

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

**EE 5356**

**DIGITAL IMAGE PROCESSING**

**PROJECT # 6**

**by**

**SOUTRIK MAITI**

**1001569883**

**Presented to**

**Dr. K.R.RAO**

**Mar 18, 2019**

# Non-Linear Filters

***MATLAB CODE:***

%% Read the image

img = imread('lena512.dib.bmp');

img = img(:,:,1);

%% Add different noise to the image

gauss\_img = imnoise(img,'gaussian',0,0.01);

poisson\_img = imnoise(img,'poisson');

snp\_img = imnoise(img,'salt & pepper',0.05);

speckle\_img = imnoise(img,'speckle',0.04);

%% Display original image

figure(1)

imshow(img); title('Orignal Image');

saveas(gca,'original\_image.jpg');

%% Display noisy images

figure(2)

subplot(2,2,1)

imshow(uint8(gauss\_img)); title('Gaussian Noise Img');

subplot(2,2,2)

imshow(uint8(poisson\_img)); title('Poisson Noise Img');

subplot(2,2,3)

imshow(uint8(snp\_img)); title('Salt & Pepper Noise Img');

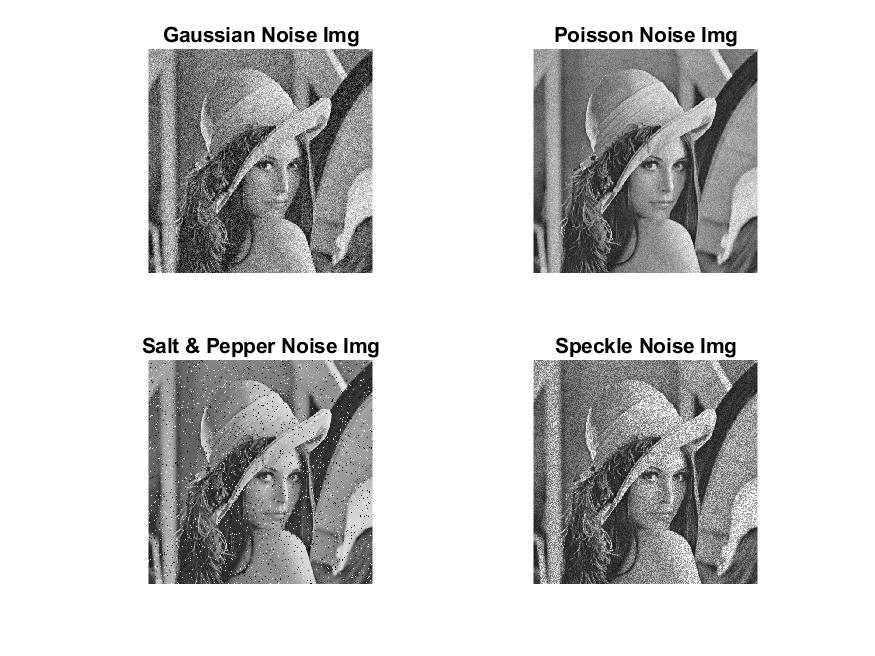
subplot(2,2,4)

imshow(uint8(speckle\_img)); title('Speckle Noise Img');

saveas(gca,'noisy\_images.jpg');

**Results:**

****

****

***Applying Arithmetic mean filter to the images:***

%% Arithmetic mean filter

f = @(x) mean(x(:));

g\_img\_gm = nlfilter(gauss\_img,[3 3],f);

p\_img\_gm = nlfilter(poisson\_img,[3 3],f);

snp\_img\_gm = nlfilter(snp\_img,[3 3],f);

s\_img\_gm = nlfilter(speckle\_img,[3 3],f);

figure;

subplot(2,2,1)

imshow(uint8(g\_img\_gm));

title('Gaussian Noise Arithmetic Mean');

subplot(2,2,2)

imshow(uint8(p\_img\_gm));

title('Poisson Noise Arithmetic Mean');

subplot(2,2,3)

imshow(uint8(snp\_img\_gm));

title('Salt & Pepper Noise Arithmetic Mean');

