COSC 190: HW#2
Retinex: Histogram Equalization Part 2
Instructor: Dr. Fred Park

1. Write a function called 'my_histcounts' that takes in an image or array f along with the bin centers B and returns a vector of the same length as B of the histogram counts. Apply to a small synthetic data-set of you choice. Show that this works by manually computing by hand and showing that it coincides with the result from your algorithm. Use the 'bar' plot command to view the histogram of your data. Apply it to an image and repeat. This should get you started using the bar plotting:
```
f = [0 1 1 2 2 2 3 3 4]; %generate data
B = [0:1:4] %bin centers
hcf = my_histcounts(f,B); %calculate hist counts
figure(1); bar(B,hcf); title('Histograms for Original Image');
xlabel('bin centers'); ylabel('counts');
```

2. Write a function called 'my_histeq' that takes in an image or array of data and processes the data by equalizing the associated histogram to said data by the method outlined in class. It should model the following:

$$\begin{aligned} s_k &= T(r_k) && (1) \\ &= (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j && (2) \end{aligned}$$

for $k = 0, \ldots, L-1$ and $n_j :=$ the number of pixels having value r_j, $0 \leq r_j \leq L-1$. For a uint8 image, L = 256. For our purposes, you will need to modify the equation above to:

$$s_k = T(r_k) = \text{round}\left(\frac{(L-1)}{MN}\sum_{j=0}^{k} n_j\right), \qquad \text{for } k = 0, \ldots, L-1. \quad (3)$$

Here, "round" denotes the rounding operator in matlab.

3. In a separate script called 'uniform_dist_test.m', create a vector of values normally distributed. Apply the histogram equalization method discussed in class and show that when you apply the equalization to this set of data points, you obtain a (nearly) uniform distribution back.

4. In another script called 'gauss_dist_test.m', create a 25×25 image of normally distributed pixels taking values between [0, 255] with mean 128 and standard deviation 10 using the matlab 'randn' function. Equalize it. This code should get you started:
```
mn = 128; %mean of synthetic data
stdv = 10; %standard deviation of synthetic data
g = uint8(mn + stdv.*randn(25,25)); %uint8 rounds back to ints from 0 to 255
```

5. In another script called 'hist_equalize_test.m', test your histogram equalization on the 'chapel3.jpg', 'landscape' and 'factory.jpg' images found on my website.

6. Take a selfie in low light, load it into matlab, and equalize the histogram. Are you happy with the results? Why or why not?

7. For what types of images and their associated histograms does this particular method work the best on? Explain?

8. Compare this method to the previous linear mapping method. Which one works better and why? What are the advantages or disadvantages of each method? Can you think of an even better one?

9. How can a grayscale image be represented in the RGB color space? Give an example of one such. Take the cameraman.tif image from my website and find a way to embed this image into the RGB colorspace, then view it as a color image.

10. What are the most salient features in an image? List some of them. What is the single most? Can you think of a way to extract them?

11. How can you find the jumps in an image? Is there a way to extract them consistently?

12. Consider the following notation for an image f that is grayscale:

$$f_x(i,j) \approx f(i+1,j) - f(i,j) \; := \; \text{the jump in the x-direction} \tag{4}$$
$$f_y(i,j) \approx f(i,j+1) - f(i,j) \; := \; \text{the jump in the y-direction.} \tag{5}$$

Let us define the discrete gradient of an image f in the following manner:

$$\nabla f(i,j) := \langle f_x(i,j), f_y(i,j) \rangle. \tag{6}$$

The magnitude of the discrete gradient vector associated to f is defined as the following:

$$|\nabla f(i,j)| := |\langle f_x(i,j), f_y(i,j) \rangle| = \sqrt{(f_x(i,j))^2 + (f_y(i,j))^2}. \tag{7}$$

By staring at the above equation (7) for an extended period of time, one sees that the jump information in both the x-direction and y-direction is embedded into the magnitude of the gradient. Does this take into account a positive or negative jump? Does it discriminate? Write out an explicit example.

13. From the discrete gradient defined in 12. construct an edge detector that has value close to 1 near an edge and 0 away from it. Test it on the cameraman image and statue image.