EEE-6512: Image Processing and Computer Vision

September 29, 2017
Lecture #6: Binary Image Processing

Damon L. Woodard, Ph.D.
Dept. of Electrical and Computer Engineering
dwoodard@ece.ufl.edu

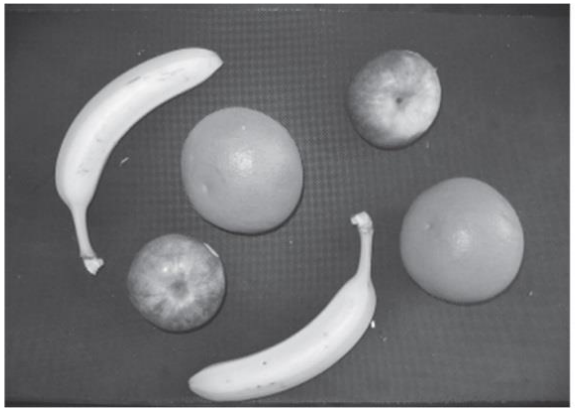
Chapter Outline

- Morphological Operations
- Labeling Regions
- Computing Distance in a Digital Image
- Region Properties
- Skeletonization
- Boundary Representations

Morphological Operations

- **Mathematical morphology:** an entire branch of mathematics that has been developed to process images by considering the shape of the pixel regions.

Figure 4.1 Left: A grayscale image of several types of fruit on a dark conveyor belt. Right: A binary image resulting from thresholding. The white pixels are ON and indicate the foreground, while the black pixels are OFF and indicate the background.



Stan Birchfield

Mathematical Morphology (cont.)

- The language of mathematical morphology is **set theory**.
- It is a mathematical tool for investigating **geometric structure** in an image.
- It is an approach for processing a digital image based on its **shape**.

Mathematical Morphology Goals

- To simplify image data, preserve essential shape characteristics, and eliminate noise
- Allows the underlying shape to be identified and reconstructed optimally from its noisy distorted forms.

Mathematical Morphology Uses

- Image pre-processing (noise filtering, shape simplification)
- Enhancing object structures (skeletonization, thinning, convex hull, object marking)
- Segmentation of the object from background
- Quantitative descriptors of objects (area, perimeter, projection, Euler-Poincaré characteristics)

Mathematical Morphology Main Idea

- Morphological operators often take a binary image and a structuring element as input and combine them using a set operator (intersection, union, inclusion, complement).

- The structuring element is shifted over the image and at each pixel of the image its elements are compared with the set of the underlying pixels.

- If the two sets of elements match the condition defined by the set operator (e.g. if set of pixels in the structuring element is a subset of the underlying image pixels), the pixel underneath the origin of the structuring element is set to a pre-defined value (0 or 1 for binary images).

- A morphological operator is therefore defined by its **structuring element** and the **applied set operator.**

Binary Image as a Set

- A **binary image** is an array of values such that $I(x, y)$ returns 1 or 0 for each pixel location (x, y).

- Two fundamental set operators are *union* and *intersection*:

$$\mathcal{A} \cup \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} \in \mathcal{A} \text{ or } \mathbf{z} \in \mathcal{B}\} \qquad \text{(union)}$$

$$\mathcal{A} \cap \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} \in \mathcal{A} \text{ and } \mathbf{z} \in \mathcal{B}\} \qquad \text{(intersection)}$$

Binary Image as a Set (cont'd)

- Additional operators:

$$\mathcal{A}_{\mathbf{b}} \equiv \{\mathbf{z} : \mathbf{z} = \mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}\} \quad \text{(translation)}$$

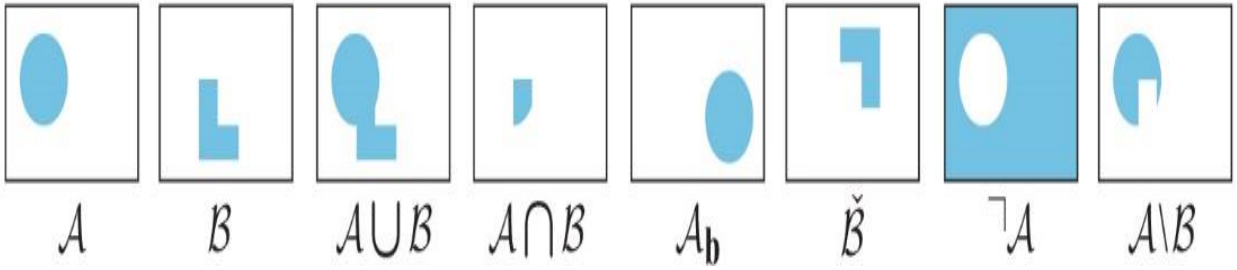
$$\check{\mathcal{B}} \equiv \{\mathbf{z} : \mathbf{z} = -\mathbf{b}, \mathbf{b} \in \mathcal{B}\} \quad \text{(reflection)}$$

$$\neg\mathcal{A} \equiv \{\mathbf{z} : \mathbf{z} \notin \mathcal{A}\} \quad \text{(complement)}$$

$$\mathcal{A} \setminus \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} \in \mathcal{A}, \mathbf{z} \notin \mathcal{B}\} = \mathbf{A} \bigcap \neg\mathcal{B} \quad \text{(difference)}$$

Set Operators

Figure 4.2 Set operators. The first two columns show two sets $\mathcal{A}$ and $\mathcal{B}$ in blue. Then, from left to right, shown are the union, intersection, shift of $\mathcal{A}$ by some amount **b** (not related to $\mathcal{B}$), reflection about the origin (assumed to be at the center), complement, and set difference.



Minkowski Addition and Subtraction

- The **Minkowski addition** of two sets *A* and *B* is defined as the set of points resulting from all possible vector additions of elements of the two sets:

$$\mathcal{A} \oplus \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} = \mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}$$

$$= \bigcup_{\mathbf{b} \in \mathcal{B}} \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathcal{A}\} = \bigcup_{\mathbf{b} \in \mathcal{B}} \mathcal{A}_{\mathbf{b}}$$

- **Minkowski subtraction** of two sets:

$$\mathcal{A} \ominus \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} - \mathbf{b} \in \mathcal{A}, \forall \mathbf{b} \in \mathcal{B}\}$$

$$= \bigcap_{\mathbf{b} \in \mathcal{B}} \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathcal{A}\} = \bigcap_{\mathbf{b} \in \mathcal{B}} \mathcal{A}_{\mathbf{b}}$$

Minkowski Addition and Subtraction

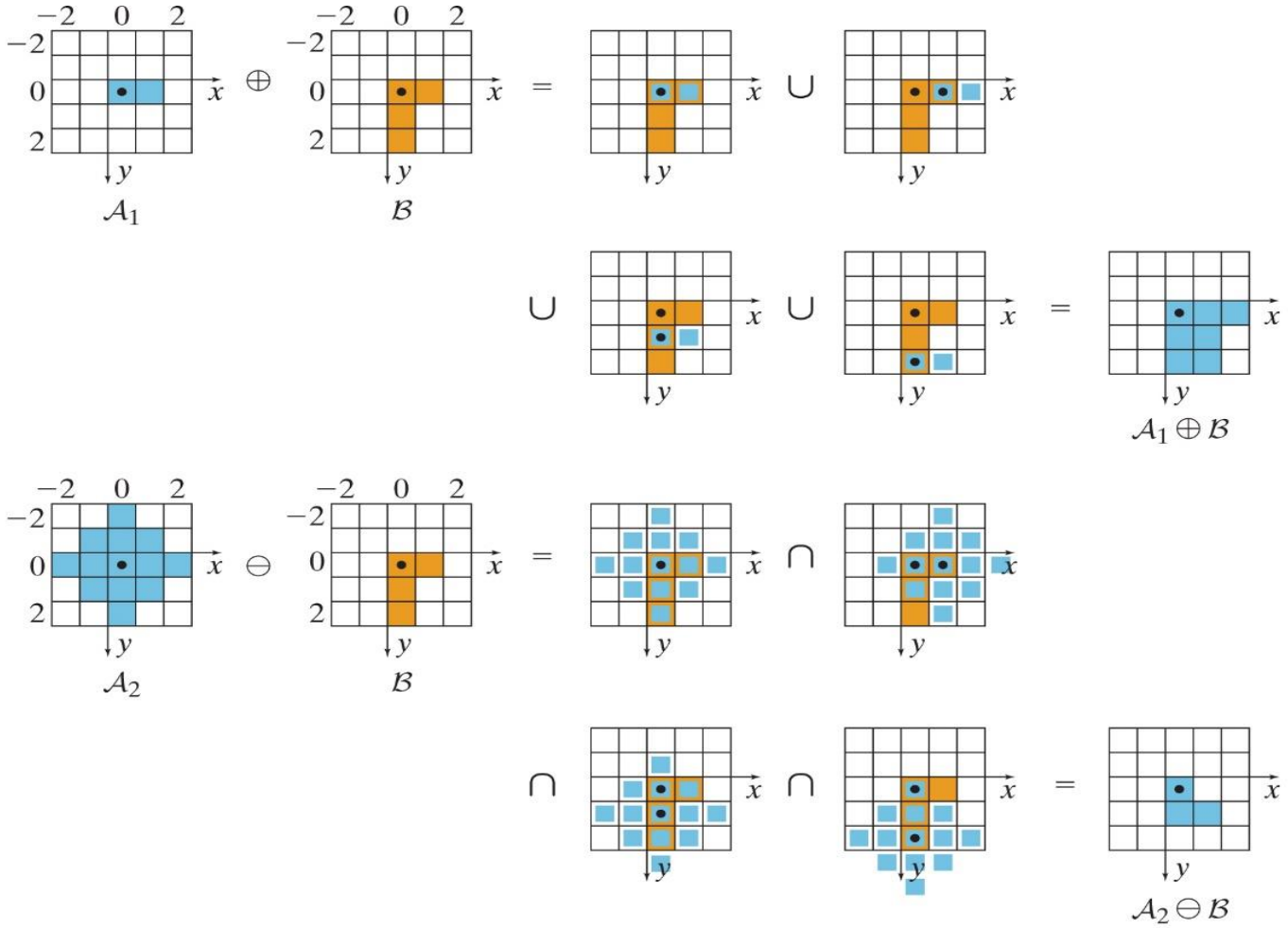


Figure 4.3 Minkowski addition and subtraction, from Equations (4.10) and (4.12) , respectively. For both operations, the first set ($\mathcal{A}_1$ or $\mathcal{A}_2$) is translated so that its origin is placed at each element of the second set ($\mathcal{B}$). The union of the blue cells then yields $\mathcal{A}_1 \oplus \mathcal{B}$, while their intersection yields $\mathcal{A}_2 \ominus \mathcal{B}$. In other words, each ON (blue) cell in $\mathcal{A}_1 \oplus \mathcal{B}$ is ON (blue) in *at least one* of the intermediate results, while each ON (blue) cell in $\mathcal{A}_2 \ominus \mathcal{B}$ is ON (blue) in *all* of the intermediate results. Colored cells are ON (value 1); white cells are OFF (value 0); and all pixels outside the 5 × 5 image are assumed to be OFF. The small black dots indicate the origins of the coordinate systems. (As explained later, this is the "center-in" approach.)

Properties of the Minkowski operators

$$\mathcal{A} \oplus \mathcal{B} = \mathcal{B} \oplus \mathcal{A} \qquad \text{(commutativity)}$$

$$\mathcal{A} \ominus \mathcal{B} = \neg\mathcal{B} \ominus \neg\mathcal{A} \qquad \text{(non-commutativity)}$$

$$(\mathcal{A} \oplus \mathcal{B}) \oplus \mathcal{C} = \mathcal{A} \oplus (\mathcal{B} \oplus \mathcal{C}) \qquad \text{(associativity, separability)}$$

$$(\mathcal{A} \ominus \mathcal{B}) \ominus \mathcal{C} = \mathcal{A} \ominus (\mathcal{B} \oplus \mathcal{C}) \qquad \text{(separability)}$$

$$(\mathcal{A} \oplus \mathcal{B}) \oplus \mathcal{C} = (\mathcal{A} \oplus \mathcal{C}) \oplus \mathcal{B} \qquad \text{(order does not matter)}$$

$$(\mathcal{A} \ominus \mathcal{B}) \ominus \mathcal{C} = (\mathcal{A} \ominus \mathcal{C}) \ominus \mathcal{B} \qquad \text{(order does not matter)}$$

$$\mathcal{A} \oplus \mathcal{B} \subseteq \mathcal{A} \text{ if} (0,0) \in \mathcal{B} \qquad \text{(extensivity)}$$

$$\mathcal{A} \ominus \mathcal{B} \subseteq \mathcal{A} \text{ if} (0,0) \in \mathcal{B} \qquad \text{(anti-extensivity)}$$

$$\mathcal{A}_1 \oplus \mathcal{B} \subseteq \mathcal{A}_2 \oplus \mathcal{B} \text{ if } \mathcal{A}_1 \subseteq \mathcal{A}_2 \qquad \text{(increasing)}$$

$$\mathcal{A}_1 \ominus \mathcal{B} \subseteq \mathcal{A}_2 \ominus \mathcal{B} \text{ if } \mathcal{A}_1 \subseteq \mathcal{A}_2 \qquad \text{(increasing)}$$

Properties of the Minkowski operators (cont'd)

$$\mathcal{A} \oplus (\mathcal{B} \cup \mathcal{C}) = (\mathcal{A} \oplus \mathcal{B}) \cup (\mathcal{A} \oplus \mathcal{C}) \qquad \text{(parallelism)}$$

$$\mathcal{A} \ominus (\mathcal{B} \cup \mathcal{C}) = (\mathcal{A} \ominus \mathcal{B}) \cap (\mathcal{A} \ominus \mathcal{C}) \qquad \text{(parallelism)}$$

$$(\mathcal{A} \cap \mathcal{B}) \ominus \mathcal{C} = (\mathcal{A} \ominus \mathcal{C}) \cap (\mathcal{B} \ominus \mathcal{C})$$

$$(\mathcal{A} \cup \mathcal{B}) \oplus \mathcal{C} = (\mathcal{A} \oplus \mathcal{C}) \cup (\mathcal{B} \oplus \mathcal{C})$$

$$\neg(\mathcal{A} \oplus \mathcal{B}) = \neg\mathcal{A} \ominus \mathcal{B} \qquad \text{(duality)}$$

$$\neg(\mathcal{A} \ominus \mathcal{B}) = \neg\mathcal{A} \oplus \mathcal{B} \qquad \text{(duality)}$$

Dilation and Erosion

- Minkowski addition grows (or "dilates") the region by increasing its size.
- Minkowski subtraction shrinks (or "erodes") the region by decreasing its size.

$$\mathcal{A} \oplus \mathcal{B} \equiv A \oplus \mathcal{B} = \{\mathbf{z} : \mathbf{z} = \mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\} \quad \text{(dilation)}$$

$$\mathcal{A} \check{\ominus} \mathcal{B} \equiv A \ominus \check{\mathcal{B}} = \{\mathbf{z} : \mathbf{z} + \mathbf{b} \in \mathcal{A}, \forall \mathbf{b} \in \mathcal{B}\} \quad \text{(erosion)}$$

- The most common use of dilation and erosion is to remove noise.

