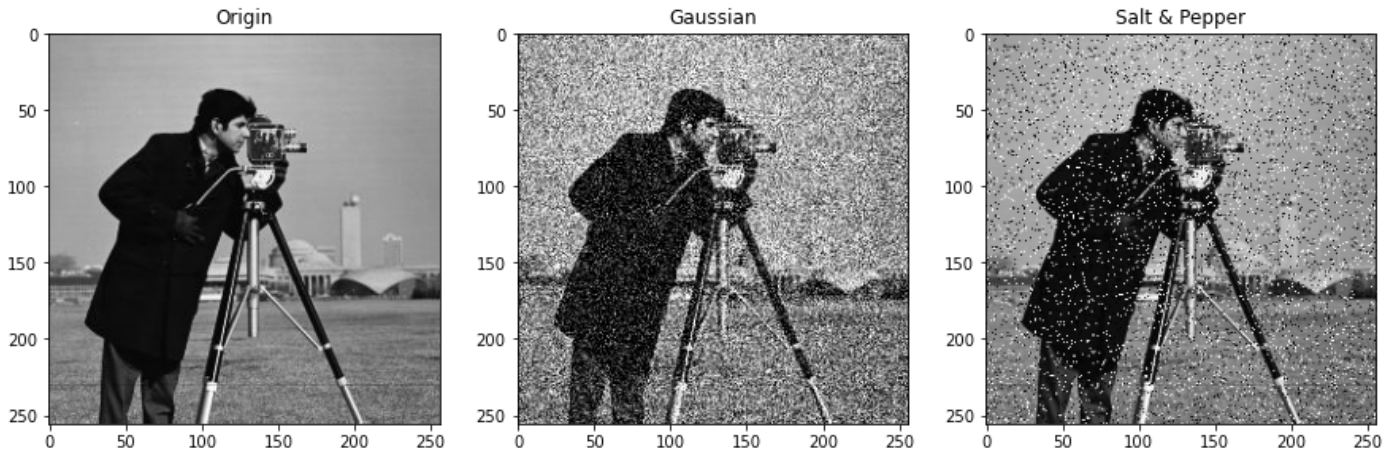
```python
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.util import random_noise
from skimage.metrics import peak_signal_noise_ratio
from skimage.util import img_as_ubyte
import os


image = img_as_ubyte(imread(os.path.join(os.getcwd(),"../cameraman.jpg"), as_gray=True))

gaussian_image = img_as_ubyte(random_noise(image, mode='gaussian', seed=1, var=0.05, mean=0))
sp_image = img_as_ubyte(random_noise(image, mode="s&p", amount=0.1, salt_vs_pepper=0.5))

g_PSNR = peak_signal_noise_ratio(image_true=image, image_test=gaussian_image)
sp_PSNR = peak_signal_noise_ratio(image_true=image, image_test=sp_image)

print(f"Gaussian PSNR:{g_PSNR:.2f} , Salt & Pepper PSNR:{sp_PSNR:.2f}")


plt.figure(figsize=(15,7))

plt.subplot(131), plt.imshow(image, cmap='gray'), plt.title('Origin')
plt.subplot(132), plt.imshow(gaussian_image, cmap='gray'), plt.title('Gaussian')
plt.subplot(133), plt.imshow(sp_image, cmap='gray'), plt.title('Salt & Pepper')

plt.savefig("1-result.jpg")

plt.show()
```

