Histogram Equalization

Muskan-230104063

October, 2024

1 Loading Image as Grayscale Image

Here are the code blocks for loading an image and performing histogram equalization:

Import Necessary Libraries

```python
# Import necessary libraries
import cv2  # for image processing
import numpy as np  # for numerical operations
import matplotlib.pyplot as plt  # for plotting
import os  # for file handling

# Create a directory to save images
output_dir = 'output_images'
os.makedirs(output_dir, exist_ok=True)
```

Load the Image in Grayscale

```python
# Load the image in grayscale
image = cv2.imread('/kaggle/input/hist-eq/acad1.jpg', cv2.
    IMREAD_GRAYSCALE)

# Save the original grayscale image
cv2.imwrite(os.path.join(output_dir, 'original_image.jpg'),
    image)
```

Display the Original Image and its Histogram

```python
# Step 1: Display the original image and its histogram
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.hist(image.ravel(), bins=256, range=[0, 256])
plt.title('Original Histogram')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.show()

# Save the histogram as an image
plt.savefig(os.path.join(output_dir, 'original_histogram.jpg
    '))
plt.close()
```

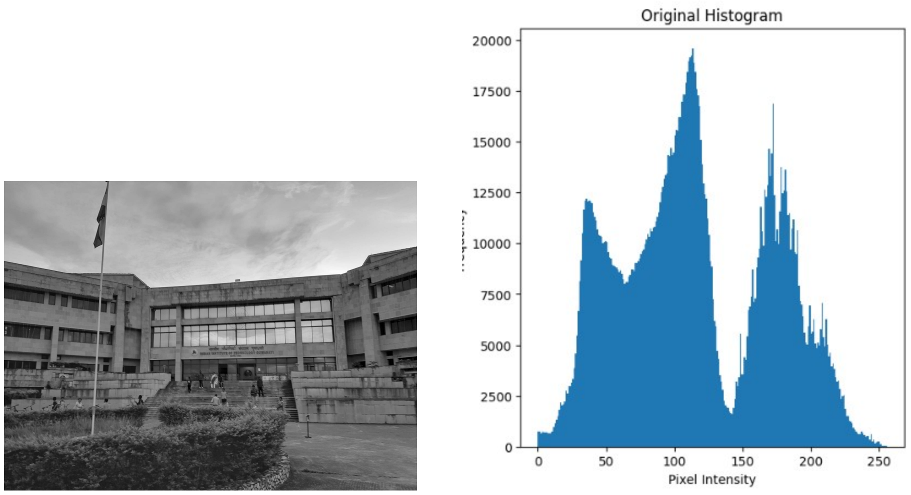

Figure 1: Original Image and Original Histogram

Compute the Cumulative Distribution Function (CDF)

```
# Step 2: Compute the cumulative distribution function (CDF)
hist, bins = np.histogram(image.flatten(), 256, [0, 256])
cdf = hist.cumsum()  # cumulative sum
cdf_normalized = cdf * hist.max() / cdf.max()  # normalize

# Display the cumulative distribution function
plt.plot(cdf_normalized, color='b')
plt.hist(image.flatten(), bins=256, range=[0, 256], color='r
    ', alpha=0.5)
plt.title('CDF and Histogram of Original Image')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.legend(('CDF', 'Histogram'), loc='upper left')
plt.show()

# Save the CDF and histogram as an image
plt.savefig(os.path.join(output_dir, 'cdf_histogram.jpg'))
plt.close()
```

