EQ2330 Image and Video Processing Project1

Chenting Zhang chzha@kth.se
Lukas Rapp lrapp@kth.se
Shaotian Wu shaotian@kth.se

November 21, 2022

Summary

The goal of this project is to investigate spatial and frequency domain image enhancement techniques. Firstly, we produce low-contrast images by using a mapping method and then do image enhancement by applying histogram equalization algorithm on them. Secondly, spatial smoothing filters and order-statistics filters are used to remove the noise from image that have been corrupted by Gaussian-noise or salt&pepper-noise. Finally, we use two different frequency domain filter, wiener filter and constrained least square filter, to implement and analyze methods to restore out-of-focus images.

1 Introduction

Spatial processing method and frequency processing method are widely used in digital image processing. For spatial processing, we are to have histogram equalization for the enhancement of the low-contrast image, and noise removal with mean filter and median filter. For frequency processing, we are to have two different processing methods that require different information about the whole system, and they are both used to do the unblurring of the given image.

2 System Description

2.1 Spatial domain processing

This part contains two sections: Histogram Equalization section and Image Denoising section.

2.1.1 Histogram Equalization

For the Histogram equalization section, we firstly draw the histogram of the image and simulate a low-contrast image by reducing the original intensity range of the image by applying a mathematical mapping as follows:

$$g(x,y) = \min(\max(\lfloor a \cdot f(x,y) + b \rceil), 0), 255), \tag{1}$$

where $\lfloor \cdot \rceil$ denotes the round operator, $f(x,y)$ and $g(x,y)$ denote the values of pixels in the original and output images. a and b are 0.2 and 50, respectively which indicates that the range of the gray level will be mapped into the interval $[50, 101]$.

After that, we use global histogram equalization to enhance the image. The algorithm is presented as follows:

$$s_k = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}, \quad \text{for } k = 0, 1, 2, \ldots, L-1. \tag{2}$$

Where n is the total number of pixels in the image, n_k is the number of pixels that have gray level r_k, L is the total number of possible gray levels in the image.

Thus, a output image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k in the output image.

2.1.2 Image Denoising

For the image denoising section, we would firstly add Gaussian noise and salt and pepper noise to the original image respectively.

The additive noise degradation model for images is written as follows:

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

where $\eta(x, y)$ denotes the additive Gaussian noise with zero mean and variance 64 in this case.

The pixel values of the output image corrupted by salt pepper noise would transform into 0 or 255 with an equal probability of 5%, which is represented as making some pixels white and some pixels black in binary image. Thus, there would be some black and white dots in the image which shaped like pepper visually.

After that, we would implement the 3×3 mean filter to the noisy image. The convolution mask for the 3×3 mean filter is written as follows:

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

In practice, every pixel value in the image would be replaced by the average value of its nine neighbor pixels.

Another noise removal method is median filtering. Median filtering is a nonlinear smoothing technique, which sets the gray value of each pixel as the median gray value of all pixels in a neighborhood window of the point which mainly uses the feature that the median is not affected by the maximum and minimum of the distribution sequence.

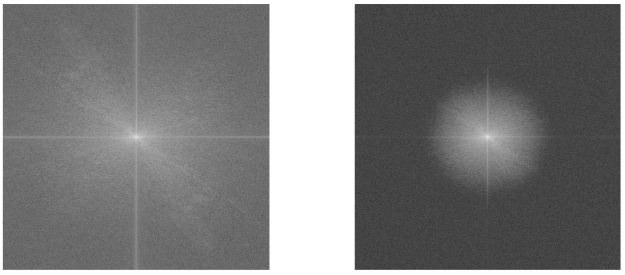
2.2 Frequency domain filtering

In the following, the reconstruction of a blurred image is considered. The degradation model is given by

$$g(x, y) = h(x, y) * f(x, y) + \eta(x, y),$$

where f is the original image, g the out-of-focus image and h is a Gaussian blur kernel. η is quantization noise which is introduced by quantizing the blurred image.

