Region Growing - Splitting

- Segmentation can never be perfect
  - there are extra or missing regions
- Correct the results of segmentation
  - delete extra regions or
  - merge regions with others
  - split regions into more regions
- Correction criteria:
  - significance (e.g., size)
  - homogeneity (e.g., uniformity of gray-level values)

Data Structures

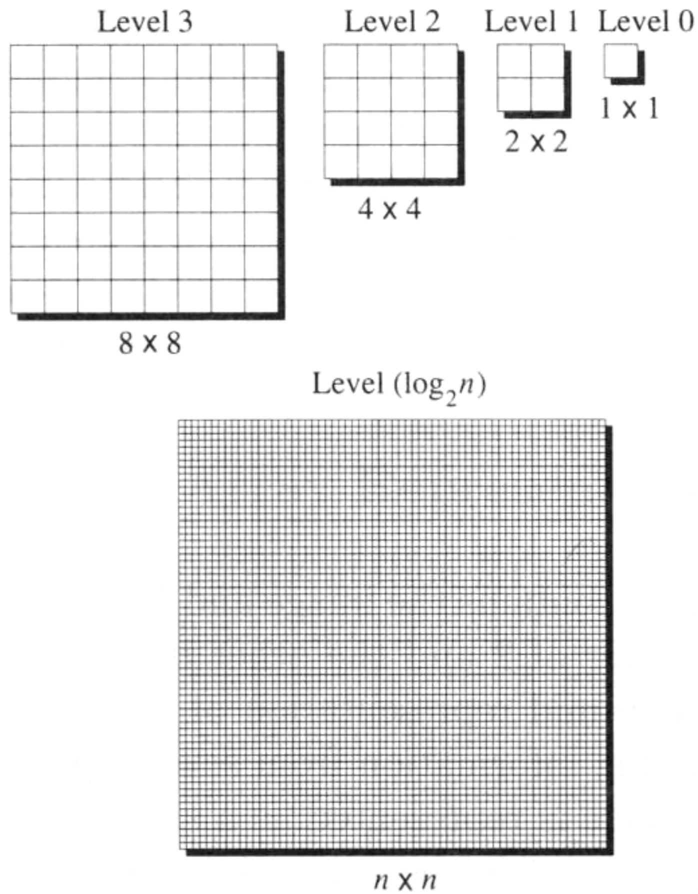
- Represent the results of a segmentation
  - array representations (e.g., image grid)
  - hierarchical representations (e.g., pyramids, Quad Trees)
  - symbolic representation (e.g., moments, Euler number)
  - Region Adjacency Graphs (RAGs)
  - Picture Trees
  - edge contours

1. Image Grid

b	b	b	a	a	a	a	a
b	b	b	b	a	a	a	c
b	b	b	a	a	a	c	c
b	b	a	a	c	c	c	c
b	a	a	a	a	c	c	d
a	a	a	c	a	c	c	d
a	b	a	a	c	c	d	d
a	a	a	a	c	d	d	d

2. Pyramid

- Hierarchical representation: the image at k degrees of resolution
  - nxn image, n/2xn/2, n/4xn/4,…, 1x1 images
- A pixel at level i represents aggregate information from 2x2 neighborhoods of pixels at level i + 1
  - image is a single pixel at level 0
  - the original image is represented at level k-1

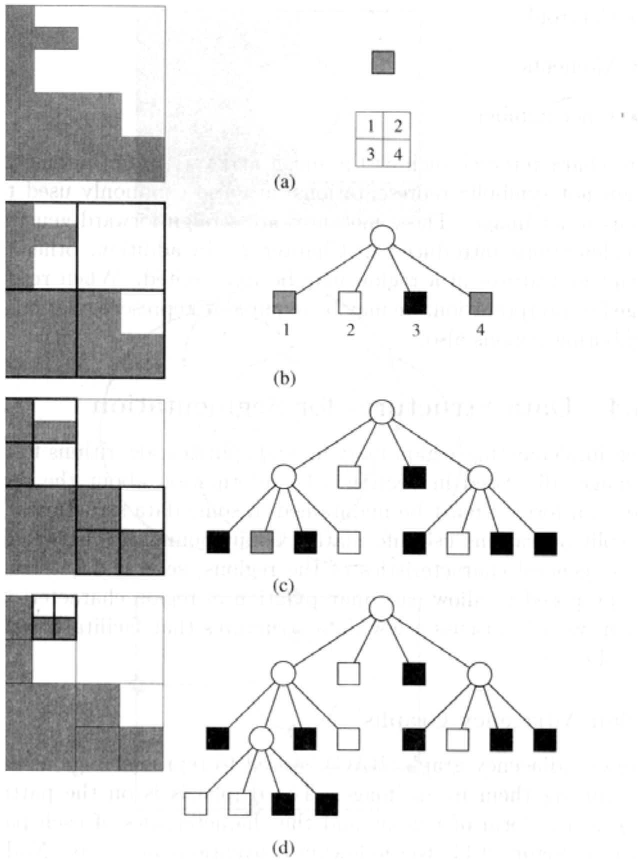

Level 3
Level 2
Level 1
Level 0
1 x 1
2 x 2
4 x 4
8 x 8
Level ($\log_2 n$)
$n \times n$

3. Quad Trees

- Hierarchical representation
  - a node represents blocks of white, black or grey pixels
  - blocks of grey may contains mix of both white and black pixels
- Obtained by recursive splitting of an image
  - each region is split into 4 sub-regions of identical size
  - each gray region is split recursively as long as it is grey
  - white or black regions are not split further

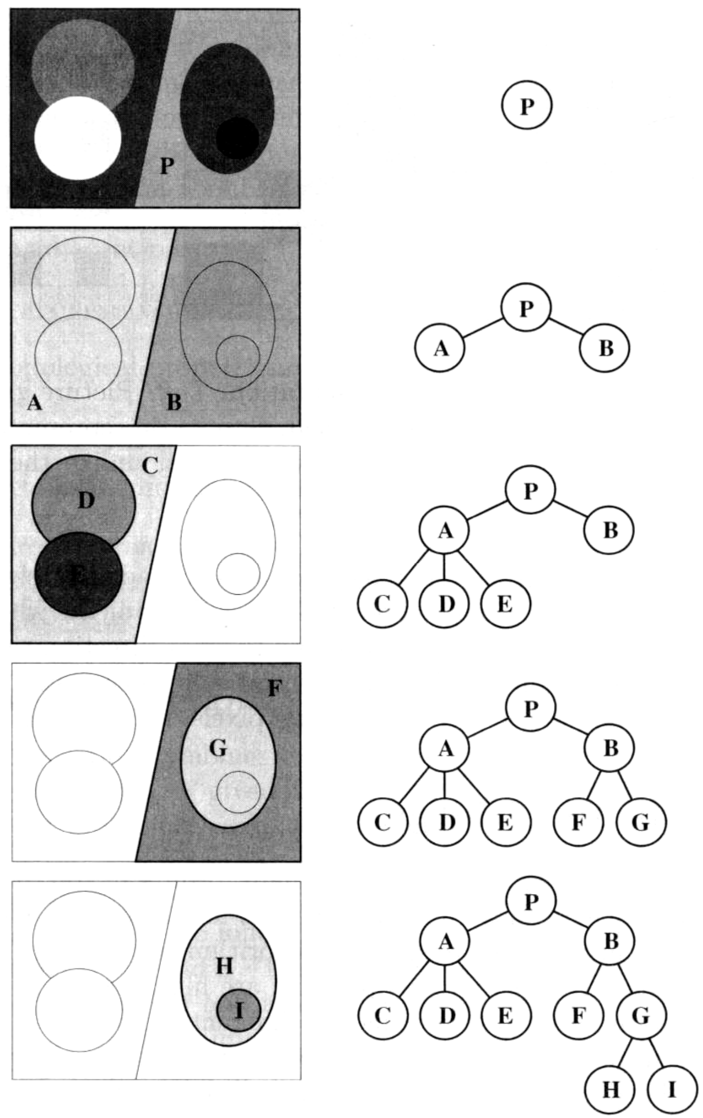
Quad Tree Example

- Original grey image
- Split of **a** into 4 regions
- Split b grey regions; one is still grey
- Spit last c grey region → final quad tree



