Image Processing and Computer Graphics

INTE 41312

Image Processing in Frequency Domain

IM/2019/112-Sandushi Weraduwa

01. Displaying the Fourier spectrum using MATLAB.

a. Provide the complete MATLAB code.

Editor - C:\Users\Sandushi\Documents\MATLAB\fourier_spectrum.m

fourier_spectrum.m

```matlab
% Clear the workspace
clear;
clc;
close all;

% Step 1: Create an image with a white rectangle and black background
f = zeros(30, 30);
f(5:24, 13:17) = 1;

% Display the original image
figure;
subplot(2,2,1);
imshow(f, 'InitialMagnification', 'fit');
title('Original Image');

% Step 2: Calculate the Discrete Fourier Transform (DFT) with zero padding
F = fft2(f, 256, 256);
F2 = abs(F);

% Display the magnitude of the Fourier transform without shift
subplot(2,2,2);
imshow(F2, [], 'InitialMagnification', 'fit');
title('Magnitude of DFT');

% Step 3: Shift zero-frequency components to the center of the array
Fshifted = fftshift(F);
F3 = abs(Fshifted);

% Display the shifted Fourier spectrum
subplot(2,2,3);
imshow(F3, []);
title('Centered Fourier Spectrum');

% Step 4: Logarithmic scaling to enhance display
F4 = log(1 + F3);

% Display the logarithmically scaled Fourier spectrum
subplot(2,2,4);
imshow(F4, []);
title('Log-Scaled Fourier Spectrum');

% Adding a figure title
sgtitle('Figure 1: Fourier Transform Steps');

% Save the figure as a JPEG image
saveas(gcf, 'C:\Users\Sandushi\Documents\4th year\4th Year 1st Sem\Image Processing\Figure1.jpeg');
```

b. Sub-plot the output image of all the four (04) steps with titles (Figure Legend: Figure 1).

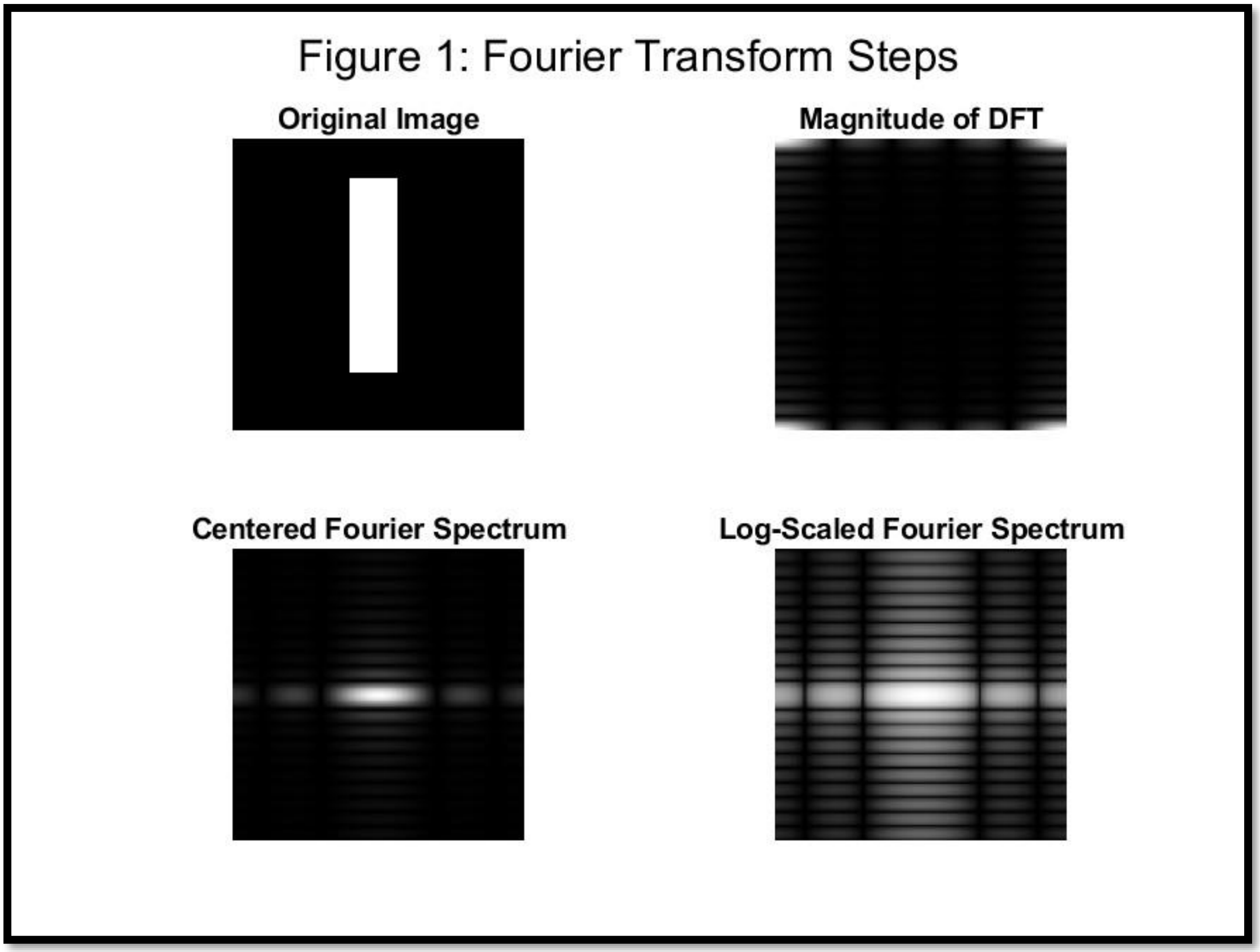

Figure 1

02. Low pass filtering in frequency domain.

a. Sub-plot the output of each step.

Editor - C:\Users\Sandushi\Documents\MATLAB\low_pass_filtering.m

fourier_spectrum.m | low_pass_filtering.m

```matlab

clear;
I = imread('tire.tif');

% Calculate the DFT of the original image
I_dft = fft2(I);
I_dft_shift = fftshift(I_dft);  % Shift the zero-frequency components to the center

% Get the size of the original image
[m, n] = size(I);

% Create a meshgrid to define the filter
[x, y] = meshgrid(1:n, 1:m);
z = sqrt((x - n/2).^2 + (y - m/2).^2);

% Define the low-pass filter
h = z < 50;  % Circular low-pass filter with a radius of 50

% Apply the low-pass filter
filtered_dft = h .* I_dft_shift;

% Perform the Inverse DFT
NEW = ifft2(ifftshift(filtered_dft));
j = uint8(abs(NEW));  % Convert the real part of the inverse DFT to uint8

% Display the results using subplots
figure;
subplot(2,3,1);
imshow(I);
title('Original Image');

subplot(2,3,2);
imshow(log(abs(I_dft_shift)+1), []);
title('Centered DFT');

subplot(2,3,3);
imagesc(h);
axis image;  % Ensure the axes are scaled equally
colormap(gray);  % Use a grayscale colormap
title('Low-Pass Filter');

subplot(2,3,4);
imshow(log(abs(filtered_dft)+1), []);
title('Filtered DFT');

subplot(2,3,5);
imshow(abs(NEW), []);
title('Inverse DFT (Magnitude)');

subplot(2,3,6);
imshow(j);
title('Final Image');

sgtitle('Steps of Low-Pass Filtering');
saveas(gcf, 'C:\Users\Sandushi\Documents\4th year\4th Year 1st Sem\Image Processing\Figure2.jpeg');
```

