Digital Image Processing

Binary image processing and analysis

Erchan Aptoula

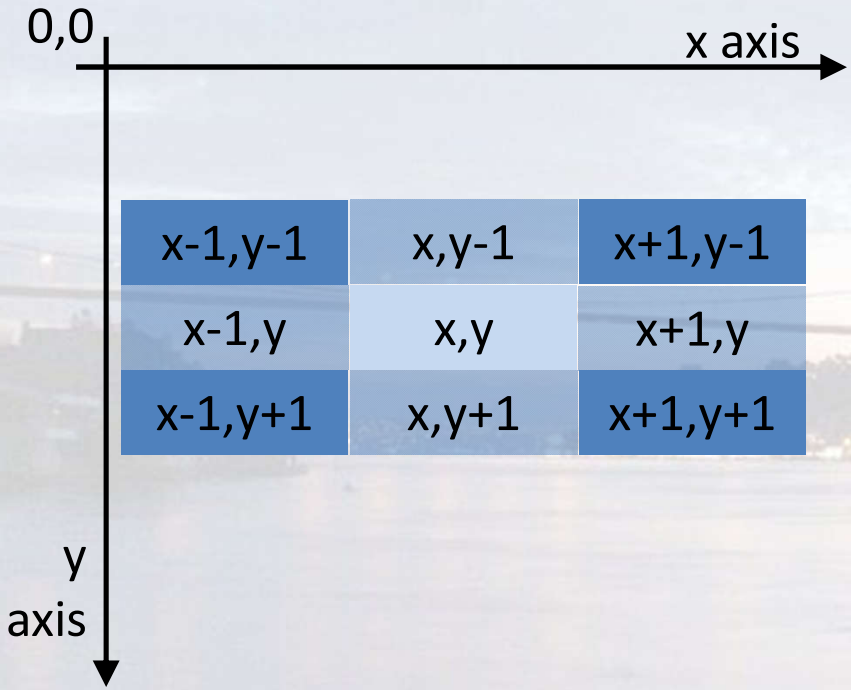
Spring 2016-2017

Today's menu:

- Pixel neighborhoods and distances
- Connected components, connectivity, paths
- Binary mathematical morphology and its wonderful arsenal

Point processing is nice and all, but processing every pixel independently of its surrounding, does not get us very far. Let's meet the **neighborhood**!

2D Digital images are most often represented through a Cartesian grid, with the origin at top-left.



The four horizontal and vertical neighbors of pixel $p = (x, y)$ constitute its **4-neighborhood** $\boldsymbol{N_4(p)}$

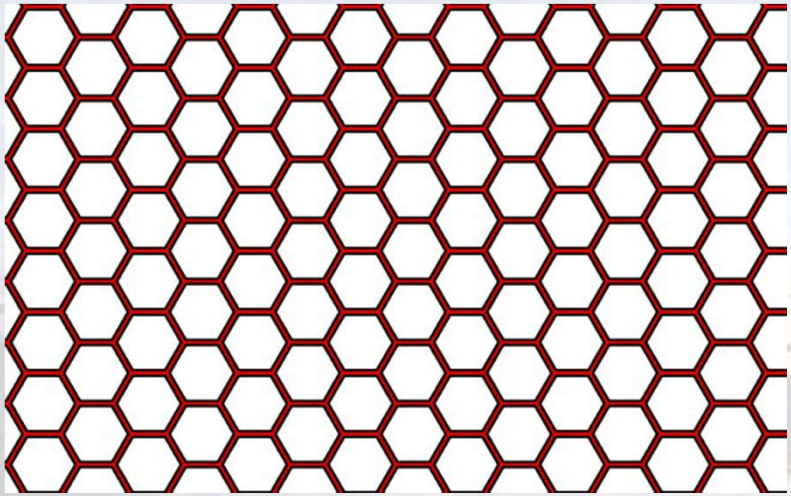
$\boldsymbol{N_D(p)}$ corresponds to its 4 diagonal neighbors

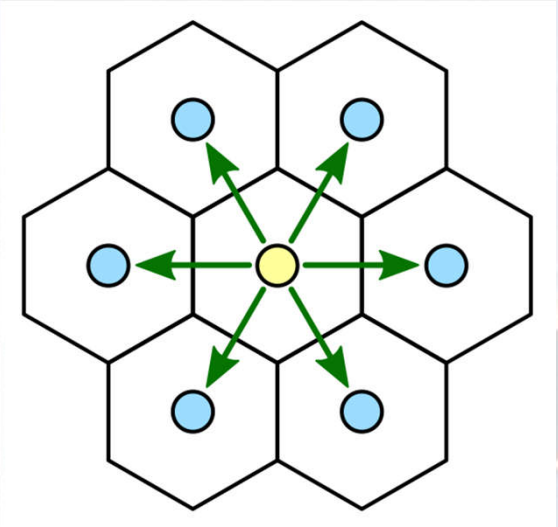
While the 8 pixels surrounding it, constitute its **8-neighborhood** $\boldsymbol{N_8(p) = N_D(p) \cup N_4(p)}$

Two pixels p and q are called 4-adjacent if $\boldsymbol{p \in N_4(q)}$

What happens if the image is 3D? How many neighbors are there then?

It is possible for images to be represented through non-cartesian grids as well; the most common of which is the hexagonal grid:





In which case a pixel has always 6 neighbors. Simpler!

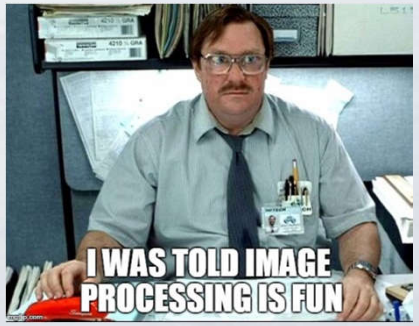
Encountered in microwave imaging.

How do we measure the distance between two pixels?

You can use any valid **distance function** (or metric), i.e. a function defining a distance between a pair of elements of a set.

A metric on a set X is a function $d: X \times X \rightarrow [0, \infty)$ satisfying:

- $d(p, q) \geq 0$:non-negativity
- $d(p, q) = 0 \Leftrightarrow p = q$:identity of indiscernibles
- $d(p, q) = d(q, p)$:symmetry
- $d(p, q) \leq d(p, r) + d(r, q)$:triangle inequality or subadditivity



The Minkowski distance is very common:

$$\forall\, p = (p_1, p_2), q = (q_1, q_2) \in \mathbb{Z}^2, d(p,q) = (|p_1 - q_1|^r + |p_2 - q_2|^r)^{\frac{1}{r}}$$

For $r = 2$ it is known as Euclidean or L_2 distance.

For $r = 1$ it is known as Manhattan, city-block or L_1 distance

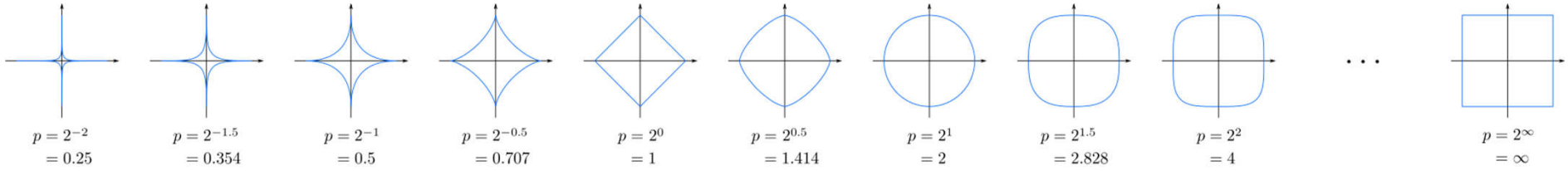
		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

For $r = \infty$ it is known as Chebyshev, L_∞ or chessboard distance:

$$max\{|p_1 - q_1|, |p_2 - q_2|\}$$

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

There are many more (en.wikipedia.org/wiki/Metric_(mathematics))

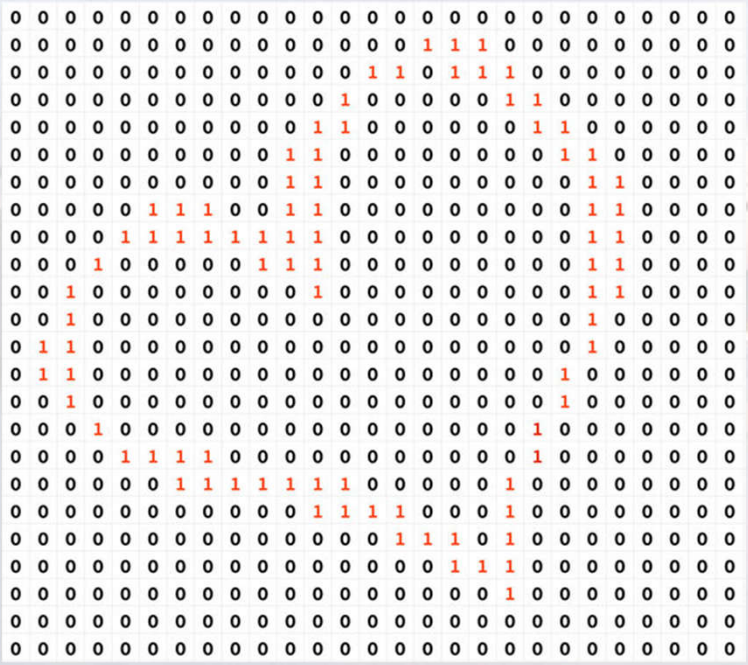


An image is considered **binary**, when it has only two pixel intensities present in it. Usually black and white; 0 and 1, or 0 and 255.

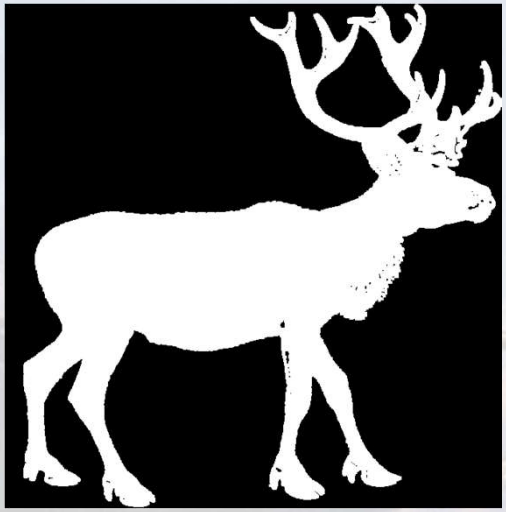
Conventionally black represents the **background**.

And white represents the **foreground**.

Thus their spectral resolution is 1bit.

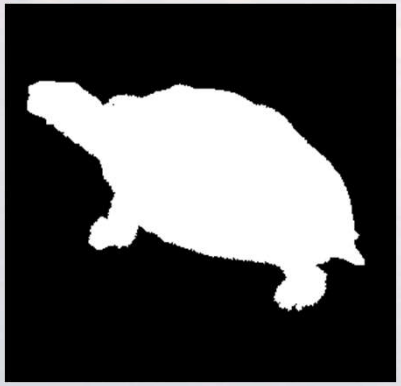


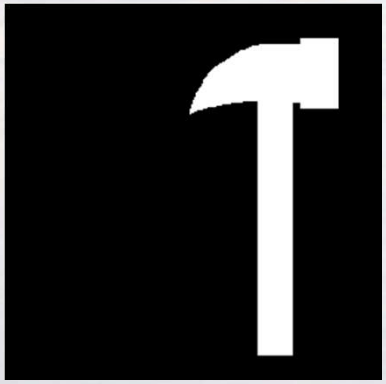
Binary images are encountered often as intermediate results of image processing chains when gray valued images are binarized in order to highlight the object(s) of interest.



- Document analysis
- Industrial vision

many more…





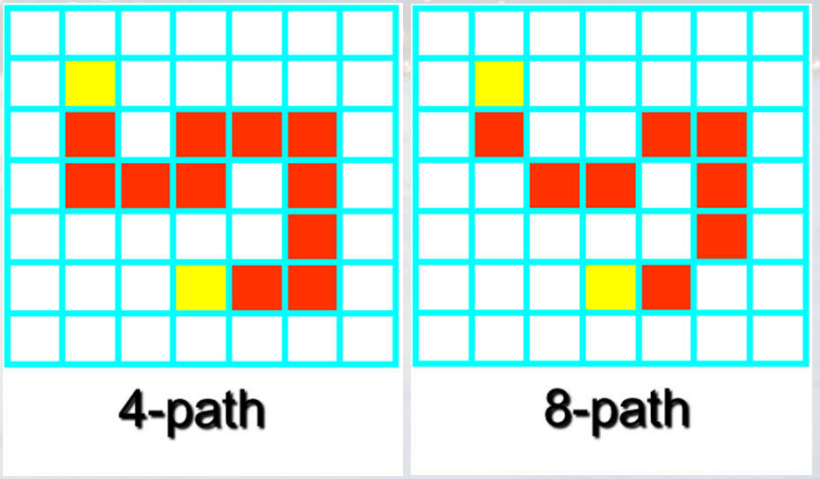
- A path from pixel $p = (x_0, y_0)$ to pixel $q = (x_n, y_n)$ is a sequence of distinct pixels:

$$< (x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) >$$

where (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$.

- n is the length of the path.

- We can define 4-, 8-paths based on the type of adjacency used.

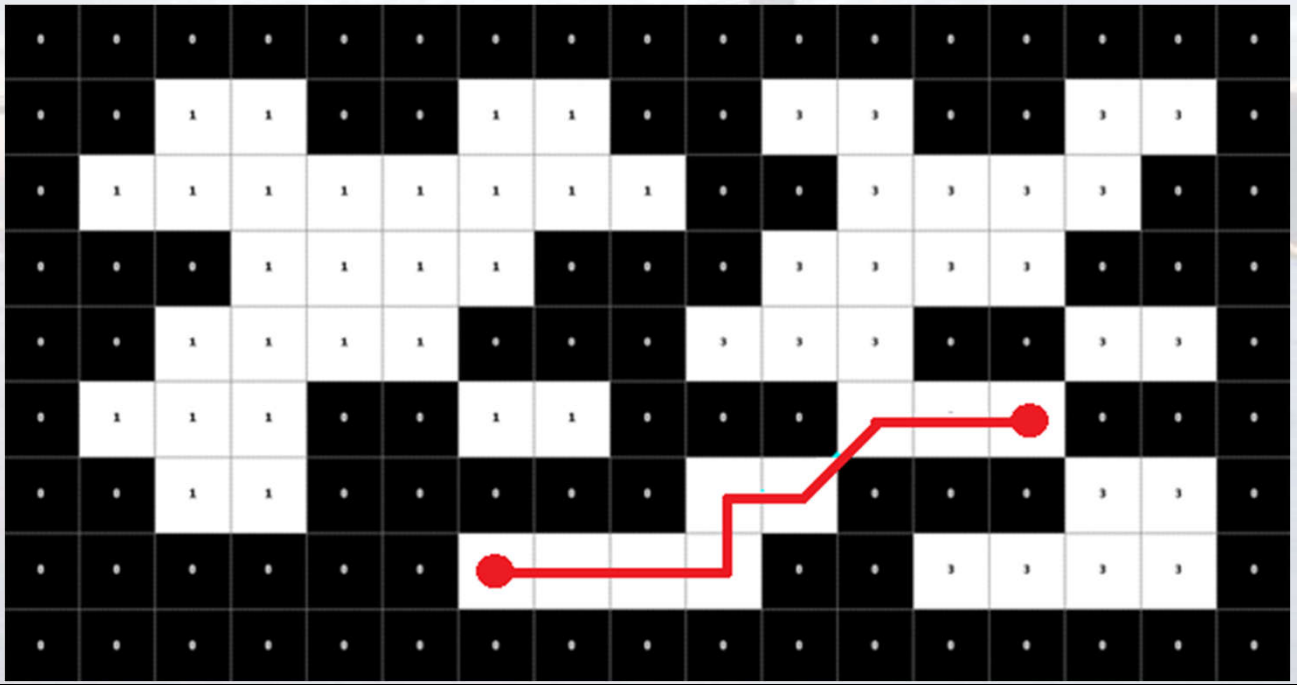


Let S represent a subset of pixels in an image.

Two pixels $p = (x_0, y_0)$ and $q = (x_n, y_n)$ are said to be **connected in S** if there exists a path $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, where $\forall\, i, 0 \leq i \leq n, (x_i, y_i) \in S$

8-connected

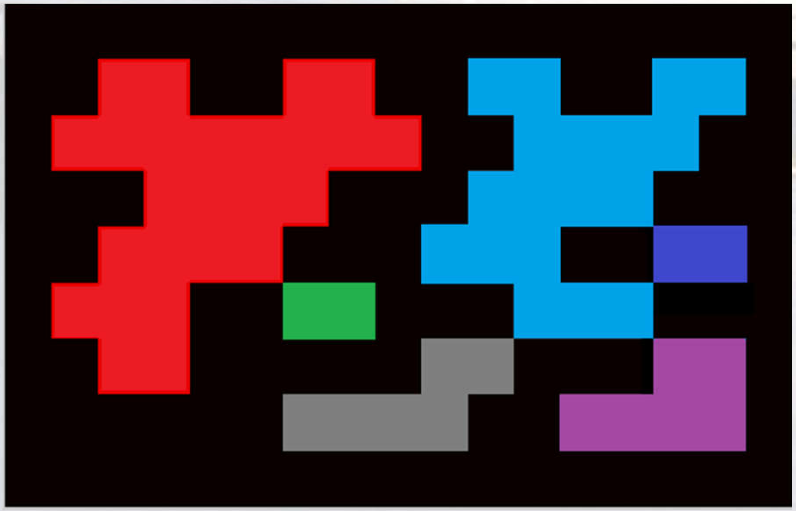
But not 4-connected

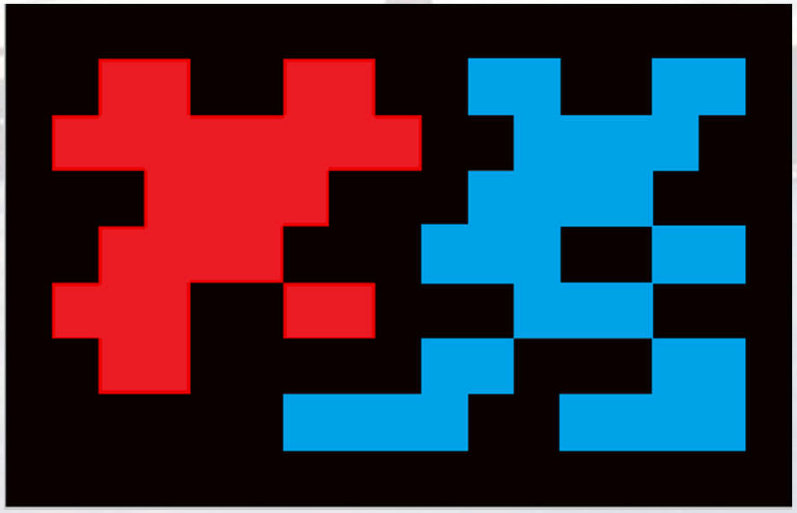


Let S represent a subset of pixels in an image.

For every pixel p in S, the set of pixels in S that are connected to p is called a ***connected component*** of S.

The foreground has 2 connected components w.r.t. 8-connectivity, and
6 w.r.t. 4-connectivity.

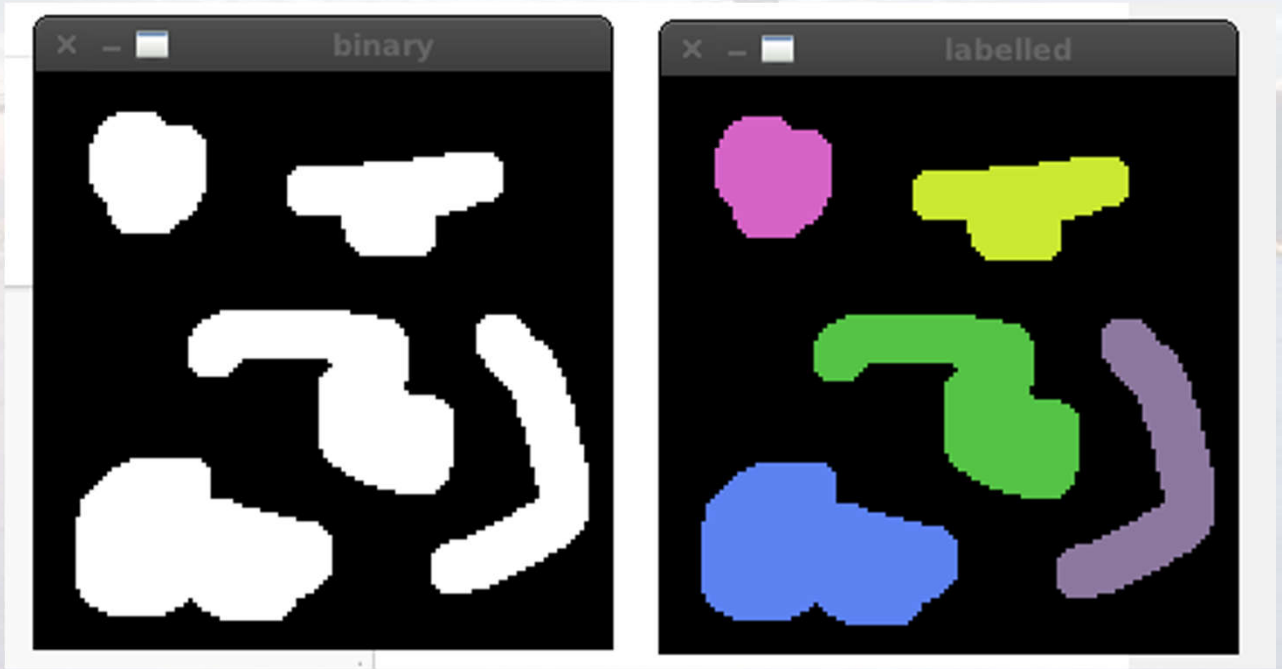




Labeling/counting the connected components of a binary image is a frequent need.

How to label the **CC**'s of an image?

- One component at a time (recursively or iteratively)
- In parallel (requires parallel hardware)
- Two-pass algorithm (common)

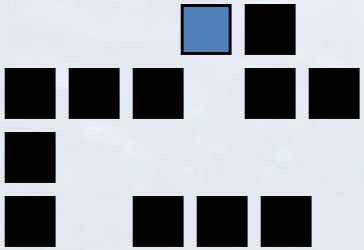


One component at a time CC labeling

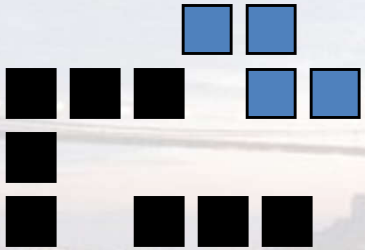
It is simple, yet, if you have
large CC's in your input, it
can very easily overflow your stack.

You can of course implement
It iteratively with a LIFO structure.

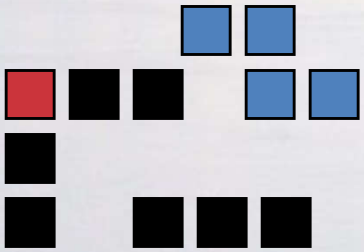
```
Compute the connected components of a binary image.
B is the original binary image.
LB will be the labeled connected component image.

procedure recursive_connected_components(B, LB);
{
LB := negate(B);
label := 0;
find_components(LB, label);
print(LB);
}

procedure find_components(LB, label);
{
for L := 0 to MaxRow
   for P := 0 to MaxCol
      if LB[L,P] == -1 then
         {
         label := label + 1;
         search(LB, label, L, P);
         }
}

procedure search(LB, label, L, P);
{
LB[L,P] := label;
Nset := neighbors(L, P);
for each (L',P') in Nset
   {
   if LB[L',P'] == -1
   then search(LB, label, L', P');
   }
}
```



First pixel



After 4 recursive calls, no 4-neighbors neighbors to color



Start again with a new color on an unmarked pixel

Iterative version of the **one component at a time** algorithm:

1. Start from the first pixel in the image. Set "curlab" (short for "current label") to 1. Go to (2).
2. If this pixel is a foreground pixel and it is not already labeled, then give it the label "curlab" and add it as the first element in a stack, then go to (3). If it is a background pixel or it was already labeled, then repeat (2) for the next pixel in the image.
3. Pop out an element from the stack, and look at its neighbors (based on any type of connectivity). If a neighbor is a foreground pixel and is not already labeled, give it the "curlab" label and push it in the stack. Repeat (3) until there are no more elements in the stack.
4. Go to (2) for the next pixel in the image and increment "curlab" by 1

The first pass: propagates a label to the right and below. If two labels get propagated to the same pixel, they are recorded as an equivalence.

The second pass: assigns to each pixel the label of its equivalence class.

The key to its performance is the **union-find** data structure used for merging and recording equivalence relations.

PARENT

1	2	3	4	5	6	7	8
2	3	0	3	7	7	0	3

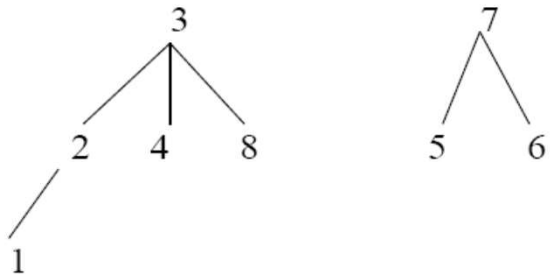


Figure 3.9: The union-find data structure for two sets of labels. The first set contains the labels { 1,2,3,4,8 } , and the second set contains labels { 5,6,7 } . For each integer label i, the value of $PARENT[i]$ is the label of the parent of i or zero if i is a root node and has no parent.

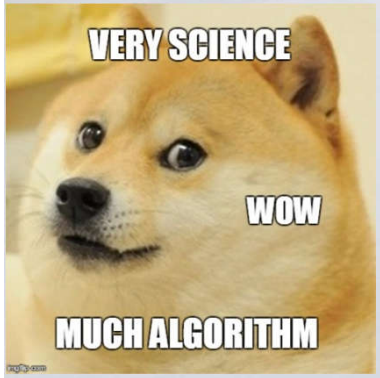
Union-Find

Find the parent label of a set.
X is a label of the set.
PARENT is the array containing the union-find data structure.

```
procedure find(X, PARENT);
{
j := X;
while PARENT[j] <> 0
   j := PARENT[j];
return(j);
}
```

Construct the union of two sets.
X is the label of the first set.
Y is the label of the second set.
PARENT is the array containing the union-find data structure.

```
procedure union(X, Y, PARENT);
{
j := X;
k := Y;
while PARENT[j] <> 0
   j := PARENT[j];
while PARENT[k] <> 0
   k := PARENT[k];
if j <> k then PARENT[k] := j;
}
```



Binary image processing and analysis

Two pass algorithm

Initialize the data structures for classical connected compor

```
procedure initialize();
"Initialize global variable label and array PARENT."
{
"Initialize label."
label := 0;
"Initialize the union-find structure."
for i := 1 to MaxLab
   PARENT[i] := 0;
}
```

Compute the connected components of a binary image.
B is the original binary image.
LB will be the labeled connected component image.

```
procedure classical_with_union-find(B,LB);
{
"Initialize structures."
initialize();
"Pass 1 assigns initial labels to each row L of the image."
for L := 0 to MaxRow
   {
   "Initialize all labels on line L to zero"
   for P := 0 to MaxCol
      LB[L,P] := 0;
   "Process line L."
   for P := 0 to MaxCol
      if B[L,P] == 1 then
         {
         A := prior_neighbors(L,P);
         if isempty(A)
         then { M := label; label := label + 1; };
         else M := min(labels(A));
         LB[L,P] := M;
         for X in labels(A) and X <> M
            union(M, X, PARENT);
         }
   }
"Pass 2 replaces Pass 1 labels with equivalence class labels."
for L := 0 to MaxRow
   for P := 0 to MaxCol
      if B[L,P] == 1
      then LB[L,P] := find(LB[L,P],PARENT);
} ;
```

Binary image processing and analysis: Two pass algorithm

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

1	1	0	2	2	2	0	3
1	1	0	2	0	2	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	7	7	3

a) after Pass 1

1	1	0	1	1	1	0	3
1	1	0	1	0	1	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	3	3	3

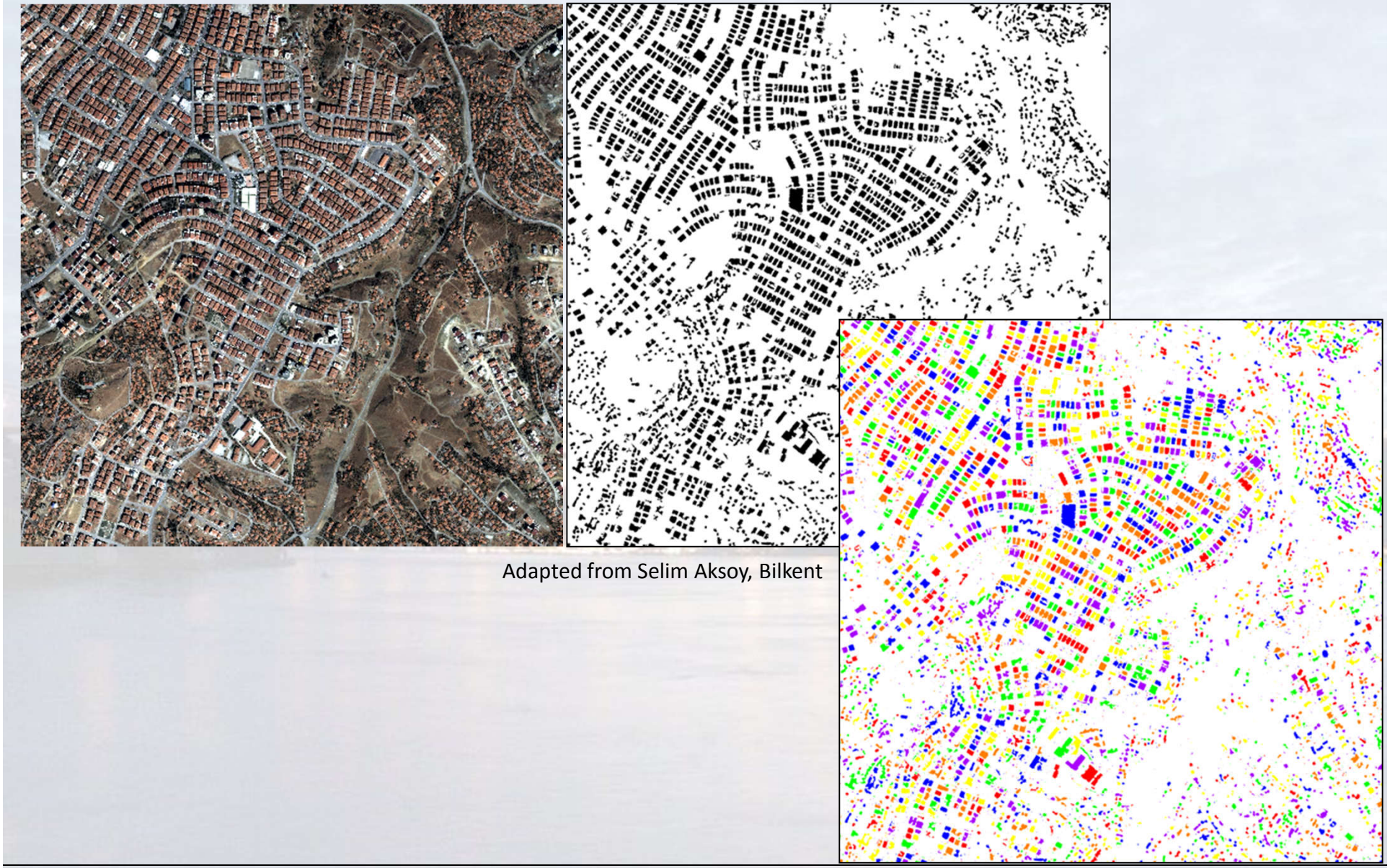
c) after Pass 2

PARENT

1	2	3	4	5	6	7
0	1	0	0	0	0	3

b) union-find structure showing equivalence classes

Binary image processing and analysis



Adapted from Selim Aksoy, Bilkent

What is **Mathematical Morphology** (*matematiksel biçimbilim*)?

It is a **non-linear** image processing/analysis approach with a **rigorous** mathematical **foundation** specializing in exploiting **spatial relationships** and **shape information**.

	Image Analysis Approaches	
Linear	**Linear:** Fourier transform, convolutions	**Statistical**
Non linear	**Morphological:** granulometry, watershed	**Syntactic**

Capable of most, if not all, image analysis tasks, and fully developed for binary, gray, color, 3D, video, multivariate images, etc.

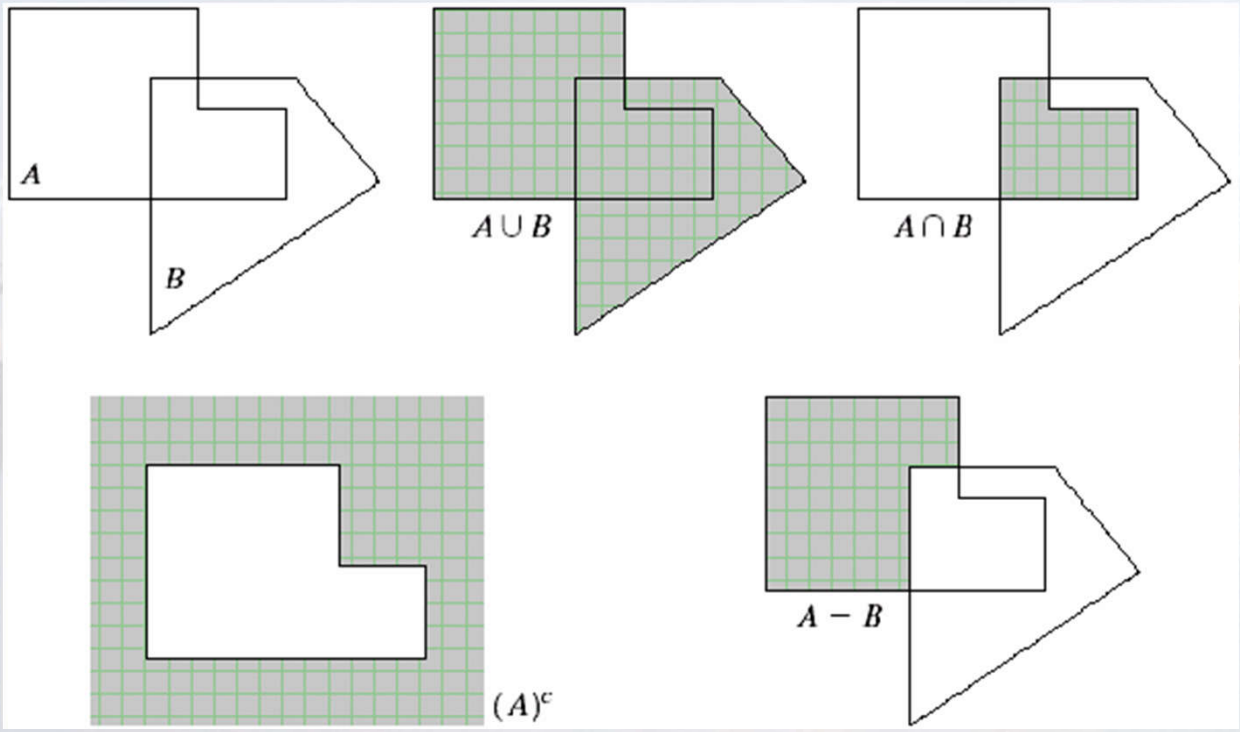
Yet sadly, it is mostly known as a framework for binary image analysis.

It was initially developed in the 60s by Georges Matheron (mining engineer) and his PhD student Jean Serra (civil engineer),

for quantifying iron ore properties from petrography images.

Mathematical morphology is formally based on **complete lattice theory** (more on this in a few weeks). For the sake of simplicity, we'll use **set theory** for the binary case.

Recall on set theory:

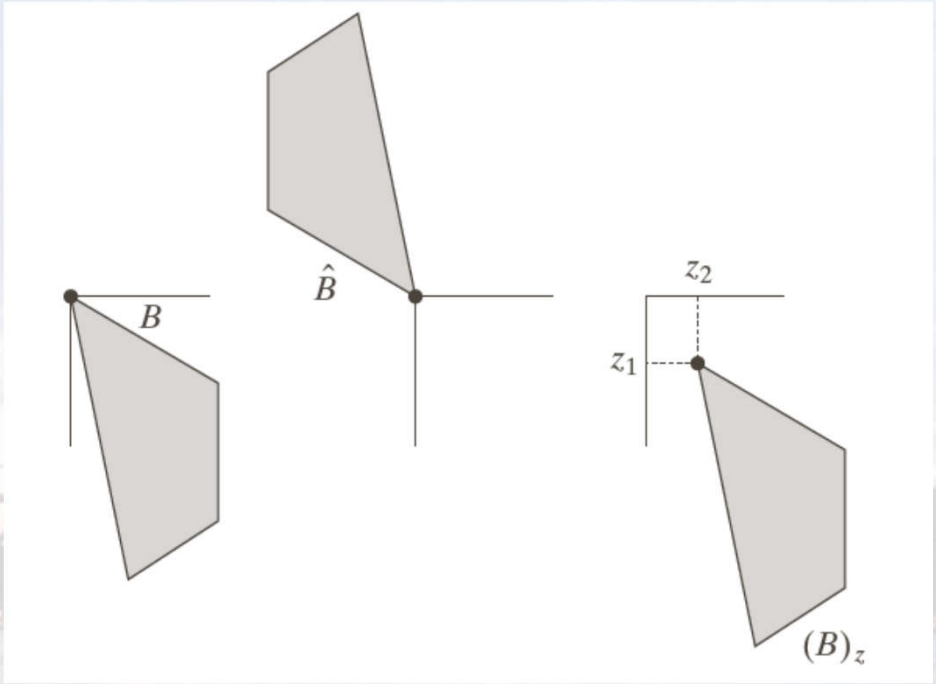


$$A^c = \{w \mid w \notin A\}$$

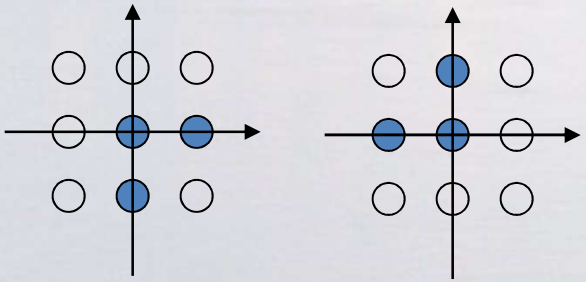
$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c$$

Translation operator

$$(A)_z = \{w \mid w = a + z, a \in A\}$$



Reflection operator

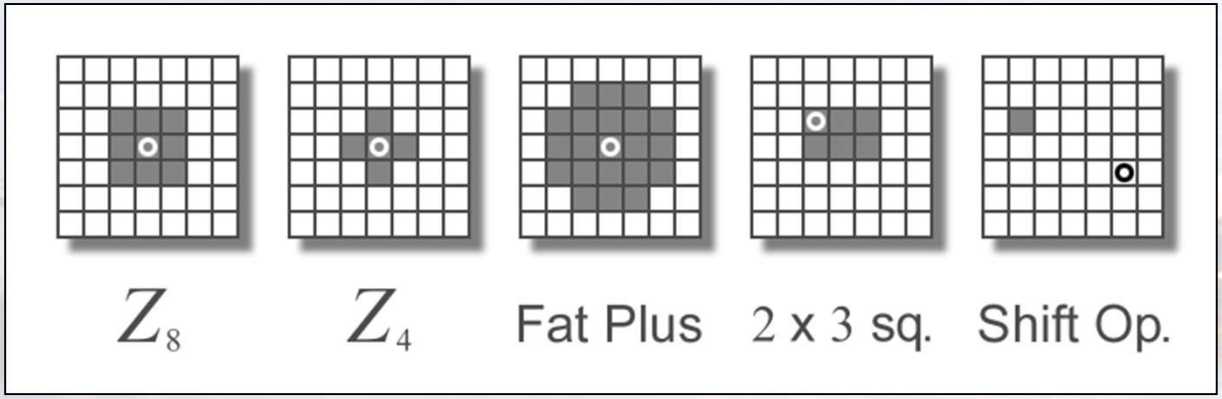


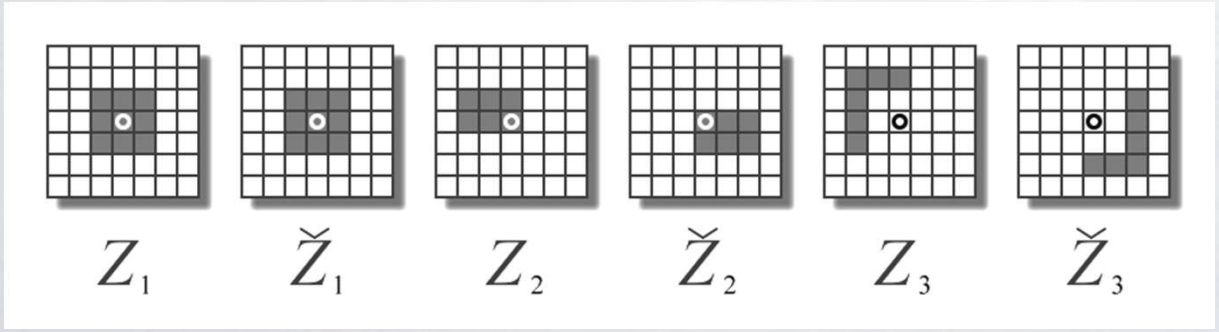
$$\check{B} = \{w | w = -b, \forall b \in B\}$$

Morphological operators are applied using a **Structuring Element** (SE)

Given a binary image $f: \mathbb{Z}^2 \rightarrow \{0,1\}$, $B \subseteq \mathbb{Z}^2$ is a SE. They can be of any shape, size or connectivity.

Every SE is also associated with an origin pixel; usually, but not necessarily, its central one.



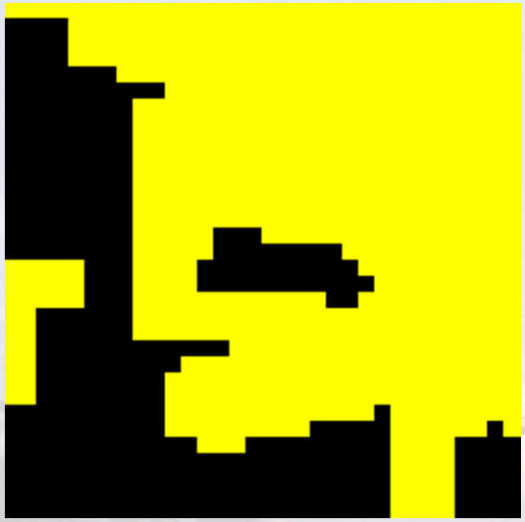


The two fundamental operators of MM are **dilation** and **erosion**, most of the other tools consist of their combinations; serially, in parallel etc.

The dilation of a binary image A with a SE B is denoted as:

$$A \oplus B = \{z | \check{B}_z \cap A \neq \emptyset\},$$
$$= \bigcup_{a \in A} B_a.$$

It consists of all the displacements z, such that $\check{B}_z$ and A overlap by at least one pixel.



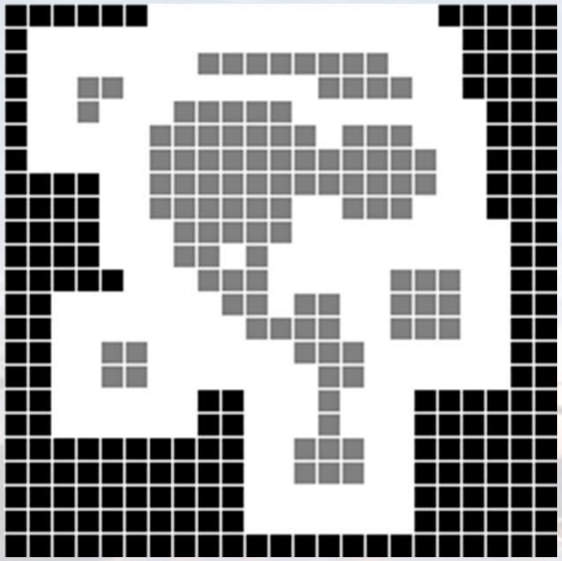
dilated image

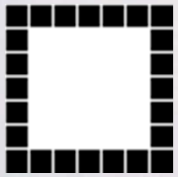
original / dilation

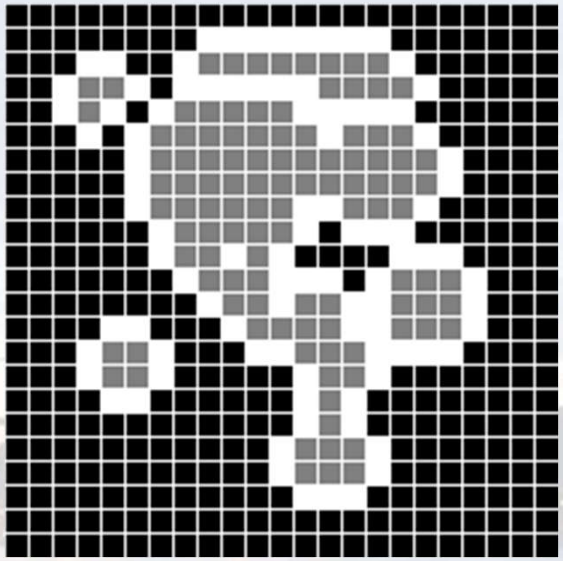
original image

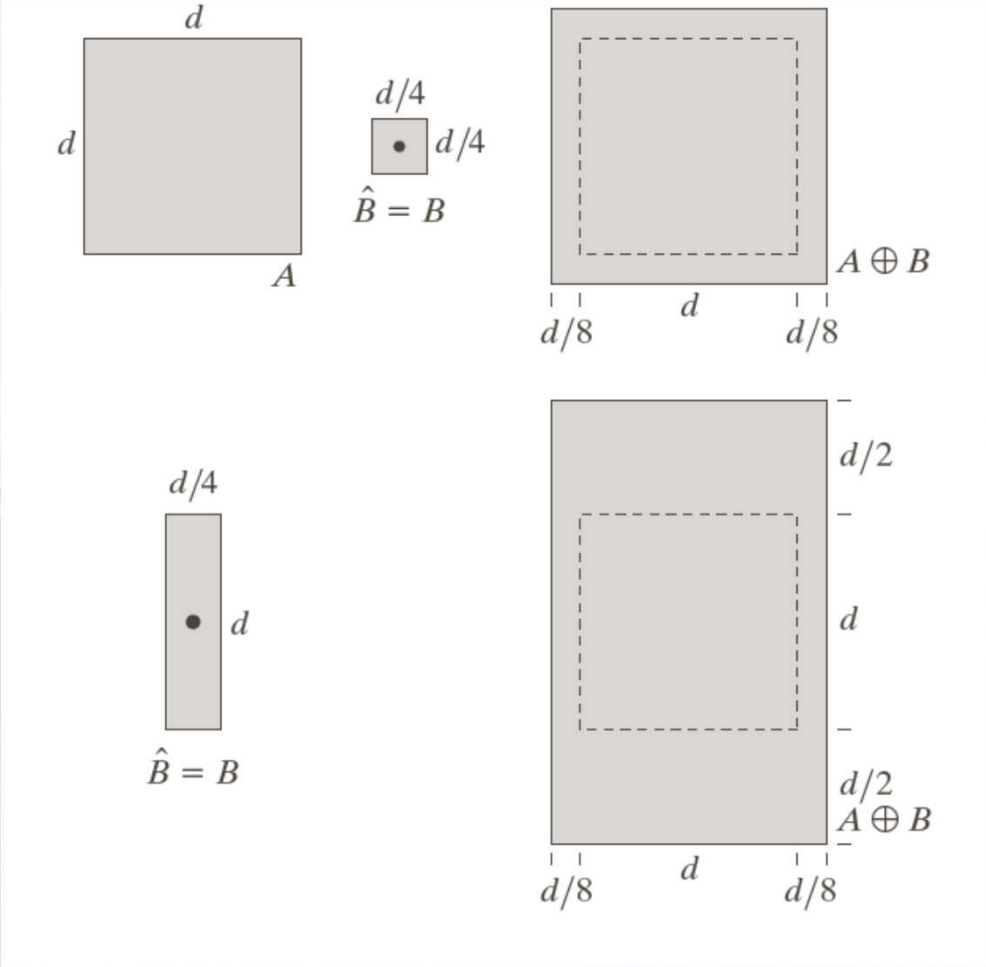
SE = Z_8

Effect of SE shape



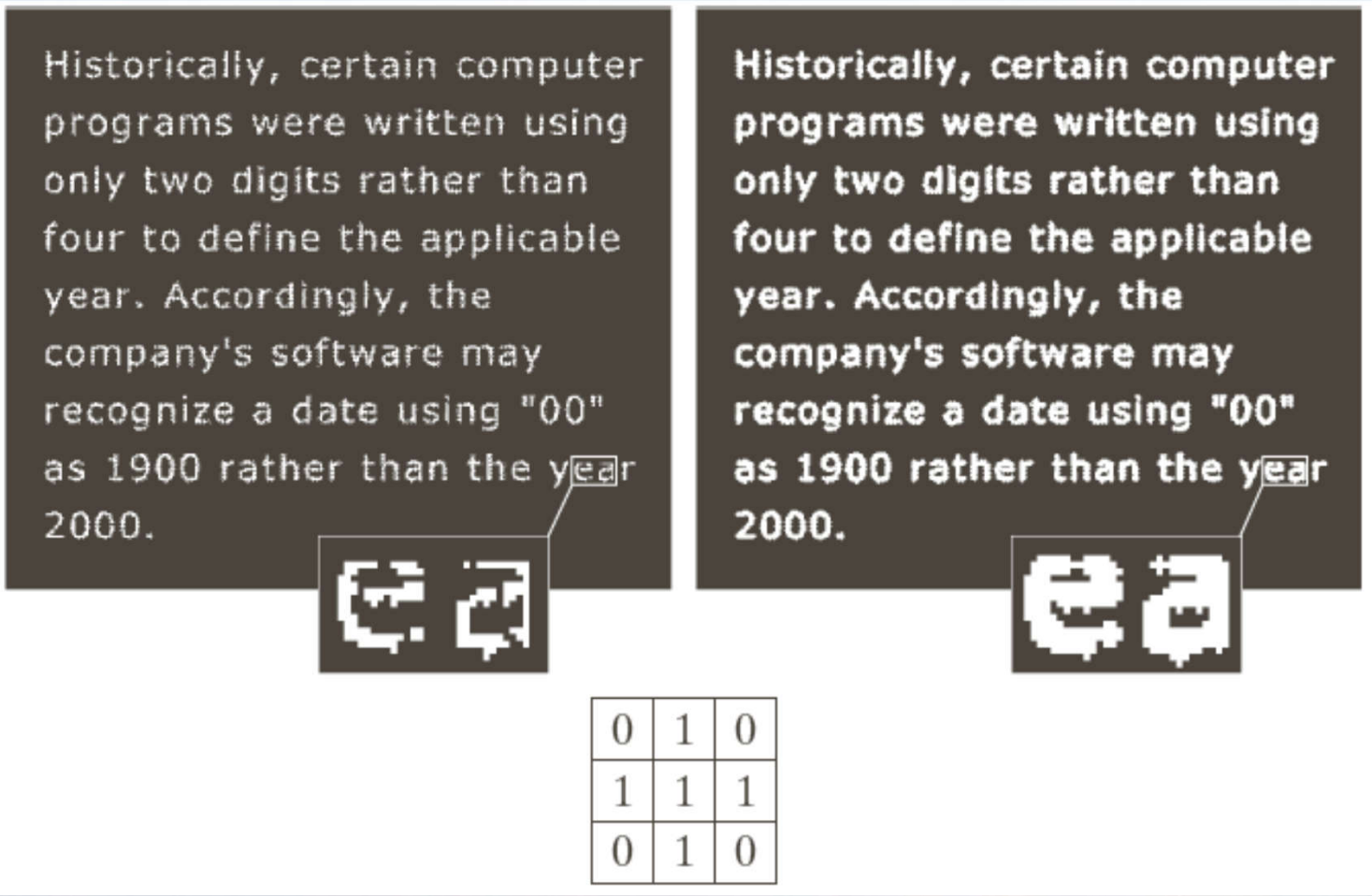






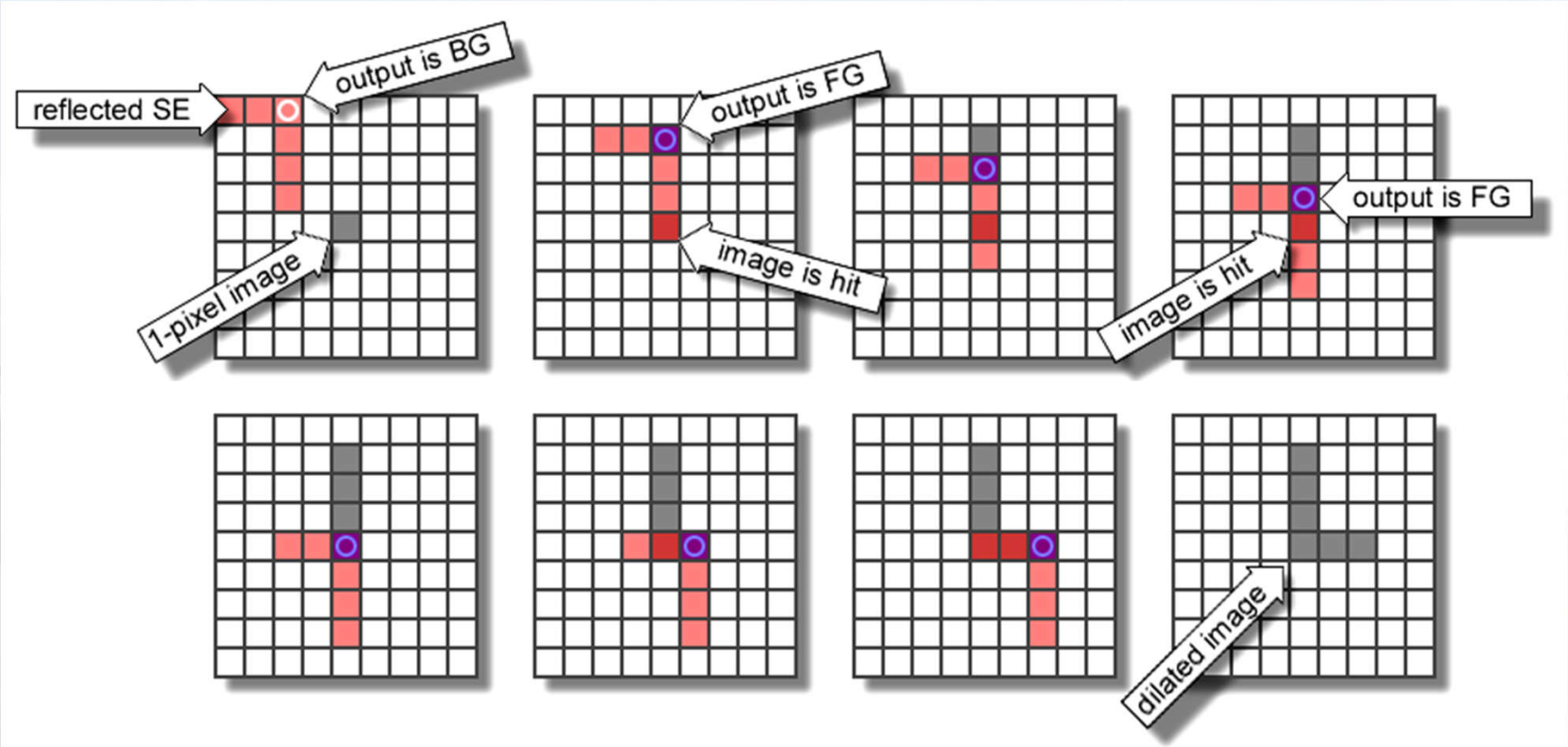
a b c
d e

FIGURE 9.6
(a) Set A. (b) Square structuring element (the dot denotes the origin). (c) Dilation of A by B, shown shaded. (d) Elongated structuring element. (e) Dilation of A using this element. The dotted border in (c) and (e) is the boundary of set A, shown only for reference



a c
b

FIGURE 9.7
(a) Sample text of poor resolution with broken characters (see magnified view).
(b) Structuring element.
(c) Dilation of (a) by (b). Broken segments were joined.


reflected SE
output is BG
1-pixel image
output is FG
image is hit
output is FG
image is hit
dilated image

Properties of dilation

Commutative

$$A \oplus B = B \oplus A$$

Associative

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

Extensive

$$if \quad 0 \in B, A \subseteq A \oplus B$$

Increasing

$$A \subseteq B \; implies \; A \oplus D \subseteq B \oplus D$$

Shift invariant

$$(A)_x \oplus B = (A \oplus B)_x$$

SE decomposition

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$$

The erosion of a binary image A with a SE B is denoted as

$$A \ominus B = \{z | B_z \subseteq A\},$$
$$= \{a | a + b \in A, \forall b \in B\}$$

It consists of all the positions z, such that B_z is contained in A.

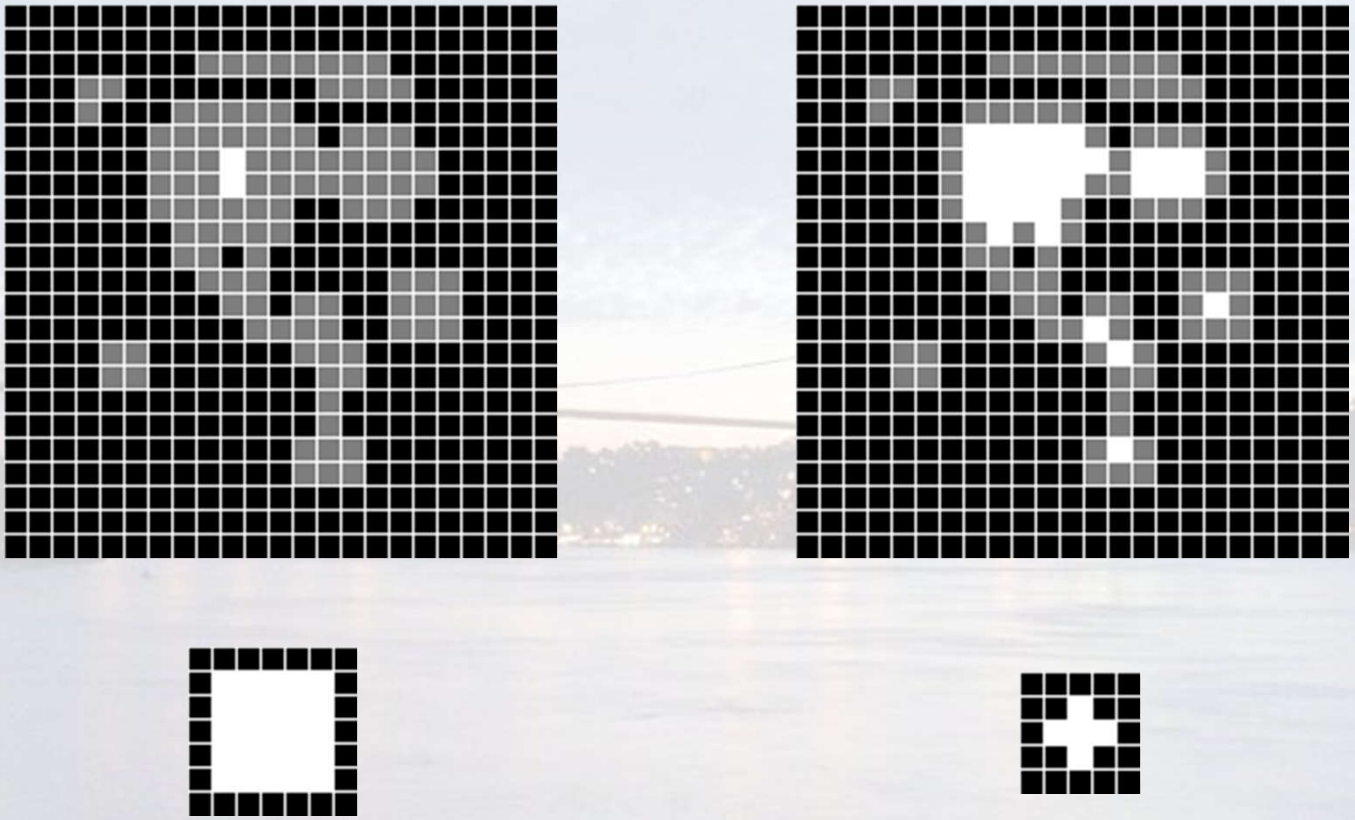
eroded image

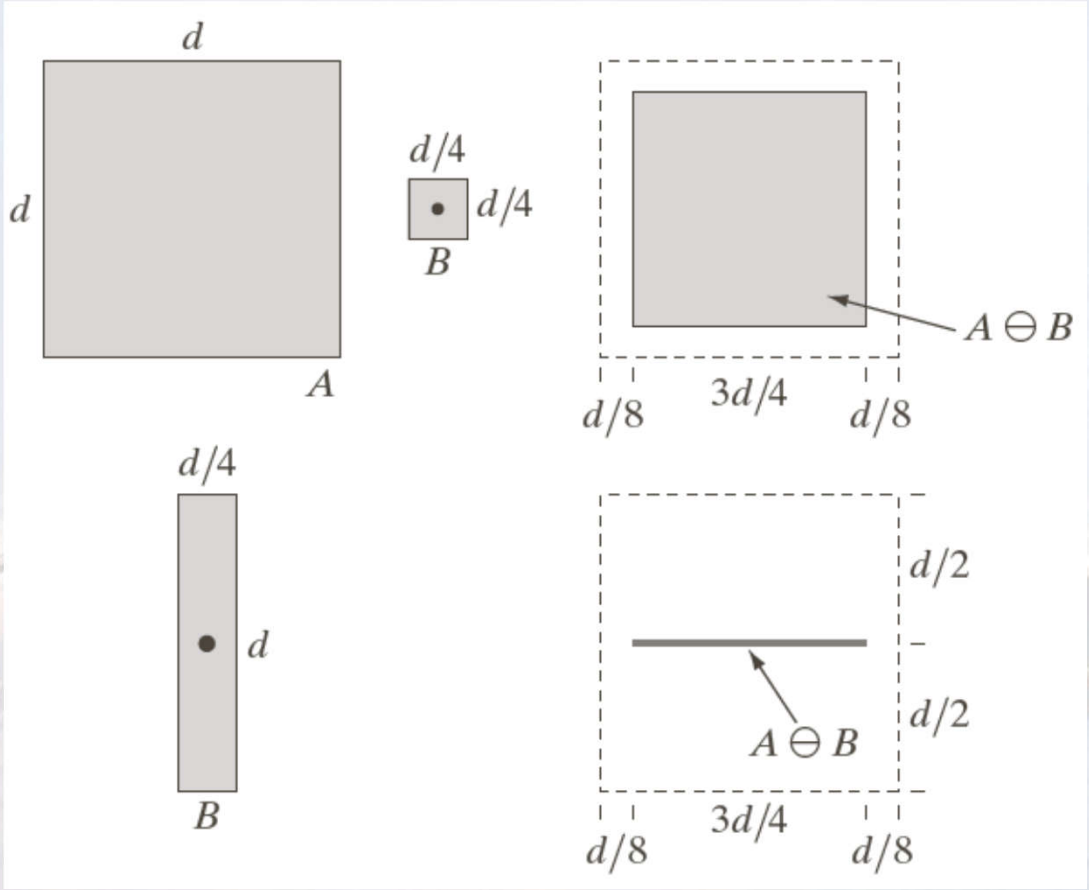
erosion / original

original image

SE = Z_8

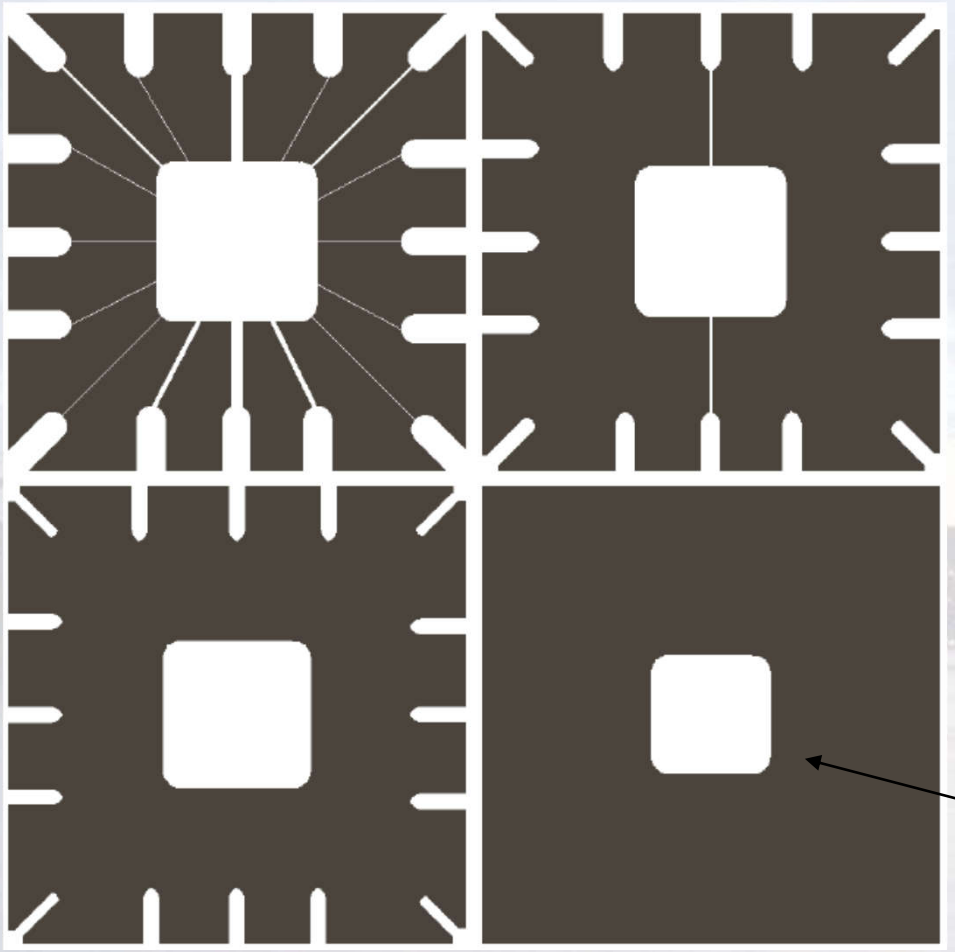
Effect of SE shape





a b c
d e

FIGURE 9.4 (a) Set A. (b) Square structuring element, B. (c) Erosion of A by B, shown shaded. (d) Elongated structuring element. (e) Erosion of A by B using this element. The dotted border in (c) and (e) is the boundary of set A, shown only for reference.



a b
c d

FIGURE 9.5 Using erosion to remove image components. (a) A 486 × 486 binary image of a wire-bond mask. (b)–(d) Image eroded using square structuring elements of sizes 11 × 11, 15 × 15, and 45 × 45, respectively. The elements of the SEs were all 1s.

Smaller than original

Properties of erosion

Not commutative

$$A \Theta B \neq B \Theta A$$

Anti-extensive

$$if \;\; 0 \in B, A \Theta B \subseteq A$$

Decreasing

$$A \subseteq C \; implies \; A \Theta B \subseteq C \Theta B, \; B \supseteq C \; implies \; A \Theta B \subseteq A \Theta C$$

Shift invariant

$$A_x \Theta B = (A \Theta B)_x, \;\; A \Theta B_x = (A \Theta B)_{-x}$$

SE decomposition

$$A \Theta (B \cup C) = (A \Theta B) \cap (A \Theta C)$$

Dualities

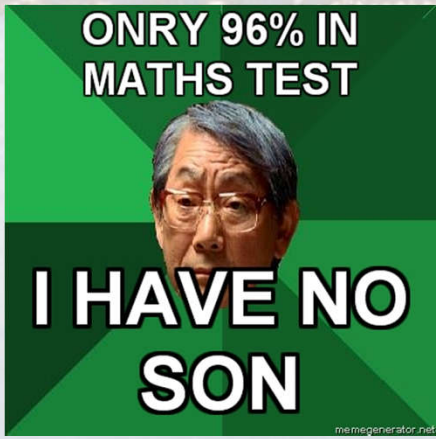
1. Adjunction

$$A \subseteq C \ominus B \Leftrightarrow A \oplus B \subseteq C$$

2. Duality by complementation

$$A \oplus B = \left(A^c \ominus \check{B}\right)^c$$
$$A \ominus B = \left(A^c \oplus \check{B}\right)^c$$

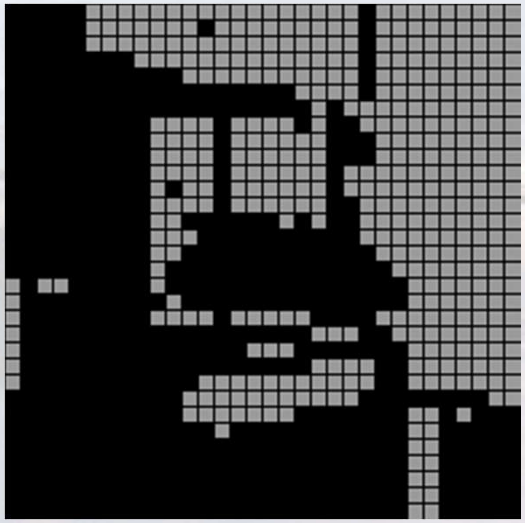
i.e. dilating an image is the same thing as eroding its background with the reflected SE.



Boundary extraction

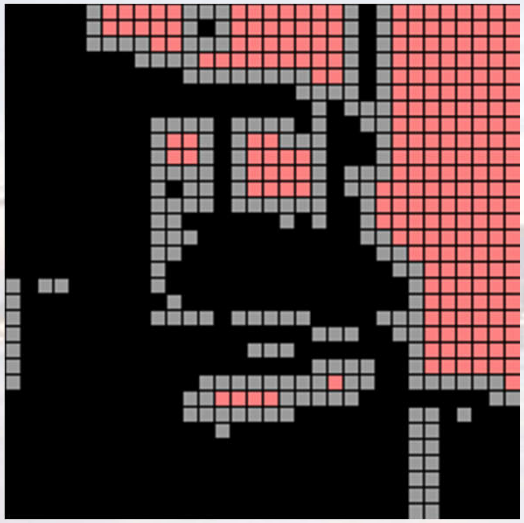
$$internal\ bou\ ndary(A) = A - (A \ominus B)$$

binary image



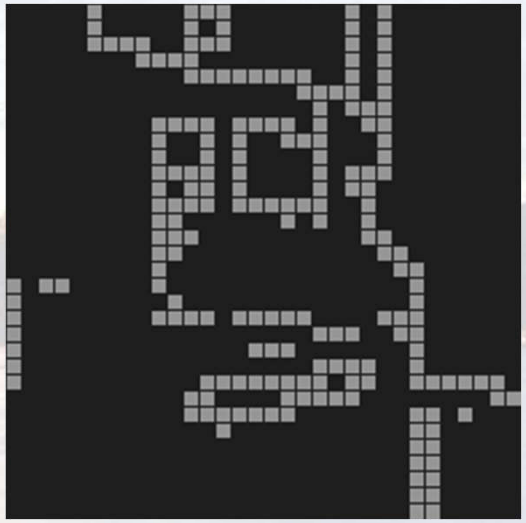
original

8-connected SE



erosion by square

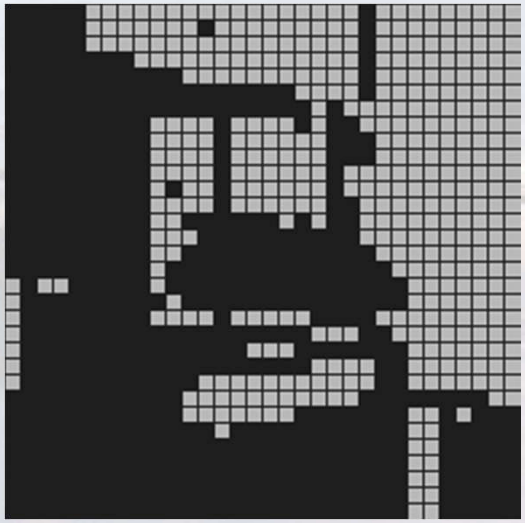
4-conn inside bdry



difference

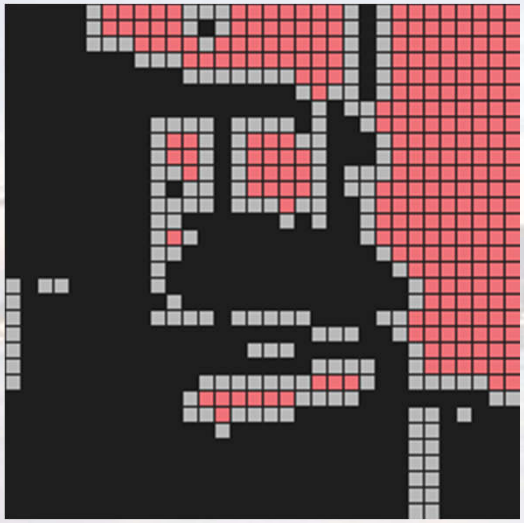
Boundary extraction

binary image



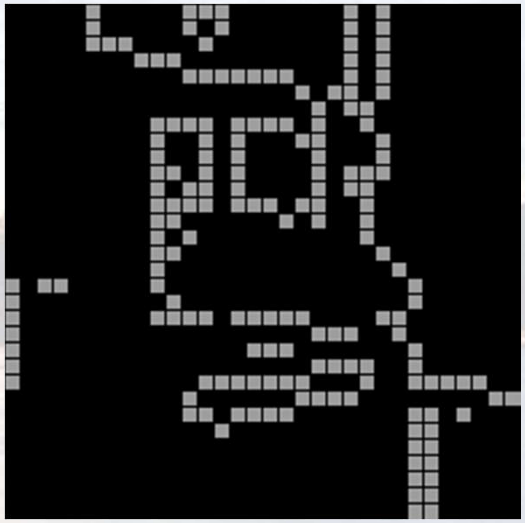
original

4-connected SE



erosion by plus

8-conn inside bdry.

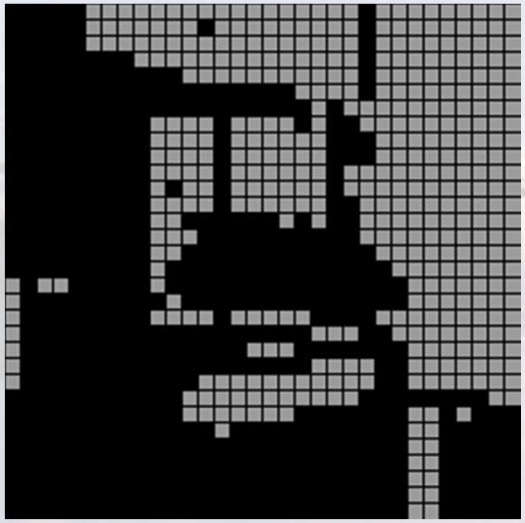


difference

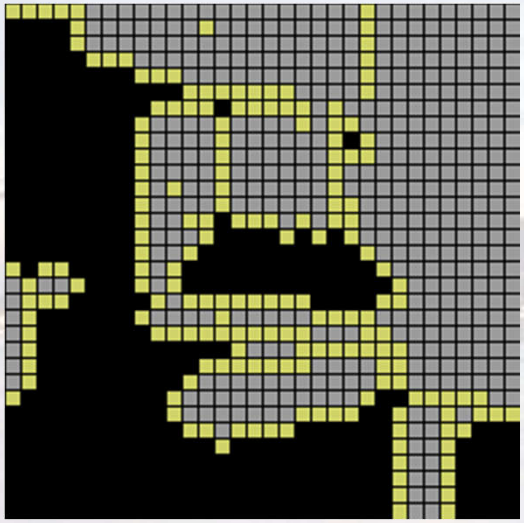
Boundary extraction

$$external\ bou\ ndary(A) = (A \oplus B) - A$$

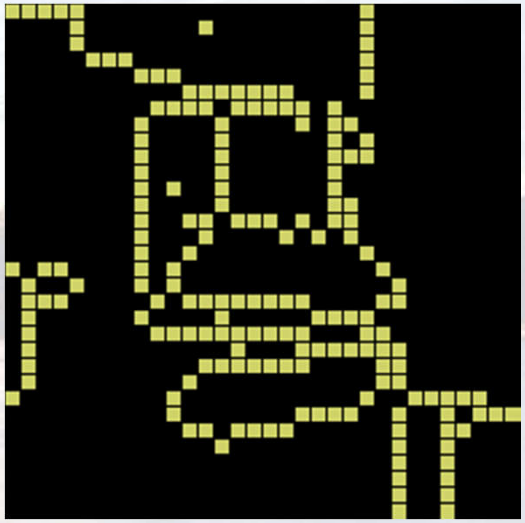
binary image



4-connected SE



8-conn outside bdry.



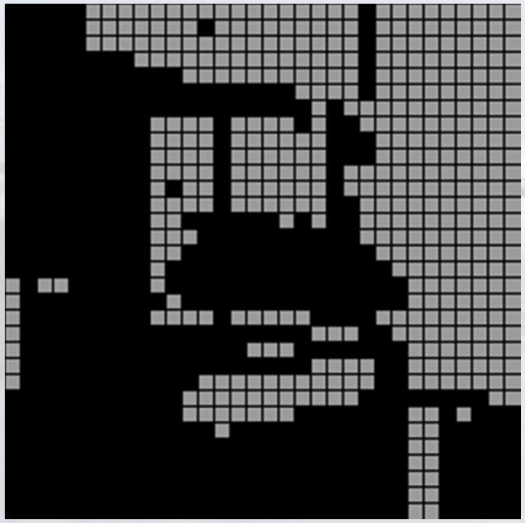
original

dilation by plus

difference

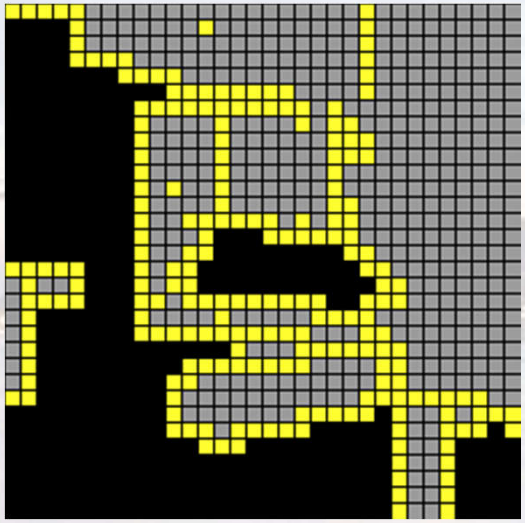
Boundary extraction

binary image



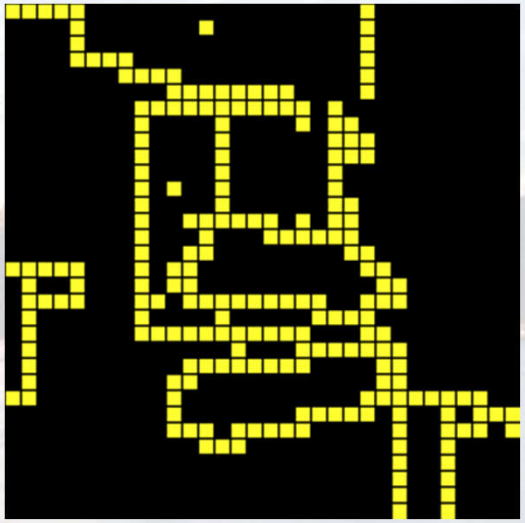
original

8-connected SE



dilation by square

4-conn outside bdry

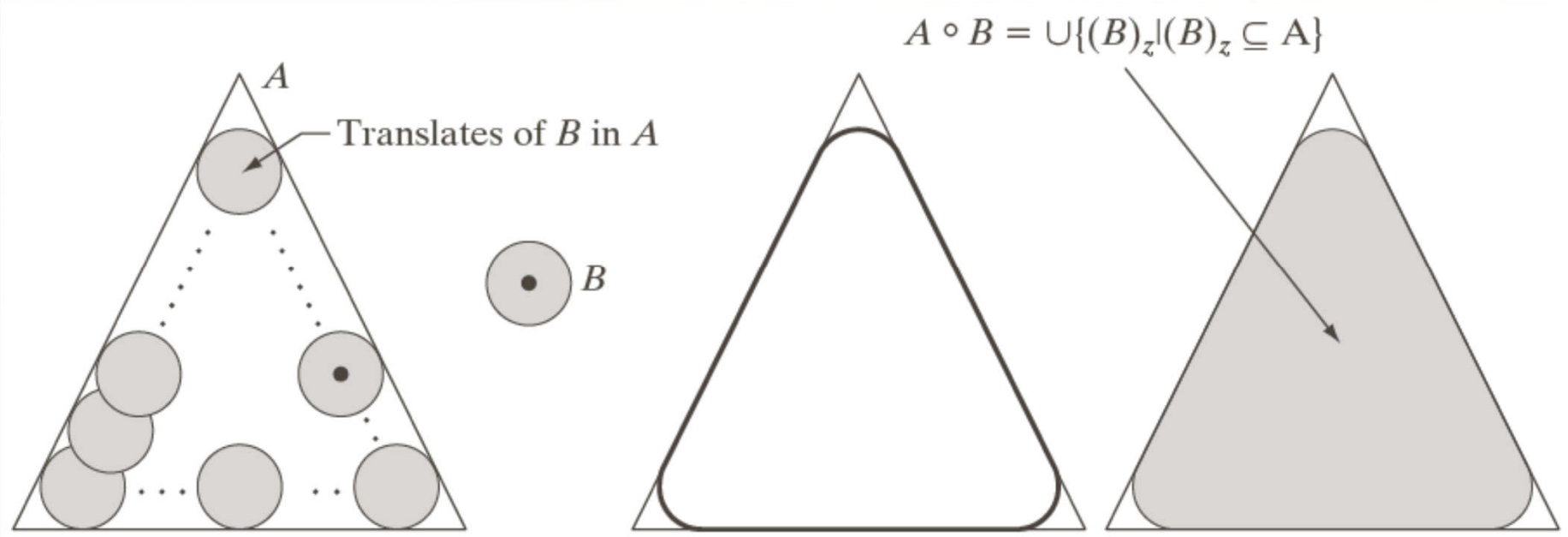


difference

The **opening** of a binary image A with a SE B is denoted as:

$$A \circ B = (A \ominus B) \oplus B$$

generally smoothens the contour of an object, breaks narrow connections, and eliminates thin protrusions

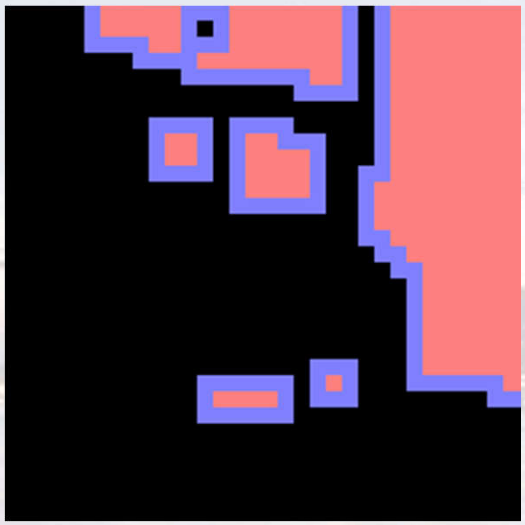


Opening is erosion followed by dilation

erode the original

erosion / original

dilate the erosion



erosion / opening

dilated erosion

opening / original

SE = Z_8

Opening is erosion followed by dilation

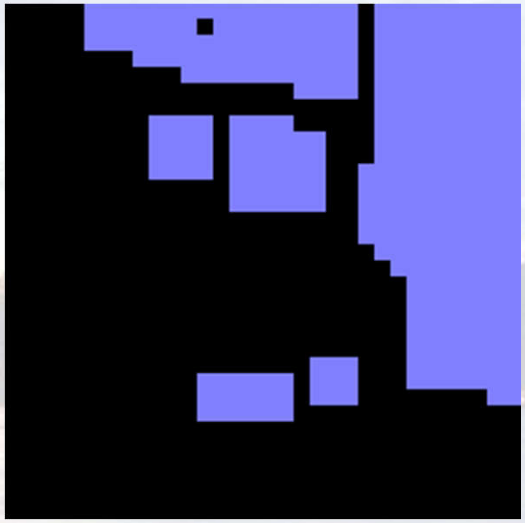
original image

original

eroded image

erosion

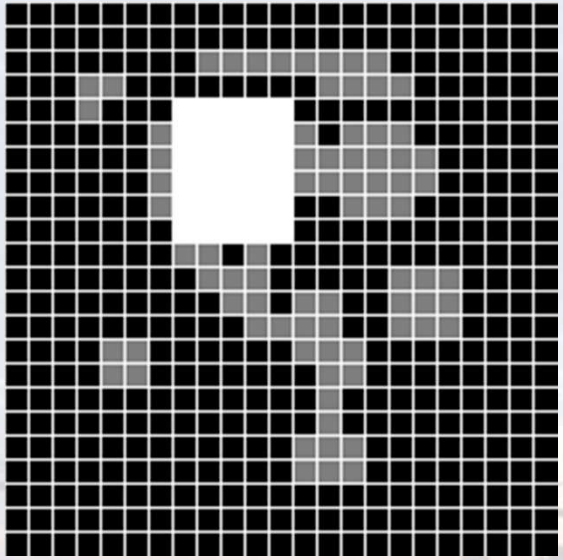
dilated erosion

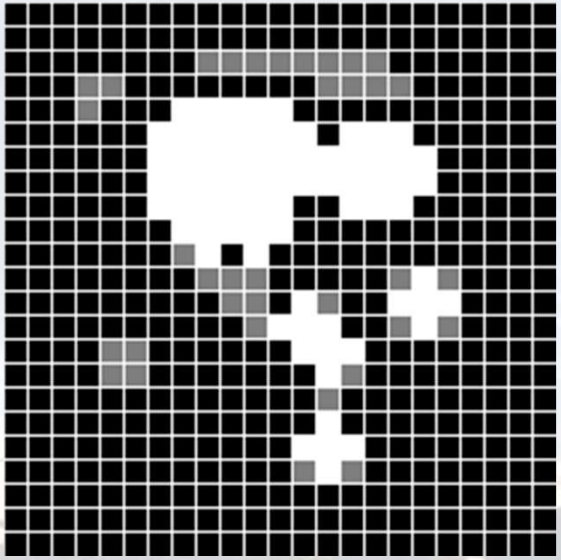


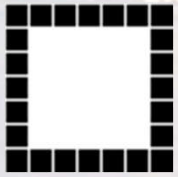
opening

SE = Z_8

Effect of SE shape





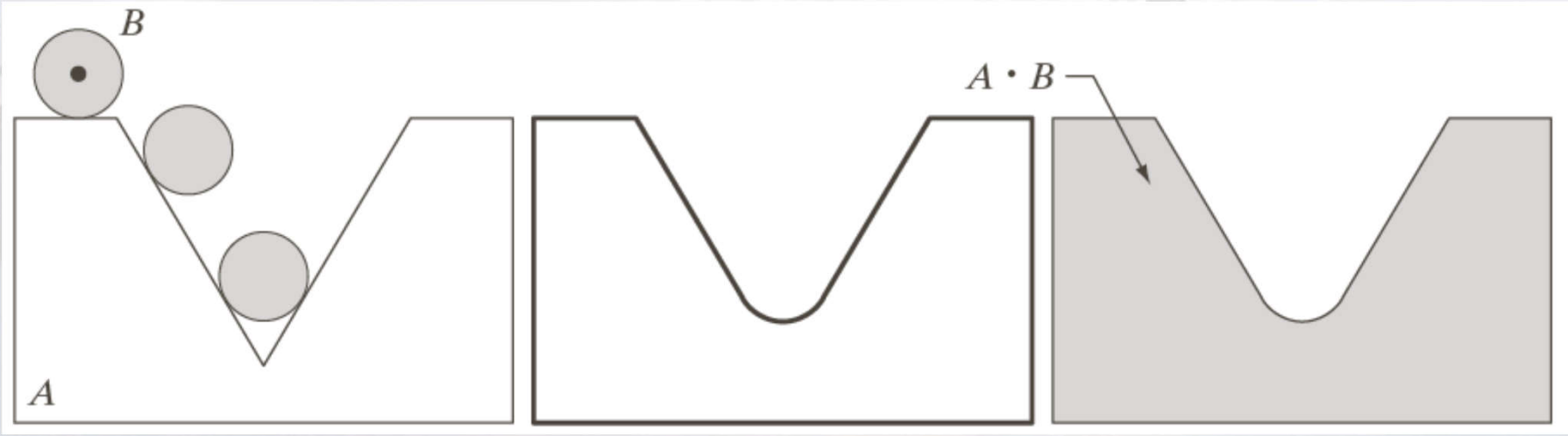


Less destructive than erosion, useful for removing unwanted foreground elements, while “somewhat” preserving the rest.

The **closing** of a binary image A with a SE B is denoted as:

$$A \bullet B = (A \oplus B) \ominus B$$

generally fills holes, eliminates gaps and fuses breaks.



Closing is dilation followed by erosion

original image

original / dilation

erode the dilation

closing / dilation

to get the closing

closing / original

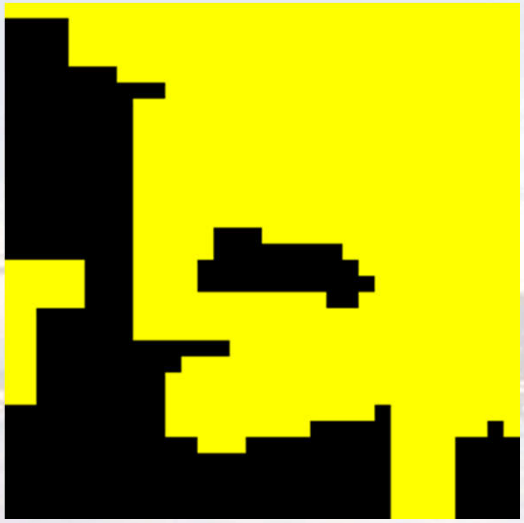
SE = Z_8

Closing is dilation followed by erosion

original image

original

dilated image



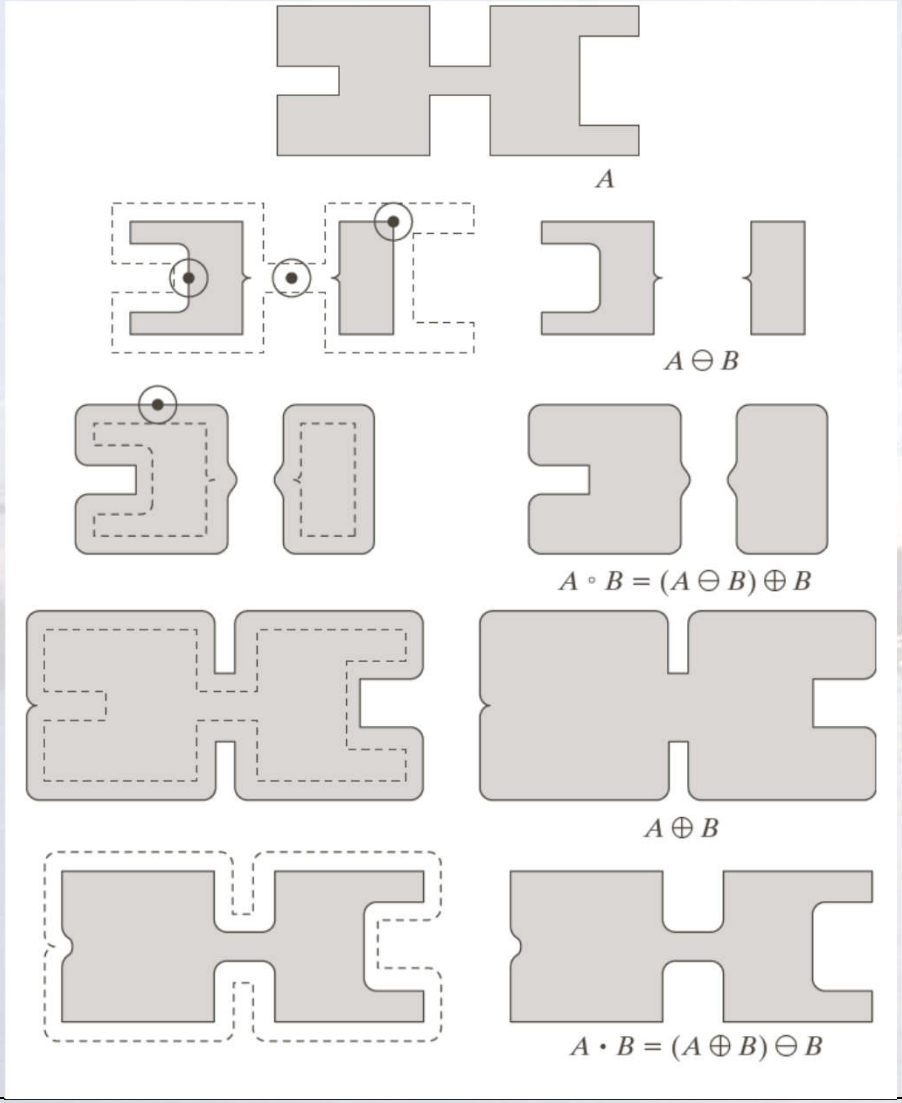
dilation

eroded dilation

closing

SE = Z_8

Closing is dilation followed by erosion



a
b c
d e
f g
h i

FIGURE 9.10 Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The SE was not shaded here for clarity. The dark dot is the center of the structuring element.

a b
c d
e f

FIGURE 9.11
(a) Noisy image. (b) Structuring element. (c) Eroded image. (d) Opening of A. (e) Dilation of the opening. (f) Closing of the opening. (Original image courtesy of the National Institute of Standards and Technology.)

Properties of **opening**

Increasing $A \subseteq C \Rightarrow A \circ B \subseteq C \circ B$

Anti-extensive $A \ominus B \subseteq A \circ B \subseteq A$

Idempotent $(A \circ B) \circ B = A \circ B$

Properties of **closing**

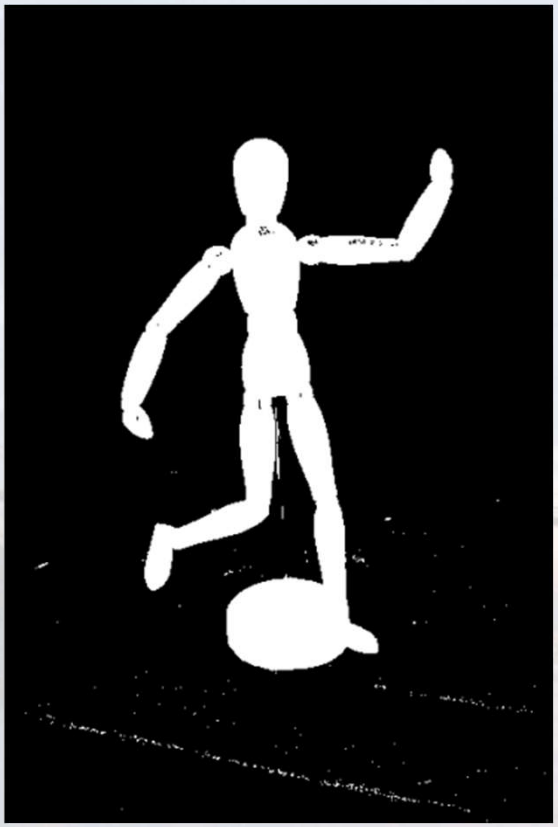
Increasing $A \subseteq C \Rightarrow A \bullet B \subseteq C \bullet B$

Extensive $A \subseteq A \bullet B \subseteq A \oplus B$

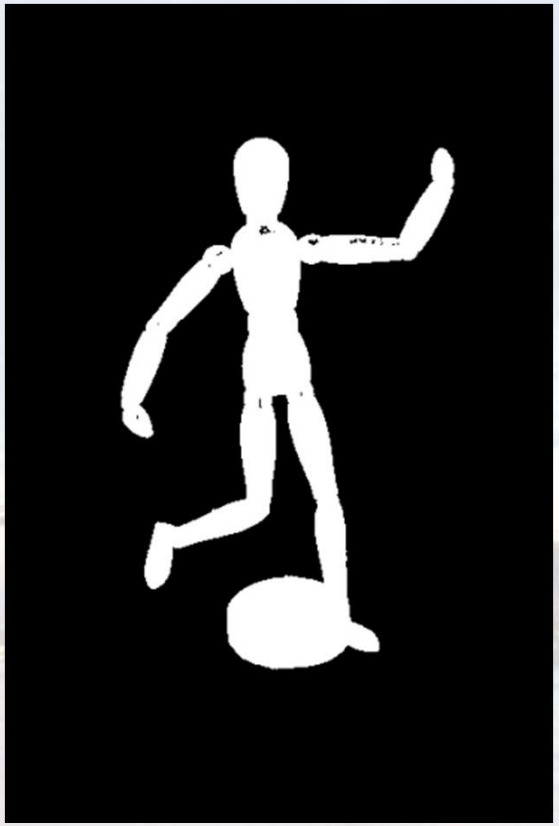
Idempotent $(A \bullet B) \bullet B = A \bullet B$

Opening and closing are also dual

$$A \bullet B = (A^c \circ B)^c$$
$$A \circ B = (A^c \bullet B)^c$$



original

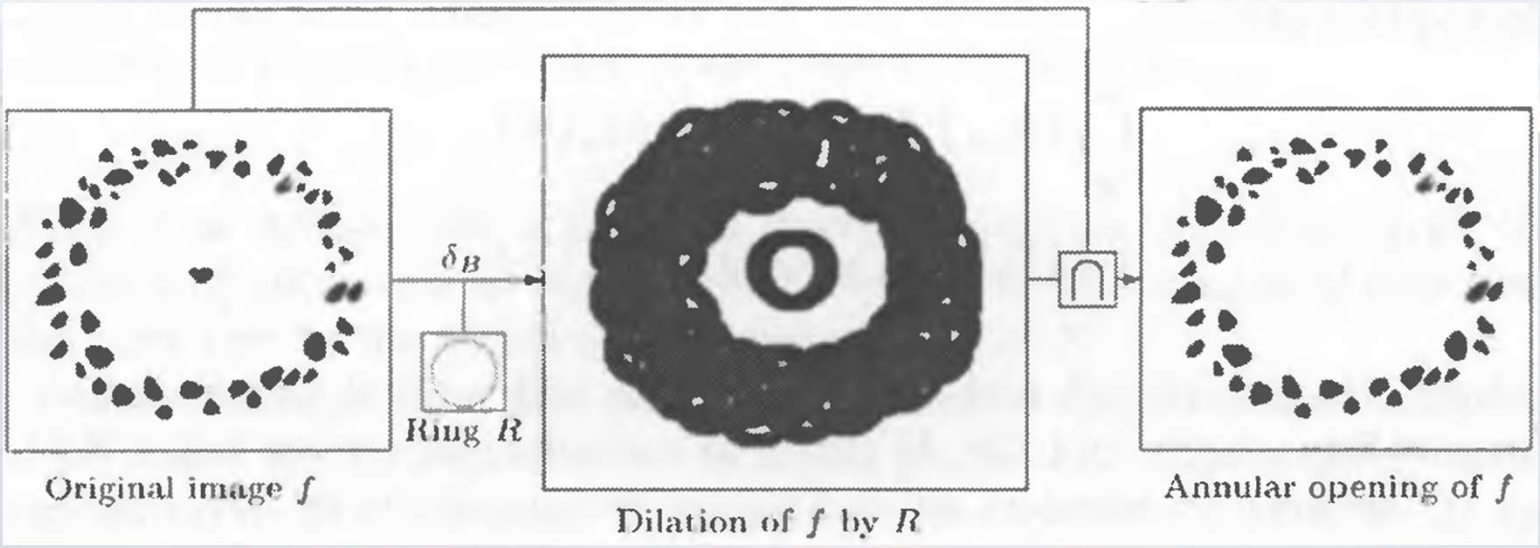


opening

closing

An annular opening

It's computed as the intersection of between the dilation of an image f with a ring (or circle) shaped SE and the original image. Since the SE does not include the origin, the input image is not included in the dilation output, hence this is dilation is not extensive.



Adapted from P. Soille

It's useful for extracting clusters of objects at predefined radii.

Conditional operator example: **Region filling**

a b c

FIGURE 9.16 (a) Binary image (the white dot inside one of the regions is the starting point for the hole-filling algorithm). (b) Result of filling that region. (c) Result of filling all holes.

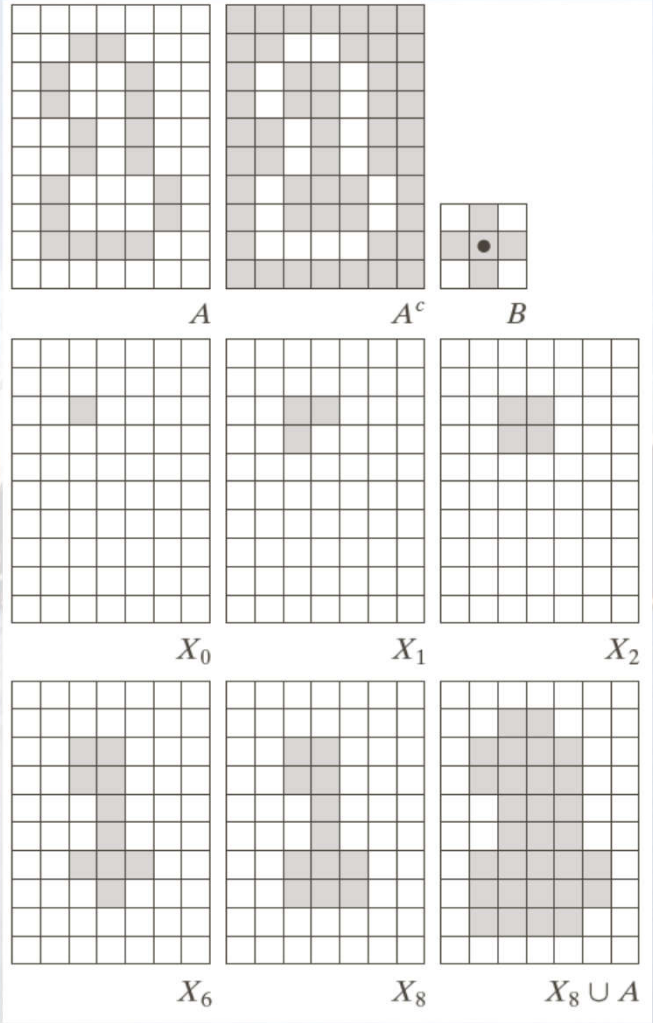
Let A represent an image containing a region's boundary, and a point p inside the boundary.

The following fills that region:

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

Where $X_0 = p$ and B is the cross SE.

The process terminates when $X_k = X_{k-1}$



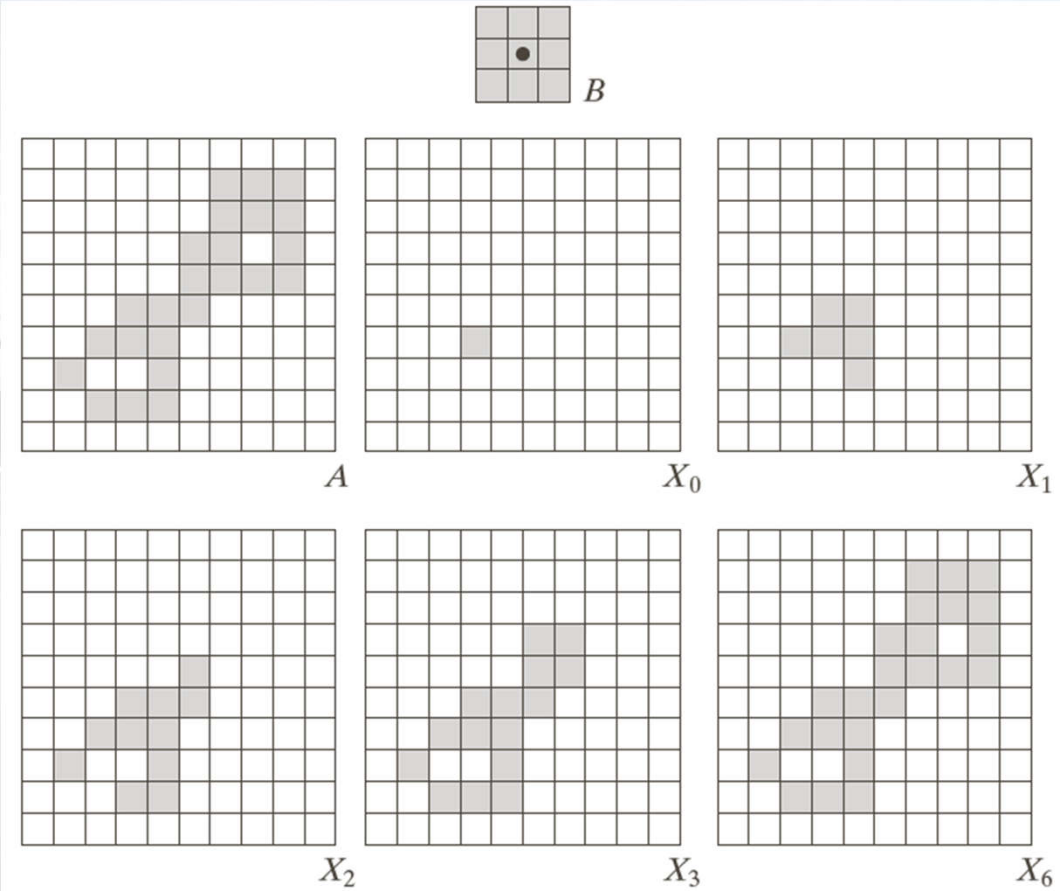
Using a similar logic we can even **extract connected components**.

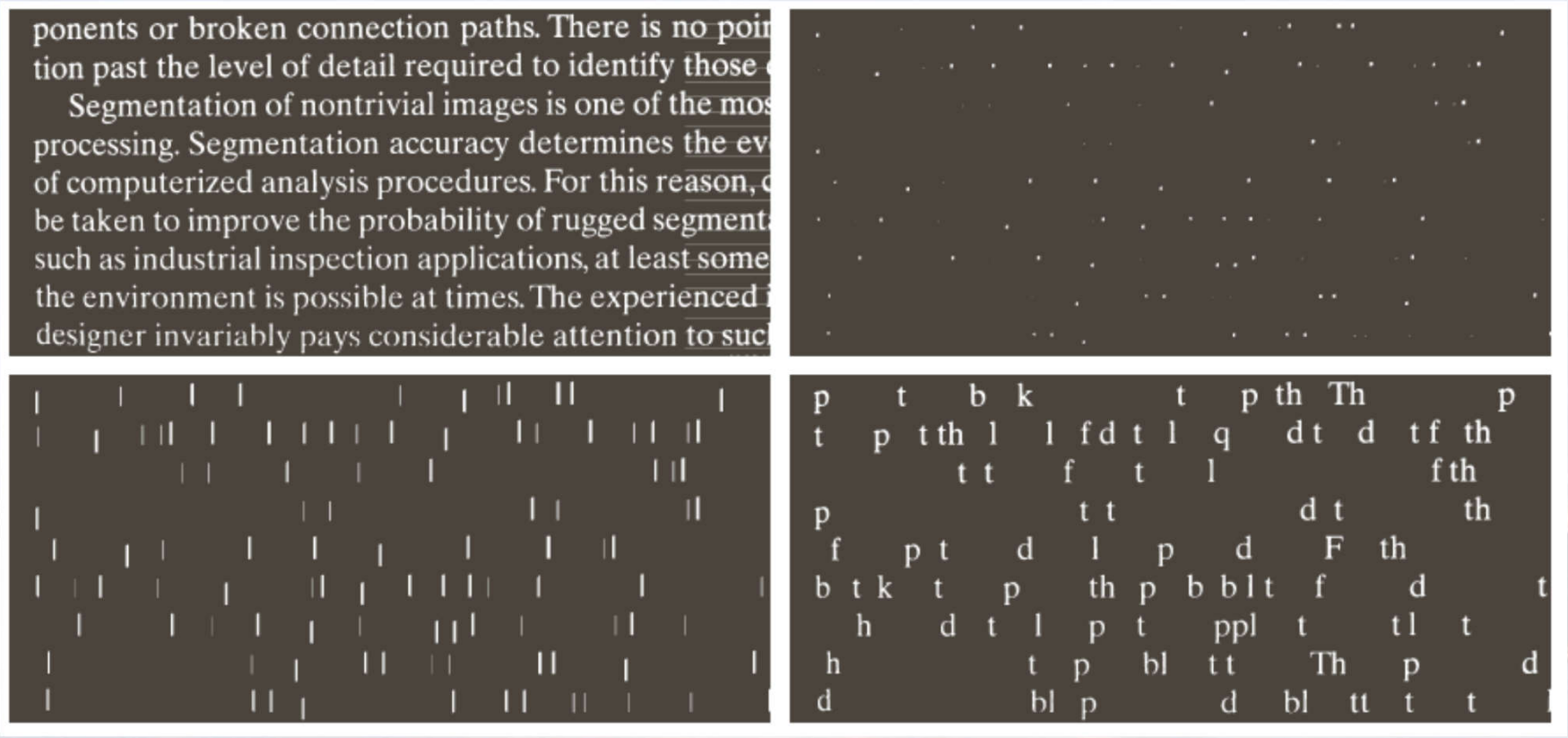
Let A represent an image containing a point p belonging to the CC. The following will reconstruct that CC:

$$X_k = (X_{k-1} \oplus B) \cap A, k = 1,2,3,..$$

where $X_0 = p$, B is the SE
and the termination criterion
is stability: $X_k = X_{k-1}$

Does the shape of the SE matter?
What about its size?





a b
c d

FIGURE 9.29 (a) Text image of size 918 × 2018 pixels. The approximate average height of the tall characters is 50 pixels. (b) Erosion of (a) with a structuring element of size 51 × 1 pixels. (c) Opening of (a) with the same structuring element, shown for reference. (d) Result of opening by reconstruction.

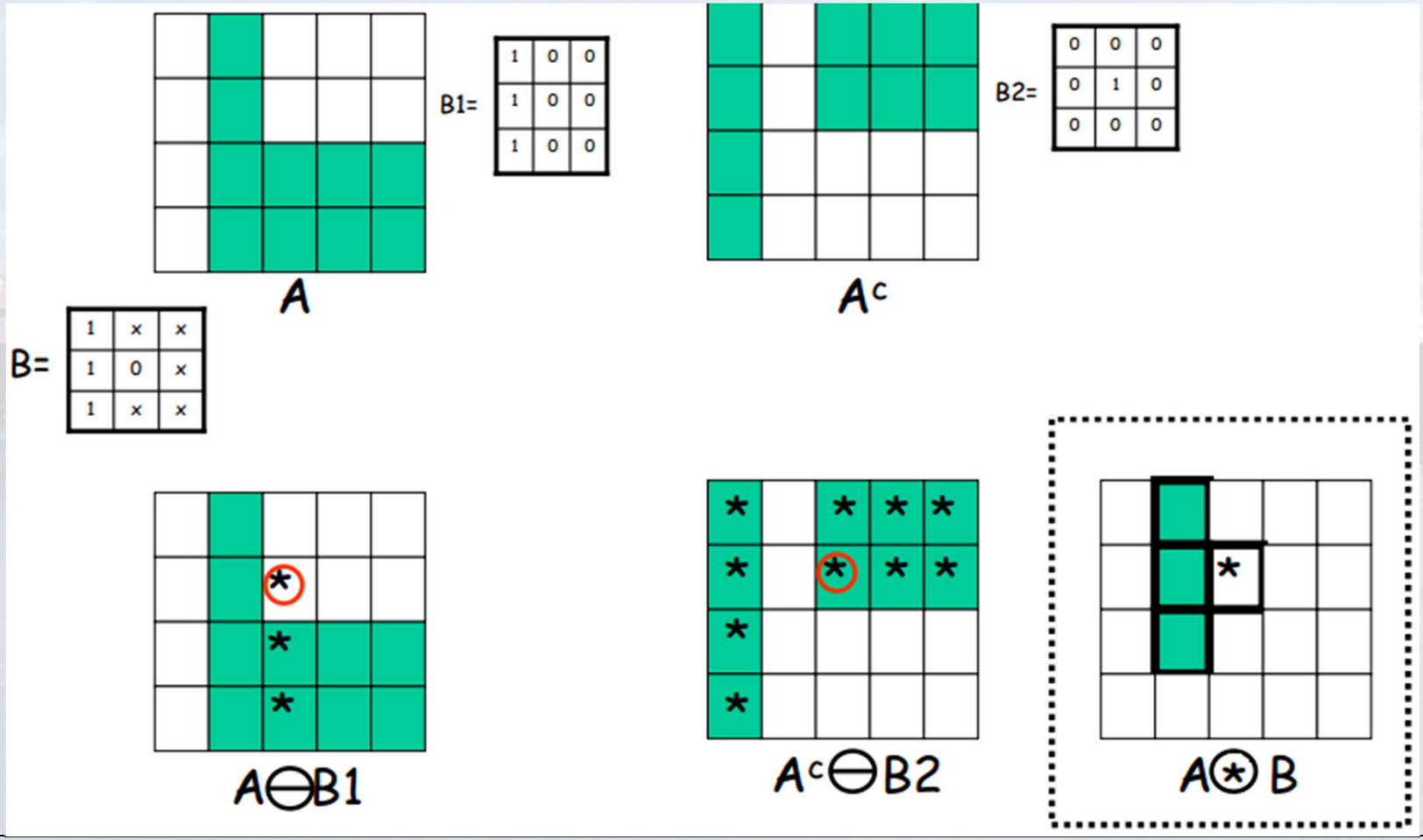
The **Hit-or-Miss transform** is a template matching tool.

Let $B = (B_1, B_2)$ denote a compound SE representing the template, where B_1 corresponds to its foreground and B_2 to its background. The Hit-or-Miss transform of A with B is denoted by:

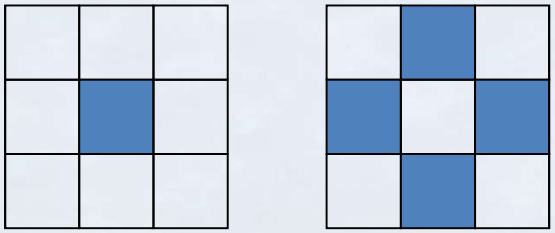
$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

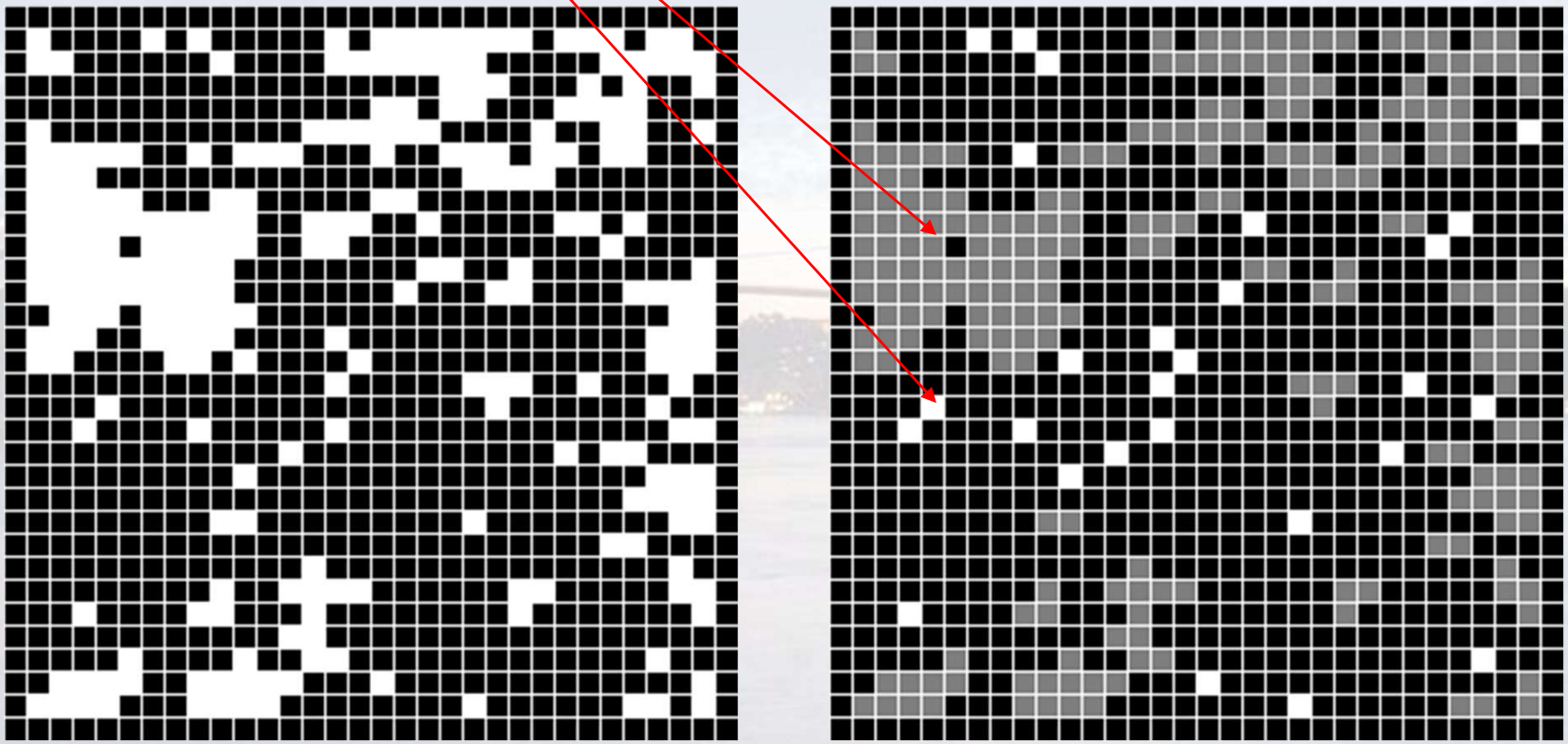
It contains all the points that simultaneously fit both the foreground ("hit") B_1 and the background ("miss") B_2.

The hit-or-miss transform works well and fast, but only if you know EXACTLY what you are looking for, and want to find EXACTLY that template.



isolated points at
4 connectivity

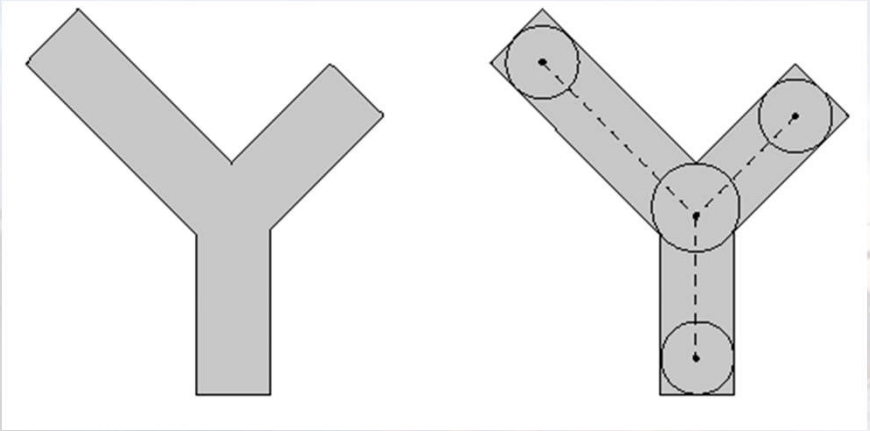


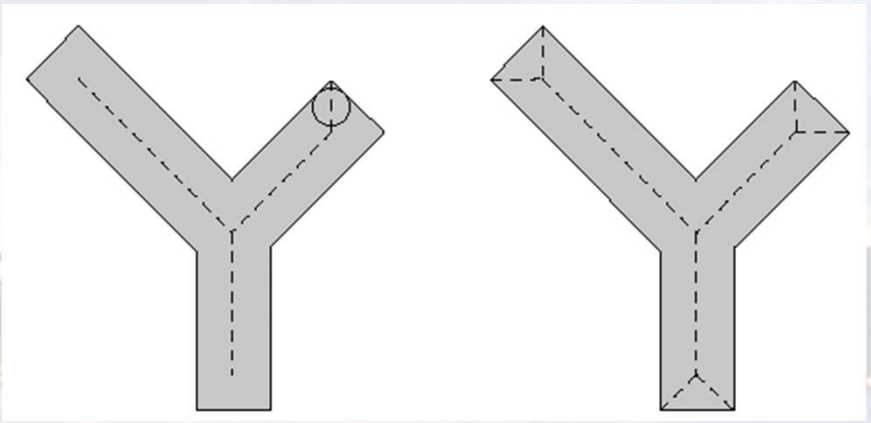


Maximal disc: a disc contained in an object, in such a way that no other disc contains it.

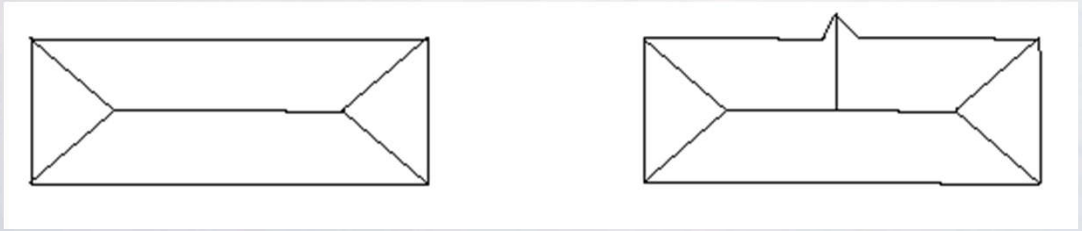
Skeleton of an object: the union of the centers of its maximal discs.

Computable with a **thinning**, useful for compression and description, though very unstable.





The slightest border variation causes large deviations

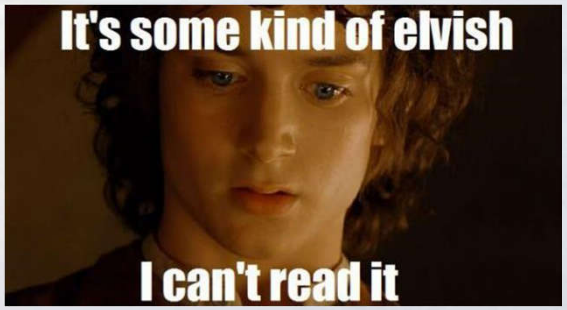


The **thinning** of an image A with a compound SE B is denoted by:

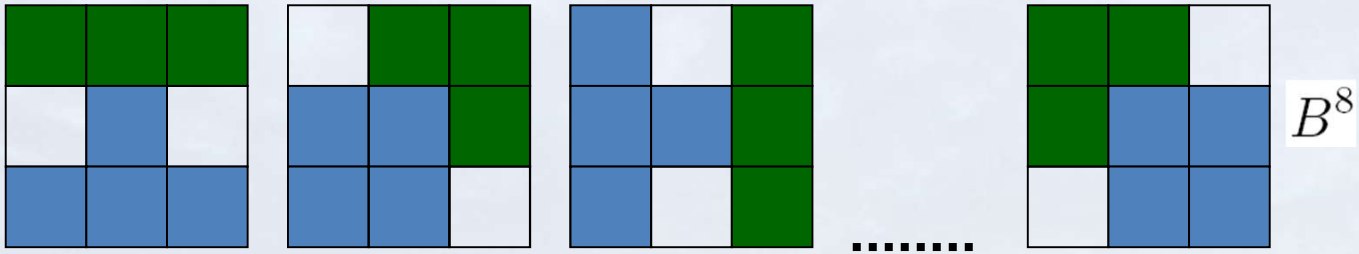
$$A \otimes B = A - (A \circledast B)$$
$$= A \cap (A \circledast B)^c$$

It removes from the image the points that have matched the SE.

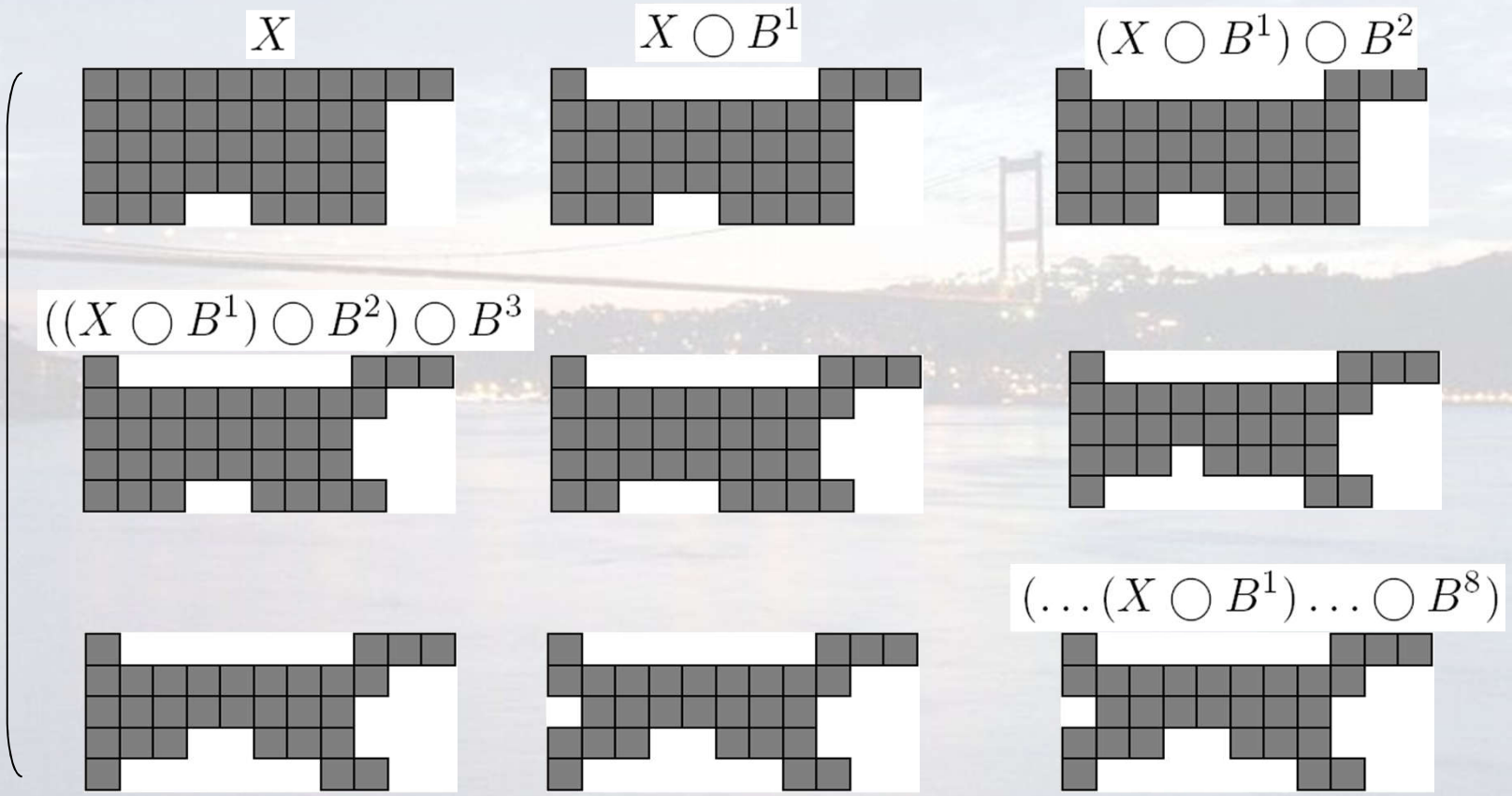
And the dual of thinning is **thickening**: $A \odot B = A \cup (A \circledast B)$

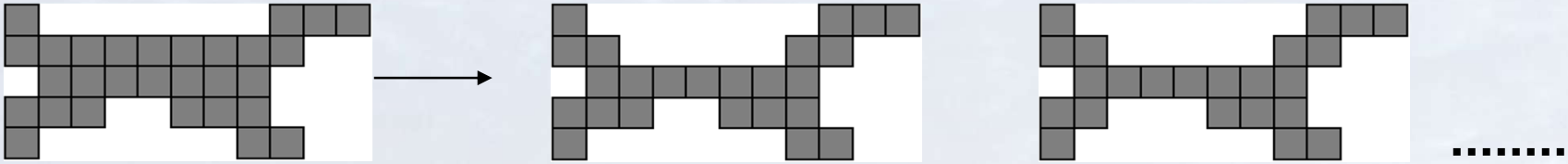


Skeleton through thinning:



1st iteration

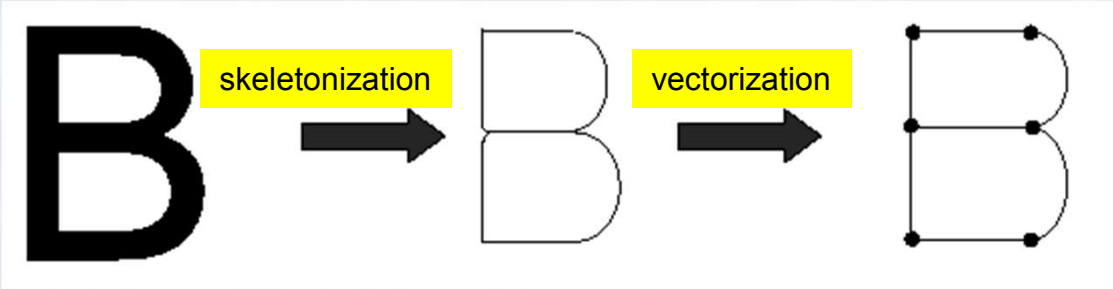


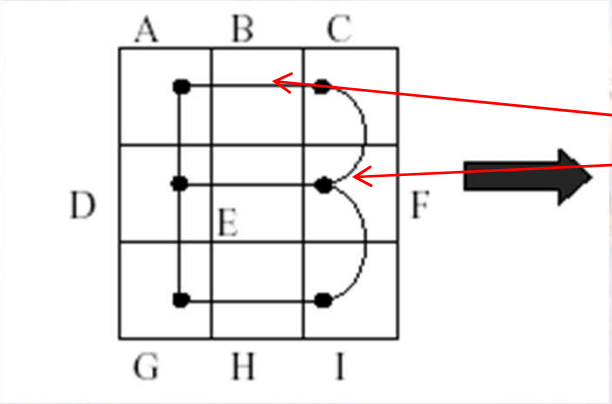


result of
1st iteration

2nd iteration reaches
idempotence

Application of **skeletonization** to OCR through graph matching.





ArC, ArD, CcF, DrF, DrG, FcI, GrI

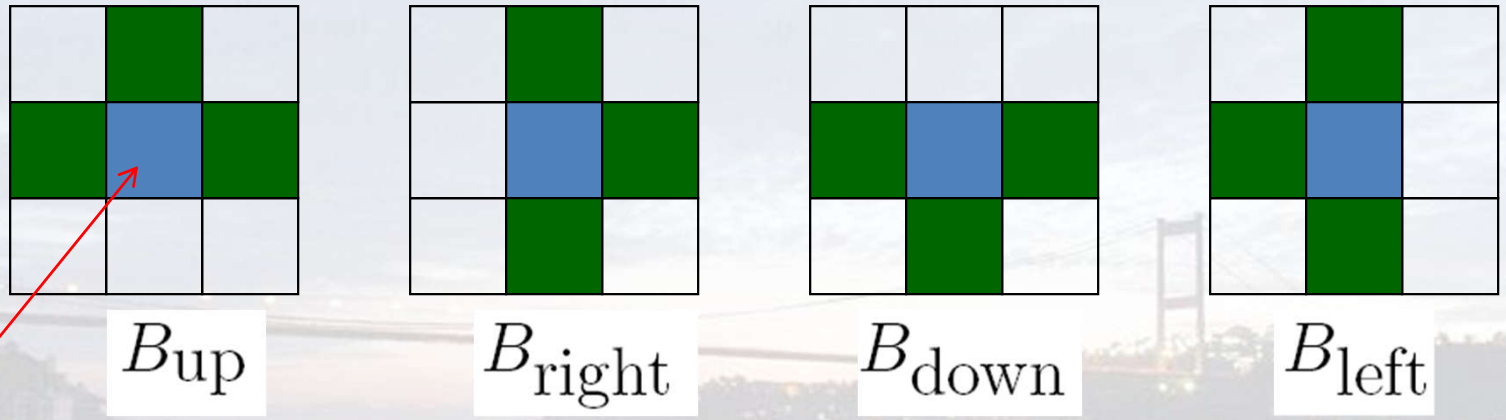
on r = recta, c = corba.

Application of **skeletonization** to OCR through graph matching.

Hit-or-Miss

and 3 rotations

Pruning with thinning; removing end points.

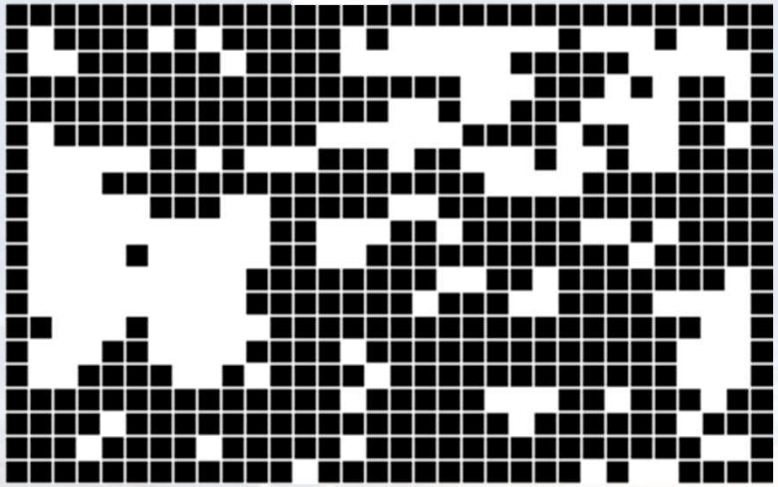


B_{up} B_{right} B_{down} B_{left}

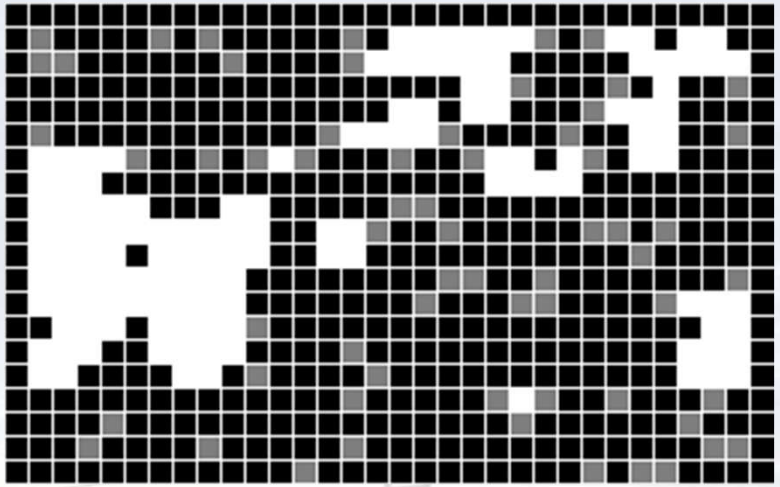
This point is removed with dark green neighbors

1 iteration = $(((X \bigcirc B_{\text{up}}) \bigcirc B_{\text{right}}) \bigcirc B_{\text{down}}) \bigcirc B_{\text{left}}$

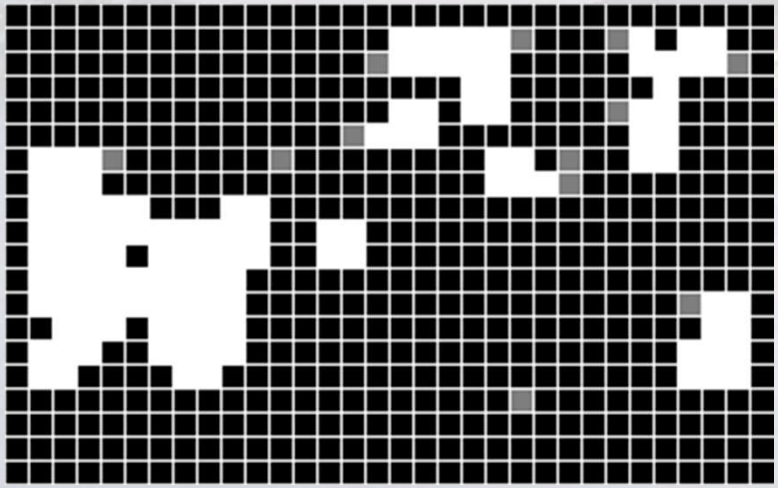
X



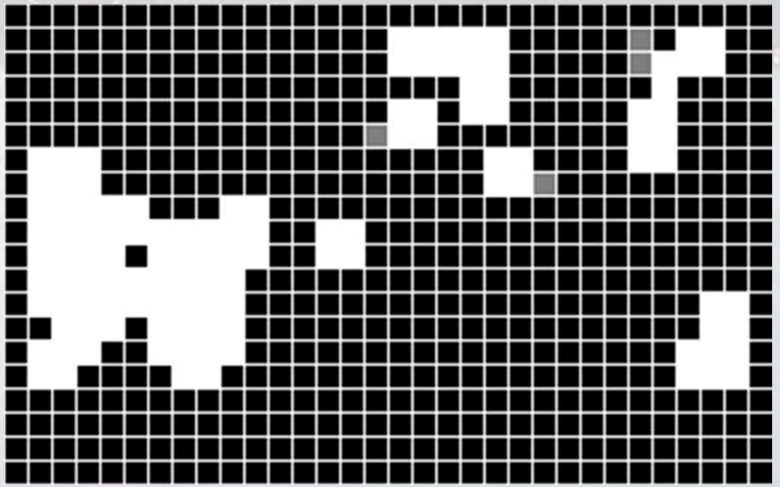
1st iteration



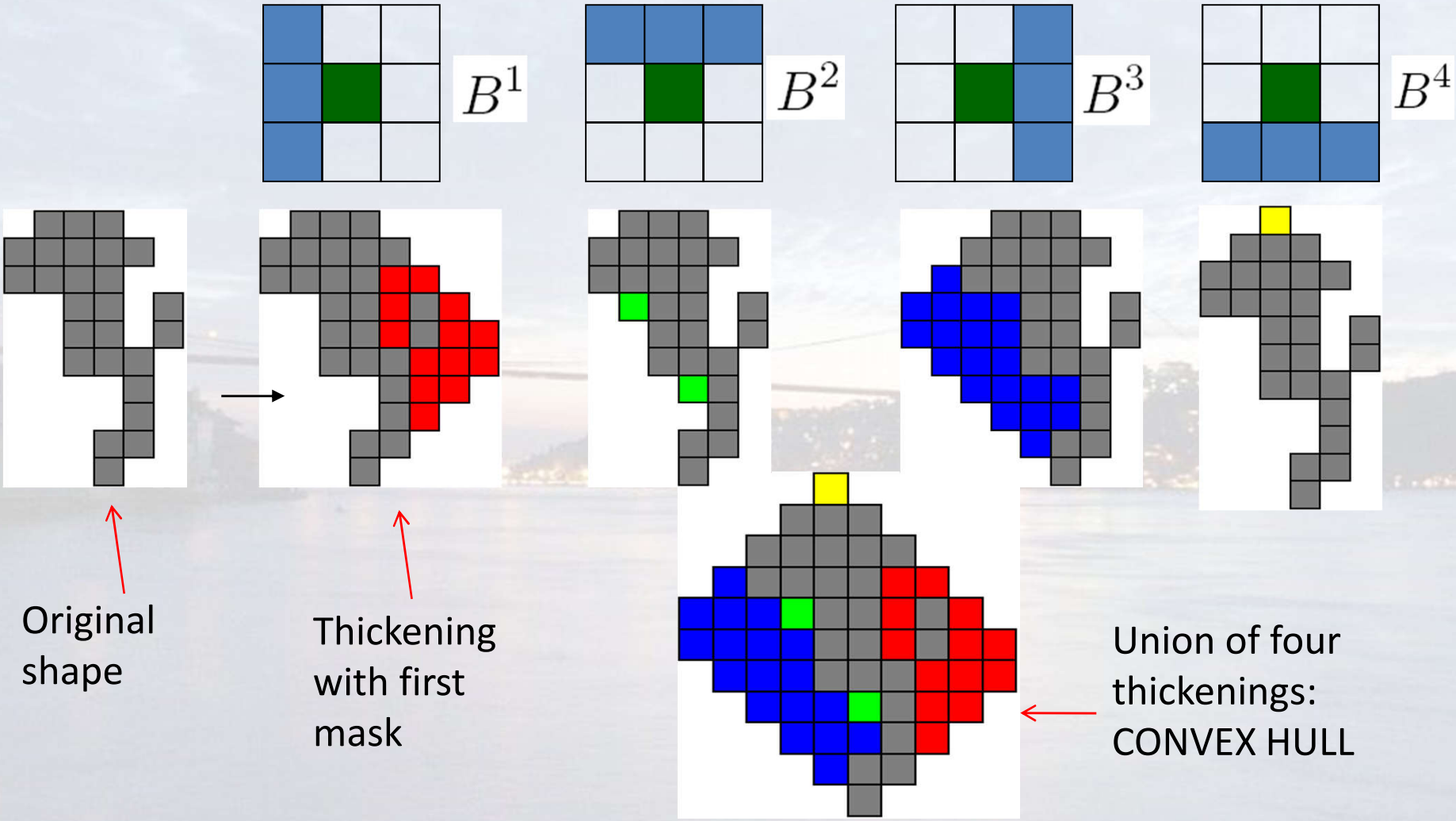
2nd iteration



3rd iteration: idempotence



Application of thickening to **convex hull** computation.



Example of convex hull calculation with thickenings.