# Digital process report 3

*Cheng Xinhao*

Dalian University of Technology

2019/3/29

# 1.Compare in spatial domain and in frequency domain

Correspondence between the convolution of the frequency domain and the spatial domain

$$f(x,y)*g(x,y)=F(u,v)H(u,v)$$
$$f(x,y)g(x,y)=F(u,v)*H(u,v)$$

Spatial domain: Operates directly on the pixels of the original, input image.The template is moved pixel by pixel in the image, and each element of the convolution kernel is multiplied by the corresponding position element of the image matrix and the result is accumulated, and the convolution result of the pixel corresponding to the center of the template is accumulated. In general, convolution is the process of weighted averaging of the entire image. The value of each pixel is obtained by weighted averaging of itself and other pixel values in the neighborhood.

Frequency domain: The frequency domain is a space defined by the Fourier transform and the frequency variable (u, v). The frequency domain filtering process is: Fourier transforming the image, converting to the frequency domain, filtering using the filter function in the frequency domain, and finally Transform the result back to the spatial domain

Relationship: Given a frequency domain filter, it can be inversely Fourier transformed to obtain a corresponding spatial domain filter. The filtering is more intuitive in the frequency domain, but the spatial domain is suitable for using a smaller filtering template to improve the filtering speed. Because the frequency domain filter is more efficient than the spatial domain filter under the same size, the spatial domain filtering requires a smaller size template approximation to obtain the desired filtering result.

## Here are the demo codes

```
ker = -ones(3,3);
ker(2,2) = 9;
Ik = uint8(conv2(I,ker,'same'));
Ikf = uint8(conv2(I,ker));
```

convolve in spatial domain

```
If = fft2(I);
keri = zeros(256,256);
keri(1:3,1:3) = ker;
kerif = fft2(keri);
Iff = If.*kerif;
Iffi = uint8(ifft2(Iff));
```

FFT convolve

## We perform the four pictures

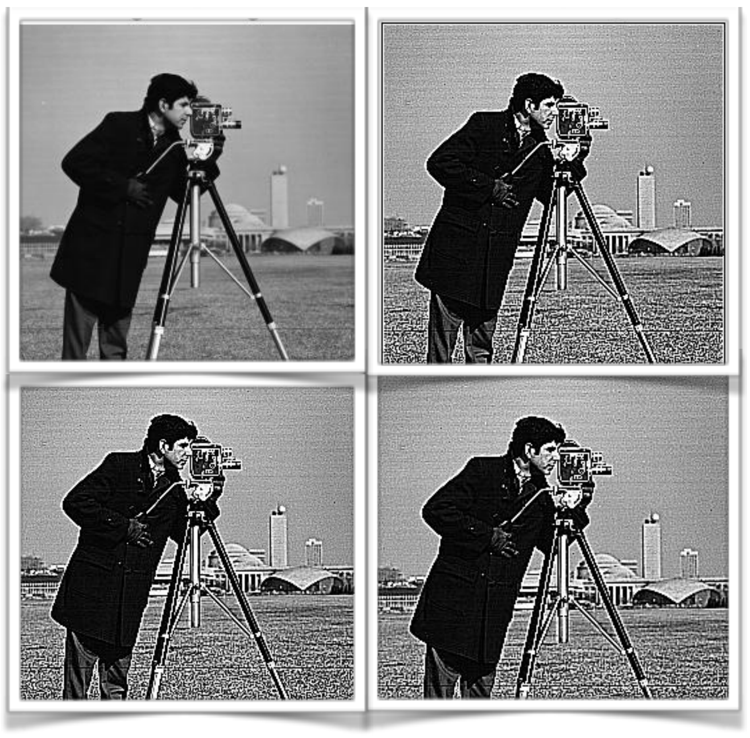
From left to right, from top to bottom:

1.original image

2.convolved image  full

3.convolved image same size as input

4.FFT convolved image



We conclude that filtering from the spatial domain and filtering in the frequency domain can give the same result.

