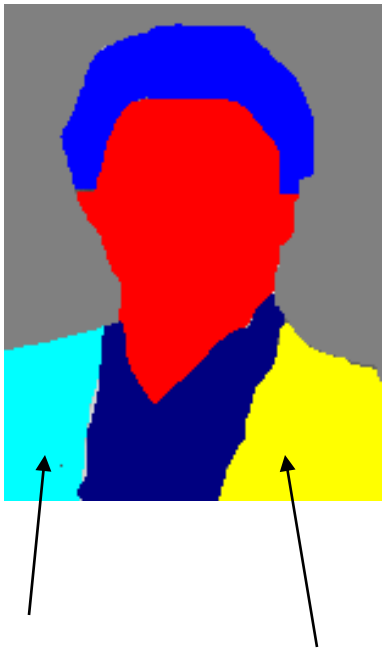
Region Segmentation

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, $R_1, R_2, \ldots, R_n$, such that:
- (a) $\bigcup_{i=1}^{n} R_i = R$
- (b) Ri is a connected region for i=1, 2, . . . , n
- (c) $R_i \bigcap R_j = \phi$ for all i and $j, i \neq j$
- (d) P(Ri) = TRUE for i=1, 2, . . . n
- (e) $P(R_i \bigcup R_j) = FALSE$ 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;

