EE475 Homework #2

Batuhan Tosun, ***2017401141***

I. NONLINEAR MONOTONIC POINT OPERATIONS

In this question, to find the R value, the range over which the gray-level histogram is non-zero, of the Lena image the absolute difference between maximum and minimum pixel values (gray levels) of the image is calculated and assigned to R, which is found as 210.

Then, the f and g transform of the image are taken with respect to the given formulas and $\alpha = 0.8$. The original and the resulting images, and their corresponding histograms are given below.

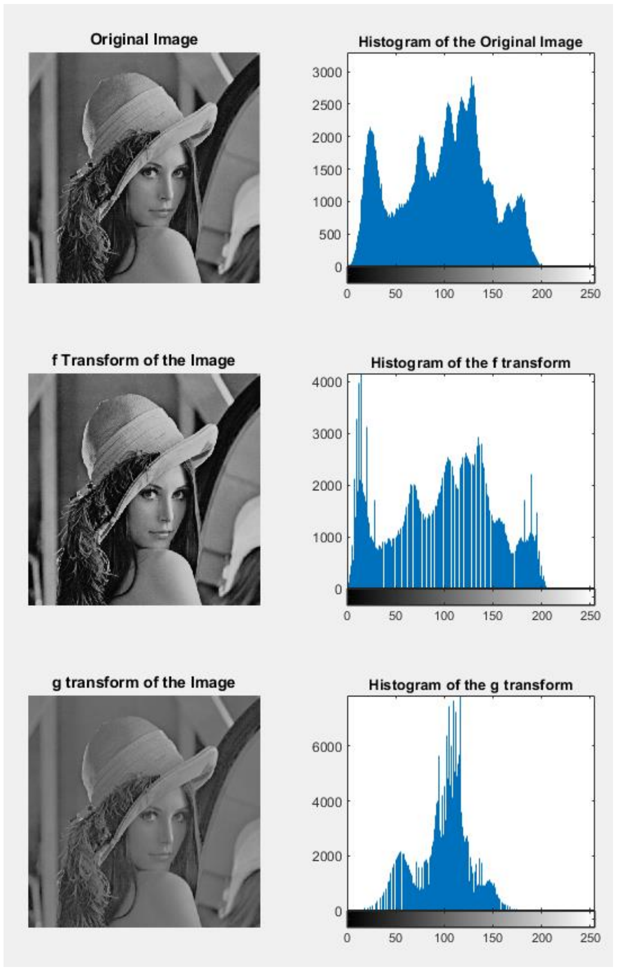


Fig 1. Original, f-transformed, g-transformed images and their corresponding histograms

From the histogram plots it can be observed that f-Transform of Lena image results in high pixel values to go higher and low pixel values to go lower without expanding the R range (pixel values are still in 0-209 range). This situation causes an image with higher contrast. g-Transform, on the other hand, results in high pixel values to go low and low pixel values to go high, therefore an increase in the number of occurrences (concentration) in the middle gray levels. This situation lessens the contrast in the image.

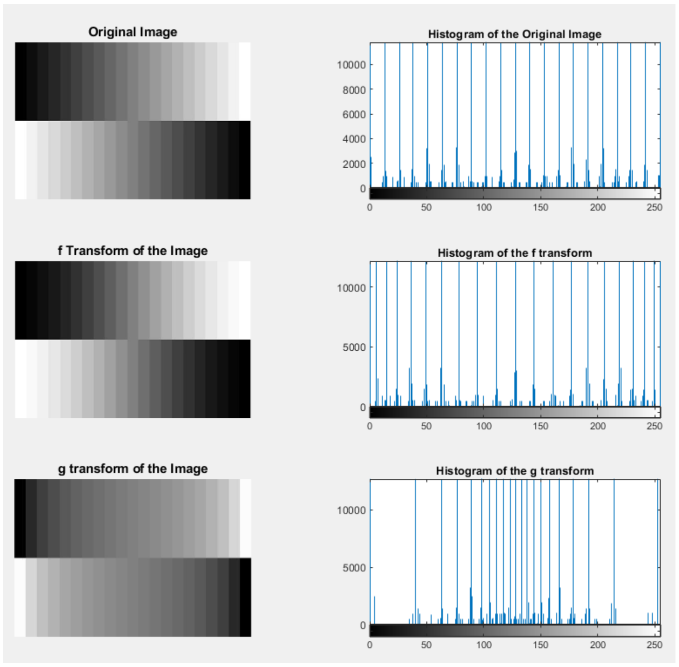


Fig 2. Original, f-transformed, g-transformed versions of the example image and their corresponding histograms

