EEE-6512: Image Processing and Computer Vision

November 13, 2017
Lecture #10: Segmentation

Damon L. Woodard
Dept. of Electrical and Computer Engineering
dwoodard@ece.ufl.edu

Chapter Outline

- Overview
- Thresholding
- Gestalt Psychology
- Image Segmentation
- Graph-Based Methods

Image Segmentation

Segmentation as partitioning

- A *partition* of image is collection of sets $S_1, .., S_N$ such that

  $I = S_1 \cup S_2 \ldots \cup S_N$ (sets cover entire image)

  $S_i \cap S_j = 0$ for all $i \neq j$ (sets do not overlap)

- A *predicate* $H(S_i)$ measures region *homogeneity*

  $H(R) = \begin{cases} \textit{true} & \text{if pixels in region } R \text{ are similar} \\ \textit{false} & \text{otherwise} \end{cases}$

- We want
  1. Regions to be homogeneous

     $H(S_i) = \textit{true}$ for all i

  2. Adjacent regions to be different from each other

     $H(S_i \cup S_j) = \textit{false}$ for all adjacent S_i, S_j

Two approaches

- **Splitting**
  (Divisive clustering)
  - start with single region covering entire image
  - repeat: split into homogeneous regions

even better:
repeat: split cluster to yield two distant components (difficult)

Property 2 is always true:
$H(S_i \cup S_j)$ = *false* for adjacent regions

Goal is to satisfy Property 1:
$H(S_i)$ = *true* for every region

- **Merging**
  (Agglomerative clustering)
  - start with each pixel as a separate region
  - repeat: merge adjacent regions if union is homogeneous
  - even better:
    repeat: merge two closest clusters

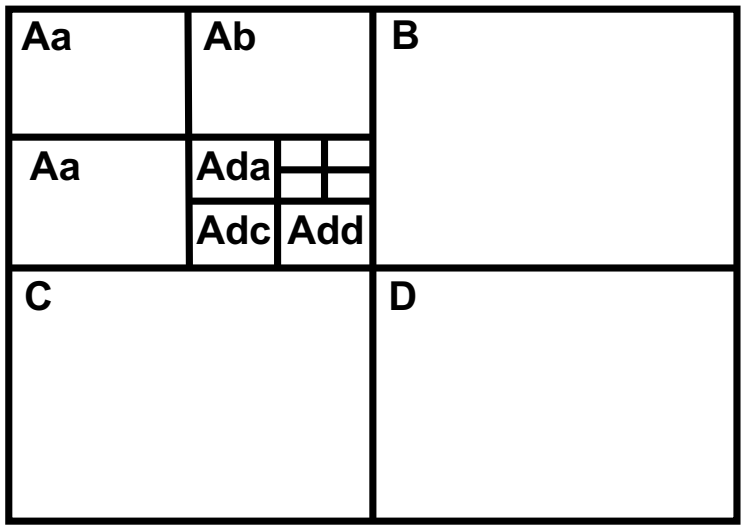
Property 1 is always true:
$H(S_i)$ = *true* for every region

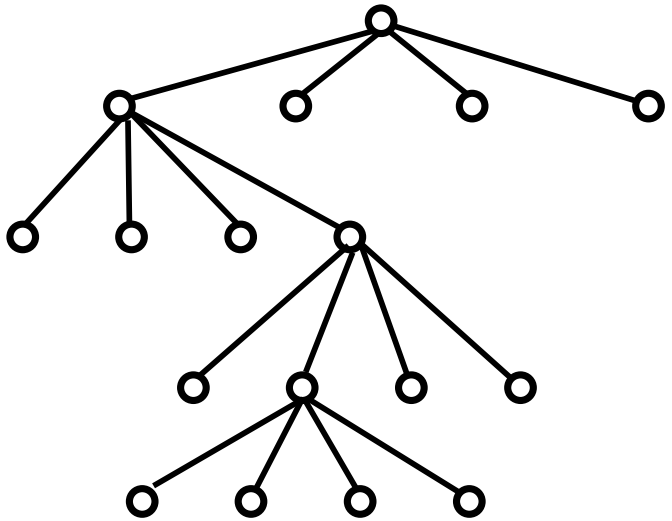
Goal is to satisfy Property 2:
$H(S_i \cup S_j)$ = *false* for adjacent regions

In practice, merging works much better than splitting

Region splitting

- Start with entire image as a single region
- Repeat:
  - Split any region that does not satisfy homogeneity criterion into subregions
- Quad-tree representation is convenient





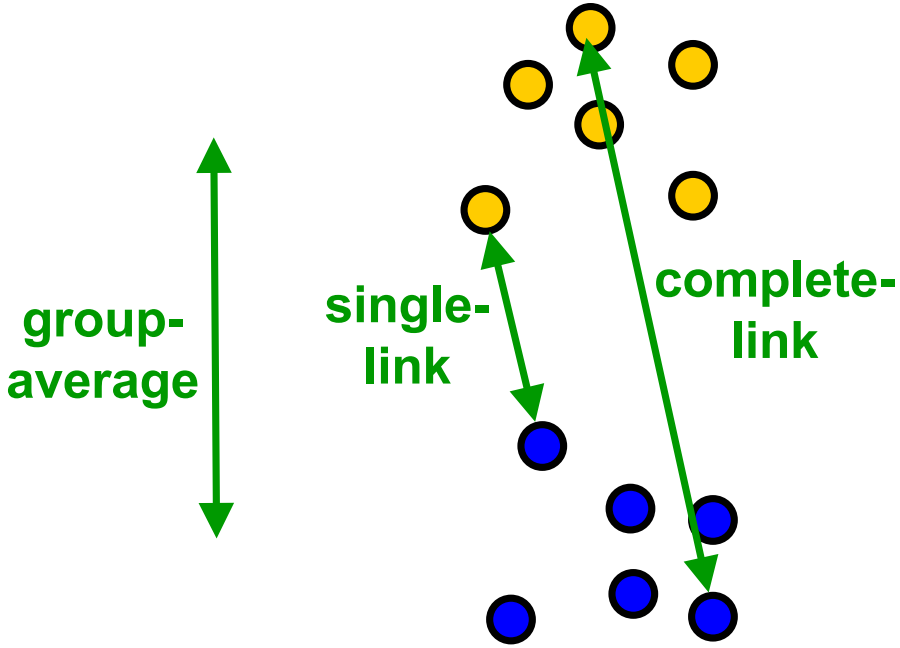
Region growing

- Start with (random) seed pixel as cluster
- Repeat:
  - Aggregate neighboring pixels that are similar to cluster model
  - Update cluster model with newly incorporated pixels
- This is a generalized floodfill
- When cluster stops growing, begin with new seed pixel and continue
- An easy cluster model:
  - Store mean and covariance of pixels in cluster
  - Use Mahalanobis distance to cluster
    This leads to a natural threshold, e.g., ± 2.5 σ
  - Update mean and covariance efficiently by keeping track of sum(x) and sum(x^2)
- One danger: Since multiple regions are not grown simultaneously, threshold must be appropriate, or else early regions will dominate

When to merge two clusters

Inter-cluster distance can be computed by

- **single-link clustering (dist. b/w closest elements) allows adaptation (but drift)**
- **complete-link clustering (dist. b/w farthest elements) avoids drift**
- **group-average clustering (use average distance) good compromise**
- **root clustering (dist. b/w initial points of clusters) variation on complete-link**