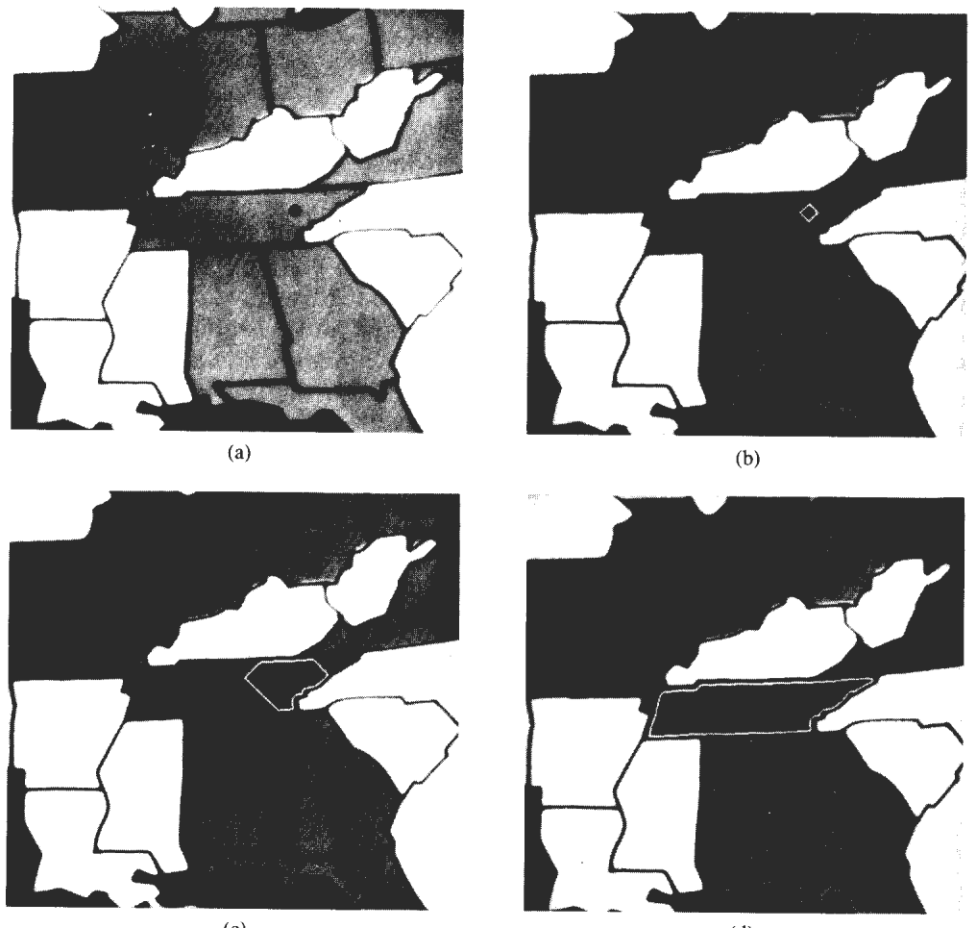
$$\boxed{P(R_i) = \text{TRUE}}$$

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



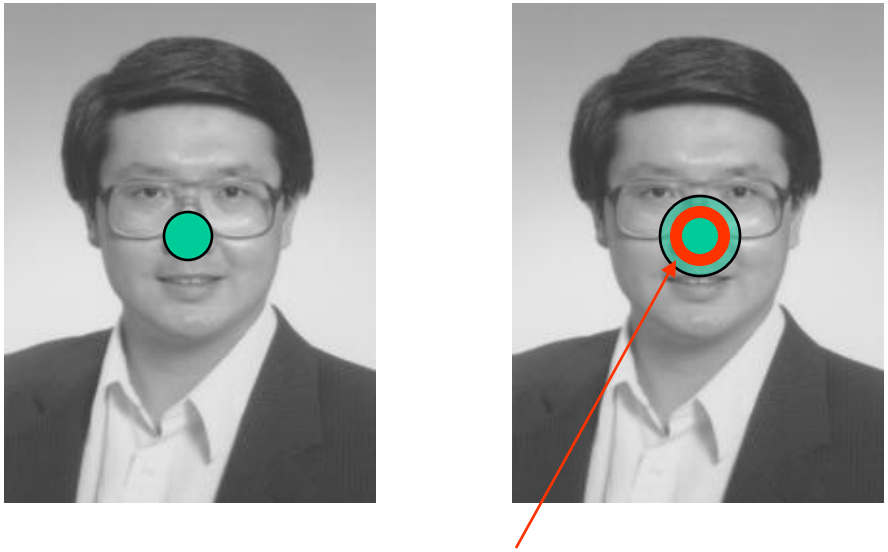
- *matlab example*

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*



Check for similarity using border pixels

This allows the region to grow given gradual intensity change.

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

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
    - $|f(x,y) - f(n8)| < T \quad \&\& \quad f(n8) < Tm$
  - 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

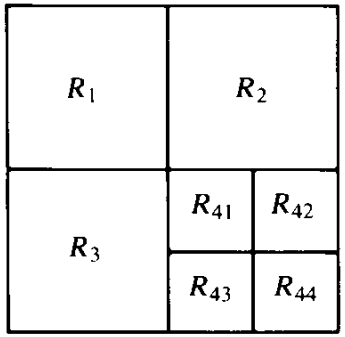
Region Splitting

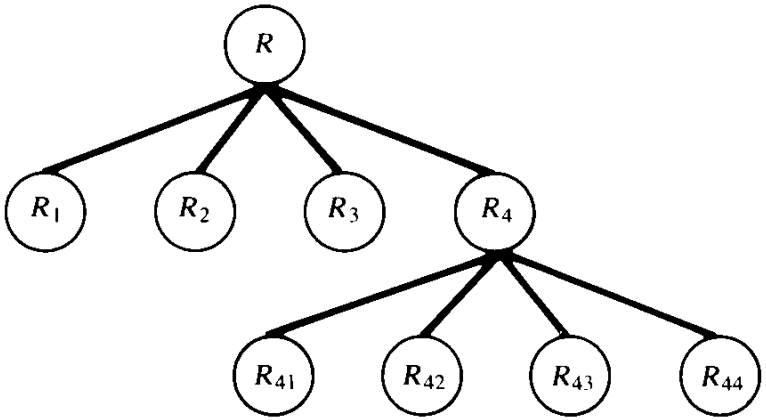
- Split the image into regions
- If the entire region doesn't satisfy the predicate P(Ri)
  - split it into smaller regions
  - repeat

