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

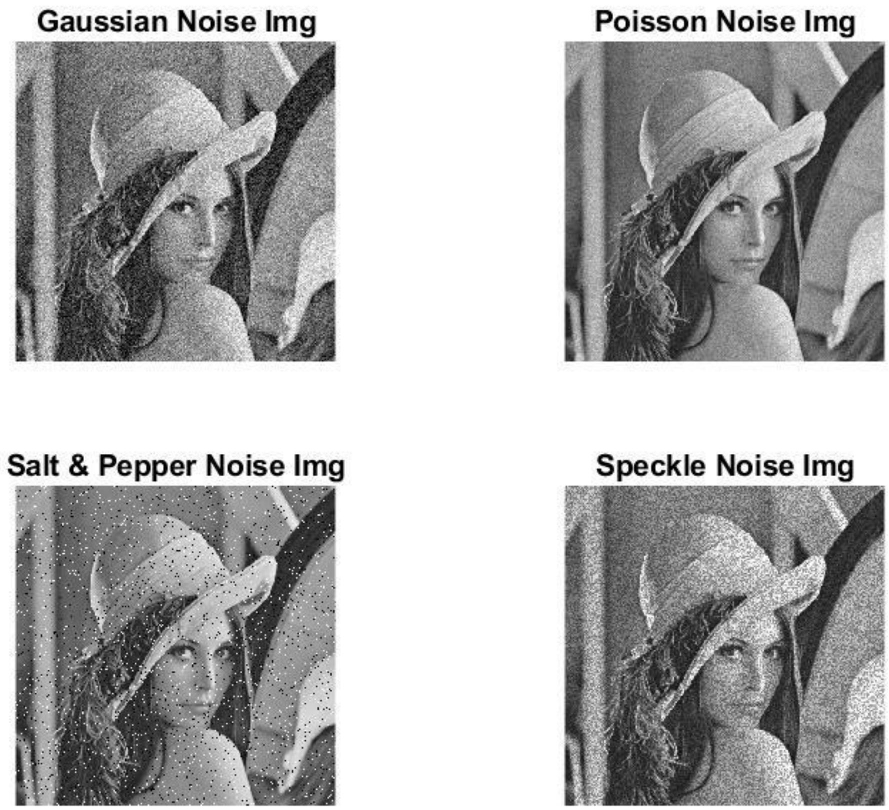
Mar 18, 2019

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:



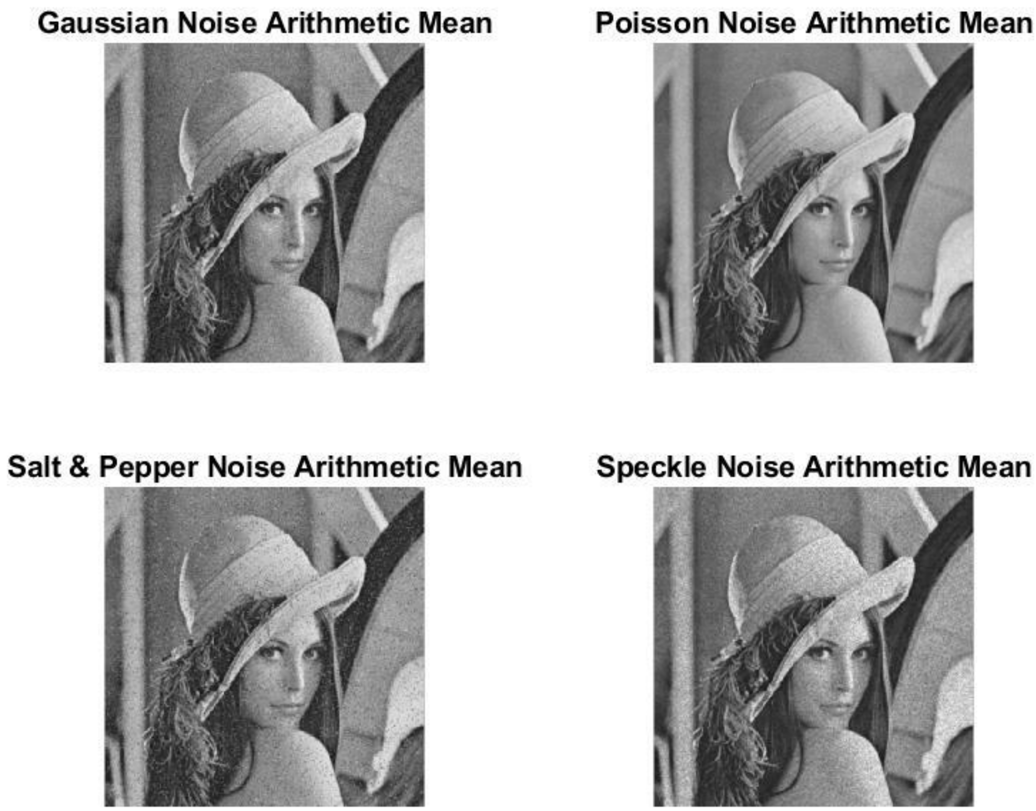
Applying Arithmetic mean filter to the images:

```
%% 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:



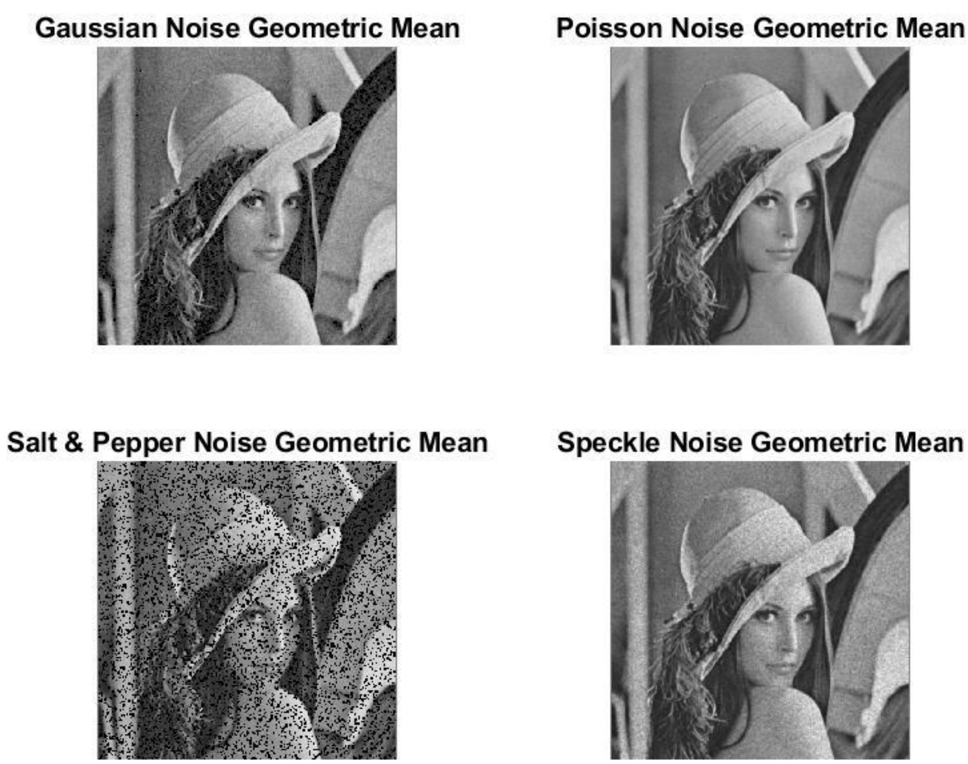
Observations:

*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:

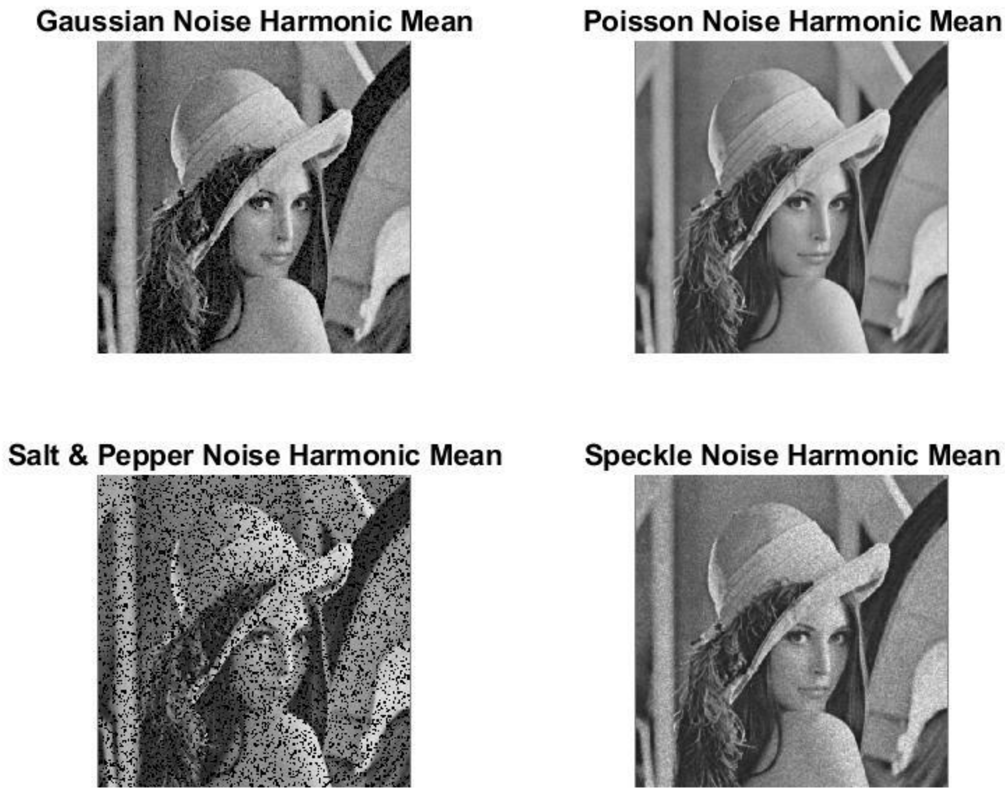
