


SYDE 575 Image Processing

Lab 3: Filter Design and Image Restoration in Frequency Domain

Group 23: Laura Chambers, ID: 

1 Introduction

The purpose of this lab is to gain an understanding of Fourier analysis and experience with image restoration and filter design in the spatial-frequency domain.

In Fourier analysis, a Fourier transform is used to break down a function into its contributing frequencies, which are visually represented in a Fourier spectrum.

In previous labs, image restoration and filtering were done in the spatial domain. Images can also be restored in the spatial-frequency domain by designing and applying filters to the Fourier spectrum of the image. This lab will consider low-pass filters, which filter out high frequencies, and notch filters, which filter out specific frequencies.

The MATLAB code for this lab can be found in the Appendix.

2 Fourier Analysis

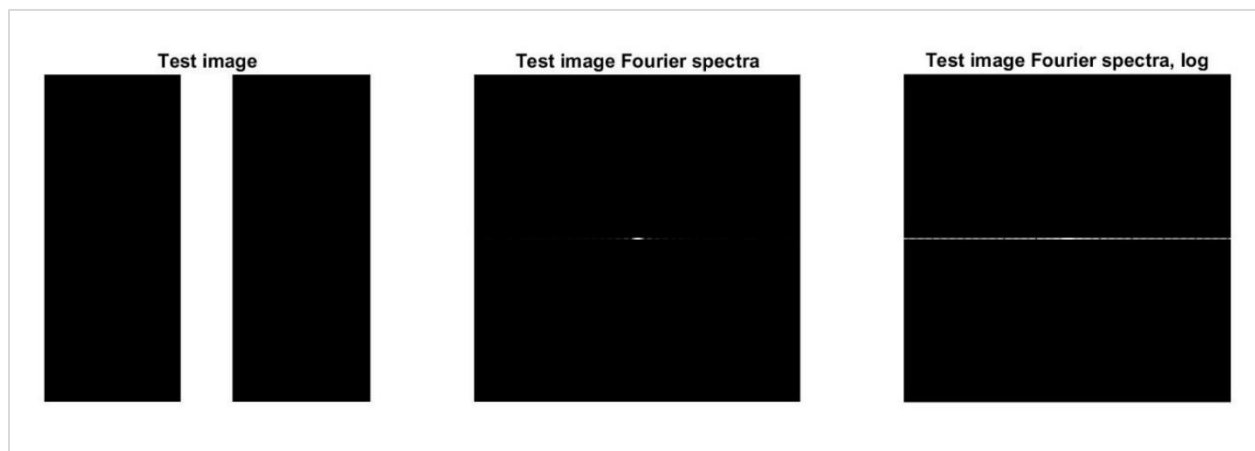


Figure 2-1 Test image and its Fourier spectra

1. The general distribution of energy in the Fourier spectra is concentrated along the horizontal axis, corresponding to the spatial rates of change in intensity in the test image. The origin ($u=v=0$) represents the average intensity of the image, which is not very prominent given the majority of pixels are black. The low frequencies correspond to the blocks of black and white in which pixels values do not change. The high frequencies correspond to the vertical edges in the test image, where the pixel values rapidly change from black to white and vice versa. The spectra does not extend in the vertical direction because the test image does not change in the vertical axis.

2. The Fourier spectra only contains frequencies along the horizontal / u-axis and none in the vertical / v-axis. Bases on this characteristic, we can infer that the test image contains only vertical edges and does not contain horizontal or diagonal edges. Based on the limited brightness at the origin, we can infer that the average intensity of the image is fairly dark.

