CS534 Introduction to Computer Vision
Binary Image Analysis

Ahmed Elgammal
Dept. of Computer Science
Rutgers University

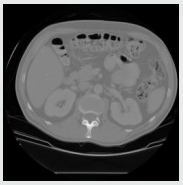
Outlines

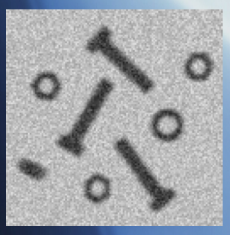
- A Simple Machine Vision System
- Image segmentation by thresholding
- Digital geometry
- Connected components
- Mathematical morphology
- Region descriptors
- Limitations
- Sources:
  - Burger and Burge “Digital Image Processing” Chapter 11
  - L. G. Shapiro and G. C. Stockman “Computer Vision”, Prentice Hall 2001.
  - Forsyth and Ponce “Computer Vision a Modern approach”
  - Slides by Prof. L. Davis @ UMD and G.C. Stockman @MSU

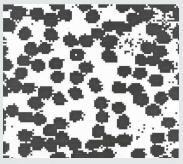
Problem definition

Objective: Design a vision system to "see" a "flat" world

- Page of text: text, line drawings, etc.
- Side panel of a truck
- Objects on an inspection belt
- X-ray image of separated potatoes
- Microscopic image of blood cells
- etc.



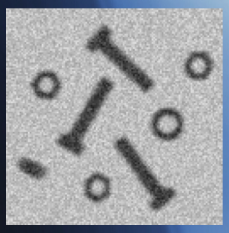




- Binary Images can also be outcome of pixel classifiers, e.g. skin detector

Problem definition

- General approach to recognition/inspection
  - Acquire gray scale image using camera
  - Reduce to black and white image - black objects on white background
  - Find individual black objects and measure their properties
  - Compare those properties to object models



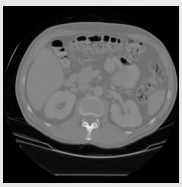
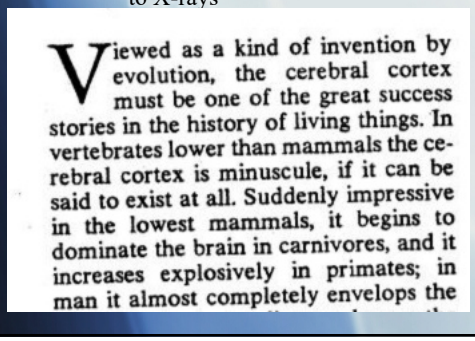



Image segmentation

- How do we know which groups of pixels in a digital image correspond to the objects to be analyzed?
  - objects may be uniformly darker or brighter than the background against which they appear
    - black characters imaged against the white background of a page
    - bright, dense potatoes imaged against a background that is transparent to X-rays


Viewed as a kind of invention by evolution, the cerebral cortex must be one of the great success stories in the history of living things. In vertebrates lower than mammals the cerebral cortex is minuscule, if it can be said to exist at all. Suddenly impressive in the lowest mammals, it begins to dominate the brain in carnivores, and it increases explosively in primates; in man it almost completely envelops the

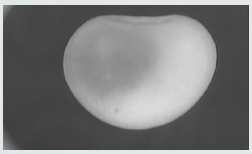


Image segmentation

- Ideally, object pixels would be black (0 intensity) and background pixels white (maximum intensity)
- But this rarely happens
  - pixels overlap regions from both the object and the background, yielding intensities between pure black and white - edge blur
  - cameras introduce "noise" during imaging - measurement "noise"
  - e.g., potatoes have non-uniform "thickness", giving variations in brightness in X-ray - model "noise"

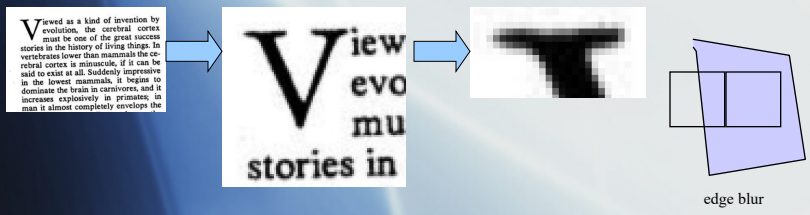
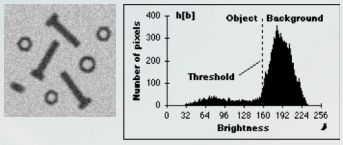


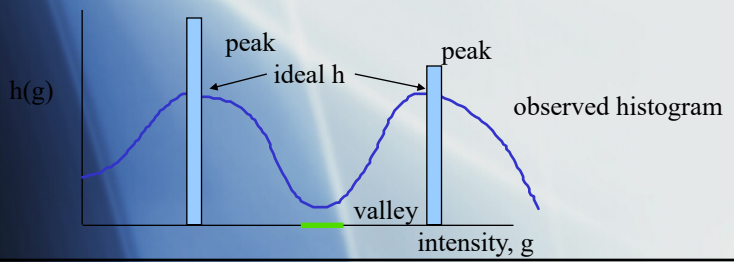
Image segmentation by thresholding

- But if the objects and background occupy different ranges of gray levels, we can "mark" the object pixels by a process called **thresholding:**
  - Let I(u,v) be the original, gray level image
  - B(u,v) is a binary image (pixels are either 0 or 1) created by thresholding I(u,v)
    - B(u,v) = 1 if I(u,v) < t
    - B(u,v) = 0 if I(u,v) >= t
    - We will assume that the 1's are the object pixels and the 0's are the background pixels
- Thresholding is a point operation
- How to choose the threshold t ?

Thresholding



- How do we choose the threshold t?
- Histogram (h) - gray level frequency distribution of the gray level image I.
  - $h_I(g)$ = number of pixels in F whose gray level is g
  - $H_I(g)$ = number of pixels in F whose gray level is <=g (cumulative)

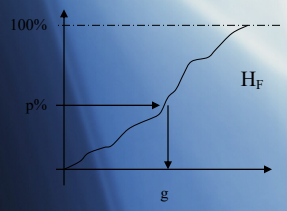


Thresholding

- Peak and valley method
  - Find the two most prominent peaks of h
    - g is a peak if $h(g) > h(g \pm \Delta g)$, $\Delta g = 1, ..., k$
  - Let g_1 and g_2 be the two highest peaks, with $g_1 < g_2$
  - Find the deepest valley, g, between g_1 and g_2
    - g is the valley if $h_F(g) <= h_F(g')$, g,g′ in [g1, g2]
  - Use g as the threshold
  - Problem: histograms are not smooth. How can we smooth a histogram?

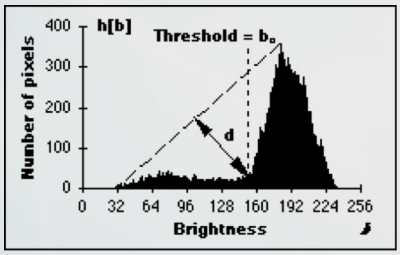
Thresholding

- P-tile method
  - in some applications we know approximately what percentage, p, of the pixels in the image come from objects
    - might have one potato in the image, or one character.
  - Cumulative histogram $H_I(g)$ can be used to find the gray level, g, such that ~p% of the pixels have intensity <= g
  - Then, we can examine h_I in the neighborhood of g to find a good threshold (low valley point)



Triangle algorithm

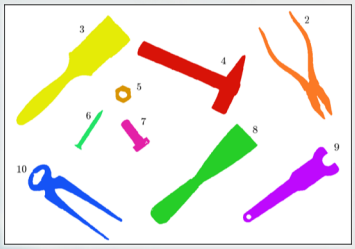
- A line is constructed between the maximum of the histogram at brightness b_{max} and the lowest value b_{min} = (p=0)% in the image.
- The distance d between the line and the histogram h[b] is computed for all values of b from b = b_{min} to b = b_{max}.
- The brightness value b_o where the distance between h[b_o] and the line is maximal is the threshold value.
- This technique is particularly effective when the object pixels produce a weak peak in the histogram.



