VIETNAM NATIONAL UNIVERSITY

HCMC UNIVERSITY OF TECHNOLOGY

OFFICE FOR INTERNATIONAL STUDY PROGRAMS

COMPUTER VISION PROJECT

WIENER FILTER

A MATLAB IMPLEMENTATION IN IMAGE PROCESSING

Lecturer: Lê Thanh Hải, PhD

Students:

Trần Tiểu Bình	1752010
Lâm Hùng Minh	1752343
Cao Tri Thức	1752533
Đặng Minh Quân	1752445

HCMC, 2020

PREFACE

Modern digital technology has made it possible to manipulate multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems. The generation and development of digital image processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry and medical science has increased.

The inverse filtering is a restoration technique for deconvolution, i.e., when the image is blurred by a known lowpass filter, it is possible to recover the image by inverse filtering or generalized inverse filtering. However, inverse filtering is very sensitive to additive noise. The approach of reducing one degradation at a time allows us to develop a restoration algorithm for each type of degradation and simply combine them. The Wiener filtering executes an optimal tradeoff between inverse filtering and noise smoothing. It removes the additive noise and inverts the blurring simultaneously.

The Wiener filtering is optimal in terms of the mean square error. In other words, it minimizes the overall mean square error in the process of inverse filtering and noise smoothing. The Wiener filtering is a linear estimation of the original image.

TABLE OF CONTENT

CHAPTER 1: THEORY

1.1. Digital image

A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a sampling process that is frequently referred to as digitization. The effect of digitization of image $a(x,y)$ is divided into N rows and M columns. The intersection of a row and a column is termed a pixel. The value assigned to the integer coordinates [m,n] with $\{m = 0,1,2,\dots,M\ 1\}$ and $n = 0,1,2,\dots,N\ 1\}$ is $a[m,n]$.

1.2. Fourier transform

A Fourier transform (FT) is a mathematical transform that decomposes functions depending on space or time into functions depending on spatial or temporal frequency, such as the expression of a musical chord in terms of the volumes and frequencies of its constituent notes. (1.1) is 1D Fourier transform and (1.2) is 2D Fourier transform of continuous functions.

$$\hat{f}(\omega) = \frac{1}{\sqrt{m}} \int_{-\infty}^{\infty} \exp(\sigma q i \omega x) f(x)\, dx \tag{1.1}$$

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(ux+vy)}\, dx\, dy \tag{1.2}$$

In image processing, we care about 1D and 2D Fourier transform for discrete functions, which are called as discrete Fourier transform (DFT), as shown below.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn} \quad for\ 0 \le k \le N-1 \tag{1.3}$$

$$F[k,l] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] e^{-j2\pi\left(\frac{k}{M}m + \frac{l}{N}n\right)} \tag{1.4}$$

1.3. Convolution theorem

The convolution theorem states that under suitable conditions the Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms.

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \tag{1.5}$$

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\} \tag{1.6}$$

1.4. Inverse filter

An inverse filter is a tool to reconstruct the signal (here we have the image) which is degraded by another filter, noise or both.

Let **f** be the original image, **h** the blurring kernel, and **g** the blurred image. The idea in inverse filtering is to recover the original image from the blurred image. Taking Fourier transform:

$$g(x,y) = f(x,y) * h(x,y) \tag{1.7}$$
$$G(u,v) = F(u,v)H(u,v) \tag{1.8}$$
$$F(u,v) = \frac{1}{H(u,v)} G(u,v) \tag{1.9}$$

So $\frac{1}{H(u,v)}$ is the inverse filter.

1.5. Wiener filter

The goal of the Wiener filter is to compute a statistical estimate of an unknown signal using a related signal as an input and filtering that known signal to produce the estimate as an output. For example, the known signal might consist of an unknown signal of interest that has been corrupted by additive noise. The Wiener filter can be used to filter out the noise from the corrupted signal to provide an estimate of the underlying signal of interest. The Wiener filter is based on a statistical approach, and a more statistical account of the theory is given in the minimum mean square error (MMSE) estimator article.