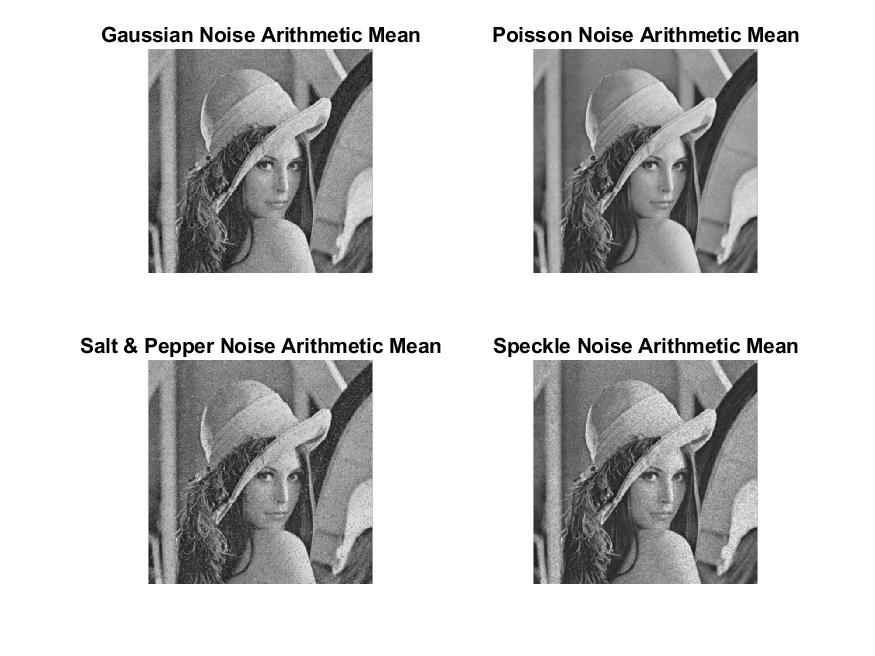
subplot(2,2,4)

imshow(uint8(s\_img\_gm));

title('Speckle Noise Arithmetic Mean');

saveas(gca,'am\_filter\_results.jpg');

***Result:***

******

***Observations:***

*From the above results, it can be observed that* ***arithmetic mean*** *works best for the image with* ***Poisson noise****.*

***Applying Geometric Mean to the noisy images:***

%% Geometric Mean

f = @(x) geomean(x(:));

g\_img\_gm = nlfilter(double(gauss\_img),[3 3],f);

p\_img\_gm = nlfilter(double(poisson\_img),[3 3],f);

snp\_img\_gm = nlfilter(double(snp\_img),[3 3],f);

s\_img\_gm = nlfilter(double(speckle\_img),[3 3],f);

figure;

subplot(2,2,1)

imshow(uint8(g\_img\_gm));

title('Gaussian Noise Geometric Mean');

subplot(2,2,2)

imshow(uint8(p\_img\_gm));

title('Poisson Noise Geometric Mean');

subplot(2,2,3)

imshow(uint8(snp\_img\_gm));

title('Salt & Pepper Noise Geometric Mean');

