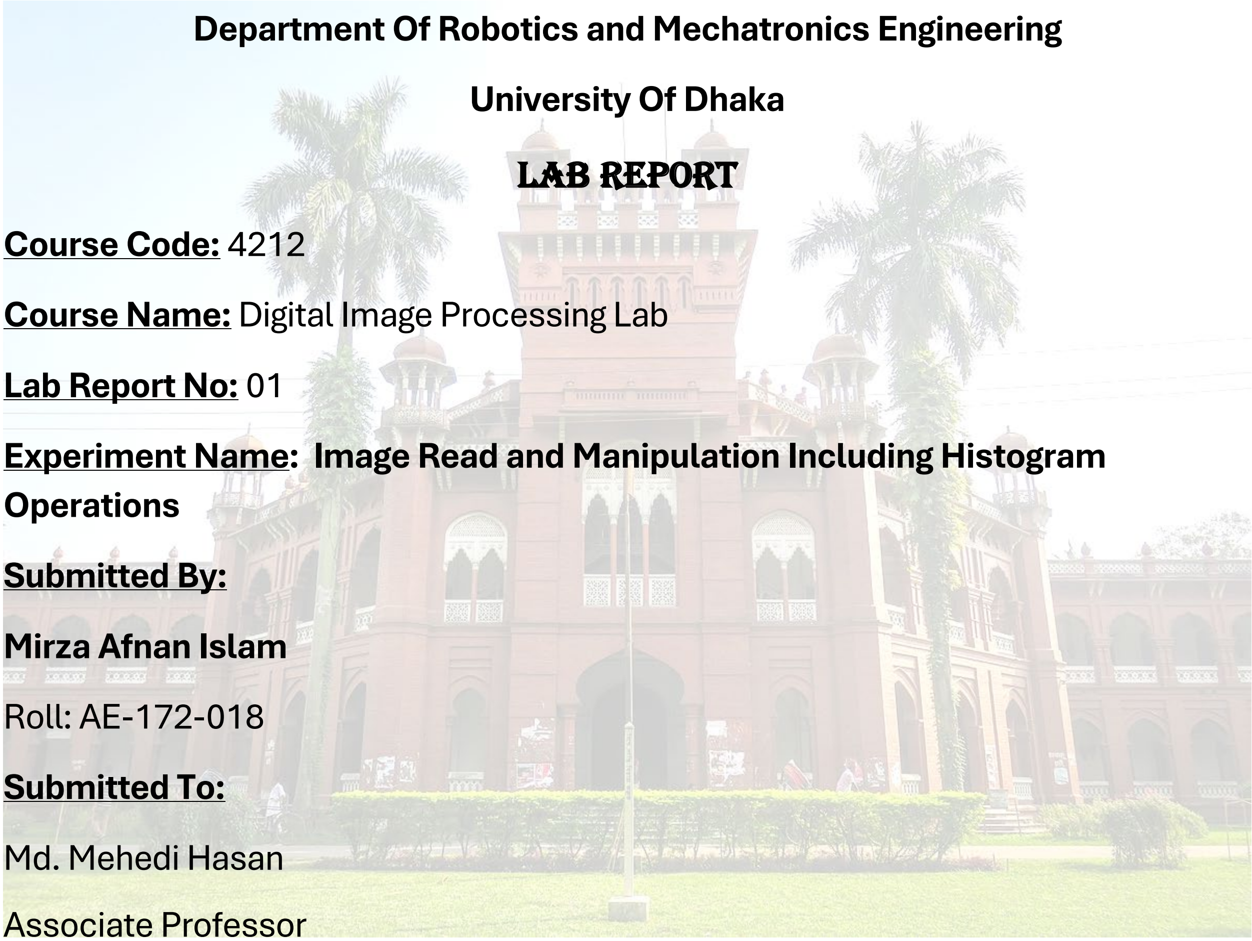


Department Of Robotics and Mechatronics Engineering

University Of Dhaka

LAB REPORT

Course Code: 4212

Course Name: Digital Image Processing Lab

Lab Report No: 01

Experiment Name: Image Read and Manipulation Including Histogram Operations

Submitted By:

Mirza Afnan Islam

Roll: AE-172-018

Submitted To:

Md. Mehedi Hasan

Associate Professor

Department of Robotics and Mechatronics Engineering

Faculty of Engineering

University Of Dhaka

Introduction:

Digital Image Processing is defined as a methodology of operation by which we can convert, modify and change images according to the requirements. Incase of digital image processing, an image is called as a two dimensional function f(x,y) where x and y both are spatial variables. By following two-dimensional array or matrix operations; an image's sharpness, boldness, contrast, resolution, size everything can be changed. Today, we have learnt some beginner level operations. The name of operations is demonstrated below:

1. Image Read
2. Image Conversion [BGR2GRAY]
3. Image Histogram Conversion
4. Image Histogram Equalization
5. Equalized Image Gray Scale Conversion
6. Contrast Limited Equalized Adaptive Histogram
7. Threshold and Adaptive Threshold
8. OTSU Binarization

Image:

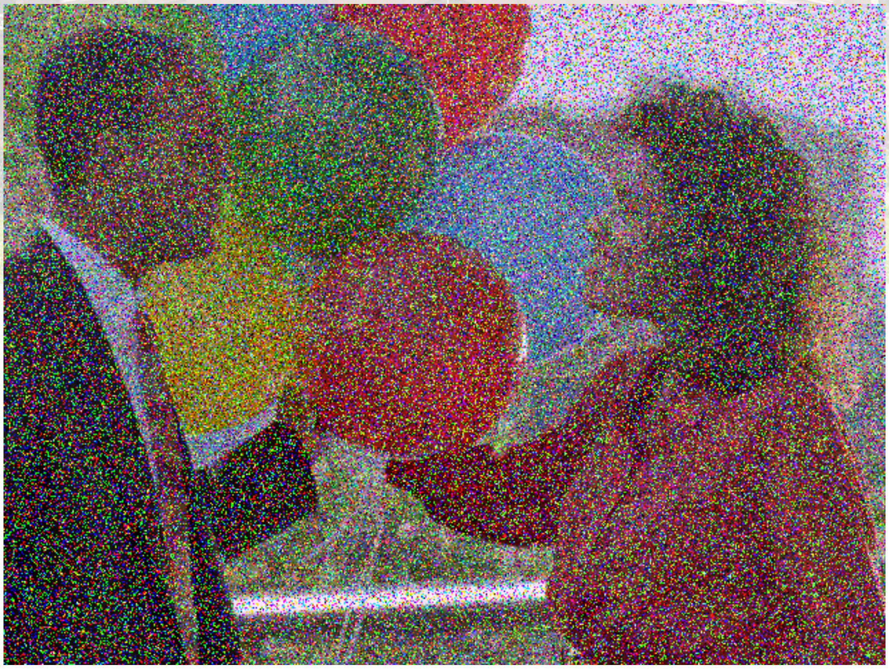


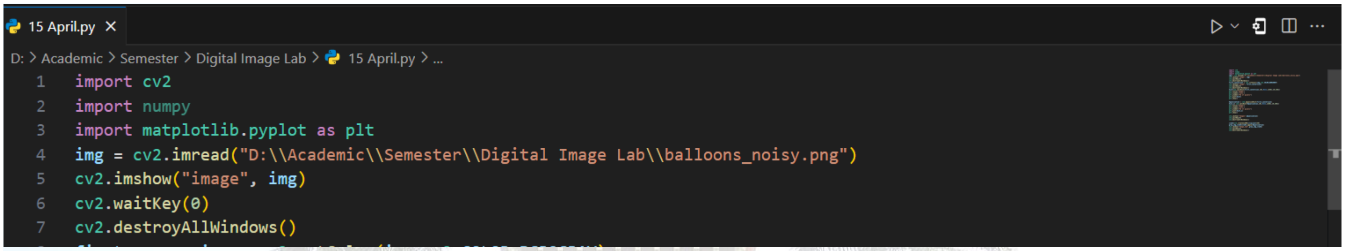
Image Path:

"D:\\Academic\\Semester\\DigitalImageLab\\balloons_noisy.png”

Workflow:

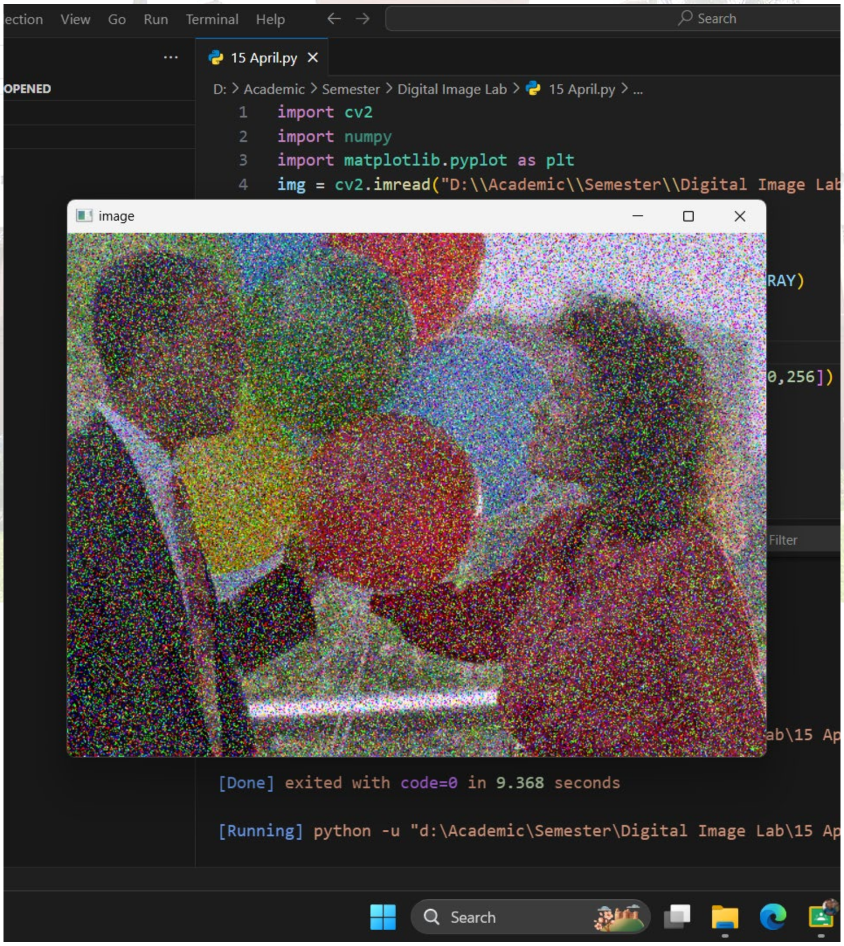
Operation – 01: [Image Read]

Initially we read image by using cv2 function called imread where we made input the path of our image.


15 April.py
D: > Academic > Semester > Digital Image Lab > 15 April.py > ...

```
import cv2
import numpy
import matplotlib.pyplot as plt
img = cv2.imread("D:\\Academic\\Semester\\Digital Image Lab\\balloons_noisy.png")
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:



```
import cv2
import numpy
import matplotlib.pyplot as plt
img = cv2.imread("D:\\Academic\\Semester\\Digital Image Lab
```

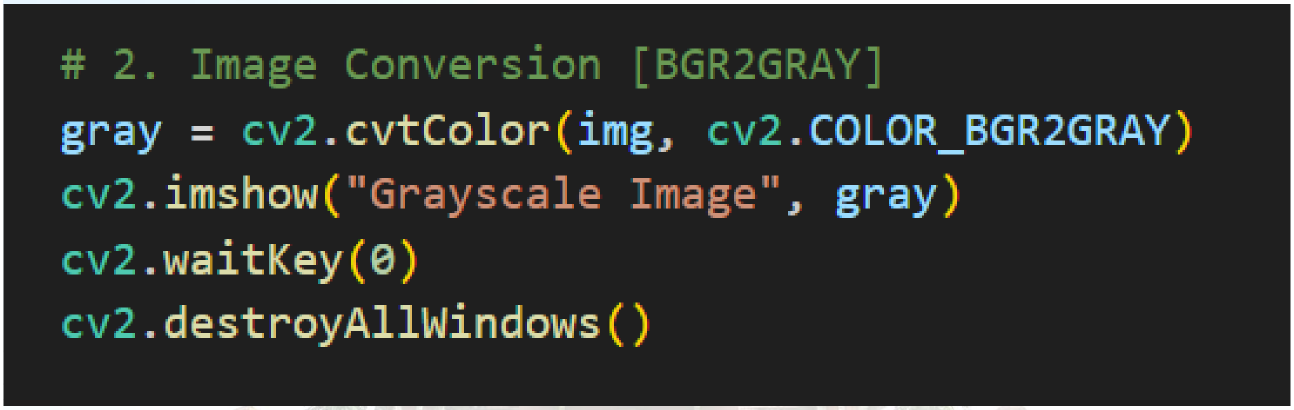
```
RAY)
0,256])
ab\15 Ap

