Histogram Equalization

Lab Assignment #3

Ralphilou Tatoy

March 30, 2022

Abstract:

Implementation of functions that performs histogram equalization. By acquiring the histogram of an image, transformation of equalization can be performed by using probabilistic solution. Using a darker or lighter grayscale image, the output will be more equal in pixel. The use of many functions such as compute_histogram, plot_histogram, equalize, and histogram transformation function creates an equalize function. Compute_histogram creates a histogram of an input image. Plot_histogram creates a plot of an input histogram. The histogram transformation creates a look up table to be used in creating an equalize image of an image that are unequaled. The equalize function uses this function to collect information to perform histogram equalization. Using the old image to translate a new pixel value for the new equalized image.

Technical Directions:

The compute histogram function uses two for loops that essentially collects data of how many times a pixel value is registered in an image. The two for loops to cover all the pixel values of the image. The histogram created must be in a percentage scale so the division respective to the size of the image is necessary.

The plot histogram outputs a plot of the histogram. I used the bar as that is the most like the example given in the prompt for lab 3. It shows the relevance of each pixel on the y axis and the pixel value in question in the x axis. Using standard function of plot, I added the title, changed the color, and added axes titles

The equalize function is a function that performs the histogram equalization process by calling the function of compute histogram and plot histogram. Once acquiring the look up table for the equalization. Using the old image and the look up vector, the function can determine which pixel value of the old pixel value is allocated to a new pixel value. Using for two for loops to get to all the pixel location of an image, the function transfers the pixel value into a new image. To ensure that the index is an integer, rounding the output of the look up table is necessary. This new image is the equalize image. To calculate the mean, I used the probabilistic methods such that the mean intensity of an image is given by adding all the possible multiplication of the pixel value and the probability of the pixel value. Need to account for all pixel intensity. For calculating the standard deviation, I also used the probabilistic methods such that the variance of intensities is given by adding all the possible multiplication of the probability of an intensity and the subtract of the pixel value and the mean squared. When the variance is calculated, use square root to calculate the standard deviation.

The histogram transform function uses an equation to create a look up table using the original image histogram. The equation is straight forward to implement. You add up the probabilistic of a pixel occurring from the starting point of zero through to the pixel number in question and multiply it by the overall number of pixels that is considered. The output will be the corresponding value for each pixel number.

Results

The result is what I would expect from the collection of functions. The darker image (figure 1) outputs a histogram indicative of a dark histogram (figure 2), and the light image (figure 5) outputs a histogram indicative of a light histogram (figure 6). Doing the histogram equalization, the image becomes a lot clearer or equalizes (figure 3 and figure 7). The histogram of the equalize image shows a histogram (figure 4 and figure 8) that shifted to more equalize along the pixel values. The relevance of the pixel value in the image is shown in the equalize image. The equalized images are almost identical. The original image seems to be mor lighter than both the equalized images. The equalize image looks better because it highlights the dark areas, so it makes it pop out a lot better than the original image.

