CSE 586A Problem Set 1

Name: Jin Yang
UID: 467527

1 Prove the properties of convolution

1.1 Associativity

Here I prove that $(f * g) * h = f * (g * h)$

Solution $((f * g) * h)(t) = \int_0^t (f * g)(s)h(t-s)ds$

$= \int_{s=0}^{t} (\int_{u=0}^{s} f(u)g(s-u)du)h(t-s)ds$ definition of $(f * g) * h$

$= \int_{s=0}^{t} \int_{u=0}^{s} f(u)g(s-u)h(t-s)duds$

$= \int_{u=0}^{s} \int_{s=0}^{t} f(u)g(s-u)h(t-s)dsdu$ change the order of two integrations

$= \int_{u=0}^{t} \int_{s=0}^{t} f(u)g(s)h(t-s-u)dsdu$

$= \int_{u=0}^{t} \int_{s=0}^{t-u} f(u)g(s)h(t-s-u)dsdu$ property of integration

$= \int_{u=0}^{t} f(u)(\int_{s=0}^{t-u} g(s)h(t-u-s)ds)du$

$= \int_{u=0}^{1} f(u)(g * h)(t-u)du$

$= (f * (g * h))(t)$

1.2 Distribution

Here I prove that $f * (g + h) = f * g + f * h$

Solution: If $g = f * h$ then can can conclude $g(x) = \int f(t)h(x-t)dt$

$f * (g + h) = (f * (g + h))(t)$

$= \int_0^t f(s)(g + h)(t-s)ds$ definition of $f * (g + h)$

$= \int_0^t f(s)g(t-s)ds + \int_0^t f(s)h(t-s)ds$ property of integration

$= f * g + f * h$

1.3 Differentiation Rule

Here I prove that $(f * g)' = f' * g = f * g'$

Solution: I will prove the first part first.

$(f * g)' = \frac{\partial}{\partial t}(f * g)(t)$

$= \frac{\partial}{\partial t} \int_0^t f(s)g(t-s)ds$ definition of $(f * g)(t)$

$= \int_0^t f(s)\frac{\partial}{\partial t}g(t-s)ds$ property of integration and derivation

$= (f * \frac{\partial}{\partial t}g)(t)$

$= f * g'$

Then I will prove the second part.

$(f * g)' = \frac{\partial}{\partial t}(f * g)(t)$

$= \frac{\partial}{\partial t}\int_0^t f(s)g(t-s)ds$ definition of $(f * g)(t)$

$= \int_0^t \frac{\partial}{\partial t}f(s)g(t-s)ds$ property of integration and derivation

$= (\frac{\partial}{\partial t}f * g)(t)$

$= f' * g$

1.4 Convolution Theorem

Here I prove that $F(g * h) = F(g)F(h)$

Solution: $F(g * h) = F(\int_{-\infty}^{+\infty} g(x)h(u-x)dx)$

$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x)h(u-x)dx e^{2\pi isu}du$ I assume that $t = u - x$

$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x)h(t)e^{2\pi is(x+t)}dxdt$ replace $u - x$ with t

$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x)e^{2\pi isx}h(t)e^{2\pi ist}dxdt$ property of index

$= \int_{-\infty}^{+\infty} g(x)e^{2\pi isx}dx \int_{-\infty}^{+\infty} h(t)e^{2\pi ist}dt$ separate integration

$= F(g)F(h)$

2 Frequency Smoothing

2.1 Solution

The source image is lenaNoise.PNG. I use fft2 function to compute Fourier transform, and then I use fftshift function to center low frequency. I create a new blank image which has the same size with the source image. Then I filter the centered Fourier transformed image based on the size that I initialize. Finally, I use ifftshift functions and ifft2 function to reconstruct denoised image.

2.2 Experiment Result

The original image is shown in Figure 1.

Figure 1. Source image

The centered Fourier transform of image is shown in Figure 2.

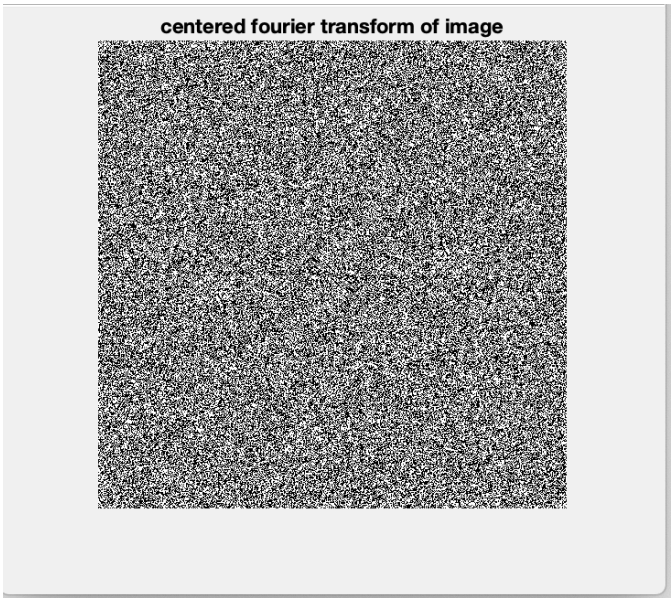

Figure 2. Centered Fourier transform of image

In order to keep different low frequencies and set all other high frequencies to 0, I change the value of parameters, size, in my program. That is to say, I can change the value of low frequencies through changing the value of size. Then based on the different values, I reconstruct the original image by using the new generated frequencies. I tried the value of size = 100, size = 200, size = 220, size = 240 and size = 256. Based on these four values, I reconstructed four original images. Figure 3 shows that situation with size = 100. Figure 4 shows the situation with size = 200. Figure 5 shows the situation with size = 220. Figure 6 shows the situation with size = 240, and Figure 7 shows the situation with size = 256.

Figure 3. Reconstructed image with size = 100

Figure 4. Reconstructed image with size = 200

Figure 5. Reconstructed image with size = 220

Figure 6. Reconstructed image with size = 240

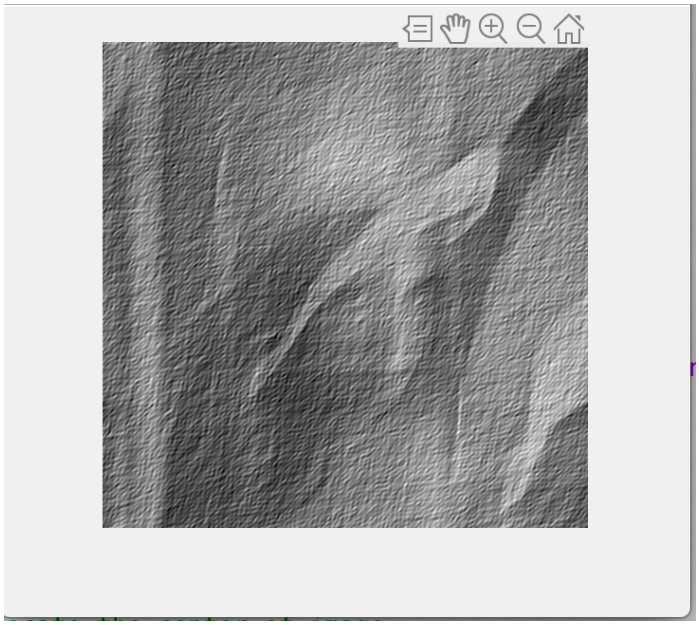

Figure 7. Reconstructed image with size = 256

3 Gradient Decent

3.1 Problem Description

The original clean image is 'cameraman.tif'. Based on this original image, Gaussian noise is added, and then I get the source image, the original image with Gaussian noise. Total variation minimization is an ideal method to achieve image denoising. The goal of this experiment is to build ROF model for the source image with total variation minimization, then implement gradient decent algorithm to remove Gaussian noise in the source image.

3.2 Solution

First, I apply central difference to compute the gradient of x and y respectively, and I continues to apply central difference to compute the gradient terms of the gradient of components in x and y direction.

