Binary Image Analysis

Chapter 3

Dept. of Computer Science & Engineering
Chittagong University of Engineering & Technology

Kaushik Deb, Ph.D.

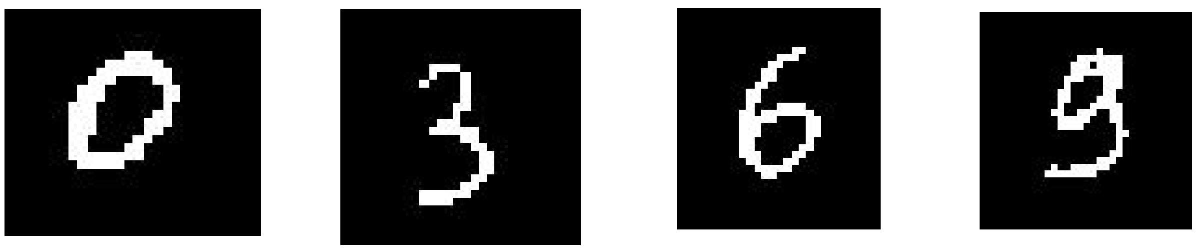
Motivation

- Need less space
- Simpler computational algorithms
- May be extended to grayscale images
- Adequate for many applications
  - Counting
  - OCR
  - Foreground-background separation

Binary Images

- ***Foreground*** : pixels that belong to objects of interest
- ***Background*** : Everything else
- Separation is important
- Simpler in binary images
- OCR :
  - Scan in gray scale
  - Convert to binary
    - Hardware or software

Examples



Binary Image

- Pixels are 0 or 1
  - 1 foreground
  - 0 background
- Assume mxn image
  - Indexed from 0 .. m-1 for rows
  - and 0 .... n-1 for columns
  - I[i,j] : refers to individual pixels

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.

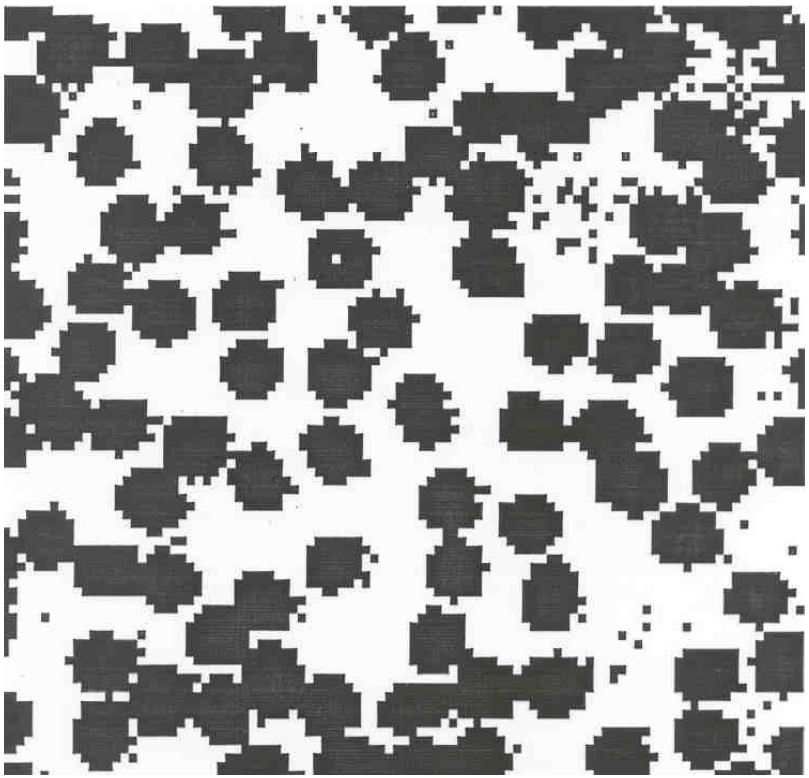
0 represents the background
1 represents the foreground

00010010001000
00011110001000
00010010001000

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

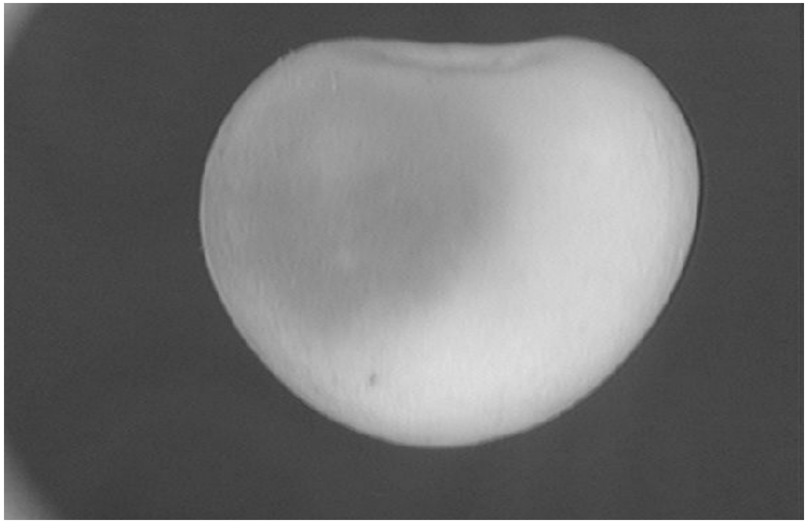
Object	Area	Centroid	Bounding Box	
1	383	(8.8 , 20)	[1 22 1 39]	
2	83	(5.8 , 50)	[1 11 42 55]	
3	11	(1.5 , 57)	[1 2 55 60]	
4	1	(1 , 62)	[1 1 62 62]	
5	1048	(19 , 75)	[1 40 35 100]	gobs
32	45	(43 , 32)	[40 46 28 35]	cell
33	11	(44 , 1e+02)	[41 47 98 100]	
34	52	(45 , 87)	[42 48 83 91]	cell
35	54	(48 , 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]	

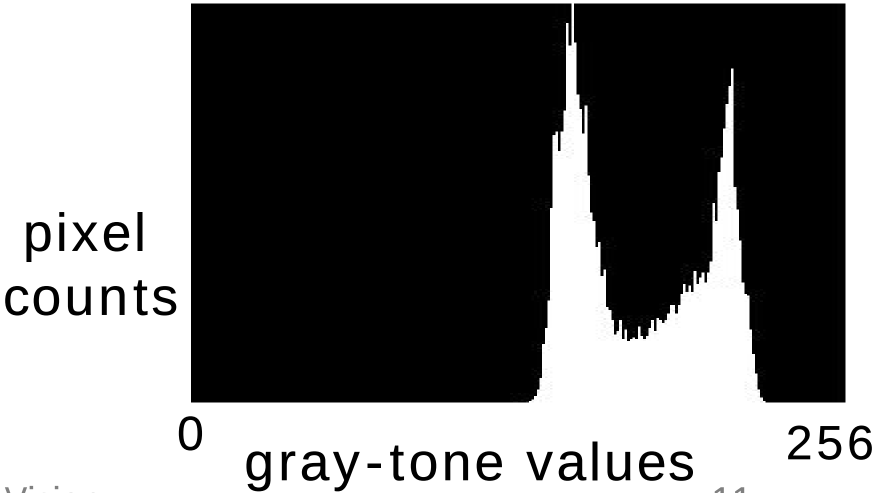
Useful Operations

1. **Thresholding a gray-tone image**
2. **Determining good thresholds**
3. **Connected components analysis**
4. **Binary mathematical morphology**
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)





Thresholding

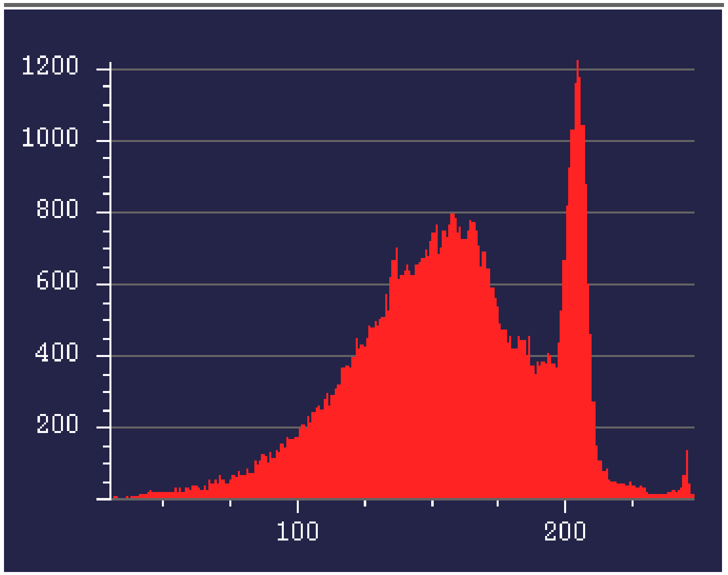
- Common way to convert a gray scale image to a binary image
- Binary Thresholding
  - Map the intensities in the image into 0 or 1
- Multilevel Thresholding
  - Map the intensities in the image into a finite number (usually small) of intensities
- Often guided by the histogram

Histogramming

- Map the frequency of intensities in an image
  - Table of frequencies
  - Graphical representation
- Shows first order properties of an image
  - E.g. Bright/dark
- Modality
  - Number of peaks of a histogram
  - Bimodal, unimodal, etc.

Histogramming

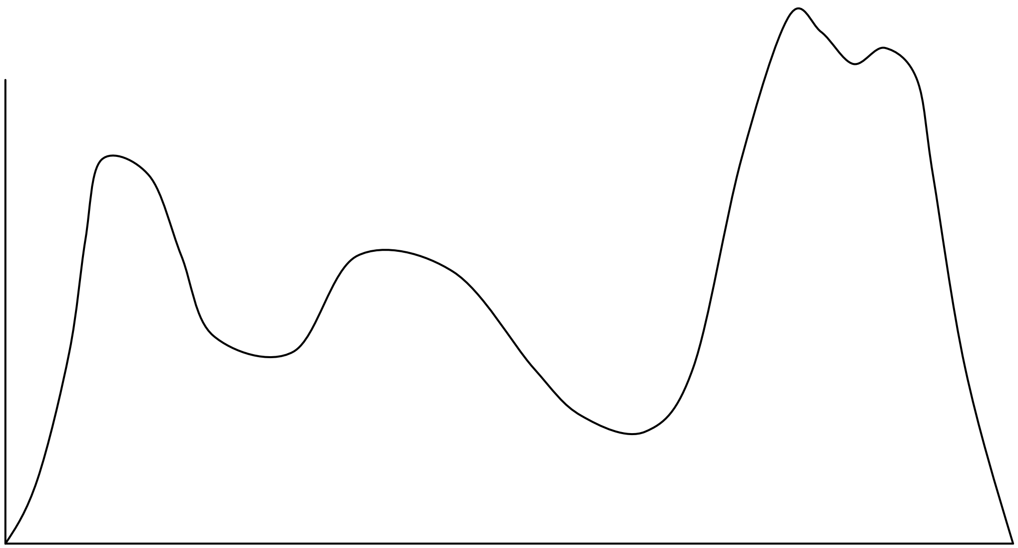
Image



Histogram

Histogram-Directed Thresholding

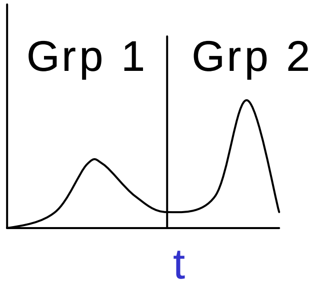
How can we use a histogram to separate an image into 2 (or several) different regions?



Is there a single clear threshold? 2? 3?

Automatic Thresholding: Otsu's Method

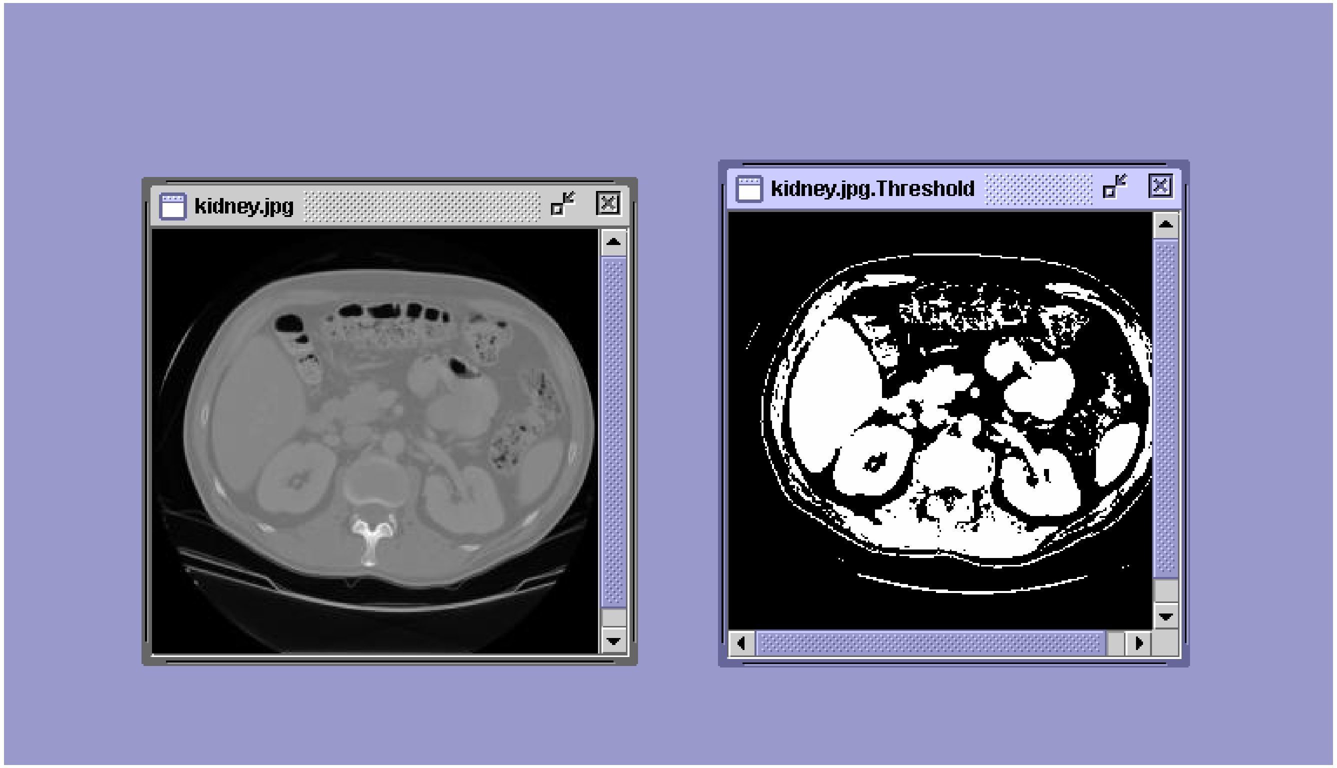
Assumption: the histogram is bimodal



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.

