ENEL503 Computer Vision

Lab 2
Mathematical, Geometrical and Histogram Operations on Images

ENG 203
Due: 5:00pm, Wednesday, Feb. 21, 2024

Name: __Gustavo Da Costa Gomez_____

UCID: __30085980_____

Purpose

The purposes of this lab assignment are as follows:

1. Practice MATLAB image processing by math, geometric, and tonicity operations;
2. Apply typical principles and algorithms for image processing and analytic ability;
3. Improve programming skills for computer vision in MATLAB;
4. Become familiar with MATLAB toll boxes for real-world problem solving.

Marks

Question	Mark allocation	Mark received
1	20	
2	30	
3	20	
4	30	
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 hard copy by the due date in the assignment drop boxes on the second floor of ICT Building. Also submit an electronic copy of your lab report by email attachment to the TA, Thoshara Malathinawar at thosharamalathinawar@ucalgary.ca.

Some questions require for MATLAB code to be inserted. 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 the equation editor in Word (under Insert > Object > Microsoft equation). The MathType equation editor is highly recommended. Hand written math expressions are not expected.

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

1. (10) Let ImG, ImC, ImR1, ImG2, and ImB3 represent a gray, color, red, green, and blue representation of an arbitrary image, respectively. Try to prove ImG = ImGs where:

$$\text{ImGs} = 0.2989*\text{ImR1} + 0.5870*\text{ImG2} + 0.1140*\text{ImB3} \quad (1)$$

$$\text{ImG} = \text{rgb2gray (ImC)} \quad (2)$$

by a MATLAB program on the given color image *Mandrill.png* as provided in the Lab 2 Materials site on D2L.

[**Hint:** The proof may be given by imperially showing that the difference of characteristic values, $\delta(\text{ImG}, \text{ImGws}) \rightarrow 0$, based on your solution for Question 4 in Lab 1.]

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

```
%% ENEL 503 Lab 2
% Gustavo Da Costa Gomez, 30085980

clc
clear
close all

%% Question 1

% Load an image using a custom function CGBSCR1G2B3 and convert it to grayscale
Im = CGBSCR1G2B3('Mandrill.png');
ImC = Im(1,1).image;
ImG = rgb2gray(ImC);

% Extract color channels (Red, Green, Blue) from the original image
ImR1 = Im(1,5).image;
ImG2 = Im(1,6).image;
ImB3 = Im(1,7).image;

% Create grayscale image using weighted combination of color channels
ImGs = 0.2989 * ImR1 + 0.5870 * ImG2 + 0.1140 * ImB3;

% Display original grayscale image and the computed grayscale image
Im_compare = repmat({}, 2);
Im_compare(1,1).image = ImG;
Im_compare(1,2).image = ImGs;

Sim = SimilarityDetermination(Im_compare);
disp("Question 1 Output: " + newline);
disp(Sim);

figure(1)
subplot(1,2,1), imshow(ImG), title('ImG');
subplot(1,2,2), imshow(ImGs), title('ImGs');
sgtitle('Question 1 Images');

function [Im] = CGBSCR1G2B3(image)
% CGBSCR1G2B3 - Color and Image Processing Function
%
```

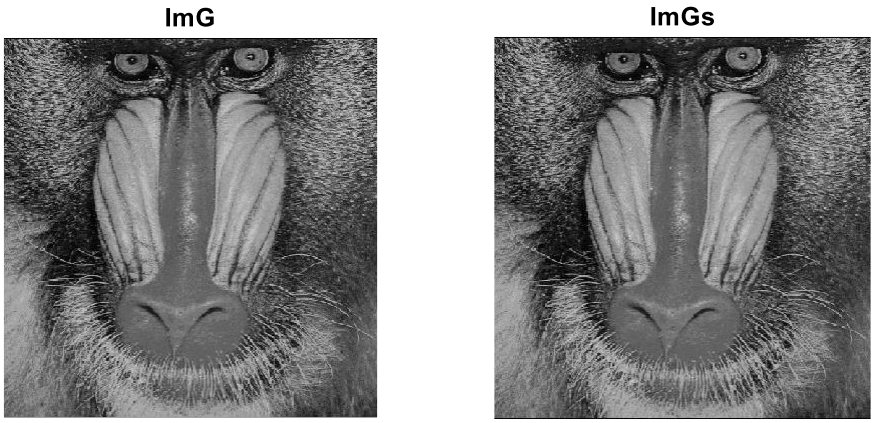
```
% Syntax:
%   [Im] = CGBSCR1G2B3(image)
%
% Input:
%   image - The input image file path or matrix
%
% Output:
%   Im - A cell array containing various processed versions of the input image
%
% Description:
%   This function takes an input image and performs several color and
%   image processing 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

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 ImG, ImGs, and the difference value (5) ----------

Question 1 Images



Question 1 Output:

```
   1.0000    0.9984
   0.9984    1.0000
```

---

2. (30) According to the principle of mathematical operations, any image may be processed at either the frame level or pixel level, fundamentally the later. Apply this principle to image morphing technologies in MATLAB for merging the given photos *Im1.jpg* and *Im2.jpg* in order to generate a series of fusions of their sequential morphs.

a) (10). Morphing at frame level according to slide L4-40, i.e.:

$$\text{ImMorphSeries|FC} = \mathop{R}_{k=1}^{n} (\text{Im2|FC} + (1 - k/n) * (\text{Im1|FC-Im2|FC})) \quad (3)$$

using the algebraic operation of Eq. 3 to generate the series of images morphing in MATLAB.

---------- MATLAB code (7) ----------

```
%% Question 2

% Q2 a) - Morphing between two images
Im21 = imread('Im21.jpg');
Im22 = imread('Im22.jpg');

Im_morph = repmat({}, 5);
n = 5;

figure(2)
for k = 1:5
    % Linear interpolation between two images
    Im = double(Im22) + (1-k/n)*(double(Im21) - double(Im22));
    Im = min(max(Im, 0), 255);
    Im_morph(1,k).image = uint8(Im);

    % Display morphed images
    subplot(1,n,k), imshow(Im_morph(1,k).image), title('Image');
    sgtitle("Question 2 a)");
end
```

---------- Plots of the 4 morphing series from Im1 to Im5 (3) ----------

Question 2 a)

---

b) (20). Design an equivalent program to implement the same morphing algorithm of (a) by pixel-by-pixel operations of serial morphing. Report your code in MATLAB and plot the testing result.

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

```
% Q2 b) - Morphing between two color images
Im21 = CGBSCR1G2B3('Im21.jpg');
Im22 = CGBSCR1G2B3('Im22.jpg');

% Extract color channels from each image
Im21R = Im21(1, 5).image;
Im21G = Im21(1, 6).image;
```

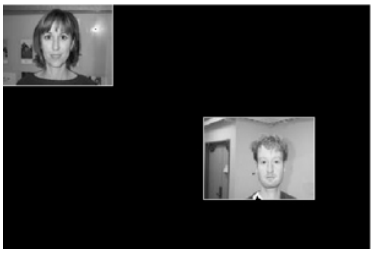
```
Im21B = Im21(1, 7).image;

Im22R = Im22(1, 5).image;
Im22G = Im22(1, 6).image;
Im22B = Im22(1, 7).image;

ImR = zeros(240, 160);
ImG = zeros(240, 160);
ImB = zeros(240, 160);

n = 5;

figure(3)
for k = 1:n
    for i = 1:240
        for j = 1:160
            % Linear interpolation for each pixel in color channels
            ImR(i, j) = double(Im22R(i, j)) + (1-k/n)*(double(Im21R(i, j)) -
double(Im22R(i, j)));
            ImR(i, j) = min(max(ImR(i, j), 0), 255);

            ImG(i, j) = double(Im22G(i, j)) + (1-k/n)*(double(Im21G(i, j)) -
double(Im22G(i, j)));
            ImG(i, j) = min(max(ImG(i, j), 0), 255);

            ImB(i, j) = double(Im22B(i, j)) + (1-k/n)*(double(Im21B(i, j)) -
double(Im22B(i, j)));
            ImB(i, j) = min(max(ImB(i, j), 0), 255);
        end
    end

    % Combine color channels to create the morphed image
    Im_morph = cat(3, ImR, ImG, ImB);

    % Display morphed color images
    subplot(1, n, k), imshow(uint8(Im_morph)), title(['Image ' num2str(k)]);
    sgtitle("Question 2 b)");
end
----------------------------------------------------------------------------------------
```

---------- Plots of the 4 morphing steps from Im1 to Im5 (5) ----------

Question 2 b)

---

3. (20) Design a MATLAB function for implementing object detection by *spatial histograms* (SH) on the image *HS_Test.tif* as given below. Once the histogram is generated, try to analyze and locate these two objects by their coordinates at the upper-left and lower-right corners, i.e., O1((x1, y1),(x2, y2)) and O2((x3, y3),(x4, y4)), by the approximate values shown in it.



[**Hint:** Refers to L5(35-39).]

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

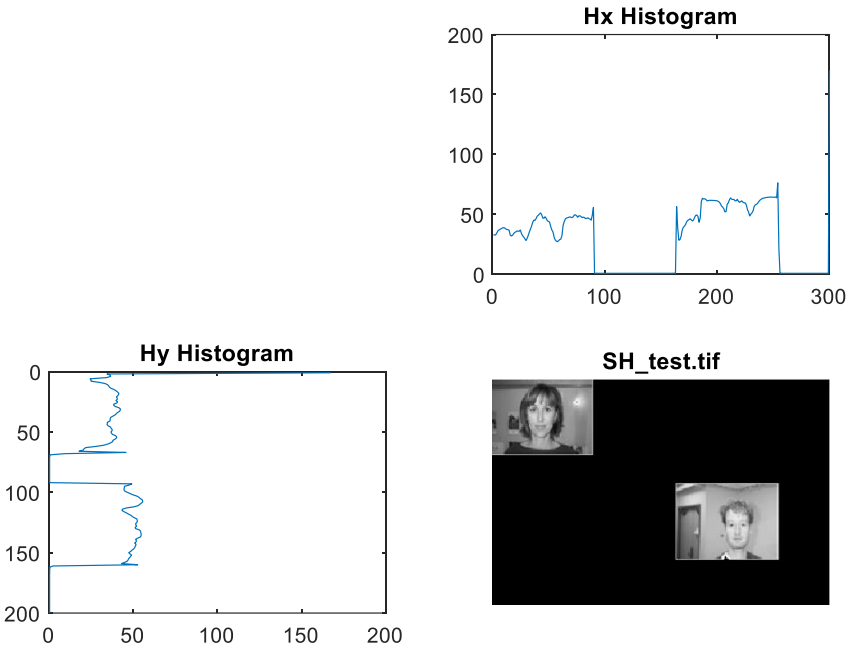
```
%% Question 3

% Load an image for spatial histogram analysis
Im_SH_test = imread('SH_Test.tif');

% Calculate spatial histograms along rows (Hx) and columns (Hy)
[Hx, Hy] = SpatialHistogram(Im_SH_test);

% Display spatial histograms and the original image
figure(4)
subplot(2,2,2), plot(1:length(Hx), Hx), title('Hx Histogram');
subplot(2,2,3), plot(Hy, 1:length(Hy)), title('Hy Histogram');
set(gca,'ydir','reverse'); % Invert y-axis to match the image orientation
subplot(2,2,4), imshow(Im_SH_test), title('SH\_test.tif');
sgtitle('Question 3');

% Initialize variables to store coordinates
[x1, x2, x3, x4, y1, y2, y3, y4] = deal(0);

% Identify peaks in Hx and Hy histograms
Hx_ones = Hx > 1;
Hy_ones = Hy > 1;

% Extract coordinates for images peaks in Hx histogram
for i = 2:length(Hx_ones)-1
    if Hx_ones(i) == Hx_ones(i-1)
        if x1 == 0
            x1 = i-1;
        end
    end
    if Hx_ones(i) ~= Hx_ones(i-1)
        if x2 == 0
            x2 = i-1;
        elseif x2 > 0
            if x3 == 0
                x3 = i-1;
            else
                x4 = i-1;
            end
        end
    end
end
```

```
% Extract coordinates for images peaks in Hy histogram
for i = 2:length(Hy_ones)-1
    if Hy_ones(i) == Hy_ones(i-1)
        if y1 == 0
            y1 = i-1;
        end
    end
    if Hy_ones(i) ~= Hy_ones(i-1)
        if y2 == 0
            y2 = i-1;
        elseif y2 > 0
            if y3 == 0
                y3 = i-1;
            else
                y4 = i-1;
            end
        end
    end
end

% Display the extracted coordinates
disp(newline)
disp("The Question 3 coordinates are O1((" + x1 ...
    + "," + y1 ...
    + "),(" + x2 ...
    + "," + y2 ...
    + ")) and O2((" + x3 ...
    + "," + y3 ...
    + "),(" + x4 ...
    + "," + y4 + ")).")

function [Hy, Hx] = SpatialHistogram(Im)
% SpatialHistogram computes spatial histograms along the rows and columns of an
image.
% The function calculates the average intensity values for each row (Hx) and each
% column (Hy) of the input image.

% INPUTS:
%   - Im: input grayscale image (2D array)

% OUTPUTS:
%   - Hy: histogram along the rows (1D array)
%   - Hx: histogram along the columns (1D array)

% Get the size of the input image
[X, Y] = size(Im);

% Initialize arrays to store histograms along rows (Hx) and columns (Hy)
Hx = zeros(1, X);
Hy = zeros(1, Y);

% Compute histogram along rows (Hx)
for i = 1:X
    Sum = 0;
    % Loop through each pixel in the current row
```

```
    for j = 1:Y
        % Accumulate intensity values in the current row
        Sum = Sum + double(Im(i, j));
    end
    % Calculate the average intensity value for the current row
    Hx(1, i) = Sum / Y;
end

% Compute histogram along columns (Hy)
for i = 1:Y
    Sum = 0;
    % Loop through each pixel in the current column
    for j = 1:X
        % Accumulate intensity values in the current column
        Sum = Sum + double(Im(j, i));
    end
    % Calculate the average intensity value for the current column
    Hy(1, i) = Sum / X;
end

end
```

---------- Plot the spatial histograms and analyze the location of the two objects (5) ----------

Question 3



The Question 3 coordinates are O1((1,1),(90,68)) and O2((163,92),(255,161)).

---

4. (30) In object detection technologies, there is often a need to remove the shadow from the real object in an image in order to actually locate the position of the object in the frame. For this purpose, the degradation of the target's image quality is not the major concern as shown in the given case study on *Car1.jpg*.

Develop a MATLAB function for detecting the shadow around the car and try to remove the shadow by resetting its pixel value as the same of the surrounding background. Your program will need to automatically test the average shadow and background values in sample boxes of 4 x 4 and 10 x 10 pixels, respectively.

[**Hint:** The spatial histogram technology as implemented in Solution 3 may be applied to guide the detection of the average shadow and background values at suitable sample places.]

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

```
%% Question 4

% Load an image for shadow removal
Im_Car = imread('Car1.jpg');

% Call removeShadow function to remove shadows from the image
Im_Car_new = removeShadow(Im_Car);

function removedShadowImage = removeShadow(image)
% removeShadowImage = removeShadow(image)
% This function removes the shadow from the input RGB image.

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

% Define the sample box sizes for shadow and background
shadowBoxSize = 4;
backgroundBoxSize = 10;

% Estimate the average pixel value of the shadow
shadowAvgValue = estimateAverageValue(grayImage, shadowBoxSize);

% Estimate the average pixel value of the background
backgroundAvgValue = mean(estimateAverageValue(grayImage, backgroundBoxSize),
"All");

% Identify the shadow region (assumption: shadow is darker than the background)
shadowMask = 40 > shadowAvgValue;
```

```
% Create a copy of the original image
removedShadowImage = image;

% Replace shadow pixel values with background pixel values
for i = 1:size(image, 1)
    for j = 1:size(image, 2)
        if shadowMask(i, j)
            % Replace RGB values at shadow pixels with background RGB values
            removedShadowImage(i, j, 1) = backgroundAvgValue;
            removedShadowImage(i, j, 2) = backgroundAvgValue;
            removedShadowImage(i, j, 3) = backgroundAvgValue;
        end
    end
end

% Convert the modified image to grayscale
removedShadowGrayImage = rgb2gray(removedShadowImage);

% Display the result
figure;
subplot(2, 2, 1), imshow(image), title('Original Image');
subplot(2, 2, 2), imshow(shadowMask, []), title('Shadow Mask');
subplot(2, 2, 3), imshow(removedShadowImage), title('Image with Removed Shadow');
subplot(2, 2, 4), imshow(removedShadowGrayImage), title('Greyscale Image with
Removed Shadow');
sgtitle("Question 4");

end

function avgValue = estimateAverageValue(image, boxSize)
    % Estimate the average pixel value within a sample box
    % INPUTS:
    %   - image: the input image (2D array)
    %   - boxSize: size of the sample box for averaging

    % Get the dimensions of the input image
    [rows, columns] = size(image);

    % Initialize a matrix to store the average pixel values
    avgValue = zeros(rows, columns);

    % Loop through the image in boxSize-sized steps
    for i = 1:rows/boxSize
        for j = 1:columns/boxSize
            % Calculate the starting indices of the current sample box
            index_y = (boxSize * i) - (boxSize - 1);
            index_x = (boxSize * j) - (boxSize - 1);

            % Extract the current sample box using imcrop
            box = imcrop(image, [index_x index_y boxSize-1 boxSize-1]);

            % Calculate the mean of all pixel values in the sample box
            avgValue(index_y:index_y+boxSize-1, index_x:index_x+boxSize-1) =
       mean(box, 'all');
        end
```

```
    end
end
```

---------- Plots of the results color and gray pairs as follows (5) ----------

Question 4

Original Image

Shadow Mask

Image with Removed Shadow

Greyscale Image with Removed Shadow