Binary Image Analysis

Content

1. Thresholding a grayscale image
   - Determine good threshold(s)
2. Binary mathematical morphology
   - Dilation & erosion
   - Opening & closing
3. Connected components (CC) analysis
4. All sorts of feature extractors
   - (area, centroid, circularity, …)

Binary Image Analysis

Binary image analysis

- consists of a set of image analysis operations that are used to produce or process binary images, usually images of 0's and 1's.

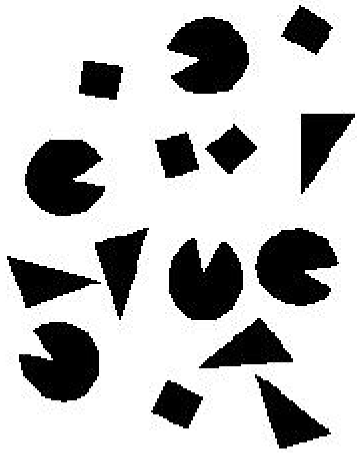
0 represents the background
1 represents the foreground

```
000100100001000
000111100001000
000100100001000
```

Binary Image Analysis

is used in a number of practical applications, e.g.

- part inspection
- riveting
- fish counting
- document processing



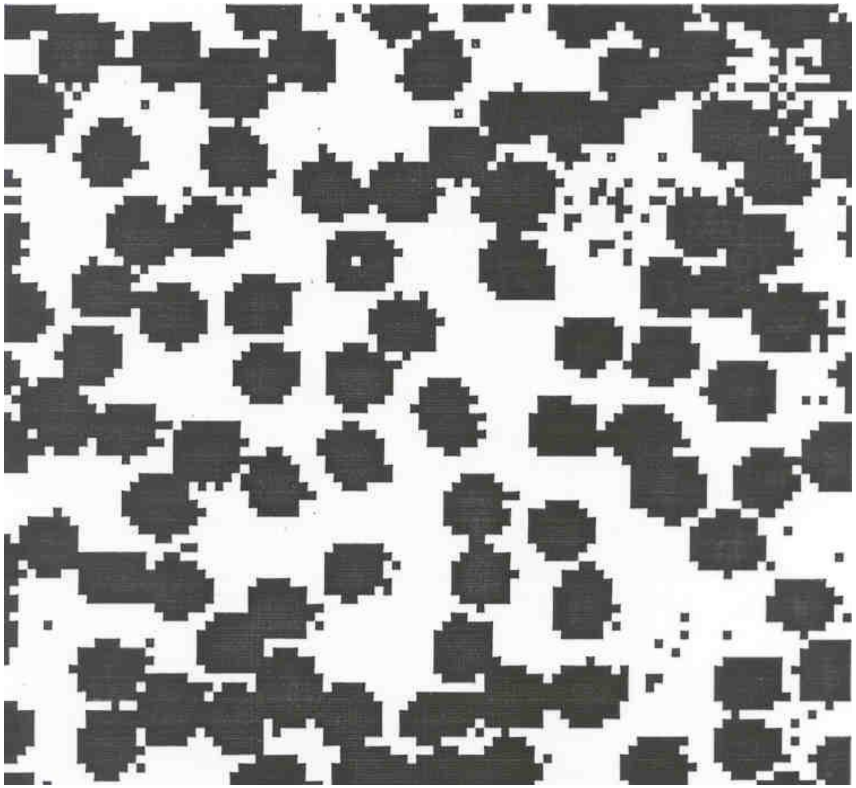
What kinds of operations?

- 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?

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]	

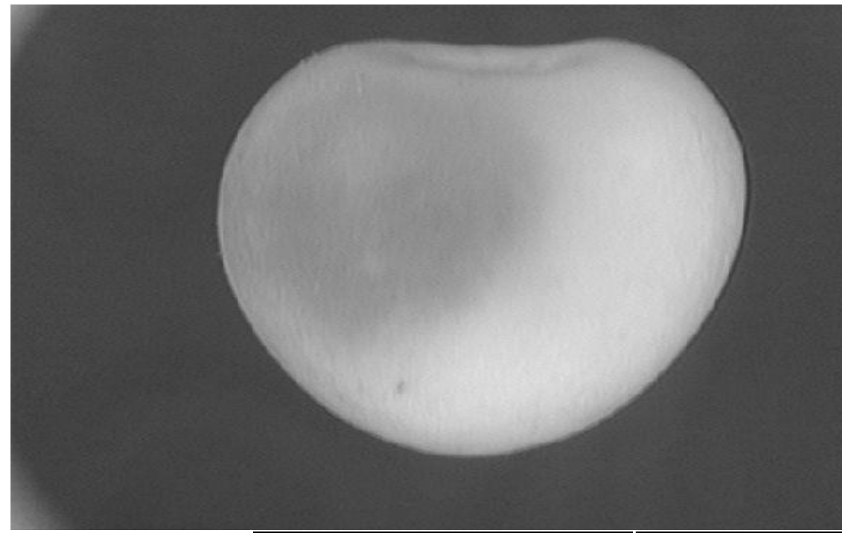
Thresholding

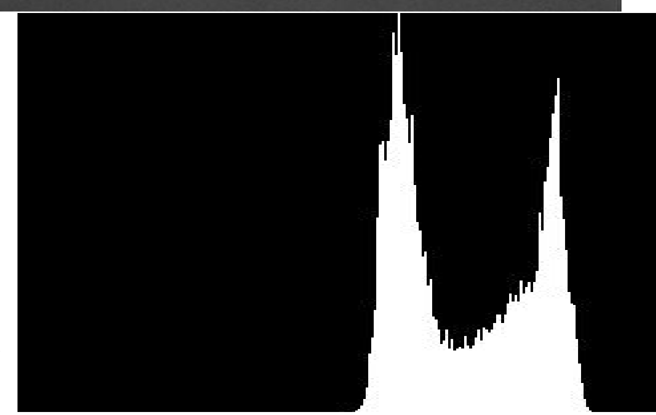
Background is black.

Healthy cherry is bright.

Bruise is medium dark.

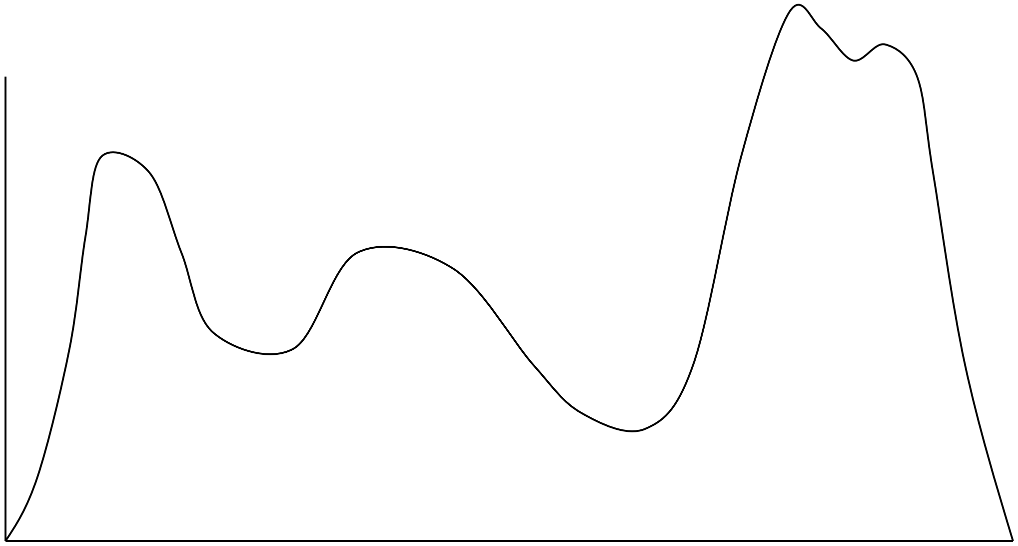
Histogram shows two cherry regions. (black background has been removed.)