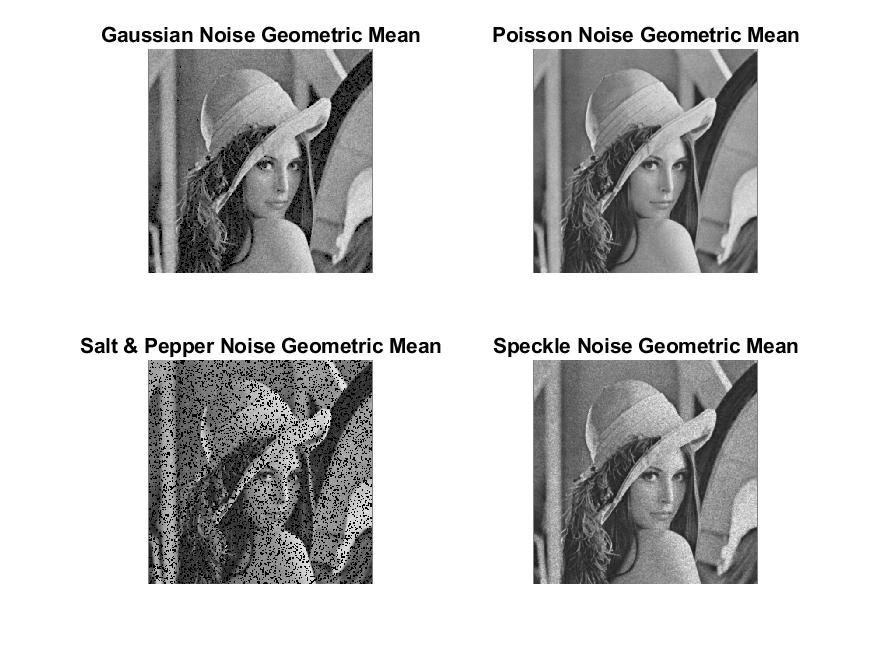
subplot(2,2,4)

imshow(uint8(s\_img\_gm));

title('Speckle Noise Geometric Mean');

saveas(gca,'gm\_filter\_results.jpg');

***Result:***

******

***Observation:***

*From the above results, it can be observed that* ***Geometric mean*** *works best for the image with* ***Poisson noise****.*

***Applying Harmonic mean to the noisy images:***

%% Harmonic Mean

f = @(x) hm\_filter(x(:));

g\_hm = nlfilter(double(gauss\_img),[3 3],f);

p\_hm = nlfilter(double(poisson\_img),[3 3],f);

snp\_hm = nlfilter(double(snp\_img),[3 3],f);

s\_hm = nlfilter(double(speckle\_img),[3 3],f);

figure;

subplot(2,2,1)

imshow(uint8(g\_hm)); title('Gaussian Noise Harmonic Mean');

subplot(2,2,2)

imshow(uint8(p\_hm)); title('Poisson Noise Harmonic Mean');

subplot(2,2,3)

imshow(uint8(snp\_hm)); title('Salt & Pepper Noise Harmonic Mean');

subplot(2,2,4)

imshow(uint8(s\_hm)); title('Speckle Noise Harmonic Mean');

saveas(gca,'hm\_filter\_results.jpg');

%% Implementing Harmonic Mean filter

function mean = hm\_filter(img)

[m,n] = size(img);

sum = 0;Q = 1;

