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

1

Image Segmentation: Definitions

- "Segmentation is the process of partitioning an image into semantically interpretable regions."
  - H. Barrow and J. Tennenbaum, 1978
- "An image segmentation is the partition of an image into a set of nonoverlapping regions whose union is the entire image. The purpose of segmentation is to decompose the image into parts that are meaningful with respect to a particular application." - R. Haralick and L. Shapiro, 1992

2

Image Segmentation: Definitions

- "The neurophysiologists' and psychologists' belief that figure and ground constituted one of the fundamental problems in vision was reflected in the attempts of workers in computer vision to implement a process called *segmentation*. The purpose of this process is very much like the idea of separating figure from ground ..." - D. Marr, 1982

3

Image Segmentation: Definitions

- "The partitioning problem is to delineate regions that have, to a certain degree, coherent attributes in the image. We will refer to this problem as the *image partitioning problem*. It is an important problem because, on the whole, objects and coherent physical processes in the scene project into regions with coherent image attributes. Thus, the image partitioning problem can be viewed as a first approximation to the *scene partitioning problem*..." - Y. LeClerc, 1989

4

Formal Definition

- Given region R and uniformity criterion U, define predicate P(R) = True, if $\exists\ a\ \ni\ |U(i,j) - a| < \varepsilon$, $\forall\ (i,j) \in R$
- Partition image into subsets R_i, $i = 1, ..., m$, such that
  - Complete: Image = $\cup R_i$, $i = 1, ..., m$
  - Disjoint subsets: $R_i \cap R_j = \varnothing$, $\forall\ i \neq j$
  - Uniform regions: $P(R_i)$ = True, $\forall\ i$
  - Maximal regions: $P(R_i \cup R_j)$ = False, $\forall\ i \neq j$

5

- Segmentation

  What criterion to optimize?

  - Low-level image properties: intensity, color, texture, motion
  - Intrinsic properties: depth, orientation, scene motion
  - Mid-level properties: symmetry, size
  - High-level properties: Object models

⇒ Optimization problem

But ...

- possibly multiple interpretations, so perhaps a Bayesian method is appropriate
- need efficient algorithms to solve optimization problem

6

Segmentation

- Segmentation as a clustering problem

  ⇒ use statistical clustering methods

  E.g.
  K-means
  Eigenvectors
  Graph-theoretic techniques

- Segmentation is application-specific

  ⇒ task determines models

  How many clusters?

  What is a good inter-cluster distance?

7

Segmentation by Clustering

Approach 1: Divisive Clustering
Partitioning
Splitting

Recursively divide image into more and more components

Approach 2: Agglomerative Clustering
Grouping
Merging

Starting from small components (small groups of pixels), recursively merge clusters

8

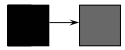
Major Issues in Clustering

- What is a good inter-cluster distance?
  - ⇒ define distance function and threshold; stopping criterion
- How many clusters are there?
- Which candidate clusters should be evaluated?
  - Combinatorial methods not feasible
  - Greedy methods
- What is an optimal solution?

9

Image Segmentation

- Ideally, object pixels would be black (0 intensity) and background pixels white (maximum intensity)
- But this rarely happens because
  - 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"



10

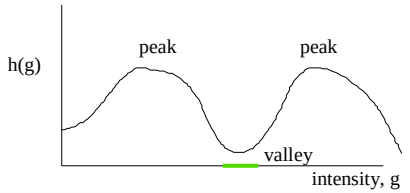
Image Segmentation by Thresholding

- 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

11

Thresholding

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



12

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)
    - Could also examine the binary images corresponding to alternative thresholds to choose a "best" one. E.g., one with straightest edges, most easily recognized objects, etc.

13

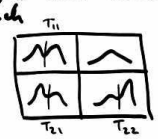
Thresholding

- **Mode (peak and valley) method**
  - Find the two most prominent peaks of h
    - *g* is a peak if $h_F(g) > h_F(g \pm \Delta g)$, $\Delta g = 1, ..., k$
  - 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 $h_F(g) < h_F(g')$, $\forall g, g' \in [g_1, g_2]$
  - Use *g* as the threshold
  - When image contains 2 normally-distributed classes, can prove that the probability of misclassification is minimized when *g* is at the minimum point

14

Adaptive Thresholding

- One common problem with the Mode Method is uneven illumination, causing no single threshold to be adequate
- Solution: Partition image into many subimages and select a threshold for each



- If a window is not sufficiently bimodal, interpolate threshold using neighboring subimages

