# 实验四

## 实验目的

1. 掌握中值滤波方法，掌握图像锐化方法，比较各个梯度算子锐化效果。掌握频域滤波方法，观察低通滤波和高通滤波。

## 实验原理

中值滤波是一种非线性平滑滤波，它是用一个有奇数点的滑动窗口，将窗口中心点的值用窗口各点的中值代替。

图像的锐化是使边缘和轮廓线模糊的图像变得清晰，使其细节更加清晰。从数学上看，图像模糊的实质是图像受到平均或者积分运算的影响，因此对其进行逆运算（如微分运算）就可以使图像清晰。

在频域上卷积被表示为乘积，因此在频域上对图像进行滤波就变得更加直观。在频域上进行滤波的步骤：

1. 计算需增强的图像的傅里叶变化；
2. 将其与一个传递函数相乘；
3. 再将结果进行傅里叶逆变化得到增强的图像。

## 实验内容

### 中值滤波

**步骤**

1. 导入图像，并对图像添加高斯、椒盐、泊松噪声；
2. 设定中值滤波模板的大小
3. 将中值模板在图像上移动改变图像上的灰度值。

**源代码**

clc

% 导入图像

img = imread('lena.png');

img = rgb2gray(img);

[hight, width] = size(img);

% 对图像分别加入高斯、椒盐、泊松噪声

GausNoi = imnoise(img, 'gaussian', 1, 0.5);

SaltPep = imnoise(img, 'salt & pepper', 0.02);

PoisNoi = imnoise(img, 'poisson');

% 设定中值滤波模板大小

fsize = 3;

pad = (fsize - 1) / 2;

%对加入高斯噪声的图像过滤噪声

GausNew = GausNoi;

for i = 1+pad : hight-pad

for j = 1+pad : width-pad

filter = GausNew(i-pad:i+pad, j-pad:j+pad);

fil\_arr = sort(reshape(filter, [1,fsize\*fsize]));

GausNew(i, j) = fil\_arr((fsize\*fsize-1)/2);

end

end

% 对加入椒盐噪声的图像过滤噪声

PeppNew = SaltPep;

for i = 1+pad : hight-pad

for j = 1+pad : width-pad

filter = PeppNew(i-pad:i+pad, j-pad:j+pad);

fil\_arr = sort(reshape(filter, [1,fsize\*fsize]));

PeppNew(i, j) = fil\_arr((fsize\*fsize-1)/2);

end

end

% 泊松噪声

PoisNew = PoisNoi;

for i = 1+pad : hight-pad

for j = 1+pad : width-pad

filter = PoisNew(i-pad:i+pad, j-pad:j+pad);

fil\_arr = sort(reshape(filter, [1,fsize\*fsize]));

PoisNew(i, j) = fil\_arr((fsize\*fsize-1)/2);

end

end

% plot image

subplot(2,3,1);

imshow(GausNoi);title('高斯噪声');

subplot(2,3,2);

imshow(SaltPep);title('椒盐噪声');

subplot(2,3,3);

imshow(PoisNoi);title('泊松噪声');

subplot(2,3,4);

imshow(GausNew);title('高斯噪声（滤波后）');

subplot(2,3,5);

imshow(PeppNew);title('椒盐噪声（滤波后）');

subplot(2,3,6);

imshow(PoisNew);title('泊松噪声（滤波后）');

**实验结果**

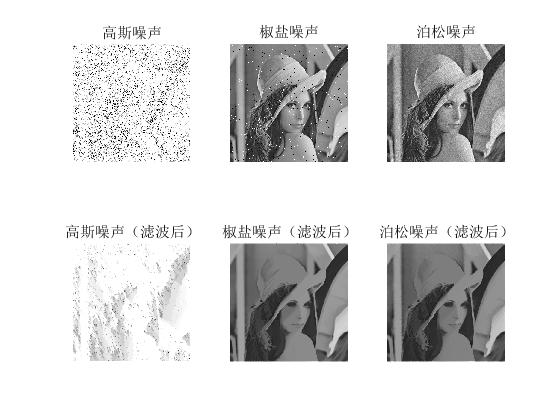


图 1 图像的中值滤波

### 梯度算子

**步骤**

1. 导入图像，得到图像的大小；
2. 构造Sober算子和Prewitt算子的模板
3. 分别利用Sober算子和Prewitt算子两种模板再图像中运行，改变图像的灰度值。

**Sober算子滤波器源代码**

clc

img = imread('lena.png'); % load original image

img = rgb2gray(img);

subplot(2,2,1)

imshow(img);title('Ô­Í¼');

[hight, width] = size(img);

k\_shape = [3, 3]; % set kernel size

padding = (k\_shape(1)-1) / 2; % padding size

% generate sobel kernel

h\_kernel = [1,2,1;0,0,0;-1,-2,-1]; % horizon sobel kernel

v\_kernel = [-1,0,1;-2,0,2;-1,0,1]; % vertical sobel kernel

h\_img = zeros(hight + padding\*2, width + padding\*2);

% generate new image

h\_img(1+padding:hight+padding, 1+padding:width+padding) = double(img);

new\_h = zeros(hight + padding\*2, width + padding\*2);

for i = (1+padding) : (hight+padding)

for j = (1+padding) : (width+padding)

% execute horizon filter

new\_h(i, j) = sum(sum(h\_img(i-padding : i+padding, j-padding : j+padding) .\* h\_kernel));

end

end

new\_h = uint8(new\_h(1+padding:hight+padding, 1+padding:width+padding));

subplot(2,2,2);

imshow(new\_h);title('水平sobel');

v\_img = zeros(hight + padding\*2, width + padding\*2);

% generate new image

v\_img(1+padding:hight+padding, 1+padding:width+padding) = double(img);

new\_v = zeros(hight + padding\*2, width + padding\*2);

for i = (1+padding) : (hight+padding)

for j = (1+padding) : (width+padding)

% execute vertical filter

new\_v(i, j) = sum(sum(v\_img(i-padding : i+padding, j-padding : j+padding) .\* v\_kernel));

end

end

new\_v = uint8(new\_v(1+padding:hight+padding, 1+padding:width+padding));

subplot(2,2,3);

imshow(new\_v);title('垂直sober');

norm\_img = zeros(hight + padding\*2, width + padding\*2);

% generate new image

norm\_img(1+padding:hight+padding, 1+padding:width+padding) = double(img);

new\_norm = zeros(hight + padding\*2, width + padding\*2);

for i = (1+padding) : (hight+padding)

for j = (1+padding) : (width+padding)

% execute grad\_norm filter

h\_norm = sum(sum(norm\_img(i-padding : i+padding, j-padding : j+padding) .\* h\_kernel))^2;

v\_norm = sum(sum(norm\_img(i-padding : i+padding, j-padding : j+padding) .\* v\_kernel))^2;

new\_norm(i, j) = sqrt(h\_norm + v\_norm);

end

end

new\_norm = uint8(new\_norm(1+padding:hight+padding, 1+padding:width+padding));

subplot(2,2,4);

imshow(new\_norm);title('sober模值');

**Sober实验结果**

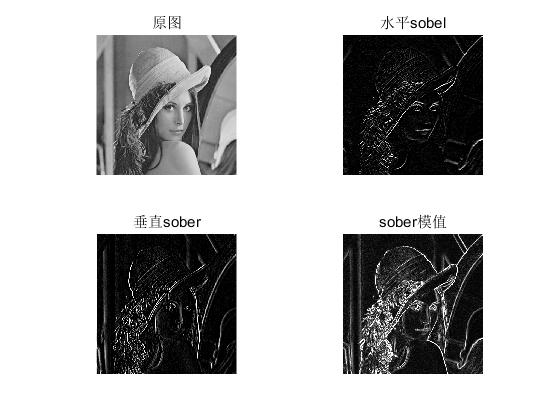


图 2 Sober实验结果图

**Prewitt滤波器源代码**

clc

img = imread('lena.png'); % load original image

img = rgb2gray(img);

figure()

imshow(img);

[hight, width] = size(img);

k\_shape = [3, 3]; % set kernel size

padding = (k\_shape(1)-1) / 2; % padding size

% generate prewitt kernel

h\_kernel = [1,1,1;0,0,0;-1,-1,-1]; % horizon prewitt kernel

v\_kernel = [-1,0,1;-1,0,1;-1,0,1]; % vertical prewitt kernel

h\_img = zeros(hight + padding\*2, width + padding\*2);

% generate new image

h\_img(1+padding:hight+padding, 1+padding:width+padding) = double(img);

new\_h = zeros(hight + padding\*2, width + padding\*2);

for i = (1+padding) : (hight+padding)

for j = (1+padding) : (width+padding)

% execute horizon filter

new\_h(i, j) = sum(sum(h\_img(i-padding : i+padding, j-padding : j+padding) .\* h\_kernel));

end

end

new\_h = uint8(new\_h(1+padding:hight+padding, 1+padding:width+padding));

figure();

imshow(new\_h);

v\_img = zeros(hight + padding\*2, width + padding\*2);

% generate new image

v\_img(1+padding:hight+padding, 1+padding:width+padding) = double(img);

new\_v = zeros(hight + padding\*2, width + padding\*2);

for i = (1+padding) : (hight+padding)

for j = (1+padding) : (width+padding)

% execute vertical filter

new\_v(i, j) = sum(sum(v\_img(i-padding : i+padding, j-padding : j+padding) .\* v\_kernel));

end

end

new\_v = uint8(new\_v(1+padding:hight+padding, 1+padding:width+padding));

figure();

imshow(new\_v);

norm\_img = zeros(hight + padding\*2, width + padding\*2);

% generate new image

norm\_img(1+padding:hight+padding, 1+padding:width+padding) = double(img);

new\_norm = zeros(hight + padding\*2, width + padding\*2);

for i = (1+padding) : (hight+padding)

for j = (1+padding) : (width+padding)

% execute grad\_norm filter

h\_norm = sum(sum(norm\_img(i-padding : i+padding, j-padding : j+padding) .\* h\_kernel))^2;

v\_norm = sum(sum(norm\_img(i-padding : i+padding, j-padding : j+padding) .\* v\_kernel))^2;

new\_norm(i, j) = sqrt(h\_norm + v\_norm);

end

end

new\_norm = uint8(new\_norm(1+padding:hight+padding, 1+padding:width+padding));

figure();

imshow(new\_norm);

**Prewitt实验结果**

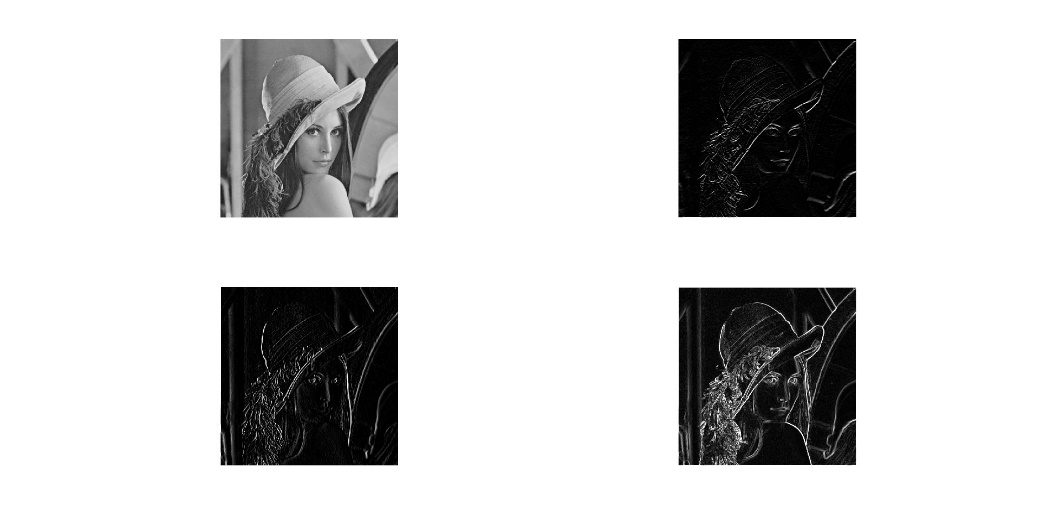
****

图 3 Prewitt实验结果图

1. **理想低通滤波器**

**步骤**

1. 导入图片；
2. 利用fft函数将空域上的图像转换为频域上的图像，并改变频率中心位置；并获得中心点的坐标，即频率最低频率；
3. 设定理想低通滤波器的设定最低频率；
4. 通过计算图像中每一点的频率到中心店的距离获得频率值，若大于设定频率则，否则保留。

**源代码**

clc

% load image

img = imread('lena.png');

subplot(131);

imshow(img);

img = double(img);

ff = fft2(img);

g = fftshift(ff);

subplot(132);

imshow(log(abs(g)),[]), color(jet(64));

[hight, width] = size(ff);

n1 = floor(hight/2);

n2 = floor(width/2);

d0 = 5;

for i = 1:hight

for j = 1:width

d = sqrt((i-n1)^2 + (j-n2)^2);

if(d <= d0)

h = 1;

else

h = 0;

end

g(i,j) = h \* g(i,j);

end

end

g = ifftshift(g);

g = uint8(real(ifft2(g)));

subplot(133);

imshow(g);

**实验结果**

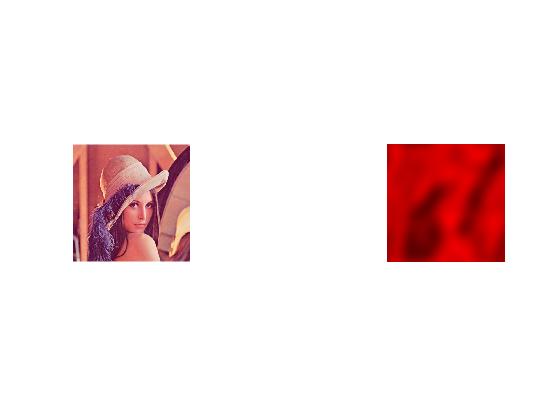
****

图 4 低通滤波器实验结果图

1. **巴特沃斯高通滤波器**

**步骤**

1. 导入图片；
2. 利用fft函数将空域上的图像转换为频域上的图像，并改变频率中心位置；并获得中心点的坐标，即频率最低频率；
3. 设定理想低通滤波器的设定高频阈值，并且设置高通滤波器的阶数以及偏移量；
4. 通过计算图像中每一点的频率到中心店的距离获得频率值，将此距离代入巴特沃斯高通滤波器的计算公式，得到滤波后的结果。

**源代码**

clc

% load image

img = imread('lena.png');

img = rgb2gray(img);

subplot(121);

imshow(uint8(img));title('原图');

% fourier trans

img = double(img);

ff = fft2(img);

g = fftshift(ff);

[hight, width] = size(ff);

h\_cent = floor(hight/2);

w\_cent = floor(width/2);

% set parameters of butterworth filter

d0 = 20;

n = 2;

c = 0;

for i = 1:hight

for j = 1:width

d = sqrt((i-h\_cent)^2 + (i-w\_cent)^2);

if(d == 0)

h = 0;

else

h = 1 / (1+ (d0/d)^(2\*n)) + c;

end

g(i, j) = h \* g(i, j);

end

end

g = ifftshift(g);

g = uint8(real(ifft2(g)));

subplot(122);

imshow(g);title('高通滤波');

**实验结果**

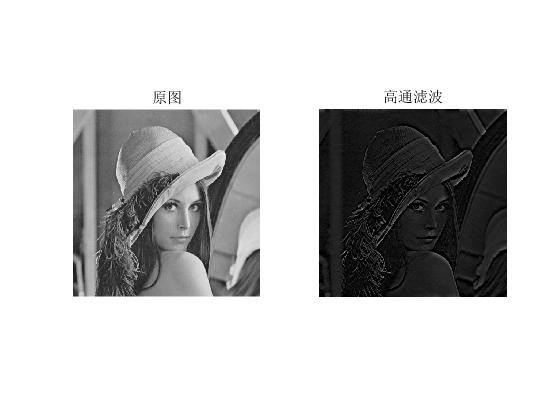
****

图 5 巴特沃斯高通滤波实验结果

## 四、分析思考

**依次给出“均值滤波器、中值滤波器、Laplace滤波器”是线性还是非线性的。**

（1）均值滤波器：其过滤器模板形如

这说明该过滤器将模板内部的几个像素点灰度相加然后取均值，因此均值滤波器是一个线性滤波器。

（2）中值滤波器：其是将模板内部的几个像素点灰度作比较，然后取其中间数赋值，因此不是一个非线性滤波器。

（3）Laplace滤波器：其滤波器模板形如

这说明是将图像中某一像素的四邻域的灰度和这一像素的灰度相减，因此这也是一个线性滤波器。