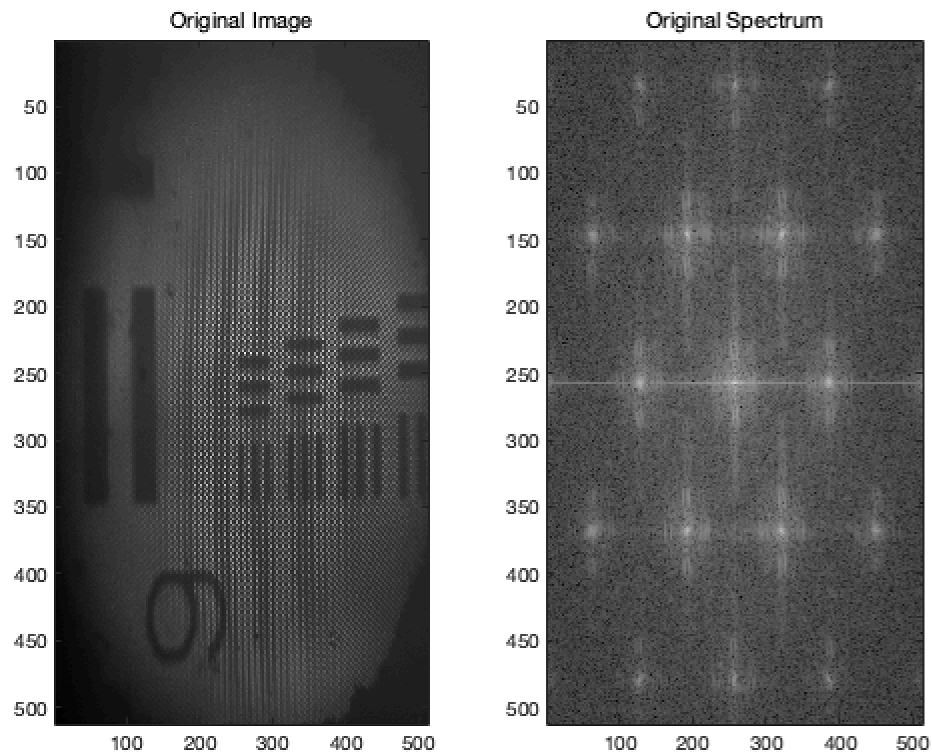
Table of Contents


```
clc;
clear;
close all;
```

Load the Image

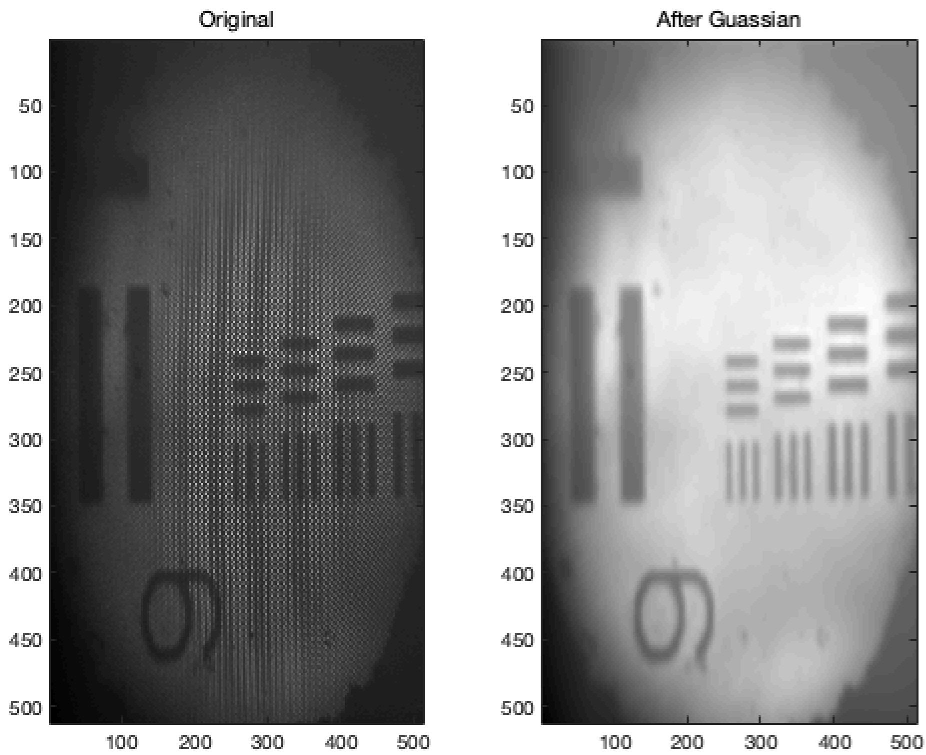
```
img = double(imread("HW_lecture9_image.tiff"))/65535*255;

figure;
subplot(1,2,1);
axis on;colormap(gray);imagesc(img);title('Original Image');
img_fft = fftshift(fft2(img));
subplot(1,2,2);
axis on;colormap(gray);imagesc(log(abs(img_fft)+1));title('Original
 Spectrum');
```

