# Report for EQ2330 Image and Video Processing

EQ2330 Image and Video Processing, Project 1

Zhenghao Li zhenghl@kth.se
Letao Feng letao@kth.se
Heng Zhao hengzhao@kth.se

November 19, 2023

## Summary

In this report, we investigate techniques for image enhancement in both spatial and frequency domains. The initial task focuses on histogram equalization, simulating a low-contrast image by reducing the range of original image. A histogram equalization algorithm was then performed and the resulting histograms before and after equalization are compared. The second task centers on image denoising, we use mynoisegen function to introduce Gaussian and salt & pepper noises and apply the mean filter and median filter to the image respectively. Mean filtering is suitable for processing continuous noise such as Gaussian noise, and median filtering is suitable for processing salt and pepper noise. In the third task, a blurred image is generated using a degradation model, and Fourier spectra before and after degradation are plotted and compared. we design a wiener filter to restore the out-of-focus image.

## 1 Introduction

The spatial and frequency domains are different perspectives from which we solve problems in digital image processing. In spatial domain, image processing focuses on pixel-level operations. We apply a histogram equalization algorithm to enhance image which can be denoted by the expression $g(x, y) = T[f(x, y)]$ where $T$ is the equalization operator on the input image $f(x, y)$. Then we implement a $3 \times 3$ median filter and $3 \times 3$ mean filter to evaluate its denoising effect on different noise types. In the frequency domain, processing is mainly through techniques such as Fourier transform to operate on the spectrum.

## 2 System Description

### 2.1 Histogram equalization

In this section, we need to draw the histogram of the original image. Then we simulate a low-contrast image by reducing the dynamic range of the image. The formula is shown below:

$$g(x, y) = \min(\max(\lfloor 0.2f(x, y) + 50 \rceil, 0), 255). \tag{1}$$

where $\lfloor \cdot \rceil$ denotes the round operator.

Histogram equalization improves the visual quality of an image by redistributing the gray levels of the image so that they are more evenly distributed throughout the dynamic range. For discrete case, The histogram equalization transformation can be expressed as:

$$s_k = (L-1)\sum_{i=0}^{k} p_i = (L-1)\sum_{i=0}^{k} \frac{n_i}{mn}. \tag{2}$$

Where L is the number of gray levels in the picture, $n_i$ is the number of pixels with gray level i and mn is total pixels of image. After obtaining the low-contrast image, we implement the global histogram equalization. Firstly, use a loop to iterate through each pixel grey-scale value (0 to 255) and compute the probability of occurrence for each grey-scale value in the image. Then using the cumsum function to calculate the Cumulative Mass Function (CMF). The values of CMF are then mapped to a range of 0 to 255. Finally, using the arrayfun function, each pixel value of the original image is mapped to a new grey level for histogram equalization.

### 2.2 Image Denoising

In this section, we first need to generate two noisy images. One contains the Gaussian noise, and the other one contains the salt & pepper noise.

For the image containing Gaussian noise, we name it $g_1(x, y)$. The Gaussian noise is additive, so we have

$$g_1(x, y) = f(x, y) + \eta(x, y). \tag{3}$$

where $\eta(x, y)$ is the Gaussian noise with zero mean and variance 64.

For the image containing salt & pepper noise, we name it $g_2(x, y)$. It is generated by converting certain pixels to either 0 or 255. Thus, there would be some black and white dots in the output image. The probability that a pixel value in the original image changes to either of these values is 0.05.

To investigate the denoising effect of the spatial smoothing filter, we implement the 3 × 3 mean filter and apply it to the noisy images. The unweighted 3 × 3 mean filter mask is like

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

We then use the conv2 function in Matlab to do the convolution calculation.

To investigate the denoising effect of the order-statistics filter, we implement the 3 × 3 median filter and again apply it to the noisy images. The basic principle of median filtering is to replace the value of a pixel in a digital image with the median of the values of the pixels in a neighbourhood of that pixel, thus eliminating isolated noise points. We use a median filter function that we wrote ourselves. Median filtering can also be implemented in Matlab by using the medfilt2() function directly.

