Table of Contents

IVP Assignment 1

```
% Name: Chanakya Ajit Ekbote
% Institute: IIT, Bhubaneswar%DATE: 19.09.2020
% Degree: Btech
% Branch: Electronics and Communication
% Roll Number: 17EC01041
```

Creating a new environment.

```
clc;
clear all;
close all;
```

Functions Created for the Assignment: rgb_to_hsi

```
% Function that converts an rgb image to hsi.

function [hue, sat, inten] = rgb_to_hsi(red, green, blue, img)
    hue = acos((1/2 * ((red - green)+(red - blue)))./((red - green).^2
 + sqrt((red-blue).*(green - blue))+0.000001));
    hue(blue>green)= 360 - hue(blue>green);
    sat = 1 - 3./(sum(img, 3)) .* min(img,[],3);
    inten = sum(img, 3)./3;
```

Functions Created for the Assignment: image_negative

```
% Function that computes the negative of an image.

function [negative] = image_negative(image)
    negative = image;
    [~, dim] = size(size(image));

    % If conditional is used to check whether the image is 2D or 3D.
    if dim == 2
        [row, col] = size(image);
        for i = 1:row
            for j = 1:col
                negative(i, j) = 255 - negative(i, j);
            end
        end
    else
        [row, col, channels] = size(image);
        for i = 1:row
            for j = 1:col
                for k = 1: channels
                    negative(i, j, k) = 255 - negative(i, j, k);
                end
            end
        end
    end
```

Functions Created for the Assignment: dft_2d

```
% Function that computes the 2D-DFT for an image.

function [dft2d] = dft_2d(image)
    image = double(image);
    [M, N] = size(image);

    % m, n should go from -pi to pi for better interpretation.
    m = -(M-1)/2:1:(M-1)/2;
    n = -(N-1)/2:1:(N-1)/2;

    % Creates the X exponentials required to compute the DFT.
    exponential_x = m' * m;
    exponential_x = exp(-2 * pi * 1i / M .* exponential_x);

    % Creates the Y exponentials required to compute the DFT.
    exponential_y = n' * n;
    exponential_y = exp(-2 * pi * 1i / N .* exponential_y);
```

```
    % Final FFT Computation.
    dft2d = exponential_x * image * exponential_y;
```

Functions Created for the Assignment: log_transform

```
% Function that computes the log transform for an image.

function [log_trans] = log_transform(image, c)
    log_trans = double(image);
    [~, dim] = size(size(image));

    % If conditional is used to check whether the image is 2D or 3D.
    if dim == 2
        [row, col] = size(image);
        for i = 1:row
            for j = 1:col
                log_trans(i, j) = c * log( 1+ (log_trans(i, j)));
            end
        end
    else
        [row, col, channels] = size(image);
        for i = 1:row
            for j = 1:col
                for k = 1: channels
                    log_trans(i, j, k) = c * log(1 + (log_trans(i, j,
 k)));
                end
            end
        end
    end
```

Functions Created for the Assignment: gamma_transform

```
% Function that computes the gamma transform for an image.

function [gamma_trans] = gamma_transform(image, c, gamma)
    gamma_trans = double(image);
    [~, dim] = size(size(image));

    % If conditional is used to check whether the image is 2D or 3D.
    if dim == 2
        [row, col] = size(image);
        for i = 1:row
            for j = 1:col
```

```
                gamma_trans(i, j) = c * (gamma_trans(i, j))^(gamma);
            end
        end
    else
        [row, col, channels] = size(image);
        for i = 1:row
            for j = 1:col
                for k = 1: channels
                    gamma_trans(i, j, k) = c * (gamma_trans(i, j,
k))^(gamma);
                end
            end
        end
    end
```

Functions Created for the Assignment: pixel_hist_2d

```
% Function that computes the frequency of pixels of a particular
% intensity in an image.

function [pixel_hist_] = pixel_hist_2d(image)
    pixel_hist_ = zeros(1, 256);
    [row, col] = size(image);
    for i = 1:row
        for j = 1:col
            % Statement that adds one value to each array position
 where
            % the pixel intensities lie.
            pixel_hist_(image(i, j)+1) = pixel_hist_(image(i, j)+1) +
1;
        end
    end
```

