Histogram Equalization

Sigi Lopez

10/22/18

Histogram Equalization is a technique that uses probability theory to accomplish a simple goal which is to make a higher contrast image. Since images are discrete (not continuous) you can find the histogram really easily by finding all probabilities of pixels from range 0-255 since it's a grey scale image. After finding all probabilities you need to distribute them equally to create your new histogram equalized image. This method works for both images that have a really dark tone and images that have a light tone. Mathematically this method does not require that much computation therefore it is very efficient. To increase efficiency, you can even create a lookup table for probabilities instead of calculating each pixel back to back.

Technical Discussion

The first step to find the histogram equalization is to calculate all the probabilities of the pixel values. Like I described in the abstract, the best way to solve for histogram equalization is to make a look up table of all of your pixel probabilities. So, to find the pixel probabilities you need to use the first equation below, M is the number of rows and N is the number of columns. Its important to add M and N since they tell you the total number of pixels that you have in your image. Now $\boldsymbol{n_k}$ is the value of the pixel you want to find, so essentially this equation gives you the average that value pixel happens in your image. Now the second equation helps you find the transformation which gives you the new values for all of your pixels. L is the maximum value a pixel can have. The sigma used in this equation helps you get all the probabilities before the value you want to get. So if you want to get pixel value 3 you need to add all probabilities up to 3 and then multiply it by max value minus 1 (The max minus one value is 255 in this problem of 256 bit pixels).

