APPM4058A & COMS7238A: Digital Image Processing

Hairong Wang

School of Computer Science & Applied Mathematics
University of the Witwatersrand, Johannesburg

2019-3-13

Contents

Outline

1. Mathematical Morphology Applications
   - Boundary extraction
   - Noise removal
   - Hit-or-miss transform
   - Region filling
   - Connected components
   - Thinning

Outline

1. Mathematical Morphology Applications
   - Boundary extraction
   - Noise removal
   - Hit-or-miss transform
   - Region filling
   - Connected components
   - Thinning

Boundary extraction

A is an image, B is a small structuring element consisting of points symmetrically placed about the origin, then boundary of A can be one of the following:

- $A - (A \ominus B)$ - 'internal boundary' which consists of those pixels in A at its edge;
- $(A \oplus B) - A$ - 'external boundary' which consists of those pixels not in A just next to its edge;
- $(A \oplus B) - (A \ominus B)$ - morphological gradient consists of both internal and external boundaries.

Outline

Noise removal

Suppose A is a binary image corrupted by impulse noise - salt and pepper noise. That is, some of the white pixels are black, and some of the black pixels are white.

- $A \ominus B$ will remove the single black pixels, but will enlarge the holes. We can fill the holes by dilating twice:

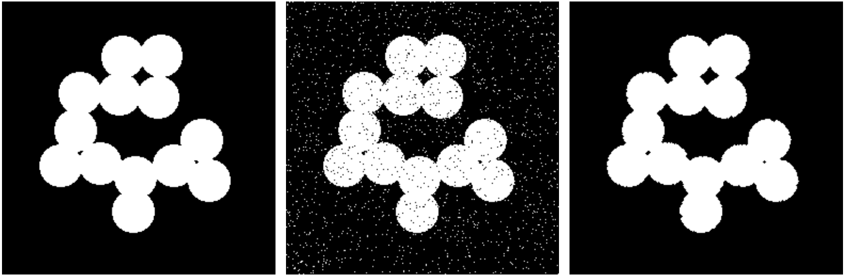
$$((A \ominus B) \oplus B) \oplus B. \tag{1}$$

- The first dilation returns the holes to their original size; the second dilation removes them. But this will enlarge the objects in the image.
- To reduce them to their correct size, perform a final erosion:

$$(((A \ominus B) \oplus B) \oplus B) \ominus B = (A \circ B) \bullet B. \tag{2}$$

WITS UNIVERSITY

Noise removal example



(a) (b) (c)

Figure: Noise removal. (a) Original image, (b) Added impulse noise, (c) Noise removal using a cross SE.

Outline

1. Mathematical Morphology Applications
   - Boundary extraction
   - Noise removal
   - Hit-or-miss transform
   - Region filling
   - Connected components
   - Thinning

Hit-or-miss transform

This is a method for finding shapes in images. Suppose we wish to find 3×3 square shapes

		•	•		•	•	•		•	•	•			
		•	•		•	•	•		•	•	•	•		
		•	•		•	•	•		•	•	•			

Table: Image A

Hit-or-miss example

						●				●				

Table: After $A \ominus B$

Hit-or-miss example

●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
●	●			●				●				●	●	●
●	●			●				●					●	●
●	●			●				●				●	●	●
●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Table: A^c

●	●	●	●	●
●				●
●				●
●				●
●	●	●	●	●

Table: Structuring element C

Hit-or-miss example

		•				•								

Table: After $A^c \ominus C$

						•				•				

Table: After $A \ominus B$

The intersection of $A \ominus B$ and $A^c \ominus C$ produces only one pixel.

Hit-or-miss example

						•								

Table: After $(A \ominus B) \cap (A^c \ominus C)$

- This combination of erosions form the hit-or-miss transform. In general,
  - Design two structuring elements, one (B_1) being the same shape as the shape we are looking for, and the other (B_2) being able to fit around the shape. The hit-or-miss transform is

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2), \tag{3}$$

where $B = (B_1, B_2)$.

Hit or miss example

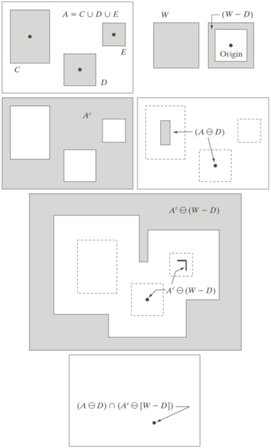
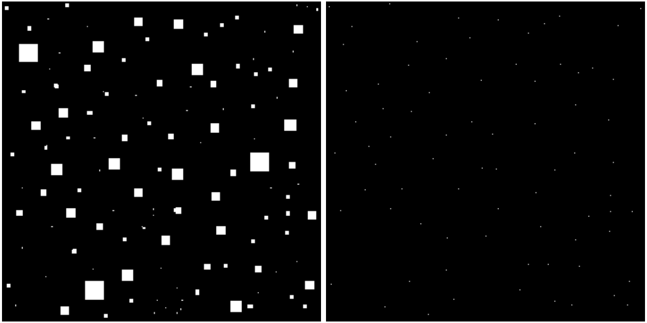


Figure: Hit or miss transform example

Hit-or-miss example



(a) (b)

Figure: (a) Original image; (b) hit-or-miss transform.

Hit-or-miss example cont.

The above example in Matlab

```
B1=strel([0 0 0; 0 1 1; 0 1 0]);
B2=strel([1 1 1; 1 0 0; 1 0 0]);
C=bwhitmiss(im,B1,B2);
```

in Python

```
B1 = np.array([[0, 0, 0],[0, 1, 1], [0, 1, 0]])
B2 = np.array([[1, 1, 1], [1, 0, 0], [1, 0, 0]])
C = binary_hit_or_miss(im,structure1=B1,structure2=B2)
```

in `scipy.ndimage`

WITS UNIVERSITY

Outline

1. Mathematical Morphology Applications
   - Boundary extraction
   - Noise removal
   - Hit-or-miss transform
   - Region filling
   - Connected components
   - Thinning

Region filling

Suppose in a binary image, we have a region bounded by 8-connected boundary. Given a pixel, p, within the region, we want to fill up the entire region with 1's.

- Starting with p, we dilate as many times as necessary using a cross shaped structuring element.
- Each time taking an intersection with A^c before continuing.
- Thus, a sequence of sets, $X_0, X_1, X_2, \ldots, X_k$ is created. For which

$$X_n = (X_{n-1} \oplus B) \cap A^c \tag{4}$$

- The process stops when there is no change between X_k and X_{k-1}. Finally, $X_k \cup A$ is the filled region.

Region filling

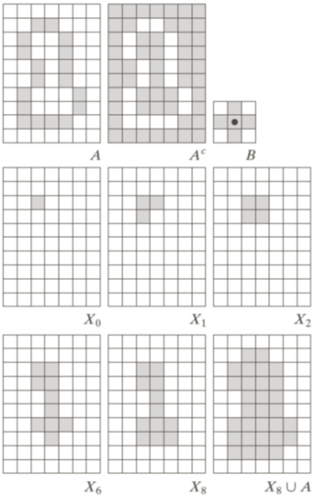


Figure: Region filling example

Region filling example

		●	●			
	●			●		
	●			●		
		●		●		
		●		●		
	●				●	
	●				●	
	●	●	●	●		

(a)

●	●	●	●	●	●	●
●	●			●	●	●
●		●	●		●	●
●		●	●		●	●
●	●		●		●	●
●	●		●		●	●
●		●	●	●		●
●		●	●	●		●
●					●	●

(b)

Table: (a) A; (b) A^c

Region filling example

•	•	•	•	•	•	•
•	•			•	•	•
•		•	•		•	•
•		•	•		•	•
•	•		•		•	•
•	•		•		•	•
•		•	•	•		•
•		•	•	•		•
•					•	•

(a)

		6	5			
		5	4			
			3			
			2			
		2	1	2		
		1	p	1		

(b)

Table: (a) A^c; (b) Filled region: $X_0 = \{p\}$, $X_1 = \{p, 1\}$, $X_2 = \{p, 1, 2\}$, ...

Example

(a) (b) (c)

Figure: (a) Original image; (b) boarder extraction; (c) region filling.

Outline

1. Mathematical Morphology Applications
   - Boundary extraction
   - Noise removal
   - Hit-or-miss transform
   - Region filling
   - Connected components
   - Thinning

Connected components

Similar algorithm can be used to fill a connected component. Assume Y is a connected component in set A, and a point $p \in Y$ is known. Then all the elements of Y can be found by filling up the rest of the component starting from the pixel p.

- Using a structuring element, starting from a pixel p, we fill up the rest of the component by creating a sequence of sets

$$X_0 = \{p\}, X_1, X_2, \dots \tag{5}$$

such that

$$X_k = (X_{k-1} \oplus B) \cap A \tag{6}$$

until $X_k = X_{k-1}$.

Connected components

•	•		•	•	
•	•	•		•	
			•	•	•
•	•	•			
•	•	•			
•	•	•			

(a)

2	1	2			
1	p	1			
2	1	2			

(b)

5	4		4	4	
5	4	3		3	
			2	3	4
1	1	1			
1	p	1			
1	1	1			

(c)

Table: (a) Original image; (b) Using a cross SE; (c) Using a square SE.

Connected components cont.

- $N_4(p)$ - 4-connected. The pixel p with coordinates (x, y) has two horizontal and two vertical neighbours – $(x+1, y), (x-1, y), (x, y+1), (x, y-1)$
- $N_8(p)$ - 8-connected. The pixel p has 8 neighbours, include $N_4(p)$ and 4 diagonal neighbours, i.e., $N_D(p) = \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$.
- Two pixels p and q are 4-adjacent if $q \in N_4(p)$, and 8-adjacent if $q \in N_8(p)$.
- Two foreground pixels p and q are said to be 4-connected if there exists a 4-connected path between them, consisting of foreground pixels only.

WITS UNIVERSITY

Connected components cont.

- They are 8-connected if there exists an 8-connected path between them.
- For any foreground pixel, p, the set of all foreground pixels connected to it is called the connected component containing p.

(a)

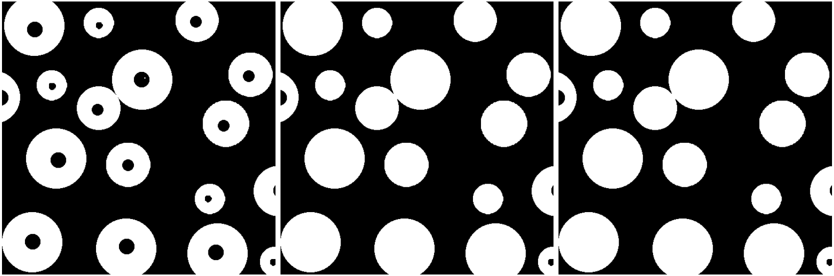
1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0

(b)

1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0

Table: (a) 4-connected components; (b) 8-connected components

Connected components cont.



(a) (b) (c)

Figure: (a) Original image; (b) After filling using *imfill*; (c) Connected components.

For (b), in Matlab using `im_F=imfill(im,'holes');` or in Python using Scipy `binary_fill_holes(im, SE)`, and (c), in Matlab using `bwlabel(im_F,8);` or in Python using `label(im)` in `skimage.measure`.

WITS UNIVERSITY

Outline

1. Mathematical Morphology Applications
   - Boundary extraction
   - Noise removal
   - Hit-or-miss transform
   - Region filling
   - Connected components
   - Thinning

Thinning

- The thinning of a set A by a structuring element B, denoted by $A \otimes B$, can be defined in terms of the hit-or-miss transform:

$$A \otimes B = A - (A \circledast B) = A \cap (A \circledast B)^c. \tag{7}$$

- As we are interested only in pattern matching with the structuring elements, so no background operation is required in the hit-or-miss transform.
- A more useful expression of thinning is based on a sequence of structuring elements:

$$\{B\} = \{B^1, B^2, B^3, \dots, B^n\} \tag{8}$$

where B^i is rotated version of B^{i-1}. Using $\{B\}$, thinning is defined as

$$A \otimes \{B\} = ((\dots((A \otimes B^1) \otimes B^2) \dots) \otimes B^n) \tag{9}$$

The process is to thin A by one pass with B^1, then thin the result with one pass of B^2, and so on.

Thinning

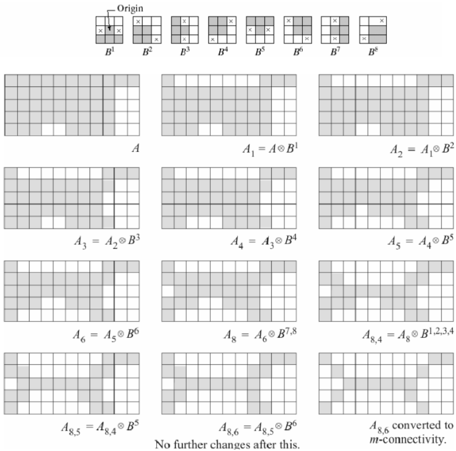


Figure: Thinning example

Thinning

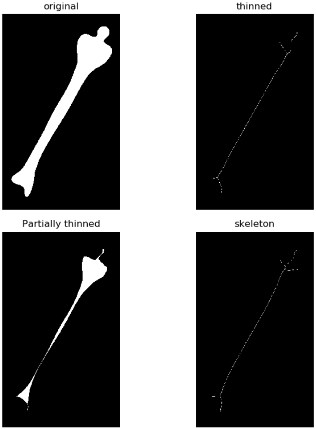


Figure: Thinning example

Thinning

In Matlab, you can use

```
g = bwmorph(im, 'thin', i);
```

where i can be in integer, such as 1 or 2; when $i = Inf$, it gives the skeletonization. In Python (`skimage.morphology`),

```
im_thin_partial = thin(im, max_iter=20)
```

gives the partial thinning, while

```
im_thin_partial = thin(im)
```

gives the skeletonization.

WITS UNIVERSITY