



# Computer Vision 09 – Image processing and analysis 2a

doc. dr. Janez Perš  
(with contributions by prof. Stanislav Kovačič)

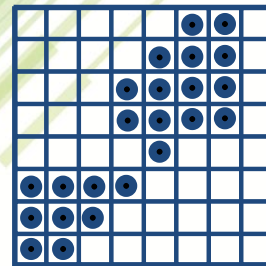
Laboratory for Machine Intelligence  
Faculty of Electrical Engineering  
University of Ljubljana

# Quick recap of the previous lectures

- Image formation
- Color
- Image processing
  - Value of the pixel depends only on that pixel
- Local image operations
- Edges
- Hough transform

# Outline

- Segmentation (conceptual)
  - Important class of CV algorithms
- Connected components labeling



Connectivity

- Morphological operations

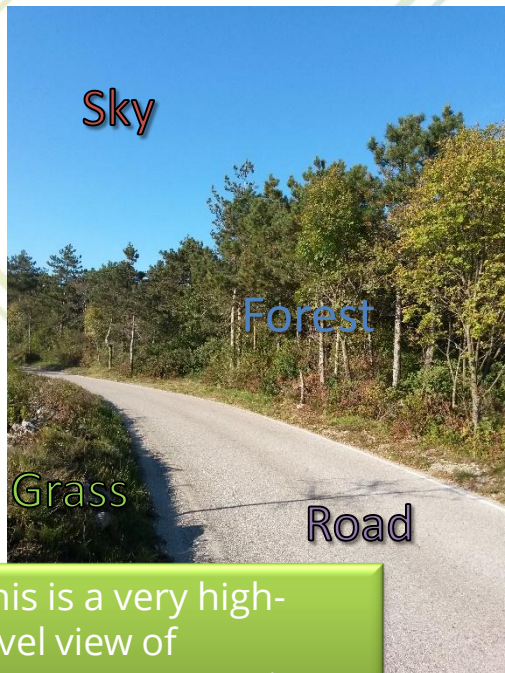


Thinning



# A few words on image segmentation

- Image Segmentation
  - a process of decomposing image into its meaningful constitutional parts, or components.



This is a very high-level view of image segmentation.



Green area

This is what we can reasonably expect



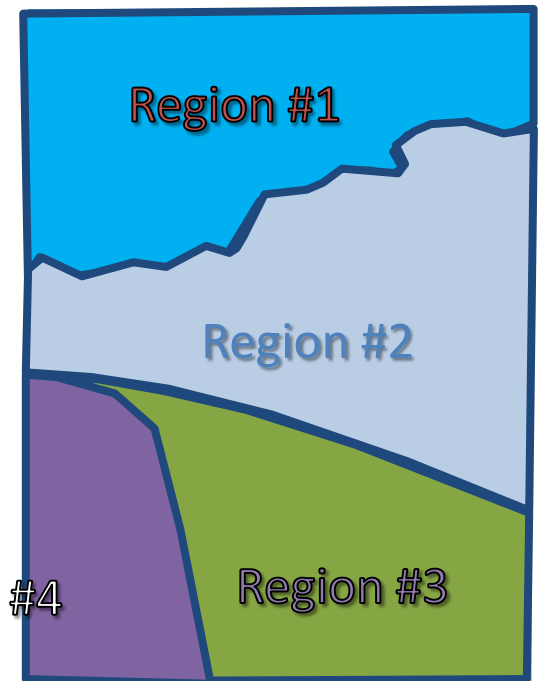
This is the result we usually get

# A few words on image segmentation

- *Part* of an image, or an image *component*, is a set of pixels that have something in common. That is, all pixel in the set share some property of interest, e.g., gray value, color, texture, appearance...



Most of the time, an image component is an area within image that is called *region*.



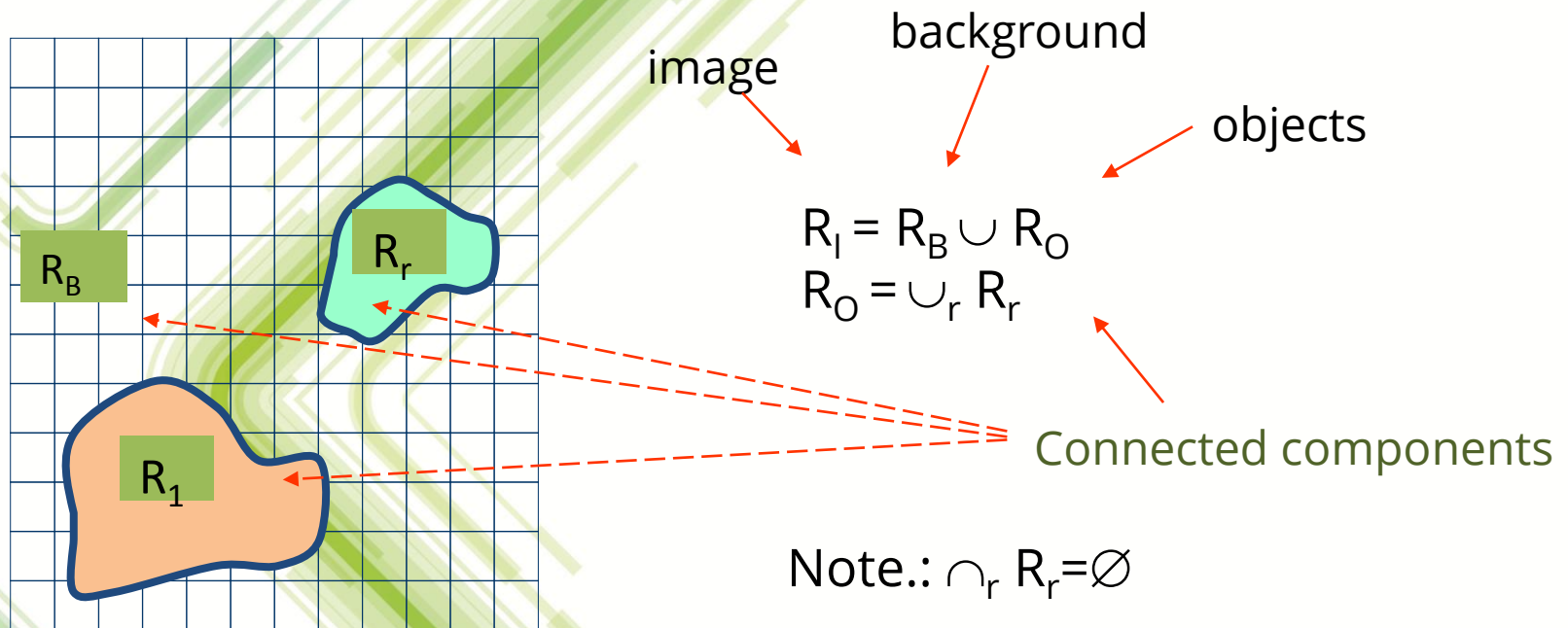


# A few words on image segmentation

- Image region
  - set of *connected* image pixels.
  - Besides being connected, these pixels share some common property, and therefore, they belong together.
- Region feature / Region descriptor
  - Property, or more of them, are sometimes called features, or descriptors.
  - Formally, region  $R_r$  is a set of pixels
$$R_r = \{ I(x_i, y_i) \mid P(I(x_i, y_i)) = P_r \}$$
  - where
    - $\forall I(x_i, y_i) \in R_r$  are connected,
    - $P$  is a suitable property mapping function,
    - and  $P_r$  is a predefined property of  $R_r$

# Images vs. regions

- An image is decomposed into regions.
  - Some regions correspond to the *objects* (objects/regions of interest)
  - The rest belong to the *background*



# Regions of interest



It is possible (depending on application), that there is only single region of interest, e.g. road in case of driving, riding, or walking, or scuba diver, and all the rest would be background



Region of interest -- ROI  
Region detection / object detection.



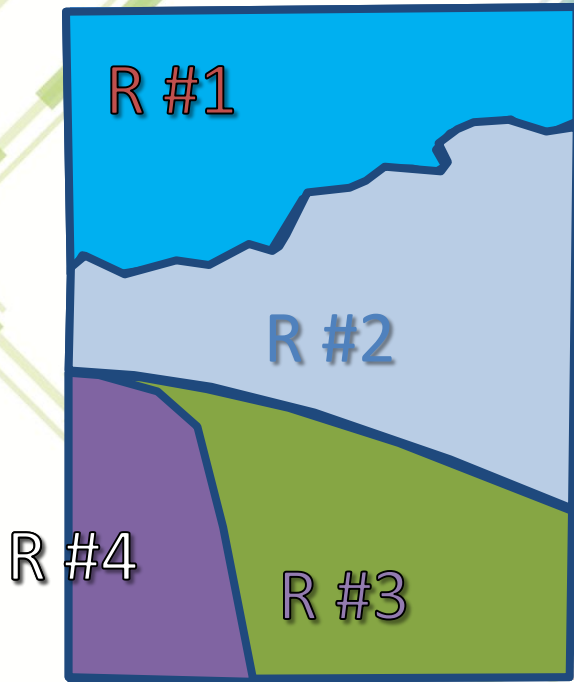
# Regions and segmentation

- Sometimes, image components are not regions.
  - For example, we can decompose an image into straight or curved line segments.
- Often, and most notably in IP community,
  - even edge, corner, as well as many other related detection tasks are treated within the subject of image segmentation.
- Nevertheless,
  - by segmentation we will refer to *region segmentation*,
  - unless explicitly stated otherwise.

# Snapshot of segmentation approaches

- Diverse segmentation techniques exists, e.g. segmentation by:
  - Thresholding, background subtraction
  - Connected component labeling
  - Split, merge, or both, split and merge (decomposition&agglomeration)
  - Graph-based techniques, hierarchical structures
  - Statistical pattern recognition, clustering
  - Model-based
  - Matching, and fitting
  - Active models, deformable contour, shape, appearance models
  - ...

# Region segmentation strategies



- Region-based segmentation
  - Segmentation techniques that explore the notion of region directly
  - This way we identify (label) all pixels that belong to region(s).
- Alternatively
  - we can describe region(s) by boundary curve(s) or bounding contour(s) and perform contour-based segmentation.
  - This way we identify only pixels on the border of region(s).



# Region based segmentation

- Region
  - defined by the pixels that belong to that region.
  - We look for connected sets of pixels and label them.
  - 2 approaches:
- Region growing
  - start with a pixel (,seed') and adjoin neighboring pixels (connected component labelling, or so-called ,coloring')
- Split and merge
  - split what does not fit together, and merge

# Connectivity

- But, what does it mean ,to be connected'?
- Two pixels within region, or two (sub)sets of pixels, are *connected*, if there exists *at least one path* (a sequence of *pairwise adjacent* pixels) that are all in that region.

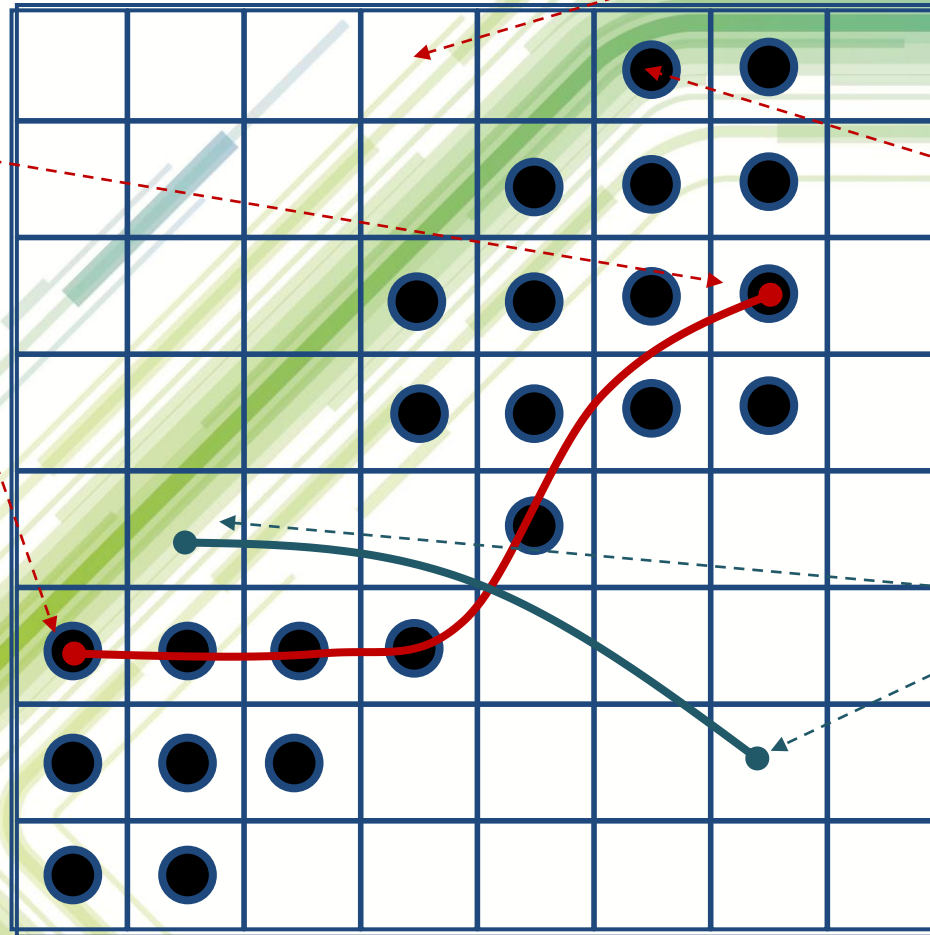
# Connectivity

Are these  
two pixels  
connected?

Background  
pixel

Object pixel  
(foreground)

Are these  
two pixels  
connected?

