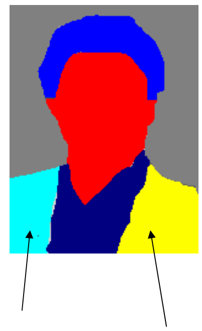
Region Segmentation

Idea

- Edge detection
  - Found boundaries between regions (edges)
  - Didn't return the actual region
- *Segmentation*
  - *Partition image into regions*
  - find regions based on similar pixel intensities, textures, etc. . .
  - very hard

Region vs. Edges



Different Regions

Basic Formulation

- Let R represent the entire image region. We want to partition R into *n* subregions, R_1, R_2, . . ., R_n, such that:
- (a) $\bigcup_{i=1}^{n} R_i = R$
- (b) Ri is a connected region for i=1, 2, . . . , n
- (c) $R_i \cap R_j = \phi$ for all i and $j, i \neq j$
- (d) P(Ri) = TRUE for i=1, 2, . . . n
- (e) $P(R_i \cup R_j) = FALSE$ for $i \neq j$

where P(Ri) is a logic predicate over the points in set Ri and ϕ is the empty set

Basic Formulation

- (a) segmentation must be complete
  - all pixels must belong to a region
- (b) pixels in a region must be connected
- (c) Regions must be disjoint
- (d) states that pixels in a region must all share the same property
  - The logic predicate P(Ri) over a region must return TRUE for each point in that region
- (e) indicates that regions are different in the sense of the predicate P.

Region Segmentation Problem

- Very difficult task
  - Application specific
  - May need magic numbers
  - May need user to select starting points
- All-purpose generic algorithm
  - Rarely gives the desired results

Common Approach

- Pixel Aggregation
  - Start with some seed points
  - From these seeds
    - grow region by appending neighbor pixels
    - choose pixels that have similar property;

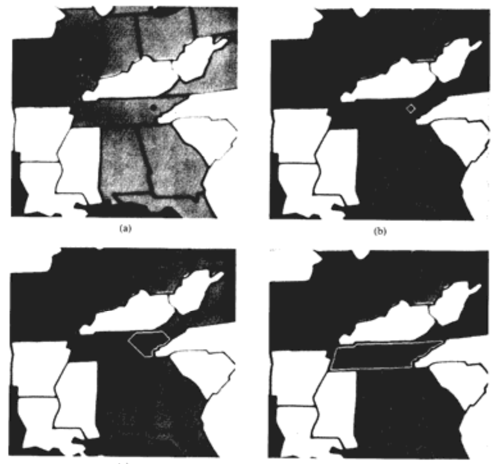
P(Ri) = TRUE

P(Ri) can be thought of as a similarity measurement

Similarity Measure

- 1st. Compare Candidates to Seed Pixel
  - Algorithm
    - Is = f(x,y) -- f(x,y) is the seed
    - do while
      - examine N8 neighbors in region
      - if |f(N8) –Is| < T
        - » add pixel to region
    - repeat until no more pixels can be added

Example



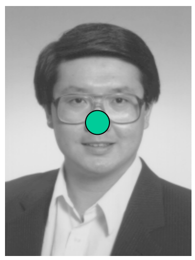
- *matlab example*

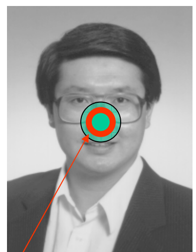
Similarity Criteria

- 2nd Use *Neighborhood of R*
  - *Not just seed pixel, but similarity to region pixels (border pixels)*
  - Algorithm
    - do while
      - if |f(N8) –f(x,y)| < T
        - » add pixel to region
    - repeat until no more pixels can be added

Similarity Criteria

- 2nd Use *Neighborhood of R*





This allows the region to grow given gradual intensity change.

The rate of *change* allowed is controlled by a threshold, T.

Check for similarity using border pixels

Comparing to Region Statistics

- 3rd. Compare candidate pixels to some information specific to the entire region
- For example, the mean intensity of all the pixels currently in Ri
  - seed dominates at first
  - mean is allowed to drift
    - this is often referred to as:
      - *centroid* region growing

Multiple Seeds

- User gives multiple seeds
  - This gives us a starting mean value
  - AND, a variance
- Use this mean and variance combination to determine a predicate P(Ri)
  - Use of mean and variance often try to find regions with a certain "texture"

Use of Counterexamples

- User can give two sets of inputs
  - Region Seeds
    - Pick pixels like this region (region you want to segment)
    - Ie, multiple seeds
  - Counterexamples
    - Seeds of pixels not in the region
- Used combined predicates to choose candidate pixels

Region Growing

- Add Heuristics when to stop growing
  - Gradient Magnitude
    - $|f(x,y) - f(n8)| < T$ && $f(n8) < Tm$
  - Edge Boundary
    - Run a canny detector
    - If a point is on a boundary, it can't be added to the region
  - Application specific

Region Growing Algorithms

- Similarity measure is the key to success
  - We have seen some examples
    - Use original seed
    - Boundary Neighbors
    - Region Statistics
    - Multiple Seeds
    - Counterexamples
- You can use any heuristic that gives you a reasonable P(Ri) for the given application
  - Results must satisfy the basic formulation given on slide 4.