Then we pick a picture, convert it to YCBCR color, and pick the Y channel. And we take spatial domain gaussian filter and frequency domain gaussian filter to deal with the Y channel, then combine them with CB channel and CR channel. At last, we inverse the two results to the RGB color, and compare them.

Here are the demo codes

```
close all
clear
clc
im1 = imread('dog1.jpg');
im1 = rgb2ycbcr(im1);
im1_2 = im1(:,:,1);
%%
h1 = fspecial('gaussian',5,2.0);
im1_3 = uint8(conv2(im1_2,h1,'same'));
im1(:,:,1) = im1_3;
im1 = ycbcr2rgb(im1);
figure;
imshow(im1);
%%
im2 = imread('dog1.jpg');
im2 = rgb2ycbcr(im2);
im2_1 = im2(:,:,1);
[m,n]=size(im2_1);
Do=0.1*m;
F=fft2(im2_1);


u=0:(m-1);
v=0:(n-1);
idx=find(u>m/2);
u(idx)=u(idx)-m;
idy=find(v>n/2);
v(idy)=v(idy)-n;
[V,U]=meshgrid(v,u);
D=sqrt(U.^2+V.^2);

H=exp(-(D.^2)./(2*(Do^2)));

im2_2 = F.*H;
im2_3=real(ifft2(im2_2));
im2_3=uint8(im2_3);
im2(:,:,1) = im2_3;
im2 = ycbcr2rgb(im2);
figure;
imshow(im2);
```

## We perform the results:

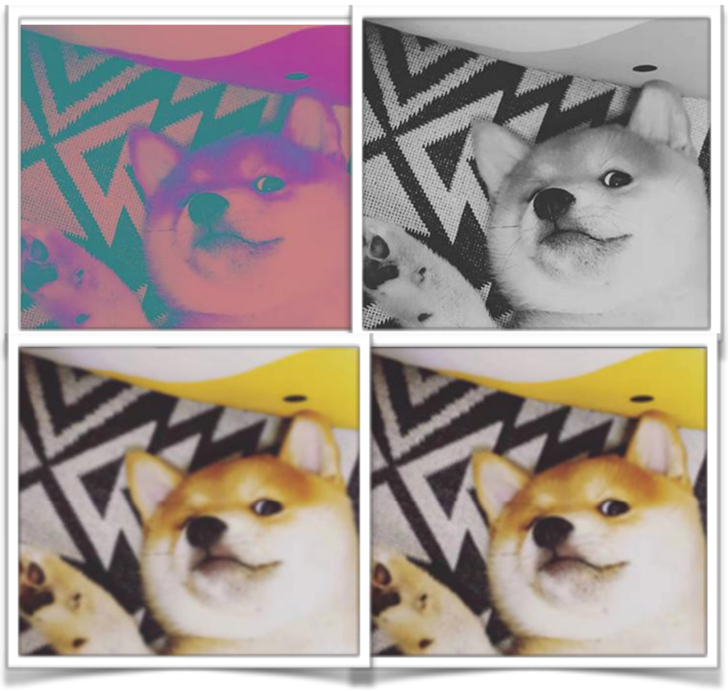
the YCBCR color picture:

From left to right, from top to bottom:

1.The YCBCR color image

2.The Y channel of YCBCR

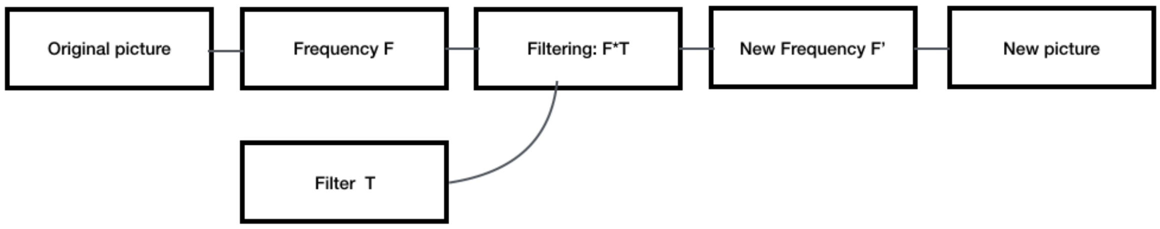
3. compare the spatial convert and frequency convert.



We can conclude that the two images are identical.

## 2.Different method of filtering in frequency domain

Frequency domain filtering is a method of processing images in the frequency domain:



## Ideal low/high pass filter: 
The high-pass filter is: let high-frequency information pass, filter low-frequency information; low-pass filtering is reversed.

D0 represents the passband radius, and D(u,v) is the distance from the center of the spectrum (Euclidean distance)

M and N represent the size of the spectrum image, and (M/2, N/2) is the center of the spectrum.

Formula:

$$D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$$

$$H(u,v) = 1(D < D0)$$

$$H(u,v) = 0(D >= D0)$$

## Butterworth high/low pass filtering: 
The Butterworth filter is a signal processing filter with a very flat frequency response curve. It is also known as the maximum flat filter. This filter was first proposed by the British engineer and physicist Steffen Butterworth in the 1930 paper "The Theory of Filter Amplifiers”.

Formula:

$$H(u,v) = \frac{1}{1 + [D(u,v)/D0]^{2n}}$$

The shape of the filter can be changed by the power factor n. The larger n is, the closer the filter is to the ideal filter

## Gaussian high/low pass filtering: 
The particularity of the Gaussian function: the Gaussian function Fourier transform is still a Gaussian function, but the standard deviation has changed, the larger the standard deviation in the frequency domain (the wider the Gaussian function), the smaller the standard deviation of the spatial domain after the transformation (the narrower the Gaussian function)

$$H(u,v) = e^{\frac{-D(u,v)^2}{2D0^2}}$$

## Here are the demo codes

```matlab
clear;
clc;
I = imread('cameraman.tif');
[m,n]=size(I);
Do=0.1*m;
F=fft2(I);
%figure;
%imshow(log(fftshift(abs(F))+1),[]);

u=0:(m-1);
v=0:(n-1);
idx=find(u>m/2);
u(idx)=u(idx)-m;
idy=find(v>n/2);
v(idy)=v(idy)-n;
[V,U]=meshgrid(v,u);

D=sqrt(U.^2+V.^2);%%Distance matrix, a matrix representing the distance between each pixel and the center of the image
%%
way=inputdlg('input the name of filter: ILPF,BLPF,GLPF,IHPF,BHPF,GHPF','filter');
way=char(way);
switch way
    case 'ILPF'
        H=double(D<=Do);%%Ideal low pass filter
    case 'BLPF'
        H=1./(1+(D./Do).^4);%%butterworth low pass filter
    case 'GLPF'
        H=exp(-(D.^2)./(2*(Do^2)));%%guassian high pass filter
    case 'IHPF'
        H=double(D>=Do);%%Ideal high pass filter
    case 'BHPF'
        H=1./(1+(Do./D).^4);%%butterworth high pass filter
    case 'GHPF'
        H=1-exp(-(D.^2)./(2*(Do^2)));%%guassian high pass filter
    otherwise
        errordlg('false');
```

```
end
%%
Fout=F.*H;
%imshow(log(fftshift(abs(F))+1),[]); %Display the spectrogram before filtering
%figure;
%imshow(log(fftshift(abs(Fout))+1),[]); %Display the spectrogram after filtering
f=real(ifft2(Fout));
f=uint8(f);
figure,imshow(f),title([way,'after filtering']);
figure,imshow(I),title('before filtering');
```

Take the Ideal low pass filter as an example:(other filters are also available in the code)

