**EX.NO:1 DATE: 09.01.2025**

**RGB TO GRAYSCALE CONVERSION, HISTOGRAM EQUALISATION AND IMAGE RESIZING**

# Aim

To load image, and perform grayscaling, Resizing and Histogram Equalisation using OpenCV module in Python

# Algorithm

**1. RGB to Grayscale Conversion (Without Function)**

1. Read the Input Image:
   * Load the RGB image using cv2.imread().
2. Color Space Transformation:
   * Change the color format from BGR to RGB using cv2.cvtColor().
3. Channel Separation:
   * Extract individual Red, Green, and Blue channels from the image.
4. Define Luminance Weights:
   * Use the weights for luminance:
     1. r\_const = 0.2126 (Red weight)
     2. g\_const = 0.7152 (Green weight)
     3. b\_const = 0.0722 (Blue weight)
5. Calculate Grayscale Values:
   * Apply the grayscale formula:
     1. Gray Value = (r\_const \* R) + (g\_const \* G) + (b\_const \* B)
   * Optionally, incorporate gamma correction if needed (default gamma: 1).
6. Visualize and Save:
   * Use matplotlib to show both RGB and grayscale versions.
   * Save the grayscale image via cv2.imwrite().

**2. RGB to Grayscale Conversion (Using Function)**

1. Import Required Modules:
   * Import libraries like cv2 and matplotlib.pyplot.
2. Load the Input Image:
   * Read the image in RGB format using cv2.imread().
3. Convert to Grayscale:
   * Utilize cv2.cvtColor() with the parameter cv2.COLOR\_BGR2GRAY for conversion.
4. Display Both Images:
   * Show the original RGB and grayscale images using cv2.imshow() or matplotlib’s imshow().
5. Save the Grayscale Output (Optional):
   * Save the resultant grayscale image with cv2.imwrite().

**3. Histogram Equalization (Using a Library Function)**

1. Load Libraries:
   * Import cv2 and matplotlib.pyplot.
2. Read the Image in Grayscale Mode:
   * Use cv2.imread() to load the grayscale image.
3. Perform Histogram Equalization:
   * Use cv2.equalizeHist() to equalize the histogram.
4. Display and Compare Results:
   * Show both the original and the equalized images using either matplotlib or cv2.imshow().

**4. Histogram Equalization (Manual Method)**

1. Import Dependencies:
   * Import numpy and cv2.
2. Load the Grayscale Image:
   * Read the image using cv2.imread().
3. Calculate the Histogram:
   * Compute the image histogram with cv2.calcHist().
4. Normalize the Histogram:
   * Normalize to create a probability distribution.
5. Find the Cumulative Distribution Function (CDF):
   * Compute the CDF from the normalized histogram.
6. Remap Pixel Intensities:
   * Use the CDF to remap pixel intensity values.
7. Generate Equalized Image:
   * Create the equalized image based on the remapped values.
8. Visualize Results:
   * Display both the original and the equalized images.

**5. Image Resizing**

1. Read the Input Image:
   * Load the image using cv2.imread() and convert from BGR to RGB.
2. Perform Resizing:
   * Create three resized versions of the image:
     1. One scaled down (smaller).
     2. One scaled up (larger).
     3. One resized using linear interpolation.
3. Prepare for Visualization:
   * Assign titles such as "Original", "Half", "Bigger", and "Interpolation Nearest".
   * Add resized images to a list.
4. Show Resized Outputs:
   * Display the original and resized images in a grid layout using matplotlib or cv2.imshow().

# Code

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns

print(cv2.\_\_version\_\_)

4.9.0

img=cv2.imread("desktop-1920x1080.jpg",cv2.IMREAD\_COLOR)

if img is None:  
 print("Image not loaded properly.")  
else:  
 print("Image loaded successfully.")

