RUTGERS UNIVERSITY
School of Electrical and Computer Engineering

ECE 447 Fall Course
Laboratory No. 3 Solutions
Image Processing in MATLAB – Fourier Analysis and Filtering of Images

---

1 Preliminaries

You can access MATLAB toolkits from either the departmental computers in CoRE Room 548 or from any computer with both MATLAB and the Image Processing Toolbox. The image files referenced in this lab can be accessed from the course website at:

`cronos.rutgers.edu/~lrr/`

If you are a new MATLAB user, try running the built-in demos. Just type `demos` at the MATLAB prompt.

2 Exercise 1 – 2-D Fourier Transforms

As we learned in class, given an image $f(x, y)$ defined for $x = 0, 1, \ldots, M-1$ and $y = 0, 1, \ldots, N-1$, the 2-D discrete Fourier transform (DFT) of $f(x, y)$ is denoted by $F(u, v)$ and is given by:

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

and is defined for $u = 0, 1, \ldots, M-1$ and $v = 0, 1, \ldots, N-1$. Similarly the inverse discrete Fourier transform is given by:

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

again for $x = 0, 1, \ldots, M-1$ and $y = 0, 1, \ldots, N-1$. The MATLAB routines for computing the 2-D DFT and the inverse 2-D DFT are the routines `fft2` and `ifft2`.

Using the image files `hardware.tif`, `shuttle.tif`, and `xray.tif`, read in each of the images and compute the 2-D DFT magnitude for each of the images. On a single plot (using a grid of 2 x 2 images), plot the following:

- the original gray scale image in the upper left cell

- the image representation of the 2-D DFT magnitude of the image being studied in the upper right cell
- a clipped and scaled version of the 2-D DFT magnitude that is clipped and scale at a level where you can see the nature of the 2-D DFT magnitude for each image; this plot should be placed in the lower left cell
- the log transformed 2-D DFT magnitude plotted again on a scale that enables you to see the structure of the transform magnitude; this plot should be placed in the lower right cell

You will need to use the MATLAB routine `FS=fftshift(F)` to shift the DC magnitude point from the upper left corner of the image to the center of the magnitude range. In your Lab 3 report, include the MATLAB code along with image plots of the three images for which you repeat the analysis. Can you say how the image properties (in a gross sense) are seen in the 2-D DFT magnitude plots?

****

SOLUTION

The MATLAB code for reading in an image file, computing the magnitude spectrum, and plotting the requested images is as follows:

```
% plot_image_transform
%
    % f=imread('shuttle.tif','tiff');
    % f=imread('xray.tif','tiff');
    f=imread('hardware.tif','tiff');
% plot original image in first quadrant
    figure(1),subplot(221),imshow(f);
    stitle=sprintf('original tif file');
    title(stitle);
% compute image spectrum magnitude and plot in second quadrant
    F=abs(fftshift(fft2(f)));
    subplot(222),imshow(F, [  ]);
    stitle1=sprintf('abs of 2D FFT');
    title(stitle1);
    % axis([-1 1 1 -1]);
    xlabel('horizontal frequency'),ylabel('vertical frequency');
% determine minimum and maximum of spectrum magnitude, print results
    Fmin=min(min(F));
    Fmax=max(max(F));
    fprintf('Fmin, Fmax: %f %f \n',Fmin,Fmax);
% scale magnitude based on Fmin and Fmax and plot in third quadrant
    subplot(223),imshow(F,  [1.5*Fmin, Fmax*1.e-3]);
    stitle2=sprintf('clipped/scaled abs of 2D FFT');
```

```
    title(stitle2);
    xlabel('horizontal frequency'),ylabel('vertical frequency');
% log scale magnitude and plot in fourth quadrant
    subplot(224),imshow(log(1+F), [  ]);
    stitle3=sprintf('log transformed abs of 2D FFT');
    title(stitle3);
    xlabel('horizontal frequency'),ylabel('vertical frequency');
```

The resulting plots from the three files (`hardware.tif`, `shuttle.tif` and `xray`) are shown in Figures 1, 2 and 3.

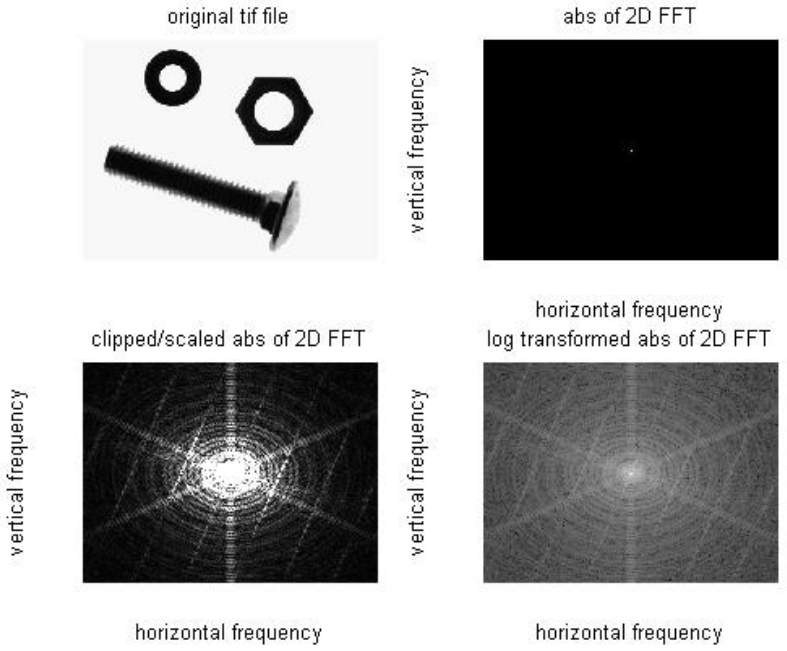

Figure 1: Plots of the original image (first quadrant), the raw spectral magnitude (second quadrant), the scaled and clipped spectral magnitude (third quadrant), and the log intensity transformed spectral magnitude (fourth quadrant).

