ECE 253: HW2

Academic Integrity Policy: Integrity of scholarship is essential for an academic community. The University expects that both faculty and students will honor this principle and

in so doing protect the validity of University intellectual work. For students, this means that all academic work will be done by the individual to whom it is assigned, without unauthorized aid of any kind.

By including this in my report, I agree to abide by the Academic Integrity Policy mentioned above.

****

To run all files at once, load folder named “main” into Current Folder section in MATLAB(This contains all scripts as well as all images used). Then run “main” in the MATLAB console. This file will output results for the whole HW. To run each section individually, follow directions in each section below.

****

Ajinkya Bagde
U07800884
11/4/2017

Problem 1: Adaptive Histogram Equalization

To run:
beach = imread('beach.png');
AHE(beach,<win_size>)
%%%Takes some time for win_size 129!%%%

Output Below(original image and MATLAB HE are constant for thumbnails, just included for visual comparison):

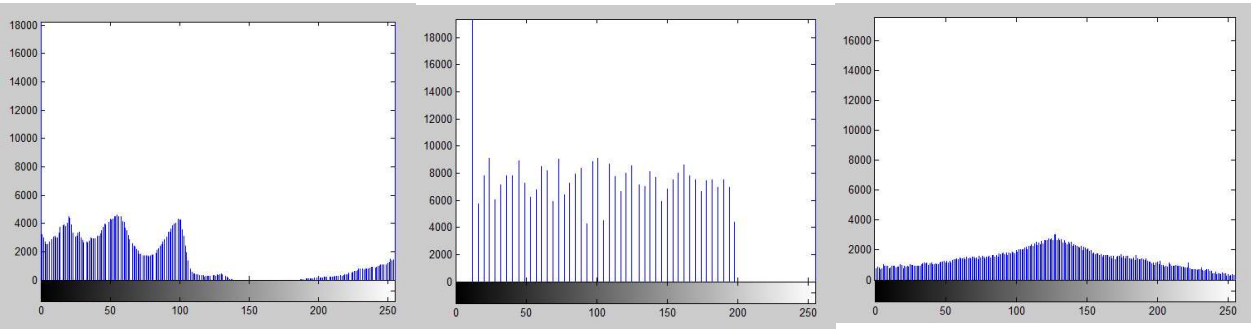
win_size = 33

win_size=65

win_size=129

1. How does the original image qualitatively compare to the images after AHE and HE respectively?
    - Assuming the final goal was to enhance the contrast of the underexposed original image, we see that from the image results, that the contrast has been increased(histograms stretched) for both AHE and HE. However, for both HE and AHE, some parts of the image look unrealistic compared to the original - for example, take a look at the 'sky' in the original image, image after AHE(any window size), and image after MATLAB's HE. On the other hand, darker regions such as the pillars underneath the house have greater contrast after both AHE and HE.

Original Hist *Hist after HE* *Hist after AHE(65)*



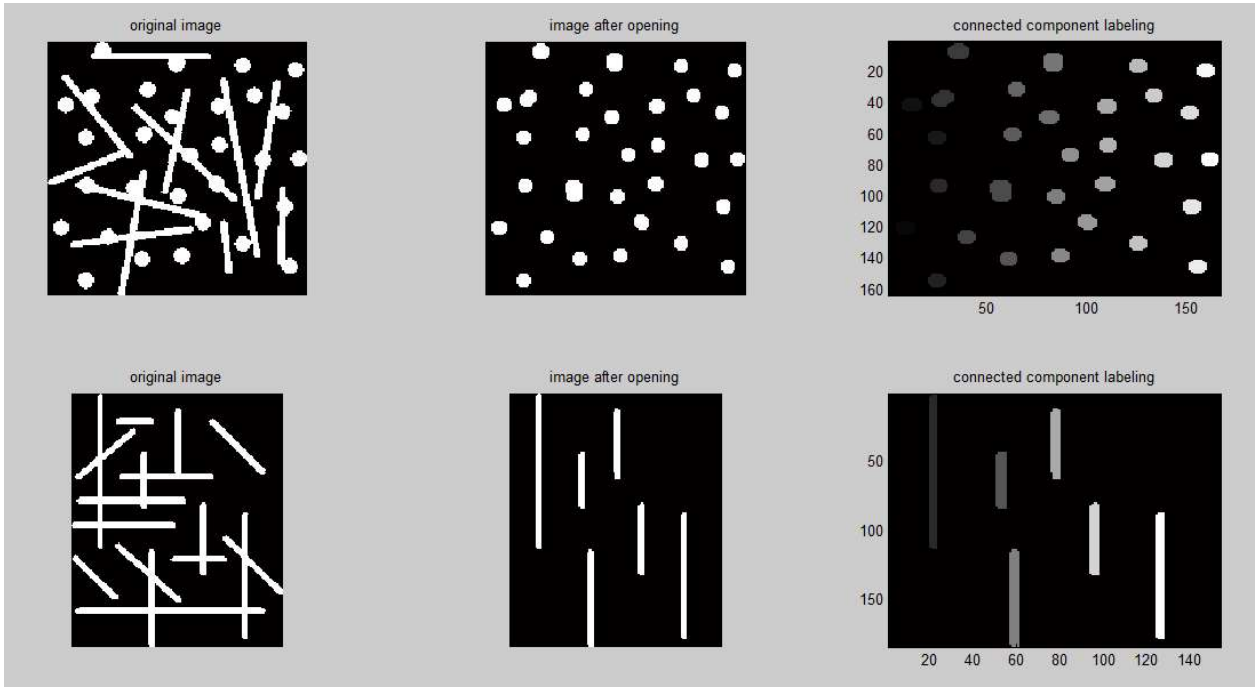
2. Which strategy(AHE or HE) works best for *beach.png* and why? Is this true for any image in general?
    - For this image, HE seems to work the best. This is because the original image as a whole has low contrast, we see most pixels are located between values of 0 → 130. HE works best when the contrast of the image of a whole has to be increased. AHE provides better contrast in local areas compared to HE - if we had an image whose histogram covered the entire spectrum of intensity values, then AHE would increase contrast in localized areas, but not the image as a whole.

