Segmenting out rocks from Martian surface

In [2]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import skimage
import skimage.io as io
```

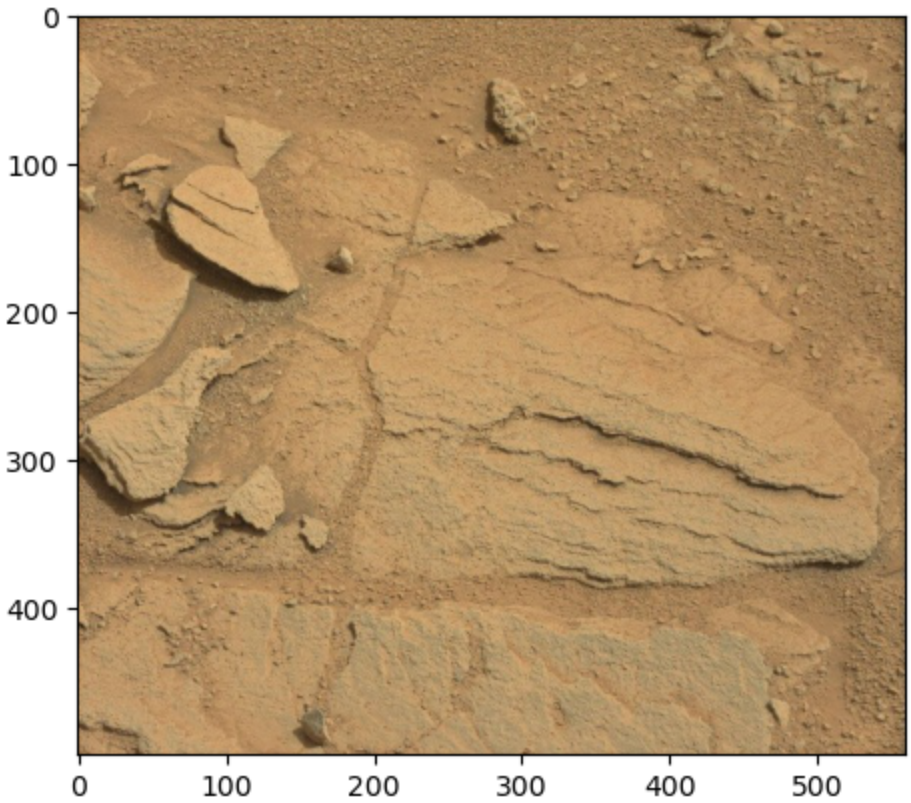
Custom functions for processing these images

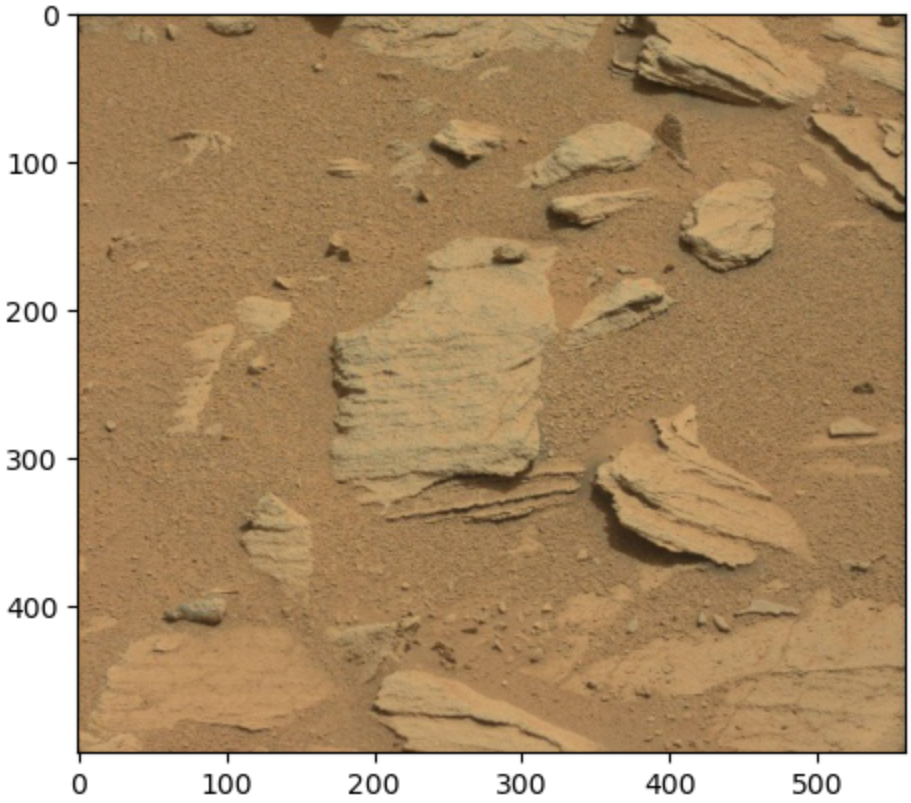
In [10]:
```
def ImageComparison(image1, image2):
    fig, ax = plt.subplots(1, 2, figsize = (24, 24))

    plt.subplot(1, 2, 1)
    plt.imshow(image1, cmap = 'gray')
    plt.title('Original Image')
    plt.xticks([])
    plt.yticks([])

    plt.subplot(1, 2, 2)
    plt.imshow(image2, cmap = 'gray')
    plt.title('Processed Image')
    plt.xticks([])
    plt.yticks([])

    plt.show()
    print('\n')

def GrayscaleImageShortenedRepresentation(image):
    ImageMatrix = list(image)
    ProcessedImageMatrix = []

    for ImageRow in ImageMatrix:
        ProcessedImageRow = []
        for pixel in ImageRow:
            PixVal = pixel[0]
            ProcessedImageRow.append(PixVal)
        ProcessedImageMatrix.append(ProcessedImageRow)

    ProcessedImage = np.array(ProcessedImageMatrix)

    return ProcessedImage

def GrayscaleImageAveragePixelValue(image):
    Image = GrayscaleImageShortenedRepresentation(image)

    dimensions = Image.shape
    b = dimensions[0]
    l = dimensions[1]
    NumberOfPixels = l * b