[Done] exited with code=0 in 9.368 seconds

[Running] python -u "d:\Academic\Semester\Digital Image Lab\15 Ap
```

Operation – 02: [Image Conversion [BGR2GRAY]]

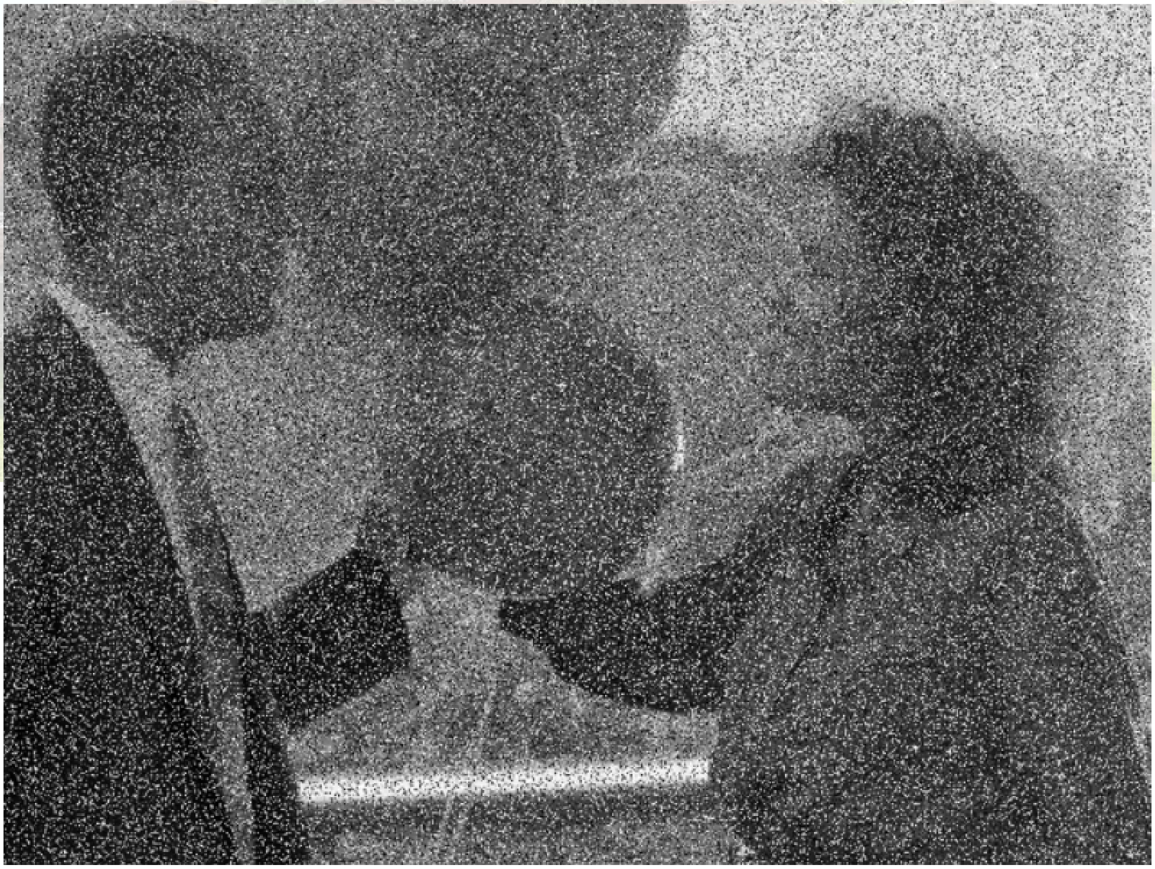
This function converts a BGR image to Grayscale by following NTSC formula:

0.299 · Red + 0.587 · Green + 0.114 · Blue.

This formula closely represents the average person's relative perception of the brightness of red, green, and blue light

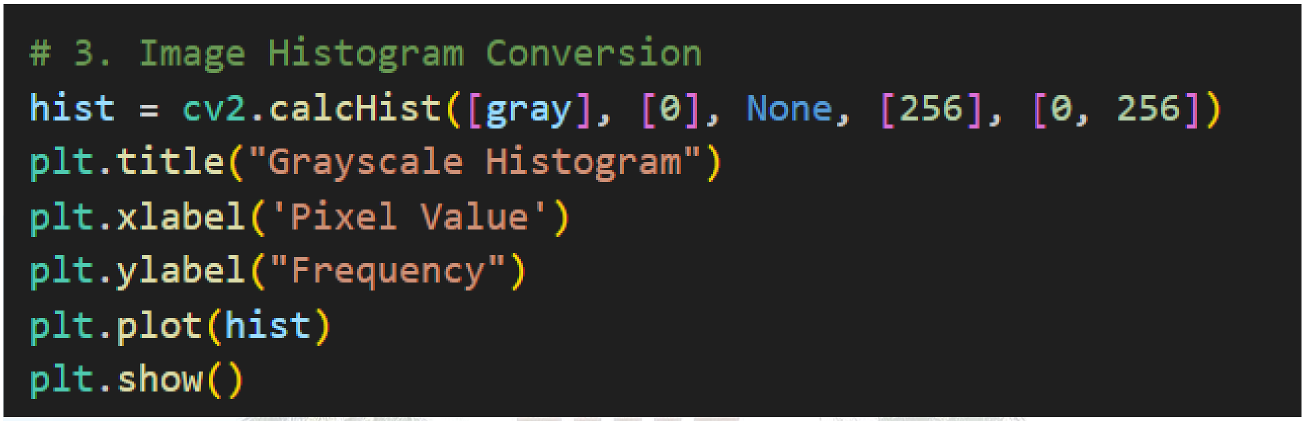


```
# 2. Image Conversion [BGR2GRAY]
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Grayscale Image", gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

