Hit-or-miss transform

- Used to extract pixels with specific neighbourhood configurations from an image
- Grey scale extension exist
- Uses two structure elements B1 and B2 to find a given foreground and background configuration, respectively

\[ HMT_B(X) = \{ x | (B_1)_x \subseteq X, (B_2)_x \subseteq X^C \} \]

- Example:
9.4 The hit-or-miss transformation

Illustration...
Objective is to find a disjoint region (set) in an image

If $B$ denotes the set composed of $X$ and its background, the match/hit (or set of matches/hits) of $B$ in $A$, is

$$A \odot B = (A \ominus X) \cap [A^c \ominus (W - X)]$$

Generalized notation: $B = (B_1, B_2)$

- $B_1$: Set formed from elements of $B$ associated with an object
- $B_2$: Set formed from elements of $B$ associated with the corresponding background

[Preceeding discussion: $B_1 = X$ and $B_2 = (W - X)$]

More general definition:

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

$A \odot B$ contains all the origin points at which, simultaneously, $B_1$ found a hit in $A$ and $B_2$ found a hit in $A^c$
Hit-or-miss transform

\[ HMT_B(X) = \{ x | (B_1)_x \subseteq X, (B_2)_x \subseteq X^C \} \]

- Can be written in terms of an intersection of two erosions:

\[ HMT_B(X) = \varepsilon_{B_1}(X) \cap \varepsilon_{B_2}(X^c) \]
**Hit-or-miss transform**

- Simple example usages - locate:
  - Isolated foreground pixels
    - no neighbouring foreground pixels
  - Foreground endpoints
    - one or zero neighbouring foreground pixels
  - Multiple foreground points
    - pixels having more than two neighbouring foreground pixels
  - Foreground contour points
    - pixels having at least one neighbouring background pixel
Hit-or-miss transform example

- Locating 4-connected endpoints

SEs for 4-connected endpoints  Resulting Hit-or-miss transform
Hit-or-miss opening

- **Objective:** keep all points that fit the SE.
- **Definition:**
  \[
  \gamma_B(X) = \delta_{\tilde{B}_1} HMT_B(X) = \delta_{\tilde{B}_1} \varepsilon_{B_1}(X) \cap \varepsilon_{B_2}(X^c)
  \]
hit-or-miss opening example

Resulting hit-or-miss opening
• Alternative definition:

\[ A \circledast B = (A \ominus B_1) - (A \oplus \hat{B}_2) \]

• A background is necessary to detect disjoint sets

• When we only aim to detect certain patterns within a set, a background is not required, and simple erosion is sufficient

9.5 Some basic morphological algorithms

When dealing with binary images, the principle application of morphology is extracting image components that are useful in the representation and description of shape

9.5.1 Boundary extraction

The boundary \( \beta(A) \) of a set \( A \) is

\[ \beta(A) = A - (A \ominus B), \]

where \( B \) is a suitable structuring element
Illustration...

Example 9.5: Morphological boundary extraction
9.5.2 Region filling

- Begin with a point \( p \) inside the boundary, and then fill the entire region with 1’s

- All non-boundary (background) points are labeled 0

- Assign a value of 1 to \( p \) to begin...

- The following procedure fills the region with 1’s,

\[
X_k = (X_{k-1} \oplus B) \cap A_c, \quad k = 1, 2, 3, \ldots,
\]

where \( X_0 = p \), and \( B \) is the symmetric structuring element in figure 9.15 (c)

- The algorithm terminates at iteration step \( k \) if \( X_k = X_{k-1} \)

- The set union of \( X_k \) and \( A \) contains the filled set and its boundary

Note that the intersection at each step with \( A_c \) limits the dilation result to inside the region of interest
Example 9.6: Morphological region filling

Figure 9.15 (a) Set $A$, (b) Complement of $A$, (c) Structuring element $B$, (d) Initial point inside the boundary, (e)-(h) Various steps of Eq. (9.5-2), (i) Final result [union of (a) and (h)].

Figure 9.16 (a) Binary image (the white dot inside one of the regions is the starting point for the region-filling algorithm), (b) Result of filling that region (c) Result of filling all regions.
9.5.3 Extraction of connected components

Let \( Y \) represent a connected component contained in a set \( A \) and assume that a point \( p \) of \( Y \) is known. Then the following iterative expression yields all the elements of \( Y \):

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

where \( X_0 = p \), and \( B \) is a suitable structuring element. If \( X_k = X_{k-1} \), the algorithm has converged and we let \( Y = X_k \).

This algorithm is applicable to any finite number of sets of connected components contained in \( A \), assuming that a point is known in each connected component.

**FIGURE 9.17** (a) Set \( A \) showing initial point \( p \) (all shaded points are valued 1, but are shown different from \( p \) to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.
Example 9.7:

9.5.4 Convex hull

Morphological algorithm for obtaining the convex hull, $C(A)$, of a set $A$...

Let $B_1$, $B_2$, $B_3$ and $B_4$ represent the four structuring elements in figure 9.19 (a), and then implement the equation ...
\[ X_k^i = (X_{k-1} \boxplus B^i) \cup A, \quad i = 1, 2, 3, 4, \quad k = 1, 2, \ldots, \quad X_0^i = A \]

Now let \( D^i = X_{\text{conv}}^i \), where “conv” indicates convergence in the sense that \( X_k^i = X_{k-1}^i \). Then the convex hull of \( A \) is

\[ C(A) = \bigcup_{i=1}^{4} D^i \]
Shortcoming of above algorithm: convex hull can grow beyond the minimum dimensions required to guarantee convexity.

Possible solution: Limit growth so that it does not extend past the vertical and horizontal dimensions of the original set of points.

![Diagram showing the result of limiting growth of convex hull algorithm](image)

**FIGURE 9.20** Result of limiting growth of convex hull algorithm to the maximum dimensions of the original set of points along the vertical and horizontal directions.

Boundaries of greater complexity can be used to limit growth even further in images with more detail.

### 9.5.5 Thinning

The thinning of a set $A$ by a structuring element $B$:

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

Symmetric thinning: sequence of structuring elements,

$$\{B\} = \{B^1, B^2, B^3, \ldots, B^n\},$$

where $B^i$ is a rotated version of $B^{i-1}$. 

\[ A \otimes \{B\} = (((A \otimes B^1) \otimes B^2) \ldots) \otimes B^n \]

Illustration: Note that figure 9.21 (in the handbook) has many errors — this one is correct...

\[ A_1 = A \otimes B_1, \quad A_2 = A \otimes B_2, \quad A_3 = A_2 \otimes B_3, \quad A_4 = A_3 \otimes B_4, \quad A_5 = A_4 \otimes B_5, \quad A_6 = A_5 \otimes B_6, \quad A_7 = A_6 \otimes B_7, \quad A_8 = A_7 \otimes B_8, \quad A_{8,4} = A_8 \otimes B^{1,2,3,4},\quad A_{8,5} = A_{8,4} \otimes B^5, \quad A_{8,6} = A_{8,5} \otimes B^6, \quad A_{8,6} \text{ converted to } m\text{-connectivity.} \]
Thinning

- Used to shrink objects in binary images
- Differs from erosion in that objects are never completely removed
  - Preserving the homotopy is often an objective
- Successive thinning until stability results in object skeletons
- Thinning is defined as:
  \[
  THIN(X, B) = X \setminus HMT_B(X)
  \]
- Example structure elements used in thinning (rotated 90 degrees 3 times to create 8 structure elements):
Skeletons

- Compact or minimal representation of objects in an image while retaining homotopy of the image
- As stated earlier, the skeletons of objects in an image can be found by successive thinning until stability
- The thinning cannot be executed in parallel since this may cause the homotopy of the image to change
- Example:
Skeleton

- The skeleton of an object is often defined as the medial axis of that object.
  - Pixels are then defined to be skeleton pixels if they have more than one “closest neighbours”.
- Some skeleton algorithms are based on this definition and are computed through the distance transform
- Other algorithms produce skeletons that are smaller than the defined medial axis (such as minimal skeletons)
Skeletons

- Problem: Finding a minimal representation
  - Solution 1: Pruning of smaller branches
    - Can use HMT to locate and remove endpoints successively

Skeleton with unwanted branches
Skeletons

• Pruning is dependent on parameter choices (maximum branch length of branches to be removed)

• Solution 2: Skeleton algorithm producing minimal skeleton
  – One such algorithm is described in [Thinning Methodologies-A Comprehensive Survey," IEEE TrPAMI, vol. 14, no. 9, pp. 869-885, 1992.]
  – HMT is not used in this algorithm

Minimal skeleton
Skeletons

- Some skeletons can be used to reconstruct the original objects in an image.

Example taken from SDC Morphology Toolbox for MATLAB (www.mmorph.com)
9.5.7 Skeletons

The algorithm proposed in this section is similar to the medial axis transformation (MAT). The MAT transformation is discussed in section 11.1.5 and is far inferior to the skeletonization algorithm introduced in section 11.1.5. The skeletonization algorithm proposed in this section also does not guarantee connectivity. We therefore do not discuss this algorithm.

Illustration of the above comments...

---

**FIGURE 9.23**
(a) Set A.
(b) Various positions of maximum disks with centers on the skeleton of A.
(c) Another maximum disk on a different segment of the skeleton of A.
(d) Complete skeleton.
A further illustration...

<table>
<thead>
<tr>
<th>$k$</th>
<th>$A \otimes kB$</th>
<th>$(A \otimes kB) \cdot B$</th>
<th>$S_k(A)$</th>
<th>$\bigcup_{k=0}^{K} S_k(A)$</th>
<th>$S_k(A) \oplus kB$</th>
<th>$\bigcup_{k=0}^{K} S_k(A) \oplus kB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**Figure 9.24** Implementation of Eqs. (9.5-11) through (9.5-15). The original set is at the top left, and its morphological skeleton is at the bottom of the fourth column. The reconstructed set is at the bottom of the sixth column.

### 9.5.8 Pruning

- Cleans up “parasitic” components left by thinning and skeletonization
- Use combination of morphological techniques
**Illustrative problem:** hand-printed character recognition

- Analyze shape of skeleton of character
-Skeletons characterized by spurs ("parasitic" components)
- Spurs caused during erosion of non-uniformities in strokes
- We assume that the length of a parasitic component does not exceed a specified number of pixels

**Figure 9.25**
(a) Original image, (b) and (c) Structuring elements used for deleting end points, (d) Result of three cycles of thinning, (e) End points of (d), (f) Dilation of end points conditioned on (a), (g) Pruned image.
Any branch with three or less pixels is to be eliminated

(1) Three iterations of:

$$X_1 = A \otimes \{B\}$$

(2) Find all the end points in $X_1$:

$$X_2 = \bigcup_{k=1}^{8} (X_1 \oplus B^k)$$

(3) Dilate end points three times, using $A$ as a delimiter:

$$X_3 = (X_2 \oplus H) \cap A, \quad H = \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{array}$$

(4) Finally:

$$X_4 = X_1 \cup X_3$$
<table>
<thead>
<tr>
<th>Operation</th>
<th>Equation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>((A)_z = { w</td>
<td>w = a + z, \text{ for } a \in A } )</td>
</tr>
<tr>
<td>Reflection</td>
<td>( \hat{B} = { w</td>
<td>w = -b, \text{ for } b \in B } )</td>
</tr>
<tr>
<td>Complement</td>
<td>( A' = { w</td>
<td>w \notin A } )</td>
</tr>
<tr>
<td>Difference</td>
<td>( A - B = { w</td>
<td>w \in A, w \notin B } = A \cap B' )</td>
</tr>
<tr>
<td>Dilation</td>
<td>( A \oplus B = { z</td>
<td>(\hat{B}_z \cap A) \neq \emptyset } )</td>
</tr>
<tr>
<td>Erosion</td>
<td>( A \ominus B = { z</td>
<td>(\hat{B}_z \subseteq A } )</td>
</tr>
<tr>
<td>Opening</td>
<td>( A \circ B = (A \ominus B) \oplus B )</td>
<td>Smooths contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks. (I)</td>
</tr>
<tr>
<td>Closing</td>
<td>( A \bullet B = (A \oplus B) \ominus B )</td>
<td>Smooths contours, fuses narrow breaks and long thin gulfs, and eliminates small holes. (I)</td>
</tr>
<tr>
<td>Hit-or-miss</td>
<td>( A \boxplus B = (A \ominus B_1) \cap (A' \ominus B_2) )</td>
<td>The set of points (coordinates) at which, simultaneously, ( B_1 ) found a match (“hit”) in ( A ) and ( B_2 ) found a match in ( A' ).</td>
</tr>
<tr>
<td>transform</td>
<td>( = (A \ominus B_1) - (A \ominus B_2) )</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td>( \beta(A) = A - (A \ominus B) )</td>
<td>Set of points on the boundary of set ( A ). (I)</td>
</tr>
<tr>
<td>extraction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Region filling</td>
<td>( X_k = (X_{k-1} \oplus B) \cap A; X_0 = p \text{ and } k = 1, 2, 3, \ldots )</td>
<td>Fills a region in ( A ), given a point ( p ) in the region. (II)</td>
</tr>
<tr>
<td>Connected</td>
<td>( X_k = (X_{k-1} \oplus B) \cap A; X_0 = p \text{ and } k = 1, 2, 3, \ldots )</td>
<td>Finds a connected component ( Y ) in ( A ), given a point ( p ) in ( Y ). (I)</td>
</tr>
<tr>
<td>components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convex hull</td>
<td>( X_i^j = (X_{i-1}^j \oplus B^j) \cup A; i = 1, 2, 3, 4; k = 1, 2, 3, \ldots; X_1^0 = A; \text{ and } D^j = X_{x_{conv}}^j )</td>
<td>Finds the convex hull ( C(A) ) of set ( A ), where “conv” indicates convergence in the sense that ( X_k' = X_{k-1}' ). (III)</td>
</tr>
<tr>
<td>Operation</td>
<td>Equation</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| Thinning  | $A \odot B = A - (A \odot B)$  
$= A \cap (A \odot B)^c$ | Thins set $A$. The first two equations give the basic definition of thinning. The last two equations denote thinning by a sequence of structuring elements. This method is normally used in practice. (IV) |
| Thickening| $A \odot B = A \cup (A \odot B)$ | Thickens set $A$. (See preceding comments on sequences of structuring elements.) Uses IV with 0's and 1's reversed. |
| Skeletons | $S(A) = \bigcup_{k=0}^{K} S_k(A)$  
$S_k(A) = \bigcup_{z=0}^{K} \{(A \odot kB)^c \}$  
Reconstruction of $A$: $A = \bigcup_{k=0}^{K} (S_k(A) \odot kB)$  
$A = \bigcup_{k=0}^{K} (S_k(A) \odot kB)$ | Finds the skeleton $S(A)$ of set $A$. The last equation indicates that $A$ can be reconstructed from its skeleton subsets $S_k(A)$. In all three equations, $K$ is the value of the iterative step after which the set $A$ erodes to the empty set. The notation $(A \odot kB)$ denotes the $k$th iteration of successive erosion of $A$ by $B$. (I) |
| Pruning  | $X_1 = A \odot \{B\}$  
$X_2 = \bigcup_{k=1}^{s} (X_1 \odot B')$  
$X_3 = (X_2 \odot H) \cap A$  
$X_4 = X_1 \cup X_3$ | $X_4$ is the result of pruning set $A$. The number of times that the first equation is applied to obtain $X_1$ must be specified. Structuring elements $V$ are used for the first two equations. In the third equation $H$ denotes structuring element 1. |
3D skeletons

- 3D skeletons can be divided into two groups: medial surfaces and medial lines:
3D skeletons

- Not all 2D skeleton algorithms can be directly extended into 3D
  - Such as the example skeleton algorithm producing a minimal skeleton
- 3D skeletonization is a difficult task and much research effort has been put into this field
Thickening

- Thickening consists of adding border pixels instead of removing them:
  \[ \text{THICK}(X, B) = X \cup HMT_B(X) \]
- Thickening and thinning are dual operators:
  \[ \text{THIN}(X, B) = C \text{THICK}(X, B^c) C \]
9.5.6 Thickening

Thickening is the morphological dual of thinning and is defined by

\[ A \odot B = A \cup (A \odot B), \]

where \( B \) is a structuring element.

Similar to thinning...

\[ A \odot \{B\} = (\ldots ((A \odot B^1) \odot B^2) \ldots) \odot B^n \]

Structuring elements for thickening are similar to those of figure 9.21 (a), but with all 1’s and 0’s interchanged.

A separate algorithm for thickening is seldom used in practice — we thin the background instead, and then complement the result.

\[ a \quad b \]
\[ c \quad d \]

**Figure 9.22** (a) Set \( A \). (b) Complement of \( A \). (c) Result of thinning the complement of \( A \). (d) Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.
Practical example – reconstruction of liver vessels from CT or MR scans

• The liver is largest internal organ in the human body
• Has several functions:
  – Secretes bile
  – Produces glycogen
  – Assimilates carbohydrates, fats, and proteins
  – Manufactures essential blood components
  – Filters harmful substances from the blood

(MeVis, Preoperative Planning in Liver Surgery)
Tumours in the liver

- Considered a serious medical complication
- Three ways to remove the tumours
  - Chemotherapy
  - Open surgery
  - Laparoscopic surgery (minimally invasive surgery)

(www.nucleusinc.com)
Planning liver surgery

- CT or MR scans are studied prior to a liver surgery
  - Blood vessels are tracked, and tumours are identified and located
    - Important with respect to the choice of surgical procedure
    - Problem: Human interpretation is time-consuming and error-prone
    - Our goal: To perform this analysis automatically
Automatic modelling of liver anatomy

- Step 1 – Segment the liver
Automatic modelling of liver anatomy

• Step 2 – Segment the liver vessels
Automatic modelling of liver anatomy

- Step 3 – Identify the vessel paths
Automatic modelling of liver anatomy

- Resulting 3D skeleton
- A vessel graph (nodes and interconnections) is then extracted from this skeleton
Automatic modelling of liver anatomy

- Resulting reconstruction and visualisation

Reconstruction based on the skeletons, distances between skeleton structures, and node sizes.

Postprocessing of vessel graph removing unlikely interconnections.
9.6 Extensions to grey-scale images

\( f(x, y) \): Input image

\( b(x, y) \): Structuring element image

9.6.1 Dilation

Grey-scale dilation of \( f \) by \( b \), is defined as

\[
(f \oplus b)(s, t) = \max \{ f(s - x, t - y) + b(x, y) \mid (s - x), (t - y) \in D_f; (x, y) \in D_b \},
\]

where \( D_f \) and \( D_b \) are the domains of \( f \) and \( b \), respectively.
Simple 1D example. For functions of one variable:

\[(f \oplus b)(s) = \max \{f(s - x) + b(x) \mid (s - x) \in D_f; x \in D_b\}\]

General effect of dilation of a grey-scale image:

(1) If all values of \(b(x, y)\) are positive \(\rightarrow\) output image brighter

(2) Dark details are reduced or eliminated, depending on size
9.6.2 Erosion

Grey-scale erosion of $f$ by $b$, is defined as

$$(f \ominus b)(s, t) = \max \{f(s + x, t + y) - b(x, y) \mid (s + x), (t + y) \in D_f; (x, y) \in D_b\},$$

where $D_f$ and $D_b$ are the domains of $f$ and $b$, respectively

**Simple 1D example.** For functions of one variable:

$$(f \ominus b)(s) = \max \{f(s + x) - b(x) \mid (s + x) \in D_f; x \in D_b\}$$
General effect of erosion of a grey-scale image:

(1) If all values of $b(x, y)$ are positive $\Rightarrow$ output image darker

(2) Bright details are reduced or eliminated, depending on size

Example 9.9: Dilation and erosion on grey-scale image

$f(x, y)$: 512 x 512

$b(x, y)$: “flat top”, unit height, size of 5 x 5
9.6.3 Opening and closing

The **opening** of image $f$ by $b$, is defined as

$$ f \circ b = (f \ominus b) \oplus b $$

The **closing** of image $f$ by $b$, is defined as

$$ f \circ b = (f \oplus b) \ominus b $$

**Explanation using “rolling ball”:**

![Diagram](image.png)

**FIGURE 9.30**
(a) A gray-scale scan line.
(b) Positions of rolling ball for opening.
(c) Result of opening.
(d) Positions of rolling ball for closing.
(e) Result of closing.
Opening and closing of “horse image”:

![Images of horse image](image1.png)

**FIGURE 9.31** (a) Opening and (b) closing of Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)

### 9.6.4 Some applications of grey-scale morphology

**Morphological smoothing**

Opening followed by closing $\leadsto$ remove or attenuate bright and dark artifacts or noise

![Morphological smoothing](image2.png)

**FIGURE 9.32** Morphological smoothing of the image in Fig. 9.29(a). (Courtesy of Mr. A. Morris, Leica Cambridge, Ltd.)
Morphological gradient

Definition: \( g = (f \oplus b) - (f \ominus b) \)

Top-hat transformation

Definition: \( h = f - (f \circ b) \)
Textural segmentation

Input image (left): Two texture regions
Output image (right): Boundary between the two regions

Algorithm:

(1) Close input image using succ larger struct elements. When size(struct element) \(\approx\) size(small blobs), blobs are removed

(2) Single opening with struct element that is large in relation to separation between large blobs \(\leadsto\) light patches between blobs removed \(\leadsto\) light region on left, dark region on right

(3) Thresholding \(\leadsto\) boundary
Granulometry

Granulometry: Field that deals with determining the size distribution of particles in an image

Algorithm:

(1) Opening with struct elements of increasing size

(2) Difference between original image and its opening computed after each pass

(3) Differences are normalized and used to construct histogram

The histogram indicates the presence of three predominant particle sizes