Image Deblurring and Interpolation Using Least Squares Estimation

ECE 4260 Project Final Report

Ha, Soo Kwon

Yoo, Seung Min

Submitted

December 4, 2019

Abstract

Image manipulation and reconstruction has been an active research field regarding quality enhancement, compression, super resolution and so on. Various non-adaptive and adaptive interpolation methods are being used nowadays to reconstruct any corrupted image. Lease squares method is one of the optimization methods which proposes to minimize the error between the image to be reconstructed and the original image. We have implemented least square based method as a mean of reconstruction: deblurring a blurred image and magnifying a compressed image. For deblurring problem, we created blurred image by using Gaussian 2-D blurring kernel and motion blurring matrix, and then we computed convolution matrix from the blurred image to solve least square based deconvolution problem. Wiener filter was also used for deblurring process in order to compare the performances. For interpolation, we compressed image and then interpolated to the original size using corresponding LS based method. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) were used as evaluation metrics to compare between the original reference image and the reconstructed image.

1. Introduction

1.1 Background

In the real-life practice of image processing, images go through different kinds of manipulation and inevitably experience some degrades in image quality during manipulation-reconstruction stages. Blur in image can be defined as some vague and unclear distinctions between high and low frequency sections (i.e. edges) which in result make the image look “unfocused”. Especially during image capturing through cameras, images may not seem ideal due to different types of blur such as motion blur or gaussian blur (unclear focus). Deblurring is a technique to remove the blur and make images more distinctive. Image interpolation means obtaining a high-resolution image from a low-resolution image, which means the low-resolution pixels are interpolated to increase the image resolution. Image compression is reducing the size in bytes of images without severe degrading in image quality. Image can be compressed with many techniques, and similarly, the compressed images can be reconstructed back to normal size image by using both non-adaptive and adaptive interpolation techniques.

1.2 Least Squares and Image Deblurring

The usual objective of least square problem is to solve the following optimization problem:

$$\min_x \|Ax - y\|_2^2 \qquad (1)$$

where A is a known M by N matrix, x is N by L vector of parameters to be estimated and y is given observed data. Speaking in 2-D image processing domain, especially associating with image blurring and deblurring, we can state that A is M by N blurring matrix, x is the reconstructed or deblurred image and y is the observed image with blur.

Least square based image deblurring utilizes the fundamental 1-D deconvolution problem and maps to 2-D image domain. The 1-D least square based signal deconvolution defines as:

$$y[n] = h[0]x[n] + h[1]x[n-1] + \ldots + h[N]x[n-N] \qquad (2)$$

where y[n] is the output signal, h[n] represents impulse response of LTI system and x[n] is the estimated signal. We can rewrite this equation as matrix form of y = Hx where the convolution matrix H is shown in figure 1.

$$\begin{bmatrix} h_n & h_{n-1} & \cdots & h_1 & 0 & 0\cdots & 0 \\ 0 & h_n & h_{n-1} & \cdots & h_1 & 0\cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_n & h_{n-1} & \cdots & h_1 \end{bmatrix}$$

Figure 1. Visualization of convolution matrix H

Adding on top of the generic least square equation, the least square based deconvolution problem is defined as:

$$f(x) = \|y - Hx\|_2^2 + \lambda \|x\|_2^2 \quad (3)$$

where $\lambda > 0$ is a control parameter, also known as weight parameter or "smoothness" parameter. This parameter determines the smoothness between each pixel, so as λ approaches infinity the image becomes more of same color throughout the surface. Remember that the whole purpose is to find the estimator that minimizes the least square problem, so essentially, we are looking for a solution x that satisfies $\nabla f(x) = 0$. Rewriting the equation as

$$\nabla [(y - Hx)^T(y - HX) + (\lambda x^T x)] = 0 \quad (4)$$

derives to the solution for x as:

$$x = (H^T H + \lambda I)^{-1} H^T y \quad (5)$$

This equation is applied for both the cases of motion blur and gaussian blur with corresponding convolution matrix H and control parameter λ to reconstruct x with the known corrupted image y.

1.3 Least Squares and Image Interpolation

New-Edge Directed Interpolation (NEDI) is a linear spatially adaptive interpolation method which uses least squares to find structural similarity between the high-resolution and low-resolution pixels. NEDI method separates the interpolation process into three types of grid pixels: one with 4 closest neighbors in diagonal direction (blues in figure 2), one with 2 closest neighbors in vertical direction (greens in figure 2) and one with 2 closest neighbors in horizontal direction (yellows in figure 2). The reds represent the low-resolution pixels from compressed image.

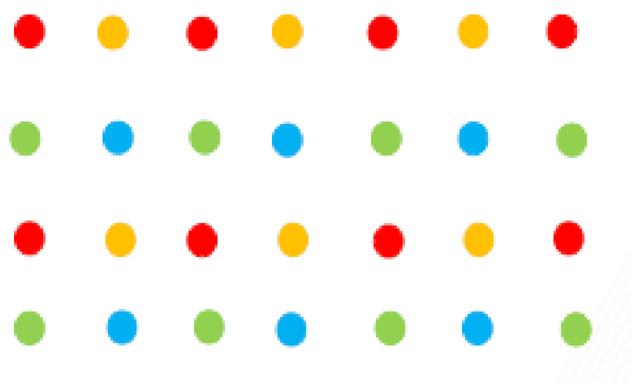

Figure 2. 3 types of grid pixels visualized in 3 different colors

The order of operation is first on blue pixels, and then on green pixels, and lastly on yellow pixels which will have 4 closest neighbors by then.

If a single set of interpolated pixel and its neighbor pixels are to be represented as shown in figure 3,

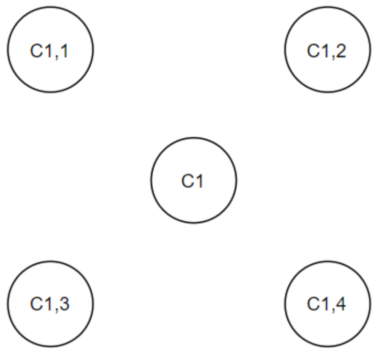

Figure 3. An estimation of C1 with 4 neighbor pixels

then M by M estimation window is created, and matrices C and y are represented as in figure 4 where C is a matrix of size 4 by M^2 containing interpolation neighbors of each pixel and y is a column vector of size M^2 containing the pixels in the window.

$$\mathbf{C} = \begin{bmatrix} C_{1,1} & C_{1,2} & C_{1,3} & C_{1,4} \\ C_{2,1} & C_{2,2} & C_{2,3} & C_{2,4} \\ \vdots & \vdots & \vdots & \vdots \\ C_{M^2,1} & C_{M^2,2} & C_{M^2,2} & C_{M^2,2} \end{bmatrix} \quad \overline{y} = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{M^2} \end{bmatrix}$$

Figure 4. Matrix C and y to be used for linear estimation

The goal of this approach is to find the coefficients $\alpha = [\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4]^T$ so that the $\|\text{error}\|^2$ is minimized where the error parameter is expressed as:

$$e = \underbrace{\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{M^2} \end{bmatrix}}_{\mathbf{y}} - \underbrace{\begin{bmatrix} C_{1,1} & C_{1,2} & C_{1,3} & C_{1,4} \\ C_{2,1} & C_{2,2} & C_{2,3} & C_{2,4} \\ \vdots & \vdots & \vdots & \vdots \\ C_{M^2,1} & C_{M^2,2} & C_{M^2,2} & C_{M^2,2} \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}}_{\boldsymbol{\alpha}}$$

Figure 5. Error minimization with estimated α

This leads to the least-square problem with the equation

$$\alpha = (C^TC)^{-1}(C^Ty) \tag{6}$$

where α is the reconstructed image and y is the input blurred image.

1.4 Wiener Filter

Wiener filter is a linear time-invariant (LTI) filter that is used to make estimation from an observed noisy process. The filter is expressed with the following equation:

$$W(u,v) = \frac{\hat{H}^*(u,v)}{|\hat{H}(u,v)|^2 + \frac{S_{vv}(u,v)}{S_{ff}(u,v)}} \tag{7}$$

where W(u,v) is the Wiener filter, H(u,v) is the Fourier transform of the point-spread function, and Svv/Sff can be interpreted as the reciprocal of signal-to-noise ratio (SNR).

Wiener filter can be used when the blurring operator and the power spectrum of the original image is known. The main objective of Wiener filter method is to find w[m,n] that minimizes the mean square error in the relationship S(u,v) = W(u,v)X(u,v) where S(u,v) is the original image spectrum and X(u,v) is the Fourier transform of the degraded image. This can be achieved by taking inverse discrete Fourier transform (iDFT) of the whole system and doing deconvolution between the blurring matrix and the degraded image.

