Binary Image Analysis

- Segmentation produces homogenous regions
  - each region has uniform gray-level
  - each region is a binary image (*0*: background, *1*: object *or* the reverse)
  - more intensity values for overlapping regions
- Binary images are easier to process and analyze than gray level images

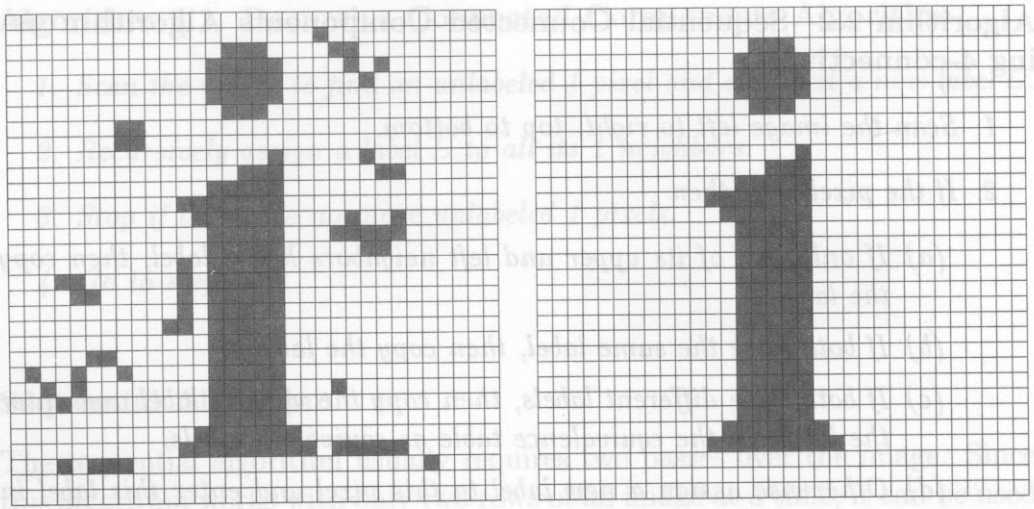
Binary Image Analysis Tasks

- Noise suppression
- Run-length encoding
- Component labeling
- Contour extraction
- Medial axis computation
- Thinning
- Filtering (morphological operations)
- Feature extraction (size, orientation etc.)

Noise suppression

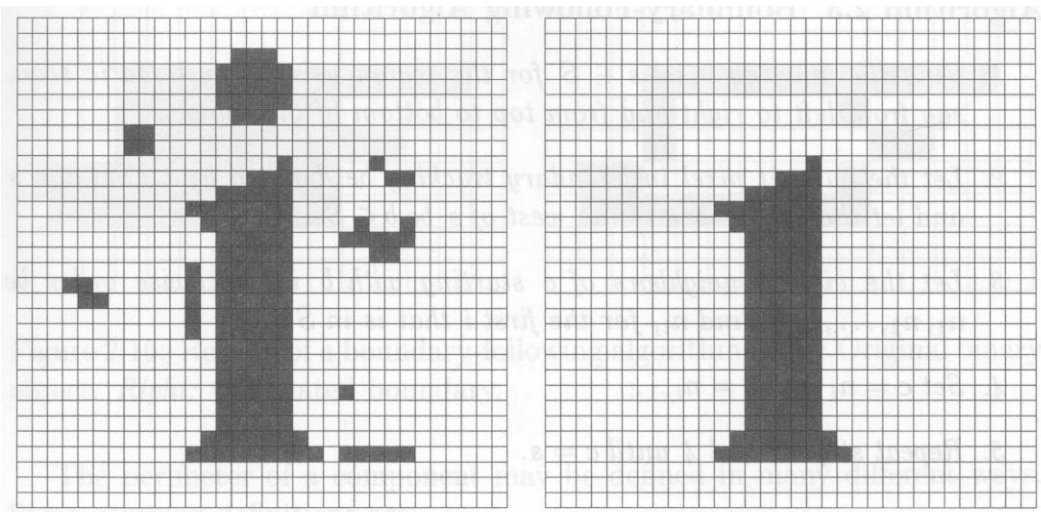
- Small regions are not useful information
  - apply "size filter" to remove such regions
  - all regions below *T* pixels in size are removed by changing the value of their pixels to *0* (background)
  - it is generally difficult to find a good value of *T*
  - if *T* is small, some noise will remain
  - if *T* is large, useful information will be lost

original noisy image



filtered Image T=10

original noisy image



filtered image T=25

too high!