for i=1:m

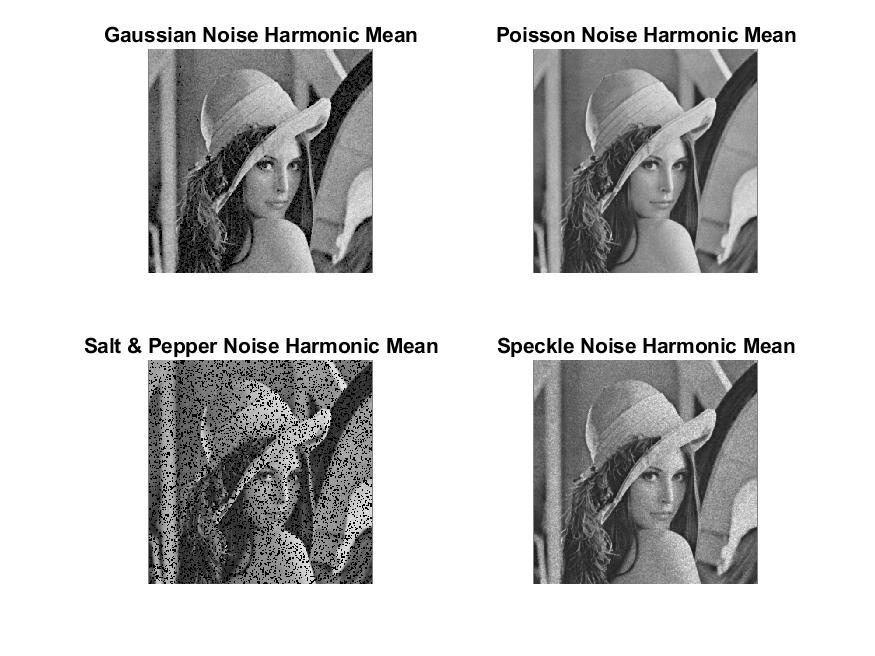
for j=1:n

sum = sum + 1/img(i,j);

end

end

mean = (m\*n)/sum;

***Result:***

***Observation:***

*From the above results, it can be observed that* ***Harmonic mean*** *works best for the image with* ***Poisson noise*** *with* ***satisfactory results*** *for images with* ***Gaussian*** *and* ***Speckle noise****.*

***Applying Contra-Harmonic mean filter to the images:***

%% Contraharmonic Mean

f = @(x) c\_hm\_filter(x(:));

g\_c\_hm = nlfilter(double(gauss\_img),[3 3],f);

p\_c\_hm = nlfilter(double(poisson\_img),[3 3],f);

snp\_c\_hm = nlfilter(double(snp\_img),[3 3],f);

s\_c\_hm = nlfilter(double(speckle\_img),[3 3],f);

figure;

subplot(2,2,1)

imshow(uint8(g\_c\_hm));

title('Gaussian Noise ContraHarmonic Mean');

subplot(2,2,2)

imshow(uint8(p\_c\_hm));

title('Poisson Noise ContraHarmonic Mean');

subplot(2,2,3)

imshow(uint8(snp\_c\_hm));

title('Salt & Pepper Noise ContraHarmonic Mean');

subplot(2,2,4)

imshow(uint8(s\_c\_hm));

title('Speckle Noise ContraHarmonic Mean');

saveas(gca,'c\_hm\_filter\_results.jpg');

%% Implementing Contr-Harmonic Mean filter

function mean = c\_hm\_filter(img)

[m,n] = size(img);

s0 = 0;s1 = 0;Q = 1;

for i=1:m

for j=1:n

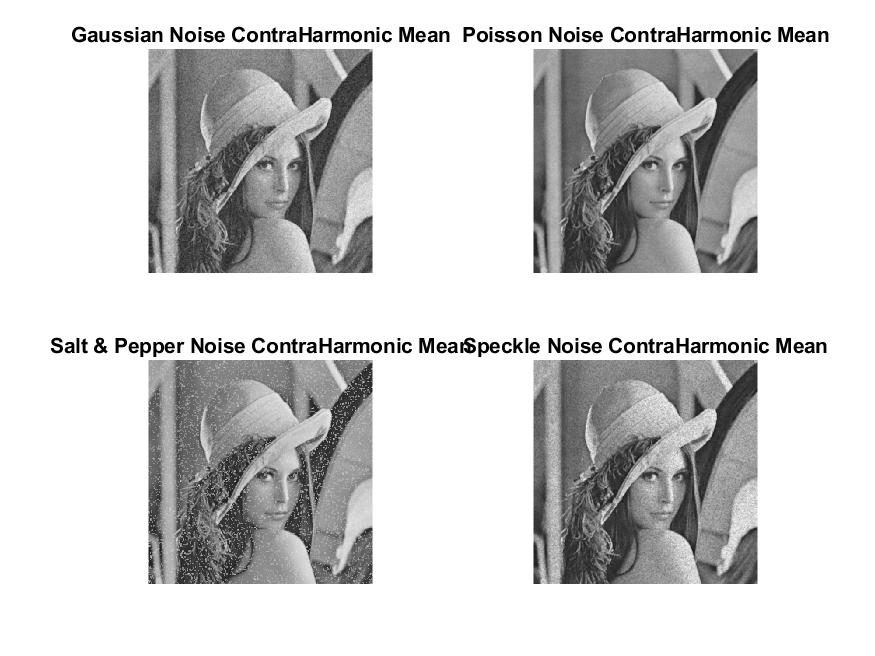
s0 = s0 + img(i,j)^(Q+1);

s1 = s1 + img(i,j)^Q;

end

end

mean = s0/s1;

