

DIP Assignment # 2

Section: C

Members:

1. Asad Haroon **SP17-BCS-012**
2. Fahad Hussain **SP17-BCS-015**
3. Rasib Nadeem **SP17-BCS-141**

Question:

1. Prepare a dataset of 20 images
2. Add different type of Noise in the data.
3. Apply different noise removal techniques
4. Compare their results and discuss which method perform better

Note: I have showed one Dummy image plot in every plot because I have to balance all filter images side by side and their arrangement only.

Import Libraries

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import cv2
import math
from scipy.misc import imread
from skimage.util import random_noise
from skimage.filters import rank
from scipy import ndimage
import scipy.ndimage.filters as filters
import scipy.ndimage.morphology as morphology
from scipy.signal import tools import wiener
```

1. Dataset

You can download faces dataset from link: https://drive.google.com/drive/folders/13XqlMr4SPWffMxL4c_rDDXm-uzzTdGR5?usp=sharing
[\(https://drive.google.com/drive/folders/13XqlMr4SPWffMxL4c_rDDXm-uzzTdGR5?usp=sharing\)](https://drive.google.com/drive/folders/13XqlMr4SPWffMxL4c_rDDXm-uzzTdGR5?usp=sharing)

In [2]: #faces dataset

```
img1=imread("dataset\\1.jpg",False,'L')
img2=imread("dataset\\5.jpg",False,'L')
img3=imread("dataset\\10.jpg",False,'L')
img4=imread("dataset\\20.jpg",False,'L')
img5=imread("dataset\\21.jpg",False,'L')
img6=imread("dataset\\16.jpg",False,'L')
fig, axes = plt.subplots(3, 2, figsize=(15,15))
ax = axes.ravel()

ax[0].imshow(img1,plt.cm.gray)
ax[0].set_title("Image 1")
ax[1].imshow(img2,plt.cm.gray)
ax[1].set_title("Image 2")
ax[2].imshow(img3,plt.cm.gray)
ax[2].set_title("Image 3")
ax[3].imshow(img4,plt.cm.gray)
ax[3].set_title("Image 4")
ax[4].imshow(img5,plt.cm.gray)
ax[4].set_title("Image 5")
ax[5].imshow(img6,plt.cm.gray)
ax[5].set_title("Image 6")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

This is separate from the ipykernel package so we can avoid doing imports until

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

after removing the cwd from sys.path.

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
```

```
Use ``imageio.imread`` instead.
```

```
"""
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: DeprecationWarning: `imread` is deprecated!
```

```
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
```

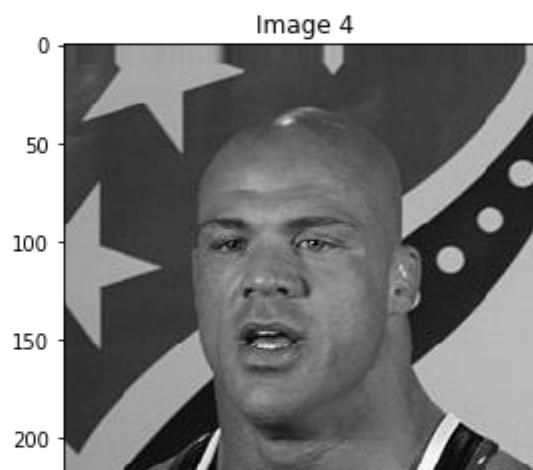
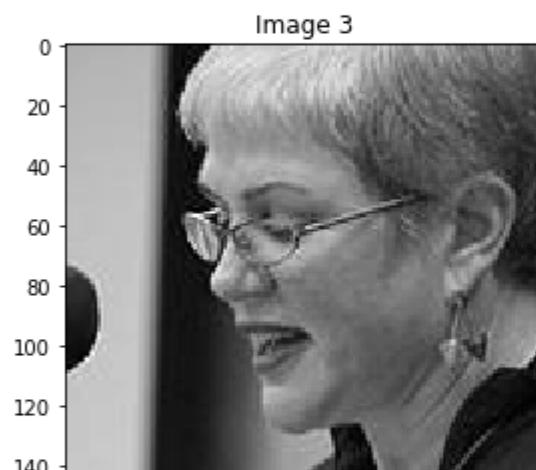
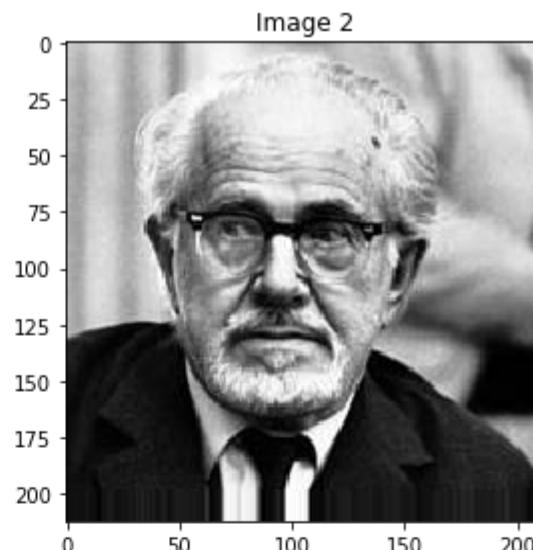
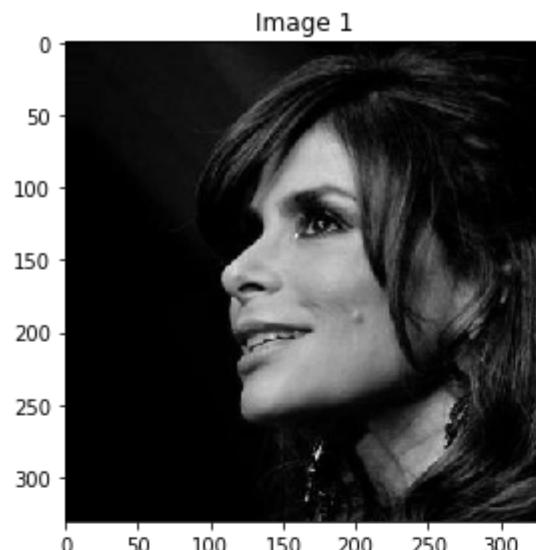
```
Use ``imageio.imread`` instead.
```

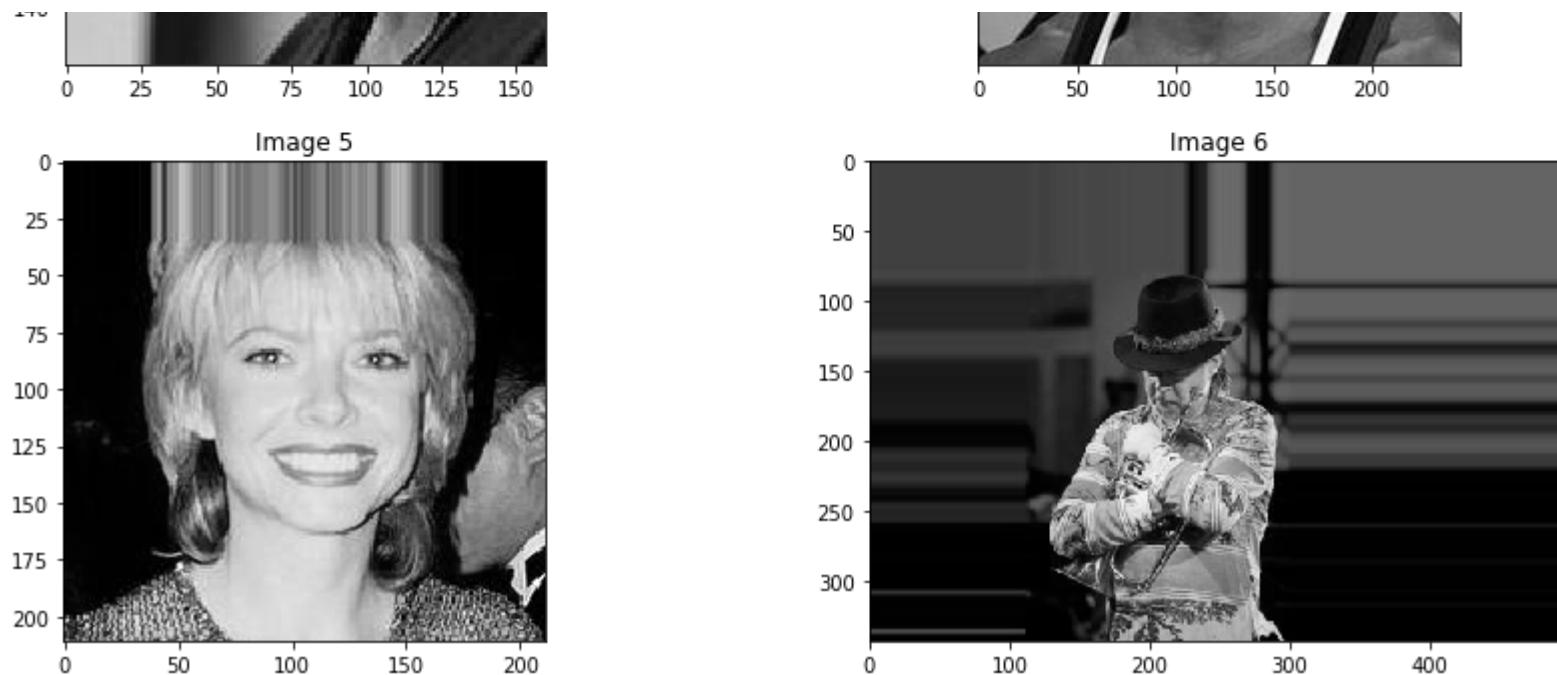
```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: DeprecationWarning: `imread` is deprecated!
```

```
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
```

```
Use ``imageio.imread`` instead.
```

```
import sys
```



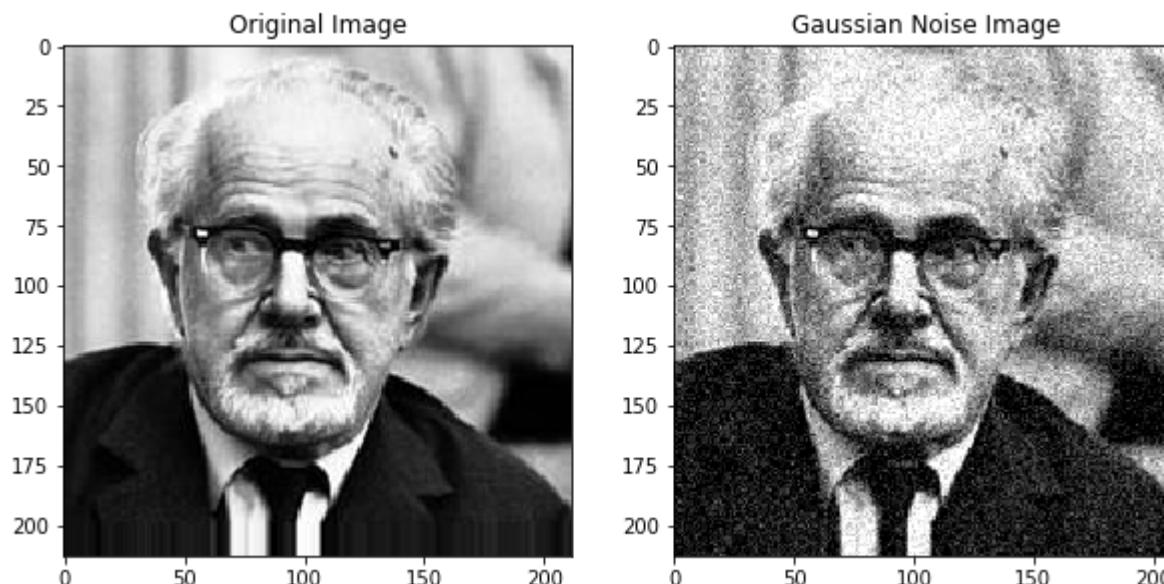


2. Adding Noise in Data

2.1. Gaussian Noise

```
In [3]: gaussian_noise_img=random_noise(img2, mode='gaussian')
fig, axes = plt.subplots(1, 2, figsize=(10,10))
ax = axes.ravel()

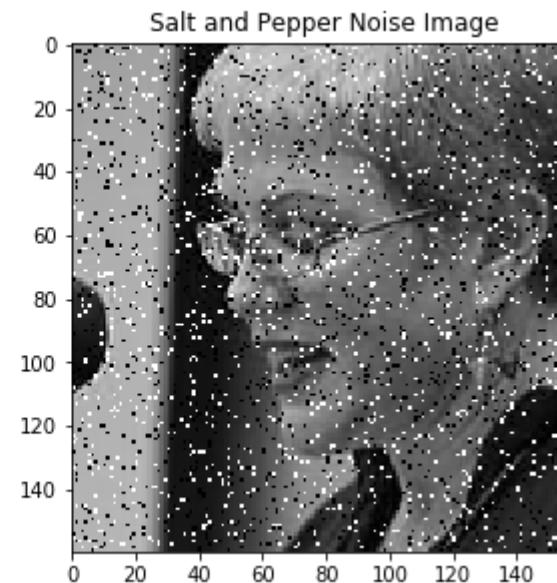
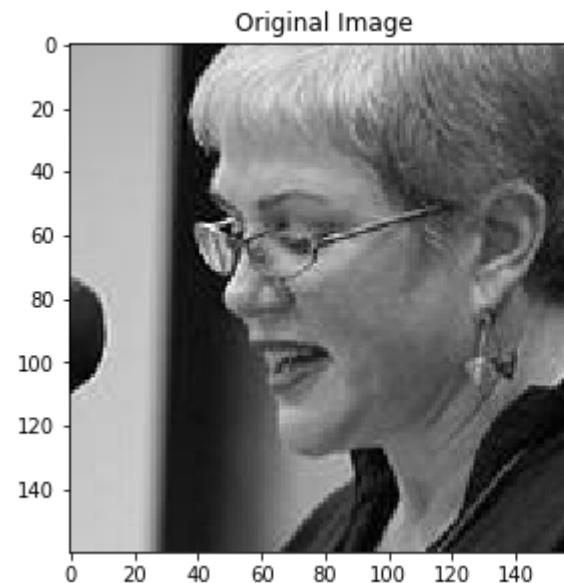
ax[0].imshow(img2,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(gaussian_noise_img,plt.cm.gray)
ax[1].set_title("Gaussian Noise Image")
plt.show()
```



2.2. Salt and Pepper Noise

```
In [4]: import random
p=0.05
thres=1-p
sp=img3.copy()
output_image=np.zeros(sp.shape,np.uint8)
rows,cols=sp.shape
for i in range(rows):
    for j in range(cols):
        r=random.random()
        if r<p:
            output_image[i][j]=0
        elif r>thres:
            output_image[i][j]=255
        else:
            output_image[i][j]=sp[i][j]
salt_and_pepper=output_image.copy()
fig, axes = plt.subplots(1, 2, figsize=(10,10))
ax = axes.ravel()

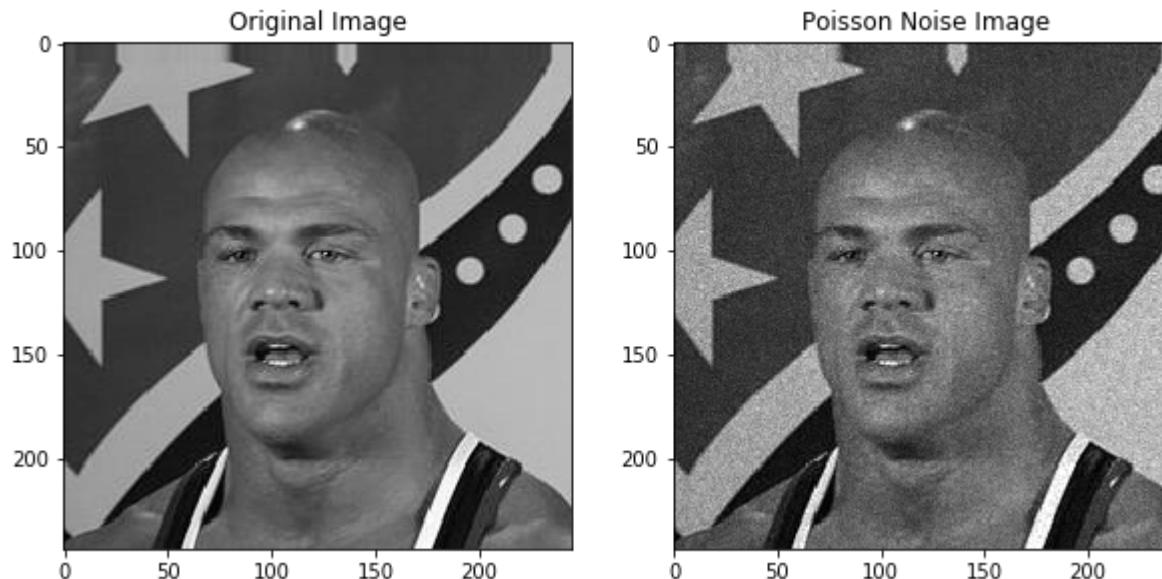
ax[0].imshow(img3,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(salt_and_pepper,plt.cm.gray)
ax[1].set_title("Salt and Pepper Noise Image")
plt.show()
```



2.3. Poisson Noise