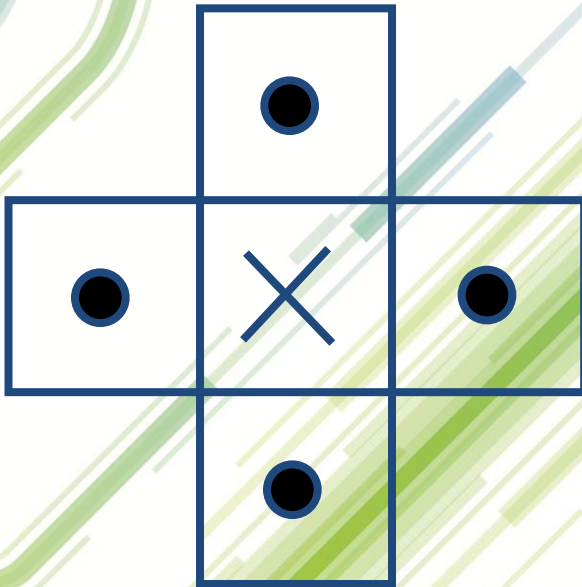


How many background regions are there?  
How many object regions are there?

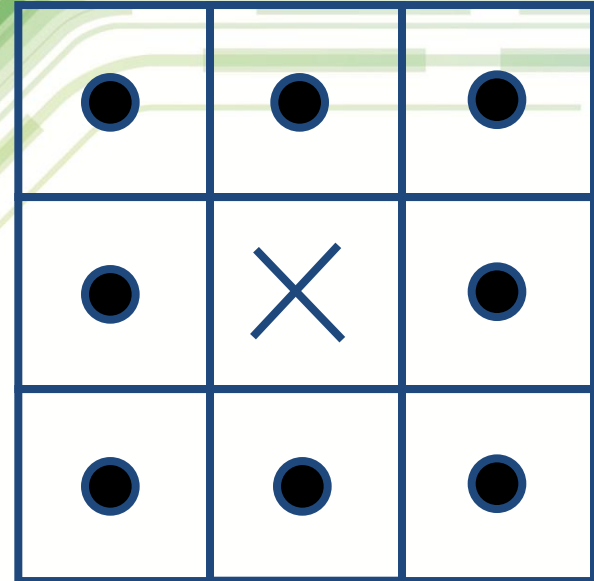



# Connectivity-definition

4-connected



8- connected



- *Neighbors* – directly connected (adjacent) pixel(s) of 
- *Neighborhood* – the set of neighbors
- If foreground is 8-connected, then background is 4-connected, and vice versa.

# Connected components labeling

- Also called region growing
  - Raster scan the image (because it is easy, and systematic) to find the first non-background pixel (*seed*) in the image, label it
  - Repeat
    - Test (visit) its neighbors
    - if neighbor pixel is found that *belongs to the object's region*, label it, move to that pixel, and continue.
  - Thus, all connected pixels are labeled with the same value, i.e. ,color'.
- Typical recursion, sometimes called ,flooding' or coloring.
  - If adjacent pixel shares same (or similar) property, color it.

# Connected components labeling

- More on *“if pixel belongs to that region”*
- There are many ways to define this property, e.g. *similarity*, or *distance*, or *homogeneity*, that is used to evaluate the membership.



# Labeling, flooding, coloring - algorithm

```
Coloring( x, y ); // first pixel (x,y) is called ,seed point'  
{  
  Color(x,y); // e.g. Image[x, y] = 255; (255 must be unique)  
  if condition( Image[ x, y-1] ) Coloring( x, y-1 );  
  if condition( Image[ x, y+1] ) Coloring( x, y+1 );  
  if condition( Image[ x-1, y] ) Coloring( x-1, y );  
  if condition( Image[ x+1, y] ) Coloring( x+1, y );  
}
```

- Important points
  - Definition of connectivity, which one applies in this case?
  - *condition* for coloring, how to define it.
  - *Recursion* needs sufficient stack space, could overflow.
- Matlab: `imfill()`, `bwlabel()`, `bwconncomp()`, ...

# Labeling, flooding, coloring – another approach

- Should solve “U case”.

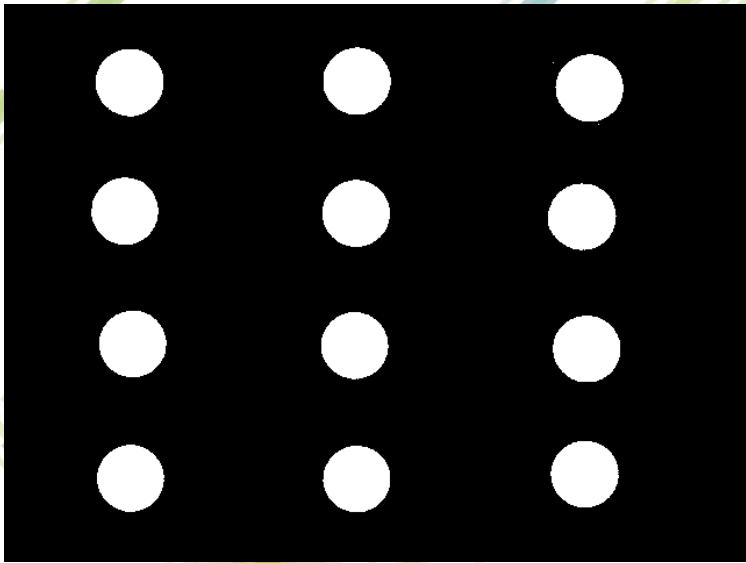
```
X X . . . . X X .  
X X X . . . X X X .  
X X X X . X X X X X  
X X X X X X X X X X  
X X X X X X X X X X
```



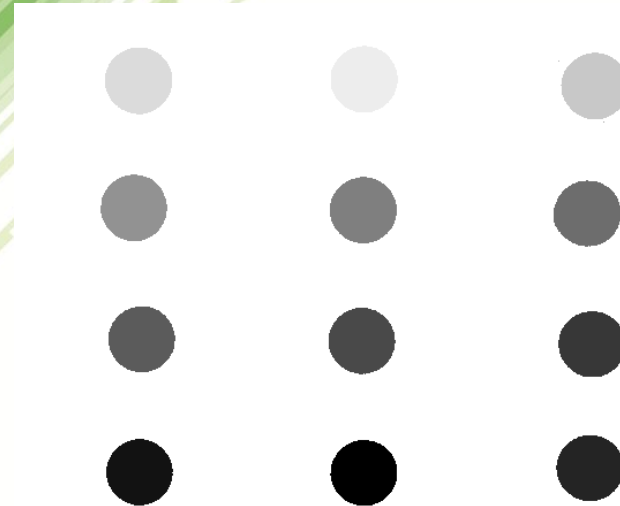
```
1 1 . . . . 2 2 .  
1 1 1 . . . 2 2 2 .  
1 1 1 1 . 2 2 2 2 2  
1 1 1 1 2 2 2 2 2 2  
1 1 1 2 2 2 2 2 2 2
```

- General approach:
  - scan the image left to right, top to bottom
  - if pixel probably belongs to new region, label it with new label
  - If one or more neighbors were already equally colored, use that label to label the pixel
  - if there are more than one of the neighbors labeled differently, label the pixel and equate the labels.

# Simple example



Input binary picture



Labeled object regions

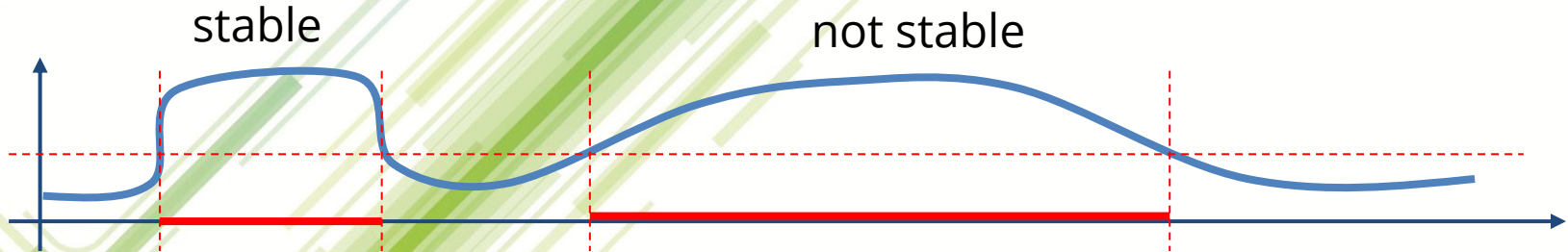


# Region-based segmentation

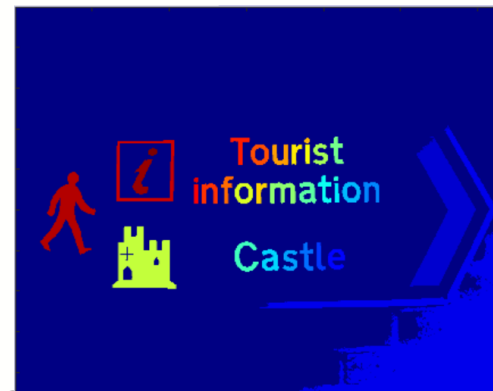
- A simple and effective approach to segmenting an image
  - by thresholding and then labeling object regions.
  - (Another related approach is background subtraction)
- But, usually, there is no single threshold that would work for all regions within image.
- Solution:
  - Use multiple thresholds
  - But, how to select them?

# MSER algorithm – for illustration only

- MSER – Maximally Stable Extremal Regions
  - Threshold image with threshold as parameter,  $t = 0, 1, 2, \dots, 255$ . Find regions that are most stable with respect to the threshold change.



Input image



Color coded extremal regions

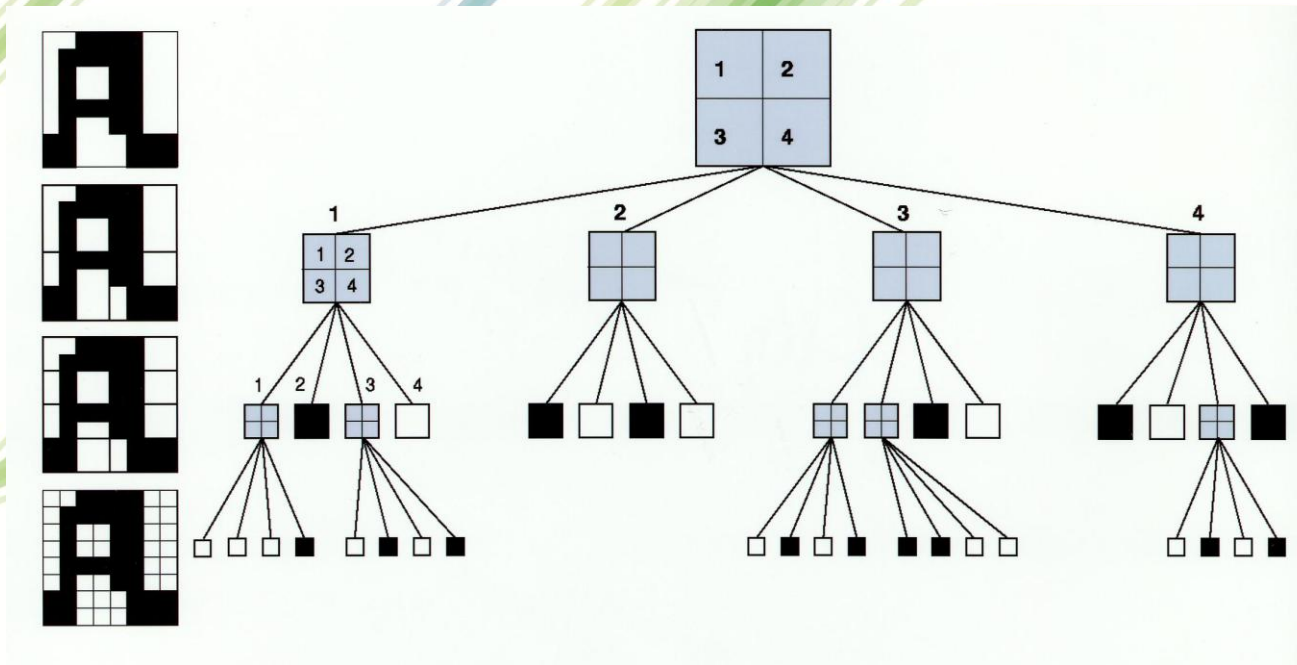
# Split & Merge

- The basic idea is quite simple:
  - First, define homogeneity criteria (grey value, color, texture, etc.)
  - Start with an apriori division of image into regions (could also start with a whole image as a region)
  - If homogeneity criterion within region is not satisfied, split
  - If homogeneity criterion of two regions is satisfied, merge
  - Iterate
- One way to implement split&merge is a so-called quad-tree



# Split & Merge - illustration

- Illustration based on quad-tree



# Mean shift

- An old idea for finding modes (maxima) of pdf (probability density function)
- Basic principle:
  - start from an initial pixel
  - define an (arbitrary) region around it
  - compute the mean (center position, i.e. centroid) within region
  - move to the centroid position
  - repeat until convergence, i.e., the maximum (“mode”) is found.

# Mean shift

- Segmentation:
  - You can start from many different initial positions.
  - Many will end up in the same solution (mode).
  - This is a ,basin of attraction' that defines a single region.
  - Others will end in other modes. Each mode defines an image region.