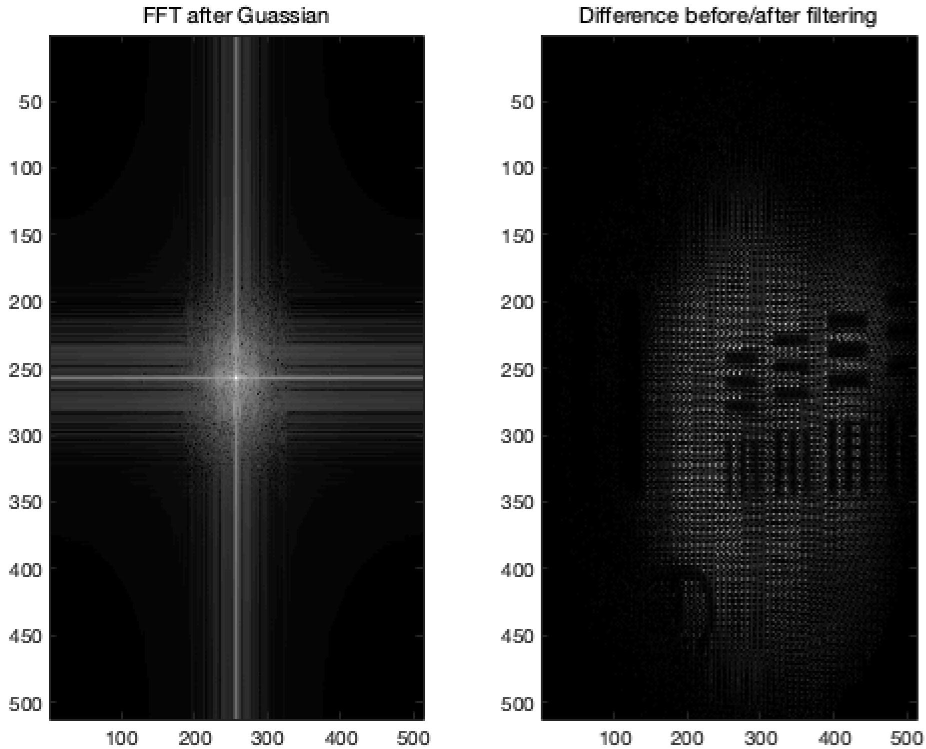


guassian filter

```
clear
close all

img=double(imread("HW_lecture9_image.tiff"))/65535*255;

sigma=3;
window=double(uint8(3*sigma)*2+1);%half of winsize

H=fspecial('gaussian', window, sigma);%generate kernal
%using replicate to keep the block edge
img_gauss=imfilter(img,H,'replicate');

figure()
subplot(1,2,1),imagesc(img),colormap(gray),axis on,title('Original');
subplot(1,2,2),imagesc(img_gauss);
colormap(gray),axis on;title('After Guassian');

img_fft = ifftshift(fft2(img_gauss));
figure();
subplot(1,2,1)
colormap(gray),axis on;imagesc(log(abs(img_fft)+1)),title('FFT after
 Guassian');
```

```
subplot(1,2,2)
colormap(gray),axis on;imagesc(abs(img - img_gauss));
title('Difference before/after filtering');
% Although it's obvious the image suffers from periodic noise#we try
 some
% really simple methods(gaussian, mean) at first to see what will
 happen.

% Easy to see the guassian remove most of the high frequency noise but
% scarify some high frequency part
```