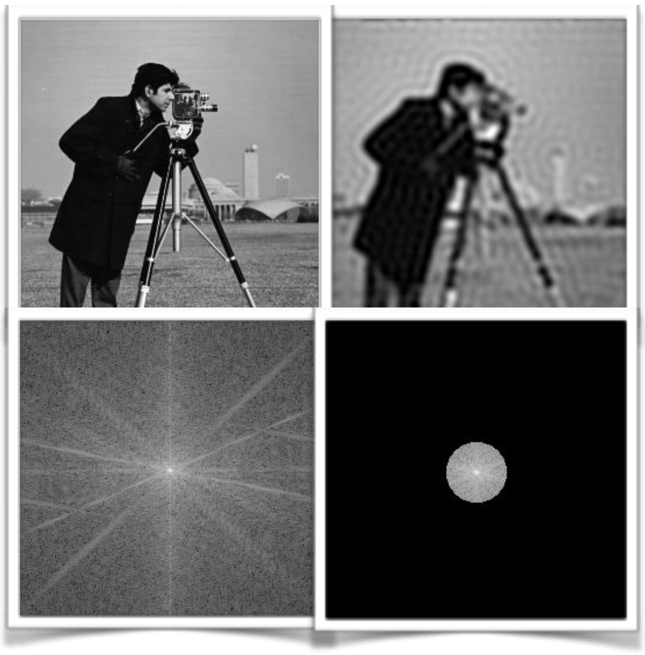
From left to right, from top to bottom:

1.original image

2.image after Ideal low pass filtering

3.original Spectrum

4.spectrum after filtering



## 3.DCT processing

DFT is a discrete Fourier transform for discrete signals and spectra. DFT is a change in DTFT, in fact, it is to change the continuous time t to nT. The reason is that computer works in a digital environment. It is impossible to see or process continuous signals in reality, only to perform discrete calculations, and to approximate the continuous signal as much as possible in terms of authenticity. So DFT is created for us to analyze signals with tools. Usually we have very few opportunities to use DTFT directly.

DCT is a form of DFT. The so-called "cosine transformation" is in the DTFT Fourier series expansion, if the function to be expanded is a real function, then the Fourier series only contains the cosine term, and then discretizes it (DFT) to derive the cosine transform. It is therefore called the discrete cosine transform (DCT). In fact, DCT is a subset of DFT. DCT is used for voice and image processing.

Formula:

One-dimensional transformation:

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos[\frac{\pi(2x+1)u}{2N}]$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u \neq 0 \end{cases}$$

Inverse transformation:

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) C(u) cos[\frac{\pi(2x+1)u}{2N}]$$

Two-dimensional transformation:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos[\frac{\pi(2x+1)u}{2N}] cos[\frac{\pi(2y+x)v}{2N}]$$

Inverse transformation:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) cos[\frac{\pi(2x+1)u}{2N}] cos[\frac{\pi(2y+1)v}{2N}]$$

## Here are the demo codes

```
close all;
clear;
clc;
I = imread('cameraman.tif');
figure;
imshow(I);
Inew = dct2(I);
%%
```

```
figure
imshow(log(abs(Inew)),[])
colormap(gca,jet(64))
colorbar
title('  most of the energy is in the upper left corner');
%%
way=inputdlg('input remove low or high frequencies ','DCT method');
way=char(way);
switch way
    case 'low'
        Inew(abs(Inew) > 100) = 0;%%low filter
    case 'high'
         Inew(abs(Inew) < 10) = 0;%%high filter
    otherwise
        errordlg('false');
end
%%
Inew3 = dct2(Inew);
figure;
imshowpair(I,Inew3,'montage');
title('Original Grayscale Image (Left) and Processed Image (Right)');
```

## If we remove the low frequency:

From left to right, from top to bottom:

1.original image

2.the energy is in the upper left corner

3.compare with imagine removed low frequency