Histogram-directed Thresholding

How can we use a histogram to separate an image into 2 (or several) different regions?



Is there a single clear threshold? 2? 3?

Global Thresholding

Assumption: intensity distribution of objects and background pixels are sufficiently distinct.

1. Select an initial estimate of the global threshold, T.

2. Segment the image using T to produce two groups of pixels: $G1$ of all pixels with intensity value > T and $G2$ of all pixels with intensity value >= T.

3. Compute the average intensity values $m1$ and $m2$ for the pixels in $G1$ and $G2$, respectively.

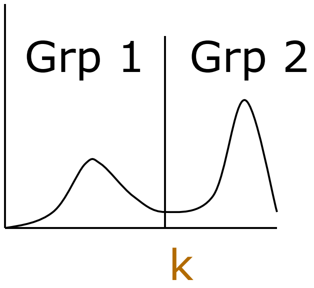
4. Compute a new threshold value:

$$T = \frac{1}{2}(m1 + m2)$$

5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined parameter ΔT.

Otsu's Method

Assumption: intensity distribution of objects and background pixels are sufficiently distinct.



Method: exhaustively search to find the optimal threshold value ***k*** that maximizes the weighted sum of **between-group variance** for the two groups that result from separating the grey level at *k*.

Otsu's Method

1. Compute the normalized histogram of the input image, $p_i, i = 0,1,2, \ldots, L-1$.

2. Computer the cumulative sums,$P_1(k), k = 0,1,2, \ldots, L-1$.

3. Compute the cumulative means, $m(k), k = 0,1,2, \ldots, L-1$.

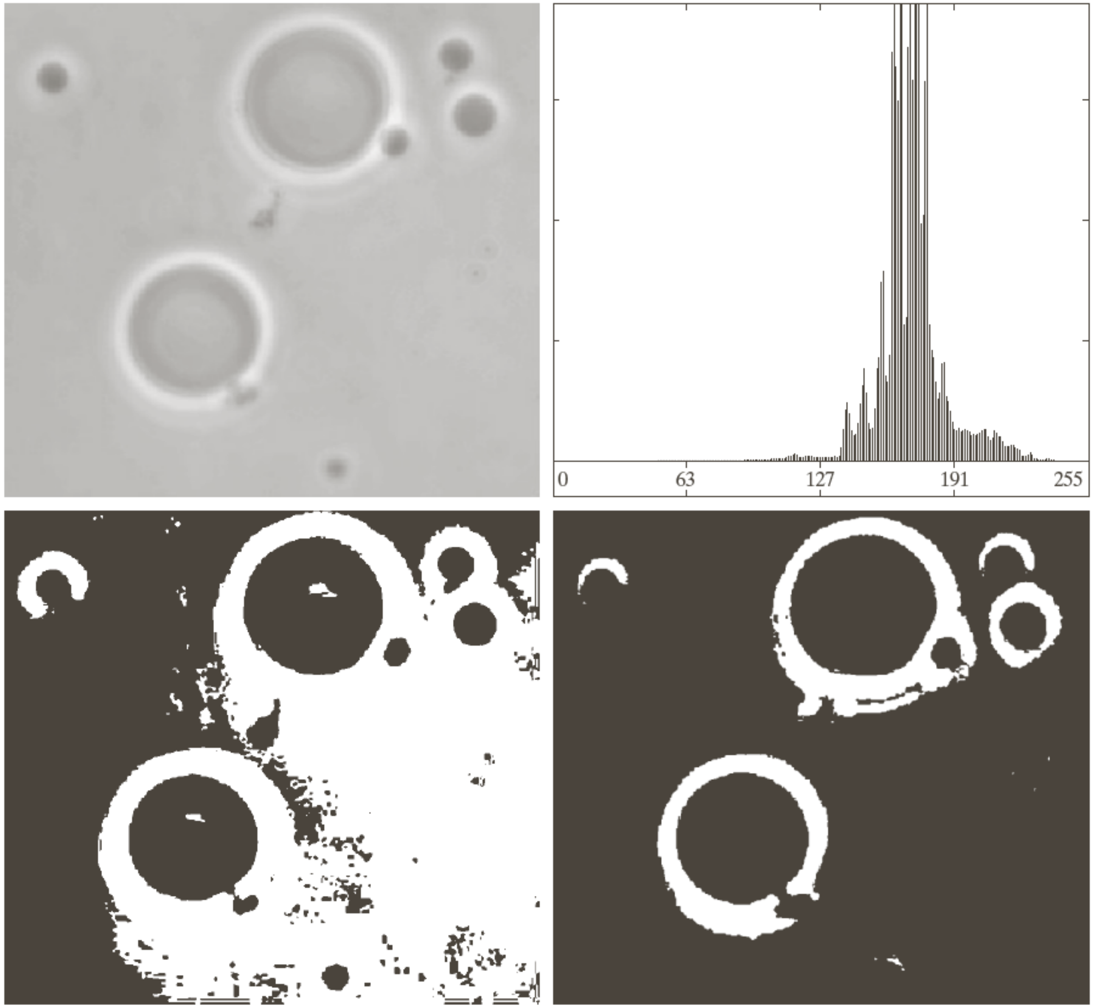
4. Compute the global intensity mean, m_G

5. Computer the between-group variance, $\sigma_B^2(k)$, $k = 0,1,2, \ldots, L-1$

$$\sigma_B^2(k) = \frac{[\, m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

6. Obtain the Otsu threshold k^* with the maximum between-group variance. If the maximum is not unique, obtain k^* by averaging the value of k corresponding to the various maxima detected.

Comparison



a b
c d

FIGURE 10.39 (a) Original image. (b) Histogram (high peaks were clipped to highlight details in the lower values). (c) Segmentation result using the basic global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

Thresholding Example

original grayscale image

binary thresholded image

Mathematical Morphology

Binary mathematical morphology consists of two basic operations

dilation and erosion

and several composite relations

closing and opening

conditional dilation

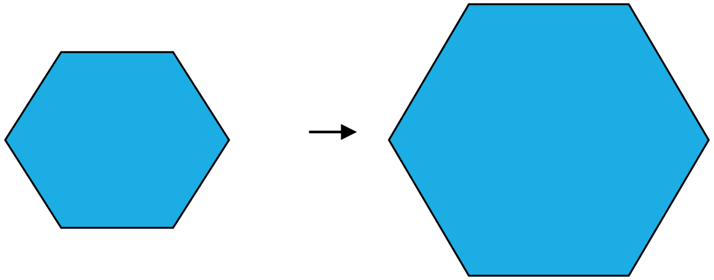
. . .