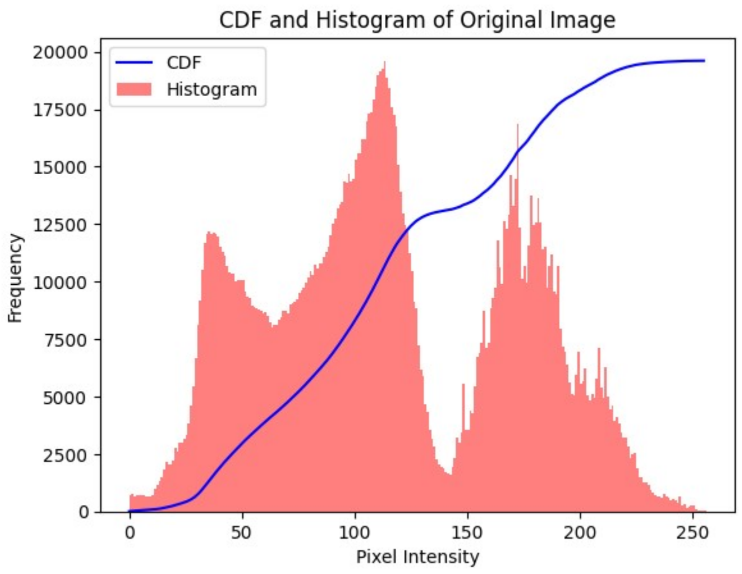

Figure 2: CDF and Histogram of Original Image

Apply Histogram Equalization

```
# Step 3: Apply histogram equalization
equalized_image = cv2.equalizeHist(image)

# Display the equalized image and its histogram
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(equalized_image, cmap='gray')
plt.title('Equalized Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.hist(equalized_image.ravel(), bins=256, range=[0, 256])
plt.title('Equalized Histogram')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.show()

# Save the equalized image and its histogram
cv2.imwrite(os.path.join(output_dir, 'equalized_image.jpg'),
     equalized_image)
plt.savefig(os.path.join(output_dir, 'equalized_histogram.
    jpg'))
plt.close()
```

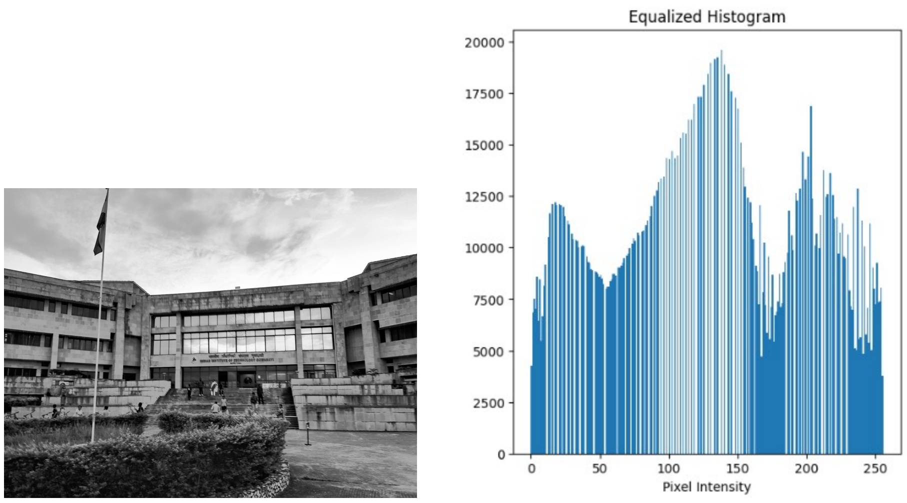

Figure 3: Equalized Image and Equalized Histogram

Show Both Original and Equalized Images for Comparison

```
# Step 4: Show both original and equalized images for
    comparison
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(equalized_image, cmap='gray')
plt.title('Equalized Image')
plt.axis('off')
plt.show()

# Save the comparison image
plt.savefig(os.path.join(output_dir, 'comparison_image.jpg')
    )
plt.close()
```

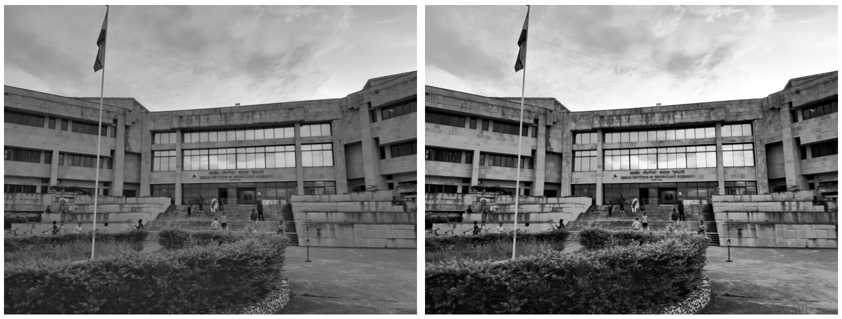
Figure 4: Comparison of Original and Equalized Images

2 Loading Image as RGB Image

Here are the code blocks for loading an image and performing histogram equalization:

Import Necessary Libraries

```
# Import necessary libraries
import cv2  # for image processing
import numpy as np  # for numerical operations
import matplotlib.pyplot as plt  # for plotting
```

Loading image as RGB and displaying original image

```
#Load the second image in color
image2 = cv2.imread('/kaggle/input/hist-eq/acad3.jpg')

# Convert from BGR to RGB format
image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)
```

```
# Display the original image
plt.figure(figsize=(6, 6))
plt.imshow(image2)
plt.title('Original Image 2')
plt.axis('off')
plt.show()
```

