Lab 9: Noise reduction using spatial-domain Techniques

Libraries used:

- `import cv2`
- `from PIL import Image`
- `import numpy as np`
- `import skimage.util as sk`
- `import skimage`
- `from scipy import ndimage`
- `from skimage import filters`
- `from google.colab.patches import cv2_imshow`
- `from skimage.morphology import disk, ball`
- `from skimage.filters.rank.generic import geometric_mean`

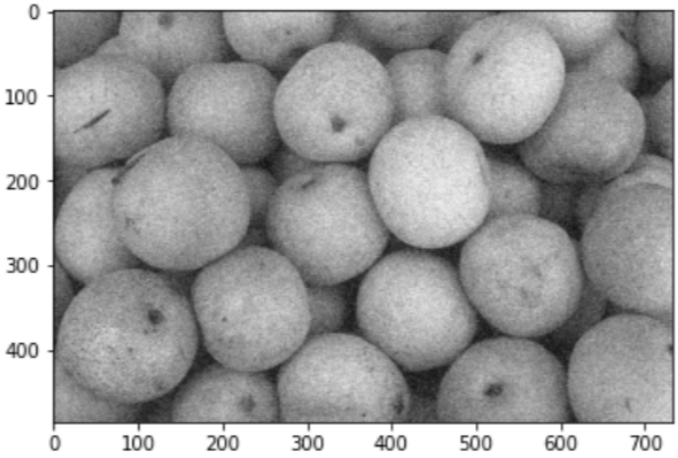
Task 1

Code:

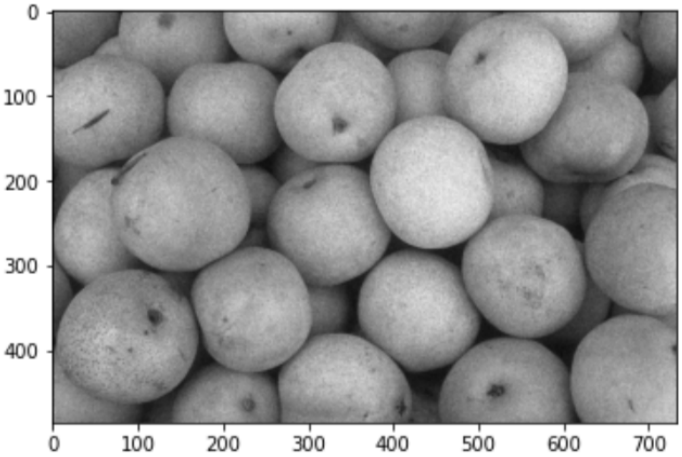
```
im = cv2.imread("/content/pears.png")
im = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

guassian_noise=sk.random_noise(im,"gaussian")
poisson_noise=sk.random_noise(im,"poisson")
salt_pepper=sk.random_noise(im,"s&p")
speckle=sk.random_noise(im,"speckle")
salt_image=sk.random_noise(im,"salt")
pepper_image=sk.random_noise(im,"pepper")

# showing image
plt.imshow(pepper_image,cmap="gray")
plt.show()
```