***Result:***

***Observation:***

*From the above results, it can be observed that* ***Contra-Harmonic mean*** *works best for the image with* ***Poisson noise****.*

***Applying Median Filter to the images:***

%% Median Filter

g\_median = medfilt2(gauss\_img);

p\_median = medfilt2(poisson\_img);

snp\_median = medfilt2(snp\_img);

s\_median = medfilt2(speckle\_img);

figure;

subplot(2,2,1)

imshow(uint8(g\_median));

title('Gaussian Noise Median Filter');

subplot(2,2,2)

imshow(uint8(p\_median));

title('Poisson Noise Median Filter');

subplot(2,2,3)

imshow(uint8(snp\_median));

title('S & P Noise Median Filter');

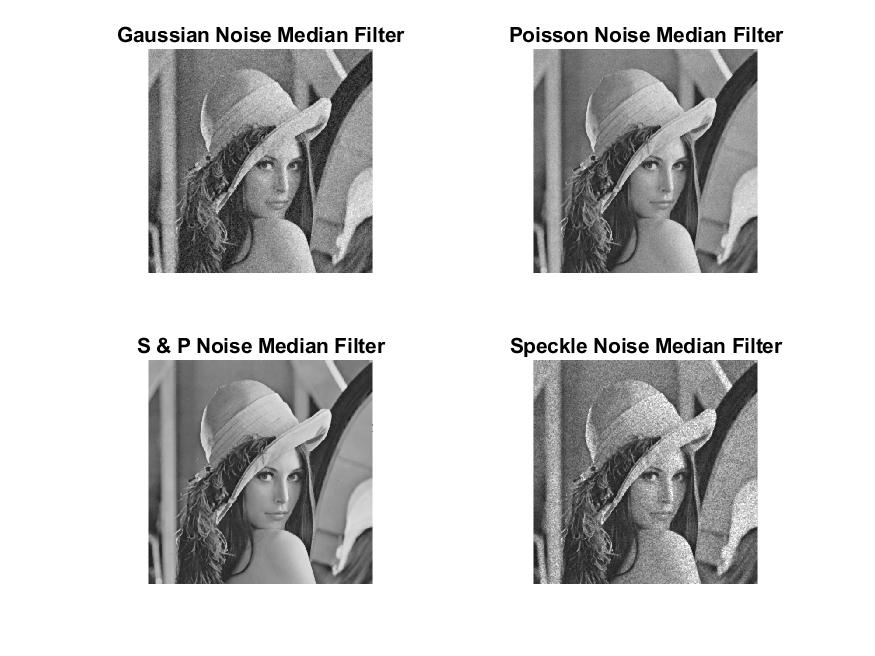
subplot(2,2,4)

imshow(uint8(s\_median));

title('Speckle Noise Median Filter');

saveas(gca,'median\_filter\_results.jpg');

***Result:***

******

***Observation:***

*From the above results, it can be observed that* ***Median filter*** *works best for the image with* ***Salt and Pepper noise****.*

***Applying Min Filter to the images:***

%% Min Filter

g\_min = ordfilt2(gauss\_img,1,ones(3,3));

p\_min = ordfilt2(poisson\_img,1,ones(3,3));

snp\_min = ordfilt2(snp\_img,1,ones(3,3));

s\_min = ordfilt2(speckle\_img,1,ones(3,3));

figure;

subplot(2,2,1)

imshow(uint8(g\_min)); title('Gaussian Noise Min Filter');

subplot(2,2,2)

imshow(uint8(p\_min)); title('Poisson Noise Min Filter');

