INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Electronics & Electrical Communication

M.Tech. [Semester-1]

Vision and Intelligence Systems

EC69211 – Image and Video Processing Laboratory

Experiment Number:- 5

Frequency domain filtering

Submitted by:

Ayush Jangid (22EC65R24)
Suraj Kumar(22EC65R14)

Contents

1. Objective
2. Introduction
3. Filter responses
4. Algorithm
5. Output Results
6. Discussion

Objective-

Q1. Perform the following frequency domain filtering by writing your own Python functions (LPF= Low Pass Filter, HPF = High Pass Filter). a. Ideal LPF, Ideal HPF
b. Gaussian LPF, Gaussian HPF
c. Butterworth LPF, Butterworth HPF
INPUT: Image filename and cut off frequency are the input arguments.
OUTPUT: Display the (shifted) magnitude spectrums of the input, of the filter and that of filtered output. You may use the tracker/slider function to choose 1) images, 2) filter types and 3) cut-off frequencies.

Q2. Read the "leopard_elephant.jpg" image. This is an example of an image illusion in which the perception of an image changes with viewing angle, time spent viewing, or image size. A leopard can be seen in the image if you view it at its full spatial resolution. However, if you view the image at a lower spatial resolution, an elephant will appear. Using your frequency domain filtering idea, create an identical optical illusion. Read the files "einstein.png" and "marilyn.png." Make a hybrid image so that when viewed at a higher spatial resolution, the illusion appears to be Einstein, and when viewed at a lower spatial resolution, it appears to be Marilyn.
INPUT: einstein.png and marilyn.png.
OUTPUT: Hybrid image formed by einstein.png and marilyn.png

Q3. Read the "cameraman_noisy1.jpg" image. What kind of distortion did this image undergo? Could you provide some insight into the physical events that caused this distortion? Create a function that will automatically remove noise from images of this kind. Check your algorithm's robustness by seeing how well it can eliminate noise from the "cameraman_noisy2.jpg" image.
INPUT: cameraman_noisy1.jpg, cameraman_noisy2.jpg.
OUTPUT: Respective filtered image.

Introduction-

In this experiment our aim is to work on frequency domain filtering. Frequency Domain Filters are used for smoothing and sharpening of image by removal of high or low frequency components. Smoothening is achieved by attenuation of high frequency i.e. by applying low pass filter.

We consider three types of low pass filter:

Ideal low pass filter:

A 2D low pass filter that passes without attenuation all frequencies within a circle of radius D_0 from the origin and cut off all frequencies outside this circle is called ideal low pass filter. It is specified by the function

$$H(u,v) = 1 \text{ if } D(u,v) <= D_0$$

$$= 0 \text{ if } D(u,v) <= D_0$$

Where D_0 is a positive constant and D(u,v) is distance between a point (u,v) in frequency domain and center of the frequency rectangle.

$$D(u,v) = [(u – M/2)^2 + (v – N/2)^2]^{0.5}$$ M
and N are dimensions of image.

Gaussian low pass filter:

$$H(u,v) = \exp(-D^2(u,v)/2D_0^2)$$

Where D_0 is a positive constant and D(u,v) is distance between a point (u,v) in frequency domain and center of the frequency rectangle.

$$D(u,v) = [(u – M/2)^2 + (v – N/2)^2]^{0.5}$$ M
and N are dimensions of image.

Butterworth Low Pass Filter:

The transfer function of a Butterworth low pass filter of order n, and with cutoff frequency at a distance D_0 from the origin is defined as

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

Ideal High pass Filter:

A high pass filter is obtained from a low pass filter by applying following equation:

$$H_{HP}(u,v) = 1 – H_{LP}(u,v)$$

Gaussian High pass Filter:

The transfer function of the Gaussian high pass filter with cut off frequency locus at a distance D_0 from the center of the frequency rectangle is given by:

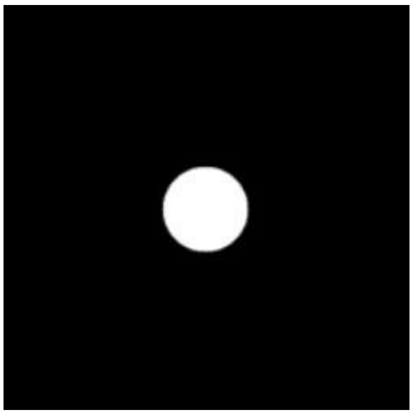
$$H(u,v) = 1 – \exp(- D^2(u,v)/2D_0^2)$$

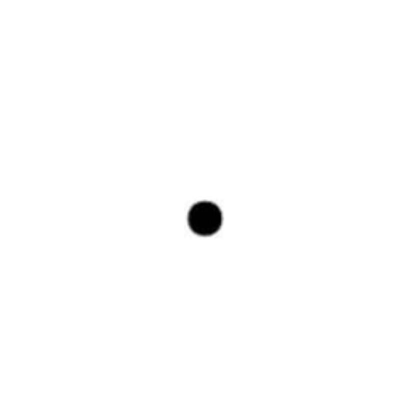
Butterworth High Pass Filter:

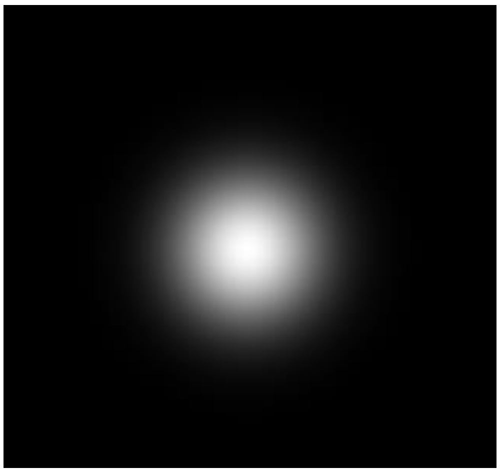
A 2D Butterworth high pass filter of order n and cut off frequency D_0 is given as:

$$H_{HP} = 1/\{1 + [D_0/D(u,v)]^{2*n}\}$$

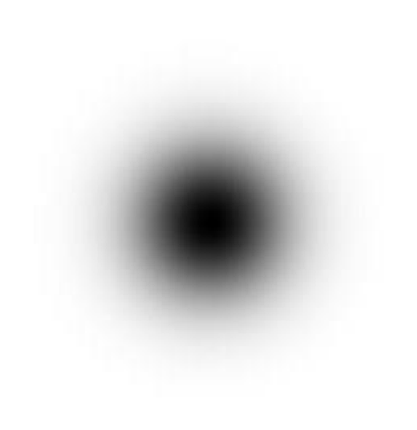
Filters displayed as an image:

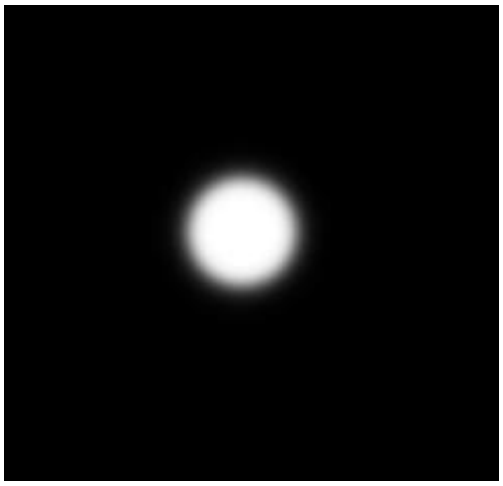

Ideal low pass filter

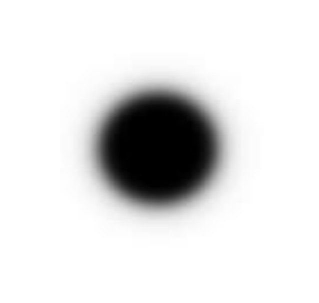

Ideal low pass filter


Gaussian Low pass Filter

$D_0 = 50$


Gaussian High pass Filter


Butterworth Low pass filter


Butterworth High pass filter

$D_0 = 50$, $n = 10$

Algorithm:

Ideal LPF-

1. The attached code is an ipynb file that has been compiled in Python IDLE 3.10.
2. First Routine designed is the code for Ideal Low Pass Filter. Input the file name and the cut off frequency
3. Calculate FFT of input file by using in-built numpy function “fft2()” and then shift using “fftshift()” function.
4. Initialize a low pass filter with all values zeroes.
5. Calculate distance of every pixel from center pixel and if distance is less than cutoff frequency, put those pixel values as 1. Here, cutoff frequency can be altered as required.
5. Multiply centered fft of input image with designed ideal low pass filter to obtained fft of filtered image.
6. Use inbuilt “ifftshift()” function to create shifted version of fft of filtered image, and then use “ifft2()” to create resultant filtered image.

Ideal HPF-

1. Create an ideal HPF by subtracting an ideal LPF from 1.
2. Repeat the steps from 2 to 6 as mentioned above for Ideal LPF.

Gaussian LPF-

1. Firstly, a subroutine for generating Gaussian LPF is written, which takes cut off frequency and the input image as the arguments.
2. Calculate the distance of each pixel of the initialized filter and then use Gaussian Formula to give weights to pixels.
3. Multiply centered fft of image with designed filter to generate fft of filtered image.
4. Use inbuilt “ifftshift()” function to create shifted version of fft of filtered image, and then use “ifft2()” to create resultant filtered image.

Gaussian HPF-

1. Create a Gaussian HPF by subtracting generated Gaussian LPF from 1.
2. Repeat the steps from 2 to 6 as mentioned above for Ideal LPF.

Butterworth LPF-

1. Firstly, a subroutine for generating Butterworth LPF is written, which takes cutoff frequency, input image and order of filter as the arguments.

2. Calculate the distance of each pixel of the initialized filter and then use Butterworth filter Formula to give weights to pixels.
3. Multiply centered fft of image with designed filter to generate fft of filtered image.
4. Use inbuilt "ifftshift()" function to create shifted version of fft of filtered image, and then use "ifft2()" to create resultant filtered image.

Butterworth HPF-

1. Create a Butterworth HPF by subtracting generated Butterworth LPF from 1.
2. Repeat the steps from 1 to 6 as mentioned above for Butterworth LPF.

Hybrid Image-

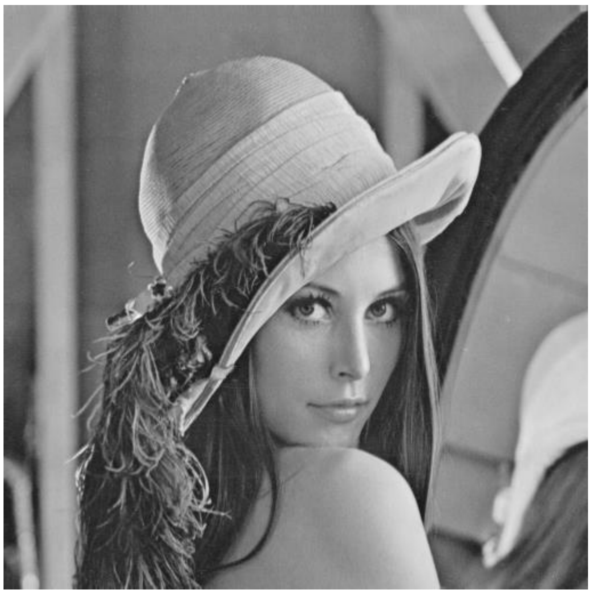
1. Read both the images using "imread()" function one by one.
2. Calculate FFT of input files by using in-built numpy function "fft2()" and then shift using "fftshift()" function.
3. Create filtered images by multiplying image that should be visible at a higher spatial resolution by high pass filter and the other image by low pass filter.
4. Generate shifted versions of both the filtered images by using inbuilt "ifftshift()" function.
5. Take average of both images to generate the hybrid image.