### 2.3 Frequency Domain Filtering

In this section, a frequency domain restoration is operated to a degraded image. The system schematic is shown in figure 1, where $f(x, y)$ is the original image, $g(x, y)$ is the degraded image and $\hat{f}(x, y)$ is the restored image using the wiener filter.

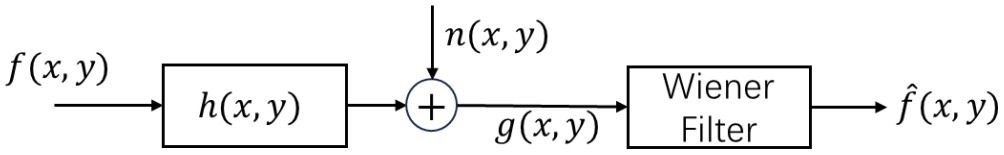


Figure 1: image degrading and restoration

The additive noise, in this case, is assumed to be the quantization error of the degraded image which is represented using 8 bits.

The restoration function, taking the degraded image, blurring function and the noise variance as the input, is implemented with the Wiener filter algorithm. Taking the minimum mean square error into evaluation, referring to [1], the filtering function is given in frequency domain by the expression equation 5.

$$\hat{F}(u, v) = \left(\frac{H^*(u, v)}{|H(u, v)|^2 + K}\right) G(u, v) \tag{5}$$

The constant K is computed by the expression $Var(n)/Var(g)$ since in the equation it is to substitute $S_\mu(u, v)/S_f(u, v)$, which is impossible to acquire given merely the variance of noise.

Moreover, some measure needs to be taken in order to mitigate the the issues caused by sharp edges of the image, which will be talked in detail in the next section.

## 3 Results

### 3.1 Histogram equalization

For the histogram equalization part, the original image, low-contrast image, equalized image and their histograms are shown in figure 2.

As you can see in the figure, the low-contrast image histogram is more centralized. Thus the image looks

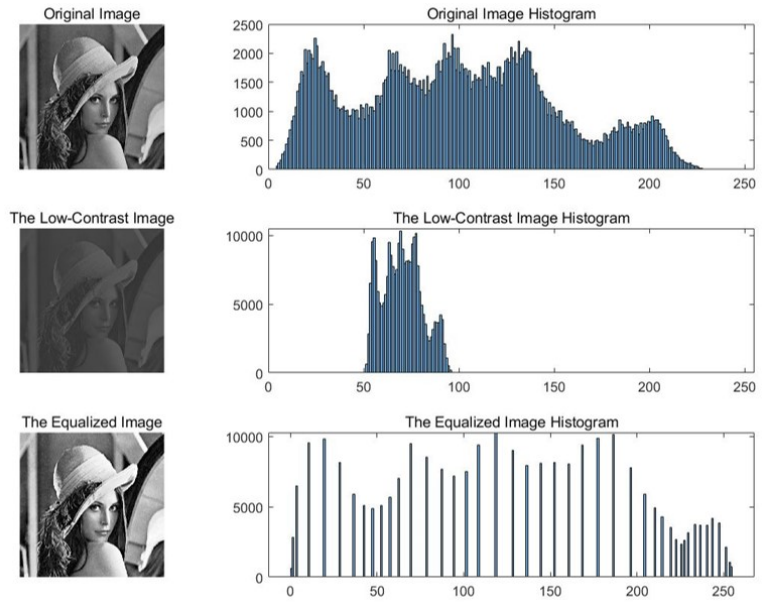


Figure 2: The original image, low-contrast image, equalized image and their histograms

greyer and has less contrast. However, it has less gray levels than original one.

The histogram is not flat after the equalization. Histogram equalization is derived from continuous situations. In discrete situations, histogram equalization only makes the probability of each value appearing as uniform as possible by merging certain values with low probability. This simple merging cannot make all probabilities Values equal.

