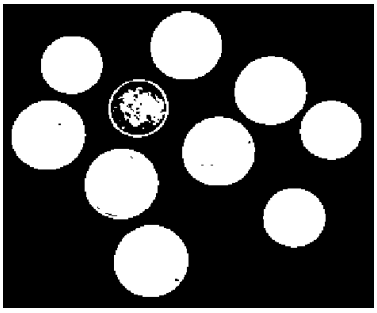
Log边缘检测



（2）全局阈值分割结果：

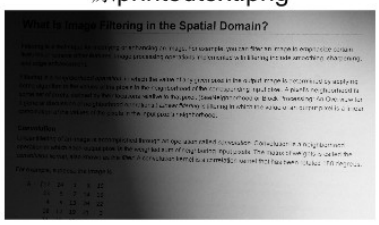
原 coins-gray.png

全局阈值分割

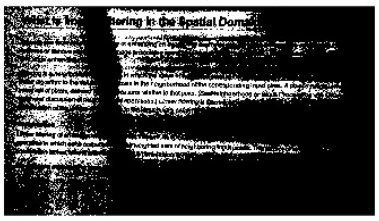


(3)自适应阈值分割结果：

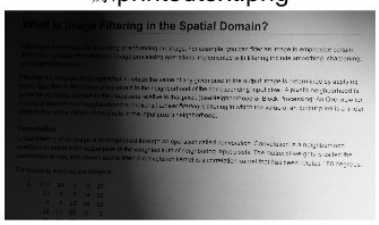
原printedtext.png



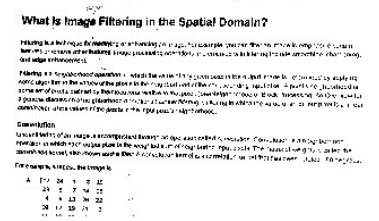
自适应阈值分割



原printedtext.png



调参自适应阈值分割



四、思考题

1. 实验中所使用的两种算子所得到的边界有什么异同？

答：（1） Log 算子首先通过高斯函数对图像进行平滑处理，因此对噪声的抑制作用比较明显，但同时也可能将原有的边缘也平滑了，造成某些边缘无法检测到。

（2） Canny 算子也采用高斯函数对图像进行平滑处理，因此具有较强的去噪能力，但同样存在容易平滑掉一些边缘信息，其后所采用的一阶微分算子的方向性较 Log 算子要好，因此边缘定位精度较高。该算子与其它边缘检测算子的不同之处在于，它使用 2 种不同的阈值分别检测强边缘和弱边缘，并且仅当弱边缘相连时才将弱边缘包含在输出图像中，因此这种方法较其它方法而言不容易被噪声“填充”更容易检测出真正的弱边缘。

2. 为什么对于图像 printedtext.png 采用全局阈值分割效果不好？

答：因为全局阈值分割技术基于一个前提：即图像中的前景与背景在灰度上有明显的差异。

但是，这种方法在处理像 printedtext.png 这样复杂的图像时可能不太适用。因为复杂图像可能存在照明不均或前景与背景灰度相近的问题，而这会导致全局阈值分割失效。与此相对地，自适应阈值分割技术可以根据图像的局部特征而来动态调整分割阈值，因此往往能够取得更佳的分割效果。

而本实验中，我们正是使用自适应阈值分割法对灰度图像 printedtext.png 进行灰度分割处理，通过调整相关参数，从而清晰显示出了文字。

本次实验相关代码如下：

编辑器 - D:\Code\DIP\Test2_3.m

Test2_3.m

```m
% 读取图像 circuit.tif 并转换成灰度图
grayCircuitImage = imread('images/circuit.tif');

% 对灰度图进行 Canny 和 LoG 边缘检测
cannyEdges = edge(grayCircuitImage, 'Canny');
logEdges = edge(grayCircuitImage, 'log');

% 显示边缘检测结果
figure;
subplot(1, 3, 1);
imshow(grayCircuitImage);
title('原Circuit.tif');

subplot(1, 3, 2);
imshow(cannyEdges);
title('Canny边缘检测');

subplot(1, 3, 3);
imshow(logEdges);
title('Log边缘检测');

% 读取灰度图像 coins_gray.png
coinsGrayImage = imread('images/coins_gray.png');

% 使用全局阈值分割处理
globalThreshold = graythresh(coinsGrayImage);
coinsBinaryImage = imbinarize(coinsGrayImage, globalThreshold);

% 显示分割结果图
figure;
subplot(1, 2, 1);
imshow(coinsGrayImage);
title('原coins_gray.png');

subplot(1, 2, 2);
imshow(coinsBinaryImage);
title('全局阈值分割');

% 读取灰度图像 printedtext.png
printedTextImage = imread('images/printedtext.png');

% 使用自适应阈值分割
adaptiveThresh = adaptthresh(printedTextImage, 0.5); % 这里的敏感度可以调整
adaptiveBinaryImage = imbinarize(printedTextImage, adaptiveThresh);

% 显示分割结果图
figure;
subplot(1, 2, 1);
imshow(printedTextImage);
title('原printedtext.png');

subplot(1, 2, 2);
imshow(adaptiveBinaryImage);
title('自适应阈值分割');

% 调整参数使得前景比背景暗，并清晰显示文字
adaptiveBinaryImageAdjusted = imbinarize(printedTextImage, 'adaptive', ...
    'ForegroundPolarity', 'dark', 'Sensitivity', 0.4);

% 显示调整参数后的分割结果图
figure;
subplot(1, 2, 1);
imshow(printedTextImage);
title('原printedtext.png');

subplot(1, 2, 2);
imshow(adaptiveBinaryImageAdjusted);
title('调参自适应阈值分割');

```