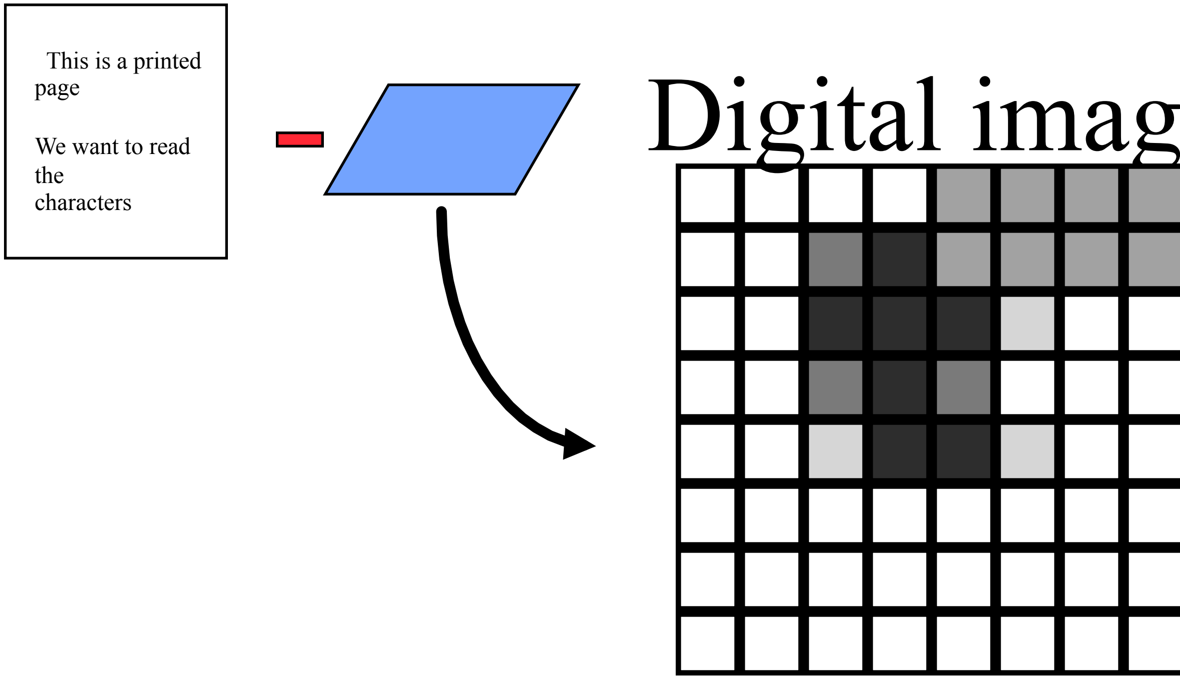
Machine vision systems

- Problem definition
- Image acquisition
- Image segmentation
- Connected component analysis

Problem definition

- Design a vision system to “see” a “flat” world
  - Page of text
  - Side panel of a truck
  - X-ray image of separated potatoes
- 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

Digital image acquisition



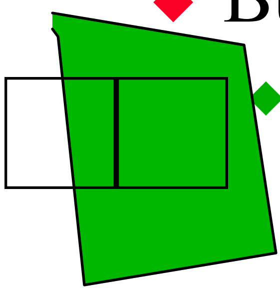
- Camera, such as a scanner, measures intensity of reflected light over a regularly spaced grid of positions
  - individual grid elements are called **pixels**
    - typical grid from a TV camera is 512 x 480 pixels created by a process called sampling.
    - scanner can produce much higher **resolution** images
  - measurements at pixels are called **intensities** or **gray levels**
    - typical camera provides 6-8 bits of intensity per pixel created by a process called quantization.

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

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”
  - potatoes have non-uniform “thickness”, giving variations in brightness in X-ray - model “noise”

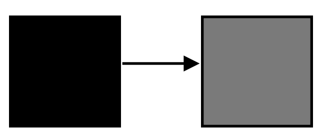
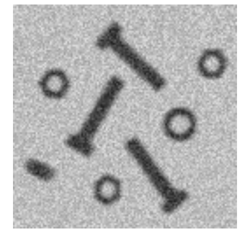
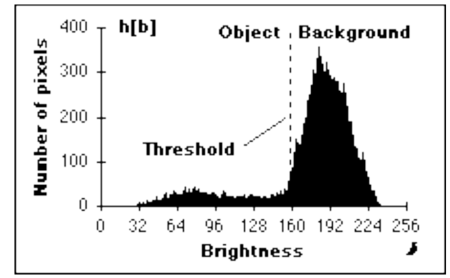


Image segmentation by thresholding

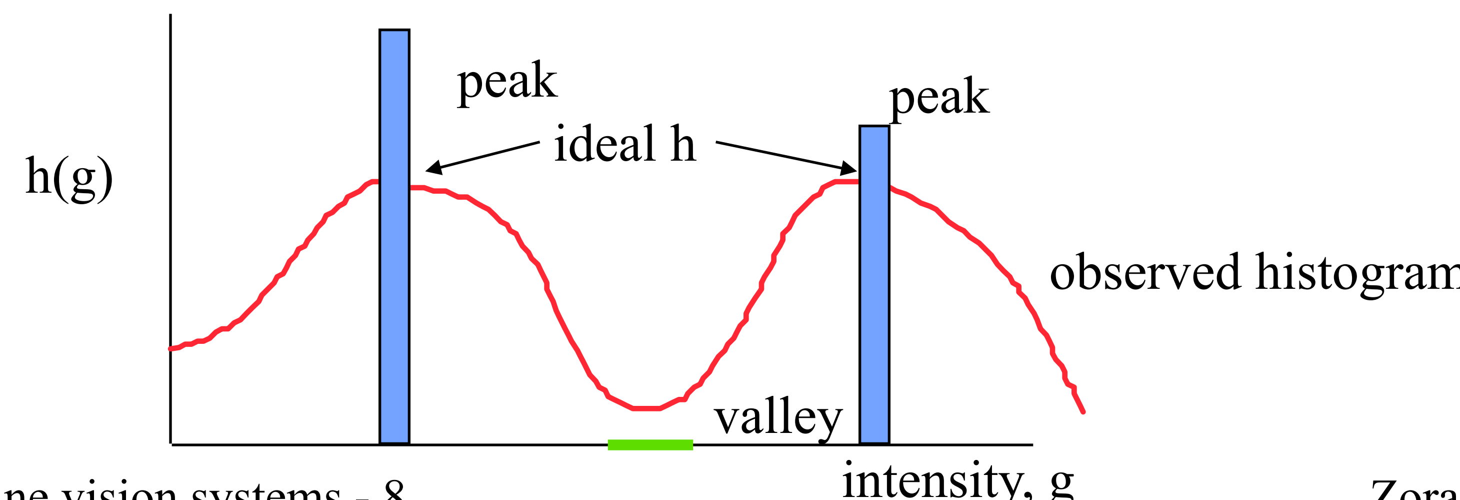
- But if the objects and background occupy different ranges of gray levels, we can "mark" the object pixels by a process called **thresholding:**
  - Let F(i,j) be the original, gray level image
  - B(i,j) is a **binary image** (pixels are either 0 or 1) created by **thresholding** F(i,j)
    - B(i,j) = 1 if F(i,j) < t
    - B(i,j) = 0 if F(i,j) >= t
    - We will assume that the 1's are the object pixels and the 0's are the background pixels