From the above results, it can be observed that ***Geometric mean*** *works best for the image with* ***Poisson noise****.*

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**.*

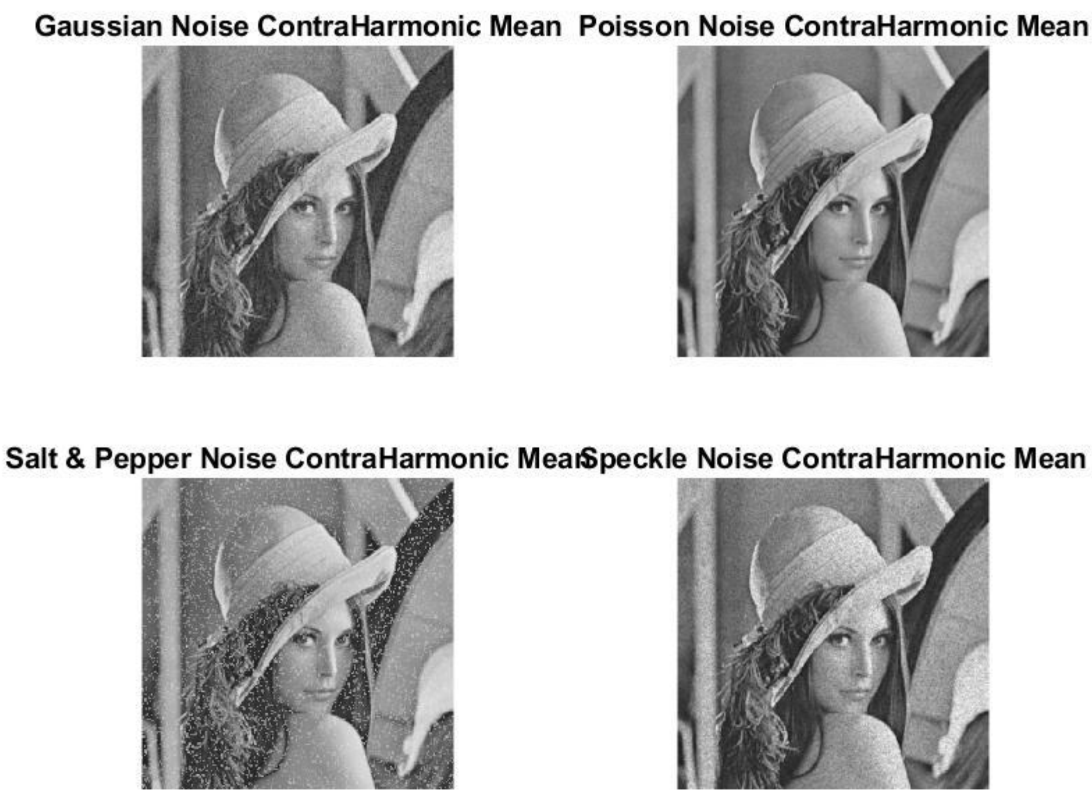
Applying Contra-Harmonic mean filter to the images:

```
%% 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:



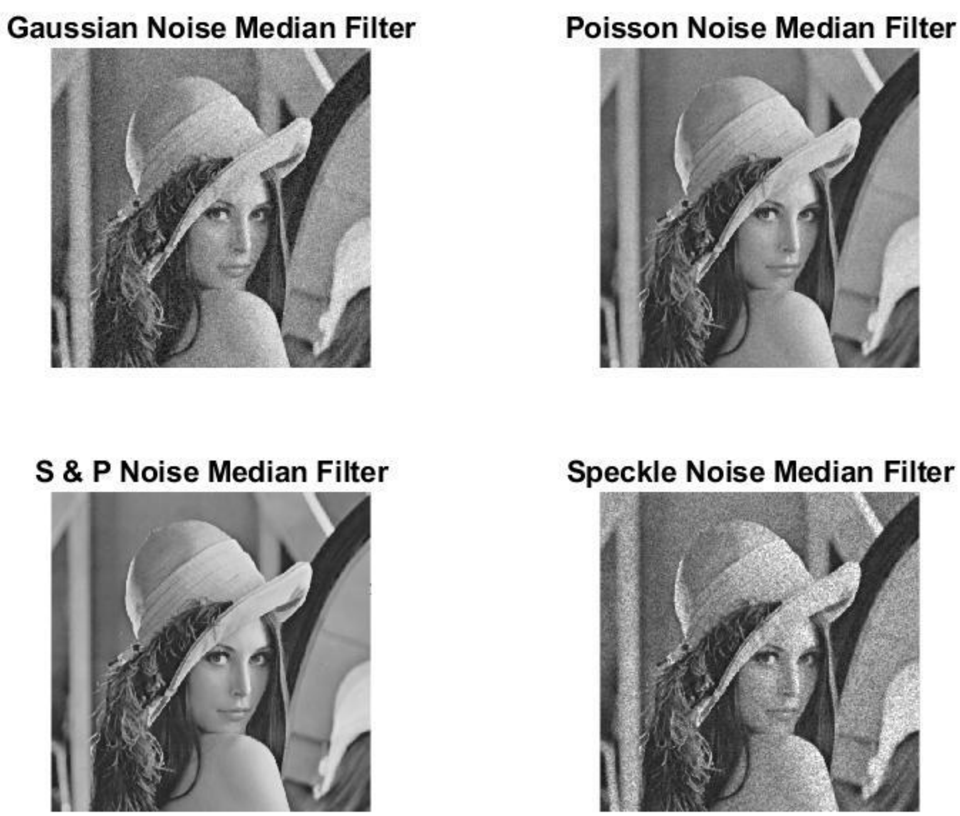
Observation:

*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:

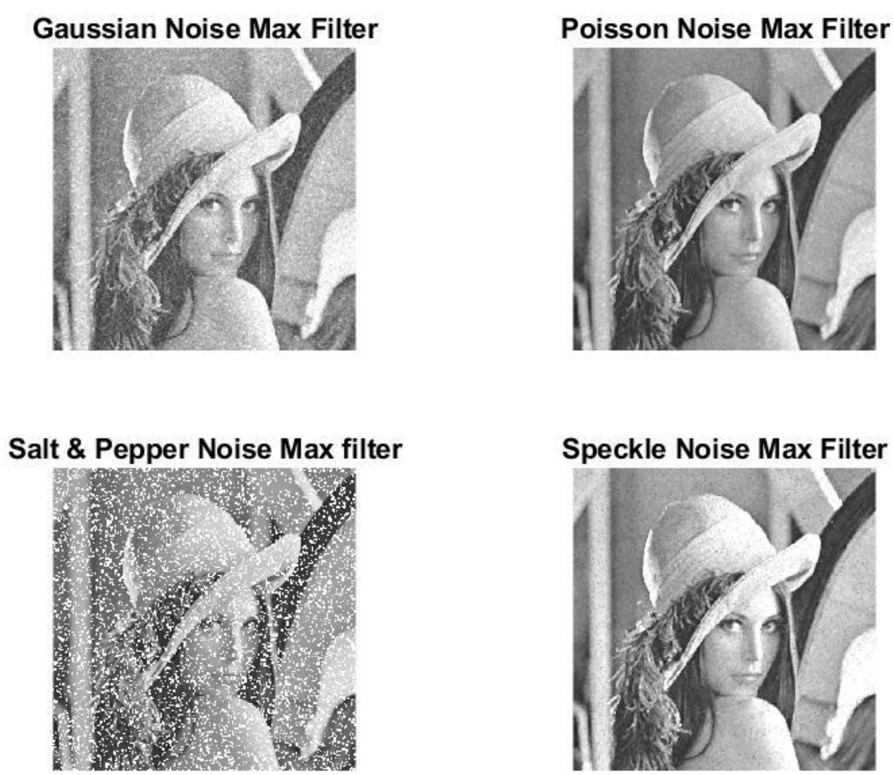
Observation:

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:

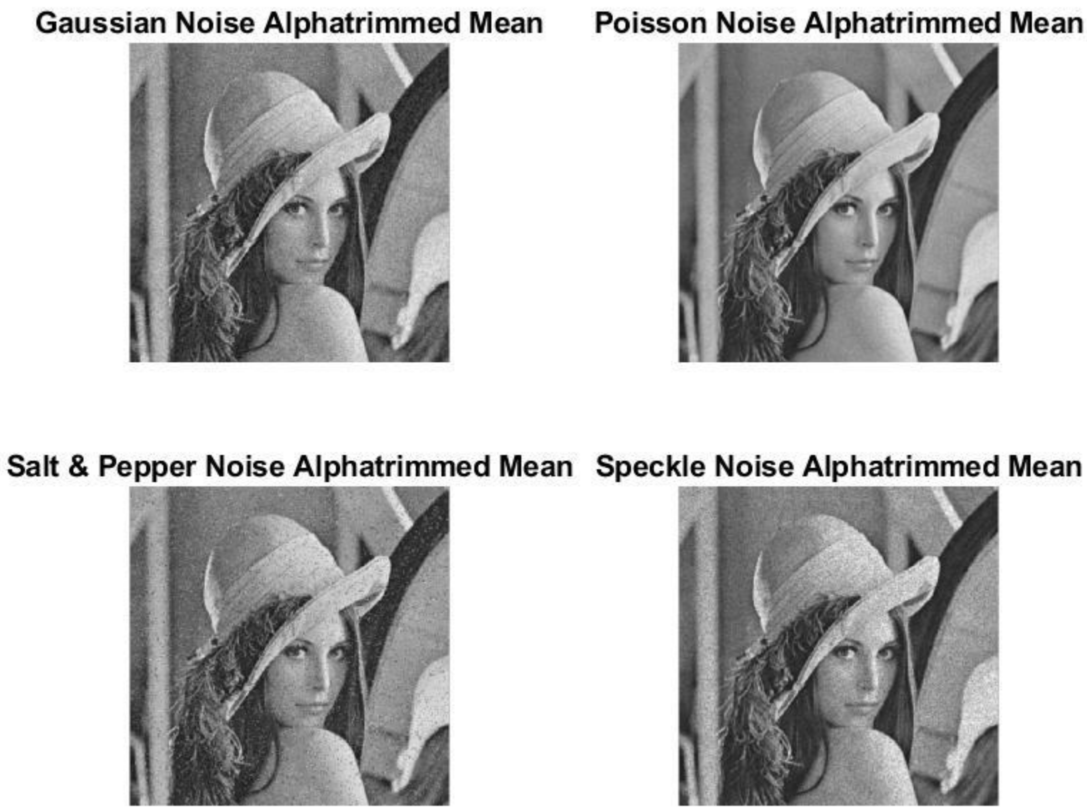
Satisfactory results in image with Poisson noise.