Run-Length encoding

Binary image:

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

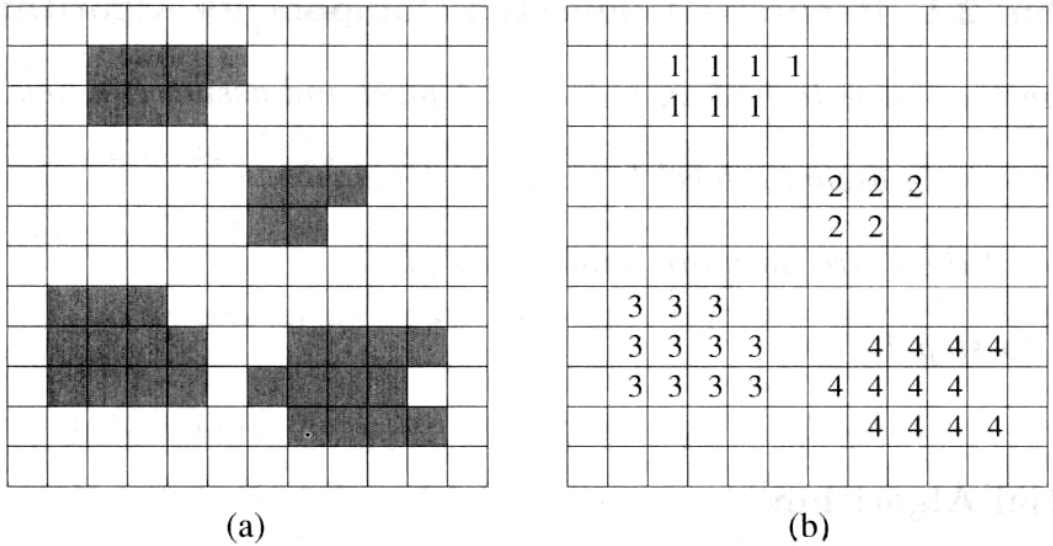
Start and length of 1 runs: $(1,3)\ (7,2)\ (12,4)\ (17,2)\ (20,3)$
$(5,13)\ (19,4)$
$(1,3)\ (17,6)$

Length of 1 and 0 runs: $3,3,2,3,4,1,2,1,3$
$0,4,13,1,4$
$3,13,6$

- Compact representation of a binary image
- Find the lengths of "*runs*" of *1* pixels sequences

Component Labeling

binary image



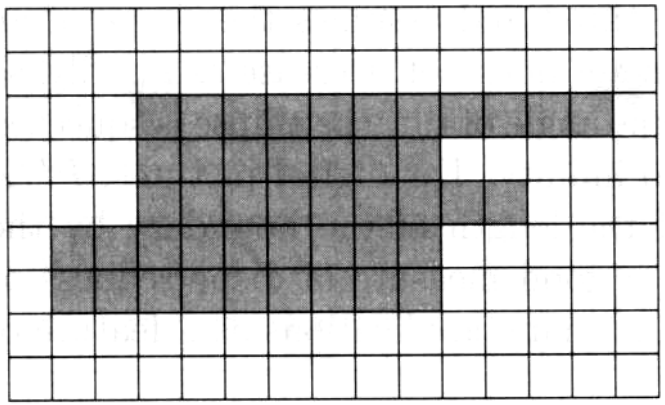
labeled connected components

- Assign different labels to pixels belonging to different regions (components)
  - connected components
  - not necessary for images with one region

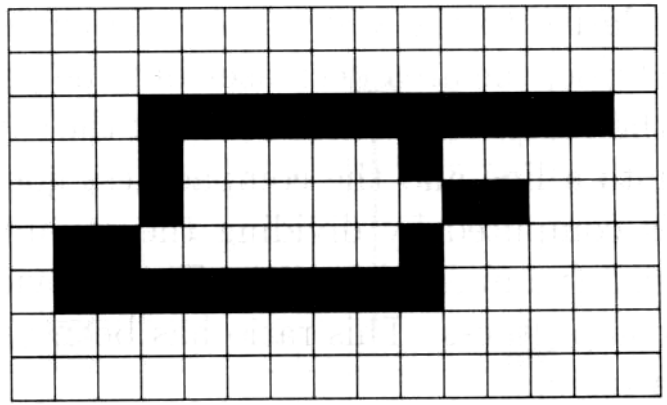
Sequential Algorithm

1. Scan the image from left to right, top to bottom; if the pixel is *1* then
   a) if only one of the upper *or* left pixels has a label, copy this label to current pixel
   b) if both have the same label, copy this label
   c) if they have different labels, copy one label and mark these two labels as equivalent
   d) if there are no labeled neighbors, assign it a new label
2. Scan the labeled image and replace all equivalent labels with a common label
3. If there are no neighbors, go to *1*

Contour Extraction



binary region



boundary

- Find all 8-connected pixels of a region that are adjacent to the background
  - select a starting pixel and track the boundary until it comes back with the starting pixel