Thresholding





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

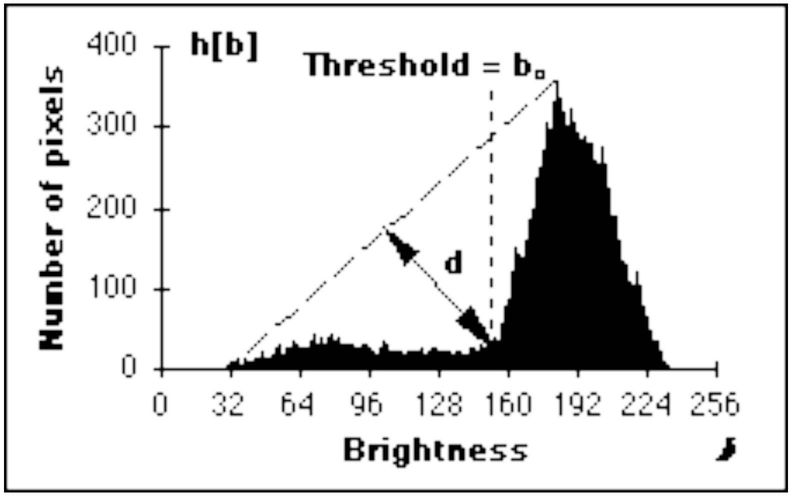


Thresholding

- 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.
  - H_F can be used to find the gray level, g, such that ~p% of the pixels have intensity <= g
  - Then, we can examine h_F in the neighborhood of g to find a good threshold (low valley point)

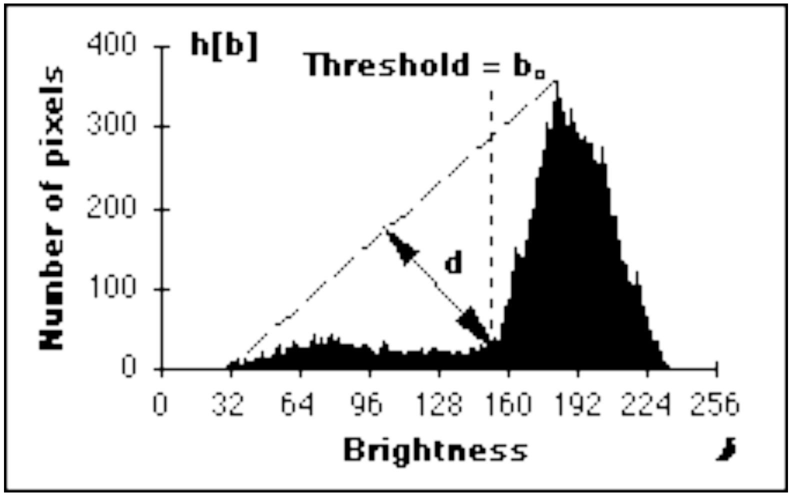
Thresholding

- Peak and valley method
  - Find the two most prominent peaks of h
    - g is a peak if $h_F(g) > h_F(g \pm \Delta g)$, $\Delta g = 1, \ldots, k$
  - 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 $[g_1, g_2]$
  - Use g as the threshold



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
  - select a threshold by hand at the beginning of the day
  - use that threshold all day long!
- 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, most easily recognized objects, etc.
  - Local thresholding
    - apply thresholding methods to image windows

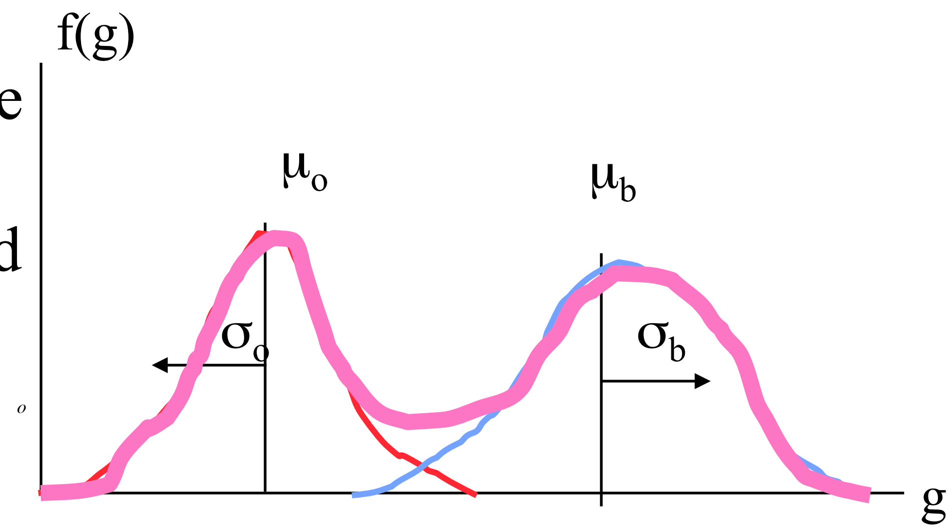
An advanced probabilistic threshold selection method - minimizing Kullback information distance

- The observed histogram, f, is a mixture of the gray levels of the pixels from the object(s) and the pixels from the background
  - in an ideal world the histogram would contain just two spikes
  - but
    - measurement noise,
    - model noise (e.g., variations in ink density within a character) and
    - edge blur (misalignment of object boundaries with pixel boundaries and optical imperfections of camera)

  spread these spikes out into hills

Kullback information distance

- Make a parametric model of the shapes of the component histograms of the objects(s) and background
- Parametric model - the component histograms are assumed to be Gaussian
  - p_o and p_b are the proportions of the image that comprise the objects and background
  - μ_o and μ_b are the mean gray levels of the objects and background
  - σ_o and σ_b- are their standard deviations



$$f_o(g) = \frac{p_o}{\sqrt{2\pi}\sigma_o} e^{-1/2\left(\frac{g-\mu_o}{\sigma_o}\right)^2}$$

$$f_b(g) = \frac{p_b}{\sqrt{2\pi}\sigma_b} e^{-1/2\left(\frac{g-u_b}{\sigma_b}\right)^2}$$

Kullback information distance

- Now, if we hypothesize a threshold, t, then all of these unknown parameters can be approximated from the image histogram.
- Let f(g) be the observed and normalized histogram
  - f(g) = percentage of pixels from image having gray level g

$$p_o(t) = \sum_{g=0}^{t} f(g) \qquad p_b(t) = 1 - p_0(t)$$

$$\mu_o(t) = \sum_{g=0}^{t} f(g)g \qquad \mu_b(t) = \sum_{g=t+1}^{\max} f(g)g$$

Kullback information distance

- So, for any hypothesized t, we can "predict" what the total normalized image histogram **should** be if our model (mixture of two Gaussians) is correct.
  - $P_t(g) = p_o f_o(g) + p_b f_b(g)$
- The total normalized image histogram is **observed to be** f(g)
- So, the question reduces to:
  - determine a suitable way to measure the similarity of P and f
  - then search for the t that gives the highest similarity

Kullback information distance

- A suitable similarity measure is the Kullback directed divergence, defined as

$$K(t) = \sum_{g=0}^{\max} f(g)\log[\frac{f(g)}{P_t(g)}]$$

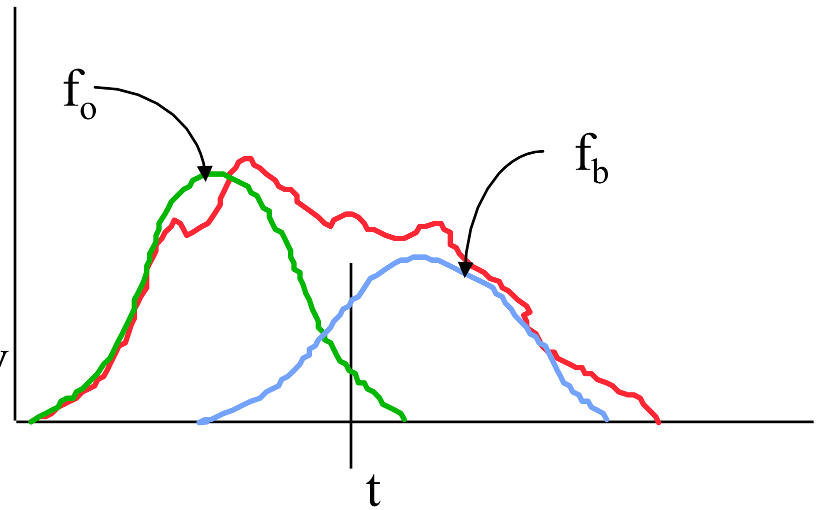
- If P_t matches f exactly, then each term of the sum is 0 and K(t) takes on its minimal value of 0
- Gray levels where P_t and f disagree are penalized by the log term, weighted by the importance of that gray level (f(g))

An alternative - minimize probability of error

- Using the same mixture model, we can search for the t that minimizes the predicted probability of error during thresholding
- Two types of errors
  - background points that are marked as object points. These are points from the background that are darker than the threshold
  - object points that are marked as background points. These are points from the object that are brighter than the threshold

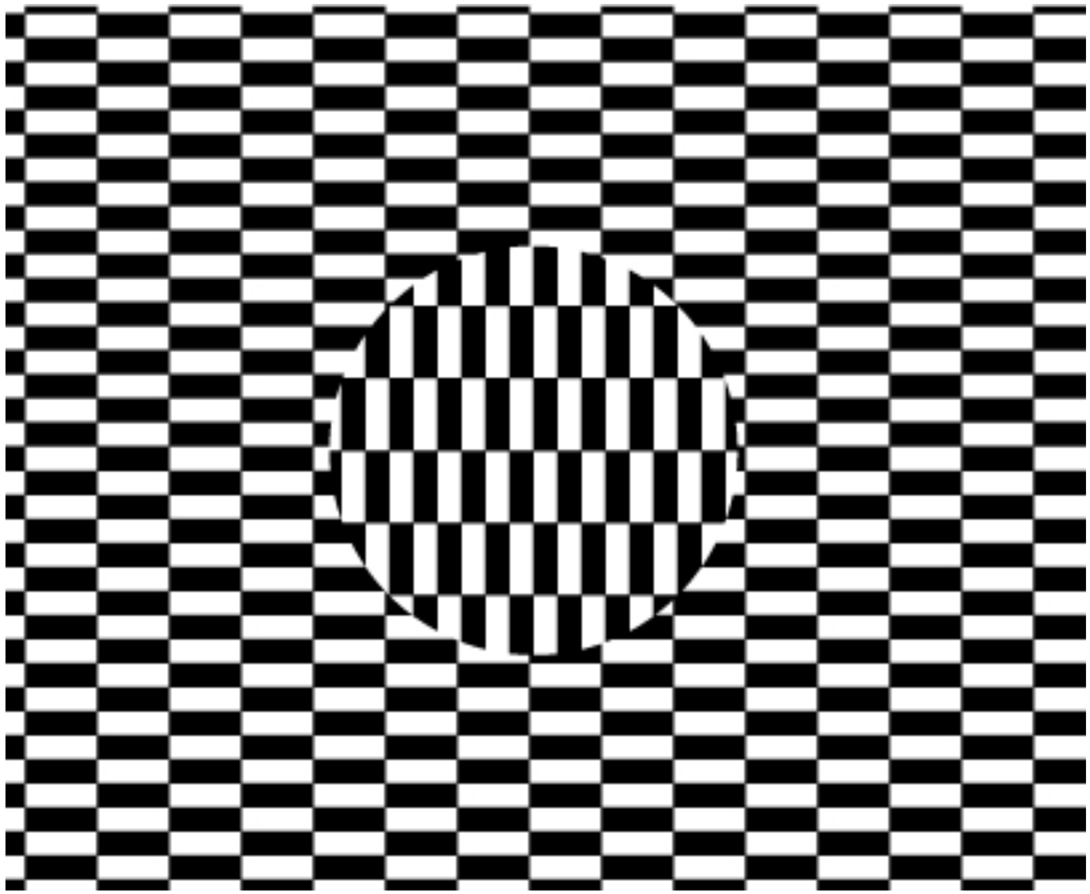
An alternative - mimimize probability of error

- For each "reasonable" threshold
  - compute the parameters of the two Gaussians and the proportions
  - compute the two probability of errors



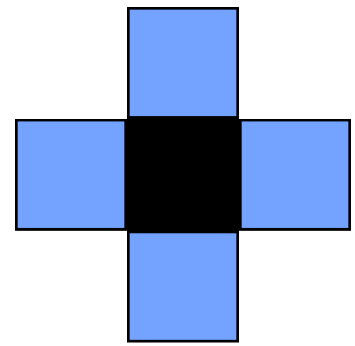
- Find the threshold that gives
  - minimal overall error
  - most equal errors

$$e_b(t) = p_b \sum_{g=0}^{t} f_b(g) \qquad e_o(t) = p_o \sum_{g=t+1}^{max} f_o(g)$$

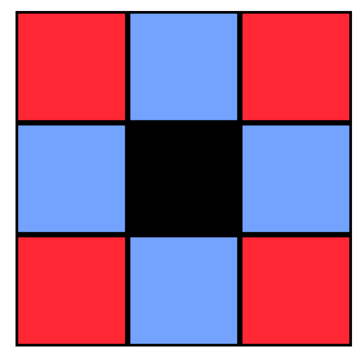


Object extraction from binary images - connected components

- 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 $(i\pm i, j)$ and $(i,j \pm 1)$



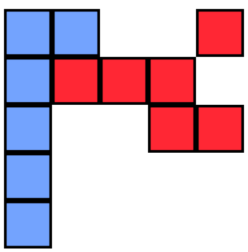
- Definition: Given a pixel (i,j) its 8-neighbors are the points (i',j') such that $\max(|i-i'|,|j-j'|) = 1$
  - the 8- neighbors are $(i, j\pm 1)$, $(i\pm 1, j)$ and $(i\pm 1, j\pm 1)$

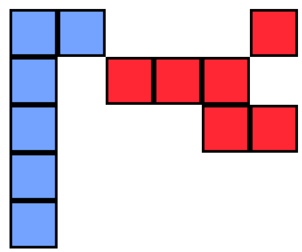


Zoran Duric

Adjacency

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

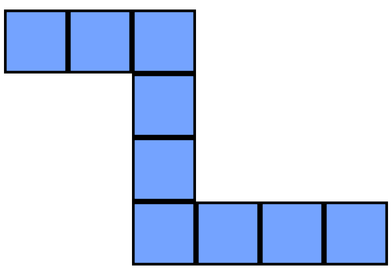




Connected components

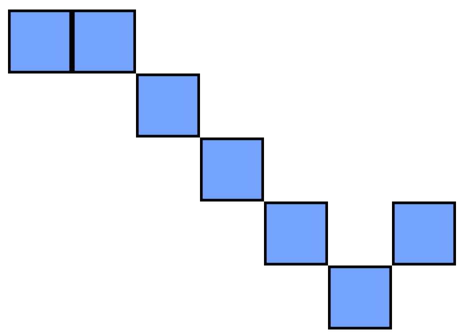
- Definition: A 4-(8)path from pixel (i_0,j_0) to (i_n,j_n) is a sequence of pixels (i_0,j_0) (i_1,j_1) (i_2,j_2) , ... (i_n,j_n) such that (i_k, j_k) is a 4-(8) neighbor of (i_{k+1}, j_{k+1}), for k = 0, ..., n-1

(i_0,j_0)



(i_n, j_n)

(i_0,j_0)



(i_n, j_n)

Every 4-path is an 8-path!

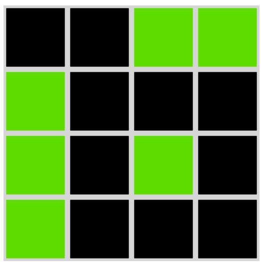
Connected components

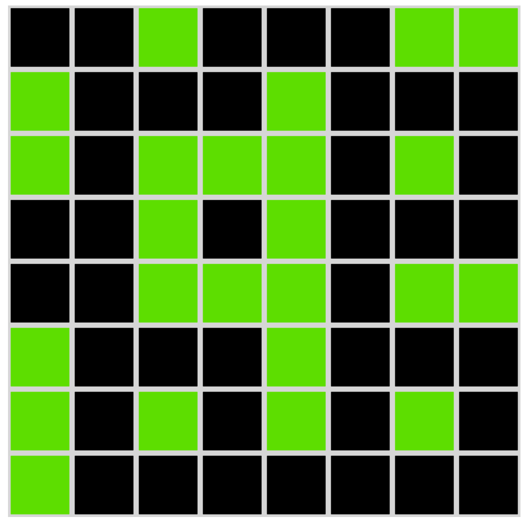
- Definition: Given a binary image, B, the set of all 1's is called the foreground and is denoted by S
- Definition: Given a pixel p in S, p is 4-(8) connected to q in S if there is a path from p to q consisting only of points from S.
- The relation "is-connected-to" is an equivalence relation
  - Reflexive - 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

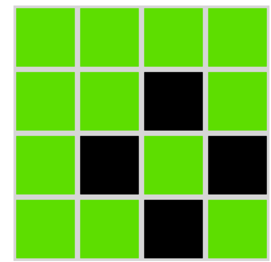
Connected components

- 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: 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 - Black = 1, Green = 0



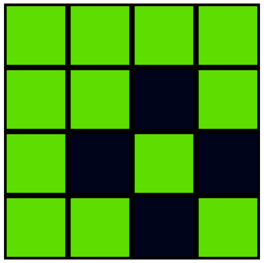




How many 4- (8) components of S?
What is the background?
Which are the 4- (8) holes?

Background and foreground connectivity

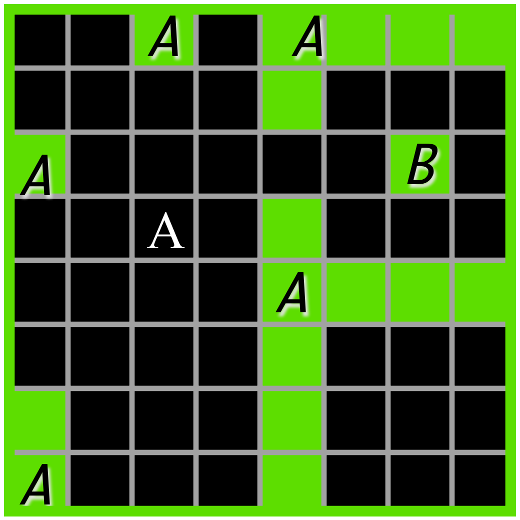
- Use opposite connectivity for the foreground and the background
  - 4-foreground, 8-background: 4 single pixel objects and no holes
  - 4-background, 8-foreground: one 4 pixel object containing a 1 pixel hole

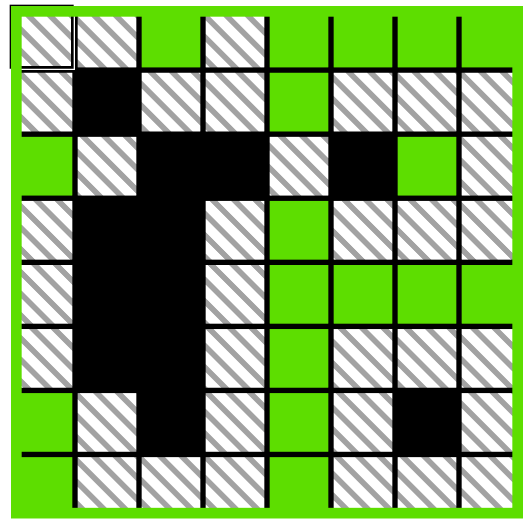


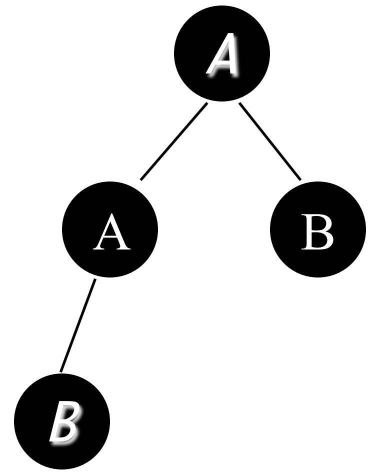
Boundaries

- 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.

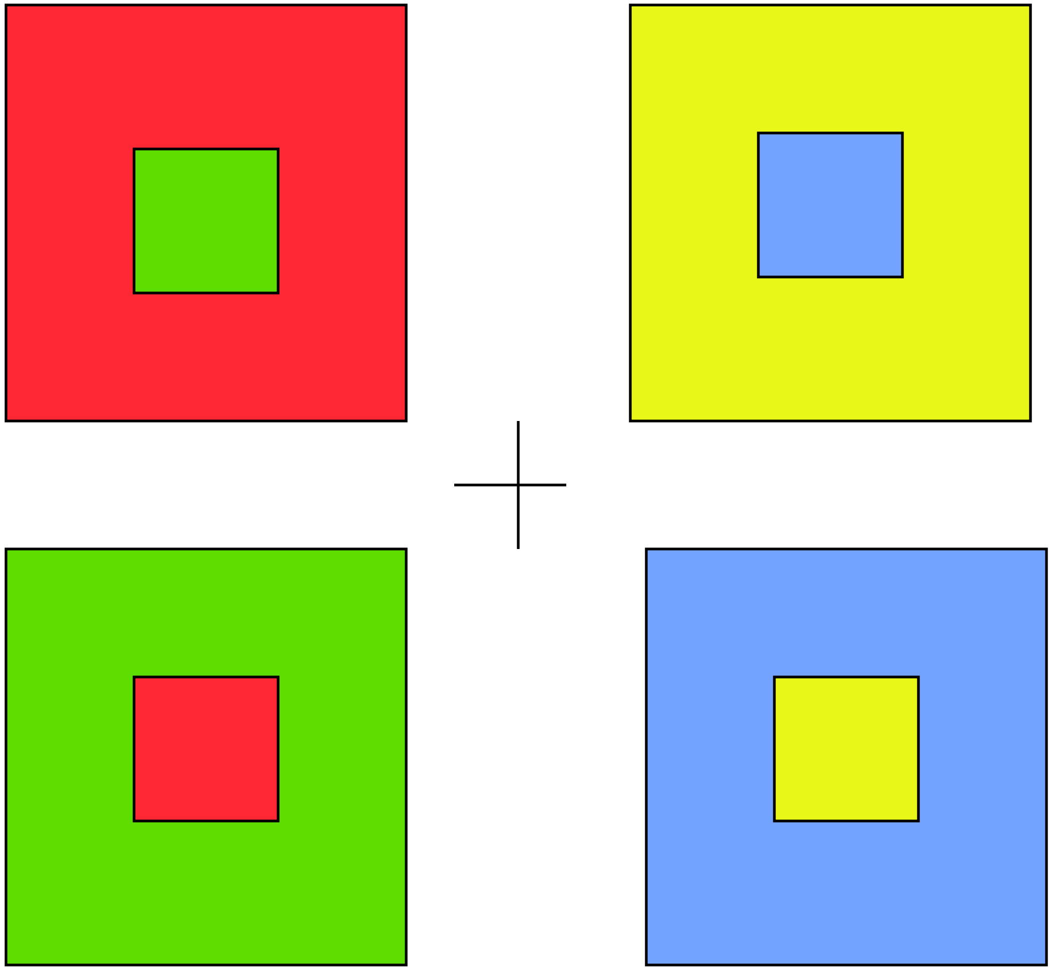
Examples

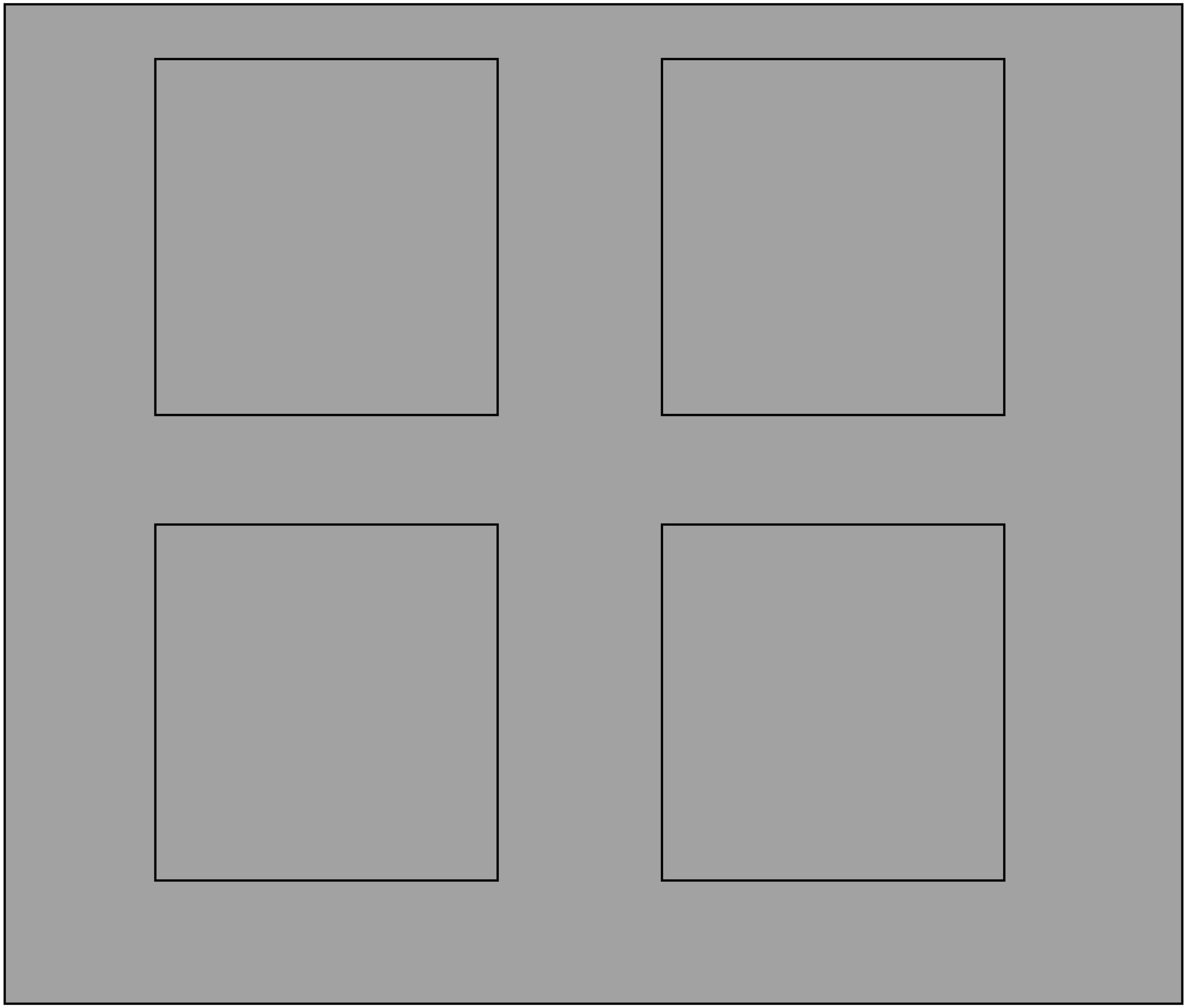






Even levels are components of 0's
The background is at level 0
Odd levels are components of 1's

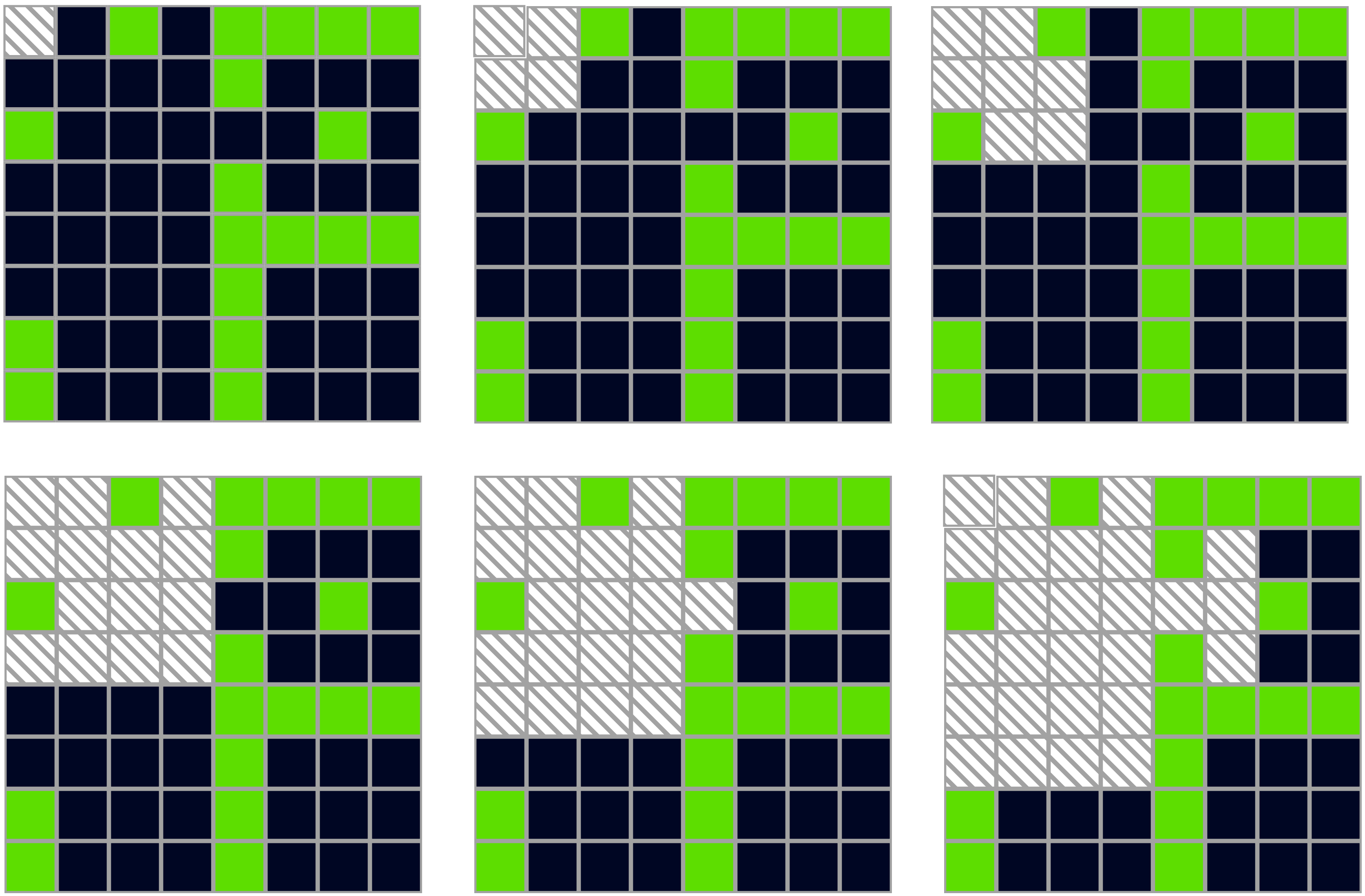




Component labeling

- 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



Disadvantages of recursive algorithm

- Speed
  - requires number of iterations proportional to the largest **diameter** of any connected component in the image
- Topology
  - not clear how to determine which components of 0's are holes in which components of 1's

 Zoran Duric

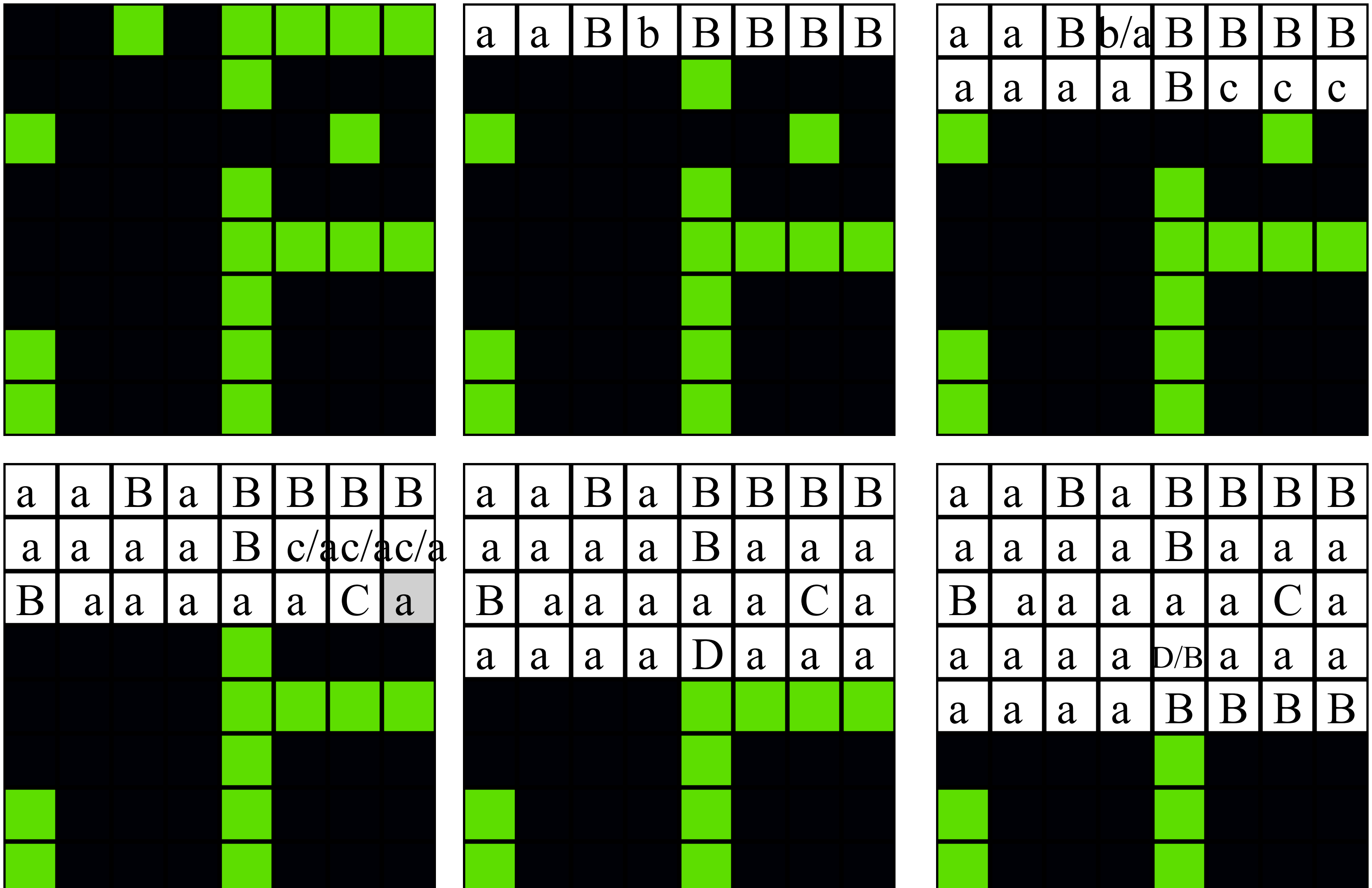
Solution 2 - row scanning up and down

- 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.

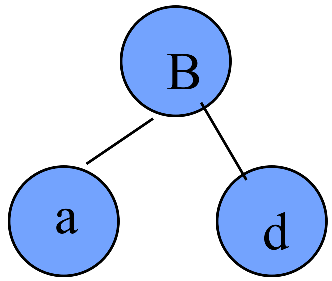
Example

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

• 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

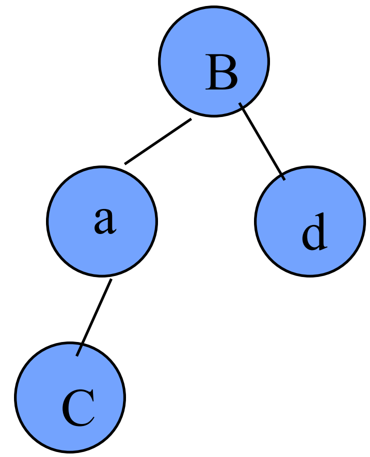


process row 4

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

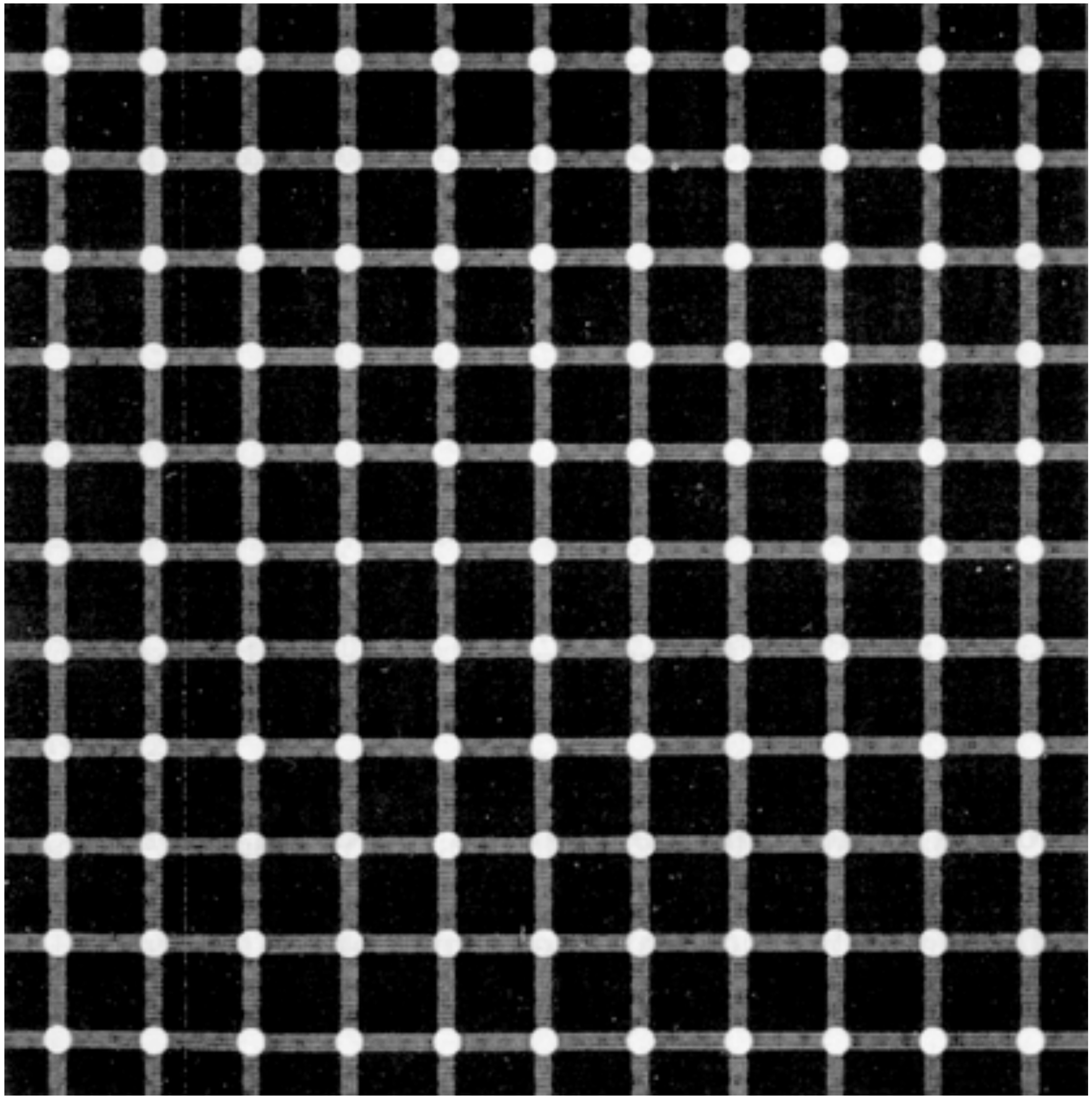
process row 3

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	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



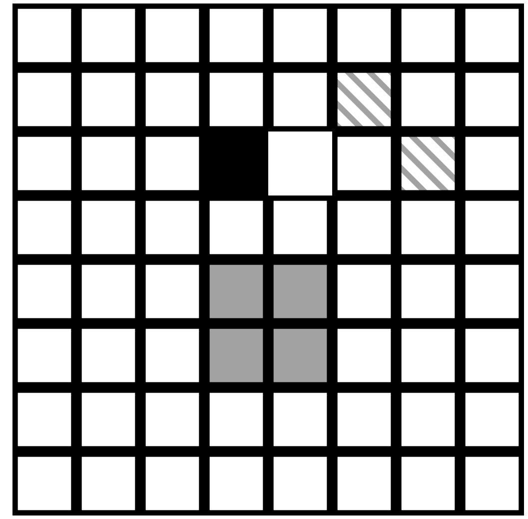
Properties

- 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.

Properties

- Area
- Perimeter
- Compactness: P^2/A
  - smallest for a circle: $4\pi^2r^2/\pi r^2 = 4\pi$
  - higher for elongated objects
- Properties of holes
  - number of holes
  - their sizes, compactness, etc.

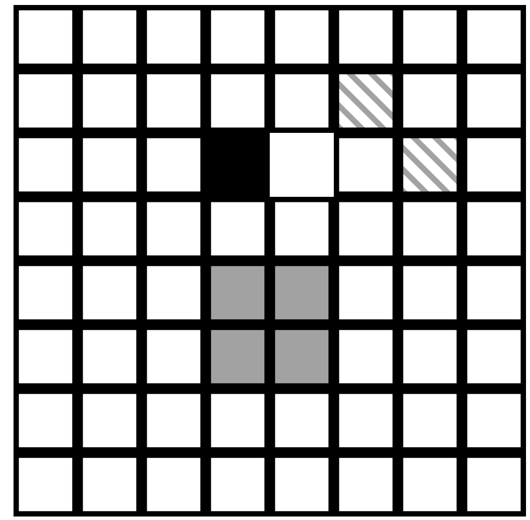
How do we compute the perimeter of a connected component?