EEE-6512: Image Processing and Computer Vision

October 2&4, 2017
Lecture #6, parts II&III: Binary Image Processing
Walter O'Dell, Ph.D.
Dept. of Radiation Oncology
wodell@ufl.edu
odell.radonc.med.ufl.edu

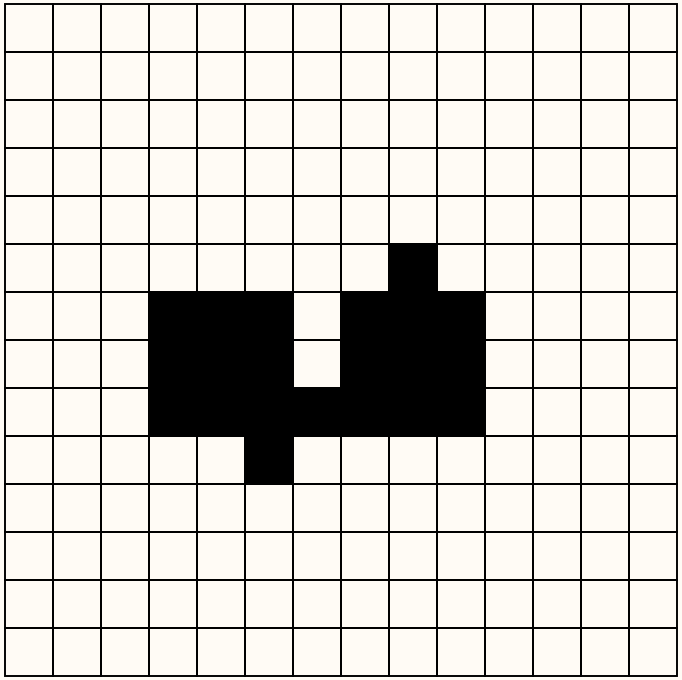
Dilation and Erosion

Dilations make objects thicker, can be used to fill in holes, multiple applications can result in image which looks different from input

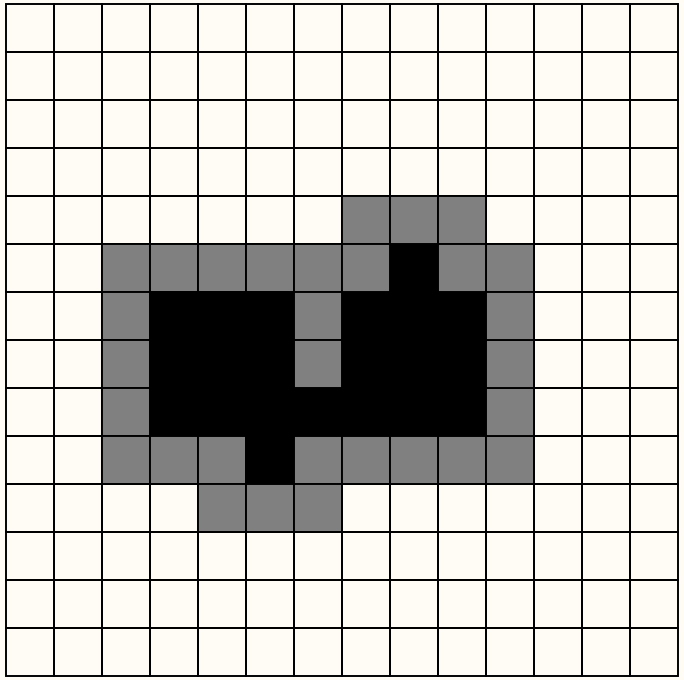
Erosions remove pixels from object boundaries, used to simplify structure of object. Objects with width one will disappear.

Example of Dilation

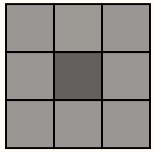
if any pixel in the structural element that is ON lands on an ON pixel of the image, then the output pixel is made ON



F



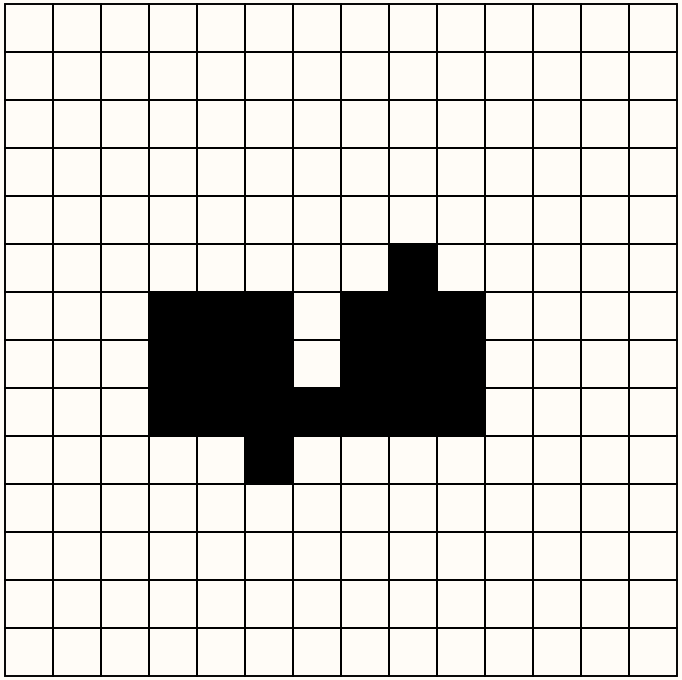
G



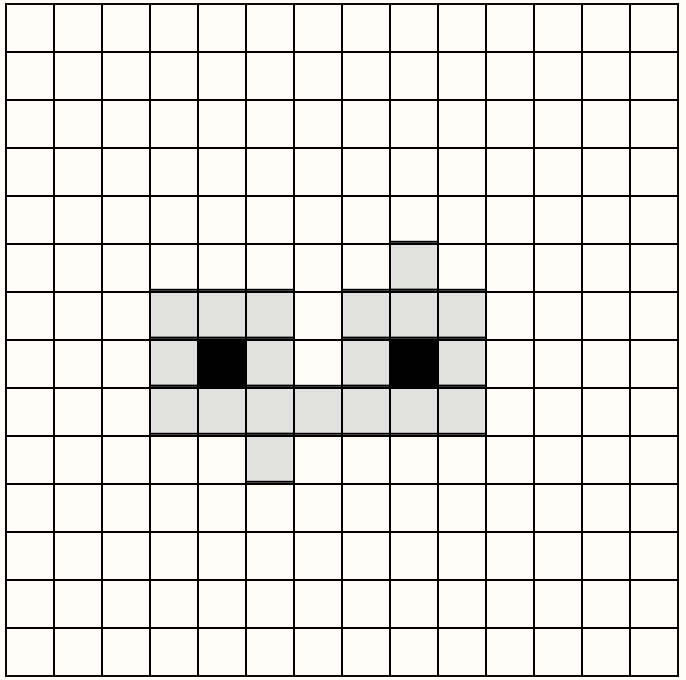
H, 3x3, origin at the center

Example of Erosion

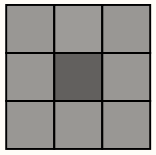
if any pixel in the structural element that is ON lands on an OFF pixel of the image, then the output pixel is made OFF



F



G



H, 3x3, origin at the center

Structuring element

- The shape, size, and orientation of the structuring element depend on application.
- A symmetrical one will enlarge or shrink the original set in all directions.
- A vertical one, will only expand or shrink the original set in the vertical direction.

Hit-Miss Operator

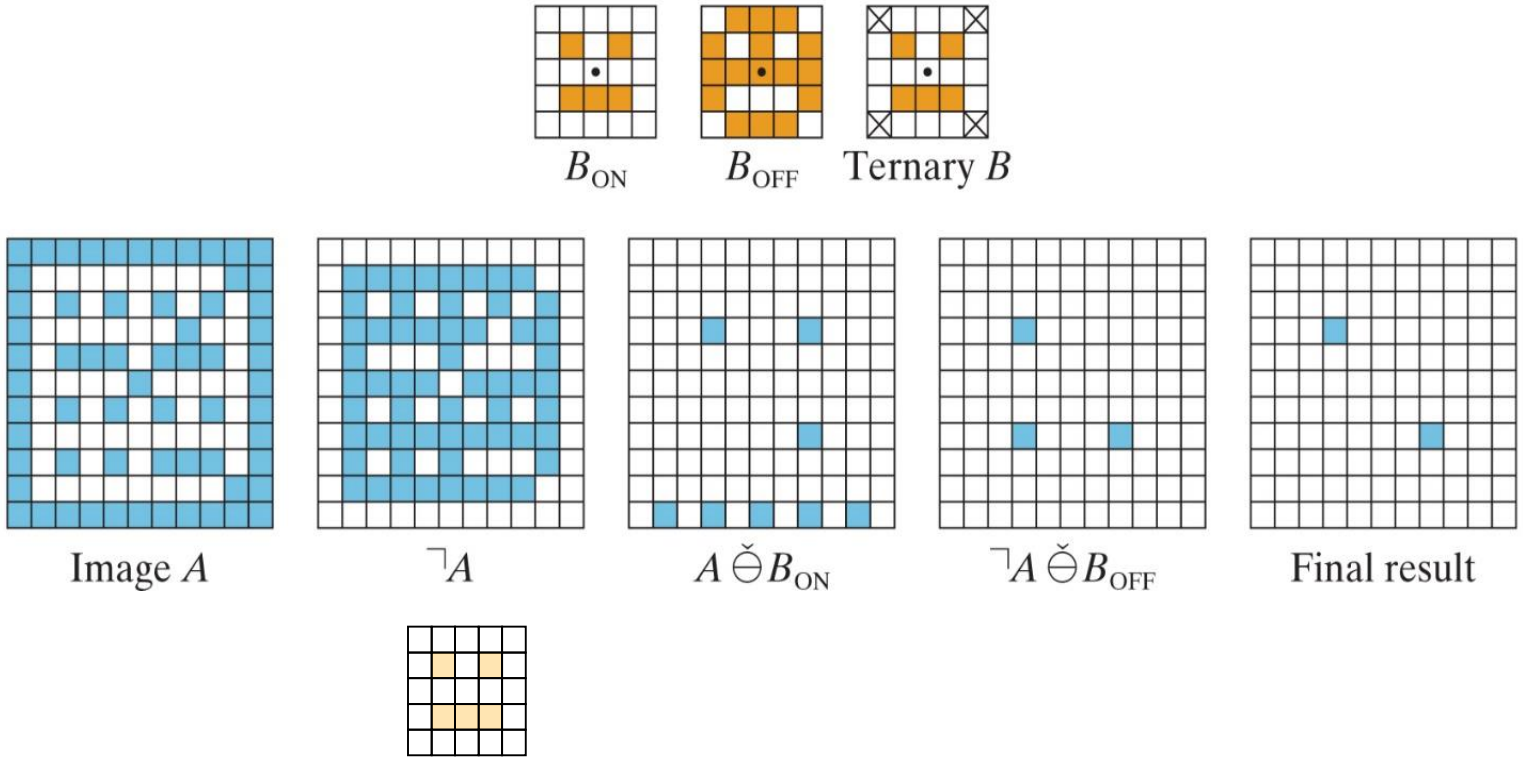
= pattern detector

- To detect the shape in the image, the **hit-miss operator** uses erosion to find all the places in the image where B_{ON} matches the foreground and B_{OFF} matches the background:

$$A \circledast (B_{ON}, B_{OFF}) \equiv (A \check{\ominus} B_{ON}) \cap (\neg A \check{\ominus} B_{OFF}) \qquad \text{(hit-miss operator)}$$

Hit or Miss Operator

Figure 4.12 Hit-miss operator. TOP: The foreground pattern B_{ON}, the background pattern B_{OFF}, and the ternary representation B with X indicating DONT-CARE. BOTTOM: From left-to-right: The image A within which to search for the pattern, the negation $\neg A$ of the image, the erosion of the image with the foreground pattern, the erosion of the negated image with the background pattern, and the final result, which shows two successful detections of the smiley face.



Hit-Miss Operator (cont'd)

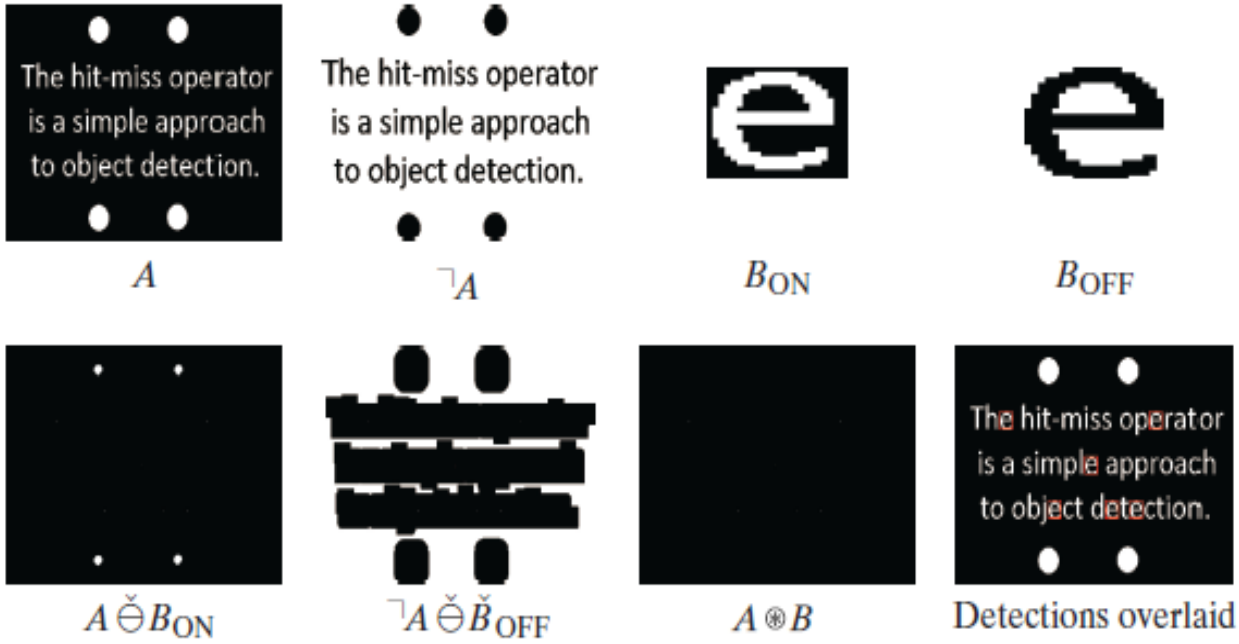


Figure 4.13 The hit-miss operator applied to a binary image, with white indicating ON and black indicating OFF. TOP: From left-to-right: The image, the inverted image, the foreground pattern, and the background pattern. BOTTOM: From left-to-right: The erosion of the image with the foreground pattern, the erosion of the inverted image with the background pattern, the result of hit-miss, and the outlines of the detections overlaid on the original image.