Image loaded successfully.

cv2.startWindowThread()  
cv2.imshow("image",img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

# Loading Image

lena1 = cv2.imread("lena.jpeg",cv2.IMREAD\_COLOR)  
lena2 = cv2.imread("lena.jpeg",cv2.IMREAD\_GRAYSCALE)

cv2.startWindowThread()  
cv2.imshow("Lena Colored",lena1)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

plt.imshow(lena1)  
plt.axis('off')  
plt.show()



cv2.startWindowThread()  
cv2.imshow("Lena Grayscale",lena2)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
plt.imshow(lena2)  
plt.axis('off')  
plt.show()



lenaRGB = cv2.cvtColor(lena1,cv2.COLOR\_BGR2RGB)  
plt.imshow(lenaRGB)  
plt.axis('off')  
plt.show()

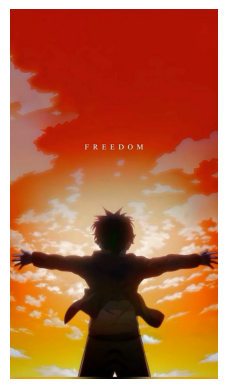


eren1 = cv2.imread("eren.jpg",cv2.IMREAD\_COLOR)  
eren2 = cv2.imread("eren.jpg",cv2.IMREAD\_GRAYSCALE)

erenRGB = cv2.cvtColor(eren1,cv2.COLOR\_BGR2RGB)  
plt.imshow(erenRGB)  
plt.axis('off')  
plt.show()



cv2.startWindowThread()  
cv2.imshow("Eren Colored",eren1)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
plt.imshow(eren1)  
plt.axis('off')  
plt.show()



cv2.startWindowThread()  
cv2.imshow("Eren Grayscale",eren2)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
plt.imshow(eren2)  
plt.axis('off')  
plt.show()



# RGB to GRAY (inbuilt)

lenaGray = cv2.cvtColor(lenaRGB,cv2.COLOR\_RGB2GRAY)  
plt.imshow(lenaGray,cmap = 'gray')  
plt.axis('off')  
plt.show()



ErenGRAY = cv2.cvtColor( erenRGB, cv2.COLOR\_RGB2GRAY)  
plt.imshow(ErenGRAY,cmap='gray')  
plt.axis('off')  
plt.show()



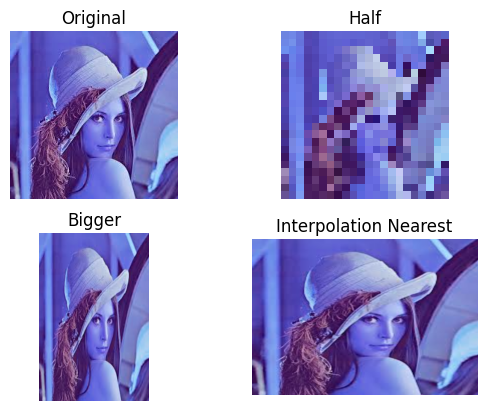
# RGB TO GRAY (MANUAL)

(row, col) = lena1.shape[0:2]

for i in range(row):  
 for j in range(col):  
 lena1[i, j] = sum(lena1[i, j]) \* 0.33  
cv2.startWindowThread()  
cv2.imshow('Grayscale Image', lena1)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

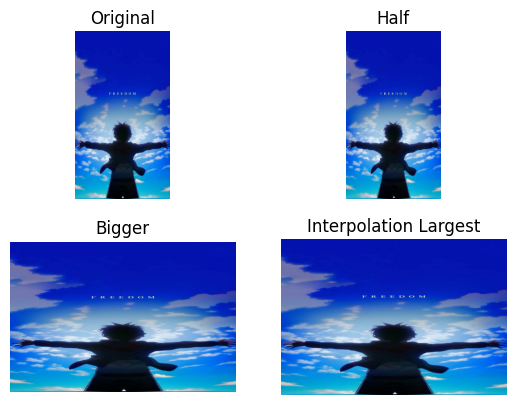
# IMAGE RESIZING

image = cv2.imread("lena.jpeg", 1)  
  
half = cv2.resize(image, (0, 0), fx = 0.1, fy = 0.1)  
bigger = cv2.resize(image, (1050, 1610))  
  
stretch\_near = cv2.resize(image, (780, 540),   
 interpolation = cv2.INTER\_LINEAR)  
  
