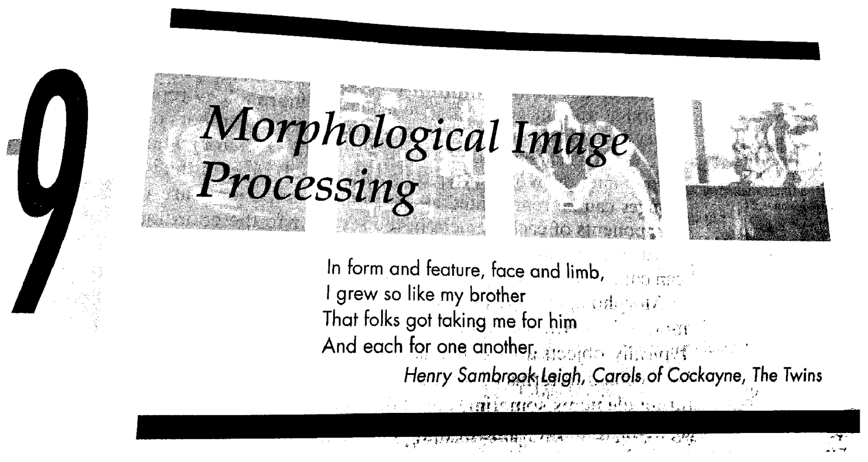
ECE 253, Homework 1

Due: Friday October 7, 2022 by 11:59pm

The first three problems should be done using Matlab, and the last two should be done by hand. Submit your homework electronically in Canvas on Gradescope. Upload one PDF file (part-1) with your answers for problems 1 & 2, and a second PDF file (part-2) with your write-up for problems 3, 4, & 5. These should include your answers to each question and your Matlab code (cut and paste it in). Make sure the two report pieces both mention your full name and PID.

1) **Cleaning by cluster removal**
   A photo of a partial textbook page has been thresholded to be a binary image, called `badIm`, which looks like this:

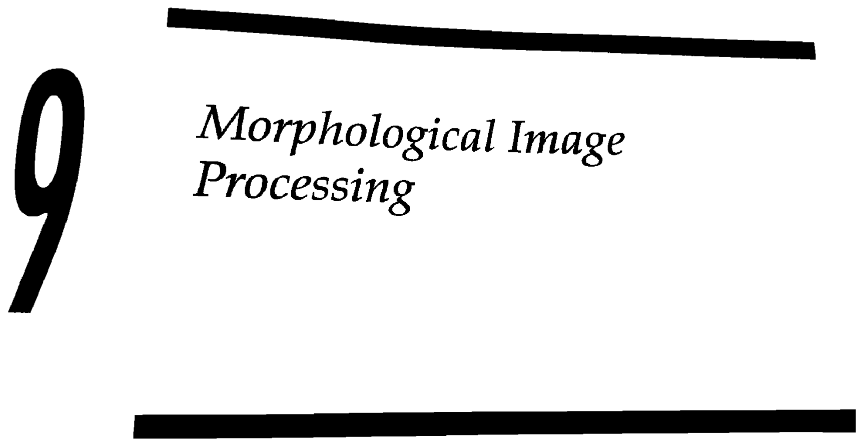


9

Morphological Image Processing

In form and feature, face and limb,
I grew so like my brother
That folks got taking me for him
And each for one another.
Henry Sambrook Leigh, Carols of Cockayne, The Twins

We want to extract the chapter number and chapter title and the large black lines, but get rid of everything else. So the clean image we are aiming for, called `idealIm`, looks like this:



9

Morphological Image Processing

We can use `xor` to find all the pixels where `idealIm` and `badIm` are not the same. If we sum up over the rows and columns of the xor output, we get a count of how many pixels are not the same:

```
count = sum(sum(xor(idealIm,badIm)))
```

and we find there are 127,994 pixels different. We will try to make this count as small as possible.

In Matlab, the function `bwareaopen` removes small objects (objects with fewer than P pixels) from an image. Because there are a lot of clusters of black pixels (zero-valued pixels) that need to be removed, and because `bwareaopen` considers objects to be one-valued pixels, we will need to invert the image to operate on it, and then invert again to return the image to its normal appearance. So, for example, to remove clusters of black pixels with less than 10 pixels, we could use:

```
clean10 = 1 - bwareaopen(1-badIm,10);
```

We can then use `xor` to find and count all the pixels where `clean10` is different from `idealIm`. We will call these bad pixels.

a) Make a plot where the x-axis (going from 1 to 2500) represents the number of pixels P for which black objects with fewer than P pixels get removed, and the y-axis is the corresponding number of bad pixels (that is, the number of pixels where the resulting `bwareaopen` result (with parameter P) differs from `idealIm`). Include the plot in your write-up and the matlab code which you used to generate it.

b) Explain why the plot has the shape that it does. In particular, there is an initial portion where the bad pixel count decreases monotonically. What is happening there? Then the curve bottoms out and goes generally up, but is not monotonic– it has some little ups and downs within a general upward trend. Why do those occur?

c) What is the minimum value of your curve, and what is the value of x at which that minimum occurs?

2) **Erosion and Dilation**

Now we will see how dilation and erosion can do a much better job of this task than the simple cluster removal.

a) Use `imdilate` and `imerode` commands on `badIm`, using the structuring element `ones(3)`. How does this do in terms of the count of bad pixels? Try ones(5) and ones(7). Show your Matlab code and give your numeric results (count of bad pixels) and explain what is going on in your results.

b) If e5 and e7 denote the output from the previous part with structuring elements ones(5) and ones(7), it appears visually that e7, compared to e5, gets rid of more noise, but also gets rid of more of the letters that we want to keep. So we can potentially deploy e7 and e5 with an approach like “geodesic dilation” and “morphological reconstruction” from Section 9.6, to try to get back the portions of letters that were eaten away, without getting back the noise. Make a plot of the number of bad pixel versus the iteration number in this approach, using the structuring element ones(3). Show your code, and discuss the result.

c) Your choice: Try something else using dilation and erosion, potentially in combination with other operations or different structuring elements, that gets you even closer to `idealIm`. Whatever you try, you have to apply it to the whole image– you can’t just selectively apply your operator to a section of the image where there is noise.

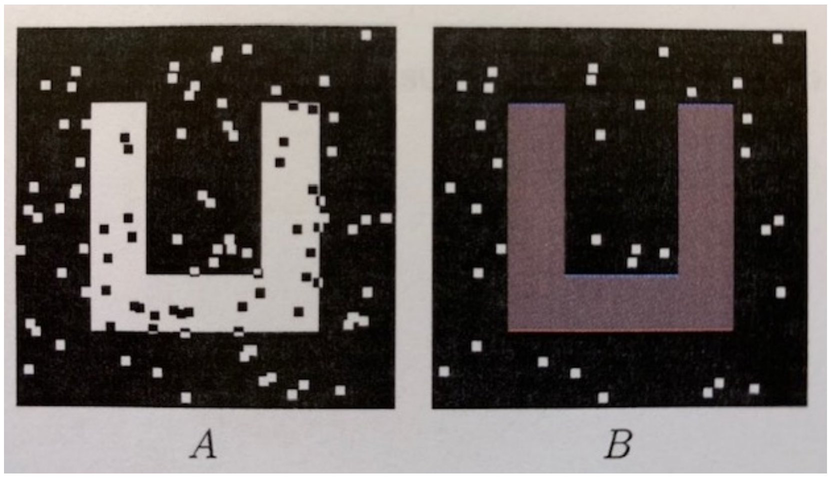
3) **Skeletonization**

a) Starting with the image `idealIm`, use the `bwmorph` function with `'skel'` to fully skeletonize the image. (You’ll need to invert the image first, or you’ll get weird results.) Explain the appearance of your results

b) Propose a sequence of steps (you do not have to carry them out) to get rid of the spurs. Your proposed method should be able to handle the fact that the skeleton components (and their spurs) are of very different sizes.

4) **Pencil and paper problem: Erosion and Dilation**

Image B below was produced from A by an operator that used two structuring elements: S_1, which exactly matches one of the small square boxes, and S_2 which fits snugly around the small square boxes. For example, if the boxes are 3x3 pixels, then S_1 would be the 3x3 structuring element `ones(3)`, and S_2 would be a 5x5 ring, where the outer ring pixels have value 1 and the inner 3x3 pixels have value 0. For both S_1 and S_2, the origin of the structuring element is at the center. The gray object in B is shown only for reference, and is actually not present.



Determine which of the operations listed below was used in this transformation. Give a brief explanation.

a) $B = [(A \ominus S_1) \cap (A^c \ominus S_2)] \ominus S_1$

b) $B = [(A \ominus S_1) \cap (A^c \ominus S_2)] \oplus S_1$

c) $B = [(A \oplus S_1) \cap (A^c \ominus S_2)] \cap A$

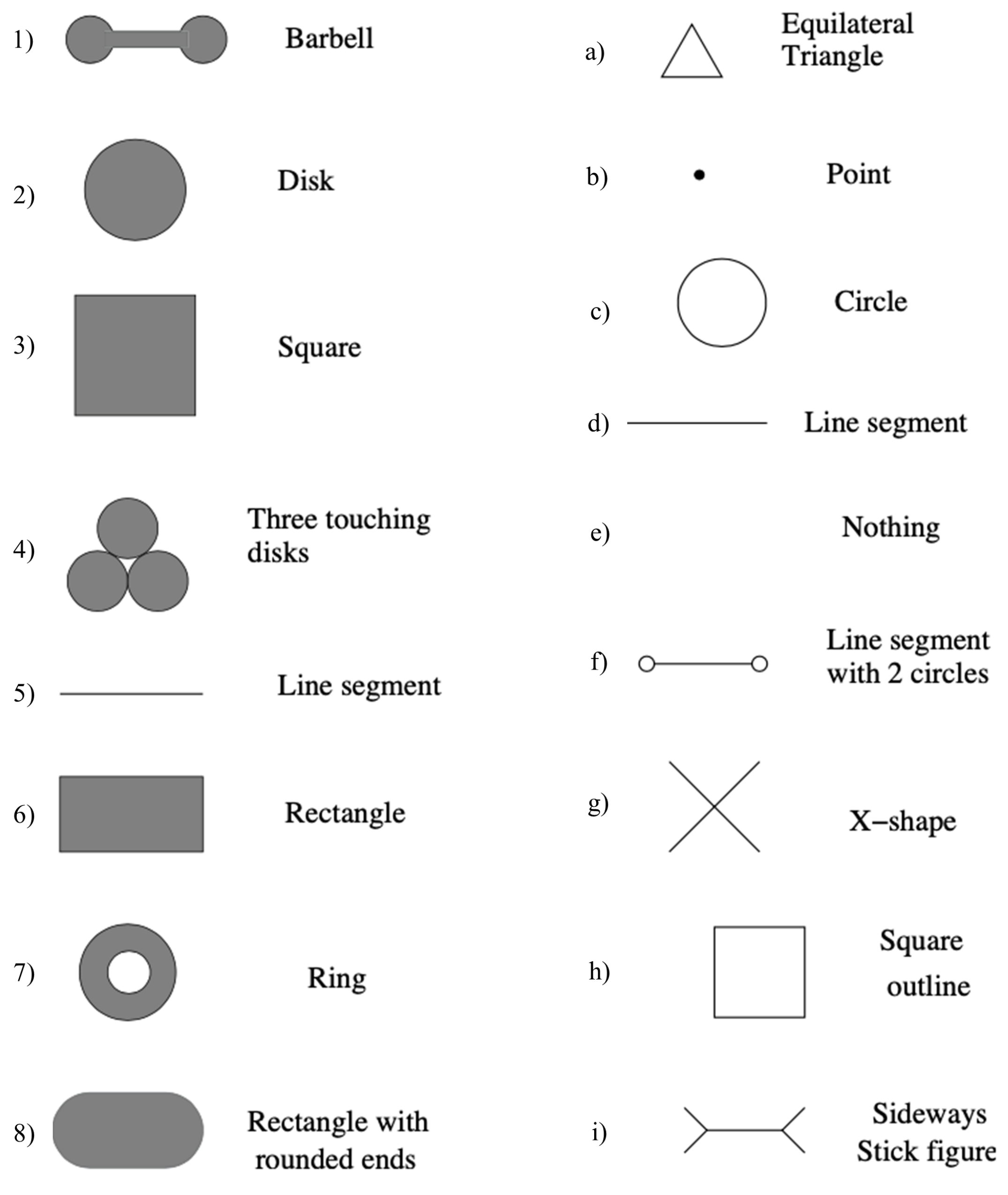
d) $B = [(A \ominus S_1) \cap (A^c \ominus S_2)] \cap A$

5) **Pencil and paper problem: Skeletons**

Match the sets in R^2 in the left column with their "skeleton by maximal balls" in the right column. In other words, for each numbered shape on the left, find the letter of the skeleton it corresponds to on the right. If you find that an item on the left has no match on the right, then draw what the skeleton would look like. No explanations are required.

Note #1: The shapes and their skeletons are not necessarily drawn to the same scale, so don't rule things out on scale grounds. For example, the line segment in the right column is longer than the diameter of the disk in the left column; that alone would be enough to rule out a match if these items were drawn to scale. But you should ignore the scale.

Note #2: Consider these sets are in R^2, and not on a pixel grid in Z^2.


1) Barbell
a) Equilateral Triangle
2) Disk
b) Point
c) Circle
3) Square
d) Line segment
4) Three touching disks
e) Nothing
f) Line segment with 2 circles
5) Line segment
6) Rectangle
g) X−shape
7) Ring
h) Square outline
8) Rectangle with rounded ends
i) Sideways Stick figure