```
In [5]: poisson_noise=random_noise(img4, mode='poisson',seed=2)
fig, axes = plt.subplots(1, 2,figsize=(10,10))
ax = axes.ravel()

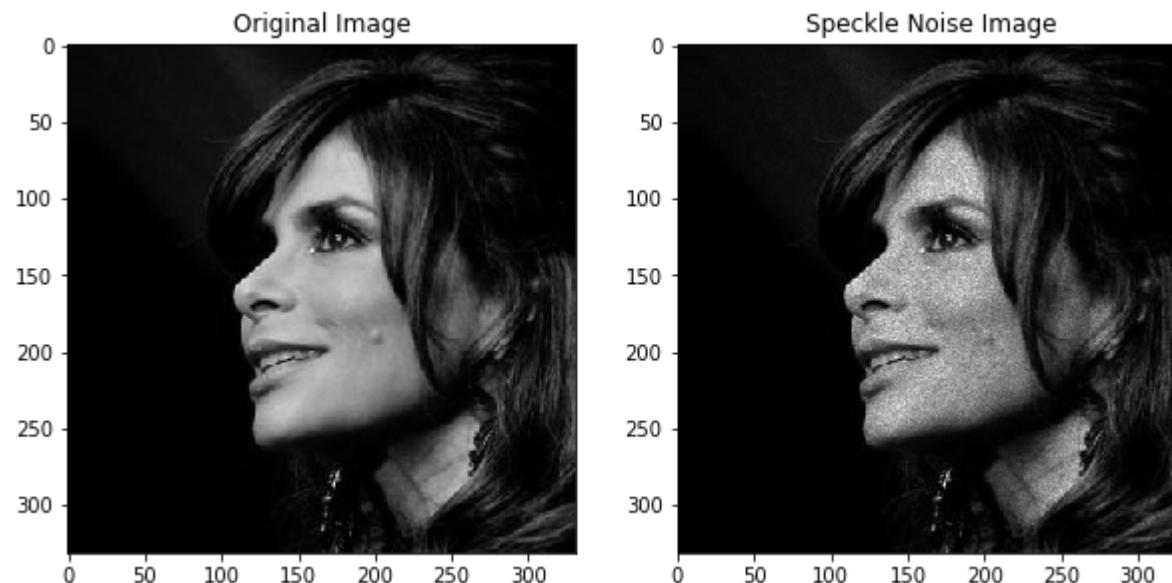
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(poisson_noise,plt.cm.gray)
ax[1].set_title("Poisson Noise Image")
plt.show()
```



2.4. Speckle Noise

```
In [6]: speckle_noise=random_noise(img1, mode='speckle',seed=2)
fig, axes = plt.subplots(1, 2,figsize=(10,10))
ax = axes.ravel()

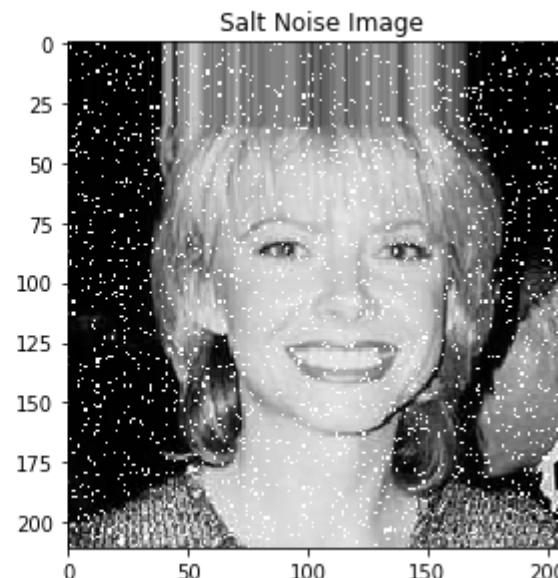
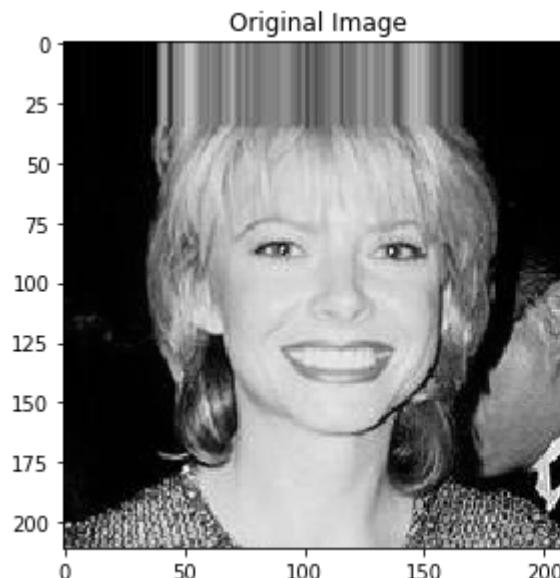
ax[0].imshow(img1,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(speckle_noise,plt.cm.gray)
ax[1].set_title("Speckle Noise Image")
plt.show()
```



2.5. Salt Noise

```
In [7]: salt_noise=random_noise(img5, mode='salt',seed=2)
fig, axes = plt.subplots(1, 2,figsize=(10,10))
ax = axes.ravel()

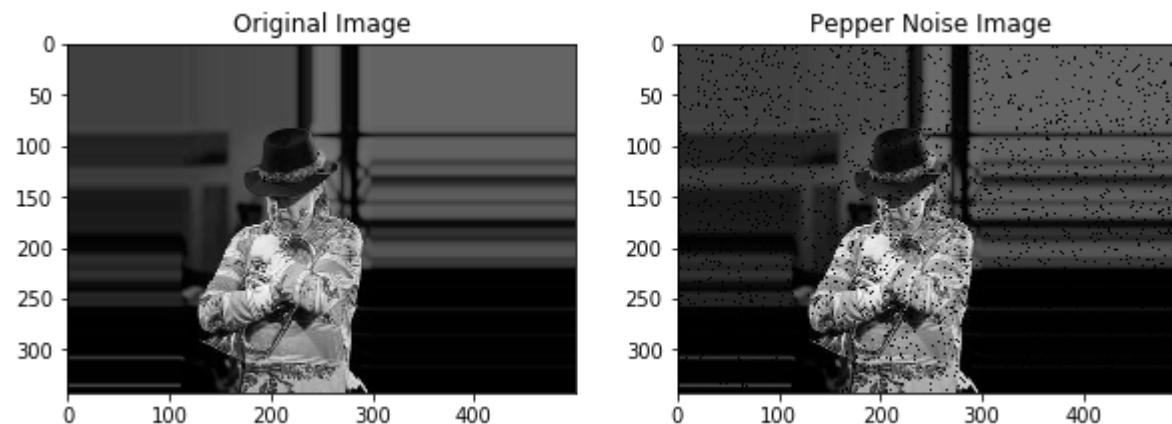
ax[0].imshow(img5,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(salt_noise,plt.cm.gray)
ax[1].set_title("Salt Noise Image")
plt.show()
```



2.6. Pepper Noise

```
In [8]: pepper_noise=random_noise(img6, mode='pepper',seed=2)
fig, axes = plt.subplots(1, 2,figsize=(10,10))
ax = axes.ravel()

ax[0].imshow(img6,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(pepper_noise,plt.cm.gray)
ax[1].set_title("Pepper Noise Image")
plt.show()
```



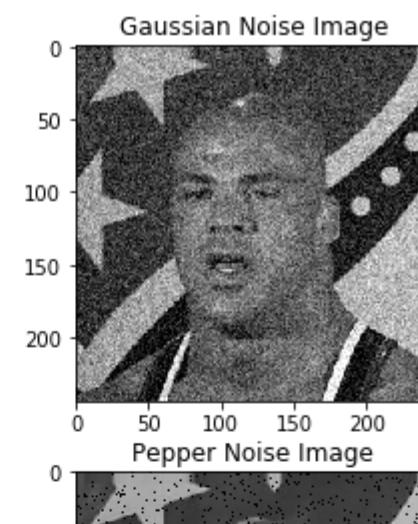
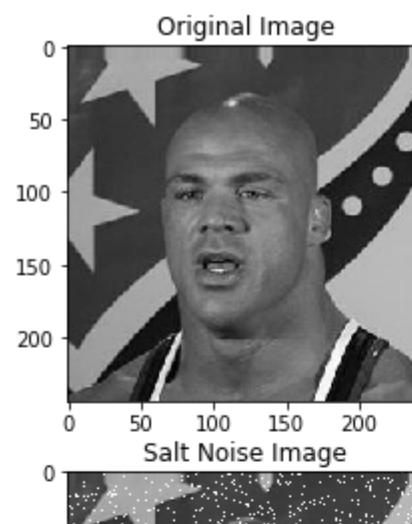
2.7. Comparing all Noise together

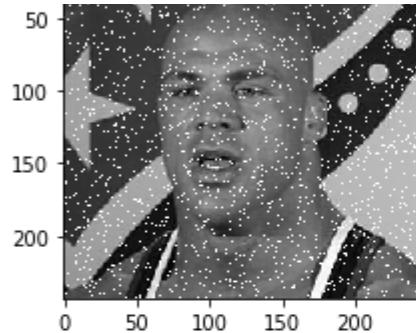
In [9]: *#Applying All Noise*

```
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

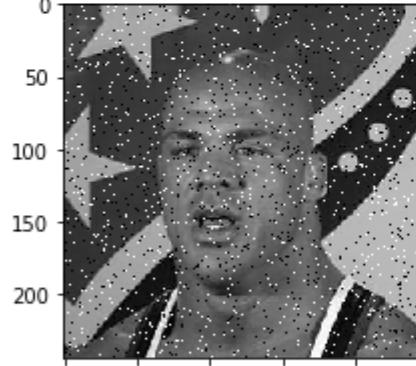
fig, axes = plt.subplots(4, 2,figsize=(15,15))
ax = axes.ravel()

ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(gaussian_noise,plt.cm.gray)
ax[1].set_title("Gaussian Noise Image")
ax[2].imshow(salt_noise,plt.cm.gray)
ax[2].set_title("Salt Noise Image")
ax[3].imshow(pepper_noise,plt.cm.gray)
ax[3].set_title("Pepper Noise Image")
ax[4].imshow(salt_p_noise_img,plt.cm.gray)
ax[4].set_title("Salt and Pepper Noise Image")
ax[5].imshow(poisson_noise,plt.cm.gray)
ax[5].set_title("Poisson Noise Image")
ax[6].imshow(speckle_noise,plt.cm.gray)
ax[6].set_title("Speckle Noise Image")
plt.show()
```

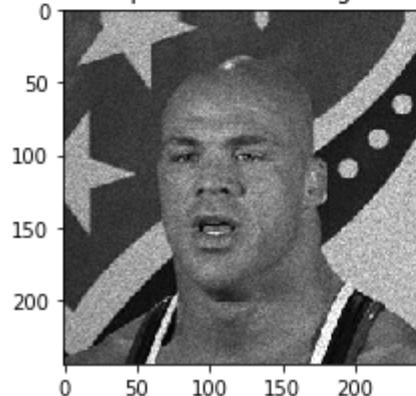




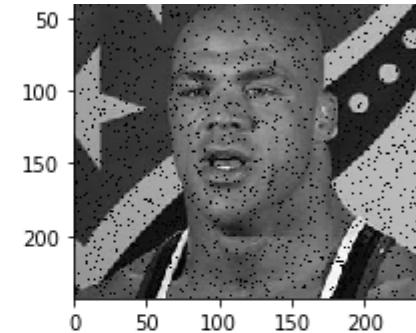
Salt and Pepper Noise Image



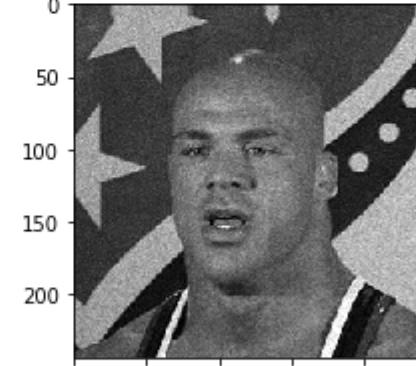
Poisson Noise Image



Speckle Noise Image

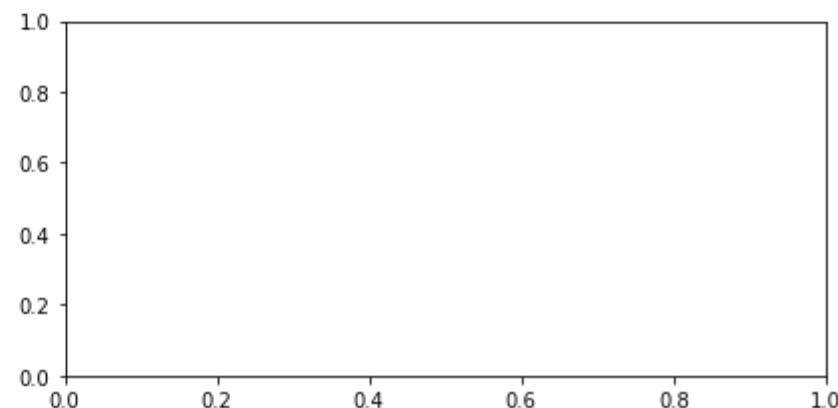


Salt and Pepper Noise Image



Poisson Noise Image

Speckle Noise Image



3. Applying Noise Removal Techniques

3.1. Median Filter

```
In [10]: def findMedian(x):
    a=np.median(x)
    return a

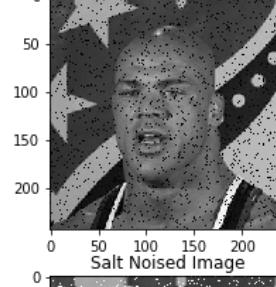
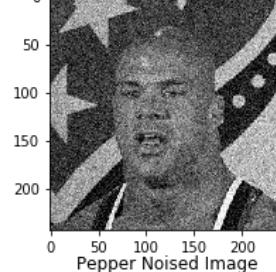
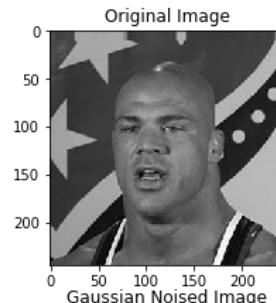
def applyMedianFilter(img):
    rows,cols=img.shape
    for i in range(1,rows-1):
        for j in range(1,cols-1):
            ans=img[i-1:i+2,j-1:j+2]
            ans=findMedian(ans)
            img[i,j]=ans
    return img

#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

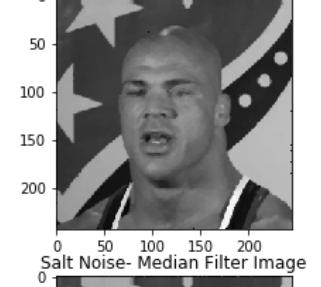
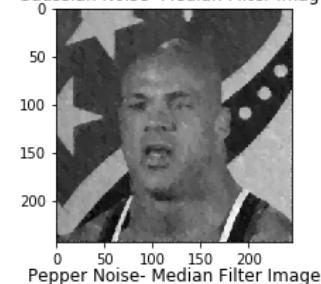
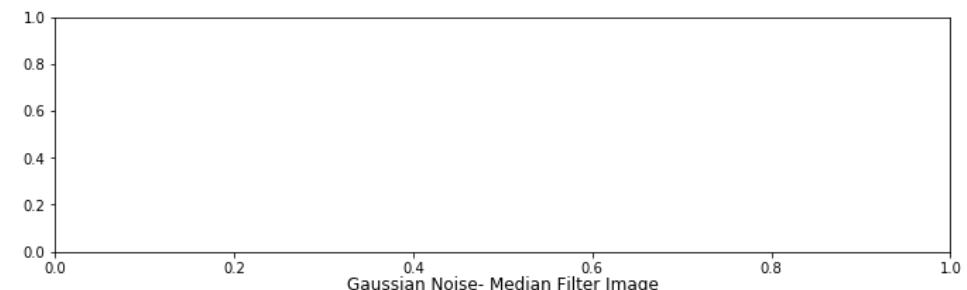
#applying median filter on noises
gauss_median=applyMedianFilter(gaussian_noise.copy())
pepper_median=applyMedianFilter(pepper_noise.copy())
salt_median=applyMedianFilter(salt_noise.copy())
speckle_median=applyMedianFilter(speckle_noise.copy())
poisson_median=applyMedianFilter(poisson_noise.copy())
salt_pepper_median=applyMedianFilter(salt_p_noise_img.copy())