Functions Created for the Assignment: histogram_equalization

```
% Function that performs histogram equalization.

function [histeqimage] = histogram_equalization(image)
    [row, col] = size(image);
    keys = [];
    histeqimage = image;

    % hist_map contains is a hash map that contains the freq
 histogram.
```

```
    hist_map = containers.Map();
    % hist_map contains is a hash map that contains the cdf.
    cdf_map = containers.Map();
    % hist_map contains is a hash map that contains the transformed
 results.
    hist_eq_map = containers.Map();

    % Computing the frequency.
    for i=1:row
        for j=1:col
            key = char(image(i, j));
            if isKey(hist_map, key)
                hist_map(key) = hist_map(key) + 1;
            else
                hist_map(key) = 1;
                keys = [keys; key];
            end
        end
    end

    keys = sort(keys);
    sum = 0;
    cdf_min = hist_map(keys(char(1)));

    [key_length, ~] = size(keys);

    % Computing the CDF.
    for i=1:key_length
        sum = sum + hist_map(keys(i));
        cdf_map(keys(i)) = sum;
    end

    % Computing the transformation function.
    for i=1:key_length
        hist_eq_map(keys(i)) = round((cdf_map(keys(i))-cdf_min)*255/
(row*col-cdf_min));
    end

    % Transforming the Image.
    for i=1:row
        for j=1:col
            key = char(image(i, j));
            histeqimage(i,j) = hist_eq_map(key);
        end
    end
```

Image Imports

```
cameraman = imread('C:\Chanakya\Projects\ivp-assignments
\Assignment-1\images\cameraman.tif');
lena_color = imread('C:\Chanakya\Projects\ivp-assignments
\Assignment-1\images\lena_color_256.tif');
```

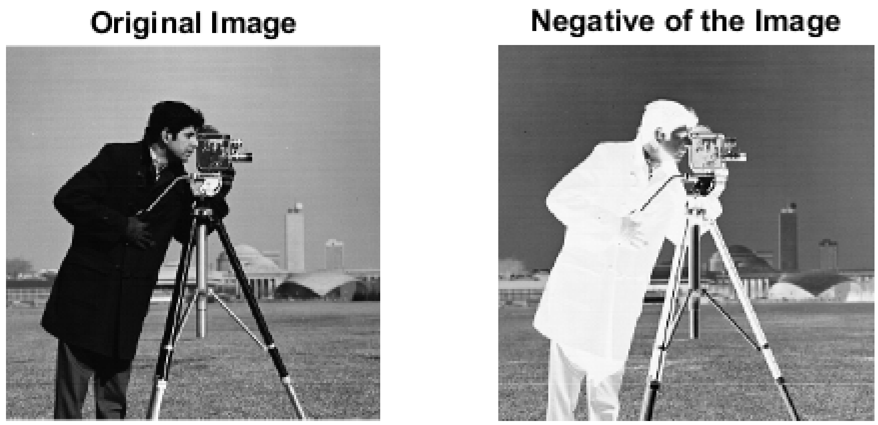
Question 1: Seperate an RGB image into its constituent colours and then convert the RGB to HSI format.

```
% Decomposing the image to its constituent colors.
red = lena_color(:,:,1);
green = lena_color(:,:,2);
blue = lena_color(:,:,3);

% Plotting the image and its constituent RGB Colors.
figure('Name', 'Decomposing an RGB Image to its Constituent Colours');
subplot(2,2,1);
imshow(lena_color);
title('Original Image');

subplot(2,2,2);
imshow(red);
title('Red Channel');

subplot(2,2,3);
imshow(blue);
title('Blue Channel');

subplot(2,2,4);
imshow(green);
title('Green Channel');

% Calling the rgb_to_hsi function.
[hue, sat, int] = rgb_to_hsi(double(red), double(green), double(blue),
 double(lena_color));

% Comparing the hue, saturation and intensity to the original image.
figure('Name', 'Decomposing an RGB Image to HSI');
subplot(2,3,1);
imshow(lena_color);
title('Original Image');

subplot(2,3,2);
imshow(uint8(hue));
title('Hue Channel');

subplot(2,3,3);
imshow(uint8(100 * sat));
title('Saturation Channel');

subplot(2,3,4);
imshow(uint8(int));
title('Intensity Channel');

subplot(2,3,4);
imshow(uint8(int));
```

```
title('Intensity Channel');

% Computing the RBG image assuming HSI channels.
his_image(:,:,1) = hue; his_image(:,:,2) = sat; his_image(:,:,3) =
 int;
subplot(2,3,5);
imshow(uint8(his_image));
title('Image using HSI as RGB');
```