Previous Example

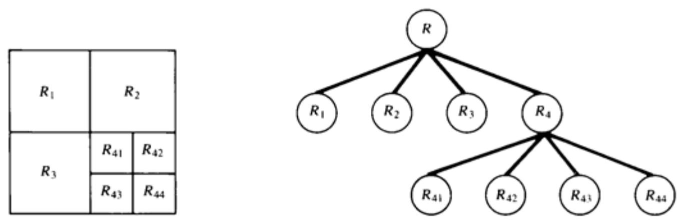
- User must specify seed points
- Different seed points will give different results
- We want a more automated approach

Region Splitting

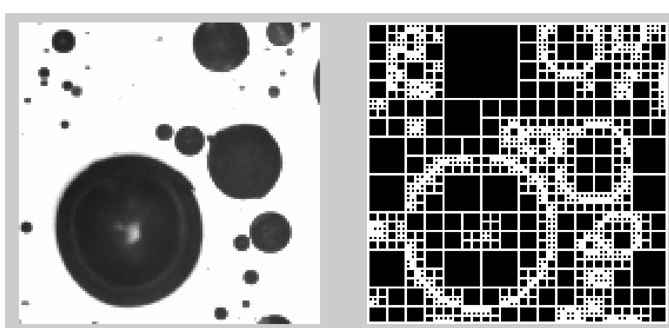
- Split the image into regions
- If the entire region doesn't satisfy the predicate P(Ri)
  - split it into smaller regions
  - repeat
- Use the quad-tree data structure

Quad Tree

- Data Structure
  - Each root has 4 children
  - Encodes a 2D spatial relationship



Quad-tree Example



Input Image　　QT decomposition

Matlab example

Problem with Splitting Alone

- Neighboring regions could have the same property
- We would like to merge these regions into a single region
  - Introduce a merge step into the algorithm
  - This a split and merge approach

Split and Merge Segmentation

- Split image into regions
- After each split, try to merge regions with similar P(R)=True
- If region can't be merged, and all pixels in this region P(Ri) != TRUE
  - subdivide region further
- Repeat

Split and Merge Example

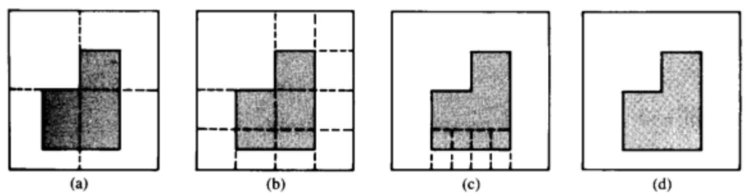


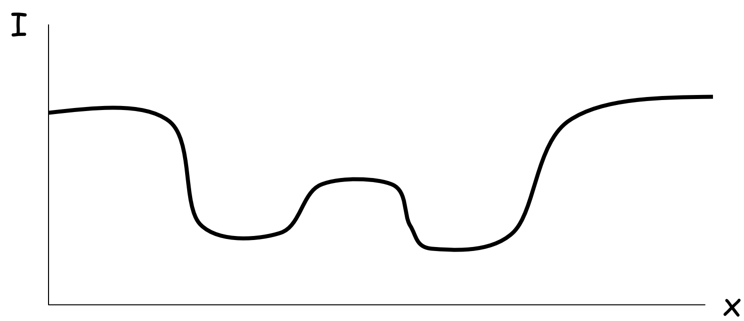
Figure 7.38 *Example of split-and-merge algorithm. (From Fu, Gonzalez, and Lee [1987].)*

Split and Merge

- Tries to eliminate the need for seeds
  - Sort of an all purpose algorithms
- Still requires a predicate P(Ri)
  - This can require a magic number
  - P(Ri) needs to be fairly generic
- The key to this algorithm is how to merge the regions