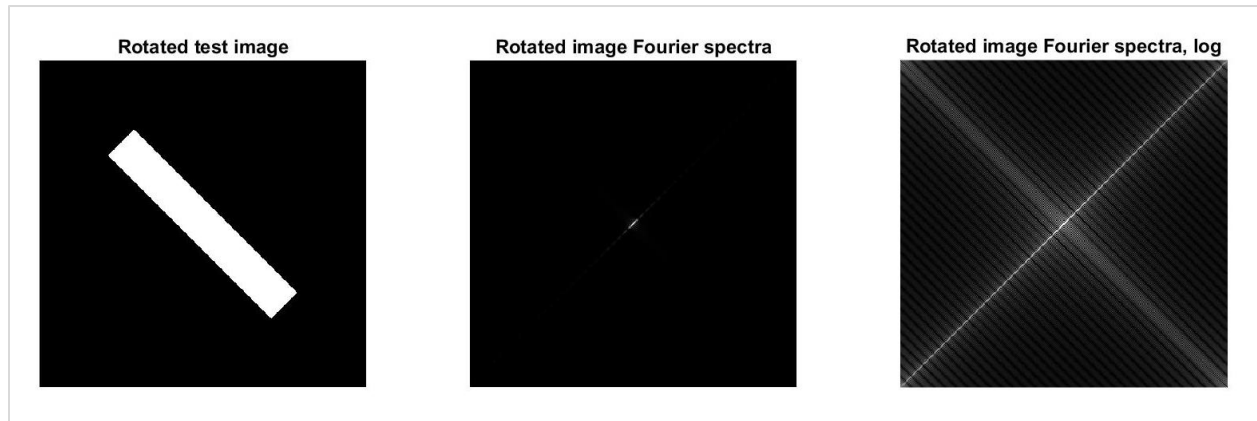


Figure 2-2 Test image rotated 45 degrees and its Fourier spectra

3. The Fourier spectra of the rotated test image also rotated relative to the original. The rotated Fourier spectra shows energy distributed throughout the u-v plane, rather than in one axis only. The energy appears as an “x” pattern. This is because the test image now has edges at two different angles. The angles of the lines in the Fourier spectra correspond to the angles of the edges in the test image of ± 45 degrees.
4. By observing the test images, it can be concluded that the angles of lines in the Fourier spectra correspond to edges in the image. Energy distributed along the horizontal axis of the spectrum represents spatial rates of change in the horizontal axis of the image. Likewise, energy distributed along the vertical axis of the spectrum represents spatial rates of change in the vertical axis of the image.

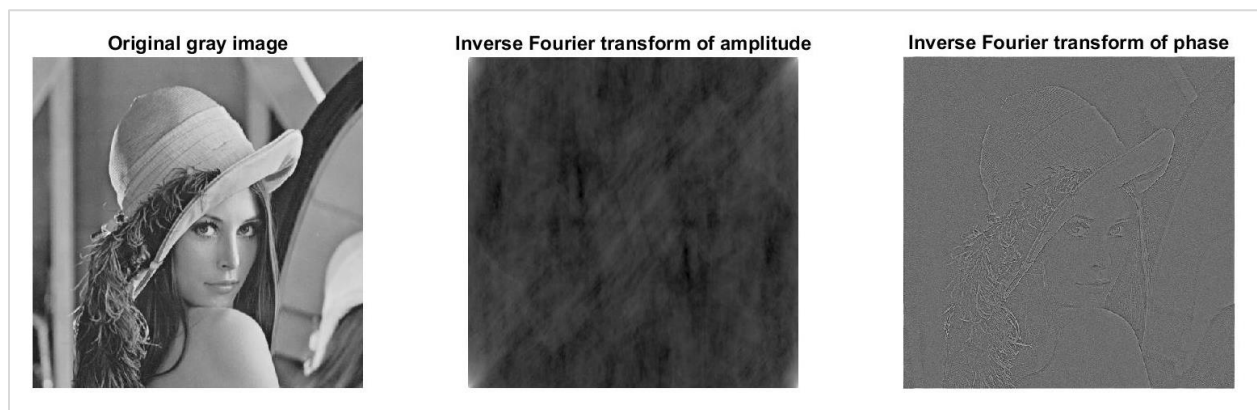


Figure 2-3 Contribution of amplitude and phase to the original image

5. The reconstructed image from the amplitude component does not resemble the original image. It appears as an abstract pattern of gray patches. The amplitude component captures the intensity information of an image.
6. The reconstructed image from the phase component shows a recognizable outline of the original image within an otherwise uniform gray intensity. The phase component captures the shape characteristics of an image, like edges.