    PixelSum = 0

    ImageMatrix = list(Image)
```

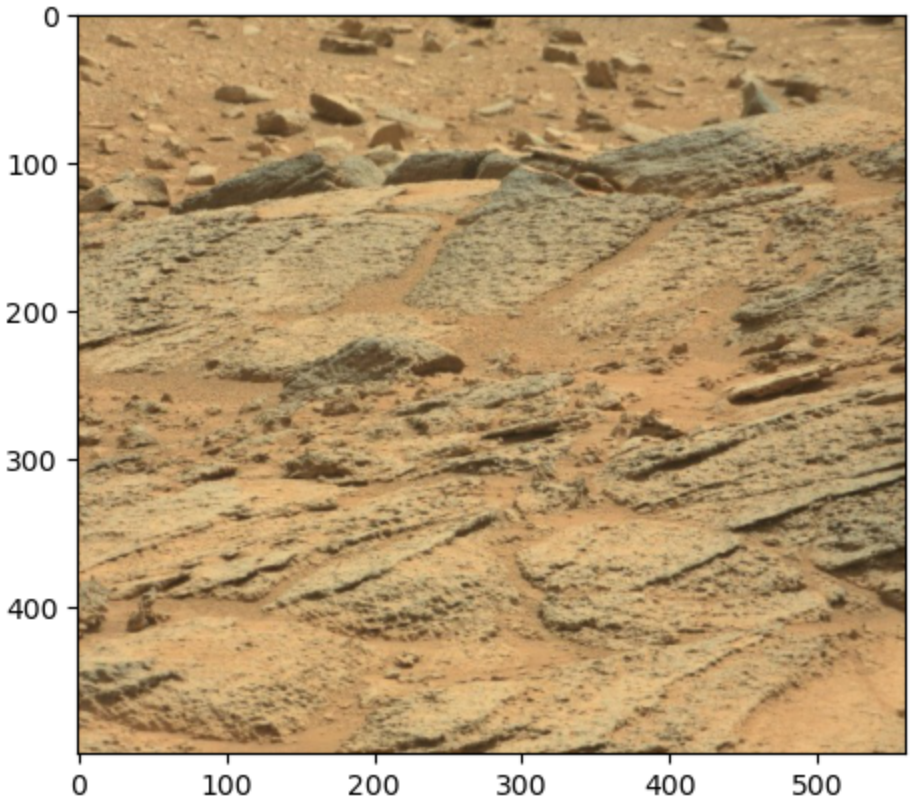
```
    for ImageRow in ImageMatrix:
        for PixVal in ImageRow:
            PixelSum = PixelSum + PixVal

    AveragePixelValue = round((PixelSum / NumberOfPixels), 4)

    return AveragePixelValue
```

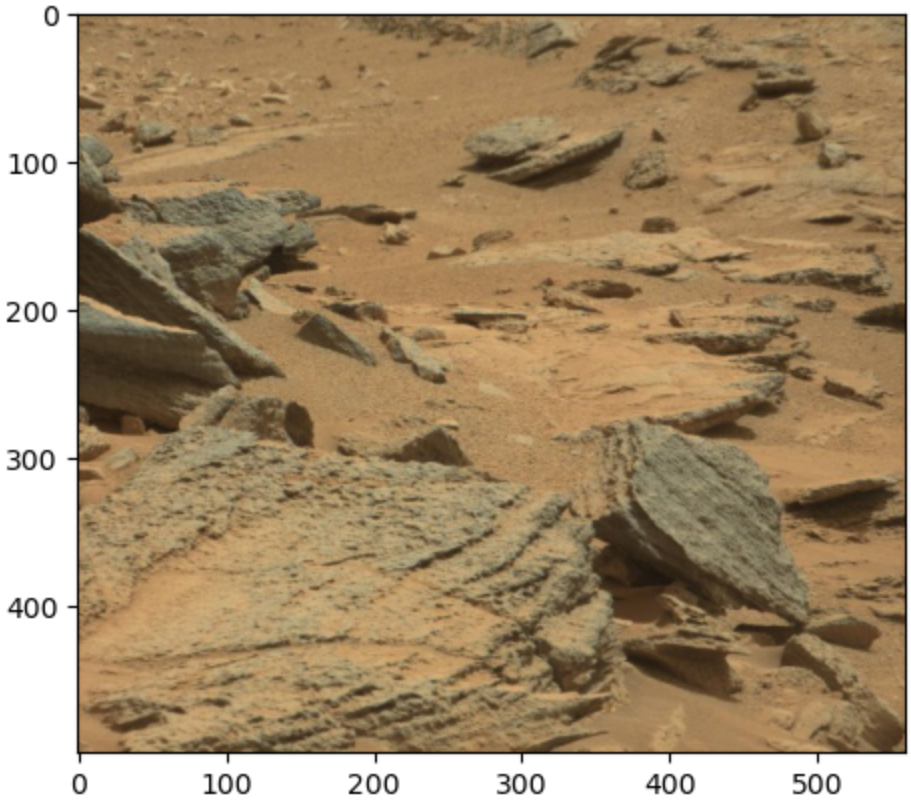
some sample images

In [7]:
```
base_path = 'C:\\Users\\ASHOK\\Desktop\\Python files\\Analysis of Martian surface and

img1 = io.imread(base_path + '1.jpg')
img2 = io.imread(base_path + '2.jpg')
img3 = io.imread(base_path + '3.jpg')
img4 = io.imread(base_path + '6.jpg')