It can easily be seen that the image plot of the raw spectral magnitude contains virtually no information about the image that is of note, whereas the clipped and scaled or the log-transformed intensity images show the major concentrations of spectral energy and their locations in a fairly prominent manner. It can also be seen that the spectral magnitude is greatest along certain angles and radii, indicating the nature of the image being processed.

****

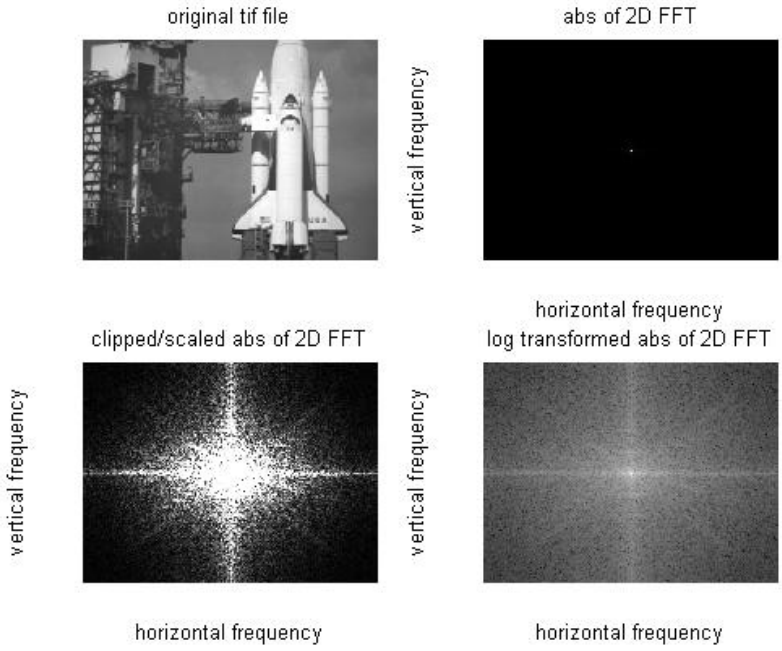

Figure 2: Plots of the original image (first quadrant), the raw spectral magnitude (second quadrant), the scaled and clipped spectral magnitude (third quadrant), and the log intensity transformed spectral magnitude (fourth quadrant).

3 Exercise 2 – Importance of DFT Phase

The phase angle of the Fourier transform of an image contains a great deal of information about the image. To see this we will next perform a simple set of experiments. First we define the quantities of interest precisely.

If we denote the gray scale (sampled) image as $f[m,n]$ with 2-D DFT, $F[i,k]$. then we can define the "magnitude only" image corresponding to $f[m,n]$ as the image obtained by taking the 2-D inverse DFT of the image magnitude function, i.e.,

$$f_{\text{mag}}[m,n] \longleftrightarrow |F[i,k]|$$

i.e., $f_{\text{mag}}[m,n]$ is the inverse 2-D DFT of $|F[i,k]|$. We can also define the "phase only" image corresponding to $f[m,n]$ as the image obtained by taking the 2-D inverse DFT of the image phase function, i.e.,

$$f_{\text{ph}}[m,n] \longleftrightarrow e^{j \arg\{F[i,k]\}}$$

where $\arg\{F[i,k]\}$ is the phase angle of $F[i,k]$.

(a) Write a MATLAB program to compute the magnitude only and phase only versions of the two gray scale images, `shuttle.tif` and `hardware.tif`, and plot the results in a 2 x 2 grid with the following format:

- the upper left and lower left images should be the original gray scale image files (so that you can compare them to the transformed images directly).

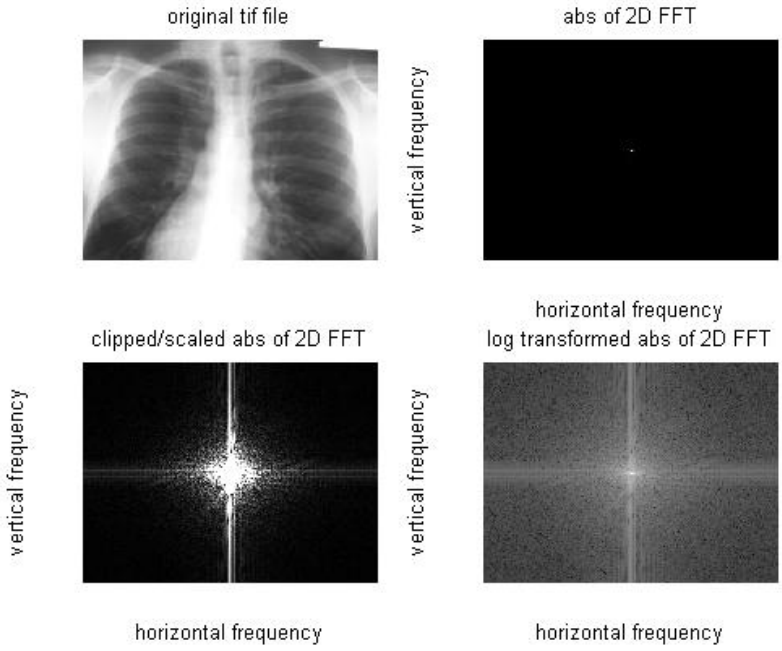

Figure 3: Plots of the original image (first quadrant), the raw spectral magnitude (second quadrant), the scaled and clipped spectral magnitude (third quadrant), and the log intensity transformed spectral magnitude (fourth quadrant).

