# Abstract

The goal of this lab is to perform histogram equalization on a grayscale image. Histogram equalization is a method in image processing of contrast adjustment using the image’s histogram. This MATLAB script in this project includes an equalize function that produces histograms of an input image and a contrast-adjusted version. This report will discuss the techniques used to successfully equalize a histogram of an input grayscale image, and what additional information the results of the equalization transformation reveal.

# Technical Discussion

## Loading Test Images into the Program

The program begins by initializing matrices “inputMatrix1” and “inputMatrix2” as variables containing pixel intensity values of two test images. The first input image is labeled as “Lab\_03\_image1\_dark.tif”, which represents a darkened grayscale image. The second input image is labeled as “Lab\_03\_image2\_light.tif”, which also represents a grayscale image, but whose contrast is lower (more white pixels) compared to the first input image. The dark and light image are assigned to the variables inputMatrix1 and inputMatrix2 using the imread() function. This process is shown below in Figure 1.

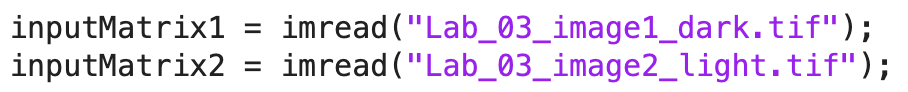


Figure 1: Importing grayscale test images.

## Pre-processing Methods

After the matrices “inputMatrix1” and “inputMatrix2” are created and assigned, they are each displayed. As shown below in Figure 2, the program creates a figure that will show the first original test image. This is done for both test images. Next, the program calls the helper functions compute\_mean() and compute\_std\_deviation(), which return the average pixel intensity value of the input image and the standard deviation of pixel intensities, respectively.

Graphical user interface, text

Description automatically generated

Figure 2: Displaying the first test image and prompting helper functions compute\_mean() and compute\_std\_deviation().

To create the histogram of the input images, the program prompts the helper function equalize(), which takes an input image as an argument. The matrices from Figure 1 will be used as arguments. The return value is an output matrix which includes the transformed image with an equalized histogram. Similarly, the output image is displayed after it is returned from the equalize() function and the program computes its average pixel intensity and standard deviation of pixel intensities. This process is shown below in Figure 3.

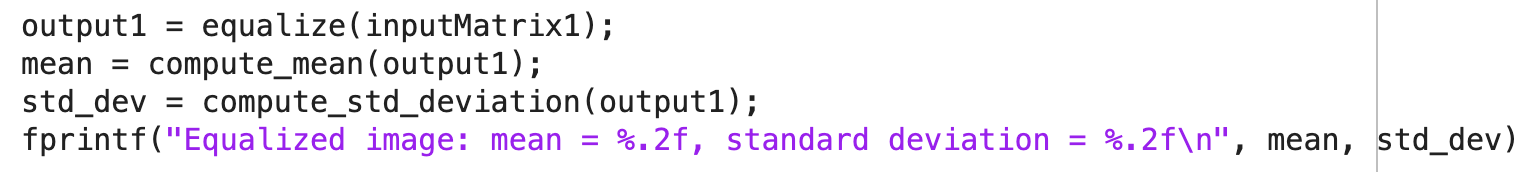


Figure 3: Displaying the transformed grayscale image and computing its average pixel intensity and standard deviation.

## The Structure of the compute\_mean() Function

The compute\_mean() function, as shown below in Figure 4, takes a grayscale image as an argument and returns a variable labeled “mean”. This return value has a type of “double” and is assigned at the end of the function. The function begins by initializing two variables “M” and “N”, whose values contain the number of pixel rows and columns in the input image, respectively. Next, the input image is type-casted into a “double” because grayscale images are registered as type “uint8”. By going through each pixel value in the input image, the total pixel intensity can be measured, and the average value can be assigned. The average pixel value is assigned at the end of the function.

Text, letter

Description automatically generated

Figure 4: The compute\_mean() helper function.

## The Structure of the compute\_std\_deviation() Function

The compute\_std\_deviation() function, as shown below in Figure 5, takes a grayscale image as an argument and returns a variable labeled “std\_dev”. Similar to the compute\_mean() helper function, compute\_std\_deviation() has a return value of “double” type and is assigned at the end of the function. The function begins by computing the mean pixel intensity of the input image by prompting compute\_mean(). Next, the program iterates through each pixel in the input image and assigns the standard deviation value to “std\_dev”. The equation for calculating standard deviation is shown in Figure 6, and more information of standard deviation can be found in Appendix A.

Text, letter

Description automatically generated

Figure 5: The compute\_std\_deviation() helper function.

Graphical user interface, text

Description automatically generated

Figure 6: Equation for determining standard deviation of a data population.

Results and Discussion

Conclusion

Appendix A