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)

i. Read the Input Image:
   - Load the RGB image using cv2.imread().

ii. Color Space Transformation:
   - Change the color format from BGR to RGB using cv2.cvtColor().

iii. Channel Separation:
   - Extract individual Red, Green, and Blue channels from the image.

iv. Define Luminance Weights:
   - Use the weights for luminance:
     i. r_const = 0.2126 (Red weight)
     ii. g_const = 0.7152 (Green weight)
     iii. b_const = 0.0722 (Blue weight)

v. Calculate Grayscale Values:
   - Apply the grayscale formula:
     i. Gray Value = (r_const * R) + (g_const * G) + (b_const * B)
   - Optionally, incorporate gamma correction if needed (default gamma: 1).

vi. 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)

i. Import Required Modules:

- Import libraries like cv2 and matplotlib.pyplot.

ii. Load the Input Image:

- Read the image in RGB format using cv2.imread().

iii. Convert to Grayscale:

- Utilize cv2.cvtColor() with the parameter cv2.COLOR_BGR2GRAY for conversi on.

iv. Display Both Images:

- Show the original RGB and grayscale images using cv2.imshow() or matplotli b's imshow().

v. Save the Grayscale Output (Optional):

- Save the resultant grayscale image with cv2.imwrite().

3. Histogram Equalization (Using a Library Function)

i. Load Libraries:

- Import cv2 and matplotlib.pyplot.

ii. Read the Image in Grayscale Mode:

- Use cv2.imread() to load the grayscale image.

iii. Perform Histogram Equalization:

- Use cv2.equalizeHist() to equalize the histogram.

iv. Display and Compare Results:

- Show both the original and the equalized images using either matplotlib or cv 2.imshow().

4. Histogram Equalization (Manual Method)

i. Import Dependencies:

- Import numpy and cv2.

ii. Load the Grayscale Image:

- Read the image using cv2.imread().

iii. Calculate the Histogram:

- Compute the image histogram with cv2.calcHist().

iv. Normalize the Histogram:

- Normalize to create a probability distribution.

v. Find the Cumulative Distribution Function (CDF):

- Compute the CDF from the normalized histogram.

vi. Remap Pixel Intensities:

- Use the CDF to remap pixel intensity values.

vii. Generate Equalized Image:

- Create the equalized image based on the remapped values.

viii. Visualize Results:

- Display both the original and the equalized images.

5. Image Resizing

i. Read the Input Image:

- Load the image using cv2.imread() and convert from BGR to RGB.

ii. Perform Resizing:

- Create three resized versions of the image:
    i. One scaled down (smaller).
    ii. One scaled up (larger).
    iii. One resized using linear interpolation.

iii. Prepare for Visualization:

- Assign titles such as "Original", "Half", "Bigger", and "Interpolation Nearest".
- Add resized images to a list.

iv. Show Resized Outputs:

- Display the original and resized images in a grid layout using matplotlib or cv 2.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()
```

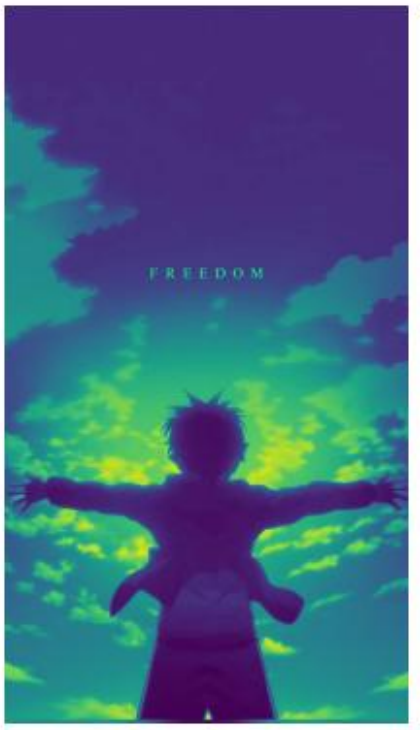
```
cv2.startWindowThread()
cv2.imshow("Lena Grayscale",lena2)
cv2.waitKey(0)
cv2.destroyAllWindows()
plt.imshow(lena2)
plt.axis('off')
plt.show()
```