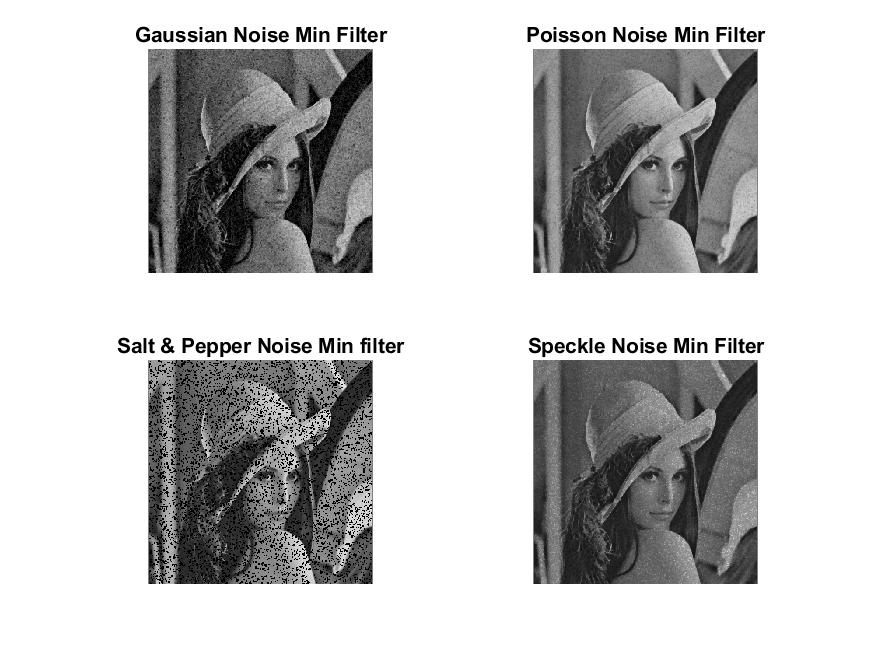
subplot(2,2,3)

imshow(uint8(snp\_min)); title('Salt & Pepper Noise Min filter');

subplot(2,2,4)

imshow(uint8(s\_min)); title('Speckle Noise Min Filter');

saveas(gca,'min\_filter\_results.jpg');

***Result:***

***Observation:***

*Satisfactory results for images with Poisson Noise and Speckle Noise.*

***Applying Max Filter to the images:***

%% Max Filter

g\_max = ordfilt2(gauss\_img,9,ones(3,3));

p\_max = ordfilt2(poisson\_img,9,ones(3,3));

snp\_max = ordfilt2(snp\_img,9,ones(3,3));

s\_max = ordfilt2(speckle\_img,9,ones(3,3));

figure;

subplot(2,2,1)

imshow(uint8(g\_max));

title('Gaussian Noise Max Filter');

subplot(2,2,2)

imshow(uint8(p\_max));

title('Poisson Noise Max Filter');

subplot(2,2,3)

imshow(uint8(snp\_max));

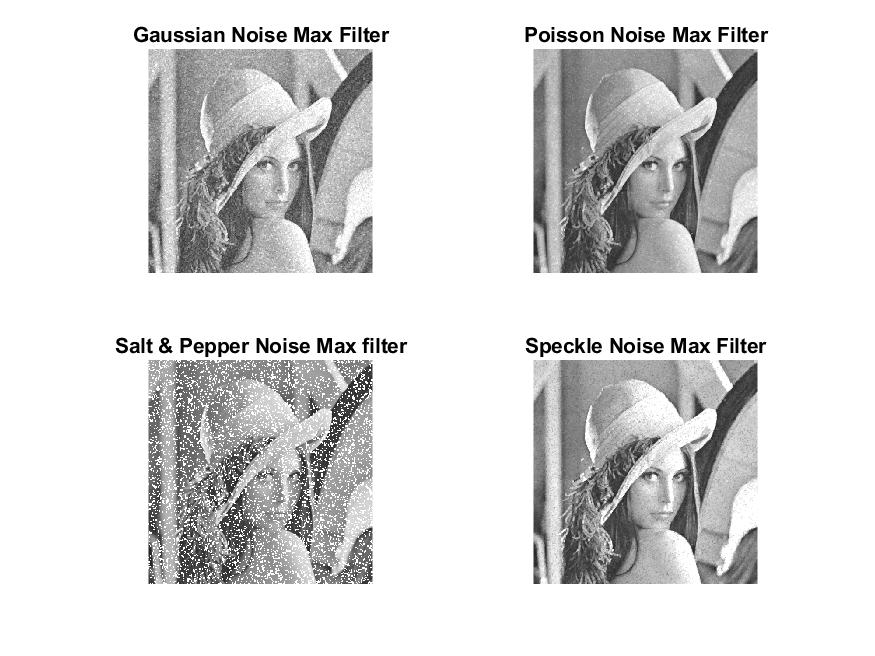
title('Salt & Pepper Noise Max filter');