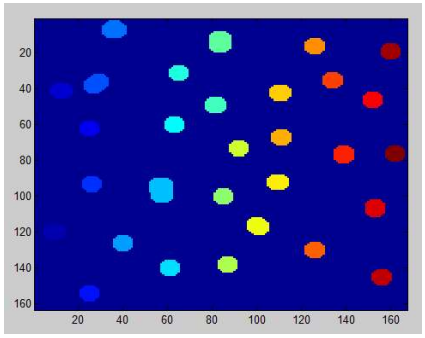
```
%Problem 1: Adaptive Histogram Equalization
function enhanced_image = AHE(input_image,win_size)
%Pad input image based on window size, so conextual region for edge pixels
%remains valid. Window size in always MxM, where M is odd.
%Window will always be contered on pixel of interest. This means that the
%padding on each size has to be exactly (M-1)/2
pad = (win_size -1)/2;
padded_image = padarray(input_image,[pad pad],'symmetric');

[rows,cols] = size(input_image);

%Allocate enhanced_image matrix
enhanced_image = zeros(rows,cols);

for x=1:rows
   for y=1:cols
      rank = 0;

      %current pixel in terms of the padded matrix(offset)
      current_pixel = padded_image(x+pad,y+pad);
      for i = -pad:1:pad
         for j = -pad:1:pad
            if padded_image(x+pad+i,y+pad+j) > current_pixel
               rank = rank+1;
            end
         end
      end
      output_pixel = rank*255/(win_size*win_size);
      enhanced_image(x,y) = output_pixel;
   end
end

test_image = histeq(input_image);

subplot(1,3,1)
imshow(input_image)
title('original image')
subplot(1,3,2)
imshow(enhanced_image, [])
title('enhanced image - AHE')
subplot(1,3,3)
imshow(test_image)
title('MATLAB HE')
```

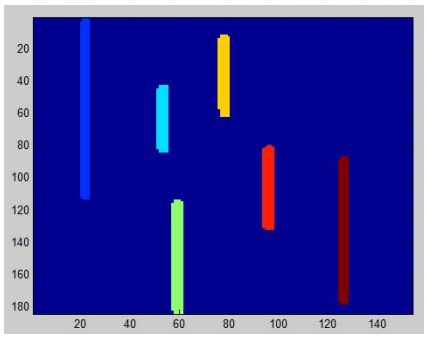
Problem 2: Binary Morphology

To run:
To run, type “prob2” into MATLAB console, make sure both JPEGS are in current workspace

Structuring Elements used:
Part 1 - strel('disk',5)
Part 2 - strel('line',8,90)







Info for Part 1:

Connected Component	Centroid	Area
1	120.000000, 9.500000	78
2	41.000000, 12.500000	78
3	62.000000, 25.000000	69
4	154.000000, 25.000000	69
5	93.000000, 26.000000	69
6	37.000000, 28.000000	95
7	6.500000, 36.000000	92
8	126.000000, 40.000000	69
9	96.500000, 57.000000	136
10	140.000000, 61.000000	69
11	60.000000, 63.000000	69
12	31.000000, 65.000000	69
13	49.000000, 81.500000	78
14	13.500000, 83.500000	108
15	100.000000, 85.000000	69
16	138.000000, 87.000000	69
17	73.000000, 92.000000	69
18	116.500000, 100.500000	82
19	92.000000, 109.500000	78
20	42.000000, 110.500000	78
21	67.000000, 111.000000	69
22	16.000000, 126.000000	69
23	130.000000, 126.000000	69

