Homework 3 Report

Name: Xuhong Lin
Date: 11/17/2023
Instructor: Prof. Vishal Monga

Question1

Figure 1. Noised Image with Gaussian NOise of mean:0, variance:0.02

Figure 2. Applied Squared Average Filter to noised cameraman

Figure 3. Applied Linear Adaptive Filter to noised cameraman

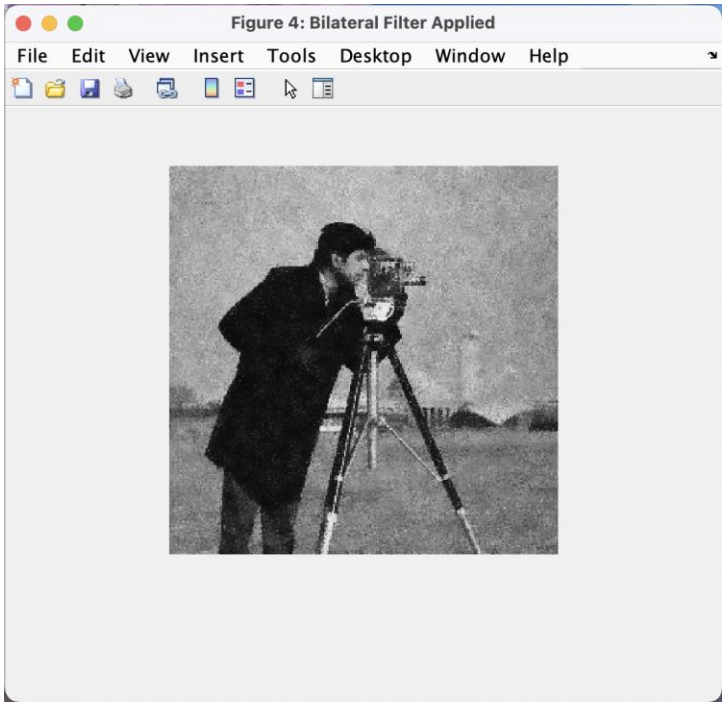

Figure 4. Applied Bilateral Filter to noised cameraman

Based on Figures 1 to 4, The Bilateral Filter has the best performance to denoise the Gaussian noise. Compared to the other two filters, the square average filter indeed reduced the noise since it did decrease the variance of the noise, but it also removed the fine detail or high-frequency component from the image. The Linear Adaptive Filter also reduces the noise but the original image has also been affected and we can see some unexpected points shown in the image. Finally, the Bilateral Filter reduces the noise. Compared to the other two filters, the edges and the person are more sharp than

the other two. By just looking at the image, the Bilateral filter has better performance visually.

```
%% Q1 Image Denoising
% read the image and make it double type
I = imread("cameraman.tif");
I = im2double(I);
% add the guassian noise with mean = 0 and variance = 0.02
I_noised = imnoise(I, "gaussian", 0, 0.02);
figure(Name="Noised Image with Gaussian Noise of mean: 0, variance:0.02");
imshow(I_noised);

% Square Average Filter
box_filter = 1/9 * [1 1 1; 1 1 1; 1 1 1];
% print image after applying Square Average Filter
I_Square = imfilter(I_noised, box_filter);
figure(Name="Square Average Filter applied"); imshow(I_Square);

% Linear Adaptive Filter
% Variable declearation
m = size(I_noised,1);
n = size(I_noised,2);
gaussVar = 0.02;
% try to apply the linear adaptive filter to get the output image with
% windoe size of 3x3
I_Linear = zeros(m, n);
for i = 1:m-2
    for j = 1:n-2
        window = I_noised(i:i+2, j:j+2);
        arithMean = mean(mean(window));
        arithVar = sum(sum(((window - arithMean).^2)/9));
        I_Linear(i:i+2,j:j+2) = I_noised(i:i+2,j:j+2) - (gaussVar/arithVar)*(I_noised(i:i+2,j:j+2)-arithMean);
    end
end
% plot the image
```

```
% Linear Adaptive Filter
% Variable declearation
m = size(I_noised,1);
n = size(I_noised,2);
gaussVar = 0.02;
% try to apply the linear adaptive filter to get the output image with
% windoe size of 3x3
I_Linear = zeros(m, n);
for i = 1:m-2
    for j = 1:n-2
        window = I_noised(i:i+2, j:j+2);
        arithMean = mean(mean(window));
        arithVar = sum(sum(((window - arithMean).^2)/9));
        I_Linear(i:i+2,j:j+2) = I_noised(i:i+2,j:j+2) - (gaussVar/arithVar)*(I_noised(i:i+2,j:j+2)-arithMean);
    end
end
% plot the image
figure(Name="Linear Adaptive Filter applied"); imshow(I_Linear);

% Bilateral Filter
% Inspect a patch of the image from the sky region. Compute the variance of
% the patch, which approximates the variance of the noise.
patch = imcrop(I_noised,[170, 35, 50 50]);
patchVar = std2(patch)^2;
DoS = 2*patchVar;
I_bilateral = imbilatfilt(I_noised,DoS,3);
figure(Name="Bilateral Filter Applied"); imshow(I_bilateral);
```