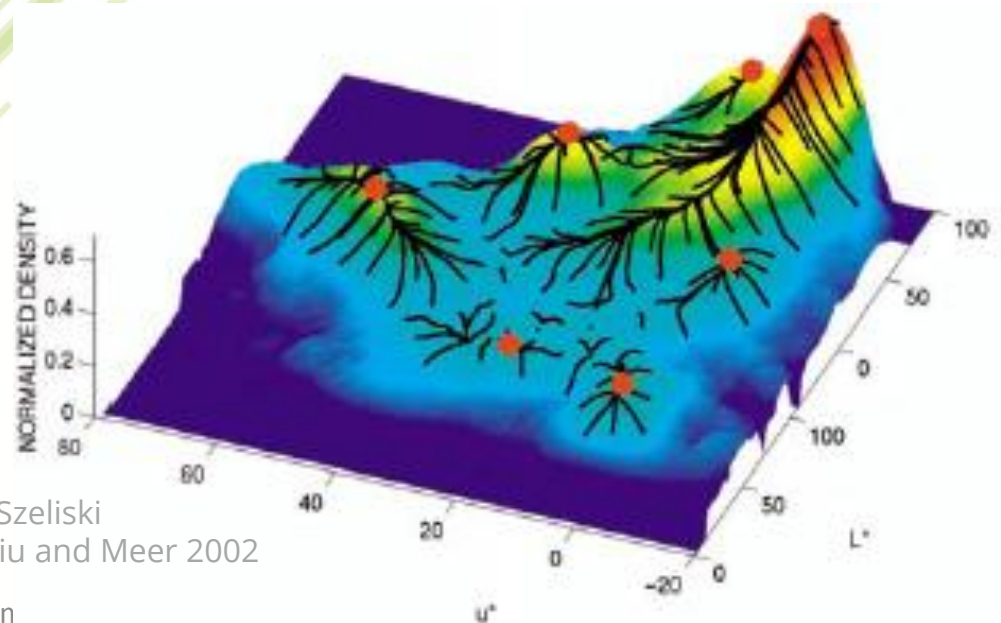


Image source: book R. Szeliski  
From authors Comaniciu and Meer 2002



# Morphological filtering

- Basic operations:
  - erode / shrink
  - dilate / expand
  - open (opening)
  - close (closing)
  - hit or miss
  - thinning, skeletonization
  - and others, (tophat, bottomhat, distance transform, watershed, ...)
- Various fields of use:
  - microscopy, medicine, industrial vision, and others.

# Morphological operations

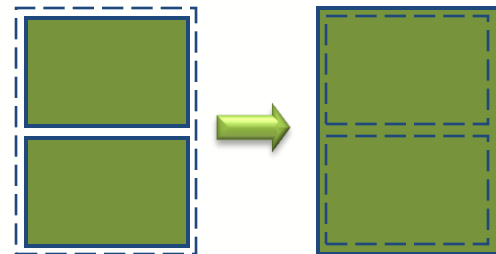
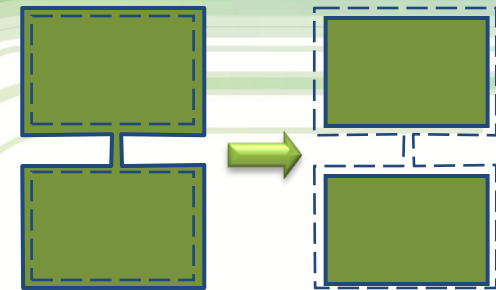
- Basic operations:

- Erode

- Regions get smaller
    - Could disconnect
    - Small regions could disappear
    - Matlab: `imerode()`, `bwmorph()`

- Dilate

- Regions become larger
    - Could join with other region
    - Holes can close – disappear
    - Matlab: `imdilate()`, `bwmorph()`

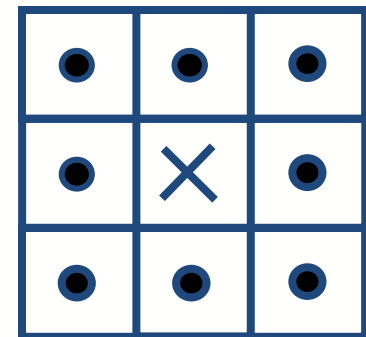
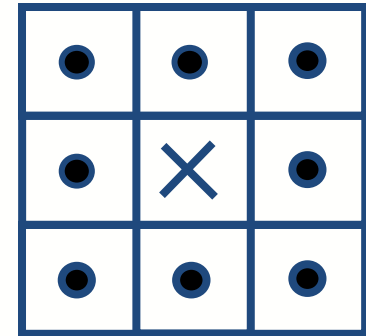


Note:

- eroding object means dilating background and the opposite.
- erode acts on object's pixels, dilate acts on background pixels.

# Morphological operations

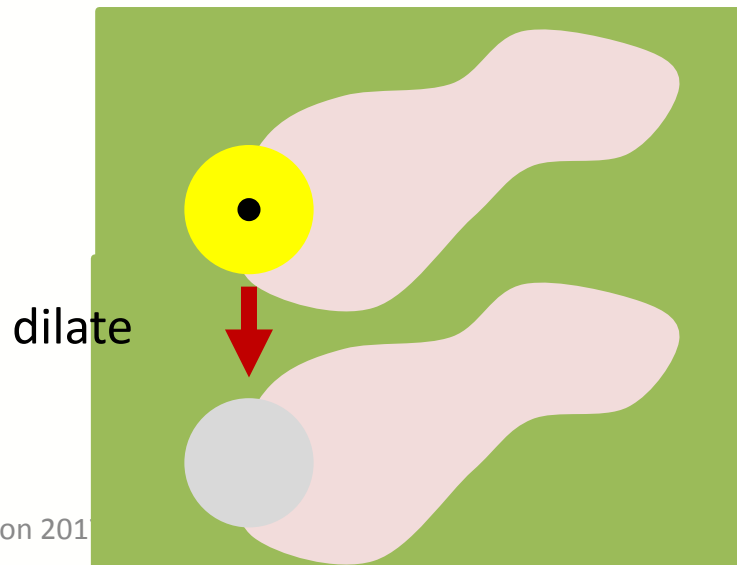
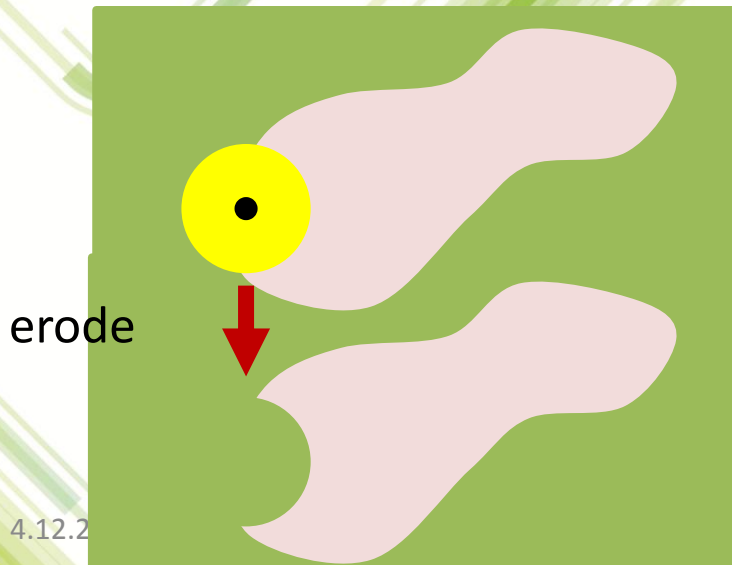
- The bases of all morphological operations are *structuring elements*.
- Similar to *convolution*, except that define (include / exclude) pixels that are involved in operations.
- Informal definition of erode
  - If at least one of the neighbors belongs to the background, then erode the central pixel. Thus, it becomes part of the background, (*acts on objects*).
- Informal definition of dilate
  - If at least one of the neighbors belongs to the object, then dilate the central pixel. Thus, it becomes part of the object, (i.e., *acts on background*)





# Morphological operations

- One can think of structuring element as a paint brush of suitable size.
- Using that paint brush (structuring element) one can either apply “paint” to cover background (dilate) or to remove pain from an object (erode).
- By applying a paint brush to an image, one can then dramatically change the shape of a region.



# Erosion – formal definition

$$E = I \ominus SE = \{ p \} \subseteq I$$

$I$  binary input image

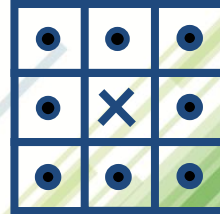
$SE$  structuring element

$E$  eroded binary output image

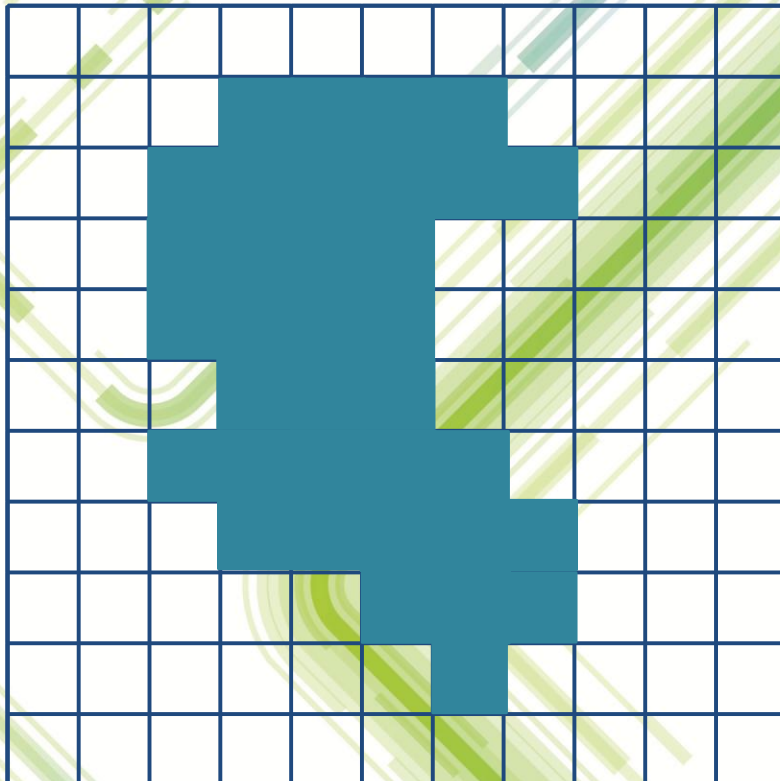
$E$  is a set of pixels  $p$  such that  $SE$  translated to  $p$  overlaps entirely only with foreground (object) pixels in  $I$ .

