

**THE UNIVERSITY OF TEXAS AT ARLINGTON, TEXAS  
DEPARTMENT OF ELECTRICAL ENGINEERING**

**EE 5356**

**DIGITAL IMAGE PROCESSING**

**PROJECT # 11**

**by**

**SOUTRIK MAITI**

**1001569883**

**Presented to**

**Dr. K.R.RAO**

**April 19, 2019**

**Inverse and Wiener Filter**

***MATLAB Code:***

clc;

clear all;

close all;

%% Reading and displaying the original image

img = imread('lena512.bmp');

%% Doing a FFT and shift in the Fourier Domain

FFT\_img = fft2(img);

FFT2\_img = fftshift(FFT\_img);

%% Initializing the matrices for the Wiener filters

NN = 512; % Size of the matrices

WF\_1 = wiener\_fil(0.0025);

WF\_2 = wiener\_fil(0.001);

WF\_3 = wiener\_fil(0.00025);

%% Making DFT of White Noise matrix of size NN

white\_noise = randn(NN,NN);

DFT\_white\_noise = fft2(white\_noise);

DFT\_1 = fft2(zeros(512));

DFT\_2 = fft2(zeros(512));

DFT\_3 = fft2(zeros(512));

%% Images corrupted with white noise for different k in Fourier Domain

G\_uv\_1 = FFT2\_img.\*WF\_1 + DFT\_white\_noise;

G\_uv\_2 = FFT2\_img.\*WF\_2 + DFT\_white\_noise;

G\_uv\_3 = FFT2\_img.\*WF\_3 + DFT\_white\_noise;

%% Taking IFT of the degraded images in the Fourier Domain

deg\_img\_1 = ifft2(ifftshift(G\_uv\_1));

deg\_img\_2 = ifft2(ifftshift(G\_uv\_2));

deg\_img\_3 = ifft2(ifftshift(G\_uv\_3));

%% Displaying Original image and image corrupted with noise

figure(1);

subplot(2,2,1);

imshow(img);

title('original image');

subplot(2,2,2);

imshow(uint8(deg\_img\_1));

nme\_one\_1=sprintf('Image degraded for k = 0.0025');

title(nme\_one\_1);

subplot(2,2,3);

imshow(uint8(deg\_img\_2));

nme\_one\_1=sprintf('Image degraded for k = 0.001');

title(nme\_one\_1);

subplot(2,2,4);

imshow(uint8(deg\_img\_3));

nme\_one\_1=sprintf('Image degraded for k = 0.00025');

title(nme\_one\_1);

%% saving the results

saveas(gca,'white\_noise\_wf.jpg');

%% Restoring the image with Inverse filter

%% Initializing the inverse filter for different k

iwf\_1 = inv\_fil\_123(WF\_1);

iwf\_2 = inv\_fil\_123(WF\_2);

iwf\_3 = inv\_fil\_123(WF\_3);

%% Filtering the images

id\_img\_1 = G\_uv\_1 .\* iwf\_1;

id\_img\_2 = G\_uv\_2 .\* iwf\_2;

id\_img\_3 = G\_uv\_3 .\* iwf\_3;

%% taking IFT of the filtered images

res\_img\_1 = ifft2(ifftshift(id\_img\_1));

res\_img\_2 = ifft2(ifftshift(id\_img\_2));

res\_img\_3 = ifft2(ifftshift(id\_img\_3));

%% Displaying the results

figure(2);

subplot(2,2,1);

imshow(img);

title('original image');

subplot(2,2,2);

imshow(uint8(res\_img\_1));

title('Inverse filtering for k = 0.0025');

subplot(2,2,3);

imshow(uint8(res\_img\_2));

title('Inverse filtering for k = 0.001');

subplot(2,2,4);

imshow(uint8(res\_img\_3));

title('Inverse filtering for k = 0.00025');

%% saving the results

saveas(gca,'res\_img\_iwf.jpg');

%% Restoring images with Wiener filter

rtio1=sum(sum(abs(DFT\_white\_noise)))/sum(sum(abs(DFT\_1)))

rtio2=sum(sum(abs(DFT\_white\_noise)))/sum(sum(abs(DFT\_2)))

rtio3=sum(sum(abs(DFT\_white\_noise)))/sum(sum(abs(DFT\_3)))

R\_u\_1 = zeros(512);

R\_u\_1 = abs(fftshift(ifft2(fft2(img).\*conj(fft2(img)))))./(512^2);

R\_n\_1 = zeros(512);

R\_n\_1 = abs(fftshift(ifft2(fft2(white\_noise).\*conj(fft2(white\_noise)))))./(512^2);

S\_u\_1 = fftshift(fft2(R\_u\_1));