II. INTENSITY TRANSFORMATIONS

A. The Log Transformation

The log transformation of the Fractured_spine image is taken with c = 60 to get a meaningful image.

B. The Power-Law Transform

The power-law transformation of the Fractured_spine is taken with c=40 and $\gamma = 0.4$ to get a meaningful image.

In both transforms, the parameters are chosen in a way that the spine becomes visible in the image. The original and the resulting images are given below.

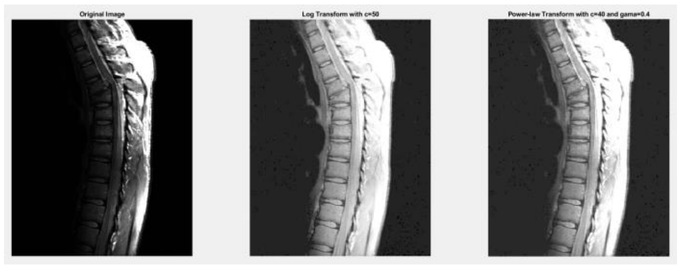


Fig 3. Original, Log transformed, and power-law transformed images

Before commenting on the differences between the log and power-law transforms, we should comment on their characteristics first. The log transformation maps a narrow range of low intensity values in the input into wider range of output levels and compress the ones in the higher-level values. Considering the characteristics of the log function this makes sense. Power-law transformation, on the other hand, might behave very distinct depending on the chosen γ value. For γ<1, power-law transformation performs a very similar task to log-transform but for γ>1 it performs exactly the opposite. Since we are working on the same image, the γ parameter is set to a value below 1. In the case of an image which needs an operation like mapping a narrow range of high intensity values to wider range, the power-law operation with γ>1 is more suitable for the enhancement.

III. HISTOGRAM EQUALIZATION

A. My Own Computer Program for Computing Histograms

In this question, I have implemented my own computer program (MATLAB function: myHistogram) for computing the histogram of an image. myHistogram function takes the image and a Boolean variable for plotting the histogram as inputs and returns two arrays, one holds the frequency of occurrence values and the other holds the gray-level range.

To find the total number of occurrences for each gray-level value in an image a basic for loop is designed (can be seen in the source code). After this step, if the given Boolean parameter is True the function plots the histogram as well in a bar plot.

B. My Computer Program for Histogram Equalization

In this question, I have implemented my own computer program (MATLAB function: myHistEq) for implementing histogram equalization. myHistEq function takes two inputs: an image matrix and an array with the number of occurrence information of each gray level.

First, the function, computes the probability mass function of gray level values by dividing the occurrence array by the total number of pixels in the given image (MxN). The using MATLAB's cumsum function the cumulative mass function (stored in another array) of gray level values are found. Then, each gray level value in the image is changed with its corresponding cumulative mass function value. To rescale, the pixel values in the resulting image we multiply each value with 255.

C. Histogram Equalization on the Image

Using myHistogram and myHistEq functions, I have implemented histogram equalization on the Fractured_spine image and calculated its spectrogram. The original and resulting images, and their corresponding histograms are given below.

