Assignment 6 (50)

Mathematical Morphology.

A MatLab reference: **https://www.mathworks.com/help/images/morphological-filtering.html**

A Python tutorial: **https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html**

Problem 1: Erosion and Dilation (20)

Choose a gray scale image (or convert a color image to grayscale). Convert to binary based on a mean-intensity threshold. Apply erosion twice and then dilation twice with a structuring element of your choice. (Defaults are fine.) Then, starting with the original binary image, dilate twice and erode twice. Show the two resulting images and contrast them.

In the initial sequence (Beginning with Erosion and Following with Dilation), the erosion operations primarily eliminate finer details and cause the binary image's objects to contract in size. Subsequently, the dilation operations work to expand the objects, but they may not completely restore the original shapes.

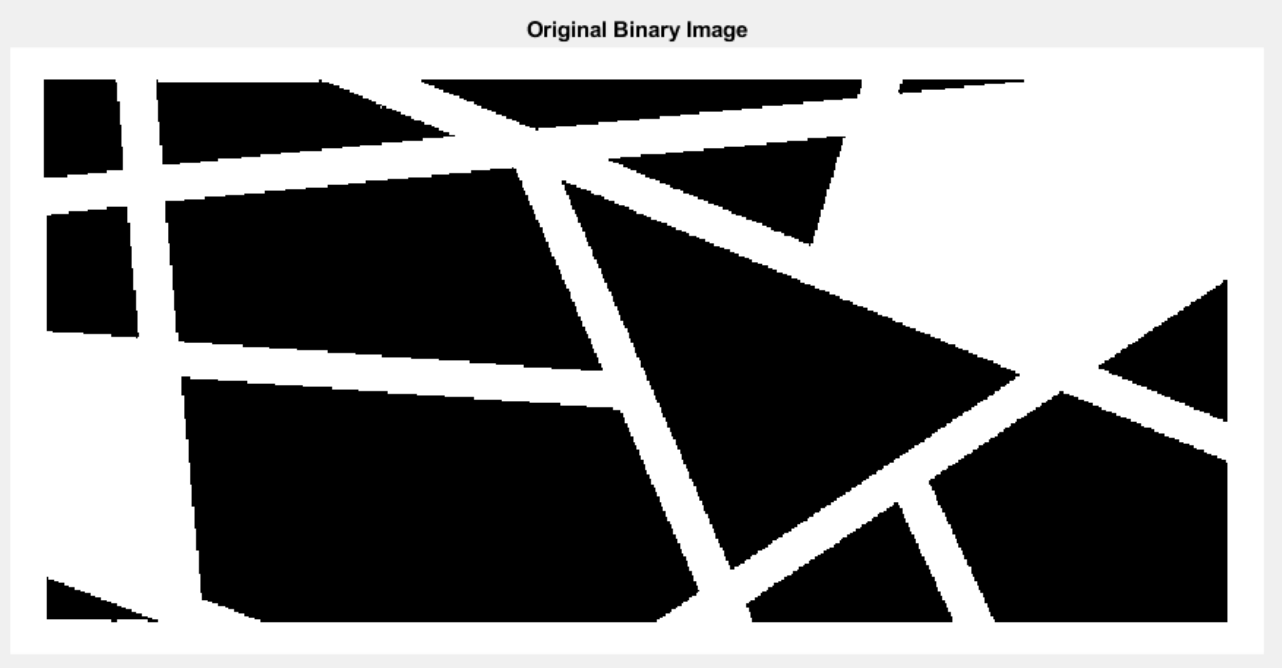
Conversely, in the second sequence (Commencing with Dilation and Subsequently Applying Erosion), the dilation operations are the initial step, resulting in the expansion of objects within the binary image. Following this, the subsequent erosion operations aim to eliminate portions of the expanded regions. This order of operations can yield results distinct from the first sequence, contingent on the specific characteristics present in the binary image.

