

**Digital Image Processing (CSE-438)**

**Sec:03**

**Lab: 05 & 06**

**Submitted to:**

Md. Asif Khan Rifat

Lecturer  
Department of Computer Science & Engineering

East West University

**Submitted by:**

Prinom Mojumder

2021-2-60-098

1.

Apply Fourier transform to transform any image (above) from the spatial domain to the frequency domain. Apply inverse Fourier transform to transform the image from the frequency domain to the spatial domain.

Code:

img1 = imread('Picture1.jpg');

img2 = imread('Picture2.jpg');

img3 = imread('Picture3.jpg');

img1\_gray = rgb2gray(img1);

img2\_gray = rgb2gray(img2);

img3\_gray = rgb2gray(img3);

F1 = fft2(img1\_gray);

F2 = fft2(img2\_gray);

F3 = fft2(img3\_gray);

F1\_shifted = fftshift(F1);

F2\_shifted = fftshift(F2);

F3\_shifted = fftshift(F3);

I1 = ifft2(F1);

I2 = ifft2(F2);

I3 = ifft2(F3);

I1\_real = uint8(abs(I1));

I2\_real = uint8(abs(I2));

I3\_real = uint8(abs(I3));

figure;

subplot(3,3,1), imshow(img1\_gray), title('Image 1 - Original');

subplot(3,3,2), imshow(log(abs(F1\_shifted)),[]), title('Image 1 - Frequency Domain');

subplot(3,3,3), imshow(I1\_real), title('Image 1 - Restored');

subplot(3,3,4), imshow(img2\_gray), title('Image 2 - Original');

subplot(3,3,5), imshow(log(abs(F2\_shifted)),[]), title('Image 2 - Frequency Domain');

subplot(3,3,6), imshow(I2\_real), title('Image 2 - Restored');

subplot(3,3,7), imshow(img3\_gray), title('Image 3 - Original');

subplot(3,3,8), imshow(log(abs(F3\_shifted)),[]), title('Image 3 - Frequency Domain');

subplot(3,3,9), imshow(I3\_real), title('Image 3 - Restored');

Output:

A collage of images of a brain

AI-generated content may be incorrect.

2. Apply three types of high pass filtering in the frequency domain in **Picture 4**

andfind out which one is better to produce the enhanced image (sharpen).

1. Ideal high pass filter (IHPF)

Code:

img = imread('Picture4.jpg');

if size(img, 3) == 3

  img\_gray = rgb2gray(img);

else

  img\_gray = img;

end

F = fft2(double(img\_gray));

F\_shifted = fftshift(F);

[M, N] = size(img\_gray);

D0 = 30;

[x, y] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));

D = sqrt(x.^2 + y.^2);

IHPF = double(D > D0);

Filtered\_F = F\_shifted .\* IHPF;

F\_inverse\_shifted = ifftshift(Filtered\_F);

Processed\_img = real(ifft2(F\_inverse\_shifted));

figure;

subplot(2,3,1), imshow(img\_gray, []), title('Original Image');

subplot(2,3,2), imshow(log(abs(F\_shifted)+1), []), title('Centered Fourier Spectrum');

subplot(2,3,3), imshow(IHPF, []), title('Ideal High-Pass Filter');

subplot(2,3,4)

mesh(x, y, IHPF);

title('Ideal High-Pass Filter (3D)');

xlabel('X'), ylabel('Y'), zlabel('Amplitude');

subplot(2,3,5), imshow(log(abs(Filtered\_F)+1), []), title('Filtered Fourier Spectrum');

subplot(2,3,6), imshow(uint8(Processed\_img)), title('Processed Image');

Output:

A collage of images of a brain

AI-generated content may be incorrect.