S\_n\_1 = fftshift(fft2(R\_n\_1));

%% Applying Wiener filter with different k in the FD

W\_img\_1 = conj(WF\_1).\*S\_u\_1./((abs(WF\_1).^2).\*S\_u\_1+S\_n\_1);

W\_img\_2 = conj(WF\_2).\*S\_u\_1./((abs(WF\_2).^2).\*S\_u\_1+S\_n\_1);

W\_img\_3 = conj(WF\_3).\*S\_u\_1./((abs(WF\_3).^2).\*S\_u\_1+S\_n\_1);

Inv\_1 = G\_uv\_1.\*W\_img\_1;

Inv\_2 = G\_uv\_2.\*W\_img\_2;

Inv\_3 = G\_uv\_3.\*W\_img\_3;

%% Taking the inverse FT of the filtered images

restrd\_img1\_mtrx\_512\_1 = ifft2(ifftshift(Inv\_1));

restred\_img2\_mtrx\_512\_2 = ifft2(ifftshift(Inv\_2));

restred\_img3\_mtrx\_512\_3 = ifft2(ifftshift(Inv\_3));

%% Displaying the results

figure(3);

subplot(2,2,1);

imshow(img);

title('original image');

subplot(2,2,2);

imshow(uint8(restrd\_img1\_mtrx\_512\_1));

title('Wiener Filtering for k = 0.0025');

subplot(2,2,3);

imshow(uint8(restred\_img2\_mtrx\_512\_2));

title('Wiener Filtering for k = 0.001');

subplot(2,2,4);

imshow(uint8(restred\_img3\_mtrx\_512\_3));

title('Wiener Filtering for k = 0.00025');

%% saving the results

saveas(gca,'res\_img\_wf.jpg');

%% Functions for this project

%% Algorithm for inverse filter

function [ K ] = inv\_fil\_123(H1)

e = 0.001;

N\_s = 512;

for uu\_1 = 1:N\_s

for vv = 1:N\_s

if(H1(uu\_1,vv) < e)

K(uu\_1,vv) = 0;

else

K(uu\_1,vv) = 1/H1(uu\_1,vv);

end

end

end

end

%% Algorithm for wiener filter

function Hh = wiener\_fil(k)

N\_s = 512;

for uu\_11 = 1:N\_s

for vv\_11 = 1:N\_s

Hh(uu\_11,vv\_11) = exp(-k\*((uu\_11-N\_s/2)^2+(vv\_11-N\_s/2)^2)^(5/6));

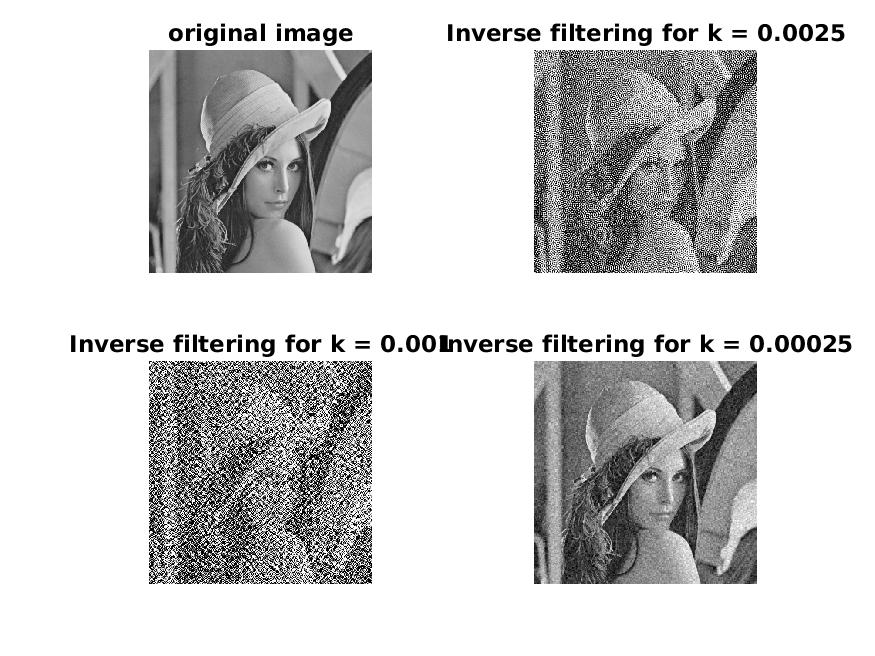
end

end

end

***Results:***

******





***Conclusion:***

* From the output of a noiseless linear system, the inverse filter restores a blurred image perfectly. But it does not work perform well when there is presence of additive white noise.
* Wiener filtering performs far better in restoring images even in the combination of presence of blur and noise.