Figure 5,6. MATLAB code for question 1

Question 2

(1)

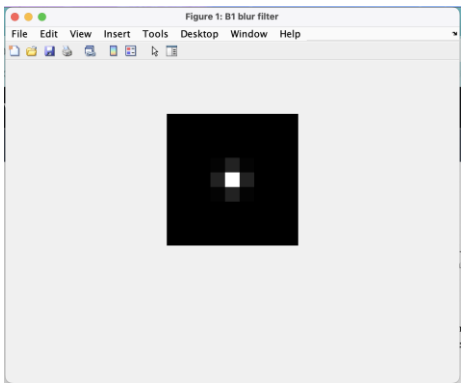

Figure 7. Image visualization of B1 Blur Matrix

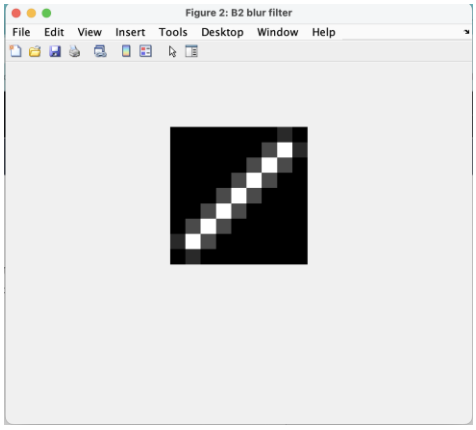

Figure 8. Image visualization of B2 Blur Matrix

```
%% Question 2
B1 = [0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0;
      0 0 0 0.0113 0.0837 0.0113 0 0 0;
      0 0 0 0.0837 0.6187 0.0837 0 0 0;
      0 0 0 0.0113 0.0837 0.0113 0 0 0;
      0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0];
B2 = [0 0 0 0 0 0 0 0.0145 0;
      0 0 0 0 0 0 0.0262 0.0896 0.0145;
      0 0 0 0 0 0.0262 0.0896 0.0262 0;
      0 0 0 0 0.0262 0.0896 0.0262 0 0;
      0 0 0 0.0262 0.0896 0.0262 0 0 0;
      0 0 0.0262 0.0896 0.0262 0 0 0 0;
      0 0.0262 0.0896 0.0262 0 0 0 0 0;
      0.0145 0.0896 0.0262 0 0 0 0 0 0;
      0 0.0145 0 0 0 0 0 0 0];
% show the B1 and B2 filter
%B1_norm = B1./norm(B1);
%B2_norm = B2./norm(B2);
B1_norm = (B1 - min(B1(:)))./(max(B1(:)) - min(B1(:)));
B2_norm = (B2 - min(B2(:)))./(max(B2(:)) - min(B2(:)));
Scale_B1_norm = imresize(B1_norm, 10,"nearest");
Scale_B2_norm = imresize(B2_norm, 10,"nearest");
figure(Name="B1 blur filter"); imshow(Scale_B1_norm);
figure(Name="B2 blur filter"); imshow(Scale_B2_norm);
```

