Semester I, Academic Year 2022–2023

Computational Optical Imaging

[Course Code: PYL759]

Submission: Assignment 2

Name : Mayand Dangi

Entry Number : 2019PH10637

Code Snippets

Importing Package

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import tqdm
```

Function for obtaining TV and h

Input parameter is gray-scale image $g(x, y)$ and output parameters are Total Variation TV and h defined as:

$$TV = \sum_m \sqrt{(\nabla g_x)^2 + (\nabla g_y)^2}$$

$$h = \nabla_g TV(g) = -\nabla \cdot \left(\frac{\nabla g}{\sqrt{|\nabla g|^2 + \delta^2}} \right)$$

```
'''
    @params
            g: grayscale image g(x, y)
    @return
            TV and h
'''
def getTV_and_h(g, delta = 10e-08):
```

```
    gy, gx = np.gradient(g)
    temp = np.sqrt(gx**2 + gy**2 + delta**2)
    h = -(np.gradient(gx/temp)[1] + np.gradient(gy/temp)[0])
    TV = np.sum(np.sqrt(gx**2 + gy**2))
    return TV, h
```

Function for performing iterations of the following operation

$$g \leftarrow g - 10^{-3}||g||_2 \times \frac{h}{||h||_2}$$

```
'''
    @params
        g   :   grayscale image
        it  :   number of iterations
        tau :   steps size