1. Butterworth high pass filter (BHPF)

Code:

img = imread('Picture4.jpg');

if size(img, 3) == 3

   img\_gray = rgb2gray(img);

else

   img\_gray = img;

end

F = fft2(double(img\_gray));

F\_shifted = fftshift(F);

[M, N] = size(img\_gray);

D0 = 30;

n = 2;

[u, v] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));

D = sqrt(u.^2 + v.^2);

BHPF = 1 ./ (1 + (D0 ./ (D + eps)).^(2\*n));

Filtered\_F = F\_shifted .\* BHPF;

% Inverse Fourier Transform

F\_inverse\_shifted = ifftshift(Filtered\_F);

Processed\_img = real(ifft2(F\_inverse\_shifted));

figure;

subplot(2,3,1), imshow(img\_gray, []), title('Original Image');

subplot(2,3,2), imshow(log(abs(F\_shifted)+1), []), title('Centered Fourier Spectrum');

subplot(2,3,3), imshow(BHPF, []), title('Butterworth High-Pass Filter');

subplot(2,3,4)

mesh(u, v, BHPF);

title('Butterworth High-Pass Filter (3D)');

xlabel('X'), ylabel('Y'), zlabel('Amplitude');

subplot(2,3,5), imshow(log(abs(Filtered\_F)+1), []), title('Filtered Fourier Spectrum');

subplot(2,3,6), imshow(uint8(Processed\_img)), title('Processed Image');

Output:

A collage of images of a brain

AI-generated content may be incorrect.

iii. Gaussian high pass filter (GHPF)

Code:

img = imread('Picture4.jpg');

if size(img, 3) == 3

   img\_gray = rgb2gray(img);

else

   img\_gray = img;

end

F = fft2(double(img\_gray));

F\_shifted = fftshift(F);

[M, N] = size(img\_gray);

D0 = 30;

[u, v] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));

D = sqrt(u.^2 + v.^2);

GHPF = 1 - exp(-(D.^2) / (2 \* (D0^2)));

Filtered\_F = F\_shifted .\* GHPF;

F\_inverse\_shifted = ifftshift(Filtered\_F);

Processed\_img = real(ifft2(F\_inverse\_shifted));

figure;

subplot(2,3,1), imshow(img\_gray, []), title('Original Image');

subplot(2,3,2), imshow(log(abs(F\_shifted)+1), []), title('Centered Fourier Spectrum');

subplot(2,3,3), imshow(GHPF, []), title('Gaussian High-Pass Filter');

subplot(2,3,4)

mesh(u, v, GHPF);

title('Gaussian High-Pass Filter (3D)');

xlabel('X'), ylabel('Y'), zlabel('Amplitude');

subplot(2,3,5), imshow(log(abs(Filtered\_F)+1), []), title('Filtered Fourier Spectrum');

subplot(2,3,6), imshow(uint8(Processed\_img)), title('Processed Image');

Output:

A collage of images of a brain

AI-generated content may be incorrect.

3. Apply three types of low pass filtering in the frequency domain in **Picture 4** andfind out which one is better to produce the enhanced image (sharpen).

i. Ideal low pass filter (ILPF)

Code:

img = imread('Picture4.jpg');

if size(img, 3) == 3

  img\_gray = rgb2gray(img);

else

  img\_gray = img;

end

F = fft2(double(img\_gray));

F\_shifted = fftshift(F);

[M, N] = size(img\_gray);

D0 = 30;

[x, y] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));

D = sqrt(x.^2 + y.^2);

ILPF = double(D <= D0);

Filtered\_F = F\_shifted .\* ILPF;

F\_inverse\_shifted = ifftshift(Filtered\_F);

Processed\_img = real(ifft2(F\_inverse\_shifted));

figure;

subplot(2,3,1), imshow(img\_gray, []), title('Original Image');

subplot(2,3,2), imshow(log(abs(F\_shifted)+1), []), title('Centered Fourier Spectrum');

subplot(2,3,3), imshow(ILPF, []), title('Ideal Low-Pass Filter');

subplot(2,3,4)

mesh(x, y, ILPF);

title('Ideal Low-Pass Filter (3D)');

xlabel('X'), ylabel('Y'), zlabel('Amplitude');

subplot(2,3,5), imshow(log(abs(Filtered\_F)+1), []), title('Filtered Fourier Spectrum');

subplot(2,3,6), imshow(uint8(Processed\_img)), title('Processed Image');

Output:

A close-up of a brain scan

AI-generated content may be incorrect.

ii. Butterworth low pass filter (BLPF)

Code:

img = imread('Picture4.jpg');

if size(img, 3) == 3

  img\_gray = rgb2gray(img);

else

  img\_gray = img;

end

F = fft2(double(img\_gray));