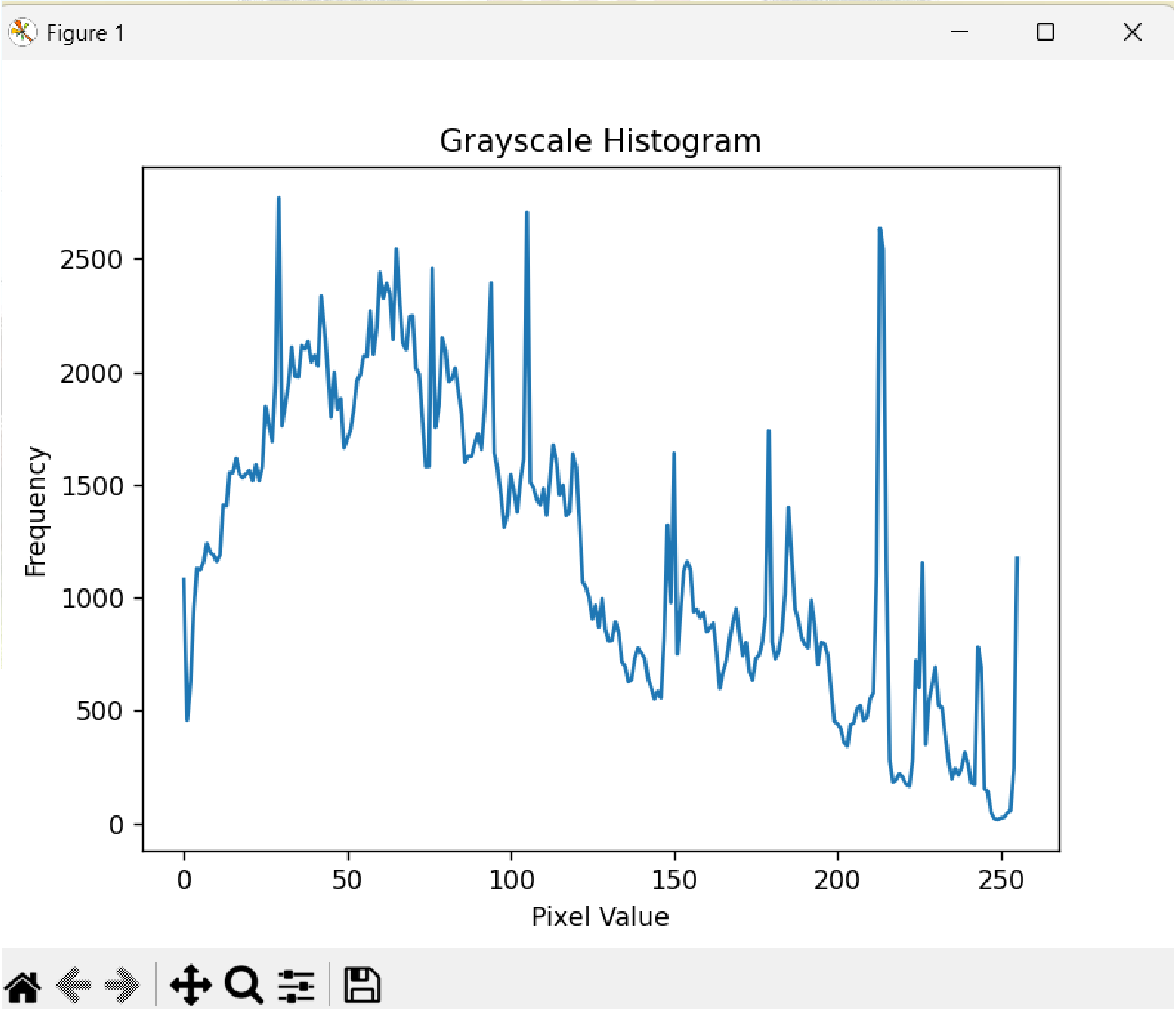


Operation 03: [Image Histogram Conversion]

An image histogram is a gray-scale value distribution showing the frequency of occurrence of each gray-level value.

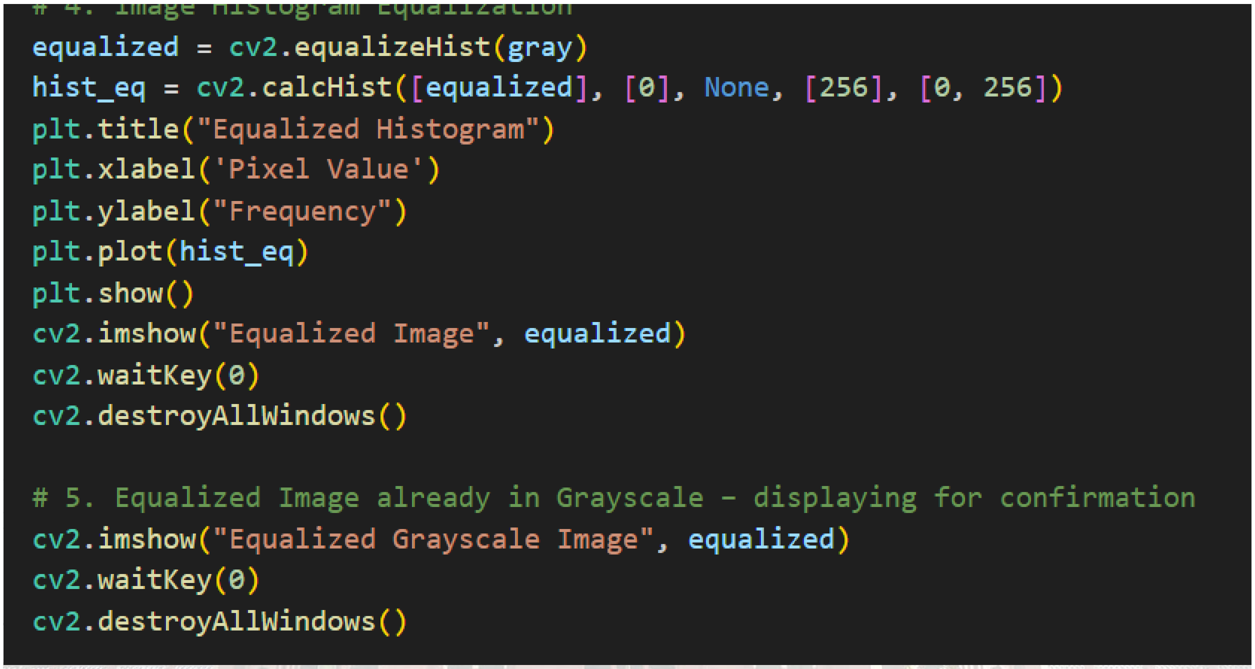


```
# 3. Image Histogram Conversion
hist = cv2.calcHist([gray], [0], None, [256], [0, 256])
plt.title("Grayscale Histogram")
plt.xlabel('Pixel Value')
plt.ylabel("Frequency")
plt.plot(hist)
plt.show()
```