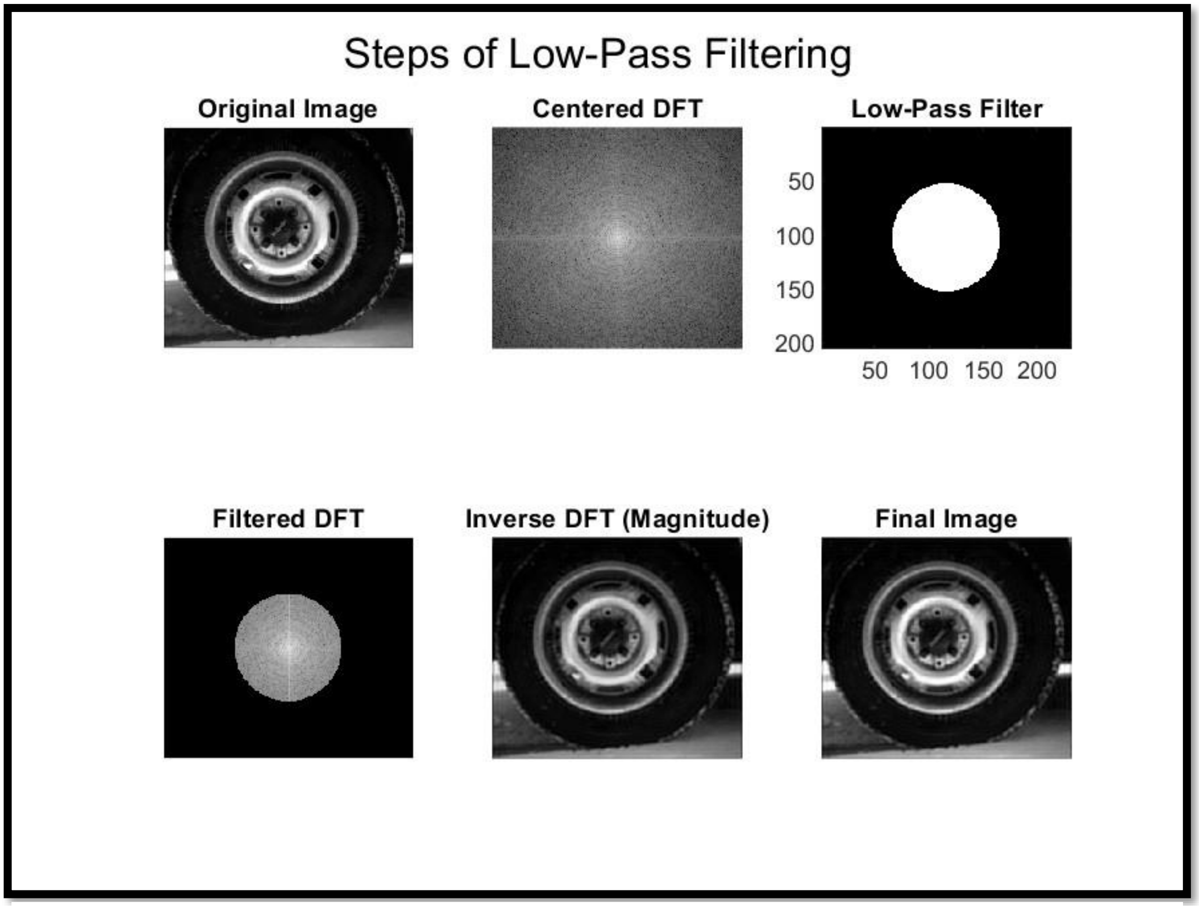

Figure 2

b. Briefly explain how "h" affects the filter.

- 'h' is defined as a circular binary mask where its value is 1 (true) for all locations within a radius of 50 units from the center of the frequency spectrum and 0 (false) otherwise.
- This mask is used to perform a low pass filtering operation. In the frequency domain, applying h as a multiplier to the shifted DFT of the image effectively retains the low-frequency components while eliminating high-frequency components. Low frequencies in images represent the broad, general features without much detail, while high frequencies represent edges and detailed features.
- By applying this filter, the resultant image emphasizes the larger, smoother variations in intensity, effectively blurring and reducing details and noise. This is particularly useful

for tasks such as noise reduction, image smoothing, or preparing an image for further analysis where fine details may not be necessary.