plt.imshow(img1)
plt.show()
plt.imshow(img2)
plt.show()
plt.imshow(img3)
plt.show()
plt.imshow(img4)
plt.show()
```




0
100
200
300
400
0
100
200
300
400
500

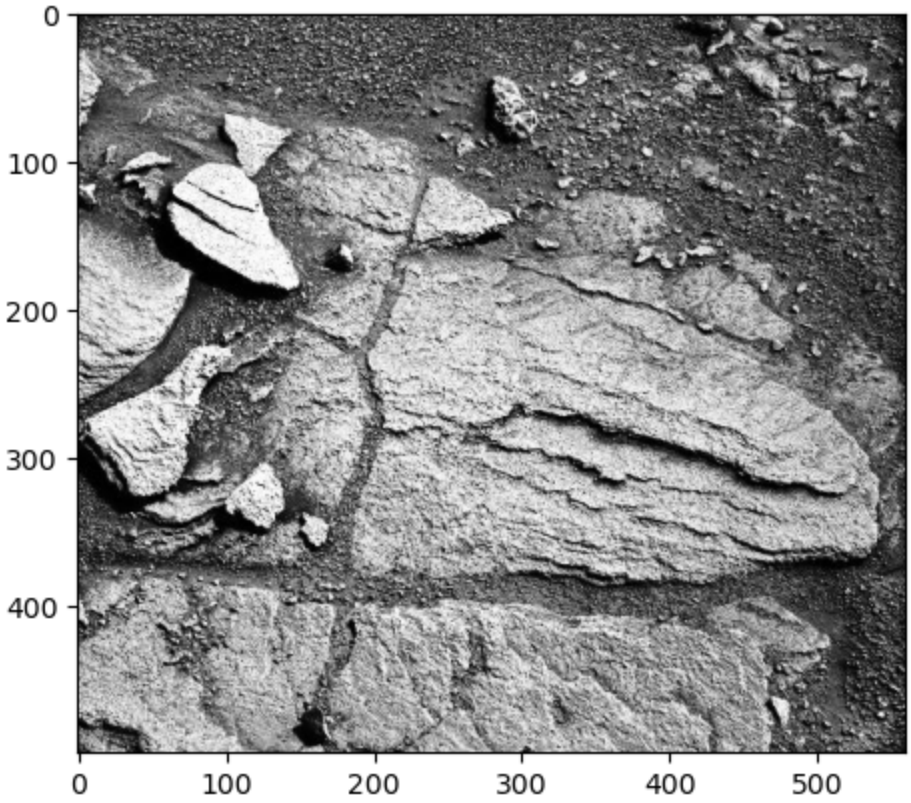

0
100
200
300
400
0
100
200
300
400
500

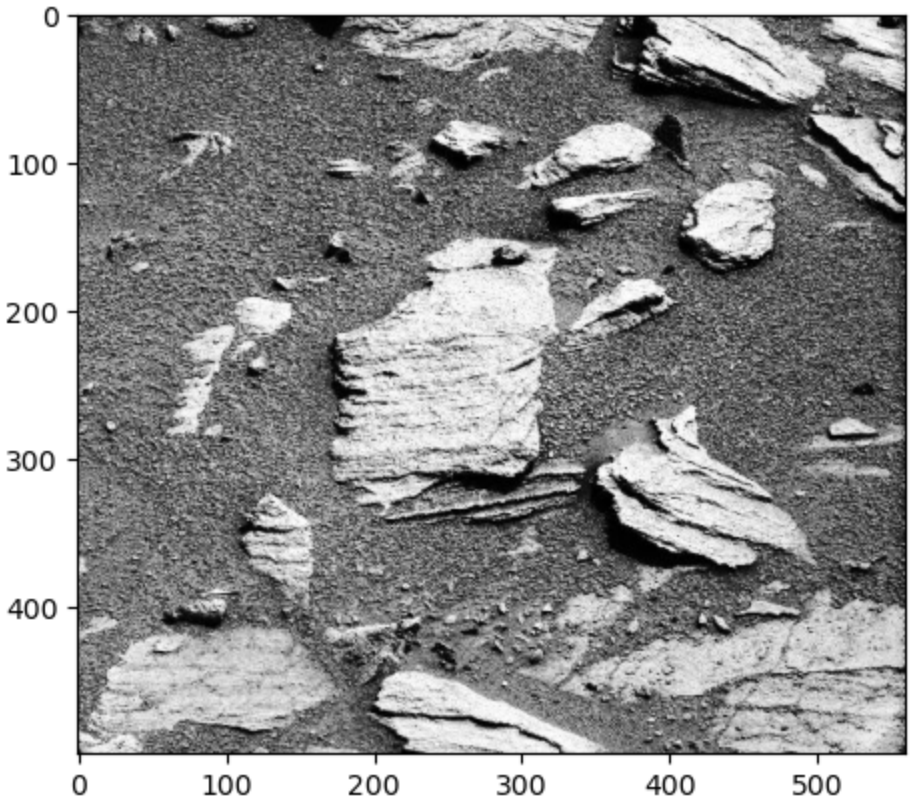


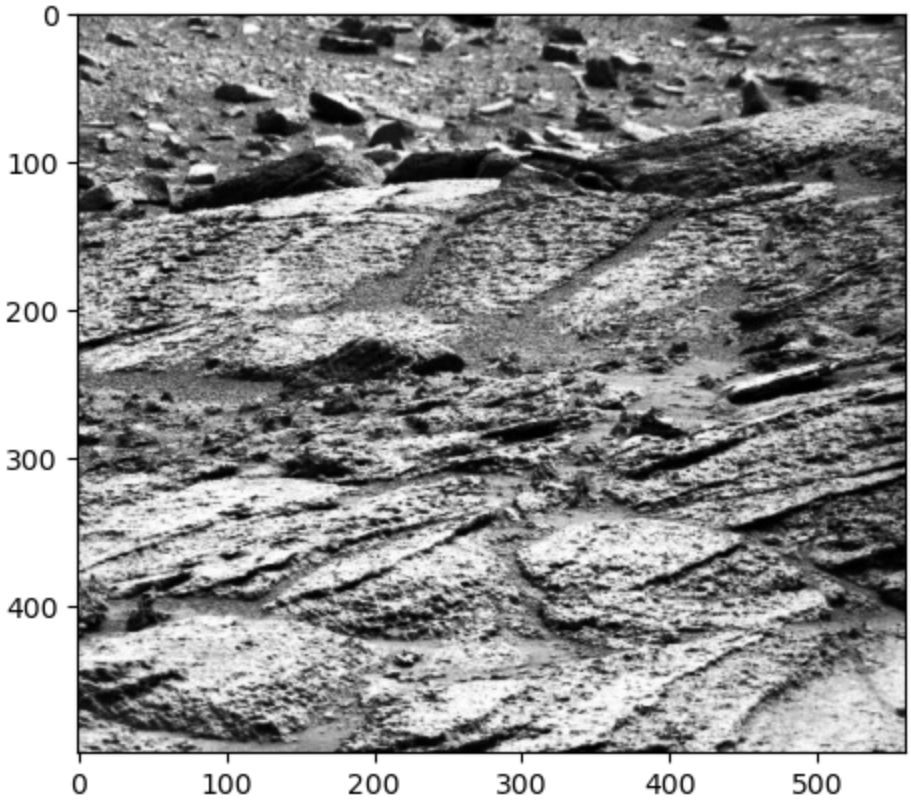
Converting these images to Grayscale and applying Histogram Equalization

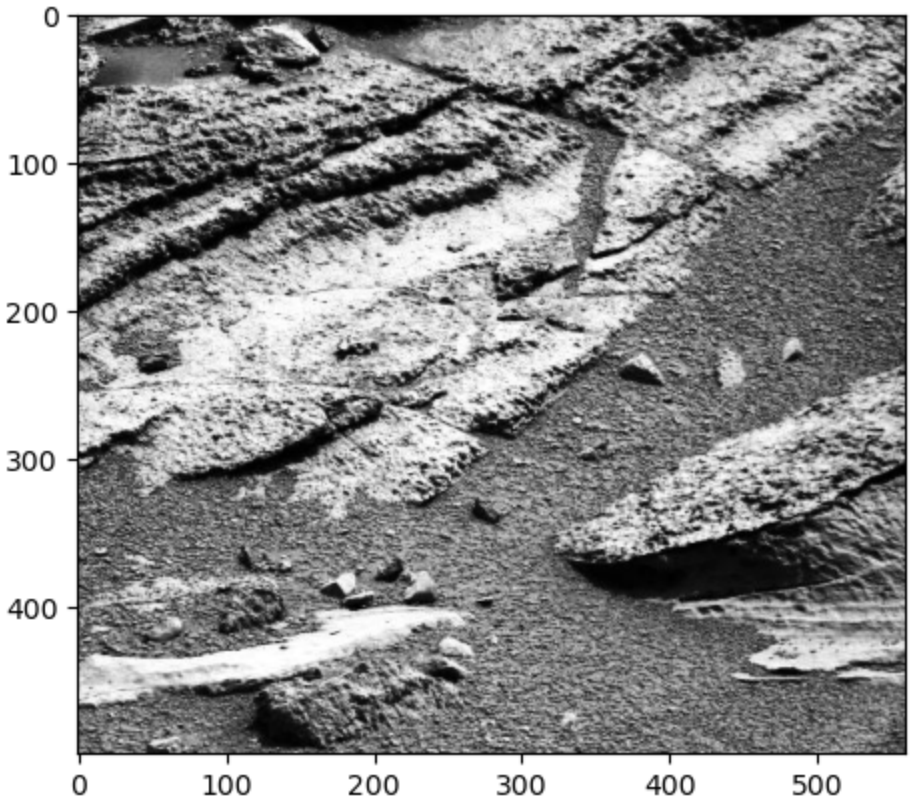
Displaying those images

In [9]:
```
base_path = 'C:\\Users\\ASHOK\\Desktop\\Python files\\Analysis of Martian surface and