    @return
        g   :   denoised grayscale image
'''
def iter_g(g, it, tau):
    for i in tqdm.tqdm(range(it)):
        TV, h = getTV_and_h(g
        norm2_g = np.linalg.norm(g, ord = 'fro')
        norm2_h = np.linalg.norm(h, ord = 'fro')
        g = g - tau * norm2_g*(h/norm2_h)
    return g
```

Loading the image and converting it to grayscale

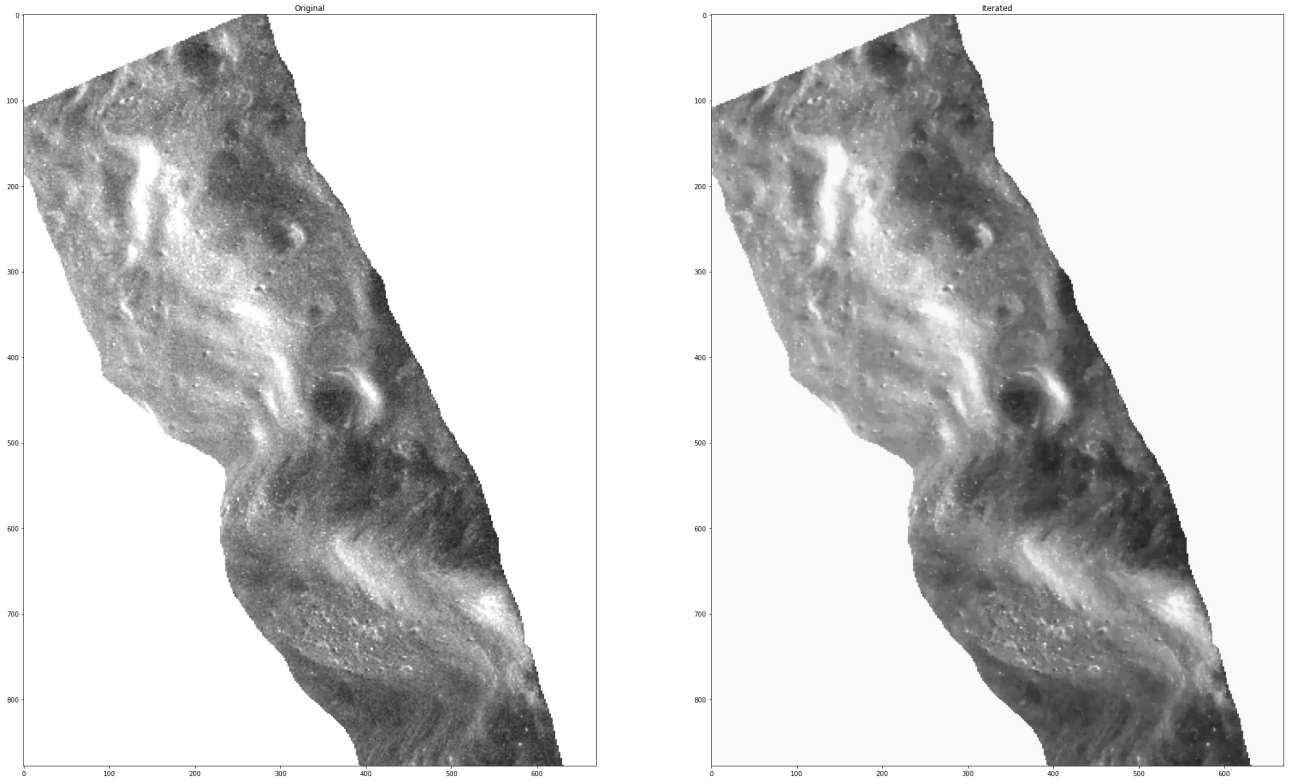
```
img = Image.open(file_location)
img = img.convert('L')
g = np.asarray(img)
```

Denoising the Image and Plotting it

```
it = 25           #number of iterations to be runned