Thresholding

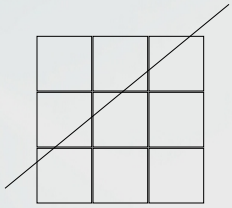
- Hand selection
  - Arbitrary select a threshold by hand!
- Many threshold selection methods in the literature
  - Probabilistic methods
    - make parametric assumptions about object and background intensity distributions and then derive “optimal” thresholds
  - Structural methods
    - Evaluate a range of thresholds wrt properties of resulting binary images
      - one with straightest edges, maximum contrast, most easily recognized objects, etc.
  - Local thresholding
    - apply thresholding methods to image windows

- What’s next after thresholding
- We need to detect separate regions in the binary image.
- This process is called:
  - Region labeling
  - Connected component labeling



Elements of Digital Geometry

- Emerged with the rise of computer technologies in the second half of the 20th century
- Pioneers: A. Rosenfeld, J. L. Pfaltz
- Fundamental for computer graphics and digital image processing
- Deals with sets of grid points
- Mathematical roots: graph theory and discrete topology.
- Three Interrelated Areas:
  - Digital Topology: concepts of open and closed sets, connectedness, digital Jordan curve theorem, Euler Numbers,
  - Digital Geometry: the study of geometric properties of sets of lattice points produced by digitizing regions or curves in the plane: digitization, digital convexity, digital straightness.
  - Graph theory and combinatorics of digital spaces: regions on the digital grid as a undirected graphs

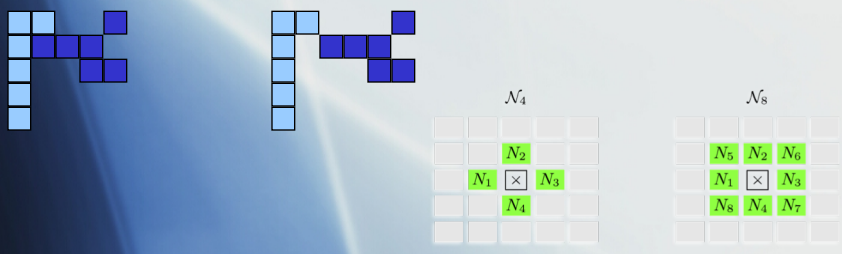


Neighborhood and Connectivity

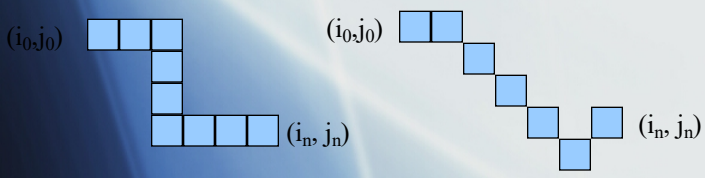
- Definition: Given a pixel (i,j) its 4-neighbors are the points (i',j') such that |i-i'| + |j-j'| = 1
  - the 4-neighbors are (i±1, j) and (i,j±1)
- Definition: Given a pixel (i,j) its 8-neighbors are the points (i',j') such that max(|i-i'|,|j-j'|) = 1
  - the 8- neighbors are (i, j±1), (i±1, j) and (i±1, j±1)

Connectivity

- 4-connectivity vs. 8-connectivity:
- Definition: Given two disjoint sets of pixels, A and B, A is 4(8)-connected to B if there is a pixel in A that is a 4(8) neighbor of a pixel in B
- So, we have to chose 4 or 8 neighborhood notion to use.



- Definition: A 4-(8)path from pixel (i_0,j_0) to (i_n,j_n) is a sequence of pixels (i_0,j_0) (i_1,j_1) (i_2,j_2) , ... (i_n,j_n) such that (i_k, j_k) is a 4-(8) neighbor of (i_{k+1}, j_{k+1}), for k = 0, ..., n-1



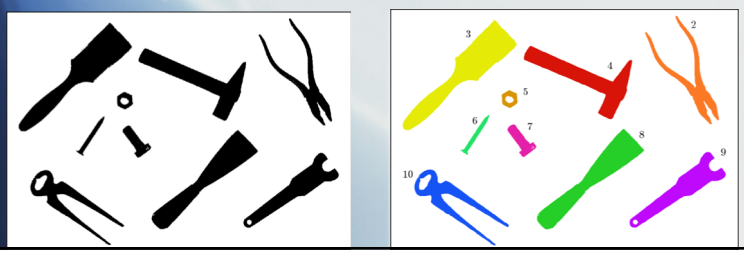
Every 4-path is an 8-path!

Connected-to relation

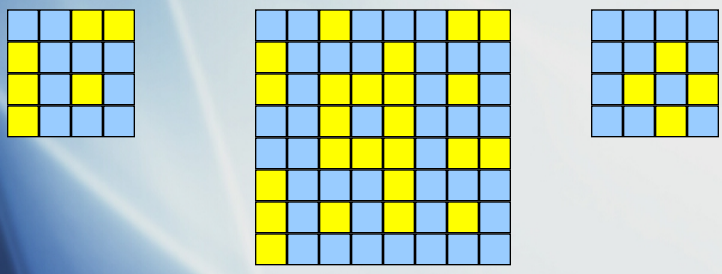
- Definition: Given a binary image, B, the set of all 1's is called the foreground and is denoted by S
- Definition: Given a pixel p in S, p is 4(8) connected to q in S if there is a path from p to q consisting only of points from S using 4(8) neighbors.
- The relation "is-connected-to" is an equivalence relation
  - Reflexive - p is connected to itself by a path of length 0
  - Symmetric - if p is connected to q, then q is connected to p by the reverse path
  - Transitive - if p is connected to q and q is connected to r, then p is connected to r by concatenation of the paths from p to q and q to r

Connected components

- Since the "is-connected-to" relation is an equivalence relation, it partitions the set S into a set of equivalence classes or components
  - these are called connected components, or connected regions
- Definition: S' is the complement of S - it is the set of all pixels in B whose value is 0
  - S' can also be partitioned into a set of connected components
  - Regard the image as being surrounded by a frame of 0's
  - The component(s) of S' that are adjacent to this frame is called the background of B.
  - All other components of S' are called holes



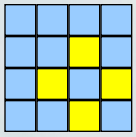
Examples - Yellow = 1, Green = 0



How many 4- (8) components of S?
What is the background?
Which are the 4- (8) holes?

Background and foreground connectivity

- Use opposite connectivity for the foreground and the background
  - 4-foreground, 8-background: 4 single pixel objects and no holes
  - 4-background, 8-foreground: one 4 pixel object containing a 1 pixel hole



