IPPR Lab 4-B

Name : Arya Shah

Roll No. E071

Class : BTech CSBS

Aim: Apply Median Filter To Reduce The Effect Of Salt & Pepper Noise On The Given Image

```
#Importing Libraries
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
import numpy as np
from scipy import signal
from random import randint
```

```
#Import Image
image_color=io.imread('baboon.png')
```

```
image_color.shape
```

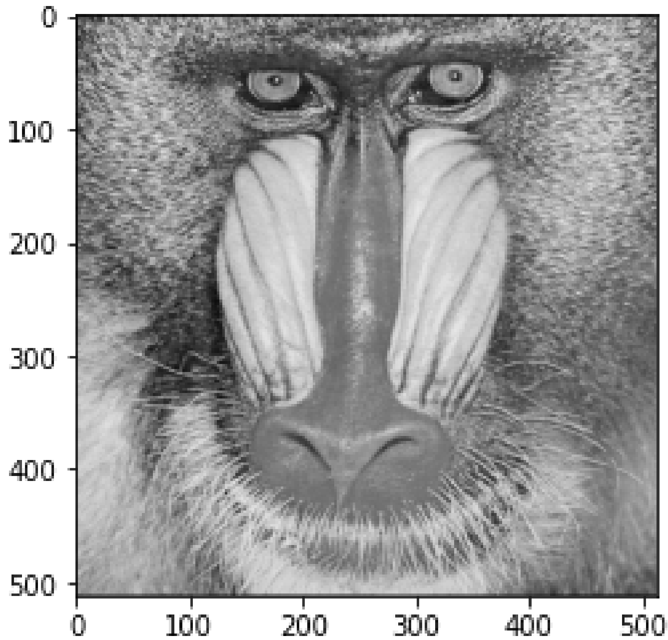
```
(512, 512, 3)
```

```
image = rgb2gray(image_color)
```

```
image=255*image
```

```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x12e2fe25a08>
```




```
image.shape
```

```
(512, 512)
```

```
sh = image.shape
```

```
rows = sh[0]
cols = sh[1]
```

```
no_pixels = rows*cols
no_pixels
```

```
262144
```

```
# % of pixels corrupted by noise
a = 0.1
no_rp = int(a * no_pixels)
```

```
no_rp
```

```
26214
```

```
sp = 255
```

```
image_sp = image.copy()
```

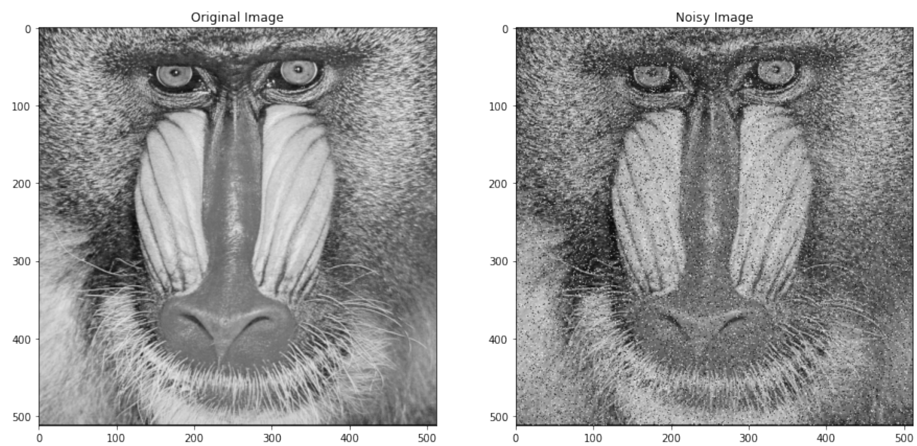
```
# Take random row and column and change the pixel value. Alternate pixels assumed as salt a
for i in range(no_rp):
    temp1 = randint(0,rows-1)
```

```
    temp2 = randint(0,cols-1)
    image_sp[temp1][temp2] = sp
    if sp == 255:
        sp = 0
    else:
        sp = 255

#Display original and noisy image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_sp, cmap = 'gray')
plt.title('Noisy Image')
```

```
Text(0.5, 1.0, 'Noisy Image')
```




```
image_sp_filt = image_sp.copy()
```

```
sz = 3
# central value
cent = int((sz-1)/2)
med = int(((sz*sz)-1)/2) # getting the fifth value which is the middle value in a 3x3 matri
```

```
print(cent)
print(med)
```

```
1
4
```

```
for r in range(0,rows-sz):
    for c in range(0,cols-sz):
```