## 3.2 Image Denoising

For the image denoising part, the original image, two noisy images and their histograms are shown in figure 3.

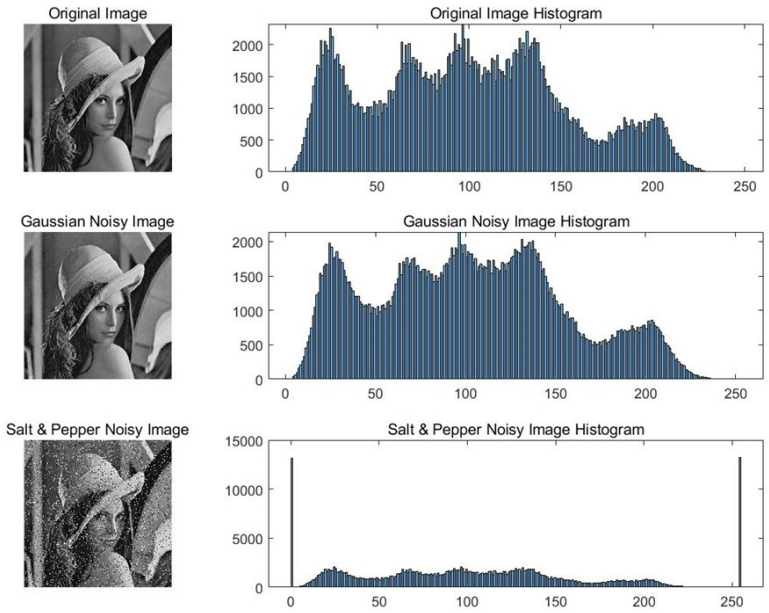


Figure 3: The original image, two noisy images and their histograms

After applying the $3 \times 3$ mean filter to the noisy images, we obtain figure 4.

After applying the $3 \times 3$ median filter to the noisy images, we obtain figure 5.

From the results, we find that the mean filter is effective in reducing Gaussian noise. The mean filter works well when the noise is relatively uniform across the image. It doesn't work well in reducing salt & pepper noise.