03. Implement high-pass filter in frequency domain.

a. Provide the complete code.

Editor - C:\Users\Sandushi\Documents\MATLAB\high_pass_filtering.m

fourier_spectrum.m | low_pass_filtering.m | high_pass_filtering.m

```
clear;
I = imread('tire.tif');

% Calculate the DFT
I_fft = fft2(double(I));

% Rearrange the Fourier transform by shifting the zero-frequency component to the center of the array
I_shifted = fftshift(I_fft);

% Get the size of the original image
[m, n] = size(I);

% High-pass filter
[x, y] = meshgrid(1:n, 1:m);
z = sqrt((x - n/2).^2 + (y - m/2).^2);
h = z > 50;  % Change here to create a high-pass filter

% Apply the filter
filtered_I = h .* I_shifted;

% Inverse DFT
I_new = ifftshift(filtered_I);
I_new = ifft2(I_new);

% Display the image
I_final = uint8(abs(I_new));

% Subplots of the outputs
figure;

% Original Image
subplot(2,3,1);
imshow(I);
title('Original Image');

% Frequency Spectrum (Magnitude)
subplot(2,3,2);
imagesc(log(abs(I_shifted) + 1));
colormap('gray');
title('Centered Spectrum');

% High-Pass Filter (Mask)
subplot(2,3,3);
imagesc(h);
colormap('gray');
title('High-Pass Filter h');

% Filtered Spectrum
subplot(2,3,4);
imagesc(log(abs(filtered_I) + 1));
colormap('gray');
title('Filtered Spectrum');

% Inverse FFT Image (Before Converting to uint8)
subplot(2,3,5);
imshow(abs(I_new), []);
title('Inverse FFT (abs)');

% Final Image
subplot(2,3,6);
imshow(I_final);
title('Final Image (uint8)');
saveas(gcf, 'C:\Users\Sandushi\Documents\4th year\4th Year 1st Sem\Image Processing\Figure3.jpeg');
```

b. Sub-plot the output of each step with titles.