3 Noise Reduction in the Frequency Domain

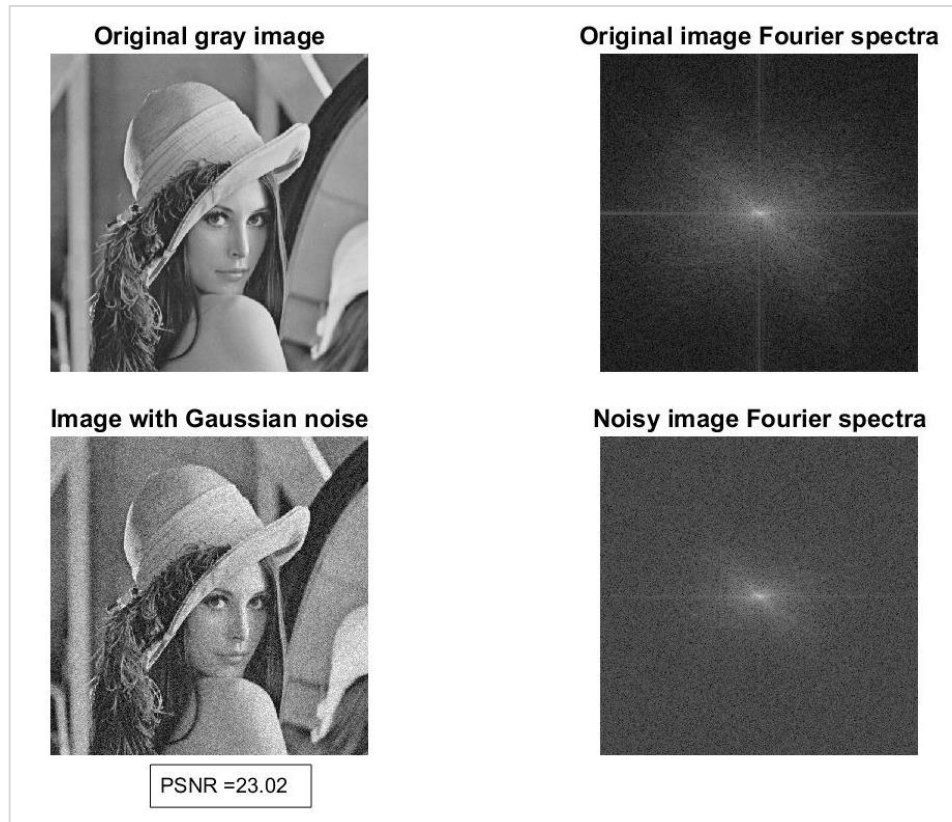


Figure 3-1 Original image and Log Fourier spectra with and without Gaussian noise

7. The Fourier spectrum for the noisy image appears less clear and more muted than the spectrum of the original image. In the Fourier spectrum for the original image, the horizontal and vertical axis as well as a diagonal line from the upper left to bottom right are visible. These features are less distinct in the noisy spectrum. This is because the Gaussian noise obscures the shape characteristics of the image. It decreases the rate of change in intensities and the sharpness of edges, making these features less apparent in the Fourier spectrum.



Figure 3-2 Result of the ideal low-pass filter, radius of 60

8. The denoised image has a less speckled appearance compared to the noisy image but lacks the clarity of the original image. The ideal low-pass filter preserves low frequencies in the image while blocking high frequencies, which are associated with edges and fine detail. This means that the sharpness of the edges and the resolution of fine details in the image have been reduced and blurred.
9. The artifact present in the restored image is called “ringing”. The artifact is caused by oscillations in output frequencies. This occurs because the impulse response of the ideal low-pass filter is the sinc function, which has a rippled shape. As a result, the image has a mottled appearance. At locations with rapid changes in intensity such as edges, the signal overshoots the transition and then oscillates outward like ripples in water.



Figure 3-3 Result of the ideal low-pass filter, radius of 20

10. The denoised image in Figure 3-3 is considerably blurrier than the image in Figure 3-2. The reduced radius of the ideal low-pass filter (i.e., 20 vs. 60) attenuates even more mid and high-level frequencies, which blurs the edges and fine details in the image. The PSNR reflects this, with Figure 3-3 having a lower value than Figure 3-2.
11. Noise, edges, and fine detail all correspond to high frequencies in the Fourier spectrum. As the radius of an ideal low-pass filter decreases, more high frequencies are removed from the image. With fewer high frequencies, an image becomes blurrier. Noise is reduced, but the trade-off is reduced sharpness of edges and details.



Figure 3-4 Result of Gaussian filter, standard deviation of 60

12. The denoised image resulting from the Gaussian filter has a better appearance than the images resulting from a low-pass filter. The PSNR value for the Gaussian filter is the highest of the three images. The Gaussian filter effectively removed the noise from the image without causing the ringing artifact since, unlike the ideal low-pass filter, its impulse response is non-oscillating. The Gaussian-filtered image is not as clear as the original image because it still attenuates higher frequencies corresponding to edges.

