Student: Melis Kilic

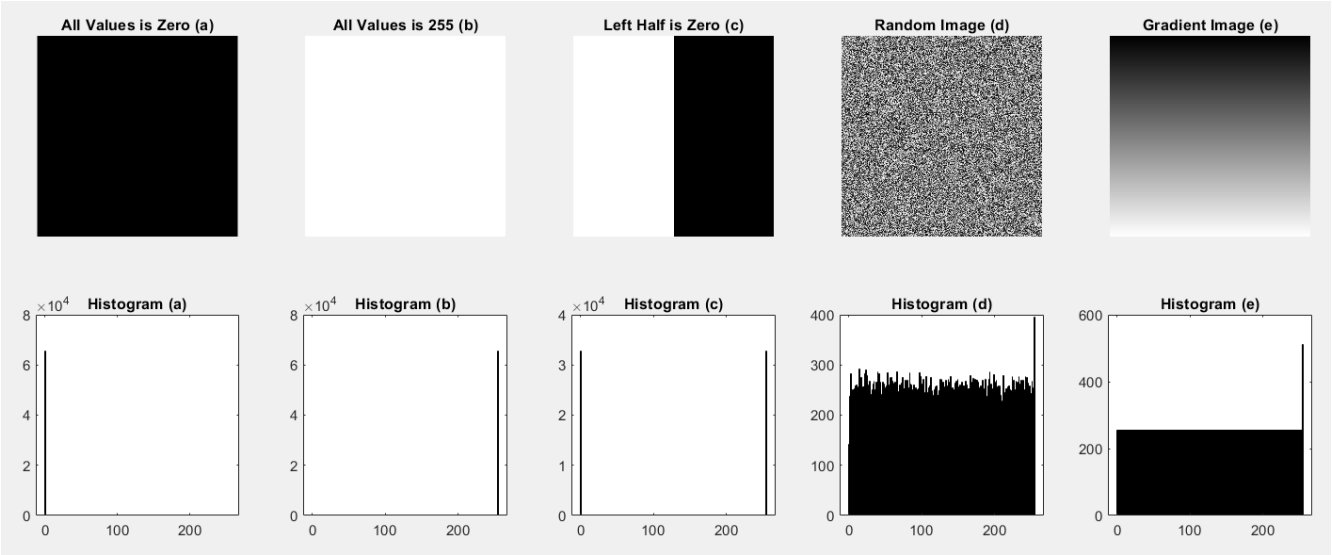
Histograms

H-1. Histograms of artificial images

```
clearvars;
close all;
clc;

% Create 256x256 images with:

% a) All pixels with value 0
all_zero = zeros(256,256);

% b) All pixels with value 255
all_255 = 255 * ones(256,256);

% c) Left half of the image with values 0 and the right half with values 255
left_half_zero = zeros(256,256);
left_half_zero (: , 1 : 256/2) = 255;

% d) Random image with pixels values from uniform distribution between 0 and 255
% (use function rand)
random_image = uint8 (255 * rand(255,255));

% e) With fluently changing grayscale from 0 (left side of the image) to 255
% (right side of the image). To the input image use matrix multiplication of
% column vector with ones with row vector [0:255].

gradient_vector = (0:255)';
gradient_image = uint8(gradient_vector * ones(1,256));

% When displaying images with imshow, either convert them to uint8 using
% Matlab command uint8(…) or usescaling: imshow(…, []).

figure('position', [100, 100, 1400, 500]);

subplot(2,5,1);
imshow(all_zero, []);
title('All Values is Zero (a)');

subplot(2,5,2);
imshow(all_255, []);
title('All Values is 255 (b)');

subplot(2,5,3);
imshow(left_half_zero, []);
title('Left Half is Zero (c)');

subplot(2,5,4);
imshow(random_image, []);
title('Random Image (d)');

subplot(2,5,5);
imshow(gradient_image, []);
title('Gradient Image (e)');
```

```
% Then calculate them using histogram(I, 0:256), where I is the image and
% the second argument are bin edges.

subplot(2,5,6);
histogram(all_zero(:), 0:255);
title('Histogram (a)');

subplot(2,5,7);
histogram(all_255(:), 0:255);
title('Histogram (b)');

subplot(2,5,8);
histogram(left_half_zero(:), 0:255);
title('Histogram (c)');

subplot(2,5,9);
histogram(random_image(:), 0:255);
title('Histogram (d)');

subplot(2,5,10);
histogram(gradient_image(:), 0:255);
title('Histogram (e)');
```

Result of the code:

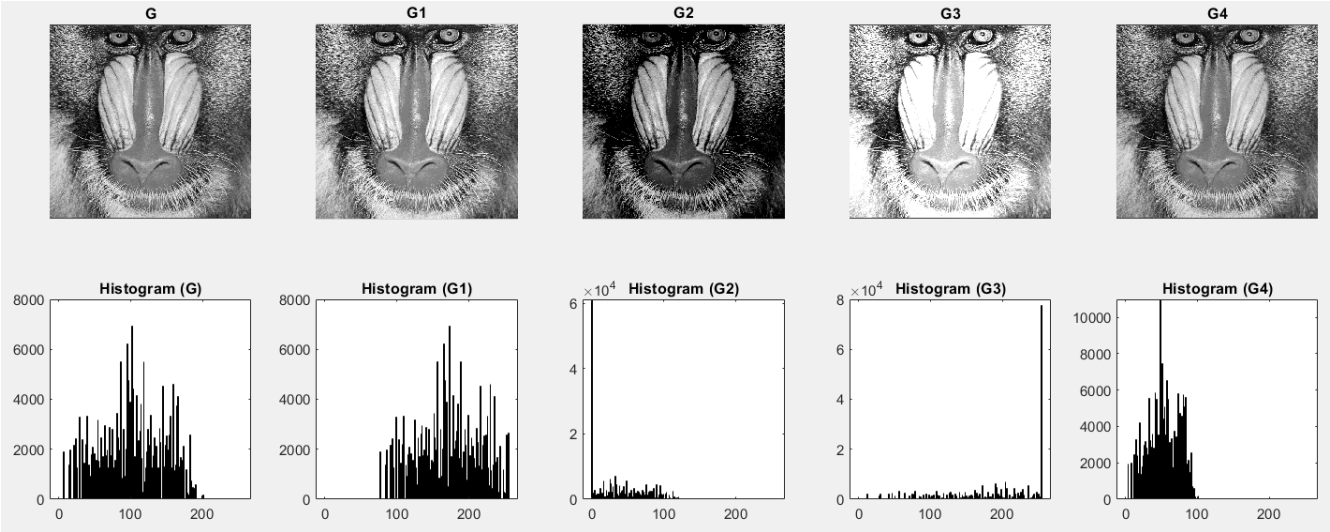


H-2. Histograms of greyscale images with different brightness and contrast

```
clearvars;
close all;
clc;

% Open baboon.bmp and convert to grayscale.

image = imread('baboon.bmp');
grayscale_image = rgb2gray(image);

% Create versions of the original image with different grayscale and contrast:

% G – original image in grayscale
g = grayscale_image;

% G1 – values of all pixels increased by 70
g1 = g +70;

% G2 – values of all pixels decreased by 70
g2 = g -70;

% G3 – increased contrast, e.g. G3=2*G
g3 = 2 * g;

% G4 – decreased contrast, G4=0.5*G
g4 = 0.5 * g;

% Display all images and their histograms in subplots:
figure('position', [100, 100, 1400, 500]);

subplot(2,5,1);
imshow(g, []);
title('G');

subplot(2,5,2);
imshow(g1, []);
title('G1');

subplot(2,5,3);
imshow(g2, []);
title('G2');

subplot(2,5,4);
imshow(g3, []);
title('G3');

subplot(2,5,5);
imshow(g4, []);
title('G4');

% ----------------------------------------

subplot(2,5,6);
histogram(g(:), 0:255);
title('Histogram (G)');
```

```
subplot(2,5,7);
histogram(g1(:), 0:255);
title('Histogram (G1)');

subplot(2,5,8);
histogram(g2(:), 0:255);
title('Histogram (G2)');

subplot(2,5,9);
histogram(g3(:), 0:255);
title('Histogram (G3)');

subplot(2,5,10);
histogram(g4(:), 0:255);
title('Histogram (G4)');
```

Result of the code:

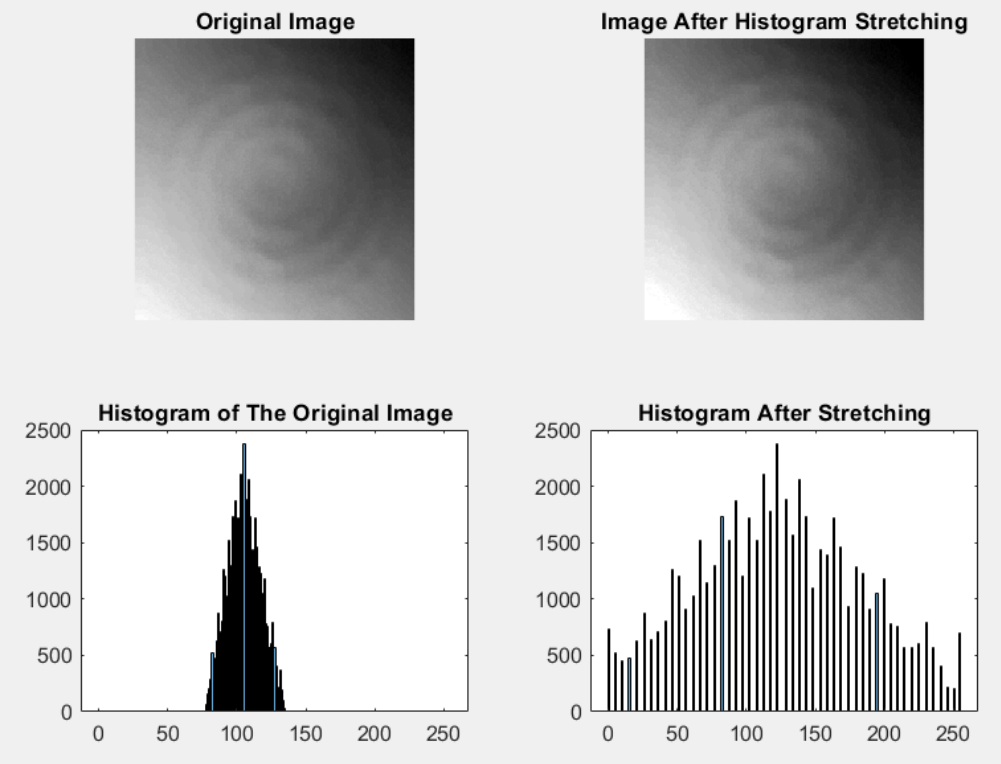


H-3. Histogram stretching

```
clearvars;
close all;
clc;

% Refer to the documentation and implement histogram stretching for the hist1.bmp
image.

image = imread('hist1.bmp');
stretched_image = imadjust(image);

%  Display the result of the operation in subplots:

figure('position', [100, 100, 700, 500]);

subplot(2,2,1);
imshow(image, []);
title('Original Image');
```

```
subplot(2,2,2);
imshow(stretched_image, []);
title('Image After Histogram Stretching');

subplot(2,2,3);
histogram(image(:), 0:255);
title('Histogram of The Original Image');

subplot(2,2,4);
histogram(stretched_image(:), 0:255);
title('Histogram After Stretching');
```

Result of the code:



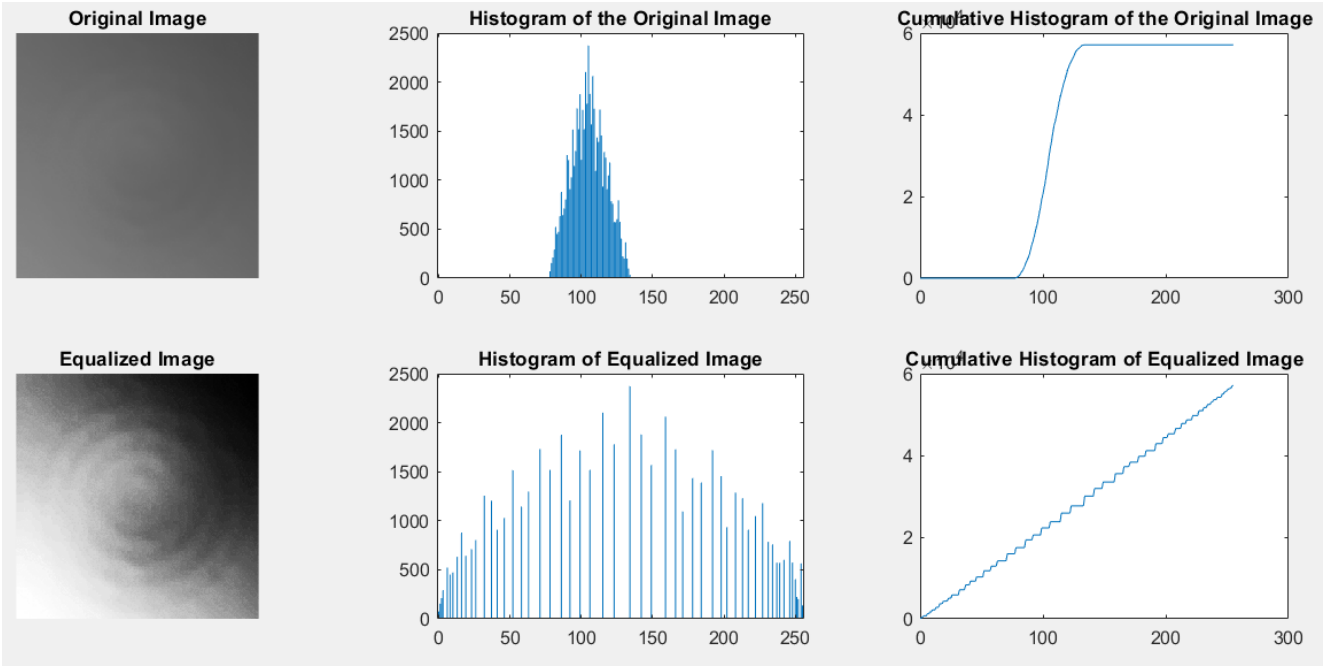
H-4. Histogram equalization

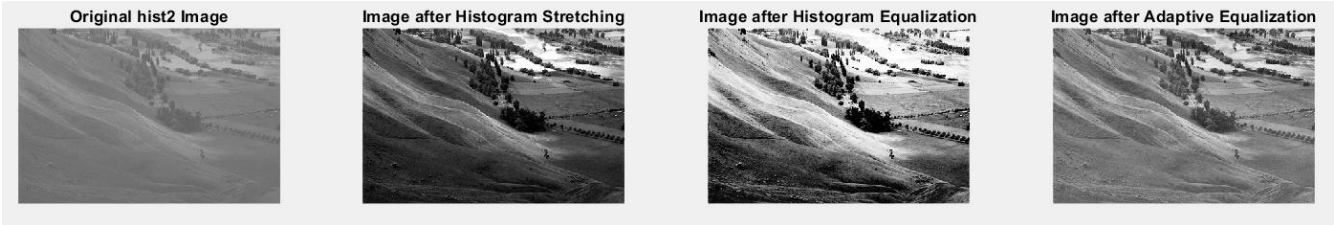
```
clearvars;
close all;
clc;

% Determine the cumulative histogram for the hist1.bmp image. The imhist
% function returns vectors describing the histogram (counts and corresponding
% brightness levels): [H, x] = imhist (I, n).

image = imread('hist1.bmp');
[H, x] = imhist(image, 256);

% For the calculation of the cumulative histogram use the cumsum function.
cumulative_hist = cumsum(H);
```

```
% Implementing the histogram equalization algorithm

I = imread('hist1.bmp');
[H, x] = imhist(I, 256);

% Rescaling
maxCumHist = max(cumulative_hist);
scaledCumHist = (255 * cumulative_hist) / maxCumHist;

% LUT transform
lut = uint8(scaledCumHist);

equalizedImage = intlut(I, lut);

[H_equalized, x_equalized] = imhist(equalizedImage, 256);
cumulative_hist_equalized = cumsum(H_equalized);

% Display the results
figure('position', [100, 100, 1200, 500]);

subplot(2, 3, 1);
imshow(I);
title('Original Image');

subplot(2, 3, 2);
bar(x, H);
title('Histogram of the Original Image');

subplot(2, 3, 3);
plot(x, cumulative_hist);
title('Cumulative Histogram of the Original Image');

subplot(2, 3, 4);
imshow(equalizedImage);
title('Equalized Image');

subplot(2, 3, 5);
bar(x_equalized, H_equalized);
title('Histogram of Equalized Image');

subplot(2, 3, 6);
plot(x_equalized, cumulative_hist_equalized);
title('Cumulative Histogram of Equalized Image');


%-----------------------------------------------------

originalImage = imread('hist2.bmp');

% Histogram stretching
stretchedImage = imadjust(originalImage);

% Histogram equalization
equalizedImageHisteq = histeq(originalImage);

% Adaptive histogram equalization
equalizedImageAdapthisteq = adapthisteq(originalImage);
```

```
% Display

figure('position', [100, 100, 1400, 400]);

subplot(1, 4, 1);
imshow(originalImage);
title('Original hist2 Image');

subplot(1, 4, 2);
imshow(stretchedImage);
title('Image after Histogram Stretching');

subplot(1, 4, 3);
imshow(equalizedImageHisteq);
title('Image after Histogram Equalization');

subplot(1, 4, 4);
imshow(equalizedImageAdapthisteq);
title('Image after Adaptive Equalization');
```

Result of the code:





H-5. Histogram matching

```
clearvars;
close all;
clc;

phobos = imread('phobos.bmp');
figure('position', [50, 100, 1400, 650]);

subplot(2, 5, 1);
imshow(phobos);
title('Original Phobos Image');

phobos_stretched = imadjust(phobos);
subplot(2, 5, 2);
imshow(phobos_stretched);
title('After Histogram Stretching');

phobos_he = histeq(phobos);
subplot(2, 5, 3);
imshow(phobos_he);
title('After Histogram Equalization (HE)');

phobos_clahe = adapthisteq(phobos);
subplot(2, 5, 4);
imshow(phobos_clahe);
title('After Adaptive Equalization');

load desiredHistogram;
phobos_matched = histeq(phobos, desiredHistogram);
subplot(2, 5, 5);
imshow(phobos_matched);
title('After Histogram Matching (HM)');

% Calculate and display histograms for all images
subplot(2, 5, 6);
imhist(phobos);
title('Histogram');

subplot(2, 5, 7);
imhist(phobos_stretched);
title('Histogram');

subplot(2, 5, 8);
imhist(phobos_he);
title('Histogram');

subplot(2, 5, 9);
imhist(phobos_clahe);
title('Histogram');

subplot(2, 5, 10);
imhist(phobos_matched);
title('Histogram');
```

Result of the code: