Region-based Segmentation

Overview of segmentation - Sonka

Chap. 5

Segmentation - divide an image into parts that have a strong correlation with objects or areas of the real world contained in the image

complete - results in a set of disjoint regions corresponding uniquely with objects in the input image

partial - regions do not directly correspond with image objects

complete requires cooperation with higher level processing levels which use domain specific knowledge

but there is a whole class of segmentation problems that can be solved using lower-level processing -

- usually high-contrast objects on a uniform background
- can ~~obtain~~ complete ~~segmentation~~ using ~~global~~ processing

partial - divide an image into regions that are homogeneous with respect to a chosen property such as brightness, color, etc.

partially segmented images typically are input to stages which use domain knowledge and object-specific models.

types of segmentation methods

- global knowledge - typically histogram based
- edge-based
  - edge image thresholding
  - edge relaxation
  - border tracing -
  - graph searching
  - dynamic programming (optimization)
  - Hough transform
  - a priori border location info - divide & conquor from endpoints
  - region construction from borders.

duals since boundary encloses a region and vice versa

- region-based
  - merging
  - splitting
  - split-merge
  - watershed
  - matching

other
- template matching
- texture

Gonzalez & Woods

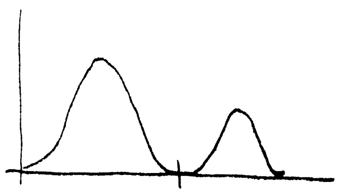
7.3 Thresholding

7.4 Region-Oriented Segmentation

7.5 Motion in Segmentation

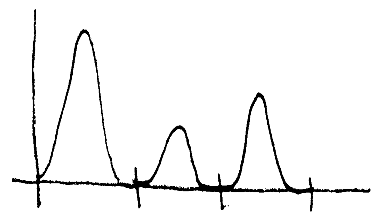
7.3 Thresholding - one of the most important approaches to segmentation

7.3.1. Foundation



T

single threshold



T_1 T_2

multi-level Thresholding
not reliable due to difficulty of establishing thresholds that isolate regions of interest

$$\text{thresholded image} \quad g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T & \text{objects} \\ 0 & \text{if } f(x,y) \le T & \text{background} \end{cases}$$

global when T depends only on $f(x,y)$, the gray level at (x,y)

local when T depends on $f(x,y)$ and $p(x,y)$, some local property such as average gray level of neighborhood centered at (x,y).

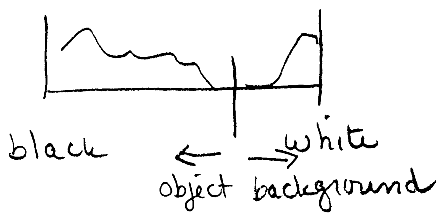
dynamic T depends on $f(x,y)$, $p(x,y)$ and is a function of (x,y).

3.2.1. Automatic Thresholding

should use as much knowledge as possible

- intensity characteristics of objects
- object sizes
- fractions of an image occupied by an object
- # of different types of objects in image

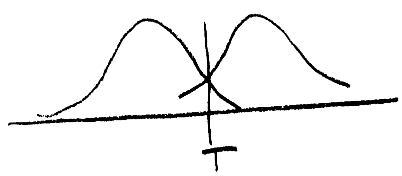
automatic thresholding
use histogram <u>and</u> object knowledge to select best threshold.

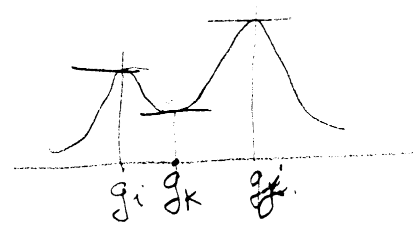


<u>P-tile method</u>

if object occupies p% of image area pick Threshold so that p% of pixels assigned to object

<u>mode-method</u>





<u>Algorithm 3.1 Peakiness Algorithm</u>

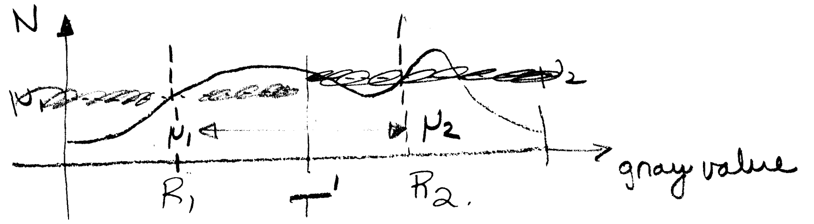
1. Find two highest local maxima in the histogram that are at some minimum distance apart. Suppose these occur at gray values g_i & g_j
2. Find the lowest point g_k in the histogram H between g_i and g_j
3. Find the peakiness, defined as $\dfrac{\min[H(g_i), H(g_j)]}{H(g_k)}$
4. Use the combination (g_i, g_j, g_k) with highest peakiness to threshold the image. The value g_k is a good threshold to separate objects corresponding to g_i and g_j

Iterative thresholding

start with some approx. Threshold
use some image (object) property to update this estimate,
i.e. area of object is a good one

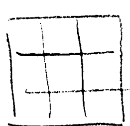
Algorithm 3.2 Iterative Threshold Selection

1. Select an initial estimate of the threshold T. Usual initial guess is average intensity of image.
2. Partition image into two groups, R_1 and R_2, using threshold T
3. Calculate mean gray values μ_1 and μ_2 of partitions R_1 and R_2.
4. Select a new threshold
$$T' = \frac{1}{2}(\mu_1 + \mu_2)$$
5. Repeat 2 to 4 until mean values μ_1 and μ_2 in successive iterations do not change.

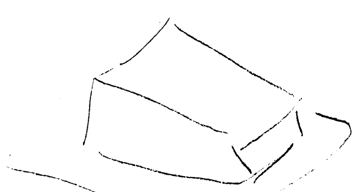


Adaptive Thresholding

bad lighting, shadows, direction of illumination



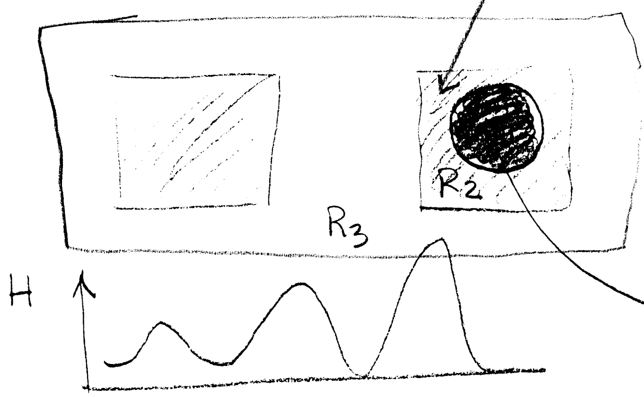
partition into m x n subimages.



background normalization
fit function to gray scale values
difficult to threshold.

Double Thresholding

wants to threshold to get this part of image.



core
use conservative threshold T_1
grow by using 2nd threshold T_2
and only add those pixels connected
to original set of points.

Algorithm 3.1

1. Select the two thresholds T_1 and T_2
2. Partition image into three regions:

   $R_1 \quad g < T_1$

   $R_2 \quad T_1 \le g \le T_2$

   $R_3 \quad g > T_2$.

3. Examine all $p \in R_2$. If p has a neighbor in R_1 then assign p to R_1.
4. Repeat 3 until no pixels change assignment.
5. Assign any pixels left in region R_2 to region R_3.

3.2.2. Limitations of histogram methods in general

– only useful for objects with constant gray values, i.e. no lighting problems.

– histogram does not use <u>spatial information</u>
  histogram is global
  does not exploit "surface coherence", i.e. connectivity of points

7.3.5. Threshold Selection Based On Boundary Characteristics

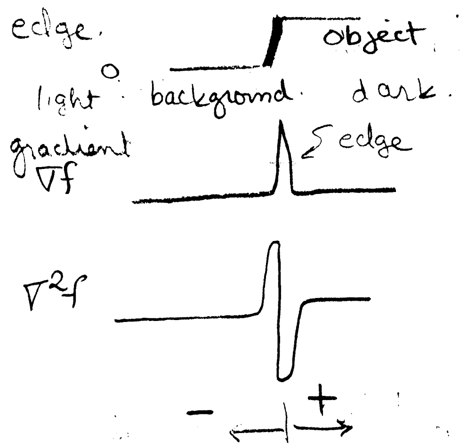
You can't use thresholding for segmentation unless you get good bimodal peaks. How to improve histogram shapes?

(i) consider only pixels on or near object/background boundary.
BUT how do you find boundary in first place

Compute gradient or Laplacian of original gray scale image

Form new three-level image according to

$$s(x,y) = \begin{cases} 0 & \text{if } \nabla f < T \quad \text{not much change so ignore} \\ + & \text{if } \nabla f \geq T \text{ and } \nabla^2 f \geq 0 \quad \text{pixels on light side of an edge} \\ - & \text{if } \nabla f \geq T \text{ and } \nabla^2 f < 0 \quad \text{pixels on dark side of edge.} \end{cases}$$



interior of object all zero, possibly +, but only − denotes edge from object to background.

this might then be line of an image.

$(0\,0\,0)\ (-,+)(0 \text{ or } +)(+,-)(\ldots)$

any combination

background to object transition. this is an edge.

object to background transition this is another edge.

7.3.6. Thresholds based on Several Variables

RGB — do a 3D histogram
then do clustering in 3-D color space

can also use HSI
hue, saturation, intensity

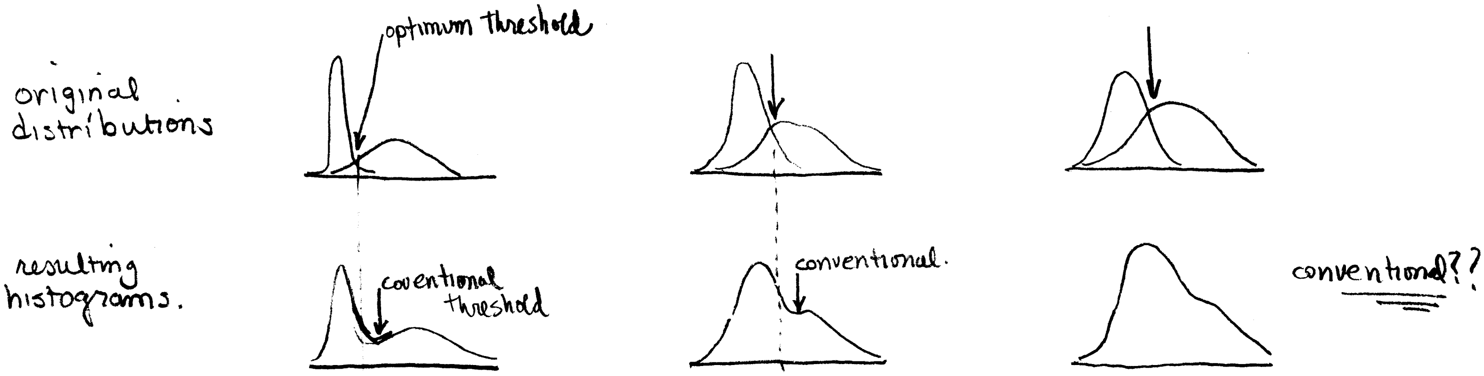
Other thresholding techniques

use a band-thresholding to detect borders.

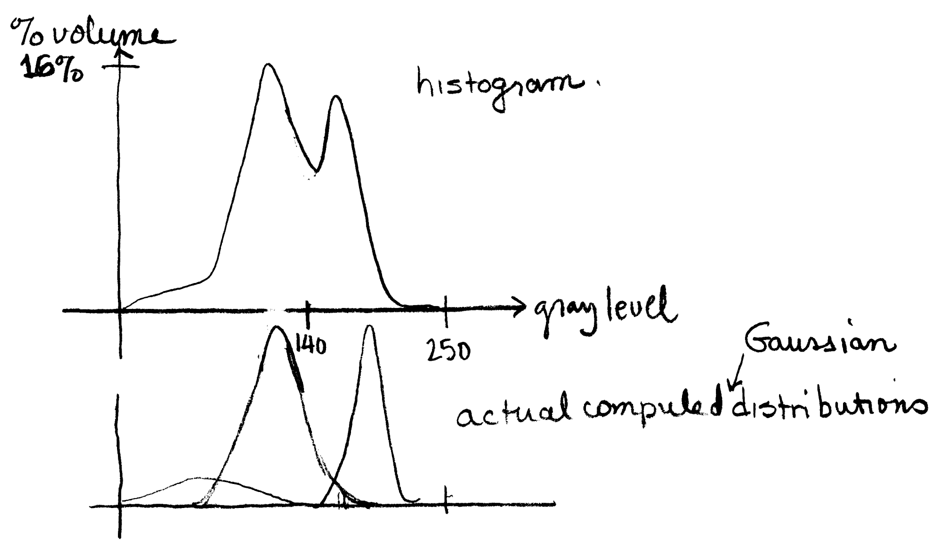
optimal thresholding

represent a histogram as a sum of probability density functions and optimize some aspect of a two-distribution mix

- minimum error segmentation — minimize # of pixels misclassified
- maximize gray-level variance between object and background (Otsu)



Example: segment 3-D T1-weighted MRI images into white matter (WM), gray matter (GM), and cerebro-spinal fluid (CSF)



7.3.3 ~~Simple Global Thresholding~~
~~good in highly controlled environments~~

7.3.4. Optimal Thresholding

uses probability density functions

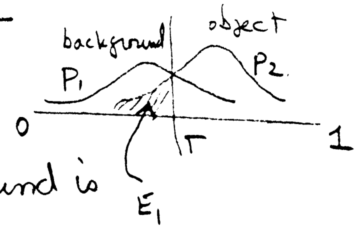
assume histogram ~ brightness pdf $p(z)$.

Imagine image as sum (mixture) of two densities (objects), one bright and one dark.

$$p(z) = P_1 p_1(z) + P_2 p_2(z).$$

where P_1, P_2 are the a priori class probabilities and $P_1 + P_2 = 1$.

$$p(z) = \frac{P_1}{\sqrt{2\pi}\,\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\pi}\,\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma_2^2}}$$



Probability of (erroneously) classifying object as background is

$$E_1(T) = \int_{-\infty}^{T} p_2(z)\,dz$$

Probability of erroneously classifying background as object is

$$E_2(T) = \int_{T}^{\infty} p_1(z)\,dz.$$

prob. object

prob. error in classifying object as background.

Overall probability of error is $E(T) = P_2 E_1(T) + P_1 E_2(T)$.

Optimimum error when $\frac{\partial E(T)}{\partial T} = 0$.

i.e. $P_1 p_1(T) = P_2 p_2(T)$.

For gaussian case when

$$AT^2 + BT + C = 0 \quad \text{where} \quad \begin{aligned} A &= \sigma_1^2 - \sigma_2^2 \\ B &= 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2) \\ C &= \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\sigma_2^2 \ln\left(\frac{\sigma_2 P_1}{\sigma_1 P_2}\right) \end{aligned}$$

Good in theory, not in practice since distributions not generally known.

Problem is that you have only the "mix" histogram $h(z)$ and must use it to estimate the mix pdf $p(z) = P_1 p_1(z) + P_2 p_2(z)$

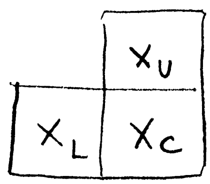
i.e. find parameters which minimize

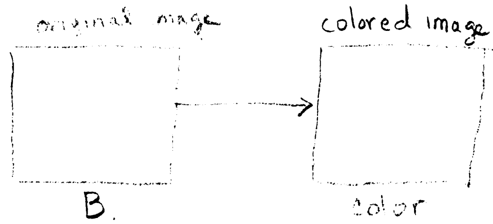
$$e_{mix} = \frac{1}{n}\sum_{i=1}^{n} |p(z_i) - h(z_i)|^2$$

labeling

Simplest algorithm
"blob" coloring

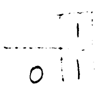
given a binary image containing <u>four-connected</u> blob's of 1's on a background of 0's, assign each blob a different label, i.e. color.
Scan the image left to right, top to bottom with the L-shaped template shown below.



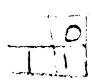


Algorithm 5.1 (Ballard & Brown).
Let the initial color, $k=1$. Scan $L \rightarrow R$, top to bottom.

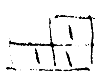
If $B[X_c] = 0$ then continue
else
    begin



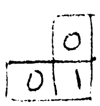
    if $B[X_u]=1$ and $B[X_L]=0$ then color $(X_c) :=$ color (X_u).



    if $B[X_L]=1$ and $B[X_U]=0$ then color $(X_c) :=$ color (X_L)



    if $B[X_L]=1$ and $B[X_u]=1$ then
        begin
        color $(X_c) :=$ color (X_L)
        color (X_L) is equivalent to color(X_u)
        end



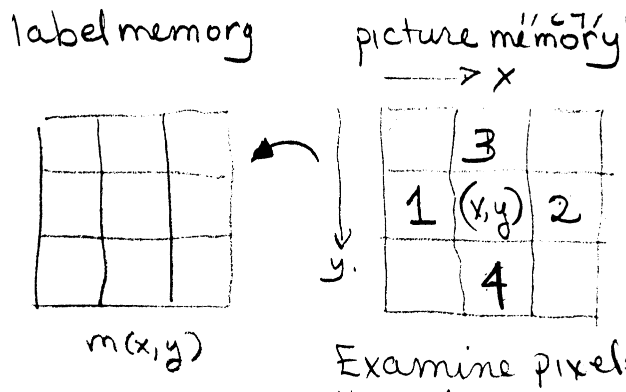
    if $B[X_L]=0$ and $B[X_u]=0$.
    then begin
        color $(X_c) := k$; $k := k+1$
        end.
    comment: now color

    end.

Recursive region growing (Snyder)

label memory $m(x, y)$

picture memory $p(x, y)$



Examine pixels in this order, around each pixel.

Initial color $N := 0$

Repeat until no unlabeled $\langle x, y \rangle$ for which $B[x, y] = 1$

Find a $\langle x, y \rangle$ which has $B[x, y] = 1$ and $m(x, y) = 0$. * $m = 0$ unlabeled, $p = 0$ "black"

Push $\langle x, y \rangle$ onto stack. Color $N := N + 1$.

While stack not empty

Begin

Pop $\langle x, y \rangle$ from stack.

If $B[x-1, y] = 1$ and $m(x-1, y) = 0$
push $\langle x-1, y \rangle$ onto the stack.

If $B[x+1, y] = 1$ and $m(x+1, y) = 0$
push $\langle x+1, y \rangle$ onto the stack.

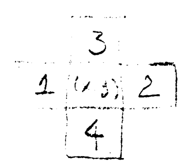
If $B[x, y-1] = 1$ and $m(x, y-1) = 0$
push $\langle x, y-1 \rangle$ onto the stack.

If $B[x, y+1] = 1$ and $m(x, y+1) = 0$
push $\langle x, y+1 \rangle$ onto the stack.

$m(x, y) := N$

end.

this algorithm has problems with equivalences.



check around (x, y) in this order. As soon as you find unlabeled pixel push onto stack check all then pop.

2.3.2 Connected components algorithm

typical iterative algorithm
→ accross (L to R)
(then down) ↓

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

original binary image.

⟹

labeled image

0	0	1	0	0	0	2
0	0	1	1	0	0	2
0	0	1	1	1	0	2
0	0	1	1	1	1	A

} connected region 2

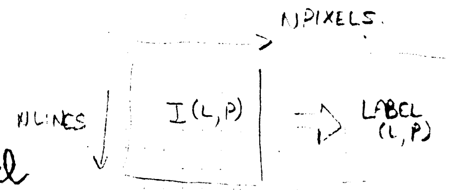
connected region A

algorithm assigns label to A to make regions 1 & 2 equivalent

How to compare algorithms

1. What label should be assigned to A?
2. How does algorithm keep track of equivalence of two (or more labels)?
3. How does algorithm use equivolence information?

2.3.3 An Iterative algorithm




```
procedure Iterate
\* initialize each 1-pixel to a unique value label
for L:=1 to NLINES do
   for P:=1 to NPIXELS do
      if I(L,P)=1
      then LABEL(L,P) := NEWLABEL() else LABEL(L,P):=0
   end for
end for
\* iteration of top-down pass followed by bottom up pass *\
repeat till no changes
\* top-down pass
CHANGE := false;
for L:=1 to NLINES do
   for P:=1 to NPIXELS do
      if LABEL(L,P)<>Ø then
         begin
         m:= MIN (LABELS (NEIGHBORS( (L,P)) U (L,P)));
         if m<> LABEL(L,P) then CHANGE:=true;
         LABEL(L,P) := m
         end
   end for
end for;
```

returns the minimum label

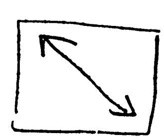
returns the set of labels

returns set of already-labeled neighbors on its own line or above

including current pixel

1	2
3	4

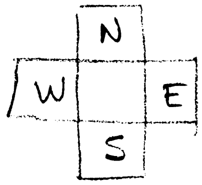
basically repeat for bottom-up pass.
except starts at lower right.



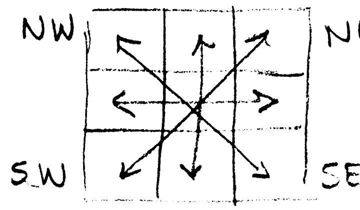
until no change

definitions and issues in computer vision

two 1-pixels p & q belong to same connected component C
if there is a sequence of 1-pixels $(p_0, p_1, \ldots, p_n) \in C$
where $p_0 = p$, $p_n = q$ and p_i is a neighbor of p_{i-1} for $i = 1, \ldots, n$



4-connected



neighbors of a pixel are adjacent to that pixel.
border of a connected component of 1-pixels is the subset of pixels belonging to C that is also adjacent to ϕ-pixels.

same image

		1		
	1		1	
		1		

4 connected object & background.

a	a	a	a	a
a	a	1	a	a
a	2	b	3	a
a	a	4	a	a
a	a	a	a	a

8 connected object & background.

a	a	a	a	a
a	a	1	a	a
a	1	a	1	a
a	a	1	a	a
a	a	a	a	a

8 connected 1-pixels (object)
4 connected ϕ-pixels (background).

a	a	a	a	a
a	a	1	a	a
a	1	b	1	a
a	a	1	a	a
a	a	a	a	a

letters - background objects

numbers - foreground objects

This leads to Minkowski set algebra.
Euler number
Image morphology.

Euler number & Connectivity (Horn)

Euler # $\triangleq$ # of objects − # of holes

a concavity is <u>NOT</u> a complete hole.

Probably in next lecture.

3.3. Region Representation

- array representations
- hierarchial representations
- symbolic representations

array representations

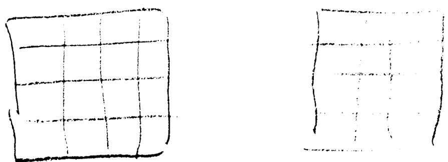


image
can be grayscale

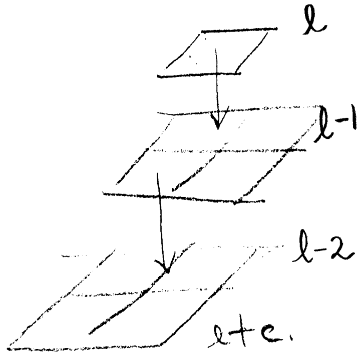
(1) label or region image — already talked about from connected component analysis

(2) multiple membership arrays
each one is a binary mask which can be overlaid on original image
allows multiple memberships

---

hierarchial representations

Pyramid



pixel at level l combines info. from multiple pixels at level l−1

This is what is used for generating Thumbnail sketches (images) for photoshop, etc.

quadtrees (next page)

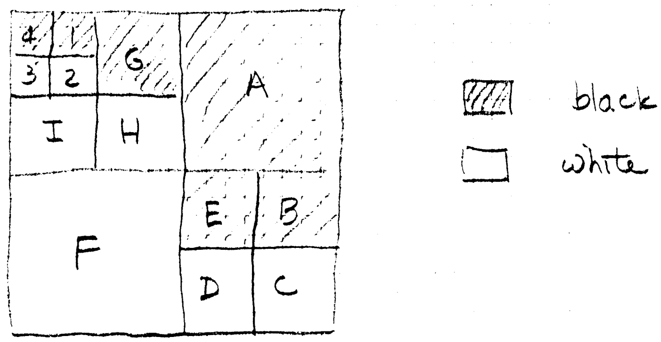
variable resolution — just like a pyramid but only continues until all pixels in region are uniform

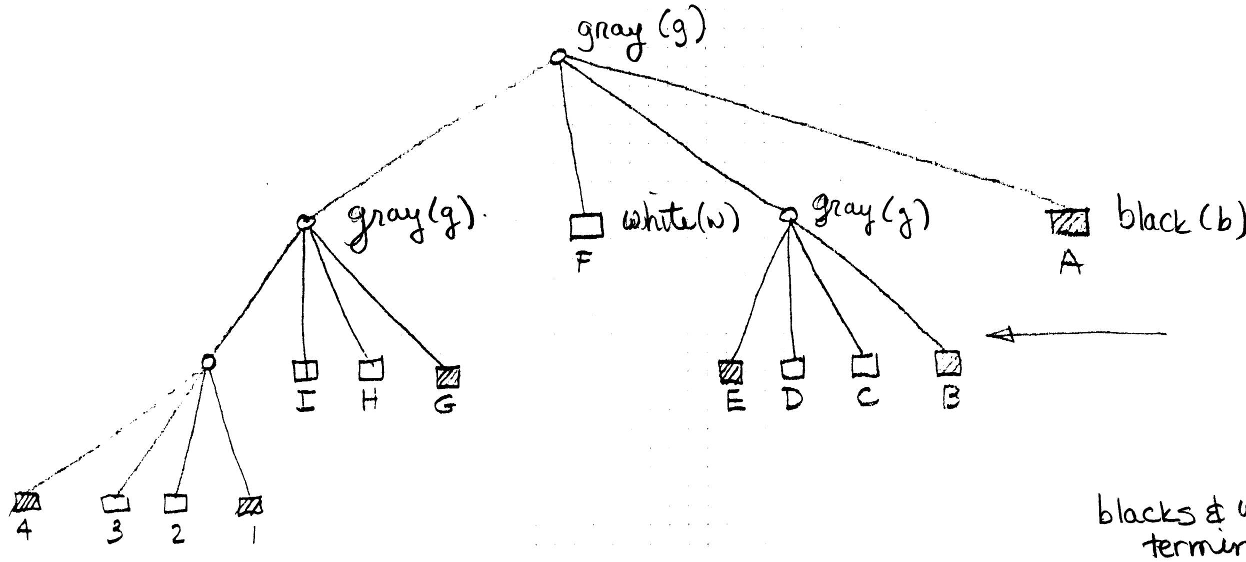
symbolic representations

- enclosing rectangles
- centroid
- moments
- Euler #'s.

quad-tree representation (how to represent shape of objects)

alternative to run-length encoding





blacks & whites are terminal
grays recurse.

code gbgbwwbwgbwwgbwwb

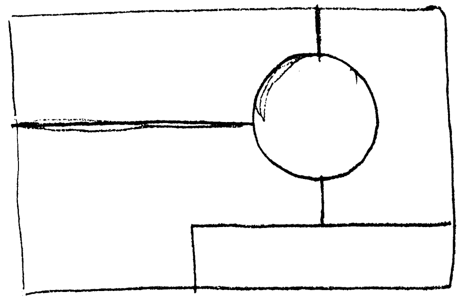
decode g(bg(bwwb)wg(bwwg(bwwb)))

→ always in groups of four.

quad-tree representation
- more efficient in terms of compression than RLE
- computing perimeters & moments more difficult.

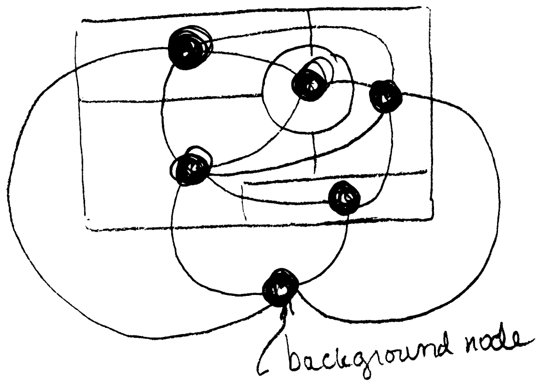
Ref: H. Samet, "Region Representation: Quadtrees from Boundary Codes." Comm. ACM 23 (March 1980): 163-170

Region Adjacency Graphs.

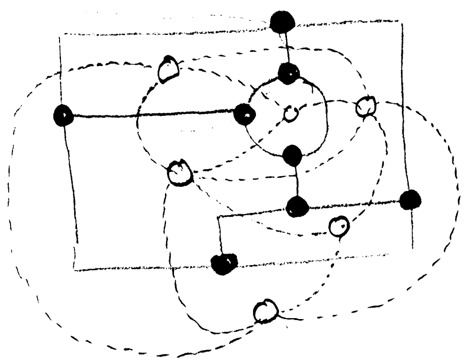


Nodes – represent regions

arcs between nodes represent common boundaries



Region adjacency graph.



this is the dual graph
nodes represent boundaries
arcs represent regions

label array

Algorithm 3.4 Constructing the Region Adjacency Graph from a labled array.

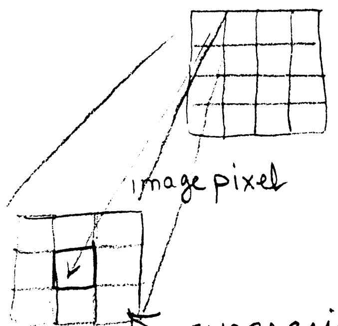
1. Scan the membership array a and perform the following steps at each pixel index [i, j]
2. Let $r_1 = a[i,j]$ region 1. i.e. look at the label
3. Visit the neighbors [k, l] of the pixel at [i, j]. For each neighbor of [i, j] perform 4.
4. Let $r_2 = a[k,l]$. label of region 2. If $r_1 \neq r_2$, add an arc between nodes r_1 and r_2 in the region adjacency graph.

~~Picture Trees~~

~~symbolic trees~~ ~~Simply a tree showing the inclusion of one region within another~~

Super Grid

put original image into an expanded image called the super grid.
Non-image pixels contain the boundary information



4×4
$N = 4$
$(2N+1) \times (2N+1)$ ← super grid is much bigger image.
9×9

← supergrid pixels indicate whether there is a boundary between the two pixels, and in what direction the boundary runs.

A variation of this is crack edges — edges located between pixels.

3.4 Split/merge.

thresholding usually doesn't work well & gives too many regions because of
- high-frequency noise
- gradual transition between gray values in different regions

split/merger
- eliminate false boundaries & spurious regions by merging adjacent regions which belong to the same object.

could also use a quadtree

Algorithm 3.5

1. Form initial regions in the image using thresholding (or similar approach) followed by ~~components~~ labeling.
(2. Prepare a region adjacency graph (RAG) for the image.)
3. For each region:
   (a) Consider its adjacent region and test to see if they are similar
   ((b) for regions that are similar, merge them and modify the RAG)
4. Repeat (3) until no regions are merged.

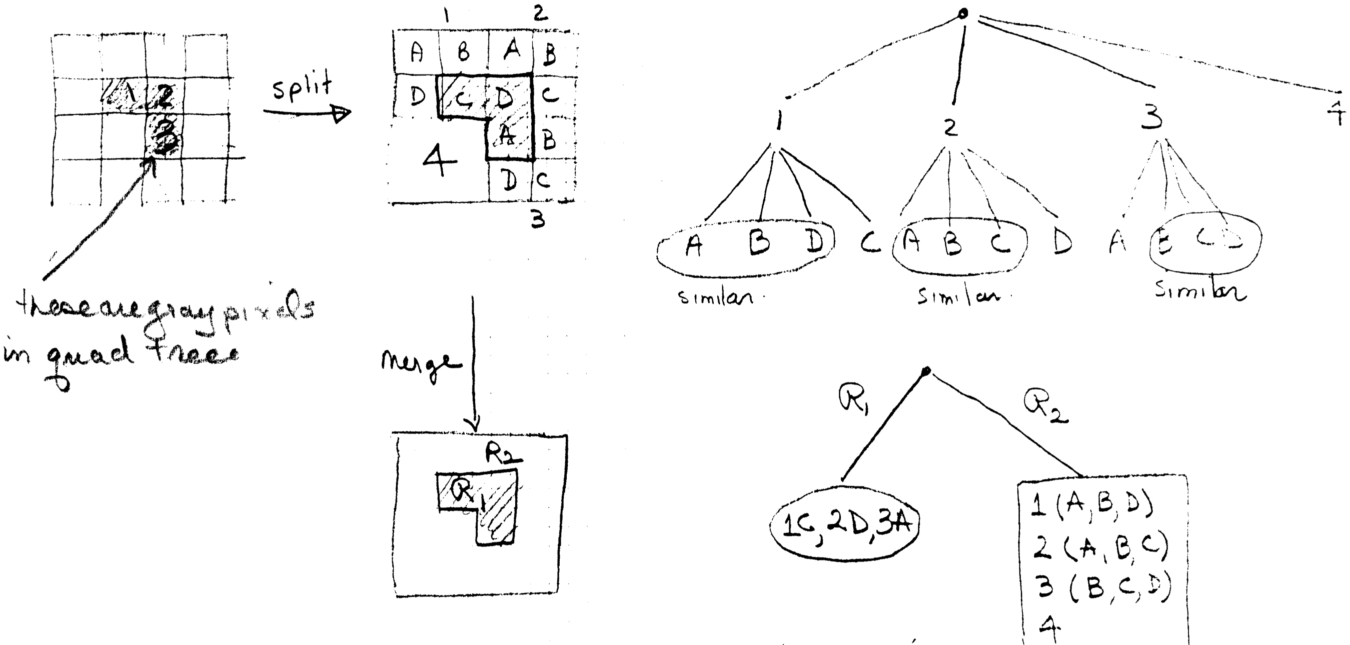
strategies
- \+ merge adjacent regions with similar characteristics (similarity measure)
- \+ remove questionable edges (low edge value)
- \+ use topological properties of the regions (get rid of holes)
- \+ use shape information about objects in the scene (clean up edges)
- \+ use semantic information about the scene (usually texture)

~~Other techniques for merging~~

Representation of object using quad trees

Do all splitting first. For example, a quadtree.

SKIP THIS



You need to segment & parameterize regions

How to merge. (region growing) One algorithm

1. Merge R_i and R_j if $\dfrac{w}{P_m} > \theta_1$

large # of weak boundaries pixels say the # below a threshold

is one of the objects small, i.e. small perimeter.

where $P_m = \min(P_i, P_j)$ where P_i, P_j are perimeters of R_i and R_j

w (these are between P_i & P_j) → w is # of weak boundary locations (i.e. magnitude change across boundary $< \sigma$, some threshold)

θ_1 controls size of region to be merged.

usually $\theta_1 = 0.5$

$\theta_1 \cong 1 \Rightarrow$ two regions will be merged only if one of the regions almost surrounds the other.

2. merge R_i and R_j if $\dfrac{W}{I} \geq \theta_2$

large # of weak boundary points

short common boundary

$\underline{I}$ = length of common boundary between regions

merge regions if boundary is sufficiently weak.

typically $\theta_2 = 0.75$

3. merge R_i and R_j if there are no strong edges between them (RLE connected component analysis is essentially this)

4. merge R_i and R_j if their similarity distance is less than a threshold. we have not talked about similarity

3.4 Split and merge.

intensity segmentation alone gives too many regions

- high frequency noise
- gradual transition between gray values in different regions

3.4.1 Region Merging

Algorithm 3.5

1. Form initial regions in the image using thresholding (or something similar) followed by component labeling.
2. Prepare a region adjacency graph (RAG) for the image.
3. For each region in an image, perform the following steps:
   (a) consider its adjacent region, and test to see if they are similar
   (b) for regions that are similar, merge them and modify the RAG.

Continue of previous algorithm

means of pixel characteristics

Region-Based Merging

1. if $|\mu_1 - \mu_2| < T$ then merge
   can also do with surface fitting
2. merge based upon hypothesis testing (similar to Kullback Inf.)

hypothesis H_0: both set of pixels belong to same region

H_1: regions belong to different objects

this one is based on gray levels g_i

normally distributed in region

probability of graylevels within a region $\longrightarrow$ $$p(g_i) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(g_i-\mu)^2}{2\sigma^2}}$$

under hypothesis H_0 $$p(g_1, g_2, \ldots, \underbrace{g_{m_1+m_2}}_{\#\text{ of points}}) = \prod_{i=1}^{m_1+m_2} p(g_i \mid H_0)$$

probability that all these pixels in region H_0 is product of individual pixel's

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

all pixels have same dist.

This is for combined region
basically its the product of probabilities

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

these just add

just gets raised to power

the σ_0^2's cancel out

basically the result is a composite

$$= \frac{1}{\underbrace{(\sqrt{2\pi}\,\sigma_0)^{m_1+m_2}}_{\text{these just multiply}}} e^{-\left(\frac{m_1+m_2}{2}\right)}$$

where we used definitions

$$\mu = \frac{1}{n}\sum_{i=1}^{n} g_i$$

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

under hypothesis H_1 (different regions).

same cancelation

$$p(g_1, g_2, \dots, g_{m1+m2} \mid H_1) = \frac{1}{(\sqrt{2\pi}\,\sigma_1)^{m_1}} e^{-\frac{m_1}{2}} \bullet \frac{1}{(\sqrt{2\pi}\,\sigma_2)^{m_2}} e^{-\frac{m_2}{2}}$$

Likelihood ratio. which hypothesis best fits the data..

$$L = \frac{p(g_1, g_2, \dots \mid H_1)}{p(g_1, g_2, \dots \mid H_0)} = \frac{\sigma_0^{m_1+m_2}}{\sigma_1^{m_1}\,\sigma_2^{m_2}}$$

variance of combined region

variance of individual regions

∴ estimate standard deviations for the two regions and the combined region.
if $L <$ threshold then combine regions

Algorithm 3.6 Region Splitting
1. Form initial regions in the image
2. For each region in the image, recursively perform
   (a) compute variance in gray scale for region
   (b) if $\sigma > T$, split region along appropriate regions

This is quad-tree.

homogeneity measure

Algorithm 3.7 Split and merge segmentation
1. Start with entire image as a single region
2. Pick a region R. If $H(R)$ is false then split into 4 sub regions.
3. Consider any two or more neighboring subregions, $R_1, R_2, \dots, R_N$ in the image. If $H(R_1 \cup R_2 \cup \dots \cup R_n)$ is true then merge the n regions into a single region.
4. Repeat these steps until no further splits or merges take place.

problems

1. how to select initial seeds that represent regions of interest
2. what properties to use for region growing

one way in IR images - use brightest points (hottest targets) as seeds.
another was is to use histogram to pick most common values.

descriptions
- color
- texture
- intensity
- spatial properties such as moments.

} can all be used for seeds.

must also include connectivity & adjaincy info to get meaningful results.

problem is how to stop. Use

all require a priori knowledge {
- size
- moving average of a descriptor as a similarity measure
- shape

sample algorithm

1. $|f_i - f_j| < 10\% |\underbrace{max - mix}_{\text{entire image}}|$

2. any pixel added must be 8-connected to at least one pixel previously included.

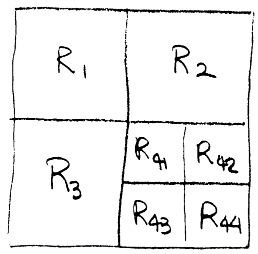
7.4.3 Region splitting & merging

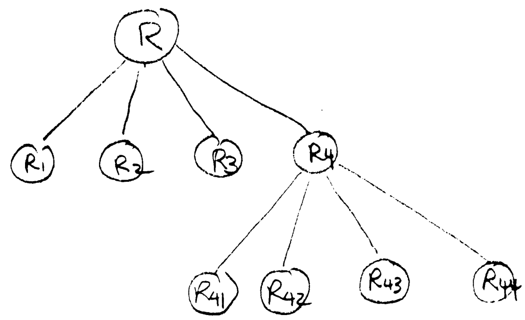
(1) split into four disjoint quadrants any region R_i where $P(R_i) = \text{FALSE}$
$\uparrow$ some logical predicate

(2) merge any adjacent regions for which $P(R_j \cup R_k) = \text{TRUE}$

(3) stop when no further merging or splitting is possible

quadtree





predicates do not need to be 100% to be true

e.g. $P(R_i) = \text{TRUE}$ if $\geq 80\%$ of pixels in R_i satisfy $|z_i - m_i| \leq 2\sigma_i$

mean of region i

std dev of region

basically an intro to texture which we will not do.