UNIVERSITÄT OSNABRÜCK

University of Osnabrück, Institute of Cognitive Science

Computer Vision:

5 – Morphological Operators

WS 2019 / 2020

Gunther Heidemann

Morphological operators

- Idea: Search image for a (small) pre-defined pattern
- Originally defined for binary images, extension to gray values possible
- Morphological operation is specified by structuring element *S*
- Operation is translation invariant and nonlinear
- Application examples
  - Smoothing of segmented regions
  - Removal of disruptions
  - Recognition of simple patterns
- Very simple operation
- Well suited for parallel implementation

Morphological operators

- Let S be a binary matrix (“*structuring element*”): $S(i,j) \in \{0,1\}$.
- S moves across the image line by line like a filter kernel
- S and the underlying image patch are compared and the resulting similarity value is assigned to a result matrix (same size as the input image) at the location corresponding to the anchor point of S. Commonly the center of S is the anchor point.
- S must be shaped according to expected patterns in the image.
- Two ways to apply S:
  - Erosion
  - Dilation
- First only binary images and binary S.

Erosion

University of Osnabrück, Institute of Cognitive Science

1. Erosion:

- Assign 1 to the result pixel if and only if *all* 1-elements of S cover 1-pixels of image g, else assign 0:

  $$g'(x,y) = \Theta\Big(\big(\textstyle\sum_{k\in[-m,m]} \sum_{l\in[-n,n]} S(k+m,l+n)\cdot g(x+k,y+l)\big) - N + \tfrac{1}{2}\Big)$$

  where $N = \sum_{k\in[0,2m+1]} \sum_{l\in[0,2n+1]} S(k,l)$ = # 1s in S

- Alternative notation:

  $$g'(x,y) = \bigwedge_{k\in[-m,m]} \bigwedge_{l\in[-n,n]} \big(S(k+m,l+n) \rightarrow g(x+k,y+l)\big),$$

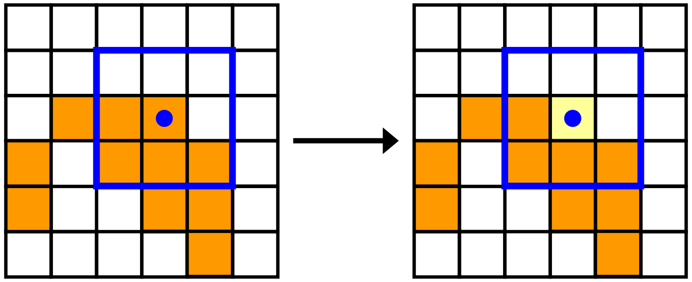
  where $\rightarrow$ denotes the (logical) implication.

- Short: $g' = g \ominus S$

- Erosion cuts off fringe of objects

$$S = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

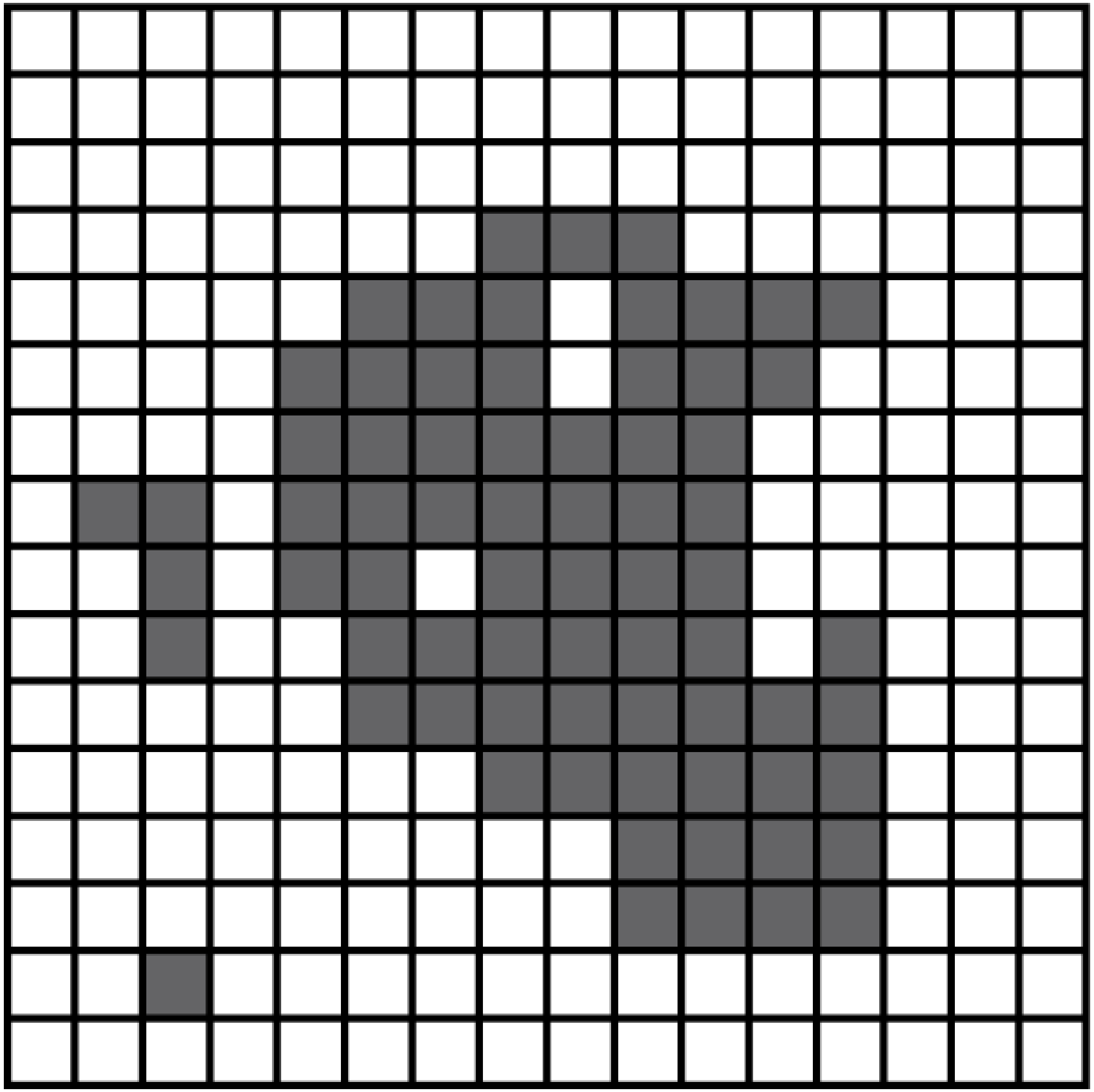
1 = 



[H]

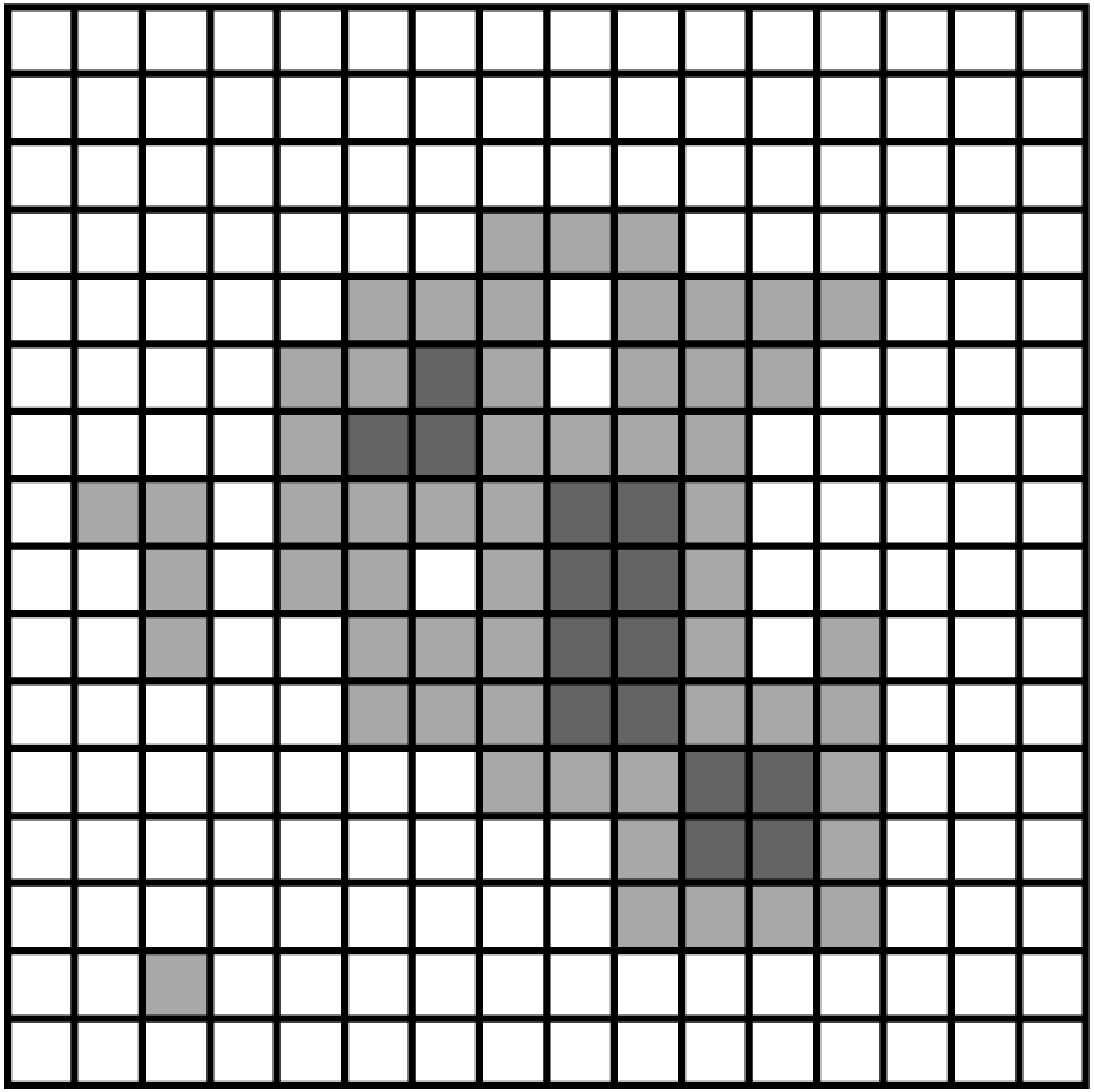
Erosion

Binary object



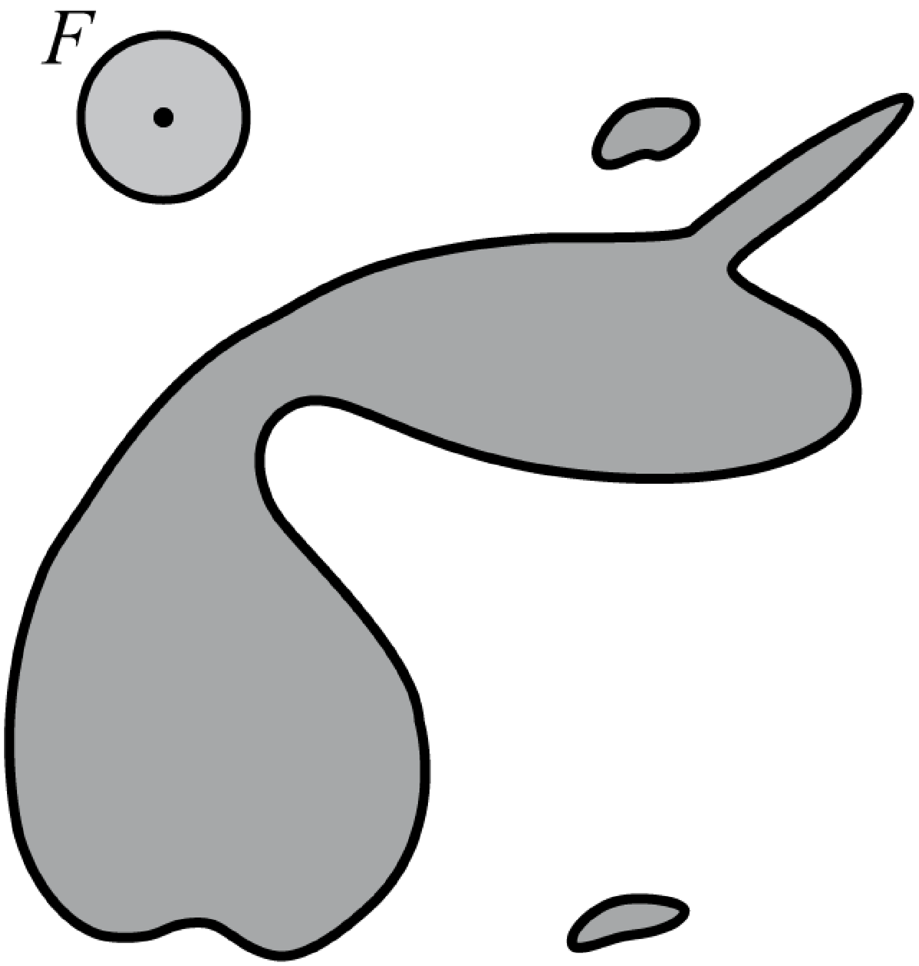
Erosion

Erosion using 3x3 structuring element of 1s.



Erosion

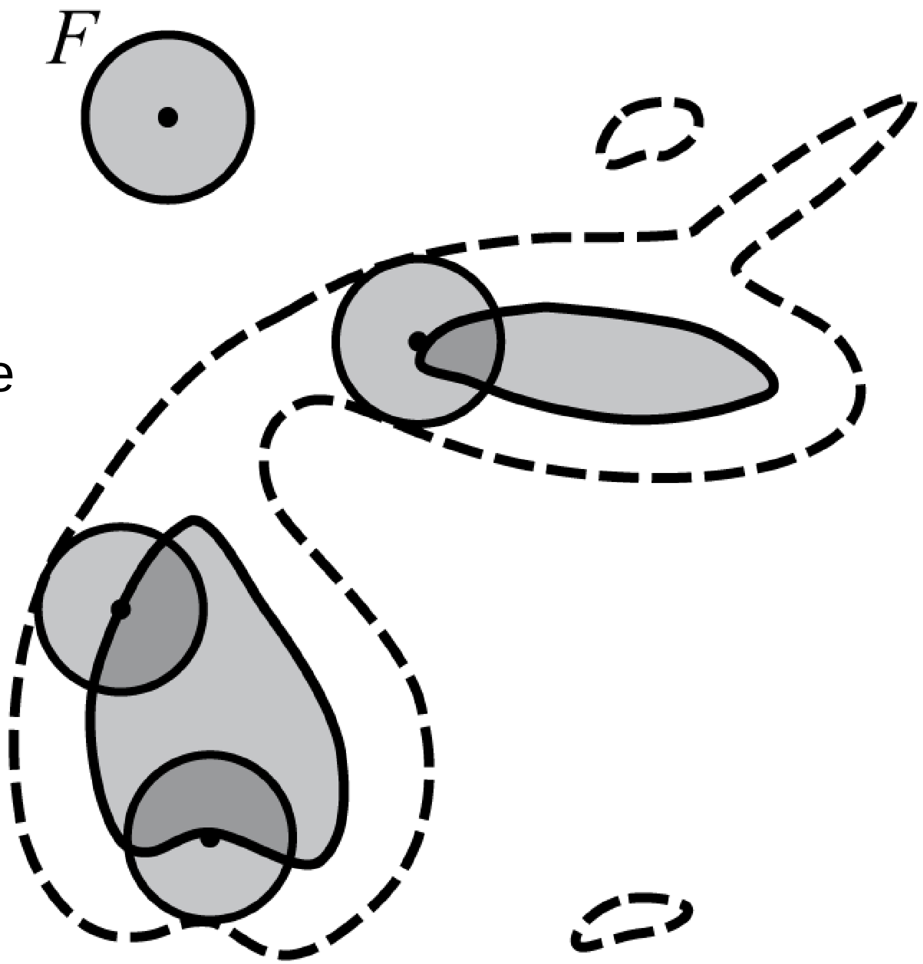
Input



Erosion

Erosion using structuring element *F*:

Roll wheel on the inside of the figure, axis is a knife cutting away the outer layer.



2. Dilation

- Assign 1 to result pixel if *at least* one 1-element of S covers a 1-pixel of g, else assign 0:

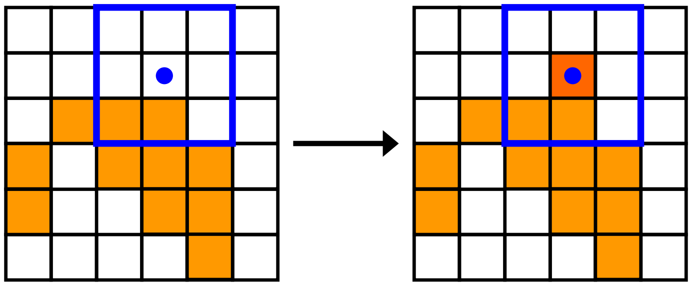
$$g'(x,y) = \Theta\left(\left(\sum_{k\in[-m,m]} \sum_{l\in[-n,n]} S(k+m,l+n) \cdot g(x+k,y+l)\right) - \tfrac{1}{2}\right)$$

- Alternative notation:

$$g'(x,y) = \bigvee_{k\in[-m,m]} \bigvee_{l\in[-n,n]} S(k+m,l+n) \wedge g(x+k,y+l)$$

- Short: $g' = g \oplus S$
- Dilation adds pixels at the fringe of objects and fills in holes

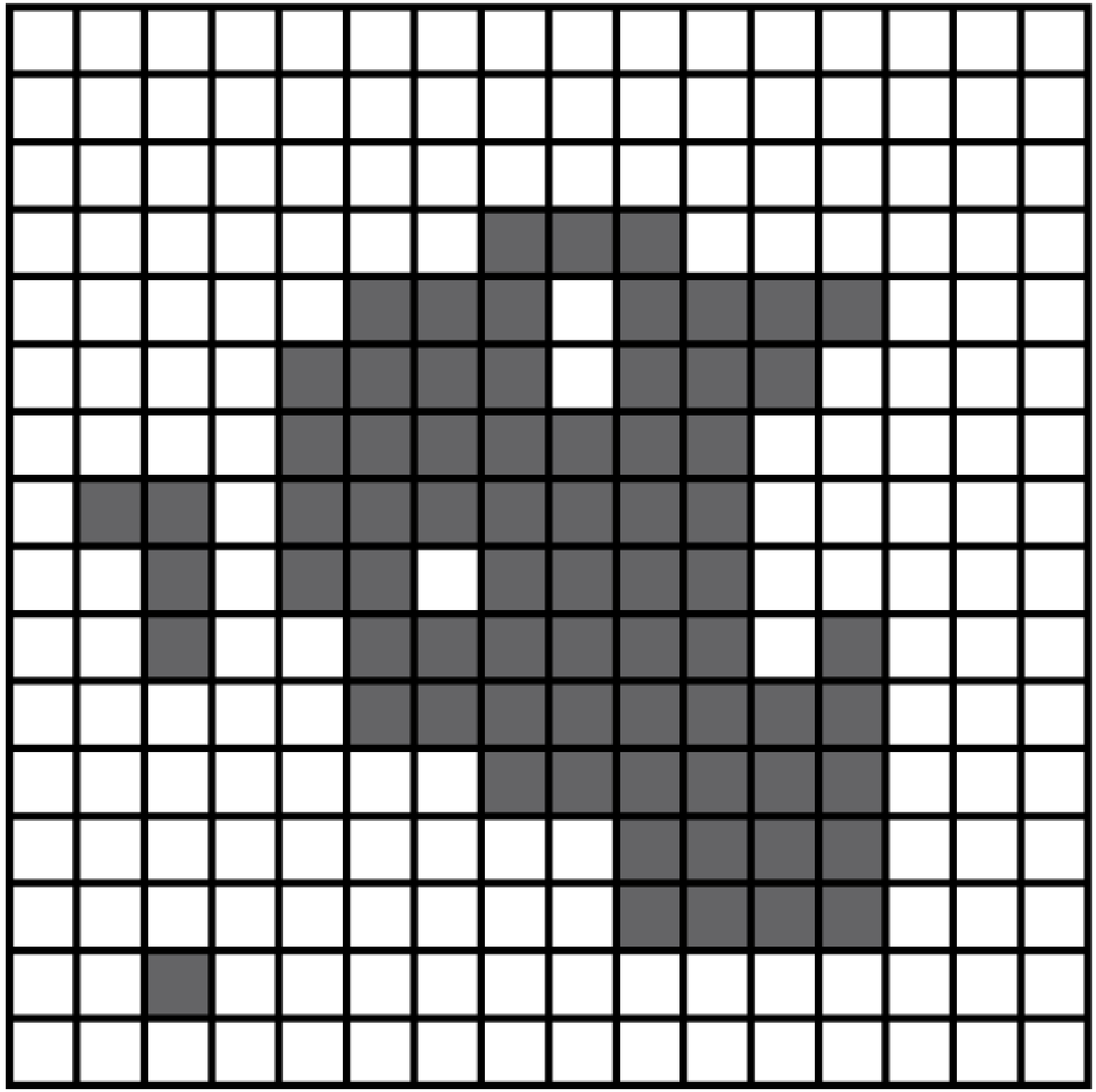
$$S = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad 1 = \blacksquare$$



[H]