```
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()
```

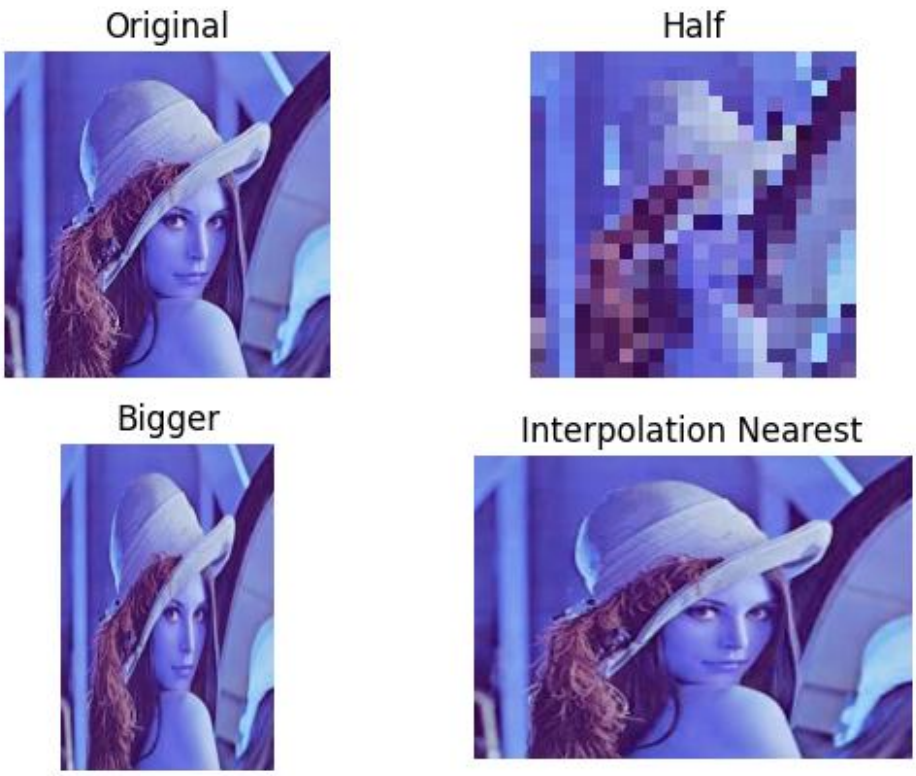
```
ErenGRAY = cv2.cvtColor( erenRGB, cv2.COLOR_RGB2GRAY)
plt.imshow(ErenGRAY,cmap='gray')
plt.axis('off')
plt.show()
```

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()
```

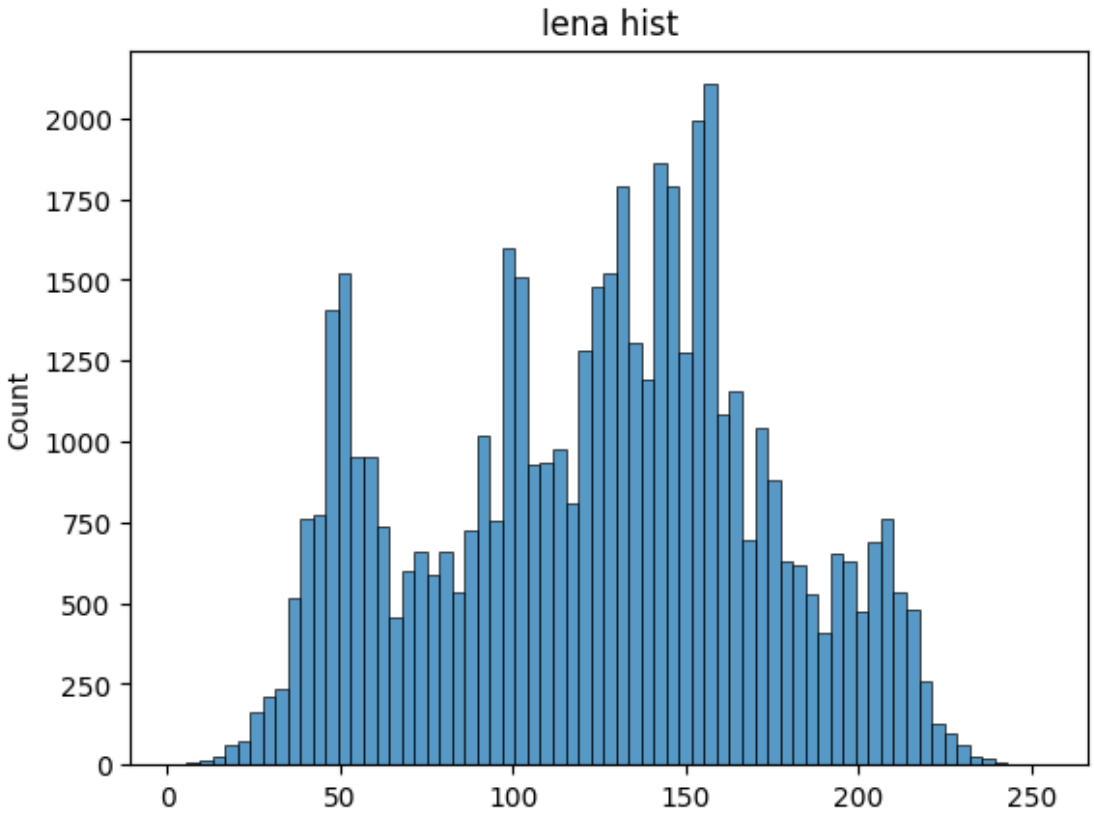
Histogram Equalization

```
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)
```