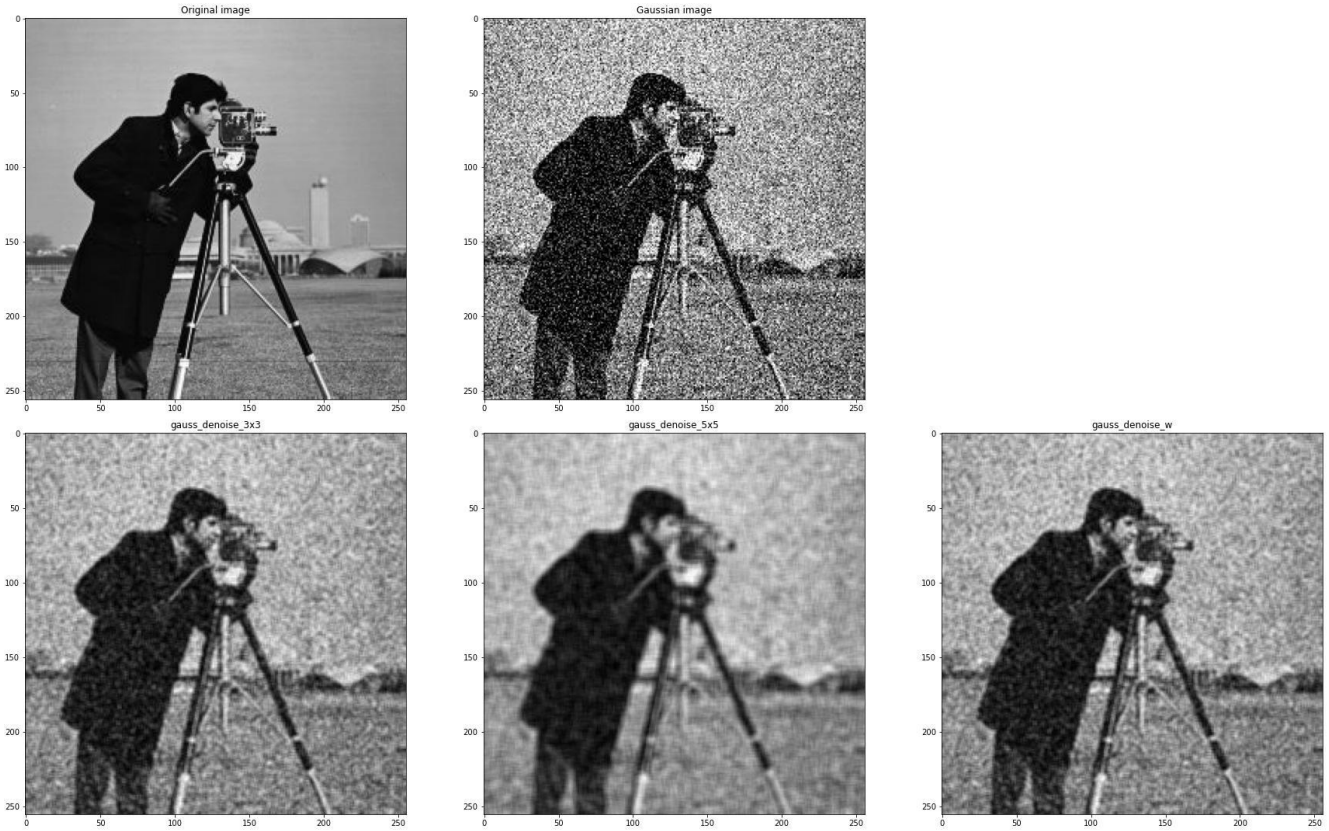
Gaussian PSNR:13.93 , Salt & Pepper PSNR:15.04



1-result.jpg

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.util import random_noise
from skimage.metrics import peak_signal_noise_ratio
from skimage.util import img_as_ubyte
from skimage.filters.rank import mean


image = img_as_ubyte(imread(os.path.join(os.getcwd(),'../cameraman.jpg'), as_gray=True))

gaussian_image = img_as_ubyte(random_noise(image, mode='gaussian', seed=1, var=0.05, mean=0))
sp_image = img_as_ubyte(random_noise(image, mode="s&p", amount=0.1, salt_vs_pepper=0.5))

box_filter_3x3 = (1 / 9) * np.ones((3, 3))
box_filter_5x5 = (1 / 25) * np.ones((5, 5))
weighted_avg = (1 / 16) * np.array([[1, 2, 1],
                                    [2, 4, 2],
                                    [1, 2, 1]])

gauss_denoise_3x3 = mean(gaussian_image, box_filter_3x3)
gauss_denoise_5x5 = mean(gaussian_image, box_filter_5x5)
gauss_denoise_w = mean(gaussian_image, weighted_avg)

g_d_3x3_PSNR = peak_signal_noise_ratio(image_true=image,image_test=gauss_denoise_3x3)
g_d_5x5_PSNR = peak_signal_noise_ratio(image_true=image,image_test=gauss_denoise_5x5)
g_d_w_PSNR = peak_signal_noise_ratio(image_true=image,image_test=gauss_denoise_w)

print(f"Gaussian Denoise 3x3 PNSR:{g_d_3x3_PSNR:.2f}")
print(f"Gaussian Denoise 5x5 PNSR:{g_d_5x5_PSNR:.2f}")
print(f"Gaussian Denoise Weighted Average PNSR:{g_d_w_PSNR:.2f}")

