# Image Processing

CS 554 – Computer Vision

Pinar Duygulu
Bilkent University

# Today

Image Formation

Point and Blob Processing

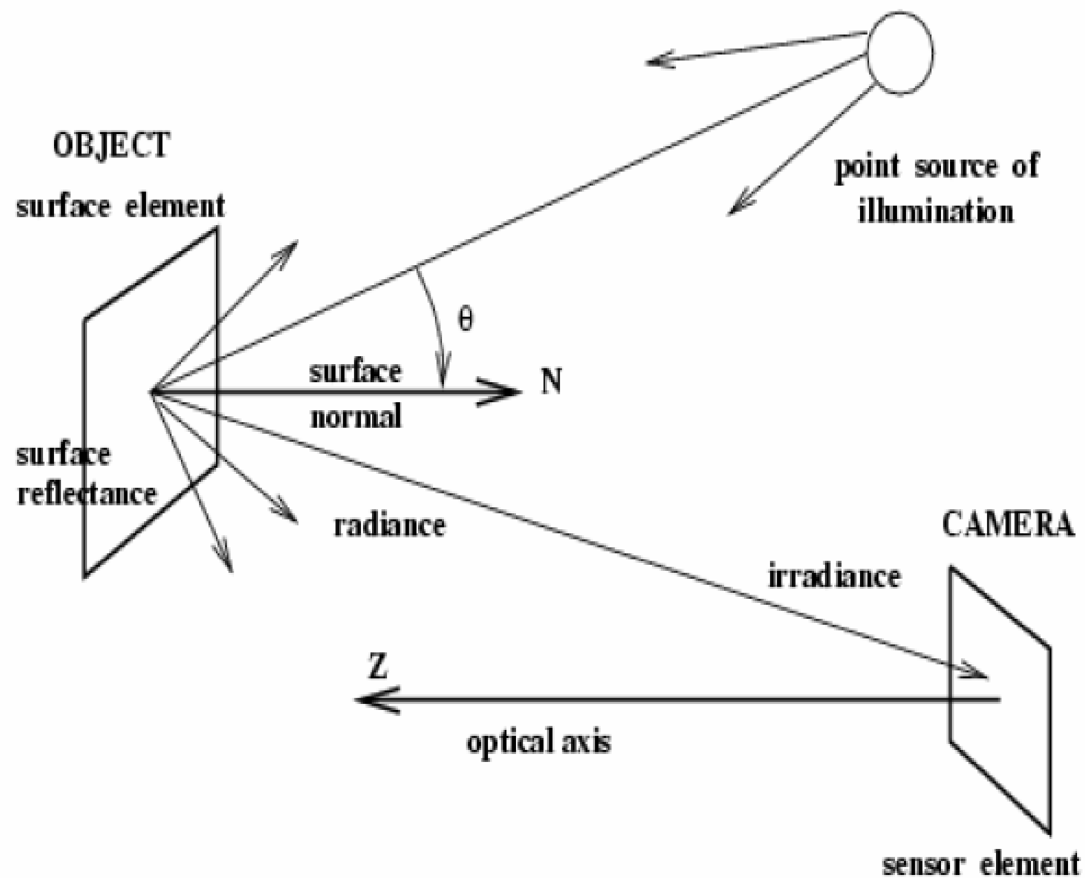
Binary Image Processing

Readings:
- Gonzalez & Woods, Ch. 3

Slides are adapted from Alyosha Efros and Shapiro & Stockman

# Imaging Process

- Light reaches surfaces in 3D.
- Surfaces reflect.
- Sensor element receives light energy.
- Intensity is important.
- Angles are important.
- Material is important.




Adapted from Shapiro and Stockman

# Image Formation

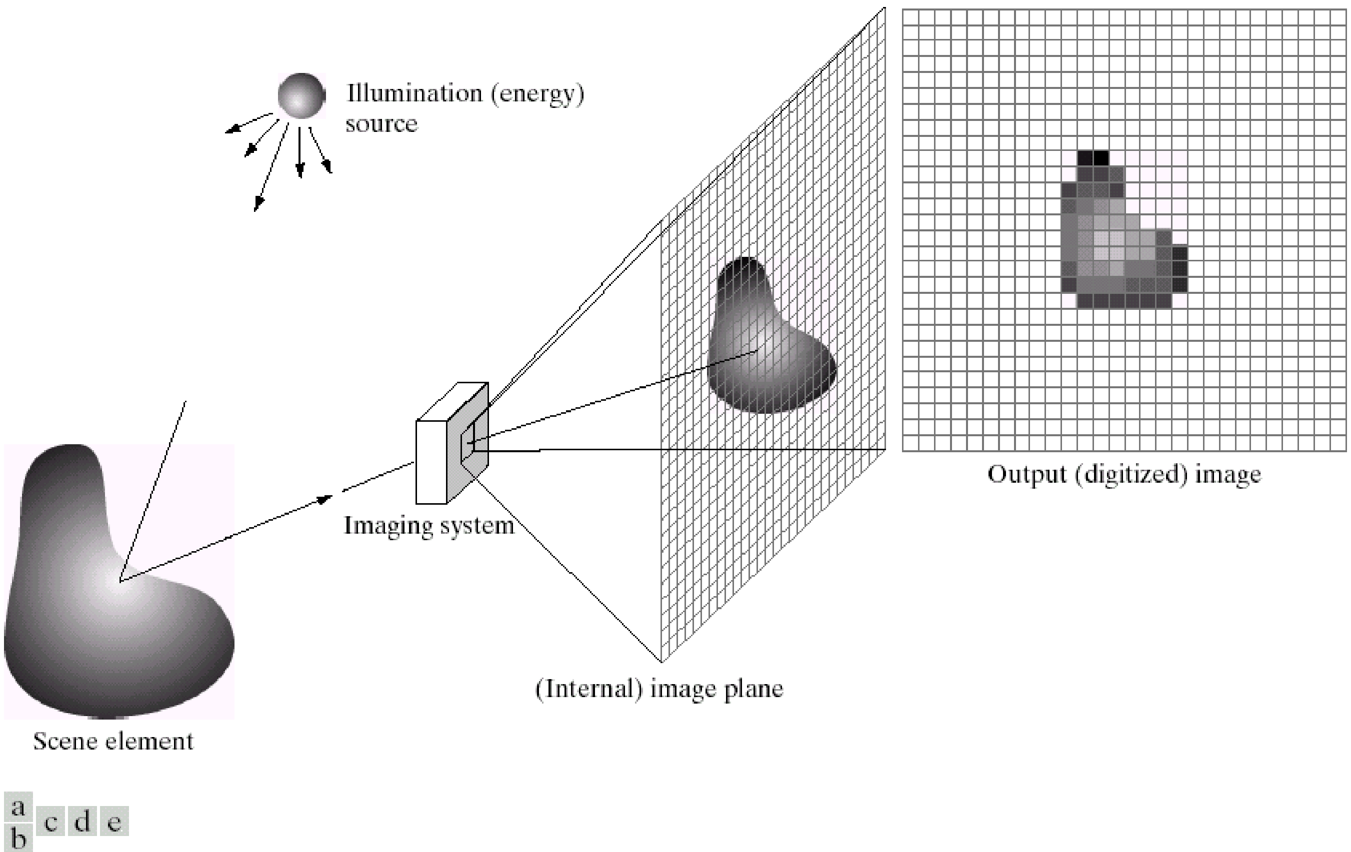



**FIGURE 2.15** An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

# Sampling and Quantization

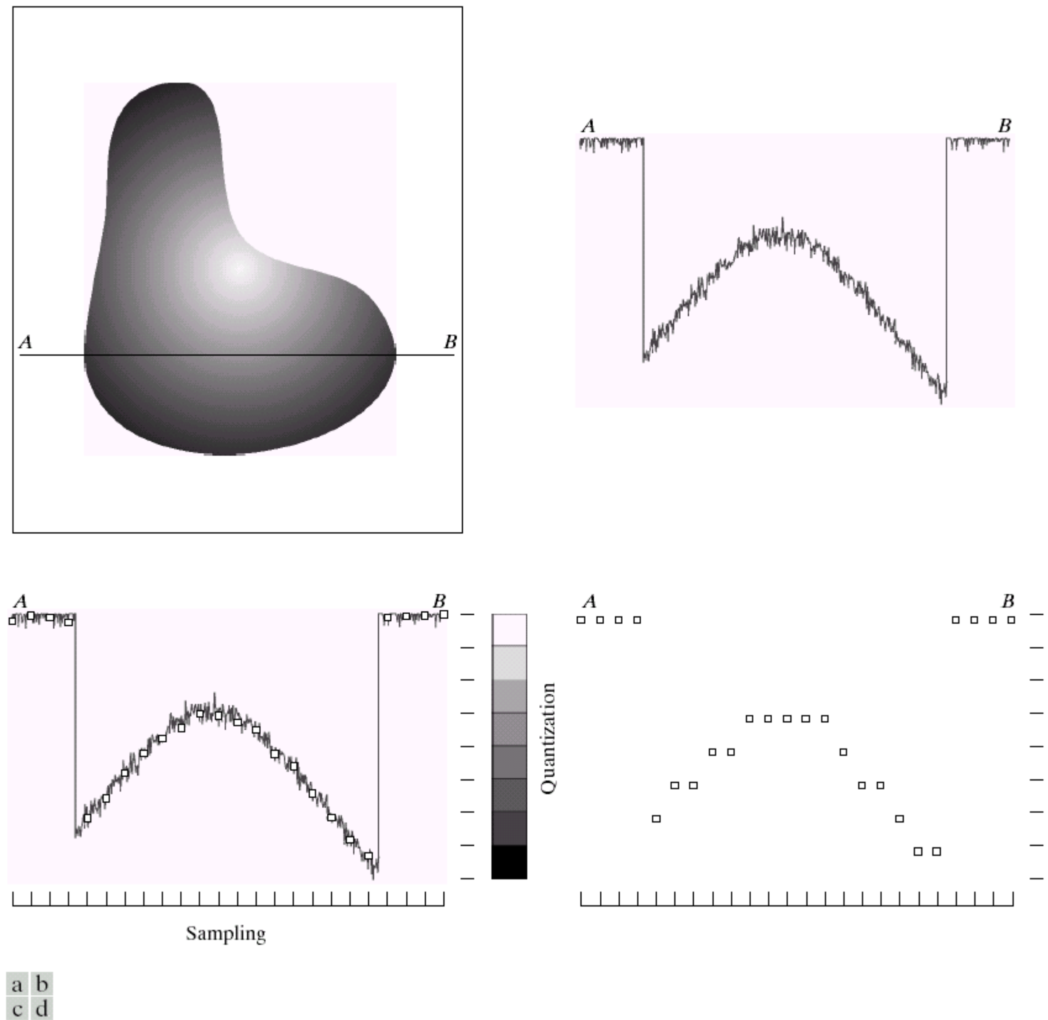



**FIGURE 2.16** Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

# Sampling and Quantization

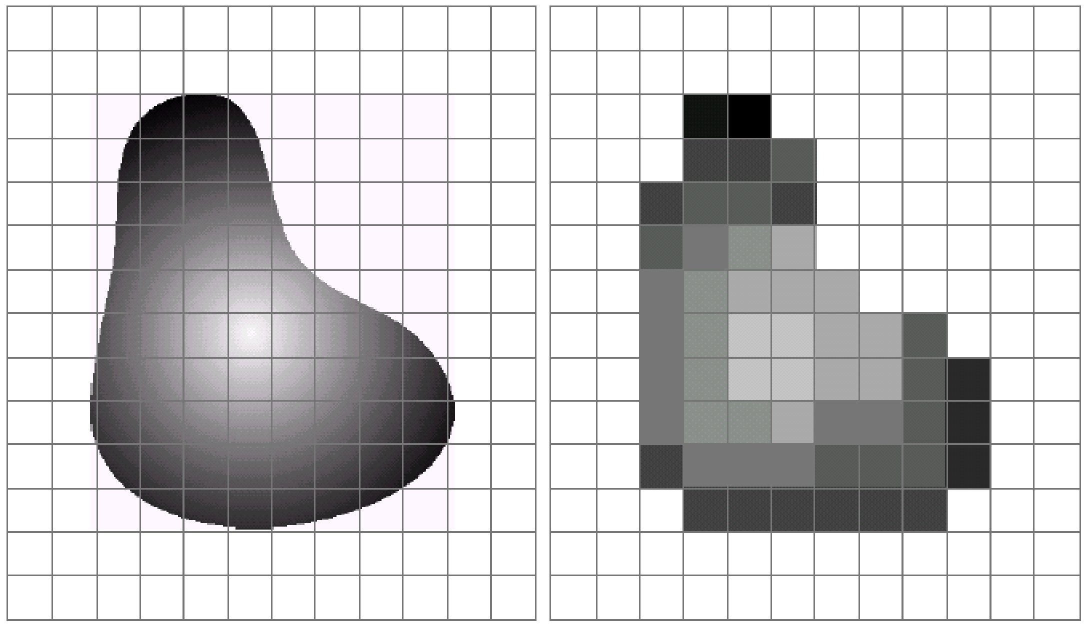




**FIGURE 2.17** (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

# What is an image?

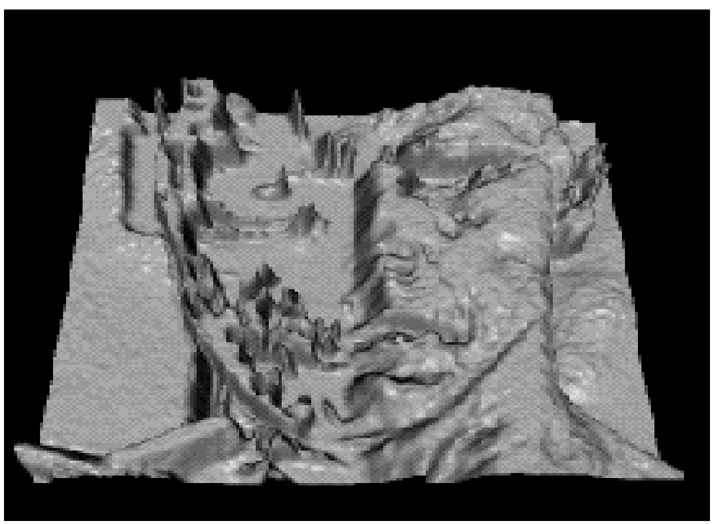
We can think of an **image** as a function, *f*, from $R^2$ to R:

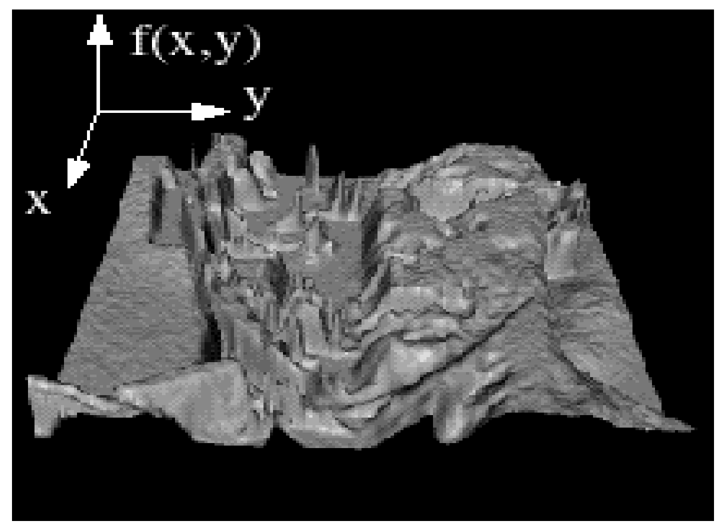
- $f(x, y)$ gives the **intensity** at position $(x, y)$
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - $f: [a,b] \times [c,d] \rightarrow [0,1]$

A color image is just three functions pasted together. We can write this as a "vector-valued" function:

$$f(x,y) = [r(x,y)\ g(x,y)\ b(x,y)]$$

# Images as functions





# What is a digital image?

We usually operate on **digital (discrete)** images:

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)

If our samples are Δ apart, we can write this as:

$f[i\ ,j] = \text{Quantize}\{\ f(i\ \Delta, j\ \Delta)\ \}$

The image can now be represented as a matrix of integer values

$j \rightarrow$

$i \downarrow$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 62 | 79 | 23 | 119 | 120 | 105 | 4 | 0 |
| 10 | 10 | 9 | 62 | 12 | 78 | 34 | 0 |
| 10 | 58 | 197 | 46 | 46 | 0 | 0 | 48 |
| 176 | 135 | 5 | 188 | 191 | 68 | 0 | 49 |
| 2 | 1 | 1 | 29 | 26 | 37 | 0 | 77 |
| 0 | 89 | 144 | 147 | 187 | 102 | 62 | 208 |
| 255 | 252 | 0 | 166 | 123 | 62 | 0 | 31 |
| 166 | 63 | 127 | 17 | 1 | 0 | 99 | 30 |

# Point Processing

An **image processing** operation typically defines a new image *g* in terms of an existing image *f*.