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****.*

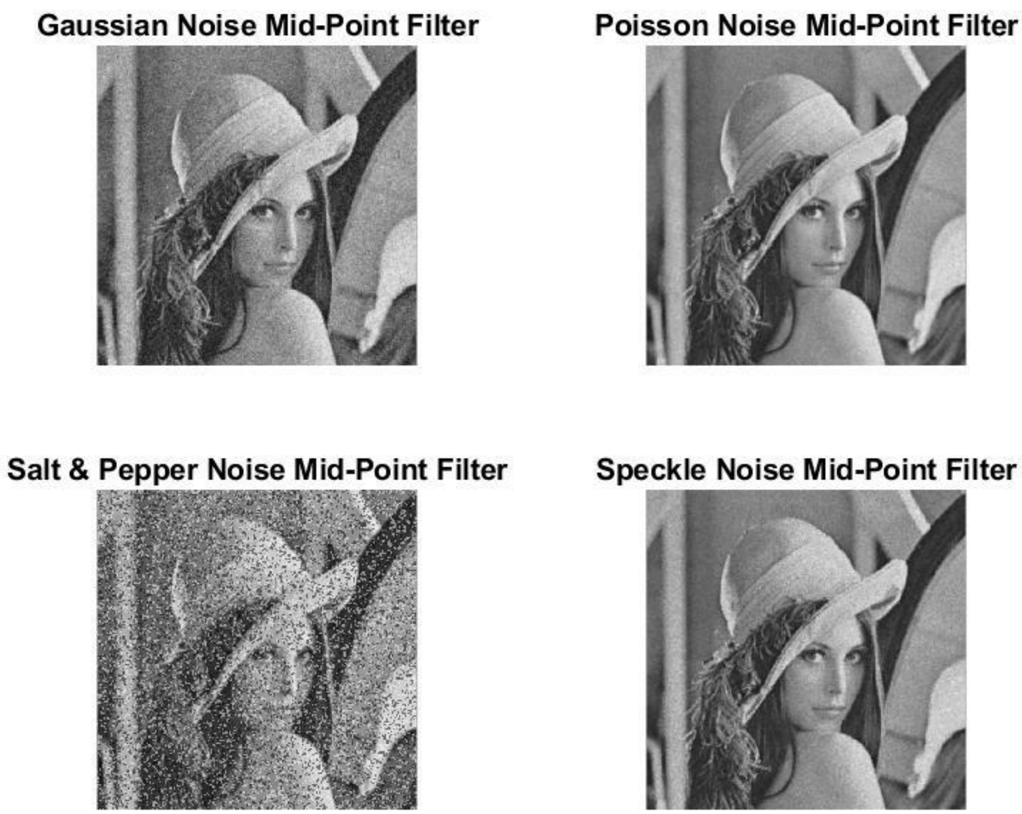
Applying Mid-point filter to the images:

```
%% 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.