Table of Contents


```
%NAME: Vatsalya Chaubey
%INST: IIT, Bhubaneswar
%DATE: 4.10.2020
%CATEGORY: Btech
%BRANCH: Electronics and Communication
%Roll Number: 17EC01044
```


Image and Video Processing Assignment - 3

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

Question 1: Laplacian Sharpening in the frequency domain

```
% Function which performs image sharpening in frequqncy domain using
% Laplacian Filter

function [out] = Laplacian_sharpening(img)
% img: Grayscale input image

dft = fftshift(fft2(img));
[row, col] = size(img);

center_x = (1 + col) /2;
center_y = (1 + row) /2;

filter = zeros(size(dft));
for i=1:row
    for j=1:col
```

```
            filter(i,j) = 1 + 4*pi^2*((j-center_x)^2 + (i-center_y)^2);
        end
    end

    % This finds the inverse FFT after shifting the spectrum back
    out = real(ifft2(ifftshift(filter.*double(dft))));

    end
```

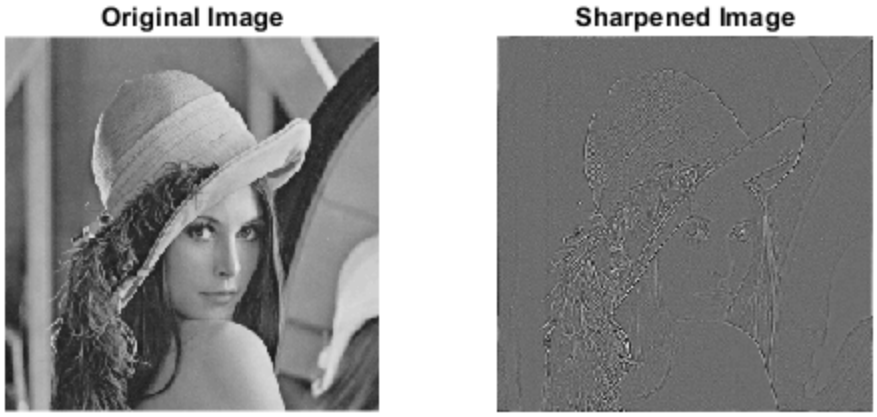
Laplacian sharpening example

The image is first converted into frequency domain and then Laplacian filter in frequency domain is multiplied to it. The inverse transform of the output gives the sharpened image.

The frequency response of Laplacian filter is

$$H(u,v) = -4 * \pi^2 * (u^2 + v^2)(F(u,v))$$

```
% Read the input image
% Read the input image as a double
orig_img = imread('lena_gray_256.tif');
img = double(orig_img);

% Sharpening
out = Laplacian_sharpening(img);
out = mat2gray(out);

figure('Name', 'Laplacian Sharpening');
subplot(121)
imshow(orig_img);
title('Original Image');

subplot(122)
imshow(out);
title('Sharpened Image');
```



Question 2: Gaussian Low pass filter in frequency domain

```
% Function which performs image smoothening in frequqncy domain using
% Low pass Gaussian Filter

function [out] = low_pass_gaussian(img, sigma)
% img: Grayscale input image

dft = fftshift(fft2(img));
[row, col] = size(img);

center_x = (1 + col) /2;
center_y = (1 + row) /2;

filter = zeros(size(dft));
for i=1:row
    for j=1:col
        filter(i,j) = exp(-((j-center_x)^2 + (i-center_y)^2)/ (2 *
 sigma^2));
    end
end
```

```
% This finds the inverse FFT after shifting the spectrum back
out = real(ifft2(ifftshift(filter.*double(dft))));

end
```

Gaussian Low pass example

This demonstrates image smoothing through low pass filtering. Here also the same methodology is used and filtering is done in frequency domain.