Equations

Probability Equation

$$P(r_k) = \frac{n_k}{M * N} \qquad k = 0, 1, 2, 3, \dots, L - 1$$

Transformation Equation

$$T(r_k) = (L - 1) \sum_{j=0}^{k} P(r_j) \qquad k = 0, 1, 2, 3, \dots, L - 1$$

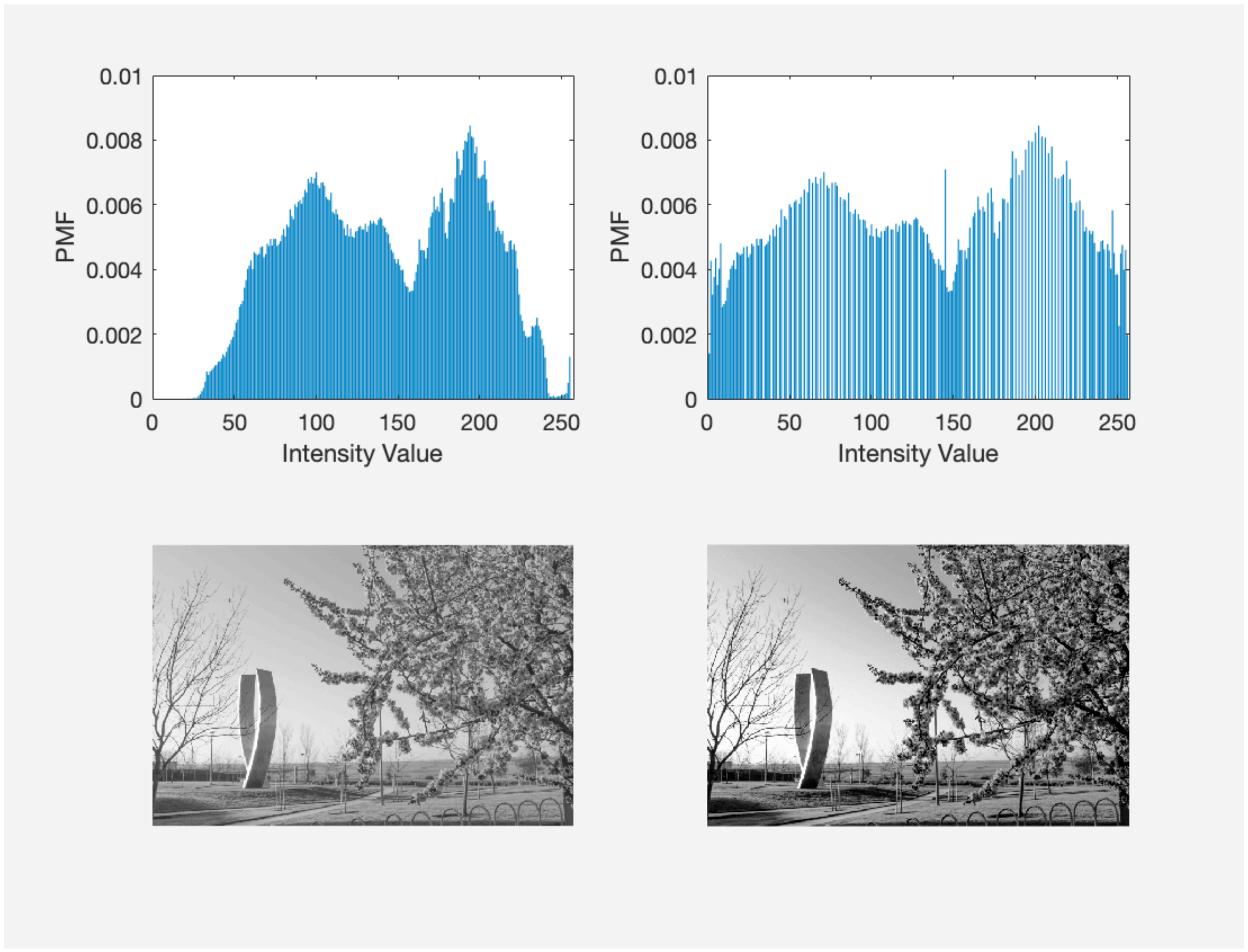
Discussion of results

For my results I ended up calculating both the mean and standard deviation for before and after histogram equalization. There were two images to test, one was dark and the other was brighter. I decided to also do histogram equalization in the original image, I was wondering if the image would stay the same or change. To begin with histogram equalization turned out to be very successful for the dark and bright images. Both dark, bright and original histogram equalized images seems like they have more detail than their original forms. Of course, there was no addition of detail but instead the details that were hidden are more defined. This higher contrast essentially shows how histogram equalization has very good applications for enhancing images. Now for the quantitative results there was some really interesting effects that happened specially with mean. For example, the dark image there was a clear understanding that the mean originally would be lower than the histogram equalized version. The opposite happened to the bright image, at first the mean was high and then it lowered. Now for standard deviation it seems that no matter what image was histogram equalized the final standard deviation would be similar to the others. This result might show that deviation is clearly controlled by the size of the image or the amount of values you can pick for a pixel.

Results

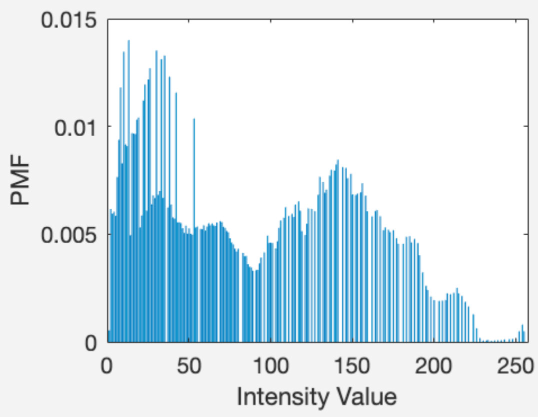
Original Image

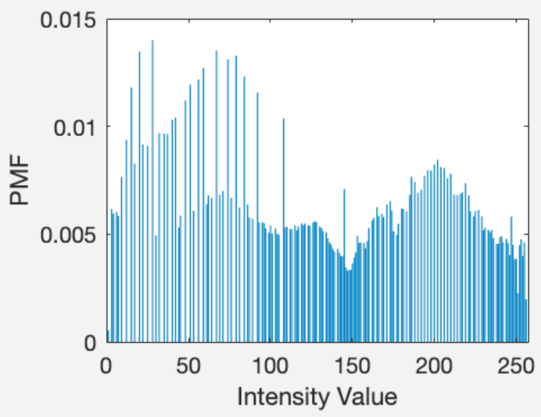
Before:	Mean = 140.8798	STD = 53.1634
After:	Mean = 128.1994	STD = 73.6326

