Image Restoration

ENGR652

Vy Bui

I. INTRODUCTION

Restoring an image corrupted by an LSI blur operator and with additive white Gaussian noise is the classic problem in image restoration. Here we study the use of the pseudo-inverse filter and two forms of the Wiener filter in contesting these degradation. In particular, we define an impulse invariant system to model simple uniform linear motion blur. We artificially degrade an image using our model and then we will restore the image. We quantitatively and subjectively compare various image restoration methods and we investigate the performance of the filters as a function of their free tuning parameters.

II. THEORETICAL STUDY

An image corrupted by an LSI blur operator and with additive noise can be expressed as:

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

where, $h(x,y)$: the degradation function is motion average impulse response. $f(x,y)$: the original image. $\eta(x,y)$: white Gaussian noise with standard deviation 2.

The equivalent frequency domain model is:

$$G(u,v) = H(u,v).F(u,v) + N(u,v)$$

We can estimate $f(x,y)$ by using the Pseudo Inverse System and Wiener filter, here I denote as $r(x,y)$.

$$\hat{f}(x,y) = g(x,y) * r(x,y)$$

The equivalent frequency domain model is:

$$\hat{F}(x,y) = G(x,y) * R(x,y)$$

A. Pseudo Inverse System

We begin by restoring the degraded image using a pseudo inverse filter.

$$R(u,v) = \begin{cases} \dfrac{1}{G(u,v)} & |G(u,v)| > Threshold \\ 0 & otherwise \end{cases}$$

B. Wiener filter

There is a critical drawback of inverse filter that it can't handle noise appropriately. Wiener comes into play since it can hadle both the degradation function and the statistical characteristics of noise into the restoration process. The objective of Wiener filter is to find an estimate $\hat{f}$ of the image f such that the mean square error between them is minimized.

$$\min[e^2] = \min[E\{(f-\hat{f})^2\}]$$

The two version are constant NSR and parametric of Wiener filter are used to image restoration process.

Constant NSR Wiener filter

Autocorrelation function: $r_{gg}(m,n) = E\{g(x,y)g(x-m,y-n)\}$

Cross correlation: $r_{fg}(m,n) = E\{f(x,y)g(x-m,y-n)\}$

PSD and cross PSD: $|G(u,v)|^2 = S_{gg}(u,v) = DSFT\{r_{gg}(m,n)\}$

$$F(u,v)G^*(u,v) = S_{fg}(u,v) = DSFT\{r_{fg}(m,n)\}$$

Random process through LSI system

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$
$$S_{gg}(u,v) = |H(u,v)|^2 S_{ff}(u,v) + S_{\eta\eta}(u,v)$$
$$S_{fg}(u,v) = H(u,v)^* S_{ff}(u,v)$$

Since we assume noise is independent from signal so the cross correlation is zero.

For LSI Degradation and Noise:

$$H^w(u,v) = \frac{S_{fg}(u,v)}{S_{gg}(u,v)} = \frac{H(u,v)^* S_{ff}(u,v)}{|H(u,v)|^2 S_{ff}(u,v) + S_{\eta\eta}(u,v)} = \frac{H(u,v)^*}{|H(u,v)|^2 + NSR}$$

Parametric Wiener filter

$$H^w(\omega_1,\omega_2) = \frac{S_{fg}(\omega_1,\omega_2)}{S_{gg}(\omega_1,\omega_2)} = \frac{H(\omega_1,\omega_2)^* S_{ff}(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 S_{ff}(\omega_1,\omega_2) + S_{\eta\eta}(\omega_1,\omega_2)}$$
$$= \frac{H(\omega_1,\omega_2)^*}{|H(\omega_1,\omega_2)|^2 + NSR(\omega_1,\omega_2)}$$
$$NSR(\omega_1,\omega_2) = \frac{\gamma}{S_{ff}(\omega_1,\omega_2)} \; where \; S_{ff}(\omega_1,\omega_2) = \frac{1}{|P(\omega_1,\omega_2)|^2}$$