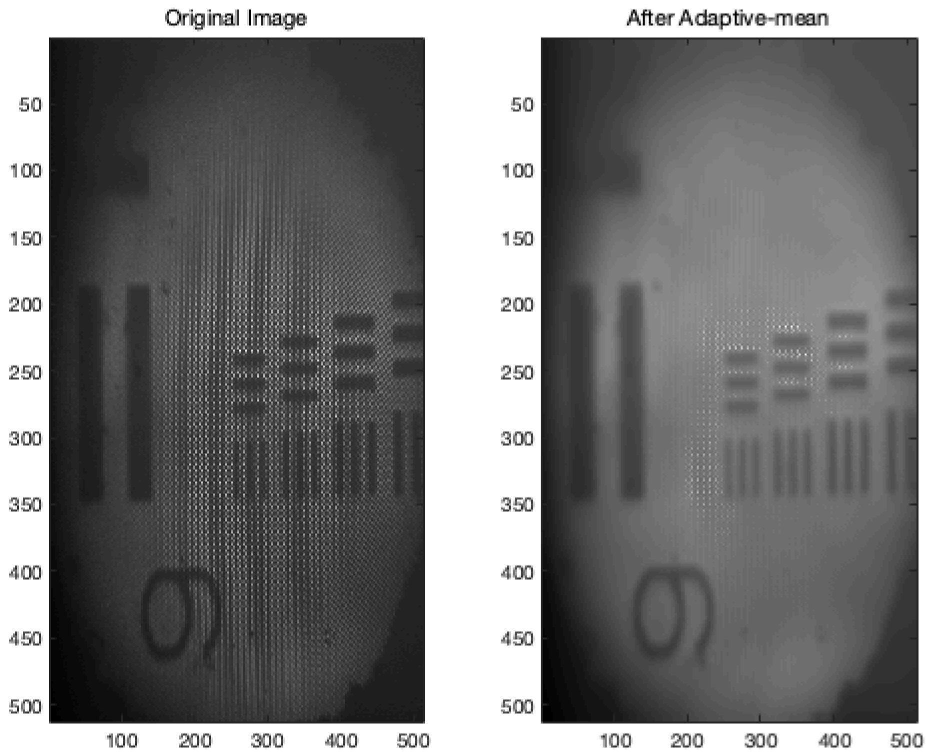


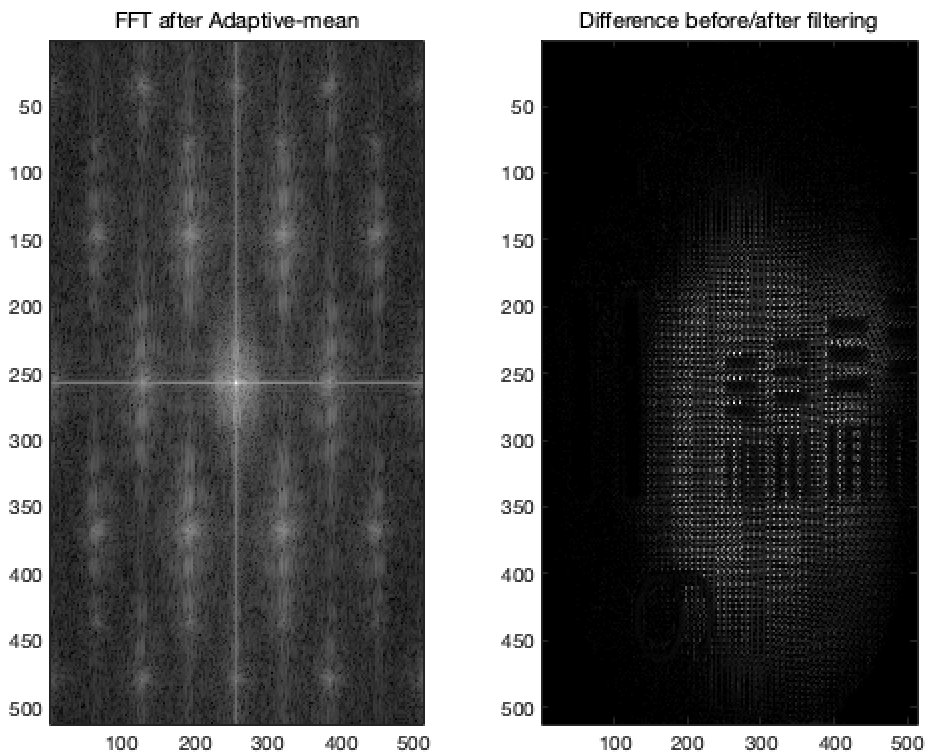


adaptive mean

```
clear
close all

img=double(imread("HW_lecture9_image.tiff"))/65535*255;
Wsize = 5;
output = My_adaptivemean2(img,Wsize);
figure();
subplot(1,2,1),imagesc(img),colormap(gray),title('Original Image');
subplot(1,2,2),imagesc(output),colormap(gray);
title('After Adaptive-mean');

img_fft = fftshift(fft2(output));
figure();
subplot(1,2,1)
colormap(gray),axis on;imagesc(log(abs(img_fft)+1)),title('FFT after
 Adaptive-mean');

subplot(1,2,2)
colormap(gray),axis on;imagesc(abs(img - output));
title('Difference before/after filtering');