  
Titles =["Original", "Half", "Bigger", "Interpolation Nearest"]  
images =[image, half, bigger, stretch\_near]  
count = 4  
  
for i in range(count):  
 plt.subplot(2, 2, i + 1)  
 plt.title(Titles[i])  
 plt.imshow(images[i])  
 plt.axis('off')  
plt.show()



free = cv2.imread("eren.jpg",1)  
freeRGB = cv2.cvtColor(free, cv2.COLOR\_BGR2RGB)  
Half = cv2.resize(freeRGB, (0,0), fx= 0.25, fy= 0.25)  
Bigger = cv2.resize(freeRGB, (1080,720))  
Stretch\_long = cv2.resize(freeRGB, (780,540), interpolation = cv2.INTER\_AREA)

Titles =["Original", "Half", "Bigger", "Interpolation Largest"]  
images =[freeRGB, Half, Bigger, Stretch\_long]  
count = 4  
  
for i in range(count):  
 plt.subplot(2, 2, i + 1)  
 plt.title(Titles[i])  
 plt.imshow(images[i])  
 plt.axis('off')  
plt.show()



# Histogram Equalization

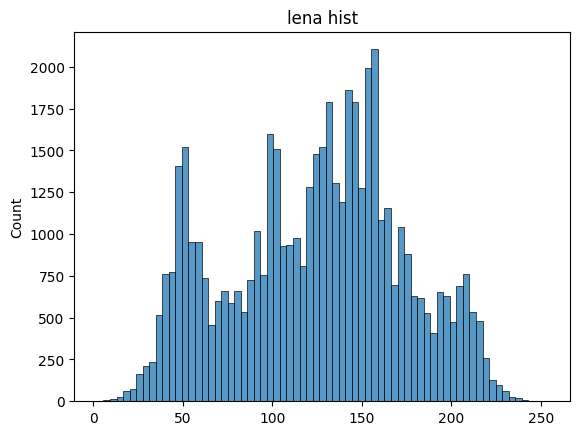
lenaequ = cv2.equalizeHist(lenaGray)  
lenares = np.hstack((lenaequ, lena2))  
  
cv2.imshow("image",lenares)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
plt.imshow(lenares,cmap = 'gray')  
plt.axis('off')

(-0.5, 449.5, 224.5, -0.5)



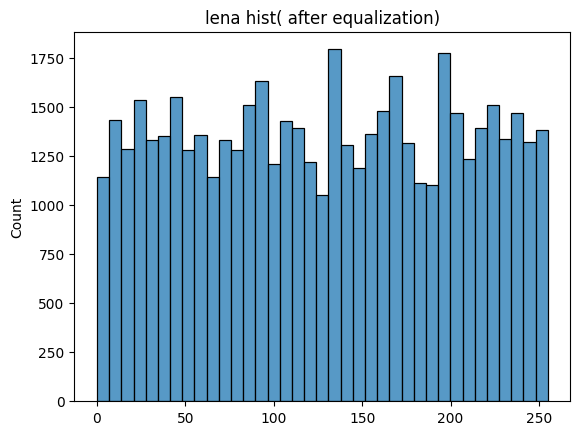
sns.histplot(lena2.flatten())  
plt.title('lena hist')

Text(0.5, 1.0, 'lena hist')



sns.histplot(lenaequ.flatten())  
plt.title('lena hist( after equalization)')

Text(0.5, 1.0, 'lena hist( after equalization)')



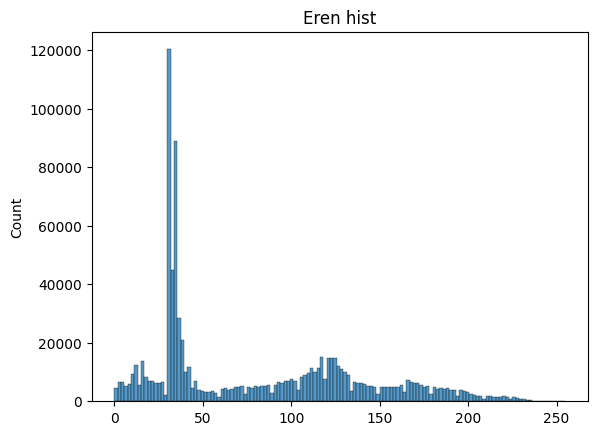
erenequ = cv2.equalizeHist(ErenGRAY)  
erenres = np.hstack((erenequ,ErenGRAY))  
cv2.imshow("Eren Histogram Equalised Images",erenres)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
plt.imshow(erenres,cmap = 'gray')  
plt.axis('off')

(-0.5, 1471.5, 1307.5, -0.5)