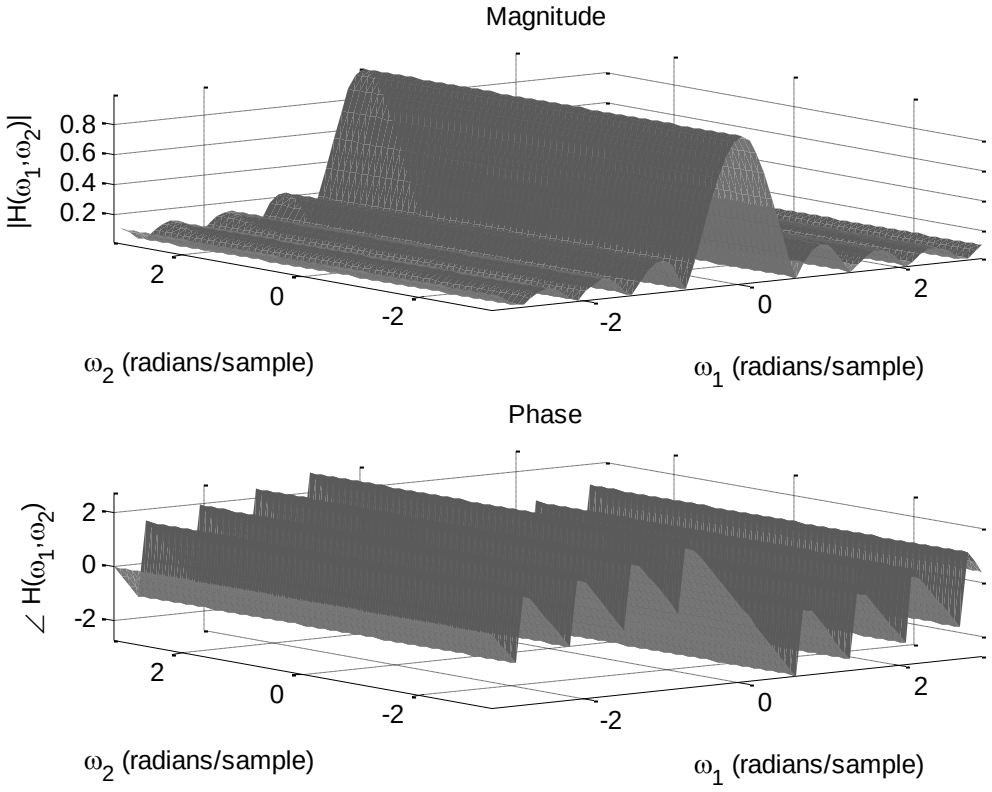
$P(\omega_1,\omega_2)$ is the DSFT of a high pass filter.

$$P(\omega_1,\omega_2) = DSFT\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

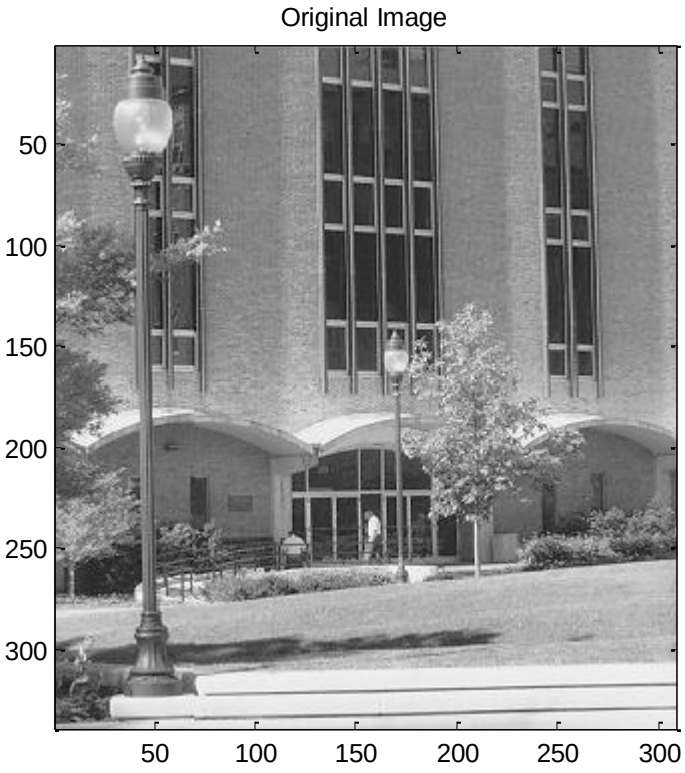
This means we assume the input image is a low frequency image (As most of the images are). Thus NSR increases with frequency.

