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:

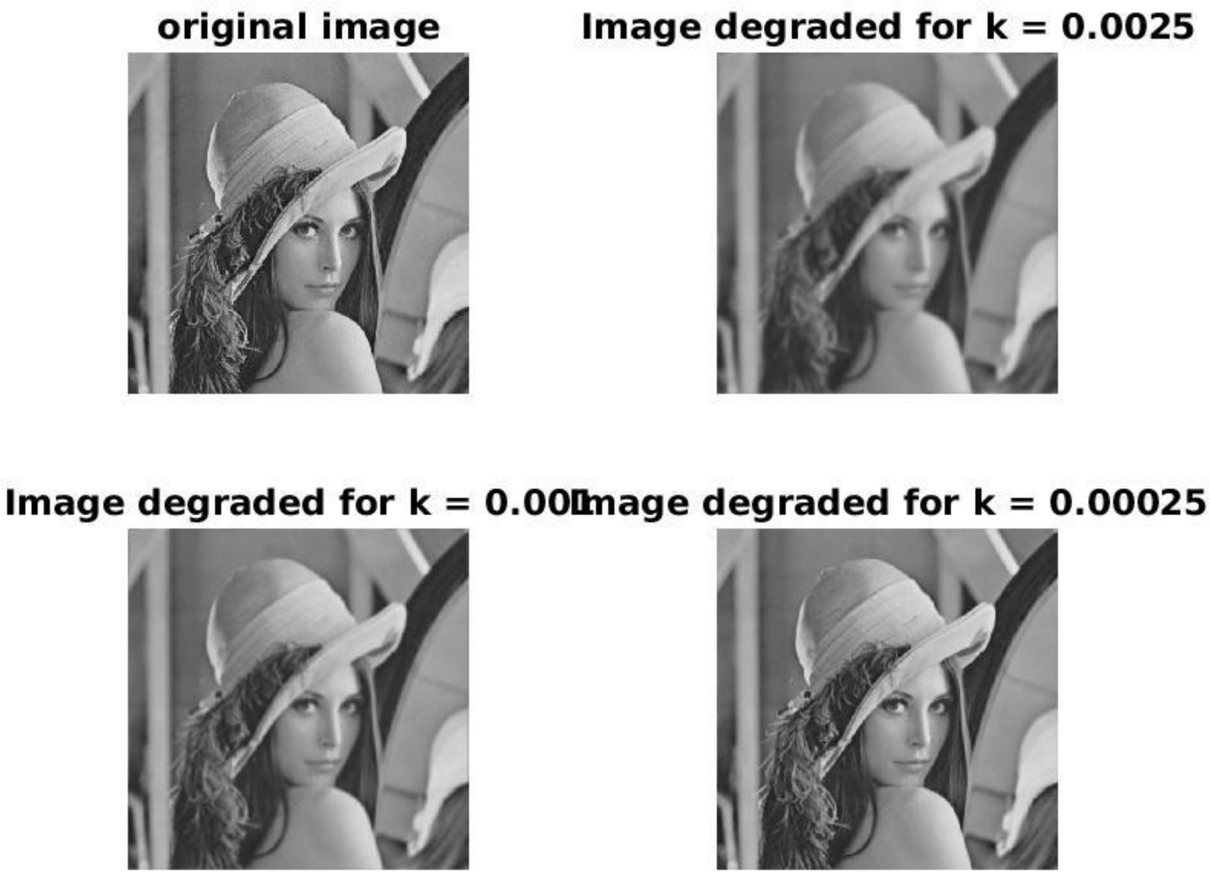
```
clc;
clear all;
close all;

%% 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);
```