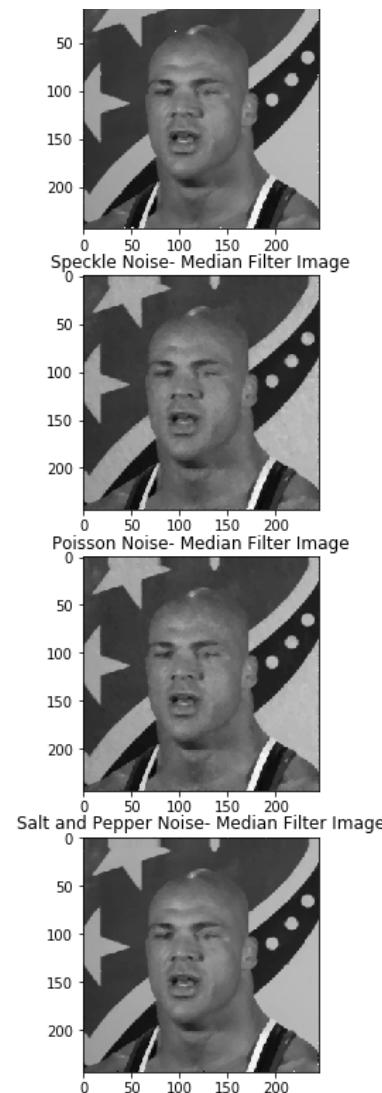
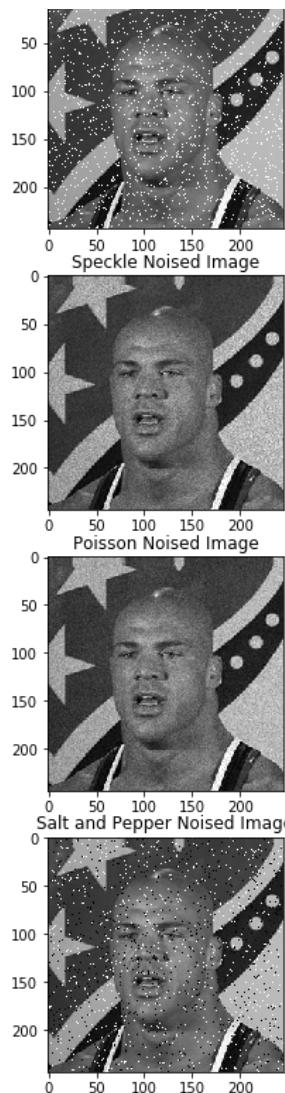
#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_median,plt.cm.gray)
ax[3].set_title("Gaussian Noise- Median Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_median,plt.cm.gray)
ax[5].set_title("Pepper Noise- Median Filter Image")
```

```
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_median,plt.cm.gray)
ax[7].set_title("Salt Noise- Median Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_median,plt.cm.gray)
ax[9].set_title("Speckle Noise- Median Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_median,plt.cm.gray)
ax[11].set_title("Poisson Noise- Median Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_median,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- Median Filter Image")
plt.show()
```



localhost:8888/notebooks/DIP Assignment %23 2 (Section C)(SP17-BCS-012%2C015%2C141).ipynb





3.2. Arithmetic Mean Filter

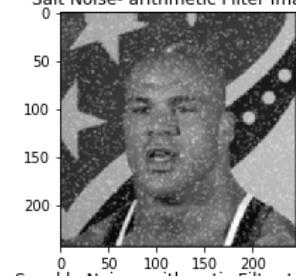
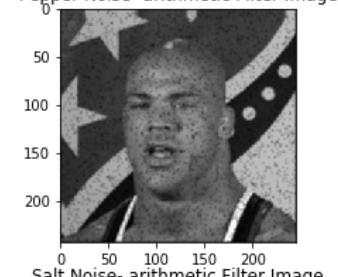
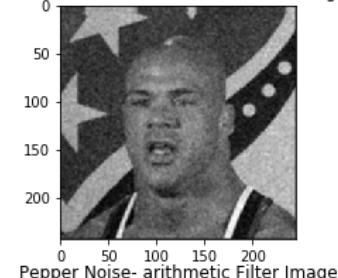
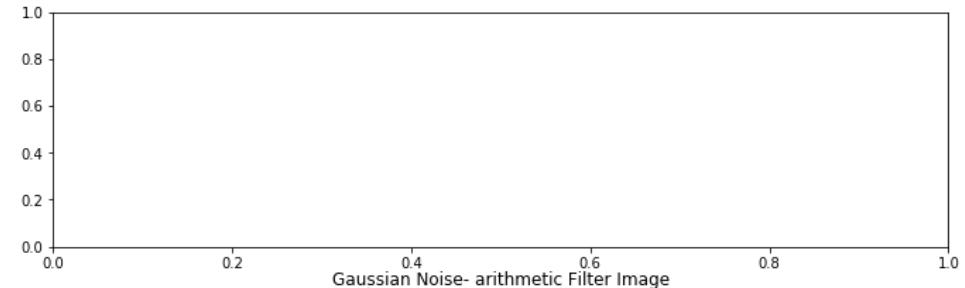
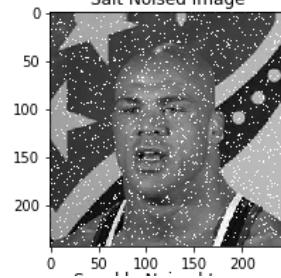
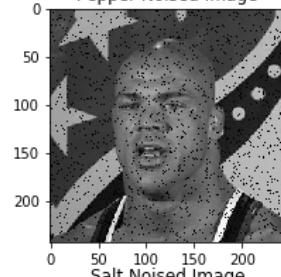
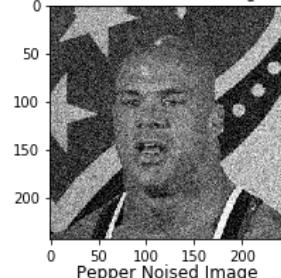
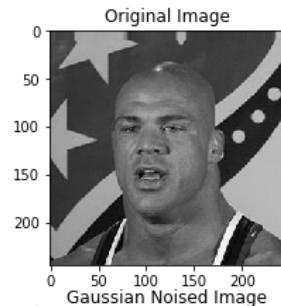
```
In [11]: def arithmetic_mean_filter(x):
    kernel = 1/9*np.ones((3,3))
    img=ndimage.convolve(x, kernel, mode='constant', cval=0.0)
    return img

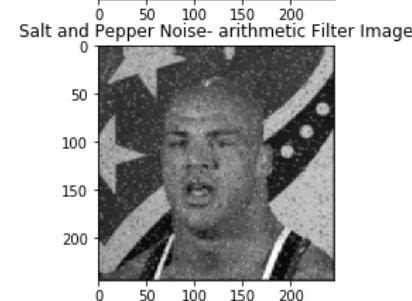
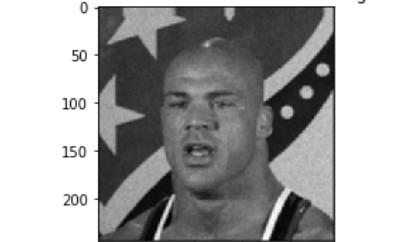
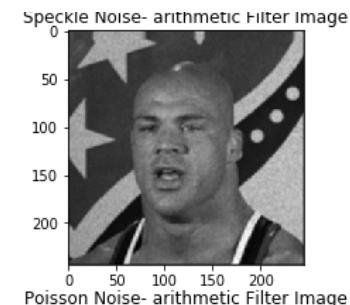
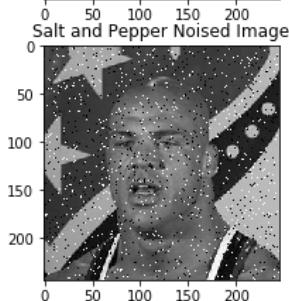
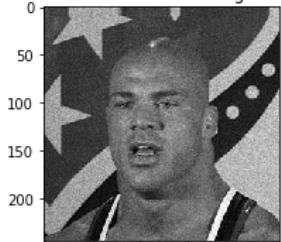
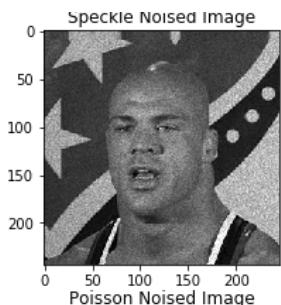
#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

#applying arithmetic mean filter on noises
gauss_arithmetic=arithmetic_mean_filter(gaussian_noise.copy())
pepper_arithmetic=arithmetic_mean_filter(pepper_noise.copy())
salt_arithmetic=arithmetic_mean_filter(salt_noise.copy())
speckle_arithmetic=arithmetic_mean_filter(speckle_noise.copy())
poisson_arithmetic=arithmetic_mean_filter(poisson_noise.copy())
salt_pepper_arithmetic=arithmetic_mean_filter(salt_p_noise_img.copy())

#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_arithmetic,plt.cm.gray)
ax[3].set_title("Gaussian Noise- arithmetic Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_arithmetic,plt.cm.gray)
ax[5].set_title("Pepper Noise- arithmetic Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_arithmetic,plt.cm.gray)
ax[7].set_title("Salt Noise- arithmetic Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_arithmetic,plt.cm.gray)
ax[9].set_title("Speckle Noise- arithmetic Filter Image")
```

```
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_arithmatic,plt.cm.gray)
ax[11].set_title("Poisson Noise- arithmetic Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_arithmatic,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- arithmetic Filter Image")
plt.show()
```





3.3. Geometric Mean Filter

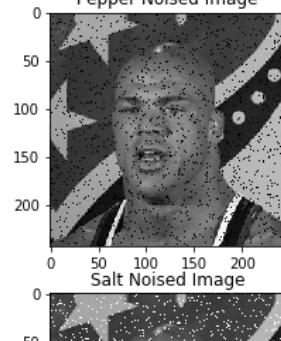
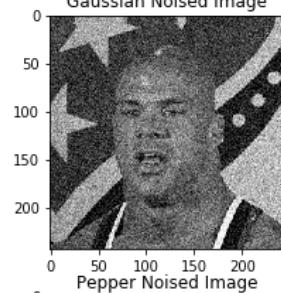
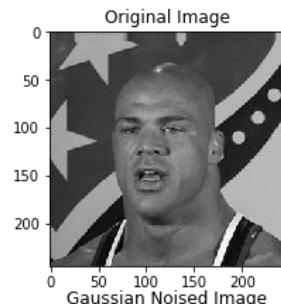
```
In [12]: def geometric_mean_filter(x):
    x=x*255
    rows,cols=x.shape
    img_geo=np.zeros((rows,cols))
    for i in range(1,rows-1):
        for j in range(1,cols-1):
            ans=x[i-1:i+2,j-1:j+2]
            ans=round(pow(np.prod(ans),1/9))
            img_geo[i,j]=ans
    return img_geo

#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

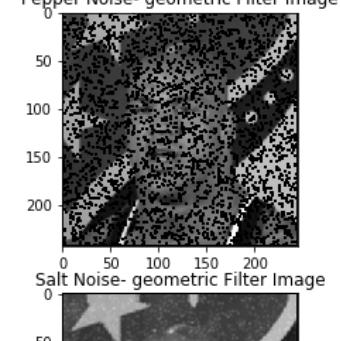
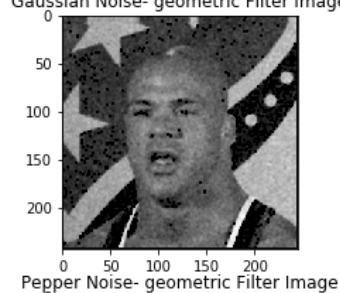
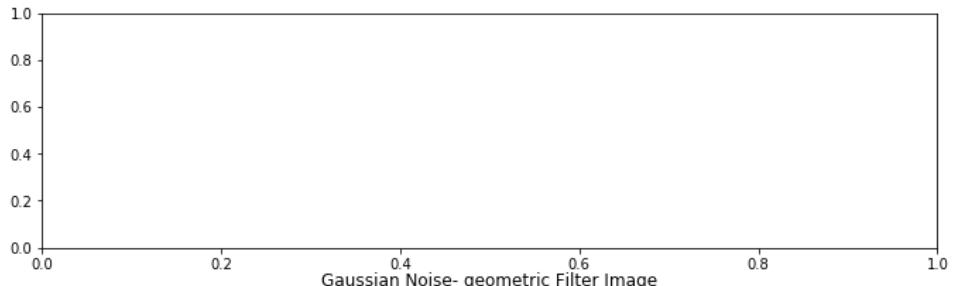
#applying geometric mean filter on noises
gauss_geometric=geometric_mean_filter(gaussian_noise.copy())
pepper_geometric=geometric_mean_filter(pepper_noise.copy())
salt_geometric=geometric_mean_filter(salt_noise.copy())
speckle_geometric=geometric_mean_filter(speckle_noise.copy())
poisson_geometric=geometric_mean_filter(poisson_noise.copy())
salt_pepper_geometric=geometric_mean_filter(salt_p_noise_img.copy())

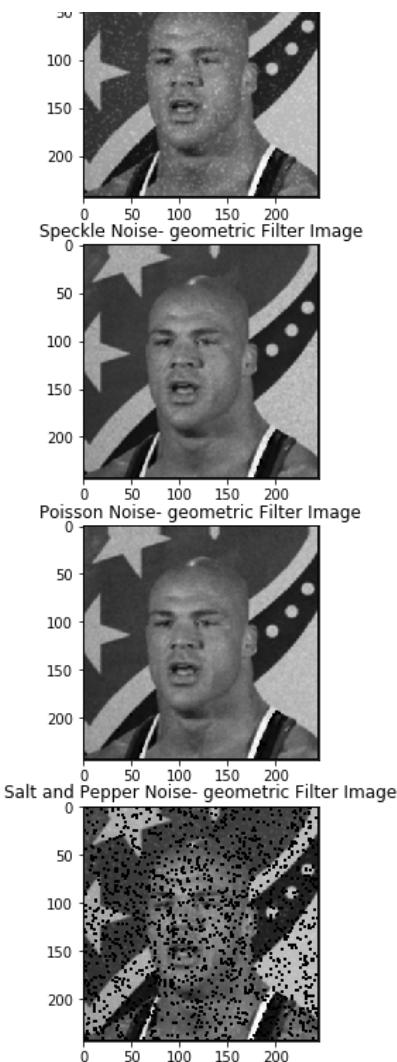
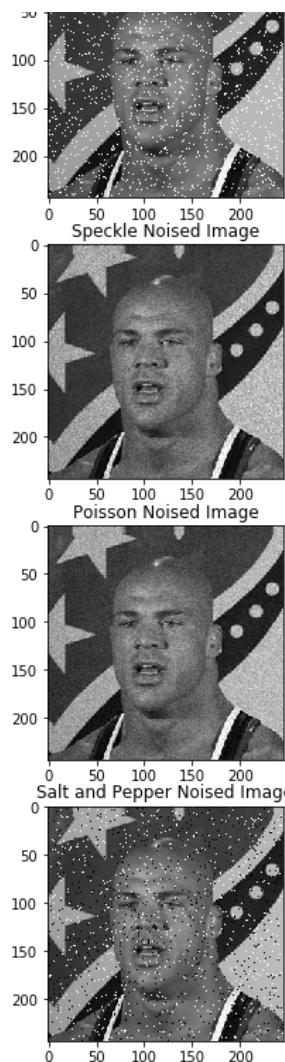
#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_geometric,plt.cm.gray)
ax[3].set_title("Gaussian Noise- geometric Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_geometric,plt.cm.gray)
ax[5].set_title("Pepper Noise- geometric Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
```

```
ax[7].imshow(salt_geometric,plt.cm.gray)
ax[7].set_title("Salt Noise- geometric Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_geometric,plt.cm.gray)
ax[9].set_title("Speckle Noise- geometric Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_geometric,plt.cm.gray)
ax[11].set_title("Poisson Noise- geometric Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_geometric,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- geometric Filter Image")
plt.show()
```



Salt Noised Image





3.4. Harmonic Mean Filter

```
In [13]: def harmonic_mean_filter(x):
    rows,cols=x.shape
    x=x*255
    img_harmonic=np.zeros((rows,cols))
    for i in range(1,rows-1):
        for j in range(1,cols-1):
            ans=x[i-1:i+2,j-1:j+2]
            ans=9/np.sum(1/ans)
            ans=round(ans)
            img_harmonic[i,j]=ans
    return img_harmonic

