Digital Image Processing Laboratory

Experiment Report

Experiment Title Fourier Transform and Filtering in the Frequency Domain

Student's Name _____

Student's ID _____

Class _____

Date handed in _____

International school, Jinan University

A. Objectives

(1) To know how to use Discrete Fourier Transform (DFT).
(2) To know how to process image in frequency domain by high-pass filter or low-pass filter in MATLAB.

B. Technique

In this project, the image **bag.png** will be used.
(1) Apply Fourier transform and inverse Fourier transform to images.
(2) Apply a low-pass filter to the image.
(3) Apply a high-pass filter to the image.

C. Experiment Content

(1) Read the image **bag.png** (or your **selfie**, if it is color photo, please convert it into gray) given in the folder, show it, perform Fourier transform on it, show the **magnitude** image and **phase angle** image in frequency domain. Then perform inverse Fourier transform, show image, compare with the original one to see if they are identity.

1. Read the image, convert it into gray and display it.
2. Perform Fourier transform, compute the magnitude and phase angle images.

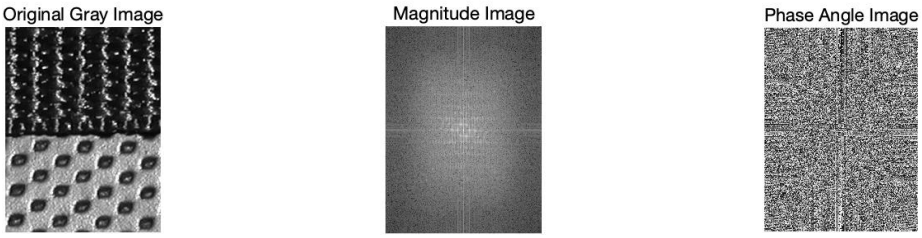

Figure 1: Images of the original image and its magnitude and phase angle

The frequency magnitude image exhibits a central bright spot, which means that the low-frequency information of the image is concentrated in the center of the spectrum.

Since the amplitude spectrum has been logarithmically transformed (log(1+abs(F))), it can more clearly show the details of high-frequency and low-frequency components.

Phase image looks chaotic, like random noise, but it contains important information about the structure of the image. Phase information defines the relative position of each frequency component in the spectrum. If the phase information is subjected to an inverse Fourier transform, the structure of the image can be reconstructed to some extent even without amplitude information.

3. Perform inverse Fourier transform and display the reconstructed image.
4. Compare the reconstructed image with the original one to see if they are identity.

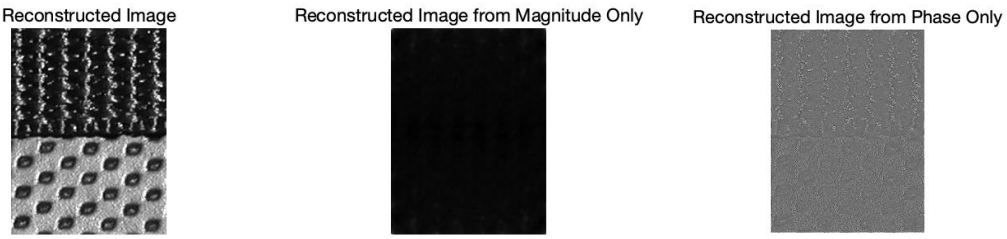

Figure 2: Reconstructed images using different information

Images reconstructed using only frequency magnitude information appear almost dark, indicating that without phase information, the structure and features of the image are nearly unrecognizable. Since the frequency magnitude information only tells us the brightness distribution of the image without structural information, it is impossible to reconstruct any meaningful image content.

Although the image reconstructed using only phase information is very different from the original image in brightness and contrast, we can identify certain structures in the image. This shows that phase information indeed carries the main features and structural information of the image. Although the details may be incomplete, the basic outline of the image is discernible.

The reconstructed image using frequency magnitude and phase information looks very similar to the original grayscale image, indicating that the frequency magnitude and phase information combined can successfully reconstruct the image. Structure, light-dark relationships, and details are all well restored, demonstrating the importance of frequency magnitude and phase information in the Fourier transform.

