4.1(c) 计算[-9, -8, -7, -6]的DFT

```
seq = [-9, -8, -7, -6];
seq_fft = fft(seq)
```

```
seq_fft = 1×4 complex
  -30.0000 + 0.0000i   -2.0000 + 2.0000i   -2.0000 + 0.0000i   -2.0000 - 2.0000i
```

4.4(b) 使用matlab验证卷积定理

```
clc; clear;
squ_1 = [7, 6, 5, 4, -4, -5, -6, -7];
squ_2 = [2, 2, -5, -5, 6, 6, -7, -7];

% 傅里叶反变换计算
dft_1 = fft(squ_1);
dft_2 = fft(squ_2);
conv_1_2 = ifft(dft_1 .* dft_2)
```

```
conv_1_2 = 1×8
   -22.0000   -26.0000   -44.0000   -62.0000    22.0000   106.0000    44.0000   -18.0000
```

```
% 周期为8的循环卷积
cconv(squ_1, squ_2, 8)
```

```
ans = 1×8
   -22.0000   -26.0000   -44.0000   -62.0000    22.0000   106.0000    44.0000   -18.0000
```

```
% 手写循环卷积
rep_squ_1 = repmat(squ_1, [1, 2])
```

```
rep_squ_1 = 1×16
     7     6     5     4    -4    -5    -6    -7     7     6     5     4    -4 ···
```

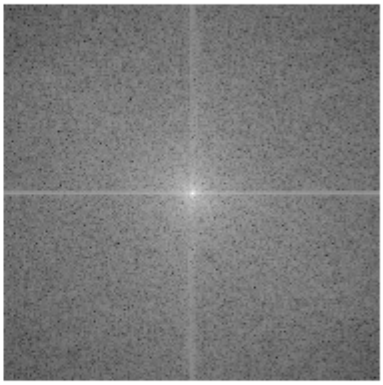
```
conv_1_2_re = zeros([1, 8]);
for i = 1:8
    conv_1_2_re(9-i) = sum(squ_2(1:8) .* rep_squ_1(17-i : -1 : 10-i));
end
conv_1_2_re
```

```
conv_1_2_re = 1×8
   -22   -26   -44   -62    22   106    44   -18
```

4.9 打开图像使用傅里叶变换后， 使用理想的低通高通滤波器，巴特沃斯滤波器，高斯滤波器来对进一步处理

```
clc; clear;
```

```
en = imread('DIP_Images\DIP Images\twins.tif');
en = rgb2gray(en);
subplot(121), imshow(en);
img_fft = fftshift(fft2(en));
subplot(122), fftshow(img_fft, 'log');
```



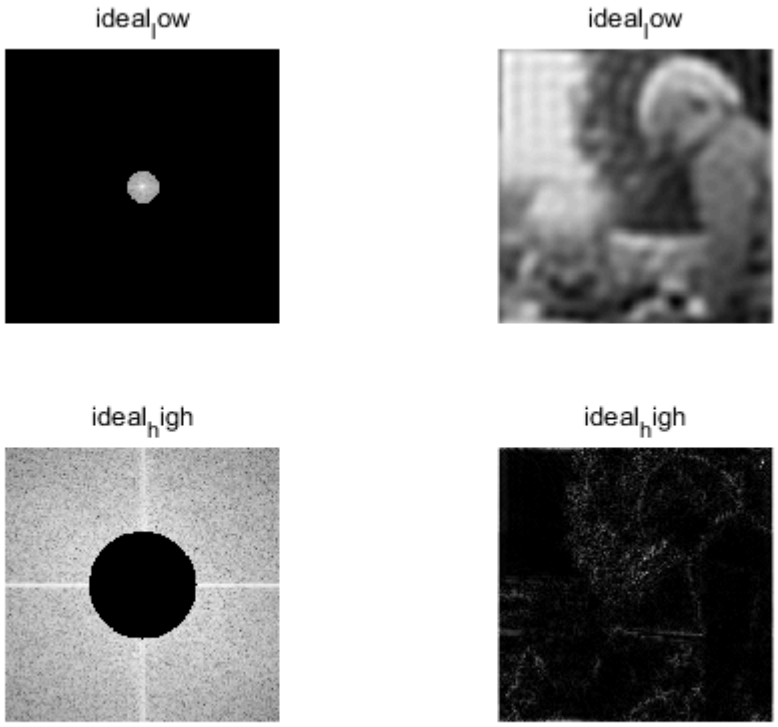
定制滤波器

```
[x, y] = meshgrid(-128:127, -128:127);
% 构造理想滤波器
z = sqrt(x.^2 + y.^2);
filter_ideal_low = (z < 15);
filter_ideal_high = (z > 50);
% 构造巴特沃斯滤波器
bl = lbutter(filter_ideal_low, 15, 1);
bh = hbutter(filter_ideal_high, 50, 1);
% 构造高斯滤波器
gf = mat2gray(fspecial('gaussian', 256, 10));
```