Least square estimation converges to Wiener filter solution in the signal processing domain with assumptions that the signal and noise are stationary linear stochastic processes with known autocorrelation.

2. Problem Statement

2.1 Objective

The main objective of the whole experiment is to apply least squares estimation method to reconstruct the blurred image and compressed image based on the mathematical derivations presented in section 1.2, 1.3 and 1.4. The reconstructed images are compared with the original reference images to see if the reconstruction was successful.

2.2 Design Approach

The images are manipulated using known methods: for compression, resampling method is used, and for blurring, 2D Gaussian kernel and motion blurring matrix is applied on the selected image to make Gaussian blur and motion blur. The blurred image and the original image were compared to extract the convolution matrix H, and H is used to compute the least square estimated image using the equation (5). Different coefficients for weight parameter λ are used to compare the deblurred results. Wiener deconvolution is also done with both motion blur and gaussian blur with the same blurring matrices applied, and similarly the deblurred image and the original image are evaluated with PSNR and SSIM.

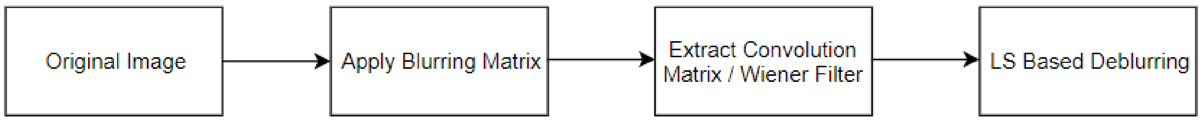

Figure 6. The chosen design of generalized deblurring process

In case of image compression and interpolation, each pixel and corresponding neighbor pixels are used to estimate each new pixel to be generated, therefore each covariances is calculated for each interpolation process. For the reference image for compression, we have chosen lena.png and conducted downsampling to have lower-size image.

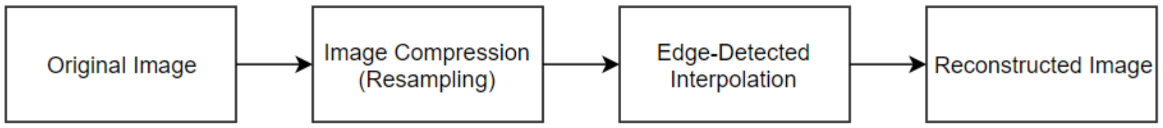

Figure 7. The chosen design of generalized enlargement process

3. Results and Analysis

3.1 Experiment Results for Deblurring

The vertical motion blur was created by averaging each pixel with 10 pixels in upper and lower part of image and then constructing sparse matrix to blur. Figure 8 shows the comparison between the original image and the vertically motion-blurred image.

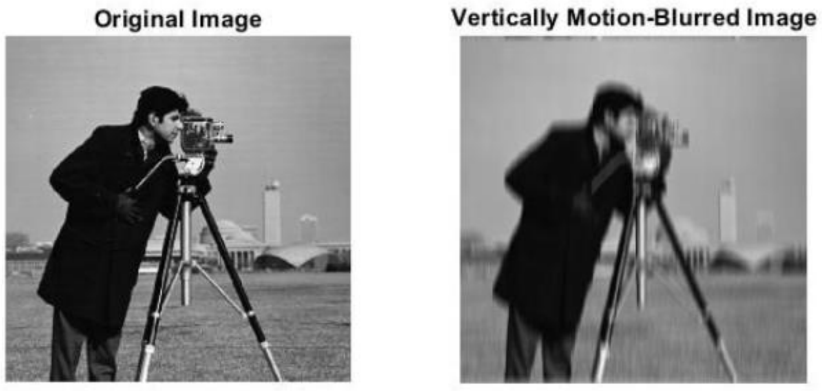

Figure 8. Illustration of vertical motion blur

The convolution matrix H was computed by directly doing H = yx^{-1} where y is the blurred image and x is the original input image. The computed H was verified by conducting $\tilde{y} = Hx$ and comparing with the original blurred image y if $y = \tilde{y}$.

According to the least square solution derived in equation (5), different λ values were chosen to see how smoothly the blur was removed. Figure 9 shows the result of deblurred images with different weight parameter.