The code to complete this experiment is as follows:

```
clc;

clear;



% Read the image

img = imread('bag.png');



% Convert to grayscale

if size(img, 3) == 3

    img_gray = rgb2gray(img);

else
```

```
    img_gray = img;
end


% Display the original grayscale image
figure, imshow(img_gray), title('Original Gray Image');


% Perform Fourier transform
F = fft2(double(img_gray));
F_shifted = fftshift(F);
F_magnitude = log(abs(F_shifted) + 1); % Calculate the frequency magnitude
F_phase = angle(F_shifted); % Calculate the phase angle


% Display the magnitude image
figure, imshow(F_magnitude, []),
title('Magnitude Image');


% Display the phase angle image
figure, imshow(F_phase, []),
title('Phase Angle Image');


% Reconstructed Image from Phase Only(Set magnitude as 1)
F_phase_only = 1 .* exp(1i * angle(F_shifted));
img_phase_only = real(ifft2(ifftshift(F_phase_only)));
figure, imshow(img_phase_only, []),
title('Reconstructed Image from Phase Only');


% Reconstructed Image from Magnitude Only(Set phase angle as 0)
F_magnitude_only = abs(F_shifted) .* exp(1i * zeros(size(angle(F_shifted))));
```

```
img_magnitude_only = real(ifft2(ifftshift(F_magnitude_only)));

figure, imshow(img_magnitude_only, []),

title('Reconstructed Image from Magnitude Only');


% Perform inverse Fourier transform

F_inverse = ifftshift(abs(F_shifted) .* exp(1i * angle(F_shifted)));

img_reconstructed = real(ifft2(F_inverse));

figure, imshow(img_reconstructed, []),

title('Reconstructed Image');
```

(2) Apply several Low-pass filters on your **selfie** (as above, keep it as a gray image). The low-pass filters include ideal low-pass filter (ILPF), Butterworth low-pass filter (BLPF) and Gaussian low-pass filter (GLPF). Observe the differences between the original image and filtered image under different cut-off frequencies when the same type of LPF has been performed.

1. Read the image, convert it into grey.
2. Fourier transform the image and shift it.
3. Apply several low-pass filters on image(ILPF, BLPF and GLPF) under different cut-off frequencies.
4. Display the comparison among the origial image and the modified images.

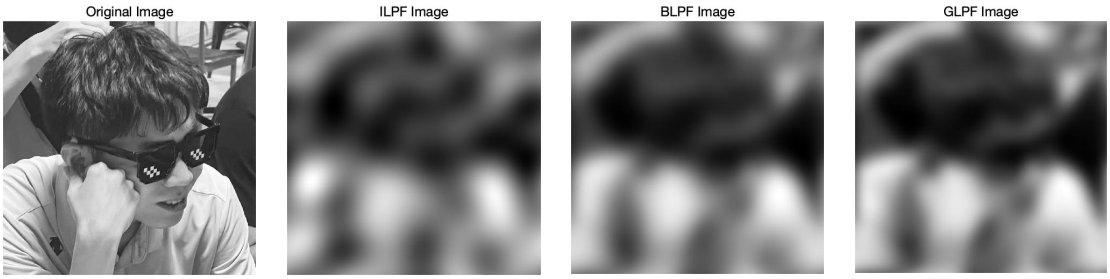

Figure 3: Comparison when cut-off frequency = 5

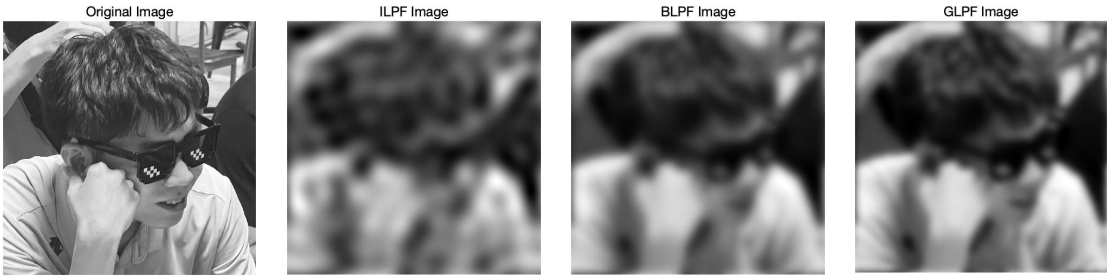

Figure 4: Comparison when cut-off frequency = 10

Figure 5: Comparison when cut-off frequency = 50

The above images represent filtered results using Butterworth (BLPF), Gaussian (GLPF), and Ideal (ILPF) low-pass filters at cut-off frequencies of 5, 10, and 50.

