1. Objectives
1. Computing of the Fourier Transform for an image and displaying the spectral image.
2. Designing of filters in the frequency domain 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, and 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 two-dimensional \( M \)-by-\( N \) DFT and inverse \( M \)-by-\( N \) DFT relationships are given by:

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

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

The values \( F(u,v) \) are the DFT coefficients of \( f(x,y) \). The zero-frequency coefficient, \( F(0,0) \), is often called the “DC component.” (Note that matrix indices in MATLAB always start at 1 rather than 0; therefore, the matrix elements \( f(1,1) \) and \( F(1,1) \) correspond to the mathematical quantities \( f(0,0) \) and \( F(0,0) \), respectively.) The MATLAB functions \texttt{fft}, \texttt{fft2}, and \texttt{fftn} implement the fast Fourier transform algorithm for computing the one-dimensional DFT, two-dimensional DFT, and \( N \)-dimensional DFT, respectively. The functions \texttt{ifft}, \texttt{ifft2}, and \texttt{ifftn} compute the inverse DFT. The function \texttt{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.

2.2 Filters in the Frequency Domain
The Fast Fourier Transform (FFT) is the most widely known example of the Spectral method for computational problems. In Fourier transformations, the mapping is from the time-domain to the frequency-domain. The FFT is widely used in the field of image processing, where one commonly describe an image in terms of intensity values in a two-dimensional matrix. Better results can be achieved with a Gaussian shaped filter function. The advantage is that the Gaussian has the same shape in the spatial and Fourier domains and therefore does not incur the ringing effect in the spatial domain of the filtered image. A commonly used discrete
approximation to the Gaussian is the Butterworth filter. Applying this filter in the frequency domain shows a similar result to the Gaussian smoothing in the spatial domain. One difference is that the computational cost of the spatial filter increases with the standard deviation (i.e. with size of the filter kernel), whereas the costs for a frequency filter are independent of the filter function. The spatial Gaussian filter is more appropriate for narrow lowpass filters, while the Butterworth filter is a better implementation for wide lowpass filters. The same principles apply to highpass filters. We obtain a highpass filter function by inverting the corresponding lowpass filter, e.g. an ideal highpass filter blocks all frequencies smaller than $D_o$ and leaves the others unchanged.

The form of the filter function determines the effects of the operator. There are basically three different kinds of filters: lowpass, highpass and bandpass filters. A low-pass filter attenuates high frequencies and retains low frequencies unchanged. The result in the spatial domain is equivalent to that of a smoothing filter; as the blocked high frequencies correspond to sharp intensity changes, i.e. to the fine-scale details and noise in the spatial domain image.

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. We obtain the filter function of a bandpass by multiplying the filter functions of a lowpass and of a highpass in the frequency domain, where the cut-off frequency of the lowpass is higher than that of the highpass. A highpass filter, on the other hand, yields edge enhancement or edge detection in the spatial domain, because edges contain many high frequencies. Areas of rather constant graylevel consist of mainly low frequencies and are therefore suppressed. A bandpass attenuates very low and very high frequencies, but retains a middle range band of frequencies. Bandpass filtering can be used to enhance edges (suppressing low frequencies) while reducing the noise at the same time (attenuating high frequencies).

3. Appendix
1) Run the following code.
2) Change the image and repeat running.
3) Comment on the results you got.

close all

clear
cle

%===================================
% 1) Displaying the Fourier Spectrum:
%===================================

I=imread('fig4_18_a.jpg');
I=im2double(I);
FI=fft2(I);
FI_S=abs(fftshift(FI));
% I1=ifft2(FI);
% I2=real(I1);
subplot(131),imshow(I),title('Original'),
subplot(132),imagesc(0.5*log10(1+FI_S)),title('Fourier Spectrum')
subplot(133),imshow(I2),title('Reconstructed')
pause
close all

% 2) Low-Pass Gaussian Filter:
LP=fspecial('gaussian',[11 11],1.3);
FLP=fft2(LP,size(I,1),size(I,2));
FLP_S=abs(fftshift(FLP));
LP_OUT=FLP.*FI;
I_OUT_LP=abs(ifft2(LP_OUT));
subplot(221),imshow(I),title('Original'),
subplot(222),imagesc(0.5*log10(1+FLP_S)),colormap(gray),title('LowPass Filter'),
subplot(223),imagesc(0.5*log10(1+abs(fftshift(LP_OUT)))),colormap(gray),title('LowPass Spectrum'),
subplot(224),imshow(I_OUT_LP),title('LowPass Output')
pause
close all

% 3) High-Pass Laplacian Filter:
HP=fspecial('laplacian');
FHP=fft2(HP,size(I,1),size(I,2));
FHP_S=abs(fftshift(FHP));
HP_OUT = FHP.*FI;

I_OUT_HP = abs(fft2(HP_OUT));

subplot(221), imshow(I), title('Original'),

subplot(222), imagesc(0.5*log10(1+FHP_S)), colormap(gray), title('HighPass Filter'),

subplot(223), imagesc(0.5*log10(1+abs(fftshift(HP_OUT)))), colormap(gray), title('HighPass Spectrum'),

subplot(224), imshow(I_OUT_HP), title('HighPass Output')

pause

close all

%-----------------------------------------------------

Home Exercise:

Search MATLAB and try to implement a Band-pass filter to the image used and comment …