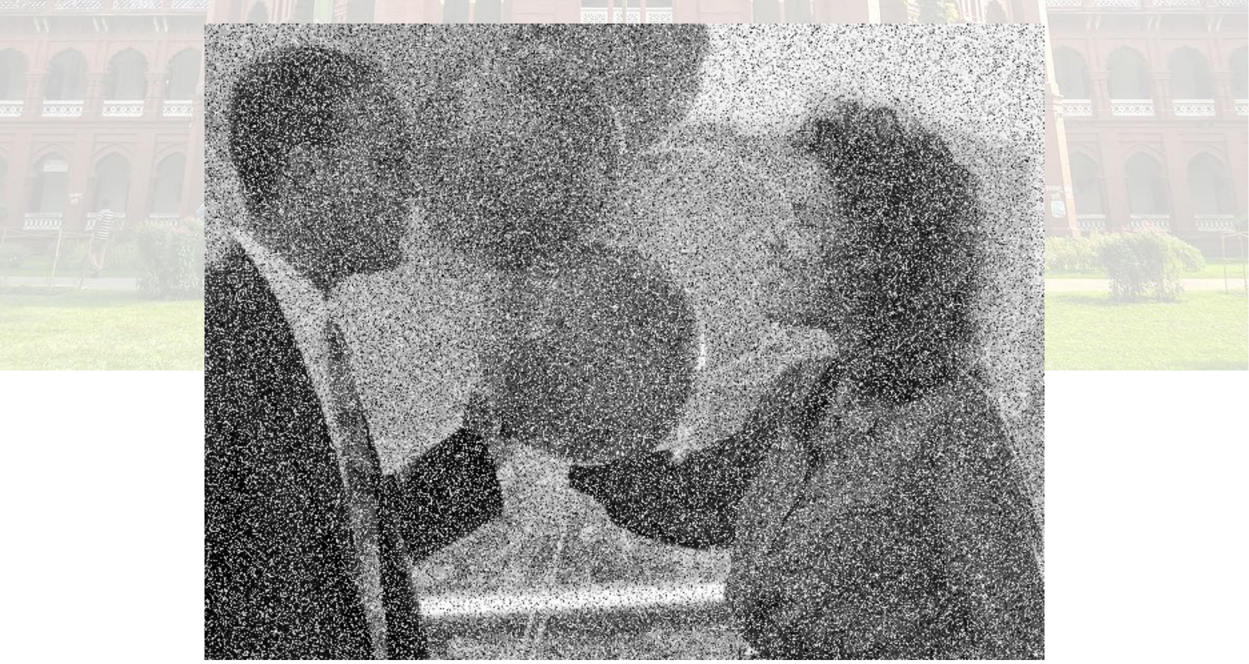


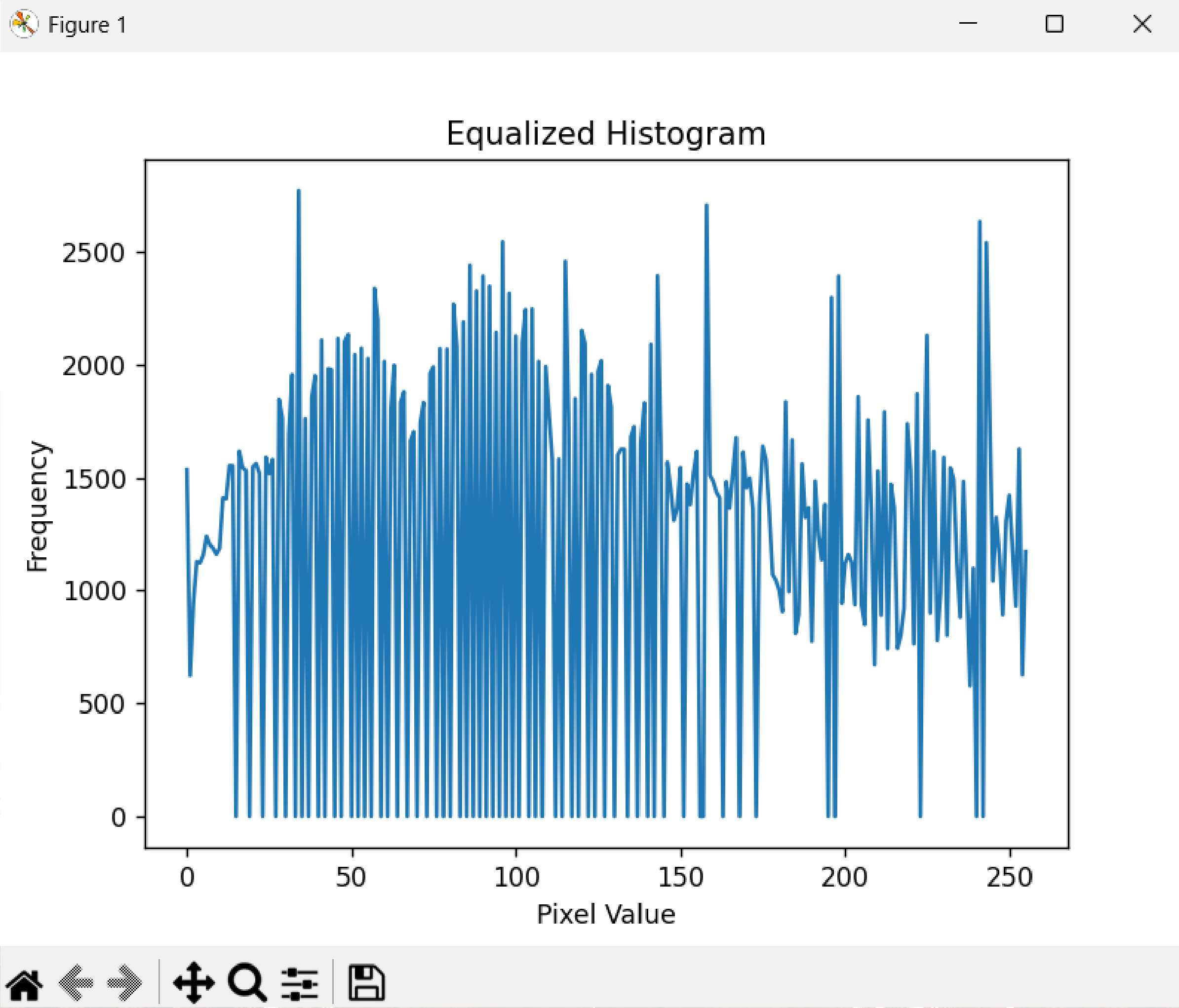
Operation 04: [Image Histogram Equalization and display grayscale]

Histogram Equalization is an image processing technique that adjusts the contrast of an image by using its histogram.

