Digital Image Processing Homework 2

Jiajie Li 1750655

December 1st, 2020

Abstract

The sharpening tool allows you to quickly focus on blurred edges, improve the sharpness or focus of an area of an image, and bring out the colors in specific areas of an image. Image denoising is the process of reducing noise in digital images. In reality, digital images are often affected by noise interference between the imaging device and the external environment during the digitization and transmission process, which is known as noise-bearing image or noisy image. We will apply what we have learned in this course and use matlab to process the images.

1 Introduction and Overview

Spatial filtering is an image enhancement method using filtering. Its theoretical basis is spatial convolution and spatial correlation. The purpose is to improve image quality, including removal of high-frequency noise and interference, and image edge enhancement, linear enhancement, and deblurring. There are low-pass filtering (smoothing), high-pass filtering (sharpening), and band-pass filtering. The two processing methods are computer processing (digital filtering) and optical information processing.

2 Problem 1: Gaussian Noise

2.1 Problem 1-1

Read and display the image. Filter the image using MATLAB function 'filter2' or 'conv2' with masks of all ones of different sizes. Display and submit. What's the difference between 'filter2' and 'conv2'? How dose the size of mask affect the blurring and noise reduction? Which mask do you think provides the best tradeoff between blurring and noise reduction in this image?

2.1.1 Theoretical Background - Arithmetic Mean Filter

This is the simplest mean value filter that removes uniform and Gaussian noise, but causes some blurring of the image. Let S_{xy} represent the filter window with the center at (x, y) and size $m \times n$. The arithmetic mean filter is simply calculating the mean value of the pixels in the window area and assigning the mean value to the pixels at the center of the window.

$$f(x,y) = \frac{1}{mn} \sum_{(x,y)\in S_{xy}} g(s,t) \tag{1}$$

where $g(s,t)$ denotes the original image and $f(x,y)$ denotes the mean filtered image. Based on the above formula, the window template of the arithmetic mean filter can be easily obtained, and the following is an example of 3×3.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

2.1.2 Computational Results

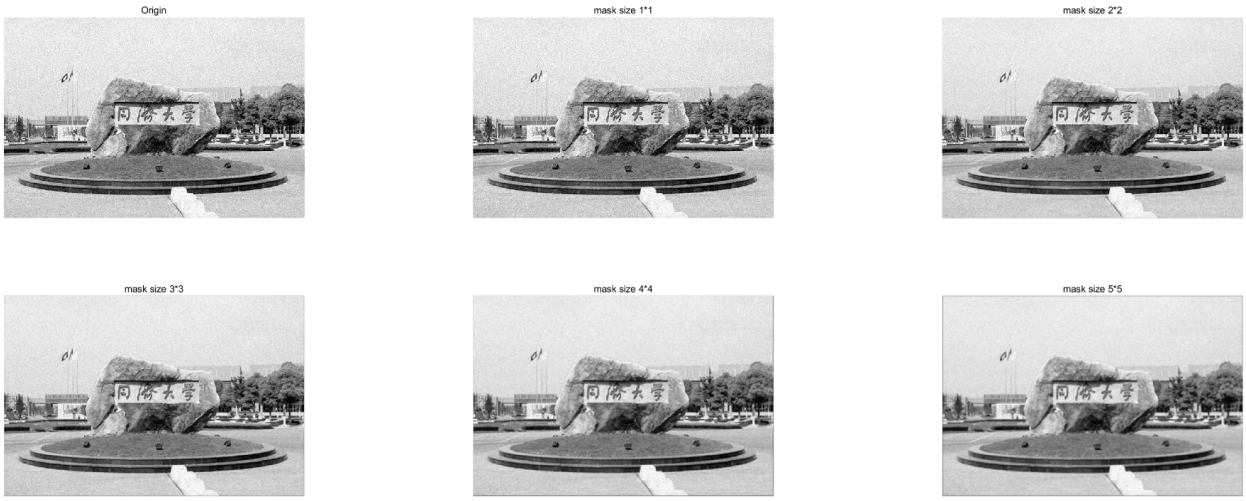

Figure 1: Computational Results of Problem 1-1

2.1.3 Comments

1. If the kernel is not centrally symmetrical, then convolution and filtering will give completely different results. Also, if padding is not applied, convolution will change the image size while filtering will not.
2. The larger the mask, the blurrier the image.
3. A mask of size 4 achieves the best trade-off.

2.2 Problem 1-2

Apply gaussian filter to the image 'p1' using gaussConv(image,sigma) and make a comparison with the MATLAB function 'imfilter'. Display and submit.