4 Filter Design

The following steps were taken to design and implement a spatial-frequency domain filter to remove the periodic noise from the original image shown on the left of Figure 4-1. MATLAB code is included in the Appendix under file lab3_prt4_filter_design.m.

Step 1: Fourier transform

First, the log Fourier spectrum of the original image was generated and shifted to the center (right of Figure 4-1). In the spectrum, four bright spots are visible, which are the frequencies causing the periodic noise in the original image.

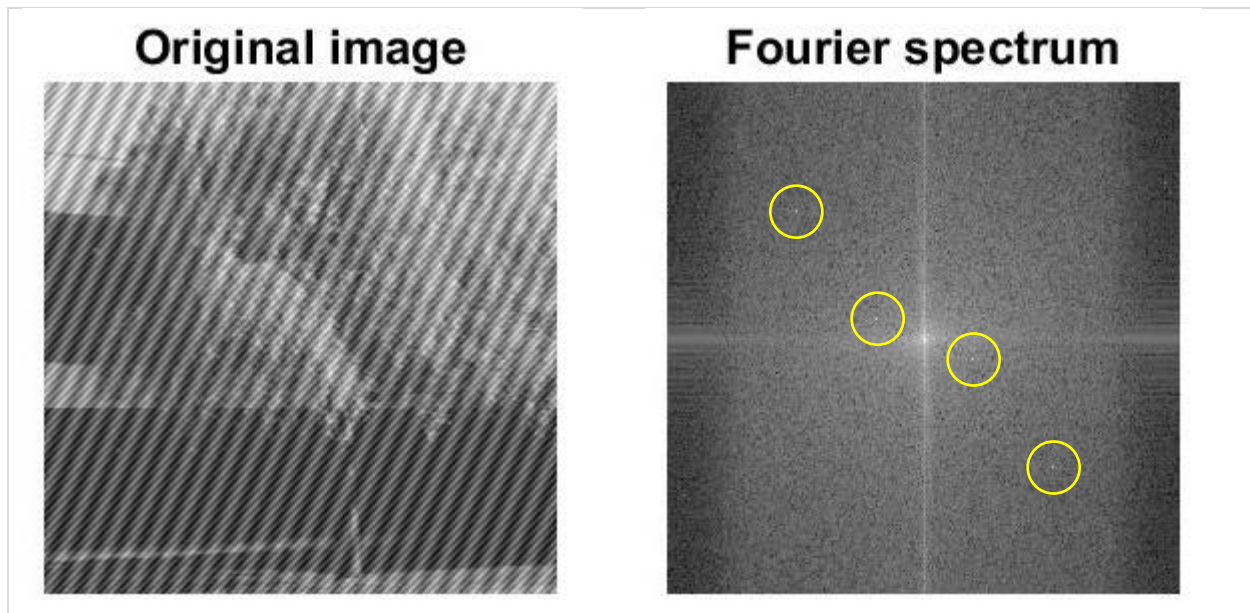


Figure 4-1 Original image (left) and its Fourier spectrum with bright peaks circled

Step 2: Notch filter

To remove the periodic noise, a notch filter was designed. The bright spots in the original Fourier spectrum were isolated. This was done by finding the maximum value in the image array, determining a threshold value based on the maximum value via trial and error, and then comparing all array values to this threshold. Values greater than the threshold were set to 1 and values less than the threshold were set to 0. The result can be seen in Figure 4-2.

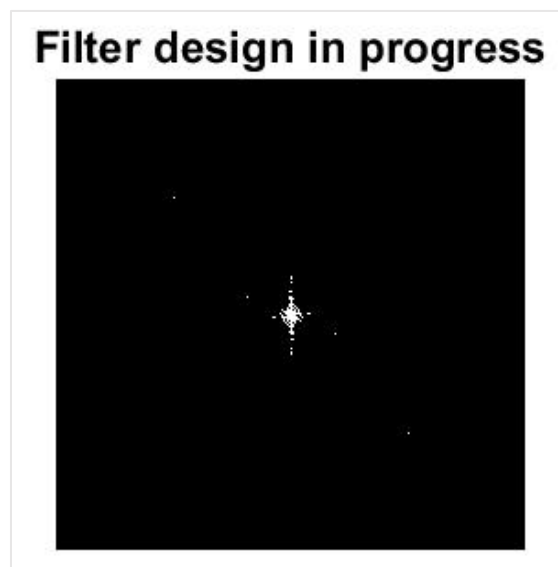


Figure 4-2 Step 2 of filter design in progress

Next the brightness around the origin of the spectrum was removed, as shown in the left of Figure 4-3. The filter was then inverted, swapping values of 0 for 1 and vice versa. The final notch filter is shown in the right of Figure 4-3 (outline added for visibility).

