**CS 559 Fall 2020**

**Assignment 4 Grading Sheet**

**Please attach this sheet to your solutions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Last Name: Mineev  First Name: Mikhail  Red ID: 820252323 | | |  |  | | --- | --- | | Questions 1 |  | | Question 2 |  | | Question 3 |  | | Question 4 |  | | Question 5 |  | | Extra Credit (10%) |  | | Total |  | |

**Question 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Possible scores | Earned | Comments |
| 1. Power of 2 assumption | 4 |  |  |
| 1. Consequences of symmetry/periodic | 4 |  |  |
| 1. Bit reversal | 4 |  |  |
| 1. Ringing effect | 4 |  |  |
| 1. Power spectrum | 4 |  |  |

**Question 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria: | Possible scores | Earned | Comments |
| 1. Gaussian bandpass | 8 |  |  |
| 1. Demonstration | 6 |  |  |

**Question 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Possible scores | Earned | Comments |
| (a) Propose suitable filter | 5 |  |  |
| (b) Images | 10 |  |  |
| (c) Discussion | 7 |  |  |

**Question 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Possible scores | Earned | Comments |
| (a) Propose suitable filter | 5 |  |  |
| (b) Images | 10 |  |  |
| (c) Discussion | 7 |  |  |

**Question 5**

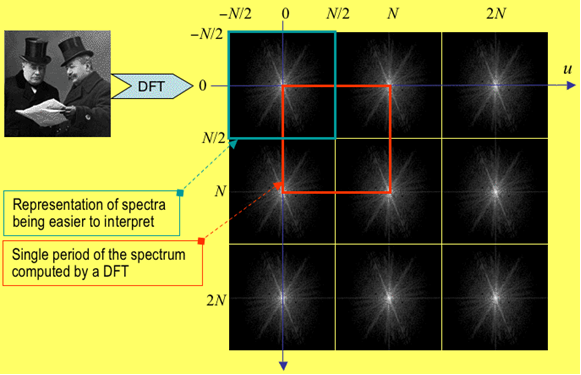
|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Possible scores | Earned | Comments |
| (a) Propose suitable filter | 5 |  |  |
| (b) Images | 10 |  |  |
| (c) Discussion | 7 |  |  |

1. a) FFT uses base 2 logs by default. It is necessary that the number of data points entered equal to 2^X where x can be any number. This allows for easy calculations to get the log of an image.

b) According to equation 5.16

***F(u,v) = F(u+n, v) = F(u, v+n) = F(u+n, v+n)***

Meaning that the Fourier transform is actually a periodic function with a period n. This is demonstrated further by Fugure 5.16.



Here we can see that the pattern repeats every period N

c)Bit reversal is necessary for obtaining the correct sequences in order to perform the Fast Fourier Transform.

