University of Chittagong

Computer Science and Engineering

Lab report on Digital Image Processing lab

Course: Digital Image Processing Lab (CSE 812)

Name: Lab report

Submitted to:

Dr. Rashed Mustafa

Professor, Dept. of Computer Science & Engineering

University of Chittagong, Bangladesh.

Submitted by:

Ishra Naznin

ID: 18701069

Session: 2017-18

Dept. of Computer Science & Engineering

University of Chittagong, Bangladesh

1. Read a color image, covert it to gray image and adjust intensity level.

Ans: The pseudo code for a MATLAB program to read a color image, convert it to a grayscale image, and adjust its intensity level is given below:

1. Read the color image using imread() function and store it in a variable 'image'.
2. Convert the color image to grayscale using rgb2gray() function and store it in a variable 'gray_image'.
3. Adjust the intensity level of the grayscale image using imadjust() function and store it in a variable 'adjusted_image'.
4. Display the original color image, grayscale image, and adjusted grayscale image using subplot() and imshow() functions.

Implementation: The matlab program to implement the problem is given below:

```
% Read the color image
image = imread('1.jpg');
% Convert the color image to grayscale
gray_image = rgb2gray(image);
% Adjust the intensity levels using imadjust function
adjusted_image = imadjust(gray_image, [0 1], [0.2 0.8]);

% Display the original and adjusted images
subplot(1, 2, 1); imshow(image); title('Original Image');
subplot(1, 2, 2); imshow(adjusted_image); title('Adjusted Image');
```

Output:

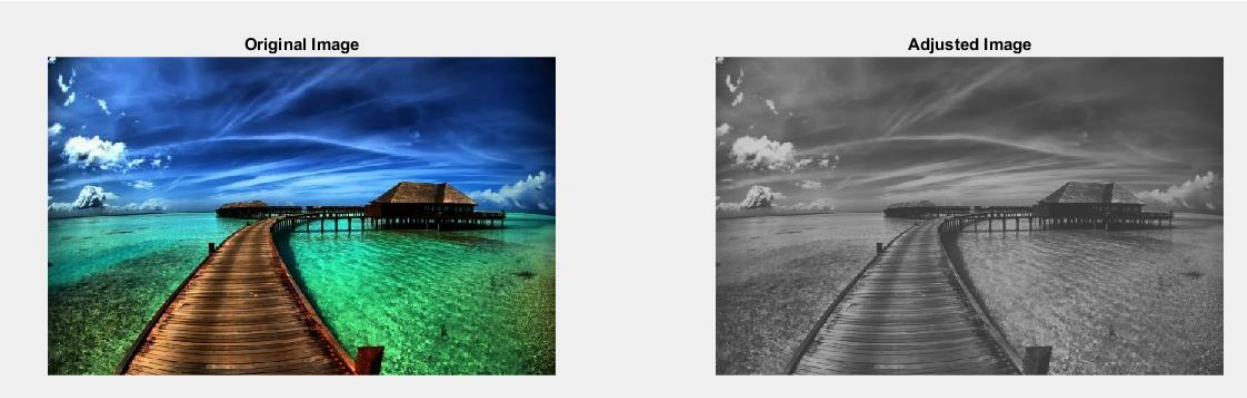

Fig 1: Output of the problem solution

2. Convert a RGB image into binary image. Add 10% one.

Ans: The pseudo code for a MATLAB program that reads a color image, converts it to grayscale, and randomly sets 10% of its pixels to 1 is given below:

1. Read the RGB image using imread() function and store it in a variable rgbImage.
2. Convert the RGB image to grayscale using rgb2gray() function and store it in a variable grayImage.
3. Convert the grayscale image to binary using imbinarize() function and store it in a variable binaryImageBefore.
4. Convert the grayscale image to binary using imbinarize() function and store it in a variable binaryImage.
5. Calculate the total number of pixels in the binary image using numel() function and store it in a variable numPixels.
6. Calculate the number of pixels to be set as ones (10% of the total pixels) using round() function and store it in a variable numOnes.
7. Randomly select pixels to set as ones using randperm() function and store the indexes in a variable randomIndexes.
8. Set the selected pixels to ones in the binary image using binaryImage(randomIndexes) = 1.
9. Display the original RGB image and the resulting binary image using subplot() and imshow() functions.

Implementation: The matlab program to implement the problem is given below:

```
% Read the RGB image
rgbImage = imread('2.jpg');

% Convert the RGB image to grayscale
grayImage = rgb2gray(rgbImage);

% Convert the grayscale image to binary
binaryImageBefore = imbinarize(grayImage);
binaryImage = imbinarize(grayImage);

% Calculate the total number of pixels in the binary image
numPixels = numel(binaryImage);

% Calculate the number of pixels to be set as ones (10% of the total pixels)
```

```
numOnes = round(0.1 * numPixels);

% Randomly select pixels to set as ones
randomIndexes = randperm(numPixels, numOnes);
binaryImage(randomIndexes) = 1;

% Display the original RGB image and the resulting binary image
subplot(1, 3, 1);imshow(rgbImage);title('Original RGB Image');
subplot(1, 3, 2);imshow(binaryImageBefore);title('Binary Image ');
subplot(1, 3, 3);imshow(binaryImage);title('Binary Image with 10% Ones');
```

Output:

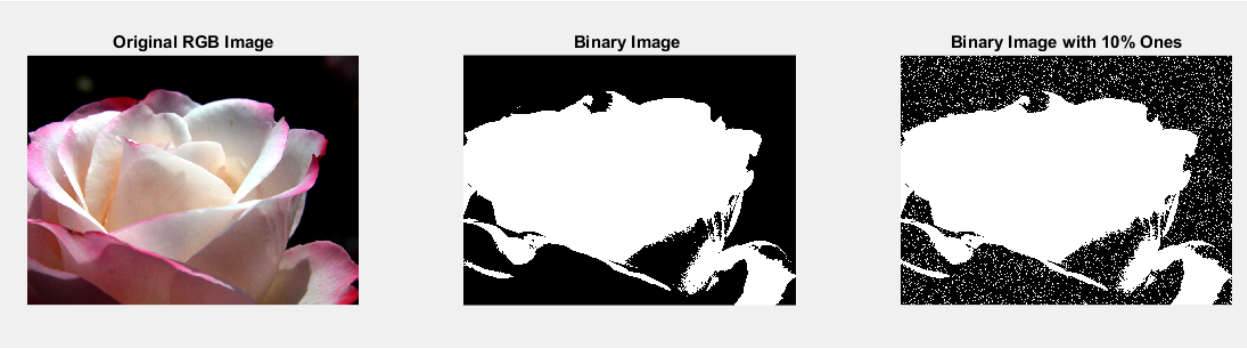

Fig 2: Output of the problem solution

3. Find compression ratio of a digital image.

Ans: Here is the pseudo code for a MATLAB program that calculates the compression ratio of a JPEG image:

1. Use the imfinfo() function to get information about the JPEG image and store it in a variable K.
2. Calculate the total number of bytes in the uncompressed image using the formula image_byte = K.Width * K.Height * K.BitDepth / 8.
3. Get the size of the compressed JPEG file in bytes using the FileSize field of the K structure and store it in a variable compressed_byte.
4. Calculate the compression ratio using the formula compression_ratio = image_byte / compressed_byte.
5. Display the compression ratio using the disp() function.

Implementation: The matlab program to implement the problem is given below:

```
K=imfinfo('2.jpg');
image_byte=K.Width*K.Height*K.BitDepth/8;
compressed_byte=K.FileSize;

compression_ratio=image_byte/compressed_byte;

fprintf('Compression ratio is %.2f\n', compression_ratio);
```

Output:

```
>> Number3Asignment
Compression ratio is 13.49
>>
```

Fig 3: Output of the problem solution

4. Resize two digital images and make them equal size.

Ans: The pseudo code for a MATLAB program that reads two color images, resizes them to equal size, and displays them side by side:

1. Read the first color image using imread() function and store it in a variable color_img_1st.
2. Display the original first image using subplot() and imshow() functions.
3. Resize the first color image to 300x300 pixels using imresize() function and store it in a variable img_1st_re.
4. Display the resized first image using subplot() and imshow() functions.
5. Read the second color image using imread() function and store it in a variable color_img_2nd.
6. Display the original second image using subplot() and imshow() functions.
7. Resize the second color image to 300x300 pixels using imresize() function and store it in a variable img_2nd_re.
8. Display the resized second image using subplot() and imshow() functions.

Implementation: The matlab program to implement the problem is given below:

```
clc
color_img_1st=imread('1.jpg');
subplot(2,2,1); imshow(color_img_1st); title('Original 1st Image');
```

```
img_1st_re= imresize(color_img_1st,[300,300])
subplot(2,2,2);imshow(img_1st_re);title('Resize Image');

color_img_2nd=imread('2.jpg');
subplot(2,2,3); imshow(color_img_2nd); title('Original 2nd Image');

img_2nd_re= imresize(color_img_2nd,[300,300])
subplot(2,2,4);imshow(img_2nd_re);title('Resize Image');
```

Output:

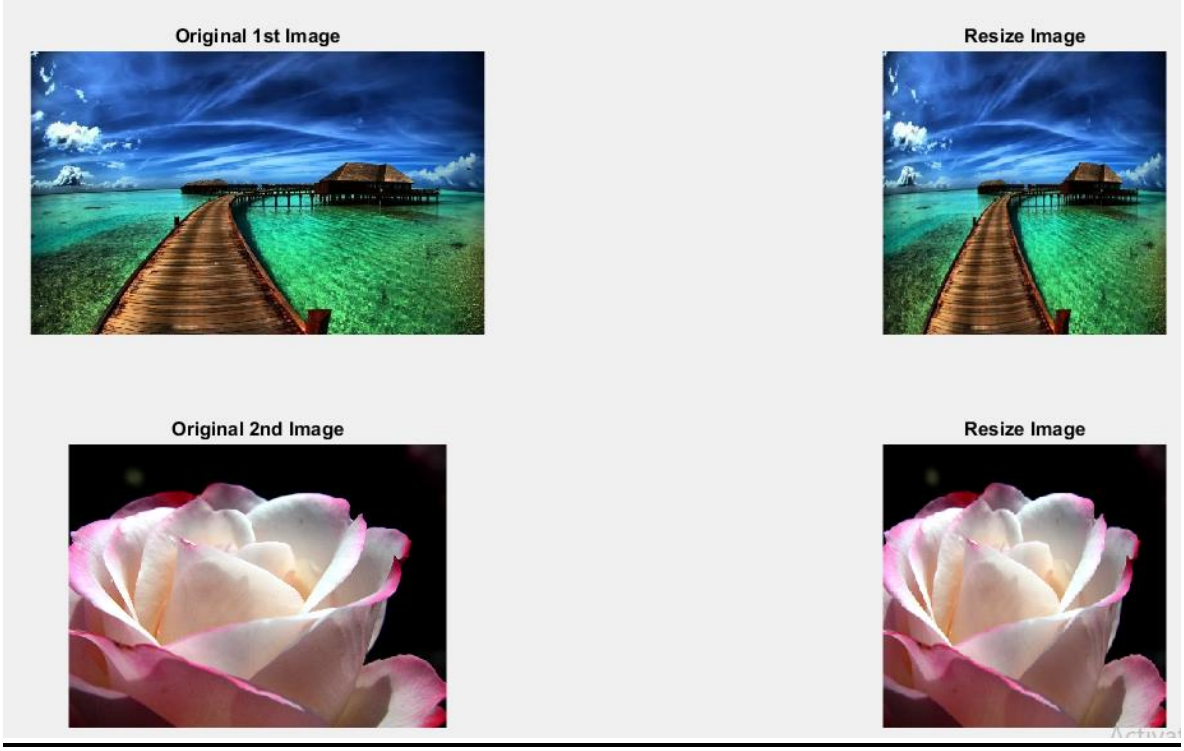

Fig 4: Output of the problem solution

5. Perform OR, AND, XOR, XNOR operation of two digital images.

Ans: The pseudo code for a MATLAB program that reads two color images, resizes them, converts them to binary, performs logical operations on them, and displays the results:

1. Read the first color image using imread() function and store it in a variable color_img_1st.
2. Display the original first image using subplot() and imshow() functions.

3. Resize the first color image to 300x300 pixels using imresize() function and store it in a variable img_1st_re.
4. Convert the resized first image to binary using im2bw() function and store it in a variable binary_img_1st.
5. Display the binary first image using subplot() and imshow() functions.
6. Read the second color image using imread() function and store it in a variable color_img_2nd.
7. Display the original second image using subplot() and imshow() functions.
8. Resize the second color image to 300x300 pixels using imresize() function and store it in a variable img_2nd_re.
9. Convert the resized second image to binary using im2bw() function and store it in a variable binary_img_2nd.
10. Display the binary second image using subplot() and imshow() functions.
11. Perform the logical OR operation on the two binary images using the or() function and store the result in a variable OR_result.
12. Display the OR result using subplot() and imshow() functions.
13. Perform the logical AND operation on the two binary images using the and() function and store the result in a variable AND_result.
14. Display the AND result using subplot() and imshow() functions.
15. Perform the logical XOR operation on the two binary images using the xor() function and store the result in a variable XOR_result.
16. Display the XOR result using subplot() and imshow() functions.
17. Perform the logical XNOR operation on the two binary images using the not() and xor() functions and store the result in a variable XNOR_result.
18. Display the XNOR result using subplot() and imshow() functions.

Implementation: The matlab program to implement the problem is given below:

```
color_img_1st=imread('1.jpg'); color_img_2nd=imread('2.jpg');
subplot(4,2,1);imshow(color_img_1st);title('Original 1st Image');
subplot(4,2,3);imshow(color_img_2nd);title('Original 2nd Image');

img_1st_re= imresize(color_img_1st,[300,300]); binary_img_1st=im2bw(img_1st_re);
img_2nd_re= imresize(color_img_2nd,[300,300]);
binary_img_2nd=im2bw(img_2nd_re);
subplot(4,2,2);imshow(binary_img_1st);title('After binary & resize 1st Image');
subplot(4,2,4);imshow(binary_img_2nd);title('After binary & resize 2nd Image');
```

```
OR_result = or(binary_img_1st , binary_img_2nd);
subplot(4,2,5);imshow(OR_result);title('Output of Or Operation');

AND_result = and(binary_img_1st, binary_img_2nd);
subplot(4,2,6);imshow(AND_result);title('Output of And Operation');

XOR_result = xor(binary_img_1st , binary_img_2nd);
subplot(4,2,7);imshow(XOR_result);title('Output of XOR Operation');

XNOR_result = ~xor(binary_img_1st , binary_img_2nd);
subplot(4,2,8);imshow(XOR_result);title('Output of X-NOR Operation');
```

Output:

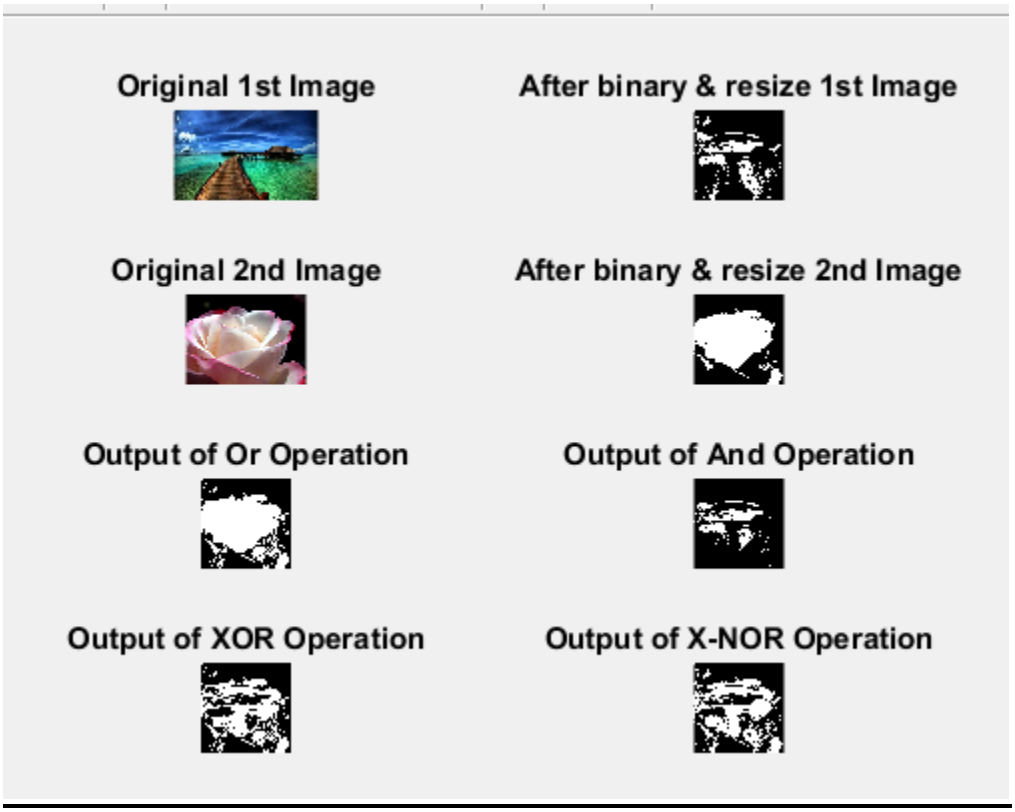

Fig 5: Output of the problem solution

6. Implement image interpolation (NN, BL, BC)

Ans: The pseudo code for a MATLAB program that reads a color image, resizes it, and displays it using different interpolation methods is given below:

1. Read the color image using imread() function and store it in a variable color_img_1st.

2. Display the original image using subplot() and imshow() functions.
3. Resize the color image to 90x50 pixels using imresize() function and store it in a variable img_re.
4. Convert the resized image to binary using im2bw() function and store it in a variable binary_img.
5. Display the binary image using subplot() and imshow() functions.
6. Resize the binary image to 300x600 pixels using imresize() function with nearest neighbor interpolation and store it in a variable nearest_op.
7. Display the nearest neighbor result using subplot() and imshow() functions.
8. Resize the binary image to 300x600 pixels using imresize() function with bicubic interpolation and store it in a variable bicubic_op.
9. Display the bicubic interpolation result using subplot() and imshow() functions.
10. Resize the binary image to 300x600 pixels using imresize() function with bilinear interpolation and store it in a variable bilinear_op.
11. Display the bilinear interpolation result using subplot() and imshow() functions.

Implementation: The matlab program to implement the problem is given below:

```
color_img_1st=imread('1.jpg');

img_re= imresize(color_img_1st,[90,50]);
subplot(2,2,1);imshow(color_img_1st);title('Original Image');

nearast_op = imresize(img_re,[300,600],'nearest');
subplot(2,2,2);imshow(nearast_op);title('Nearest Neighbour');

bicubic_op= imresize(img_re,[300,600],'bicubic');
subplot(2,2,3); imshow(bicubic_op); title('Bicubic Interpolation');

bilinear_op = imresize(img_re,[300,600],'bilinear');
subplot(2,2,4); imshow(bilinear_op); title('Bilinear Interpolation');
```

Output:

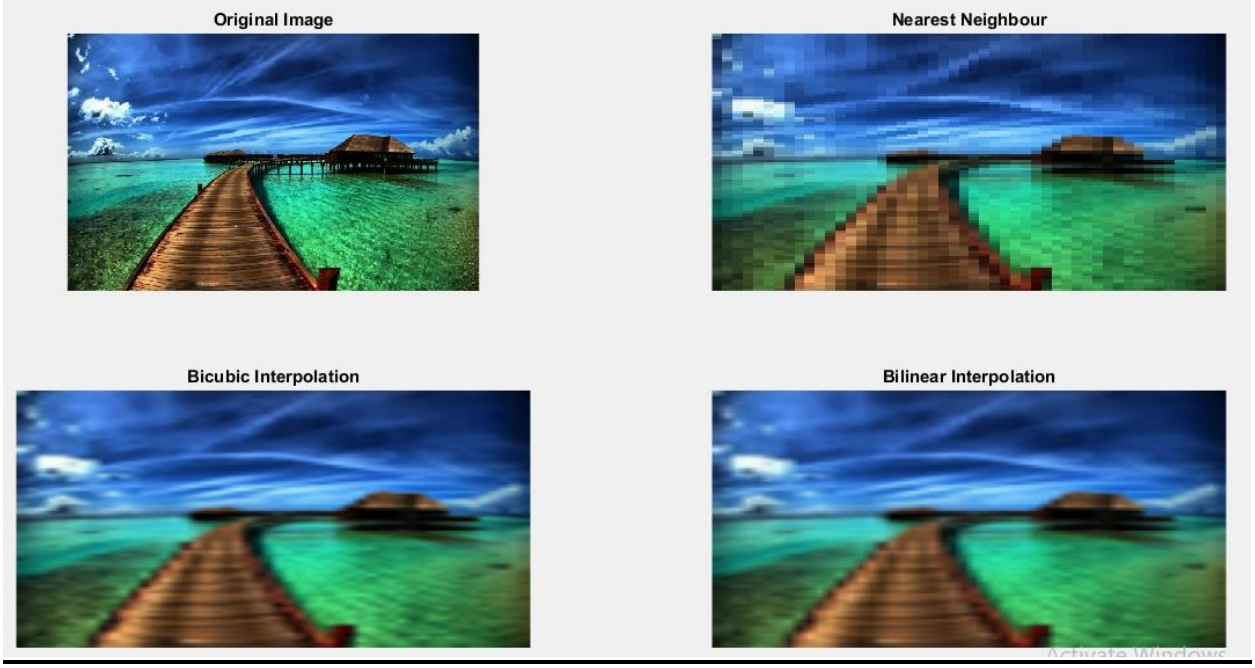

Fig 6: Output of the problem solution

7. Draw sine and cosine wave.

Ans: The pseudo code for a MATLAB program that plots sine and cosine waves in separate subplots is given below:

1. Define the x-axis values using the linspace() function and store them in a variable x.
2. Calculate the y-values for the sine and cosine waves using the sin() and cos() functions and store them in variables y_sin and y_cos, respectively.
3. Create a figure window using the figure() function.
4. Create the first subplot using the subplot() function with arguments 2, 1, 1 to create a 2-row, 1-column grid of subplots and select the first subplot.
5. Plot the sine wave using the plot() function with arguments x and y_sin, and customize the plot using the xlabel(), ylabel(), title(), and grid() functions.
6. Create the second subplot using the subplot() function with arguments 2, 1, 2 to select the second subplot.

7. Plot the cosine wave using the plot() function with arguments x and y_cos, and customize the plot using the xlabel(), ylabel(), title(), and grid() functions.

Implementation: The matlab program to implement the problem is given below:

```
% Define the x-axis values (time or angle)
x = linspace(0, 4*pi, 1000); % Range from 0 to 2*pi with 1000 points
% Calculate the y-values for sine and cosine waves
y_sin = sin(x);      y_cos = cos(x);
% Plot the sine wave
subplot(2, 1, 1);    plot(x, y_sin, 'b', 'LineWidth', 2);
xlabel('x');ylabel('sin(x)');    title('Sine Wave');    grid on;
% Plot the cosine wave
subplot(2, 1, 2);    plot(x, y_cos, 'r', 'LineWidth', 2);
xlabel('x');    ylabel('cos(x)');    title('Cosine Wave');    grid on;
```

Output:

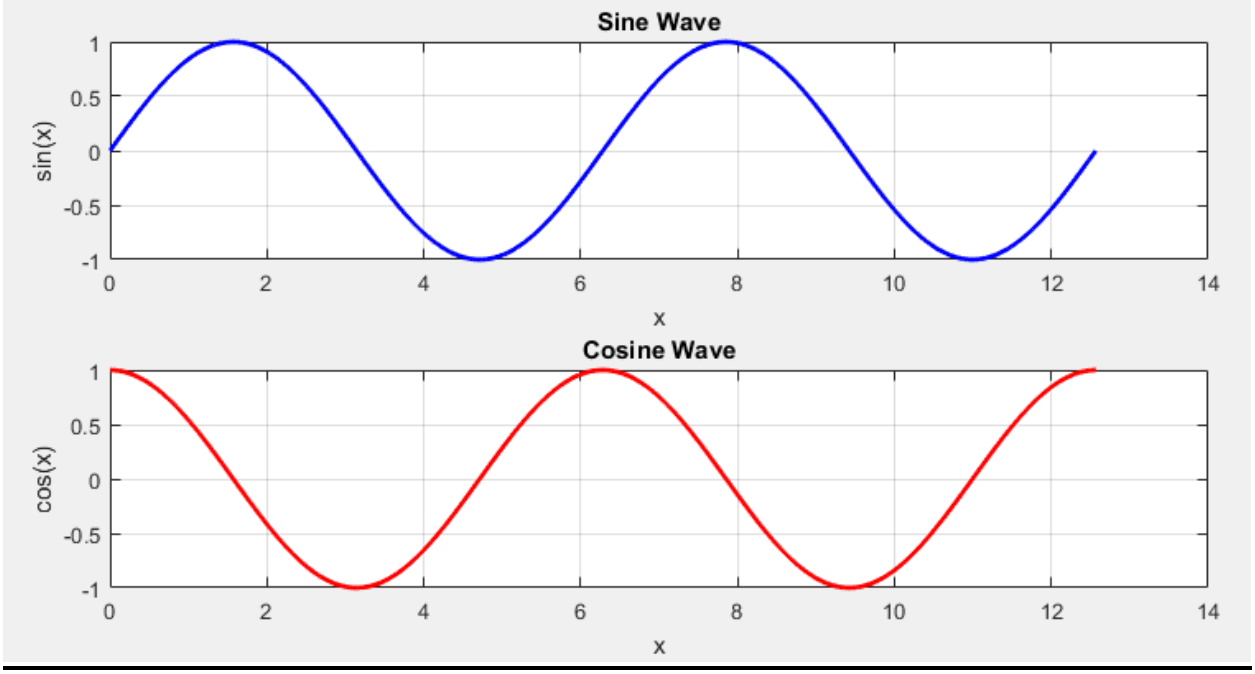

Fig 7: Output of the problem solution

8. Find histogram of images.

Ans: The following is the pseudo code for a MATLAB program that reads an image, converts it to grayscale, calculates its histogram, and displays the histogram:

1. Clear the command window using clc.
2. Clear all variables using clear all.
3. Close all figure windows using close all.
4. Read the image using imread() function and store it in a variable image.
5. Convert the image to grayscale using rgb2gray() function and store it in a variable grayImage.
6. Calculate the histogram of the grayscale image using imhist() function and store it in a variable histogram.
7. Display the histogram using bar() function and customize the plot using xlabel(), ylabel(), title(), and grid() functions.

Implementation: The matlab program to implement the problem is given below:

```
clc
clear all;
close all;
% Read the image
image = imread('1.jpg');
% Convert the image to grayscale if necessary
grayImage = rgb2gray(image);

% Calculate the histogram
histogram = imhist(grayImage);
% Display the histogram
bar(histogram);
title('Histogram of Image');
xlabel('Pixel Intensity');
ylabel('Frequency');
```

Output:

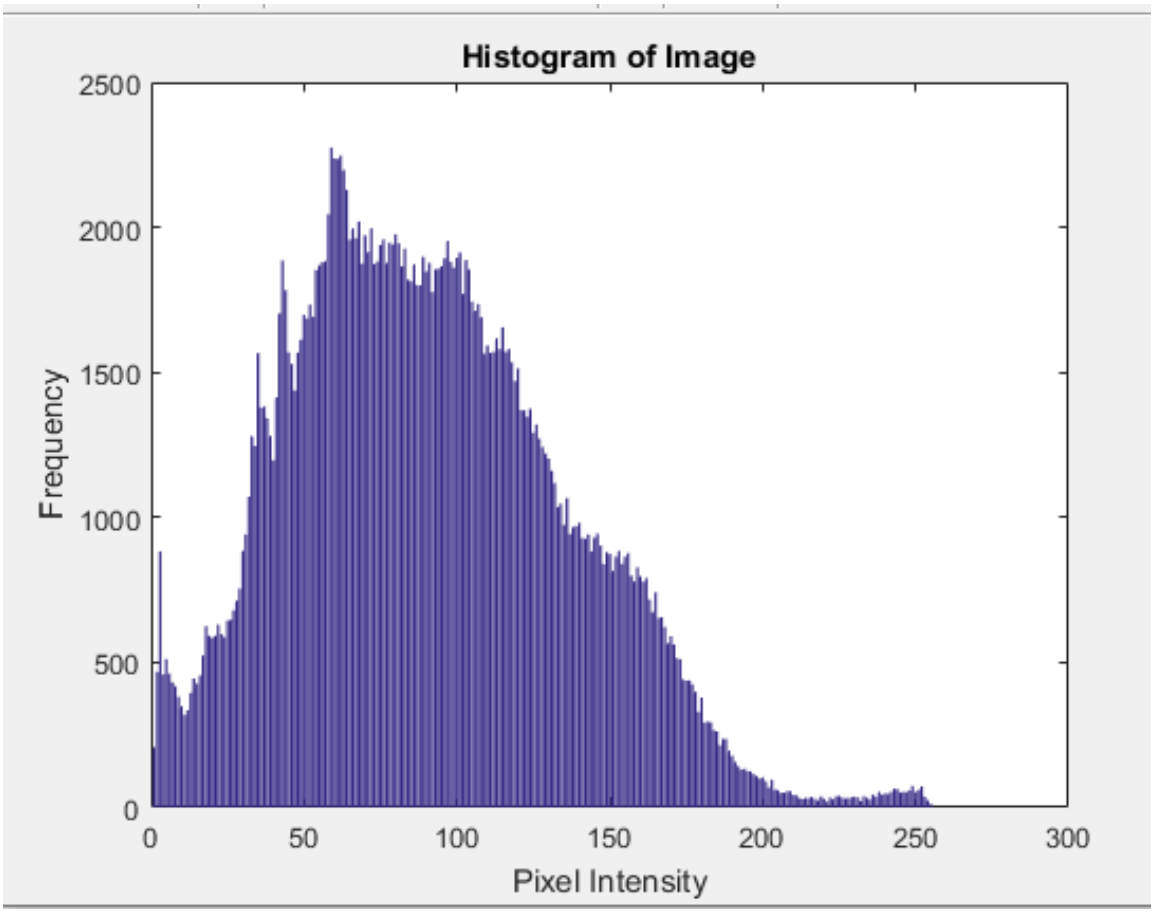

Fig 8: Output of the problem solution

9. Implement histogram equalization.

Ans: Here is the pseudo code for a MATLAB program that reads an image, converts it to grayscale, performs histogram equalization, and displays the original and equalized images and their histograms:

1. Read the image using imread() function and store it in a variable a.
2. Convert the image to grayscale using rgb2gray() function and store it in a variable grayImage.
3. Perform histogram equalization on the grayscale image using histeq() function and store it in a variable equalizedImage.
4. Create a figure window using the figure() function.
5. Create the first subplot using the subplot() function with arguments 2, 2, 1 to create a 2-row, 2-column grid of subplots and select the first subplot.

6. Display the original grayscale image using imshow() function and customize the plot using the title() function.
7. Create the second subplot using the subplot() function with arguments 2, 2, 2 to select the second subplot.
8. Display the histogram of the original grayscale image using imhist() function and customize the plot using the title() function.
9. Create the third subplot using the subplot() function with arguments 2, 2, 3 to select the third subplot.
10. Display the equalized grayscale image using imshow() function and customize the plot using the title() function.
11. Create the fourth subplot using the subplot() function with arguments 2, 2, 4 to select the fourth subplot.
12. Display the histogram of the equalized grayscale image using imhist() function and customize the plot using the title() function.

Implementation: The matlab program to implement the problem is given below:

```
%Histogram equalization is a technique used to enhance the contrast of an image by
redistributing its pixel intensities
a=imread('1.jpg');

grayImage=rgb2gray(a);

equalizedImage=histeq(grayImage);

subplot(2,2,1);imshow(grayImage);title('Original gray scle Image');

subplot(2,2,2);imhist(grayImage);title('Histogram of Original gray scle Image');

subplot(2,2,3);imshow(equalizedImage);title('Image after histogram equalization');

subplot(2,2,4);imhist(equalizedImage);title('histogram of image after histogram equalization');
```

Output:

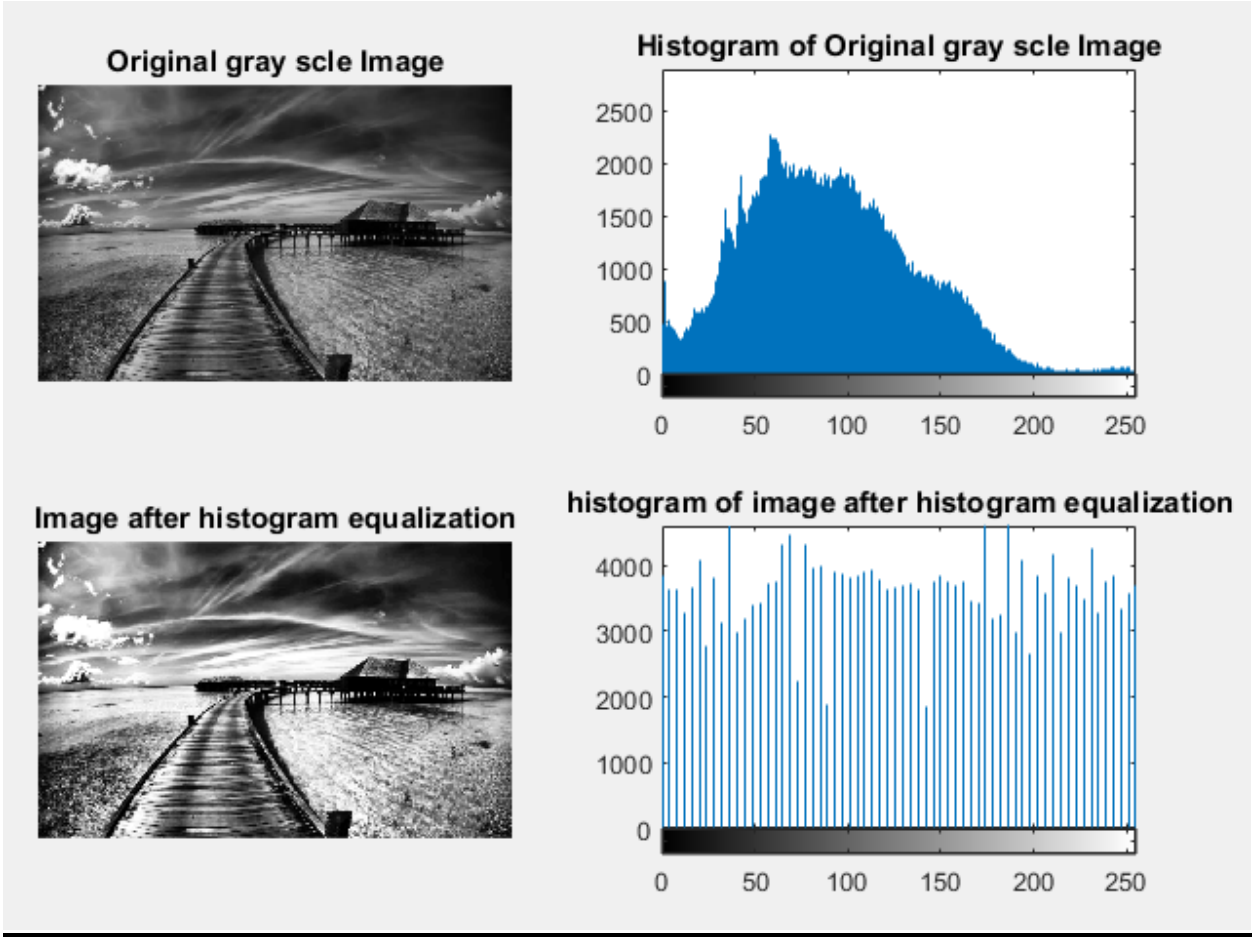

Fig 9: Output of the problem solution

10. Find center point of a digital image.

Ans: Here is the pseudo code for a MATLAB program that reads an image, calculates its center point, and displays the image with the center point:

1. Read the image using imread() function and store it in a variable image.
2. Get the dimensions of the image using size() function and store the height, width, and channels in variables height, width, and channels, respectively.
3. Calculate the center point of the image by dividing the width and height by 2 and rounding down using the floor() function, and store the results in variables centerX and centerY.
4. Display the image using imshow() function and store the handle to the image object in a variable hImage.
5. Hold the current plot using the hold() function.

6. Plot the center point on the image using the plot() function with arguments centerX, centerY, 'r*', and 'MarkerSize', 15.
7. Customize the plot using the title() function.

Implementation: The matlab program to implement the problem is given below:

```
clc
clear all;
close all;
% Read the image
image = imread('1.jpg');

% Get the dimensions of the image
[height, width, channels] = size(image);
% Calculate the center point
centerX = floor(width / 2);
centerY = floor(height / 2);

% Display the center point
imshow(image);
hold on;
plot(centerX, centerY, 'r*', 'MarkerSize', 15);
title('Image with Center Point');
```

Output:

Fig 10: Output of the problem solution

11. Find Euclidean distance between two pixels.

Ans: Here is the pseudo code for a MATLAB program that reads an image, selects two pixels, calculates the Euclidean distance between them, and displays the distance:

1. Read the image using imread() function and store it in a variable image.
2. Define the coordinates of the first pixel using x1 and y1 variables.
3. Define the coordinates of the second pixel using x2 and y2 variables.
4. Get the pixel values at the specified coordinates using image(y1, x1, :) and image(y2, x2, :) and convert the pixel values to a 1-D vector using reshape() function and store them in variables pixel1 and pixel2, respectively.
5. Calculate the Euclidean distance between the pixels using norm() function and store it in a variable distance.
6. Display the Euclidean distance using disp() function.

Implementation: The matlab program to implement the problem is given below:

```
% Read the image
image = imread('1.jpg');

% Define the coordinates of the first pixel
x1 = 100; % x-coordinate of the first pixel
y1 = 200; % y-coordinate of the first pixel

% Define the coordinates of the second pixel
x2 = 150; % x-coordinate of the second pixel
y2 = 300; % y-coordinate of the second pixel

% Get the pixel values at the specified coordinates and convert the pixel values to a 1-D vecto
pixel1 = double(reshape(image(y1, x1, :), [], 1));
pixel2 = double(reshape(image(y2, x2, :), [], 1));

% Calculate the Euclidean distance between the pixels
distance = norm(pixel1 - pixel2);
% Display the Euclidean distance
disp(['Euclidean Distance: ', num2str(distance)]);
```

Output:

```
>> Number11Asignment
Euclidean Distance: 176.2073
```

Fig 11: Output of the problem solution

12. Perform the following operations. (scan line, flip, complement)

Ans: Here is the pseudo code for a MATLAB program that performs scan line, flip, complement image operations:

1. Read the image using imread() function and store it in a variable image.
2. Convert the image to grayscale using rgb2gray() function and store it in a variable grayImg.
3. Select a row index and store it in a variable rowIndex then get the pixel values of the selected row using grayImg(rowIndex,:) , store them in a variable rowValue.
4. Flip the image horizontally using flip() function and store it in a variable flipImage_horizontally.
5. Flip the image vertically using flip() function and store it in a variable flipImage_vertically.
6. Invert the pixel values of the image using imcomplement() function and store it in a variable complementImage.
7. Create a figure window using the figure() function.
8. Create the first subplot using the subplot() function with arguments 3, 2, 1 to create a 3-row, 2-column grid of subplots and select the first subplot.
9. Display the original image using imshow() function and customize the plot using the title() function.
10. Create the second subplot using the subplot() function with arguments 3, 2, 2 to select the second subplot.
11. Plot the selected row using plot() function and customize the plot using the title() function.
12. Create the third subplot using the subplot() function with arguments 3, 2, 3 to select the third subplot.
13. Display the horizontally flipped image using imshow() function and customize the plot using the title() function.
14. Create the fourth subplot using the subplot() function with arguments 3, 2, 4 to select the fourth subplot.
15. Display the vertically flipped image using imshow() function and customize the plot using the title() function.
16. Create the fifth subplot using the subplot() function with arguments 3, 2, 5 to select the fifth subplot.

17. Display the complemented image using imshow() function and customize the plot using the title() function.

Implementation: The matlab program to implement the problem is given below:

```
% Read the image
image = imread('1.jpg');

% Scan line operation vertical
grayImg=rgb2gray(image);    rowIndex=160;    rowValue=grayImg(rowIndex,:);

% Flip operation
flipImage_horizontally = flip(image, 2); % Flip horizontally
flipImage_vertically = flip(image, 1); % Flip vertically

% Complement operation
complementImage = imcomplement(image); % Invert pixel values

subplot(3, 2, 1);imshow(image);title('Original Image');
subplot(3, 2, 2);plot(rowValue);title('Scan Line Image');
subplot(3, 2, 3);imshow(flipImage_horizontally);title('Flipped Horizontally Image');
subplot(3, 2, 4);imshow(flipImage_vertically );title('Flipped Vertically Image');
subplot(3, 2, 5);imshow(complementImage);title('Complement Image');
```

Output:

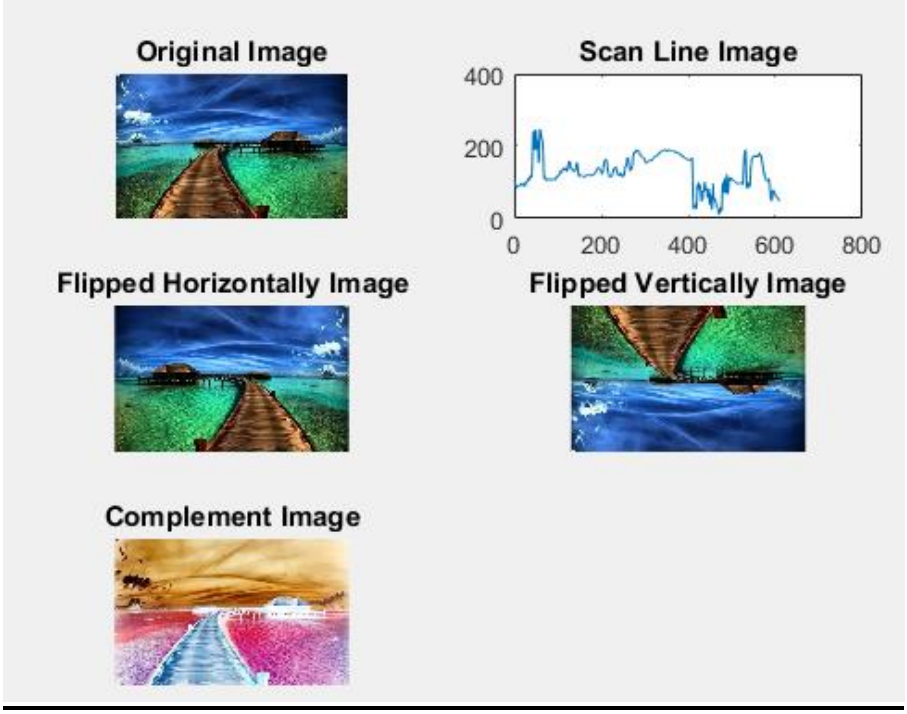

Fig 12: Output of the problem solution

13. Read an RGB image show its three channels separately. Add all channels and compare with original image.

Implementation: The matlab program to implement the problem is given below:

```
% Read the RGB image
image = imread('1.jpg');

% Display the original image
subplot(3, 2, 1);imshow(image);title('Original Image');

% Extract color channels
redChannel = image(:, :, 1);
greenChannel = image(:, :, 2);
blueChannel = image(:, :, 3);

% Display the red channel
subplot(3, 2, 2);imshow(redChannel);title('Red Channel');

% Display the green channel
subplot(3, 2, 3);imshow(greenChannel);title('Green Channel');

% Display the blue channel
subplot(3, 2, 4);imshow(blueChannel);title('Blue Channel');

% Add all channels
combinedImage = redChannel + greenChannel + blueChannel;

% Compare the combined image with the original image
subplot(3, 2, 5);imshow(image);title('Original Image');
subplot(3, 2, 6);imshow(combinedImage);title('Combined Image');
```

Output:

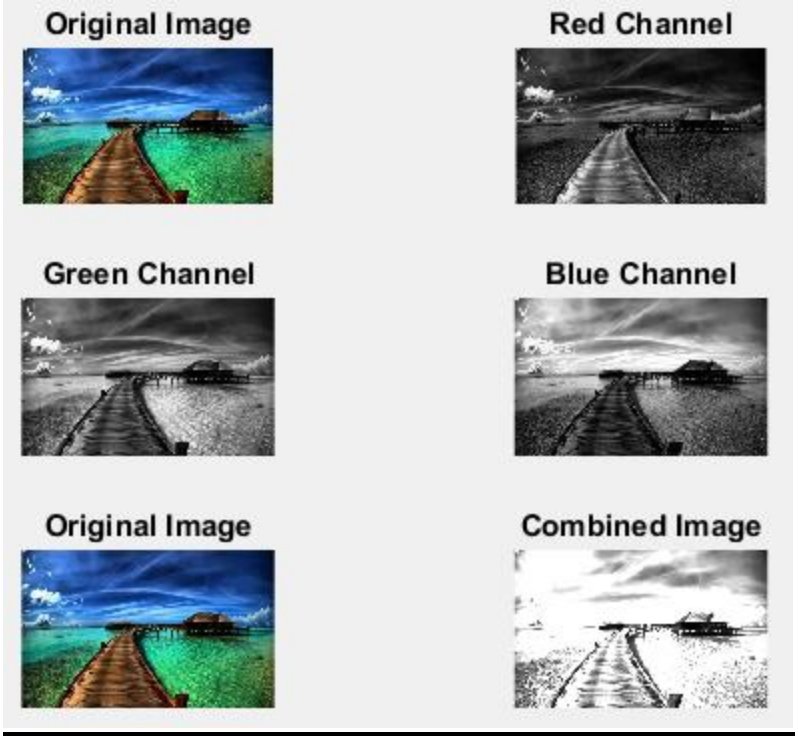

Fig 13: Output of the problem solution

14. Convert RGB to YIQ (Vice versa)

Implementation: The matlab program to implement the problem is given below:

```
% Read the RGB image
rgbImage = imread('1.jpg');

% Convert RGB to YIQ
yiqImage = rgb2ntsc(rgbImage);
% Convert YIQ to RGB
reconstructedImage = ntsc2rgb(yiqImage);

% Display the YIQ image
subplot(1, 3, 1);imshow(rgbImage);title('RGB Image');
subplot(1, 3, 2);imshow(yiqImage);title('YIQ Image');
subplot(1, 3, 3);imshow(reconstructedImage);title('Reconstructed RGB Image');
```

Output:

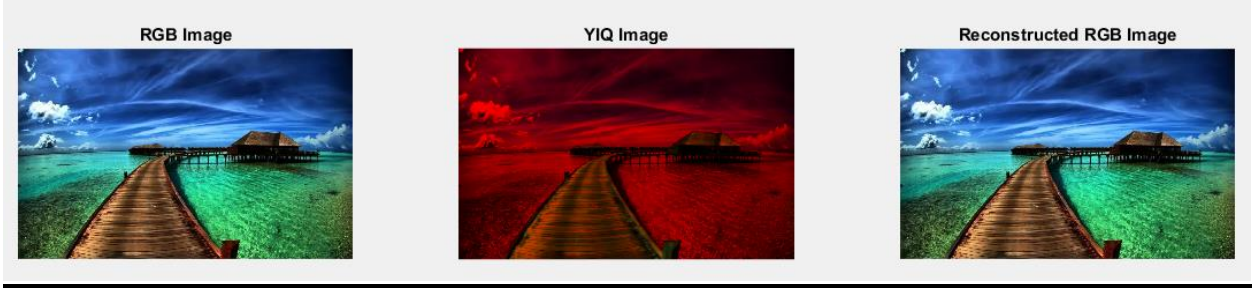

Fig 14: Output of the problem solution

15. Convert RGB to YCbCr (Vice versa)

Implementation: The matlab program to implement the problem is given below:

```
% Read the RGB image
rgbImage = imread('1.jpg');
subplot(1, 3, 1);imshow(rgbImage);title('RGB Image');

% Convert RGB to YCbCr
ycbcrImage = rgb2ycbcr(rgbImage);
subplot(1, 3, 2);imshow(ycbcrImage);title('YCbCr Image');

% Convert YCbCr to RGB
reconstructedImage = ycbcr2rgb(ycbcrImage);
subplot(1, 3, 3);imshow(reconstructedImage);title('Reconstructed RGB Image');
```

Output:

Fig 15: Output of the problem solution

16. Convert RGB to HSI (Vice versa)

Implementation: The matlab program to implement the problem is given below:

```
% MATLAB program for RGB to HSI image conversion.
img = imread('1.jpg');

% Represent the RGB image in [0 1] range
```

```
        I = double(img) / 255;
        R = I(:,:,1);
        G = I(:,:,2);
        B = I(:,:,3);

% Converting the image to HSV to
% obtain the Hue and Saturation Channels
        HSV = rgb2hsv(img);
        H = HSV(:,:,1);
        S = 1 - 3 * min(min(R, G), B) ./ (R + G + B );

% Intensity
        I = R+G+B/3;

% Creating the HSL Image
HSI = zeros(size(img));
HSI(:,:,1) = H;
HSI(:,:,2) = S;
HSI(:,:,3) = I;
subplot(1,2,1);imshow(img); title('Original Image');
subplot(1,2,2);imshow(HSI); title('HSI Image');
```

Output:

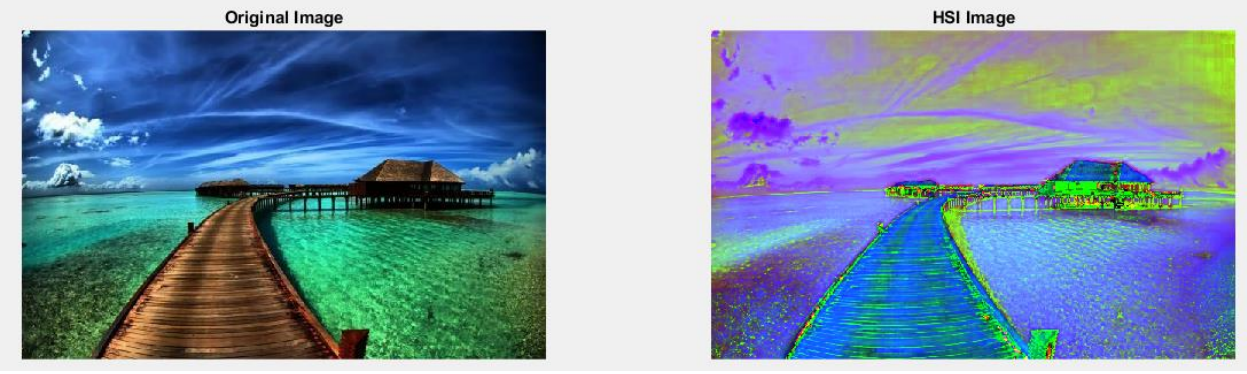

Fig 16: Output of the problem solution

17. Convert JPG to BMP, PNG and WMF.

Implementation: The matlab program to implement the problem is given below:

```
% Read the JPG image
jpgImage = imread('1.jpg');

% Convert JPG to BMP
bmpFilename = '1.bmp';
imwrite(jpgImage, bmpFilename, 'bmp');
```

```
bmpImage = imread(bmpFilename);

% Convert JPG to PNG
pngFilename = '1.png';
imwrite(jpgImage, pngFilename, 'png');
pngImage = imread(pngFilename);

% Convert JPG to WMF
fig = figure;
wmfFilename = '1.wmf';
saveas(fig, wmfFilename, 'meta');

% Display the images
subplot(1, 3, 1);imshow(jpgImage);title('JPG Image');
subplot(1, 3, 2);imshow(bmpImage);title('BMP Image');
subplot(1, 3, 3);imshow(pngImage);title('PNG Image');
```

Output:

Fig 17: Output of the problem solution

18. Convert an RGB image to JPG, BMP and PNG. Compare their sizes.

Implementation: The matlab program to implement the problem is given below:

```
% Read the RGB image
rgbImage = imread('1.jpg');

% Convert RGB to JPG and save the file
jpgFilename = '1.jpg';
imwrite(rgbImage, jpgFilename, 'jpg');

% Convert RGB to BMP and save the file
bmpFilename = '1.bmp';
imwrite(rgbImage, bmpFilename, 'bmp');

% Convert RGB to PNG and save the file
pngFilename = '1.png';
imwrite(rgbImage, pngFilename, 'png');
```

```
% Get the file sizes
jpgInfo = dir(jpgFilename);     bmpInfo = dir(bmpFilename);     pngInfo = dir(pngFilename);

% Display the file sizes
disp(['JPG File Size: ', num2str(jpgInfo.bytes), ' bytes']);
disp(['BMP File Size: ', num2str(bmpInfo.bytes), ' bytes']);
disp(['PNG File Size: ', num2str(pngInfo.bytes), ' bytes']);
```

Output:

Fig 18.1: Generated image of the problem solution

```
JPG File Size: 45015 bytes
BMP File Size: 698046 bytes
PNG File Size: 371860 bytes
```

Fig 18.2: Comparison redult of images sizes of the problem solution

19. Read all JPEG image from a directory and convert all images to PNG.

Implementation: The matlab program to implement the problem is given below:

```
% Specify the directory path
directory = 'D:/8th semister/matlab/New folder/image 2/';
% Get a list of all JPEG files in the directory
jpegFiles = dir(fullfile(directory, '*.jpg'));

% Loop through each JPEG file
for i = 1:numel(jpegFiles)
    % Read the JPEG image
    jpegFilename = fullfile(directory, jpegFiles(i).name);
    jpegImage = imread(jpegFilename);

     % Convert the JPEG image to PNG format
    [~, pngFilename, ~] = fileparts(jpegFilename);
    pngFilename = fullfile(directory, [pngFilename '.png']);
    imwrite(jpegImage, pngFilename, 'png');
end
disp('Conversion completed.');
```

Output:

Fig 19.1: Before conversion the folder of images

Fig 19.2: After conversion the folder of images

20. Find DCT of JPEG images.

Implementation: The matlab program to implement the problem is given below:

```
% Read the JPEG image
jpegImage = imread('image/img5.jpg');

% Convert the image to grayscale if needed
if size(jpegImage,3)==3
   grayscaleImage = rgb2gray(jpegImage);
end

% Perform DCT on the grayscale image
dctImage = dct2(grayscaleImage);
```

```
% Display the DCT coefficients
imshow(log(abs(dctImage)+1),[]);

title('DCT of JPEG Image');
```

Output:

Fig 20: Output of the problem solution