img1 = io.imread(base_path + '1.jpg')
img2 = io.imread(base_path + '2.jpg')
img3 = io.imread(base_path + '3.jpg')
img4 = io.imread(base_path + '6.jpg')

plt.imshow(img1, cmap = 'gray')
plt.show()
plt.imshow(img2, cmap = 'gray')
plt.show()
plt.imshow(img3, cmap = 'gray')
plt.show()
plt.imshow(img4, cmap = 'gray')
plt.show()
```


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500

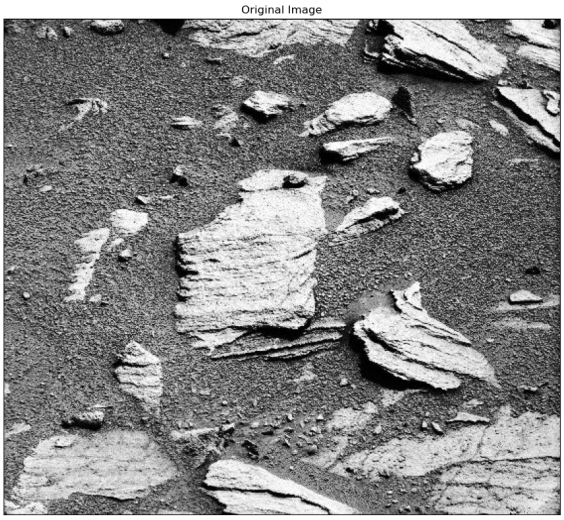


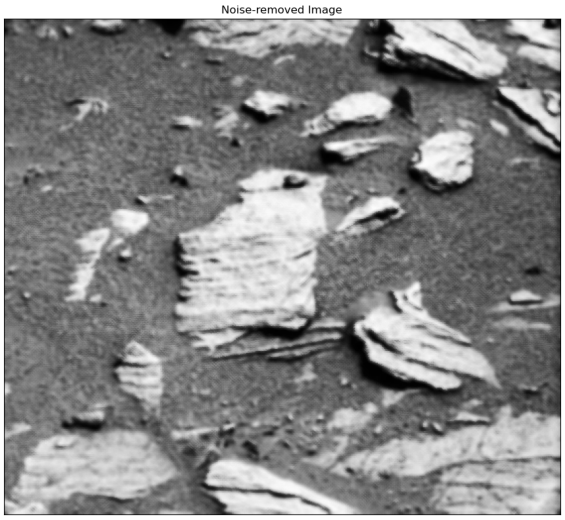


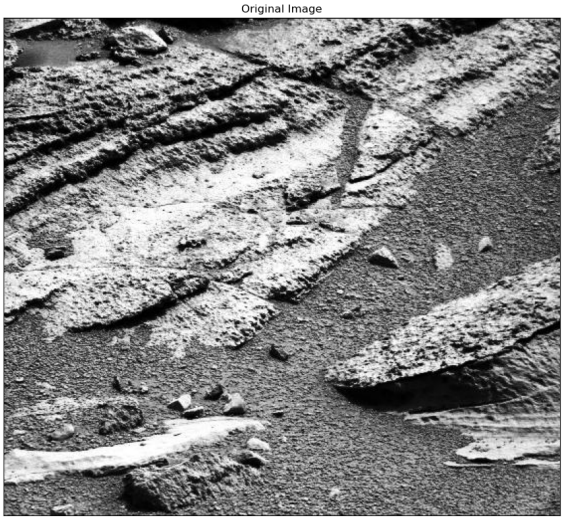
Applying Bilateral Filtering to these images

In [15]:
```
BilateralFilterImage1 = cv.bilateralFilter(img1, 7, 200, 200)
BilateralFilterImage2 = cv.bilateralFilter(img2, 7, 200, 200)
BilateralFilterImage3 = cv.bilateralFilter(img3, 7, 200, 200)
```

```
BilateralFilterImage4 = cv.bilateralFilter(img4, 7, 200, 200)

ImageComparison(img1, BilateralFilterImage1)
ImageComparison(img2, BilateralFilterImage2)
ImageComparison(img3, BilateralFilterImage3)
ImageComparison(img4, BilateralFilterImage4)
```

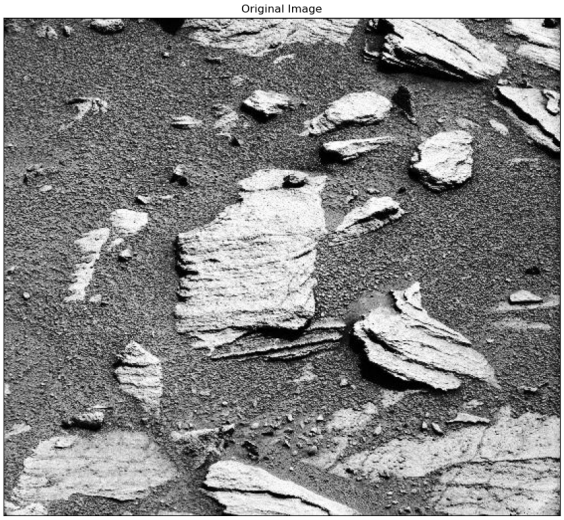


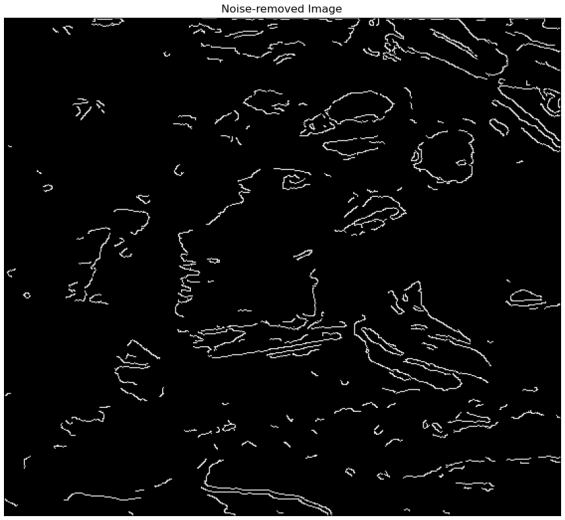


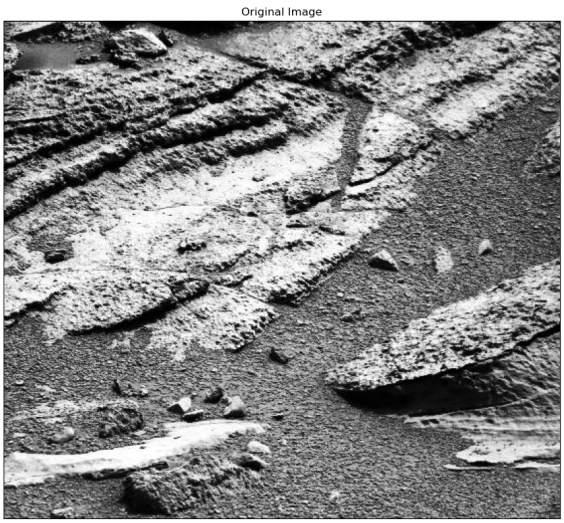


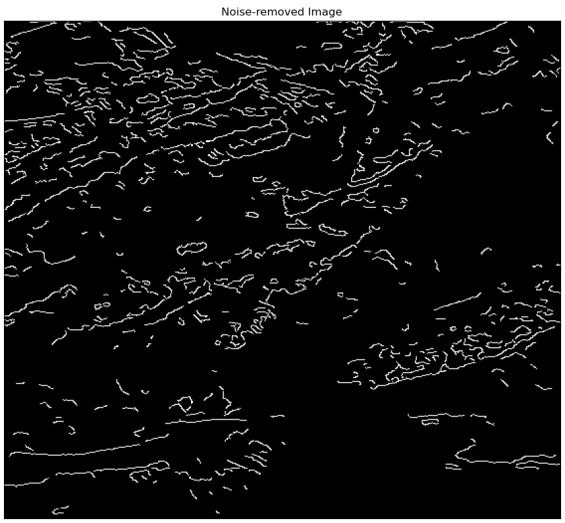
Applying Canny Edge Detection algorithm

In [17]:
```
Canny1 = cv.Canny(BilateralFilterImage1, 200, 300)
Canny2 = cv.Canny(BilateralFilterImage2, 200, 300)
Canny3 = cv.Canny(BilateralFilterImage3, 200, 300)
Canny4 = cv.Canny(BilateralFilterImage4, 200, 300)

