Assignment 1

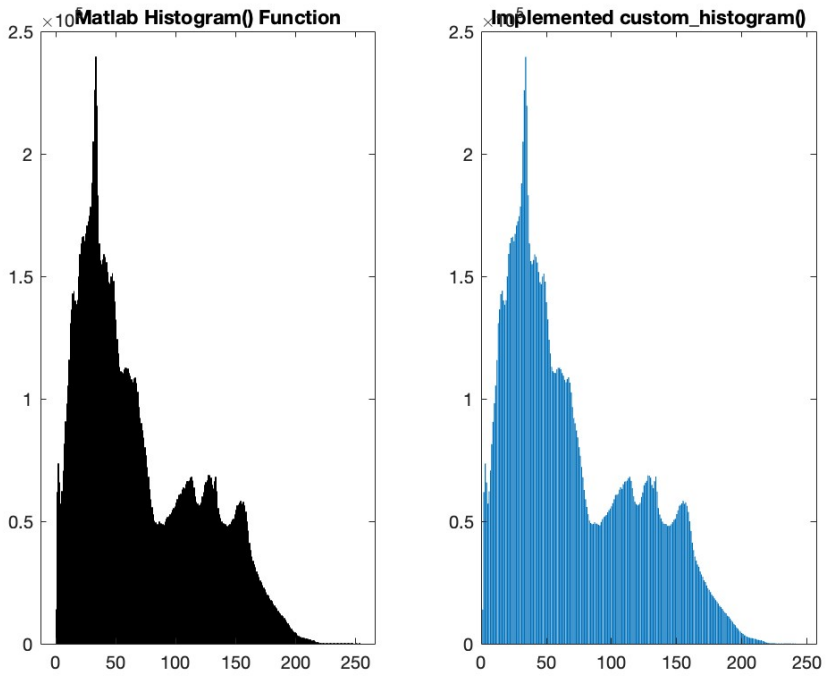
CS 615 - Digital Image Processing

Spencer Au

Main Function to Run A and B

```
img = imread('assets/ada_cat.JPG');

grayscale = rgb2gray(img);
colormap(gray)

subplot(1, 2, 1), histogram(grayscale)
title('Matlab Histogram() Function');
subplot(1, 2, 2), custom_histogram(grayscale)
title("Implemented custom\_histogram()");

% image_threshold(img)
```

Part A: Histogram




```
function [outputArg1, min_output, max_output, output_img] = custom_histogram(inputIMG)
%CUSTOM_HISTOGRAM Summary of this function goes here
%   Detailed explanation goes here

% grayscale the image if necessary
if size(inputIMG, 3) == 3
    grayscale = rgb2gray(inputIMG);
else
    grayscale = inputIMG;
end

colormap(gray)

% grab the size of the image
[height, width] = size(grayscale);

% lets use this instead since i want an array of counts of each graylevel,
% where the index is the respective gray level, so if gLC[5] = 10, that
% means there is 10 counts of a graylevel of 5 in the image
grayLevelCounts = zeros(1, 256);

min = 256;
```

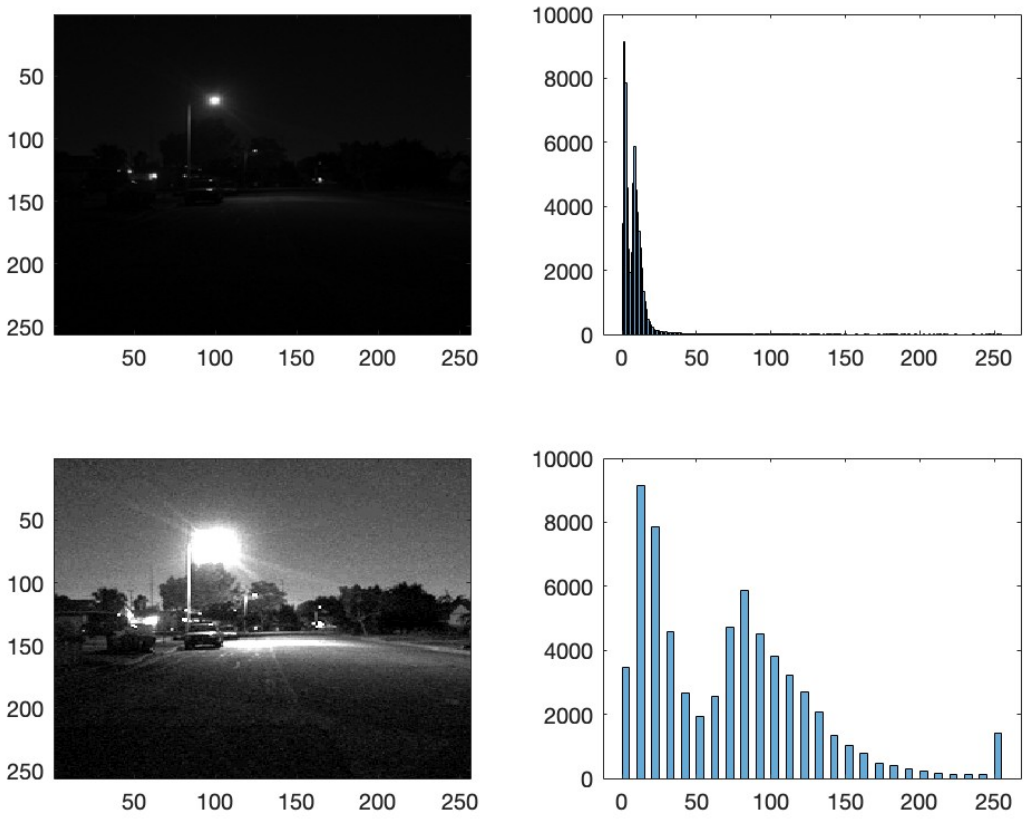
```
max = -1;

% iterate through image and
% Gray Level 0 corresponds to gLC(1)
for i = 1:height
    for j = 1:width
        grayValue = grayscale(i, j) + 1;
        % can't use += in MATLAB
        grayLevelCounts(grayValue) = grayLevelCounts(grayValue) + 1;
        if grayValue > max
            max = grayValue;
        elseif grayValue < min
            min = grayValue;
        end
    end
end

bar(grayLevelCounts)

outputArg1 = grayLevelCounts;
min_output = min;
max_output = max;
output_img = grayscale;

end
```