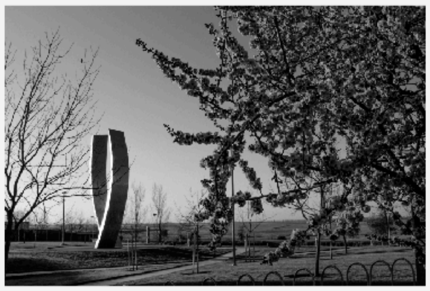


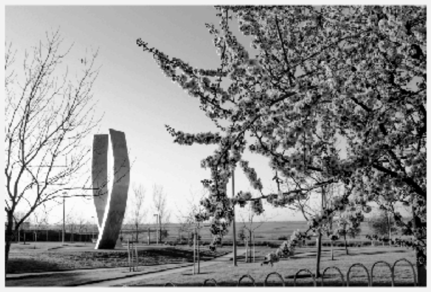
Dark Image

Before:	Mean = 82.8169	STD = 60.3535
After:	Mean = 128.3990	STD = 73.3994



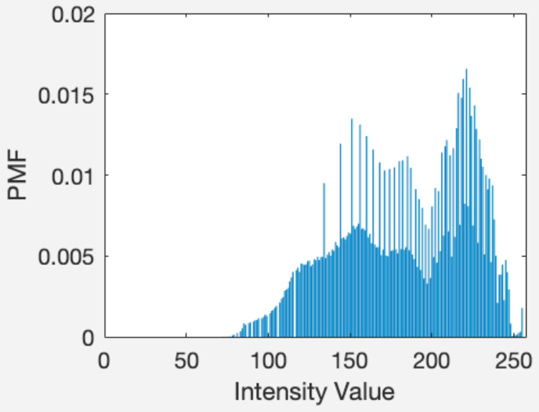


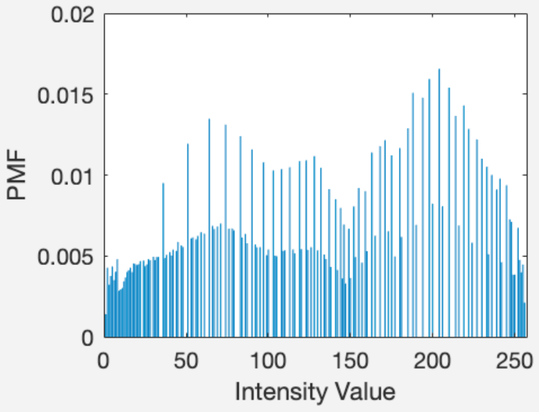


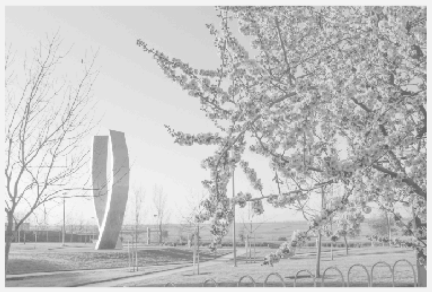


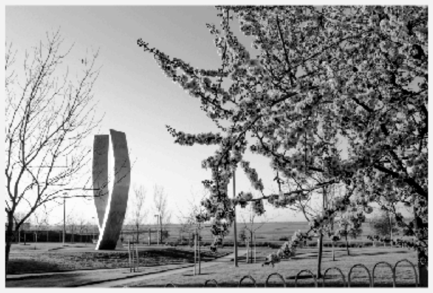
Bright Image

Before:	Mean = 182.0237	STD = 39.1508
After:	Mean = 128.5759	STD = 73.8169









Code

```
%Test
equalize(imread('Lab_03_image_original.tif'))
equalize(imread('Lab_03_image1_dark.tif'))
equalize(imread('Lab_03_image2_light.tif'))

function [histogram] = compute_histogram(image)
%Compute Histogram
%   Takes in grayscale image and outputs a vector histogram of size 256

 mn = size(image);
 m = mn(1);
 n = mn(2);

 %create empty matrix for final values
 prob = zeros(1,256);

 %calculate probabilities
 for i = 1:m
     for j = 1:n
         prob(image(i,j)+1) = prob(image(i,j)+1) + 1;
     end
 end

 for f = 1:256
         prob(f) = ((prob(f))/(m*n));
 end

 histogram = prob;
end


function [outputArg1] = plot_histogram(histogram)
%Plot Histogram
%   Takes in a histogram and plots it into a graph

 bar(histogram);

 %Graph labels
 xlabel("Intensity Value");
 ylabel("PMF");

outputArg1 = 0;
end


function [transformation] = histogram_transform(histogram)
%Histogram Transform
%   Takes in a histogram as input and returns a transformation vector 256
%   bit

%create empty space to store values
transformation = zeros(1,256);
```

```
    %Iterate Sum all probabilities before i
    for i = 1:256
        for j = 1:i
            transformation(i) = transformation(i) + histogram(j);
        end
    end

    %Multiply by total number of values
    for k = 1:256
       transformation(k) = round(transformation(k)*(255));
    end

end

function [outputArg1] = equalize(image)
%Equalize
%   Takes in as input a grayscale image 256 bits and returns the histogram
%   equalized version.

%gets index sizes
 mn = size(image);
 m = mn(1);
 n = mn(2);

%create a matrix for final processed image
finalImage = zeros(m,n);

%store probability values on a 256 vector
histogram = compute_histogram(image);

%get transformation vector that is size 256
transformation = histogram_transform(compute_histogram(image));

%plot calculated values to final image
for i = 1:m
    for j = 1:n
       finalImage(i,j) = transformation(image(i,j)+1);
    end
end

%calculate histogram of final transformed image
histogram2 = compute_histogram(finalImage);

finalImage = uint8(finalImage);

%PLOT EVERYTHING
figure;
subplot(2,2,1),plot_histogram(histogram);
subplot(2,2,2),plot_histogram(histogram2);

subplot(2,2,3),imshow(image);
subplot(2,2,4),imshow(finalImage);

image = double(image);
```

```
finalImage = double(finalImage);

%PRINT MEAN AND STANDARD DEVIATION

disp("Mean Origianl")
disp(mean(image(:)))

disp("Standard Deviation Origianl")
disp(std(image(:)))

disp("Mean Equalized")
disp(mean(finalImage(:)))

disp("Standard Deviation Equalized")
disp(std(finalImage(:)))

outputArg1 = 0;

end
```