Figure 1: Original Dark Image

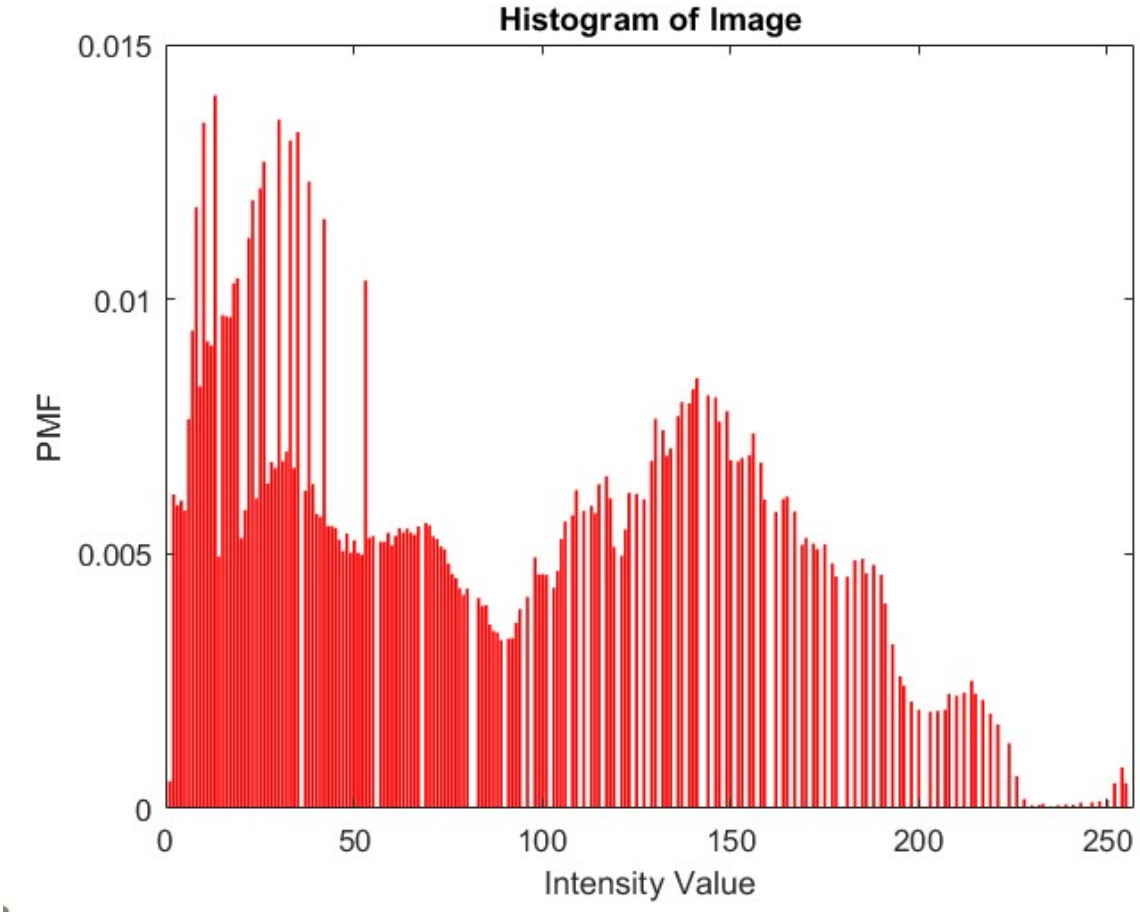

Figure 2: Histogram of Original Dark Image

Figure 3: Equalize Dark Image

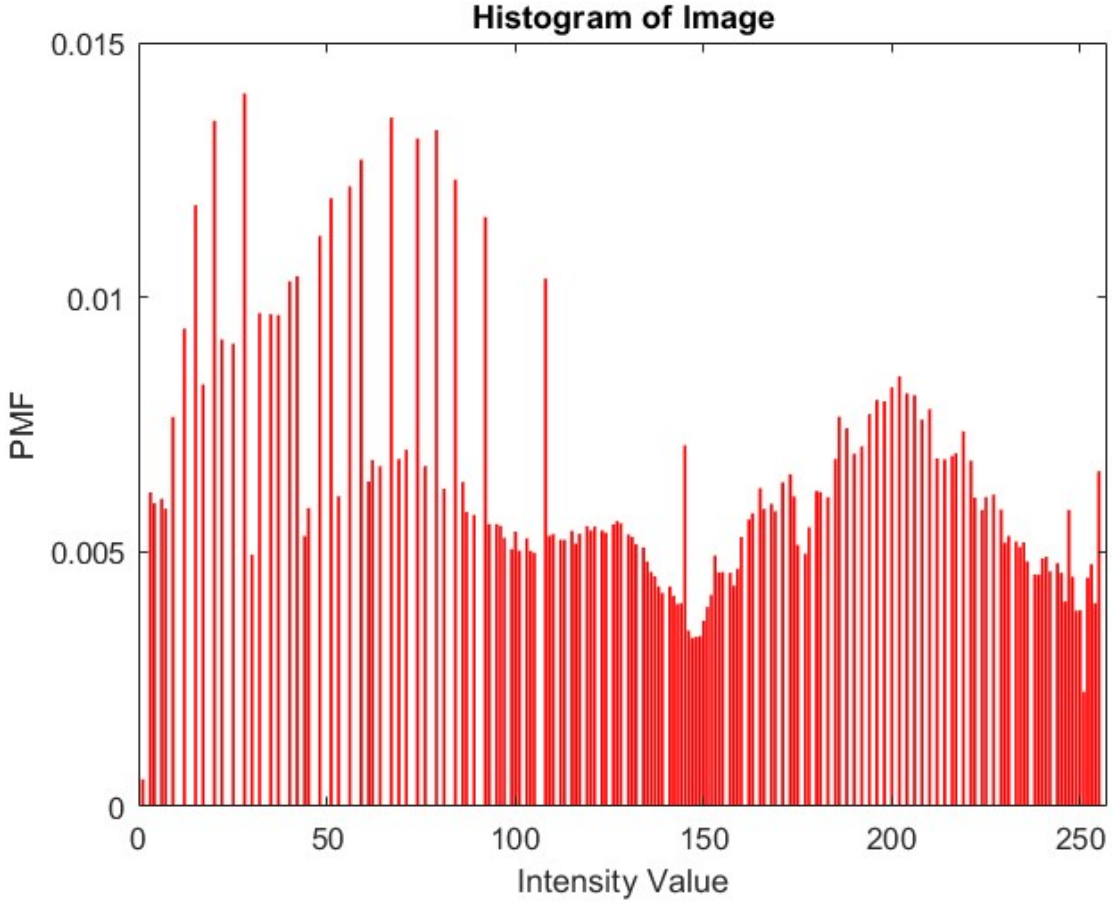

Figure 4: Histogram of Equalize Image

Figure 5: Original light image

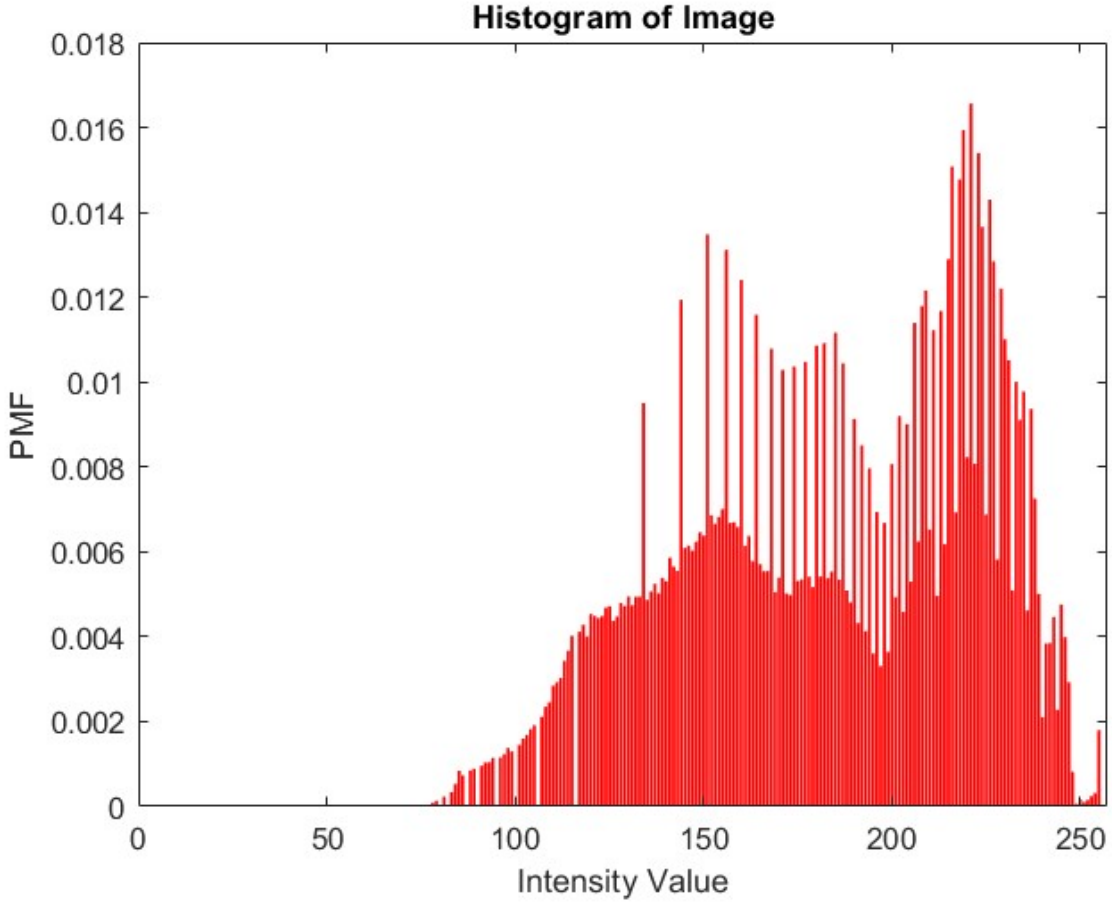

Figure 6: Histogram of original light image

Figure 7: Equalize light image

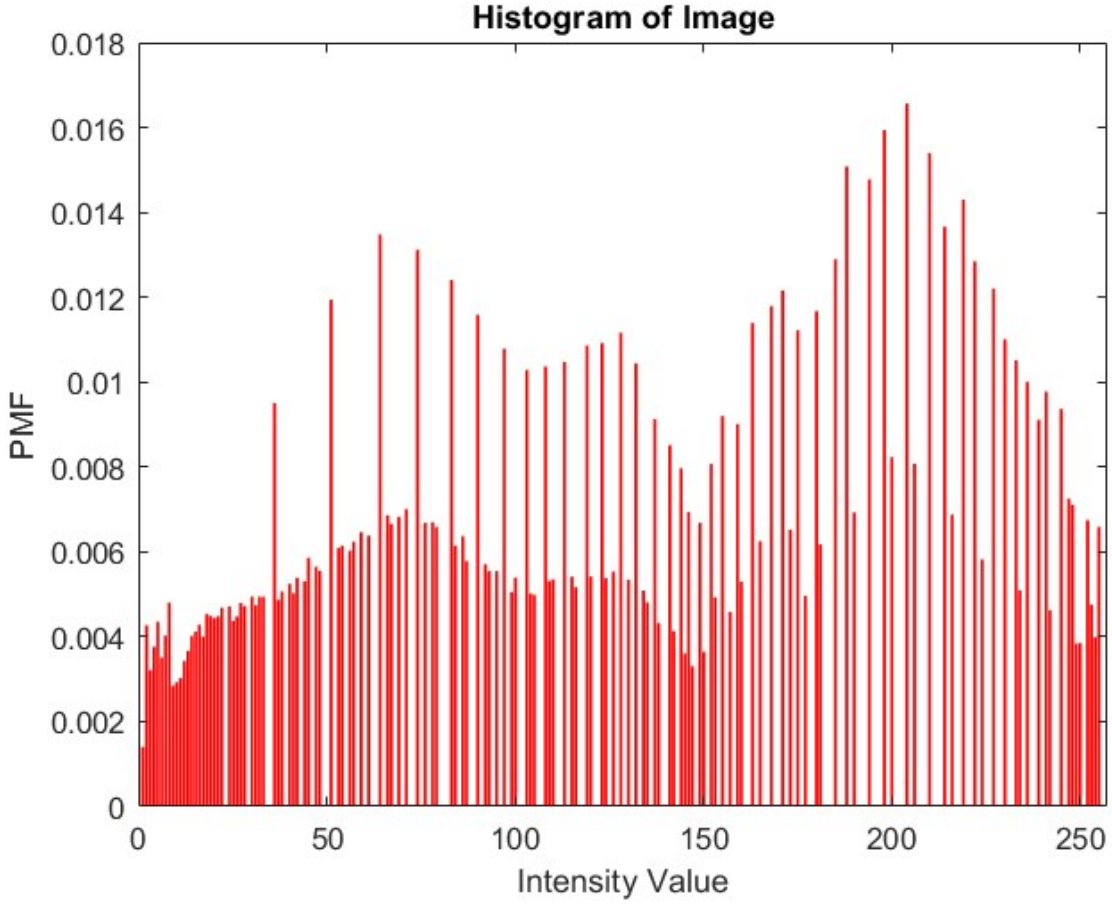

Figure 8: Histogram of equalize light image

Figure 9: Original Image

```
fprintf("Dark Image\n");
in_image = imread("Lab_03_image1_dark.tif");
out_image = equalize1(in_image);
figure;
imshow(in_image);
figure;
imshow(out_image);
fprintf("Light Image\n");
in_image1 = imread("Lab_03_image2_light.tif");
out_image1 = equalize1(in_image1);
figure;
imshow(in_image1);
figure;
imshow(out_image1);
```

```
function image2 = equalize1(image1)
% The equalize function is to implement the overall process of making a
% histogram, histogram transformation, and using histogram transmation to
% equalize an image.
%
%
% Syntax:
% image2 = equalize1(image1)
%
% Input:
% image 1- grayscale image
%
% Output:
% image2 - equalize grayscale image
%
% History:
% Ralphilou S. Tatoy 3/29/22 - Created
% Ralphilou S. Tatoy 3/29/22 - Finished implementing the transformation
%   using the function T to do look up table.
% Ralphilou S. Tatoy 3/30/22 - Finished mean and standard deviation
%   implementation.
% Ralphilou S. Tatoy 3/30/33 - Finished commenting

%Size of original image
[sizeX,sizeY] = size(image1);

%Calling to create histogram of the original image
h = compute_histogram(image1);

%Plotting original image histogram
plot_histogram(h);

%Performing look up table
T = histogram_transform(h);

%Creating uint8 image for output (Equalize image)
image2 = uint8(zeros(sizeX,sizeY));

%For loop to intergrate equalize image by use of the look up
for x=1:sizeX
    for y =1:sizeY
        image2(x,y) = T(image1(x,y)+1);
    end
end

%Create histogram the equalize image
h1 = compute_histogram(image2);

%Plot the equalize histogram
```

```
plot_histogram(h1);

%Initialize standard deviation
meanI1 = 0;
meanI1EQUAL = 0;
standardDEV = 0;
standardDEVEQ = 0;
%Probabilistic equation for mean (average) intensity
for i = 1:256
    meanI1 = meanI1 + ((i-1)*h(i));
end
%Probabilistic equation for mean (average) intensity
for j = 1:256
    meanI1EQUAL = meanI1EQUAL + ((j-1)*h1(j));
end
%Probabilistic equation for variance of intensities
for i1 = 1:256
    standardDEV =standardDEV + ((i1 -1 - meanI1)^2)*(h(i1));
end
%Probabilistic equation for variance of intensities
for j1 = 1:256
    standardDEVEQ = standardDEVEQ + ((j1 -1 - meanI1EQUAL)^2)*(h1(j1));
end
%Standard Dev. calculation
standardDEV = sqrt(standardDEV);
standardDEVEQ = sqrt(standardDEVEQ);
%Output of the calculation of standard dev. and mean
fprintf('The mean of image unequalized = %f, Standard of image unequalized = %f\n',↙
meanI1,standardDEV);
fprintf('The mean of image equalized = %f, Standard of image1 equalized = %f\n',↙
meanI1EQUAL,standardDEVEQ);
end
```

```
function outVector = compute_histogram(image1)
% Compute_histogram function takes in a grayscale image and returns a
% vector that represents the normalized histogram of the values in the image.
%
%
% Syntax:
% outVector = compute_histogram(image1)
%
% Input:
% image1 - grayscale image
%
% Output:
% outVector = a length 256 vector which is the normalized histogram of the
% values in the range
%
% History:
% Ralphilou S. Tatoy 3/29/22 - Created
% Ralphilou S. Tatoy 3/29/22 - Finished implementing making histogram
% Ralphilou S. Tatoy 3/30/33 - Finished commenting

%Create vector to out an histogram
outVector(1:256) = 0;
%outVector = zeros(256,1);

%Size of input image
[sizeX, sizeY] = size(image1);

%Create an histogram of the image by recording how many times a given pixel
%value is presented in an image
for row=1:sizeX
    for col=1:sizeY
        %Extracting the pixel value of the coordinates
        value = image1(row,col);
        %The indexing is different as it starts with 1 instead of 0,
        %therefore need to account for it to avoid errors
        outVector(value+1) = outVector(value+1) +1;
    end
end

%Need to do this to account of the size of the image
for vectors=1:256
        %By using the collected information of the intensitify value for
        %each pixel and calculating the percentage instead of the number of
        %times it has appeared.
        outVector(vectors) = (outVector(vectors))/(sizeX*sizeY);
end
return;
end
```

```
function plot_histogram(inVector)
% Takes an input of the histogram as a vector and plots it
%
%
% Syntax:
% plot_histogram(inVector)
%
% Input:
% inVector - histogram in a vector form returned from compute_histogram function
%
% Output:
% No output but the function plots an output graph of the histogram
%
% History:
% Ralphilou S. Tatoy 3/29/22 - Created
% Ralphilou S. Tatoy 3/29/22 - Finished implementing using bar to plot the
% histogram
% Ralphilou S. Tatoy 3/30/33 - Finished commenting

%Create histogram plot by using bar type function
figure;
bar(inVector,'red');

%x axes title
xlabel("Intensity Value");

%y axes title
ylabel("PMF");

%plot title
title("Histogram of Image");
return;
end
```

```
function vectorOUT= histogram_transform(vectorIN)
% Histogram transform takes in a histogram and returns a histogram form as
% a vector that represents a histogram equalization look up table.
%
%
% Syntax:
% vectorOUT= histogram_transform(vectorIN)
%
% Input:
% vectorIN - Histogram input of a grayscale image
%
% Output:
% vectorOUT - The equalized histogram transformation
%
%  History:
%  Ralphilou S. Tatoy 3/29/22 - Created
%  Ralphilou S. Tatoy 3/29/22 - Finished implementing histogram
%  transformation
% Ralphilou S. Tatoy 3/30/33 - Finished commenting

%Create a vector or the look up table
vectorOUT = zeros(1,256);

%Number of pixel in question
L = 256;

%For loops to create a look up table by using the histogram of the
%unequalize image
for m = 1:256
    for n = 1:m
        %Add up the probabilistic of the pixel value starting from zero to
        %the specified pixel value
        vectorOUT(m) = vectorOUT(m) + double(vectorIN(n));
    end
    %Get the pixel value that will be used for transformation process
    vectorOUT(m) = round((vectorOUT(m)*(L-1)));
end

end
```

```
>> start
Dark Image
The mean of image unequalized = 82.816412, Standard of image unequalized = 60.351960
The mean of image equalized = 128.397044, Standard of image1 equalized = 73.395873
Light Image
The mean of image unequalized = 182.023193, Standard of image unequalized = 39.149768
The mean of image equalized = 128.573807, Standard of image1 equalized = 73.813131
>>
```