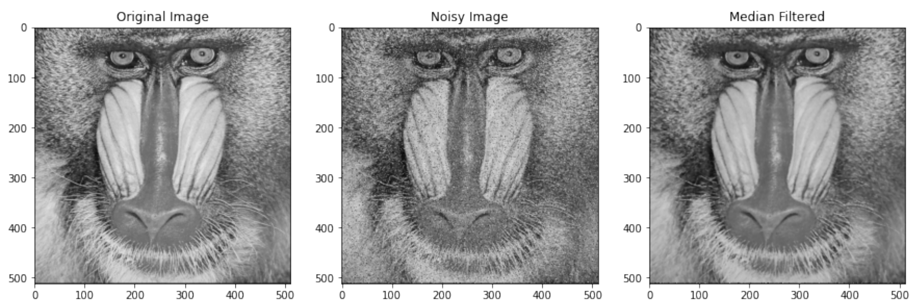
```
        temp1 = image_sp[r:r+sz,c:c+sz] # 0,1,2 rows and cols values extracted
        temp2 = np.reshape(temp1, (1,sz*sz)) # Reshaped to single vector
        temp3 = np.sort(temp2)
        image_sp_filt[r+cent][c+cent] = temp3[0][med]

plt.figure(figsize=(15,15))
plt.subplot(1,3,1)
plt.imshow(image,cmap='gray')
plt.title('Original Image')

plt.subplot(1,3,2)
plt.imshow(image_sp,cmap='gray')
plt.title('Noisy Image')

plt.subplot(1,3,3)
plt.imshow(image_sp_filt, cmap = 'gray')
plt.title('Median Filtered')
```

```
Text(0.5, 1.0, 'Median Filtered')
```




```
# Increasing Noise
# % of pixels corrupted by noise
a = 0.3
no_rp = int(a * no_pixels)
```

```
no_rp
```

```
78643
```

```
sp =255
```

```
image_sp1 = image.copy()
```

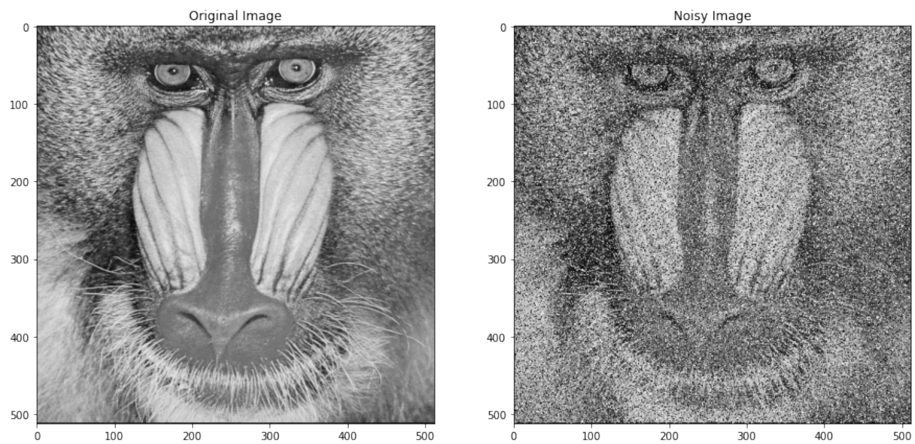
```
# Take random row and column and change the pixel value. Alternate pixels assumed as salt a
for i in range(no_rp):
    temp1 = randint(0,rows-1)
    temp2 = randint(0,cols-1)
    image_sp1[temp1][temp2] = sp
    if sp == 255:
```

```
        sp = 0
    else:
        sp = 255

#Display original and noisy image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_sp1, cmap = 'gray')
plt.title('Noisy Image')
```

```
Text(0.5, 1.0, 'Noisy Image')
```




```
image_sp_filt1 = image_sp1.copy()
```

```
sz = 3
# central value
cent = int((sz-1)/2)
med = int(((sz*sz)-1)/2) # getting the fifth value which is the middle value in a 3x3 matri
```

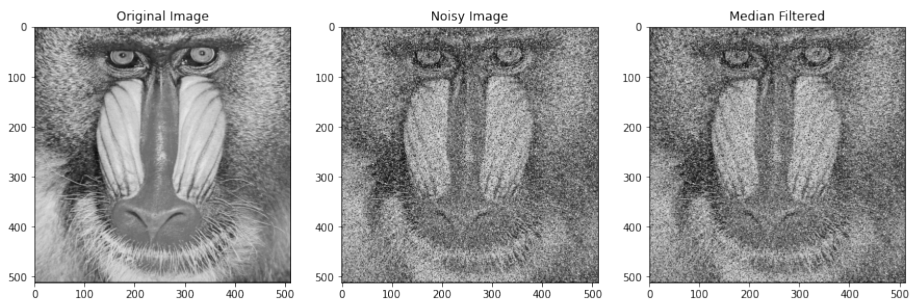
```
for r in range(0,rows-sz):
    for c in range(0,cols-sz):
        temp1 = image_sp1[r:r+sz,c:c+sz] # 0,1,2 rows and cols values extracted
        temp2 = np.reshape(temp1, (1,sz*sz)) # Reshaped to single vector
        temp3 = np.sort(temp2)
        image_sp_filt[r+cent][c+cent] = temp3[0][med]
```

```
plt.figure(figsize=(15,15))
plt.subplot(1,3,1)
plt.imshow(image,cmap='gray')
plt.title('Original Image')
```

```
plt.subplot(1,3,2)
plt.imshow(image_sp1,cmap='gray')
plt.title('Noisy Image')

plt.subplot(1,3,3)
plt.imshow(image_sp_filt1, cmap = 'gray')
plt.title('Median Filtered')
```

```
Text(0.5, 1.0, 'Median Filtered')
```




```
# Taking 5x5 mask and a=0.4
# Increasing Noise
# % of pixels corrupted by noise
a = 0.4
no_rp = int(a * no_pixels)
```

```
no_rp
```

```
104857
```

```
sp=255
```

```
image_sp2 = image.copy()
```

