



山东大学

# School of Information Science and Engineering

2019 – 2020 school year

## Digital Image Processing Experiment Report

Course Name: Digital Image Processing

Title of Experiment: The Fourier Transform and Its  
Appliance

Major and Class Class 3, Communication Engineering

Student ID 201600121116

Student Name 张子超

Date 2019/3/30

# 目录

1. Objectives: .....	3
2. Experiment Content: .....	3
3. Experiment Principle: .....	3
4. Experiment Steps: .....	3
1). Fourier transformation.....	3
2). Approximate impulse function two-dimensional Fourier transform: .....	4
3). Spatial filtering and frequency domain filtering.....	6
5. Conclusions and Experiences: .....	7
6. Appendix (Code) : .....	7

## 1. Objectives:

- 1.Master the DFT transform in two dimensions and its physical significance.
- 2.Handle the MATLAB program of DFT transform in two dimensions.
- 3.Spatial filtering and frequency domain filtering.

## 2. Experiment Content:

Learn to use functions `fft2`, `ifft2`, `abs`, `angle`, `fftshift`, `imfilter`, `fspecial`, `freqz2`, and the method for displaying in logarithmic form. Implement the fourier transform forward and reverse, Gaussian low-pass filtering and the laplacian high-pass filtering.

## 3. Experiment Principle:

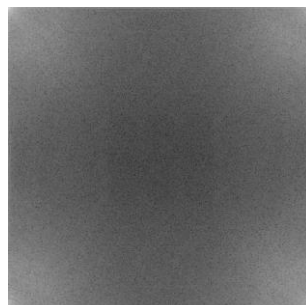
The image can be processed exactly as how the signal in one dimension is processed. Like the signals, images also have spectrum, representing the variability of one image. Thus, filtering can also be applied in digital processing.

## 4. Experiment Steps:

### 1). Fourier transformation

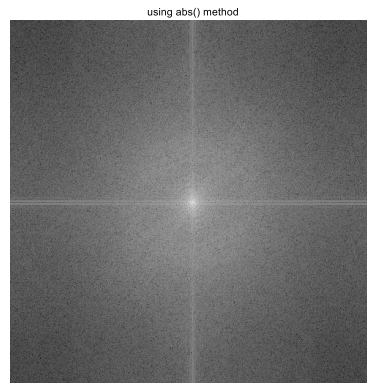
Load and display image `Fig0316(3)(third_from_top).tif` and conduct a FFT in two dimension on this figure with an output `F`. Then shift its DC component to the center of spectrum, get `F1`. Calculate its real part `RR`, imaginary part `II` and phase angle `Angle`. Calculate its magnitude `A1=abs(F1)` and `A2=sqrt(RR.^2+II.^2)` in two ways and display them separately to draw a comparison between them.

As we know, in digital frequency domain, the highest frequency lies in  $\pi$ , and origin point represents the lowest frequency. So directly use the function `fft2` will bring the result as below:



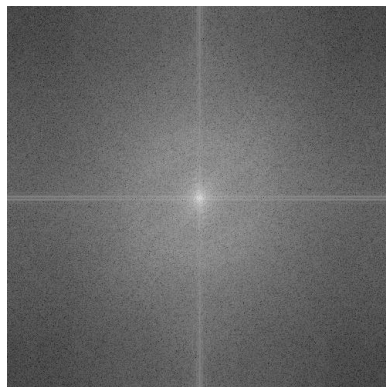
The spectrum embodies our theory, the brightest part are the four corners while the darkest part is the center.

So this is the right place for `fftshift`. This function shifts the highest frequency to the central part, making it more easier to tell the spectrum composition.



Now the spectrum do look better.

There is another way to obtain the magnitude of spectrum, that is, extract the real part and the imaginary part of spectrum separately and add them together after squared, at last, square it.



We can see that there's no difference between them.

2). Approximate impulse function two-dimensional Fourier transform:

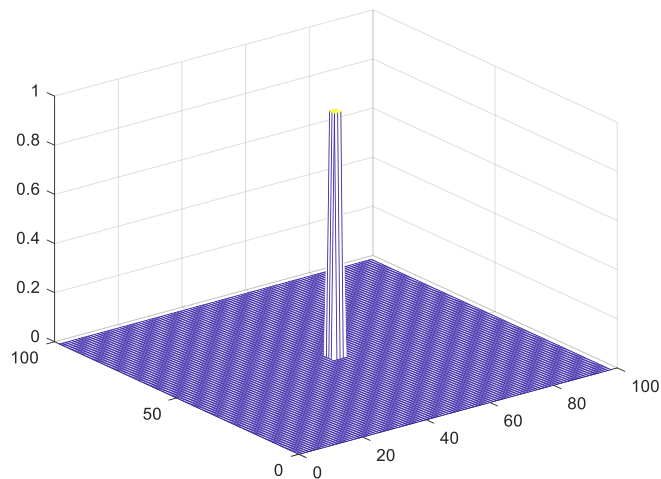
`A=zeros(99,99);A(49:51,49:51)=1`. Conduct the fourier transform `B` of `A`, then shift the DC component of `B` to the center, with an output `B1`. Display the logarithm of `A` and `B`'s magnitude separately with function `imshow` and `mesh`.