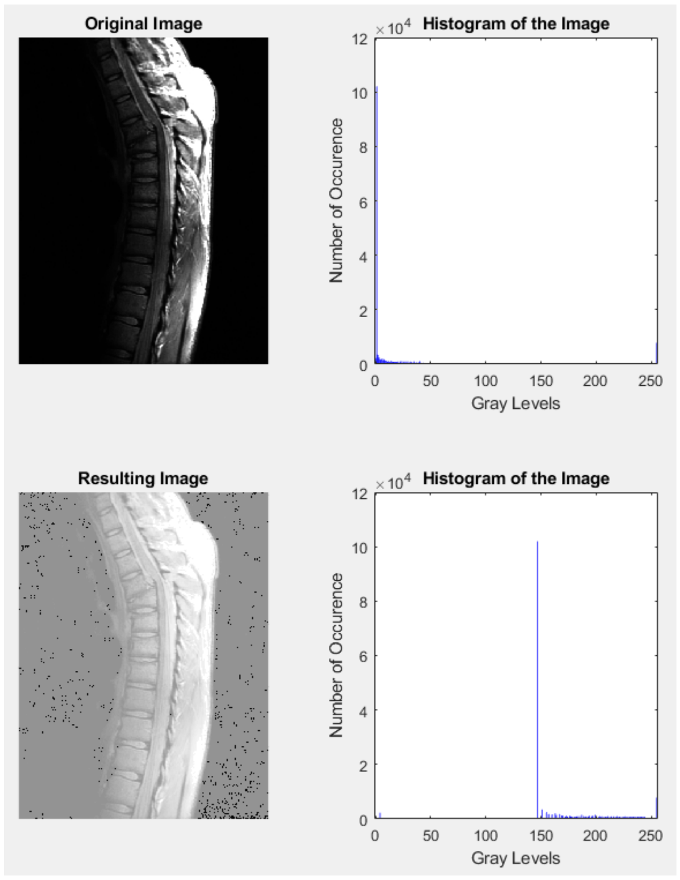


Fig 3. The original (Fractured_spine.jpg) and the resulting image after histogram equalization and their corresponding histograms

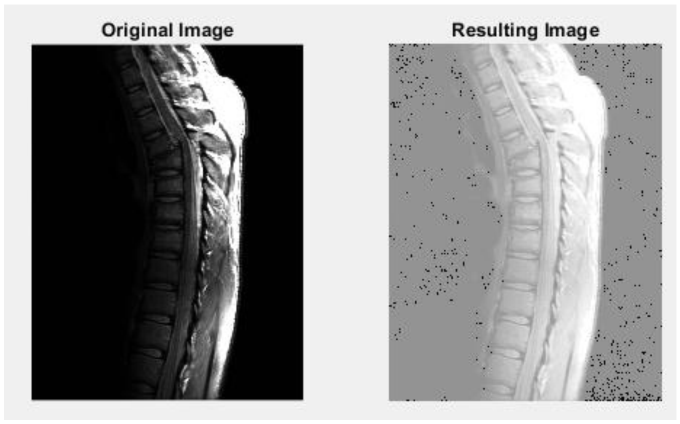


Fig 4. The original (Fractured_spine.jpg) and the resulting image after histogram equalization (only images)

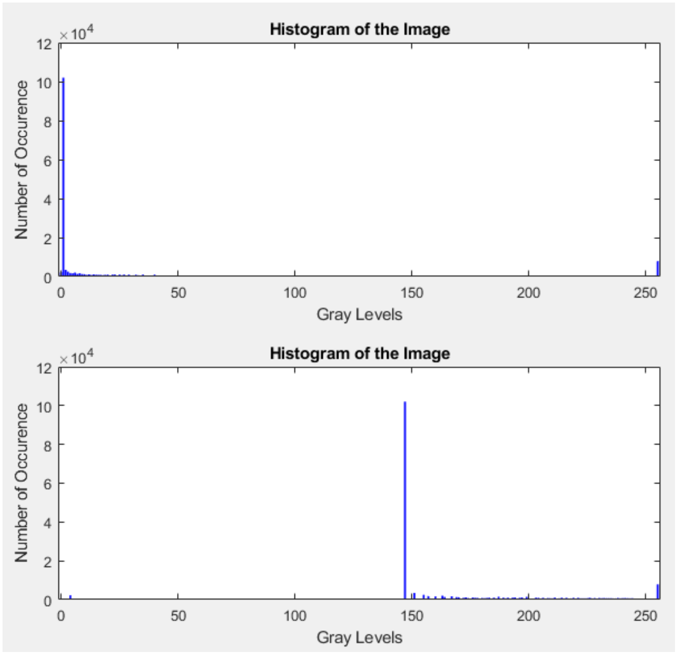


Fig 5. Only the histograms of the original (above) and the resulting(below) images