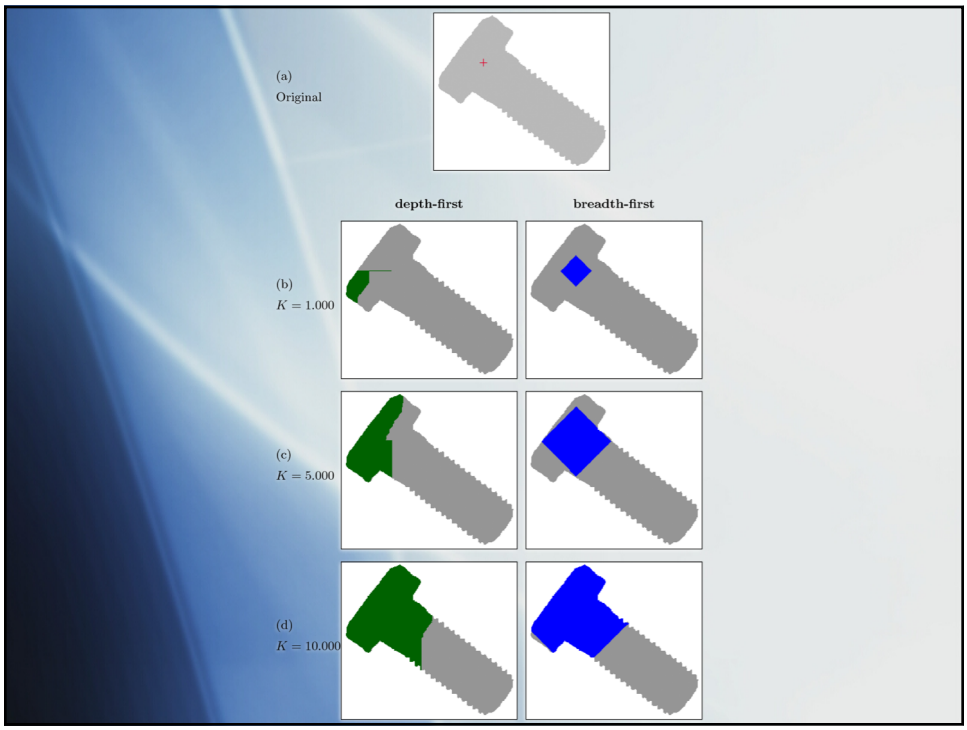
FloodFill region labeling

```
1:  RegionLabeling(I)
        I: binary image (0 = background, 1 = foreground)
        The image I is labeled (destructively modified) and returned.
2:      Initialize m ← 2 (the value of the next label to be assigned).
3:      Iterate over all image coordinates (u, v).
4:          if I(u, v) = 1 then
5:              FloodFill(I, u, v, m)        ▷ use any of the 3 versions below
6:              m ← m + 1.
7:      return the labeled image I.

8:  FloodFill(I, u, v, label)                ▷ Recursive Version
9:      if coordinate (u, v) is within image boundaries and I(u, v) = 1 then
10:         Set I(u, v) ← label
11:         FloodFill(I, u+1, v, label)
12:         FloodFill(I, u, v+1, label)
13:         FloodFill(I, u, v−1, label)
14:         FloodFill(I, u−1, v, label)
15:     return.

16: FloodFill(I, u, v, label)                ▷ Depth-First Version
17:     Create an empty stack S
```


```
1:  RegionLabeling(I)
        I: binary image (0 = background, 1 = foreground)
        The image I is labeled (destructively modified) and returned.
2:      Initialize m ← 2 (the value of the next label to be assigned).
3:      Iterate over all image coordinates (u, v).
4:          if I(u, v) = 1 then
5:              FloodFill(I, u, v, m)        ▷ use any of the 3 versions below
6:              m ← m + 1.
7:      return the labeled image I.

8:  FloodFill(I, u, v, label)                ▷ Recursive Version
9:      if coordinate (u, v) is within image boundaries and I(u, v) = 1 then
10:         Set I(u, v) ← label
11:         FloodFill(I, u+1, v, label)
12:         FloodFill(I, u, v+1, label)
13:         FloodFill(I, u, v−1, label)
14:         FloodFill(I, u−1, v, label)
15:     return.

16: FloodFill(I, u, v, label)                ▷ Depth-First Version
17:     Create an empty stack S
18:     Put the seed coordinate ⟨u, v⟩ onto the stack: Push(S, ⟨u, v⟩)
19:     while S is not empty do
20:         Get the next coordinate from the top of the stack:
                ⟨x, y⟩ ← Pop(S)
21:         if coordinate (x, y) is within image boundaries and I(x, y) = 1
                then
22:             Set I(x, y) ← label
23:             Push(S, ⟨x+1, y⟩)
24:             Push(S, ⟨x, y+1⟩)
25:             Push(S, ⟨x, y−1⟩)
26:             Push(S, ⟨x−1, y⟩)
27:     return.

28: FloodFill(I, u, v, label)                ▷ Breadth-First Version
29:     Create an empty queue Q
30:     Insert the seed coordinate ⟨u, v⟩ into the queue: Enqueue(Q, ⟨u, v⟩)
31:     while Q is not empty do
32:         Get the next coordinate from the front of the queue:
                ⟨x, y⟩ ← Dequeue(Q)
33:         if coordinate ⟨x, y⟩ is within image boundaries and I(x, y) = 1
                then
34:             Set I(x, y) ← label
35:             Enqueue(Q, ⟨x+1, y⟩)
36:             Enqueue(Q, ⟨x, y+1⟩)
37:             Enqueue(Q, ⟨x, y−1⟩)
38:             Enqueue(Q, ⟨x−1, y⟩)
39:     return.
```

Connected Componenet Labeling

- Flood Fill like algorithms, breadth first, depth first
- Sequential Region Labeling: More efficient two-pass algorithm for region labeling



(a)

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

0 Background

1 Foreground

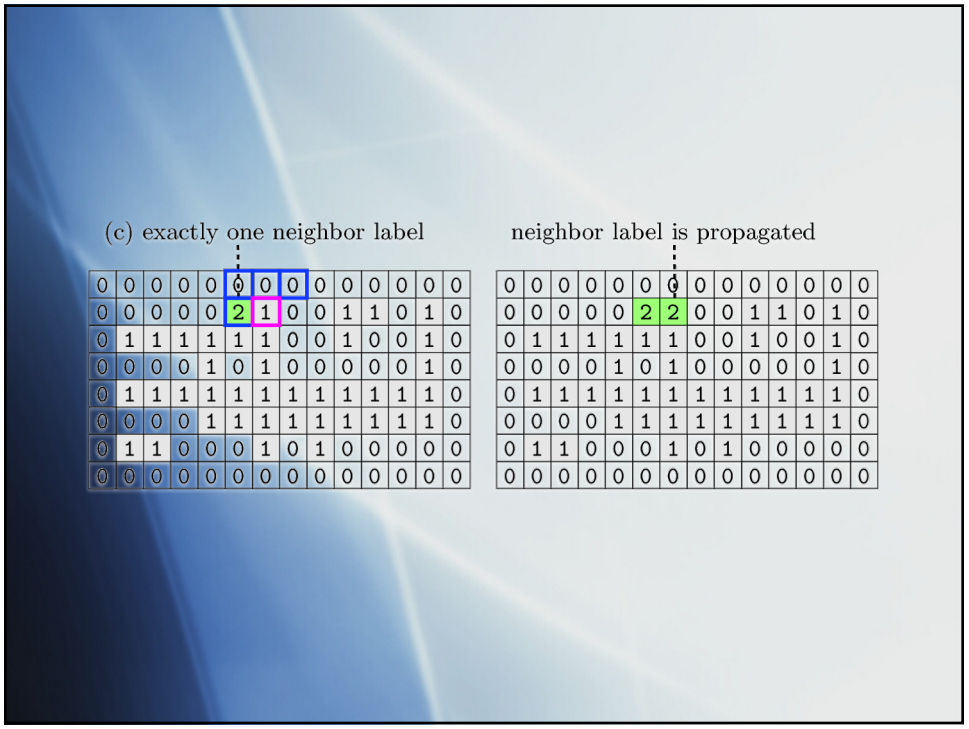
$\mathcal{N}_4(u,v) =$ N_2 / N_1 $\times$; or $\mathcal{N}_8(u,v) =$ N_2 N_3 N_4 / N_1 $\times$

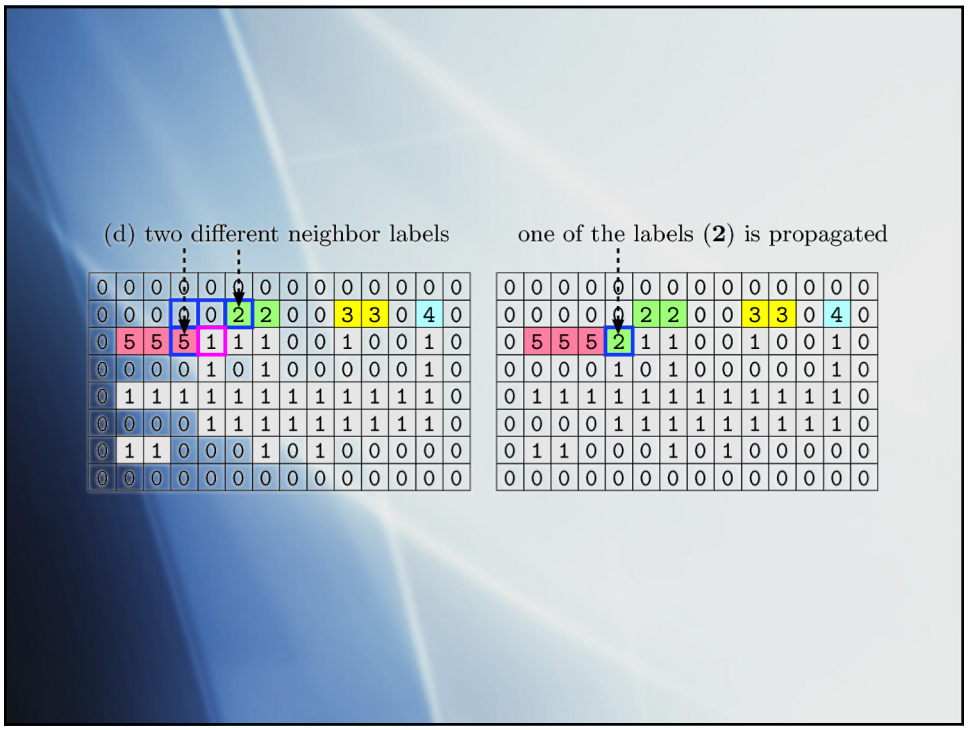
(b) only background neighbors

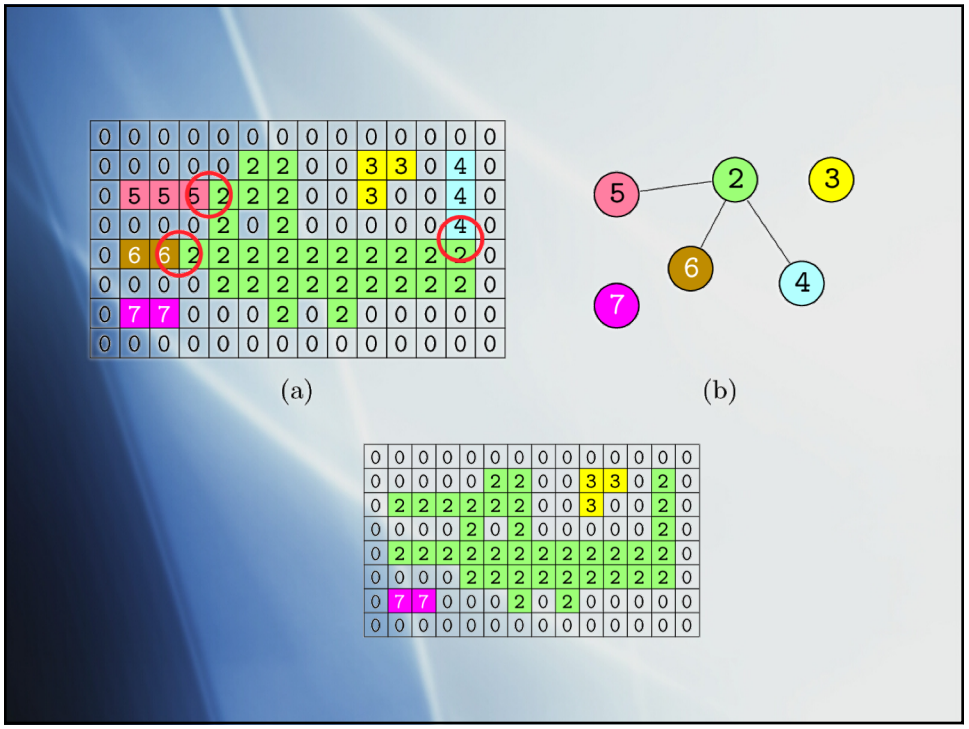
new label (**2**)

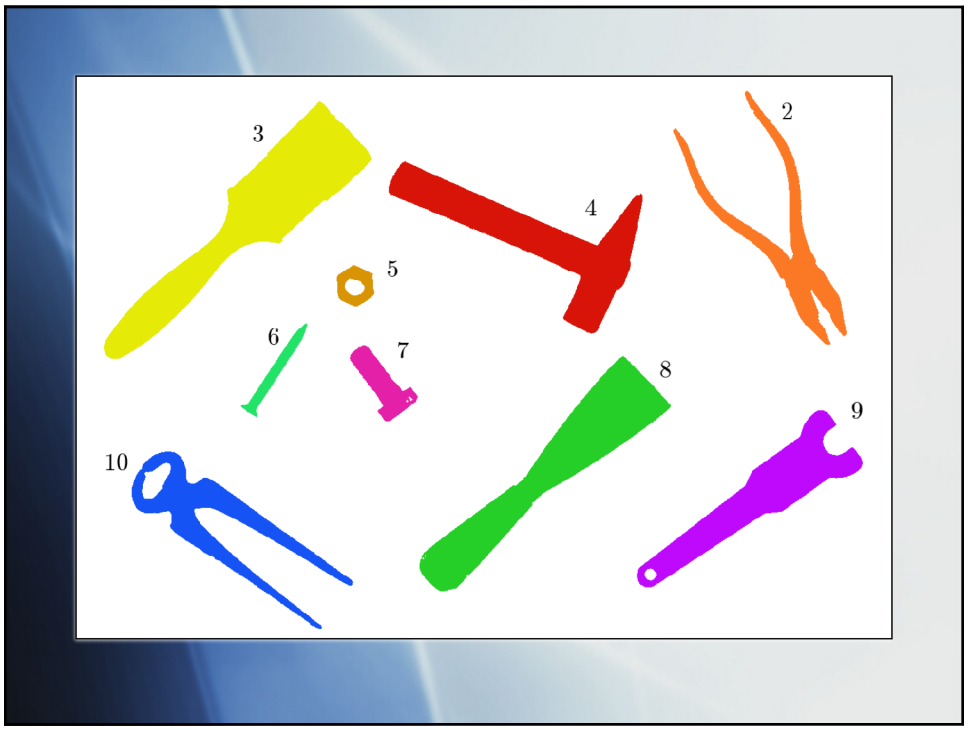
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	1	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

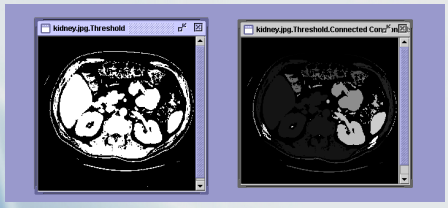

(c) exactly one neighbor label
neighbor label is propagated


(d) two different neighbor labels
one of the labels (2) is propagated


(a)
(b)


3
4
2
5
6
7
8
9
10

- Things might not be perfect!
- Different regions might be connected due to noise or wrong threshold.
- What to do about that?



Mathematical Morphology

Binary mathematical morphology consists of two basic operations

dilation and erosion

and several composite relations

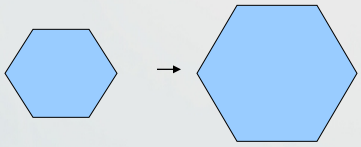
closing and opening
conditional dilation
. . .

Dilation

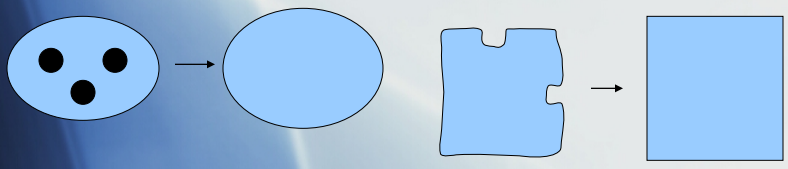
Dilation expands the connected sets of 1s of a binary image.

It can be used for

1. growing features



2. filling holes and gaps

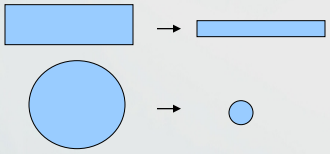


Erosion

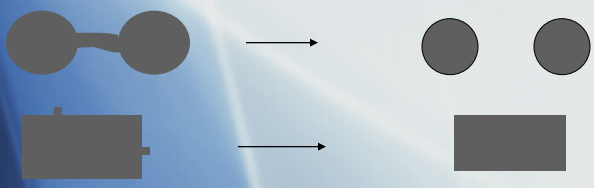
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for

1. shrinking features



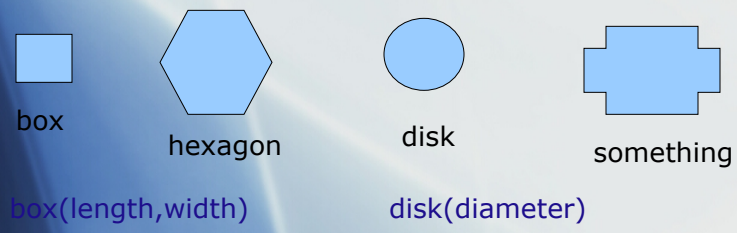
2. Removing bridges, branches and small protrusions



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)

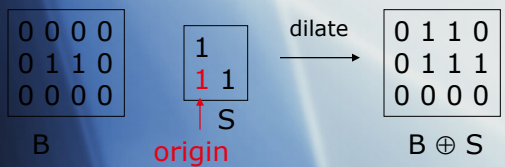
disk(diameter)

Dilation with Structuring Elements

The arguments to dilation and erosion are

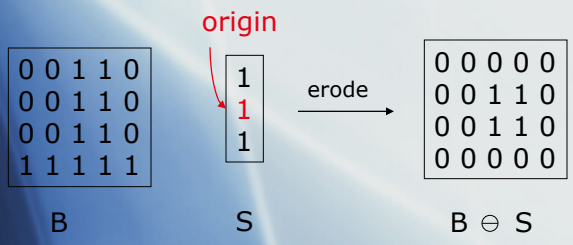
1. **a binary image B**
2. **a structuring element S**

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.

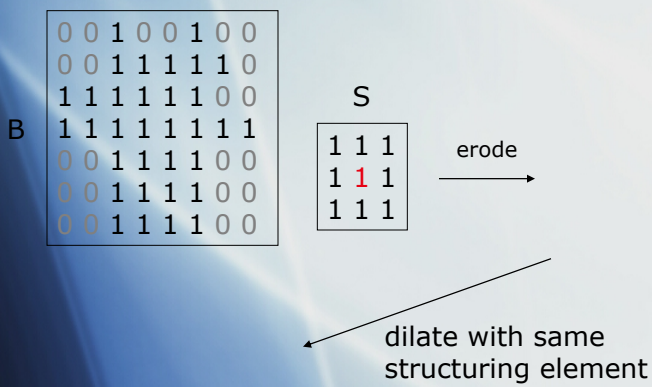


Erosion with Structuring Elements

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.



Example 1 to Try



Example 2 to 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
```

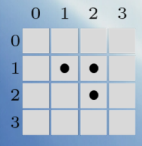
First erode and then dilate with the same S.

- Point Sets: For a binary image $I(u,v) \in \{0,1\}$

The corresponding point set consists of the coordinate pairs $\boldsymbol{p} = (u,v)$ for all foreground pixels,

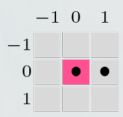
$$\mathcal{Q}_I = \{\boldsymbol{p} \mid I(\boldsymbol{p}) = 1\}$$

I



$I \equiv \mathcal{Q}_I = \{(1,1),(2,1),(2,2)\}$

H

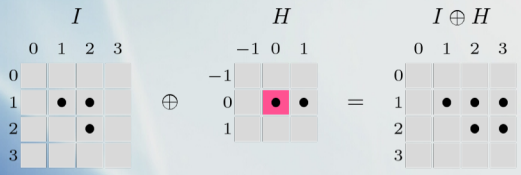


$H \equiv \mathcal{Q}_H = \{(0,0),(1,0)\}$

Dilation

- Definition

$$I \oplus H \equiv \{(\boldsymbol{p}+\boldsymbol{q}) \mid \text{for some } \boldsymbol{p} \in I \text{ and } \boldsymbol{q} \in H\}$$



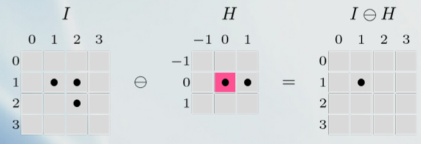
$I \equiv \{(1,1),(2,1),(2,2)\},\ H \equiv \{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{0})\}$

$$\begin{aligned} I \oplus H \equiv \{ &(1,1)+(\mathbf{0},\mathbf{0}),\ (1,1)+(\mathbf{1},\mathbf{0}),\\ &(2,1)+(\mathbf{0},\mathbf{0}),\ (2,1)+(\mathbf{1},\mathbf{0}),\\ &(2,2)+(\mathbf{0},\mathbf{0}),\ (2,2)+(\mathbf{1},\mathbf{0}) \} \end{aligned}$$

Erosion

- Definition

$$I \ominus H \equiv \{\boldsymbol{p} \in \mathbb{Z}^2 \mid (\boldsymbol{p}+\boldsymbol{q}) \in I,\ \text{for every } \boldsymbol{q} \in H\}$$



$I \equiv \{(1,1),(2,1),(2,2)\},\ H \equiv \{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{0})\}$

$I \ominus H \equiv \{(1,1)\}$ because
$(1,1)+(\mathbf{0},\mathbf{0}) = (1,1) \in I$ **and** $(1,1)+(\mathbf{1},\mathbf{0}) = (2,1) \in I$

- Dilation

$$I \oplus H = H \oplus I$$

$$(I_1 \oplus I_2) \oplus I_3 = I_1 \oplus (I_2 \oplus I_3)$$

- Erosion

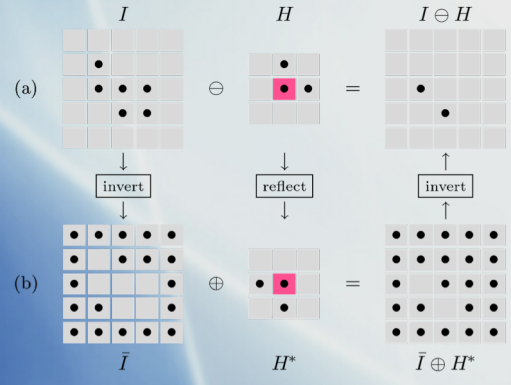
$$I \ominus H \neq H \ominus I$$

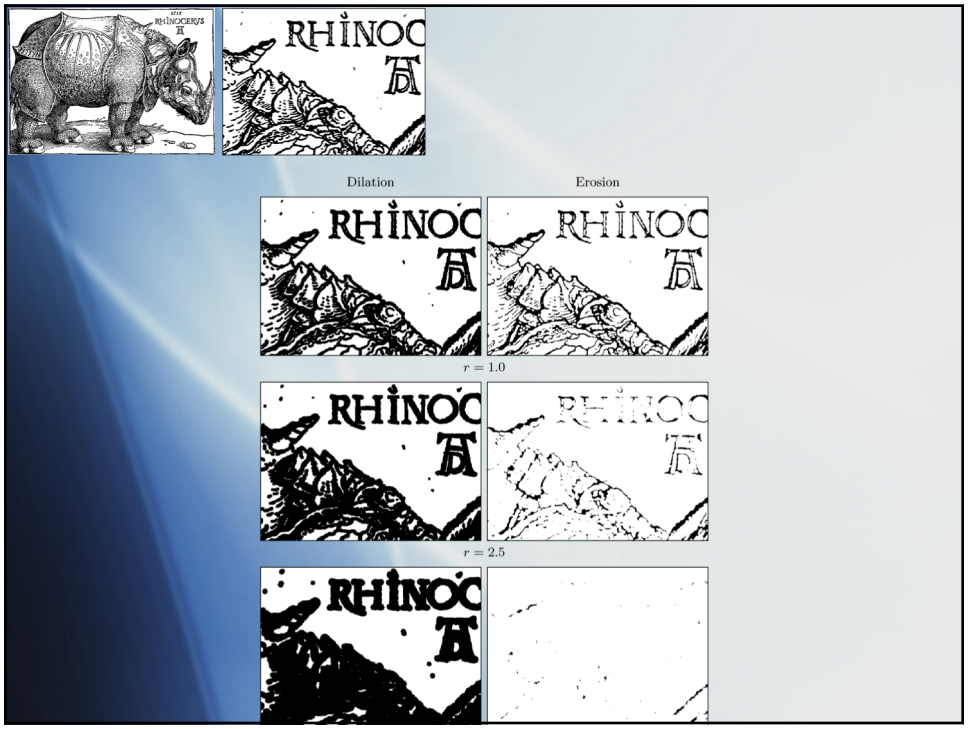
$$(I_1 \ominus I_2) \ominus I_3 = I_1 \ominus (I_2 \oplus I_3)$$

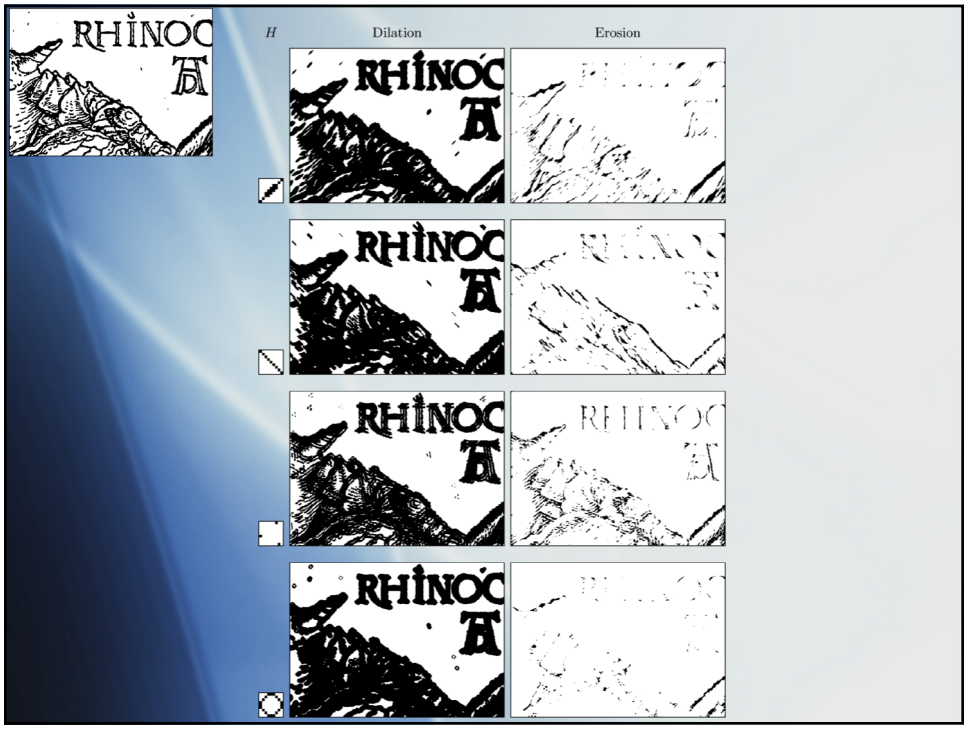
$$I \oplus \delta = \delta \oplus I = I, \qquad \text{with } \delta \equiv \{(0,0)\}$$

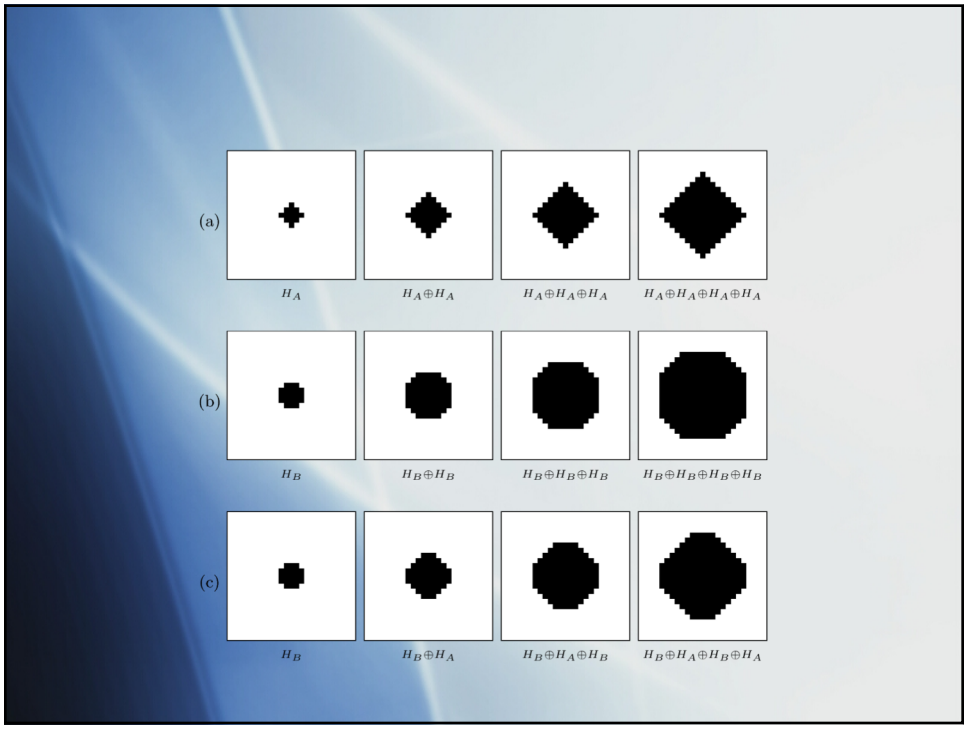
$$I \oplus H \equiv \overline{(\bar{I} \ominus H^*)}$$

$$I \ominus H \equiv \overline{(\bar{I} \oplus H^*)}$$




Dilation
Erosion
r = 1.0
r = 2.5


H
Dilation
Erosion



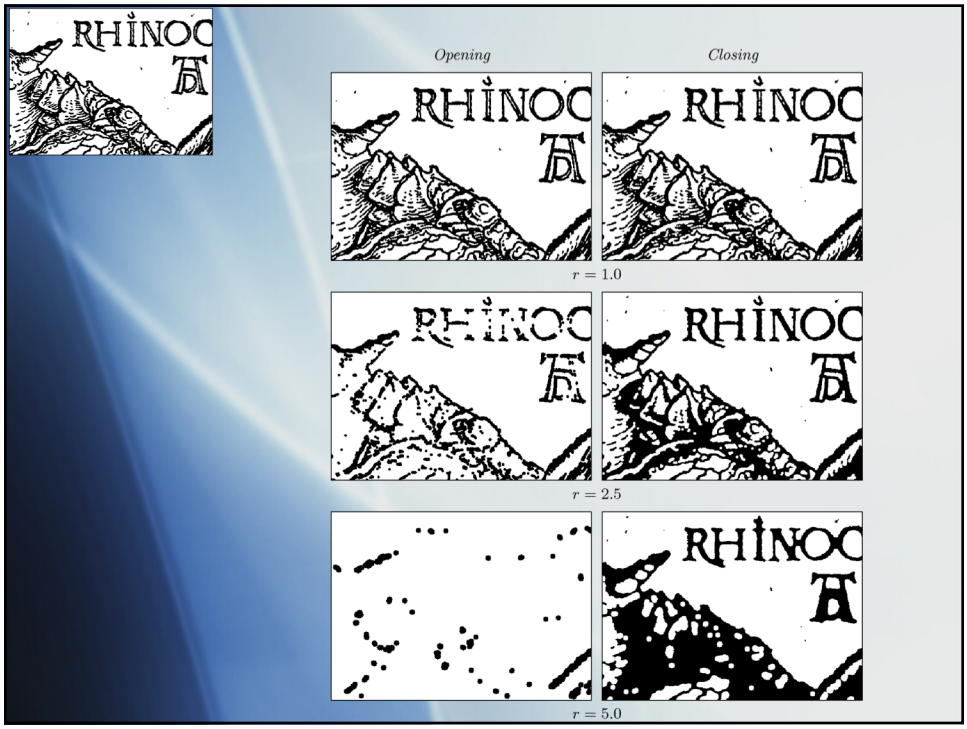
Opening and Closing

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

$$B \bullet S = (B \oplus S) - S$$

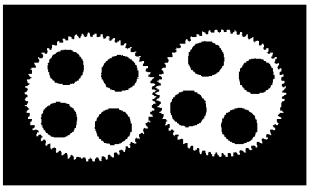
- Opening is the compound operation of erosion followed by dilation (with the same structuring element)

$$B \circ S = (B - S) \oplus S$$


Opening
Closing
$r = 1.0$
$r = 2.5$
$r = 5.0$

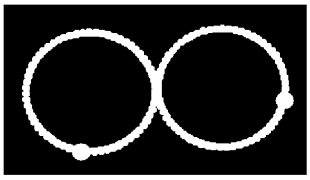
Gear Tooth Inspection

Sternberg 1985: use morphology to detect missing or broken watch gear teeth

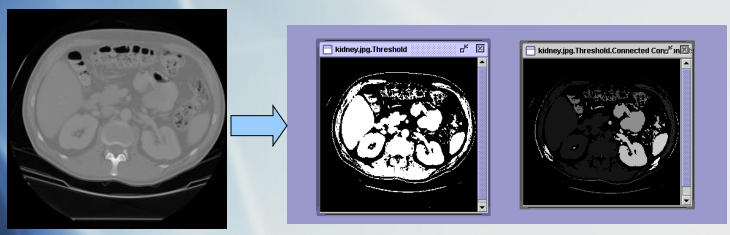


Original binary image

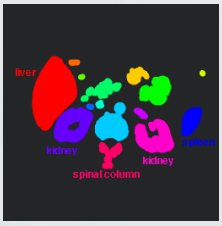
How did they do it?



Detected defects



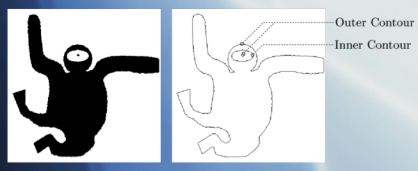
Connected Components after thresholding

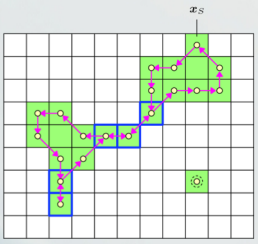


Connected Components after Morphology

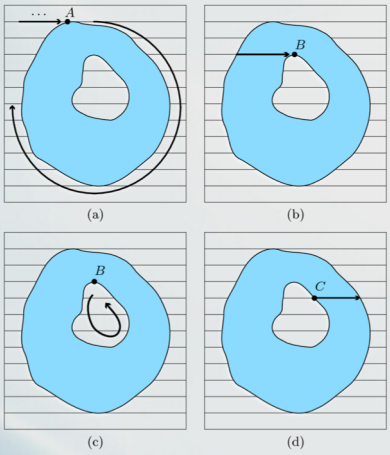
Region Contours

- Tracing a region contour is very useful to extract a descriptor of the region.
- Difficulties:
  - External and internal contours
  - The contour might run through the same pixel more than once

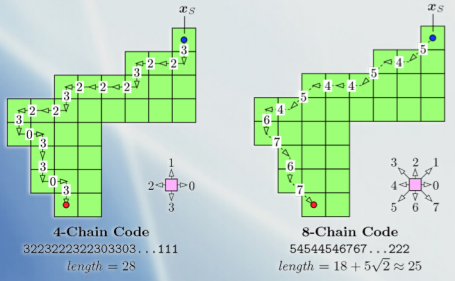




- Contour finding can be done after region labeling
- Algorithms for both region labeling and contour finding: when you first hit a contour, trace it.

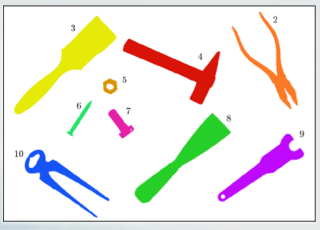


- Chain Codes: (Freeman codes) a method for contour encoding
- Begin at a given point and represent the contour by a sequence of discrete directional changes
  - Absolute chain code $\boldsymbol{c}'_{\mathcal{R}} = [5, 4, 5, 4, 4, 5, 4, 6, 7, 6, 7, \ldots 2, 2, 2]$
  - Differential chain code $\boldsymbol{c}''_{\mathcal{R}} = [7, 1, 7, 0, 1, 7, 2, 1, 7, 1, 1, \ldots 0, 0, 3]$



Properties of Binary Regions

- Our goal is to recognize each connected component as one of a set of known objects
  - letters of the alphabet
  - good potatoes versus bad potatoes
- We need to associate measurements, or properties, with each connected component that we can compare against expected properties of different object types.
- An important aspect is the invariance of the descriptor:
  - Invariant to translation
  - Invariant to scaling (size)
  - Invariant to rotation
  - Other invariants



Properties of Binary Regions

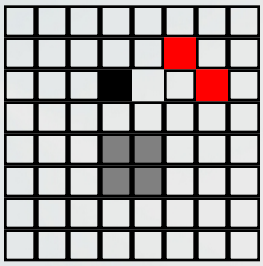
- Geometric Features
- Statistical Shape Properties
- Moment-based geometrical properties
- Projections
- ….

Geometric Features

- Perimeter
- Area
- Compactness and Roundness
- Bounding box
- Convex hull

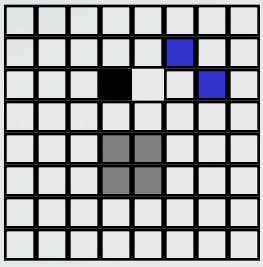
How do we compute the perimeter of a connected component?

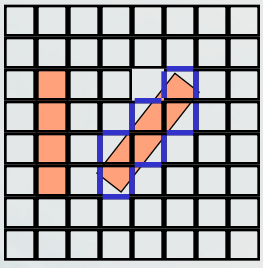
1. Count the number of pixels in the component adjacent to 0's
   - perimeter of black square would be 1
   - but perimeter of gray square, which has 4x the area, would be 4
   - but perimeter should go up as sqrt of area
2. Count the number of 0's adjacent to the component
   - works for the black and gray squares, but fails for the red dumbbell



How do we compute the perimeter of a connected component?

3) Count the number of sides of pixels in the component adjacent to 0's
   - these are the **cracks** between the pixels
   - clockwise traversal of these cracks is called a crack code
   - perimeter of black is 4, gray is 8 and blue is 8
- What effect does rotation have on the value of a perimeter of the digitization of a simple shape?
   - rotation can lead to large changes in the perimeter and the area!



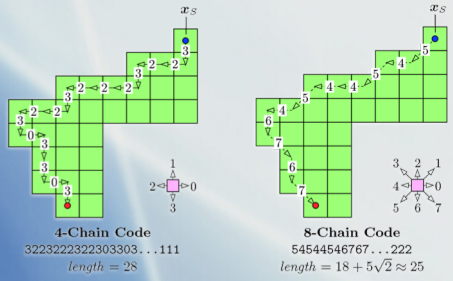


- Perimeter from a chain code

$$\text{Perimeter}(\mathcal{R}) = \sum_{i=0}^{M-1} \text{length}(c_i')$$

$$\text{with} \quad \text{length}(c) = \begin{cases} 1 & \text{for } c = 0, 2, 4, 6 \\ \sqrt{2} & \text{for } c = 1, 3, 5, 7 \end{cases}$$

$$P(\mathcal{R}) \approx \text{Perimeter}_{\text{corr}}(\mathcal{R}) = 0.95 \cdot \text{Perimeter}(\mathcal{R})$$



4-Chain Code
3223222322303303...111
length = 28

8-Chain Code
54544546767...222
length = $18 + 5\sqrt{2} \approx 25$

- Perimeter

$$\text{Perimeter}(\mathcal{R}) = \sum_{i=0}^{M-1} \text{length}(c_i')$$

$$\text{with} \quad \text{length}(c) = \begin{cases} 1 & \text{for } c = 0, 2, 4, 6 \\ \sqrt{2} & \text{for } c = 1, 3, 5, 7 \end{cases}$$

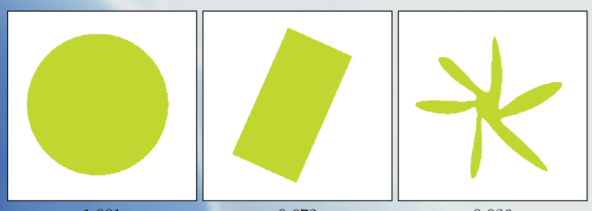
- Area

$A(\mathcal{R}) = |\mathcal{R}| = N.$

- Compactness and Roundness

$$\text{Circularity}(\mathcal{R}) = 4\pi \cdot \frac{A(\mathcal{R})}{P^2(\mathcal{R})}$$

$$\text{Circularity}(\mathcal{R}) \approx 4\pi \cdot \frac{A(\mathcal{R})}{\text{Perimeter}^2_{\text{corr}}(\mathcal{R})}$$

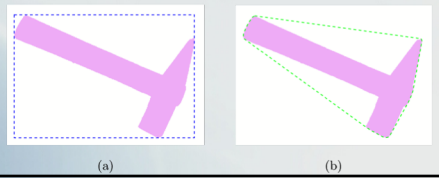


1.001 0.672 0.086

- Bounding box: minimal axis-parallel rectangle that encloses all the points of the region

$$\text{BoundingBox}(\mathcal{R}) = \langle u_{\min}, u_{\max}, v_{\min}, v_{\max} \rangle$$

- Convex hull: the smallest polygon within which the all points in the region fit.
  - Convexity: relationship between the length of the convex hull and the perimeter of the region
  - Density: the ratio between the area of the region and the area of the convex hull



Statistical shape properties

- Centroid $\bar{x} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} u$ and $\bar{y} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} v$
- Moments

$$m_{pq} = \sum_{(u,v)\in\mathcal{R}} I(u,v) \cdot u^p v^q$$

For binary images: $m_{pq} = \sum_{(u,v)\in\mathcal{R}} u^p v^q$

$$A(\mathcal{R}) = |\mathcal{R}| = \sum_{(u,v)\in\mathcal{R}} 1 = \sum_{(u,v)\in\mathcal{R}} u^0 v^0 = m_{00}(\mathcal{R})$$

$$\bar{x} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} u^1 v^0 = \frac{m_{10}(\mathcal{R})}{m_{00}(\mathcal{R})} \qquad \bar{y} = \frac{1}{|\mathcal{R}|} \cdot \sum_{(u,v)\in\mathcal{R}} u^0 v^1 = \frac{m_{01}(\mathcal{R})}{m_{00}(\mathcal{R})}$$

- Central Moments: position-independent (translation invariant) moments: shift the coordinate system to the centroid:

$$\mu_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} (u-\bar{x})^p \cdot (v-\bar{y})^q$$

- Normalized central moments: invariant to the size (scale) of the region

$$\bar{\mu}_{pq}(\mathcal{R}) = \mu_{pq} \cdot \left(\frac{1}{\mu_{00}(\mathcal{R})}\right)^{(p+q+2)/2}$$

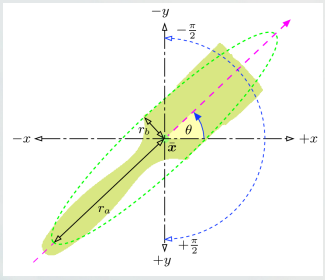
Geometric properties from moments

Orientation:

Direction of the major axis

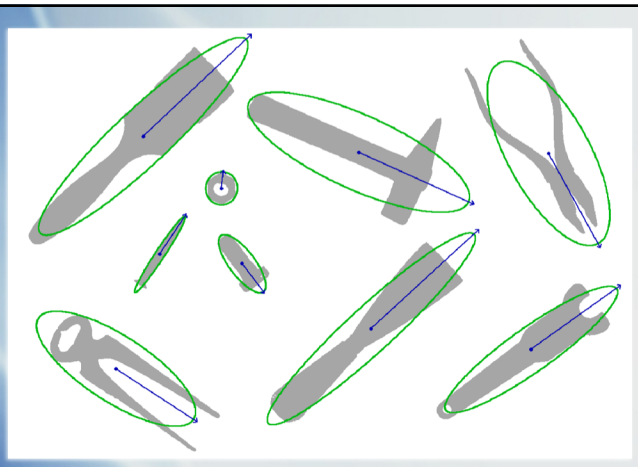
$$\tan(2\,\theta_{\mathcal{R}}) = \frac{2\cdot\mu_{11}(\mathcal{R})}{\mu_{20}(\mathcal{R}) - \mu_{02}(\mathcal{R})}$$

$$\theta_{\mathcal{R}} = \frac{1}{2}\tan^{-1}\left(\frac{2\cdot\mu_{11}(\mathcal{R})}{\mu_{20}(\mathcal{R}) - \mu_{02}(\mathcal{R})}\right)$$



Eccentricity: "elongatedness"

$$\text{Ecc}(\mathcal{R}) = \frac{a_1}{a_2} = \frac{\mu_{20}+\mu_{02}+\sqrt{(\mu_{20}-\mu_{02})^2+4\cdot\mu_{11}^2}}{\mu_{20}+\mu_{02}-\sqrt{(\mu_{20}-\mu_{02})^2+4\cdot\mu_{11}^2}}$$



$$\boldsymbol{A} = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

$$\text{Ecc}(\mathcal{R}) = \frac{a_1}{a_2} = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4 \cdot \mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4 \cdot \mu_{11}^2}}$$

$$r_a = 2 \cdot \left(\frac{\lambda_1}{|\mathcal{R}|}\right)^{\frac{1}{2}} = \left(\frac{2\,a_1}{|\mathcal{R}|}\right)^{\frac{1}{2}} \qquad r_b = 2 \cdot \left(\frac{\lambda_2}{|\mathcal{R}|}\right)^{\frac{1}{2}} = \left(\frac{2\,a_2}{|\mathcal{R}|}\right)^{\frac{1}{2}}$$

Invariant moments

- Hu's moments: seven combinations of the normalized centeral moments.
- Invariant to translation, scaling and orientation.

$$\begin{aligned}
H_1 &= \bar{\mu}_{20} + \bar{\mu}_{02} \\
H_2 &= (\bar{\mu}_{20} - \bar{\mu}_{02})^2 + 4\,\bar{\mu}_{11}^2 \\
H_3 &= (\bar{\mu}_{30} - 3\,\bar{\mu}_{12})^2 + (3\,\bar{\mu}_{21} - \bar{\mu}_{03})^2 \\
H_4 &= (\bar{\mu}_{30} + \bar{\mu}_{12})^2 + (\bar{\mu}_{21} + \bar{\mu}_{03})^2 \\
H_5 &= (\bar{\mu}_{30} - 3\,\bar{\mu}_{12}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + (3\,\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot \left[3\,(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
H_6 &= (\bar{\mu}_{20} - \bar{\mu}_{02}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + 4\,\bar{\mu}_{11} \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \\
H_7 &= (3\,\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3\,(\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + (3\,\bar{\mu}_{12} - \bar{\mu}_{30}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot \left[3\,(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right]
\end{aligned}$$

Projections

- Horizontal and vertical projection of the shapes

$$P_{\text{hor}}(v_0) = \sum_{u=0}^{M-1} I(u, v_0) \qquad \text{for } 0 < v_0 < N$$

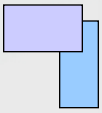
$$P_{\text{ver}}(u_0) = \sum_{v=0}^{N-1} I(u_0, v) \qquad \text{for } 0 < u_0 < M$$

Shortcomings of our machine vision system

- Object detection
  - thresholding will not extract intact objects in complex images
    - shading variations on object surfaces
    - texture
  - advanced segmentation methods
    - edge detection - locate boundaries between objects and background, between objects and objects
    - region analysis - find homogeneous regions; small combinations might correspond to objects.

Shortcomings of our machine vision system

- Occlusion
  - What if one object is partially hidden by another?
    - properties of the partially obscured, or occluded, object will not match the properties of the class model
  - Correlation - directly compare image of the "ideal" objects against real images
    - in correct overlap position, matching score will be high
  - Represent objects as collection of local features such as corners of a rectangular shape
    - locate the local features in the image
    - find combinations of local features that are configured consistently with objects



Shortcomings of our machine vision system

- Recognition of three dimensional objects
  - the shape of the image of a three dimensional object depends on the viewpoint from which it is seen
- Model a three dimensional object as a large collection of view-dependent models
- Model the three dimensional geometry of the object and mathematically relate it to its possible images
  - mathematical models of image geometry
  - mathematical models for recognizing three dimensional structures from two dimensional images

Shortcomings of our machine vision system

- Articulated objects
  - pliers
  - derricks
- Deformable objects
  - faces
  - jello
- Amorphous objects
  - fire
  - water