`mesh(X,Y,Z)` draws a wireframe mesh with color determined by `Z`, so color is proportional to surface height. If `X` and `Y` are vectors, `length(X) = n` and `length(Y) = m`, where `[m,n] = size(Z)`. In this case, `(X(j), Y(i), Z(i,j))` are the intersections of the wireframe grid lines; `X` and `Y` correspond to the columns and rows of `Z`, respectively.

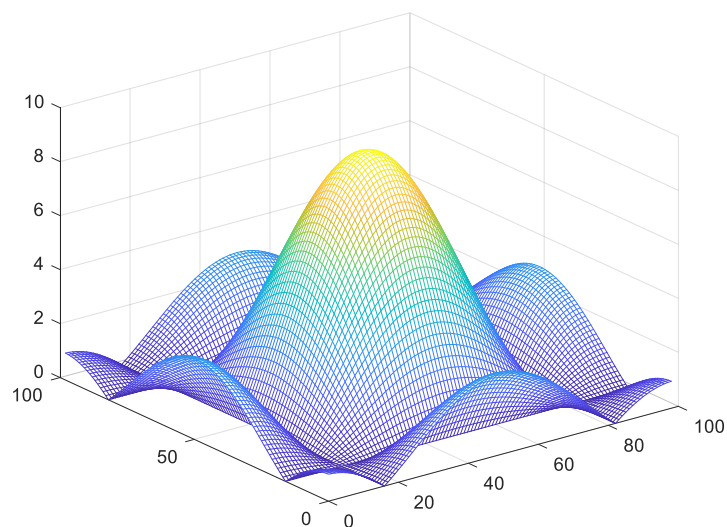
First of all, get `A`:



Use the same method, we get the spectrum of A.



With the function `mesh`, we can get the solid figure of A like above, there's one thing should be emphasized that `mesh` function requires three input: X, Y, and Z, X and Y can be structured by `meshgrid` function, but Z should be a function whose independent variables are X and Y. So I write this code: `mesh(X,Y,A)`. Used A to substitute Z. As well, the spectrum can also be displayed this way:



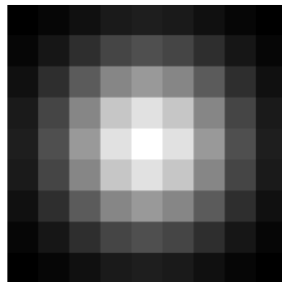
Though, I didn't use the logarithm of B in order to correspond to the figure in the instruction book.

### 3). Spatial filtering and frequency domain filtering.

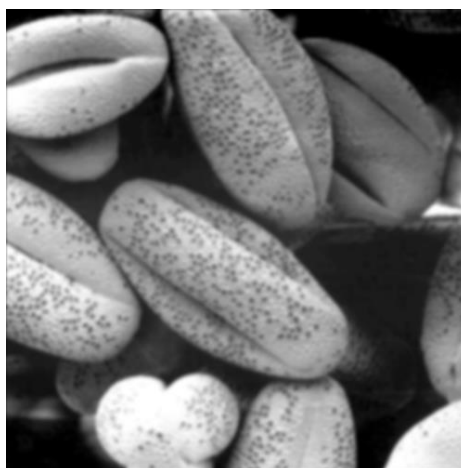
Spatial filtering: obtain a gaussian filter whose size is  $9 \times 9$  and standard deviation is 2 with the function `fspecial`.

Frequency domain filtering: use function `freq2` to obtain the frequency domain form ( $256 \times 256$ ) of aforementioned gaussian filter and filter the image in frequency domain. At last, conduct an anti fourier transform.

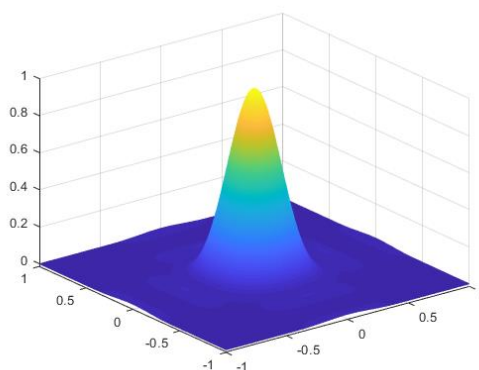
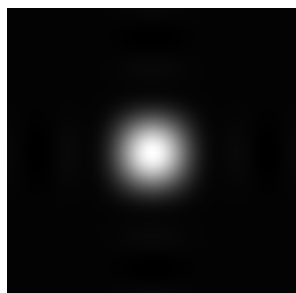
Use function to create a  $9 \times 9$  gaussian filter `w`, display it.



