# Lecture 2

# Computer Vision: Algorithms and Applications


**Dr. Ahmed Elngar**

**Faculty of Computers and Artificial Intelligence**

**Beni-Suef University**

# Course Syllabus



- Introduction.
- Image Morphology.
- Image Segmentation & Color Spaces.
- Feature Detection.
- Feature Matching.
- Image Clustering.
- Image Classification.
- Face Detection & Recognition.
- Image Stitching.
- Scene Detection.
- Video Shot Detection.
- Object Tracking.

- Image Terminology
- Binary Operations
- Filtering
- Edge Operators

# Digital Image Terminology:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 95 | 96 | 94 | 93 | 92 |
| 0 | 0 | 92 | 93 | 93 | 92 | 92 |
| 0 | 0 | 93 | 93 | 94 | 92 | 93 |
| 0 | 1 | 92 | 93 | 93 | 93 | 93 |
| 0 | 0 | 94 | 95 | 95 | 96 | 95 |

pixel (with value 94)

its 3x3 neighborhood

region of medium intensity

resolution (7x7)

- binary image
- gray-scale (or gray-tone) image
- color image
- multi-spectral image
- range image
- labeled image

# The Three Stages of Computer Vision



- low-level

  image ⟶ image

- mid-level

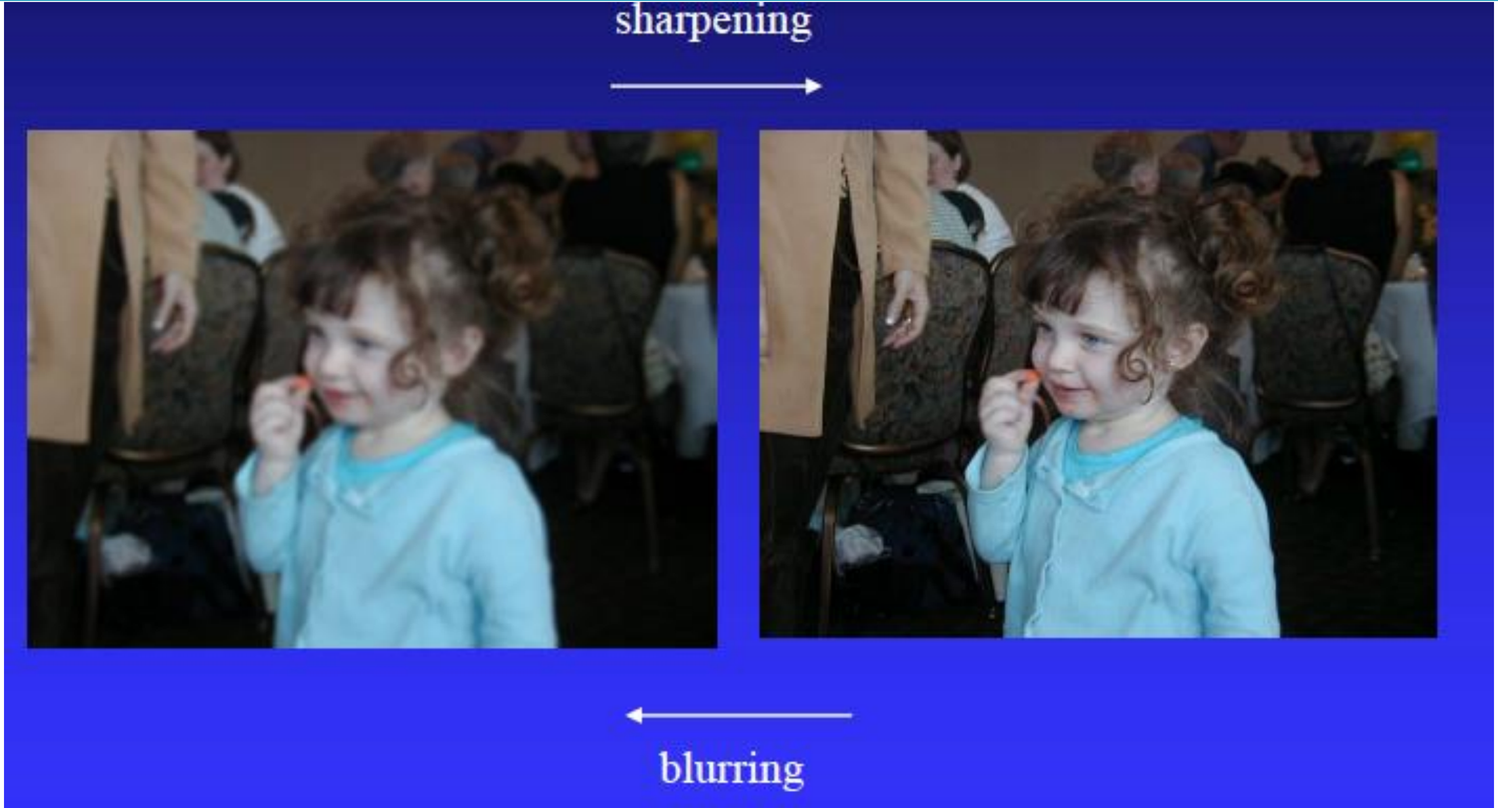
  image ⟶ features

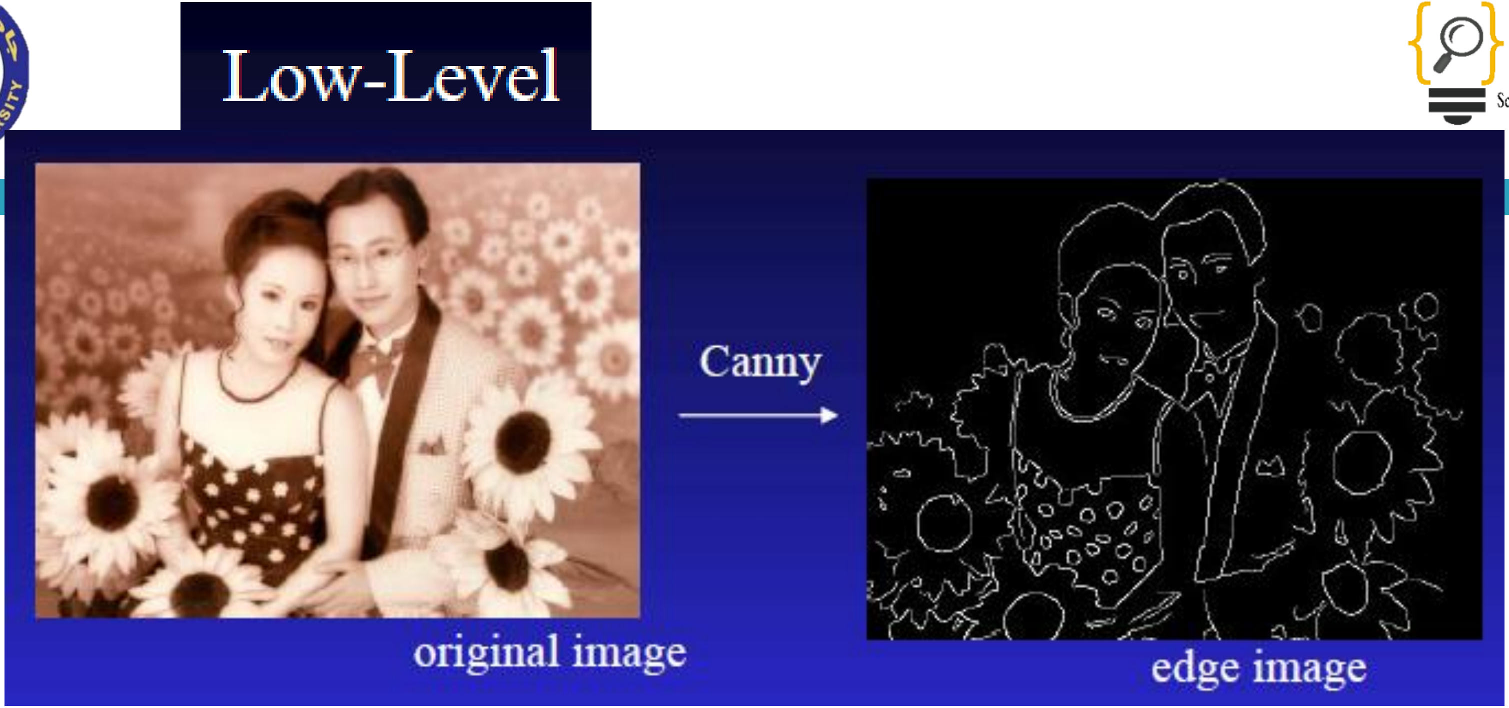
- high-level

  features ⟶ analysis

# Low-Level







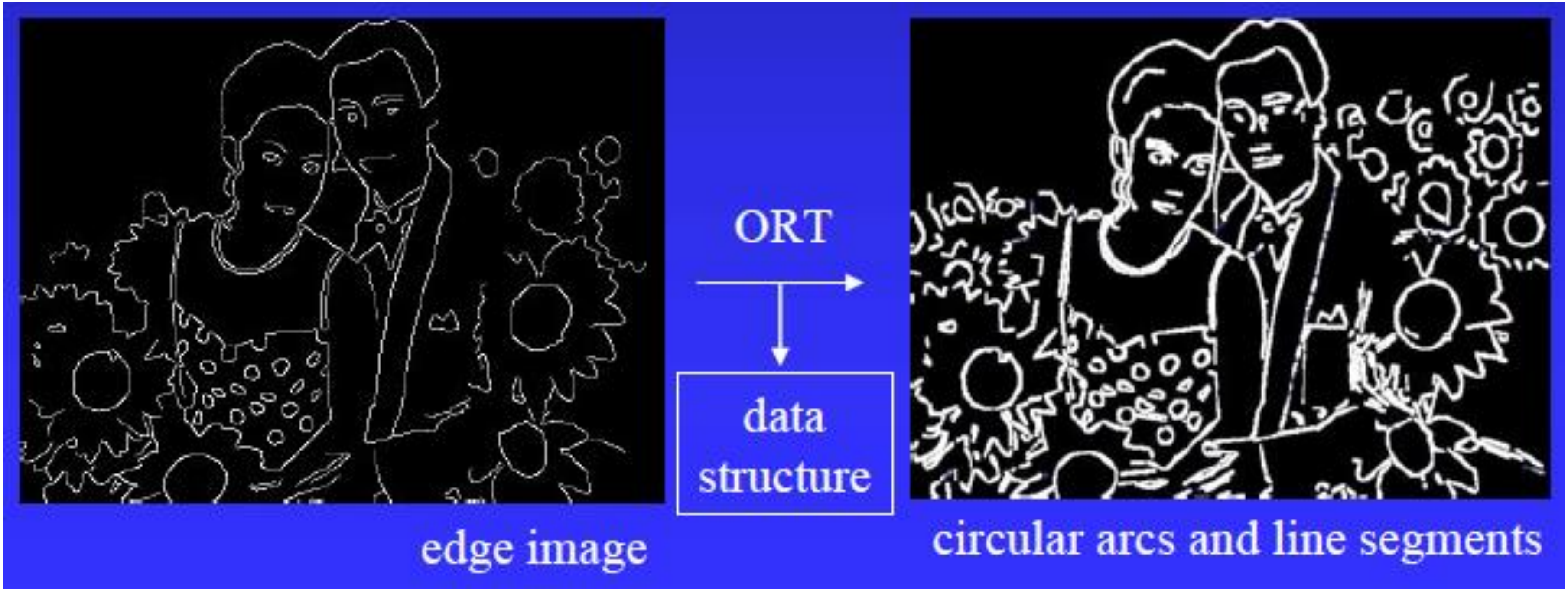

**The Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

**Canny edge detection** is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

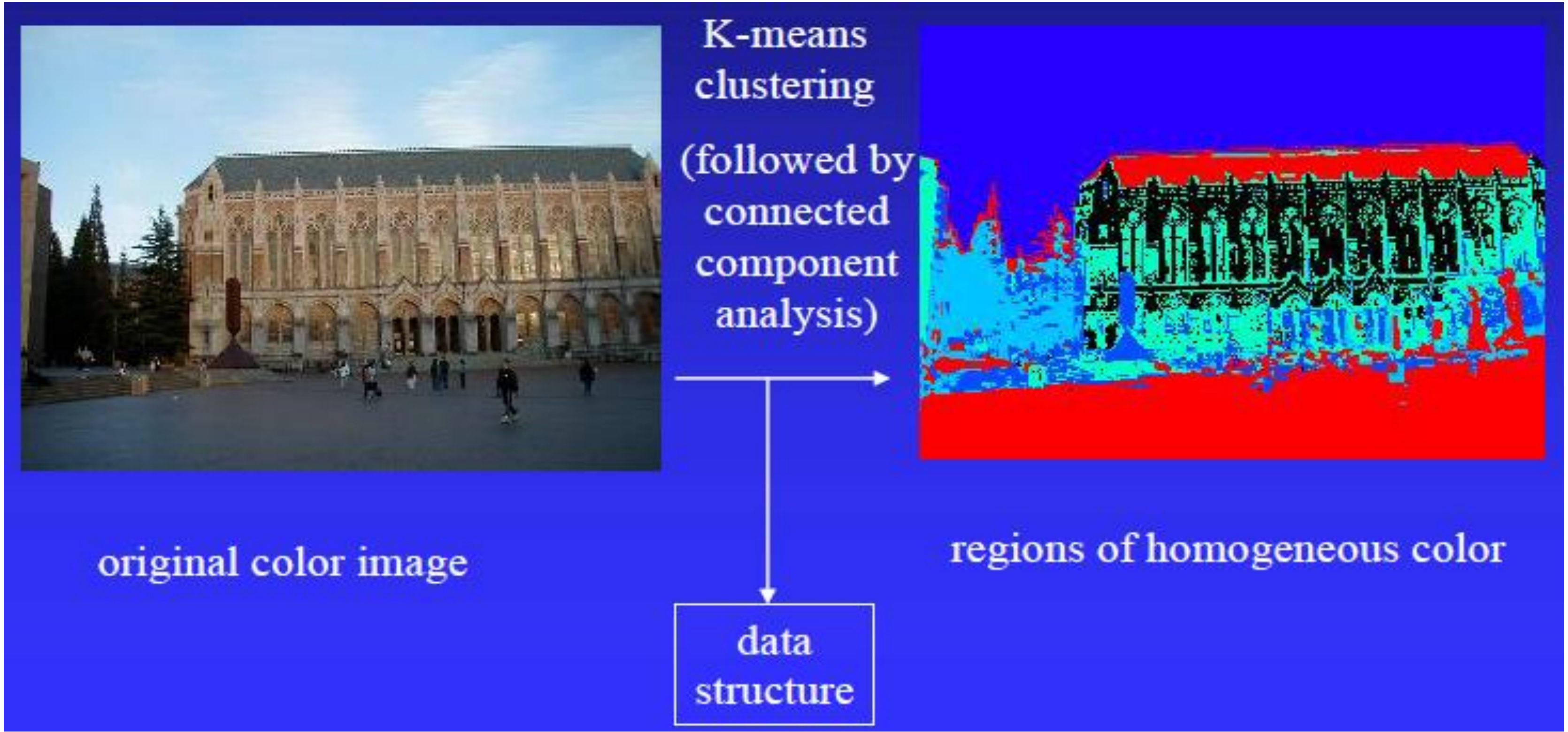
1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

# Mid-Level

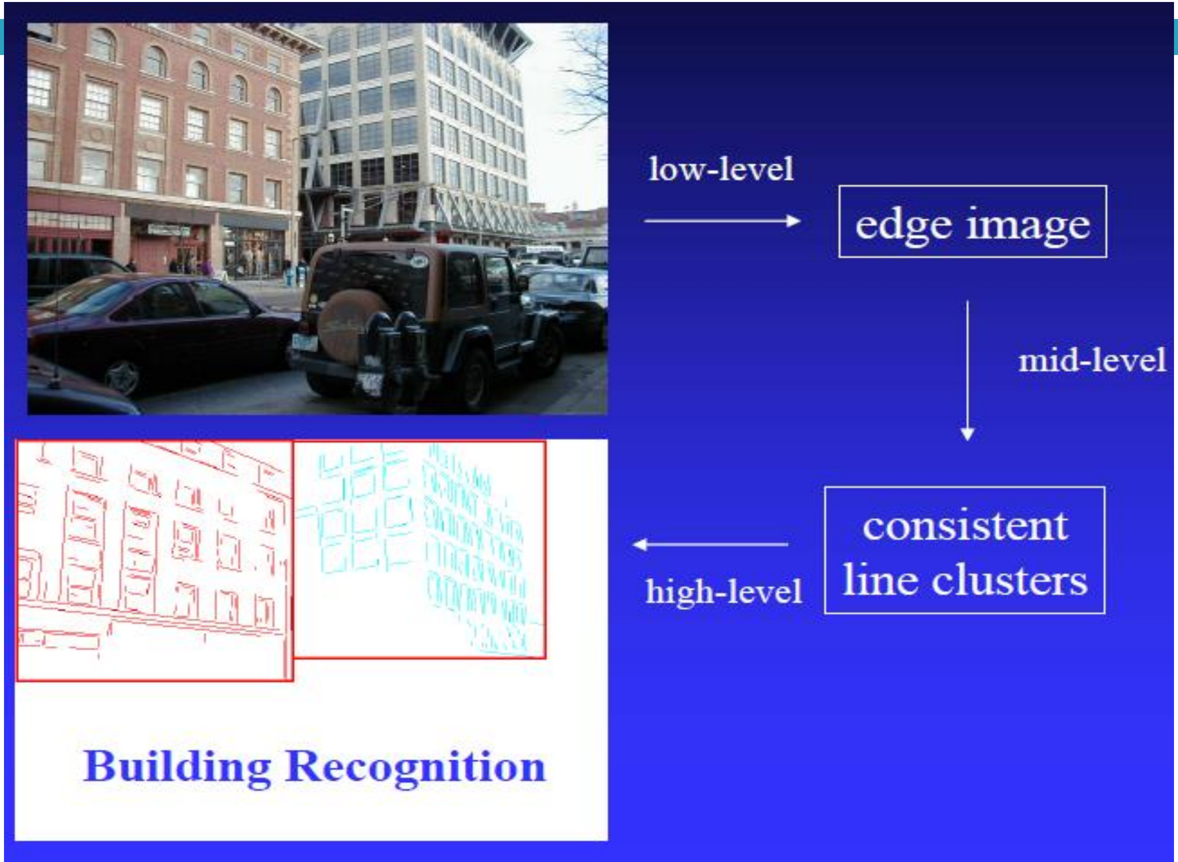



ORT
data structure
edge image
circular arcs and line segments

# Mid-level


K-means
clustering
(followed by
connected
component
analysis)
original color image
regions of homogeneous color
data
structure

# Low- to High-Level



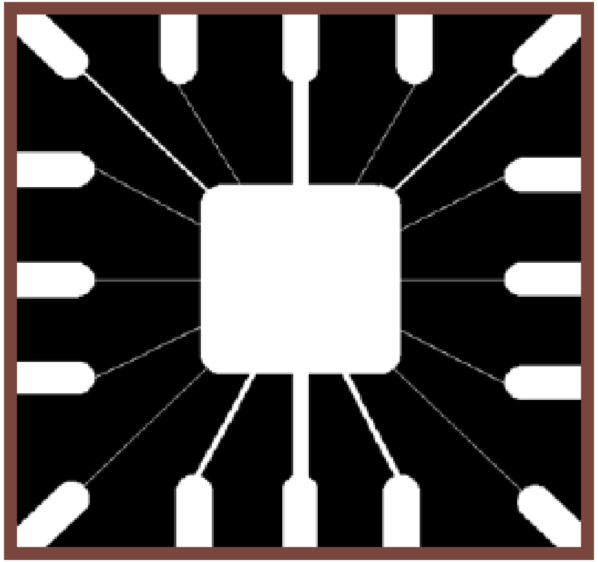
# Binary Image Analysis



Operations that produce or process binary images, typically 0's and 1's

- 0 represents background
- 1 represents foreground

```
00010010001000
00011110001000
00010010001000
```

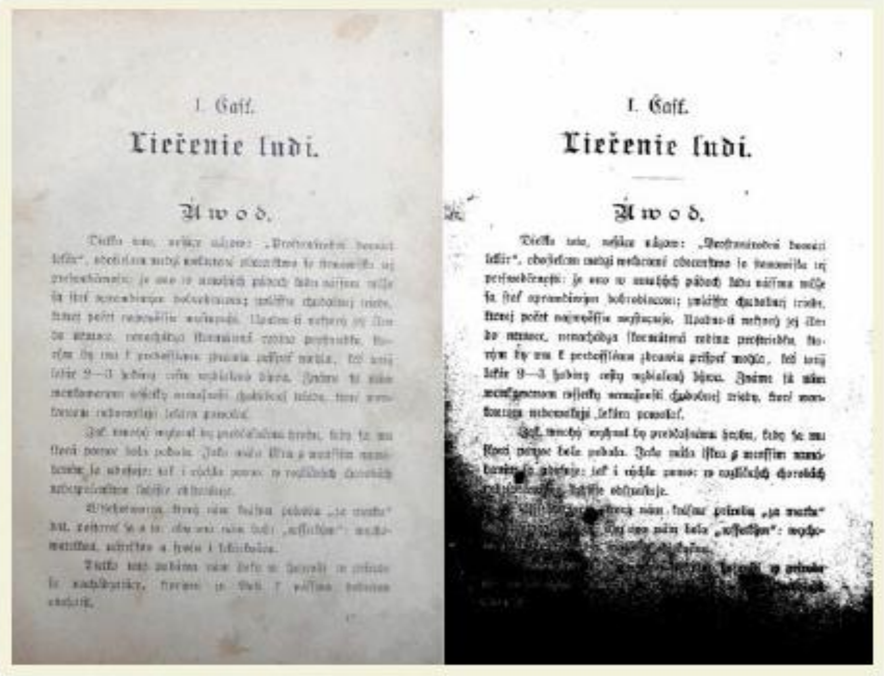


# Binary Image Analysis



Used in a number of practical applications

- Part inspection
- Manufacturing
- Document processing

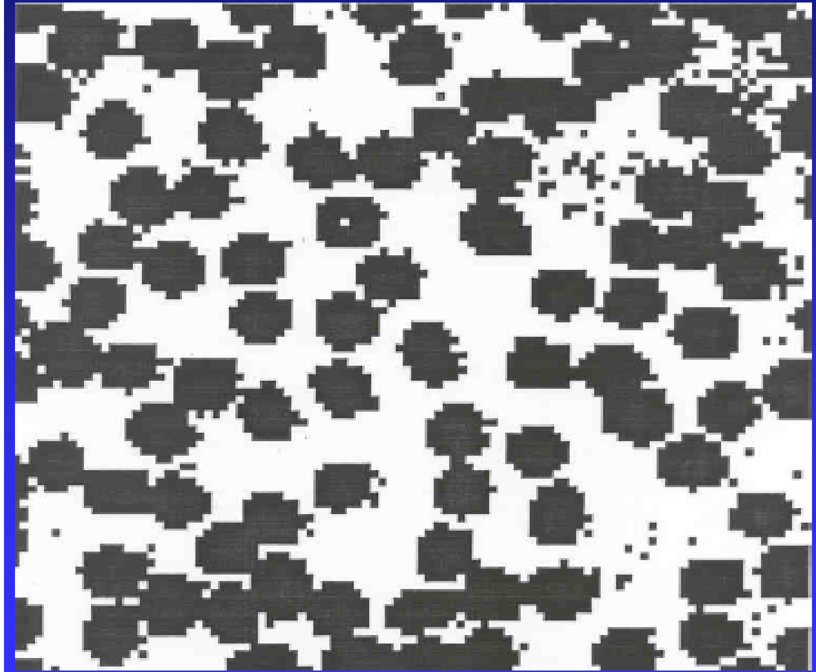


# What kinds of operations?



- Separate objects from background and from one another
- Aggregate pixels for each object
- Compute features for each object

# Example: red blood cell image



- Many blood cells are separate objects
- Many touch – bad!
- Salt and pepper noise from thresholding
- How useable is this data?
- What operations are needed to clean it up?

