

National University of Science and Technology
School of Electrical Engineering and Computer Science

Department of Software Engineering

EE433: Digital Image Processing

Class: BESE-5

Lab 8: Frequency Domain Filtering Techniques

Date: 7th November, 2017

Time: 2pm-5pm

Instructor: Dr. Muhammad Moazam Fraz

Lab Engineer: Ms Iram Tariq Bhatti

Course Learning Outcomes (CLOs)			
Upon completion of the course, students should demonstrate the ability to:		PLO Mapping	BT Level
CLO 1	Understanding the fundamentals and basic concepts of image processing	PLO 1	C2
CLO 2	Analyze images using mathematical transformations and operations	PLO 2	C4
CLO 3	Develop solutions by using modern tools to solve practical problems.	PLO 5	C5
	* BT= Bloom's Taxonomy, C=Cognitive domain, P=Psychomotor domain, A= Affective domain		

Table of Contents

Learning Outcome	3
Goal	3
Objectives	3
Tools/Software Requirement	3
1. Discrete Fourier Transform	3
1.1 How to Display a Fourier Spectrum using MATLAB	4
2. Frequency Domain Versions of Spatial Filters	6
2.1 Basic Steps in DFT Filtering	7
3. Frequency Domain Specific Filters	8
3.1 Low pass Filters	8
3.2 High pass Filters	10
3.3 Notch Filters	13
4. Exercises	14
Task A) Identifying and Using High and Low Pass Filters.	14
Task B) Load the cameraman image, convert it to double, and generate its FT.	15
Task C) Take any image and apply Gaussian high pass filter and Gaussian low pass filter on it.	15
Task D) CSI Style Image Enhancement (for bonus marks)?	15
Deliverables	17

Learning Outcome

CLO 2: Analyze images using mathematical transformations and operations.

Goal

The goal of this lab is to learn how to compute and display the Fourier Transform (FT) of an image and how to develop filters to be used in the frequency domain.

Objectives

- Learn how to use the `fft2` function to compute the FT of a monochrome image.
- Learn how to visualize the FT results.
- Learn how to generate filters to be used in the frequency domain.

Tools/Software Requirement

MATLAB 201x / Python 2 or 3 with associated libraries.

1. Discrete Fourier Transform

The concept behind the Fourier transform is that any waveform can be constructed using a sum of sine and cosine waves of different frequencies. The general idea is that the image ($f(x,y)$) of size ($M \times N$) will be represented in the frequency domain ($F(u,v)$). The equation for the two-dimensional discrete Fourier transform (DFT) is:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

The exponential in the above formula can be expanded into sines 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:

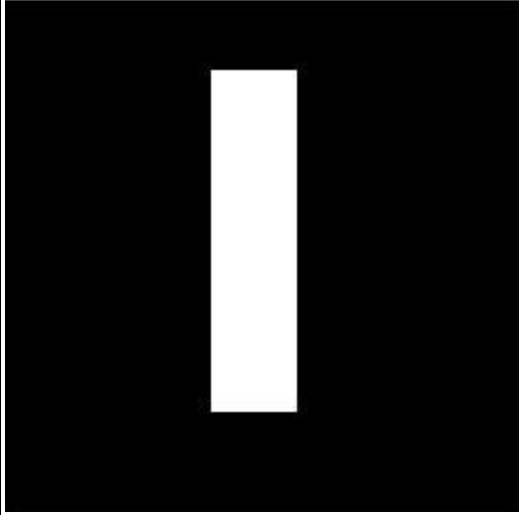
$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

- We visually analyze a Fourier transform by computing a **Fourier spectrum**.

