Digital Image processing Lab

Islamic University – Gaza
Engineering Faculty
Department of Computer Engineering
2013
EELE 5110: Digital Image processing Lab
Eng. Ahmed M. Ayash

Lab # 8

Fourier Transform and Frequency-Domain Filters

April 03, 2013

1. Objectives

- Computing of the Fourier Transform for an image and displaying the spectral image.
- Designing of filters in the frequency domain (lowpass and highpass filters) and apply them to images.

2. Theory

2.1 Fourier Transform:

The Fourier transform is a representation of an image as a sum of complex exponentials of varying magnitudes, frequencies, and phases. The Fourier transform plays a critical role in a broad range of image processing applications, including enhancement, analysis, restoration, and compression.

Working with the Fourier transform on a computer usually involves a form of the transform known as the **discrete Fourier transform (DFT)**. There are two principal reasons for using this form:

1) The input and output of the DFT are both discrete, which makes it convenient for computer manipulations.
2) There is a fast algorithm for computing the DFT known as the **fast Fourier transform (FFT)**.

The DFT is usually defined for a discrete function *f*(*x*,*y*) that is nonzero only over the finite region $0 \leq x \leq M$-1 and $0 \leq y \leq N$-1.
The general idea is that the image (*f(x,y)* of size *M* x *N*) will be represented in the frequency domain (*F(u,v)*). The equation for the two-dimensional discrete Fourier transform (DFT) is:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)}$$

For u=0, 1, 2,……., M-1
For v=0, 1, 2,……., N-1

The concept behind the Fourier transform is that any waveform that can be constructed using a sum of sine and cosine waves of different frequencies. The exponential in the above formula can be expanded into sins and cosines with the variables *u* and *v* determining these frequencies
The inverse of the above discrete Fourier transform is given by the following equation:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M + vy/N)}$$

For x=0, 1, 2,……., M-1
For y=0, 1, 2,……., N-1

Thus, if we have $F(u,v)$, we can obtain the corresponding image ($f(x,y)$) using the inverse discrete Fourier transform.

Things to note about the discrete Fourier transform are the following:

- The value of the transform at the origin of the frequency domain, at $F(0,0)$, is called the **DC component**
  - $F(0,0)$ is equal to MN times the average value of $f(x,y)$.
  - In MATLAB, $F(0,0)$ is actually $F(1,1)$ because array indices in MATLAB start at 1 rather than 0.
- The values of the Fourier transform are complex, meaning they have real and imaginary parts. The imaginary parts are represented by i, which is the square root of -1
- We visually analyze a Fourier transform by computing a **Fourier spectrum** (the magnitude of $F(u,v)$) and display it as an image.
  - The Fourier spectrum is symmetric about the center.
- The fast Fourier transform (FFT) is a fast algorithm for computing the discrete Fourier transform.
- MATLAB has three functions to compute the DFT:
  1. fft - for one dimension (useful for audio)
  2. fft2 - for two dimensions (useful for images)
  3. fftn - for n dimensions
- MATLAB has three functions that compute the inverse DFT:
  1. ifft
  2. ifft2
  3. ifftn
- The function ***fftshift*** is used to shift the zero-frequency component to center of spectrum. Note that it is so important to apply a logarithmic transformation function on the spectral image before displaying so as spectral details are efficiently displayed.

How does the Discrete Fourier Transform relate to Spatial Domain Filtering?

The following convolution theorem shows an interesting relationship between the spatial domain and frequency domain:

$$f(x,y)*h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

And, conversely,

$$f(x,y)h(x,y) \Leftrightarrow F(u,v)*H(u,v)$$

The symbol "*" indicates convolution of the two functions. The important thing to extract out of this is that the multiplication of two Fourier transforms corresponds to the convolution of the associated functions in the spatial domain.