The filters with a cut-off frequency of 5 show a very blurred image where only the most general shape of the subject can be discerned. When the filters with a cut-off frequency of 10, the blurring is reduced, and more details begin to emerge, though the image is still quite soft. At a cut-off frequency of 50, the image has much more detail, and the blurring effect is much less pronounced, resulting in a clearer image that resembles the original more closely.

Ideal Filter (ILPF) has a sharp cut-off, which can introduce ringing artifacts near the edges of the frequency boundaries, making it less ideal for applications where image quality is a concern.

Butterworth Filter (BLPF) provides a smoother transition between passband and stopband, resulting in less ringing and fewer artifacts than ILPF, but still retains some blurring at lower cut-off frequencies.

Gaussian Filter (GLPF) produces a smoother effect than the ILPF and BLPF due to its nature; transitions are very smooth, which prevents ringing but can result in a loss of some detail even at higher cut-off frequencies.

The code to complete this experiment is as follows:

```
clc;

clear;



% Read the image

img = imread('selfie.jpg');



% Convert to grayscale
```

```
if size(img, 3) == 3
    img_gray = rgb2gray(img);
else
    img_gray = img;
end

% Perform Fourier transform
F = fft2(double(img_gray));
F_shifted = fftshift(F);

% Define the size of the image
[M, N] = size(F_shifted);

% Initialize low-pass filter masks
ILPF_mask = zeros(M, N);
BLPF_mask = zeros(M, N);
GLPF_mask = zeros(M, N);

% Create a grid of distances from the center of the frequency rectangle
[u, v] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));
D = sqrt(u.^2 + v.^2);

% Set the cut-off frequencies
cutoff_frequencies = [5, 10, 50];

for D0 = cutoff_frequencies
    % Create the Ideal Low-Pass Filter (ILPF) mask
    ILPF_mask(D <= D0) = 1;
```

```
    % Create the Butterworth Low-Pass Filter (BLPF) mask
    n = 2; % Order of the filter
    BLPF_mask = 1 ./ (1 + (D./D0).^(2*n));

    % Create the Gaussian Low-Pass Filter (GLPF) mask
    GLPF_mask = exp(-(D.^2)./(2*(D0^2)));

    % Apply the filters by multiplying in the frequency domain
    ILPF_result = F_shifted .* ILPF_mask;
    BLPF_result = F_shifted .* BLPF_mask;
    GLPF_result = F_shifted .* GLPF_mask;

    % Perform inverse Fourier transform to get the filtered images
    img_ILPF = real(ifft2(ifftshift(ILPF_result)));
    img_BLPF = real(ifft2(ifftshift(BLPF_result)));
    img_GLPF = real(ifft2(ifftshift(GLPF_result)));

    % Display the filtered images
    figure;
    subplot(1, 4, 1), imshow(img_gray), title('Original Image');
    subplot(1, 4, 2), imshow(img_ILPF, []), title('ILPF Image');
    subplot(1, 4, 3), imshow(img_BLPF, []), title('BLPF Image');
    subplot(1, 4, 4), imshow(img_GLPF, []), title('GLPF Image');

    % print out the cut-off frequency of present loop
    fprintf('The cut-off frequency is %d.\n', D0);
end
```

(3) Apply ideal high-pass filter (IHPF), Butterworth high-pass filter (BHPF) and Gaussian high-pass filter (GHPF) on your **selfie**, respectively. Observe the differences between the original image and filtered image under different cut-off frequencies when the same type of HPF has been performed.

1. Read the image, convert it into grey.
2. Fourier transform the image and shift it.
3. Apply several high-pass filters on image(IHPF, BHPF and GHPF) under different cut-off frequencies.
4. Display the comparison among the origial image and the modified images.