- 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 related functions that compute the inverse DFT:
 1. `ifft`
 2. `ifft2`
 3. `ifftn`

1.1 How to Display a Fourier Spectrum using MATLAB

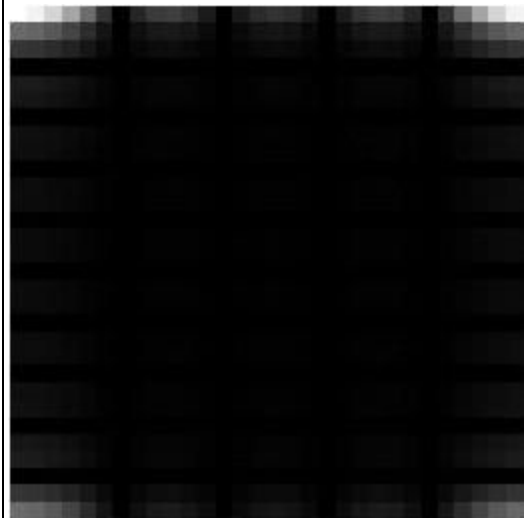
The following table is meant to describe the various steps behind displaying the Fourier Spectrum.

MATLAB code	Image Produced
<pre> %Create a black 30x30 image f=zeros(30,30); %With a white rectangle in it. f(5:24,13:17)=1; imshow(f,'InitialMagnification','fit') </pre>	

```
%Calculate the DFT.
F=fft2(f);

%There are real and imaginary parts to F.
%Use the abs function to compute the
magnitude
%of the combined components.
F2=abs(F);

figure, imshow(F2,[],
'InitialMagnification','fit')
```



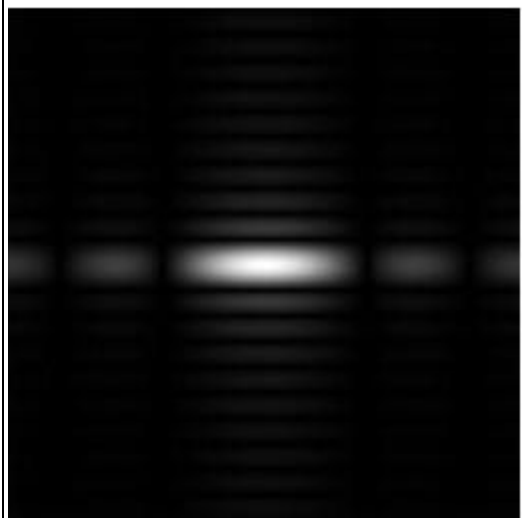
```
%To create a finer sampling of the Fourier
transform,
%you can add zero padding to f when
computing its DFT
%Also note that we use a power of 2, 2^256
%This is because the FFT -Fast Fourier
Transform -
%is fastest when the image size has many
factors.
F=fft2(f, 256, 256);

F2=abs(F);
figure, imshow(F2, [])
```



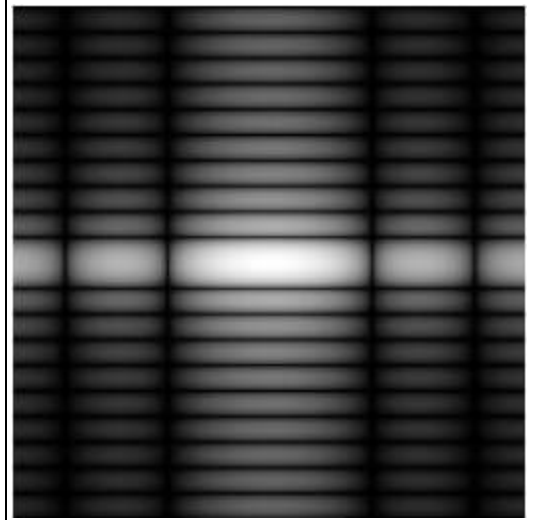
```
%The zero-frequency coefficient is displayed
in the upper left hand corner. To display it
in the center,
%you can use the function fftshift.
F2=fftshift(F);

F2=abs(F2);
figure, imshow(F2, [])
```



```
%In Fourier transforms, high peaks are so
high they
%hide details. Reduce contrast with the log
function.
F2=log(1+F2);

figure,imshow(F2,[])
```



- ❖ To get the results shown in the last image of the table, you can also combine MATLAB calls as in:

```
f=zeros(30,30);
f(5:24,13:17)=1;
F=fft2(f, 256,256);
F2=fftshift(F);
figure,imshow(log(1+abs(F2)),[])
```

Notice in these calls to `imshow`, the second argument is empty square brackets. This maps the minimum value in the image to black and the maximum value in the image to white.

2. Frequency Domain Versions of Spatial Filters

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

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

and, conversely,

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

where 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. This means we can perform linear spatial filters as a simple component-wise multiply in the frequency domain.

This suggests that we could use Fourier transforms to speed up spatial filters. This only works for large images that are correctly padded, where multiple transformations are applied in the frequency domain before moving back to the spatial domain.

When applying Fourier transforms padding is very important. Note that, because images are infinitely tiled in the frequency domain, filtering produces wraparound artefacts if you don't zero pad the image to a larger size. The `paddedsz` function below calculates a correct padding size to avoid this problem. The `paddedsz` function can also help optimize the performance of the DFT by providing power of 2 padding sizes. See `paddedsz`'s help header for details on how to do this.

2.1 Basic Steps in DFT Filtering

The following summarize the basic steps in DFT Filtering:

1. Obtain the padding parameters using function `paddedsz`:

```
PQ=paddedsz(size(f));
```

2. Obtain the Fourier transform of the image with padding:

```
F=fft2(f, PQ(1), PQ(2));
```

3. Generate a filter function, `H`, the same size as the image

4. Multiply the transformed image by the filter:

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

5. Obtain the real part of the inverse FFT of `G`:

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

6. Crop the top, left rectangle to the original size:

```
g=g(1:size(f, 1), 1:size(f, 2));
```

3. Frequency Domain Specific Filters

There are three commonly discussed filters in the frequency domain:

- ❖ Low pass filters, sometimes known as smoothing filters
- ❖ High pass filters, sometimes known as sharpening filters
- ❖ Notch filters, sometimes known as band-stop filters

3.1 Low pass Filters

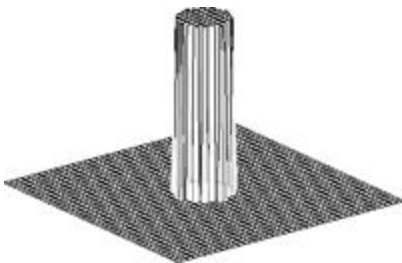
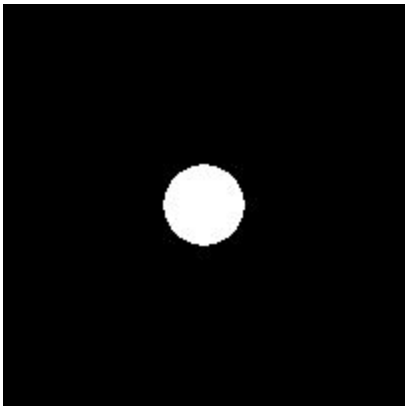
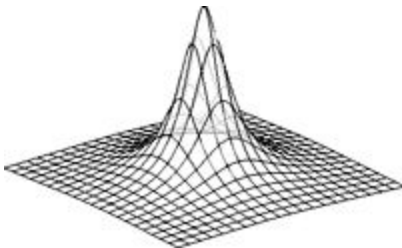
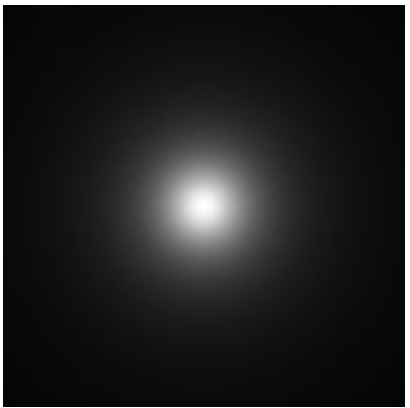
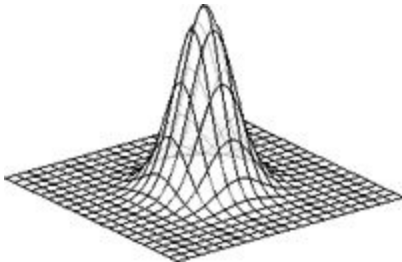