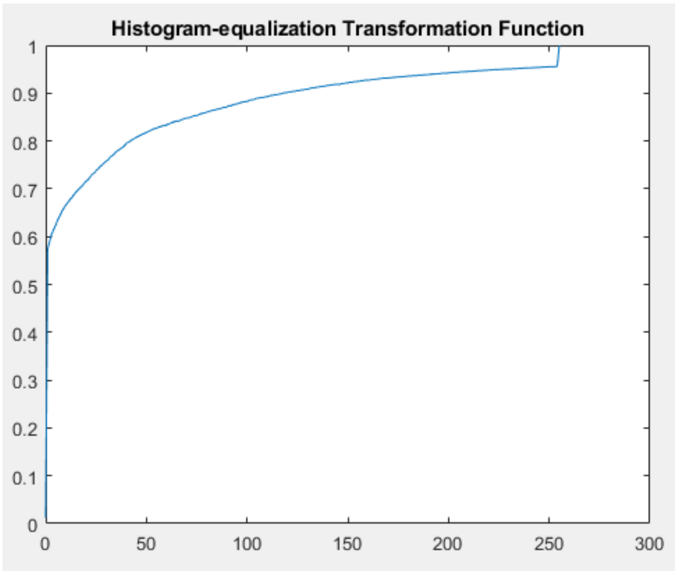


Fig 6. Plot of Histogram-equalization Transformation Function

To validate my results obtained using my own histogram equalizer function, now MATLAB's histeq function is used to implement histogram equalization on the same image. In addition to histeq, MATLAB's adapthisteq function is used as well to perform adaptive histogram equalization method. The original, histogram equalized, and adaptive histogram equalized images and their corresponding histograms are given below.

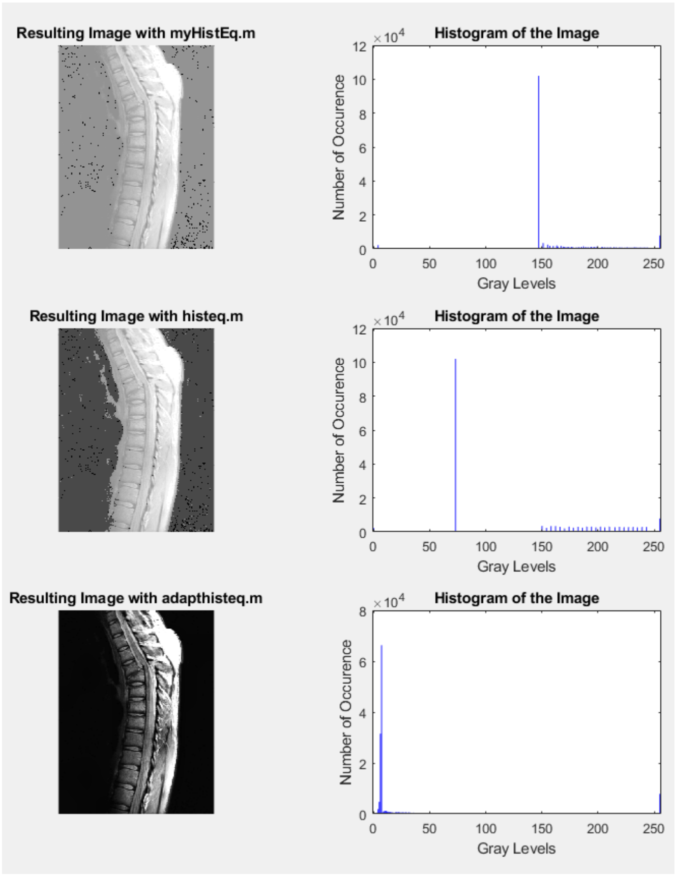
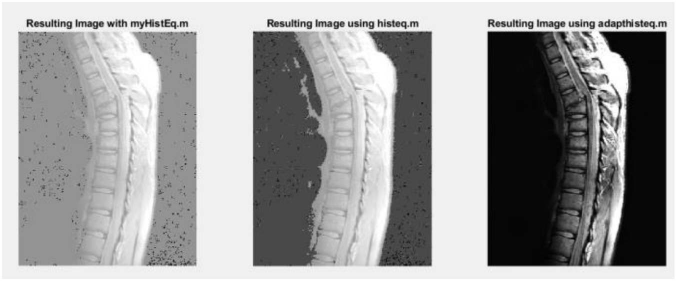