15

Thresholding when Objects are not Uniform

- If regions or background don't have a characteristic range of gray levels ⇒ no threshold will separate
- Solution: Find a characteristic local property of gray levels and select a threshold for that property
- Example:

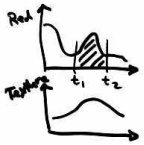
1. Compute local variance — i.e. for each pixel compute variance in 5×5 window
2. Regularization — local averaging of vals
3. Apply mode method to "smoothed property image"

16

Recursive Thresholding

- **Problem:** Multiple objects of varying size and varying gray level ranges
- **Algorithm:**

```
Region_set = {Image}
while Region_set ≠ ∅ do
    Select and remove a region R from Region_set
    Compute "property histograms" H_1, ..., H_k of R
    Select histogram H_i with "best peak" in non-unimodal histogram
    If "peakiness" > t then
        { choose thresholds, t_1, t_2, to isolate peak
          Apply thresholds to find regions R_1, ..., R_m within R
          Smooth boundaries of R_i's
          Add R_i's to Region_set }
    else Add R back to Region_set and mark it done
```



17

Segmentation by Region Growing

- Choose seed regions, $R_1, \ldots, R_n$
  while unclassified pixels remain do
    foreach region R_i do
      foreach pixel p in R_i with unclassified neighbors do
        merge "similar" unclassified neighbors of p into R_i
- Grows seed regions until they are maximally connected regions that satisfy a given property
- Uses heuristics of proximity and similarity

18

REGION GROWING USING SURFACE-FITTING (Best Fit)

- Regions are not constant wrt some property, but each region can be modeled by a single low-order polynomial function.
- Approximate each region by best fitting smooth surface
- Segmentation means finding union of simple surfaces that fit image
- Merge adjacent regions if a single planar or biquadratic function fits the data in new region

19

- Let possible surfaces be planar or biquadratic. Then model =

$$f(x,y,a,m) = \sum_{i+j \le m} a_{ij} x^i y^j \qquad 0 \le m \le 2$$

$$= a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2$$

$m=0 \Rightarrow$ constant: a_{00}

$m=1 \Rightarrow$ planar: a_{00}, a_{10}, a_{01}

$m=2 \Rightarrow$ biquadratic: 6D subspace

- (non) Homogeneity Predicate:

$$\text{Error}(R,a,m) = \sum_{(x,y)\in R} \left[g(x,y) - f(x,y,a,m) \right]^2$$

g: original image or "property" image

- Given R, can find m, a that minimizes $\text{Error}(R,a,m)$ using SVD

20

ALGORITHM

1. Detect initial "seed regions"
   Partition image into 7×7 blocks, R_i
   If Error$(R_i, a, 0) < T_1$ for a_{00} = avg. g.l.
   then accept region R_i

2. Find adjacent compatible points
   Foreach R_i find all compatible, adjacent points (x,y) ∋
   (x,y) neighbor of R_i
   $\left[g(x,y) - f(x,y,\underline{a},m)\right]^2 < \varepsilon$
   assoc. w/ current model for R_i

3. Increase order of model if necessary
   If no compatible points,
   m++ and goto Step 2 if $m \leq 2$
   ⇒ increase complexity of hypothesized surface

4. Merge compatible points to form new region
   Create $R_i^{(k+1)} = R_i^{(k)} \cup C_i^{(k)}$
   set of compatible (x,y)'s

5. Determine change in surface fit
   $\rho^{(k+1)} = Err(m, a^{(k+1)}, R_i^{(k+1)}) - Err(m, a^{(k)}, R_i^{(k)})$

   If $\rho^{(k+1)} < T_2$
   then goto step 2
   else
   m++
   if m > 2 then halt
   if $Err(R_i^{(k)}, a, m) - Err(R_i^{(k)}, a, m-1) < T_3$
   then goto step 2
   else halt

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 with respect to properties of resulting binary images
  - Local thresholding
    - Apply thresholding methods to image windows

An Advanced 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 and edge blur spread these spikes out into hills
- Make a parametric model of the shapes of the component histograms of the objects(s) and background

Kullback Information Distance

- 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

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

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

Kullback Information Distance

- Now, once we choose a threshold, *t*, then all of these unknown parameters are determined.
- 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, once *t* is chosen 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 *really* *f*(*g*)
- So, the question reduces to:
  - Determine a suitable way to measure the similarity of P_t and *f*
  - Find 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*))

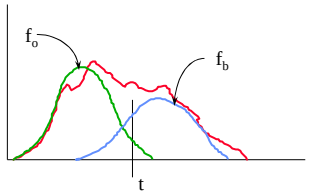
Another Threshold Selection Method: Minimize Probability of Error

- Using the same mixture model, we can find 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

29

Minimize Probability of Error

- For each threshold
  - Compute the parameters of the two Gaussians and the proportions
  - Compute the two probability of errors



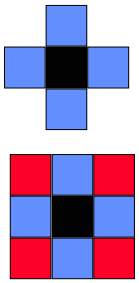
- Find the threshold that gives
  - Minimal overall error
  - Most equal errors

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

30

Object Extraction from Binary Images: Connected Components

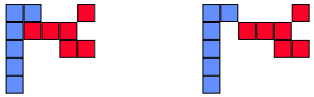
- *Definition*: Given a pixel (i,j) its **4-neighbors** are the points (i',j') such that |i-i'| + |j-j'| = 1
  - the 4-neighbors are (i±i, j) and (i,j±1)
- *Definition*: Given a pixel (i,j) its **8-neighbors** are the points (i',j') such that max(|i-i'|,|j-j'|) = 1
  - the 8- neighbors are (i, j±1),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



31

Adjacency

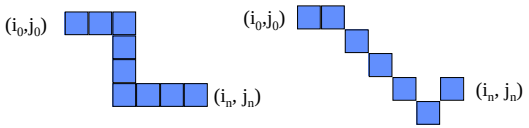
- Definition: Given two disjoint sets of pixels, S and T, S is **4-(8) adjacent** to T is there is a pixel in S that is a 4-(8) neighbor of a pixel in T



32

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



Every 4-path is an 8-path!

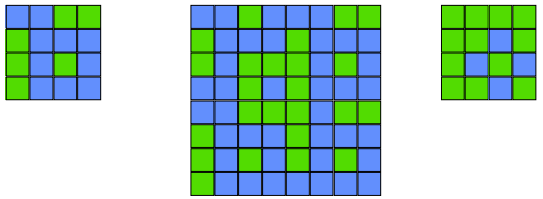
Connected Components

- *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 1
  - 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
  - Called **connected components**
- *Definition*: $\underline{S}$ is the complement of S - it is the set of all pixels in B whose value is 0
  - $\underline{S}$ 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 $\underline{S}$ that are adjacent to this frame is called the **background** of B
  - All other components of $\underline{S}$ are called **holes**

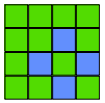
Examples: Blue = 1, Green = 0



How many 4- (8) components of S?
What is the background?
Which are the 4- (8) holes?
Jordan Curve Theorem: Any closed curve defines two connected regions

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



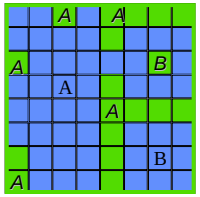
37

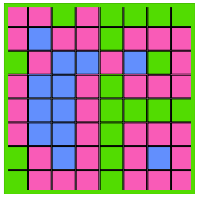
Boundaries

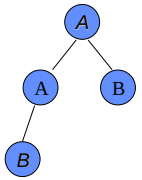
- The **boundary** of S if the set of all pixels of S that have 4-neighbors in S. The boundary set is denoted as S'
- The **interior** is the set of pixels of S that are not in its boundary: S-S'
- *Definition*: Region T **surrounds** region R (or R is **inside** T) if any 4-path from any point of R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