subplot(2,2,4)

imshow(uint8(s\_max));

title('Speckle Noise Max Filter');

saveas(gca,'max\_filter\_results.jpg');

*** Result:***

***Observation:***

*Satisfactory results in image with Poisson noise.*

***Applying Alpha Trimmed filter to the images:***

%% Alphatrimmed Filter

f = @(x) a\_trim\_filter(x(:));

g\_alpha = nlfilter(double(gauss\_img),[3 3],f);

p\_alpha = nlfilter(double(poisson\_img),[3 3],f);

snp\_alpha = nlfilter(double(snp\_img),[3 3],f);

s\_alpha = nlfilter(double(speckle\_img),[3 3],f);

figure;

subplot(2,2,1)

imshow(uint8(g\_alpha)); title('Gaussian Noise Alphatrimmed Mean');

subplot(2,2,2)

imshow(uint8(p\_alpha)); title('Poisson Noise Alphatrimmed Mean');

subplot(2,2,3)

imshow(uint8(snp\_alpha)); title('Salt & Pepper Noise Alphatrimmed Mean');

subplot(2,2,4)

imshow(uint8(s\_alpha)); title('Speckle Noise Alphatrimmed Mean');

saveas(gca,'alpha\_trim\_filter\_results.jpg');

%% Implementing alpha-trimmed filter

function mean = a\_trim\_filter(img)

[m,n] = size(img);

sum = 0;d = 0;

for i=1:m

for j=1:n

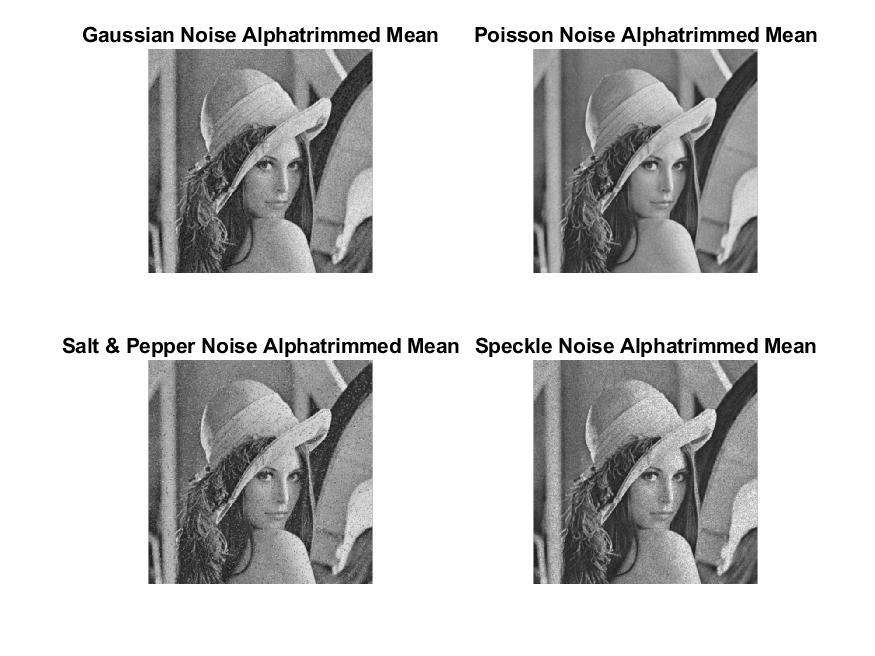
sum = sum + img(i,j);

end

end

mean = sum\*(1/m\*n - d);

