Homework 4 – Frequency Domain

Arya Araban

Abstract

Apart from the spatial domain, a way to study digital images is by converting the image into the frequency domain. By doing this, we have control over aspects of an image that we may not be able to control in the Spatial Domain. One notable application for this is being able to classify handwritten text.

Report Info

Date: 1399/09/21

Keywords:
Frequency
Fourier Transform
Filtering

1- Introduction

In order to convert an image from the spatial domain to the frequency domain, we perform what is known as the Fourier Transform (Named after the French mathematician Joseph Fourier). After which we're able to apply the changes we want to on an image by using a desired filter. After applying the filter in the frequency domain, we perform what is known as an Inverse Fourier Transform on the original image to convert the image back from the frequency domain into the spatial domain.

2- Technical Description

Problem 4.1.1:

For each of the filters given below, first determine if they are separable or not, if they are, compute the 1-D FT on each of the vectors forming them, else compute the 2-D FT using the entire filter. based on the filter coefficients as well as its frequency response, determine the function of the filter.

Solution:

In order to determine weather the filter is seprabale or not, we have to check the rank of the filter. If the rank equates to 1, then the matrix is separable, otherwise it is not.

One thing to note is that after we perform the fourier transform on any image, The zero-frequency coefficient is displayed in the four corners. To display it in the center, we shift it into center using a function. After doing this, the low frequencies will be in the center, and the more far we consider from the center, the higher the frequencies will become.

In this section we will determine the function of each of the filters as well as if they are separable or not. The details of how applying the filter after a Fourier Transform affects the image well be discussed in the next section.

the corners, the higher frequencies become. However, this can be problematic for us to make changes to since the lower frequencies aren't gathered together, so for our ease of use, we shift the low frequencies to be in the middle, and the high frequency points will be relative to their distance with the center.

If we don't apply the logarithmic transformation, the dynamic range of the Fourier coefficients is too large to be displayed on the screen, therefore all other values appear as black, except for a very small point in the center (If we have shifted).

If we apply the logarithmic transformation we'll get a better visualization of the magnitude response, and it can be realized that the resulted image contains components of all frequencies, but that their magnitude gets smaller for higher frequencies. Hence, the point can be made that low frequencies contain more image information than the higher ones.

Problem 4.2.1:

i)suppose an image size is 256 × 256 and a filter size is 11 × 11; What is the required size of the DFT to obtain the convolution of these two?

*ii) if we consider the image as remaining the same size (256*256), considering Z(m, n) as Z = IDFT (DFT(X).∗DFT(H)), for what points does Z(m, n) equal Y(m, n) (which we found in the previous question)*

Solution:

i) In order to perform 2D convolution we need to make sure that we don't go out of bounds, however due to filter being of size 11*11 and 256 not being divisible by 11, we need a zero padding in order to perform the convolution. we can notice that the nearest

$$a)\ \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

After a bit of computations, it becomes clear that this matrix is separable to two vectors. The first being (1/4) .* [1;2;1] and the second being (1/4) .* [1,2,1]
(multiplying these two gives us the original filter) , it can also clearly be seen that based on the filter coefficients this filter is a 3*3 Guassian Blur filter.

$$b)\ \begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$

This filter cannot be seperable due to the matrix rank being equal to 2.
Such 3*3 filters where the center is a positive integer and the others are equal to -1 are used for edge detection.

$$c)\ \begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

This filter also canot be seperable due to the matrix rank also being equal to 2.
Based on the coefficients, we can realize this filter is widely used for the sharpening of an image.

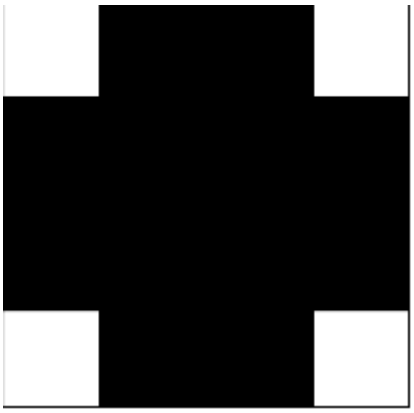
Problem 4.1.2:

Discuss what happens if we perform 2D DFT, with and without shifting the frequencies to the center, also with and without using a logarithmic transformation.

