college-assignment (/github/sachi1406/college-assignment/tree/main)
/
IP_Expt_4_Sachi_Shah.ipynb (/github/sachi1406/college-assignment/tree/main/IP_Expt_4_Sachi_Shah.ipynb)

IP-Experiment No. 4 : Histogram Processing

- Name: Sachi Shah
- Roll No.: C094
- Batch: EB1
- Sap Id: 70321018081

Aim:

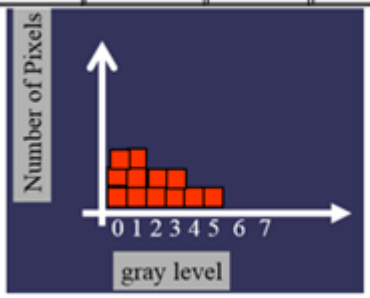
a. Classify the test images as low contrast, high contrast, dark and bright images by plotting their histograms.
b. Implement histogram equalization on the low contrast, dark and bright images.
c. Examine the effect of equalization on the test images by comparing the histograms of the test images with the equalized images.

Histogram:
A histogram of an image is a graph with x axis as pixel intensity (in the range of 0 to L-1) and y axis as the frequency of the pixel intensity.
If n_k = count of number of pixels in the image with intensity value of r_k If the image has M rows and N columnns, then the total number of pixels in the image is MN. The normalized histogram is obtained as follows: p(r_k)= n_k/MN

Example :

$$A=\begin{bmatrix}1 & 3 & 0\\ 0 & 1 & 2\\ 1 & 2 & 4\\ 3 & 5 & 0\end{bmatrix}$$

Pixel intensity	0	1	2	3	4	5
No. of Pixels	3	3	2	2	1	1



In [4]:

```
# import libraries
from skimage import io
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

Plot the histogram of an image by counting the number of times each pixel occurs in the image

In [ ]:

```
# read the cameraman.tif image as img
img = io.imread('cameraman.tif')

