**COMPUTER PROJECT 2**

**DIGITAL IMAGE PROCESSING**

**Nidumolu Koundinya (652271268)**

**Kamireddy Lakshmi Yasaswi (651771619)**

**OBJECTIVE:**

The objective of the project is to design a MATLAB code and simulate the code along with the SNR produced after performing the following restoration techniques on a 256 x 256 image.

1. Inverse Filter
2. Least Square Estimation
3. Constrained Least Square Estimation

a. Constrained Least Square Pseudo Inverse

b. Laplacian Constrained Least Square

c. Parametric Wiener Filter

1. Minimum Mean Square Estimation Wiener Filter
2. Maximum Entropy Pseudo Inverse

**METHODOLOGY:**

**Degradation:**

g

f

H

n

**Restoration:**

H-1

g

f ^

**Terminology:**

F^(K1,K2) -Restored image pixel at (K1,K2) in frequency domain

H(K1,K2) –Channel transfer function at ( K1,K2) in frequency domain

G(K1,K2)-Degraded image pixel at (K1,K2) in frequency domain

Sn(K1,K2)-Power spectral density of noise at (K1,K2) in frequency domain

Sf(K1,K2)-Power spectral density of image at (K1,K2) in frequency domain

1. **Inverse Filter:**

Inverse filter restores a blurred image perfectly from an output of a noiseless linear system. The simplest approach to restoration is direct inverse filtering. The equation is given as:

1. **Least Square Estimation:**

Least mean squares (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). Consider g=Hf+n and let g˜=HBTf˜ Let us assume H characterizes time invariant system then



1. **Constrained Least Square Estimation:**

The equation for constrained least square estimation is given by

Now depending on various estimations are possible.

1. **Constrained Least Square Pseudo Inverse:**

For this method Q is considered as Identity and the following equation is obtained

1. **Laplacian Constrained Least square :**

In this method Q is considered such that it can find the rate of rate of change. This can be denoted as a laplacian which is an operator for calculating the rate of rate of change.

Q matrix = .

1. **Parametric wiener filter:**

In this method Q is considered as where Rf covariance matrix of image f and Rn covariance matrix of noise .The equation for this method is as follows:

1. **Minimum Mean Square Estimation Wiener Filter:**

The Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant filtering an observed noisy process. The Wiener filter minimizes the mean square error between the estimated random process and the desired process. It removes the additive noise and inverts the blurring simultaneously. The Wiener filtering is optimal in terms of the mean square error. The equation is given by:

1. **Maximum Entropy Pseudo Inverse:**

In this we consider the image as a probability density function and we maximize the entropy of this function. We obtain the equation as:

**RESULTS AND CONCLUSION:**

In this project, we implemented the restoration of a linear degradation of 256x256 image in MATLAB. A few of the features of a good introductory image processing program have been seen. There are many more complex modifications you can make to the images. For example, you can apply a variety of filters to the image. The filters use mathematical algorithms to modify the image.

The SNR’S produced are:

1. Inverse Filter = 4.8486e+16
2. Least-Squares Filter = 1.9844e+5
3. Maximum Entropy Pseudo-Inverse Filter = 144.9312
4. Norm-Constrained Least-Squares Pseudo-Inverse Filter = 6.48e+4
5. Laplacian-Constrained Least-Squares Filter = 1.5184e+5
6. Covariance-Constrained Least-Squares Parametric Wiener Filter = 144.9191
7. Linear Minimum Mean-Square Error Wiener Filter = 0.9947
8. **Inverse Filter**



**SNR:**



1. **Least Square Estimation**



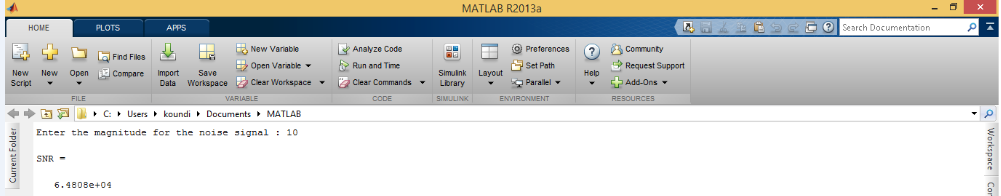
**SNR:**



1. **Constrained Least Square Estimation**
2. **Constrained Least Square Pseudo Inverse**