In a first step, the degradation model was implemented in Matlab and analyzed. The amplitude of the Fourier transformation of the blurred image and of the original image can be found in figure 1. High frequency components are dampened in the blurred image because the Gaussian kernel acts as low-pass



(a) Original Image (b) Blurred Image

Figure 1: Fourier transformation of the Lena example image visualized by the logarithm of its amplitude (images are normalized).

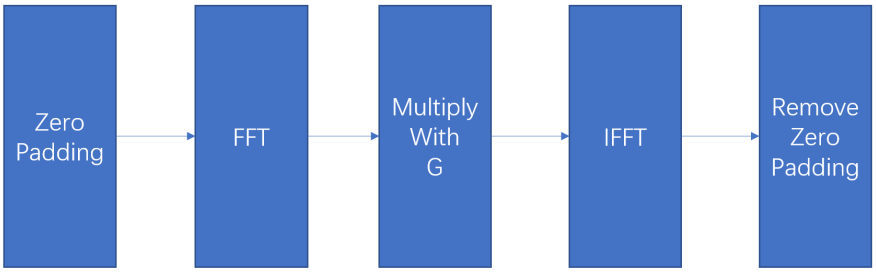


Figure 2: The flow chart of the reconstruction process

filter. To reconstruct the original image, the high frequency components must be amplified by a filter. However, a filter design must also consider the noise η because the noise is amplified too and can worsen the quality of the image.

The following two reconstruction filters are designed and analyzed in this project:

- Minimum mean square error (Wiener) filtering [1, Sec. 5.8]
- Constrained least squares filtering [1, Sec. 5.9]

2.2.1 Wiener filter

The wiener filter minimizes the expected mean square error between the uncorrected and corrupted image. It is given in frequency domain by [1, Sec. 5.8]

$$G(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)},$$

where S_η and S_f are the power spectral densities of the noise and image, respectively. However, for the given images of this project, both power spectral densities are not known. Therefore, we follow the approach of [1, Sec. 5.8] and replace the fraction by a constant $K \geq 0$, which is optimized by hand to achieve the best visual result:

$$G(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K}.$$

To reconstruct the image, the blurred image is filtered by G in frequency domain (see figure 2). For this, the image is transformed by a FFT, multiplied by G and then back-transformed by a IFFT. However, the multiplication in

frequency domain corresponds to a cyclic convolution in spatial domain because the FFT transformation implicitly assumes that the signal is period. To avoid this undesired behaviour, the image is zero padded before the FFT is applied. The width of the borders of the padded zeros must be larger or equal than the length of the impulse response of G to avoid errors. After the IFFT, the zeros are removed. Because of the zero padding, the multiplication in frequency domain behaves like a normal convolution in spatial domain.

The actual implementation of the reconstruction method can be found in Sec. A.2.3. One important difference between theory and implementation is the shift of the filters in line 54-56 and line 71-74. In the "fft2"' function in MatLab the center of the filter is in the upper left corner. This leads to artifacts in the reconstruction. To solve this issue, the reconstruction filters are shifted by modulation in frequency domain so that their center is at $(0, 0)$.

2.2.2 Constrained Least Squares Filter

A problem of the Wiener filtering is that it requires too many knowledge about the given picture. It requires the power spectra of the undegraded image and noise must be known also. In reality this requirement is not that easy to fulfill. Constrained least squares filter is another filter which only require the knowledge of the mean and the variance of the noise, which can be easily calculated from a given degraded image, so it is an important advantage. It is given in frequency domain by

$$G(u, v) = \frac{H^*(u, v)}{|H(u, v)|^2 + \gamma |P(u, v)^2|},$$

where $\gamma \geq 0$ is a parameter that must be adjusted so that the constraint in

$$||\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}||^2 = ||\boldsymbol{\eta}||^2$$

is satisfied. $P(u, v)$ is the Fourier transform of the function

$$p(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The reconstruction method of the constrained least squares filter is much similar to the Wiener filter. To reconstruct the image, the blurred image is filtered by G in frequency domain as described in Sec. 2.2.1 and Fig. 2.

3 Results

In this report, we would use the "Lena" image 512×512 as an example to demonstrate our results.