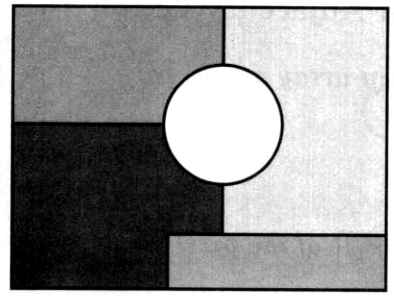
4. Picture Tree

- Emphasis on nesting regions
- A picture tree is produced by recursively splitting the image into component regions
- Splitting stops when with only uniform regions has been produced

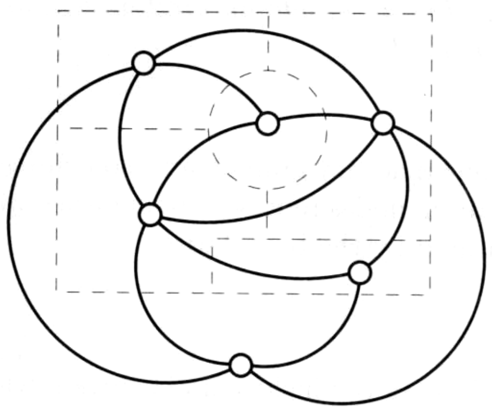

P
A
B
C
D
E
F
G
H
I

5. Region Adjacency Graphs (RAGs)

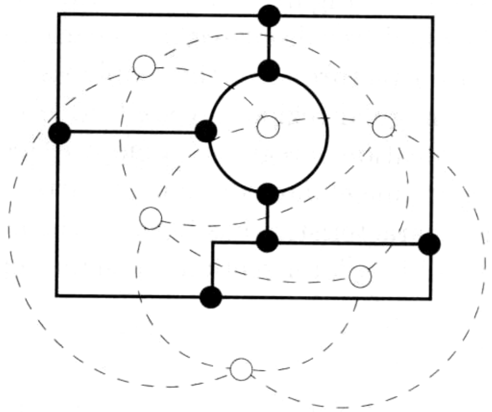
- Adjacency relationships between regions
  - graph structures
  - nodes represent regions (and their features – see symbolic representations)
  - arcs between nodes represent adjacency between regions
- Dual RAGs: nodes represent boundaries and arcs represent regions separated by these boundaries



segmented image



Region Adjacency Graph (RAG)



Dual RAG

RAG Algorithm

- Create RAG from segmented image

1. take a region R_i
2. create node n_i
3. for each neighbor region R_j of R_i create node n_j
4. connect n_i with n_j
5. repeat steps 3-4 for each region until all regions have been considered

6. Symbolic representations

- Each region is represented by a set of features
  - Bounding Enclosing Rectangle
  - Orientation, Roundness
  - Centroid, First, Second and Higher order Moments
  - Euler Number
  - Mean and variance of intensity values
  - Relative distance, orientation, adjacency, overlapping etc.

Region Merging

- Two or more regions are merged if they have similar characteristics
  - mostly intensity criteria (mean intensity values)
  - more criteria can be applied
  - boundary criteria
  - combination of criteria

Region Merging algorithm

- Input: a segmented image and its RAG

1. for each region R_i (node n_i)
   a. take its neighbor regions R_j (node n_j)
   b. if similar* → merge them to one
   c. update RAG (delete one of n_i, n_j and its arcs)
2. repeat until no regions are merged

* Similarity Criterion: similar average intensities e.g., $|\mu_i-\mu_j| < \varepsilon$, contour continuity etc.

Statistical Criterion for Region Similarity

- Input: region R_1 with m_1 points and region R_2 with m_2 points
- Output: determines whether they should be merged or not
  - assumption: image intensities are drawn from a Gaussian distribution

$$p(g_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(g_i-\mu)^2}{2\sigma^2}}$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} g_i \qquad \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(g_i - \mu)^2$$

1. Statistical Criterion: Case H_0

- Regions R_1, R_2 must be merged to form a single region
  - the intensities of the new region are drawn from a single Gaussian distribution (μ_0,σ_0)

$$P_0 = P(g_1, g_2, \ldots, g_{m_1+m_2} \mid H_0) =$$

$$\prod_{i=1}^{m_1+m_2} P(g_i \mid H_0) = \prod_{i=1}^{m_1+m_2} \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{(g_i-\mu_0)^2}{2\sigma_1^2}} =$$

$$\frac{1}{(\sigma_1\sqrt{2\pi})^{m_1+m_2}} e^{-\frac{\sum_{i=1}^{m_1+m_2+2}(g_i-\mu_0)^2}{2\sigma_0^2}} = \frac{1}{(\sigma_0\sqrt{2\pi})^{m_1+m_2}} e^{-\frac{m_1+m_2}{2}}$$

2. Statistical Criterion: Case H_1

- R_1, R_2 should not merge
  - their intensities are drawn from two separate Gaussian distributions (μ_1,σ_1), (μ_2,σ_2)

$$P_1 = P(g_1, g_2, \ldots, g_{m_1} \mid H_0, g_{m_{1+1}}, \ldots, g_{m_1+m_2} \mid H_1) =$$

$$P(g_1, g_2, \ldots, g_{m_1} \mid H_0) P(g_{m_{1+1}}, \ldots, g_{m_1+m_2} \mid H_1) =$$

$$\frac{1}{(\sigma_1\sqrt{2\pi})^{m_1}} e^{-\frac{(g_i-\mu_1)^2}{2\sigma_1^2}} \frac{1}{(\sigma_2\sqrt{2\pi})^{m_2}} e^{-\frac{(g_i-\mu_2)^2}{2\sigma_2^2}} = \frac{1}{(\sigma_1\sqrt{2\pi})^{m_1}} e^{-\frac{m_1}{2}} \frac{1}{(\sigma_2\sqrt{2\pi})^{m_2}} e^{-\frac{m_2}{2}}$$

3. Statistical Criterion: Decision

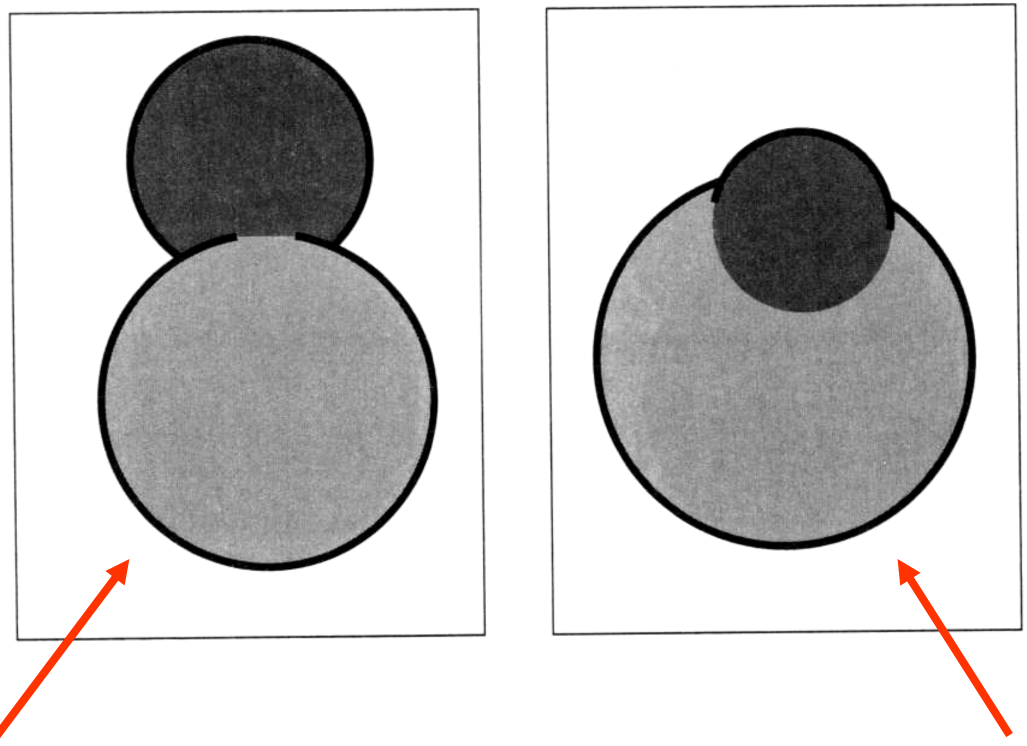
- If the ratio L is below a threshold, there is strong evidence that R_1,R_2 should be merged

$$L = \frac{P_1}{P_0} =$$

$$\frac{P(g_1, g_2, \ldots, g_{m_1} \mid H_0, g_{m_{1+1}} \ldots, g_{m_1+m_2} \mid H_1)}{P(g_1, g_2, \ldots, g_{m_1+m_2} \mid H_0)} = \frac{{\sigma_0}^{m_1+m_2}}{{\sigma_1}^{m_1} {\sigma_2}^{m_2}}$$

Region Merging With Boundary Criteria

- Two regions should merge if the boundary between them is weak
- Two Criteria:
  - the weak boundary is small compared to the boundary of the smaller region
  - the weak boundary is small compared to the common boundary



the regions should not be merged because the weak boundary is very short compared to the boundary of the smaller region

the two regions should be merged because the weak boundary is large compared to the boundary of the smaller region

Region Splitting

- If a region is not homogeneous (uniform) it should be split into two or more regions
- Large regions are good candidates for splitting
  - e.g., start from the entire image as input
  - intensity criteria (variance of intensity values)
  - a problem is to decide how and where to split
  - usually a region is split into n equal-sized parts

Region Splitting Algorithm

- Input: initial segmentation and RAG or Quad Tree
  - for each region R_i in the image recursively perform the following steps
    - compute the variance σ_i of the intensities of R_i
    - if $\sigma_i > \varepsilon^*$ split the region into n^* equal parts
    - update RAG or Quad Tree

* ε, n are user defined

Split and Merge

- Combination of Region Splitting and Merging for image segmentation

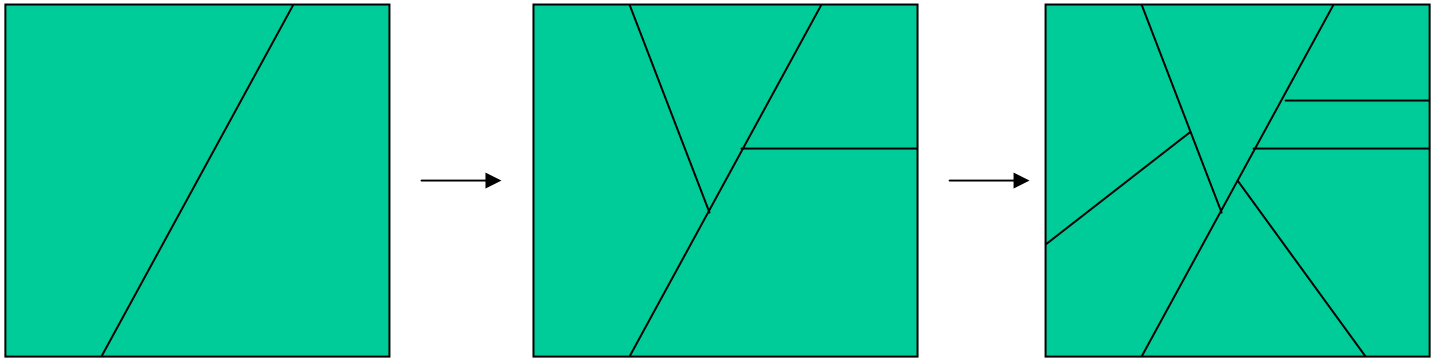
1. for each region R (or entire image)
   a. if R is not uniform split it into 4 equal parts
   b. update the RAG or Quad Tree
2. for each group of (e.g, 2 or 4) regions
   a. if merging criteria are met
   b. merge the regions
   c. update the RAG or Quad Tree
3. repeat steps 1, 2 until no regions are merged or split

More Segmentation Algorithms

- "Adaptive Split and Merge Segmentation Based on Least Square Piecewise Linear Approximation", X. Wu, IEEE Trans. PAMI, No. 8, pp. 808-815, 1993
- K-means Region Segmentation Algorithm
- Hough Transform (find lines, circles, known shapes in general)
- Relaxation Labeling (edge, region segmentation)

Adaptive Split and Merge Segmentation Based on Least Square Piecewise Linear Approximation

- Basic Idea: Successive region splitting in many directions until some homogeneity criterion is met



1. Adaptive Split Criteria

- Let $G=g(x,y)$ be the original image
- G is split into k regions $G_1,G_2,\ldots G_k$
  - produce k homogeneous regions
  - minimize a global criterion of homogeneity

$$\sum_{i=1}^{k} E(G_i) = \sum_{i=1}^{k} \sum_{x,y \in G} \{g(x,y) - \mu(G_i)\}^2$$

$$\mu(G_i) = \frac{\sum_{x,y \in G_i} g(x,y)}{\|G_i\|}$$

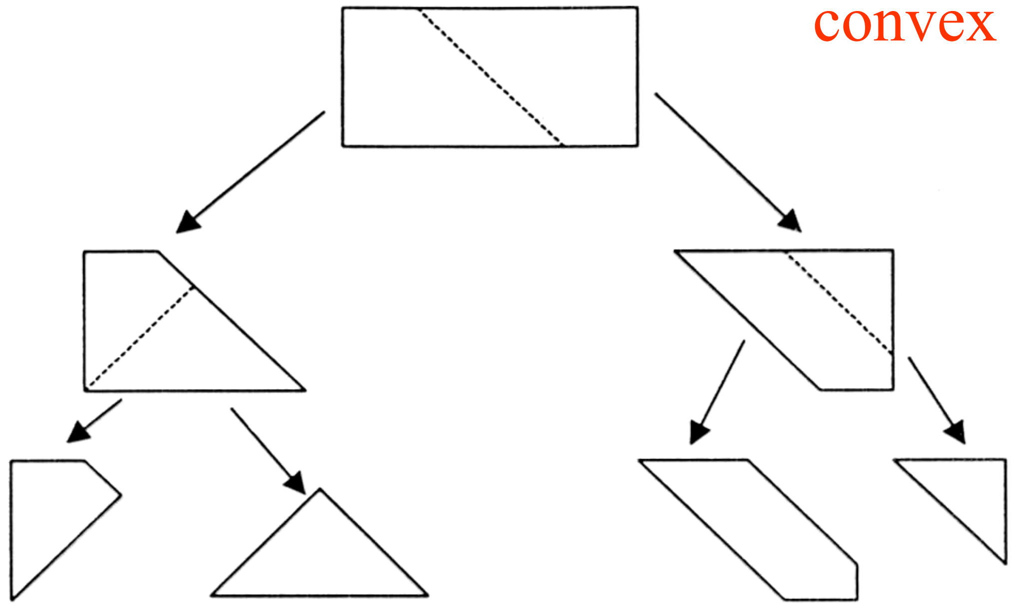
2. Adaptive Split Satisfaction

- There are too many ways to split a region into sub-regions
  - accept only horizontal, vertical, 45^o and 135^o split directions
  - split at two directions at a time

$$E(G) = \sum_{x,y \in G} \{g(x,y) - \mu(G)\}^2 > \varepsilon$$

- Every region is split as long as
  - ε is user defined
  - at the end either $E(G) < \varepsilon$ or G is one pixel

splitting produces convex regions



Recursive Optimal Four Way Split (ROFS) Algorithm

Function ROFS(G) {

If $E(G) < \varepsilon$ then return (G)

else {

partition G into G_1 and G_2 by minimizing

$$\sum_{i=1}^{k} E(G_i) = \sum_{i=1}^{k} \sum_{x,y \in G} \{g(x,y) - \mu(G_i)\}^2$$

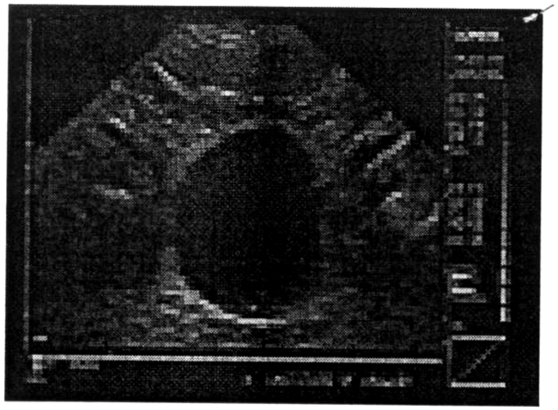
over all possible 45 $*$ i degree cuts, i = 0,1,2,3

ROFS(G_1)

ROFS(G_2)

}

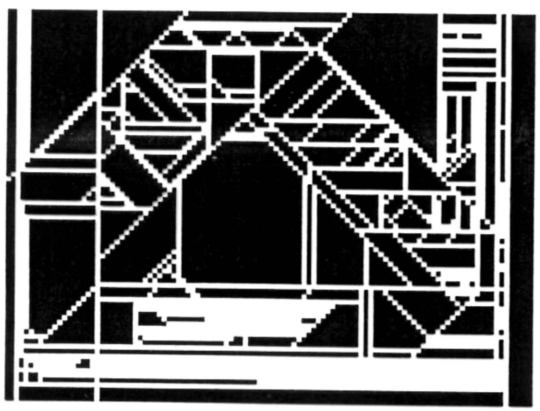
}



initial image

the number of polygons which are produced depends on ε

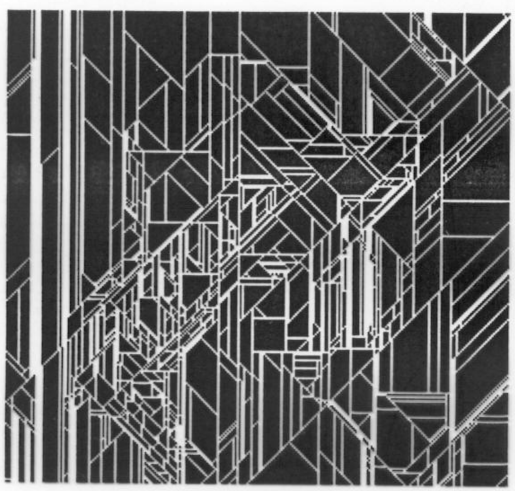
$\varepsilon_1 > \varepsilon_2$



243 polygons

1007 polygons

initial image

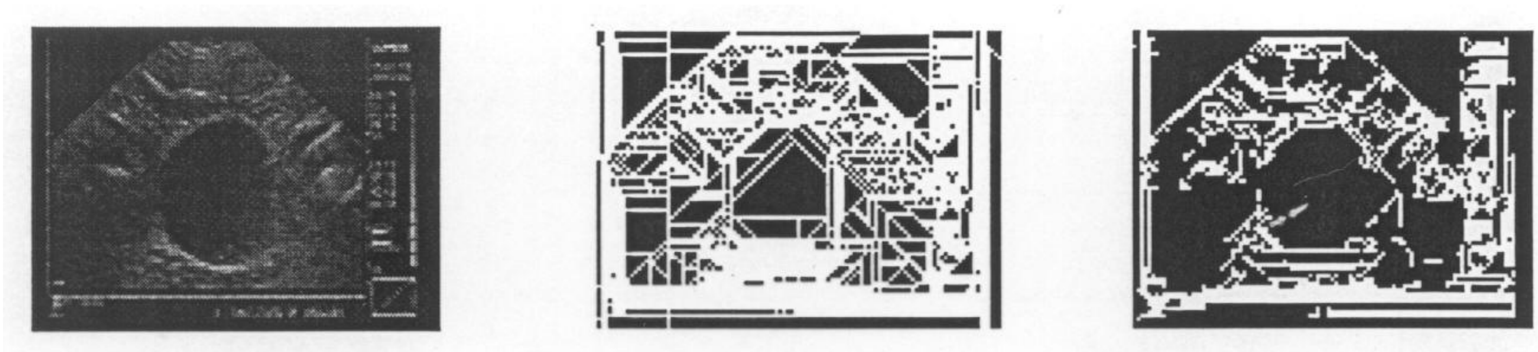


930 polygons

4521 polygons

Merging

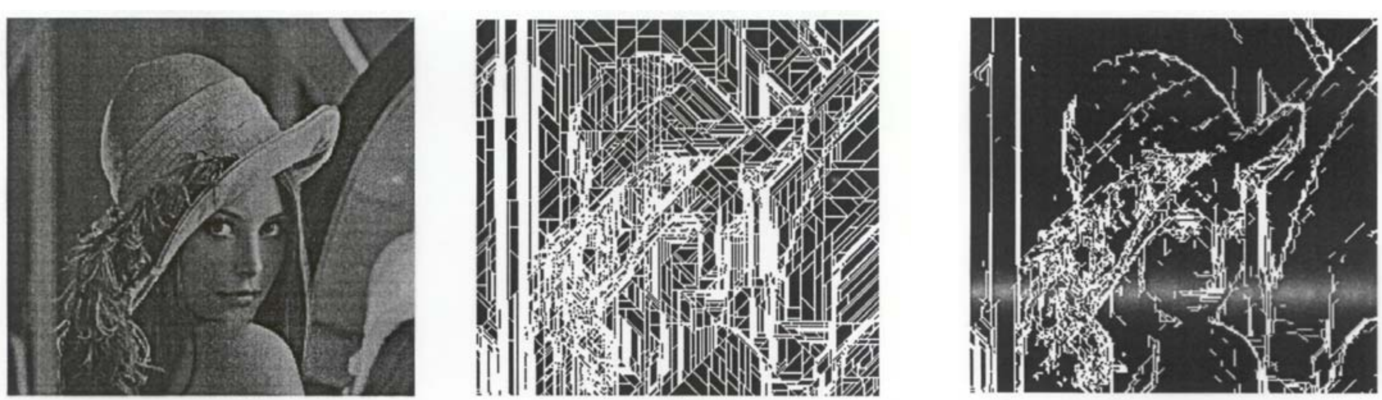
- The number of polygons which are produced by ROFS can be very large
  - merge any two neighbor regions G_i, G_j satisfying $|\mu(G_i)-\mu(G_j)| < m$
  - *m* is the "merging parameter"
  - *m* is user defined



original image

segmented image: after splitting

segmented image: after merging



Merging: Problem 1

- Examine all pairs of regions to find whether they are neighbors
  - their number K can be very large
  - examine K^2 regions
  - is it possible to know in advance the pairs of neighboring regions ?
  - Yes ! through the Region Adjacency Graph (RAG)

Merging Using RAGs

- RAG is always planar with small degree *e*
  - algorithms from graph theory
  - small *e*: the algorithms are fast

Merging of G_i, G_j:

- update the RAG: keep one of the G_i, G_j and delete the other along with all its incoming and outgoing arcs
- complexity $O(Ke)$

Region Merging: Problem 2

- Specification of ε, m
  - user defined, by experimentation
- The performance of the method depends on ε, m
- The performance of the method does not depend on pixel intensity values
  - the method is robust against noise

K-Means Segmentation

- Segmentation as a classification problem
  - assume *K* regions, *K* known in advance
  - each pixels has to be classified as belonging to one of the *K* regions
  - a region is represented by its center
  - classification criteria: intensity, proximity
  - each pixel: (x,y,d) normalized in $[0..1]$
  - a pixel belongs to the region whose center (x_c,y_c,d_c) which is closest to it

K-means Segmentation Algorithm

- Input: N points $(x,y)_i$ ←→ S_i centers of K regions
  - a pixel is a triple
  - d: intensity

$$\vec{X} = (x, y, d)$$

1. Classify image pixels: $\vec{X} \in S_i \Leftrightarrow \left\| \vec{X} - \vec{Z}_i \right\| < \left\| \vec{X} - \vec{Z}_j \right\|, \forall i \neq j$

2. Compute N new points (centers)

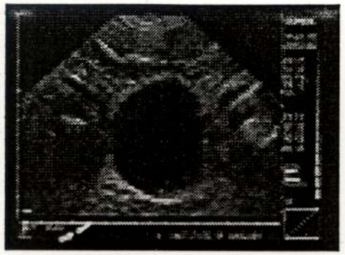
$$\vec{Z}_i = \frac{1}{N_i} \sum_{\vec{X} \in S_i} \vec{X} \Leftrightarrow \vec{Z} : \sum_{\vec{X} \in S} \left\| \vec{X} - \vec{Z}_i \right\|^2 : \min, i = 1,2 \ldots N_i$$

3. Repeat steps 1,2 until the centers do not change significantly (use a distance threshold) or until homogeneous regions $\sigma < \tau$

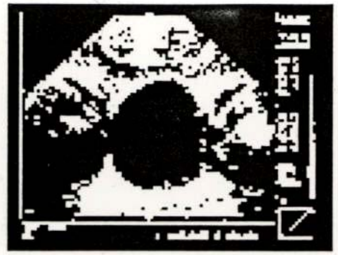
original image

K=2,σ=4

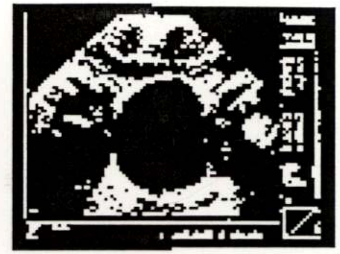
K=2, σ=8



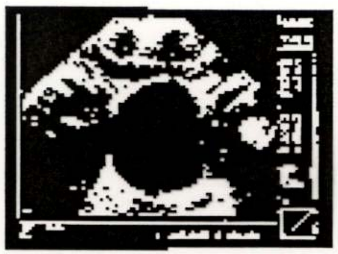
original image



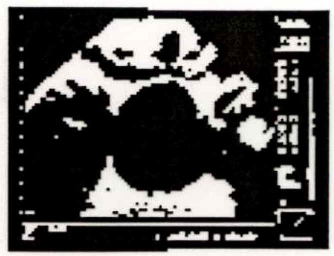
K=2



*K=2, σ=0.7**

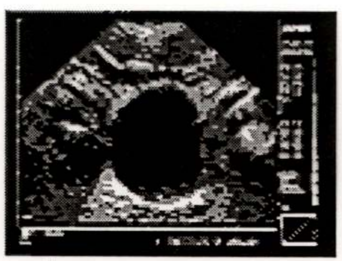


K=2, σ=4

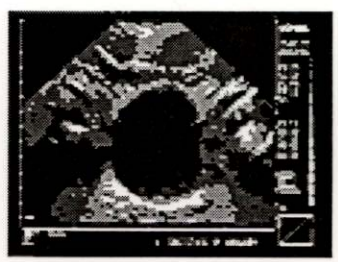


K=2, σ=8

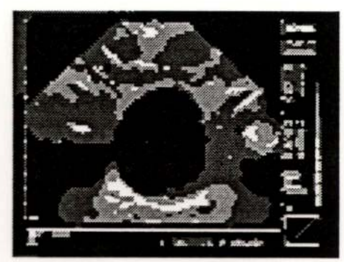
*A. Matamis, Msc.Thesis Dept. of Electronic and Comp. Eng., TUC, 1996



K=4, σ=0.7



K=4, σ=4

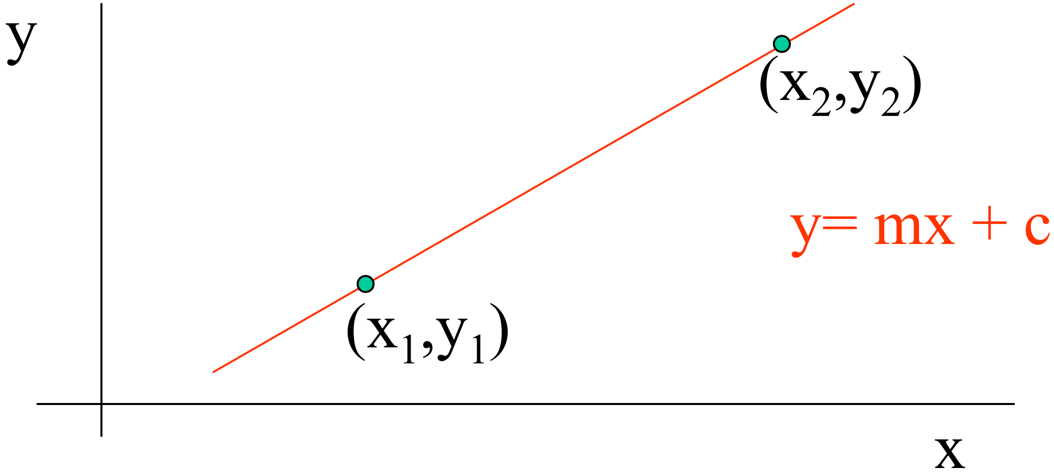


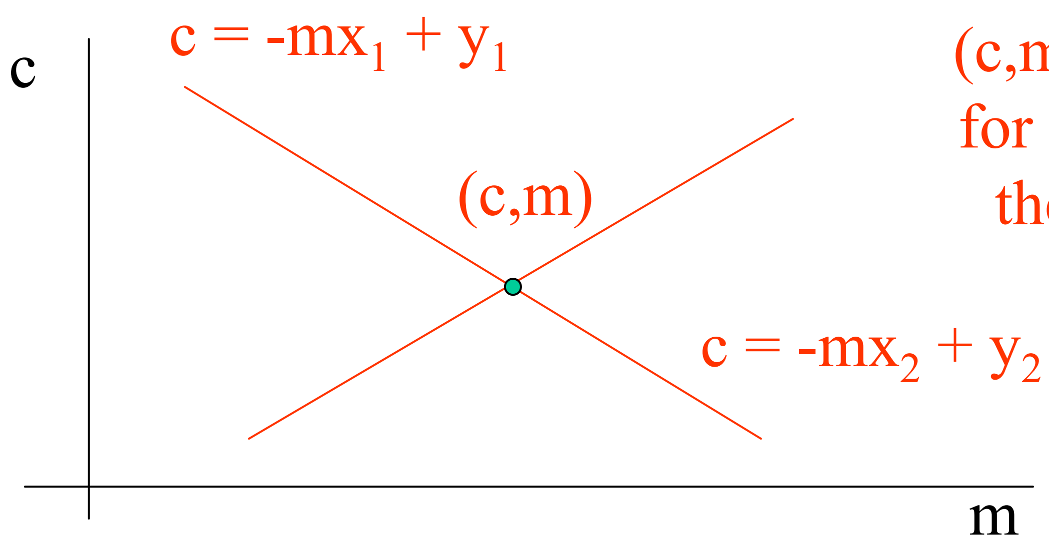
K=4, σ=8

Hough Transform (Duda & Hart 1972)

- Find Shapes whose curve can be expressed by an analytic function
  - find lines, circles, ellipses etc.
  - lines: $y = mx + c$
  - circles: $(x-a)^2 + (y-b)^2 = r^2$
- Works in a parametric space
  - $(x,y) \rightarrow (m,c)$ for lines (or ρ,θ)
  - $(x,y) \rightarrow (a,b,r)$ for circles

1. Line Detection





(c,m) is the same for the points of the same line

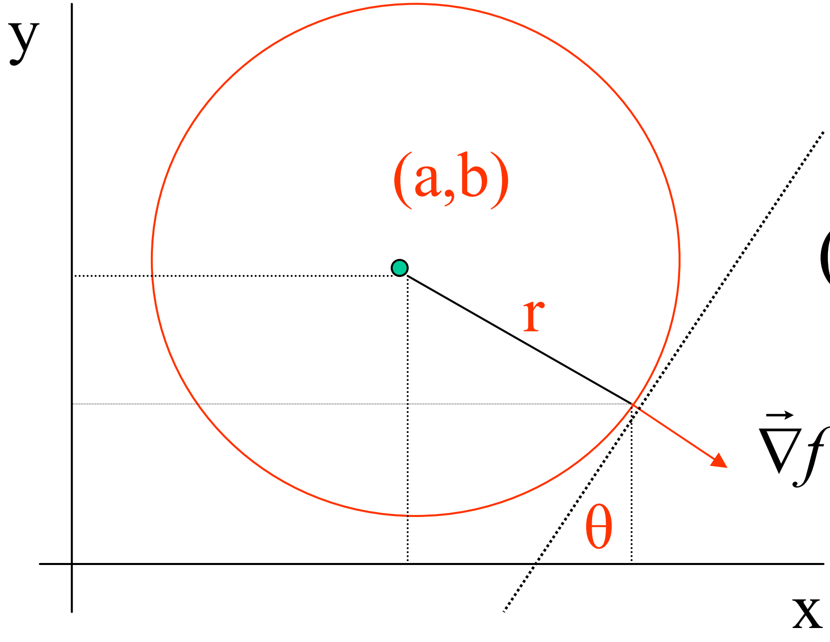
2. Line Detection Algorithm

- Input: Gradient $\vec{\nabla} f(x,y) : (s(x,y), \theta(x,y))$
- Output: Accumulator Array A(m,c)

1. for each point in the Gradient image compute:
   a) M = tan{θ(x,y)}
   b) C = -mx + y
   c) update A(m,c): A(m,c) = A(m,c) + g(x,y)*
2. points on a line update the same point in A(m,c)
   - lines: local maxima in A(m,c)

* g(x,y) intensity, strong intensity points contribute more

3. Circle Detection



$$(x-a)^2+(y-b)^2=r^2$$

$$\vec{\nabla} f(x,y)$$

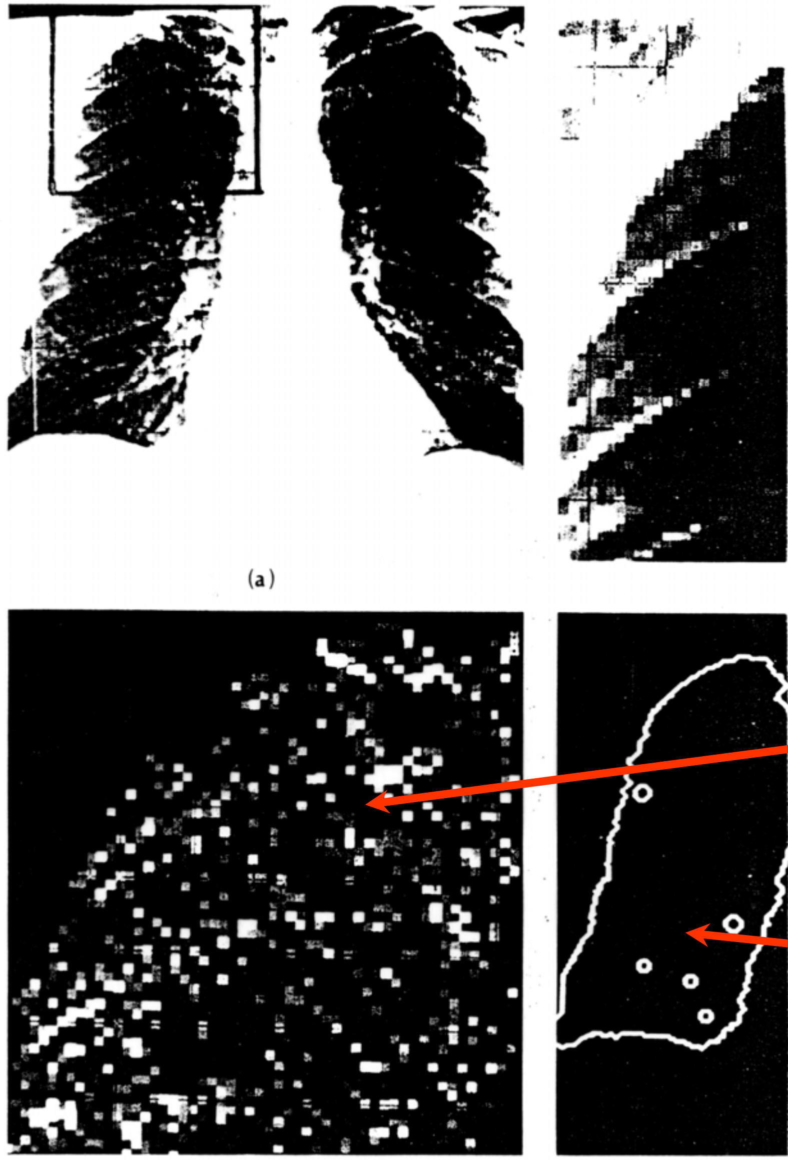
- All Circles (x,y) → 3-dim. space (a,b,r)
- Circles with fixed radius r → 2-dimensional space

4. Circle Detection Algorithm

- Input: Gradient $\vec{\nabla} f(x, y) : (s(x, y), \theta(x, y))$
- Output: Accumulator Array A(m,c)

1. for each point (x,y) in the image compute:
   a) $a = x + r \sin\theta$
   b) $b = y - r \cos\theta$
   c) udpdate A(a,b) = A(a,b) + g(x,y)
2. points on a circle update the same point in A(a,b)

– to detect all circles, compute different A(a,b) for different radius
– this can be very slow



Hough Transform for detecting circles in an X-chest Radiograph (from Ballard And Brown)

accumulator array for r = 3

results of maxima detection

Comments on Hough Transform

- Pros:
  - detects even noisy shapes
  - the shapes may have gaps or may overlap
  - effective for low dimensionality parametric spaces (e.g., 2, 3)
- Cons:
  - the shapes must be known
  - can be very slow for complex shapes
  - complex shapes are mapped to high dimensional spaces

Relaxation Labeling

- Edge or Region segmentation as a special case of pattern classification problem
  - *two classes:*
    - region/background for region segmentation
    - edges/background for edges segmentation
- Probabilistic approach:
  - initial probability estimates are revised in later steps depending on compatibility estimates

Edge Segmentation with Relaxation Labeling

- For each point (x_i,y_i) on a Gradient image compute its probability P_i to belong to an edge
  - if point (x_j,y_j) is very close to point (x_i,y_i) and has large P_j to belong to an edge
    - then the two events (P_i and P_j belong to the same edge) are compatible → increase P_i
  - if point (x_j,y_j) is very close to point (x_i,y_i) and has low P_j to belong to an edge
    - then the two events are incompatible → decrease P_i

General Relaxation Labeling Model

- Classify "*Objects*" $A_1, A_2, \ldots A_n$ to $C_1, C_2, \ldots C_m$ classes
- P_{ij}: Probability for A_i to belong to C_j
- $C(i,j;h,k)$: compatibility between P_{ij} and P_{hk}
  - $C(i,j;h,k) > 0$: compatible (increase probabilities)
  - $C(i,j;h,k) < 0$: incompatible (decrease probabilities)
  - $C(i,j;h,k) = 0$: don't care (do nothing)

Adaptation of Probabilities

- Adaptation of P_{ij} due to P_{hk}:

$$g_{ij} = C(i,j;j,k)P_{hk}$$

- Adaptation due to every other point:

$$g_{ij} = \frac{1}{n-1}\sum_{\substack{h=1 \\ h \neq i}}^{n}\left\{\sum_{k=1}^{m} C(i,j;h,k)P_{hk}\right\}$$

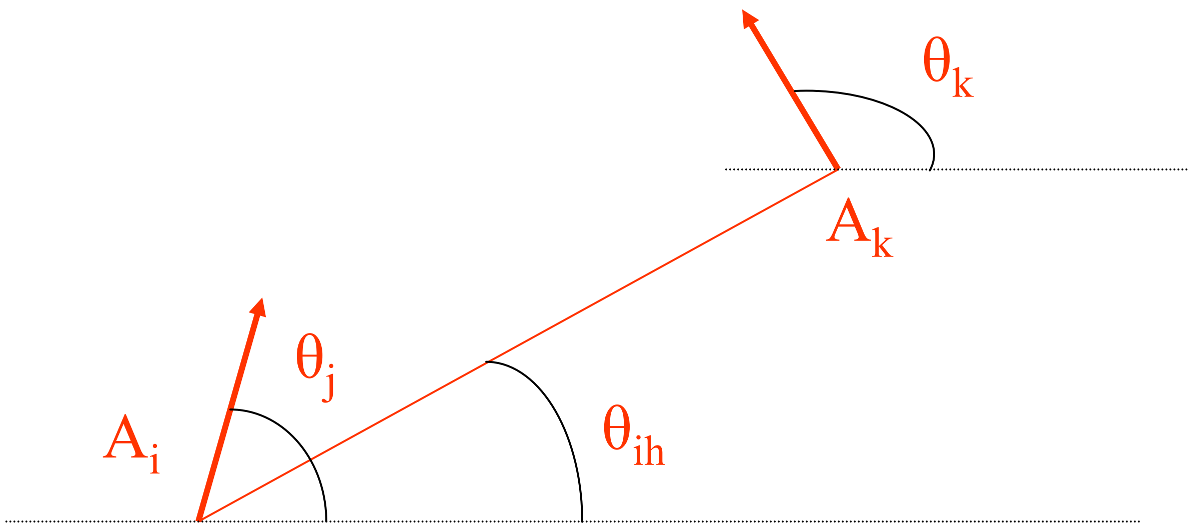
- At every step P_{ij} becomes

$$P_{ij} = \frac{P_{ij}(1+q_{ij})}{\sum_{j=1}^{m} P_{ij}(1+q_{ij})}$$

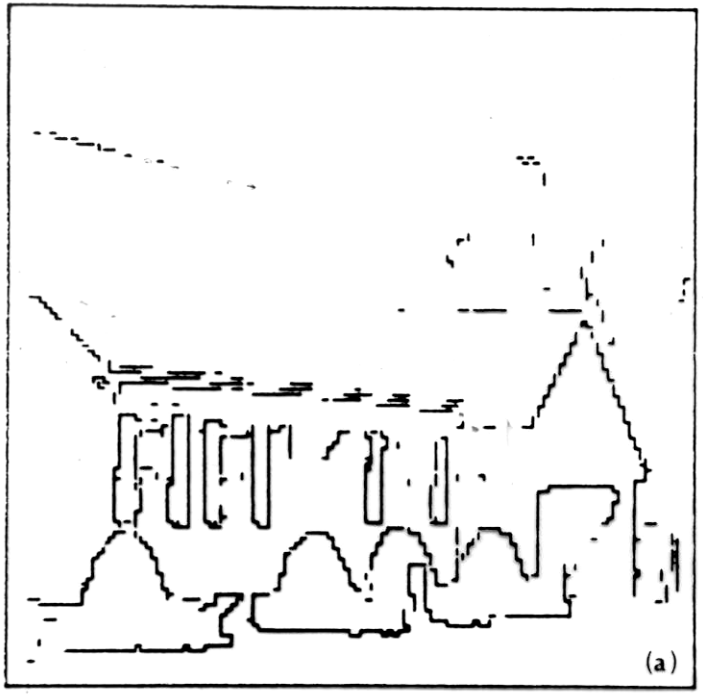
Segmentation using Relaxation Labeling

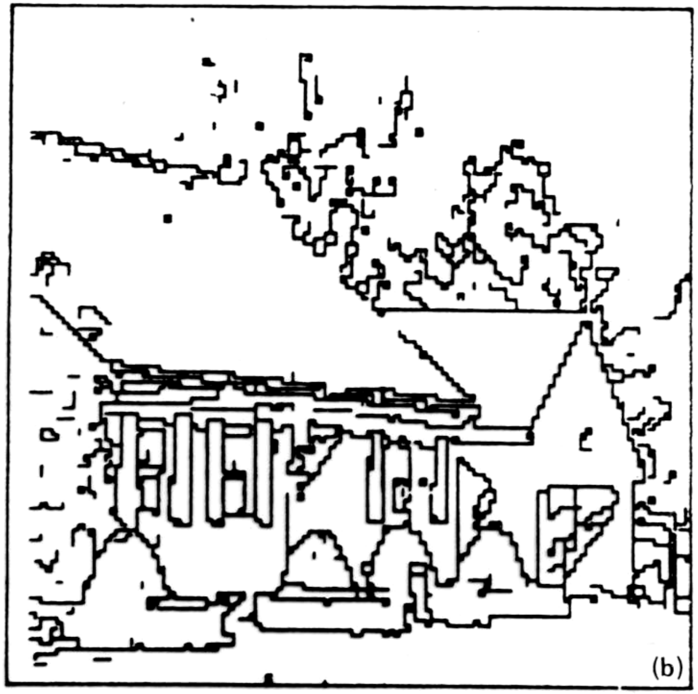
- Two classes:
  - ✓ Edge, Background
  - ✓ Region, Background
- P_{i1}: pixel *i* belongs to class *1* (edge, region)
- P_{i2}: pixel *i* belongs to class *2* (background)
- $P_{i1} = 1 - P_{i2}$
  - ✓ $P_{i1} = g_i/g_{max}$ where g_i: intensity of *i* and g_{max}: max intensity in the image

Edge Segmentation Example



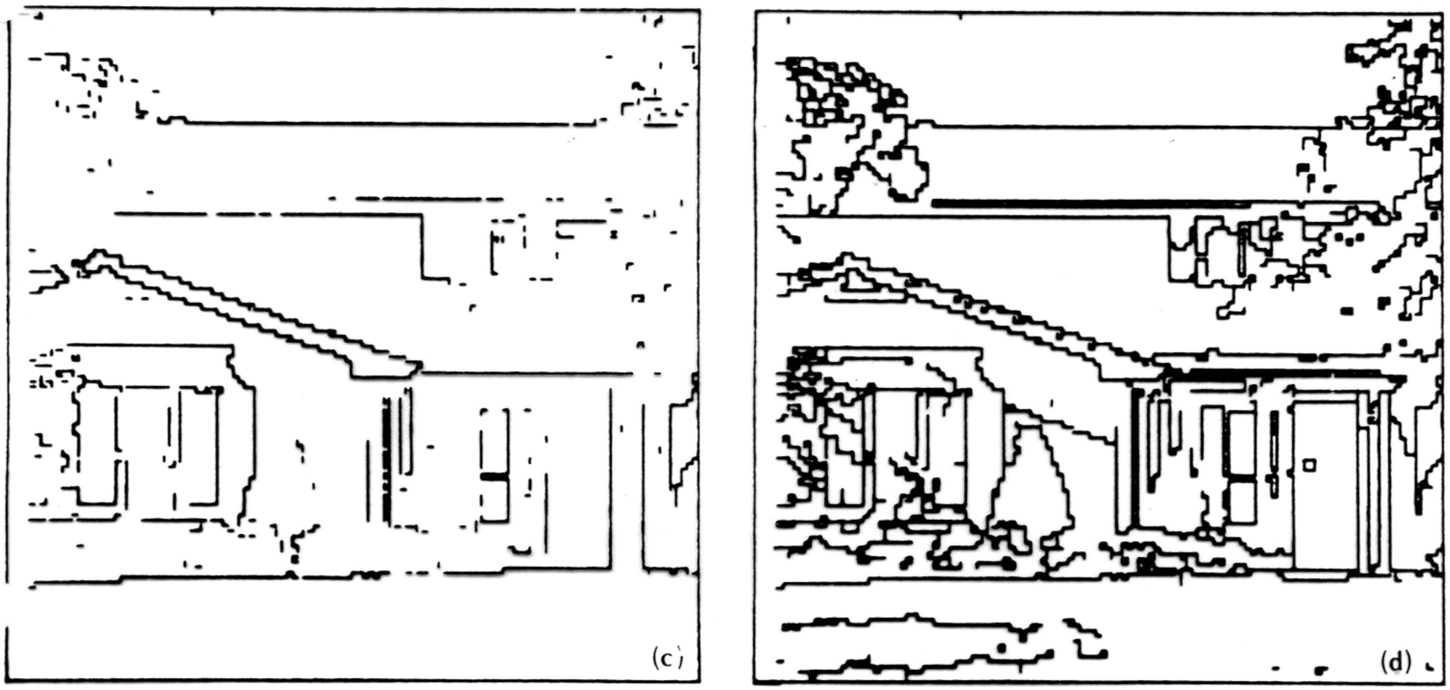
- $C(i,j;h,k)$ is defined only for the nearest neighbors
- $C(i,j;h,k) = cos(\theta_j - \theta_{ih})cos(\theta_k - \theta_{ih})$
  - if θ_j, θ_k // θ_{ih} $\rightarrow$ $C(i,j;h,k) = 1$
  - if θ_j, θ_{ih} | θ_{ih} or θ_k, θ_k | θ_{ih} $\rightarrow$ $C(i,j;h,k) = 0$





Raw edges. Initial edge strengths thresholded at 0.35 (removes some noise)

Results of relaxation segmentation after 5 iterations



Raw edges. Initial edge strengths thresholded at 0.25 Better initial estimates!!

Results after 5 iterations. Notice the effect of having better initial estimates