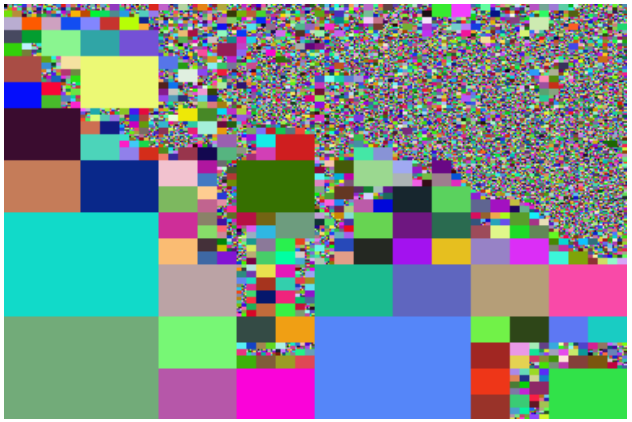


Region growing results

Split-and-Merge

- Split-and-merge algorithm combines these two ideas
  - Split image into quadtree, where each region satisfies homogeneity criterion
  - Merge neighboring regions if their union satisfies criterion (like connected components)

image



after split

after merge

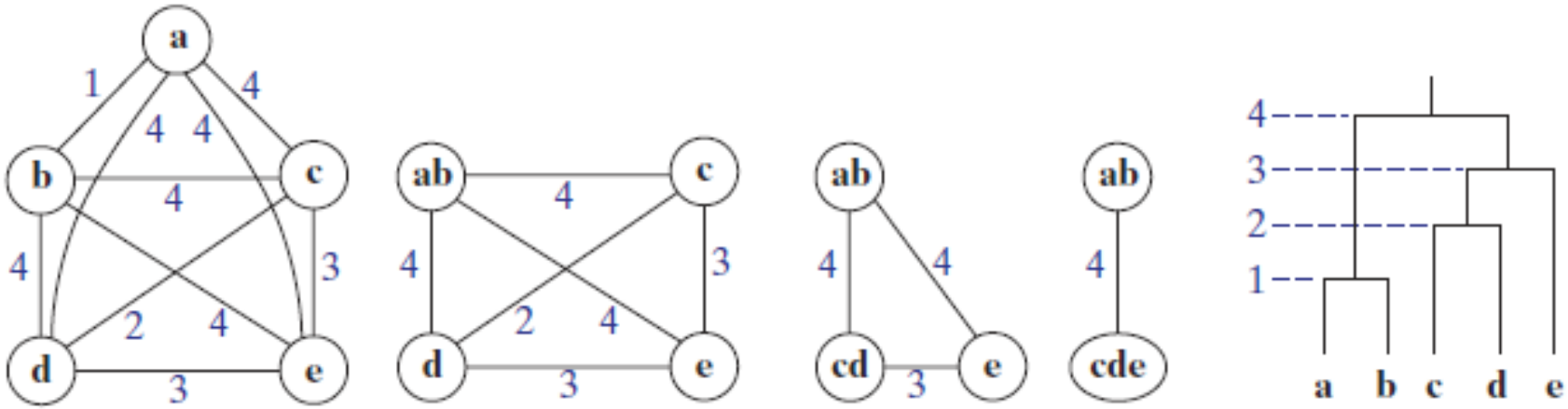
S. L. Horowitz and T. Pavlidis, Picture Segmentation by a Tree Traversal Algorithm, 1976

Hierarchical Clustering Scheme (HCS)

- Segmentation can be viewed as a specific application of data clustering.
- **Hierarchical clustering scheme (HCS):** a popular approach to data clustering.
- An HCS operates on a fully connected graph containing a vertex for each data point.
- The weights of the edges are the distances between the data points as computed in some feature space.

Hierarchical Clustering Scheme (HCS) (cont.)