tau = 1e-03       #step size
```

```
3
4  #denoising the image
5  img_arr_denoised = iter_g(img_arr, it, tau)
6
7  #plotting the image
8  im, ax = plt.subplots(1, 2, figsize= (35,110))
9  ax[0].imshow(img_arr, cmap='gray')
10 ax[0].set_title('Original Image')
11 ax[1].imshow(img_arr_new, cmap='gray')
12 ax[1].set_title('Iterated Image')
```

Results and Discussions



(a) (b)

Figure 1: (a) Original Image (b) Denoised Image iterated 25 times with steps size 10^{-3}

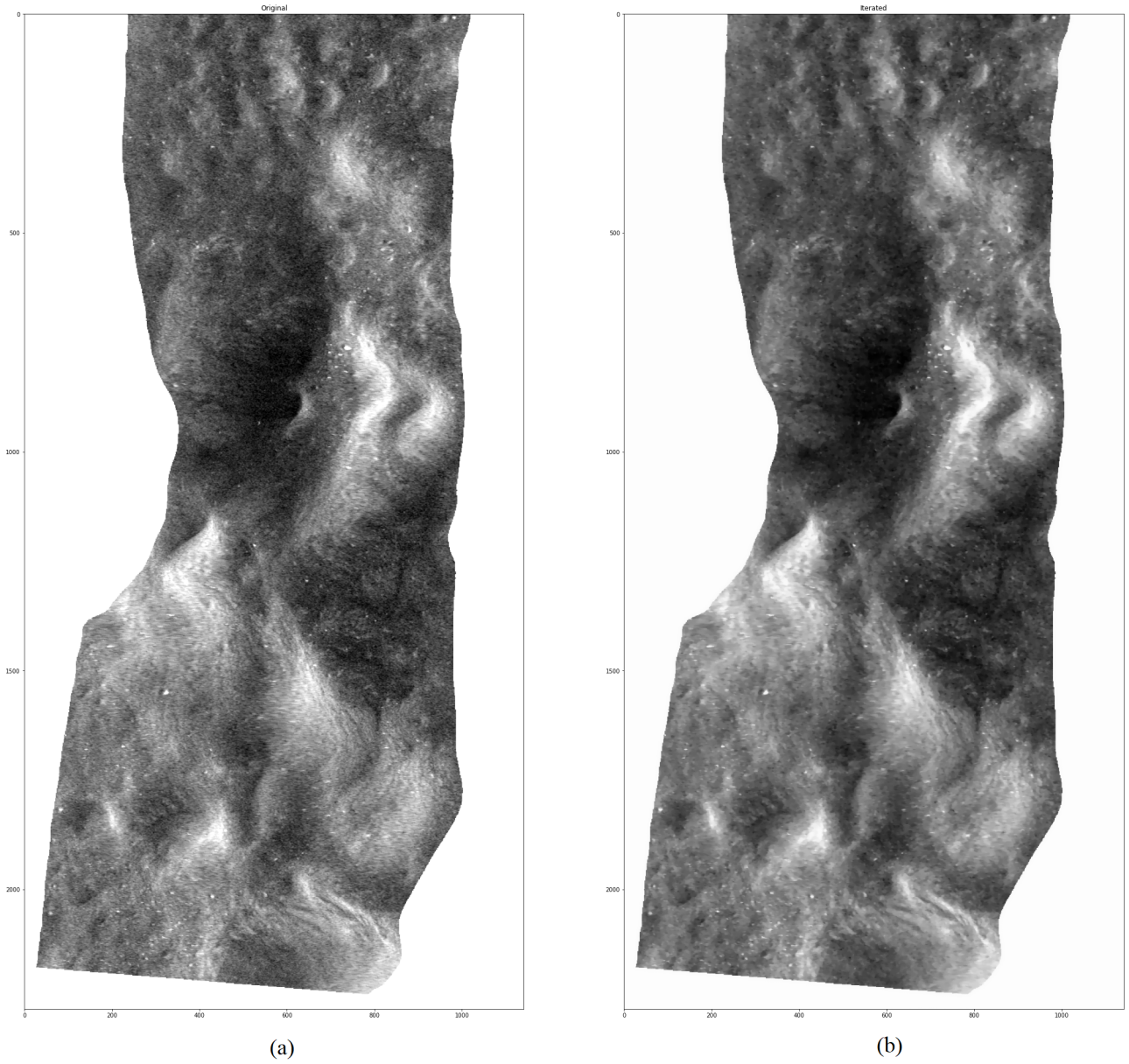


Figure 2: (a) Original Image (b) Denoised Image iterated 25 times with steps size 2×10^{-3}