Thinning

- **Thinning** a binary image involves removing pixels from the foreground (setting pixels to OFF) while maintaining as much as possible the structure and connectivity of the foreground regions.

- Thinning takes an image and a ternary SE and removes all the points detected by the SE:

$$I \boxminus B \equiv I - (I \circledast B) = I \cap \neg(I \circledast B)$$

Thinning

The matching operation

Thinning by structuring element B :

$A \otimes B = A - (A * B)$.

Goal to remove on matching pixels from edge

Denote a sequence of structuring elements

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

Thinning by a sequence of structuring elements $\{B\}$:

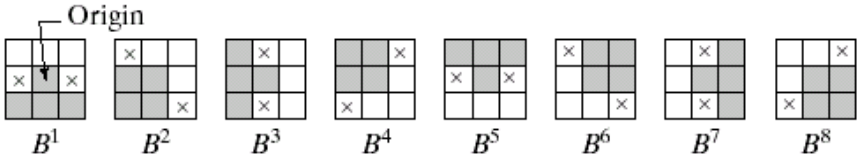
$A \otimes \{B\} = ((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$.

The process stops until no further changes occur.

Thinning (2)

Top top-rt right bot-rt bottom bot-left left top-left

top edge SE



Consider surrounding pixels to be OFF

Goal to remove pixels from the edge but symmetrically to get the center line.
1. Pass each SE over entire image once.

2. Do each SE in succession.
3. Repeat as needed.

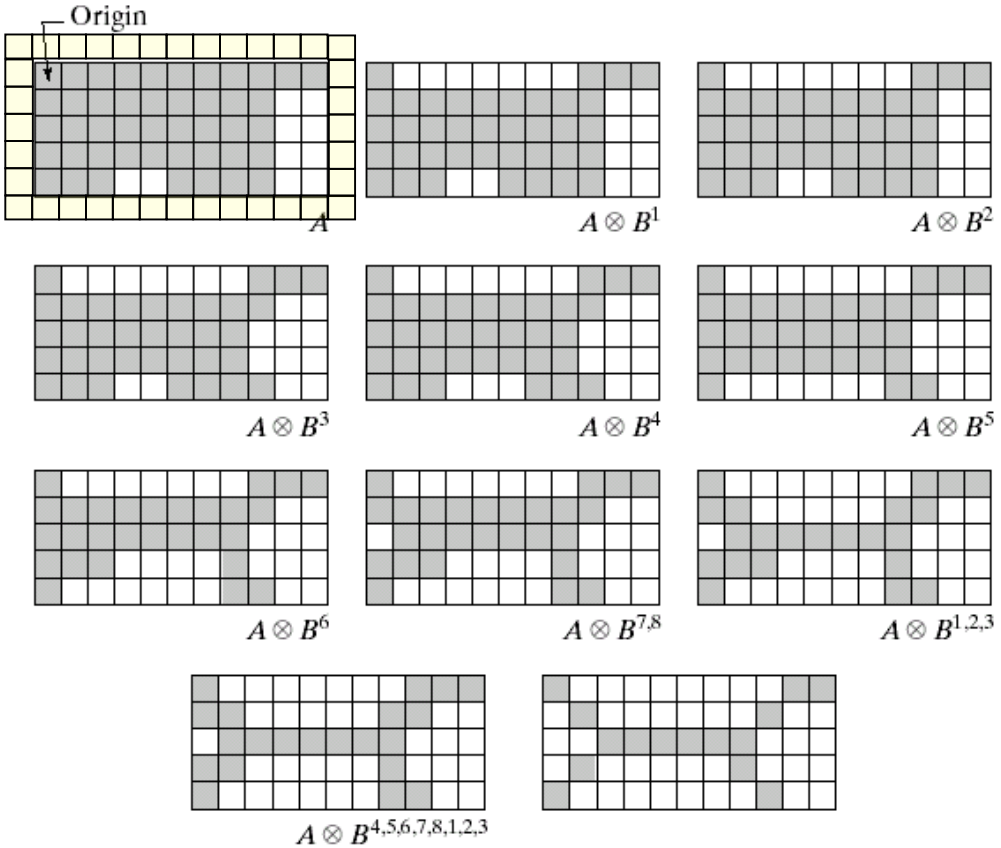
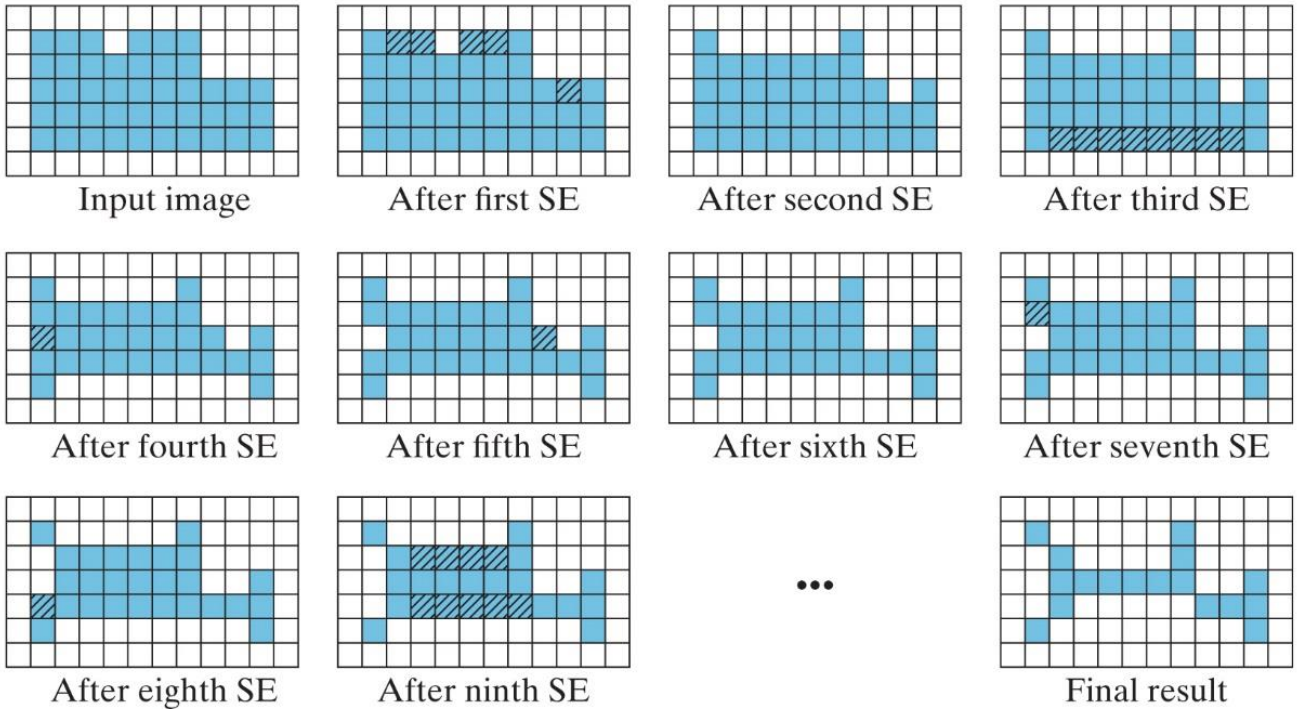


FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A. (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there were no changes for the next two elements). (k) Result after convergence. (l) Conversion to m-connectivity.

Morphological thinning of a binary image

Figure 4.15 Morphological thinning of a binary image using the SEs of Figure 4.14 treated as a sequence. The first SE matches pixels (indicated by hashed squares) along the top of the region, which are then removed. The second SE matches no pixels, while the third SE matches pixels along the bottom, which are then removed. This process continues until convergence, yielding an approximate skeleton of the original image.



Thickening

- **Thickening** a binary image involves adding pixels
- to the foreground (that is, setting pixels to ON) while maintaining as much as possible the overall shape of the foreground regions.

$$I \boxplus B \equiv I \cup (I \circledast B)$$

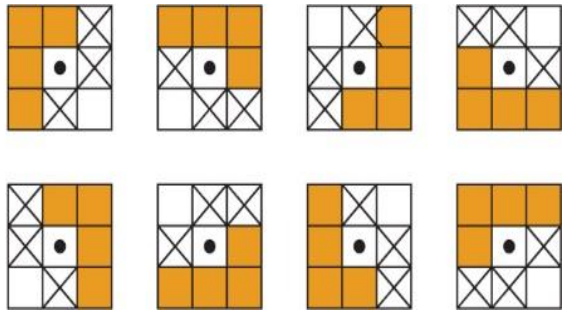
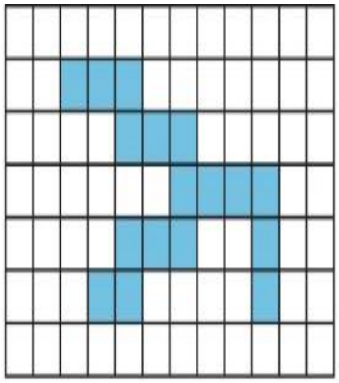


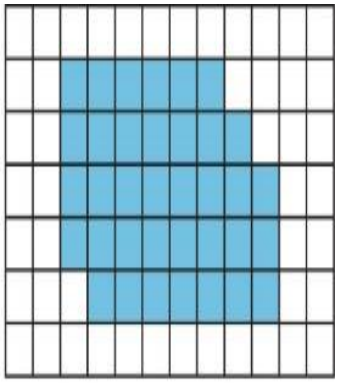
Figure 4.17 Structuring elements commonly used for morphological thickening.

Thickening (cont.)

Figure 4.18 Morphological thickening of a binary image using the SEs in Figure 4.17. Shown are the original image (left) and the final result after convergence (right). The thickened result is an approximation to the convex hull.