The simplest kind of range transformations are these independent of position x,y:

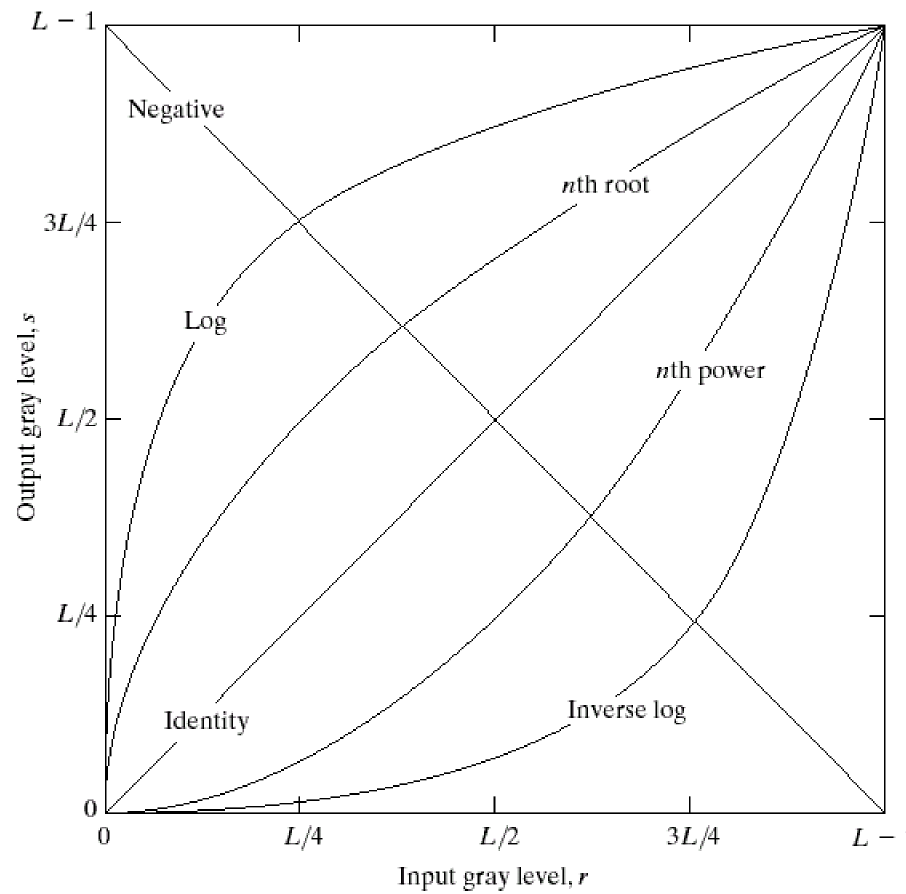
$$g = t(f)$$

This is called point processing.

**Important:** every pixel for himself – spatial information completely lost!

# Basic Point Processing

**FIGURE 3.3** Some basic gray-level transformation functions used for image enhancement.



# Negative

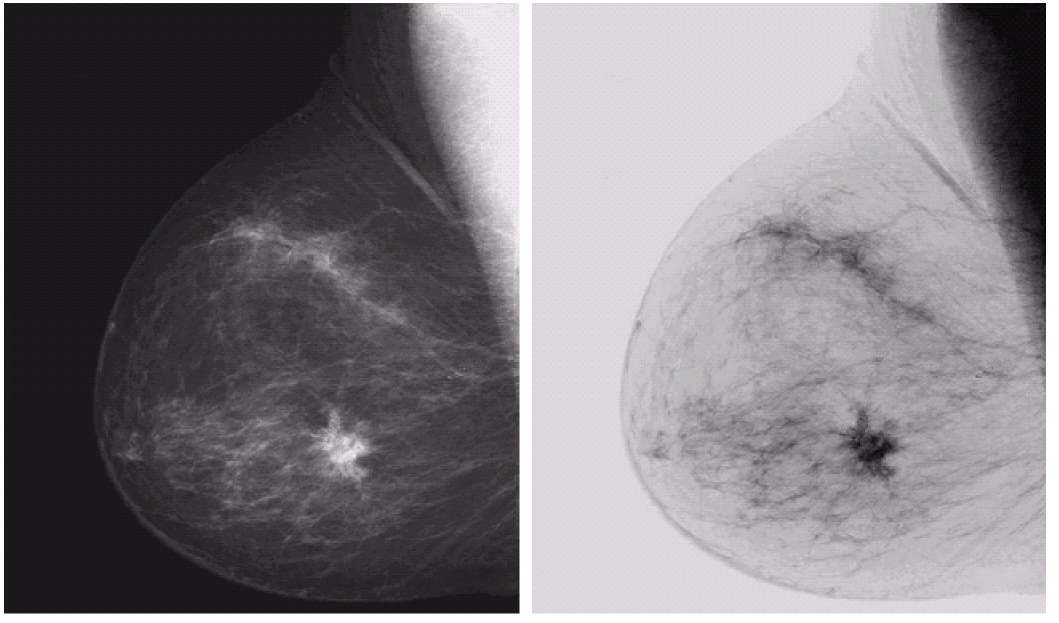


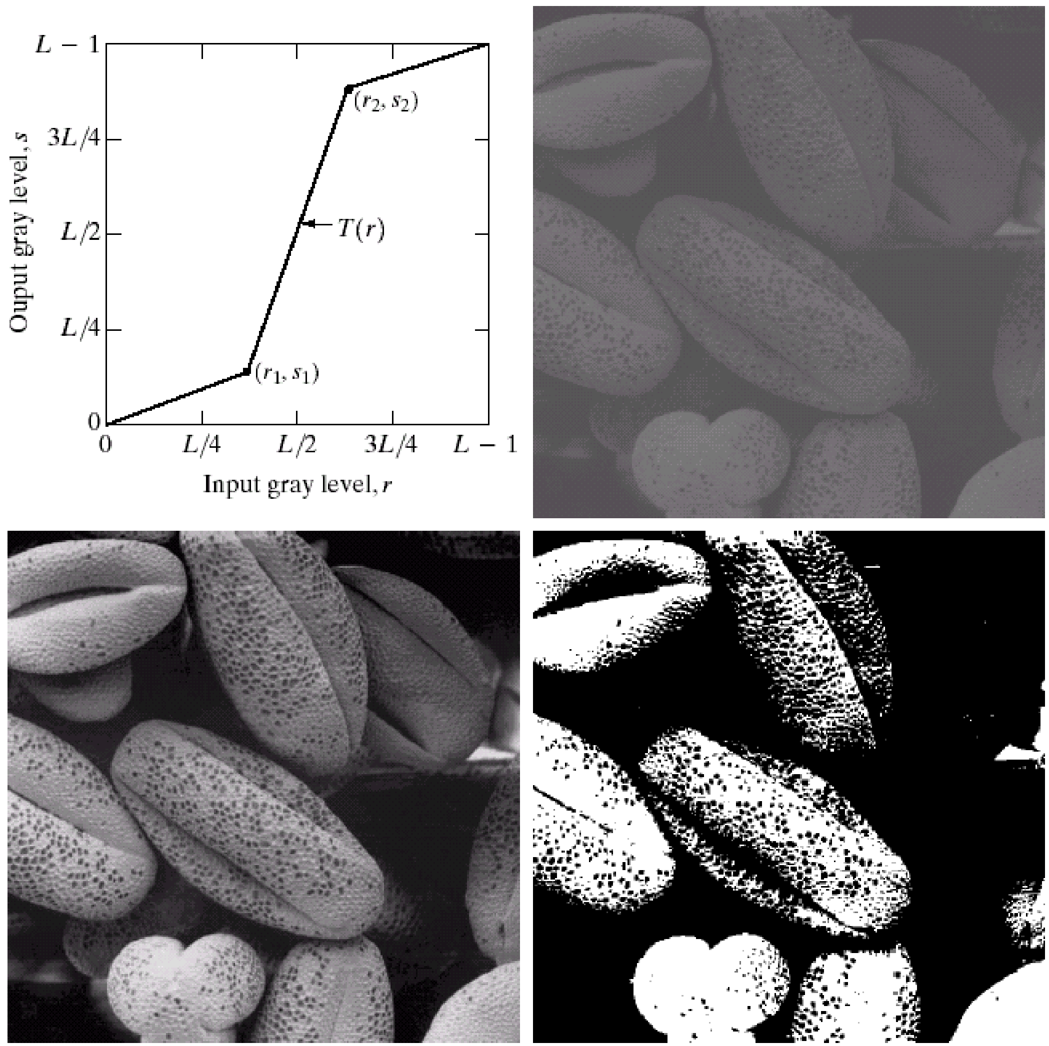

**FIGURE 3.4** (a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

# Image Enhancement

a b
c d

**FIGURE 3.9**
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0,$ and $5.0$, respectively. (Original image for this example courtesy of NASA.)

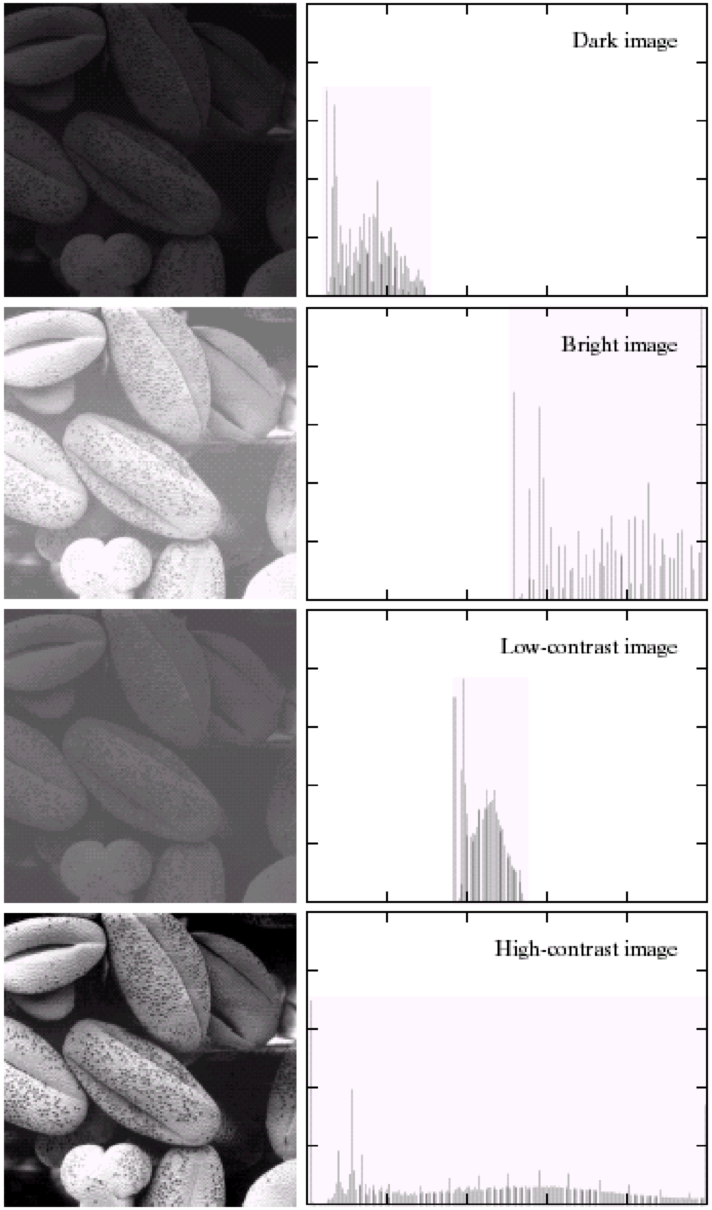
# Contrast Streching




a b
c d

**FIGURE 3.10** Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

# Image Histograms

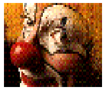
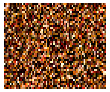



a b

**FIGURE 3.15** Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

# Neighborhood Processing (filtering)

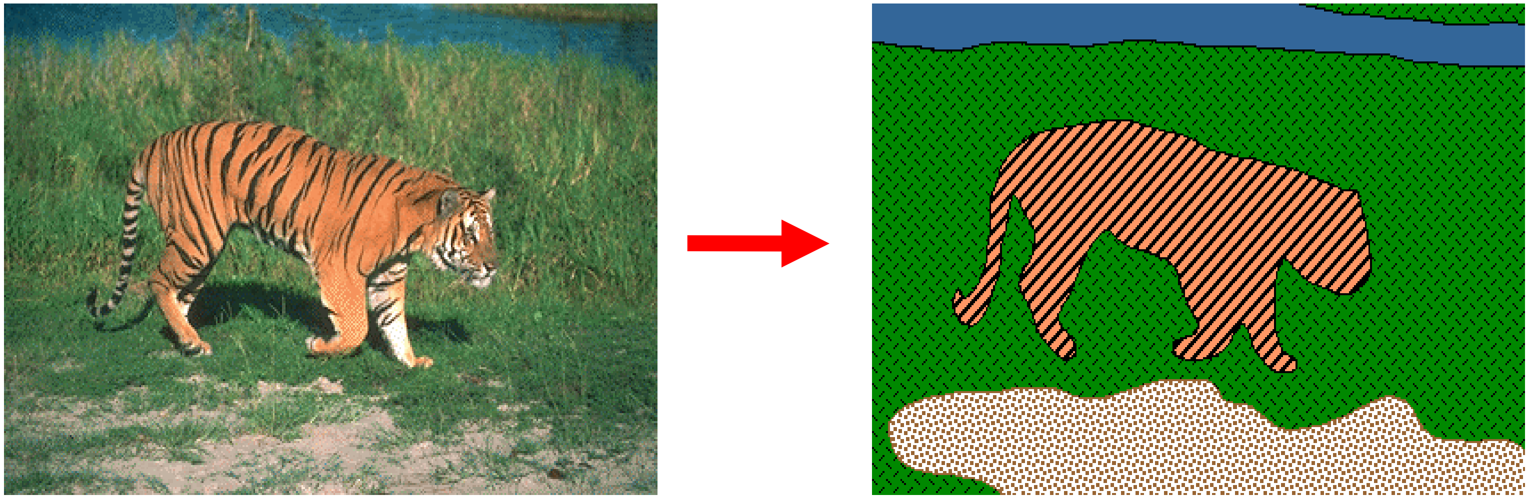
Q: What happens if I reshuffle all pixels within the image?

