【Experiment name】

Lab 4 Image frequency domain filter, median filter

【Purpose】

Master the image frequency domain filter and median filter method in digital image processing based on matlab .

【Experimental content】

Use the given image to complete the method of image frequency domain filter and median filter .

PRO JECT 04

The three experimental requirements of this project are closely related to each other. I used a complete . m script to realize these three functions, and the experimental results are shown in the figure.

Core algorithm:

① Realization of each filter function

```
%% 理想低通滤波函数
function[Bw]=ILPF(D0,P,Q,m,n)
    Bw = zeros(P, Q);
    for u = 1 : P
        for v = 1 : Q
            temp = ((u-(m+1.0))^2 + (v-(n+1.0))^2)^(1/2);
            if temp<=D0
                Bw(u,v)=1;
            end
        end
    end
end
%% 布特沃斯滤波函数
function[Bw]=BLPF(D0,N,P,Q,m,n)
    Bw = zeros(P, Q);
    a = D0^(2 * N);
    for u = 1 : P
        for v = 1 : Q
            temp = (u-(m+1.0))^2 + (v-(n+1.0))^2;
            Bw(u, v) = 1 / (1 + (temp^N) / a);
        end
    end
end
%% 高斯低通滤波函数
function[Bw]=GLPF(D0,P,Q,m,n)
    Bw = zeros(P, Q);
    a = 2*D0^2;
    for u = 1 : P
        for v = 1 : Q
            temp = ((u-(m+1.0))^2 + (v-(n+1.0))^2)^(1/2);
            Bw(u, v) =exp(-temp^2/a);
        end
    end
end
```

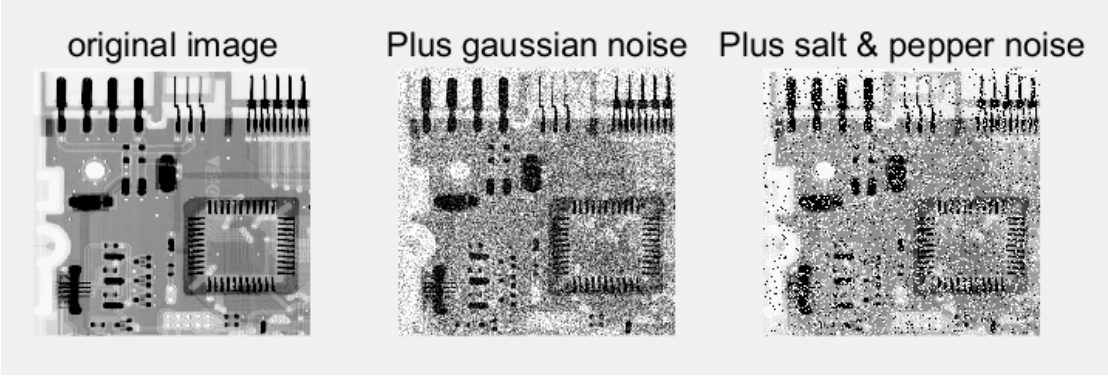
② Image expansion and Fourier transform realization

```
% 对图像填充0,并且乘以(-1)^(x+y) 以移到变换中心
for i = 1 : m
    for j = 1 : n
        fp(i, j) = double(image_in(i, j)) * (-1)^(i+j);
    end
end
% 对填充后的图像进行傅里叶变换
F1 = fft2(fp);
```

PRO JECT 05 -

Add Gaussian noise and salt and pepper noise respectively based on the given image.

Experimental results:



Core algorithm:

①Respectively realize Gaussian noise function and salt and pepper noise function

```
%% 高斯噪声函数
function [result]=Gaussian(image,a,b)
[row,col]=size(image);
u1=rand(row,col);
u2=rand(row,col);
x=b*sqrt(-2*log(u1)).*cos(2*pi*u2)+a;
result=double(image)/255+x;
result=uint8(255*result);
disp('高斯噪声处理完成');
end
%% 椒盐噪声函数
function [result]=SaltPepper(image,p1,p2)
[row,col]=size(image);
result=image;
a1=rand(row,col)<p1;
a2=rand(row,col)<p2;
result(a1&a2)=0;
result(a1& ~a2)=255;
disp('椒盐噪声处理完成');
end
```

【Experiment Summary】

In this experiment, I learned the image frequency domain filter and median filter method in digital image processing based on matlab . The first half of the experiment mainly deals with the frequency domain, so it is necessary to be able to convert the spatial domain into the frequency domain, so it is necessary to be able to implement the step of expanding the image to Fourier transform well. At the same time, since the three filtering methods are relatively similar except for the filtering function, I will list the three filtering functions corresponding to the

formula separately, share the same frequency conversion function, and obtain the same result as the book. In the experiment of the median filtering part, we should pay attention to how to select the median, not to be confused with the mean filtering. Through this experiment, I have deepened my understanding of the concepts related to frequency domain filtering and median filtering, enhanced the ability to code and use, and laid a good foundation for future experiments.

appendix:

PROJECT 04

```
%% PROJECT 04
close all;
clear all;
clc;

%% Get image and parameters
showX=3;
showY=6;
I = imread('Fig_characters_test_pattern.tif');
subplot(showX,showY,1+showY*0);imshow(I);title('Original');
subplot(showX,showY,1+showY*1);imshow(I);title('Original');
subplot(showX,showY,1+showY*2);imshow(I);title('Original');
D0=80;
N=2;
alf=[5,15,30,80,230];


%% Call the filter function and display the result
for k=1:showY
D0=alf(k);
result=Bfilter(I,D0,N,1);
subplot(showX,showY,k+showY*0+1);imshow(result);title('ILPF');
result=Bfilter(I,D0,N,2);
subplot(showX,showY,k+showY*1+1);imshow(result);title('BLPF');
result=Bfilter(I,D0,N,3);
subplot(showX,showY,k+showY*2+1);imshow(result);title('GLPF');
end

%% frequency domain filter
function [image_out] = Bfilter(image_in, D0, N, choose)
% filter in the frequency domain
% The input is the grayscale image that needs to be filtered, the cut-off frequency D0, the order N of the Butterworth filter
% The output is the grayscale image after filtering

[m, n] = size(image_in);
P = 2*m;
Q = 2 * n;
```

```
fp = zeros(P, Q);
% Fill the image with 0, and multiply by (-1)^(x+y) to move to the center of the transformation
for i = 1 : m
for j = 1 : n
fp(i, j) = double(image_in(i, j)) * (-1)^(i+j);
end
end
% Perform Fourier transform on the filled image
F1 = fft2(fp);

% Generate a Butterworth filter function, centered at (m+1,n+1)
if choose==1
Bw=ILPF(D0,P,Q,m,n);
elseif choose==2
Bw=BLPF(D0,N,P,Q,m,n);
elseif choose==3
Bw=GLPF(D0,P,Q,m,n);
end

% to filter
G = F1 .* Bw;
% inverse Fourier transform
gp = ifft2(G);
% processed image
image_out = zeros(m, n, 'uint8');
gp = real(gp);
g = zeros(m, n);
for i = 1 : m
for j = 1 : n
g(i, j) = gp(i, j) * (-1)^(i+j);

end
end
mmax = max(g(:));
mmin = min(g(:));
range = mmax-mmin;
for i = 1 : m
for j = 1 : n
image_out(i,j) = uint8(255 * (g(i,j)-mmin) / range);
end
end
end

%% ideal low-pass filter function
function[Bw]=ILPF(D0,P,Q,m,n)
Bw = zeros(P, Q);
for u = 1 : P
for v = 1 : Q
temp = ((u-(m+1.0))^2 + (v-(n+1.0))^2)^(1/2);
if temp<=D0
Bw(u,v)=1;
end
end
end
end
%% Butterworth filter function
function[Bw]=BLPF(D0,N,P,Q,m,n)
Bw = zeros(P, Q);
```

```
a = D0^(2 * N);
for u = 1 : P
for v = 1 : Q
temp = (u-(m+1.0))^2 + (v-(n+1.0))^2;
Bw(u, v) = 1 / (1 + (temp^N) / a);
end
end
end
%% Gaussian low-pass filter function
function[Bw]=GLPF(D0,P,Q,m,n)
Bw = zeros(P, Q);
a = 2*D0^2;
for u = 1 : P
for v = 1 : Q
temp = ((u-(m+1.0))^2 + (v-(n+1.0))^2)^(1/2);
Bw(u, v) =exp(-temp^2/a);
end
end
end
```

P ROJECT 05

```
%% PROJECT 05-01 [Multiple Uses] Noise Generators
close all;
clear all;
clc;

% original image
img = imread('ckt-board-orig.tif');
subplot(1,3,1);imshow(img);title('original image');
% parameter settings
mean=0.2; % Gaussian mean
variance=0.01; % Gaussian variance
probability=0.2; % salt and pepper rate
p1=0.2;
p2=0.2;

% add Gaussian noise
% processed1 = imnoise(img, 'gaussian', mean, variance);
processed1 = Gaussian(img,variance,mean);
subplot(1,3,2);
imshow(processed1);
title('Plus gaussian noise');

% Add salt and pepper noise
% processed2 = imnoise(img, 'salt & pepper', probability);
processed2 = SaltPepper(img,p1,p2);
subplot(1,3,3);
imshow(processed2);
title('Plus salt & pepper noise');

disp('Gaussian noise parameters');
disp(['mean: ', num2str(mean), 'variance: ', num2str(variance)]);
disp('salt and pepper noise');
disp(['p1: ', num2str(p1), ' p2: ', num2str(p2)]);
```

```
%% Gaussian noise function
function [result]=Gaussian(image,a,b)
[row,col]=size(image);
u1=rand(row,col);
u2=rand(row,col);
x=b*sqrt(-2*log(u1)).*cos(2*pi*u2)+a;
result=double(image)/255+x;
result=uint8(255*result);
disp('Gaussian noise processing completed');
end
%% salt and pepper noise function
function [result]=SaltPepper(image,p1,p2)
[row,col]=size(image);
result=image;
a1=rand(row,col)<p1;
a2=rand(row,col)<p2;
result(a1&a2)=0;
result(a1& ~a2)=255;
disp('Salt and pepper noise processing completed');
end
```