count , value = np.histogram(img,255,[0,255])  #(image, largest value, range)
print("Pixel value = ", value)
print("Count = ", count)
print("Cumulative sum of count = ", count.cumsum())
```

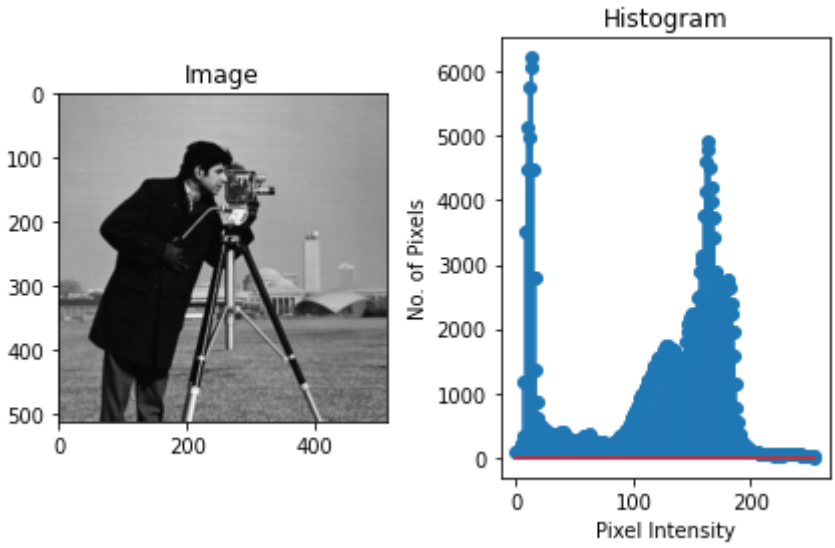
```
Pixel value =  [  0.   1.   2.   3.   4.   5.   6.   7.   8.   9.  10.
  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.
  28.  29.  30.  31.  32.  33.  34.  35.  36.  37.  38.  39.  40.  41.
  42.  43.  44.  45.  46.  47.  48.  49.  50.  51.  52.  53.  54.  55.
  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.
  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82.  83.
  84.  85.  86.  87.  88.  89.  90.  91.  92.  93.  94.  95.  96.  97.
  98.  99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111.
 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125.
 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139.
 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153.
 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167.
 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181.
 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195.
 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209.
 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223.
 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237.
 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251.
 252. 253. 254. 255.]
Count =  [ 102   76   89  114  159  209  335 1173 3523 5129 4490 4980 5
 6067 4480 2805 1375  860  625  498  503  426  430  379  398  418  402
  358  362  354  349  355  381  371  379  348  375  390  398  394  349
  327  345  311  263  256  279  274  258  258  245  242  261  265  268
  285  276  334  308  337  309  313  370  276  301  245  230  244  242
  222  217  230  238  235  224  205  220  213  208  164  203  222  216
  230  243  251  281  315  328  334  327  333  381  417  420  473  525
  573  620  650  687  727  760  803  816  924  986  981 1002 1085 1070
 1174 1202 1219 1190 1314 1321 1432 1399 1453 1539 1477 1509 1598 1592
 1691 1659 1745 1705 1734 1702 1663 1598 1655 1673 1626 1556 1435 1435
 1398 1449 1391 1514 1590 1800 1964 2096 2176 2147 2225 2163 2185 2117
 2258 2494 2524 2885 3045 3134 3767 4147 4596 4906 4790 4497 4195 3993
 3737 3415 2906 2785 2746 2615 2635 2596 2555 2702 2737 2662 2686 2778
 2646 2392 2236 1961 1595 1162  776  565  381  275  235  209  200  176
  146  160  126  109  113   86  103   84   90   81   75   72   56   60
   54   53   51   60   62   70   53   58   51   38   57   58   55   58
   40   48   46   37   45   42   51   52   51   52   60   55   63   53
   69   49   48   44   54   28   28   24   14   17   27   29   22   26
   16   10   36]
Cumulative sum of count =  [   102    178    267    381    540    749
  15399  20379  26141  32353  38420  42900  45705  47080  47940  48565
  49063  49566  49992  50422  50801  51199  51617  52019  52377  52739
  53093  53442  53797  54178  54549  54928  55276  55651  56041  56439
  56833  57182  57509  57854  58165  58428  58684  58963  59237  59495
  59753  59998  60240  60501  60766  61034  61319  61595  61929  62237
  62574  62883  63196  63566  63842  64143  64388  64618  64862  65104
  65326  65543  65773  66011  66246  66470  66675  66895  67108  67316
  67480  67683  67905  68121  68351  68594  68845  69126  69441  69769
  70103  70430  70763  71144  71561  71981  72454  72979  73552  74172
  74822  75509  76236  76996  77799  78615  79539  80525  81506  82508
  83593  84663  85837  87039  88258  89448  90762  92083  93515  94914
  96367  97906  99383 100892 102490 104082 105773 107432 109177 110882
```

```
112616 114318 115981 117579 119234 120907 122533 124089 125524 126959
128357 129806 131197 132711 134301 136101 138065 140161 142337 144484
146709 148872 151057 153174 155432 157926 160450 163335 166380 169514
173281 177428 182024 186930 191720 196217 200412 204405 208142 211557
214463 217248 219994 222609 225244 227840 230395 233097 235834 238496
241182 243960 246606 248998 251234 253195 254790 255952 256728 257293
257674 257949 258184 258393 258593 258769 258915 259075 259201 259310
259423 259509 259612 259696 259786 259867 259942 260014 260070 260130
260184 260237 260288 260348 260410 260480 260533 260591 260642 260680
260737 260795 260850 260908 260948 260996 261042 261079 261124 261166
261217 261269 261320 261372 261432 261487 261550 261603 261672 261721
261769 261813 261867 261895 261923 261947 261961 261978 262005 262034
262056 262082 262098 262108 262144]
```

In [ ]:

```
# read all the four images (E2(einstein high contrast).tif,E2(einstein low contrast).t

plt.subplot(121)
plt.imshow(img,cmap = 'gray')
plt.title('Image')

plt.subplot(122)
plt.stem(count)           #To plot standing graph or histogram
plt.title('Histogram')
plt.xlabel('Pixel Intensity')   #Labels X-axis
plt.ylabel('No. of Pixels')     #Labels Y-axis

plt.tight_layout()        #To space out the 2 images
```

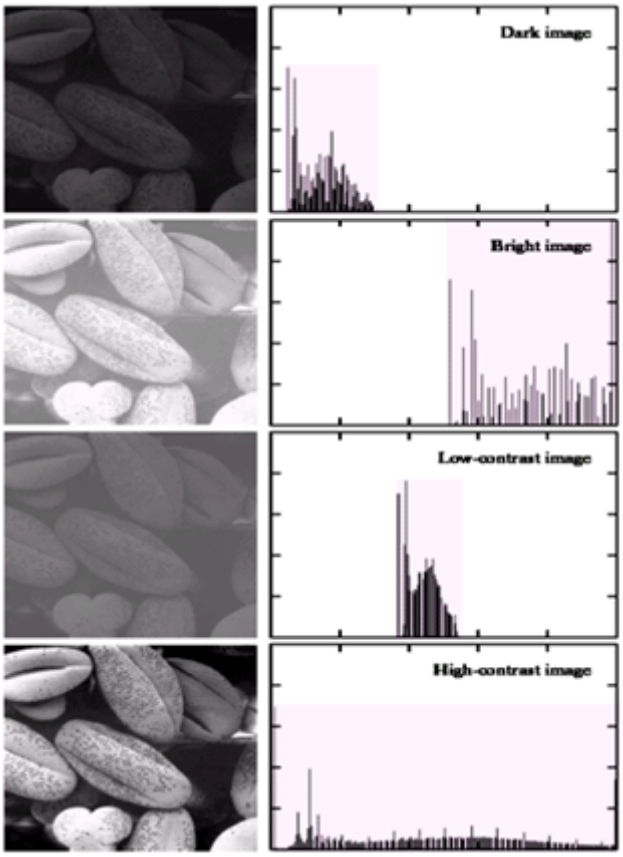
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWar
```



Note: (self)

- To plot Image: imshow
- To plot continuous graph: plt.plot
- To plot discrete graph: plt.stem

Histogram of low contrast, high contrast, dark and bright image

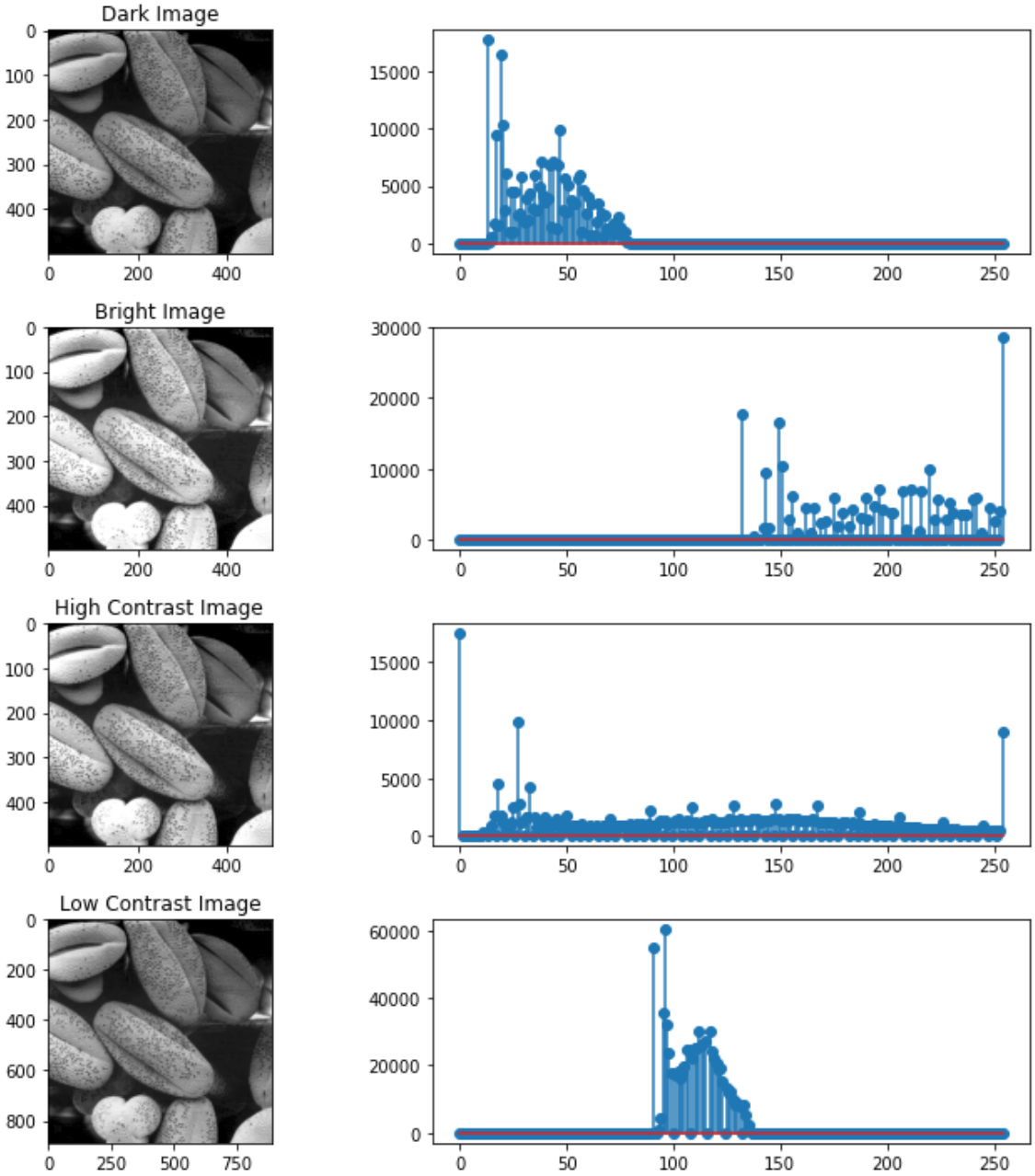


To plot histogram for high contrast and low contrast image using an inbuilt function

In [ ]:

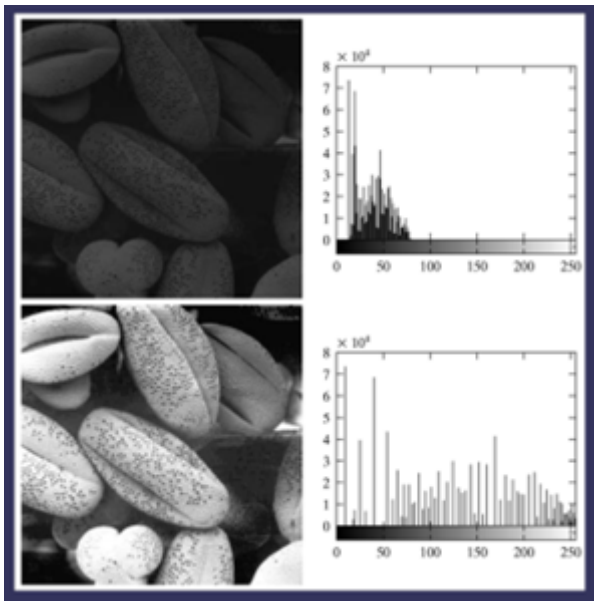
```
# plot the four images and their historgram
#Compute the histogram using built in function --> np.histogram(imga.ravel(),256,[0,25


img_bright = io.imread('Bright image.tif')
img_dark = io.imread('Dark_pollen_image.tif')
img_high_contrast = io.imread('high-contrast-image.tif')
img_low_contrast = io.imread('low-contrast-image.tif')

c_bright , v_bright = np.histogram(img_bright,255,[0,255])
c_dark , v_dark = np.histogram(img_dark,255,[0,255])
c_hc , v_hc = np.histogram(img_high_contrast,255,[0,255])
c_lc , v_lc = np.histogram(img_low_contrast,255,[0,255])

plt.figure(figsize=(10,10))

plt.subplot(4,2,1)
plt.title('Dark Image')
plt.imshow(img_dark, cmap='gray')

plt.subplot(422)
plt.stem(c_dark)

plt.subplot(423)
plt.title('Bright Image')
plt.imshow(img_bright, cmap='gray')

plt.subplot(424)
plt.stem(c_bright)

plt.subplot(425)
plt.title('High Contrast Image')
plt.imshow(img_high_contrast, cmap='gray')

plt.subplot(426)
plt.stem(c_hc)

plt.subplot(427)
plt.title('Low Contrast Image')
plt.imshow(img_low_contrast, cmap='gray')

plt.subplot(428)
plt.stem(c_lc)

plt.tight_layout()        #To space out the 2 images
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: UserWarni
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:30: UserWarni
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:37: UserWarni
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:44: UserWarni
```