Dilation

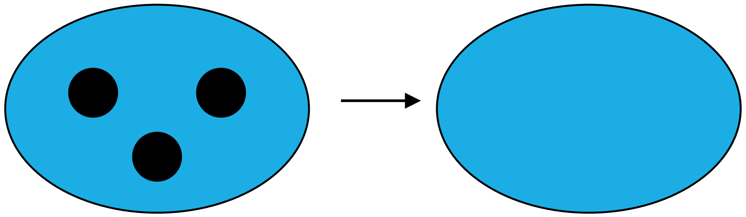
Dilation expands the connected sets of 1s of a binary image.

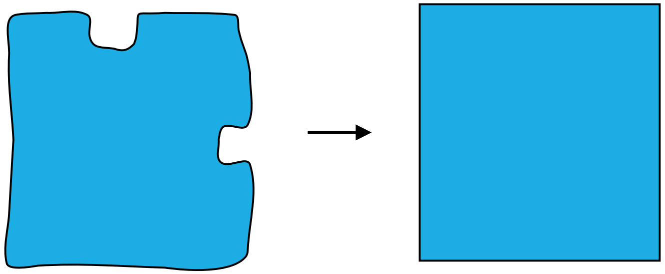
It can be used for

1. growing features



2. filling holes and gaps

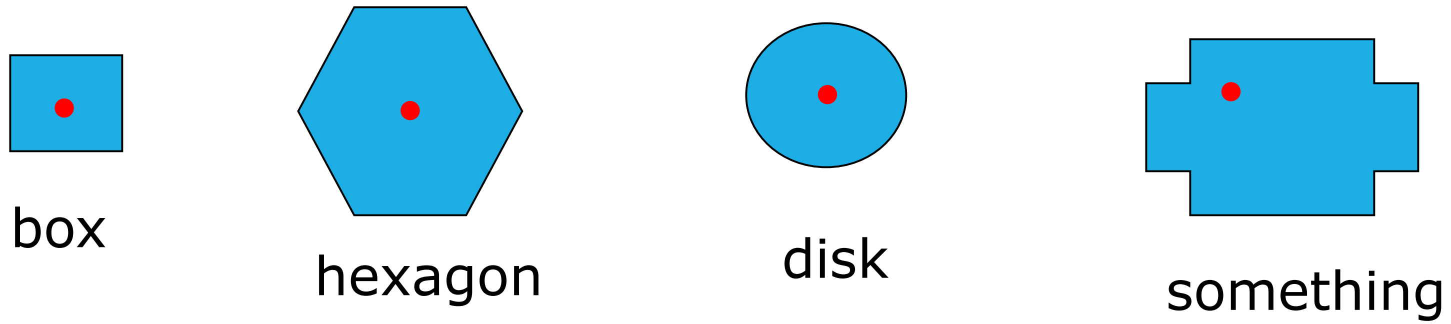




Structuring Element

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.



Dilation

The arguments to dilation and erosion are

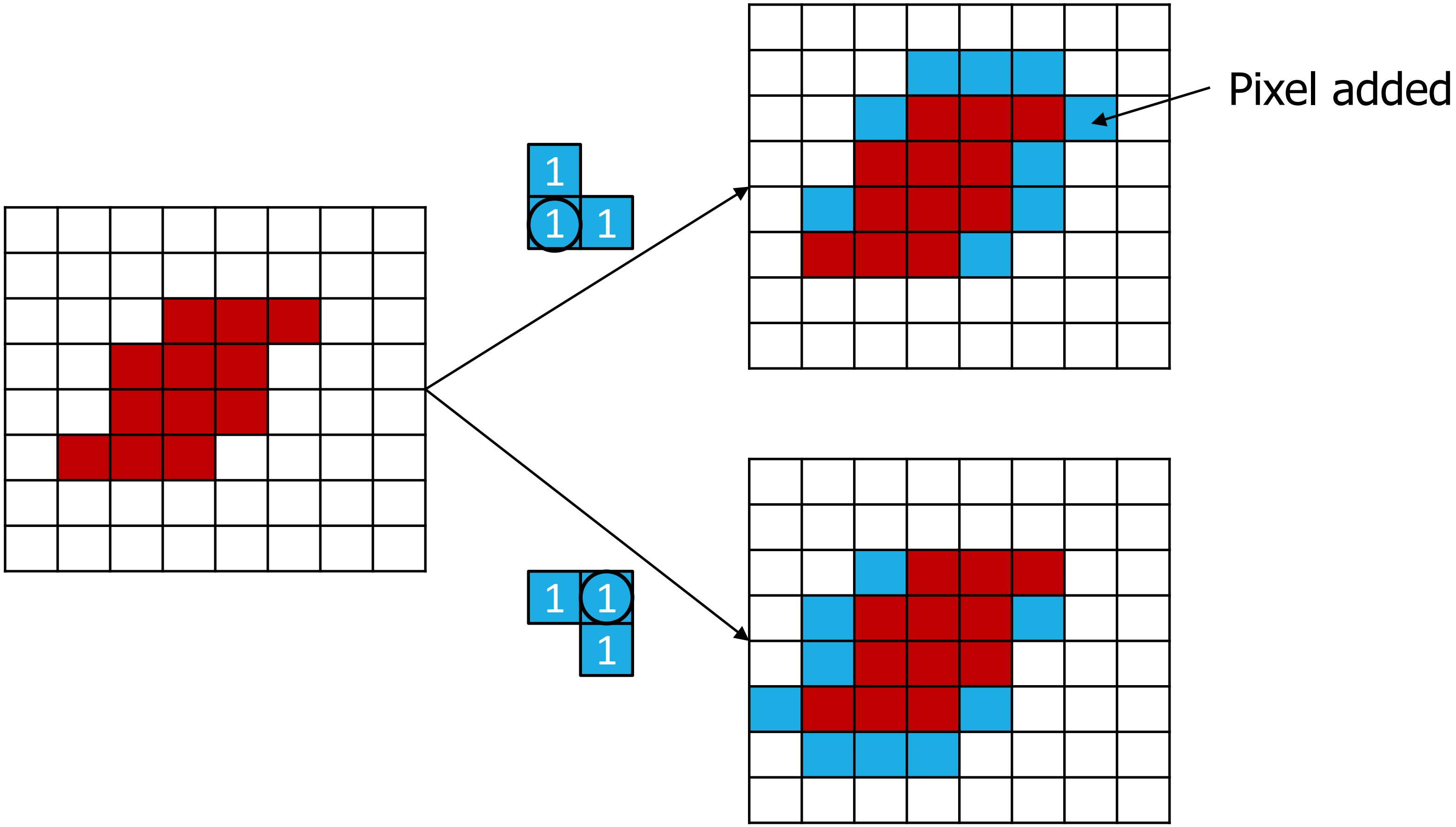
1. **a binary image B**
2. **a structuring element S**

dilate(B,S) takes binary image B, places the origin of structuring element S over each 1-pixel, and ORs the structuring element S into the output image at the corresponding position.