38

Examples







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

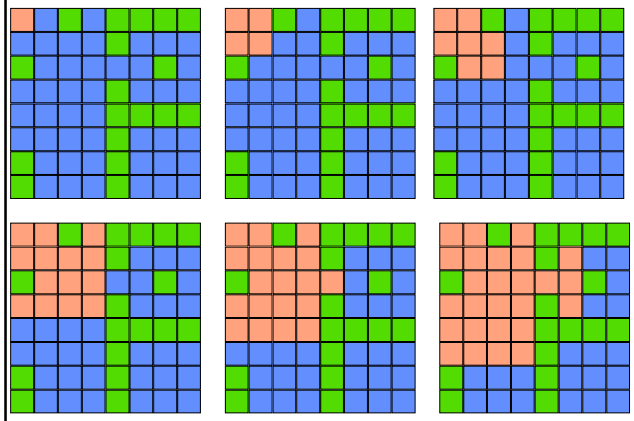
39

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

40

Example



41

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

42

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

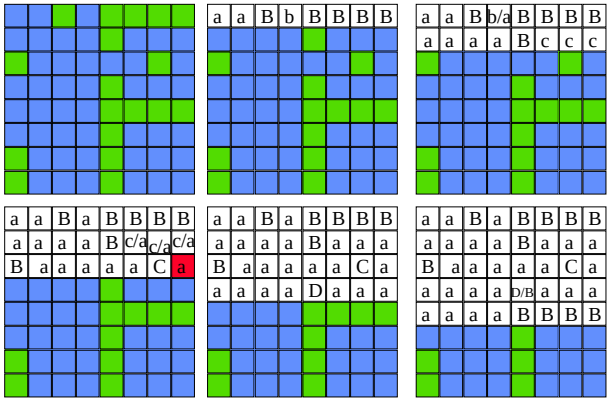
43

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

44

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

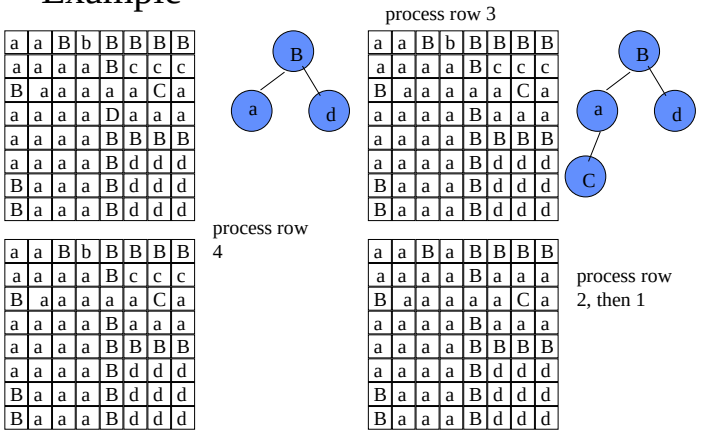
Upward Scan

- 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) is adjacent to *no* run of 1's (0's) on the subsequent row, and its label is *not* in the table
    - 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) *is* adjacent to one or more run of 1's on the subsequent row, it is marked with the common label of those runs, and the table properties are updated

Example



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.

2D Shape Representation

- Run-Length Encoding (RLE)
  - used for compression in TIFF format images
  - Exploit the fact that each image row of the form: $(0^*1^*)^*$
  - Method 1:
    For each row
    Represent each run of 1's as (start-col, length)

    Example:
    111 00 11 000 ...
    → (1,3), (6,2), ...

- Method 2:
  For each row
  Represent lengths of alternating runs of 0's and 1's
  (assume first run is 1's)

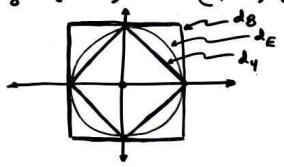
  Example:
  111 00 11 000 ...
  → 3, 2, 2, 3, ...

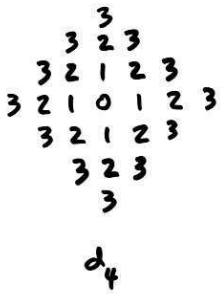
Medial-Axis Transform (MAT)

- 2D version of Run-length encoding in that here we describe maximal 2D disks instead of 1D "disks" (i.e., runs)
- 1. At each "1" find radius of largest disk centered at given pixel and entirely contained in 1's
  2. Retain only maximal points, i.e., points whose disks are not entirely contained inside another point's disk
  3. The set of maximal points and their radii are called the MAT (medial axis) (aka "grassfire transform")

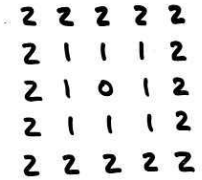
Digital Metrics

- Euclidean (L_2 norm)
  $d_E((p,q),(r,s)) = \sqrt{(p-r)^2 + (q-s)^2}$
- City Block (L_1 norm)
  $d_4((p,q),(r,s)) = |p-r| + |q-s|$
- Chessboard (L_∞ norm)
  $d_8((p,q),(r,s)) = \max(|p-r|, |q-s|)$





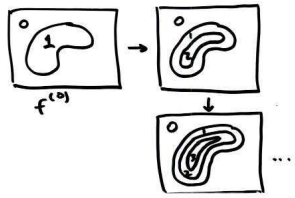
d_4



d_8

Distance Transformation

- One way to compute MAT is to first compute distance transform, i.e., for each "1" compute distance to closest "0"
- Parallel, iterative algorithm:
  $f^{(m)}(i,j) = f^{(0)}(i,j) + \min_{\substack{\text{8-neighbors} \\ (u,v) \text{ of} \\ (i,j) \text{ and } (i,j)}} f^{(m-1)}(u,v)$