Basic Steps in DFT Filtering

The following summarize the basic steps in DFT Filtering

1. Obtain the Fourier transform:

   ```
   F=fft2(f);
   ```

2. Generate a filter function, H
3. Multiply the transform by the filter:

   ```
   G=H.*F;
   ```

4. Compute the inverse DFT:

   ```
   g=ifft2(G);
   ```

5. Obtain the real part of the inverse FFT of g:

   ```
   g2=real(g);
   ```

2.2 Filters in the Frequency Domain

Based on the property that multiplying the FFT of two functions from the spatial domain produces the convolution of those functions, you can use Fourier transforms as a fast convolution on large images. As a note, on small images, it is faster to work in the spatial domain.
However, you can also create filters directly in the frequency domain. There are three commonly discussed filters in the frequency domain:

- Lowpass filters, sometimes known as smoothing filters
- Highpass filters, sometimes known as sharpening filters
- Bandpass filters.

- ✓ A Lowpass filter attenuates high frequencies and retains low frequencies unchanged.
- ✓ A Highpass filter blocks all frequencies smaller than D_o and leaves the others unchanged.
- ✓ Bandpass filters are a combination of both lowpass and highpass filters. They attenuate all frequencies smaller than a frequency D_o and higher than a frequency D_1, while the frequencies between the two cut-offs remain in the resulting output image.

Lowpass filters:

- create a blurred (or smoothed) image
- attenuate the high frequencies and leave the low frequencies of the Fourier transform relatively unchanged

Three main lowpass filters are discussed in *Digital Image Processing Using MATLAB*:

1. **Ideal lowpass filter (ILPF):** The simplest low pass filter that cutoff all high frequency components of the Fourier transform that are at the distance greater than distance D_0 from the center.

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

- Where D_0 is a specified nonnegative quantity (cutoff frequency), and D(u,v) is the distance from point (u,v) to the center of the frequency rectangle.
- The center of frequency rectangle is (M/2,N/2)
- The distance from any point (u,v) to the center D(u,v) of the Fourier transform is given by:

$$D(u,v) = \left[(u - M/2)^2 + (v - N/2)^2\right]^{1/2}$$

M and *N* are sizes of the image.

2. **Butterworth lowpass filter (BLPF)**: of order n, and with cutoff frequency at a distance D_0 from the center.

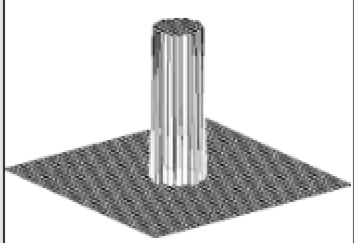
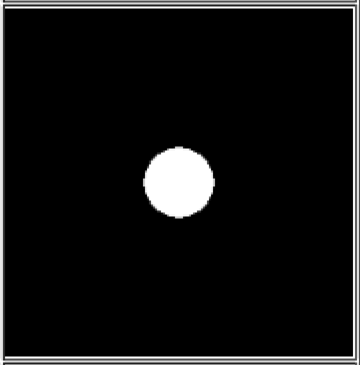
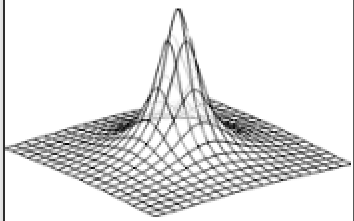
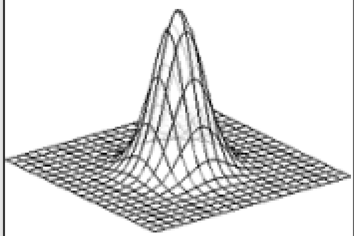
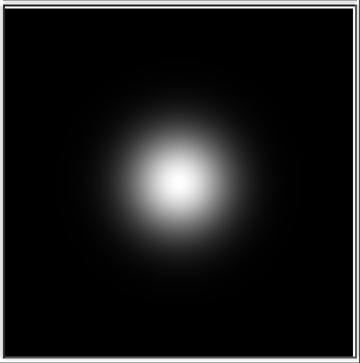
$$H(u,v) = \frac{1}{1 + \left[D(u,v)/D_0\right]^{2n}}$$

3. **Gaussian lowpass filter (GLPF)**

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

The GLPF did not achieve as much smoothing as the BLPF of order 2 for the same value of cutoff frequency

The corresponding formulas and visual representations of these filters are shown in the table below. In the formulae, D_0 is a specified nonnegative number (cutoff frequency). *D(u,v)* is the distance from point *(u,v)* to the center of the filter. Butterworth low pass filter (BLPF) of order n.