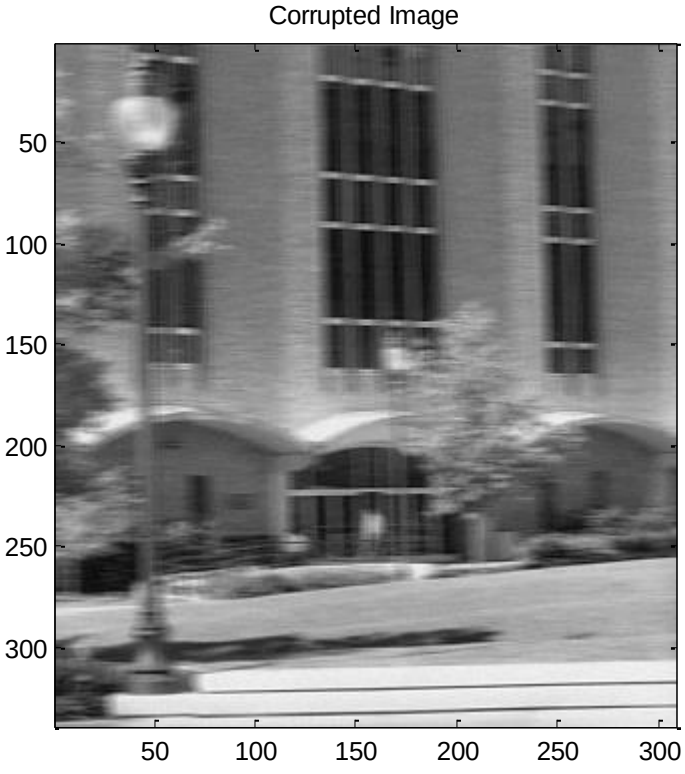
III. EXPERIMENTAL RESULTS

Show the magnitude frequency response of 9x1 MA filter

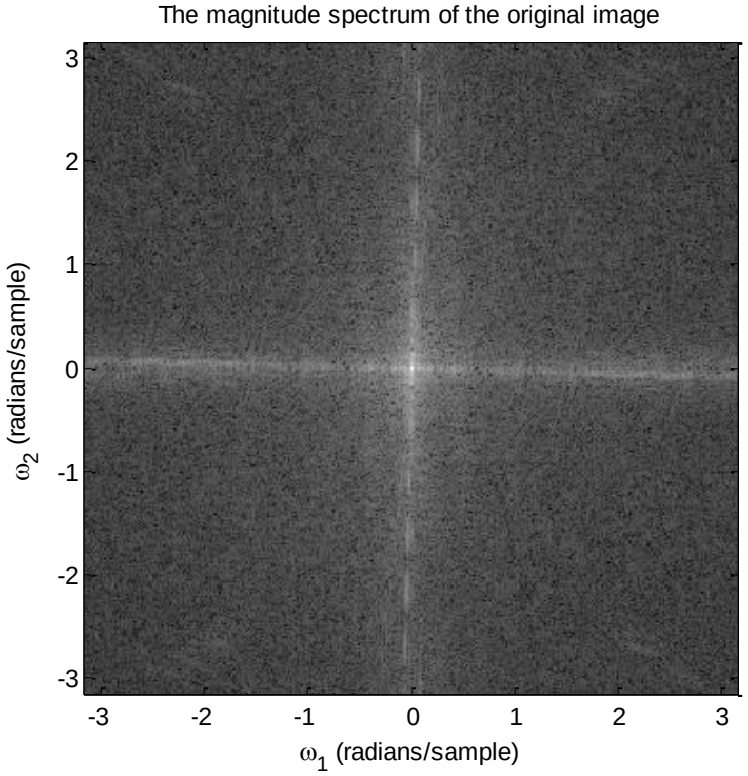


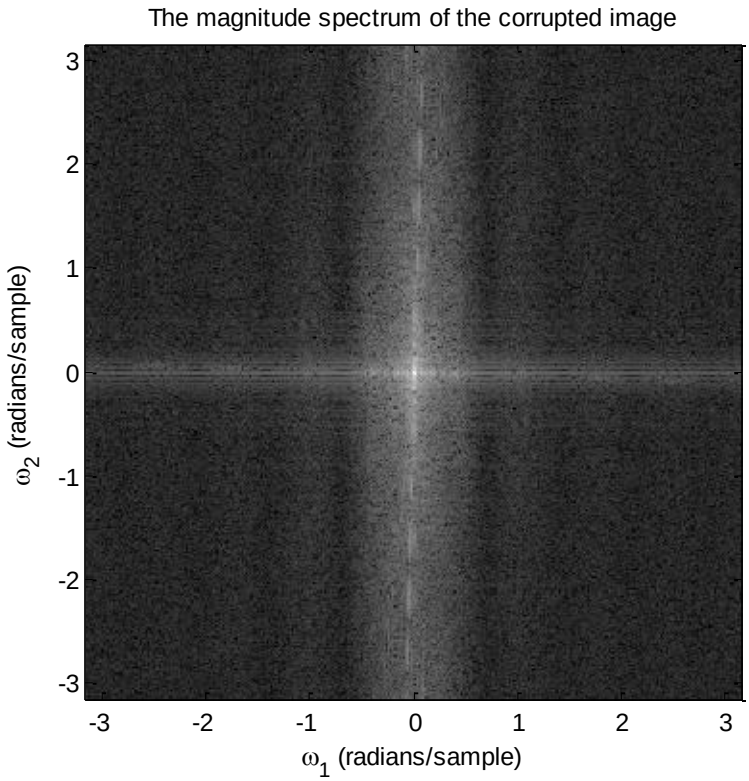
Show the original and corrupted images



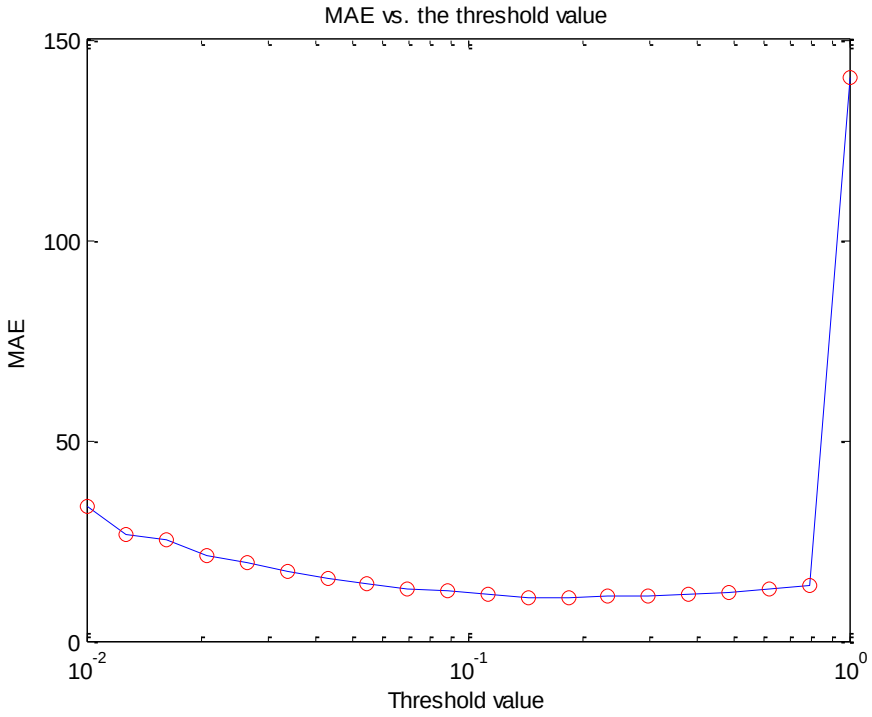


Show the magnitude spectrum of original and corrupted