F\_shifted = fftshift(F);

[M, N] = size(img\_gray);

D0 = 30; n = 2;

[u, v] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));

D = sqrt(u.^2 + v.^2);

BLPF = 1 ./ (1 + (D ./ D0).^(2\*n));

Filtered\_F = F\_shifted .\* BLPF;

F\_inverse\_shifted = ifftshift(Filtered\_F);

Processed\_img = real(ifft2(F\_inverse\_shifted));

figure;

subplot(2,3,1), imshow(img\_gray, []), title('Original Image');

subplot(2,3,2), imshow(log(abs(F\_shifted)+1), []), title('Centered Fourier Spectrum');

subplot(2,3,3), imshow(BLPF, []), title('Butterworth Low-Pass Filter');

subplot(2,3,4)

mesh(u, v, BLPF, 'EdgeColor', [0.5 0 0.5], 'FaceColor', 'interp');

title('Butterworth Low-Pass Filter (3D)');

xlabel('X'), ylabel('Y'), zlabel('Amplitude');

subplot(2,3,5), imshow(log(abs(Filtered\_F)+1), []), title('Filtered Fourier Spectrum');

subplot(2,3,6), imshow(uint8(Processed\_img)), title('Processed Image');

Output:

A close-up of a brain

AI-generated content may be incorrect.

iii. Gaussian low pass filter (GLPF)

Code:

img = imread('Picture4.jpg');

if size(img, 3) == 3

  img\_gray = rgb2gray(img);

else

  img\_gray = img;

end

F = fft2(double(img\_gray));

F\_shifted = fftshift(F);

[M, N] = size(img\_gray);

D0 = 30;

[u, v] = meshgrid(-floor(N/2):floor((N-1)/2), -floor(M/2):floor((M-1)/2));

D = sqrt(u.^2 + v.^2);

GLPF = exp(-(D.^2) / (2 \* (D0^2)));

Filtered\_F = F\_shifted .\* GLPF;

F\_inverse\_shifted = ifftshift(Filtered\_F);

Processed\_img = real(ifft2(F\_inverse\_shifted));

figure;

subplot(2,3,1), imshow(img\_gray, []), title('Original Image');

subplot(2,3,2), imshow(log(abs(F\_shifted)+1), []), title('Centered Fourier Spectrum');

subplot(2,3,3), imshow(GLPF, []), title('Gaussian Low-Pass Filter');

subplot(2,3,4)

mesh(u, v, GLPF, 'EdgeColor', [0.5 0 0.5], 'FaceColor', 'interp');

title('Gaussian Low-Pass Filter (3D)');

xlabel('X'), ylabel('Y'), zlabel('Amplitude');

subplot(2,3,5), imshow(log(abs(Filtered\_F)+1), []), title('Filtered Fourier Spectrum');

subplot(2,3,6), imshow(uint8(Processed\_img)), title('Processed Image');

Output:

A close-up of a brain scan

AI-generated content may be incorrect.

**Note:** Better than the other Low Pass Filter.

4. Compress the above images using Discrete Cosine Transform (DCT), Haar Transform, and DCT-Haar, and find out which one is better in terms of compression ratio and PSNR for the given images.

Code:

images = {'Picture1.jpg', 'Picture2.jpg', 'Picture3.jpg'};

threshold\_dct = 0.02;

threshold\_haar = 0.02;

level = 2;

results = {};

figure('Name', 'Compression Comparison', 'NumberTitle', 'off');

for i = 1:length(images)

   img = im2double(imread(images{i}));

   if size(img,3)==3

       img = rgb2gray(img);

   end

   dct\_img = dct2(img);

   dct\_img(abs(dct\_img) < threshold\_dct) = 0;

   rec\_dct = idct2(dct\_img);

   cr\_dct = numel(dct\_img) / nnz(dct\_img);

   psnr\_dct = psnr(rec\_dct, img);

   [C,S] = wavedec2(img, level, 'haar');

   C(abs(C) < threshold\_haar) = 0;

   rec\_haar = waverec2(C, S, 'haar');

   cr\_haar = numel(C) / nnz(C);

   psnr\_haar = psnr(rec\_haar, img);

   dct\_hybrid = dct2(img);

   dct\_hybrid(abs(dct\_hybrid) < threshold\_dct) = 0;

   idct\_hybrid = idct2(dct\_hybrid);