- Use the quad-tree data structure

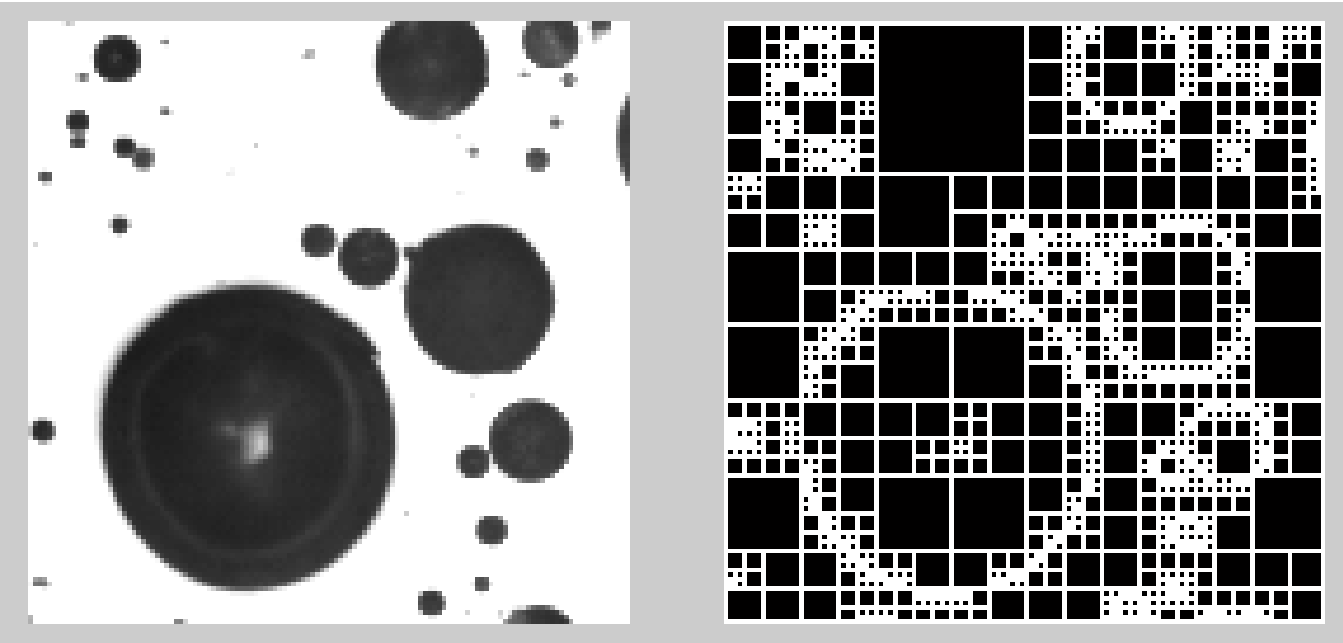
Quad Tree

- 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

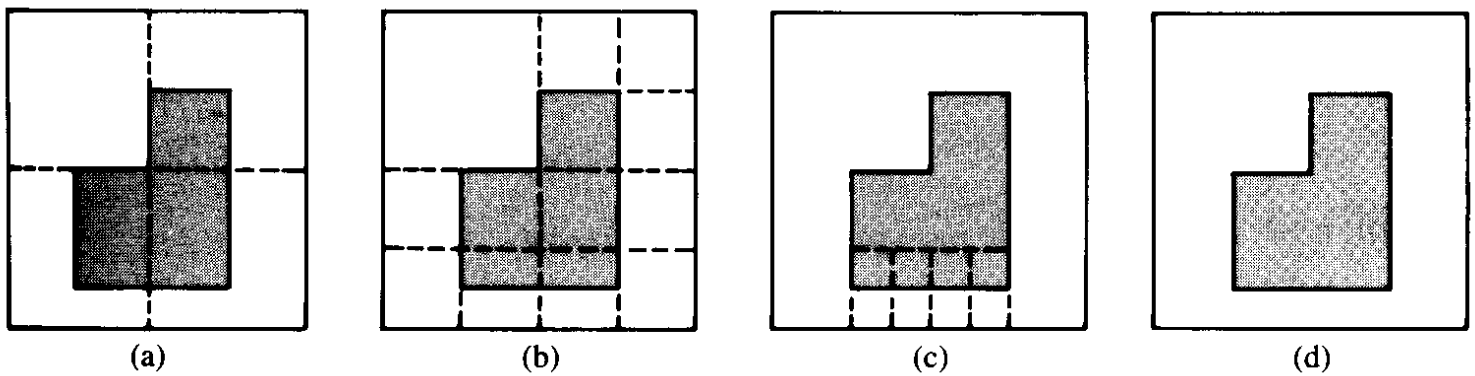


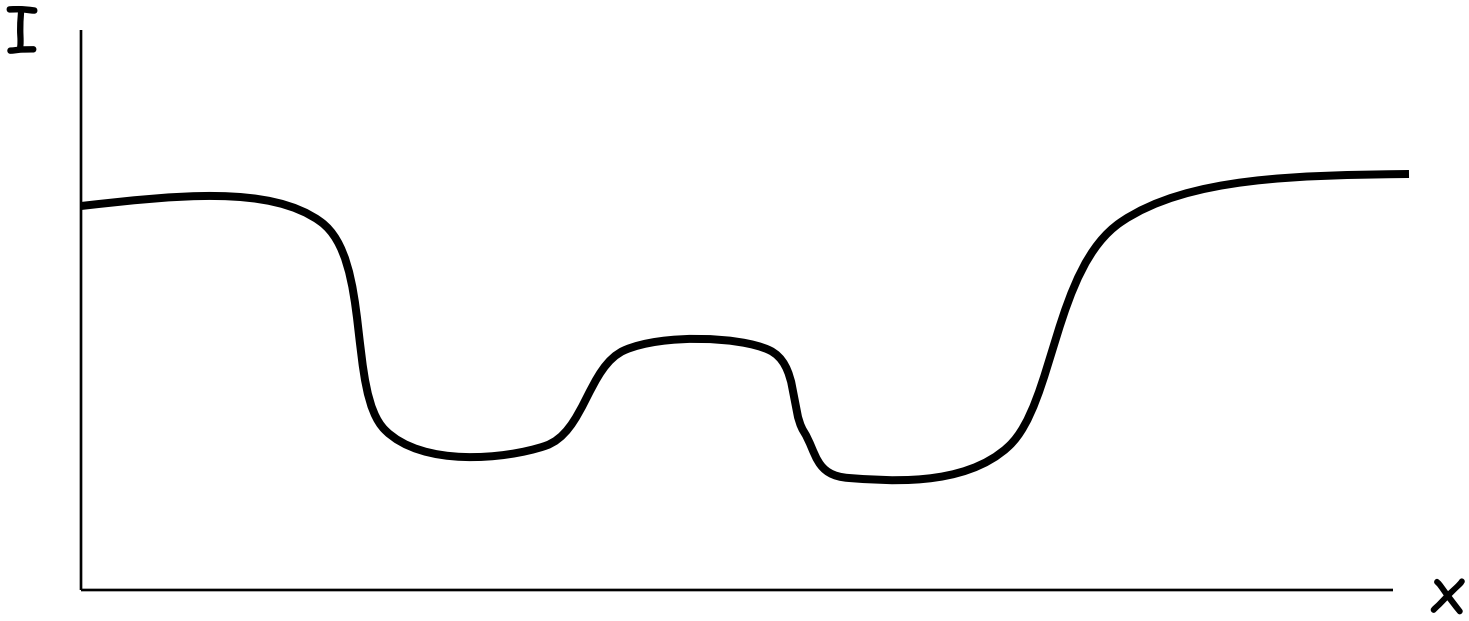
Figure 7.38 *Example of split-and-merge algorithm. (From Fu, Gonzalez, and Lee [1987].)*