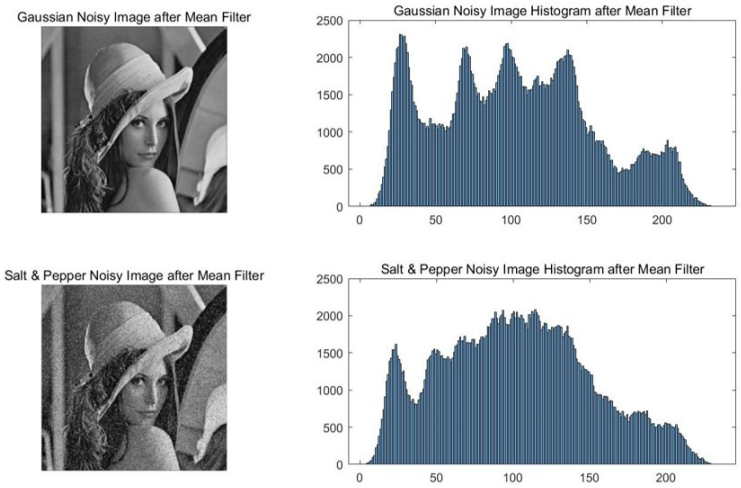


Figure 4: Mean filter through two noisy images and their histograms

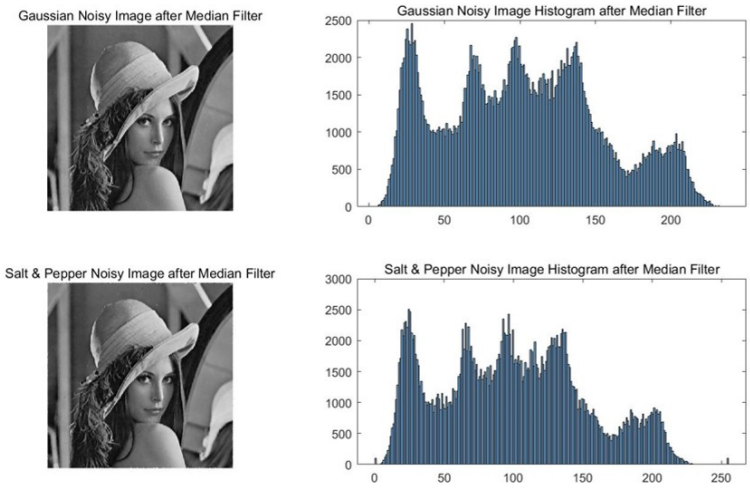


Figure 5: Median filter through two noisy images and their histograms

However, the median filter is particularly effective in reducing salt & pepper noise, which is characterized by random, isolated occurrences of very bright or very dark pixels. The median filter is able to preserve edges and fine details in the image while effectively removing this type of noise.

In summary, the mean filter is more effective at reducing Gaussian noise, while the median filter is better suited for reducing salt & pepper noise.

### 3.3 Frequency Domain Filtering

It is clear in figure 6 that the blurring function is a low-pass filter and the spectrum of image after implementing the Gaussian blur kernel lost some high-frequency components. It can also be seen that in the out-of-focus image, some details are lost due to the loss of information residing in high-frequency. In figure 8, two restored images are shown, which is clear that the restored image is sharper than the out-of-focus one.

In the last section, it is mentioned that without taking certain measure, there can be some artefacts destroying the restored image. This is because that the Discrete Fourier Transform (DFT) operates under the assumption of periodicity. When you apply the DFT to an image, it essentially assumes that the image repeats itself indefinitely in both the horizontal and vertical directions. This assumption can lead to problems when the input image has sharp edges or other features near its boundaries, which is called "frequency leakage". If the image has sharp edges, these edges may not align perfectly when the image is assumed to be periodic. The DFT treats the image as if it were composed of repeating patterns, and if the edges do not match up, it can introduce artifacts and distortions in the frequency domain.

For this reason, we padded horizontally and vertically some pixels to make the image after padding is symmetric at the boundaries and the length and width are larger than the summation of sizes of blurred image

and blurring function. The effect of this can be easily seen by comparison in figure 8.

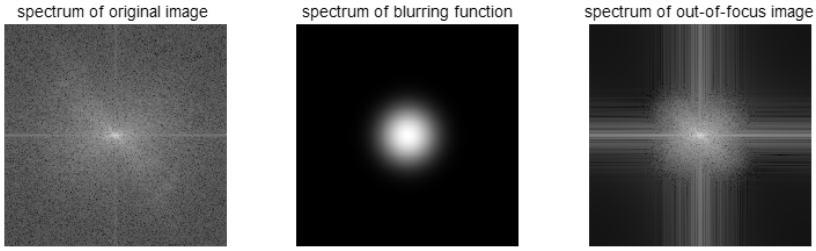


Figure 6: The spectra of the original image, blurred image and the blurring function

(a) original image (b) out-of-focus image

Figure 7: before restoration

(a) with padding (b) w/o padding

Figure 8: restored image

# 4 Conclusions

In this project we explore techniques for enhancing digital images in spatial and frequency domains. Firstly, we enhance the contrast of image by making the gray levels of the image span a wider range of gray levels. But after equalization we cannot get a completely flat histogram because the distribution function is discrete. We then use two filters to denoise the image. It was found that the mean filter can effectively reduce high-frequency noise but cannot retain the details of the image, and the median filter can effectively remove isolated noise points. Lastly, we design a wiener filter to restore the image and the recovered image exhibits better visual quality. In order to avoid the sharp edges in the image, we introduce padding to prevent artifacts during the Discrete Fourier Transform.