CODE:

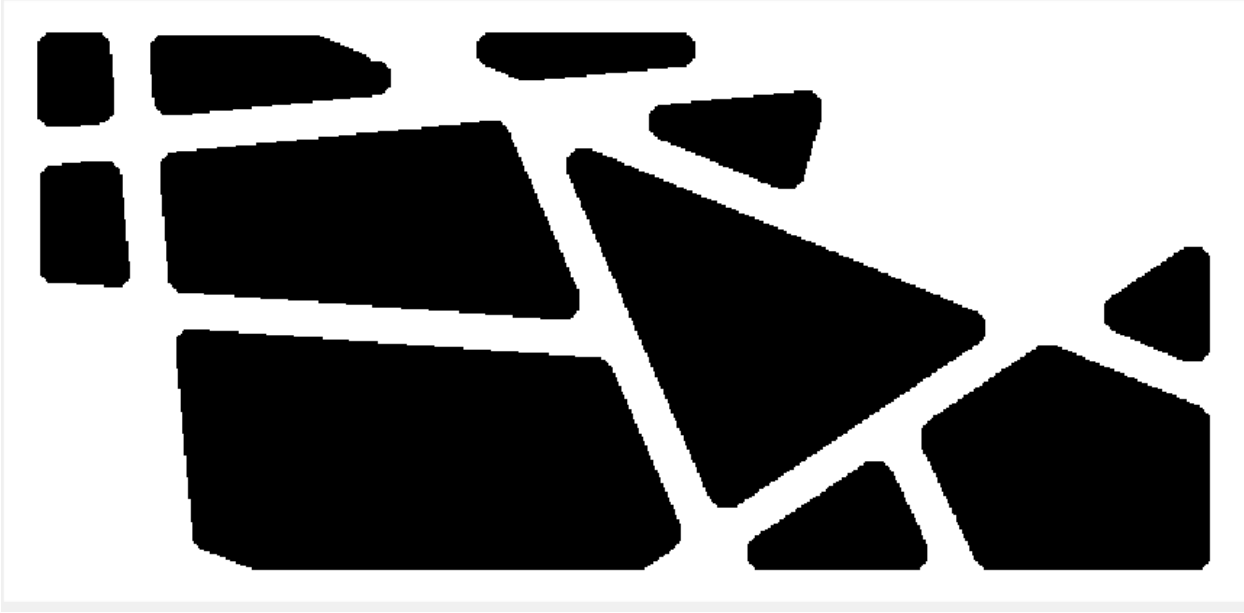
```
OG_Image = imread('shapes.jpg');
if size(OG_Image, 3) == 3
    G_Image = rgb2gray(OG_Image);
else
    G_Image = OG_Image;
end

% Converting to binary based on mean intensity threshold
mean_thresh = mean(G_Image(:));
BI_Image = G_Image > mean_thresh;

SE = strel('disk', 5);

Erode_Image = imerode(BI_Image, SE);
Erode_Image = imerode(Erode_Image, SE);
Dilate_Image = imdilate(Erode_Image, SE);
Dilate_Image = imdilate(Dilate_Image, SE);

figure;
subplot(1, 2, 1);
imshow(BI_Image);
title('Original Binary Image');
```

```
subplot(1, 2, 2);
imshow(Dilate_Image);
title('Dilated Twice After Erosion Twice');

Dilate_Image = imdilate(BI_Image, SE);
Dilate_Image = imdilate(Dilate_Image, SE);
Erode_Image = imerode(Dilate_Image, SE);
Erode_Image = imerode(Erode_Image, SE);

figure;
subplot(1, 2, 1);
imshow(BI_Image);
title('Original Binary Image');

subplot(1, 2, 2);
imshow(Erode_Image);
title('Eroded Twice After Dilation Twice');
```

O/P:



Eroded Twice After Dilation Twice



Dilated Twice After Erosion Twice

Problem 2: Opening and Closing (20)

Using the binary image from Problem 1, apply opening twice and then clsoing twice with a structuring element of your choice. (Defaults are fine.) Then, starting with the original binary image, close twice and open twice. Show the two resulting images and contrast them. Then, compare how your two images from Problem 2 visually differ from the two images of Problem 1.