# Illustration of erosion

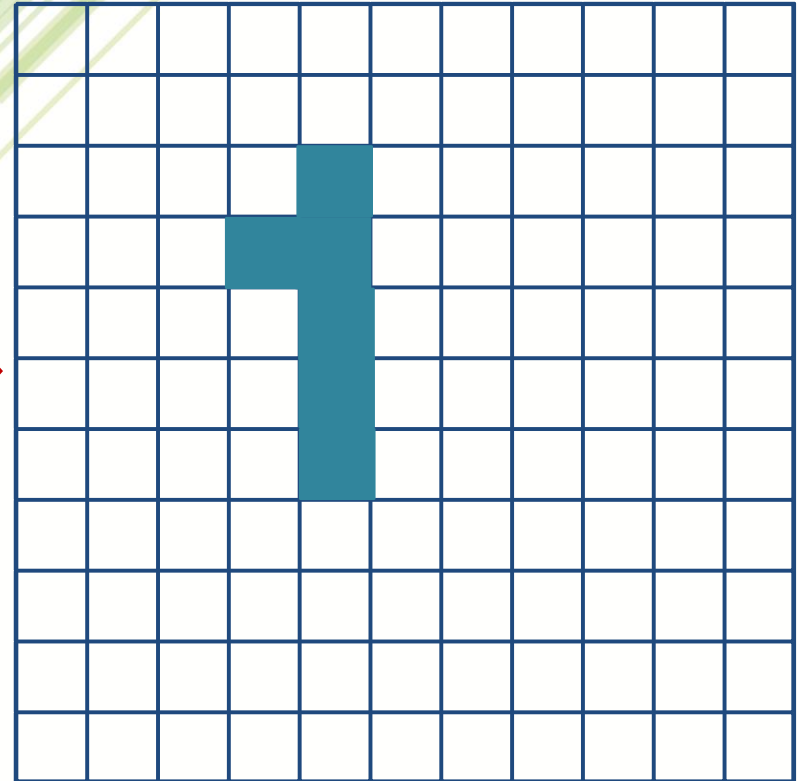
SE



Input image



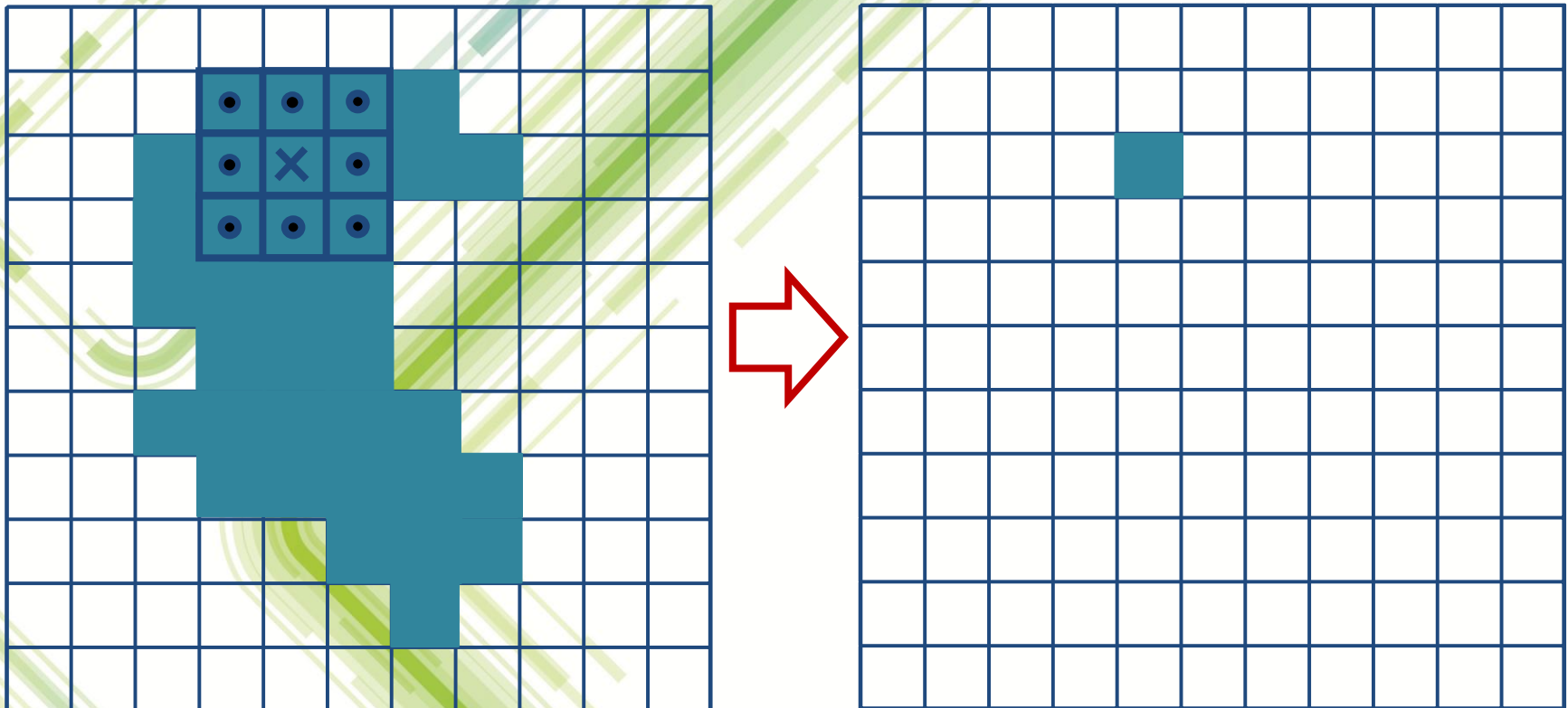
Eroded image



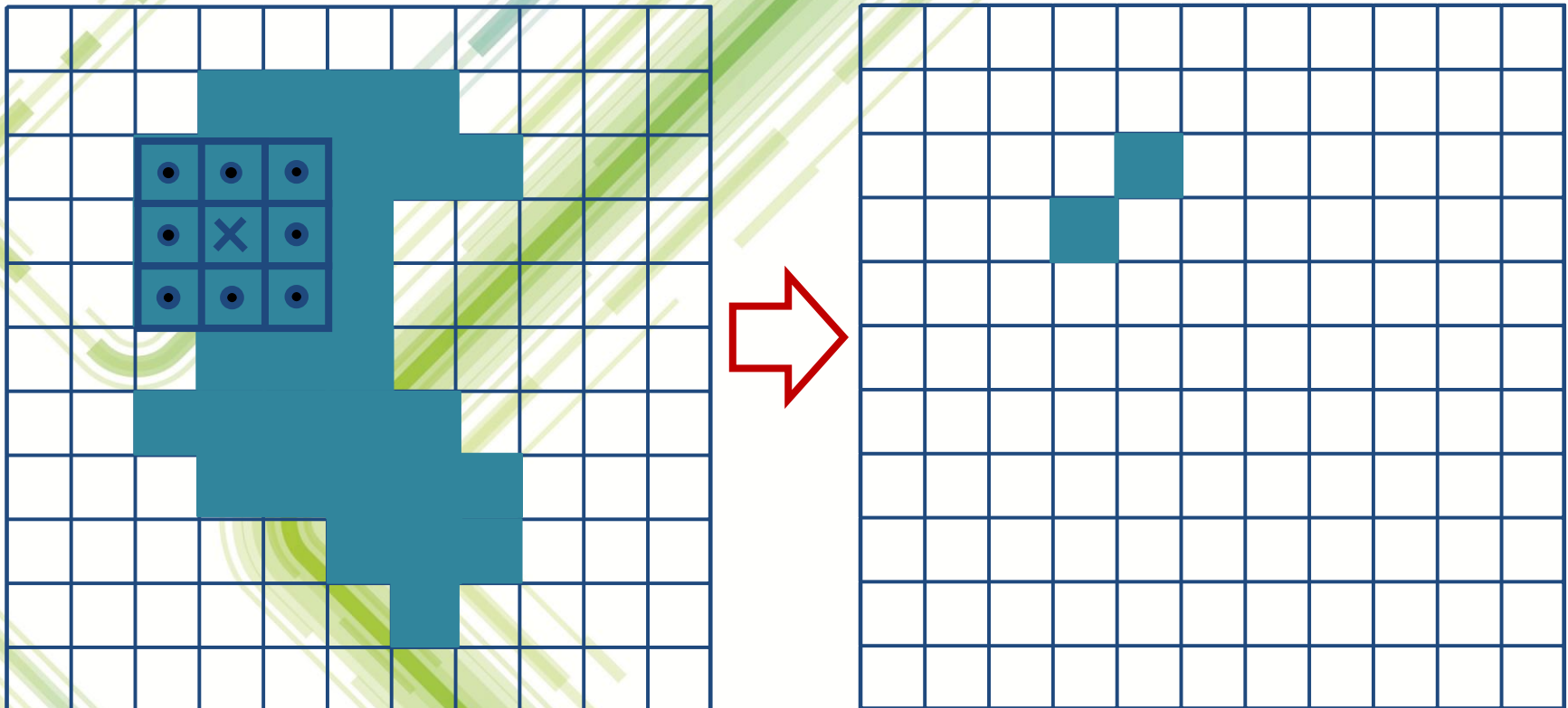


# Illustration of erosion

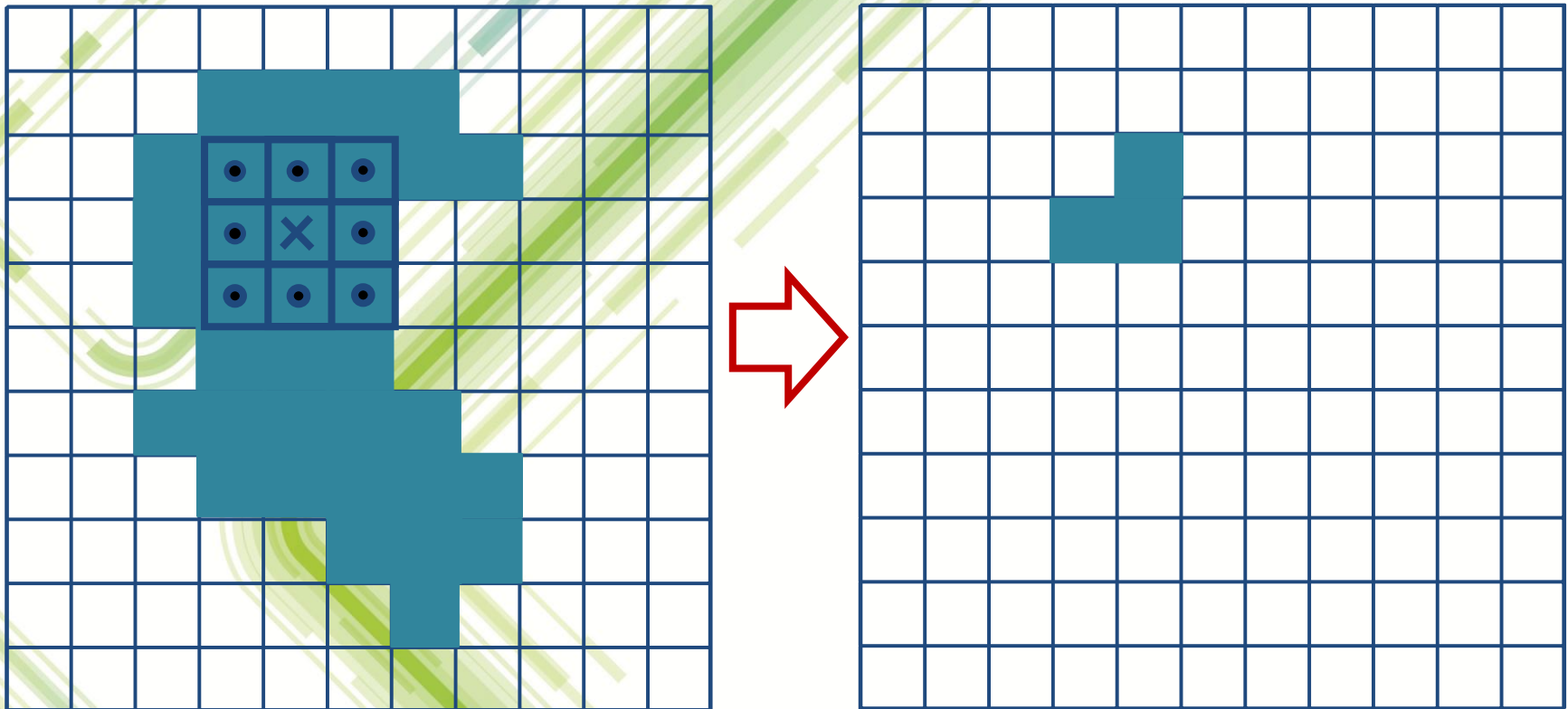
Raster scan the image, e.g., left to right, top to bottom.  
This is the first pixel that is not eroded.