Low pass filters:

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

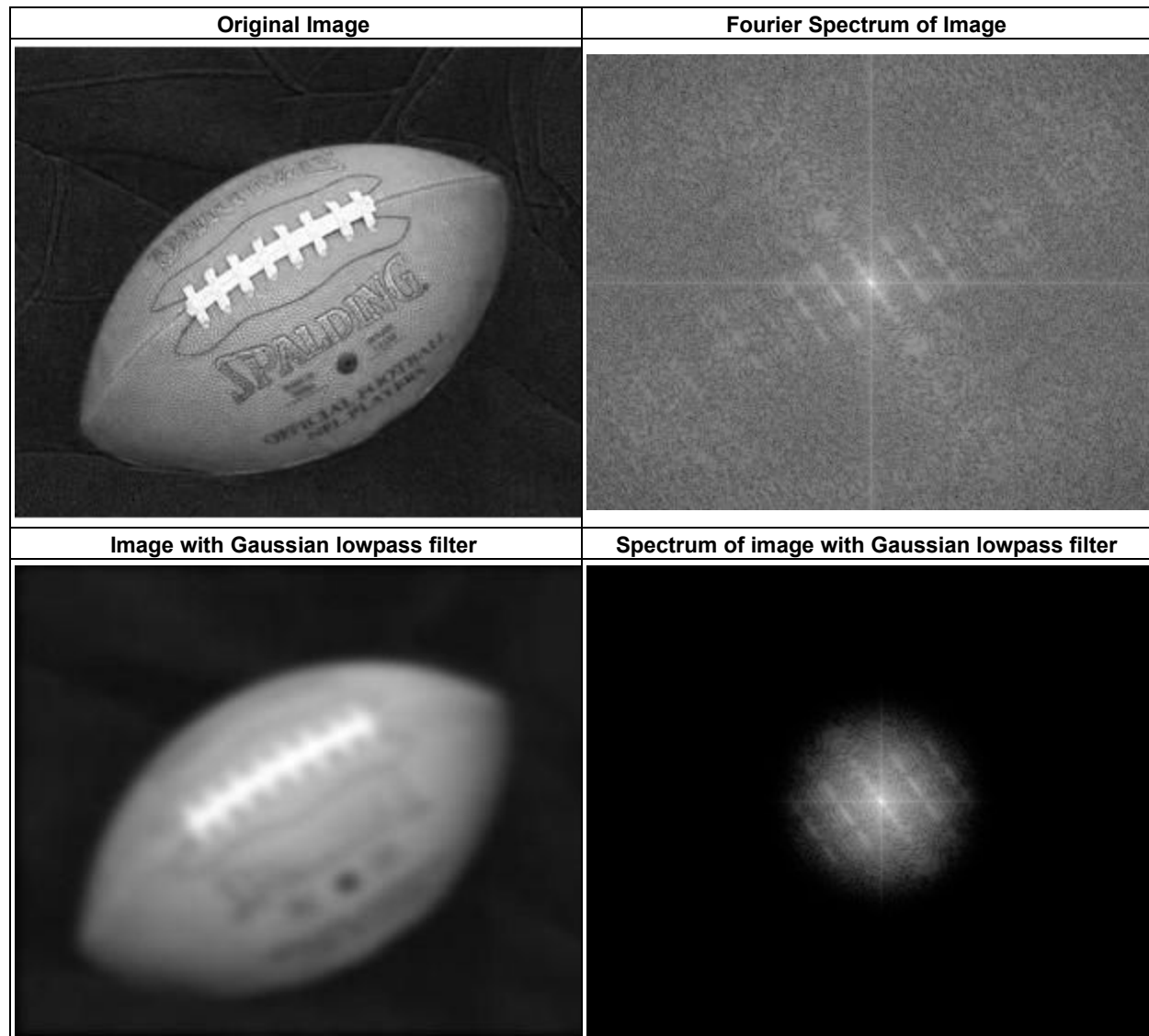
Three main low pass filters are discussed in this lab:

1. ideal low pass filter (ILPF)
2. Butterworth low pass filter (BLPF)
3. Gaussian low pass filter (GLPF)

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. $D(u,v)$ is the distance from point (u,v) to the center of the filter.

Low pass Filter	Mesh	Image
<p>Ideal:</p> $H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$		
<p>Butterworth:</p> $H(u, v) = \frac{1}{1 + [D(u, v) / D_0]^{2n}}$		
<p>Gaussian:</p> $H(u, v) = e^{-D^2(u, v) / 2D_0^2}$		

The following is the result of applying a Gaussian low pass filter on an image.



3.2 High pass Filters



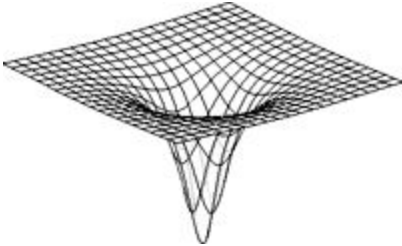
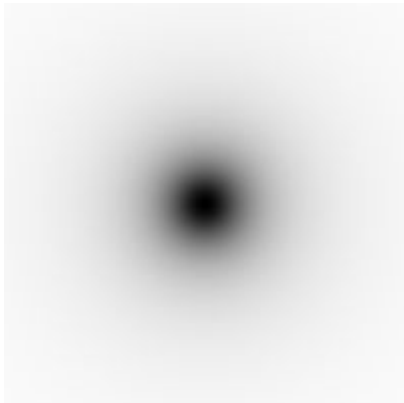
High pass filters:

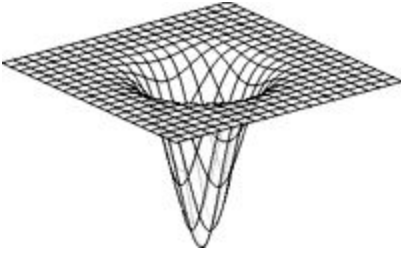
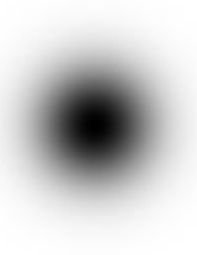
- sharpen (or shows the edges of) an image
- attenuate the low frequencies and leave the high frequencies of the Fourier transform relatively unchanged

The high pass filter (H_{hp}) is often represented by its relationship to the low pass filter (H_{lp}):

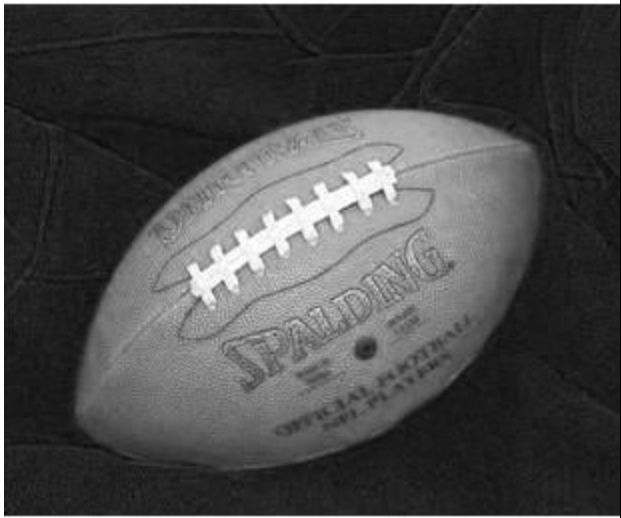
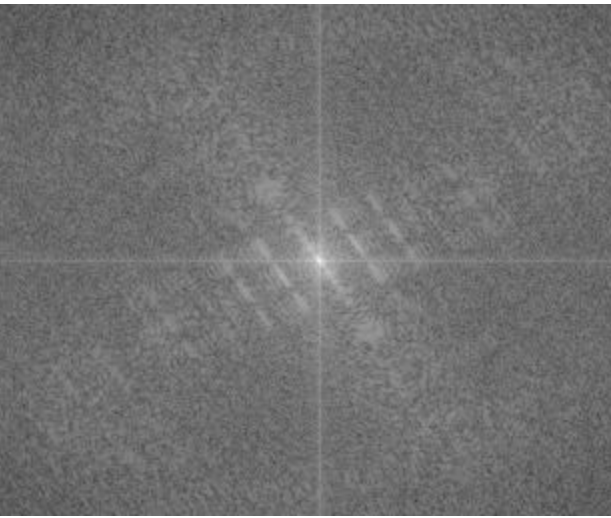
$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

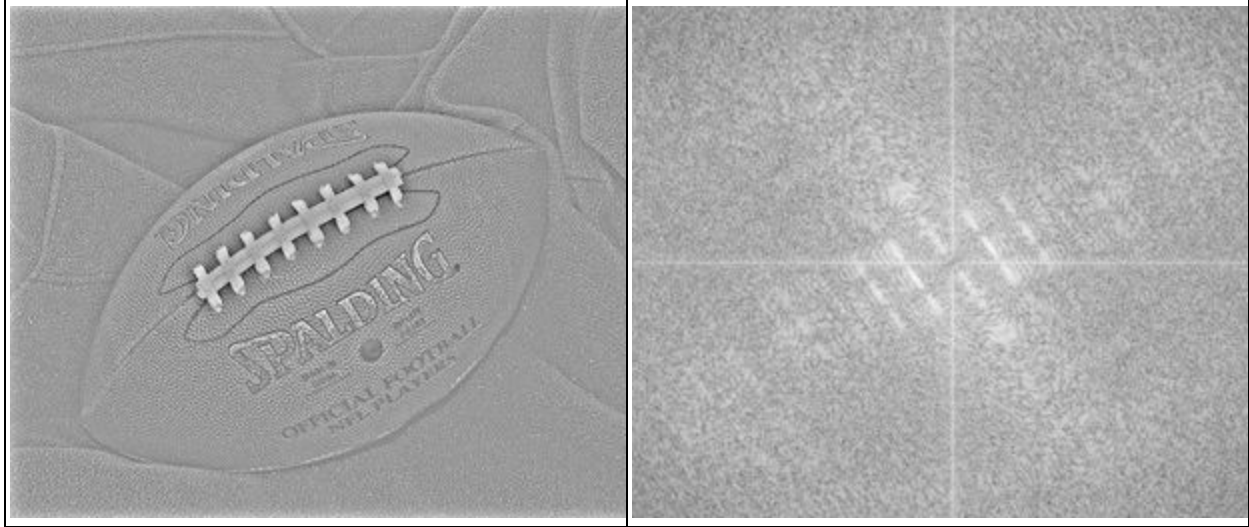
Because high pass filters can be created in relationship to low pass filters, the following table shows the three corresponding high pass filters by their visual representations:

High pass Filter	Mesh	Image
Ideal		
Butterworth		

Gaussian		
----------	---	--

The following is the result of applying a Gaussian high pass filter on an image.

Original Image	Fourier Spectrum of Image
	
Image with Gaussian highpass filter	Spectrum of image with Gaussian highpass filter



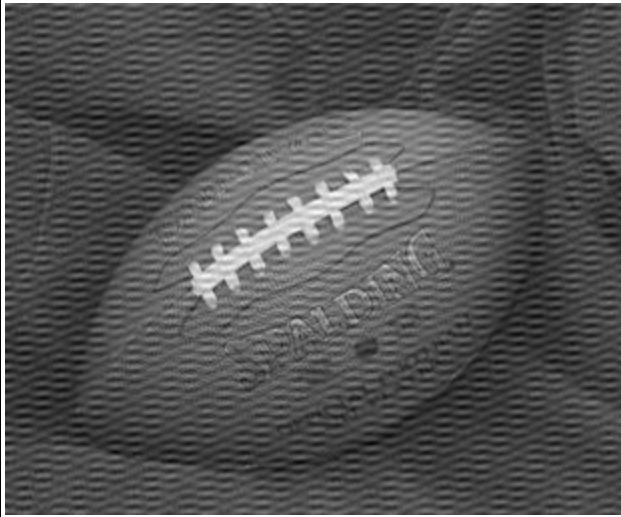
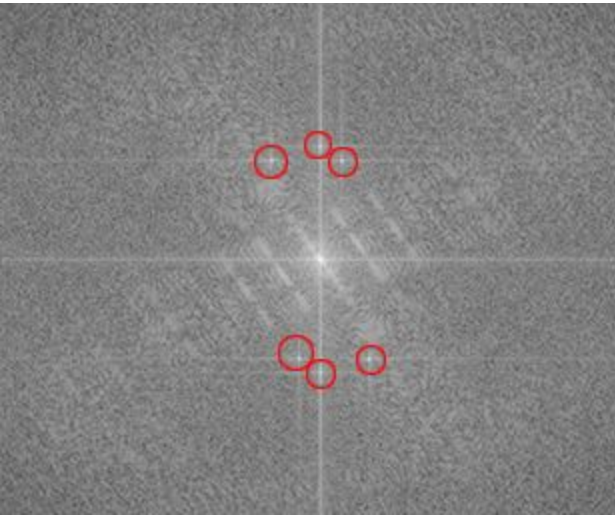
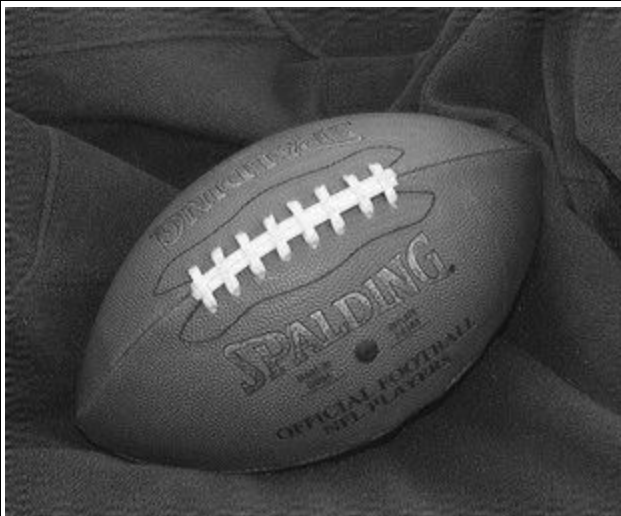
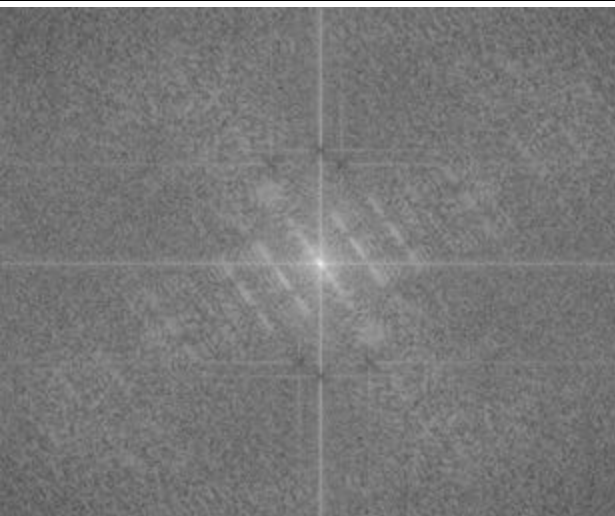
3.3 Notch Filters

Notch filters:

- are used to remove repetitive "Spectral" noise from an image
- are like a narrow high pass filter, but they "notch" out frequencies other than the dc component
- attenuate a selected frequency (and some of its neighbors) and leave other frequencies of the Fourier transform relatively unchanged

Repetitive noise in an image is sometimes seen as a bright peak somewhere other than the origin. You can suppress such noise effectively by carefully erasing the peaks. One way to do this is to use a notch filter to simply remove that frequency from the picture. This technique is very common in sound signal processing where it is used to remove mechanical or electronic hum, such as the 60Hz hum from AC power. Although it is possible to create notch filters for common noise patterns, in general notch filtering is an ad hoc procedure requiring a human expert to determine what frequencies need to be removed to clean up the signal.

The following is an example of removing synthetic spectral "noise" from an image.

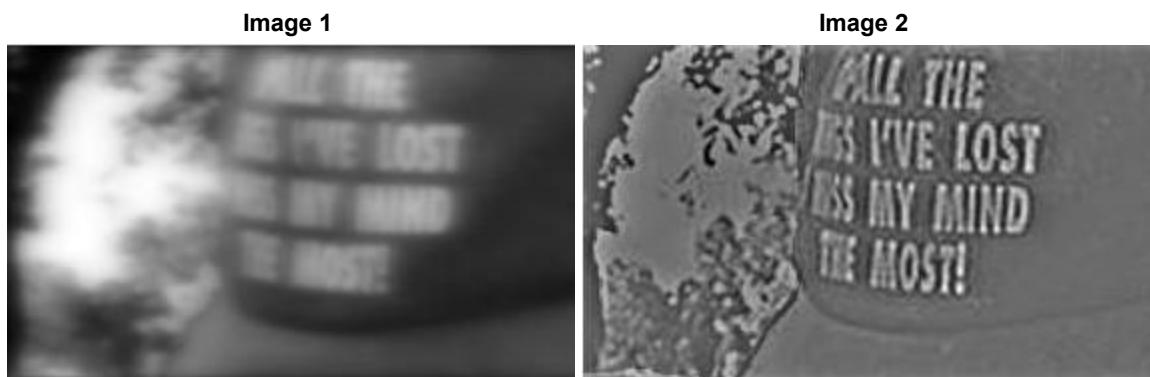
Noisy Image	Fourier Spectrum of Image (noise peaks circled)
	
Image after Butterworth notch filters	Spectrum of image after Butterworth notch filters
	

4. Exercises

Task A) Identifying and Using High and Low Pass Filters.

1. Download the following image "[Cap.jpg](#)" attached with lab; and store it in MATLAB's "Current Directory". Identify which of the following Image1

and Image2 is the result of a low pass or high pass Butterworth filter and reproduce the results.



2. Reproduced high pass and low pass filter for **Cap.jpg**
3. Display the Fourier spectrum for Cap.jpg.

Task B) Load the cameraman image, convert it to double, and generate its FT.

Question 1 What are the minimum and maximum values of the resulting discrete Fourier transform coefficients for the cameraman image?

Question 2 Compare the ideal-filtered FT image and the Butterworth-filtered FT image.

Task C) Take any image and apply Gaussian high pass filter and Gaussian low pass filter on it.

Task D) CSI Style Image Enhancement (for bonus marks)?

You are the image processing expert for a local police department. A detective has reopened a cold case and part of his evidence is a newspaper print of a car. You have been asked to do some CSI style magic to see if you can learn the suspect's license plate number or see his face.

1. Download the image "[halftone.png](#)" and store it in MATLAB's "Current Directory".



2. Use a set of notch filters to remove the peaks from the image's Fourier transform.

TIPS:

- create a function that takes a list of peaks as an argument
 - the peaks form a repetitive pattern. Figure out the pattern to save time.
3. Fine tune your results by trying varying widths of the three notch filter types. Provide a best effort solution for each one. Also provide a blurring based solution in Photoshop.

4. The following is one possible solution:



Deliverables

[Perform above mentioned practice tasks in Matlab or Python.](#)

Just make a new folder named as DIP-Lab8 in existing private repository and then add the teacher as a collaborator. All the code must be in runnable format in order to get the credit.

1. A file with commented source code representing the work accomplished for this lab.
2. All files should contain author in the comments at the top of the file.