College of Computer Studies

**Laboratory Exercise No. 3**

| Topic: | Histogram Equalization and Contrast Adjustment in Computer Vision | Week No. | 4 |
| --- | --- | --- | --- |
| Course Code: | CSST106 | Term: | 1st Semester |
| Course Title: | Perception and Computer Vision | Academic Year: | 2025-2026 |

**Lab Activity: Histogram Equalization and Contrast Adjustment in Computer Vision**

**Objective**

* Understand and implement histogram equalization to enhance image contrast.
* Perform global and adaptive histogram equalization to improve image quality.
* Apply contrast adjustment techniques and analyze their impact on image visualization.

**Background**

Histogram equalization is a technique used in image processing to improve contrast by redistributing pixel intensity values over the entire range. It enables better visibility of details in images with poor contrast. Contrast adjustment modifies the brightness and contrast of an image to highlight features for further computer vision tasks. This lab involves using OpenCV and Python for practical implementation.

**Materials Required**

* Computer with Python installed
* OpenCV, NumPy, and Matplotlib libraries installed
* Sample images with varying contrast conditions

**Procedure**

*Part 1: Histogram Equalization*

1. Load an image: Read a grayscale or color image.
2. Convert to Grayscale (if needed): Convert color images to grayscale.
3. Calculate Histogram: Compute the histogram of the input image.
4. Apply Histogram Equalization:
   1. Use built-in functions like cv2.equalizeHist() for grayscale images.
   2. For color images, apply equalization on the luminance channel in the YUV or HSV color space.
5. Display Results:
   1. Show the original and equalized images side by side.
   2. Plot histograms before and after equalization to observe changes.

*Part 2: Contrast Adjustment*

1. Linear Contrast Adjustment:
   1. Implement contrast stretching by scaling pixel values.
2. Adaptive Histogram Equalization (CLAHE):
   1. Apply CLAHE using OpenCV’s cv2.createCLAHE() function.
   2. Compare results with standard histogram equalization.
3. Visualize:
   1. Display the original image, globally equalized image, and CLAHE result.
   2. Plot histograms for each version.

*Part 3: Create a python code with regards to histogram equalization.*

*Part 4: Observations*

1. Compare the visibility of details and overall image contrast.
2. Discuss differences between standard histogram equalization and CLAHE.
3. Note any amplification of noise or artifacts.

***Code***

1. import cv2

2. import numpy as np

3. import matplotlib.pyplot as plt

4.

5. image = cv2.imread("aso.jpg")

6. gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

7.

8. equalized\_gray = cv2.equalizeHist(gray)

9.

10. yuv = cv2.cvtColor(image, cv2.COLOR\_BGR2YUV)

11. yuv[:, :, 0] = cv2.equalizeHist(yuv[:, :, 0])

12. equalized\_color = cv2.cvtColor(yuv, cv2.COLOR\_YUV2BGR)

13.

14. min\_val = np.min(gray)

15. max\_val = np.max(gray)

16. linear\_contrast = ((gray - min\_val) / (max\_val - min\_val) \* 255).astype(np.uint8)

17.

18. clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

19. clahe\_gray = clahe.apply(gray)

20.

21. def plot\_hist(image, title):

22.     plt.hist(image.ravel(), 256, [0, 256])

23.     plt.title(title)

24.     plt.xlabel("Pixel Intensity")

25.     plt.ylabel("Frequency")

26.

27. plt.figure(figsize=(15, 12))

28.

29. # Original grayscale

30. plt.subplot(3, 3, 1), plt.imshow(gray, cmap="gray"), plt.title("Original Grayscale")

31. plt.subplot(3, 3, 2), plt.imshow(equalized\_gray, cmap="gray"), plt.title("Equalized (Gray)")

32. plt.subplot(3, 3, 3), plt.imshow(equalized\_color[:, :, ::-1]), plt.title("Equalized (Color - YUV)")

33.

34. # Contrast adjustments

35. plt.subplot(3, 3, 4), plt.imshow(linear\_contrast, cmap="gray"), plt.title("Linear Contrast Stretching")

36. plt.subplot(3, 3, 5), plt.imshow(clahe\_gray, cmap="gray"), plt.title("CLAHE (Adaptive Equalization)")

37.

38. # Histograms

39. plt.subplot(3, 3, 6), plot\_hist(gray, "Original Histogram")

40. plt.subplot(3, 3, 7), plot\_hist(equalized\_gray, "Equalized Histogram")

41. plt.subplot(3, 3, 8), plot\_hist(clahe\_gray, "CLAHE Histogram")

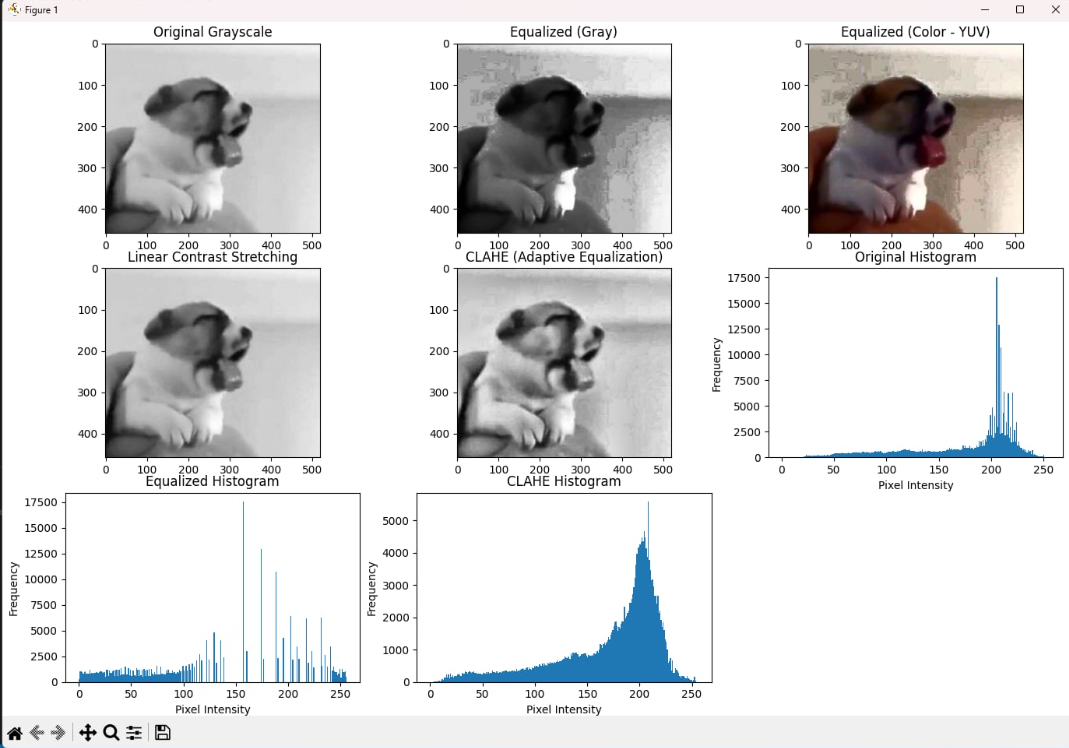
42.

43. plt.tight\_layout()

44. plt.show()

45.

***Output***



Question to Answer:

**1. What is histogram equalization?**  
Histogram equalization is a technique in image processing used to improve the contrast of an image. It works by redistributing the pixel intensity values so that they spread more evenly across the entire range. This makes details in dark or bright areas more visible without altering the overall content of the image.

**2. Why do we need histogram equalization?**  
We need histogram equalization because many images suffer from poor contrast due to lighting conditions or sensor limitations. By enhancing contrast, it allows hidden details to become clearer and makes the image easier to analyze. This is especially useful in medical imaging, satellite images, and other fields where clarity is important.

**3. How does histogram equalization work?**  
The process calculates the histogram of the image, which shows how pixel intensities are distributed. Then it redistributes these values so that they cover the full range of possible intensities, making the image appear clearer. In short, it stretches out areas where pixel values were clustered and balances brightness across the image.

**4. How does CLAHE work?**  
CLAHE (Contrast Limited Adaptive Histogram Equalization) improves on standard histogram equalization by applying it locally to small regions of the image instead of the whole image at once. It also limits the amplification of contrast to prevent noise from becoming too strong. This makes CLAHE especially good for enhancing details without overexposing or creating artifacts.

**5. What is the possible real-world application for histogram equalization?**  
Histogram equalization is widely used in fields that rely on image clarity. For example, it helps radiologists interpret X-rays more easily, improves the visibility of satellite or aerial images, and enhances photos taken in poor lighting. In everyday life, it is also used in smartphone cameras to automatically improve picture quality.

Conclusion:

In conclusion, histogram equalization and CLAHE are powerful techniques for enhancing image contrast, making hidden details clearer and images more useful for analysis. By redistributing pixel intensities, they improve visibility in poor lighting or low-contrast conditions, with CLAHE offering finer control to avoid noise amplification. These methods have practical applications in fields like medical imaging, remote sensing, and photography, where clarity is essential.