IPPR Lab 3

Name : Arya Shah

Roll No. E071

Class : BTech CSBS

Aim: To Apply Histogram Equalization And Improve The Contrast Of The Image

Notebook 1 of 3

```
#Importing Libraries
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
```

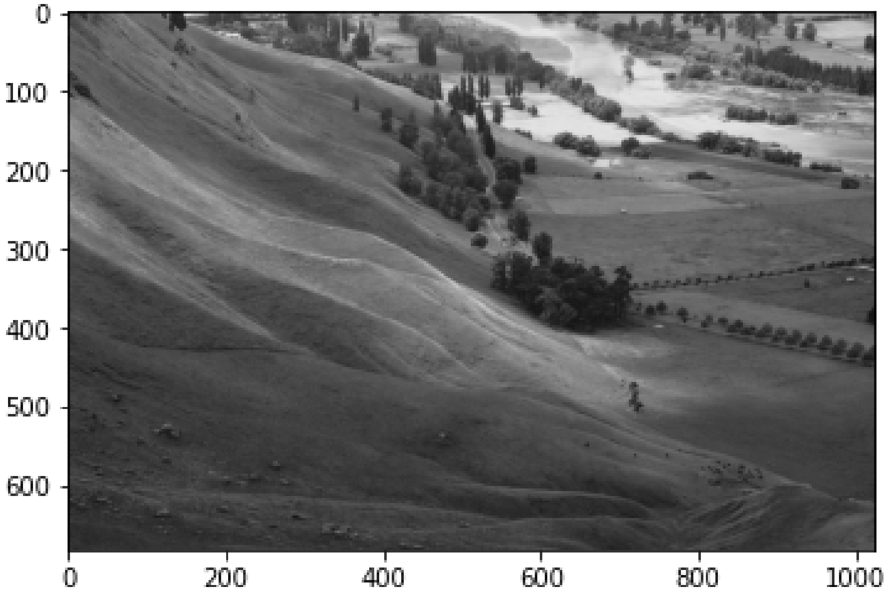
```
#Import Image
image1=io.imread('Unequalized_image.jpg')
```

```
sh=image1.shape
sh
```

```
(683, 1024, 3)
```

```
from skimage.color import rgb2gray

image1=rgb2gray(image1)
plt.imshow(image1,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x2a1ffdd0b48>
```



```
image1=image1*255
```

```
import numpy as np
sh=image1.shape
rows=sh[0]
cols=sh[1]
nr,bins=np.histogram(image1.flatten(),256)
```

```
c=nr.cumsum()
```

```
c
```

```
array([     1,      1,      2,      2,      2,     14,     14,     14,
           33,     33,     82,     82,     82,    175,    175,    175,
          395,    395,    395,    883,    883,   1944,   1944,   1944,
         3691,   3691,   3691,   5851,   5851,   8569,   8569,   8569,
        11828,  11828,  11828,  15867,  15867,  15867,  20975,  20975,
        27370,  27370,  27370,  35833,  35833,  35833,  46443,  46443,
        46443,  59973,  59973,  76088,  76088,  76088,  95213,  95213,
        95213, 117399, 117399, 143095, 143095, 143095, 171796, 171796,
       171796, 200120, 200120, 200120, 225879, 225879, 250248, 250248,
       250248, 273826, 273826, 273826, 295601, 295601, 315986, 315986,
       315986, 335825, 335825, 335825, 355606, 355606, 355606, 375051,
       375051, 393907, 393907, 393907, 412364, 412364, 412364, 430270,
       430270, 430270, 448187, 448187, 465993, 465993, 465993, 484478,
       484478, 484478, 502158, 502158, 517957, 517957, 517957, 532120,
       532120, 532120, 544932, 544932, 544932, 556508, 556508, 567259,
       567259, 567259, 577158, 577158, 577158, 586538, 586538, 586538,
       595401, 595401, 603800, 603800, 603800, 611204, 611204, 611204,
       617607, 617607, 623518, 623518, 623518, 629109, 629109, 629109,
       634259, 634259, 634259, 638869, 638869, 643189, 643189, 643189,
       647049, 647049, 647049, 650665, 650665, 654095, 654095, 654095,
       657158, 657158, 657158, 659807, 659807, 659807, 662020, 662020,
       664066, 664066, 664066, 665925, 665925, 665925, 667617, 667617,
       667617, 669101, 669101, 670463, 670463, 670463, 671753, 671753,
       671753, 672906, 672906, 674138, 674138, 674138, 675438, 675438,
       675438, 676958, 676958, 676958, 678342, 678342, 679869, 679869,
       679869, 681604, 681604, 681604, 683455, 683455, 684987, 684987,
       684987, 686346, 686346, 686346, 687760, 687760, 687760, 689243,
       689243, 690784, 690784, 690784, 692379, 692379, 692379, 694748,
       694748, 694748, 697558, 697558, 698701, 698701, 698701, 699121,
       699121, 699121, 699249, 699249, 699305, 699305, 699305, 699346,
       699346, 699346, 699364, 699364, 699364, 699374, 699374, 699384,
       699384, 699384, 699390, 699390, 699390, 699391, 699391, 699392],
      dtype=int64)
```

```
numpix=rows*cols
s=c*255/numpix
```

```
s
```

```
array([3.64602398e-04, 3.64602398e-04, 7.29204795e-04, 7.29204795e-04,
       7.29204795e-04, 5.10443357e-03, 5.10443357e-03, 5.10443357e-03,
       1.20318791e-02, 1.20318791e-02, 2.98973966e-02, 2.98973966e-02,
       2.98973966e-02, 6.38054196e-02, 6.38054196e-02, 6.38054196e-02,
       1.44017947e-01, 1.44017947e-01, 1.44017947e-01, 3.21943917e-01,
       3.21943917e-01, 7.08787061e-01, 7.08787061e-01, 7.08787061e-01,
       1.34574745e+00, 1.34574745e+00, 1.34574745e+00, 2.13328863e+00,
       2.13328863e+00, 3.12427794e+00, 3.12427794e+00, 3.12427794e+00,
       4.31251716e+00, 4.31251716e+00, 4.31251716e+00, 5.78514624e+00,
       5.78514624e+00, 5.78514624e+00, 7.64753529e+00, 7.64753529e+00,
       9.97916762e+00, 9.97916762e+00, 9.97916762e+00, 1.30647977e+01,