Figure 9. MATLAB code to show B1 and B2 Blur Matrix

(2)

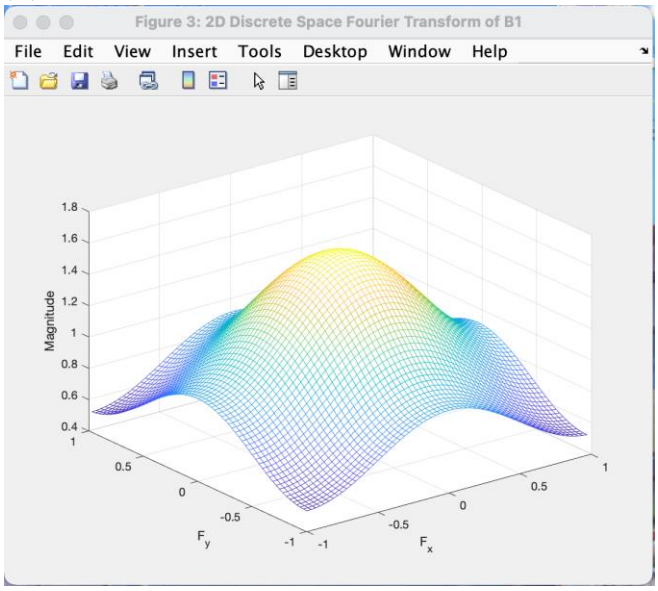

Figure 10. 2D Discrete Space Fourier Transform of B1

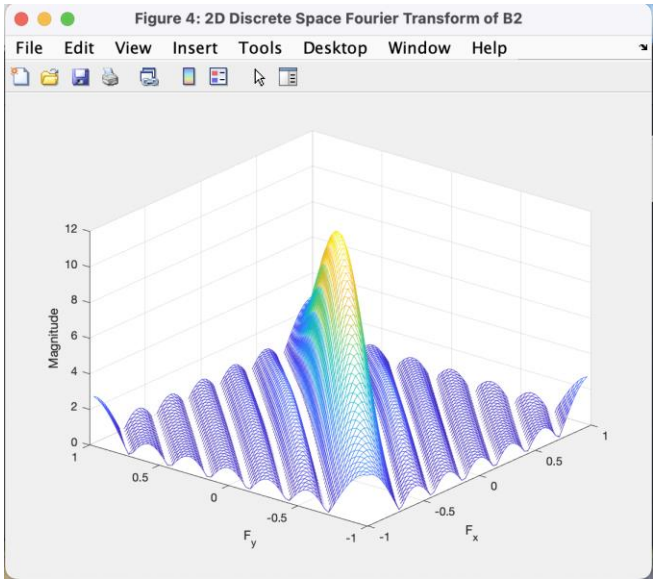

Figure 11. 2D Discrete Space Fourier Transform of B2

The 2D Discrete Space Fourier Transform of B1 looks like it is a lowpass filter in both the x and y directions. The 2D Discrete Space Fourier Transform of B2 is a lowpass filter in the diagonal direction of the X and Y axes. And as we can see from the plot, B1 has the same frequency characteristic in all directions so B1 is isotropic. Then, the B2 is more directionally sensitive.

```
% Display the 2D Discrete Space Fourier Transform using freqz2
figure(Name="2D Discrete Space Fourier Transform of B1");
freqz2(B1_norm);
figure(Name="2D Discrete Space Fourier Transform of B2");
freqz2(B2_norm);
```

Figure 12. MATLAB code to plot the 2D discrete Space Fourier Transform of B1 and B2

(3)

The B1 is the Gaussian Blur Filter and the B2 is the Motion Blur filter. Since Gaussian blur has the property of high intensity at the center of the filter and decays the intensity from the center. And the B1 fits this property. Then, take a look at the B2, we see that the diagonal direction has a high magnitude. When applying this filter to images, the resulting image will focus more on the diagonal direction of the image. It will look like the original image will be stretched out along the diagonal direction. Therefore, B2 is the Motion Blur filter.