The frequency response of Laplacian filter is

$$H(u,v) = exp^{\frac{-D(u,v)^2}{2*D_0^2}} * F(u,v)$$

```
% Read the input image
% Read the input image as a double
orig_img = imread('lena_gray_256.tif');
img = double(orig_img);

% Gaussian Low pass filter
out1 = low_pass_gaussian(img, 10);
out2 = low_pass_gaussian(img, 50);

% Converting matrix to image
out1 = mat2gray(out1);
out2 = mat2gray(out2);

figure('Name', 'Gaussian Low Pass');
subplot(131)
imshow(orig_img);
title('Original Image');

subplot(132)
imshow(out1);
title('Low passed image (Sigma = 10)');

subplot(133)
imshow(out2);
title('Low passed image (Sigma = 50)');
```

Question 3: Gaussian High pass filter in frequency domain

```
% Function which performs image smoothening in frequqncy domain using
% High pass Gaussian Filter

function [out] = high_pass_gaussian(img, sigma)
% img: Grayscale input image

dft = fftshift(fft2(img));
[row, col] = size(img);

center_x = (1 + col) /2;
center_y = (1 + row) /2;

filter = zeros(size(dft));
for i=1:row
    for j=1:col
        filter(i,j) = 1 - exp(-((j-center_x)^2 + (i-center_y)^2)/ (2 *
 sigma^2));
    end
end
```

```
% This finds the inverse FFT after shifting the spectrum back
out = real(ifft2(ifftshift(filter.*double(dft))));

end
```

Gaussian High pass example

This demonstrates image smoothing through low pass filtering. Here also the same methodology is used and filtering is done in frequency domain.

The frequency response of Laplacian filter is

$$H(u,v) = (1 - exp^{\frac{-D(u,v)^2}{2*D_0^2}}) * F(u,v)$$

```
% Read the input image
% Read the input image as a double
orig_img = imread('lena_gray_256.tif');
img = double(orig_img);

% Gaussian High pass filter
out1 = high_pass_gaussian(img, 10);
out2 = high_pass_gaussian(img, 50);

% Converting matrix to image
out1 = mat2gray(out1);
out2 = mat2gray(out2);

figure('Name', 'Gaussian High Pass');
subplot(131)
imshow(orig_img);
title('Original Image');

subplot(132)
imshow(out1);
title('High passed image (Sigma = 10)');

subplot(133)
imshow(out2);
title('High passed image (Sigma = 50)');
```

Erosion

```
% Function to perform erosion on a binary image using a structuring
 element

function [out] = erosion(img, ele)
% img: Grayscale input image
% ele: Structuring element

[row, col] = size(img);
[row_e, col_e] = size(ele);

% Output image is same as the input and then we will perform erosion
 on it
out = img;

for i = 1 + floor(row_e / 2): row - floor(row_e/2)
    for j = 1 + floor(col_e / 2): col - floor(col_e / 2)

        % Calculating the image section on which the erosion will be
        % applied
        x_start = i - floor(row_e/2);
        y_start = j - floor(col_e/2);
```

```
            x_end = i + floor(row_e/2);
            y_end = j + floor(col_e/2);

            img_section = img(x_start:x_end, y_start:y_end);

            % Checking whether the image section and structuring elements
 are
            % matching
            if sum(sum(img_section.*ele)) == sum(sum(ele))
                out(i,j) = 1;
            else
                out(i,j) = 0;
            end

        end

    end
```

Dilation

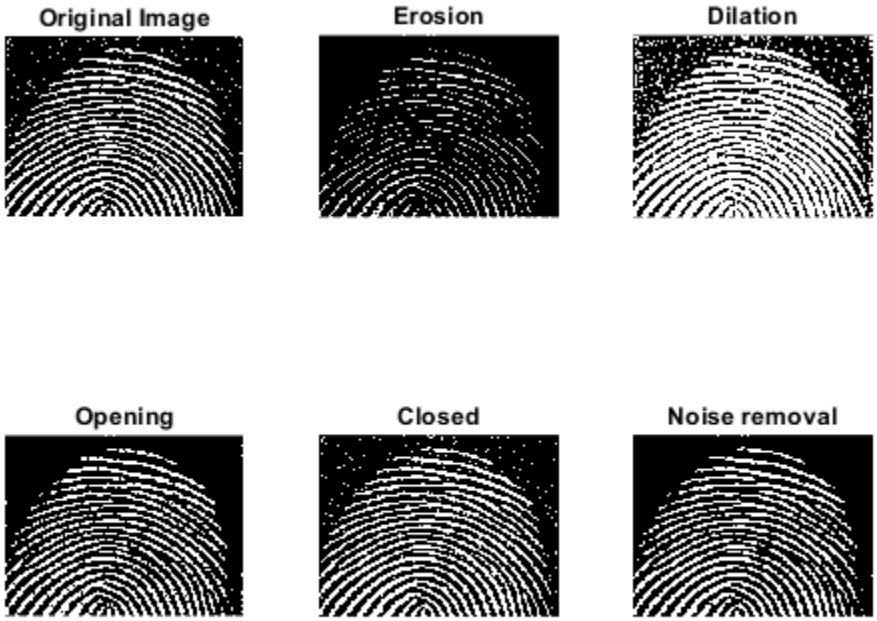
```
% Function to perform dilation on a binary image using a structuring
 element

function [out] = dilation(img, ele)
% img: Grayscale input image
% ele: Structuring element

[row, col] = size(img);
[row_e, col_e] = size(ele);

% Output image is same as the input and then we will perform dilation
 on it
out = img;

for i = 1 + floor(row_e / 2): row - floor(row_e/2)
    for j = 1 + floor(col_e / 2): col - floor(col_e / 2)

        % Calculating the image section on which the dilation will be
        % applied
        x_start = i - floor(row_e/2);
        y_start = j - floor(col_e/2);

        x_end = i + floor(row_e/2);
        y_end = j + floor(col_e/2);

        img_section = img(x_start:x_end, y_start:y_end);

        % Checking whether the image section and structuring elements
 have
        % some overlap
```

```
            if sum(sum(img_section.*ele)) >= 1
                out(i,j) = 1;
            else
                out(i,j) = 0;
            end

        end

    end
```

Question 4: Noise removal through erosion and dilation

```
% Read the input image
% Read the input image as a double
orig_img = imread('fingerprint.tif');
img = double(orig_img);

% Structuring element
ele = [1,1,1;1,1,1;1,1,1];

% Erosion
eroded = erosion(img, ele);

% Dilation
dilated = dilation(img, ele);

% Opening: One of the methods to remove noise using erosion and
 dilation.
% Opening removes narrow isthumes and small protrusion of an image. It
 is
% usually done by first eroding the image and then dilating it.

opened = dilation(erosion(img, ele), ele);

% Cloasing: Another method to remove noise using erosion and dilation.
% Closing joins narrow isthumes and fills in small protrusion of an
 image. It is
% usually done by first dilating the image and then eroding it.

closed = erosion(dilation(img, ele), ele);

% One more useful way to remove noise is to perform closing after
 opening
% on an image
noise_removed = erosion(dilation(opened, ele), ele);

figure('Name', 'Erosion and Dilation');
subplot(231)
imshow(orig_img);
```

```
title('Original Image');

subplot(232)
imshow(eroded);
title('Erosion');

subplot(233)
imshow(dilated);
title('Dilation');

subplot(234)
imshow(opened);
title('Opening');

subplot(235)
imshow(closed);
title('Closed');

subplot(236)
imshow(noise_removed);
title('Noise removal');
```



Conclusion

This experiment illustrates how low pass and high pass filtering can be optimally performed using DFT and IDFT such that image enhancements such as Sharpening and Blurring can be done. We also saw the

use of basic morphological transforms like erosion and dilation. It is quite clear that using erosion and dilation and different forms of structuring elements we can come up with sophisticated techniques for noise removal, hole filling, joining connected components and other transformations.

Published with MATLAB® R2020b