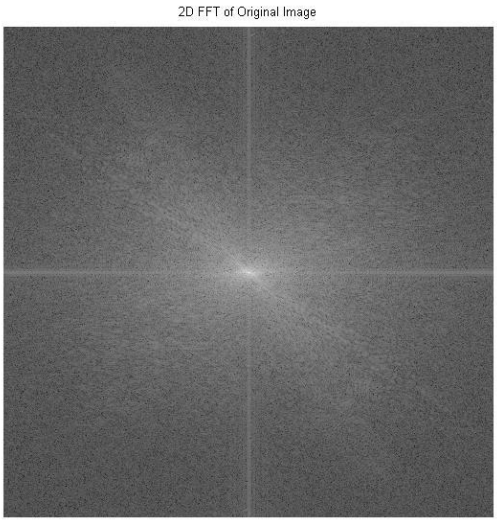
COMP ENG 4TN4

Assignment 1
Frequency Domain Filters

Sara Jamil – jamils2

Frequency Domain Filters

In order to implement frequency domain filters, the original image being filtered must be converted into the frequency domain. This can be done by taking the fast Fourier transform of the image. The log of the magnitude spectrum of the original image is shown below, represented in gray-scale.



In this image, as in most natural images, the highest values of the magnitude of the frequency spectrum occur in the low frequencies. Applying a low-pass filter will keep most of the energy in the low frequencies but attenuate the high frequencies of the image. Therefore, the high frequency edges and details are expected to be lost after low-pass filtering.

Ideal Low-Pass Filter

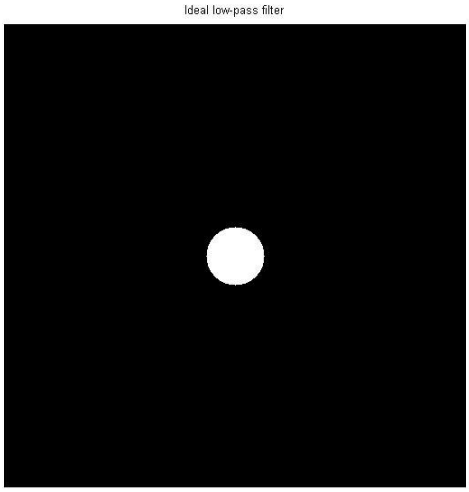
An ideal low-pass filter has the following transfer function

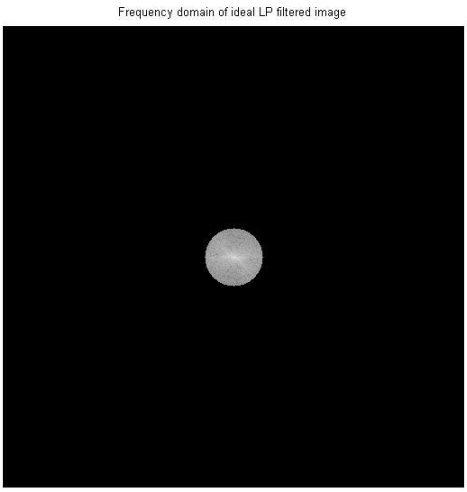
$$H(u,v)=\begin{cases}1 & D(u,v)\le D_0\\ 0 & D(u,v)>D_0\end{cases}$$

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2}$$

(M/2,N/2): center in frequency domain

where D_0 is a specified non-negative number that represents the cutoff and $D(u,v)$ is the distance from the point (u,v) to the center of the filter. When applying this filter the image, the filter H multiplies the Fourier transform of the image and "cuts off" all components outside the circle while leaving all the components in the circle unchanged. The following figures show the frequency spectrum of the ideal low-pass filter as well as the filtered image frequency spectrum. In this example, the cutoff frequency $D_0 = 32$.





After preforming the inverse Fourier transform of the filtered frequency spectrum, the resulting image in the spatial domain is shown below.