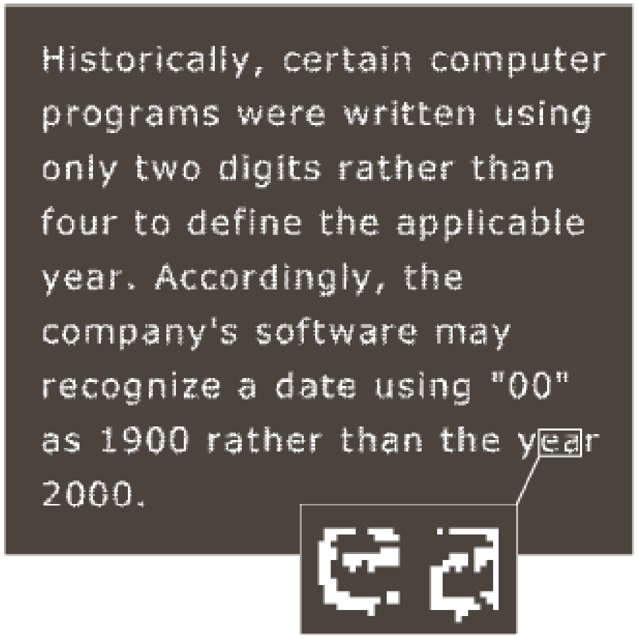
$$\underset{B}{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}} \quad \underset{S}{\begin{bmatrix} 1 & \\ 1 & 1 \end{bmatrix}} \xrightarrow{\text{dilate}} \underset{B \oplus S}{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}$$

origin

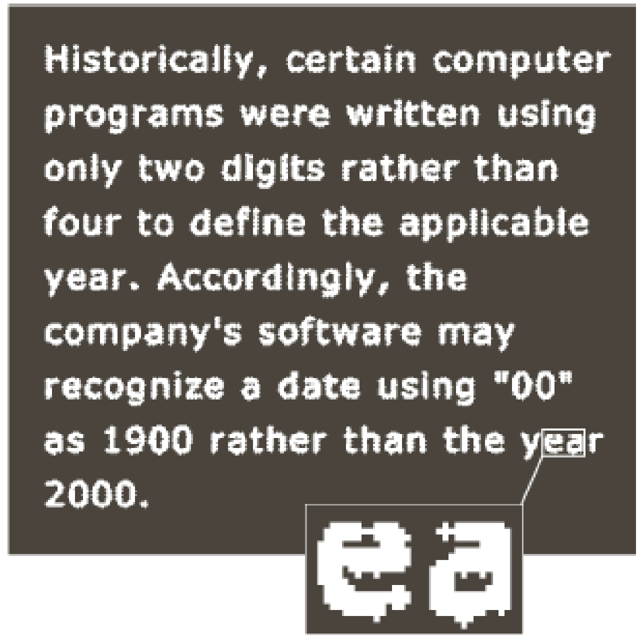
Dilation



Dilation Example-Text



0	1	0
1	1	1
0	1	0



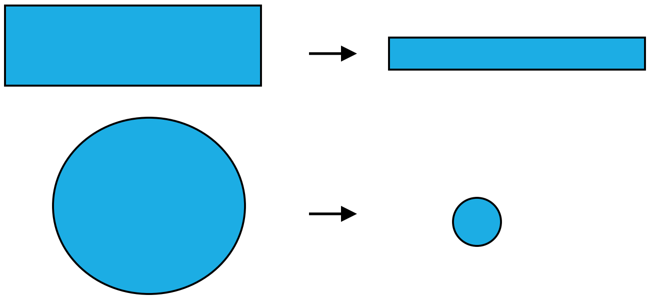
a c
b

FIGURE 9.7
(a) Sample text of poor resolution with broken characters (see magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.

Erosion

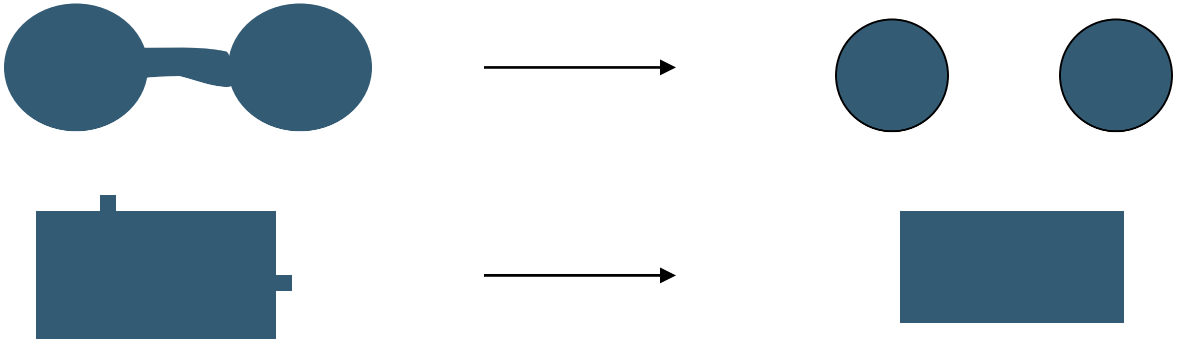
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for