- **Original Image**

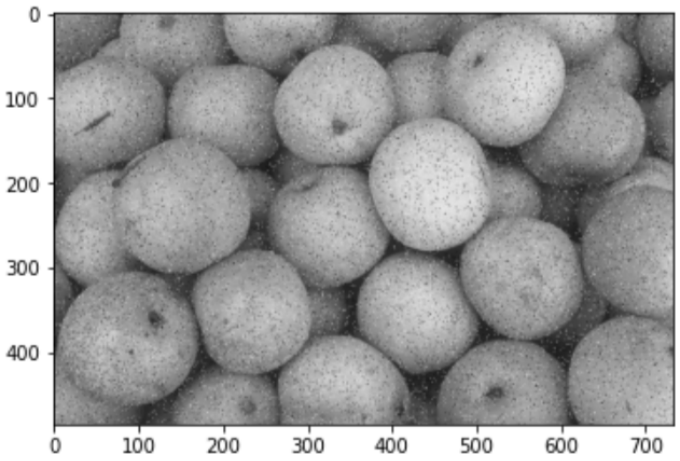
- **Gaussian**



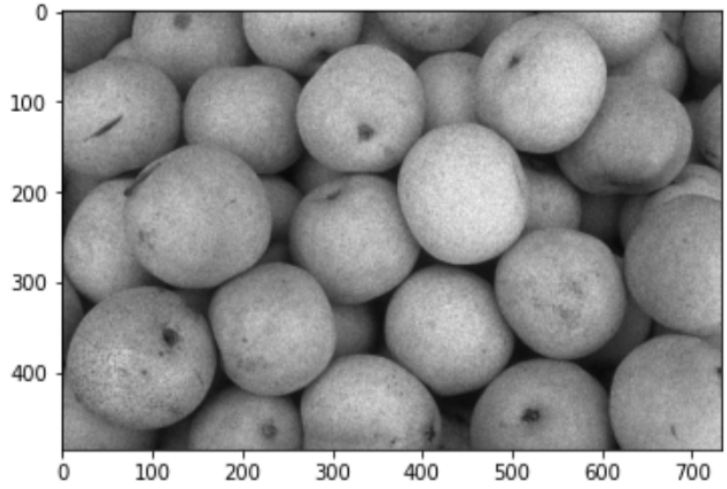
- **Poisson**



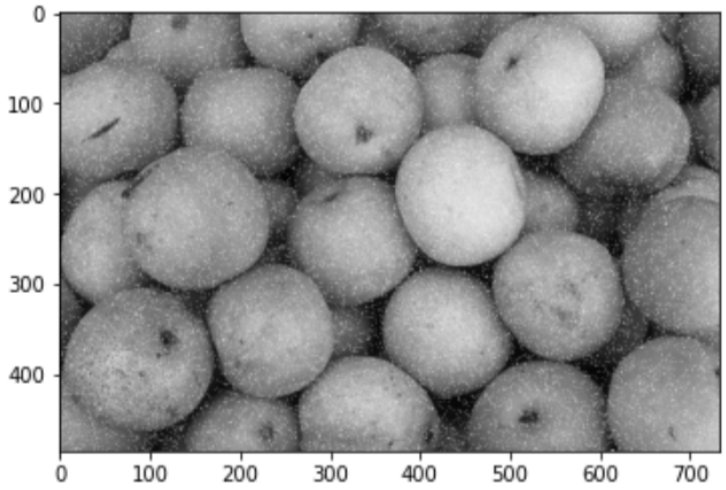
- **Salt & Pepper**



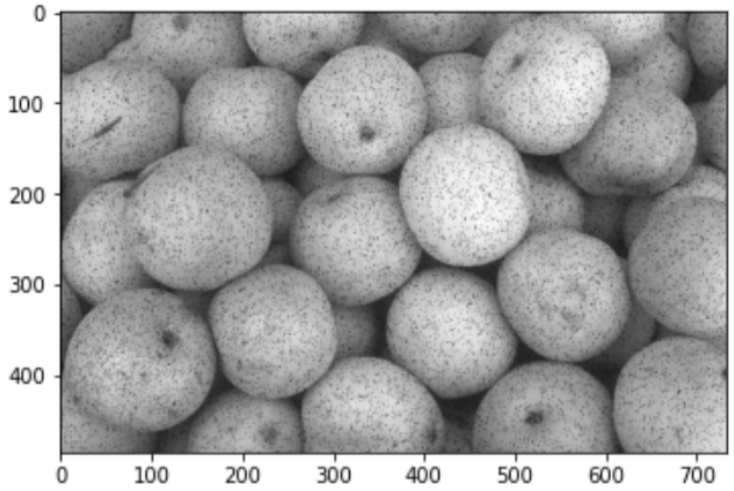
- **Speckle**



- **Salt only noise**



- **Pepper only noise**



Task 2

- **Kernal Size Used (3x3)**

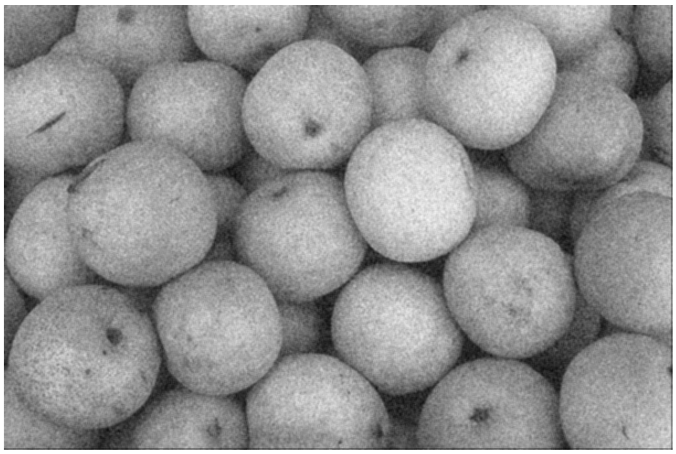
Arithmetic mean

Code:

```
#Arithmetic Mean
im = Image.open("/content/pi.png")
n=3
filter = np.ones((n,n))

im=np.array(im)
for x in range(0,n):
```

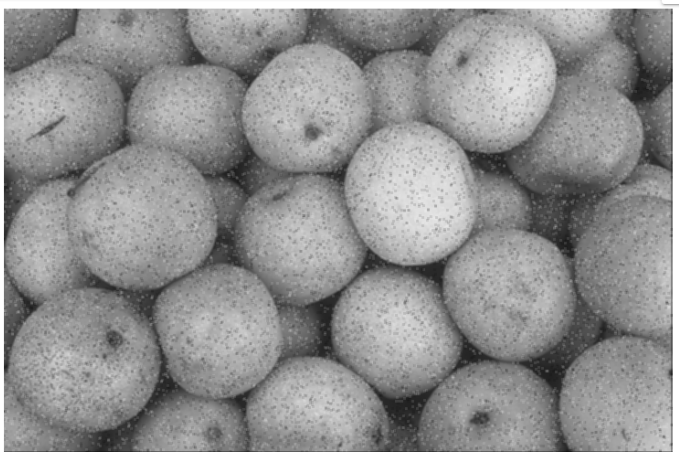
```
  for y in range(0,n):
