## Contents

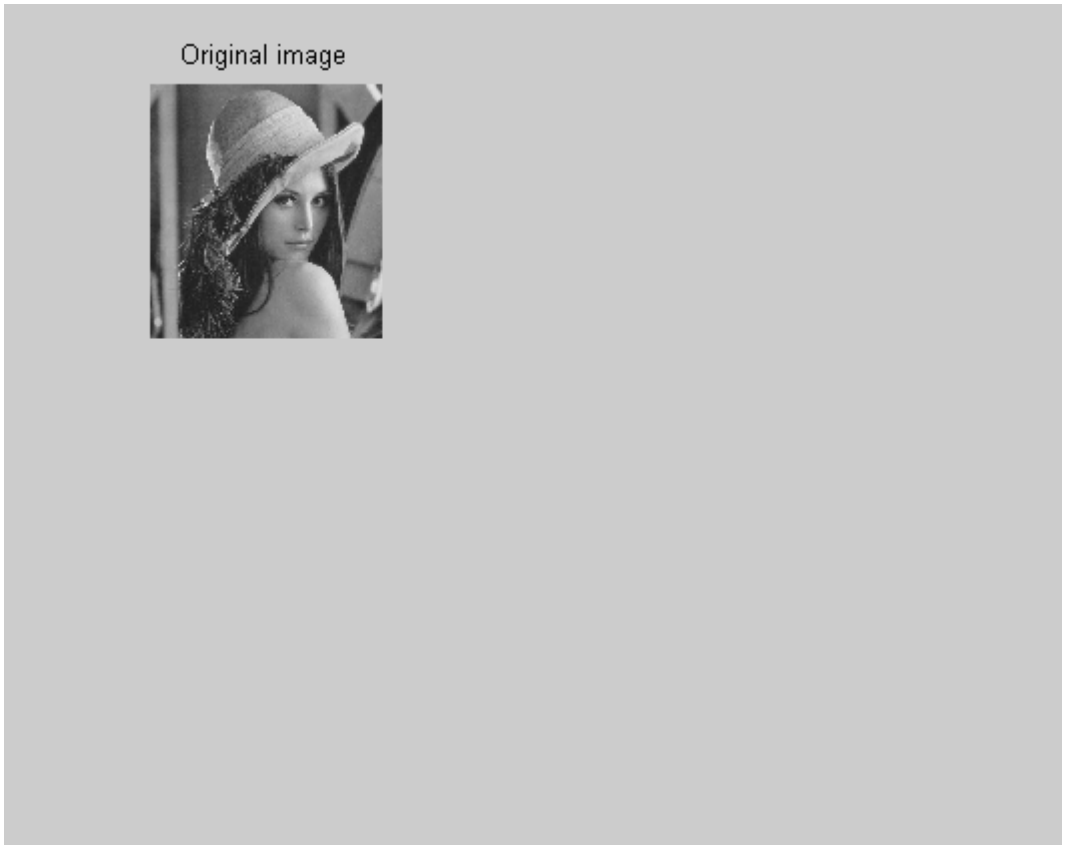

- Image processing template
- Reading and showing the image
- Discrete Fourier transform of the image
- Defining the filter in frequency domain. (Assignment)
- Shifting the filter
- Applying the filter
- Inverse Fourier transform


```
%PHYS 258 SJSU S2010 Nayer Eradat
```