       1.30647977e+01, 1.30647977e+01, 1.69332291e+01, 1.69332291e+01,
       1.69332291e+01, 2.18662996e+01, 2.18662996e+01, 2.77418672e+01,
       2.77418672e+01, 2.77418672e+01, 3.47148881e+01, 3.47148881e+01,
       3.47148881e+01, 4.28039569e+01, 4.28039569e+01, 5.21727801e+01,
       5.21727801e+01, 5.21727801e+01, 6.26372335e+01, 6.26372335e+01,
       6.26372335e+01, 7.29642318e+01, 7.29642318e+01, 7.29642318e+01,
       8.23560249e+01, 8.23560249e+01, 9.12410208e+01, 9.12410208e+01,
       9.12410208e+01, 9.98376161e+01, 9.98376161e+01, 9.98376161e+01,
       1.07776833e+02, 1.07776833e+02, 1.15209253e+02, 1.15209253e+02,
       1.15209253e+02, 1.22442600e+02, 1.22442600e+02, 1.22442600e+02,
       1.29654800e+02, 1.29654800e+02, 1.29654800e+02, 1.36744494e+02,
       1.36744494e+02, 1.43619437e+02, 1.43619437e+02, 1.43619437e+02,
       1.50348903e+02, 1.50348903e+02, 1.50348903e+02, 1.56877474e+02,
       1.56877474e+02, 1.56877474e+02, 1.63410055e+02, 1.63410055e+02,
       1.69902165e+02, 1.69902165e+02, 1.69902165e+02, 1.76641840e+02,
       1.76641840e+02, 1.76641840e+02, 1.83088011e+02, 1.83088011e+02,
       1.88848364e+02, 1.88848364e+02, 1.88848364e+02, 1.94012228e+02,
       1.94012228e+02, 1.94012228e+02, 1.98683514e+02, 1.98683514e+02,
       1.98683514e+02, 2.02904151e+02, 2.02904151e+02, 2.06823991e+02,
       2.06823991e+02, 2.06823991e+02, 2.10433191e+02, 2.10433191e+02,
       2.10433191e+02, 2.13853161e+02, 2.13853161e+02, 2.13853161e+02,
       2.17084632e+02, 2.17084632e+02, 2.20146928e+02, 2.20146928e+02,
       2.20146928e+02, 2.22846444e+02, 2.22846444e+02, 2.22846444e+02,
       2.25180993e+02, 2.25180993e+02, 2.27336158e+02, 2.27336158e+02,
       2.27336158e+02, 2.29374650e+02, 2.29374650e+02, 2.29374650e+02,
       2.31252352e+02, 2.31252352e+02, 2.31252352e+02, 2.32933169e+02,
       2.32933169e+02, 2.34508251e+02, 2.34508251e+02, 2.34508251e+02,
       2.35915617e+02, 2.35915617e+02, 2.35915617e+02, 2.37234019e+02,
       2.37234019e+02, 2.38484605e+02, 2.38484605e+02, 2.38484605e+02,
       2.39601382e+02, 2.39601382e+02, 2.39601382e+02, 2.40567214e+02,
       2.40567214e+02, 2.40567214e+02, 2.41374079e+02, 2.41374079e+02,
       2.42120056e+02, 2.42120056e+02, 2.42120056e+02, 2.42797852e+02,
       2.42797852e+02, 2.42797852e+02, 2.43414759e+02, 2.43414759e+02,
       2.43414759e+02, 2.43955829e+02, 2.43955829e+02, 2.44452417e+02,
       2.44452417e+02, 2.44452417e+02, 2.44922754e+02, 2.44922754e+02,
       2.44922754e+02, 2.45343141e+02, 2.45343141e+02, 2.45792331e+02,
       2.45792331e+02, 2.45792331e+02, 2.46266314e+02, 2.46266314e+02,
       2.46266314e+02, 2.46820510e+02, 2.46820510e+02, 2.46820510e+02,
       2.47325120e+02, 2.47325120e+02, 2.47881867e+02, 2.47881867e+02,
       2.47881867e+02, 2.48514453e+02, 2.48514453e+02, 2.48514453e+02,
       2.49189332e+02, 2.49189332e+02, 2.49747902e+02, 2.49747902e+02,
       2.49747902e+02, 2.50243397e+02, 2.50243397e+02, 2.50243397e+02,
       2.50758945e+02, 2.50758945e+02, 2.50758945e+02, 2.51299650e+02,
       2.51299650e+02, 2.51861503e+02, 2.51861503e+02, 2.51861503e+02,
       2.52443043e+02, 2.52443043e+02, 2.52443043e+02, 2.53306786e+02,
       2.53306786e+02, 2.53306786e+02, 2.54331319e+02, 2.54331319e+02,
       2.54748060e+02, 2.54748060e+02, 2.54748060e+02, 2.54901193e+02,
       2.54901193e+02, 2.54901193e+02, 2.54947862e+02, 2.54947862e+02,
       2.54968280e+02, 2.54968280e+02, 2.54968280e+02, 2.54983228e+02,
       2.54983228e+02, 2.54983228e+02, 2.54989791e+02, 2.54989791e+02,
       2.54989791e+02, 2.54993437e+02, 2.54993437e+02, 2.54997083e+02,
       2.54997083e+02, 2.54997083e+02, 2.54999271e+02, 2.54999271e+02,
       2.54999271e+02, 2.54999635e+02, 2.54999635e+02, 2.55000000e+02])
```

```
s=s.astype(int)
```

```
s
```