Fig 7. The histogram equalized (with myHistEq.m and histeq.m), adaptive histogram equalized (with adapthisteq.m) images and their corresponding histograms



Resulting Image with myHistEq.m
Resulting Image using histeq.m
Resulting Image using adapthisteq.m

Fig 8. The histogram equalized with myHistEq.m (leftmost) and histeq.m (middle), adaptive histogram equalized with adapthisteq.m (rightmost) images

From the figures, we can observe that the resulting images obtained by using my (myHistEq.m) and MATLAB's (histeq.m) histogram equalization functions are not completely the same. It can be clearly seen that my function is insufficient in obtaining/assigning very low pixel values in/to the useless regions of the image while MATLAB's function does this properly. However, these functions have similar performances on enhancing information lies in the useful regions (where the fractured spine is) of the image.

From fig. 8 it can be observed that the adaptive histogram equalization function/method has the best performance in improving the contrast in the image. The success of this method lies in the fact that it computes not one but several histograms, each corresponding to a different section of the given image. Then, it uses these histograms to arrange the brightness values of the sections. After that, those sections are brought back together and creating contrast enhanced image.

D. Histogram Equalization on another Image

Using myHistogram and myHistEq functions, I have implemented histogram equalization on the Baby image and calculated its spectrogram. The original and resulting images, and their corresponding histograms are given below.

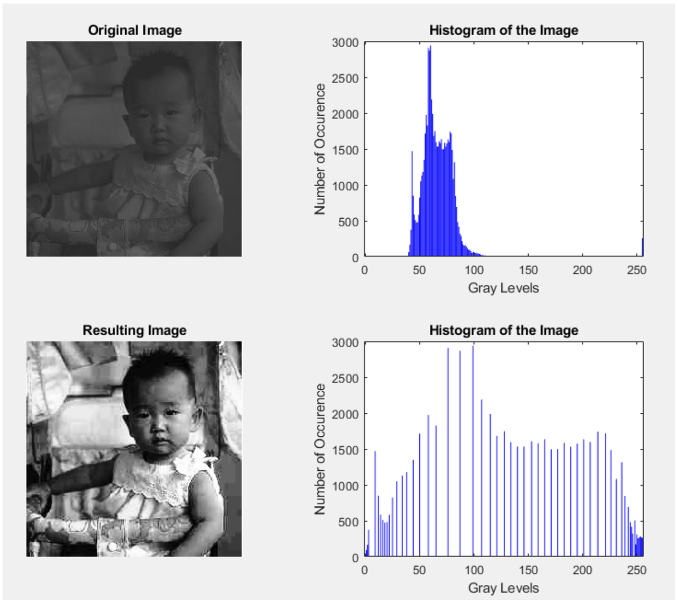


Fig 9. The original (Baby.jpg) and the resulting image after histogram equalization and their corresponding histograms

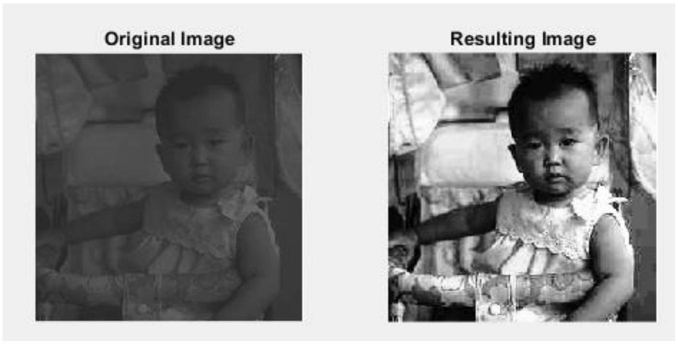


Fig 10. The original (Baby.jpg) and the resulting image after histogram equalization