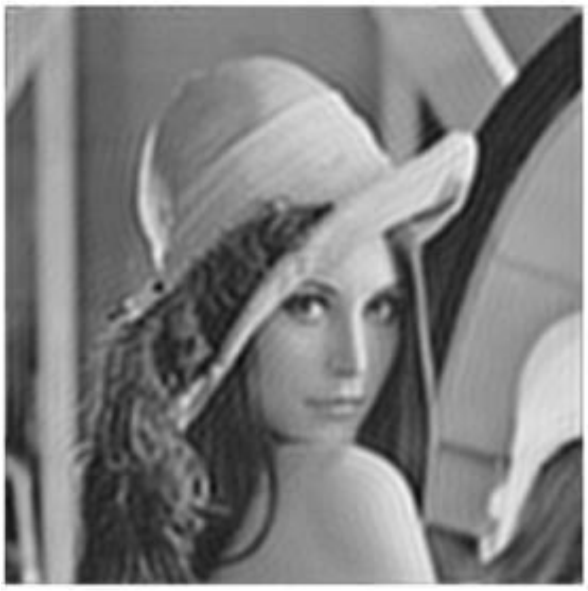
Image Restoration-

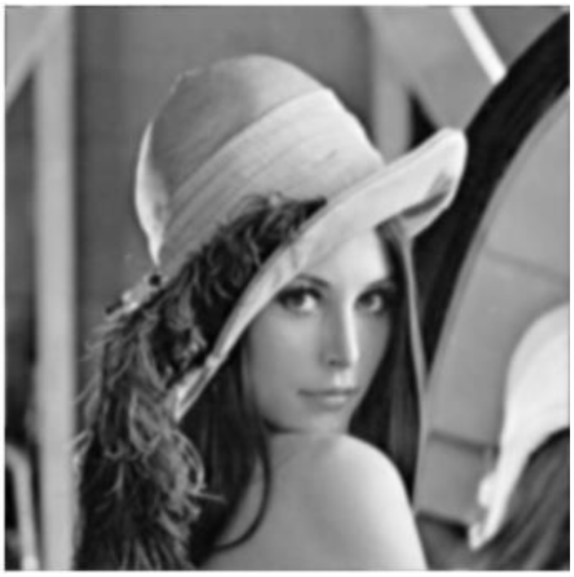
1. Read the noisy input image. Calculate FFT of input file by using in-built numpy function “fft2()” and then shift using “fftshift()” function.
2. The noise is periodic noise, from the frequency spectrum it was clear that the range of k defines the number of pixels in the centered fft of noisy input image that will be nullified. We have chosen k=20 to 30.
3. Shift the centered fft of the back to corners by using “ifftshift()” function and restore image using “ifft()” function. And display the absolute value of ifft as image

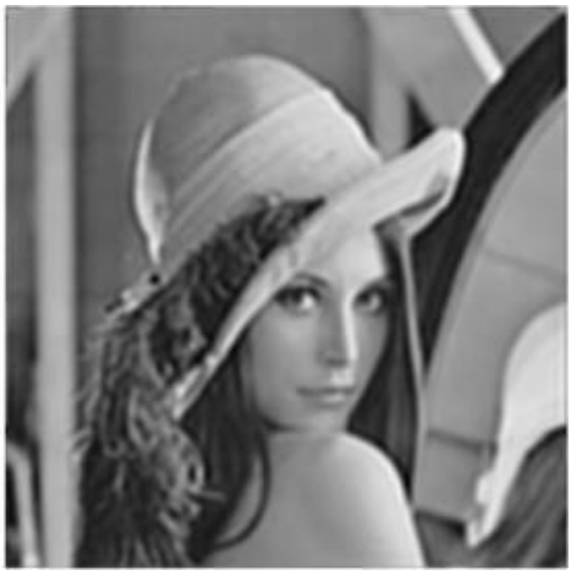
Input and Output Images:

Low Pass Filter:


Input image

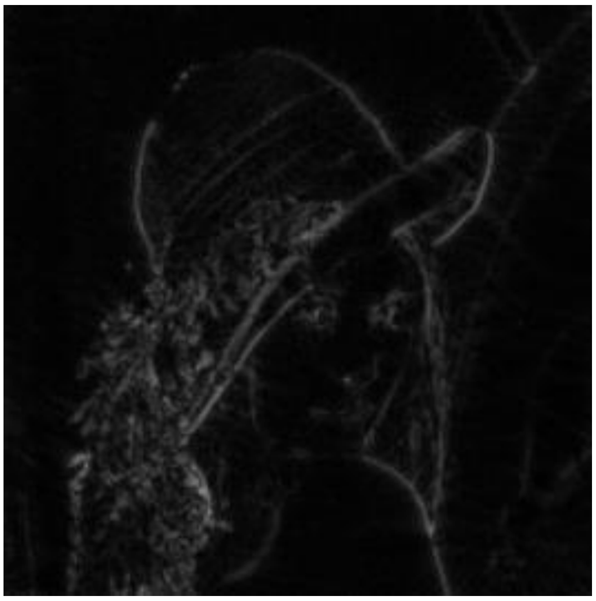

Ideal low pass filter output

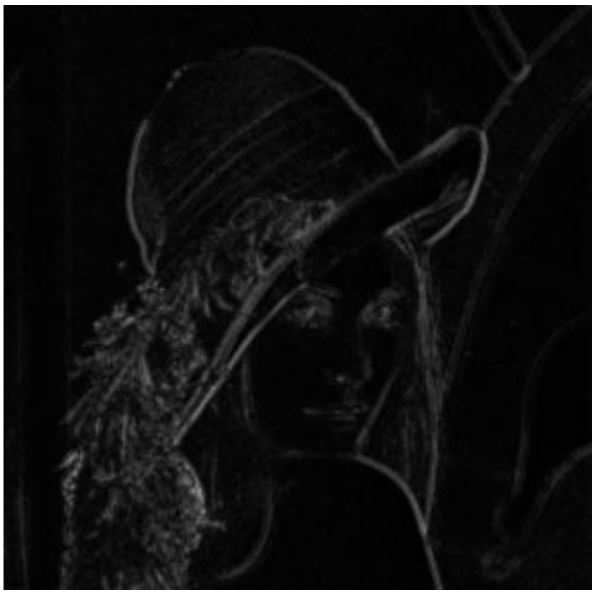

Gaussian low pass filter output

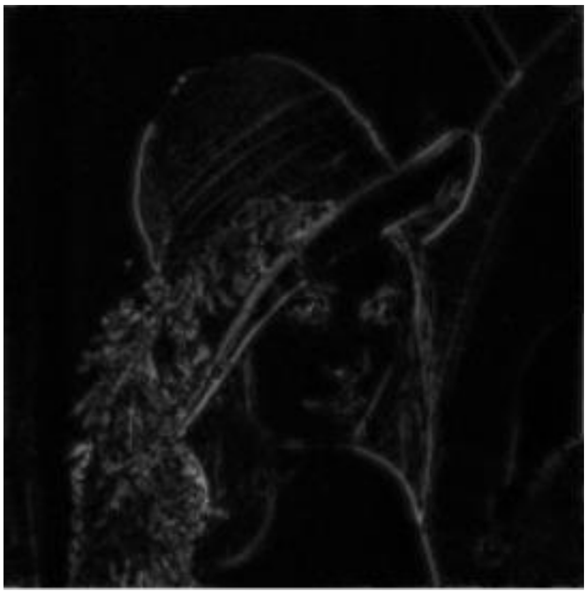

Butterworth low pass filter output

High Pass Filter:

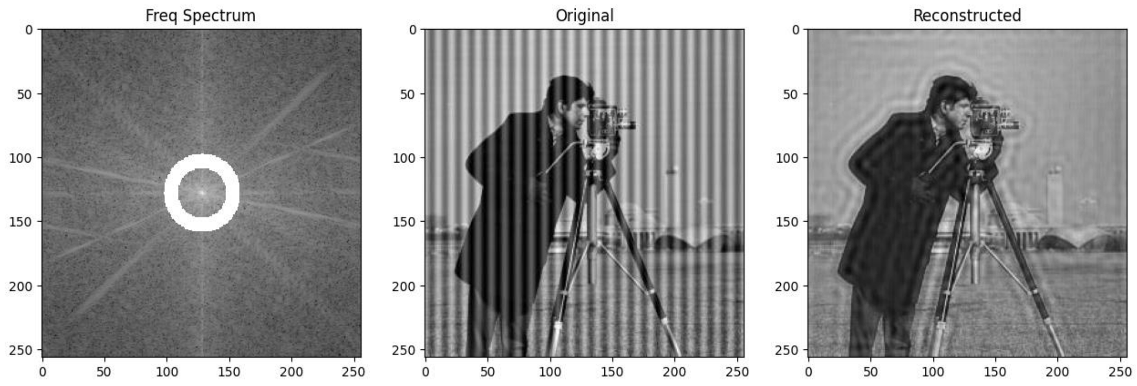
Input image


Ideal high pass filter output

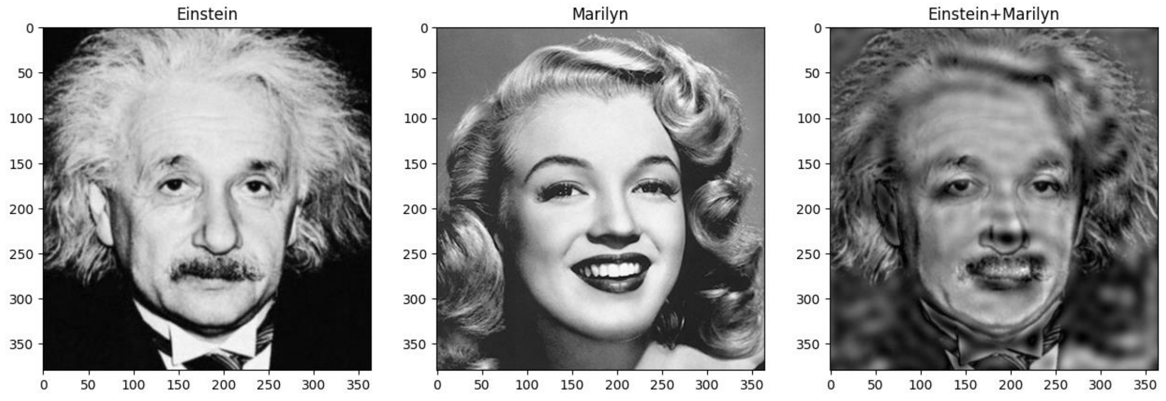

Gaussian high pass filter output


Butterworth high pass filter output

USING IDEAL FILTER



Optical Illusion



Discussion:

1. Frequency domain filtering techniques are used for smoothing and sharpening of image by applying filters in frequency domain.
2. Since it's just multiplication operation of frequency response of image and filter transfer function it's easier to compute the response.
3. Ideal filters have sharp transitions, which leads to Ringing Effect.
4. Ringing effect refers to effect in images appearing as "Rippling Rings" near sharp edges in the input image.
5. By applying Gaussian Filter, ie, by using filters with smooth transitions, ringing effect can be avoided.
6. Images with low pass filters applied on them become smooth, ie, their sharp edges are suppressed. When viewed from a close distance, such images are not visible properly, but from a distance, our brain is able to visually perceive such smooth images.
7. Reverse is the case with High Pass filtered images. As their edges are enhanced, they appear sharper when viewed from a close distance.
8. These properties are used to create hybrid images by blending two different high pass and low pass images. Such Images look completely different when viewed from different distances.
9. Images with stripes embedded are infected with periodic noise. Such type of noise can be blocked by manually masking high intensity components of frequency spectrum, except the DC Component at the center.

References:

1. Digital Image Processing by: Rafael C. Gonzalez and Rafael C. Gonzalez
2. NPTEL Digital Image Processing Lectures by P.K. Biswas
3. http://olivalab.mit.edu/
4. https://www.researchgate.net/