EQ2330 Image and Video Processing

EQ2330 Image and Video Processing, Project 1

Yuqi Zheng
yuqizh@kth.se

Ernan Wang
ernan@kth.se

Shuyi Chen
shuyic@kth.se

November 22, 2021

Summary

In this project, we look into the image enhancement in both spatial and frequency domain. Based on `lena512.bmp`, we adopt histogram equalization to enhance the low-contrast image, use mean and median filter to remove Gaussian and salt & pepper noises, and design a wiener filter to restore the out-of-focus image.

1 Introduction

In this project, we set the resolution of image to be 8 bits, which means each pixel has an intensity value of [0,255]. The original image is denoted by $f(x,y)$.We simulate a low-contrast image by the model as follows,

$$g(x,y) = min(max(\lfloor h(x,y) * f(x,y) \rceil , 0), 255), \tag{1}$$

where $\lfloor \cdot \rceil$ denotes the round operator and we use a=0.2, b=50. Then, we compare the histograms of the two images. The low-contrast image can be enhanced through histogram equalization and we can observe the difference between the histograms.

Next, we use mean filter and median filter to deal with the introduced Gaussian and salt & pepper noises. Hence, we can observe the effect of the two filters on these two kinds of noises.

Finally, we use the degradation model to blur the image as follows,

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y). \tag{2}$$

To restore the blurred image, we design a wiener filter. We also consider the effect of the sharp edges of the image.

2 System Description

2.1 Spatial domain processing

2.1.1 Histogram equalization

In a digital image, the probability of gray level r_k is

$$p_r(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, \ldots, L-1, \tag{3}$$

where MN is the total number of pixels in the image, and n_k is the number of pixels with gray level r_k.

We use the function hist to get the unnormalized histogram of the input image. Then, we could get the desired histogram through normalization.

We simulate the low-contrast image by reducing the dynamic range of the image. To perform the histogram equalization, we implement the algorithm as follows[1],

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \qquad k = 0, 1, 2, \ldots, L-1. \tag{4}$$

In this way, the equalized histogram is obtained by mapping each pixel of gray level r_k of the input image to the corresponding pixel of gray level s_k. In this case, we use the function `cumsum` to calculate the CMF. We can see that histograms are approximations of pdf, and new gray levels are not allowed in processing. Therefore, the histogram is not perfectly flat after the equalization in general. In addition, we use the function `histeq` to get the equalized image.

2.1.2 Image denoising

We use the function `mynoisegen` to introduce the Gaussian and salt & pepper noises. Then, we implement the 3×3 mean and median filter to denoise the images, respectively. The mean filter uses a linear method to average the pixel values in the window, i.e.,

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5}$$

In other words, we replace the pixels in the image with the average gray values of 3×3 neighborhoods centered on these pixels. In fact, the filtering process is the two-dimensional convolution of the mask with the original image. We use the function `conv2` to achieve this.

The median filter adopts the nonlinear method. It sorts the pixel values under the 3×3 mask in ascending order and selects the intermediate value as the result. We use the function `medfilt2` to perform the process.

2.2 Frequency domain filtering

According to the definition in [1], the two-dimensional Fourier transform of image $f(x, y)$ is

$$F(u, v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^{-j2\pi(ux/M+vy/N)}. \tag{6}$$

To get the power spectrum, we use the function `fft2` which adopts the fast Fourier transform algorithm to get the two-dimensional Fourier transform of the image. Then, we use the function `fftshift` to center the zero-frequency component. In addition, we regard the log of the amplitude of the two-dimensional Fourier transform as the amplitude of the Fourier spectrum. In this section, to enhance the visual contrast of the spectrum, we normalize the amplitude.

We use the degradation model

$$g_1(x, y) = h(x, y) * f(x, y) \tag{7}$$

to generate the degraded image $g_1(x, y)$, where the blur function $h(x, y)$ is obtained by using the function `myblurgen`.

The blurred image can be quantized by

$$g(x, y) = min(max(\lfloor h(x, y) * f(x, y) \rceil, 0), 255). \tag{8}$$

To restore the blurred image, we plan to design a wiener filter as follows,

$$R_W = \frac{H^*(u,v)}{|H(u,v)|^2 + NSR} \tag{9}$$

where NSR denotes the noise-to-signal ratio. As Figure 1, the noise can be regarded as the difference between the quantized image and the blurred image. Then, we use the function `ifft2` to get the restored image in spatial domain after wiener filtering.