# Appendix

## Who Did What

| | |
|---|---|
| Histogram equalization | Zhenghao Li, Letao Feng |
| Image denoising | Letao Feng, Heng Zhao |
| Frequency domain filtering | Heng Zhao, Zhenghao Li |

## MatLab code

Include the well documented MatLab code that you have used.

```
function [ img ] = median_filter( image, m )
%-------------------------------------------
%median
%input:
%image:original
%m:size of kernel

%output:
%img: the output image
%-------------------------------------------
    n = m;
    [ height, width ] = size(image);
    x1 = double(image);
    x2 = x1;
    for i = 1: height-n+1
        for j = 1:width-n+1
            mb = x1( i:(i+n-1),  j:(j+n-1) );%acquire n*n matrix
            mb = mb(:);
            mm = median(mb);%taking the median value
            x2( i+(n-1)/2,  j+(n-1)/2 ) = mm;

        end
    end

    img = uint8(x2);
end

function f_hat = deblur(g, h, n_var)
    [g_M, g_N] = size(g);
    [h_M, h_N] = size(h);

    % padd pixels arround the image so the boundaries
    g_padded = padarray(g, size(h), 'symmetric');

    %compute the constant K
    K=n_var/var(g(:));
    [M, N] = size(g_padded);
    %M = g_M+h_M;
    %N = g_N+h_N;
    %g_padded=g;

    % to the frequency domain
    H = fft2(h, M, N);
    G = fft2(g_padded,M,N);
    % equation of the wiener filter
    filter = (1 ./ H) .* ((abs(H).^2) ./ (abs(H).^2 + K));
    % apply wiener filter to the blurred image
    F_hat = G .* filter;
    f_hat = abs(ifft2(F_hat));
    %remove the padding
    f_hat = f_hat(ceil(double(h_M) / 2):ceil(double(h_M) / 2) + g_M - 1, ceil(double(h_N) / 2):ceil(dou
end

main
%% part 1 Histogram equalization
clear
```

```
clc
close all

f=imread("lena512.bmp");
subplot(3,2,1);
imshow("lena512.bmp");
title('Original Image');
subplot(3,2,2)
histogram(f,'BinWidth',1);
title('Original Image Histogram');
axis([0 255 0 2500])

%low-contrast image
a=0.2;
b=50;
for x=1:512
    for y=1:512
g(x,y)=min(max(round(a*f(x,y))+b,0),255);
    end
end

subplot(3,2,3)
imshow(g);
title('The Low-Contrast Image');
subplot(3,2,4)
histogram(g,'BinWidth',1)
title('The Low-Contrast Image Histogram');
axis([0 255 0 10500])

prpixel=zeros(1,256);
for i=0:255
   prpixel(i+1)=length(find(g==i))/(512*512);  %find(g==i) find the gray value is i in the image
end

cdf=cumsum(prpixel);
cdf_pixel=round(255*cdf);
equalizedImage=uint8(arrayfun(@(x) cdf_pixel(x),g));

subplot(3,2,5)
imshow(equalizedImage);
title('The Equalized Image');
subplot(3,2,6)
histogram(equalizedImage,'BinWidth',1);
title('The Equalized Image Histogram');

% hold on
% figure
% j=histeq(g,255);
% subplot(221)
% imshow(j)
% subplot(222)
% histogram(j,'BinWidth',1)
%
% subplot(223)
% imshow(equalizedImage);
% subplot(224)
% histogram(equalizedImage,'BinWidth',1)
```

```
%% part 2 Image Denoising

clear; clc; close all;

im = imread('lena512.bmp');

% Generate Gaussian noise
gaussian_noise = mynoisegen('gaussian', 512, 512, 0, 64);
im_gaussian = im + uint8(gaussian_noise);

% Generate salt & pepper noise
salt_pepper_noise = mynoisegen('saltpepper', 512, 512, 0.05, 0.05);
im_saltp = im;
im_saltp(salt_pepper_noise == 0) = 0;
im_saltp(salt_pepper_noise == 1) = 255;

% Plot the histograms
figure;
subplot(3, 2, 1);
imshow(im);
title('Original Image');
subplot(3, 2, 3);
imshow(im_gaussian);
title('Gaussian Noisy Image');
subplot(3, 2, 5);
imshow(im_saltp);
title('Salt & Pepper Noisy Image');

subplot(3, 2, 2);
% imhist(im);
histogram(im,'BinWidth',1);
% xlabel(r);
title('Original Image Histogram');
subplot(3, 2, 4);
% imhist(im_gaussian);
histogram(im_gaussian,'BinWidth',1);
title('Gaussian Noisy Image Histogram');
subplot(3, 2, 6);
% imhist(im_saltp);
histogram(im_saltp,'BinWidth',1);
title('Salt & Pepper Noisy Image Histogram');

% Apply 3x3 mean filter to Gaussian noisy image
h_mean = ones(3, 3) / 9;
im_gaussian_mean = conv2(double(im_gaussian), h_mean, 'same');
% imshow(uint8(im_gaussian_mean))

% Apply 3x3 mean filter to salt & pepper noisy image
im_saltp_mean = conv2(double(im_saltp), h_mean, 'same');

% Plot histograms
figure;
subplot(221);
imshow(uint8(im_gaussian_mean));
title('Gaussian Noisy Image after Mean Filter');
subplot(222);
histogram(uint8(im_gaussian_mean),'BinWidth',1);
```

```
% imhist(uint8(im_gaussian_mean));
title('Gaussian Noisy Image Histogram after Mean Filter');
subplot(223);
imshow(uint8(im_saltp_mean));
title('Salt & Pepper Noisy Image after Mean Filter');
subplot(224);
histogram(uint8(im_saltp_mean),'BinWidth',1);
% imhist(uint8(im_saltp_mean));
title('Salt & Pepper Noisy Image Histogram after Mean Filter');

im_gaussian_median = median_filter(im_gaussian,3);
im_saltp_median = median_filter(im_saltp,3);

figure;
subplot(221);
imshow(uint8(im_gaussian_median));
title('Gaussian Noisy Image after Median Filter');
subplot(222);
histogram(uint8(im_gaussian_median),'BinWidth',1);
% imhist(uint8(im_gaussian_median));
title('Gaussian Noisy Image Histogram after Median Filter');
subplot(223);
imshow(uint8(im_saltp_median));
title('Salt & Pepper Noisy Image after Median Filter');
subplot(224);
histogram(uint8(im_saltp_median),'BinWidth',1);
% imhist(uint8(im_saltp_median));
title('Salt & Pepper Noisy Image Histogram after Median Filter');



%% part 3 Frequency Domain Filtering
clear
clc
close all

% read the image and implement the blurring
lena=double(imread('images\lena512.bmp'));
h=myblurgen('gaussian',8);
blur=min(max(conv2(lena,h,"same"),0),255);
figure;
subplot(121)
imshow(uint8(lena))
title('original')
subplot(122)
imshow(uint8(blur));
title('blurred')

%sketching the spectra
LENA=log(abs(fft2(lena))+1);
BLUR=log(abs(fft2(blur))+1);
H=log(abs(fft2(h,512,512))+1);
figure;
subplot(131)
imshow(fftshift(LENA),[]);
title('spectrum of original image')
subplot(132)
imshow(fftshift(H),[]);
title('spectrum of blurring function')
```

```
subplot(133)
imshow(fftshift(BLUR),[]);
title('spectrum of out-of-focus image')

% compute the noise variance
mu=blur-double(uint8(blur));
figure;
imshow(mu,[]);
MU=mu(:);
n=var(MU);

% deblur the image and show
f_hat=deblur(double(uint8(blur)),h,n);
imshow(uint8(f_hat))
```

# References

[1] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Prentice Hall, 2nd ed., 2002