**PROJECT-11**

**INVERSE & WIENER**

**FILTERING**

EE5356 Digital Image Processing

Dr. K. R. Rao

Submitted By:

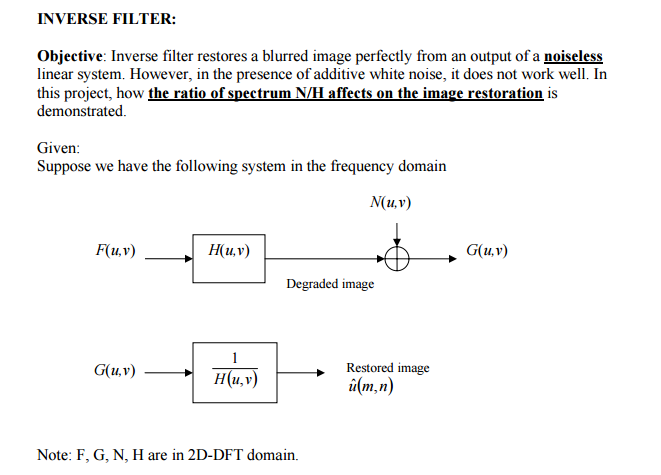
Ashutosh Desai

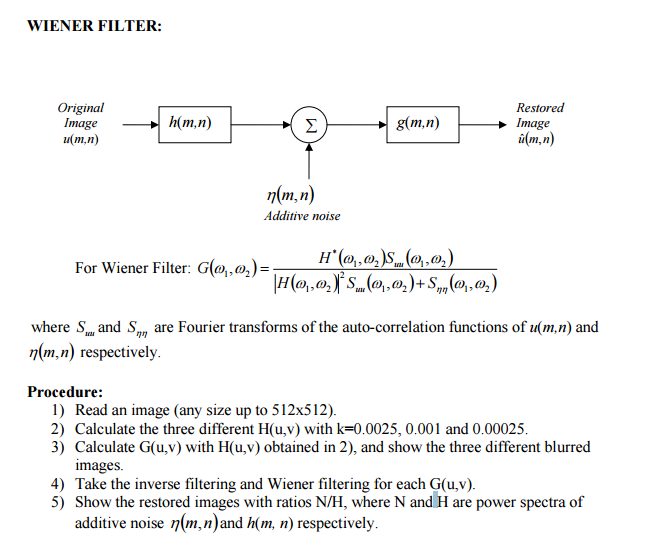
UTA ID:1001388602

Email: - ashutoshrajesh.desai@mavs.uta.edu

SPRING-2018

QUESTION :

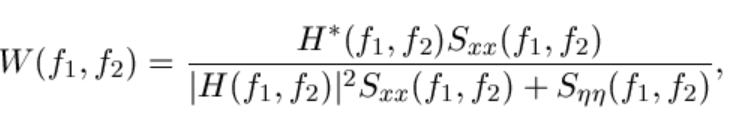




**WEINER FILTERING**

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.

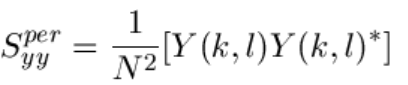
The Wiener filtering is optimal in terms of the mean square error. In other words, it minimizes the overall mean square error in the process of inverse filtering and noise smoothing. The Wiener filtering is a linear estimation of the original image. The approach is based on a stochastic framework. The orthogonality principle implies that the Wiener filter in Fourier domain can be expressed as follows:



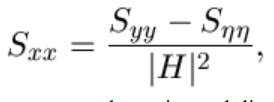
where http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/images/sxxee1.gif are respectively power spectra of the original image and the additive noise, and http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/images/hff.gif is the blurring filter. It is easy to see that the Wiener filter has two separate part, an inverse filtering part and a noise smoothing part. It not only performs the deconvolution by inverse filtering (highpass filtering) but also removes the noise with a compression operation (lowpass filtering).