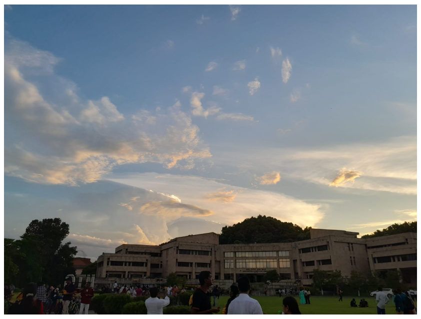

Figure 5: Original Images

Separating Red Green and Blue Channels

```
# Separate the RGB channels
r, g, b = cv2.split(image2)

# Display the separated RGB channels
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(r, cmap='Reds')
plt.title('Red Channel')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(g, cmap='Greens')
plt.title('Green Channel')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(b, cmap='Blues')
plt.title('Blue Channel')
plt.axis('off')
plt.show()
```

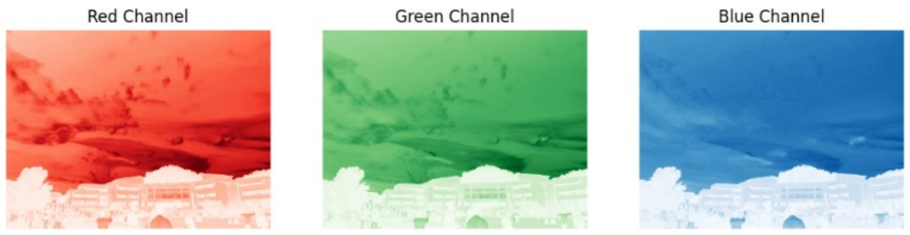

Figure 6: Separated Red, Green and Blue Channels

Histograms and CDFs for each Channel

```python
# Plot histograms and CDFs for each channel
def plot_histogram_and_cdf(channel, title):
    hist, bins = np.histogram(channel.flatten(), 256, [0,
        256])
    cdf = hist.cumsum()  # cumulative sum
    cdf_normalized = cdf * hist.max() / cdf.max()  #
        normalize

    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.hist(channel.flatten(), bins=256, range=[0, 256],
        color='gray')
    plt.title(f'Histogram of {title}')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')

    plt.subplot(1, 2, 2)
    plt.plot(cdf_normalized, color='b')
    plt.hist(channel.flatten(), bins=256, range=[0, 256],
        color='r', alpha=0.5)
    plt.title(f'CDF of {title}')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
    plt.legend(('CDF', 'Histogram'), loc='upper left')
    plt.show()

# Plot for each channel
plot_histogram_and_cdf(r, 'Red Channel')
plot_histogram_and_cdf(g, 'Green Channel')
plot_histogram_and_cdf(b, 'Blue Channel')
```

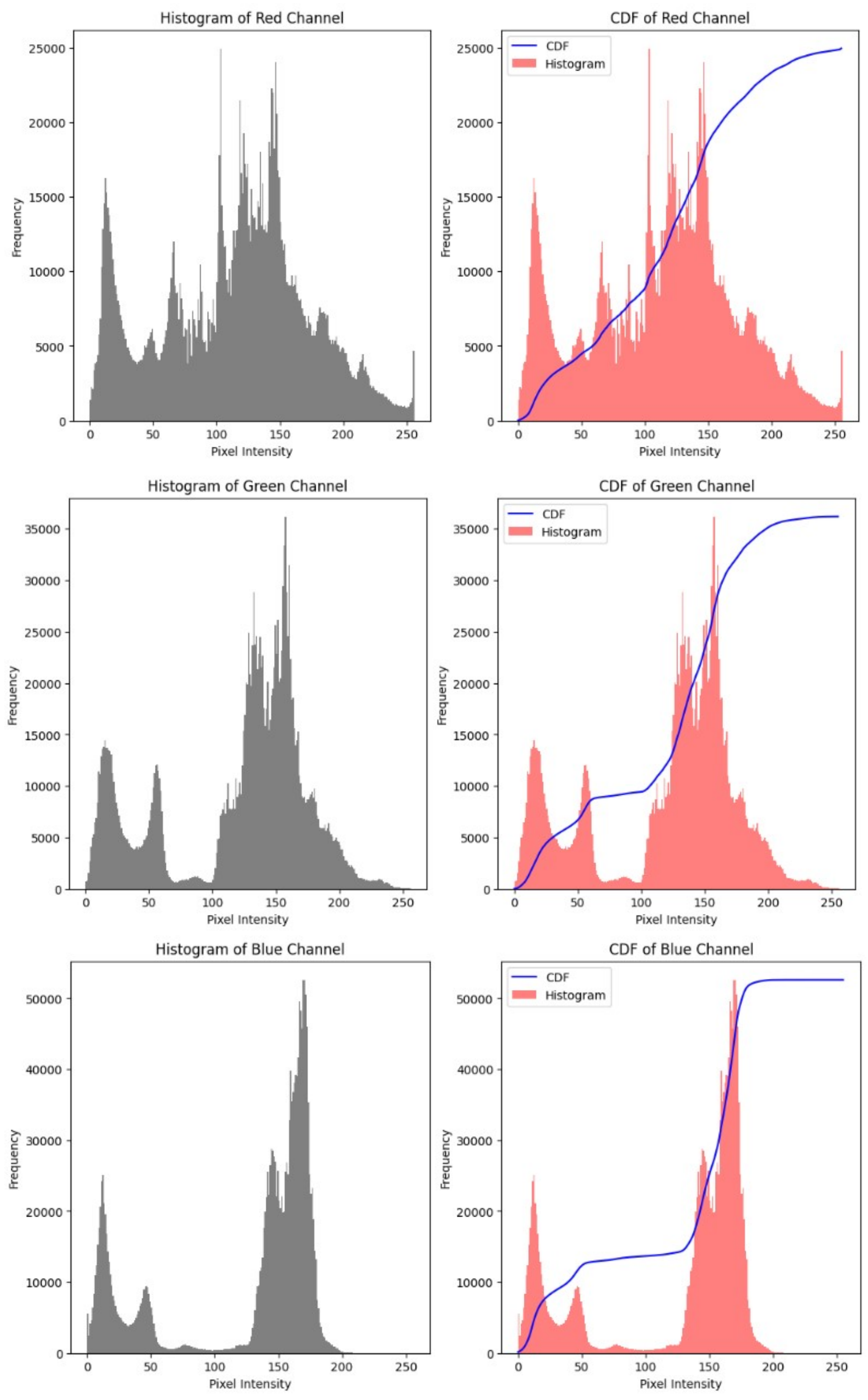

Figure 7: Histograms and CDFs for each channel

Apply histogram equalization to each channel

```
# Apply histogram equalization to each channel
equalized_r = cv2.equalizeHist(r)
equalized_g = cv2.equalizeHist(g)
equalized_b = cv2.equalizeHist(b)

# Display the equalized channels
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(equalized_r, cmap='Reds')
plt.title('Equalized Red Channel')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(equalized_g, cmap='Greens')
plt.title('Equalized Green Channel')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(equalized_b, cmap='Blues')
plt.title('Equalized Blue Channel')
plt.axis('off')
plt.show()
```

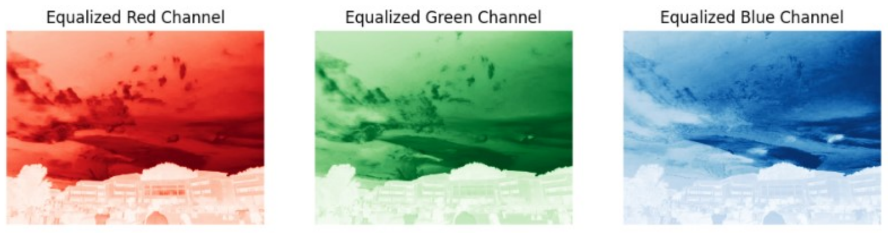

Figure 8: Equalized each channel

Histograms and CDFs for the equalized channels

```
# Plot histograms and CDFs for the equalized channels
plot_histogram_and_cdf(equalized_r, 'Equalized Red Channel')
plot_histogram_and_cdf(equalized_g, 'Equalized Green Channel
    ')
plot_histogram_and_cdf(equalized_b, 'Equalized Blue Channel'
    )
```