plt.figure(figsize=(25,15))
plt.subplot(2,3,1), plt.imshow(image, cmap="gray"), plt.title("Original image")
plt.subplot(2,3,2), plt.imshow(gaussian_image, cmap="gray"), plt.title("Gaussian image")
plt.subplot(2,3,4), plt.imshow(gauss_denoise_3x3, cmap="gray"), plt.title("gauss_denoise_3x3")
plt.subplot(2,3,5), plt.imshow(gauss_denoise_5x5, cmap="gray"), plt.title("gauss_denoise_5x5")
plt.subplot(2,3,6), plt.imshow(gauss_denoise_w, cmap="gray"), plt.title("gauss_denoise_w")
plt.tight_layout()
plt.savefig('2-result-g.jpg')
plt.show()
```

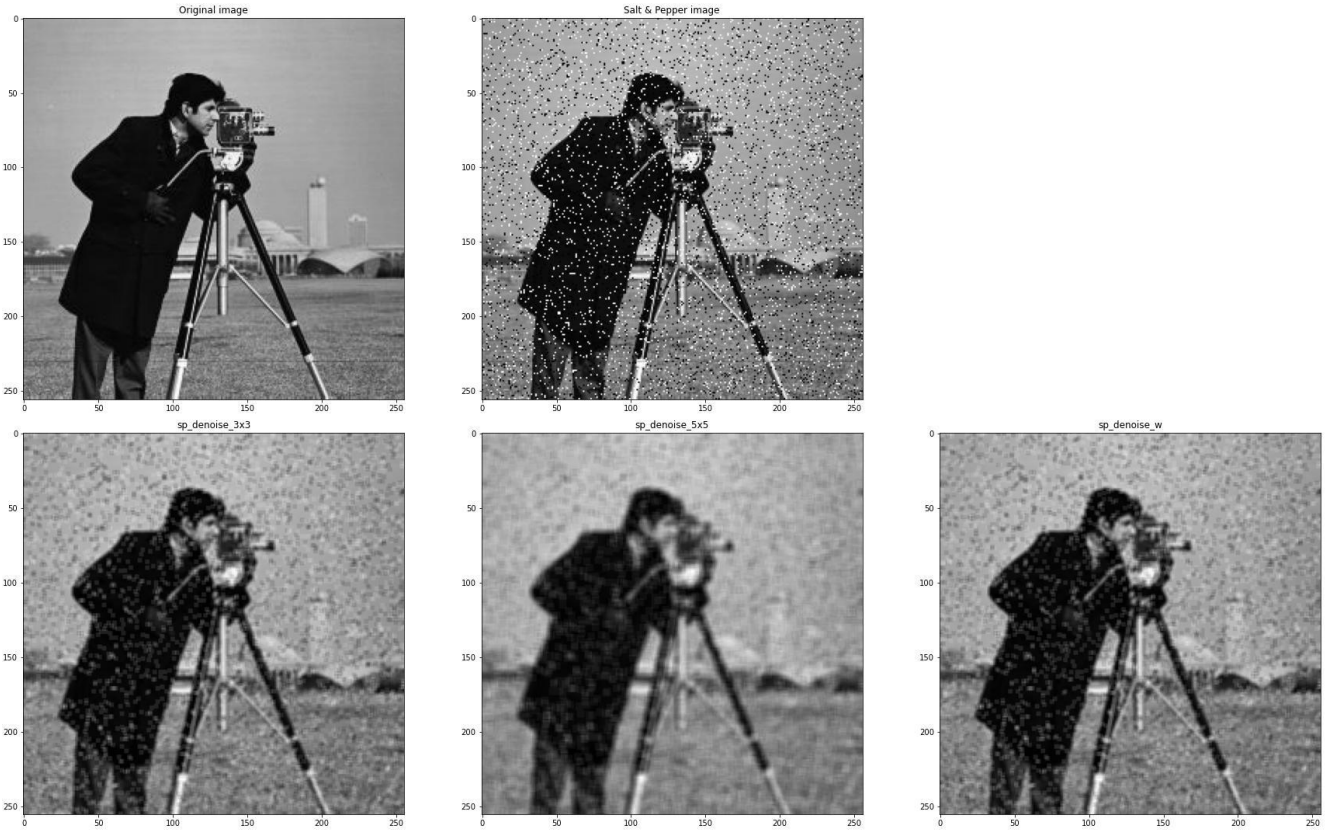
Gaussian Denoise 3x3 PNSR:20.94
Gaussian Denoise 5x5 PNSR:21.13
Gaussian Denoise Weighted Average PNSR:20.94



2-result-g.jpg

