

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

**EE 5356**

**DIGITAL IMAGE PROCESSING**

**PROJECT # 12**

**by**

**SOUTRIK MAITI**

**1001569883**

**Presented to**

**Dr. K.R.RAO**

**April 26, 2019**

**Geometric Mean Filter**

***MATLAB Code:***

clc;

clear all;

close all;

%% Reading the image

img = imread('lena512.bmp');

img = img(:,:,1);

%% Setting the size for the GM filter

Gx\_x = size(img);

Nm\_0 = Gx\_x(1);

%% Plotting the original image

figure(1);

imshow(img);

title('Original Image');

saveas(gca,'Origin.jpg');

%% Converting the image to the Fourier transform

imge\_ft = fft2(img);

img3\_one = fftshift(imge\_ft);

Tx\_x = zeros(Nm\_0);

Tx\_y = zeros(Nm\_0);

Tx\_z = zeros(Nm\_0);

%% Filtering the images with different k

Tx\_x = h\_fil\_ter(0.0025,Nm\_0);

Tx\_y = h\_fil\_ter(0.001,Nm\_0);

Tx\_z = h\_fil\_ter(0.00025,Nm\_0);

Aa\_x = randn(Nm\_0,Nm\_0);

Aax\_FT = fft2(Aa\_x);

h1\_Filtone = fft2(Tx\_x);

h2\_Filttwo = fft2(Tx\_y);

h3\_Filtthree = fft2(Tx\_z);

Ww\_x = img3\_one.\*Tx\_x + Aax\_FT;

Img3\_Tx = ifft2(ifftshift(Ww\_x));

Ww\_y = img3\_one.\*Tx\_y + Aax\_FT;

Img3\_Ty = ifft2(ifftshift(Ww\_y));

Ww\_z = img3\_one.\*Tx\_z + Aax\_FT;

Imge\_Tz = ifft2(ifftshift(Ww\_z));

%% Displaying the degraded images

figure(2);

subplot(3,1,1);

imshow(uint8(Img3\_Tx));

title('Degraded image with k = 0.0025');

subplot(3,1,2);

imshow(uint8(Img3\_Ty));

title('Degraded image with k = 0.001');

subplot(3,1,3);

imshow(uint8(Imge\_Tz));

title('Degraded image with k = 0.00025');

saveas(gca,'deg\_imgs.jpg');

%% Steps for image restoration

Qq\_u = zeros(Nm\_0);

Qq\_u = abs(fftshift(ifft2(fft2(img).\*conj(fft2(img)))))./(Nm\_0^2);

Qq\_v = zeros(Nm\_0);

Qq\_v = abs(fftshift(ifft2(fft2(Aa\_x).\*conj(fft2(Aa\_x)))))./(Nm\_0^2);

Qq\_w = fftshift(fft2(Qq\_u));

Qq\_x = fftshift(fft2(Qq\_v));

Ee\_x = Geo\_Mean\_filt(Qq\_w,Qq\_x,Tx\_x,0.25,Nm\_0);

Ee\_y = Geo\_Mean\_filt(Qq\_w,Qq\_x,Tx\_y,0.25,Nm\_0);

Ee\_z = Geo\_Mean\_filt(Qq\_w,Qq\_x,Tx\_z,0.25,Nm\_0);

ResultantA = Ww\_x.\*Ee\_x;

ResultantB = Ww\_y.\*Ee\_y;

ResultantC = Ww\_z.\*Ee\_z;

IR\_fil\_1 = ifft2(ifftshift(ResultantA));

IR\_fil\_2 = ifft2(ifftshift(ResultantB));

IR\_fil\_3 = ifft2(ifftshift(ResultantC));

%% Displaying Images with different K values

figure(3);

subplot(3,1,1);

imshow(uint8(IR\_fil\_1));

title('Restored image with values k = 0.0025 and s= 0.25');

subplot(3,1,2);

imshow(uint8(IR\_fil\_2));

title('Restored image with values k = 0.001 and s= 0.25');

subplot(3,1,3);

imshow(uint8(IR\_fil\_3));

title('Restored imagewith values k = 0.00025 and s= 0.25');

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

% Manually Defined Functions

% Geometric Mean Filter:

function Ff\_x = Geo\_Mean\_filt(aa,bb,cc,dd,ee)

Cc\_x = invese\_filtr(cc,ee);

Ff\_x = ((Cc\_x).^dd).\*(aa.\*conj(cc)./(aa.\*(abs(cc).^2)+bb)).^(1-dd);

end

% H Filter:

function Ss\_x = h\_fil\_ter(Ssy,Ssz)

for Hs\_x = 1:Ssz

for Hs\_y = 1:Ssz

Ss\_x(Hs\_x,Hs\_y) = exp(-Ssy\*((Hs\_x-Ssz/2)^2+(Hs\_y-Ssz/2)^2)^(5/6));

end

end

end

% Inverse Filter:

function Cc\_x = invese\_filtr(Cc\_y,Cc\_z)

ep = 0.001;

for Sx = 1:Cc\_z

for Sy = 1:Cc\_z

if(Cc\_y(Sx,Sy) < ep)

Cc\_x(Sx,Sy) = 0;

else

Cc\_x(Sx,Sy) = 1/Cc\_y(Sx,Sy);

end

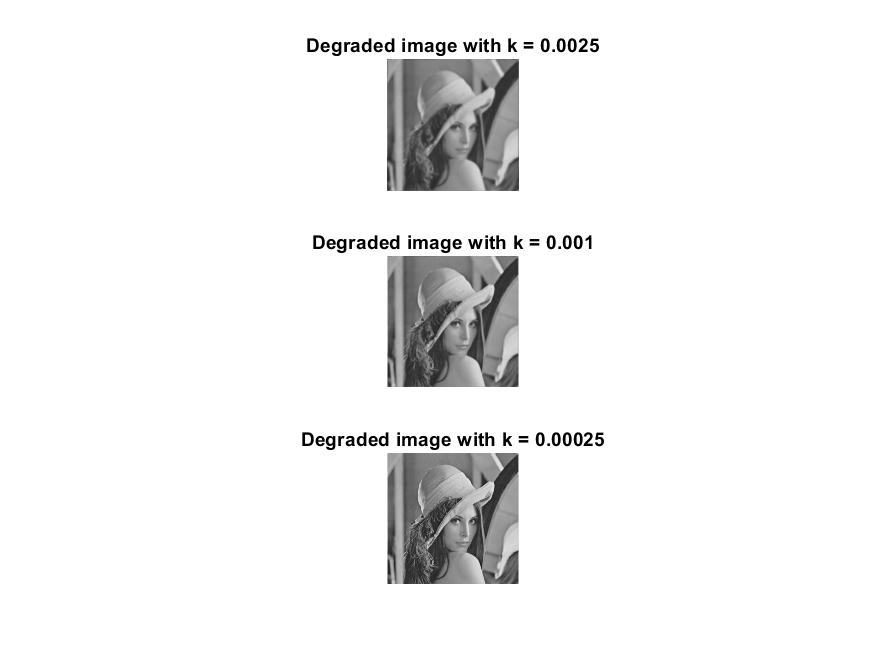
end

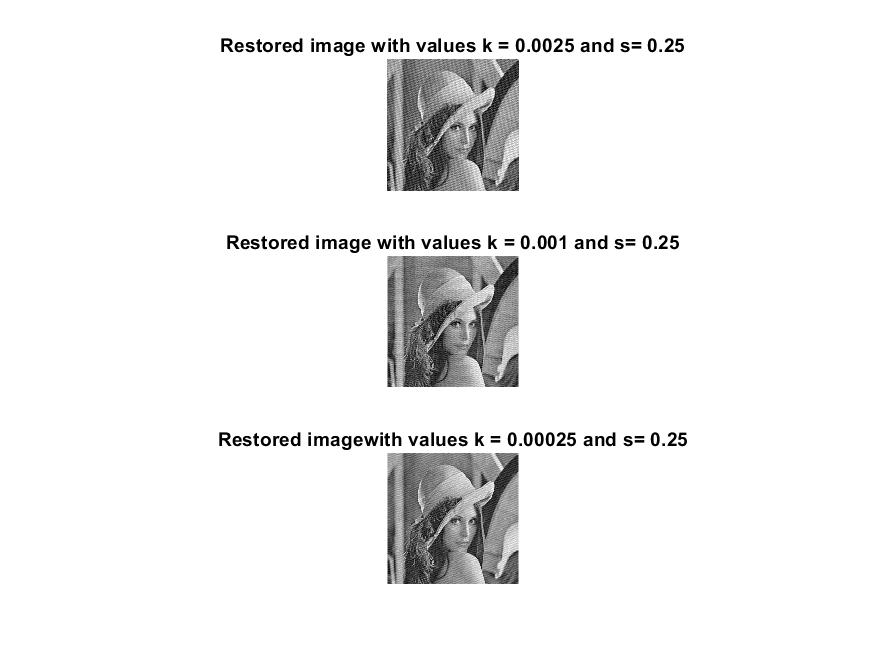
end

end

***Results:***

******





***Conclusion:***

* The formula for the geometric mean is gicen as:



* The GMF is better at eliminating Gaussian type noise as well as at the same time conserving edges when compared to the arithmetic mean filter (AMF).
* The GMF is highly prone to negative outliners.
* The GMF is a combination of both the Wiener filter and constrained least squares principles.
* The resultant image obtained at the end of the geometric mean filter is found to be smoother and sharp.
* The filters are easy to develop as they are designed using Fourier domain and the approximate circuit matrix. It is developed using approximate circuit matrix and Fourier Domain. Filters used here are easy to develop, implement and understand.