Lowpass Filter	Mesh	Image
Ideal: $H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)>D_0\end{cases}$		
Butterworth: $H(u,v)=\dfrac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$		
Gaussian: $H(u,v)=e^{-D^2(u,v)/2D_0^2}$		

Highpass filters:

- sharpen (or shows the edges of) an image
- attenuate the low frequencies and leave the high frequencies of the Fourier transform relatively unchanged

The highpass filter (H_{hp}) is often represented by its relationship to the lowpass filter (H_{lp}):

$$H_{HP}(u,v)=1-H_{LP}(u,v)$$

Because highpass filters can be created in relationship to lowpass filters, the following table shows the three corresponding highpass filters by their visual representations:

1. **Ideal highpass filter (IHPF)**

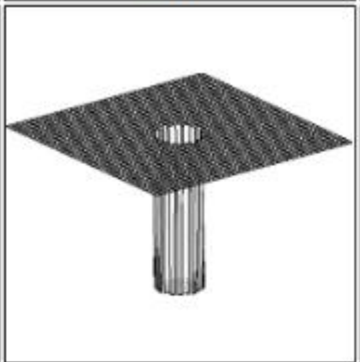
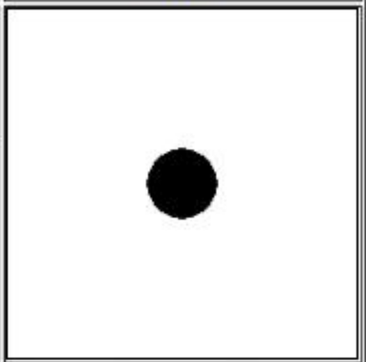
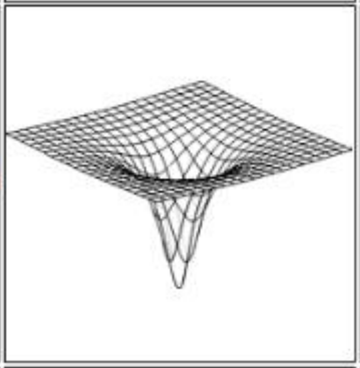
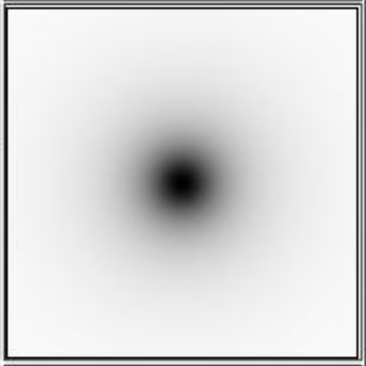
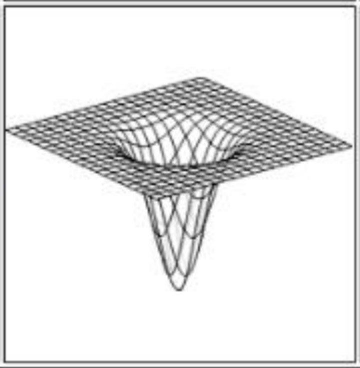
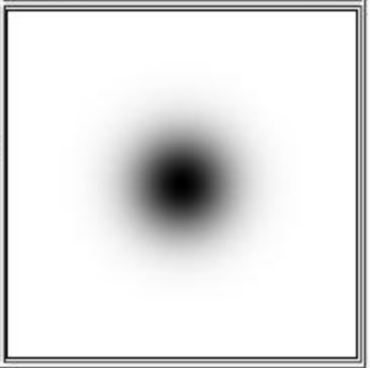
$$H(u,v)=\begin{cases}0 & D(u,v)\le D_0\\ 1 & D(u,v)>D_0\end{cases}$$

2. **Butterworth highpass filter (BHPF)**

$$H(u,v) = \frac{1}{1+\left[D_0/D(u,v)\right]^{2n}}$$

3. **Gaussian highpass filter (GHPF)**

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

Highpass	Mesh	Image
Ideal		
Butterworth		
Gaussian		

In Matlab, to get **lowpass filter** we use this command:

```
H = fspecial('gaussian',HSIZE,SIGMA)
```

- Returns a rotationally symmetric Gaussian lowpass filter of size HSIZE with standard deviation SIGMA (positive).