    filter[x][y]=(filter[x][y])/(n*n)

im = cv2.filter2D(src=im, ddepth=0, kernel=filter)
im=Image.fromarray(im)
im
```

1. 'Gaussian'



2. 'Poisson'

3. 'salt & pepper



4. 'speckle'

5. Salt-only noise

6. Pepper-only noise

- **Findings:**

Arithmetic mean filter implementation blurs the image. It works well for gaussian, Poisson and speckle noise images.

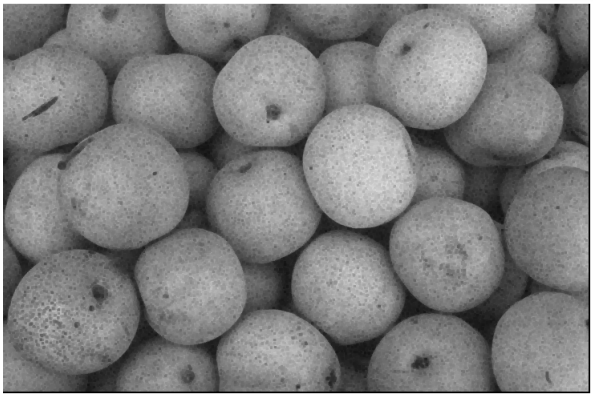
Min filter

Code:

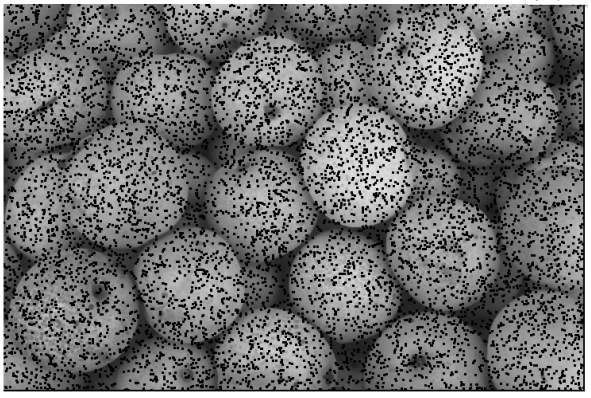
```
#Min Filter
im = Image.open("/content/pi.png")
im=np.array(im)

im=ndimage.minimum_filter(im,3)
im=Image.fromarray(im)
im
```