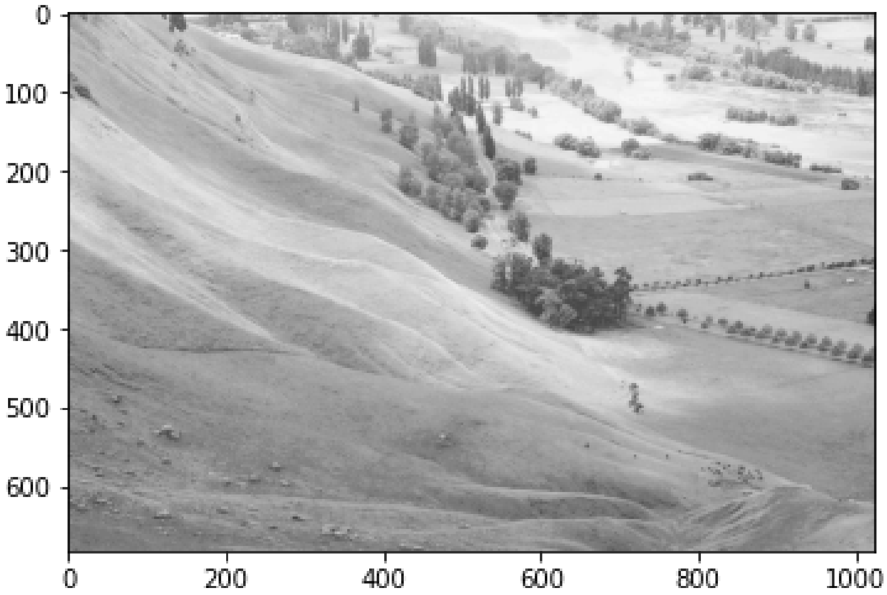
```
array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1,   1,
         1,   2,   2,   3,   3,   3,   4,   4,   4,   5,   5,   5,   7,
         7,   9,   9,   9,  13,  13,  13,  16,  16,  16,  21,  21,  27,
        27,  27,  34,  34,  34,  42,  42,  52,  52,  52,  62,  62,  62,
        72,  72,  72,  82,  82,  91,  91,  91,  99,  99,  99, 107, 107,
       115, 115, 115, 122, 122, 122, 129, 129, 129, 136, 136, 143, 143,
       143, 150, 150, 150, 156, 156, 156, 163, 163, 169, 169, 169, 176,
       176, 176, 183, 183, 188, 188, 188, 194, 194, 194, 198, 198, 198,
       202, 202, 206, 206, 206, 210, 210, 210, 213, 213, 213, 217, 217,
       220, 220, 220, 222, 222, 222, 225, 225, 227, 227, 227, 229, 229,
       229, 231, 231, 231, 232, 232, 234, 234, 234, 235, 235, 235, 237,
       237, 238, 238, 238, 239, 239, 239, 240, 240, 240, 241, 241, 242,
       242, 242, 242, 242, 242, 243, 243, 243, 243, 243, 244, 244, 244,
       244, 244, 244, 245, 245, 245, 245, 245, 246, 246, 246, 246, 246,
       246, 247, 247, 247, 247, 247, 248, 248, 248, 249, 249, 249, 249,
       249, 250, 250, 250, 250, 250, 250, 251, 251, 251, 251, 251, 252,
       252, 252, 253, 253, 253, 254, 254, 254, 254, 254, 254, 254, 254,
       254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254,
       254, 254, 254, 254, 254, 254, 254, 254, 255])
```

```
image_eq1=image1.copy()
```

```
for r in range(rows):
    for c in range(cols):
        temp=image1[r][c]
        image_eq1[r][c]=s[int(temp)]
```

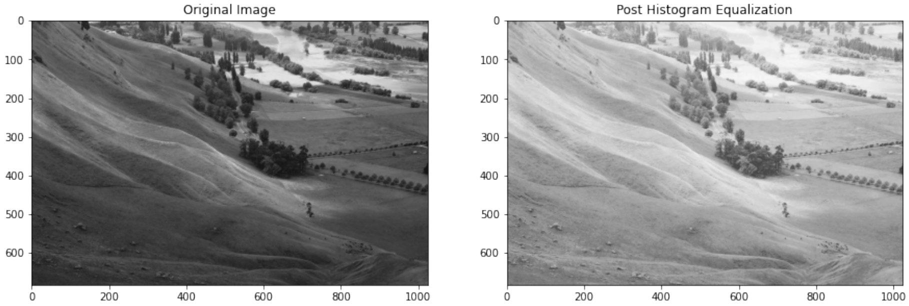
```
plt.imshow(image_eq1, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x2a191130d08>
```



```
#Display original and transformed image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image1,cmap='gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_eq1,cmap='gray')
plt.title('Post Histogram Equalization')
```

```
Text(0.5, 1.0, 'Post Histogram Equalization')
```