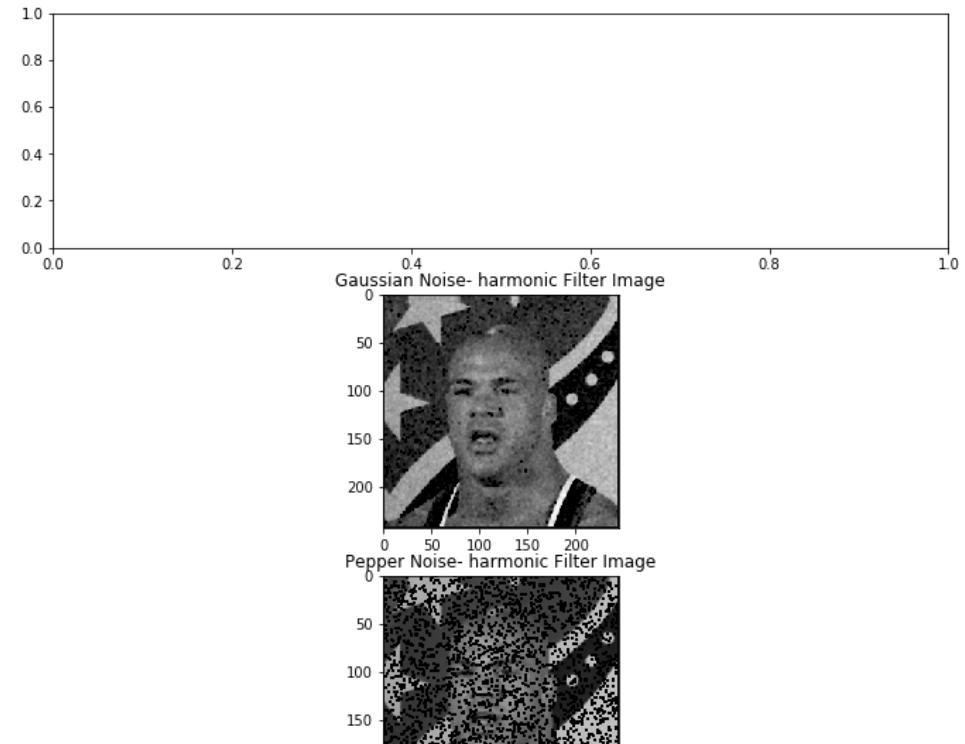
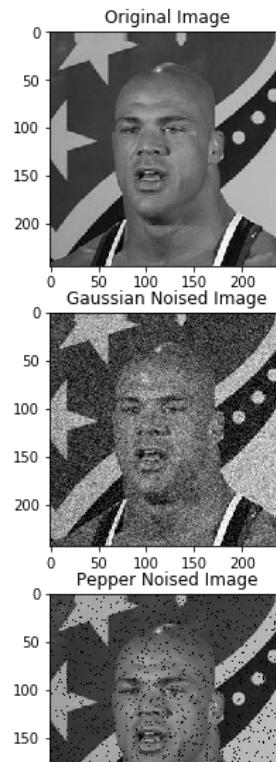
#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

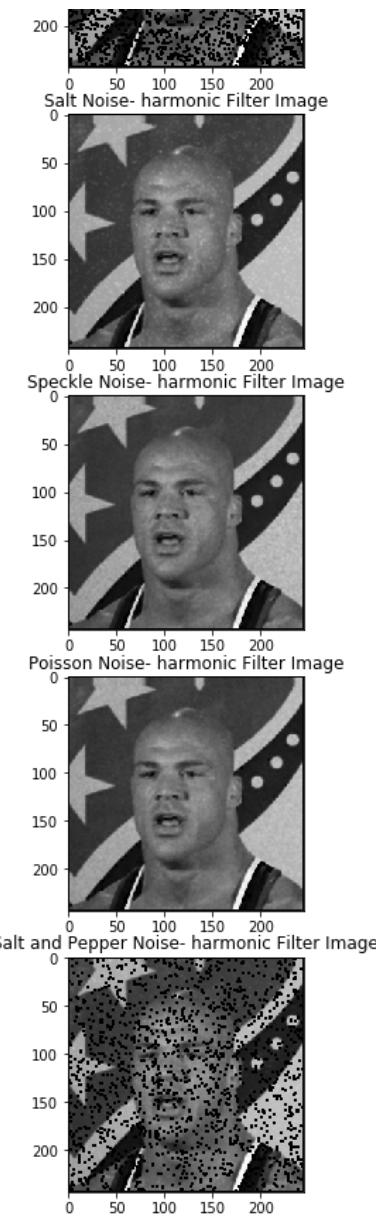
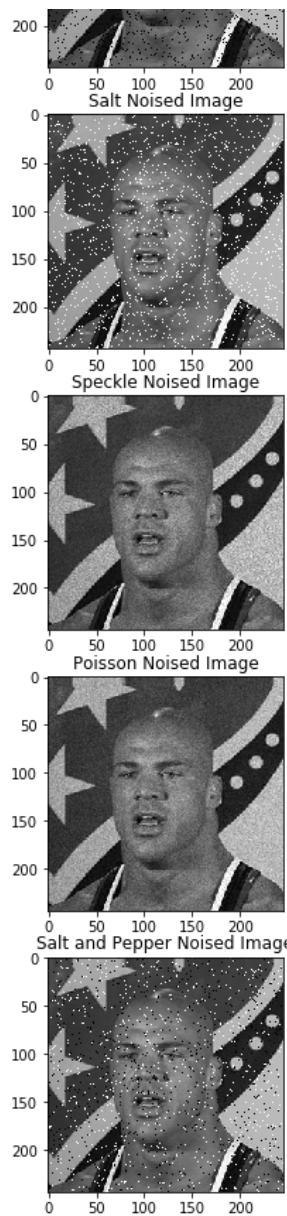
#applying harmonic filter on noises
gauss_harmonic=harmonic_mean_filter(gaussian_noise.copy())
pepper_harmonic=harmonic_mean_filter(pepper_noise.copy())
salt_harmonic=harmonic_mean_filter(salt_noise.copy())
speckle_harmonic=harmonic_mean_filter(speckle_noise.copy())
poisson_harmonic=harmonic_mean_filter(poisson_noise.copy())
salt_pepper_harmonic=harmonic_mean_filter(salt_p_noise_img.copy())

#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_harmonic,plt.cm.gray)
ax[3].set_title("Gaussian Noise- harmonic Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_harmonic,plt.cm.gray)
ax[5].set_title("Pepper Noise- harmonic Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
```

```
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_harmonic,plt.cm.gray)
ax[7].set_title("Salt Noise- harmonic Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_harmonic,plt.cm.gray)
ax[9].set_title("Speckle Noise- harmonic Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_harmonic,plt.cm.gray)
ax[11].set_title("Poisson Noise- harmonic Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_harmonic,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- harmonic Filter Image")
plt.show()
```

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: RuntimeWarning: divide by zero encountered in true_divide





3.5. Contra Harmonic Mean Filter

```
In [14]: def contra_harmonic(Q=0,img=[]):
    rows,cols=img.shape[:2]
    img=img*255
    img_contra_harmonic=np.zeros((rows,cols))
    for i in range(1,rows-1):
        for j in range(1,cols-1):
            ans=img[i-1:i+2,j-1:j+2]
            numerator=ans**(Q+1)
            if Q== -1:
                denominator=1/ans
            else:
                denominator=ans**(Q)
            ans1=np.sum(numerator)
            ans2=np.sum(denominator)
            ans3=ans1/ans2
            ans=round(ans3)
            img_contra_harmonic[i,j]=ans
    return img_contra_harmonic

#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

#applying contra harmonic filter on noises
gauss_contra_harmonic=contra_harmonic(Q=0,img=gaussian_noise.copy())
pepper_contra_harmonic=contra_harmonic(Q=1,img=pepper_noise.copy())
salt_contra_harmonic=contra_harmonic(Q=-1,img=salt_noise.copy())
speckle_contra_harmonic=contra_harmonic(Q=-1,img=speckle_noise.copy())
poisson_contra_harmonic=contra_harmonic(Q=-1,img=poisson_noise.copy())
salt_pepper_contra_harmonic=contra_harmonic(Q=0.5,img=salt_p_noise_img.copy())

#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
```

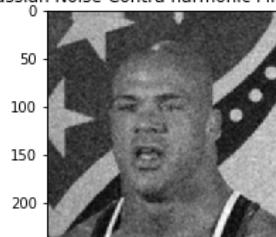
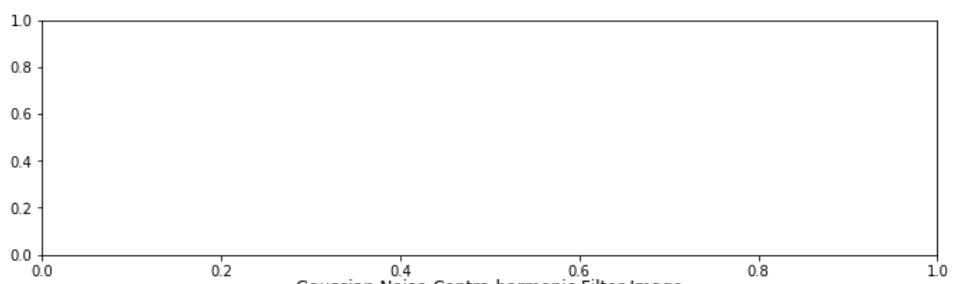
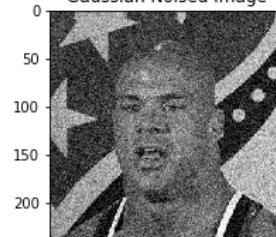
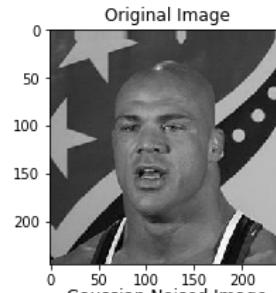
```

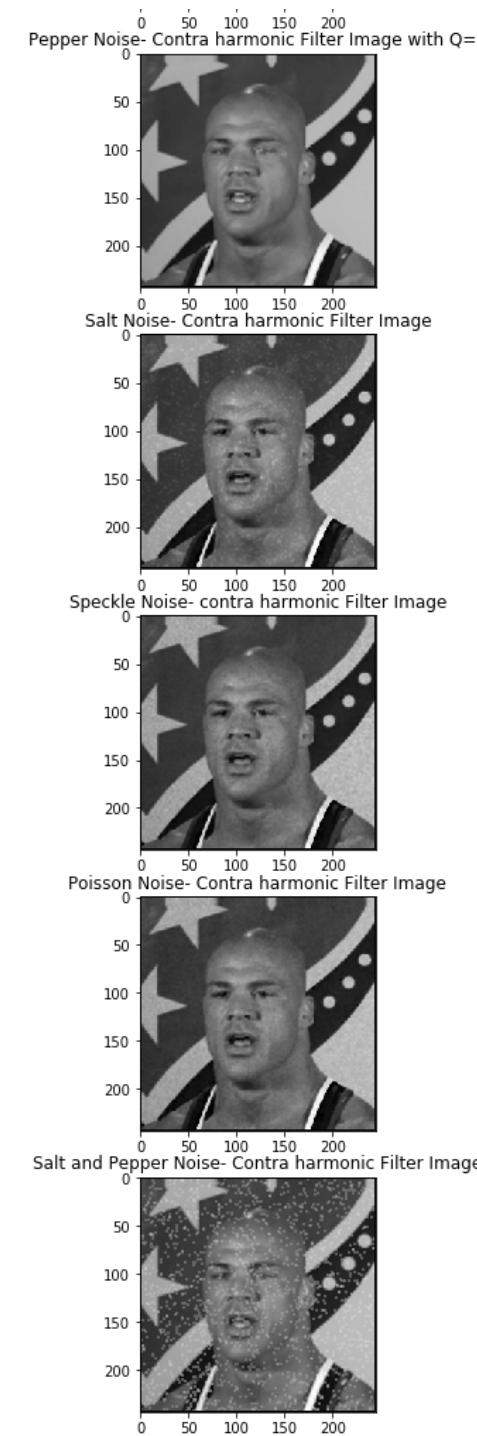
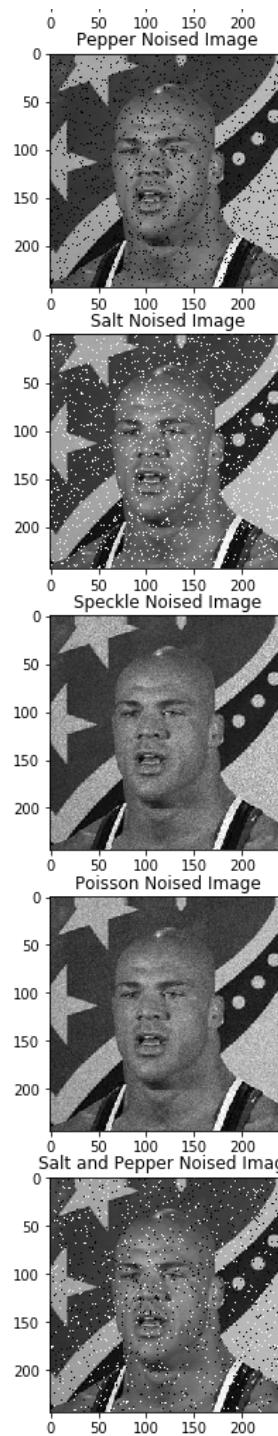
ax[3].imshow(gauss_contra_harmonic,plt.cm.gray)
ax[3].set_title("Gaussian Noise-Contra harmonic Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_contra_harmonic,plt.cm.gray)
ax[5].set_title("Pepper Noise- Contra harmonic Filter Image with Q=1")
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_contra_harmonic,plt.cm.gray)
ax[7].set_title("Salt Noise- Contra harmonic Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_contra_harmonic,plt.cm.gray)
ax[9].set_title("Speckle Noise- contra harmonic Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_contra_harmonic,plt.cm.gray)
ax[11].set_title("Poisson Noise- Contra harmonic Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_contra_harmonic,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- Contra harmonic Filter Image")
plt.show()

```

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: RuntimeWarning: divide by zero encountered in true_divide

Remove the CWD from sys.path while we load stuff.





3.6. Min Filter

In [15]:

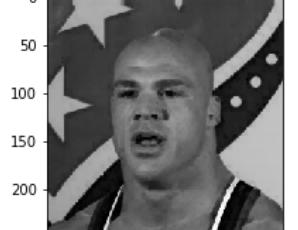
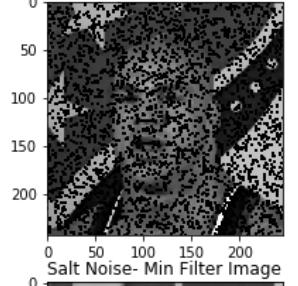
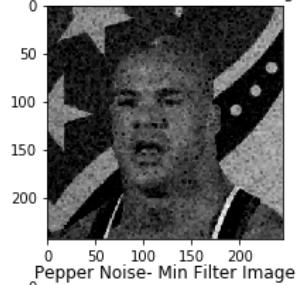
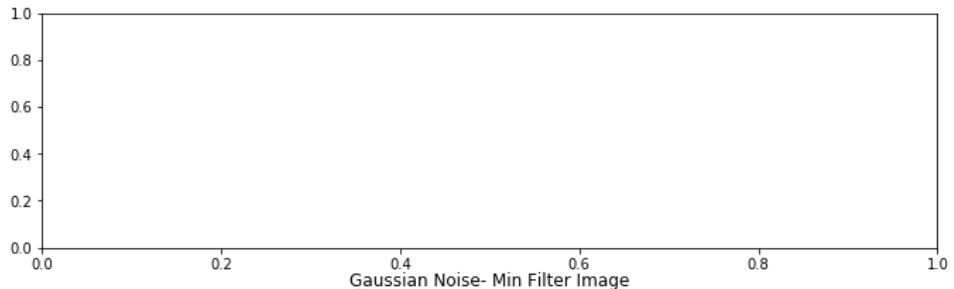
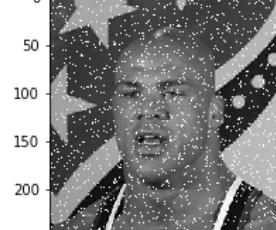
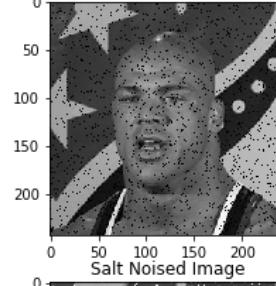
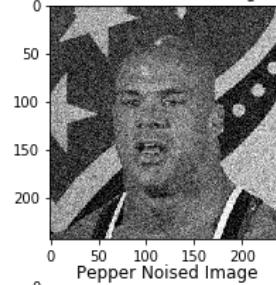
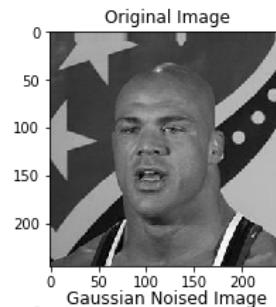
```
def min_filter(img):
    neighborhood = morphology.generate_binary_structure(len(img.shape),2)
    i=filters.minimum_filter(img, footprint=neighborhood)
    return i