理想滤波器

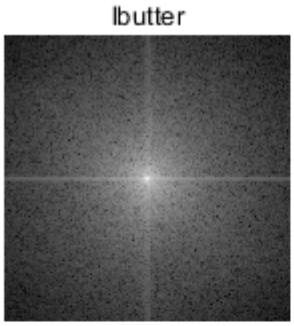
```
figure;
img_ideal_low = img_fft .* filter_ideal_low;
img_ideal_high = img_fft .* filter_ideal_high;
img_ideal_low_ifft = ifft2(img_ideal_low);
img_ideal_high_ifft = ifft2(img_ideal_high);
```

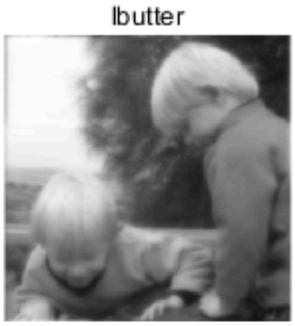
```
subplot(221), fftshow(img_ideal_low, 'log'), title('ideal_low');
subplot(222), fftshow(img_ideal_low_ifft, 'abs'), title('ideal_low');
subplot(223), fftshow(img_ideal_high, 'log'), title('ideal_high');
subplot(224), fftshow(img_ideal_high_ifft, 'abs'), title('ideal_high');
```

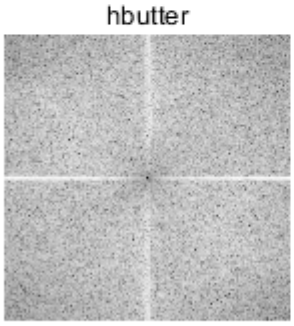


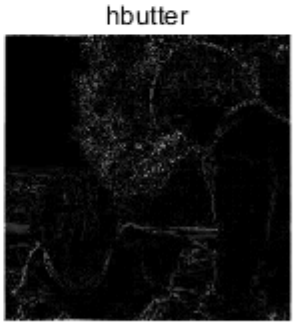
巴特沃斯滤波器

```
figure;
img_lbutter = img_fft .* bl;
img_hbutter = img_fft .* bh;
img_lbutter_ifft = ifft2(img_lbutter);
img_hbutter_ifft = ifft2(img_hbutter);
subplot(221), fftshow(img_lbutter, 'log'), title('lbutter');
subplot(222), fftshow(img_lbutter_ifft, 'abs'), title('lbutter');
subplot(223), fftshow(img_hbutter, 'log'), title('hbutter');
subplot(224), fftshow(img_hbutter_ifft, 'abs'), title('hbutter');
```



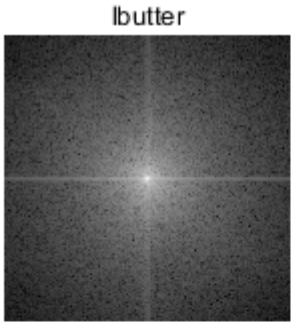


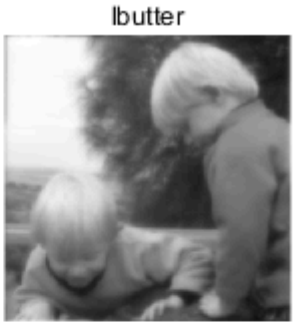


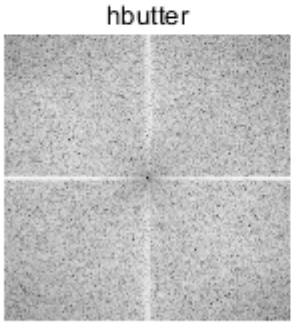


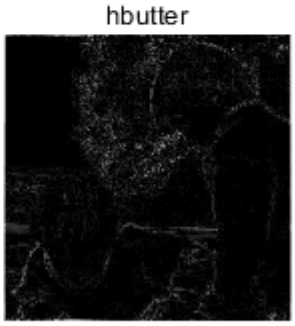
高斯滤波器

```
figure;
img_lbutter = img_fft .* bl;
img_hbutter = img_fft .* bh;
img_lbutter_ifft = ifft2(img_lbutter);
img_hbutter_ifft = ifft2(img_hbutter);
subplot(221), fftshow(img_lbutter, 'log'), title('lbutter');
subplot(222), fftshow(img_lbutter_ifft, 'abs'), title('lbutter');
subplot(223), fftshow(img_hbutter, 'log'), title('hbutter');
subplot(224), fftshow(img_hbutter_ifft, 'abs'), title('hbutter');
```

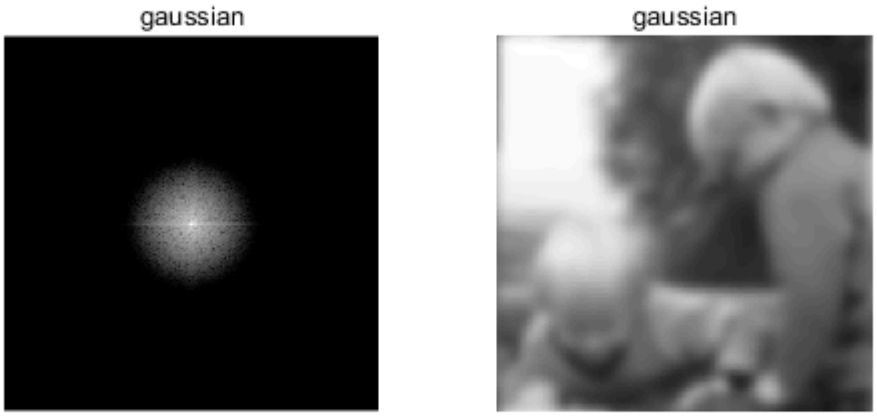






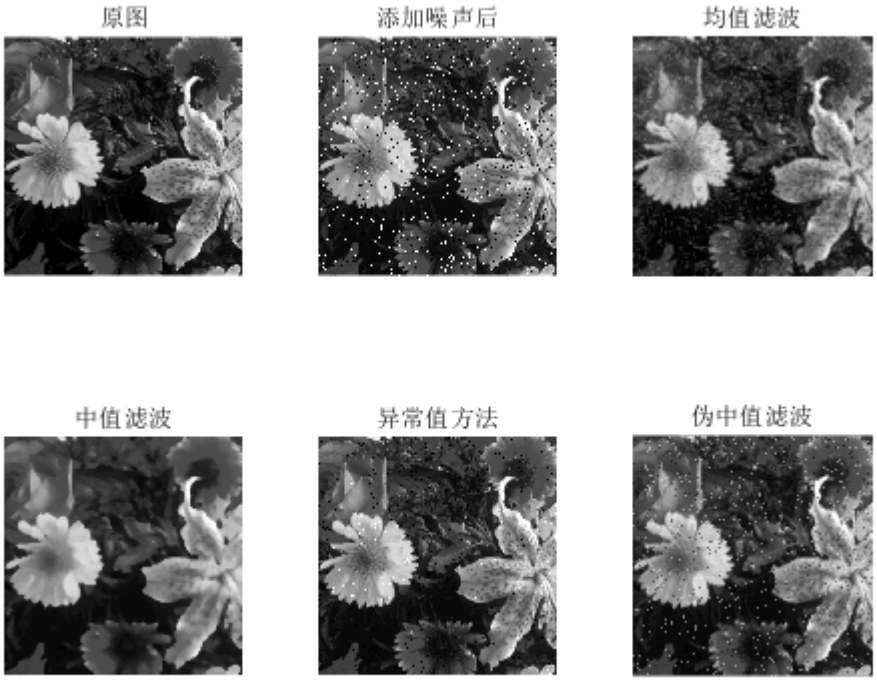



```
figure;
img_gaussian = img_fft .* gf;
img_gaussian_ifft = ifft2(img_gaussian);
subplot(121), fftshow(img_gaussian, 'log'), title('gaussian');
subplot(122), fftshow(img_gaussian_ifft, 'abs'), title('gaussian');
```



5.6 从一个彩色索引图像中生成一个灰度图，添加5%的椒盐噪声，尝试使用均值滤波器、中值滤波器、outlier method、pesudo-median滤波器进行噪声的滤波，哪种方法最好？

```
clc; clear; figure;
img = imread('DIP_Images\DIP Images\flowers.tif');
img_gray = rgb2gray(img);
img_gray = im2uint8(img_gray(30:285, 60:315));
img_noise = imnoise(img_gray, 'salt & pepper', 0.05);
subplot(231), imshow(img_gray, []), title('原图');
subplot(232), imshow(img_noise, []), title('添加噪声后');
% 使用滤波器处理噪声
filter_ave = fspecial('average');
img_ave = filter2(filter_ave, img_noise);
subplot(233), imshow(img_ave, []), title('均值滤波');
img_med = medfilt2(img_noise, [5, 5]);
subplot(234), imshow(img_med, []), title('中值滤波');
img_outlier = outlier(img_noise, 0.4);
subplot(235), imshow(img_outlier, []), title('异常值方法');
img_psmed = pseudo_median(img_noise, 3);
subplot(236), imshow(img_psmed, []), title('伪中值滤波');
```

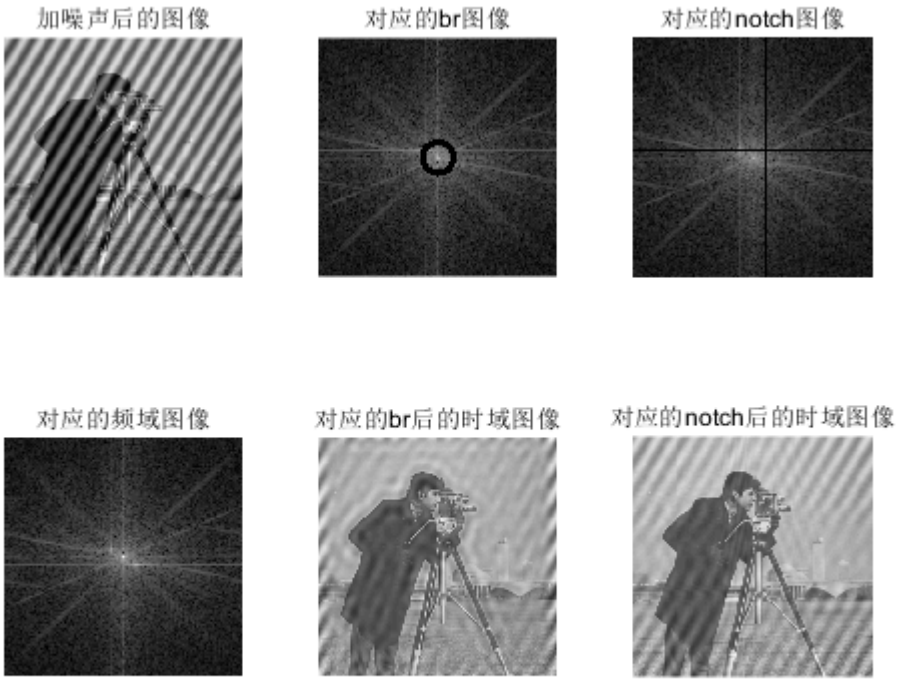


6.2 对于每个正弦波，将其添加到图像上，显示出来。尝试移除周期噪声，通过使用band-reject滤波器，或者是notch滤波器，哪一种噪声是最容易去掉的？

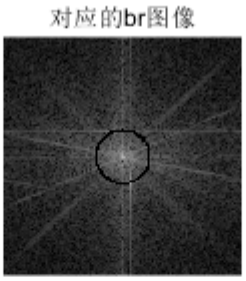
从下面的结果来看， 第二种噪声更容易去掉一些。

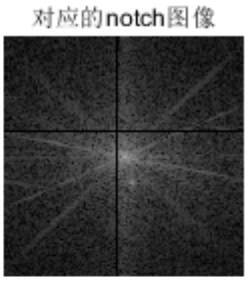
```
clc; clear; figure;
[x, y] = meshgrid(1:256, 1:256);
z = sqrt((x-129).^2+(y-129).^2);
noise_1 = 1+sin(x/3+y/5);
br = (z < 13 | z > 20);
img = imread('DIP_Images\DIP Images\cameraman.tif');
img_noise_1 = (double(img)/128+noise_1)/4;
subplot(231), imshow(img_noise_1), title('加噪声后的图像');
img_noise_1_fft = fftshift(fft2(img_noise_1));
subplot(234), fftshow(img_noise_1_fft, 'log'), title('对应的频域图像');
img_noise_1_br = img_noise_1_fft .* br;
subplot(232), fftshow(img_noise_1_br, 'log'), title('对应的br图像');
img_noise_1_after_br = ifft2(img_noise_1_br);
subplot(235), fftshow(img_noise_1_after_br, 'abs'), title('对应的br后的时域图像');
img_noise_1_fft(137, :) = 0; img_noise_1_fft(:, 142) = 0;
img_noise_1_fft(121, :) = 0; img_noise_1_fft(:, 116) = 0;
subplot(233), fftshow(img_noise_1_fft, 'log'), title('对应的notch图像');
img_noise_1_after_notch = ifft2(img_noise_1_fft);
```

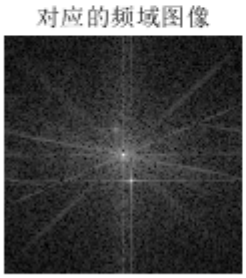
```
subplot(236), fftshow(img_noise_1_after_notch, 'abs'), title('对应的notch后的时域图像');
```



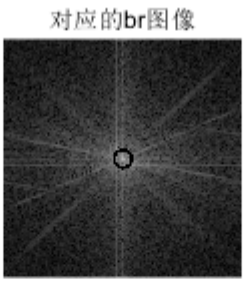

```
noise_2 = 1+sin(x/5+y/1.5);
br = (z < 27 | z > 30);
figure;
img_noise_2 = (double(img)/128+noise_2)/4;
subplot(231), imshow(img_noise_2), title('加噪声后的图像');
img_noise_2_fft = fftshift(fft2(img_noise_2));
subplot(234), fftshow(img_noise_2_fft, 'log'), title('对应的频域图像');
img_noise_2_br = img_noise_2_fft .* br;
subplot(232), fftshow(img_noise_2_br, 'log'), title('对应的br图像');
img_noise_2_after_br = ifft2(img_noise_2_br);
subplot(235), fftshow(img_noise_2_after_br, 'abs'), title('对应的br后的时域图像');
img_noise_2_fft(156, :) = 0; img_noise_2_fft(:, 137) = 0;
img_noise_2_fft(102, :) = 0; img_noise_2_fft(:, 121) = 0;
subplot(233), fftshow(img_noise_2_fft, 'log'), title('对应的notch图像');
img_noise_2_after_notch = ifft2(img_noise_2_fft);
subplot(236), fftshow(img_noise_2_after_notch, 'abs'), title('对应的notch后的时域图像');
```

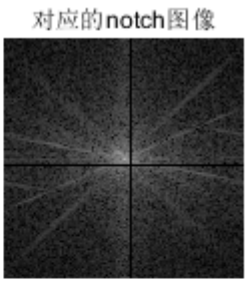


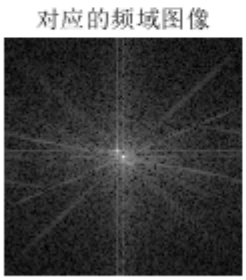





```
noise_3 = 1+sin(x/6+y/6);
br = (z < 8 | z > 11);
figure;
img_noise_3 = (double(img)/128+noise_3)/4;
subplot(231), imshow(img_noise_3), title('加噪声后的图像');
img_noise_3_fft = fftshift(fft2(img_noise_3));
subplot(234), fftshow(img_noise_3_fft, 'log'), title('对应的频域图像');
img_noise_3_br = img_noise_3_fft .* br;
subplot(232), fftshow(img_noise_3_br, 'log'), title('对应的br图像');
img_noise_3_after_br = ifft2(img_noise_3_br);
subplot(235), fftshow(img_noise_3_after_br, 'abs'), title('对应的br后的时域图像');
img_noise_3_fft(136, :) = 0; img_noise_3_fft(:, 136) = 0;
img_noise_3_fft(122, :) = 0; img_noise_3_fft(:, 122) = 0;
subplot(233), fftshow(img_noise_3_fft, 'log'), title('对应的notch图像');
img_noise_3_after_notch = ifft2(img_noise_3_fft);
subplot(236), fftshow(img_noise_3_after_notch, 'abs'), title('对应的notch后的时域图像');
```



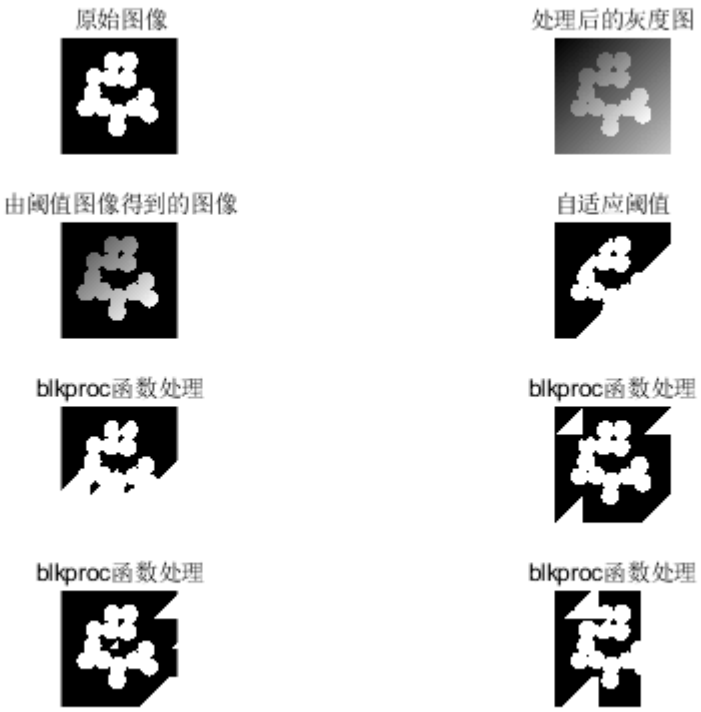




7.7 创建一个圆图，对其添加阈值，来获得一个单独的圆形，使用自适应阈值和blkproc函数，怎样调节块可以得到最好的结果？

可以看出来，`[64, 128]` 效果好些

```
clc; clear; figure;
img = imread('DIP_Images\DIP Images\circles.tif');
[x, y] = meshgrid(1:256, 1:256);
img_2 = double(img) .* ((x+y)/2+64)+x+y;
img_3 = uint8(255 * mat2gray(img_2));
subplot(421), imshow(img, []), title('原始图像');
subplot(422), imshow(img_3, []), title('处理后的灰度图');
img_3_threshold = img_3 .* uint8(img);
subplot(423), imshow(img_3_threshold, []), title('由阈值图像得到的图像');
% 自适应阈值
threshold_adp = graythresh(img_3);
subplot(424), imshow(imbinarize(img_3, threshold_adp), []), title('自适应阈值');
% 使用blkproc函数
fun = @(x) imbinarize(x, graythresh(x));
subplot(425), imshow(blkproc(img_3, [256, 64], fun), []), title('blkproc函数处理');
subplot(426), imshow(blkproc(img_3, [64, 64], fun), []), title('blkproc函数处理');
subplot(427), imshow(blkproc(img_3, [64, 128], fun), []), title('blkproc函数处理');
subplot(428), imshow(blkproc(img_3, [64, 96], fun), []), title('blkproc函数处理');
```



8.6 读取灰度图，添加噪声，使用Roberts, Prewitt, Sobel, Marr-Hildreth(Log), Canny算子来进行边缘提取

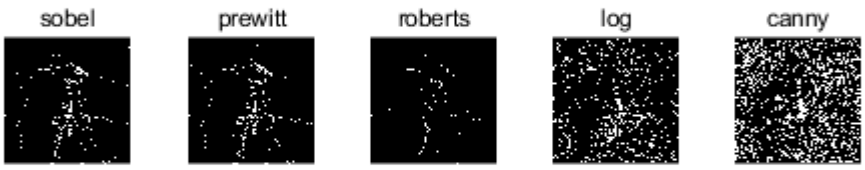
从下面的效果可以看出来，表现最好的是sobel算子，最差的是canny算子

```
clc; clear; figure;
img = imread('DIP_Images\DIP Images\cameraman.tif');
img_sp = imnoise(img, 'salt & pepper', 0.1);
img_gau = imnoise(img, 'gaussian', 0, 0.02);
method = {'sobel', 'prewitt', 'roberts', 'log', 'canny'};
img_noise = {img_sp, img_gau};

for i = 1:2
    figure;
    img_edge = edge(img_noise{i}, method{1});
    subplot(151), imshow(img_edge, []), title(method(1));
    img_edge = edge(img_noise{i}, method{2});
    subplot(152), imshow(img_edge, []), title(method(2));
    img_edge = edge(img_noise{i}, method{3});
    subplot(153), imshow(img_edge, []), title(method(3));
    img_edge = edge(img_noise{i}, method{4});
    subplot(154), imshow(img_edge, []), title(method(4));
    img_edge = edge(img_noise{i}, method{5});
    subplot(155), imshow(img_edge, []), title(method(5));
```

```
end
```


sobel
prewitt
roberts
log
canny


sobel
prewitt
roberts
log
canny

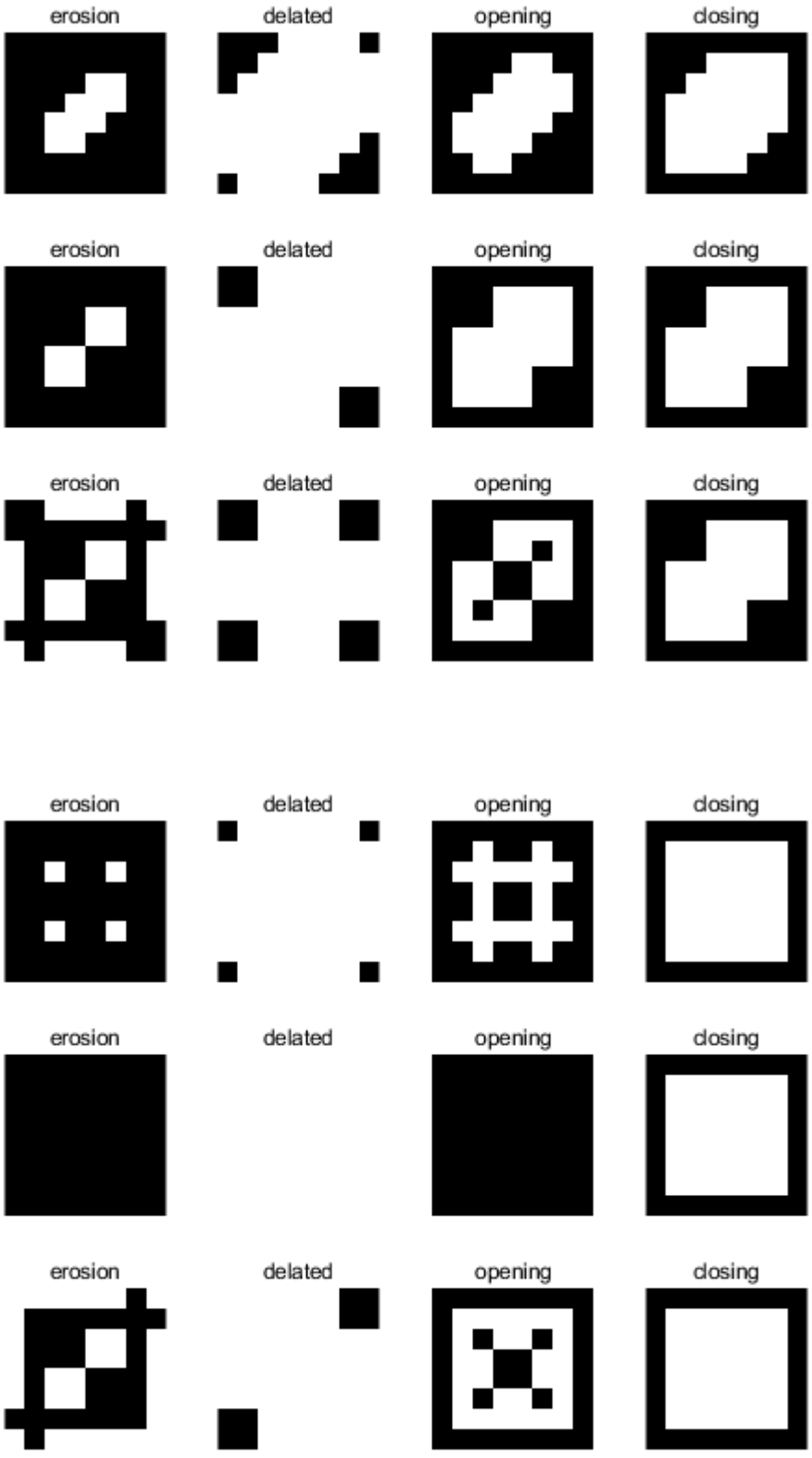
9.1 对于下面的图片A和结构元素B，计算B对A的腐蚀操作，以及AB之间的开操作，闭操作。从MATLAB中验证结果