Original Image

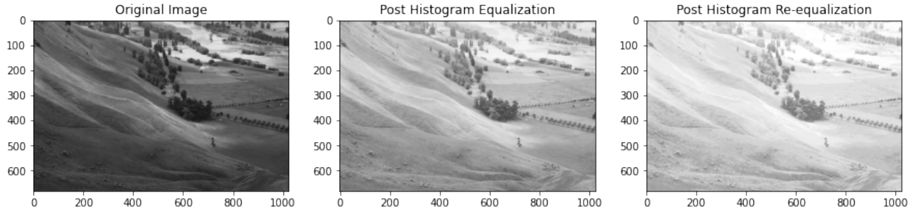
Post Histogram Equalization

```
image_eq2=image_eq1.copy()
```

```
sh=image_eq1.shape
rows=sh[0]
```

```
cols=sh[1]

nr,bins=np.histogram(image_eq1.flatten(),256)

c=nr.cumsum()

numpix=rows*cols
s=c*255/numpix

s=s.astype(int)

for r in range(rows):
    for c in range(cols):
        temp=image_eq1[r][c]
        image_eq2[r][c]=s[int(temp)]

#Display original and transformed image
plt.figure(figsize=(15,15))
plt.subplot(1,3,1)
plt.imshow(image1,cmap='gray')
plt.title('Original Image')

plt.subplot(1,3,2)
plt.imshow(image_eq1,cmap='gray')
plt.title('Post Histogram Equalization')

plt.subplot(1,3,3)
plt.imshow(image_eq2,cmap='gray')
plt.title('Post Histogram Re-equalization')
```

```
Text(0.5, 1.0, 'Post Histogram Re-equalization')
```



End Of Notebook: Part 1

Notebook 2 of 3

```
#Importing Libraries
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
```

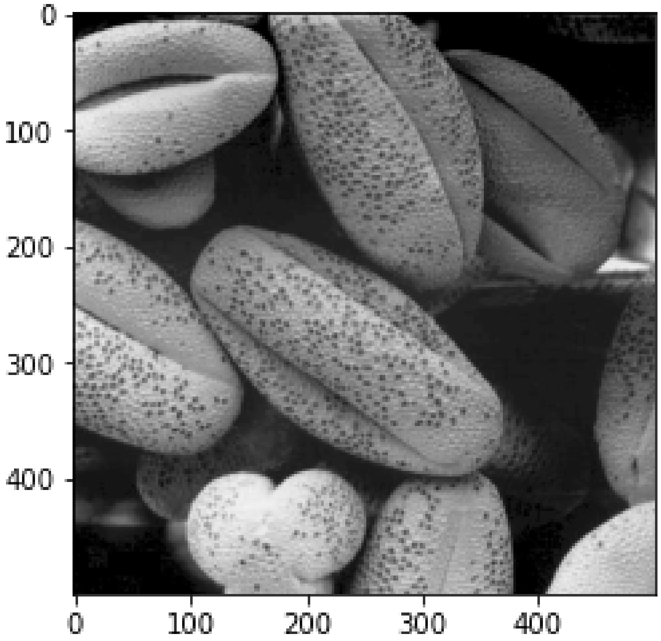
```
#Import Image
image1=io.imread('pollen_dark.tif')
```

```
sh=image1.shape
sh
```

```
(500, 500)
```

```
#image1=rgb2gray(image1)
plt.imshow(image1,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1e7fc211370>
```



```
import numpy as np
sh=image1.shape
rows=sh[0]
cols=sh[1]
nr,bins=np.histogram(image1.flatten(),256)
```

```
c=nr.cumsum()
```

```
c
```

```
array([ 17496,  17496,  17496,  17496,  17496,  17496,  17496,  17496,
        17496,  17496,  17765,  17765,  17765,  17765,  17765,  17765,
        19192,  19192,  19192,  19192,  19192,  30392,  30392,  30392,
        30392,  30392,  30392,  49470,  49470,  49470,  49470,  49470,
        59591,  59591,  59591,  59591,  59591,  59591,  67085,  67085,
        67085,  67085,  67085,  72689,  72689,  72689,  72689,  72689,
        72689,  72689,  72689,  72689,  72689,  72689,  78212,  78212,
        78212,  78212,  78212,  84028,  84028,  84028,  84028,  84028,
        84028,  89159,  89159,  89159,  89159,  89159,  94870,  94870,
        94870,  94870,  94870,  94870, 101176, 101176, 101176, 101176,
       101176, 109064, 109064, 109064, 109064, 109064, 109064, 116831,
       116831, 116831, 116831, 116831, 116831, 116831, 116831, 116831,
       116831, 116831, 123884, 123884, 123884, 123884, 123884, 131818,
       131818, 131818, 131818, 131818, 139731, 139731, 139731, 139731,
       139731, 139731, 149266, 149266, 149266, 149266, 149266, 157537,
       157537, 157537, 157537, 157537, 157537, 166061, 166061, 166061,
       166061, 166061, 174685, 174685, 174685, 174685, 174685, 174685,
       174685, 174685, 174685, 174685, 174685, 184211, 184211, 184211,
       184211, 184211, 184211, 191804, 191804, 191804, 191804, 191804,
       198806, 198806, 198806, 198806, 198806, 205244, 205244, 205244,
       205244, 205244, 205244, 211176, 211176, 211176, 211176, 211176,
       217334, 217334, 217334, 217334, 217334, 217334, 222093, 222093,
       222093, 222093, 222093, 222093, 222093, 222093, 222093, 222093,
       222093, 226216, 226216, 226216, 226216, 226216, 229962, 229962,
       229962, 229962, 229962, 229962, 233858, 233858, 233858, 233858,
       233858, 236897, 236897, 236897, 236897, 236897, 239602, 239602,
       239602, 239602, 239602, 239602, 242383, 242383, 242383, 242383,
       242383, 244938, 244938, 244938, 244938, 244938, 244938, 244938,
       244938, 244938, 244938, 244938, 247593, 247593, 247593, 247593,
       247593, 247593, 249231, 249231, 249231, 249231, 249231, 249928,
       249928, 249928, 249928, 249928, 249928, 249997, 249997, 249997,
       249997, 249997, 249999, 249999, 249999, 249999, 249999, 250000],
      dtype=int64)
```

```
numpix=rows*cols
s=c*255/numpix
```

```
s
```

```
array([ 17.84592,  17.84592,  17.84592,  17.84592,  17.84592,  17.84592,
        17.84592,  17.84592,  17.84592,  17.84592,  18.1203 ,  18.1203 ,
        18.1203 ,  18.1203 ,  18.1203 ,  18.1203 ,  19.57584,  19.57584,
        19.57584,  19.57584,  19.57584,  30.99984,  30.99984,  30.99984,
        30.99984,  30.99984,  30.99984,  50.4594 ,  50.4594 ,  50.4594 ,
        50.4594 ,  50.4594 ,  60.78282,  60.78282,  60.78282,  60.78282,
        60.78282,  60.78282,  68.4267 ,  68.4267 ,  68.4267 ,  68.4267 ,
        68.4267 ,  74.14278,  74.14278,  74.14278,  74.14278,  74.14278,
        74.14278,  74.14278,  74.14278,  74.14278,  74.14278,  74.14278,
        79.77624,  79.77624,  79.77624,  79.77624,  79.77624,  85.70856,
        85.70856,  85.70856,  85.70856,  85.70856,  85.70856,  90.94218,
        90.94218,  90.94218,  90.94218,  90.94218,  96.7674 ,  96.7674 ,
        96.7674 ,  96.7674 ,  96.7674 ,  96.7674 , 103.19952, 103.19952,
       103.19952, 103.19952, 103.19952, 111.24528, 111.24528, 111.24528,
       111.24528, 111.24528, 111.24528, 119.16762, 119.16762, 119.16762,
       119.16762, 119.16762, 119.16762, 119.16762, 119.16762, 119.16762,
       119.16762, 119.16762, 126.36168, 126.36168, 126.36168, 126.36168,
       126.36168, 134.45436, 134.45436, 134.45436, 134.45436, 134.45436,
       142.52562, 142.52562, 142.52562, 142.52562, 142.52562, 142.52562,
       152.25132, 152.25132, 152.25132, 152.25132, 152.25132, 160.68774,
       160.68774, 160.68774, 160.68774, 160.68774, 160.68774, 169.38222,
       169.38222, 169.38222, 169.38222, 169.38222, 178.1787 , 178.1787 ,
       178.1787 , 178.1787 , 178.1787 , 178.1787 , 178.1787 , 178.1787 ,
       178.1787 , 178.1787 , 178.1787 , 187.89522, 187.89522, 187.89522,
       187.89522, 187.89522, 187.89522, 195.64008, 195.64008, 195.64008,
       195.64008, 195.64008, 202.78212, 202.78212, 202.78212, 202.78212,
       202.78212, 209.34888, 209.34888, 209.34888, 209.34888, 209.34888,
       209.34888, 215.39952, 215.39952, 215.39952, 215.39952, 215.39952,
       221.68068, 221.68068, 221.68068, 221.68068, 221.68068, 221.68068,
       226.53486, 226.53486, 226.53486, 226.53486, 226.53486, 226.53486,
       226.53486, 226.53486, 226.53486, 226.53486, 226.53486, 230.74032,
       230.74032, 230.74032, 230.74032, 230.74032, 234.56124, 234.56124,
       234.56124, 234.56124, 234.56124, 234.56124, 238.53516, 238.53516,
       238.53516, 238.53516, 238.53516, 241.63494, 241.63494, 241.63494,
       241.63494, 241.63494, 244.39404, 244.39404, 244.39404, 244.39404,
       244.39404, 244.39404, 247.23066, 247.23066, 247.23066, 247.23066,
       247.23066, 249.83676, 249.83676, 249.83676, 249.83676, 249.83676,
       249.83676, 249.83676, 249.83676, 249.83676, 249.83676, 249.83676,
       252.54486, 252.54486, 252.54486, 252.54486, 252.54486, 252.54486,
       254.21562, 254.21562, 254.21562, 254.21562, 254.21562, 254.92656,
       254.92656, 254.92656, 254.92656, 254.92656, 254.92656, 254.99694,
       254.99694, 254.99694, 254.99694, 254.99694, 254.99898, 254.99898,
       254.99898, 254.99898, 254.99898, 255.     ])
```

```
s=s.astype(int)
```

```
s
```