ImageComparison(img1, Canny1)
ImageComparison(img2, Canny2)
ImageComparison(img3, Canny3)
ImageComparison(img4, Canny4)
```





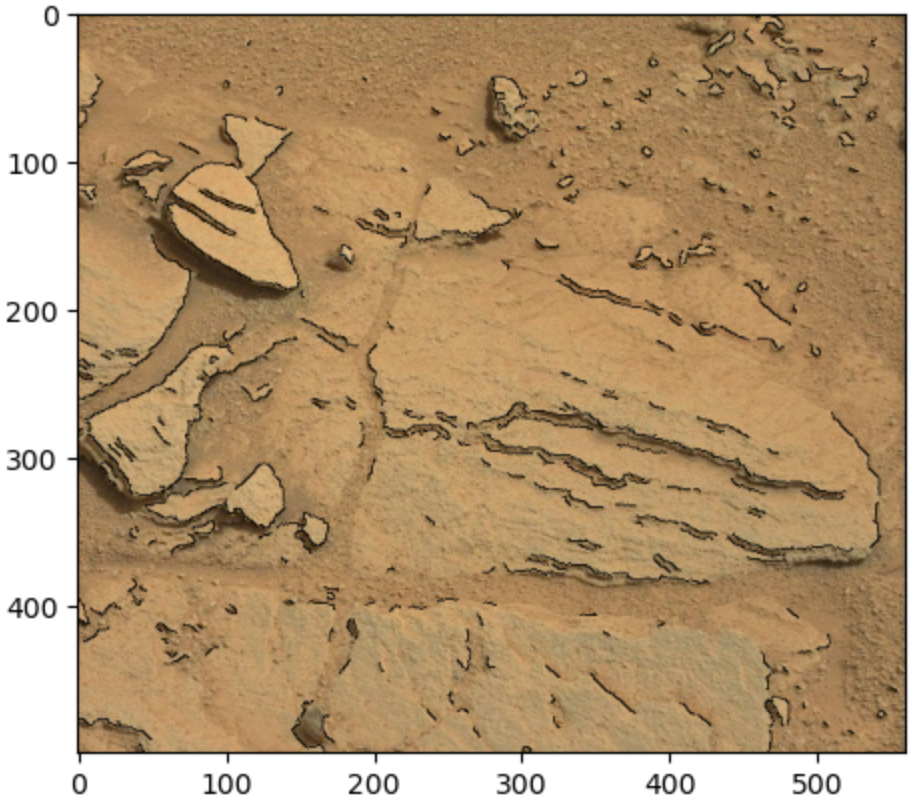


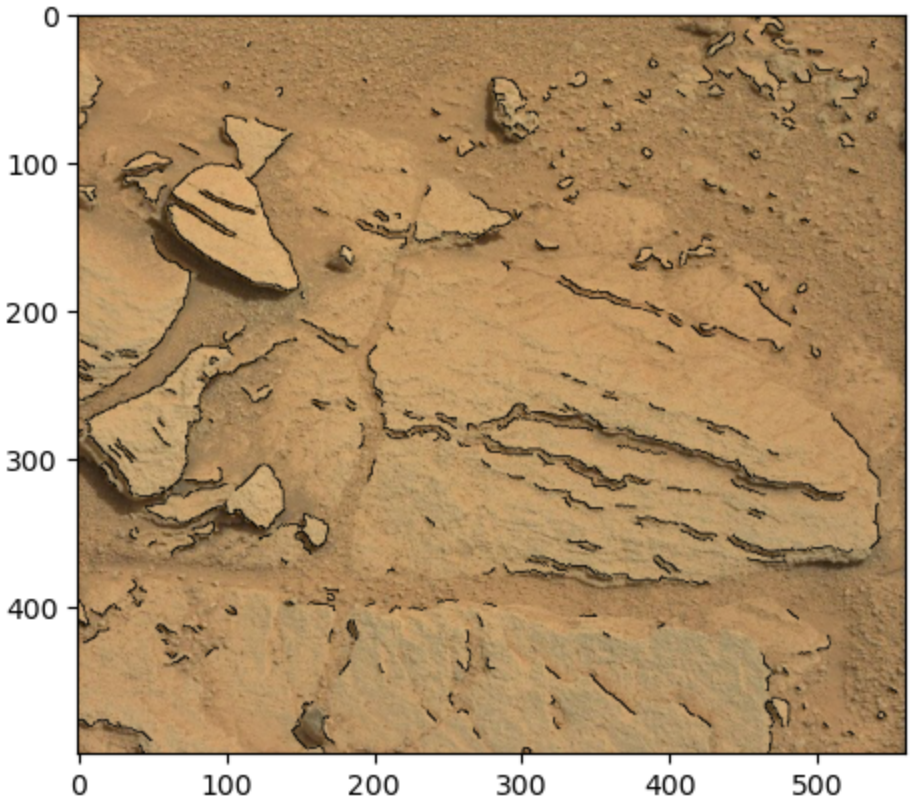


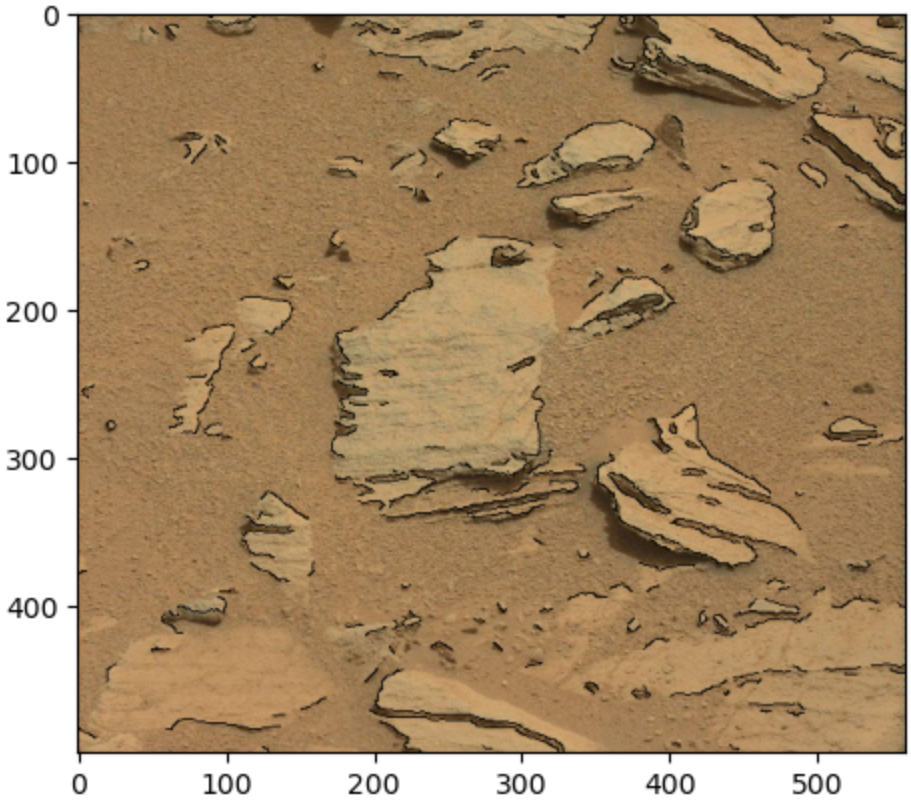
Superimposing edges

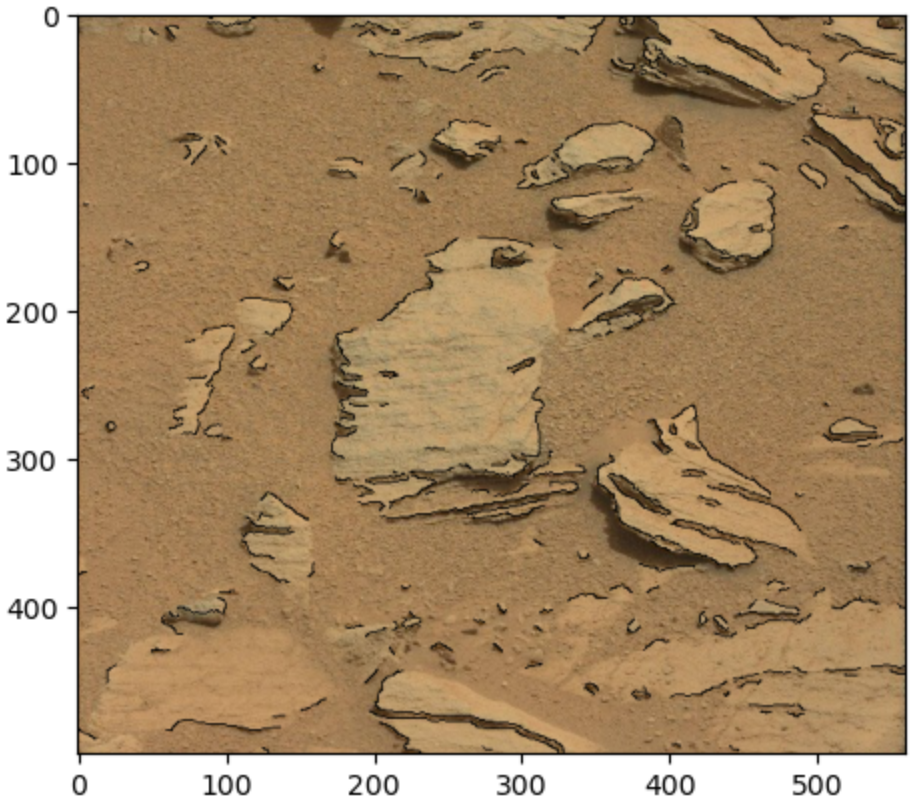
In [22]:
```
def EdgeSuperimposition(image, CannyEdgeImage):
    Dimensions = image.shape
    l = Dimensions[1]
```

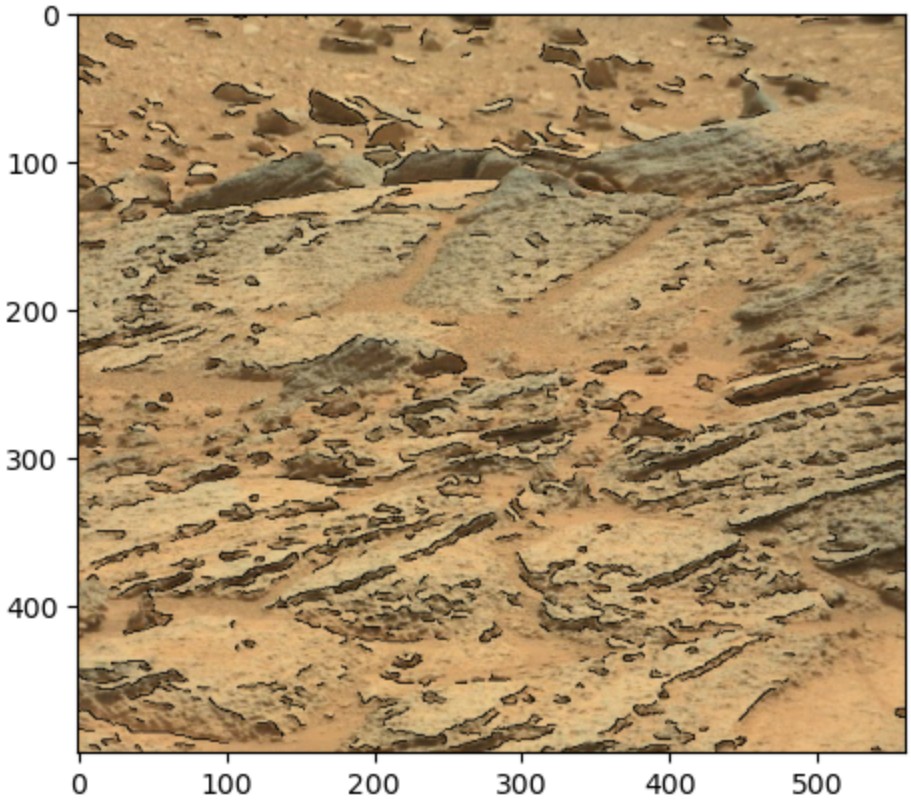
```
    b = Dimensions[0]