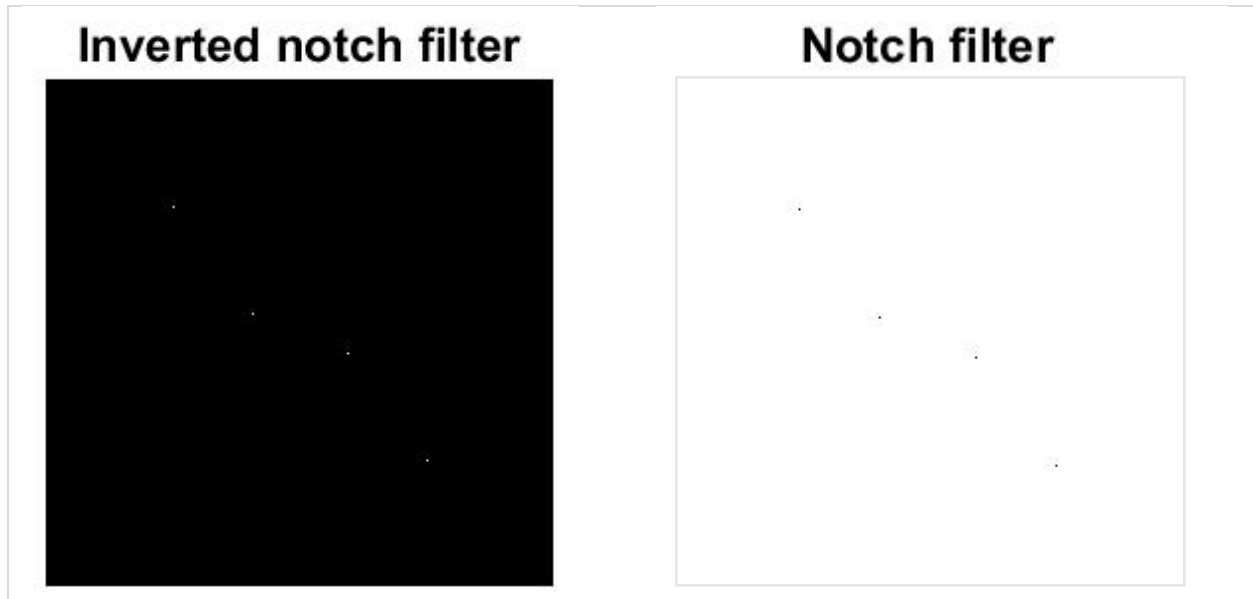


Figure 4-3 Brightness at origin removed (left) and the final notch filter (right)

Step 3: Applying the notch filter

The notch filter was multiplied with the Fourier spectrum of the original image to remove the bright spots. Figure 4-4 shows the resulting spectrum with the original bright spots no longer visible.

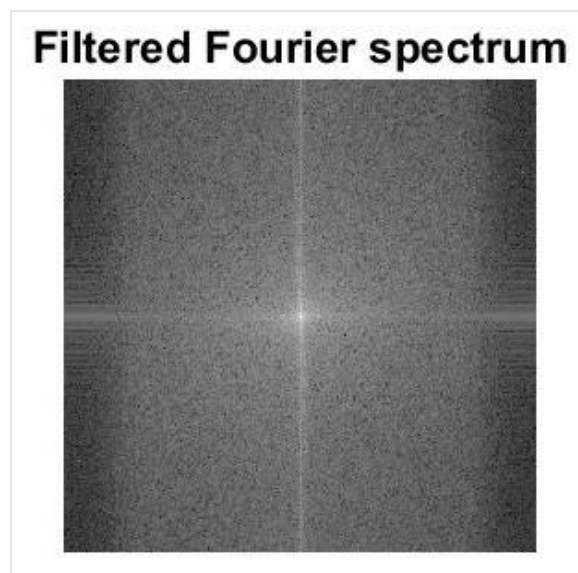


Figure 4-4 The filtered Fourier spectrum

Step 3: Inverse Fourier transform

Finally, the filtered spectrum was reverse-shifted, and the inverse Fourier transform of the filtered image was generated. Figure 4-5 shows the filtered image with the periodic noise removed.