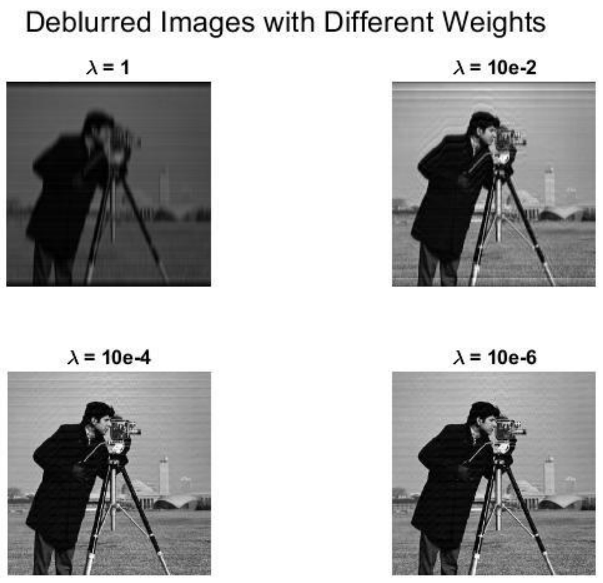

Figure 9. Results of deblurring with various values of λ

Gaussian blur was applied with 2-D gaussian kernel that has kernel size of 10 and variance of 1. This kernel acts as a low-pass filter on a target image, so all parts on image that correspond to the high frequencies are filtered out and therefore the image looks blurry because there are fewer sharp changes on the edges.

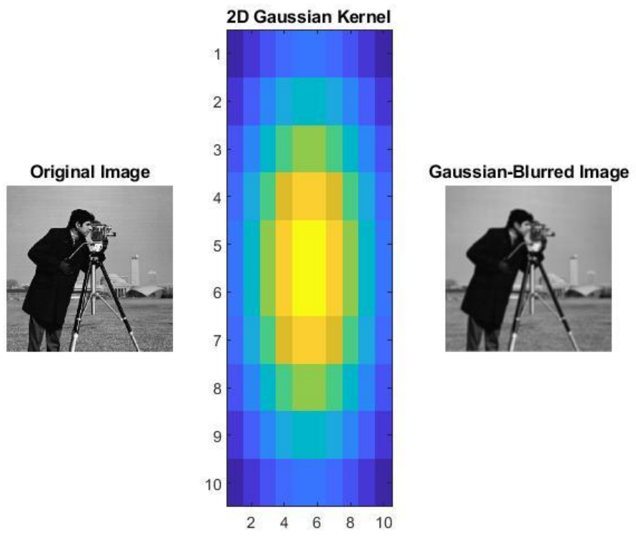

Figure 10. Illustration of Gaussian blur and lowpass filter

Same as the motion blur case, the convolution matrix was computed and applied to make a deblurred images as shown in figure 11.

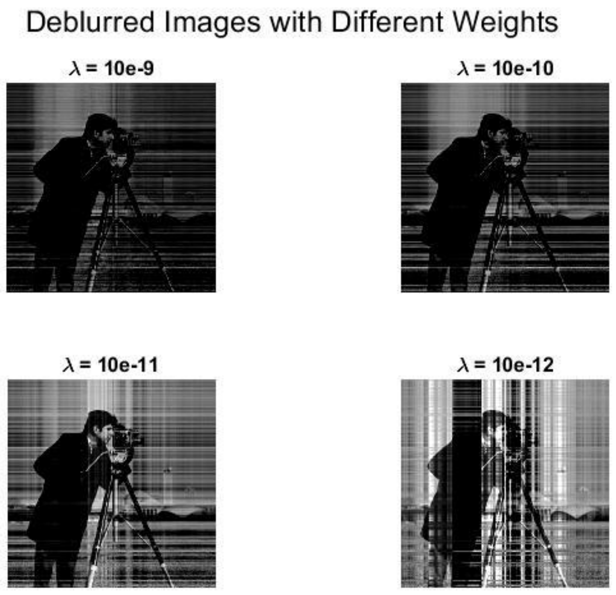

Figure 11. Results of deblurring with various values of λ

3.3 Experiment Results with Wiener Filter

For the Wiener filter case, the same blurring matrices from previous section were used for each motion blur and gaussian blur. Figure 12 and 13 show the Wiener filtered images.

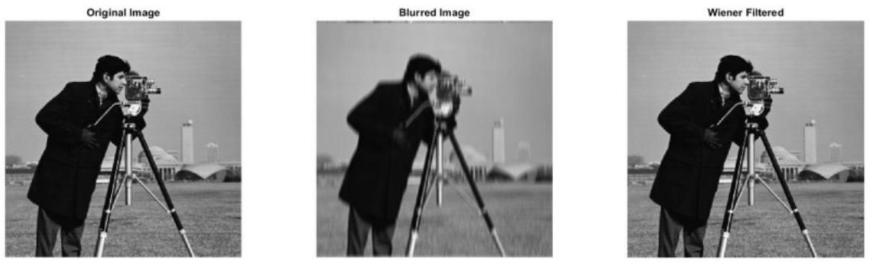

Figure 12. Motion deblurring with Wiener filter

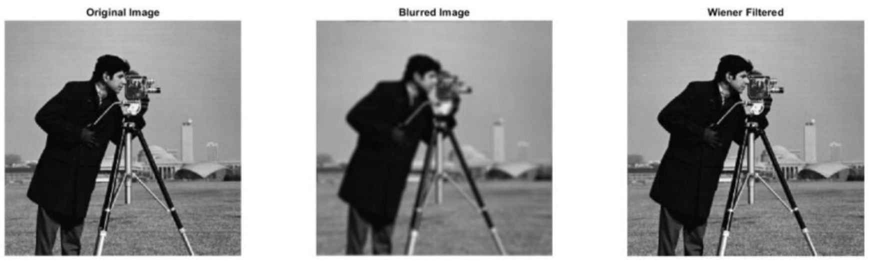

Figure 13. Gaussian deblurring with Wiener filter

3.4 Experiment Results for Interpolation

Lena.png was used for the reference image. The least square was applied through existing NEDI algorithm as presented in equation (6). The original image is 512 by 512 size, and the compressed image was downsampled to have a size of 128 by 128. The compressed image is then interpolated to have 512 by 512 size with empty pixels correspondingly estimated by using neighboring pixels. Figure 14 shows the comparison between the original image and the interpolated image from compression.

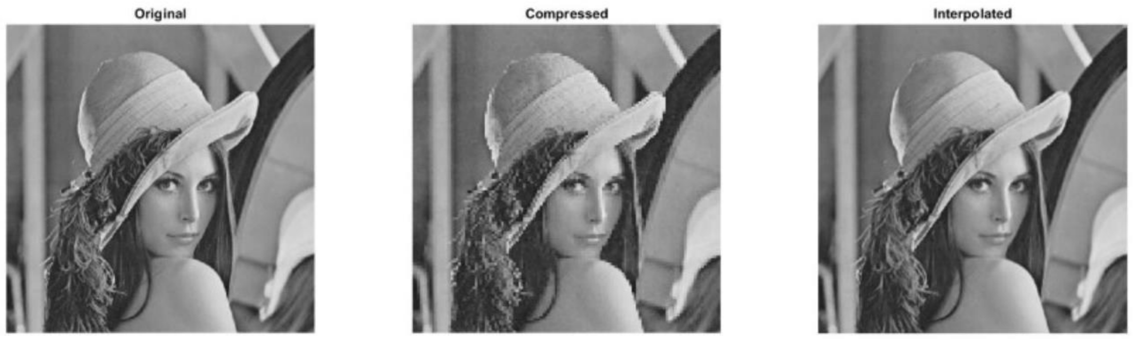

Figure 14. Illustration of compressed and interpolated lena.png

The downsampled image presents more 'boxy' texture compare to the original and the interpolated images.

3.5 Evaluation

Table 1. Reconstruction performance respect to the reference image

Case		PSNR (dB)	SSIM
Motion Blur	$\lambda = 1$	11.0363	0.5622
	$\lambda = 10^{-2}$	26.0479	0.8204
	$\lambda = 10^{-4}$	34.2779	0.9120
	$\lambda = 10^{-6}$	36.2839	0.9428
Gaussian Blur	$\lambda = 10^{-9}$	6.8795	0.0333
	$\lambda = 10^{-10}$	7.5367	0.0686
	$\lambda = 10^{-11}$	11.8938	0.2661
	$\lambda = 10^{-12}$	8.0340	0.3419
Motion Blur (Wiener Filter)		279.5073	0.9998
Gaussian Blur (Wiener Filter)		55.2152	0.9982
Interpolation (Enlargement)		33.6972	0.9125

PSNR measures the ratio between the maximum power of a signal and the power of noise, and SSIM measures the perceived quality of digital images. Higher values of PSNR and SSIM represent more successful reconstruction from corrupted image.