The equation to compute forward difference is

$$\nabla I(i,j) = I(i,j+1) - I(i,j)$$

The method to compute backward difference is

$$\nabla I(i,j) = I(i,j) - I(i,j-1)$$

The method to compute central difference is

$$\nabla I(i,j) = [I(i,j+1) - I(i,j-1)]/2$$

Then I compute the gradient of the source images.

$$\nabla E(u) = -\lambda(f-u) - \frac{div(\nabla u)}{|\nabla u|}$$

The method to update the gradient and implement gradient decent:

$$u^{k+1} = u^k + \alpha \nabla E(u)$$

The method that to compute energy function to examine the correctness of my algorithm(convergence function):

$$E = \|f-u\|_2^2 + \|\nabla E(u)\|_2^2$$

3.3 Experimental Detail

The forward / backward difference for computing image gradient is given in Dx.m / Dxt.m, and I tried these two methods to compute gradients, forward difference and backward difference, but the performance is not very satisfying, so I applied central difference instead to get a better performance.

The type of the source image with noise is unit8, but the type of denoised image is double, so I need to use method, double, to transfer the type of source image from unit8 to double to ensure the correctness of calculation.

I add a very tiny number, epsilon when I calculate the gradient. The reason that I add this number to compute the gradient is to avoid computation error when the division is zero. I assume epsilon is equal to 0.0001, so it will not change result too much, and it can ensure that my program will implement successfully.

3.4 Experiment Result

I add Gaussian noise in a clean image and then use imshow function to show this image. The source image with noise is shown in Figure 8.

Figure 8. Source Image with Noise

The graph of the energy function is shown in Figure 9. From this graph, it is clear that my energy function is deceasing through the process of gradient descent, so I can assume that my energy function (convergence function) will converge with the increase of the number of step.

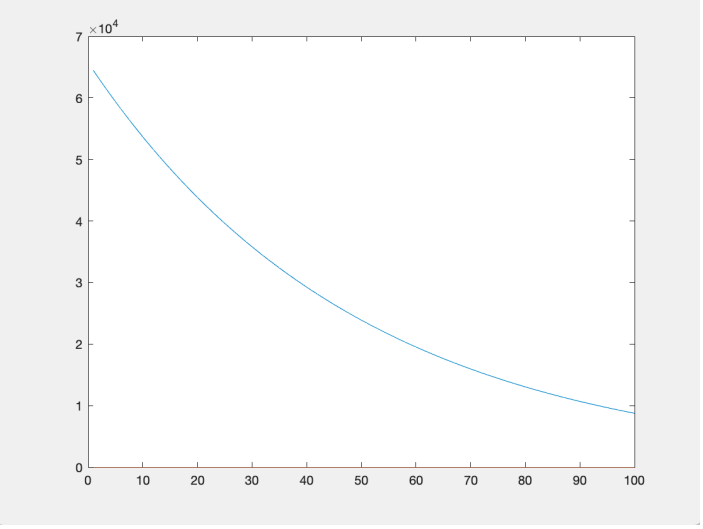

Figure 9. Energy Function/Convergence Function

The denoised image is shown in Figure 10.

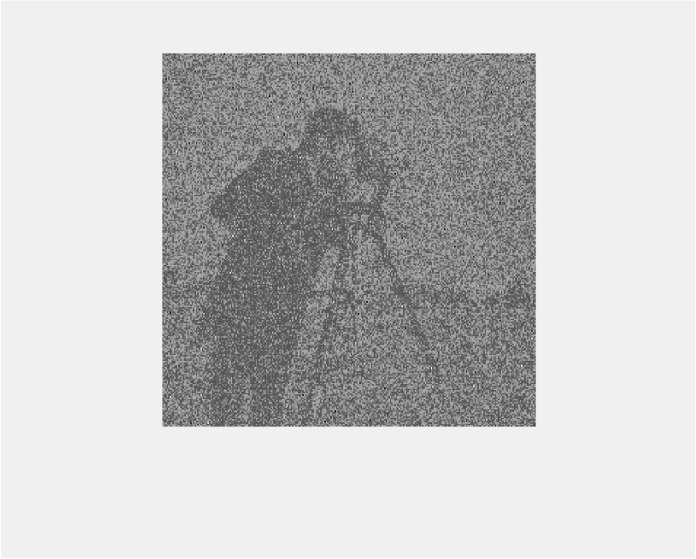

Figure 10. Denoised Image