3.1 Spatial domain processing

The original image and its corresponding histogram are presented as Fig. 3, and the low-contrast image and its corresponding histogram are presented as Fig. 4 below.

Observing from Fig. 4, we could conclude that the gray level of the low-contrast histogram is squeezed into a small interval due to the mapping, which visually lowers the contrast and makes the image grayer.

After the histogram equalization, the enhanced image and its corresponding histogram are presented as Fig. 5.

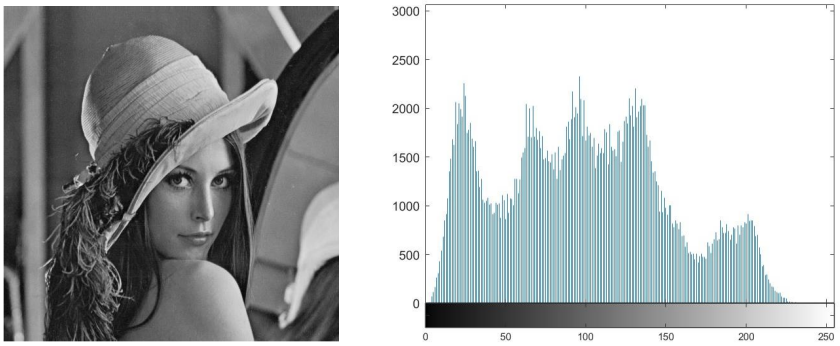


Figure 3: Original image and its histogram

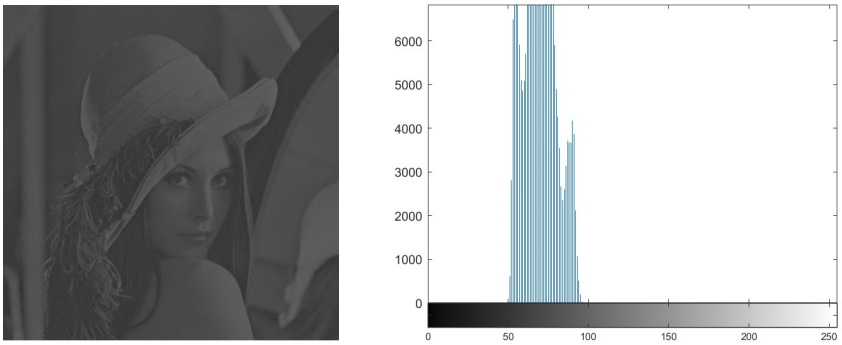


Figure 4: Low-contrast image and its histogram

For the continuous histogram equalization, the histogram of the output image would theoretically be flat and the output gray level would be characterized by a uniform probability density function. However, it is not the case in discrete histogram equalization. In the discrete case, the range of the gray level is stretched and the values are mapped into the interval $[0, 255]$, thus, enhancing the contrast of the image visually. However, the number of gray levels stays the same as in the low-contrast image.

By comparing the enhanced image with the original one, the enhanced image mainly preserves the shape and characteristics. However, it has less gray levels than that of the original one in the histogram because of the squeeze in the first step. Thus, there are more high frequencies in the image and the image