Typical deterministic filters are designed for a desired frequency response. However, the design of the Wiener filter takes a different approach. One is assumed to have knowledge of the spectral properties of the original signal and the noise, and one seeks the linear time-invariant filter whose output would come as close to the original signal as possible. Wiener filters are characterized by the following:

1. Assumption: signal and (additive) noise are stationary linear stochastic processes with known spectral characteristics or known autocorrelation and cross-correlation

2. Requirement: the filter must be physically realizable/causal (this requirement can be dropped, resulting in a non-causal solution)
3. Performance criterion: minimum mean-square error (MMSE)

This filter is frequently used in the process of deconvolution; for this application, see Wiener deconvolution.

The Wiener filter is a popular inverse filter in image processing.

The formula of Wiener filter is

$$H(u,v) = \frac{K^*(u,v)}{K^*(u,v)K(u,v) + \left[S_n(u,v)/S_f(u,v)\right]}$$

where: $K^*(u, v)$ is a complex conjugate of $K(u,v)$, and $S(u, v)_n$ and $S(u,v)_f$ are the power spectrum of the noise and ideal image respectively. If the noise is uncorrelated then its power spectrum is easy available by:

$$S_n(u,v) = \sigma_n^2 \quad \text{for all } (u,v)$$

where: σ_n^2 is a noise variance.

We can estimate $S_f(u,v)$ by

$$S_f(u,v) \approx S_g(u,v) - \sigma_n^2$$

CHAPTER 2: MATLAB IMPLEMENTATION

All code and MATLAB script in this implementation is uploaded on GitHub: *https://github.com/Study-Group-BK/ComputerVision*

Here we use MATLAB R2020a

2.1. Construct Discrete Fourier Transform (DFT) script

Now we try to write a code to take Fourier transform of discrete function, following formula (1.3).

```
function [Xk] = dft(xn)
%   For length N input vector x, the DFT is a length N vector X,
%   with elements
%                    N-1
%      X(k) =       sum  x(n)*exp(-i*2*pi*k*n/N), 0 <= k <= N-1.
%                   n=0
       xn=xn(:); %make sure xn is a column vector
       N = length(xn);
       n = 0:1:N-1; %row vector for n
       k = 0:1:N-1; %row vector for k
       WN = exp(-i*2*pi/N); %Twiddle factor (w)
       nk = n'*k; %creates a N by N matrix of nk values
       WNnk = WN .^ nk; %DFT matrix
       Xk = WNnk*xn;
 end
```

And now we write function 2D DFT

```
function X = dft2(x)
        C = size(x,2); %number of columns
        R = size(x,1); %number of rows
        nR = 0:1:R-1; %row vector for n
        kR = 0:1:R-1; %row vector for k
           nkR=nR'*kR;
        nC = 0:1:C-1; %row vector for n
        kC = 0:1:C-1; %row vector for k
           nkC=nC'*kC;
        y = zeros(size(x));
        y1 = y;
        for c = 1:C
            y(:,c) = dft_row(x(:,c),R,nkR);
        end
        for r = 1:R
            y1(r,:) =dft_col(y(r,:).',C,nkC);
        end
        X = y1;
end

function [Xk] = dft_col(xn,N,nk)
       WN = exp(-i*2*pi/N); %Twiddle factor (w)
       WNnk = WN .^ nk; %DFT matrix
       Xk = WNnk*xn;
```

```
end
 function [Xk] = dft_row(xn,N,nk)
       WN = exp(-i*2*pi/N); %Twiddle factor (w)
       WNnk = WN .^ nk; %DFT matrix
       Xk = WNnk*xn;
 end
```

We use recursive functions to calculate 2D DFT, there are some constants that we put out of recursive function for better calculate time.