**SNR:**



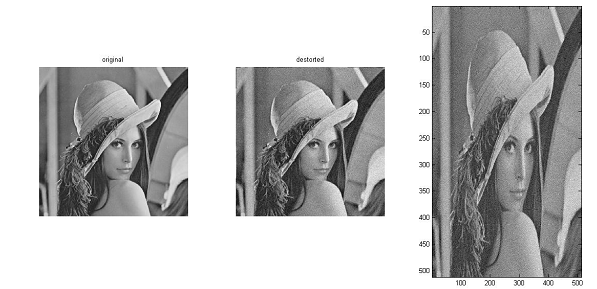
1. **Laplacian Constrained Least Square**



**SNR:**



1. **Parametric Wiener Filter**



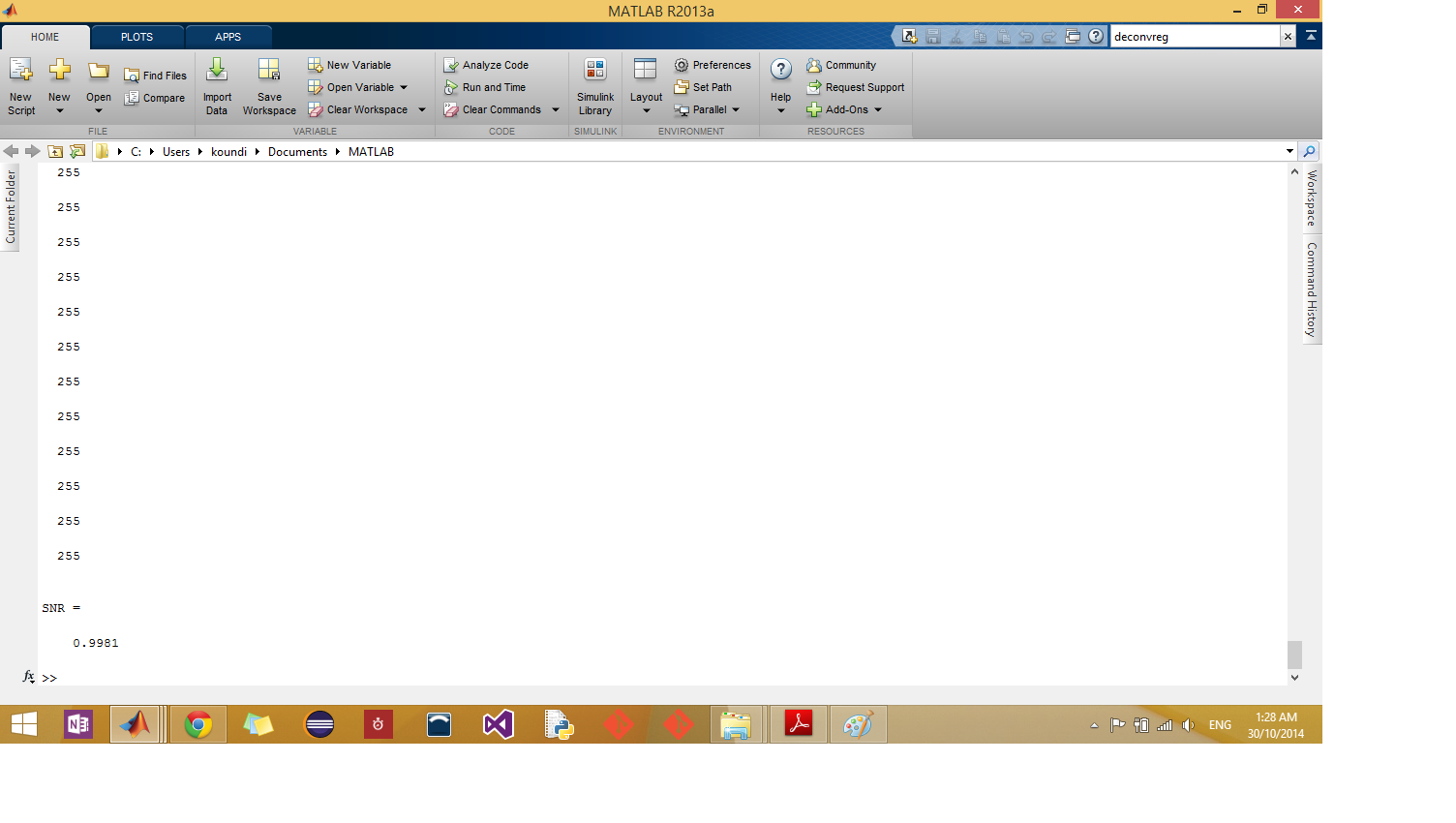
**SNR:**



1. **Minimum Mean Square Estimation Wiener Filter**



**SNR:**



1. **Maximum Entropy Pseudo Inverse**



**SNR:**



**APPENDIX**

**Inverse Filter**

clc;

close all;

clear all;

%input

img=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

img=rgb2gray(img);

F= fft2(img); %input image in freq domain

M=size(img,1); %rows of input image

N=size(img,2); %columns of the input image

%channel

h=ones(4,4)./8; %channel in spatial domain

H=fft2(h,M,N); %channel in freq domain

%Degradation

G= F.\*H ; %degraded image in spatial domain

g = (ifft2(G)); %degraded image in freq domain

g=mat2gray(g);

%Restoration

Hin=1./H;

Fcap = G.\*Hin;

fcap = abs(ifft2(Fcap)); %restored image in freq domain

fcap= mat2gray(fcap);

%SNR calculation

SNR = norm(F,2)/norm(Fcap-G,2)

%Plotting of Results

figure;

subplot(1,3,1)

imshow(img);

title('original');

subplot(1,3,2)

imshow(g);

title('degraded');

subplot(1,3,3);

imshow(fcap);

title('Restored');

**Least Square Estimation**

clc;

close all;

clear all;

%input

img=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

img=rgb2gray(img);

F= fft2(img); %input image in freq domain

M=size(img,1); %rows of input image

N=size(img,2); %columns of the input image

%channel

h=ones(M,N); %channel in spatial domain

H=fft2(h,M,N); %channel in freq domain

%Noise

n=randn(M,N); %Noise in spatial domain

Ns=fft2(n,M,N); %noise in freq domain

%Degradation

G= F.\*H +Ns; %degraded image in spatial domain

g = abs(ifft2(G)); %degraded image in freq domain

g=mat2gray(g);

%Restoration

Fcap = G./H;

fcap = abs(ifft2(Fcap)); %restored image in freq domain

fcap= mat2gray(fcap);

%SNR calculation

SNR = norm(Fcap,2)/norm(Ns,2)

%Plotting of Results

figure;

subplot(1,3,1)

imshow(img);

title('original');

subplot(1,3,2)

imshow(g);

title('degraded');

subplot(1,3,3);

imshow(fcap),title('Restored');

**Constrained Least Square Pseudo Inverse**

clc;

close all;

clear all;

%input

img=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

img=rgb2gray(img);

F= fft2(img); %input image in freq domain

M=size(img,1); %rows of input image

N=size(img,2); %columns of the input image

%channel

h = fspecial('gaussian', [3 3], 0.5); %channel in spatial domain

H= fft2(h,M,N); %channel in freq domain

%Noise

noise = input('Enter the magnitude for the noise signal : ');

n = fspecial('gaussian', [3 3], noise); %Noise in spatial domain

%n=noise\*rand(M,N);

Ns=fft2(n,M,N); %noise in freq domain

%Degradation

G= F.\*H +Ns; %degraded image in spatial domain

g = abs(ifft2(G)); %degraded image in freq domain

g=mat2gray(g);

%Restoration

lambda=1;

Fcap = zeros(M,N);

for k1=1:M

for k2=1:N

Fcap(k1,k2) = ( conj( H(k1,k2) ) / ( pow2( abs( H(k1,k2) ) )+ lambda^-1 ) )\* G(k1,k2); % restored image in spatial domain

end

end

fcap = abs(ifft2(Fcap)); %restored image in freq domain

fcap= mat2gray(fcap);

%SNR calculation

SNR = norm(Fcap,2)/norm(Ns,2)

%Plotting of Results

figure;

subplot(1,3,1)

imshow(img);

title('original');

subplot(1,3,2)

imshow(g);

title('degraded');

subplot(1,3,3);

imshow(fcap),title('Restored');

**Laplacian Constrained Least Square**

clc;

close all;

clear all;

%input

img=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

img=rgb2gray(img);

F= fft2(img); %input image in freq domain

M=size(img,1); %rows of input image

N=size(img,2); %columns of the input image

%channel

h = fspecial('gaussian', [3 3], 0.5); %channel in spatial domain

H= fft2(h,M,N); %channel in freq domain

%Noise

noise = input('Enter the magnitude for the noise signal : ');

n = fspecial('gaussian', [3 3], noise); %Noise in spatial domain

Ns=fft2(n,M,N); %noise in freq domain

%Degradation

G= F.\*H +Ns; %degraded image in spatial domain

g = abs(ifft2(G)); %degraded image in freq domain

g=mat2gray(g);

%Restoration

q=fspecial('laplacian');

Q=fft2(q,M,N);

lambda=1;

Fcap = zeros(M,N);

for k1=1:M

for k2=1:N

Fcap(k1,k2) = ( conj( H(k1,k2) ) / ( pow2( abs( H(k1,k2) ) )+ lambda^-1\*pow2( abs( Q(k1,k2) ) ) ) )\* G(k1,k2); % restored image in spatial domain

end

end

fcap = abs(ifft2(Fcap)); %restored image in freq domain

fcap= mat2gray(fcap);

%SNR calculation

SNR = norm(Fcap,2)/norm(Ns,2)

%Plotting of Results

figure;

subplot(1,3,1)

imshow(img);

title('original');

subplot(1,3,2)

imshow(g);

title('degraded');

subplot(1,3,3);

imshow(fcap),title('Restored');

**Parametric Wiener Filter**

clc;

close all;

clear all;

%input

img=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

noise = input('Enter the magnitude for the noise signal : ');

img=rgb2gray(img);

F= fft2(img); %input image in freq domain

M=size(img,1); %rows of input image

N=size(img,2); %columns of the input image

%channel

h = fspecial('gaussian', [3 3], 0.5); %channel in spatial domain

H= fft2(h,M,N); %channel in freq domain

%Noise

n=noise\*randn(M,N); %Noise in spatial domain

Ns=fft2(n,M,N); %noise in freq domain

%Degradation

G= F.\*H +Ns; %degraded image in spatial domain

g = abs(ifft2(G)); %degraded image in freq domain

%Restoration

q=fspecial('laplacian');

Q=fft2(q,M,N);

lamda=1;

Fcls=zeros(M,N);

for k1=1:M

for k2=1:N

Fcls(k1,k2)=(conj(H(k1,k2))\*G(k1,k2))/(abs(H(k1,k2)))^2+lamda^-1\*(abs(Q(k1,k2))^2); %restored image in freq domain

end;

end;

f=ifft2(Fcls);

%SNR calculation

SNR=norm(F,2)/norm(Ns,2)

%Plotting of Results

subplot(1,3,1)

imshow(img);

title('original')

subplot(1,3,2)

g=mat2gray(g);

imshow(g);

title('destorted')

subplot(1,3,3)

imagesc(f);

colormap(gray)

**Minimum Mean Square Estimation Wiener Filter**

clc;

clear all;

close all;

%input

RGB=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

org = rgb2gray(RGB);

%Degradation

degr=imnoise(org,'gaussian',0,0.005);

%In built Wiener Functions

rest10=wiener2(degr,[10 10]);

rest6=wiener2(degr,[6 6]);

p=double(0);

disp(p);

for x=1:1:256

disp(p);

for y=1:1:256

q=double((org(x,y)).^2)/((org(x,y)-rest6(x,y)).^2);

p=p+q;

end;

end;

%SNR calculation

SNR=norm(double(rest10))/norm(double(degr))

%Pltoting the Results

figure

subplot(2,2,1)

imshow(org),title('Original');

subplot(2,2,2)

imshow(degr),title('Degraded');

subplot(2,2,3)

imshow(rest10),title('Restoration10');

subplot(2,2,4)

imshow(rest6),title('Restoration6');

**Maximum Entropy Pseudo Inverse**

clc;

close all;

clear all;

%input

img=imread('C:\Users\koundi\Desktop\lena.jpg'); %input image in spatial domain

img=rgb2gray(img);

F= fft2(img); %input image in freq domain

M=size(img,1); %rows of input image

N=size(img,2); %columns of the input image

%channel

h = fspecial('gaussian', [3 3], 0.5); %channel in spatial domain

H= fft2(h,M,N); %channel in freq domain

%Noise

noise = input('Enter the magnitude for the noise signal : ');

n =noise\*randn(M,N);

%n =fspecial('gaussian', [3 3], noise); %Noise in spatial domain

Ns=fft2(n,M,N); %noise in freq domain

%Degradation

G= F.\*H +Ns; %degraded image in spatial domain

g = abs(ifft2(G)); %degraded image in freq domain

g=mat2gray(g);

%Restoration

lambda=1;

Fcap = zeros(M,N);

for k1=1:M

for k2=1:N

Fcap(k1,k2) = ( conj( H(k1,k2) ) / ( pow2( abs( H(k1,k2) ) )+ 0.5\*lambda^-1\*pow2( abs( H(k1,k2) ) ) ) )\* G(k1,k2); % restored image in spatial domain

end

end

fcap = abs(ifft2(Fcap)); %restored image in freq domain

fcap= mat2gray(fcap);

%SNR calculation

SNR = norm(F,2)/norm(Ns,2)

%Plotting of Results

figure;

subplot(1,3,1)

imshow(img);

title('original');

subplot(1,3,2)

imshow(g);

title('degraded');

subplot(1,3,3);

imshow(fcap),title('Restored');