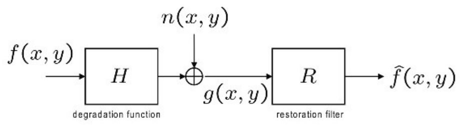

Figure 1: The degradation model.

Besides, in case of the problems caused by the sharp edges of the input image, before wiener filtering, we use the function `edgetaper` to blur the edge of it with the blur function $h(x,y)$ defined before.

3 Results

We can observe from Figure 2 that compared to the histogram of the original image, the histogram of the low contrast image is narrow and the values in it locate typically toward the middle of the intensity scale. We can also observe from Figure 2 that after equalization, the histogram of an enhanced image will cover a whole gray scale with high value of occurrences and less narrower intensity values. The reason why the histogram of the equalized image is not flat is that in the discrete case, a histogram is an approximation to a PDF, and no new allowed intensity levels are created in the process, perfectly flat histograms are rare unlike which in the continuous cases.

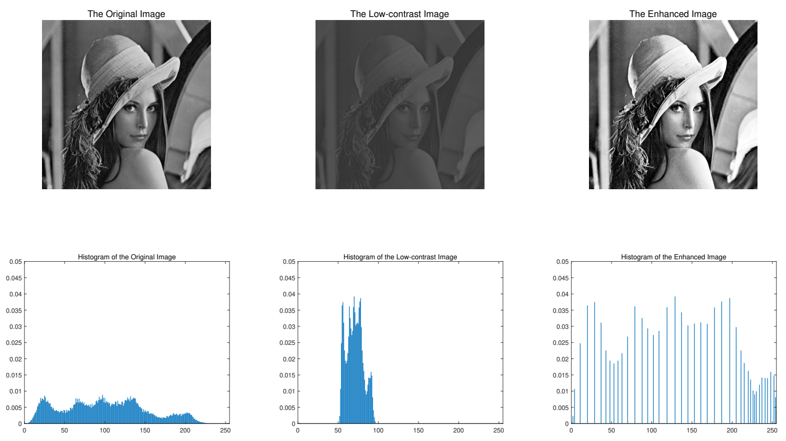

Figure 2: Equalization

By plotting the histograms of Gaussian noised and salt and pepper noised images Figure 3, we can observe that the Gaussian noise has a smoothing effect on the original image and the salt and pepper noise leads that the number of 0 and 255 intensity values increases.

We can observe from Figure 3 that the mean filter and median filter produce almost same acceptable result for the Gaussian noise. The only difference of these two filters is that the mean filter tends to smooth the image and blur the edges of the image, while the median filter preserves more edges.

As can be seen from Figure 3, the mean filter has bad performance on reducing salt and pepper noise, while the median filter produces almost perfect results for salt and pepper noise.

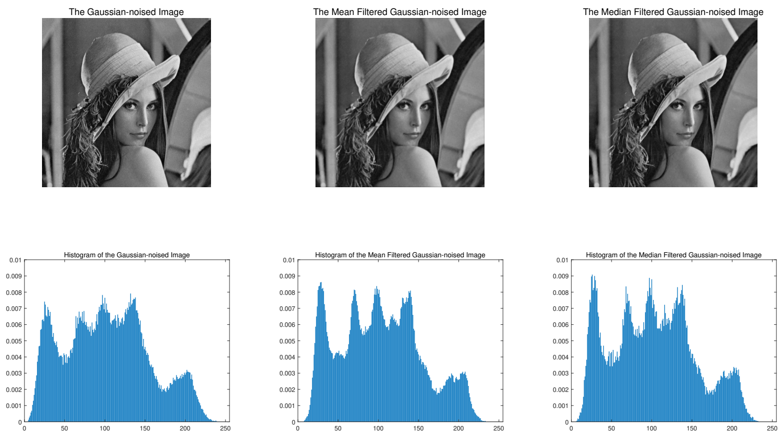

Figure 3: Gaussian

From Figure 4, we can see that the blurred image loses details in high frequency. After wiener filtering, the lost high intensity values in high frequencies reappear, as can be seen from Figure 4.

The horizontal and vertical lines in the spectrum of the original image are caused by the finite size of the images, which can create ringing artifacts at the edges when we apply algorithms to the blurred image. To reduce the ringing artifacts, we add a window function into our wiener filter to taper the edges. As can be seen from Figure 4, the wiener filter with tapering performs well in deblurring the image.