- Another algorithm for computing the Distance Transform:

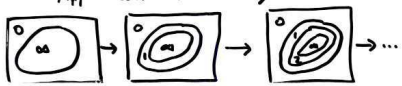
Let foreground = ∞'s
background = 0's

$$f^{(m)}(i,j) = \min_{-1 \le k,l \le 1} \left(f^{(m-1)}(i+k, j+l) + c(k,l) \right)$$

where $c =$

b	a	b
a	0	a
b	a	b

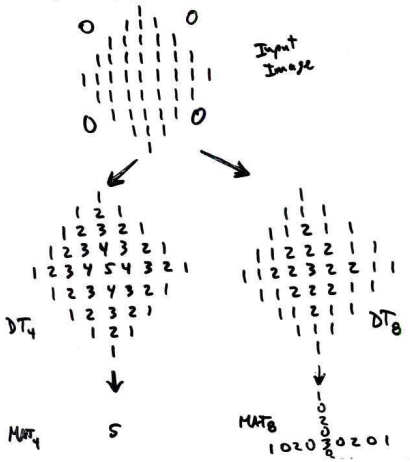
City Block: $a=1,\ b=\infty$
Chessboard: $a=1,\ b=1$
Euclidean: $a=1,\ b=\sqrt{2}$
Approx Euclidean: $a=3,\ b=4$



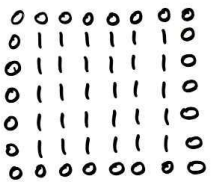
- Compute maximal points of MAT from Distance Transform:

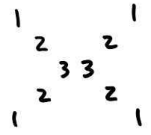
if $f^{(n)}(i,j) < f^{(n)}(u,v)$
for some 8-neighbor (u,v)
of (i,j) then (i,j) is <u>not</u>
a maximal point

Example



Example





City Block MAT

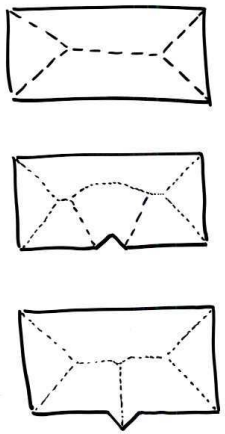
3 3

Chessboard MAT

MAT Properties

- Euclidean MAT is always connected for any connected shape
- City-Block and Chessboard and approximate Euclidean MAT may not be connected
- Invertible — exact reconstruction of shape possible
- Noise sensitive

61



62

Thinning

- Skeletonization procedure that preserves connectivity
- Goals:
  - "Peel away" boundary points
  - Never remove a point whose removal locally disconnects object
  - Never remove endpoints of arcs
- Not invertible — can't reconstruct object from thinned description

63

Thinning

- Consider a 3x3 neighborhood of a binary image in which the center pixel is a "1"
  - The center point is a **simple point** if changing it from a 1 to a 0 does *not* change the number of connected components of the 3x3 neighborhood

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

    - The first is 8-simple but not 4-simple
    - The second is neither 4 nor 8 simple
- Removal of a simple point will not change the number of connected components in a binary image
- An **end point** is a "1" with exactly one 1-neighbor

Thinning

- A 1-pixel (i,j) in a binary image is a **North border point** if pixel (i,j+1) is a 0
  - Similarly define East, West and South border points
- Simple thinning algorithm
  - For D = N,E,W,S do

    Eliminate all D border points that are simple points and *not* end points

```
0000000
0111100
0111100
0000000
```

- Must do the directions in sequence and not together or we could erase a component
- Result depends on the order in which the directions are considered

Example: 4-Simple Points

```
00000000      00000000      00000000
01011100      01001000      01000000
01011001      01011001      01011001
01111110  N   01011010  E   01010010
01111110  →   01111110  →   01111110
01100110      01100110      01000010
00000000      00000000      00000000
                  W
                  ↓
              00000000
              01001000
              01001001
              01001010
              01111110
              00100010
              00000000
```

Shock Graph

- Generalization of Blum's MAT
- Given curve $C_0(s) = (x_0(s), y_0(s))$

  s: parameter along curve
- Iteratively deform curve by moving each point some amount in some direction

- One definition:

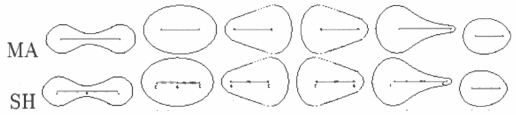
$$\begin{cases} \frac{\partial C}{\partial t} = (1 + \alpha \mathcal{K})\vec{n} \\ C(s,0) = C_0(s) \end{cases}$$

t: time, $\mathcal{K}$: curvature, $\vec{n}$: inward normal

67

- 2 terms determine local deformation
  - 1st term: constant deformation (constant motion along normal)
  - 2nd term: curvature deformation
- α is a constant (≥ 0) that controls the regularizing effects of curvature $\mathcal{K}$
  - α large $\Rightarrow$ geometric heat equation
  - $\alpha = 0 \Rightarrow \equiv$ MAT
- At each "shock point" label it according to the local variation of the radius function along the axis

68

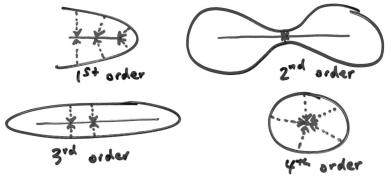


Medial Axis: Locus of maximal disks

Shock Set: Attach a sense of flow to each point

69

- 4 types of shock points:



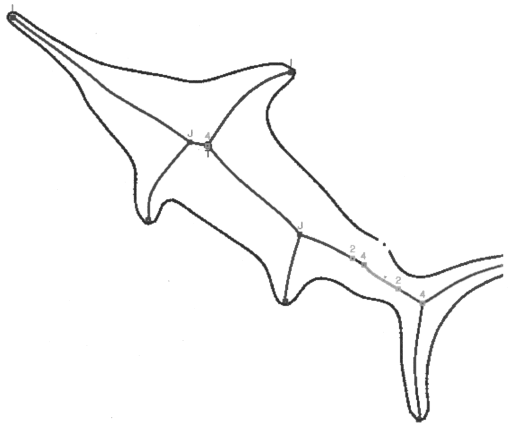
1st order: 1-shock — Derives from a "protrusion" ⇒ radius function changes monotonically

2-shock: Occurs at a "neck" ⇒ local minimum of radius function

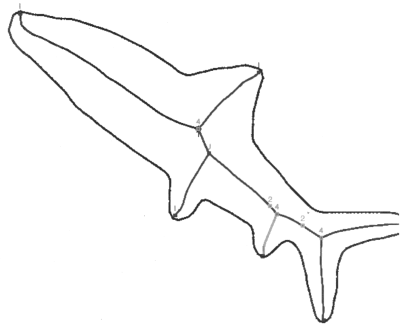
3-shock: "Bead with parallel sides" ⇒ constant radius function

4-shock: "point" ⇒ local maximum of radius function

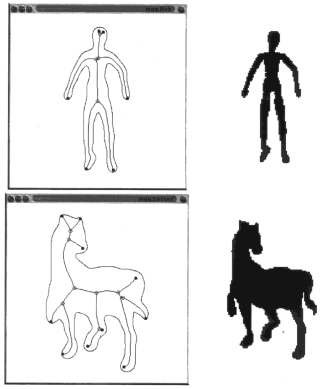
70



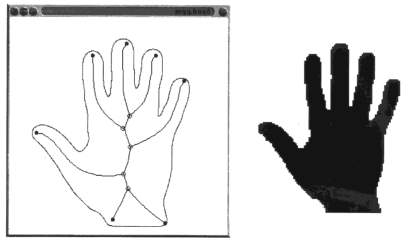
71



72



73



74

Shock Point Labeling Algorithm

Let $R(x)$ = radius of largest disk at x

$N(x,\varepsilon)$ = other shock points (not x) in ε-neighborhood of x

If $\exists \varepsilon > 0$ s.t. $R(x) > R(y)$ $\forall y \in N(x,\varepsilon)$
then label x as a 4-shock pt

else if $\exists \varepsilon > 0$ s.t. $R(x) = R(y)$ $\forall y \in N(x,\varepsilon)$
and $N(x,\varepsilon) \neq \emptyset$
then label x as a 3-shock pt

else if $\exists \varepsilon > 0$ s.t. $R(x) < R(y)$
$\forall y \in N(x,\varepsilon)$ **and** $N(x,\varepsilon) \neq \emptyset$
and $N(x,\varepsilon)$ not connected
then label x a 2-shock pt

else label x a 1-shock pt

75

Next, convert shock points into a shock graph

- Algorithm: "reverse" grassfire evolution by starting at 4-shock points

76

Shock Graph Construction

1. Compute Euclidean Distance Transform, D
2. Compute gradient vector field, ∇D
3. Compute divergence of ∇D over a neighborhood of each point

Flux $\equiv 0$ $\Rightarrow$ ~ constant gradient vectors

Flux < 0 $\Rightarrow$ "sink" of energy i.e., skeletal points

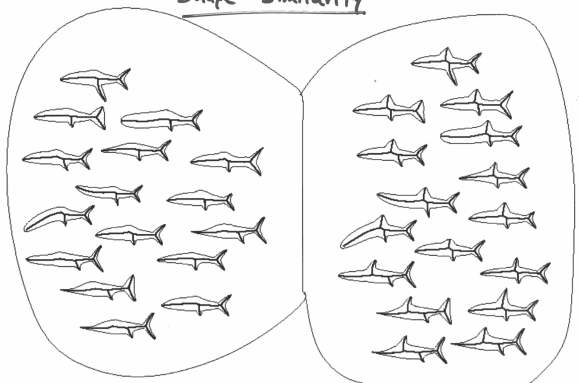
Flux > 0 $\Rightarrow$ "source" of energy i.e., skeletal points of background

77

4. Thin
   * without disconnecting skeleton
   * based on flux values

78

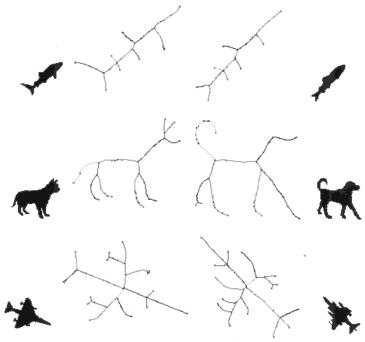
Shape Similarity



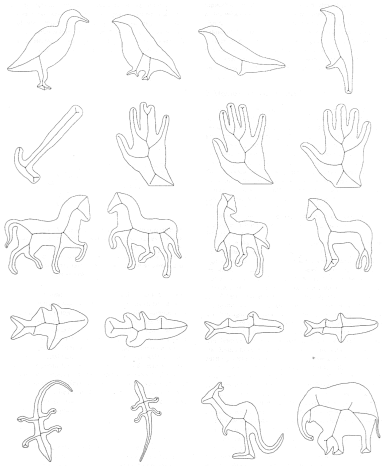
Measure similarity in terms of shock graph attributes and topology changes

79

Shape Matching by Shock Graph "Edit Distance"



80



81

Quadtrees

- Represent a connected component as a union of maximal, non-overlapping disks whose sizes must be powers of 2, and whose positions must be at powers of 2 coordinates
- Variable resolution representation

82

Quadtree Construction

Input: $2^n \times 2^n$ binary image

Output: 4-ary tree

Algorithm:

1. Create root node and associate input image with it
2. If current node's image is all 1's (0's) then {label node Black (white) leaf; halt }
   else {create 4 children nodes of current node and subdivide current node's $2^k \times 2^k$ image into 4 $2^{k-1} \times 2^{k-1}$ quadrants; Goto step 2 for each child }

NW	NE
SW	SE

83

FIG. 1. AN IMAGE, ITS MAXIMAL BLOCKS, AND THE CORRESPONDING QUADTREE. BLOCKS IN THE IMAGE ARE SHADED.

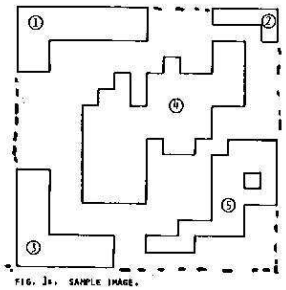


FIG. 1a. SAMPLE IMAGE.

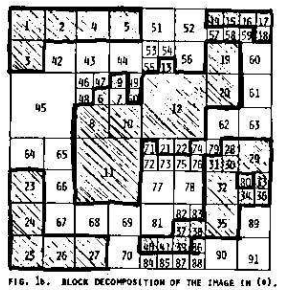
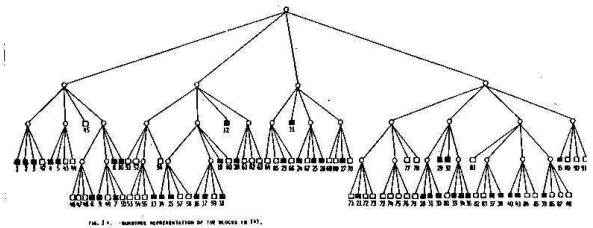


FIG. 1b. BLOCK DECOMPOSITION OF THE IMAGE IN (a).



85

FIND NEIGHBOR

```
node procedure EQ_ADJ_NEIGHBOR (P, D);
/* locate equal-sized neighbor of P in direction D */
begin
   value node P;
   value direction D;
   return (SON (if ADJ(D, SONTYPE(P))
                then EQ_ADJ_NEIGHBOR(FATHER(P),D)
                else FATHER(P),
           REFLECT (D, SONTYPE(P))));
end;
```

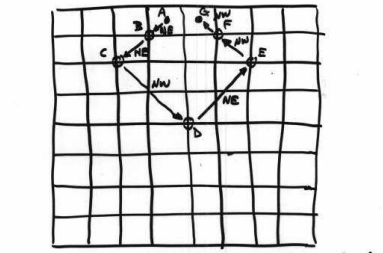
ADJ(S,Q)

S \ Q	NW	NE	SW	SE
N	T	T	F	F
E	F	T	F	T
S	F	F	T	T
W	T	F	T	F

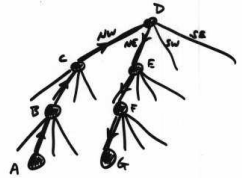
REFLECT(S,Q)

S \ Q	NW	NE	SW	SE
N	SW	SE	NW	NE
E	NE	NW	SE	SW
S	SW	SE	NW	NE
W	NE	NW	SE	SW

86



FINDING THE EASTERN NEIGHBOR of A

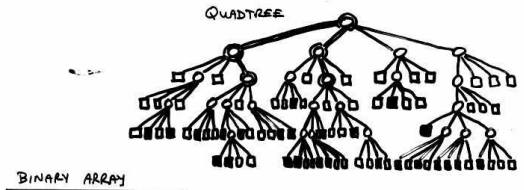


87

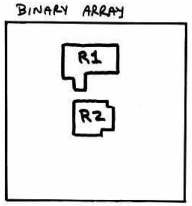
TRACE of EQ_ADJ_NEIGHBOR

→ EAN (A, East)
  Sontype (A) = NE ⇒ ADJ(East, NE) = T
→ EAN (B, East)
  Sontype (B) = NE ⇒ ADJ(East, NE) = T
→ EAN (C, East)
  Sontype (C) = NW ⇒ ADJ(East, NW) = F
← RETURN: SON(D, REFLECT(East, NW))
  NE
  E
← RETURN: SON(E, REFLECT(East, NE))
  NW
  F
← RETURN: SON(F, REFLECT(East, NE))
  NW
  G

88



BINARY ARRAY



DISADVANTAGES of QUADTREES

- QUADTREE SIZE DEPENDS ON OBJECT POSITION & IMAGE SIZE
- REPRESENTATION of MULTIPLE OBJECTS ⇒ MANY NODES AND EDGES ARE SHARED.
- BOTTLENECK AT ROOT ⇒ PARALLEL ALGS NO FASTER THAN SEQUENTIAL ALGS.

89

2D Boundary Representation

- Any connected component is completely determined if we know its borders
- 4-border of S = set of points of S that have at least 1 4-neighbor in $\bar{S}$
- A point may be on more than 1 border:

- Outer border = set of border points adjacent to background (closed curve)
- Hole border = closed curve of border pts adjacent to a hole.

90

Edge-Crawl Algorithm

- Trace the 4-border of an 8-connected component C given border point P_0 and 4-adjacent neighbor $q_0 \in \bar{C}$

Step 1: $i = 0$

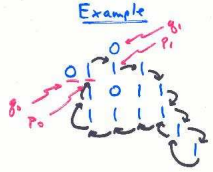
Step 2: // Let neighbors of p_i in clockwise order starting at q_i be $q_i = r_1, r_2, \ldots, r_8$ //

for ($j = 2$ until 8) do
  if $r_j \in C$ then goto Step 3
endfor
halt; // C = 1 pixel only

Step 3: $p_{i+1} = r_j$ // $r_j \in C$
$q_{i+1} = r_{j-1}$ // $r_{j-1} \notin C$
if ($p_0 == p_{i+1}$ and $q_0 \in \{r_1, \ldots, r_j\}$)
  then halt
  else goto Step 2

91

Example



- Consecutive p_i's are 8-neighbors
- Instead of representing coords of p_i's, represent direction to next border point

3	2	1
4	P_i	0
5	6	7

ith neighbor in direction $45i°$ rel. to positive x-axis

⇒ 00767733444 22

Chain Code

92

- Why termination test includes g_0

or A border point may occur multiple times on a given border

- Perimeter = $(n_0 + n_2 + n_4 + n_6) + \sqrt{2}(n_1 + n_3 + n_5 + n_7)$

  where n_i = # occurrences of link i

- Chain code is translation invariant but not rotation invariant

93