% We also try the adaptive mean, easy to see it still contains many
 high
```

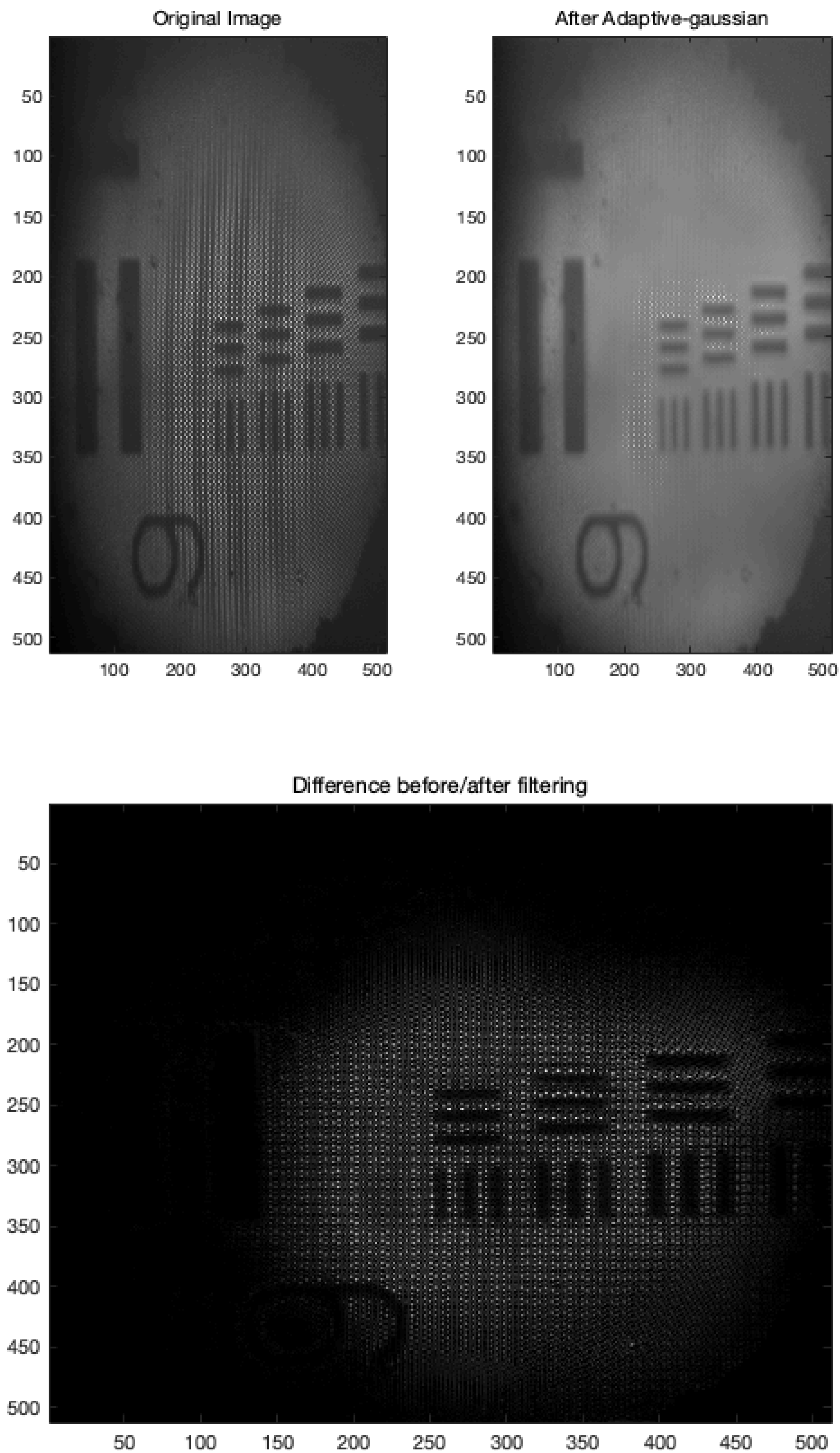
```
% frequency noise nearby the edge of the cneter black bars. It's
 because
% in these place the Var_local > Var_globel so the filter keep the
 original
% pixel value.

% Following are 2 self-designed adaptive filter, the performances are
 not
% very good, because 1) the adaptive gaussian shares have the same
 problem
% as the adaptive mean; 2) the adaptive bilateral tends to keep the
 noise
% since there is a "edge". Therefore, a small sigma_r doesn't help
 with
% denoising and a big sigma_r will change the bilateral to be a
 gaussian.
```