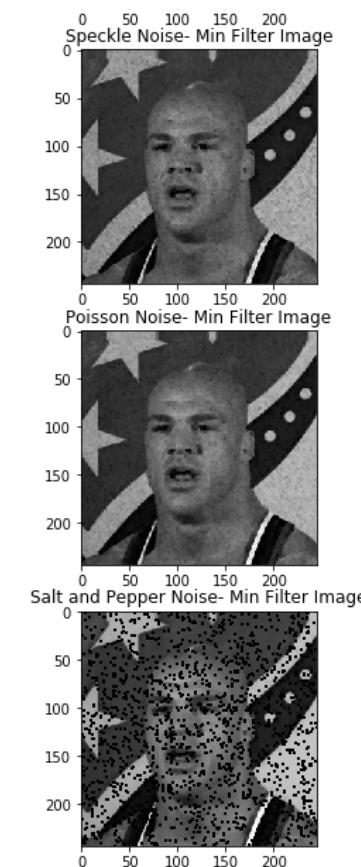
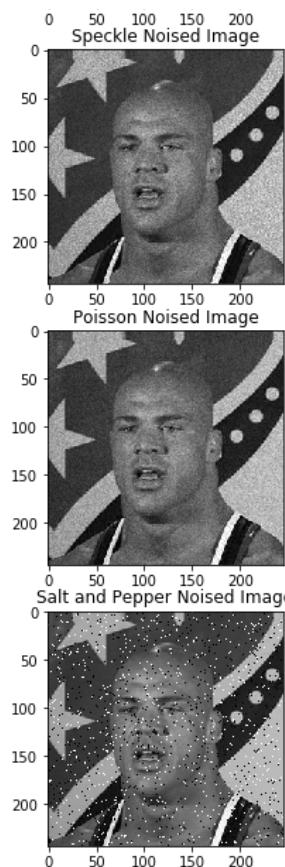
#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

#applying min filter on noises
gauss_min=min_filter(gaussian_noise.copy())
pepper_min=min_filter(pepper_noise.copy())
salt_min=min_filter(salt_noise.copy())
speckle_min=min_filter(speckle_noise.copy())
poisson_min=min_filter(poisson_noise.copy())
salt_pepper_min=min_filter(salt_p_noise_img.copy())

#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_min,plt.cm.gray)
ax[3].set_title("Gaussian Noise- Min Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_min,plt.cm.gray)
ax[5].set_title("Pepper Noise- Min Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_min,plt.cm.gray)
ax[7].set_title("Salt Noise- Min Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_min,plt.cm.gray)
```

```
ax[9].set_title("Speckle Noise- Min Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_min,plt.cm.gray)
ax[11].set_title("Poisson Noise- Min Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_min,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- Min Filter Image")
plt.show()
```





3.7. Max Filter

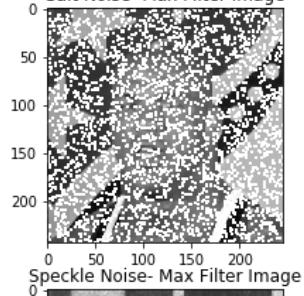
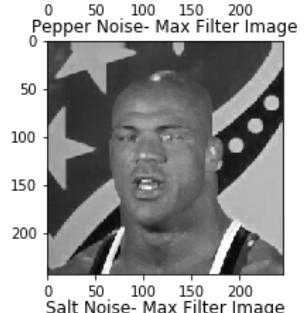
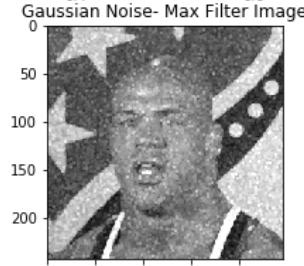
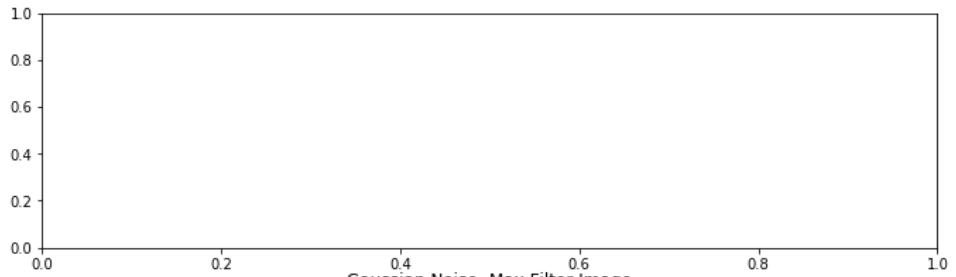
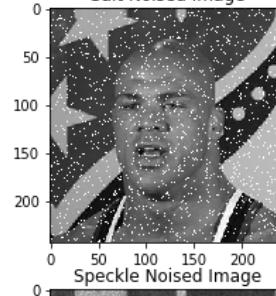
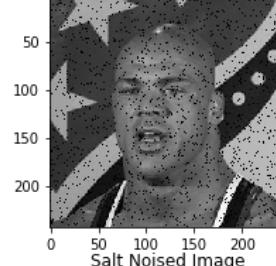
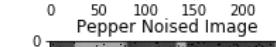
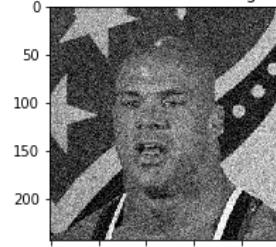
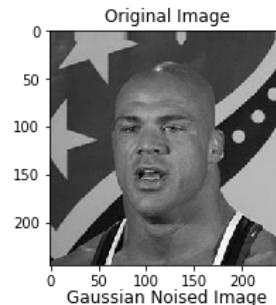
```
In [16]: def max_filter(img):
    neighborhood = morphology.generate_binary_structure(len(img.shape),2)
    i=filters.maximum_filter(img, footprint=neighborhood)
    return i

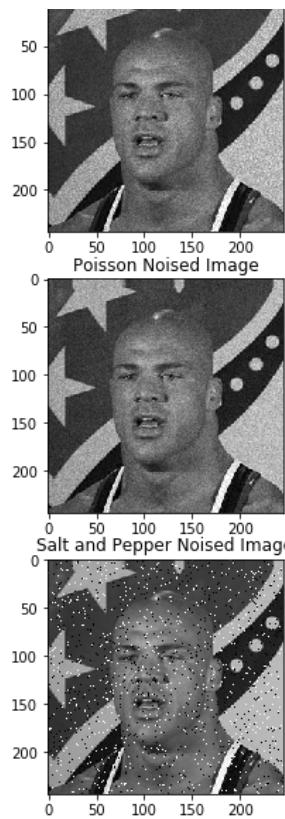
#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

#applying max filter on noises
gauss_max=max_filter(gaussian_noise.copy())
pepper_max=max_filter(pepper_noise.copy())
salt_max=max_filter(salt_noise.copy())
speckle_max=max_filter(speckle_noise.copy())
poisson_max=max_filter(poisson_noise.copy())
salt_pepper_max=max_filter(salt_p_noise_img.copy())

#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_max,plt.cm.gray)
ax[3].set_title("Gaussian Noise- Max Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_max,plt.cm.gray)
ax[5].set_title("Pepper Noise- Max Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_max,plt.cm.gray)
ax[7].set_title("Salt Noise- Max Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_max,plt.cm.gray)
ax[9].set_title("Speckle Noise- Max Filter Image")
```

```
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_max,plt.cm.gray)
ax[11].set_title("Poisson Noise- Max Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_max,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- Max Filter Image")
plt.show()
```





3.8. Midpoint Filter

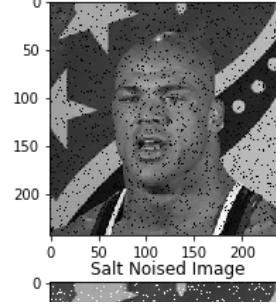
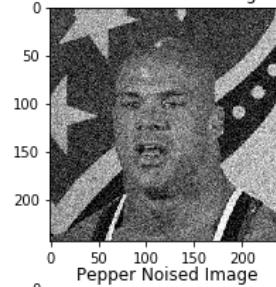
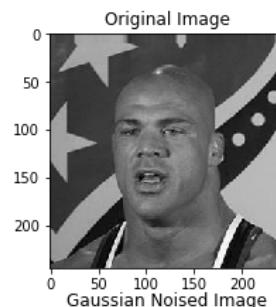
```
In [17]: def midpoint_filter(img):
    rows,cols=img.shape
    img=img*255
    for i in range(1,rows-1):
        for j in range(1,cols-1):
            ans=img[i-1:i+2,j-1:j+2]
            a1=np.amin(ans)
            a2=np.amax(ans)
            ans=1/2*(a1+a2)
            img[i,j]=ans
    return img

#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

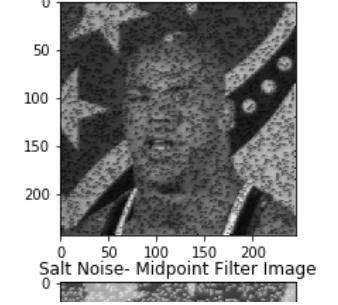
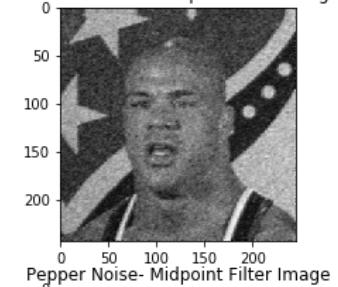
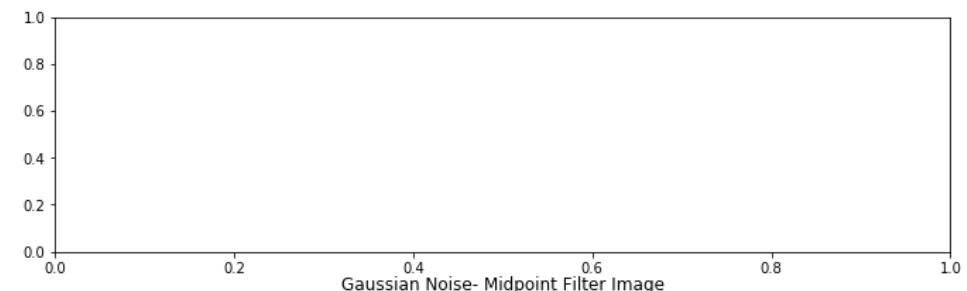
#applying midpoint filter on noises
gauss_midpoint=midpoint_filter(gaussian_noise.copy())
pepper_midpoint=midpoint_filter(pepper_noise.copy())
salt_midpoint=midpoint_filter(salt_noise.copy())
speckle_midpoint=midpoint_filter(speckle_noise.copy())
poisson_midpoint=midpoint_filter(poisson_noise.copy())
salt_pepper_midpoint=midpoint_filter(salt_p_noise_img.copy())

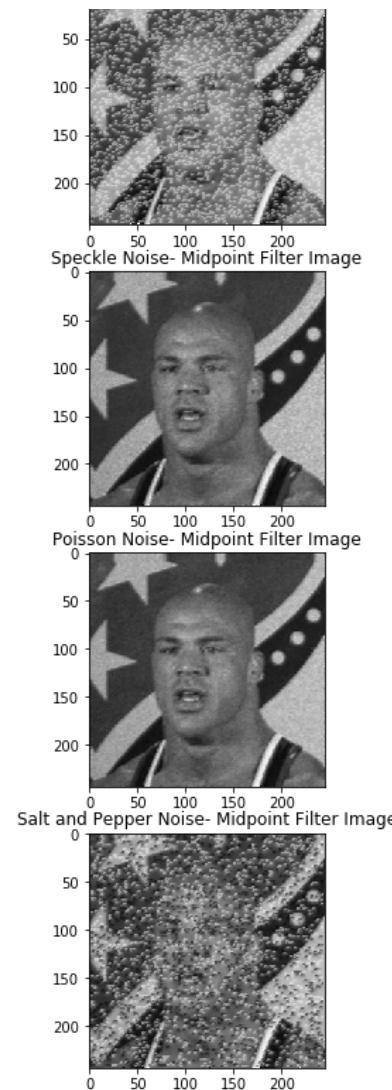
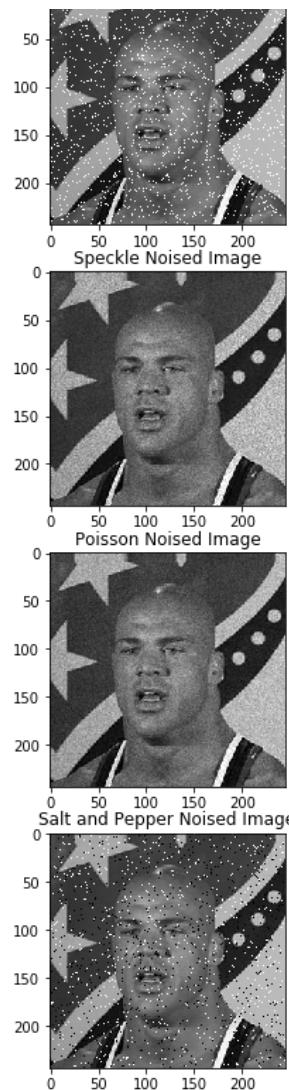
#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_midpoint,plt.cm.gray)
ax[3].set_title("Gaussian Noise- Midpoint Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_midpoint,plt.cm.gray)
ax[5].set_title("Pepper Noise- Midpoint Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
```

```
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_midpoint,plt.cm.gray)
ax[7].set_title("Salt Noise- Midpoint Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_midpoint,plt.cm.gray)
ax[9].set_title("Speckle Noise- Midpoint Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_midpoint,plt.cm.gray)
ax[11].set_title("Poisson Noise- Midpoint Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_midpoint,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- Midpoint Filter Image")
plt.show()
```



Salt Noised Image





3.9. Wiener Filter

```
In [18]: def wiener_filter(x):
    filtered_img = wiener(x, (3, 3))
    return filtered_img

#applying noise
gaussian_noise=random_noise(img4.copy(), mode='gaussian')
pepper_noise=random_noise(img4.copy(), mode='pepper',seed=2)
salt_noise=random_noise(img4.copy(), mode='salt',seed=2)
speckle_noise=random_noise(img4.copy(), mode='speckle',seed=2)
poisson_noise=random_noise(img4.copy(), mode='poisson',seed=2)
salt_p_noise_img=random_noise(img4.copy(), mode='s&p')

#applying wiener filter on noises
gauss_wiener=wiener_filter(gaussian_noise.copy())
pepper_wiener=wiener_filter(pepper_noise.copy())
salt_wiener=wiener_filter(salt_noise.copy())
speckle_wiener=wiener_filter(speckle_noise.copy())
poisson_wiener=wiener_filter(poisson_noise.copy())
salt_pepper_wiener=wiener_filter(salt_p_noise_img.copy())

#setting axis
fig, axes = plt.subplots(7, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[2].imshow(gaussian_noise,plt.cm.gray)
ax[2].set_title("Gaussian Noised Image")
ax[3].imshow(gauss_wiener,plt.cm.gray)
ax[3].set_title("Gaussian Noise- Wiener Filter Image")
ax[4].imshow(pepper_noise,plt.cm.gray)
ax[4].set_title("Pepper Noised Image")
ax[5].imshow(pepper_wiener,plt.cm.gray)
ax[5].set_title("Pepper Noise- Wiener Filter Image")
ax[6].imshow(salt_noise,plt.cm.gray)
ax[6].set_title("Salt Noised Image")
ax[7].imshow(salt_wiener,plt.cm.gray)
ax[7].set_title("Salt Noise- Wiener Filter Image")
ax[8].imshow(speckle_noise,plt.cm.gray)
ax[8].set_title("Speckle Noised Image")
ax[9].imshow(speckle_wiener,plt.cm.gray)
ax[9].set_title("Speckle Noise- Wiener Filter Image")
ax[10].imshow(poisson_noise,plt.cm.gray)
```

```

ax[10].set_title("Poisson Noised Image")
ax[11].imshow(poisson_wiener,plt.cm.gray)
ax[11].set_title("Poisson Noise- Wiener Filter Image")
ax[12].imshow(salt_p_noise_img,plt.cm.gray)
ax[12].set_title("Salt and Pepper Noised Image")
ax[13].imshow(salt_pepper_wiener,plt.cm.gray)
ax[13].set_title("Salt and Pepper Noise- Wiener Filter Image")
plt.show()