1. 'Gaussian'

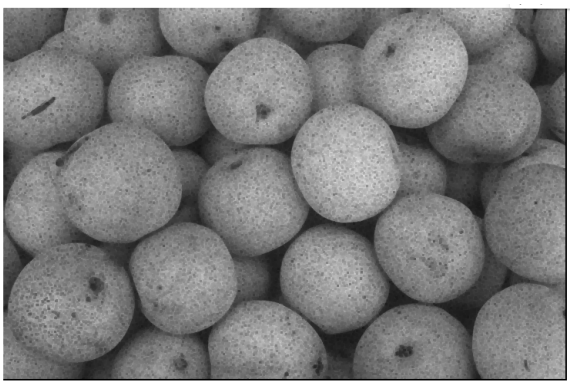
2. 'Poisson'



3. 'salt & pepper

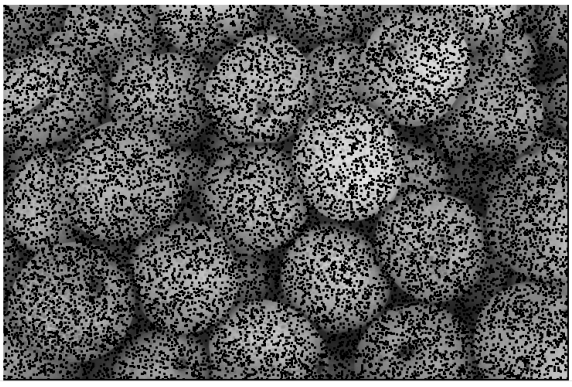


4. 'speckle'



5. Salt-only noise

6. Pepper-only noise



- **Findings:**

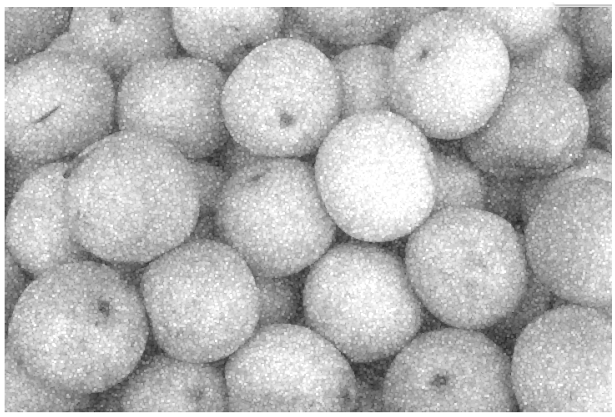
Min filter implementation reduces the salt noise. It works well for salt, Poisson and speckle noise images. If pepper noise is added to an image, the filter replicates more pepper noise.

Max filter

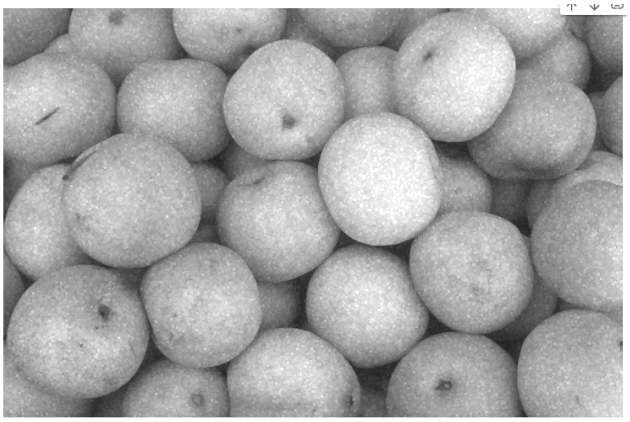
Code:

```
#Max Filter
im = Image.open("/content/pi.png")
im=np.array(im)

im=ndimage.maximum_filter(im,3)
im=Image.fromarray(im)
im
```

