**Assignment NO #02**

****

Spring 2023

CSE-423 Digital Image Processing

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Doctor. Abeer Irfan**

**June** 07, 2023

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Task01:**

Write MATLAB script to implement Thresholding.

**MATLAB Code:**

% Read the input image

gray\_image = imread('lena.png');

% Define the threshold value

threshold = 50;

% Apply thresholding to the grayscale image

thresholded\_image = gray\_image;

thresholded\_image(gray\_image < threshold) = 0;

thresholded\_image(gray\_image >= threshold) = 255;

% Display the original and thresholded images

subplot(1, 2, 1);

imshow(gray\_image);

title('Original Image');

subplot(1, 2, 2);

imshow(thresholded\_image);

title('Thresholded Image');

Result:



**Task02:**

Write a MATLAB script to implement the negative transformation of an input image.

**MATLAB Code:**

% Read the input image

gray\_image = imread('lena.png');

% Check bit depth

info = imfinfo('lena.png');

bit\_depth = info.BitDepth;

disp(['Bit Depth: ' num2str(bit\_depth)]);

% As the imgae is 8-bits Compute the maximum intensity value (L-1)

L = 256;

max\_intensity = L - 1;

% Perform the negative transformation

negative\_image = max\_intensity - gray\_image;

% Display the original and negative transformed images

subplot(1, 2, 1);

imshow(gray\_image);

title('Original Image');

subplot(1, 2, 2);

imshow(negative\_image);

title('Negative Transformed Image');

Result:



**Task03:**

Log transformation can be obtained by the following relation between input, s, and output image, r.

**s=clog(r+1).**

Find log transform and inverse log of provided image.

**MATLAB Code:**

% Read the input image

input\_image = imread('peppers256.png');

% Convert the input image to double precision for accurate calculations

input\_image = im2double(input\_image);

% Log transformation

c = 0.5; % Scaling constant

log\_transformed\_image = c \* log(input\_image + 1);

% Inverse log transformation

inverse\_log\_transformed\_image = exp(log\_transformed\_image) - 1;

% Display the images

subplot(1, 3, 1);

imshow(input\_image);

title('Input Image');

subplot(1, 3, 2);

imshow(log\_transformed\_image);

title('Log Transformed Image');

subplot(1, 3, 3);

imshow(inverse\_log\_transformed\_image);

title('Inverse Log Transformed Image');

**Result:**



**Task04:**

Write MATLAB code to implement power law transformation of an image.

**MATLAB Code:**

% Read the input image

input\_image = imread('barbara.png');

% Convert the input image to double precision for accurate calculations

input\_image = im2double(input\_image);

% Define the gamma values

gamma\_values = [0.5, 1, 2];

% Apply the gamma transformation for each gamma value

for i = 1:numel(gamma\_values)

gamma = gamma\_values(i);

c = 1;

% Apply the gamma transformation

gamma\_transformed\_image = c \* (input\_image.^gamma);

% Display the transformed image

figure;

subplot(1, 2, 1);

imshow(input\_image);

title('Original Image');

subplot(1, 2, 2);

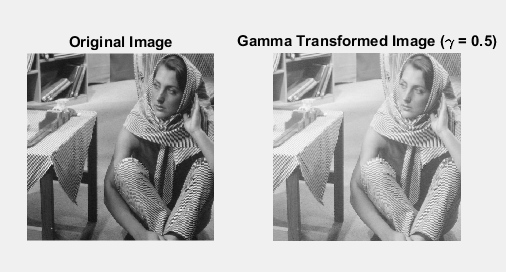
imshow(gamma\_transformed\_image);

title(['Gamma Transformed Image (\gamma = ' num2str(gamma) ')']);

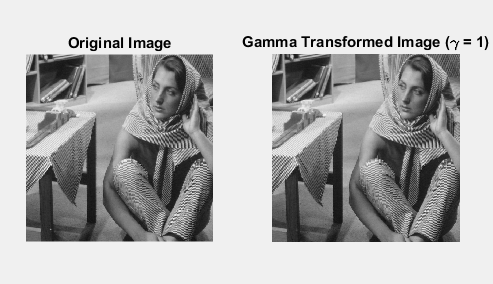
end

**Result:**

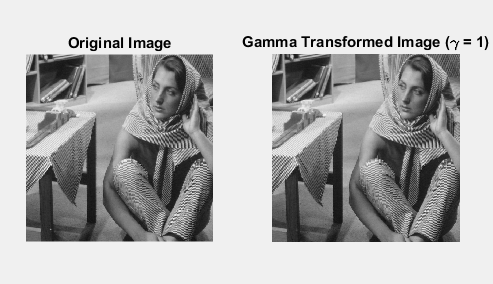
**When gamma < 1:**



**When gamma = 1:**



**When gamma > 1:**



**Task05:**

Write MATLAB script to obtain g (l), piecewise transformation function.

**MATLAB Code:**

% Read the input image

input\_image = imread('boat.png');

% Convert the input image to double precision for calculations

input\_image = im2double(input\_image);

% Define the parameters for contrast stretching

r\_min = min(input\_image(:));

r\_max = max(input\_image(:));

s\_min = 0;

s\_max = 1;

% Compute the slope and intercept for the piecewise transformation function

slope = (s\_max - s\_min) / (r\_max - r\_min);

intercept = s\_min - slope \* r\_min;

% Apply the piecewise transformation function

output\_image = slope \* input\_image + intercept;

% Display the original and contrast-stretched images

figure;

subplot(1, 2, 1);

imshow(input\_image);

title('Original Image');

subplot(1, 2, 2);

imshow(output\_image);

title('Contrast-Stretched Image');

**Result:**



**Task06:**

Write MATLAB code to take an input image and implement gray-level slicing.

**MATLAB Code:**

% Read the input image

input\_image = imread('house.png');

% Define the range of gray levels to be highlighted

lower\_threshold = 50;

upper\_threshold = 150;

% Create a binary mask based on the gray level range

binary\_mask = (input\_image >= lower\_threshold) & (input\_image <= upper\_threshold);

% Apply the binary mask to the input image to obtain the gray-level sliced image

output\_image = input\_image;

output\_image(~binary\_mask) = 0;

% Display the original and gray-level sliced images

figure;

subplot(1, 2, 1);

imshow(input\_image);

title('Original Image');

subplot(1, 2, 2);

imshow(output\_image);

title('Gray-Level Sliced Image');

**Result:**



**Task07:**

Write MATLAB code to implement bit-plane slicing.

**MATLAB Code:**

% Read the grayscale input image

input\_image = imread('cameraman.jpg');

% Convert the grayscale image to binary planes

bit\_planes = zeros(size(input\_image, 1), size(input\_image, 2), 8, 'uint8');

for bit = 0:7

bit\_planes(:, :, bit+1) = bitget(input\_image, bit+1);

end

% Display the bit-plane images

figure;

for bit = 0:7

subplot(2, 4, bit+1);

imshow(bit\_planes(:, :, bit+1), []);

title(['Bit-Plane ', num2str(bit)]);

end

**Result:**