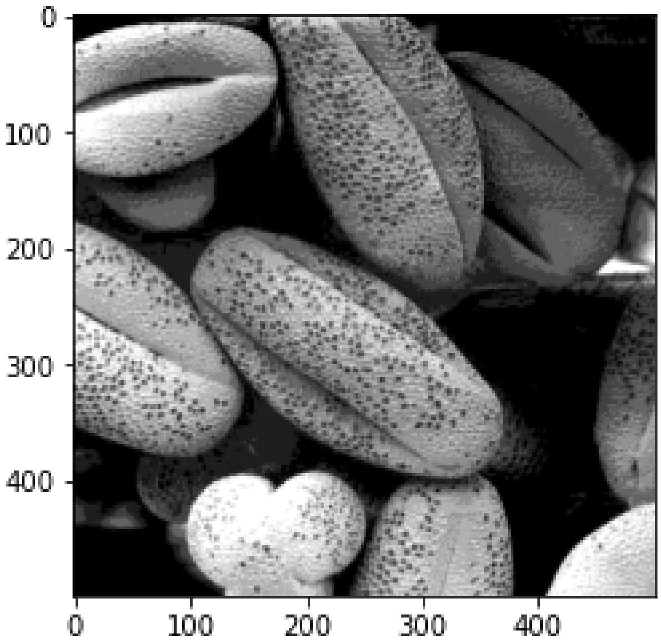
```
array([ 17,  17,  17,  17,  17,  17,  17,  17,  17,  17,  18,  18,  18,
        18,  18,  18,  19,  19,  19,  19,  19,  30,  30,  30,  30,  30,
        30,  50,  50,  50,  50,  50,  60,  60,  60,  60,  60,  60,  68,
        68,  68,  68,  68,  74,  74,  74,  74,  74,  74,  74,  74,  74,
        74,  74,  79,  79,  79,  79,  79,  85,  85,  85,  85,  85,  85,
        90,  90,  90,  90,  90,  96,  96,  96,  96,  96,  96, 103, 103,
       103, 103, 103, 111, 111, 111, 111, 111, 111, 119, 119, 119, 119,
       119, 119, 119, 119, 119, 119, 119, 126, 126, 126, 126, 126, 134,
       134, 134, 134, 134, 142, 142, 142, 142, 142, 142, 152, 152, 152,
       152, 152, 160, 160, 160, 160, 160, 160, 169, 169, 169, 169, 169,
       178, 178, 178, 178, 178, 178, 178, 178, 178, 178, 178, 187, 187,
       187, 187, 187, 187, 195, 195, 195, 195, 195, 202, 202, 202, 202,
       202, 209, 209, 209, 209, 209, 209, 215, 215, 215, 215, 215, 221,
       221, 221, 221, 221, 221, 226, 226, 226, 226, 226, 226, 226, 226,
       226, 226, 226, 230, 230, 230, 230, 230, 234, 234, 234, 234, 234,
       234, 238, 238, 238, 238, 238, 241, 241, 241, 241, 241, 244, 244,
       244, 244, 244, 244, 247, 247, 247, 247, 247, 249, 249, 249, 249,
       249, 249, 249, 249, 249, 249, 249, 252, 252, 252, 252, 252, 252,
       254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254, 254,
       254, 254, 254, 254, 254, 254, 254, 254, 255])
```

```python
image_eq1=image1.copy()
```

```python
for r in range(rows):
    for c in range(cols):
        temp=image1[r][c]
        image_eq1[r][c]=s[int(temp)]
```