- the upper right image should be the image reconstructed from magnitude only information
- the lower right image should be the image reconstructed from phase only information

Include your MATLAB code in the report, along with the two requested plots (one for each set of images).

(b) Next perform the following experiment. Compute the 2-D DFTs of the two images used in part (a) of this exercise. From these transforms, form two new DFTs in which the magnitudes and phases are interchanged. Compute the corresponding two images and display them, along with the original gray scale images, in a 2 x 2 grid of the form:

- the upper left image should be the first original image
- the upper right image should be the image reconstructed with the cross magnitude and the correct image phase signal.
- the lower left image should be the second original image
- the lower right image should be the image reconstructed with the cross magnitude and the correct image phase signal.

Include your MATLAB code in the report, along with the requested plot. On the basis of this exercise (as well as the results from Exercise 1) what can you say about the relative

importance of magnitude and phase in preserving image properties. Why do you think the phase contributes so much more than the magnitude to preserving some of the original image properties.

****

SOLUTION

(a) MATLAB code for reading in two image files and computing the "magnitude-only" and "phase-only" versions of the images and plotting these images on a common plot is given below.

```
% mag_phase_analysis
%
% read in two image files; first is hardware, second is shuttle
    f=imread('hardware.tif','tiff');
    g=imread('shuttle.tif','tiff');
    % f=imread('xray.tif','tiff');
    % g=imread('lighthou.tif','tiff');
% plot original first image file in first quadrant
    figure,subplot(221),imshow(f,[   ]);
% transform first image to frequency domain and compute magnitude and phase
    F=fft2(f);
    FA=abs(F);
    FP=atan2(imag(F),real(F));
    FB=exp(j*FP);
% convert magnitude only back to image; convert phase only back to image
    FM=real(ifft2(FA));
    FN=real(ifft2(FB));
% determine maximum and minimum of magnitude image and print on screen
    FMmax=max(max(FM));
    FMmin=min(min(FM));
    fprintf('FMmin, FMmax: %f %f \n',FMmin,FMmax);
% convert magnitude only and phase only images to uint8 format for display
    FMS=im2uint8(mat2gray(FM));
    FNS=im2uint8(mat2gray(FN));
% display magnitude only image in second quadrant
    subplot(222),imshow(FMS, [0 25]);
    stitle=sprintf('image reconstructed from magnitude only');
    title(stitle);
% display original image in third quadrant;
% phase only image in fourth quadrant using the range [75 175] determined empiracally
    subplot(223),imshow(f,[  ]);
    subplot(224),imshow(FNS,[75  175]);
    stitle=sprintf('image reconstructed from phase only');
    title(stitle);
    figure,subplot(221),imshow(g,[   ]);
```

```
% determine magnitude and phase for the second image;
% convert to magnitude-only and phase-only images for the second image
    G=fft2(g);
    GA=abs(G);
    GP=atan2(imag(G),real(G));
    GB=exp(j*GP);
    GM=real(ifft2(GA));
    GN=real(ifft2(GB));
% determine maximum and minimum of magnitude-only second image and print
    GMmax=max(max(GM));
    GMmin=min(min(GM));
    fprintf('GMmin, GMmax: %f %f \n',GMmin,GMmax);
% convert magnitude-only and phase-only images to uint8 format
    GMS=im2uint8(mat2gray(GM));
    GNS=im2uint8(mat2gray(GN));
% plot original second image in first quadrant
% plot magnitude only second image in second quadrant
    figure; subplot(221),imshow(g,[  ]);
    subplot(222),imshow(GMS, [0 25]);
    stitle=sprintf('image reconstructed from magnitude only');
    title(stitle);
% plot original second image in third quadrant
% plot phase-only second image in fourth quadrant
    subplot(223),imshow(g,[  ]);
    subplot(224),imshow(GNS,[75  175]);
    stitle=sprintf('image reconstructed from phase only');
    title(stitle);
```

Figures 4 and 5 show plots of the original image (top and bottom left images) and the reconstructed magnitude-only (top right image) and phase-only (bottom right) images (for the two gray scale images). These plots show that the magnitude-only image is a washed out image with only the vaguest hints of the image details and character, whereas the phase-only reconstructed image retains the image outlines with some small degree of shading information. Clearly the phase-only reconstruction has far more signal-dependent information for an image than the magnitude-only reconstruction.

(b) The MATLAB code for crossing the magnitude and phase of the two images and reconstructing full images based on the crossed information is given below (note that the MATLAB code is added to the end of the previous code example):

```
% cross magnitudes and phases of the two images; convert to uint8 images
% and plot intensity histograms of the two crossed images
    F1=FA.*GB;  % GB
    G1=GA.*FB;  % FB
    g1=real(ifft2(G1));
    g1s=im2uint8(mat2gray(g1));
```