    ImageMatrix = list(image)
    CannyEdgeImageMatrix = list(CannyEdgeImage)

    for i in range(0, b):
        for j in range(0, l):
            if CannyEdgeImageMatrix[i][j] == 255:
                ImageMatrix[i][j] = [0, 0, 0]

    ProcessedImage = np.array(ImageMatrix)

    return ProcessedImage

def AltImageComparison(image1, image2):
    fig, ax = plt.subplots(1, 2, figsize = (24, 24))

    plt.subplot(1, 2, 1)
    plt.imshow(image1)
    plt.title('Original Image')
    plt.xticks([])
    plt.yticks([])

    plt.subplot(1, 2, 2)
    plt.imshow(image2)
    plt.title('Processed Image')
    plt.xticks([])
    plt.yticks([])

    plt.show()
    print('\n')

Img1 = io.imread('C://Users//ASHOK//Desktop//Python files//Analysis of Martian surface
Img2 = io.imread('C://Users//ASHOK//Desktop//Python files//Analysis of Martian surface
Img3 = io.imread('C://Users//ASHOK//Desktop//Python files//Analysis of Martian surface
Img4 = io.imread('C://Users//ASHOK//Desktop//Python files//Analysis of Martian surface

I1 = EdgeSuperimposition(Img1, Canny1)
I2 = EdgeSuperimposition(Img2, Canny2)
I3 = EdgeSuperimposition(Img3, Canny3)
I4 = EdgeSuperimposition(Img4, Canny4)

plt.imshow(I1)
plt.show()

plt.imshow(I2)
plt.show()

plt.imshow(I3)
plt.show()