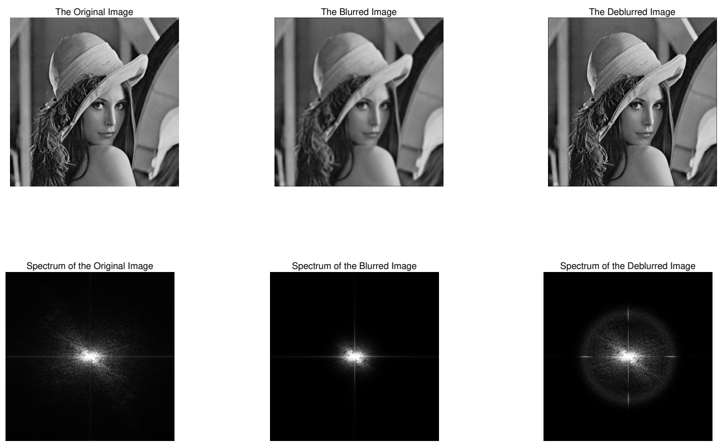
The Original Image
The Blurred Image
The Deblurred Image
Spectrum of the Original Image
Spectrum of the Blurred Image
Spectrum of the Deblurred Image

Figure 4: Wiener filter

4 Conclusions

Histogram equalization is a powerful tool to reconstruct a low-contrast image, but it is unable to completely reemerge the original picture because of its discrete nature. Additionally, the spatial image filters can play an important role in removing noise. Different spatial filters may lead to different results when filtering different kinds of noise. Thus, in order to achieve the best performance of noise reduction, we should select suitable spatial filters according to the characteristics of the specific noise. Finally, wiener filter has been shown to have a good performance in deblurring images. The key points for realizing this filter is to estimate two spectral densities in order to achieve optimized results.

Appendix

Who Did What

Report

Summary: Shuyi Chen
Introduction: Shuyi Chen
System description: Yuqi Zheng
Results:Ernan Wang
Conclusion:Ernan Wang, Shuyi Chen

Code

t3_wiener: Ernan Wang
Others: Yuqi Zheng

MatLab code

Histogram equalization

```
close all;
clear;

%% original image
lena=imread('lena512.bmp');
figure(1);
imshow(lena);
title('The Original Image');

hist1=hist(lena(:),0:255);
histlena=hist1./sum(hist1);
figure(2);
bar(histlena);
xlim([0,255]);
ylim([0,0.05]);
title('Histogram of the Original Image');

%% low-contrast image
a=0.2;
b=50;
lenalc=min(max(round(a*lena+b),0),255);
figure(3);
imshow(lenalc);
title('The Low-contrast Image');
```

```
hist2=hist(lenalc(:),0:255);
histlc=hist2./sum(hist2);
figure(4);
bar(histlc);
xlim([0,255]);
ylim([0,0.05]);
title('Histogram of the Low-contrast Image');
% figure(5);
% imagesc(lenalc,[50 100]);
% title('The Low-contrast Image');

%% equalization
s=round(255.*cumsum(histlc))+1;
hist3=zeros(1,256);
for ii=1:length(s)
    kk=s(ii);
    hist3(kk)=hist3(kk)+hist2(ii);
end
histeh=hist3./sum(hist3);
figure(6);
bar(histeh);
xlim([0,255]);
ylim([0,0.05]);
title('Histogram of the Enhanced Image');

figure(7);
lenaeh=histeq(lenalc);
imshow(lenaeh);
title('The Enhanced Image');
```

Image denoising

Gaussian Noises

```
close all;
clear;

%% original image
lena=imread('lena512.bmp');
figure(1);
imshow(lena);
title('The Original Image');

hist1=hist(lena(:),0:255);
histlena=hist1./sum(hist1);
figure(2);
bar(histlena);
xlim([0,255]);
ylim([0,0.01]);
title('Histogram of the Original Image');

%% Gaussian noise
gs=mynoisegen('gaussian',512,512,0,64);
lenand=lena+uint8(round(gs));
```

```
figure(3);
imshow(lenand);
title('The Gaussian-noised Image');

hist2=hist(lenand(:),0:255);
histnd=hist2./sum(hist2);
figure(4);
bar(histnd);
xlim([0,255]);
ylim([0,0.01]);
title('Histogram of the Gaussian-noised Image');

%% mean filter
% meanf=fspecial('average',[3,3]);
% lenameanf=imfilter(lenand,meanf);
meanf=ones(3)./9;
lenameanf=uint8(conv2(lenand,meanf,'same'));
figure(5);
imshow(lenameanf);
title('The Mean Filtered Gaussian-noised Image');

hist3=hist(lenameanf(:),0:255);
histmeanf=hist3./sum(hist3);
figure(6);
bar(histmeanf);
xlim([0,255]);
ylim([0,0.01]);
title('Histogram of the Mean Filtered Gaussian-noised Image');

%% median filter
lenamedf=medfilt2(lenand,[3,3]);
figure(7);
imshow(lenamedf);
title('The Median Filtered Gaussian-noised Image');

hist4=hist(lenamedf(:),0:255);
histmedf=hist4./sum(hist4);
figure(8);
bar(histmedf);
xlim([0,255]);
ylim([0,0.01]);
title('Histogram of the Median Filtered Gaussian-noised Image');
```