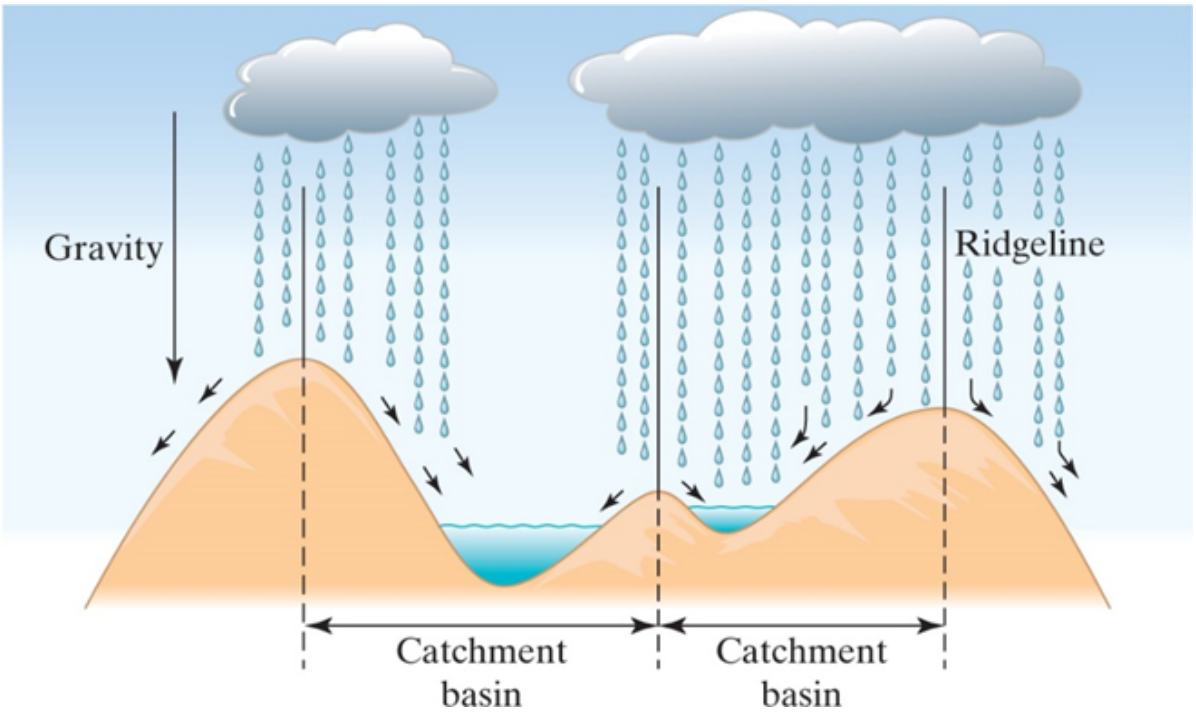
Figure 10.24 Left: An example of five data points labeled a through e, viewed as a graph with the edge weights indicating the distances (in some feature space) between the points. Middle columns: Initially considering each data point as a separate cluster, sequential iterations of the HCS procedure merge the two closest clusters until all clusters have been merged. Because the weights satisfy the ultrametric inequality, no updating of the weights is needed. Right: The dendrogram is a way to visualize the resulting hierarchical clustering, with the original data points along the horizontal axis and the distances used for merging along the vertical axis.



Watershed Methods

Watershed Method

Figure 10.27 In watershed segmentation, the segmentation function is interpreted as a topographical surface. The most common choice for the segmentation function is the gradient magnitude image.



Watershed Method

- **Watershed segmentation:** combines ideas from region growing and edge detection.

- The watershed approach to segmentation determines the segmented regions of the original image by computing the catchment basins of the terrain indicated by the **segmentation function**.

- The typical choice for the segmentation functions is the **gradient magnitude of image.**

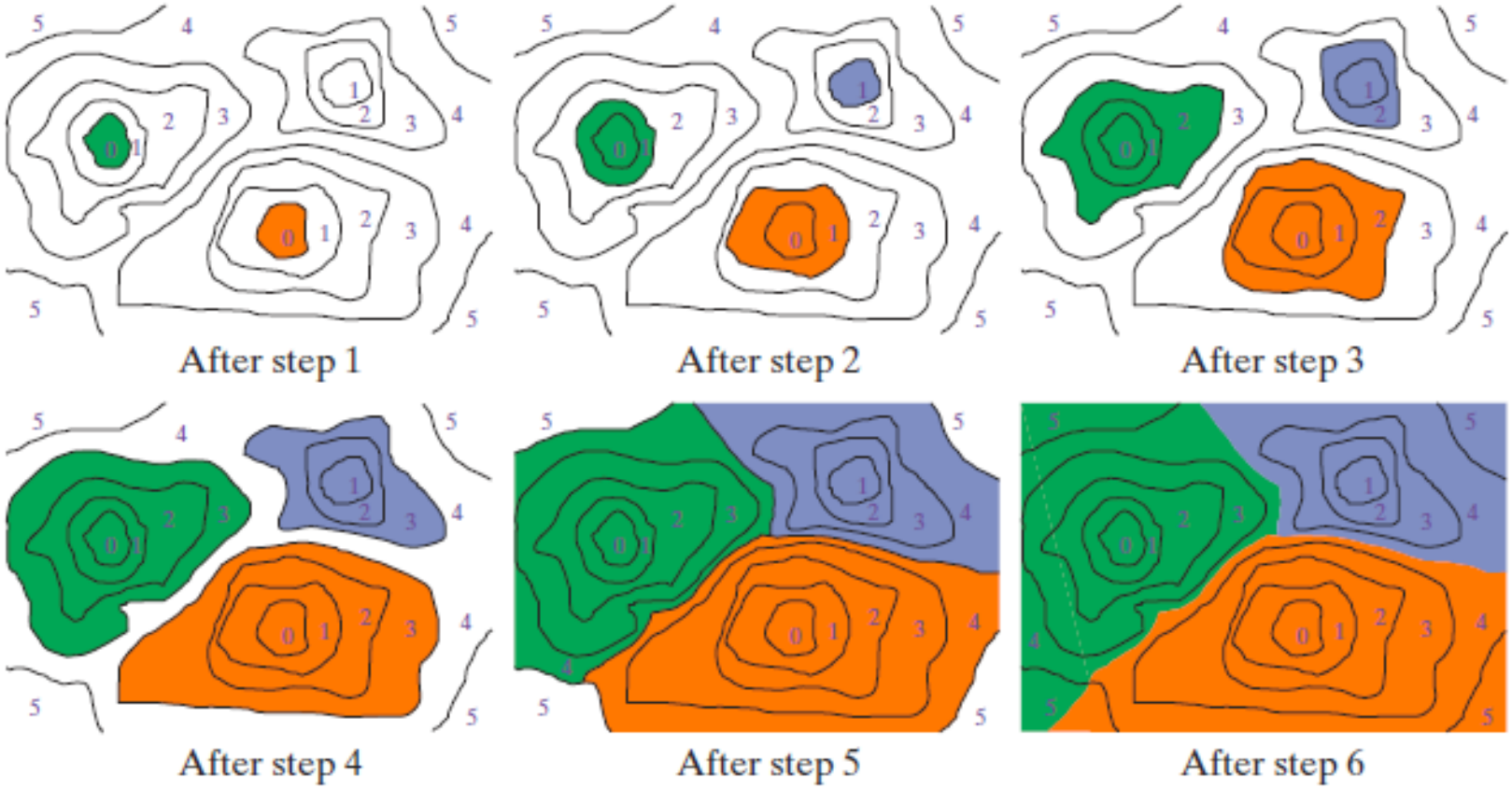
- **Objective:** To find the watershed lines separating regions

Watershed algorithms

- **Water immersion (Vincent-Soille)**
  - Puncture hole at each local minimum, immerse in water
  - Grow level by level, starting with dark pixels
  - *Sorting step:* For efficiency, precompute for each graylevel a list of pixels with that graylevel (histogram with pointers)
  - *Flooding step:* Then, repeat:
    - Breadth-first search (floodfill) of level g+1 given flooding up to level g
    - For each pixel with value g+1, either assign to closest catchment basin or declare new catchment basin (geodesic influence zone)
- **Tobogganing**
  - Find downstream path from each pixel to local minimum
  - Difficult to define for discrete (quantized) images because of plateaus

Watershed Method (cont.)

Figure 10.28 Step-by-step results of immersion-based watershed segmentation on a segmentation function with 6 levels (0 through 5). The different colors indicate the unique labels of the three different regions. The contour lines of the segmentation function are shown, with numbers indicating the levels of the pixels.