# Results of analysis

- 63 separate objects detected
- Single cells have area about 50
- Noise spots
- Gobs of cells

| Object | Area | Centroid | Bounding Box | |
|---|---|---|---|---|
| 1 | 383 | ( 8.8 , 20) | [1 22 1 39] | |
| 2 | 83 | ( 5.8 , 50) | [1 11 42 55] | |
| 3 | 11 | ( 1.5 , 57) | [1 2 55 60] | |
| 4 | 1 | ( 1 , 62) | [1 1 62 62] | |
| 5 | 1048 | ( 19 , 75) | [1 40 35 100] | gobs |
| 32 | 45 | ( 43 , 32) | [40 46 28 35] | cell |
| 33 | 11 | ( 44 , 1e+02) | [41 47 98 100] | |
| 34 | 52 | ( 45 , 87) | [42 48 83 91] | cell |
| 35 | 54 | ( 48 , 53) | [44 52 49 57] | cell |
| 60 | 44 | ( 88 , 78) | [85 90 74 82] | |
| 61 | 1 | ( 85 , 94) | [85 85 94 94] | |
| 62 | 8 | ( 90 , 2.5) | [89 90 1 4] | |
| 63 | 1 | ( 90 , 6) | [90 90 6 6] | |

# Useful Operations

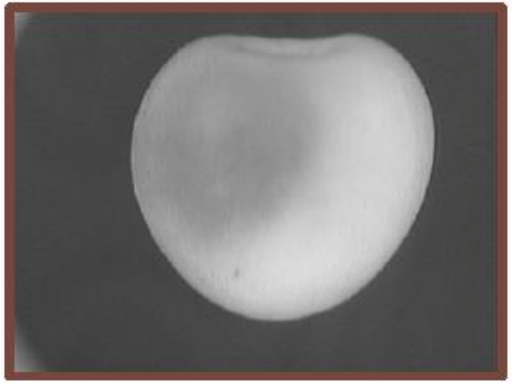


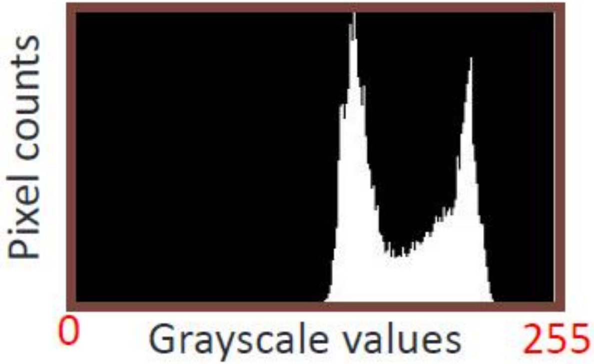
1. Thresholding a gray-tone image
2. Determining good thresholds
3. Filtering with mathematical morphology
4. Connected components analysis
5. Numeric feature extraction
    - location features
    - gray-tone features
    - shape features ...

# Thresholding



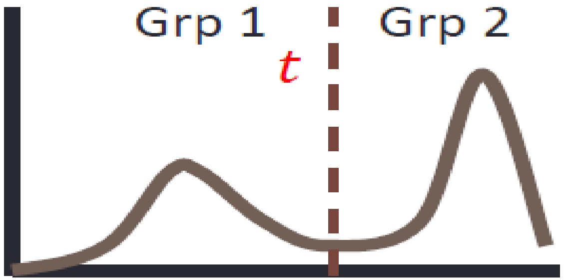
- Background is black
- Healthy cherry is bright
- Bruise is medium dark
- Histogram shows two cherry regions (black background has been removed)





# Automatic Thresholding: Otsu's Method

Assumption: The histogram is bimodal

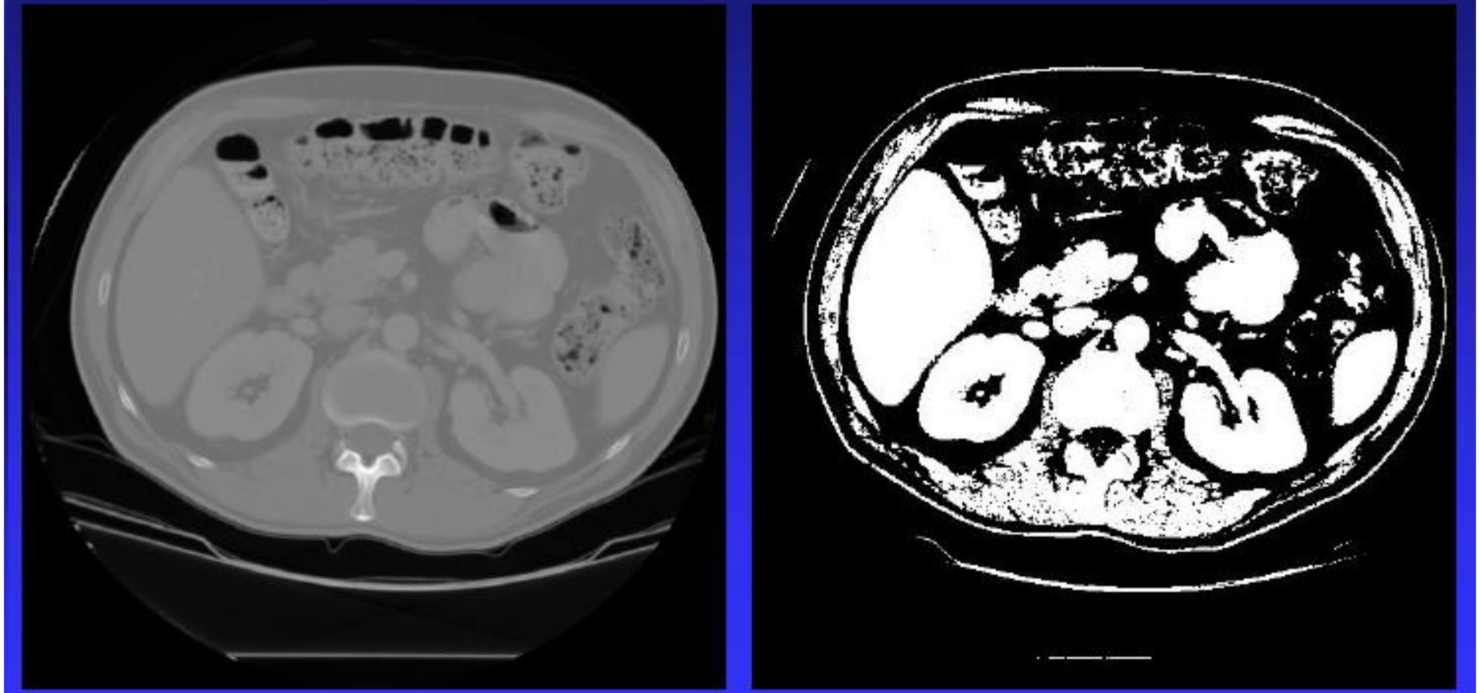



Method: Find the threshold $t$ that minimizes the weighted sum of within-group variances for the two groups that result from separating the gray tones at value $t$

Works well if the assumption holds.

# Thresholding Example





original image

pixels above threshold

# 3.3.2 Morphology

## Introduction

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

# Mathematical Morphology



- The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory.
- The operators are particularly useful for the analysis of binary images and common usages include:
  - edge detection,
  - noise removal,
  - image enhancement
  - and image segmentation.

# Binary Images

- **Binary images** are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or 255 for white.

# Structuring Elements



- The field of mathematical morphology provides a number of important image processing operations, including erosion, dilation, opening and closing.
- All these morphological operators take two pieces of data as input. One is the input image, which may be either binary or grayscale for most of the operators. The other is the *structuring element*. It is this that determines the precise details of the effect of the operator on the image.
- The structuring element is sometimes called the *kernel*, but we reserve that term for the similar objects used in convolutions.

# The structuring element



- The structuring element is already just a set of point coordinates (although it is often represented as a binary image). It differs from the input image coordinate set in that it is normally much smaller, and its coordinate origin is often not in a corner, so that some coordinate elements will have negative values.
- Note that in many implementations of morphological operators, the structuring element is assumed to be a particular shape (*e.g.* a 3×3 square) and so is hardwired into the algorithm.

- The structuring element consists of a pattern specified as the coordinates of a number of discrete points relative to some origin.
- Normally cartesian coordinates are used and so a convenient way of representing the element is as a small image on a rectangular grid.
- Figure 1 shows a number of different structuring elements of various sizes. In each case the origin is marked by a ring around that point.
- The origin does not have to be in the center of the structuring element, but often it is.
- As suggested by the figure, structuring elements that fit into a 3×3 grid with its origin at the center are the most commonly seen type.

# Morphological image processing

- Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image.
- morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images.
- Morphological operations can also be applied to greyscale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest.
- In signal processing, a nonlinear (or non-linear) filter is a filter whose output is not a linear function of its input. That is, if the filter outputs signals R and S for two input signals r and s separately, but does not always output $\alpha R + \beta S$ when the input is a linear combination $\alpha r + \beta s$.

# Introduction

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

# Structuring Element

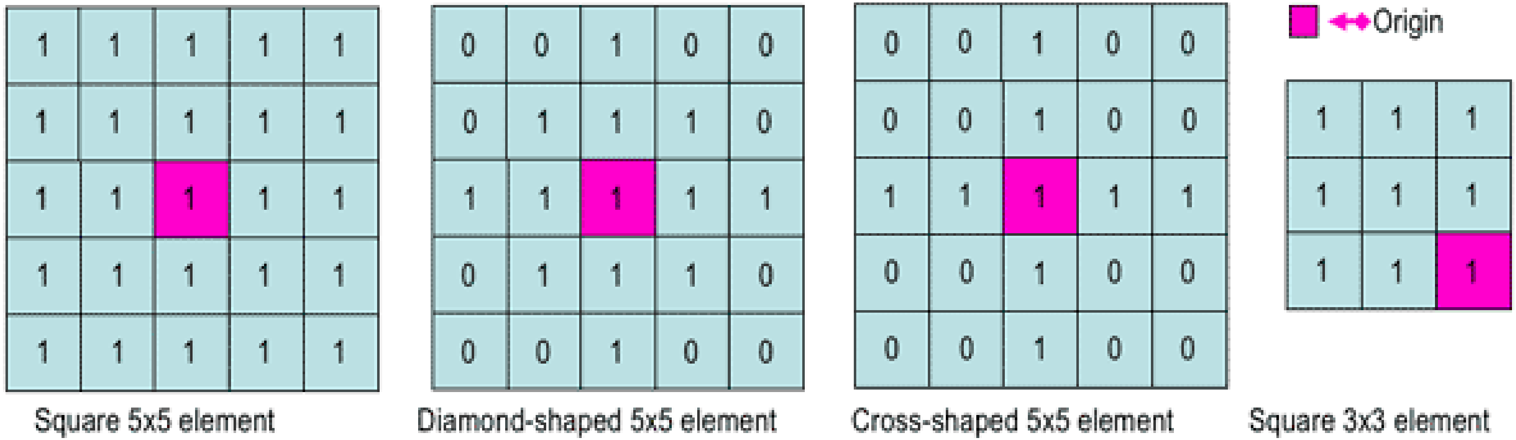


- Morphological techniques probe an image with a small shape or template called a **structuring element**.

- The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels.

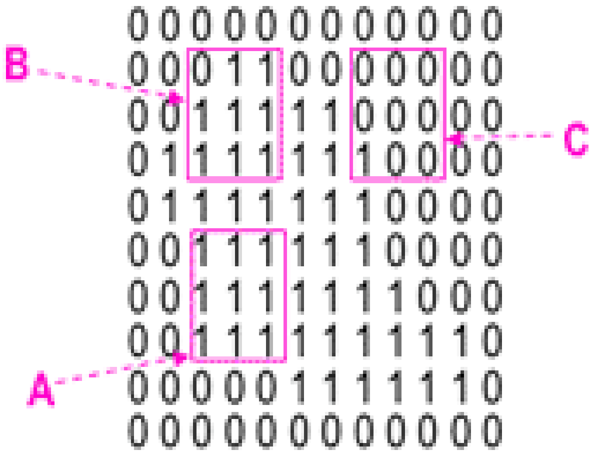
- Some operations test whether the element "fits" within the neighbourhood, while others test whether it "hits" or intersects the neighbourhood:

- The **structuring element** is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one:
- The matrix dimensions specify the *size* of the structuring element.
- The pattern of ones and zeros specifies the *shape* of the structuring element.
- An *origin* of the structuring element is usually one of its pixels, although generally the origin can be outside the structuring element.




Examples of simple structuring elements.

- A common practice is to have odd dimensions of the structuring matrix and the origin defined as the center of the matrix. Structuring elements play in morphological image processing the same role as convolution kernels in linear image filtering.
- Convolution provides a way of `multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.
- When a structuring element is placed in a binary image, each of its pixels is associated with the corresponding pixel of the neighborhood under the structuring element.
- **The structuring element is said to fit the image if, for each of its pixels set to 1, the corresponding image pixel is also 1. Similarly, a structuring element is said to hit, or intersect, an image if, at least for one of its pixels set to 1 the corresponding image pixel is also 1.**




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

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

| | | 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 |

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

# 1- Structuring Element

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

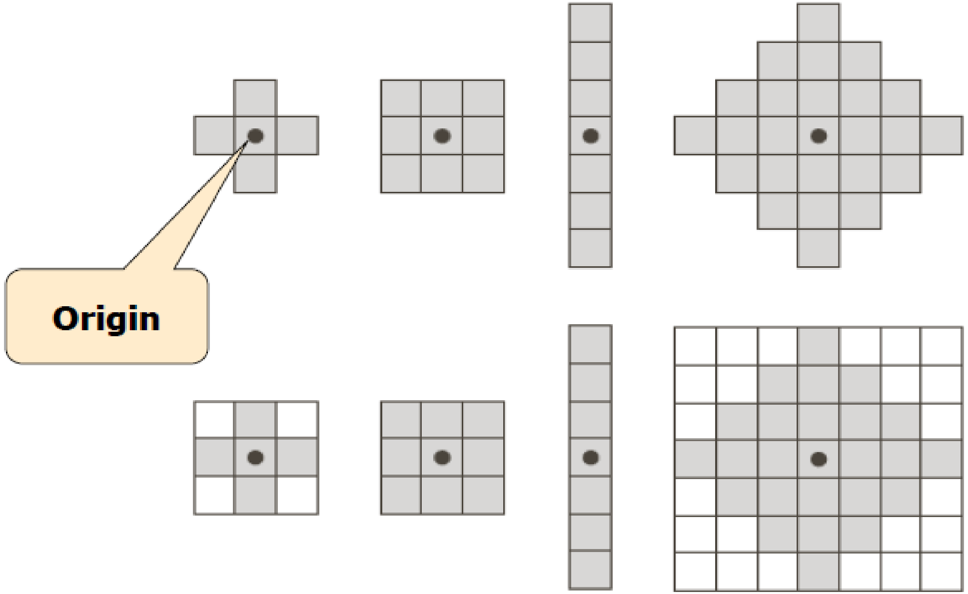
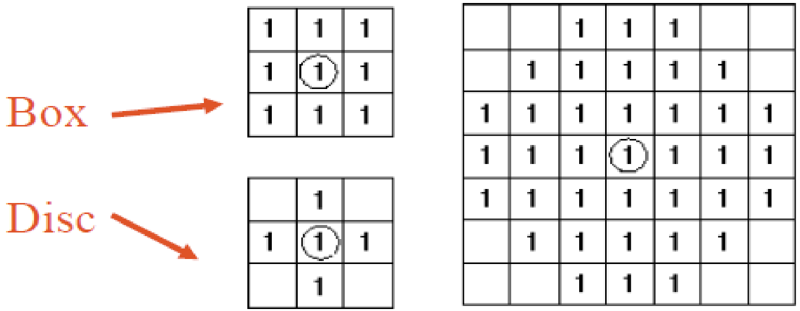



**FIGURE 9.2** First row: Examples of structuring elements. Second row: Structuring elements converted to rectangular arrays. The dots denote the centers of the SEs.

# Structuring Element

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




Examples of structuring elements

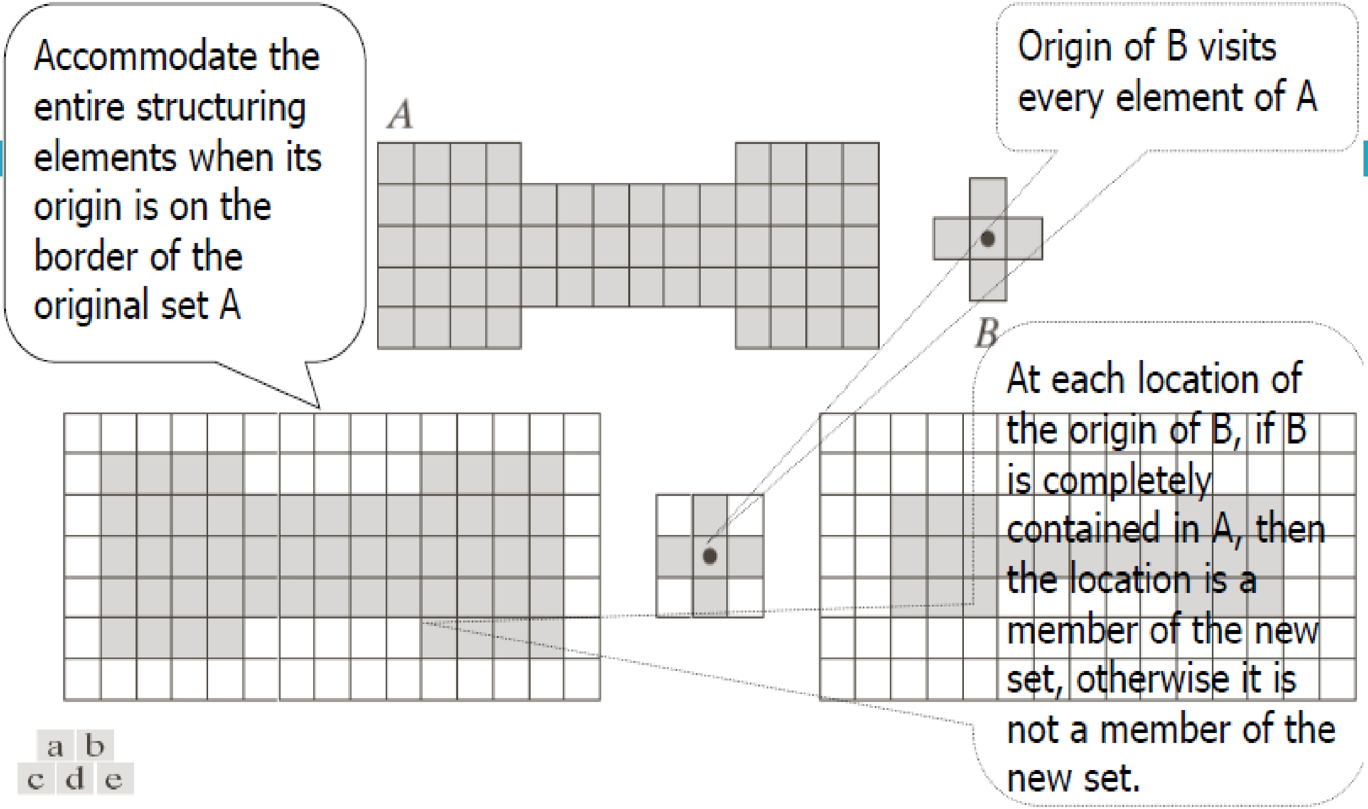



**FIGURE 9.3** (a) A set (each shaded square is a member of the set). (b) A structuring element. (c) The set padded with background elements to form a rectangular array and provide a background border. (d) Structuring element as a rectangular array. (e) Set processed by the structuring element.

6

# 2- Erosion

- Erosion is one of the two basic operators in the area of mathematical morphology, the other being dilation. It is typically applied to binary images, but there are versions that work on grayscale images.

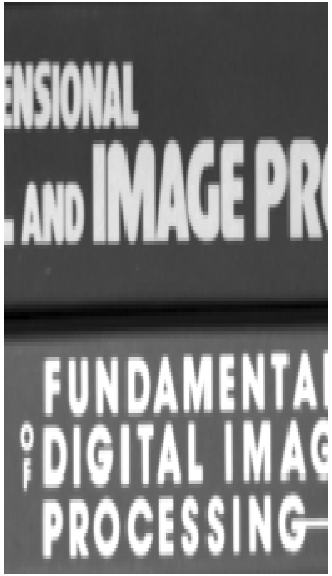
- The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels (*i.e.* white pixels, typically).

- Thus areas of foreground pixels shrink in size, and holes within those areas become larger.

The **erosion** of a binary image *f* by a structuring element *s* (denoted $f \ominus s$) produces a new binary image $g = f \ominus s$ with ones in all locations *(x,y)* of a structuring element's origin at which that structuring element *s* fits the input image *f*, i.e. $g(x,y) = 1$ is *s* fits *f* and 0 otherwise, repeating for all pixel coordinates *(x,y)*.

In many vision applications, it is useful to be able to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background. Thresholding often provides an easy and convenient way to perform this segmentation on the basis of the different intensities or colors in the foreground and background regions of an image.

In addition, it is often useful to be able to see what areas of an image consist of pixels whose values lie within a specified range, or *band* of intensities (or colors). Thresholding can be used for this as well.






Greyscale image

Binary image by thresholding

Erosion: a 2×2 square structuring element

The mathematical definition of erosion for *binary* images is as follows:

> Suppose that *X* is the set of Euclidean coordinates corresponding to the input binary image, and that *K* is the set of coordinates for the structuring element.
>
> Let *Kx* denote the translation of *K* so that its origin is at *x*.
>
> Then the erosion of *X* by *K* is simply the set of all points *x* such that *Kx* is a subset of *X*.

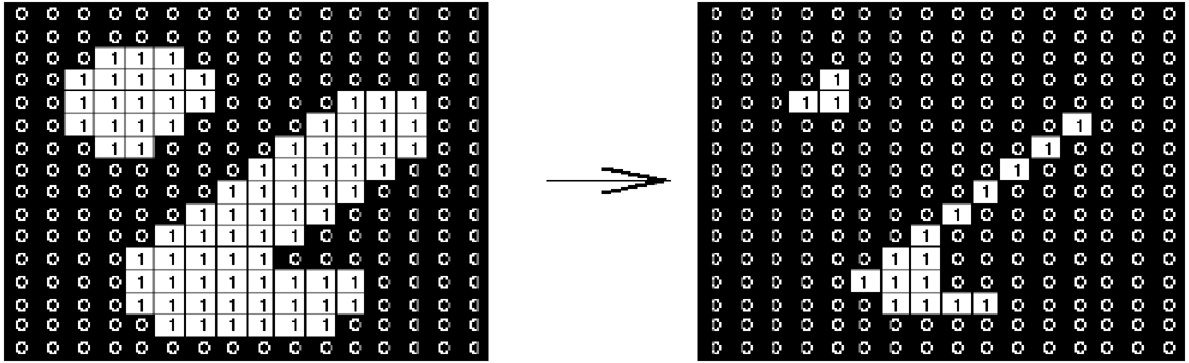
The mathematical definition for grayscale erosion is identical except in the way in which the set of coordinates associated with the input image is derived. In addition, these coordinates are 3-D rather than 2-D.

- The mathematical definition for grayscale erosion is identical except in the way in which the set of coordinates associated with the input image is derived. In addition, these coordinates are 3-D rather than 2-D.

- As an example of binary erosion, suppose that the structuring element is a 3×3 square, with the origin at its center as shown in Figure 1. Note that in this and subsequent diagrams, foreground pixels are represented by 1's and background pixels by 0's.

# 2- Erosion

- **Erosion** is an important morphological operation



- Applied **Structuring Element:**

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

Set of coordinate points =

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

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

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

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

Input image

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

Structuring Element

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

Output Image

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | | | | | | | | |

Input image

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

Structuring Element

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

Output Image

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | | | | | | | |

# Cont..

Input image

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

Structuring Element

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

Output Image

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | | | | | | |

Input image

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

Structuring Element

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

Output Image

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | | | | | |

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

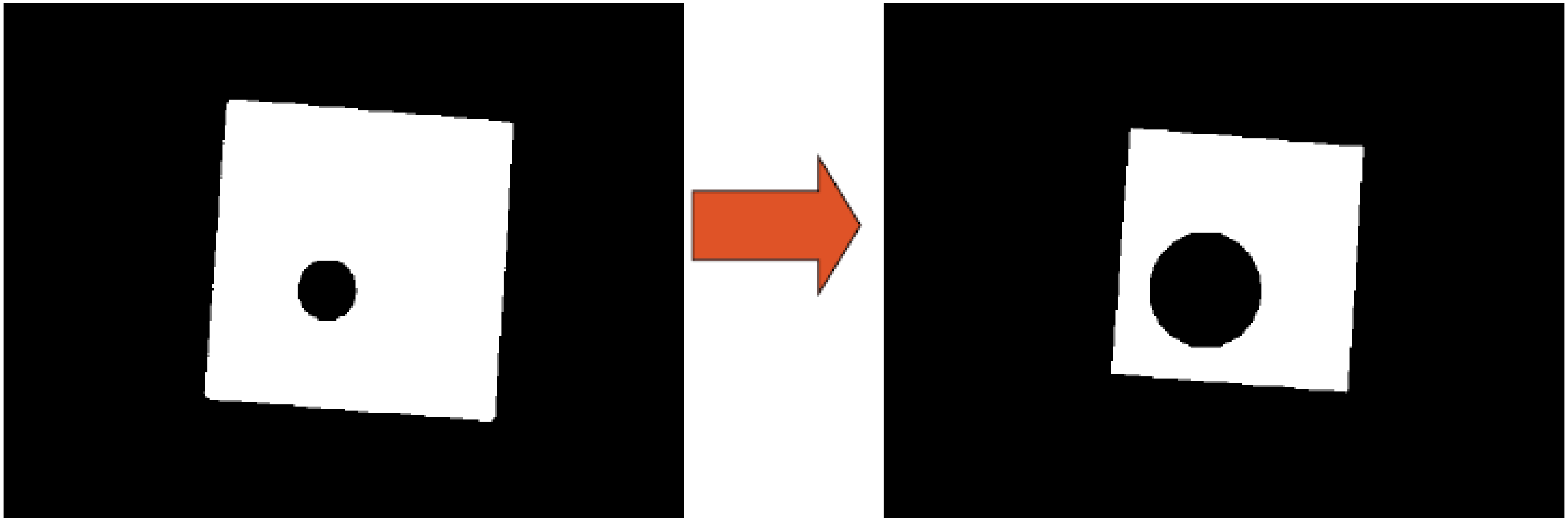
Structuring Element

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

Output Image

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

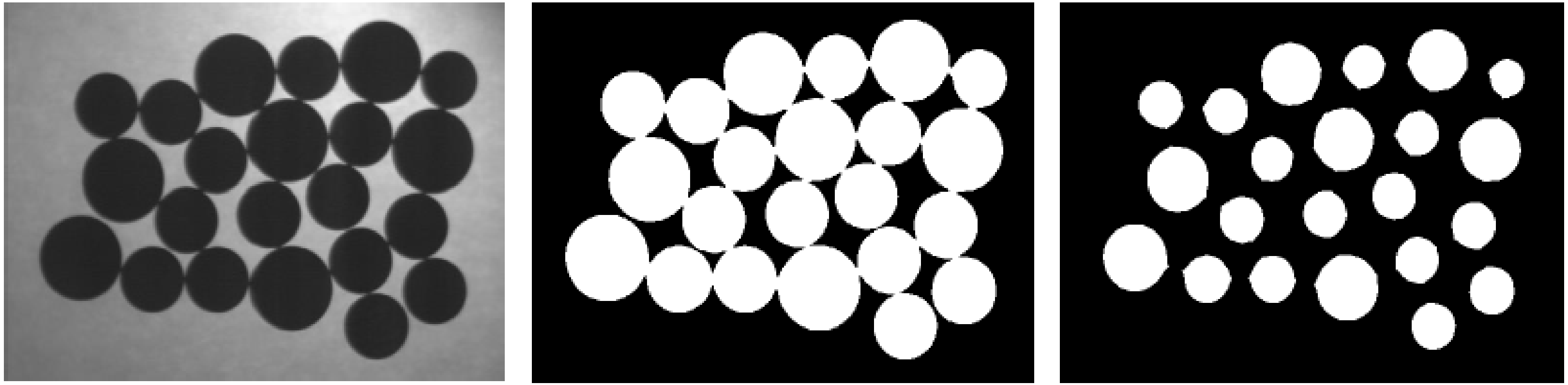
# 2- cont: Another example of erosion



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

# Counting Coins Example

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



# 3- Dilation



- Dilation is one of the two basic operators in the area of mathematical morphology, the other being erosion.
- It is typically applied to binary images, but there are versions that work on grayscale images.
- The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (*i.e.* white pixels, typically). Thus areas of foreground pixels grow in size while holes within those regions become smaller.

The mathematical definition of dilation for *binary* images is as follows:

Suppose that *X* is the set of Euclidean coordinates corresponding to the input binary image, and that *K* is the set of coordinates for the structuring element.

Let *Kx* denote the translation of *K* so that its origin is at *x*.

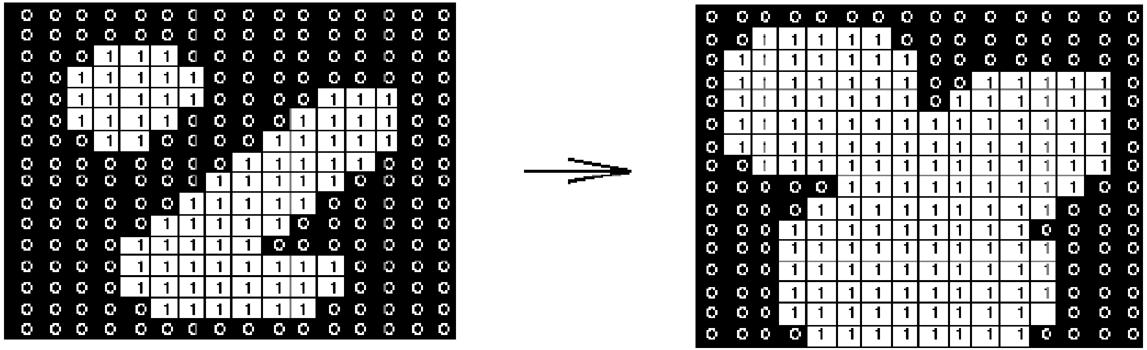
Then the dilation of *X* by *K* is simply the set of all points *x* such that the intersection of *Kx* with *X* is non-empty.

The mathematical definition of grayscale dilation is identical except for the way in which the set of coordinates associated with the input image is derived. In addition, these coordinates are 3-D rather than 2-D.

- The mathematical definition of grayscale dilation is identical except for the way in which the set of coordinates associated with the input image is derived. In addition, these coordinates are 3-D rather than 2-D.
- As an example of binary dilation, suppose that the structuring element is a 3×3 square, with the origin at its center, as shown in Figure 1. Note that in this and subsequent diagrams, foreground pixels are represented by 1's and background pixels by 0's.

# 3- Dilation

- **Dilation** is an important morphological operation



- Applied **Structuring Element:**

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

Set of coordinate points =

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

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

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

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

# An Example

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

Output Image

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

Input image

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

Structuring Element

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

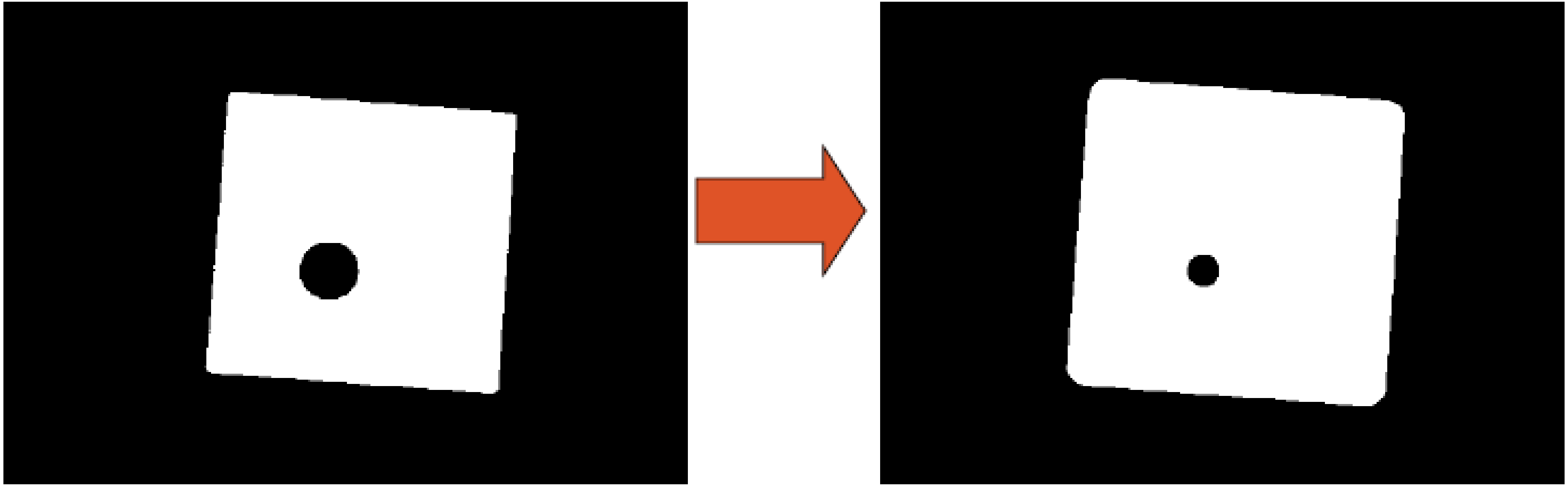
Output Image

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

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

# Another Example



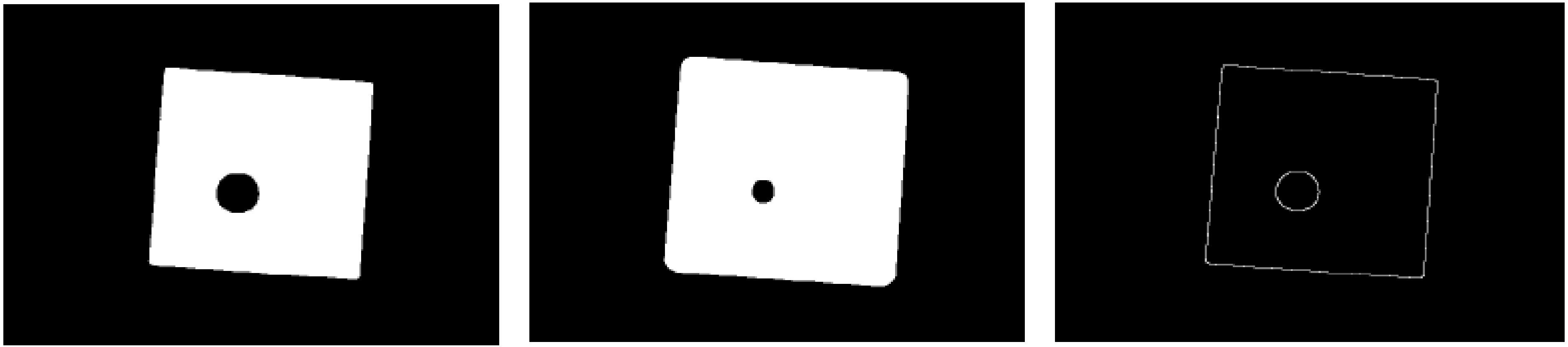


- Image get lighter, more uniform intensity

# Edge Detection



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



# Dilation & Erosion



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

# Opening & Closing



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

# Opening



- Opening and closing are two important operators from mathematical morphology. They are both derived from the fundamental operations of erosion and dilation. Like those operators they are normally applied to binary images, although there are also graylevel versions.
- The basic effect of an opening is somewhat like erosion in that it tends to remove some of the foreground (bright) pixels from the edges of regions of foreground pixels. However it is **less destructive** than erosion in general.
- As with other morphological operators, the exact operation is determined by a structuring element.
- The effect of the operator is to **preserve *foreground* regions** that have a similar shape to this structuring element, or that can completely contain the structuring element, while eliminating all other regions of foreground pixels.

# 4- Opening



Structured removal of image region boundary pixels

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

# Opening



- While erosion can be used to eliminate small clumps of undesirable foreground pixels, *e.g.* `salt noise', quite effectively, it has the big disadvantage that it will affect *all* regions of foreground pixels **indiscriminately**.
- Opening gets around this by performing both an erosion and a dilation on the image.
- The effect of opening can be quite easily visualized. Imagine taking the structuring element and sliding it around *inside* each foreground region, without changing its orientation.
- All pixels which can be covered by the structuring element with the structuring element being entirely within the foreground region will be preserved.

# Opening



- However, all foreground pixels which cannot be reached by the structuring element without parts of it moving out of the foreground region will be eroded away.
- After the opening has been carried out, the new boundaries of foreground regions will all be such that the structuring element fits inside them, and so further openings with the same element have no effect.
- The effect of an opening on a binary image using a 3×3 square structuring element is illustrated in Figure

# Opening

- Structuring element: 3x3 square

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

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

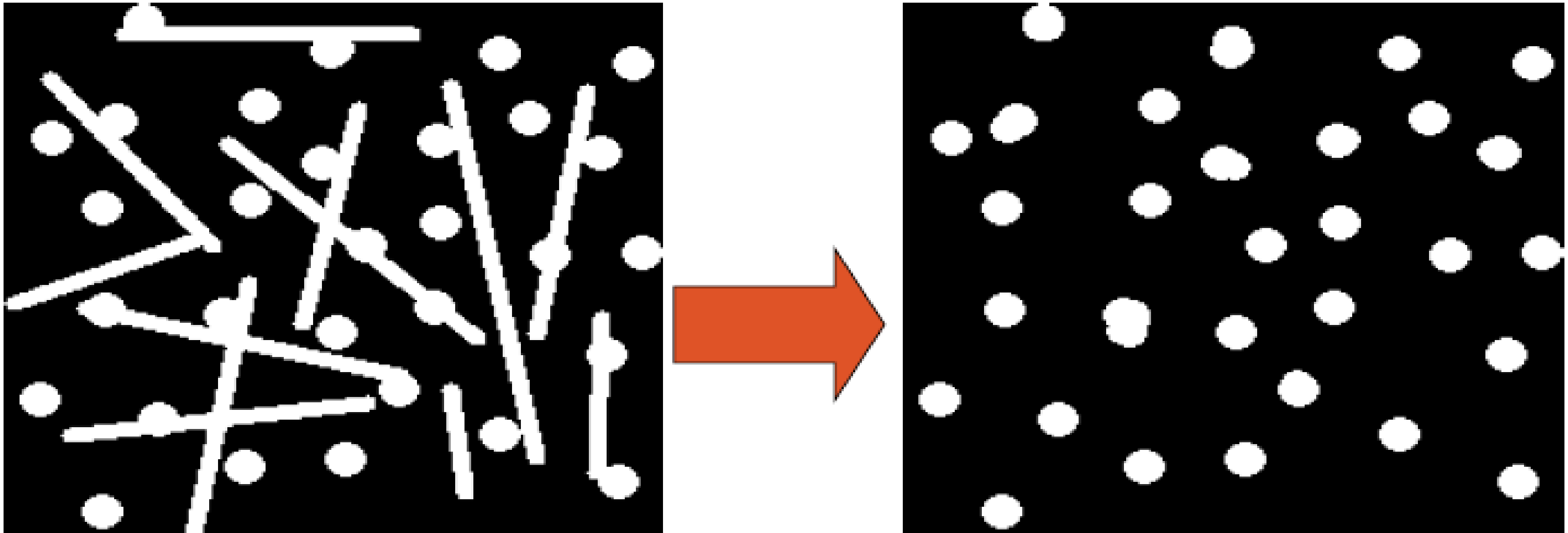
# Opening



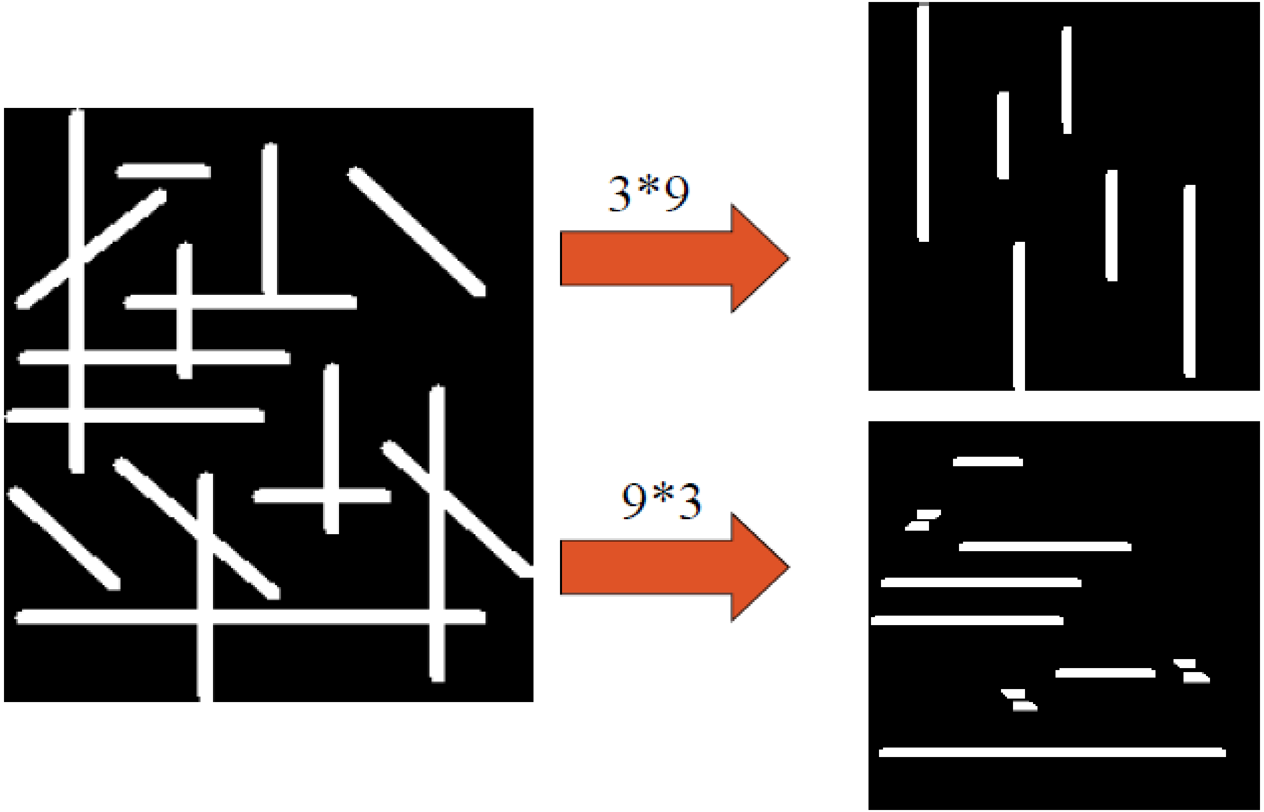
- Take the structuring element (SE) and slide it around *inside* each foreground region.
  - All pixels which can be covered by the SE with the SE being entirely within the foreground region will be preserved.
  - All foreground pixels which can not be reached by the structuring element without lapping over the edge of the foreground object will be eroded away!

# An Example

- Opening with a 11 pixel diameter disc

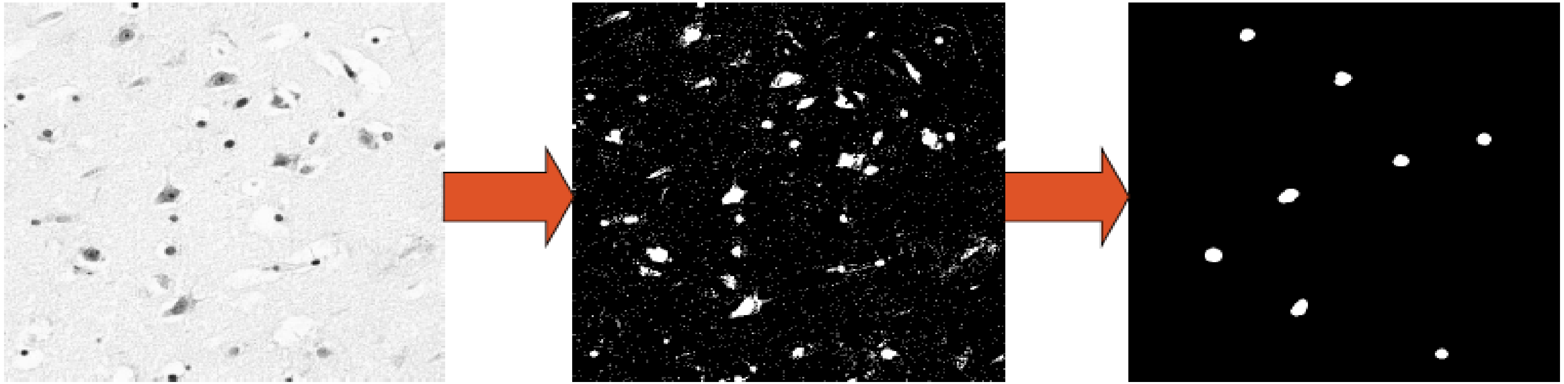


- 3x9 and 9x3 Structuring Element



# Use Opening for Separating Blobs

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



# 5- Closing

- Closing is an important operator from the field of mathematical morphology. Like its dual operator opening, it can be derived from the fundamental operations of erosion and dilation.
- Like those operators it is normally applied to binary images, although there are graylevel versions. Closing is similar in some ways to dilation in that it tends to enlarge the boundaries of foreground (bright) regions in an image (and shrink background color holes in such regions), but it is less destructive of the original boundary shape. As with other morphological operators, the exact operation is determined by a structuring element.
- The effect of the operator is to preserve *background* regions that have a similar shape to this structuring element, or that can completely contain the structuring element, while eliminating all other regions of background pixels.

- Closing is opening performed in reverse. It is defined simply as a dilation followed by an erosion *using the same structuring element for both operations*. See the sections on erosion and dilation for details of the individual steps. The closing operator therefore requires two inputs: an image to be closed and a structuring element.
- Graylevel closing consists straightforwardly of a graylevel dilation followed by a graylevel erosion.
- Closing is the dual of opening, *i.e.* closing the foreground pixels with a particular structuring element, is equivalent to closing the background with the same element.

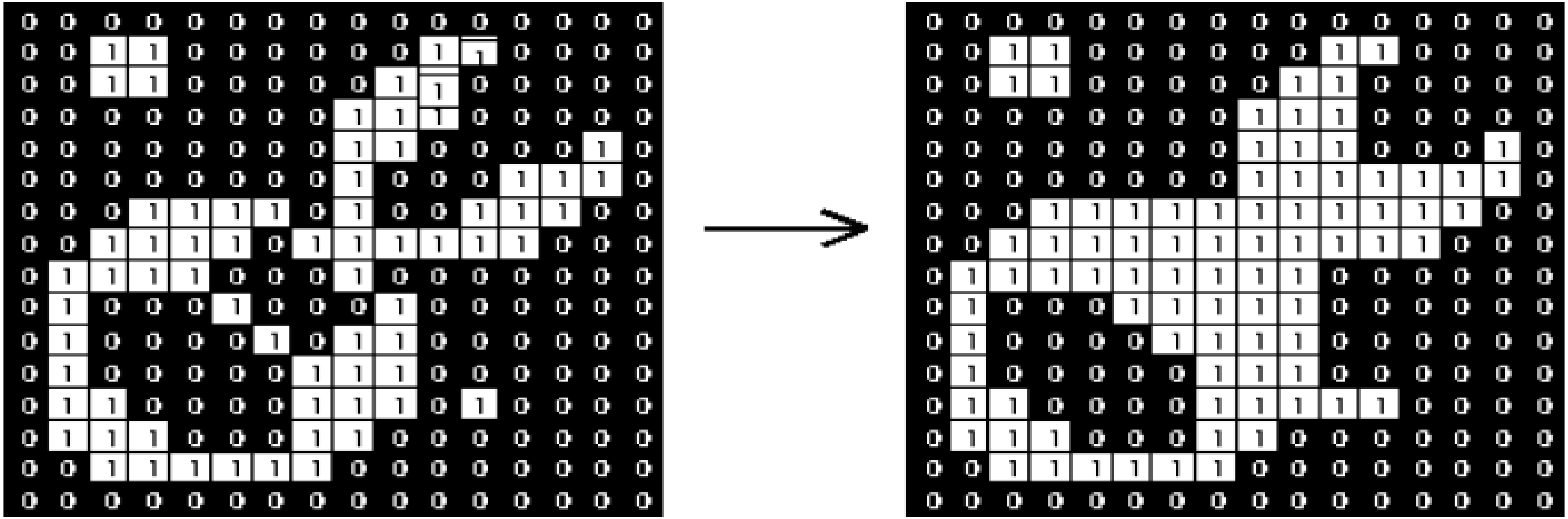
# 5- Closing



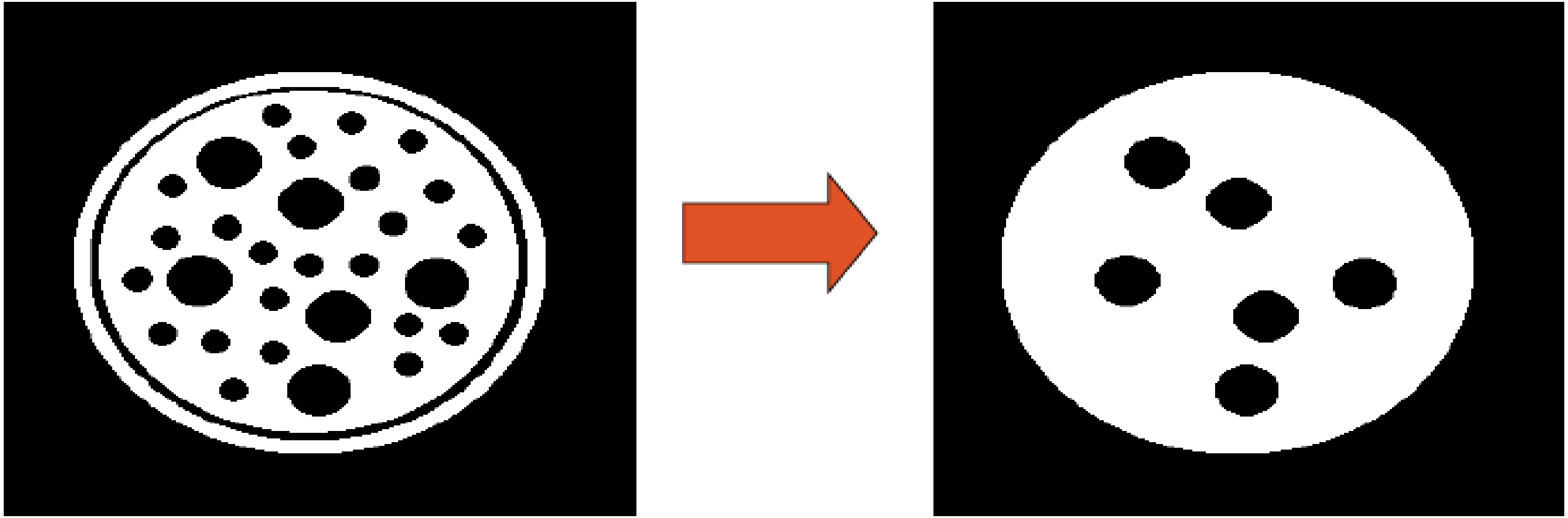
- **Similar to Dilation**
  - Removal of holes
  - Tends to enlarge regions, shrink background
- Closing is defined as a **Dilatation followed by an Erosion** using the same structuring element for both operations.

- Take the structuring element (SE) and slide it around *outside* each foreground region.
  - All background pixels which can be covered by the SE with the SE being entirely within the background region will be preserved.
  - All background pixels which can *not* be reached by the structuring element without lapping over the edge of the foreground object will be turned into a foreground.

- Structuring element: 3x3 square

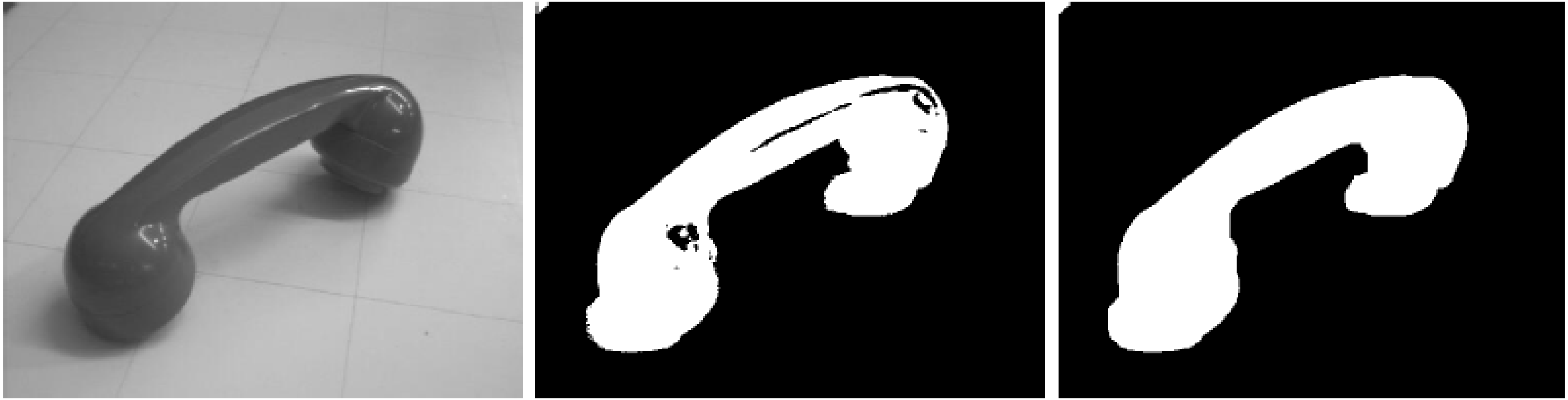


- Closing operation with a 22 pixel disc
- Closes small holes in the foreground

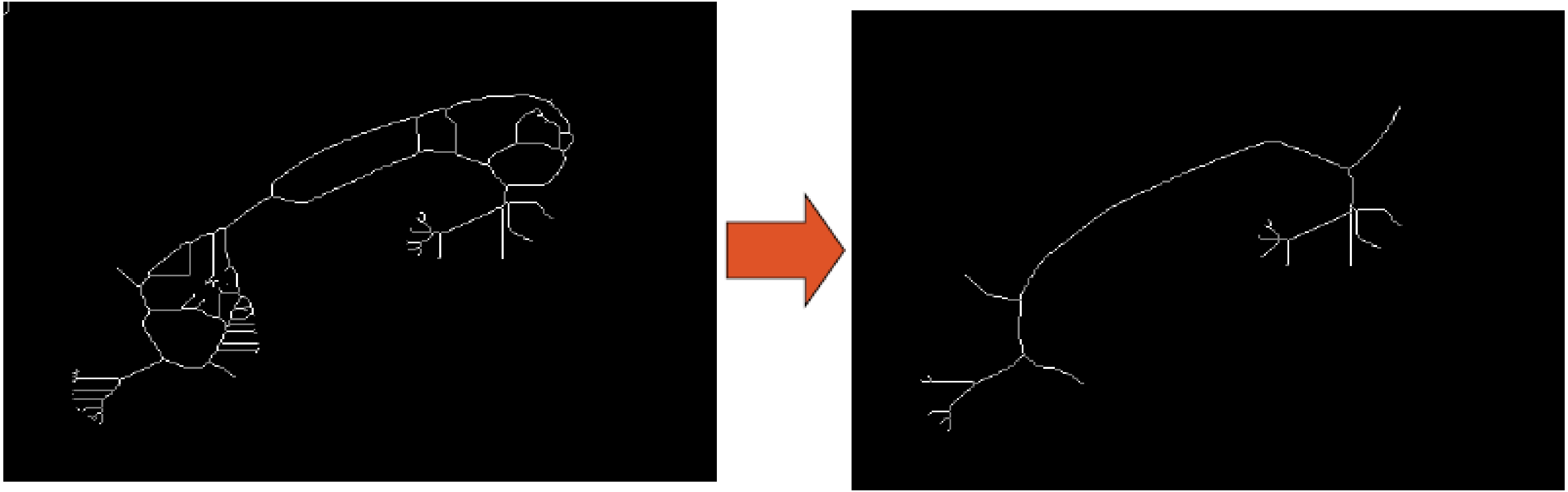


# An Example





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



# Opening & Closing



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

# 7- Thinning



- Thinning is a morphological operation that is used to remove selected foreground pixels from binary images, somewhat like erosion or opening.
- It can be used for several applications, but is particularly useful for skeletonization.
- In this mode it is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output.

# 7- Thinning



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

- If foreground and background fit the structuring element exactly, then the pixel at the origin of the SE is set to 0

- Note that the value of the SE at the origin is 1 or *don't care*!

# 8- Thickening



- Thickening is a morphological operation that is used to *grow* selected regions of foreground pixels in binary images, somewhat like dilation or closing.
- It has several applications, including determining the approximate *convex hull* of a shape, and determining the *skeleton by zone of influence*.
- Thickening is normally only applied to binary images, and it produces another binary image as output.

# 8- Thickening



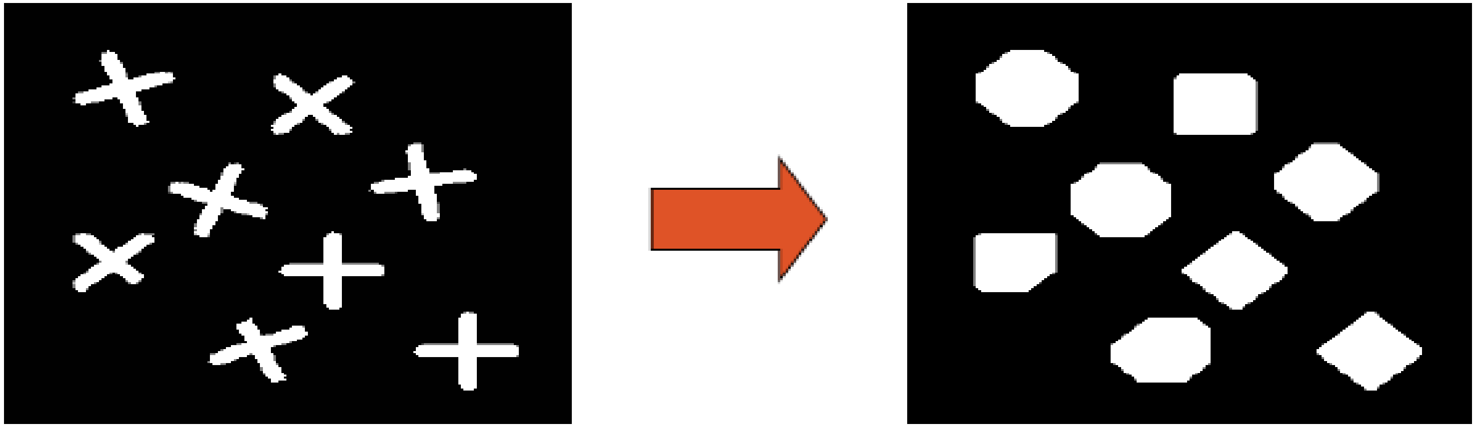
- Used to grow selected regions of foreground pixels
- If foreground and background match exactly the SE, then set the pixel at its origin to 1!
- Note that the value of the SE at the origin is 0 or *don't care*!

# An Example



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

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



# 6- Hit-and-miss Transform



- The hit-and-miss transform is a general binary morphological operation that can be used to look for particular patterns of foreground and background pixels in an image.
- It is actually the basic operation of binary morphology since almost all the other binary morphological operators can be derived from it.
- As with other binary morphological operators it takes as input a binary image and a structuring element, and produces another binary image as output.

# 6- Hit-and-miss Transform



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

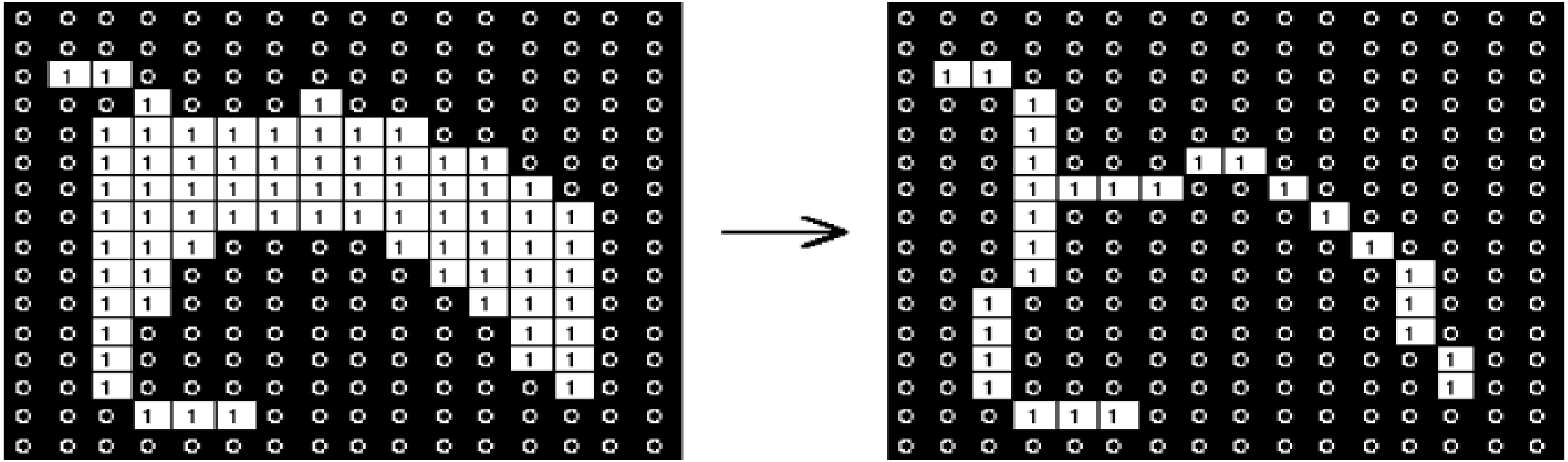
- Example for a Hit-and-miss Structuring Element
- Contains **0's, 1's and don't care's**.
- Usually a "**1**" at the origin!

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

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

# An Example

We use two Hit-and-miss Transforms



# Corner Detection with Hit-and-miss Transform



- Structuring Elements representing four corners

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

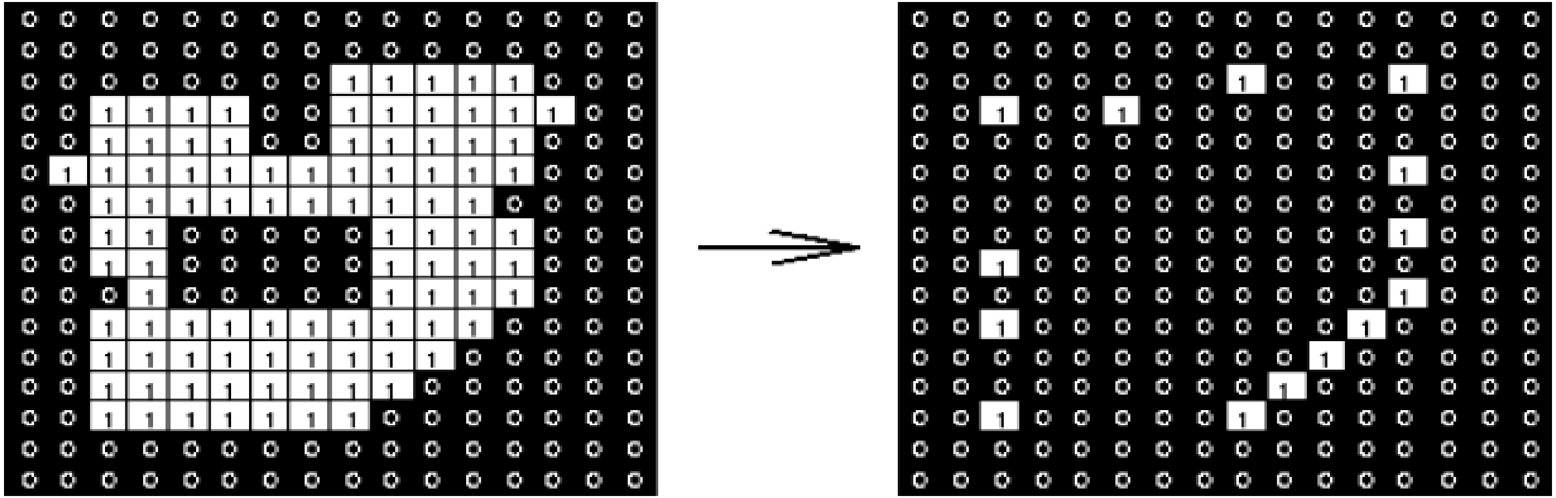
| | | |
|---|---|---|
| | 1 | |
| 1 | 1 | 0 |
| | 0 | 0 |

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

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

# Cont..

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



# Aim of our Research Group:



The aim of our Scientific Innovation research Group (SIRG) to evaluate the IOT performance by propose a secure architecture for the IoT security issues for Education.

# Thanks and Acknowledgement



# Thank you