2.2.1 Theoretical Background - Gaussian Filter Function

For images, a Gaussian filter is a 2-dimensional convolution operator using a Gaussian kernel for image blurring (removing detail and noise). Theoretically, the Gaussian distribution has non-negative values over all defined domains, which requires an infinitely large convolutional kernel. In practice, you only need to take the values within 3 times the standard deviation around the mean, and simply remove the outer part. The figure below shows an integer value Gaussian kernel with a standard deviation of 1.0.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{3}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4}$$

The Gaussian function has five important properties that make it particularly useful in early image processing. These properties indicate that Gaussian smoothing filters are very effective low-pass filters in both the spatial and frequency domains, and have been used effectively by engineers in practical image processing. Gaussian functions have five very important properties, which are

1. The two-dimensional Gaussian function has rotational symmetry.

2. Gaussian functions are single-valued functions.
3. The Fourier transform spectrum of the Gaussian function is single-valved.
4. The Gaussian filter width (which determines the degree of smoothing) is characterized by the parameter σ, and the relationship between σ and the degree of smoothing is very simple.
5. Due to the separability of the Gaussian function, a larger Gaussian filter can be effectively implemented.

2.2.2 Computational Results

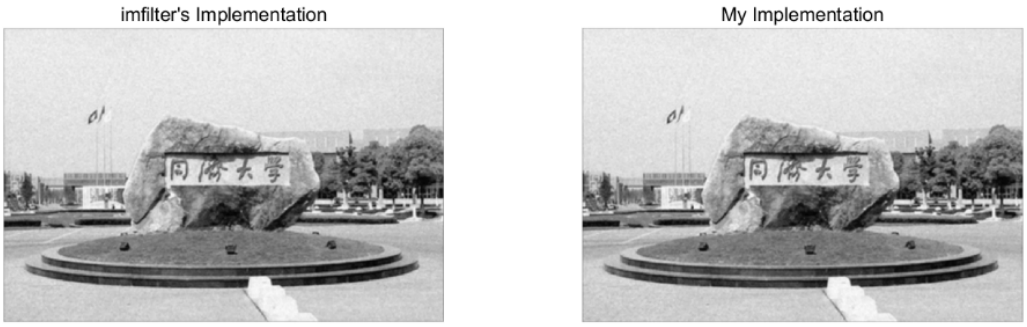

Figure 2: Computational Results of Problem 1-2

3 Problem 2: Pepper and salt noise

3.1 Problem 2-1

Read and display 'p2d1'. Write a function minFilter(img, n) which apply min filter to 'img' with window of size $n \times n$. Apply min filter to the 'p2d1' with n in $[1, 3, 5, 7]$. How does the window size affect the result?

3.1.1 Computational Results

Figure 3: Computational Results of Problem 2-1

3.1.2 Theoretical Background - Minimum Filter

To find the darkest points in an image. Finds the minimum value in the area encompassed by the filter. Reduces the salt noise as a result of the min operation. The 0^{th} percentile filter is min filter.

$$f(x, y) = \min_{(s,t)\in} \{g(s, t)\} \tag{5}$$

3.1.3 Comments

The larger the mask size, the darker the filtered image and the less salt noise.

3.2 Problem 2-2

Read and display 'p2d2'. Write a function maxFilter(img, n) which apply max filter to 'img' with window of size $n \times n$. Apply min filter to the 'p2d2' with n in $[1, 3, 5, 7]$. How does the window size affect the result?

3.2.1 Computational Results

Figure 4: Computational Results of Problem 2-2

3.2.2 Theoretical Background - Maximum Filter

To find the brightest points in an image. Finds the maximum value in the area encompassed by the filter. Reduces the pepper noise as a result of the max operation. The 100^{th} percentile filter is min filter.

$$f(x, y) = \max_{(s,t)\in} \{g(s, t)\} \tag{6}$$

3.2.3 Comments

The larger the mask size, the brighter the filtered image and the less pepper noise.

3.3 Problem 2-3

Read and display the image in 'p2d3'. Apply median filter to 'p2d3' with the MATLAB function 'ordfilt2' with window size in [1, 3, 5, 7]. How does the window size affect the result?

3.3.1 Computational Results

Figure 5: Computational Results of Problem 2-3

3.3.2 Theoretical Background - Median Filter

Median filtering of images is a nonlinear image processing method and is a typical application of statistical sorting filters. Different from the previously introduced idea of mean value processing, median filtering determines the gray level of the center pixel by sorting the pixels in the neighborhood by gray level. The procedure is as follows: a moving window with an odd number of points is used to replace the value of the center point of the window with the median value of the points within the window. If there are 5 points in the window with values of 1, 2, 3, 4 and 5, then the median value of each point in the window is 3, that is, 3 is used instead of the center pixel value. Median filtering is the most effective way to filter out impulse interference and image scan noise, and it can also overcome the blurring of image details caused by linear filters such as field simple smoothing filters.

$$\hat{f}(x,y) = \text{median}_{(s,t)\in S_{xy}}\{g(s,t)\} \tag{7}$$

3.3.3 Comments

The larger the mask size, the less pepper and salt noise there is.