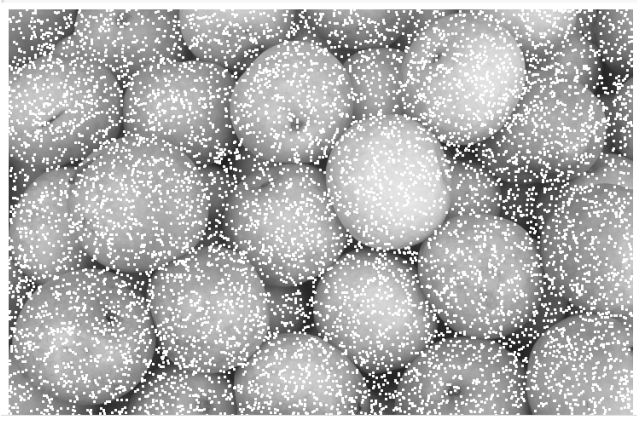
1. 'Gaussian'



2. 'Poisson'

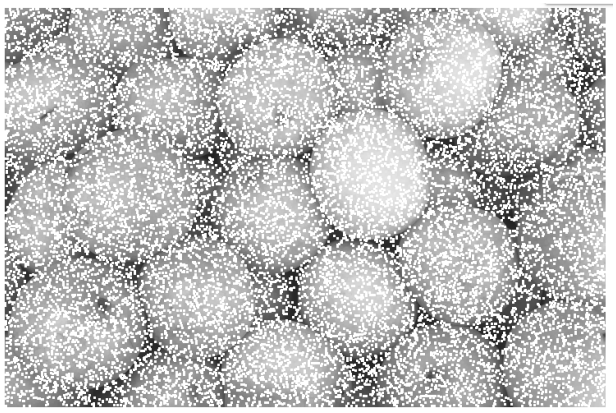


3. 'salt & pepper



4. 'speckle'

5. Salt-only noise



6. Pepper-only noise

- **Findings:**

Max filter implementation reduces pepper noise in the image. It works well for pepper, Poisson and speckle noise images. The filter does not cater for images with salt noise.

Harmonic filter

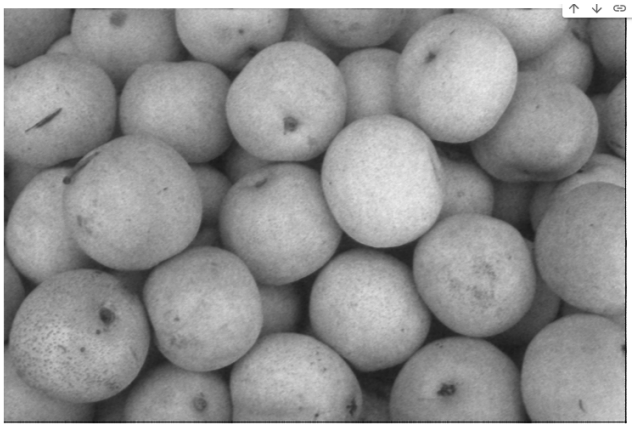
Code:

```
#Harmonic Mean
import statistics as st
im=Image.open("/content/pi.png")
im=np.array(im)

#Resolution: (732, 486)
#Resolution: (734, 488)
#Adding 2 Rows and 2 Coloumns
im=np.concatenate([[im[0]],im],axis=0)
im=np.concatenate([im,[im[resolution[0]-1]]],axis=0)
im=np.transpose(im)
im=np.concatenate([[im[0]],im],axis=0)
im=np.concatenate([im,[im[resolution[0]-1]]],axis=0)
im=np.transpose(im)

for x in range(0,resolution[0]+1):
 for y in range(0,resolution[1]+1):
   if im[x][y]==0:
     im[x][y]=1
 temp = [0]*9 #temp value causing some effect
```

