**EXPERIMENT NO. 2**

**PERFORM HISTOGRAM CALCULATION AND EQUALIZATION**

**EXPERIMENT NO. 2**

**AIM**: To perform histogram calculation and equalization

**OBJECTIVES:**

1. To obtain and plot histogram of the given image
2. To perform enhancement of image using histogram equalization.

**EQUIPMENTS/SOFTWARE: Python, OpenCV, NumPy, Matplotlib**

**THEORY:**

**Histogram Equalisation**

Histogram of a digital image with gray levels in range [0,L-1] is a discrete function h(rk) = nk where rk -kth gray level and nk = no. of pixels of an image having gray level rk.

In histogram there are 3 possibilities as follows,

1. For a dark image the components of histogram on the low (dark) side.

2. For a bright image the component are on high ( bright ) side

3. For an image with low contrast they are in the middle of gray side.

Histogram equalization is done to spread there component uniformly over the gray scale as far as possible.

This is obtained by function

Where

Thus processed image is obtained by mapping each pixel with level rk into a corresponding pixel with level Sk in output image. This transformation is called Histogram equalization

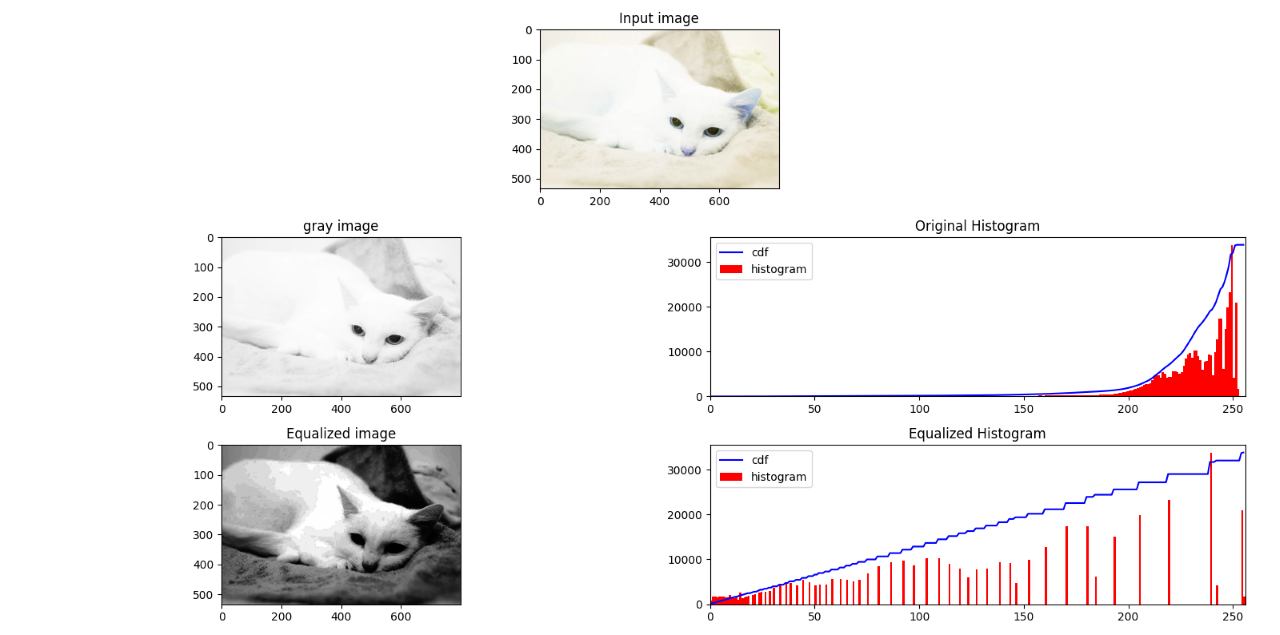
**ALGORITHM:**

1. Read the input image and its size.
2. Obtain probability of each the gray level values of each pixel from the image.
3. Compute CDF for each gray value.
4. Compute new value for each input grey level by multiplying its CDF by 255.
5. Replace the input gray values with corresponding new values Sk.
6. Plot the equalized histogram and original histogram
7. Display the original and the equalized image.

**CODE:**

import numpy as np  
from matplotlib import pyplot as plt  
from matplotlib import gridspec  
import cv2  
img = cv2.imread("C:\\Users\\admin\\Downloads\\low contrast.jpeg")  
  
def histplot(gray):  
    hist,bins = np.histogram(gray.flatten(),256,[0,256])  
    cdf = hist.cumsum()  
    cdf\_normalized = cdf\*float(hist.max(initial=None))/cdf.max()  
    return cdf\_normalized  
  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
cdf\_normalized1 = histplot(gray)  
equ = cv2.equalizeHist(gray)  
cdf\_normalized2 = histplot(equ)  
  
fig = plt.figure(figsize=(8,8))  
G = gridspec.GridSpec(3, 2)  
  
axes\_1 = plt.subplot(G[0, :])  
plt.title('Input image')  
plt.imshow(img)  
  
axes\_2 = plt.subplot(G[1, :-1])  
plt.imshow(gray, cmap=plt.cm.gray)  
plt.title('gray image')  
  
axes\_3 = plt.subplot(G[1, 1:])  
plt.plot(cdf\_normalized1, color='b')  
plt.hist(gray.flatten(), bins=256, range=[0,256], color='r')  
plt.xlim([0, 256])  
plt.legend(('cdf','histogram'), loc = 'upper left')  
plt.title('Original Histogram')  
  
axes\_4 = plt.subplot(G[2, :-1])  
plt.imshow(equ, cmap=plt.cm.gray)  
plt.title('Equalized image')  
  
axes\_5 = plt.subplot(G[2, 1:])  
plt.plot(cdf\_normalized2, color='b')  
plt.hist(equ.flatten(), bins=256, range=[0,256], color='r')  
plt.xlim([0, 256])  
plt.legend(('cdf','histogram'), loc = 'upper left')  
plt.title('Equalized Histogram')  
plt.tight\_layout()  
plt.show()

**OUTPUT:**



**CONCLUSION:**

**In this experiment we are able to perform Histogram calculation and equalization with python and its image processing libraries like OpenCV, matplotlib and Numpy. Now we are to able perform image conversion (RGB to Grayscale), Plotting of histogram, reading and showing the output image on screen.**