Image And Video Processing Project Report

Name: Sahil Kumarwar

Registration No.: 201060035

Project Aim: Image Enhancement using Histogram Equalization

Theory:

Image Enhancement:

Image Enhancement is the process of improving visual quality of an image by manipulating its attributes, such as Contrast, Sharpness or Brightness, to enhance the image and make it more appealing. It involves techniques like Histogram Equalization, filtering, point operations etc. to enhance the overall appearance of the image.

Histogram Equalization:

In image processing, a histogram is a graphical representation of the distribution of pixel intensities in an image. It displays the frequency of occurrence of each intensity level in the image. The x-axis represents the intensity values, and the y-axis represents the frequency or count of pixels with that intensity.

Histogram equalization is a technique used to enhance the contrast and improve the details in an image by redistributing the pixel intensities. The main goal of histogram equalization is to transform the intensity distribution of an image so that it becomes more uniformly distributed across the entire range of possible intensity values.

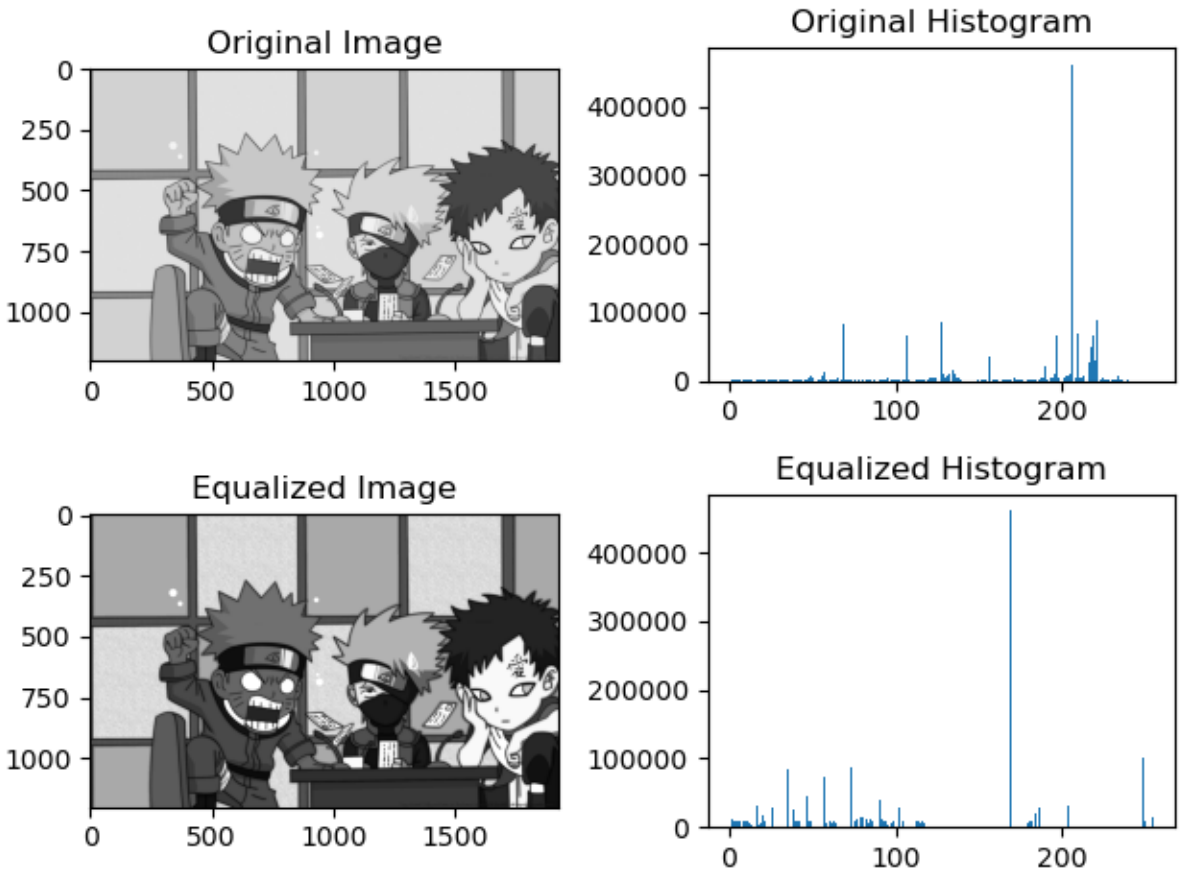
Steps to perform Histogram Equalization:

1. Import required libraries such as OpenCV, numpy and matplotlib.
2. Load the image using OpenCV and convert it to grayscale.
3. Compute the Histogram of the Image.
4. Compute the CDF by cumulatively summation.
5. Normalize the CDF to obtain cumulative Histogram.
6. Map the intensity values to their corresponding values in CDF.
7. Calculate the histogram of equalized image.
8. Display the original and equalized image.

Code:

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Read the image and convert it to grayscale
image_path = 'image2.jpg'
image = cv2.imread(image_path, 0)

# Calculate the histogram manually
histogram = np.zeros(256, dtype=int)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        intensity = image[i, j]
        histogram[intensity] += 1

# Compute the cumulative distribution function (CDF)
cdf = np.zeros(256, dtype=float)
cdf[0] = histogram[0]
for i in range(1, 256):
    cdf[i] = cdf[i-1] + histogram[i]

# Normalize the CDF
cdf_normalized = (cdf - cdf.min()) / (image.shape[0] * image.shape[1] - cdf.min())

# Map intensity values to their corresponding values in the CDF
equalized_image = np.zeros_like(image)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        intensity = image[i, j]
        equalized_image[i, j] = np.round(cdf_normalized[intensity] * 255)

# Calculate the histogram of the equalized image
equalized_histogram = np.zeros(256, dtype=int)
for i in range(equalized_image.shape[0]):
    for j in range(equalized_image.shape[1]):
        intensity = equalized_image[i, j]
        equalized_histogram[intensity] += 1

# Display the original and equalized images side by side
plt.subplot(2, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')

plt.subplot(2, 2, 2)
plt.bar(range(256), histogram)
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.bar(range(256), equalized_histogram)
plt.title('Equalized Histogram')

plt.tight_layout()
plt.show()

# Save the equalized image
output_path = 'equalized_image.jpg'
cv2.imwrite(output_path, equalized_image)
```

Output:



Conclusion:

In this project, we have implemented histogram equalization to enhance the contrast of the image. The result shows that the histogram equalization effectively enhanced the image by redistributing the pixel intensities and improving contrast. The enhanced image has enhanced details and the visual appearance is improved. The equalized histogram also indicates a more balanced distribution of pixel intensities.