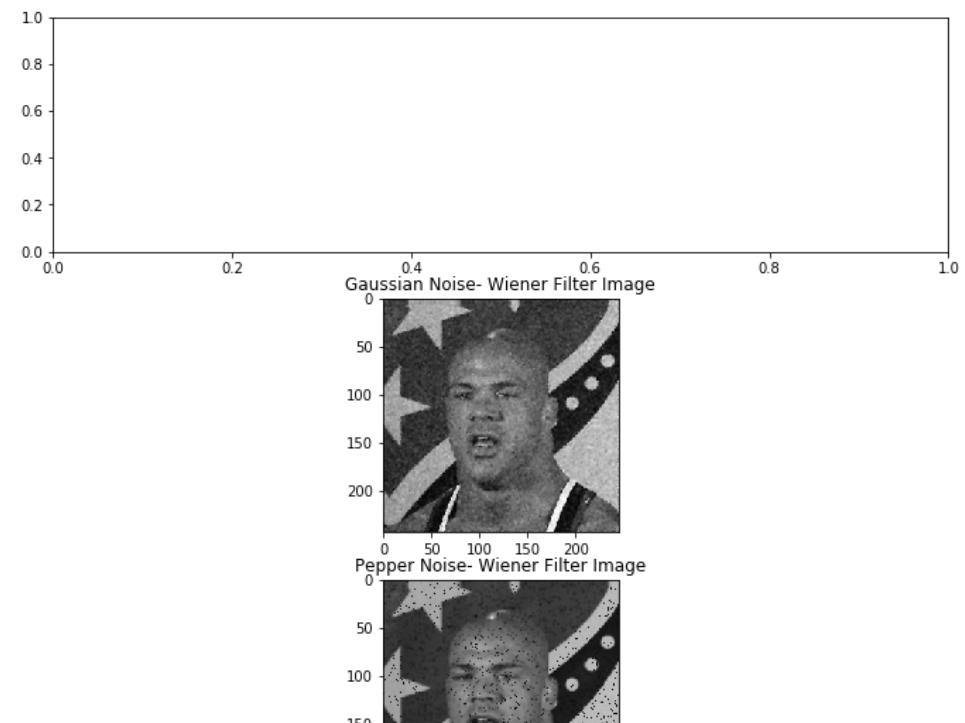
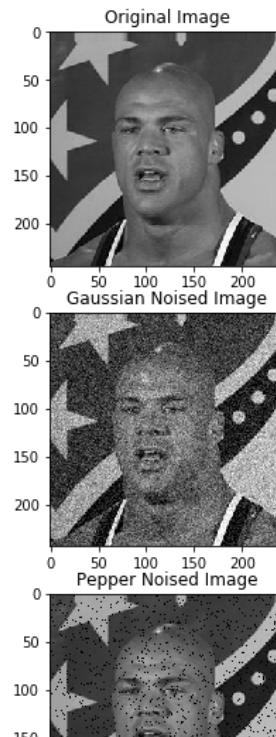
```

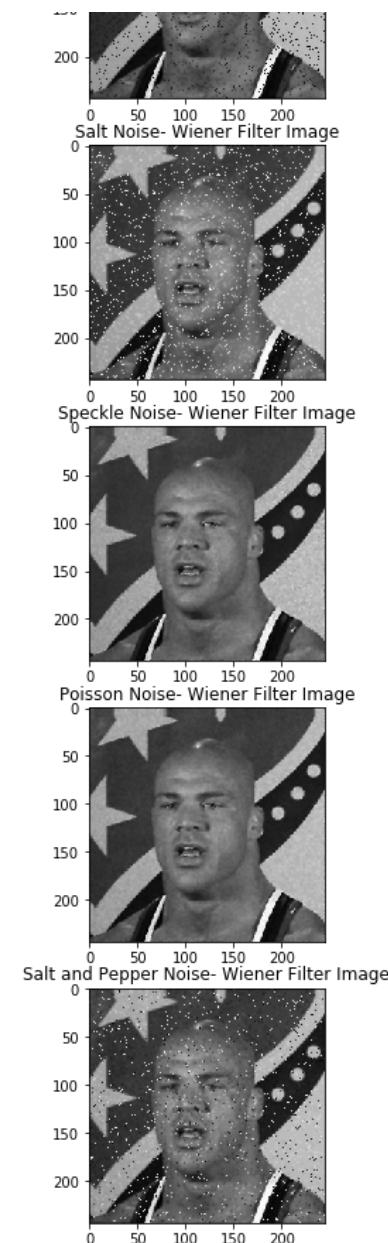
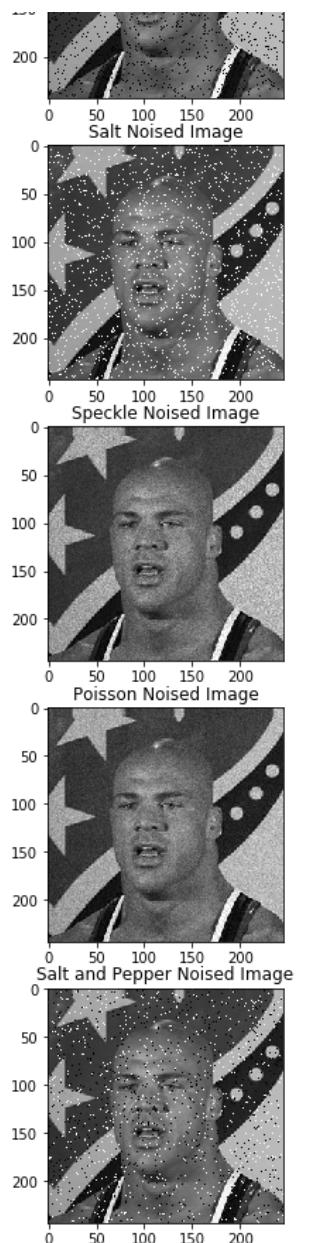
C:\ProgramData\Anaconda3\lib\site-packages\scipy\signal\signaltools.py:491: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.

```

    return x[reverse].conj()
C:\ProgramData\Anaconda3\lib\site-packages\scipy\signal\signaltools.py:940: RuntimeWarning: divide by zero encountered in true_divide
    res *= (1 - noise / lVar)
C:\ProgramData\Anaconda3\lib\site-packages\scipy\signal\signaltools.py:940: RuntimeWarning: invalid value encountered in multiply
    res *= (1 - noise / lVar)

```



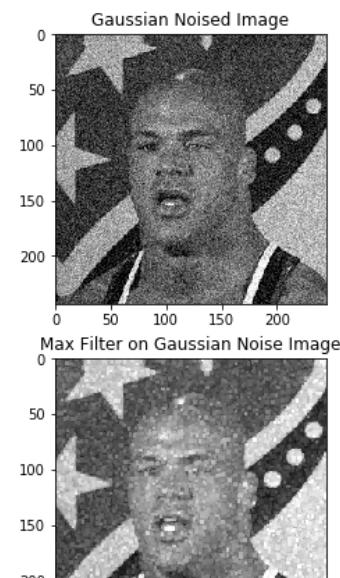
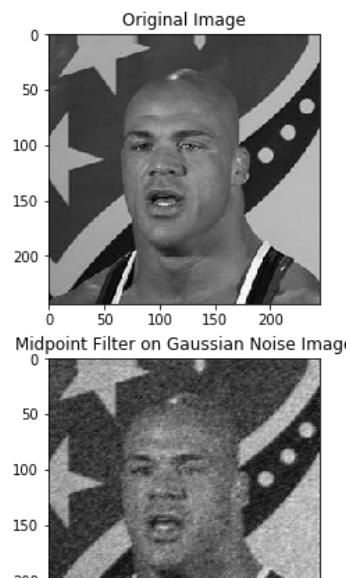


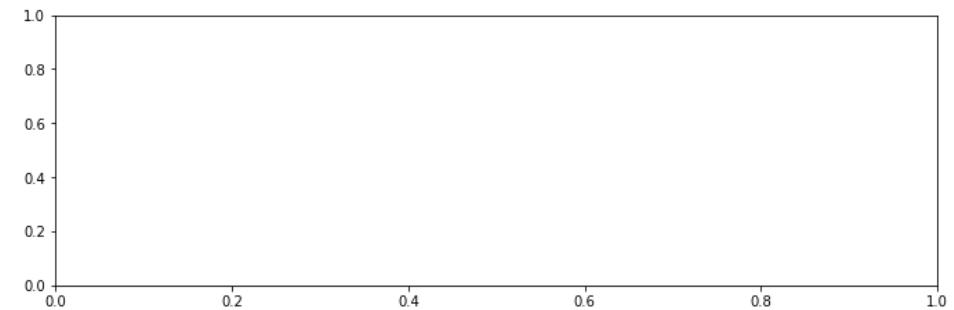
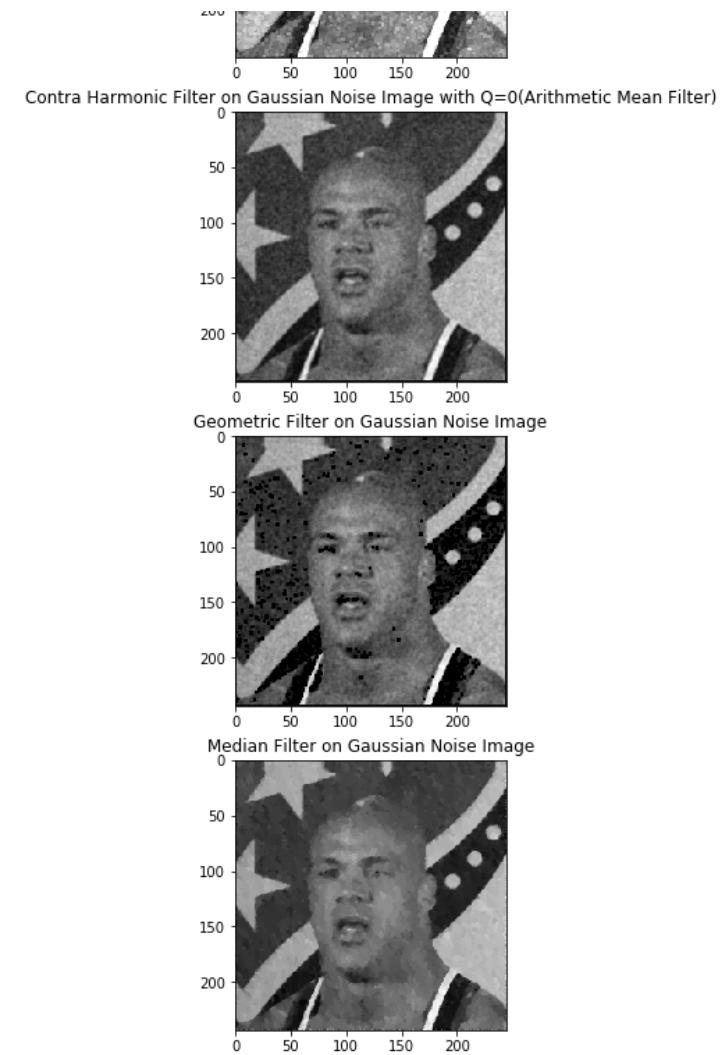
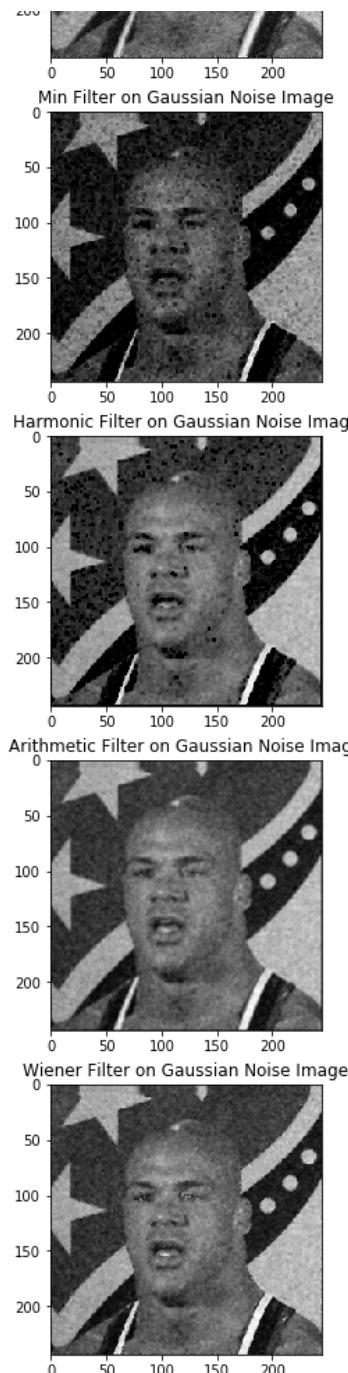
4. Discussion. Which Filter perform better on Specific Noise?

4.1. For Gaussian Noise

In [19]:

```
fig, axes = plt.subplots(6, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(gaussian_noise,plt.cm.gray)
ax[1].set_title("Gaussian Noised Image")
ax[2].imshow(gauss_midpoint,plt.cm.gray)
ax[2].set_title("Midpoint Filter on Gaussian Noise Image")
ax[3].imshow(gauss_max,plt.cm.gray)
ax[3].set_title("Max Filter on Gaussian Noise Image")
ax[4].imshow(gauss_min,plt.cm.gray)
ax[4].set_title("Min Filter on Gaussian Noise Image")
ax[5].imshow(gauss_contra_harmonic,plt.cm.gray)
ax[5].set_title("Contra Harmonic Filter on Gaussian Noise Image with Q=0(Arithmetic Mean Filter)")
ax[6].imshow(gauss_harmonic,plt.cm.gray)
ax[6].set_title("Harmonic Filter on Gaussian Noise Image")
ax[7].imshow(gauss_geometric,plt.cm.gray)
ax[7].set_title("Geometric Filter on Gaussian Noise Image")
ax[8].imshow(gauss_arithmetic,plt.cm.gray)
ax[8].set_title("Arithmetic Filter on Gaussian Noise Image")
ax[9].imshow(gauss_median,plt.cm.gray)
ax[9].set_title("Median Filter on Gaussian Noise Image")
ax[10].imshow(gauss_wiener,plt.cm.gray)
ax[10].set_title("Wiener Filter on Gaussian Noise Image")
plt.show()
```





As you can see **Arithmetic Mean, Midpoint, Wiener, Median Filter** performs well on image with gaussian noise as compared to other

Filters.

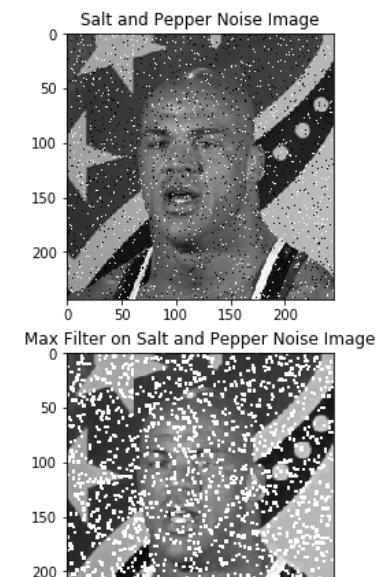
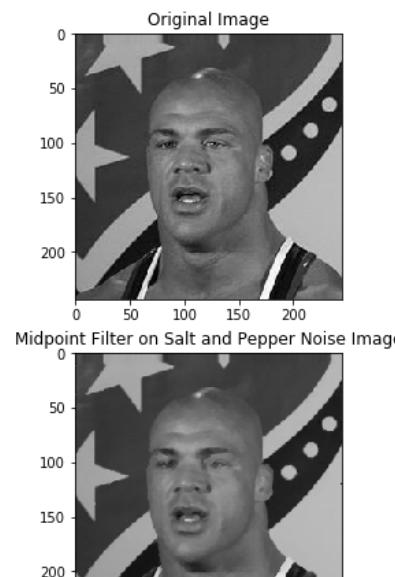
Arithmetic Mean Filter Image is blurry as compared to other filters because arithmetic mean filter reduces the noise but causes blurring to image.

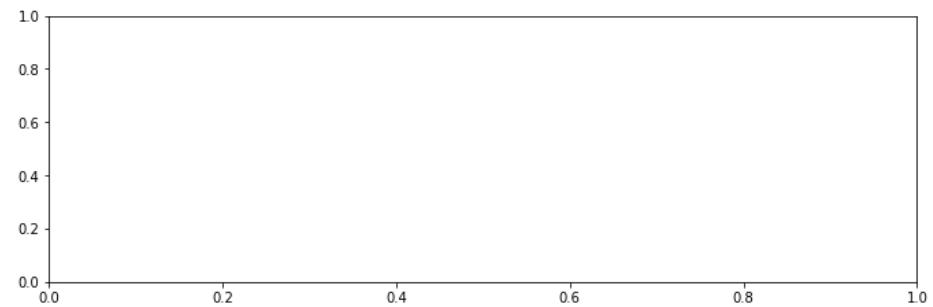
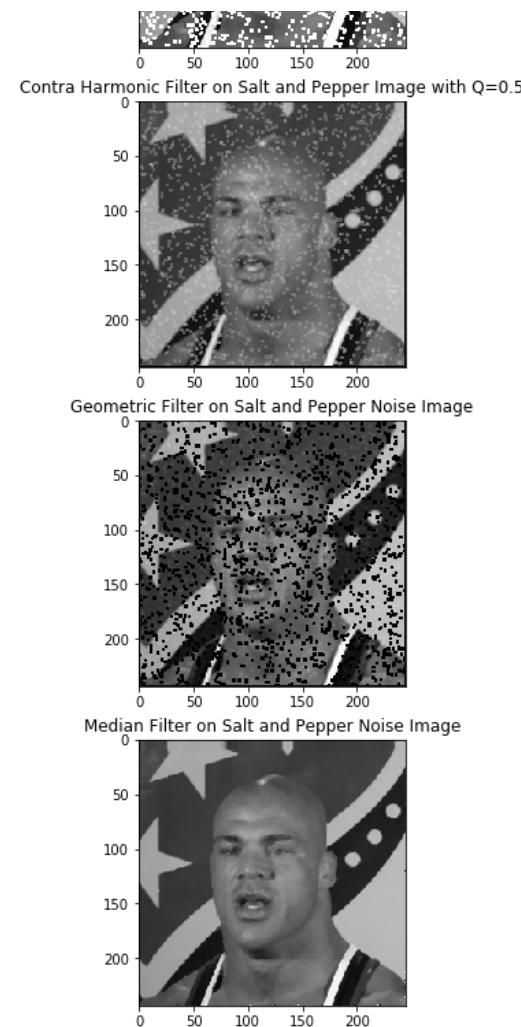
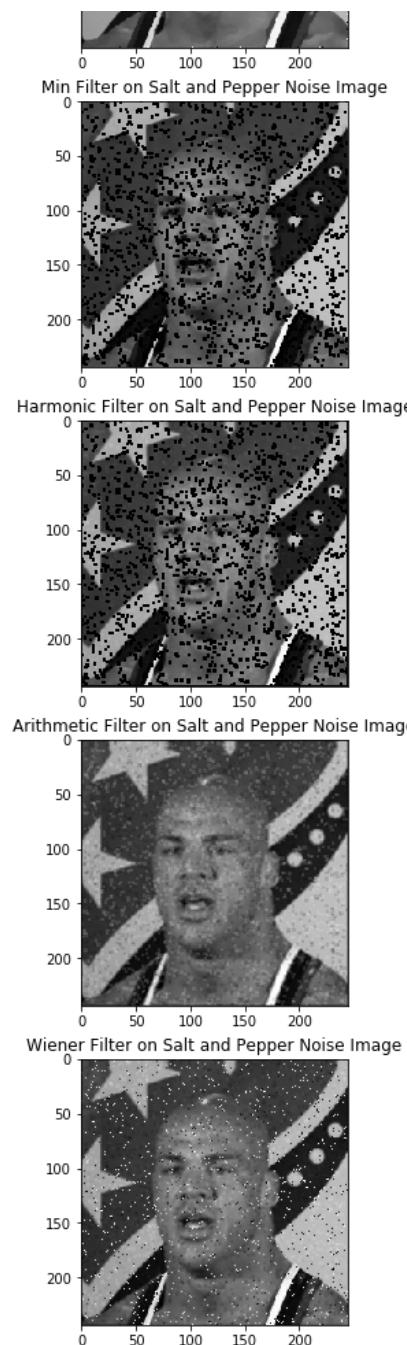
Median Filter reduces the noise of image by keeping edges sharp.

Midpoint Filter reduces the noise of image by averaging and order statistics.

4.2. For Salt and Pepper Noise