3.4 Problem 2-4

Can we apply the max filter to 'p2d1' or apply the min filter to 'p2d2'? Can you describe the scope of application of these three different filters?

3.4.1 Computational Results

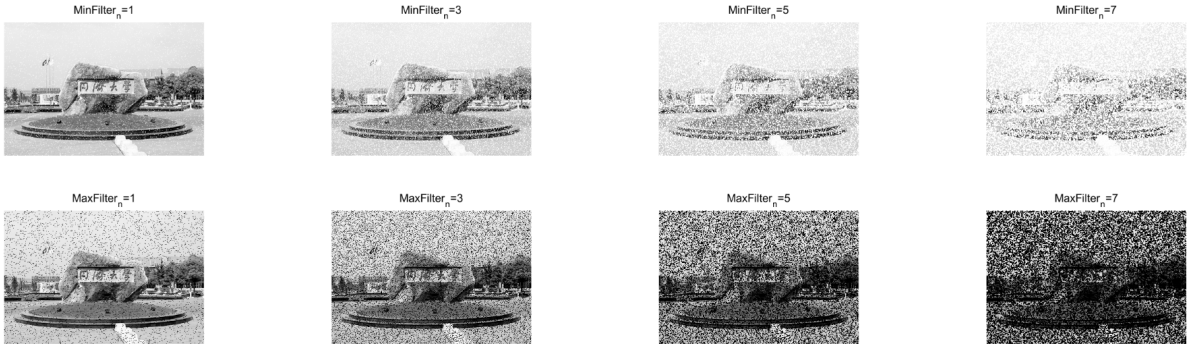

Figure 6: Computational Results of Problem 2-4

3.4.2 Comments

min Filter is good for salt noise, max Filter is good for pepper noise, and median filter is good for pepper and salt noise.

4 Problem 3: Sharpen

4.1 Problem 3-1

Sharpen the best result in Problem 1 using the Laplacian mask:

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} \tag{8}$$

Display and submit. Does this operation affect the blurring and the noise reduction?

4.1.1 Theoretical Background

Sharpening As the opposite of low-pass filtering for image smoothing and noise reduction, high-pass filtering can sharpen the image, thereby enhancing and emphasizing the detailed information (high spatial frequency components) in the image.

High-pass filtering can be carried out by subtracting the low-pass filtered image from its original version, which can be considered as all-pass filtered by a delta function kernel.

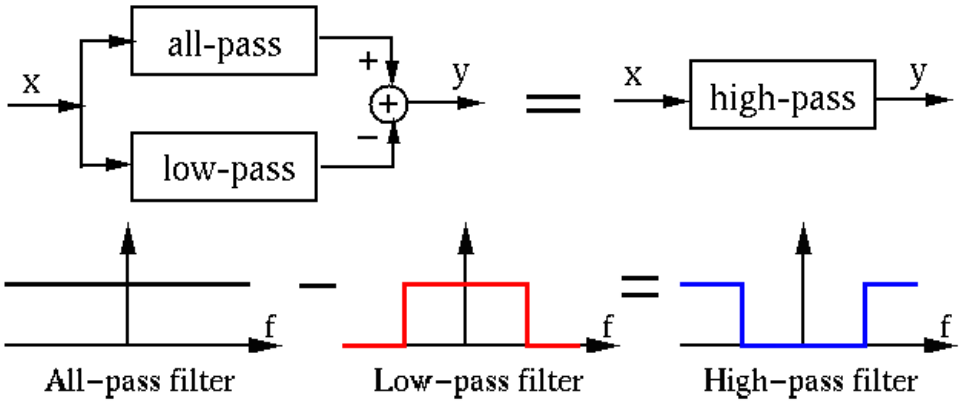

Figure 7: Diagram of the Sharpening Principle

Laplace Operator The Laplace operator is a scalar operator defined as the dot product (inner product) of two gradient vector operators:

$$\Delta = \nabla \cdot \nabla = \nabla^2 = \left[\frac{\partial}{\partial x_1}, \cdots, \frac{\partial}{\partial x_N}\right] \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_N} \end{bmatrix} = \sum_{n=1}^{N} \frac{\partial^2}{\partial x_n^2} \tag{9}$$