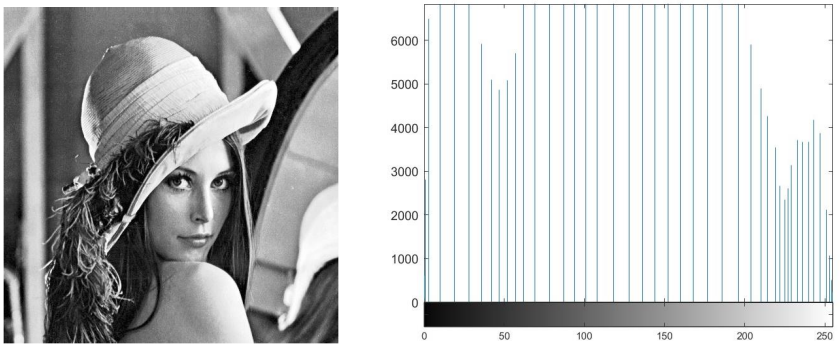


Figure 5: Enhanced image and its histogram

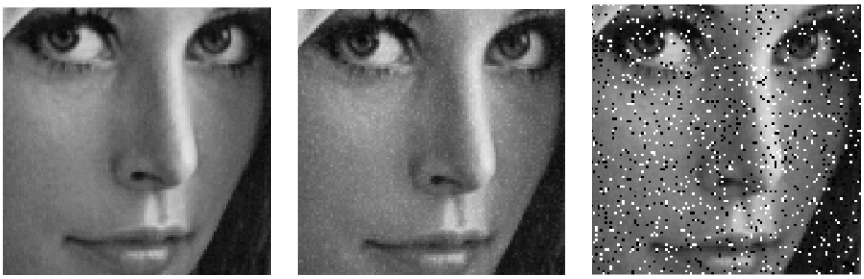
becomes sharper visually.

After adding Gaussian noise and salt and pepper noise to the original image, the corrupted images and their corresponding histograms are presented as Fig. 6 and Fig.7.

Figure 6: Gaussian noise corrupted image

Figure 7: Salt and pepper noise corrupted image

In Fig. 8, we compare the difference between original image and two kinds of corrupted images through the local part of the image (Lena's face).



(a) Original (b) with Gaussian noise (c) with salt noise

Figure 8: Cutout of Lean's face showing the corruption by noise.

Then, we implement mean filter and median filter to do image restoration. By implementing the 3×3 mean filter, every pixel value in the image would be replaced by the average value of its nine neighbor pixels. The restored images are presented as Fig. 9.

Figure 9: Mean filter through Gaussian and salt-pepper noise corrupted images

By implementing the median filter, the gray value of each pixel will be set

as the median gray value of all pixels in a neighboring window of the point. The restored images are presented as Fig. 10.

Figure 10: Median filter through Gaussian and salt-pepper noise corrupted images

We could observe from the images that the mean filter is more effective in Gaussian noise corrupted image than in the salt and pepper noise corrupted image. By blurring the image, the Gaussian noise is dissolved in the original image. However, due to some extreme pixel values $(0, 255)$ in the salt and pepper noise corrupted image, it may lead to some large deviation from the "right" pixel value after the averaging process. However, the median filter is an effective implementation for the salt and pepper noise due to the fact that the median value will not be affected by the maximum and minimum of the distribution sequence. Thus, the negative effect of the extreme values will be largely eliminated. As shown in the Fig. 10 (right), the black and white dots are largely removed.

3.2 Frequency domain processing

Fig. 11 shows the reconstruction process on the Lena example image. The reconstruction of the blurred image in Fig. 11(b) is almost perfect at first glance. On closer inspection of the images, it is noticeable that the reconstructed images are a bit grainy which is noticeable in large areas of constant color. In addition, there is a small loss of high frequency details in detailed areas, e.g. the hairs.

Fig. 11(e) shows a cutout of the reconstructed image in Fig. 11(c) showing artifacts on the border of the image which are due to the reconstruction filter. The reason for these artifacts is the blurring of the image with the convolution method "same". This method throws away the results of the convolution which are outside of the original image borders which leads to an information loss. This leads to the artifacts because the reconstruction algorithm uses partially zeros instead of the blurred image components to recover the image parts at the borders. The artifacts can be avoid by using a convolution with mode "full' which is shown in Fig. 11(f).

The reconstruction methods were also tested on the two out-of-focus test images. The results can be found in Fig. 12. The reconstruction methods improve the quality of the images significantly without introducing noticeable artifacts. Details in the image are visible again. The quality of the reconstruction based on the wiener filter and constrained least squares filter are almost identical.

(a) Original (b) Blurred (c) Reconstructed: Wiener Filter

(d) Reconstructed: Constrained least squares filter (e) Reconstructed cutout with convolution "same" (f) Reconstructed cutout with convolution "full"

Figure 11: Results reconstruction of the blurred Lena image. Images (e) and (f) are cutouts of the reconstructed image with the wiener filter (s. Fig. (c)), whereby the original image was blurred by different methods.