Figure 4-5 Filtered image with periodic noise removed

5 Conclusions

This lab provided hands-on experience with the spatial-frequency domain, including Fourier spectrum analysis, low-pass and notch filters, and practice using image processing tools in MATLAB.

From this lab, it is evident that the Fourier spectrum communicates characteristics about its spatial image. Brightness at the centre of the Fourier spectrum corresponds to the average intensity of the image. Energy in the Fourier spectrum corresponds to the rate of change in intensity in the spatial image. Low frequencies correspond to slow rates of change, while high frequencies correspond to fast rates of change, such as edges, fine details, and noise. Edges in the original image appear as lines in the Fourier spectrum with the same angles.

Both phase and amplitude information are contained in the Fourier transform. Amplitude contributes intensity information while phase contributes shape characteristics to the image. When reconstructing an image, the contribution of phase is more significant to producing a recognizable image.

Periodic noise in the spatial domain can be corrected by designing and applying filters in the spatial-frequency domain. The frequencies associated with the periodic noise appear as bright spots in the Fourier spectrum. By cancelling out the bright spots in the spatial-frequency domain, noise is removed in the spatial domain.

Appendix – MATLAB Code

File: lab3_prt2_fourier_analysis.m

```
clear all; close all; clc;
fontSize = 12;

% Generate test image
f = zeros(256, 256);
f(:,108:148) = 1;

% Generate the Fourier spectra of the test image
X = fft2(f);
X_shift = fftshift(X);
X_log = log(1+abs(X_shift));

% Plot images
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(1, 3, 1);
imshow(f);
title('Test image', 'FontSize', fontSize);
subplot(1, 3, 2);
imshow(abs(X_shift), []);
title('Test image Fourier spectra', 'FontSize', fontSize);
subplot(1, 3, 3);
imshow(abs(X_log), []);
title('Test image Fourier spectra, log', 'FontSize', fontSize);

% Rotate the image and plot the Fourier spectra
f_rotate = imrotate(f, 45);
Y = fft2(f_rotate);
Y_shift = fftshift(Y);
Y_log = log(1+abs(Y_shift));

% Plot rotated images
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(1, 3, 1);
imshow(f_rotate);
title('Rotated test image', 'FontSize', fontSize);
subplot(1, 3, 2);
imshow(abs(Y_shift), []);
title('Rotated image Fourier spectra', 'FontSize', fontSize);
subplot(1, 3, 3);
imshow(abs(Y_log), []);
title('Rotated image Fourier spectra, log', 'FontSize', fontSize);

% Amplitude and phase analysis
lena = imread('lena.tiff');
lena_gray = convert2grayscale(lena);
lena_fft = fft2(lena_gray);
lena_amplitude = abs(lena_fft);
lena_phase = lena_fft./lena_amplitude;
lena_amplitude_inv = log(ifft2(lena_amplitude));
lena_phase_inv = ifft2(lena_phase);

% Plot images
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(1, 3, 1);
imshow(lena_gray);
title('Original gray image', 'FontSize', fontSize);
```

```

subplot(1, 3, 2);
imshow(lena_amplitude_inv, []);
title('Inverse Fourier transform of amplitude', 'FontSize', fontSize);
subplot(1, 3, 3);
imshow(lena_phase_inv, []);
title('Inverse Fourier transform of phase', 'FontSize', fontSize);

```

File: lab3_prt3_noise_reduction.m

```

clear all; close all; clc;
fontSize = 12;

% Load images, convert to grayscale, and normalize intensities
lena = imread('lena.tiff');
lena_gray = convert2grayscale(lena);
f1 = double(lena_gray)/255;

% Add noise to lena_gray_norm
f2 = imnoise(f1, 'gaussian', 0, 0.005);
psnr1 = PSNR_norm(f1, f2);

% Generate the Fourier spectra of the images
F1 = fft2(f1);
F1_sh = fftshift(F1);
F1_sh_log = log(1+abs(F1_sh));

F2 = fft2(f2);
F2_sh = fftshift(F2);
F2_sh_log = log(1+abs(F2_sh));

% Plot images
figure;
set(gcf, 'Position', get(0, 'Screensize'));
subplot(2, 2, 1);
imshow(f1);
title('Original gray image', 'FontSize', fontSize);
subplot(2, 2, 3);
imshow(f2, []);
title('Image with Gaussian noise', 'FontSize', fontSize);
subplot(2, 2, 2);
imshow(F1_sh_log, []);
title('Original image Fourier spectra', 'FontSize', fontSize);
subplot(2, 2, 4);
imshow(F2_sh_log, []);
title('Noisy image Fourier spectra', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f', psnr1));
annotation('textbox', [0.25, 0, 0.1, 0.1], 'string', txt);

% LOW-PASS FILTER
% Create ideal low-pass filter with r = 60
r = 60;
[M N] = size(f1);
h60 = fspecial('disk', r); h60(h60 > 0) = 1;
H60 = zeros(M, N);
H60(M/2-r:M/2+r, N/2-r:N/2+r) = h60;

% Apply filter to image and plot
G60_sh = F2_sh.*H60;
G60 = ifftshift(G60_sh);
g60 = real(ifft2(G60));

```

```

psnr2 = PSNR_norm(f1,g60);

figure;
imshow(g60);
title('Filtered image, r = 60', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr2));
annotation('textbox', [0.4, 0, 0.1, 0.1], 'string', txt);

% Create ideal low-pass filter with r = 20
r = 20;
h20 = fspecial('disk',r); h20(h20 > 0) = 1;
H20 = zeros(M,N);
H20(M/2-r:M/2+r,N/2-r:N/2+r) = h20;

% Apply filter to image and plot
G20_sh = F2_sh.*H20;
G20 = ifftshift(G20_sh);
g20 = real(ifft2(G20));
psnr2 = PSNR_norm(f1,g20);

figure;
imshow(g20);
title('Filtered image, r = 20', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr2));
annotation('textbox', [0.4, 0, 0.1, 0.1], 'string', txt);

% GAUSSIAN FILTER
% Create Gaussian filter
H_gauss = double(fspecial('gaussian', [M N], 60))
H = H_gauss./max(H_gauss,[],'all')
plotFilter(H, 'Gaussian low-pass filter');

% Apply filter to image and plot
G_sh = F2_sh.*H;
G = ifftshift(G_sh);
g = real(ifft2(G));
psnr3 = PSNR_norm(f1,g);

figure;
imshow(g);
title('Filtered image, Gaussian', 'FontSize', fontSize);
txt = strcat('PSNR = ', sprintf('%.2f',psnr3));
annotation('textbox', [0.4, 0, 0.1, 0.1], 'string', txt);

```

File: lab3_prt4_filter_design.m

```

clear all; close all; clc;
fontSize = 12;

% Load image
f = imread('frequnoisy.tif');

% Generate the Fourier spectrum of the original image
F = fft2(f);
F_sh = fftshift(F);
F_sh_log = log(abs(F_sh));

% Design notch filter to cancel out bright peaks in the Fourier spectrum
maxVal = max(F_sh_log,[],'all');
threshold = maxVal - 5;
H = F_sh_log >= threshold;
figure(1);

```

```

imshow(H, []); title('Filter design in progress', 'FontSize', fontSize);
H(100:150, 110:150) = 0;
J = imcomplement(H);

% Apply filter to original image
G_sh = F_sh.*J;
G = ifftshift(G_sh);
g = real(ifft2(G));

% Plot images and spectra
figure(2);
imshow(f); title('Original image', 'FontSize', fontSize);
figure(3);
imshow(F_sh_log, []); title('Fourier spectrum', 'FontSize', fontSize);
figure(4);
imshow(H, []); title('Inverted notch filter', 'FontSize', fontSize);
figure(5);
imshow(J, []); title('Notch filter', 'FontSize', fontSize);
figure(6);
imshow(log(abs(G_sh)), []); title('Filtered Fourier spectrum',...
    'FontSize', fontSize);
figure(7);
imshow(g, []); title('Filtered image', 'FontSize', fontSize);

```

File: PSNR_norm.m

```

function [PSNR_out] = PSNR_norm(f,g)
    PSNR_out = 10*log10(1/mean2((f-g).^2));
end

```

File: plotFilter.m

```

function [] = plotFilter(filter, type)
    fontSize = 12;
    figure;
    colormap(gray);
    imagesc(filter);
    title(type, 'FontSize', fontSize);
end

```