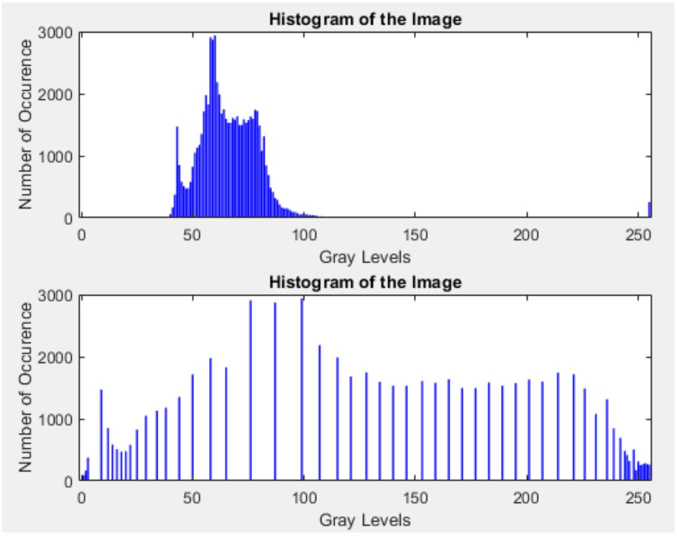


Fig 11. Only the histograms of the original (above) and the resulting(below) images

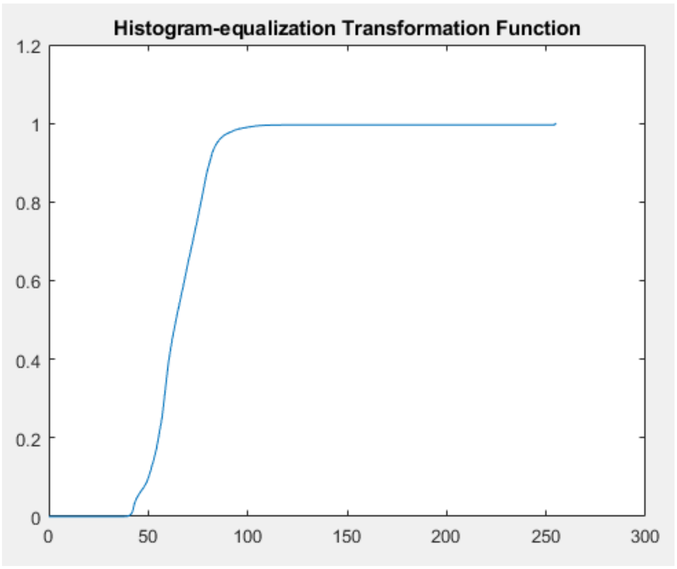


Fig 12. Plot of Histogram-equalization Transformation Function

To validate my results obtained using my own histogram equalizer function, now MATLAB's histeq function is used to implement histogram equalization on the same image. In addition to histeq, MATLAB's adapthisteq function is used as well to perform adaptive histogram equalization method. The original, histogram equalized, and adaptive histogram equalized images and their corresponding histograms are given below.