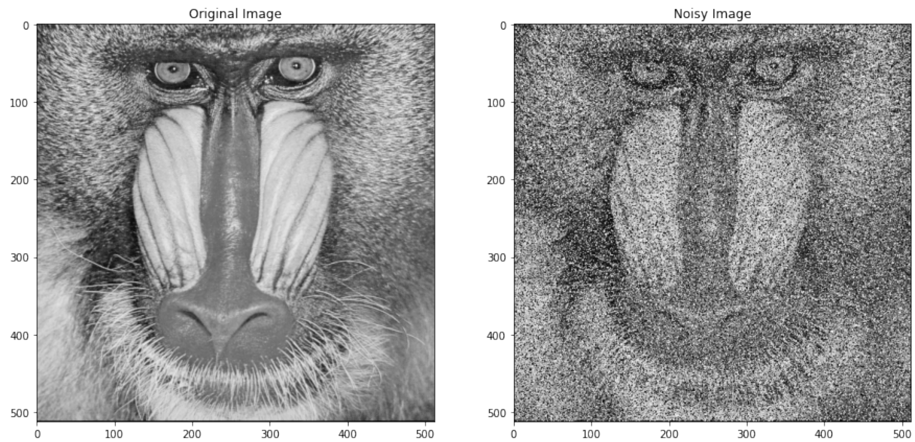
```
# Take random row and column and change the pixel value. Alternate pixels assumed as salt a
for i in range(no_rp):
    temp1 = randint(0,rows-1)
    temp2 = randint(0,cols-1)
    image_sp2[temp1][temp2] = sp
    if sp == 255:
        sp = 0
    else:
        sp = 255
```

```
#Display original and noisy image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
```

```
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_sp2, cmap = 'gray')
plt.title('Noisy Image')
```

```
Text(0.5, 1.0, 'Noisy Image')
```




```
image_sp_filt2 = image_sp2.copy()
```

```
sz = 5
# central value
cent = int((sz-1)/2)
med = int(((sz*sz)-1)/2) # getting the fifth value which is the middle value in a 3x3 matri
```

```
print(cent)
print(med)
```

```
2
12
```

```
for r in range(0,rows-sz):
    for c in range(0,cols-sz):
        temp1 = image_sp2[r:r+sz,c:c+sz] # 0,1,2 rows and cols values extracted
        temp2 = np.reshape(temp1, (1,sz*sz)) # Reshaped to single vector
        temp3 = np.sort(temp2)
        image_sp_filt2[r+cent][c+cent] = temp3[0][med]
```

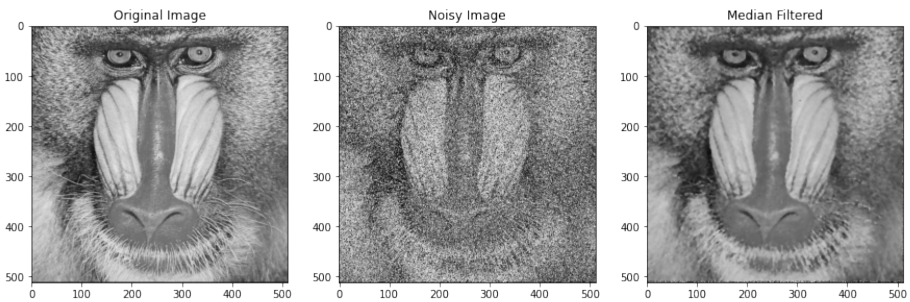
```
plt.figure(figsize=(15,15))
plt.subplot(1,3,1)
plt.imshow(image,cmap='gray')
plt.title('Original Image')

plt.subplot(1,3,2)
```

```
plt.imshow(image_sp2,cmap='gray')
plt.title('Noisy Image')

plt.subplot(1,3,3)
plt.imshow(image_sp_filt2, cmap = 'gray')
plt.title('Median Filtered')
```

```
Text(0.5, 1.0, 'Median Filtered')
```



Conclusion

- The given image is corrupted with salt and pepper noise.
- If 10% of the pixels are corrupted by the noise then median filter of size 3x3 is effective in reducing the noise to almost negligible value.
- If 40% of the image is corrupted by the noise, then the size of the median filter is to be increased to get the significant effect of the filter.
- However if the size of the filter is increased, then the sharpness of the edges reduces. This is because size of the filter is more than the size of the object.