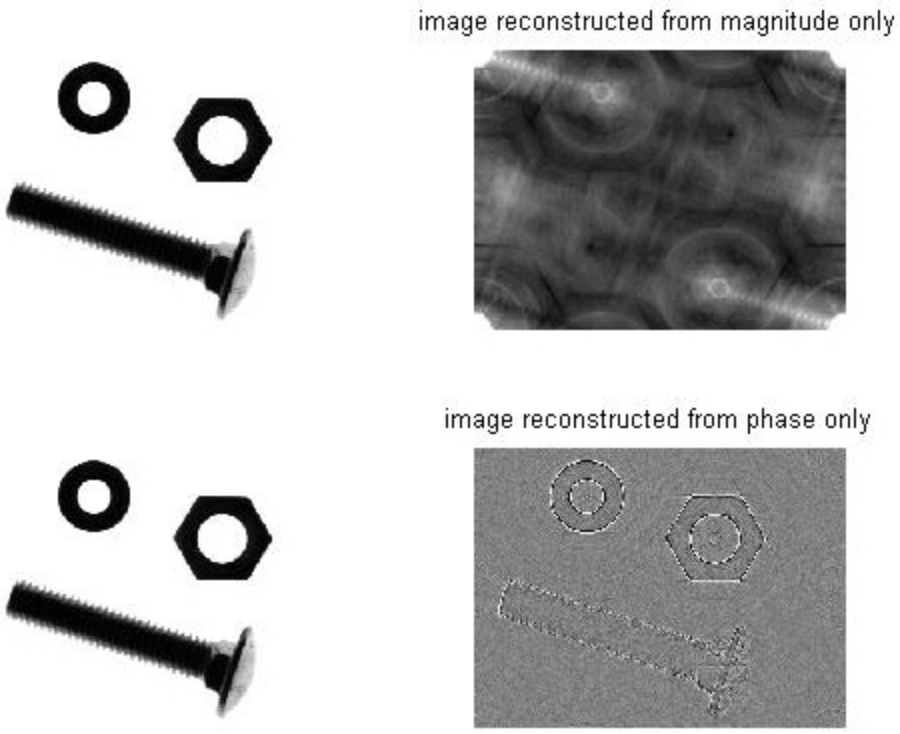

Figure 4: Plots of the original image (upper left image), the magnitude-only image (upper right image), the original image (lower left image), and the phase-only image (lower right image).

```
    figure,plot(imhist(g1s));
    f1=real(ifft2(F1));
    f1s=im2uint8(mat2gray(f1));
    figure,plot(imhist(f1s));
% plot original first image in first quadrant
    figure,subplot(221),imshow(g,[   ]);
% plot crossed first image (correct phase, incorrect magnitude) in second
% quadrant
    subplot(222),imshow(f1s, [ ]);
    stitle=sprintf('image magnitude different, phase correct');
    title(stitle);
% plot original second image in third quadrant
    subplot(223),imshow(f,[  ]);
% plot crossed second image (correct phase, incorrect magnitude) in fourth
% quadrant
    subplot(224),imshow(g1s,[ ]);
    stitle=sprintf('image magnitude different, phase correct');
    title(stitle);
```

Figure 6 shows the plots of the crossed-magnitudes for the two images. We see that the resulting images contain most of the picture information that describes the image elements,

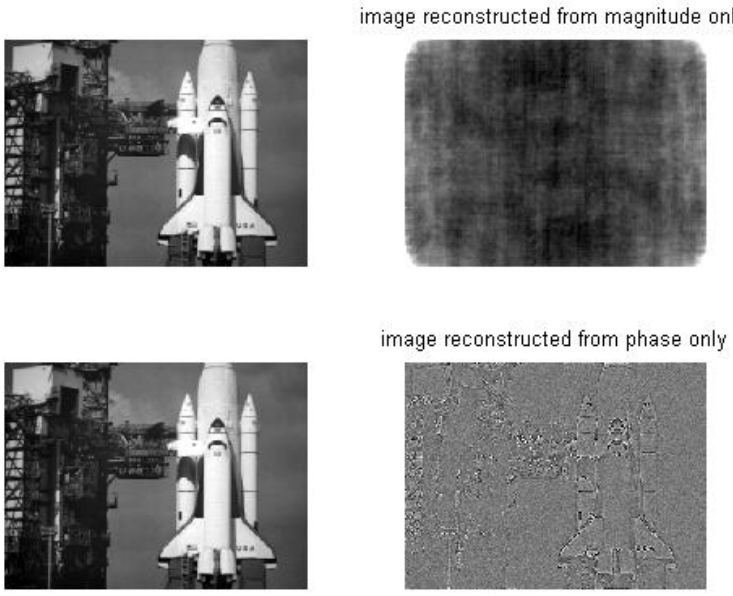

Figure 5: Plots of the original image (upper left image), the magnitude-only image (upper right image), the original image (lower left image), and the phase-only image (lower right image).

although with a fair amount of smearing and distortion due to the incorrect magnitude information. Clearly the phase information is far more important than the magnitude information in preserving key aspects of individual images. The phase information seems to retain a lot of information about image edges and image orientations.
****

4 Exercise 3 – Linear Filtering of Images

(a) Using the MATLAB routine `H=lpfilter(type, M, N, D0, n)` for designing lowpass filters, where the input arguments are:

```
type='gaussian', 'ideal' or 'btw' (for Butterworth filters)
M,N = dimensionality of filter frequency response
D0 = filter cutoff frequency (normalized to range [0, M] or [0
N])
n = filter order for Butterworth filters
```

and the output `H` is the frequency response of the filter, write a MATLAB m-file that accepts as input the filter type `(ftype='gaussian','ideal','btw')`, the cutoff frequency `D0`, and (optionally for the Butterworth filter) the filter order `n`. Within the MATLAB program,

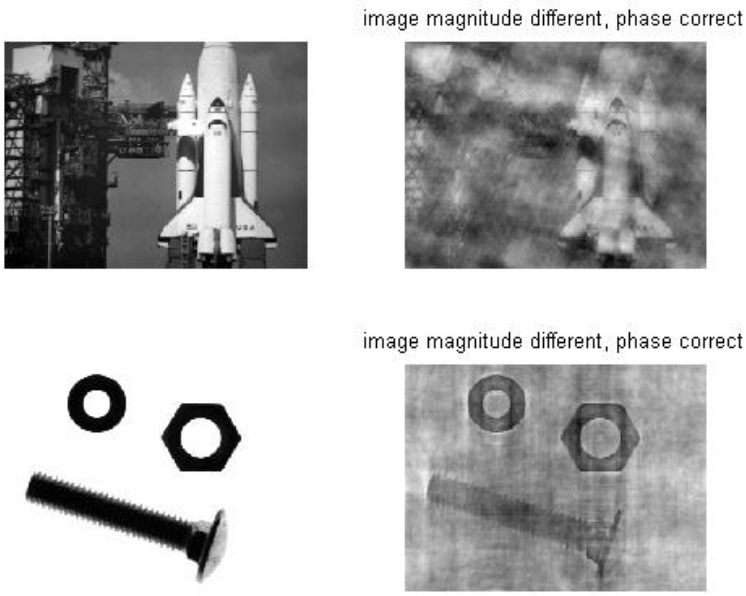

Figure 6: Plots of the original first image (upper left image), the cross-magnitude/correct phase image (upper right image), the original second image (lower left image), and the cross-magnitude/correct phase image (lower right image).

plot the filter frequency response (using the `mesh` command) along with the filter impulse response. Test your program for the following conditions:

1. `ftype='gaussian', D0=50`
2. `ftype='ideal', D0=50`
3. `ftype='btw', n=4, D0=50`

Include the graphics for the three frequency and impulse responses in your Laboratory report.

(b) How would you convert your MATLAB code to change the design from a lowpass filter to a highpass filter? Demonstrate this capability by designing a Gaussian highpass filter with cutoff `D0=50` and include the plot of the frequency and impulse response of the resulting filter.

(c) Using the original input gray-scale image, `lena.tif`, write another MATLAB m-file which reads in the original image (what is the image size), then design an appropriate lowpass filter with appropriate padding, filter the image in the frequency domain, convert back to the image domain and plot the processed image (remember to crop the filtered image so that you have the same region of support as the original image).
Compare the processed images with the three lowpass filters used in part (a) of this Exercise. Also compare the effects of differing values of `D0` between 20 and 200. What is the effect of lowpass filtering on the image?

(d) Include designs for highpass filters in your MATLAB code. Repeat part (c) using highpass filters in place of the lowpass filters. What is the effect of highpass filtering? What is the major difference between the three filter types?

****

SOLUTION

(a) The MATLAB code for choosing a lowpass filter type and designing the lowpass filter (using the MATLAB routine `lpfilter`) is as follows:

```
% display_filter_responses
%
% enter filter type and design parameters
    D0=50;  % cutoff frequency (range 0 to 250)
    D0=input('value for cutoff frequency (range 0 to 250):');
    ftype=input('filter type (gaussian, ideal, btw (butterworth)):','s');
    if (ftype(1:3) == 'gau')
        H0=lpfilter('gaussian',500,500,D0);
        stitle1=sprintf('gaussian filter response with D0= %d',D0);
        stitle2=sprintf('gaussian impulse response with D0= %d',D0);
    elseif (ftype(1:3) == 'ide')
        H0=lpfilter('ideal',500,500,D0);
        stitle1=sprintf('ideal filter response with D0= %d',D0);
        stitle2=sprintf('ideal impulse response with D0= %d',D0);
    elseif (ftype(1:3) == 'btw')
        n=input('butterworth filter order (1 to 6):');
        H0=lpfilter('btw',500,500,D0,n);
        stitle1=sprintf('butterworth filter response with D0= %d, n= %d',D0,n);
        stitle2=sprintf('butterworth impulse response with D0= %d, n= %d',D0,n);
    else
        error('improper filter type specified');
    end
    ilphp=input('lowpass (1) or highpass (0) filter:');
    if (ilphp == 0) H0=1-H0;
    end

% display frequency response of lowpass filter using mesh plot
    H=fftshift(H0);
    figure,mesh(H(1:10:500, 1:10:500));
    axis([0 50 0 50 0 1]);
    title(stitle1);

% convert from spectral to spatial domain
    h=real(ifft2(H));
    hmax=max(max(h));
    h=h/hmax;
    hc=fftshift(h);
```

```
% determine region of support for impulse response
    hsize=size(hc);
    middle=hsize(1)/2;
    hmax=max(max(hc));
    h1=find(hc(middle,:)>hmax/100.);
    startpt=max(h1(1)-10,1);
    endpt=min(h1(end)+10,hsize(1));

% display impulse response of lowpass filter using mesh plot
    figure,mesh(hc(1:10:500, 1:10:500));
    axis([0 50 0 50 0 1]);
    title(stitle2);

% display response on finer scale
    figure,mesh(hc(225:2:275, 225:2:275));
    axis([0 25 0 25 0 1]),title(stitle2);
    figure,mesh(hc(startpt:1:endpt, startpt:1:endpt));
    axis([0 (endpt-startpt+1) 0 (endpt-startpt+1) 0 1]),title(stitle2);

% display impulse response at center spatial line
    figure,plot(hc(middle,startpt:endpt),'b','LineWidth',2);
    axis tight,title(stitle2);
```

Figure 7 shows plots of the frequency response of three lowpass filters (designed using the above code), namely:

1. a Gaussian lowpass filter with parameters `M=N=500`, `D0=50`
2. an ideal lowpass filter with parameters `M=N=500`, `D0=50`
3. a Butterworth lowpass filter with parameters `M=N=500`, `D0=50, n=4`

The differences in the shape and smoothness of the three lowpass filters are clear in this figure. Plots of both the frequency and impulse response (with 3 different resolutions) of the Gaussian lowpass filter are given in Figure 8 below. The width of the impulse response is inversely proportional to the width of the frequency response; hence for the wide frequency response of the Gaussian filter, we see a fairly narrow response region for the Gaussian impulse response.

A similar set of plots of the frequency and impulse response of the ideal lowpass filter and the Butterworth lowpass filter are shown in Figure 9 (for the ideal lowpass filter) and in Figure 10 (for the Butterworth lowpass filter).

(b) To convert the design from a lowpass filter to a highpass filter, you merely subtract the lowpass response from 1 (this is already reflected in the MATLAB code above). A plot of the frequency and impulse responses of a Gaussian highpass filter is given in Figure 11 below.

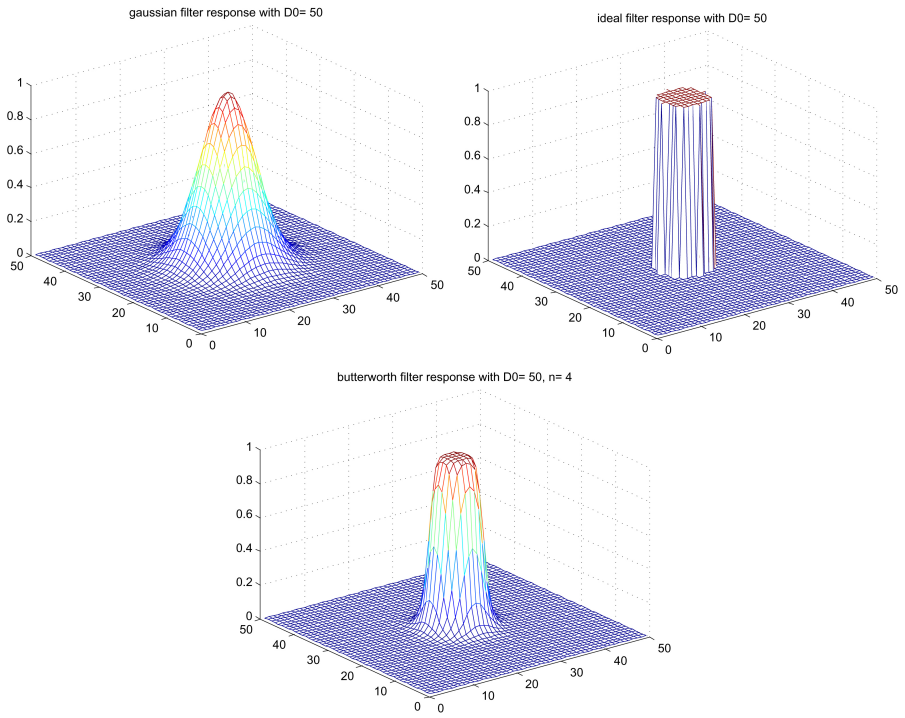

Figure 7: Plots of the frequency response of a Gaussian lowpass filter (top left), an ideal lowpass filter (top right) and a 4-th order Butterworth lowpass filter (bottom plot).

(c) We can now use the filters designed above for filtering an image file. We use the file `lena.tif` for this exercise. The MATLAB code for choosing the appropriate lowpass filter and filtering an image file (in the frequency domain) is given in the MATLAB code below:

```
% lp_hp_filter_image
%
% read in tif image lena and size the image
    f=imread('lena.tif','tiff');
    [M,N]=size(f);

% pad image for convolution
    PQ=paddedsize(size(f));

% enter filter type and design parameters
    D0=50;  % cutoff frequency (range 0 to 250)
    D0=input('value for cutoff frequency (range 0 to 250):');
    ftype=input('filter type (gaussian, ideal, btw (butterworth)):','s');
    if (ftype(1:3) == 'gau')
        Hp=lpfilter('gaussian',PQ(1),PQ(2),D0);
    elseif (ftype(1:3) == 'ide')
        Hp=lpfilter('ideal',PQ(1),PQ(2),D0);
    elseif (ftype(1:3) == 'btw')
```