images. We can see 8 dark vertical stripes in the corrupted image since in this project we use 9x1 MA filter.

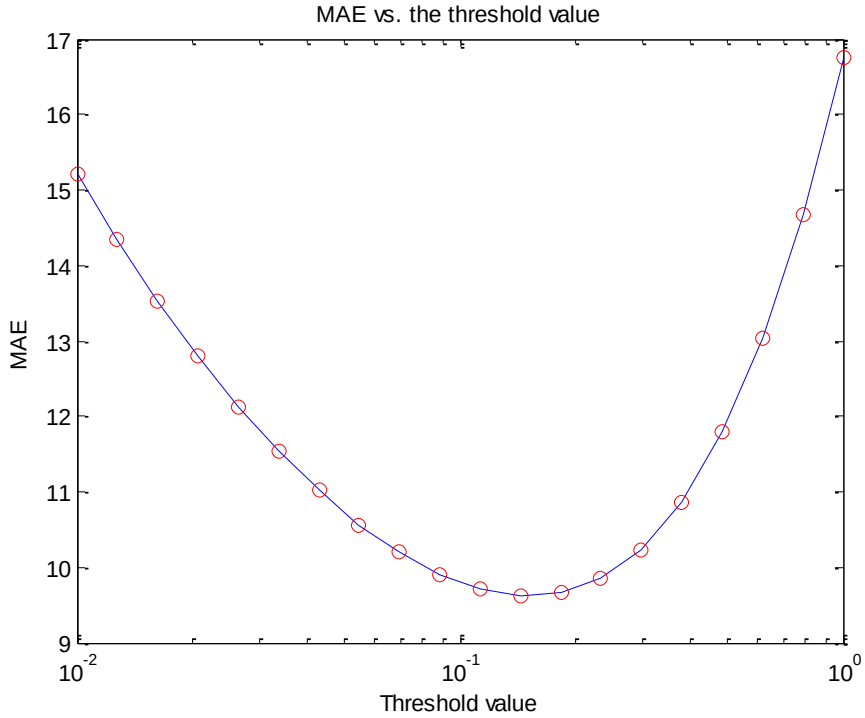




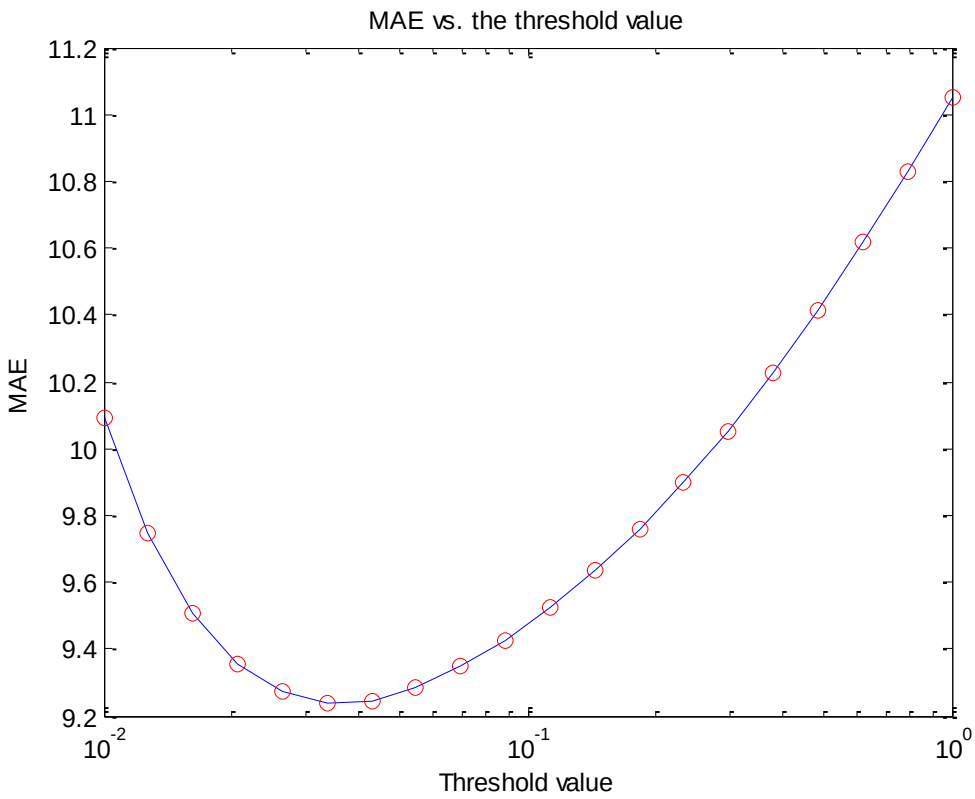
Semilog plot of MAE vs. pseudo-inverse filter threshold. The last MAE is high compared to others because all samples of it are less than the threshold.



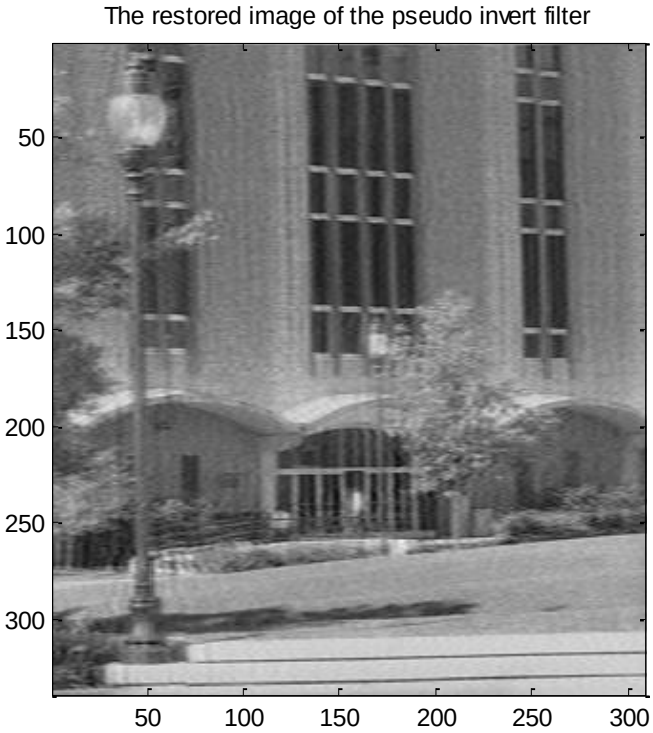
Semilog plot of MAE vs. constant NSR for the Wiener filter.



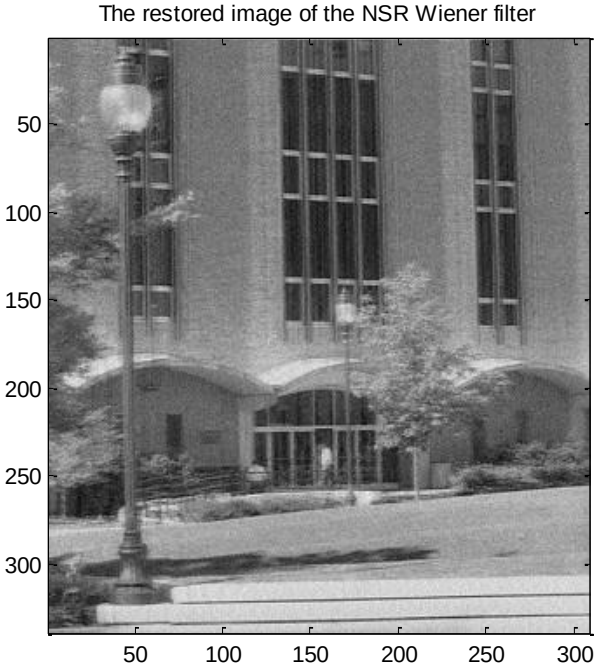
Semilog plot of MAE vs. gamma for the parametric Wiener filter.



Show the restored image using the best (in an MAE sense) of the pseudo-inverse filters. The minimum MAE will give the best result.



Show the restored image using the best(in an MAE sense) of the constant NSR Wiener filters. The minimum MAE will give the best result.



Show the restored image using the best(in an MAE sense) of the parametric Wiener filters. The minimum MAE will give the best result.

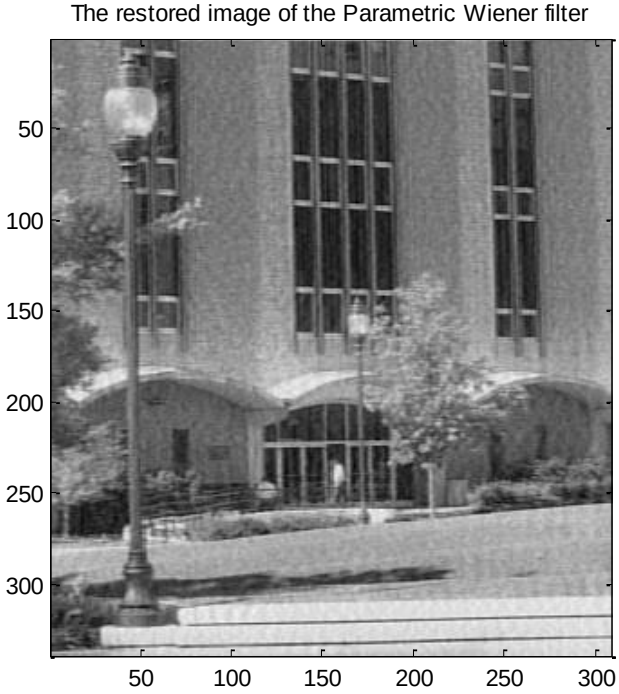
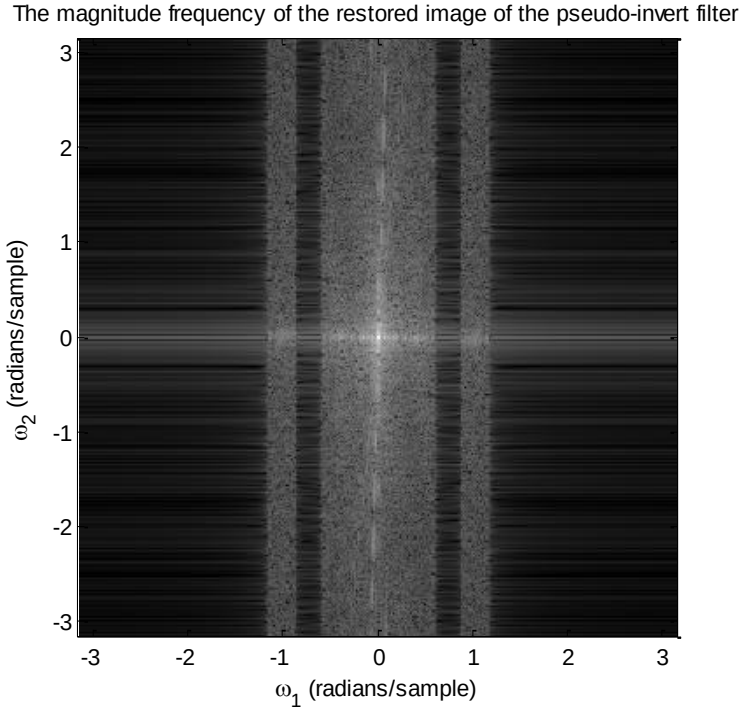


Table of the minimum MAE's for the three filters along with the MAE for the unfiltered image.

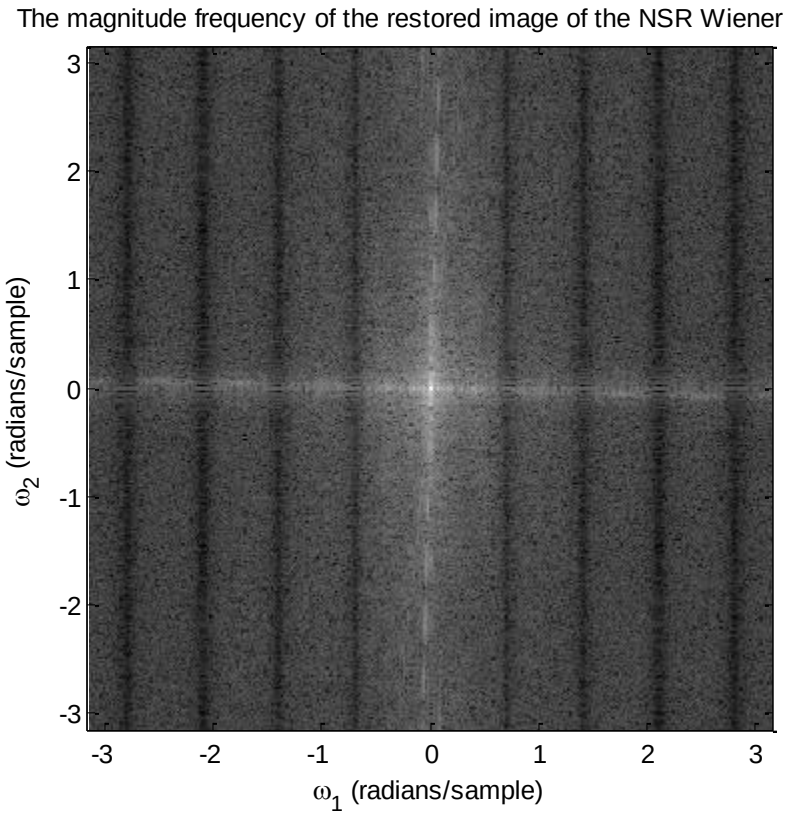
Minimum MAE of pseudo inverse filter threshold	10.8578
Minimum MAE of constant NSR for the Wiener filter	9.6061
Minimum MAE of the parametric Wiener filter	9.2153
MAE of the unfiltered image	12.0595

The parametric Wiener filter gives the best result since it can restore more information of the original image. This will be explained in the next part.

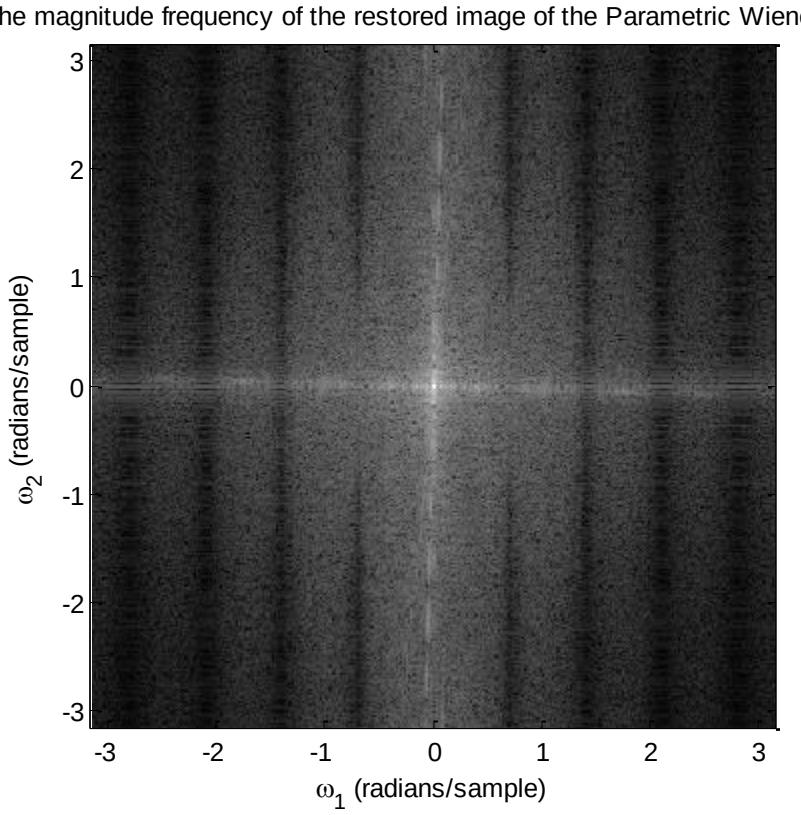
The magnitude frequency responses of the pseudo-inverse filters. This PI filter ignore high frequency responses.



The magnitude frequency responses of the constant NSR Wiener filters. Unlike PI filter, NSR filter can restore the high frequencies at the dark stripes.



The magnitude frequency responses of the parametric Wiener filters. The parametric Wiener filter can restore the low frequency.



IV. CONCLUSION

In this report, I study and implement the PI and Wiener filter in Matlab. The result shows that the parametric Wiener filter gives the better result compared to PI filter and NSR Wiener filter. The theory and the experiences are consistent since Wiener filter gives less MAE compared to PI filter.

V. SOURCE CODE

main.m

```
clc
clear all
close all
% Load the kett.mat image
load kett
% Generate a 9x1 (horizontal) moving avg impulse response
h = fspecial('motion'); %default is 9
% Convole this with the kett image
[sy,sx] = size(kett);
original_img=kett(1:sy-1,1:sx);
sy=sy-1;
motion_img = conv2(softpad(original_img,0,4,0,4),h,'valid');
% Add Gaussian noise with std = 2
motion_noise_img = motion_img + 2*randn(sy,sx);
% Show the magnitude and freq. response of the 9x1 MA filter
dsftl(h);
% Show the original and corrupted image
figure, imagesc(original_img), colormap(gray(256)), title('Original Image'),
axis image;
figure, imagesc(motion_noise_img), colormap(gray(256)), title('Corrupted
Image'), axis image;
% Show the magnitude spectrum of original and corrupted images
imspecxy(original_img,100)
title('The magnitude spectrum of the original image');
imspecxy(motion_noise_img,100)
title('The magnitude spectrum of the corrupted image');
MAE_unfiltered_img = mean(mean(abs(motion_noise_img-original_img)));

%% Pseudo-inverse
Threshold = logspace(-2,0,20);
i = 1;
while i <= 20,
    % Set the parameter border = 100 in pseudoinv2d.m
    [out,HI] = pseudoinv2d(motion_noise_img,h,Threshold(i),100);
    % Compute MAE between each and the original
    mean_absolute_error(i) = mean(mean(abs(out-original_img)));
    i = i + 1;
end
% Showing MAE vs. the threshold value
figure
semilogx(Threshold,mean_absolute_error, '-o', 'MarkerEdgeColor','r');
title('MAE vs. the threshold value');
xlabel('Threshold value');
ylabel('MAE');
% Show the restored image using the best MAE
[min_MAE_value,min_MAE_index] = min(mean_absolute_error);
[out,HI] = pseudoinv2d(motion_noise_img, h, Threshold(min_MAE_index), 100);
figure, imagesc(out), colormap(gray(256)), title('The restored image of the
pseudo invert filter'), axis image;
MAE_Pseudo_inverse = min_MAE_value;
% Show the magnitude spectrum
imspecxy(out,100)
```

```
title('The magnitude frequency of the restored image of the pseudo-invert
filter');

%% NSR_Wiener_filter
NSR = logspace(-3,-1,20);
i = 1;
while i <= 20,
    % Set the parameter border = 100 in NSR_Wiener_filter.m
    [out,HI] = NSR_Wiener_filter(motion_noise_img,h,100,NSR(i));
    % Compute MAE between each and the original
    mean_absolute_error(i) = mean(mean(abs(out-original_img)));
    i = i + 1;
end
% Showing MAE vs. the threshold value
figure
semilogx(Threshold,mean_absolute_error, '-o', 'MarkerEdgeColor','r');
title('MAE vs. the threshold value');
xlabel('Threshold value');
ylabel('MAE');
% Show the restored image using the best MAE
[min_MAE_value,min_MAE_index] = min(mean_absolute_error);
[out,HI] = NSR_Wiener_filter(motion_noise_img, h, 100, NSR(min_MAE_index));
figure, imagesc(out), colormap(gray(256)), title('The restored image of the
NSR Wiener filter'), axis image;
MAE_NSR_Wiener_filter = min_MAE_value;
% Show the magnitude spectrum
imspecxy(out,100)
title('The magnitude frequency of the restored image of the NSR Wiener
filter');

%% Parametric_Wiener_filter
P = [0 -1 0; -1 4 -1; 0 -1 0];
gamma = logspace(-3,-1,20);
i = 1;
while i <= 20,
    % Set the parameter border = 100
    [out,HI] = Parametric_Wiener_filter(motion_noise_img, h, 100, gamma(i),
sx, sy, P);
    mean_absolute_error(i) = mean(mean(abs(out-original_img)));
    i = i + 1;
end
% Showing MAE vs. the threshold value
figure
semilogx(Threshold,mean_absolute_error, '-o', 'MarkerEdgeColor','r');
title('MAE vs. the threshold value');
xlabel('Threshold value');
ylabel('MAE');
% Show the restored image using the best MAE
[min_MAE_value,min_MAE_index] = min(mean_absolute_error);
[out,HI] = Parametric_Wiener_filter(motion_noise_img, h, 100,
gamma(min_MAE_index), sx, sy, P);
figure, imagesc(out), colormap(gray(256)), title('The restored image of the
Parametric Wiener filter'), axis image;
MAE_Parametric_Wiener_filter = min_MAE_value;
% Show the magnitude spectrum
imspecxy(out,100)
```

```
title('The magnitude frequency of the restored image of the Parametric Wiener
filter');

%% Table of the minimum MAE's for the three filters along with the MAE for
the unfiltered image
MAE =
[MAE_unfiltered_img;MAE_Pseudo_inverse;MAE_NSR_Wiener_filter;MAE_Parametric_W
iener_filter]
```