A: It's histogram won't change. No point processing will be affected…

Need spatial information to capture this.

# Goal: Extract "Blobs"



What are "blobs"?

– Regions of an image that are somehow coherent

Why?

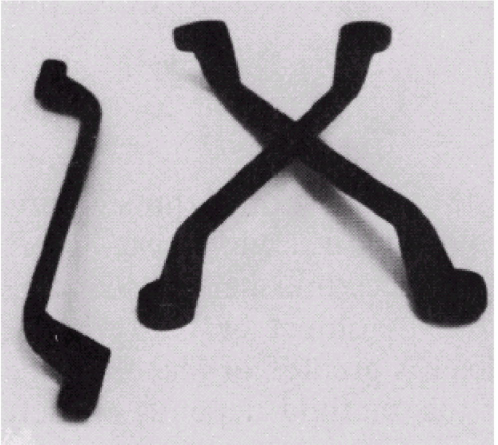
– Object extraction, object removal, compositing, etc.

– …but are "blobs" objects?

– No, not in general

# Blob's coherence

Simplest way to define blob coherence is as similarity in brightness or color:



The tools become blobs

The house, grass, and sky make different blobs

# The meaning of a blob

Other interpretations of blobs are possible, depending on how you define the input image:

- Image can be a <u>response</u> of a particular detector
  - Color Detector
  - Face detector
  - Motion Detector
  - Edge Detector

# Why is this useful?

AIBO RoboSoccer (VelosoLab)

# Ideal Segmentation

# Result of Segmentation

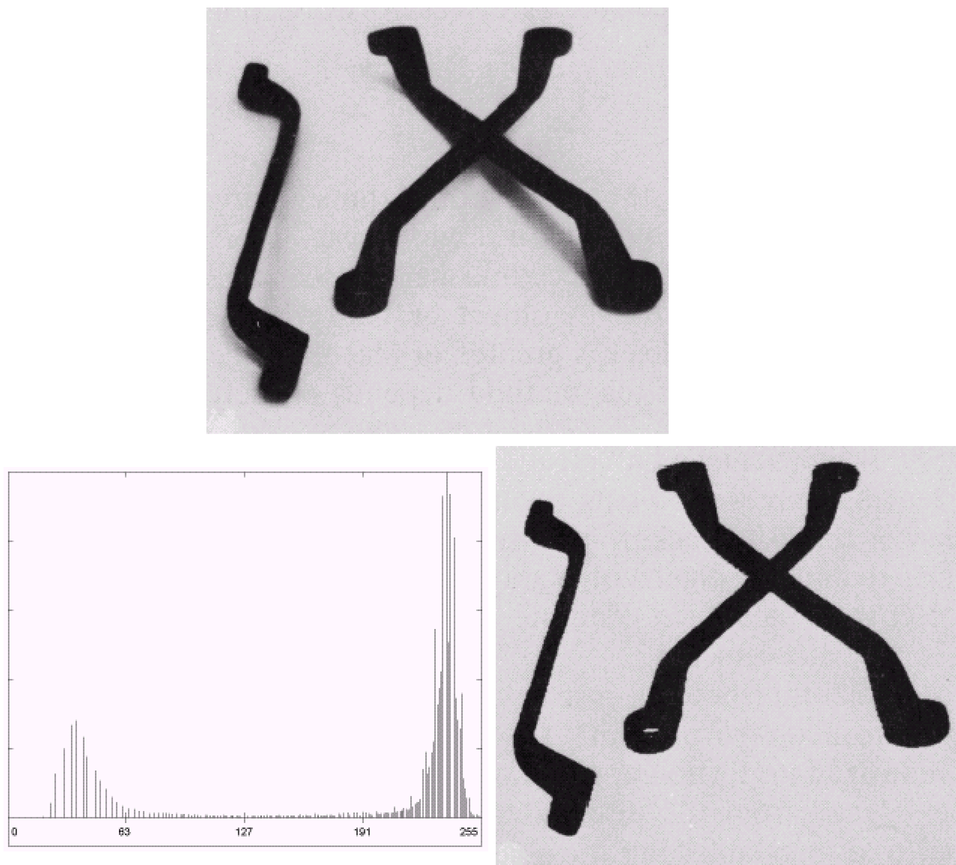
# Thresholding

Basic segmentation operation:

$$mask(x,y) = 1 \text{ if } im(x,y) > T$$
$$mask(x,y) = 0 \text{ if } im(x,y) < T$$

T is threshold
- User-defined
- Or automati

Same as histogram partitioning:




a
b c

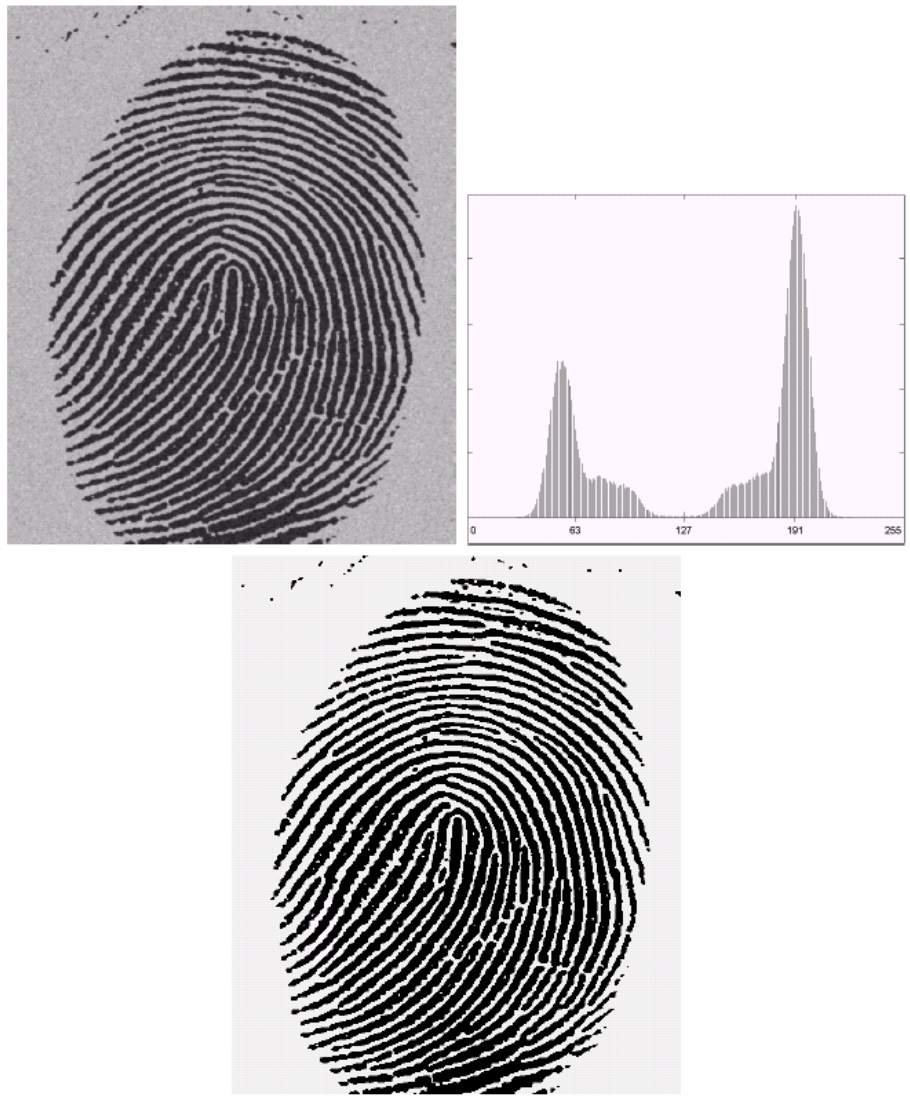
**FIGURE 10.28** (a) Original image. (b) Image histogram. (c) Result of global thresholding with $T$ midway between the maximum and minimum gray levels.

# As Edge Detection

$gx^2+gy^2$

$gx^2+gy^2 > T$

# Sometimes works well…




a b
c

**FIGURE 10.29** (a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

What are potential Problems?

# …but more often not

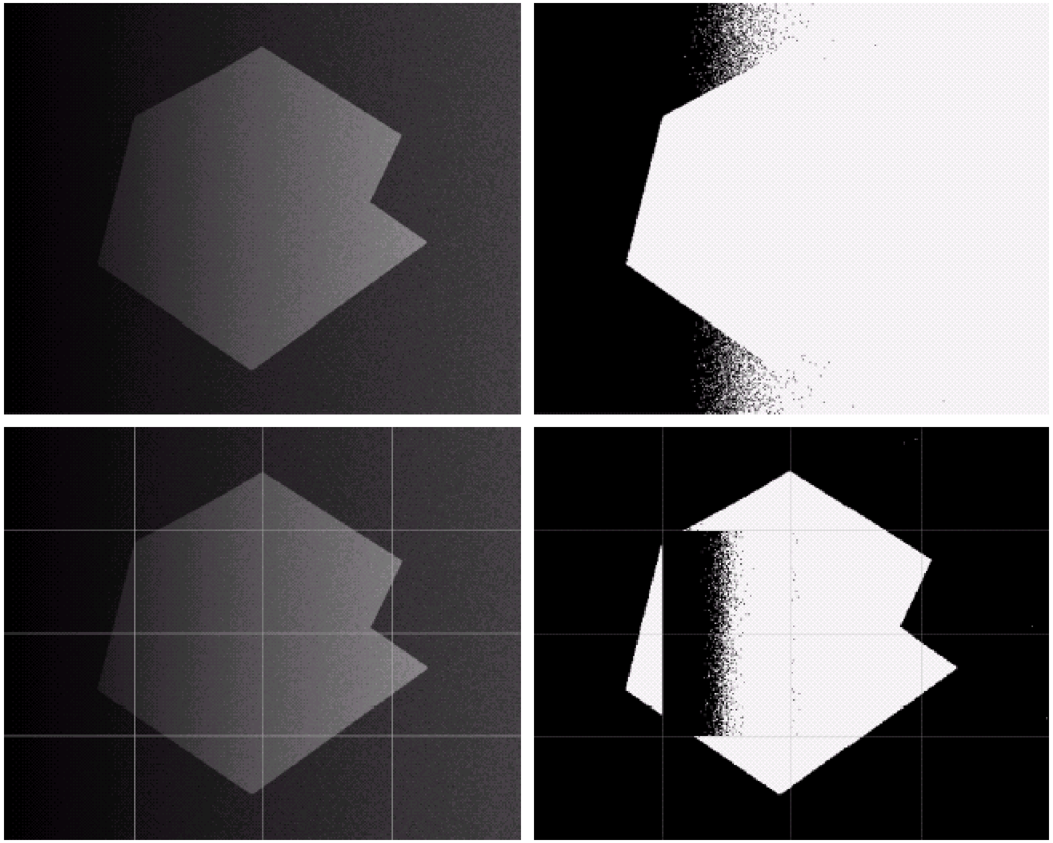
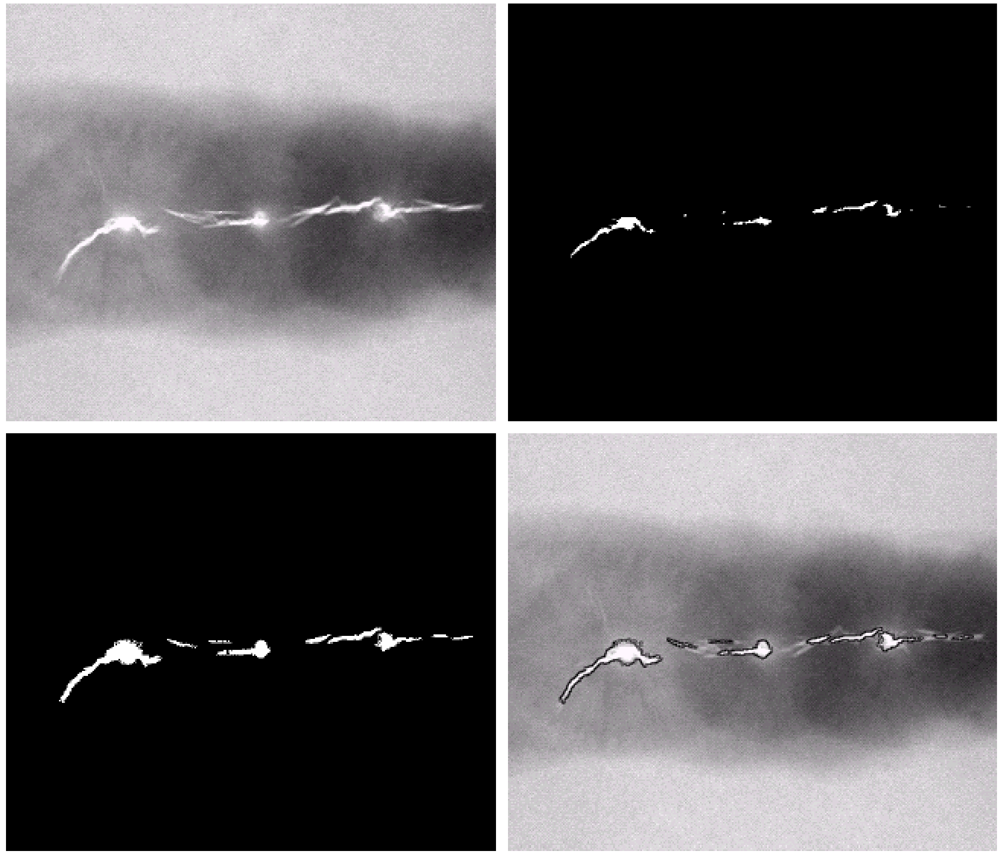




**FIGURE 10.30** (a) Original image. (b) Result of global thresholding. (c) Image subdivided into individual subimages. (d) Result of adaptive thresholding.