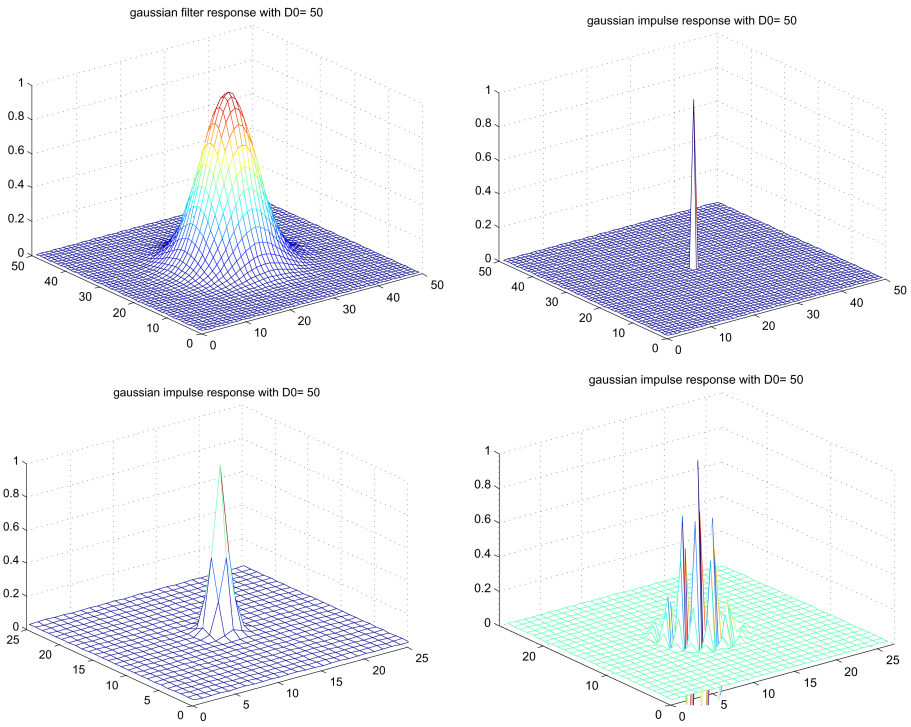

Figure 8: Plots of the frequency response of a Gaussian lowpass filter (top left), the impulse response of the filter (top right), and two different blow ups of the impulse response (bottom plots).

```
        n=input('butterworth filter order (1 to 6):');
        Hp=lpfilter('btw',PQ(1),PQ(2),D0,n);
    else
        error('improper filter type specified');
    end
    ilphp=input('lowpass (1) or highpass (0) filter:');
    if (ilphp == 0) Hp=1-Hp;
    end

% transform image to frequency domain using zero-padded image
    Fp=fft2(f,PQ(1),PQ(2));

% display impulse response of lowpass filter using mesh plot
    H=fftshift(Hp);
    figure,mesh(H(1:10:PQ(1), 1:10:PQ(2)));
    axis([0 round(PQ(1)/10) 0 round(PQ(2)/10) 0 1]);

% filter in frequency domain by multiplying 2-D FFTs
    Gp=Hp.*Fp;
```

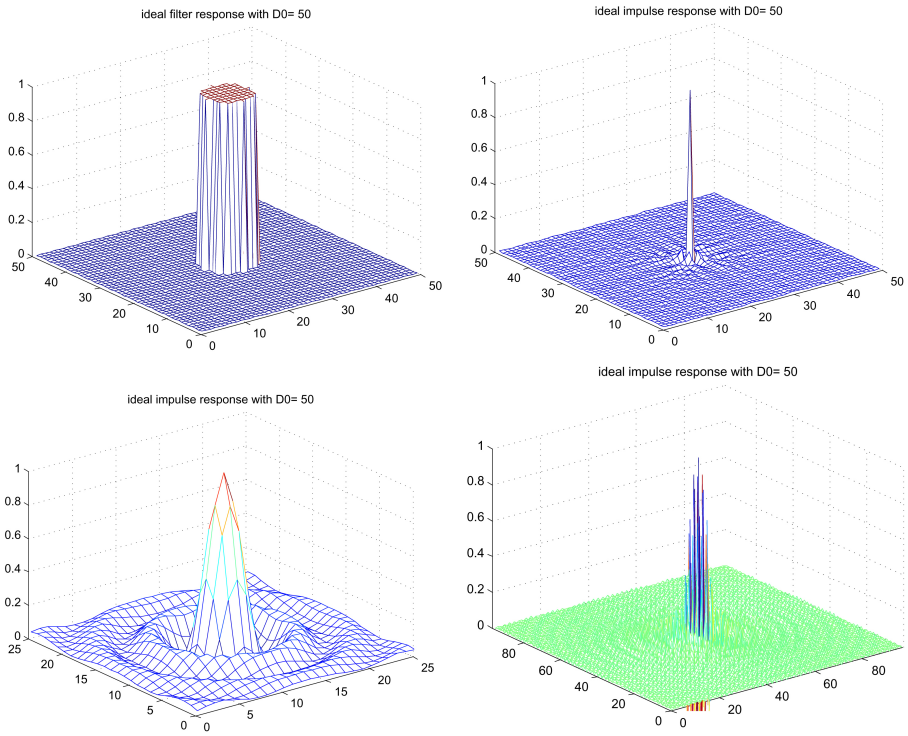

Figure 9: Plots of the frequency response of an ideal lowpass filter (top left), the impulse response of the filter (top right), and two different blow ups of the impulse response (bottom plots).

```
% convert back to image plane, using real value of inverse transform
    gp=real(ifft2(Gp));

% crop image to original size and display uncropped and cropped image
    gpc=gp(1:size(f,1),1:size(f,2));
    figure,imshow(gp,[ ]);
    figure,imshow(gpc,[ ]);
```

Figure 12 below shows the effects of varying the cutoff frequency of the Gaussian lowpass filter on the resulting image. The cutoff frequencies for the four sub-images are 20 (top left image), 50 (top right image), 100 (bottom left image) and 200 (bottom right image) where the cutoff frequencies are relative to the number of pixels in the image-512 in this case. It is clearly seen that narrow band lowpass filters smear the image a great deal and make it very fuzzy, whereas wideband lowpass filters have no clear detrimental effect on the image quality.

Figure 13 below shows a comparison of the effects of three lowpass filters (all with the same cutoff frequency) on the lena image. The upper left plot is the original image; the upper right plot is the one from a Gaussian filter with a cutoff frequency of 50; the lower left plot is the one from an ideal lowpass filter with a cutoff frequency of 50; the lower right plot is the one from a Butterworth lowpass filter of order 4, with cutoff frequency 50. The amount of aliasing from the ideal lowpass filter is clear in this figure.