(4)

Figure 13. B1 (Gaussian blur) blur filter was applied to the cameraman

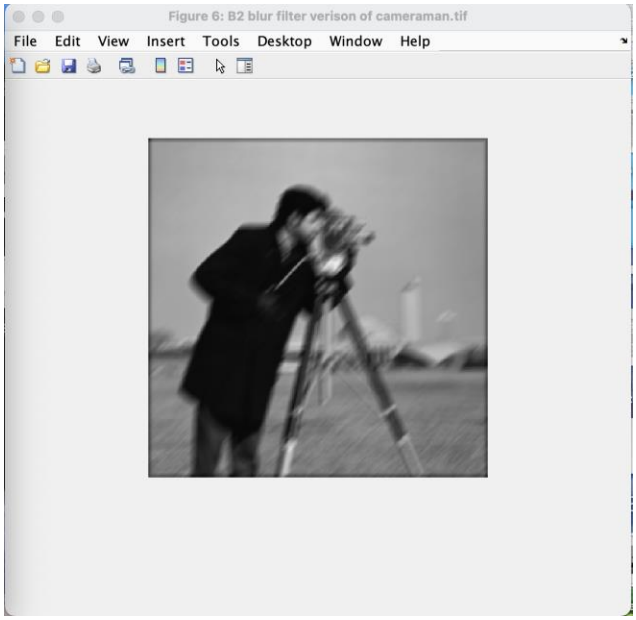

Figure 14. B2 (Motion Blur) blur filter applied to the cameraman

```
% Display the blurred version of cameraman
I_B1 = imfilter(I, B1, "conv");
I_B2 = imfilter(I, B2, "conv");
figure(Name="B1 blur filter version of cameraman.tif");
imshow(I_B1);
figure(Name="B2 blur filter verison of cameraman.tif");
imshow(I_B2);
```