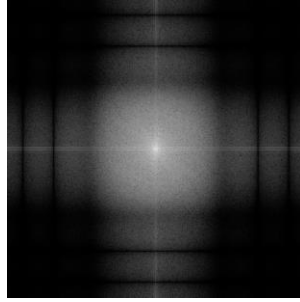
Then use function `imfilter` to filter the figure, the result is as follows:



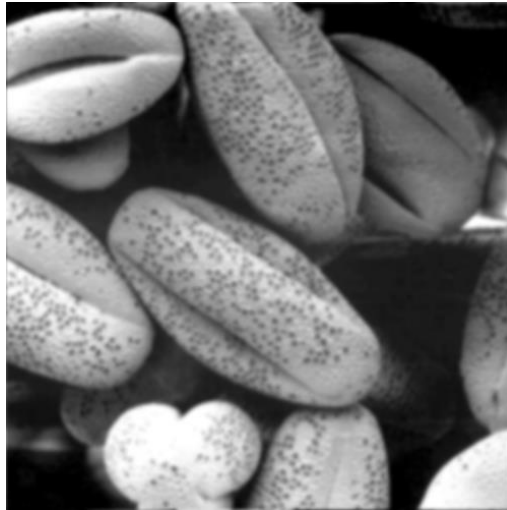
The image turns out a little fuzzier than before. Then use the function `freqz2` to create a frequency domain filter `W`. And the solid figure is rather easy to draw.



At last, simply multiple the `W` and spectrum of figure together is OK. The result is as follows



And conduct the `ifft2` function, the result is rather surprising.



It's quite the same with the result we got previously with time domain filtering method.

## 5. Conclusions and Experiences:

During the experiment, I made some mistakes that I should not have made, like I mistaken result of `fft` as real number, which caused the final result screwed up. The result of frequency domain filtering and the time domain filtering is the same, cause the product in frequency is equivalent to the convolution in time domain. The high frequency part embodies the varying part of the image while the DC component embodies the plain part of image. So when I filtered the image with a low-pass filter, the image got fuzzier, which is because the boundaries are the most dramatic parts.

## 6. Appendix (Code) :

Part 1:

```

[y,mapy]=imread('G:\desktop documents\2018 大三下\数字图像处理资料 For students\For students\程序与
图像\图像库\Fig0316(3)(third_from_top).tif);

figure;

imshow(y);

F=fft2(y);% non-shifted
logF=log(1+abs(F));

figure;

imshow(logF,[]);

F1=fftshift(F);%shifted figure
logF1=log(1+abs(F1));

figure;

imshow(abs(logF1),[]);

title('using abs() method')

realF=real(F1)
imagF=imag(F1)
awF1=sqrt(realF.^2+imagF.^2)
logawF1=log(1+awF1)

figure

imshow(logawF1,[])

Part 2:

A=zeros(99,99)

A(49:51,49:51)=1

figure

imshow(A,[])

fftA=fft2(A)

fftAs=fftshift(fftA)

```



```
logfftAs=log(1+abs(fftAs))
```

```
figure
```

```
imshow(logfftAs)
```

```
[X,Y]=meshgrid(0:98)
```

```
Z=0:0.1:1
```

```
figure
```

```
mesh(X,Y,abs(fftAs))
```

Part 3:

```
fdomainy=fft2(y);
```

```
fdomainysh=fftshift(fdomainy);
```

```
figure
```

```
imshow(log(1+abs(fdomainysh)),[]);
```

```
w=fspecial('gaussian',[9 9],2);
```

```
figure
```

```
imshow(w,[])
```

```
set(gca,'position',[0 0 1 1])
```

```
fi1=imfilter(y,w,'conv','same');
```

```
figure;
```

```
imshow(fi1,[]);
```

```
[W,f1,f2]=freqz2(w,500,500);
```

```
figure
```

```
mesh(f1,f2,W)
```

```
figure
```

```
imshow(W,[])
```

```
freq_dom_fil=W.*fdomainysh;
```

```
figure
```

```
imshow(log(1+abs(freq_dom_fil)),[])
```

```
ifft2=ifft2(freq_dom_fil);
```

```
figure
```

```
imshow(abs(ifft2),[])
```