(9) Morphological Image Processing

- Morphology
  → A branch of biology that deals with the form and structure of animals and plants

- Mathematical morphology
  → A tool to extract image components for representing and describing region shapes
  • E.g.: boundary, skeleton, convex hull…

(a) Basic set operations

- Definitions
  • If \( w \) is an element of set \( A \): \( w \in A \)
  • If \( w \) is not an element of \( A \): \( w \notin A \)
  • If set \( B \) of pixel coordinates satisfies a condition: \( B = \{ w \mid \text{condition} \} \)
  • Complement of \( A \): \( A^c = \{ w \mid w \notin A \} \)
• Union of $A$ and $B$: $A \cup B$
• Intersection of $A$ and $B$: $A \cap B$
• Difference of $A$ and $B$: $A - B = \{ w \mid w \in A, w \notin B \} = A \cap B^c$
• Reflection of $B$: $\hat{B} = \{ w \mid w = -b, \text{ for } b \in B \}$
• Translation of $A$ by point $z = (z_1, z_2)$: $(A)_z = \{ c \mid c = a + z, \text{ for } a \in A \}$

- **MATLAB set operations on binary images**

<table>
<thead>
<tr>
<th>Set Operation</th>
<th>MATLAB Expression</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \cap B$</td>
<td>$A &amp; B$</td>
<td>AND</td>
</tr>
<tr>
<td>$A \cup B$</td>
<td>$A \mid B$</td>
<td>OR</td>
</tr>
<tr>
<td>$A^c$</td>
<td>$\sim A$</td>
<td>NOT</td>
</tr>
<tr>
<td>$A - B$</td>
<td>$A &amp; \sim B$</td>
<td>DIFFERENCE</td>
</tr>
</tbody>
</table>

**Dilation**
- Dilation: “grow” or “thicken” an object in a binary image
  • Extent of thickening controlled by a *structuring element*
• Dilation of image $A$ and structuring element $B$: $A \oplus B$

$$A \oplus B = \{ z \mid (\hat{B})_z \cap A \neq \emptyset \}$$

→ The set of all points $z$ such that the intersection of $(\hat{B})_z$ with $A$ is nonempty

• E.g., a five-pixel-long diagonal line with the origin at the center

<table>
<thead>
<tr>
<th>0 0 0 0 0 0 0 0 0 0</th>
<th>0 0 0 0 0 0 0 0 0 0</th>
<th>0 0 0 0 0 0 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1 1 1 1 1 0 0</td>
<td>0 0 0 1 1 1 1 1 0 0</td>
<td>0 0 0 1 1 1 1 1 0 0</td>
</tr>
<tr>
<td>0 0 0 1 1 1 1 1 0 0</td>
<td>0 0 0 1 1 1 1 1 0 0</td>
<td>0 0 0 1 1 1 1 1 0 0</td>
</tr>
<tr>
<td>0 0 0 1 1 1 1 1 0 0</td>
<td>0 0 0 1 1 1 1 1 0 0</td>
<td>0 0 0 1 1 1 1 1 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

→ When the structuring element overlaps 1-valued pixels, the pixel at the origin is marked 1

• Commutative: $A \oplus B = B \oplus A$
• Associative: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
  * If $B = (B_1 \oplus B_2)$, then $A \oplus B = A \oplus (B_1 \oplus B_2) = (A \oplus B_1) \oplus B_2$
    → Dilate $A$ by $B_1$, and then dilate the result by $B_2$ (decomposition)
  * E.g., decomposing a structuring element saves computational cost
    → MATLAB decomposes structuring element automatically

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\oplus
\begin{array}{c}
1 \\
1 \\
1 \\
1 \\
1 \\
\end{array}
\]

• MATLAB: use dilation to bridge gaps
  $A = \text{imread('text.tif'); } B = [0 \ 1 \ 0; \ 1 \ 1 \ 1; \ 0 \ 1 \ 0]; \ A2 = \text{imdilate}(A, B); \ \text{imshow}(A2);$
• **strel** function: create morphological structuring elements
  \[ SE = \text{strel}(\text{shape}, \text{parameters}) \]
  
  * **shape**: 'arbitrary', 'diamond', 'disk', 'line', 'square', 'rectangle'...

(c) **Erosion**

- Erosion: “shrink” or “thin” an object in a binary image
  
  • Extent of shrinking controlled by a *structuring element*
  
  • Erosion of image \( A \) and structuring element \( B \): \( A \ominus B \)

  \[
  A \ominus B = \{ z \mid (B)_z \cap A^c \neq \emptyset \}
  \]

  → The set of all points \( z \) such that the intersection of \( (B)_z \) with \( A^c \) is nonempty

  • E.g., a three-pixel-long vertical line with the origin at the center

  \[
  \begin{array}{cccccccccccccc}
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  \end{array}
  \]
When the structuring element overlaps *only* 1-valued pixels, the pixel at the origin is marked 1 (i.e., does not overlap background).

- MATLAB: use erosion to eliminate irrelevant details
  
  ```matlab
  A = imread('dots.tif'); B = ones(7);
  A1 = imerode(A, B); A2 = imdilate(A1, B);
  ```

(d) Opening and closing

- Opening: smooths the contour, breaks narrow isthmuses, and eliminates thin protrusions

  \[
  (A \circ B) = (A \ominus B) \oplus B = \bigcup \{(B)_z \mid (B)_z \subseteq A\}
  \]

  → Erosion followed by dilation (\(\bigcup\): union of all the sets inside the braces),
MATLAB: `imopen()`

- Closing: smooths the contour, fuses narrow breaks and long thin gulfs, and eliminates small holes

\[(A \bullet B) = (A \oplus B) \ominus B = \{z \mid (B)_z \cap A \neq \emptyset\}\]

→ Dilation followed by erosion, MATLAB: `imclose()`
(e) Hit-or-miss transformation

- Hit-or-miss transformation: identify special configuration of pixels

\[ A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2) \]

- E.g., identify

\[
\begin{array}{c|c|c}
0 & 1 & 0 \\
1 & 1 & 1 \\
0 & 1 & 0 \\
\end{array}
\rightarrow
\begin{array}{c|c|c|c|c}
B_1 & & & & \\
1 & 1 & 1 & & \\
\end{array}
B_2
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A: & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A \ominus B_1: & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A^c: & & & & & & \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A^c \ominus B_2: & & & & & & \\
1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A \otimes B_1: & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A \otimes B_1: & & & & & & \\
1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
A \otimes B_1: & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
• MATLAB: \( C = \text{bwhitmiss}(A, B1, B2); \)

(f) **Basic morphological operations**

- Boundary extraction: extract the boundary of an object
  \[ \beta(A) = A - (A \ominus B) \]

• MATLAB
  \[
  A = \text{imread('A.tif')}; B = \text{ones}(3); A1 = A - \text{imerode}(A, B);
  \]

- Region filling
  \[
  X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3\ldots
  \]

• \( X_0 \): a background point inside the object; converged when \( X_k = X_{k-1} \)
MATLAB

A = im2bw(imread('eye.tif')); B = [0 1 0; 1 1 1; 0 1 0];
Xk = zeros(size(A)); Xk1 = Xk; Xk(85, 70) = 1;
while any(Xk(:) ~= Xk1(:))
    Xk1 = Xk;
    Xk = imdilate(Xk1, B) & ~A;
end
A1 = Xk | A;

* Problem: need to find the initial point
  → Solution?

- Extraction of connected components
\[ X_k = (X_{k-1} \oplus B) \cap A \quad k = 1, 2, 3… \]

- \( X_0 \): a point of the object; converged when \( X_k = X_{k-1} \)
- **MATLAB**
  
  ```matlab
  A = im2bw(imread('a.tif')); B = ones(3);
  Xk = zeros(size(A)); Xk1 = Xk; Xk(30, 40) = 1;
  while any(Xk(:) ~= Xk1(:))
    Xk1 = Xk;
    Xk = imdilate(Xk1, B) & A;
  end
  A1 = Xk;
  ```

- **MATLAB** `bwlabel`: find all connected components
[label number] = bwlabel(im, 4); or [label number] = bwlabel(im, 8);

* label: output image with labeled objects (4- or 8-connectivity)
* number: the number of labeled objects

- Convex hull
  - A set $A$ is convex if the straight line segment joining any two points in $A$ lies entirely within $A$
  - Convex hull $H$ of set $S$ is the smallest convex set containing $S$
  - $H - S$: convex deficiency of $S$

![Convex hull image](image)

- Four structuring elements: $B^i$, $i = 1, 2, 3, 4$, ($\times$: don’t care)
Convex hull of $A$: $C(A)$

$$X_k^i = (X_{k-1}^i \otimes B^i) \cup A, \quad i = 1, 2, 3, 4$$

$$D^i = X_{\text{conv}}^i, \quad C(A) = \bigcup_{i=1}^{4} D^i$$

- Thinning

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

where $\{B\} = \{B_1, B_2, \ldots, B^n\}$

- Eight structuring elements: $B^i, i = 1, 2, \ldots, 8$
- Problem: connectivity not guaranteed

- Thickening

\[ A \circ B = A \cup (A \otimes B) \]

where \( \{B\} = \{B^1, B^2, \ldots, B^n\} \)

- Eight structuring elements are the same as those of thinning

- Skeletonization

  - Repeatedly delete the contour points provided the following conditions are satisfied
    * End points are not deleted
    * Connectivity are not broken
    * Do not cause excessive erosion of the region

  - Algorithm: repeat following steps until no contour points
    (1) Delete all contour points according to Definition 1
    (2) Delete all contour points according to Definition 2

  - Definition 1: right, bottom, and upper left corner contour points
(a) $2 \leq N(p_1) \leq 6$
(b) $T(p_1) = 1$
(c) $p_2 \cdot p_4 \cdot p_6 = 0$
(d) $p_4 \cdot p_6 \cdot p_8 = 0$

* $N(p_1)$: number of 1’s in the neighborhood of $p_1$

$$N(p_1) = \sum_{i=2}^{9} p_i$$

* $T(p_1)$: number of 0-1 transitions in the ordered sequence $p_2, p_3, \ldots, p_8, p_9, p_2$ (clockwise)

# E.g.:

$$N(p_1) = 4, \quad T(p_1) = 3$$

$$p_2 \cdot p_4 \cdot p_6 = 0$$

$$p_4 \cdot p_6 \cdot p_8 = 0$$

$$\begin{array}{ccc}
0 & 0 & 1 \\
1 & p_1 & 0 \\
1 & 0 & 1 \\
\end{array}$$

• Definition 2: left, top, and lower right corner contour points
  (a) and (b) are the same as those in definition 1
  (c') $p_2 \cdot p_4 \cdot p_8 = 0$;
\( (d') \ p_2 \cdot p_6 \cdot p_8 = 0; \)

- **Description:**
  * (a): if there is only one 1 in the neighborhood, \( p_1 \) is an end point and should not be deleted; if there are seven 1’s, deleting \( p_1 \) would cause erosion; if there are eight 1’s, \( p_1 \) is not a contour point
  
  * (b): \( T(p_1) \neq 1 \): \( p_1 \) is an arc point and deleting \( p_1 \) would break the connectivity
  
  - If the mask consists of only two connected regions, \( T(p_1) = 1 \)

* (c) and (d): \( p_4 = 0 \) or \( p_6 = 0 \) or \( p_2 = p_8 = 0 \)
  - Right, bottom, and upper left corner contour points
* (c') and (d'): $p_2 = 0$ or $p_8 = 0$ or $p_4 = p_6 = 0$
→ Top, left, and lower right corner contour points

• Examples of contour definitions 1 and 2:

(Def. 1)  (Def. 2)
- E.g.:

<table>
<thead>
<tr>
<th>Image</th>
<th>Thinning</th>
<th><code>bwmorph(..., 'thin', Inf)</code></th>
<th><code>bwmorph(..., 'skel', Inf)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image](117x87 to 246x216)</td>
<td>![Image](252x87 to 381x216)</td>
<td>![Image](387x87 to 516x216)</td>
<td>![Image](556x216 to 600x216)</td>
</tr>
<tr>
<td>![Image](117x260 to 246x389)</td>
<td>![Image](252x260 to 381x389)</td>
<td>![Image](387x260 to 516x389)</td>
<td>![Image](556x389 to 600x389)</td>
</tr>
<tr>
<td>![Image](117x435 to 246x564)</td>
<td>![Image](252x435 to 381x564)</td>
<td>![Image](387x435 to 516x564)</td>
<td>![Image](556x564 to 600x564)</td>
</tr>
</tbody>
</table>
- MATLAB morphing function

\[ \text{bwmorph}(bw, \text{operation}) \quad \text{or} \quad \text{bwmorph}(bw, \text{operation}, n) \]

- \textit{bw}: binary image, \textit{n}: number of repetitions of the operation
- \textit{operation}:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'bothat'</td>
<td>&quot;Bottom hat&quot; operation using a $3 \times 3$ structuring element; use \text{imbothat} for other structuring elements</td>
</tr>
<tr>
<td>'erode'</td>
<td>Erosion using a $3 \times 3$ structuring element; use \text{imerode} for other structuring elements</td>
</tr>
<tr>
<td>'shrink'</td>
<td>Shrink objects with no holes to points; shrink objects with holes to rings</td>
</tr>
<tr>
<td>'bridge'</td>
<td>Connect pixels separated by single-pixel gaps</td>
</tr>
<tr>
<td>'fill'</td>
<td>Fill in single-pixel holes; use \text{imfill} for larger holes</td>
</tr>
<tr>
<td>'skel'</td>
<td>Skeletonize an image</td>
</tr>
<tr>
<td>'clean'</td>
<td>Remove isolated foreground pixels</td>
</tr>
<tr>
<td>'hbreak'</td>
<td>Remove H-connected foreground pixels</td>
</tr>
<tr>
<td>'spur'</td>
<td>Remove spur pixels</td>
</tr>
<tr>
<td>'close'</td>
<td>Closing using a $3 \times 3$ structuring element; use \text{imclose} for other structuring elements</td>
</tr>
<tr>
<td>'majority'</td>
<td>Makes pixel ( p ) a foreground pixel if ( N_8(p) \geq 5 ); otherwise make ( p ) a background pixel</td>
</tr>
<tr>
<td>'thicken'</td>
<td>Thicken objects without joining disconnected 1s</td>
</tr>
<tr>
<td>'diag'</td>
<td>Fill in around diagonally connected foreground pixels</td>
</tr>
<tr>
<td>'open'</td>
<td>Opening using a 3×3 structuring element; use <code>imopen</code> for other structuring elements</td>
</tr>
<tr>
<td>'thin'</td>
<td>Thin objects without holes to minimally connected strokes; thin objects with holes to rings</td>
</tr>
<tr>
<td>'dilate'</td>
<td>Dilation using a 3×3 structuring element; use <code>imdilate</code> for other structuring elements</td>
</tr>
<tr>
<td>'remove'</td>
<td>Remove “interior” pixels</td>
</tr>
<tr>
<td>'tophat'</td>
<td>“Top hat” operation using a 3×3 structuring element; use <code>imtophat</code> for other structuring elements</td>
</tr>
</tbody>
</table>

(g) **Gray-scale morphology**

- Morphology
  - Binary image: change the shape of the foreground objects
  - Gray-scale image: change the shape of the image surface (3D)

- Gray-scale dialtion and erosion
  - Dialtion: \( f \oplus b(x, y) = \max \{f(x-x', y-y') + b(x', y') \mid (x', y') \in D_b\} \)
where $D_b$ is the domain of $b$, and $f(x, y)$ is assumed to equal $-\infty$ outside the domain of $f$

• Erosion: $f \ominus b(x, y) = \min\{f(x+x', y+y') - b(x', y') \mid (x', y') \in D_b\}$
  where $D_b$ is the domain of $b$, and $f(x, y)$ is assumed to equal $+\infty$ outside the domain of $f$

- Opening and closing
- Example: using opening to compensate for nonuniform background illumination
  
  - Fig. 1: rice grains on nonuniform background (darker towards bottom), `f = imread('rice.tif')`
  - Fig. 2: simple thresholding: `fbw = im2bw(f, graythresh(f))` causes grains improperly separated at the bottom portion
  - Fig. 3: opening with `se = strel('disk', 10); fo = imopen(f, se)`: since the size of `se` is set to be larger than the grains, only background remains
  - Fig. 4: subtracting from the original `f2 = imsubtract(f, fo)`: results in a more uniform background
  - Fig. 5: thresholding with `fbw = im2bw(f2, graythresh(f2))` obtains a better result

  Subtracting an opened image from the original is called the top-hat transformation, a single step in Matlab: `f2 = imtophat(f, se)`