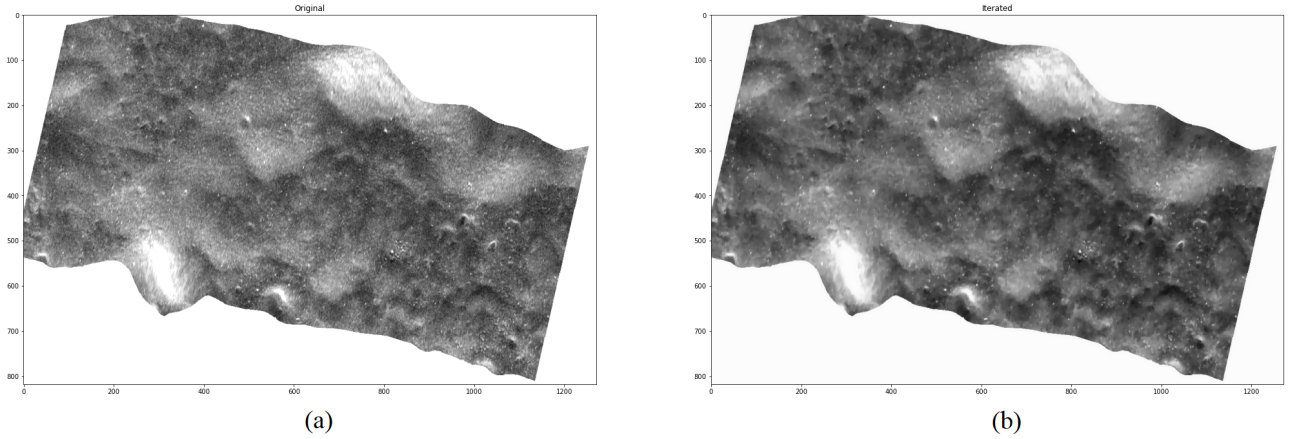


Figure 3: (a) Original Image (b) Denoised Image iterated 25 times with steps size 2×10^{-3}

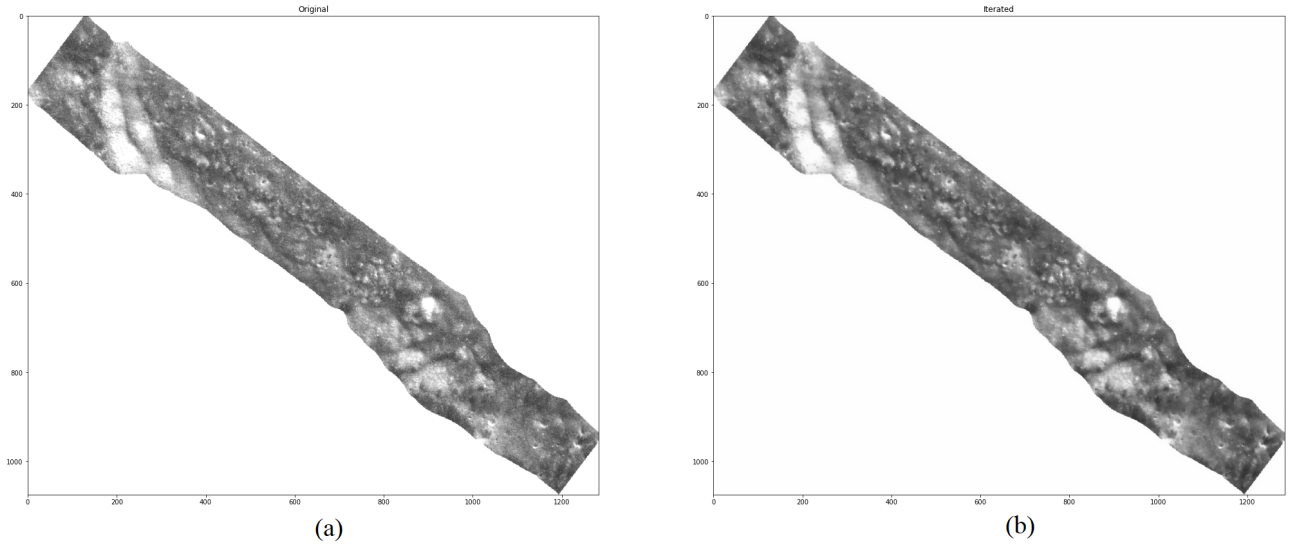


Figure 4: (a) Original Image (b) Denoised Image iterated 25 times with steps size 2×10^{-3}

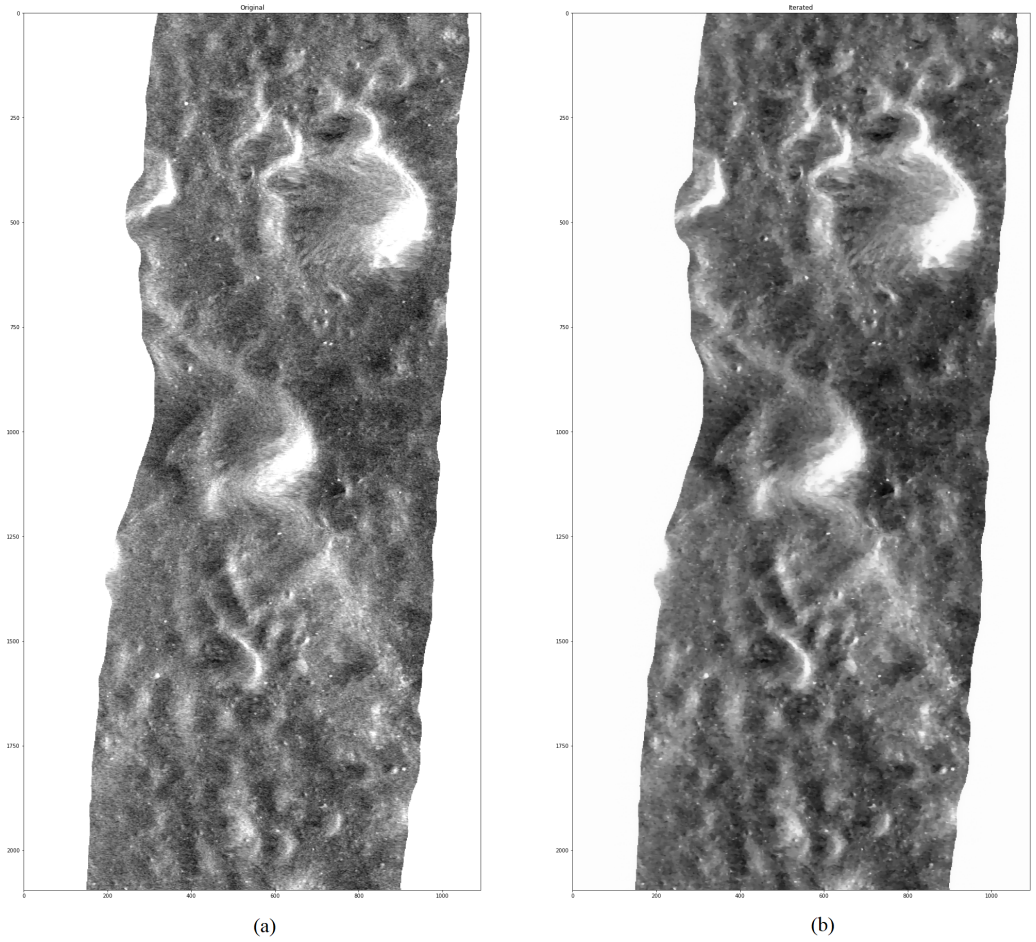


Figure 5: (a) Original Image (b) Denoised Image iterated 30 times with steps size 2×10^{-3}

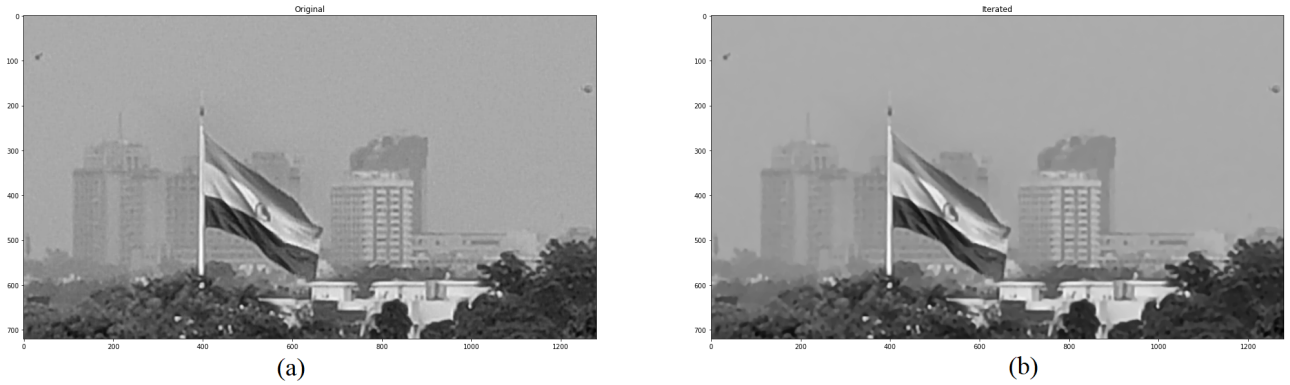


Figure 6: (a) Original Image (b) Denoised Image iterated 25 times with steps size 10^{-3}