Salt & Pepper Noises

```
close all;
clear all;

%% original image
lena=imread('lena512.bmp');
figure(1);
imshow(lena);
title('The Original Image');

hist1=hist(lena(:),0:255);
```

```
histlena=hist1./sum(hist1);
figure(2);
bar(histlena);
ylim([0,0.01]);
title('Histogram of the Original Image');

%% Salt&pepper noise
lenand=lena;
sp=mynoisegen('saltpepper',512,512,.05,.05);
lenand(sp==0)=0;
lenand(sp==1)=255;
figure(3);
imshow(lenand);
title('The Salt&pepper-noised Image');

hist2=hist(lenand(:),0:255);
histnd=hist2./sum(hist2);
figure(4);
bar(histnd);
ylim([0,0.01]);
title('Histogram of the Salt&pepper-noised Image');

%% mean filter
% meanf=fspecial('average',[3,3]);
% lenameanf=imfilter(lenand,meanf);
meanf=ones(3)./9;
lenameanf=uint8(conv2(lenand,meanf,'same'));
figure(5);
imshow(lenameanf);
title('The Mean Filtered Salt&pepper-noised Image');

hist3=hist(lenameanf(:),0:255);
histmeanf=hist3./sum(hist3);
figure(6);
bar(histmeanf);
ylim([0,0.01]);
title('Histogram of the Mean Filtered Salt&pepper-noised Image');

%% median filter
lenamedf=medfilt2(lenand,[3,3]);
figure(7);
imshow(lenamedf);
title('The Median Filtered Salt&pepper-noised Image');

hist4=hist(lenamedf(:),0:255);
histmedf=hist4./sum(hist4);
figure(8);
bar(histmedf);
ylim([0,0.01]);
title('Histogram of the Median Filtered Salt&pepper-noised Image');
```

Frequency domain filtering

```
close all;
clear;
```

```
%% original image
lena=imread('lena512.bmp');
figure(1);
imshow(lena);
title('The Original Image');

lenaf=t3_spectragen(lena);
figure(2);
imshow(lenaf);
title('Spectrum of the Original Image');

%% blurred image
h=myblurgen('gaussian',8);
lenabld=uint8(min(max(round(conv2(lena,h,'same')),0),255));
figure(3);
imshow(lenabld);
title('The Blurred Image');

lenabf=t3_spectragen(lenabld);
figure(4);
imshow(lenabf);
title('Spectrum of the Blurred Image');

%% deblurred image
diff=min(max(round(conv2(lena,h,'same')),0),255)-conv2(lena,h,'same');
vn=var(diff(:));
lenadblr=t3_deblur(lenabld,h,vn);
figure(5);
imshow(lenadblr);
title('The Deblurred Image');

lenadblrf=t3_spectragen(lenadblr);
figure(6);
imshow(lenadblrf);
title('Spectrum of the Deblurred Image');

function [J] = t3_spectragen(I)
    fftI=fft2(I);
    sfftI=fftshift(fftI);
    A=abs(sfftI);
    J=(A-min(min(A)))./(max(max(A)-min(min(A)))).*255;
    % J=mat2gray(log(A));
end

function [J] = t3_deblur(g,h,v)
    g1=double(g);
    nsr=v/var(g1(:));
    g_taper=edgetaper(g1,h);
    J=uint8(t3_wiener(g_taper,h,nsr));
end

function [J] = t3_wiener(I,PSF,nsr)
% G(k,l) = (H*(k,l)./(|H(k,l)|^2 + NSR
%% H
    H = psf2otf(PSF, size(I));

%% G
    denom = (abs(H).^2) + nsr;
    G = conj(H) ./ denom;
    J = ifft2(G .* fft2(I));
end
```

References

[1] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Pearson Education, 4th ed., 2018