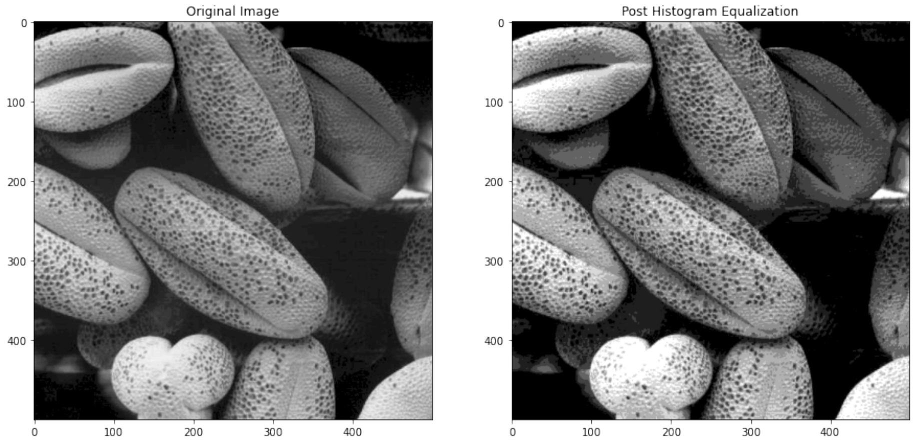
```python
plt.imshow(image_eq1, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1e7fc269070>
```



```
#Display original and transformed image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image1,cmap='gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_eq1,cmap='gray')
plt.title('Post Histogram Equalization')
```

```
Text(0.5, 1.0, 'Post Histogram Equalization')
```



```
image_eq2=image_eq1.copy()
```

```
sh=image_eq1.shape
rows=sh[0]
cols=sh[1]
```

```
nr,bins=np.histogram(image_eq1.flatten(),256)
```

```
c=nr.cumsum()
```

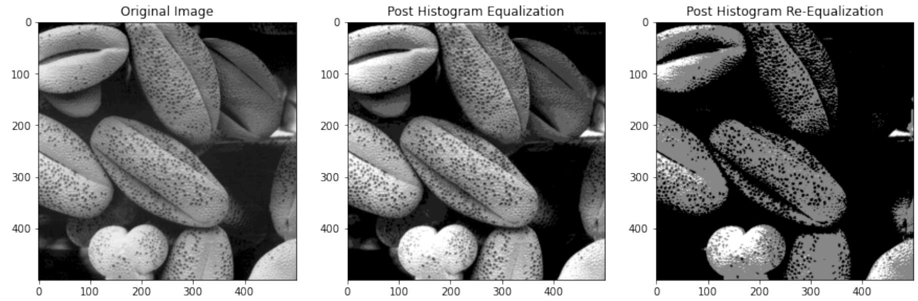
```
numpix=rows*cols
s=c*255/numpix
```

```
s=s.astype(int)
```

```
for r in range(rows):
    for c in range(cols):
        temp=image_eq1[r][c]
        image_eq2[r][c]=s[int(temp)]
```