# Illustration of erosion

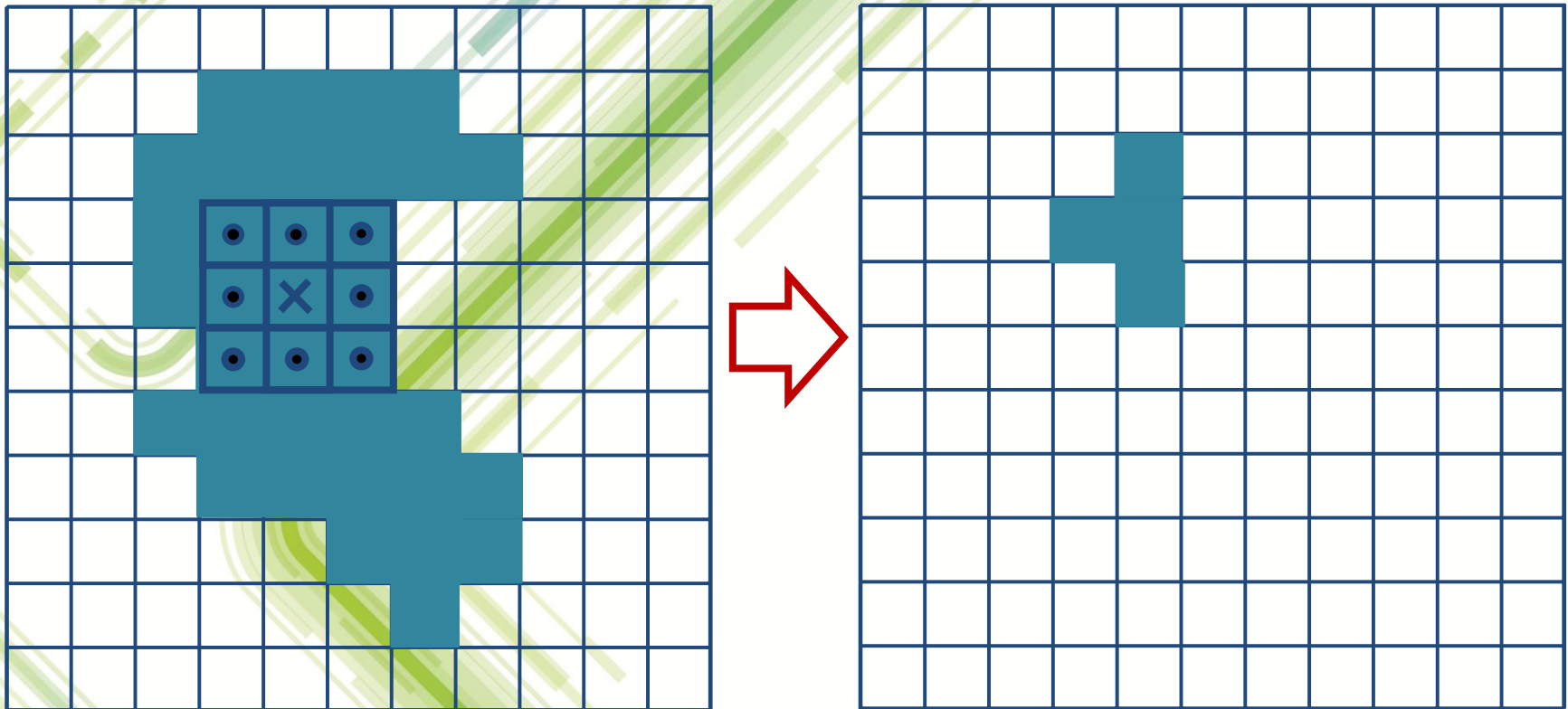


# Illustration of erosion

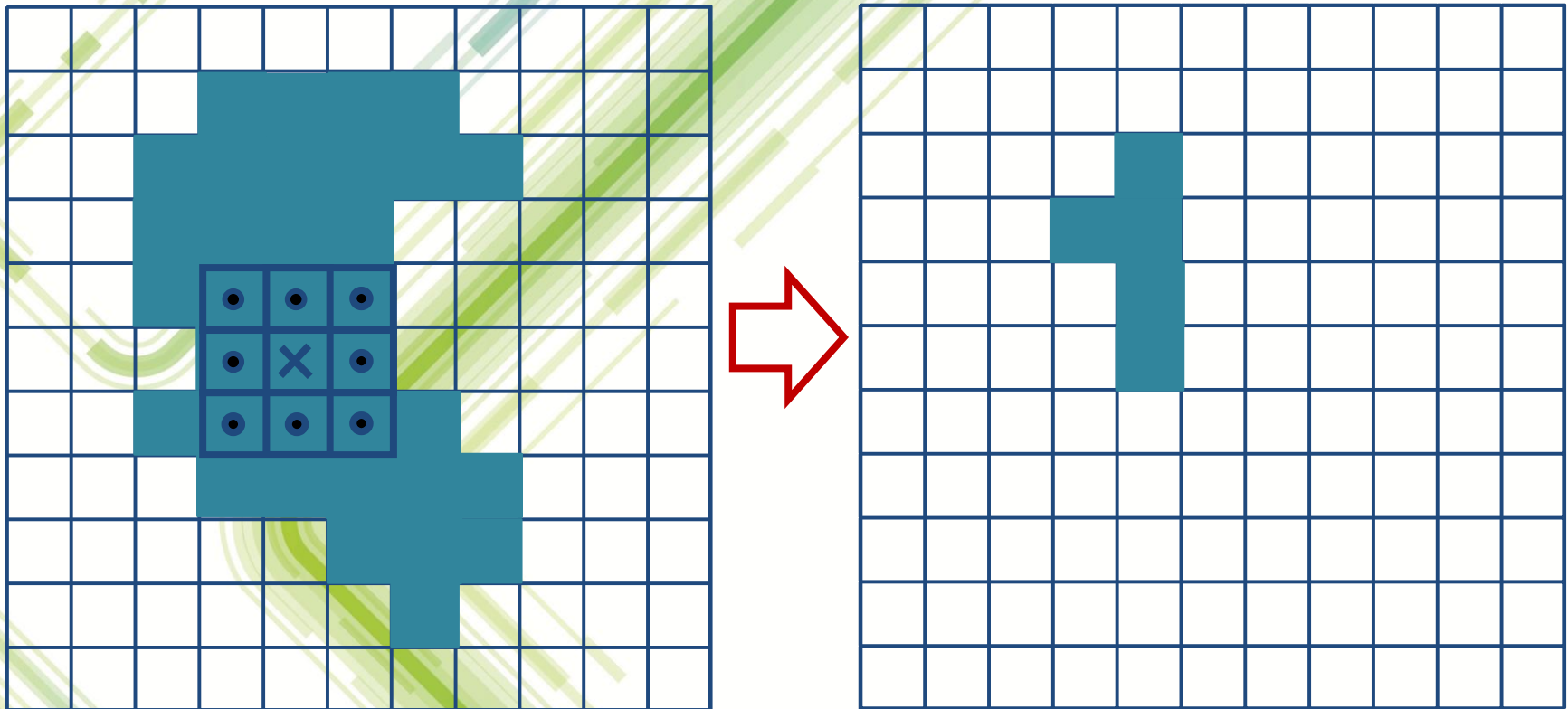




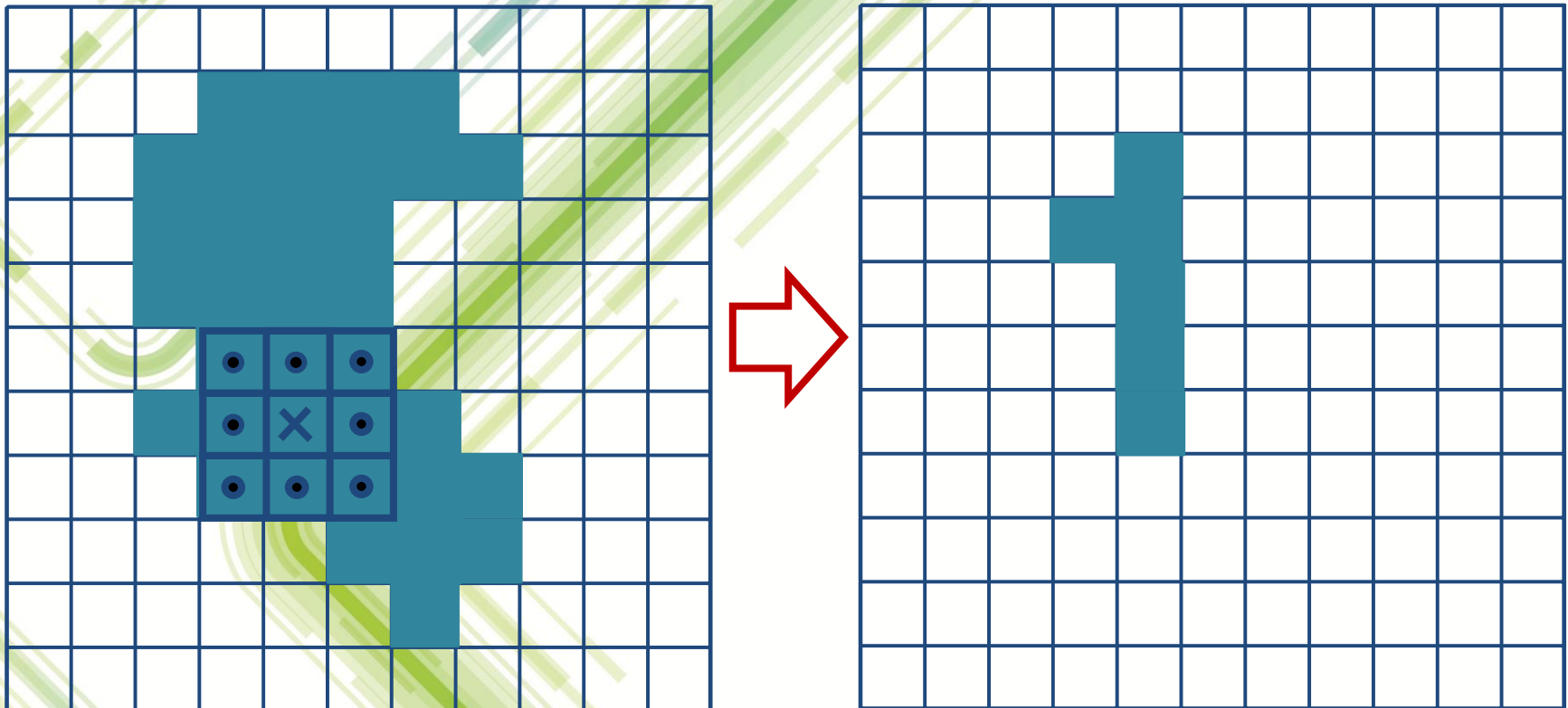
# Illustration of erosion



# Illustration of erosion



# Illustration of erosion





# Dilation – formal definition

$$D = I \oplus SE = \{ p \} \subseteq I$$

$I$  binary input image

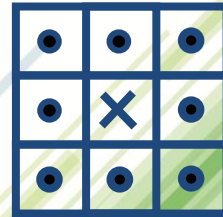
$SE$  structuring element

$D$  dilated binary output image

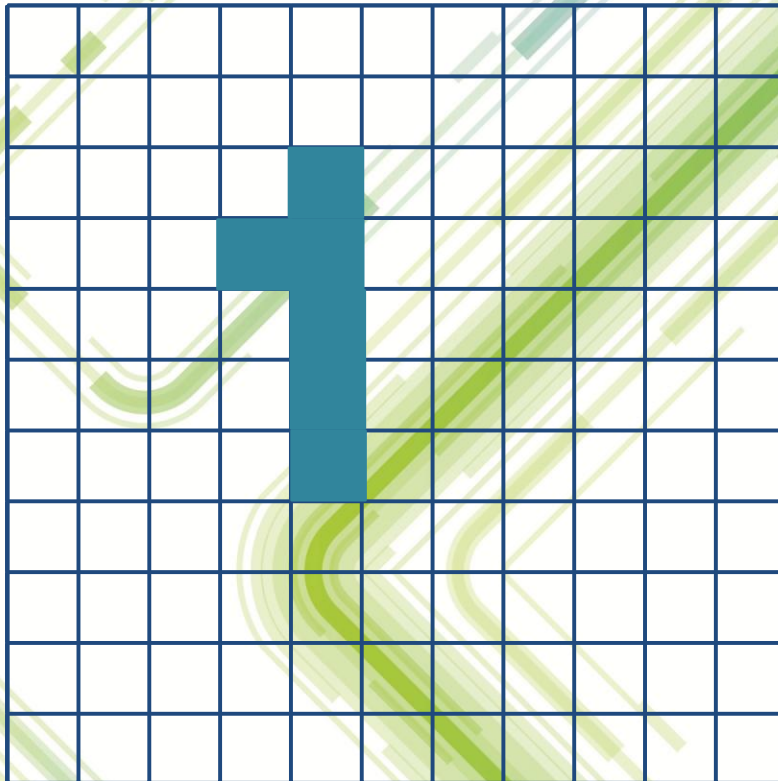
$D$  is a set of pixels  $p$  such that  $SE$  translated to  $p$  overlaps with at least one foreground (object) pixels in  $I$ .

# Illustration of dilation

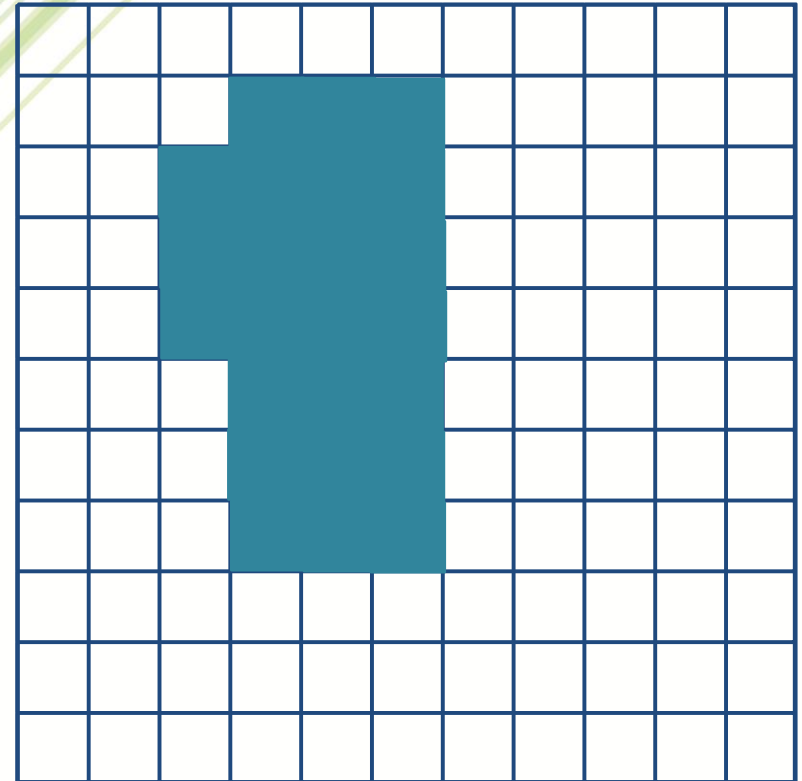
SE



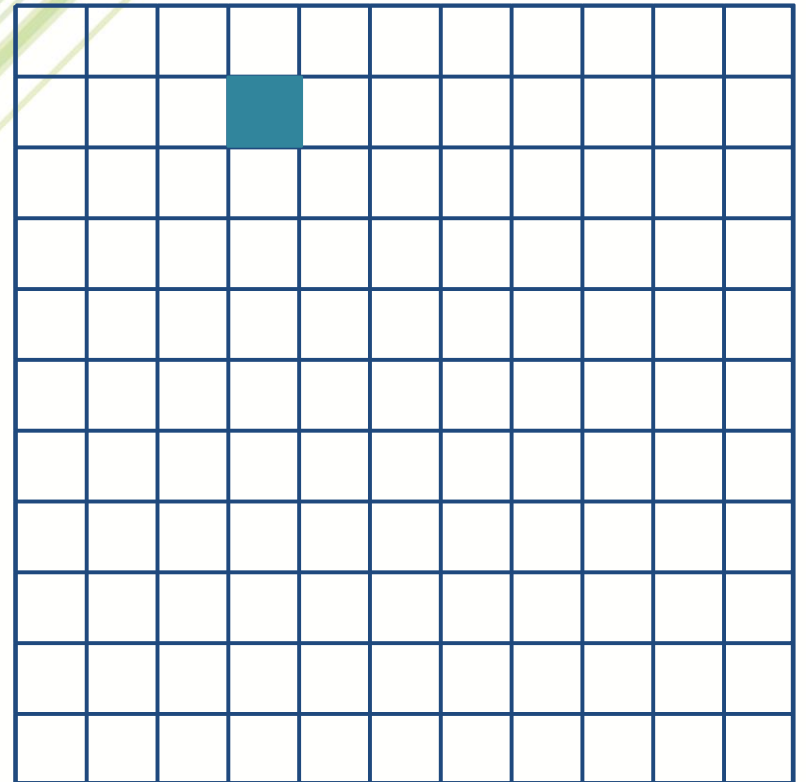
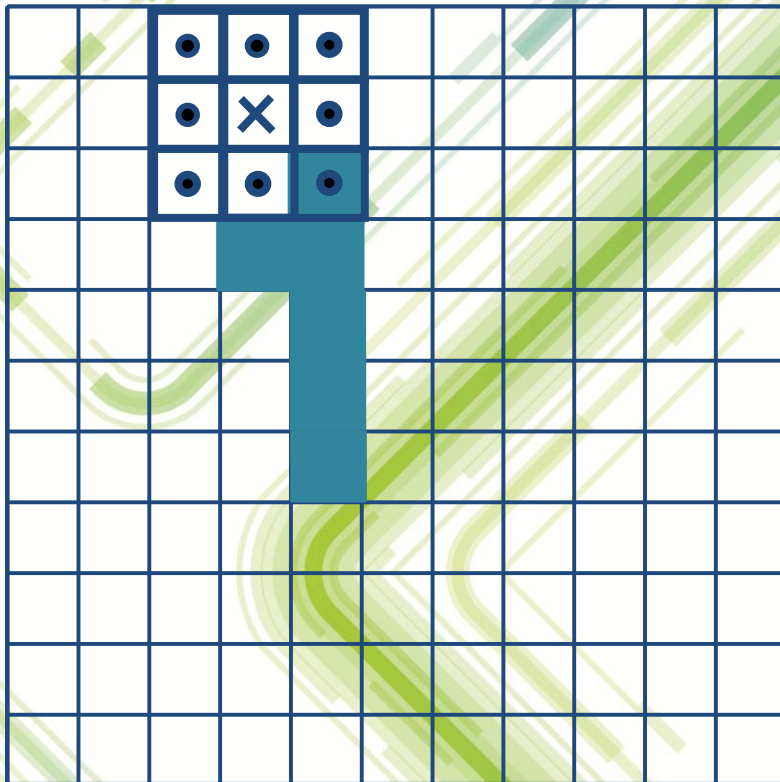
Input image



(final) dilated image

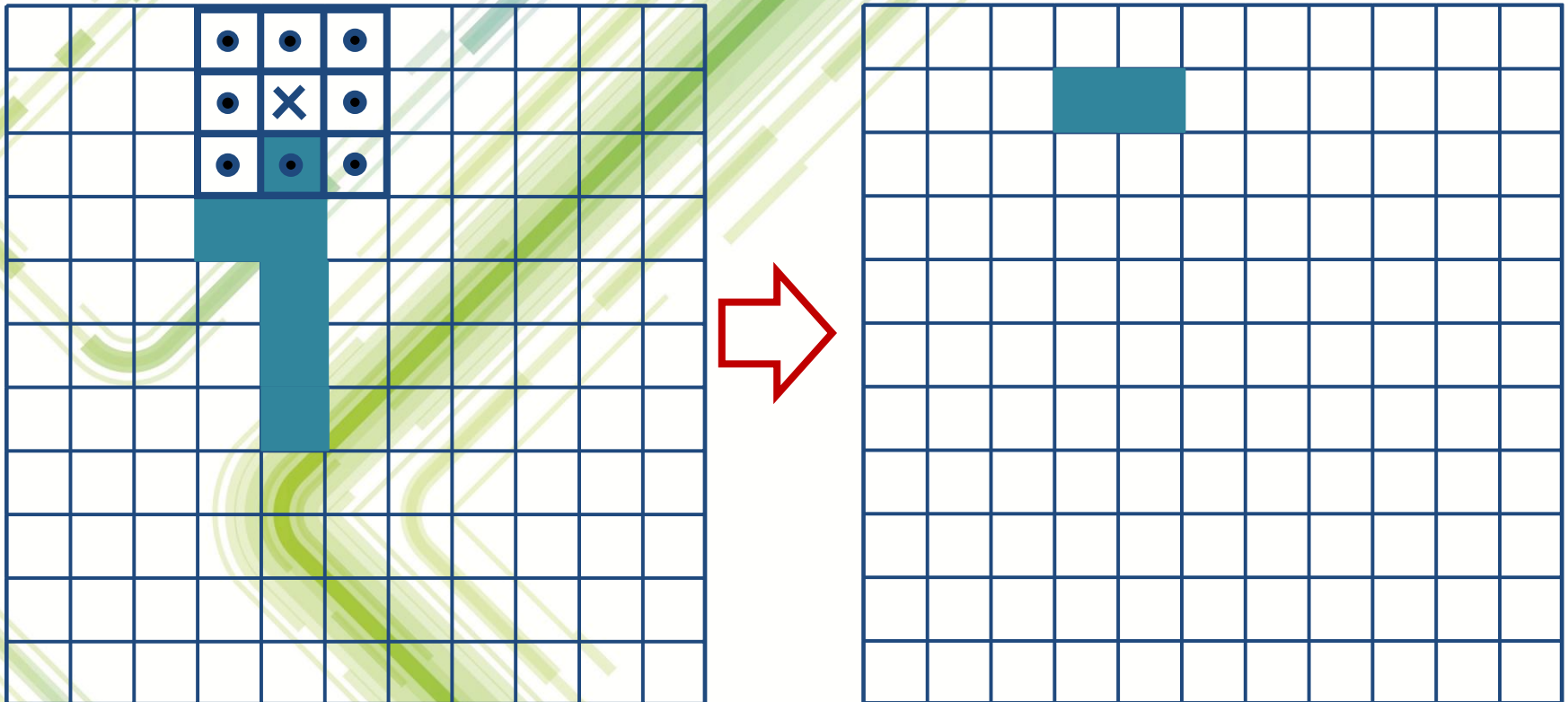


# Illustration of dilation

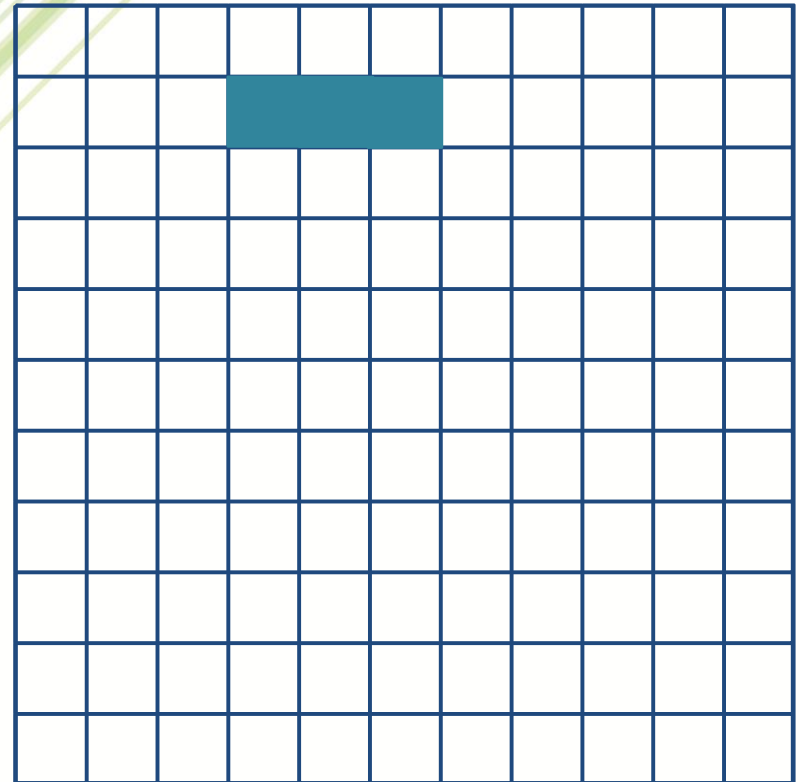
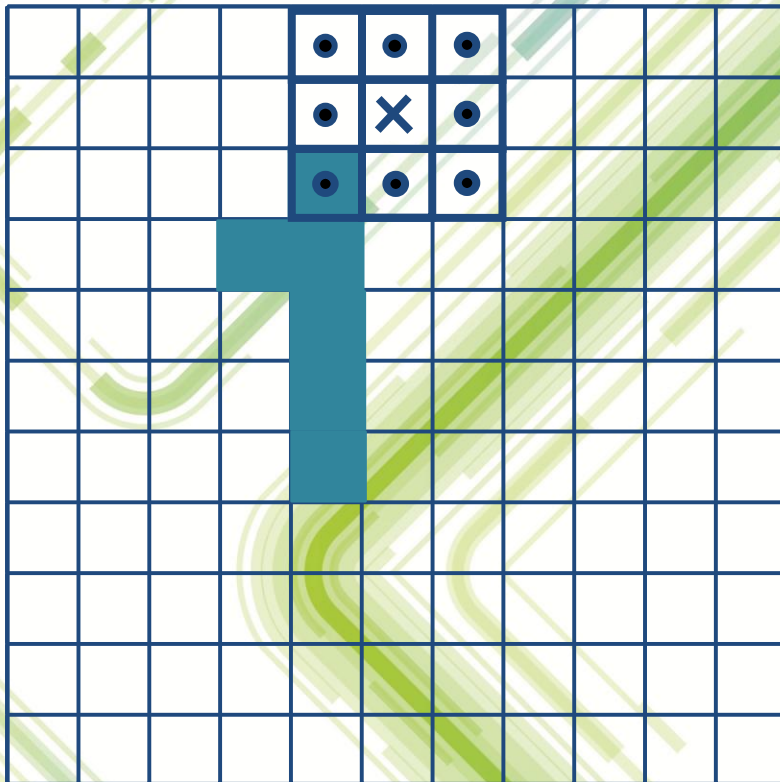




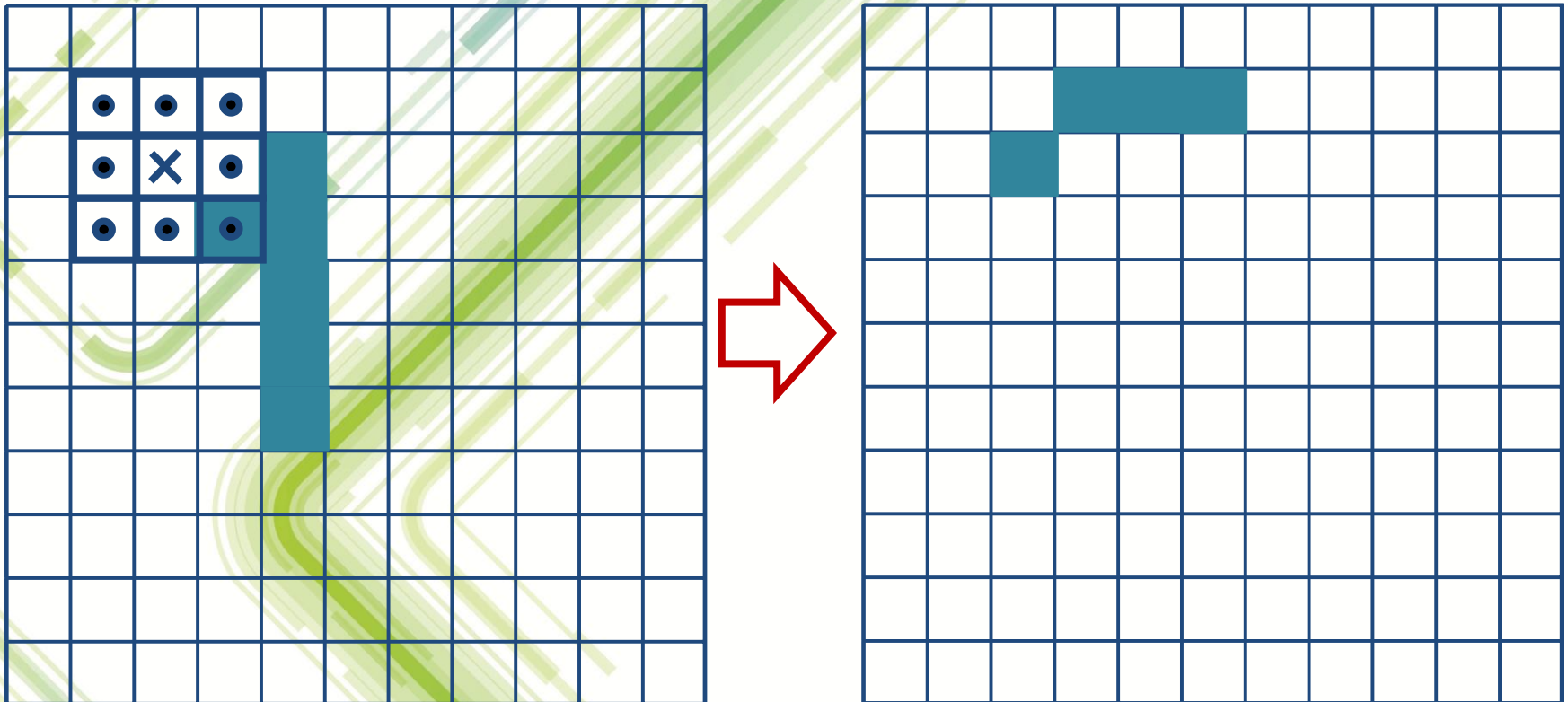
# Illustration of dilation



# Illustration of dilation



# Illustration of dilation



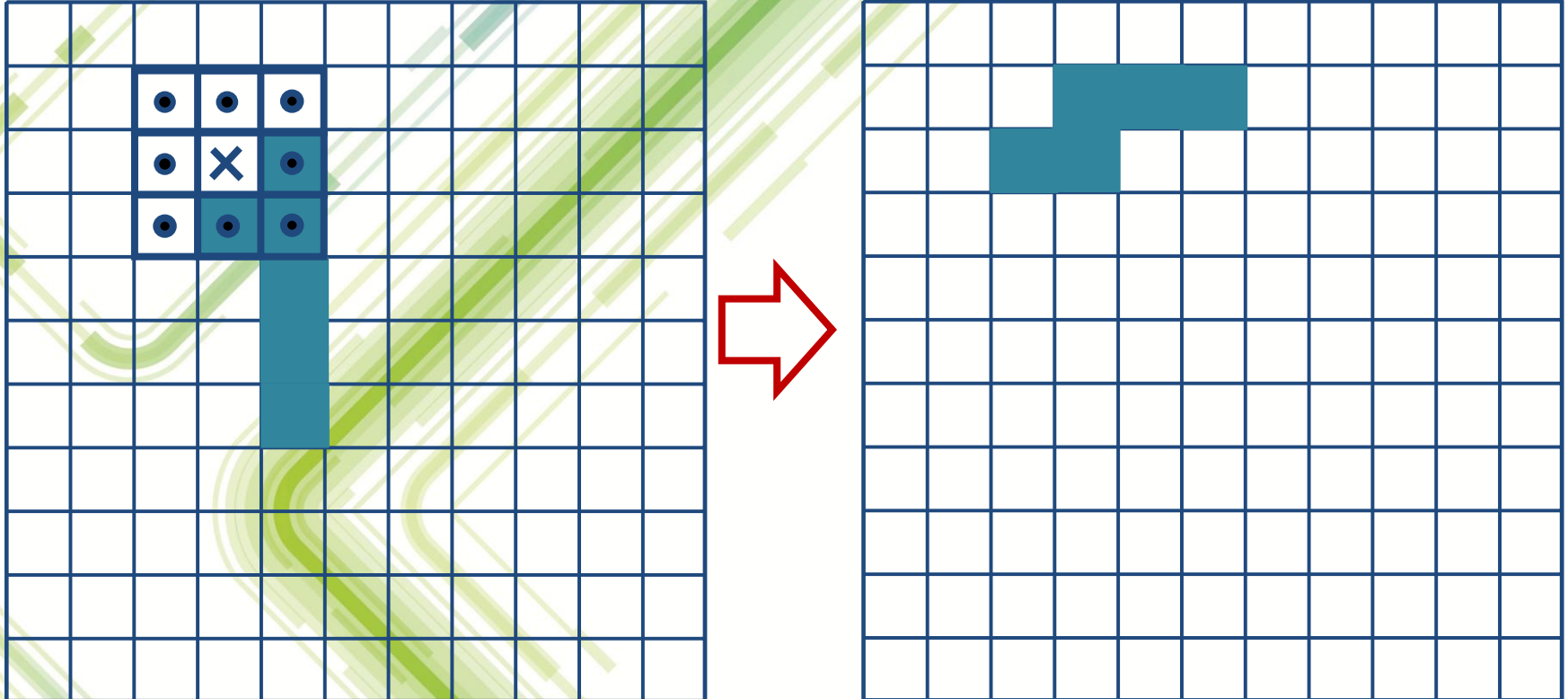


# Illustration of dilation

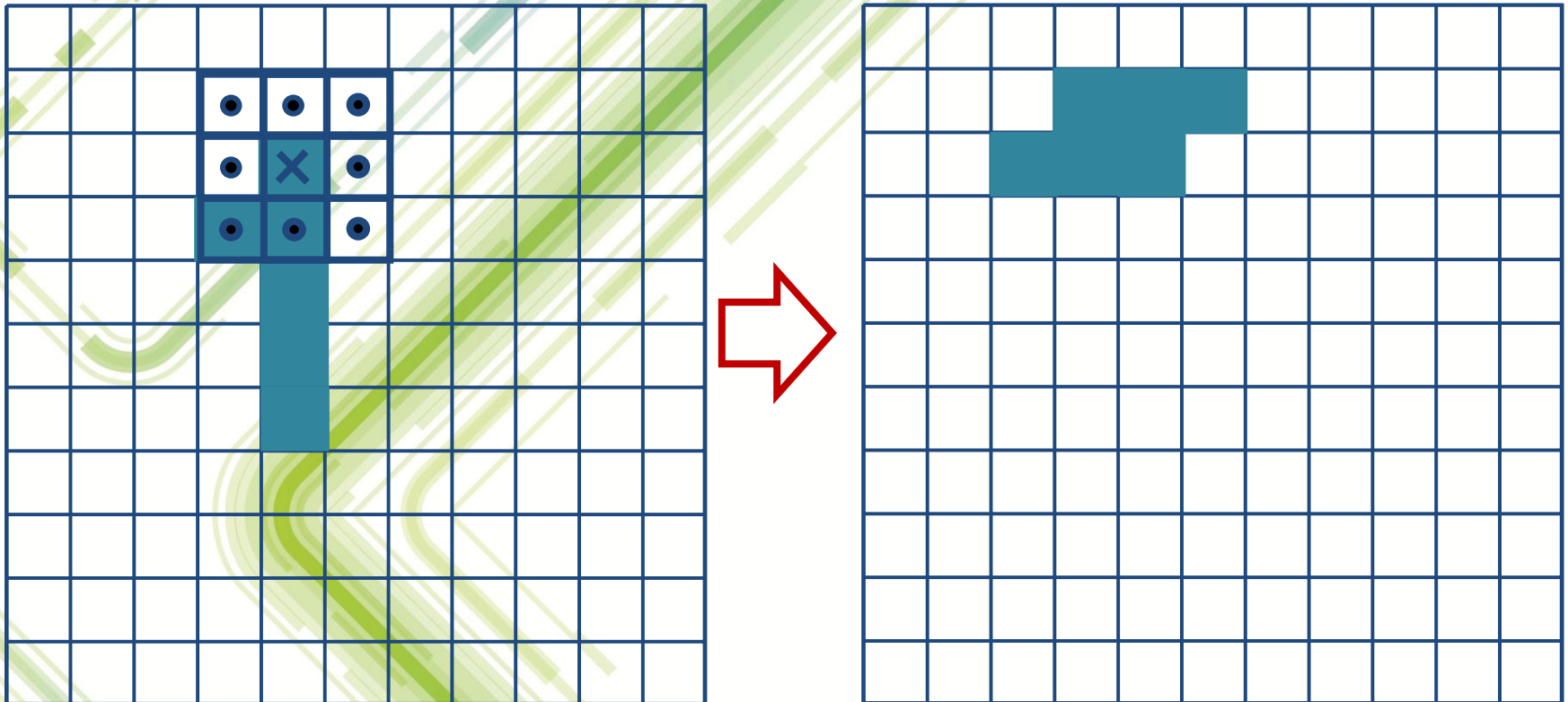
The diagram illustrates the process of dilation on a 10x10 grid. On the left, a 3x3 kernel is shown with a center 'X' and blue dots at (2,2), (2,3), (2,4), (3,2), (3,3), (3,4), (4,2), (4,3), and (4,4). A red arrow points to the right grid, which shows the result of dilating the kernel by a factor of 2, resulting in a 6x6 blue shape.

Grid	Row	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10
Input Grid	1										
	2			•	•	•					
	3		•	X	•						
	4		•	•	•						
	5										
	6										
	7										
	8										
	9										
	10										
Output Grid	1										
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										
	10										

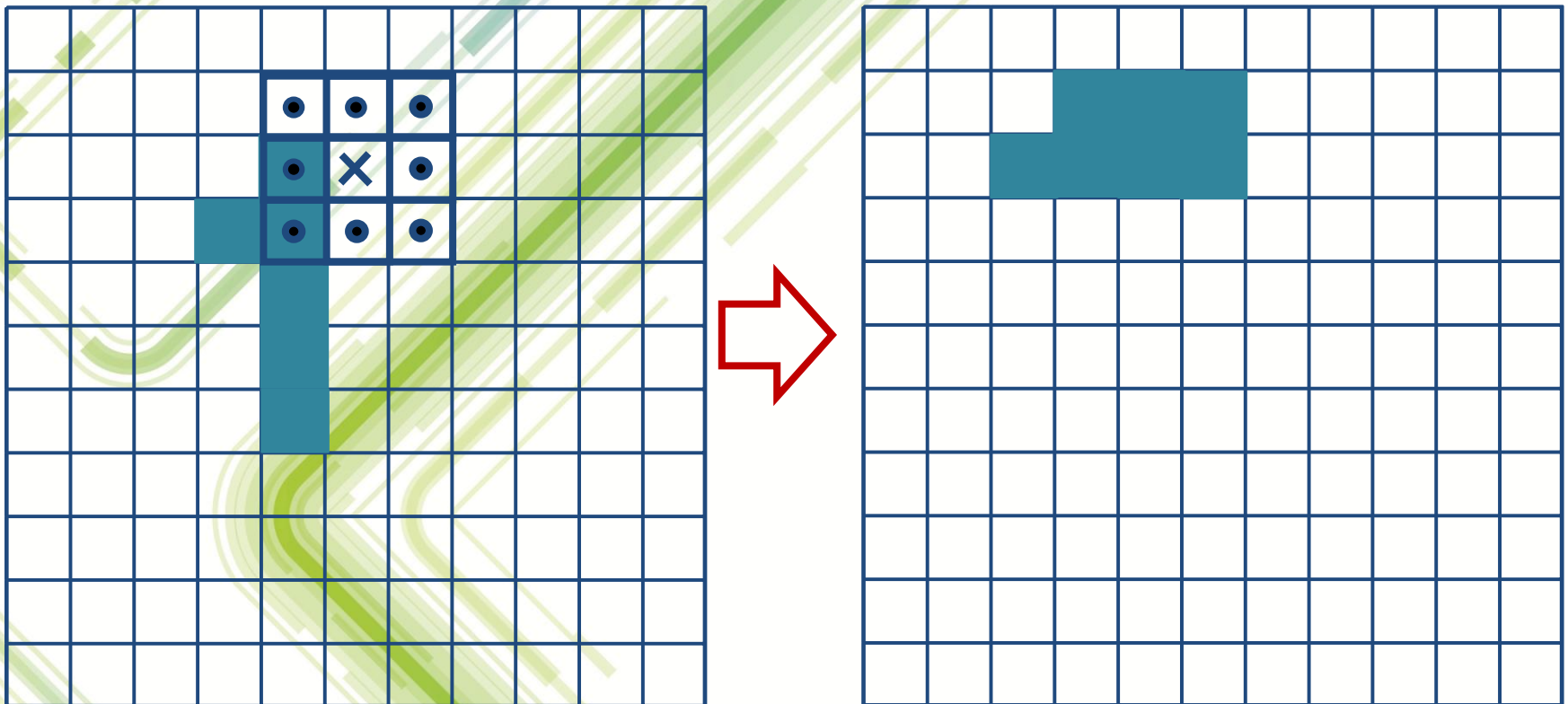
4.12.2018 Computer Vision 2017/2018 45



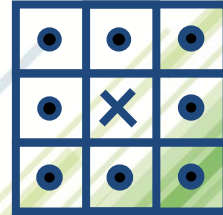
# Illustration of dilation



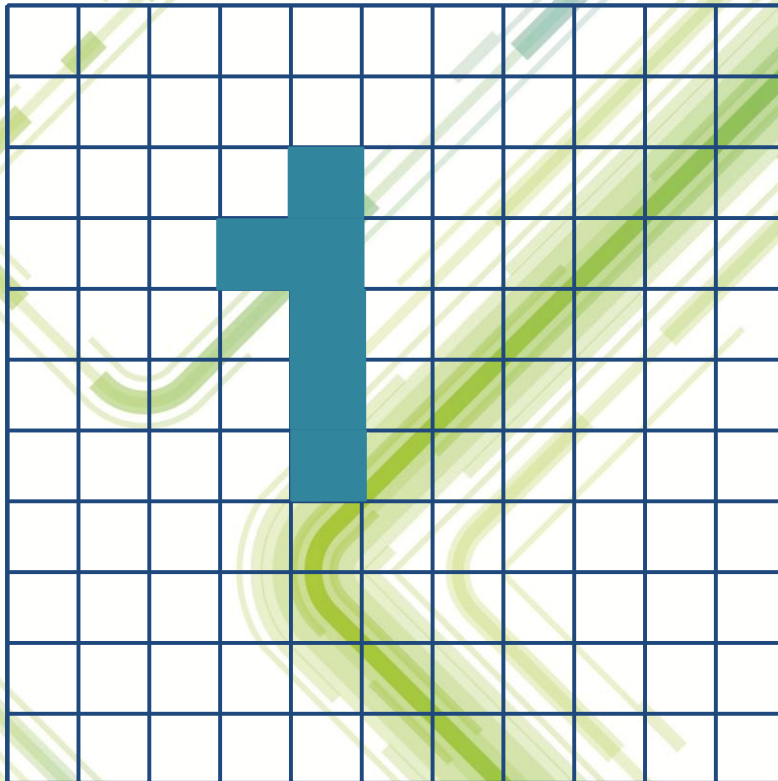
# Illustration of dilation



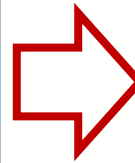
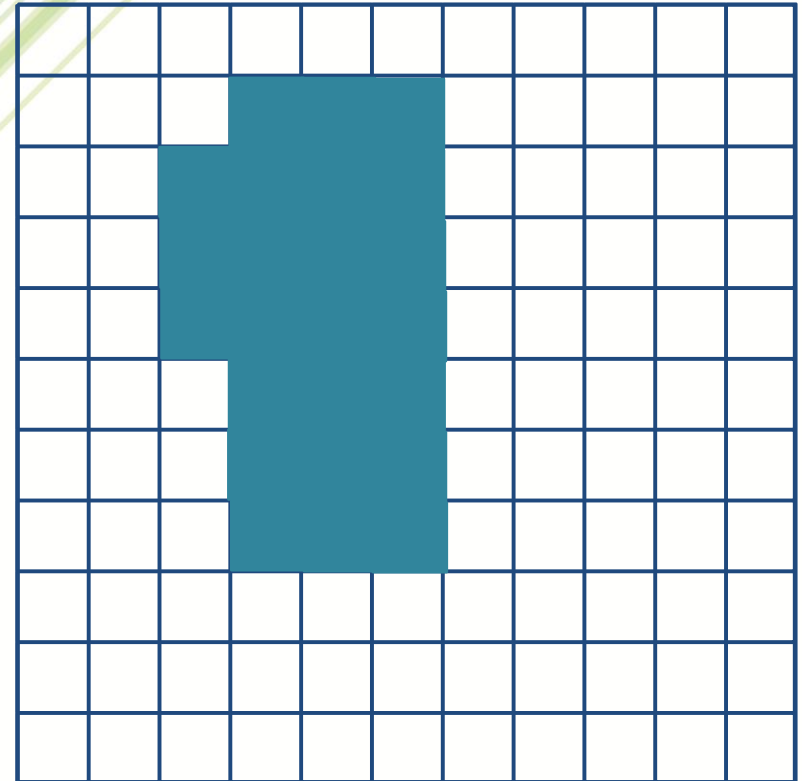
# Illustration of dilation



Input image



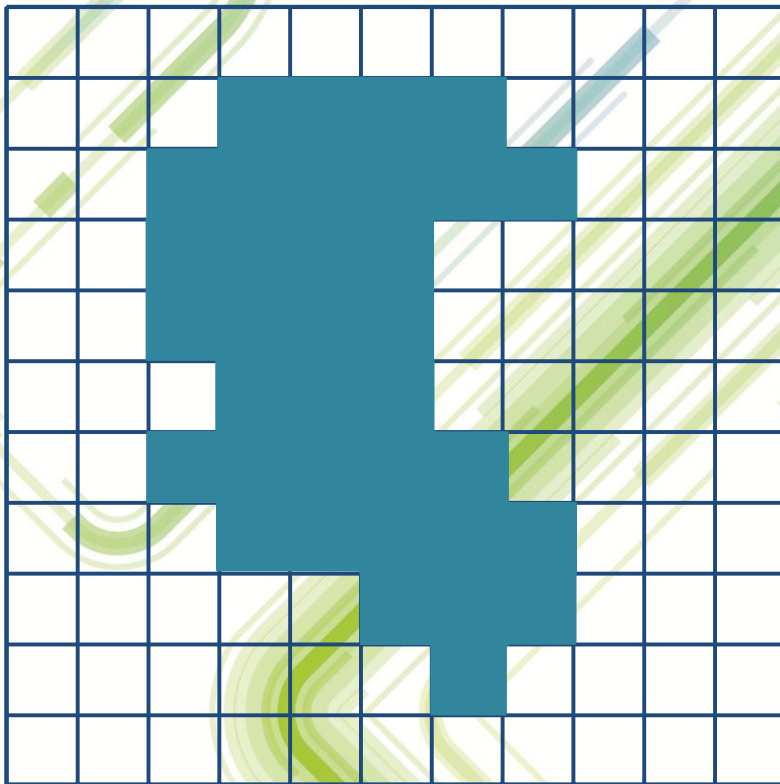
Dilated image



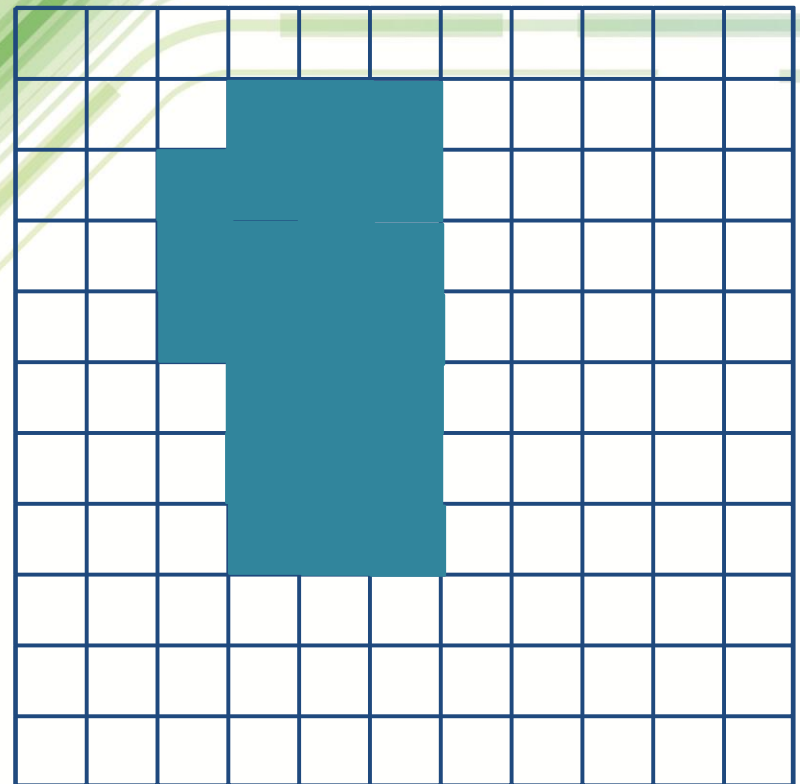


# Illustration of erosion+dilation

Original input image



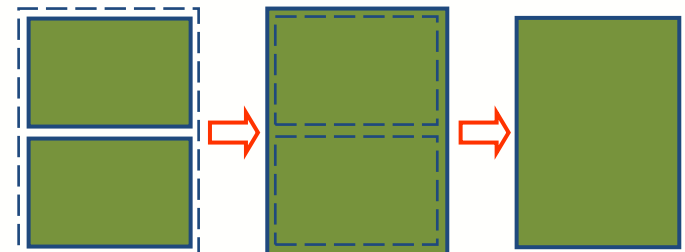
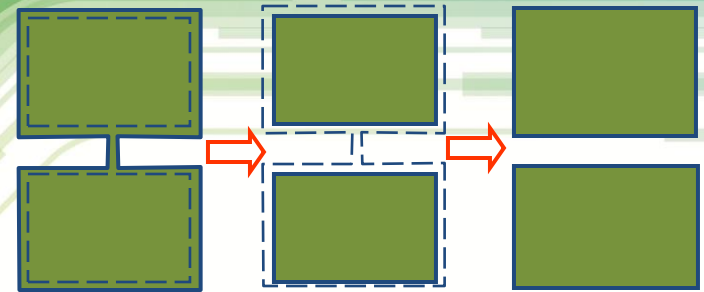
Eroded, then dilated image



Erosion followed by dilation does NOT recover the original image!

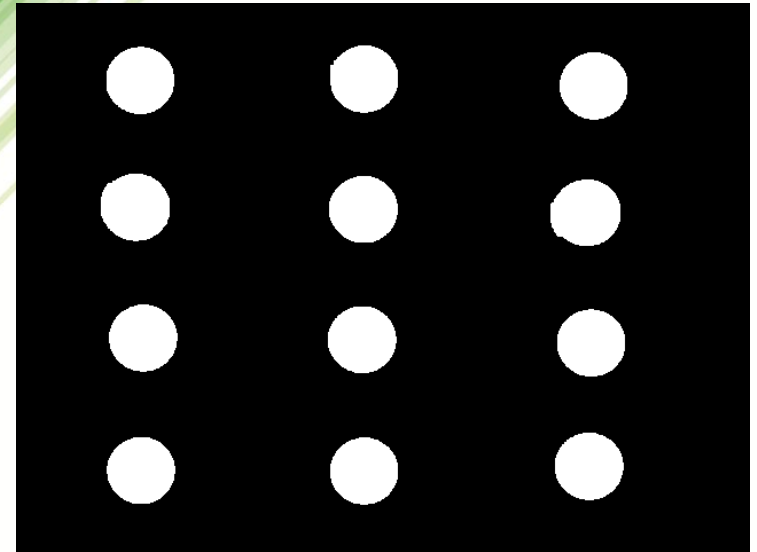
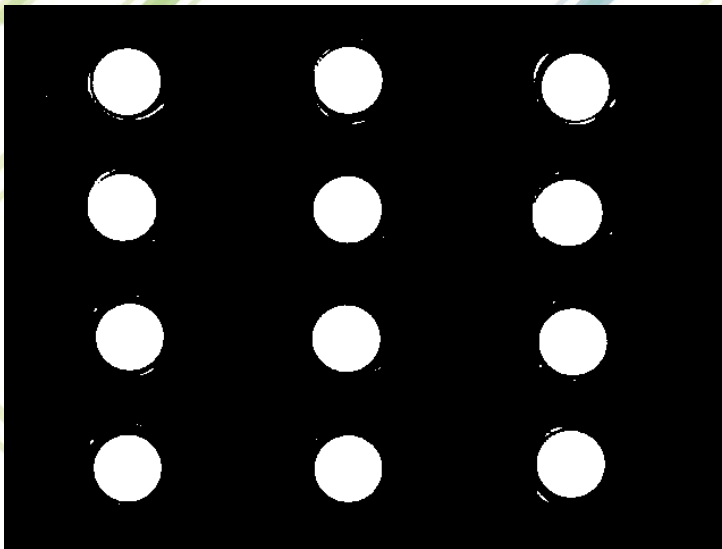
# Opening/closing

- Composite operations:
  - Opening (erode + dilate)
    - First shrink (erode) then expand (dilate)
    - By doing so what remains after erosion, is recovered by dilation to its (nearly) original situation.
  - Closing (dilate + erode)
    - First dilate, then erode.
    - The overall effect is, that what remains after dilation is recovered to its (nearly) initial state.
  - Matlab: `imopen()`, `imclose()`



# Opening example

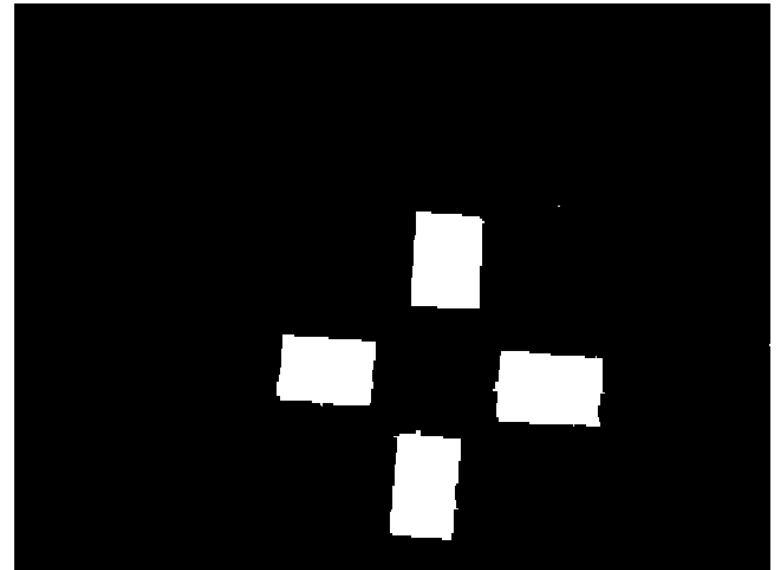
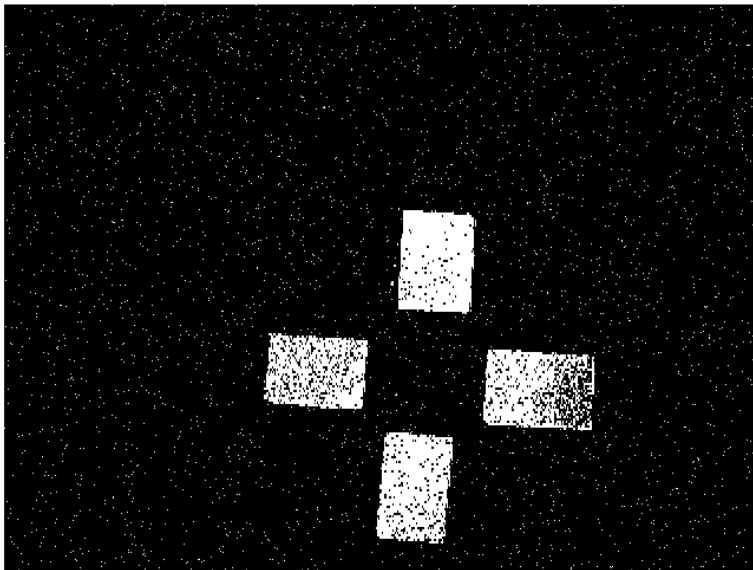
- Remember: Erosion followed by dilation



- Deletes small features (usually noise!)

# Closing example

- Structuring element, square 5x5, `SE = strel('square',5);`  
`Ic = imclose(I, SE); Ie = imerode(Ic, SE);`



Removed noise from the foreground and background!



# Hit or miss

- In this case structuring element is used to define both pixels that must belong to the background as well as pixels that should be part of the object's region.

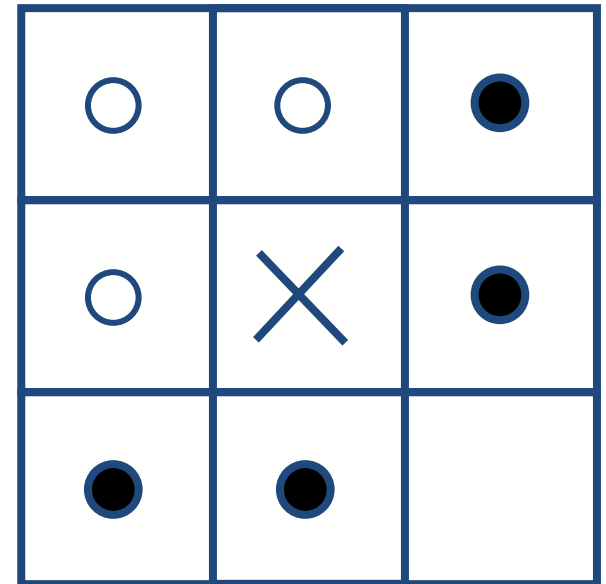
- The operation takes place (the result is 1) only if all of the neighbors within the structuring element match, otherwise the result is 0.

- Example: thinning

● Pixel should be part of the object

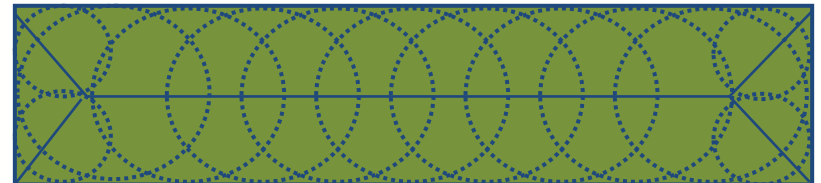
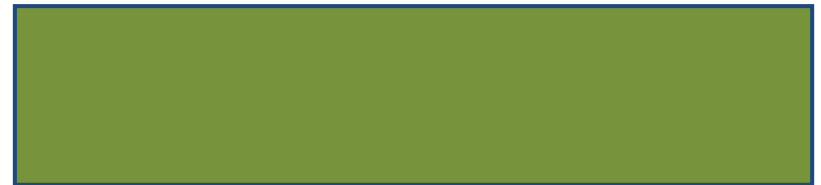
○ Pixel should be part of the background

Empty: don't care



# Thinning

- Thinning and skeletonization
  - Skeleton is a set of pixels that belong to region and are equally distant from at least two border pixels
  - Thinning algorithms produce approximation of skeleton.
    - The result of thinning a region is a set of connected pixels of unit (single pixel) width.
  - Many thinning algorithms exist
    - parallel
    - sequential



# Thinning example

Original image



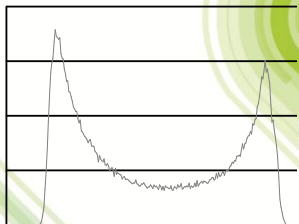
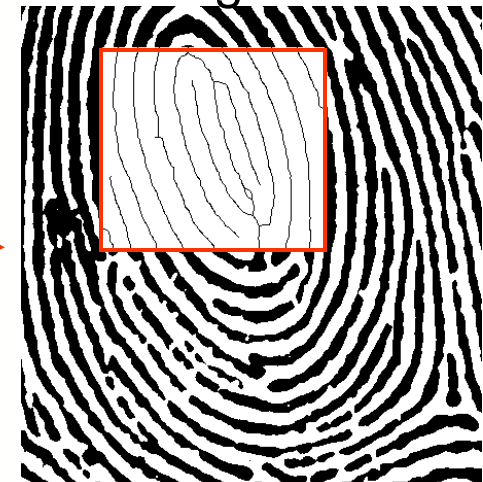
Binary image



Opening



Thinned images





The background features a series of overlapping, curved lines in various shades of green and blue, creating a sense of depth and movement. The lines are of varying thickness and are arranged in a way that suggests a complex, interconnected network or a stylized representation of data flow.

# Questions?