Part B: Image Thresholding

```
function image_threshold(inputImage)

[grayLevelCounts, minGray, maxGray, image] = custom_histogram(inputImage);

% get the midpoint of the min and max graylevel
midpoint = (minGray + maxGray)/2;
fprintf('Midpoint: %d\n', midpoint);

% prob need to change this to what the initial should be
t0 = 1;

t = midpoint;

diff = 999;
grayLevels = 0:255;
iter = 1;

while diff > t0
    fprintf("Running Iteration %d\n", iter);

    % need to cast t as an int and clamp it between [1, 255]
    t = int32(round(t));
    t = max(1, min(t, length(grayLevelCounts) - 1));

    bw = (image >= t);

    % use .* which does element wise multiplication

    % white region
    g1_list = grayLevelCounts(t+1:end);
    g1_total = sum(g1_list);
    g1 = sum(grayLevels(t+1:end) .* g1_list)/g1_total;

    % black region
    g2_list = grayLevelCounts(1:t);
    g2_total = sum(g2_list);
    g2 = sum(grayLevels(1:t) .* g2_list)/g2_total;

    % calculate new threshold value
    t_new = (g1 + g2)/2;

    diff = abs(t_new - t);
    t = t_new;
```

```
    fprintf('diff: %d\n', diff);

    iter = iter + 1;

end

% display both images side by side
imshowpair(image, bw, 'montage');
title('Original (Left) vs Thresholded B/W (Right)');

end
```

Part C: Full Scale Contrast Stretching




```
img = imread('assets/spot.jpg');

subplot(2, 2, 1), imagesc(img)
```

```
subplot(2, 2, 2), histogram(img)

% A is the global minimum gray level
A = min(min(img));
% B is the global maximum gray level
% 25 might be better since theres like one or two occurences of 255, etc
B = 25;

% K is the maximum number of gray levels; since the image is 8 bit, that
% corresponds to 256 GL
K = 256;

% equation to use is J(i,j) = (K - 1)/(B - A) * [I(i,j) - A]
scaled_img = img;
for i = 1:256
    for j = 1:256
        % above is kind of janky; just gonna do one line
        scaled_img(i, j) = ((K - 1)/(B - A)) * (img(i, j) - A);
    end
end

colormap(gray)

subplot(2, 2, 3), imagesc(scaled_img)
subplot(2, 2, 4), histogram(scaled_img)
```

Part D: Fading of Images

Running the actual program will generate a "video" that shows the fading effect. I simply showed it as a montage of the intermediate images for simplicity.

```
ada_orig = imread('assets/ada_kitchen.JPG');
hux_orig = imread('assets/huxley.JPG');
```

```
ada_orig = rot90(ada_orig, -1);
hux_orig = rot90(hux_orig, -1);

% this will be the original image
gray_ada = rgb2gray(ada_orig);
% this will be the output after fading
gray_hux = rgb2gray(hux_orig);
colormap(gray)

% define the number of intermediate images (should prob change to funct)
num_int_img = 20;
[height, width] = size(gray_ada);
% set an array of images so i can display side by side
int_image_array = cell(1, num_int_img + 2);
int_image_array{1} = gray_ada;

imshow(gray_ada)

for x = 1:num_int_img
    int_img = gray_ada;
    for i = 1:height
        for j = 1:width
            % generate intermediate image at x
            dist = double(gray_hux(i, j)) - double(gray_ada(i, j)); % NOT uint8
            single_dist = dist / num_int_img;
            int_img(i, j) = int_img(i, j) + (single_dist * x);
        end
    end
    %drawnow()
    int_image_array{x + 1} = int_img;
    drawnow()
    imshow(int_img)
end

%imshow(gray_hux)

%int_image_array{num_int_img + 2} = gray_hux;

%montage(int_image_array, 'Size', [1, numel(int_image_array)]);
```