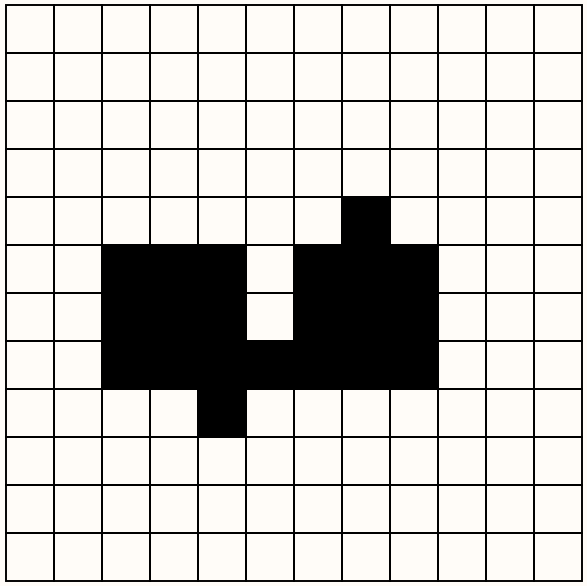


Input image

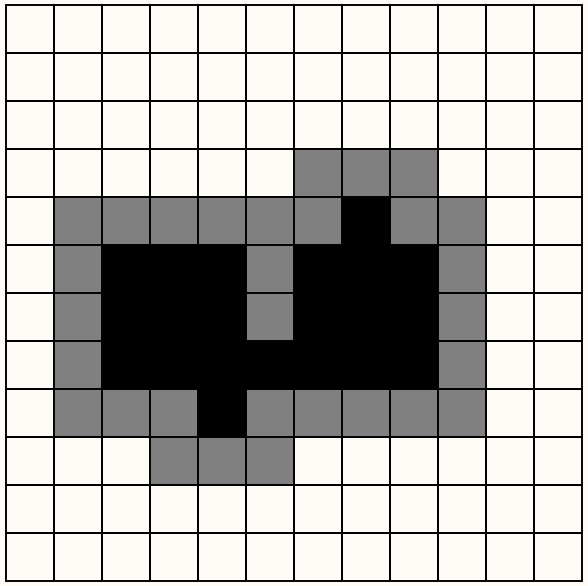


Final result

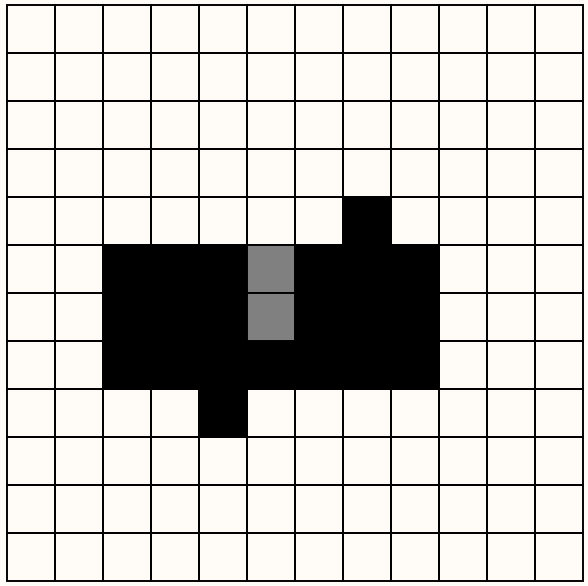
Closing = dilation then erosion



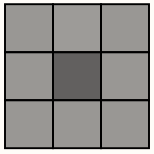
F



$F \oplus H$

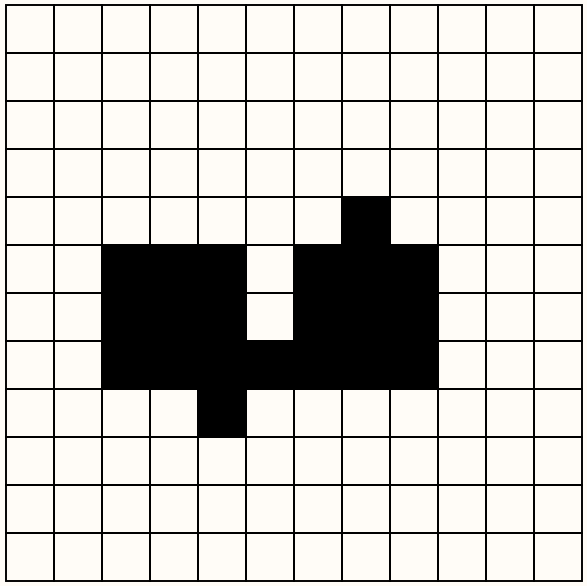


$(F \oplus H) \ominus H$

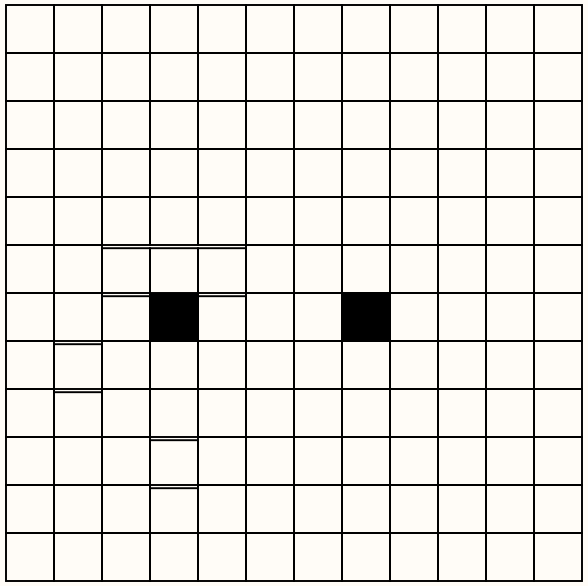


H, 3x3, origin at the center

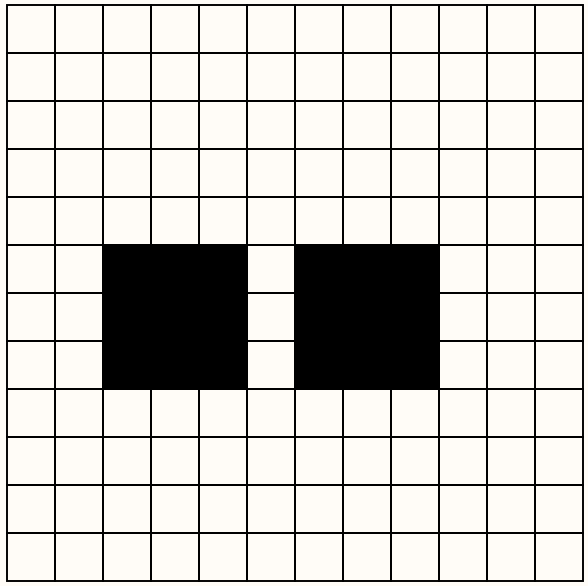
Opening = erosion then dilation



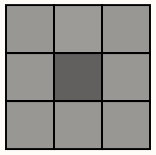
F



$F \ominus H$



$(F \ominus H) \oplus H$



H, 3x3, origin at the center

Labeling Regions

Neighbors and Connectivity

- A pixel q = (q_x, q_y) is a **neighbor** of pixel p = (p_x, p_y) if **q** is in the **neighborhood** of **p**, denoted $q \in N(p)$ where *N* is the neighborhood function.

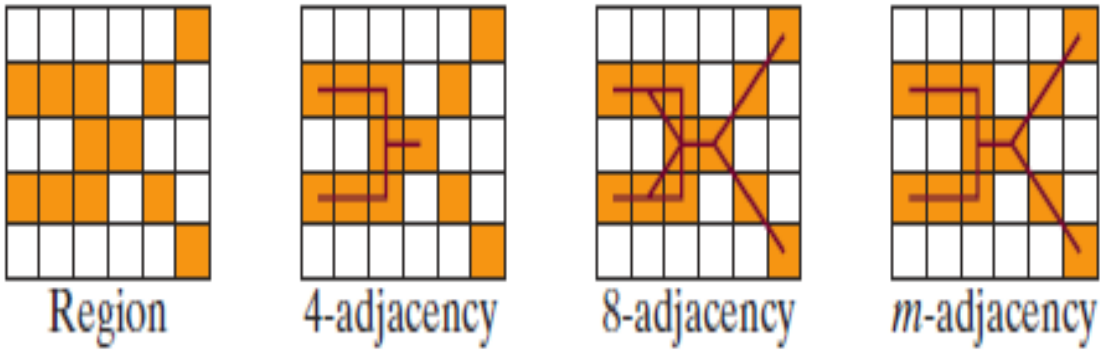
Figure 4.20 Commonly used neighborhoods. From left to right: $\mathcal{N}_4$, $\mathcal{N}_8$, and $\mathcal{N}_D$.

Neighbors and Connectivity (cont'd)

- Two pixels are said to be **adjacent** if they have the same value and if they are neighbors of each other.
- Pixels are said to be **connected** (or contiguous) if there exists a path between them.
- A **path** is defined as a sequence of pixels $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}$ such that $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$ are adjacent for all $i = 1, \ldots, n - 1$.
- A **region** in an image is therefore a set of connected pixels.

Neighbors and Connectivity (cont'd)

Figure 4.21 A binary region and the 4-, 8-, and *m*-adjacency of its pixels. Note that *m*-adjacency removes the loops that sometimes occur with 8-adjacency.



Floodfill

- **Floodfill (seed fill):** the problem of coloring all the pixels that are connected to a *seed pixel* with some desired new color.

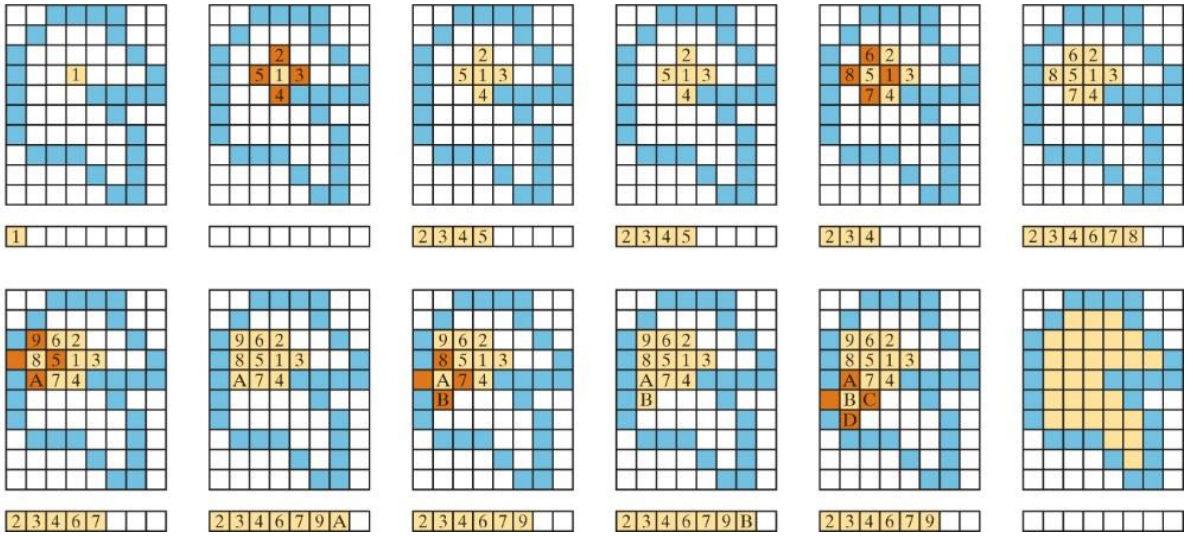


Figure 4.22 Step-by-step illustration of the 4-neighbor FLOODFILL algorithm on a small image. The frontier is shown below the image. Starting from the seed pixel labeled 1, the interior region of white pixels is changed to yellow by the algorithm, while orange is used to indicate the pixels being considered in the current expansion. The labels are artificially introduced to aid in associating pixels in the image with those in the frontier.

Conditional dilation

- The **conditional dilation** of an image I with respect to another image C is defined as dilation followed by intersection:

$$I \oplus_C B \equiv (I \oplus B) \bigcap C$$

- The image I is called the *marker image*, while C is called the *mask image*.

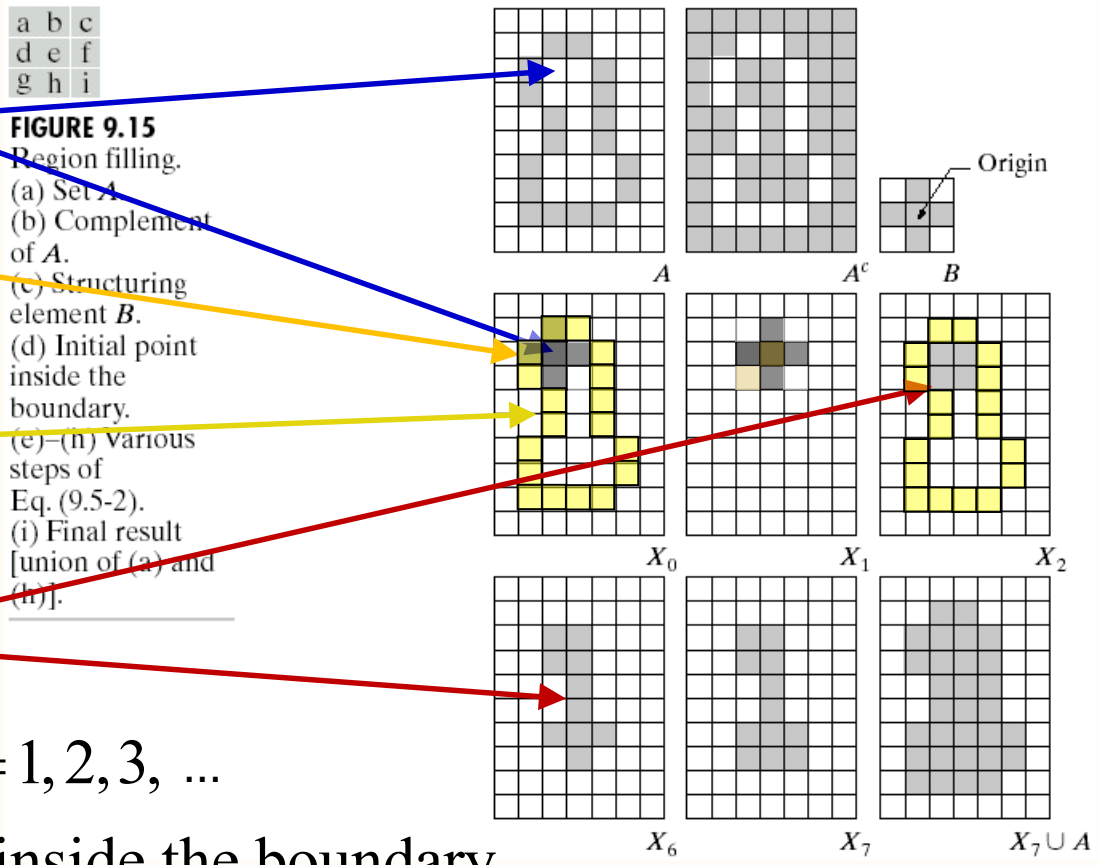
Region Filling via conditional dilation

1. Manually select a point inside the hole region.

2. Dilate that point.

3. Remove points that landed on A via intersection with inverse-A.

4. Repeat iteratively at each new point.



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

where $X_0 = p$, p is a point inside the boundary,

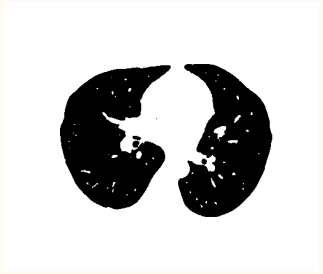
B is the symmetric structuring element.

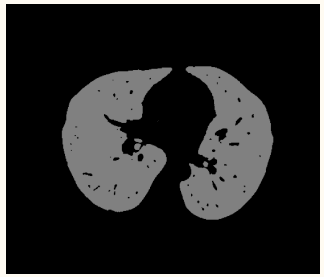
The algorithm teminates when $X_k = X_{k-1}$.

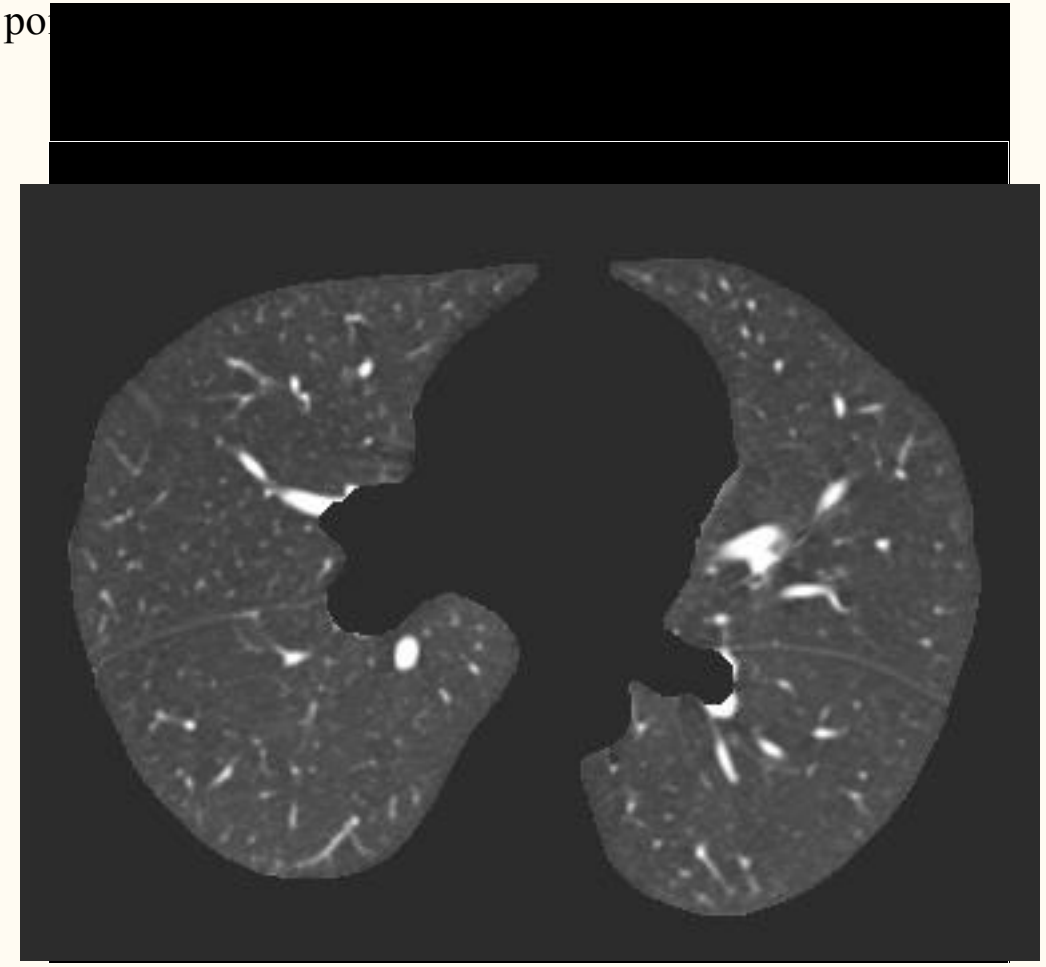
Final result: $X_k \cup A$

Example: Lung segmentation

- Apply an intensity threshold: -370 Hu
- Fill outside body.
- Invert and scale 0..128.
- Flood-fill outside, keeping only the added po
- Invert.
- Apply mask to CT image.