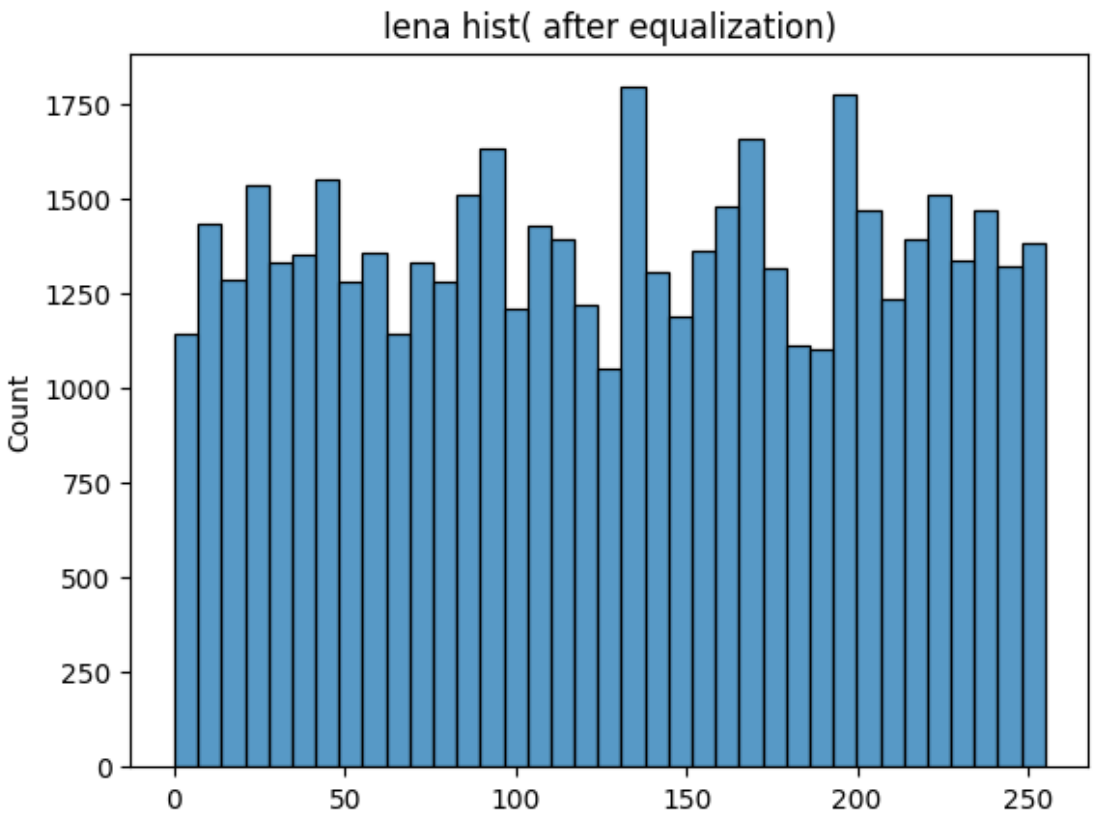
```
sns.histplot(lena2.flatten())
plt.title('lena hist')
```

```
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)')
```