Figure 12: Reconstructed results: First column: Out-of-focus images, Second column: Wiener filter reconstruction result, Third column: result constrained least squares filter

4 Conclusions

In this project we use two different kinds of ways to process the image. One of them is in spatial processing way and another is in frequency processing way. For spatial processing way we managed to use histogram equalization and two filters, mean filter and median filter, to process the image and denoise the image from gaussian noise and salt and pepper noise. For frequency processing way we managed to use two different filters, wiener filter and constrained least square filter to do the deblurring of the image. All of these are different ways to deal with the original image to make it clearer.

A Appendix

A.1 Distribution

Author:

Chenting Zhang for spatial domain processing
Lukas Rapp for wiener filter
Shaotian Wu for constrained least square filter

A.2 MatLab code

A.2.1 Question 2.1

```
clear;
clc;
I=imread('lena512.bmp');
figure(1);
title('original figure')
subplot(1,2,1);
imshow(I);
%img=im2gray(I);
subplot(1,2,2);
imhist(I);
[pixelCounts1,grayLevels1]=imhist(I);
%high constrast

%do the mapping
I_map=uint8(zeros(512));
for i=1:512
    for j=1:512
        I_map(i,j)=min(max(round(0.2*I(i,j)+50),0),255);
    end
end
figure(2);
title('low-contrast image')
subplot(1,2,1);
%imagesc(I_map, [0 255]);
imshow(I_map);
subplot(1,2,2);
imhist(I_map);
[pixelCounts2,grayLevels2]=imhist(I_map);

% do the equalization
```

```
I_map_eq=uint8(zeros(512));
n=512*512;
gray_level_eq=255.*(cumsum(pixelCounts2)./n);

for i=1:512
    for j=1:512
        I_map_eq(i,j)=gray_level_eq(I_map(i,j));
    end
end
figure(3);
title('low-contrast image after equalization')
subplot(1,2,1);
%imagesc(I_map, [0 255]);
imshow(I_map_eq);
subplot(1,2,2);
imhist(I_map_eq);
```

A.2.2 Question 2.2

```
%generate Gaussian noise
noise_gaussian = mynoisegen('gaussian', 512, 512, 0, 64);
I_gaussian = I + uint8(noise_gaussian);

%generate saltpepper noise
I_saltp = I;
noise_saltpepper = mynoisegen('saltpepper', 512, 512,
    ↪ .05, .05);
I_saltp(noise_saltpepper==0) = 0;
I_saltp(noise_saltpepper==1) = 255;

figure(4);
subplot(1,2,1);
imshow(I_gaussian);
subplot(1,2,2);
imhist(I_gaussian);

figure(5);
subplot(1,2,1);
imshow(I_saltp);
subplot(1,2,2);
imhist(I_saltp);

% implement mean filter
f1=(1/9)*ones(3);
I_gaussian_output1 = uint8(conv2(I_gaussian, f1, 'same'))
    ↪ ;
I_saltp_output1 = uint8(conv2(I_saltp, f1, 'same'));

% implement median filter

I_gaussian_output2 = medfilt2(I_gaussian);
I_saltp_output2 = medfilt2(I_saltp);

figure(6);
```

```
subplot(1,2,1);
imshow(I_gaussian_output1);
subplot(1,2,2);
imhist(I_gaussian_output1);
figure(7);
subplot(1,2,1);
imshow(I_saltp_output1);
subplot(1,2,2);
imhist(I_saltp_output1);
figure(8);
subplot(1,2,1);
imshow(I_gaussian_output2);
subplot(1,2,2);
imhist(I_gaussian_output2);
figure(9);
subplot(1,2,1);
imshow(I_saltp_output2);
subplot(1,2,2);
imhist(I_saltp_output2);
```