Here is the inverse Fourier transform

```
function [xn] = idft(Xk)
% The inverse DFT  is given by
%                     N-1
%      x(n) = (1/N) sum  X(k)*exp(i*2*pi*k*n/N), 0 <= n <= N-1.
%                    k=0
       Xk=Xk(:); %make sure xk is a column vector
       N = length(Xk);
       n = 0:1:N-1; %row vector for n
       k = 0:1:N-1; %row vector for k
       WN = exp(i*2*pi/N); %Twiddle factor (w)
       nk = n'*k; %creates a N by N matrix of nk values
       WNnk = WN .^nk; %DFT matrix
       xn = (1/N)*WNnk*Xk;
 end
```

And 2D inverse Fourier transform

```
function X = idft2(x)
        C = size(x,2); %number of columns
        R = size(x,1); %number of rows
        nR = 0:1:R-1; %row vector for n
        kR = 0:1:R-1; %row vector for k
        nkR=nR'*kR;
        nC = 0:1:C-1; %row vector for n
        kC = 0:1:C-1; %row vector for k
        nkC=nC'*kC;
        y = zeros(size(x));
        y1 = y;
        for c = 1:C
            y(:,c) = idft_row(x(:,c),R,nkR);
        end
        for r = 1:R
            y1(r,:) =idft_col(y(r,:).',C,nkC);
        end
        X = y1;
end

function [Xk] = idft_col(xn,N,nk)
       WN = exp(i*2*pi/N); %Twiddle factor (w)
       WNnk = WN .^ nk; %DFT matrix
       Xk = (1/N)*WNnk*xn;
end
 function [Xk] = idft_row(xn,N,nk)
       WN = exp(i*2*pi/N); %Twiddle factor (w)
       WNnk = WN .^ nk; %DFT matrix
       Xk = (1/N)*WNnk*xn;
```

```
end
```

2.2. Construct Wiener filter script

The Wiener filter is

```
function RestoredImage = WienerFilter_dft(y,h,sigma);
% RestoredImage = WienerFilter_fft(y,h,sigma);
% y: Degraded image (with blur and noise)
% h: Degrade kernel
% sigma: standard deviation of noise
R = size(y,1); %get number of rows
C = size(y,2); %get number of columns
Yf = dft2(y);
h=[h zeros(size(h,2),C-size(h,2));zeros(R-size(h,1),C)]; %Padding h with zeros
Hf = dft2(h);
Syy = abs(Yf).^2/(R*C);
See=sigma^2;
Sxx=(Syy-See);
Gf = conj(Hf)./(abs(Hf.^2)+sigma^2./Sxx);
eXf = Gf.*Yf;
RestoredImage = real(idft2(eXf));
return
```

Here is the script that load image, add noise, blur and restore it.

Clear

```
clear all
close all
clc
```

Input and display the binary image

```
I0 = imread('lenna.tif');
row = size(I0,1);
column=size(I0,2);
% I0=imresize(I0,[max(row,column) max(row,column)]);
% row=max(row,column);
% column=max(row,column);
I=rgb2gray(I0);
I=double(I);
```

Blur the image, corrupt the image using WGN and display it

h is the blurring kernel, and sigma is the noise standard deviation

```
h = ones(3,3)/9;

sigma = 10;

Xf = fft2(I);
```

```
Hf = fft2(h,row,column);

y = real(ifft2(Hf.*Xf))+sigma*randn(row,column); % circular convolution
```

