Institute of Informatics
Eötvös Loránd University
Budapest, Hungary

Basic Algorithms for Digital Image Analysis: a course

Dmitrij Csetverikov

with help of Attila Lerch, Judit Verestóy, Zoltán Megyesi, Zsolt Jankó and Levente Hajder

http://visual.ipan.sztaki.hu

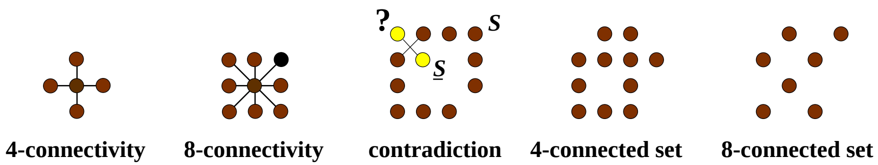
Lecture 11: Binary image processing I

- Basic notions of digital geometry
  - 4-connectivity and 8-connectivity
  - Connected components
  - Holes and borders
  - Shrinking and expanding
- Image data structures and run-length code (RLC)
- RLC-based connected component analysis
- Skeletal representation
  - Medial axis
  - Distance transform and medial axis
  - Thinning and skeleton
  - Summary of skeletal representation

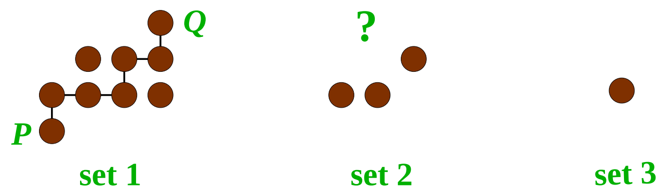
Basic notions of digital geometry

Connectivity in digital images:

- 8 neighbours or 4 neighbours: 8-connectivity, 4-connectivity
- **Contradiction** arises if same type of connectivity is used for object (S) and background ($\underline{S}$): pairs of pixels from the two sets may be connected 'across each other'.
- **Solution**: Use different types of connectivity for S and $\underline{S}$
  - 8-connectivity in $S \Rightarrow$ 4-connectivity in $\underline{S}$
  - 4-connectivity in $S \Rightarrow$ 8-connectivity in $\underline{S}$



Definition: Let S be a set of pixels and pixels $P, Q \in S$. P is **connected to** Q if there exists a sequence of pixels $P = P_0, P_1, \ldots, P_n = Q$, $P_i \in S$, such that P_i is a neighbour of P_{i-1}, $i \in [1, n]$.



Connected pixels and sets. Set 2 is only connected for 8-connectivity.

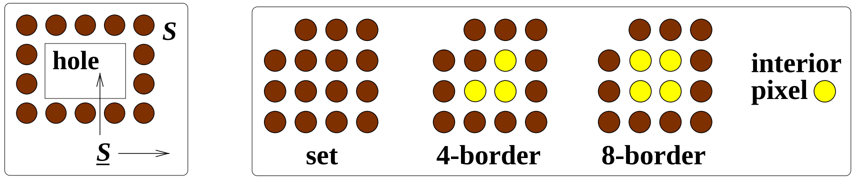
- Relation **'is connected to'** is reflexive, symmetric and transitive

  ⇒ It is a relation of **equivalence**
  ⇒ It partitions the image into connected components

- **Connected components** are the maximal connected sets of pixels

Definitions:

- **Hole** in S is a component of $\underline{S}$ surrounded by S.
- Pixel $P \in S$ is called a **border pixel** if it has a neighbour in $\underline{S}$ in the sense of connectivity used in $\underline{S}$.
  - Otherwise, S is called an **interior pixel**.

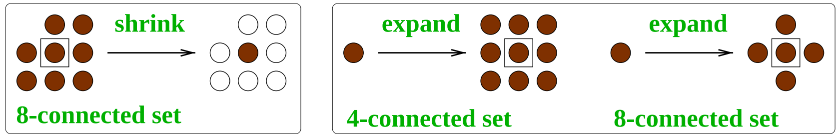


Illustrations to definitions of hole and border.

Operations of shrinking and expanding

Definitions

- $S^{(-1)}$ denotes **shrinking** of S: delete border of S.
  - $S^{(-k)}$ is k**-step shrinking** of S: repeat shrinking k times.
- $S^{(1)}$ denotes **expanding** of S: interchange the connectivities of S and $\underline{S}$, then shrink $\underline{S}$.
  - $S^{(k)}$ is k**-step expanding** of S: repeat expanding k times.



Examples of shrinking and expanding. For 4-connected S, $S^{(-1)}$ is empty: shrinking does not preserve connectivity and may remove objects.

Image data structures and run-length code

Binary image processing: Processing bilevel images using **geometric** criteria.

Different **image data structures** are used for binary image processing. The data structures differ in data compression and in operations they support.

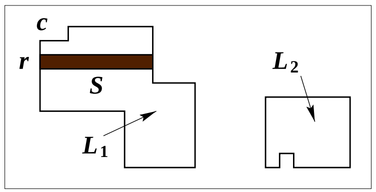
- Original image matrix
- Quadtree (area-based)
- Run-length code (area-based)
- Chain code (contour-based)

Run-length code (RLC):

- A binary image is represented by a set of horizontal **runs**.
- A run is a maximal continuous sequence of object pixels.

Each run R is coded by

- row r and column c of the starting pixel
- length S: number of pixels in the run
- label L: identifies the connected component to which the run belongs



run R : {row r, column c, length S, label L}

Run-length coding. L_1 and L_2 are labels.

An image can be **restored** from its RLC, labelled or unlabelled.

RLC-based connected component analysis

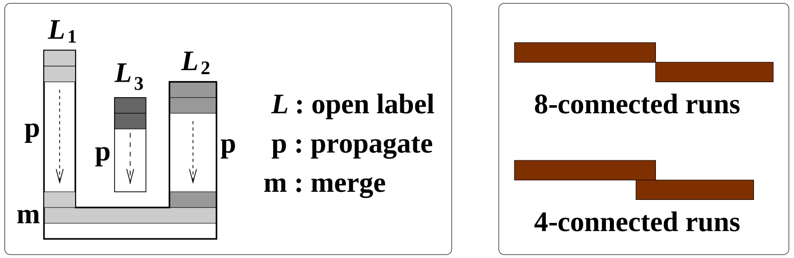
Connected component analysis: Finding connected components of a binary image.

Different algorithms for connected component analysis exist, based on different data structures. In particular:

- The simplest algorithm is **recursive**: Find next unlabelled pixel in image matrix, spread recursively to neighbours, mark visited pixels by current label.
  - Short: a few lines of code
  - Inefficient, or even dangerous, when components are large and recursion is deep
- **Chain coding** by contour tracing: Described later in this course.
- **RLC-based** connected component analysis: Obtain unlabelled RLC first, then assign a label to each run.

A typical RLC-based connected component algorithm allocates a **label equivalency table**, inputs unlabelled RLC, and operates in **two passes**.

- The first pass includes three basic procedures:
  - opening a new label
  - propagating an existing label
  - merging the equivalent labels and updating the equivalency table
- The second pass recomputes the labels based on the equivalency table.



RLC-based connected component analysis.

Algorithm 1: Connected component labelling of RLC

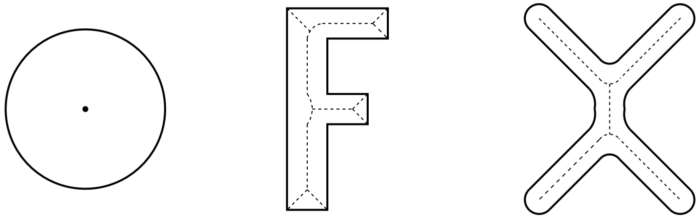
1. Select connectivity (8 or 4). Allocate and initialise equivalency table (matrix) $E_{ij} = 0$ if $i \neq j$, $E_{ii} = 1$. Set label counter $N_L = 0$.
2. Scan RLC and find the next unlabelled run $R(r, c, S)$. Search the previous row $r' = r - 1$ and find runs R'_k, $k = 1, 2, \dots, N_c$, which are **connected to** R. (R'_k are already labelled.)
   - If $N_c = 0$ (no connected runs), **open a new label** by assigning label N_L to run R, then increment N_L.
   - If $N_c = 1$ and there is single connected run R'_1 with label L'_1, then **propagate label** by assigning label L'_1 to run R.
   - If $N_c > 1$, select the smallest of the connected run labels L'_k, L'_s, then **merge labels** by assigning label L'_s to run R and setting $E_{ks} = 1$ for all k.

   Repeat until the end of RLC is reached.
3. Use the final equivalency table E_{ij} to re-label the runs and obtain the true number of connected components.

Skeletal representation

Definition: An interior point of a shape belongs to the **medial axis** (MA) of the shape if this point lies at the **same distance** from two or more **nearest contour points**.

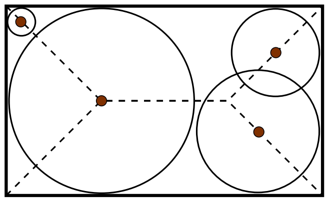
- Medial axis is the **locus** of such points.
- Operation that finds the medial axis of a shape is called **medial axis transform** (MAT).



Examples of medial axes of simple shapes. MA of a disc is its centre. Note that **bisector of each angle** *is a branch of MA.*

Properties of medial axis

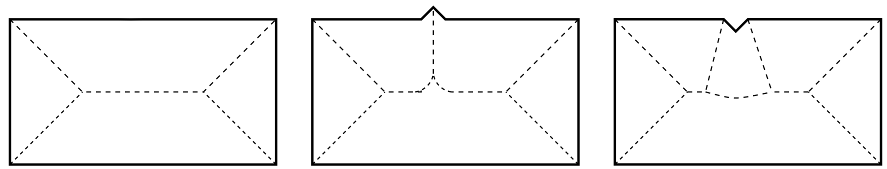
- In **continuous case**, MA is always a **connected** set of points.
- MA is a compact and efficient representation of shapes consisting of **elongated** parts, for instance, letters, chromosomes.
- Medial axis is a **regenerative** description if the **distance** is assigned to each point of MA.
  - Then the shape can be **restored** from its MA as the locus of inscribed circles



Medial axes as the locus of centres of inscribed circles.

Properties of medial axis (continued)

- MA is very sensitive to small (e.g., noisy) variations of shape.
- Medial axis of shape can be obtained using **distance transform** (DT) of shape.
  - DT assigns to each point the distance from that point to contour of shape.
  - In discrete case, distance to the closest background pixels is used. Different discrete approximationsto DT exist.
- MA of **discrete shape** may be **disconnected**.



Sensitivity of medial axis to small shape variations.

Distance transform and MAT

The simple algorithm given below

- inputs a binary image $u(r,c)$;
- computes its **distance transform** $u_{DT}(r,c)$ and its **medial axis** $u_{MA}(r,c)$.

Comments:

- $u_{DT}(r,c)$ is a greyscale image coding the **distance** from object pixel (r,c) to the nearest background point.
- $u_{MA}(r,c)$ is a binary image.
- We approximate the Euclidean distance is by the city-block distance.
- In practice, finer approximations are used, but principle of operation is the same.

Algorithm 2: Simple discrete DT and MAT

1. Select 4-connectivity for object pixels. Denote by $\Delta(r,c;i,j)$ the distance between (r,c) and (i,j). Initalise $u_0(r,c) = u(r,c)$.
2. Compute DT recursively for $k = 1, 2, \ldots$ as

$$u_k(r,c) = u_0(r,c) + \min_{i,j} \{u_{k-1}(i,j); (i,j) : \Delta(r,c;i,j) \leq 1)\} \tag{1}$$

   Iterate (1). Stop when no more changes occur: $u_k(r,c) = u_{k-1}(r,c)$ for all r, c. Set $u_{DT}(r,c) = u_k(r,c)$.

3. Compute MAT $u_{MA}(r,c)$ as the set of points

$$\{(r,c) : u_{DT}(r,c) \geq u_{DT}(i,j); \Delta(r,c;i,j) \leq 1\}$$

- DT is finished when k equals half the **maximum thickness** of the object.
- The MAT procedure can be **reversed** to restore the object from its skeleton if the **distance information** is preserved.

Example of computing DT and MAT:

$$\begin{array}{c} \begin{array}{ccccccc} 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1 \end{array} \\ u_0(r,c) \end{array} \rightarrow \begin{array}{c} \begin{array}{ccccccc} 1&1&1&1&1&1&1\\ 1&2&2&2&2&2&1\\ 1&2&2&2&2&2&1\\ 1&2&2&2&2&2&1\\ 1&1&1&1&1&1&1 \end{array} \\ u_1(r,c) \end{array} \rightarrow \begin{array}{c} \begin{array}{ccccccc} 1&1&1&1&1&1&1\\ 1&2&2&2&2&2&1\\ 1&2&3&3&3&2&1\\ 1&2&2&2&2&2&1\\ 1&1&1&1&1&1&1 \end{array} \\ u_2(r,c) \end{array}$$

$$\begin{array}{c} \begin{array}{ccccccc} 1&1&1&1&1&1&1\\ 1&2&2&2&2&2&1\\ 1&2&3&3&3&2&1\\ 1&2&2&2&2&2&1\\ 1&1&1&1&1&1&1 \end{array} \\ u_{DT} = u_3 = u_2 \end{array} \rightarrow \begin{array}{c} \begin{array}{ccccccc} 1&&&&&&1\\ &2&&&&2&\\ &&3&3&3&&\\ &2&&&&2&\\ 1&&&&&&1 \end{array} \\ \text{maxima of } u_{DT} \end{array} \rightarrow \begin{array}{c} \begin{array}{ccccccc} \bullet&&&&&&\bullet\\ &\bullet&&&&\bullet&\\ &&\bullet&\bullet&\bullet&&\\ &\bullet&&&&\bullet&\\ \bullet&&&&&&\bullet \end{array} \\ \text{medial axis } u_{MA} \end{array}$$

Examples of distance transform

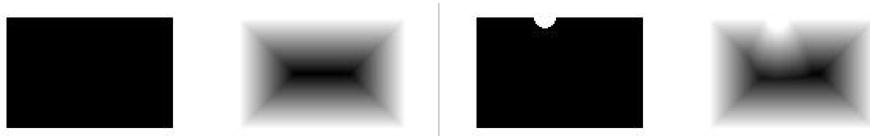


image DT distorted image DT

image DT

In DT images, dark points are large distances from background.

Thinning and skeleton

Thinning is a special kind of iterative shrinking that **preserves topology**. The result of thinning is the **skeleton** of the object.

Definition: S is **thinned down** to a skeleton by successively deleting its border pixels, provided that:

1. no such deletion disconnects S;
2. the endpoints are not deleted.

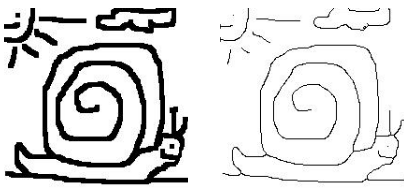


image skeleton

An example of thinning.

A thinning algorithm for 8-connected skeleton

Consider a point P_1 and its 3×3 neighbourhood.

P_3	P_2	P_9
P_4	P_1	P_8
P_5	P_6	P_7

Labelling point P_1 and its neighbours.

Introduce

- $Z0(P_1)$: the number of **zero to nonzero transitions** in the sequence $\{P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2\}$
  - $Z0(P_1)$ is calculated by **going around** the pixel P_1.
  - This counting operation is also often used to analyse **thinned images**, for example, to find line endpoints.
- $NZ(P_1)$: the number of nonzero neighbours of P_1

Algorithm 3: Thinning

1. Allocate two images, $u_1(r,c)$ and $u_2(r,c)$, of the same size as the input binary image $u_0(r,c)$. Initialise $u_2 = u_1 = u_0$.

2. Scan u_1, delete points in u_2.

3. In each current point P_1 compute $Z0(P_1)$, $NZ(P_1)$, $Z0(P_2)$ and $Z0(P_4)$.

4. Delete P_1 if the following conditions are **simultaneously satisfied**:

$$\begin{cases} 2 \le NZ(P_1) \le 6 \\ Z0(P_1) = 1 \\ P_2 \cdot P_4 \cdot P_8 = 0 \quad \text{OR} \quad Z0(P_2) \neq 1 \\ P_2 \cdot P_4 \cdot P_6 = 0 \quad \text{OR} \quad Z0(P_4) \neq 1 \end{cases}$$

5. When scanning of u_1 is finished, stop if no points were deleted. Otherwise, copy u_2 onto u_1 and go to step 2.

1	1	0
1	P_1	1
0	0	0

(a)

0	0	0
1	P_1	0
0	0	0

(b)

1	0	1
0	P_1	0
1	1	1

(c)

Examples where $P_1 = 1$ is **not** *deletable: (a) deleting P_1 splits region; (b) deleting P_1 shortens line ends; (c) $2 \le NZ(P_1) \le 6$, but P_1 is not deletable.*

Comments to the thinning algorithm:

- To compute $Z0(P_2)$ and $Z0(P_4)$, at each point we examine pixels from a 4×4 **neighbourhood**.

- The neighbourhood is formed and used in an **asymmetric way**: for example, $Z0(P_2)$ and $Z0(P_4)$ are computed, while $Z0(P_6)$ and $Z0(P_8)$ are not.

- This asymmetry allows the algorithm to obtain 1-pixel wide skeletons for lines of **even width**.

  ⇒ The result may be slightly (0.5 pixel) shifted wrt the 'true' skeleton.

Properties of skeleton

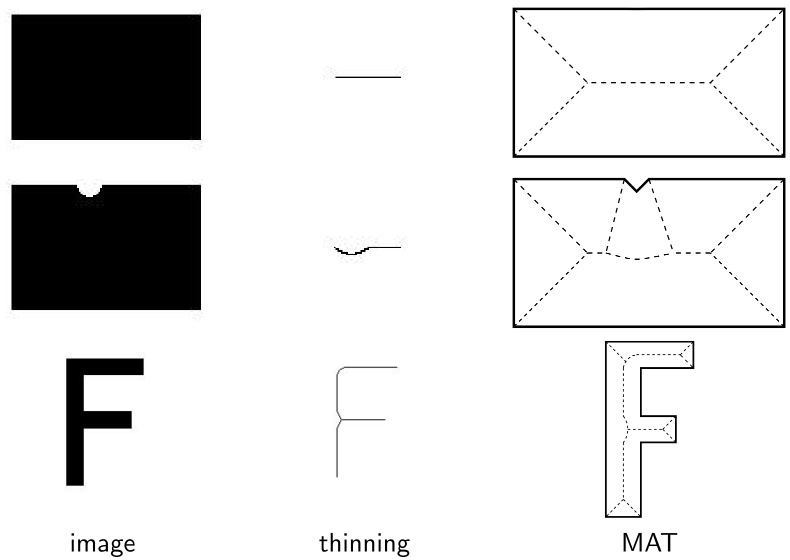
- Skeleton is, by definition, a **connected** set.

- Like medial axis, skeleton is the sum of **branches**.
  - Skeleton follows shape of object comprised of **elongated parts**.
  - In other cases, relation between object and skeleton may not be that obvious.

- Skeleton is **usually similar** to medial axis.

  ⇒ Often, no distinction between skeleton and MA is made. Thinning and MAT are viewed as alternative ways of obtaining a **skeletal representation** of an object. Then the medial axis is also called 'skeleton'.

- Skeleton **may significantly differ** from MA. (Compare the skeleton and the medial axis of a rectangle.)



Comparison of thinning and MAT.

Summary of skeletal representation

Advantages of skeletal representation of shape:

- Compresses data.
- Reflects structure of shape.
- Rotation-invariant. (In discrete case, only approximately.)
- Regenerative: Original shape can be restored from skeleton and distance data.

Drawbacks:

- Mainly useful for shapes comprised of elongated parts.
- MAT is sensitive to noise and distortions.