## Image processing template

## Reading and showing the image

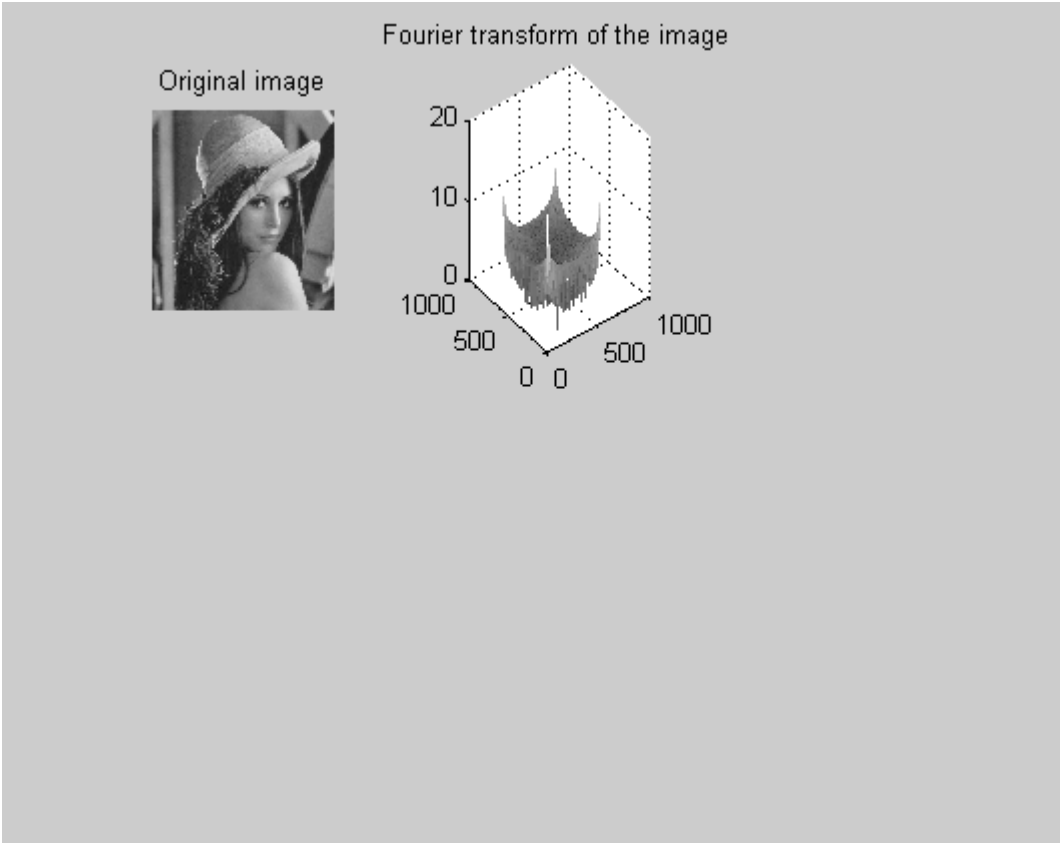
```
clear
im_read = imread ( 'lenabw.jpg'); % read input file
fig = figure;
%get dimensions of the image matrix third dimension is the color
[x y z]=size(im_read);
%Select only one color or first layer for black and white images
%Make a 2D array out of the image file
im1 = im_read(:,:,1);
%Size of the image array
siz = [x,y];
%Divide the screen to 6 parts and show the image on the first part
subplot(2,3,1);
imshow(im1);
title('Original image');
```




### Discrete Fourier transform of the image

take the discrete Fourier transform of the im1 file and store the spatial frequency values in im

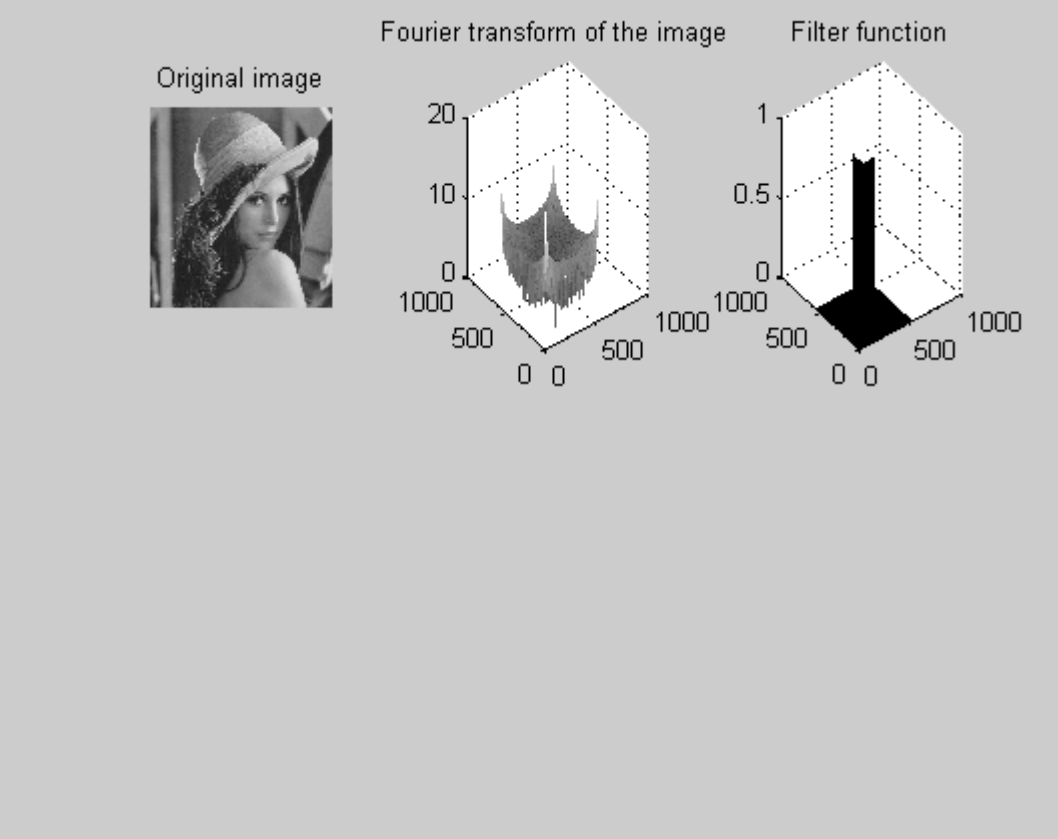
```
im1_f = (fftn(double(im1),siz));
subplot(2,3,2);
mesh(log(im1_f)); %3D plot in log scale that enhances th features
title('Fourier transform of the image' );
%save the data
%print ('-djpeg', 'lenabw-f.jpg', '-r0');
```




**Defining the filter in frequency domain. (Assignment)**

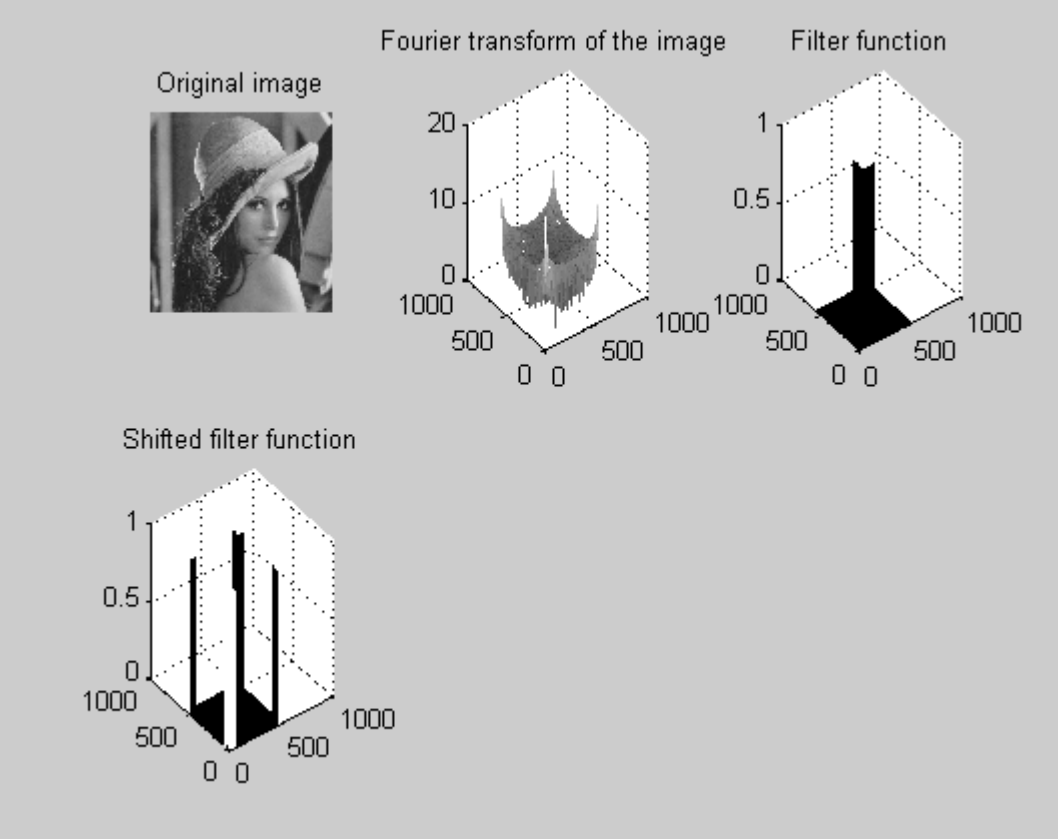
```
%You may have a high-pass,low-pass, or bandpass filter. Have an est
%of the frequency content of the image and use your knowledge of th
%Fourier optics to make filters for sharpening, smoothing, edge det
%increasing / decreasing the contrast, and compressing a picture.

%Here is an example filter for smoothing
ax = 10; %coefficient to define size of the filter in x
ay = 10; %coefficient to define size of the filter in y
%This defines a rectangular finction in the middle of the matrix wi
% 1/15 dimention of the frequency matrix
for i = 1:1:siz(1,1);
    for j= 1:1:siz(1,2)
     if abs(i-siz(1,1)/2) < siz(1,1)/ax && abs(j-siz(1,2)/2) < siz(
          filter1(i,j) = 1;
     else
          filter1(i,j) = 0;
     end
    end
end
```

```
subplot(2,3,3);
mesh(filter1); %3D plot of the filter
title('Filter function');
```




**Shifting the filter**

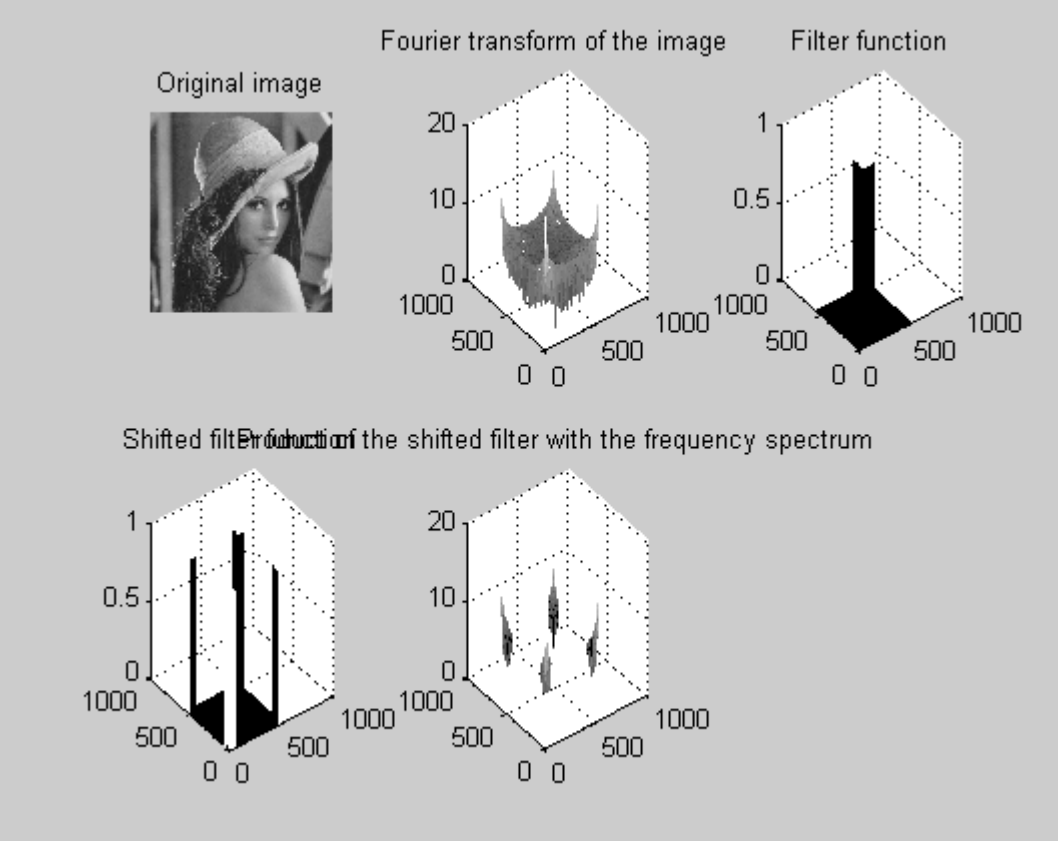
```
%We shift 1/4 of the rectangle to each corner (see the fftshift com
%When we took the fftn of the image the low-frequency content was
%accumulated on the corners so to generate a low-pass filter we are
%allowing the corners to pass.
filter1_shift =fftshift(filter1);
subplot(2,3,4);
mesh(filter1_shift); %3D plot of the shifted filter
title('Shifted filter function');
```




### Applying the filter

```
%Element-by-element multiplication of the filter and the frequency
im1_filter1_f = (im1_f.*filter1_shift);
%im1_filter_f = (im1_f.*filter);
subplot(2,3,5);
mesh(log(im1_filter1_f)); %Log scale 3D plot of the filter * freq s
title('Product of the shifted filter with the frequency spectrum' )
```

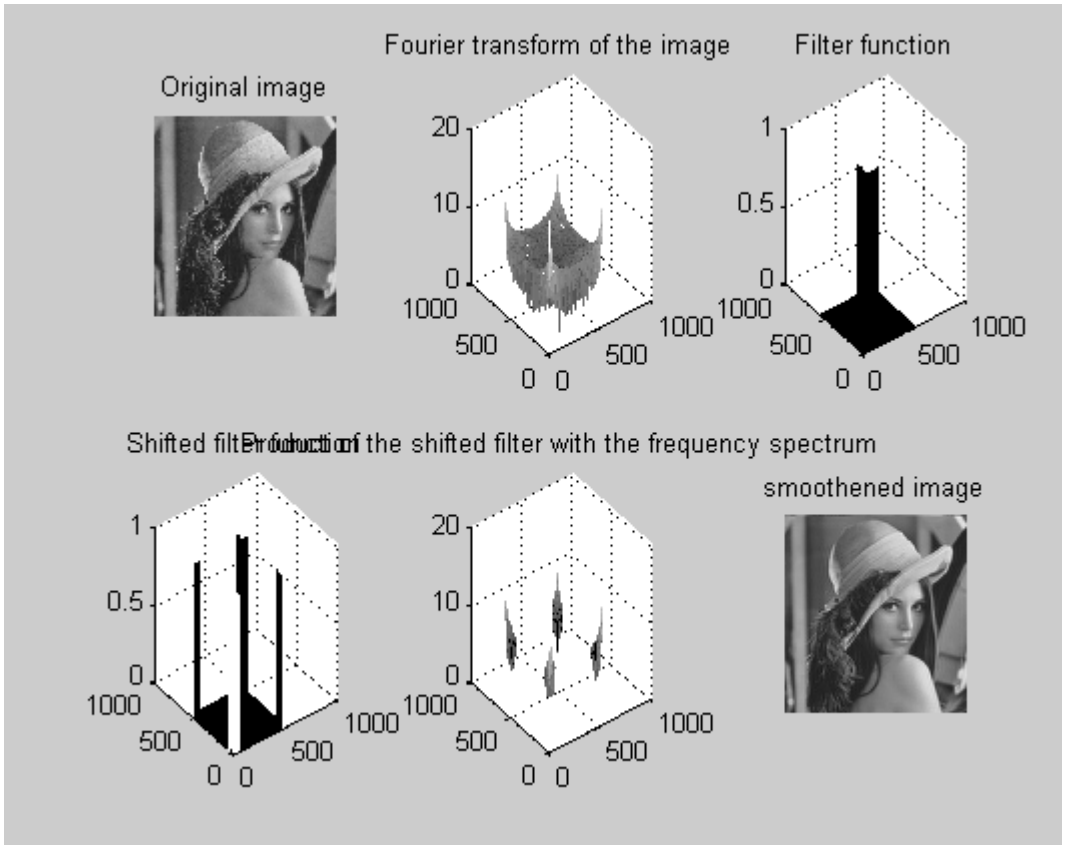
```
Warning: Log of zero.
```




### Inverse Fourier transform

```
%Now we take the inverse fourier transform to recover the filtered
im1_filter1_ff = ifftn(double(im1_filter1_f),siz);
subplot(2,3,6);
imshow(uint8(im1_filter1_ff));  %Showing the image. This function
%does accepr unsigned integers
title('smoothened image');
```

```
Warning: Displaying real part of complex input.
```


Original image
Fourier transform of the image
Filter function
Shifted filter function
Product of the shifted filter with the frequency spectrum
smoothened image
20
10
0
1000
500
0
1
0.5

*Published*