**Image Restoration**

**ENGR652**

**Vy Bui**

1. **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.

1. **THEORETICAL STUDY**

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

where, : the degradation function is motion average impulse response. : the original image. : white Gaussian noise with standard deviation 2.

The equivalent frequency domain model is:

We can estimate by using the Pseudo Inverse System and Wiener filter, here I denote as .

The equivalent frequency domain model is:

1. **Pseudo Inverse System**

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

1. **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 of the image such that the mean square error between them is minimized.

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

**Constant NSR Wiener filter**

Autocorrelation function:

Cross correlation:

PSD and cross PSD:

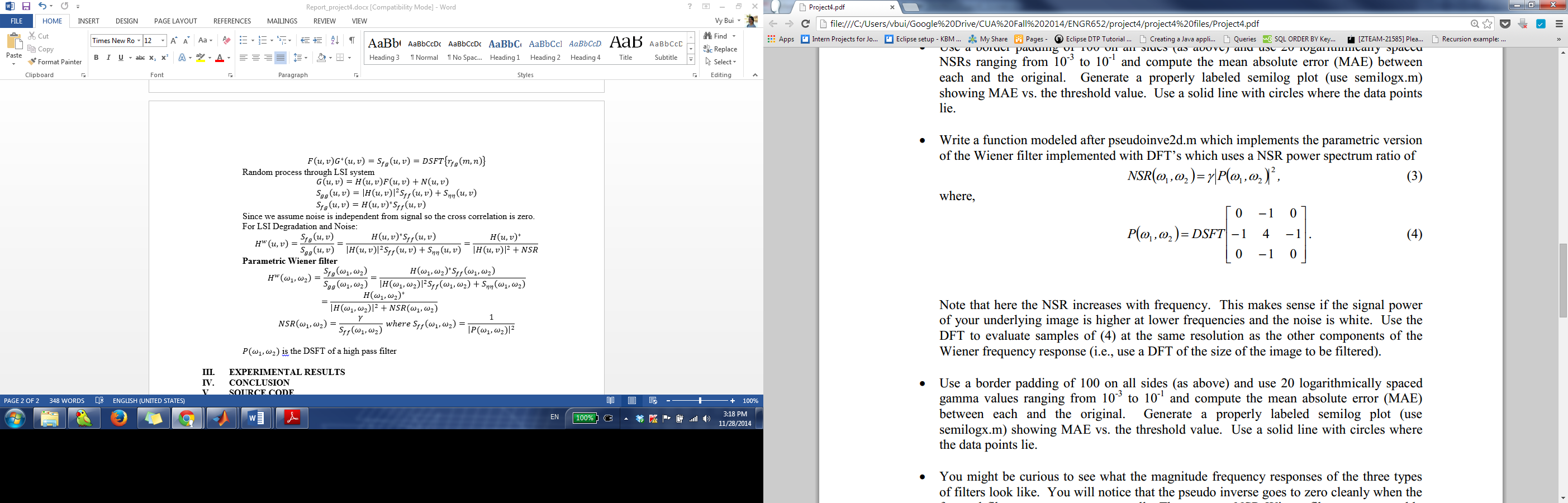
Random process through LSI system

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

For LSI Degradation and Noise:

**Parametric Wiener filter**

is the DSFT of a high pass filter.



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

1. **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.



1. **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.

1. **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\_Wiener\_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