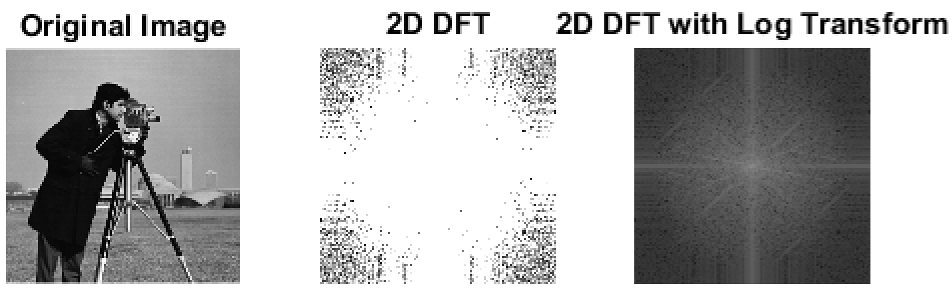
Question 2: Obtaining the negative of the image.

```
image = cameraman;
% Calling the image_negative function.
negative = image_negative(image);

% Comparing the image and the image negative.
figure('Name', 'Transforming an image to its negative.');
subplot(1,2,1);
imshow(image);
title('Original Image');

subplot(1,2,2);
imshow(negative);
title('Negative of the Image');
```



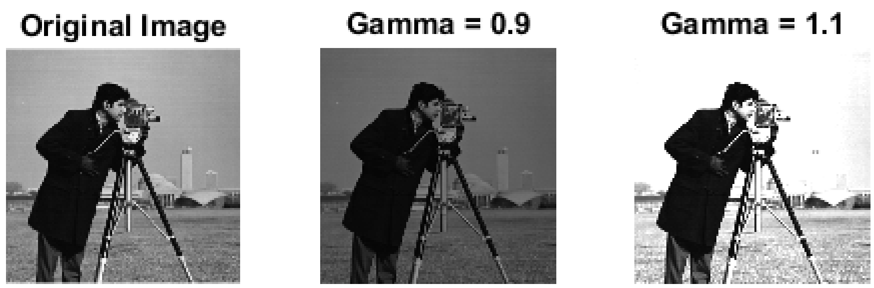
Question 3: Computing the 2D-DFT of the image and then its log transform

```
% Calling the dft_2d function.
dft2d = dft_2d(cameraman);

% Comparing the image, the 2D-DFT and the log transform of the 2D DFT.
figure('Name', 'Computing the 2D-DFT of the image.');
subplot(1,3,1);
imshow(cameraman);
title('Original Image');

subplot(1,3,2);
imshow(uint8(abs(dft2d)));
title('2D DFT');

subplot(1,3,3)
imshow(uint8(log_transform(abs(dft2d), 10)));
title('2D DFT with Log Transform');

% The DFT can be easily visualized after the log transform.
```



Question 4: Computing the gamma transform of images.