- HSIZE can be a vector specifying the number of rows and columns in H or scalar, in which case H is a square matrix.

- The default HSIZE is [3 3], the default SIGMA is 0.5.

In Matlab, to get **highpass laplacian filter** we use this command:

```
H = fspecial('laplacian',ALPHA)
```

- Returns a 3-by-3 filter approximating the shape of the two-dimensional Laplacian operator.

- The parameter ALPHA controls the shape of the Laplacian and must be in the range 0.0 to 1.0.

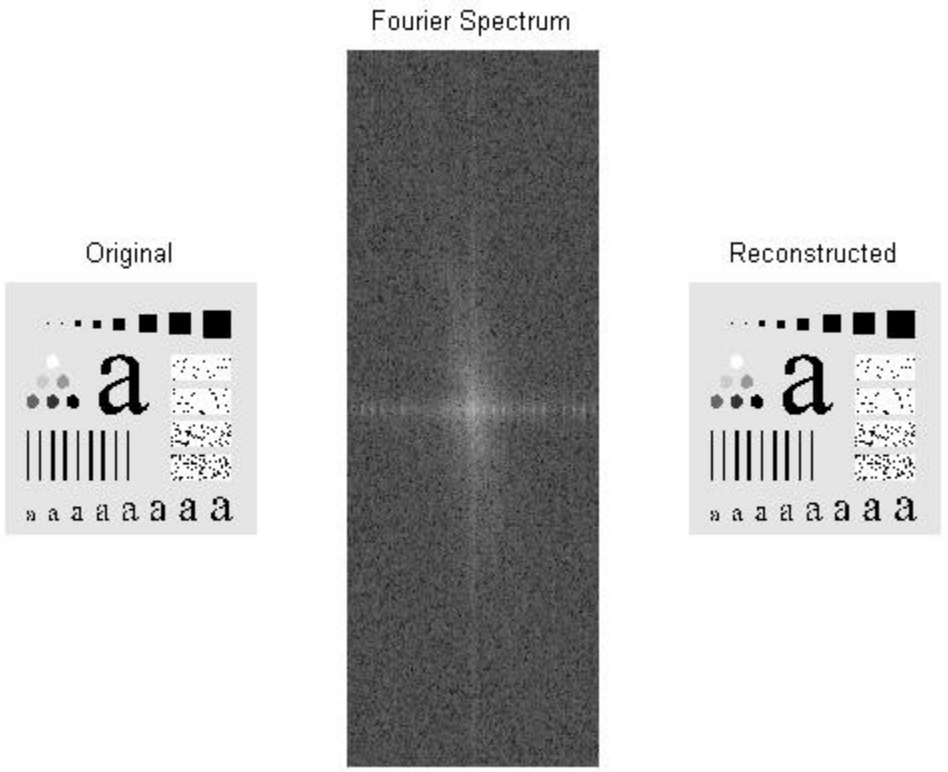
- The default ALPHA is 0.2

3. Exercises

Exercise1: Apply FFT and IFFT.

```
%ex1.m
close all
clear
clc
%=====================================
% 1) Displaying the Fourier Spectrum:
%=====================================
I=imread('Lab8_1.jpg');
I=im2double(I);
FI=fft2(I);    %(DFT) get the frequency  for the image
FI_S=abs(fftshift(FI));%Shift zero-frequency component to center of
img_spectrum.
I1=ifft2(FI);
I2=real(I1);
subplot(131),imshow(I),title('Original'),
subplot(132),imagesc(0.5*log(1+FI_S)),title('Fourier Spectrum'),axis off
subplot(133),imshow(I2),title('Reconstructed')
%imagesc: the data is scaled to use the full colormap.
```