In $N = 2$ dimensional space, we have:

$$\Delta = \nabla \cdot \nabla = \nabla^2 = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{10}$$

When applied to a 2-D function $f(x, y)$, this operator produces a scalar function:

$$\Delta f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{11}$$

In discrete case, the second order differentiation becomes second order difference. In 1-D case, if the first order difference is defined as

$$\nabla f[n] = f[n+1] - f[n] \tag{12}$$

then the second order difference is

$$\Delta f[n] = \nabla(\nabla f[n]) = \nabla f[n] - \nabla f[n-1] = (f[n+1] - f[n]) - (f[n] - f[n-1]) = f[n+1] - 2f[n] + f[n-1] \tag{13}$$

Note that $\triangle f[n]$ is so defined that it is symmetric to the center element $f[n]$. The Laplace operation can be carried out by 1-D convolution with a kernel $[-1, 2, -1]$. In 2-D case, Laplace operator is the sum of two

second order differences in both dimensions:

$$\begin{aligned}\Delta f[m,n] &= \triangle_m[f[m,n]] + \Delta_n[f[m,n]] \\ &= f[m+1,n] - 2f[m,n] + f[m-1,n] + f[m,n+1] - 2f[m,n] + f[m,n-1] \\ &= f[m+1,n] + f[m-1,n] + f[m,n+1] + f[m,n-1] - 4f[m,n]\end{aligned} \tag{14}$$

This operation can be carried out by 2-D convolution kernel(added original image):

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} \tag{15}$$

4.1.2 Computational Results

Figure 8: Computational Results of Problem 3-1

4.1.3 Comments

yes, laplace operator may detect edges as well as noise (isolated, out-of-range).

4.2 Problem 3-2

Filter the image 'p3' using Laplacian mask:

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix} \tag{16}$$

And apply Laplacian of Gaussian(LoG) operator to 'p3' using matlab function 'fspecial' and 'imfilter' with window size of 3. Adjust the parameter sigma to show the correlation between the two operators.

4.2.1 Theoretical Background

Laplacian of Gaussian (LoG) As Laplace operator may detect edges as well as noise (isolated, out-of-range), it may be desirable to smooth the image first by a convolution with a Gaussian kernel of width σ

$$G_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{17}$$

to suppress the noise before using Laplace for edge detection:

$$\triangle\left[G_\sigma(x,y) * f(x,y)\right] = \left[\triangle G_\sigma(x,y)\right] * f(x,y) = LoG * f(x,y) \tag{18}$$

The first equal sign is due to the fact that

$$\frac{d}{dt}[h(t) * f(t)] = \frac{d}{dt}\int f(\tau)h(t-\tau)d\tau = \int f(\tau)\frac{d}{dt}h(t-\tau)d\tau = f(t) * \frac{d}{dt}h(t) \tag{19}$$

So we can obtain the Laplacian of Gaussian $\triangle G_\sigma(x, y)$ first and then convolve it with the input image. To do so, first consider

$$\frac{\partial}{\partial x}G_\sigma(x, y) = \frac{\partial}{\partial x}e^{-(x^2+y^2)/2\sigma^2} = -\frac{x}{\sigma^2}e^{-(x^2+y^2)/2\sigma^2} \tag{20}$$

and

$$\frac{\partial^2}{\partial^2 x}G_\sigma(x, y) = \frac{x^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2} - \frac{1}{\sigma^2}e^{-(x^2+y^2)/2\sigma^2} = \frac{x^2-\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2} \tag{21}$$

Note that for simplicity we omitted the normalizing coefficient $1/\sqrt{2\pi\sigma^2}$. Similarly we can get

$$\frac{\partial^2}{\partial^2 y}G_\sigma(x, y) = \frac{y^2-\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2} \tag{22}$$

Now we have LoG as an operator or convolution kernel defined as

$$L_{OG} \triangleq \Delta G_\sigma(x, y) = \frac{\partial^2}{\partial x^2}G_\sigma(x, y) + \frac{\partial^2}{\partial y^2}G_\sigma(x, y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2} \tag{23}$$

The edges in the image can be obtained by these steps:

1. Applying LoG to the image
2. Detection of zero-crossings in the image
3. Threshold the zero-crossings to keep only those strong ones (large difference between the positive maximum and the negative minimum)

4.2.2 Computational Results

Figure 9: Computational Results of Problem 3-2

4.2.3 Comments

The Gaussian filter width (which determines the degree of smoothing) is characterized by the parameter σ, and the relationship between σ and the degree of smoothing is very simple: the larger the σ, the wider the band of the Gaussian filter and the better the degree of smoothing, and the less the Laplace will be affected by noise, but at the same time the sharpening effect will be worse.