array binary bit array

index representation reversal index

0 0000 0000 0

1 0001 1000 8 

2 0010 0100 4

3 0011 1100 12

4 0100 0010 2

5 0101 1010 10

6 0110 0110 6

7 0111 1110 14

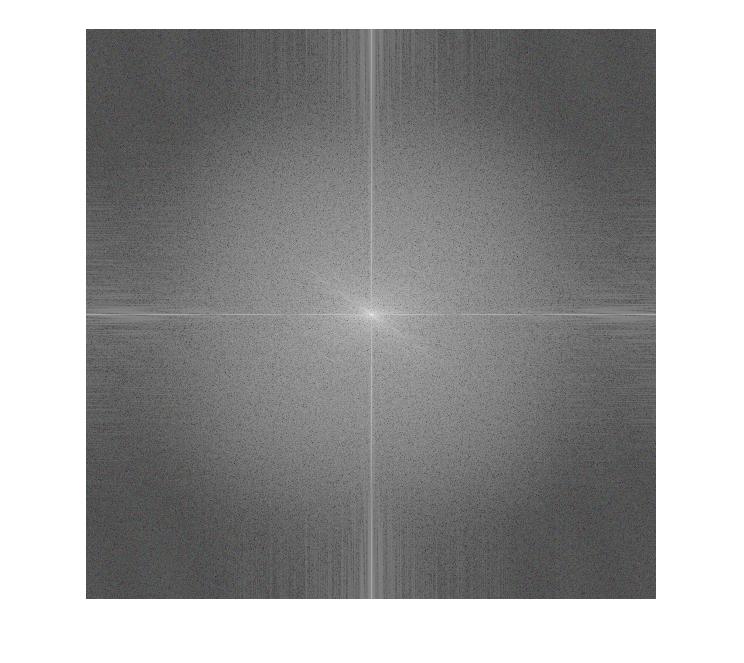
15 1111 1111 15

As we can see here since n is the power of 2 a certain pattern emerges. Due to this recursive pattern the complexity of FFT becomes O(nlogn)

d) Ringing effect occurs when there is a fast change in the characteristics of the filter. It can be reduced by smoothing out this sharp change using either a Gaussian or Butterworth filters.

e) The spectrum image provides the frequency components of an image on a two-dimensional representation, usually an 8bit gray level image. In images of non-synthetic objects, the frequencies are usually concentrated in the center, while in images of synthetic objects the frequencies are all over the place.

Here is an example of the latter.



2) a) The equation I propose for the Gaussian Bypass Filter is as follows



b) The following code is responsible for producing images demonstrated after.

input = imread('Fingerprint.png'); %input image (a)

A = padarray(input, [(1),(0)]); %pad the image to make it square

r0 = 10; %frequency threshhold

w = 1; %bandwidth

[u,v]=meshgrid(-99:98, -99:98); %creating

Y = (-(4.\*w.\*r0.^2) ./ (u.^2 + v.^2)); %Gaussian

GBS = exp(Y); %Bypass Filter

imfft=fftshift(fft2(A)); %using fft on image

imshow(mat2gray(log(abs(imfft)))); %show the fft (b)

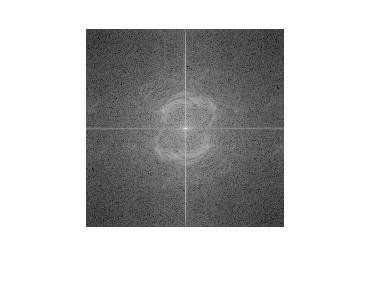
FilteredIm=imfft.\*GBS; %applying the filter on image w/ fft

imshow(mat2gray(log(1+abs(FilteredIm)))); % shows the filtered image (c)

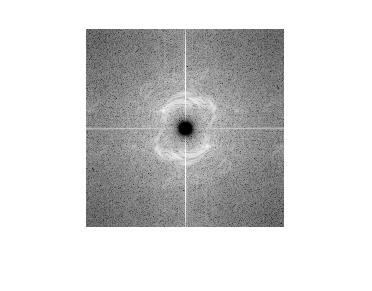
InverseFilteredIm=ifft2(FilteredIm); %inversing the image to get result

imshow(mat2gray(abs(InverseFilteredIm))); %shows result (d)

1. Original image (b)Image after Fourier transform



(c)After filter application (d) Resulting image



In the case of our filter it is easy to see that it made the output image of the fingerprint much more detailed. Essentially what this filter achieved as shown by image (c) is completely remove the higher frequencies, making those black. With those higher frequencies removed we made it so the less pronounced features were able to stand out.

Due to this property of making the less pronounced features stand out this kind of filter would be useful for simpler images that were overexposed during the development process or have been faded away due to being in the sun for far too long.

3)

For this program I decided to use a Gaussian Low Pass Filter in the frequency domain in order to clean up the lines as much as possible. Unfortunately in the process the image ended up being rather blurry.

The following code is responsible for producing images demonstrated after.

input = imread('lena.png'); %input image (a)

padIm = padarray(input, [(0),(2)]); %pad the image to make it square

r0 = 18; %frequency threshold

[u,v]=meshgrid(-218:217, -218:217); %Creating

X = (-(u.^2 + v.^2) ./ (2.\* r0.^2)); %Gaussian Low

gLp = exp(X); %Pass Filter

imfft=fftshift(fft2(padIm));%using fft on image

imshow(mat2gray(log(abs(imfft)))); %show the fft (b)

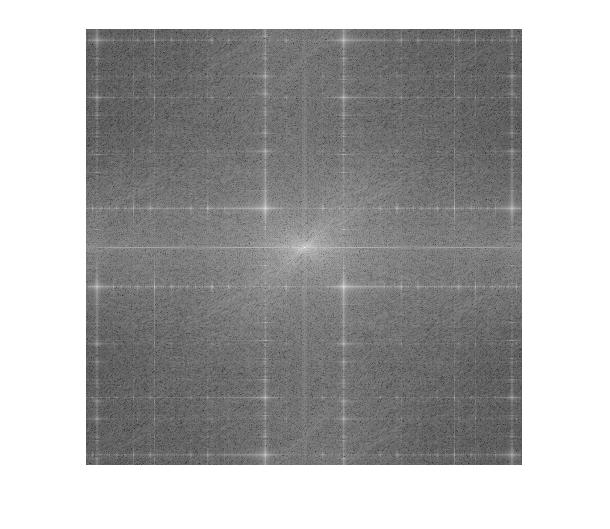
FilteredIm=imfft.\*gLp;%applying the filter on image w/ fft

imshow(mat2gray(log(1+abs(FilteredIm))));% shows the filtered image (c)

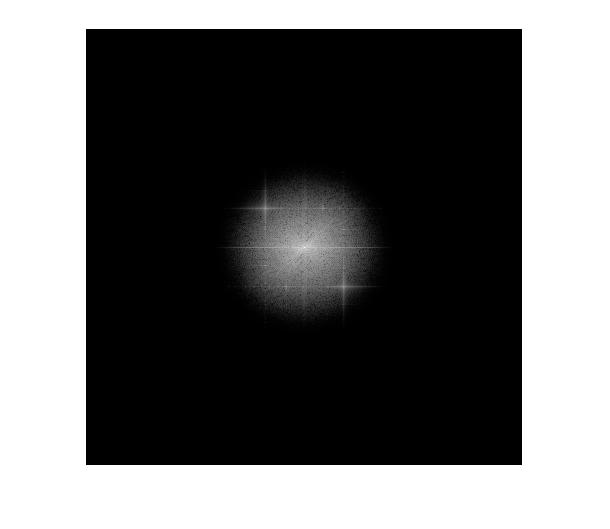
InverseFilteredIm=ifft2(FilteredIm);%inversing the image to get result

imshow(mat2gray(abs(InverseFilteredIm)));%show result (d)

1. Original image (b)Image after Fourier transform



(c)After filter application (d) Resulting image



The main reason why I chose the Gaussian Filter is because it is one of the most effective ways to smooth out and filter anu sharp edges and artifact present in an image, the drawback however is quite obvious with the image becoming rather blurry in the process. Some other choices which needed to be done for this program is the frequency threshold or cutoff.

After testing the code multiple times I found the best compromise for this parameter to be 18.

Anything lower than that and the image still contains highly visible lines like in the (a) Original Image. Anything higher than 18 and the image would become almost unrecognizable with a few blurry shapes filling the image.

That was the main problem with this filter, it would make the image too blurry to understand in order to remove all of the artifacts on it.

4) The filter I decided to use for problem 4 is the Butterworth Low Pass Filter in the frequency domain in order to clean up the image as much as possible.

The following code is responsible for producing images demonstrated after.

input = imread('lena4.png'); %input image (a)

A = padarray(input, [(12),(4)]); %pad the image to make it square

targetSize = [570 570];

rad = centerCropWindow2d(size(A),targetSize);

padIm = imcrop(A,rad); %making image square

p = 2; %order of the filter

r0 = 20; %frequency threshold

[u,v]=meshgrid(-285:284, -285:284); %creating

r = sqrt(u.^2+v.^2); %Butterworth Low

BwFl = (1./(1 + (r./ r0).^(2.\* p))); %Pass Filter

imfft=fftshift(fft2(padIm)); %using fft on image

imshow(mat2gray(log(abs(imfft)))); %show the fft (b)

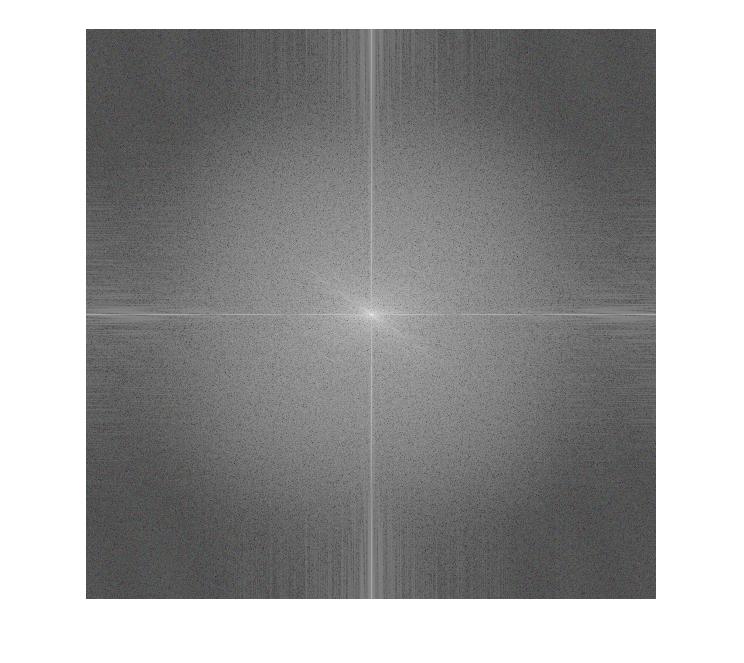
FilteredIm=imfft.\*BwFl; %applying the filter on image w/ fft

imshow(mat2gray(log(1+abs(FilteredIm)))); % shows the filtered image (c)

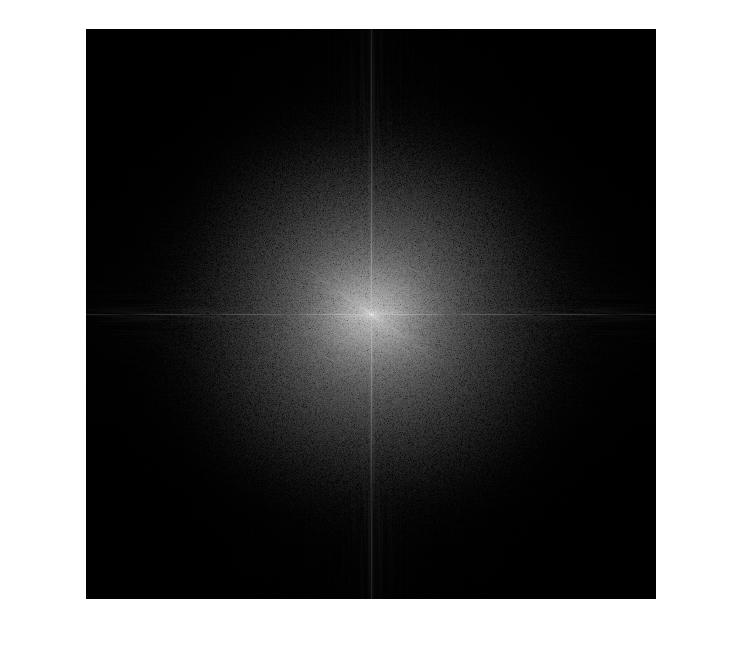
InverseFilteredIm=ifft2(FilteredIm); %inversing the image to get result

imshow(mat2gray(abs(InverseFilteredIm))); %shows result (d)

1. Original image (b)Image after Fourier transform



(c)After filter application (d) Resulting image

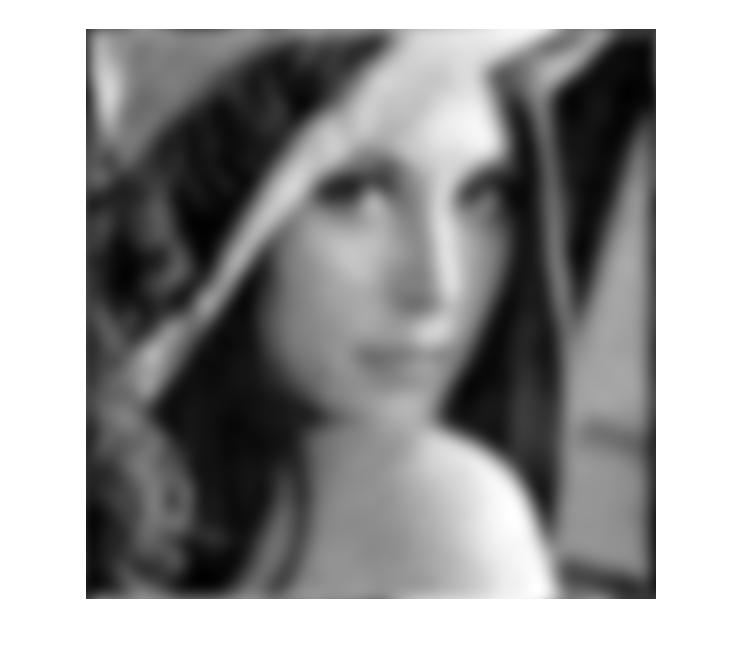


The main reason why I decided to use the Butterworth Low Pass Filter, because it is supposed to make the highest frequencies be lowered. This would allow for the noise to be removed while allowing the blurring to remain at a minimum.

To achieve this I decided to set the order at 2 and se the frequency threshold at 20. This turned out to be a relatively ok compromise which removed most of the noise without making the image unrecognizable due to blurring.

The main problem withthis filter is the problem if 2 parameters which need to be adjusted in order to produce a halfway decent image. With my relatively short testing I came up with the images above, however with more extensive trials it should be possible to make the image clearer while removing the noise.

For comparison if we use a Gaussian Low Pass filer to remove the noise, our image becomes much blurrier than with Butterworth.



5) For this problem I decided to use the median filter in the special domain to attempt to remove the salt and pepper noise while making the image as clear as possible.

The following code is responsible for producing images demonstrated after.

input = imread('colr.png'); %input image

r = input(:, :, 1); %seperate the red channel

g = input(:, :, 2); %seperate the green channel

b = input(:, :, 3); %seperate the blue channel

redFiltered = medfilt2(r, [6 6]); %apply median filter on red

greenFiltered = medfilt2(g, [6 6]); %apply median filter on green

blueFiltered = medfilt2(b, [6 6]); %apply median filter on blue

recombine = cat(3, redFiltered, greenFiltered, blueFiltered); %recombine all channels

imshow(recombine); %output the filtered image

1. Original Image (b) Filtered Image



Here I used the median filter simply because it is one of the best filters for removing salt and pepper noise from an image, true enough it did not disappoint.

The main challenge I faced in this problem is what size of a median mask I should have set for each of the channels. I ended up settling on [6 6] because it minimized the artifacts while keeping the image relatively clear.

However, this filter has a slight problem, that problem is the fact that it is unable to filter out the noise from the edges, as can be seen from the above image. A solution which might work is either cropping the size or padding the edge with a layer of white or black pixels and then removing that layer after the image is fully filtered.