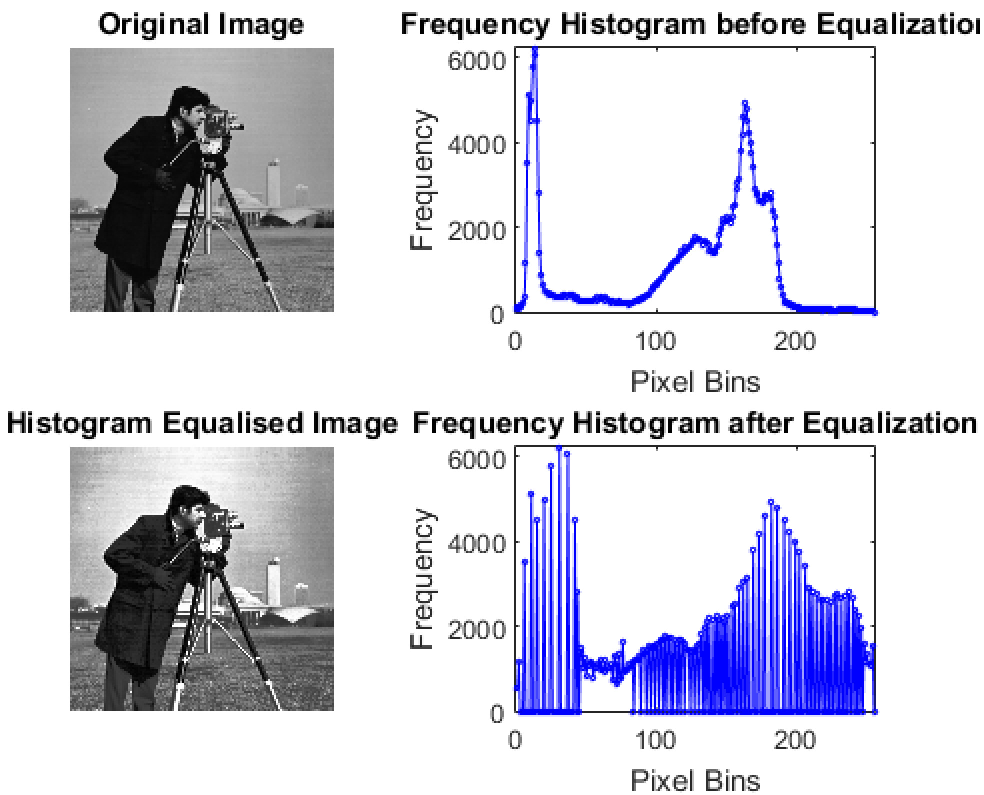
```
% Calling the gamma_transform function.
image_1 = gamma_transform(cameraman, 1, 0.9);
image_2 = gamma_transform(cameraman, 1, 1.1);

% Comparing the iamge, the 2D-DFT and the log transform of the 2D DFT.
figure('Name', 'Computing the 2D-DFT of the image.');
subplot(1,3,1);
imshow(cameraman);
title('Original Image');

subplot(1,3,2);
imshow(uint8(image_1));
title('Gamma = 0.9');

subplot(1,3,3)
imshow(uint8(image_2));
title('Gamma = 1.1');
```



Question 5: Using Histogram Equalization on the image.

```
% Calling the hist_2d function to get the histogram before
 equalization.
hist_before = pixel_hist_2d(cameraman);

% Calling the histogram_equalization function to get the histogram
% equalised image.
histeq_image = histogram_equalization(cameraman);

% Calling the hist_2d function to get the histogram after
 equalization.
hist_after = pixel_hist_2d(histeq_image);

% Comparing the iamge, the 2D-DFT and the log transform of the 2D DFT.
figure('Name', 'Computing the 2D-DFT of the image.');
subplot(2,2,1);
imshow(cameraman);
title('Original Image');

subplot(2,2,2);
plot(0:1:255, hist_before, '-bo', 'MarkerSize', 2);
title('Frequency Histogram before Equalization');
```

```
xlabel('Pixel Bins');
ylabel('Frequency');
axis tight;

subplot(2,2,3)
imshow(histeq_image);
title('Histogram Equalised Image');

subplot(2,2,4);
plot(0:1:255, hist_after, '-bo', 'MarkerSize', 2);
title('Frequency Histogram after Equalization');
xlabel('Pixel Bins');
ylabel('Frequency');
axis tight;
```



References

```
% 1. https://en.wikipedia.org/wiki/Histogram_equalization
% 2. https://www.imageeprocessing.com/2013/05/converting-rgb-image-to-
hsi.html
```

Published with MATLAB® R2015a