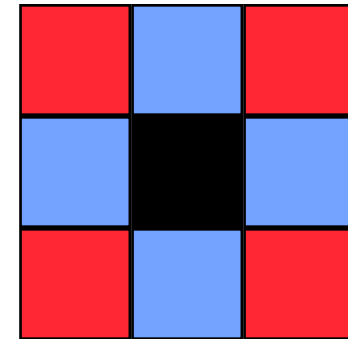
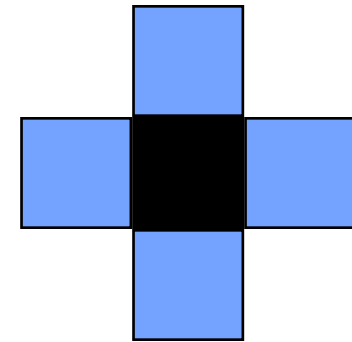


Connected Components

- ◆ Basic definitions
 - ◆ Connectivity, Adjacency, Connected Components
 - ◆ Background/Foreground, Boundaries, Morphological operations
 - ◆ Run-length encoding
- ◆ Component Labeling
 - ◆ Recursive algorithm
 - ◆ Two-scan algorithm
- ◆ Chain Codes
- ◆ Integral Images
- ◆ Histograms
 - ◆ Gray and Color Histograms
 - ◆ Edge Histograms
- ◆ Properties

4(8) 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 $(i\pm 1, j)$ and $(i, j\pm 1)$
- ◆ 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 $(i, j\pm 1)$, $(i\pm 1, j)$ and $(i\pm 1, j\pm 1)$



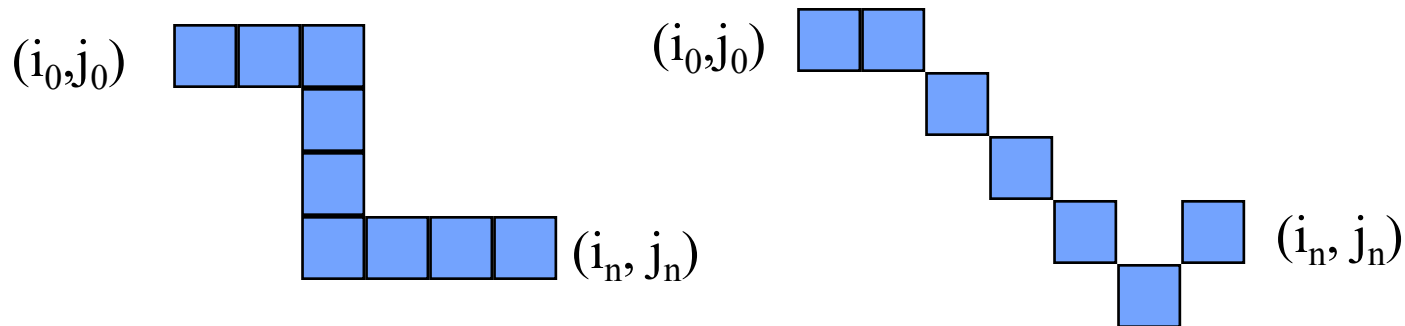
Adjacency

- ◆ Definition: Given two disjoint sets of pixels, A and B, A is **4-(8) adjacent** to B if there is a pixel in A that is a 4-(8) neighbor of a pixel in B



Connected components

- ◆ 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 $k = 0, \dots, n-1$



Every 4-path is an 8-path!

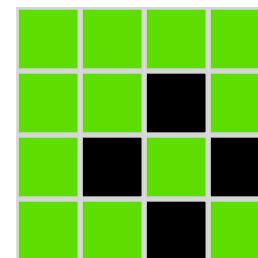
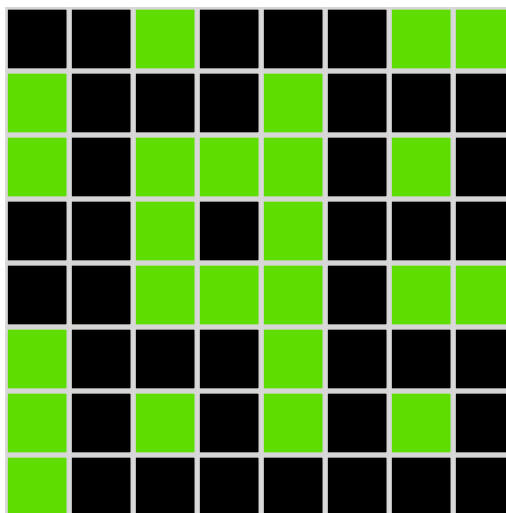
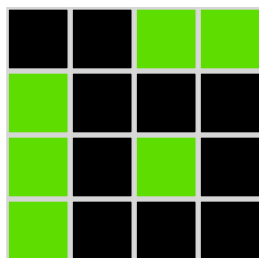
Connected components

- ◆ Definition: Given a binary image, B , the set of all 1's is called the **foreground** and is denoted by S
- ◆ Definition: Given a pixel p in S , p is **4-(8)connected** to q in S if there is a path from p to q consisting only of points from S .
- ◆ The relation “is-connected-to” is an equivalence relation
 - ◆ 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**
- ◆ Definition: \bar{S} is the complement of S - it is the set of all pixels in B whose value is 0
 - ◆ \bar{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 \bar{S} that are adjacent to this frame is called the **background** of B .
 - ◆ All other components of \bar{S} are called holes

Examples - Black = 1, Green = 0



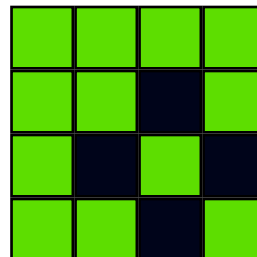
How many 4-(8) components of S?

What is the background?

Which are the 4-(8) holes?

Background and foreground connectivity

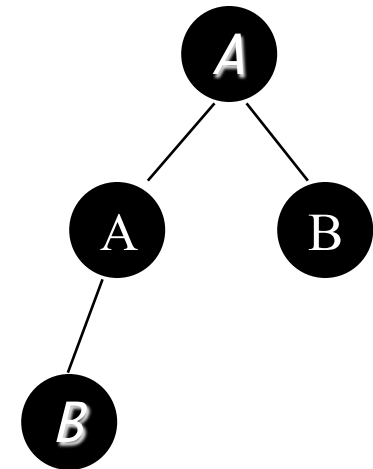
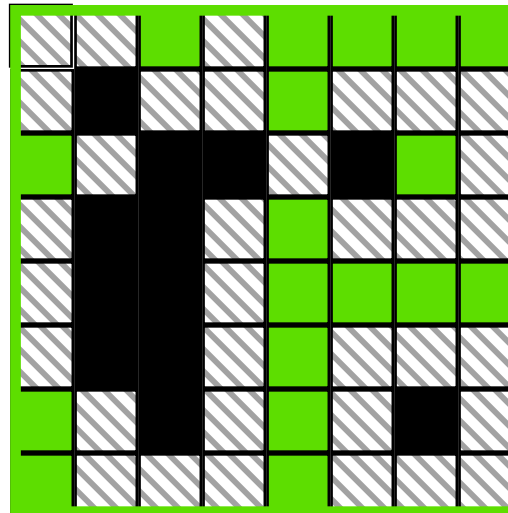
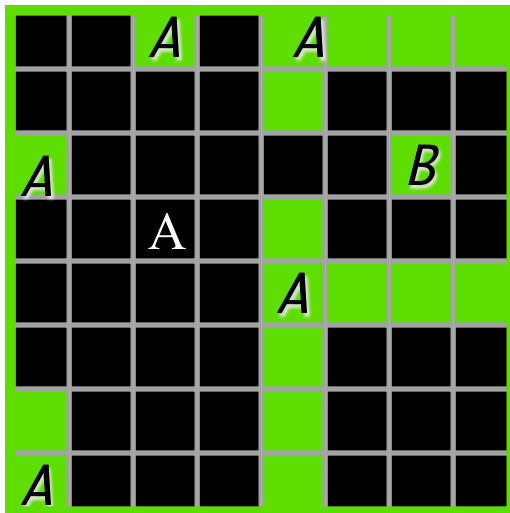
- ◆ Use opposite connectivity for the foreground and the background
 - ◆ 4-foreground, 8-background: 4 single pixel objects and no holes
 - ◆ 4-background, 8-foreground: one 4 pixel object containing a 1 pixel hole



Boundaries

- ◆ The **boundary** of S is the set of all pixels of S that have 4-neighbors in S . The boundary set is denoted as S' .
- ◆ The **interior** is the set of pixels of S that are not in its boundary: $S - S'$
- ◆ Definition: Region T **surrounds** region R (or R is **inside** T) if any 4-path from any point of R to the background intersects T
- ◆ Theorem: If R and T are two adjacent components, then either R surrounds T or T surrounds R .

Examples



Even levels are components of 0's
 The background is at level 0
 Odd levels are components of 1's

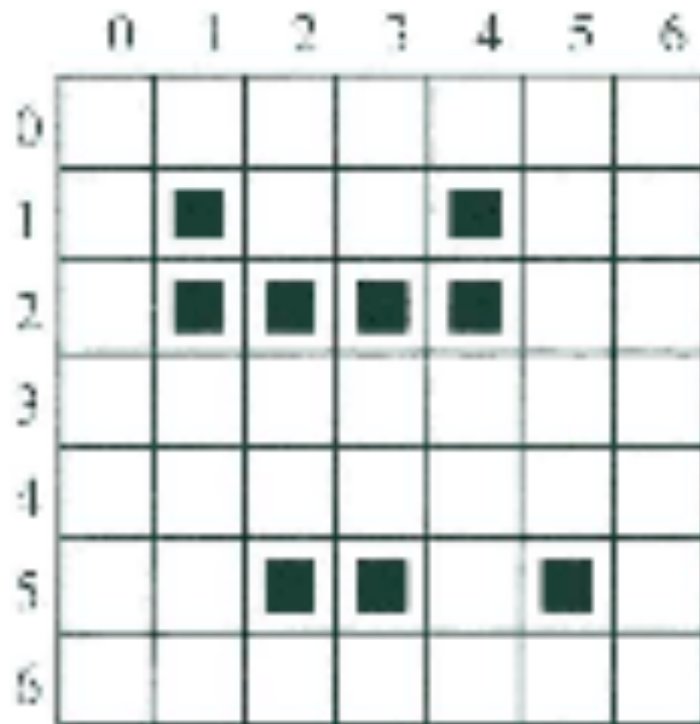
Morphological Operations

- ◆ Parallel operations – cannot be done in place
- ◆ Erosion
 - ◆ In binary images: replace the boundary by the background pixels
 - ◆ In gray level images: replace the pixel value by the min value of its neighbors
- ◆ Dilation
 - ◆ In binary images grows a region by replacing background 4(8) neighbors of all foreground pixels by foreground
 - ◆ In gray level images: replace each pixel value by the max value of its neighbors

Run-length Encoding

Used mainly to represent binary images such as faxes.

Various approaches exist, this approach represents only foreground.



Foreground is a list of lists.

Each non-zero row is represented by a list.

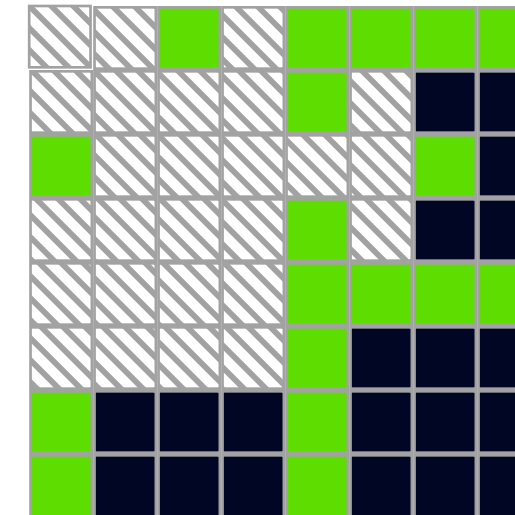
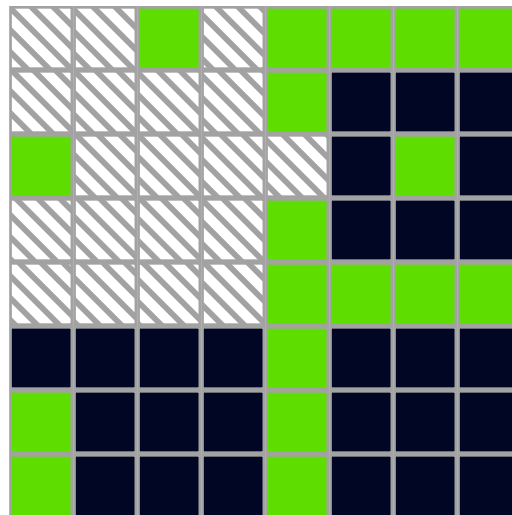
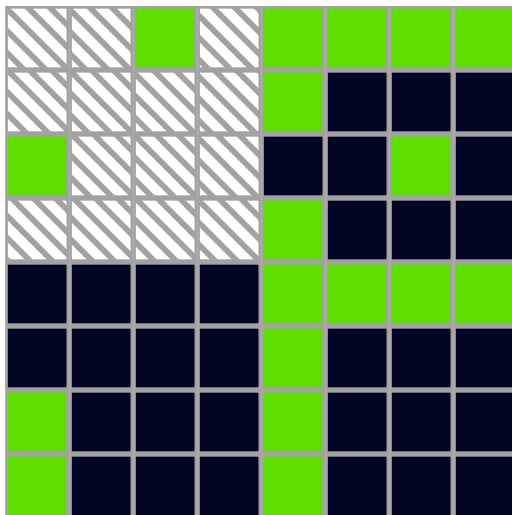
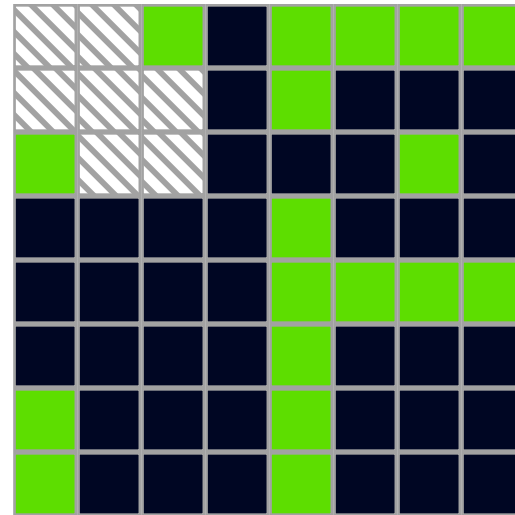
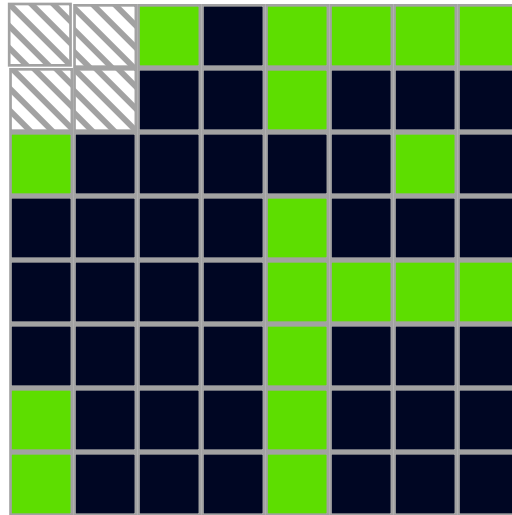
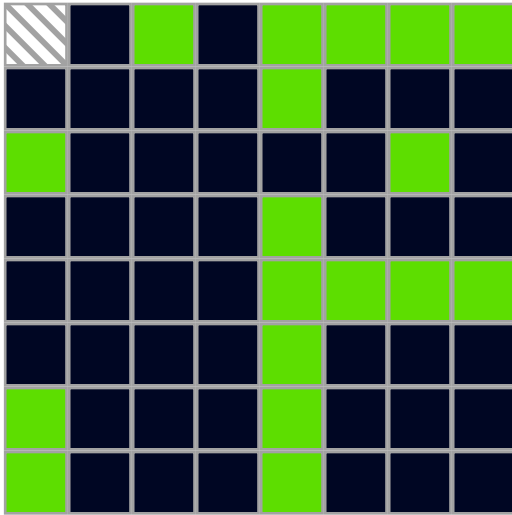
$(r\ b_1e_1\ b_2e_2\dots)$ where r is the row and b_i, e_i are the beginning and ending column indices for a foreground run

$((11144)(214)(52355))$

Component labeling

- ◆ Given: Binary image B
- ◆ Produce: An image in which all of the pixels in each connected component are given a unique label.
- ◆ Solution 1: Recursive, depth first labeling
 - ◆ Scan the binary image from top to bottom, left to right until encountering a 1 (0).
 - ◆ Change that pixel to the next unused component label
 - ◆ Recursively visit all (8,4) neighbors of this pixel that are 1's (0's) and mark them with the new label

Example



Connected components - 14

Recursive Algorithm

1. Create stack S, initially empty
2. Scan the binary image from top to bottom, left to right until encountering a 1 (0).
3. Change that pixel's label to the next unused component label
4. Push the pixel on S (push the coordinates)
5. While S is not empty
 - Pop a pixel p from S
 - For each unlabeled neighbor of p if it's value is 1(0)
 - label it with the current label
 - push it on S

Topology Challenge

- ◆ How to determine which components of 0's are holes in which components of 1's
- ◆ Scan labeled image:
 - ◆ When a new label is encountered make it the child of the label on the left

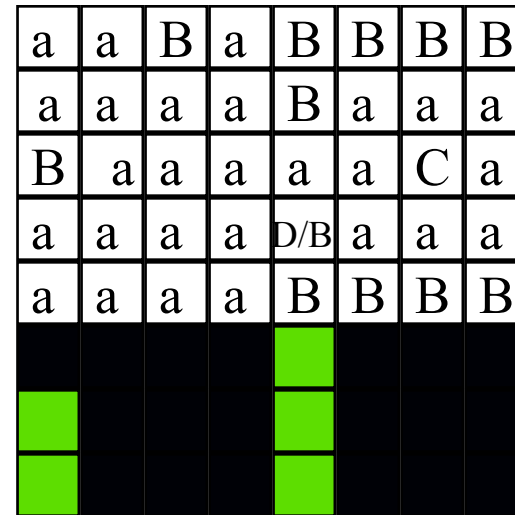
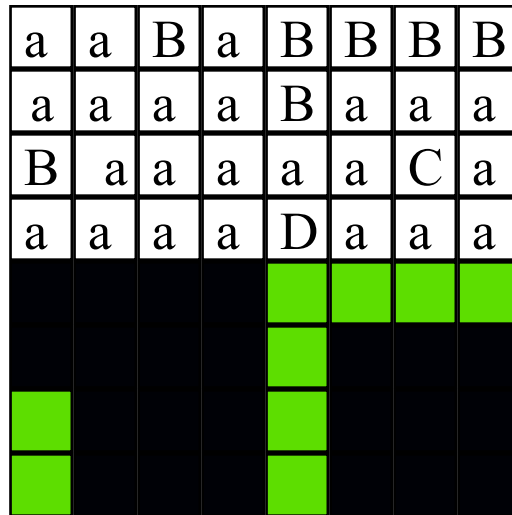
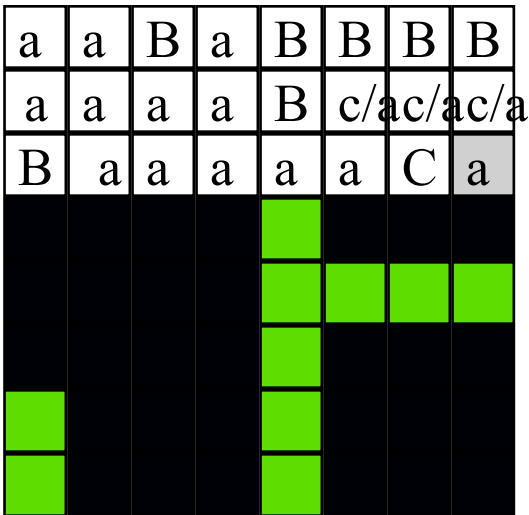
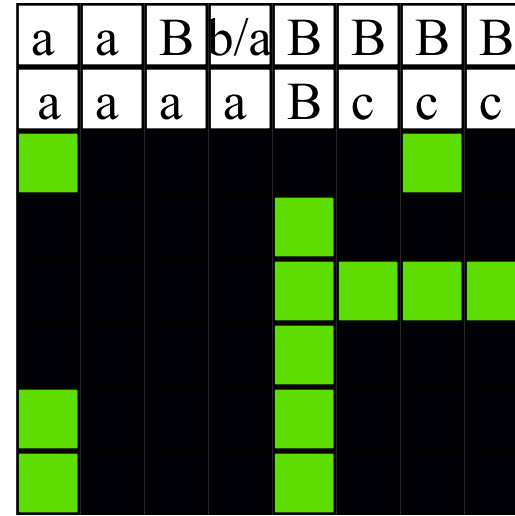
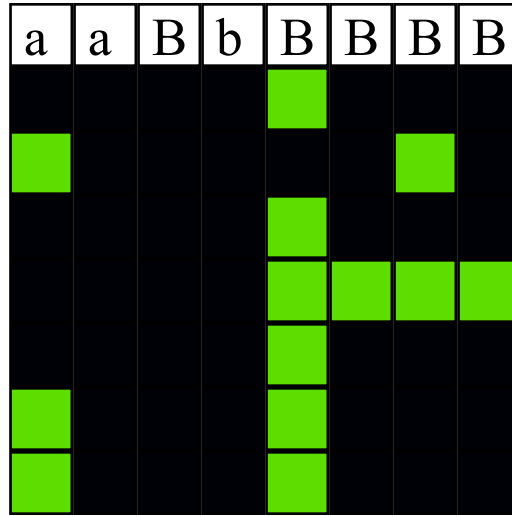
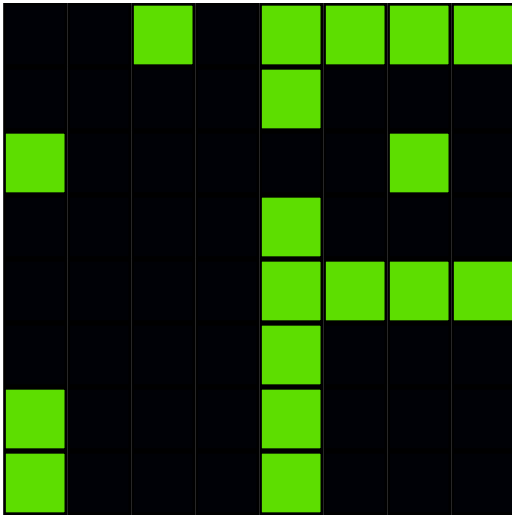
Solution 2 - row scanning up and down

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

Local relabeling

- ◆ What is the point of the last step?
 - ◆ We want the following invariant condition to hold after each row of the image is processed on the downward scan: The label assigned to the runs in the last row processed in any connected component is the **minimum** label of any run belonging to that component in the previous rows.
 - ◆ Note that this only applies to the connectivity of pixels in that part of B already processed. There may be subsequent merging of components in later rows

Example



If we did not change the c' s to a' s, then the rightmost a will be labeled as a c and our invariant condition will fail.

Upward scan

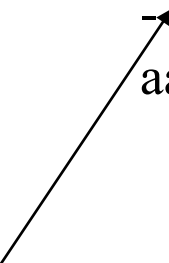
- ◆ A bottom to top scan will assign a unique label to each component
 - ◆ we can also compute simple properties of the components during this scan
- ◆ Start at the bottom row
 - ◆ create a table entry for each unique component label, plus one entry for the background if there are no background runs on the last row
 - ◆ Mark each component of 1's as being “inside” the background

Upward scan

- ◆ For all subsequent rows
 - ◆ if a run of 1' s (0' s) (say with label c) is adjacent to no run of 1' s (0' s) on the subsequent row, and its label is not in the table, and no other run with label c on the current row is adjacent to any run of 1' s on the subsequent row, then:
 - ◆ create a table entry for this label
 - ◆ mark it as inside the run of 0' s (1' s) that it is adjacent to on the subsequent row
 - ◆ property values such as area, perimeter, etc. can be updated as each run is processed.
 - ◆ if a run of 1' s (0' s) (say, with label c) is adjacent to one or more run of 1' s on the subsequent row, then it is marked with the common label of those runs, and the table properties are updated.
 - ◆ All other runs of “c' s” on the current row are also given the common label.

Example

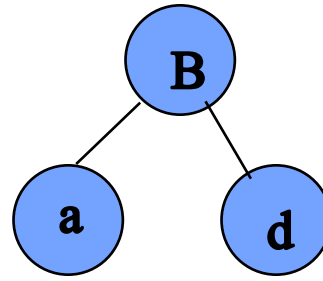
-----aaa
ccc---aaa
c-c---aaa
c-c---aaa
--c---aaa
aaaaaaaa

An arrow originates from the first column of the bottom row (the row containing 'aaaaaaaa') and points diagonally upwards to the first column of the row above it (the row containing '--c---aaa').

- changed to a during first pass
- but c's in first column will not be changed to a's on the upward pass unless all runs are once equivalence is detected

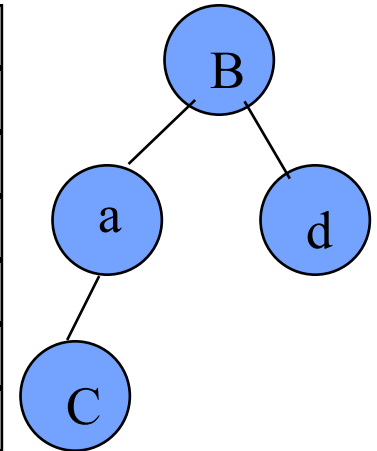
Example

a	a	B	b	B	B	B	B
a	a	a	a	B	c	c	c
B	a	a	a	a	a	C	a
a	a	a	a	D	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d



process row 3

a	a	B	b	B	B	B	B
a	a	a	a	B	c	c	c
B	a	a	a	a	a	C	a
a	a	a	a	B	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d



a	a	B	b	B	B	B	B
a	a	a	a	B	c	c	c
B	a	a	a	a	a	C	a
a	a	a	a	B	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d

process row
4

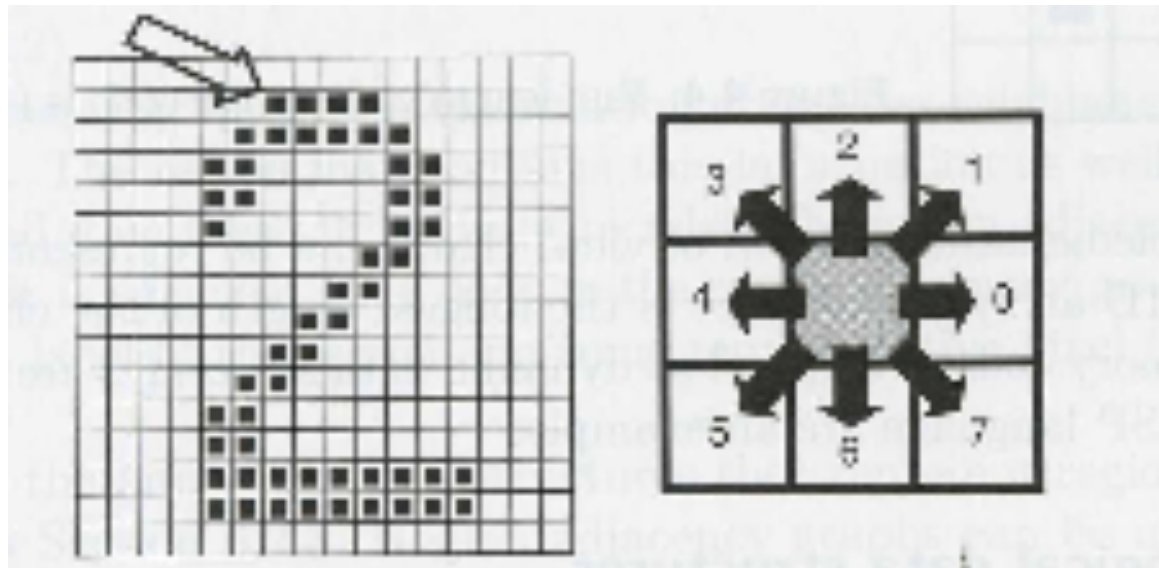
a	a	B	a	B	B	B	B
a	a	a	a	B	a	a	a
B	a	a	a	a	a	C	a
a	a	a	a	B	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d

process row
2, then 1

Chain Codes

Used for efficient boundary representation. First (reference) pixel is recorded.

All other pixels are given by their relative displacement index.



An example chain code. The reference pixel starting the chain is marked by an arrow:

0007766555556700000064444444222111112234445652211

Integral Images

Values $ii(i,j)$ at location (i,j) represent the sums of all the original pixel values left of and above (i,j)

$$ii(i, j) = \sum_{k \leq i, l \leq j} f(k, l)$$

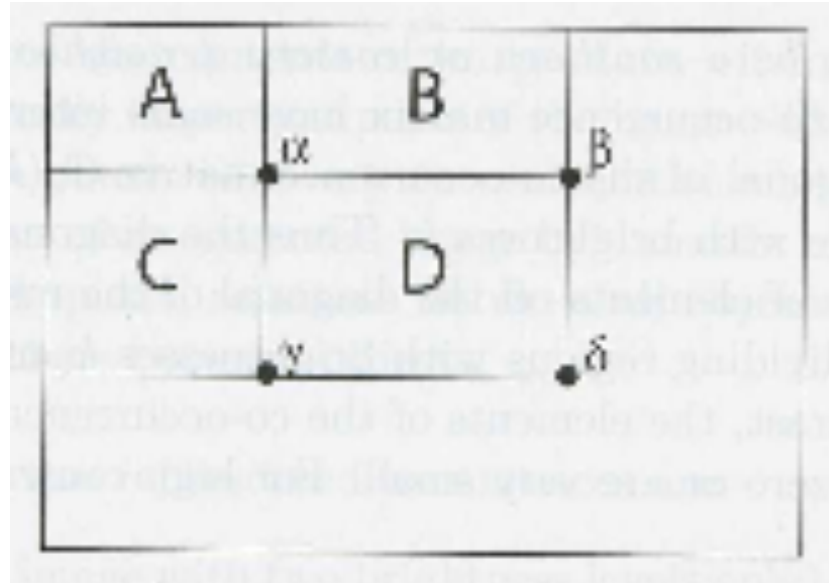
Computing Integral Images:

1. Let $s(i,j)$ denote a cumulative row sum, let $s(i,-1)=0$.
2. Let $ii(i,j)$ be an integral image, let $ii(-1,j)=0$.
3. Using a single row-by-row scan of the image, calculate $s(i,j)$ and $ii(i,j)$ using the following iterative formulas

$$s(i,j) = s(i,j-1) + f(i,j)$$

$$ii(i,j) = ii(i-1,j) + s(i,j)$$

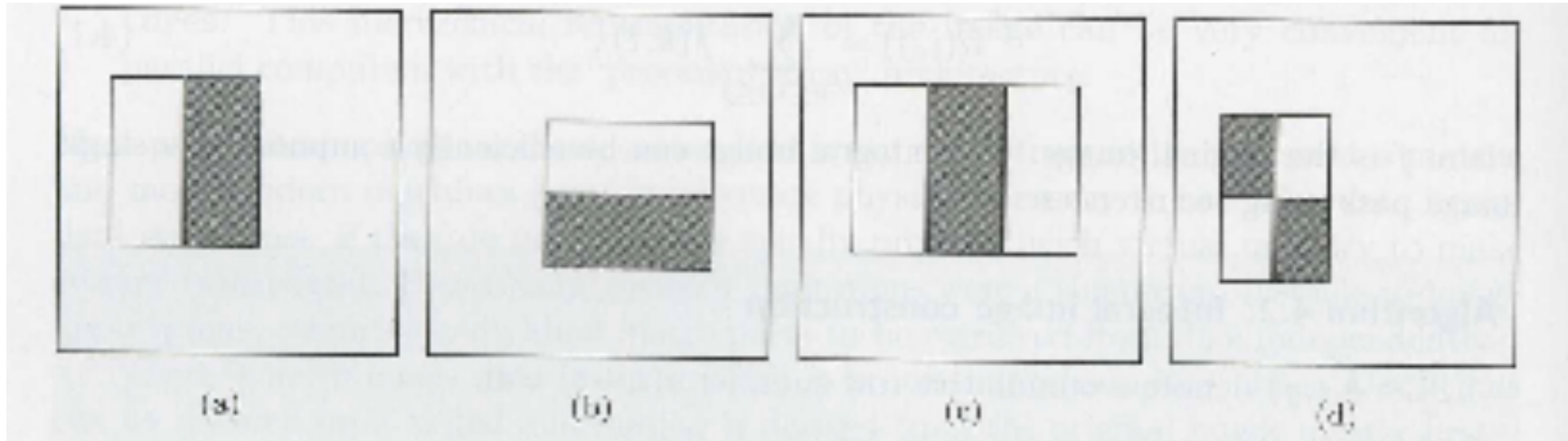
Integral Images: Computing Sums in an Area



The sum of values in area D can be obtained using ii

$$D_{sum} = ii(\delta) + ii(\alpha) - ii(\beta) - ii(\gamma)$$

Integral Images: Computing Rectangle Features



- Rectangle-based features are computed from an integral image.
- These features are computed by subtracting the sum in the shaded rectangle(s) from the sum in the non-shaded rectangle(s).
- Pictures show: (a-b) two-rectangle, (c) three-rectangle, (d) four rectangle.
- These features can easily be computed at different scales/sizes.

Gray Image Histograms

Histogram h :

gray-level frequency distribution of
the gray level image f

$h_f(g)$: # of pixels in f whose gray level is g

Cumulative histogram $H_f(g)$:

of pixels in F whose gray level is $\leq g$

In Matlab: imhist

Color Histograms

- ◆ Reduced color representation =
$$C = (R/16) * 256 + (G/16)*16 + (B/16)$$

(This results in a 24 -> 12 bit color depth reduction)
- ◆ This results in a 4096 bin histogram
 - lowest 4 bits are less useful
 - requires less storage
 - faster implementation - easier to compare histograms

Edge Histograms

- ◆ Use edge detector to compute edges (G_x, G_y)
- ◆ The edge strength is given by the vector magnitude and the orientation is the angle of the vector
- ◆ Histogram bin index is determined using edge orientation (N bins total), and the bin count is incremented using the edge magnitude

Histogram Matching

◆ Histogram Intersection

$$I_h(h_c, h_b) = \frac{\sum_i \min\{h_c(i), h_b(i)\}}{\sum_i \max\{h_c(i), h_b(i)\}}$$

◆ Chi Squared Formula

$$\chi^2(h_c, h_b) = \sum_i 2 \frac{(h_c(i) - h_b(i))^2}{h_c(i) + h_b(i)}$$

Region Properties

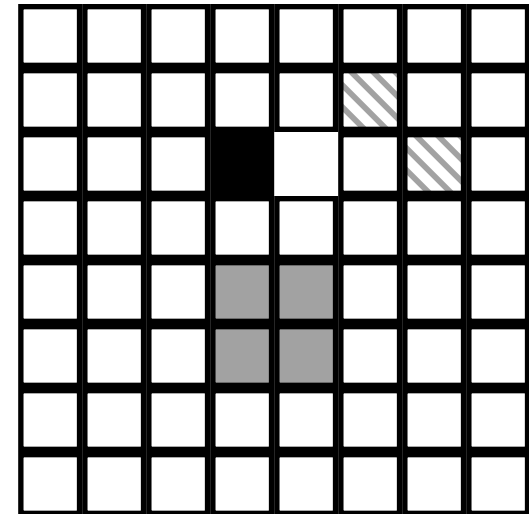
- ◆ Our goal is to recognize each connected component as one of a set of known objects
 - ◆ letters of the alphabet
 - ◆ simple objects
 - ◆ object parts
- ◆ We need to associate measurements, or properties, with each connected component that we can compare against expected properties of different object types.

Properties

- ◆ Area: $A = |S|$
- ◆ Perimeter: $P = \text{boundary length}$
- ◆ Euler's number: $v = S - N$
 - S – number of contiguous parts of an object
 - N – number of holes in an object
- ◆ Compactness: P^2/A
- ◆ Projections
- ◆ Eccentricity
- ◆ Elongatedness
- ◆ Rectangularity
- ◆ Convex Hull
- ◆ Moments

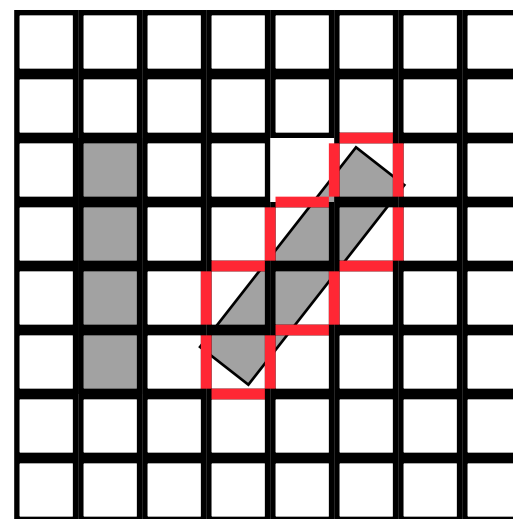
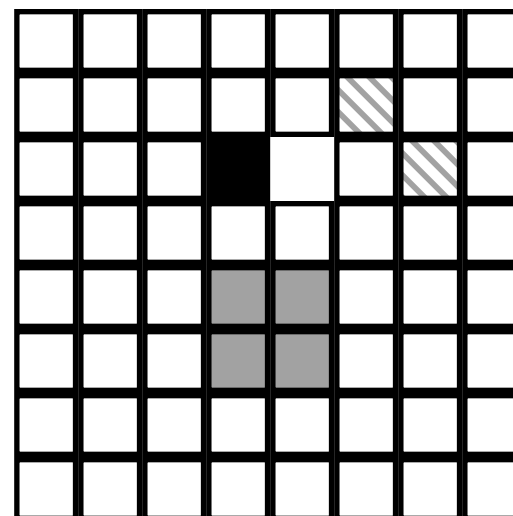
How do we compute the perimeter of a connected component?

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 red 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!

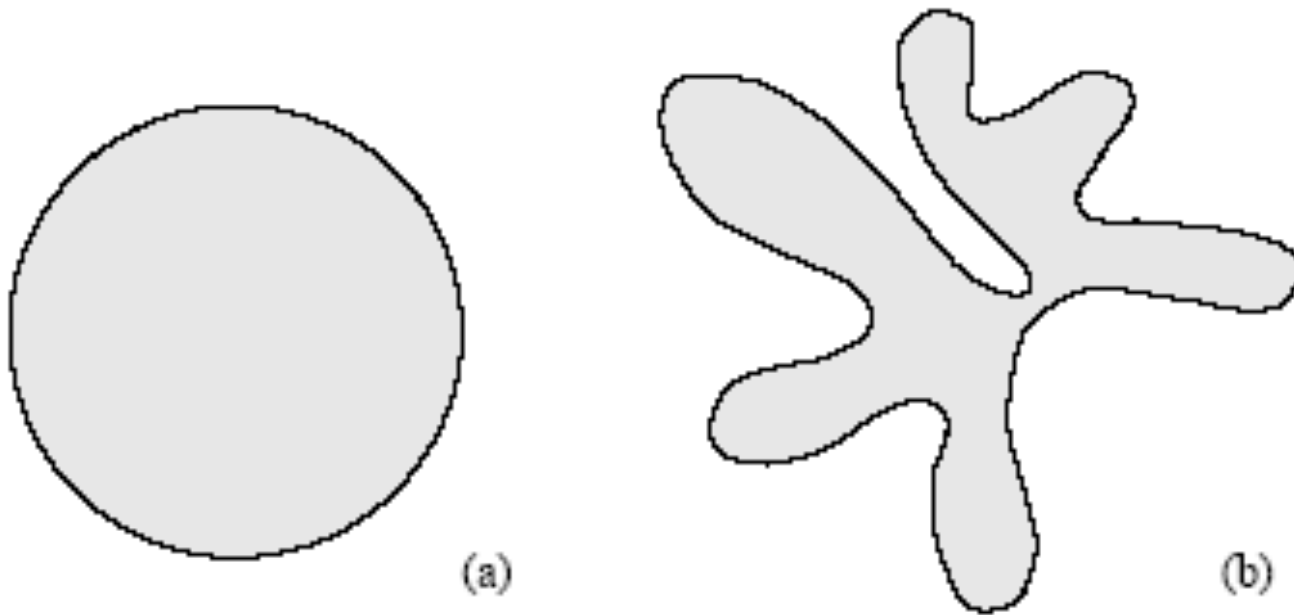


Perimeter computation (cont.)

- ◆ We can give different weights to boundary pixels
 - ◆ 1 – vertical and horizontal pairs
 - ◆ $2^{1/2}$ – diagonal pairs
- ◆ The boundary can be approximated by a polygon line (or splines) and its length could be used
- ◆ It matters most for small (low resolution objects)

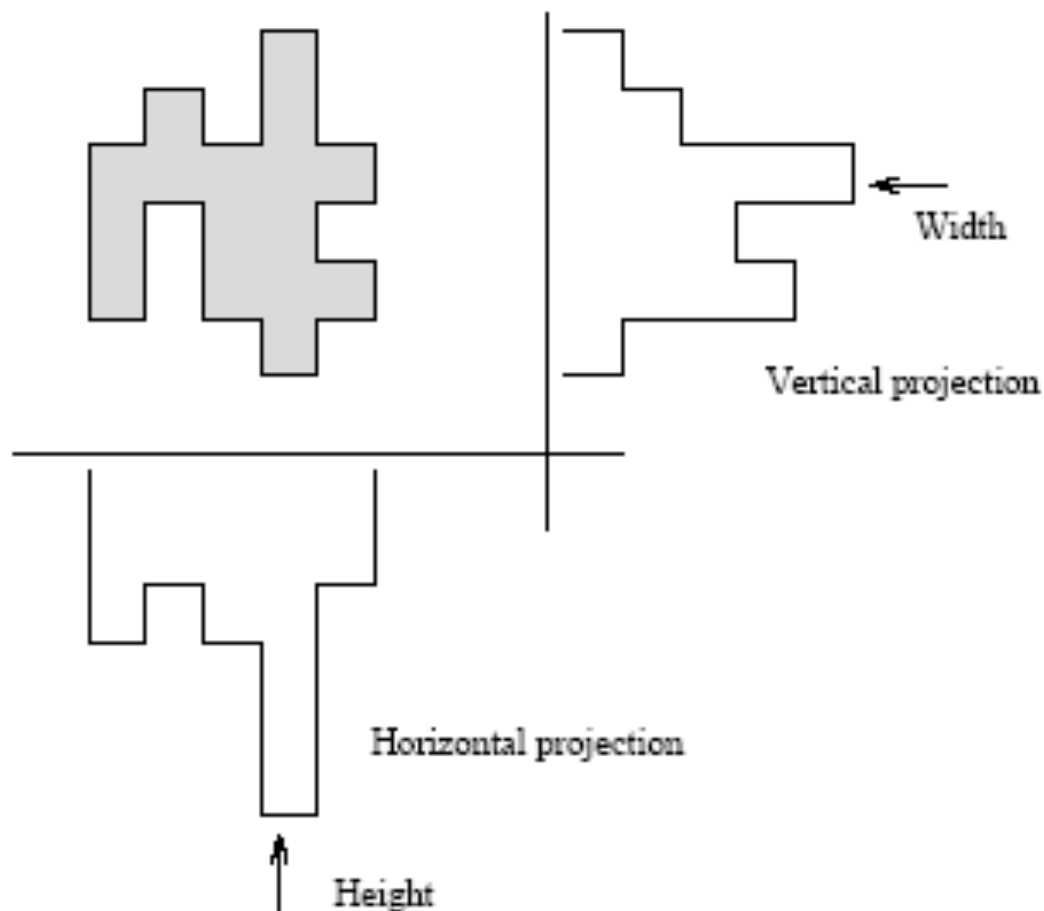
Compactness

- smallest for a circle: $4\pi^2r^2/\pi r^2 = 4\pi$
- higher for elongated objects



a) A compact object. b) no-compact object

Projections



$$p_h(i) = \sum_j f(i, j)$$

$$p_v(j) = \sum_i f(i, j)$$

Eccentricity

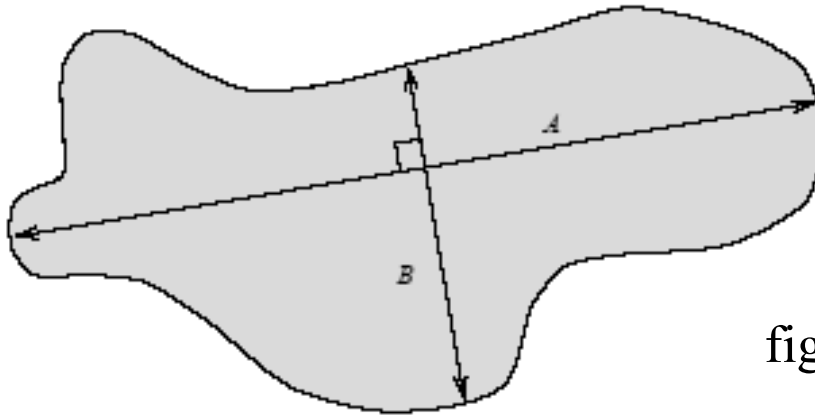


figure from Sonka, Hlavac, Boyle

- ◆ Ratio of the length of the maximum chord A to the maximum chord B perpendicular to A (ratio of major and minor axes of an object)
- ◆ An approximate measure could be based on a ratio of main region axes of inertia

Elongatedness

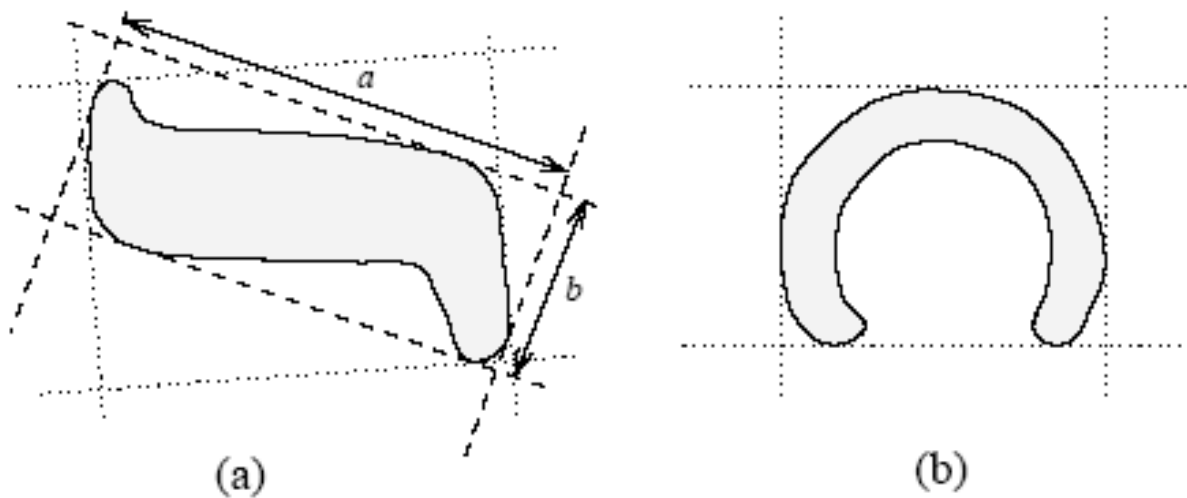


figure from Sonka, Hlavac, Boyle

- ◆ Ratio of the length of the sides of the region bounding rectangle
- ◆ To make it work for regions like (b) sometimes redefined as the ratio of area and maximum region thickness
 - ✧ The thickness can be computed by counting the number of erosion steps d needed to completely remove the object

$$E = \frac{A}{(2d)^2}$$

Rectangularity

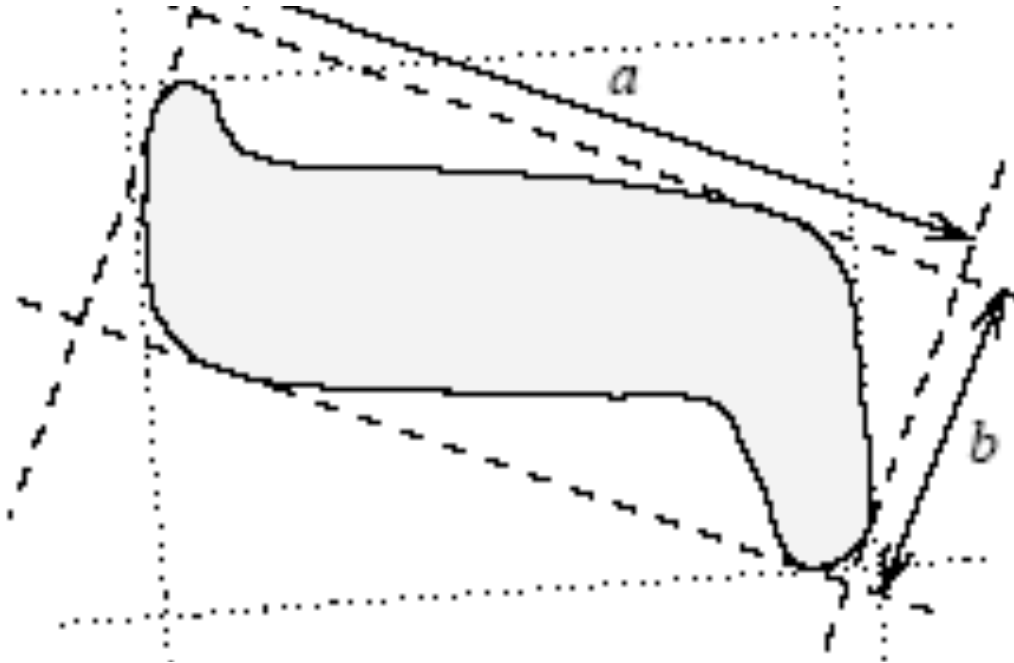
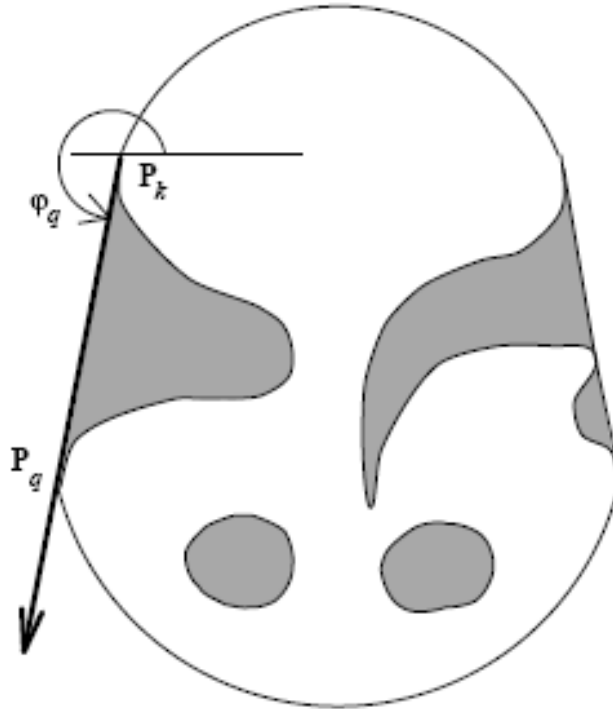


figure from Sonka, Hlavac, Boyle

- ◆ Ratio of area of bounding rectangle ab and area of the object

Convex Hull



Features

- Area of the CH
- Ratio of the area of the CH and the region

figure from Sonka, Hlavac, Boyle

- ◆ Computed by creating a monotone polygon (min and max pixel coordinate in each row)
- ◆ Followed by iteratively filling concave regions (computed using triples of pixels along the boundary)
- ◆ Number of steps linear in number of boundary pixels

Concavity Tree

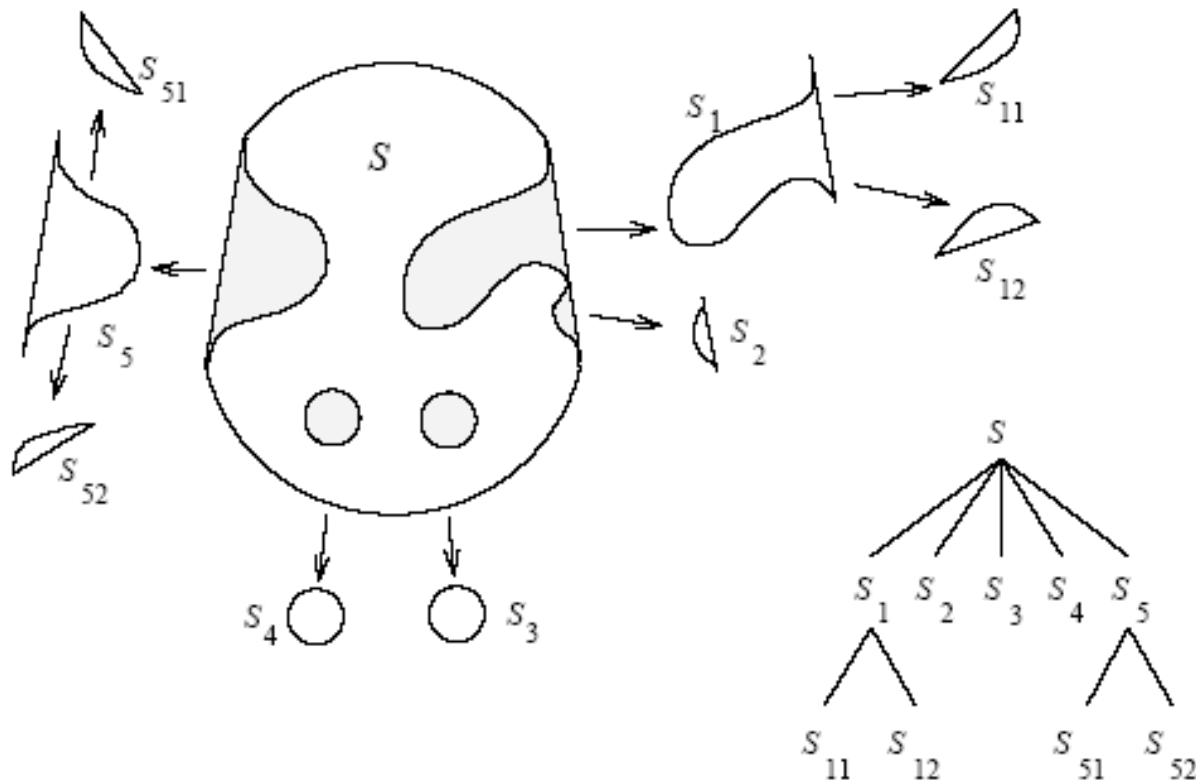
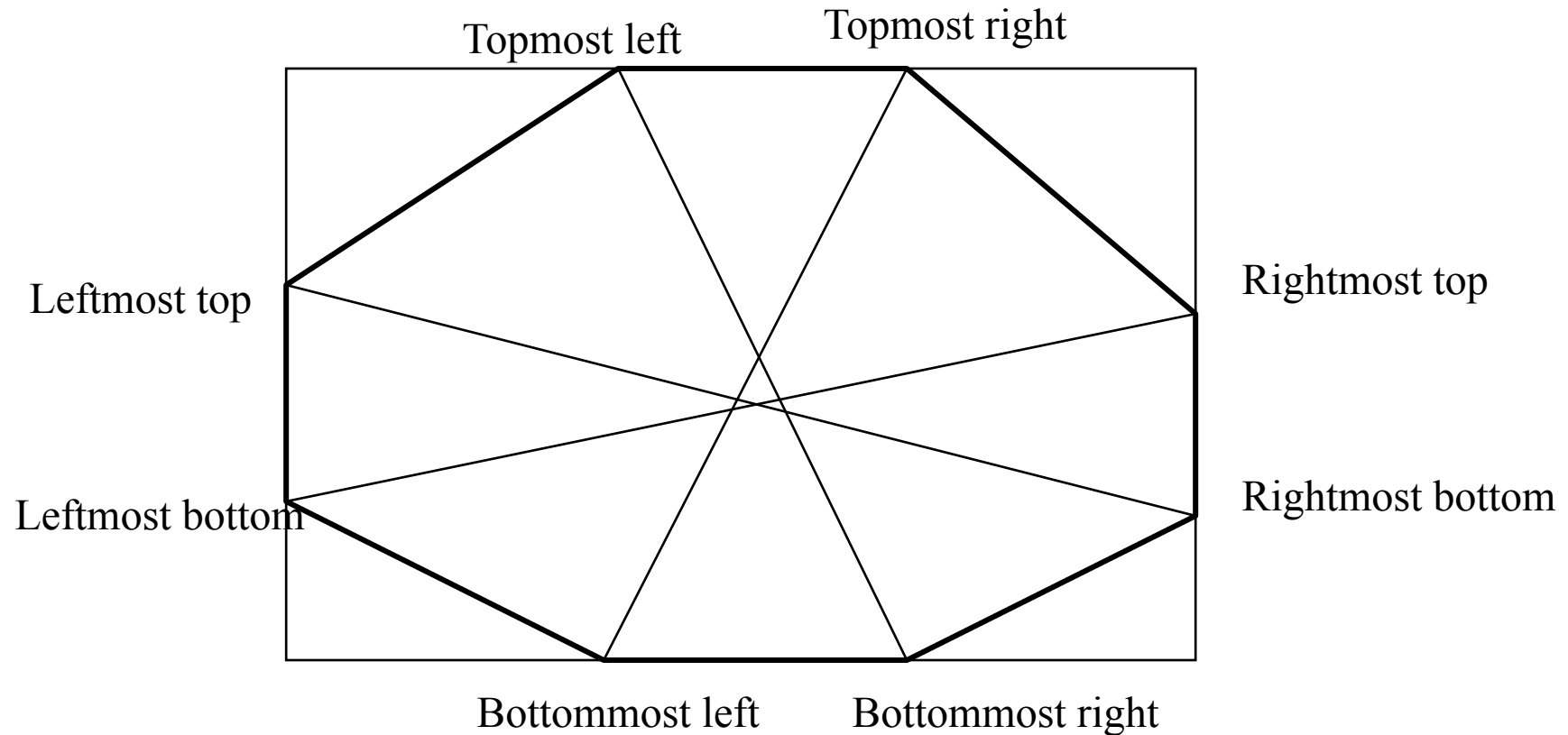


figure from Sonka, Hlavac, Boyle

- ◆ Constructed iteratively starting from the convex hull

Bounding Box and Extremal Points



Can be used to approximate various properties

Moments

- ◆ An “ideal” set of features should be independent of
 - ◆ the position of the connected component
 - ◆ the orientation of the connected component
 - ◆ the size of the connected component
 - ◆ ignoring the fact that as we “zoom in” on a shape we tend to see more detail
- ◆ These problems are solved by features called moments

Central moments

- ◆ Let S be a connected component in a binary image
 - ◆ generally, S can be any subset of pixels, but for our application the subsets of interest are the connected components
- ◆ The (j,k) 'th moment of S is defined to be

$$M_{jk}(S) = \sum_{(x,y) \in S} x^j y^k$$

Central moments

- ◆ M_{00} = the area of the connected component

$$M_{00} = \sum_{(x,y) \in S} x^0 y^0 = \sum_{(x,y) \in S} 1 = |S|$$

- ◆ The center of gravity of S can be expressed as

$$\bar{x} = \frac{M_{10}(S)}{M_{00}(S)} = \frac{\sum x}{|S|}$$

$$\bar{y} = \frac{M_{01}(S)}{M_{00}(S)} = \frac{\sum y}{|S|}$$

Central moments

- ◆ Using the center of gravity, we can define the central (j,k)'th moment of S as

$$\mu_{jk} = \sum_{(x,y) \in S} (x - \bar{x})^j (y - \bar{y})^k$$

- ◆ If the component S is translated, this means that we have added some numbers (a,b) to the coordinates of each pixel in S
 - ◆ for example, if a = 0 and b = -1, then we have shifted the component up one pixel

Central moments

- ◆ Central moments are not affected by translations of S . Let $S' = \{(x', y') : x' = x + a, y' = y + b, (x, y) \in S\}$
 - ◆ The center of gravity of S' is the c.o.g. of S shifted by (a, b)

$$\bar{x}(S') = \frac{\sum x'}{|S'|} = \frac{\sum (x + a)}{|S|} = \frac{\sum x}{|S|} + \frac{\sum a}{|S|} = \bar{x} + a$$

- ◆ The central moments of S' are the same as those of S

$$\begin{aligned}\mu_{jk}(S') &= \sum (x' - \bar{x}(S'))^j (y' - \bar{y}(S'))^k \\ &= \sum (x + a - [\bar{x}(S) + a])^j (y + b - [\bar{y}(S) + b])^k \\ &= \sum (x - \bar{x})^j (y - \bar{y})^k = \mu_{jk}(S)\end{aligned}$$

Object Orientation

- ◆ Object orientation can be computed from central moments

$$\theta = \frac{1}{2} \arctan \left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right)$$

Central moments

- ◆ The standard deviations of the x and y coordinates of S can also be obtained from central moments:

$$\sigma_x = \sqrt{\frac{\mu_{20}}{|S|}}, \quad \sigma_y = \sqrt{\frac{\mu_{02}}{|S|}}$$

- ◆ We can then create a set of normalized coordinates of S that we can use to generate moments unchanged by translation and scale changes

$$\tilde{x} = \frac{x - \bar{x}}{\sigma_x}, \quad \tilde{y} = \frac{y - \bar{y}}{\sigma_y}$$

Normalized central moments

- ◆ The means of these new variables are 0, and their standard deviations are 1. If we define the normalized moments; m_{jk} as follows

$$m_{jk} = \frac{\sum \tilde{x}^j \tilde{y}^k}{M_{00}}$$

- ◆ then these moments are not changed by any scaling or translation of S
- ◆ Let $S^* = \{(x^*, y^*) : x^* = ax + b, y^* = ay + c, (x, y) \text{ in } S\}$
 - ◆ if b and c are 0, then we have scaled S by a
 - ◆ if a is 0, then we have translated S by (b,c)

Normalized central moments

$$\begin{aligned} m_{jk}(S^*) &= \frac{1}{|S|} \sum \left(\frac{x^* - \overline{x(S^*)}}{\sigma_x(S^*)} \right)^j \left(\frac{y^* - \overline{y(S^*)}}{\sigma_y(S^*)} \right)^k \\ &= \frac{1}{|S|} \sum \frac{a^j (x - \bar{x}(S))^j}{a^j \sigma_x^j(S)} \frac{a^k (y - \bar{y}(S))^k}{a^k \sigma_y^k(S)} \\ &= m_{jk}(S) \end{aligned}$$

◆ Details of the proof are simple.