24	35.000000, 134.000000	69
25	76.500000, 139.000000	78
26	46.000000, 152.000000	69
27	106.500000, 153.000000	78
28	145.000000, 156.000000	69
29	19.000000, 160.000000	69
30	76.000000, 162.000000	69

Info for Part 2:

Connected Component	**Centroid**	**Length**
1	57.000000, 21.500000	112
2	63.500000, 53.000000	41
3	149.000000, 59.000000	70
4	37.000000, 78.000000	50
5	106.000000, 96.000000	52
6	132.500000, 126.500000	91

```
%Problem 2 - Binary Morphology
clc;
clear;

%Part1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
cl = imread('circles_lines.jpg');
cl = im2bw(cl);

%First step is to remove lines from the image
se = strel('disk',5);
cl_open = imopen(cl, se);

subplot(2,3,1)
imshow(cl)
title('original image')
subplot(2,3,2)
imshow(cl_open)
title('image after opening')

%Now do the connected component labeling
%We get 30, which is the number of circles
labels = bwlabel(cl_open,4);
labels1 = labels;
subplot(2,3,3)
imagesc(labels)
title('connected component labeling')



%Find number of connected components
number_components = max(max(labels));

%Now find info on each component
for x = 1:number_components

    %Find all rows/columns that contain an element of the component
    [row, col] = find(labels==x);

    %Area is just instances of that component in the overall matrix
    fprintf('Area of component %d is: %d\n',x,length(row))

    %Find dimensions of bounding box that contains the component
```

```
    min_row = min(row);
    max_row = max(row);
    min_col = min(col);
    max_col = max(col);

    cent_row = (max_row + min_row)/2;
    cent_col = (max_col + min_col)/2;

    fprintf('Component %d centroid located at:\n',x)
    fprintf('%f, %f\n', [cent_row,cent_col]')
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Part2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
cl = imread('lines.jpg');
cl = im2bw(cl);

%First step is to remove lines from the image
se = strel('line',8,90);
cl_open = imopen(cl, se);

subplot(2,3,4)
imshow(cl)
title('original image')
subplot(2,3,5)
imshow(cl_open)
title('image after opening')

%Now do the connected component labeling
%We get 30, which is the number of circles
labels = bwlabel(cl_open,4);
labels2 = labels;
subplot(2,3,6)
imagesc(labels)
title('connected component labeling')

%Find number of connected components
number_components = max(max(labels));
```

```
%Now find info on each component
for x = 1:number_components

    %Find all rows/columns that contain an element of the component
    [row, col] = find(labels==x);


    %Find dimensions of bounding box that contains the component
    min_row = min(row);
    max_row = max(row);
    min_col = min(col);
    max_col = max(col);

    cent_row = (max_row + min_row)/2;
    cent_col = (max_col + min_col)/2;

    %Length is just max_row-min_row
    fprintf('Length of component %d is: %d\n',x,max_row-min_row)


    fprintf('Component %d centroid located at:\n',x)
    fprintf('%f, %f\n', [cent_row,cent_col]')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%


figure
imagesc(labels1)

figure
imagesc(labels2)
```

Problem 3: Lloyd-Max Quantizer

To run:
Make sure QUANT_MSE and myQuantize functions are in the current folder, as well as images

lena = imread('lena512.tif');
QUANT_MSE(lena)
lena = histeq(lena,256);
QUANT_MSE(lena)
%%Similar process for 'diver.tif'

Part 1:

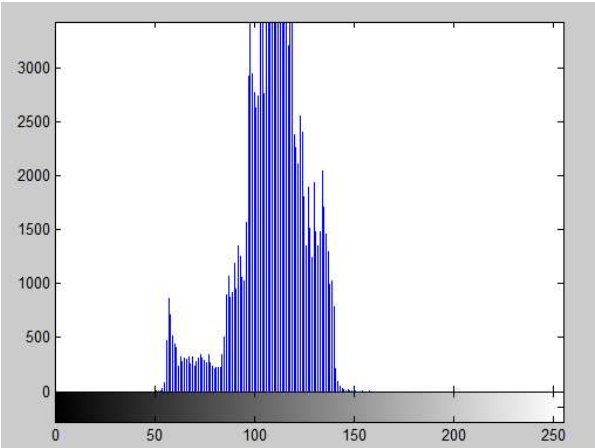
For this part, I wrote the myQuantize function. It takes in an image and a quantize bit, and returns the quantized image(1-7 bit, depending on input).