```
sp_denoise_3x3 = mean(sp_image, box_filter_3x3)
sp_denoise_5x5 = mean(sp_image, box_filter_5x5)
sp_denoise_w = mean(sp_image, weighted_avg)

sp_d_3x3_PSNR = peak_signal_noise_ratio(image_true=image, image_test=sp_denoise_3x3)
sp_d_5x5_PSNR = peak_signal_noise_ratio(image_true=image, image_test=sp_denoise_5x5)
sp_d_w_PSNR = peak_signal_noise_ratio(image_true=image, image_test=sp_denoise_w)

print(f"Salt & Pepper Denoise 3x3 PNSR:{sp_d_3x3_PSNR:.2f}")
print(f"Salt & Pepper Denoise 5x5 PNSR:{sp_d_5x5_PSNR:.2f}")
print(f"Salt & Pepper Denoise Weighted Average PNSR:{sp_d_w_PSNR:.2f}")

plt.figure(figsize=(25,15))
plt.subplot(2,3,1), plt.imshow(image, cmap="gray"), plt.title("Original image")
plt.subplot(2,3,2), plt.imshow(sp_image, cmap="gray"), plt.title("Salt & Pepper image")
plt.subplot(2,3,4), plt.imshow(sp_denoise_3x3, cmap="gray"), plt.title("sp_denoise_3x3")
plt.subplot(2,3,5), plt.imshow(sp_denoise_5x5, cmap="gray"), plt.title("sp_denoise_5x5")
plt.subplot(2,3,6), plt.imshow(sp_denoise_w, cmap="gray"), plt.title("sp_denoise_w")
plt.tight_layout()
plt.savefig('2-result-sp.jpg')
plt.show()
```

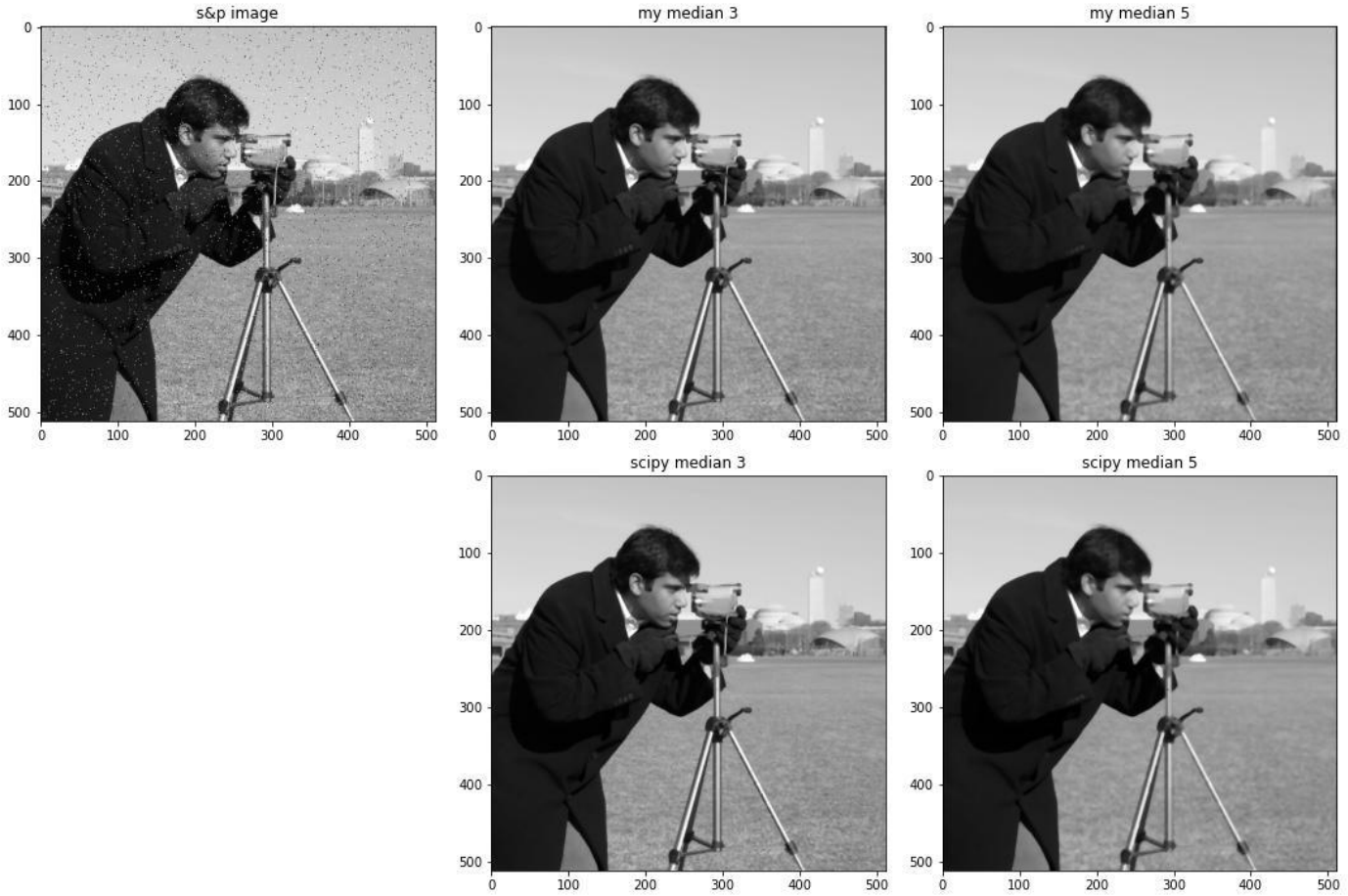
Salt & Pepper Denoise 3x3 PNSR:21.81
Salt & Pepper Denoise 5x5 PNSR:21.59
Salt & Pepper Denoise Weighted Average PNSR:21.81



2-result-sp.jpg

```
import matplotlib.pyplot as plt
from skimage import data
from skimage.util import img_as_ubyte
from skimage.util import random_noise
from scipy import ndimage
import numpy as np



def my_median_filter(data, filter_size):
    temp = []
    indexer = filter_size // 2
    data_final = []
    data_final = np.zeros((len(data),len(data[0])))
    for i in range(len(data)):

        for j in range(len(data[0])):

            for z in range(filter_size):
                if i + z - indexer < 0 or i + z - indexer > len(data) - 1:
                    for c in range(filter_size):
                        temp.append(0)
                else:
                    if j + z - indexer < 0 or j + indexer > len(data[0]) - 1:
                        temp.append(0)
                    else:
                        for k in range(filter_size):
                            temp.append(data[i + z - indexer][j + k - indexer])

            temp.sort()
            data_final[i][j] = temp[len(temp) // 2]
            temp = []
    return data_final


image = data.camera()
sp_image = img_as_ubyte(random_noise(image, mode="s&p", amount=0.02, salt_vs_pepper=0.5))

my_med_3 = my_median_filter(sp_image, 3)
scipy_med_3 = ndimage.median_filter(sp_image,size=3)

my_med_5 = my_median_filter(sp_image, 5)
scipy_med_5 = ndimage.median_filter(sp_image,size=5)
```

```
plt.figure(figsize=(15,10))
plt.subplot(2,3,1), plt.imshow(sp_image, cmap="gray"), plt.title("s&p image")
plt.subplot(2,3,2), plt.imshow(my_med_3, cmap="gray"), plt.title("my median 3")
plt.subplot(2,3,5), plt.imshow(scipy_med_3, cmap="gray"), plt.title("scipy median 3")
plt.subplot(2,3,3), plt.imshow(my_med_5, cmap="gray"), plt.title("my median 5")
plt.subplot(2,3,6), plt.imshow(scipy_med_5, cmap="gray"), plt.title("scipy median 5")
plt.tight_layout()
plt.savefig('3-result.jpg')
plt.show()
```



3-result.jpg

```python
from skimage.io import imread
from skimage.util import img_as_ubyte
import os
import matplotlib.pyplot as plt
import numpy as np

from skimage.filters import laplace, sobel
from skimage.filters.rank import mean
from skimage.exposure import adjust_gamma


image = img_as_ubyte(imread(os.path.join(os.getcwd(),'./skeleton.tif'), as_gray=True))

laplacian = (laplace(image, ksize=3))

add_laplacian = image + laplacian

sobel_img = sobel(image)

box_filter_5x5 = (1 / 25) * np.ones((5, 5), dtype=np.float64)

average_sobel = mean(sobel_img,box_filter_5x5)