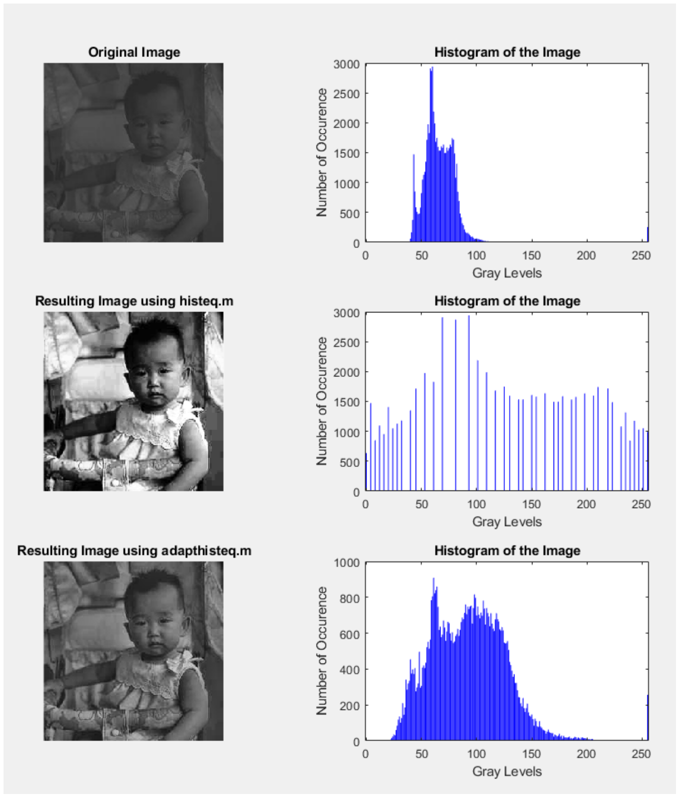
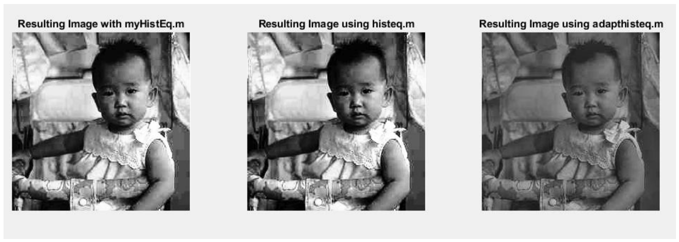


Fig 13. The original (Baby.jpg), histogram equalized, adaptive histogram equalized images and their corresponding histograms



Resulting Image with myHistEq.m
Resulting Image using histeq.m
Resulting Image using adapthisteq.m

Fig 14. The histogram equalized with myHistEq.m (leftmost) and histeq.m (middle), adaptive histogram equalized with adapthisteq.m (rightmost) images

From the figures, we can observe that the resulting images obtained by using my (myHistEq.m) and MATLAB's (histeq.m) histogram equalization functions are almost the same this time. Also, from fig. 8, it can be observed that the adaptive histogram equalization function/method has the best performance in improving the contrast in the image.

IV. IMAGE RESIZING

In this question, I developed a program to resize the image "cameraman.tif" from its original size of 256x256 to an enlarged size of 400x400 using bilinear interpolation.

To achieve this task, first, I rescaled (squeezed) the desired pixel coordinate values [1:400] to [1:256]. After that, to find the value of each pixel (in total 400x400=16x10^4) in our desired image bilinear interpolation is implemented. The resulting 400x400 image "RESIZED_CAMERA.jpg" then saved using MATLAB's imwrite function.

(Detailed explanation for each operation done for image resizing is given in the source code!!!)

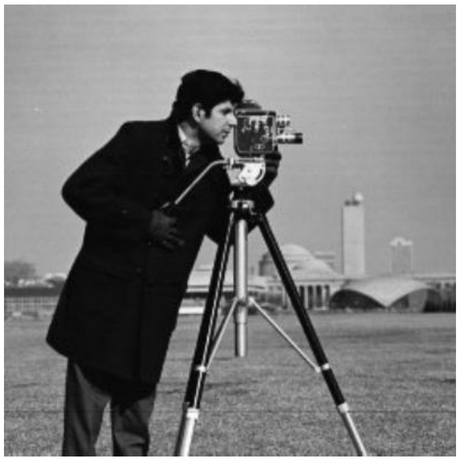


Fig 15. The resulting "RESIZED_CAMERA.jpg" image with size 400x400

V. COLOR IMAGE ENHANCEMENT

A. *Contrast Stretching for RGB Channels of an Image*

In this part of the question, I performed contrast stretching for each color channel separately and then combined the resulting RBG layers to obtain resulting image. In this process, I have chosen the lower limit a as 0 and the upper limit b as 255 for every channel. The lower and upper limit values (c and d) in each layer of our image are also recorded. The lower limit value c is the same for all layers and it is equal to 0. The upper limit value d is different for all layers and for R channel it is 144, for G it is 133 and for B it is 137. Using these values, the contrast stretching operation is implemented according to the given formula. The original and the resulting images are given below.

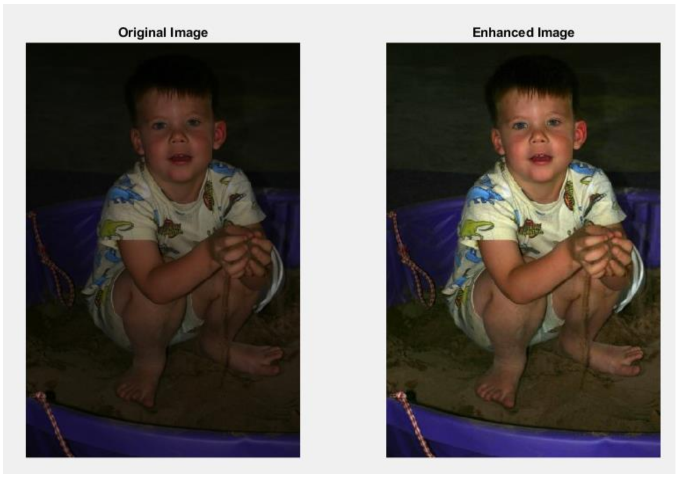


Fig 16. Original and Contrast Stretched (in RGB space) Images

B. *Contrast Stretching for Value Channel of an Image*

In this part of the question, the RGB image is first converted into HSV color space by using MATLAB's rgb2hsv function. This time, contrast stretching operation is applied to only the value channel. Since the value channel values are normally in between 0 and 1, I have chosen the lower limit value a as 0 and the upper limit value b as 1. I also find the lower and upper values in the value channel and implemented contrast stretching according to the given formula again. After this, the resulting image is converted back to RGB color space using MATLAB's hsv2rgb function. Then, the resulting image is multiplied with 255 to get pixel values in between 0 and 255. The original and the resulting images are given below.

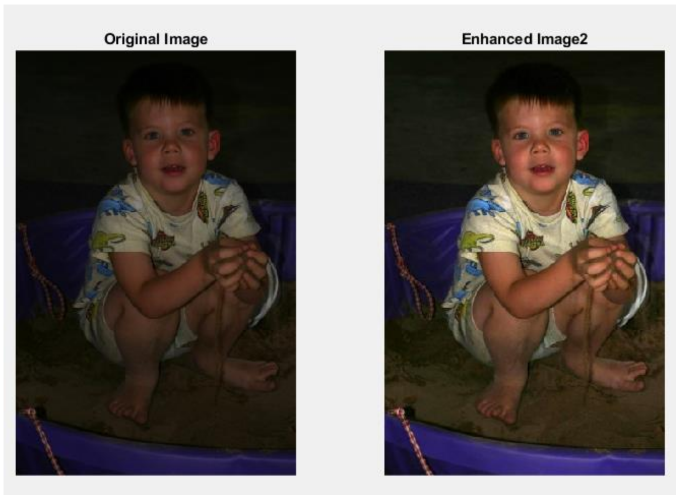


Fig 17. Original and Contrast Stretched (in HSV space) Images

First of all, it can be observed that the original image suffers from low contrast. As a result, we loss information about both the child and its surroundings. In this question, the low-contrast problem is solved by a technique called "contrast stretching" done following two different ways. Contrast stretching is basically spanning the range of intensity values in an image to a desired range of values. The first approach was applying contrast stretching on the RGB channels of the image separately and then bringing them together. The resulting image for this approach can be observed in Fig 16. It can be observed that the contrast of the image is improved and now we can see better both the child and its surroundings. To apply the second approach, the image is first converted to HSV format using MATLAB's rgb2hsv. Then the contrast stretching is only applied on the value channel (in which the information about the brightness values is kept). Then, we convert the image back to RGB color space. The resulting image after this approach can be seen in Fig 17. Again, it can be observed that the contrast of the image is improved and now we can see better both the child and its surroundings.