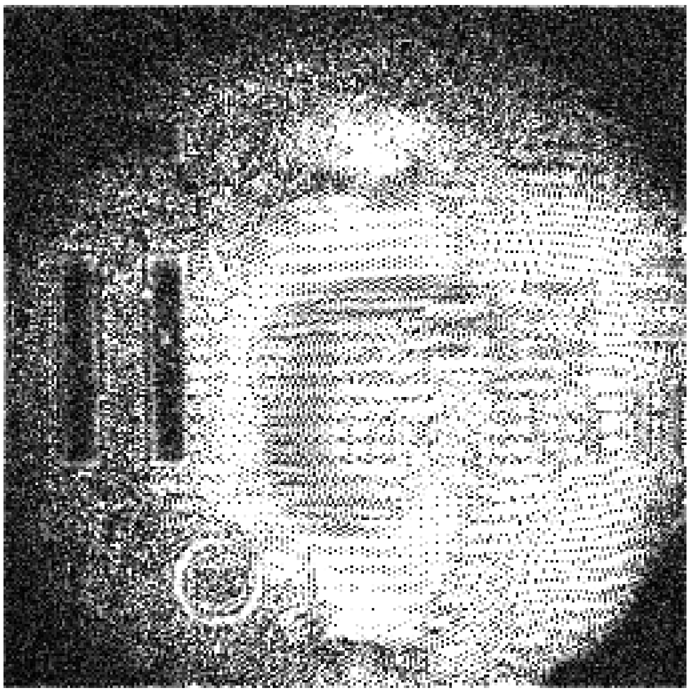
adaptive gaussian

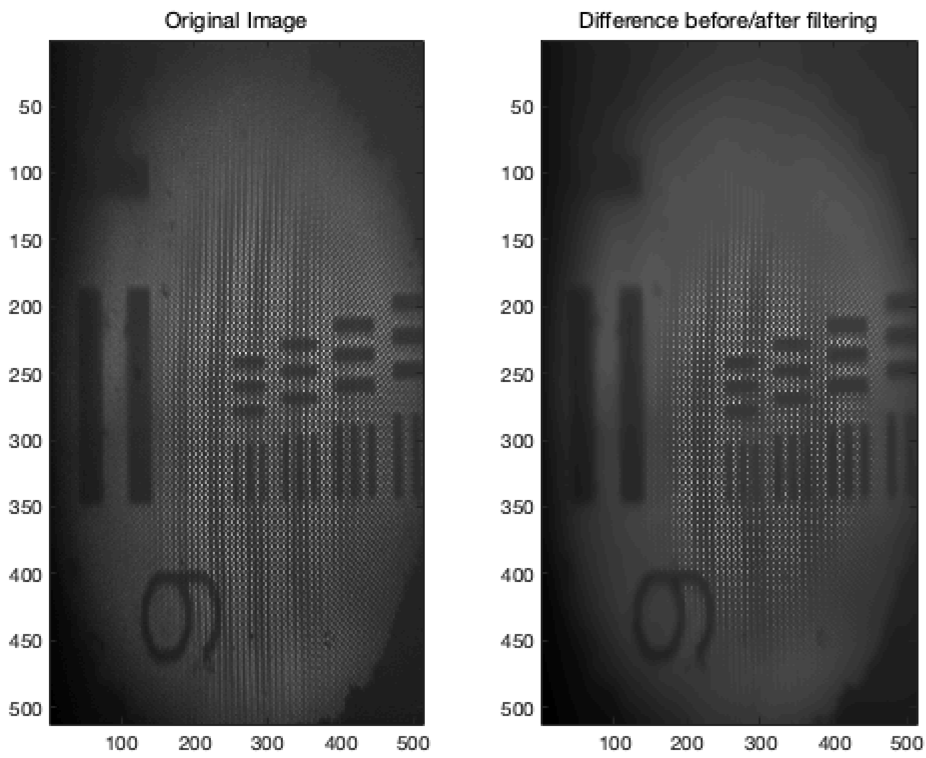
```
clear
close all

img=double(imread("HW_lecture9_image.tiff"))/65535*255;
Wsize = 5;
output = My_adaptiveGaussian(img,Wsize);
figure();
subplot(1,2,1),imagesc(img),colormap(gray),title('Original Image');
subplot(1,2,2),imagesc(output),colormap(gray);
title('After Adaptive-gaussian');
figure();
colormap(gray),imagesc(abs(img - output));
title('Difference before/after filtering');
```