1. shrinking features

2. Removing bridges, branches and small protrusions



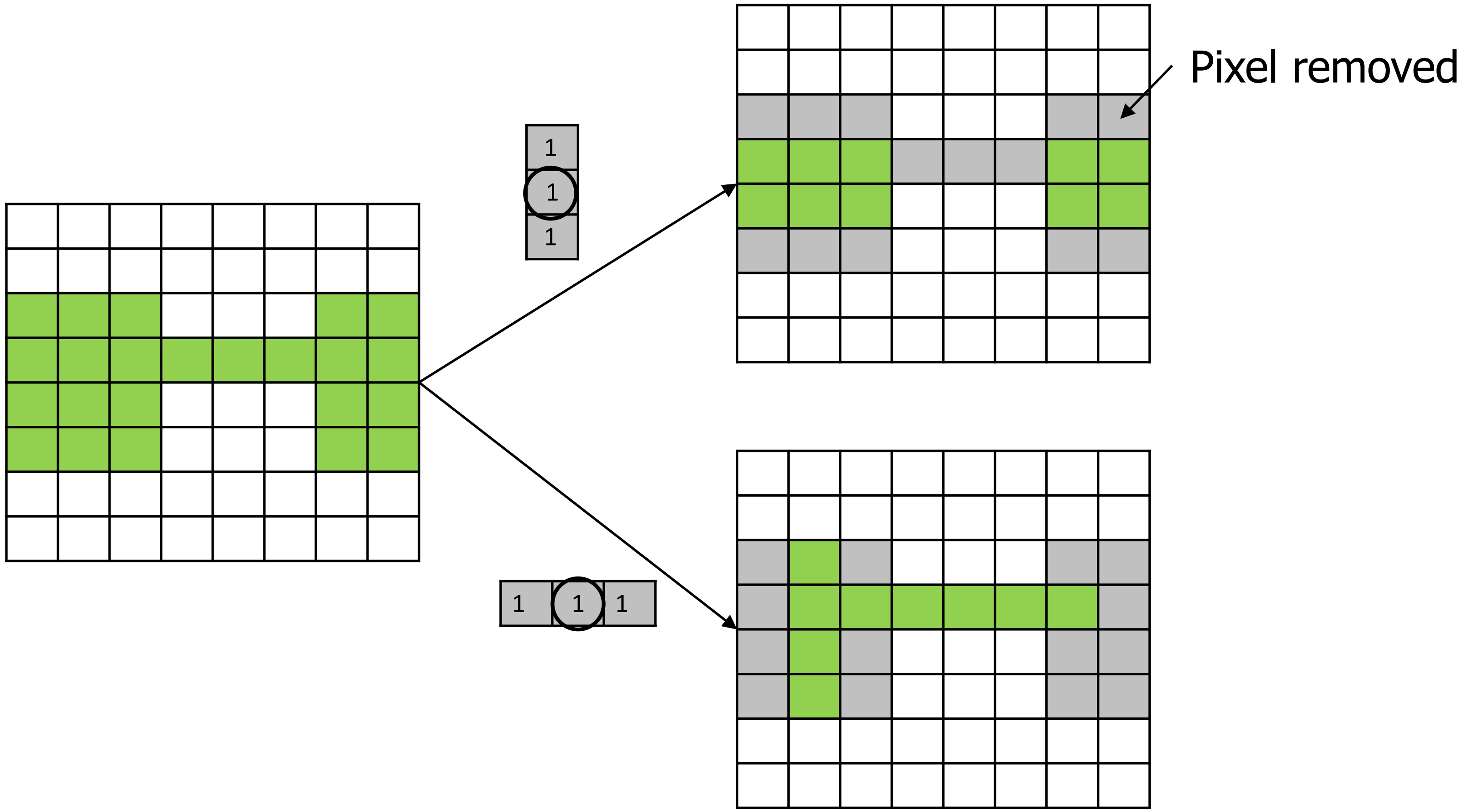
Erosion

erode(B,S) takes a binary image B, places the origin of structuring element S over every pixel position, and ORs a binary 1 into that position of the output image only if every position of S (with a 1) covers a 1 in B.

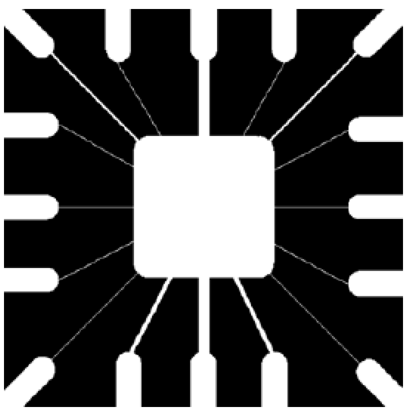
origin

$$\underset{B}{\boxed{\begin{matrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}}} \qquad \underset{S}{\boxed{\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}}} \quad \xrightarrow{\text{erode}} \quad \underset{B \ominus S}{\boxed{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}}}$$

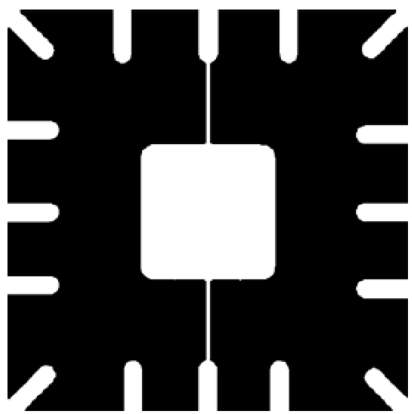
Erosion



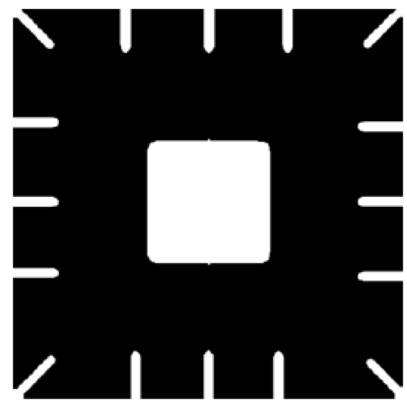
Effect of disk size on erosion



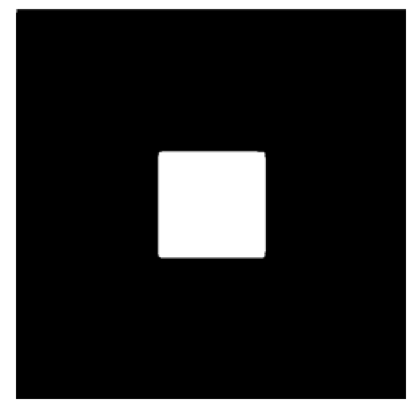
Original image



Erosion with a disk of radius 5



Erosion with a disk of radius 10



Erosion with a disk of radius 20

Erode then dilate

B

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

S

1	1	1
1	1	1
1	1	1

erode

?

dilate with same structuring element

?

Erode then dilate

B

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

S

1	1	1
1	1	1
1	1	1

erode

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

dilate with same structuring element

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

OPENING

Dilate then Erode

B

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

S

1	1	1
1	1	1
1	1	1

dilate

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

erode with same structuring element

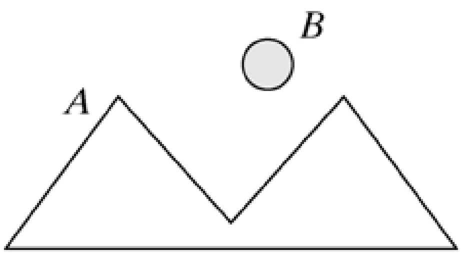
0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
1	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

CLOSING

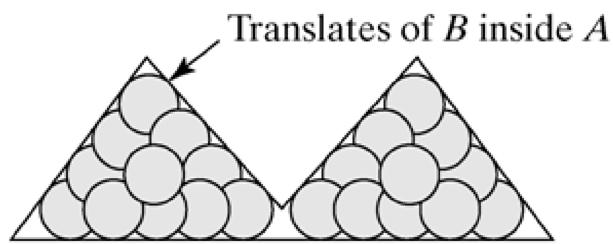
Opening

- Opening is the compound operation of **Erosion followed by Dilation** (with the **same** structuring element).
- Opening is to remove some foreground pixels from the edges of foreground regions and preserve foreground regions that can completely contain the structuring element.
- Opening is less destructive to the shape of the foreground pixels than erosion.

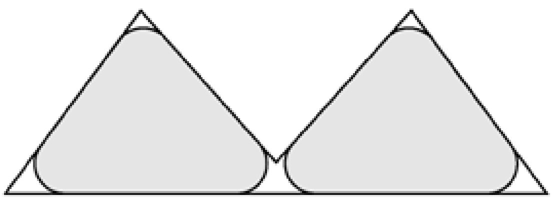
Opening



Binary image A and structuring element B.



Translations of B that fit entirely within A.