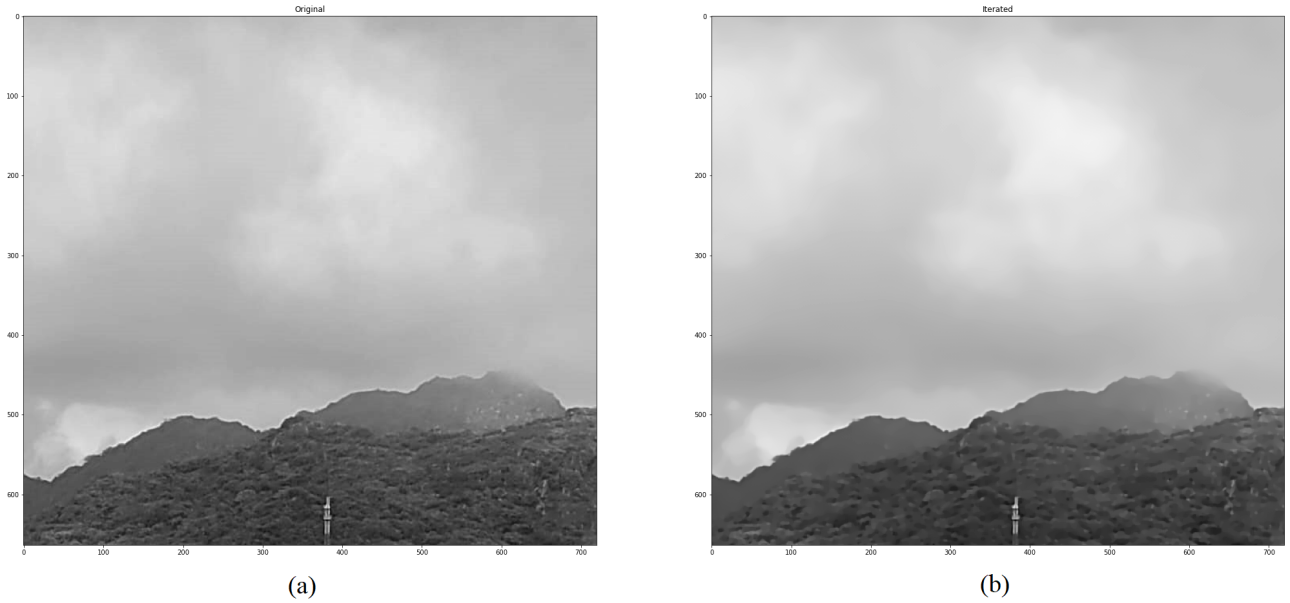


Figure 7: (a) Original Image (b) Denoised Image iterated 25 times with steps size 10^{-3}

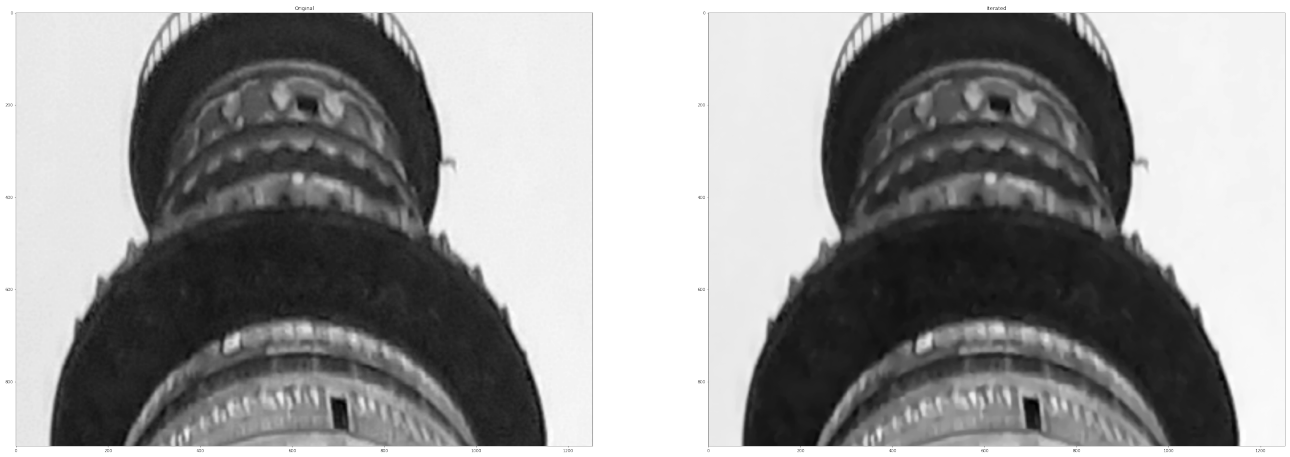


Figure 8: (a) Original Image (b) Denoised Image iterated 25 times with steps size 10^{-3}

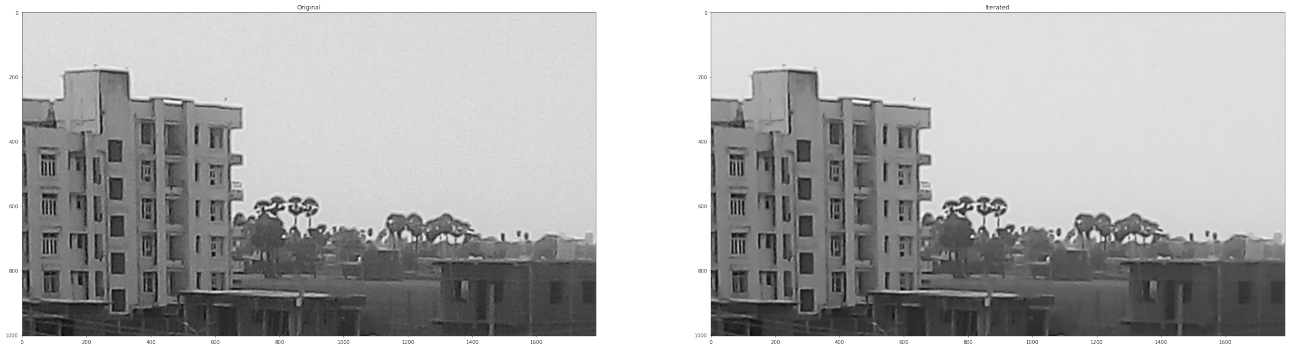


Figure 9: (a) Original Image (b) Denoised Image iterated 25 times with steps size 10^{-3}

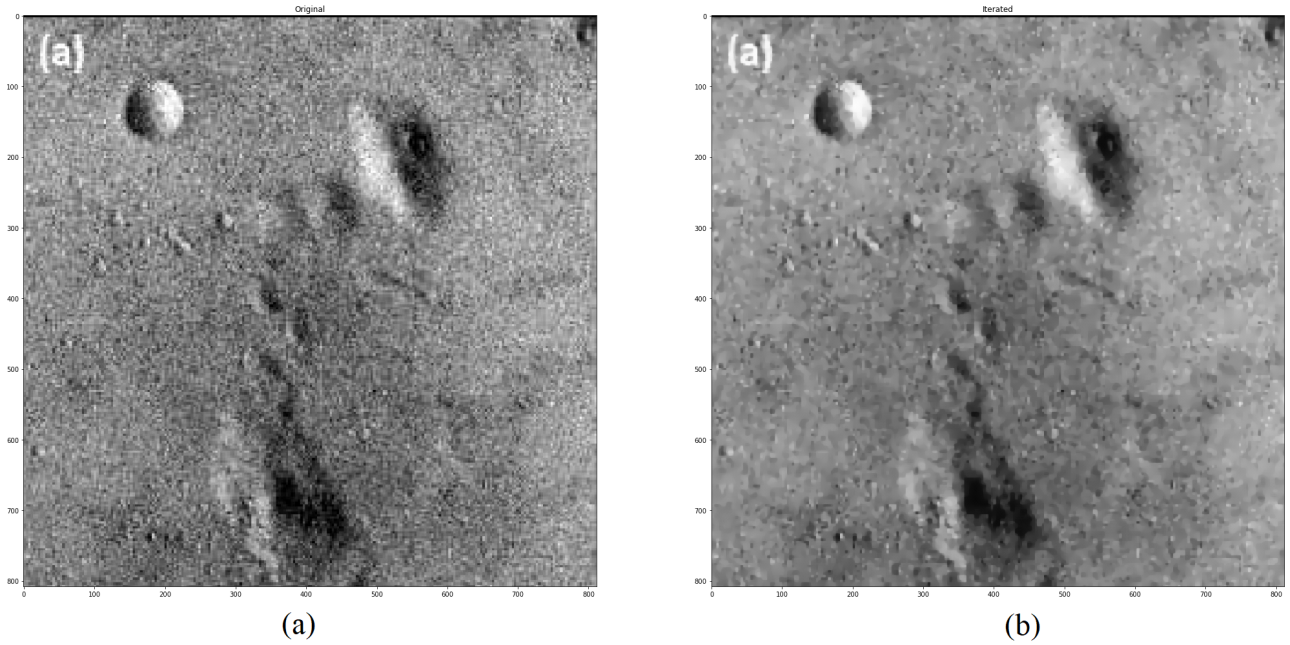


Figure 10: (a) Original Image (b) Denoised Image iterated 25 times with steps size 5×10^{-3}

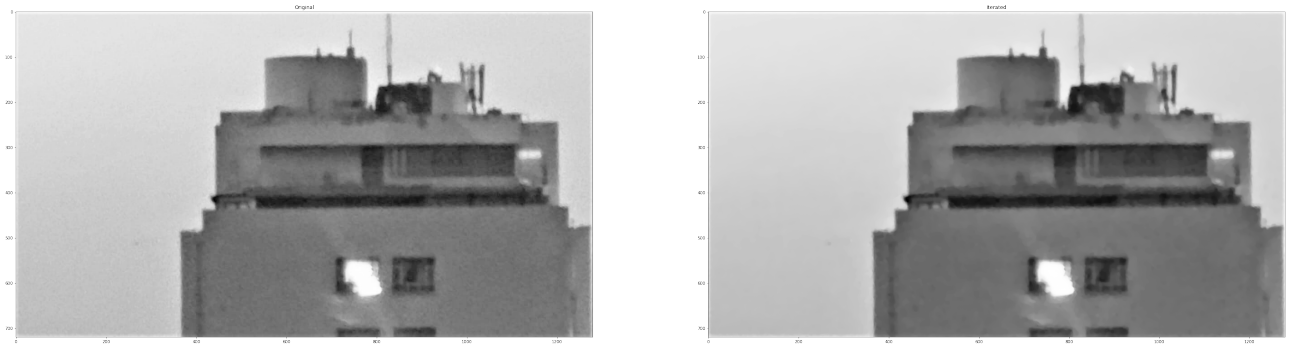


Figure 11: (a) Original Image (b) Denoised Image iterated 25 times with steps size 10^{-3}

In this assignment, we aimed for the denoising of the image. I have used different sets of images to see the effects of total variation in denoising the image. Figures 1, 2, 3, 4 and 5 are SAR images of the Moon captured by the Chandrayaan-2 DFSAR Payload. In these images, we can see the slight grainy structure that appeared in the original image, and after denoising, it has been demolished. In this case, our algorithm might detect small craters as well as noise, so we have stopped the iteration quite earlier. Figure 10 is also the image of the moon surface having large noise, which was taken from lecture slides, and after deniosing it, the noise has been suppressed. Figures 6(a), 8(a), 9(a), and 11(a) show the image's sky regime's grainy pattern clearly; after denoising, the sky appears smooth. Our algorithm suppresses the noise beacuse TV denoising diffuses regions with small oscillations more than regions with larger oscillations due to which small details are lost in the image. It preserves the edges as the edges have larger oscillations.