Lecture 5: Binary Morphology

Last modified on January 15, 2000 at 3:00 PM

Contents

Reading

SH&B, 11.1–11.3
Castleman, 18.7.1–18.7.4

5.1 What is Mathematical Morphology?

"Morphology" can literally be taken to mean "doing things to shapes". "Mathematical morphology" then, by extension, means using mathematical principals to do things to shapes.

5.2 Image Regions as Sets

The basis of mathematical morphology is the description of image regions as sets. For a binary image, we can consider the "on" (1) pixels to all comprise a set of values from the "universe" of pixels in the image. Throughout our discussion of mathematical morphology (or just "morphology"), when we refer to an image A, we mean the set of "on" (1) pixels in that image.

The "off" (0) pixels are thus the set compliment of the set of on pixels. By A^c, we mean the compliment of A, or the off (0) pixels. We can now use standard set notation to describe image operations:

A	the image
A^c	the compliment of the image (inverse)
$A \cup B$	the union of images A and B
$A \cap B$	the intersection of images A and B
$A - B = A \cap B^c$	the difference between A and B (the pixels in A that aren't in B)
$\#A$	the cardinality of A (area of the object(s))

We'll also need to introduce the notion of a translated image set. The image A translated by movement vector t is

$$A_t = \{c \mid c = a + t \text{ for some } a \in A\} \tag{5.1}$$

Now, we have the necessary notation to introduce most of the rest of mathematical morphology. The first two basic operations are dilation and erosion.

5.3 Basic Binary Operations

5.3.1 Dilation

Dilation (sometimes called "Minkowsky addition") is defined as follows.

$$A \oplus B = \{c \mid c = a + b \text{ for some } a \in A \text{ and } b \in B\} \tag{5.2}$$

One way to think of this is to take copies of A and translate them by movement vectors defined by each of the pixels in B. (This means we have to define an origin for B.) If we union these copies together, we get $A \oplus B$. This can be written as

$$A \oplus B = \bigcup_{t \in B} A_t \tag{5.3}$$

Alternatively, we can take copies of B and translate them by movement vectors defined by each of the pixels in A—dilation is commutative. An interpretation of this latter way of thinking about it is to put a copy of B at each pixel in A. If we "stamp" a copy of B at each pixel in A and union all of the copies, we get $A \oplus B$. This can be written this way:

$$A \oplus B = \bigcup_{t \in A} B_t \tag{5.4}$$

In this way, dilation works much like convolution: slide a kernel to each position in the image and at each position "apply" (union) the kernel. In the case of morphology, we call this kernel B the *structuring element*.

Dilation is also associative as well as commutative:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)) \tag{5.5}$$

This follows naturally from the way we are unioning multiple translated sets together.

This associativity is useful for decomposing a single large dilation into multiple smaller ones. Suppose, for example, that you have available a board that does 3×3 dilation in hardware, but you want to perform dilation by a 7×9 structuring element. Does there exist a set of 3×3 element that when dilated by each other in succession gives the 7×9 one? If so, you can write the larger dilation as the result of successively dilating by the smaller elements.

5.3.2 Erosion

Erosion (sometimes called "Minkowsky subtraction") is defined as follows.

$$A \ominus B = \{x \mid x + b \in A \text{ for every } b \in B\} \tag{5.6}$$

One way to think of this is to take copies of A and again translate them by movement vectors defined by each of the pixels in B. However, this time we move them in the opposite direction ($-b$) and *intersect* the copies together. This can be written as

$$A \ominus B = \bigcap_{t \in B} A_{-t} \tag{5.7}$$

An interpretation of this latter way of thinking about it is to again put a copy of B at each pixel in A. If count only those copies whose translated structuring elements lie entirely in A and mark the pixels in A that these copies were translated to, we get $A \ominus B$.

Unlike dilation, erosion is not commutative. (Much like how addition is commutative while subtraction is not.)

Also unlike dilation, erosion is not associative:

$$(A \ominus B) \ominus C \stackrel{?}{=} A \ominus (B \ominus C) \tag{5.8}$$

5.3.3 Duality of Dilation and Erosion

When one operation is the *dual* of the other, it means that one can be written in terms of the other.

This does *not*, however, mean that they are opposites. Unlike arithmetic addition and subtraction,

$$(A \oplus B) \ominus B \stackrel{?}{=} A \tag{5.9}$$

Dilation and erosion are related as follows. If $\breve{B}$ denotes the reflection of B,

$$\begin{array}{rcl} (A \oplus B)^c & = & A^c \ominus \breve{B} \\ (A \ominus B)^c & = & A^c \oplus \breve{B} \end{array} \tag{5.10}$$

In other words, dilating the "foreground" is the same as eroding the "background", but the structuring element reflects between the two. Likewise, eroding the foreground is the same as dilating the background.

So, strictly speaking we don't really need *both* dilate and erode: with one or the other, and with set complement and reflection of the structuring element, we can achieve the same functionality. Hence, dilation and erosion are duals.

This duality can also be used to derive an equivalent to associativity for erosion:

$$\begin{array}{rcl} (A \ominus B) \ominus C & = & ((A \ominus B)^c \oplus \breve{C})^c \\ & = & ((A^c \oplus \breve{B}) \oplus \breve{C})^c \\ & = & (A^c \oplus (\breve{B} \oplus \breve{C}))^c \\ & = & (A \ominus (\breve{B} \breve{\oplus} \breve{C}) \\ & = & (A \ominus (B \oplus C) \end{array} \tag{5.11}$$

Thus, we can combine the effects of eroding by first B then C into a single erosion by the *dilation* of B by C. Again making the analogy to normal arithmetic, this really isn't all that different from how $(A - B) - C = A - (B + C)$.

5.4 Some Examples of Using Dilation and Erosion

Suppose that you wanted to find all pixels lying on the boundary of an object. We could perform the following operations:

$$\text{Bound}_{\text{ext}}(A) = (A \oplus B) - A \tag{5.12}$$

where B was a 3×3 structuring element containing all 1s. This would give us all background pixels that bordered the object.

Or, if we wanted all foreground pixels that bordered the background, we could use

$$\text{Bound}_{\text{int}}(A) = A - (A \ominus B) \tag{5.13}$$

You could also extend this to give you minimum (8-connected) distance from an object:

$$\text{Dist}_i = (A \oplus_i B) - (A \oplus_{i-1} B) \tag{5.14}$$

where $\oplus_i$ denotes i applications of the dilation operator.

Do you see how you can create 4-connected equivalents of these?

5.5 Proving Properties of Mathematical Morphology

To construct proofs using mathematical morphology, you employ algebraic proofs using the properties shown here and other properties from set algebra. The example in Eq. 5.11 is one such type of proof. In some of your homework problems, you will use similar techniques to prove other useful properties.

5.6 Hit-and-Miss

When eroding, the "0"s in the structuring element act like "don't care" conditions—they don't really require that the image be on or off at that point, only that the remaining "1" pixels fit inside the object. In other words, it finds places that "look like this" (for the 1s), but has no way to say "but doesn't look like this" (for the 0s).

We can can combine erosion and dilation to produce an operator that acts like this: the "hit and miss" operator. The operator takes two elements: one that must "hit" and one that must "miss".

The operator is defined as follows. If the structuring elements J (hit) and K (miss) are applied to the image A:

$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K) \tag{5.15}$$

In other words, the structuring element J must fit inside the object and the element K must fit *outside* the object at that position.

This gives us a form of binary template matching. For example, the following structuring elements give an "upper right" corner detector:

0	0	0
1	1	0
0	1	0

J

0	1	1
0	0	1
0	0	0

K

The J element finds the points with connected left and lower neighbors, and the H element finds the points *without* upper, upper right, and right neighbors.

In some cases, we'll simply use a single structuring element B, with the assumption that $J = B$ and $K = B^c$. I.e.,

$$A \otimes B = A \otimes (B, B^c) = (A \ominus B) \cap (A^c \ominus B^c)$$

Notice that this doesn't, however, allow for a third case: "don't care" pixels—ones that could be either inside or outside the shape. Some authors will for this reason write the two operators as a single one using 1s for the hits, 0s for the misses, and "x"s for the don't-care positions. In the previous example, this would be written as

x	0	0
1	1	0
x	1	x

This form is perhaps more useful for visualizing what the structuring element is designed to find. Remember, though, that you still have to apply two different operators, one for hit (the 1s) and one for miss (the 0s), when implementing hit-and-miss.

5.7 Opening and Closing

5.7.1 Opening

An *opening* is an erosion followed by a dilation with the same structuring element:

$$A \circ B = (A \ominus B) \oplus B \tag{5.16}$$

Remember that erosion finds all the places where the structuring element fits inside the image, but it only marks these positions at the origin of the element. By following an erosion by a dilation, we "fill back in" the full structuring element at places where the element fits inside the object. So, an opening can be consider to be the union of all translated copies of the structuring element that can fit inside the object.

Openings can be used to remove small objects, protrusions from objects, and connections between objects. So, for example, opening by a 5×5 square element will remove *all* things that are less than 5 pixels high or less than 5 pixels tall.

5.7.2 Closing

Closing works in an opposite fashion from opening:

$$A \bullet B = (A \oplus B) \ominus B \tag{5.17}$$

Whereas opening removes all pixels where the structuring element won't fit inside the image foreground, closing *fills in* all places where the structuring element will not fit in the image *background*.

Remember, though, duality doesn't imply inverse operations: an opening following a closing will *not* restore the original image.

5.7.3 Properties of Opening and Closing

Duality

Closing is the dual of opening:

$$(A \circ B)^c = A^c \bullet \check{B} \tag{5.18}$$

Just as erosion can be implemented using dilation (and vice versa), opening can be implemented using closing (and vice versa)

Idempotence

An important property of opening and closing operations is that they are *idempotent*: if you apply one more than once, nothing changes after the first application. So,

$$A \circ B \circ B = A \circ B \tag{5.19}$$

and

$$A \bullet B \bullet B = A \bullet B \tag{5.20}$$

5.7.4 Applications of Opening and Closing

Opening and closing are the basic workhorses of morphological noise removal. Opening removes small objects, and closing removes small holes.

Finding Things

You can also use opening and closing to find specific shapes in an image. While these may not be "noise", they may just be other things that you don't care about. (In this sense, perhaps noise is just things you don't care about?) Opening can be used to only find things into which a specific structuring element can fit.

Subdivision into Parts

You can also extend this idea to part subdivision. Suppose that you know that a particular object has specific parts with specific shapes. Openings can be used to separate those parts by shaping the structuring elements to "fit" into those parts. They don't have to exactly match the parts—it's sufficient to only fit into some parts while not fitting into others.

Vocabulary

- Mathematical Morphology
- Dilation
- Erosion

- Dual
- Hit-and-Miss
- Opening
- Closing
- Idempotence