Original Image
After Adaptive-gaussian
Difference before/after filtering
50
100
150
200
250
300
350
400
450
500

adaptive biliateral

```
clear
close all

img=double(imread("HW_lecture9_image.tiff"))/65535*255;
Wsize = 5;
% variance of the image
sigma_s=5;
sigma_r=5;
output = Adaptive_bilateral(img,sigma_r,sigma_s,Wsize);
figure();
subplot(1,2,1),imagesc(img),colormap(gray),title('Original Image');
subplot(1,2,2),imagesc(output),colormap(gray),title('After Adaptive-
Gaussian');
figure();
colorbar(),imshow(abs(img - output)),title('Difference before/after
 filtering');
```

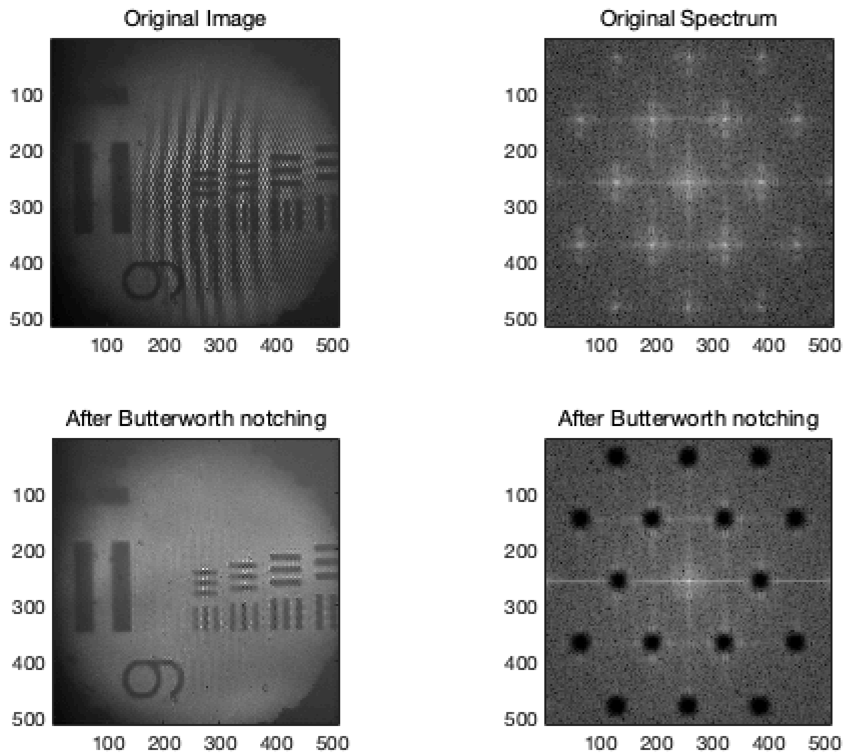


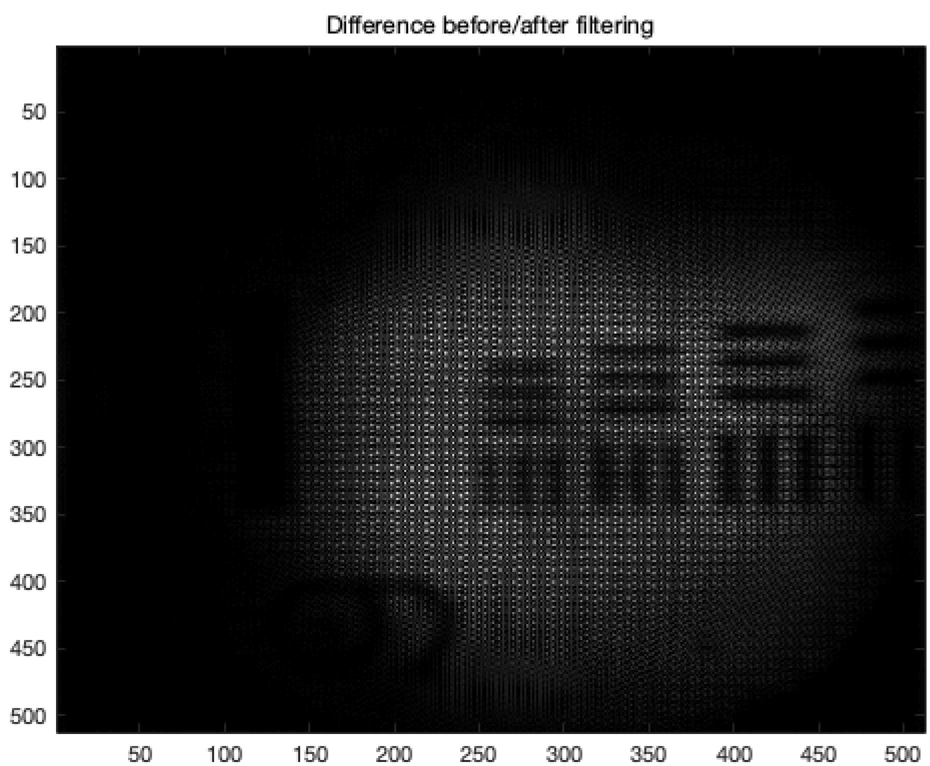


notching filter to remove periodical noise

```
clear
close all
img=double(imread("HW_lecture9_image.tiff"))/65535*255;
img_fft = fftshift(fft2(img));

% Remove all the high frequency part
center = [128,35;256,35;384,35;
    64,145;64*3,145;64*5,145;64*7,145;
    131,256;385,256;
    64,367;64*3,367;64*5,367;64*7,367;
    128,480;256,480;384,480];
% Use a high pass Butterworth Mask
Wsize = 30; % half of the mask size
n = 4; % the order of the butterworth
D_0 = 25; %the cut-f of highpass
[x, y]=meshgrid(-Wsize:Wsize,-Wsize:Wsize);
D = sqrt(x.^2 + y.^2);
b_filter = 1./(1+(D_0./D).^(2*n));
% Apply the filter
img_bf = img_fft;