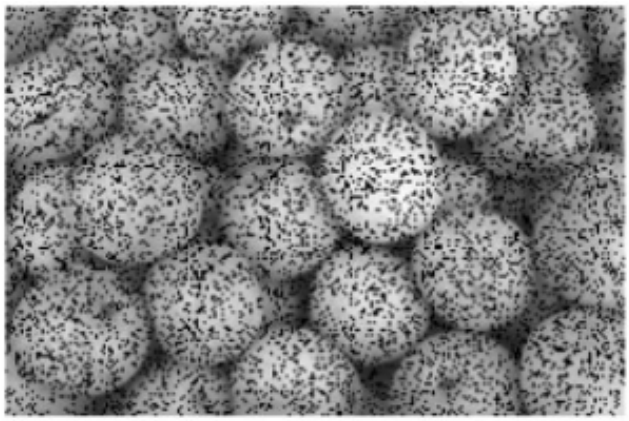
```
for x in range(0,resolution[0]-1):
 for y in range(0,resolution[1]-1):
   temp[0]=(im[x][y])
   temp[1]=(im[x][y-1])
   temp[2]=(im[x][y+1])
   temp[3]=(im[x-1][y])
   temp[4]=(im[x-1][y-1])
   temp[5]=(im[x-1][y+1])
   temp[6]=(im[x+1][y])
   temp[7]=(im[x+1][y-1])
   temp[8]=(im[x+1][y+1])
   im[x][y]=st.harmonic_mean(temp)

im=Image.fromarray(im)
im
```

1. 'Gaussian'

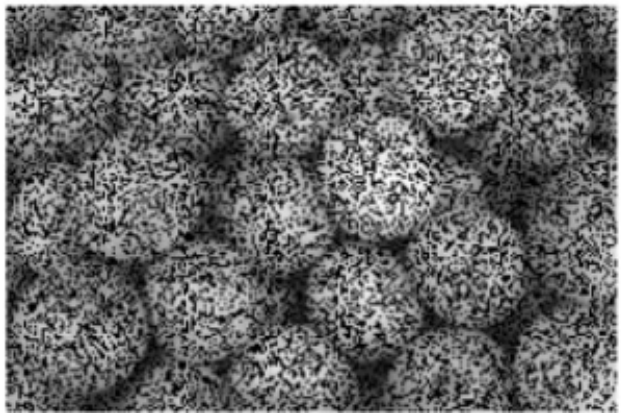
2. 'Poisson'



3. 'salt & pepper

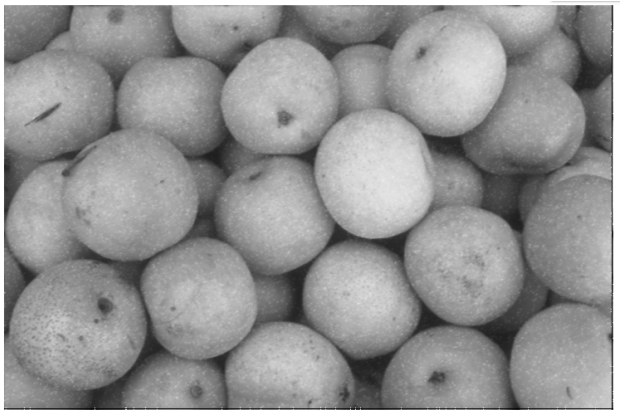


4. Pepper-only noise



5. 'speckle'

6. Salt-only noise



- **Findings:**

The harmonic mean filter is better at removing Gaussian-type noise and preserving edge features than the arithmetic mean filter.

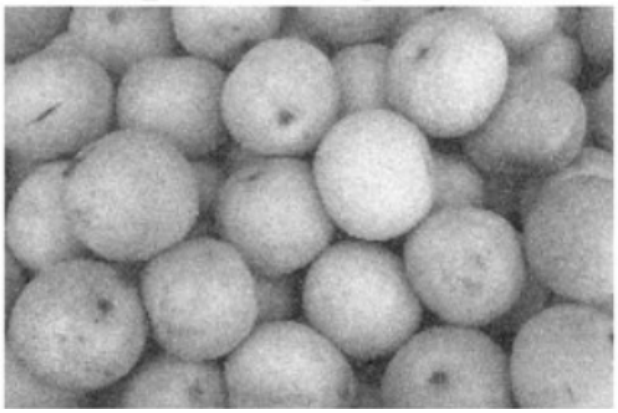
Contra-Harmonic filter

Code:

```
#Contra-Harmonic Mean
import statistics as st
im=Image.open("/content/pi.png")
im=np.array(im)

#Resolution: (732, 486)
#Resolution: (734, 488)
#Adding 2 Rows and 2 Coloumns
im=np.concatenate([[im[0]],im],axis=0)
im=np.concatenate([im,[im[resolution[0]-1]]],axis=0)
im=np.transpose(im)
im=np.concatenate([[im[0]],im],axis=0)
im=np.concatenate([im,[im[resolution[0]-1]]],axis=0)
im=np.transpose(im)

temp = [0]*9
sum_square=0
sum=0

Q=5 #Greater than 0 is better for pepper
```

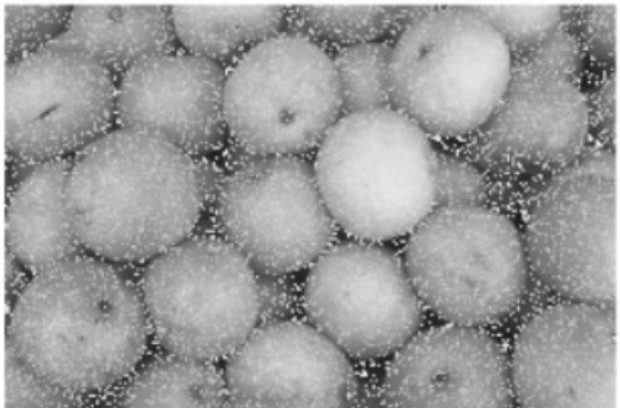
```
#,and less 0 is better for salt noise.

for x in range(0,resolution[0]+1):
 for y in range(0,resolution[1]+1):
   temp[0]=(im[x][y])
   temp[1]=(im[x][y-1])
   temp[2]=(im[x][y+1])
   temp[3]=(im[x-1][y])
   temp[4]=(im[x-1][y-1])
   temp[5]=(im[x-1][y+1])
   temp[6]=(im[x+1][y])
   temp[7]=(im[x+1][y-1])
   temp[8]=(im[x+1][y+1])
   for i in range(0,9):
     sum_square=sum_square+pow(temp[i],Q+1)
     sum=sum+pow(temp[i],Q)
   print(sum_square,sum)
   im[x][y]=sum_square/sum

im=Image.fromarray(im)
im
```

1. 'Gaussian'



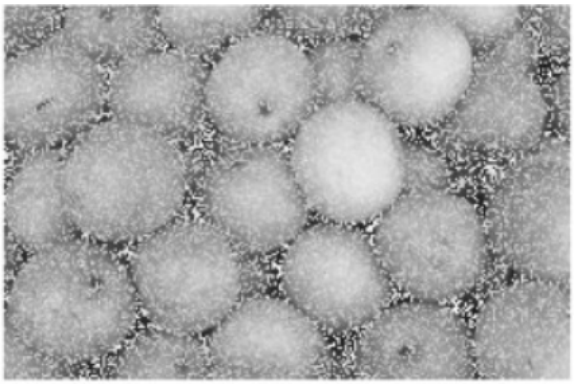
2. 'Poisson'

3. 'salt & pepper



4. 'speckle'

5. Salt-only noise



6. Pepper-only noise

- **Findings:**

A contraharmonic mean filter reduces or virtually eliminates the effects of salt-and-pepper noise. For positive values of Q, the filter eliminates pepper noise. For negative values of Q, it eliminates salt noise.

Geometric filter

Code:

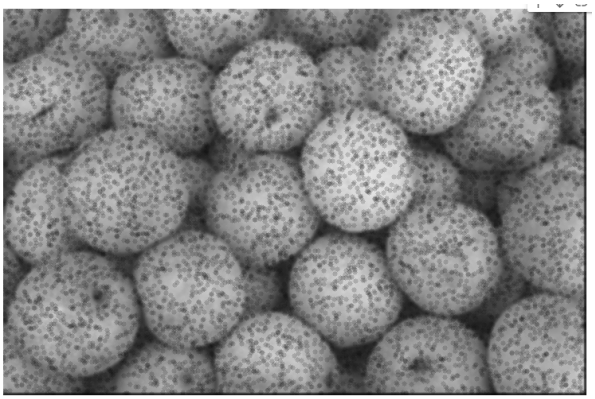
```
#Geometric mean
```

```
im=Image.open("/content/pi.png")
im=np.array(im)

im=geometric_mean(im,disk(3))
im=Image.fromarray(im)
im
```