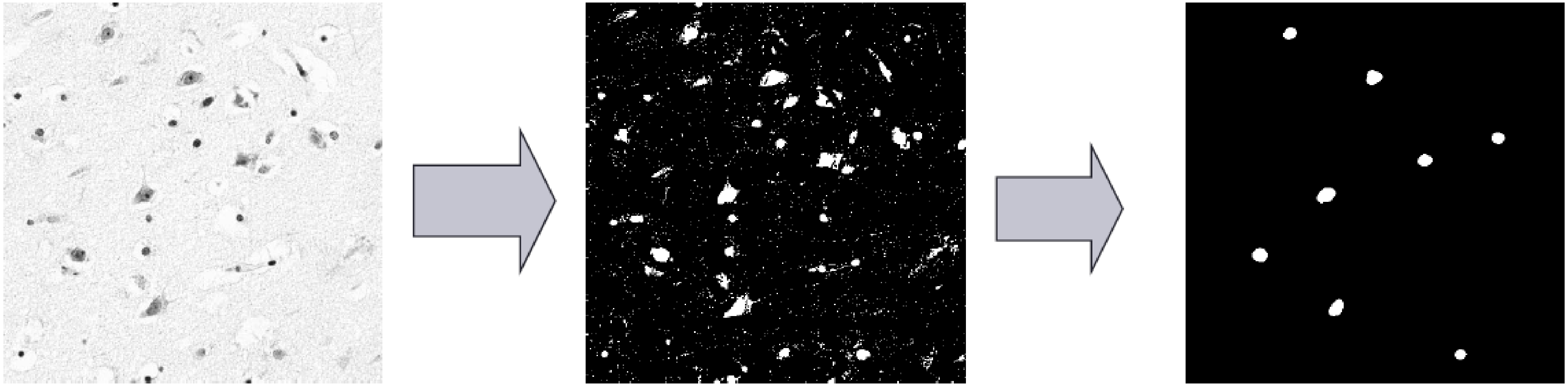


The opening of A by B is shown shaded.

Intuitively, the opening is the area we can paint when the brush has a footprint B and we are not allowed to paint outside A.

Opening example-Cell colony

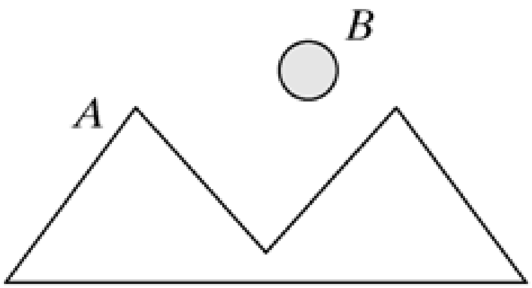
Use large structuring element that fits into the big objects



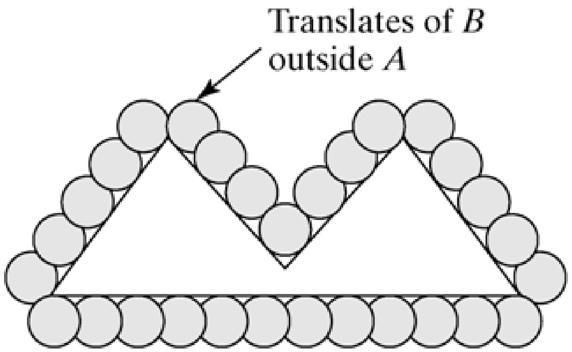
Closing

- Closing is the compound operation of **Dilation followed by Erosion** (with the same structuring element).
- Closing is to enlarge the boundaries of foreground regions and shrink background holes in such regions.
- Closing is less destructive to the shape of the foreground pixels than dilation.

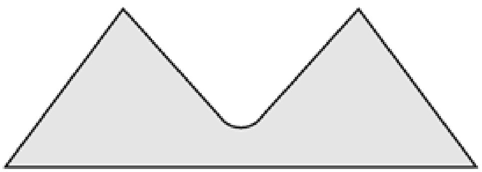
Closing



Binary image A and structuring element B.



Translations of B that do not overlap A.



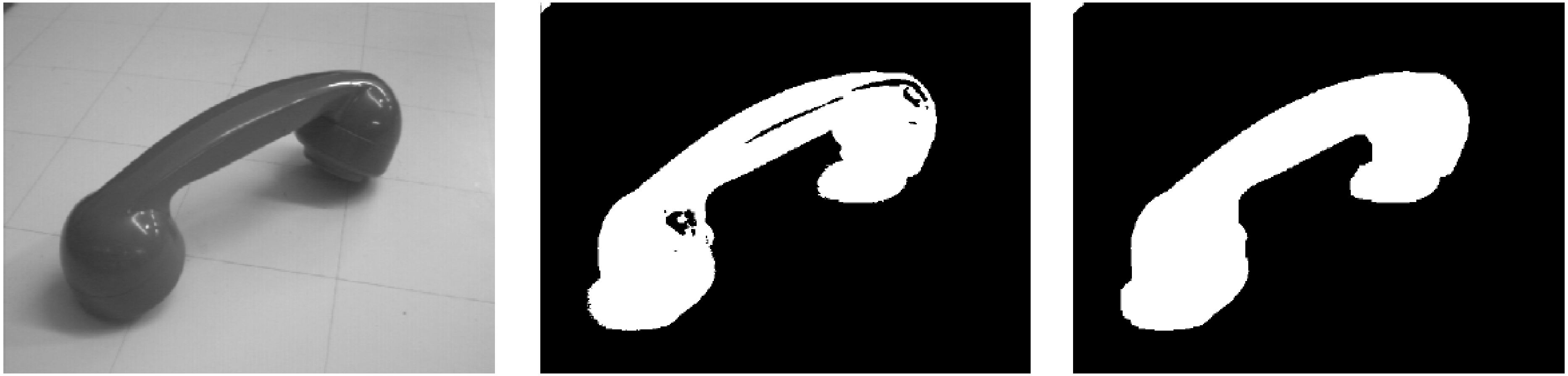
The closing of A by B is shown shaded.

Intuitively, the closing is the area we can not paint when the brush has a footprint B and we are not allowed to paint inside A.

Closing example-segmentation

Simple segmentation:

- Thresholding
- Closing with structuring element of size 20



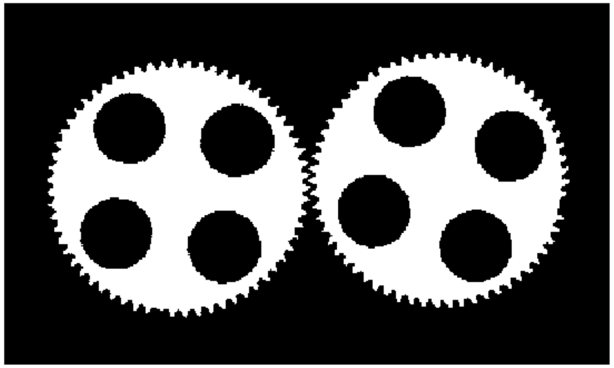
Fingerprint analysis

Original Image | Opening | Opening following by closing

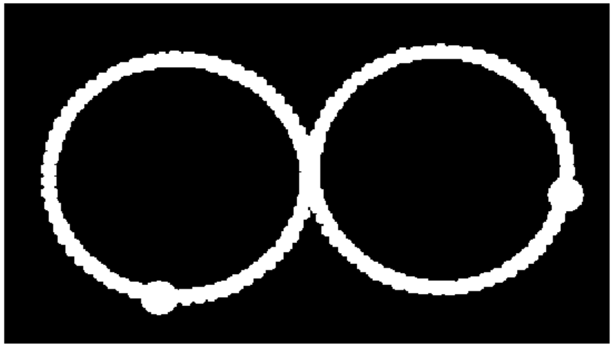
Structuring element used in this case:

1	1	1
1	1	1
1	1	1

Gear Tooth Inspection



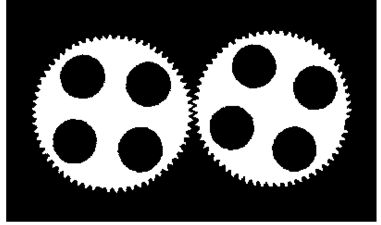
original binary image



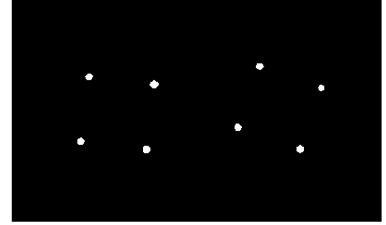
detected defects

How did they do it?

a. Original Image



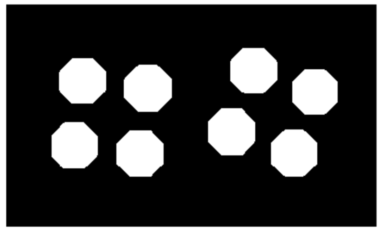
a) original image B



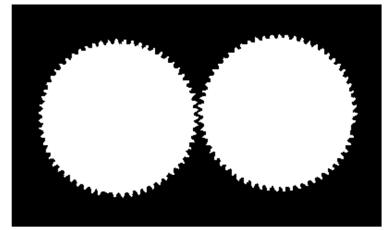
b) $B1 = B \ominus$ hole_ring

b. find the centers of holes by erosion with a ring SE

c. Dilate by a hexagon SE



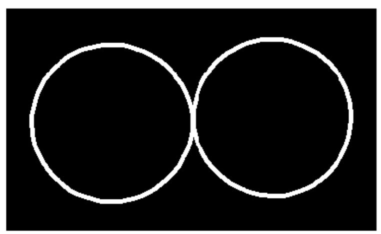
c) $B2 = B1 \oplus$ hole_mask



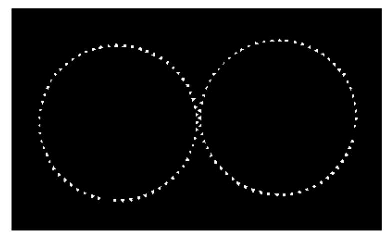
d) B3 = B OR B2

d. OR the hexagons into the original

e. Use disk with the size of the body, open to remove teeth



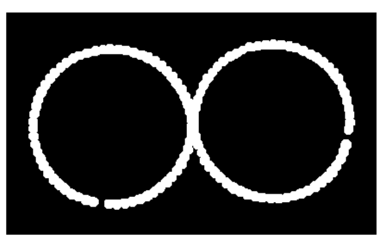
e) B7 (see text)



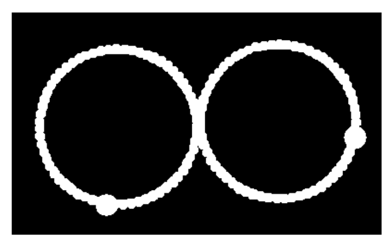
f) B8 = B AND B7

f. AND result of e) with 1) to get the teeth only

g. Dilate d) with a small element that leaves the defects as holes



g) $B9 = B8 \oplus$ tip_spacing



h) RESULT = $((B7 - B9) \oplus$ defect_cue) OR B9

h. Show defects

Connected Components (CC) Labeling

binary image after morphology

Once you have a binary image, you can identify and then analyze each **connected set of pixels**.

The connected components operation takes in a binary image and produces a **labeled image** in which each pixel has the integer label.

connected components

Connected Components

Given a binary image B, the set of all 1's is called the foreground and is denoted by *S*.

Definition: Given a pixel *p* in *S*, *p* is 4-(8) connected to *q* in *S* if there is a path from *p* to *q* consisting only of points from *S*.

- The relation "is-connected-to" is an equivalence relation. It partitions the set S into a set of equivalence classes or components.

A CC labeling algorithm finds all connected components in an image and assigns a unique label to all points in the same component.

Methods for CC labeling

1. Recursive Tracking (almost never used)

2. Parallel Growing (needs parallel hardware)

3. Row-by-Row (most common)

   - Classical Algorithm (two-pass)

   - Efficient Run-Length Algorithm
     (developed for speed in real industrial applications)

Recursive Tracking

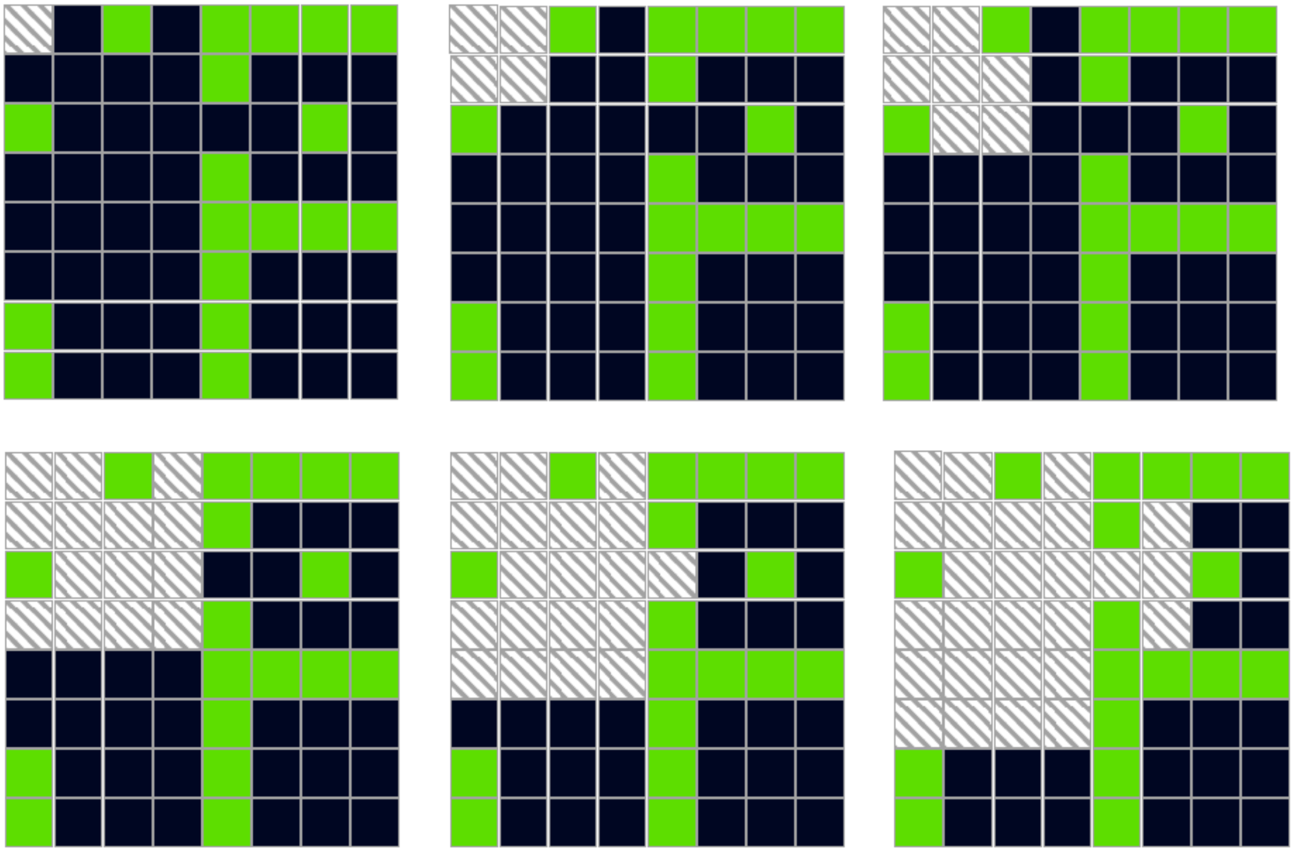
Assume that the foreground pixels are 1-pixels

1. Scan the binary image from top to bottom, left to right until encountering a 1.
2. Change that pixel's label to the next unused component label.
3. Recursively visit all (8-,4-) neighbours of this pixel that are 1's and mark them with the new label.