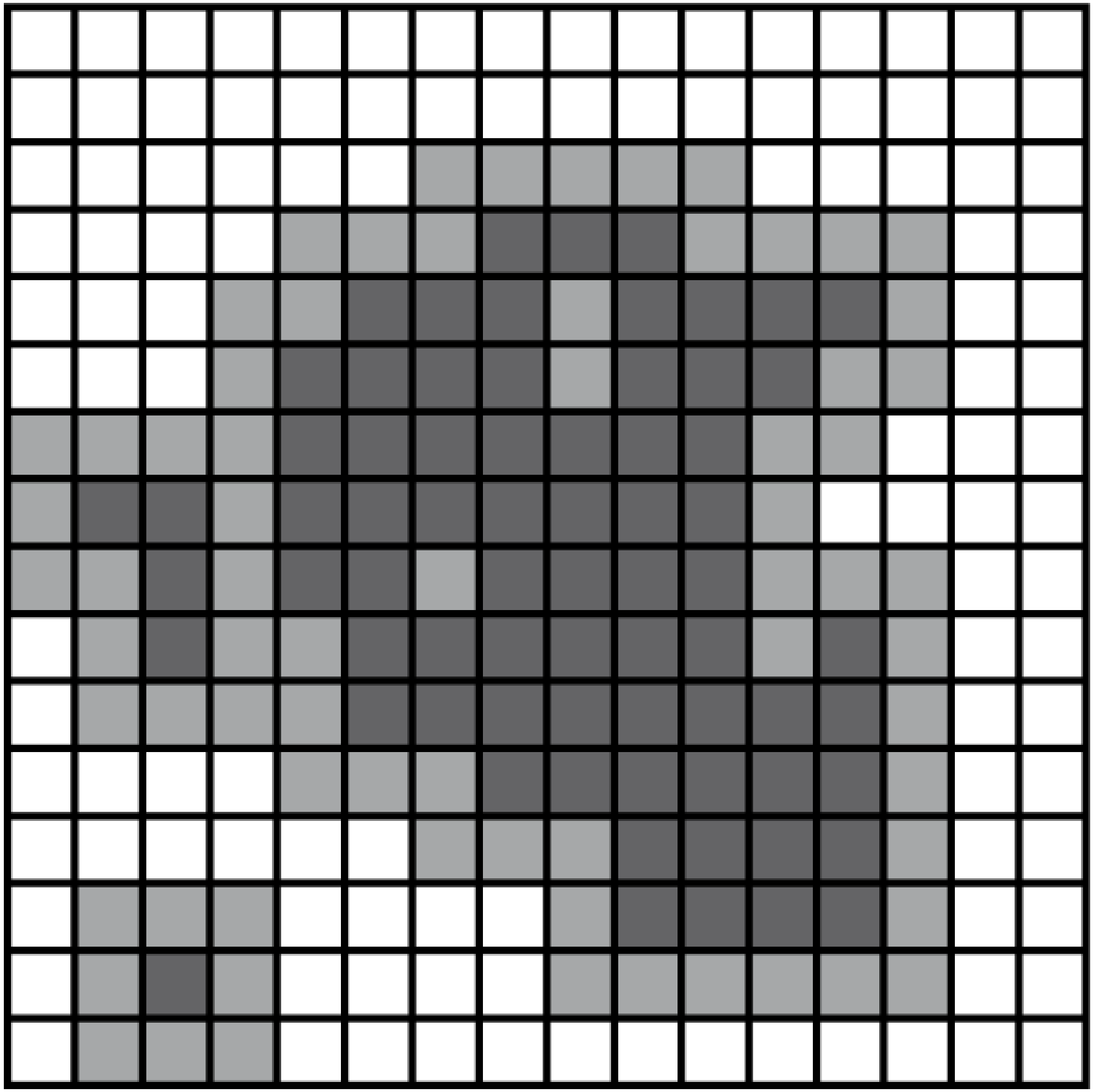
Dilation

Binary object



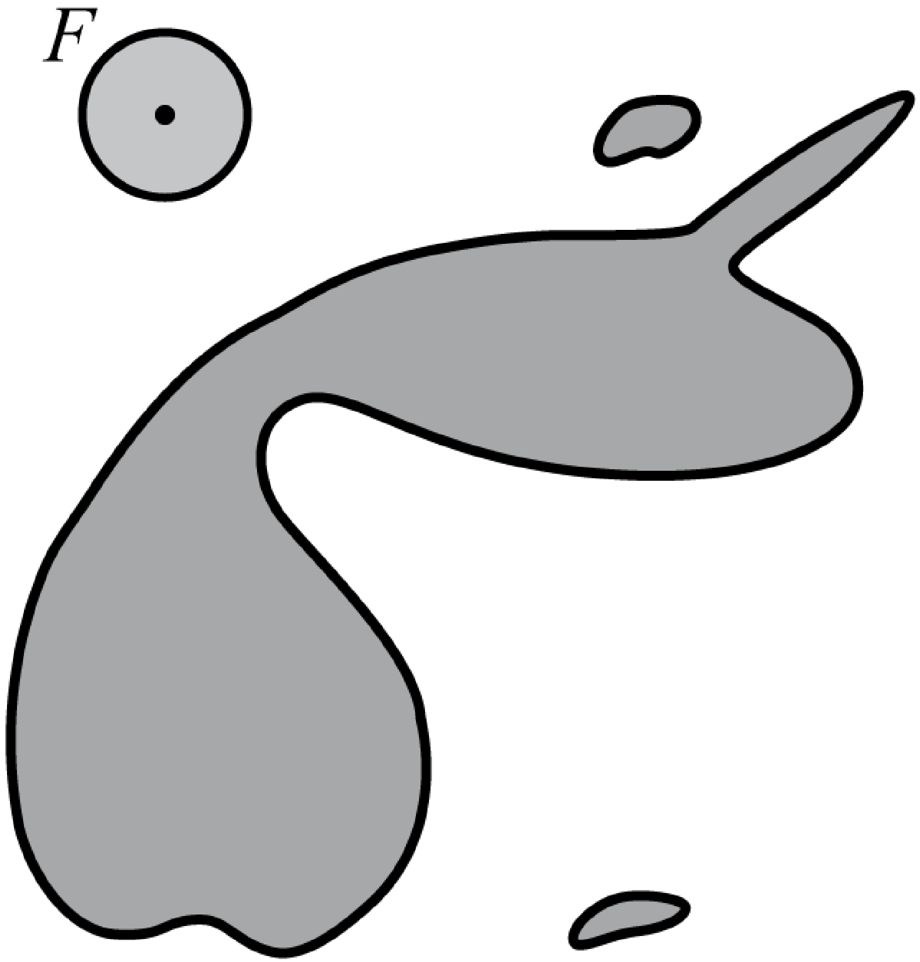
Dilation

Dilation using 3x3 structuring element of 1s



Dilation

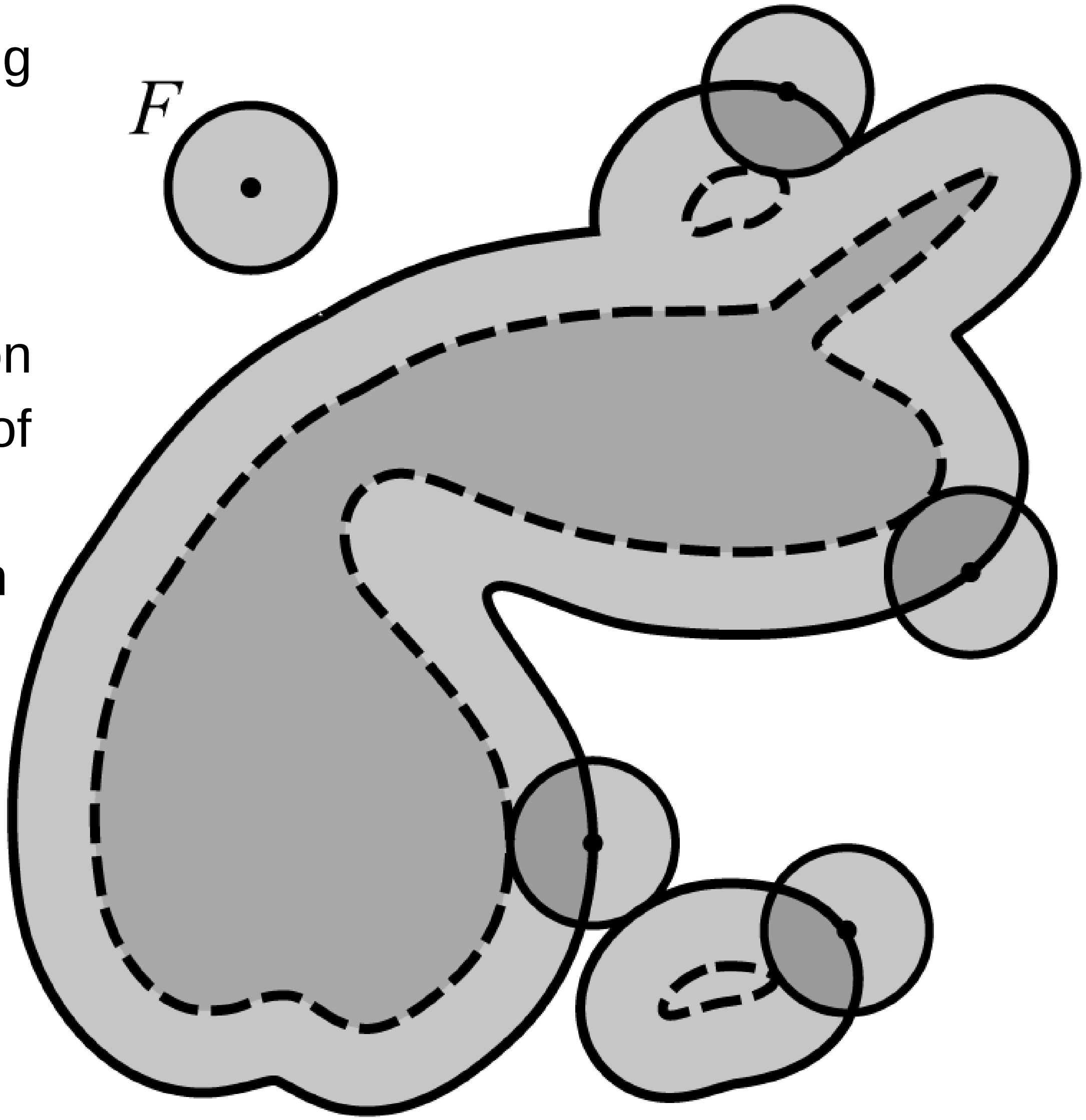
Input



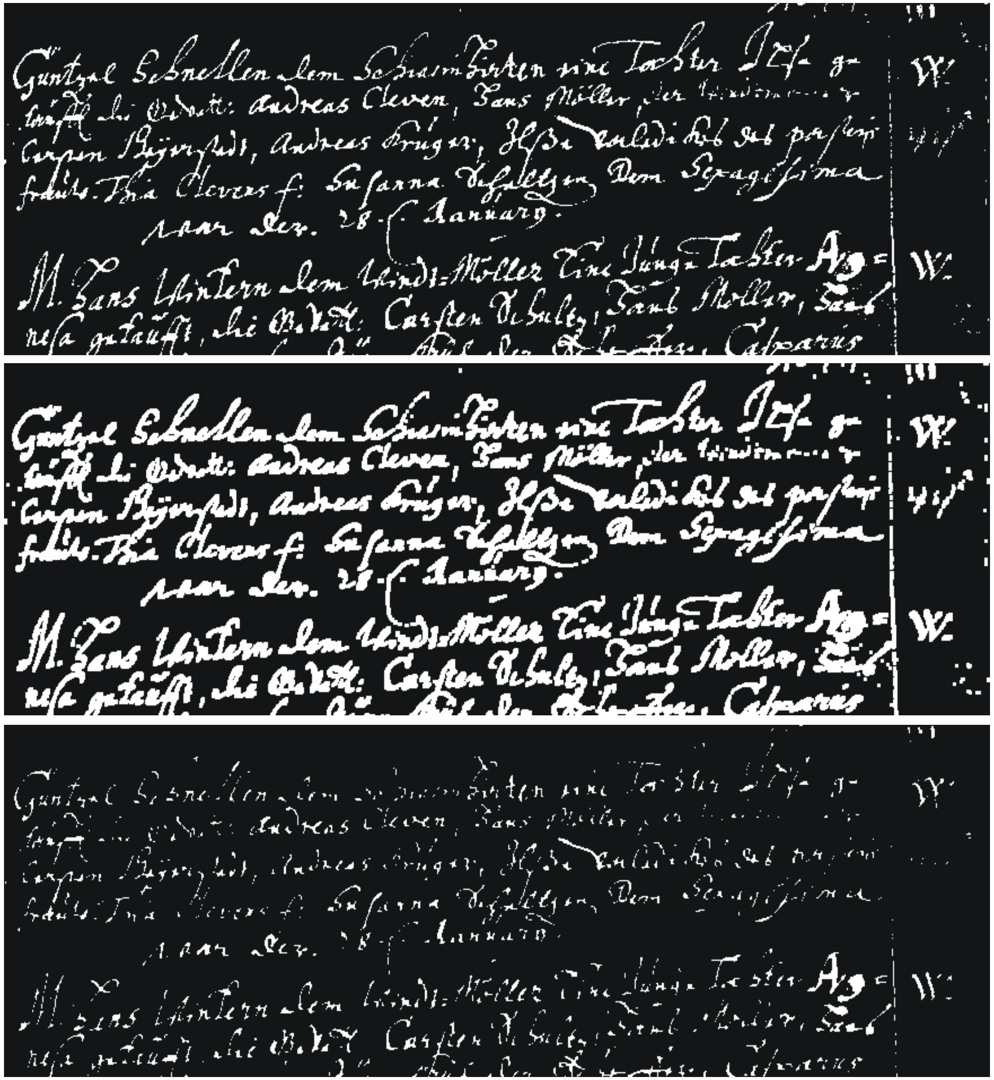
Dilation

Dilation using structuring element *F*:

Roll wheel on the outside of the figure, axis is a pen marking the new boundary.



Comparison erosion / dilation



[T]

Above: Input – Center: Dilation – Below: Erosion

Combining erosion and dilation

By choice of *S* selected structures can be removed or filled up.

- Problems:
  1. Erosion removes not only irregularities but also a smooth fringe, making objects smaller
  2. Dilation enlarges objects
- Solution: Combine both
  1. Opening = Erosion using *S* + dilation using *S´*:

     $$g' = (g \ominus S) \oplus S'$$

     E.g., removes irregularities at object boundary without making object smaller
  2. Closing = Dilation using *S* + erosion using *S´*:

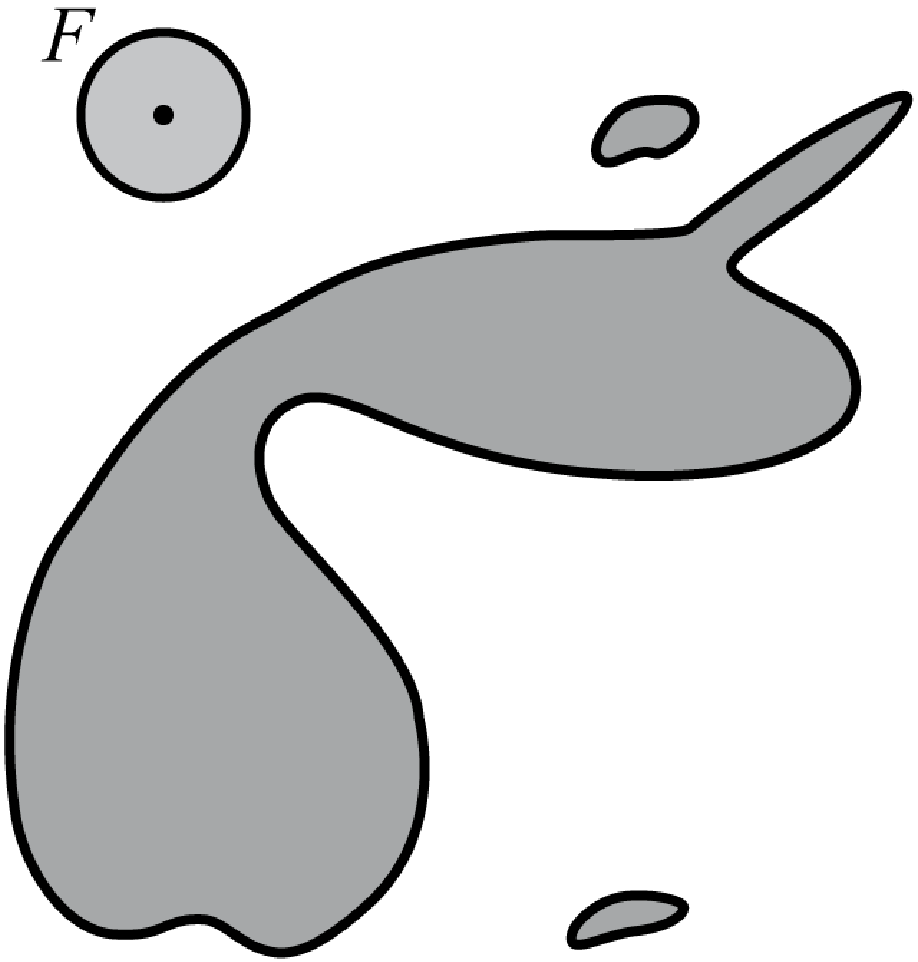
     $$g' = (g \oplus S) \ominus S'$$

     E.g., fills in holes without making object larger

S´ is a version of the structuring element mirrored at the anchor point

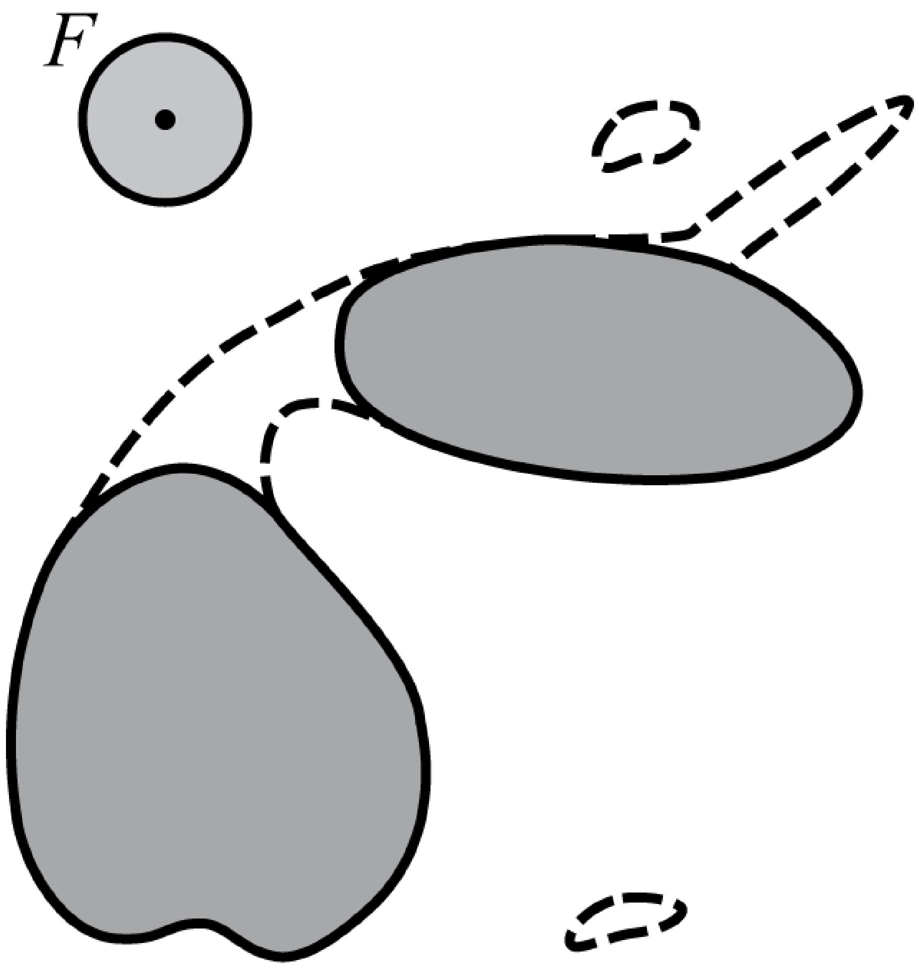
Opening

Input



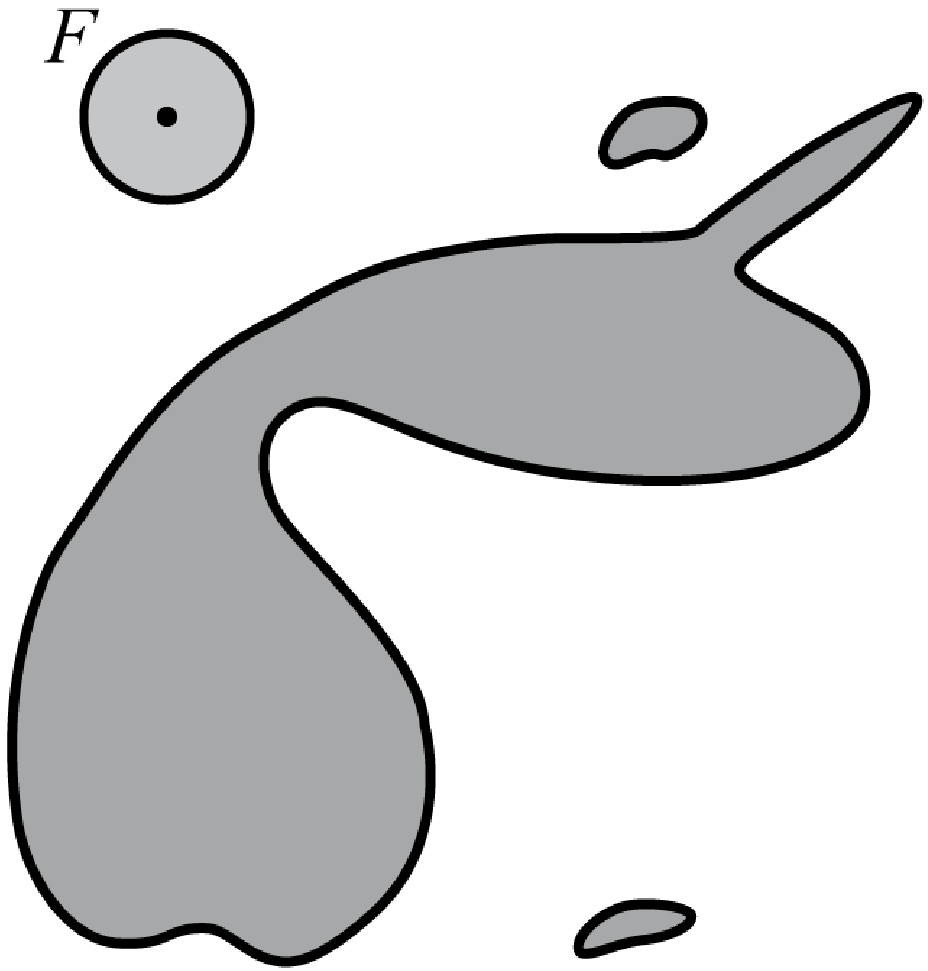
Opening

Opening using structuring element *F*



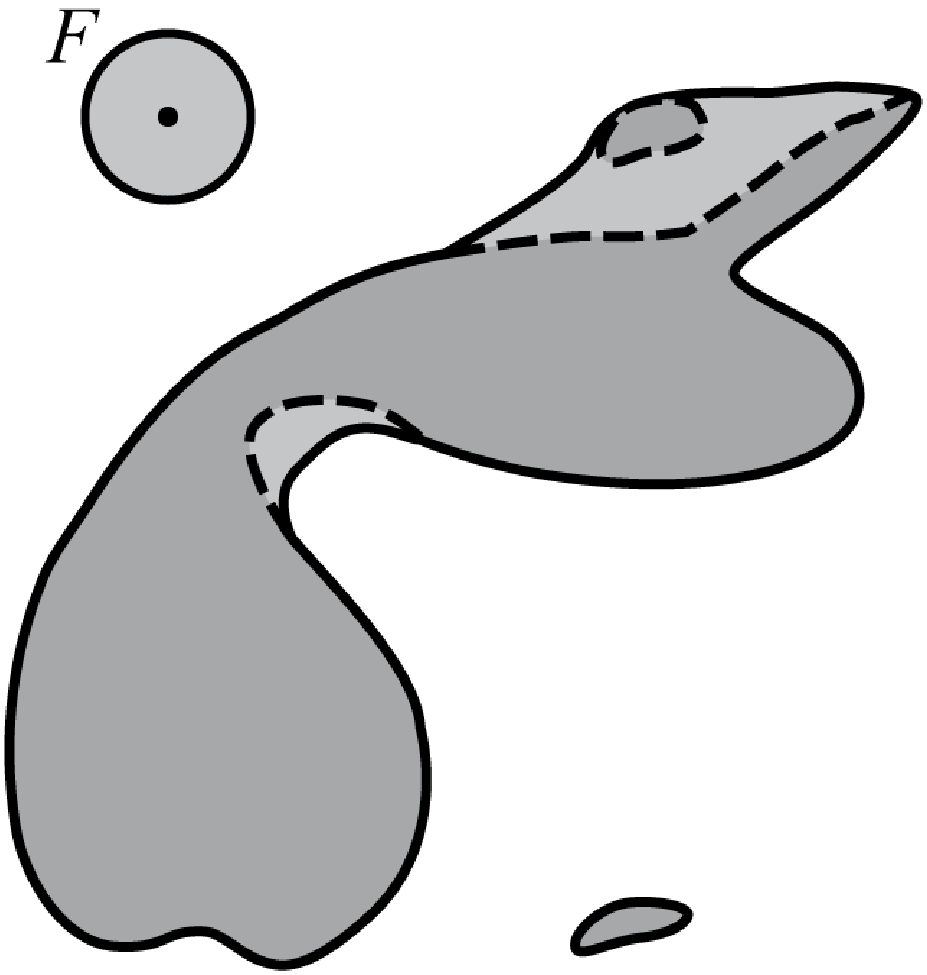
Closing

Input

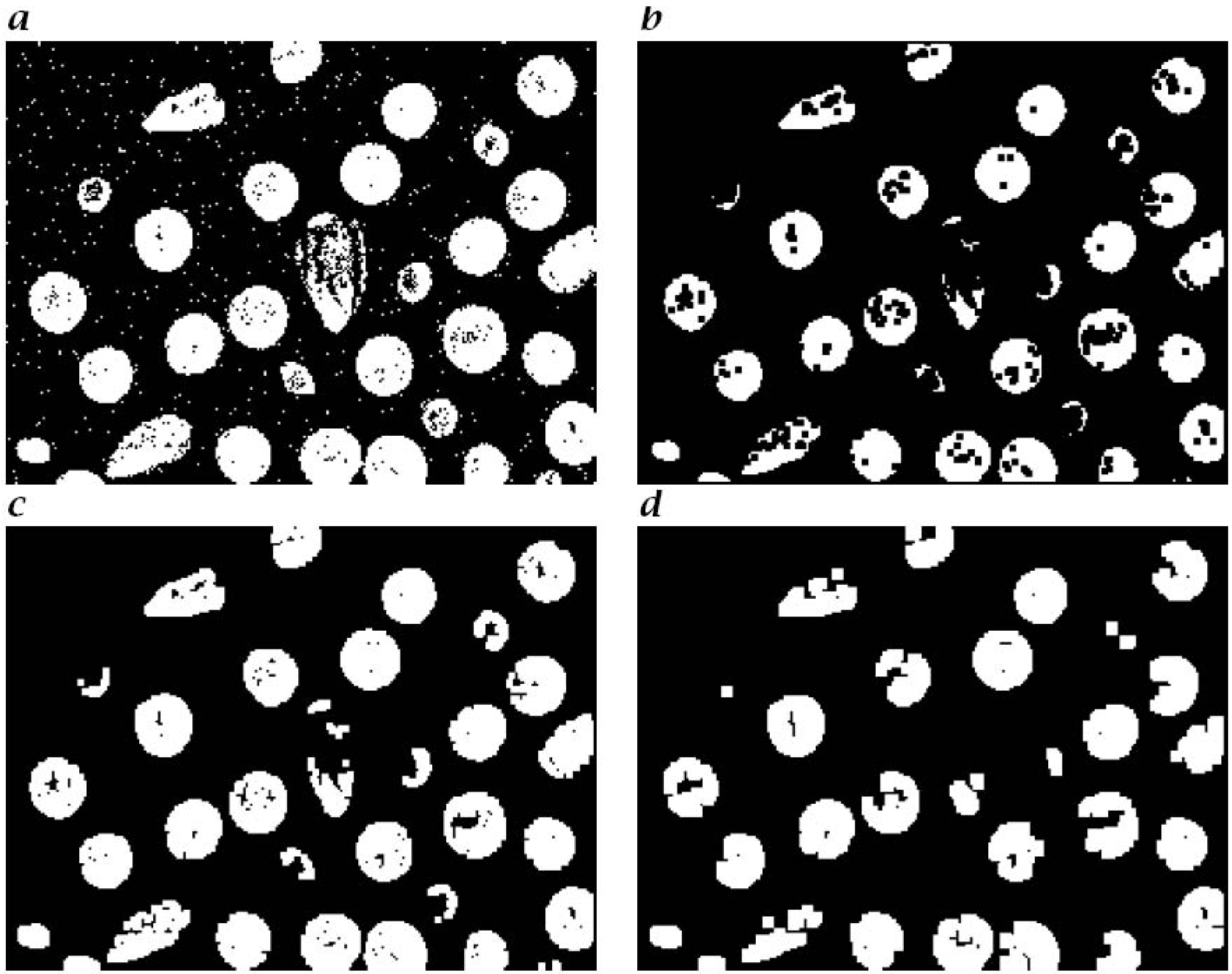


Closing

Closing using structuring element *F*



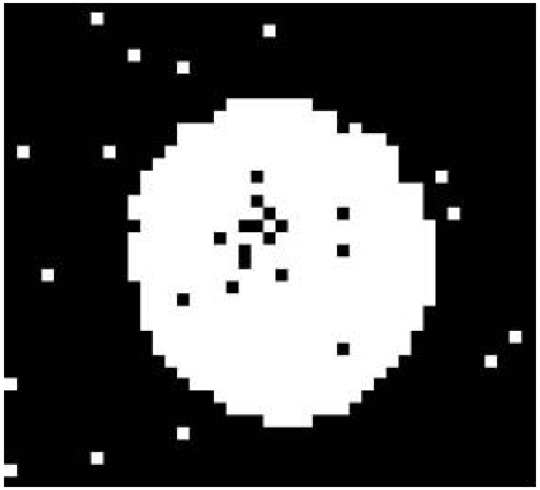
Examples



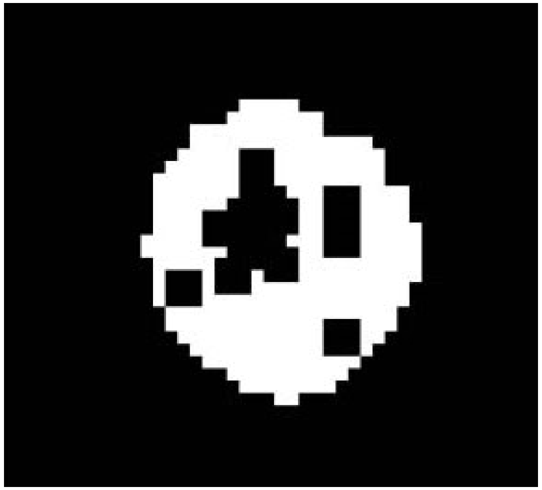
[J]

(a) Input, (b) Erosion using 3x3 *S*,
(c) Opening using 3x3 *S*, (d) Opening using 5x5 *S*

Examples

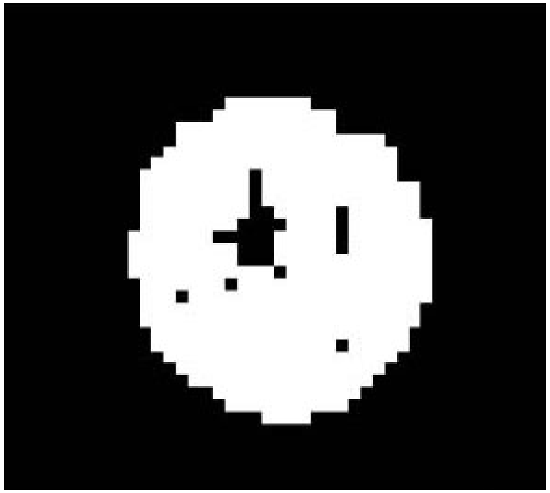


Input



Erosion using 3x3 *S*

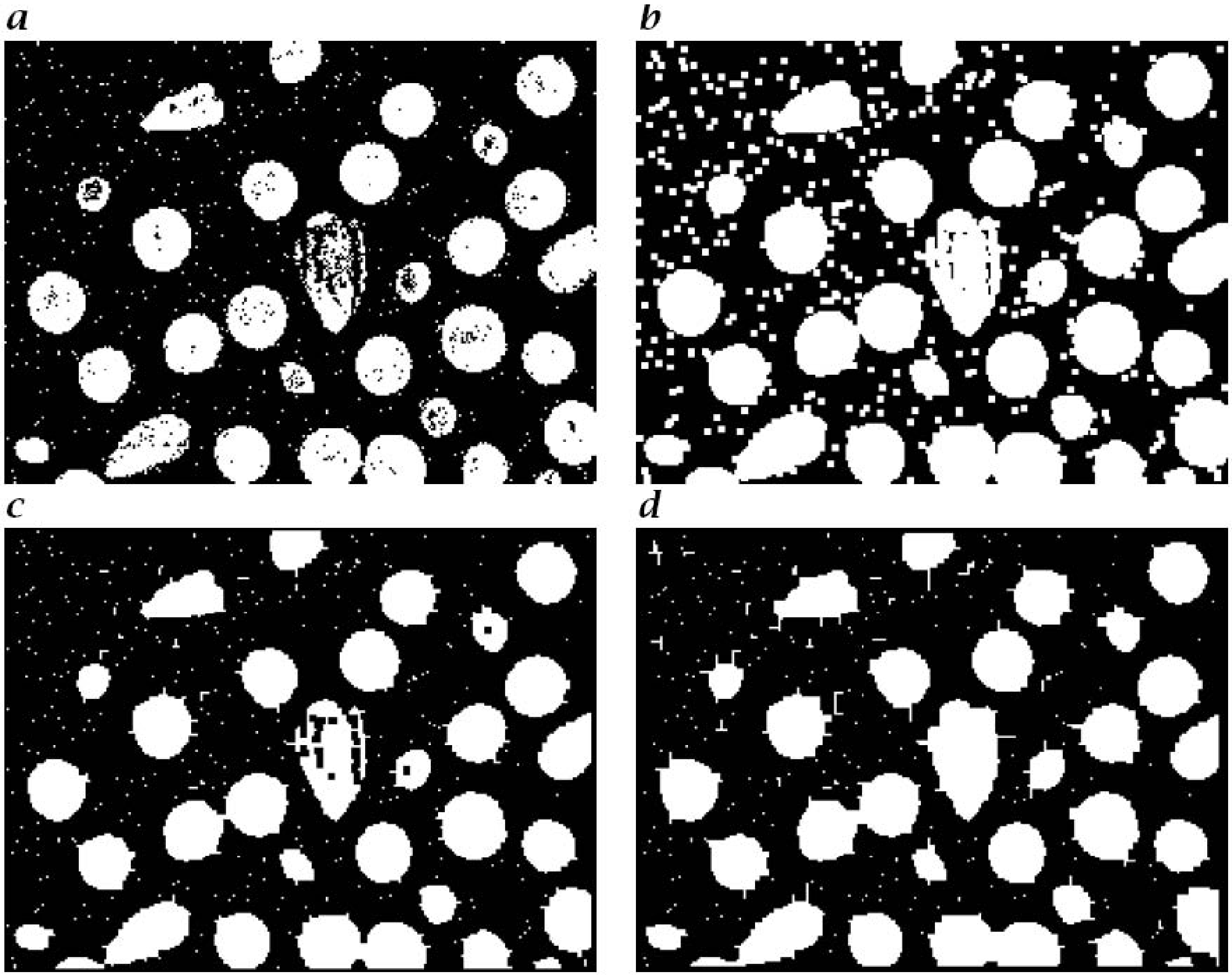
Opening using 5x5 *S*



Opening using 3x3 *S*

[J]