product_averageSobel_addLaplacian = np.multiply( add_laplacian ,average_sobel)
product_averageSobel_addLaplacian += np.abs(product_averageSobel_addLaplacian.min())
product_averageSobel_addLaplacian *= (255/product_averageSobel_addLaplacian.max())

add_image_product_add = image + product_averageSobel_addLaplacian

power = adjust_gamma(add_image_product_add,gamma=0.5)

plt.figure(figsize=(15,35))
plt.subplot(4,2,1), plt.imshow(image, cmap="gray"), plt.title("image")
plt.subplot(4,2,2), plt.imshow(laplacian, cmap="gray"), plt.title("laplacian mask")
plt.subplot(4,2,3), plt.imshow(add_laplacian, cmap="gray"), plt.title("add laplacian")
plt.subplot(4,2,4), plt.imshow(sobel_img, cmap="gray"), plt.title("sobel_img")
plt.subplot(4,2,5), plt.imshow(average_sobel, cmap="gray"), plt.title("average_sobel")
plt.subplot(4,2,6), plt.imshow(product_averageSobel_addLaplacian, cmap="gray"),
plt.title("product_averageSobel_addLaplacian")
plt.subplot(4,2,7), plt.imshow(add_image_product_add, cmap="gray"),
plt.title("add_image_product_add")
plt.subplot(4,2,8), plt.imshow(power, cmap="gray"), plt.title("power")
plt.tight_layout()
plt.savefig("4-result.jpg")
plt.show()
```

4-result.jpg

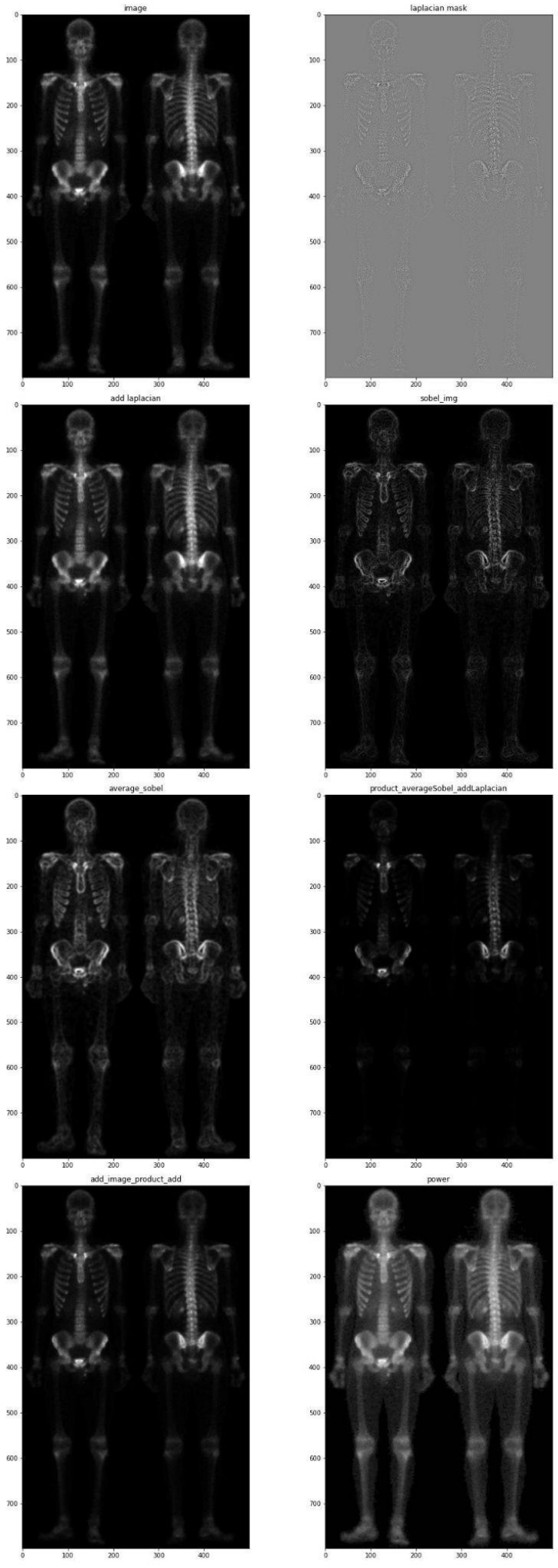


image
laplacian mask
add laplacian
sobel_img
average_sobel
product_averageSobel_addLaplacian
add_image_product_add
power