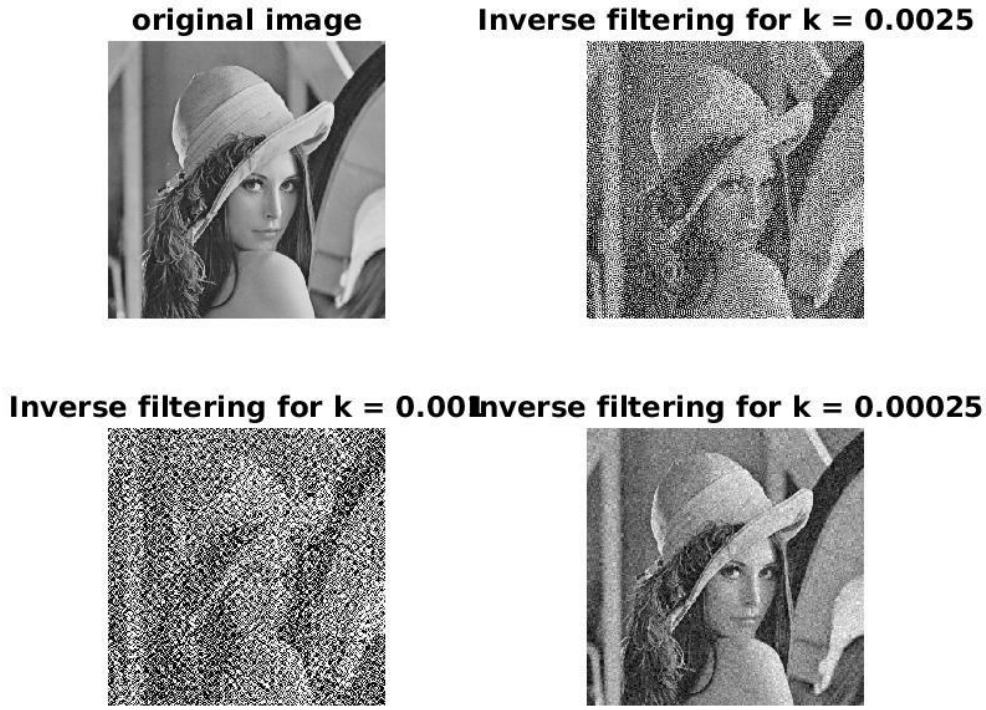
```
subplot(2,2,1);
imshow(img);
title('original image');
subplot(2,2,2);
imshow(uint8(deg_img_1));
nme_one_1=sprintf('Image degraded for k = 0.0025');
title(nme_one_1);
subplot(2,2,3);
imshow(uint8(deg_img_2));
nme_one_1=sprintf('Image degraded for k = 0.001');
title(nme_one_1);
subplot(2,2,4);
imshow(uint8(deg_img_3));
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);
```

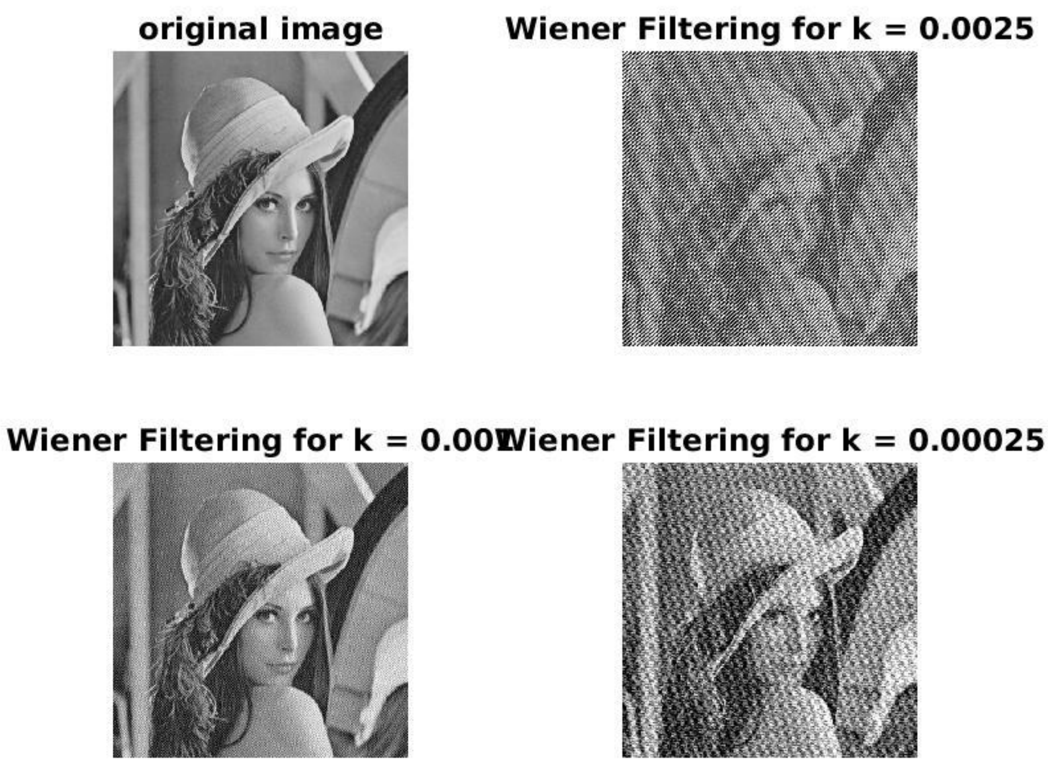
```
imshow(uint8(res_img_2));
title('Inverse filtering for k = 0.001');

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');
```

```matlab
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:


original image
Image degraded for k = 0.0025
Image degraded for k = 0.001
Image degraded for k = 0.00025


original image
Inverse filtering for k = 0.0025
Inverse filtering for k = 0.001
Inverse filtering for k = 0.00025


original image
Wiener Filtering for k = 0.0025
Wiener Filtering for k = 0.001
Wiener Filtering for k = 0.00025

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.