***Result:***

******

***Observation:***

*As seen in the above results,* ***Alpha Trimmed*** *works best for image with* ***Poisson noise****.*

***Applying Mid-point filter to the images:***

%% Mid-point Filter

f = @(x) mid\_filter(x(:));

g\_midpt = nlfilter(double(gauss\_img),[3 3],f);

p\_midpt = nlfilter(double(poisson\_img),[3 3],f);

snp\_midpt = nlfilter(double(snp\_img),[3 3],f);

s\_midpt = nlfilter(double(speckle\_img),[3 3],f);

figure;

subplot(2,2,1)

imshow(uint8(g\_midpt)); title('Gaussian Noise Mid-Point Filter');

subplot(2,2,2)

imshow(uint8(p\_midpt)); title('Poisson Noise Mid-Point Filter');

subplot(2,2,3)

imshow(uint8(snp\_midpt)); title('Salt & Pepper Noise Mid-Point Filter');

subplot(2,2,4)

imshow(uint8(s\_midpt)); title('Speckle Noise Mid-Point Filter');

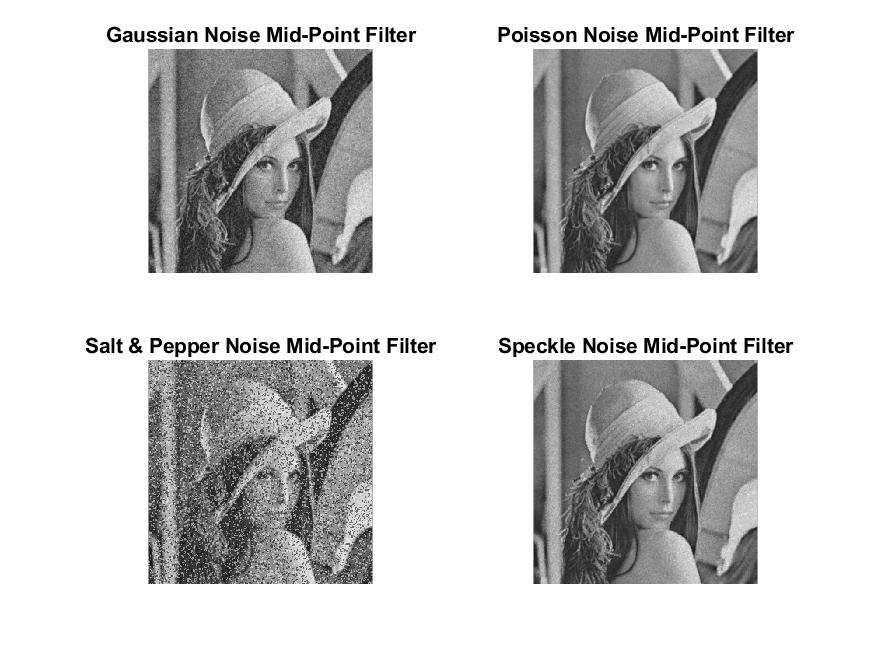
saveas(gca,'mid\_point\_filter\_results.jpg');

%% Implementing mid-point filter

function mean = mid\_filter(img)

mean = (1/2)\*(max(max(img))+min(min(img)));

***Result:***

******

***Observation:***

*Satisfactory results for image with Poisson Noise.*

***Conclusion:***

* It can be seen from the results of applying different filters to the same image with different noise that the success of filters depends on the noise the image has.
* Geometric mean filter works best for images with Poisson Noise.
* Median filter works best for images with salt and pepper noise.
* Arithmetic filter works best for images with Gaussian noise.
* Mid-Point filter works best for images with Speckle noise.