To plot histogram for dark and bright image using an inbuilt function

Perform Histogram Equalization using hard coding

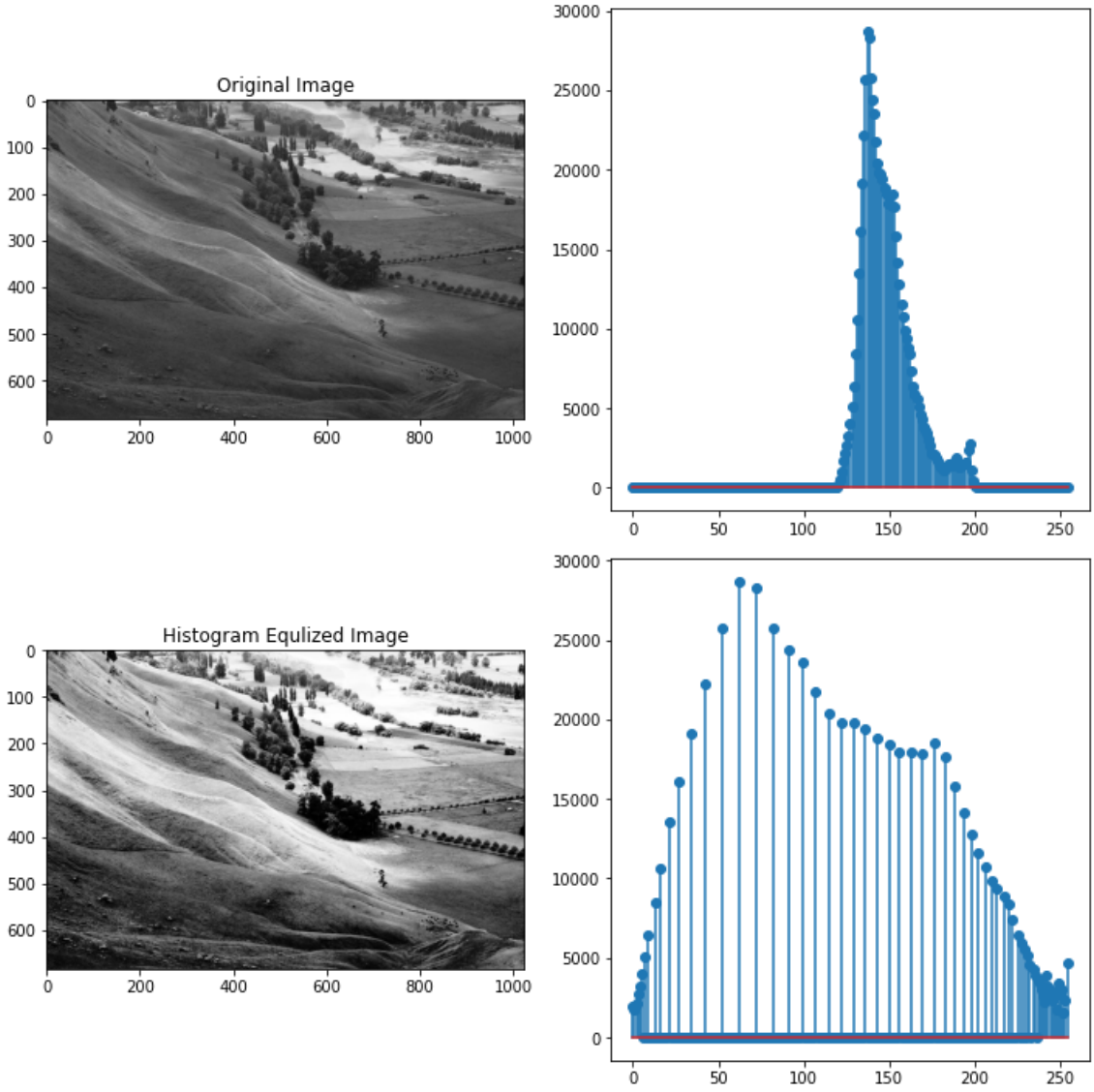


Gray Level (K = 8 total levels)	Count	Probability Density Function (Count/Total Count)	Cumulative Distribution Function CDF	(K - 1) * CDF	FLOOR((K - 1) * CDF)
0	1	0.0156	0.0156	0.109	0
1	3	0.0469	0.0625	0.438	0
2	3	0.0469	0.1094	0.766	0
3	4	0.0625	0.1719	1.203	1
4	9	0.1406	0.3125	2.188	2
5	15	0.2344	0.5469	3.828	3
6	16	0.2500	0.7969	5.578	5
7	13	0.2031	1.0000	7.000	7
	64	**1.000**			

In [18]:

```
# input image Unequalized_image.jpg
# find its histogram count using
img1 = cv2.imread('Unequalized_image.jpg',0)
img1_original = img1.copy()

count , value = np.histogram(img1_original,255,[0,255])  #(image, largest value, range

#find cdf using ---> imhist.cumsum()
cdf = count.cumsum()/count.sum()

# perform cdf_nor=((cdf / numb_pix)* 255)
HE = (cdf * 255).astype(int)           #Histogram equalized values

# replace each pixel value with its corresponding cdf_nor value
r,c = img1.shape
for i in range(r):
    for j in range(c):
        index = img1[i,j]
        img1[i,j] = HE[index]

count2 , value2 = np.histogram(img1,255,[0,255])

# plot the image
plt.figure(figsize=(10,10))

plt.subplot(221)
plt.title('Original Image')
plt.imshow(img1_original, cmap='gray')

plt.subplot(222)
plt.stem(count)

plt.subplot(223)
plt.title('Histogram Equlized Image')
plt.imshow(img1, cmap='gray')

plt.subplot(224)
plt.stem(count2)

plt.tight_layout()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:31: UserWarni
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:38: UserWarni
```

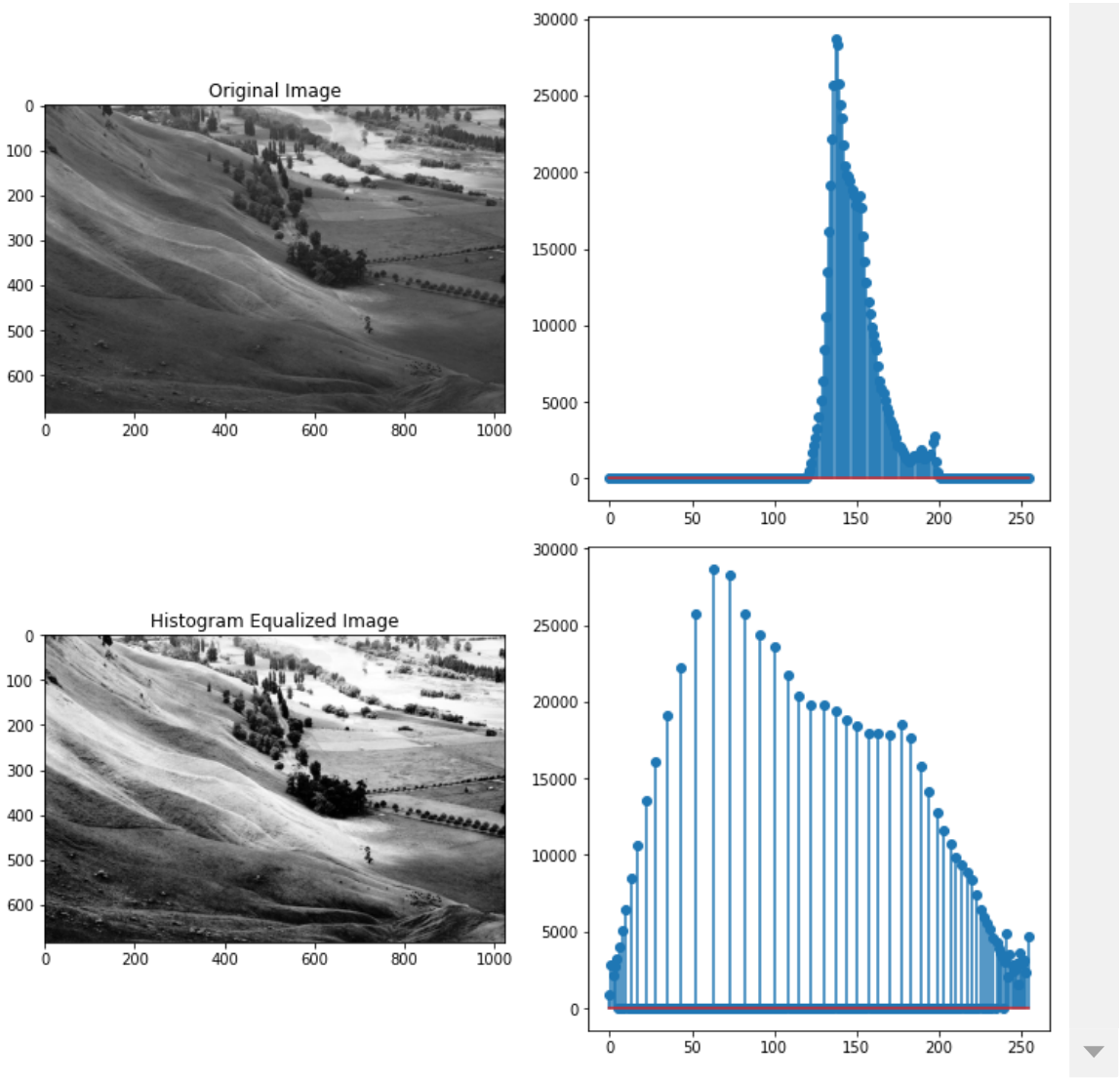


Histogram equalization using built in function

In [20]:

```
# you can do the above using a direct built in function --> cv2.equalizeHist(img)

img2 = cv2.imread('Unequalized_image.jpg',0)
equ = cv2.equalizeHist(img2)

count,value = np.histogram(img2,255,[0,255])
count2,value2 = np.histogram(equ,255,[0,255])

# plot the image

plt.figure(figsize=(10,10))

plt.subplot(221)
plt.title('Original Image')
plt.imshow(img2, cmap='gray')

plt.subplot(222)
plt.stem(count)

plt.subplot(223)
plt.title('Histogram Equalized Image')
plt.imshow(equ, cmap='gray')

plt.subplot(224)
plt.stem(count2)

plt.tight_layout()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18: UserWarni
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: UserWarni
```



Conclusion :

1. We implemented the code to plot the histogram of the given images. It was observed that the histogram of a dark image is on the lower side i.e.towards 0 and the histogram of a bright image is on the higher side and the histogram of a low contrast image is at the center.
2. In all the 3 images because the histogram is bunched very close together, we cannot discriminate/ distinguish the feautres. While the histogram of a high contrast image is evenly spread.
3. We implemented the code of the histogram equalization to equalize to convert any given image into a high contrast image. The result shows that the features of the unequalised image are clearly seen.