Figure 6: Comparison when cut-off frequency = 5

Figure 7: Comparison when cut-off frequency = 10

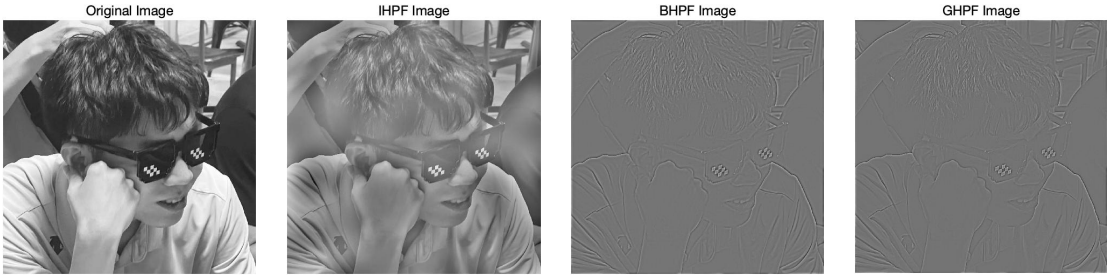

Figure 8: Comparison when cut-off frequency = 50

The above images demonstrate the effects of different high-pass filters on an image, enhancing edges and details by attenuating lower frequencies.

The Butterworth filter offers a balanced edge enhancement with smoother transitions, avoiding the abrupt changes that can lead to artifacts.

The Gaussian filter applies a softer emphasis on edges, resulting in a more natural look even at higher cut-off frequencies, which is ideal for avoiding unnatural 'ringing' effects.

The Ideal filter provides the sharpest edge enhancement, which becomes more pronounced as the cut-off frequency increases, but this sharpness comes with the potential downside of creating ringing artifacts.

As the cut-off frequency rises across all filters, the emphasis on finer details and texture becomes more pronounced, leading to a clearer distinction of the image's features.

The code to complete this experiment is as follows:

```
clc;

clear;



% Read the image

img = imread('selfie.jpg');



% Convert to grayscale

if size(img, 3) == 3

    img_gray = rgb2gray(img);

else

    img_gray = img;

end



% Perform Fourier transform

F = fft2(double(img_gray));

F_shifted = fftshift(F);



% Define the size of the image

[M, N] = size(F_shifted);
```

```
% Initialize high-pass filter masks
IHPF_mask = zeros(M, N);
BHPF_mask = zeros(M, N);
GHPF_mask = zeros(M, N);

% Create a grid of distances from the center of the frequency rectangle
[u, v] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));
D = sqrt(u.^2 + v.^2);

% Set the cut-off frequency
cutoff_frequencies = [5, 10, 50];

for D0 = cutoff_frequencies
    % Create the Ideal High-Pass Filter (IHPF) mask
    IHPF_mask(D > D0) = 1;

    % Create the Butterworth High-Pass Filter (BHPF) mask
    n = 2; % Order of the filter
    BHPF_mask = 1 - 1 ./ (1 + (D./D0).^(2*n));

    % Create the Gaussian High-Pass Filter (GHPF) mask
    GHPF_mask = 1 - exp(-(D.^2)./(2*(D0^2)));

    % Apply the high-pass filters
    IHPF_result = F_shifted .* IHPF_mask;
    BHPF_result = F_shifted .* BHPF_mask;
    GHPF_result = F_shifted .* GHPF_mask;
```

```
    % Perform inverse Fourier transform to obtain the filtered images
    img_IHPF = real(ifft2(ifftshift(IHPF_result)));
    img_BHPF = real(ifft2(ifftshift(BHPF_result)));
    img_GHPF = real(ifft2(ifftshift(GHPF_result)));

    % Display the filtered images
    figure;
    subplot(1, 4, 1), imshow(img_gray), title('Original Image');
    subplot(1, 4, 2), imshow(img_IHPF, []), title('IHPF Image');
    subplot(1, 4, 3), imshow(img_BHPF, []), title('BHPF Image');
    subplot(1, 4, 4), imshow(img_GHPF, []), title('GHPF Image');

    % print out the cut-off frequency of present loop
    fprintf('The cut-off frequency is %d.\n', D0);
end
```

D. Conclusions

This laboratory experiment has been instrumental in enhancing my comprehension of digital image processing and the Fourier transform's crucial role in the frequency domain analysis.

By applying Fourier transforms to various images, I experienced how image information is dissected into frequency components, which laid the groundwork for further manipulations with filters. Experimenting with different filters such as Ideal, Butterworth, and Gaussian enabled me to see how each filter uniquely affects an image, particularly in terms of edge enhancement and detail preservation.

The practical application of low-pass and high-pass filters deepened my understanding of the effects that various cut-off frequencies have on an image's visual characteristics.

I value this opportunity to translate mathematical foundations into tangible image processing skills using MATLAB, which I believe will be a significant asset in my future academic and professional endeavors.