Adaptive thresholding

# Region growing



a b
c d

**FIGURE 10.40** (a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).

- Start with initial set of pixels K
- Add to K any neighbors, if they are within similarity threshold
- Repeat until nothing changes

Is this same as global threshold?
What can go wrong?

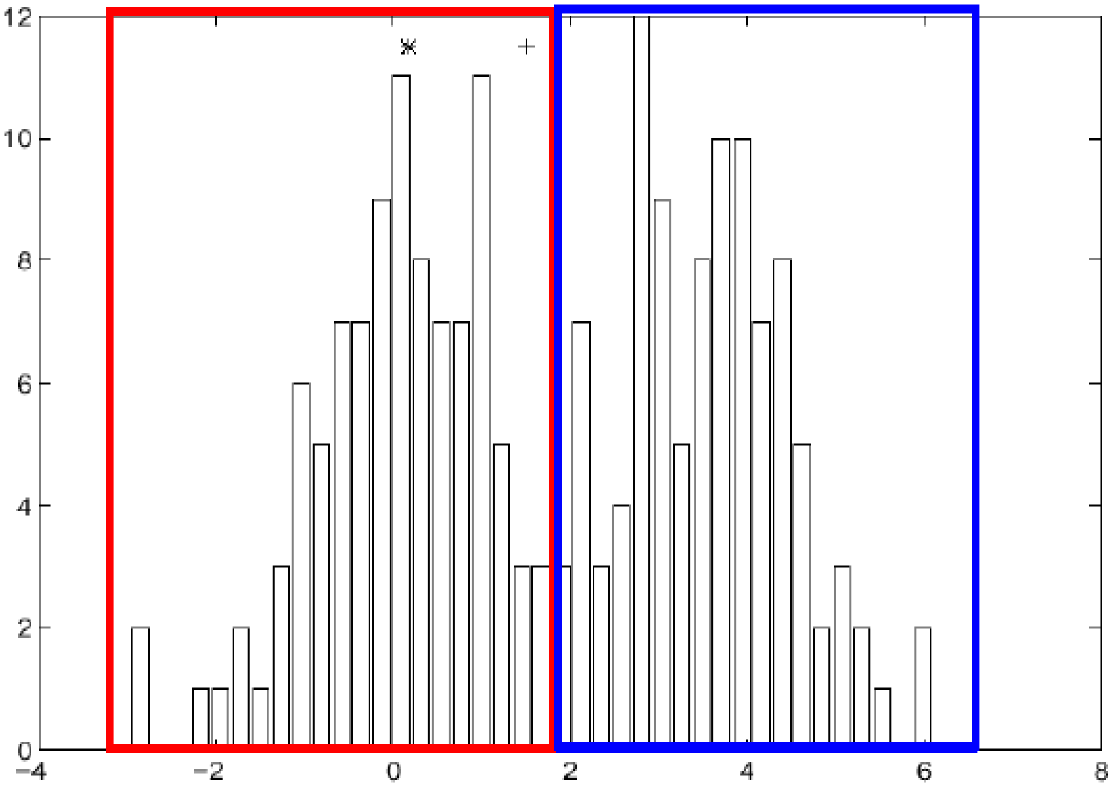
# Color-Based Blob Segmentation

Automatic Histogram Partitioning

- Given image with N colors, choose K
- Each of the K colors defines a region
  - not necessarily contiguous
- Performed by computing color histogram, looking for modes

- This is what happens when you downsample image color range, for instance in Photoshop

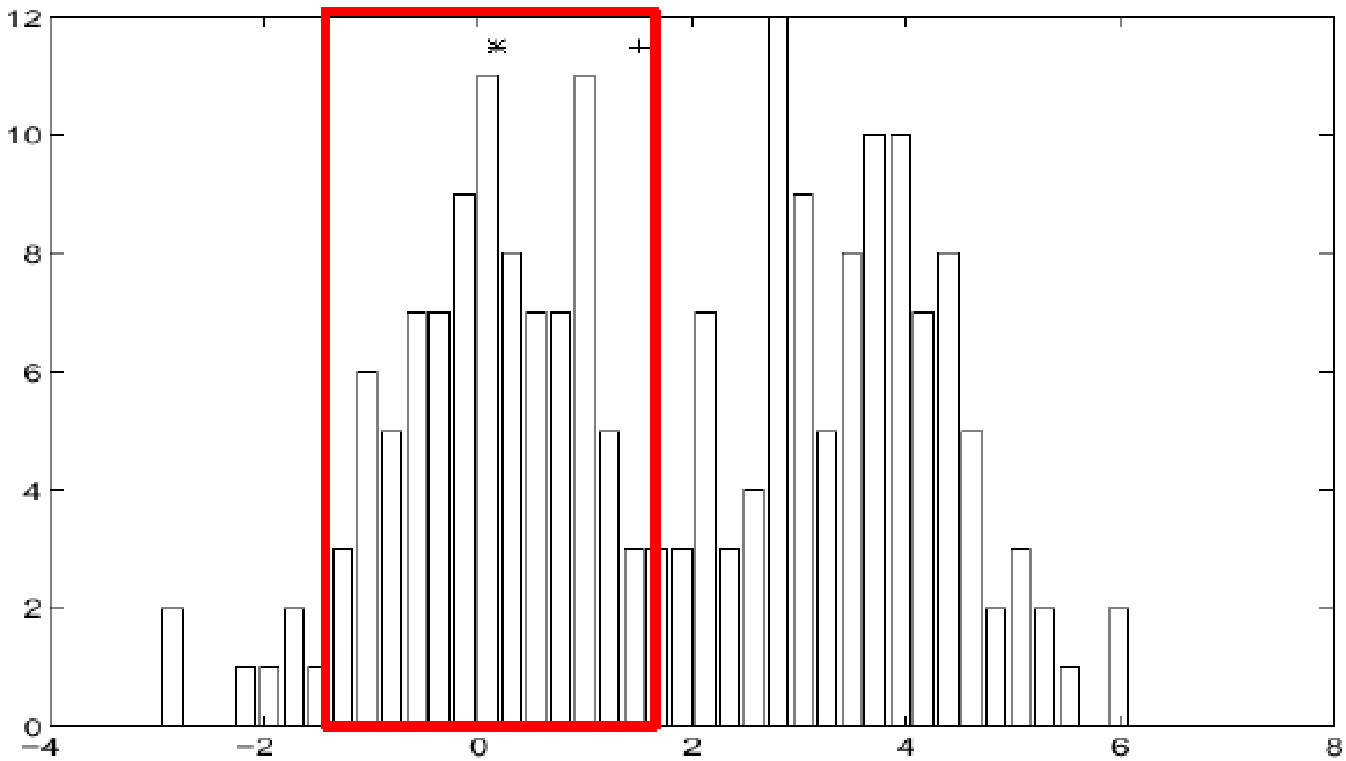
# Finding modes in a histogram




How Many Modes Are There?

- Easy to see, hard to compute

# Mean-Shift (Comaniciu & Meer)




Iterative Mode Search

1. Initialize random seed, and fixed window
2. Calculate center of gravity of the window (the "mean")
3. Translate the search window to the mean
4. Repeat Step 2 until convergence

# Mean-Shift (Comaniciu & Meer)

More Examples: http://www.caip.rutgers.edu/~comanici/segm_images.html

# Issues:

Although often useful, all these approaches work only some of the time, and are considered rather “hacky”.

Can’t even handle our tiger:

Problem is that blobs != objects!

# Binary Image Processing

Binary images are handy in many cases (sprite extraction, compositing, etc).

Binary image processing is a well-studied field, based on set theory, called Mathematical Morphology

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.

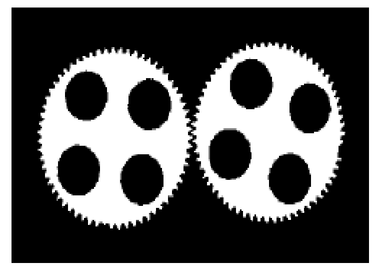
0 represents the background
1 represents the foreground

```
000100100010000
000111100010000
000100100010000
```

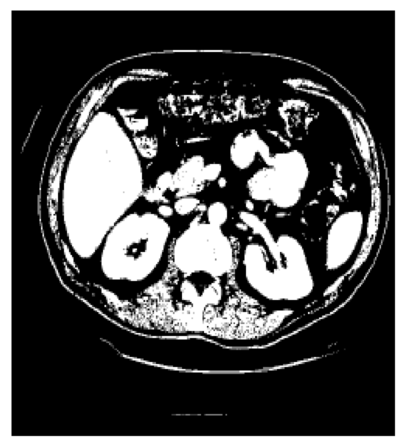
# Application Areas

Document Analysis


Industrial Inspection



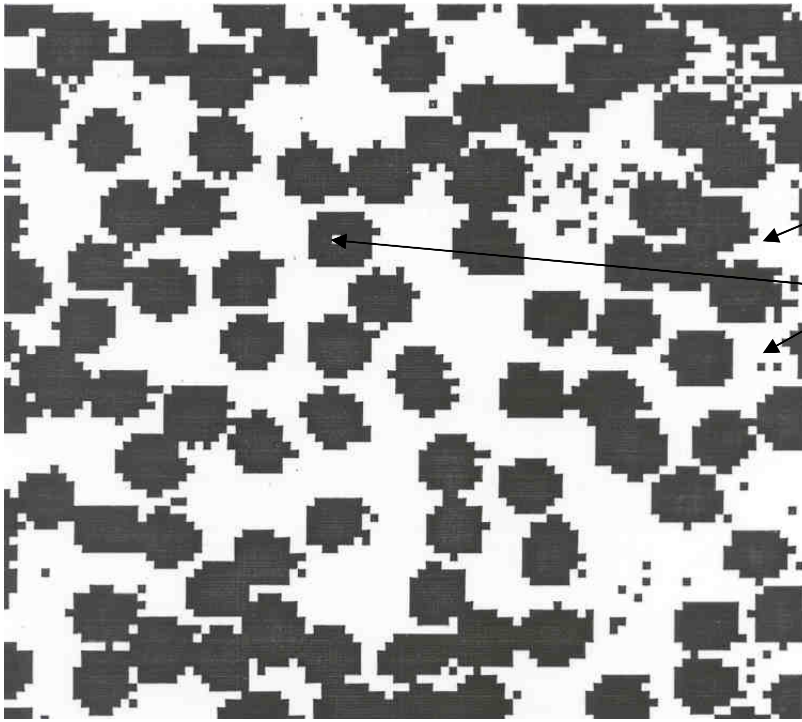
Medical Imaging



# Operations

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

# Example red blood cell image

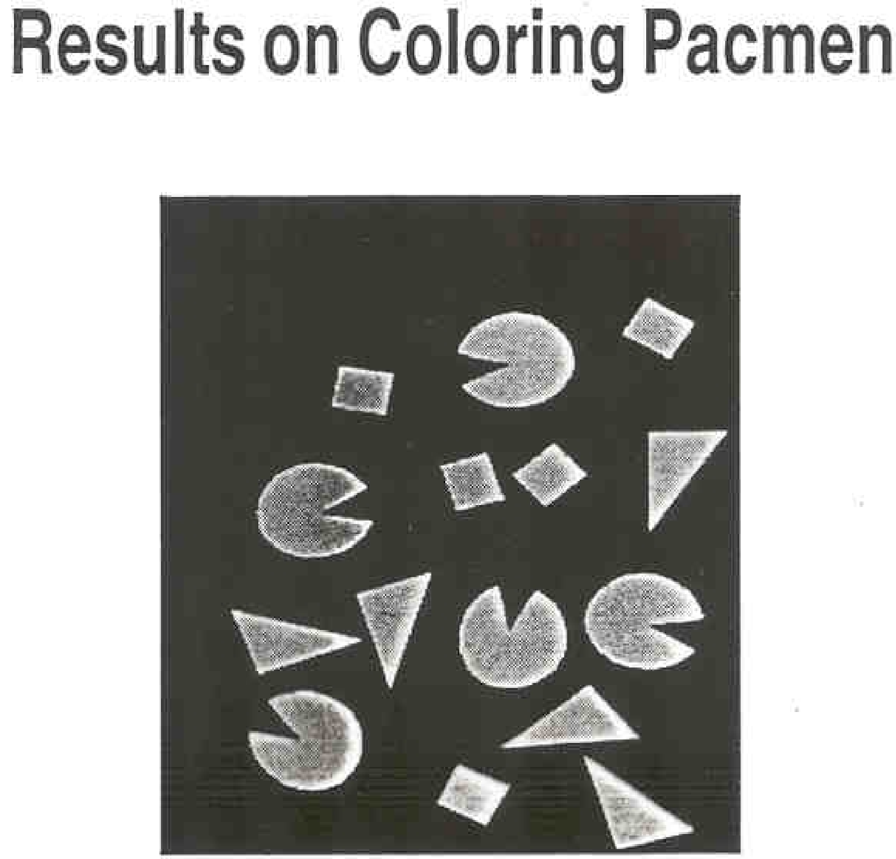


- Many blood cells are separate objects
- Many touch to each other – bad!
- Salt and pepper noise from thresholding
- How usable is this data?

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

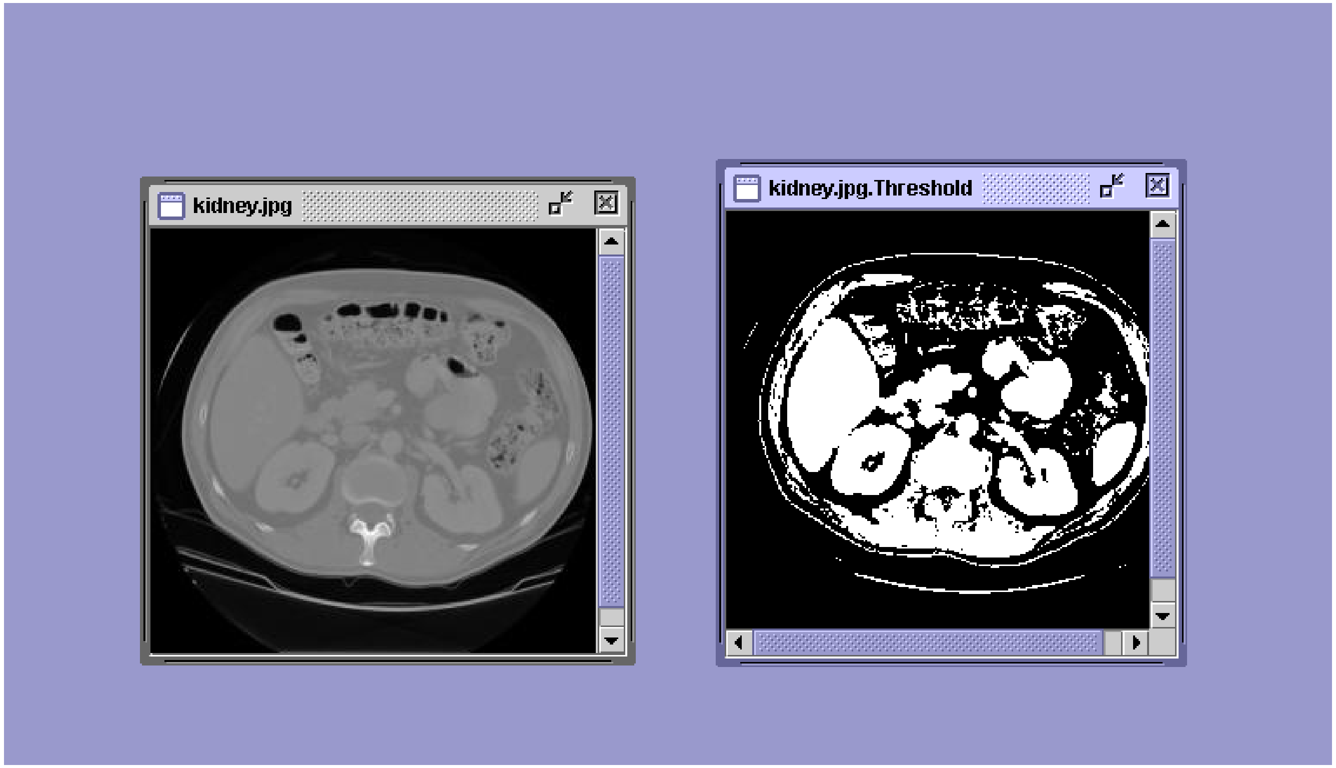
# More controlled images

- More uniform objects
- More uniform background
- Objects actually separated




- 15 objects detected
- Location known
- Area known
- 3 distinct clusters of 5 values of area; 85, 145, 293

# Thresholding



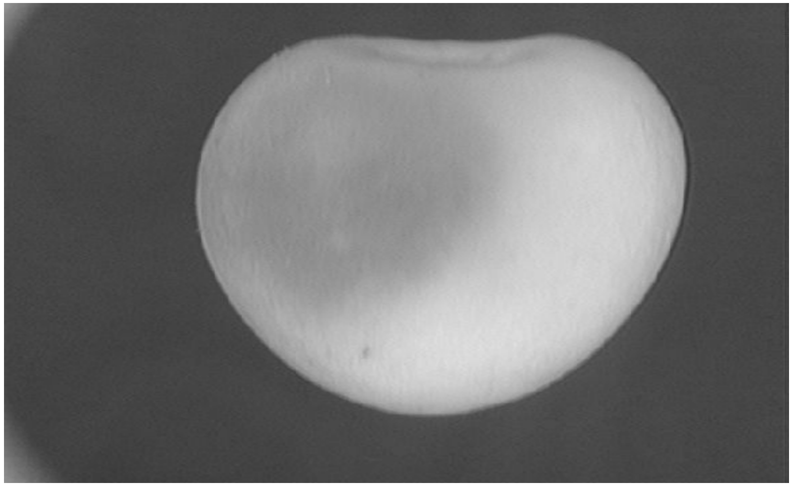

Binary images can be obtained from gray level images by thresholding

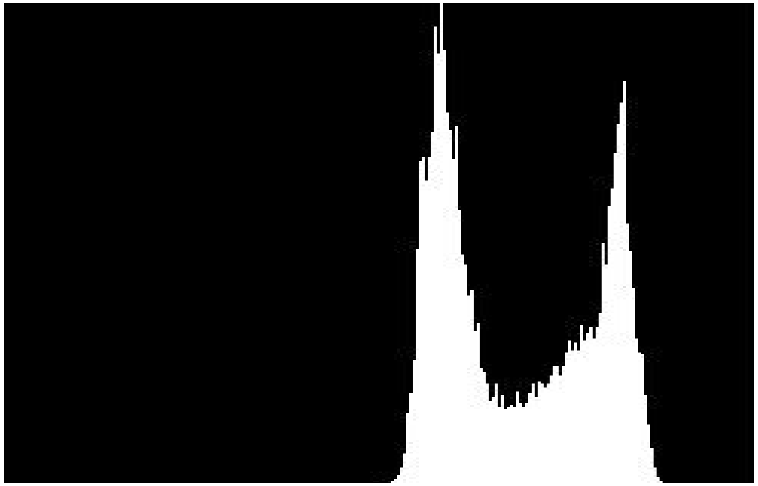
# Assumptions for thresholding

- Object region of interest has intensity distribution different from background
- Region pixel likely to be identified by intensity alone:
  - intensity > a
  - intensity < b
  - a < intensity < b
- Works OK with flat-shaded scenes or engineered scenes.
- Does not work well with natural scenes.

# Use of histograms for threshold selection

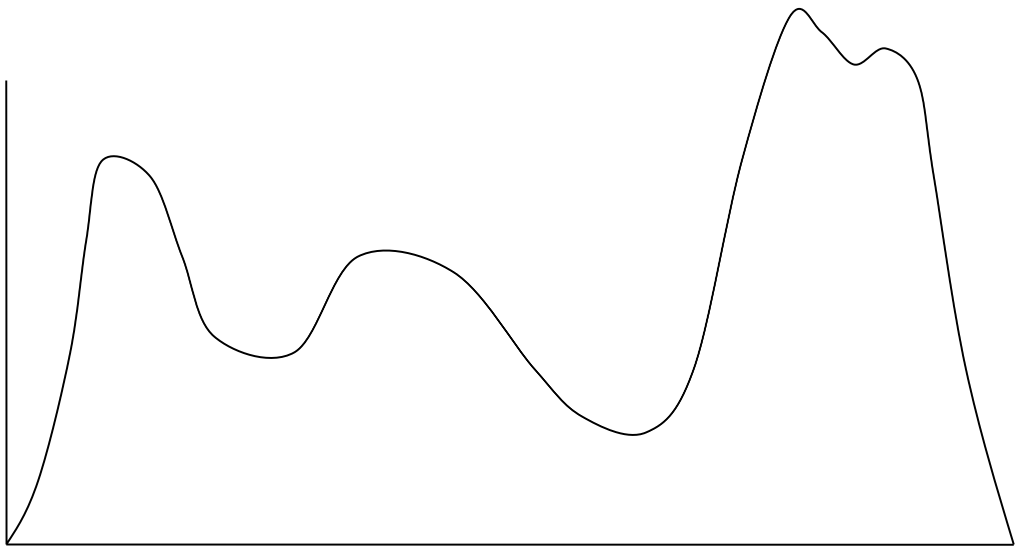
- Cherry image with 3 regions
- Background is black
- Healthy cherry is bright
- Bruise is medium dark
- Histogram shows two cherry regions (black background has been removed)





# Histogram-directed thresholding

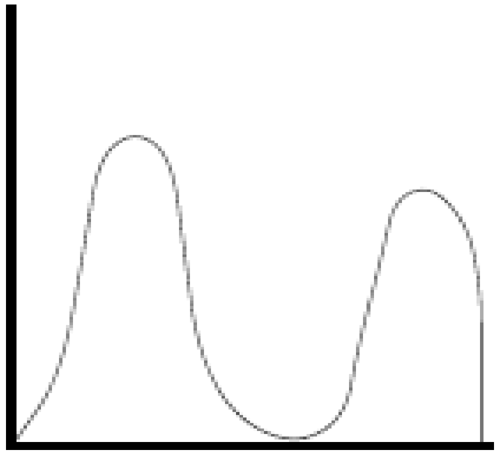
**How can we use a histogram to separate an image into 2 (or several) different regions?**



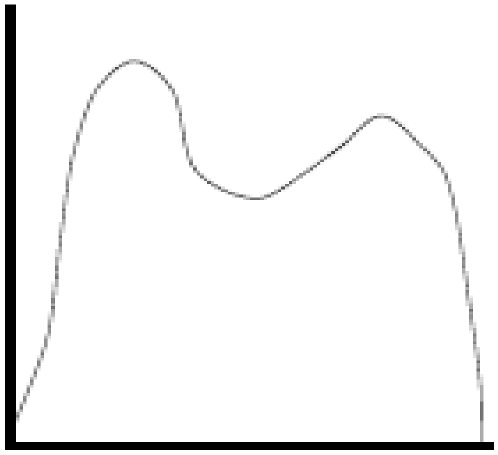
**Is there a single clear threshold? 2? 3?**

# Choosing Threshold

Detect peaks and valleys



Two distinct modes



Overlapped modes

# Choosing threshold

- Find the deepest valley between two modes of bimodal histogram
- Fit two or more Gaussian curves to the histogram
- Dynamic thresholding

# Cleaning up thresholding results

- Delete object pixels on boundary to better separate parts.
- Fill small holes
- Delete tiny objects
- (last 2 are "salt-and-pepper" noise)

# Mathematical Morphology

Binary mathematical morphology consists of two basic operations

**dilation and erosion**

and several composite relations

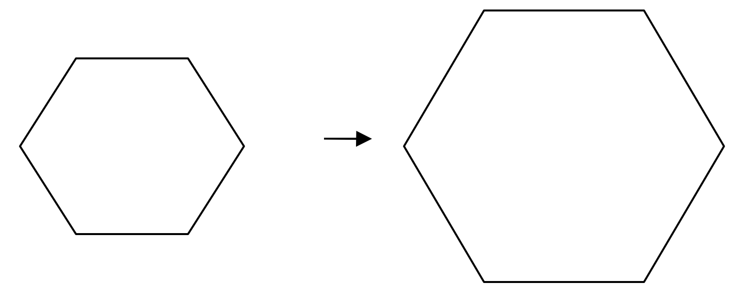
**closing and opening**
**conditional dilation**
**. . .**

# Dilation

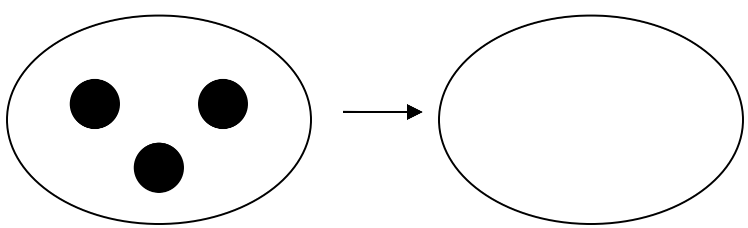
Dilation expands the connected sets of 1s of a binary image.

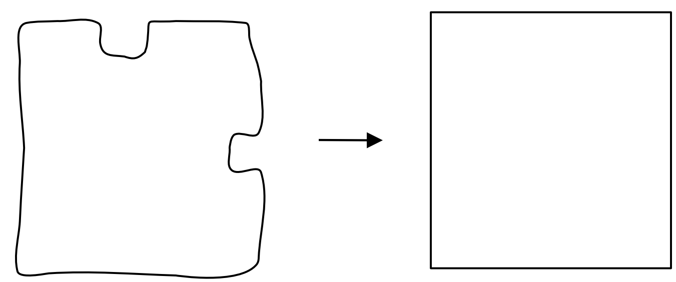
It can be used for

1. growing features



2. filling holes and gaps



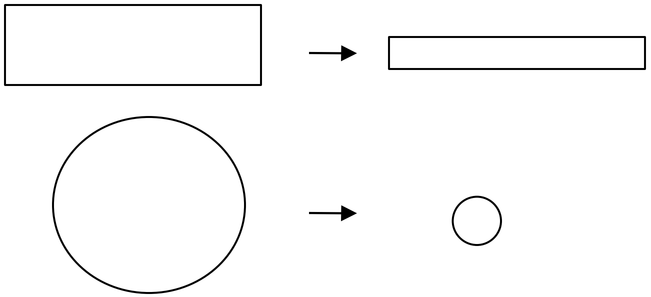


# Erosion

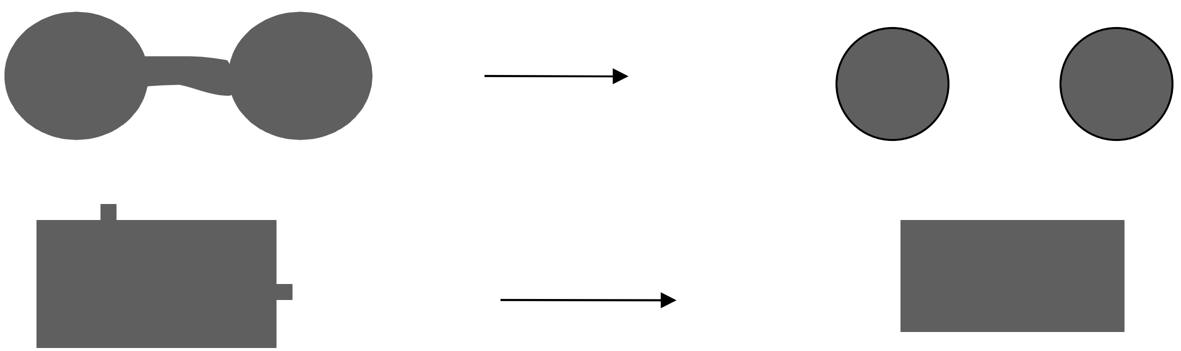
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for

1. shrinking features



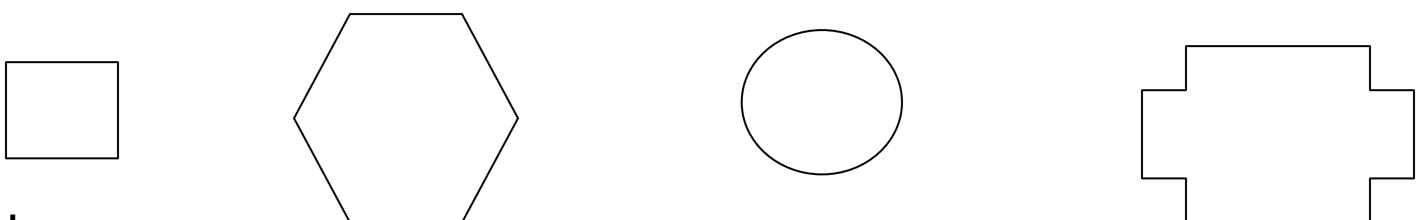
2. Removing bridges, branches and small protrusions



# 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 hexagon disk something

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.

