# BINARY IMAGE

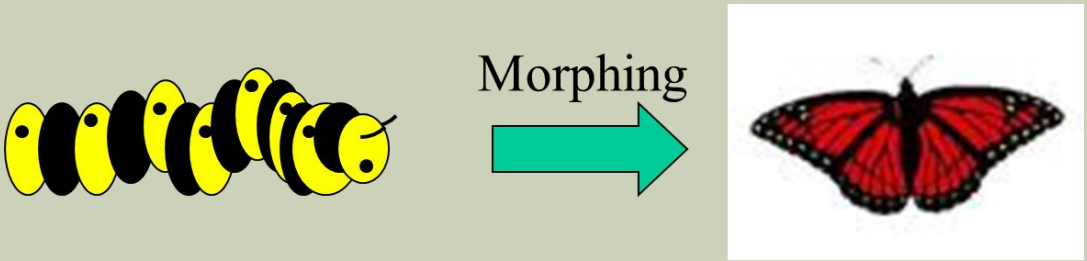
# MORPHOLOGICAL FILTERING

# MORPHOLOGICAL IMAGE PROCESSING

## Introduction

- The word ***morphology*** commonly denotes a branch of biology that deals with the **form and structure** of animals and plants.

- "morphing" in Biology which means "**changing a shape**".

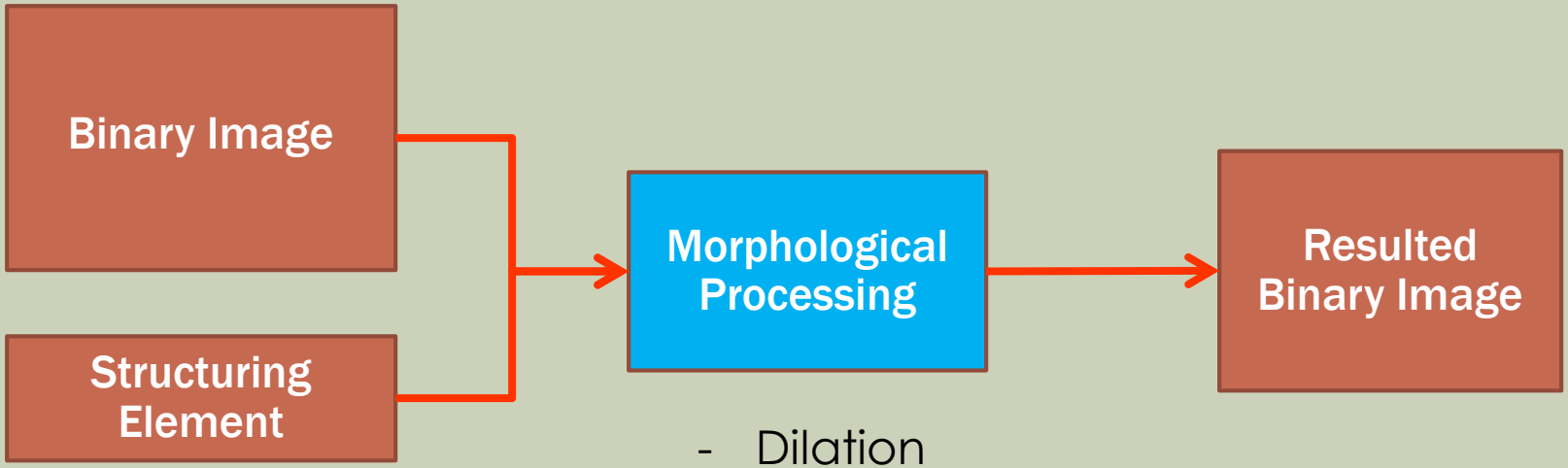



- Therefore, morphological operations are intended to affect the **shape of the object**

# BINARY MORPHOLOGICAL FILTERING

- pre-processing used to prepare binary (thresholded) images for object segmentation/recognition
  - Noise filtering, Shape simplification, …
- extracting image components that are useful in the representation and description of region shape, such as
  - boundaries extraction
  - skeletons
  - thinning
  - pruning

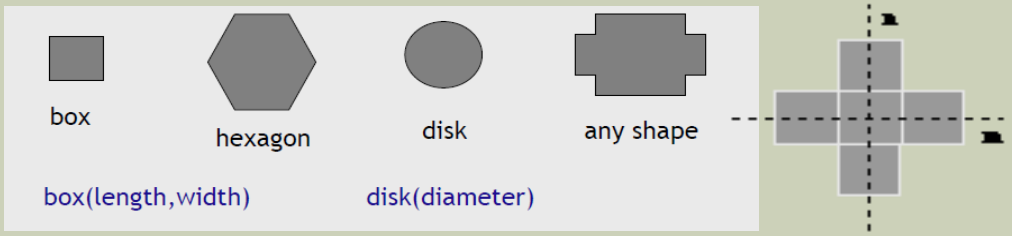
# BASIC ELEMENTS




A shape mask with any shapes or sizes

- Square (Box)
- Circle (Disk)
- Cross
- Hexagon
- Any shape

- Dilation
- Erosion
- Opening
- Closing
- Hit-Miss
- Thinning

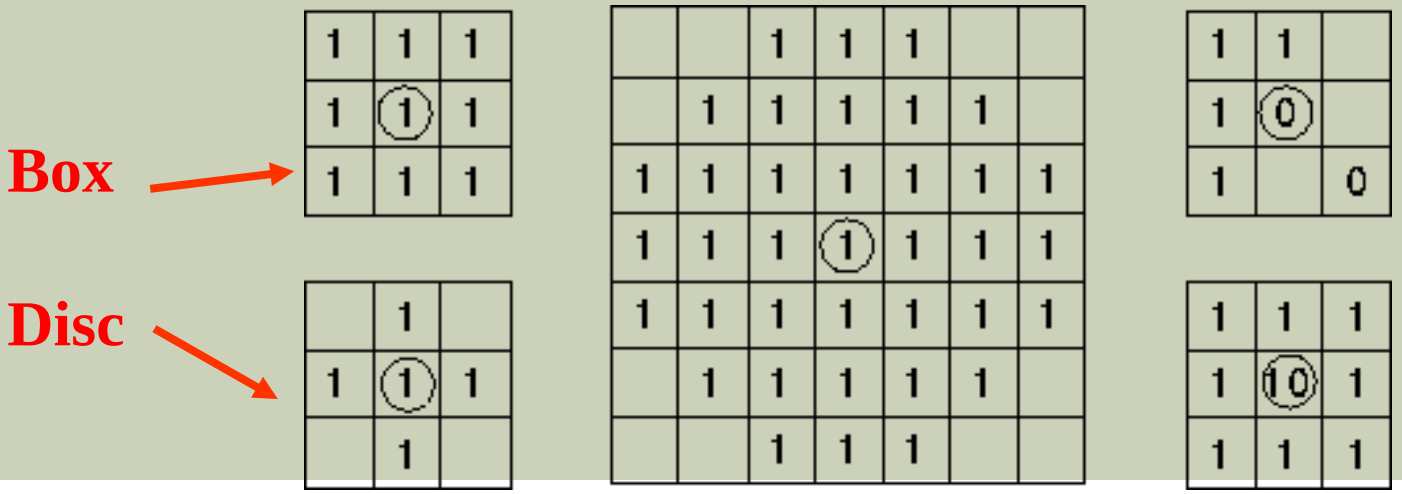


# BINARY IMAGE

- white pixels
  - normally represent foreground regions
  - Binary representation: ‘1’
- black pixels
  - denote background
  - Binary representation: ‘0’

# STRUCTURING ELEMENT (KERNEL)

- have varying shapes and sizes
- have an origin
- Usually, element values are 0,1 and
  - Other values are also permited
  - Empty spots (none(!)) are *don't care*'s!



# MORPHOLOGICAL OPERATIONS

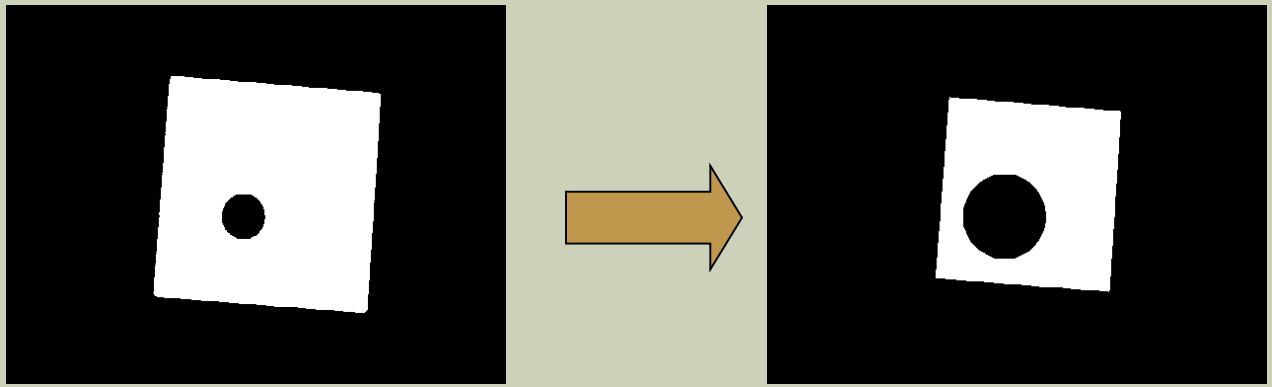
- Basic Operations
  - Erosion
    - shrinks foreground, enlarges Background


  - Dilation
    - enlarges foreground, shrinks background
    - filling holes and gaps

# EROSION OPERATION

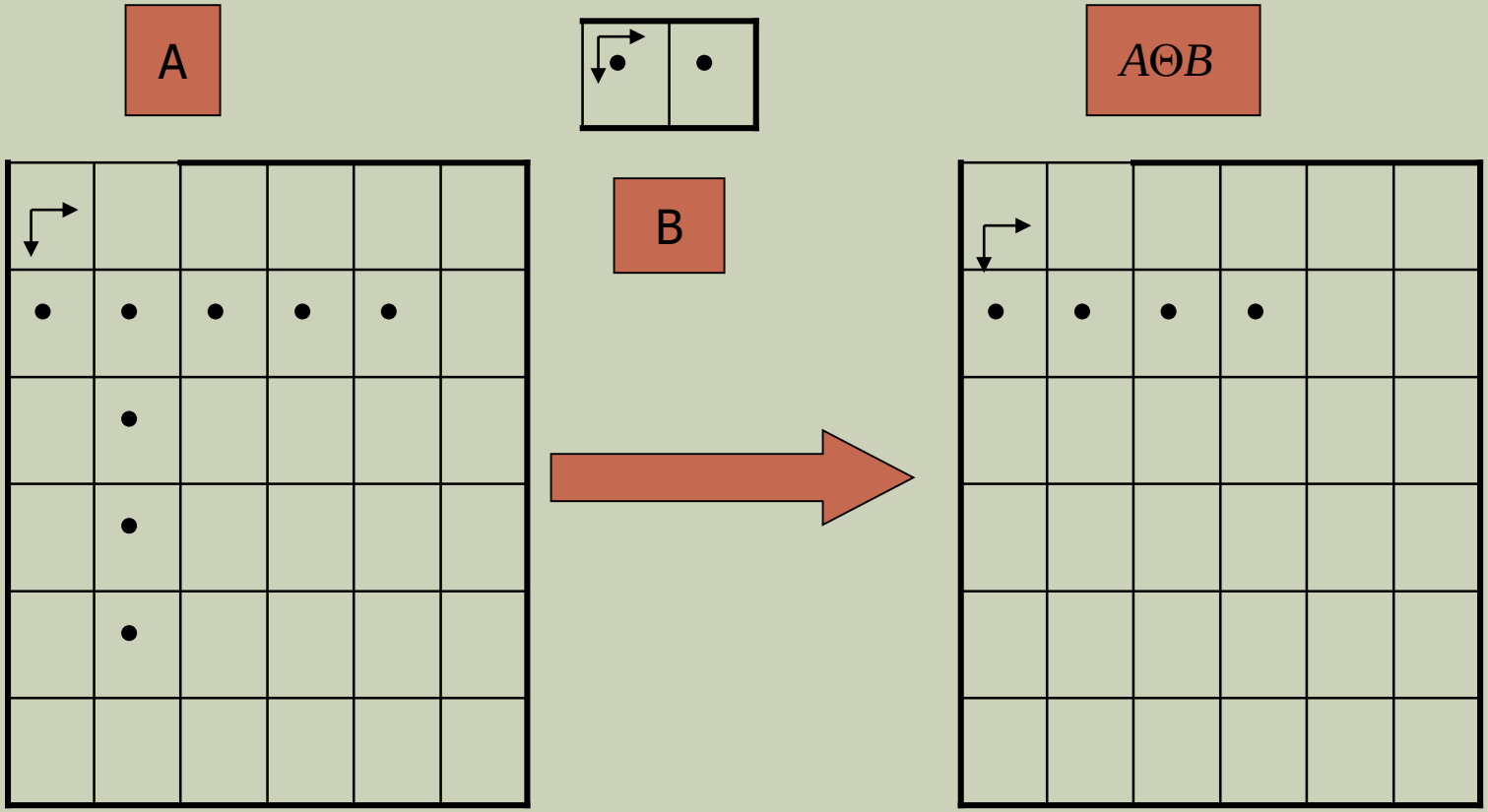
# EROSION OPERATION

- consider each of the *foreground* pixels
- superimpose the structuring element on top
- Consider each foreground pixel in the input image
  - If **all** the structuring element **fits in** foreground pixels,
    - write a "1" at the origin of the structuring element!



Pixels beyond the boundary of the image are assigned to maximum value -> '1'

# EXAMPLES OF EROSION OPERATION



# EXAMPLES OF EROSION OPERATION

origin

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

B

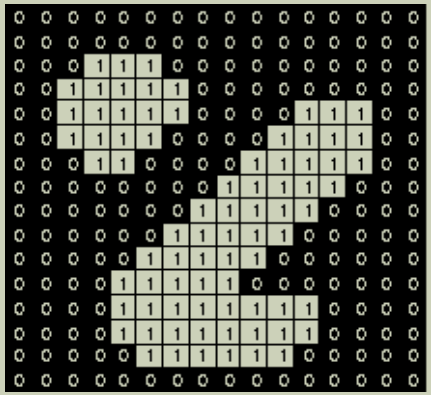
| |
|---|
| 1 |
| 1 |
| 1 |

S

erode

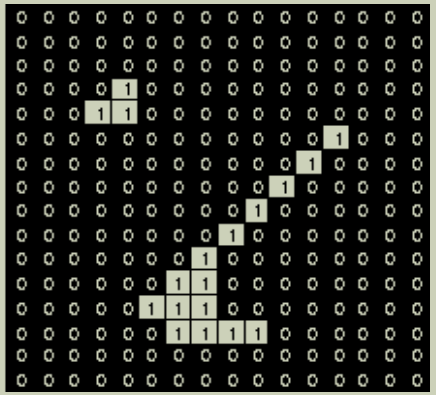
| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

$B \ominus S$

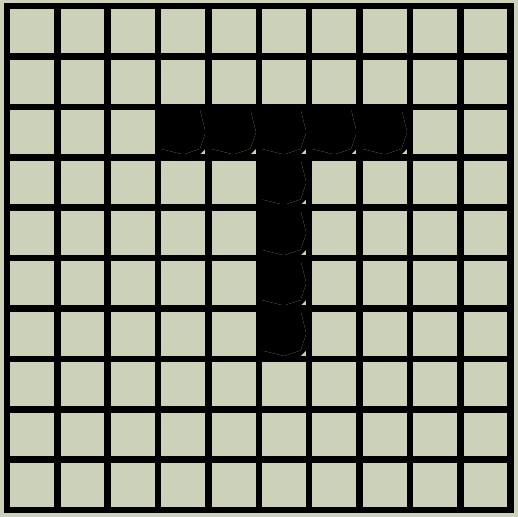


| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

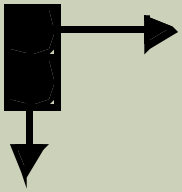
Structuring element



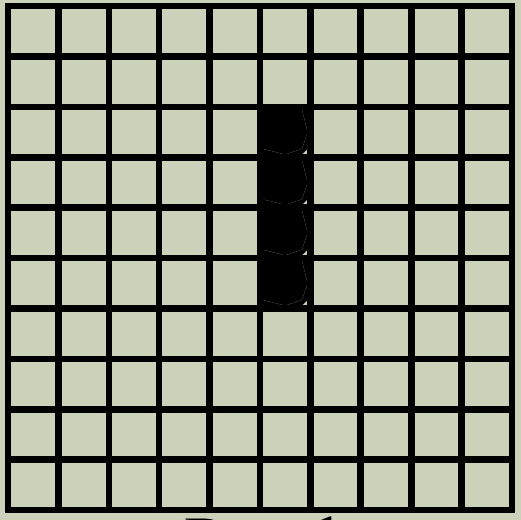
# STRUCTURING ELEMENT IN EROSION EXAMPLE



Image

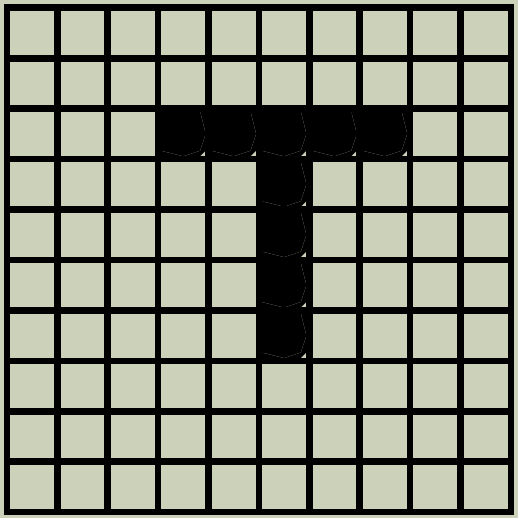


Structuring Element

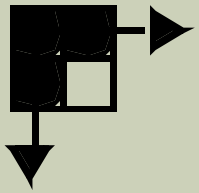


Result

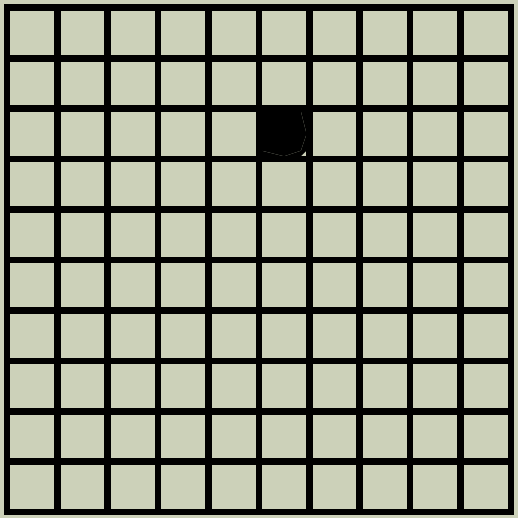
# STRUCTURING ELEMENT IN EROSION EXAMPLE



Image

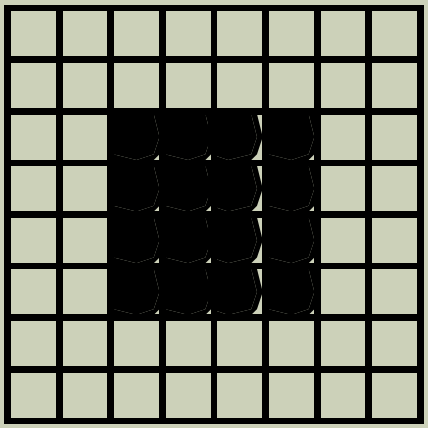


Structuring Element

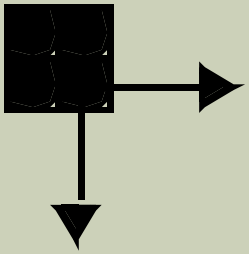


Result

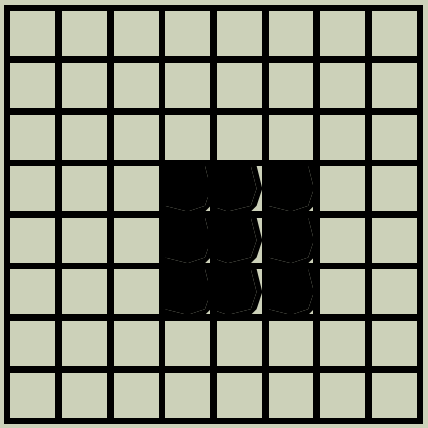
# STRUCTURING ELEMENT IN EROSION EXAMPLE



Image

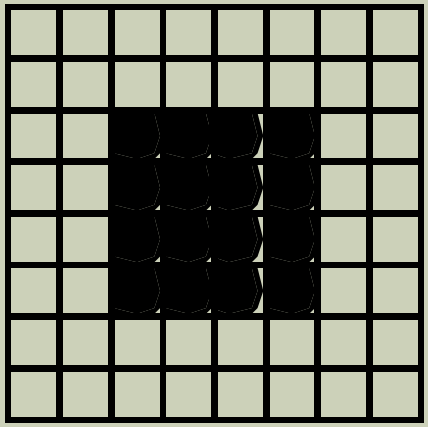


Structuring Element

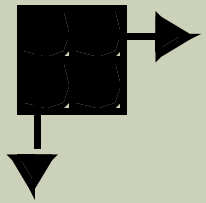


Result

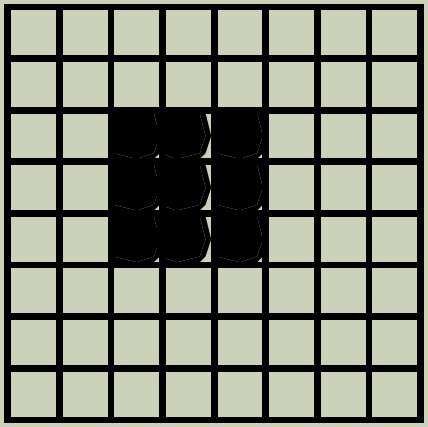
# STRUCTURING ELEMENT IN EROSION EXAMPLE



Image

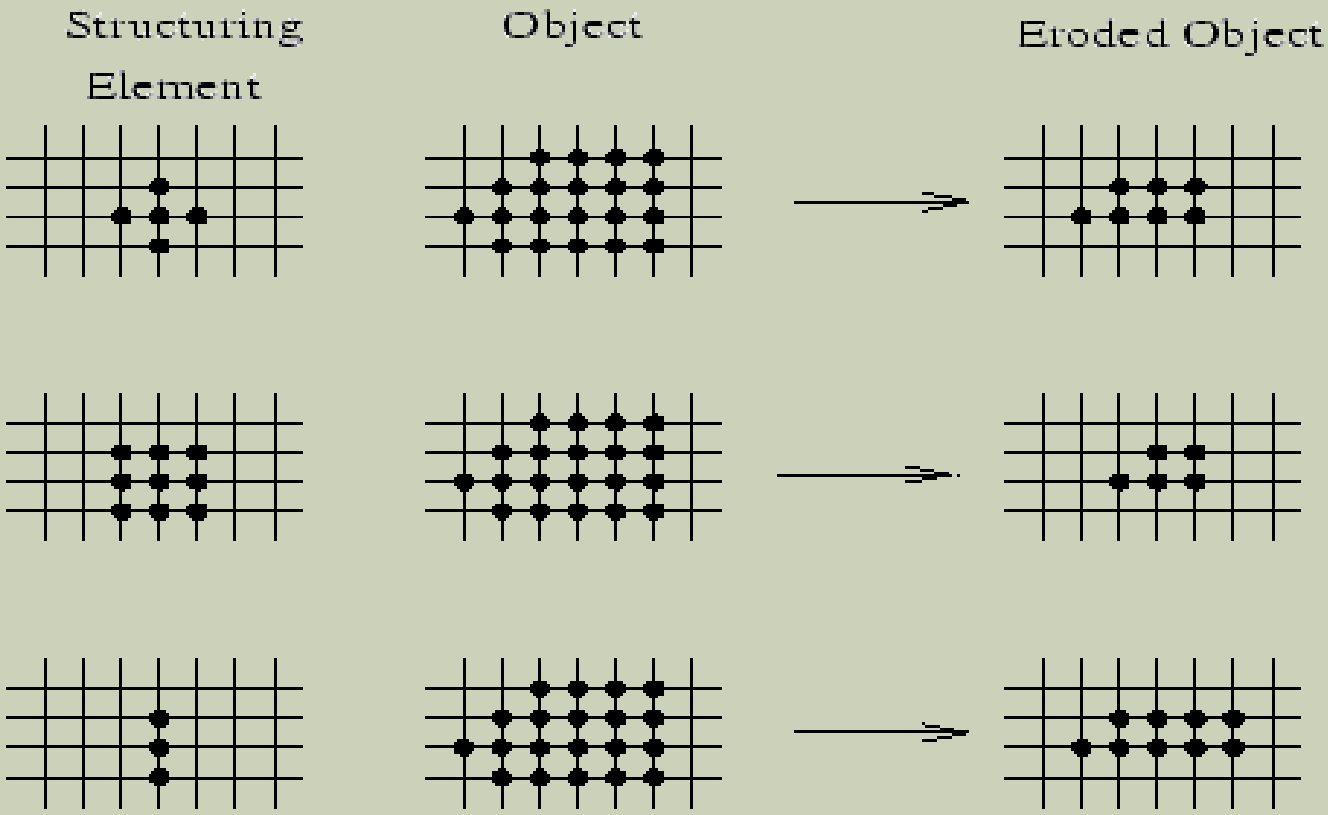


Structuring Element

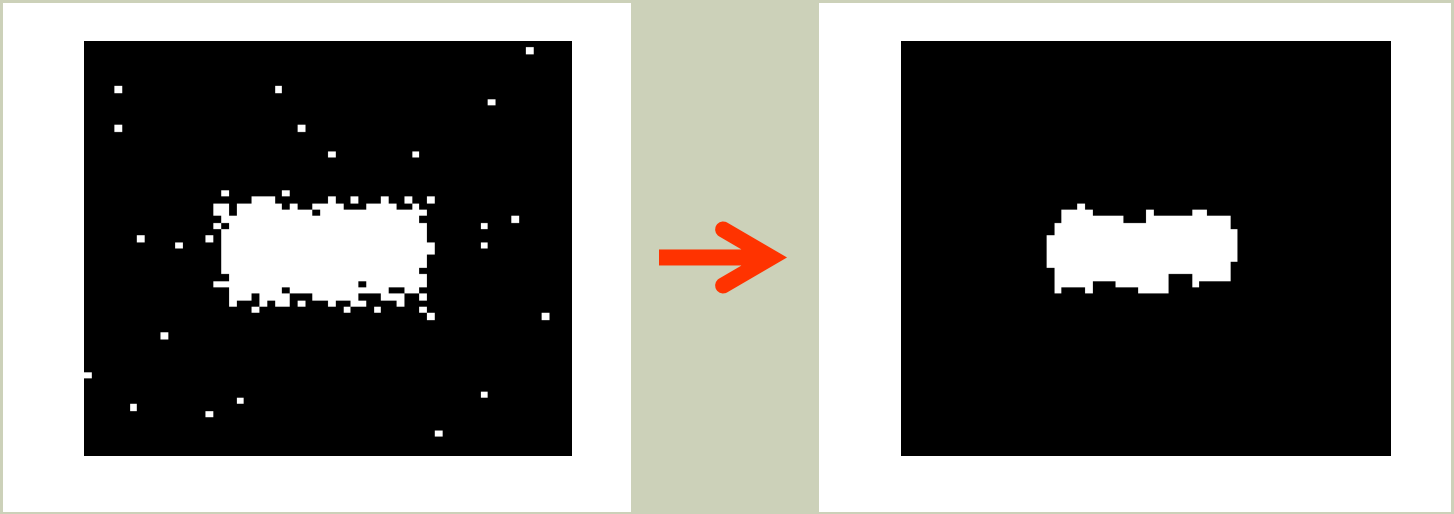


Result

# STRUCTURING ELEMENT IN EROSION EXAMPLE



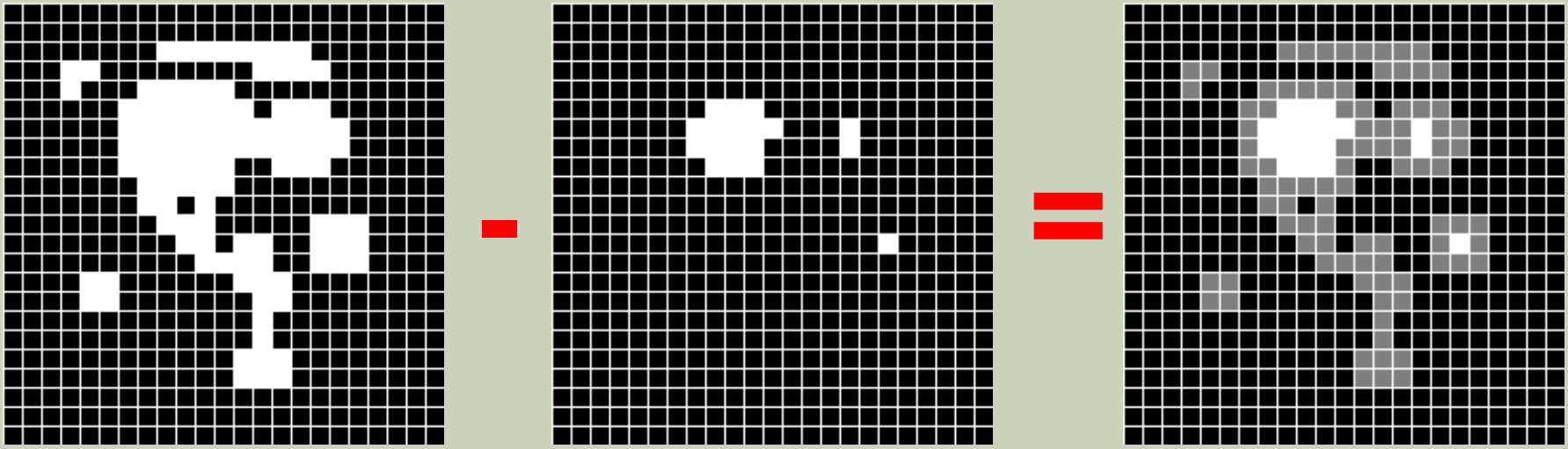
# EXAMPLE OF EROSION APPLICATIONS



- Removing noise
- Removes the outer layer of object pixels

# EXAMPLE OF EROSION APPLICATIONS

- Edge Detection
  - Erosion of an image and then subtracting it away from the original



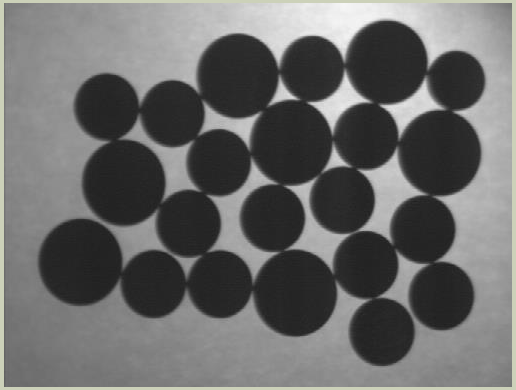
Segmented Image

Erosion Result

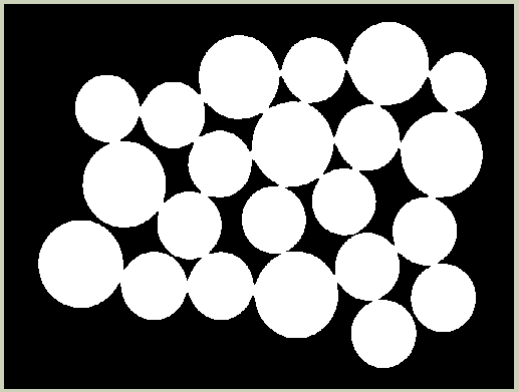
Edge Result

# EXAMPLE OF EROSION APPLICATIONS

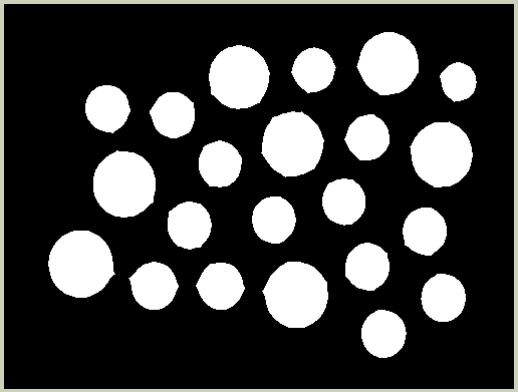
- Simple application of **pattern matching**



Original Image



Segmented Image

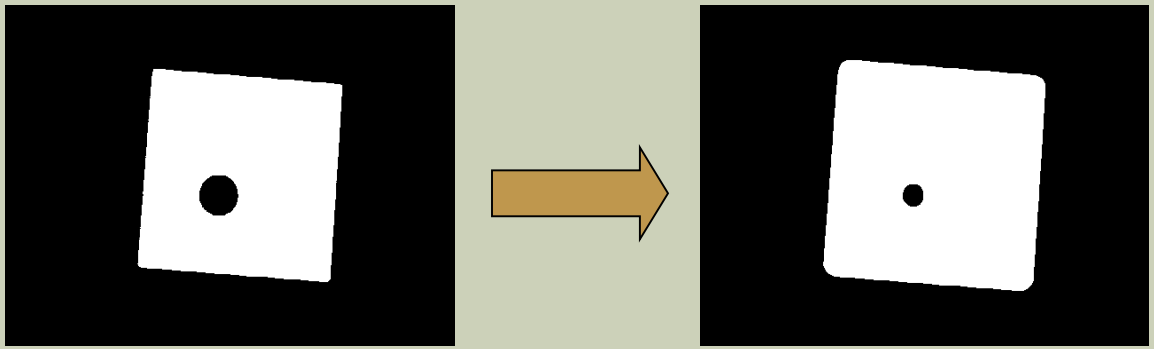


Erosion Result

# DILATION OPERATION

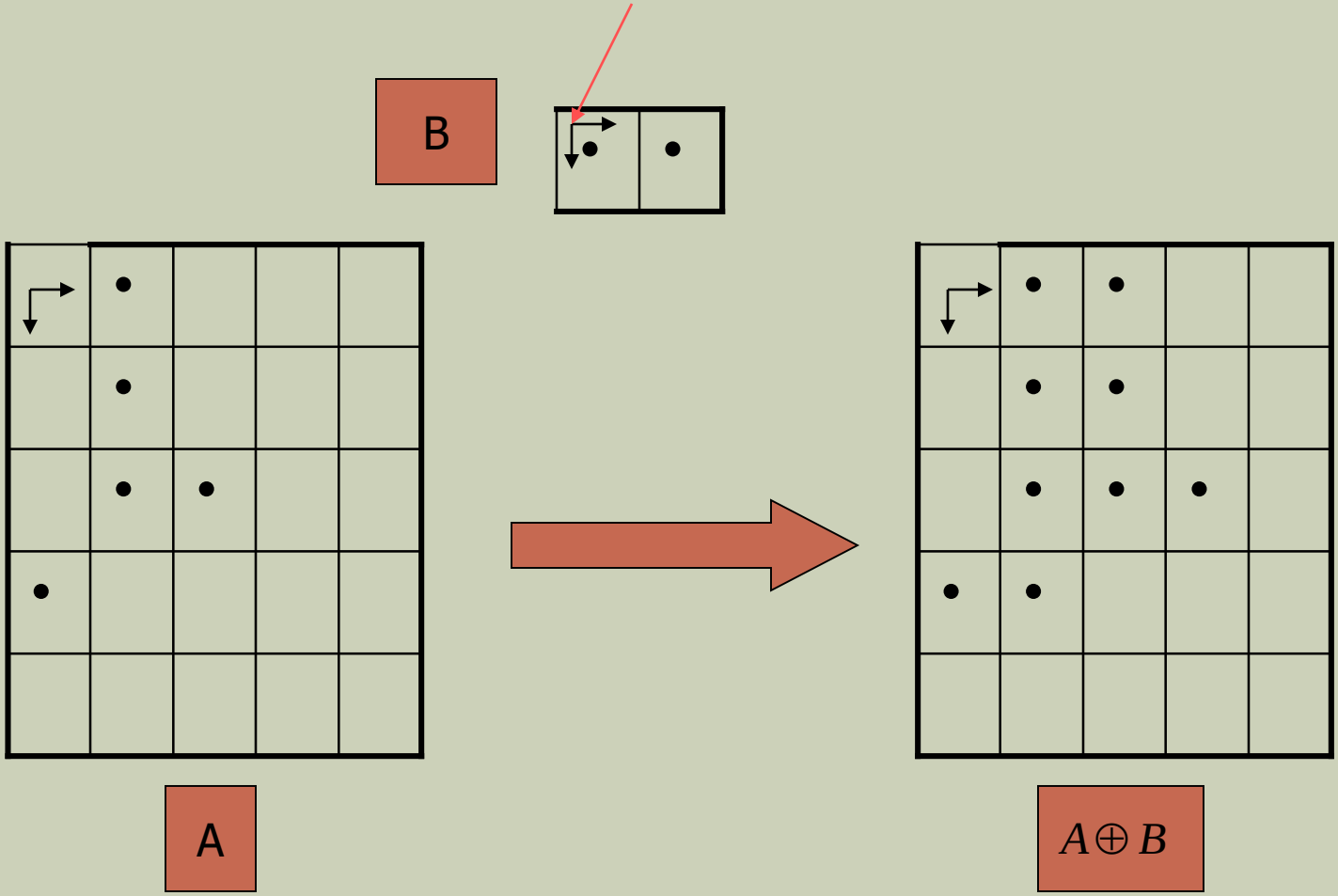
# DILATION OPERATION

- consider each of the *foreground* pixels
- superimpose the structuring element on top
- Consider each foreground pixel in the input image
  - If **any** the structuring element **touches** in foreground pixels,
    - write a "1" at the origin of the structuring element!

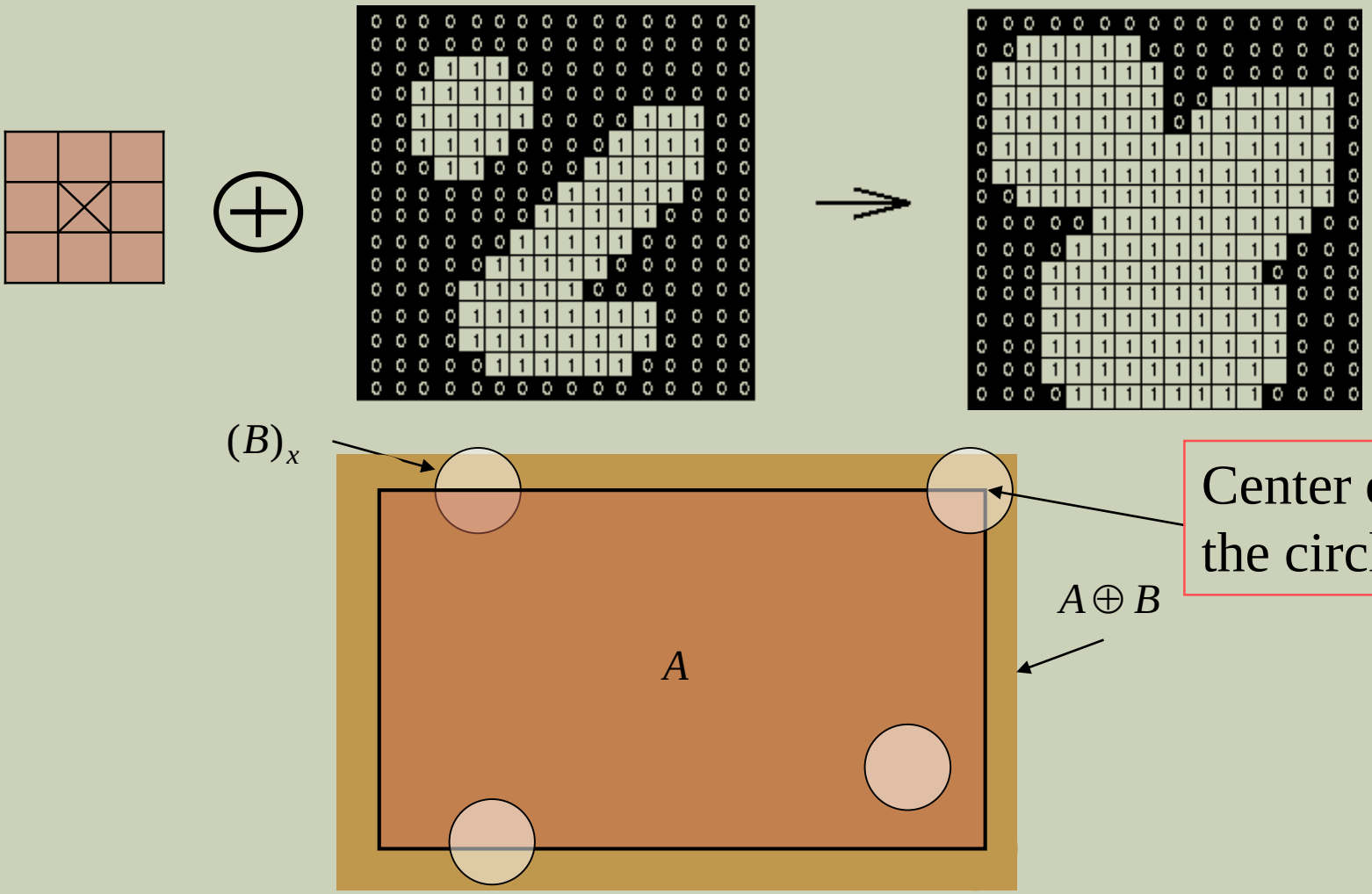


Pixels beyond the boundary of the image are assigned to minimum value -> '0'

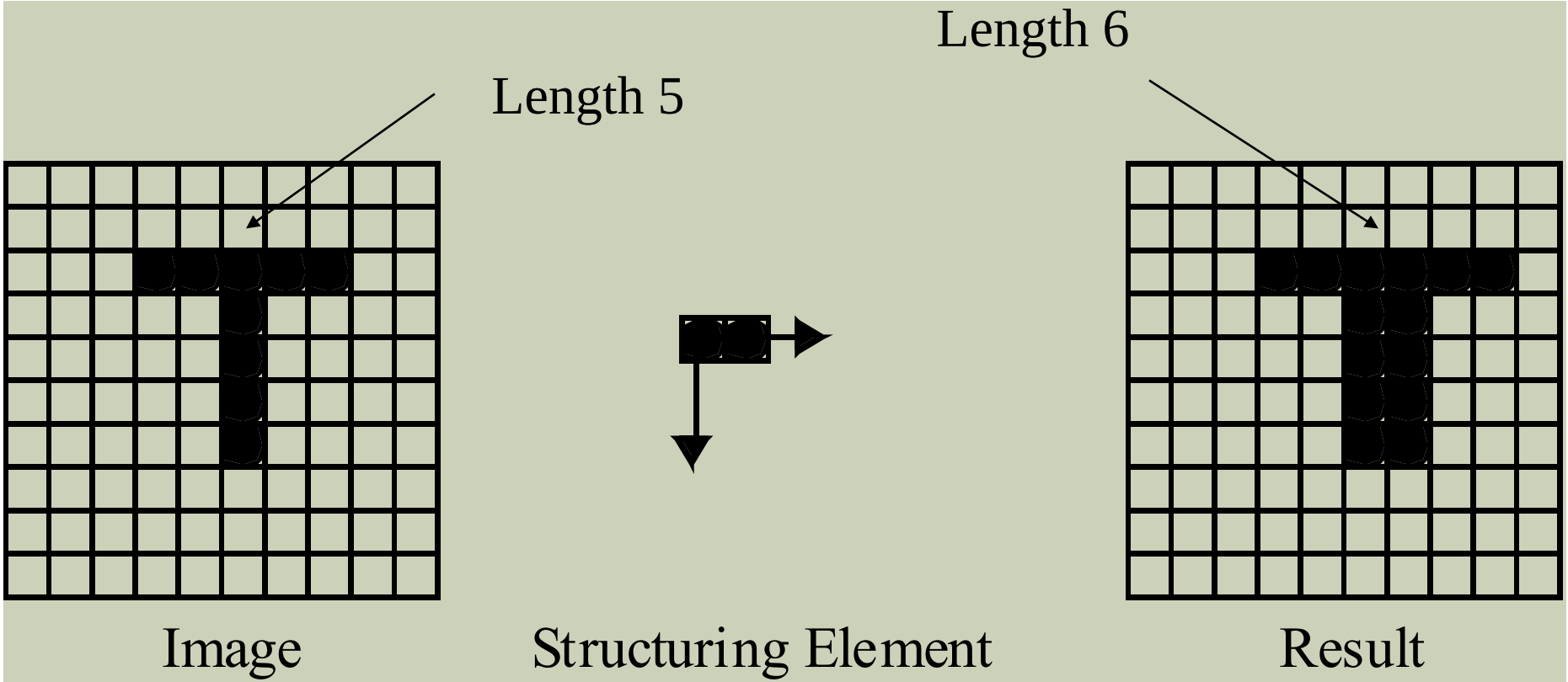
# EXAMPLES OF DILATION OPERATION



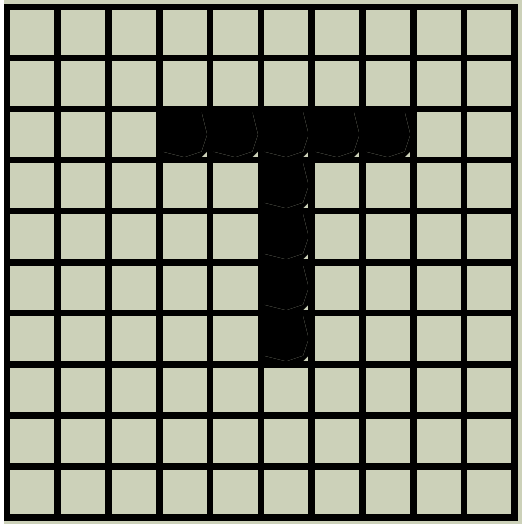
# EXAMPLES OF DILATION OPERATION



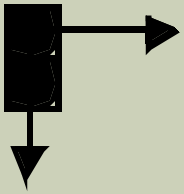
# EXAMPLES OF DILATION OPERATION



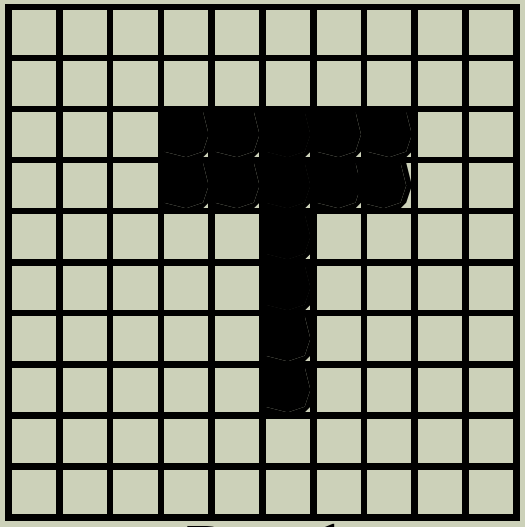
# EXAMPLES OF DILATION OPERATION



Image

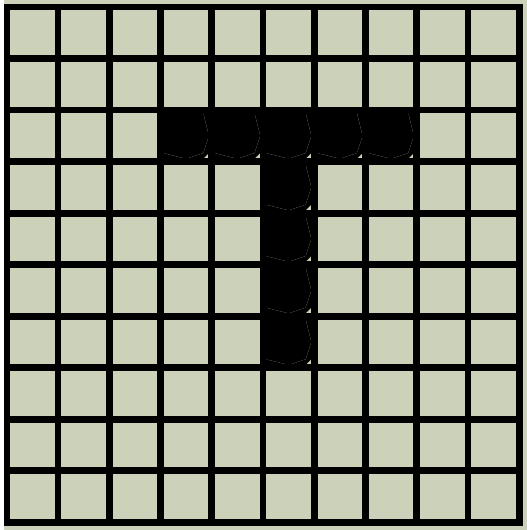


Structuring Element

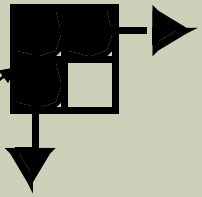


Result

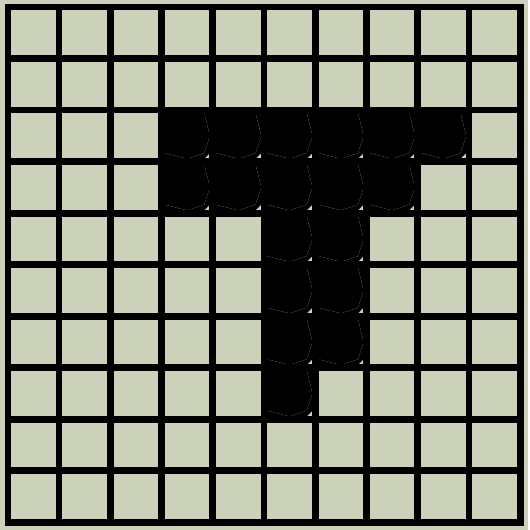
# EXAMPLES OF DILATION OPERATION



Image



Structuring Element

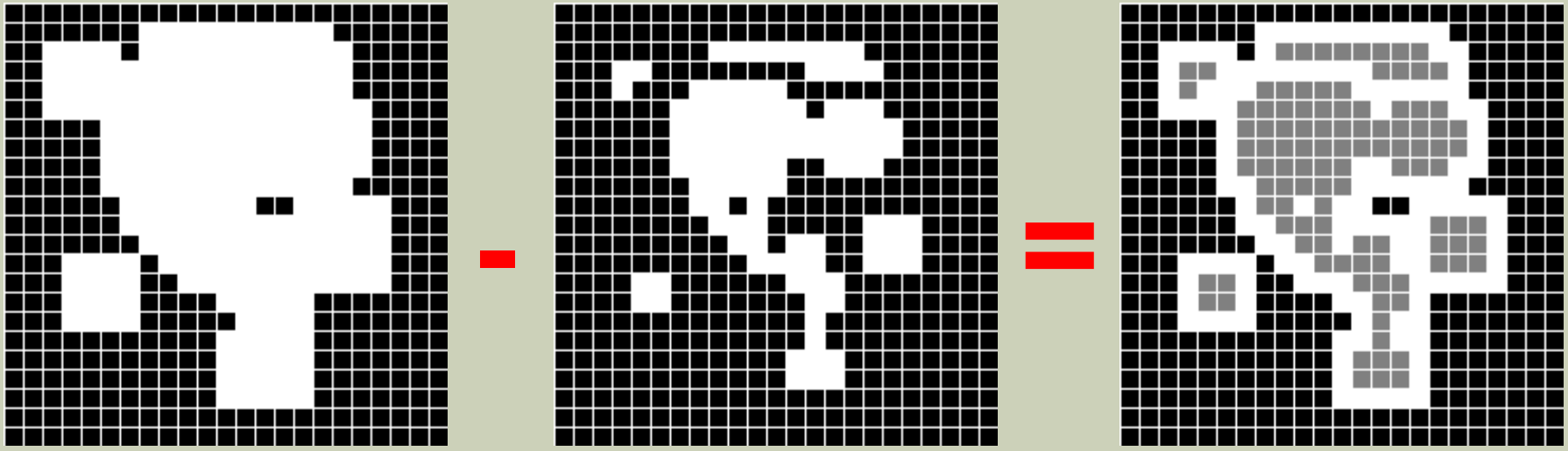


Result

Single point in Image replaced with this in the Result

# EXAMPLE OF DILATION APPLICATIONS

- Edge Detection
  - Dilation of an image and then subtracting it away from the original

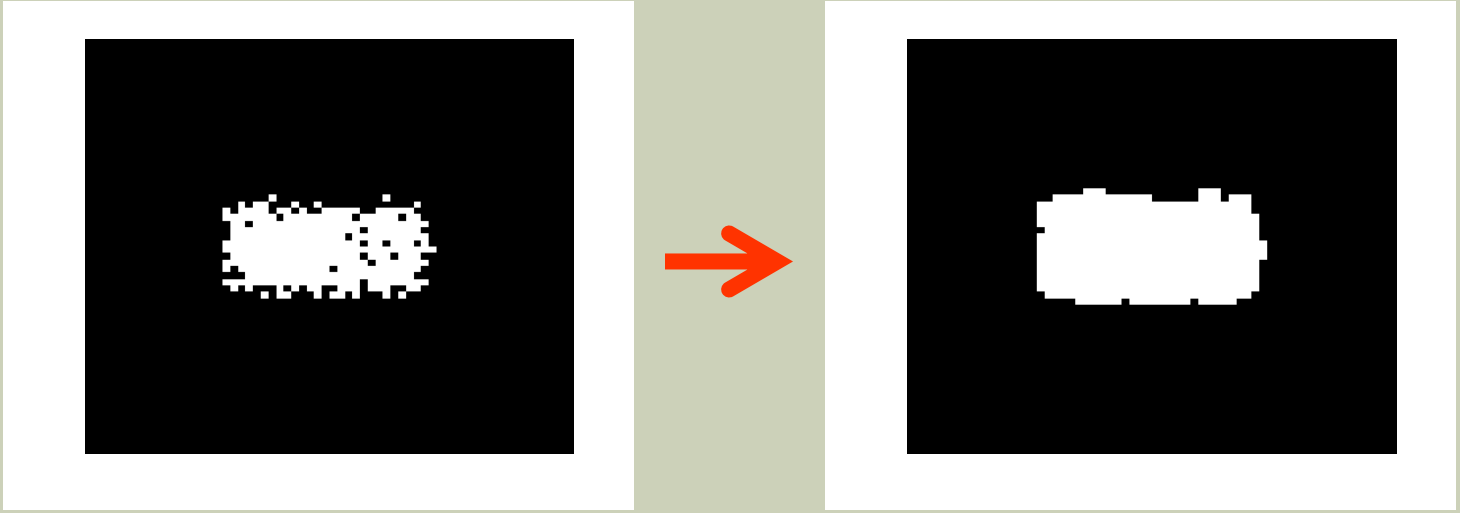


Dilation Result

Segmented Image

Edge Result

# EXAMPLE OF DILATION APPLICATIONS



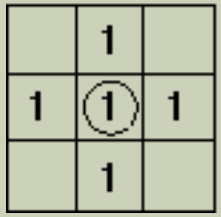
- Fill holes in objects
- Smooth object boundaries.
- Adds an extra outer ring of pixels onto object boundary, ie, object becomes slightly larger

# BOUNDARY HANDLING

- Outside an image
  - It would be either all zeros or ones
  - Sometimes depend on tools
- In Matlab
  - Erosion operation
    - Set outside pixels as zero padding
  - Dilation operation
    - Set outside pixels as one padding

# EXERCISE

**SE =**




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

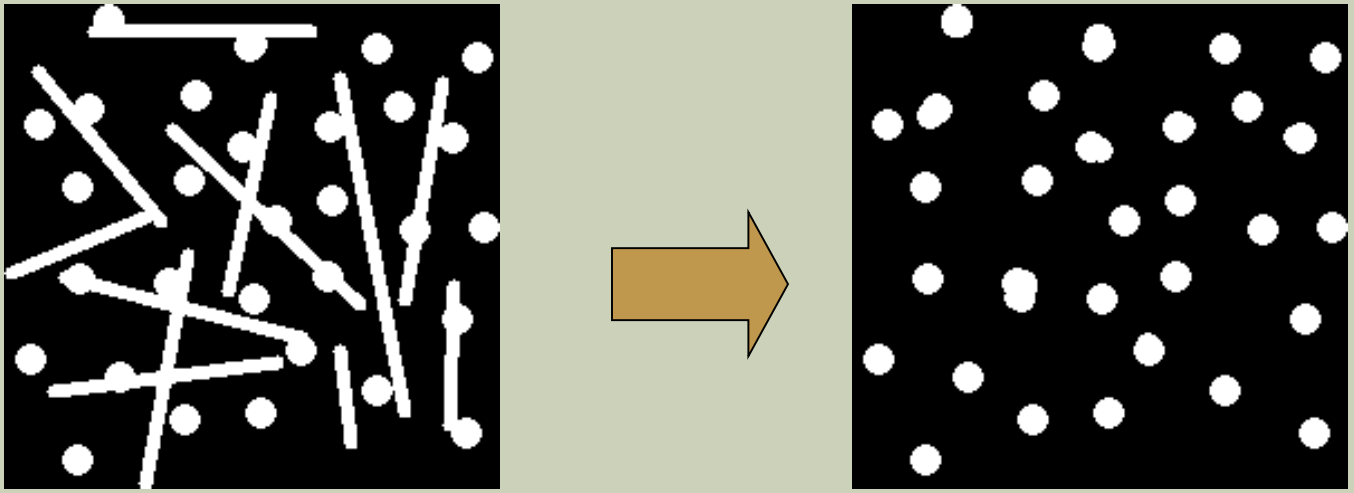
- Exercise#1
  - Erosion -> Dilation
- Exercise#2
  - Dilation -> Erosion

# OPENING OPERATION

# OPENING OPERATION

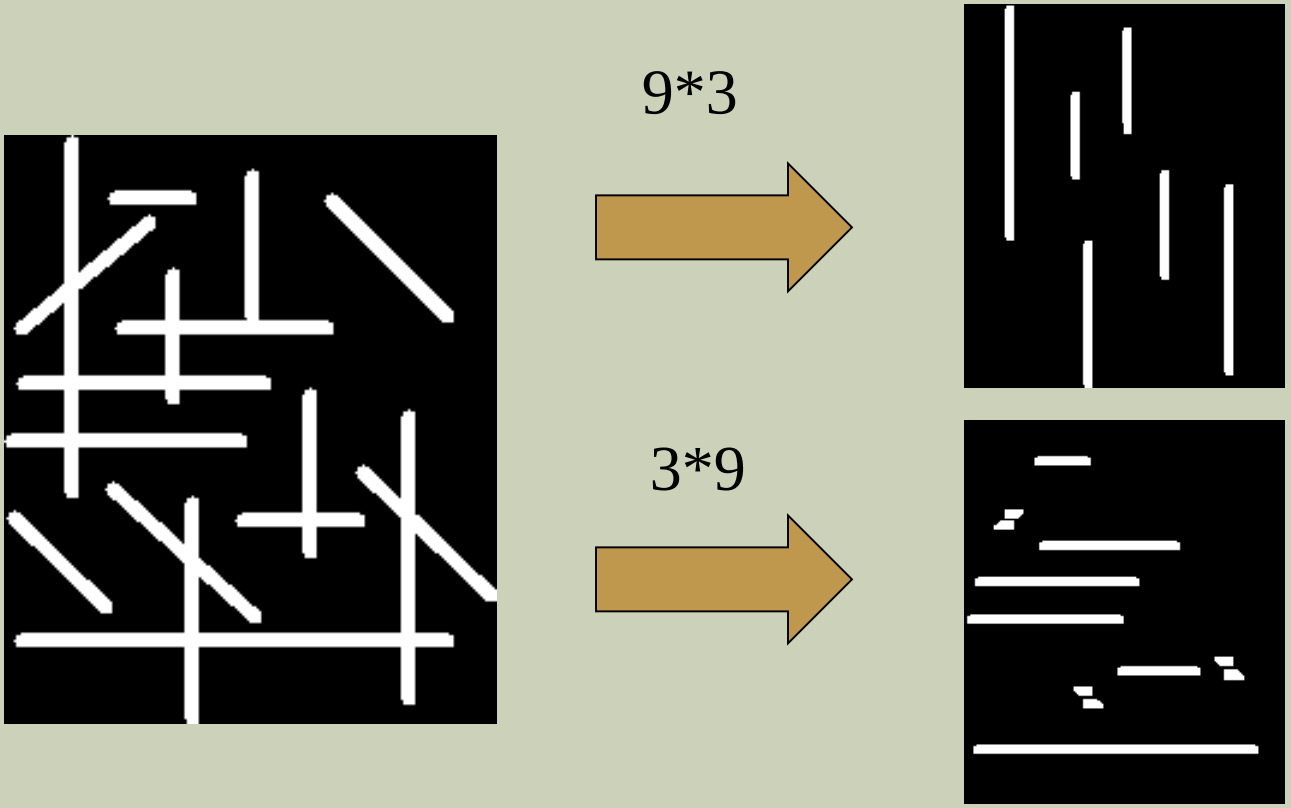
- An erosion followed by a dilation
- the *same structuring element for both operations.*
- Take the structuring element (SE) and slide it around *inside* each foreground region.
  - All pixels which can be covered by the SE with the SE being entirely within the foreground region will be preserved.
  - All foreground pixels which can *not* be reached by the structuring element without lapping over the edge of the foreground object will be eroded away!
- Opening is **idempotent:** Repeated application has no further effects

# EXAMPLE OF OPENING APPLICATIONS



- Opening with a 11 pixel diameter disc
- Preserve circle objects
  - diameter equal or greater than 11 pixels

# EXAMPLE OF OPENING APPLICATIONS




- 9x3 and 3x9 Structuring Element

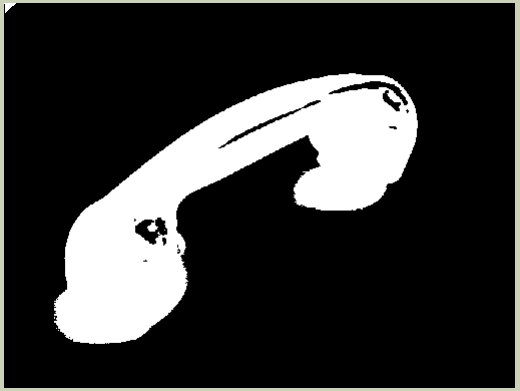
# CLOSING OPERATION

# CLOSING OPERATION

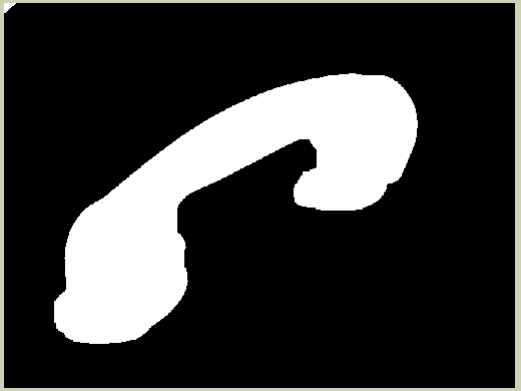
- Dilation followed by erosion
- the *same structuring element for both operations.*
- Take the structuring element (SE) and slide it around *outside* each foreground region.
  - All background pixels which can be covered by the SE with the SE being entirely within the background region will be preserved.
  - All background pixels which can *not* be reached by the structuring element without lapping over the edge of the foreground object will be turned into a foreground.
- Closing is **idempotent:** Repeated application has no further effects

# EXAMPLE OF CLOSING APPLICATIONS

Original Image



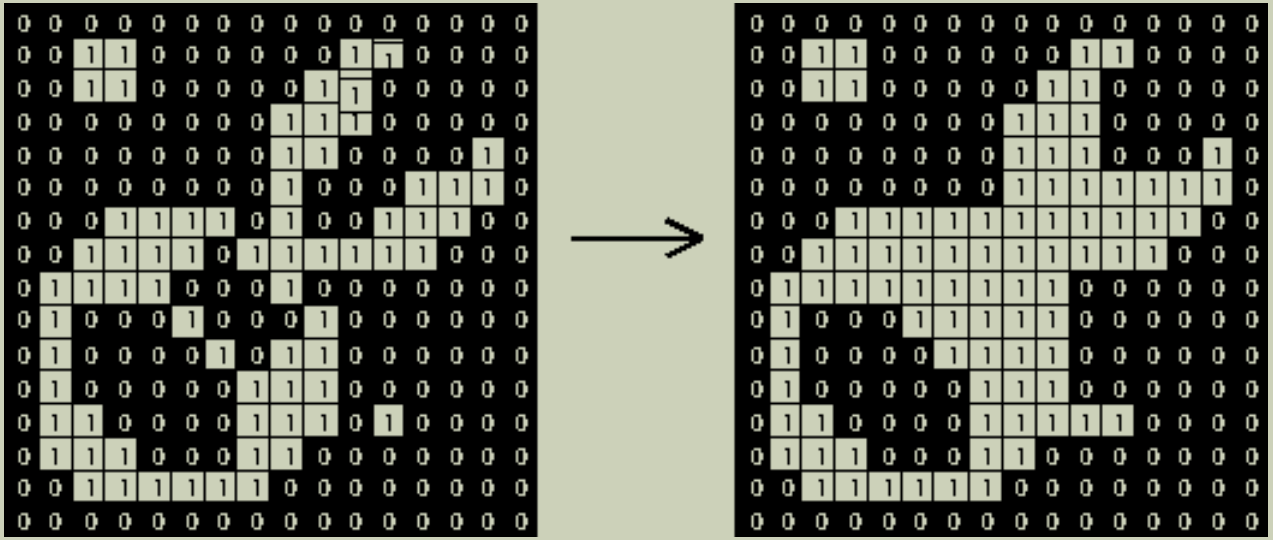
Segmented Image



Closing Result

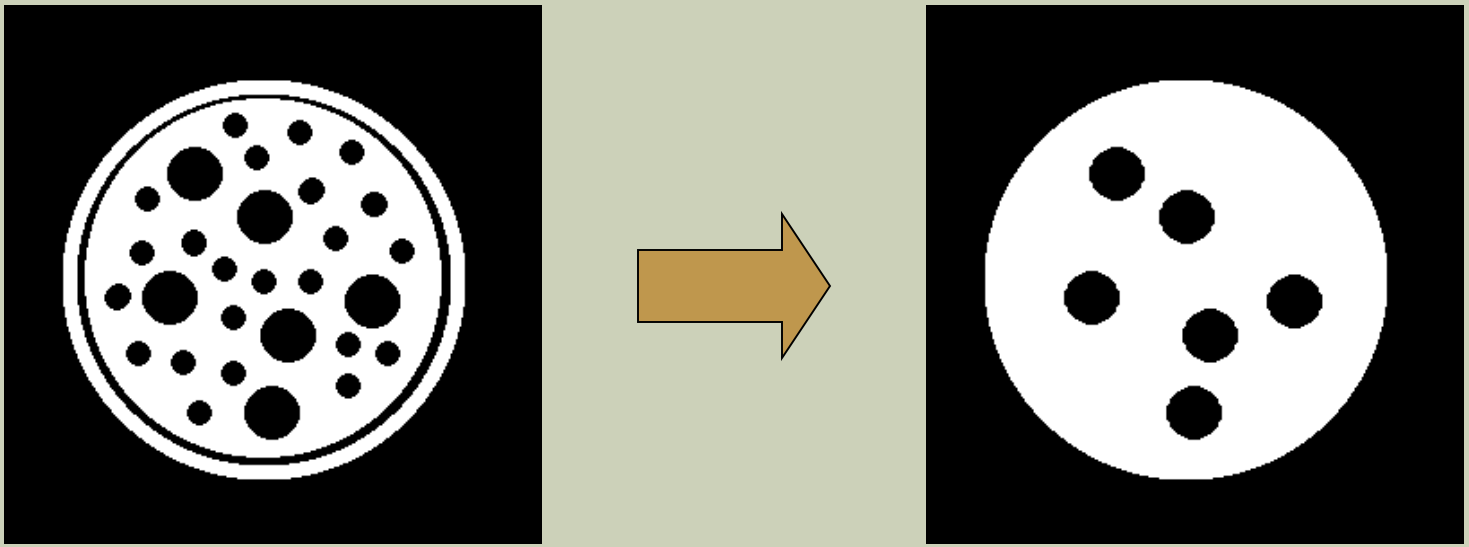
- Closing with disc of size 20

# EXAMPLE OF CLOSING APPLICATIONS



- Structuring element: 3x3 square

# EXAMPLE OF CLOSING APPLICATIONS



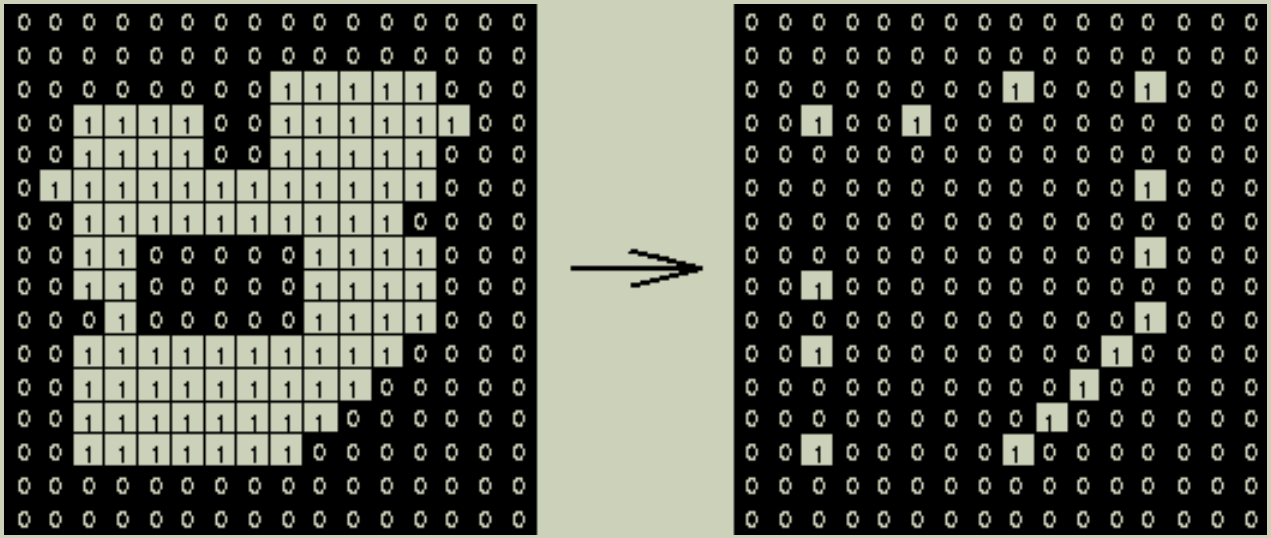
- Closing operation with a 22 pixel disc
- Closes small holes in the foreground

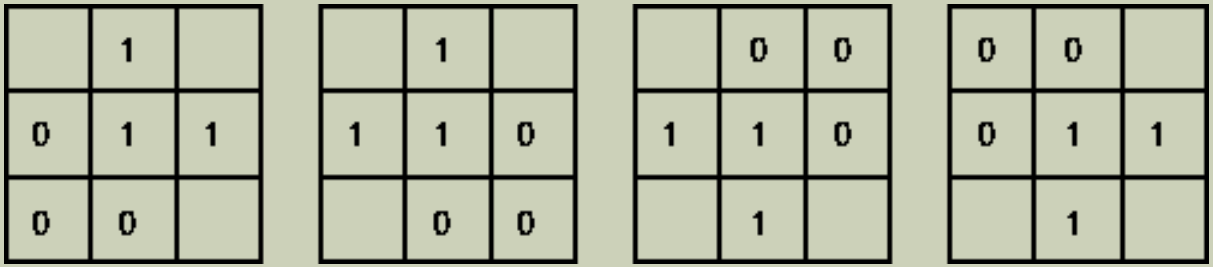
# HIT-MISS OPERATION

# HIT-MISS OPERATION

- look for particular patterns of foreground and background pixels
- Very simple object recognition
- 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 underneath the origin of the structuring element is set to the foreground color.

# EXAMPLE OF HIT-MISS APPLICATIONS






- Corner detection with
  - Structuring Elements representing four corners

# THINNING OPERATION

# THINNING OPERATION

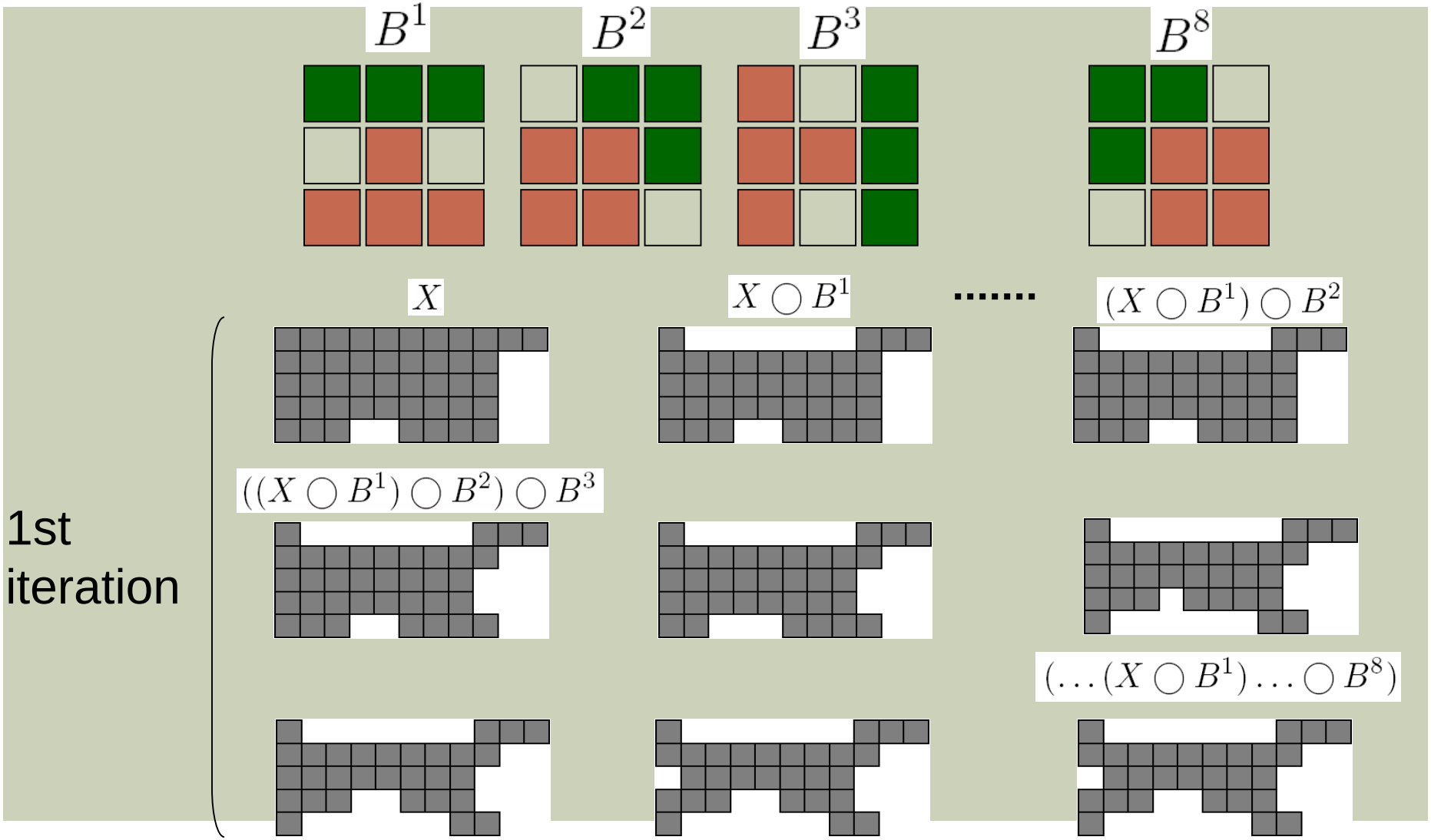
- **remove** selected **foreground pixels** from binary images
- After edge detection, lines are often **thicker than one pixel**.
  - **Thinning** can be used to thin those line to **one pixel width**.
- Let ***K*** be a kernel and ***I*** be an image

$$thin(I, K) = I - HitAndMiss(I, K)$$

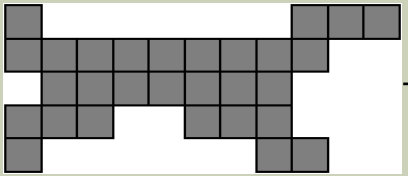
with 0-1=0!!

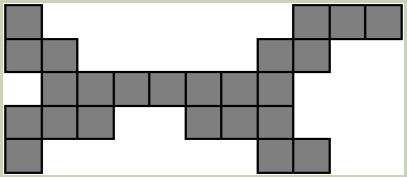
- **If** foreground and background **fit** the structuring element exactly, **then** the pixel at the origin of the SE is set to 0
- Note that the value of the SE at the origin is 1 or *don't care*!

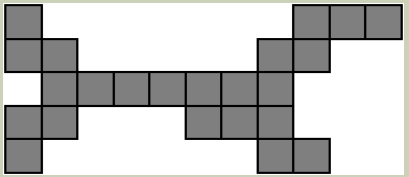
# THINNING OPERATION



# THINNING OPERATION

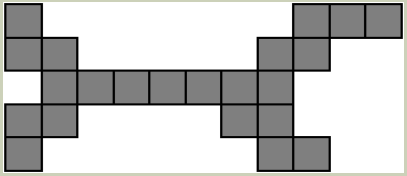


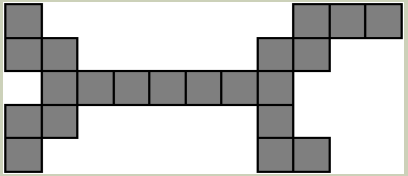




.....

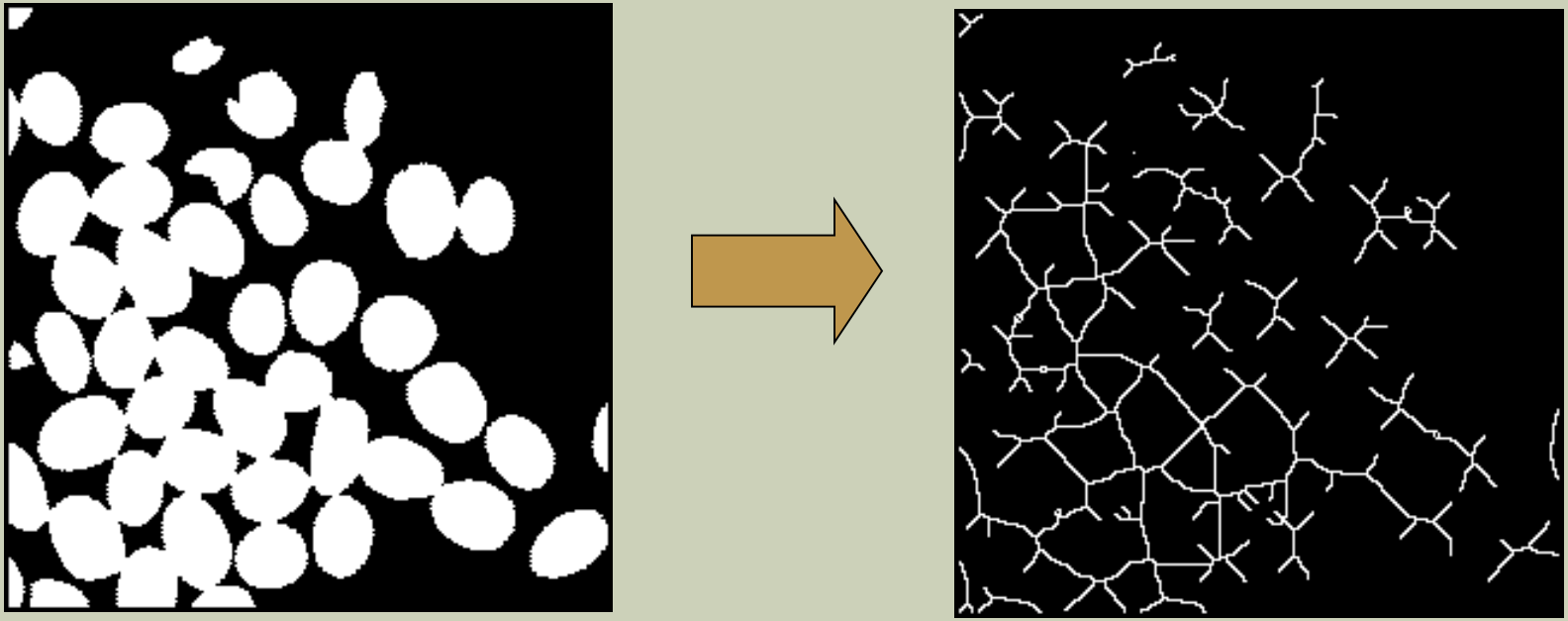
result of
1st iteration



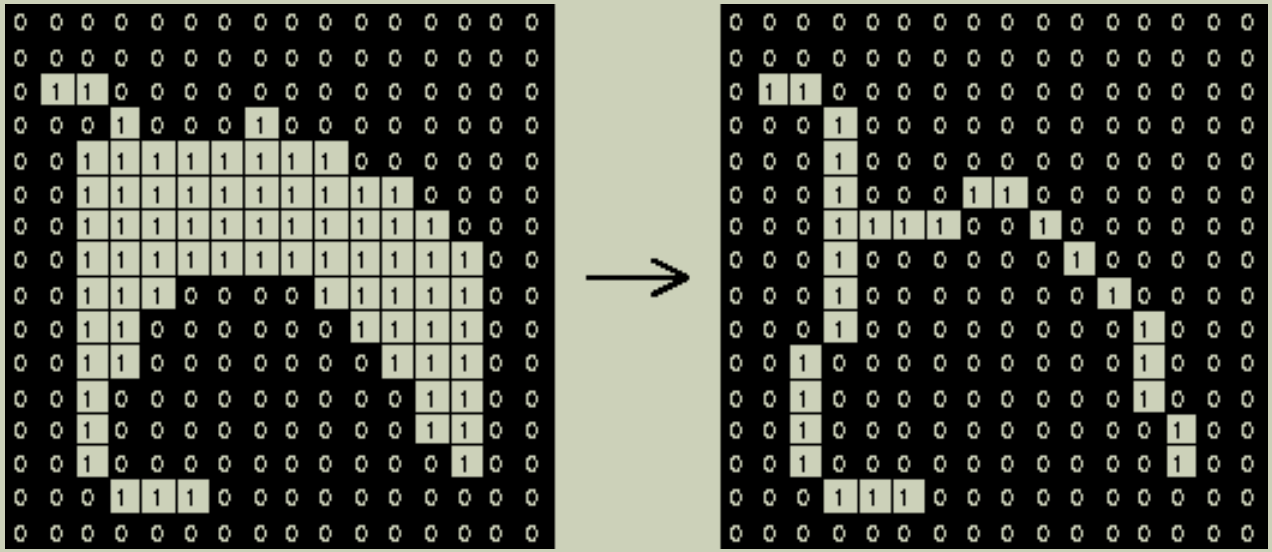


2nd iteration reaches
idempotence

# THINNING OPERATION



- 20 iterations of thinning color white



| | | |
|---|---|---|
| 0 | 0 | 0 |
| | 1 | |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| | 0 | 0 |
| 1 | 1 | 0 |
| | 1 | |

- Corner detection with
  - Structuring Elements representing two corners