   [C2,S2] = wavedec2(idct\_hybrid, level, 'haar');

   C2(abs(C2) < threshold\_haar) = 0;

   rec\_hybrid = waverec2(C2, S2, 'haar');

   cr\_hybrid = (numel(dct\_hybrid)+numel(C2)) / (nnz(dct\_hybrid)+nnz(C2));

   psnr\_hybrid = psnr(rec\_hybrid, img);

   results{i,1} = images{i};

   results{i,2} = cr\_dct;     results{i,3} = psnr\_dct;

   results{i,4} = cr\_haar;    results{i,5} = psnr\_haar;

   results{i,6} = cr\_hybrid;  results{i,7} = psnr\_hybrid;

   subplot(3,4,(i-1)\*4+1), imshow(img), title(['Original - ', images{i}]);

   subplot(3,4,(i-1)\*4+2), imshow(rec\_dct), title('DCT Reconstructed');

   subplot(3,4,(i-1)\*4+3), imshow(rec\_haar), title('Haar Reconstructed');

   subplot(3,4,(i-1)\*4+4), imshow(rec\_hybrid), title('DCT-Haar Reconstructed');

end

fprintf('\n%-12s | %-13s | %-13s | %-13s\n', 'Image', 'DCT (CR, PSNR)', 'Haar (CR, PSNR)', 'Hybrid (CR, PSNR)');

fprintf('%s\n', repmat('-',1,60));

for i = 1:3

   fprintf('%-12s | CR: %5.2f, PSNR: %5.2f | CR: %5.2f, PSNR: %5.2f | CR: %5.2f, PSNR: %5.2f\n', ...

       results{i,1}, results{i,2}, results{i,3}, results{i,4}, results{i,5}, results{i,6}, results{i,7});

end

Output:

A close-up of a chart

AI-generated content may be incorrect.

A collage of x-ray images

AI-generated content may be incorrect.

**Note:**

* DCT performs best for Picture1 and Picture2, offering the highest compression and fidelity.
* Haar Transform excels for Picture3, with significantly better compression and image quality.
* The Hybrid (DCT-Haar) approach is more balanced but does not outperform DCT or Haar in any single metric.

**5:** ApplyGaussian noise to Picture 5, and then use the following to restore the image:

     i. Geometric Mean filter

     ii. Harmonic Mean filter

     iii. Contra-harmonic Mean filter

**Code:**

img = im2double(imread('Picture5.png'));

if size(img,3) == 3

   img = rgb2gray(img);

end

noisy\_img = imnoise(img, 'gaussian', 0, 0.01);

pad\_img = padarray(noisy\_img, [1 1], 'symmetric');

[rows, cols] = size(img);

geo\_img = zeros(rows, cols);

harm\_img = zeros(rows, cols);

contra\_img = zeros(rows, cols);

Q = 1;

for i = 2:rows+1

   for j = 2:cols+1

       window = pad\_img(i-1:i+1, j-1:j+1);

       geo\_img(i-1,j-1) = exp(mean(log(window(:) + eps)));

       harm\_img(i-1,j-1) = 9 / sum(1 ./ (window(:) + eps));

       numerator = sum((window(:)).^(Q+1));

       denominator = sum((window(:)).^Q + eps);

       contra\_img(i-1,j-1) = numerator / denominator;

   end

end

figure('Name', 'Noise Removal Filters', 'NumberTitle', 'off');

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

subplot(2,3,2), imshow(noisy\_img), title('Gaussian Noisy Image');

subplot(2,3,4), imshow(geo\_img), title('Geometric Mean Filter');

subplot(2,3,5), imshow(harm\_img), title('Harmonic Mean Filter');

subplot(2,3,6), imshow(contra\_img), title(['Contra-harmonic Filter (Q=', num2str(Q), ')']);

**Output:**

A close-up of a brain scan

AI-generated content may be incorrect.

**6:**Apply Gaussian noise to Figure 5, and then use the following order statistic filters to restore the image:

     i. Median filter

     ii. Maximum filter

     iii. Minimum filter

     iv. Midpoint filter

     v. Alpha-trimmed filter

     vi. Trimmed filter

**Code:**

img = im2double(imread('Picture5.png'));

if size(img, 3) == 3

   img = rgb2gray(img);

end

noisy\_img = imnoise(img, 'gaussian', 0, 0.01);

k = 3;

pad = floor(k/2);

[rows, cols] = size(noisy\_img);

padded\_img = padarray(noisy\_img, [pad pad], 'symmetric');

median\_img = zeros(rows, cols);

max\_img = zeros(rows, cols);

min\_img = zeros(rows, cols);

midpoint\_img = zeros(rows, cols);

alpha\_trimmed\_img = zeros(rows, cols);

trimmed\_img = zeros(rows, cols);

d = 2;

for i = 1:rows

   for j = 1:cols

       window = padded\_img(i:i+2\*pad, j:j+2\*pad);

       sorted\_vals = sort(window(:));

       median\_img(i,j) = median(sorted\_vals);

       max\_img(i,j) = max(sorted\_vals);

       min\_img(i,j) = min(sorted\_vals);

       midpoint\_img(i,j) = (max(sorted\_vals) + min(sorted\_vals)) / 2;

       trimmed\_window = sorted\_vals((d/2)+1 : end-(d/2));

       alpha\_trimmed\_img(i,j) = mean(trimmed\_window);

       trimmed\_img(i,j) = mean(sorted\_vals(2:end-1));

   end

end

psnr\_values = [

   psnr(median\_img, img),

   psnr(max\_img, img),

   psnr(min\_img, img),

   psnr(midpoint\_img, img),

   psnr(alpha\_trimmed\_img, img),

   psnr(trimmed\_img, img)

];

filter\_names = ["Median", "Maximum", "Minimum", "Midpoint", "Alpha-Trimmed", "Trimmed"];

fprintf('\n--- PSNR Comparison ---\n');

for i = 1:length(psnr\_values)

   fprintf('%s Filter PSNR: %.2f dB\n', filter\_names(i), psnr\_values(i));

end

figure('Name', 'Order Statistic Filters', 'NumberTitle', 'off');

subplot(2,4,1), imshow(img), title('Original Image');

subplot(2,4,2), imshow(noisy\_img), title('Gaussian Noisy Image');

subplot(2,4,3), imshow(median\_img), title('Median Filter');

subplot(2,4,4), imshow(max\_img), title('Maximum Filter');

subplot(2,4,5), imshow(min\_img), title('Minimum Filter');

subplot(2,4,6), imshow(midpoint\_img), title('Midpoint Filter');

subplot(2,4,7), imshow(alpha\_trimmed\_img), title('Alpha-Trimmed Filter');

subplot(2,4,8), imshow(trimmed\_img), title('Trimmed Filter');

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

A collage of images of a brain

AI-generated content may be incorrect.

**7:**By observing and comparing each of the outputs, determine which filter restores the image Figure 5 closest to its original state. Mention the reasoning behind your observation.

The Alpha-Trimmed Filter and the Trimmed Filter both achieved the highest PSNR of 28.73 dB, which means they restored the noisy image closest to the original among all tested filters.

**Table 1. Filter Comparison**

|  |  |  |
| --- | --- | --- |
| **Filter** | **PSNR (dB)** | **Observation** |
| **Alpha-Trimmed** | **28.73** | Best restoration; removes extreme noise while preserving edges. |
| **Trimmed** | **28.73** | Same as alpha-trimmed with simplified trimming. |
| **Median** | 28.25 | Very close; excellent for salt & pepper and Gaussian noise. |
| **Midpoint** | 23.93 | Decent; smooths moderate noise. |
| **Minimum** | 18.00 | Tends to darken image; poor for Gaussian noise. |
| **Maximum** | 15.50 | Tends to brighten image; least effective. |

**NOTE:**

* Alpha-Trimmed and Trimmed Filters discard extreme values (outliers) in a local window, which works well against Gaussian noise that introduces small fluctuations.
* Median filter performs well but is slightly less effective than the trimmed filters in this case.
* Minimum/Maximum filters are more suited to specific noise types (pepper and salt respectively), not Gaussian.
* Midpoint filter works moderately but lacks robustness to complex noise patterns.