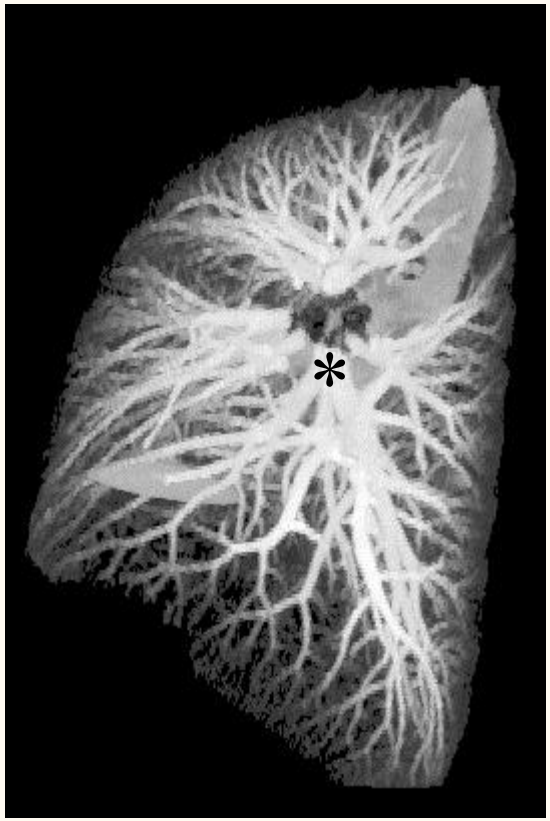
Connected Components

- **Connected component:** defined as a maximal set of pixels that are all connected with one another.

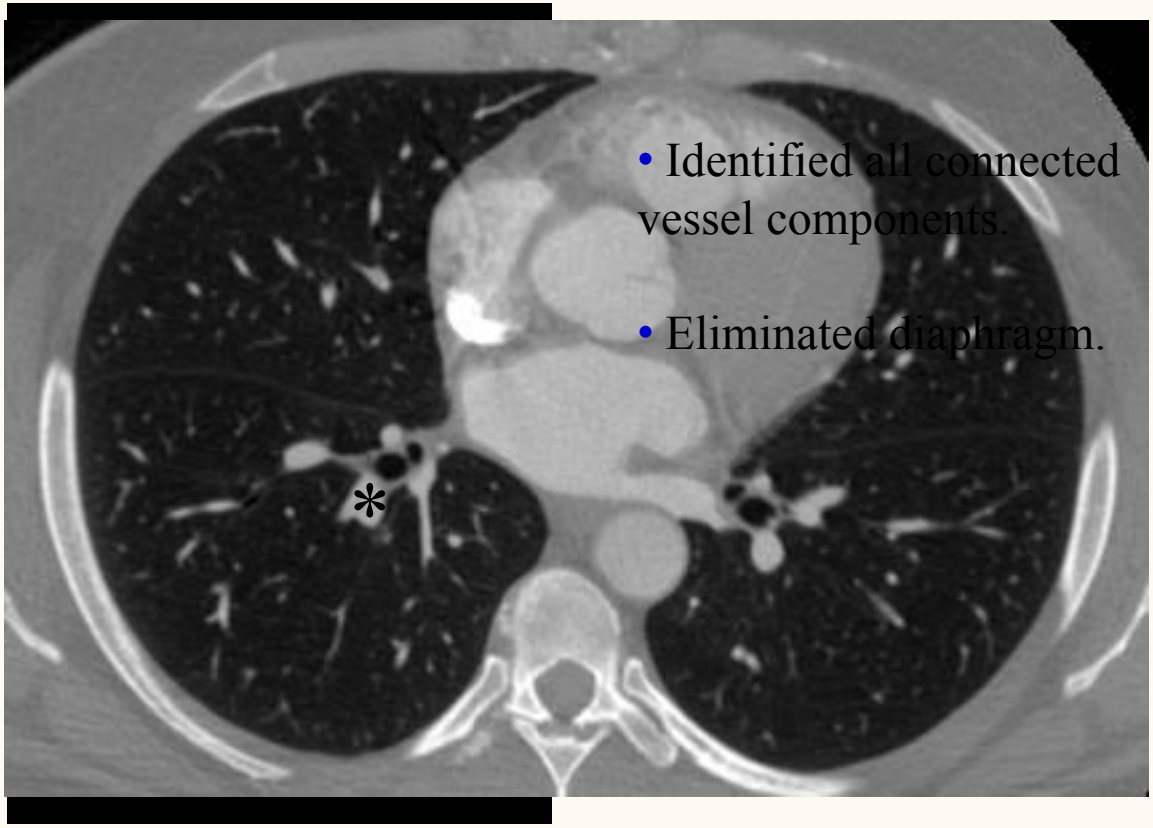
- **Connected component labeling:** the process of assigning a unique identifier to every pixel in the image indicating to which connected component it belongs.

Seeded region growing for connected component analysis

- Manually select seed point
- SRG = check whether each neighbor of seed pixel satisfies intensity threshold criterion and add to collection of 'vessel' pixels. Repeat for each accepted neighbor's neighbors.

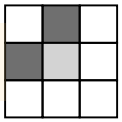


Original CT MIP



Connected vessels

Union-find algorithm

4-neighbor union element = 

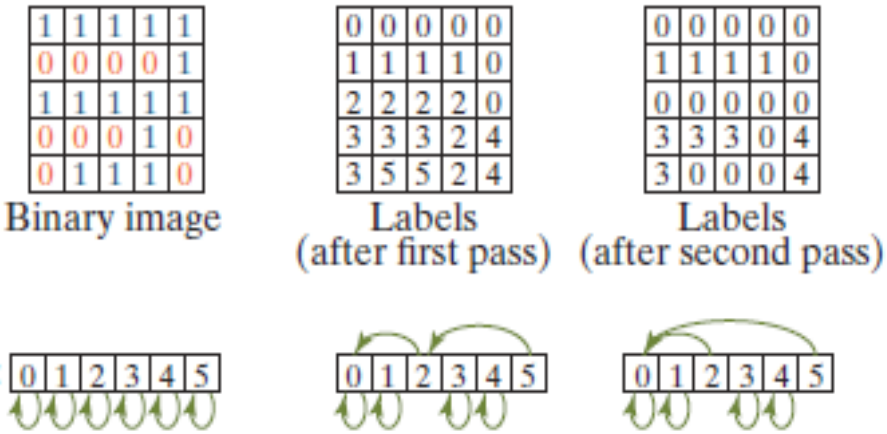


Figure 4.24 Classic union-find connected components algorithm on an example binary image. From left to right: The input image, the labels after the first pass, and the labels after the second pass. Below the image is the equivalence table, with green arrows pointing from a label to its equivalent label. Notice that the final image contains gaps; for example, no pixel is labeled 2.

Boundary Tracing

- **Region boundary**: the smallest set of pixels that encloses (in some sense) all the pixels in the region.
- **Hole boundary**: the smallest set of pixels that encloses all the holes (if any) inside the region.
- **Complete boundary**: the union of the region and hole boundaries.
- **Inner boundary**: consists of all pixels in the region that are next to some pixel not in the region.
- **Outer boundary**: consists of all pixels not in the region that are next to some pixel in the region.

Boundary Tracing

= dilated ‘D’ – original ‘D’

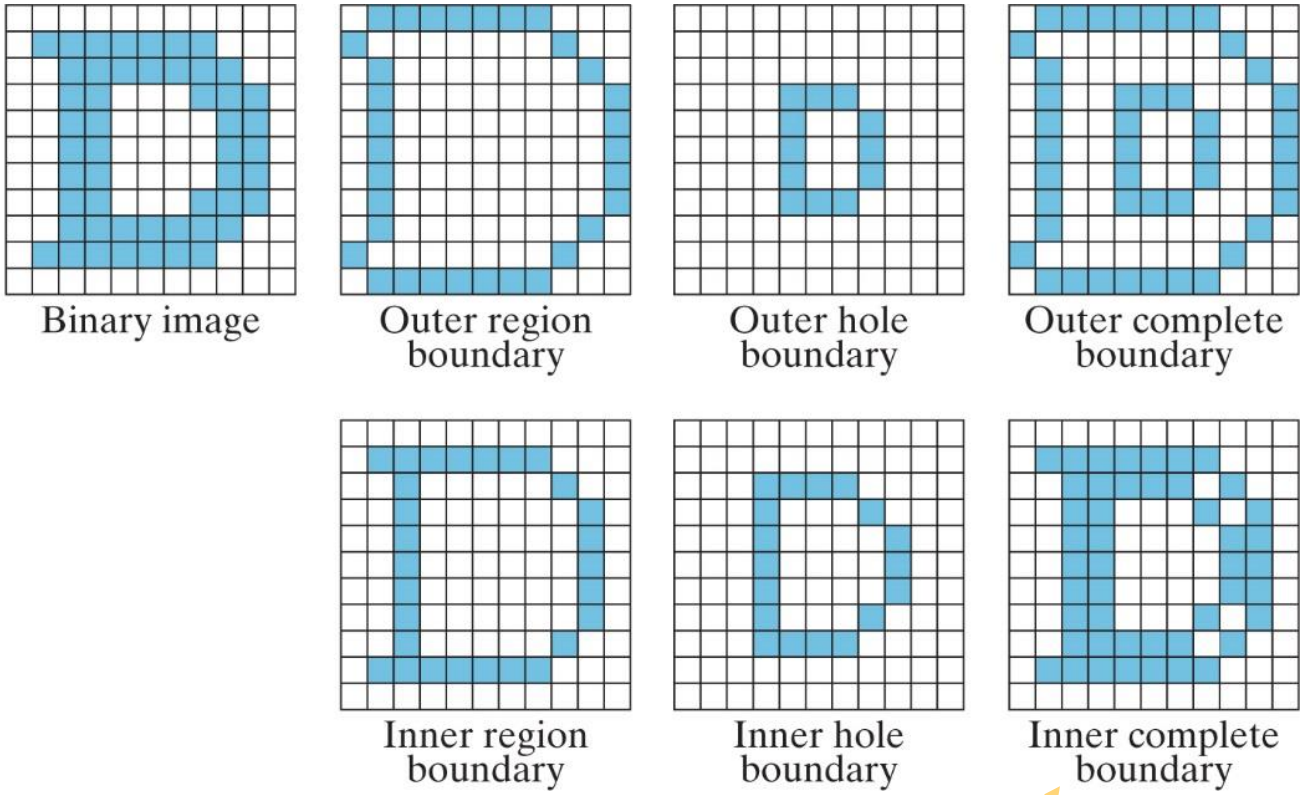


Figure 4.26 A binary region with one hole, and different definitions of the boundary of the region. All results are shown using 4-neighbors. Colored cells are ON, white cells are OFF.

= original ‘D’ - eroded ‘D’

Computing Distance in a Digital Image

Distance Functions

- Let p =(p_x, p_y) and q =(q_x, q_y) be the coordinates of two pixels in an image. A function $d(\mathbf{p},\mathbf{q})$ of two vectors is a distance function (or *metric*) if it satisfies three properties:

  - $d(\mathbf{p}, \mathbf{q}) \geq 0$ and $d(\mathbf{p}, \mathbf{q}) = 0$ iff $\mathbf{p} = \mathbf{q}$ (non-negativity and reflexivity)
  - $d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p})$ (symmetry)
  - $d(\mathbf{p}, \mathbf{q}) \leq d(\mathbf{p}, \mathbf{r}) + d(\mathbf{r}, \mathbf{q})$ (triangle inequality)

- for all possible coordinates $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r} \in \mathbb{R}^2$, where iff means "if and only if".

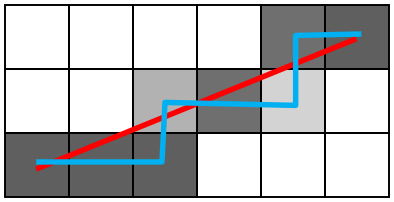
Distance Functions (Metrics) for Pixels

- **Euclidean distance**: the square root of the quadratic function.

$$d_E(\mathbf{p}, \mathbf{q}) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad \text{(Euclidean)}$$

- **Manhattan distance:** known as d_4 because the pixels that are one unit of distance away are the 4-neighbors of the pixel.

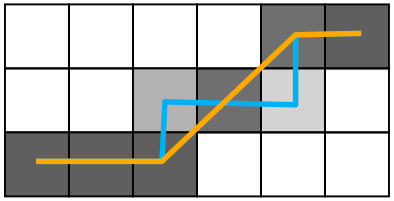
$$d_4(\mathbf{p}, \mathbf{q}) = |p_x - q_x| + |p_y - q_y| \qquad \text{(Manhattan, or city-block)}$$



Distance Functions (Metrics) for Pixels (cont'd)

- **Chessboard distance:** known as d_8 because the pixels that are one unit of distance away are the 8-neighbors of the pixel.

$$d_8(\mathbf{p}, \mathbf{q}) = \max(|p_x - q_x|, |p_y - q_y|) \qquad \text{(chessboard)}$$



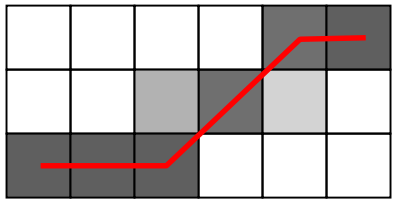
Count how many pixels are touched = 6

Path Length

- Since the Euclidean distance between two pixels that are 4-neighbors of each other is 1, and the Euclidean distance between two pixels that are D-neighbors of each other is $\sqrt{2}$, this is equivalent to measuring the length of the path Φ as:

- This is known as the **Freeman formula**.

$$\text{length}(\phi) = n_o + n_d\sqrt{2} \qquad \text{(Freeman)}$$



$n_o = 3$
$n_d = 2$

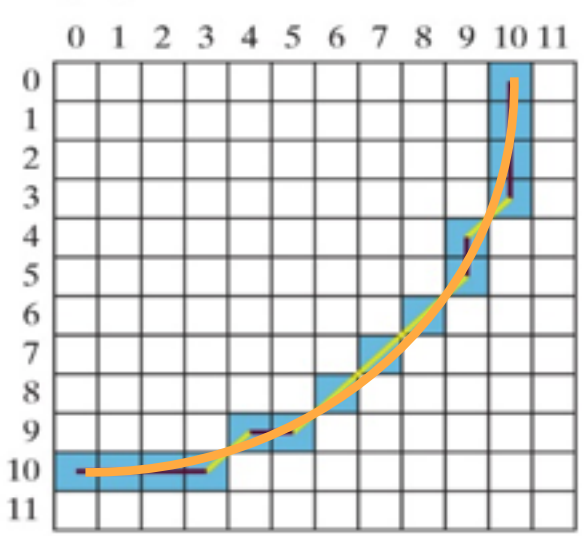
$L = 3 + 2\sqrt{2}$

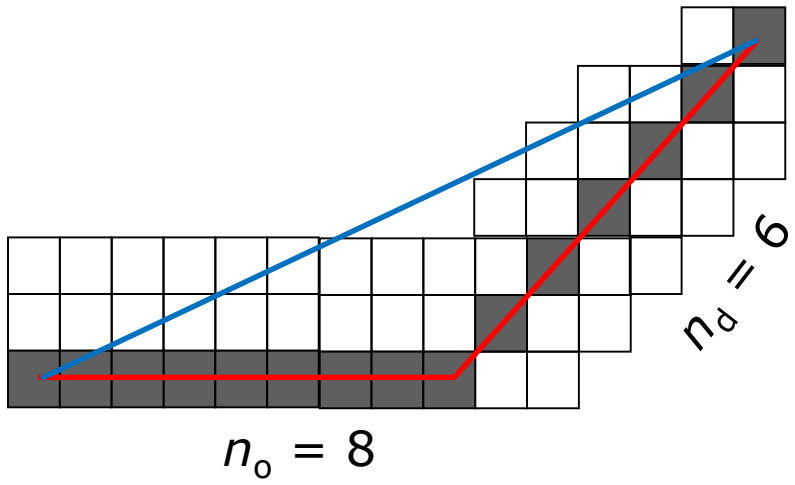
Path Length: example

Rearrange the path of pixels:

there are 8 face-to-face steps: $n_o = 8$

and 6 diagonal steps: $n_d = 6$





True underlying curve

Freeman formula > true length

Pythagorean < true length

Path Length (cont'd)

- An alternate approach is to rearrange the node pairs and use the **Pythagorean theorem** to estimate the length of the curve as the hypotenuse of the resulting right triangle:

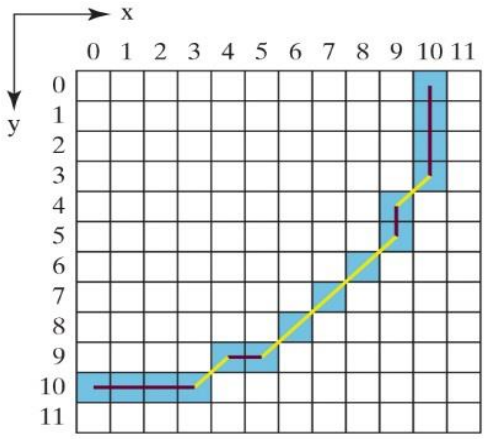
$$\text{length}(\phi) = \sqrt{n_d^2 + (n_o + n_d)^2} \qquad \text{(Pythagorean)}$$

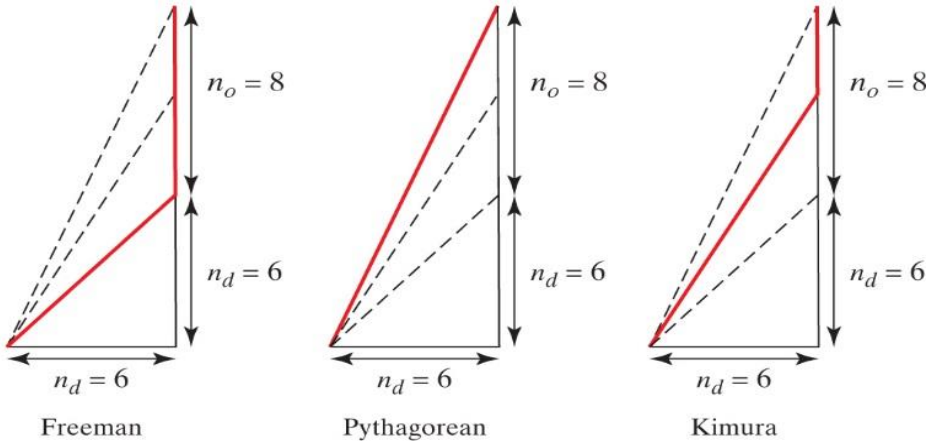
- By setting *c* to ½, a compromise is achieved between overestimation and underestimation known as the **Kimura** method.

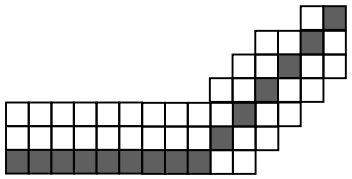
$$\text{length}(\phi) = \sqrt{n_d^2 + (n_d + cn_o)^2} + (1 - c)n_o \qquad \text{(Kimura)}$$

Path Length (cont'd)

Figure 4.29 A discretized 90-degree sector of a circle with radius 10, where purple lines indicate isothetic moves, and yellow lines indicate diagonal moves. The number of isothetic and diagonal moves are $n_o = 8$ and $n_d = 6$, respectively. The true arc length is $10(\pi/2) = 15.7$. The estimated path length according to the three formulas is 16.5 (Freeman), 15.2 (Pythagorean), and 15.7 (Kimura).







Region Properties

Moments

- **Regular Moments:** given the discrete function *f* and nonnegative integers *p* and *q*, the pq^{th} **moment** of a 2D region is defined as:

$$m_{pq} \equiv \sum_x \sum_y x^p y^q f(x, y)$$

- ***pq*th central moment:** defined as the pq^{th} regular moment about the centroid:

$$\mu_{pq} \equiv \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

Area

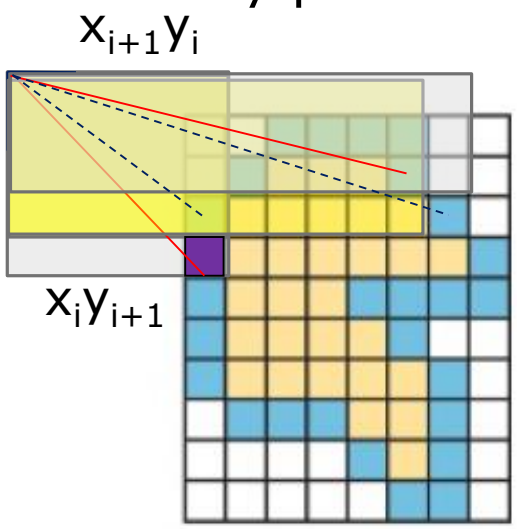
- The area of a region is given by its zeroth moment $m_{00} = \mu_{00}$.
- For a binary region, this is simply the number of pixels in the region.

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y),\ m_{00} = \sum_x \sum_y x^0 y^0 \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} = \#ON\ pixels$$

- Alternatively, given the pixels defining the boundary, the area of the polygon defined by these boundary pixels is given by:

$$\text{area} = \frac{1}{2} \sum_{i=0}^{n-1} (x_{i+1} y_i - x_i y_{i+1})$$

Area of the rectangle defined by the X coordinate of the next boundary pixel and Y value of the current boundary pixel



Centroid

- The centroid X,Y coordinates are computed via the m_{10} and m_{01} moments and the number of pixels (= area).

$$Cen_X = \frac{m_{10}}{m_{00}} = \frac{\sum_x \sum_y x^1 y^0 \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}}{m_{00}} = \frac{\sum_x \sum_y x \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}}{m_{00}}$$

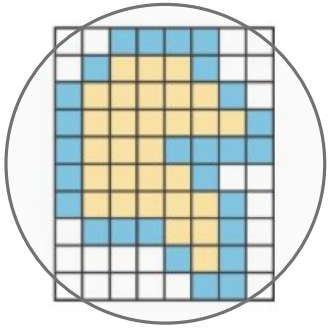
$$Cen_y = \frac{m_{01}}{m_{00}}$$

Moments (cont'd)

- **Normalized Central Moments:** invariant to translation and uniform scaling.

$$\eta_{pq} \equiv \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$$

1. Move coordinate system origin to object centroid
2. Find farthest pixel from centroid and scale pixel coordinates as % of the maximal: 0.0 to 1.0



- **Hu Moments:** invariant to translation, uniform scaling, and rotation. They are natural extensions.

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

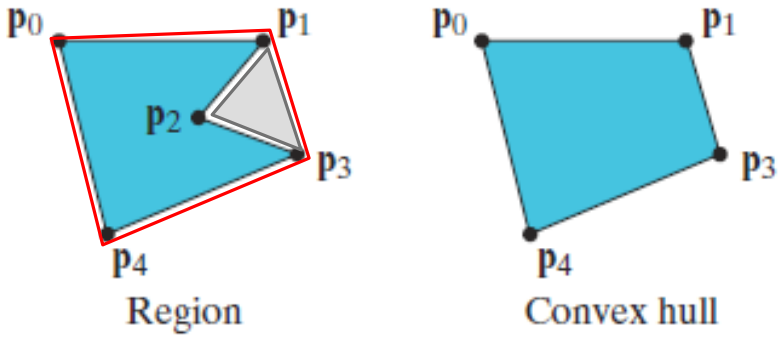
$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Convex Hull

- A set of points in the plane is **convex** if any straight line between two points in the set lies entirely within the set.
- The **convex hull** of a set is the smallest convex set containing the set.
- The difference between the set and its convex hull is called the **convex deficiency**, which is sometimes used as a descriptor of the shape of the object.

Convex Hull (cont'd)

Figure 4.35 LEFT: An arbitrarily-shaped region in the plane. RIGHT: The convex hull of the region is the shape that results from enveloping the region with a rubber band, which removes all concavities. All vertices are locally convex except for $\mathbf{p}_2$.



convex deficiency = 

Perimeter

- The perimeter of a region is typically computed by applying the Kimura distance to the boundary found by wall following. Similar to area, we have:

$$\text{perimeter}(I_1) + \text{perimeter}(I_2) \approx \text{perimeter}(I_1 \cap I_2) + \text{perimeter}(I_1 \cup I_2)$$

Compactness

- **Compactness:** a measure of how close the pixels in the region are to the center of the region.

$$\text{compactness} = \frac{4\pi(\text{area})}{(\text{perimeter})^2}$$

Eccentricity

- The **eccentricity of a region** measures its elongatedness—that is, how far it is from being rotationally symmetric around its centroid.

- Since the eigenvalues capture the variance in the two principal directions, the eccentricity of a region is defined as the difference between these variances, normalized by the larger variance.

$$\text{eccentricity} = \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_1}}$$

Topology

- **Topology:** the study of properties of objects that are preserved under continuous deformations of the objects.

- Such deformations allow for bending, stretching, and compressing, but not tearing or sewing.

- In mechanics the analogy is maintaining a 1-to-1 correspondence between points = no compression to a point or line; no singularities.

- The mathematical name for such a deformation is a **homotopy.**

Euler Number

- **Euler number (or *Euler characteristic*):** an important topological invariant which is defined as the number of regions minus the number of holes:

  Euler number = number of regions - number of holes

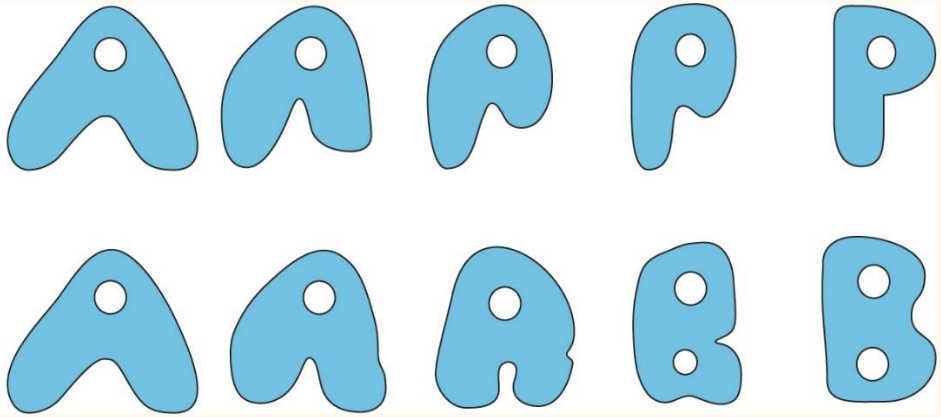


Figure 4.38 TOP: The letters "A" and "P" are related by a homotopy, because there is a continuous deformation that relates the two shapes. BOTTOM: The letters "A" and "B" are not related by a homotopy, because there is not a continuous deformation that relates the two shapes. Rather, tearing the region to produce the extra hole is necessary (or sewing the hole in the case of the reverse transformation).

Skeletonization

Skeletonization

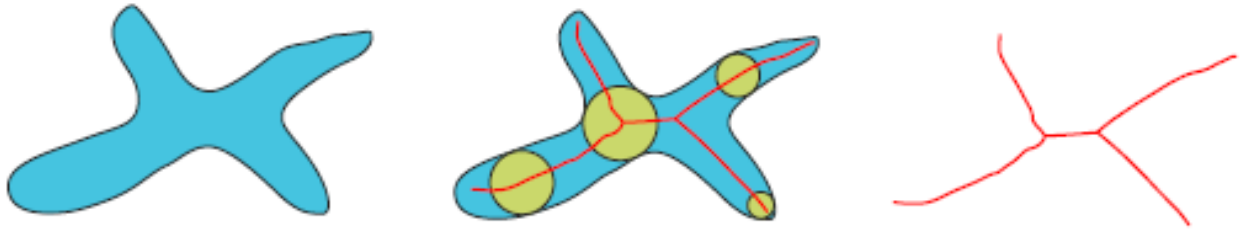


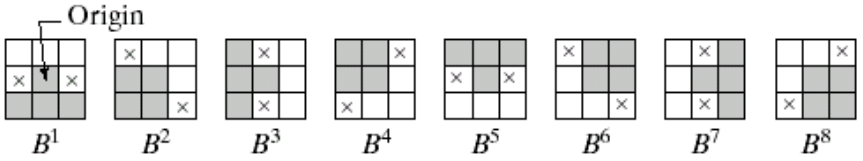
Figure 4.42 The skeleton of a binary region is defined as the locus of points where the wave fronts of fires set to the boundary meet, or equivalently as the locus of the centers of the maximal balls.

Find the largest circle that fits entirely within the object along any path, and its center lies on the skeleton

Thinning (2)

Top top-rt right bot-rt bottom bot-left left top-left

top edge SE



Consider surrounding pixels to be OFF

Goal to remove pixels from the edge but symmetrically to get the center line.
1. Pass each SE over entire image once.

2. Do each SE in succession.
3. Repeat as needed.

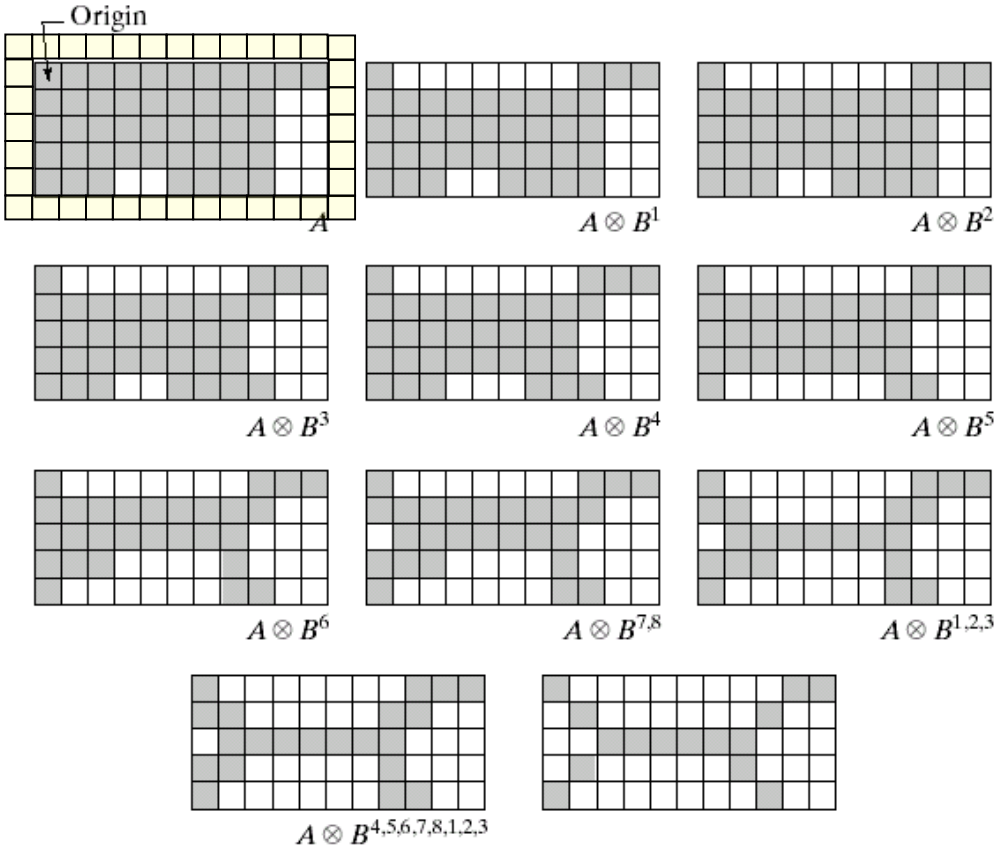
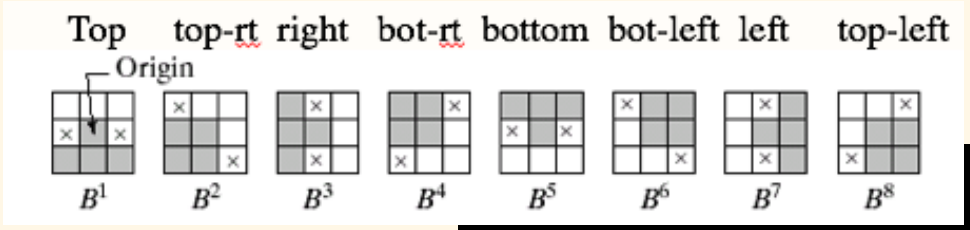
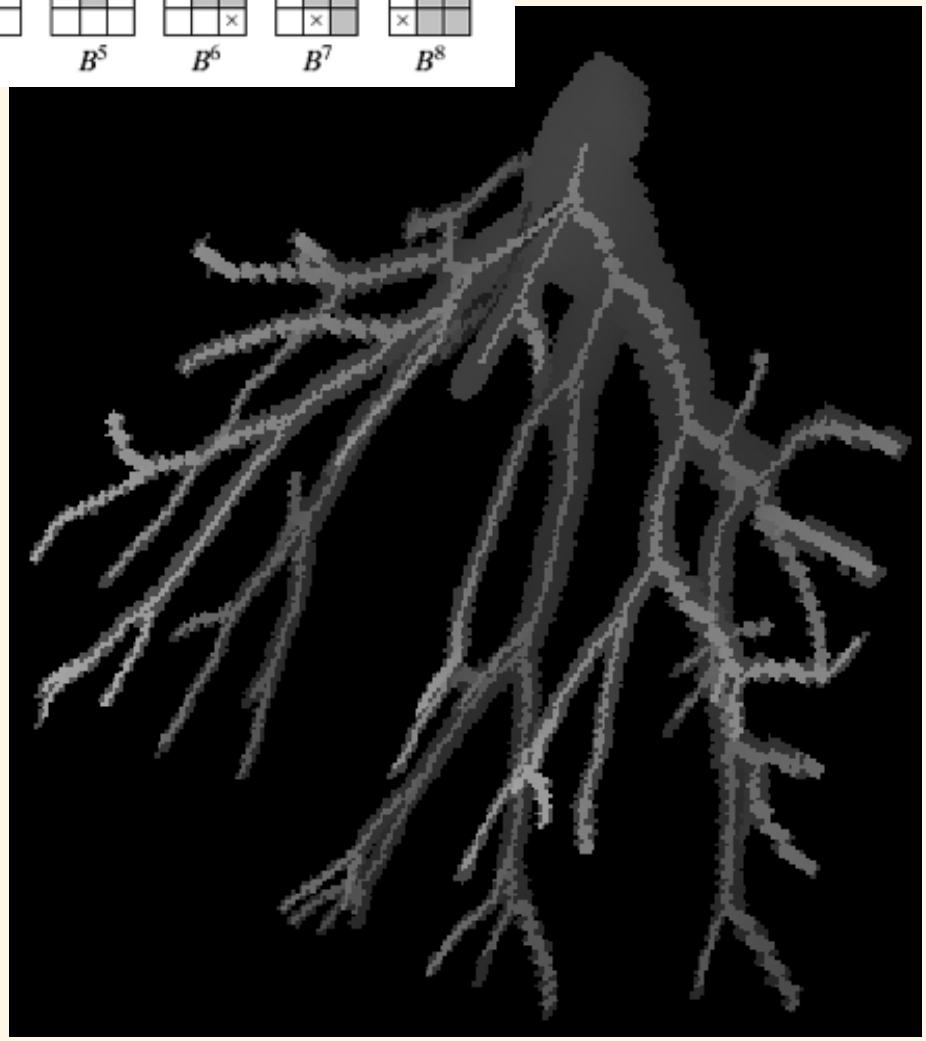


FIGURE 9.21 (a) Sequence of rotated structuring elements used for thinning. (b) Set A. (c) Result of thinning with the first element. (d)–(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first element again (there were no changes for the next two elements). (k) Result after convergence. (l) Conversion to m-connectivity.

Skeletonization: blood vessels example





Skeletonization by Thinning

- A common approach to skeletonization is to repeatedly thin the image until the result converges.

Figure 4.43 Six different continuous shapes (blue) and their skeletons (thin red lines).

Skeletonization by Thinning (cont'd)

- **Connection number**: the number of regions that are 8-connected to the central pixel.
  - The connection number is the number of 8-connected foreground (ON) regions in the 3 X 3 pattern that would remain if the central pixel were set to OFF, minus the number of holes that would be created.

$$\psi = \underbrace{\psi_{01,4}}_{0-1 \text{ transitions}} + \underbrace{\psi_{010,c}}_{\text{isolated corners}}$$

Skeletonization by Thinning (cont'd)

Textbook error:

Textbook error 2: equivalent SEs
(free root beer to the first person who finds the missing σ=4 SE)

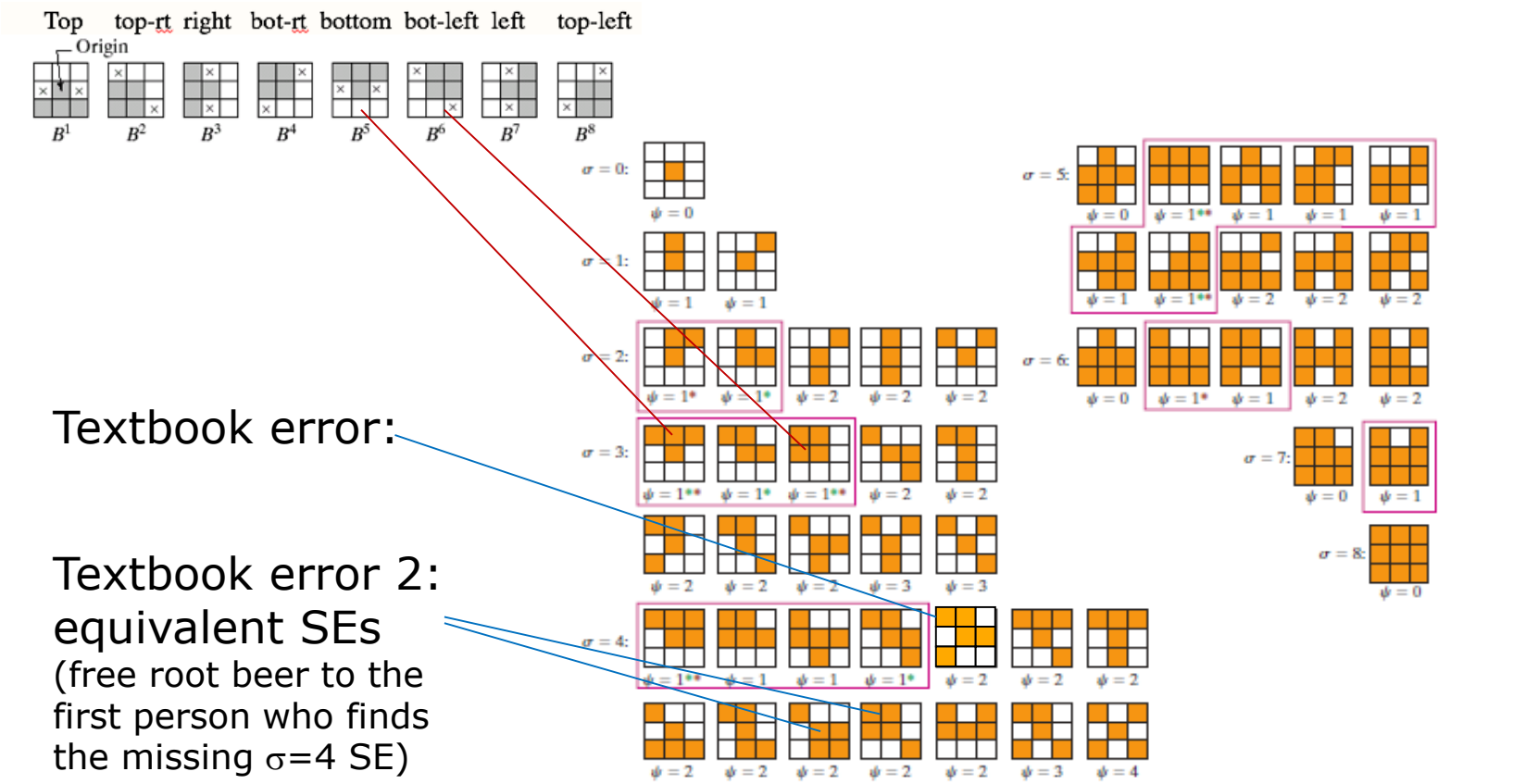


Figure 4.45 The 50 unique 3 × 3 binary patterns with ON in the center, along with their σ value (number of ONs in the 8-neighbors) and ψ value (connection number). The remaining 206 patterns are obtained by rotating and/or reflecting these patterns, which does not affect either σ or ψ. For all patterns with $\psi \neq 0$, the connection number is the number of 8-connected foreground regions that result if the central pixel is set to OFF. For patterns with $\sigma \neq 0$ and $\psi = 0$, setting the central pixel to OFF creates a hole. The purple boxes enclose the patterns whose central pixel is removed by the sigma-psi algorithm, while the green and brown asterisks indicate the patterns whose central pixel is removed by morphological thinning and Zhang-Suen, respectively. The former algorithm is more aggressive in removing pixels than the other two.

Sigma-Psi Algorithm

- This algorithm iterates through an image, examining pixels and deleting those for which $\sigma \neq 1$ and $\psi = 1$.
  - Select a pixel, scan entire SE library for a match.
  - Go on to next pixel, etc.
  - The process continues until convergence.

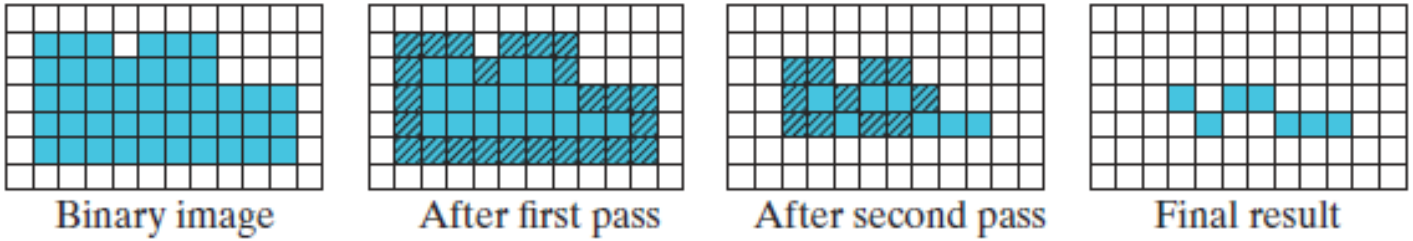


Figure 4.46 Skeletonization using the σ-ψ algorithm. Each pixel is examined in turn, and if $\sigma \neq 1$ and $\psi = 1$, then it is deleted. In the first pass, all the pixels along the border of the region are deleted. In the second pass, a number of additional pixels are deleted, with the particular pixels chosen being dependent upon the order in which they are examined. The final result is indeed a thinned version of the input. This is an S-type algorithm, because the segments touching the corners are not preserved.

Zhang-Suen Algorithm

- Zhang-Suen repeatedly applies two subiterations to the image.
- In the first subiteration, pixels are flagged for removal if they meet the following 4 tests:

a) $2 \leq \sigma \leq 6$

b) $\psi_{01,8} = 1$

c) $p_1 \cdot p_3 \cdot p_5 = 0$

d) $p_3 \cdot p_5 \cdot p_7 = 0$

p8	p1	p2
P7	p0	p3
p6	p5	p4

Zhang-Suen Algorithm (cont'd)

- In the second sub-iteration, pixels are flagged for removal if they meet four tests, the first two of which are identical to those above, while the latter two are slightly changed:

$$c') \; p_1 \cdot p_3 \cdot p_7 = 0$$

$$d') \; p_1 \cdot p_5 \cdot p_7 = 0$$

p8	p1	p2
P7	p0	p3
p6	p5	p4

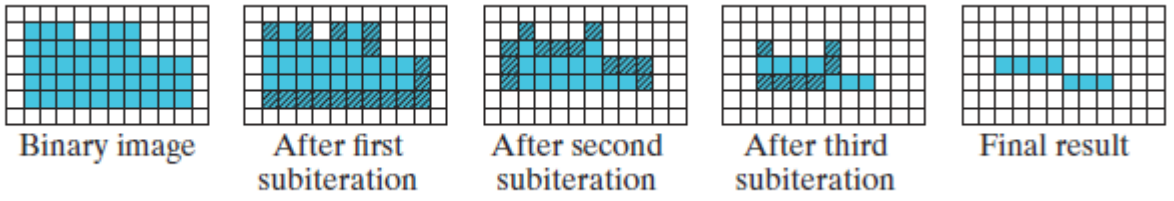
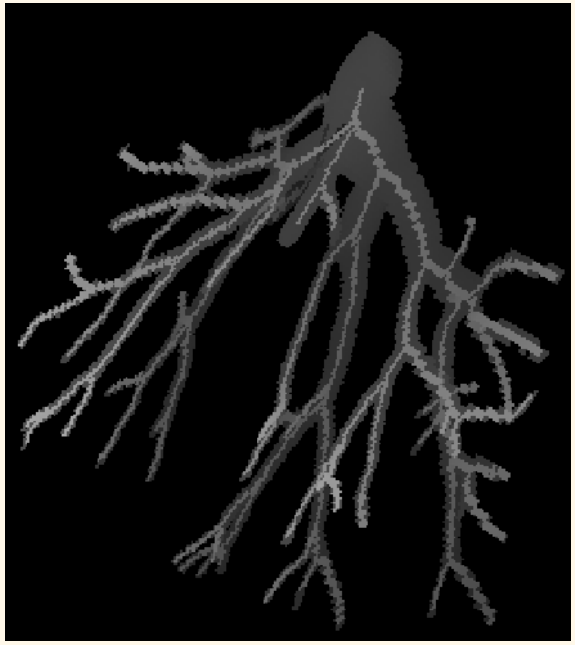


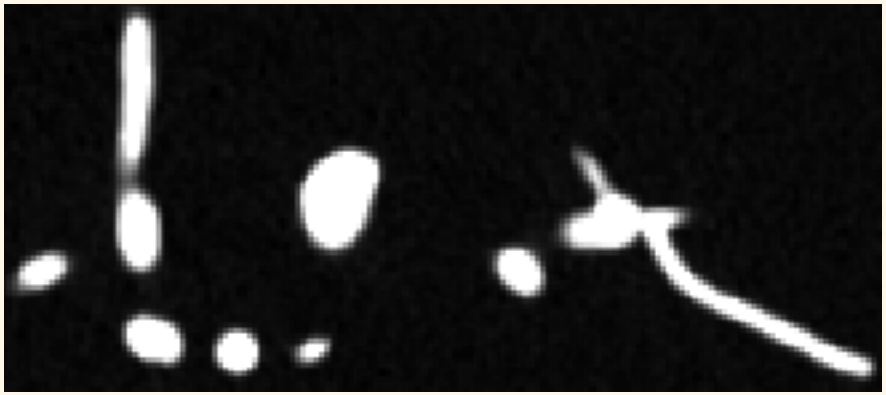
Figure 4.47 The Zhang-Suen algorithm applied to a binary image.

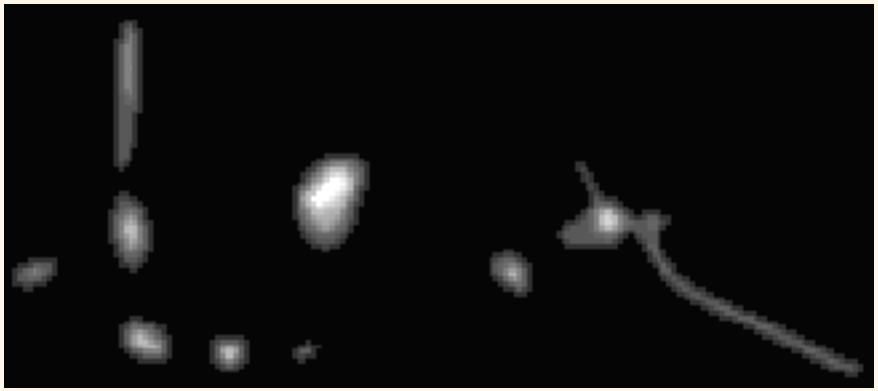
Distance transform:
= how far a pixel is from the nearest border



- For every pixel, search the neighborhood for the closest OFF (or ON) pixel, and compute Euclidean distance.
- Search in concentric circles until you find matches, and find minimum pixel.
- Slow, inefficient.

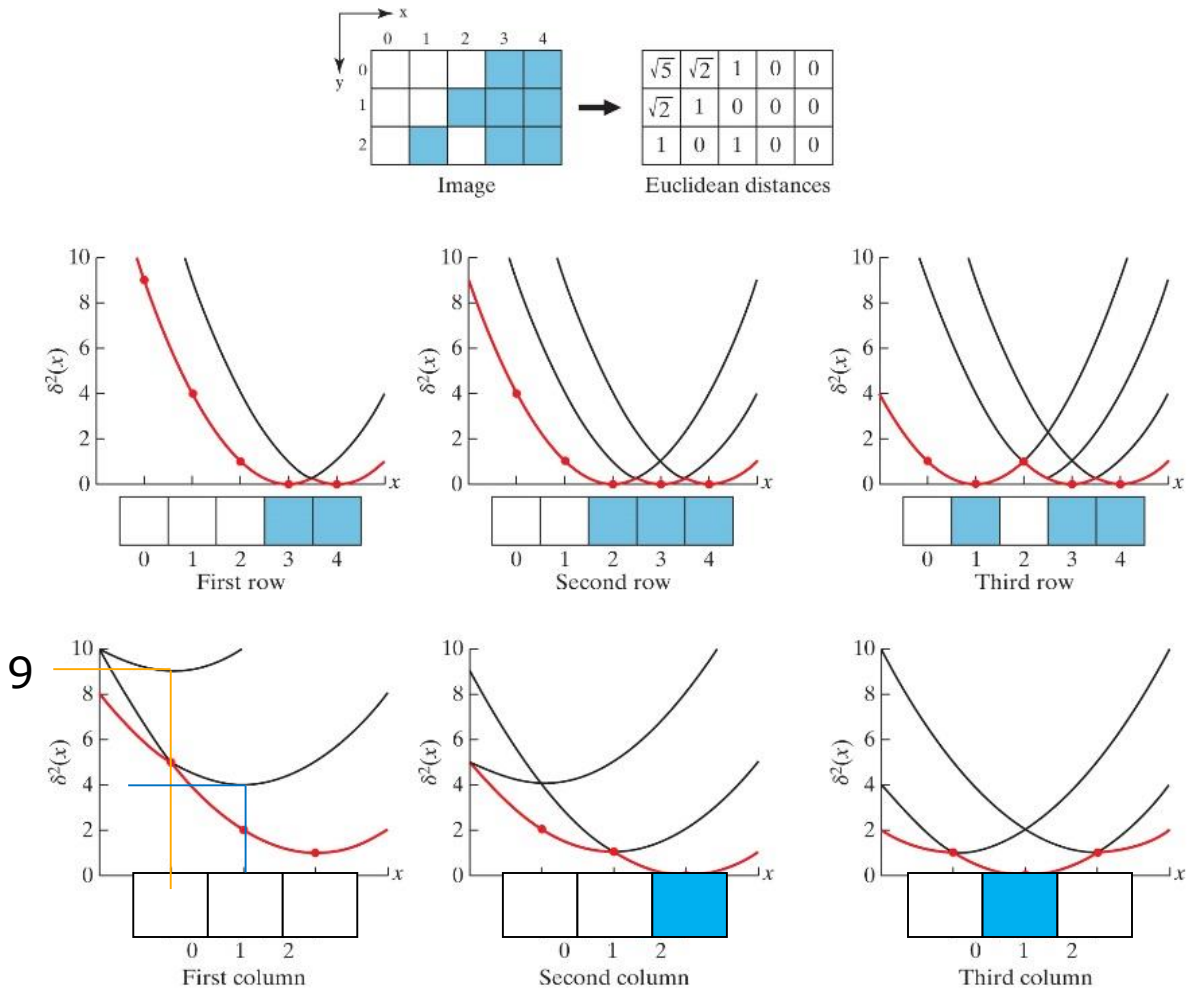
A single CT slice through the vessel tree





Distance map:
pixel intensity = distance to the vessel edge.
value at centerline pixels = vessel radius

Exact Euclidean Distance



9	4	1	0	0
4	1	0	0	0
1	0	1	0	0

Eval pixel (0,0):
1st column gives 9
2nd col gives 5
3rd col gives 5
Best match is 5 = squared distance

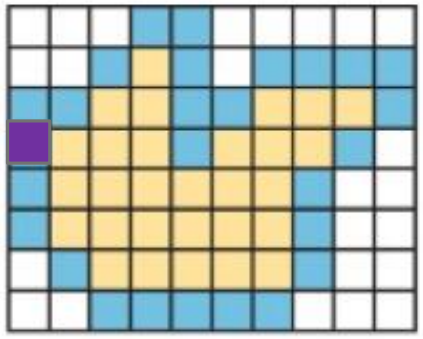
Figure 4.32 Computation of exact Euclidean distance from each pixel in a binary image to the nearest ON (blue) pixel. The first pass processes rows of the image to compute squared distances using the lower envelope of the parabolas. (The lower envelope is shown in red.) The second pass processes columns of the image using parabolas determined by the first pass. Based on P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1): 55-79, Jan. 2005.

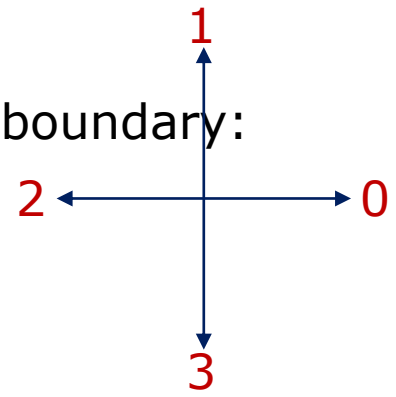
Boundary Representations

Chain Code

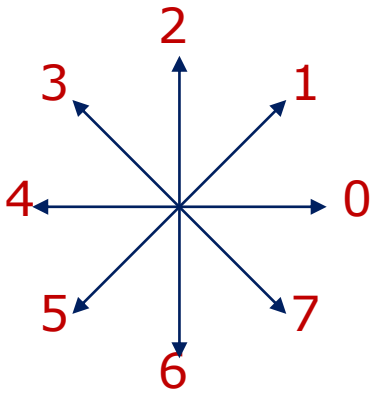
- For distinguishing the shape of the boundary, generally we want to transform the sequence into a representation that is invariant to translation, rotation, and/or scale changes, as well as to the starting pixel.

- The most famous chain code is the **Freeman chain code.**
  - Rather than prescribing pixel coordinates, give direction of travel to next pixel in path.
  - Prescribe a step direction as 0,1,2 or 3
    or 0-7 for 8-neighbor method
  - Sequence of steps draws object boundary:





or



= 20110660100065566544443332

Signature

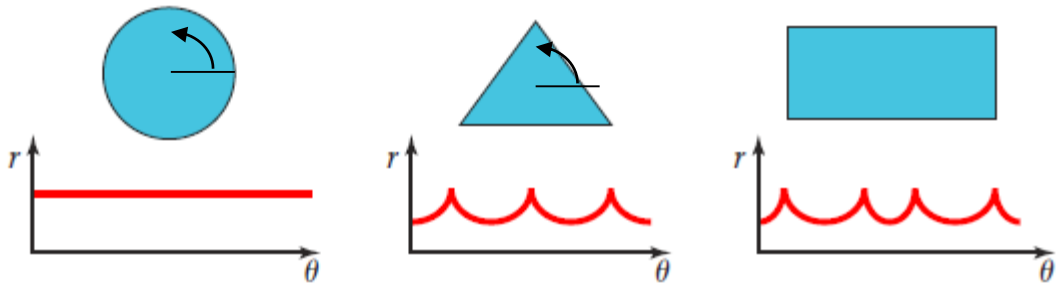
- A common approach to representing the boundary of a region is via some type of **signature**.
- For all signature representations, the region is typically first rotated using the principal axis to ensure that the representation is independent of the starting pixel.
- Then the left-right ambiguity is resolved by projecting the region onto the principal axis and computing some property of the projection function, such as the side with the most mass.

Signature (cont'd)

- The most basic type of signature is known as the **centroidal profile**, or r-θ curve.
- This approach captures the distance r from the center of the region as a function of the angle θ.

= cylindrical coordinates representation

Figure 4.52 The centroidal profile (r-θ plot) of several shapes.



B-Spline

- A uniform cubic B-spline is represented as a sequence of **control points** q_i, $i = 0, \ldots, n + 1$ in the plane.

- The spline is parameterized by a real parameter s, which varies from 1 to n, that is, $1 \leq s \leq n$. A point on the spline is represented as $\mathrm{x}(s) = (x(s), y(s))$

$$\mathbf{x}(s) = [x(s) \;\; y(s)] = \mathbf{v}^{\mathsf{T}}\mathbf{M}\mathbf{Q}_i$$

http://jsxgraph.uni-bayreuth.de/wiki/index.php/B-splines

B-Spline (cont'd)

- Where

$$\mathbf{v}^{\mathsf{T}} \equiv [\alpha^3 \;\; \alpha^2 \;\; \alpha \;\; 1] \quad \text{and} \quad \mathbf{Q}_i \equiv \begin{bmatrix} \mathbf{q}_{i-1} \\ \mathbf{q}_i \\ \mathbf{q}_{i+1} \\ \mathbf{q}_{i+2} \end{bmatrix}$$

$$\mathbf{x}(1) = \frac{1}{6}(\mathbf{q}_0 + 4\mathbf{q}_1 + \mathbf{q}_2)$$

$$\mathbf{x}(n) = \frac{1}{6}(\mathbf{q}_{n-1} + 4\mathbf{q}_n + \mathbf{q}_{n+1})$$

B-Spline (cont'd)

- **Computing the Slope of a Spline:**
  - The slope is given by:

$$\frac{\partial y(s)}{\partial x(s)} = \frac{\partial y(s)}{\partial s}\frac{\partial s}{\partial x(s)} = \frac{\mathbf{v'}^{\mathsf{T}}\mathbf{M}\mathbf{y}_i}{\mathbf{v'}^{\mathsf{T}}\mathbf{M}\mathbf{x}_i}$$

- **Constructing the Spline:**

$$\begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n+1} \end{bmatrix} = \frac{1}{6}\begin{bmatrix} 5 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 4 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & 5 \end{bmatrix}\begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_{n+1} \end{bmatrix}$$

Questions?

Slide Credits

- Image Processing and Analysis by Stan Birchfield
- Digital Image Processing using Matlab, by Gonzalez, Woods, & Eddins

Errata:

- Fig 4.12: The smiley-face SE is not rotated prior to erosion, violating the definition via Minkowski subtraction.
- Fig 4.21: The first figure is not a region using N_4 neighborhood so the label is imprecise.
- Fig 4.45: under σ=4, top row, 5th column SE, the middle-bottom pixel should be white
- Fig 4.45: under σ=4, bottom row, 3rd and 4th SEs are equivalent
- Fig 4.45: under σ=4, 1st SE in bottom row and 6th in top row are equivalent
- Fig 4.45: under σ=4, a student identified the plain cross as 1 of the 2 missing SEs.
- Fig 4.22: top row figs 2&3 are identical
- Page 197, middle of paragraph Σ should be σ.