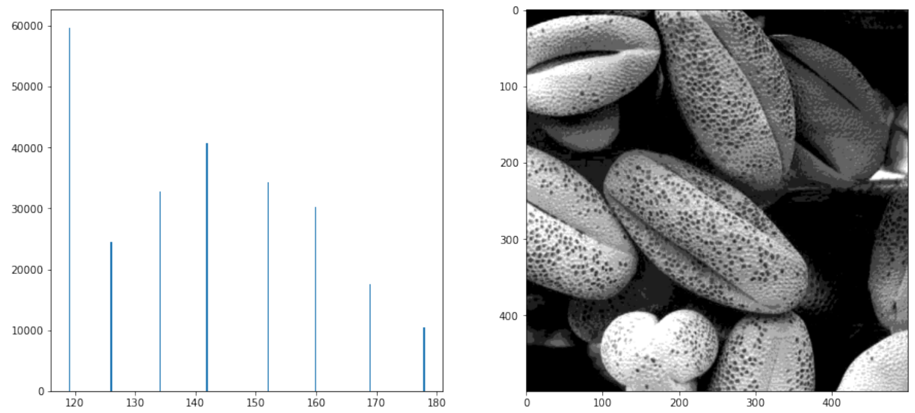
```
#Display original and transformed image
plt.figure(figsize=(15,15))
plt.subplot(1,3,1)
plt.imshow(image1,cmap='gray')
plt.title('Original Image')

plt.subplot(1,3,2)
plt.imshow(image_eq1,cmap='gray')
plt.title('Post Histogram Equalization')

plt.subplot(1,3,3)
plt.imshow(image_eq2,cmap='gray')
plt.title('Post Histogram Re-Equalization')
```

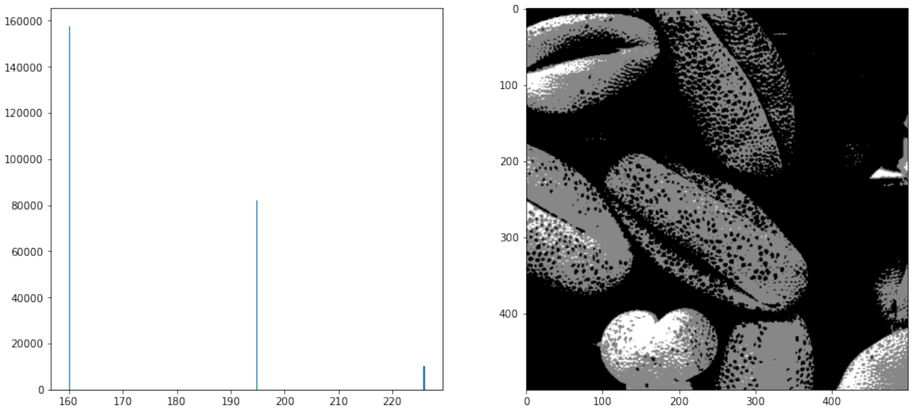
```
Text(0.5, 1.0, 'Post Histogram Re-Equalization')
```



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

image_eq1_h=image_eq1.flatten()
image_eq2_h=image_eq2.flatten()

plt.subplot(2,2,1)
ax=plt.hist(image_eq1_h,bins=256)

plt.subplot(2,2,2)
plt.imshow(image_eq1, cmap='gray')

plt.subplot(2,2,3)
ax=plt.hist(image_eq2_h,bins=256)

plt.subplot(2,2,4)
plt.imshow(image_eq2, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1e7fda07070>
```





End Of Notebook: Part 2

Notebook 3 of 3

```
#Importing Libraries
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
```

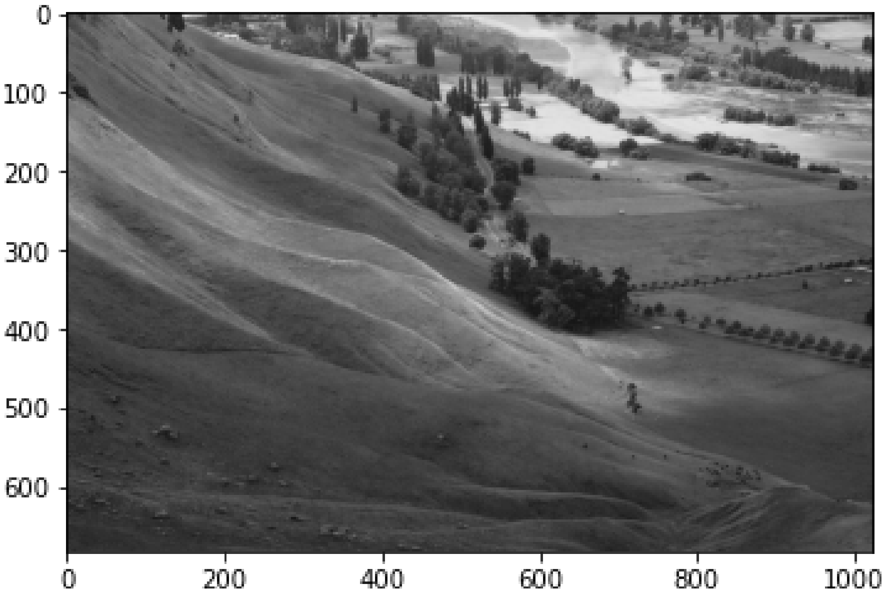
```
#Import Image
image1=io.imread('Unequalized_image.jpg')
```

```
sh=image1.shape
sh
```

```
(683, 1024, 3)
```

```
from skimage.color import rgb2gray

image1=rgb2gray(image1)
plt.imshow(image1,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x245563049d0>
```



```
image1=image1*255
```

```
import numpy as np
sh=image1.shape
rows=sh[0]
cols=sh[1]
```

```
image_a=image1[0:int(rows/2)][:]
image_b=image1[int(rows/2):][:]
```

```
image1.shape
```

```
(683, 1024)
```

```
sh1=image_a.shape
rowsa=sh1[0]
colsa=sh1[1]
```

```
sh2=image_b.shape
rowsb=sh2[0]
colsb=sh2[1]
```

```
nr,bins=np.histogram(image_a.flatten(),256)
```

```
c=nr.cumsum()
```

```
numpix=rowsa*colsa
s=c*255/numpix
```

```
s=s.astype(int)
```

```
image_eq1=image_a.copy()
```

```
for r in range(rowsa):
    for c in range(colsa):
        temp=image_a[r][c]
        image_eq1[r][c]=s[int(temp)]
```

```
nr1,bins=np.histogram(image_b.flatten(),256)
```

```
c1=nr1.cumsum()
```

```
numpix=rowsb*colsb
s1=c1*255/numpix
```

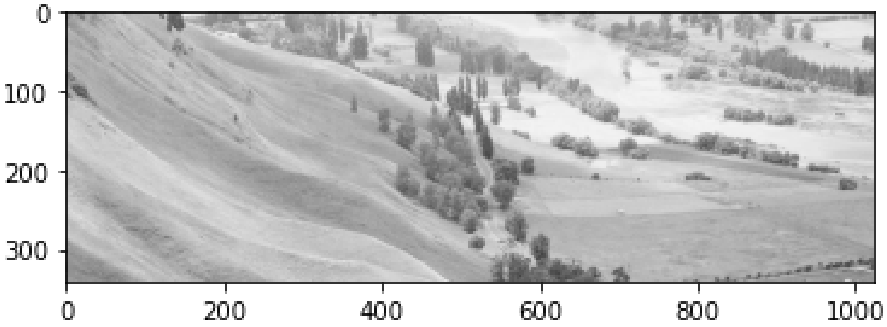
```
s1=s1.astype(int)
```

```
image_eq2=image_b.copy()
```

```
for r in range(rowsa):
    for c in range(colsa):
        temp=image_b[r][c]
        image_eq2[r][c]=s1[int(temp)]
```

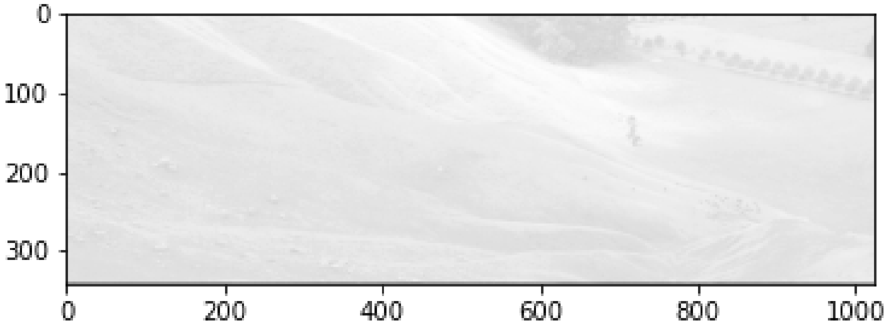
```
plt.imshow(image_eq1,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x2455712e9a0>
```



```
plt.imshow(image_eq2,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x2455770c8b0>
```



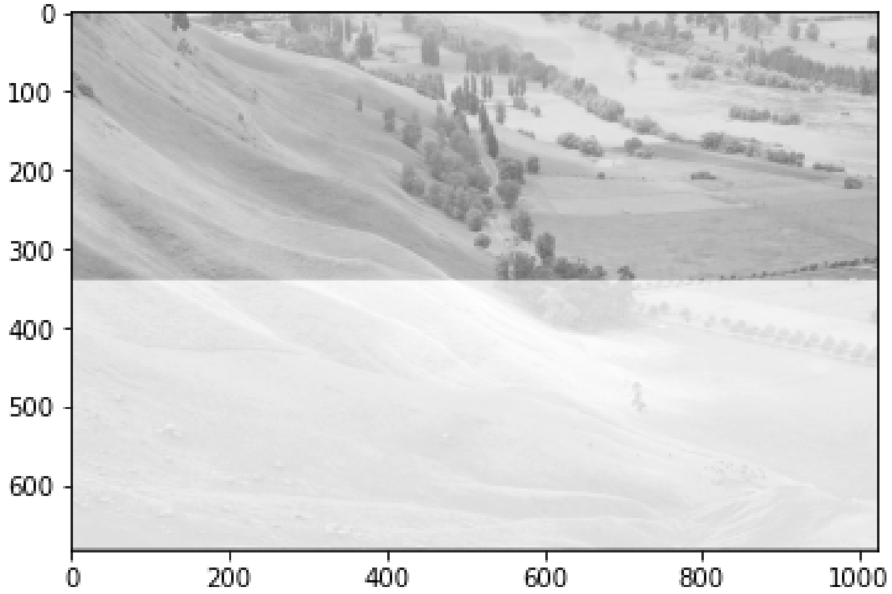
```
image_new=image1.copy()
image_new[0:int(rows/2)][:]=image_eq1
image_new[int(rows/2):][:]=image_eq2
```

```
image_new
```

```
array([[184., 211., 216., ..., 229., 228., 228.],
       [227., 218., 208., ..., 229., 229., 229.],
       [231., 211., 204., ..., 221., 218., 216.],
       ...,
       [244., 237., 238., ..., 244., 244., 243.],
       [244., 237., 237., ..., 244., 244., 243.],
       [145., 128., 129., ..., 146., 145., 144.]])
```

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

```
<matplotlib.image.AxesImage at 0x24557f81f10>
```



Conclusion:

1. Histogram Equalization is applied on the given low contrast image. It is shown that equalized image has better contrast than original image of the Desert.
2. If same image is Re-Equalized, then there is no appreciable improvement in the contrast of image.
3. The same method is applied on other images which are bright/dark/very dark and the equalized image shows improvement in the contrast.
4. If given image is split into two parts and local histogram equalization is applied on these parts, the equalized image shows separation between the two parts though individual parts are equalized.
5. The separation is only because of the objects in the image. If there is no intensity variation along the line of separation then local histogram equalization shows equalized sub parts with no separation between them.

   It can be concluded that local histogram equalization can be applied on a small subpart of the image.

End Of Notebook: Part 3