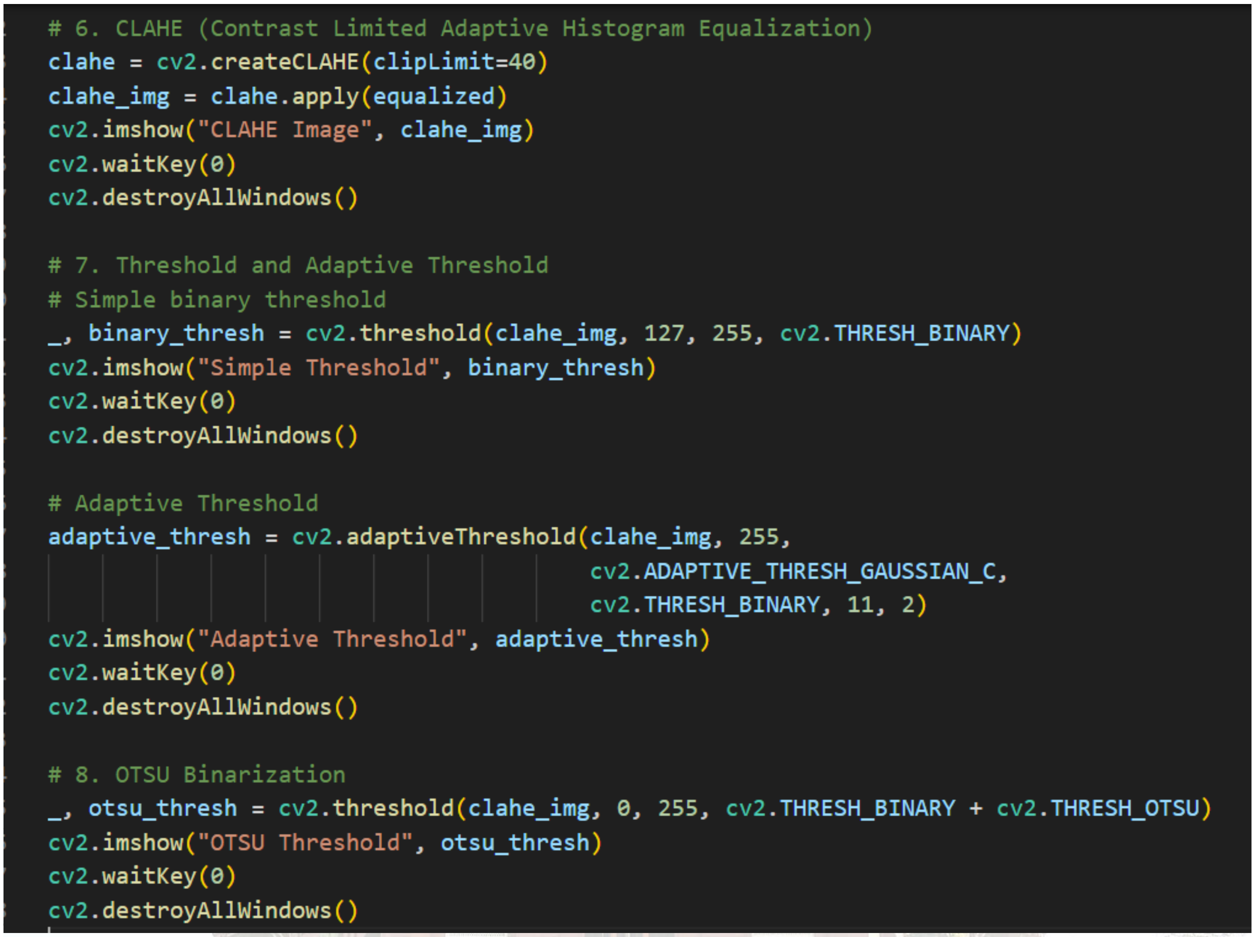


```
# 4. Image Histogram Equalization
equalized = cv2.equalizeHist(gray)
hist_eq = cv2.calcHist([equalized], [0], None, [256], [0, 256])
plt.title("Equalized Histogram")
plt.xlabel('Pixel Value')
plt.ylabel("Frequency")
plt.plot(hist_eq)
plt.show()
cv2.imshow("Equalized Image", equalized)
cv2.waitKey(0)
cv2.destroyAllWindows()

# 5. Equalized Image already in Grayscale - displaying for confirmation
cv2.imshow("Equalized Grayscale Image", equalized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

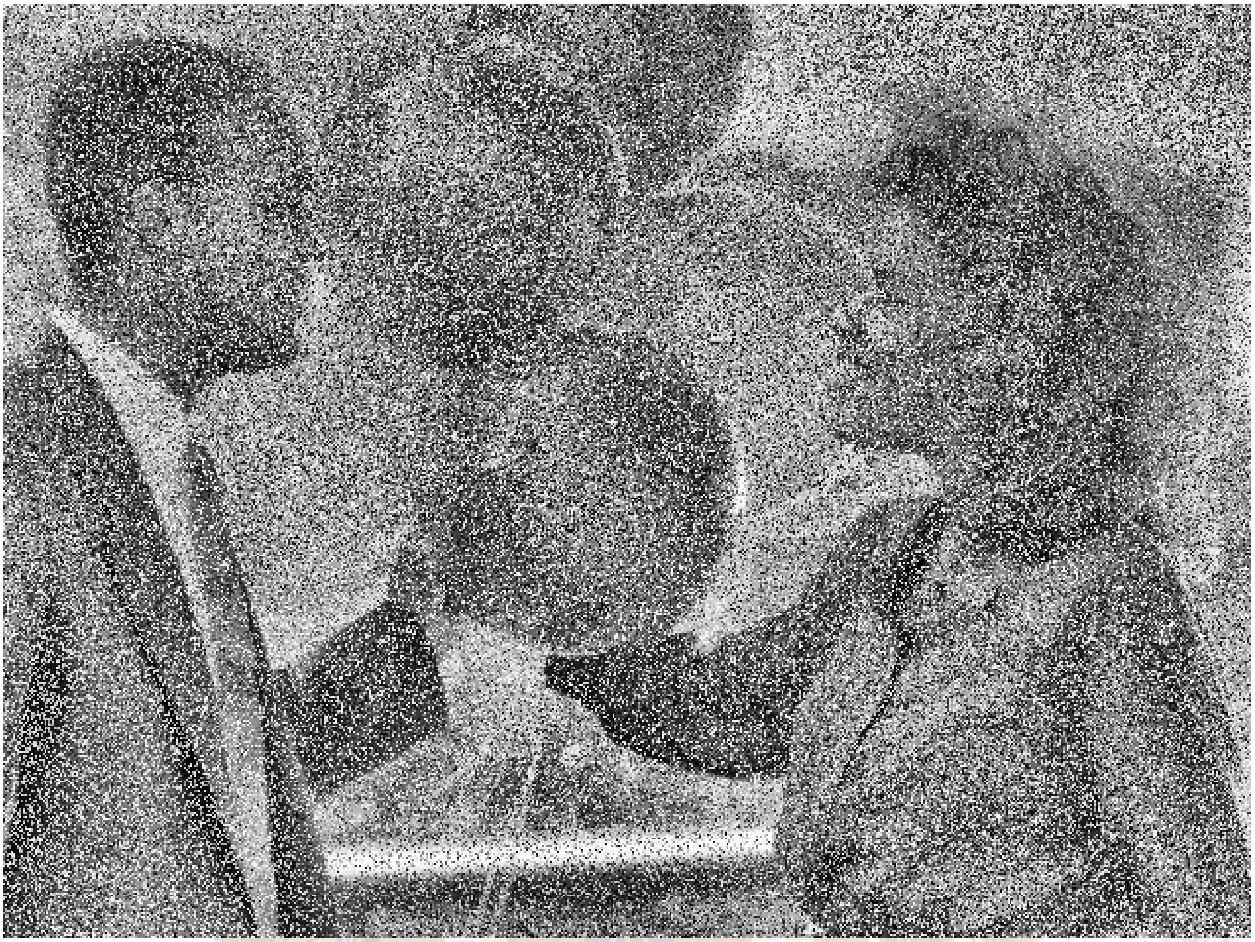




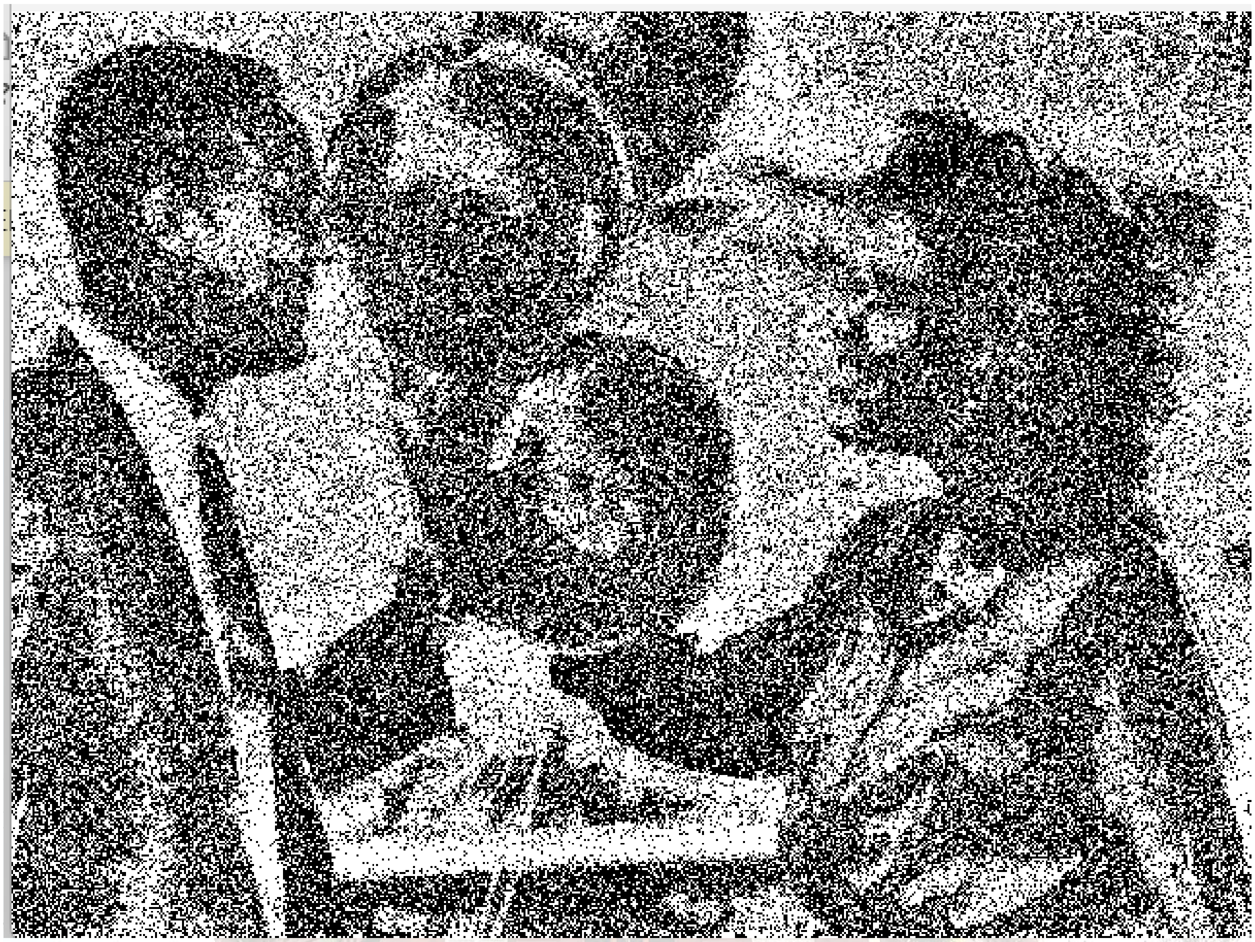
Operation-05: **Contrast Limited Equalized Adaptive Histogram, Threshold and Adaptive Threshold and OTSU Binarization**



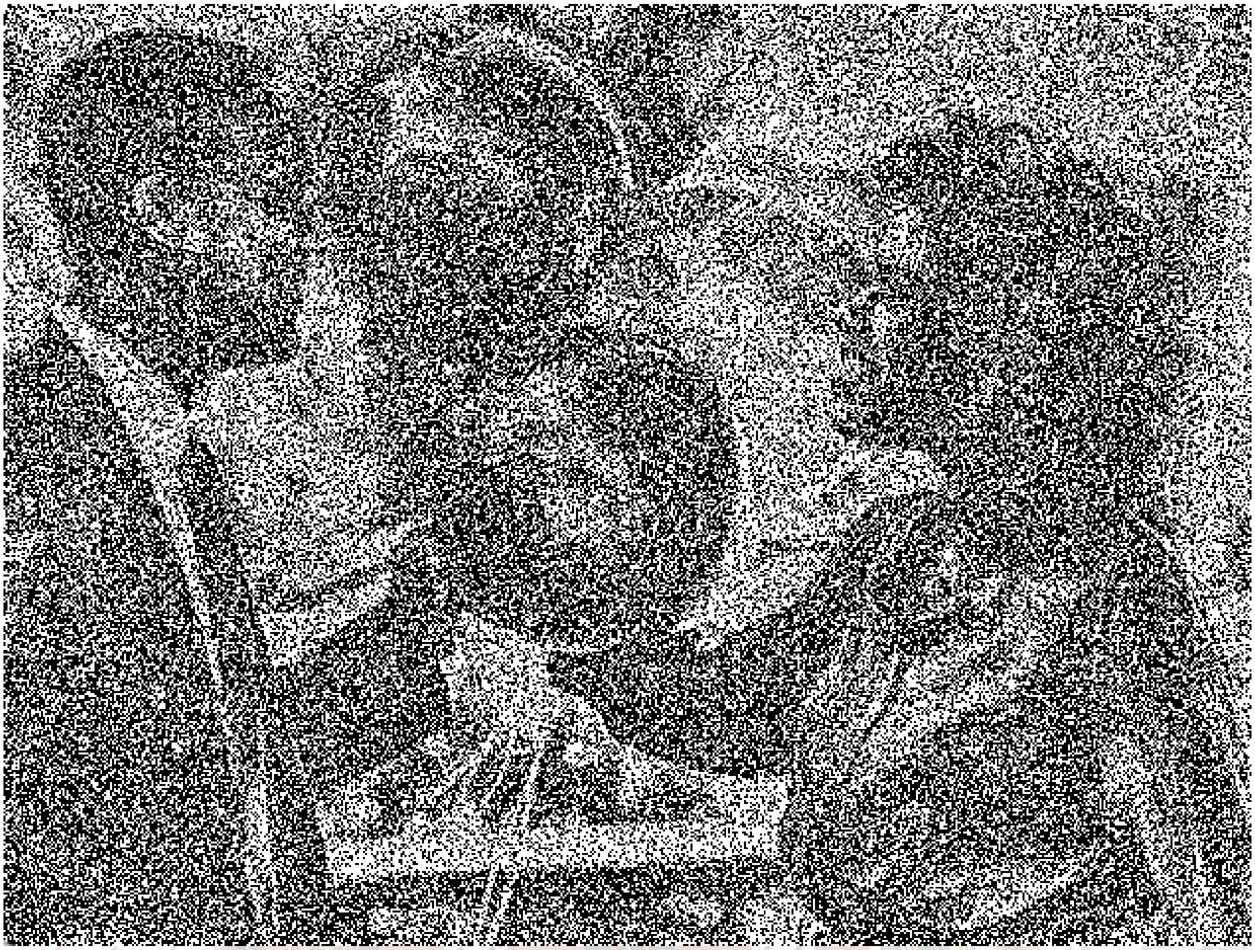
```python
# 6. CLAHE (Contrast Limited Adaptive Histogram Equalization)
clahe = cv2.createCLAHE(clipLimit=40)
clahe_img = clahe.apply(equalized)
cv2.imshow("CLAHE Image", clahe_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# 7. Threshold and Adaptive Threshold
# Simple binary threshold
_, binary_thresh = cv2.threshold(clahe_img, 127, 255, cv2.THRESH_BINARY)
cv2.imshow("Simple Threshold", binary_thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Adaptive Threshold
adaptive_thresh = cv2.adaptiveThreshold(clahe_img, 255,
                                        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                        cv2.THRESH_BINARY, 11, 2)
cv2.imshow("Adaptive Threshold", adaptive_thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()

# 8. OTSU Binarization
_, otsu_thresh = cv2.threshold(clahe_img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
cv2.imshow("OTSU Threshold", otsu_thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



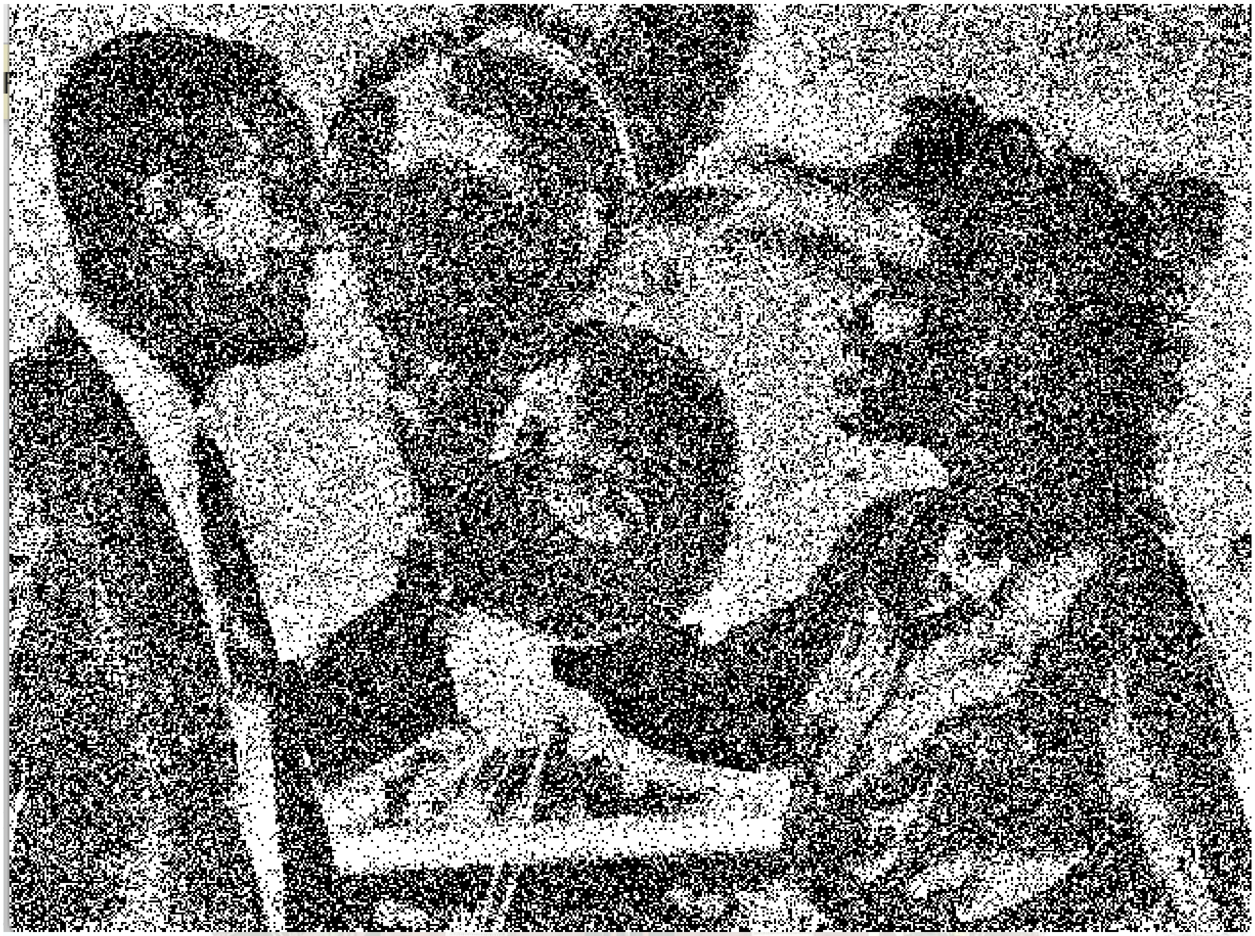
Contrast Limited Equalized Adaptive Histogram



Threshold(Binary)



(Adaptive Threshold)



(OTSU Binarization)