E/19/129
K.H. Gunawardana

CO543 - Image Processing
Lab 02

Lab Task 01

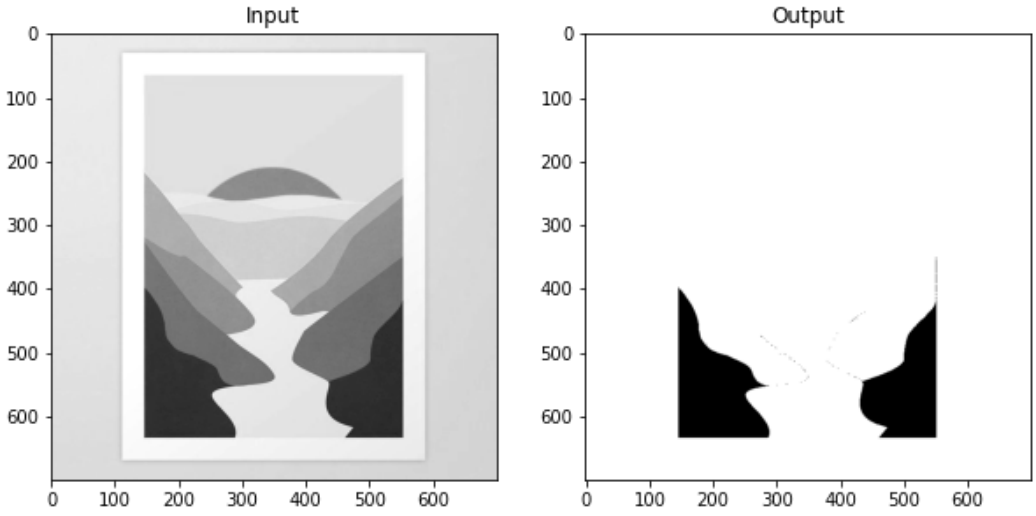

Figure 01

Lab Task 02

Inputs:

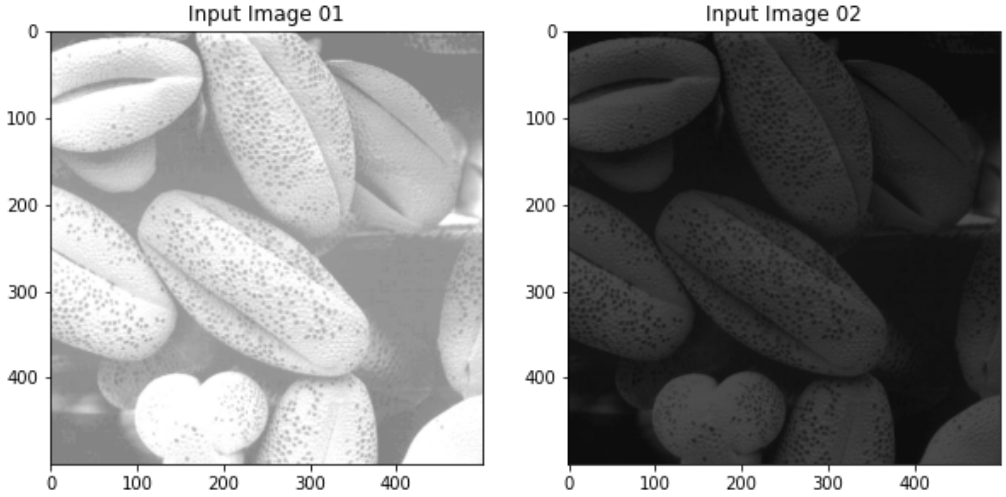

Figure 02

Addition:

- Using in-build function add()
  Output:

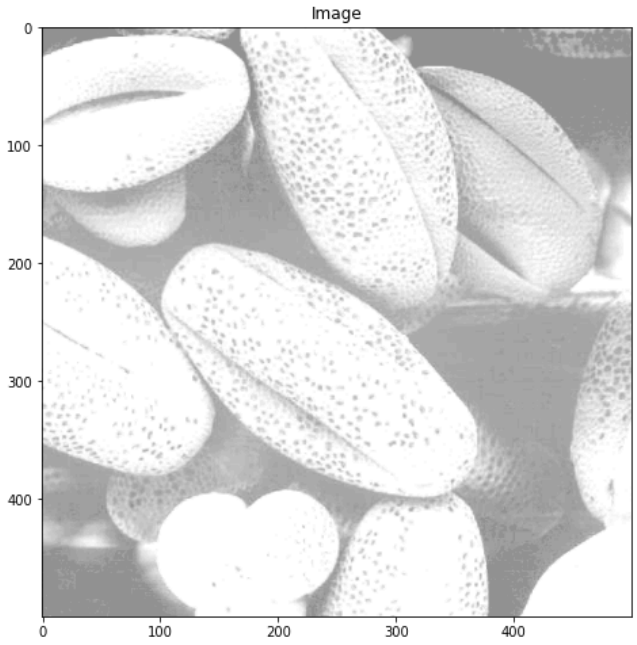

Figure 03

- Using res = img1 + img2
  Output:

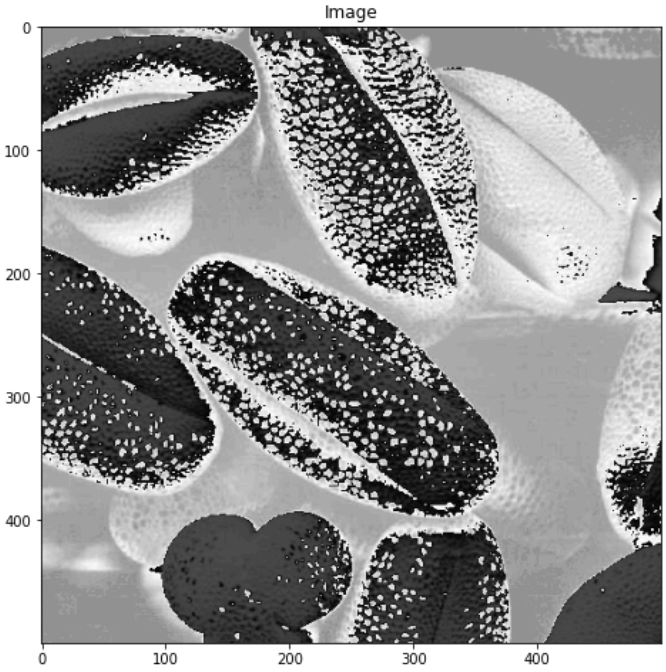

Figure 04

Subtraction:

- Using in-build function subtract()
  Output:

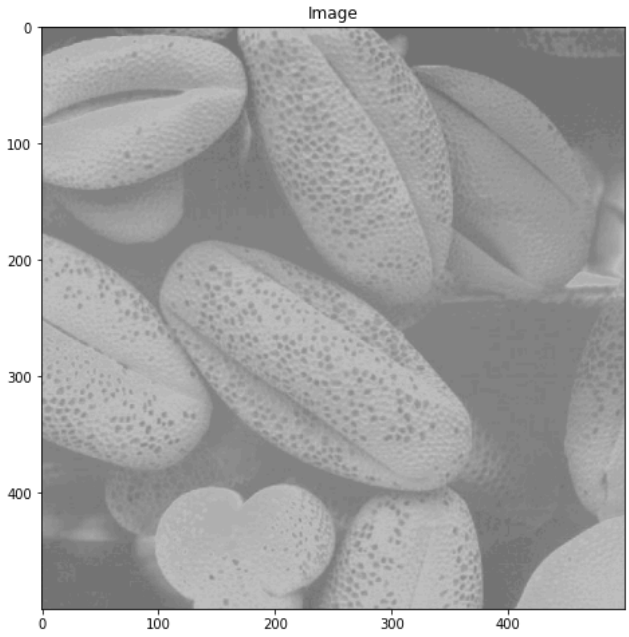

Figure 05

- Using res = img1 - img2
  Output:

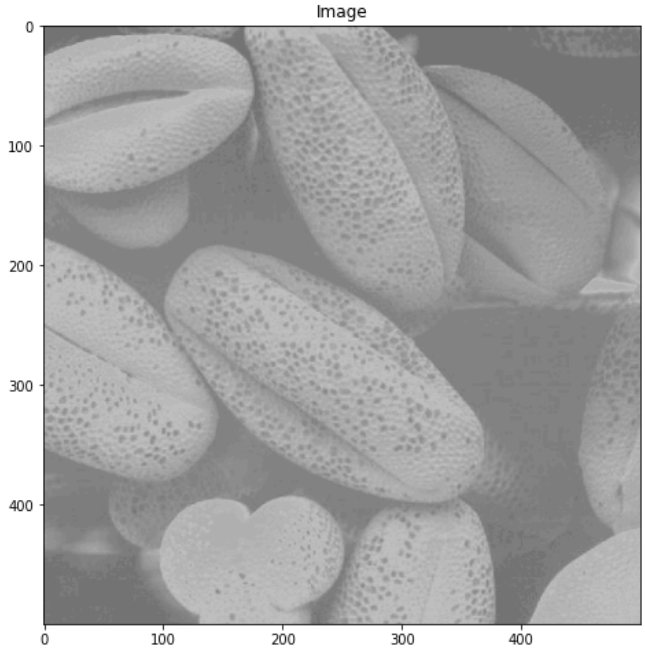

Figure 06

Lab Task 03

1. Write simple programs to demonstrate the following. Show the original and resultant images in the same figure to compare them easily.

   a.

   Output:

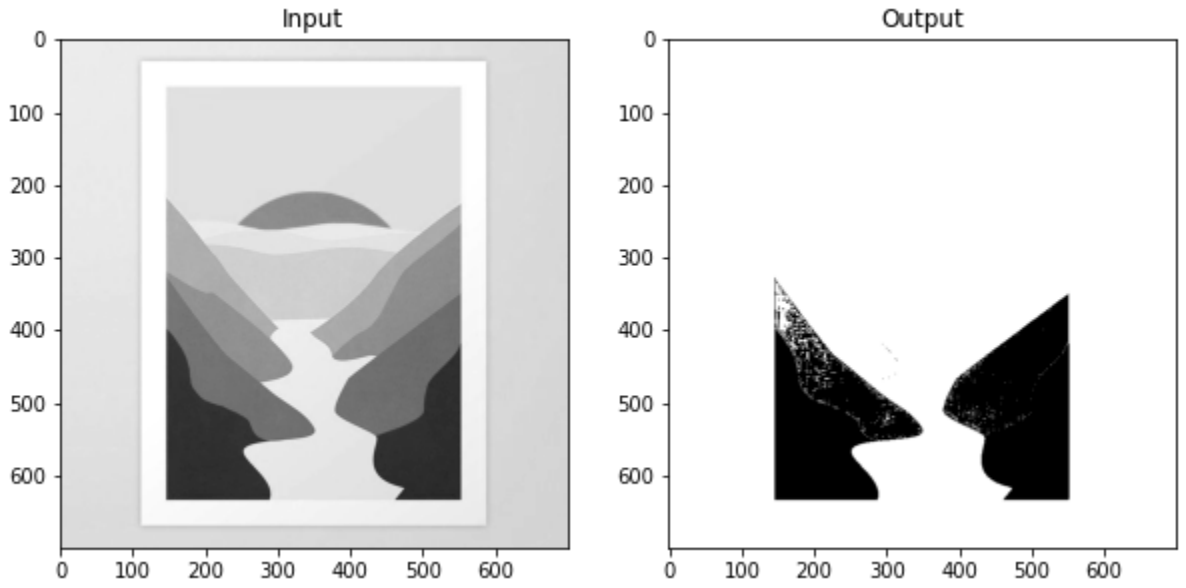

Figure 07

   b.

   Output:

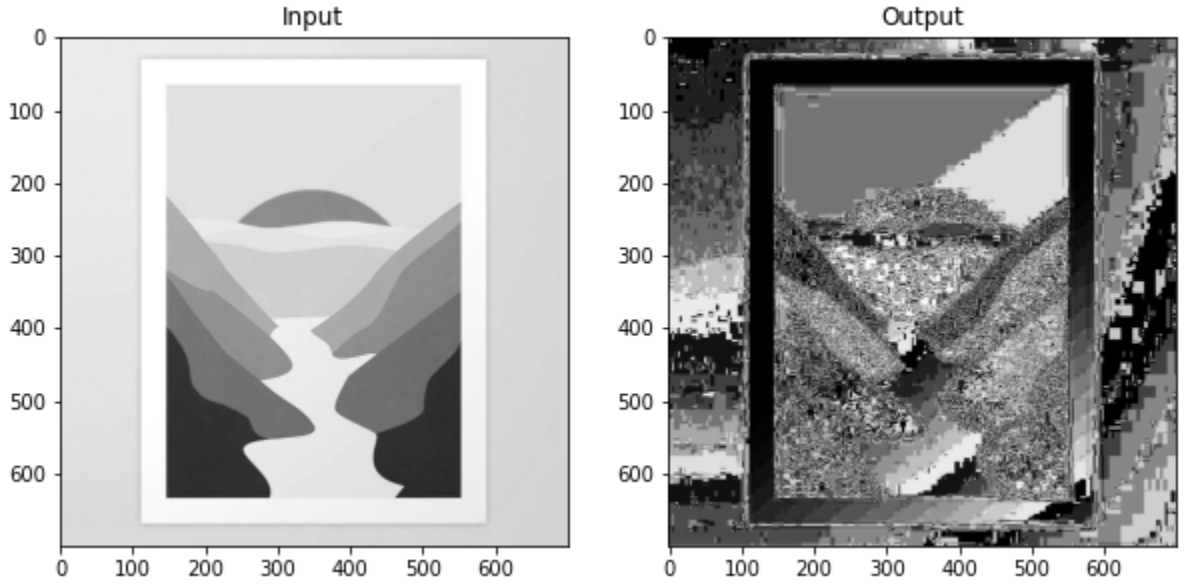

Figure 08

c.

Output:

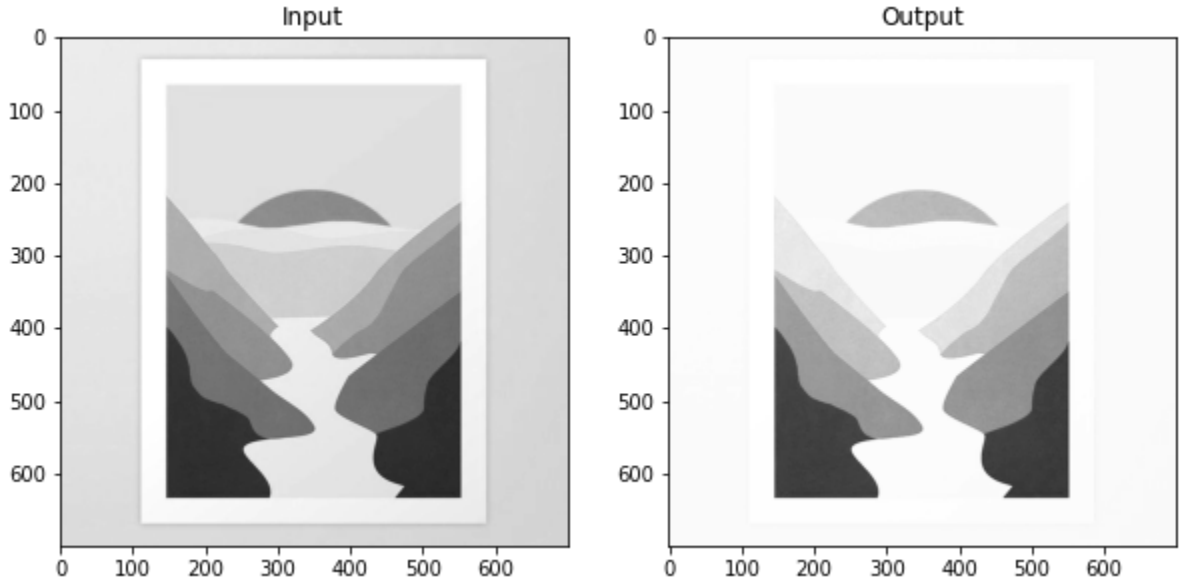

Figure 09

d.

Output:

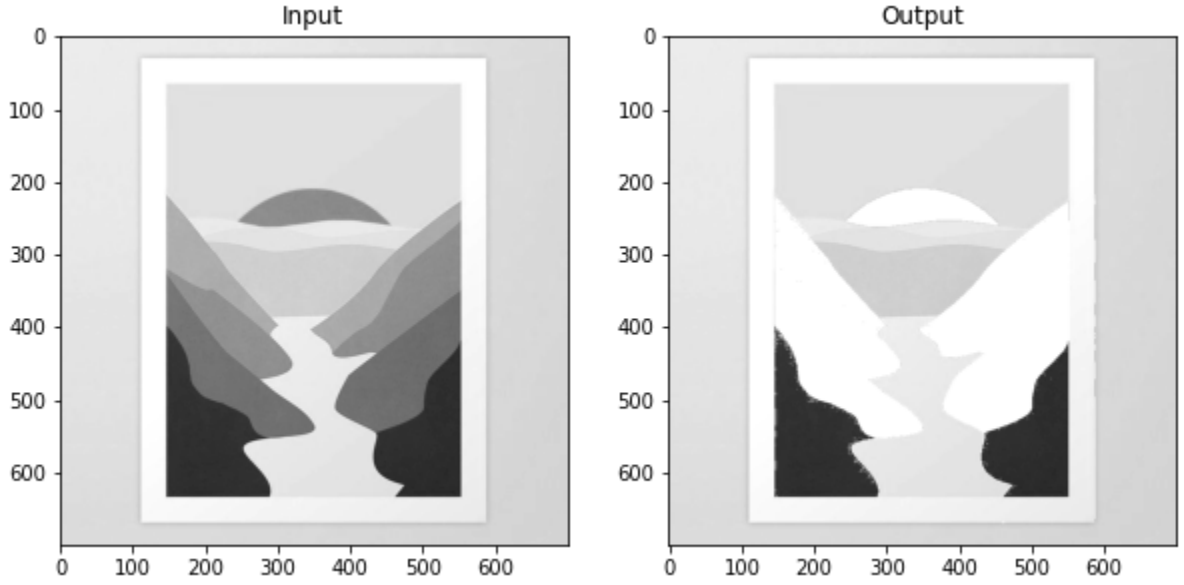

Figure 10

e.

Output:

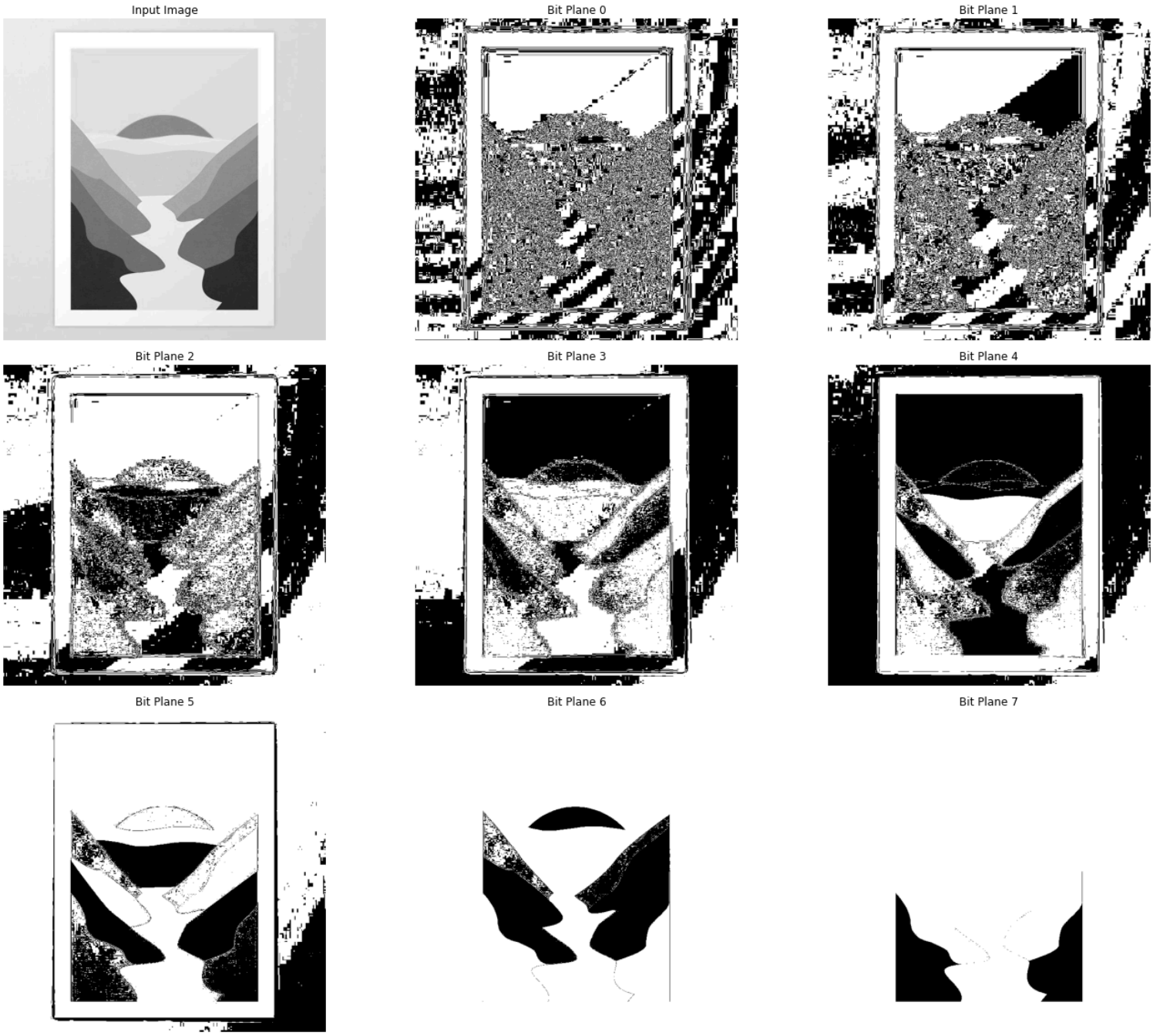

Figure 11

2. Consider the graph for a typical transformation function used for Contrast Stretching in the given figure and determine the behaviour of the function with respect to given changes.

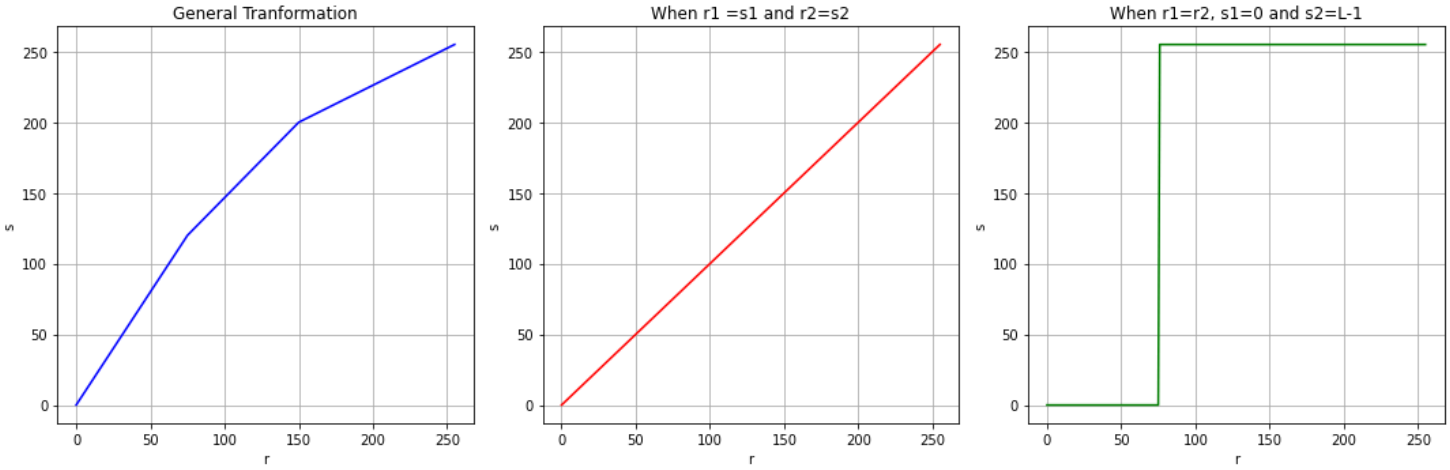

Figure 12

Lab Task 04

Write a program to read any image, and resize it to 256x256. Apply the masks shown in the following figures so that only the middle part of the image is visible.

Input Image:

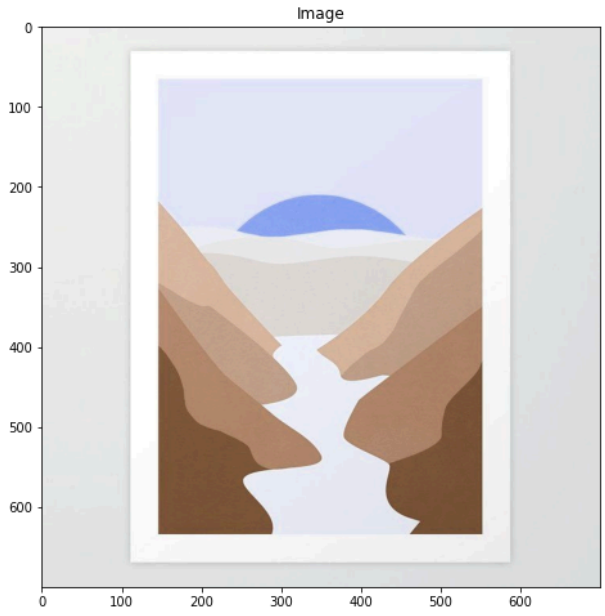

Figure 13

Resized Image:

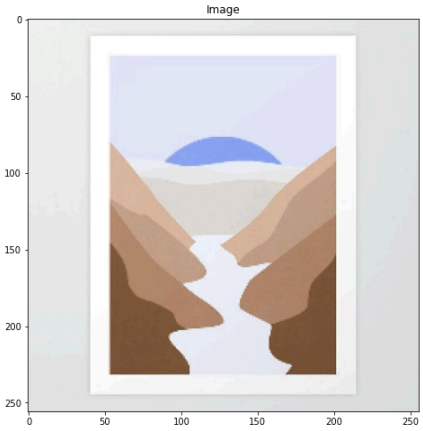

Figure 14

Square Mask:

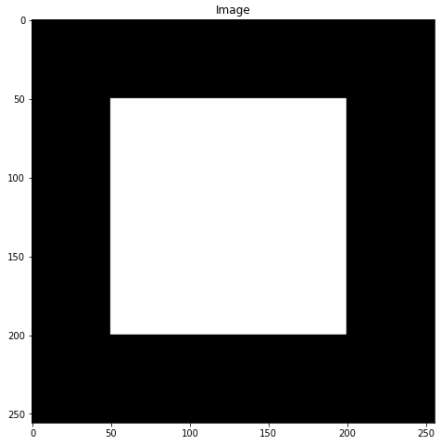

Figure 15

Output:

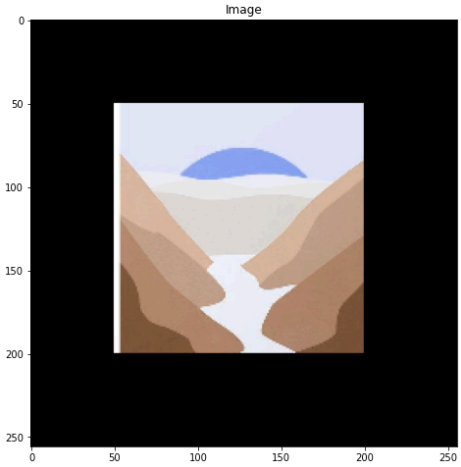

Figure 16

Circular Mask:

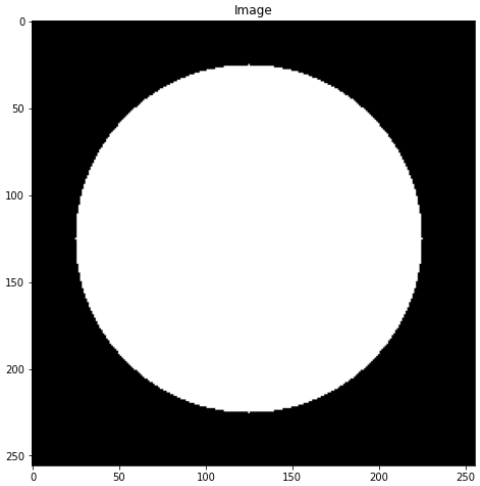

Figure 17

Output:

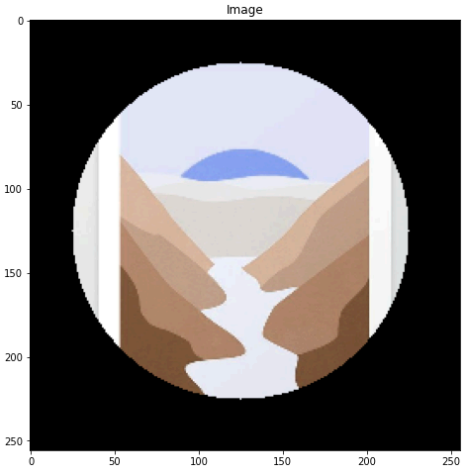

Figure 18

Lab Task 05

Write your own Python OpenCV function addbrightness() and use it to increase the brightness of a given image. (Hint: Use Image arithmetic operations)

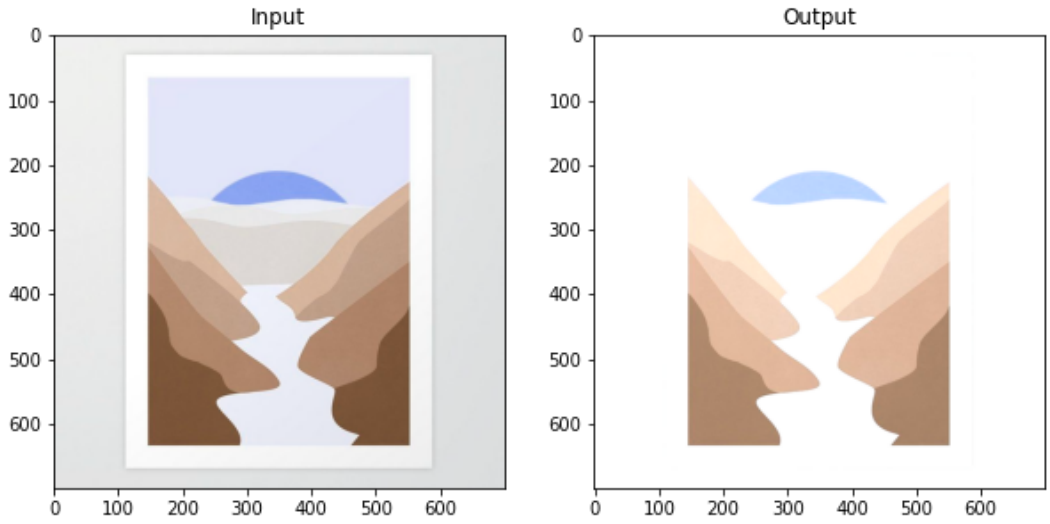

Figure 19

Lab Task 06

1. Histogram Calculation in OpenCV
   Use inbuilt OpenCV cv2.calcHist() function to display the histogram of a given image.

   Gray Image:

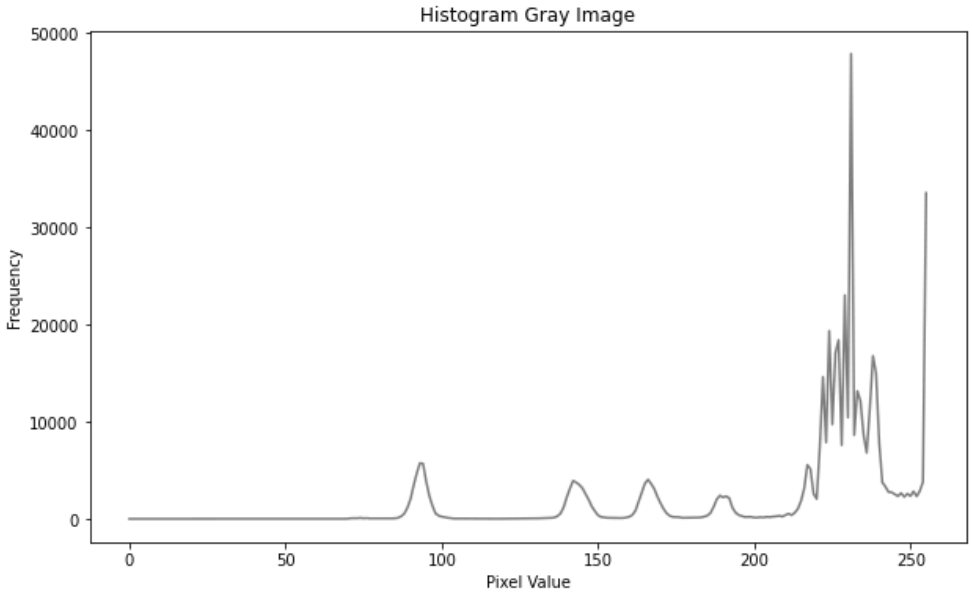

Figure 20

Colour Image:

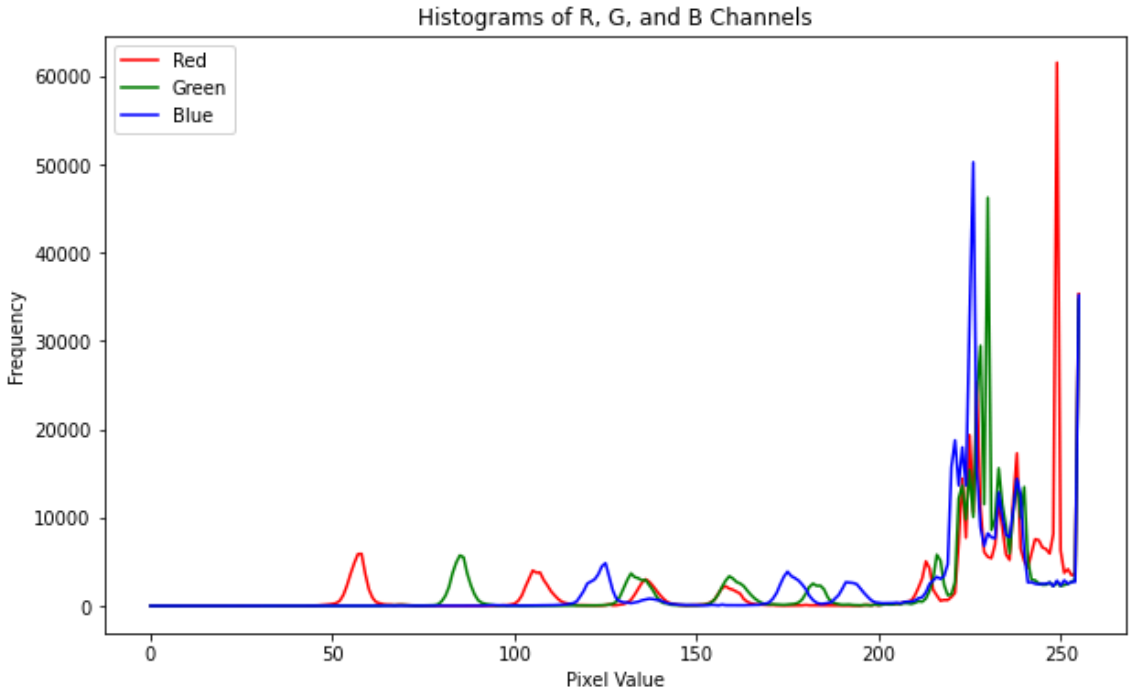

Figure 21

2. Histogram Calculation in Numpy
   Use the inbuilt numpy np.histogram() function to display the histogram of a given image.

   Gray Image:

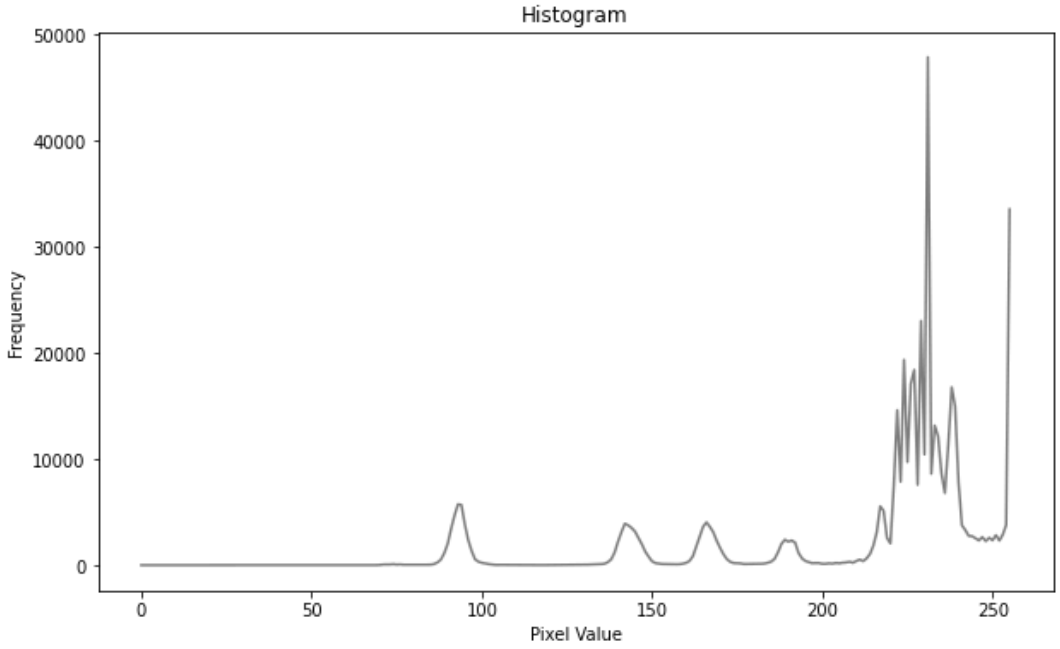

Figure 22

Colour Image:

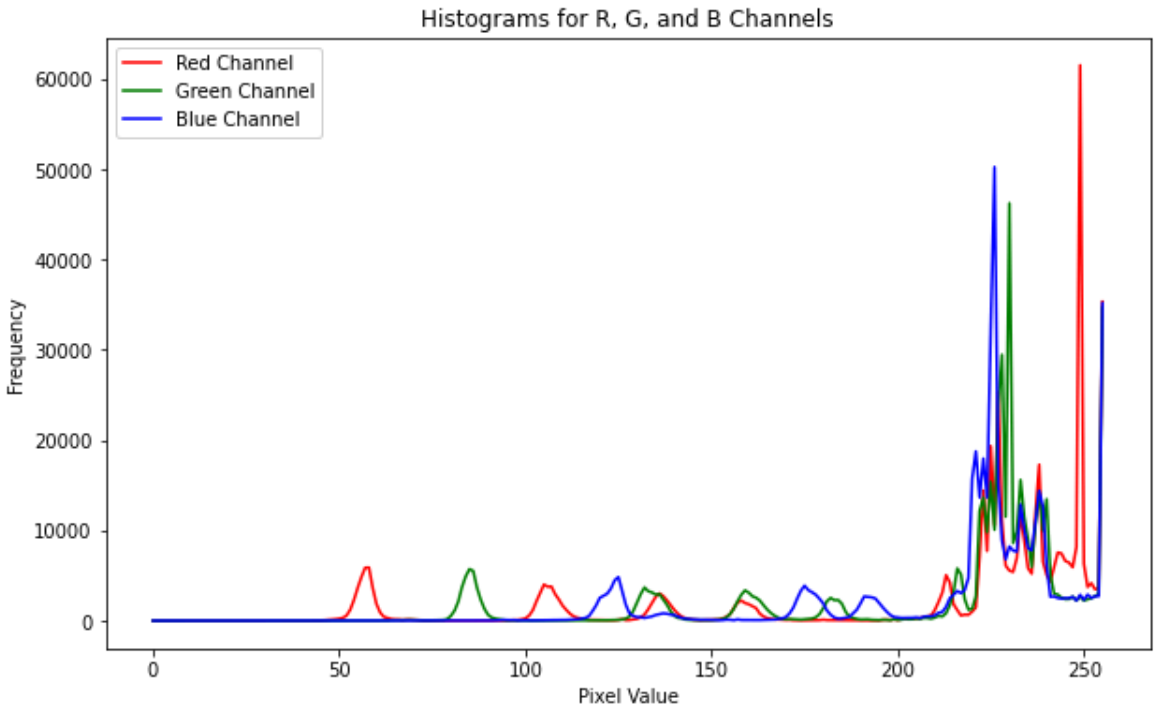

Figure 23

3. Then write your own histogram functions for the following scenarios

   a. Show a histogram plot for a grayscale image.

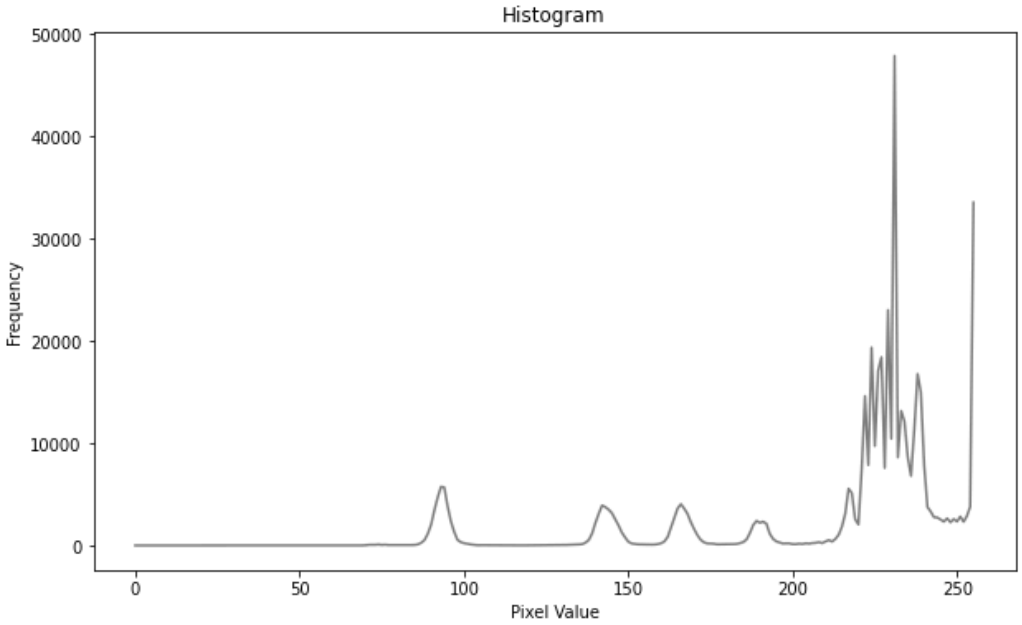

Figure 24

b. Show three histograms for a given RGB image.

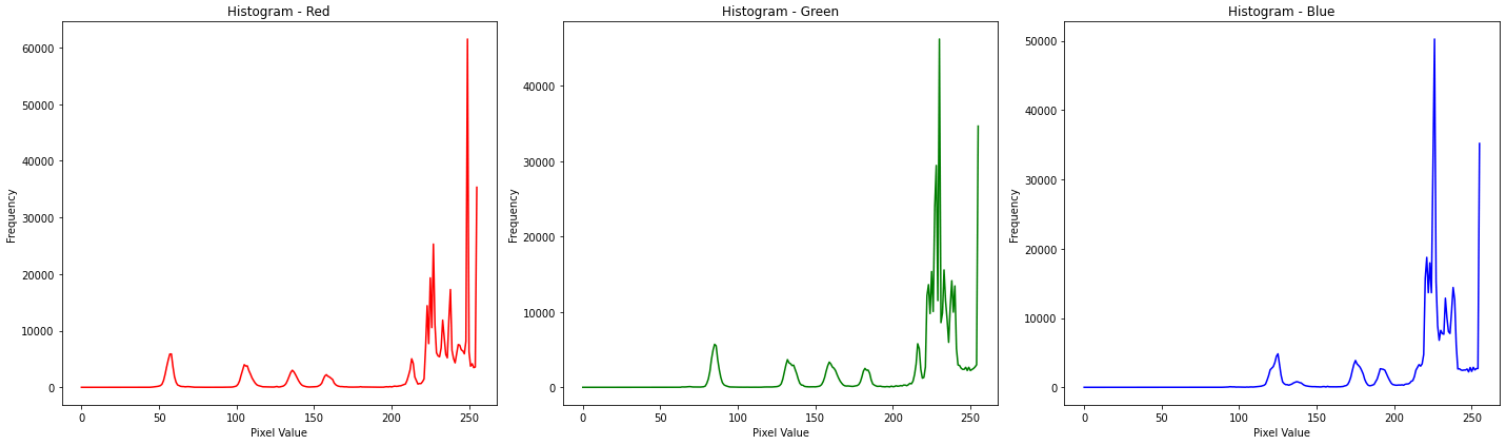

Figure 25

4. Consider the four images given in the resources folder. Plot the histogram for each image. Perform Histogram Equalization on each image and plot the histograms of the resultant images. Comment on the results you have obtained.

Image 01

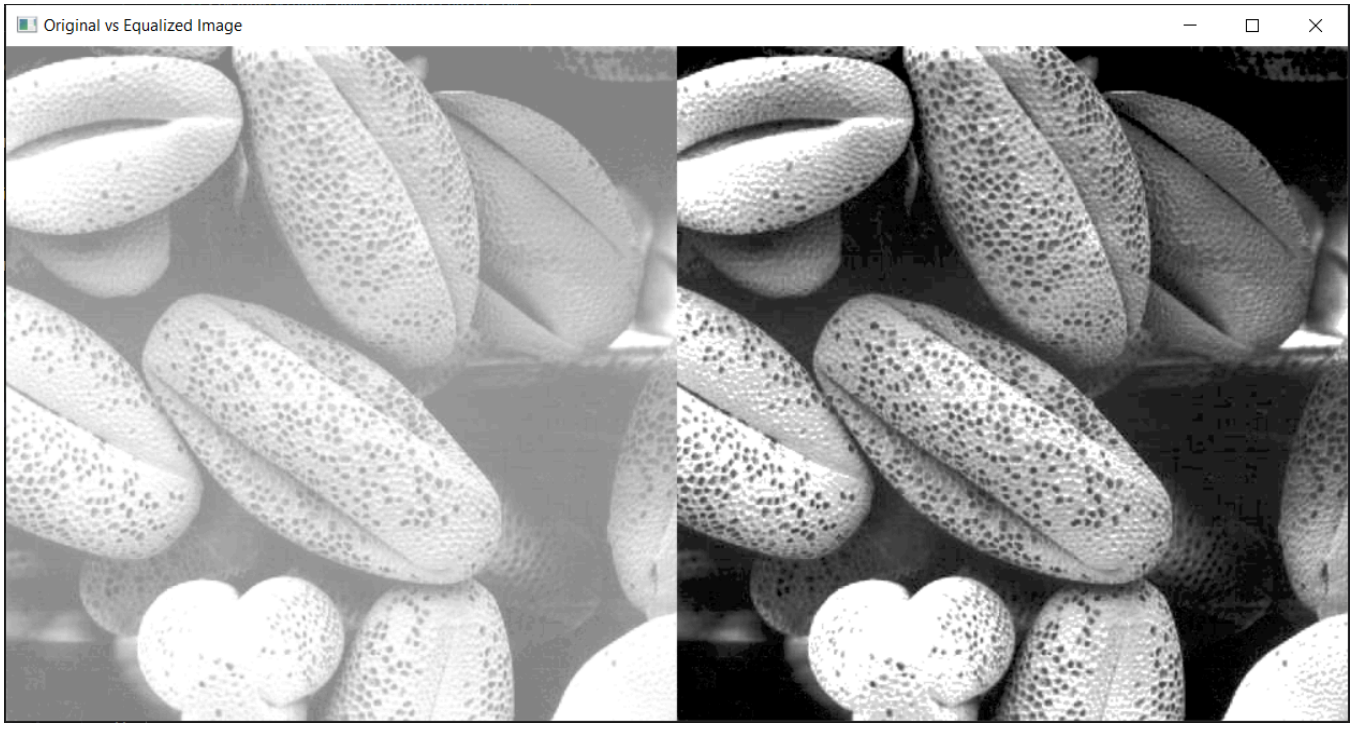

Figure 26 - Original Image and Equalized Image

Histograms:

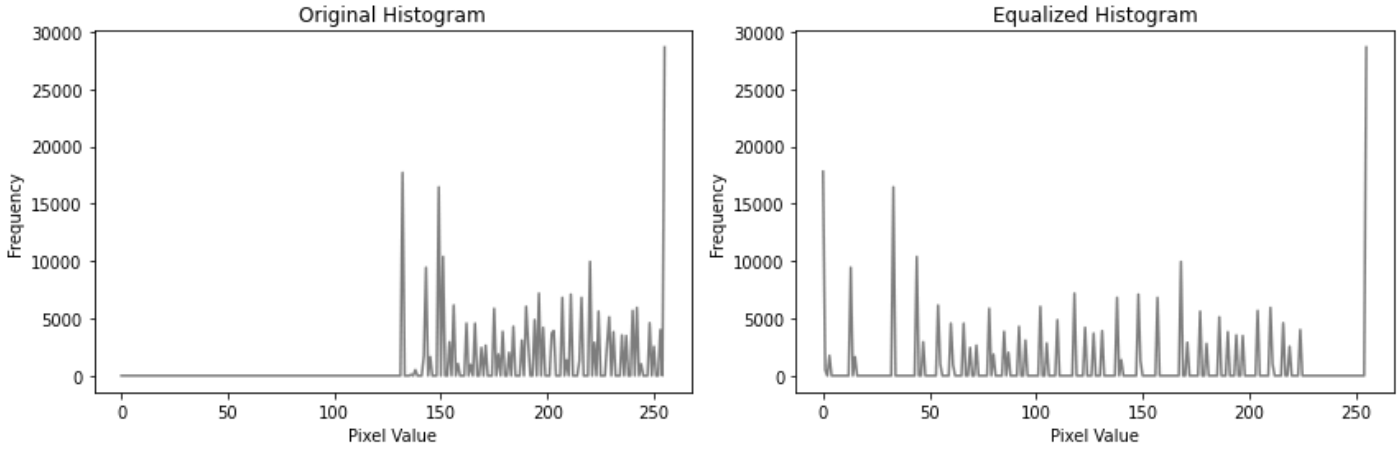

Figure 27

Image 02

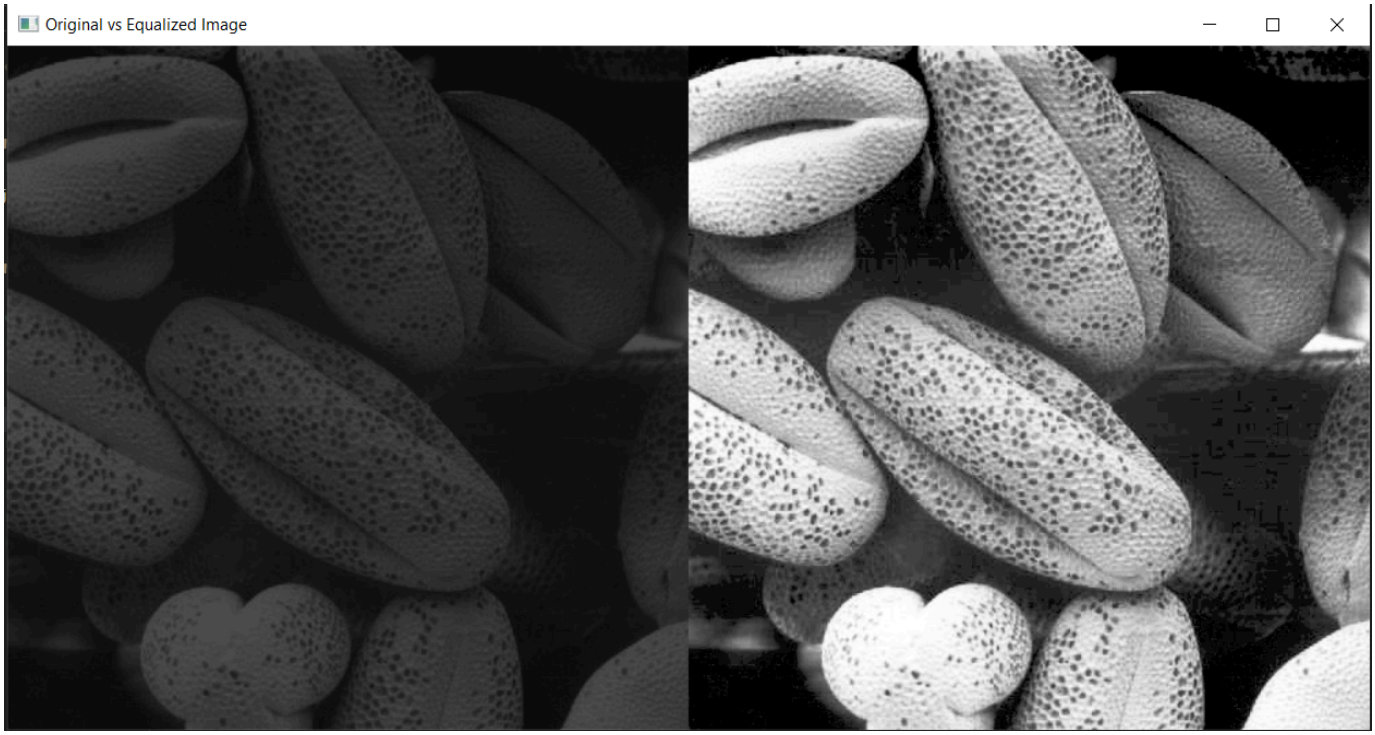

Figure 28 - Original Image and Equalized Image

Histograms:

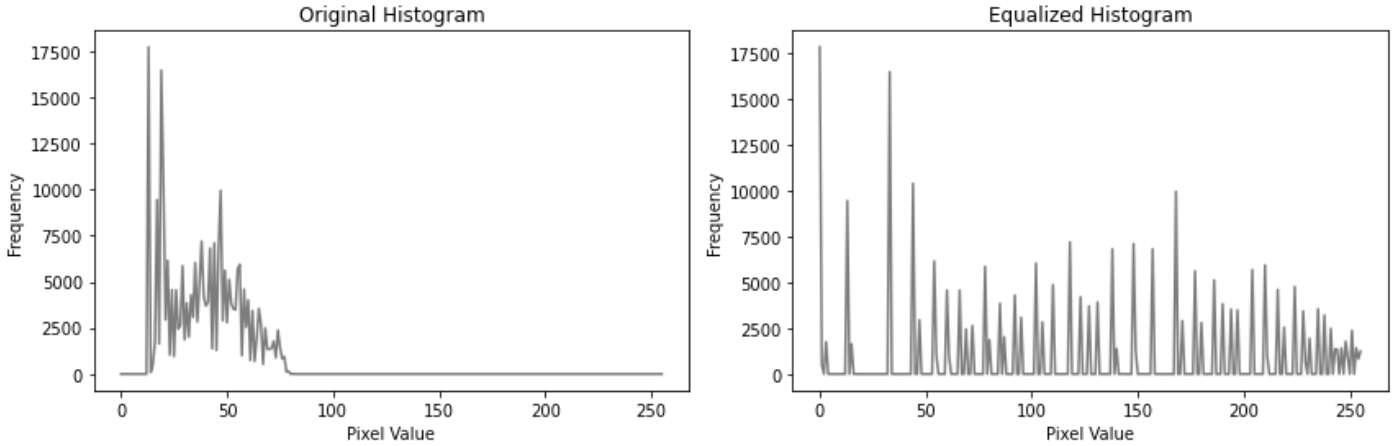

Figure 29

Image 03

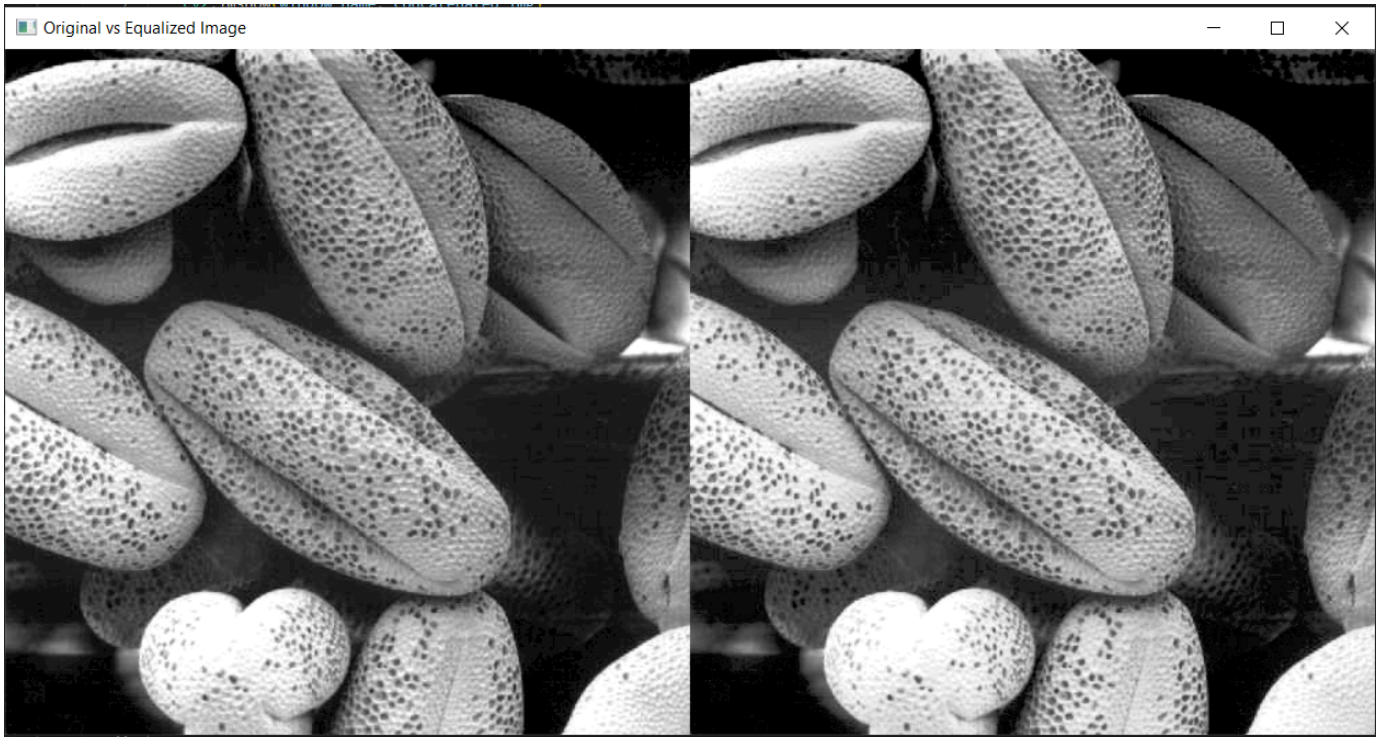

Figure 30 - Original Image and Equalized Image

Histograms:

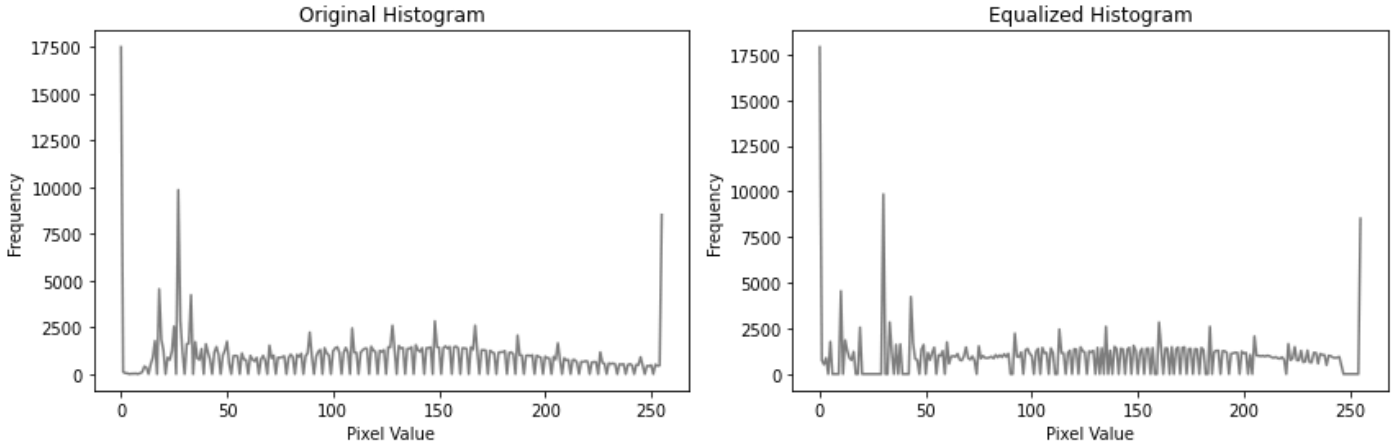

Figure 31

Image 04

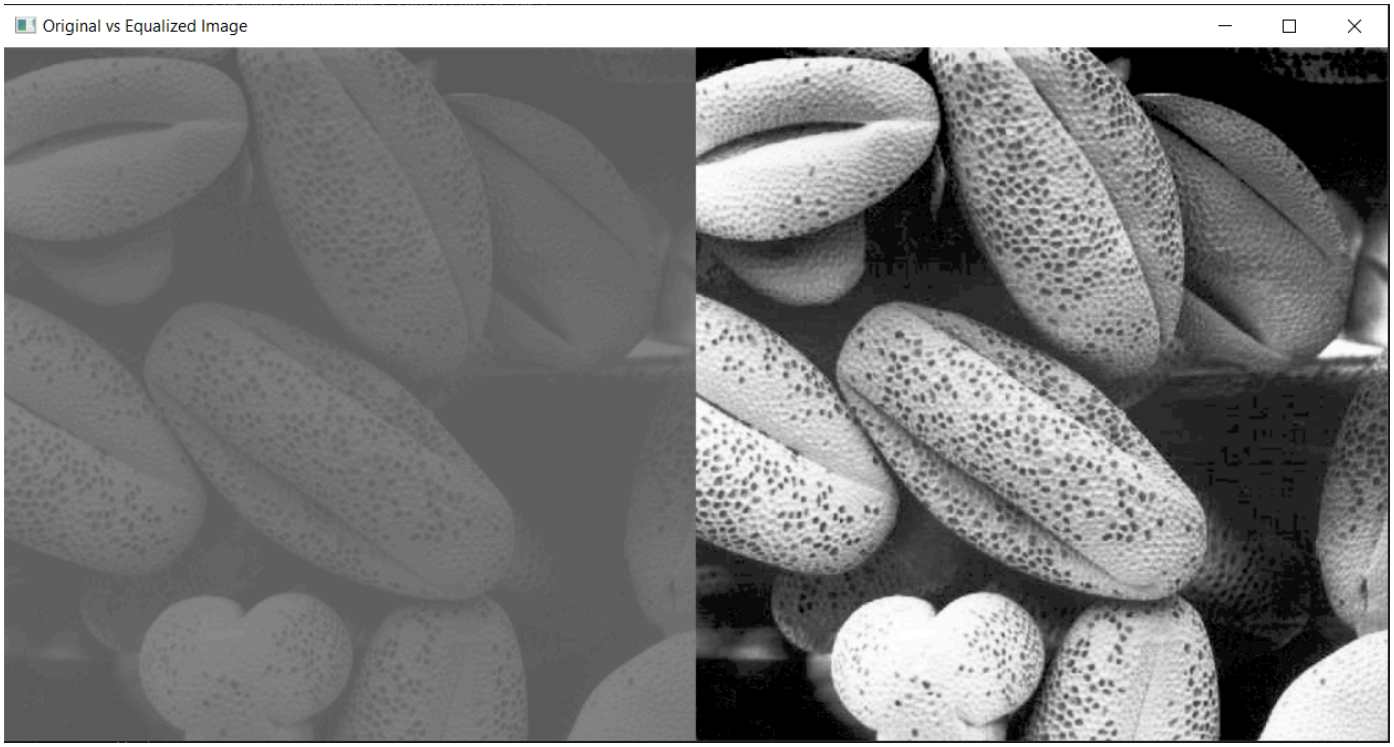

Figure 32 - Original Image and Equalized Image

Histograms:

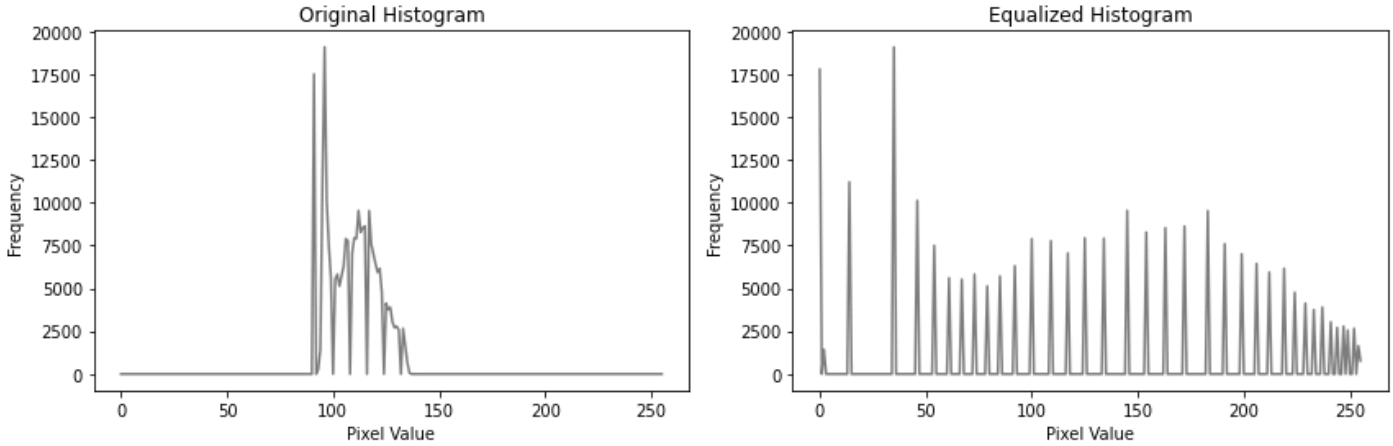

Figure 33

The original images exhibit varying concentrations of pixel intensities, reflecting their respective brightness and contrast levels. After applying histogram equalization, the histograms display a more uniform distribution of pixel intensities, resulting in improved overall contrast. However, this process can amplify noise, especially in regions with low pixel density. Despite this drawback, histogram equalization proves effective in enhancing image contrast by redistributing pixel intensities more evenly across the entire intensity range.