Restoration using generalized Wiener filtering

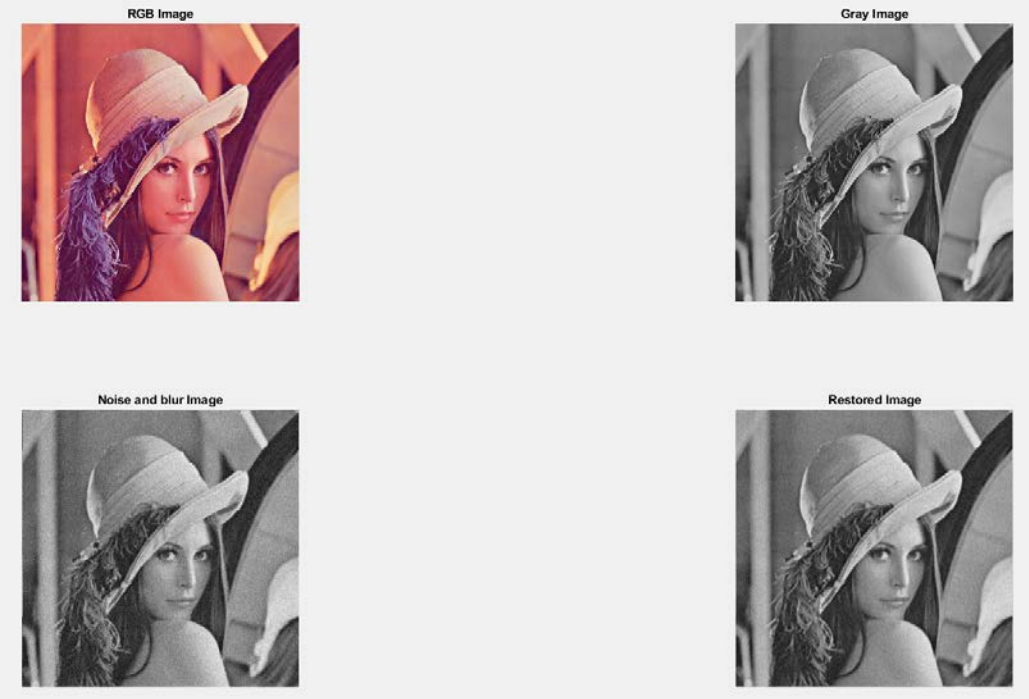
```
ewx = WienerFilter_fft(y,h,sigma);

PSNR = abs([psnr(I,I) psnr(y,I) psnr(ewx,I)]) %peak to noise ratio
MSE = [immse(I,I) immse(y,I) immse(ewx,I)] %Mean squared error
subplot(221)
imshow(I0)
title('RGB Image')
subplot(222)
imshow(I,gray(256))
title('Gray Image')
subplot(223)
imshow(y,gray(256))
title('Noise and blur Image')
subplot(224)
imshow(ewx,gray(256))
title('Restored Image')
return
```

The blur kernel we use is

$$H = \frac{1}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Here is the result.



Although the script works well, it is too slow in comparison with the script that used built-in MATLAB Fourier transform. When we measure with tic and toc function, the running time is $0.8s$ and $216s$ for built-in MALAB function and the code we write respectively.

To increase the run speed, we try to use other algorithm for Fourier transform.

2.3. Construct Wiener filter script

We use another algorithm for Fourier transform and inverse Fourier transform

Here is the new code for 2D Fourier transform

```
function K=kft2(k)
[M, N] = size(k);
wM        = zeros(M, M);
wN        = zeros(N, N);
for u = 0 : (M - 1)
        for x = 0 : (M - 1)
                wM(u+1, x+1) = exp(-2 * pi * 1i / M * x * u);
        end
end

for v = 0 : (N - 1)
        for y = 0 : (N - 1)
                wN(y+1, v+1) = exp(-2 * pi * 1i / N * y * v);
        end
end
K = wM * im2double(k) * wN;
```