Figure 15. MATLAB code to apply B1 and B2 Blur to the cameraman

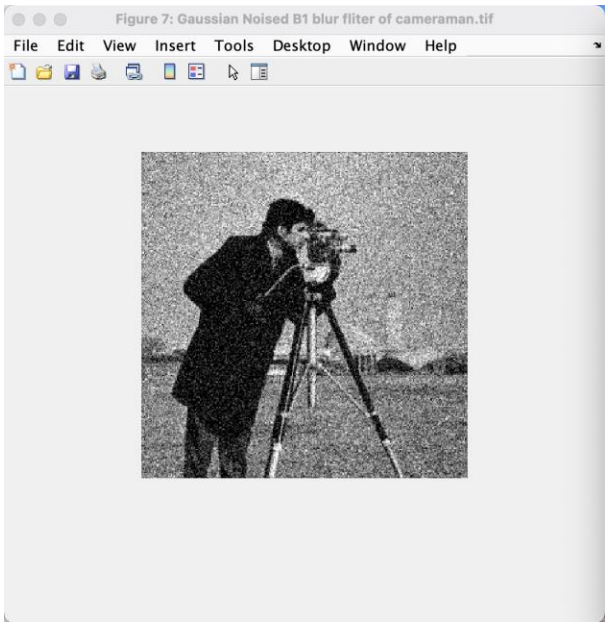

Figure 16. B1 (Gaussian Blur) + Gaussian White Noise applied to the cameraman

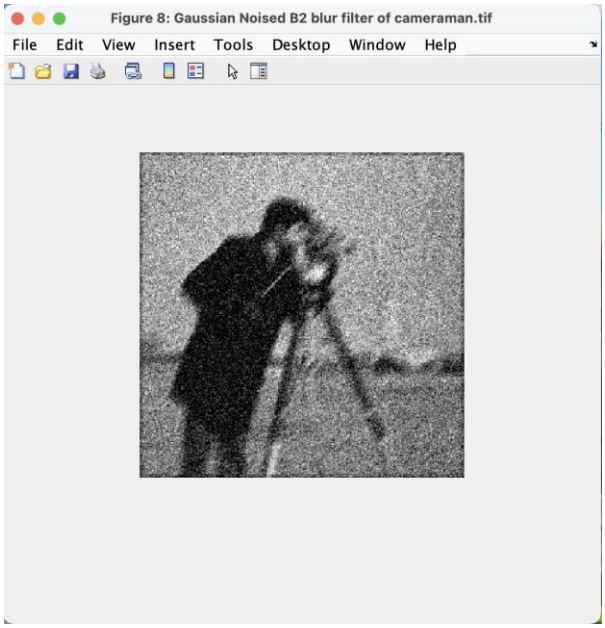

Figure 17. B2 (Motion Blur) + Gaussian White Noise applied to the cameraman

```
% Adding Gaussian white noise to Blurred image
Noised_I_B1 = imnoise(I_B1, "gaussian", 0, 0.02);
Noised_I_B2 = imnoise(I_B2, "gaussian", 0, 0.02);
figure(Name="Gaussian Noised B1 blur fliter of cameraman.tif");
imshow(Noised_I_B1);
figure(Name="Gaussian Noised B2 blur filter of cameraman.tif");
imshow(Noised_I_B2);
```

Figure 18. MATLAB code to apply Gaussian white noise with variance = 0.02 to blurred cameraman

(5)

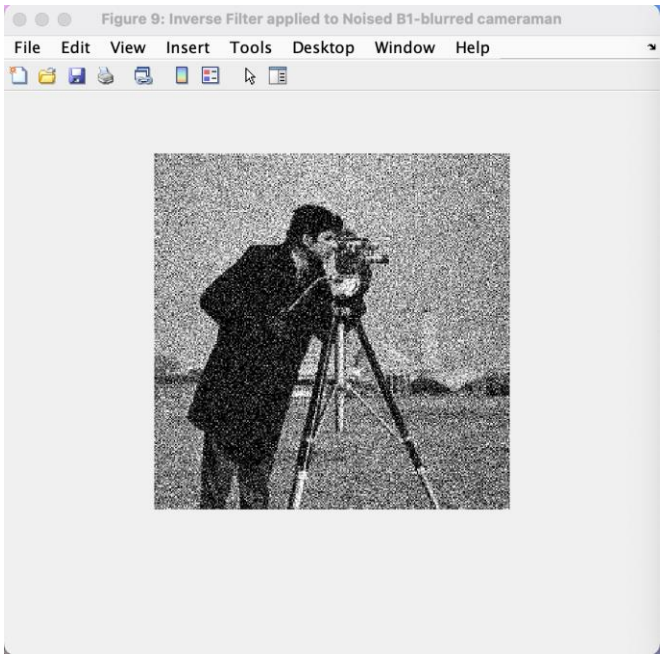

Figure 19. Inverse Filter applied to recover Noised B1 (Gaussian Blur) Blurred cameraman

Figure 20. Weiner Filter applied to recover Noised B1 (Gaussian Blur) Blurred cameraman

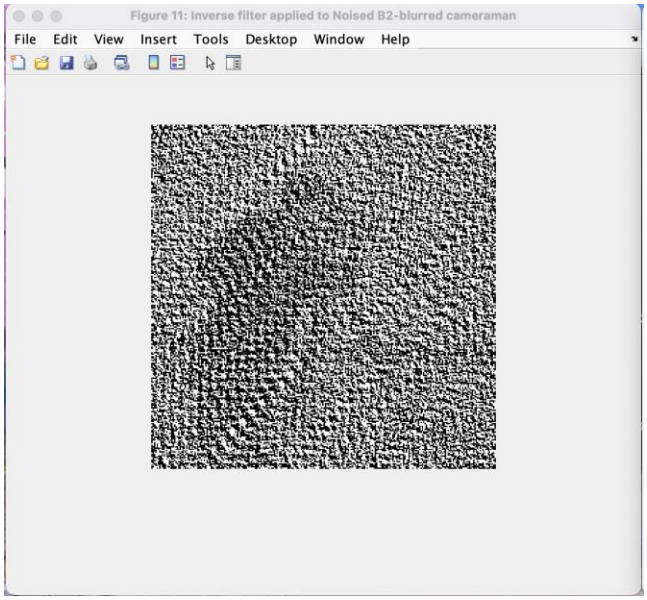

Figure 21. Inverse Filter applied to recover Noised B2 (Motion Blur) Blurred cameraman

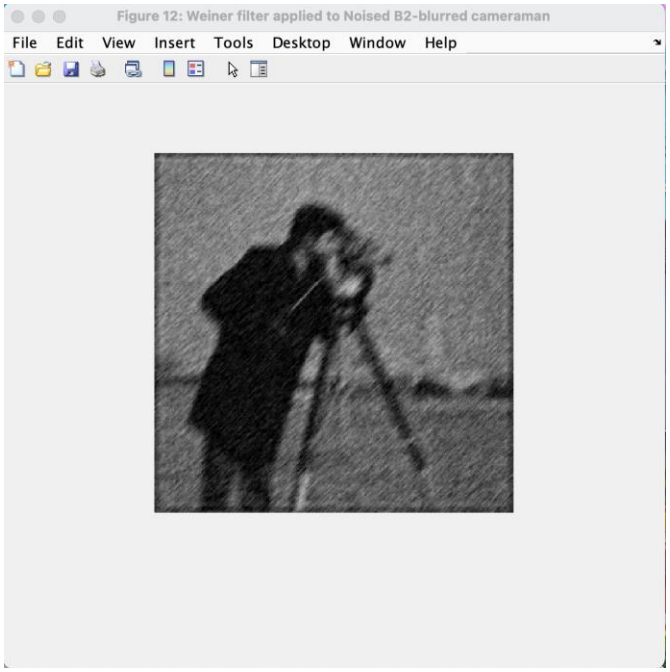

Figure 22. Weiner Filter applied to recover Noised B2 (Motion Blur) Blurred cameraman

From Figures 19 to 22, we can see that the weiner filter does a better job than an inverse filter. For the inverse filter, we can see that it did recover the B1 (Gaussian Blurred) image but the inverse filter does a worse job on the B2 (Motion Blurr) image. Weiner Filter does recover the image from both B1 (Gaussian Blur) and B2 (Motion Blur) images. However, the B2 (Motion Blur) is not recovered as well as the B1 (Gaussian Blur). B2 is slightly harder to recover for those two filters.

```
% Apply inverse filter and weiner filter to B1 blurred image
% inverse filter of B1
Invfilter_I_B1 = deconvreg(Noised_I_B1, B1);
figure(Name="Inverse Filter applied to Noised B1-blurred cameraman"); imshow(Invfilter_I_B1);
% weiner filter of B1
nsr = gaussVar/var(I(:));
weiner_I_B1 = deconvwnr(Noised_I_B1,B1,nsr);
figure(Name="Weiner filter applied to Noised B1-blurred cameraman"); imshow(weiner_I_B1);
% inverse filter of B2
Invfilter_I_B2 = deconvreg(Noised_I_B2, B2);
figure(Name="Inverse filter applied to Noised B2-blurred cameraman"); imshow(Invfilter_I_B2);
% weiner filter of B2
weiner_I_B2 = deconvwnr(Noised_I_B2,B2,nsr);
figure(Name="Weiner filter applied to Noised B2-blurred cameraman"); imshow(weiner_I_B2);
```

Figure 23. MATLAB code to apply the inverse filter and weiner filter to the B1 and B2 cameraman