NSR_Wiener_filter.m

```
function [out,HI] = NSR_Wiener_filter(in,psf,border,NSR)
%
%   Does 2D pseudo-inverse with NSR DFT Filtering
%
%   out    - filtered signal
%   HI_NSR - filter DFT samples
%   in     - input signal
%   psf    - FIR impulse response of degradation process
%   thresh    - threshold (~ .001 - .1)
%   border - border padding size to minimize ringing artifacts (~20)
%   NSR    - constant noise tosignal ration
% Author: Dr. Russell Hardie
% UD 3/25/99

% Pad image and get new size
in=softpad(in,border,border,border,border);
[L1,L2]=size(in);

% Create filter DFT
%disp('Generate Pseudo-Inverse Filter')
H=fft2(psf,L1,L2);

% Do thresholding
HI=conj(H)./(abs(H.^2)+NSR);

% Filter and Compute inverse DFT
% disp('Perform Filtering')

%flops(0);
X=fft2(in,L1,L2);
out=real(ifft2(HI.*X));

% circularly shift output to compensate for the
% PSF being defined in the 1st quadrant (circ shifted
% with respect to 0,0.
[psfy,psfx]=size(psf);
out=mycircshift(out,-round((psfx-1)/2),-round((psfy-1)/2));

% cut out original image size
out=out(1+border:L1-(border),1+border:L2-(border));
end
```

Parametric_Wiener_filter.m

```
function [out,HI] = Parametric_Wiener_filter(in,psf,border,gamma, sx, sy, P)
%
%   Does 2D pseudo-inverse with NSR DFT Filtering
%
%   out    - filtered signal
%   HI - filter DFT samples
%   in     - input signal
%   psf    - FIR impulse response of degradation process
%   thresh    - threshold (~ .001 - .1)
%   border - border padding size to minimize ringing artifacts (~20)
%   P      - DSFT of high pass filter
%   gamma  - noise variance
% Author: Dr. Russell Hardie
% UD 3/25/99

% Pad image and get new size
in=softpad(in,border,border,border,border);
[L1,L2]=size(in);

% Create filter DFT
%disp('Generate Pseudo-Inverse Filter')
H=fft2(psf,L1,L2);

% Do thresholding
dsft_P = fft2(P,sy + 2 * border,sx + 2 * border); % Equation 4
magnitude_P_squared = abs(dsft_P.^2); % Equation 3
NSR = gamma*magnitude_P_squared;
HI=conj(H)./(abs(H.^2)+NSR);

% Filter and Compute inverse DFT
%disp('Perform Filtering')

%flops(0);
X=fft2(in,L1,L2);
out=real(ifft2(HI.*X));

% circularly shift output to compensate for the
% PSF being defined in the 1st quadrant (circ shifted
% with respect to 0,0.
[psfy,psfx]=size(psf);
out=mycircshift(out,-round((psfx-1)/2),-round((psfy-1)/2));

% cut out original image size
out=out(1+border:L1-(border),1+border:L2-(border));
end
```