And for 2D inverse Fourier transform

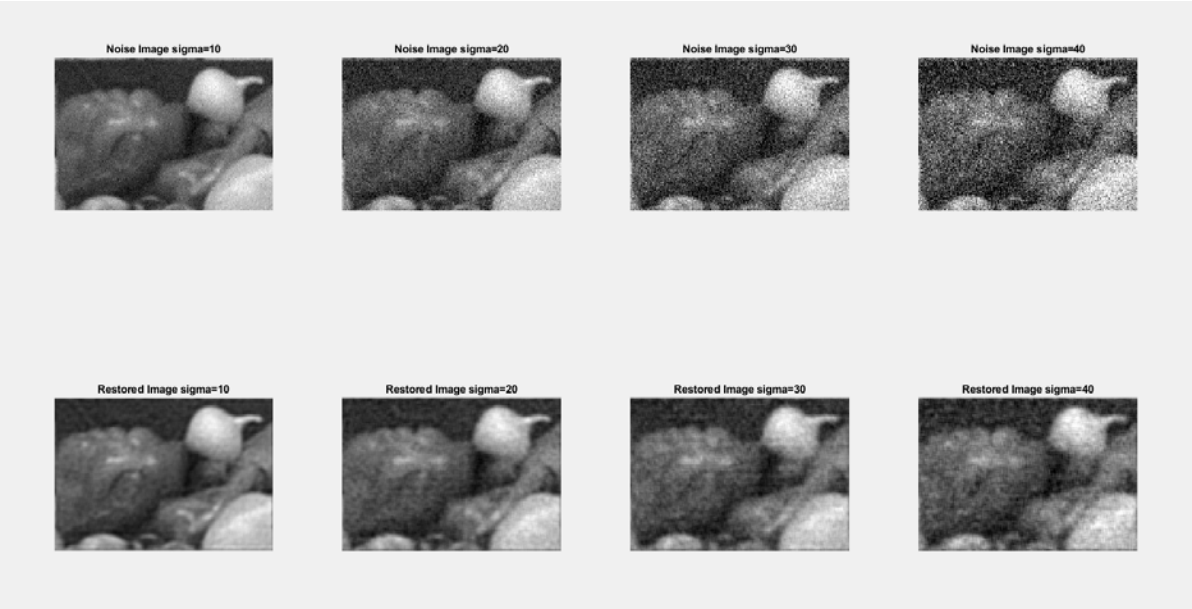
```
function KK=ikft2(K)
[M, N] = size(K);
wM = zeros(M, M);
wN = zeros(N, N);
for x = 0 : (M - 1)
        for u = 0 : (M - 1)
                wM(x+1,u+1) = exp(2 * pi * 1i / M * x * u);
        end
end
for y = 0 : (N - 1)
        for v = 0 : (N - 1)
                wN( v+1,y+1) = exp(2 * pi * 1i / N * y * v);
        end
end
KK= (1/(M*N))*(wM * im2double(K) * wN);
```

The speed seems to be faster, but it is still slow in comparison with the built-in functions of MATLAB.

2.4. Test the Wiener filter with different standard deviation of noise

The Wiener filter is very sensitive to additive noise, as mentioned above, we use Wiener filter with the assumption that the noise is stationary linear stochastic process which has mean $\mu = 0$ and standard deviation σ is known.

Now we try to apply the filter to different image with different σ



2.5. Conclusion

The Wiener filtering executes an optimal tradeoff between inverse filtering and noise smoothing. It removes the additive noise and inverts the blurring simultaneously. However, the drawback of Wiener filtering is that we have to know the kernel of the blur filter, and the

variance of the noise. In practice, the kernel $H(u, v)$ and the variance σ^2 are chosen by empirically. It is similar to Kalman filter, which we have to know the model of the system and the variance of noise to estimate the true measured value. Despite the drawbacks, the Wiener filter remains as a powerful tool to restore degraded image and it is still popular in image processing.

REFERENCES

A. Khireddine, K. Benmahammed, W. Puech. "Digital image restoration by Wiener filter in 2D case." (2004).

P. Bojarczak, Z. Łukasik. "IMAGE DEBLURRING – WIENER FILTER VERSUS TSVD APPROACH." *Advances in Electrical and Electronic Engineering* (n.d.).

Tran, Le-Anh. "Image Processing Course Project: Image Filtering with Wiener Filter and Median Filter." (2019).