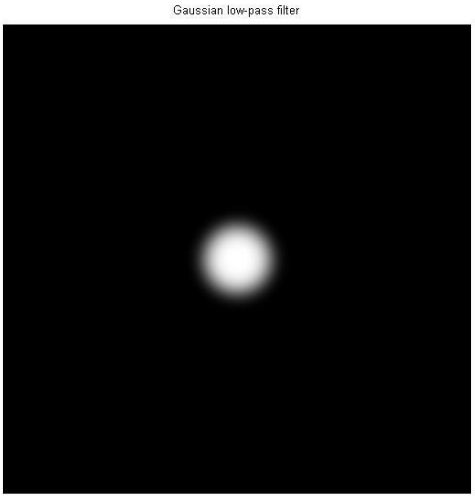
It is quite clear that there is significant ringing artifact in the image, which is a result of the properties of the ideal filter. Multiplication in the frequency domain is equivalent to convolution in spatial domain involving the sinc function.

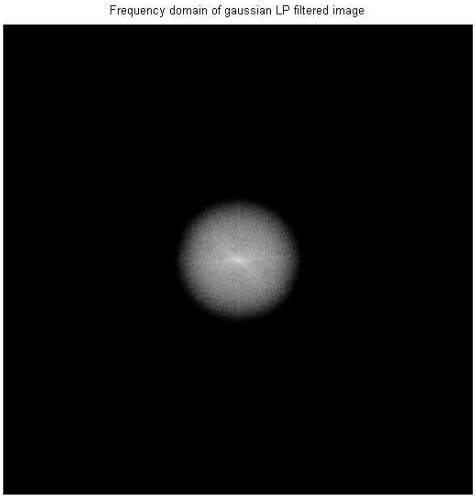
Gaussian Low-Pass Filter

The transfer function of a Gaussian low-pass filter is given by

$$H(u,v) = e^{-\frac{D^2(u,v)}{2D^2_0}}$$

where D_0 is the cutoff parameter. The following figures show the frequency spectrum of the Gaussian low-pass filter and the filtered image frequency spectrum, with the same cutoff frequency as the ideal low-pass filter.





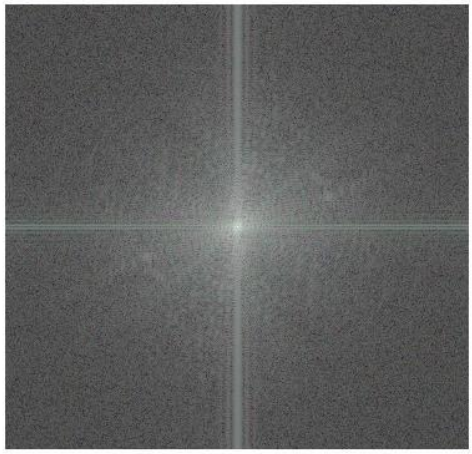
Similarly, by taking the inverse Fourier transform, the following image is constructed.

The Gaussian filtered image does not have ringing artifact like the ideal low-pass filtered image. This is also due to the shape of the Gaussian filter in the special domain. The image only appears to be blurred due to the fact that the high frequency components have been attenuated.

The same procedure has been applied to another example of an image with colour. The spatial and frequency domain representations are shown.

