1 Exercises

Please answer the following questions in the report.

1.1 Rotation (10 Points)

Fig. 1(b) was generated by:

1. Multiplying Fig. 1(a) by (-1)^$(x+y)$.;

2. Computing the discrete Fourier transform;

3. Taking the complex conjugate of the transform;

4. Computing the inverse discrete Fourier transform;

5. Multiplying the real part of the result by (-1)^$(x+y)$.

Explain (mathematically) why Fig. 1(b) appears as it does.

Sol:

This is a example of Fourier transformation, first we multiply it with (-1)^$(x+y)$, it moves the edge to the center, in the step five, we need to multiplying the real part of the result by (-1)^$(x+y)$ again to restore the image. Thus these two steps' influence can mutually counteract and have no influence to the image.The change happens in step 3:

$$G(u,v)=H(u,v)\ F(u,v)$$

Here, H(u,v) just calculate the complex conjugate of F(u,v).

According to step 2, F(u,v) = $\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} f(x,y)e^{-j2\Pi(ux/M+vy/N)}$, after step 3, $\widetilde{F}(u,v)$

= $\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} f(x,y)e^{j2\Pi(ux/M+vy/N)}$, in this way, we need to do some operation to

translate it back, we replace the x with M-x, y with N-y, we get

According to the Fourier transform, $\widetilde{F}(u,v)$ =

$\frac{1}{MN}\sum_{M-\mathrm{x}=1}^{M}\sum_{N-y=1}^{N} f(M-x,N-y)e^{j2\Pi(u(M-x)/M+v(N-y)/N)}$ =

$\frac{1}{MN}\sum_{M-\mathrm{x}=1}^{M}\sum_{N-y=1}^{N} f(M-x,N-y)e^{j2\Pi(-ux/M-yv/N)}$ =

$\frac{1}{MN}\sum_{\mathrm{x}=0}^{M-1}\sum_{y=0}^{N-1} f(M-x,N-y)e^{-j2\Pi(ux/M+yv/N)}$ = G(u,v)

And then in step 4 we computing the inverse discrete Fourier transform, getting G(u,v) = $f(M-x,Y-n)$.

Thus this transformation just put each pixel in (x,y) to (M-x, N-y), which is the central symmetry of origin image.

1.2 Fourier Spectrum (10 Points)

The (centered) Fourier spectrum in Fig. 2(b) corresponds to the original image Fig. 2(a), and the (centered) Fourier spectrum in Fig. 2(d) was obtained after Fig. 2(a) was padded with zeros(shown in Fig. 2(c)). Explain the significant increase in signal strength along the vertical and horizontal axes of Fig. 2(d) compared with Fig. 2(b).

Sol:

After Fourier transformation, the original image is displayed in its frequency domain, we move the edge frequency to the center by multiplying with a (-1)^(x+y), since the message of the image is expressed by the low frequency, thus the center is light and other place is dark, when padding with 0, we emphasize on the edge of the origin image, which highlight the difference between origin image's edge with padding 0, thus there is a corresponding line in frequency domain representing the difference, when multiplying with a (-1)^(x+y), these lines was moved to the center which gives out a clear increase in signal strength. Also, padding can be viewed as low frequency, thus the part of low frequency is enhanced.

1.3 Lowpass and Highpass (5 2 = 10 **Points)**

1. Find the equivalent filter H(u; v) in the frequency domain for the following spatial filter:

2. Is H(u; v) a low-pass filter or a high-pass filter? Prove it mathematically.

Sol:

1、We can operate Fourier transform to get h(x,y),

$h(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}H(u,v)e^{j2\Pi(ux/M+vy/N)}$, after doing some operation, we can get

$H(u,v)=\sum_{x=0}^{2}\sum_{y=0}^{2}h(x,y)e^{-j2\Pi(ux/3+vy/3)}$, *then we apply this problem's matrix to it to get the result matrix:*

$$\begin{matrix} 0 & 0 & 0 \\ 6-2\sqrt{3}i & -\frac{3}{2}-\frac{\sqrt{3}}{2}i & \sqrt{3}i \\ 6+2\sqrt{3}i & -\sqrt{3}i & -\frac{3}{2}+\frac{\sqrt{3}}{2}i \end{matrix}$$

(2) It is a high-pass filter, according to the figure above, it satisfy the equation:

$$H(u,v) = \begin{matrix} 0 & \mathrm{D(u,v)} < \mathrm{D0} \\ 1 & \mathrm{D(u,v)} > \mathrm{D0} \end{matrix}$$

Since the origin is (0,0), it is zero, which means that low frequency can't pass, and the points far from (0,0) is not equal to 0, which means that high frequency can pass the filter. In this way, it is a high-pass filter.

1.4 Exchangeability (10 Points)

Suppose the high-frequency-emphasis filtering is applied to an image for edge sharpening, and then the histogram equalization is applied to enhance contrast. Theoretically, would the result be the same if the order of these two operations were reversed? Prove it mathematically. Hint: Refer to Chapter 2.6.2 \Linear versus Nonlinear Operations".

Sol:

First, we need to have a concept that linear operation can exchange position with no influence on the result while non-linear operation can't exchange position, once the operation order is changed, the result is different.

The high-frequency-emphasis filtering is multiplying the origin matrix with another matrix H(x,y), since the matrix multiplying is linear operation, the high-frequency-emphasis filtering is linear.

The histogram equalization is not linear operation, since it involves rounding and mapping, consider a special case, there are two mutual complement matrix, which mean for matrix A、B, A(x,y)+B(x,y) = 255, obviously if we first doing histogram equalization on A、B, then add them up, it is not equal to doing histogram equalization on (A+B).

The result of histogram equalization on (A+B) is equal to (A+B) =

```
255   255   255
255   255   255
255   255   255
```

but the histogram on A and B is two averaged histogram, the corresponding position's value has been changed, thus A(x,y) + B(x,y) is not always equal to 255, thus they are not equal. In this way, the histogram equalization is non-linear operation.

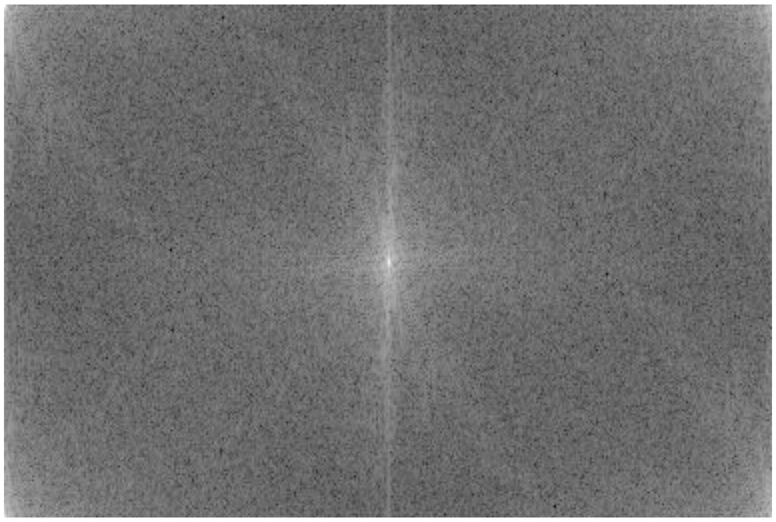
According to the discussion above, the result is different if the order of these two operations were reversed.

2 Programming Tasks

2.2 Fourier Transform

1. Perform DFT and manually paste the (centered) Fourier spectrum on your report.

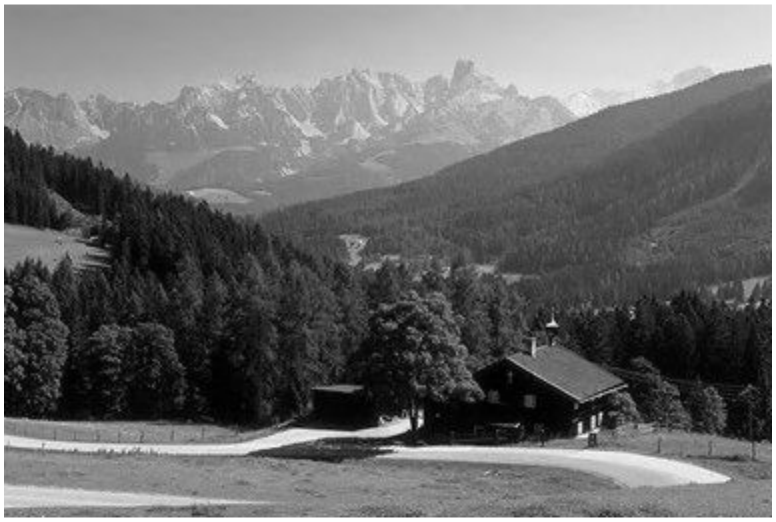
The corresponding Fourier spectrum is shown below:



2. Perform IDFT on the result of the last question, and paste the real part on your report.

The origin image is shown below:

After performing dft2d and idft2d, the corresponding image is shown below:



They are just the same, because the DFT and IDFT are invertible, thus a matrix

A,

A = IDFT(DFT(A)), but there may be a little different due to the rounding

performed by computer.

3. Detailedly discuss how you implement DFT / IDFT.

We can finish it by using the theorem F(u,v) = $\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} f(x,y)e^{-j2\Pi(ux/M+vy/N)}$ and f(x,y) = $\frac{1}{MN}\sum_{u=1}^{M}\sum_{v=1}^{N} f(u,v)e^{j2\Pi(ux/M+vy/N)}$, in this way, we can perform DFT and IDFT using quad cycle: (i) for DFT, to get each point in F(u,v), we visit each point in f(x,y), multiply it with $e^{-j2\Pi(ux/M+vy/N)}$, then add them up to get the result. (ii) for IDFT, to get each point in f(x,y), we visit each point in F(u,v), multiply it with $e^{j2\Pi(ux/M+vy/N)}$, then add them up to get the result.

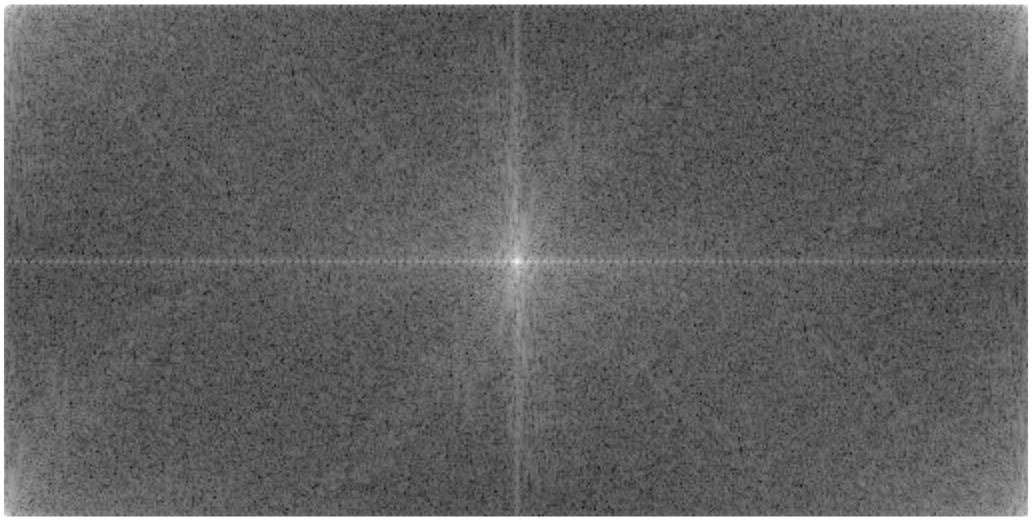
But this can be very slow, for a 384*256 matrix, it will take 30 minutes to calculate it, which is not tolerant.

To improve it, we can use two triple cycle to replace it. For DFT, we first perform Fourier transform on row, then on column. For each row in f(x, :), we visit each point in column, using $F(x,v) = \sum_{y=0}^{N-1} f(x,y)e^{-2\Pi vyj/N}$ to get result for F(x,v), and then for each column in F(x,v), we visit each point in the row, using $F(u,v) = 1/M\sum_{x=0}^{M-1} F(x,v)e^{-2\Pi uxj/M}$ to get the result for F(u,v), which is the output matrix we want. Due to the property of Fourier transformation the separate of performing transformation on row and column is equal to performing transformation on row and column simultaneously.

It is much fast than the quad cycle, it only takes about 20 seconds to calculate the result compared with 30 minutes.

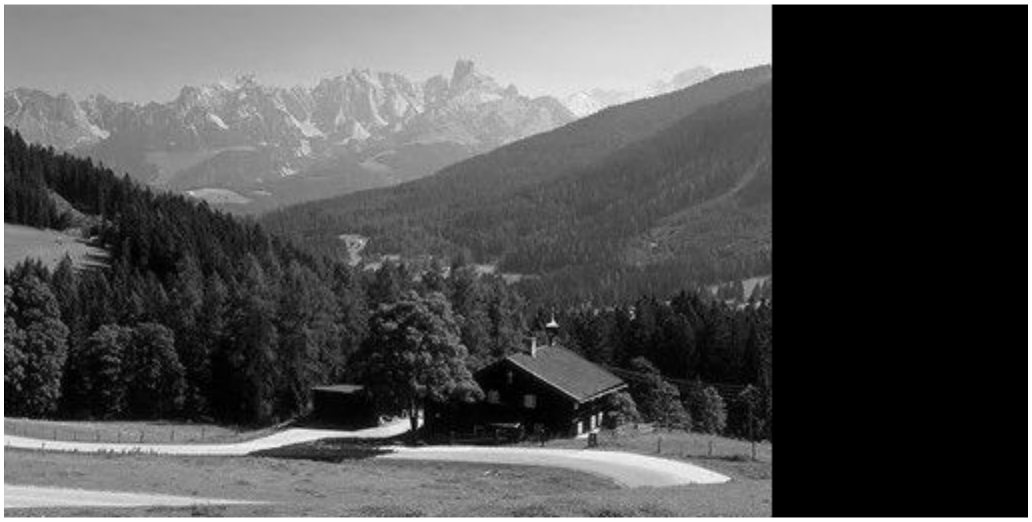
2.3 Bonus: Fast Fourier Transform

1. Perform FFT and manually paste the (centered) Fourier spectrum on your report.



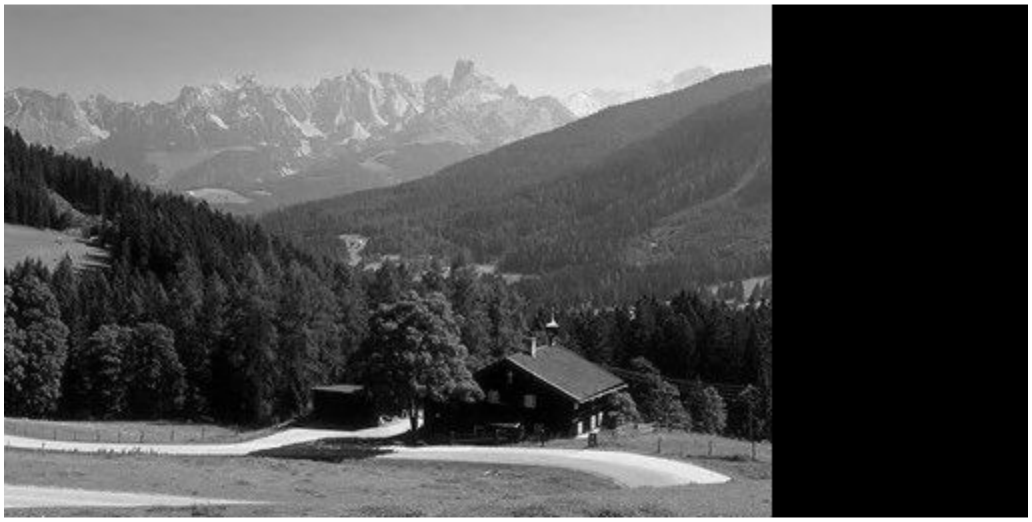
2. Perform IFFT on the result of the last question, and paste the real part on your report.

The origin image after padding the image with zeros to obtain a proper size:



256*512

After performing fft2d and ifft2d, the resulting image is shown below:

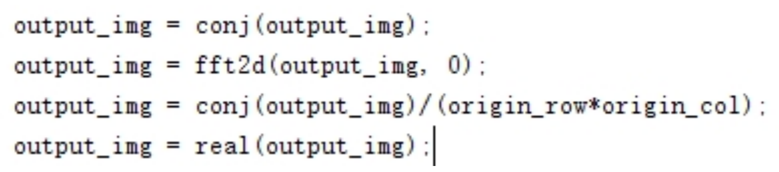


3. Detailedly discuss how you implement FFT / IFFT.

The algorithm to perform FFT worth studying. I first read the ppt and first finish FFT for one dimension array, it use the successive doubling method, it use $W_M = e^{2\Pi j/M} = e^{2\Pi j/(2M/2)}$, we can transfer 2M data into two M data, in this way 4M*M reduces into 2M*M.

For FFT, In matlab, we first operate index arithmetic, for each point in array, we get its index, represented into bit form, then revered the bit form, for example, 6 = 110, reversed form is 011, thus we exchange array[6] with array[3]. Then in second step, we operate butterfly computation, it is like iteration, using F(u) = 1/2[$F_{even}(u) + F_{odd}(u)W_{2K}^{u}$] to get the result. The two dimension matrix can use one dimension to achieve. We apply one dimension algorithm on the two dimension matrix 's each row and column, in this way, the FFT is accomplished.

For IFFT, we can use FFT to get the result, 1/M $f^*(x) = 1/M\sum_{u=0}^{M-1} F^*(u)e^{-2\Pi jux/M}$, we first get the conjugate of F(u,v), perform the FFT on it to get $1/Mf^*(x)$, then multiply it with M and get the conjugate of it, we get the f(x,y), which is the result we want. The code can be reused:



```
output_img = conj(output_img);
output_img = fft2d(output_img, 0);
output_img = conj(output_img)/(origin_row*origin_col);
output_img = real(output_img);
```

The FFT is much faster than DFT, although the two triple cycle can calculate the result in 30 seconds, the FFT only uses 3 seconds to calculate a 256*384 matrix.

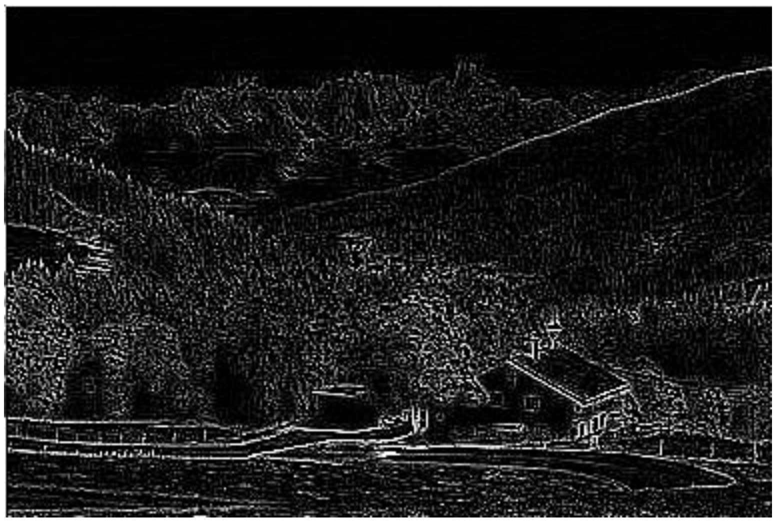
2.4 Filtering in the Frequency Domain

1. Smooth your input image with an 7   7 averaging filter. Paste your result on the report.

After smooth the input image with 7*7 averaging filter, the result is shown below:

2. Sharpen your input image with a 3 3 Laplacian filter and then paste the result.

After sharpen the input image with laplace filter, the result is shown below:



3. Detailedly discuss how you implement the filtering operation.

It is based on the DFT and IDFT, first we need to padding the two matrix to avoid wraparound error, if image matrix f(x,y) is M*N, filter h(x,y) is A*B, then padding the two matrix to (M+A,N+B) with 0.

Then we use the property of convolution theorem, the convolution in spiral domain is equal to the product in frequency domain. We use DFT to get F(u,v) and H(u,v), then get the product of them, denoted as G(u,v), then we perform IDFT to get g(x,y), then we obtain the M*N part from g(x,y), obtain the real part of it. It is the result we want, in this function, the filter can be any filter, we choose average 7*7 and laplace filter to get the result.