**EXPERIMENT:**

**AIM:** implement preprocessing techniques, including histogram analysis and histogram equalization

**SOFTWARE:** Python(spyder)

**THEORY:**

**Histogram**

Histogram is a graphical representation of the intensity distribution of an image. In simple terms, it represents the number of pixels for each intensity value considered.

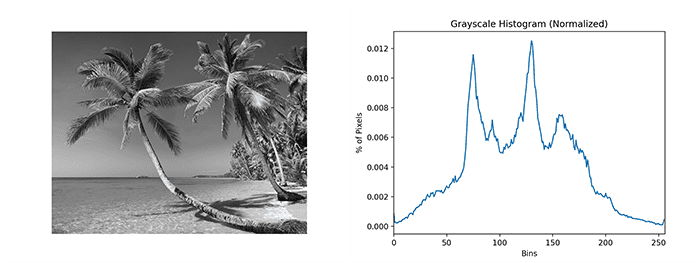


In the above figure, X-axis represents the tonal scale (black at the left and white at the right), and Y-axis represents the number of pixels in an image. Here, the histogram shows the number of pixels for each brightness level (from black to white), and when there are more pixels, the peak at the certain brightness level is higher.

**Histogram equalization:**

Histogram equalization is a basic image processing technique that can improve an image’s overall contrast.

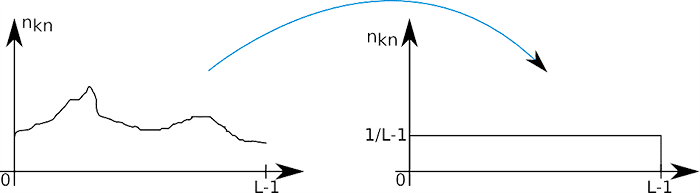
Applying histogram equalization starts by computing the histogram of pixel intensities in an input grayscale/single-channel image:

****

**Figure 1:** Left: Our original input grayscale image. Right: Computing the histogram of the grayscale image.

Notice how our histogram has numerous peaks, indicating there are a good number of pixels binned to those respective buckets. With histogram equalization, our goal is to spread these pixels to buckets that don’t have as many pixels binned to them.

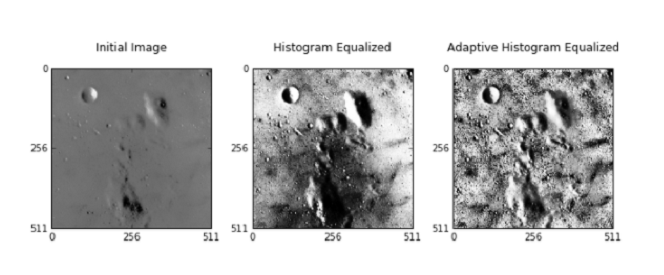
Mathematically, what this means is that we’re attempting to apply a linear trend to our cumulative distribution function (CDF):



**Figure 2:** The histogram equalization goal gives the output image a linear CDF

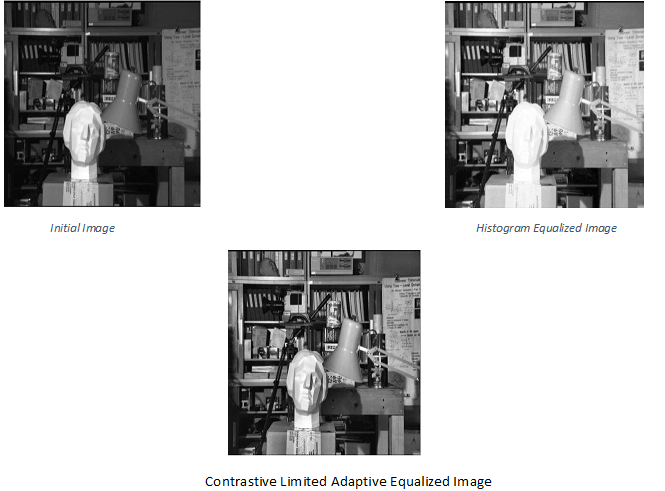
* **Adaptive Histogram Equalization**

Adaptive Histogram Equalization differs from ordinary histogram equalization in the respect that the adaptive method computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image. It is therefore suitable for improving the local contrast and enhancing the definitions of edges in each region of an image.



**Contrastive Limited Adaptive Equalization**

Contrast Limited AHE (CLAHE) differs from adaptive histogram equalization in its contrast limiting. In the case of CLAHE, the contrast limiting procedure is applied to each neighborhood from which a transformation function is derived. CLAHE was developed to prevent the over amplification of noise that adaptive histogram equalization can give rise to.



**PROGRAM:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load image in grayscale

image = cv2.imread(r"C:/Users/HP/Desktop/cv img.jpg", cv2.IMREAD\_GRAYSCALE)

# Calculate histogram

hist, bins = np.histogram(image.flatten(), 256, [0, 256])

# Apply histogram equalization

equalized\_image = cv2.equalizeHist(image)

# Calculate histogram of the equalized image

equalized\_hist, \_ = np.histogram(equalized\_image.flatten(), 256, [0, 256])

# Plot original image, original histogram, equalized image, and equalized histogram

plt.subplot(2, 2, 1)

plt.imshow(image, cmap='gray')

plt.title('Original Image')

plt.subplot(2, 2, 2)

plt.plot(hist, color='black')

plt.title('Original Histogram')

plt.subplot(2, 2, 3)

plt.imshow(equalized\_image, cmap='gray')

plt.title('Equalized Image')

plt.subplot(2, 2, 4)

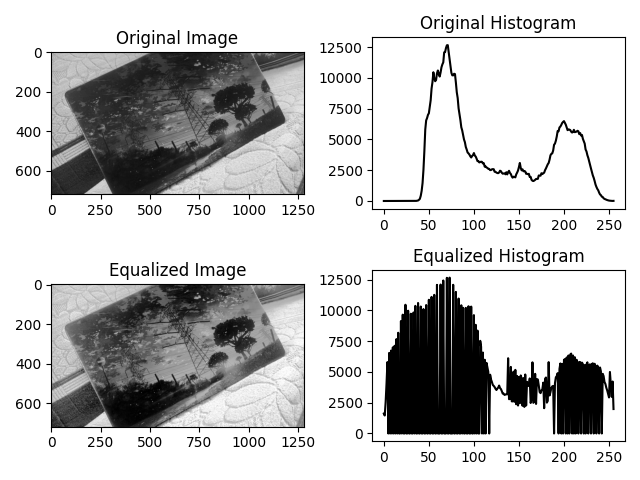
plt.plot(equalized\_hist, color='black')

plt.title('Equalized Histogram')

plt.tight\_layout()

plt.show()

output:



**conclusion:**

the experiment demonstrates the importance of histogram analysis and equalization techniques in enhancing image quality and visibility of details.