ENEL503 Computer Vision

Lab 1
Basic Image Representation and Processing with MATLAB

ENG 203
Due: 5:00pm, Wednesday, Feb. 7, 2024

Name: _Gustavo Da Costa Gomez_____

UCID: _30085980_____

Purpose

The purposes of this lab assignment are as follows:

1. Practice MATLAB image I/O techniques;
2. Manipulate MATLAB image show and plot formatting;
3. Analyze image characteristic values (κ) by implementing a Kappa Test Algorithm in MATLAB;
4. Analyze image similarity measures (σ) by implementing a Sigma Test Algorithm in MATLAB;
5. Become familiar with essential MATLAB programming skills.

Marks

Question	Mark allocation	Mark received
1	20	
2	20	
3	25	
4	25	
5	10	
Total	**100**	

General Instructions

To ensure consistent and efficient marking, a Word template will be provided for each ENEL 503 lab. Complete this Word template with your answers, MATLAB code, and plots as required. *Submit a PDF file titled by your full name* before the due date/time by email attachment to TA: **Muskan Sarvesh at <muskan.sarvesh1@ucalgary.ca>.**

Some questions require MATLAB code to be provided in related answers. Make sure that the code is commented sufficiently, but avoid being too verbose, in order to make the marking job for the TA as efficient as possible. Obscure code that is insufficiently or incorrectly commented will result in lost marks.

Equations in the report need to be represented in proper mathematical form using *MathType* or *MS Equation*. MathType equation editor is powerful and recommended. Hand-written math expressions are not encouraged.

MATLAB plots may be copied directly from the figure window of MATLAB by 'Edit > Copy Figure'. Then, you may paste the figure into the Word template. Color reports are encouraged due to the nature of this course.

1. (20) Practice image I/O, format transformation, size unification, and plotting with the set of six given color photos/images as follows. The images can be downloaded from course D2L site in the section: D2L/.../Lab Assignments/Lab1 Materials. It's noteworthy that formats and sizes of the given images may be different, and Fig12 is a composition (morphing) of Fig1 and Fig2.

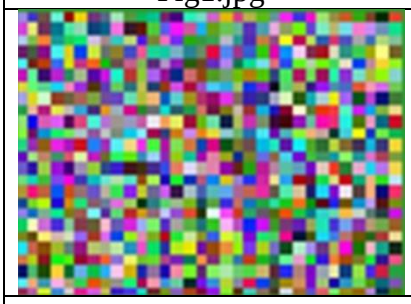
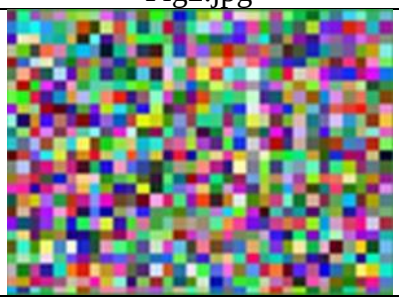
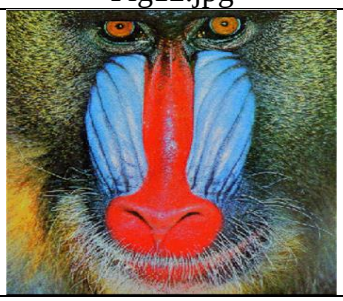
		
Fig1.jpg	Fig2.jpg	Fig12.jpg
		
Rand1.jpg	Rand2.jpg	Mandrill.png

Fig. 1 Images for processing

a) (2) Load the six color images from your ENEL503 Lab1 Directory into the MATLAB programming environment.

b) (3) Convert the given images into grayscale.

c) (15) Unify the sizes of the given images to 500 x 500 pixels using `imresize` and then save them onto your hard disk using similar file names. Then, plot the six normalized gray images, respectively, by a MATLAB program with suitable titles.

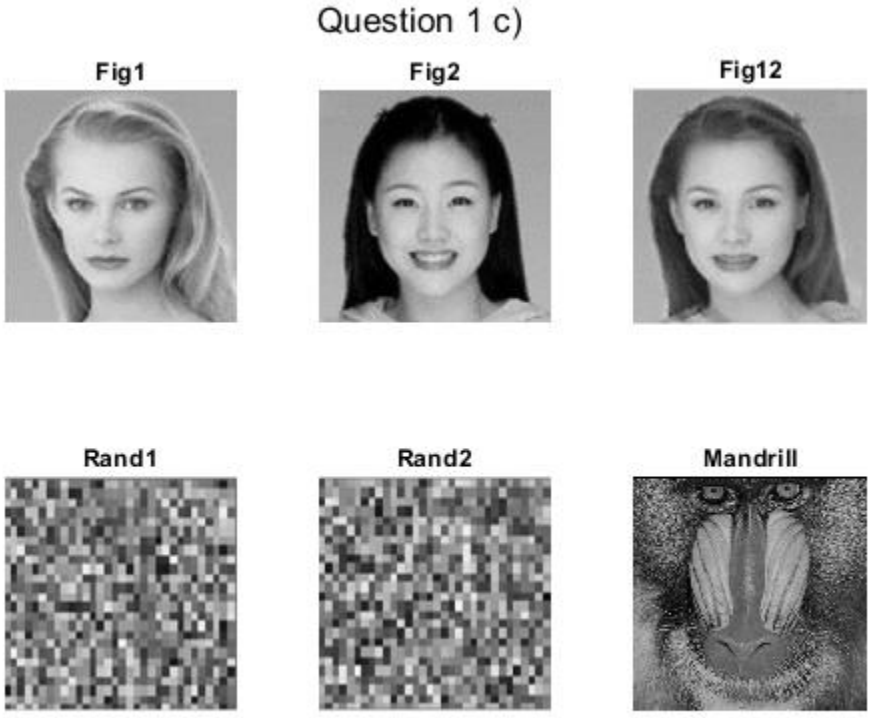
[**Hint:** In order to implement an iterative algorithm, use `Im = repmat({}, 6)` to initialize the date structure for holding the six target images such as `Im(1,1).image = imread('Fig1.jpg'); Im(1,6).image = imread('Mandrill.png'), etc.` [Other approaches may also be acceptable.]

---------- MATLAB code for Steps (a) to (c) ----------

```
%% ENEL 503 Lab 1
% Gustavo Da Costa Gomez, 30085980

%% Question 1
```

```
% Q1 a) - Load and display images
Im = repmat({}, 6);
Im(1,1).image = imread('Fig1.jpg');
Im(1,2).image = imread('Fig2.jpg');
Im(1,3).image = imread('Fig12.jpg');
Im(1,4).image = imread('Rand1.jpg');
Im(1,5).image = imread('Rand2.jpg');
Im(1,6).image = imread('Mandrill.png');

% Display images in a subplot
figure
subplot(2,3,1), imshow(Im(1,1).image), title('Fig1');
subplot(2,3,2), imshow(Im(1,2).image), title('Fig2');
subplot(2,3,3), imshow(Im(1,3).image), title('Fig12');
subplot(2,3,4), imshow(Im(1,4).image), title('Rand1');
subplot(2,3,5), imshow(Im(1,5).image), title('Rand2');
subplot(2,3,6), imshow(Im(1,6).image), title('Mandrill');
sgtitle('Question 1 a)');

% Q1 b) - Convert images to grayscale and display
for i = 1:6
    Im(1,i).image = rgb2gray(Im(1,i).image);
end
figure
subplot(2,3,1), imshow(Im(1,1).image), title('Fig1');
subplot(2,3,2), imshow(Im(1,2).image), title('Fig2');
subplot(2,3,3), imshow(Im(1,3).image), title('Fig12');
subplot(2,3,4), imshow(Im(1,4).image), title('Rand1');
subplot(2,3,5), imshow(Im(1,5).image), title('Rand2');
subplot(2,3,6), imshow(Im(1,6).image), title('Mandrill');
sgtitle('Question 1 b)');

% Q1 c) - Resize images and display
for i = 1:6
    Im(1,i).image = imresize(Im(1,i).image, [500, 500]);
end
figure
subplot(2,3,1), imshow(Im(1,1).image), title('Fig1');
subplot(2,3,2), imshow(Im(1,2).image), title('Fig2');
subplot(2,3,3), imshow(Im(1,3).image), title('Fig12');
subplot(2,3,4), imshow(Im(1,4).image), title('Rand1');
subplot(2,3,5), imshow(Im(1,5).image), title('Rand2');
subplot(2,3,6), imshow(Im(1,6).image), title('Mandrill');
sgtitle('Question 1 c)');

% Save grayscale images
imwrite(Im(1,1).image, 'Fig1_gs.jpg');
imwrite(Im(1,2).image, 'Fig2_gs.jpg');
imwrite(Im(1,3).image, 'Fig12_gs.jpg');
imwrite(Im(1,4).image, 'Rand1_gs.jpg');
imwrite(Im(1,5).image, 'Rand2_gs.jpg');
imwrite(Im(1,6).image, 'Mandrill_gs.png');
```

---------- Plots of the six gray images generated ----------



---

2. (20) In order to get familiar with image representation technologies and dataset structures, try to convert *Mandrill.png* into the following 7 formats: C/G/B/SC/R1/G2/B3, by developing a MATLAB function. The color format SC denotes the indexed pseudocolor scheme.

[**Hint:** Refer to L3-Section 4.2. The color image may be plotted for two times in the beginning of each row in a 2 x 4 subplot.]

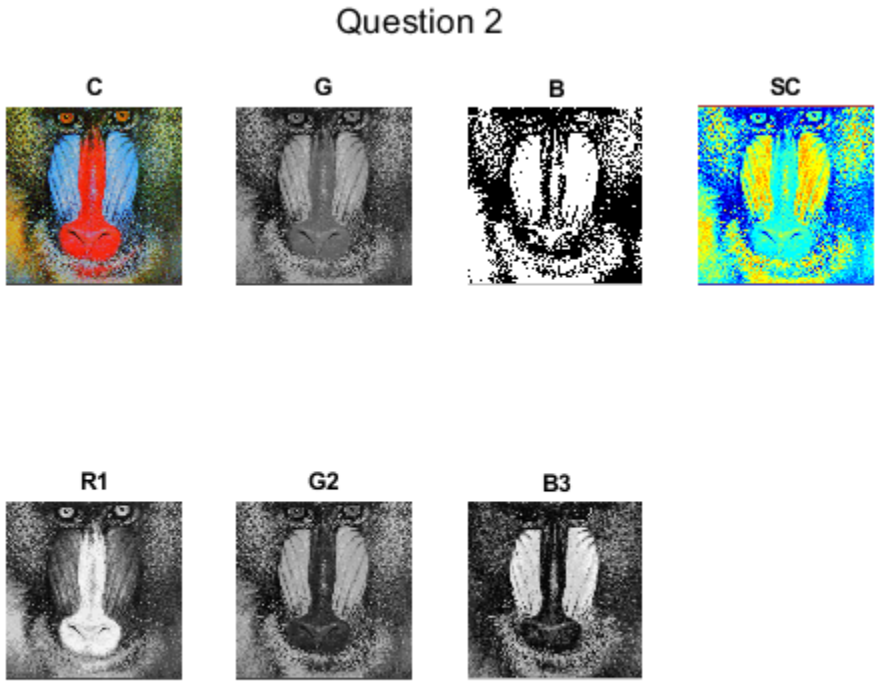
---------- MATLAB code (15) ----------

```
%% Question 2

% Process the Mandrill image using the CGBSCR1G2B3 function
Im = CGBSCR1G2B3('Mandrill.png');

% Display processed images
figure
subplot(2, 4, 1), imshow(Im(1,1).image);
subplot(2, 4, 2), imshow(Im(1,2).image);
subplot(2, 4, 3), imshow(Im(1,3).image);
subplot(2, 4, 4), imshow(Im(1,4).image, jet(20));
subplot(2, 4, 5), imshow(Im(1,5).image);
subplot(2, 4, 6), imshow(Im(1,6).image);
subplot(2, 4, 7), imshow(Im(1,7).image);
```

```
sgtitle('Question 2');
```

```
function [Im] = CGBSCR1G2B3(image)
% CGBSCR1G2B3 - Color and Image Processing Function
%
% Syntax:
%   [Im] = CGBSCR1G2B3(image)
%
% Input:
%   image - The input image file path or matrix
%
% Output:
%   Im - A cell array containing various processed versions of the input image
%
% Description:
%   This function takes an input image and performs several color and
%   image processing operations to generate different representations
%   of the original image.
%
% Processing Steps:
%   1. Read the input image.
%   2. Convert the image to grayscale.
%   3. Binarize the grayscale image.
%   4. Apply grayscale slicing with 20 levels.
%   5. Extract the red channel of the original image.
%   6. Extract the green channel of the original image.
%   7. Extract the blue channel of the original image.
%
% Example:
%   img = 'path/to/your/image.jpg';
%   processedImages = CGBSCR1G2B3(img);
%
% Author: Gustavo Da Costa Gomez

% Step 1: Read the input image
Im = repmat({}, 7);
Im(1,1).image = imread(image);

% Step 2: Convert the image to grayscale
Im(1,2).image = rgb2gray(Im(1,1).image);

% Step 3: Binarize the grayscale image
Im(1,3).image = imbinarize(Im(1,2).image);

% Step 4: Apply grayscale slicing with 20 levels
Im(1,4).image = grayslice(Im(1,2).image, 20);

% Step 5: Extract the red channel of the original image
Im(1,5).image = Im(1,1).image(:,:,1);

% Step 6: Extract the green channel of the original image
Im(1,6).image = Im(1,1).image(:,:,2);

% Step 7: Extract the blue channel of the original image
Im(1,7).image = Im(1,1).image(:,:,3);
end
```

---------- Plots of the seven layers/components of Mandrill (5) ----------



---

3. (25) Design a MATLAB function to analyze the *characteristic values* (κ) of the six gray-scale images as obtained and normalized in Problem (1). The algorithm will be based on the mathematical models in L3-Section 2.

[**Hint:** Reuse similar data structures designed for solving Problem (1).]

---------- MATLAB code (25) ----------

```
%% Question 3

% Load grayscale images for Question 3
Im = repmat({}, 6);
Im(1,1).image = imread('Fig1_gs.jpg');
Im(1,2).image = imread('Fig2_gs.jpg');
Im(1,3).image = imread('Fig12_gs.jpg');
Im(1,4).image = imread('Rand1_gs.jpg');
Im(1,5).image = imread('Rand2_gs.jpg');
Im(1,6).image = imread('Mandrill_gs.png');

% Calculate Characteristic Values and display
CV = CharacteristicValues(Im);
```

```
disp(CV);

function [CV] = CharacteristicValues(Im)
% CharacteristicValues - Calculate Characteristic Values from Images
%
% Syntax:
%   [CV] = CharacteristicValues(Im)
%
% Input:
%   Im - A cell array containing images
%
% Output:
%   CV - Characteristic Values vector for each image in Im
%
% Description:
%   This function takes a cell array of images and calculates
%   Characteristic Values for each image.
%
% Parameters:
%   X, Y - Dimensions for the temporary image matrix
%   CV - Characteristic Values vector for each image
%   Imk - Temporary image matrix for each iteration
%   Sum - Accumulator for pixel values in the image
%
% Example:
%   img1 = imread('image1.jpg');
%   img2 = imread('image2.jpg');
%   images = {struct('image', img1), struct('image', img2)};
%   charValues = CharacteristicValues(images);
%
% Author: Gustavo Da Costa Gomez

% Set temporary image dimensions
X = 500;
Y = X;

% Initialize the Characteristic Values vector
CV = zeros(1, length(Im));

% Iterate through each image in the cell array
for k = 1:length(Im)
    % Extract the image matrix
    Imk = Im(1, k).image;

    % Initialize the pixel sum accumulator
    Sum = 0;

    % Iterate through each pixel in the image
    for i = 1:X
        for j = 1:Y
            % Accumulate pixel values
            Sum = Sum + double(Imk(i, j));
        end
        % Calculate Characteristic Value for the current row
        CV(k) = Sum / (255 * X * Y);
    end
```

```
end

end
```

---------- Show test results of kappas (5) ----------

```
>> Lab1
   0.6298    0.4850    0.5490    0.5069    0.5026    0.4024
```

---

4. (25) Design a MATLAB function for implementing *image similarity* (σ) analyses (slide L3-14) on each pair of the six gray-scale images as obtained and normalized in Problem (1). Multilayer (6 x 6) iterative code, rather than linear repetitions for the 36 pairs of them, are expected.

[**Hint:** a) Reuse similar data structures designed for solving Problem (1).

b) Consider the following code for implementing `abs(x-y)` in the equation of σ:

```
Sum = Sum + abs(double(Im1(i,j)) - double(Im2(i,j)));
```
]

---------- MATLAB code (25) ----------

```
%% Question 4

% Calculate Similarity Matrix and display
Sim = SimilarityDetermination(Im);
disp(Sim);

function [Sim] = SimilarityDetermination(Im)
% SimilarityDetermination - Calculate Image Similarity Matrix
%
% Syntax:
%   [Sim] = SimilarityDetermination(Im)
%
% Input:
%   Im - A cell array containing images
%
% Output:
%   Sim - Image Similarity Matrix
%
% Description:
%   This function takes a cell array of images and calculates the
%   similarity matrix between each pair of images.
%
% Parameters:
%   X, Y - Dimensions for the temporary image matrix
%   Sim - Image Similarity Matrix
%   Imk - Temporary image matrix for each iteration
%   Sum - Accumulator for pixel differences between images
%
% Example:
%   img1 = imread('image1.jpg');
%   img2 = imread('image2.jpg');
%   images = {struct('image', img1), struct('image', img2)};
```

```
%   similarityMatrix = SimilarityDetermination(images);
%
% Author: Gustavo Da Costa Gomez

% Set temporary image dimensions
X = 500;
Y = X;

% Initialize the Image Similarity Matrix
Sim = zeros(length(Im), length(Im));

% Initialize the temporary image matrix
Imk = zeros(X, Y);

% Iterate through each pair of images
for k = 1:length(Im)
    for l = 1:length(Im)
        % Extract the image matrices
        Imk = Im(1, k).image;

        % Initialize the pixel difference accumulator
        Sum = 0;

        % Iterate through each pixel in the images
        for i = 1:X
            for j = 1:Y
                % Accumulate absolute pixel differences
                Sum = Sum + abs(double(Im(1, l).image(i, j)) - double(Imk(i, j)));
            end
            % Calculate similarity score for the current row
            Sim(k, l) = 1 - Sum / (255 * X * Y);
        end
    end
end

end
```

---------- Plots of the analysis results of 6 x 6 similarities (5) ----------

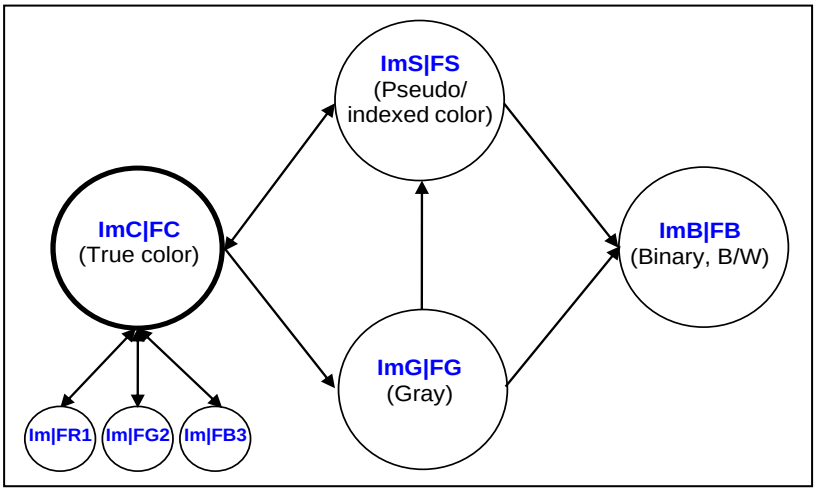
```
>> Lab1
   1.0000    0.8042    0.8861    0.7794    0.7791    0.7337
   0.8042    1.0000    0.8563    0.7233    0.7279    0.7302
   0.8861    0.8563    1.0000    0.7797    0.7762    0.7580
   0.7794    0.7233    0.7797    1.0000    0.7941    0.7817
   0.7791    0.7279    0.7762    0.7941    1.0000    0.7878
   0.7337    0.7302    0.7580    0.7817    0.7878    1.0000
```

---

5. (10) Complete the summary of the framework of MATLAB color scheme transformations in the following figure by referring to the slides in Lectures 3 and 2, as well as by surveying the related MATLAB toolboxes. Fill in the correct MATLAB function for each directed edge. Note

there may be multiple built-in function for a certain transformation, and some edges are bidirectional.

[**Hint:** Double click on the figure below to open it for inserting your answers.]



---------- Plots of expected answers (10) ----------

Image Type		Function
Converting To	**Converting From**	-
ImC|FC	ImS|FS	rgb2ind
ImS|FS	ImC|FC	ind2rgb
ImC|FC	ImG|FG	rgb2gray
ImG|FG	ImS|FS	gray2ind
ImS|FS	ImB|FB	im2bw
ImG|FG	ImB|FB	imbinarize
ImC|FC	Im|FR1	ImC(:, :, 1)
ImC|FC	Im|FG2	ImC(:, :, 3)
ImC|FC	Im|FB3	ImC(:, :, 2)
Im|FR1*	ImC|FC	`cat(3, ImC(:, :, 1), ImC(:, :, 3), ImC(:, :, 2));`
Im|FG2*	ImC|FC	`cat(3, ImC(:, :, 1), ImC(:, :, 3), ImC(:, :, 2));`
Im|FB3*	ImC|FC	`cat(3, ImC(:, :, 1), ImC(:, :, 3), ImC(:, :, 2));`

*Note: to convert individual R/G/B components back to true colour, all three components are needed

---