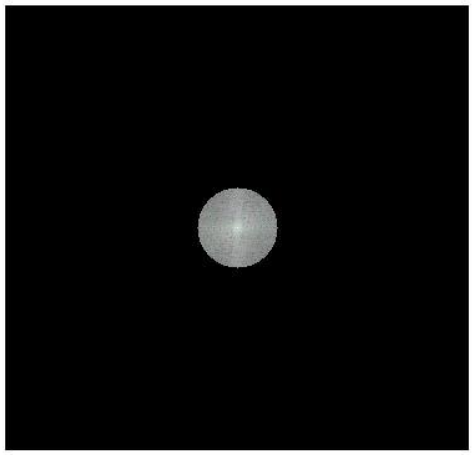
Original Image

2D FFT of Original Image



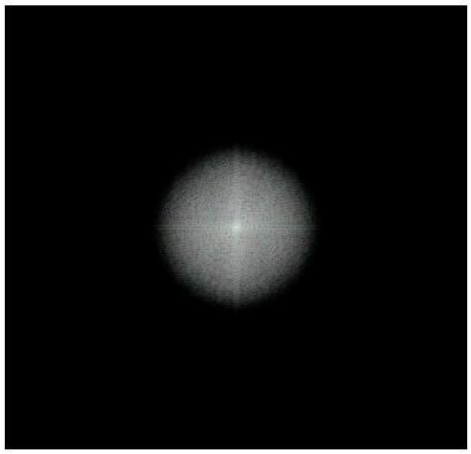
Ideal LP filtered image

Frequency domain of ideal LP filtered image



Gaussian LP filtered image

Frequency domain of gaussian LP filtered image



The ringing artifact is present in the ideal LP filtered image and is absent in the Gaussian LP filtered image.

One more example:

Original Image

Ideal LP filtered image

Gaussian LP filtered image

Matlab Implementation:

```
% Import image
img = imread('C:\Users\Sara\Desktop\Comp Eng 4TN4\Pictures\owl-kitten.jpg');
figure(1)
imshow(img);
title('Original Image');

% Frequency domain of image
imgfft = fft2(double(img));
imgfft = fftshift(imgfft);
imgfftmag = abs(imgfft);
imgfftshow = mat2gray(log(imgfftmag+1));
figure(2)
imshow(imgfftshow);
title('2D FFT of Original Image');

% Meshgrid frequency matrices
% for computing freq domain filters
D0 = 32;    %cutoff frequency
D0 = D0^2;
[M,N,P] = size(img);
u = 0:(M-1);    %set up range of variables
v = 0:(N-1);

idx = find(u>(M/2));    %compute indices for meshgrid
idy = find(v>(N/2));
u(idx) = u(idx) - M;
v(idy) = v(idy) - N;

[U, V] = meshgrid(u,v); %meshgrid arrays

D = U.^2 + V.^2;    %compute distances to center of filter

% Ideal filter
idealH = double(D <= D0);
idealH = fftshift(idealH');
idealHshow = mat2gray(idealH);
figure(3)
imshow(idealHshow);
title('Ideal low-pass filter');

% Filtering the image with ideal LPF
idealF_img = zeros(size(imgfft));
for i = 1:P     %P=3 for colour images
    idealF_img(:,:,i) = idealH.*imgfft(:,:,i);
end
idealF_imgmag = abs(idealF_img);
idealF_imgshow = mat2gray(log(idealF_imgmag+1));
figure(4)
imshow(idealF_imgshow);
title('Frequency domain of ideal LP filtered image');

% Inverse FFT to obtain image in spatial domain
idealf_img = ifft2(ifftshift(idealF_img));
figure(5)
```

```
imshow(uint8(idealf_img));
title('Ideal LP filtered image');

% Gaussian filter
gaussH = exp(-(D.^2)./(2*(D0^2)));
gaussH = fftshift(gaussH');
gaussHshow = mat2gray(gaussH);
figure(6)
imshow(gaussHshow);
title('Gaussian low-pass filter');

% Filtering the image with gaussian LPF
gaussF_img = zeros(size(imgfft));
for j = 1:P
    gaussF_img(:,:,j) = gaussH.*imgfft(:,:,j);
end
gaussF_imgmag = abs(gaussF_img);
gaussF_imgshow = mat2gray(log(gaussF_imgmag+1));
figure(7)
imshow(gaussF_imgshow);
title('Frequency domain of gaussian LP filtered image');

% Inverse FFT to obtain image in spatial domain
gaussf_img = ifft2(ifftshift(gaussF_img));
figure(8)
imshow(uint8(gaussf_img));
title('Gaussian LP filtered image');
```