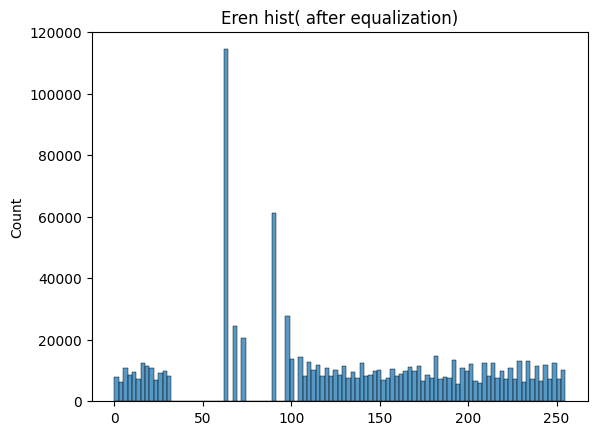
sns.histplot(eren2.flatten())  
plt.title('Eren hist')

Text(0.5, 1.0, 'Eren hist')



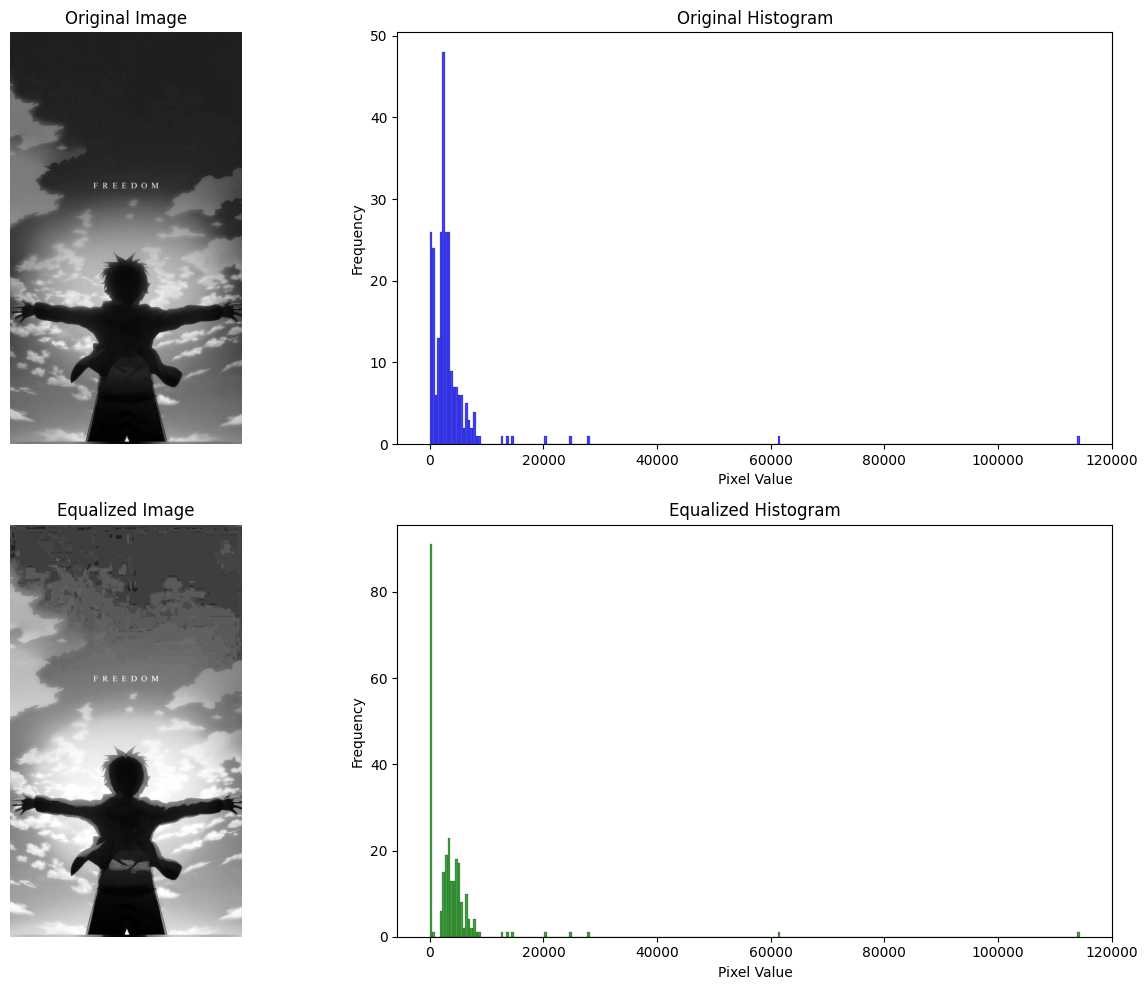
sns.histplot(erenequ.flatten())  
plt.title('Eren hist( after equalization)')