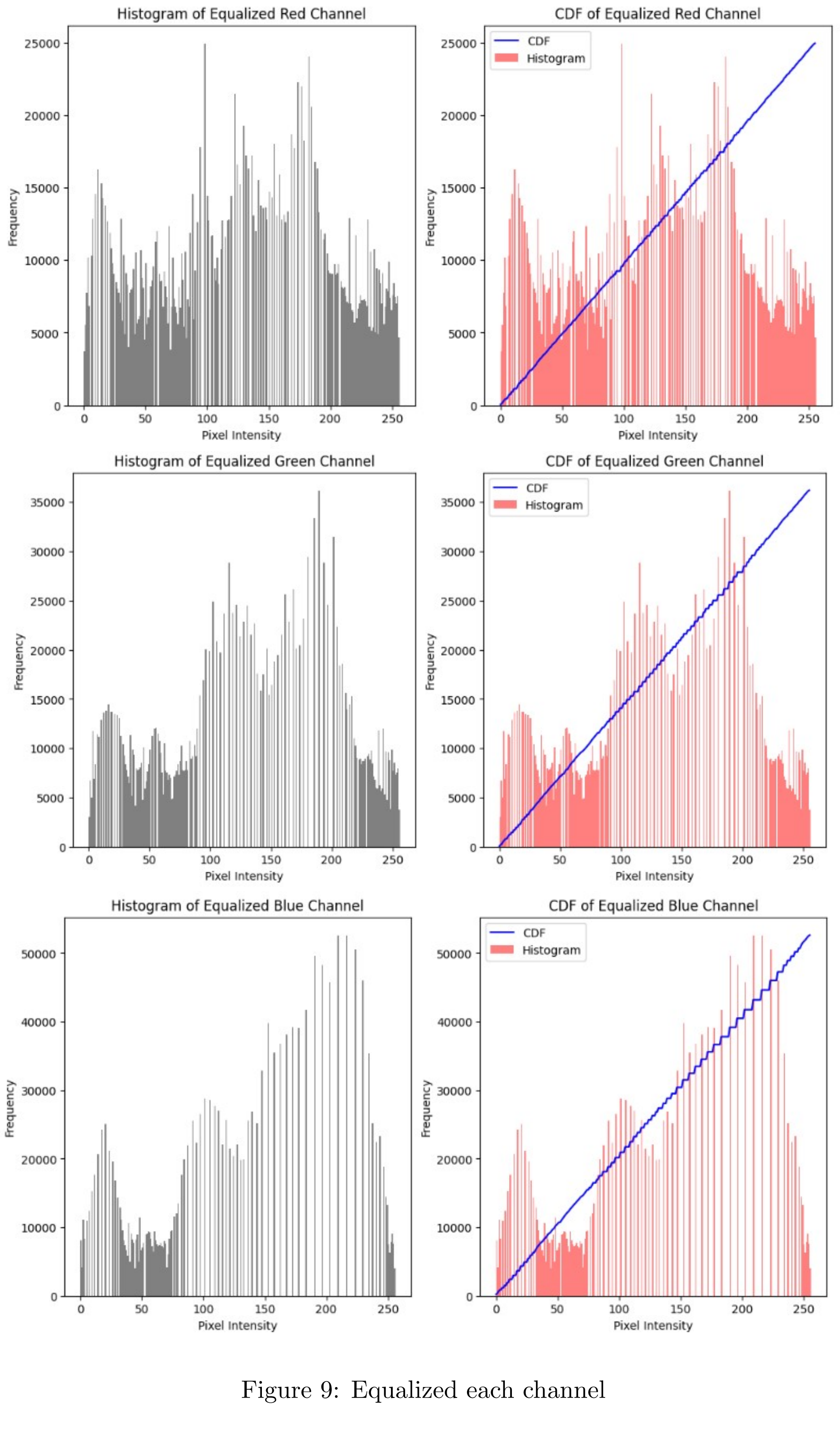

Figure 9: Equalized each channel

Merging Channels to get equalized image

```
# Merge the equalized channels back into an image
equalized_image2 = cv2.merge((equalized_r, equalized_g,
    equalized_b))

# Display the original and equalized images for comparison
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image2)
plt.title('Original Image 2')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(equalized_image2)
plt.title('Merged Equalized Image 2')
plt.axis('off')
plt.show()

# Save the results
cv2.imwrite('/kaggle/working/equalized_image2.jpg', cv2.
    cvtColor(equalized_image2, cv2.COLOR_RGB2BGR))
```

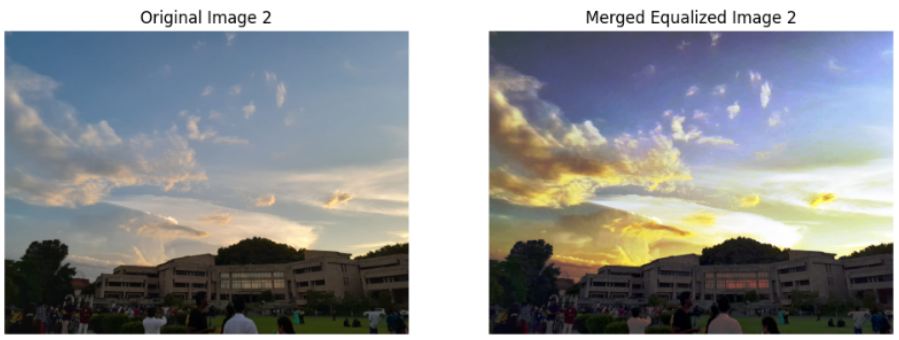

Figure 10: Comparing original and equalized image