```
In [20]: fig, axes = plt.subplots(6, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(salt_p_noise_img,plt.cm.gray)
ax[1].set_title("Salt and Pepper Noise Image")
ax[2].imshow(salt_pepper_median,plt.cm.gray)
ax[2].set_title("Midpoint Filter on Salt and Pepper Noise Image")
ax[3].imshow(salt_pepper_max,plt.cm.gray)
ax[3].set_title("Max Filter on Salt and Pepper Noise Image")
ax[4].imshow(salt_pepper_min,plt.cm.gray)
ax[4].set_title("Min Filter on Salt and Pepper Noise Image")
ax[5].imshow(salt_pepper_contra_harmonic,plt.cm.gray)
ax[5].set_title("Contra Harmonic Filter on Salt and Pepper Image with Q=0.5")
ax[6].imshow(salt_pepper_harmonic,plt.cm.gray)
ax[6].set_title("Harmonic Filter on Salt and Pepper Noise Image")
ax[7].imshow(salt_pepper_geometric,plt.cm.gray)
ax[7].set_title("Geometric Filter on Salt and Pepper Noise Image")
ax[8].imshow(salt_pepper_arithmetic,plt.cm.gray)
ax[8].set_title("Arithmetc Filter on Salt and Pepper Noise Image")
ax[9].imshow(salt_pepper_median,plt.cm.gray)
ax[9].set_title("Median Filter on Salt and Pepper Noise Image")
ax[10].imshow(salt_pepper_wiener,plt.cm.gray)
ax[10].set_title("Wiener Filter on Salt and Pepper Noise Image")
plt.show()
```





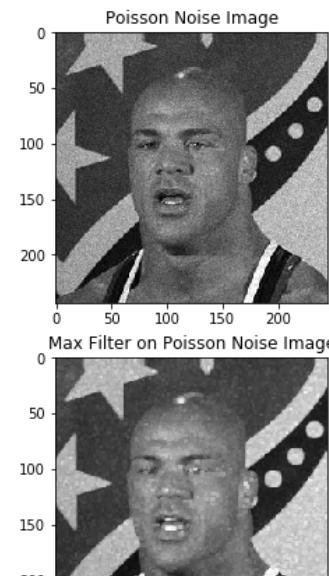
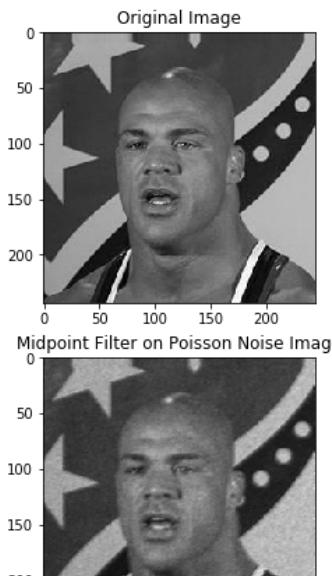
As you can see **Arithmetic Mean, Contra Harmonic with Q=0.5** performs well on image with Salt and Pepper noise as compared to other Filters.

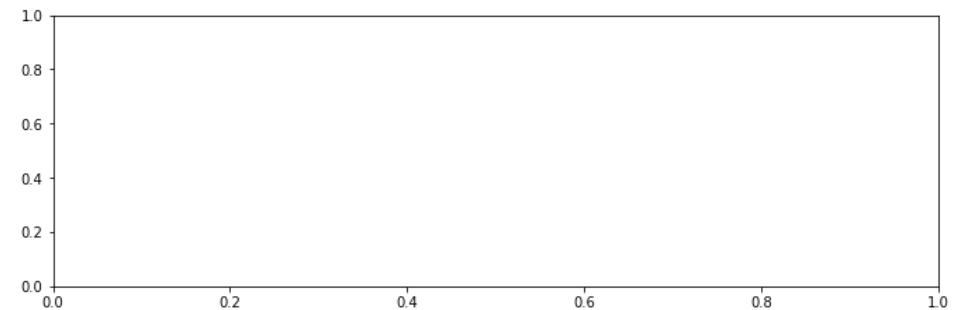
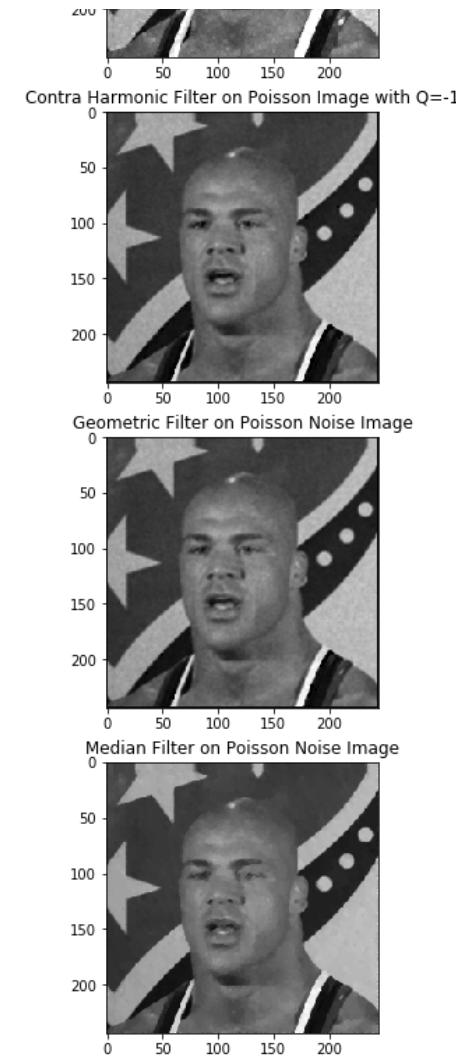
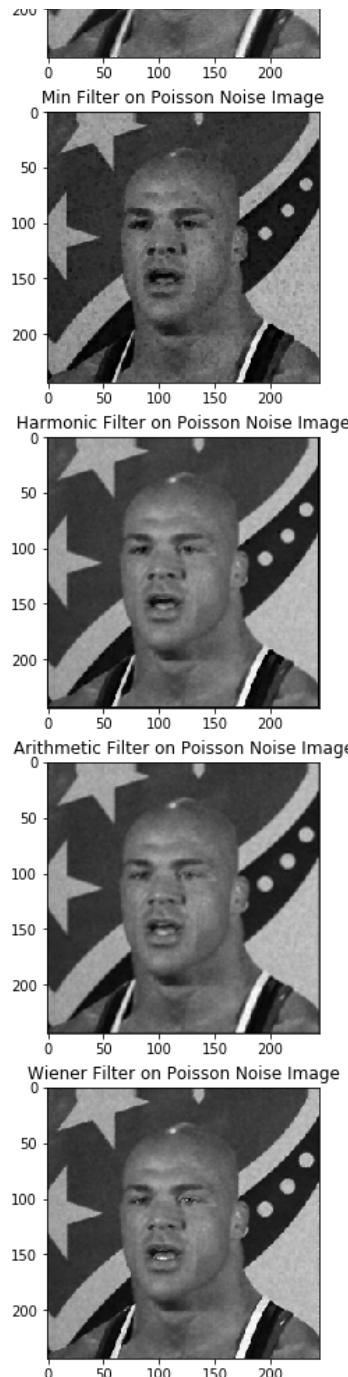
Arithmetic Mean Filter: Image is blurry as compared to other filters because arithmetic mean filter reduces the noise but causes blurring to image.

Contra Harmonic with Q=0.5: eliminates the pepper noise as Q>0 is for eliminating Pepper Noise in image. And it cannot eliminate both salt and pepper at a time. After applying it many times salt and pepper noise can be eliminated.

4.3. For Poisson Noise

```
In [21]: fig, axes = plt.subplots(6, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(poisson_noise,plt.cm.gray)
ax[1].set_title("Poisson Noise Image")
ax[2].imshow(poisson_midpoint,plt.cm.gray)
ax[2].set_title("Midpoint Filter on Poisson Noise Image")
ax[3].imshow(poisson_max,plt.cm.gray)
ax[3].set_title("Max Filter on Poisson Noise Image")
ax[4].imshow(poisson_min,plt.cm.gray)
ax[4].set_title("Min Filter on Poisson Noise Image")
ax[5].imshow(poisson_contra_harmonic,plt.cm.gray)
ax[5].set_title("Contra Harmonic Filter on Poisson Image with Q=-1")
ax[6].imshow(poisson_harmonic,plt.cm.gray)
ax[6].set_title("Harmonic Filter on Poisson Noise Image")
ax[7].imshow(poisson_geometric,plt.cm.gray)
ax[7].set_title("Geometric Filter on Poisson Noise Image")
ax[8].imshow(poisson_arithmetical,plt.cm.gray)
ax[8].set_title("Arithmetical Filter on Poisson Noise Image")
ax[9].imshow(poisson_median,plt.cm.gray)
ax[9].set_title("Median Filter on Poisson Noise Image")
ax[10].imshow(poisson_wiener,plt.cm.gray)
ax[10].set_title("Wiener Filter on Poisson Noise Image")
plt.show()
```

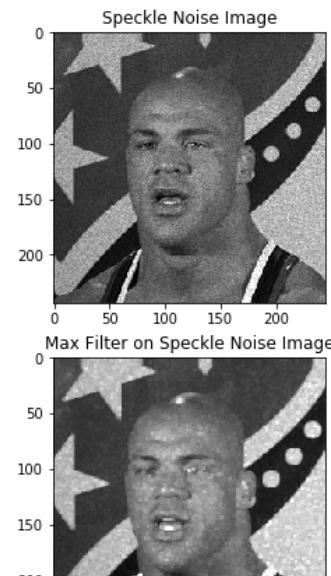
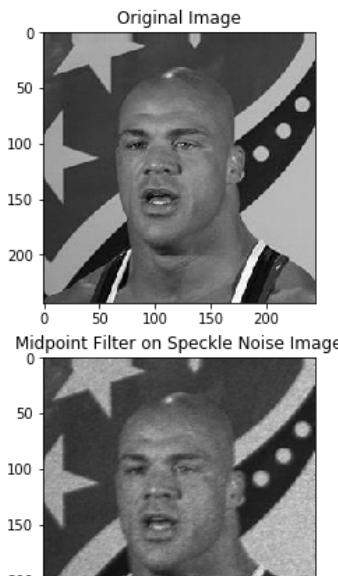


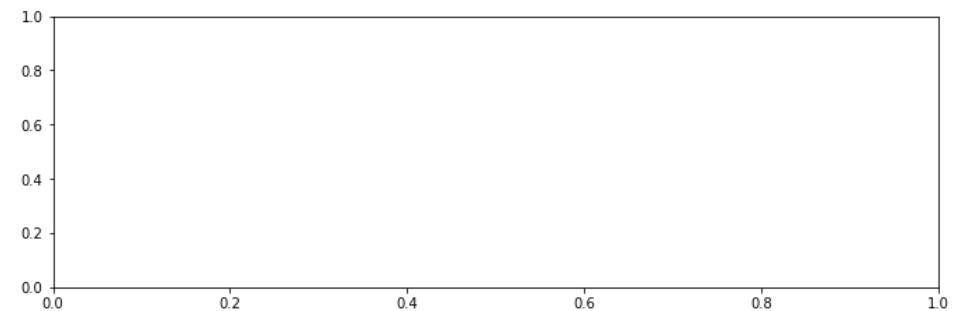
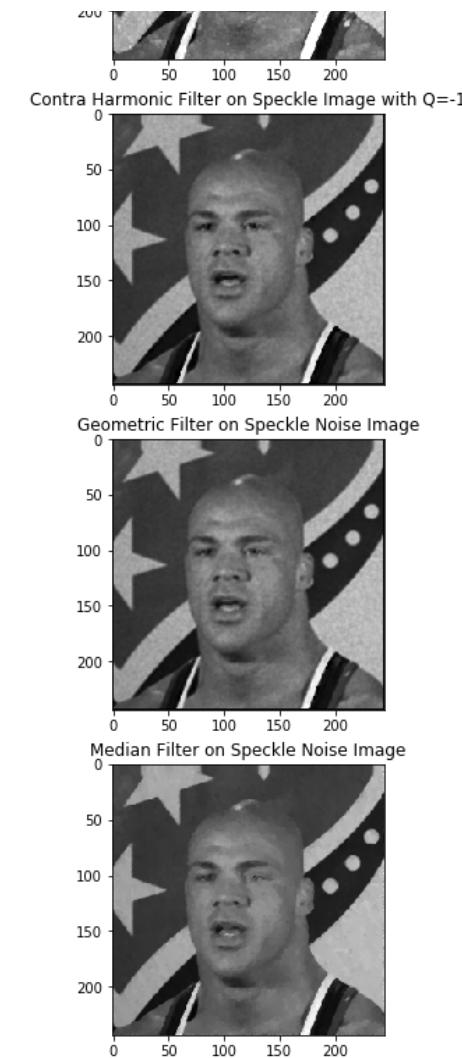
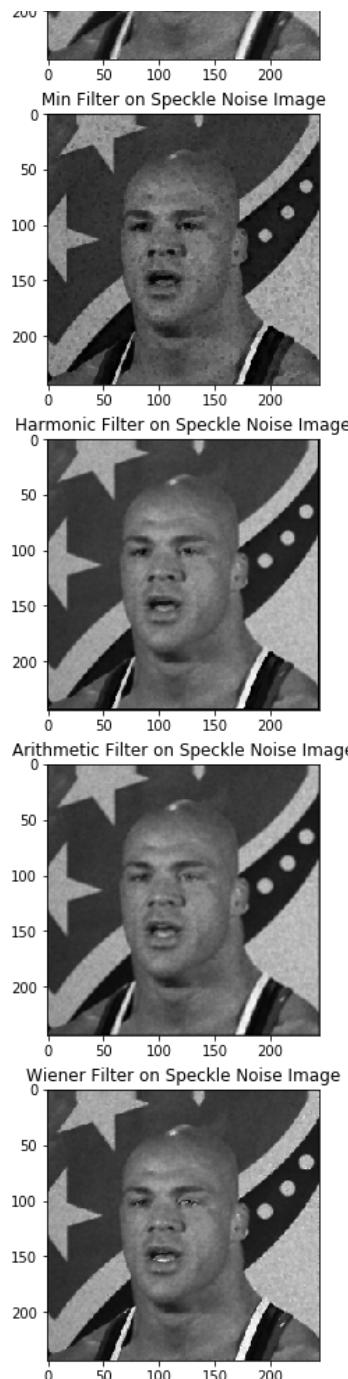


As you can see Almost all filters performs well on image with poisson noise except **Min and Max Filter**

4.4. For Speckle Noise