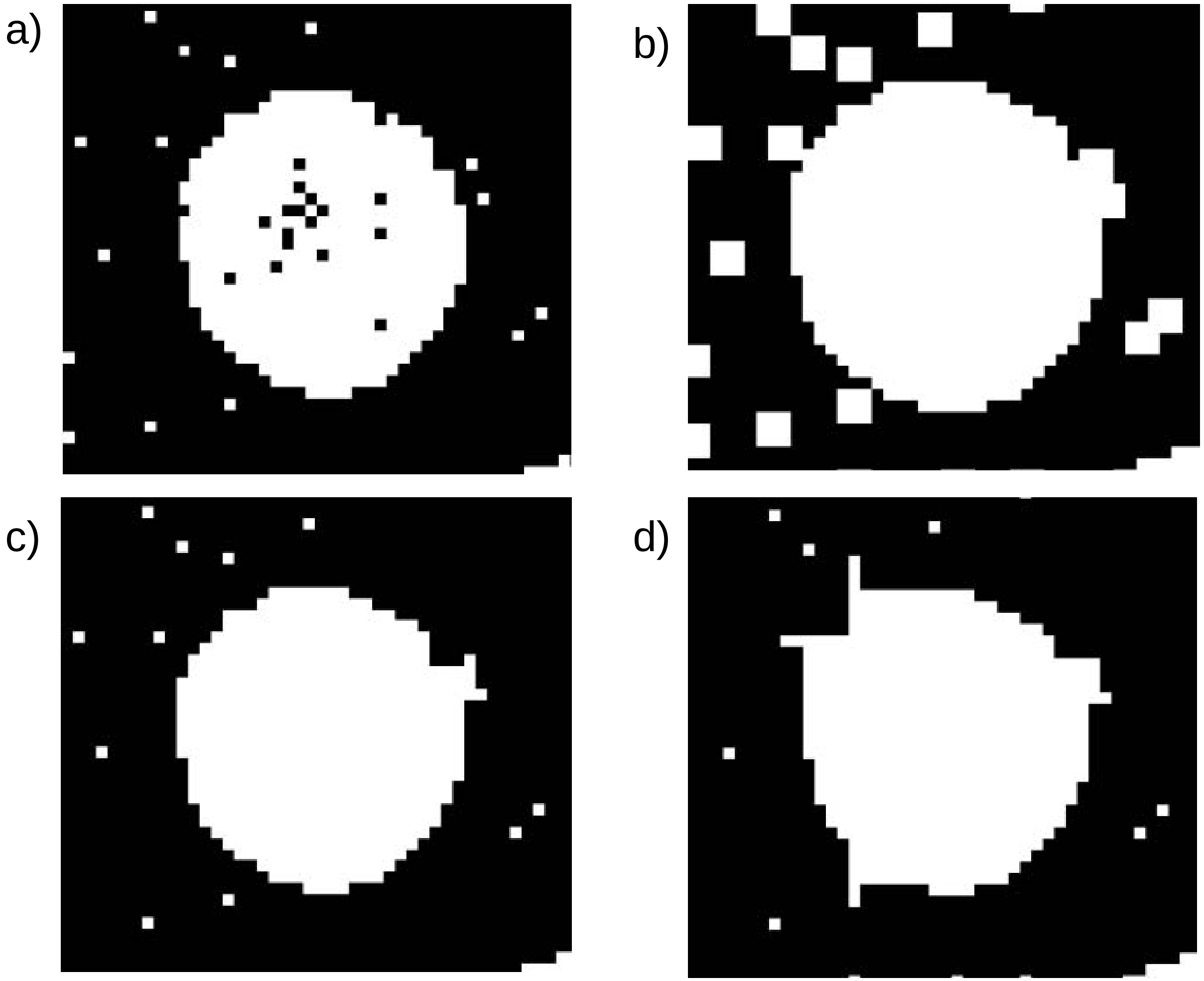
Examples



[J]

(a) Input,
(b) Dilation using 3x3 *S*,
(c) Closing using 3x3 *S*,
(d) Closing using 5x5 *S*

Examples



[J]

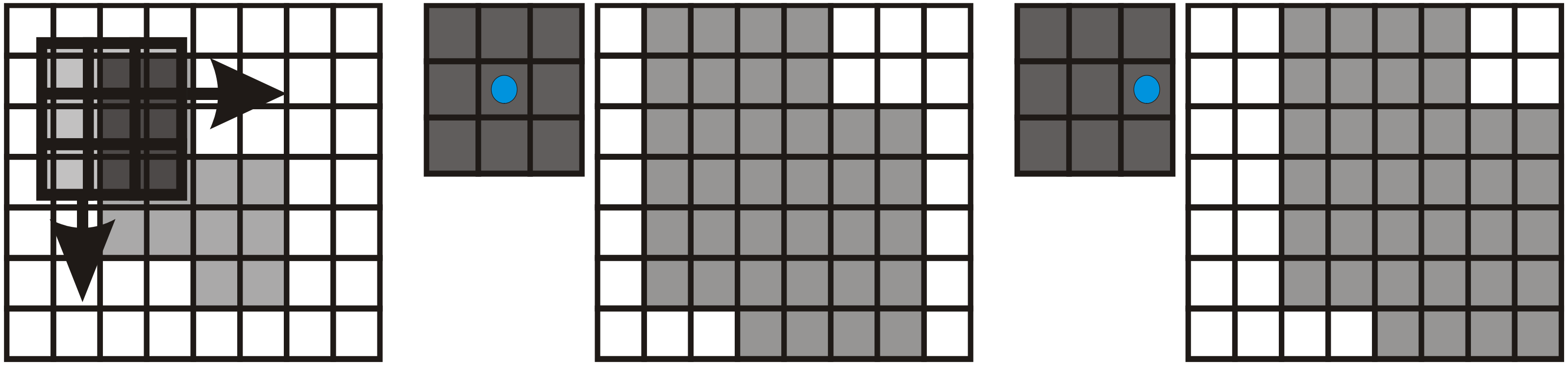
(a) Input, (b) Dilation using 3x3 *S*,
(c) Closing using 3x3 *S*, (d) Closing using 5x5 *S*

Customized structuring elements

- So far, we have used only “unstructured“ structuring elements, i.e., *nxn* structuring elements filled with 1.
- So far, the center was the anchor point.
- Useful as long as
  - there is no a priori knowledge about the image,
  - no special tasks are to be solved.
- Using specialized structuring elements it is possible
  - to fill gaps of a particular shape,
  - to detect patterns (hit-or-miss operation).

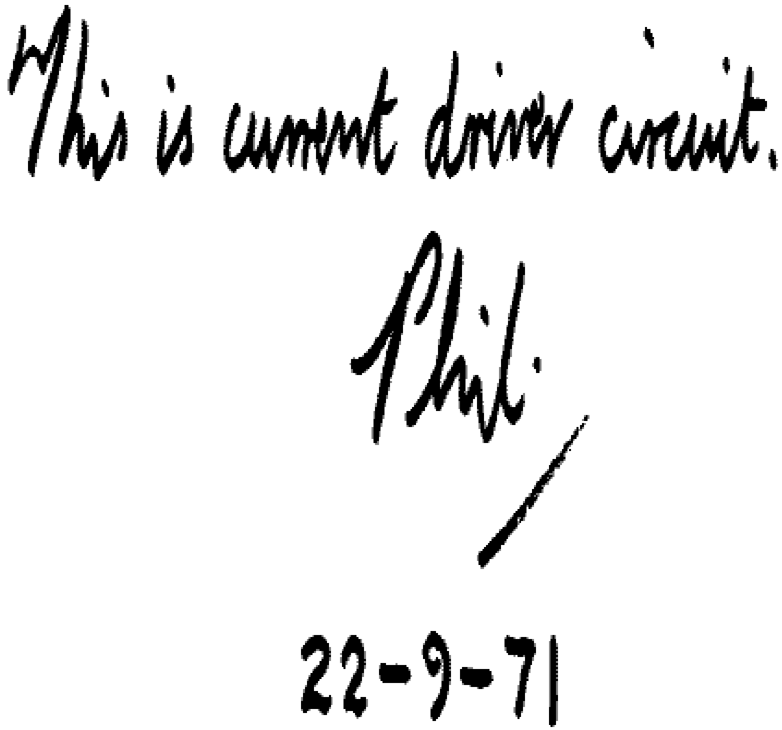
Customized structuring elements: Anchor point

Note the anchor point needs not to be in the center of the structuring element:



[T]

Customized structuring elements

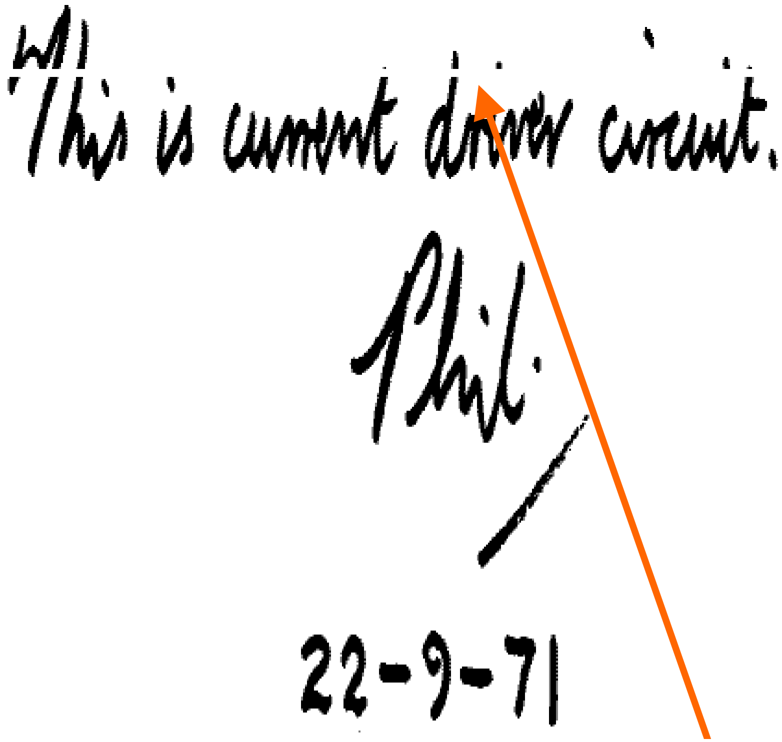


Undisturbed binary image

[T]

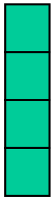
Special problems can be solved using structuring elements of a customized shape

Customized structuring elements



Erased stripe

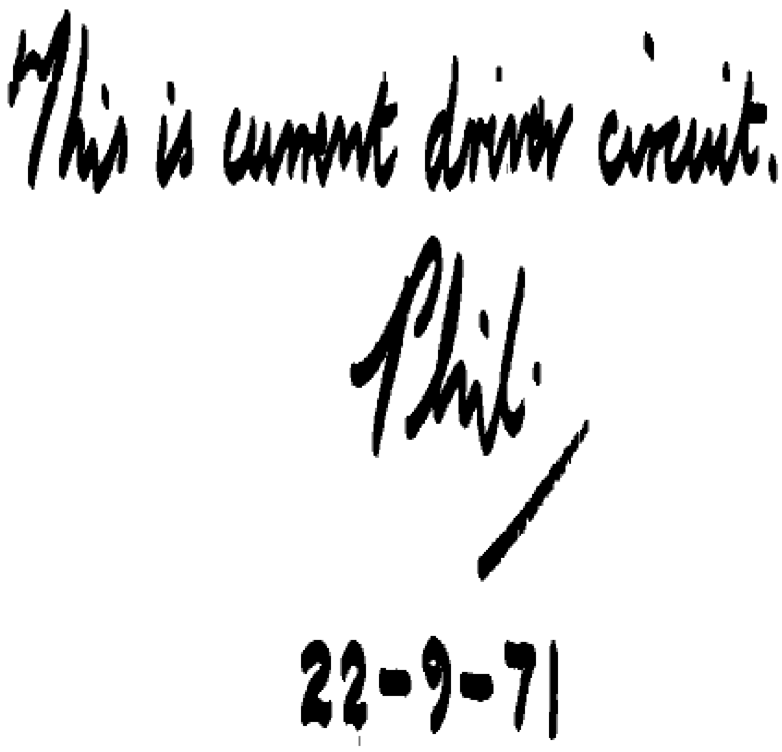
Special problems can be solved using structuring elements of a customized shape



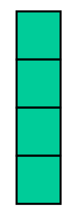
Structuring element to fill in the erased stripe

[T]

Customized structuring elements



Special problems can be solved using structuring elements of a customized shape

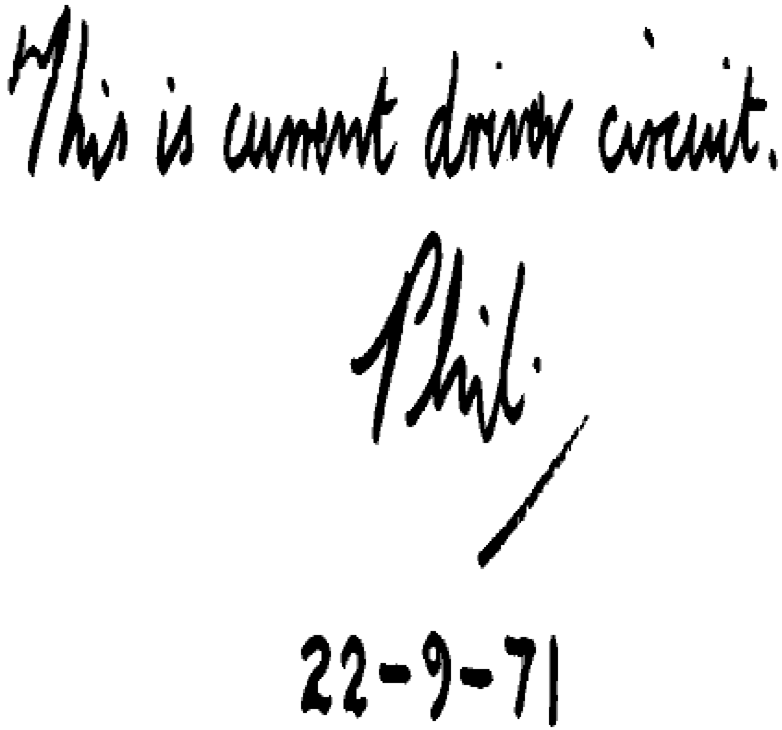


Structuring element to fill in the erased stripe

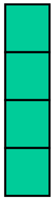
Result after dilation:
Stripe closed

[T]

Customized structuring elements



Special problems can be solved using structuring elements of a customized shape



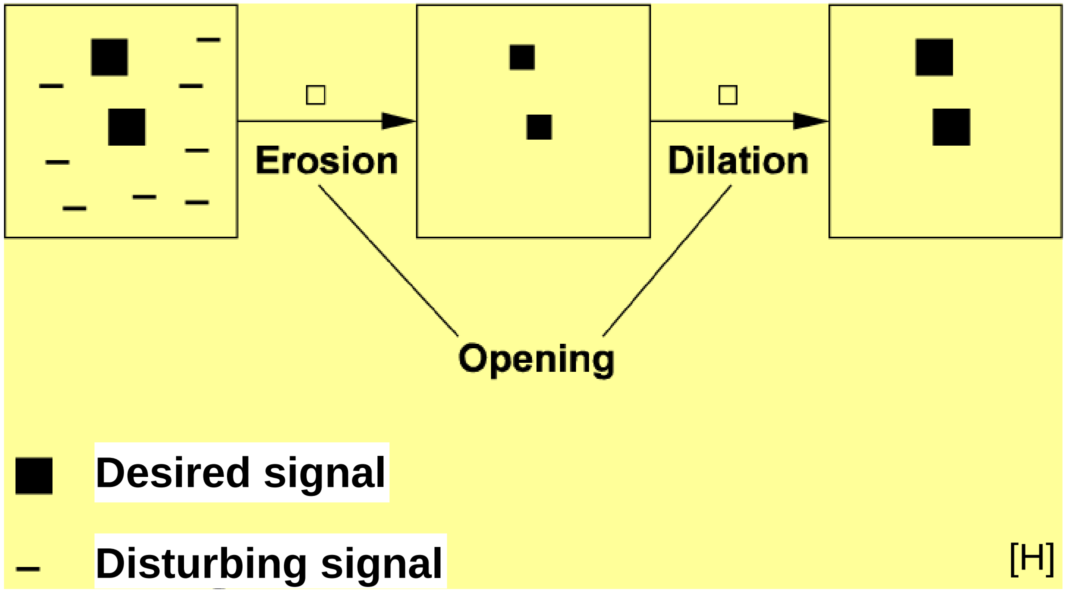
Structuring element to fill in the erased stripe

Result after subsequent erosion:
Writing has original size

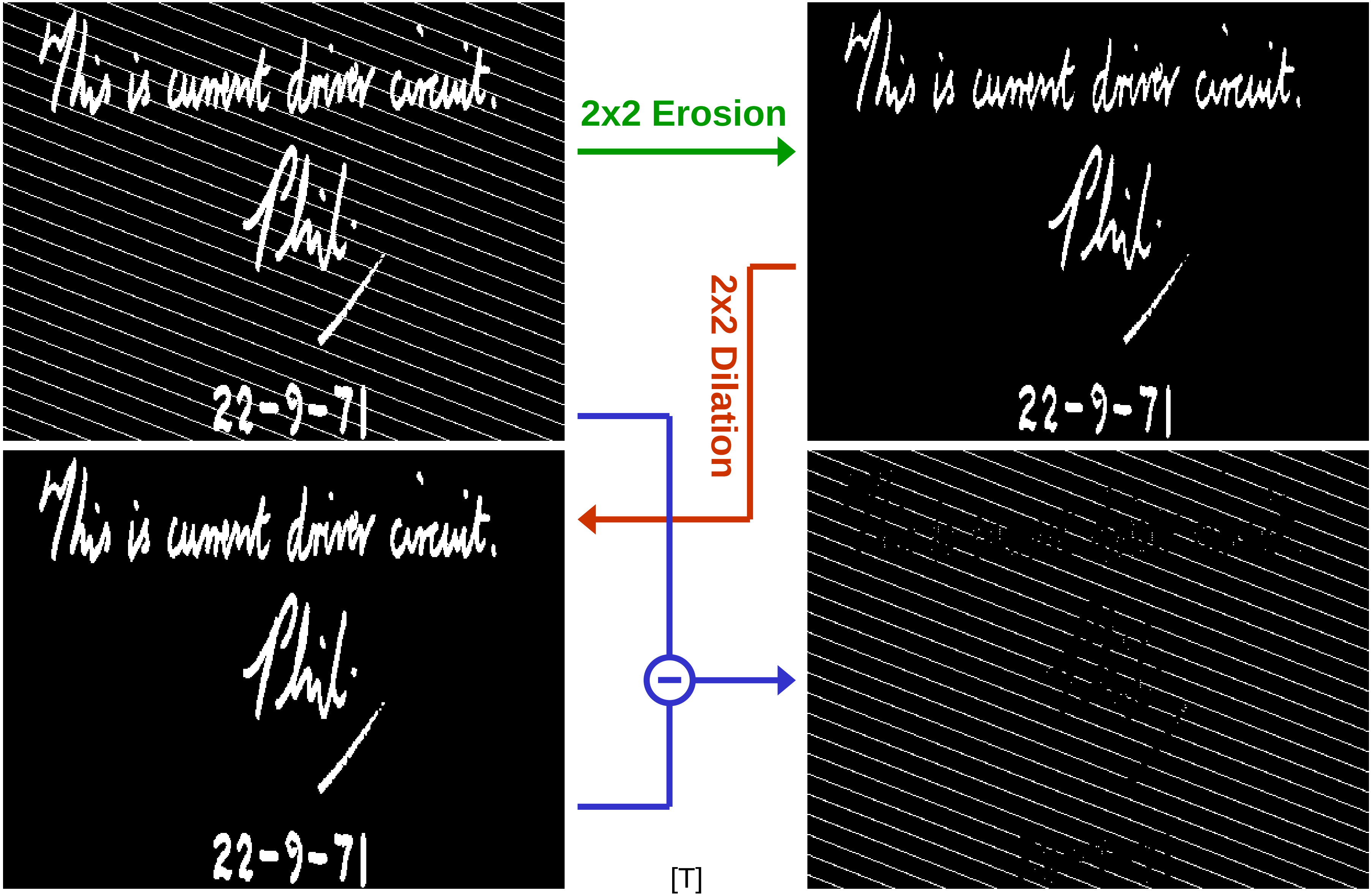
[T]

Application of morphological operations

Separation of desired signal and disturbing signal:

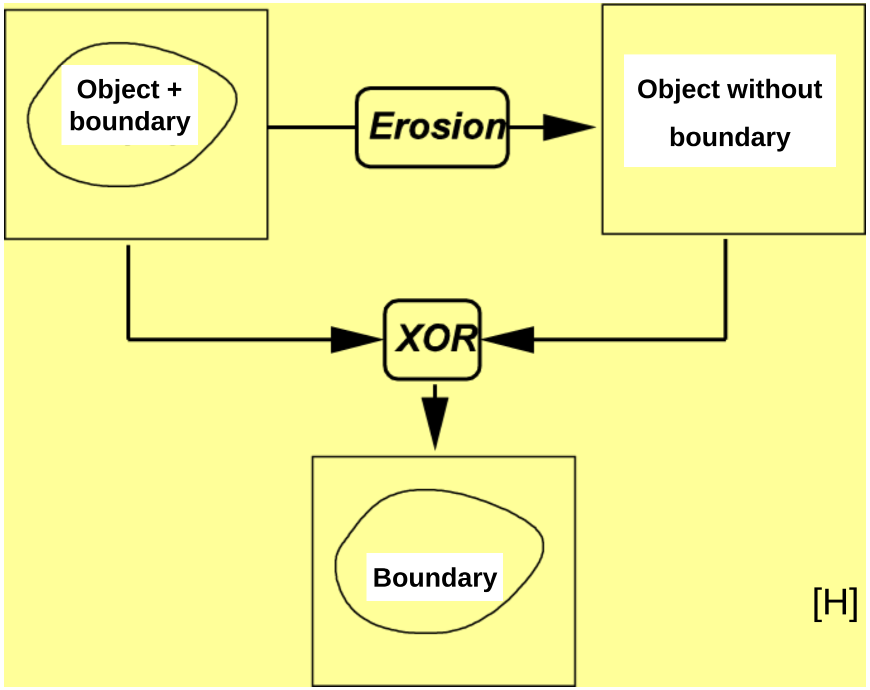


Example: Removal / extraction of lines



Boundary extraction

Boundary extraction: $g_{boundary} = g \setminus (g \ominus S)$



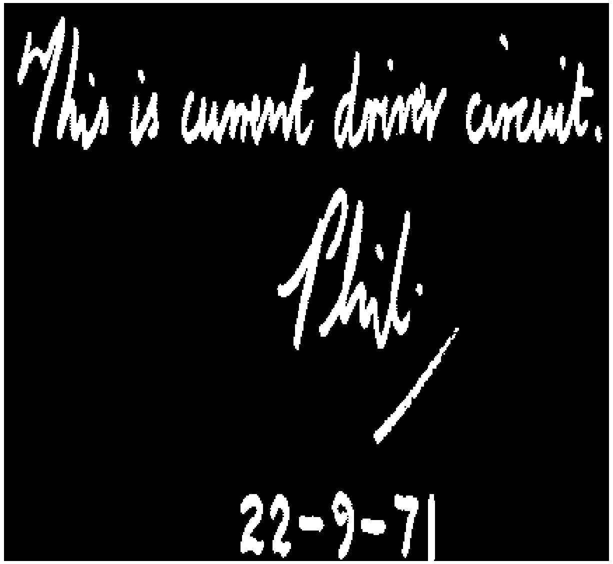
Structuring elements for 4- and 8-neighborhood:

$$S_4 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

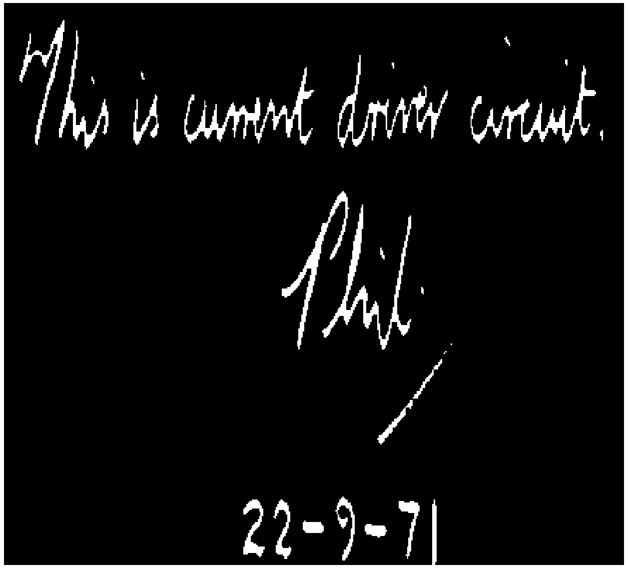
$$S_8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

S_4 and S_8 cut off all object pixels in the 4- or 8-neighborhood of which are background pixels.

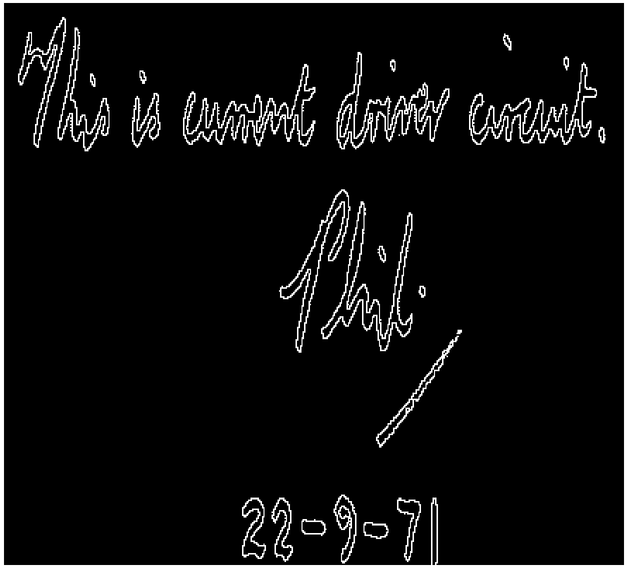
Boundary extraction



g



$g \ominus S$



$g \setminus (g \ominus S)$

[T]

Distance transform

- One-time application of operation $g^0_{boundary} = g \setminus (g \ominus S)$ yields the boundary, i.e., the set of all pixels with distance 0 to the boundary
- Object without boundary: $g \ominus S$
- Application of this operation to $g \ominus S$ yields $g^1_{boundary}$, i.e., the set of all pixels, with distance 1 to the boundary:

$$g^1_{boundary} = (g \ominus S) \setminus (g \ominus S \ominus S).$$

- In general:

$$g^n_{boundary} = \big(g\ (\ominus S)^n\big) \setminus \big(g\ (\ominus S)^{(n+1)}\big)$$

where $(\ominus S)^n$ is short for the n-time erosion using S.

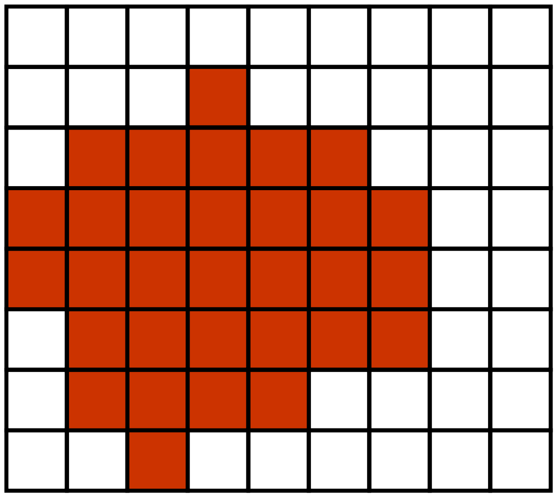
Distance transform

- The distance image D is obtained from the union of the boundaries of all distances. In D, the boundary pixels of distance i get gray value i:

$$D = \bigcup_{n=1\ldots\text{"}\infty\text{"}} n \cdot g^n_{boundary}$$

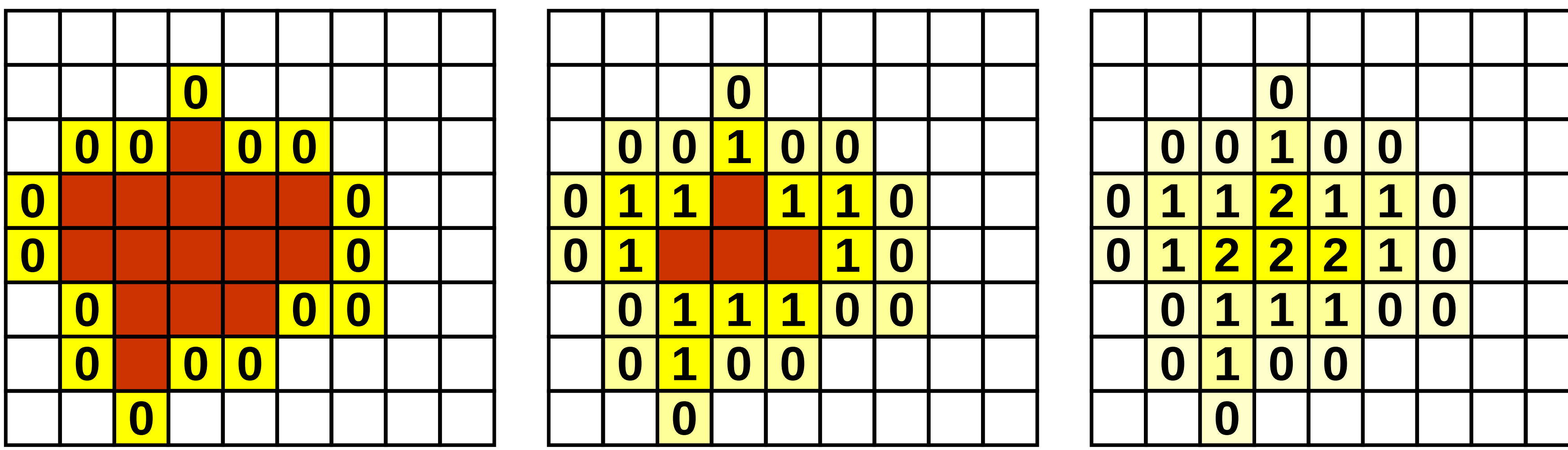
  where "$n\,\cdot$" denotes pixel-wise multiplication, provided $g^n_{boundary}$ is a binary image with 1 for the boundary.
- Each pixel in the distance image indicates the distance to the boundary, however,
  - not the Euclidean distance, but
  - the cityblock- or chessboard-distance
- Problem: Large computational effort
- There are faster algorithms to compute the Euclidean distance transform.

Distance transform



Remaining inside of object

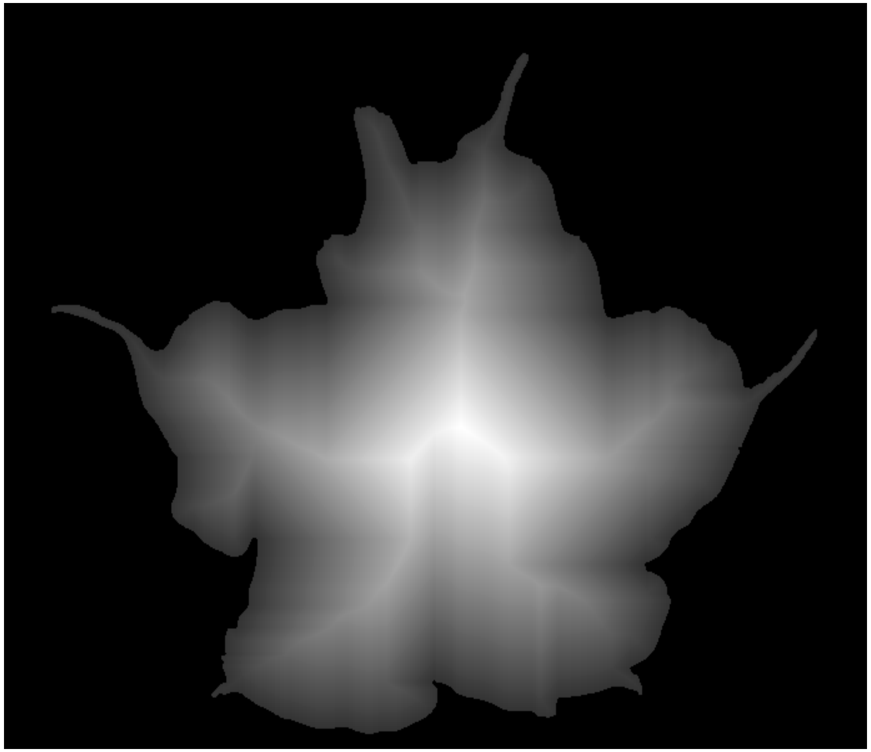
n Pixel with distance *n* to the boundary



[H]

Distance transform

Binary image



Distance transform

[T]

Distance transform

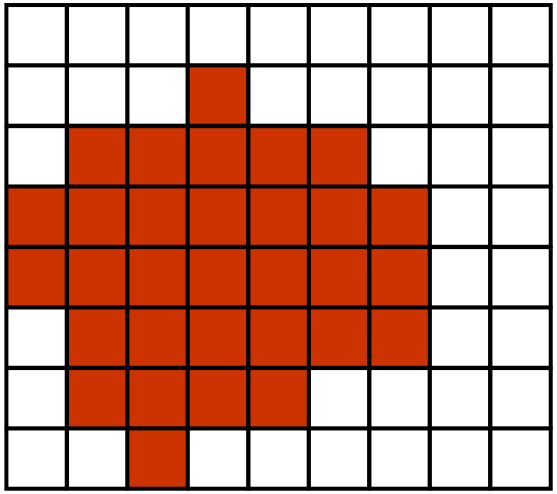
Generalize distance transform beyond the boundary of the object:

1. Input: Binary image *b*. Compute inverted binary image *b**.
2. Compute D_1 from *b*
3. Compute D_2 from *b**, i.e., the distances of the background pixels to the boundary of the object
4. Generalized distance transform:

$$D = D_1 - D_2$$

5. Normalization to 0…255 for visualization

Distance transform



			0					
	0	0		0	0			
0						0		
0						0		
	0				0	0		
	0		0	0				
		0						

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

Object

0 Boundary

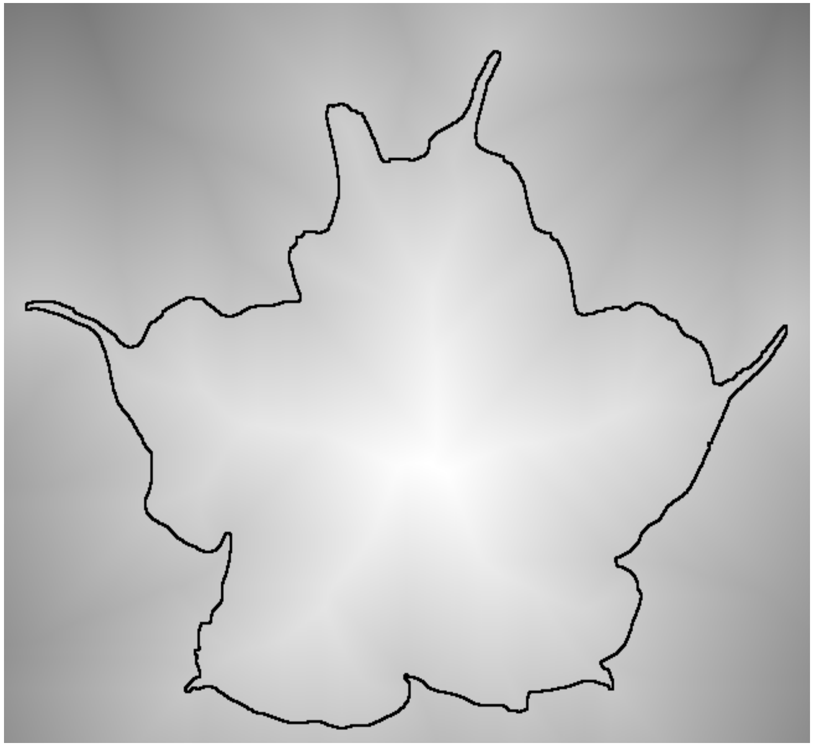
			0					
	0	0	1	0	0			
0	1	1	2	1	1	0		
0	1	2	2	2	1	0		
	0	1	1	1	0	0		
	0	1	0	0				
		0						

-3	-2	-2	-1	-2	-2	-3	-4	-5
-2	-1	-1	0	-1	-1	-2	-3	-4
-1	0	0	1	0	0	-1	-2	-3
0	1	1	2	1	1	0	-1	-2
0	1	2	2	2	1	0	-1	-2
-1	0	1	1	1	0	0	-1	-2
-1	0	1	0	0	-1	-1	-2	-3
-2	-1	0	-1	-1	-2	-2	-3	-4

[H]

Distance transform

[T]



Binary image

Generalized distance transform with normalized gray values

Morphing

- Task: Morphing from binary image b_a to b_b.
- Generalized distance transform provides a solution:

1. Compute distance transforms D_a und D_b of b_a and b_b.
2. Compute the $N-1$ intermediate distance images D_i between D_a and D_b by linear interpolation:

$$D_i = \big(i \cdot D_b + (N-i)\, D_a\big) \;/\; N \quad \text{with} \quad i = 0 \ldots N,$$

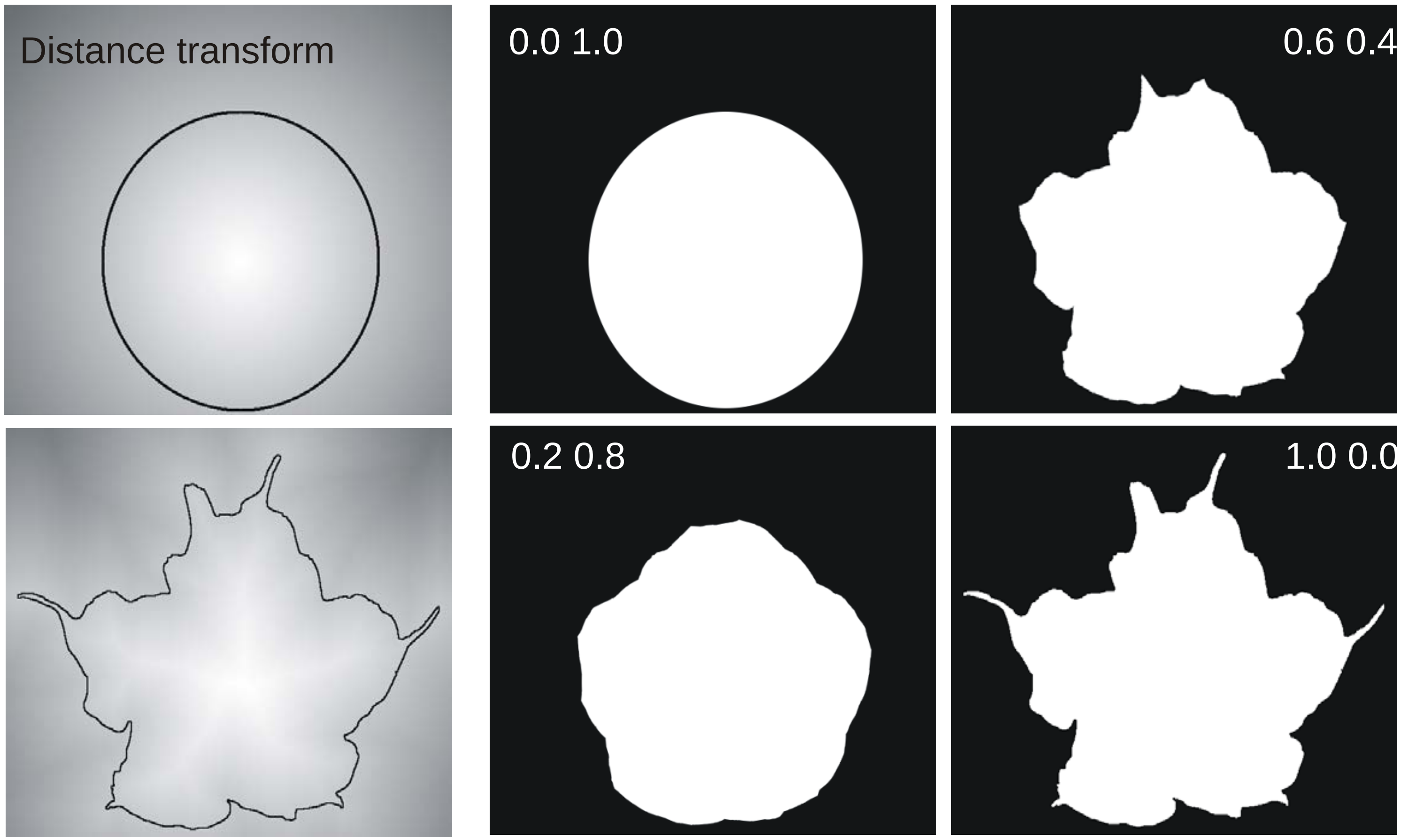
where $D_a = D_0$ and $D_b = D_N$.

3. Binarize the intermediate images to obtain b_i:

$$b_i(x,y) = \Theta(D_i\,(x,y)),$$

with $b_a = b_0$ and $b_b = b_N$.

Morphing



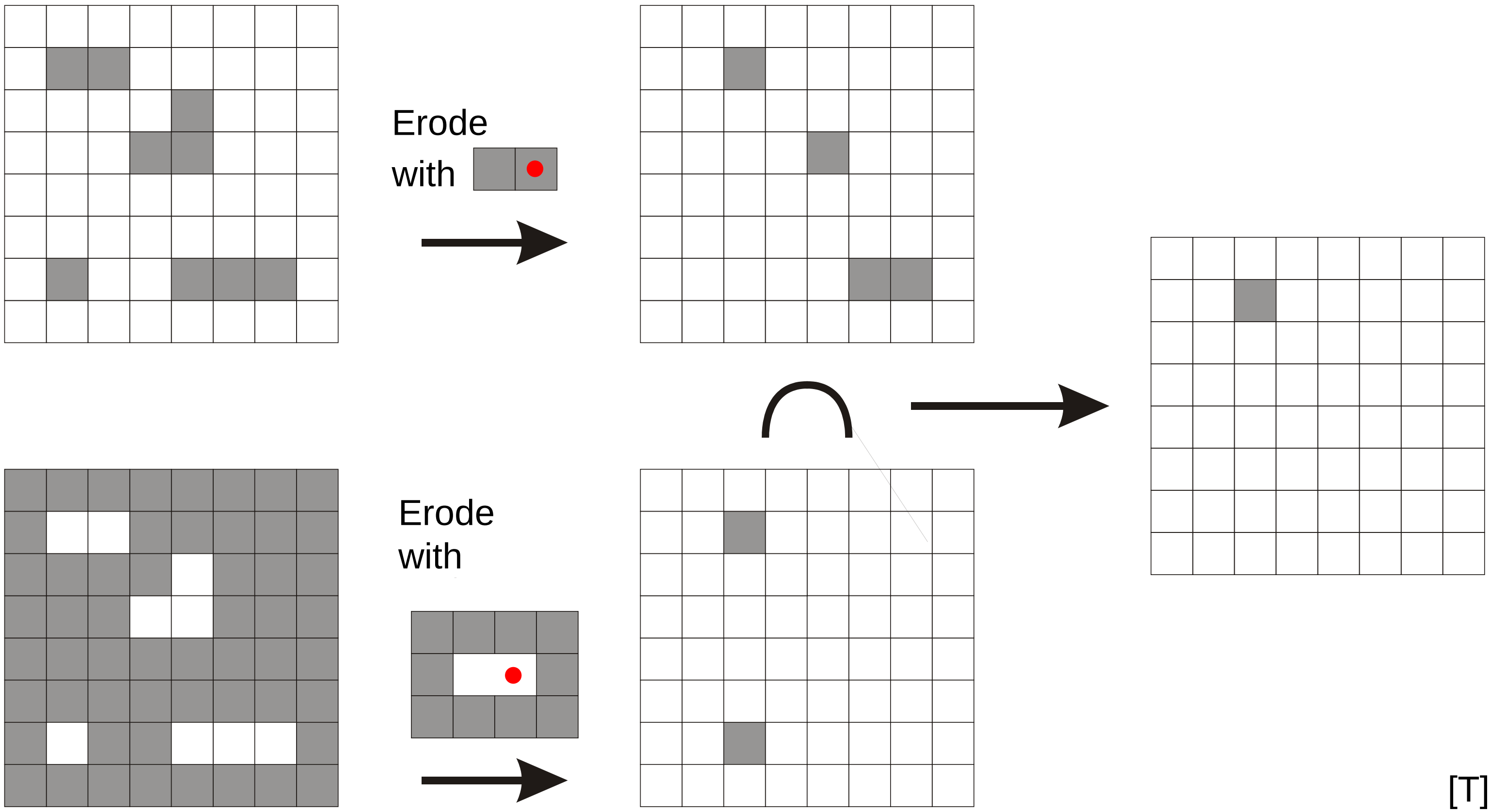
[T]

Intermediate images with mixtures *i/N* and *(N−i)/ N*

Hit-or-miss operation

- Hit-or-miss operation for explicit definition of a pattern
- Detects all image patches which exhibit the exact shape of *S*
- Principle:
  1. Hit: Find all *candidate locations* where the pattern might be by erosion using *S*. Erosion removes all “islands“ smaller than *S*.
  2. Miss: Remove all locations where the pattern can *not* be. For this the inverted binary input image is eroded using the disjoint structuring element *S** to remove all segments larger than *S*. Note *S** is not the mirrored structuring element *S´*.
  3. Intersection of the *hits* and the pixels remaining after the *miss* operation are anchors of the pattern *S*

Hit-or-miss operation

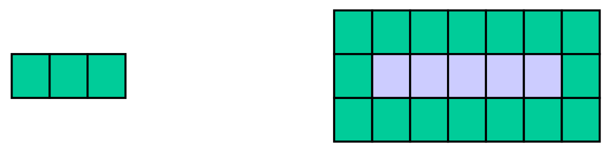


- Anchor point

Hit-or-miss operation with variable structure

Problem: So far, only the *exact* pattern can be found

Hit-or-miss operation for variable structure:



Hit Miss [T]

Accepts horizontal lines of 3, 4, and 5 pixels.

Notation for variable hit-or-miss operation:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{X} & 1 & 1 & 1 & \mathrm{X} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Hit-or-miss operation with variable structure

Variable hit-or-miss operation facilitates search for patterns as parts of segments:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

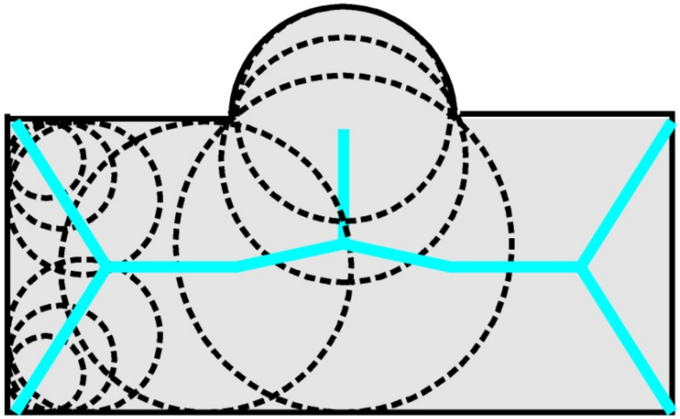
Detects single pixels

$$\begin{pmatrix} 0 & 1 & X \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Detects lower left corners

Skeletonization

- Aim: Get a meaningful, compact description of the shape of binary segments that facilitates classification.
- Skeleton of a segment =
  Set of centers of all bi-tangent circles (circles within the segment that touch the boundary at least at two points).
- Skeleton is always connected.



[T]

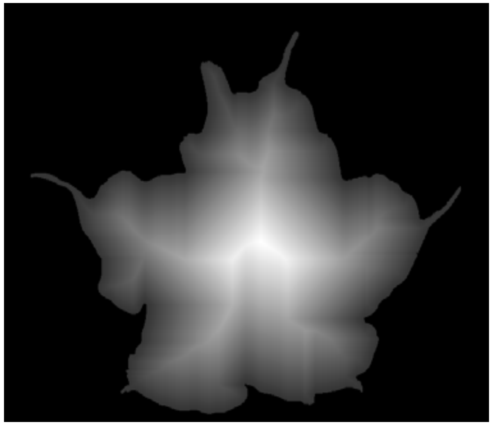
- Application, e.g., in OCR (optical character recognition): Skeletonization of characters filters information relevant for classification such as branching properties out of irrelevant information (font).

Skeletonization

Compute skeleton of a binary segment:

- We need to find all bi-tangent circles, which have the following properties:
  - Circle is completely inside the segment
  - Circle touches a boundary pixel
  - For this boundary pixel, there is no larger circle inside the segment

- Idea 1: Ridges of the distance transform are the skeleton
- Idea 2: Find ridges using the hit-or-miss operation



[T]

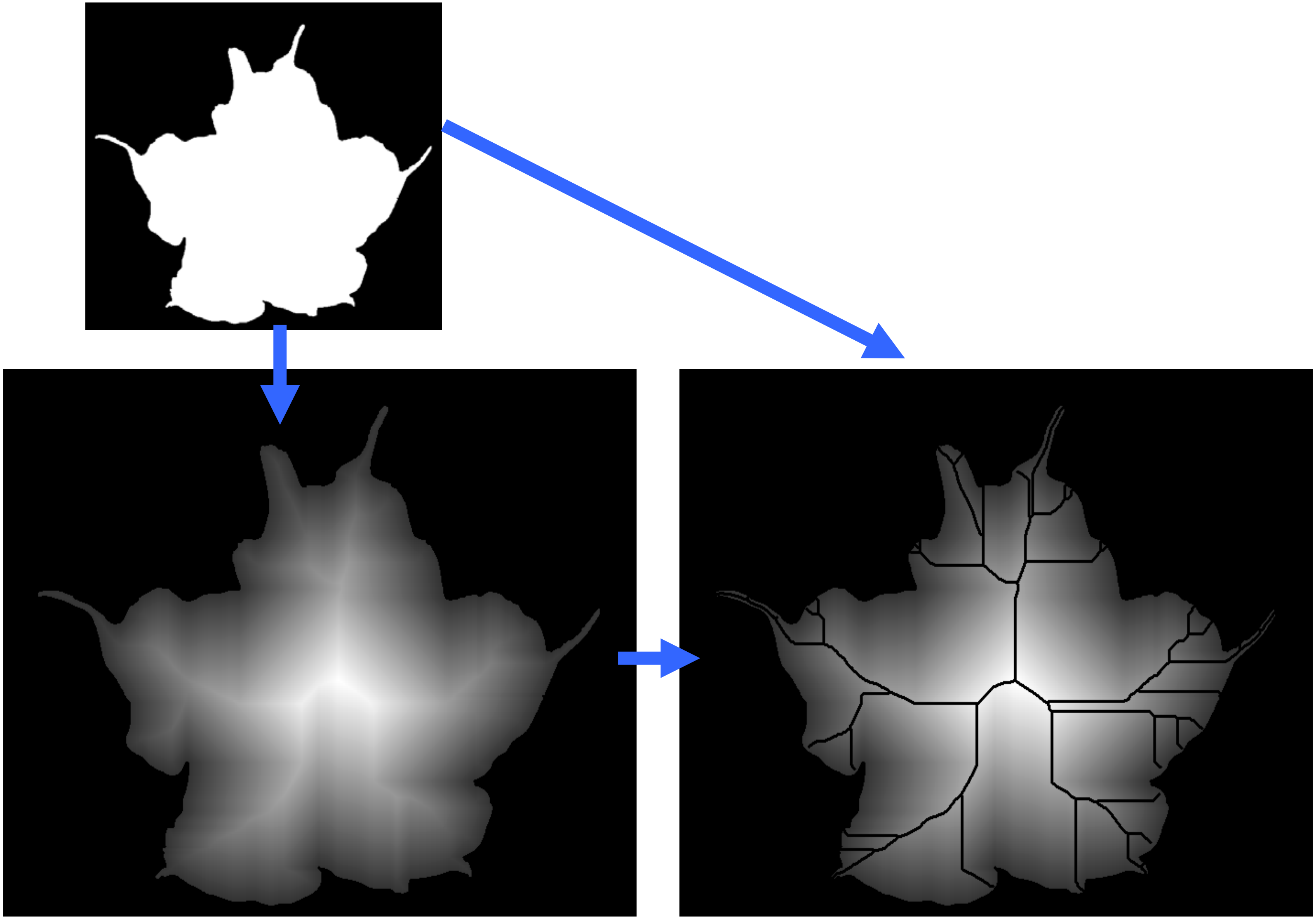
Skeletonization: Iterative removal of non-skeleton-pixels

- Shape of ridges is a priori unknown.
- Therefore remove iteratively all pixels of the distance transformed image which are not ridges.
- In each iteration apply the following eight hit-or-miss operations, which remove the boundary pixels for directions left, right, up, down and the four diagonals:

$$S_l = \begin{pmatrix} 0 & X & 1 \\ 0 & 1 & 1 \\ 0 & X & 1 \end{pmatrix} \quad S_r = \begin{pmatrix} 1 & X & 0 \\ 1 & 1 & 0 \\ 1 & X & 0 \end{pmatrix} \quad S_o = \begin{pmatrix} 0 & 0 & 0 \\ X & 1 & X \\ 1 & 1 & 1 \end{pmatrix} \quad S_u = \begin{pmatrix} 1 & 1 & 1 \\ X & 1 & X \\ 0 & 0 & 0 \end{pmatrix}$$

$$S_{lu} = \begin{pmatrix} X & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & X \end{pmatrix} \quad S_{lo} = \begin{pmatrix} 0 & 0 & X \\ 0 & 1 & 1 \\ X & 1 & 1 \end{pmatrix} \quad S_{ru} = \begin{pmatrix} 1 & 1 & X \\ 1 & 1 & 0 \\ X & 0 & 0 \end{pmatrix} \quad S_{ro} = \begin{pmatrix} X & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & X \end{pmatrix}$$

Skeletonization



[T]

Generalizing morphological operators to gray values

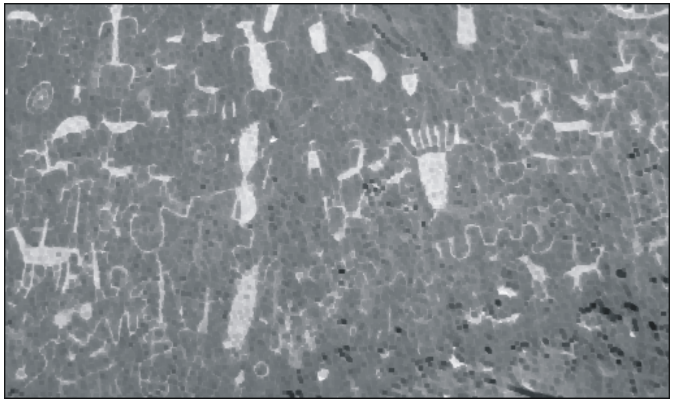
- Now: Images with many (e.g., 8 bit) gray values
- Binary structuring element

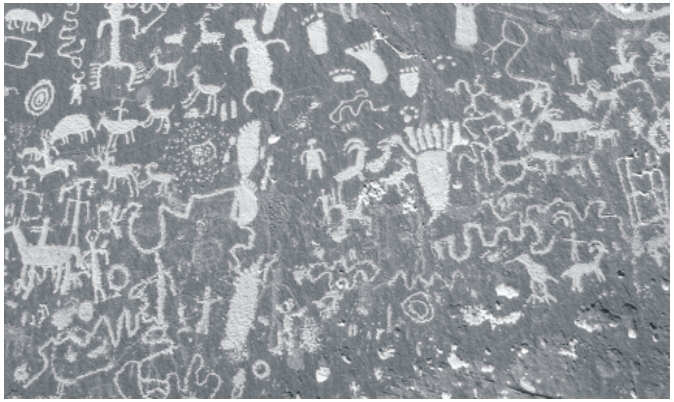
1. Erosion: Result value is the minimum of the gray values of g which are covered by 1-elements of S:

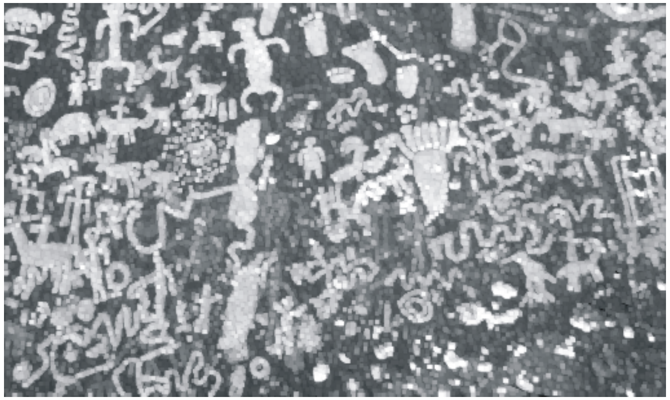
$$g'(x,y) = \min_{k\in[-m,m],l\in[-n,n]}\left(S(k+m,l+n)\cdot g(x+k,y+l)\right)$$

2. Dilation: Maximum

Generalization to gray value images







Erosion

Input

Dilation

[T]

Summary

- Morphological operations for recognition of simple patterns, specified by structuring element.
- Basic operations: Erosion, dilation.
- Composed operations: Opening, closing.
- Applications are, e.g., removal of artifacts such as disrupted pixels, unwanted connection of segments, holes in segments or uneven boundaries.
- Specialized structuring elements, e.g., for the removal of special disruptions such as stripes in a particular direction.
- Exact search for pre-defined patterns using hit-or-miss operation.
- Distance transform yields the distance between arbitrary pixels and the boundary of an object.
- Skeletonization yields compact shape description of objects.

Literature and sources

[T] Klaus D. Tönnies, *Grundlagen der Bildverarbeitung*, Pearson Studium, 2005.

[J] Bernd Jähne, *Digitale Bildverarbeitung,* Springer, 2005.

[FP] David Forsyth, Jean Ponce, *Computer Vision: A Modern Approach,* Prentice Hall, 2002.

[SS] Linda G. Shapiro, George C. Stockman, *Computer Vision,* Prentice Hall, 2001.

[BK] Henning Bässmann, Jutta Kreyss, *Bildverarbeitung Ad Oculos*, Springer, 2004.

[He] Hecht, *Optics*, Addison-Wesley, 1987.

[L] David Lowe, Slides, http://www.cs.ubc.ca/~lowe/425/.

[A] *Artexplosion Explosion® Photo Gallery*, Nova Development Corporation, 23801 Calabasas Road, Suite 2005 Calabasas, California 91302-1547, USA.

[C] Corel GALLERY™ Magic 65000, Corel Corporation, 1600 Carling Ave., Ottawa, Ontario, Canada K1Z 8R7.

[V] Vision Texture Database (VisTex). R. Picard, C. Graczyk, S. Mann, J. Wachman, L. Picard and L. Campbell. Media Laboratory, MIT. Copyright 1995 by the Massachusetts Institute of Technology. http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html

[COIL] S.A. Nene, S.K. Nayar, H. Murase: Columbia Object Image Library: COIL-100, Technical Report, *Dept. of Computer Science, Columbia Univ.,* CUCS-006-96, 1996.

[H] Copyright Gunther Heidemann, 2008.

Literature and sources

[MQ] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. 5th Berkeley Symp. Math. Stat. Probab.,* volume 1, pp. 281-297, 1965.

[CM] D. Comaniciu, P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 24(5): 603-619, 2002.

[HIm] G. Heidemann. Region Saliency as a Measure for Colour Segmentation Stability. *Image and Vision Computing,* Vol. 23, p. 861-876, 2005.

[TP] M. Turk, A. Pentland: Eigenfaces for Recognition, *J. Cognitive Neuroscience*, Vol. 3, p. 71-86, 1991.

[MN] H. Murase, S.K. Nayar: Visual Learning and Recognition of 3-D Objects from Appearance, *Int. J. of Computer Vision,* Vol. 14, p. 5-24, 1995.

[DL] D. Lowe: Distinctive image features from scale-invariant keypoints, *Int. J. of Computer Vision,* Vol. 60(2), p. 91-110, 2004.

[HCv] G. Heidemann: Combining spatial and colour information for content based image retrieval, *Computer Vision and Image Understanding*, Vol. 94(1-3), p. 234-270, 2004.

[IKH] J. Imo, S. Klenk, G. Heidemann: Interactive Feature Visualization for Image Retrieval. *Proc. 19th Int. Conf. on Pattern Recognition ICPR 2008,* 2008.