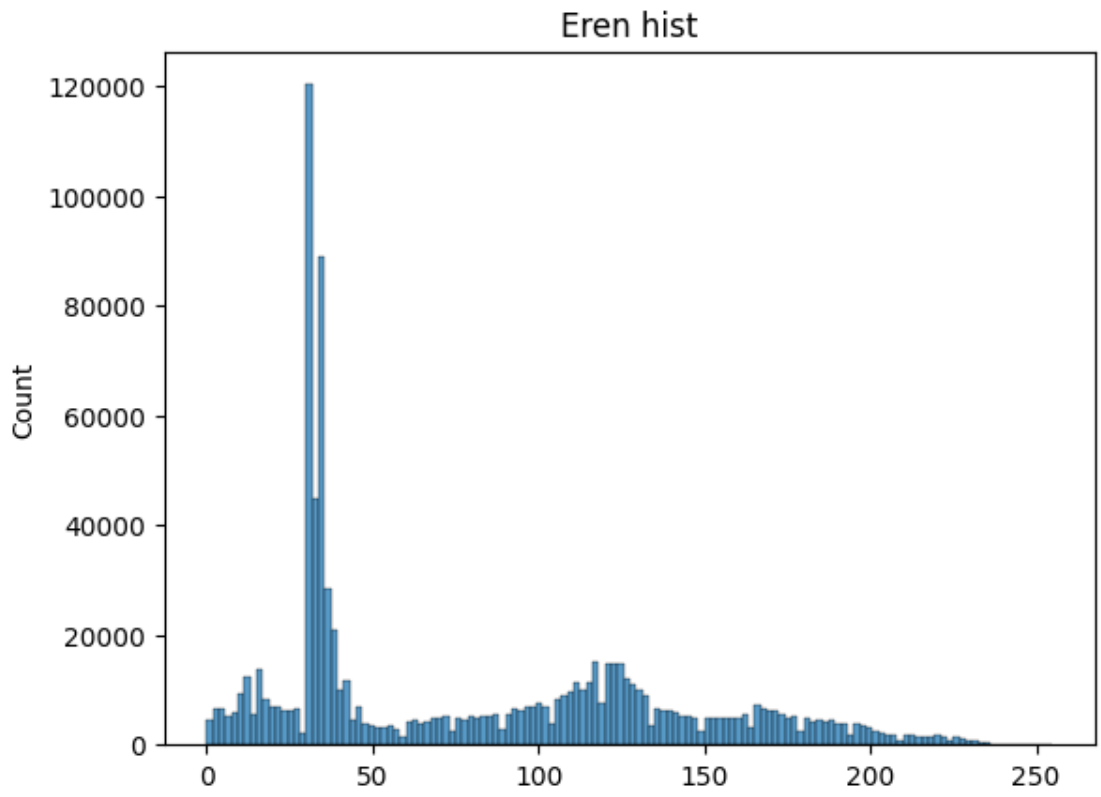



```
erenequ = cv2.equalizeHist(ErenGRAY)
erenres = np.hstack((erenequ,ErenGRAY))
cv2.imshow("Eren Histogram Equalised Images",erenres)
cv2.waitKey(0)
cv2.destroyAllWindows()
plt.imshow(erenres,cmap = 'gray')
plt.axis('off')
```

```
(-0.5, 1471.5, 1307.5, -0.5)
```