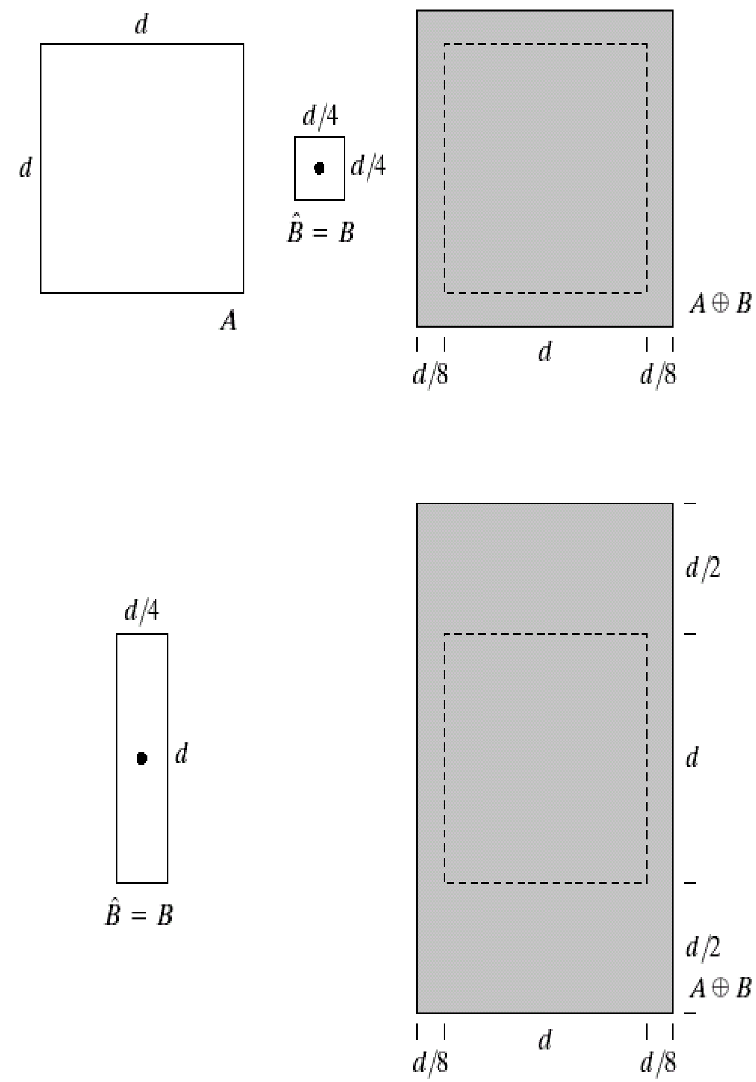
$$\underset{B}{\boxed{\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}}} \qquad \underset{S}{\boxed{\begin{matrix} 1 & \\ 1 & 1 \end{matrix}}} \qquad \xrightarrow{\text{dilate}} \qquad \underset{B \oplus S}{\boxed{\begin{matrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{matrix}}}$$

origin

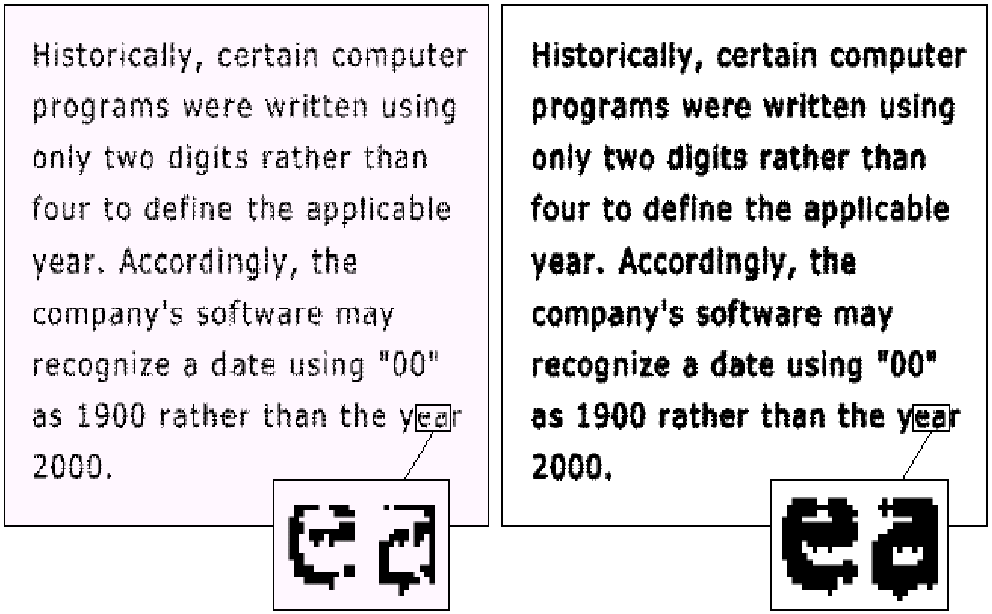
# Dilation

a b c
d e

**FIGURE 9.4**
(a) Set $A$.
(b) Square structuring element (dot is the center).
(c) Dilation of $A$ by $B$, shown shaded.
(d) Elongated structuring element.
(e) Dilation of $A$ using this element.



# Dilation




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

a c
b

**FIGURE 9.5** (a) Sample text of poor resolution with broken characters (magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.

# 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

$$\underset{B}{\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}} \qquad \underset{S}{\begin{bmatrix} 1 \\ \color{red}{1} \\ 1 \end{bmatrix}} \quad \xrightarrow{\text{erode}} \quad \underset{B \ominus S}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}$$

# Erosion

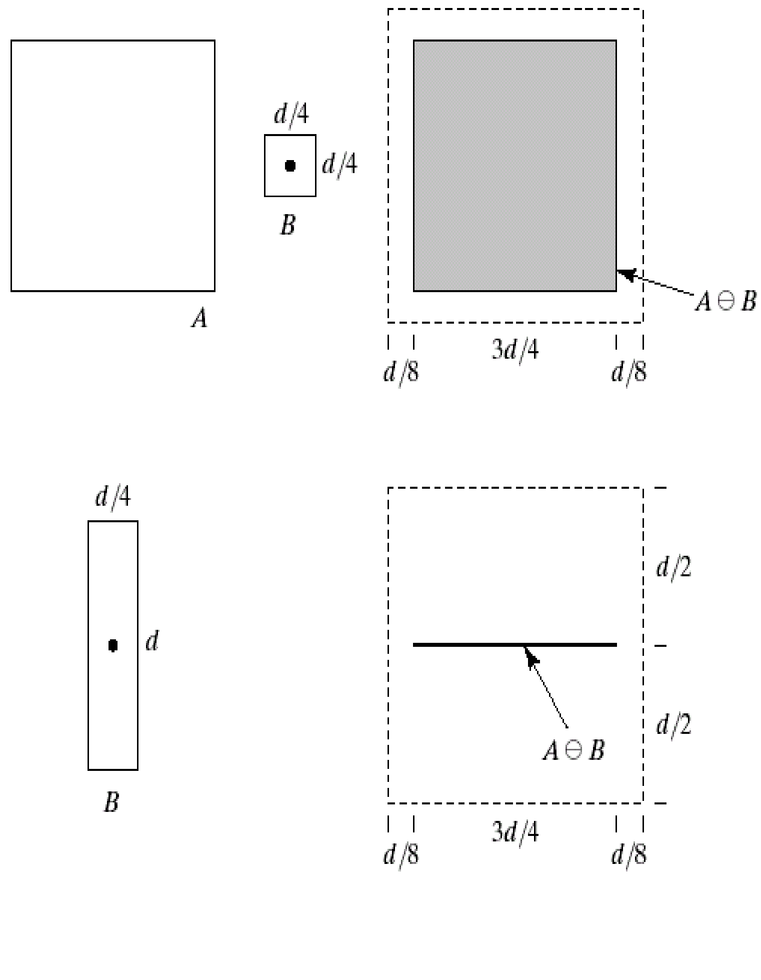
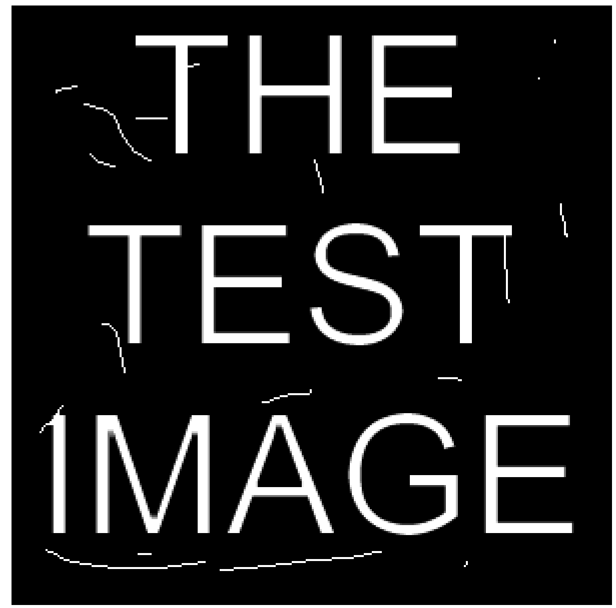



**FIGURE 9.6** (a) Set A. (b) Square structuring element. (c) Erosion of $A$ by $B$, shown shaded. (d) Elongated structuring element. (e) Erosion of $A$ using this element.

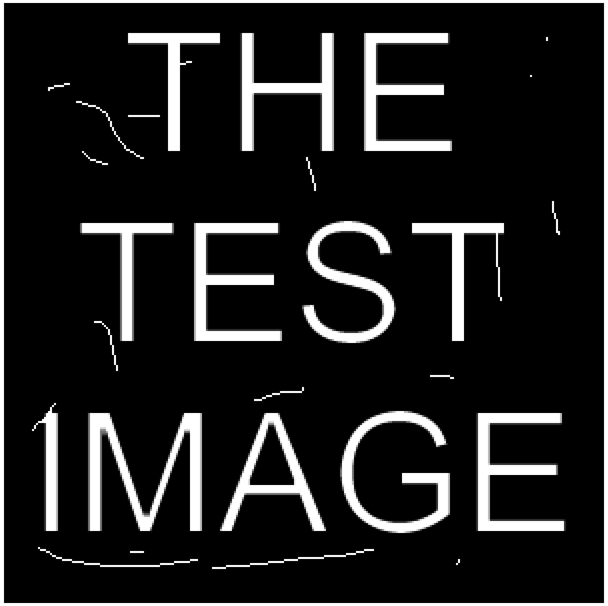
# Erosion


Original image




Eroded image

# Erosion




Eroded once


Eroded twice

# Opening and Closing

- Closing is the compound operation of dilation followed by erosion (with the same structuring element)

- Opening is the compound operation of erosion followed by dilation (with the same structuring element)

# Opening and Closing

Opening : smoothes the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions

$$A \circ B = (A \Theta B) \oplus B$$

Closing : smooth sections of contours but, as opposed to opning, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour

$$A \bullet B = (A \oplus B) \Theta B$$

Prove to yourself that they are not the same thing. Play around with `bwmorph` in Matlab.

# Example 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 | | | | |
| | | | | | | | |

a) Binary image B

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

b) Structuring Element S

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

c) Dilation $B \oplus S$

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

d) Erosion $B \ominus S$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | 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 | | | | |
| | | | | | | | |

e) Closing $B \bullet S$

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

f) Opening $B \circ S$

# Example 2: try

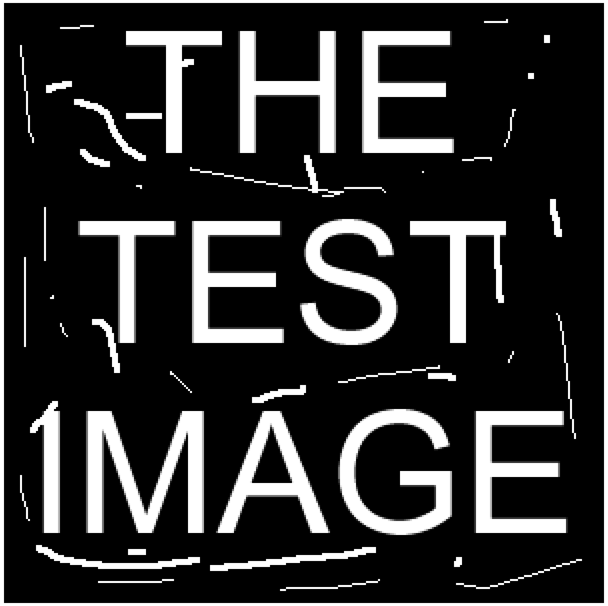
B

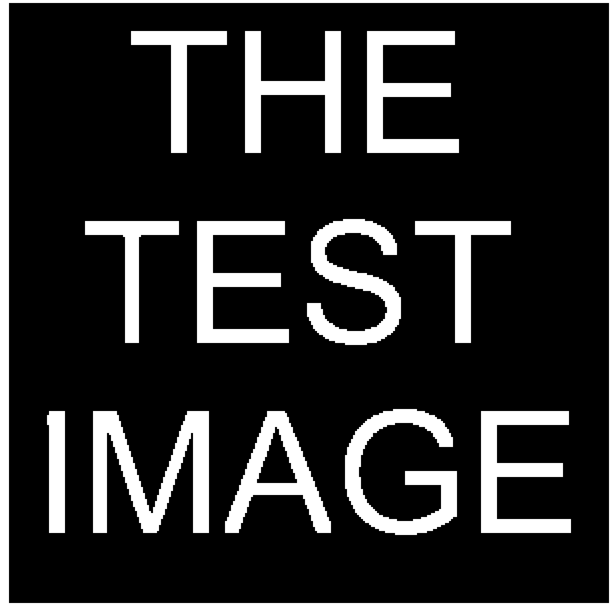
| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |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

S

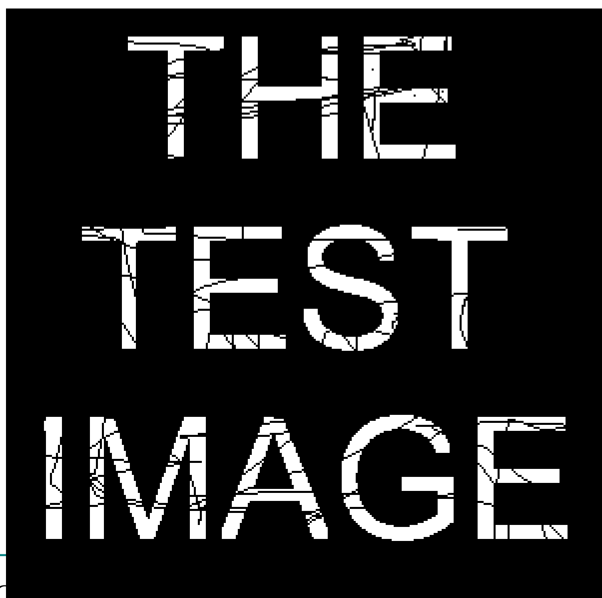
| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |

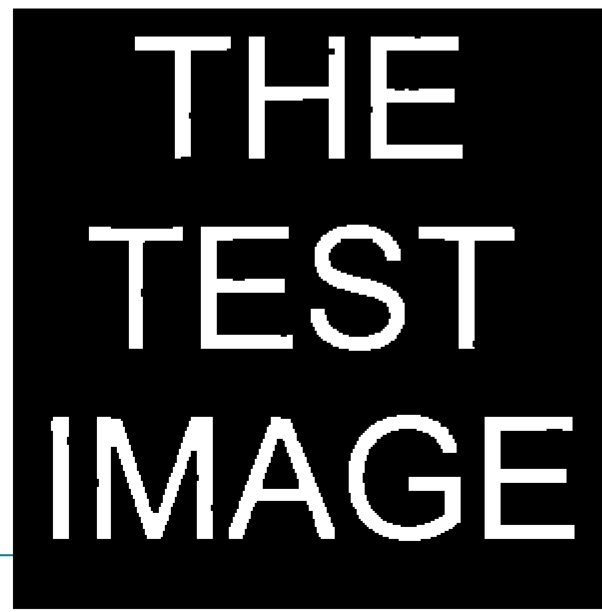
# Opening and Closing




OPENING: The original image eroded twice and dilated twice (opened). Most noise is removed




CLOSING: The original image dilated and then eroded. Most holes are filled.

# Opening and Closing

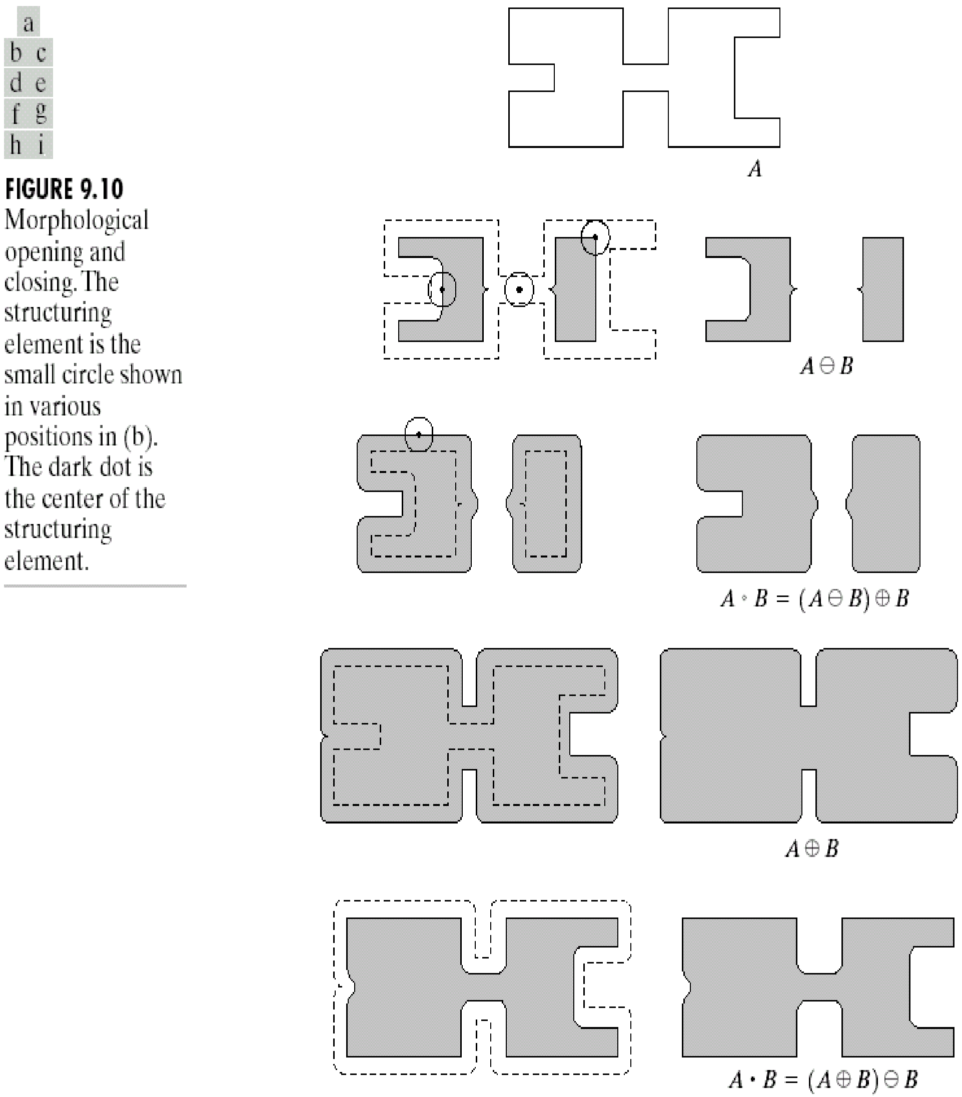
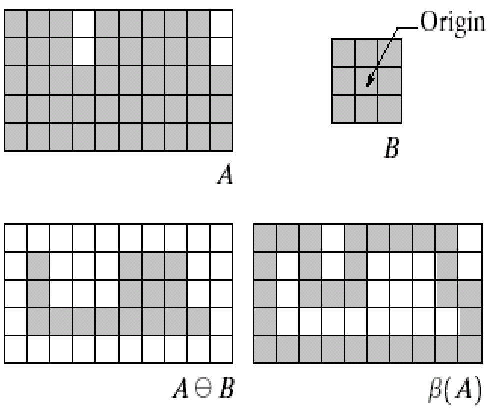



**FIGURE 9.10** Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The dark dot is the center of the structuring element.

# Boundary Extraction

$$\beta(A) = A - (A \ominus B)$$




a b
c d

**FIGURE 9.13** (a) Set $A$. (b) Structuring element $B$. (c) $A$ eroded by $B$. (d) Boundary, given by the set difference between $A$ and its erosion.

# Boundary Extraction

a b

**FIGURE 9.14** (a) A simple binary image, with 1's represented in white. (b) Result of using Eq. (9.5-1) with the structuring element in Fig. 9.13(b).

# Region Filling

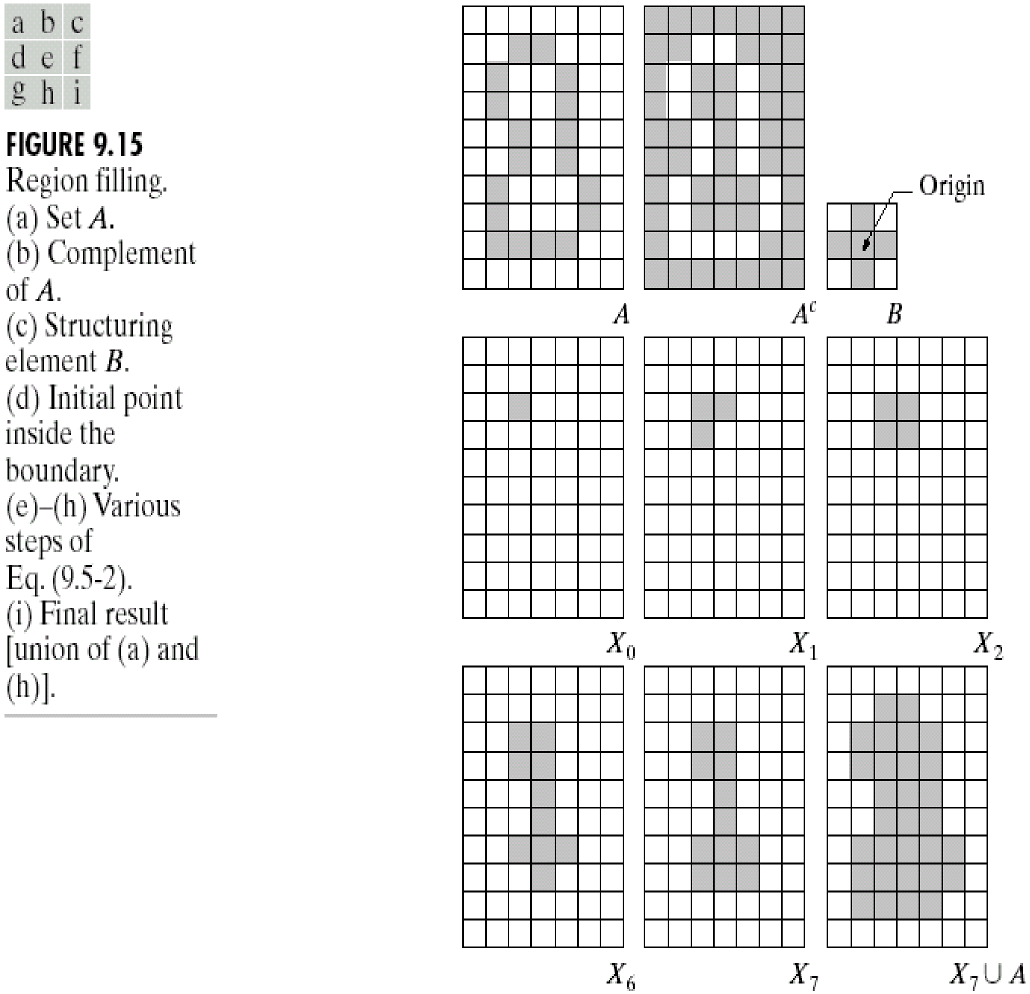


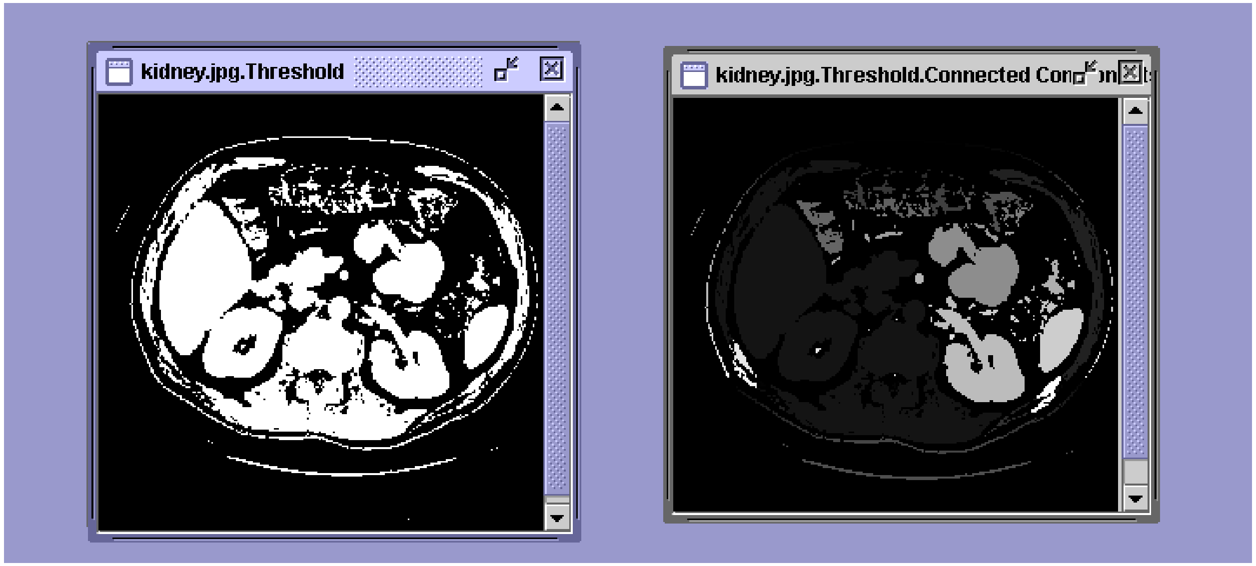

**FIGURE 9.15** Region filling. (a) Set $A$. (b) Complement of $A$. (c) Structuring element $B$. (d) Initial point inside the boundary. (e)–(h) Various steps of Eq. (9.5-2). (i) Final result [union of (a) and (h)].

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1,2,3....$$

# Connected Component 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.



# Extraction of Connected Components

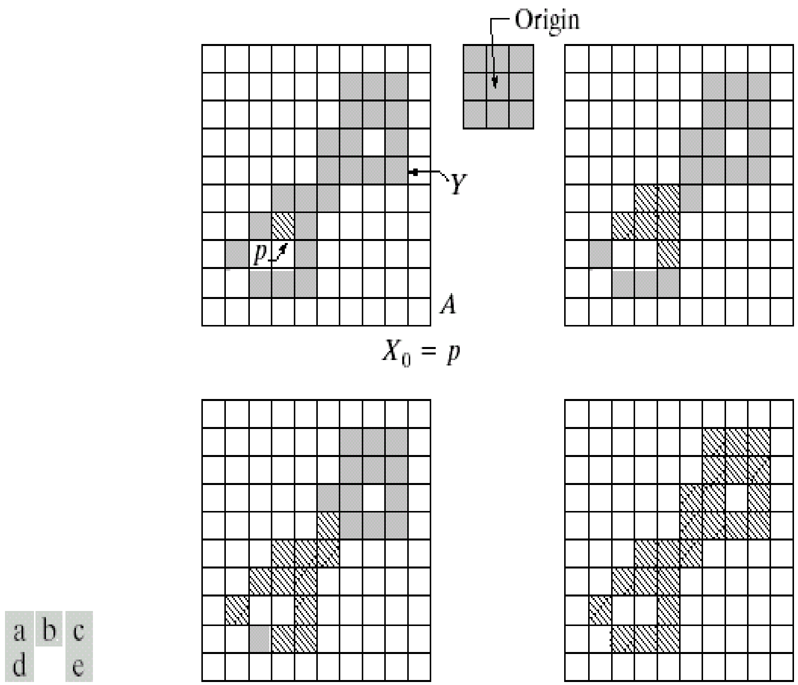
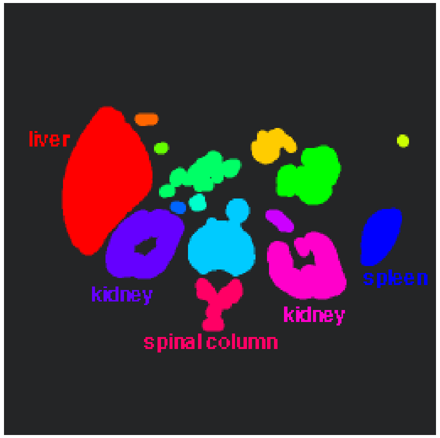


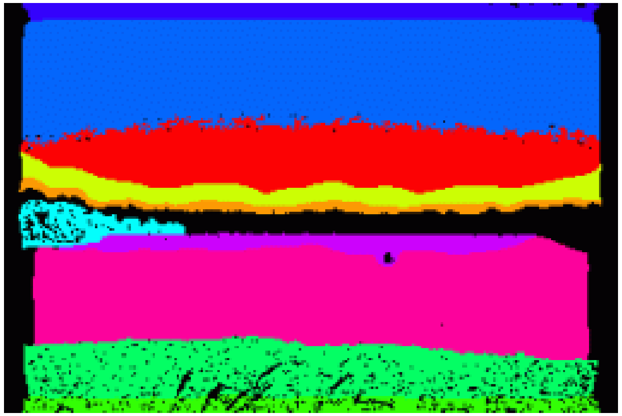

**FIGURE 9.17** (a) Set $A$ showing initial point $p$ (all shaded points are valued 1, but are shown different from $p$ to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1,2,3,\ldots$$

# Labeling shown as pseudo-color




connected components of 1's from thresholded image



connected components of cluster labels

# Connectivity

| | | |
|---|---|---|
| | N | |
| W | * | E |
| | S | |

4-neighborhood

| | | |
|---|---|---|
| NW | N | NE |
| W | * | E |
| SW | S | SE |

8-neighborhood

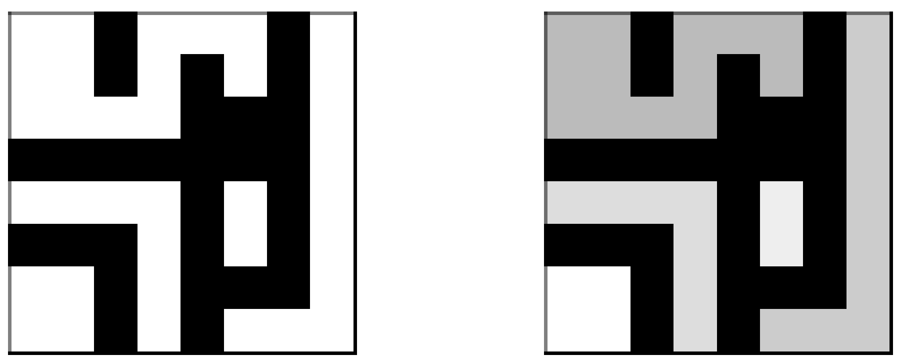
# Recursive labeling

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

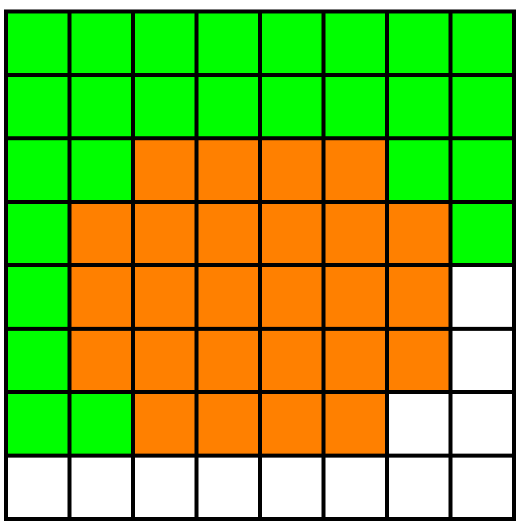
b) connected components labeling



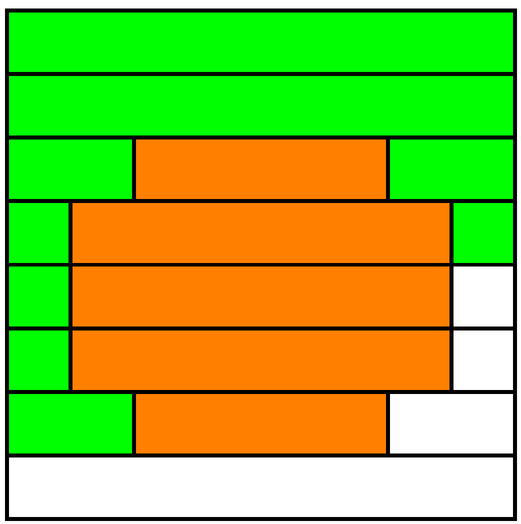
c) binary image and labeling, expanded for viewing

# First Step : Run Length Encoding

- Segment each image row into groups of similar pixels called *runs*
  - Runs store a start and end point for each contiguous row of color

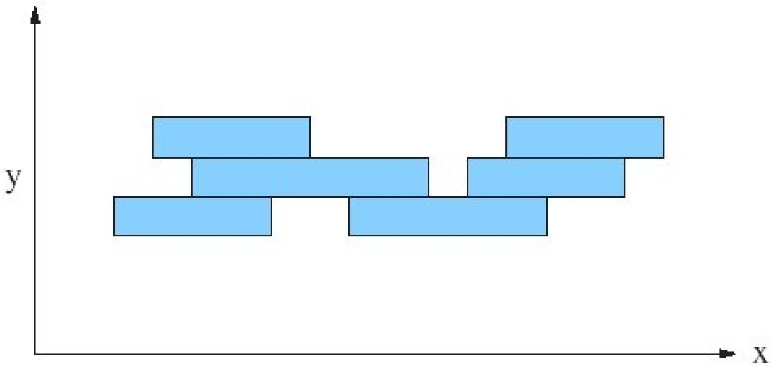


Original image

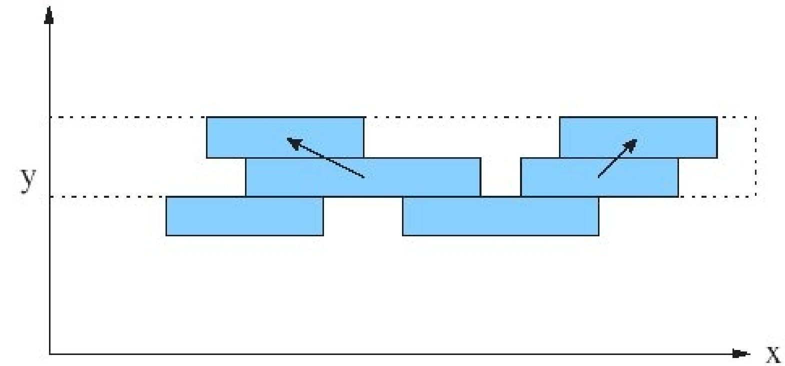


RLE image

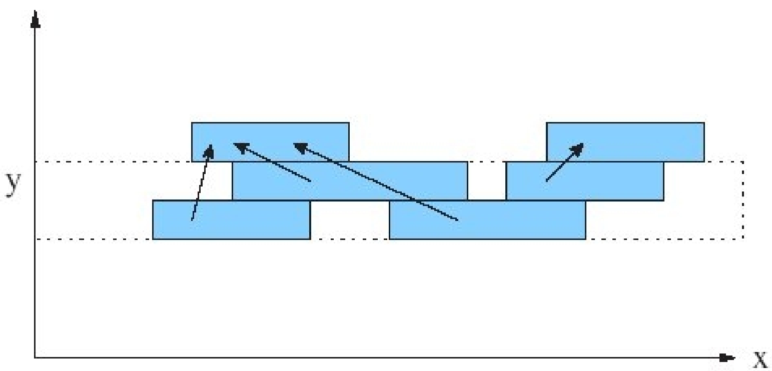
# Second Step : Merging Regions



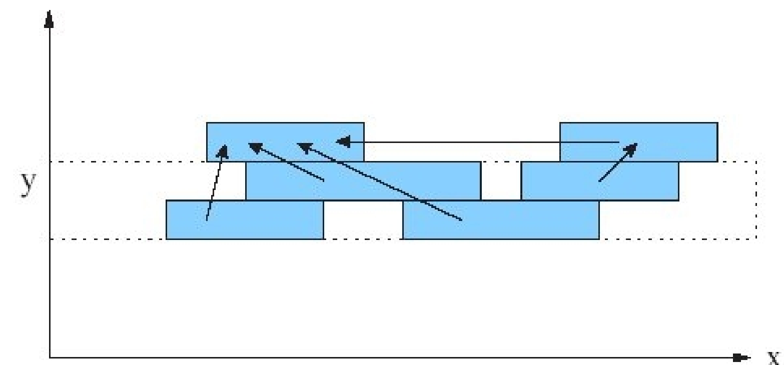

1: Runs start as a fully disjoint forest




2: Scanning adjacent lines, neighbors are merged




3: New parent assignments are to the furthest parent

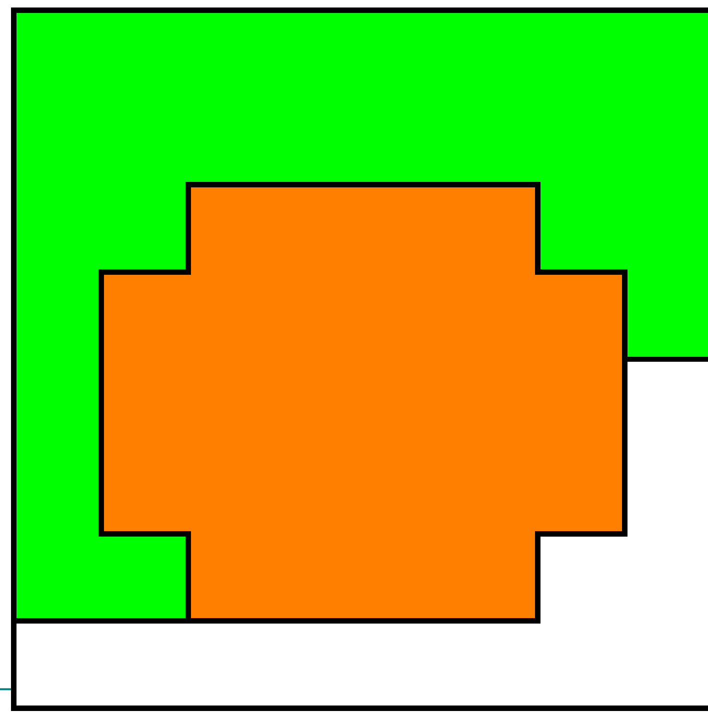



4: If overlap is detected, latter parent is updated

# Final Results

Runs are merged into multi-row regions

Image is now described as contiguous regions instead of just pixels

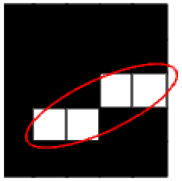


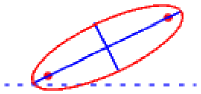
# Blob Properties

Now that we have nice, clean blobs, what can we do with them?

Compute Statistics:

- Area
- Perimeter
- Aspect ratio
- Center of mass
- best-fitting ellipse
- Average color
- Etc.





All this can be used to classify blobs and decide if they hold the objects we are interested in.

# Area an 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 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):**

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

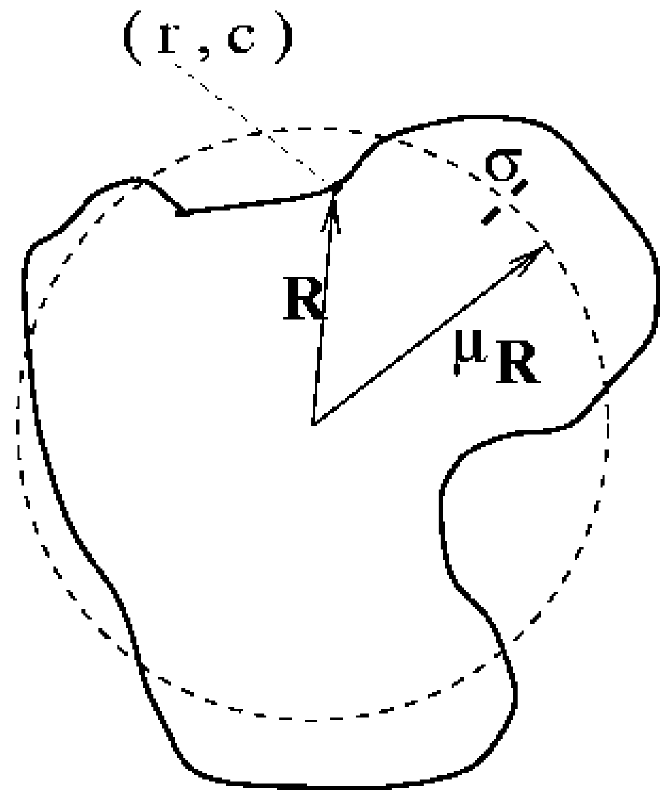
where $\mu_R$ and ${\sigma_R}^2$ are the mean and variance of the distance from the centroid of the shape to the boundary pixels $(r_k, c_k)$.

**mean radial distance:**

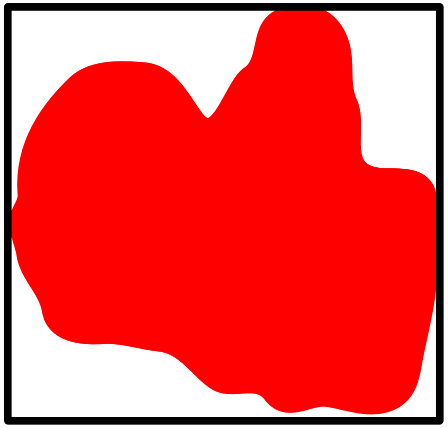
$$\mu_R = \frac{1}{K}\sum_{k=0}^{K-1} \|(r_k, c_k) - (\bar{r}, \bar{c})\|$$

**variance of radial distance:**

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



# Bounding box



# 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:**

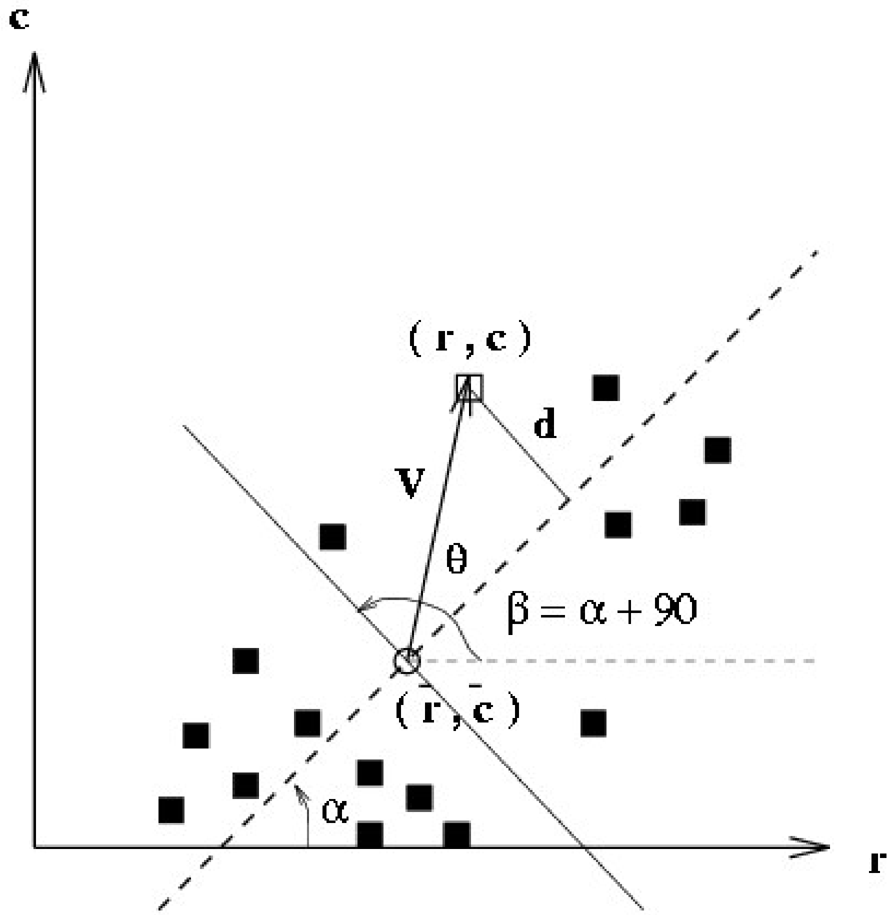
$$\mu_{cc} = \frac{1}{A} \sum_{(r,c) \in R} (c - \bar{c})^2$$

These are invariant to object location in the image.

# Axis of least inertia

An axis which we could spin the pixels with least energy input

The axis about which the pixels have least second moment



# 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.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 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 |