plt.imshow(I4)
plt.show()
```

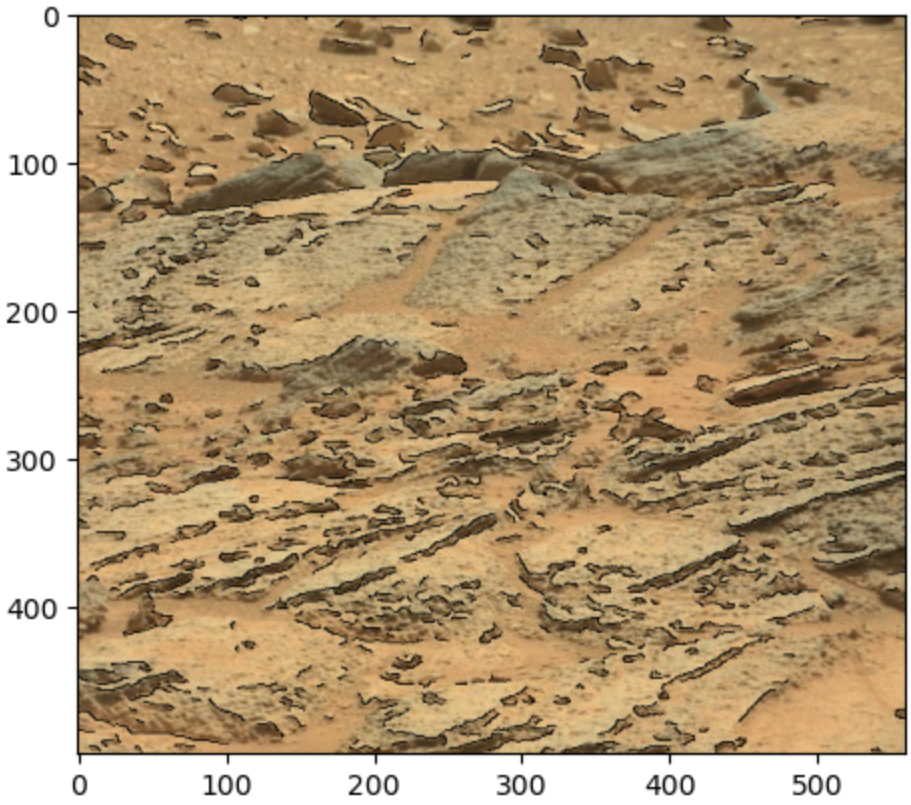

0
100
200
300
400
0
100
200
300
400
500

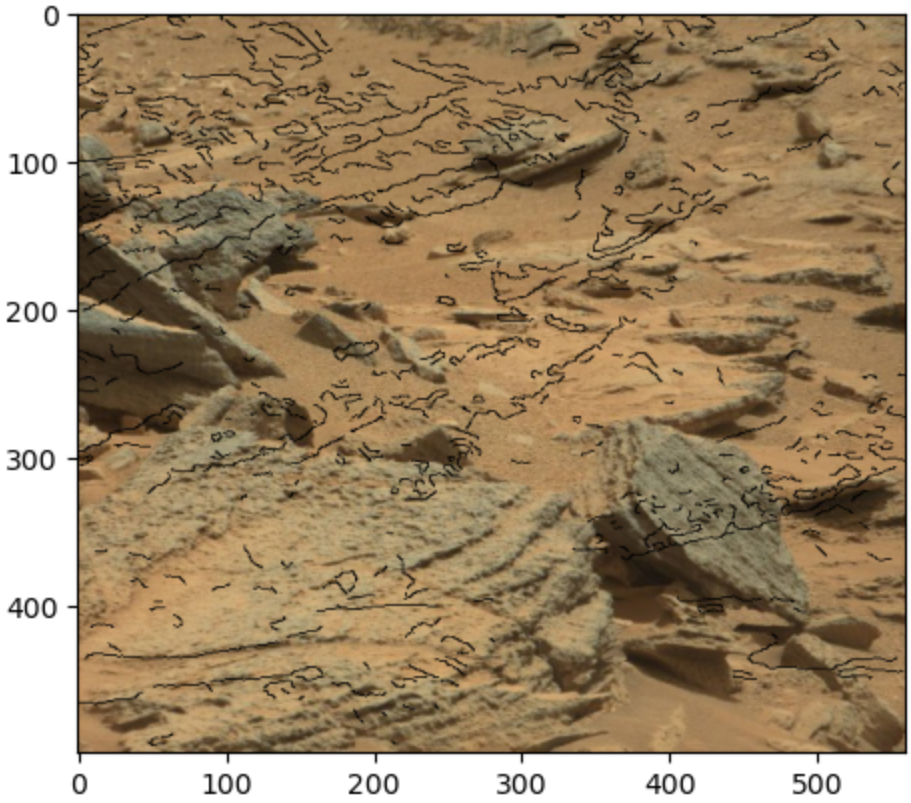

0
100
200
300
400
0
100
200
300
400
500

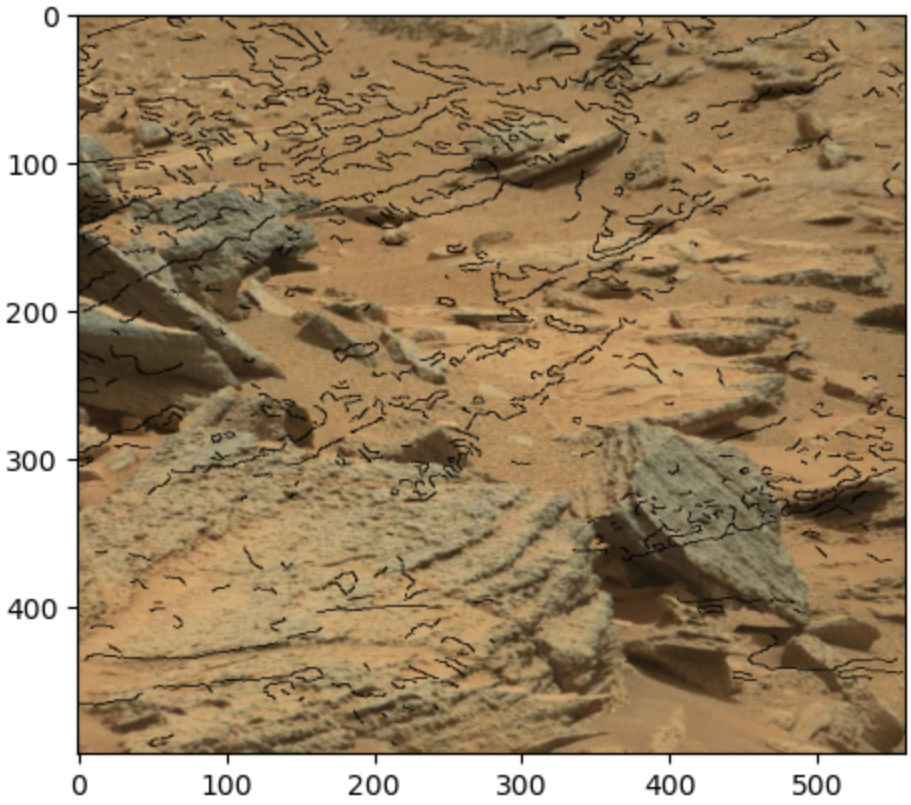

0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: