ENEL503 Computer Vision

Lab 3

Neighborhood, Morphological, and Filtering Operations on Images

Instructor: Dr. Yingxu Wang
TA: Thoshara B. Mohottalalage (Thosharamalathinawar@ucalgary.ca)
Due: 5:00pm, Wednesday, March 15, 2023

Name: _____Christina Truong_____

UCID: _____30064426_____

Purposes

The purposes of this lab assignment are as follows:

- Practice MATLAB image processing by geometrical, morphological, and filtering operations
- Apply typical principles and algorithms for image processing and analyses;
- Improve programming skills for computer vision in MATLAB;
- Become familiar with MATLAB toll boxes for real-world problem solving.

Marks

Question	Mark allocation	Mark received
1	20	
2	15	
3	20	
4	10	
5	35	
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 an electronic copy of your lab report by email attachment to TA: Thoshara B. Mohottalalage at Thosharamalathinawar@ucalgary.ca.

Some questions ask 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 also 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. (20) The key concept and mathematical model of *masks (M)* play important roles in neighborhood, convolution, correlation, filter, and morphological operations. Try to empirically prove or test the following principles by your own MATLAB programs where the textbook might be incorrect.

a) (10) Test the conceptual differences of image *Convolution* and *Correlation* using the codes in Section 4 of Lecture 6. Discuss on what conditions the MATLAB correlation and convolution operations will be equivalent.

---------- Q1.a: MATLAB code (4), plots (2), and analyses (4) ----------

```
%% Ex1
Im = imread('Lenna.jpg');
Mask = [1/9 1/9 1/9; 1/9 1/9 1/9; 1/9 1/9 1/9];
% Uniform mask
Conv = imfilter (Im, Mask,'full','conv');
Corl = imfilter (Im, Mask,'full','corr');

subplot(1,3, 1), imshow(Im), title("Original");
subplot(1,3, 2), imshow(Conv), title("Convulution");
subplot(1,3, 3), imshow(Corl), title("Correlation");
```

Correlation is the same as convolution without the mirroring (flipping) of the mask before the sums of products are computed. The difference between using correlation and convolution in 2D neighborhood processing operations is often irrelevant because many popular masks used in image processing are symmetrical around the origin. So when masks are symmetrical around the origin, convolution and correlation operations will be equivalent

---

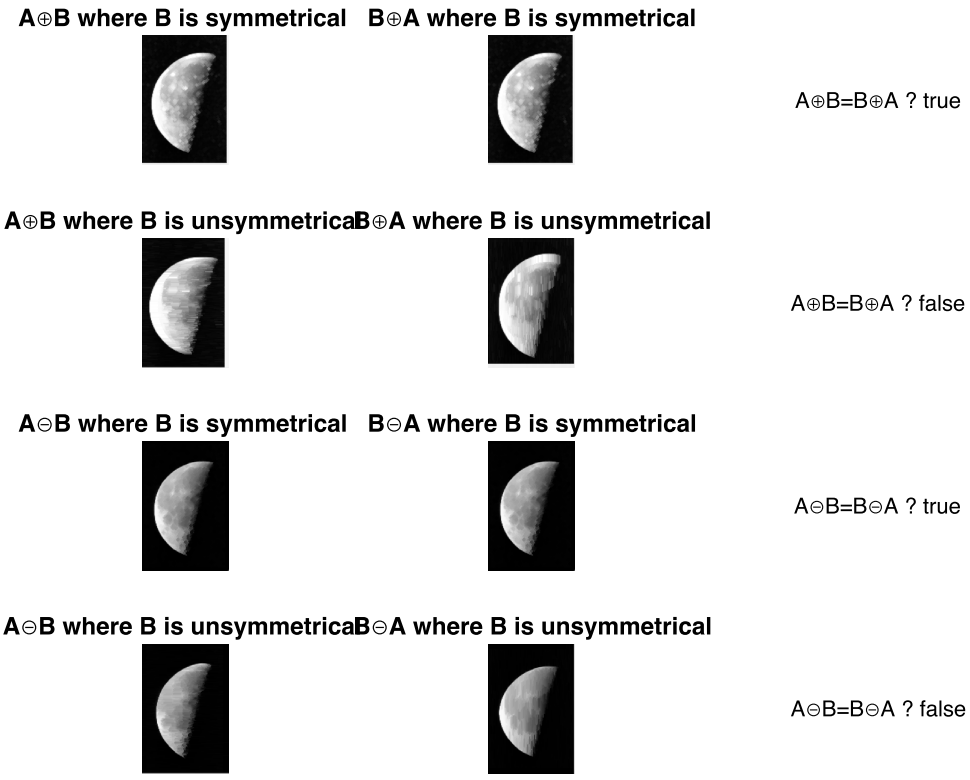
b) (5) Using the case studies in L7-15 and L7-17, test if the basic morphological operations *dilation* and *erosion* are *commutative*, i.e., if $A \ B = B \ A$ and $A \ B = B \ A$?

---------- Q1.b: MATLAB code, plots, and analysis (5) ----------

```
A = imread('Moon.tif');
A = uint8(rgb2gray(A));

% Create a symmetric structuring element
B_symmetric = strel('disk', 2);
% Create an unsymmetric structuring element
B_unsymmetric = strel('line', 11, 90);

% Dilate A with B symmetric
AB_symmetricDilate = imdilate(A, B_symmetric);
% Dilate A with B unsymmetric
```

```
AB_unsymmetricDilate = imdilate(A, B_unsymmetric);

% Dilate the transpose of B with the transpose of A
B_symmetricADilate = imdilate(A', B_symmetric')';
% Dilate the transpose of B with the transpose of A
B_unsymmetricADilate = imdilate(A', B_unsymmetric')';

% Compare the results
isDilationCommutative = isequal(AB_symmetricDilate, B_symmetricADilate);
isDilationUnsymmetricCommutative = isequal(AB_unsymmetricDilate,
B_unsymmetricADilate);


% Erode A with B symmetric
AB_symmetricErode = imerode(A, B_symmetric);
% Dilate A with B unsymmetric
AB_unsymmetricErode = imerode(A, B_unsymmetric);

% Dilate the transpose of B with the transpose of A
B_symmetricAErode = imerode(A', B_symmetric')';
% Dilate the transpose of B with the transpose of A
B_unsymmetricAErode = imerode(A', B_unsymmetric')';

% Compare the results
isErodeCommutative = isequal(AB_symmetricErode, B_symmetricAErode);
isErodeUnsymmetricCommutative = isequal(AB_unsymmetricErode,
B_unsymmetricAErode);

subplot(4,3, 1), imshow(B_symmetricADilate), title("A⊕B where B is
symmetrical ");
subplot(4,3, 2), imshow(AB_symmetricDilate), title("B⊕A where B is
symmetrical");
subplot(4,3, 3), axis off, text(0.5,0.5, "A⊕B=B⊕A ? " +
string(isDilationCommutative), "HorizontalAlignment","center");


subplot(4,3, 4), imshow(B_unsymmetricADilate), title("A⊕B where B is
unsymmetrical ");
subplot(4,3, 5), imshow(AB_unsymmetricDilate), title("B⊕A where B is
unsymmetrical");
subplot(4,3, 6), axis off, text(0.5,0.5, "A⊕B=B⊕A ? " +
string(isDilationUnsymmetricCommutative), "HorizontalAlignment","center");

subplot(4,3, 7), imshow(B_symmetricAErode), title("A⊖B where B is
symmetrical ");
subplot(4,3, 8), imshow(AB_symmetricErode), title("B⊖A where B is
symmetrical");
subplot(4,3, 9), axis off, text(0.5,0.5, "A⊖B=B⊖A ? " +
string(isErodeCommutative), "HorizontalAlignment","center");

subplot(4,3, 10), imshow(B_unsymmetricAErode), title("A⊖B where B is
unsymmetrical ");
subplot(4,3, 11), imshow(AB_unsymmetricErode), title("B⊖A where B is
unsymmetrical");
```

```
subplot(4,3, 12), axis off, text(0.5,0.5, "A⊖B=B⊖A ? " +
string(isErodeUnsymmetricCommutative), "HorizontalAlignment","center");
```



There may be special cases where dilation and erosion are commutative. One such case is when the structuring element is a symmetric and self-reverse element, as shown above for both erosion and dilation. However, this only applies to certain types of images and structuring elements, and cannot be generalized to all cases. Therefore, erosion and dilation are not commutative since there exist cases where that proof is false

---

c) (5) Based on the results of Question (b), discuss why *dilation* must be operated as $D = M \ X$ and *erosion* as $E = M \ X$.

---------- Q1.c: Analysis (5) ----------

```
The basic idea behind dilation is to add pixels to the boundaries of objects
in an image, while the basic idea behind erosion is to remove pixels from
the boundaries of objects in an image.
```

In mathematical terms, dilation and erosion can be expressed as operations between a structuring element M and an input image X. The structuring element M defines the shape and size of the neighborhood around each pixel in the input image that is considered in the operation.

Dilation is defined as the union of the translated structuring elements over the image X, that is D = M ⊕ X. This operation produces an output image where the object boundaries are expanded or dilated in size by the structuring element. The dilation operation is typically used to fill in small gaps between objects or to connect nearby objects.

On the other hand, erosion is defined as the intersection of the translated structuring elements over the image X, that is E = M ⊖ X. This operation produces an output image where the object boundaries are eroded or shrunk in size by the structuring element. The erosion operation is typically used to remove small or thin objects from an image or to separate overlapping objects.

---

2. (15) It is recognized that multiple filters and spatial operations may be composed in certain patterns in order to enhance image restoration in practice. Formally, these patterns can be described by mathematical models of *Image Frame Algebra* (IFA) as presented in the course.

A useful image enhancing technology known as the *image sharpening pattern* as shown in the following figure:

Its algebraic model is: $$SharpIm = (Im - Iblur) + Im \quad (1)$$

In the above model, *Iblur* may be generated by the imfilter function, i.e.: `Inoi = imnoise(Im,'gaussian',0.01); Fblur = fspecial('average', 5); Iblur = imfilter(Inoi,Fblur).`

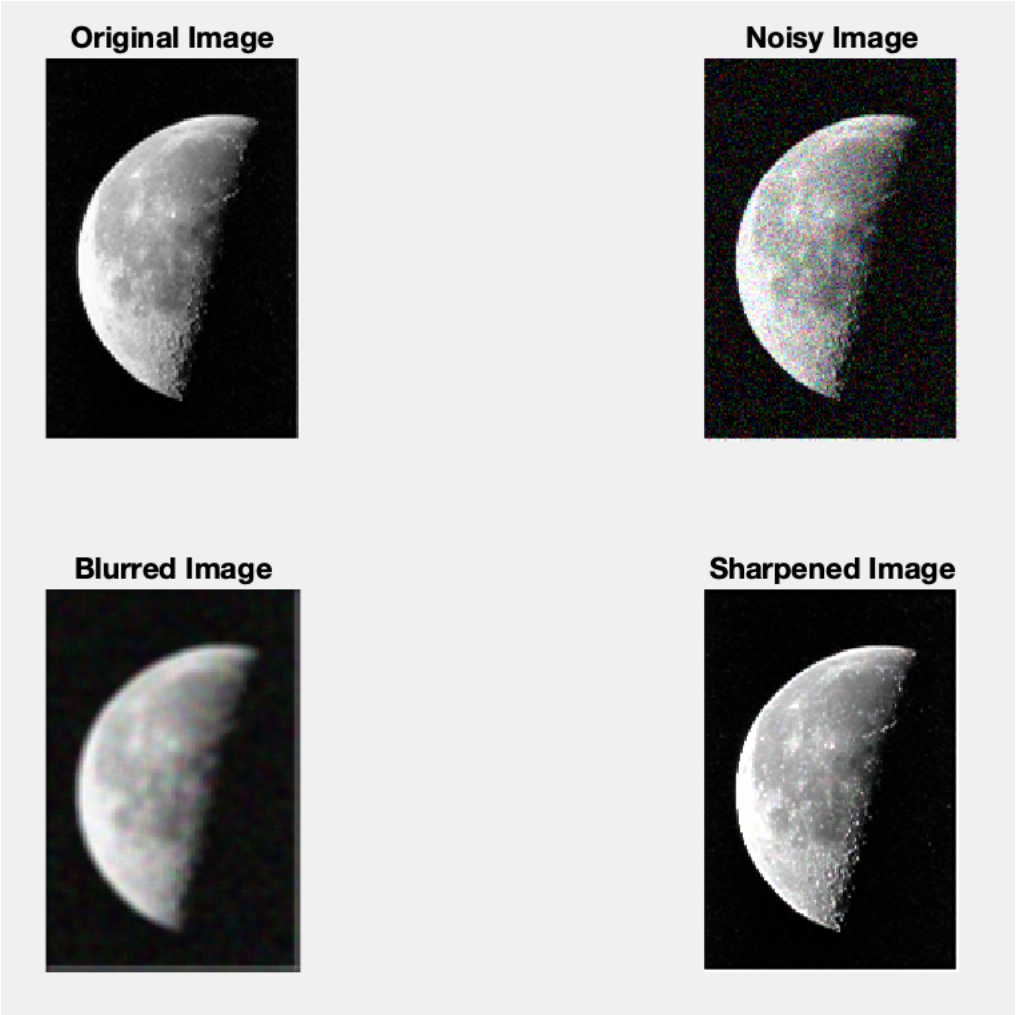
Given a sample image `'Moon.tif'` as provided in Lab3 Materials, implement the mathematical model of Ep. (1) in MATLAB and show your testing results.

---------- MATLAB code (10) ----------

```
% Read the input image
Im = imread('Moon.tif');

% Add Gaussian noise to the input image
Inoi = imnoise(Im, 'gaussian', 0.01);

% Apply average filter to the noisy image
Fblur = fspecial('average', 5);
Iblur = imfilter(Inoi, Fblur);

% Compute the sharpened image using the algebraic model
SharpIm = (Im - Iblur) + Im;

% Display the original, noisy, blurred, and sharpened images
```

```
subplot(2, 2, 1);
imshow(Im);
title('Original Image');

subplot(2, 2, 2);
imshow(Inoi);
title('Noisy Image');

subplot(2, 2, 3);
imshow(Iblur);
title('Blurred Image');

subplot(2, 2, 4);
imshow(SharpIm);
title('Sharpened Image');
```

---------- Plots of results (5) ----------



---

3. (20) Morphology provides a new perspective on image topology and geography. Test how the following morphological expressions are functioned in MATLAB for Image Contrast Enhamcement (*ImEnhance*) on the given image *Statue.png*.

a) (5) Prove based on the definitions of *top-hat* and *bottom-hat* as given above. Explain the physical meaning of the principle that you proved.

---------- Your proof and analysis (5) ----------

The top-hat transform enhances the bright regions in an image, while the bottom-hat transform enhances the dark regions. By adding the input image with its top-hat and bottom-hat transforms, we can combine the enhanced bright and dark regions with the original image to create an image with improved contrast and brightness.

---

b) (15) Question 3(a) has indicated there are two equivalent algorithms for implementing the morphological methodology for image contrast enhancement, i.e., either Try to implement both algorithms in MATLAB and show your testing results. The following built-in functions of MATLAB will be applied: `imtophat, imbothat, imopen, and imclose.`

---------- MATLAB code for Algorithm 1 (5) ----------

```
% Algorithm 1
Im = imread('Statue.png');

%create mask
se = strel('disk', 2);

%create top and bottom hat images
ImTophat = imtophat(Im, se);
ImBottomhat = imbothat(Im, se);

%enhance image
ImEnhance = Im + ImTophat - ImBottomhat;
```

---------- MATLAB code for Algorithm 2 (7) ----------

```
% Algorithm 2
closing = imclose(Im, se);
opening = imopen(Im, se);
temp = imsubtract(Im, opening);
enhanced2 = imadd(closing, temp);

% Display the input and enhanced images for both algorithms
figure;
subplot(1,3,1); imshow(Im); title('Input Image');
subplot(1,3,2); imshow(ImEnhance); title('Algorithm 1');
subplot(1,3,3); imshow(enhanced2); title('Algorithm 2');
```

---------- Plots of testing results (3) ----------

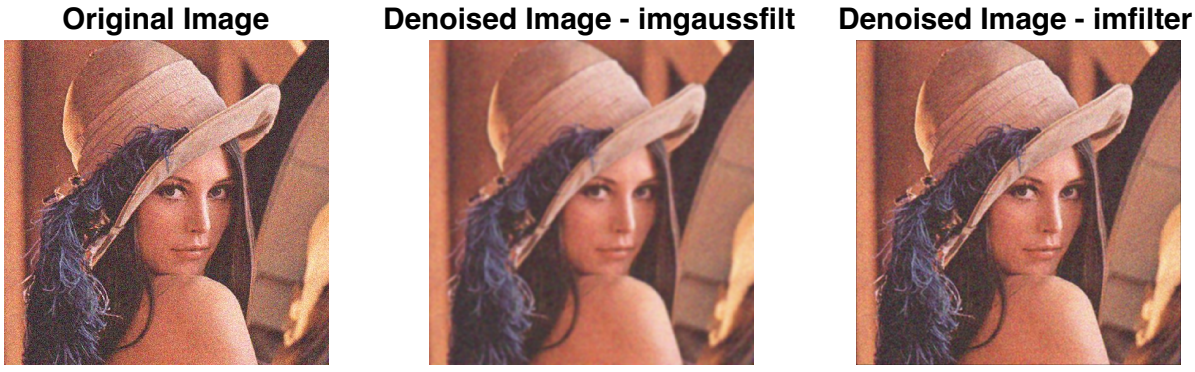
---

4. (10) Suppose a color image, In = LinnaGNoise.jpg, was affected by Gaussian noise in its R/G/B components as given in Lab3 materials. Try to recover (denoising) it by both 2D filtering technologies based on MATLAB functions `imgaussfilt(…)` and `imfilter(…)`. Plot and compare the original and filtered images.

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

```
% Read the noisy image
Im = imread('LennaGNoise.jpg');

% Denoise using imgaussfilt
sigma = 2;
Ig = imgaussfilt(Im, sigma);

% Denoise using imfilter
kernel = fspecial('gaussian', [3 3], sigma);
If = imfilter(Im, kernel);
figure;
% Display the original image
subplot(1,3,1); imshow(Im); title('Original Image');
subplot(1,3,2); imshow(Ig); title('Denoised Image - imgaussfilt');
subplot(1,3,3); imshow(If); title('Denoised Image - imfilter');
```

---------- Plots of results (3) ----------



---

5. (35) Design your own 2D barcode

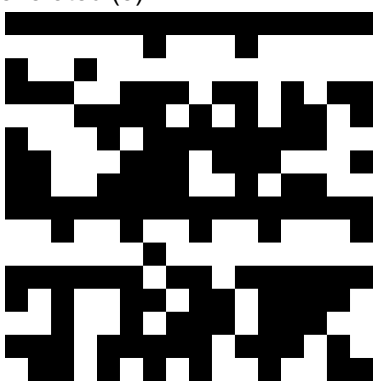
The 2D barcode is a popular technology of computer vision developed in 2000. The geographical and spatial operations of MATLAB can be applied to develop an arbitrary barcode system. In this given problem, design and implement a 16 x 16 2D barcode as follows

a) (20) In the barcode generation process, the 2D 16 x 16 *Barcode* will be structured as zeros (1, 16). Then an arbitrary *Str* = `'Hello World! This is my barcode.'` will be coded into *Barcode* using string conversion technologies: `StrB = reshape(dec2bin(Str, 8).'-'0',1,[])`. The final step will write *StrB* into the 16 x (8 + 8) segments of *Barcode* where each segment representing an ASCII letter as inputted in *Str*.

---------- MATLAB code for barcode generation (15) ----------

```
% Define the input string
Str = 'Hello World! This is my barcode.';

% Convert the string to ASCII code
ASCII = uint8(Str);

% Convert each ASCII code to an 8-bit binary representation
Bin = dec2bin(ASCII, 8);

% Reshape the binary representation into a 16x16 matrix of zeros and ones
Barcode = reshape(Bin.'-'0', 16, []);
```

---------- Plots of your barcode generated (5) ----------



b) (15) In the barcode recognition process, the 16 x (8 + 8) binary segments of *Barcode* will be converted from 16 x 2 integers (uint8) to characteristics (char) in iteration. Then, the assembled string will be the output of your recognized barcode `BarCode = char(String)`.

---------- MATLAB code for barcode recognition (10) ----------

```
Barcode_string = char(bin2dec(reshape(char(Barcode+'0'), 8,[]).'));
subplot(1,1,1), axis off, text(0.5,0.5, Barcode_string,
"HorizontalAlignment","center");
```

---------- Plots of your barcode and the recognized information (5) ----------