Text(0.5, 1.0, 'Eren hist( after equalization)')

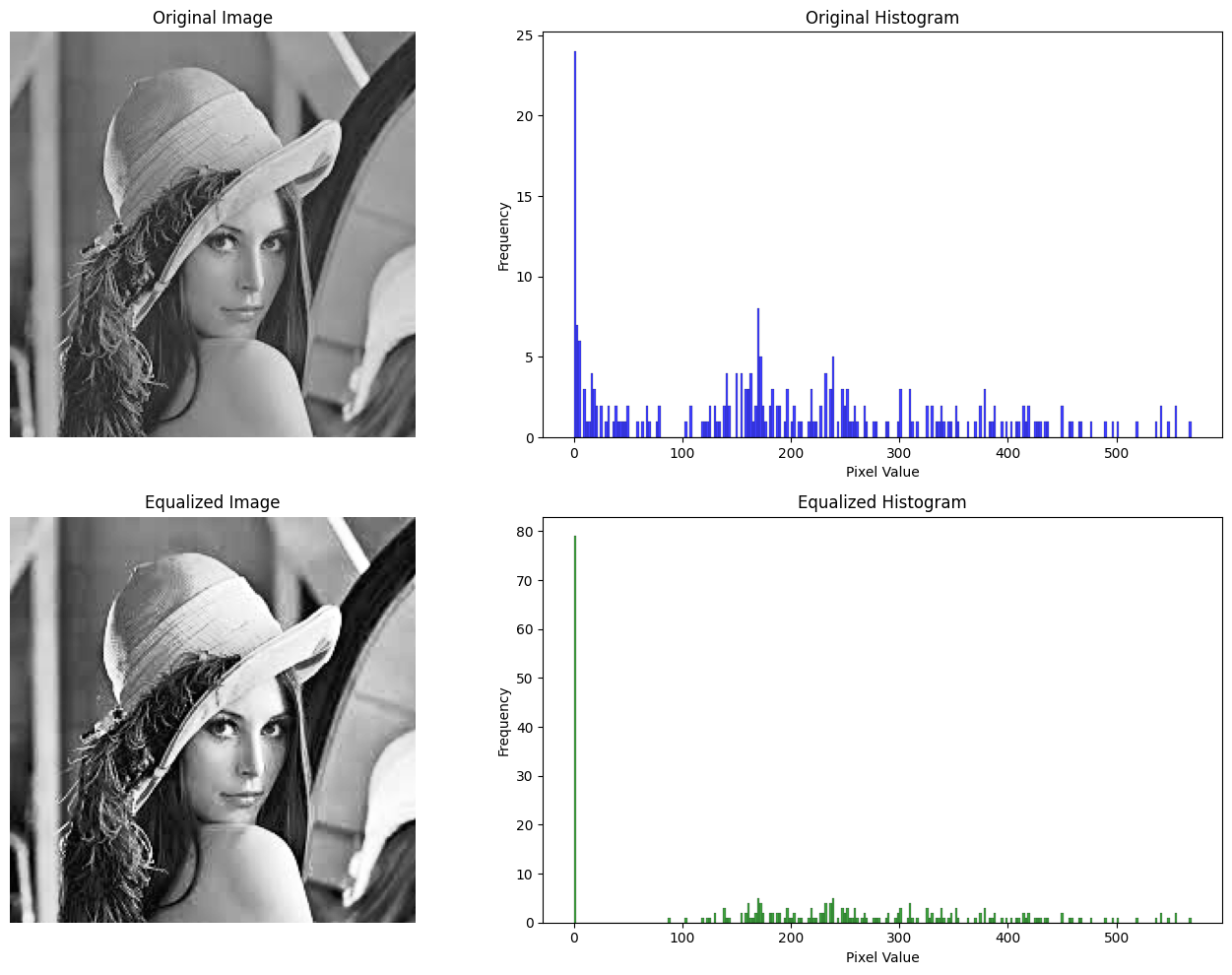


# Histogram Equalisation without using inbuilt function

def histogram\_equalization(image):  
 rows, cols = image.shape  
 histogram = np.zeros(256, dtype=int)  
 for i in range(rows):  
 for j in range(cols):  
 histogram[image[i, j]] += 1  
 total\_pixels = rows \* cols  
 normalized\_histogram = histogram / total\_pixels  
 cdf = np.zeros(256, dtype=float)  
 cdf[0] = normalized\_histogram[0]  
 for i in range(1, 256):  
 cdf[i] = cdf[i - 1] + normalized\_histogram[i]  
 cdf\_scaled = np.round(cdf \* 255).astype(np.uint8)  
 equalized\_image = np.zeros\_like(image)  
 for i in range(rows):  
 for j in range(cols):  
 equalized\_image[i, j] = cdf\_scaled[image[i, j]]  
 return equalized\_image, histogram  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 input\_image = cv2.imread("eren.jpg", cv2.IMREAD\_GRAYSCALE)  
 if input\_image is None:  
 print("Error: Image not found!")  
 exit()  
   
 equalized\_image, original\_histogram = histogram\_equalization(input\_image)  
  
 equalized\_histogram = np.zeros(256, dtype=int)  
 for value in equalized\_image.ravel():  
 equalized\_histogram[value] += 1  
  
 plt.figure(figsize=(14, 10))  
  
 plt.subplot(2, 2, 1)  
 plt.imshow(input\_image, cmap='gray')  
 plt.title("Original Image")  
 plt.axis('off')  
  
 plt.subplot(2, 2, 2)  
 sns.histplot(data=original\_histogram, bins=256, kde=False, color='blue')  
 plt.title("Original Histogram")  
 plt.xlabel("Pixel Value")  
 plt.ylabel("Frequency")  
   
 plt.subplot(2, 2, 3)  
 plt.imshow(equalized\_image, cmap='gray')  
 plt.title("Equalized Image")  
