Table of Contents


```
%NAME: Vatsalya Chaubey
%INST: IIT, Bhubaneswar
%DATE: 19.09.2020
%CATEGORY: Btech
%BRANCH: Electronics and Communication
%Roll Number: 17EC01044
```

Image and Video Processing Assignment 1

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

Question 1: Seperate intensity and colour information using different colour models

```
img = imread('lena_color_256.tif');
img = double(img)/255;
black = zeros(size(img,1), size(img,2));

% Using the RGB colour model

% Seperating all the colour channels
red = cat(3, img(:,:,1), black, black);
green = cat(3, black, img(:,:,2), black);
blue = cat(3, black, black, img(:,:,3));

% showing all
figure('NumberTitle', 'off', 'Name', 'RGB color model');
subplot(2,2,1)
imshow(img)
title('Original Image')

subplot(2,2,2)
imshow(red)
title('Red Channel Image')
```

```
subplot(2,2,3)
imshow(green)
title('Green Channel Image')

subplot(2,2,4)
imshow(blue)
title('Blue Channel Image')


% Using the HSI model (Hue, Saturation and Intensity)

% Converting the RGB information to HSI
red = double(img(:,:,1))/255.0;
green = double(img(:,:,2))/255.0;
blue = double(img(:,:,3))/255.0;

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);
hue = hue/360;

sat = 1 - 3./(sum(img, 3)) .* min(img,[],3);

intensity = sum(img, 3)./3;

% Combining all the channels
HSI(:,:,1) = hue;
HSI(:,:,2) = sat;
HSI(:,:,3) = intensity;


% showing all
figure('NumberTitle', 'off', 'Name', 'HSI color model');
subplot(3,2,1)
imshow(img)
title('Original Image')

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

subplot(3,2,3)
imshow(sat)
title('Saturation Channel Image')

subplot(3,2,4)
imshow(intensity)
title('Intensity Channel Image')

subplot(3,2,5)
imshow(HSI)
title('Final combined HSI image')
```

```
% Converting RGB to CMY
img = imread('mandril_color.tif');
img = double(img)./255.0;

red = img(:,:,1);
green = img(:,:,2);
blue = img(:,:,3);

c = 1 - red;
m = 1 - green;
y = 1 - blue;

cmy_img = cat(3, c, m, y);

% showing all
figure('NumberTitle', 'off', 'Name', 'CMY model');
subplot(1,2,1)
imshow(img)
title('Original Image')

subplot(1,2,2)
imshow(cmy_img)
title('CMY Image')
```

Warning: Displaying real part of complex input.
Warning: Displaying real part of complex input.

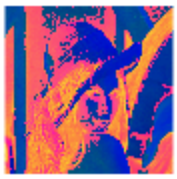
Original Image

Hue Channel Image

Saturation Channel Image

Intensity Channel Image

Final combined HSI image



Original Image

CMY Image

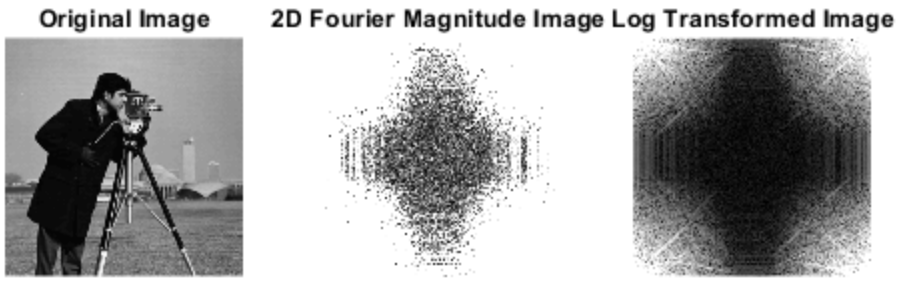
Question 2: Find the image negative for a greyscale image

```
grey_img = imread('lena_gray_256.tif');

% Finding negative by taking the complement
negative = 255 - grey_img;

% showing all
figure('NumberTitle', 'off', 'Name', 'Negative');
subplot(1,2,1)
imshow(grey_img)
title('Original Image')

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

Question 3: Find 2D fourier magnitude spectrum of an image

```
grey_img = imread('cameraman.tif');
```

```
grey_img = double(grey_img)/255.0;

M = size(grey_img,1);
N = size(grey_img,2);

% Performing 2D fourier transform as two successive 1D transforms as
 once
% over the rows and the other one over the columns

% First, over rows. Creating a _N_*_N_ weight matrix as we are taking
 a
% N-point DFT
n = 0:1:N-1;
k = 0:1:N-1;
weight_row = n' * k;
weight_row = (-2*pi*1i/N) .* weight_row;
weight_row = exp(weight_row);

% Secondly, now taking M-point DFT across the columns
m = 0:1:M-1;
k = 0:1:M-1;
weight_col = m' * k;
weight_col = (-2*pi*1i/M) .* weight_col;
weight_col = exp(weight_col);

% Taking M-point DFT of the previous N-point DFT
fourier2D = weight_row * (grey_img * weight_col);

% calculating the magntiude of the complex
magn = abs(fourier2D);

figure('NumberTitle', 'off', 'Name', '2D Fourier');
subplot(1,3,1)
imshow(grey_img)
title('Original Image')

subplot(1,3,2)
imshow(magn)
title('2D Fourier Magnitude Image')

% Applying log transformation
% This is done because the dynamic range of the output for Fourier
% Transform is very high and cannot be represented in 8 bits. So we
 squash
% the higher values using log transformation such that finer details
 become
% visible
c = 255/double((1+max(grey_img, [], 'all')));
log_transformed = uint8(log10(1 + magn)*c);

subplot(1,3,3)
imshow(log_transformed)
title('Log Transformed Image')
```



Question 4: Applying Gamma Transformation for correcting over and underexposure

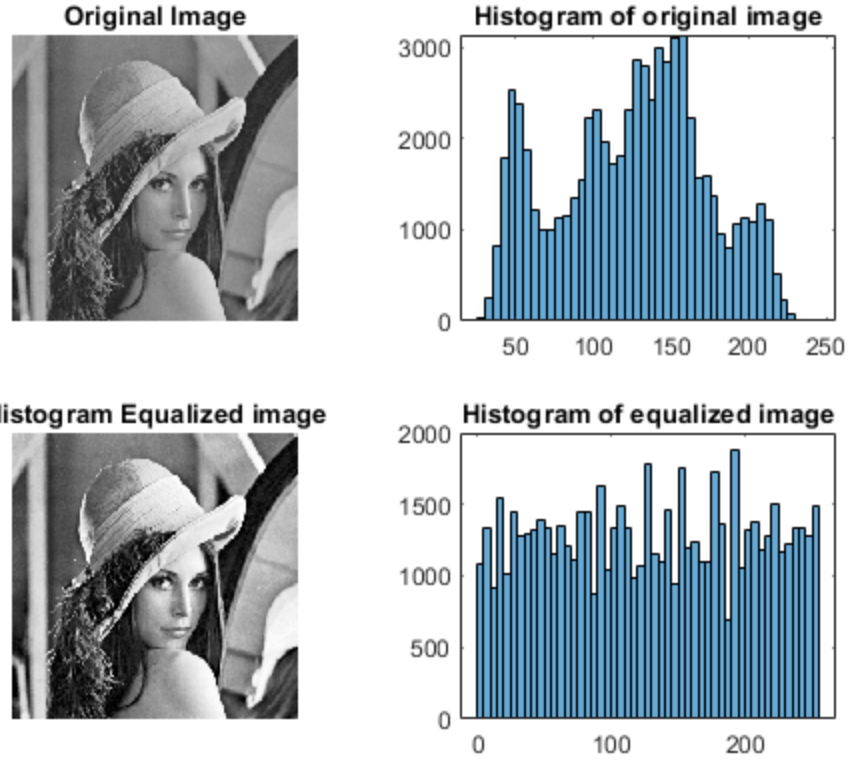
```
img = imread('cameraman.tif');
img = double(img)/255.0;

% Gamma Transformation: This is to increase or decrease the exposure
 of the
% image. Different values of gamma denote different levels of image
% exposure
gamma = 0.4;
gamma_transformed = img.^gamma;

% showing all
figure('NumberTitle', 'off', 'Name', 'Gamma Transformation');
subplot(1,3,1)
imshow(img)
title('Original Image')

subplot(1,3,2)
imshow(gamma_transformed)
title('Gamma = 0.4')

gamma = 1.5;
```

```
gamma_transformed = img.^gamma;

subplot(1,3,3)
imshow(gamma_transformed)
title('Gamma = 1.5')
```

Question 5: Histogram Equalization

```
img = imread('lena_gray_256.tif');

% Creating the frequency count and probability arrays
freq = zeros(256,1);
prob = zeros(256,1);

% size of the image
[m, n] = size(img);

% Calculating the frequency of each pixel value and its probability
for i=1:m
    for j=1:n
        pixel = img(i,j);
        freq(pixel+1) = freq(pixel+1) + 1;
    end
end
prob = freq./(m*n);
```

```
% Creating the new uniform probability array
new_prob = zeros(256,1);

% The output mappings array
out = zeros(256,1);
bins = 255;

% performing the calculations for the first pixel
new_prob(1) = freq(1)/(m*n);
out(1) = round(new_prob(1)*bins);

% Calculating the new probability and output mapping for each pixel
 value
for i=2:256
    freq(i) = freq(i) + freq(i-1);
    new_prob(i) = freq(i)/(m*n);
    out(i) = round(new_prob(i)*bins);
end

% Assigning new output to each pixel in the original image
HE = uint8(zeros(m,n));
for i=1:m
    for j=1:n
        HE(i,j) = out(img(i,j)+1);
    end
end

% showing all
figure('NumberTitle', 'off', 'Name', 'Histogram Equalization');
subplot(2,2,1)
imshow(img)
title('Original Image')

subplot(2,2,2)
histogram(img)
title('Histogram of original image')

subplot(2,2,3)
imshow(HE)
title('Histogram Equalized image')

subplot(2,2,4)
histogram(HE)
title('Histogram of equalized image')
```



Conclusion

The analysis presented in the first question where we use different color models and seperate the images into various parts can be very useful under some circumstances. Suppose, we want to understand the intensity of an image, we can directly convert to HSI to get the intensity values. Similarly, the other models can be used for other special purposes.

In the second part of the experiment, the technique to calculate the negative of an image is demonstrated. This technique is especially helpful when we want to highlight the darker potions of an image.

The fourier transform analysis of an image helps us understand the spectral nature of an image. We can clearly see that the values generated after taking the FFT have a very large dynamic range which cannot be represented on an image where each pixel can range from 0-255. So, for making the features in the FFT visible we need to squash the values obtained from FFT using log transformation.

The fourth assignment demonstrates the gamma transformation which helps in correcting the exposure of an image.

Finally, we perform histogram equalization in which we try assign equal number of pixels to each intensity value in an image. Doing this helps increase the contrast of the image and improves the overall image quality.

Published with MATLAB® R2020b