See text (at end of Chapter 3) for the recurrence relations; in practice, this operator works very well for true bimodal distributions and not too badly for others.

Thresholding Example



Another Example

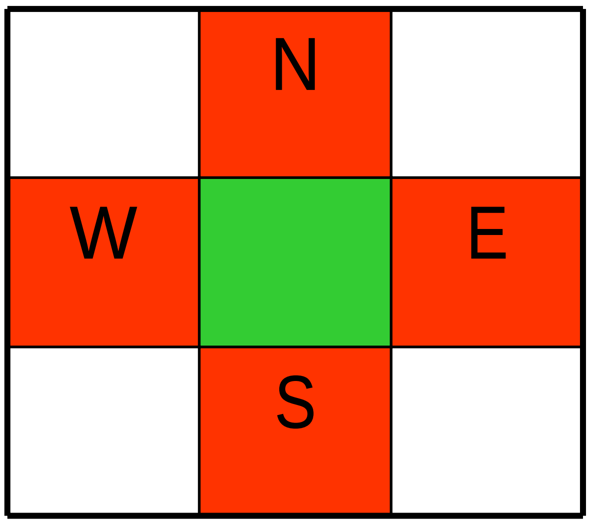
Threshold = 180

Neighborhood

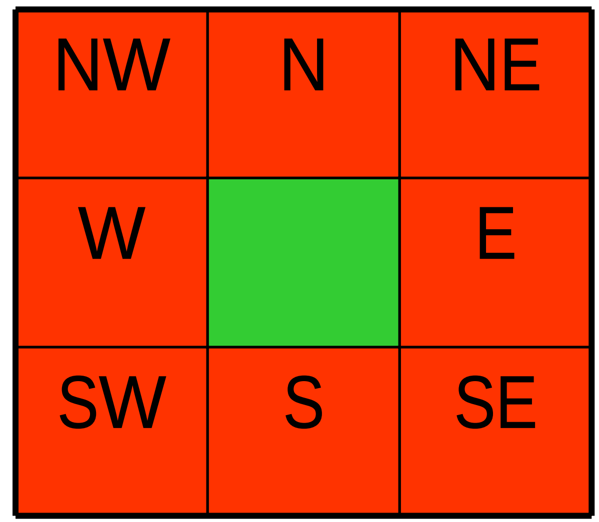
- How many neighbors does a pixel have?

NW	N	NE
W	**P**	E
SW	S	SE

4/8 Neighborhood



4-Neighborhood



8-Neighborhood

Neighborhood Ambiguity

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

- All 1's are 8-neighbors
  - Define a ring
- The red 0's are also 8-neighbors
- Foreground and background objects intersect each other
  - Not simple objects

Managing Ambiguity

- 4-neighborhood for foreground and 8-neighbor for background

- 8-neighborhood for foreground and 4-neighbor for background

Applying Masks

- Basic concept is based on convolution
- Variety of tasks can be framed in terms of mask operations
- Mask = a set of pixels
  - Have associated weights
  - Generally square
  - Has an origin (usually the center pixel)
  - Generally 2D or 1D

Example of Masks

1	1	1
1	1	1
1	1	1

-1	0	1
-1	0	1
-1	0	1

-1	1

Mask Operation

- Overlay the origin of the mask at a pixel
- Pair wise multiply
  - Pixel intensity of the image
  - Weight of the mask at the pixel
- Sum the terms
- Defines the value at that pixel
- Normalize
  - Divide by the size of the mask
- Produces a new image

Example

9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1

-1	0	1
-1	0	1
-1	0	1

Example

Mask

-1	0	1
-1	0	1
-1	0	1

9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	**9**	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1

Image

	0	0	-24	-24	0	
	0	0	-24	-24	0	
	0	0	-24	-24	0	
	0	0	-24	-24	0	
	0	0	-24	-24	0	

Result

Example

- Normalize
  - By the number of elements

	0	0	-24	-24	0	
	0	0	-24	-24	0	
	0	0	-24	-24	0	
	0	0	-24	-24	0	
	0	0	-24	-24	0	

	0	0	-3	-3	0	
	0	0	-3	-3	0	
	0	0	-3	-3	0	
	0	0	-3	-3	0	
	0	0	-3	-3	0	

Example

Mask

-1	0	1
-1	0	1
-1	0	1

9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	**9**	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1
9	9	9	9	1	1	1

Image

0	0	0	0	0	0	0
0	0	0	-24	-24	0	0
0	0	0	-24	-24	0	0
0	0	0	-24	-24	0	0
0	0	0	-24	-24	0	0
0	0	0	-24	-24	0	0
0	0	0	0	0	0	0

Result

Counting Objects

Counting Objects

- Number of foreground objects
- Counting number of holes
  - Equivalent problem
  - Swap the role of foreground and background
- Assumption
  - Objects are 4-connected
  - Foreground pixels are 1's
  - No interior holes

Counting Objects

- External Corners
- Three 0's and one 1

0	0
0	1

0	0
1	0

1	0
0	0

0	1
0	0

- Internal Corners
- Three 1's and one 0

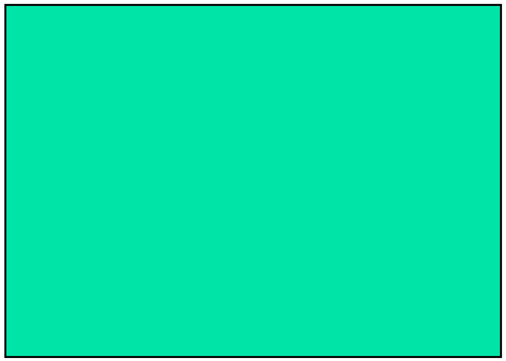
1	1
1	0

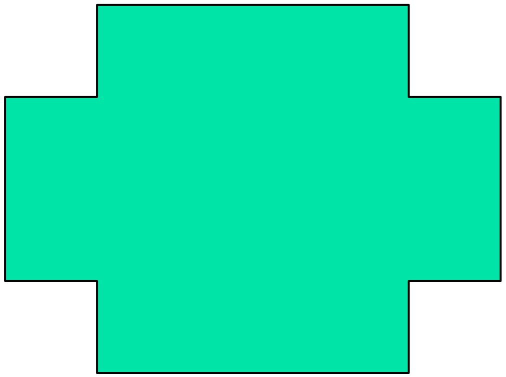
1	1
0	1

0	1
1	1

1	0
1	1

Counting Objects





- External corners=4
- Internal corners=0

- External corners=8
- Internal corners=4

- Number of objects =(External-Internal)/4

Counting Objects

- Take the 4 external masks and 4 internal masks
- Put them at each pixel in the image
- Count the number of E's and I's
  - The number of pixels where they match
- Determine the number of objects based on E and I

Counting Objects

Compute the number of foreground objects of binary image B.

Objects are 4-connected and simply connected.
E is the number of external corners.
I is the number of internal corners.

```
procedure count_objects(B);
{
E := 0;
I := 0;
for L := 0 to MaxRow - 1
    for P := 0 to MaxCol - 1
        {
        if external_match(L, P) then E := E + 1;
        if internal_match(L, P) then I := I + 1;
        } ;
return((E - I) / 4);
}
```

Connected Components

Connected Components

- Given
  - A binary image B
  - A pair of pixels $B[r,c] = B[r',c'] = v$ (0,1)
- Pixel (r,c) is connected to (r',c') if

– there is a sequence of pixels

$$(r,c) = (r_0,c_0), (r_1, c_1), \cdots, (r_n,c_n) = (r',c')$$

– $B[r_i,c_i] = v$

– (r_i,c_i) and (r_{i+1}, c_{i+1}) are adjacent

$(r_0,c_0), (r_1, c_1), \cdots, (r_n,c_n) = (r',c')$ is a connected path from (r,c) to (r',c')

Connected Components

- A connected component is a set of pixels with the same intensity value such that each pair of pixels in the set is connected
- Use 4 or 8 neighborhood
- Use complementary connectedness for foreground and background
- Usually foreground is chosen to be the 1 pixels

Connected Components Labeling

- CCL is a well –known technique in image processing that scans an image (binary and gray level) and labels its pixels connected components based on pixel connectivity. In other words all pixels in connected components share similar pixel intensity values and these components are actually connected with respect of either 4 or 8 –neighborhood definition

Connected Components Labeling

- A connected component labeling of a binary image B is a labeled image in which the value of each pixel is the label of its connected component
- Background label is 0
- Each component has unique label
- Consecutive labels starting with 1

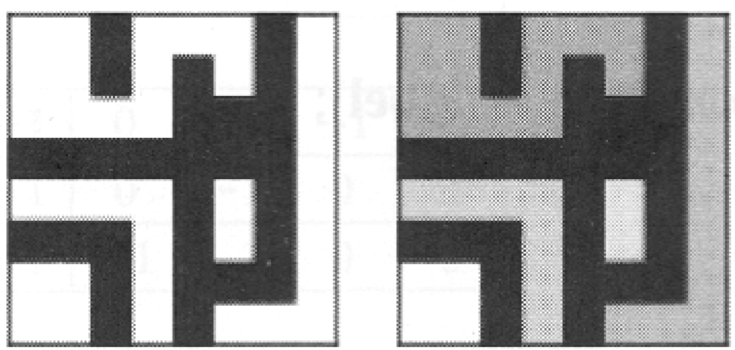
Connected Components

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

(a) Binary image

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

(b) Connected components labeling



(c) Binary image and labeling, expanded for viewing

Recursive Labeling Algorithm

- First negate the binary image
- Finding connected component, becomes one of finding a pixel which values minus one and assigning a label
- Calling procedure search to find its neighbors that have value minus one and recursive repeat the process for its neighbors
- Algorithm 3.2
- Fig 3.7, 3.8

Connected Components Labeling
Classical Algorithm

- Take a template and go over the image
- 2 pass algorithm
  - Make an initial labeling
  - Make a final labeling

	Top
Left	Current

Connected Components Labeling
Classical Algorithm

	Top 0
Left 0	Current 1

	Top 0
Left 1	Current 1

	Top 1
Left 0	Current 1

	Top 1
Left 1	Current 1

Connected Components Labeling

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

(a) Binary image

Classical Algorithm
First Pass

```
currentlabel=1
for (i=0; i<nrows; i++)
    for(j=0; j<ncols; j++)
        if (Image[i,j]==0) break;
        if ((top==0) && (left==1) label (current) = label(left);
        if ((top==1) && (left==0) label (current) = label(top);
        if ((top==0) && (left==0) {
            label (current) = currentlabel;
            currentlabel++;
        }
        if ((top==1) && (left==1) {
            label (current) = label(left);
            equivelent(label(left), label(top));
        }
```

Classical Algorithm

Second Pass

```
for (i=0; i<nrows; i++)
    for(j=0; j<ncols; j++)
        if (Image[i,j]==0) break;
        currentlabel=label(i,j);
        label(i,j)=equivalentclass(currentlabel);
```

Classical Algorithm
Union Find

- Need to merge different labels for the same connected component
- We have pairwise equivalences
- Need to merge them into equivalent classes
- Union Find algorithm is an efficient algorithm to do this

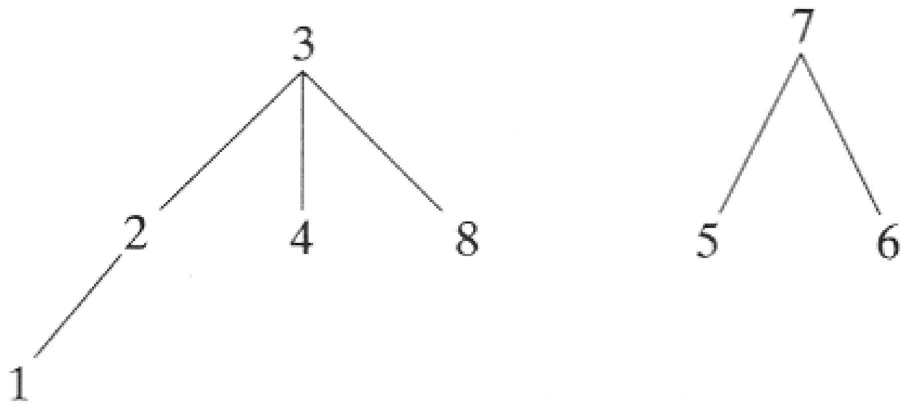
Union Find

- Stores a collection of disjoint sets
- Efficiently implements some operations
  - Union: Merges two sets into one
  - Find: Find which set an element is in
- Stores the sets as a tree structure
  - Node: Label
- Stored as a vector (2D array)

Union Find

PARENT

1	2	3	4	5	6	7	8
2	3	0	3	7	7	0	3



Union Find

Find the parent label of a set.

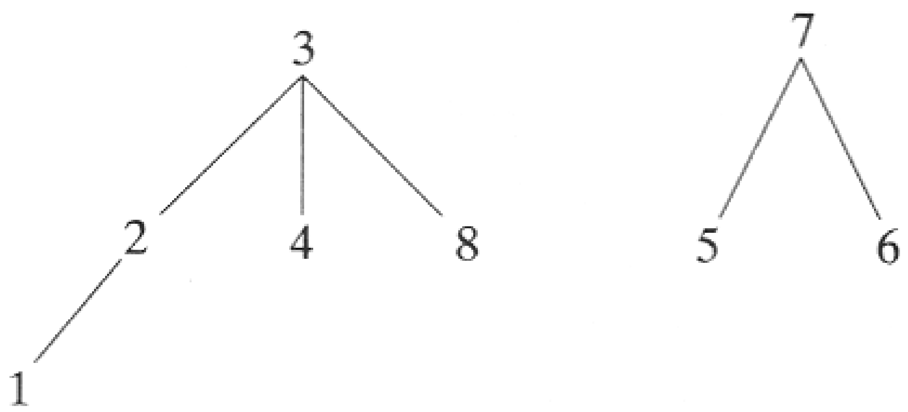
X is a label of the set.
PARENT is the array containing the union-find data structure.

```
procedure find(X, PARENT);
{
j := X;
while PARENT[j] <> 0
   j := PARENT[j];
return(j);
}
```

Union Find

PARENT

1	2	3	4	5	6	7	8
2	3	0	3	7	7	0	3



Union Find

Construct the union of two sets.