The reconstruction performance and evaluations are shown in table 1. According to the table, it is observable that deblurring worked better with smaller λ parameter applied, and the NEDI based least square method for image interpolation was also successful with high PSNR and SSIM value.

4. Conclusion

4.1 Discussion

As a conclusion, we have figured out that the least square method quite reasonably manages to reconstruct from the blurred or compressed images. The use of Wiener filter was a good choice of solution for enhancing the reconstruction quality for both motion and gaussian blur.

The one thing to discuss is that as shown in the result and the evaluated scores, the overall deblurring process worked better for the image blurred with motion blur compare to the Gaussian blur case. The reconstructed results from gaussian-blurred images contain more unexpected noises, which means that either the reference matrix is close to singular or badly scaled respect to the selected weight parameter λ. Also, an interesting result in Gaussian deblurring case is that the reconstructed image for $\lambda = 10^{-11}$ looks clearer and has higher PSNR than with the parameter $\lambda = 10^{-12}$, but the resulting SSIM for the latter case is higher than the former case. We are assuming that this is because both images are not very well reconstructed therefore the complier is looking at the former case as the image with more structural corruption.

4.2 Future Objective

Since we have achieved successful deblurring based on LS weight parameters and Wiener filter, the potential future application could be solving the equivalent problem with different estimation methods such as Maximum Likelihood (ML).

5. References

[1] Henry Stark, John Woods. 2011. Probability, Statistics, and Random Processes for Engineers. Prentice Hall. ISBN 0132311232, ISBN 978-0132311236

[2] Edward Dougherty. 1999. Random Processes for Image and Signal Processing. SPIE Optical Engineering Press. Texas A&M University. ISBN 0-8194-2513-3, ISBN 0-7803-3495-7

[3] Alberto Leon-Garcia. 2008. Probability, Statistics, and Random Processes for Electrical Engineering. Prentice Hall. ISBN 0-13-147122-8, ISBN 978-0-13-147122-1

[4] Steve Miller. The Method of Least Squares. Brown University.

[5] Zeyu Li, Jinling Wang. Least Squares Image Matching: A Comparison of the Performance of Robust Estimators

[6] Predreg Stanimirovic, Igor Stojanovic, Vasilios Katsikis, Dimitrios Pappas, Zoran Zdravev. Application of the Least Squares Solutions in Image Deblurring. Hindawi Publishing Corporation. 2015. Article ID 298689.

[7] Shimil Jose, Neethu Mohan, Sowmya V, Soman K.P. Amrita School of Engineering, Coimbatore. Least square based image deblurring. 2017. DOI: 10.1109/ICACCI.2017.8126045.

[8] Xin Li. New Edge-Directed Interpolation. IEEE. 2001. Vol.10, No.10.

[9] Kwok-Wai Hung, Wan-Chi Siu. Improved Image Interpolation Using Bilateral Filter For Weighted Least Square Estimation. Center for Sinal Processing. The Hong Kong Polytechnic University. 2010.

Appendix A

```
%Deblurring gaussian blur with Least Square and Wiener Filter

clear all
close all
clc

X = imread('cameraman.tif');
figure(1)
subplot(1,3,1)
imshow(X)
title('Original Image')
Xdouble = im2double(X);

%making 2-D Gaussian kernel
%low pass filter, therefore removes sharp edges
kernel_size = 10;
sigma = 1;
G = fspecial('gaussian',kernel_size,sigma);
blurred = imfilter(Xdouble,G,'conv','circular');
subplot(1,3,2)
imagesc(G)
title('2D Gaussian Kernel')
subplot(1,3,3)
imshow(blurred)
title('Gaussian-Blurred Image')

H = blurred*inv(Xdouble); %convolution matrix

% Reconstruction via deconvolution using LMS
figure(3)
subplot(1,3,1)
imshow(Xdouble)
title('Original Image')
subplot(1,3,2)
imshow(blurred)
title('Blurred Image')
subplot(1,3,3)
rew = deconvwnr(blurred,G);
imshow(rew)
title('Wiener Filtered')
```

```
ss = size(H);
lambda1 = 0.000000001;
re1 = inv(transpose(H)*H+lambda1*eye(ss(1)))*transpose(H)*blurred;
lambda2 = 0.0000000001;
re2 = inv(transpose(H)*H+lambda2*eye(ss(1)))*transpose(H)*blurred;
lambda3 = 0.00000000001;
re3 = inv(transpose(H)*H+lambda3*eye(ss(1)))*transpose(H)*blurred;
lambda4 = 0.000000000001;
re4 = inv(transpose(H)*H+lambda4*eye(ss(1)))*transpose(H)*blurred;
figure(4)
subplot(2,2,1)
imshow(re1)
title('{\lambda} = 10e-9')
subplot(2,2,2)
imshow(re2)
title('{\lambda} = 10e-10')
subplot(2,2,3)
imshow(re3)
title('{\lambda} = 10e-11')
subplot(2,2,4)
imshow(re4)
title('{\lambda} = 10e-12')
suptitle('Deblurred Images with Different Weights')

peaksnrw = psnr(rew,Xdouble)
ssimvalw = ssim(rew,Xdouble)

peaksnr1 = psnr(re1,Xdouble)
peaksnr2 = psnr(re2,Xdouble)
peaksnr3 = psnr(re3,Xdouble)
peaksnr4 = psnr(re4,Xdouble)

ssimval1 = ssim(re1,Xdouble)
ssimval2 = ssim(re2,Xdouble)
ssimval3 = ssim(re3,Xdouble)
ssimval4 = ssim(re4,Xdouble)
```

Appendix B

```
%Deblurring motion blur with Least Squares and Wiener Filter

clear all
close all
clc

X = imread('cameraman.tif');
figure(1)
subplot(1,2,1)
imshow(X)
title('Original Image')
Xdouble = im2double(X);
[m,n] = size(Xdouble);
mn = m*n;
blur = 10;
mindex = 1:mn;
nindex = 1:mn;
for i = 1:blur
  mindex=[mindex i+1:mn 1:mn-i];
  nindex=[nindex 1:mn-i i+1:mn];
end
D = sparse(mindex,nindex,1/(2*blur+1));
G = D*Xdouble(:);
HH = reshape(G,m,n);
subplot(1,2,2)
imshow(HH);
title('Vertically Motion-Blurred Image')

H = HH*inv(Xdouble); %convolution matrix
ss = size(H);
lambda1 = 1;
re1 = inv(transpose(H)*H+lambda1*eye(ss(1)))*transpose(H)*HH;
lambda2 = 0.01;
re2 = inv(transpose(H)*H+lambda2*eye(ss(1)))*transpose(H)*HH;
lambda3 = 0.0001;
re3 = inv(transpose(H)*H+lambda3*eye(ss(1)))*transpose(H)*HH;
lambda4 = 0.000001;
re4 = inv(transpose(H)*H+lambda4*eye(ss(1)))*transpose(H)*HH;
figure(2)
subplot(2,2,1)
imshow(re1)
```

```
title('{\lambda} = 1')
subplot(2,2,2)
imshow(re2)
title('{\lambda} = 10e-2')
subplot(2,2,3)
imshow(re3)
title('{\lambda} = 10e-4')
subplot(2,2,4)
imshow(re4)
title('{\lambda} = 10e-6')
suptitle('Deblurred Images with Different Weights')

peaksnr1 = psnr(re1,Xdouble)
peaksnr2 = psnr(re2,Xdouble)
peaksnr3 = psnr(re3,Xdouble)
peaksnr4 = psnr(re4,Xdouble)
ssimval1 = ssim(re1,Xdouble)
ssimval2 = ssim(re2,Xdouble)
ssimval3 = ssim(re3,Xdouble)
ssimval4 = ssim(re4,Xdouble)

LEN = 15; THETA = 90;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(Xdouble, PSF, 'conv', 'circular');
figure(3)
imshow(blurred)
rew = deconvwnr(blurred,PSF);
figure(4)
imshow(rew)
title('Reconstructed Image Using Wiener Filter')

figure(5)
subplot(1,3,1)
imshow(Xdouble)
title('Original Image')
subplot(1,3,2)
imshow(blurred)
title('Blurred Image')
subplot(1,3,3)
imshow(rew)
title('Wiener Filtered')
```

Appendix C

```
%NEDI

clear all
close all
clc

image=imread('lena.pgm');
image=im2double(image);
[m, n]=size(image);

image256 = im2double(zeros(256,256));
image128 = im2double(zeros(128,128));

for i=1:m/2
    for j=1:n/2
        image256(i,j)=image(2*i-1,2*j-1);
    end
end
for i=1:m/4
    for j=1:n/4
        image128(i,j)=image256(2*i-1,2*j-1);
    end
end

imagenedi1 = nedipredicted(image);
imagenedi2 = nedipredicted(image256);

predicted = im2double(zeros(m/2,n/2));

for i=1:128
    for j=1:128
        predicted(2*i-1,2*j-1)=image128(i,j);
    end
end

%nedialgorithm.m
output = nedialgorithm(image256,image128,predicted,imagenedi2,image,imagenedi1);

k = imcrop(output,[1 1 n-2 m-2]);
p = imcrop(image,[1 1 n-2 m-2]);
error = imsubtract(k,p);
```

```
errorsquared = (sum(error(:).^2));
mse = errorsquared/(511*511);

subplot(1,3,1)
imshow(image)
title('Original')
subplot(1,3,2)
imshow(image128)
title('Compressed')
subplot(1,3,3)
imshow(output)
title('Interpolated')

peaksnr = 10*log10(1/mse)
ssimval = ssim(image,output)

function y = nedipredicted(x)
[m n] = size(x);
for i = 1:m/2
    for j = 1:n/2
        xodd(i,j) = x(2*i-1,2*j-1);
    end
end
y(:,:) = edi(xodd(:,:));
end

function y = edi(x)
y1 = edi1(x);
y = edi2(y1);
x = y;
end

function y = edi1(x)
[m,n] = size(x);
for i = 1:2
   for j = 1:2
      y(i:2:2*m,j:2:2*n) = x;
   end
end
T = 5;
ix = [-1 -1 1 1];
iy = [-1 1 -1 1];
mx = [0 0 1 1];
```

```
my = [0 1 0 1];
kk = 1;
for i = -2:4
   for j = -2:4
      nx(kk) = i;
      ny(kk) = j;
      kk = kk+1;
   end
end

for i = T:m-T
   for j = T:n-T
       s = diag(x(i+mx,j+my));
       if var(s) < 0.0314
           a = ones(4,1)/4;
       else
          for k = 1:4
              C(:,k) = diag(x(i+ix(k)+nx,j+iy(k)+ny));
          end
          r = diag(x(i+nx,j+ny));
          a = inv(C'*C)*(C'*r);
       end
       y(2*i,2*j) = sum(a.*s);
       if y(2*i,2*j)<0 || y(2*i,2*j)>1
          a = ones(4,1)/4;
          y(2*i,2*j) = sum(a.*s);
       end
   end
end
end

function y = edi2(x)
y = x;
[m,n] = size(x);
T = 8;
kk = 1;
mx = [0 0 1 -1];
my = [-1 1 0 0];
kk = 1;
for i = -2:3
   for j = -2:3
      nx(kk) = i+j-1;
      ny(kk) = i-j;
```

```
        kk = kk+1;
    end
end

for i = T:2:m-T
    for j = T+1:2:n-T
        for k = 1:4
            C(:,k) = diag(x(i+2*mx(k)+nx,j+2*my(k)+ny));
        end
        r = diag(x(i+nx,j+ny));
        s = diag(x(i+mx,j+my));
        if det(C'*C) == 0 || var(s) < 0.0314
            a = ones(4,1)/4;
        else
            a = inv(C'*C)*(C'*r);
        end
        y(i,j)=sum(a.*s);
        if y(i,j)<0|y(i,j)>255
            a=ones(4,1)/4;y(i,j)=sum(a.*s);
        end
    end
end

for i = T+1:2:m-T
    for j = T:2:n-T
        for k = 1:4
            C(:,k) = diag(x(i+2*mx(k)+nx,j+2*my(k)+ny));
        end
        r = diag(x(i+nx,j+ny));
        s = diag(x(i+mx,j+my));
        if det(C'*C)==0 || var(s)<0.0314
            a = ones(4,1)/4;
        else
            a = inv(C'*C)*(C'*r);
        end
        y(i,j) = sum(a.*s);
        if y(i,j)<0 || y(i,j)>255
            a = ones(4,1)/4;
            y(i,j) = sum(a.*s);
        end
    end
end
end
```