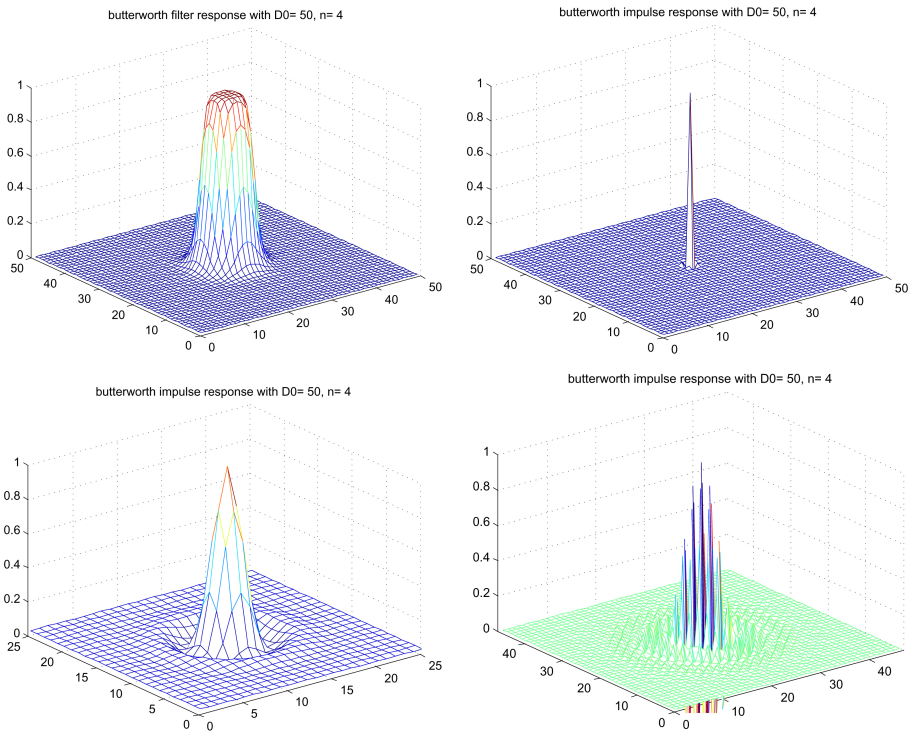

Figure 10: Plots of the frequency response of a Butterworth lowpass filter (top left), the impulse response of the filter (top right), and two different blow ups of the impulse response (bottom plots).

(d) It is trivial to extend the MATLAB code to include highpass filters (in fact the above code already has a lowpass/highpass option). To illustrate the effects of highpass filtering on the lena image, Figure 14 shows the results of Gaussian highpass filtering using cutoff frequencies of `D0=20` (upper left plot), 50 (upper right plot), 100 (lower left plot) and 200 (lower right plot). It can clearly be seen that for low cutoff frequencies (e.g., `D0=20`) the effect of the wide highpass filter is to preserve a shadow-like version of the original image; whereas as the cutoff frequency becomes larger and larger, the contrast becomes less and less and the image blends into a gray scale outline.

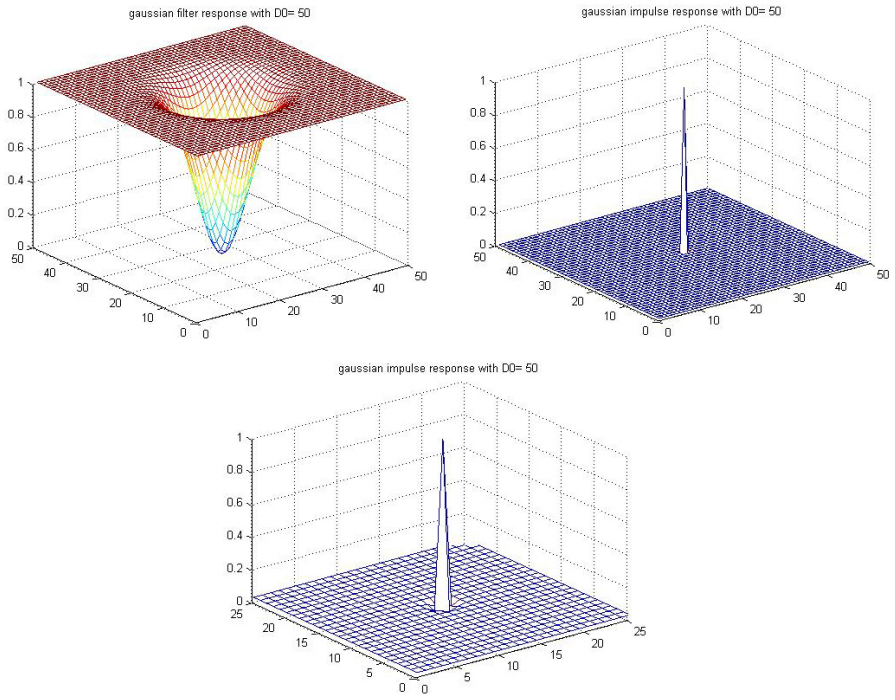

Figure 11: Plots of the frequency response of a Gaussian highpass filter (top left), the impulse response of the filter (top right), and a blow up of the impulse response (bottom plot).

Figure 12: Plots of the lena image subjected to lowpass filtering using a Gaussian lowpass filter with cutoff frequencies of 20 (top left image), 50 (top right image), 100 (bottom left image) and 200 (bottom right image).

Figure 13: Plots of the original lena image (top left image) along with lowpass filtered versions using a Gaussian filter (top right image), an ideal filter (bottom left image) and a Butterworth 4-th order filter (lower right image). The cutoff frequencies for all lowpass filters was 50.

Figure 14: Plots of the lena image subjected to higpass filtering using a Gaussian highpass filter with cutoff frequencies of 20 (top left image), 50 (top right image), 100 (bottom left image) and 200 (bottom right image).