for i = 1:size(center,1)
    % notice the x, y axis
    img_bf(center(i, 2)-Wsize:center(i, 2)+Wsize, ...
```

```
        center(i, 1)-Wsize:center(i, 1)+Wsize) ...
        = img_bf(center(i, 2)-Wsize:center(i, 2)+Wsize, ...
        center(i, 1)-Wsize:center(i, 1)+Wsize).* b_filter;
end
img_notching = ifft2(ifftshift(img_bf));
figure;
subplot(2,2,1)
imagesc(img);axis image;colormap(gray);title('Original Image')
subplot(2,2,2)
imagesc(log(1+abs(img_fft)));axis image;colormap(gray);title('Original
 Spectrum')
subplot(2,2,3)
imagesc(abs(img_notching));axis image;colormap(gray);title('After
 Butterworth notching')
subplot(2,2,4)
imagesc(log(1+abs(img_bf)));axis image;colormap(gray);title('After
 Butterworth notching')

figure();
colormap(gray),imagesc(abs(img - abs(img_notching)));
title('Difference before/after filtering');

% Here a hp butterworth filter is applied to remove the periodic
 noise,
% however, it still keeps some high-frequecy noise and sacrify part of
% edges. A way to solve this problem is to control the D_0, but it's
 not
% very flexible.
```


Original Image
Original Spectrum
After Butterworth notching
After Butterworth notching
100
200
300
400
500


Difference before/after filtering
50
100
150
200
250
300
350
400
450
500

Published with MATLAB® R2020a