1. 'Gaussian'

2. 'Poisson'

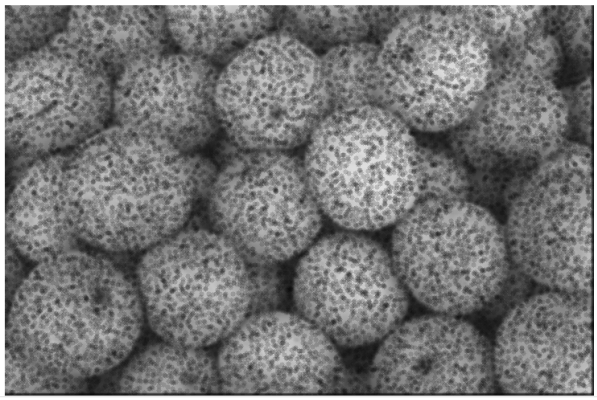
3. 'salt & pepper



4. 'speckle'

5. Salt-only noise

6. Pepper-only noise



- **Findings:**

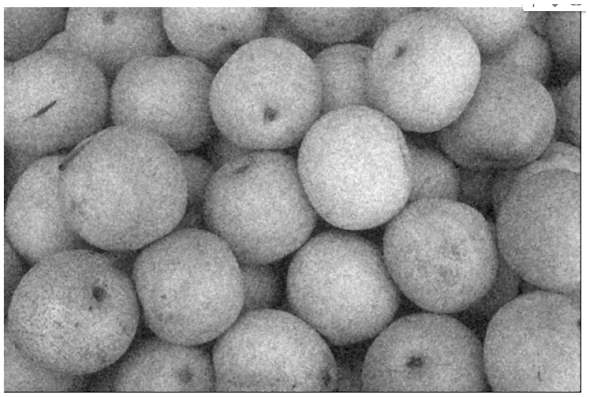
The geometric mean filter is most widely used to filter out Gaussian noise. In general, it will help smooth the image with less data loss than an arithmetic mean filter. It does not work well for pepper noise images, as it has zeros intensity pixels.

Median filter

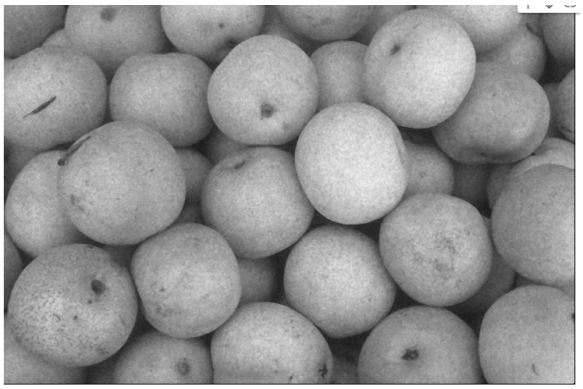
Code:

```
#Median Filter
im = Image.open("/content/pi.png")

im=np.array(im)
im=ndimage.median_filter(im,3)
im=Image.fromarray(im)
im
```

1. 'Gaussian'

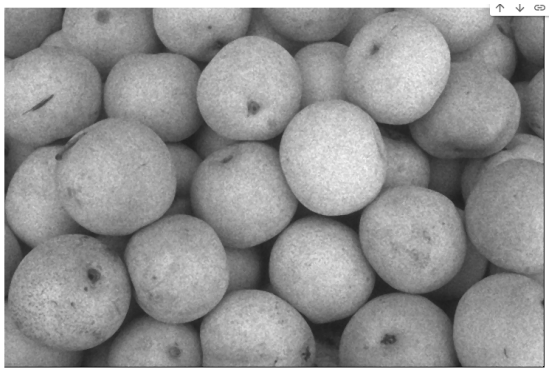


2. 'Poisson'



3. 'salt & pepper

4. 'speckle'



5. Salt-only noise

6. Pepper-only noise

- **Findings:**

The median filter is one type of non-linear filter. It is very effective at removing impulse noise, the “salt and pepper” noise, in the image. It works mostly well for removing all types of noises.