 plt.axis('off')  
  
 plt.subplot(2, 2, 4)  
 sns.histplot(data=equalized\_histogram, bins=256, kde=False, color='green')  
 plt.title("Equalized Histogram")  
 plt.xlabel("Pixel Value")  
 plt.ylabel("Frequency")  
  
 plt.tight\_layout()  
 plt.show()



if \_\_name\_\_ == "\_\_main\_\_":  
 input\_image = cv2.imread("lena.jpeg", cv2.IMREAD\_GRAYSCALE)  
 if input\_image is None:  
 print("Error: Image not found!")  
 exit()  
   
 equalized\_image, original\_histogram = histogram\_equalization(input\_image)  
  
 equalized\_histogram = np.zeros(256, dtype=int)  
 for value in equalized\_image.ravel():  
 equalized\_histogram[value] += 1  
  
 plt.figure(figsize=(14, 10))  
  
 plt.subplot(2, 2, 1)  
 plt.imshow(input\_image, cmap='gray')  
 plt.title("Original Image")  
 plt.axis('off')  
  
 plt.subplot(2, 2, 2)  
 sns.histplot(data=original\_histogram, bins=256, kde=False, color='blue')  
 plt.title("Original Histogram")  
 plt.xlabel("Pixel Value")  
 plt.ylabel("Frequency")  
  
 plt.subplot(2, 2, 3)  
 plt.imshow(equalized\_image, cmap='gray')  
 plt.title("Equalized Image")  
 plt.axis('off')  
  
 plt.subplot(2, 2, 4)  
 sns.histplot(data=equalized\_histogram, bins=256, kde=False, color='green')  
 plt.title("Equalized Histogram")  
 plt.xlabel("Pixel Value")  
 plt.ylabel("Frequency")  
  
 plt.tight\_layout()  
 plt.show()



# INFERENCE:

Had understood the concepts of image colour conversion, histogram equalisation and image resizing with the help of library open-cv

# RESULT:

Thus, the concepts of image colour conversion, histogram equalisation and image resizing have been implemented on the lena and eren image- Sample images.