# CSE4227 Digital Image Processing

## Chapter 9 – **Morphological Image Processing- (Part-I)**


Dr. Kazi A Kalpoma

Professor, Department of CSE

Ahsanullah University of Science & Technology (AUST)

Contact: kalpoma@aust.edu


Google Class code: bux3jc2

# Today's Contents

- **Morphological image processing: pixel shape based analysis**
  - **Structuring Element**
  - **basic morphology operations**
    - Dilation - grow image regions
    - Erosion - shrink image regions
    - Opening - structured removal of image region boundary pixels
    - Closing - structured filling in of image region boundary pixels

Chapter 9 from R.C. Gonzalez and R.E. Woods, Digital Image Processing (3rd Edition), Prentice Hall, 2008 [ **Section 9.1, 9.2, 9.3** ]

https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

# Morphological Image Processing

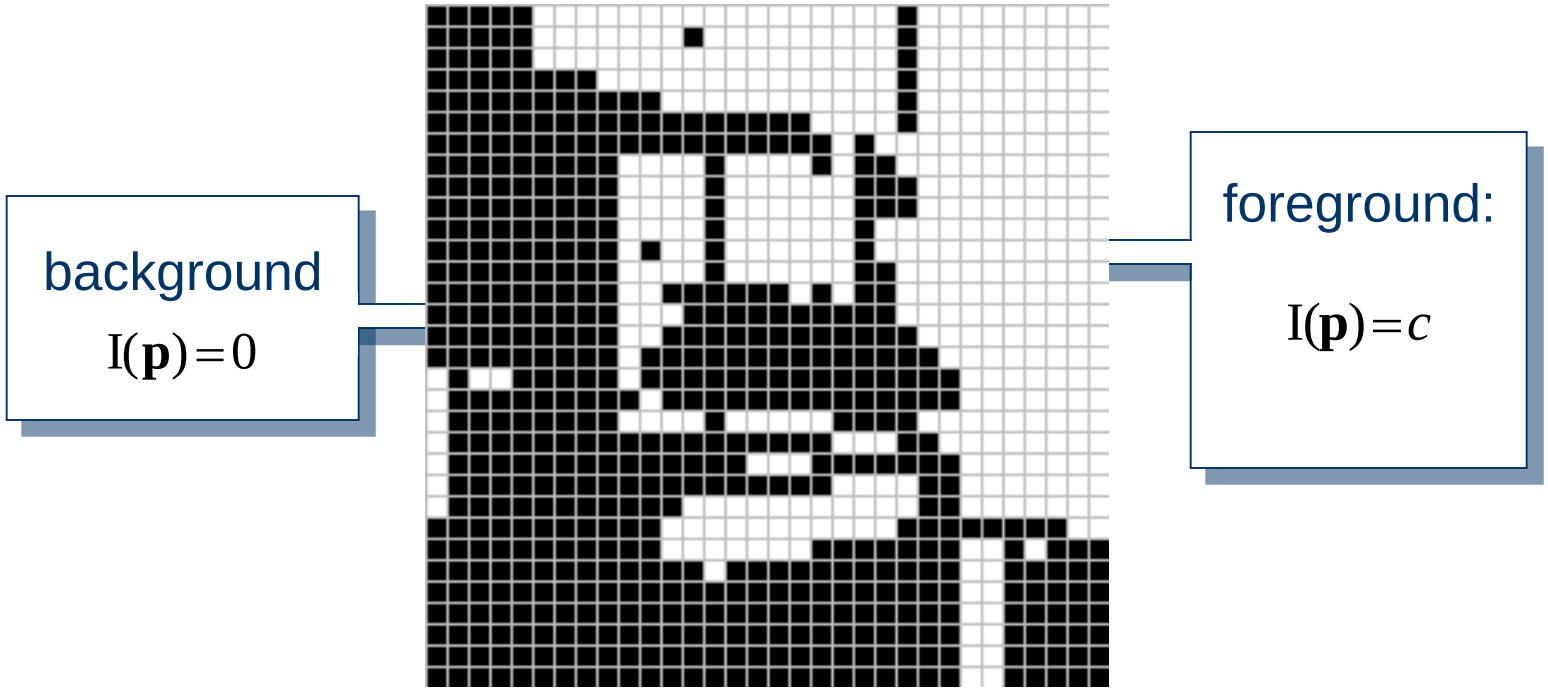
- A tool for **extracting image components** that are useful in the representation and description of image.
- It deals with the shape (or morphology) of objects/features in an image
- Rely only on the **relative ordering of pixel values**, not on their numerical values
- Probe an image with a small shape or template called a **structuring element**.
- The language of mathematical morphology of an image is Set Theory

# Morphological Image Processing

- Morphology applies certain simple rules and shapes (such as squares, circles, diamonds, cubes and spheres) to process images.
- The objective is usually to identify features of interest in images
  - as a prelude to performing high-level inspection, or
  - machining, functions.
- Two kinds :
  - Binary Morphology
  - Grey Level Morphology
- Four basic morphology operations for **binary images**:
  - **Erosion**
  - **Dilation**
  - **Openning and**
  - **Closing**

# Binary Image

- **Representation of individual pixels as 0 or 1, convention:**
  - **foreground, object = 1 (white)**
  - **background = 0 (black)**



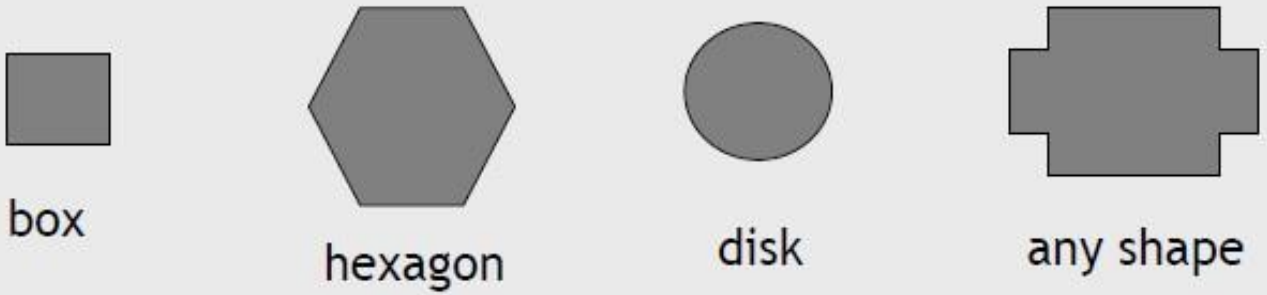

This represents a digital image. Each square is one pixel.

# Structuring Element

A small image/template that helps to produce new image from the old one i.e. a small binary array.

A structuring element is a shape mask used in the basic morphological operations.

They can be any shape and size that is digitally representable, and each has an origin.

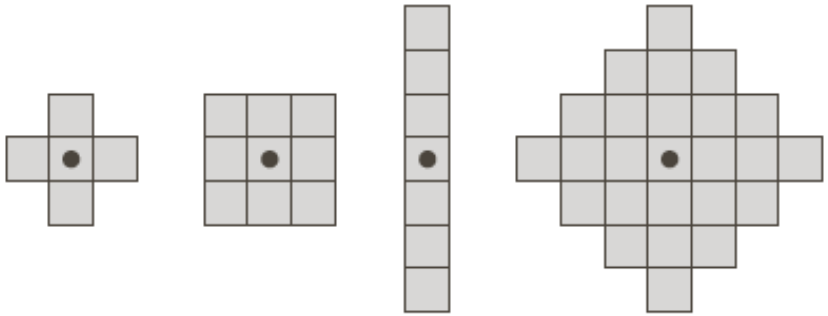



box(length,width)

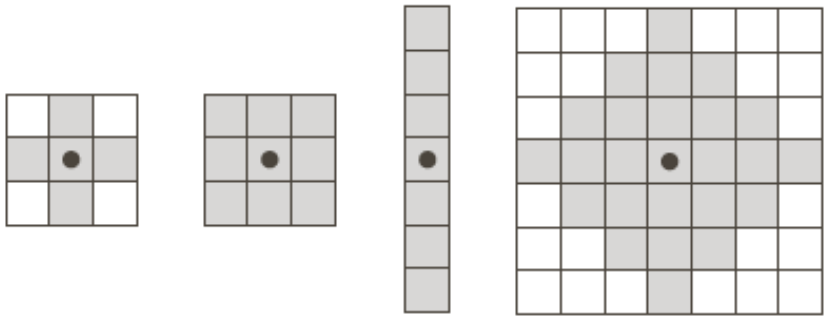
disk(diameter)

# Structuring Element

A structuring element is a small image – used as a moving window



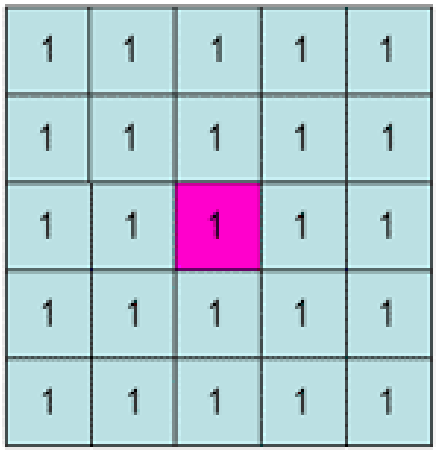
Example Structuring Elements



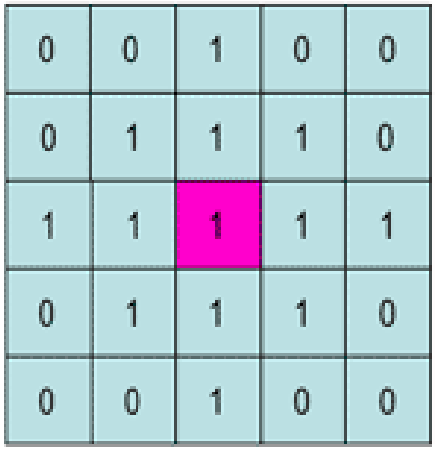
Structuring Elements converted to rectangular arrays

# Structuring Element

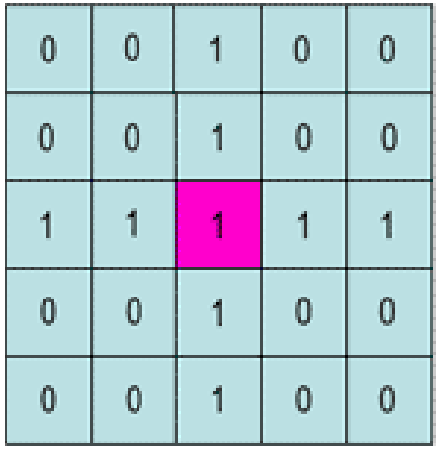
- For simplicity we will use rectangular structuring elements with their origin point.
- Origin point can be at the middle pixel or at any position.



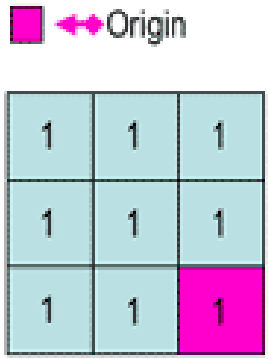
Square 5x5 element



Diamond-shaped 5x5 element



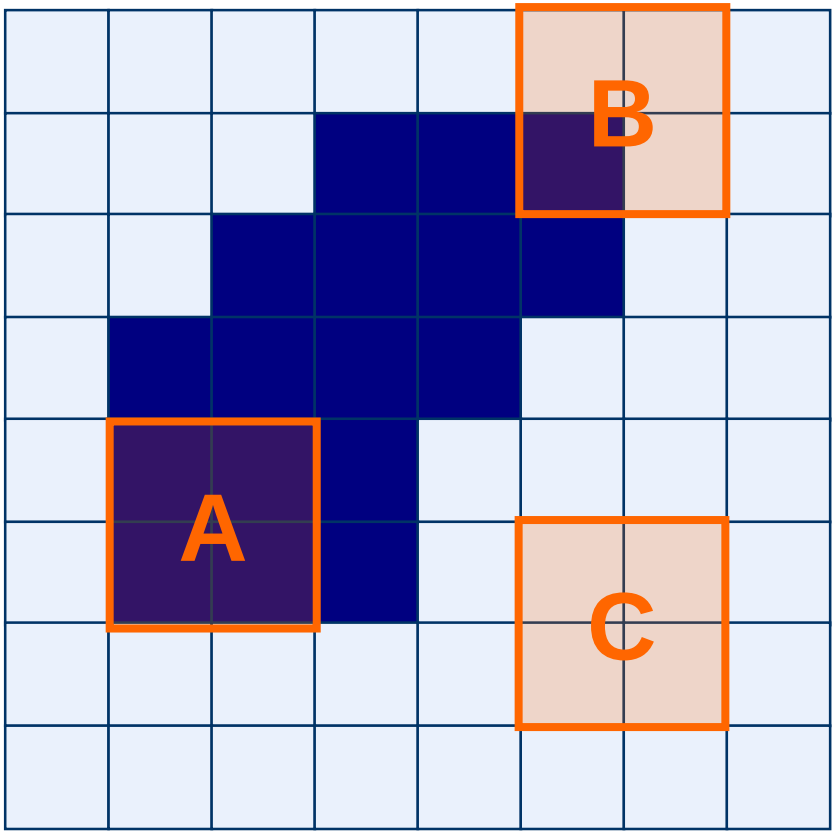
Cross-shaped 5x5 element

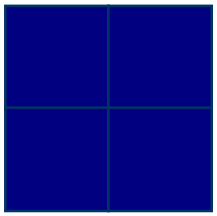


Square 3x3 element

❖ By default origin point is at the center

# Structuring Elements: Hits & Fits



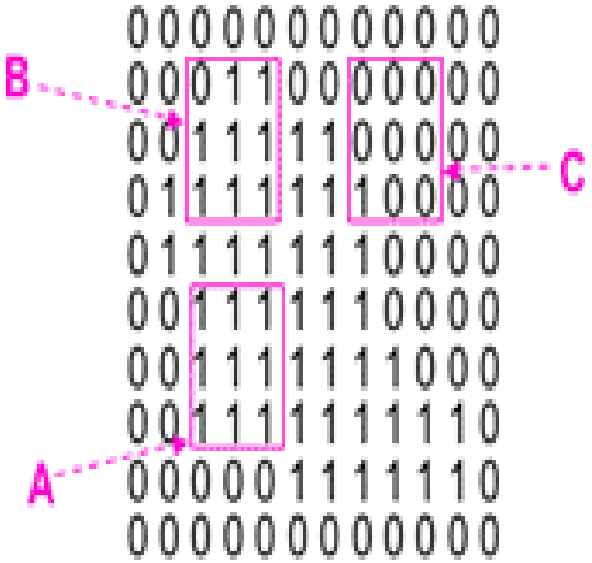
Structuring Element

**Fit:** All "***ON"*** *pixels* in the structuring element cover "***ON"*** *pixels* in the image

**Hit:** Any "***ON"*** *pixel* in the structuring element covers an "***ON"*** *pixel* in the image

All morphological processing operations are based on these simple ideas

# Fitting and hitting of a binary image with structuring elements $s_1$ and $s_2$.




$$s_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$s_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

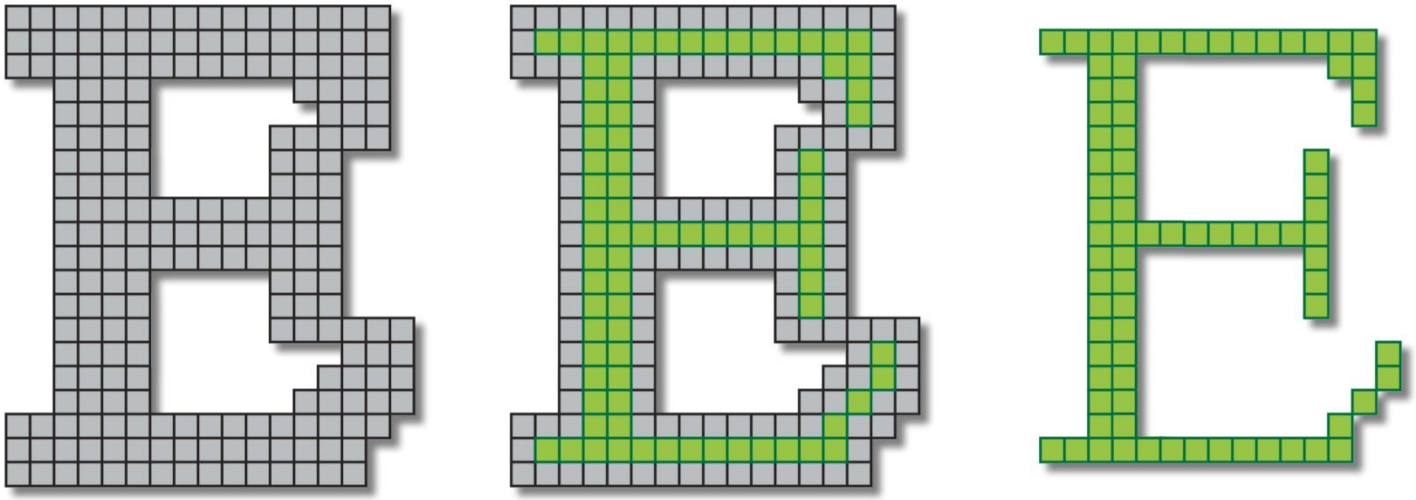
| | | A | B | C |
|---|---|---|---|---|
| **fit** | $s_1$ | yes | no | no |
| | $s_2$ | yes | yes | no |
| **hit** | $s_1$ | yes | yes | yes |
| | $s_2$ | yes | yes | no |

# Fundamental Operations

- Fundamentally morphological image processing is very like spatial filtering
- The structuring element is moved across every pixel in the original image to give a pixel in a new processed image
- The value of this new pixel depends on the operation performed

There are two basic morphological operations: **Erosion** and **Dilation**

# Erosion

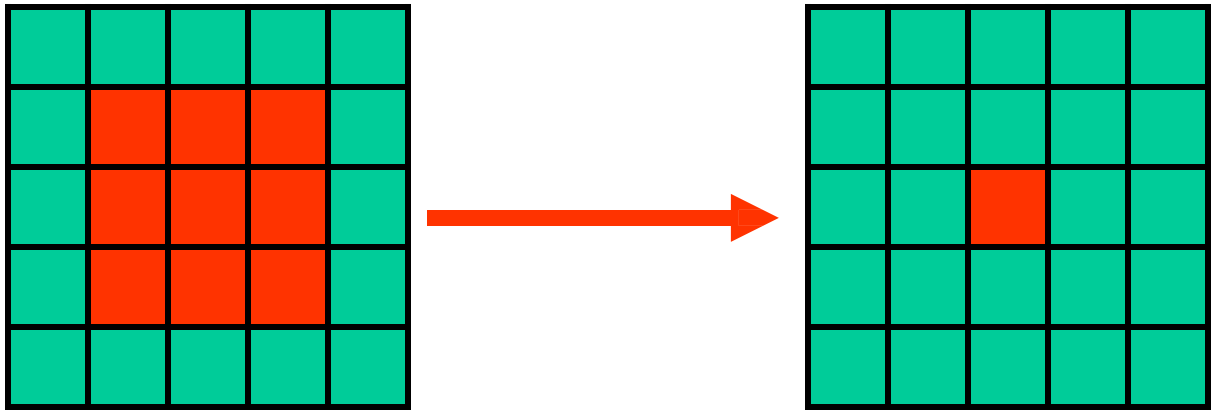


**Shrink the object**

Shrinks foreground, enlarges Background

# Erosion

- Erosion is used for shrinking of element A by using element B.
- One of the simplest uses of erosion is for eliminating irrelevant details from a binary image.



Original Image

Eroded Image

# Erosion

Definition 1:

**Fit:** All "***ON***" *pixels* in the structuring element cover "***ON***" *pixels* in the image

- Does the structuring element **fit the set?**

$$A \ominus B = \{ z \mid (B)_z \subseteq A \}$$

- Erosion of a set A by structuring element B:

  - Set of all points z, such that B translated by z is contained in A.

If YES, output pixel g(x,y) will be foreground (i.e. 1)

# Erosion

Definition 2:

Does the structuring element **fit the set?**

Erosion of image *f* by structuring element *s* is given by $f \ominus s$

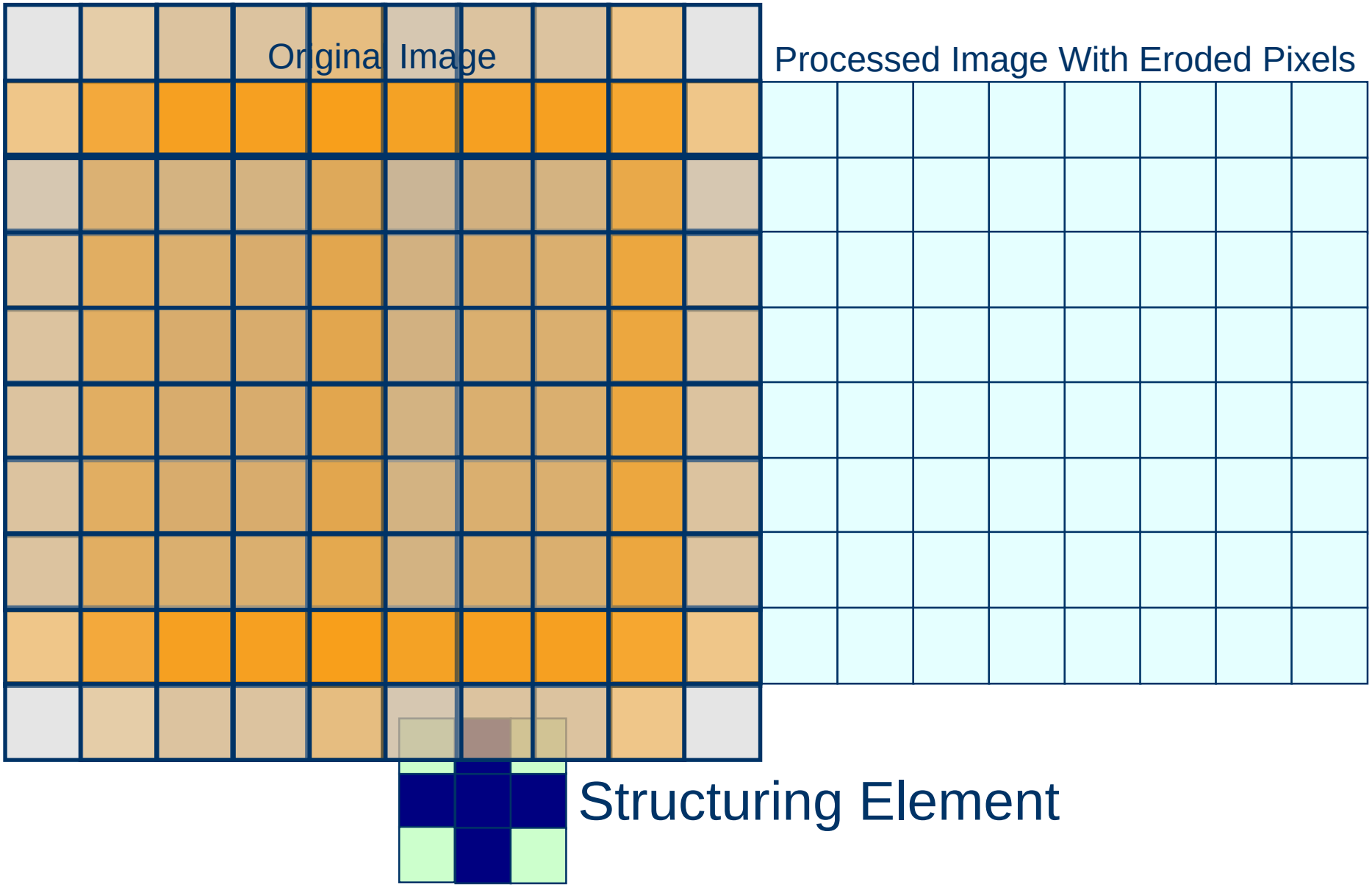
The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

$$g(x,y) = \begin{cases} 1 \text{ if } s \text{ fits } f \\ 0 \text{ otherwise} \end{cases}$$

# Erosion – How to compute

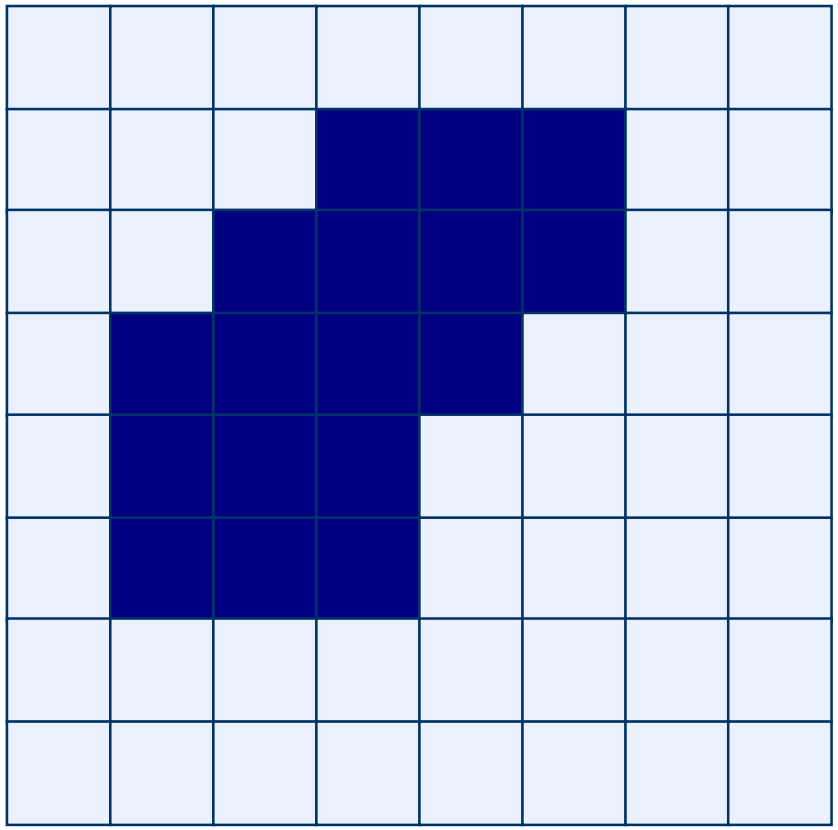
- For each foreground pixel (which we will call the *input pixel*)
  - Superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position.
  - If *for every* pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is.
  - If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.

# Example for 2D Erosion

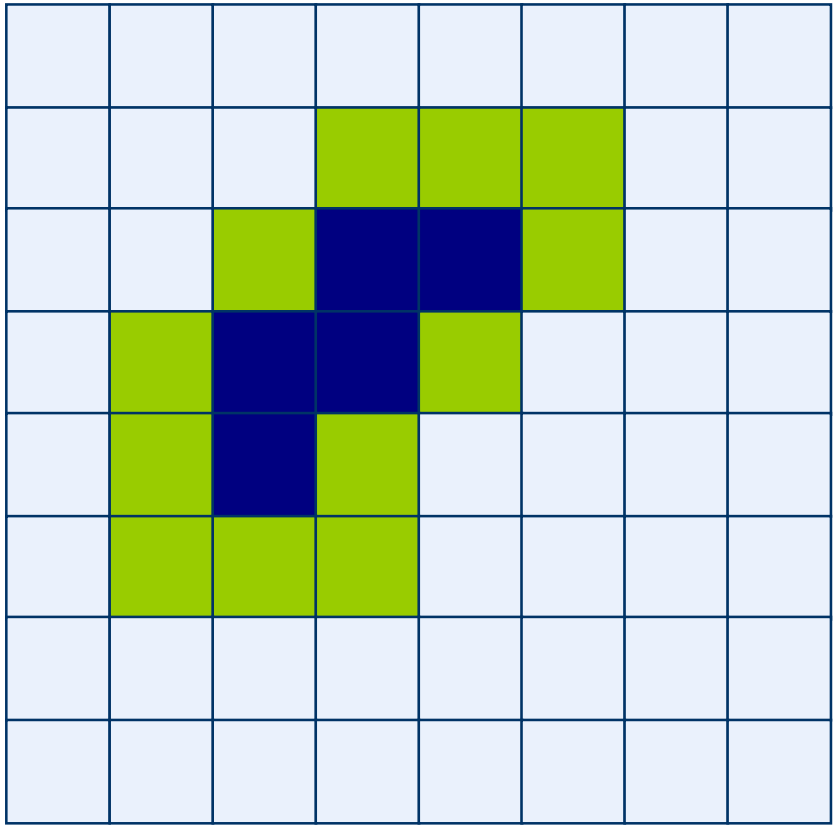


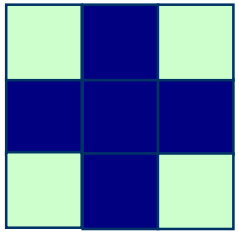
# Erosion: Example 1

Original Image



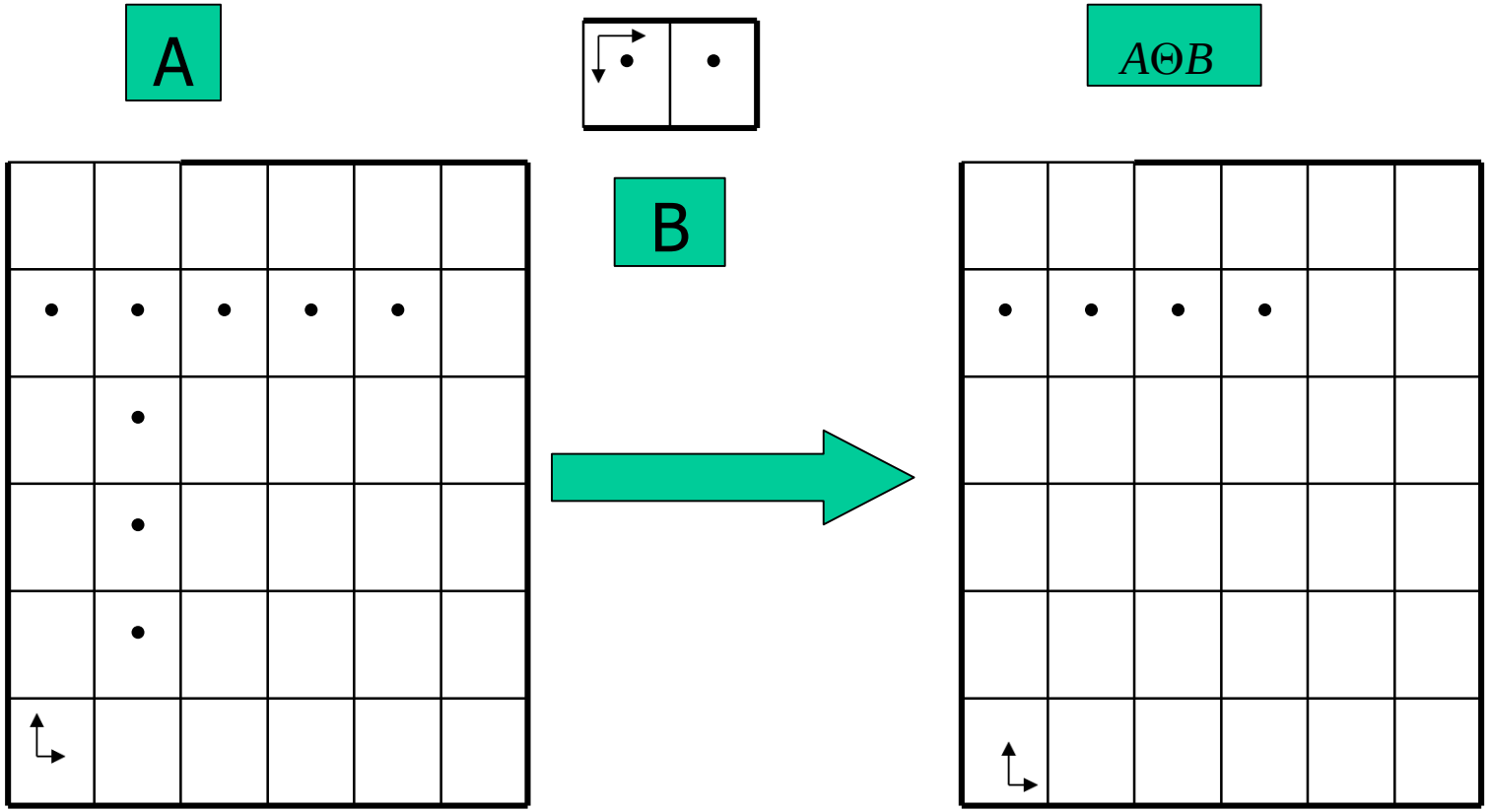
Processed Image



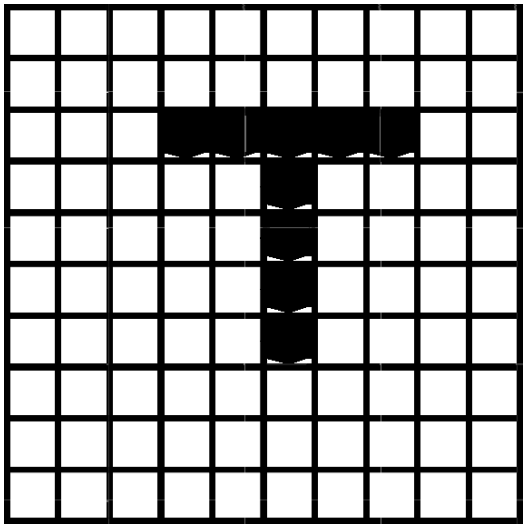


Structuring Element

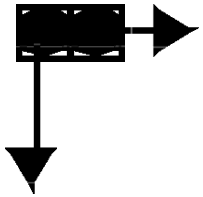
# Erosion explained pixel by pixel



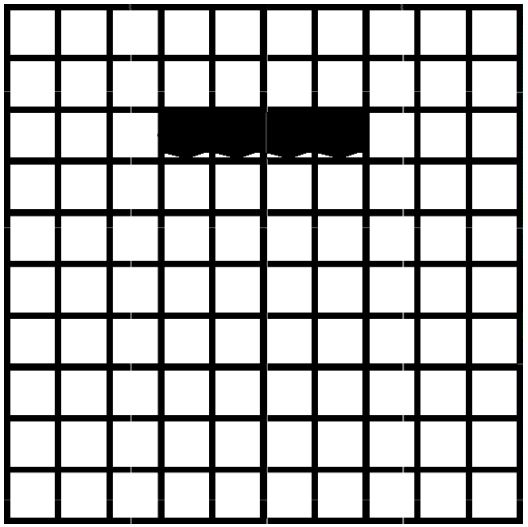
# Structuring Element in Erosion Example



Image

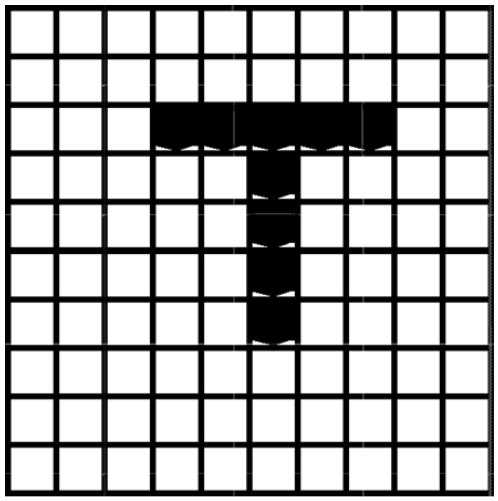


Structuring Element

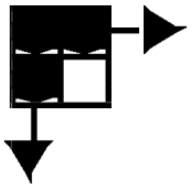


Result

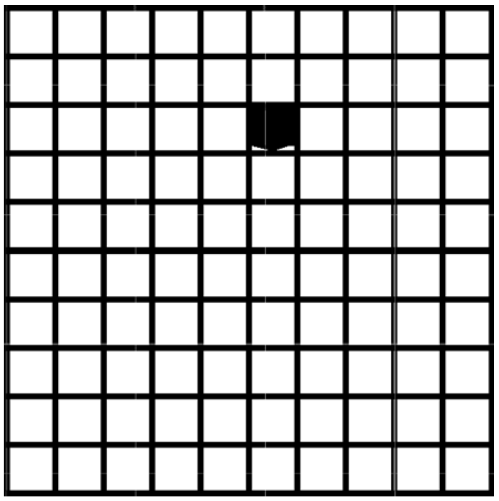
# Structuring Element in Erosion Example



Image

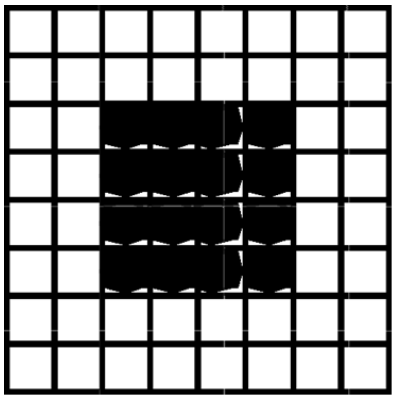


Structuring Element

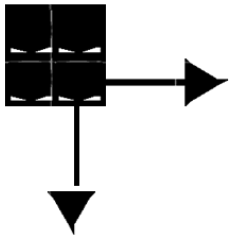


Result

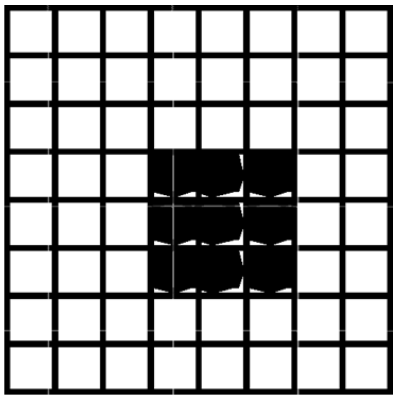
# Structuring Element in Erosion Example



Image

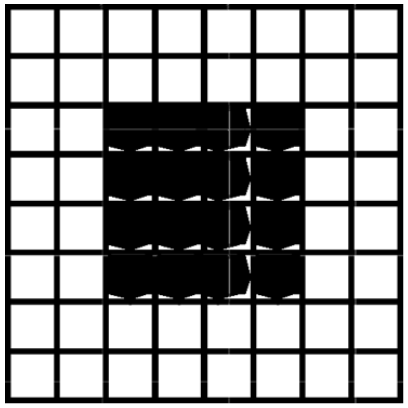


Structuring Element

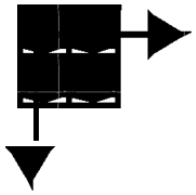


Result

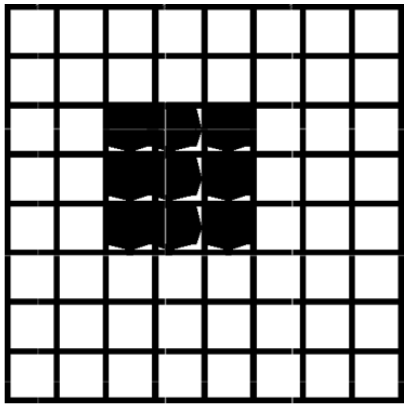
# Structuring Element in Erosion Example



Image

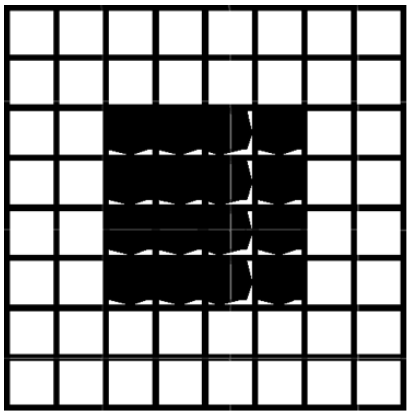


Structuring Element

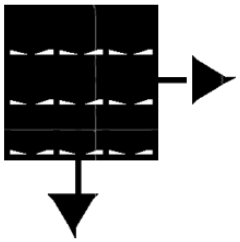


Result

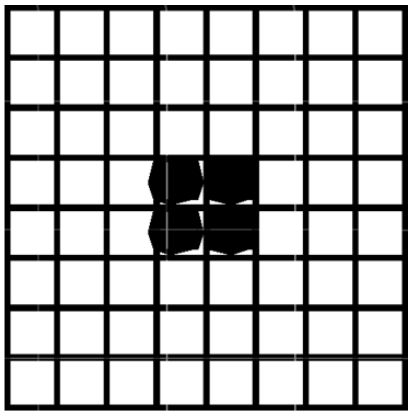
# Structuring Element in Erosion Example



Image

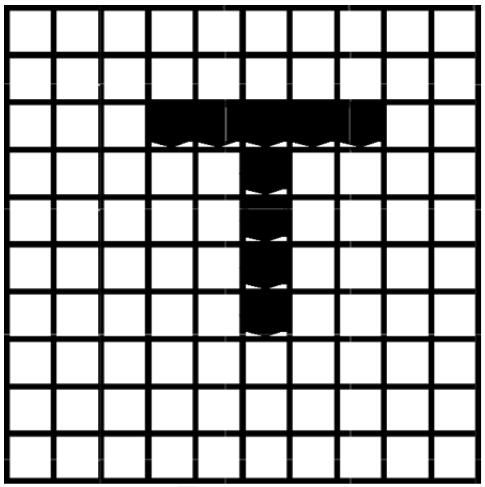


Structuring Element

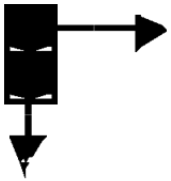


Result

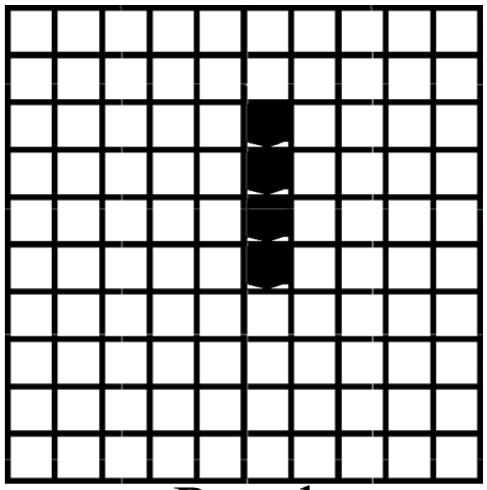
# Structuring Element in Erosion Example



Image

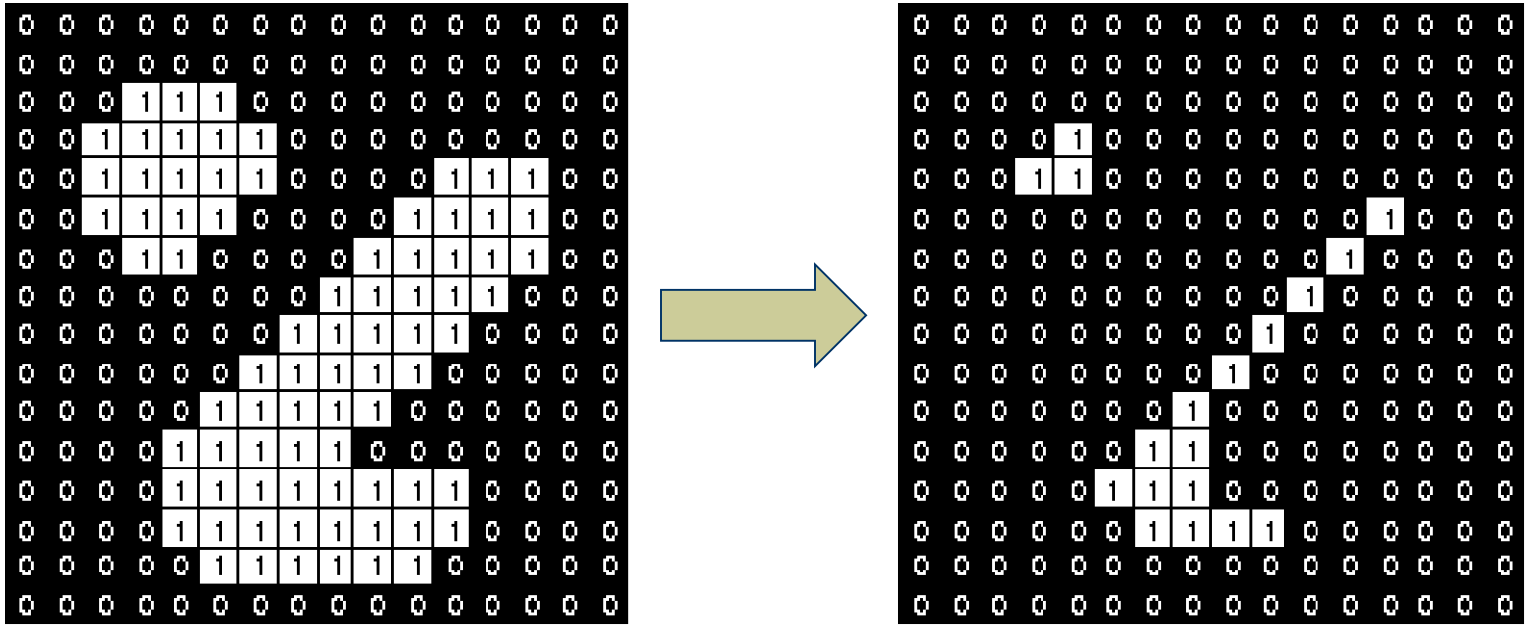


Structuring Element



Result

# Erosion: Example 2



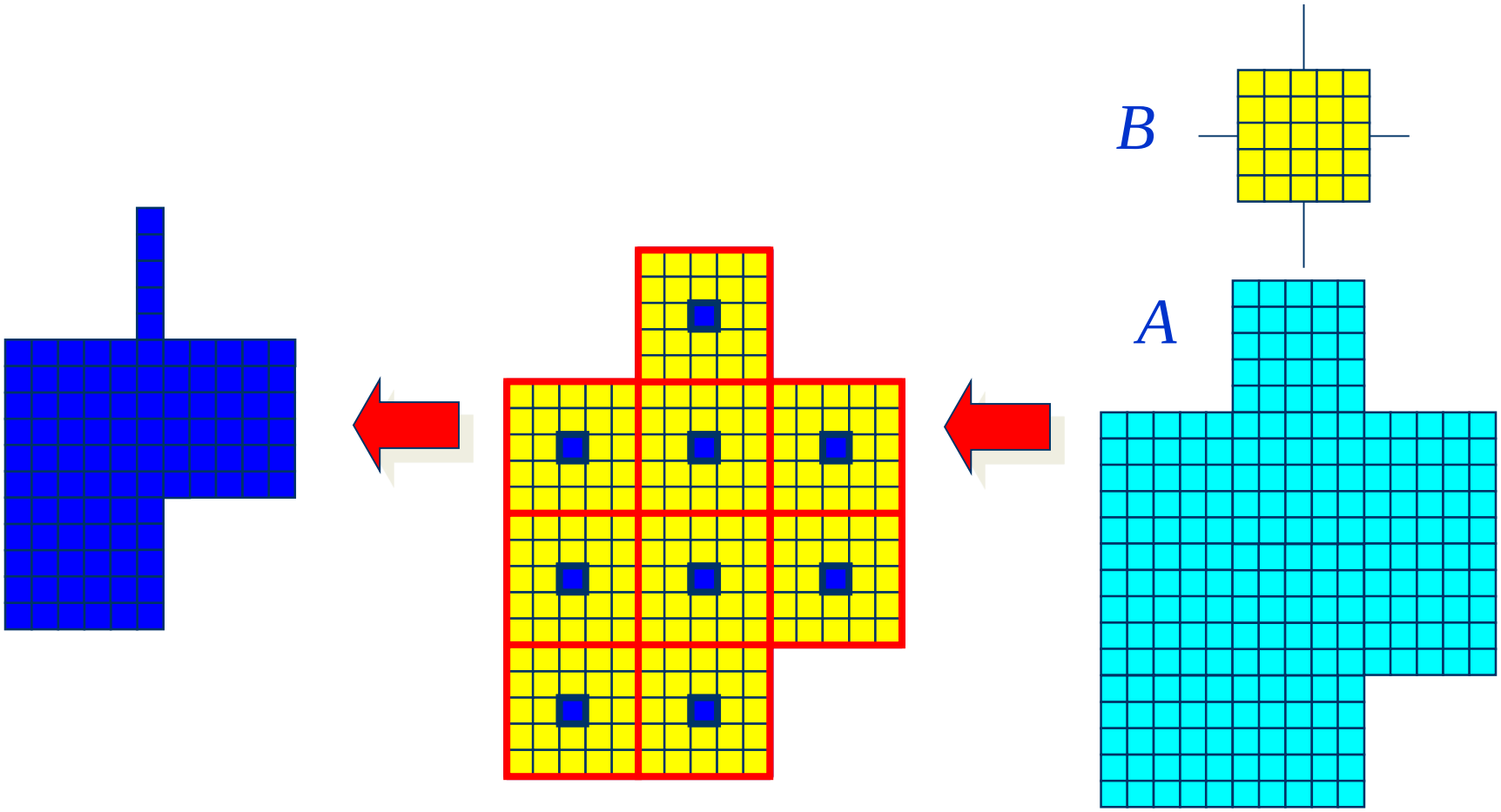
**Erosion with a structuring element of size 3x3**

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

Set of coordinate points =

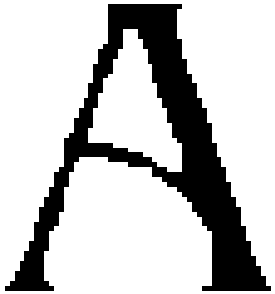
{ (-1, -1), (0, -1), (1, -1),
(-1, 0), (0, 0), (1, 0),
(-1, 1), (0, 1), (1, 1) }

# Erosion: Example 3

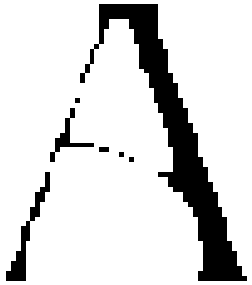


# Erosion: Example 4

Original image



Erosion by 3*3 square structuring element

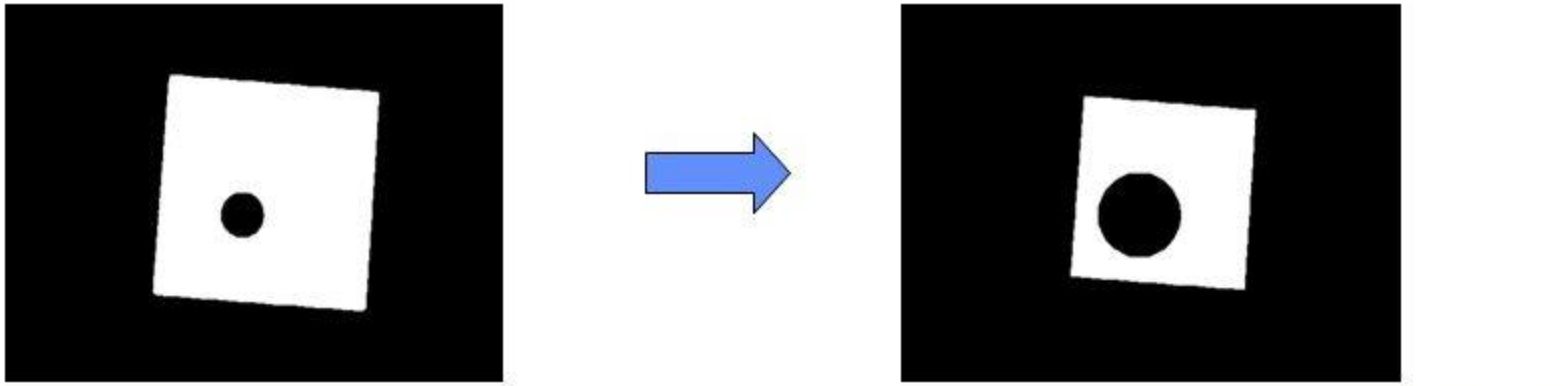


Erosion by 5*5 square structuring element

**Note:** In these examples a 1 refers to a black pixel!

# Erosion: Example 5

⇨ Example: Binary erosion



**Original thresholded image**

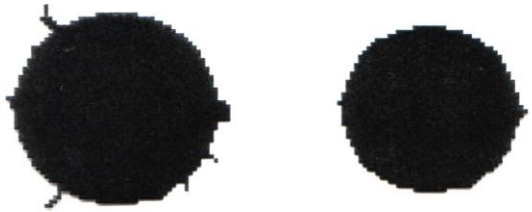
**Result of erosion four times with a disk shaped structuring element of 11 pixels in diameter**

- It shows that the hole in the middle of the image increases in size as the border shrinks.
- Erosion using a disk shaped structuring element will tend to round concave boundaries, but will preserve the shape of convex boundaries.

# Erosion: Example 6

Erosion can split apart joined objects

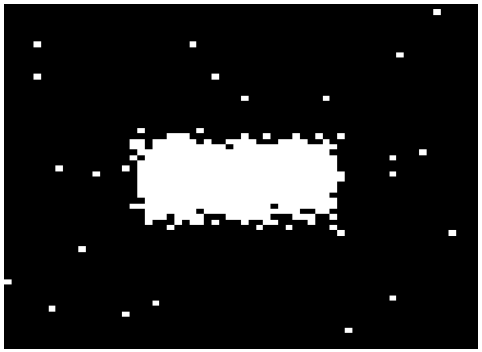
Erosion can strip away extrusions



**Watch out:** Erosion shrinks objects

# Typical use of Erosion

- Removes isolated noisy pixels.
- Smoothes object boundary(removes spiky edges).
- Removes the outer layer of object pixels:



- Object becomes slightly smaller.
- Sets contour pixels of object to background value

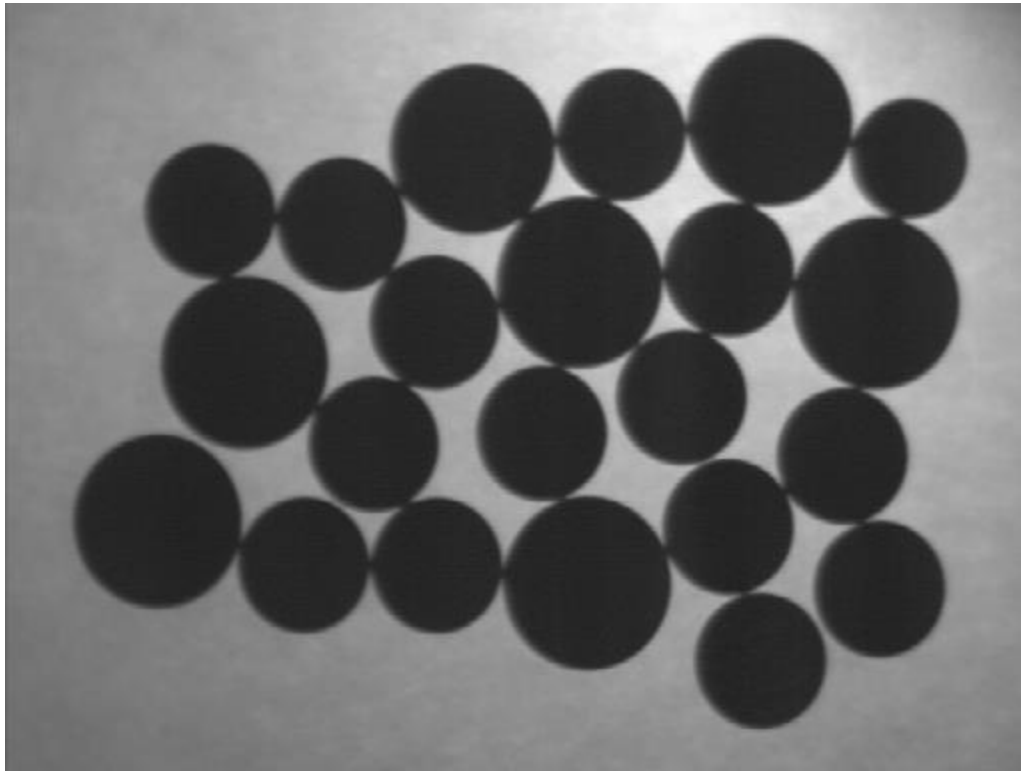
# Erosion

- Effects
  - Shrinks the size of foreground (1-valued) objects
  - Smoothes object boundaries
  - Removes small objects

- Rule for Erosion
  In a binary image, if any of the pixel (in the neighborhood defined by structuring element) is 0, then output is 0

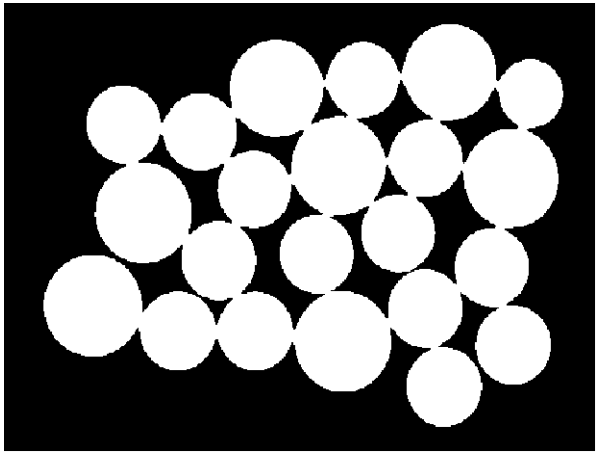
# Exercise

- Count the number of coins in the given image performing Erosion operation.

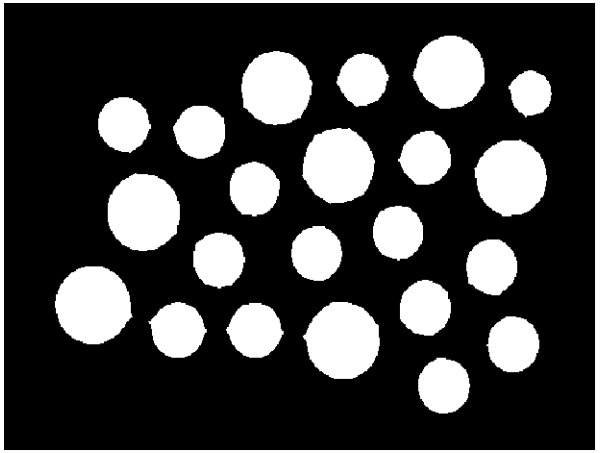


# Exercise: Solution

Binarize the image

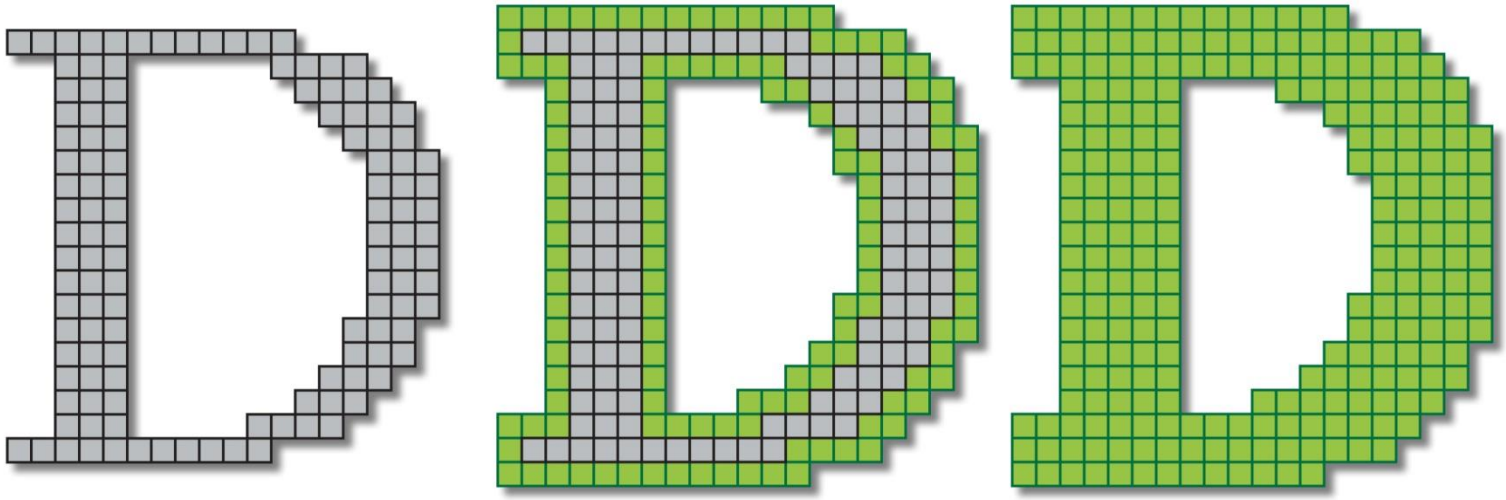


Perform Erosion



Use connected component labeling to count the number of coins

# Dilation

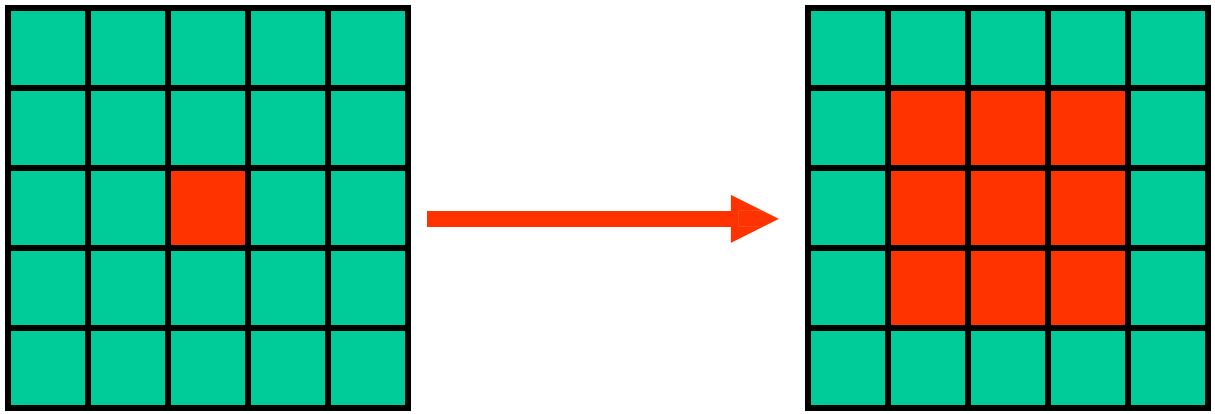


**Grows the object**

Enlarges foreground, shrinks background

# Dilation

- Dilation is used for expanding an element A by using structuring element B.
- The dilation operator takes two pieces of data as input
    1. A binary image, which is to be dilated
    2. A structuring element (or kernel), which determines the behavior of the morphological operation



Original Image

Dilated Image

# Dilation

Definition 1:

**Hit:** Any "***ON"*** *pixel* in the structuring element covers an "***ON"*** *pixel* in the image

- Does the structuring element **hit the set?**

$$A \oplus\!\!\!\!\ominus B = \left\{ z \,|\, (\hat{B})_z \cap A \neq \Phi \right\}$$

- Dilation of a set A by structuring element B:
  - all z in A such that B hits A when origin of B=z
  - such that overlap A by at least one element

# Dilation

**Definition 2:**

Dilation of image *f* by structuring element *s* is given by

$f \oplus s$

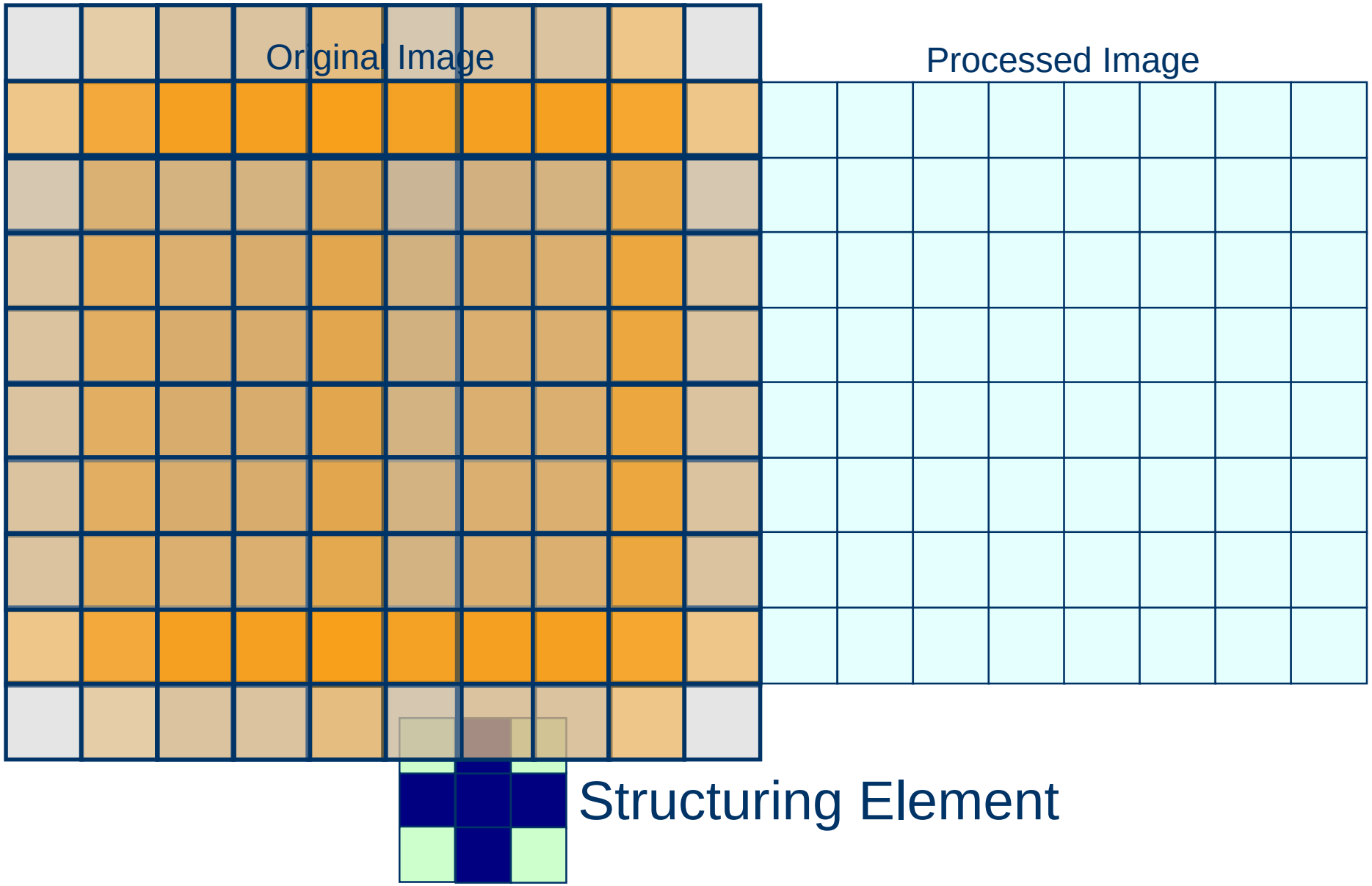
The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

$$g(x,y) = \begin{cases} 1 \text{ if } s \text{ hits } f \\ 0 \text{ otherwise} \end{cases}$$

# Dilation – How to compute

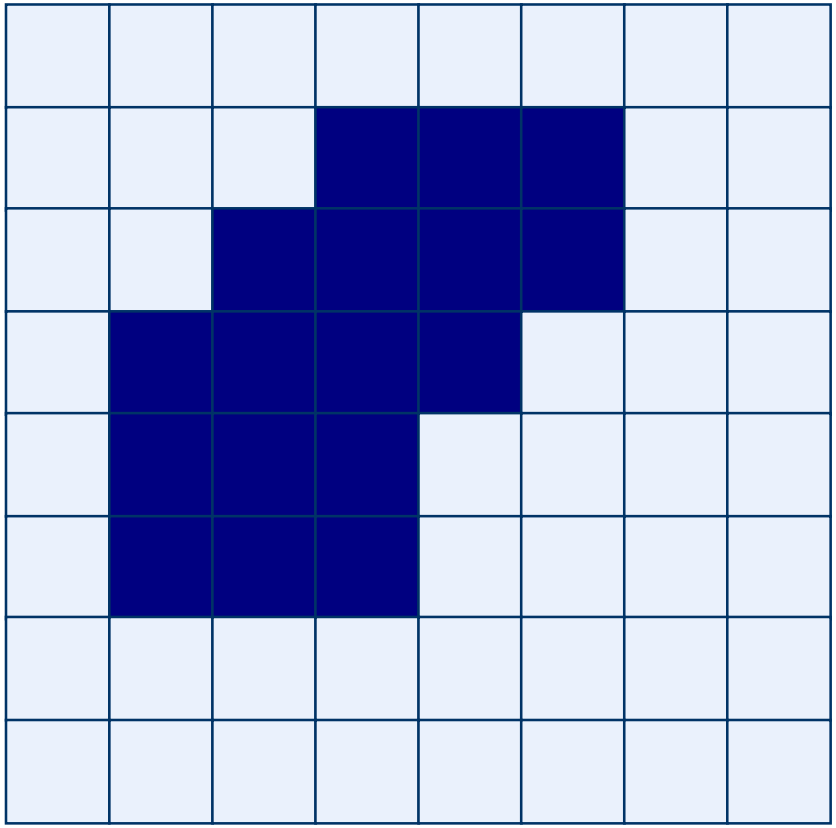
- For each background pixel (which we will call the *input pixel*)
  - Superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position
  - If *at least one* pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value
  - If all the corresponding pixels in the image are background, however, the input pixel is left at the background value

# Dilation: Example

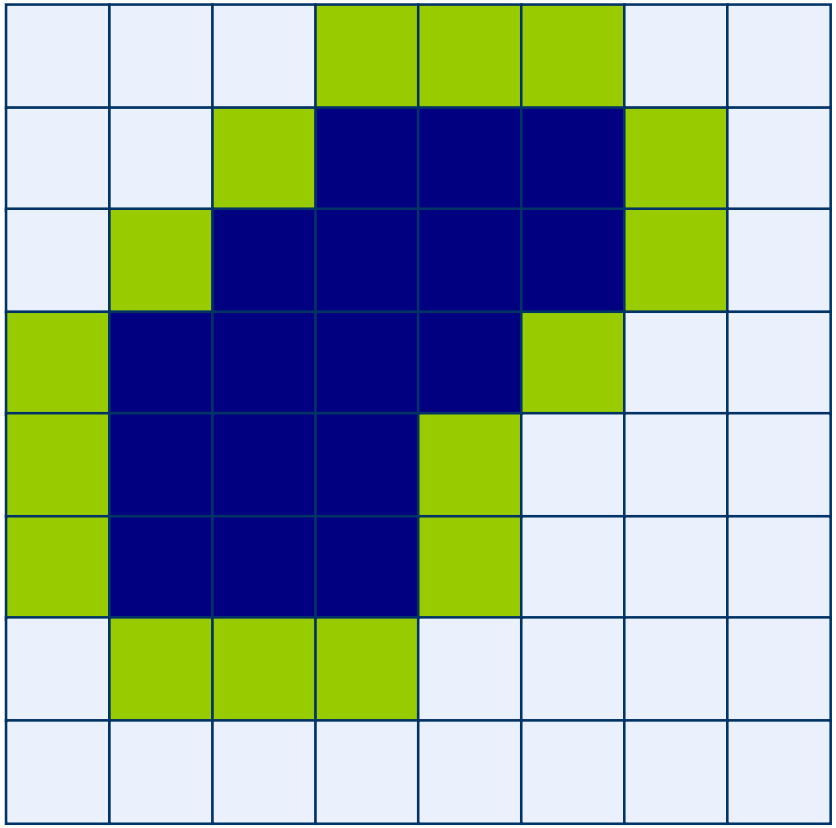


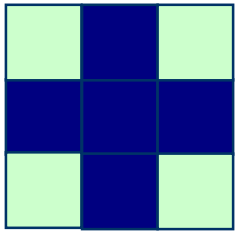
# Dilation: Example 1

Original Image



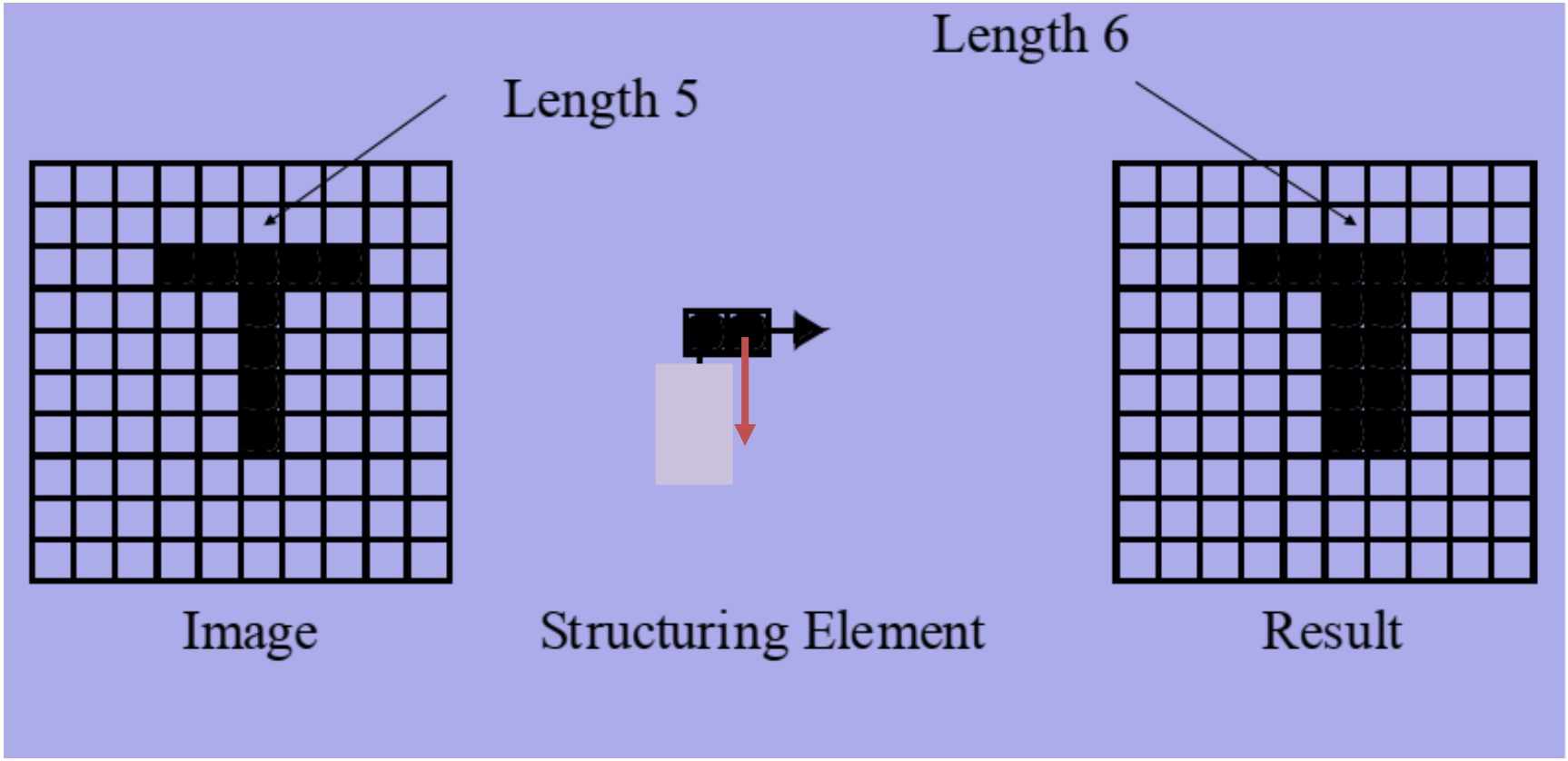
Processed Image With Dilated Pixels



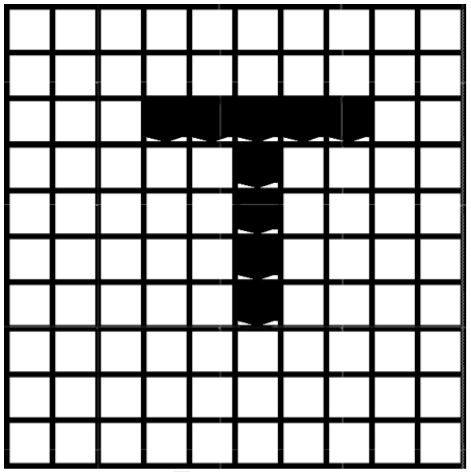


Structuring Element

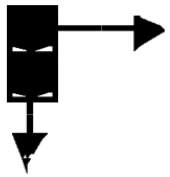
# Structuring Element in Dilation Example



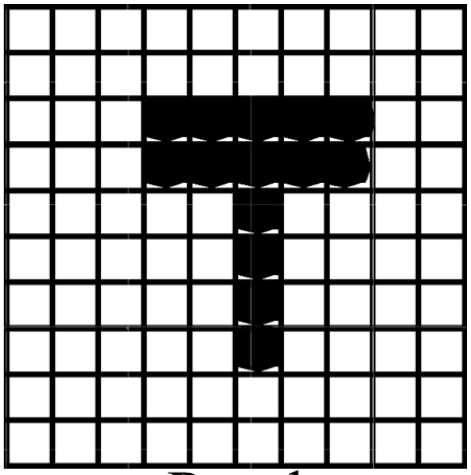
# Structuring Element in Dilation Example



Image

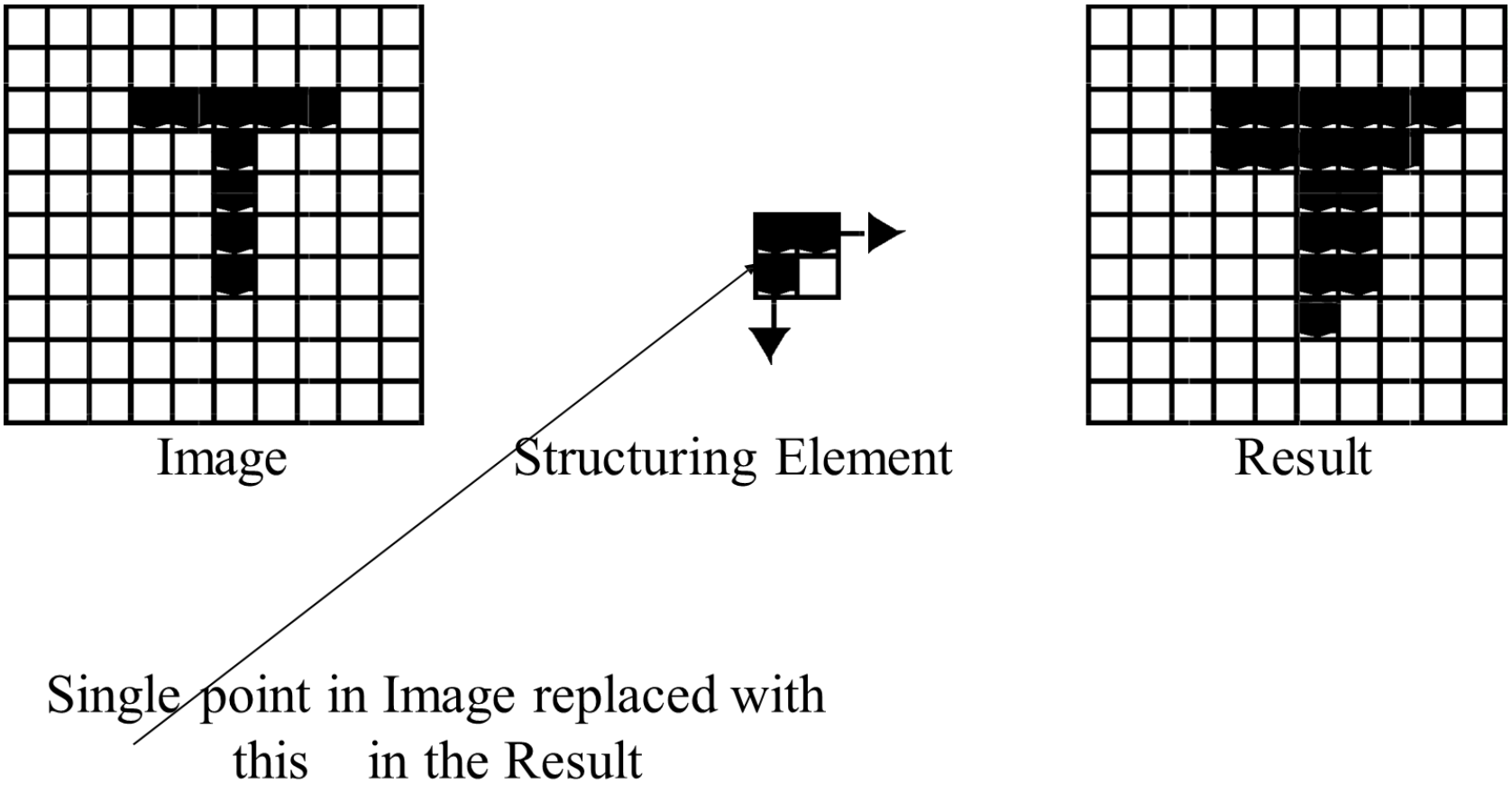


Structuring Element

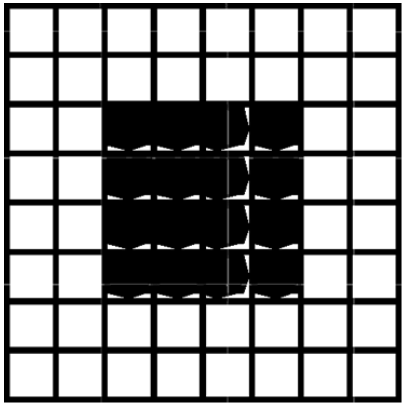


Result

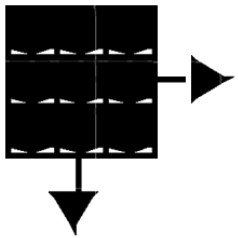
# Structuring Element in Dilation Example



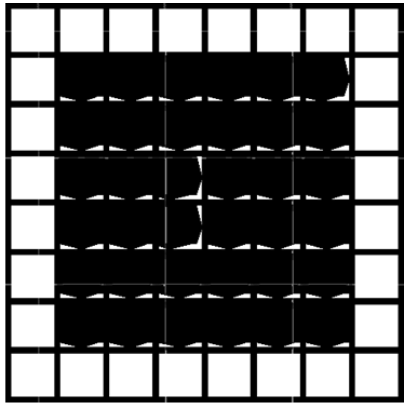
# Structuring Element in Dilation Example



Image



Structuring Element



Result

# Dilation

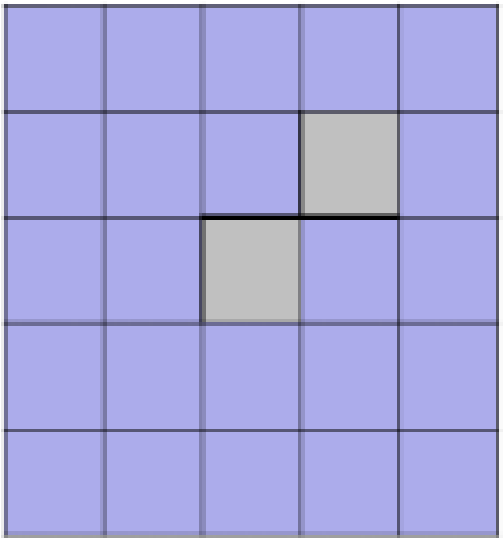
Question: Suppose that the structuring element is a 3x3 square with the origin at its center evaluate the new image

$B =$

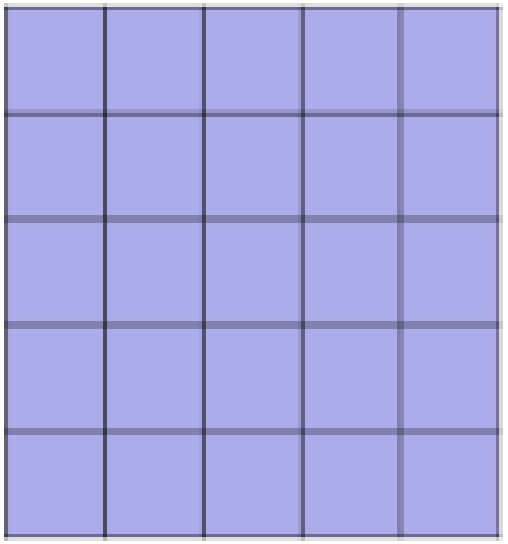
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

```
{ (-1,-1), (0,-1), (1,-1),
  (-1,0),  (0,0),  (1,0),
  (-1,1),  (0,1),  (1,1) }
```

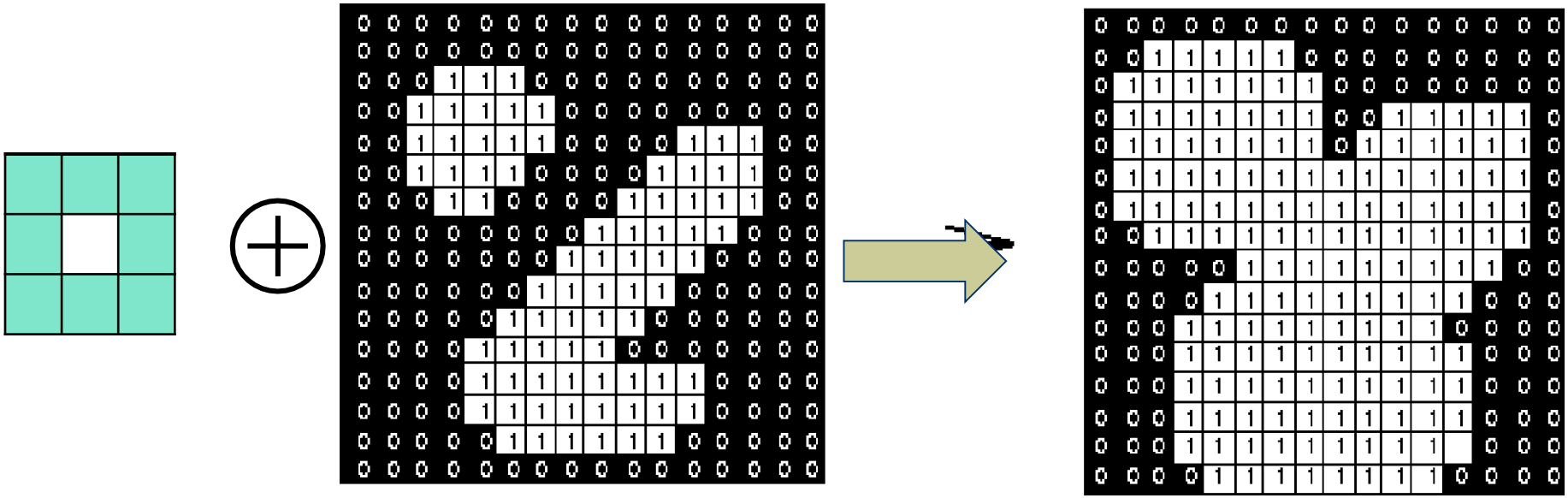
$A =$



$A \oplus B$

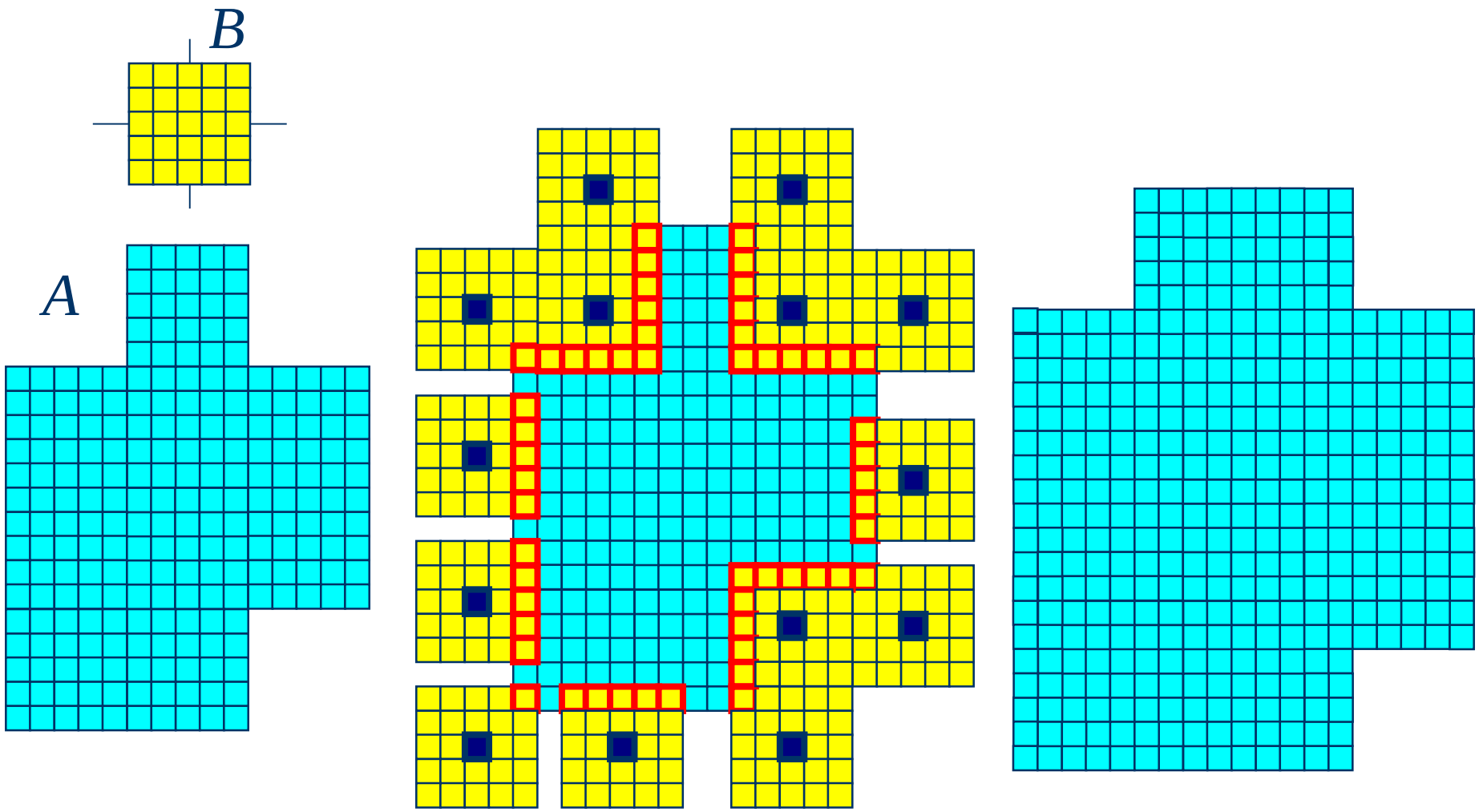


# Dilation: Example 2



*Effect of dilation using a 3 ×3 square structuring element*

# Dilation: Example 3



# Dilation: Example 4

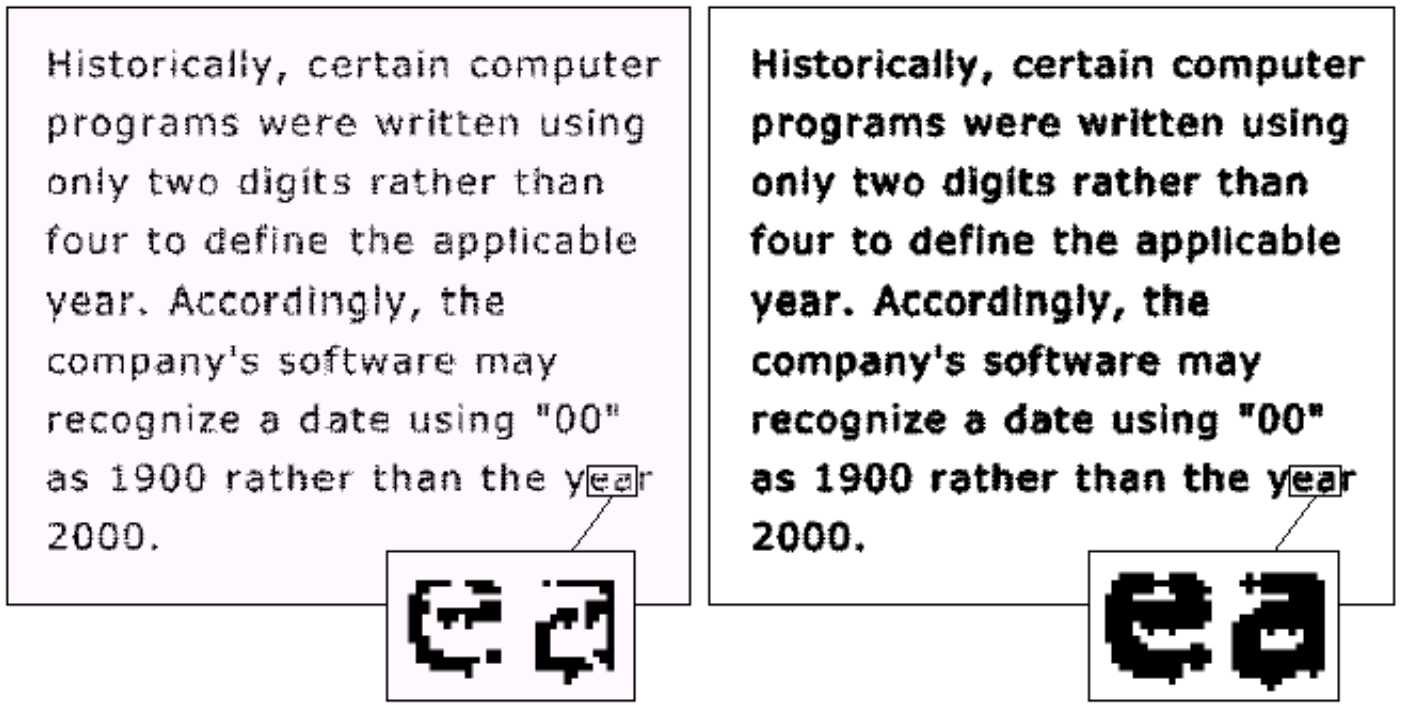
Original image

Dilation by 3*3 square structuring element

Dilation by 5*5 square structuring element

**Note:** In these examples a 1 refers to a black pixel!

# Dilation : Bridging gaps

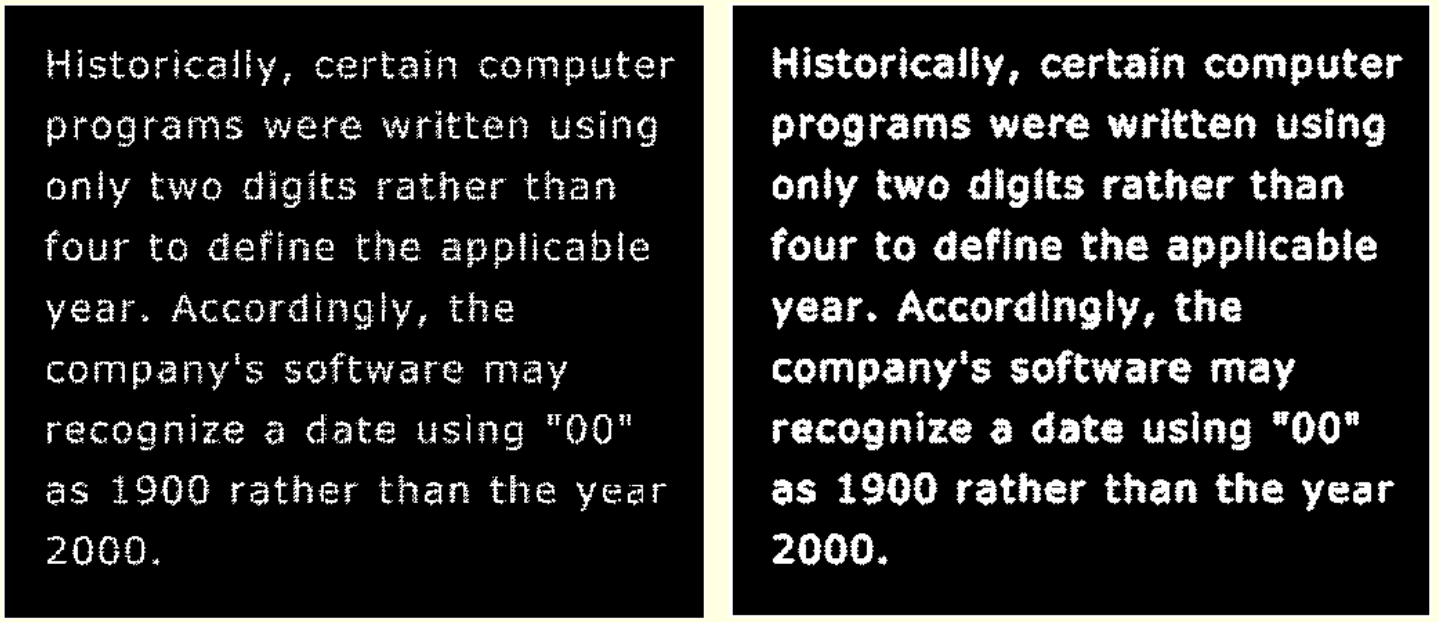



a c
b

**FIGURE 9.5**
(a) Sample text of poor resolution with broken characters (magnified view).
(b) Structuring element.
(c) Dilation of (a) by (b). Broken segments were joined.

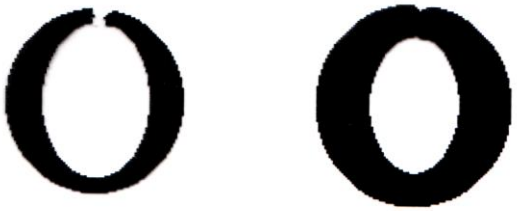
| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

# Dilation: Example 5



# Dilation: Example 6

Dilation can repair breaks



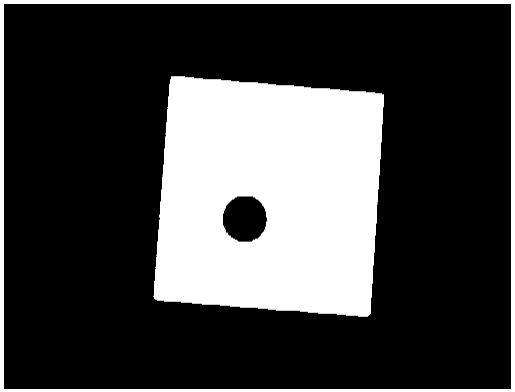
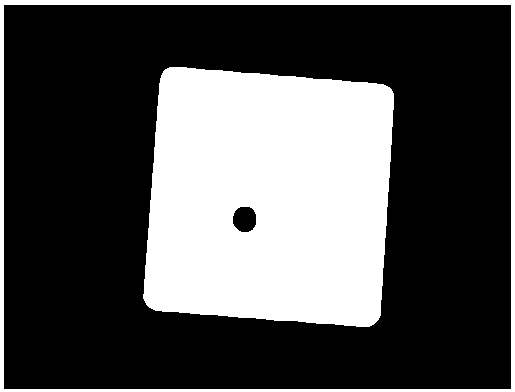
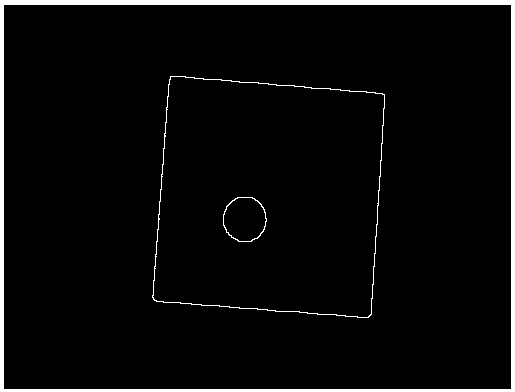
Dilation can repair intrusions

**Watch out:** Dilation enlarges objects

# Dilation: Example 7

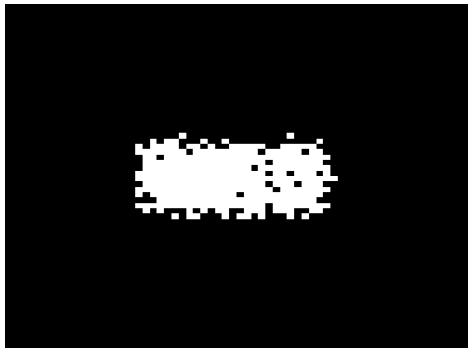
- Edge Detection

1. Dilate input image
2. Subtract input image from dilated image
3. Edges remain!

# Typical use of Dilation

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



# Dilation

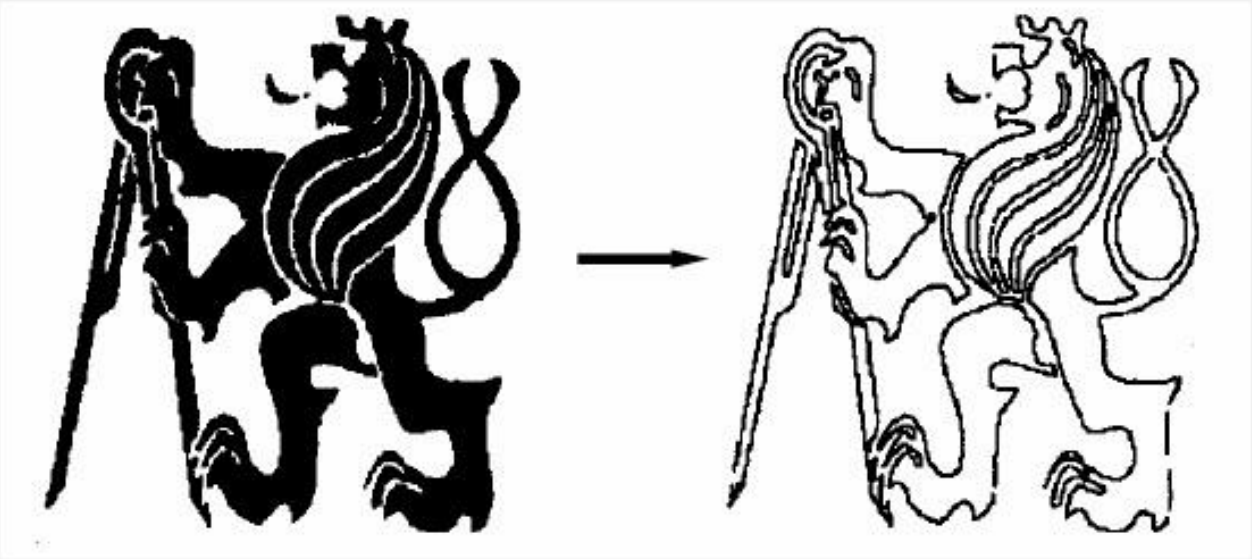
- Effects
  - Expands the size of foreground(1-valued) objects
  - Smoothes object boundaries
  - Closes holes and gaps

- Rule for Dilation
  In a binary image, if any of the pixel (in the neighborhood defined by structuring element) is 1, then output is 1

# More Applications of Erosion and Dilation

# Boundary Extraction

Contours can be extracted by subtraction of the eroded image from the original.

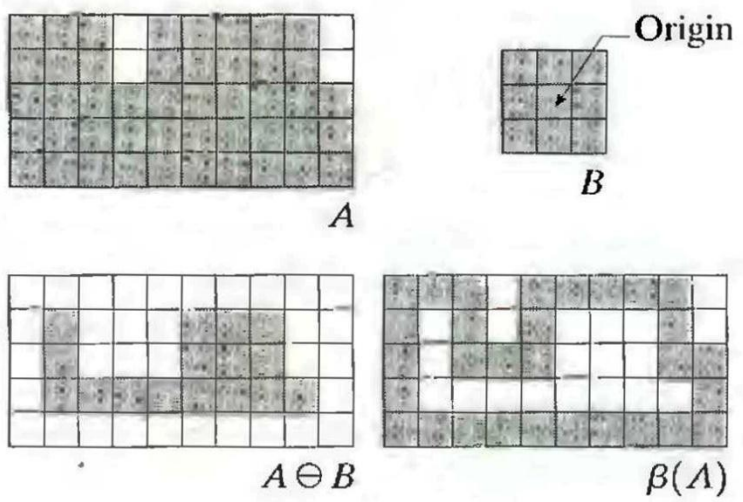


$$\beta(A) = A - (A \ominus B)$$

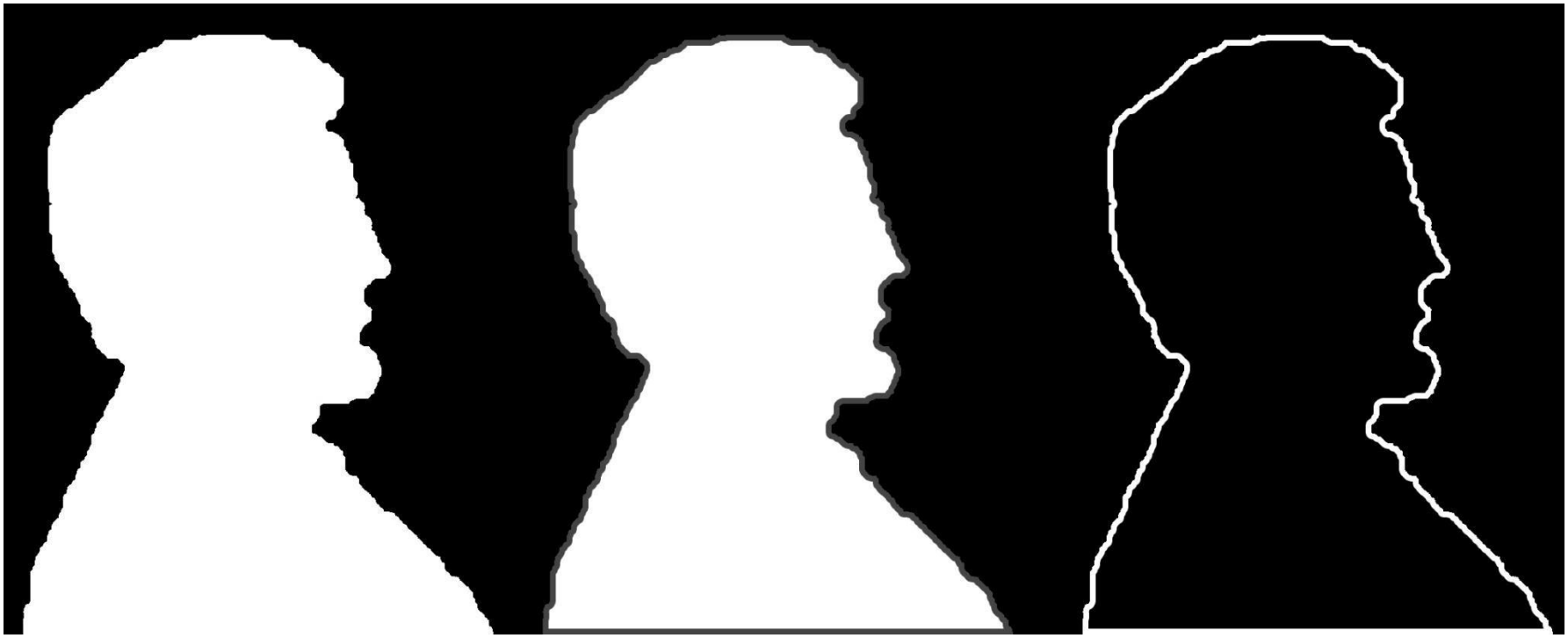
# Boundary Extraction

- First, erode A by B, then make set difference between A and the erosion
- The thickness of the contour depends on the size of constructing object – B

$$\beta(A) = A - (A \ominus B)$$



# Boundary Extraction



$$\beta(A) = A - (A \ominus B)$$

# Edge detection

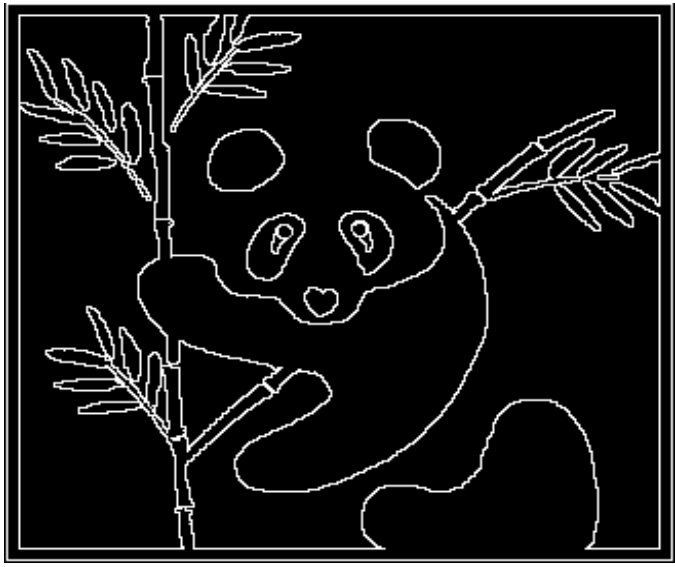
original

$A$

Dilate

$A \oplus B$

Dilate - original



$A \oplus B - A$

# Duality relationship between Dilation and Erosion

- When one operation is the dual of the other, it means that one can be written in terms of the other. This does not, however, mean that they are opposites.

- Dilation and erosion are duals of each other:

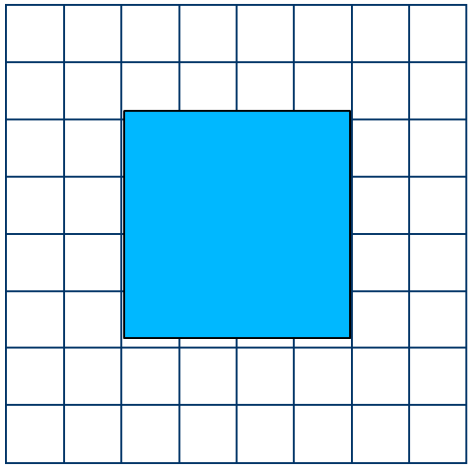
$$(A \ominus B)^c = A^c \oplus \hat{B}$$

- For a symmetric structuring element:

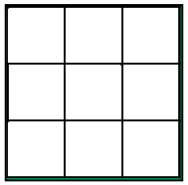
$$(A \oplus B)^c = A^c \ominus \hat{B}$$

It means that we can obtain erosion of an image A by B simply by dilating its background (i.e. $A^c$) with the same structuring element and complementing the result.

# Duality of Dilation and Erosion Example



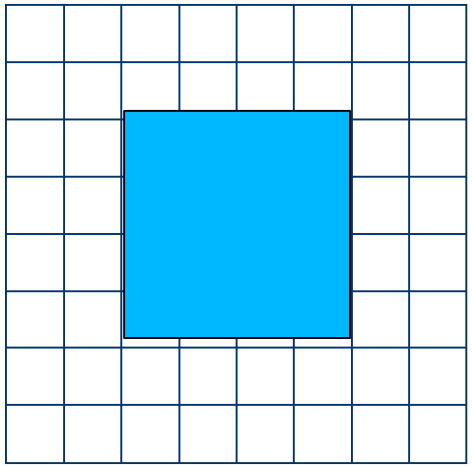
$A$



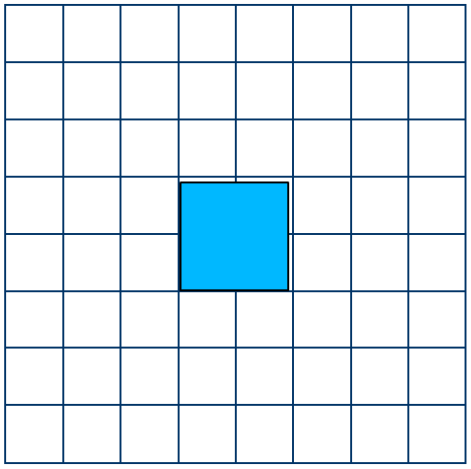
$B$

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

# Duality of Dilation and Erosion Example



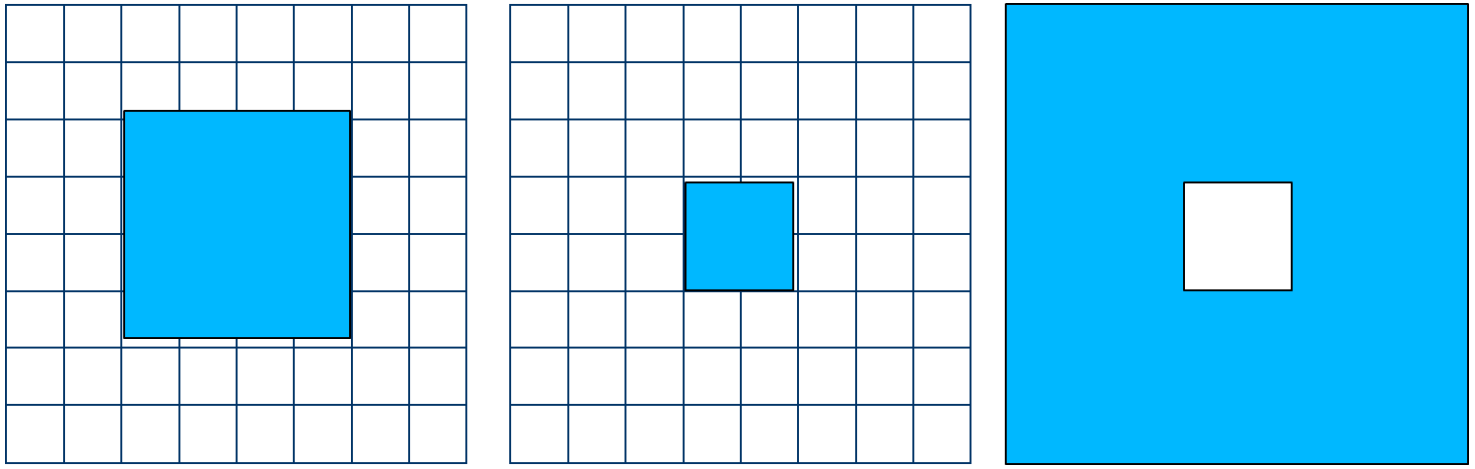
$A$



$(A \ominus B)$

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

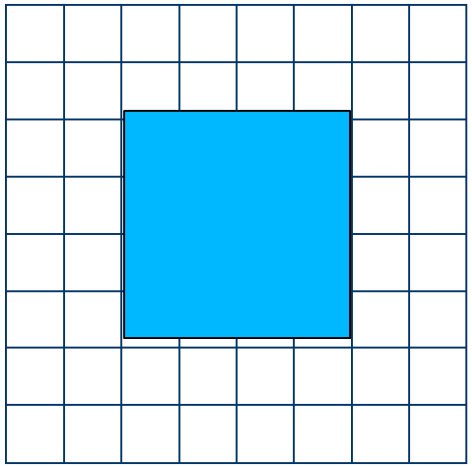
# Duality of Dilation and Erosion Example



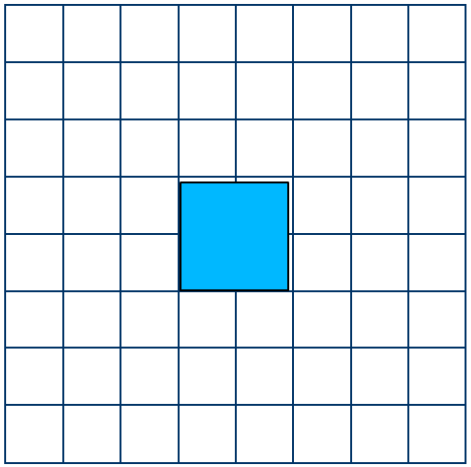
$A$ $\quad (A \ominus B)$ $\quad (A \ominus B)^c$

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

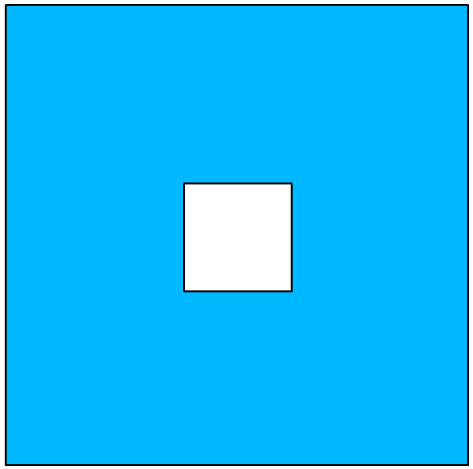
# Duality of Dilation and Erosion Example



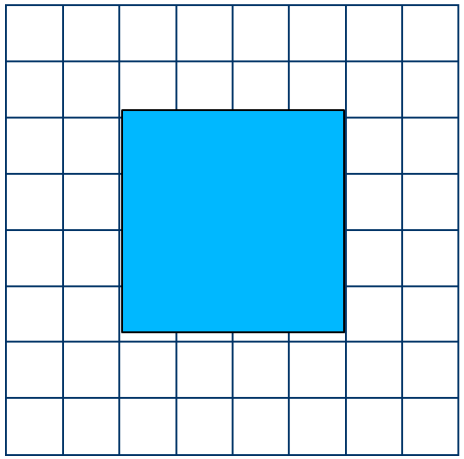
$A$



$(A \ominus B)$



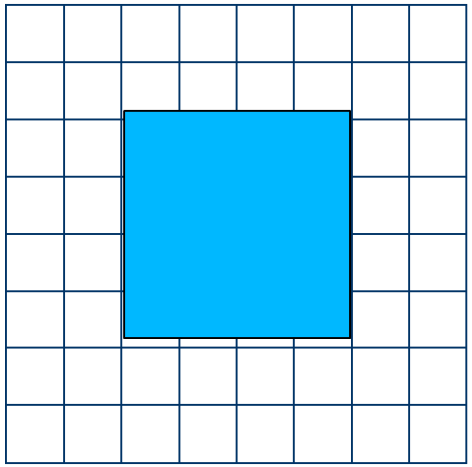
$(A \ominus B)^c$



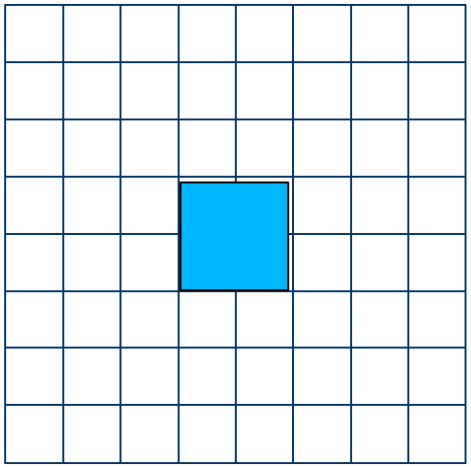
$A$

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

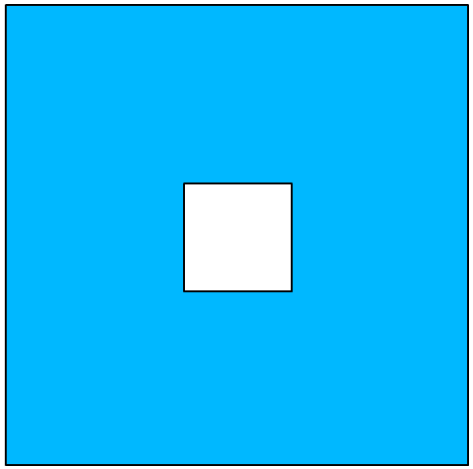
# Duality of Dilation and Erosion Example



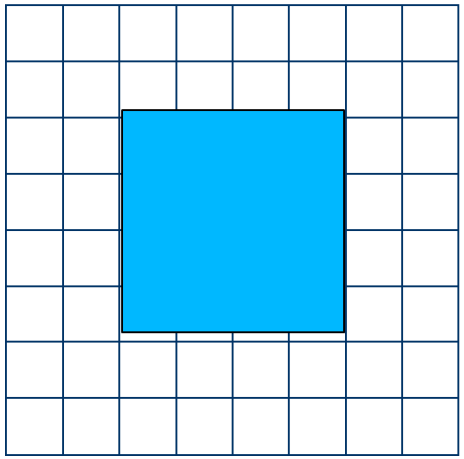
$A$



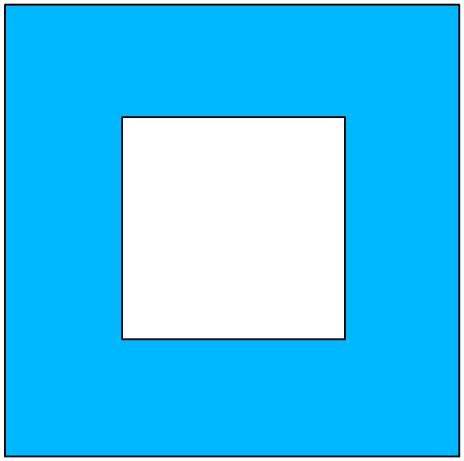
$(A \ominus B)$



$(A \ominus B)^c$

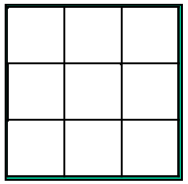


$A$



$A^c$

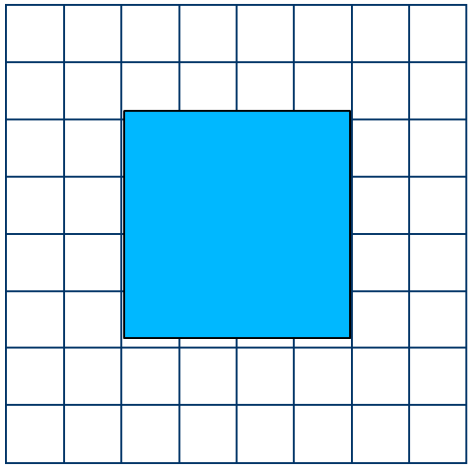
$$(A \ominus B)^c = A^c \oplus \hat{B}$$



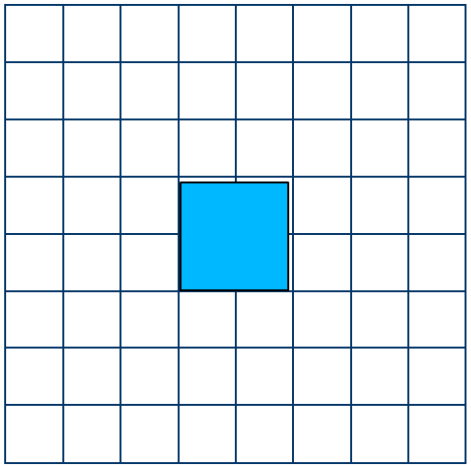
$B$

# Duality of Dilation and Erosion Example

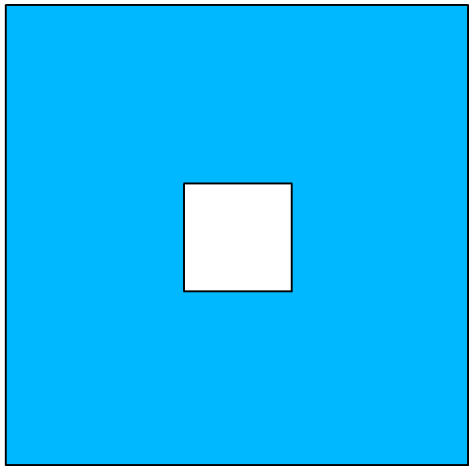
$$(A \ominus B)^c = A^c \oplus \hat{B}$$



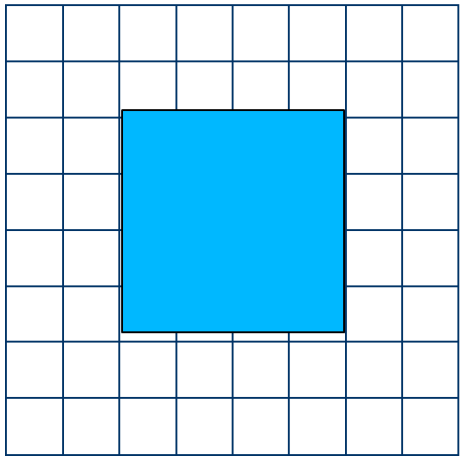
$A$



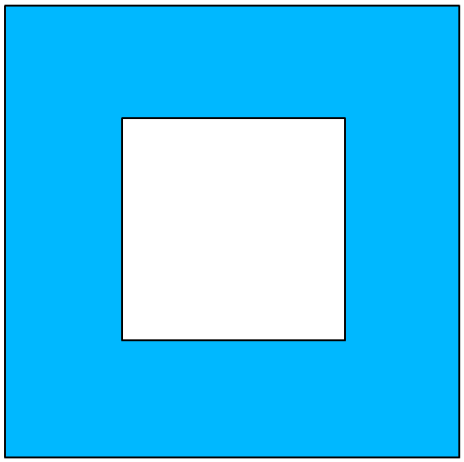
$(A \ominus B)$



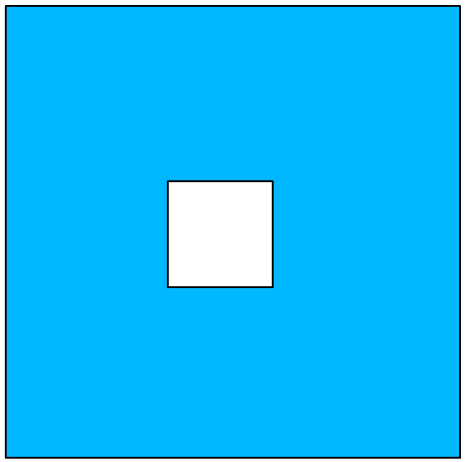
$(A \ominus B)^c$



$A$



$A^c$

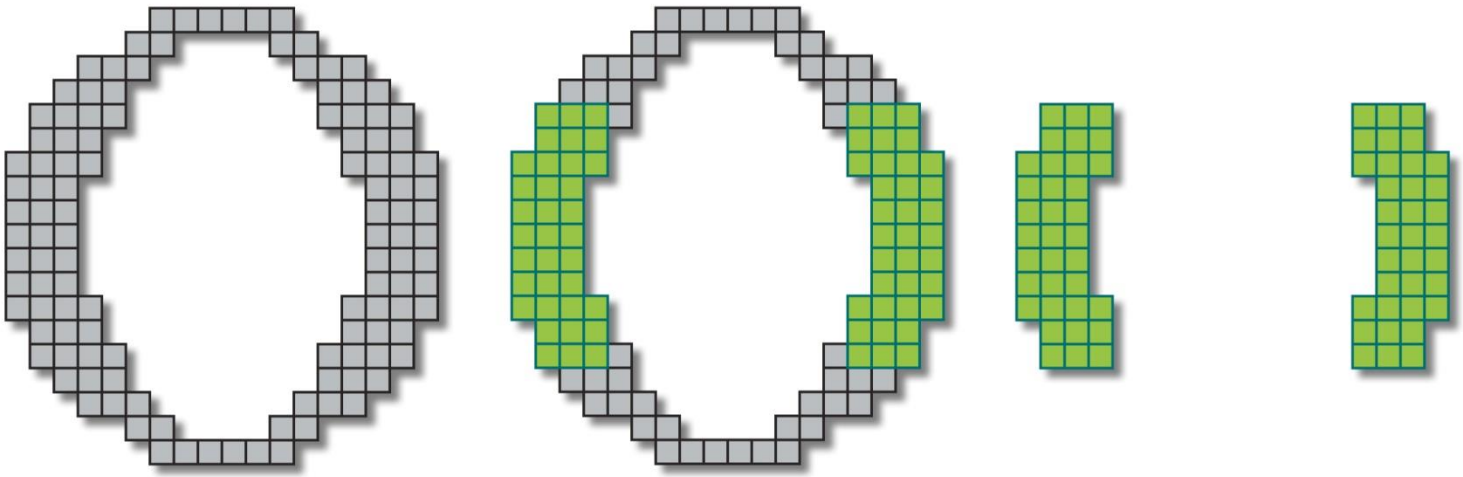


$A^c \oplus \hat{B}$

# Compound Operations

- More interesting morphological operations can be performed by performing combinations of erosions and dilations
- The most widely used of these *compound operations* are:
  - Opening
  - Closing

# Opening

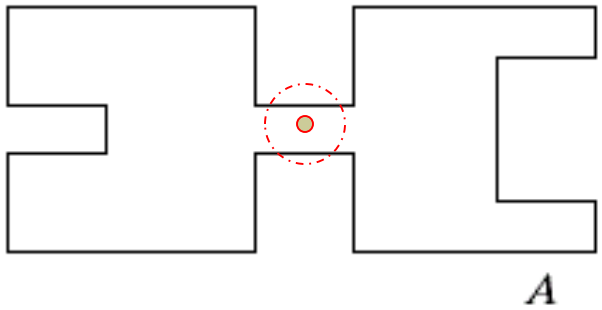


**Erosion followed by a dilation**

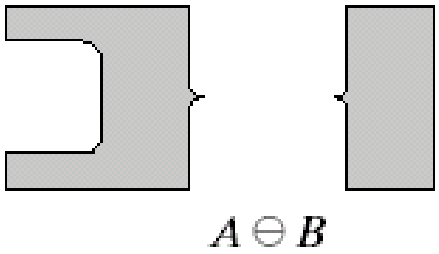
# Opening

The opening of image *A* by structuring element *B*, denoted by *A*○B is simply an erosion followed by a dilation

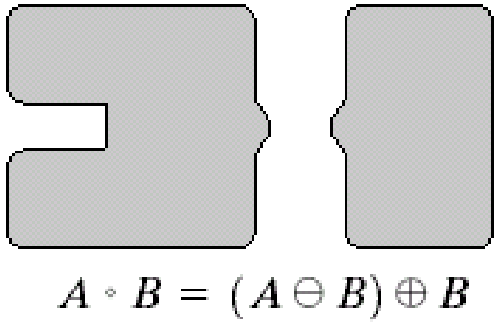
$$A \circ B = (A \ominus B) \oplus B$$




Original shape



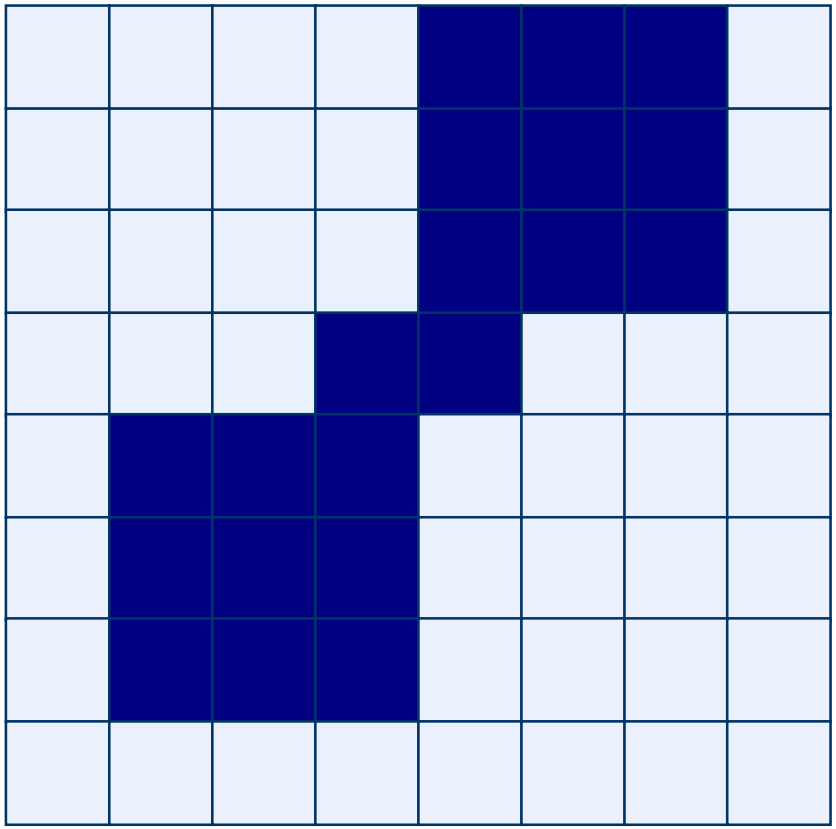

After erosion

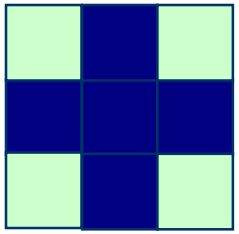



After dilation (opening)

# Opening: Example

Original Image

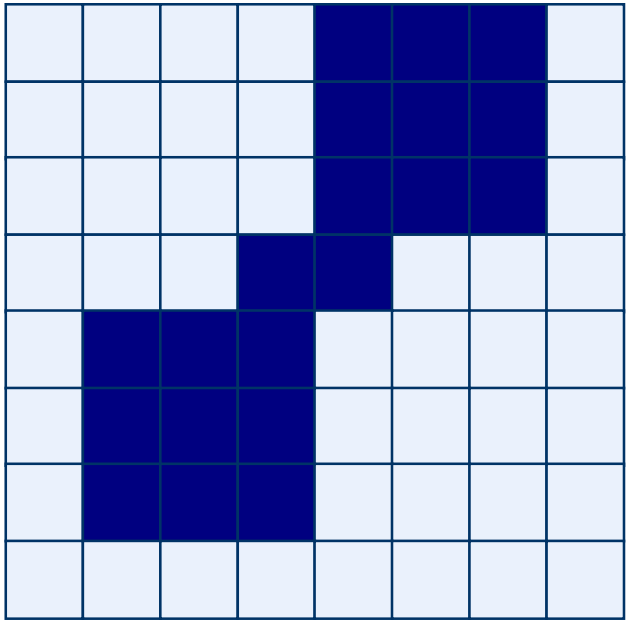




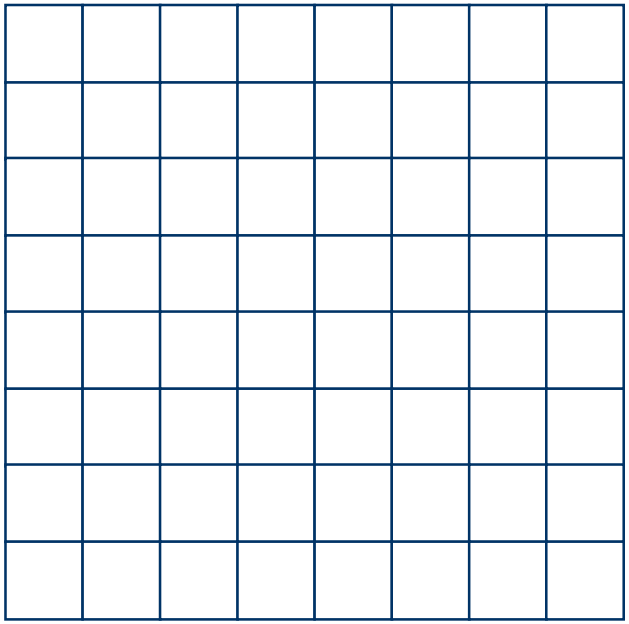
Structuring Element

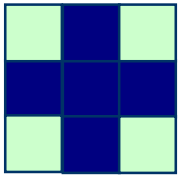
# Opening: Example

Original Image



Processed Image

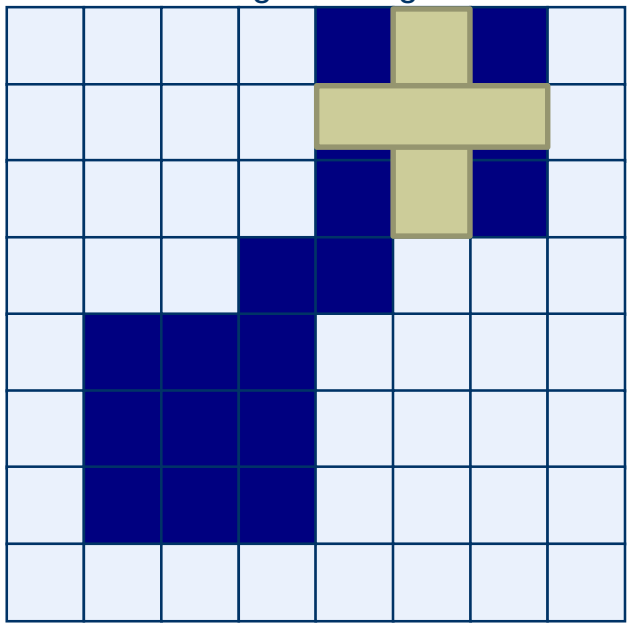




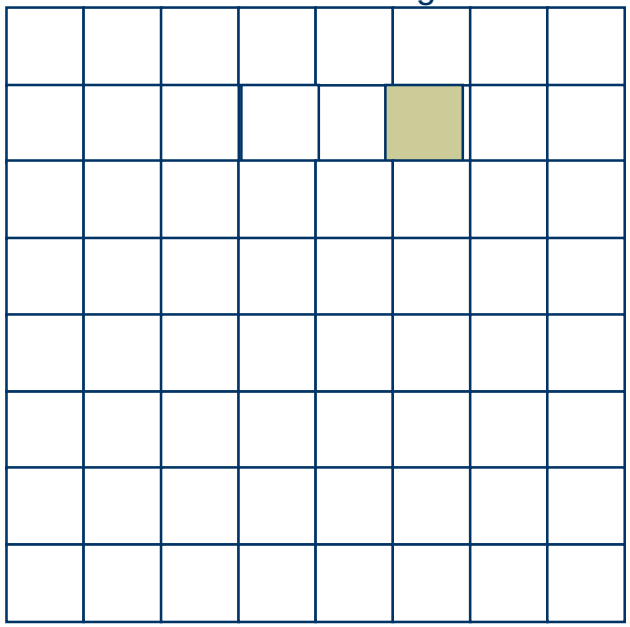
Structuring Element

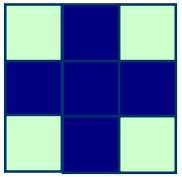
# Opening: Example (performing erosion)

Original Image



Processed Image

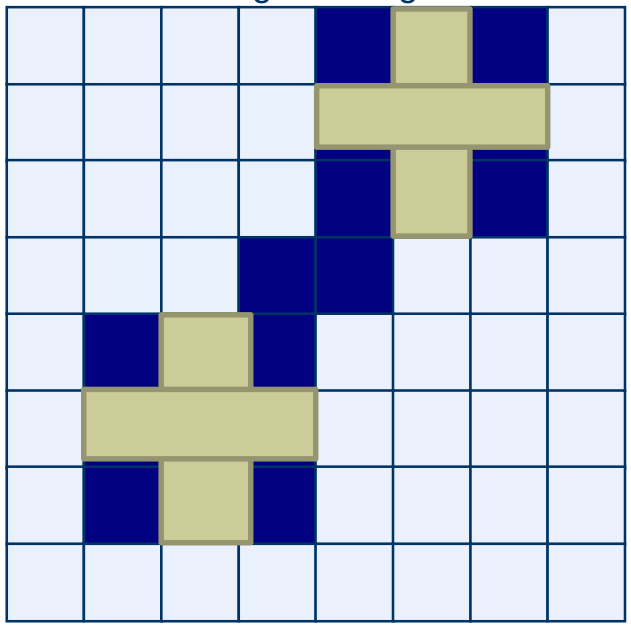




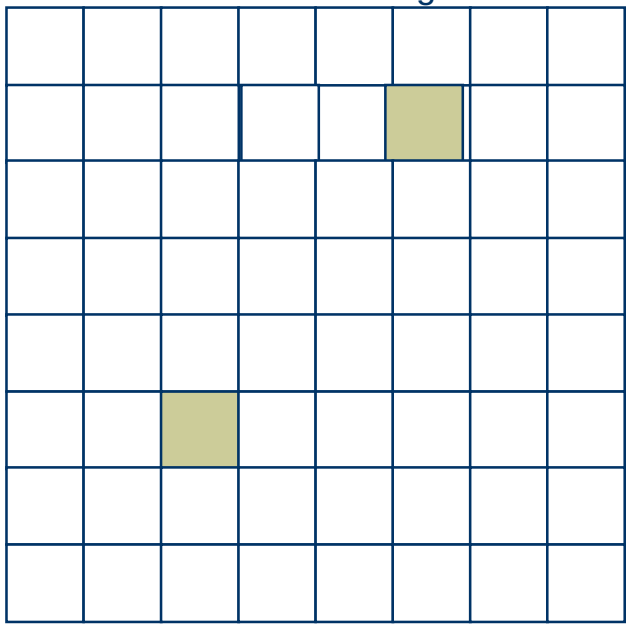
Structuring Element

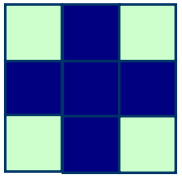
# Opening: Example (performing erosion)

Original Image



Processed Image

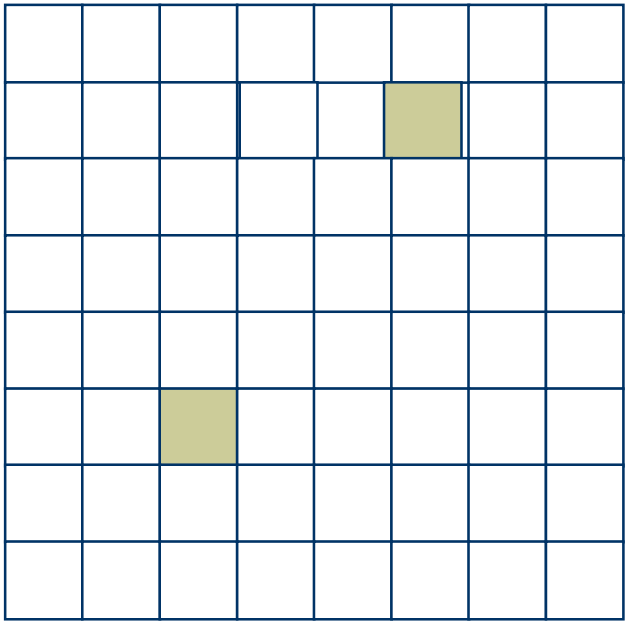




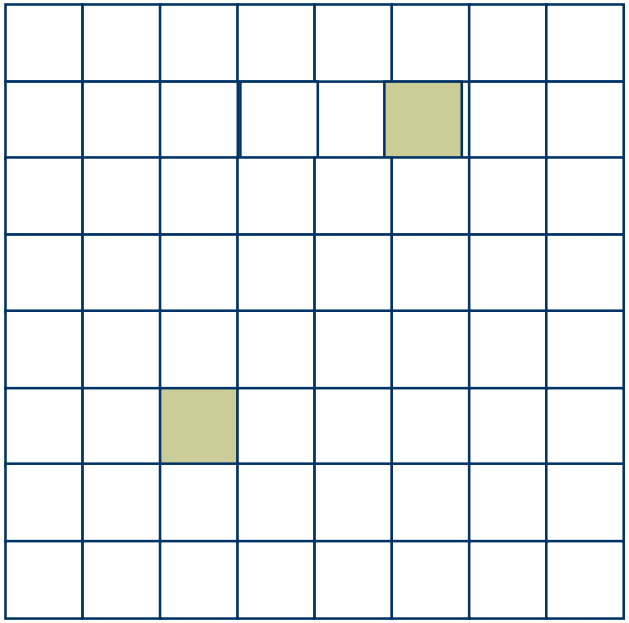
Structuring Element

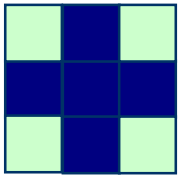
# Opening: Example (performing dilation to errored image)

Original Image



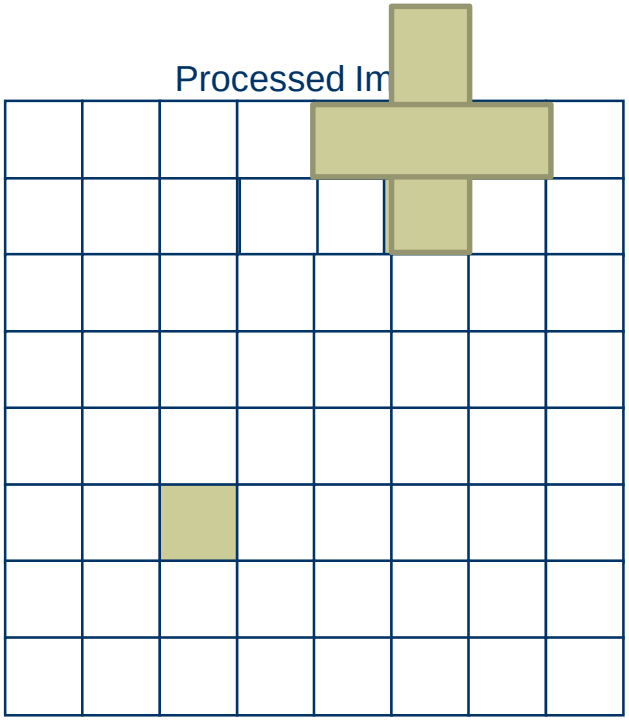
Processed Image

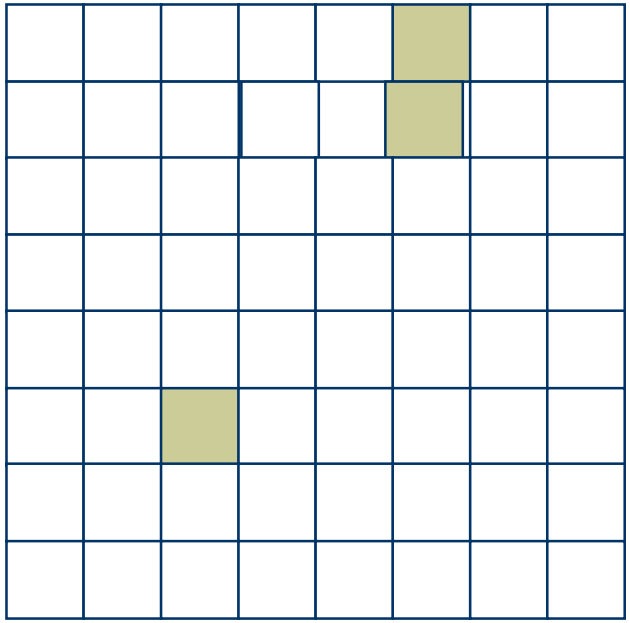


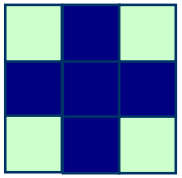


Structuring Element

# Opening: Example (performing dilation to errored image)



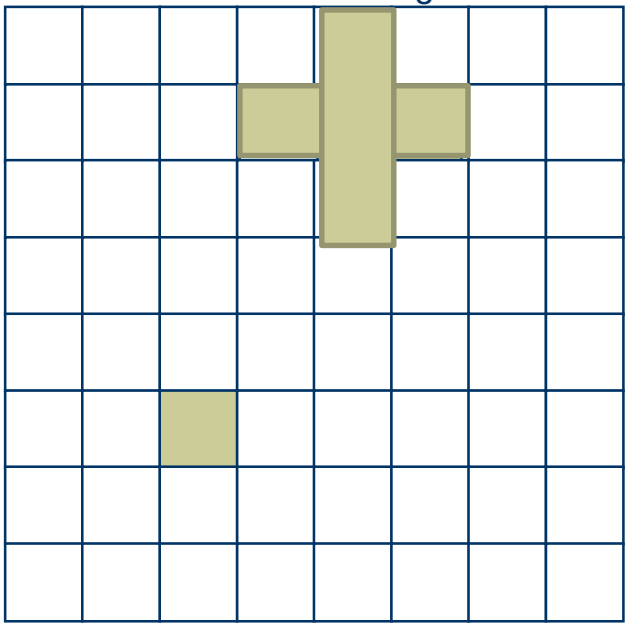



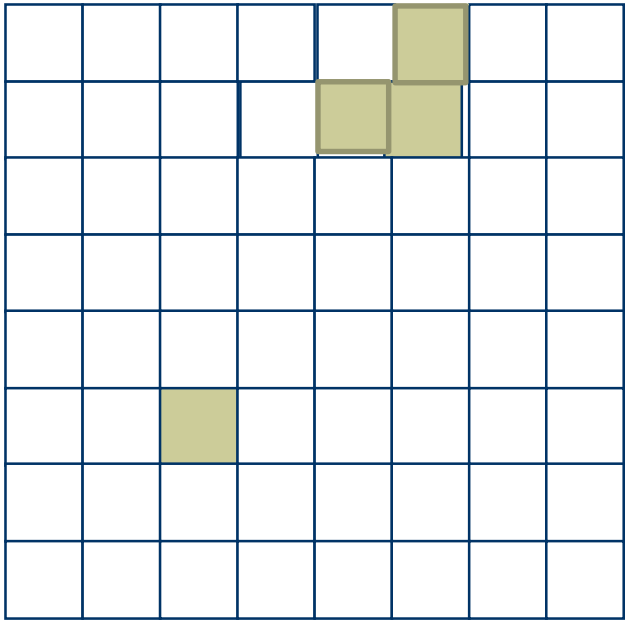


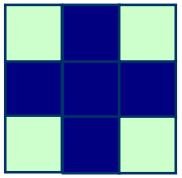
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



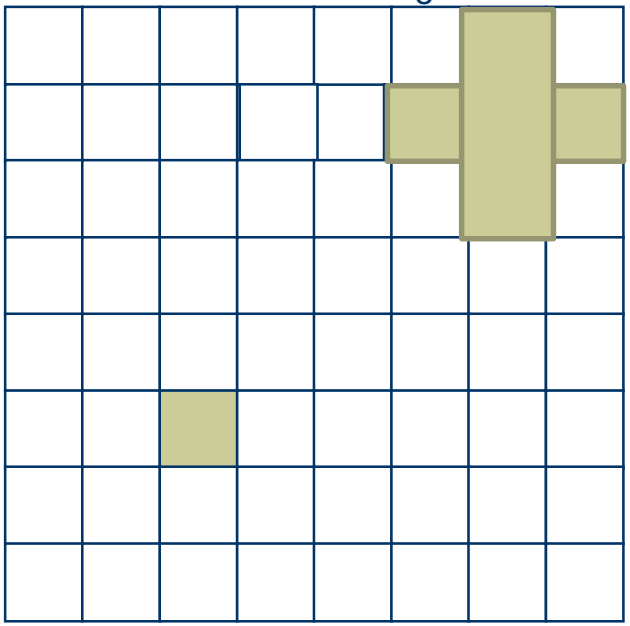


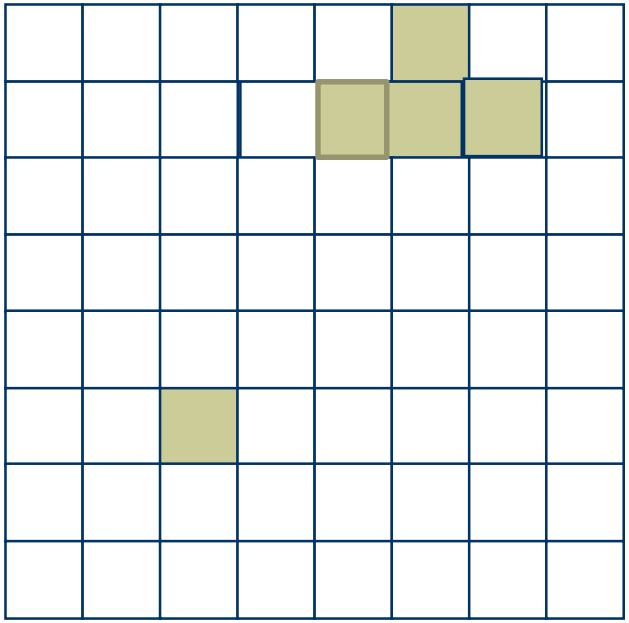


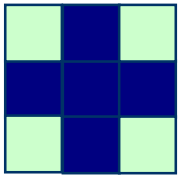
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



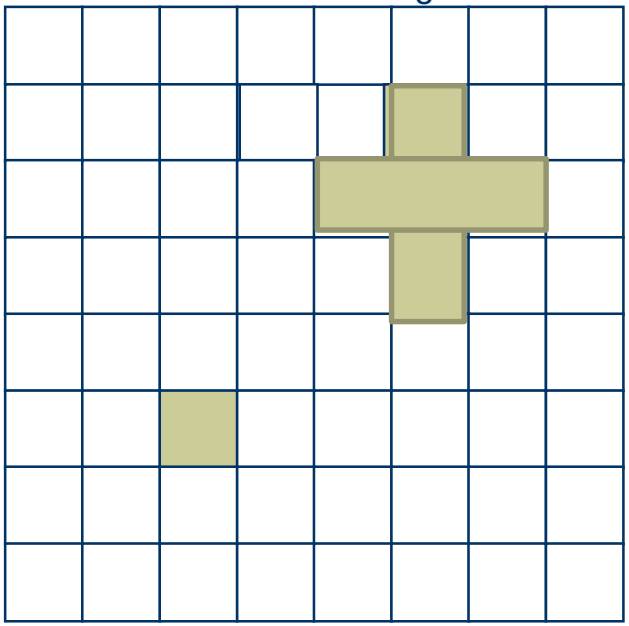


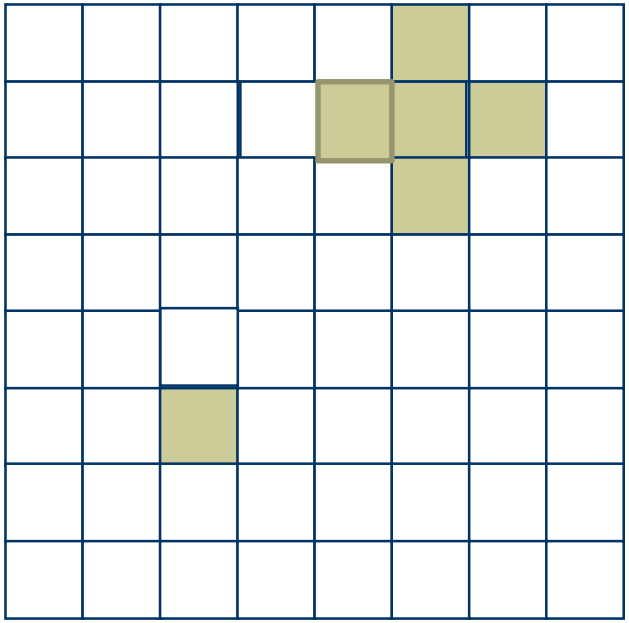


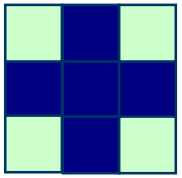
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



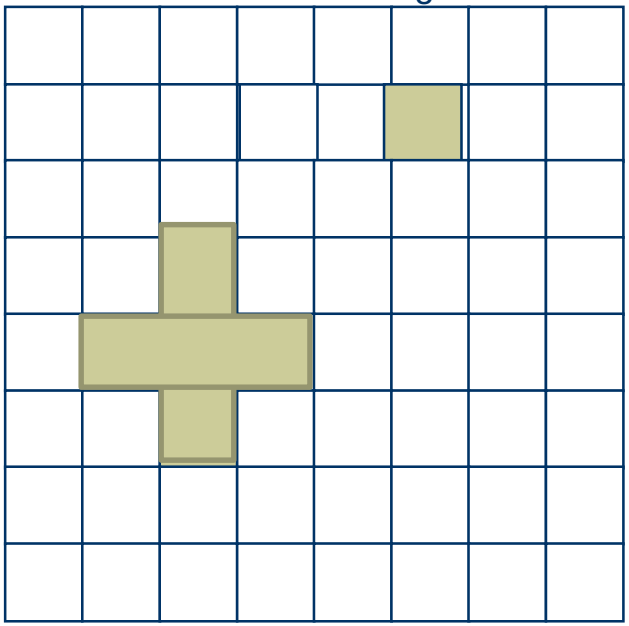


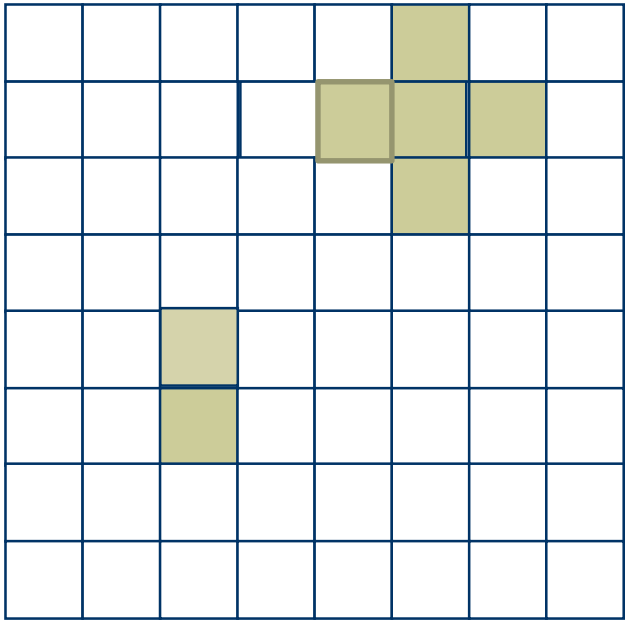


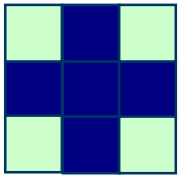
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



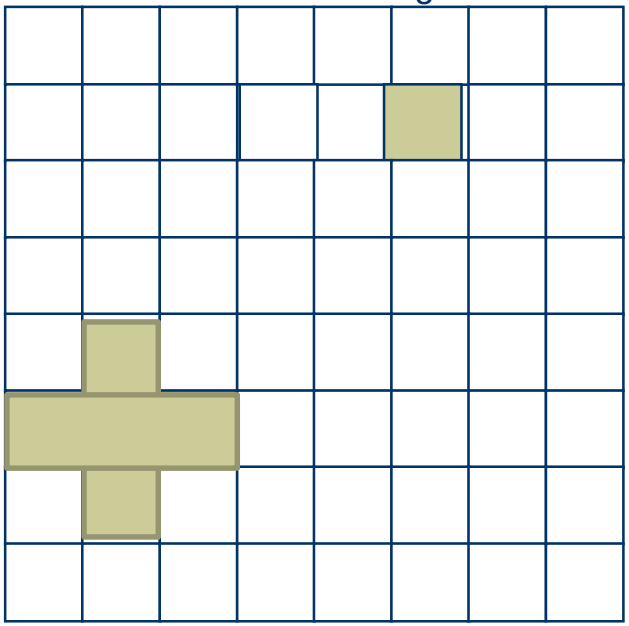


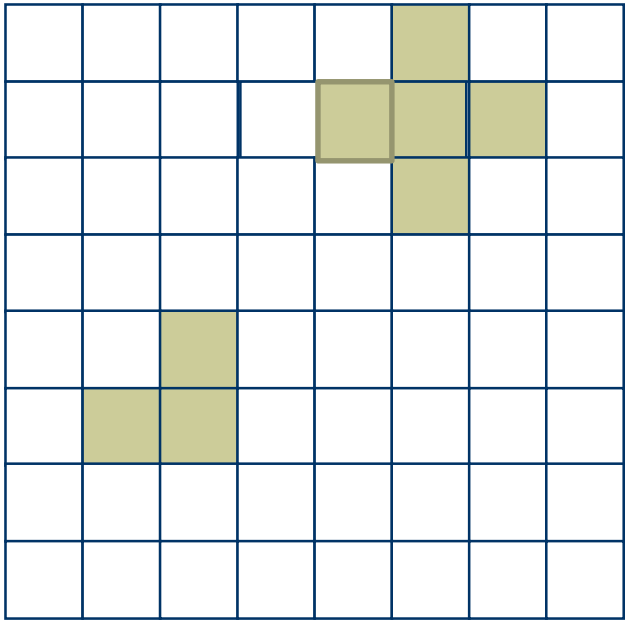


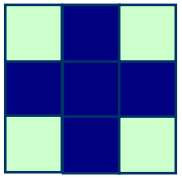
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



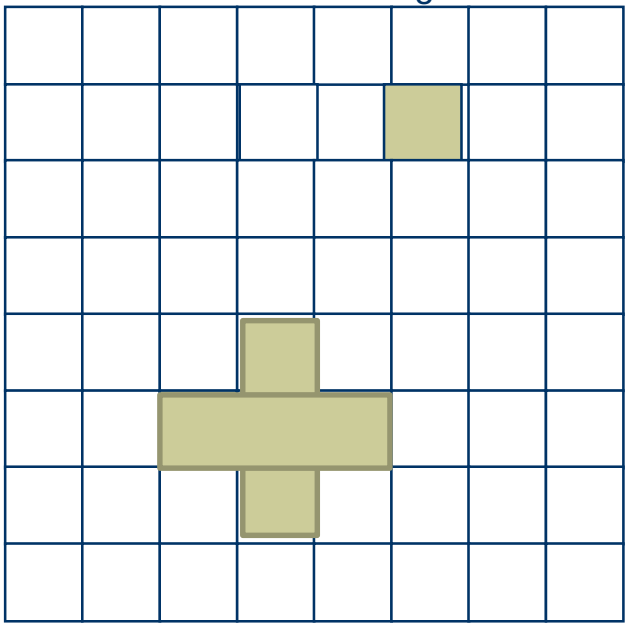


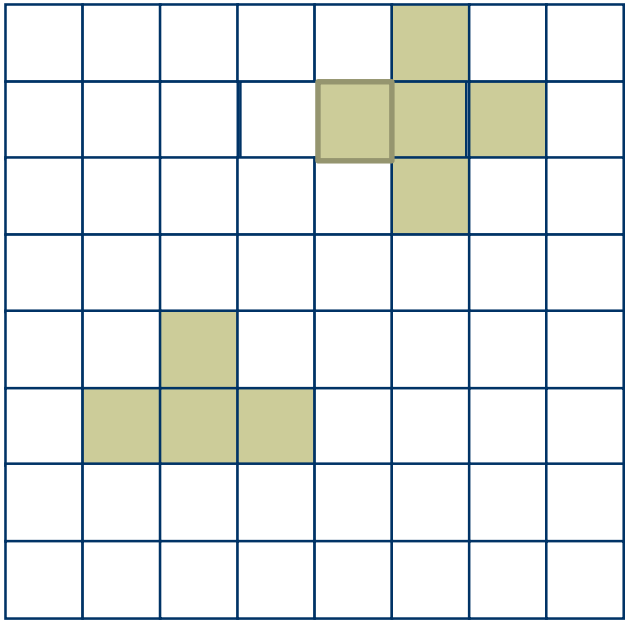


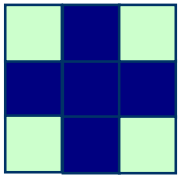
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



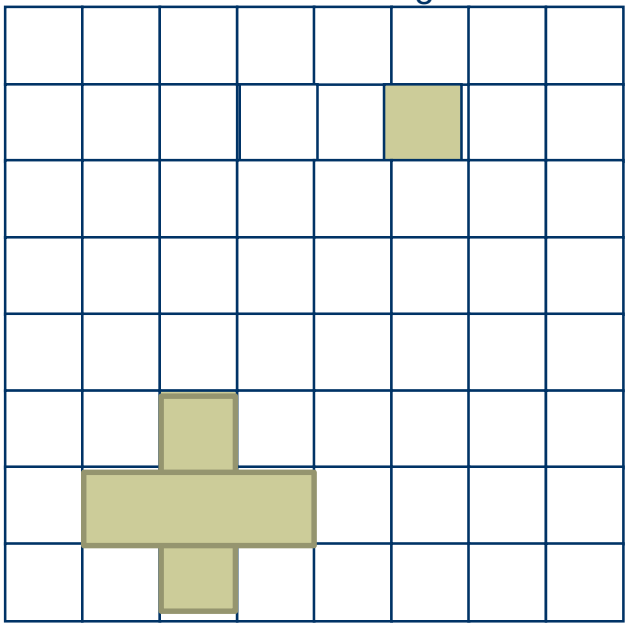


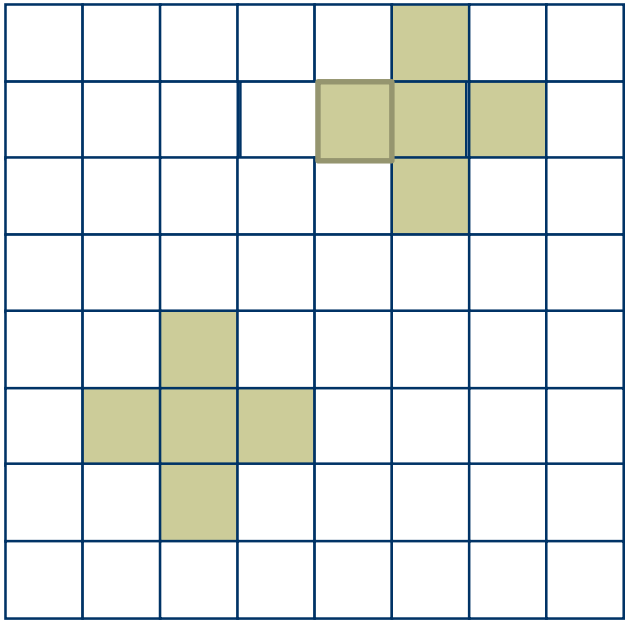


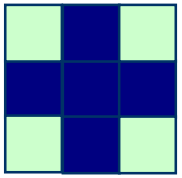
Structuring Element

# Opening: Example (performing dilation to errored image)

Processed Image



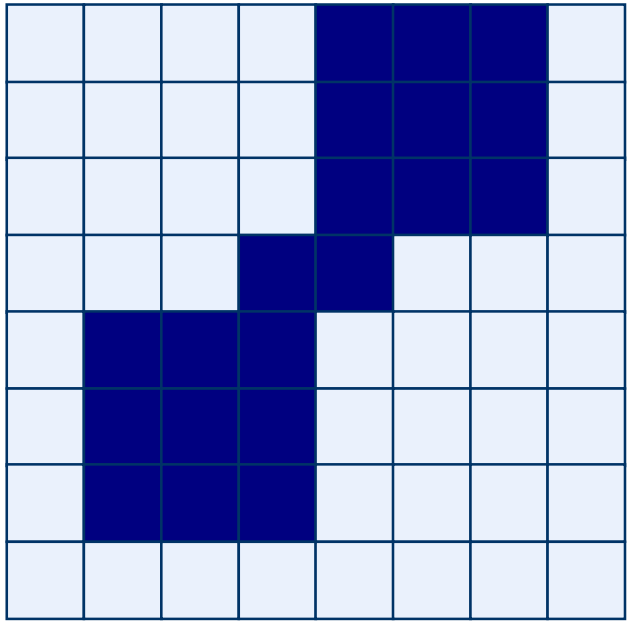




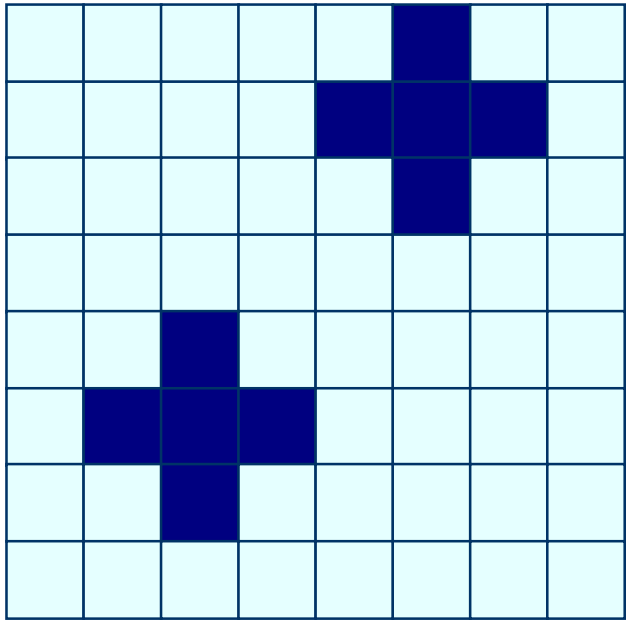
Structuring Element

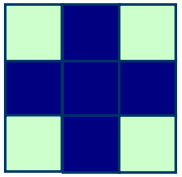
# After Opening:

Original Image



Processed Image



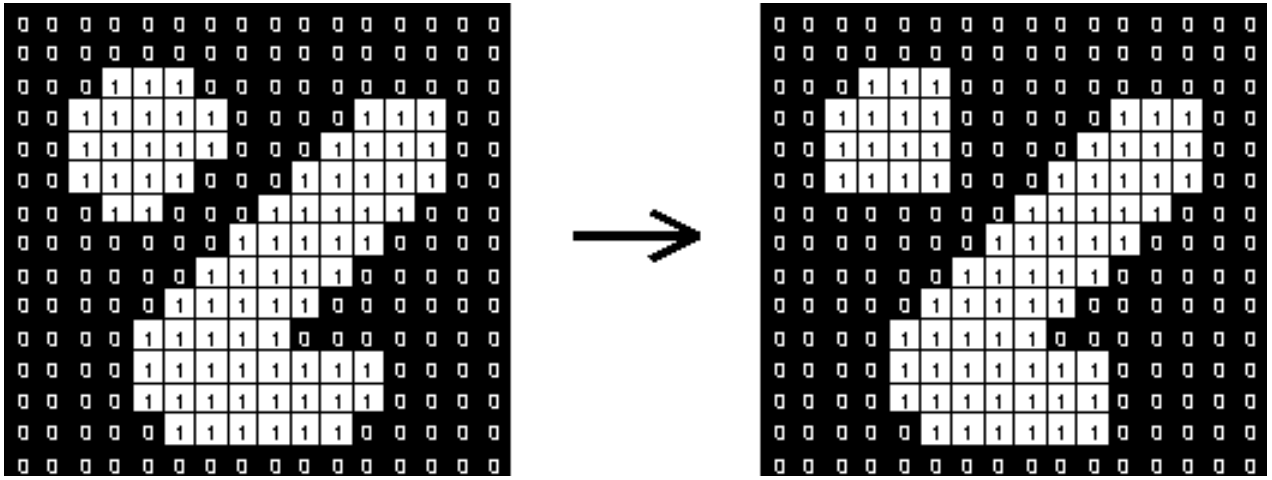


Structuring Element

# Effect of opening

- Structuring element: 3x3 square

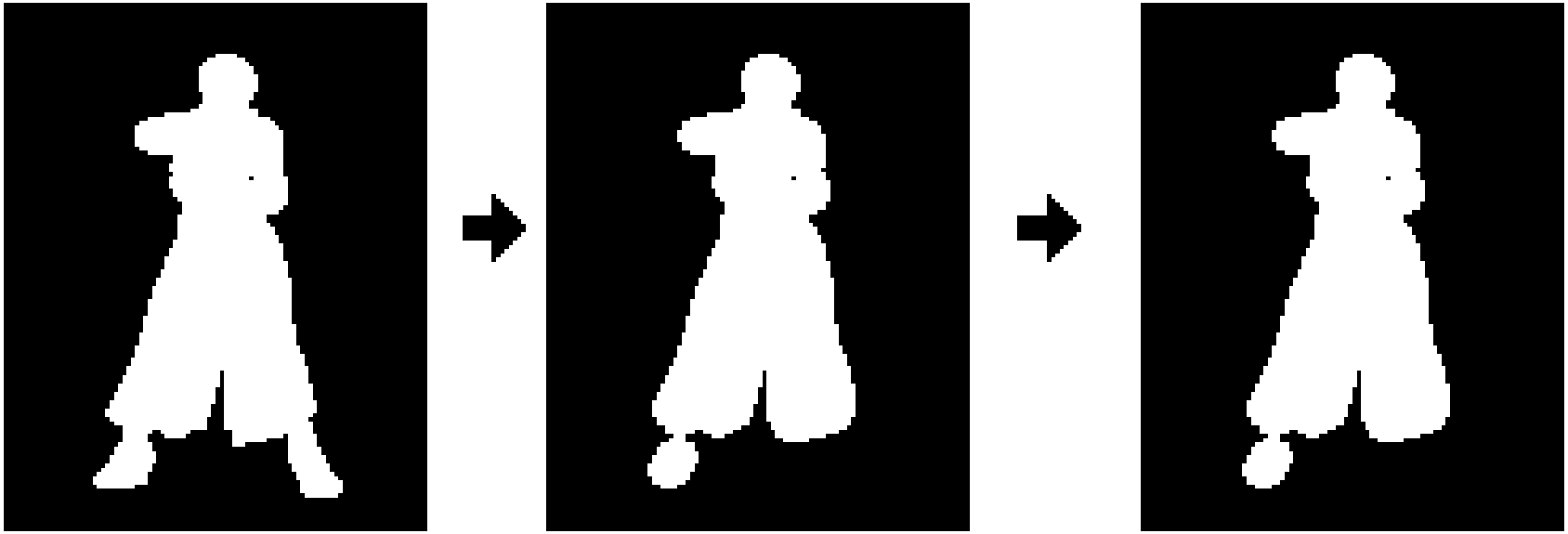
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |



# Effect of opening

- Smoothed the outline, by rounding off any sharp points, and
- Remove any parts that is smaller than the shape (SE) used.
- It will also disconnect or 'open' any thin bridges.
- It does not remove any 'holes', or gaps that may be present in the image.
- It does not make the basic 'core' size of the shape larger or smaller.

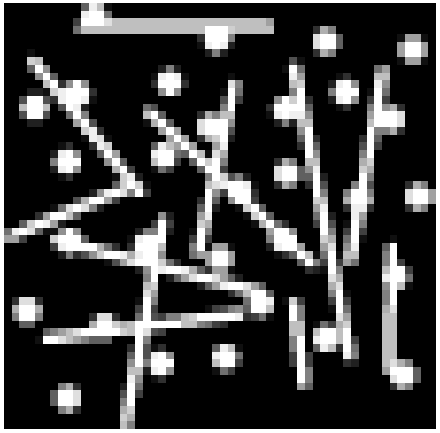
Note that performing an 'Open' on a shape that has already been opened, with the same kernel will result in no further change to the shape. For example…



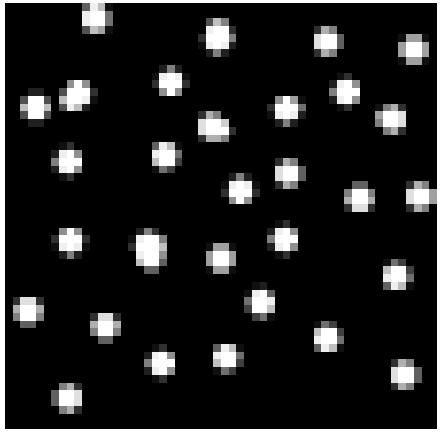
That is repeating a 'Open' operation, with the same kernel, has no effect on the result.

# Effect of opening

Original Image

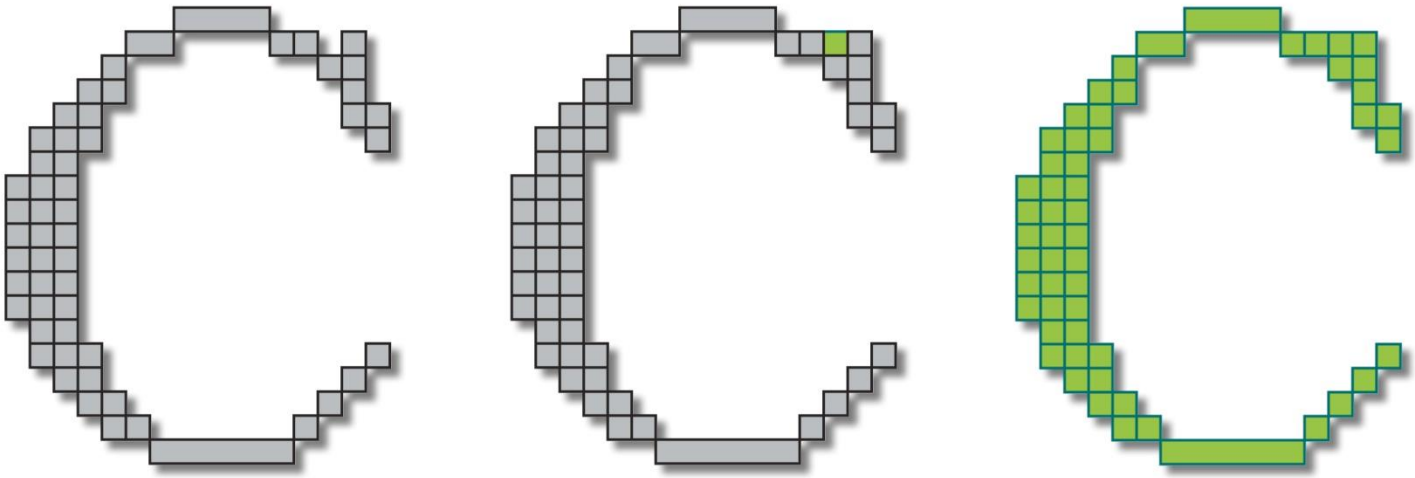


Processed Image



- Opening with a disk shaped structuring element 11 pixels in diameter gives the separation of circles from the lines.
- Some of the circles are slightly distorted, but
- In general, the lines have been almost completely removed while the circles remain almost completely unaffected.

# Closing

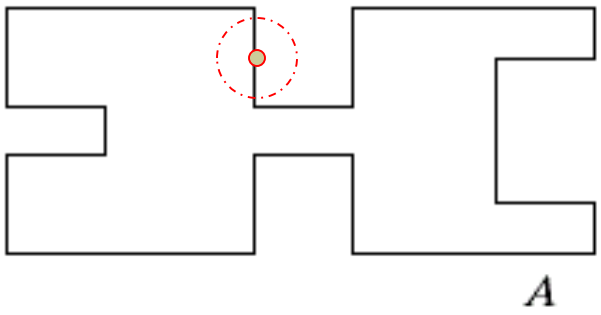


**Dilation followed by an erosion**

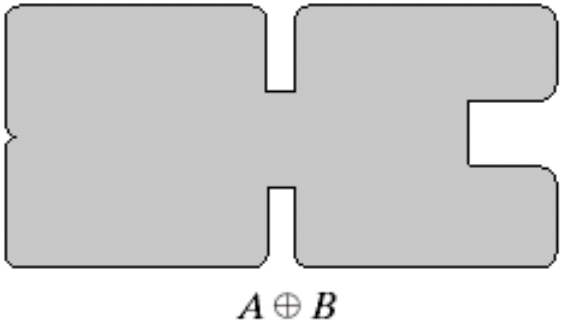
# Closing

The closing of image *f* by structuring element *s*, denoted by $A \bullet B$ is simply a dilation followed by an erosion

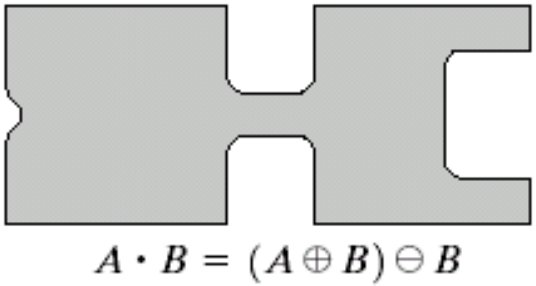
$$A \cdot B = (A \oplus B) \ominus B$$




Original shape



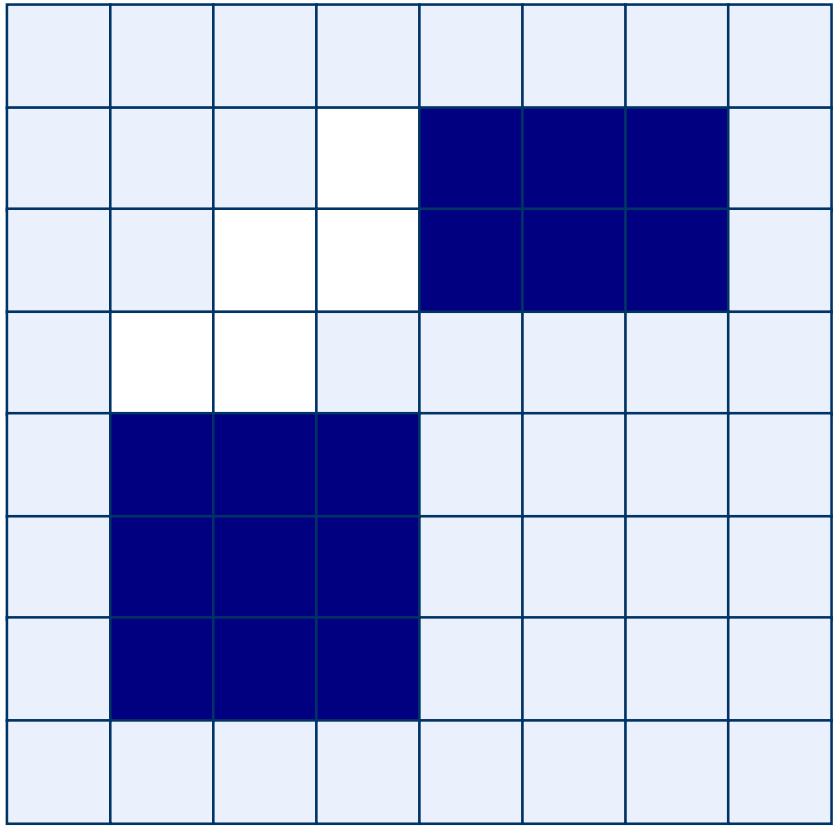

After dilation

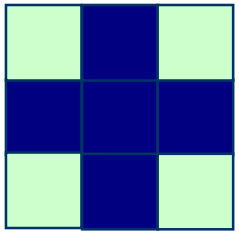



After erosion (closing)

# Closing: Example

Original Image

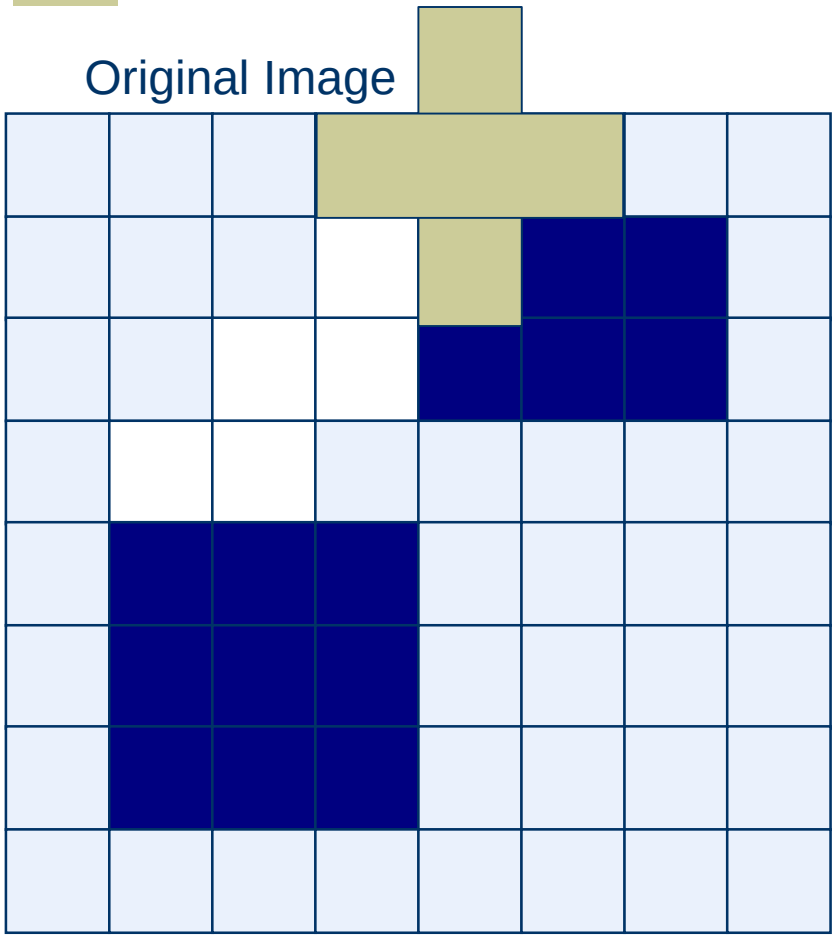




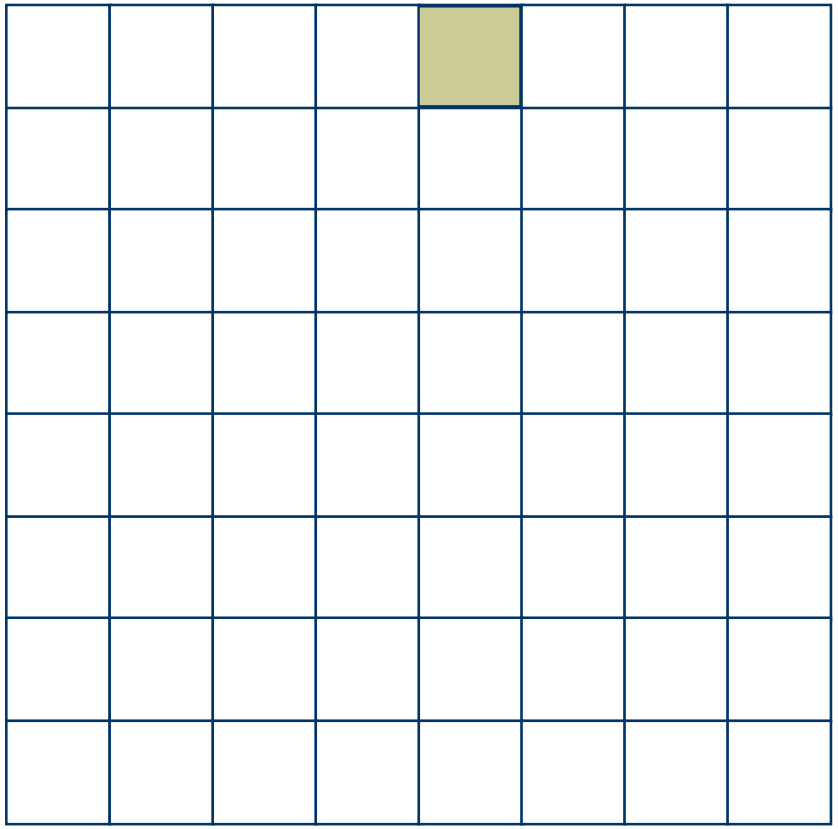
Structuring Element

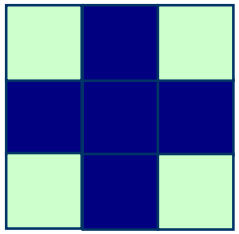
# Closing: Example (performing Dilation)

Original Image



Processed Image

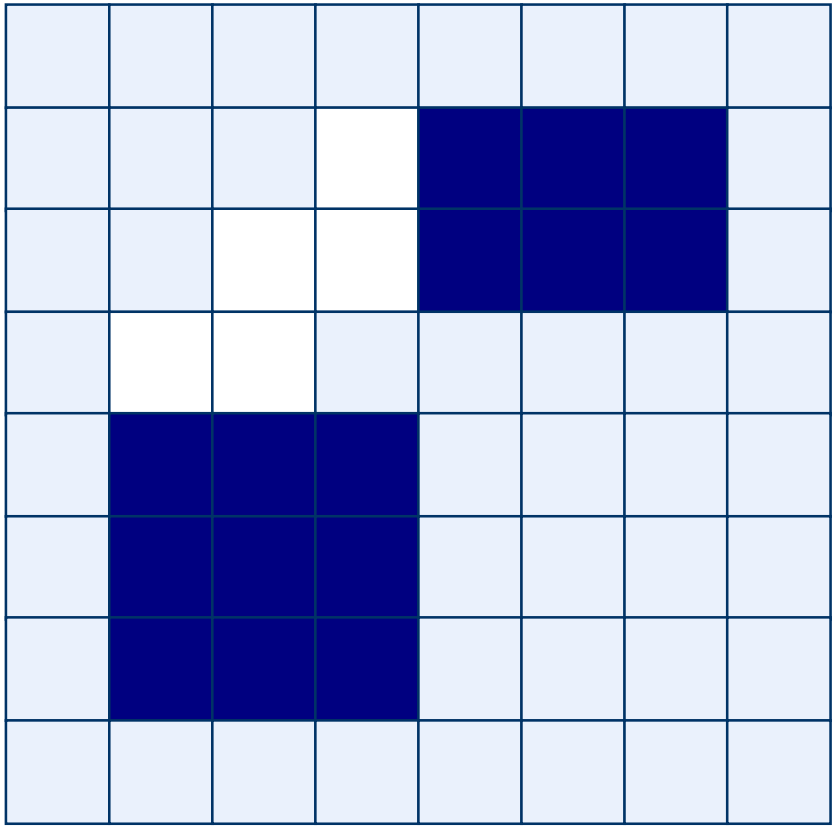




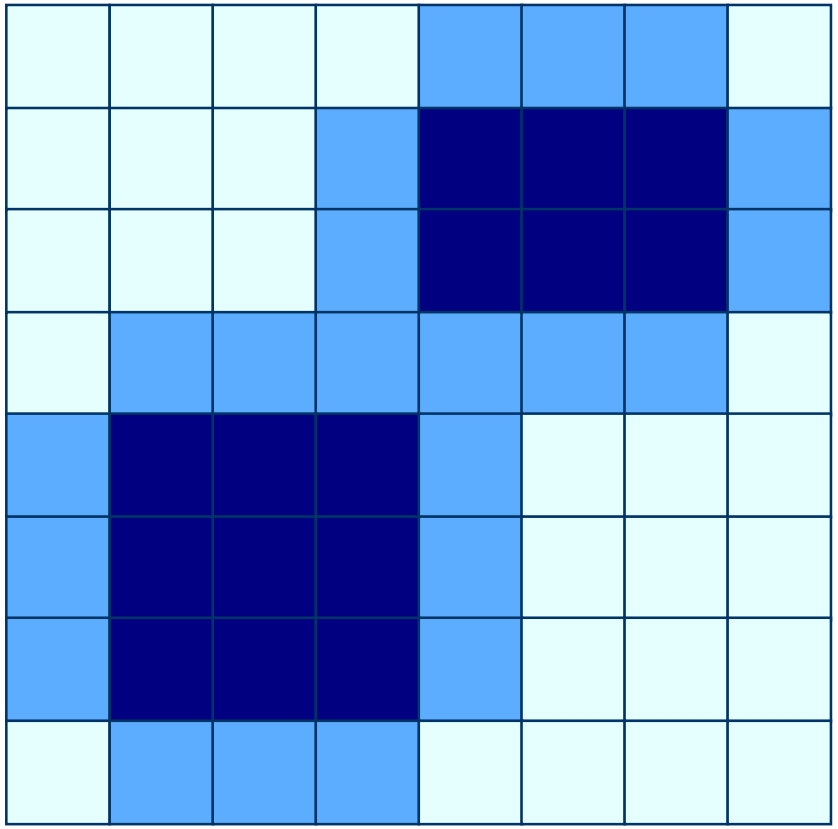
Structuring Element

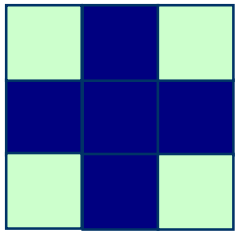
# Closing: Example (after Dilation)

Original Image



Processed Image

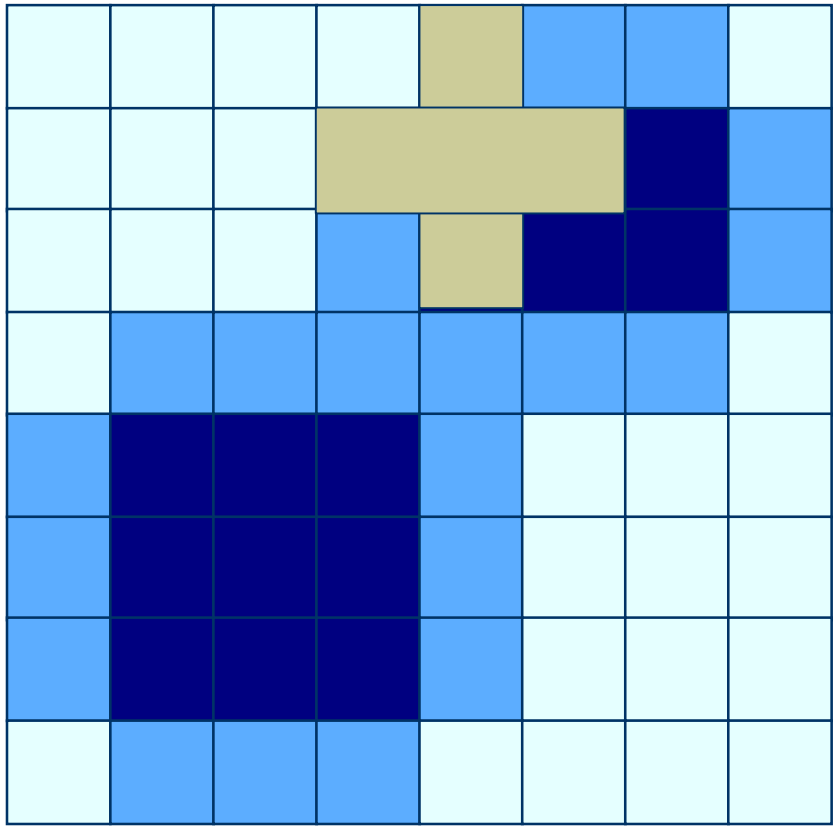




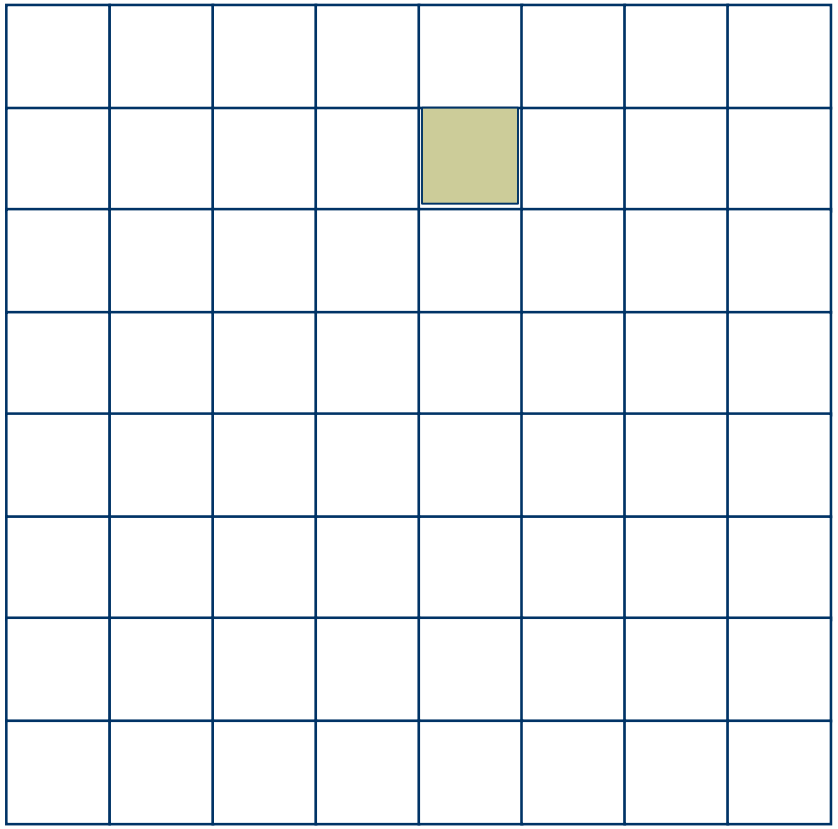
Structuring Element

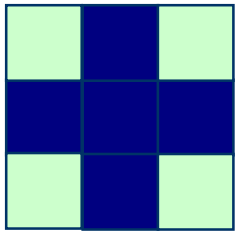
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

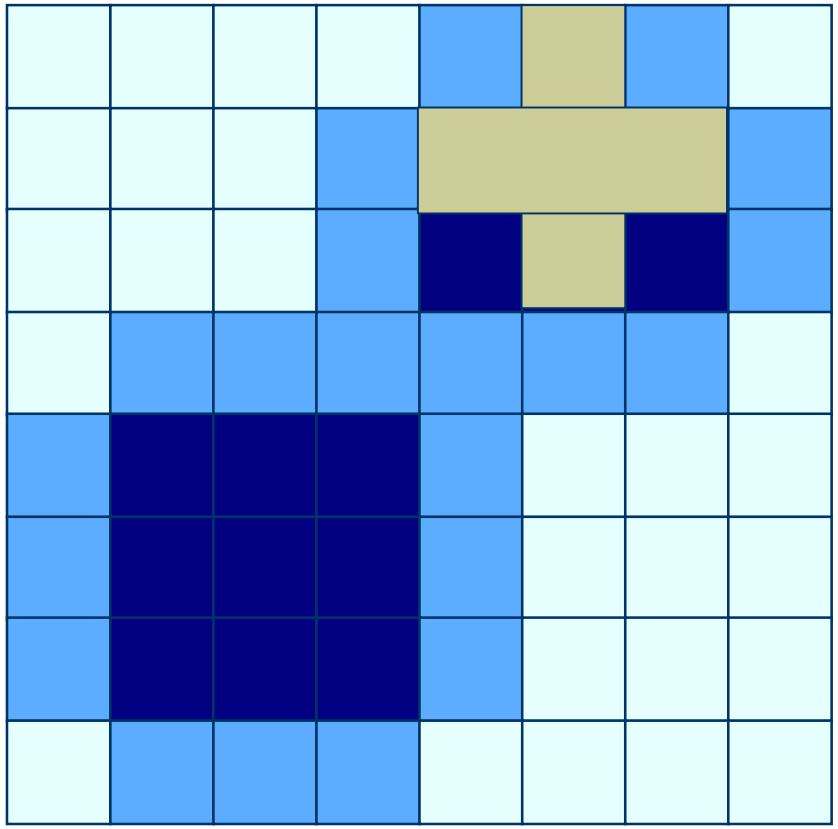




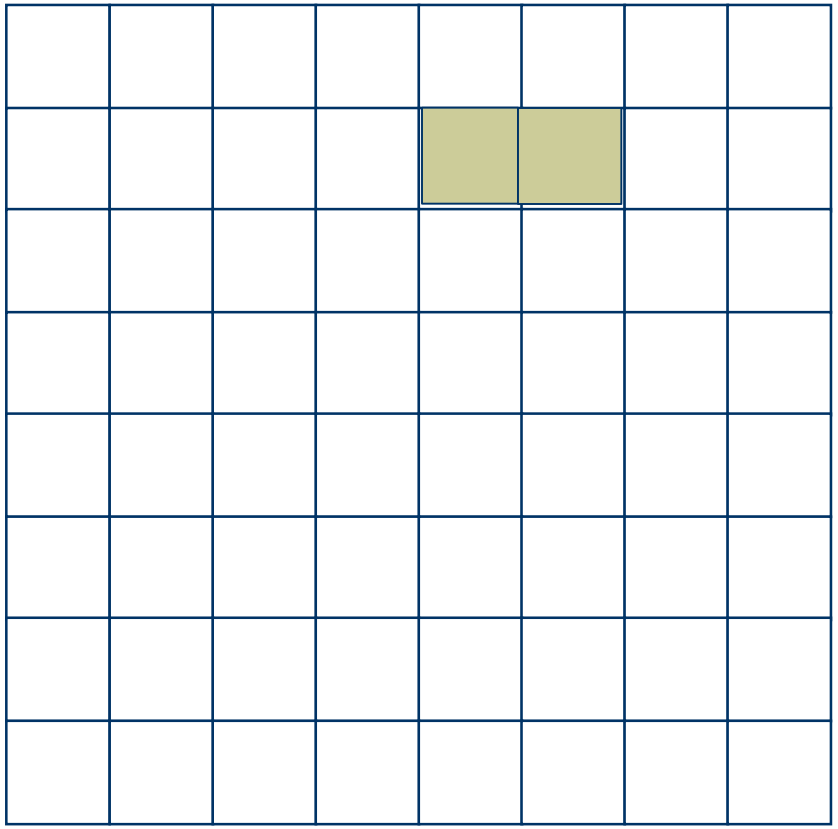
Structuring Element

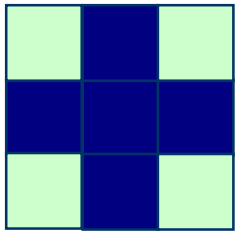
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

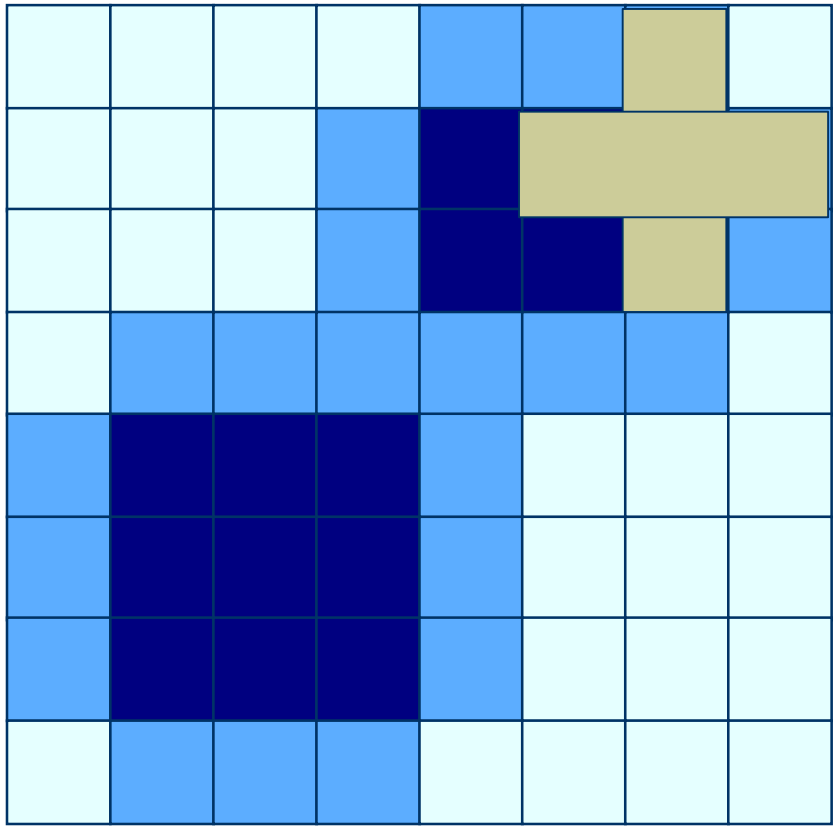




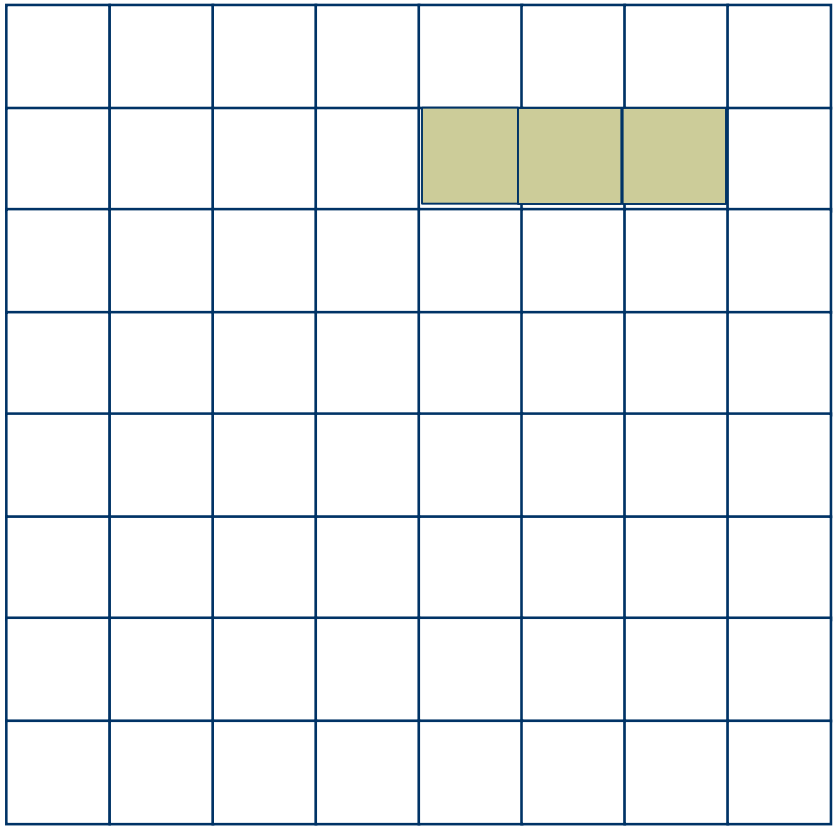
Structuring Element

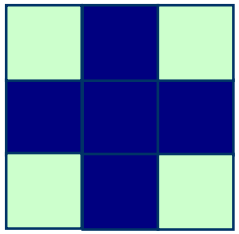
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

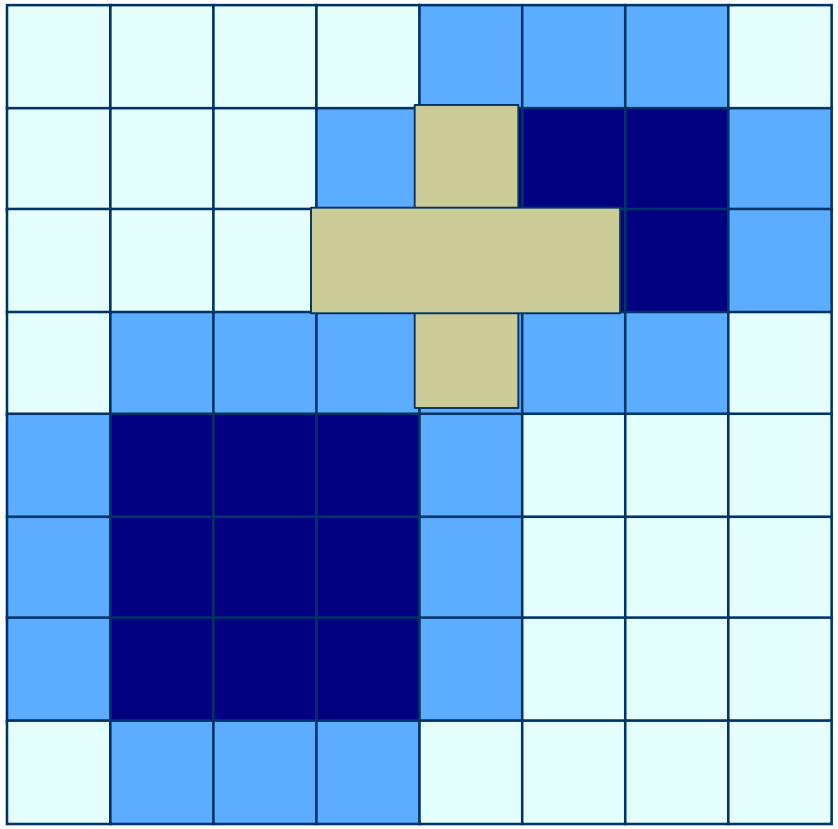




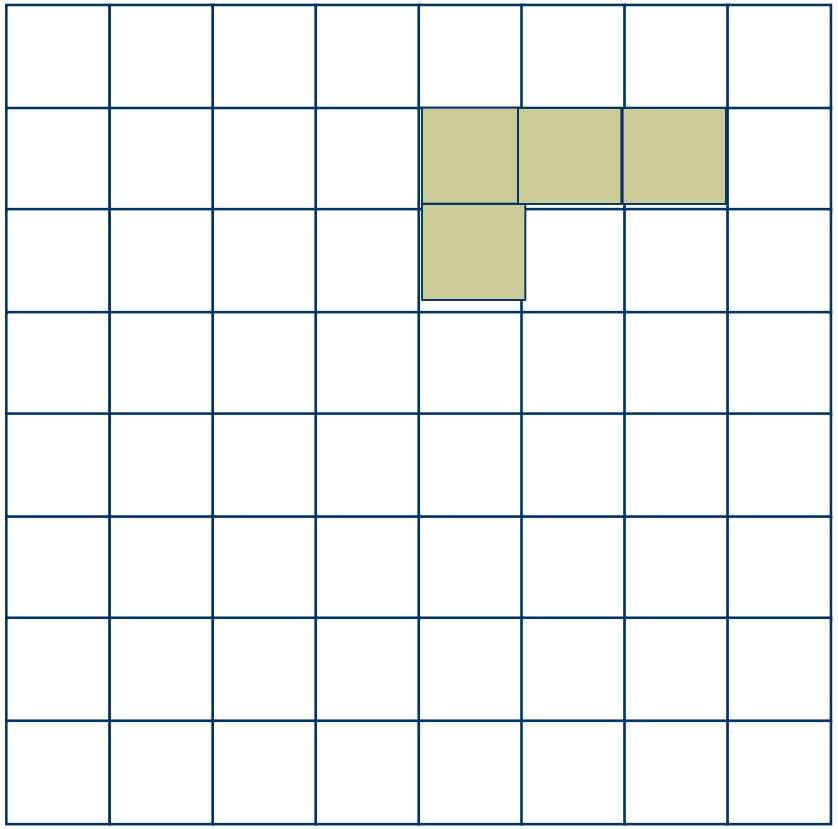
Structuring Element

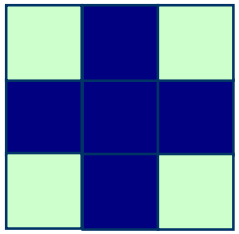
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

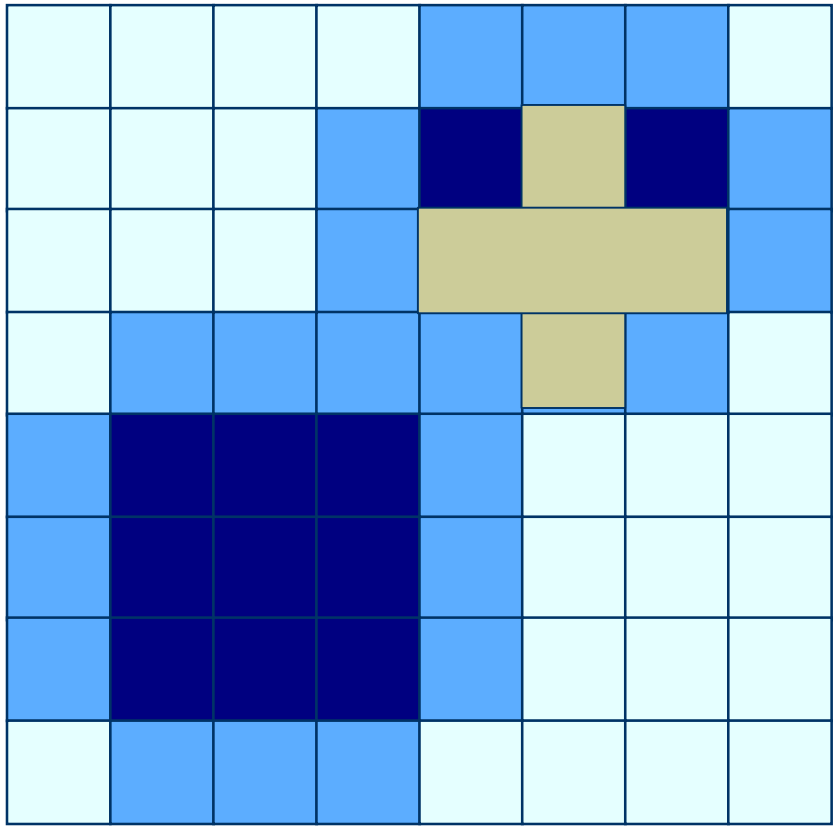




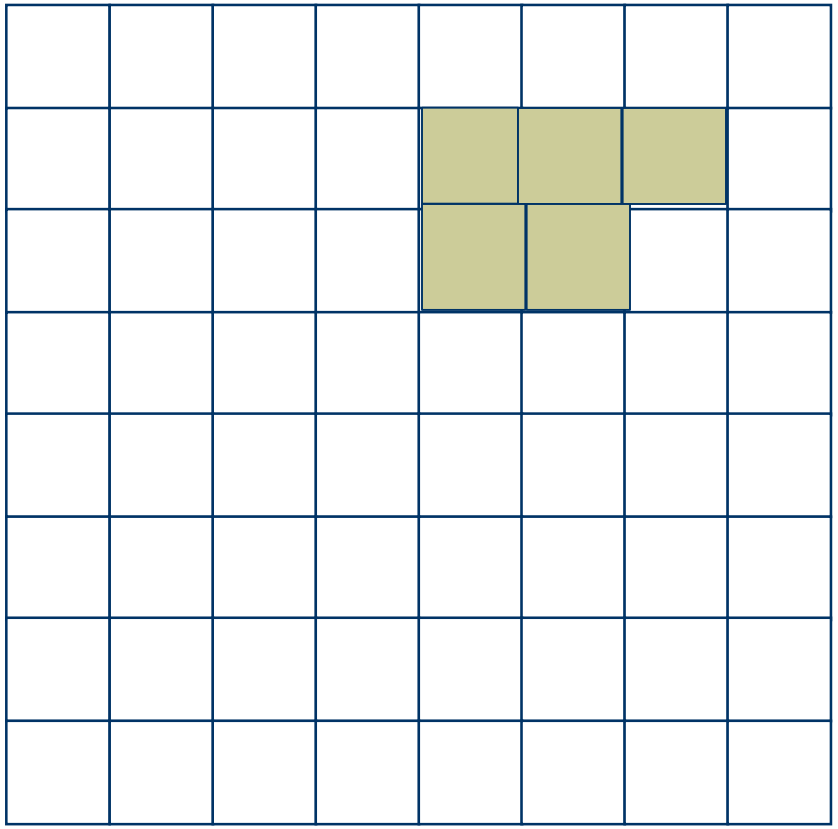
Structuring Element

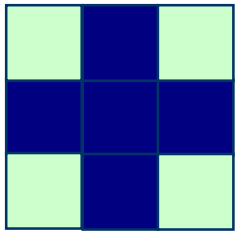
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

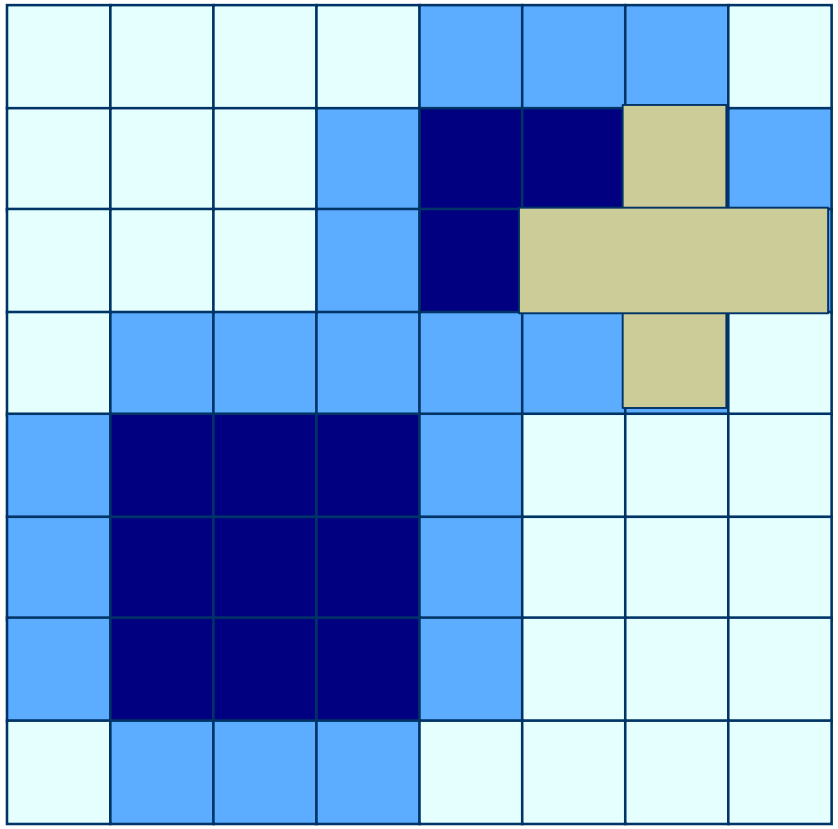




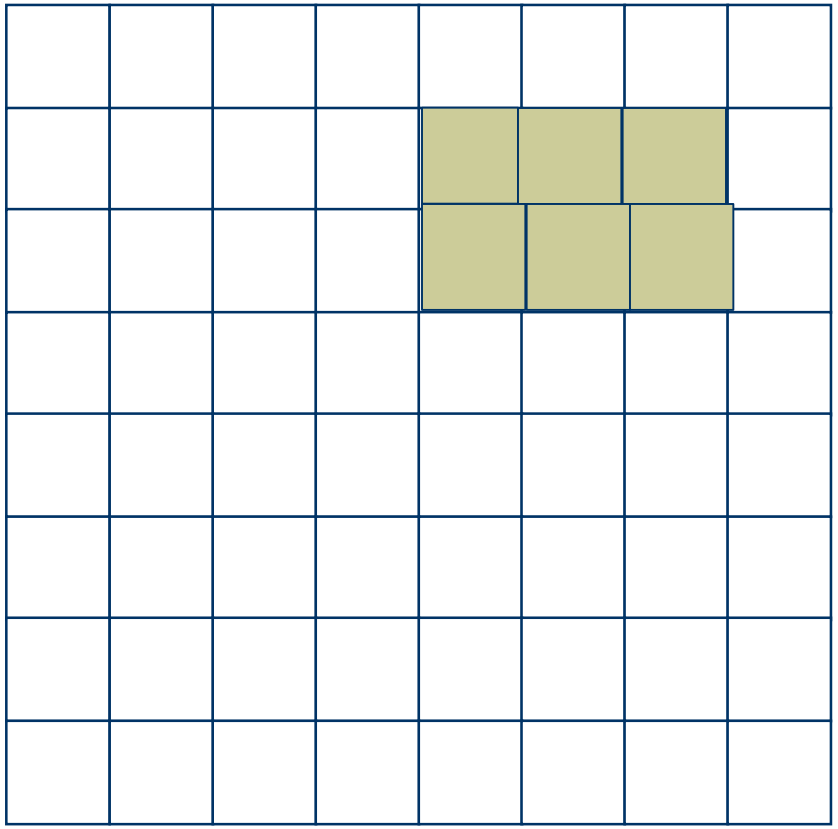
Structuring Element

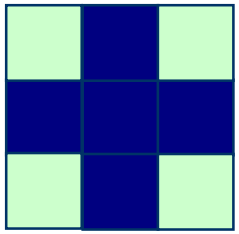
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

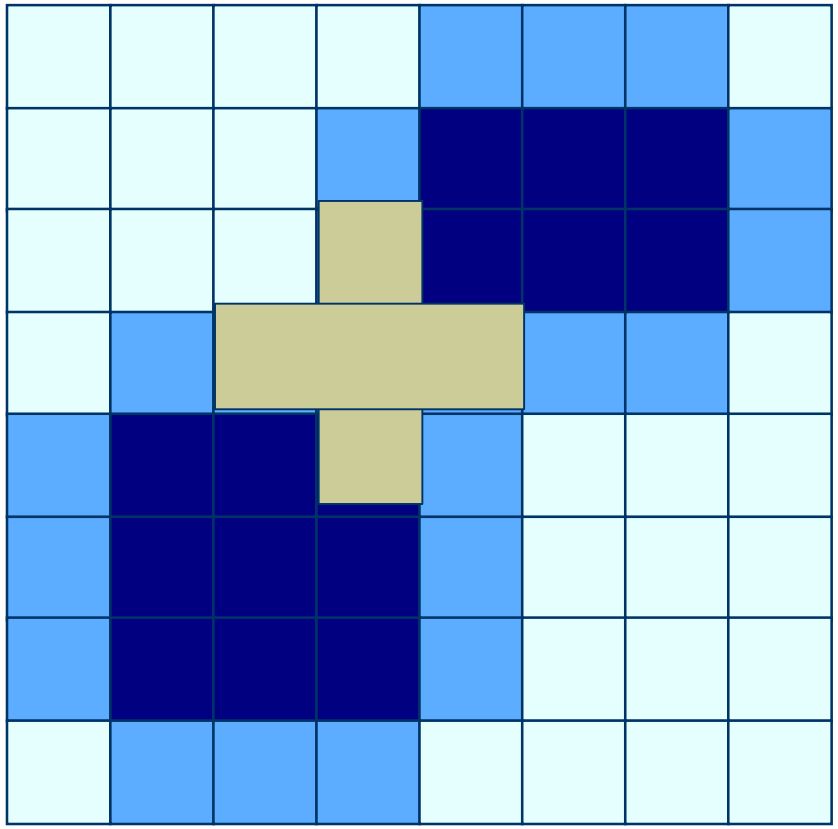




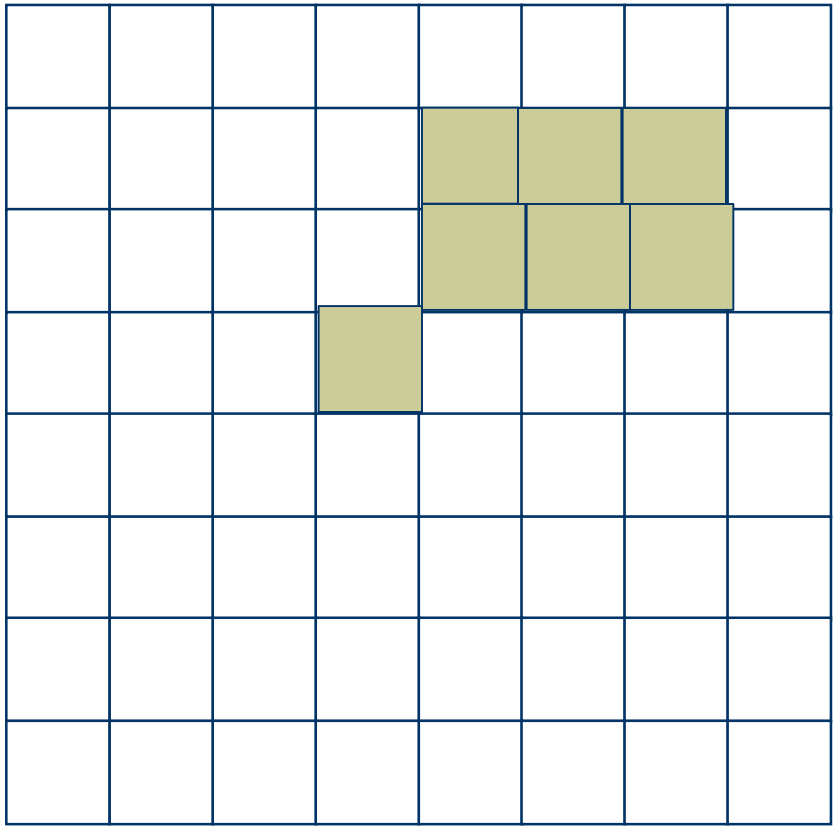
Structuring Element

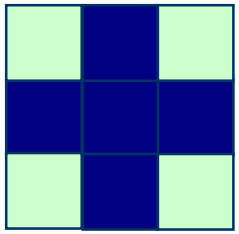
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

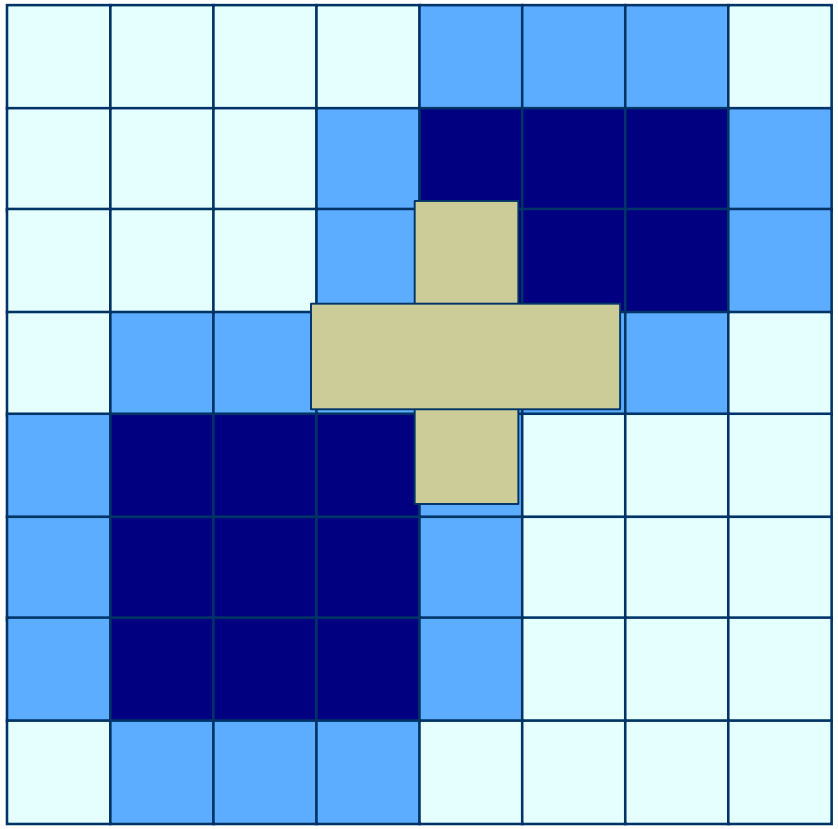




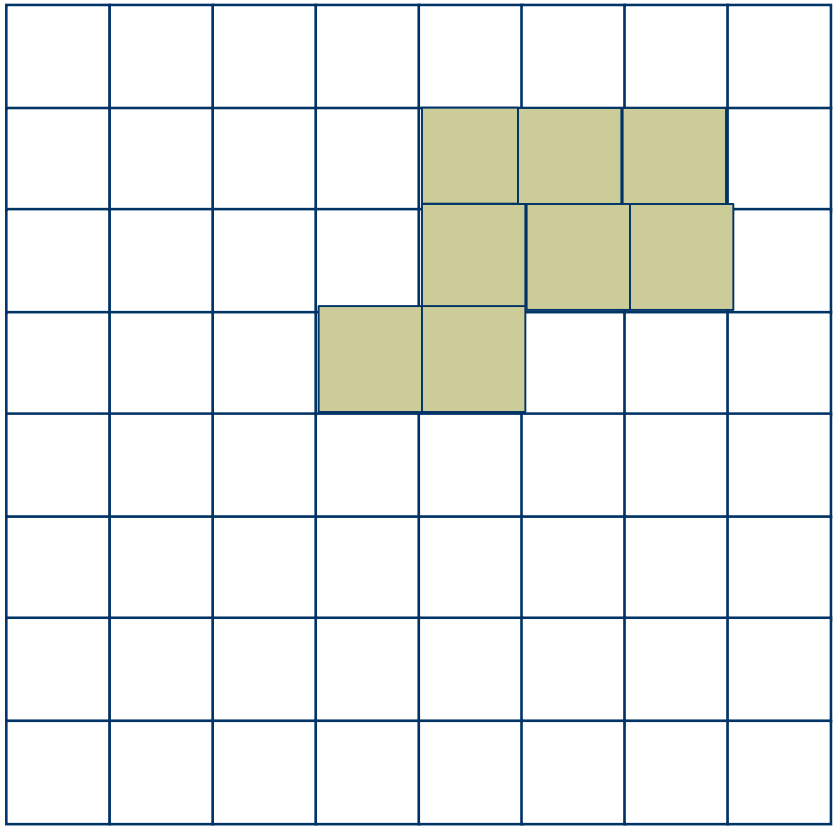
Structuring Element

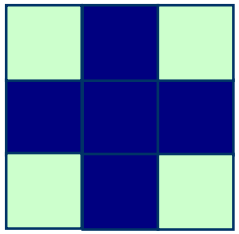
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

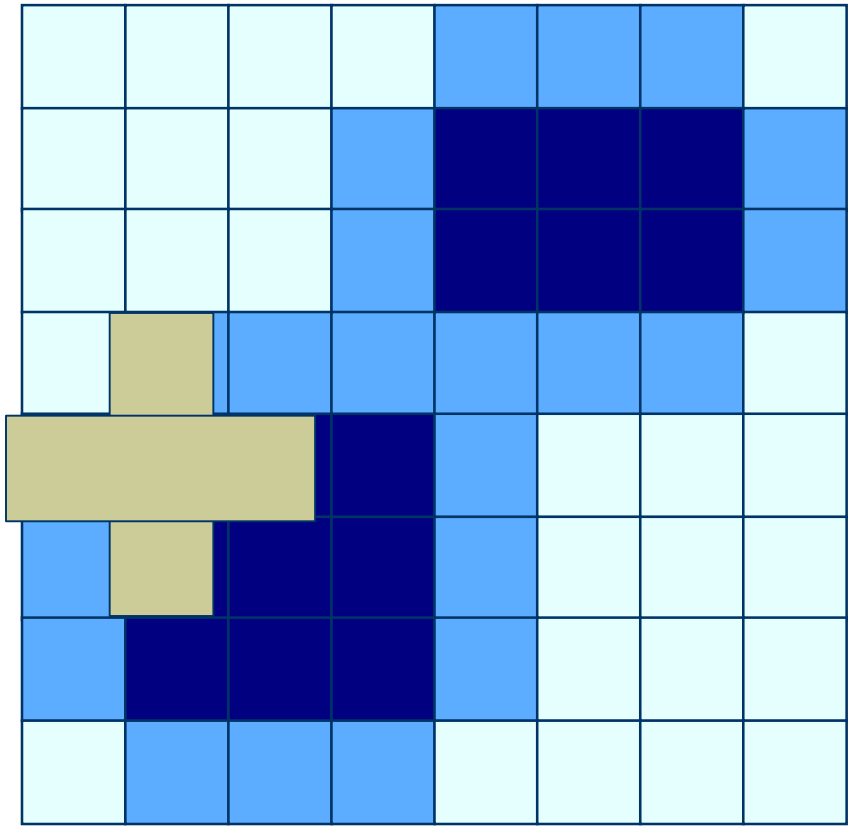




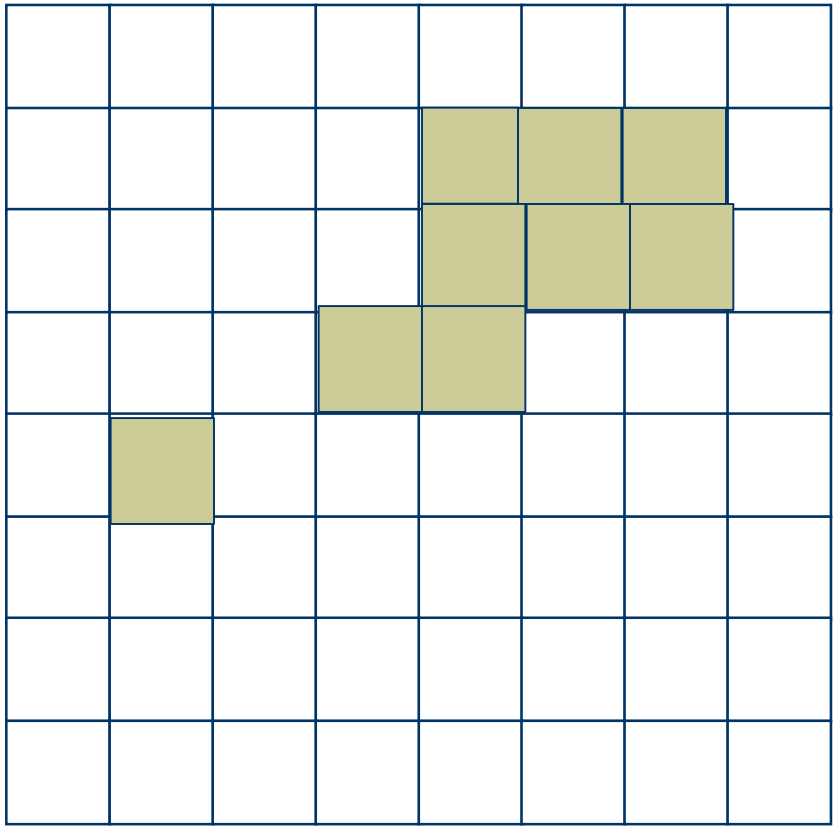
Structuring Element

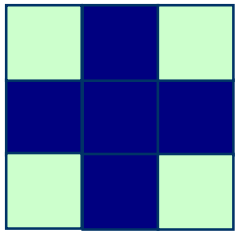
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

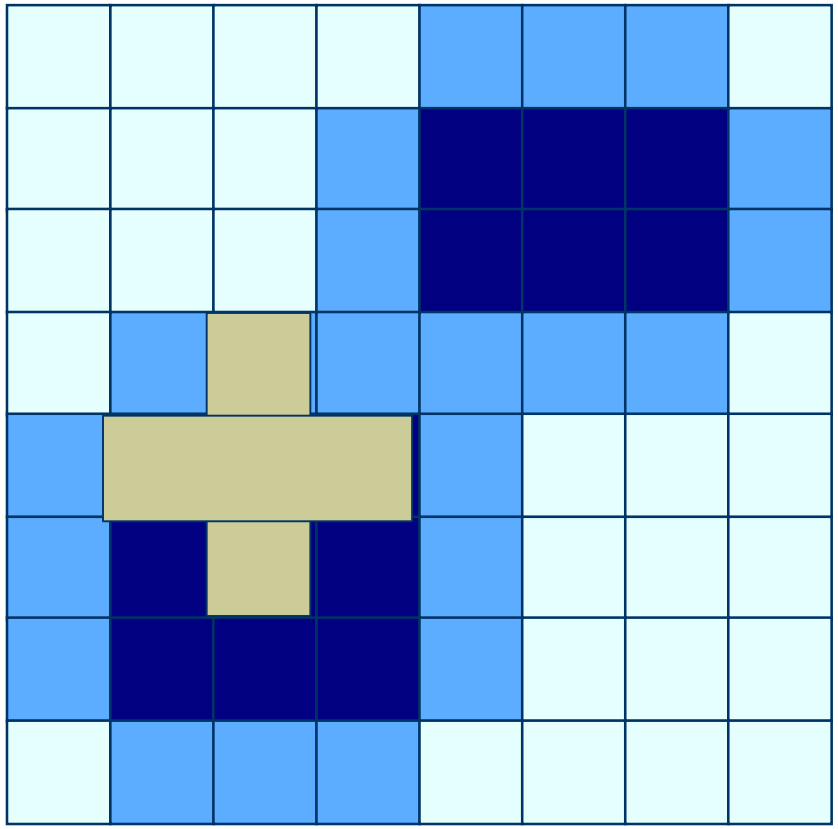




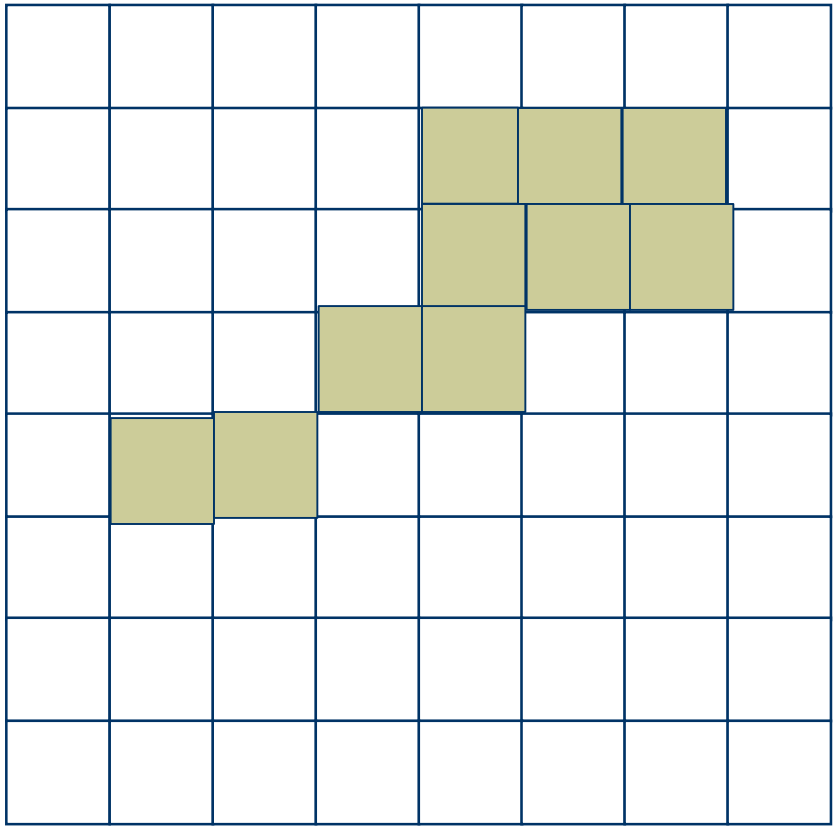
Structuring Element

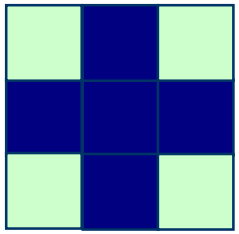
# Closing: Example(performing erosion on Dilated image)

Original Image



Processed Image

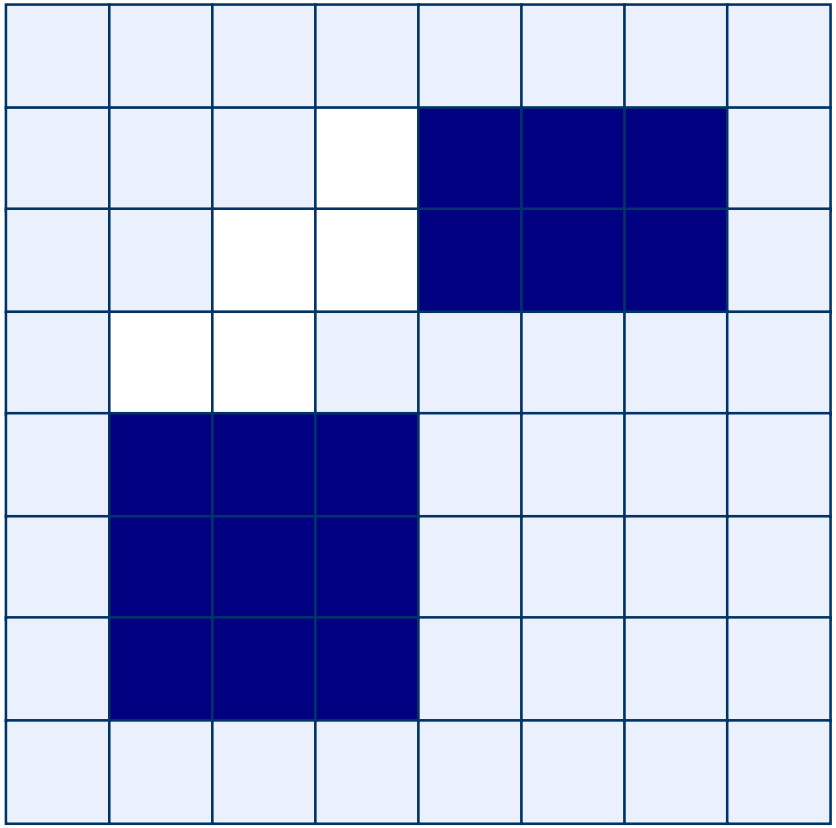




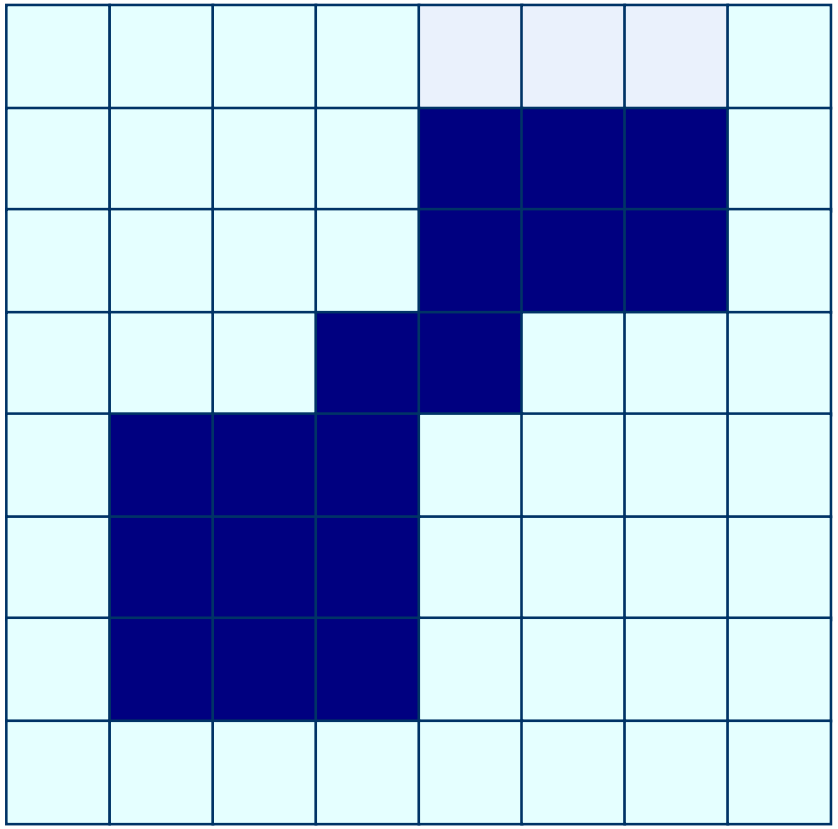
Structuring Element

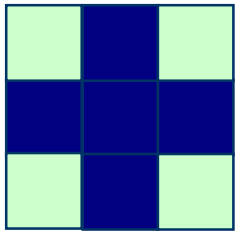
# Closing: Output

Original Image



Processed Image

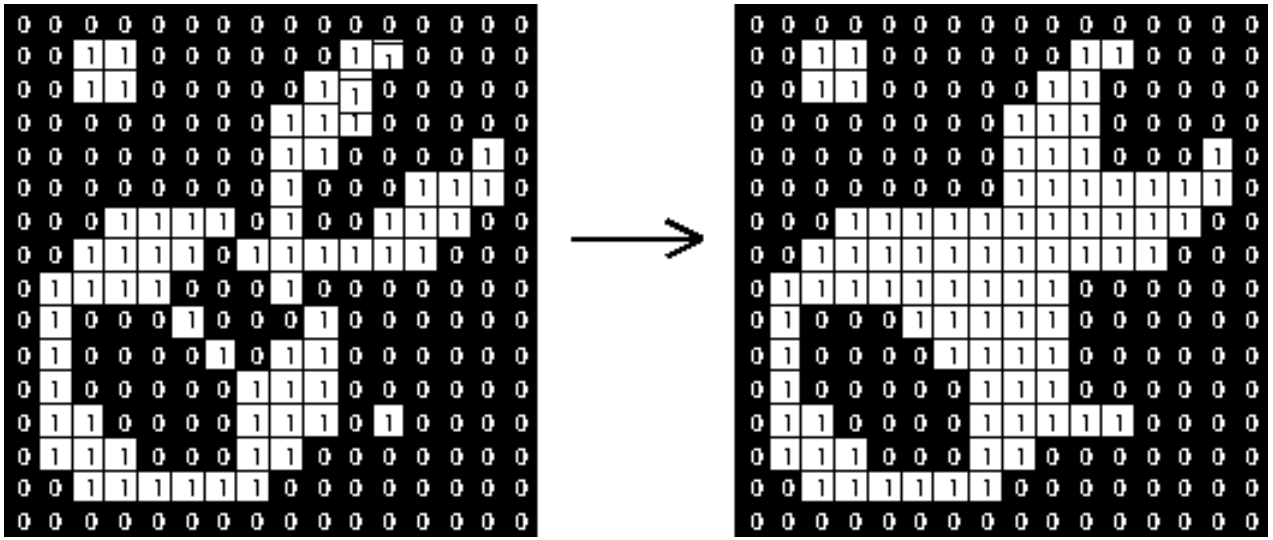




Structuring Element

# Closing

- Structuring element: 3x3 square



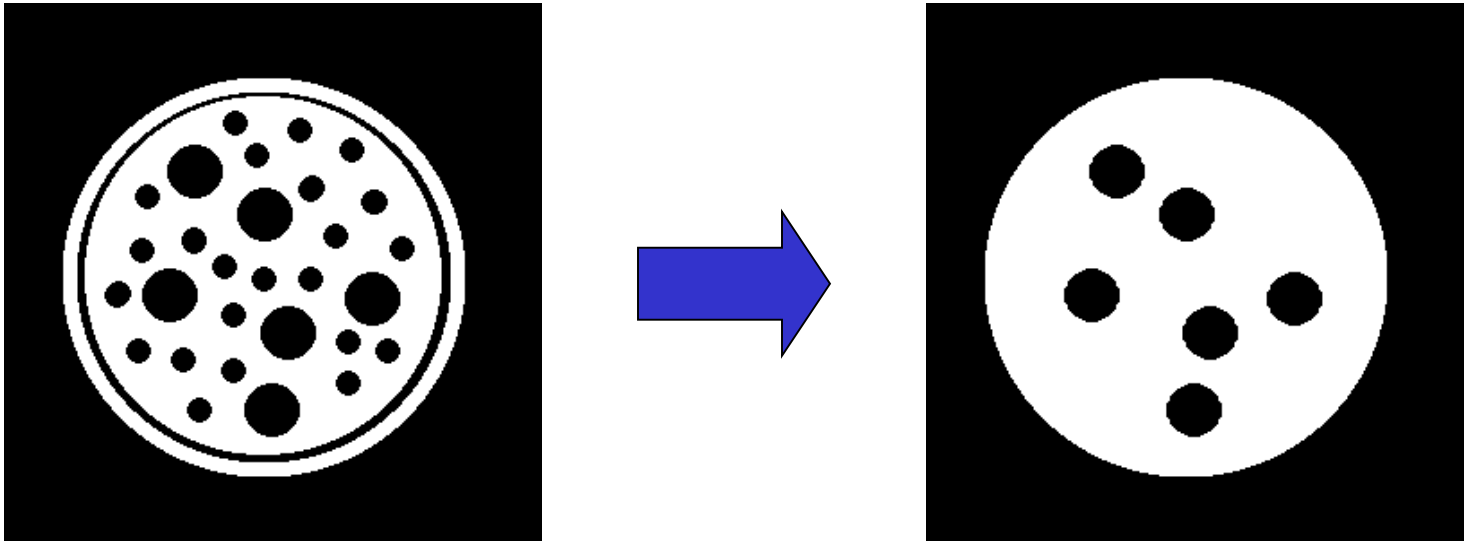
# Effect of closing

- Smoothed the outline, by filling in (closing) any holes and indentations.
- It also will form connecting 'bridges' to other shapes that are close enough for the kernel (SE) to touch both simultaneously.
- But it does not make the basic 'core' size of the shape larger or smaller.

As with **Open**', repeating the '**Close**' method with the same kernel does not make any further changes to the image.

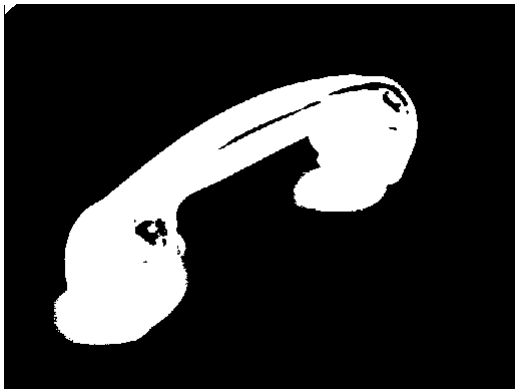
# Closing Example

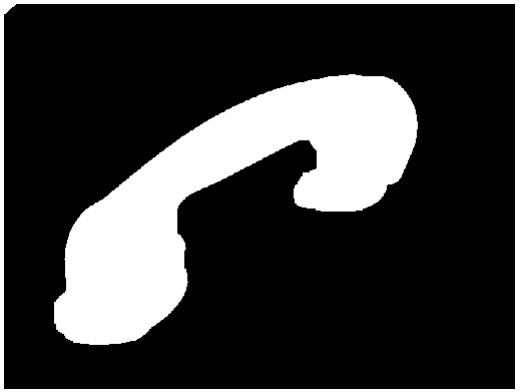
- Closing operation with a 22 pixel disc
- Closes small holes in the foreground



# Closing Example 1

1. Threshold
2. Closing with disc of size 20





Thresholded closed

# 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.

***Home Work: Make an example with simulations.

# Next class

**Outlook:**

**Hit-and-miss Transformation,**

**Thinning, and**

**Thickening**

# Basic Set Theory

- The set space of binary image is $Z^2$
  - Each element of the set is a 2D vector whose coordinates are the *(x,y)* of a black (or white, depending on the convention) pixel in the image

- The set space of gray level image is $Z^3$
  - Each element of the set is a 3D vector: *(x,y)* and intensity level.

**NOTE:**

Set Theory  Logical operations

# Basic Set Theory

- Let A be a set in $Z^2$. if $a = (a_1,a_2)$ is an element of A, then we write

$$a \in A$$

- If a is not an element of A, we write

$$a \notin A$$

- Set representation

$$A = \{(a_1, a_2), (a_3, a_4)\}$$

- Empty or Null set

$$A = \varnothing$$

# Basic Set Theory

- Subset: if every element of A is also an element of another set B, the A is said to be a subset of B

$$A \subseteq B$$

- Union: The set of all elements belonging either to A, B or both

$$C = A \bigcup B$$

- Intersection: The set of all elements belonging to both A and B

$$D = A \bigcap B$$

# Basic Set Theory

- Two sets A and B are said to be disjoint or mutually exclusive if they have no common element

$$A \bigcap B = \varnothing$$

- Complement: The set of elements not contained in A

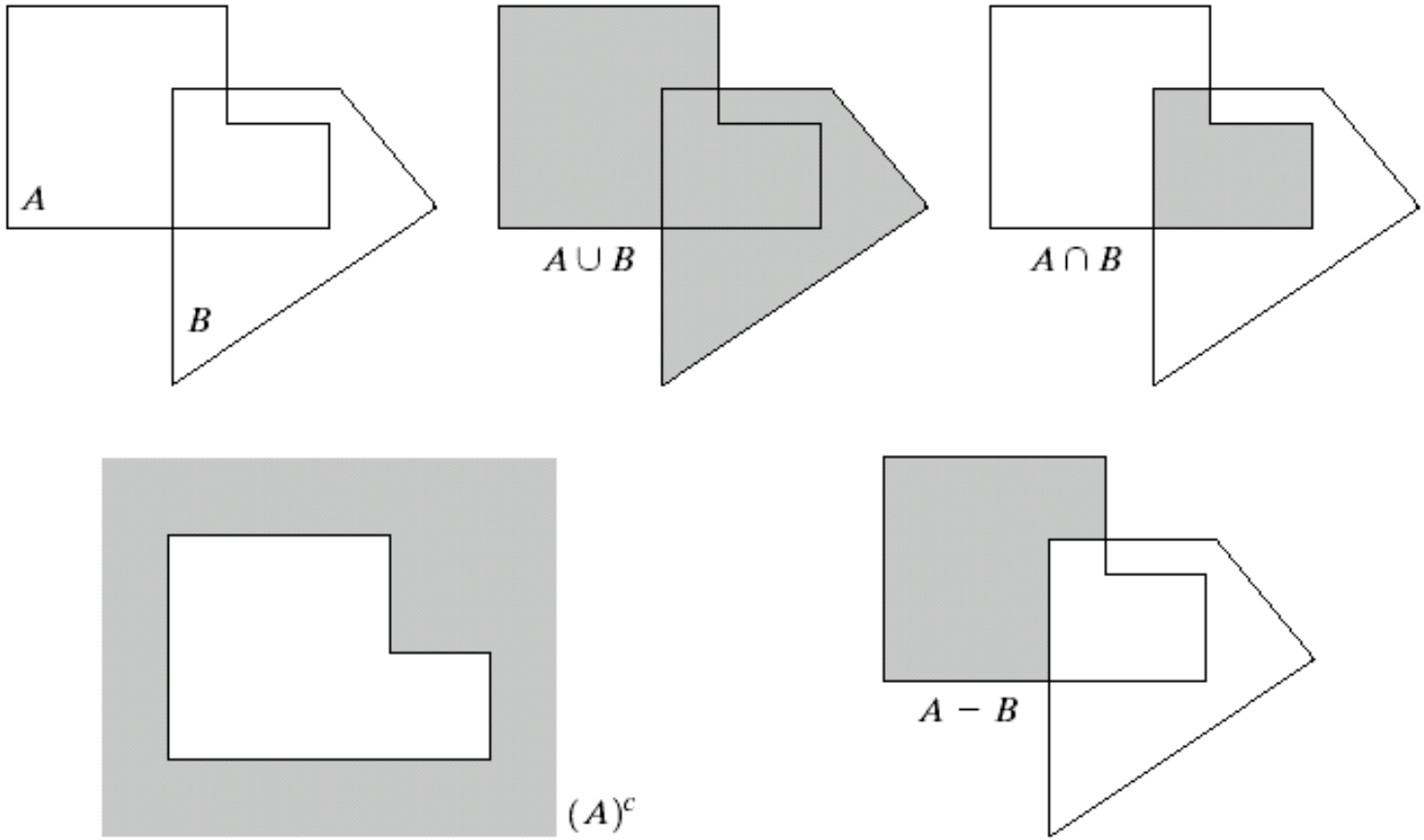
$$A^c = \{w \mid w \notin A\}$$

- Difference of two sets A and B, denoted by A – B, is defined as

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

  i.e. the set of elements that belong to A, but not to B

# Basic Set Theory




a b c
d e

**FIGURE 9.1**
(a) Two sets $A$ and $B$. (b) The union of $A$ and $B$. (c) The intersection of $A$ and $B$. (d) The complement of $A$. (e) The difference between $A$ and $B$.

# Example of some logic operations:

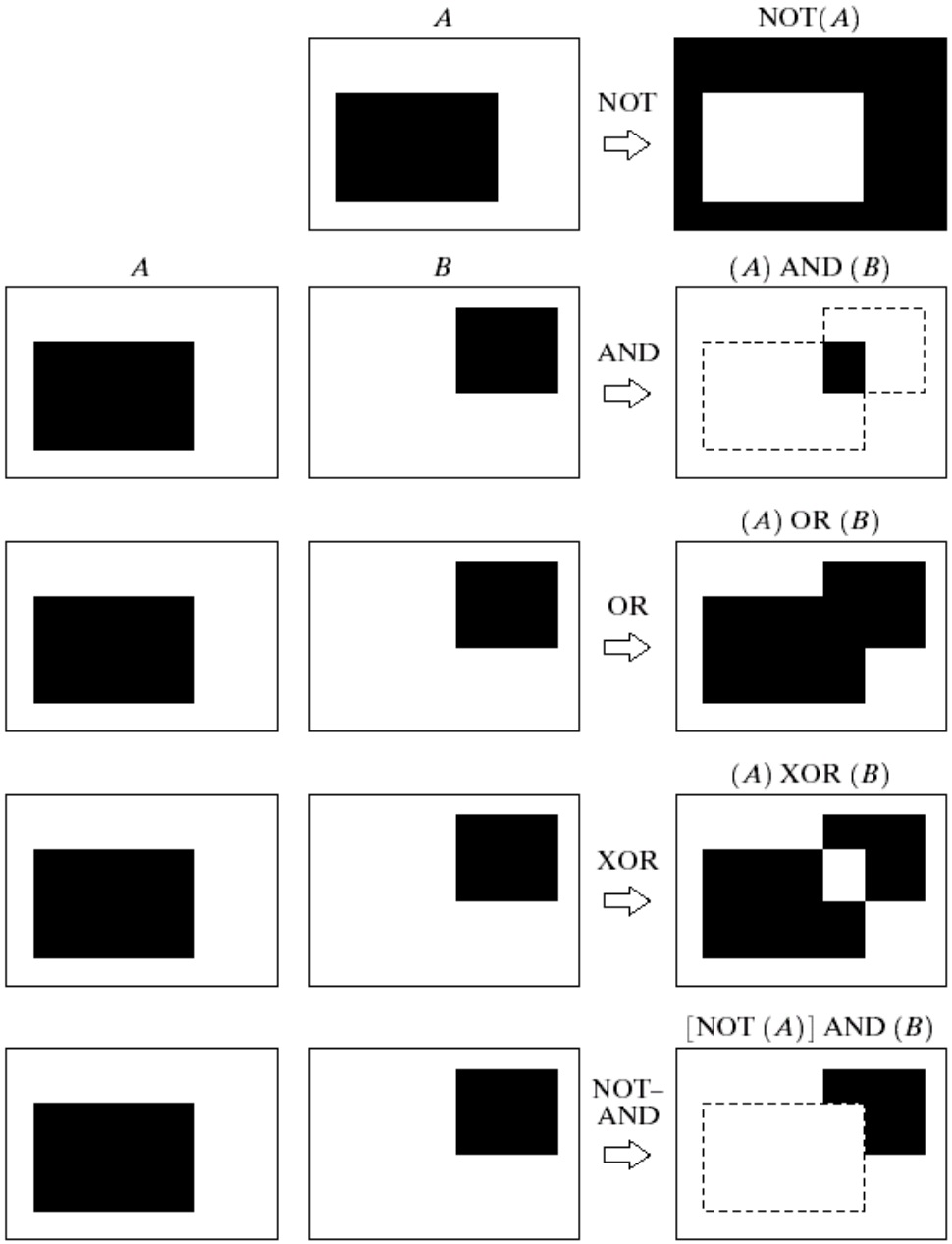


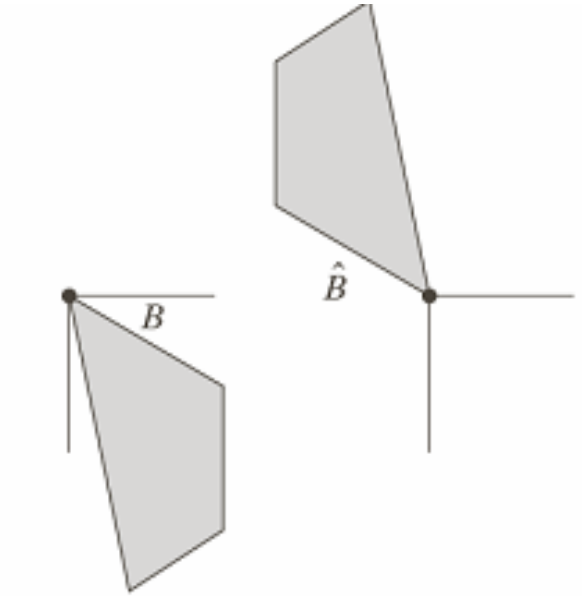

**FIGURE 9.3** Some logic operations between binary images. Black represents binary 1s and white binary 0s in this example.

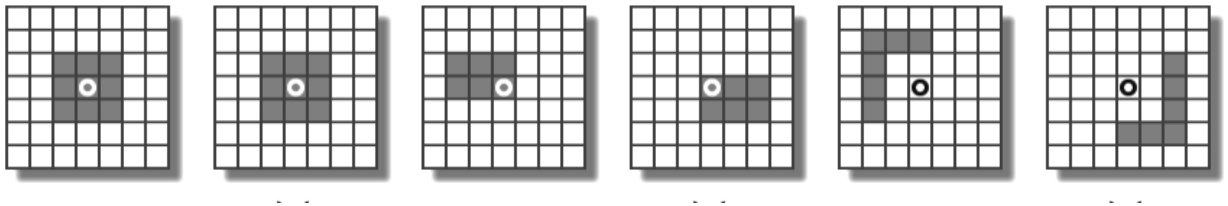
# Reflection of set B

- Reflection of set B

$$B = \{w \mid w = -b, \quad \text{for } b \in B\}$$

i.e. the set of element *w*, such that *w* is formed by multiplying each of two coordinates of all the elements of set *B* by -1

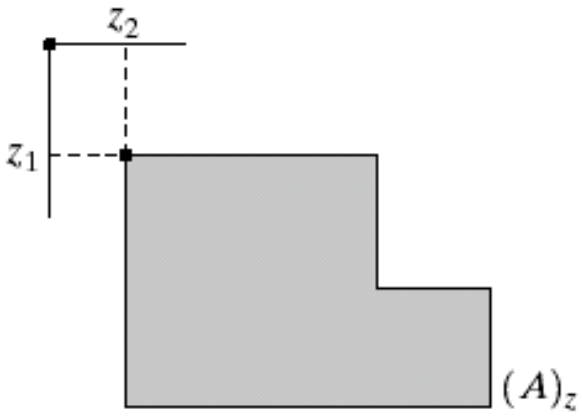


# Translation of set A

- Translation of set A by point z = $(z_1,z_2)$, denoted $(A)_z$, is defined as

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



# Structuring Element: Translation

Let I be an image and *B* a SE.

$(B)_z$ means that *B* is moved so that its origin coincides with location z in image *I*.

$(B)_z$ is the *translate* of *B* to location *z* in *I*.