```
In [22]: fig, axes = plt.subplots(6, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(speckle_noise,plt.cm.gray)
ax[1].set_title("Speckle Noise Image")
ax[2].imshow(speckle_midpoint,plt.cm.gray)
ax[2].set_title("Midpoint Filter on Speckle Noise Image")
ax[3].imshow(speckle_max,plt.cm.gray)
ax[3].set_title("Max Filter on Speckle Noise Image")
ax[4].imshow(speckle_min,plt.cm.gray)
ax[4].set_title("Min Filter on Speckle Noise Image")
ax[5].imshow(speckle_contra_harmonic,plt.cm.gray)
ax[5].set_title("Contra Harmonic Filter on Speckle Image with Q=-1")
ax[6].imshow(speckle_harmonic,plt.cm.gray)
ax[6].set_title("Harmonic Filter on Speckle Noise Image")
ax[7].imshow(speckle_geometric,plt.cm.gray)
ax[7].set_title("Geometric Filter on Speckle Noise Image")
ax[8].imshow(speckle_arithmetical,plt.cm.gray)
ax[8].set_title("Arithmetical Filter on Speckle Noise Image")
ax[9].imshow(speckle_median,plt.cm.gray)
ax[9].set_title("Median Filter on Speckle Noise Image")
ax[10].imshow(speckle_wiener,plt.cm.gray)
ax[10].set_title("Wiener Filter on Speckle Noise Image")
plt.show()
```

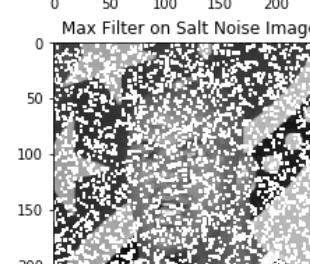
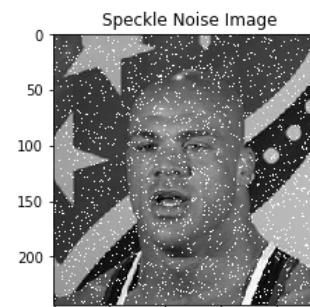
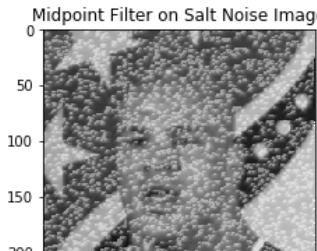
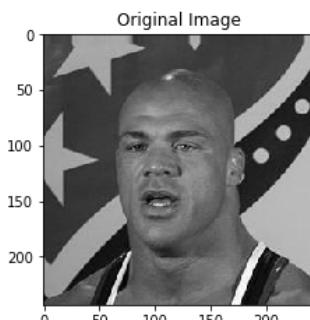


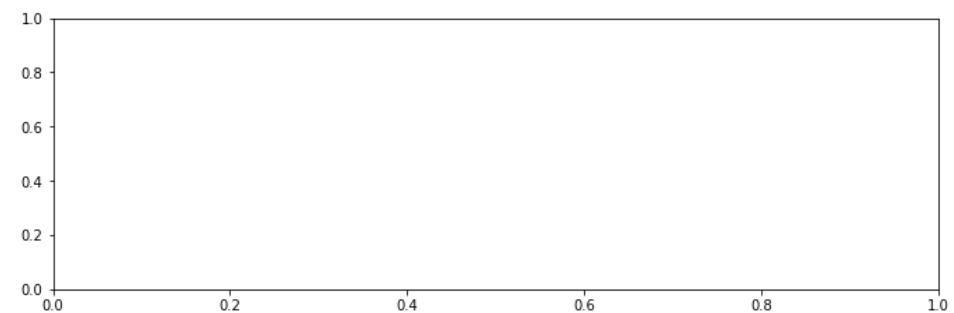
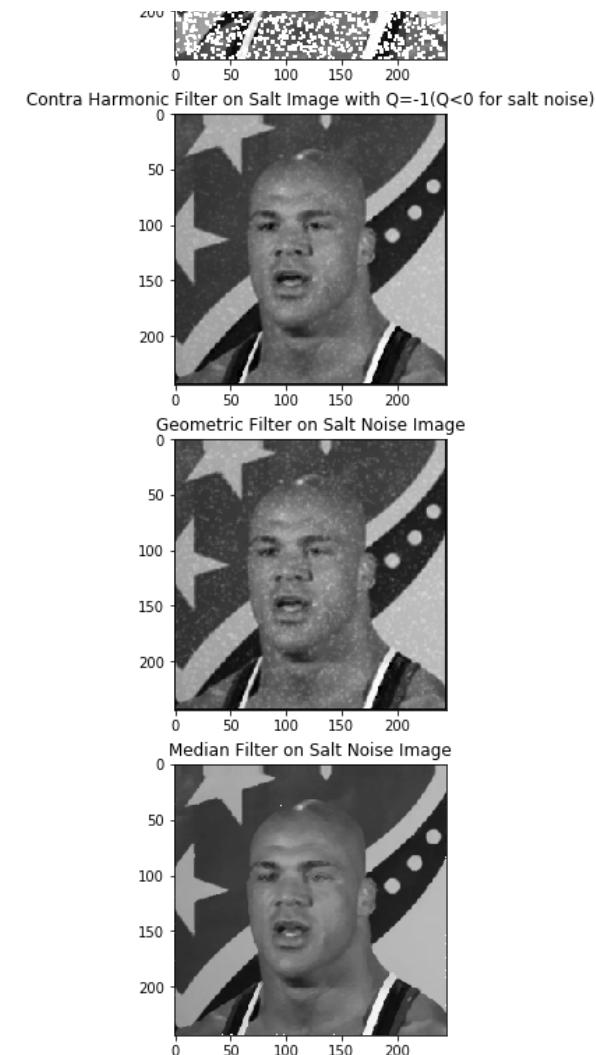
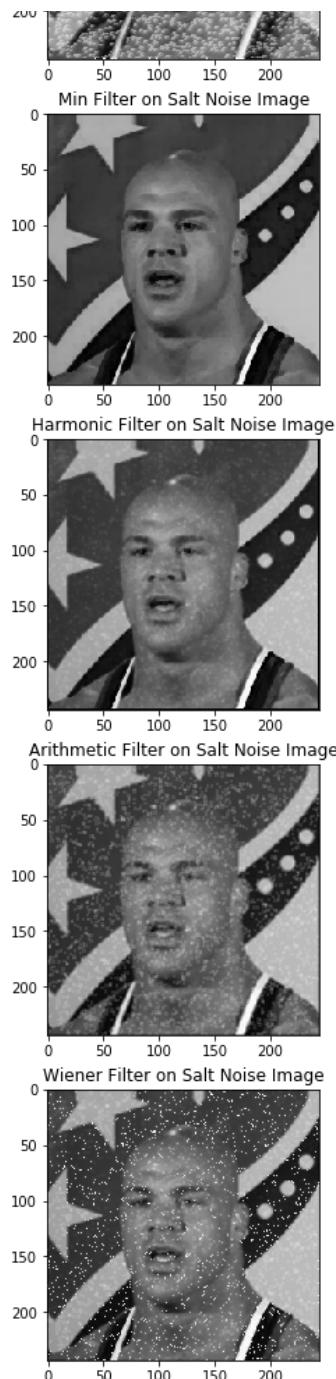


As you can see Almost all filters performs well on image with speckle noise except **Min and Max Filter**

4.5. For Salt Noise

```
In [23]: fig, axes = plt.subplots(6, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(salt_noise,plt.cm.gray)
ax[1].set_title("Speckle Noise Image")
ax[2].imshow(salt_midpoint,plt.cm.gray)
ax[2].set_title("Midpoint Filter on Salt Noise Image")
ax[3].imshow(salt_max,plt.cm.gray)
ax[3].set_title("Max Filter on Salt Noise Image")
ax[4].imshow(salt_min,plt.cm.gray)
ax[4].set_title("Min Filter on Salt Noise Image")
ax[5].imshow(salt_contra_harmonic,plt.cm.gray)
ax[5].set_title("Contra Harmonic Filter on Salt Image with Q=-1(Q<0 for salt noise)")
ax[6].imshow(salt_harmonic,plt.cm.gray)
ax[6].set_title("Harmonic Filter on Salt Noise Image")
ax[7].imshow(salt_geometric,plt.cm.gray)
ax[7].set_title("Geometric Filter on Salt Noise Image")
ax[8].imshow(salt_arithmetic,plt.cm.gray)
ax[8].set_title("Arithmetic Filter on Salt Noise Image")
ax[9].imshow(salt_median,plt.cm.gray)
ax[9].set_title("Median Filter on Salt Noise Image")
ax[10].imshow(salt_wiener,plt.cm.gray)
ax[10].set_title("Wiener Filter on Salt Noise Image")
plt.show()
```





As you can see **Min Filter, Contra Harmonic with $Q=-1$, Harmonic,Geometric Filter** performs well on image with Salt Noise.

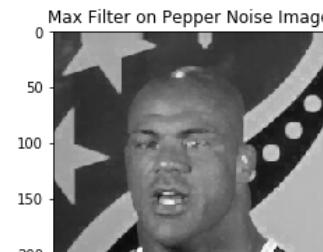
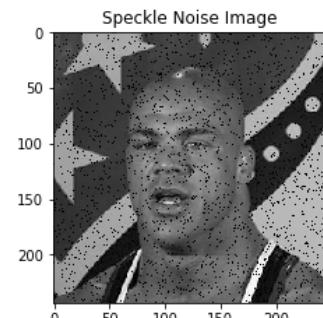
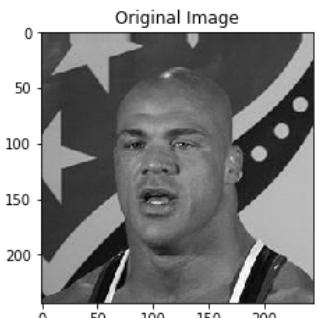
Min Filter: It helps in removing salt noise and replaces the value of pixel from image with minimum value in neighborhood. By minimizing the value we will get more dark pixels so salt noise will be eliminated.

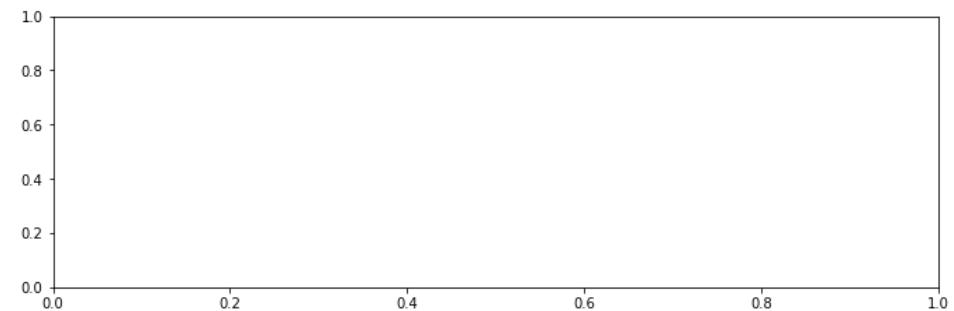
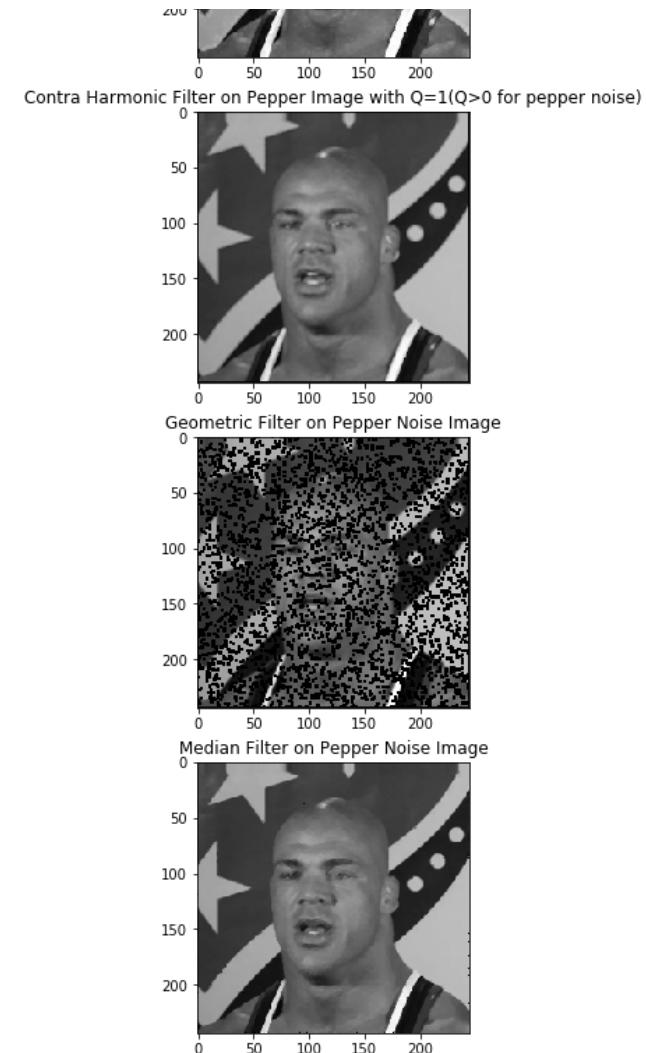
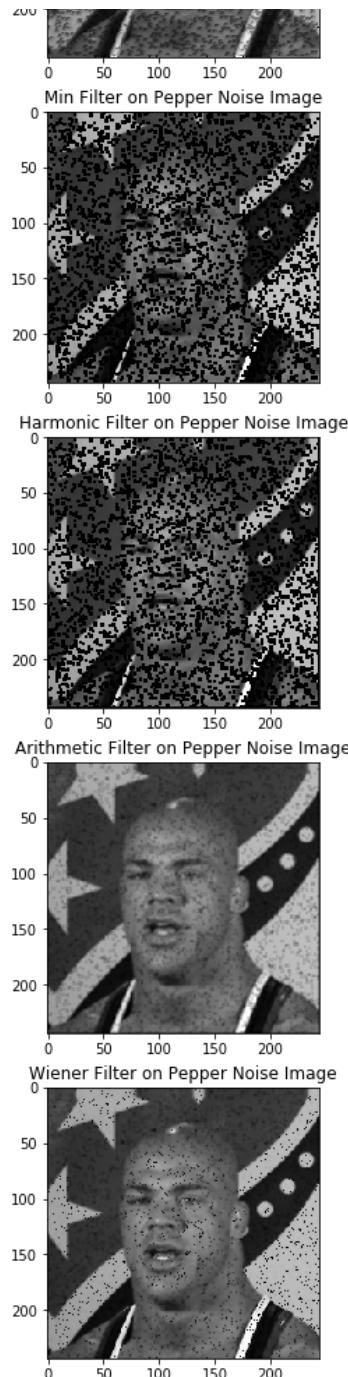
Contra Harmonic with Q=-1: Contra Harmonic with $Q < 0$, eliminates Salt Noise in Image at expense of blurring the bright Areas.

Harmonic Filter: We will get shorter intensity value and dark pixels because harmonic filter returns us more dark pixels by which Salt Noise will be reduced.

4.6. For Pepper Noise

```
In [24]: fig, axes = plt.subplots(6, 2, figsize=(25,25))
ax = axes.ravel()
ax[0].imshow(img4,plt.cm.gray)
ax[0].set_title("Original Image")
ax[1].imshow(pepper_noise,plt.cm.gray)
ax[1].set_title("Speckle Noise Image")
ax[2].imshow(pepper_midpoint,plt.cm.gray)
ax[2].set_title("Midpoint Filter on Pepper Noise Image")
ax[3].imshow(pepper_max,plt.cm.gray)
ax[3].set_title("Max Filter on Pepper Noise Image")
ax[4].imshow(pepper_min,plt.cm.gray)
ax[4].set_title("Min Filter on Pepper Noise Image")
ax[5].imshow(pepper_contra_harmonic,plt.cm.gray)
ax[5].set_title("Contra Harmonic Filter on Pepper Image with Q=1(Q>0 for pepper noise)")
ax[6].imshow(pepper_harmonic,plt.cm.gray)
ax[6].set_title("Harmonic Filter on Pepper Noise Image")
ax[7].imshow(pepper_geometric,plt.cm.gray)
ax[7].set_title("Geometric Filter on Pepper Noise Image")
ax[8].imshow(pepper_arithmetical,plt.cm.gray)
ax[8].set_title("Arithmetical Filter on Pepper Noise Image")
ax[9].imshow(pepper_median,plt.cm.gray)
ax[9].set_title("Median Filter on Pepper Noise Image")
ax[10].imshow(pepper_wiener,plt.cm.gray)
ax[10].set_title("Wiener Filter on Pepper Noise Image")
plt.show()
```





As you can see **Max Filter, Contra Harmonic with $Q=1$** performs well on image with Pepper Noise.

Max Filter: It helps in removing pepper noise and replaces the value of pixel from image with maximum value in neighborhood. By maximizing the value we will get more light pixels so pepper noise will be eliminated.

Contra Harmonic with Q=1: Contra Harmonic with $Q>0$, eliminates Pepper Noise in Image at expense of blurring the dark Areas.

The End