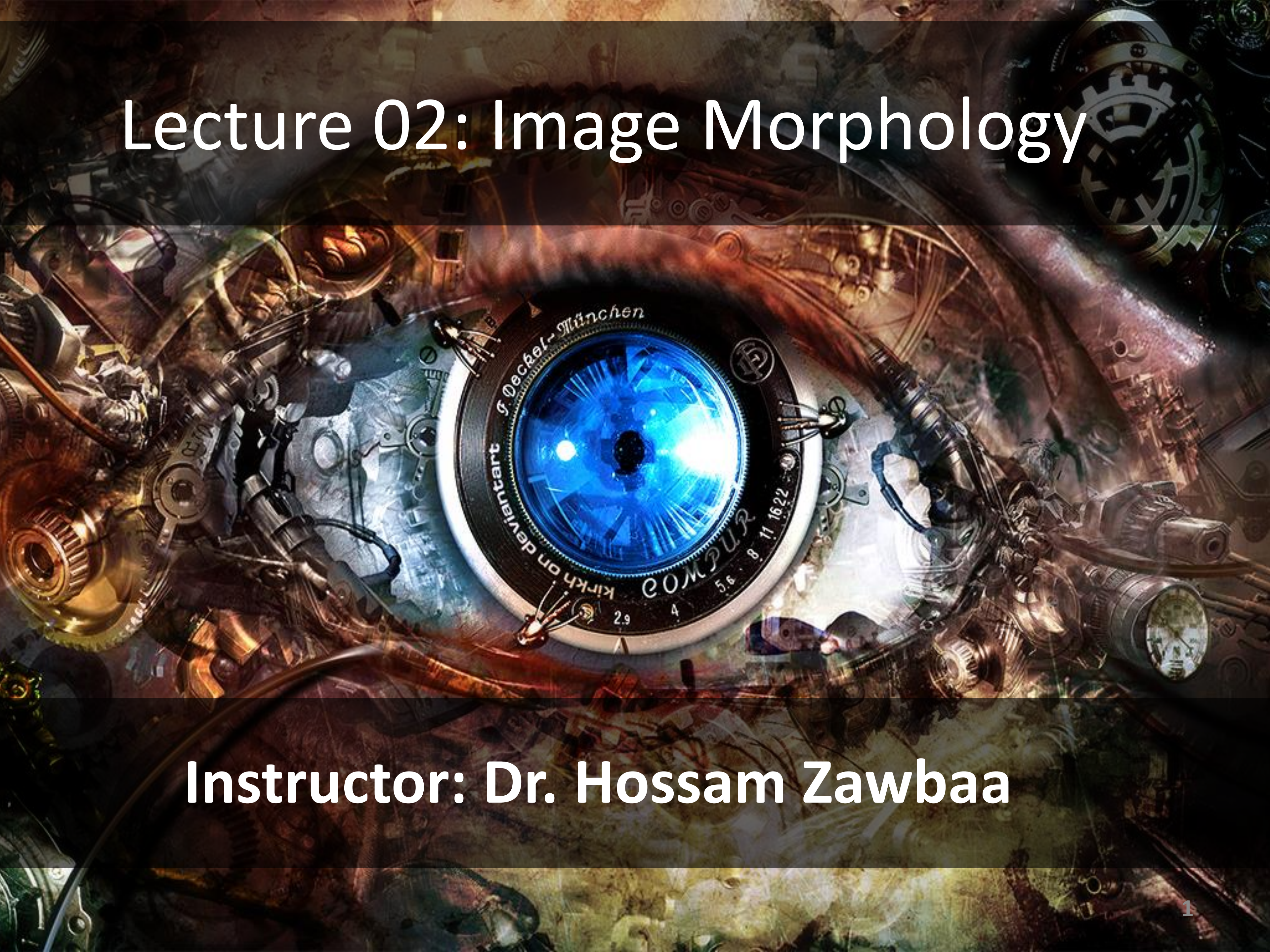


Lecture 02: Image Morphology

Instructor: Dr. Hossam Zawbaa

Introduction

- **Morphology**: a branch of biology that deals with the form and structure of animals and plants
- **Morphological Operators**
  - Used generally on **binary images**, e.g., background subtraction results!
- Good for
  - Noise removal in background
  - Removal of holes in foreground / background
- **Morphological image processing** is used to extract image components for representation and description of region shape, such as boundaries and skeletons

Introduction

- Structuring Element
- Erosion
- Dilation
- Opening
- Closing
- Hit-and-miss Operation
- Thinning
- Thickening

Structuring Element

Small sets or sub-images used to explore an image under study for properties of interest

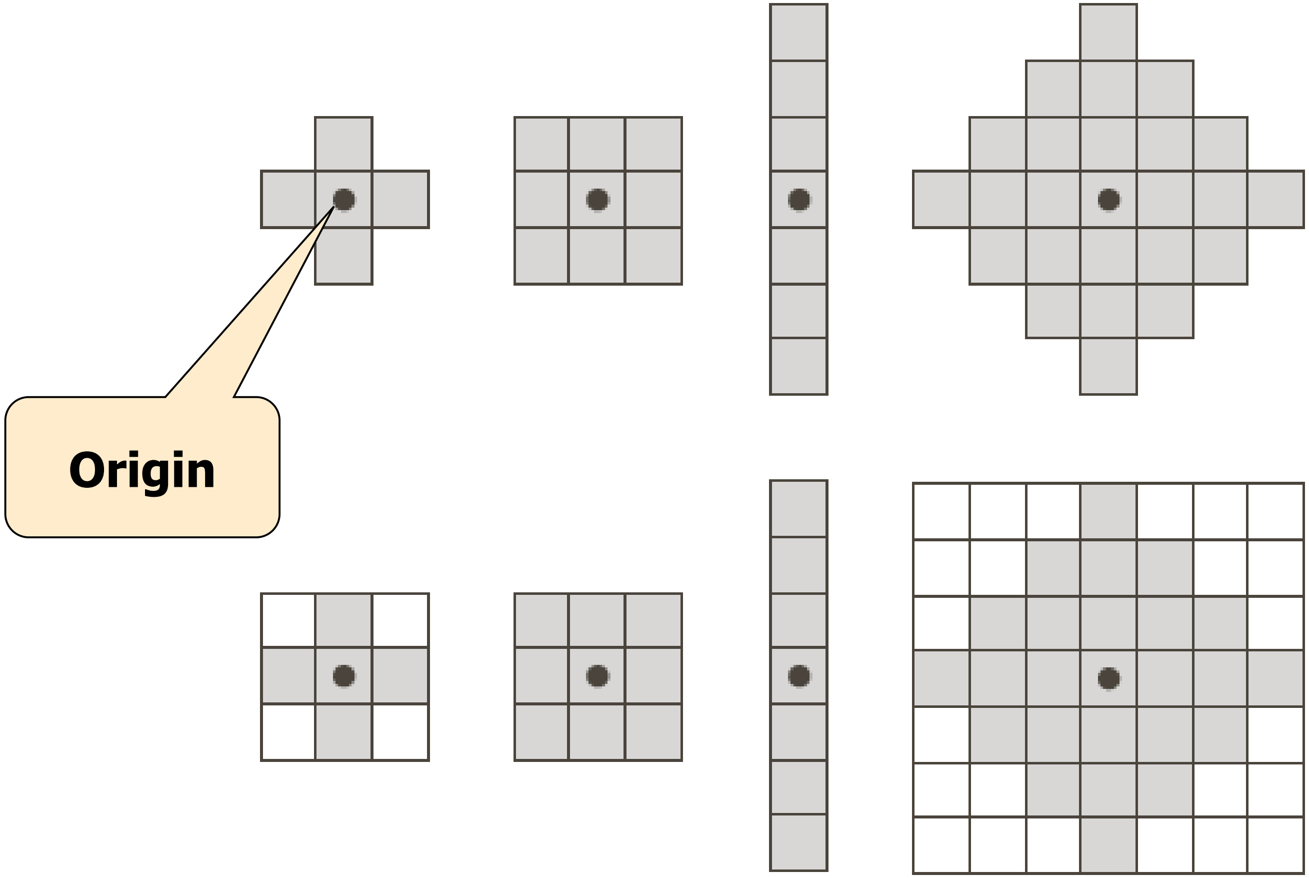
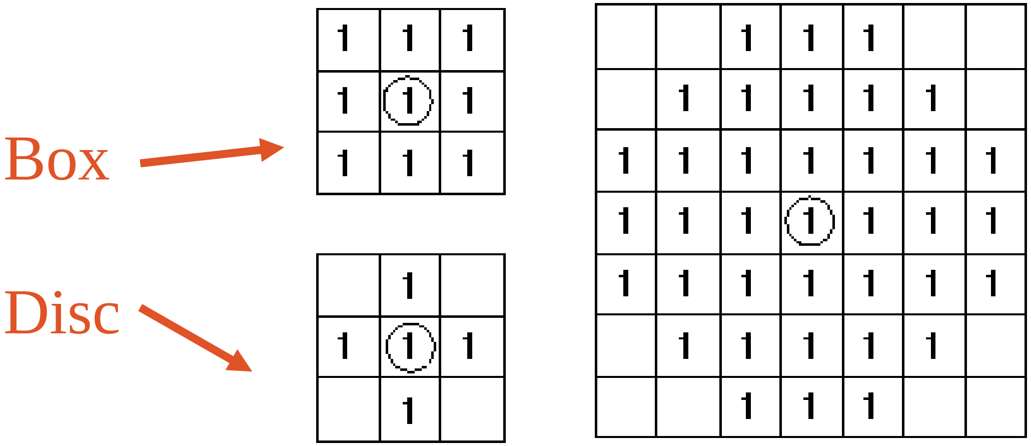


FIGURE 9.2 First row: Examples of structuring elements. Second row: Structuring elements converted to rectangular arrays. The dots denote the centers of the SEs.

Structuring Element

- Structuring Elements can have varying sizes
- Usually, element values are 0, 1 and none(!)
- Structural Elements have an origin
- Empty spots in the Structuring Elements are ***don't care***'s!



1	1	1
1	①	1
1	1	1

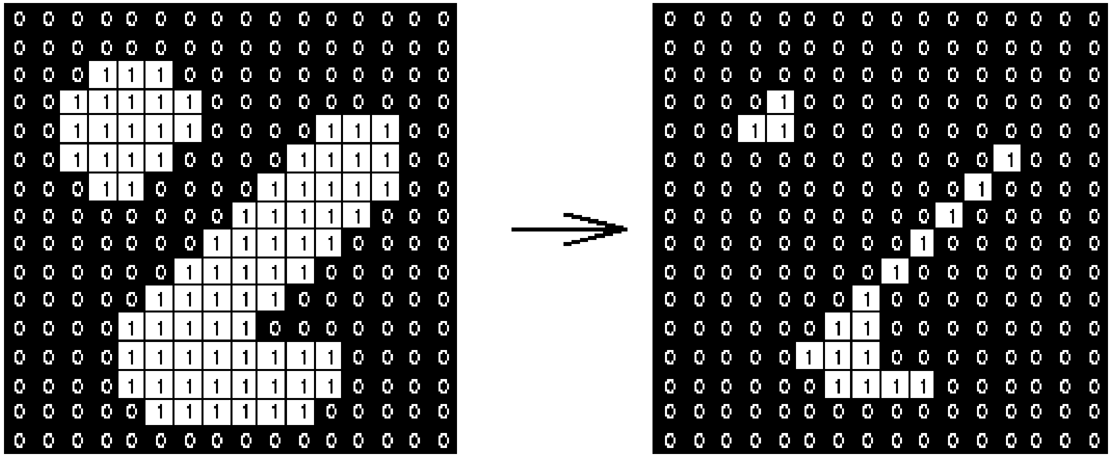
	1	
1	①	1
	1	

		1	1	1		
	1	1	1	1	1	
1	1	1	1	1	1	1
1	1	1	①	1	1	1
1	1	1	1	1	1	1
	1	1	1	1	1	
		1	1	1		

Examples of structuring elements

Erosion

- **Erosion** is an important morphological operation



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

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

- Applied **Structuring Element**:

1	1	1
1	1	1
1	1	1

Set of coordinate points =

{ (-1, -1), (0, -1), (1, -1),

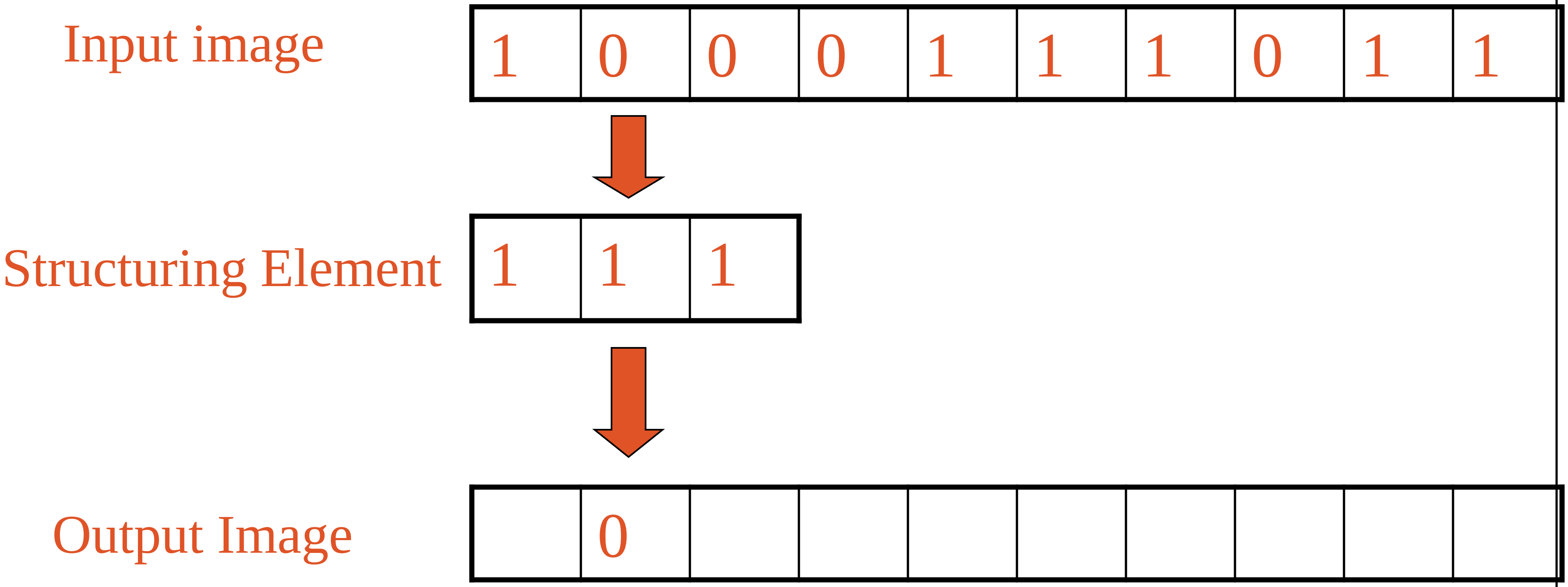
(-1, 0), (0, 0), (1, 0),

(-1, 1), (0, 1), (1, 1) }

Erosion

- **Erosion** is the set of all points in the image, where the structuring element "**fits into**".
- Consider each foreground pixel in the input image
  - If the structuring element fits in, write a "1" at the origin of the structuring element!
- Simple application of **pattern matching**
- **Input:**
  - **Binary Image (Gray value)**
  - **Structuring Element, containing only 1s!**

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

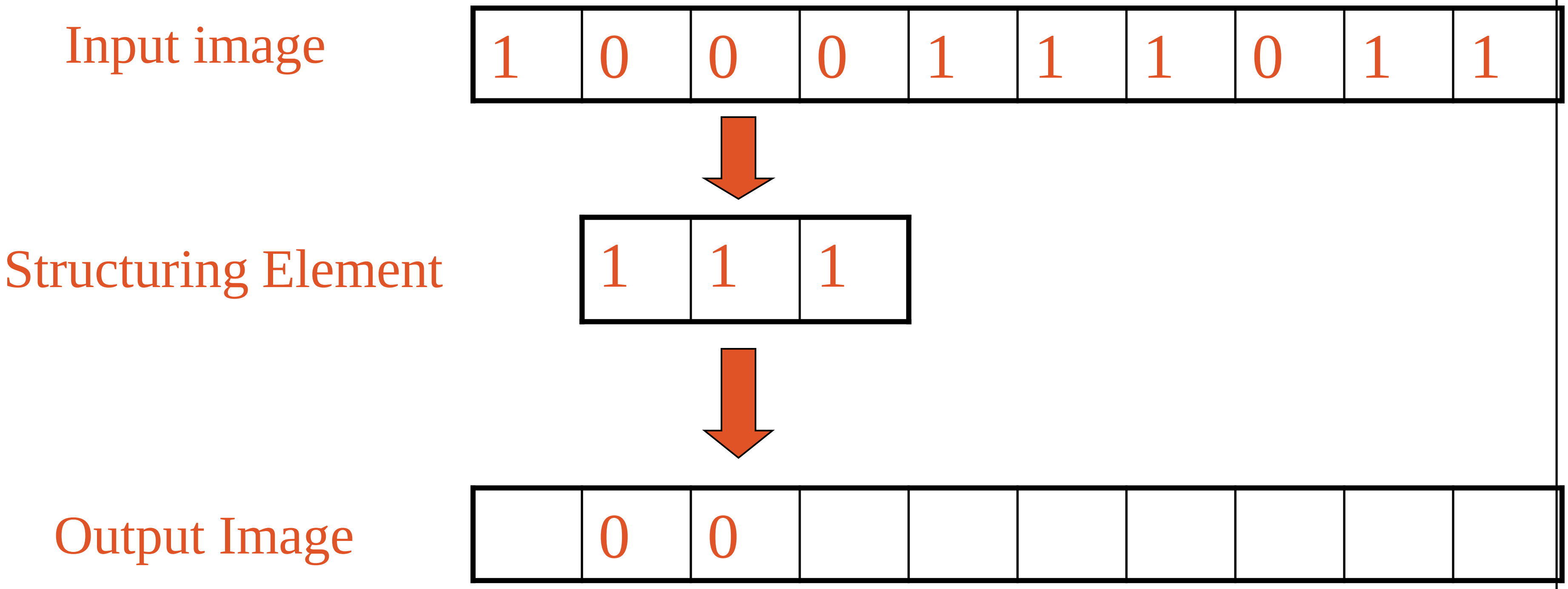
Structuring Element

1	1	1

Output Image

	0								

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

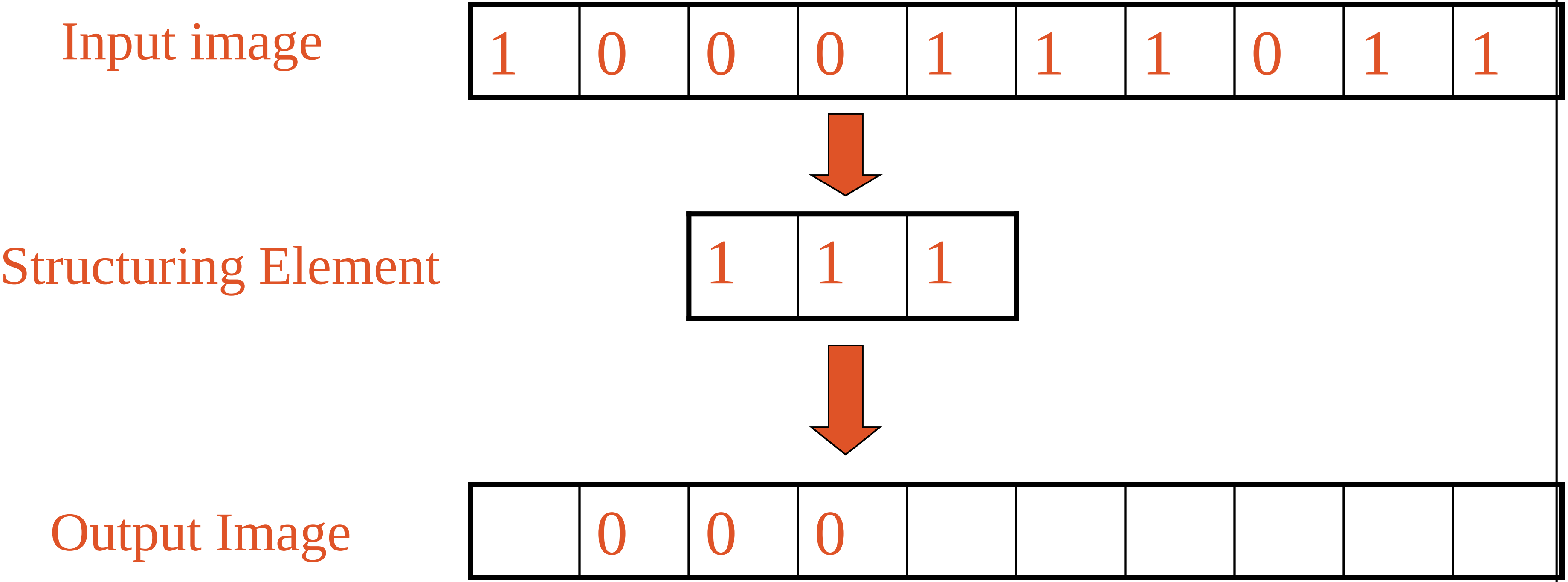
Structuring Element

1	1	1

Output Image

	0	0							

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

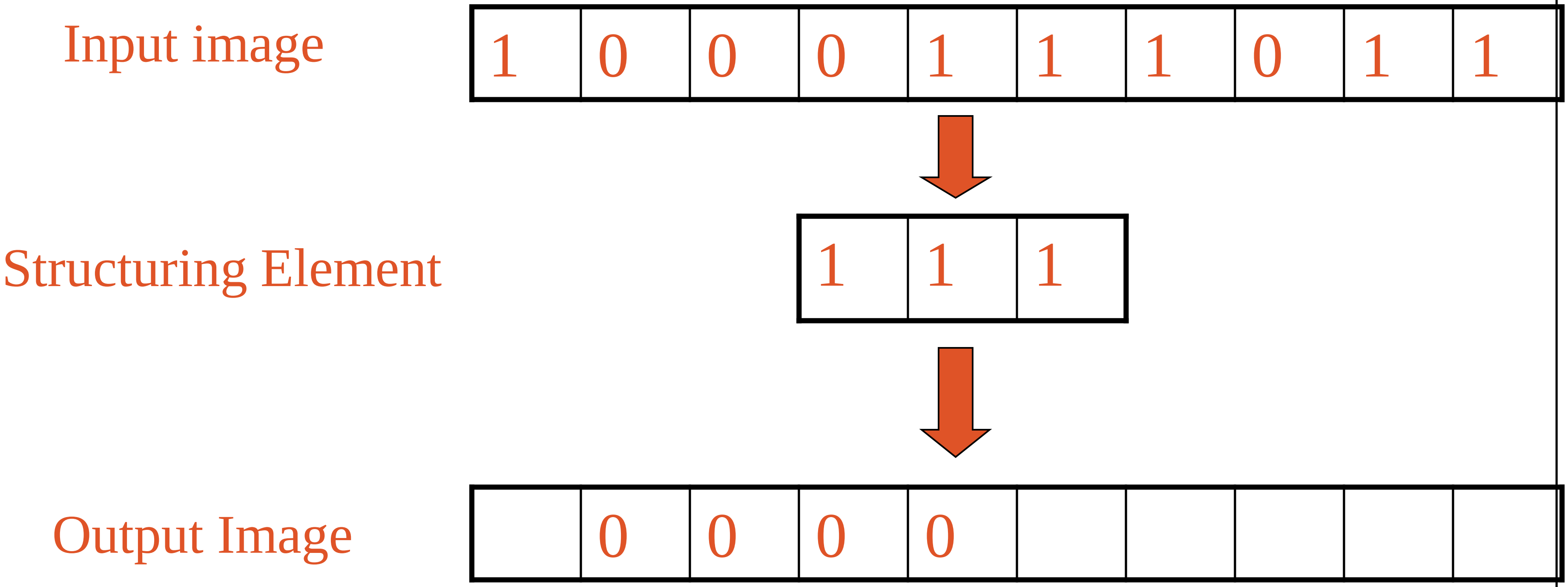
Structuring Element

1	1	1

Output Image

	0	0	0						

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

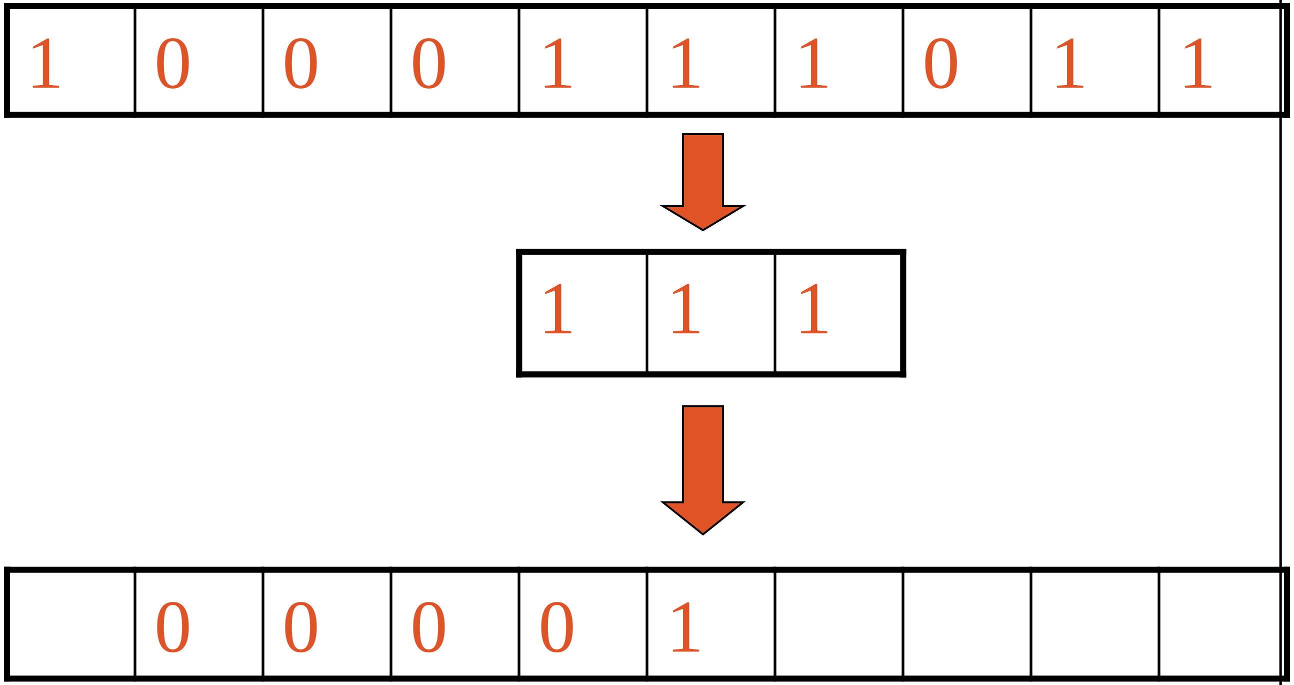
Structuring Element

1	1	1

Output Image

	0	0	0	0					

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

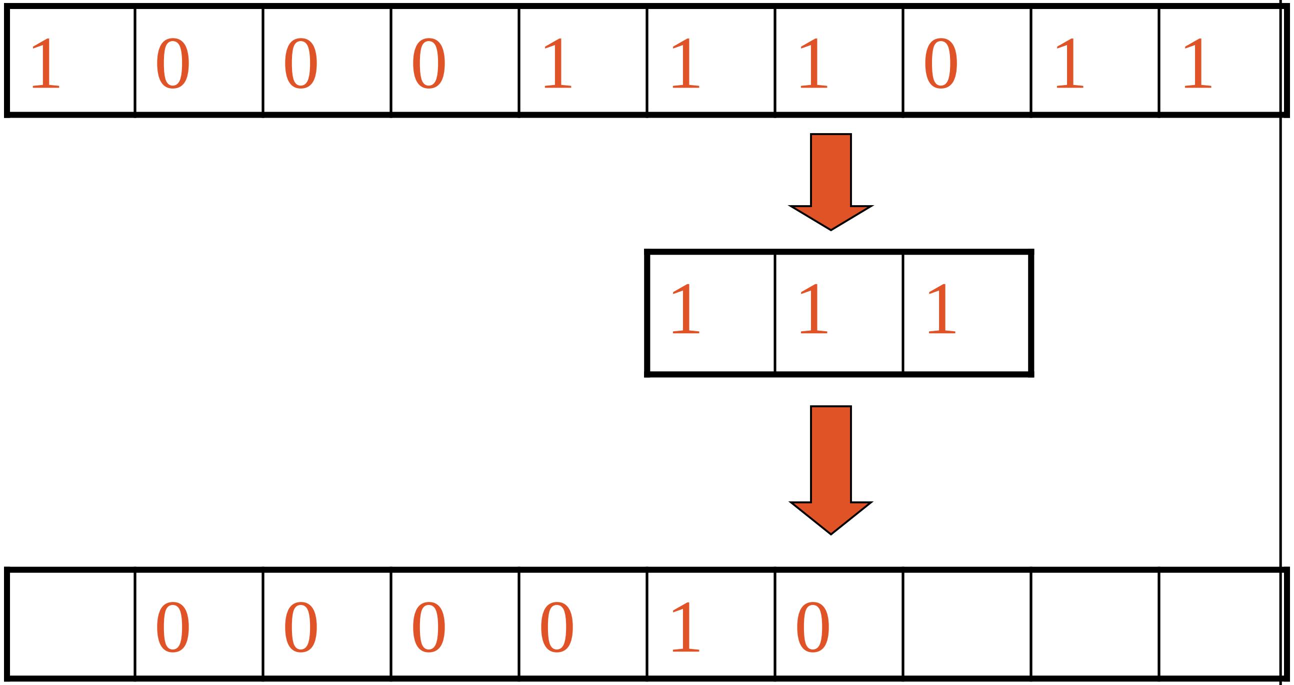
Structuring Element

1	1	1

Output Image

	0	0	0	0	1				

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

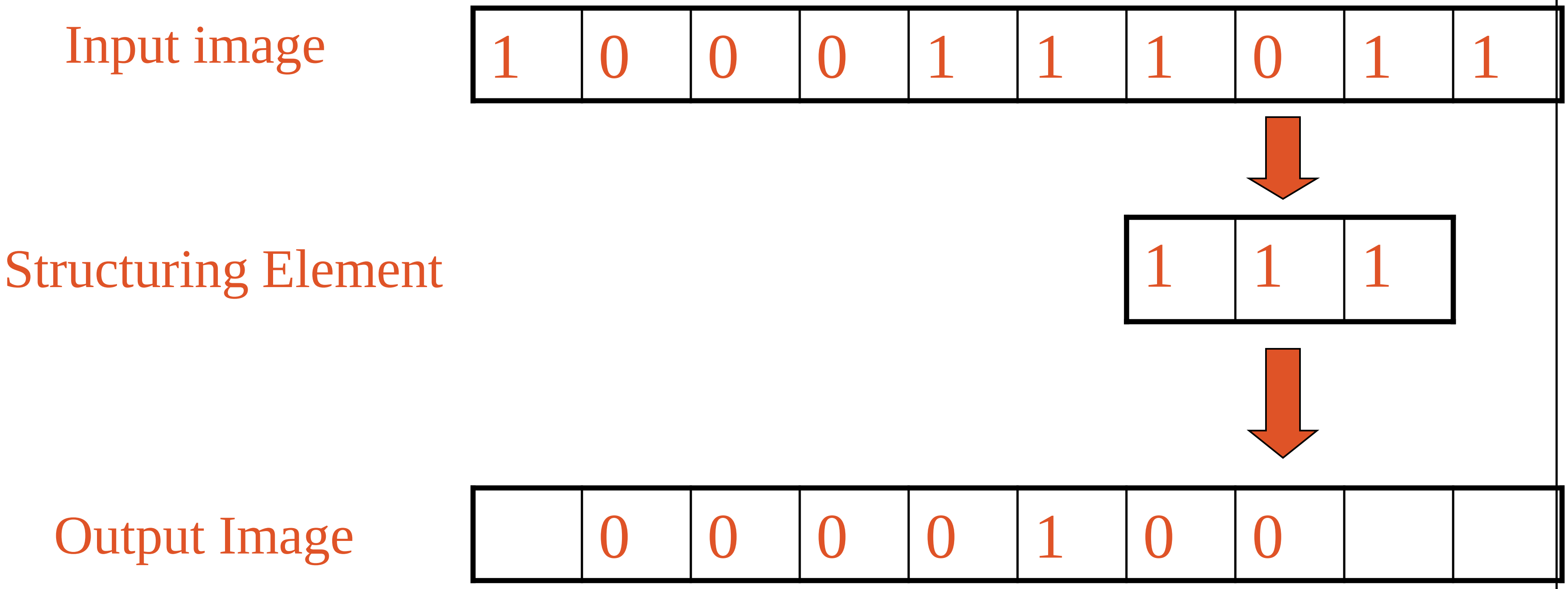
Structuring Element

1	1	1

Output Image

	0	0	0	0	1	0			

Erosion



Input image

1	0	0	0	1	1	1	0	1	1

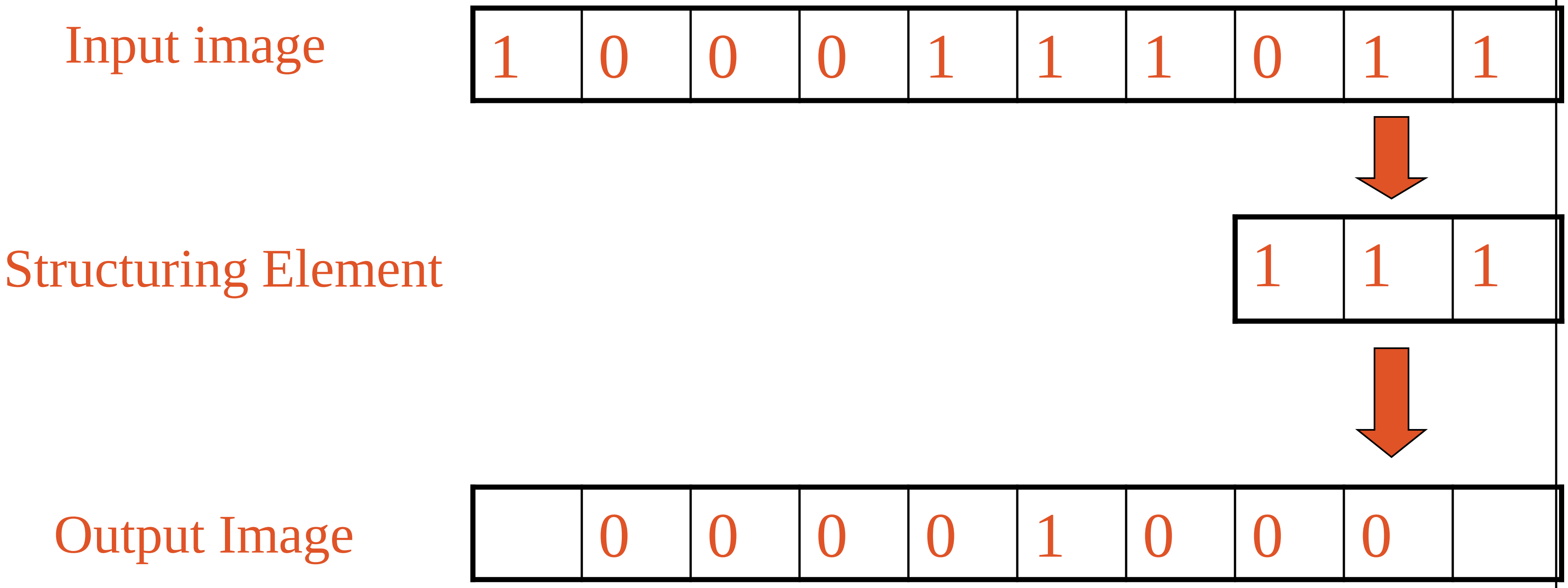
Structuring Element

1	1	1

Output Image

	0	0	0	0	1	0	0		

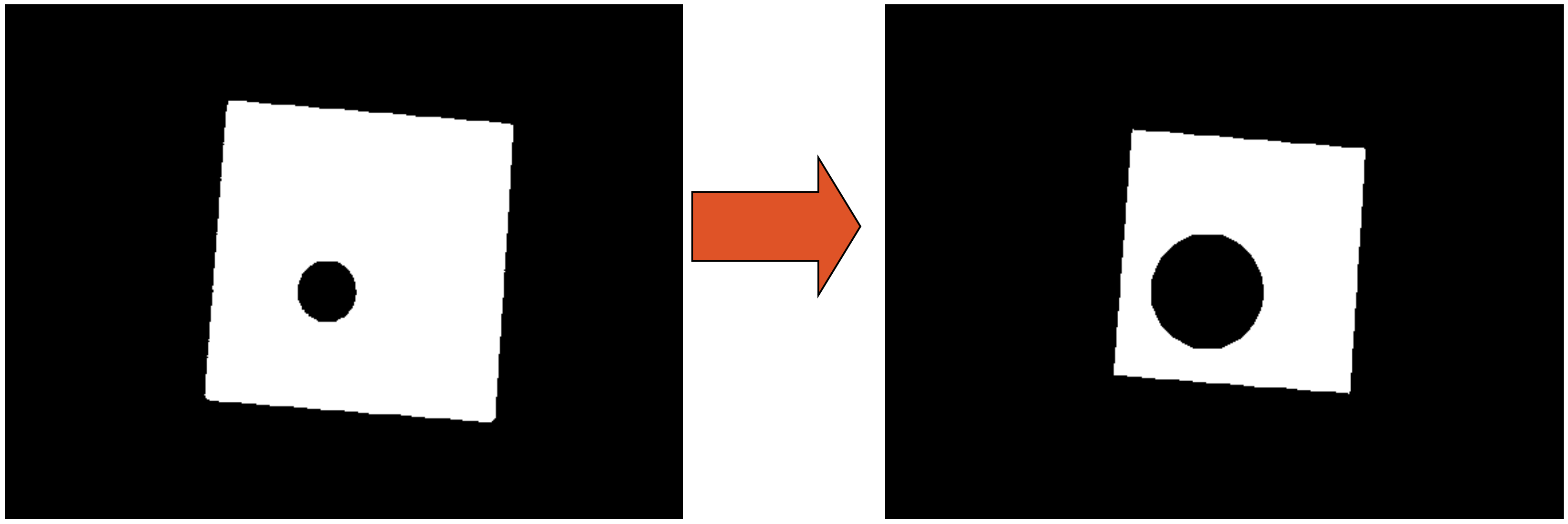
Erosion



Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element								1	1	1
Output Image		0	0	0	0	1	0	0	0	

Erosion

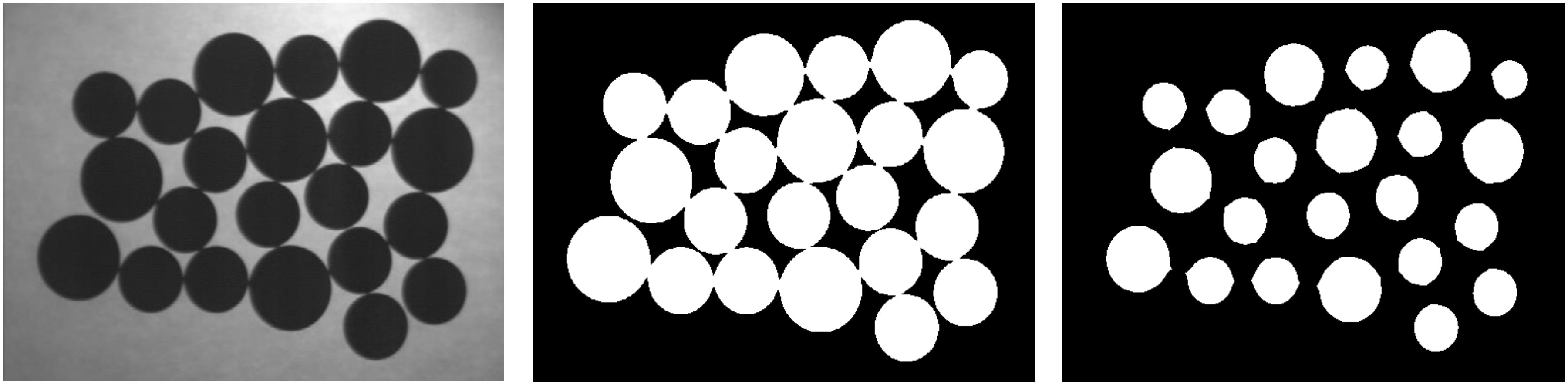
Another example of erosion



- White = 0, black = 1, dual property, image as a result of erosion gets darker

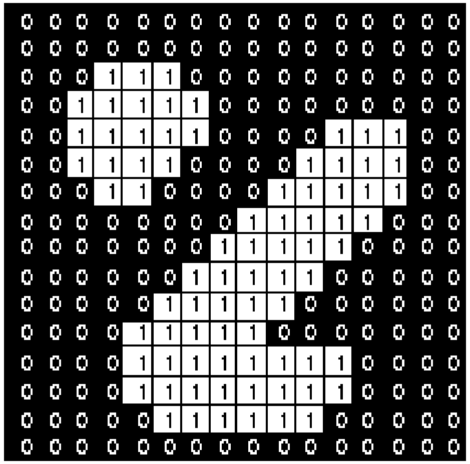
Erosion
Counting Coins Example

- Counting coins is difficult because they touch each other!
- Solution: Binarization and Erosion separates them!



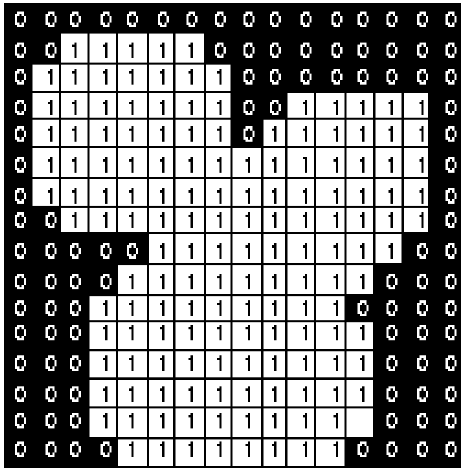
Dilation

- **Dilation** is an important morphological operation



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

→



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

- Applied **Structuring Element**:

1	1	1
1	1	1
1	1	1

Set of coordinate points =

{ (-1, -1), (0, -1), (1, -1),

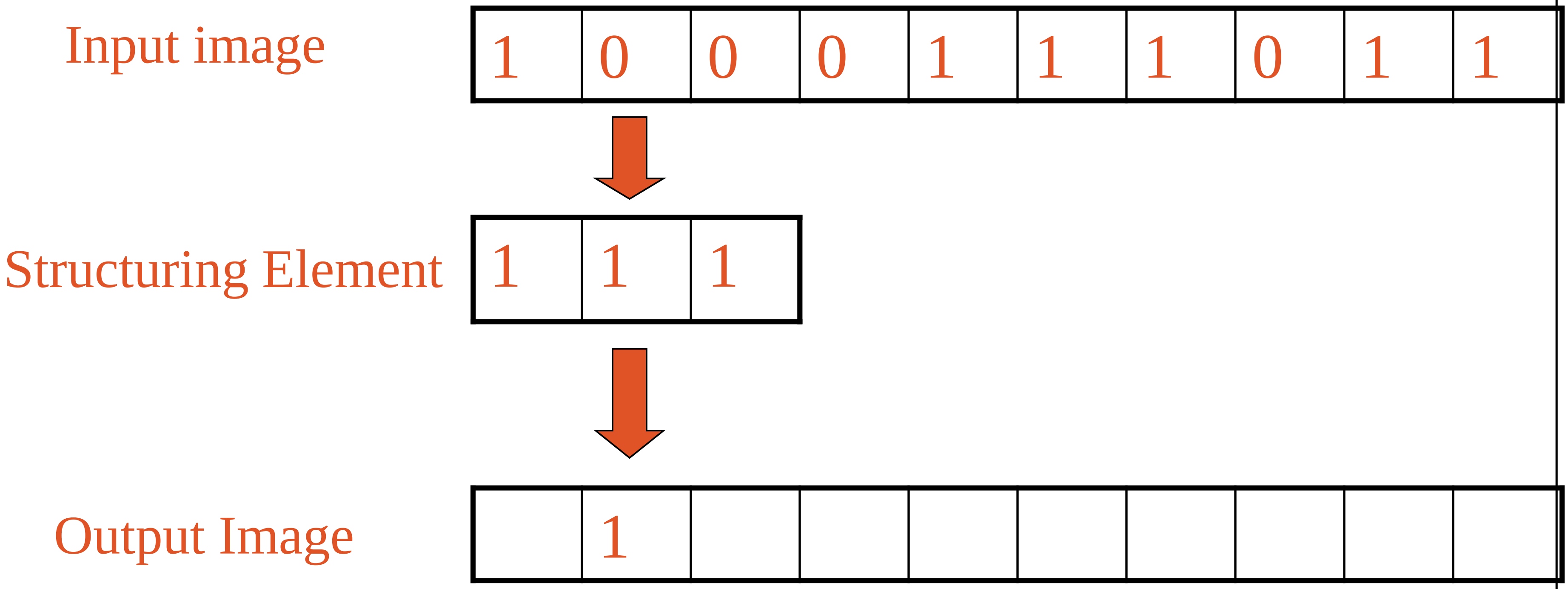
(-1, 0), (0, 0), (1, 0),

(-1, 1), (0, 1), (1, 1) }

Dilation

- **Dilation** is the set of all points in the image, where the structuring element **"touches"** the foreground.
- Consider each pixel in the input image
  - If the structuring element touches the foreground image, write a **"1"** at the origin of the structuring element!

Dilation
An Example



Input image

1	0	0	0	1	1	1	0	1	1

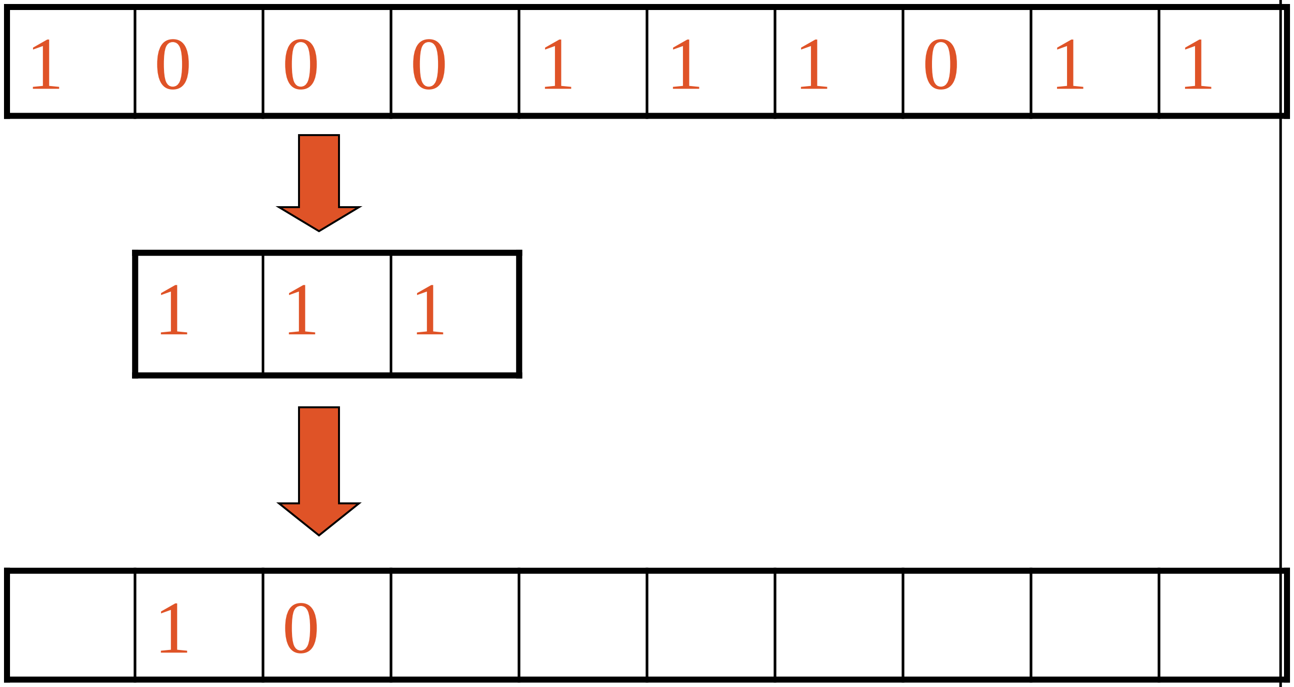
Structuring Element

1	1	1

Output Image

	1								

Dilation
An Example



Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

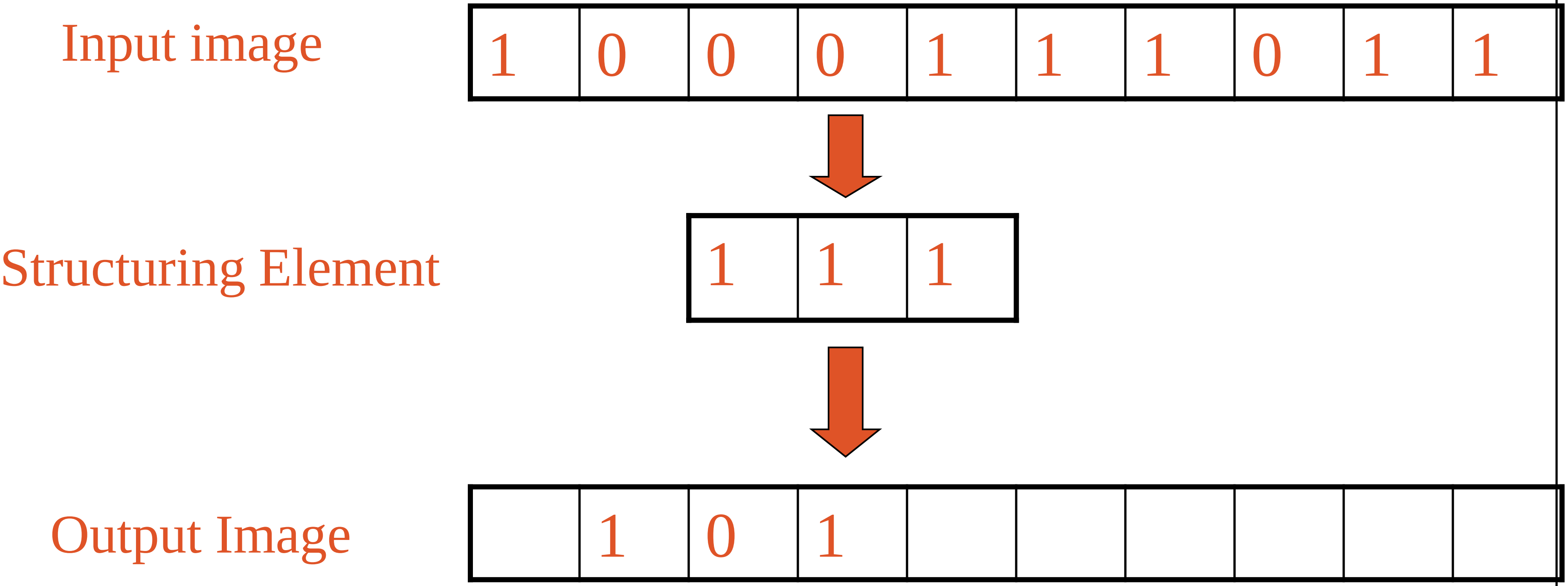
1	1	1

Output Image

	1	0							

Dilation

An Example



Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

Output Image

	1	0	1						

Dilation
An Example

Input image

1	0	0	0	1	1	1	0	1	1

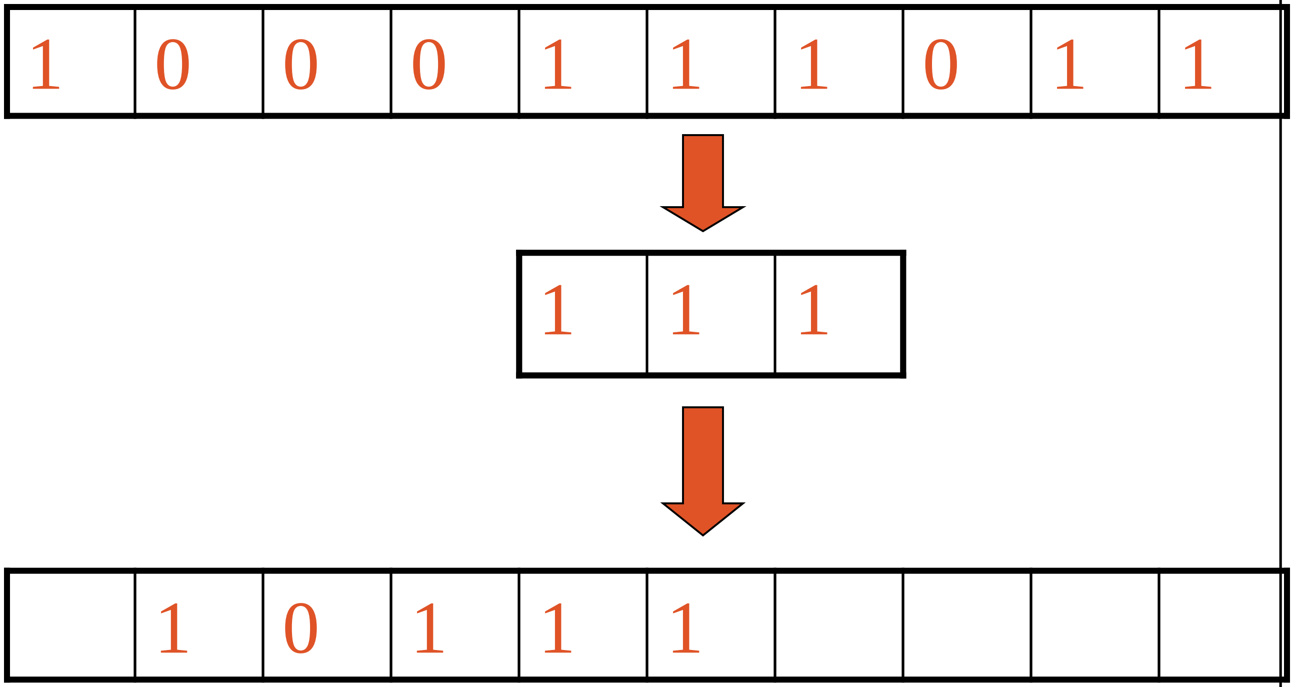
Structuring Element

1	1	1

Output Image

	1	0	1	1					

Dilation
An Example



Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

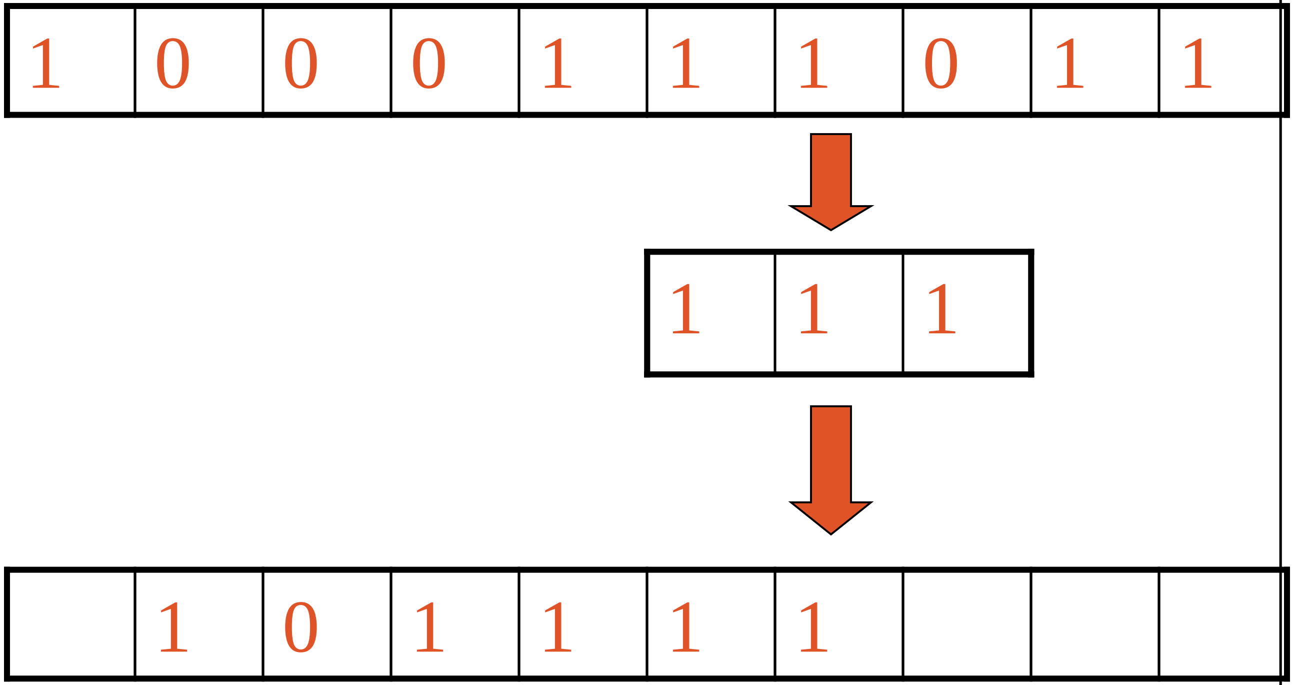
1	1	1

Output Image

	1	0	1	1	1				

Dilation

An Example



Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

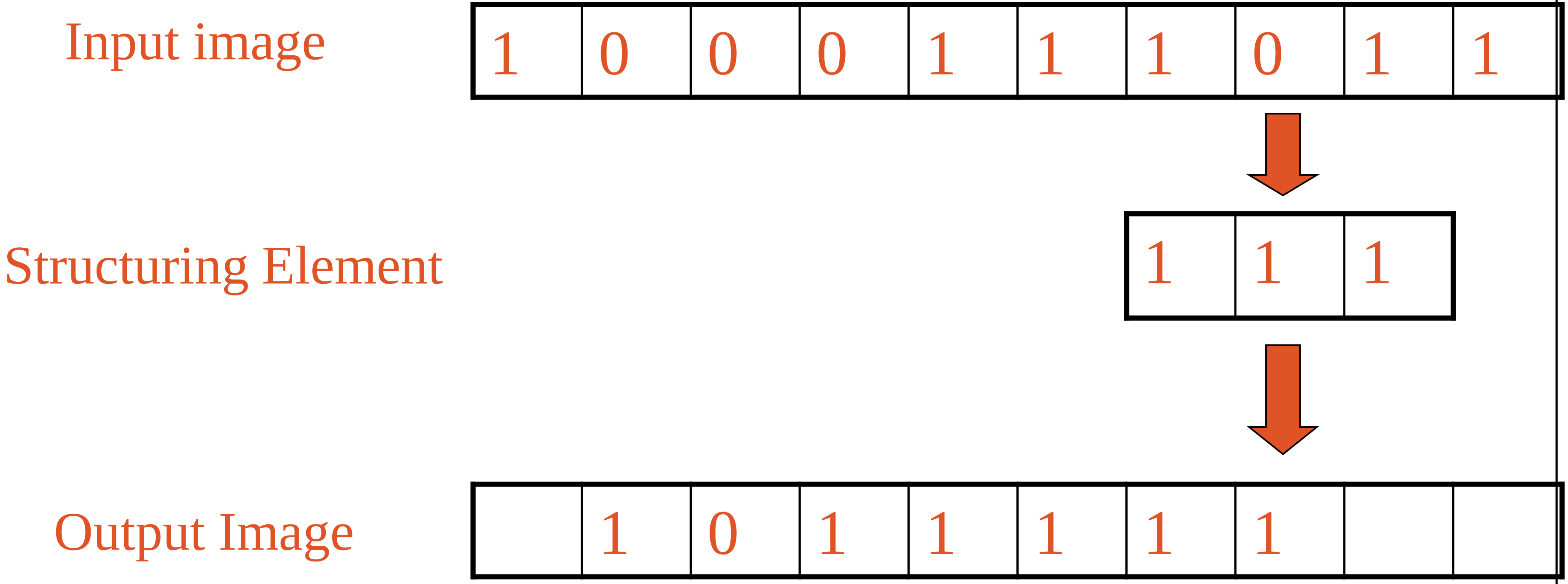
1	1	1

Output Image

	1	0	1	1	1	1			

Dilation

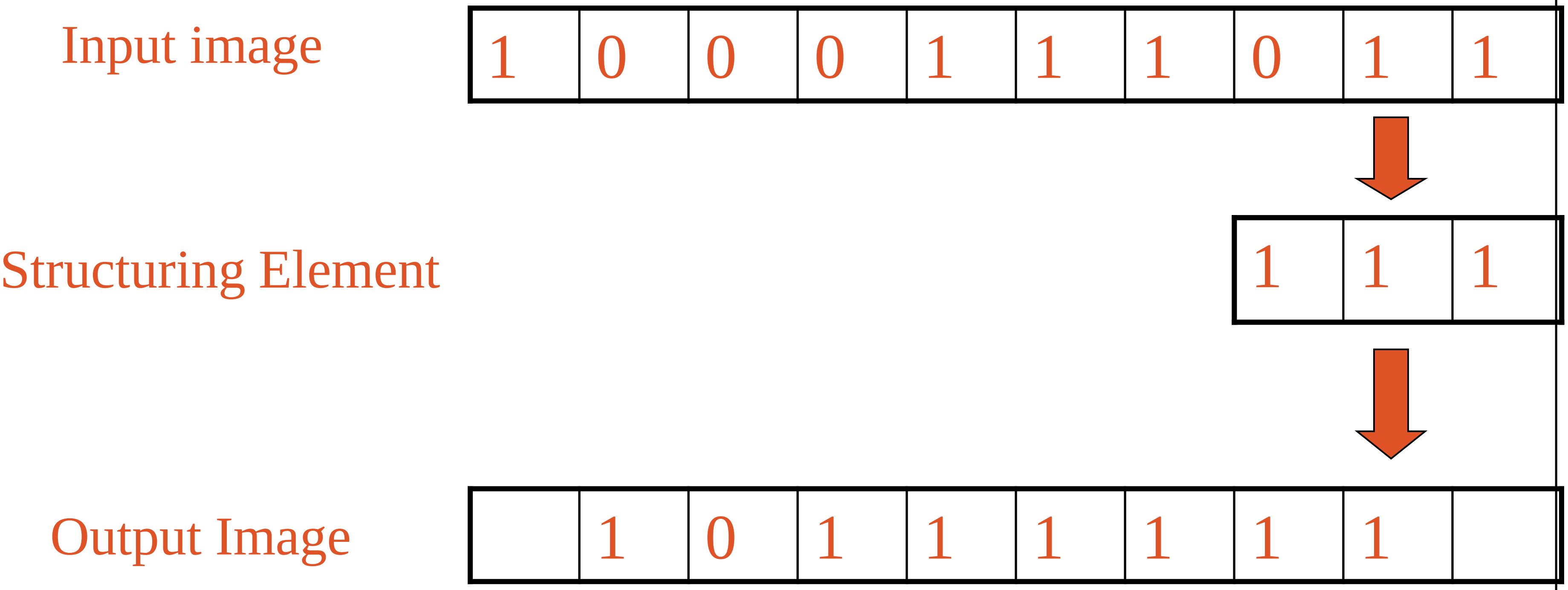
An Example



Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element							1	1	1	
Output Image		1	0	1	1	1	1	1		

Dilation

An Example



Input image

1	0	0	0	1	1	1	0	1	1

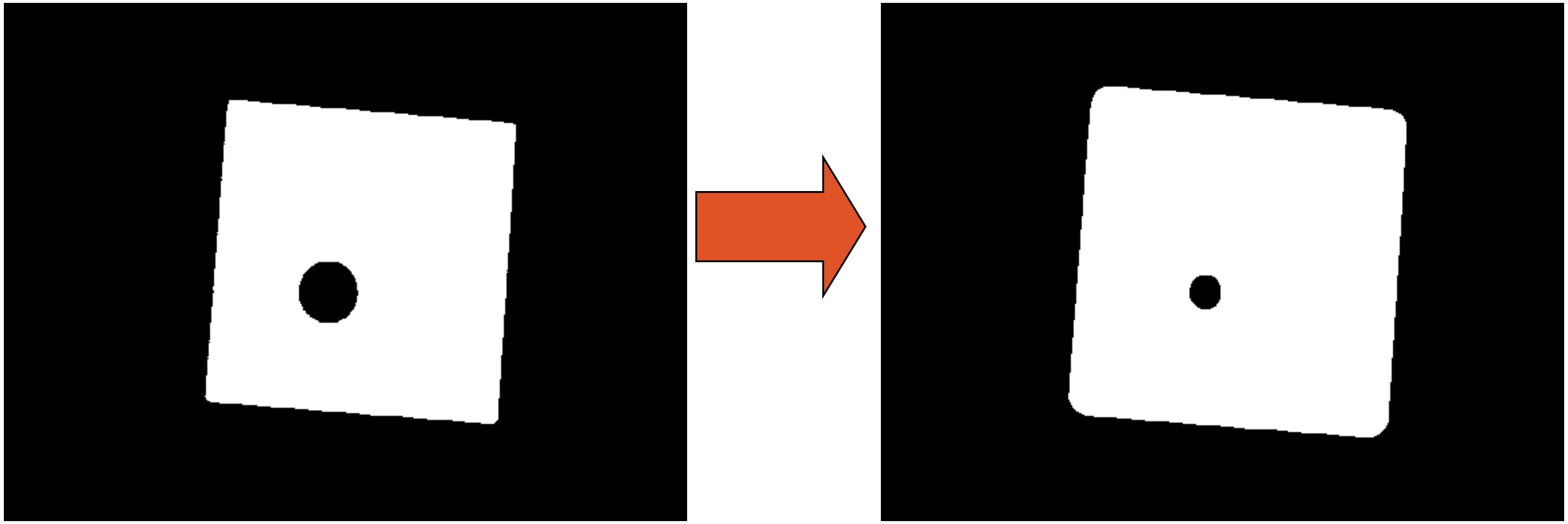
Structuring Element

1	1	1

Output Image

	1	0	1	1	1	1	1	1	

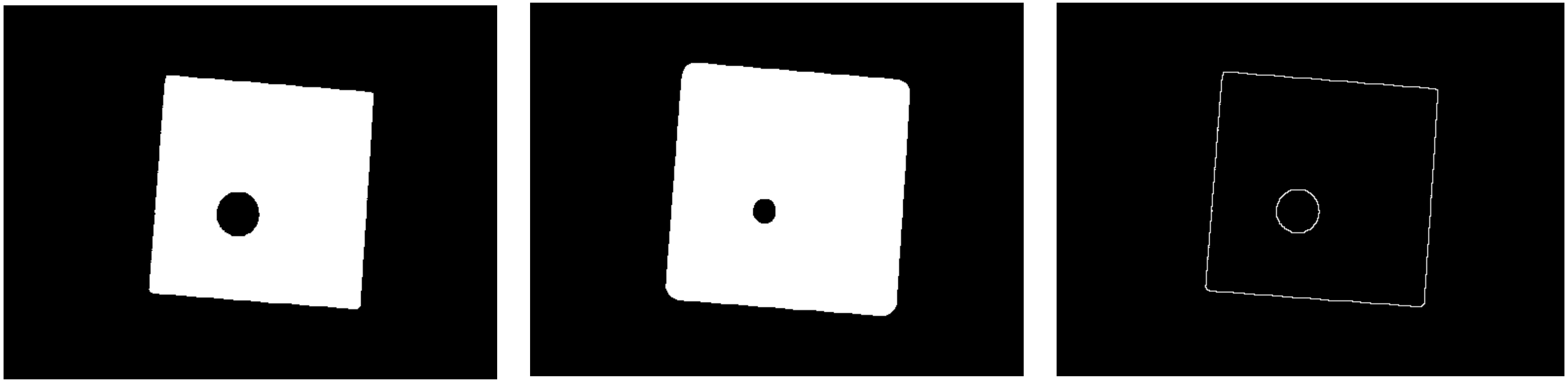
Dilation
Another Example



- Image get lighter, more uniform intensity

Edge Detection

- Edge Detection
  1. Dilate input image
  2. Subtract the dilated image from input image.
  3. Edges remain!



Dilation & Erosion

- Basic operations
- Are **dual to each other**:
  - ❑ Erosion shrinks foreground, enlarges Background
  - ❑ Dilation enlarges foreground, shrinks background

Opening & Closing

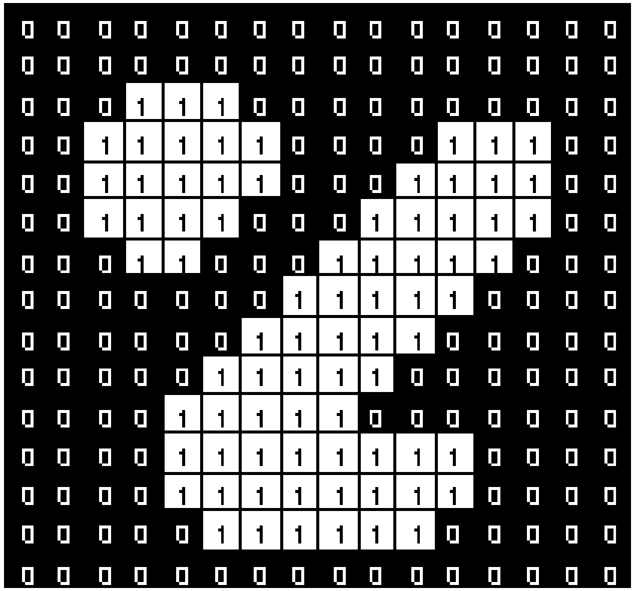
- Derived from the fundamental operations
  - ❑ **Dilatation**
  - ❑ **Erosion**
- Usually applied to **binary images**, but gray value images are also possible
- Opening and closing are **dual operations**

Opening

- **Similar to Erosion**
  - Spot and noise removal
  - Less destructive
- Opening is defined as a **Erosion followed by dilation** *using the same structuring element for both operations*.

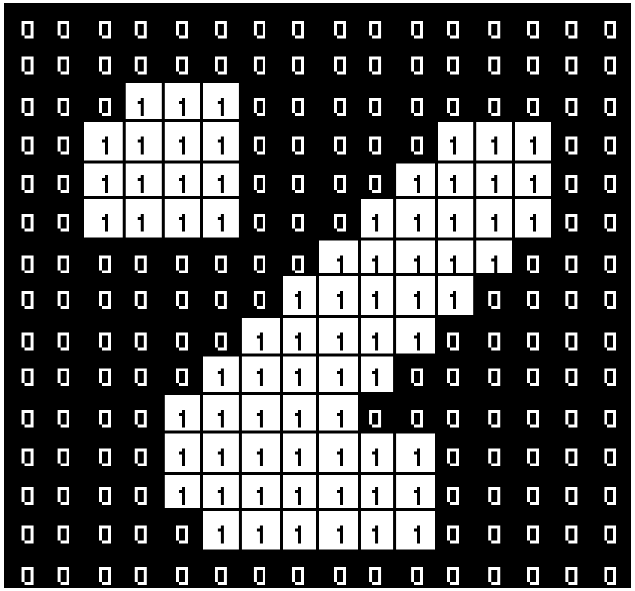
Opening

- Structuring element: 3x3 square



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

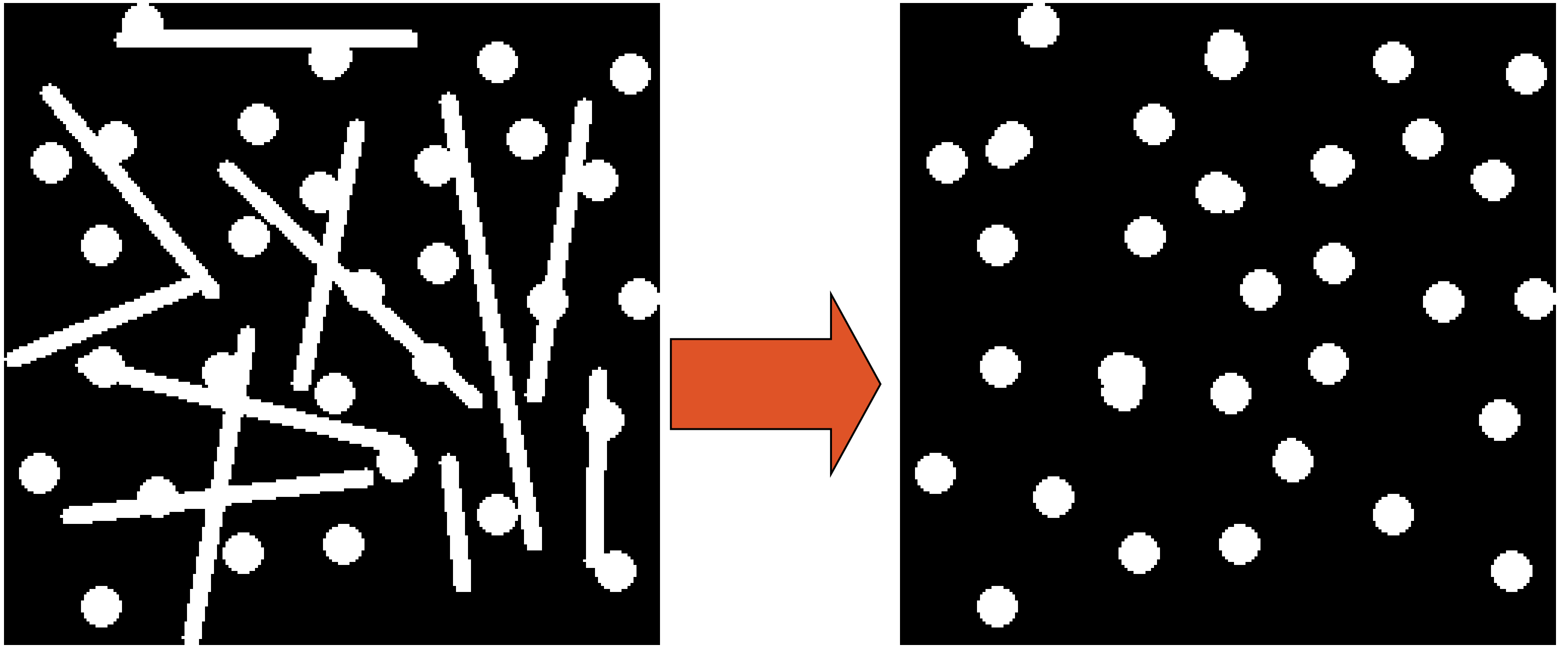
→



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

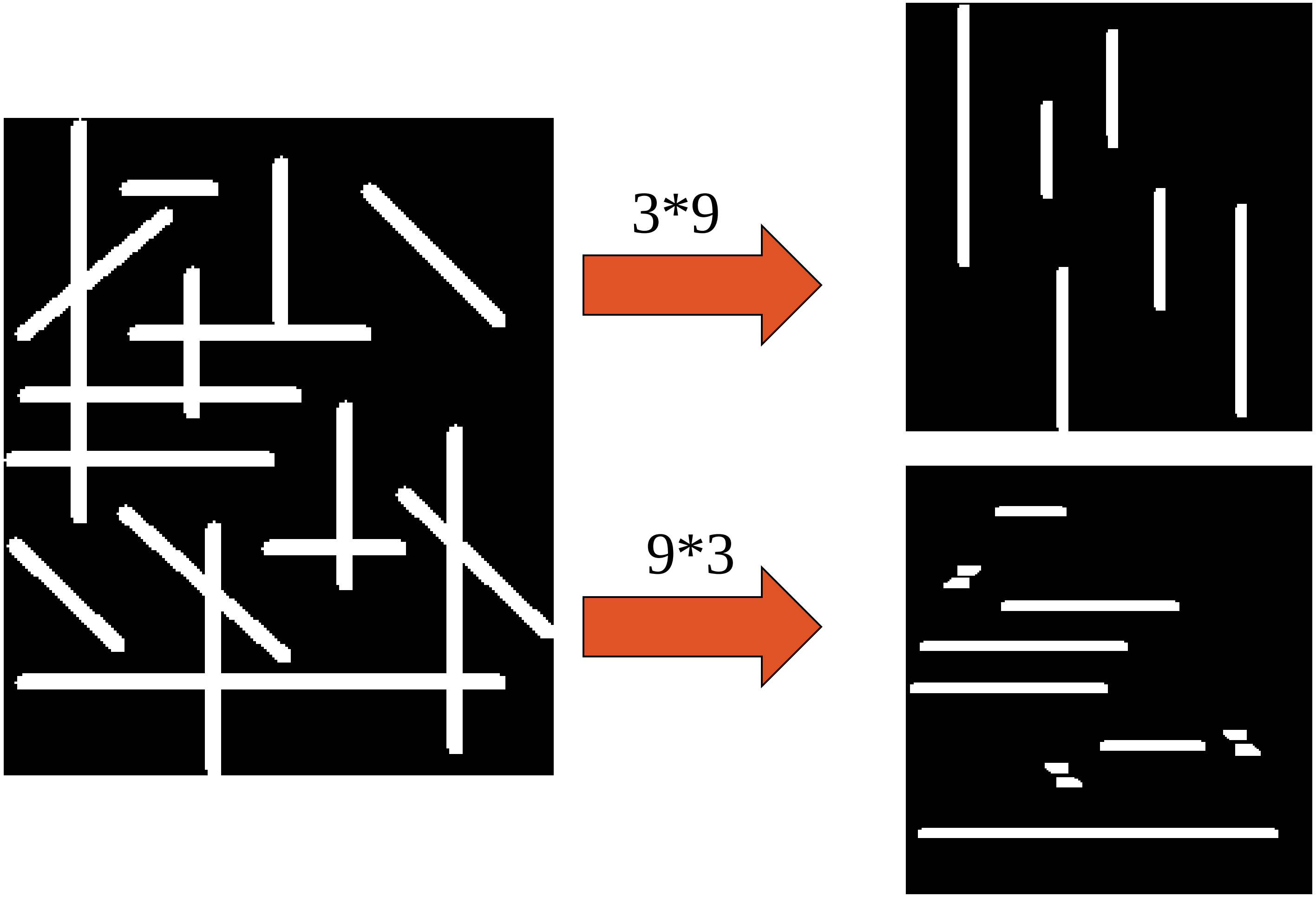
Opening
An Example

- Opening with a 11 pixel diameter disc



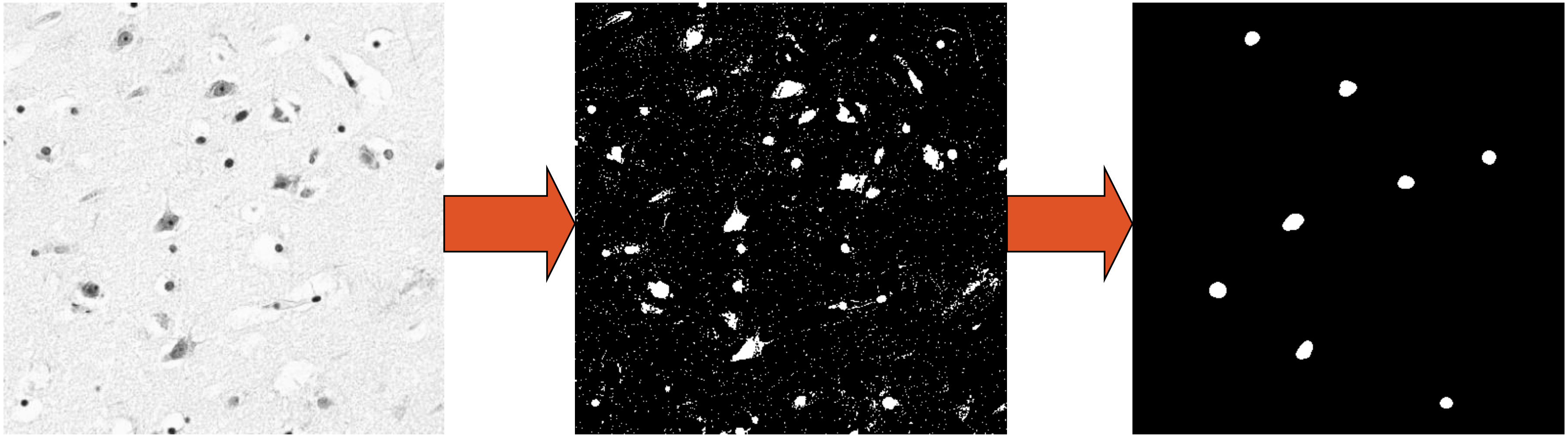
Opening
An Example

- 3x9 and 9x3 Structuring Element



Opening
Use Opening for Separating Blobs

- Use large structuring element that fits into the big blobs
- Structuring Element: 11 pixel disc

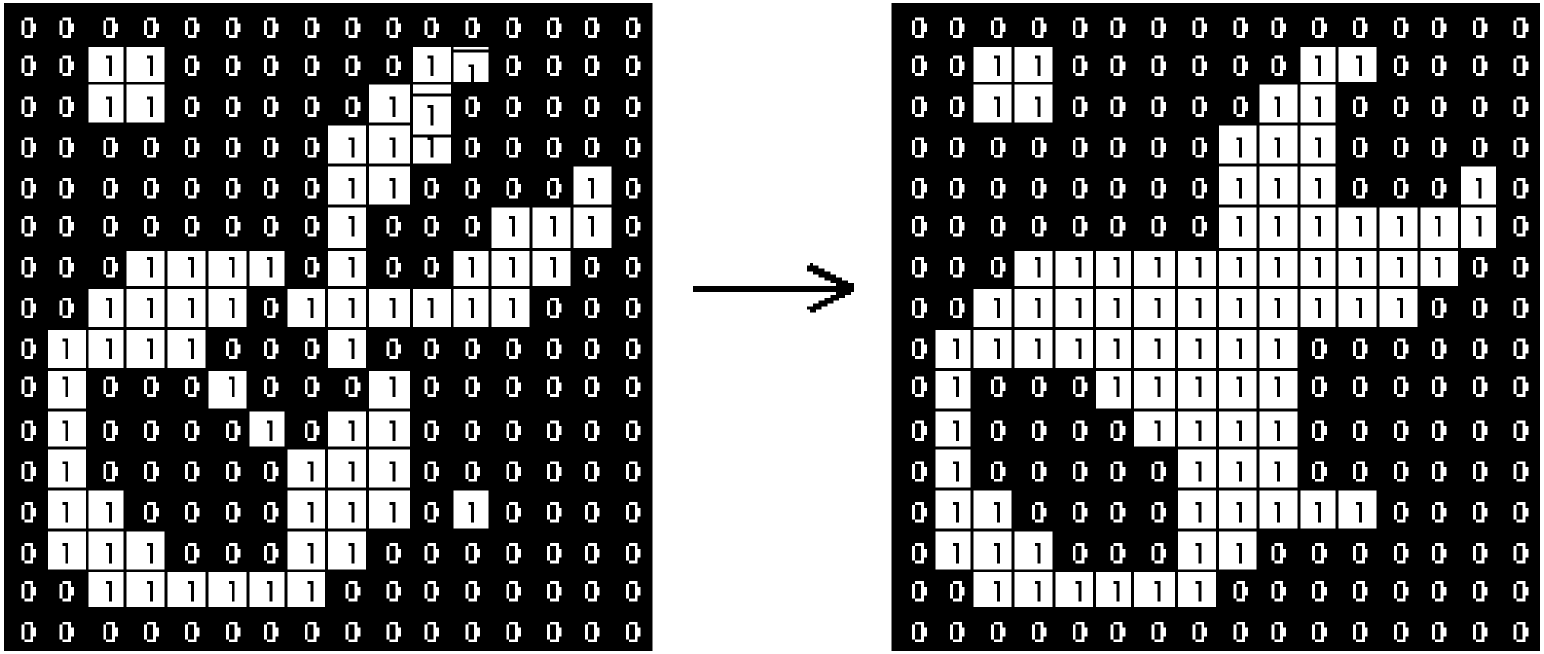


Closing

- **Similar to Dilation**
  - Removal of holes
  - Tends to enlarge regions, shrink background
- Closing is defined as a **Dilatation followed by an Erosion** using *the same structuring element for both operations*.

Closing

- Structuring element: 3x3 square

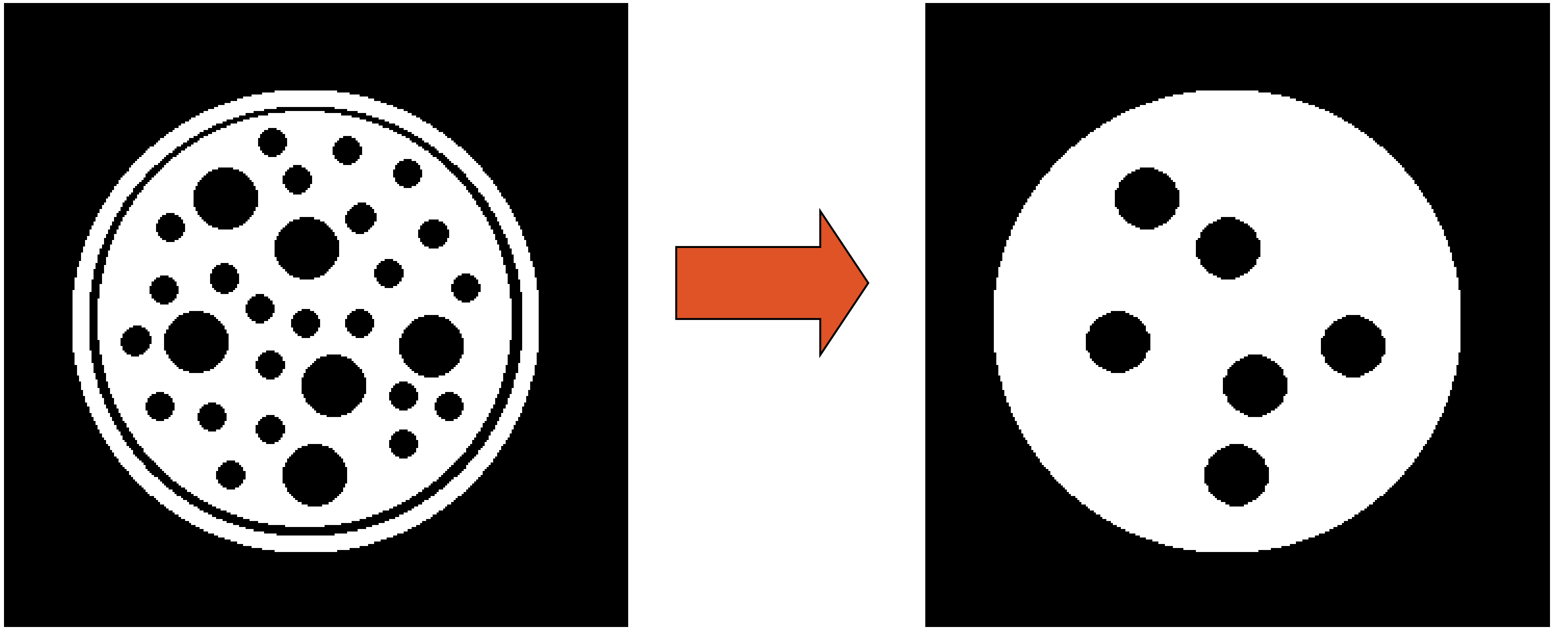


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0
0	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0
0	0	1	1	1	1	0	1	1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0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

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

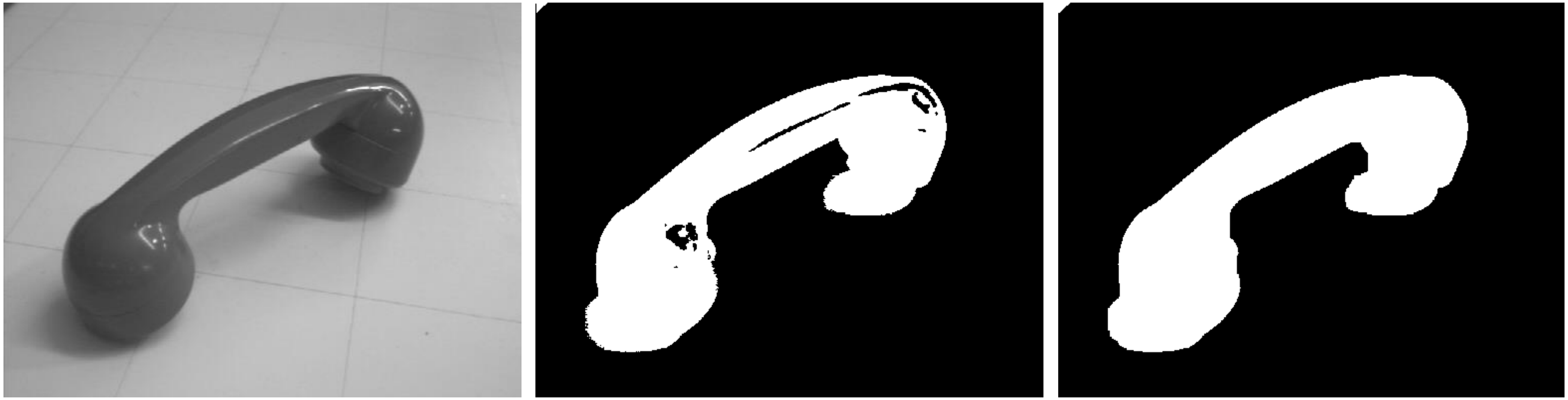
Closing

- Closing operation with a 22 pixel disc
- Closes small holes in the foreground



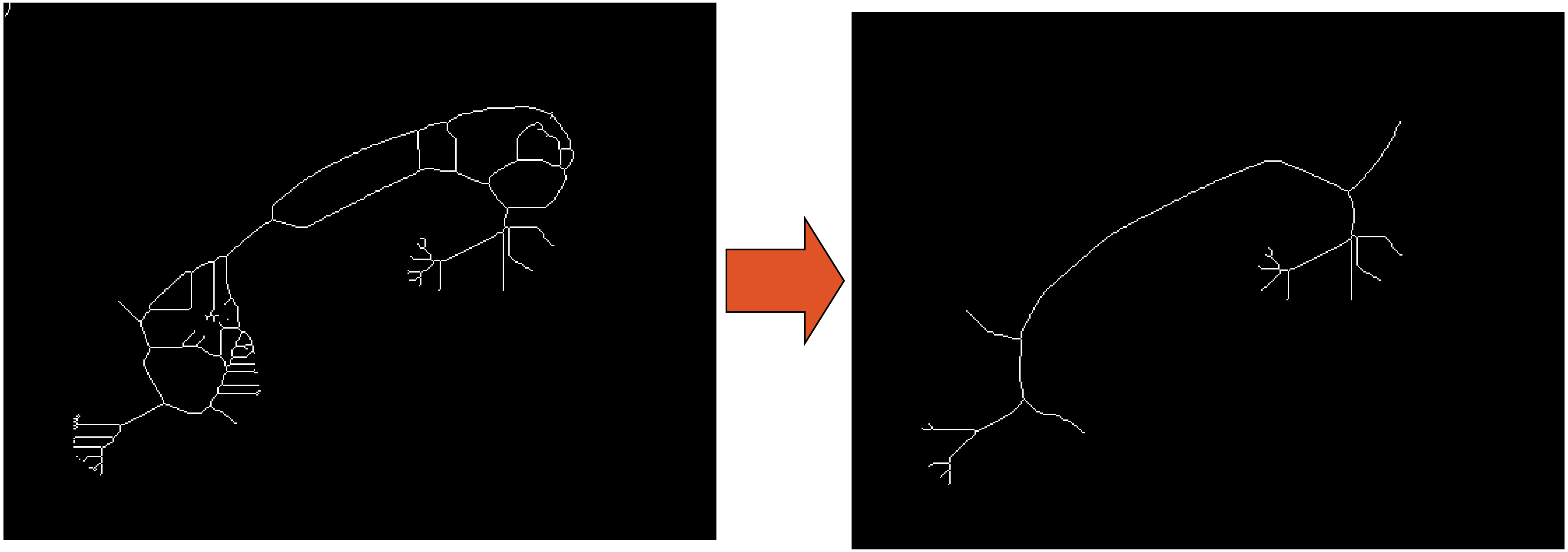
Closing
An Example

Closing with disc of size 20



Closing
An Example

- Good for further processing: E.g. Skeleton operation looks better for closed image!



Opening & Closing

- **Opening is the *dual* of closing**
- *i.e.* opening the foreground pixels with a particular structuring element
- is equivalent to closing the background pixels with the same element.

Hit-and-miss Transform

- Used to look for particular patterns of foreground and background pixels
- Very simple object recognition
- All other morphological operations can be derived from it.

Hit-and-miss Transform

- Example for a Hit-and-miss Structuring Element
- Contains **0’s, 1’s and don’t care’s**.
- Usually a "**1**" at the origin!

	1	
0	1	1
0	0	

Hit-and-miss Transform

- Similar to Pattern Matching:
- If foreground and background pixels in the structuring element *exactly match* foreground and background pixels in the image, then the pixel under the origin of the structuring element is set to the foreground color.

Corner Detection with Hit-and-miss Transform

- Structuring Elements representing four corners

	1	
0	1	1
0	0	

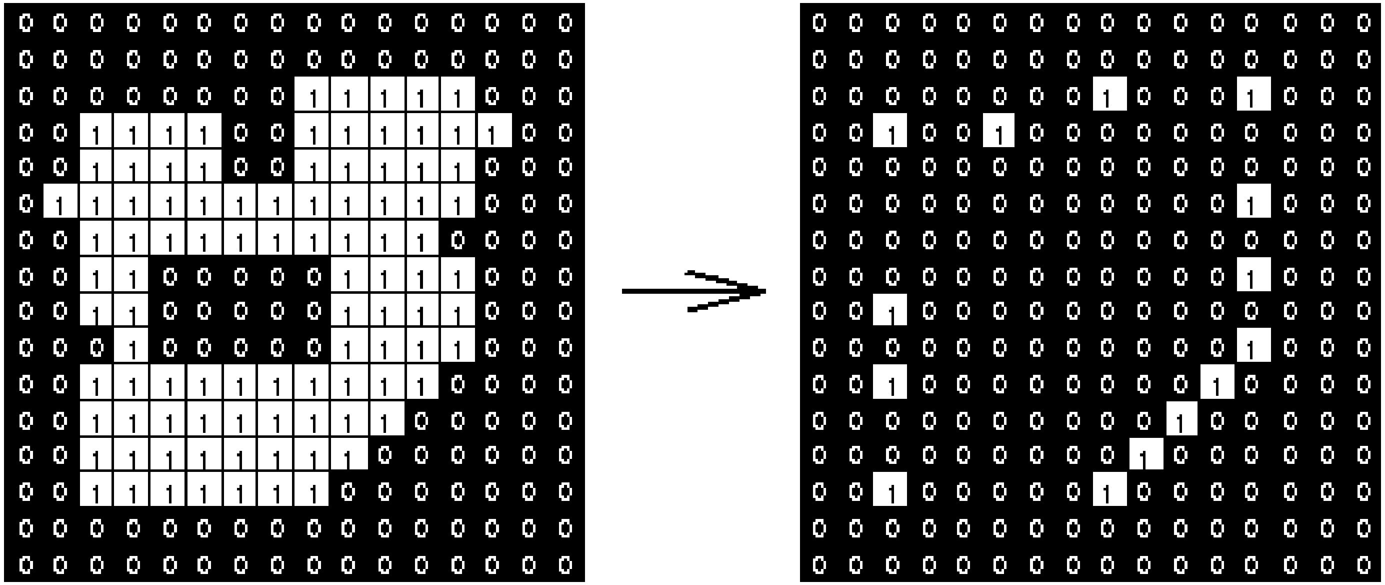
	1	
1	1	0
	0	0

	0	0
1	1	0
	1	

0	0	
0	1	1
	1	

Corner Detection with Hit-and-miss Transform

- Apply each Structuring Element
- Use OR operation to combine the four results



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

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

Thinning

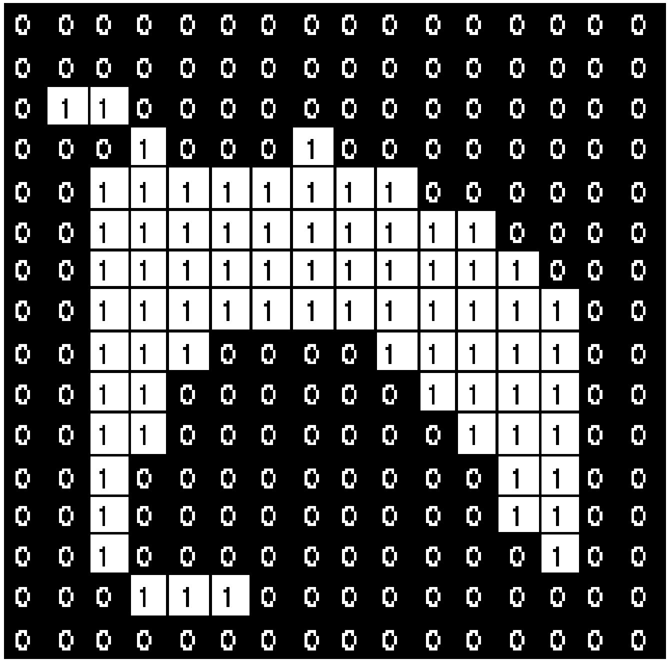
- Used to remove selected foreground pixels from binary images
- After edge detection, lines are often thicker than one pixel.
- Thinning can be used to thin those line to one pixel width.

Thinning

- If foreground and background fit the structuring element exactly, then the pixel at the origin of the SE is set to 0
- Note that the value of the SE at the origin is 1 or *don't care*!

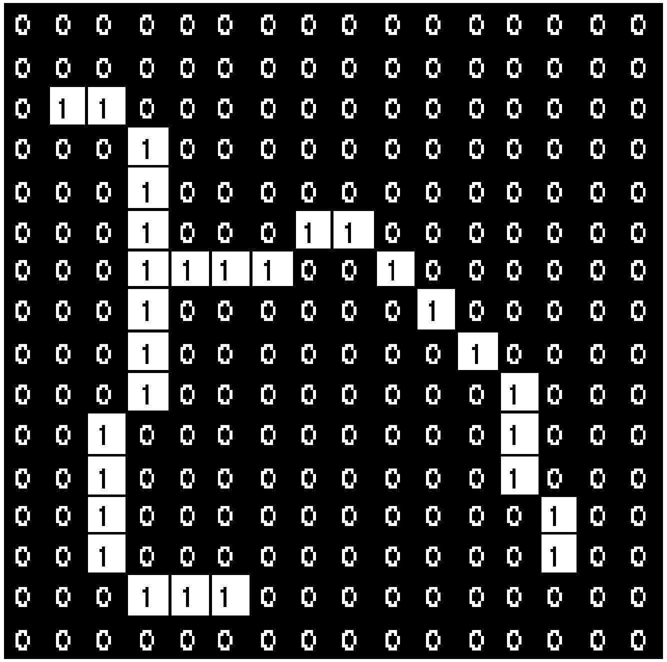
Thinning
An Example

We use two Hit-and-miss Transforms



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

→



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

Thickening

- Used to grow selected regions of foreground pixels
- If foreground and background match exactly the SE, then set the pixel at its origin to 1!
- Note that the value of the SE at the origin is 0 or *don't care*!

Thickening

An Example

1	1	
1	0	
1		0

	1	1
	0	1
0		1