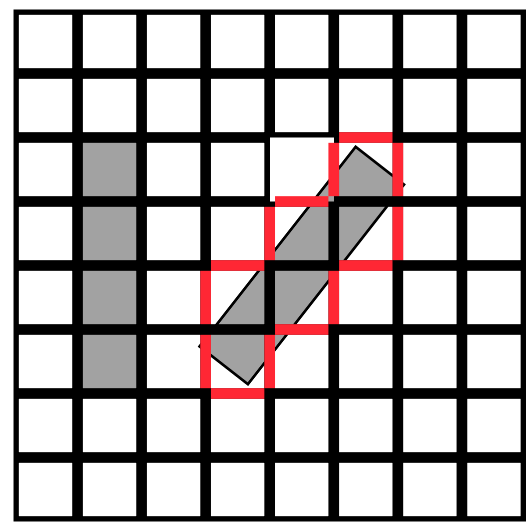


1. Count the number of pixels in the component adjacent to 0's
   - perimeter of black square would be 1
   - but perimeter of gray square, which has 4x the area, would be 4
   - but perimeter should go up as sqrt of area
2. Count the number of 0's adjacent to the component
   - works for the black and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

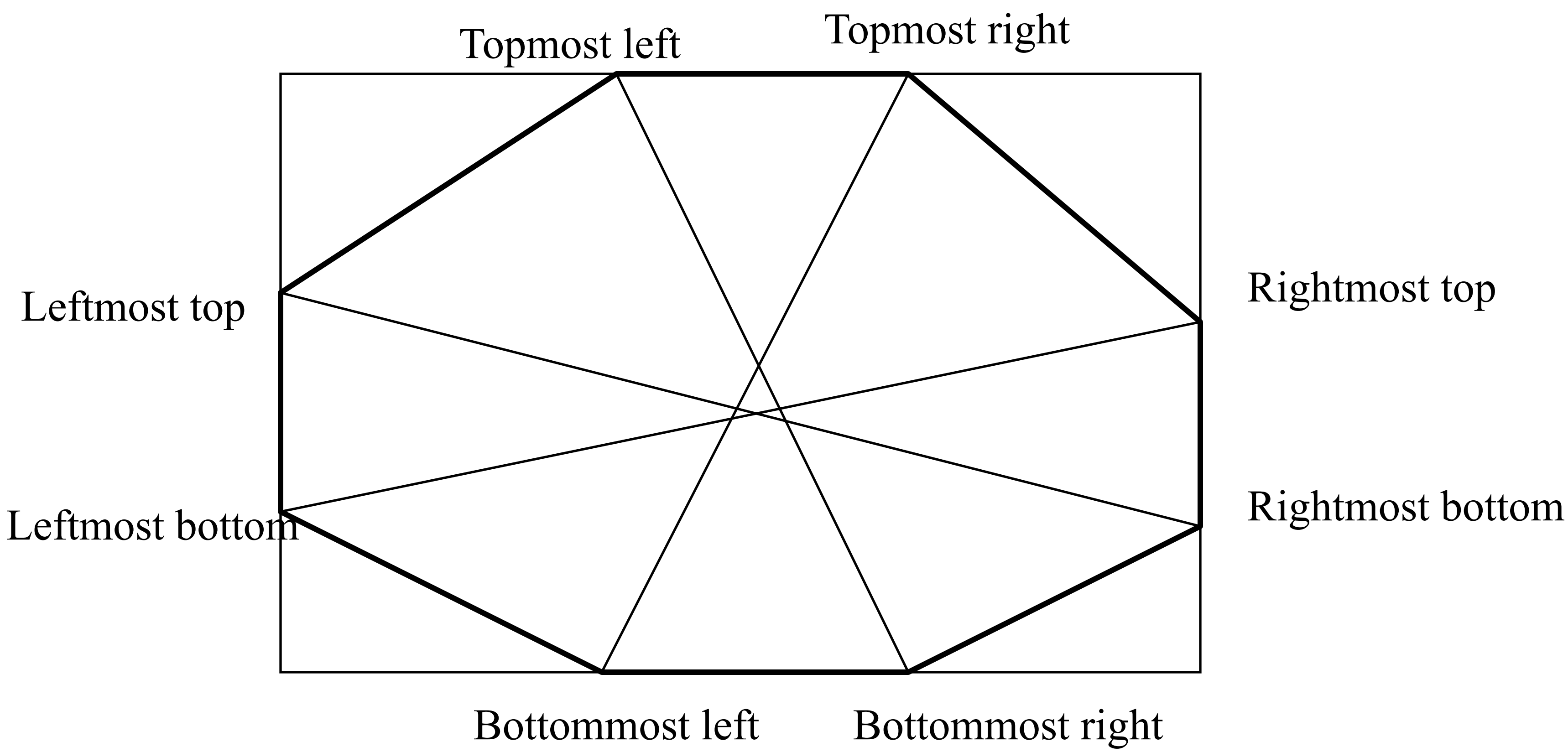




Perimeter computation (cont.)

- 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)

Bounding Box and Extremal Points



Other features

- Convex hull:
  - Create a monotone polygon from the boundary (leftmost and rightmost points in each row)
  - Connect the extremal points by removing all concavities (can be done by examining triples of boundary points)
- Minimal bounding box from the convex hull
- Deficits of convexity

A better (and universal)set of features

- 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

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

- The center of gravity of S can be expressed as

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

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

Zoran Duric

Central moments

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

$$\mu_{jk} = \sum (x - \bar{x})^j (y - \bar{y})^k$$

- 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

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

Central moments

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

$$\sigma_x = \sqrt{\frac{\mu_{20}}{|S|}}$$

$$\sigma_y = \sqrt{\frac{\mu_{02}}{|S|}}$$

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

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

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) 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

$$m_{jk}(S^*) = \frac{\sum\left(\frac{x^* - \overline{x(S^*)}}{\sigma_x(S^*)}\right)^j\left(\frac{y^* - \overline{y(S^*)}}{\sigma_y(S^*)}\right)^k}{|S|}$$

$$= \frac{\sum\left(\frac{a^j(x - \overline{x}(S))^j}{a^j\sigma_x{}^j(S)}\right)\left(\frac{a^k(y - \overline{y}(S))^k}{a^k\sigma_y{}^k(S)}\right)}{|S|}$$

$$= m_{jk}(S)$$

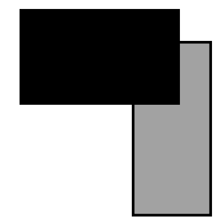
- Details of the proof are simple.

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

Agenda

- Advanced segmentation methods
  - edge detection
  - region recovery
- Occlusion in 2-D
  - correlation
  - clustering
- Articulations in 2-D
- Three dimensional object recognition
  - modeling 3-D shape
  - recognizing 3-D objects from 2-D images
  - recognizing 3-D objects from 3-D images
    - stereo
    - structured light range sensors