Part 2:

Regarding 'diver.tif':

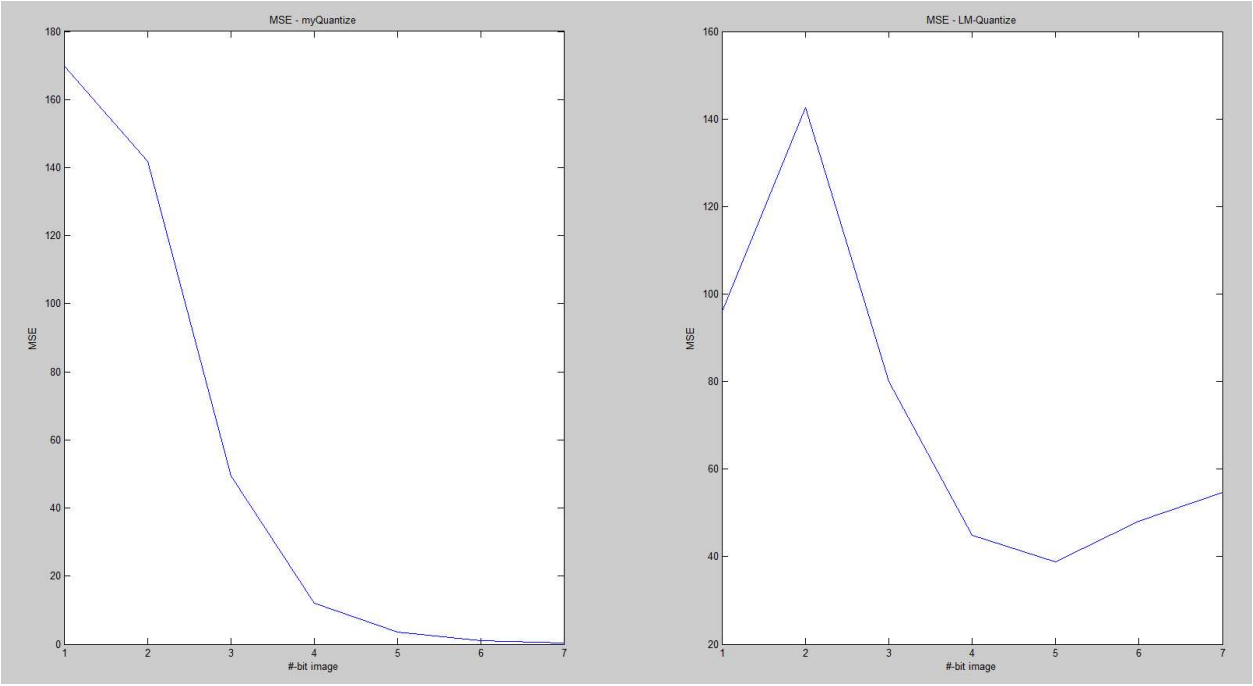
Original image

Original Histogram



Quantized Images

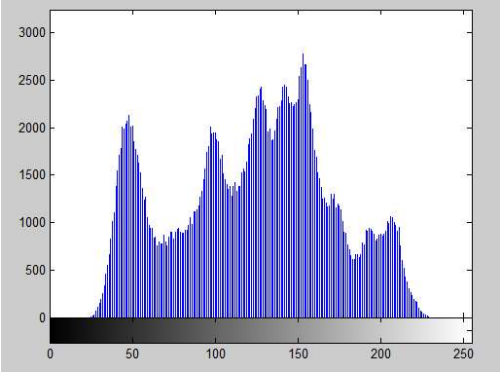
MSE



Regarding ‘lena512.tif’:

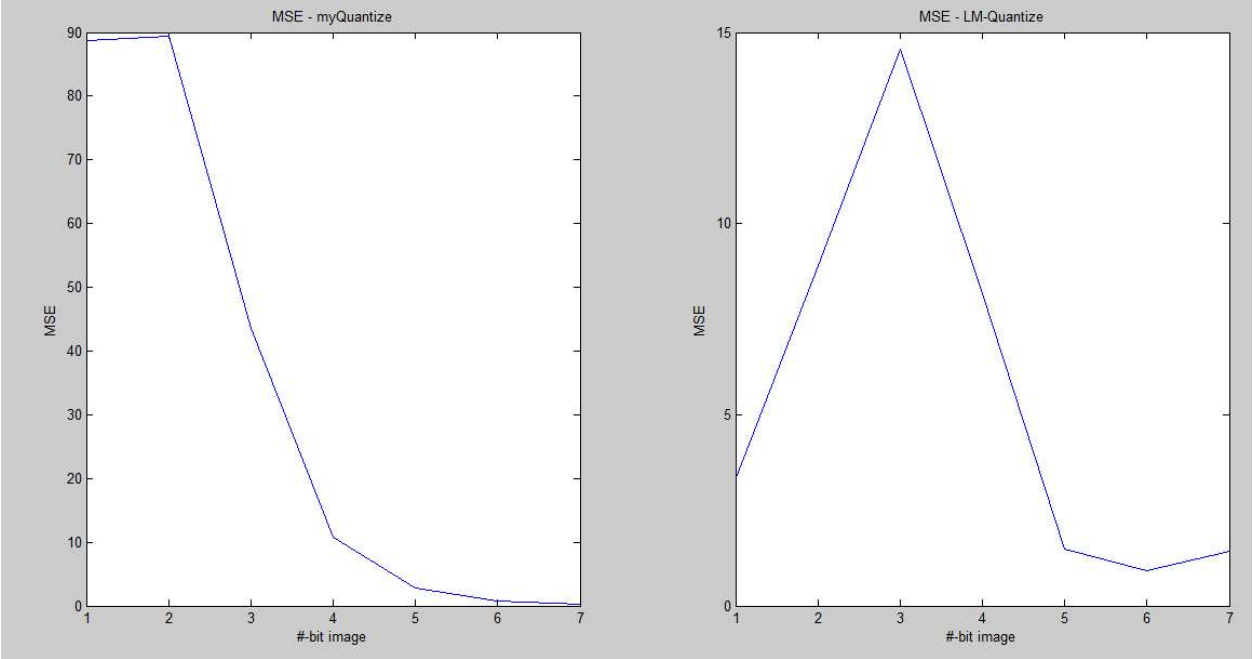
Original Image:

Original Histogram:



Quantized Images

MSE



In general, the LM Quantizer outperforms the standard quantizer - it seeks to find the optimum decision levels and reconstruction levels so that MSE is minimized. What this means is that the quantization levels are set based on local behavior of the image - slowly changing regions are finely quantized, and rapidly changing regions are coarsely quantized(from text). Therefore in theory, since LM quantization is not based on a standard unit step like myQuantize, the MSE for each level should be less than it would be for standard quantization.

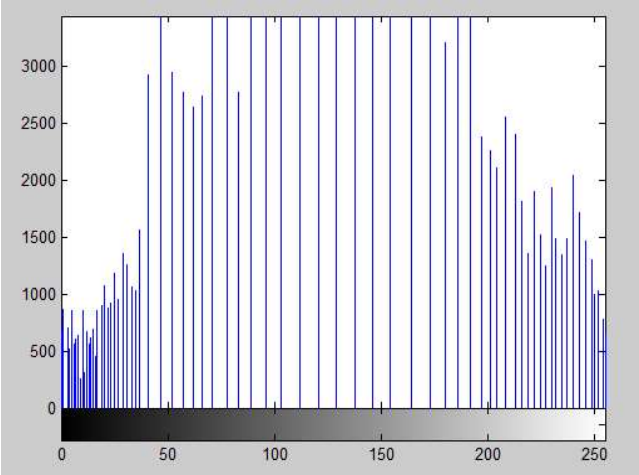
In this specific case, the LM Quantizer outperforms the standard quantizer for 'lena512.tif' by a large margin(~1 order of magnitude). If we take a look at the histogram of the original 'lena' image, we see that it has high contrast, and a decent exposure. The LM Quantizer outperforms because there is a lot of local behavior to optimize from. Now for the 'diver.tif' image, we see that the LM Quantizer does not perform so great. The histogram of this image is low contrast, but has a decent exposure. The LM Quantizer does not have a lot of local behavior to optimize from, and therefore performs almost no better than the standard quantizer. For lower bit values, the MSE is lower by around 30% compared to the standard quantizer, but the standard quantizer actually outperforms for higher bit values. I actually ran a test where I quantized up to a 12-bit image, and at that point the MSE's from both quantizers were nearly identical, around 0.

Part 3:

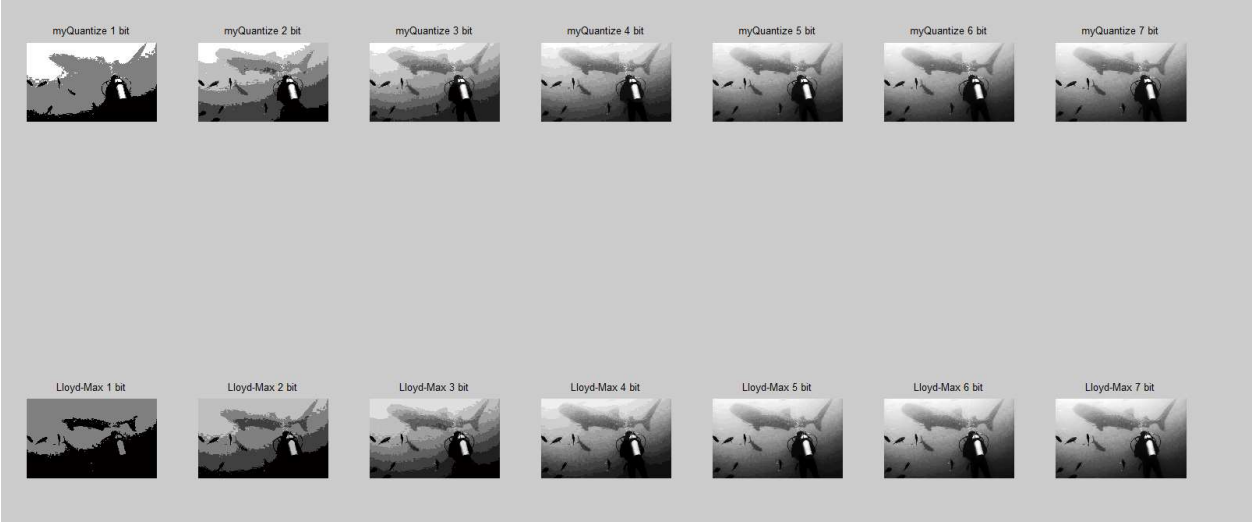
Regarding ‘diver.tif’:

Diver HEQ

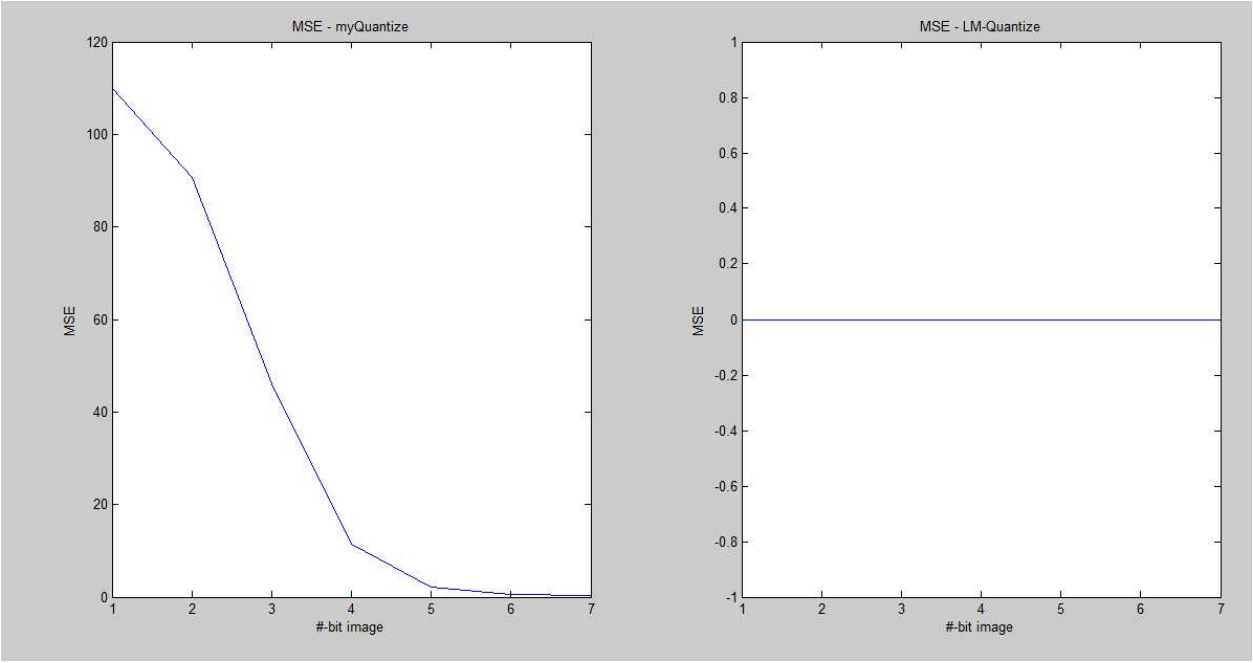
Diver HEQ Histogram



Quantized Images HEQ



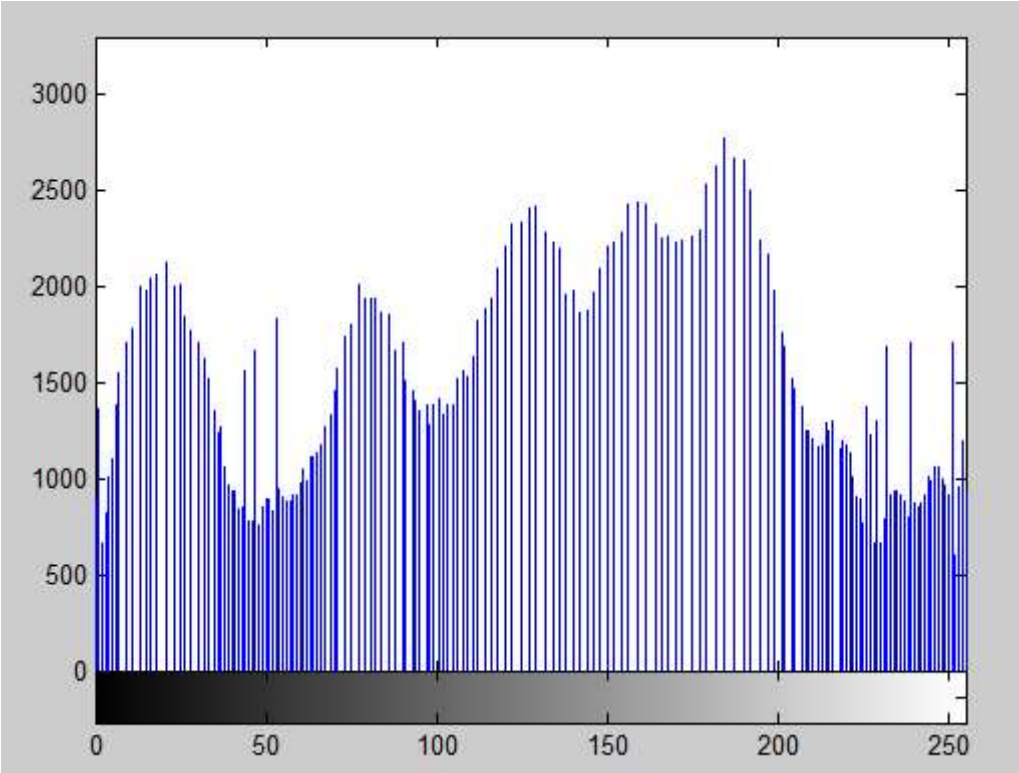
MSE HEQ



Regarding ‘lena512.tif’:

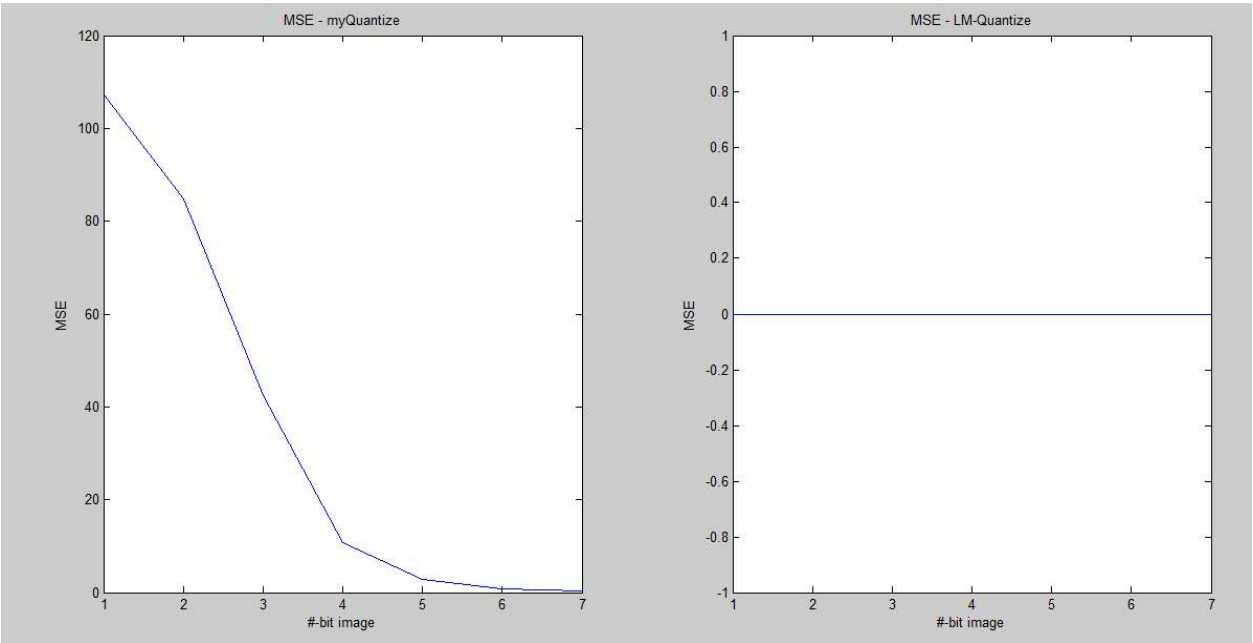
Original Image HEQ

Lena HEQ Histogram



Quantized Images HEQ

MSE HEQ



Comparing the statistics of the HEQ's images to the orginal, the MSE has changed. For standard quantization, the slopes of the MSE curve as number of bit's increase remained relatively the same. For 'diver.tif', the starting MSE for a one bit image has decreased from 170 to 110. This makes sense since the histogram got stretched. For 'lena512.tif', the results are very similar to the original image. This makes sense looking at the histograms - they have barely changed.

For the LM quantizer, the MSE has gone down to 0 for all bits, and for both images. The reasoning behind this is explained in Part 4.

Part 4

If the histogram of the equalized histogram was truly uniform, the uniform quantizer would theoretically also give an MSE of 0. However, this is never the case - looking at the histograms of both equalized images, we see that they are not uniform. The Lloyd-Max quantizer has the degree of freedom to define the decision and reconstruction levels, and effectively these are defined in a way that minimizes the mean-square quantization error. This means that the quantization function does not have a unit step, unlike the uniform quantizer, and quantized step is optimized. This optimization guarantees that the MSE is zero.

```
%Quantization function for Question3

function[quantized_image] = myQuantize(input_gs_image, N)

    %First thing is to convert input bit value to # of bins
    %2^n
    N = pow2(N);

    %Convert input greyscale image to double
    image_d = double(input_gs_image);

    %Reshaping the image into an array
    image_size = size(image_d);
    image_array = reshape(image_d, prod(image_size), 1);

    %Quantization Parameters
    total_range = 255;
    quantization_width = total_range/N;

    quantization_value = zeros(N+1,1);

    %Perform Quantization
    for  i = 1:N+1
        quantization_value(i,1) = quantization_width*(i-1);
    end

    difference_array = zeros(N,1);

    for i = 1:1:length(image_array)
        for j = 1:1:N+1

            difference_array(j,1) = abs(image_array(i,1) - quantization_value(j,1));
        end
         [~, index] = min(difference_array);
         image_array(i,1) = quantization_value(index,1);
    end




    quantized_image = reshape(image_array, image_size(1), image_size(2));
    quantized_image = uint8(quantized_image);
```

```
%Problem 3 - Lloyd-Max Quantizer

function MSE = QUANT_MSE(im)

%%Part1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
%Defined seperate function myQuantize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%Part2 and
Part3%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%

%Reshape image in order to perform Lloyd-Max quantizing
[M,N] = size(im);
training_set = reshape(im,N*M,1);

%Preallocate MSE arrays
MSE_myQ_orig = [];
MSE_LM_orig = [];

for s = 1:1:7

    %Performing uniform quantization using myQuantize function
    im_quantized = myQuantize(im, s);

    %Find MSE between myQuantize'd and original image
    MSE = sum((sum(((im_quantized - im).^2))))/numel(im);
    MSE_myQ_orig = [MSE_myQ_orig, MSE];

    %Performing Lloyd Max quantization
    len = 2.^s;
    training_set = double(training_set);
    [partition, codebook] = lloyds(training_set, len);
    [im_lloyd ,index] = quantiz(training_set,partition,codebook);
    im_lloyd = (im_lloyd/len)*255;
    im_lloyd = reshape(im_lloyd,[M,N]);
    im_lloyd = uint8(im_lloyd);

    %Find MSE between LM-Quantize'd and original image
    MSE = sum((sum(((im_lloyd - im).^2))))/numel(im);
    MSE_LM_orig = [MSE_LM_orig, MSE];
```

```
    %Show images at s-bit quantization levels
    subplot(2,7,s)
    imshow(im_quantized)
    title(sprintf('myQuantize %d bit', s))


    subplot(2,7,s+7)
    imshow(im_lloyd)
    title(sprintf('Lloyd-Max %d bit', s))
end

%Plot MSE vs bits
figure
subplot(1,2,1)
plot(MSE_myQ_orig)
title('MSE - myQuantize')
xlabel('#-bit image')
ylabel('MSE')

subplot(1,2,2)
plot(MSE_LM_orig)
title('MSE - LM-Quantize')
xlabel('#-bit image')
ylabel('MSE')



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```