A.2.3 Question 3

```
close all;
clear;

img = imread('images\lena512.bmp');
% img_out_of_focus = imread('images\boats512_outoffocus.bmp');
% img_out_of_focus = imread('images\man512_outoffocus.bmp');

% Apply degradation model
r = 8;
h = myblurgen('gaussian', r);
zero_padding = false;

if zero_padding
    img_padded = zeros(size(img).*3);
    img_padded((size(img, 1)+1):2*size(img, 1), (size(img, 2) + 1):2*size(img, 2)) = img;
else
    img_padded = img;
end
img_out_of_focus = conv2(img_padded, h, "same");
img_out_of_focus = min(max(round(img_out_of_focus), 0), 255);

% Plot FFT
img_fft = fft2(img_padded);
img_quant_padded_fft = fft2(double(img_out_of_focus));
figure();
plot_fft(img_quant_padded_fft);
title('FFT blurred image');
figure();
```

```
29 plot_fft(img_fft);
30 title('FFT original image');
31
32 K = 0.001;
33 img_reconstruced = deblurring(img_out_of_focus, true, h,
       ↪ K, "constrained");
34 imshow(uint8(img_reconstruced));
35
36 function img_reconstructed = deblurring(img_out_of_focus,
       ↪  zero_padding, h, parameter, mode)
37
38 % Add zero-padding
39 if zero_padding
40     img_padded = zeros(size(img_out_of_focus) .* 3);
41     img_padded((size(img_out_of_focus, 1)+1):2*size(
           ↪ img_out_of_focus, 1), ...
42         (size(img_out_of_focus, 2) + 1):2*size(
               ↪ img_out_of_focus, 2)) ...
43         = img_out_of_focus;
44 else
45     img_padded = img_out_of_focus;
46 end
47
48 % FFT
49 img_padded_fft = fft2(double(img_padded));
50
51 % Calculate reconstruction filter
52 shift_x = (size(h, 1) -1) / 2;
53 shift_y = -(size(h, 2) - 1) / 2;
54 h_fft = fft2(h, size(img_padded_fft, 1), size(
       ↪ img_padded_fft, 2)) ...
55     .* (exp(-1j*2*pi*shift_x*(0:(size(img_padded_fft, 1)
           ↪ -1))/size(img_padded_fft, 1))' ...
56     * exp(-1j*2*pi*shift_y*((0:size(img_padded_fft, 2)
           ↪ -1))/size(img_padded_fft, 2)));
57
58 if mode == "wiener"
59     G = conj(h_fft) ./ (abs(h_fft).^2 + parameter);
60 elseif mode == "constrained"
61     assert(mod(size(img_padded_fft, 1), 2) == 0);
62     assert(mod(size(img_padded_fft, 2), 2) == 0);
63     p = [
64         0, -1, 0;
65         -1, 4, -1;
66         0, -1, 0;
67     ];
68     shift_x = +1;
69     shift_y = -1;
70
71     p_fft = fft2(p, size(img_padded_fft, 1), size(
           ↪ img_padded_fft, 2)) ...
72     .* (exp(-1j*2*pi*(+shift_x)*(1:size(img_padded_fft,
           ↪ 1))/size(img_padded_fft, 1))' ...
73     * exp(-1j*2*pi*(+shift_y)*(1:size(img_padded_fft, 2))
```

```
        ↪ /size(img_padded_fft, 2)));
74      G = conj(h_fft) ./ (abs(h_fft).^2 + parameter * p_fft
        ↪ );
75  else
76      error("No valid mode");
77  end
78
79  % Apply reconstruction filter
80  img_reconstructed_fft = img_padded_fft .* G;
81
82  % IFFT
83  img_reconstructed_padded = ifft2(img_reconstructed_fft);
84
85  % Remove zero-padding
86  size_x = round(size(img_reconstructed_padded, 1)/3);
87  size_y = round(size(img_reconstructed_padded, 2)/3);
88  img_reconstructed = abs(img_reconstructed_padded((size_x
    ↪ +1):(2*size_x), (size_y+1):(2*size_y)));
89
90  end
91
92  function plot_fft(F)
93  F_shifted = fftshift(F);
94  F_abs_log = log(abs(F_shifted)+1);
95  F_img = mat2gray(F_abs_log);
96  imshow(F_img,[]);
97  end
```

References

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