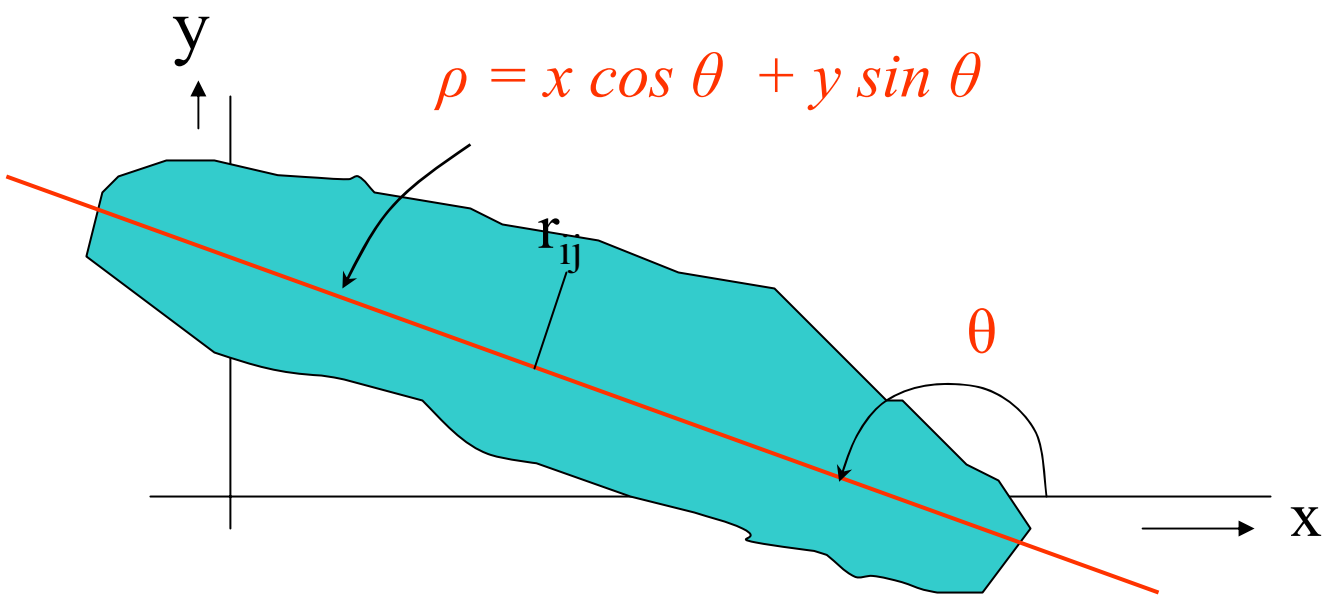
Boundary Following

- Scan the image from left to right and from top to bottom until an *1* pixel is found
  1) stop if this is the initial pixel
  2) if it is *1*, add it to the boundary
  3) go to a *0* 4-neighbor on its left
  4) check the 8-neighbors of the current pixel and go to the first *1* pixel found in clockwise order
  5) go to step *2*

Area – Center

- Binary (or gray) region *B[i,j]*
  - *B[i,j] = 1* if (i,j) in the region, *0* otherwise
  - Area: $A = \sum_{i=1}^{N}\sum_{j=1}^{M} B[i,j]$
  - Center of gravity: $x_0 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} jB[i,j]}{A}$ $y_0 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} iB[i,j]}{A}$

Orientation



- Angle with the horizontal direction
- Find angle θ minimizing $E = \sum_{i=1}^{N}\sum_{j=1}^{M} r_{ij}^2 B[i,j]$

Computing Orientation

- $E = a \sin^2\theta - b \sin\theta \cos\theta + c \cos^2\theta$, where

$$a = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij}^2 B[i,j]$$

$$b = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} y_{ij} B[i,j]$$

$$c = \sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}^2 B[i,j]$$

Computation Orientation (cont.)

- From which we get

$$E = \tfrac{1}{2}(a+c) - \tfrac{1}{2}(a-c)\cos 2\theta - \tfrac{1}{2}b\sin 2\theta$$

- Differentiating with respect to θ and setting the result to zero
  - $\tan 2\theta = b/(a\text{-}c)$ unless $b = 0$ and $a = c$
- Consequently

$$\sin 2\theta = \pm\frac{b}{\sqrt{b^2 + (a-c)^2}}, \cos 2\theta = \pm\frac{a-c}{\sqrt{b^2 + (a-c)^2}}$$

- The solution with the $+$ minimizes E
- The solution with the $-$ maximizes E

Computation Orientation (cont.)

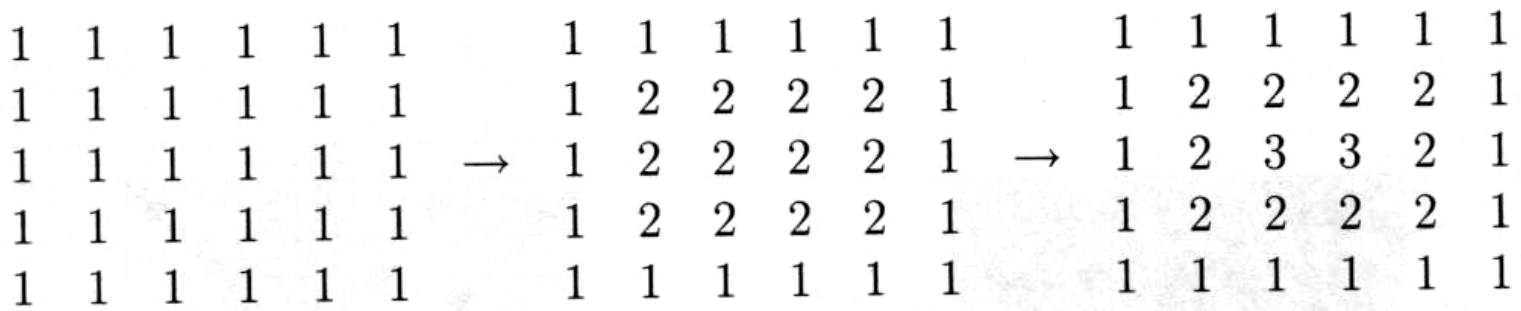
- Compute E_{min}, E_{max} minimum and maximum of the least second moment E

- The ratio $e = E_{max}/E_{min}$ represents *roundness*
  - $e \rightarrow 0$ for lines
  - $e \rightarrow 1$ for circles

Distance Transform

- Compute the distance of each pixel (i,j) from the background S
  - at iteration n compute $F^n[i,j]$:
  - $F^0[i,j] = f[i,j]$ (initial values)
  - $F^n[i,j] = F^0[i,j] + min(F^{n-1}[u,v])$
  - (u,v) are the 4-neighbor pixels of (i,j) that is pixels with $D([i,j],[u,v]) = 1$
  - repeat until no distances changes

Example of Distance Transform



- Distance transform of an image after the first and second iterations
  - on the first iteration, all pixels that are not adjacent to *S* are changed to 2
  - on succeeding iterations only pixels further away from *S* change

Skeleton (Medial Axis)

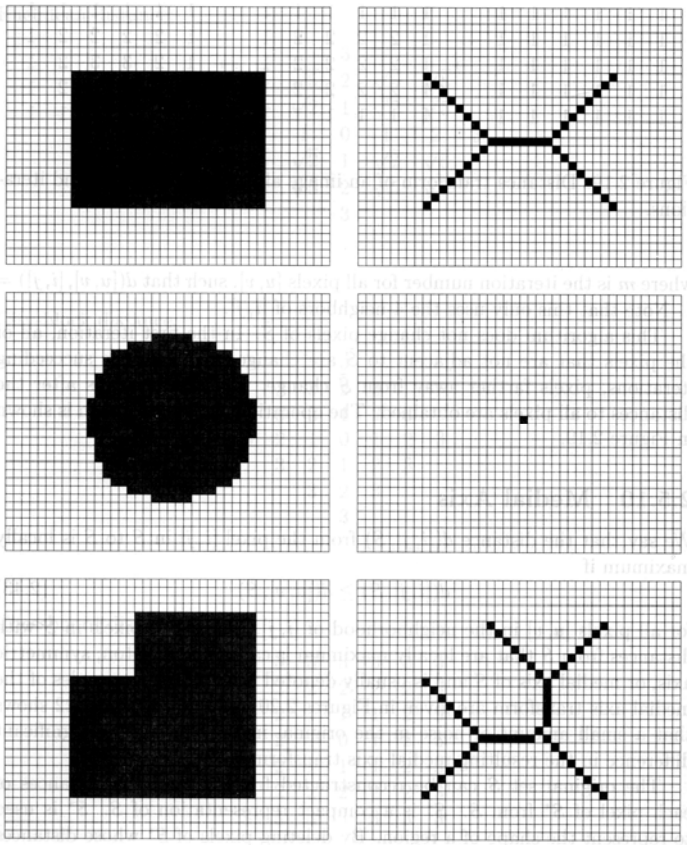
- Set of pixels with locally maximum distance from the background *S*
- Take the distance transform and keep *(i,j)* Keep a point *(i,j)* if it is the max in its 4-neighborhood:

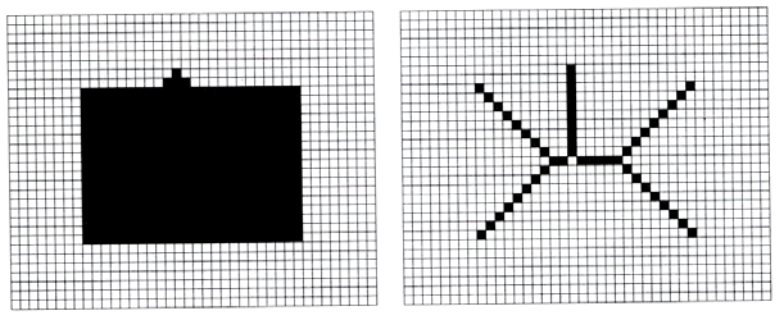
  D([i,j],S): locally maximum that is

  D([i,j],S) >= D([u,v],S) where

  (u,v) are the 4-neighbors of *(i,j)*
- The region can be reconstructed from its skeleton
  - take all pixels within distance *D(i,j)* from each pixel *(i,j)* of the skeleton

Examples of Medial Axis Transform

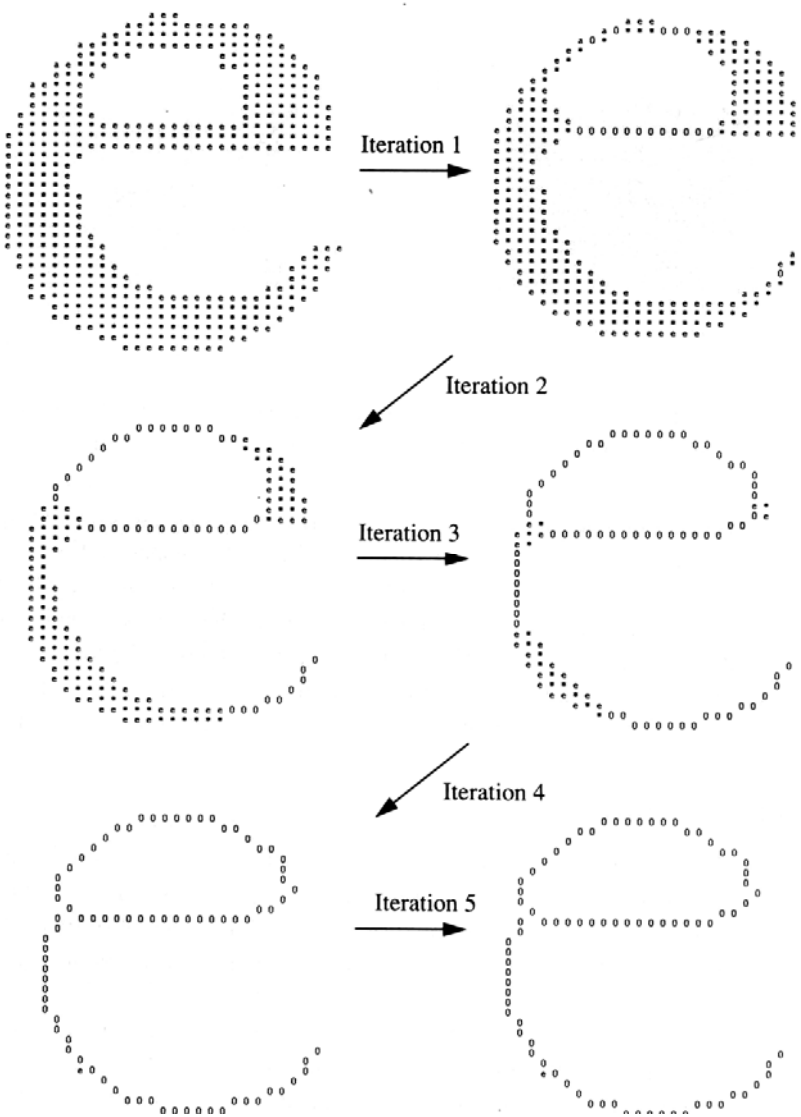




the medial axis transform
is very sensitive to noise

Thinning

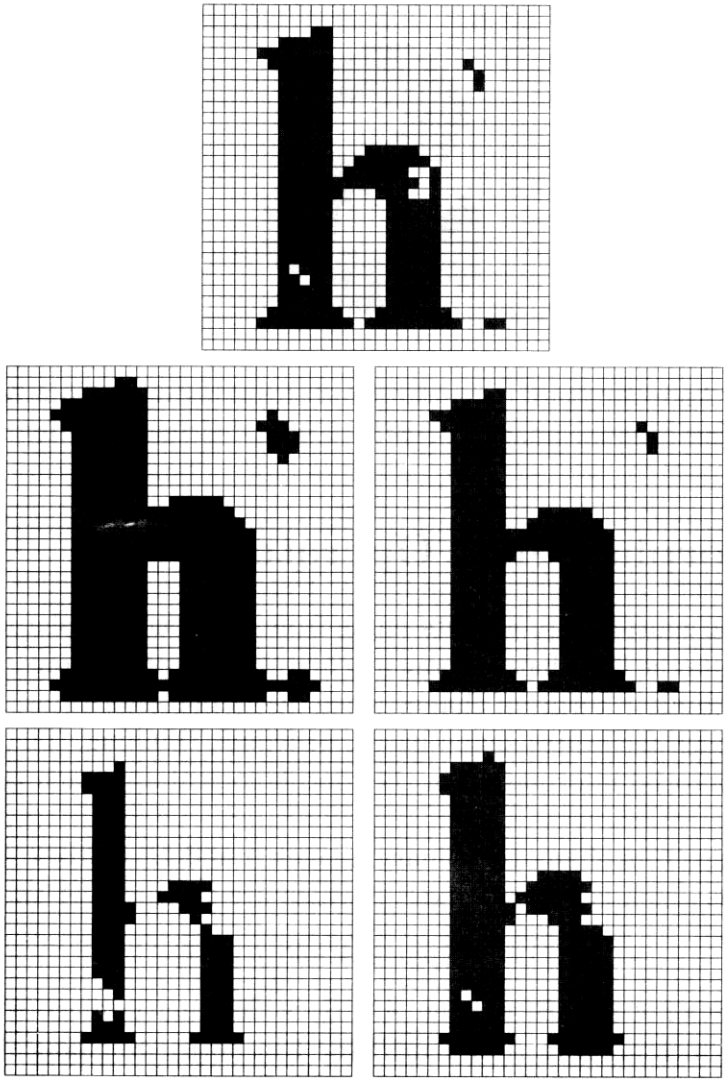
- Binary regions are reduced to their center lines
  - also called *skeletons* or core lines
  - suitable for elongated shapes and OCR
  - each region (e.g., character) is transformed to a line representation
  - further analysis and recognition is facilitated
- Iteratively check the 8-neighbors of each pixel
  - delete pixels connected with *S* unless the 8-neighbor relationship with the remaining pixels is destroyed and except pixels at the end of a line
  - until no pixels change



no pixels change
after 5 iterations

Filtering Operations

- Expansion: some background pixels adjacent to the region are changed from *0* to *1*
  - the region is expanded
  - fills gaps, the region is smoothed
- Shrinking: some pixels are changed from *1* to *0*
  - the region is shrinked
  - removes noise, thinning
- A combination of Expansion with Shrinking may achieve better smoothing



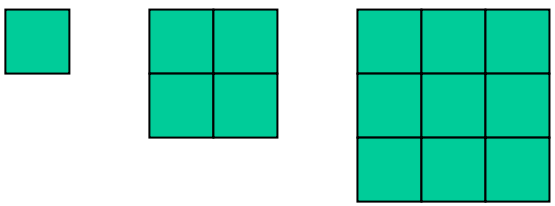
noisy image

expanding followed by shrinking
filled holes but did not eliminate noise

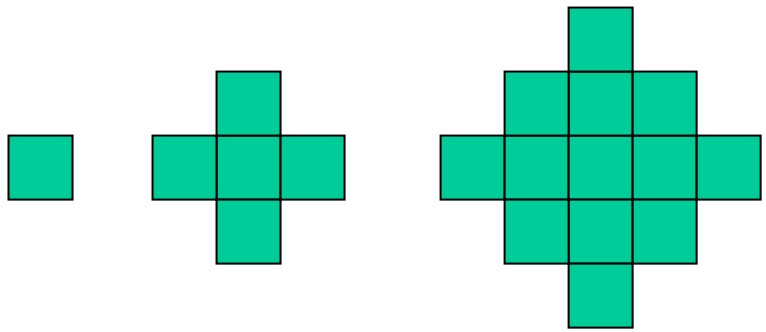
shrinking followed by expanding
eliminated noise but did not fill the holes

Morphological Filtering

- Image filtering using 4 filtering operations
  - two basic: Dilation, Erosion plus
  - two derived: Opening, Closing
  - and a Structuring Element (*SE*) whose size and shape may vary



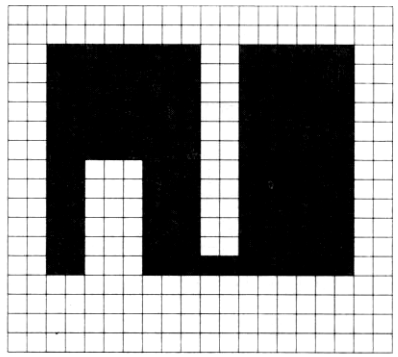
SE: rectangle with R=1,2,3



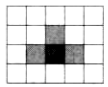
SE: circle with R=1,2,3

Erosion - Dilation

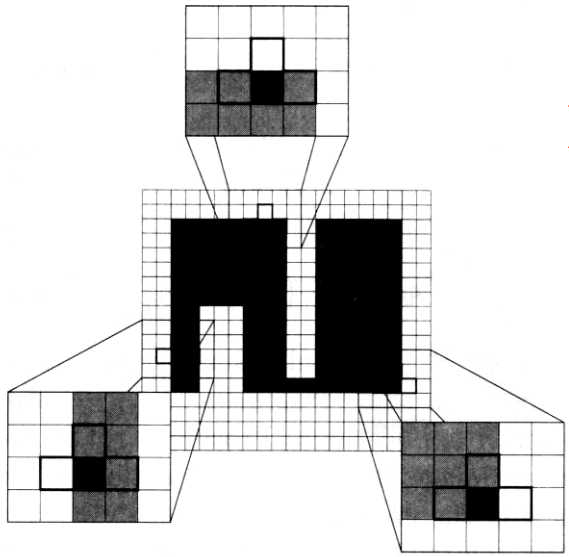
- *Erosion*: apply the *SE* on every pixel *(i,j)* of the image *f*
  - *(i,j)* is the center of the *SE*
  - if the whole SE is included in the region then, *f[i,j] = 1*
  - otherwise, *f[i,j] = 0*
  - erosion shrinks the object
- *Dilation*: if at least one pixel of the SE is inside the region *f[i,j] = 1*
  - dilation expands the region



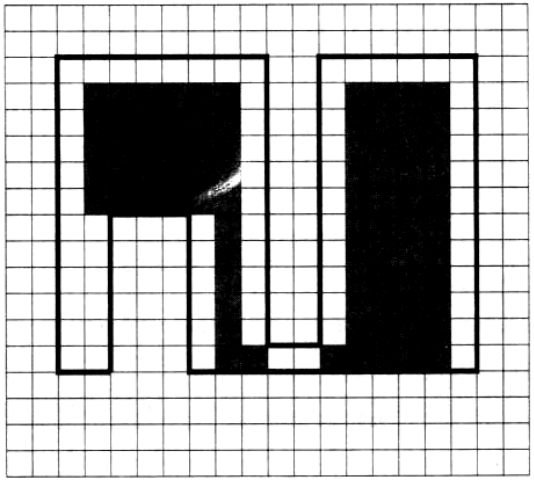
original image



Structuring Element (*SE*)

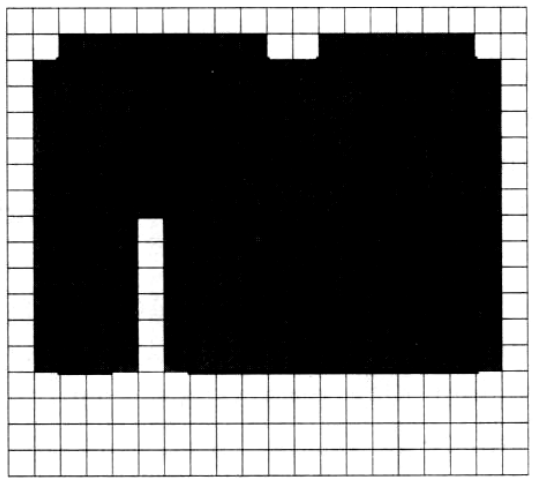


Erosion with the *SE*
at various positions
of the original image



the erosion of the original image with the *SE*

the bold line shows the border of the original image



the dilation of the original image with the *SE*

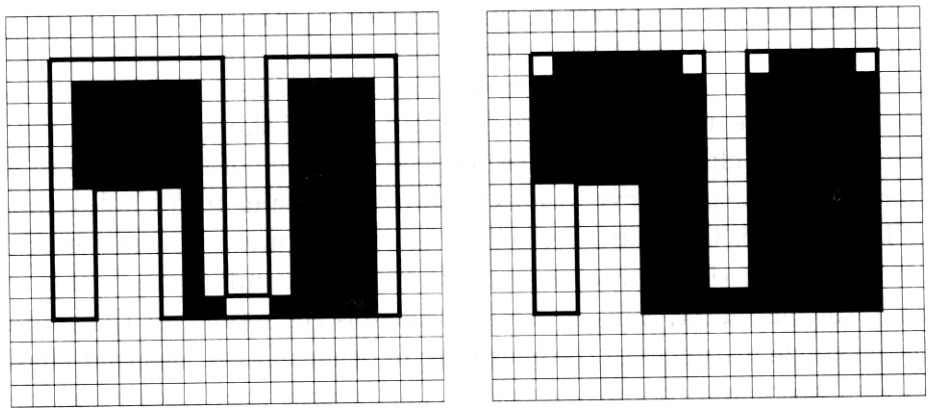
the bold line shows the border of the original image

Opening - Closing

- *Opening*: erosion followed by dilation with the same *SE*
  - filters out "positive" detail, shirks the region
- *Closing*: dilation followed by erosion with the same *SE*
  - smoothes by expansion, fills gaps and holes smaller than the *SE*

opening

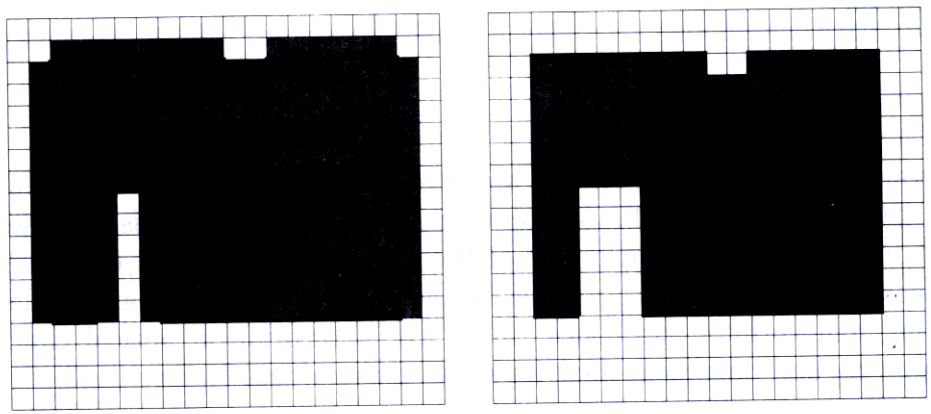
initial erosion



succeeding dilation

closing

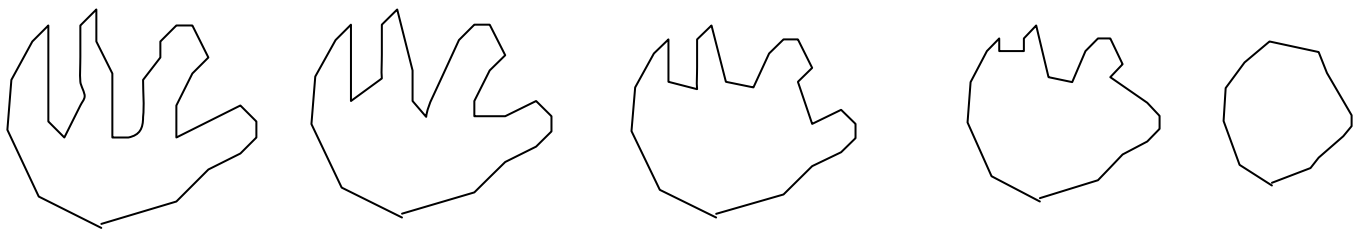
initial dilation



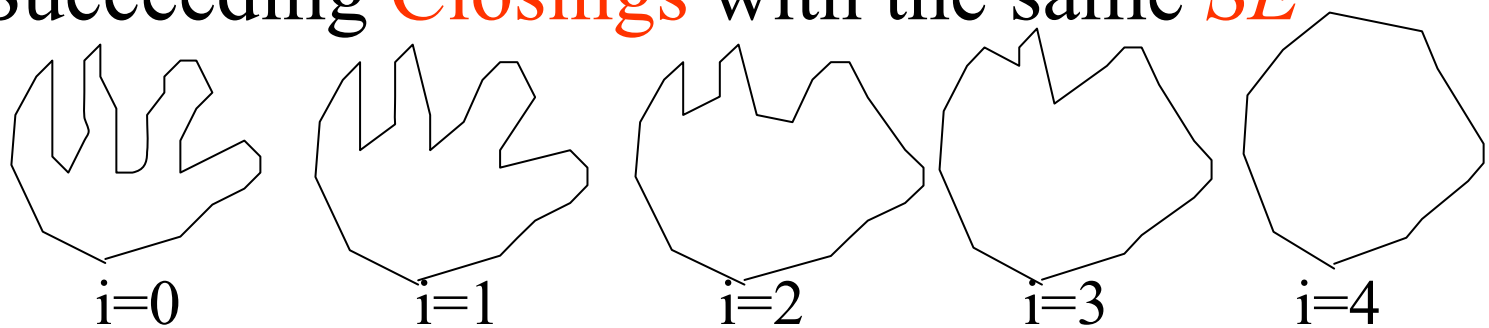
succeeding erosion

Pattern Spectrum

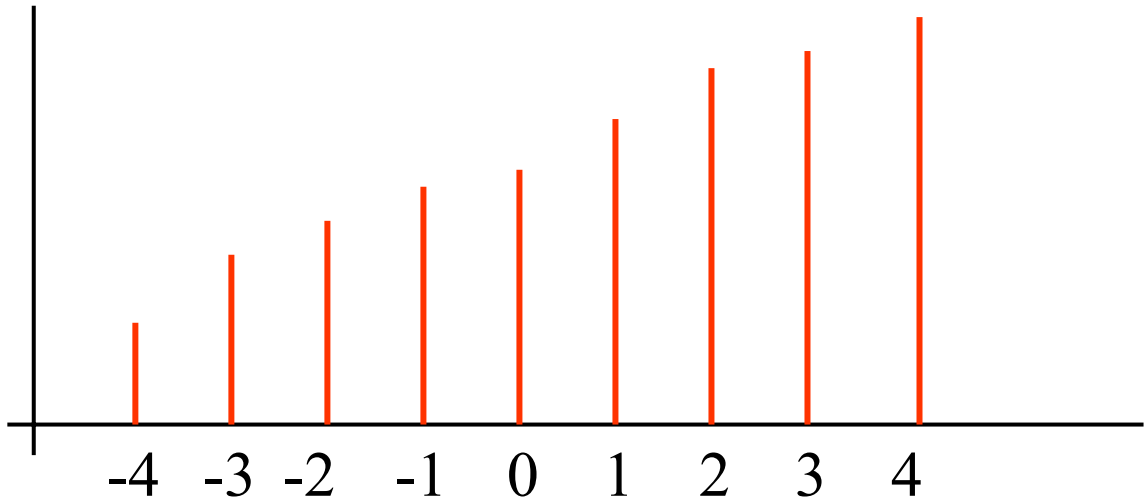
- Succeeding Openings with the same *SE*



- Succeeding Closings with the same *SE*



Example of Pattern Spectrum



- Size vector $p=(a_{-n}, a_{-n+1}, a_{-n+2}, \ldots, a_{-1}, a_1, a_2, a_3, \ldots, a_n)$

opening original closing

- Distance between shapes A,B: $D(A,B) = \left(\sum_{i=-n}^{n} (a_i - b_i)^2 \right)^{1/2}$