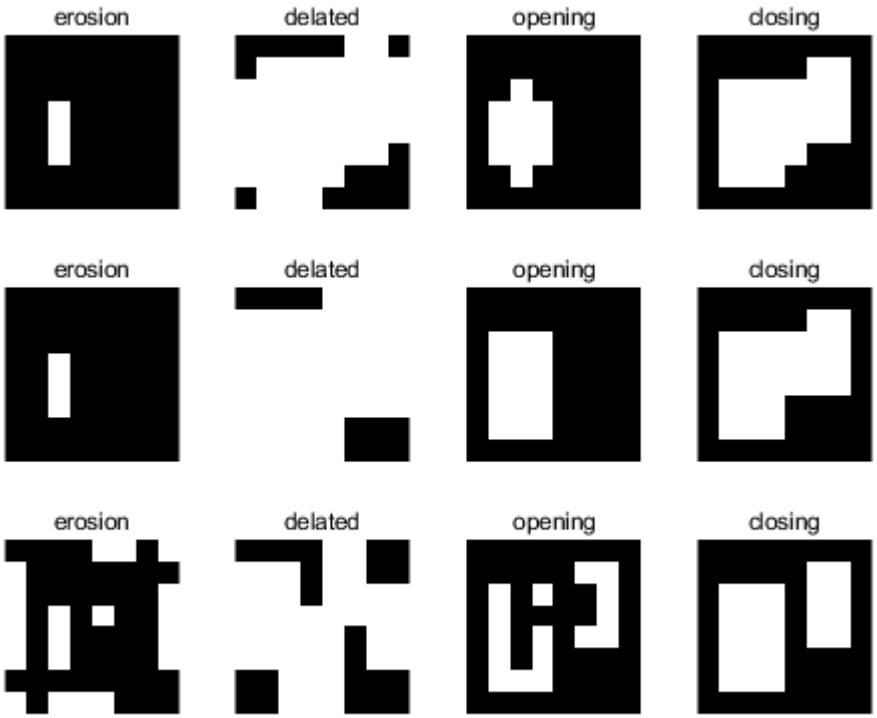
```
clc; clear;

A_1 = [ 0 0 0 0 0 0 0 0; 0 0 0 1 1 1 1 0; 0 0 0 1 1 1 1 0; 0 1 1 1 1 1 1 0;
        0 1 1 1 1 1 1 0; 0 1 1 1 1 0 0 0; 0 1 1 1 1 0 0 0; 0 0 0 0 0 0 0 0];
A_2 = [ 0 0 0 0 0 0 0 0; 0 1 1 1 1 1 1 0; 0 1 1 1 1 1 1 0; 0 1 1 0 0 1 1 0;
        0 1 1 0 0 1 1 0; 0 1 1 1 1 1 1 0; 0 1 1 1 1 1 1 0; 0 0 0 0 0 0 0 0];
A_3 = [ 0 0 0 0 0 0 0 0; 0 0 0 0 0 1 1 0; 0 1 1 1 0 1 1 0; 0 1 1 1 0 1 1 0;
        0 1 1 1 0 1 1 0; 0 1 1 1 0 0 0 0; 0 1 1 1 0 0 0 0; 0 0 0 0 0 0 0 0];
B_1 = [ 0 1 0; 1 1 1; 0 1 0];
B_2 = [ 1 1 1; 1 1 1; 1 1 1];
B_3 = [ 1 0 0; 0 0 0; 0 0 1];

A = [A_1; A_2; A_3];
B = [B_1; B_2; B_3];

for i = 1:3
    figure;
    for j = 1:3
        img_erosion = imerode(A((i-1)*8+1:i*8, :), B((j-1)*3+1:j*3, :));
        subplot(3, 4, (j-1)*4+1), imshow(img_erosion, []), title('erosion');
        img_dilated = imdilate(A((i-1)*8+1:i*8, :), B((j-1)*3+1:j*3, :));
        subplot(3, 4, (j-1)*4+2), imshow(img_dilated, []), title('delated');
        img_open = imopen(A((i-1)*8+1:i*8, :), B((j-1)*3+1:j*3, :));
        subplot(3, 4, (j-1)*4+3), imshow(img_open, []), title('opening');
        img_close = imclose(A((i-1)*8+1:i*8, :), B((j-1)*3+1:j*3, :));
        subplot(3, 4, (j-1)*4+4), imshow(img_close, []), title('closing');
    end
end
```


erosion
delated
opening
closing
erosion
delated
opening
closing
erosion
delated
opening
closing
erosion
delated
opening
closing
erosion
delated
opening
closing
erosion
delated
opening
closing




```
function fftshow(f,type)
if nargin < 2
    type = 'log';
end
if type == 'log'
    fl = log(1+abs(f));
    fm = max(fl(:));
    imshow(im2uint8(fl/fm));
elseif type == 'abs'
    fa = abs(f);
    fm = max(fa(:));
    imshow(fa/fm);
else
    error('type must be abs or log.');
end
end

function out = lbutter(im, d, n)
height = size(im, 1);
width = size(im, 2);
[x, y] = meshgrid(-floor(width/2):floor((width-1)/2), -floor(height/2):floor((height-1)/2));
out = 1./(1+(sqrt(2)-1)*((x.^2+y.^2)/d^2).^n);
end

function out = hbutter(im, d, n)
out = 1-lbutter(im, d, n);
```

```
end

function res = outlier(im, d)
f = [0.125, 0.125, 0.125; 0.125, 0, 0.125; 0.125, 0.125, 0.125];
imd = im2double(im);
imf = filter2(f, imd);
r = abs(imd-imf)-d>0;
res = im2uint8(r.*imf+(1-r).*imd);
end

function psmed = pseudo_median(im, n)
% im: 输入图片
% n: 卷积核大小

height = size(im, 1); % 行数
width = size(im, 2); % 列数
% padding
im_padding = [zeros(size(im, 1), floor(n/2)), im, zeros(size(im, 1),  floor(n/2))];
im_padding = [
    zeros(floor(n/2), size(im_padding, 2));
    im_padding;
    zeros(floor(n/2), size(im_padding, 2));
];
psmed = zeros(size(im));
length_batch = n * n;
num = ceil(length_batch/2);
% 循环行
for i = 1:height
    % 循环列
    for j = 1:width
        batch = im_padding(i:i+n-1, j:j+n-1);
        batch = batch(:);
        min_in = zeros(1, num);
        max_in = zeros(1, num);
        for k = 1:length_batch-(num-1)
            min_in(k) = min(batch(k:k+num-1));
            max_in(k) = max(batch(k:k+num-1));
        end
        psmed(i, j) = max(min_in)/2 + min(max_in)/2;
    end
end
end
```