Drawbacks: requires number of iterations !

Recursive Tracking

Example



Classical Algorithm (two-pass)

Assume that the foreground pixels are 1-pixels

Pass 1:

1. Initialize a label matrix *L* with the same size of *I*.

2. Scan a binary image *I* from left to right, top to bottom.

3. Examine the four scanned neighbours of each 1-pixel in *I*.
   - If all 4 neighbours=0, assign a new label to *L(x,y)*.
   - If only one neighbour=1, assign *the label of that neighbour to L(x, y).*
   - If more than 1 neighbours =1, assign any labels of these neighbours to L(x, y) and record the equivalences.

Pass 2:

1. Use the same label for foreground pixels with equivalences. (i.e. replace each label by the lowest label in its equivalence set)

Example: http://blogs.mathworks.com/steve/2007/05/11/connected-component-labeling-part-5/

Classical Algorithm (two-pass)

Binary image I

0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	1
0	1	1	1	0	0	0	1
0	[illegible]	0	0	0	1	1	1
0	[illegible]	0	0	0	0	0	0

The first foreground pixel is encountered.

L

Label matrix of I

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

If all 4 neighbours = 0, assign a new label to L(x,y).

Classical Algorithm (two-pass)

Binary image I

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

L

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

If all 4 neighbours = 0, assign a new label to L(x,y).

Classical Algorithm (two-pass)

Binary image I

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

L

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

Classical Algorithm (two-pass)

Binary image I

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

L

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

If only one neighbour =1, L(x, y)= the label of that neighbour.

Classical Algorithm (two-pass)

Binary image I

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

L

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

If more than 1 neighbours = 1, L(x, y)=*any labels of these neighbours* and record the equivalences.

Classical Algorithm (two-pass)

Binary image I

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

L

0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

If all 4 neighbours = 0, assign a new label to L(x,y).

Classical Algorithm (two-pass)

Binary image I

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

L

0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0
0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

If more than 1 neighbours = 1, L(x, y)=*any labels of these neighbours* and record the equivalences. (Label 1 <-> Label 2)

Classical Algorithm (two-pass)

Binary image I

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

L

0	0	0	0	0	0	0	0
0	0	0	2	0	0	0	0
0	0	0	1	0	3	3	3
0	1	1	1	0	0	0	3
0	1	0	0	0	4	4	3
0	0	0	0	0	0	0	0

Classical Algorithm (two-pass)

Pass 2:

1. Use the same label for foreground pixels with equivalences.

L

0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	3	3	3
0	1	1	1	0	0	0	3
0	1	0	0	0	3	3	3
0	0	0	0	0	0	0	0

Label equivalences
label 1 ←→ label 2
label 3 ←→ label 4

Efficient Run-Length Algorithm

1. Start at the top row of the image
   a) Partition that row into runs of 0's and 1's
   b) Each run of 0's is part of the background, and is given the special background label.
   c) Each run of 1's is given a unique component label.
2. For all subsequent rows
   1) Partition into runs.
   2) If a run of 1's has no run of 1's directly above it, then it is potentially a new component and is given a new label.
   3) If a run of 1's overlaps one or more runs on the previous row give it the minimum label of those runs.
   4) Let *a* be that minimal label and let $\{c_i\}$ be the labels of all other adjacent runs in previous row. Relabel all runs on previous row having labels in $\{c_i\}$ with *a.*

Run-Length Data Structure

Binary Image

	1	2	3	4	5
1	1	1	0	1	1
2	1	1	0	0	1
3	1	1	1	0	1
4	0	0	0	0	0
5	0	1	1	1	1

ROW	START_ID	END_ID
1	1	2
2	3	4
3	5	6
4	0	0
5	7	7

Runs

ID	ROW	START_COL	END_COL	LABEL
1	1	1	2	0
2	1	4	5	0
3	2	1	2	0
4	2	5	5	0
5	3	1	3	0
6	3	5	5	0
7	5	2	5	0

Run-Length Data Structure

Binary Image

	1	2	3	4	5
1	1	1	0	1	1
2	1	1	0	0	1
3	1	1	1	0	1
4	0	0	0	0	0
5	0	1	1	1	1

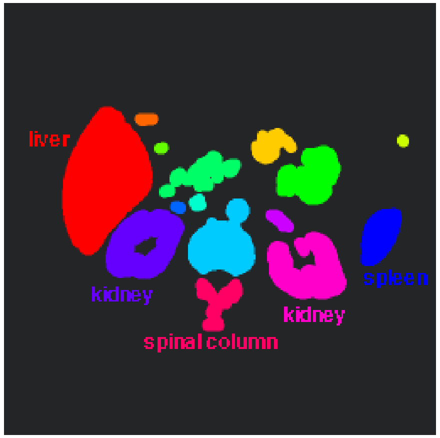
Label Image

1	1	0	2	2
1	1	0	0	2
1	1	1	0	2
0	0	0	0	0
0	3	3	3	3

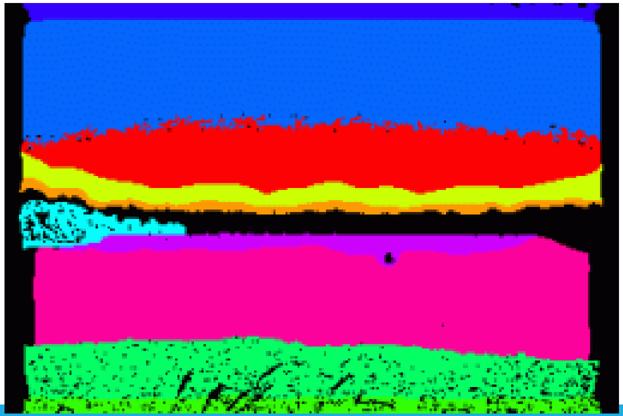
Runs

ID	ROW	START_COL	END_COL	LABEL
1	1	1	2	1
2	1	4	5	2
3	2	1	2	1
4	2	5	5	2
5	3	1	3	1
6	3	5	5	2
7	5	2	5	3

Labeling shown as Pseudo-Colour



connected components of 1's from thresholded image



connected components of cluster labels

Region Properties

Properties of the regions can be used to recognize objects.

- **geometric properties (Ch 3)**
- **gray-tone properties**
- **color properties**
- **texture properties**
- **shape properties (a few in Ch 3)**
- **motion properties**
- **relationship properties (1 in Ch 3)**

Geometric and Shape Properties

- area
- centroid
- perimeter
- perimeter length
- circularity
- elongation
- mean and standard deviation of radial distance
- bounding box
- extremal axis length from bounding box
- second order moments (row, column, mixed)
- lengths and orientations of axes of best-fit ellipse

Q&A