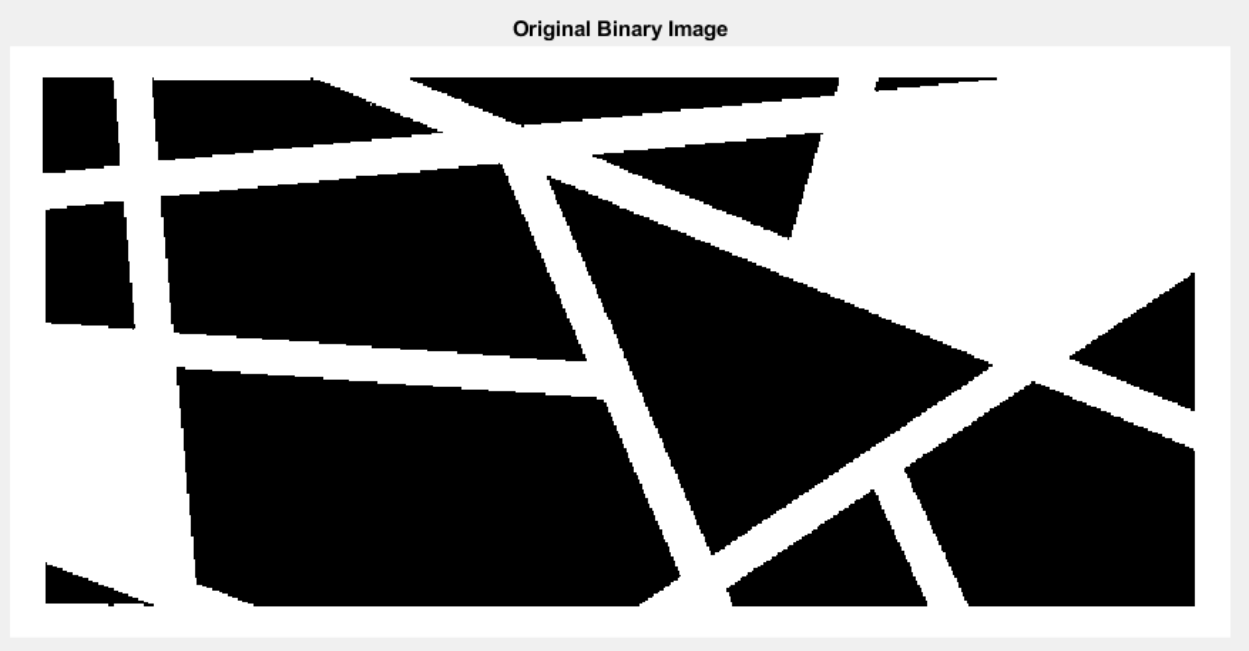
In the first sequence, opening and closing enhance object connectivity and reduce noise, while erosion and dilation modify object size and shape differently.

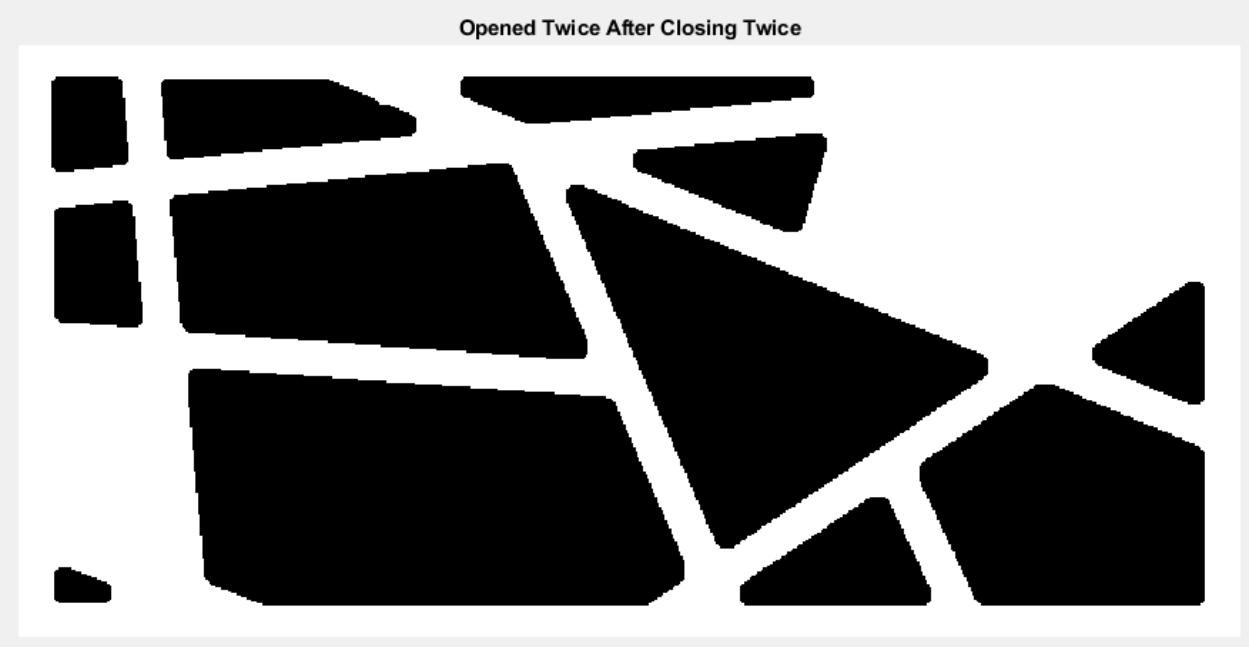
In the second sequence, closing and opening operations change object connectivity, while dilation and erosion affect object size, with the order of operations playing a key role.

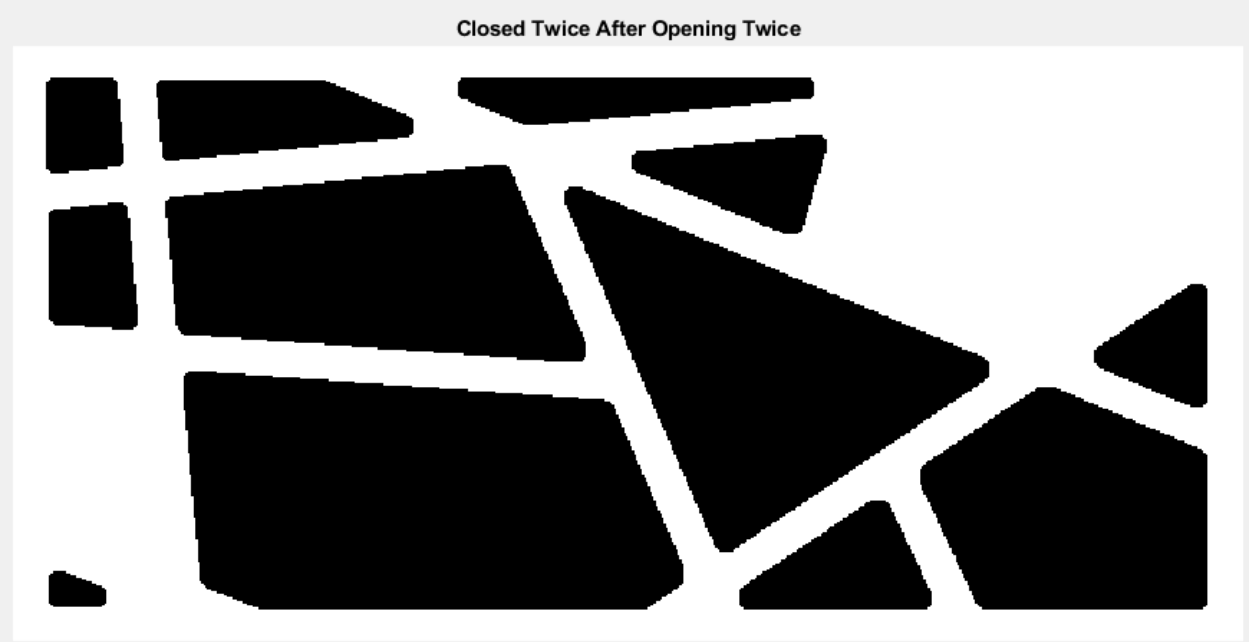
CODE:

```
SE = strel('disk', 5);

Opened_Image = imopen(BI_Image, SE);
Opened_Image = imopen(Opened_Image, SE);
Closed_Image = imclose(Opened_Image, SE);
Closed_Image = imclose(Closed_Image, SE);

figure;
subplot(1, 2, 1);
imshow(BI_Image);
title('Original Binary Image');

subplot(1, 2, 2);
imshow(Closed_Image);
title('Closed Twice After Opening Twice');

Closed_Image = imclose(BI_Image, SE);
Closed_Image = imclose(Closed_Image, SE);
Opened_Image = imopen(Closed_Image, SE);
Opened_Image = imopen(Opened_Image, SE);

figure;
subplot(1, 2, 1);
imshow(BI_Image);
title('Original Binary Image');

subplot(1, 2, 2);
imshow(Opened_Image);
title('Opened Twice After Closing Twice');
```

