Binary Image Analysis

Agenda

1. Thresholding a grayscale image
   - Determining good thresholds
2. Connected components (CC) analysis
3. Binary mathematical morphology
   - Dilation & erosion
   - Opening & closing

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.

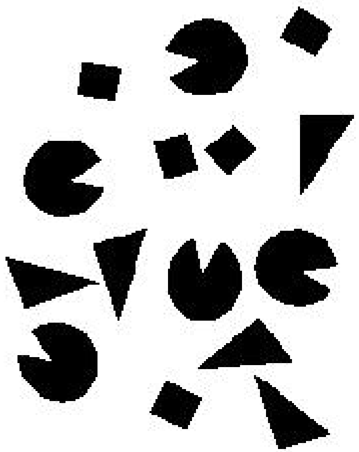
0 represents the background
1 represents the foreground

000100100001000
000111100001000
000100100001000

Binary Image Analysis

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

- part inspection
- riveting
- fish counting
- document processing



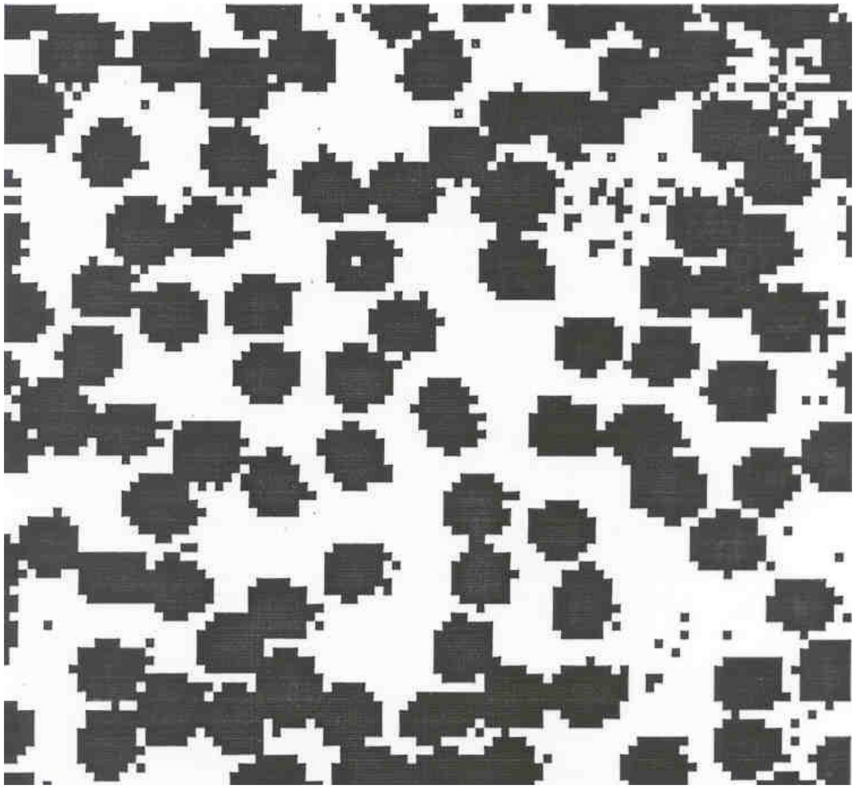
What kinds of operations?

- 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]	

Useful Operations

1. **Thresholding a grayscale image**
2. **Determining good thresholds**
3. **Binary mathematical morphology**
4. **Connected components analysis**
5. **All sorts of feature extractors (area, centroid, circularity, …)**

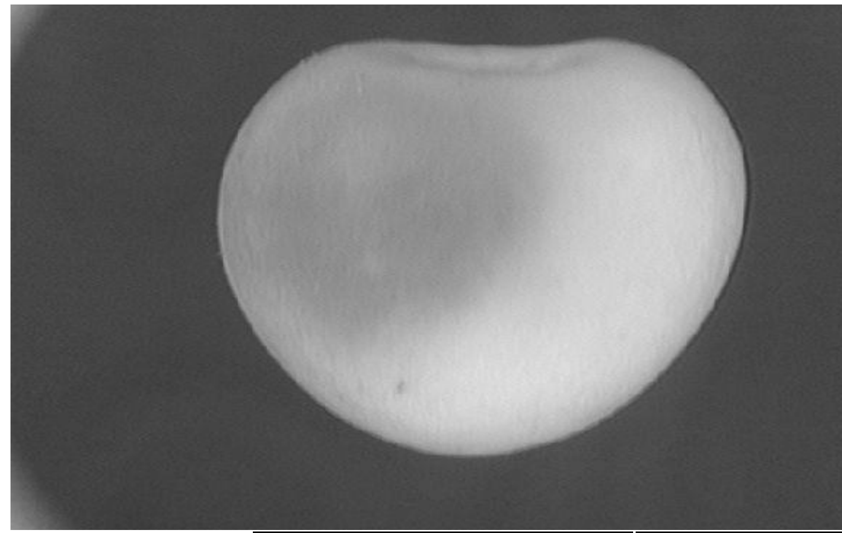
Thresholding

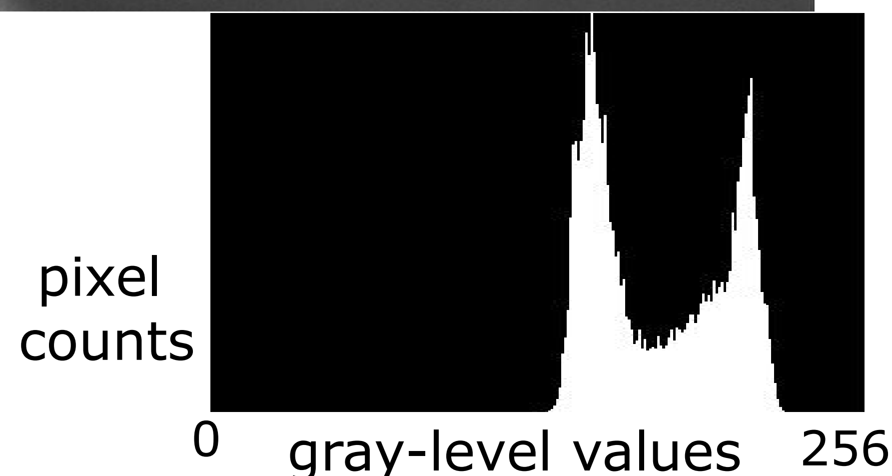
Background is black.

Healthy cherry is bright.

Bruise is medium dark.

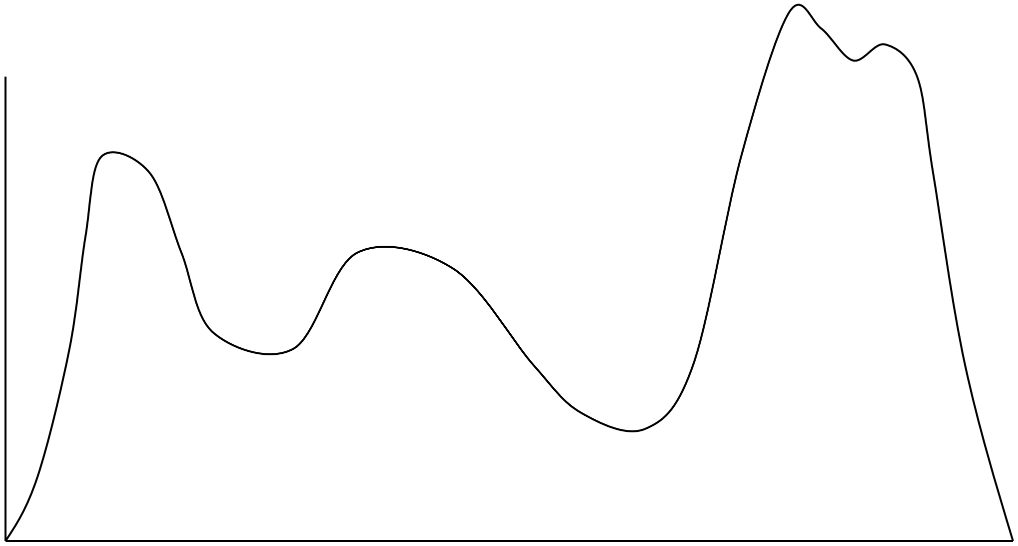
Histogram shows two cherry regions. (black background has been removed.)





Histogram-directed Thresholding

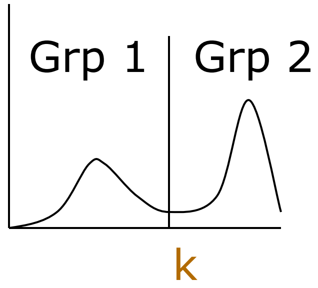
How can we use a histogram to separate an image into 2 (or several) different regions?



Is there a single clear threshold? 2? 3?

Automatic Thresholding: Otsu's Method

Assumption: the histogram is bimodal
(foreground pixels and background pixels)



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

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

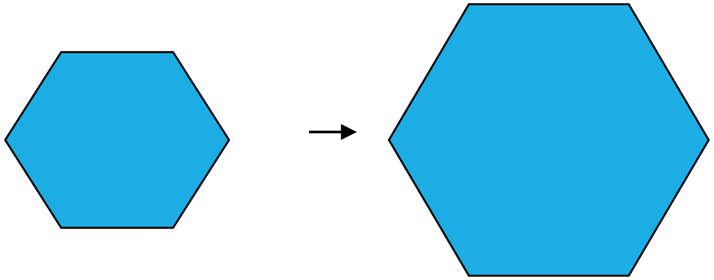
. . .

Dilation

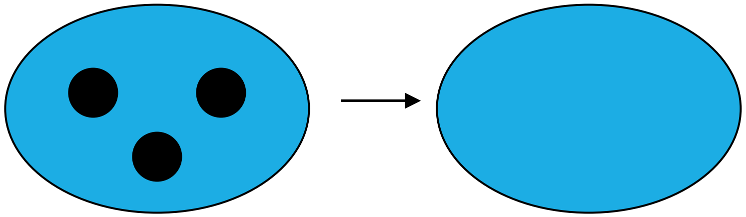
Dilation expands the connected sets of 1s of a binary image.

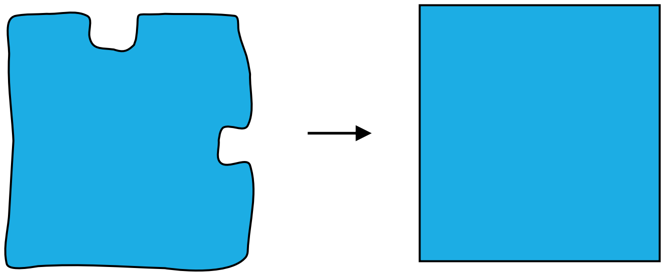
It can be used for

1. growing features



2. filling holes and gaps



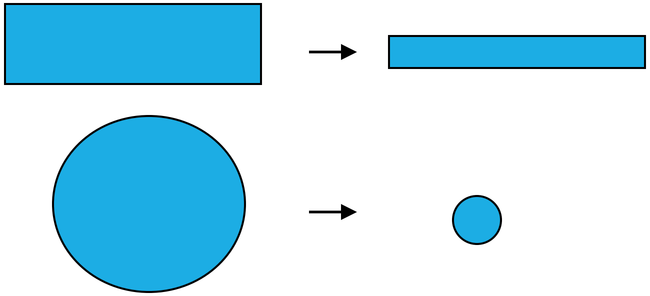


Erosion

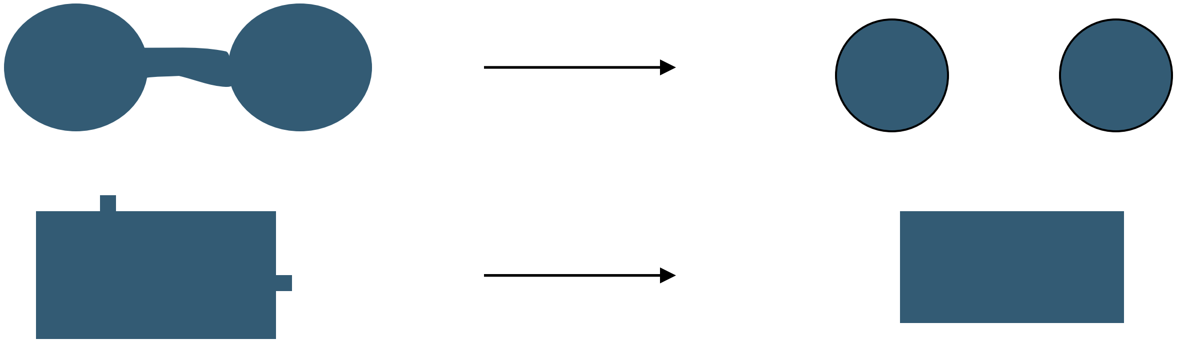
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for

1. shrinking features



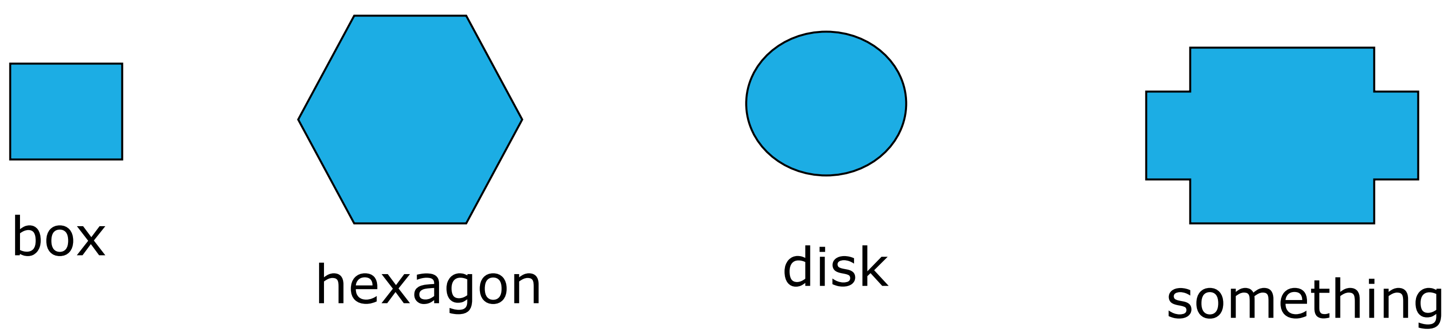
2. Removing bridges, branches and small protrusions



Structuring Elements

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

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



box(length,width)

disk(diameter)

Dilation with Structuring Elements

The arguments to dilation and erosion are

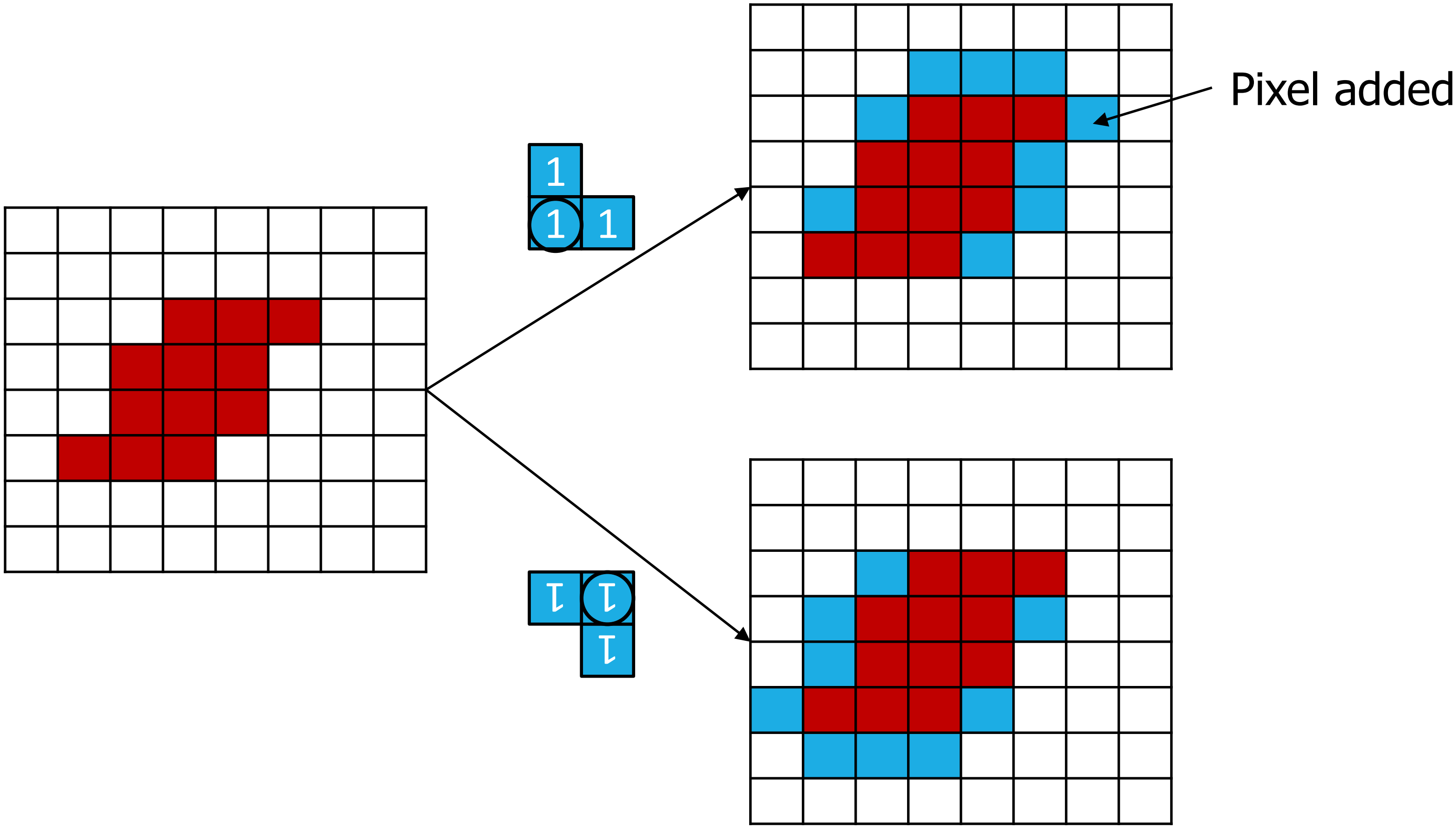
1. **a binary image B**
2. **a structuring element S**

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.

$$
\underset{B}{\begin{array}{|cccc|}\hline 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \hline\end{array}}
\qquad
\underset{S}{\begin{array}{|cc|}\hline 1 & \\ 1 & 1 \\ \hline\end{array}}
\quad \xrightarrow{\text{dilate}} \quad
\underset{B \oplus S}{\begin{array}{|cccc|}\hline 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \hline\end{array}}
$$

origin

Dilation with Structuring Elements



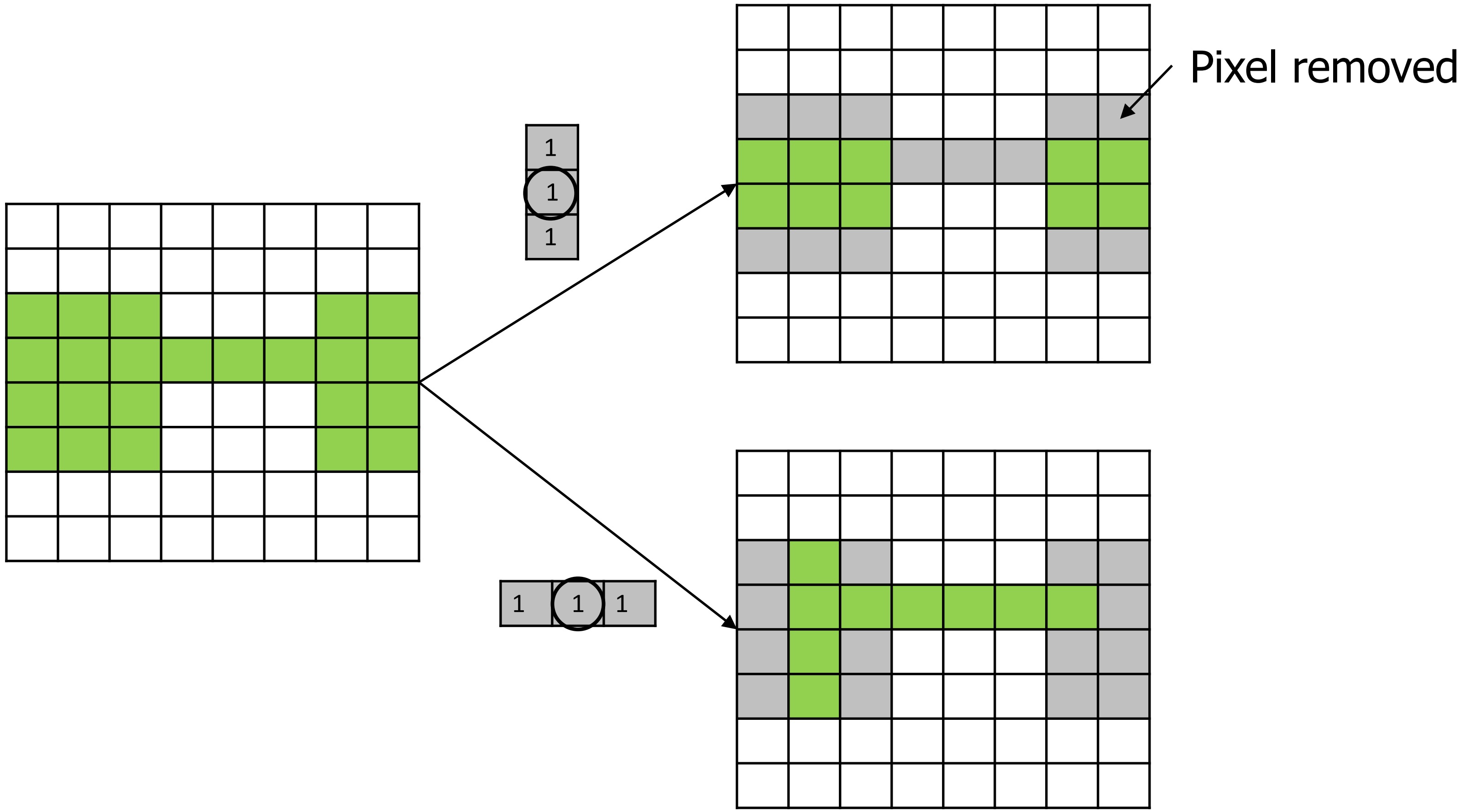
Erosion with Structuring Elements

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.

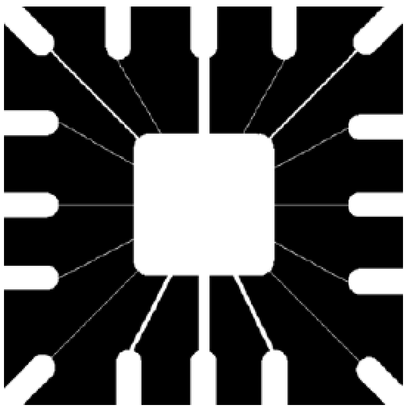
origin

$$\underset{B}{\begin{array}{|ccccc|}\hline 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ \hline\end{array}} \qquad \underset{S}{\begin{array}{|c|}\hline 1 \\ 1 \\ 1 \\ \hline\end{array}} \qquad \xrightarrow{\text{erode}} \qquad \underset{B \ominus S}{\begin{array}{|ccccc|}\hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline\end{array}}$$

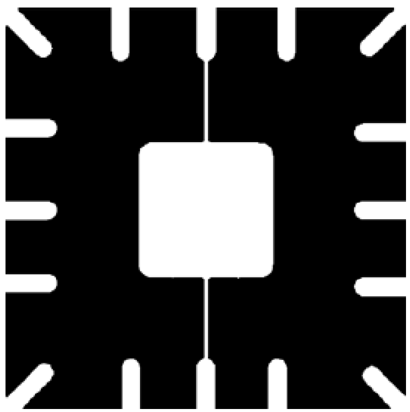
Erosion with different structuring elements



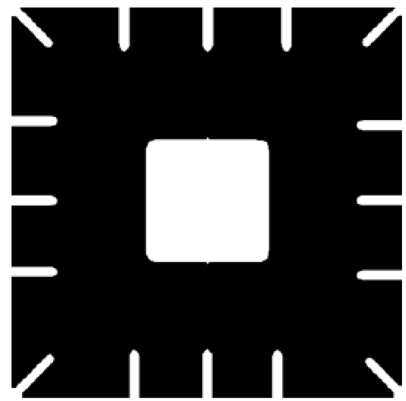
Effect of disk size on erosion



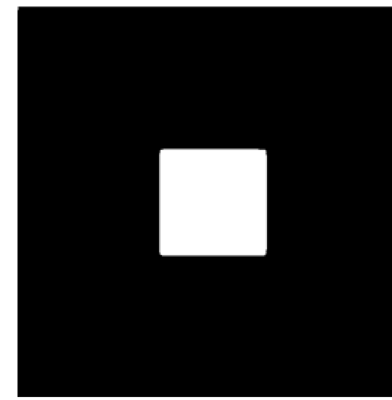
Original image



Erosion with a disk of radius 5



Erosion with a disk of radius 10

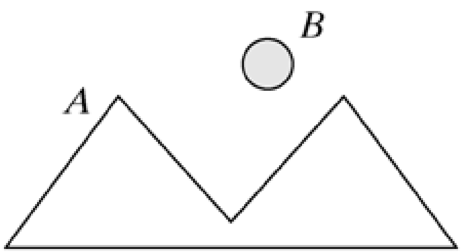


Erosion with a disk of radius 20

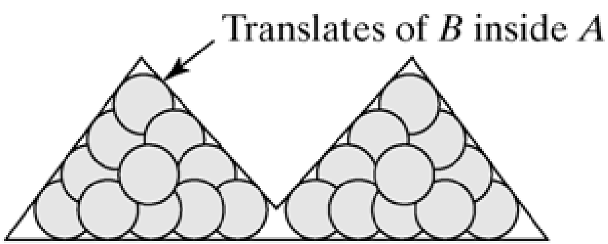
Opening

- 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.

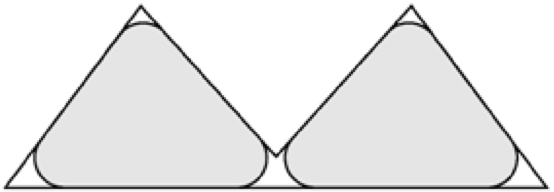
Opening



Binary image A and structuring element B.



Translations of B that fit entirely within A.

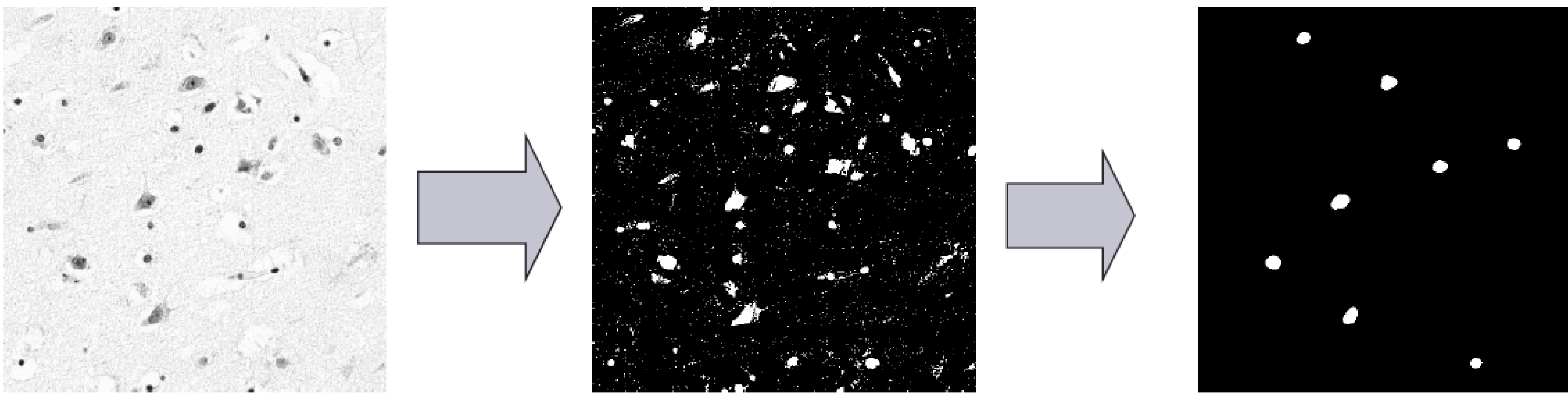


The opening of A by B is shown shaded.

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

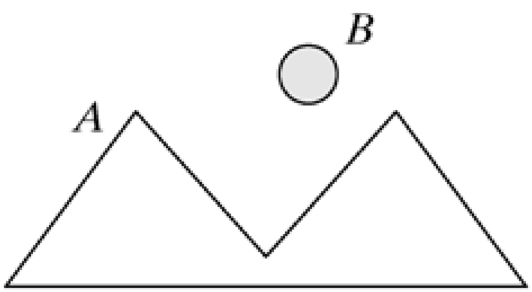
Use large structuring element that fits into the big objects



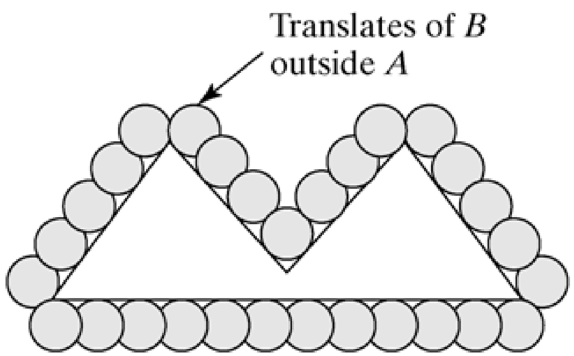
Closing

- 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.

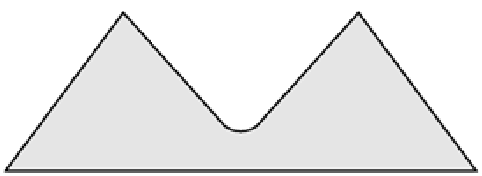
Closing



Binary image A and structuring element B.



Translations of B that do not overlap A.



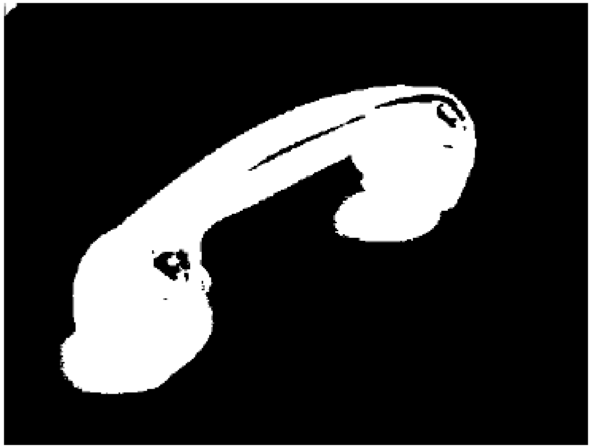
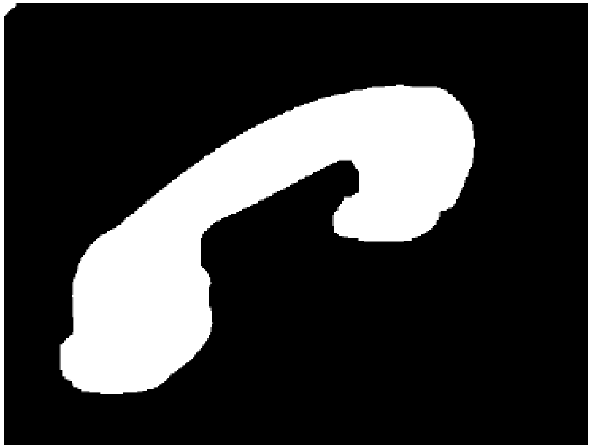
The closing of A by B is shown shaded.

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

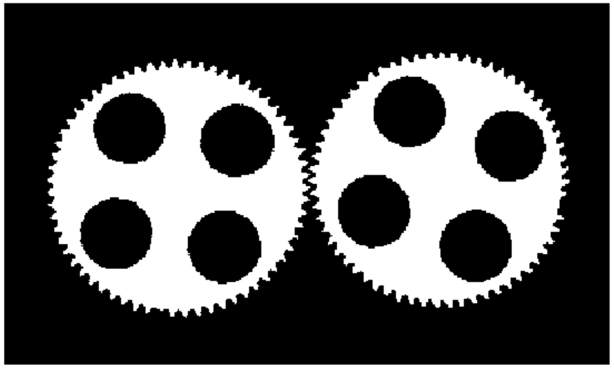
Original Image

Opening

Opening following by closing

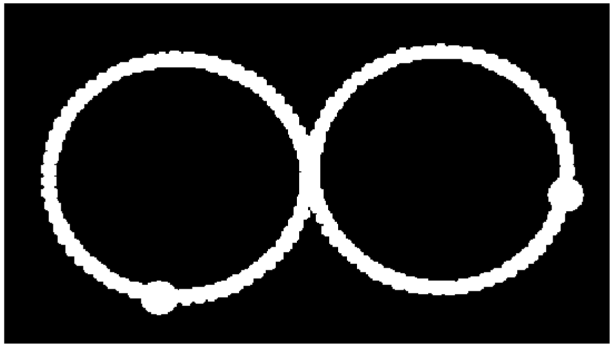
Structuring element used in this case: $\boxed{\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}}$

Gear Tooth Inspection

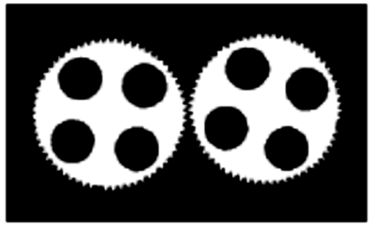


original binary image

How did they do it?

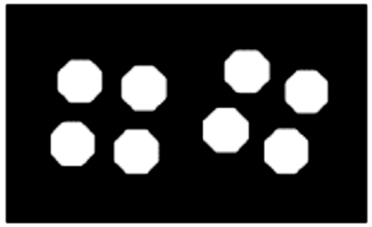


detected defects



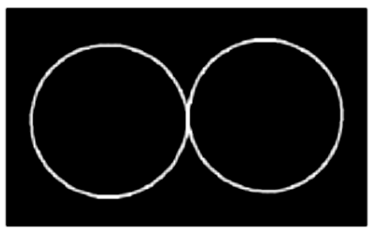
a) original image B

b) B1 = B $\ominus$ hole_ring



c) B2 = B1 $\oplus$ hole_mask

d) B3 = B OR B2



e) B7 (see text)

f) B8 = B AND B7

g) B9 = B8 $\oplus$ tip_spacing

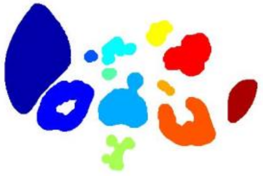
h) RESULT = ((B7 - B9) $\oplus$ defect_cue) OR B9

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 (0) or a component.



connected components

Connected Components

Definition: 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

Methods for CC Analysis

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)

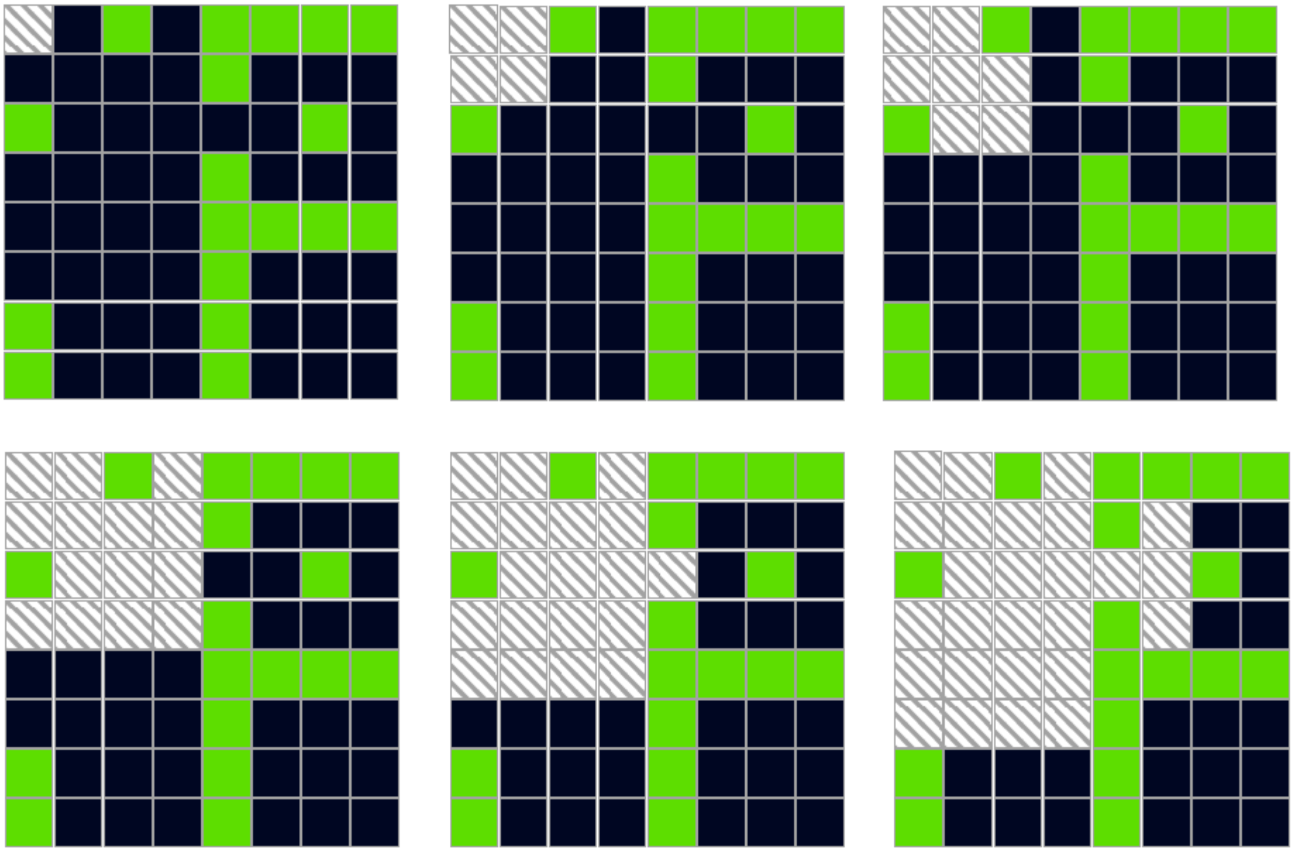
Recursive Tracking

1. Scan the binary image from top to bottom, left to right until encountering a 1(0).
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 (0's) and mark them with the new label.

Drawbacks: requires number of iterations !

Recursive Tracking

Example



Classical Algorithm (two-pass)

Pass 1:

1. scan a binary image I from left to right, top to bottom.
2. initialize a label matrix L with the same size of I.
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.

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

Classical Algorithm demonstration

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		0	0	0	1	1	1
0		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 demonstration

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 demonstration

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 demonstration

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 demonstration

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 demonstration

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 demonstration

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 demonstration

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 demonstration

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

Label equivalence relationships

Set ID	Equivalent Labels
1	label 1 <-> label 2
2	label 3 <-> label 4

Efficient Run-Length Algorithm

1. Start at the top row of the image
   - partition that row into runs of 0’s and 1’s
   - each run of 0’s is part of the background, and is given the special background label.
   - each run of 1’s is given a unique component label.
2. For all subsequent rows
   - partition into runs.
   - 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.
   - if a run of 1’s overlaps one or more runs on the previous row give it the minimum label of those runs.
   - 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	ROW_START	ROW_END
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

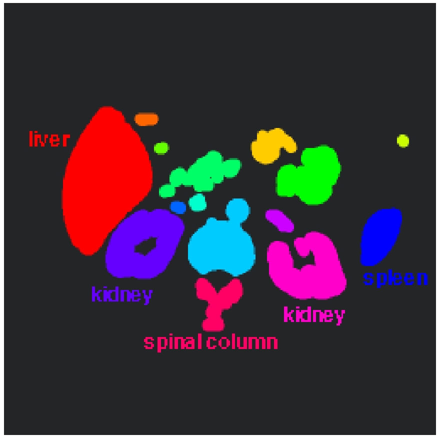
Label Image

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

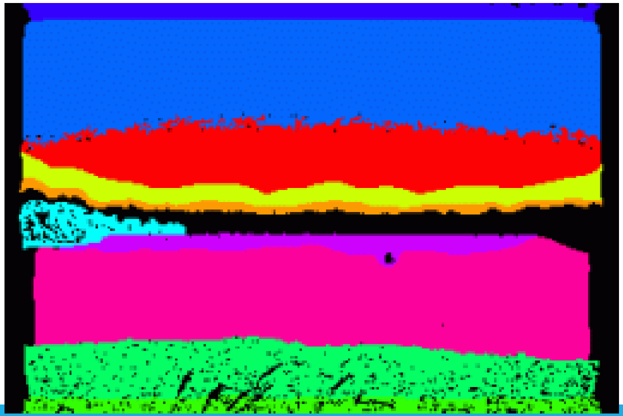
Runs

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-Color



connected components of 1's from thresholded image



connected components of cluster labels

Q&A