X is the label of the first set.
Y is the label of the second set.
PARENT is the array containing the union-find data structure.

```
procedure union(X, Y, PARENT);
{
j := X;
k := Y;
while PARENT[j] <> 0
   j := PARENT[j];
while PARENT[k]] <> 0
   k := PARENT[k];
if j <> k then PARENT[k] := j;
}
```

Classical Algorithm

First Pass

```
currentlabel=1
for (i=0; i<nrows; i++)
  for(j=0; j<ncols; j++)
      if (Image[i,j]==0) break;
      if ((top==0) && (left==1) label (current) = label(left);
      if ((top==1) && (left==0) label (current) = label(top);
      if ((top==0) && (left==0) {
            label (current) = currentlabel;
            currentlabel++;
      }
      if ((top==1) && (left==1) {
            label (current) = label(left);
            equivalent(label(left), label(top));
           PARENT[label(top)] = label(left);
      }
```

Classical Algorithm

Second Pass

```
for (i=0; i<nrows; i++)
    for(j=0; j<ncols; j++)
        if (Image[i,j]==0) break;
        currentlabel=label(i,j);
        label(i,j)=PARENT[currentlabel];
```

Classical Algorithm

1	1	0	2	2	2	0	3
1	1	0	2	0	2	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	7	7	3

(*a*) After Pass 1

1	1	0	1	1	1	0	3
1	1	0	1	0	1	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	3	3	3

(*c*) After Pass 2

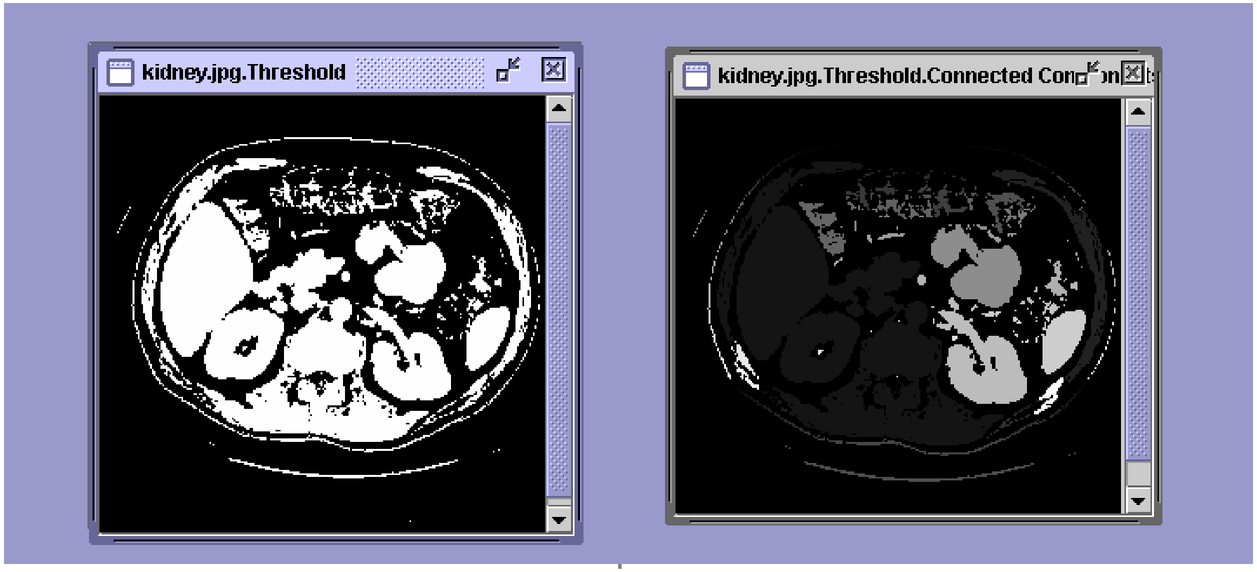
PARENT

1	2	3	4	5	6	7
0	1	0	0	0	0	3

Connected Components Labeling

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 of either the background (0) or a component.



Methods for CC Analysis

1. Recursive Tracking (almost never used)
2. Parallel Growing (needs parallel hardware)
3. Row-by-Row (most common)
   - Classical Algorithm (see text)
   - Efficient Run-Length Algorithm (developed for speed in real industrial applications)

Equivalent Labels

Original Binary Image

0	0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1
0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	**1**	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1

Equivalent Labels

The Labeling Process

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

$1 \equiv 2$
$1 \equiv 3$

Run-Length Data Structure

Binary Image

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

Row Index

	Rstart	Rend
0	1	2
1	3	4
2	5	6
3	0	0
4	7	7

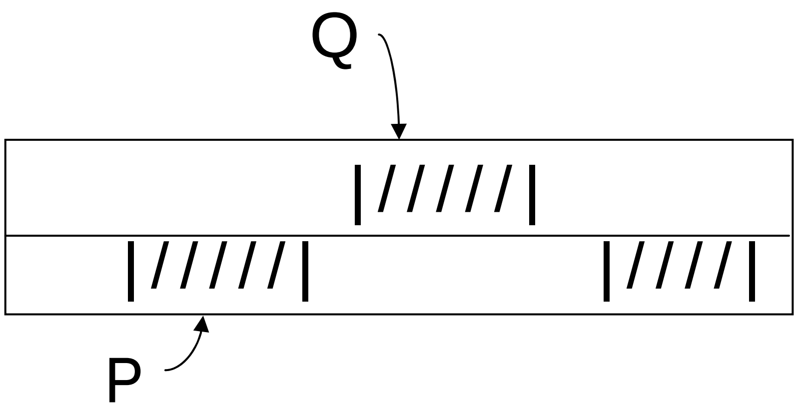
Runs

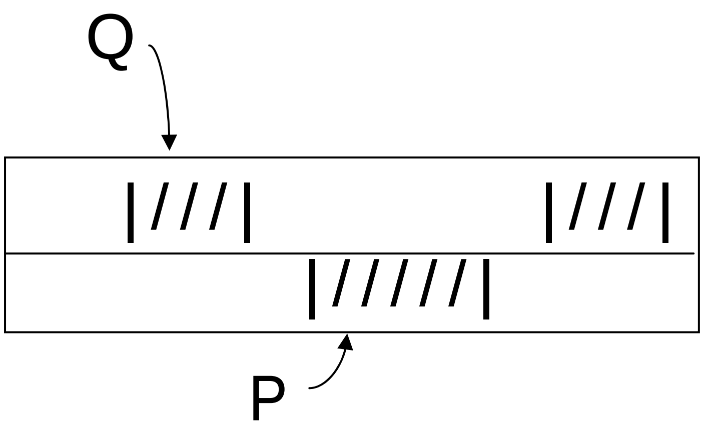
	row	scol	ecol	label
0	U N U S E D			0
1	0	0	1	0
2	0	3	4	0
3	1	0	1	0
4	1	4	4	0
5	2	0	2	0
6	2	4	4	0
7	4	1	4	0

Run-Length Algorithm

```
Procedure run_length_classical
  {
   initialize Run-Length and Union-Find data structures
   count <- 0

/*  Pass 1 (by rows) */

   for each current row and its previous row
     {
      move pointer P along the runs of current row
      move pointer Q along the runs of previous row
```

Case 1: No Overlap



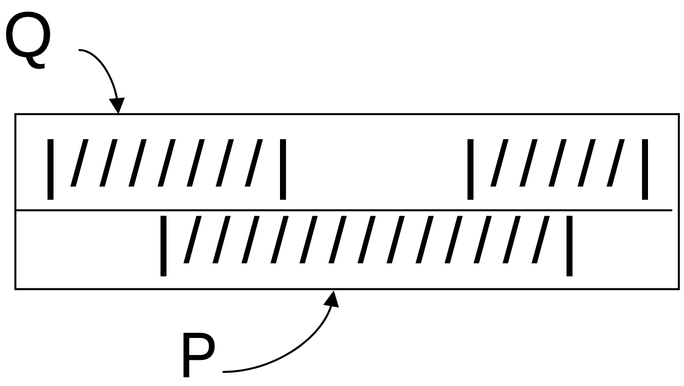



```
/* new label */
count <- count + 1
label(P) <- count
P <- P + 1
```

```
/* check Q's next run */
Q <- Q + 1
```

Case 2: Overlap

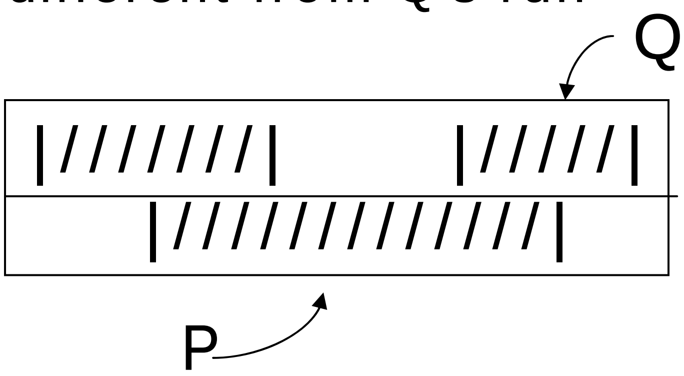
Subcase 1:
P's run has no label yet



label(P) <- label(Q)
move pointer(s)

}

Subcase 2:
P's run has a label that is different from Q's run



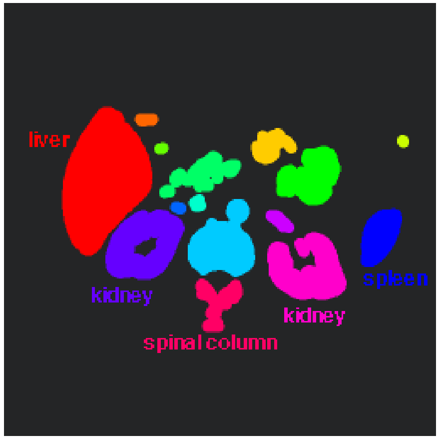
union(label(P),label(Q))
move pointer(s)

Pass 2 (by runs)

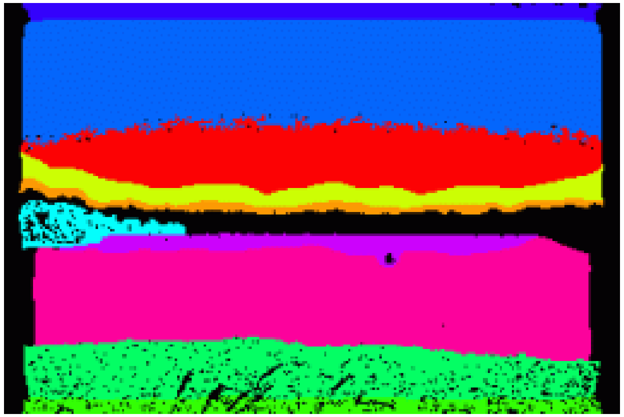
```
/* Relabel each run with the name of the
   equivalence class of its label */
For each run M
  {
   label(M) <- find(label(M))
   }

}
```

where union and find refer to the operations of the Union-Find data structure, which keeps track of sets of equivalent labels.

Labeling shown as Pseudo-Color



connected components of 1's from thresholded image



connected components of cluster labels

Morphology

Morphology

- Study of shape (form and structure)
- Mathematical morphology was designed for a set of points
- Treat the object as binary
- A shape = a set of points
- Use shape-based or iconic approach

Mathematical Morphology

Binary mathematical morphology consists of two basic operations

dilation and erosion

and several composite relations

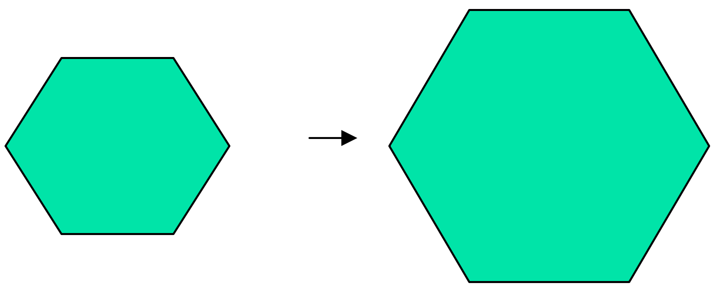
closing and opening
conditional dilation
. . .

Dilation

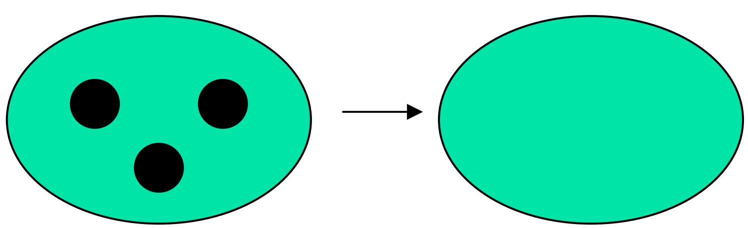
Dilation expands the connected sets of 1s of a binary image.

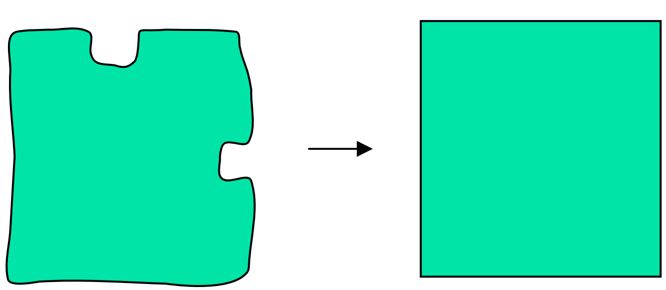
It can be used for

1. growing features



2. filling holes and gaps



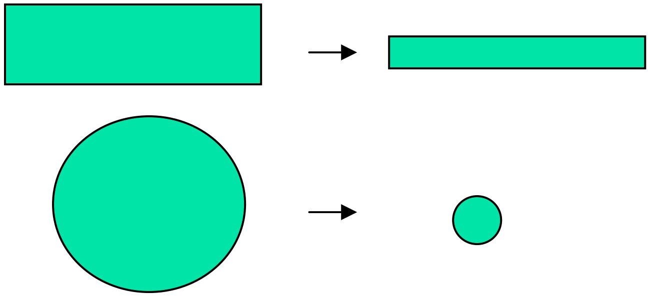


Erosion

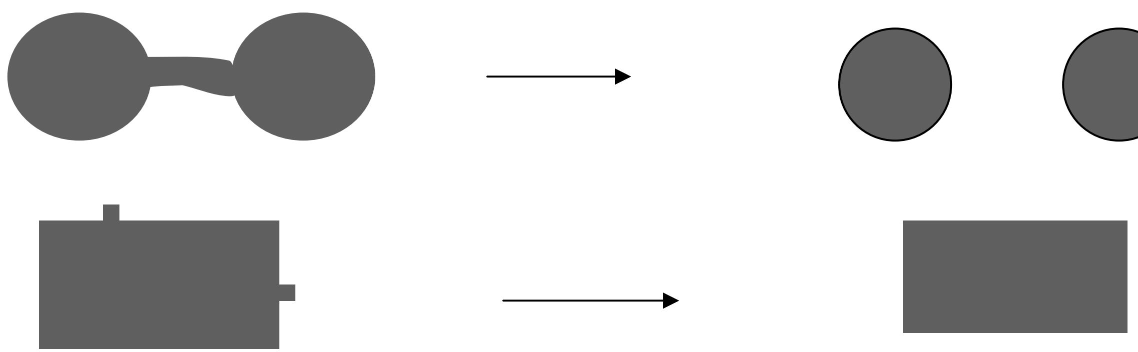
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for

1. shrinking features



2. Removing bridges, branches and small protrusions



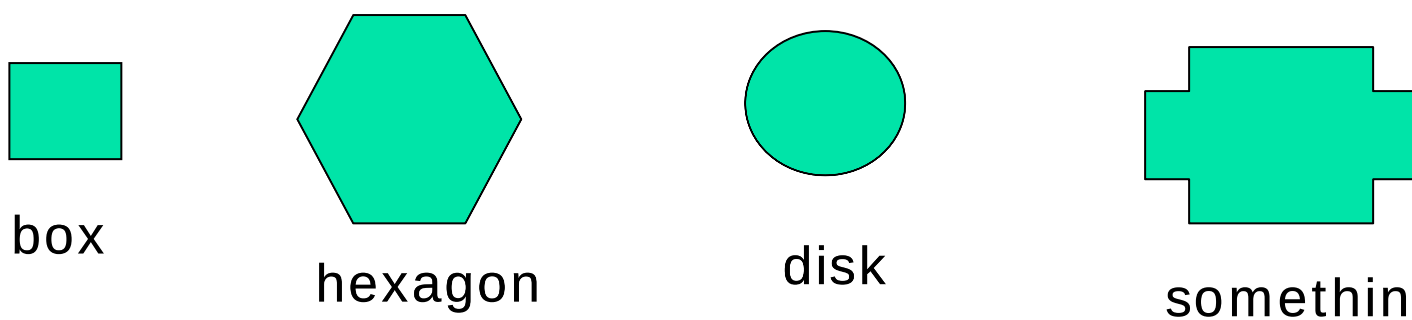
Structuring Elements

- Morphology operations involve
  - Binary image
  - Structuring element
- Structuring Element
  - Binary image
  - Can be any arbitrary binary image in general
  - Usually much smaller

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.

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{B} \qquad \underbrace{\begin{bmatrix} 1 & \\ 1 & 1 \end{bmatrix}}_{S} \qquad \xrightarrow{\text{dilate}} \qquad \underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{B \oplus S}$$

origin

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

$$
\begin{array}{ccccc}
\boxed{\begin{matrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}} & &
\boxed{\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}} &
\xrightarrow{\text{erode}} &
\boxed{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}} \\
B & & S & & B \ominus S
\end{array}
$$

Common Structuring Elements

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(*a*) BOX(3,5)

	1	1	1	
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
	1	1	1	

(*b*) DISK(5)

	1	1	1	
1				1
1				1
1				1
	1	1	1	

(*c*) RING(5)

1	1		
1	1		
1	1	1	1
1	1	1	1

(*d*)

1	1	1	1	1	1
1		1	1		1
1		1	1		1
1		1	1		1

(*e*)

1
1
1
1

(*f*)

Operations

- Intersection

$$A \bigcap B = \{p \mid p \in A \text{ and } p \in B\}$$

- Union

$$A \bigcup B = \{p \mid p \in A \text{ or } p \in B\}$$

- Complement

$$\overline{A} = \{p \mid p \in \Omega \text{ and } \mathrm{p} \notin \mathrm{A}\}$$

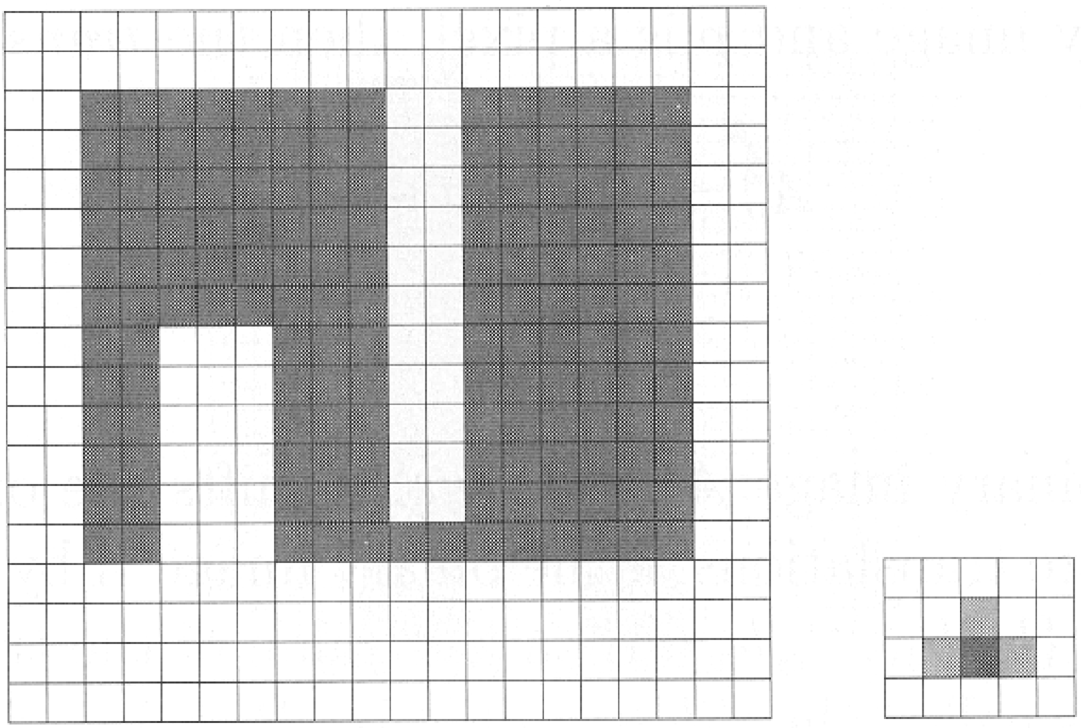
Ω is a universal binary image (all 1s)

Translation

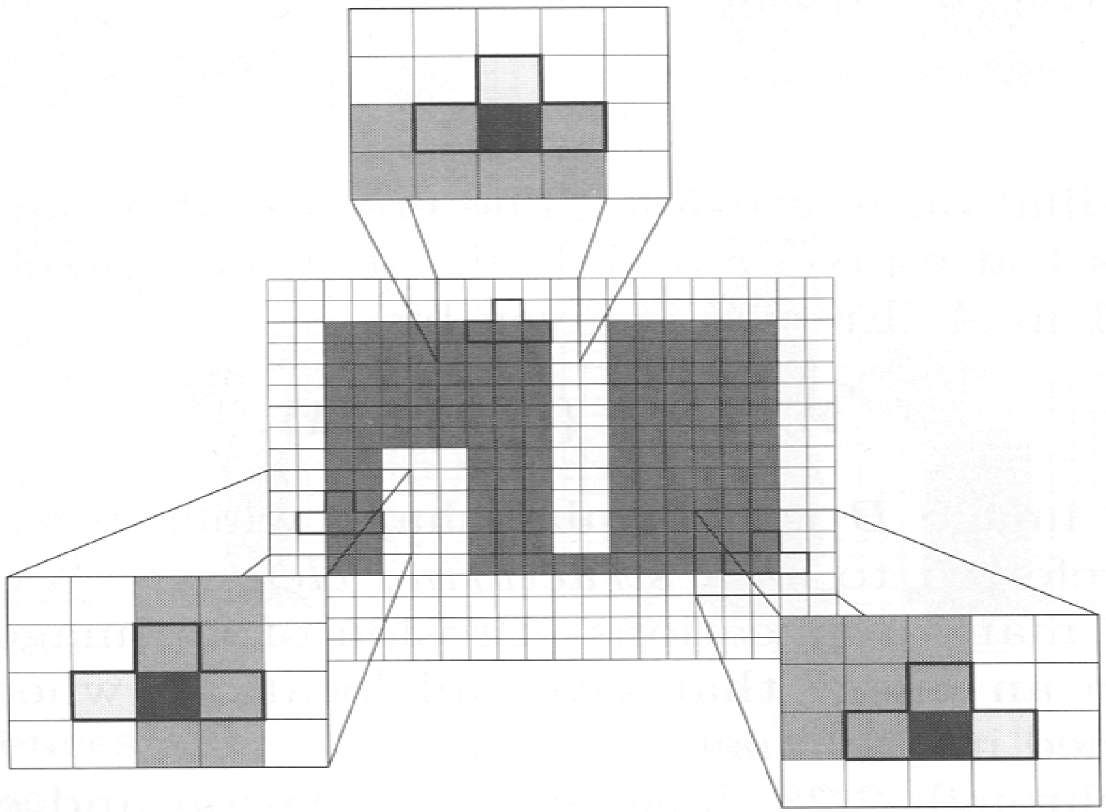
- A : binary image
- p : pixel
- Translation of A by p is an image
- Shifts the origin of A to p

$$A_p = \{a + p \mid a \in A\}$$

Example



Translation



Dilation

$$A \oplus B = \bigcup_{b_i \in B} A_{b_i}$$

$$B = \{b_1, b_2, \ldots . b_n\}$$

Dilation of A by B

A_{b_1} is the translation of A by a pixel of B

Dilation

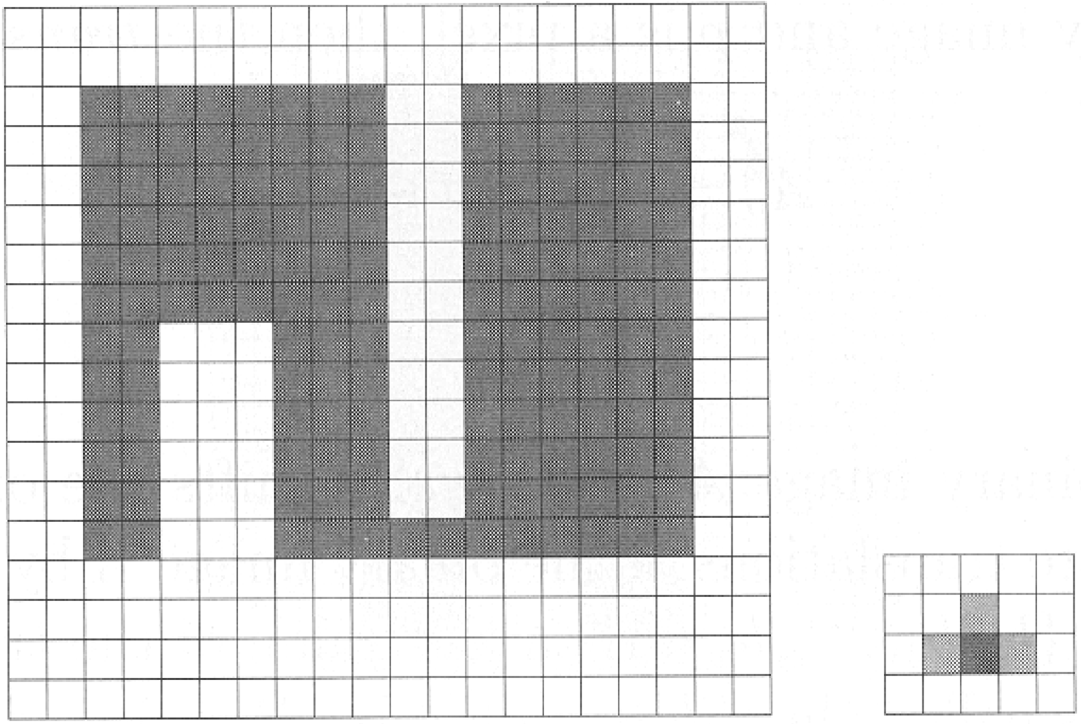
- Commutative

$$A \oplus B = B \oplus A$$

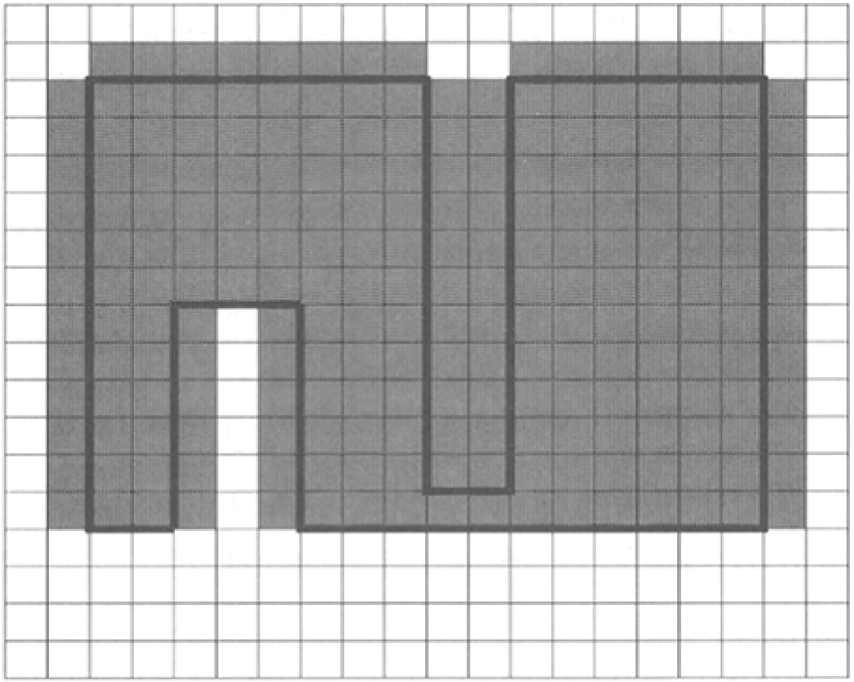
- Associative

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

Example



Dilation



$$A \oplus B = \bigcup_{b_i \in B} A_{b_i}$$

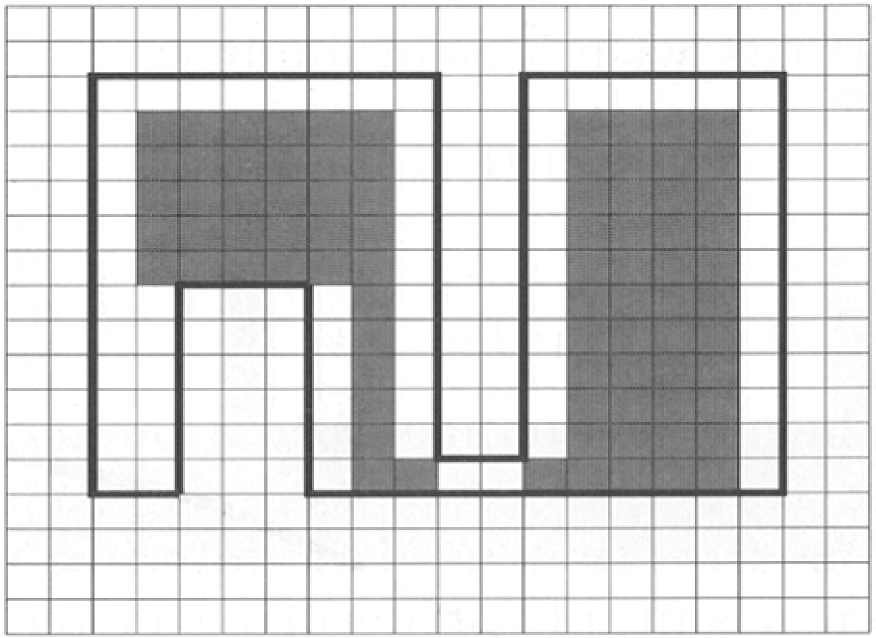
Erosion

- Opposite of Dilation

$$A \Theta B = \{p \mid B_p \subseteq A\}$$

- 1 at a pixel iff every 1 pixel in the translation of B to p is also a 1
- Structuring element is a probe
- Locate where the structuring element is located in the image

Erosion



$$A \ominus B = \{p | B_p \subseteq A\}$$

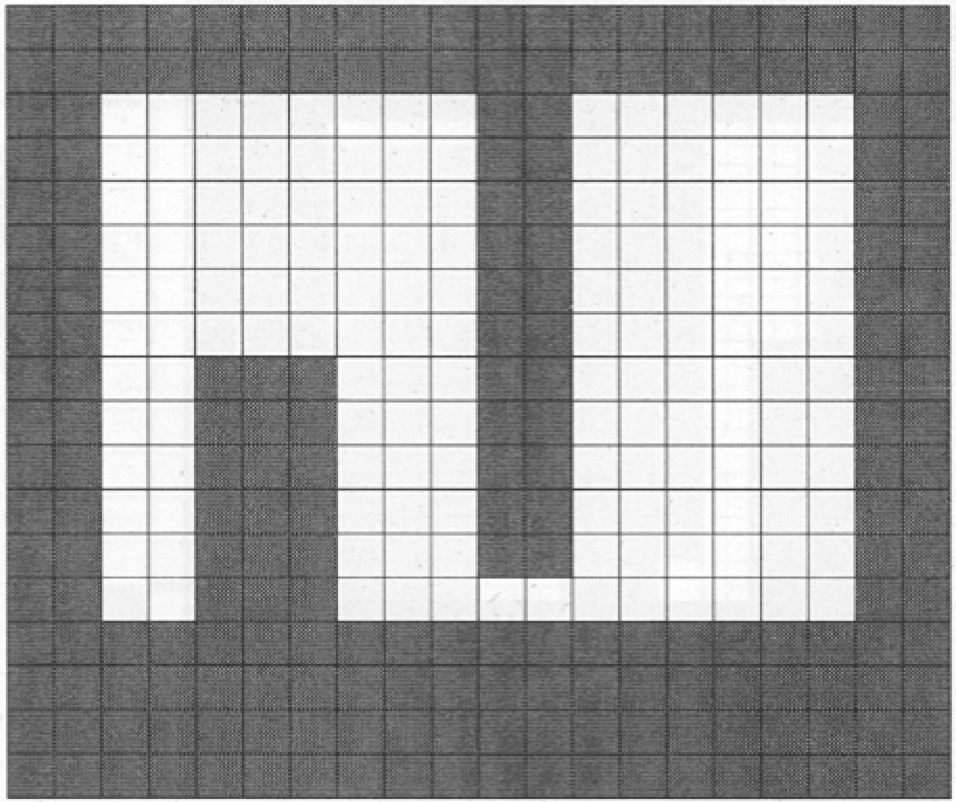
Erosion and Dilation

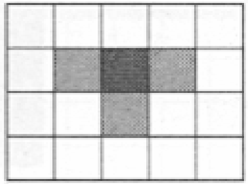
$$B' = \{-p \mid p \in B\}$$

$$\overline{A \oplus B} = \overline{A} \Theta B'$$

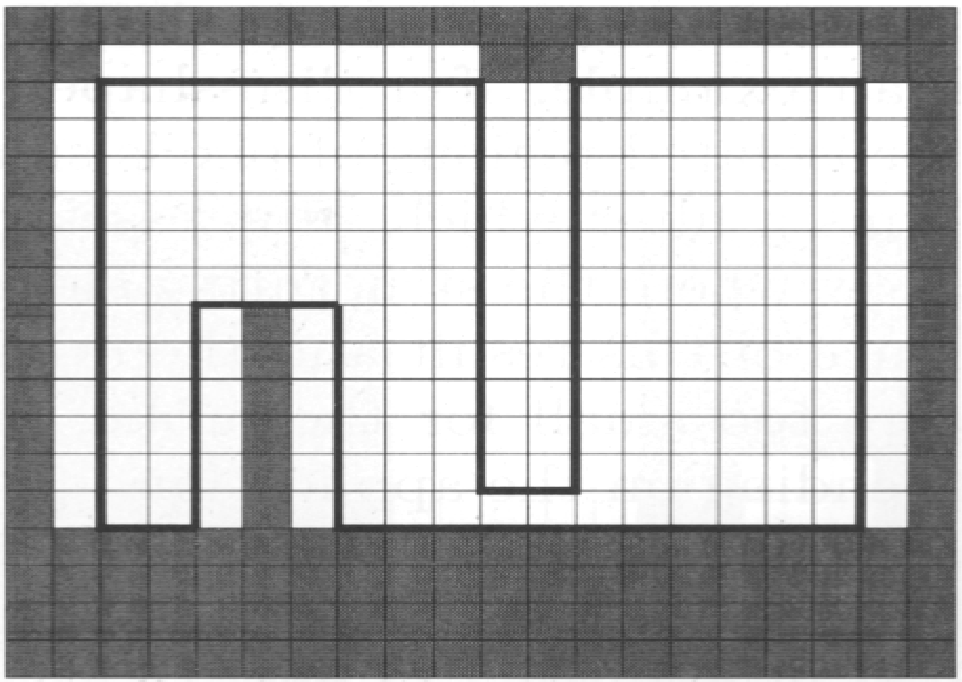
$$\overline{A \Theta B} = \overline{A} \oplus B'$$

Complement of A and probe



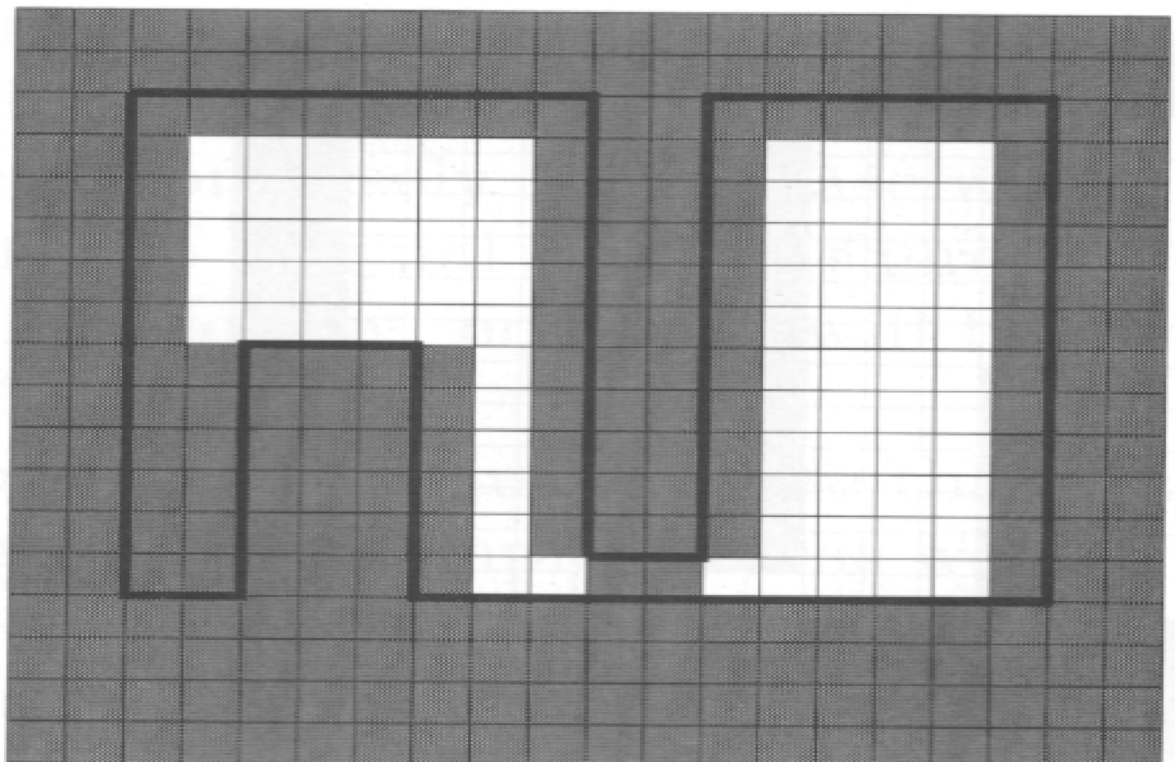


$$\overline{A} \Theta B'$$



$\overline{A} \ominus B'$

$$\overline{A} \oplus B'$$

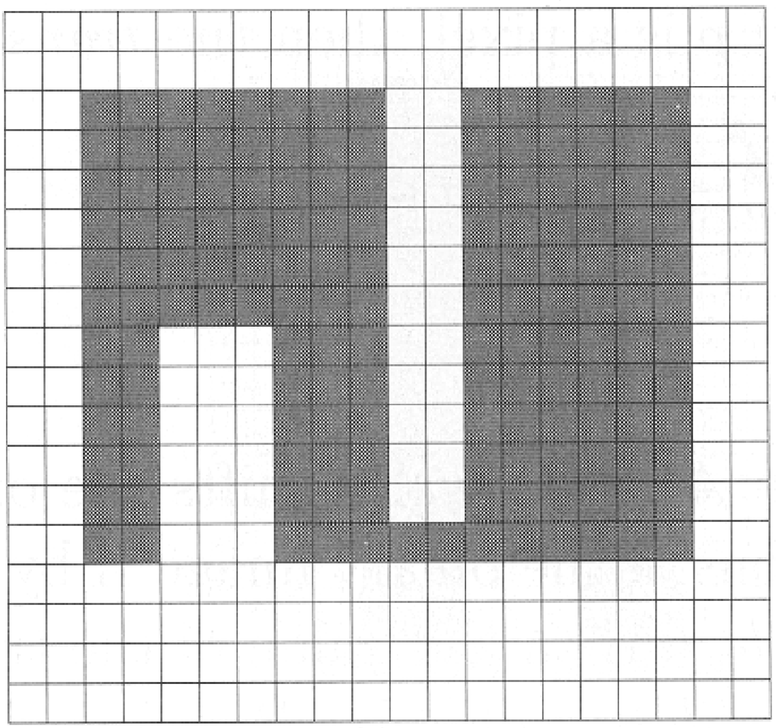


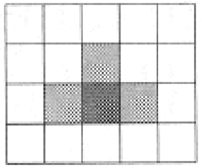
$\overline{A} \oplus B'$

Opening

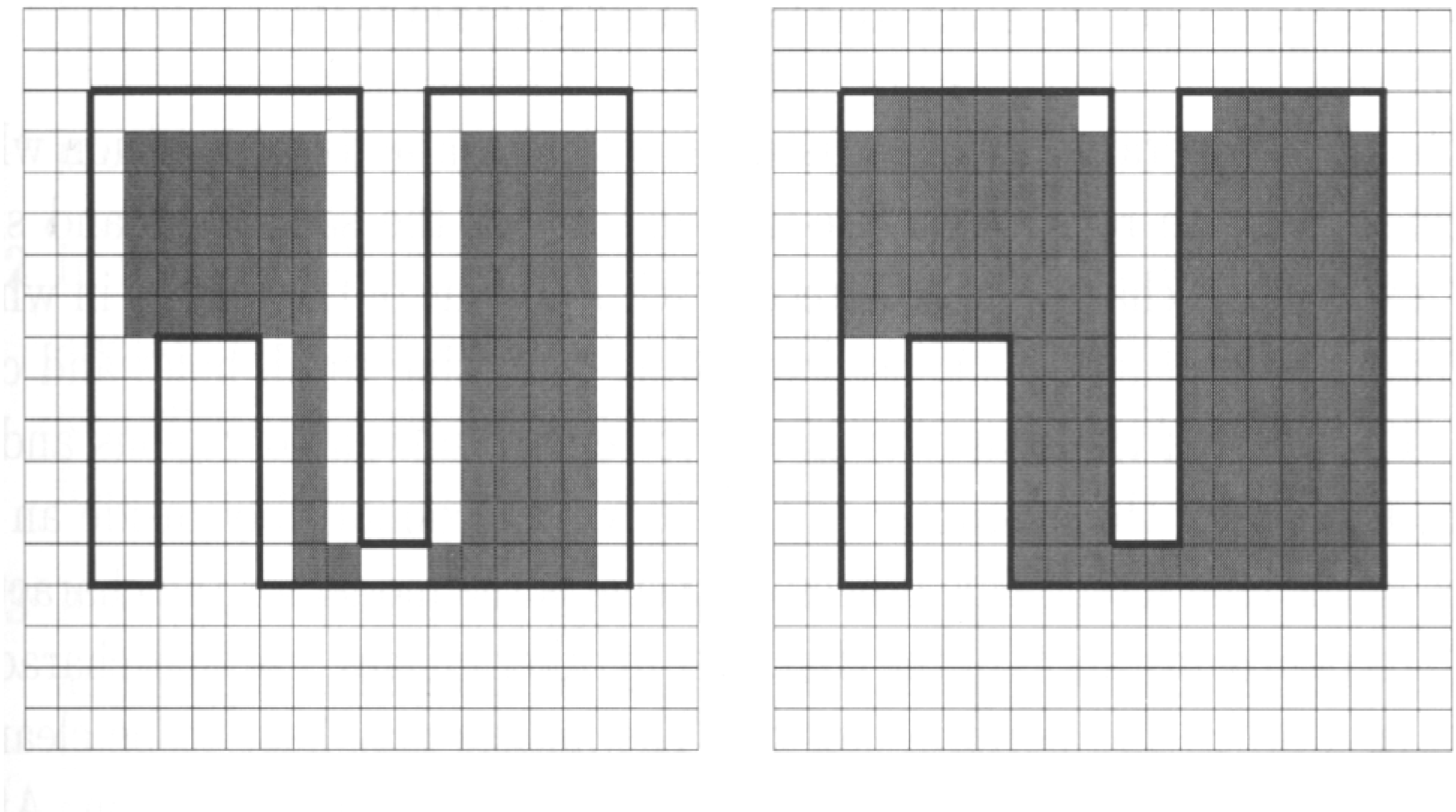
- Erosion followed by dilation
- Remove all pixels in a region too small to contain the structuring element
- Disc shaped probe
  - Eliminate all convex regions and isolated points

Example





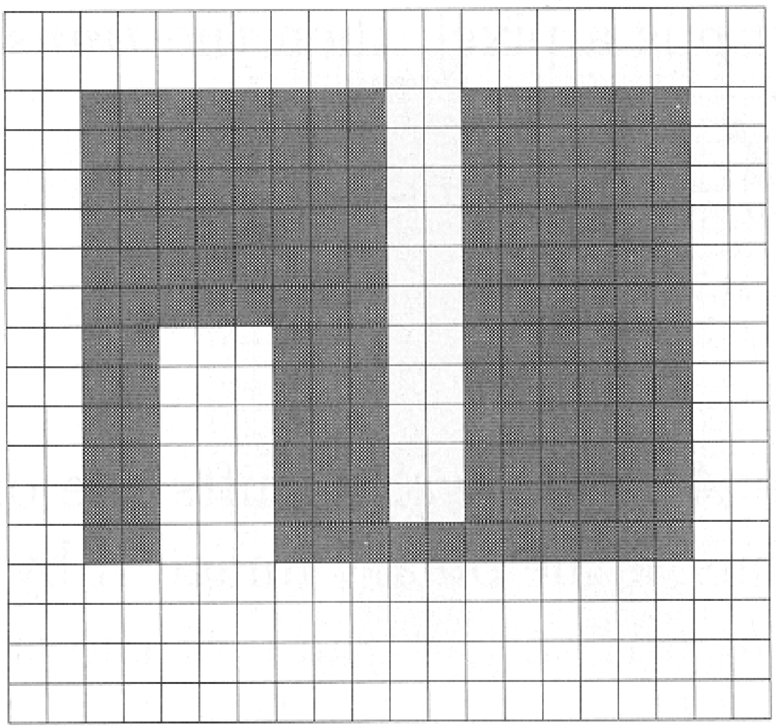
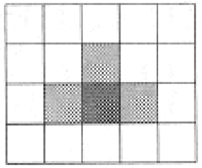
Example



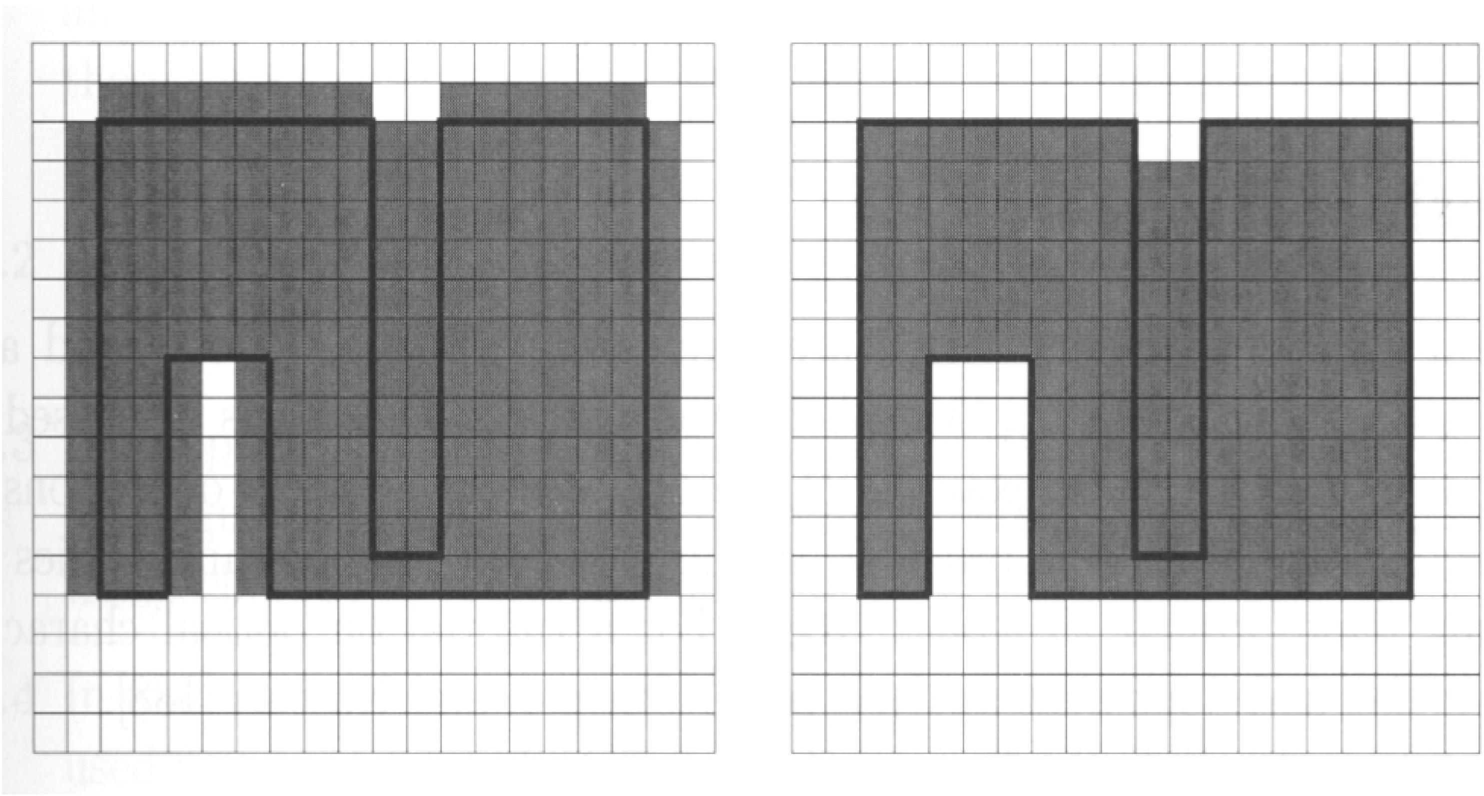
Closing

- Opposite of opening
- Dilation followed by erosion
- Fill holes smaller than the structuring element

Example

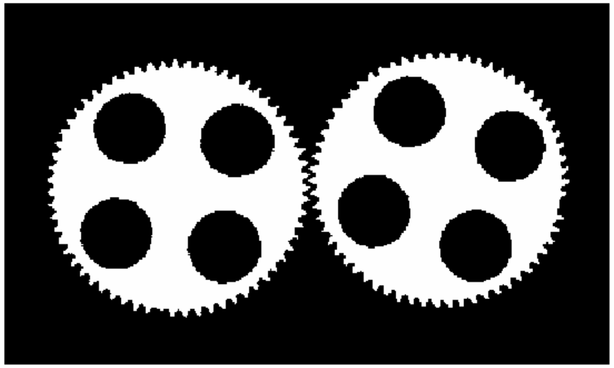
Example



Example

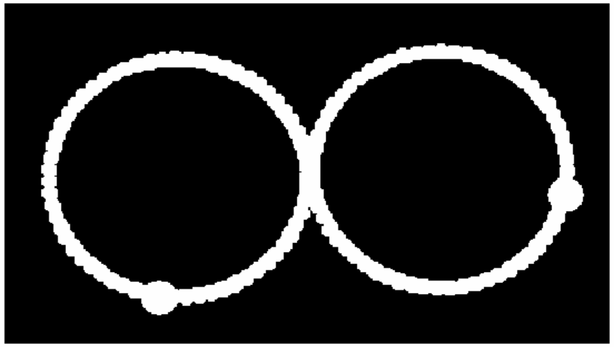


Gear Tooth Inspection



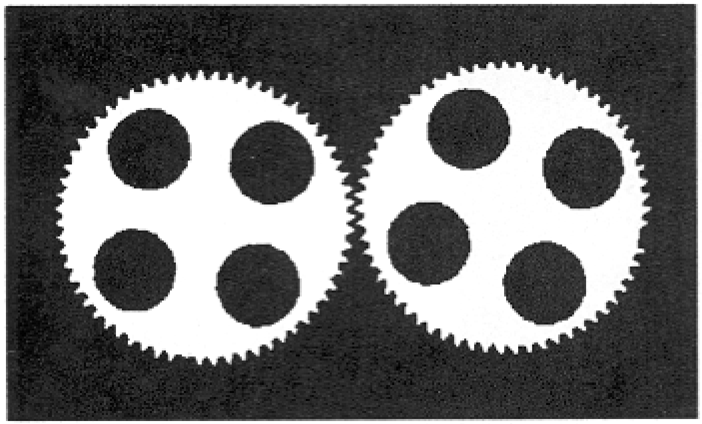
original
binary
image

How did
they do it?

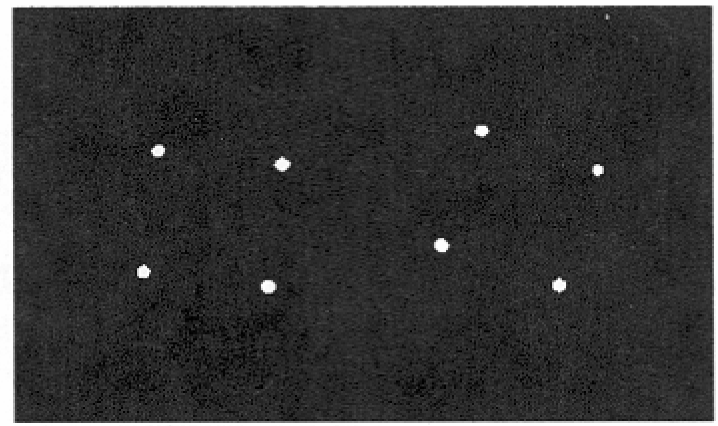


detected
defects

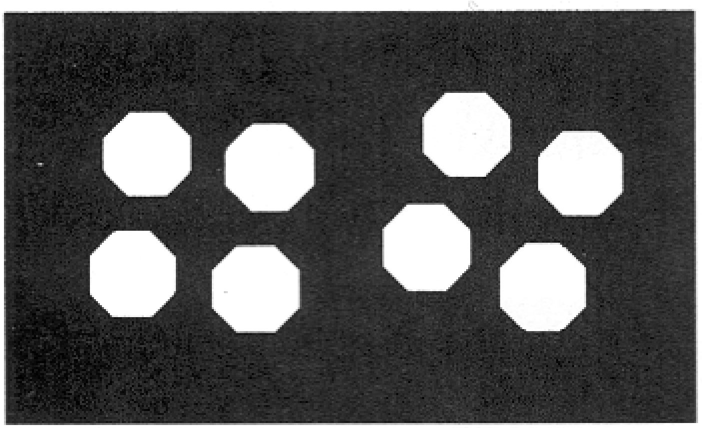
Example



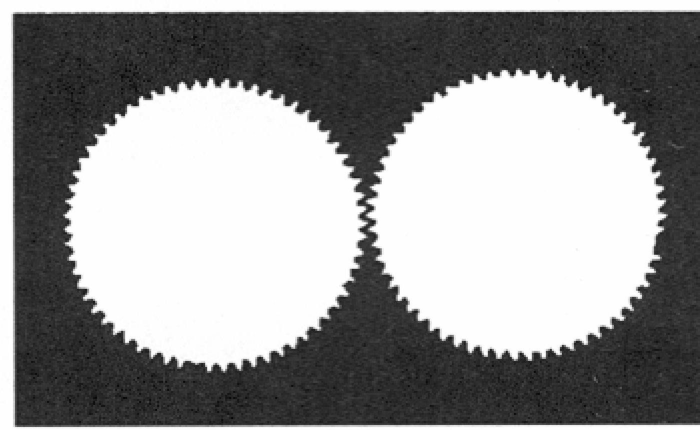
(a) Original image **B**



(b) $\mathbf{B1} = \mathbf{B} \ominus \mathbf{hole\_ring}$

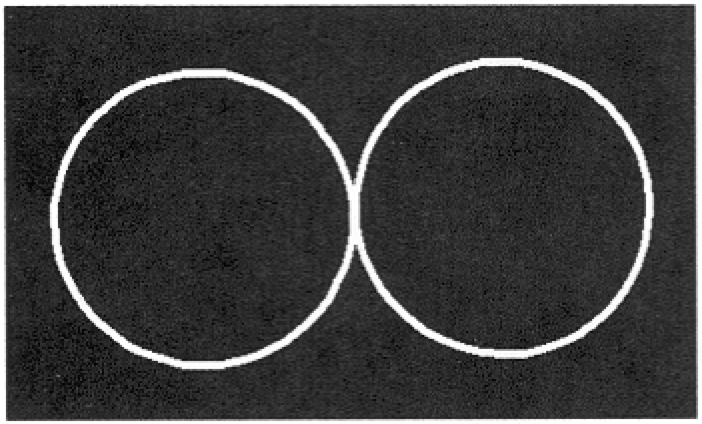


(c) $\mathbf{B2} = \mathbf{B1} \oplus \mathbf{hole\_mask}$

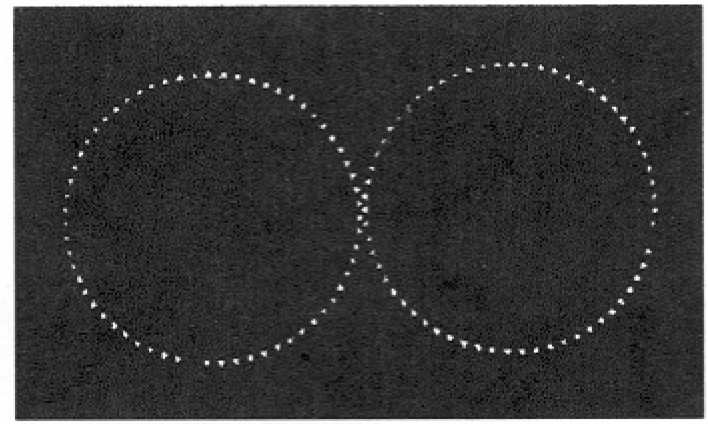


(d) **B3** = **B** OR **B2**

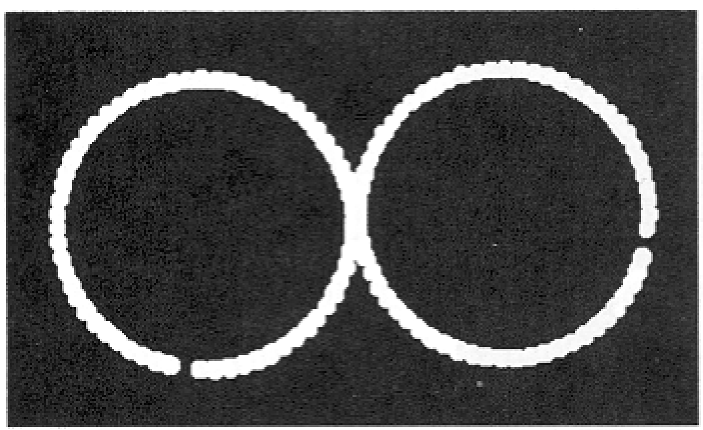
Example



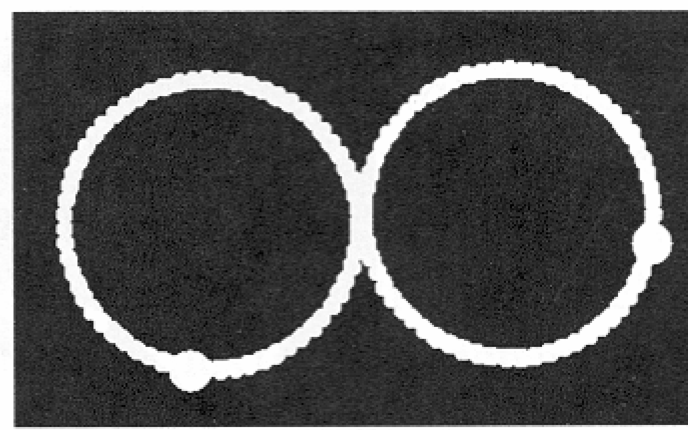
(e) **B7** (see text)



(f) **B8** = **B** AND **B7**



(g) **B9** = **B8** ⊕ **tip_spacing**



(h) RESULT = ((B7–B9) ⊕ **defect_cue**) OR B9

Shape Features

Shape Features

- Organization of points in space
- Not related to any intensity or color features
- Common Features
  - Area
  - Centroid
  - Perimeter
  - Elongatedness
  - Aspect Ratio
  - Bounding box

Area

- Number of pixels in a component
- Gives an indication of size
- Easy to compute

$$A = \sum_{(r,c)\in R} 1 \quad \textit{where } R \textit{ is a given region}$$

Area and centroid

- We denote the set of pixels in a region by R.
- **assuming square pixels:**

  area:

$$A = \sum_{(r,c)\in R} 1$$

  centroid:

$$\begin{aligned} \bar{r} &= \frac{1}{A} \sum_{(r,c)\in R} r \\ \bar{c} &= \frac{1}{A} \sum_{(r,c)\in R} c \end{aligned}$$

- $(\bar{r}, \bar{c})$ is generally not a pair of integers.
- A precision of tenths of a pixel is often justifiable for the centroid.

Perimeter

- Boundary points of the object
- Length of the perimeter is the number of pixels on the boundary of the object
- Use 4/8 neighborhood to determine

$$P_4 = \{(r,c) \in R \mid N_8(r,c) - R \neq \phi\}$$

$$P_8 = \{(r,c) \in R \mid N_4(r,c) - R \neq \phi\}$$

Perimeter

- Length of the perimeter is an important measure of the shape
- First order of approximation
  - Number of pixels on the boundary
- More accurate approximation
  - Horizontal and vertical moves : 1 unit
  - Diagonal move: 1.4 unit

Perimeter

- Order the boundary pixels in a sequence

$$P = <(r_0, c_0), \cdots, (r_k, c_k)>$$

- Pixels in the sequence are adjacent

$$|P| = |\{k \mid (r_{i+1}, c_{i+1}) \in N_4(r_i, c_i)\}| +$$

$$\sqrt{2}|\{k \mid (r_{i+1}, c_{i+1}) \in N_8(r_i, c_i) - N_4(r_i, c_i)\}|$$

- Indices are computed modulo k (wrap around)

Circularity (1)

- Degree of similarity with a circle

$$Circularity = \frac{|P|^2}{A}$$

- Based on continuous planar shapes
- smallest value for a circle?
- Make adjustments for digital shapes

Circularity (2)

$$Circularity = \frac{\mu_R}{\sigma_R}$$

$$\mu_R \ :Mean\ radial\ distance$$

$$\sigma_r \ :Standard\ deviation\ of\ radial\ distance$$

- Radial distance computed as the distance between the centroid and a boundary point

Circularity (2)

$$\mu_R = \frac{1}{k}\sum_{i=0}^{k-1}\left\|(r,c_k) - (\bar{r},\bar{c})\right\|$$

$$\sigma_r = \left(\frac{1}{k}\sum_{i=0}^{k-1}\left[\left\|(r,c_k) - (\bar{r},\bar{c})\right\| - \mu_R\right]^2\right)^{1/2}$$

$(\bar{r},\bar{c})$ *is the centroid of the region*

Region Properties – Fig 3.18

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

region num.	region area	row of center	col of center	perim. length	circu- larity$_1$	circu- larity$_2$	radius mean	radius var.
1	44	6	11.5	21.2	10.2	15.4	3.33	.05
2	48	9	1.5	28	16.3	2.5	3.80	2.28
3	9	13	7	8	7.1	5.8	1.2	0.04

Compactedness

- Approximated by the bending energy

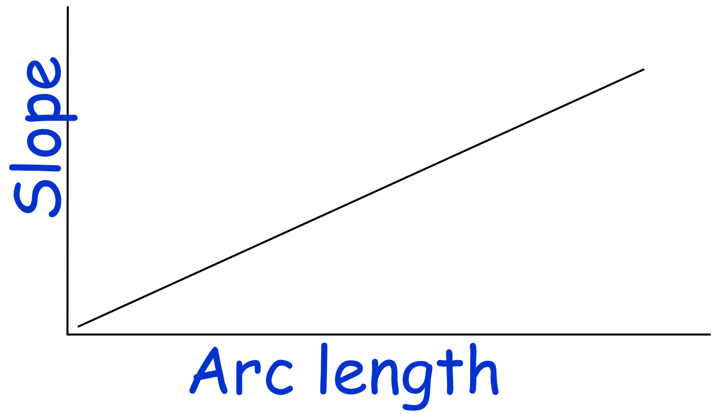
$$E = \frac{1}{P}\int_0^P |\kappa(s)|^2 ds$$

κ is the curvature

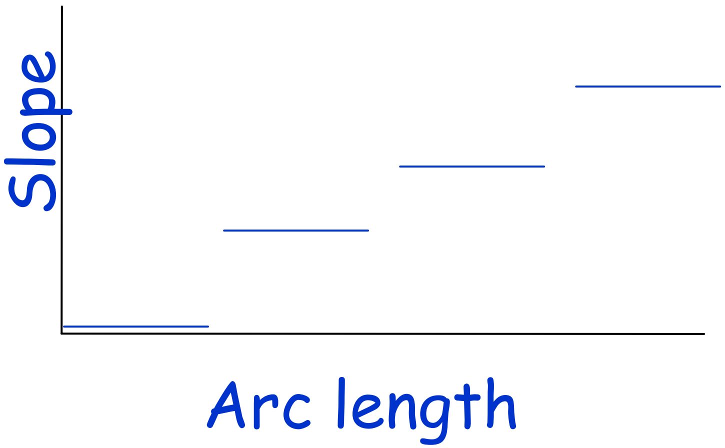
- Minimized by a circle

Slope Density Function

- Use Ψ–s representation for the curve
  - Ψ: Slope of the curve at a given point
  - s: arc length
- SDF shows the distribution of the slopes along the boundary



Circle



Square

Projections/Signatures

- Non-information preserving
- Maxima and minima are indications of landmarks
- Horizontal signature

$$p(x) = \int_y f(x, y)$$

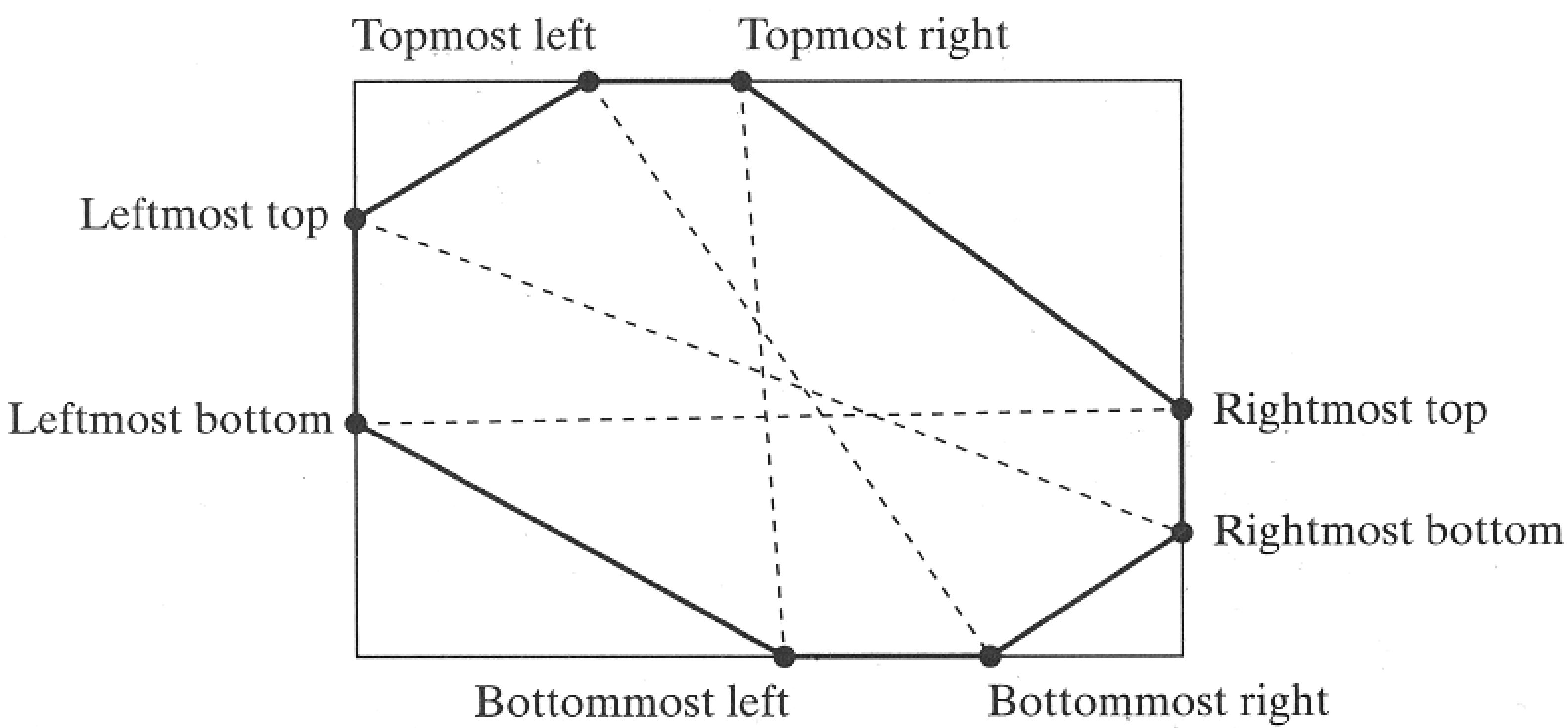
- Vertical signature

$$p(y) = \int_x f(x, y)$$

Bounding Box

- Gives a rough idea of the location of the object
- Smallest rectangle to enclose the shape
- Extremal points
  - Leftmost, rightmost, topmost and bottommost
  - Lie on the bounding box boundary
  - May have upto 8 extremal points
  - Occur in opposite pairs

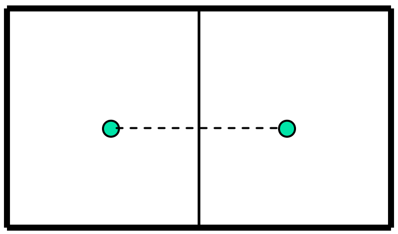
Bounding Box



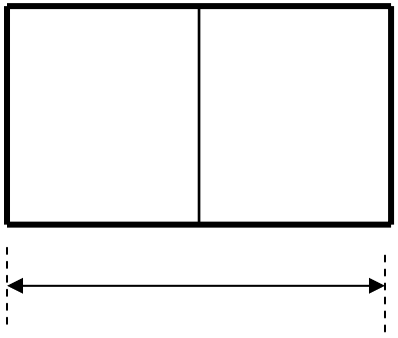
Bounding Box

- Each point of opposite extremal point pairs define an axis
- Properties of axes are useful
  - Lengths
  - Orientations
- Need to make corrections to the lengths
  - Error due to sampling

Bounding Box



- Distance between the two pixels =1



- Distance between the two ends of the pixels=2

Bounding Box

- Extremal axis length

$$D = \sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2} + Q(\theta)$$

$$(r_1, c_1)\ and\ (r_2, c_2)\ are\ two\ extremal\ points$$

$$Q(\theta) = \begin{cases} \dfrac{1}{|\cos\theta|} & |\theta| < 45^\circ \\ \dfrac{1}{|\sin\theta|} & |\theta| > 45^\circ \end{cases}$$

$$\theta : \begin{matrix} the\ angle \\ the\ angle \end{matrix}$$

Moments

- Used to indicate the shape of the objects
- Describe the distribution of points in shape
- Many moments can be defined
- Only low order moments are useful

Second-order Moments

- Row moments $$\mu_{rr} = \frac{1}{A} \sum_{(r,c) \in R} (r - \bar{r})^2$$
- Column moments $$\mu_{cc} = \frac{1}{A} \sum_{(r,c) \in R} (c - \bar{c})^2$$
- Mixed moments $$\mu_{rc} = \frac{1}{A} \sum_{(r,c) \in R} (r - \bar{r})(c - \bar{c})$$

Moments ☹

$$M_{pq} = \int_x \int_y x^p y^q r(x, y)\, dx\, dy$$

where

$$r(x, y) = \begin{cases} 1 \; if \; (x, y) \text{ is in the object} \\ 0 \text{ otherwise} \end{cases}$$

Moments

- Defines moments of order *p+q*
- Changes with location of the object
- Discrete case

$$M_{pq} = \sum_x \sum_y x^p y^q r(x, y)$$

Moments

$$M_{00} = \text{Areaof theobject}$$

$$M_{10} = \text{Sum of all x - coordinates}$$

$$M_{01} = \text{Sum of all x - coordinates}$$

$$\mathrm{x_c} = \frac{M_{10}}{M_{00}} \quad \mathrm{y_c} = \frac{M_{01}}{M_{00}}$$

Central Moments

- Moments change with the location of the object
- Not useful for comparing shapes

$$M_{pq} = \sum_x \sum_y (x - x_c)^p (y - y_c)^q r(x, y)$$

$$M_{01} = M_{10} = 0$$

$$x_c = y_c = 0$$

Central Moments

- Centroid forms the origin
- Orientation of on object is given by

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2M_{11}}{M_{20} - M_{02}}\right) + n\left(\frac{\pi}{2}\right)$$

- Eccentricity (elongatedness)

$$e = \frac{(M_{20} - M_{02})^{1/2} + 4M_{11}}{M_{00}}$$

Moments

- Row moments

$$M_{pp} = \frac{1}{A} \sum_{x} (x - x_c)^p \, r(x, y)$$

- Column moments

$$M_{qq} = \frac{1}{A} \sum_{x} (y - y_c)^q \, r(x, y)$$

Ellipses

- Circular objects get mapped to ellipses in imaging process
- Defined by their major and minor axes
- May be used to approximate elongated objects

$$dx^2 + 2exy + fy^2 \leq 1$$

Assumes that the center of the circle is at the origin

Ellipses

- May be defined as a function of the moments also
- Lengths of the major and minor axes can be derived from the moments

$$\begin{pmatrix} d & e \\ e & f \end{pmatrix} = \frac{1}{4(\mu_{xx}\mu_{yy} - \mu_{xy}^2)} \begin{pmatrix} \mu_{yy} & \mu_{xy} \\ -\mu_{xy} & \mu_{xx} \end{pmatrix}$$

Ellipse Axes

- Need to consider four cases

$$\mu_{rc} = 0 \text{ and } \mu_{rr} > \mu_{cc}$$

$$Major\ axis: Orientation = -90^\circ \quad Length = 4\mu_{rr}^{1/2}$$

$$Minor\ axis: Orientation = 0^\circ \quad Length = 4\mu_{cc}^{1/2}$$

- Angle measured counterclockwise from the column axis

Ellipse Axes

Case II: $$\mu_{rc} = 0 \text{ and } \mu_{rr} \le \mu_{cc}$$

$$Major\ axis : Orientation = 0^\circ \quad Length = 4\mu_{cc}^{1/2}$$

$$Minor\ axis : Orientation = -90^\circ \quad Length = 4\mu_{rr}^{1/2}$$

Ellipse Axes

Case III: $\mu_{rc} \neq 0 \text{ and } \mu_{rr} \leq \mu_{cc}$

Major axis:

$$Orientation = \tan^{-1}\left\{\frac{-2\mu_{rc}}{\mu_{rr} - \mu_{cc} + [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}}\right\}$$

$$Length = \left\{8\left(\mu_{rr} - \mu_{cc} + [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}\right)\right\}^{1/2}$$

Ellipse Axes

Case III: $$\mu_{rc} \neq 0 \text{ and } \mu_{rr} \leq \mu_{cc}$$

$Minor\ axis:$

$$Orientation = 90 + \tan^{-1}\left\{\frac{-2\mu_{rc}}{\mu_{rr} - \mu_{cc} + [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}}\right\}$$

$$Length = \left\{8\left(\mu_{rr} + \mu_{cc} - [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}\right)\right\}^{1/2}$$

Ellipse Axes

Case IV: $$\mu_{rc} \neq 0 \text{ and } \mu_{rr} > \mu_{cc}$$

$Major\ axis:$

$$Orientation = \tan^{-1} \frac{\left[\mu_{rr} + \mu_{cc} + [(\mu_{cc} - \mu_{rr})^2 + 4\mu_{rc}^2]^{1/2}\right]^{1/2}}{-2\mu_{rc}}$$

$$Length = \left\{8\left(\mu_{rr} + \mu_{cc} + [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}\right)\right\}^{1/2}$$

Ellipse Axes

Case IV: $$\mu_{rc} \neq 0 \text{ and } \mu_{rr} \leq \mu_{cc}$$

$$Minor\ axis:$$

$$Orientation = 90 + \tan^{-1} \frac{\left[\mu_{rr} + \mu_{cc} + [(\mu_{cc} - \mu_{rr})^2 + 4\mu_{rc}^2]^{1/2} \right]^{1/2}}{-2\mu_{rc}}$$

$$Length = \left\{ 8\left(\mu_{rr} + \mu_{cc} - [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2} \right) \right\}^{1/2}$$

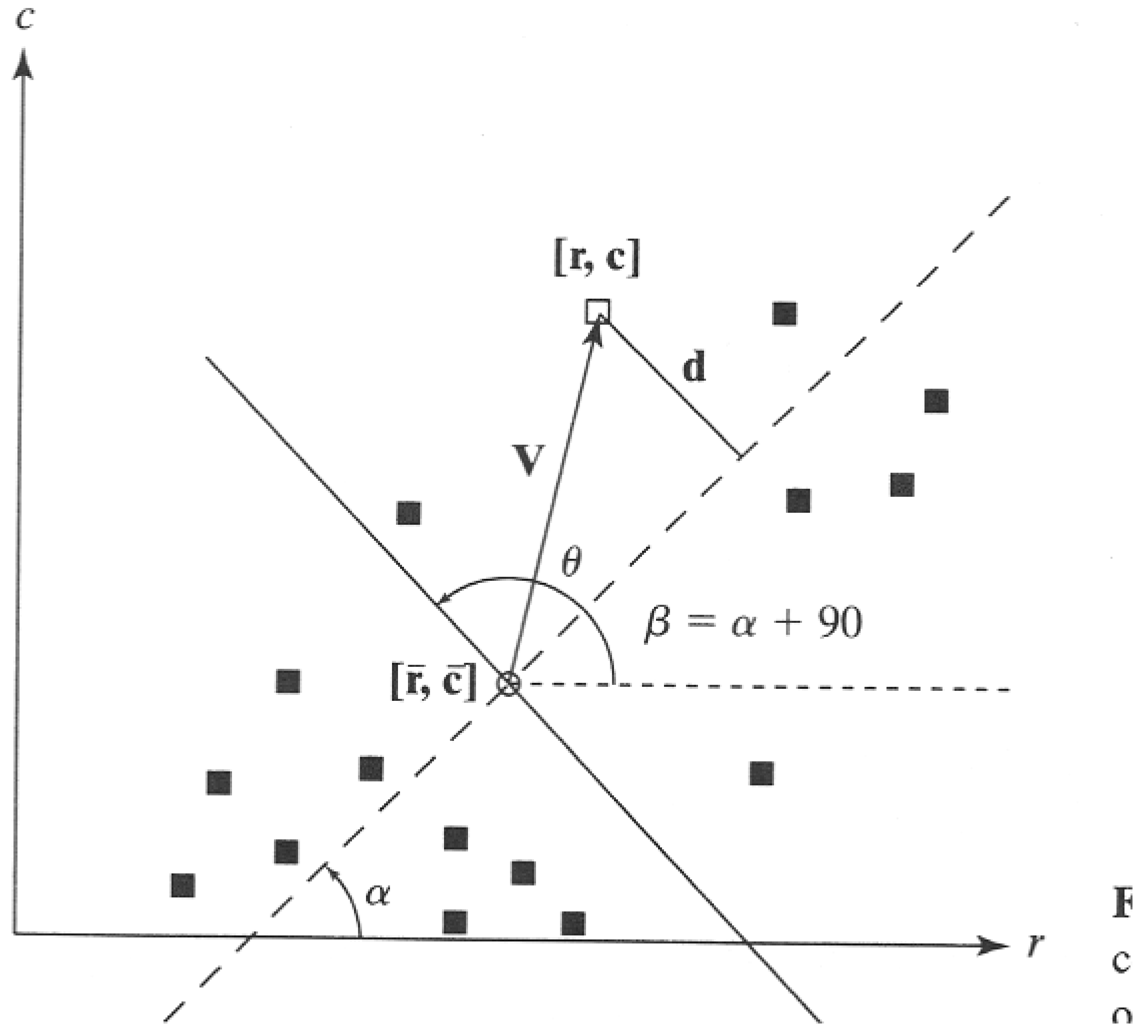
Best Axes ☹

- Some objects have natural axis
  - Pencils, Letters l,I, -
- Axis around which there is least second order moment
- Axis of inertia
  - Line about which the object can be spun with least energy input
- Circle
  - All axes have same length

Best Axes

- Axis must past through the centroid
- Need to compute moment about any arbitrary axis
- Minimize the moments as a function of angle
  - Gives the axis of least moment
  - Other principal axis is perpendicular to it

Best Axis p79



Best Axes

$$\mu_{\bar{r},\bar{c},\alpha} = \frac{1}{A} \sum_{(r,c)\in R} d^2$$

$$= \frac{1}{A} \sum_{(r,c)\in R} ((r-\bar{r})\cos\beta + (c-\bar{c})\sin\beta)^2$$

- Any axes can be defined by the three parameters

Best Axes

- Axis with the least second moment

$$\tan 2\hat{\alpha} = \frac{2\sum (r-\bar{r})(c-\bar{c})}{\sum (r-\bar{r})(r-\bar{r}) - \sum (c-\bar{c})(c-\bar{c})}$$

$$= \frac{\frac{1}{A} 2\sum (r-\bar{r})(c-\bar{c})}{\frac{1}{A}\sum (r-\bar{r})(r-\bar{r}) - \frac{1}{A}\sum (c-\bar{c})(c-\bar{c})}$$

$$= \frac{2\mu_{rc}}{\mu_{rr} - \mu_{cc}}$$

- Symmetric objects (circle/square) results in zero divide

Region Adjacency Graphs p81

- Relationships between the regions are important also
- Region adjacency is one way to show
- Two regions are adjacent if they have two pixels that are neighbors
- Binary images
  - Foreground and background regions
  - All components are adjacent to the background

Region Adjacency Graphs

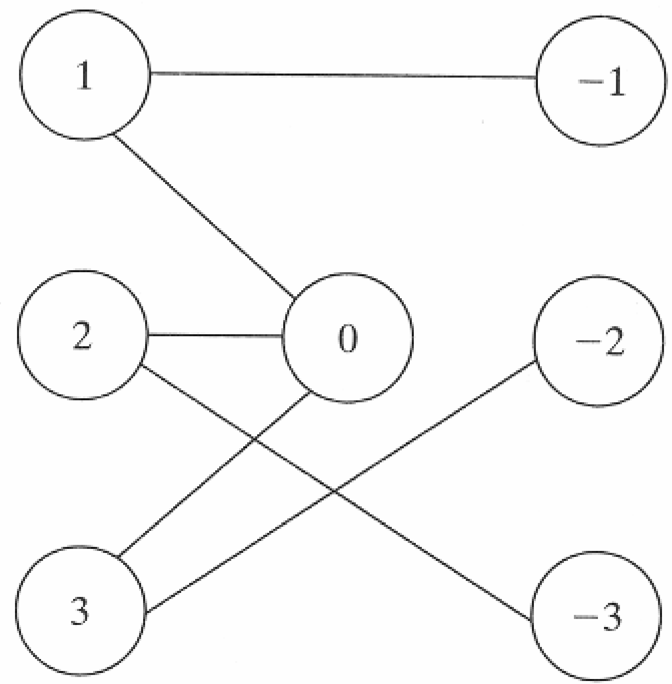
- Why?
  - Objects can have holes
  - Find connected components for the background
  - Adjacency makes sense then
  - Dealing with grayscale/color images

Region Adjacency Graphs

- A Region Adjacency Graph (RAG) is a graph
  - Nodes: Components of the image
  - Edges: If two components are adjacent, then there is an edge between the corresponding nodes

Region Adjacency Graphs

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



Region Adjacency Graphs

- Algorithm
  - Make one pass of the labeled image
  - At each pixel
    - Check the labels of the neighbors
    - If it is different from the current label
      - Add an edge in the graph it is not already there
- Efficiency is very very good
  - May have a large number of labels
  - Repetition

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

Perimeter pixels and length

- Let perimeter P be the actual set of boundary pixels.
- P must be ordered in a sequence $P = < (r_o, c_o), \ldots, (r_{K-1}, c_{K-1}) >$.
- Each pair of successive pixels in P are neighbors, including the first and last pixels.

  perimeter length:

$$
\begin{aligned}
|P| &= \#\{k | (r_{k+1}, c_{k+1}) \in N_4(r_k, c_k)\} \\
&+ \sqrt{2}\#\{k | (r_{k+1}, c_{k+1}) \in N_8(r_k, c_k) - N_4(r_k, c_k)\}
\end{aligned}
$$

  where $k+1$ is computed modulo K.

- *Perimeter can vary significantly with object orientation.*

Circularity or elongation

- common measure of circularity of a region is length of the perimeter squared divided by area.

circularity(1):

$$C_1 = \frac{|P|^2}{A}$$

Circularity as variance of "radius"

- a second measure uses variation off of a circle

  circularity(2):

$$C_2 = \frac{\mu_R}{\sigma_R}$$

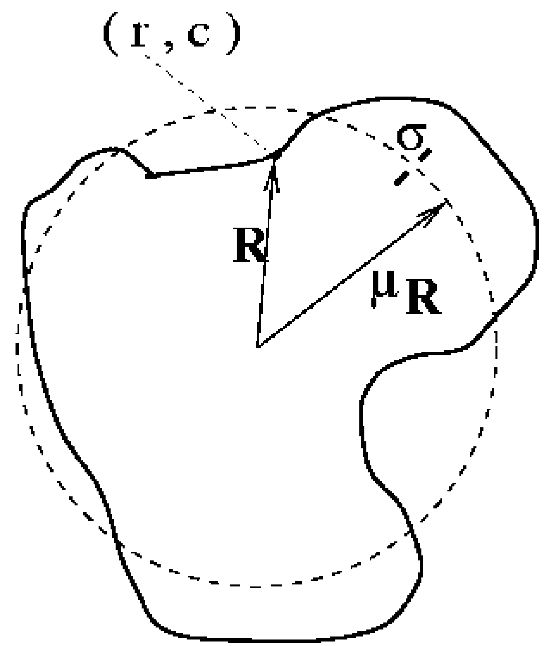
where μ_R and ${\sigma_R}^2$ are the mean and variance of the distance from the centroid of the shape to the boundary pixels (r_k, c_k).

mean radial distance:

$$\mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \| (r_k, c_k) - (\bar{r}, \bar{c}) \|$$

variance of radial distance:

$$\sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} [\| (r_k, c_k) - (\bar{r}, \bar{c}) \| - \mu_R]^2$$



Second moments

second-order row moment:

$$\mu_{rr} = \frac{1}{A} \sum_{(r,c)\in R} (r - \bar{r})^2$$

second-order mixed moment:

$$\mu_{rc} = \frac{1}{A} \sum_{(r,c)\in R} (r - \bar{r})(c - \bar{c})$$

second-order column moment:

$$\mu_{cc} = \frac{1}{A} \sum_{(r,c)\in R} (c - \bar{c})^2$$

These quantities are used as simple shape descriptors. They are invariant to translation and scale change of a 2D shape.

Region Adjacency Graph

A region adjacency graph (RAG) is a graph in which each node represents a region of the image and an edge connects two nodes if the regions are adjacent.

This is jumping ahead a little bit.

We'll consider this further for structural image analysis.