O/P:







Problem 3: Boundary Extraction (10)

Boundary extraction of an image I, β(I), can be performed by eroding I with a structuring element B and then subtracting the eroded image from I:

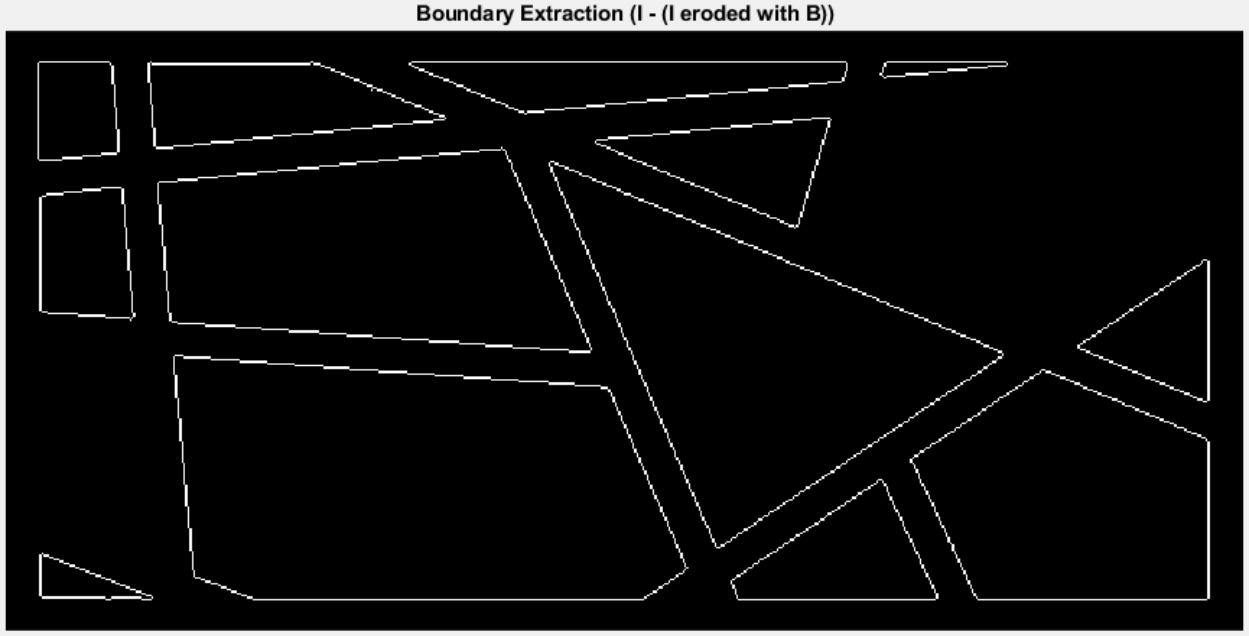
$\beta(I) = I - (I \ominus B)$

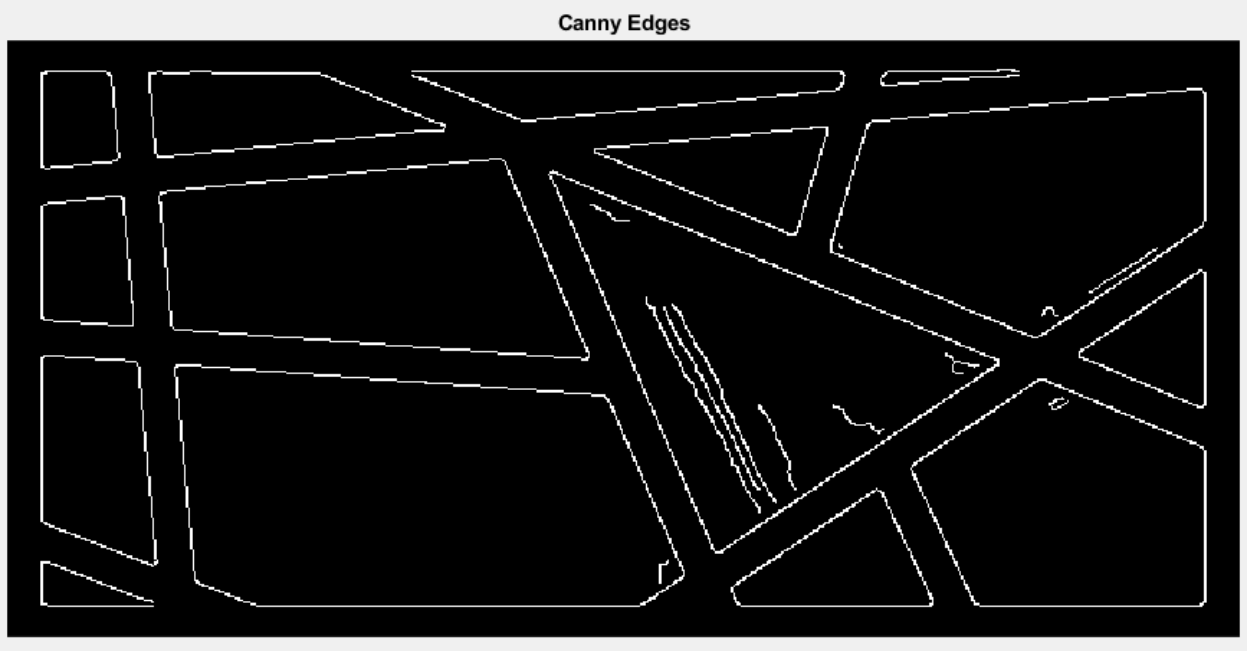
Using your binary image from the two problems above, extract the boundaries. Then, extract Canny edges from the original grayscale image. Show the two images and compare them.

CODE:

```
% Define the structuring element for boundary extraction
SE_boundary = strel('disk', 1);

Boundary_Image = BI_Image - imerode(BI_Image, SE_boundary);
Canny_Edges = edge(G_Image, 'Canny');

figure;
subplot(1, 2, 1);
imshow(Boundary_Image);
title('Boundary Extraction (I - (I eroded with B))');

subplot(1, 2, 2);
imshow(Canny_Edges);
title('Canny Edges');
```

O/P:





General submission instructions:

(a) Be kind to your aging, over-worked professor and submit only a single document. This can be pdf, MS Word, OpenOffice, etc. Do not submit a zip file.

(b) Your single document should include the input image for your problem, if required, and answers to each of the sub-problems (text, image or both, as appropriate). Your document should also include code that you wrote to generate your answers.

(c) You may use any images you like for the programming; I encourage you to use images that might be useful/interesting for your final project.

(d) Feel free to use whatever functions MatLab supplies, except where otherwise specified. Also feel free to write your own, if you are so inclined; it will take more time, but you will gain a deeper understanding of the material. It is one thing, for example, to implement Otsu thresholding using `otsuthresh`, quite another to write an thresholding technique yourself.

(e) Point values for each question are indicated as ***x/y*** in which ***x*** is the point value for 481 students and ***y*** is the point value for 381 students.