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

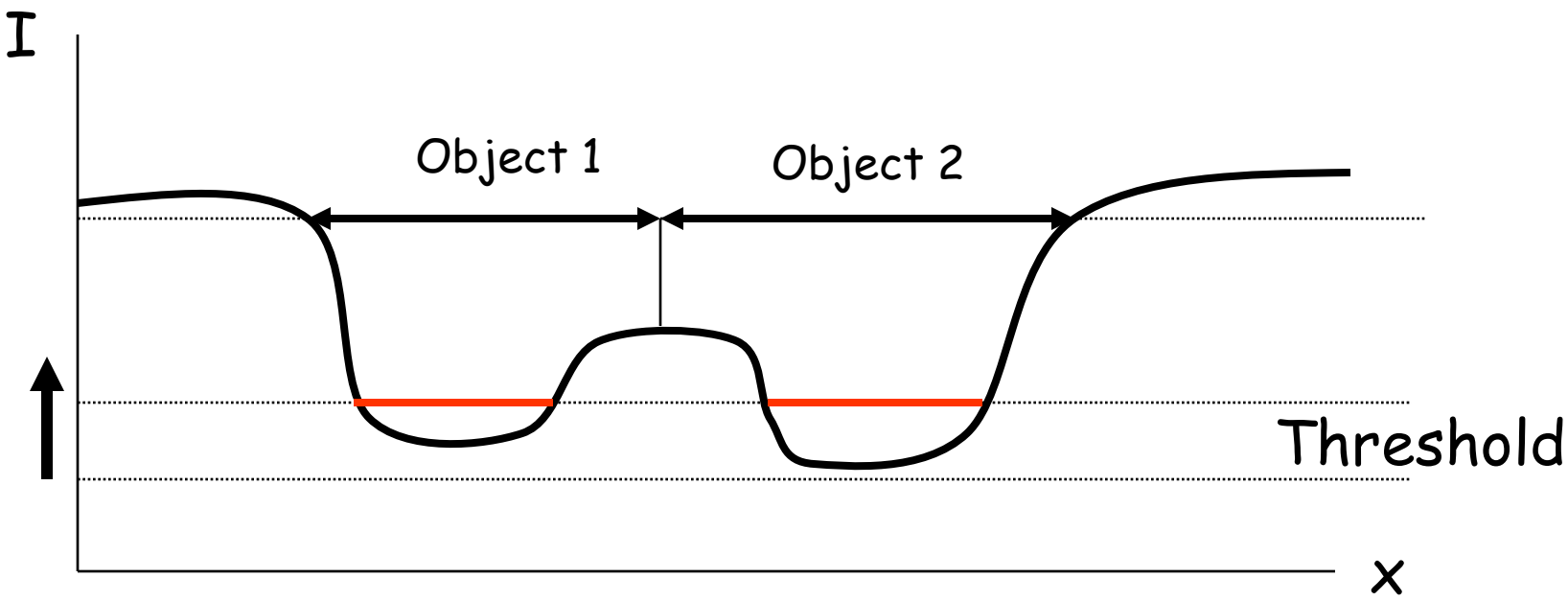
Watersheds Algorithm

- 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])

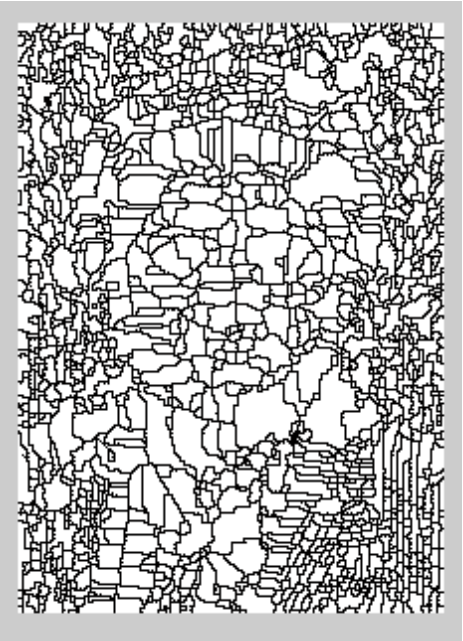
- Start with Threshold = 0
  - All pixels that are 0 form a new watershed (or basin)
  - Connected pixels are combined

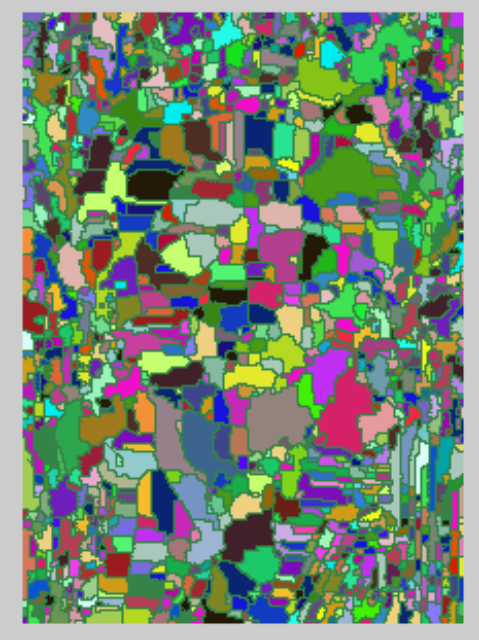
- Increment Threshold
  - For all new pixels that are equal to this threshold
    - if they are neighbors to existing watersheds; combine with that watershed
    - Otherwise they form new watersheds
    - *If two watersheds meet, they cannot be merged!*

Watersheds of the Gradient

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

Example





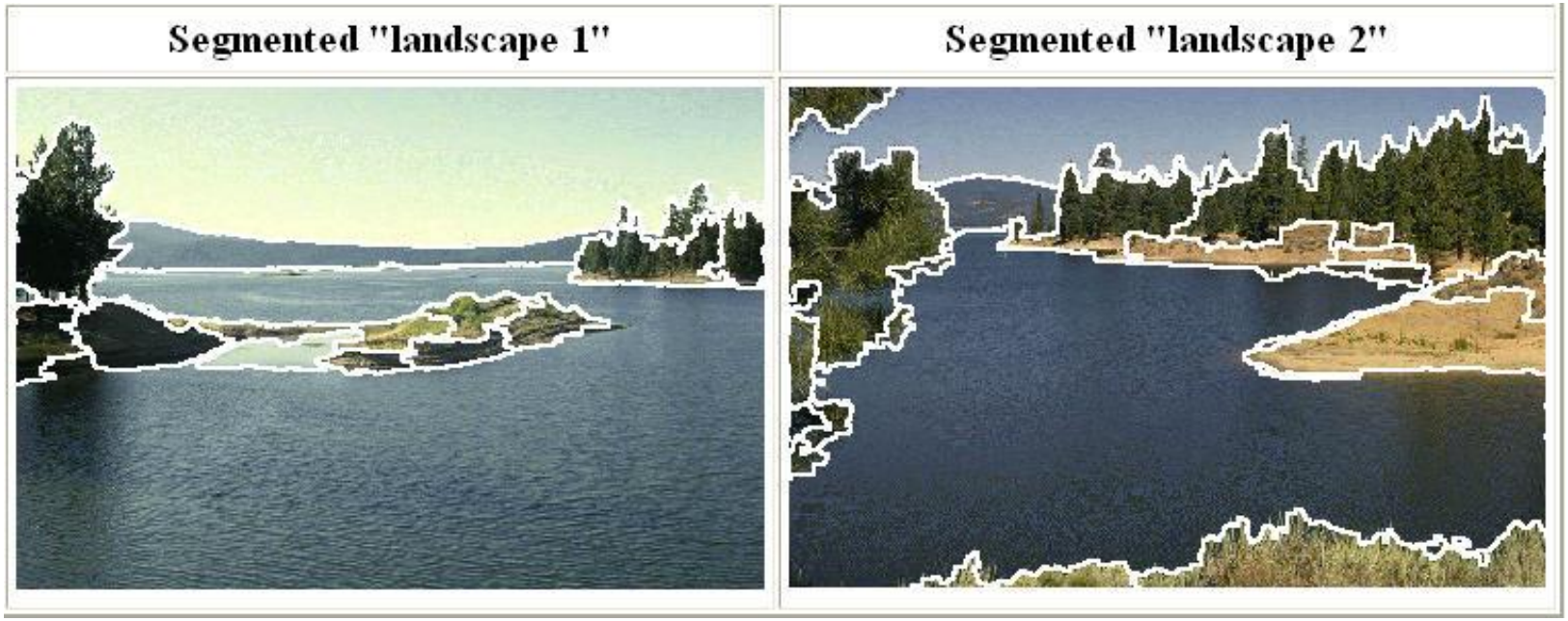
Watersheds algorithm

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

Mean Shift Segmentation

- Perhaps the best technique to date…



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResu

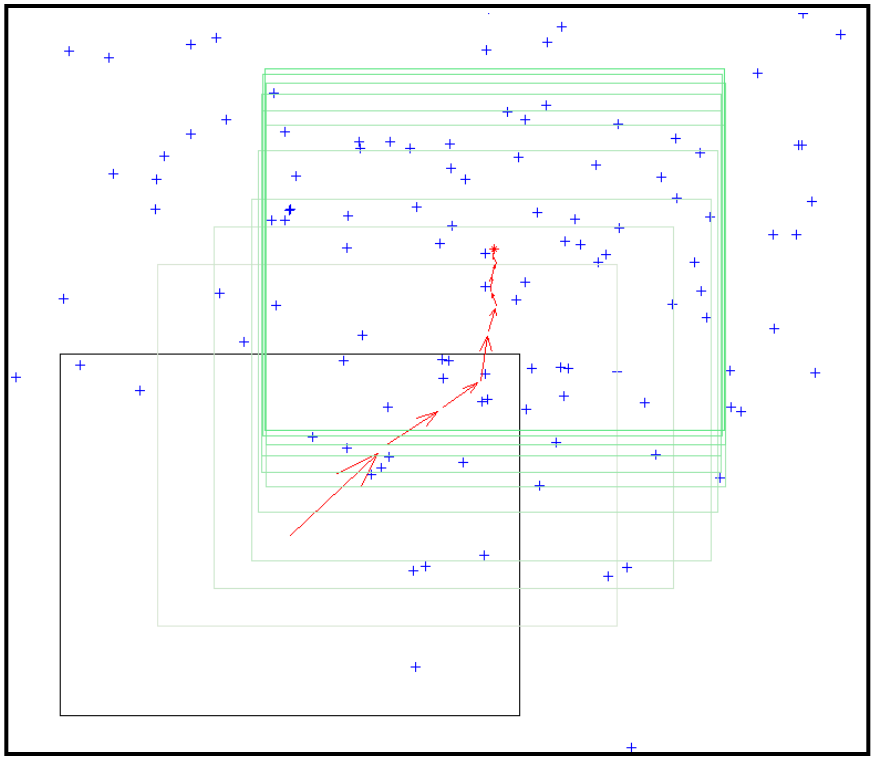
*Slides from Stanford CS223-class

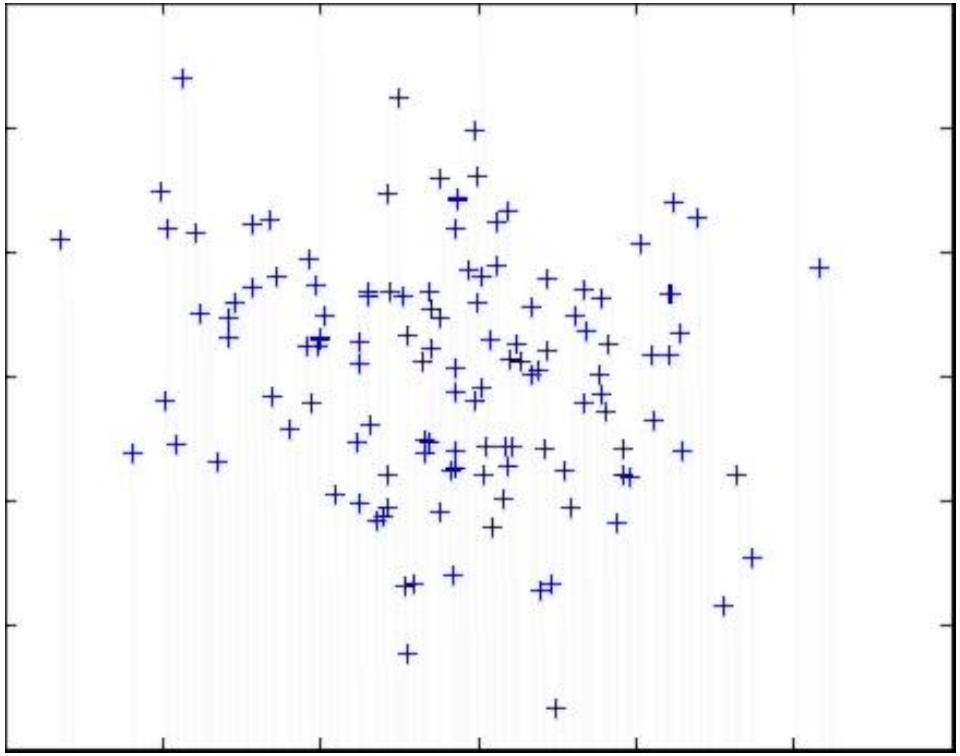
Mean Shift Algorithm

Mean Shift Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the "mode" or point of highest density of a data distribution:

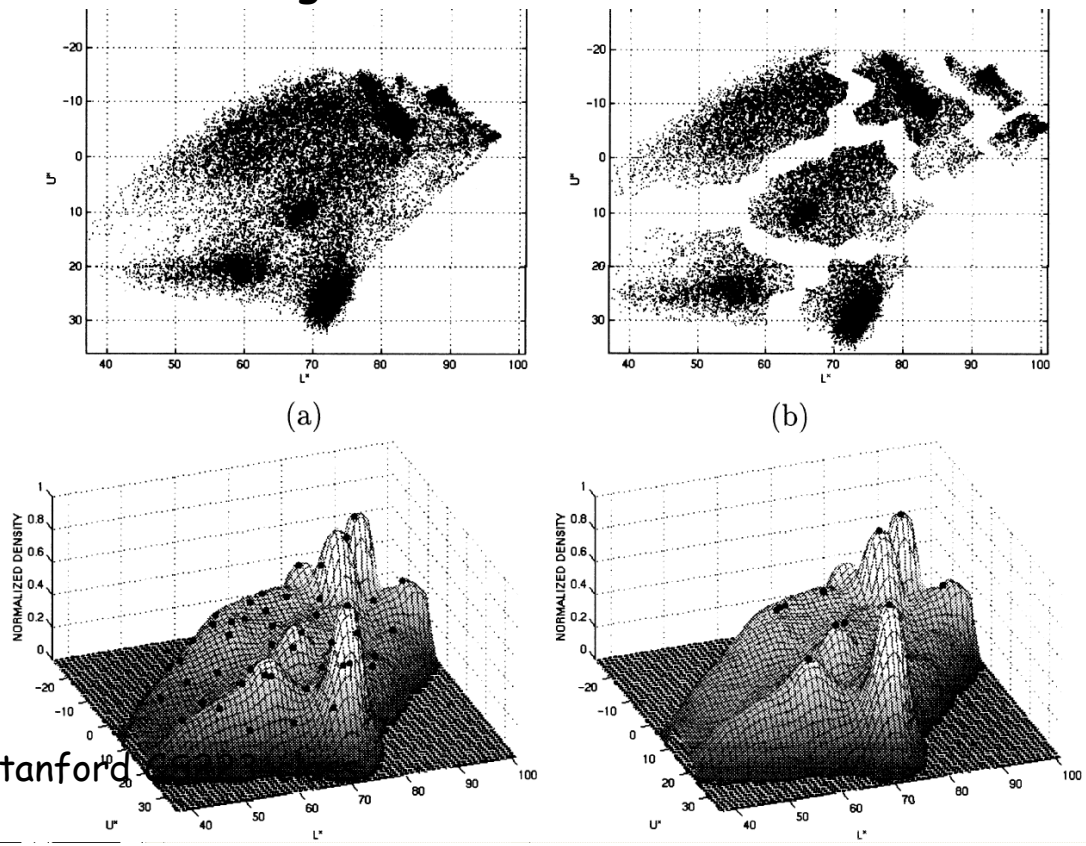




Mean Shift Segmentation

Mean Shift Setmentation Algorithm

1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same "peak" or mode.
5. The data these merged windows traversed are clustered together.



*Slides from Stanford

*Image From: Dorin Comaniciu and Peter Meer, Distribution Free Decomposition of Multivariate Data, Pattern Analysis & Applications (1999)2:22–30

Mean Shift Segmentation Extension

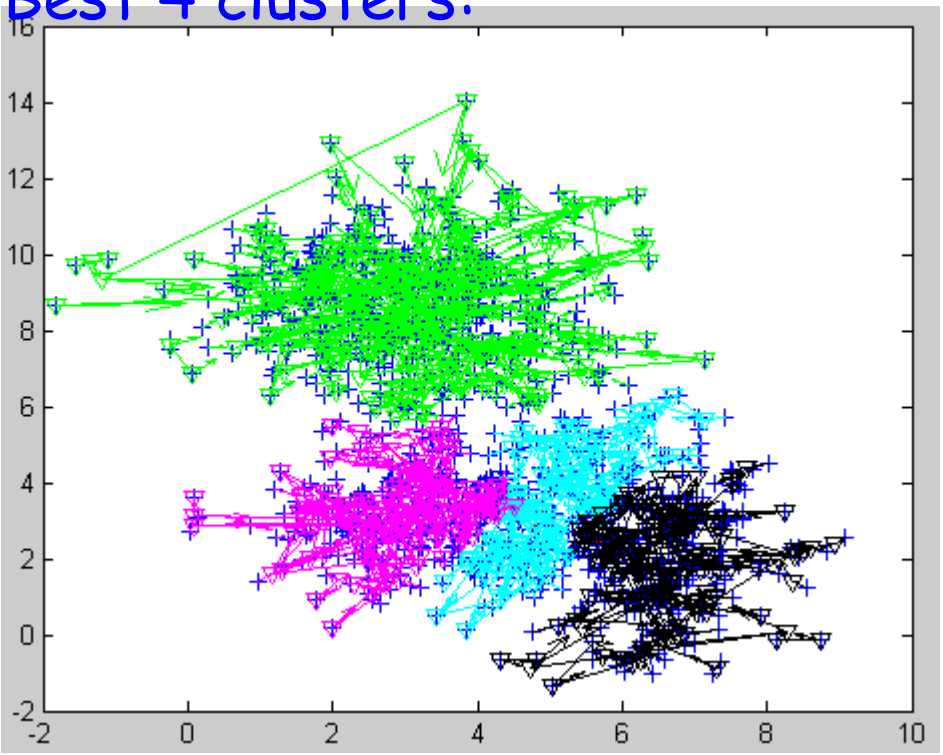
Is scale (search window size) sensitive. Solution, use all s

Gary Bradski's internally published <u>agglomerative clustering</u> extension:

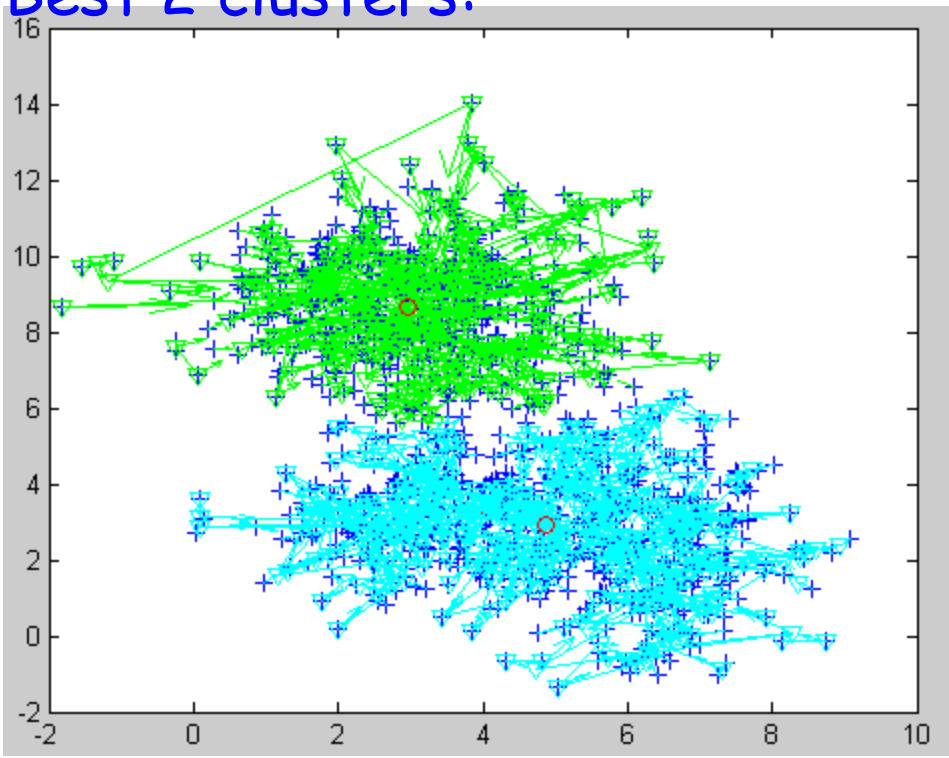
Mean shift dendrograms

1. Place a tiny mean shift window over each data point
2. Grow the window and mean shift it
3. Track windows that merge along with the data they transversed
4. Until everything is merged into one cluster

Best 4 clusters:

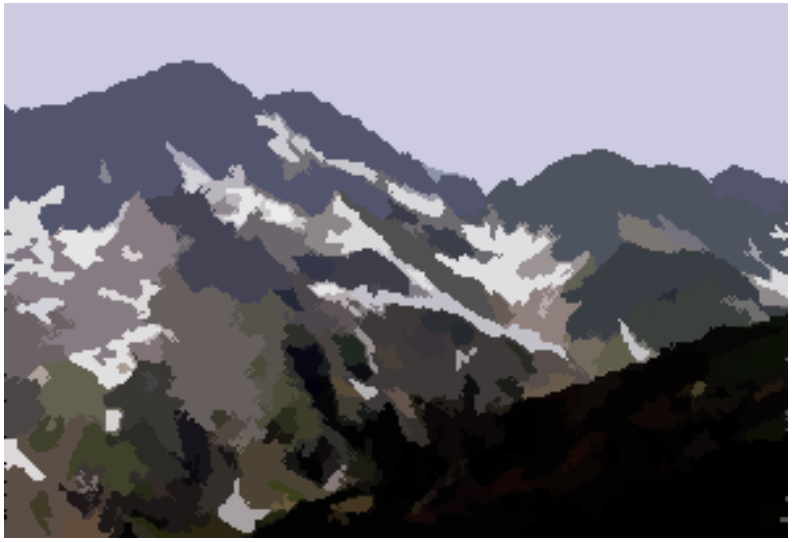


Best 2 clusters:



Advantage over agglomerative clustering: Highly paralleli

Mean Shift Segmentation Results:



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPami

K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error
- can't do this by search, because there are too many possible allocations.
- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

* From Marc Pollefeys COMP 256 2003

K-Means

```
Choose k data points to act as cluster centers

Until the cluster centers are unchanged

    Allocate each data point to cluster whose center is nearest

    Now ensure that every cluster has at least
    one data point; possible techniques for doing this include .
    supplying empty clusters with a point chosen at random from
    points far from their cluster center.

    Replace the cluster centers with the mean of the elements
    in their clusters.

end
```

Algorithm 16.5: *Clustering by K-Means*

* From Marc Pollefeys COMP 256 2003

Image Segmentation by K-Means

- Select a value of K
- Select a feature vector for every pixel (color, texture, position, or combination of these etc.)
- Define a similarity measure between feature vectors (Usually Euclidean Distance).
- Apply K-Means Algorithm.
- Apply Connected Components Algorithm.
- Merge any components of size less than some threshold to an adjacent component that is most similar to it.

* From Marc Pollefeys COMP 256 2003

Results of K-Means Clustering:

Image

Clusters on intensity

Clusters on color

K-means clustering using intensity alone and color alone

* From Marc Pollefeys COMP 256 2003

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
  - Mean-shift and K-mean algorithms

Active Research Areas

- Application specific segmentation
  - Especially in the medical community