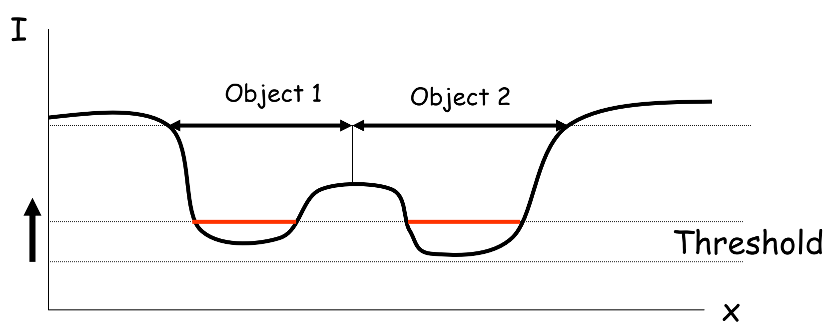
Watersheds Algorithm

- Also called "*Catch basin algorithm*"



Profile of a 1D scan line

Watersheds Algorithm



Profile of a 1D scan line

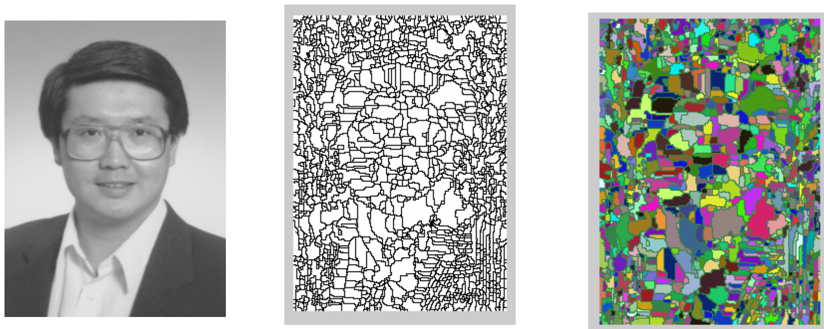
Algorithm

- Assume a dark object on a light background (gray levels = [0-255])
- Start with Threshold = 0
  - All pixels that are 0 form a new watershed (or basin)
  - Connected pixels are combined
- Increment Threshold
  - For all new pixels that are equal to this threshold
    - if they are neighbors to existing watersheds; combine with that watershed
    - Otherwise they form new watersheds
    - *If two watersheds meet, they cannot be merged!*

Watersheds of the Gradient

- Take the gradient magnitude of an image
- Start basins at local minimums
  - (seed watersheds)
- Apply the watersheds algorithm

Example



Watersheds algorithm

Watersheds Algorithm

- Watersheds often results in an *over segmented* image
  - Apply some merge heuristic
  - Generally, the merge algorithm is the "key" to the approach
    - And is very application specific

Segmentation from Motion

- Consider Temporal Images
  - f(x,y,t)
    - t is successive frames
    - often t is time (video frames)
- We can use the temporal frames to try to segment out moving objects

Temporal Differencing

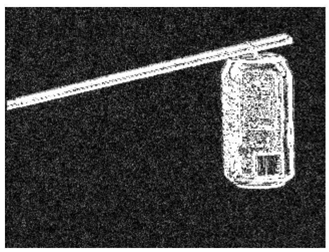
- For each temporal frame
  - Idiff(x,y) = I(x,y,t-1) – I(x,y,t)
  - for each pixel
    - if |Idiff(x,y)| > T
      - change occurred

- *Above is a backwards difference*
- *You could also compute a forward difference*

Temporal Differencing Example

I(x,y,t-1)

I(x,y,t)

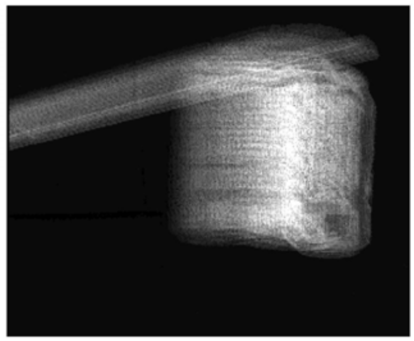


|It-1 – It| * 5
(scaled to emphasize change)

Accumulative Difference

- Threshold based on the number of times a pixel changes
  - Use an accumulation buffer
  - display the buffer
    - This tries to reduce the errors from noise
    - Avoid difficult global threshold

Accumulative Difference



accBuf(x,y)++ if (|It-1 – It| > 5)
Performed over 49 Frames

Alternative

- Compute difference from a reference frame R
- Reference frame is known in advance
- Or, reference frame is computed
  - For example
  - Take the median of each pixel over 100 frames
    - Statistically this provides a reasonable reference frame
      - Assuming a stationary camera

Summary

- Region-based Segmentation
  - Region Growing
    - User supplies seed (or seeds)
    - Similarity Criteria is the key
  - Split and Merge Approach
    - Quad-tree data structure
  - Watershed algorithm
  - Segmentation using Motion

Active Research Areas

- Application specific segmentation
  - Especially in the medical community