**Implementation**

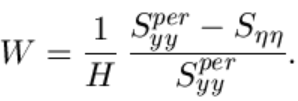
To implement the Wiener filter in practice we have to estimate the power spectra of the original image and the additive noise. For white additive noise the power spectrum is equal to the variance of the noise. To estimate the power spectrum of the original image many methods can be used. A direct estimate is the periodogram estimate of the power spectrum computed from the observat



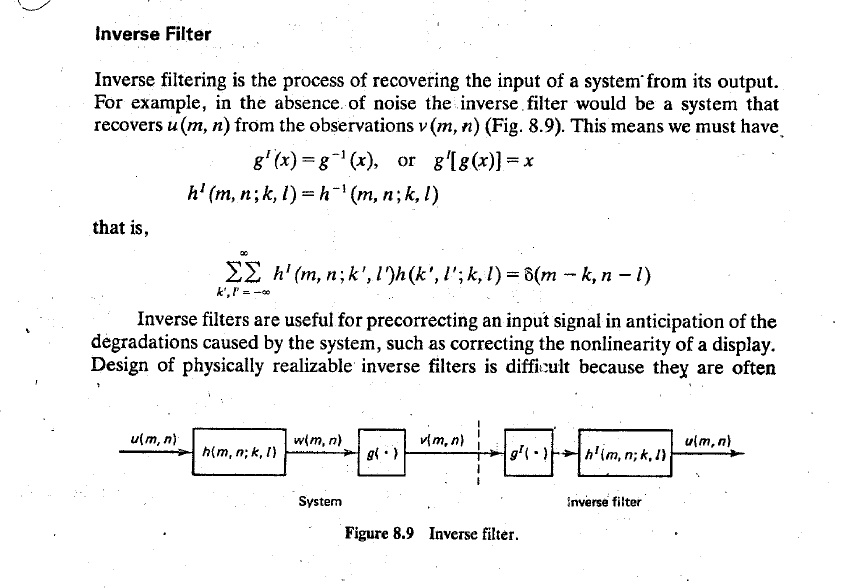
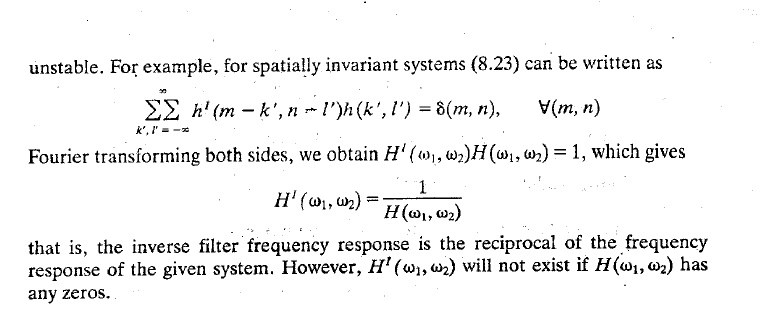
where **Y(k,l)** is the DFT of the observation. The advantage of the estimate is that it can be implemented very easily without worrying about the singularity of the inverse filtering. Another estimate which leads to a cascade implementation of the inverse filtering and the noise smoothing is



which is a straightforward result of the fact: http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/images/syyest.gif The power spectrum http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/images/syy.gif can be estimated directly from the observation using the periodogram estimate. This estimate results in a cascade implementation of inverse filtering and noise smoothing:



The disadvantage of this implementation is that when the inverse filter is singular, we have to use the generalized inverse filtering. People also suggest the power spectrum of the original image can be estimated based on a model such as the http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/images/falpha.gif model.



**MATLAB SCRIPT:-**

clc;

close all ;

clear all ;

Img = double(imread('D:\STUDY\DIP\Test img\lena512.bmp'));

[row col] = size(Img);

inv\_F1 = fft2(Img);

[u,v]=meshgrid(1:row,1:col);

%Blurring functions.

BF\_1 = exp(-0.0025\*(u.^2+v.^2).^(5/6));

BF\_2 = exp(-0.001\*(u.^2+v.^2).^(5/6));

BF\_3 = exp(-0.00025\*(u.^2+v.^2).^(5/6));

%Noise Function

Noise = randn(row,col);

Noise\_1 = fft2(Noise);

%Blurred images.

Blur\_1 = inv\_F1.\*BF\_1;

Blur\_2 = inv\_F1.\*BF\_2;

Blur\_3 = inv\_F1.\*BF\_3;

G\_1 = Blur\_1 + Noise\_1;

G\_2 = Blur\_2 + Noise\_1;

G\_3 = Blur\_3 + Noise\_1;

%Blurred and Noisy image

Img\_Noise1 = abs(ifft2(Blur\_1));

Img\_Noise2 = abs(ifft2(Blur\_2));

Img\_Noise3 = abs(ifft2(Blur\_3));

subplot(2,2,1);imshow(uint8(Img\_Noise1));title(' Blurred Image,K=0.0025')

subplot(2,2,2);imshow(uint8(Img\_Noise2));title(' Blurred Image,K=0.001')

subplot(2,2,3);imshow(uint8(Img\_Noise3));title(' Blurred Image,K=0.000025')

%Inverse filtering

Inv\_Filter\_1 = (1./BF\_1);

Inv\_Filter\_2 = (1./BF\_2);

Inv\_Filter\_3 = (1./BF\_3);

for a=1:row

for b=1:col

if Inv\_Filter\_1(a,b)>=100

Inv\_Filter\_1(a,b)=0;

end

if Inv\_Filter\_2(a,b)>=100

Inv\_Filter\_2(a,b)=0;

end

if Inv\_Filter\_3(a,b)>=100

Inv\_Filter\_3(a,b)=0;

end

end

end

Inv\_g1 = abs( ifft2(G\_1.\*Inv\_Filter\_1) );

Inv\_g2 = abs( ifft2(G\_2.\*Inv\_Filter\_2) );

Inv\_g3 = abs( ifft2(G\_3.\*Inv\_Filter\_3) );

figure;

subplot(2,3,1);imshow(uint8(Inv\_g1));title('Inverse Filtered ,K=0.0025')

subplot(2,3,2);imshow(uint8(Inv\_g2));title('Inverse Filtered ,K=0.001')

subplot(2,3,3);imshow(uint8(Inv\_g3));title('Inverse Filtered ,K=0.000025')

J1 = abs( ifft2(G\_1));

J2 = abs( ifft2(G\_2));

J3 = abs( ifft2(G\_3));

% Weiner Filtering

WF\_J1 = wiener2(J1,[3 3],Noise);

WF\_J2 = wiener2(J2,[3 3],Noise);

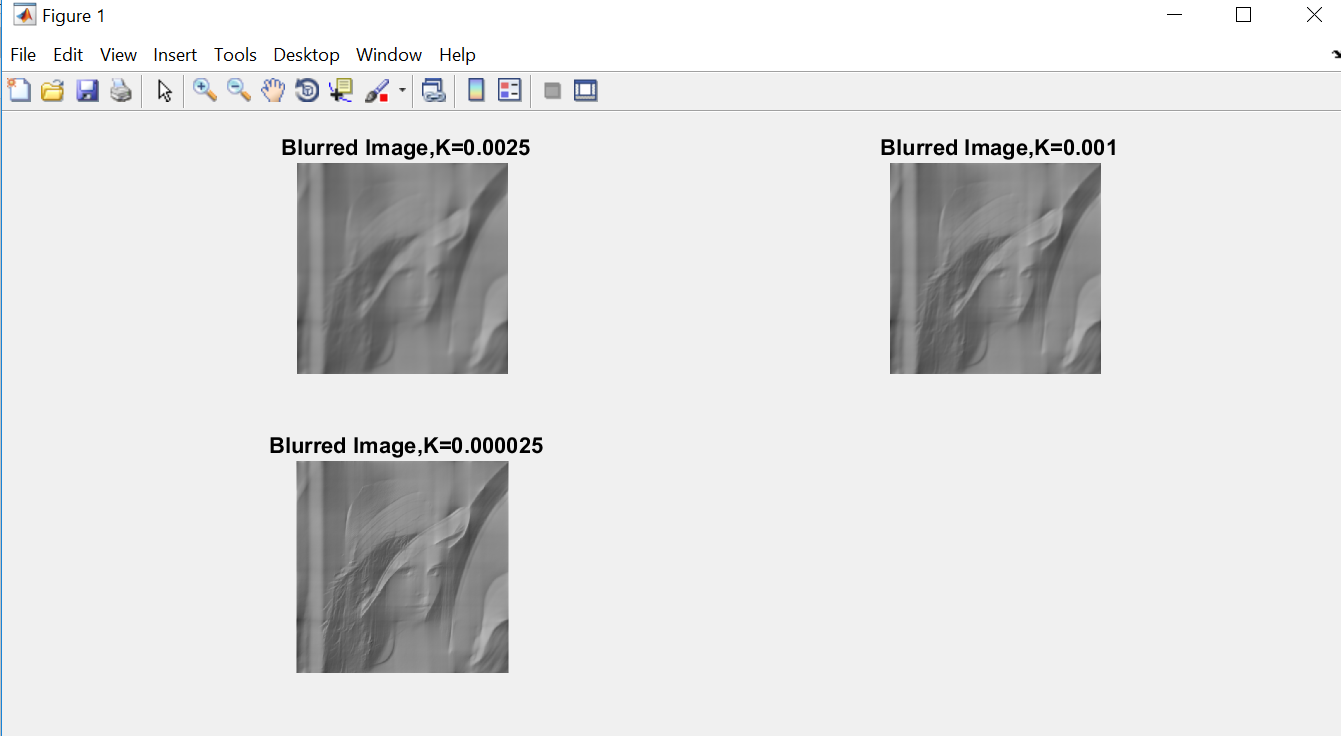
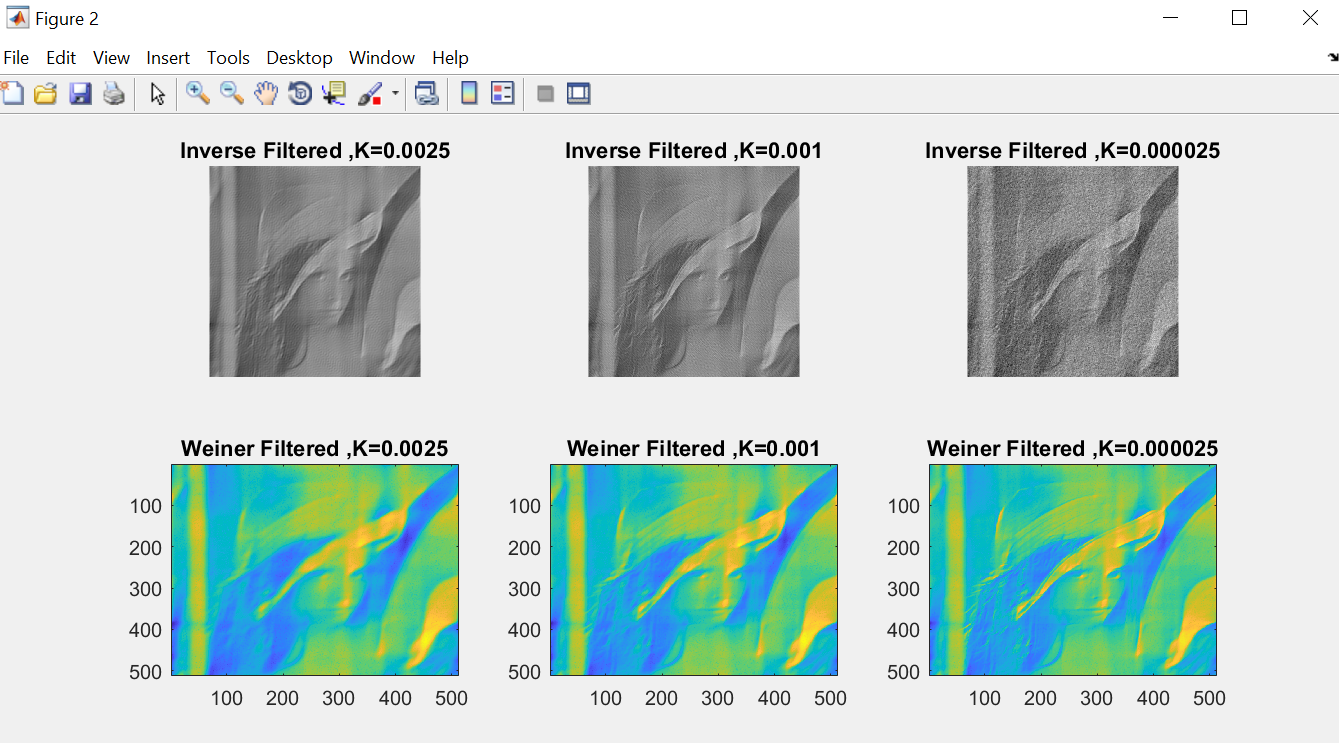
WF\_J3 = wiener2(J3,[3 3],Noise);

subplot(2,3,4);imagesc(uint8(WF\_J1));title('Weiner Filtered ,K=0.0025')

subplot(2,3,5);imagesc(uint8(WF\_J2));title('Weiner Filtered ,K=0.001')

subplot(2,3,6);imagesc(uint8(WF\_J3));title('Weiner Filtered ,K=0.000025')

**OUTPUT:**



**CONCLUSION:**

From this experiment, we studied about operation of Inverse Filter and Weiner Filter. We observed the change in the output image for each filter. So we can derive following statements as concluding statement:

* Inverse filter restores a blurred image perfectly from an output of a noiseless linear system.
* The most important technique for removal of blur in images due to linear motion or unfocussed optics is the Wiener filter.
* Both Inverse Filter and Weiner Filter remove the blur in images either due to noise or image captured in motion.