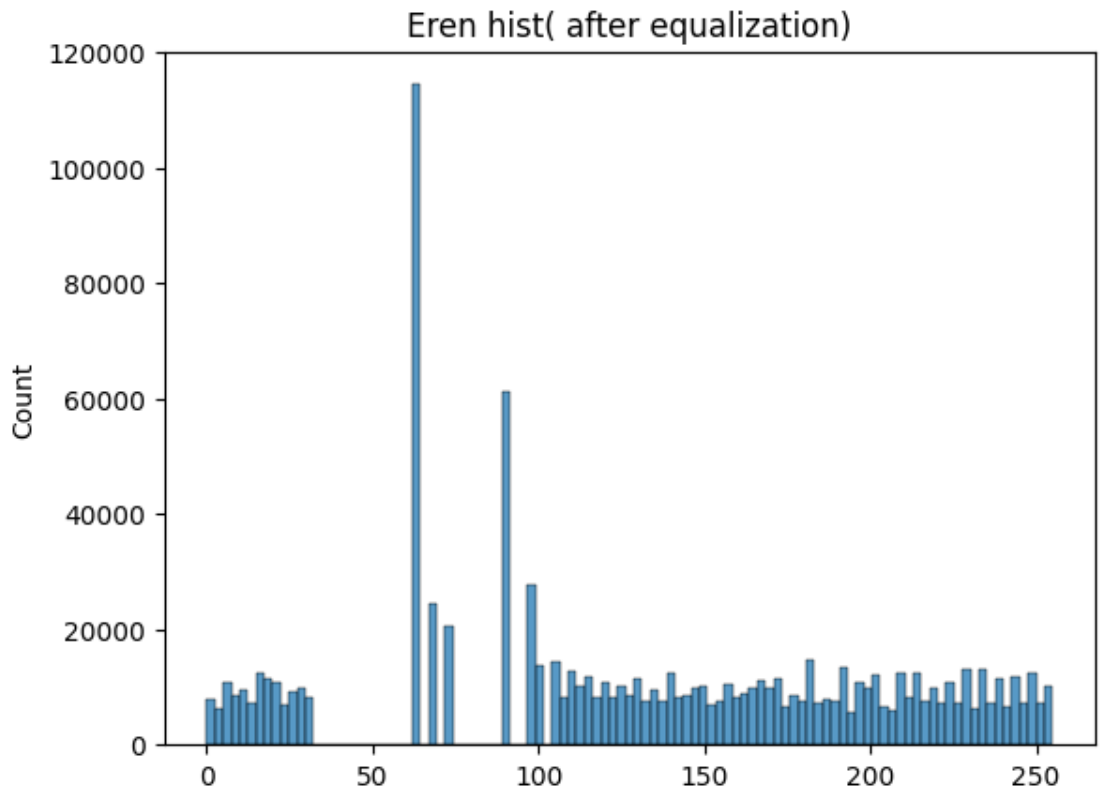

```
sns.histplot(eren2.flatten())
plt.title('Eren hist')
```

```
Text(0.5, 1.0, 'Eren hist')
```




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

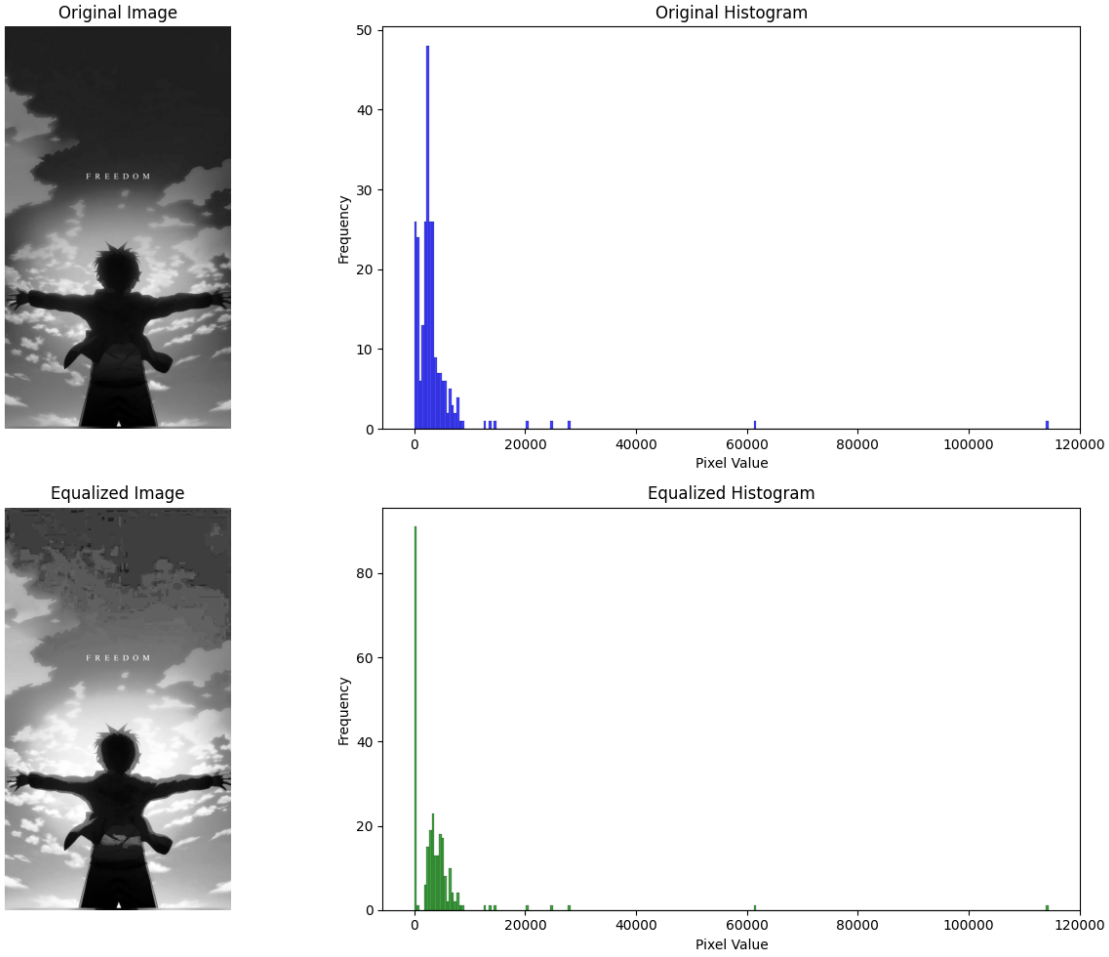
```
Text(0.5, 1.0, 'Eren hist( after equalization)')
```



Histogram Equalisation without using inbuilt function

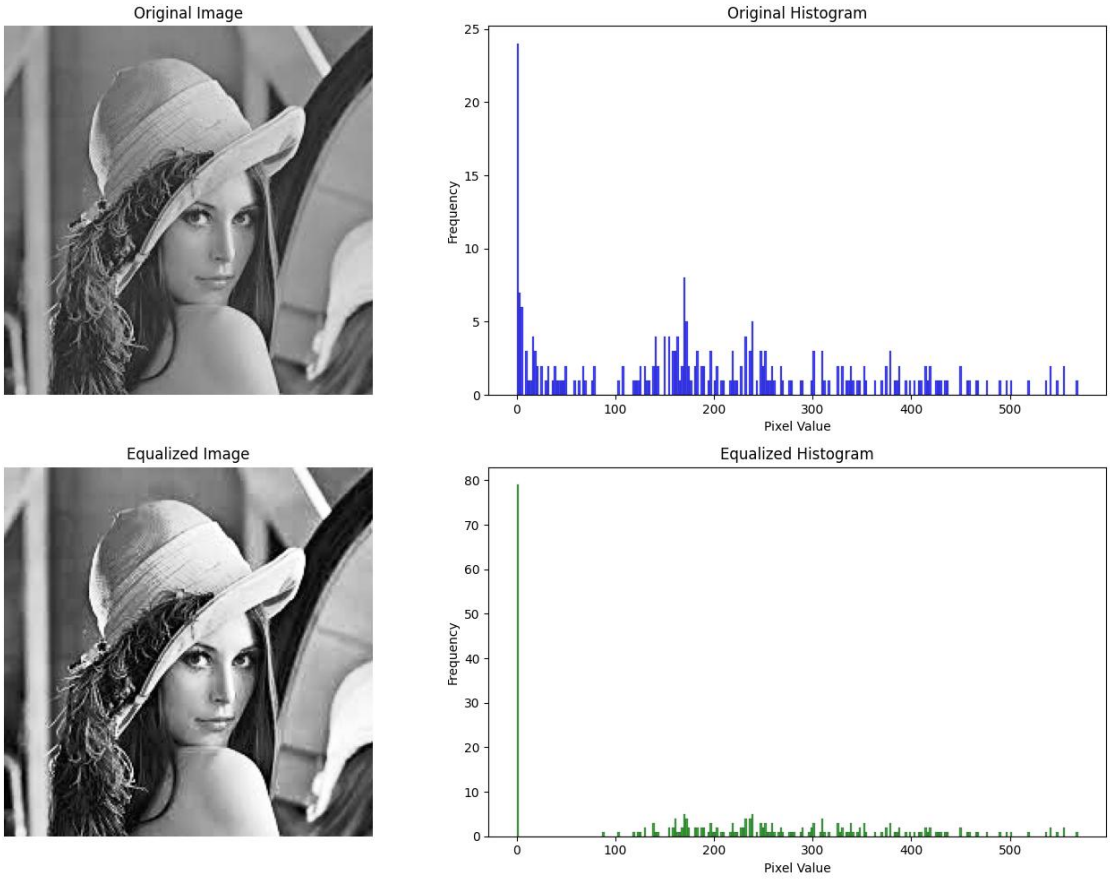
```
def histogram_equalization(image):
    rows, cols = image.shape
    histogram = np.zeros(256, dtype=int)
    for i in range(rows):
        for j in range(cols):
            histogram[image[i, j]] += 1
    total_pixels = rows * cols
    normalized_histogram = histogram / total_pixels
    cdf = np.zeros(256, 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.