Solution:

We know that by not shifting the resulted fourier transform into the middle, the lower frequencies will be in the four corners of the image, and the more far we consider from

most contains most of the image data) to be in the four corners, like below:



So for example, if we want to perform a low pass filtering we have to keep the corners, while making the other parts (that have high frequencies) equal to zero.

Note that the above example is consistent to F(k, l) = 0 for TN < {k, l} < (1 − T)N which is the first part of the current question.

3- Describing the Results

Problem 4.1.1:

The original Barbara image is as follows:

When we bring this image into the frequency domain, apply shifting to the divisible number to 256 is 11*24=264, which is also a perfect fit if we want to consider a 264*264 image, we can add four zeros to each side of the horizontal picture scope, as well as four zeroes to each side of the vertical scope, overall giving us the result of a **264*264 image** which we apply the filter (which has also been brought to the frequency domain with the scale of 264) on.

ii) based on what we found from the previous answer, applying the filter will be the same for both *Z(m, n) and Y(m, n)* only in the points **where the filter doesn't enter the zero-padded area at all**, because only in the other points does applying the filter give the same results due to the coeffiecents being the equal for both the problems.

Problem 4.2.2:

Write a program that after performing 2D DFT, zeros out the coefficients at certain frequencies (given in the question), and then finally perform IDFT to get back the original image

Solution:

The steps taken in the program I wrote are as follows:

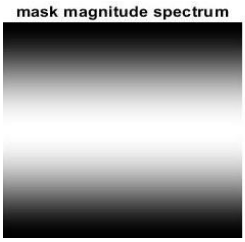
1- first we take the DFT of our image (without performing any sort of size scaling, due to not having an external filter)

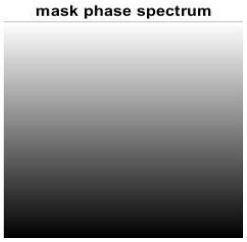
2- by using a for loop, we zero out the certain points in which the question states the frequencies become zero

3- Lastly, we perform an IDFT in which the get the filtered image which we want.

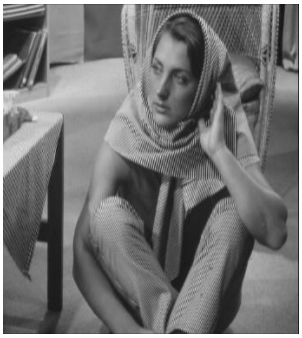
There are some things to note. Firstly we don't perform any sort of shifting, this makes it so that the low frequencies (which

Then after, we find the spectrums for the A = ¼.*[1;2;1] filter:





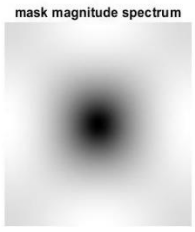
After applying B1, and then A to the image, the resulting image is as follows:

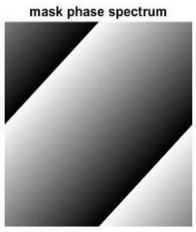


As seen, this image is slightly blurred compared to the original image.

$$b)\ \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

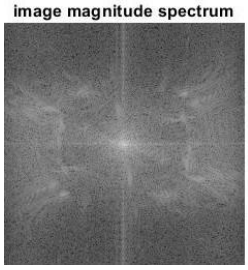
The spectrums for the b filter in the frequency domain is as follows:

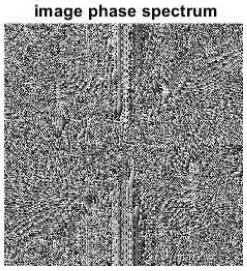




Once we apply this filter to the image, and then return the image from the frequency domain, the resulted image will be as follows:

center, and then perform logarithmic, the phase and magnitude will be like below:



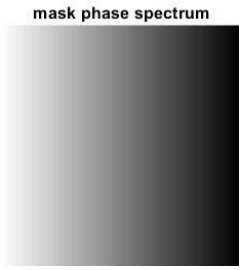


Now we'll see the results for applying the filters below (note that the we apply shifting and logarithmic for both the filters, and the original image. We also scale the filter's horizontal and vertical sizes to match the image's):

$$a)\ \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Like stated in the previous section, this filter is seperable due to the rank being equal to one. Once we separate this matrix filters into two vectors we end up with a matrix in the form of AB1^T = w, which in order to solve this equation we realize that we have to apply B1 (the row vector) first, and then apply A (the column vector) to the answer we found in the previous part.

The spectrums for the B1= ¼ .*[1,2,1] filter is as follows:

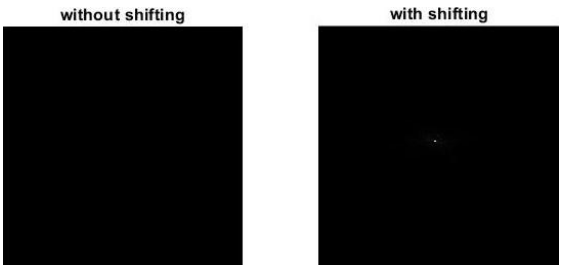


So finally, the obtained output image will be as follows:

Problem 4.2.1:

By not applying logarithmic, the result of shifting and not shifting will mostly look the same to us, no matter what the grayscale image is. This is due to the scale being very small, that we can't really find a difference between them.

For example, here is a side-by-side comparison of if we apply shifting or we don't on the Lena image, considering **we don't apply logarithmic.**



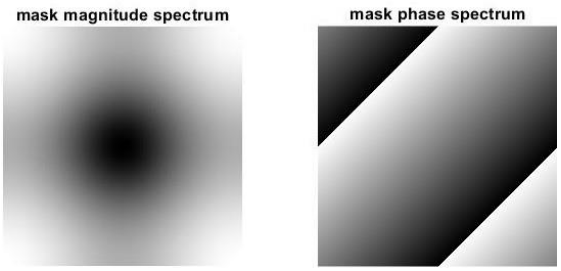
As explained previously, by shifting to the center all the low frequencies will be in the center, however if we don't apply shifting, the low frequencies will be scattered throughout the four edges of the image.

Considering the above statements, for the next few examples we will be only looking at examples considering we do apply logarithmic.

Which it can be seen that the resulted image is a high pass filter that in turn gives us the edges of the original Lena image.

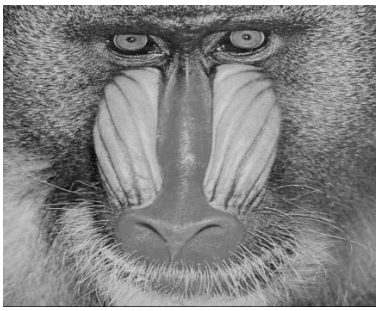
$$c) \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The spectrums for the c filters is as follows:

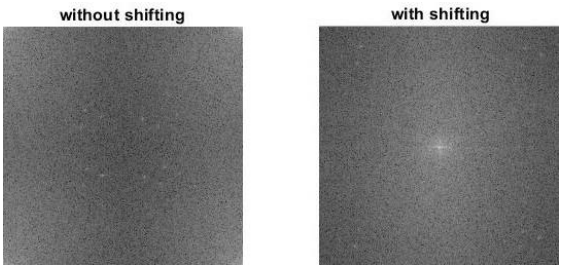


It can be seen that the magnitude of this image has a bit of similarity to the magnitude for the previous filter. This filter gives us a sharpened output compared to the original image, however, the obtained image has a lower contrast compared to the original image. This is due to the filter removing some of the low frequencies in the image.

Finally, we'll look at the baboon image. The original image:



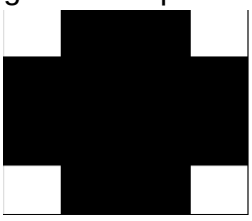
The result of the magnitude spectrum with and without shifting is as follows:



The result of this image is a bit more interesting, due to there being a lot of different horizontal and vertical edges on the baboon's face.

Problem 4.2.2:

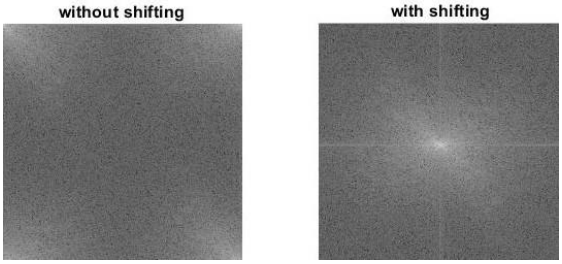
For this problem we don't perform shifting, as stated in the previous section. This will give low frequencies in the edges:



Note that for both the problems,
N = image_width = image_height
And that we'll be discussing the results for the Lena image with different T's.
The original Lena image:

we'll first look at the Lena image. The original image is as follows:

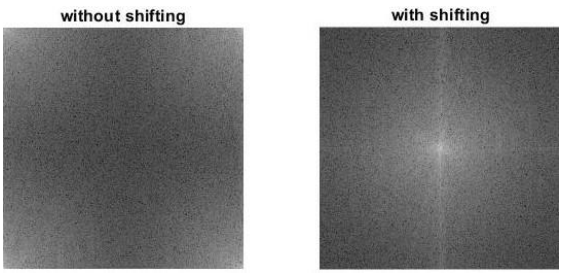
The result of the magnitude spectrum with and without shifting is as follows:



The point we made above that the without shiting, the corners have low frequencies is very apparent in this example.

Next we'll look at the F16 image the original being:

The result of the magnitude spectrum with and without shifting is as follows:



Let $F(k,l) = 0$ for the following regions:

i. $0 \leq \{k \text{ and } l\} \leq TN$;

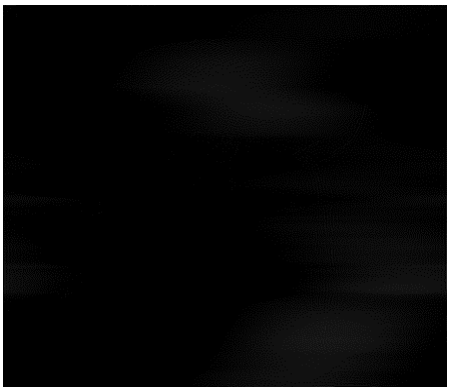
ii. $0 <= k <= TN, and\ (1-T)N \leq l \leq N-1$;

iii. $(1-T)N \leq k \leq N-1\ and\ 0 \leq \{l\} \leq TN$;

iv. $(1-T)N \leq k\ and\ l \leq N-1; T = 1/4, 1/8$

Here, the first part is removing much of the detail in the image, due to zeroing out the upper-left edge of the image which contains much of the information within the image. The other three parts are also zeroing out certain points throughout the image, but they are not as sensitive as the first part.

~~~~ T = 1/4 ~~~~



This image is a very non-detailed representation of the Lena image. As said before, Most of the detail loss is due to the first part of the zeroing out.

If we don't apply the first part, this is what we get:

Which isn't perfect, however it is a much more detailed than what we got by applying all of the zeroing out conditions.

If we only apply the first condition of zeroing out, this is what we obtain:

$F(k,l) = 0\ for\ TN < \{k,l\} < (1-T)N, T = 1/4, 1/8$

This is a Low pass filter due to us only removing frequencies which are in the higher range (not in the corners).

~~~~ T = 1/4 ~~~~

In this obtained result, we see that some of the detail in the image is lost, however we still have an image close to the original one, due to still having most of the low frequencies.

~~~~ T = 1/8 ~~~~

As can be seen, this image has even less detail than the previous image. This is due to use cutting off the low frequencies at a lower range (ranges $32 < k,l < 224$ becoming zero when T = 1/8 as opposed to the ranges $64 < k,l < 192$ becoming zero when T = 1/4)

```
A = 1/16 .* [1,2,1;2,4,2;1,2,1] %only this is seperable due to rank = 1
A1 = 1/4 .* [1;2;1];
A2 = 1/4 .* [1,2,1];
B = [-1,-1,-1;-1,8,1;-1,-1,-1;];
C = [0,-1,0;-1,5,-1;0,-1,0];
img = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\4\Barbar
figure(1); imshow(img,[]);

h = C;

%%%%% THIS EXTRA PART ONLY FOR THE FIRST FILTER %%%%%
% h1 = A2;
% h2 = A1;

PQ = 2*size(img);
%STEP 1 - PERFORM FFT
F = fft2(double(img), PQ(1), PQ(2));
H = fft2(double(h), PQ(1), PQ(2));
%STEP 2 - PERFORM SHIFT
F = fftshift(F);
H = fftshift(H);

figure(2);subplot(1,2,1); imshow(log(abs(F)+1),[]); title('image magnitude spec
subplot(1,2,2); imshow(angle(F),[]);  title('image phase spectrum');
figure(3);subplot(1,2,1);imshow(log(abs(H)+1),[]);  title('mask magnitude spect
subplot(1,2,2);imshow(angle(H),[]); title('mask phase spectrum');
%STEP 3 - APPLY FILTER
F_fH = H.*F;

%%%%% THIS EXTRA PART ONLY FOR THE FIRST FILTER %%%%%
```

```
 %%%%% THIS EXTRA PART ONLY FOR THE FIRST FILTER %%%%%

 % H = fft2(double(h2), PQ(1), PQ(2));
 % H = fftshift(H);
 % F_fH = H.*F;
 % figure(6);subplot(1,2,1);imshow(log(abs(H)+1),[]);  title('mask magnitude sp
 % subplot(1,2,2);imshow(angle(H),[]); title('mask phase spectrum');

 figure(4);subplot(1,2,1); imshow(log(abs(F_fH)+1),[]); title('image magnitude
 subplot(1,2,2); imshow(angle(F_fH),[]);  title('image phase spectrum');
 %STEP 4 - PERFORM INVERSE SHIFT
 F_fh = ifftshift(F_fH);
 ffi = ifft2(F_fH);
 %STEP 5 - RETURN THE IMAGE FROM THE FREQUENCY DOMAIN
 ffi = ffi(2:size(img,1)+1, 2:size(img,2)+1);
 ffi_abs = abs(ffi);
 figure(5); imshow(ffi_abs,[]);
```

Problem 4.2.1:

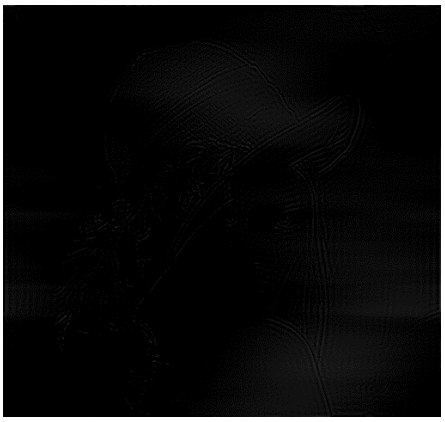
```
clear;
img1 = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\4\Lena.bmp'));
img2 = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\4\F16.bmp'));
img3 = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\4\Baboon.bmp'))
F_not_shifted = fft2(img3);

%figure(1);subplot(1,2,1);imshow(abs(F_not_shifted),[]);title('without shifting');
F_shifted= fftshift(F_not_shifted);
%subplot(1,2,2);imshow(abs(F_shifted),[]);title('with shifting');
%with log
modified_img_shift = log(1+abs(F_shifted));
modified_img_noshift = log(1+abs(F_not_shifted));

figure(3);subplot(1,2,1);imshow(modified_img_noshift,[]);title('without shifting');
subplot(1,2,2);imshow(modified_img_shift,[]);title('with shifting');
```

Which shows just how much detail is lost with that one condition if we compare it to the Original Lena image.

~~~~ T = 1/8 ~~~~



This image shows more detail than what we obtained when T=1/4, yet again because the upper-left corner is now being zeroed out by a smaller range compared to when it is equal to what we obtained previously.

4- Appendix(code)

Problem 4.1.1:

Problem 4.2.2:

```
clear;
img = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\4\Lena.bmp'));
N = size(img,1);
T=1/4;

F = fft2(double(img));

%COMMENT OUT EITHER P1 OR P2

% PROBLEM 1

for k = 1 : N
    for l = 1 : N
        if(T*N<k && k<(1-T)*N && T*N<l && l<(1-T)*N)
         F(k,l) = 0;
        end
    end
end
```

```
 %PROBLEM 2

for k = 1 : N
    for l = 1 : N
        %%%%     2_1       %%%
        if(0<=k && k<=T*N && 0<=l && l<= T*N)
         F(k,l) = 0;
        end

        %%%%     2_2       %%%

       if(0<=k && k<=T*N && (1-T)*N <=l && l<= (N-1))
        F(k,l) = 0;
       end

        %%%%     2_3       %%%

       if( (1-T)*N<=k && k<=N-1 && 0 <=l && l<= T*N )
        F(k,l) = 0;
       end

        %%%%     2_4       %%%

       if((1-T)*N<=k && l<= N-1 )
        F(k,l) = 0;
       end

    end
end
ret_img = ifft2(F);
ret_img = uint8(real(ret_img));
figure(4); imshow(ret_img); imwrite(ret_img,'3_filter2_1.4.png');
```