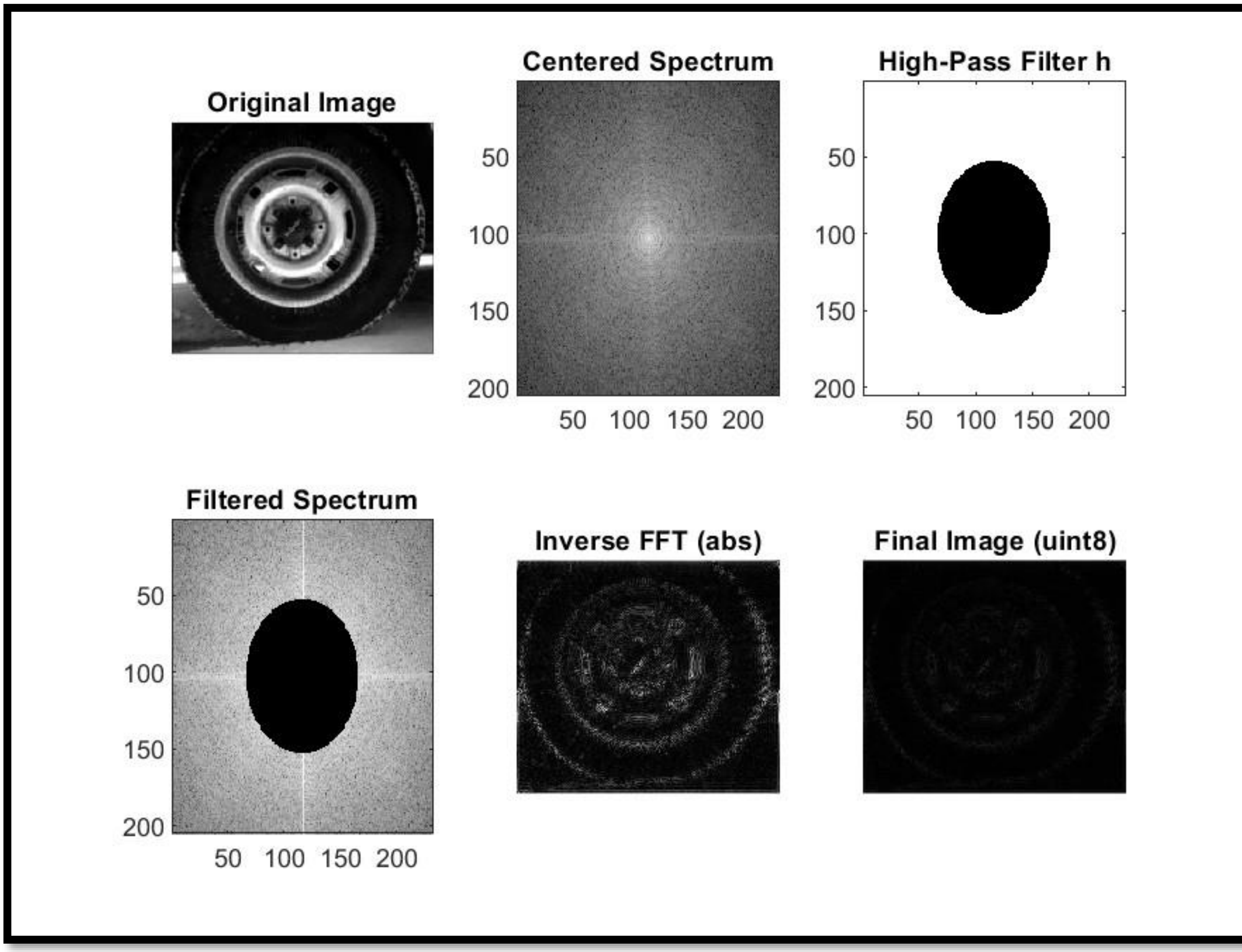

Figure 3

c. Compare the output image of Q.2. and Q.3.

To compare the high-pass filtered image with the low-pass filtered image:

Low-pass Filter: Enhances the smooth areas and suppresses the high-frequency components (edges, noise, and fine details).

High-pass Filter: Enhances edges and fine details by suppressing low-frequency components.

When comparing images, we can see that the low-pass filter image appears smoother, while the high-pass filter image highlights edges and textures. This can be visually confirmed byinspecting the images side by side, or by subtracting one image from the other to see the difference directly.

04.

a. What is image restoration, and how does it differ from image enhancement?

Image Restoration is the process of reconstructing or recovering an image that has been degraded by known distortions such as blurring, noise, and loss of data. The objective is to restore the true image as closely as possible using priori knowledge of the degradation process.

Image Enhancement, on the other hand, is aimed at improving the visual appearance of an image or making it more suitable for a specific task without consideration of the fidelity to the original image. Enhancement techniques are often subjective and are not designed to be inverted.

b. Briefly explain how working in the frequency domain is useful for image restoration.

Working in the Frequency Domain for Image Restoration: Working in the frequency domain is beneficial for image restoration for several reasons:

Decomposition: It allows the image to be decomposed into its frequency components, making it easier to identify and isolate the effects of degradation.

Filter Design: Certain types of degradation, such as blurring and noise, often affect specific frequency components more than others. In the frequency domain, appropriate filters can be designed to target and mitigate these effects more effectively.

Efficiency: Algorithms like FFT (Fast Fourier Transform) make it computationally efficient to transform to and from the frequency domain.

c. What types of filters are commonly used in frequency domain image restoration?

Wiener Filter: An optimal filter for minimizing the mean square error in the presence of additive noise.

Inverse Filter: Used for deblurring when the degradation function is known and there is no noise.