Output:

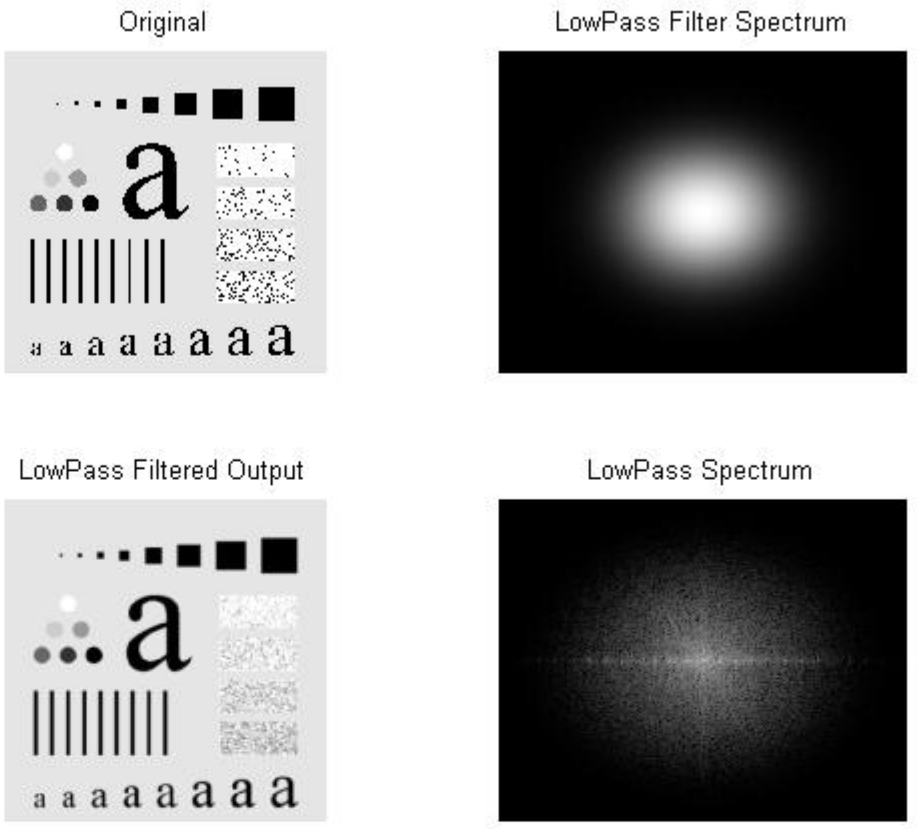


Exercise2: Apply lowpass filter.

```
%ex2.m
close all
clear
clc
%==============================
% 2) Low-Pass Gaussian Filter:
%==============================
I=imread('Lab8_1.jpg');
I=im2double(I);
FI=fft2(I);       %1.Obtain the Fourier transform
LP=fspecial('gaussian',[11 11],1.3); %2.Generate a Low-Pass filter
FLP=fft2(LP,size(I,1),size(I,2)); %3. Filter padding
LP_OUT=FLP.*FI;  %4.Multiply the transform by the filter
I_OUT_LP=ifft2(LP_OUT); %5.inverse DFT
I_OUT_LP=real(I_OUT_LP); %6.Obtain the real part(Output)

%%%%spectrum%%%%
FLP_S=abs(fftshift(FLP));%Filter spectrum
LP_OUT_S=abs(fftshift(LP_OUT));%output spectrum

subplot(221),imshow(I),title('Original'),
subplot(222),imagesc(0.5*log(1+FLP_S)),title('LowPass Filter
Spectrum'),axis off
subplot(223),imshow(I_OUT_LP),title('LowPass Filtered Output')
```

```
subplot(224),imagesc(0.5*log(1+LP_OUT_S)),title('LowPass
Spectrum'),axis off
```

Output:



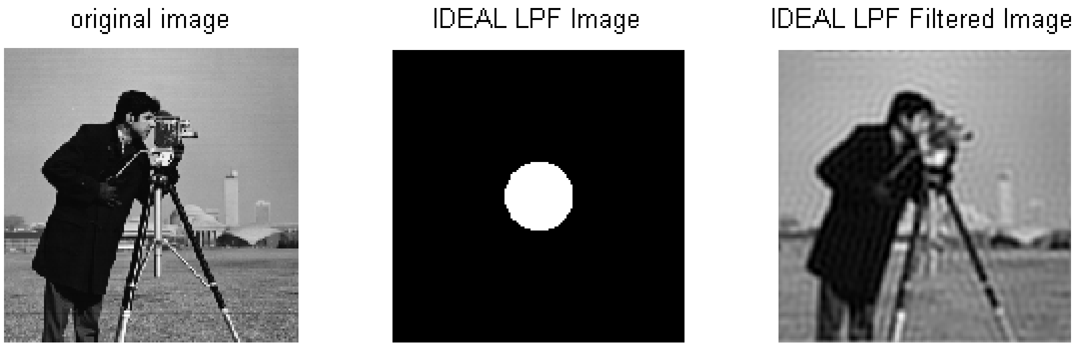
Exercise3: Apply Ideal lowpass filter.

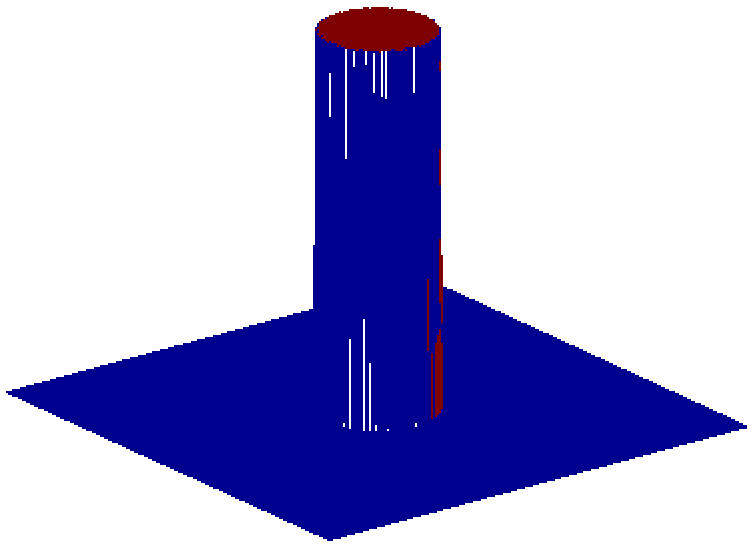
```
%ex3.m
close all
clear
clc
a=imread('Lab8_2.tif');
[M N]=size(a);
a=im2double(a);
F1=fft2(a);      %1.Obtain the Fourier transform

% Set up range of variables.
u = 0:(M-1); %0-255
v = 0:(N-1);%0-255
 % center (u,v) = (M/2,N/2)
% Compute the indices for use in meshgrid
idx = find(u > M/2);% indices 130-255
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;
```

```
%set up the meshgrid arrays needed for
% computing the required distances.
[U, V] = meshgrid(u, v);

% Compute the distances D(U, V).
D = sqrt(U.^2 + V.^2);

disp('IDEAL LOW PASS FILTERING IN FREQUENCY DOMAIN');

D0=input('Enter the cutoff distance==>');
% Begin filter computations.
H = double(D <= D0);

G=H.*F1;          %Multiply
G=ifft2(G);
G=real(G);
ff=abs(fftshift(H));
subplot(131),imshow(a),title('original image')
subplot(132),imshow(ff),title('IDEAL LPF Image')
subplot(133),imshow(G),title('IDEAL LPF Filtered Image')
figure, mesh(ff),axis off,grid off
```

Output:

IDEAL LOW PASS FILTERING IN FREQUENCY DOMAIN
Enter the cutoff distance==>30





Homework

1) Write a Matlab code to apply highpass laplacian filter on *Lab8_1.jpg* image.

2) Write a Matlab code to apply ideal highpass filter on *Lab8_1.jpg* image for D_0=100

3) Apply FFT2, IFFT2, lowpass Gaussian filter, and highpass laplacian filter on *Lab8_3.jpg* image.