Bandpass and Bandstop Filters: Used to selectively restore or enhance certain frequency ranges.

d. How does the Wiener filter optimize image restoration, and in what scenarios is it most effective?

d. Wiener Filter Optimization:

The Wiener filter is designed to minimize the overall mean square error between the estimated and the true image. It takes into account the degradation function and the statistical characteristics of both the noise and the original image. It is most effective in scenarios where:

- The noise is additive, and its statistical characteristics (like power spectral density) are known or can be estimated.
- The degradation function (often a blurring function) is known or can be accurately modeled.

Implement Weiner Filter in MATLAB to filter Gaussian noise in frequency domain.

Note:

Use an image of your choice.

Add Gaussian noise to the original image.

e. Provide the complete MATLAB function.

Editor - C:\Users\Sandushi\Documents\MATLAB\image_restoration.m

fourier_spectrum.m | low_pass_filtering.m | high_pass_filtering.m | image_restoration.m

```matlab
function image_restoration()
    % Load an image (Convert it to grayscale if it's RGB)
    originalImg = imread("C:\Users\Sandushi\Documents\4th year\4th Year 1st Sem\Image Processing\girl.jpg");

    % Display the original image
    figure;
    subplot(1,3,1);
    imshow(originalImg);
    title('Original Image');

    if size(originalImg, 3) == 3
        originalImg = rgb2gray(originalImg);
    end


    % Add Gaussian noise to the original image
    noisyImg = imnoise(originalImg, 'gaussian', 0, 0.01);  % Adjust the variance as needed
    subplot(1,3,2);
    imshow(noisyImg);
    title('Noisy Image');

    % Apply the Wiener Filter
    restoredImg = wiener2(noisyImg, [5 5]);  % The neighborhood size can be adjusted
    subplot(1,3,3);
    imshow(restoredImg);
    title('Restored Image');

    saveas(gcf, 'C:\Users\Sandushi\Documents\4th year\4th Year 1st Sem\Image Processing\Figure4.jpeg');
end
```

f. Sub-plot the output of each step with titles.

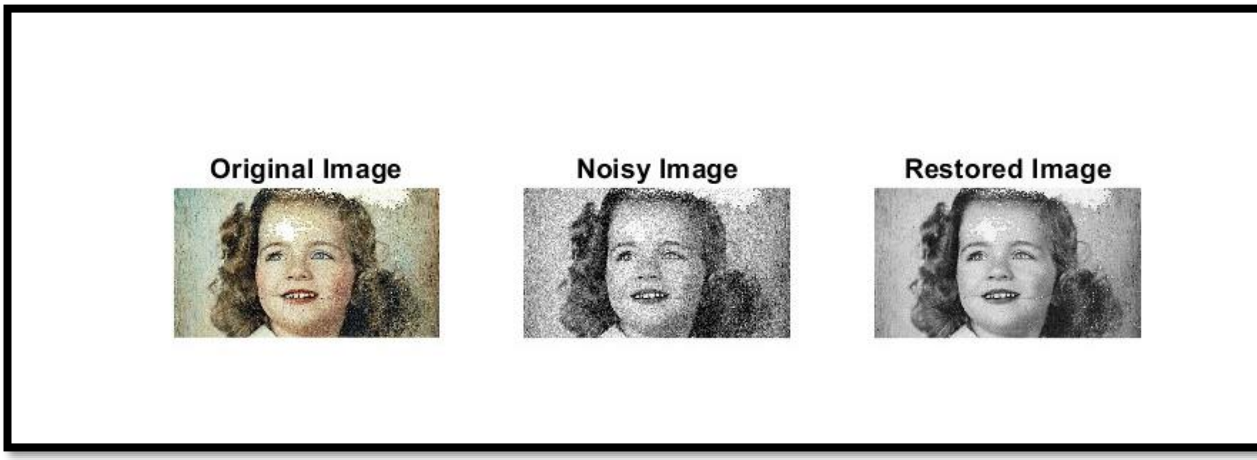

Figure 4