Watershed Method (cont.)

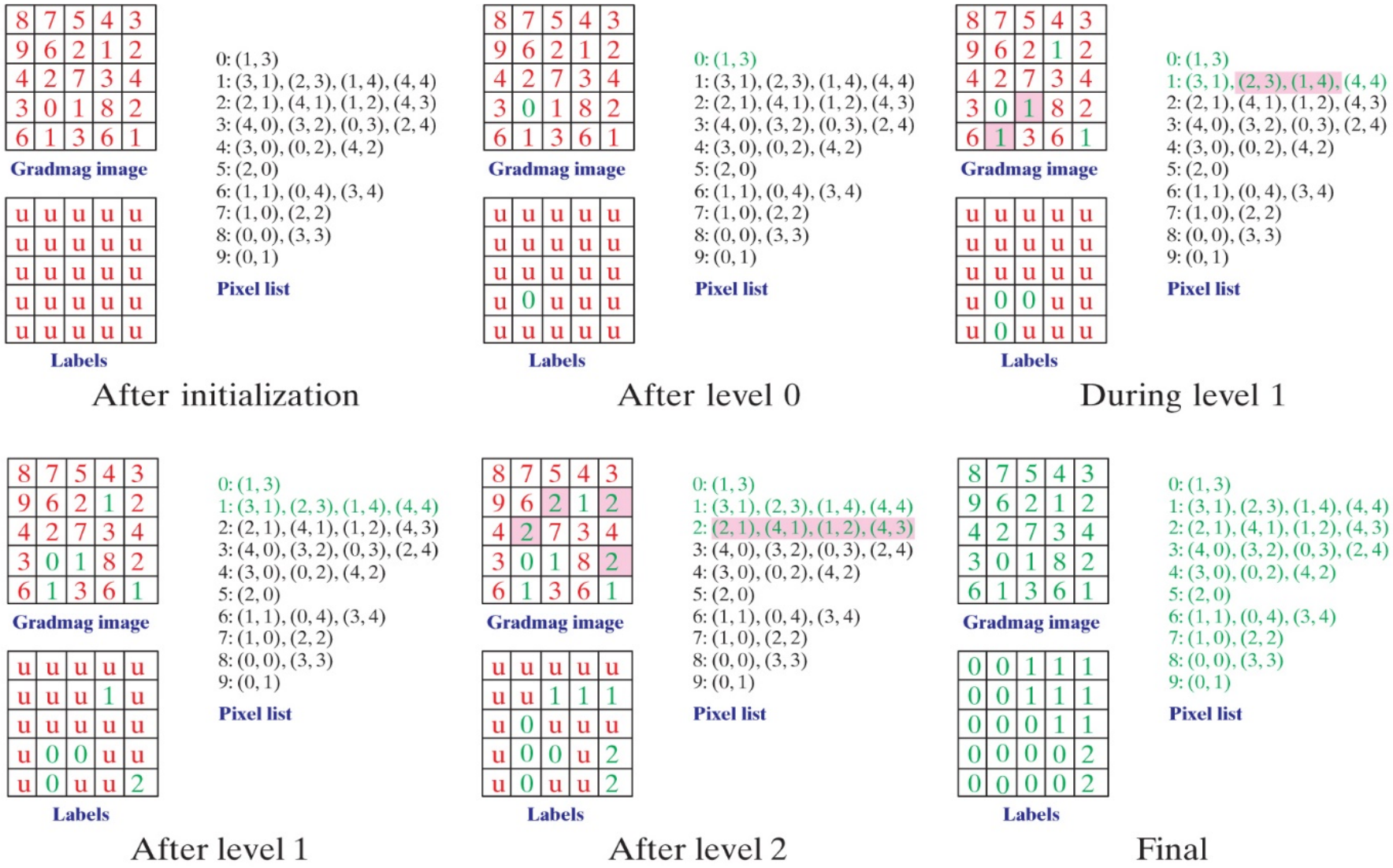
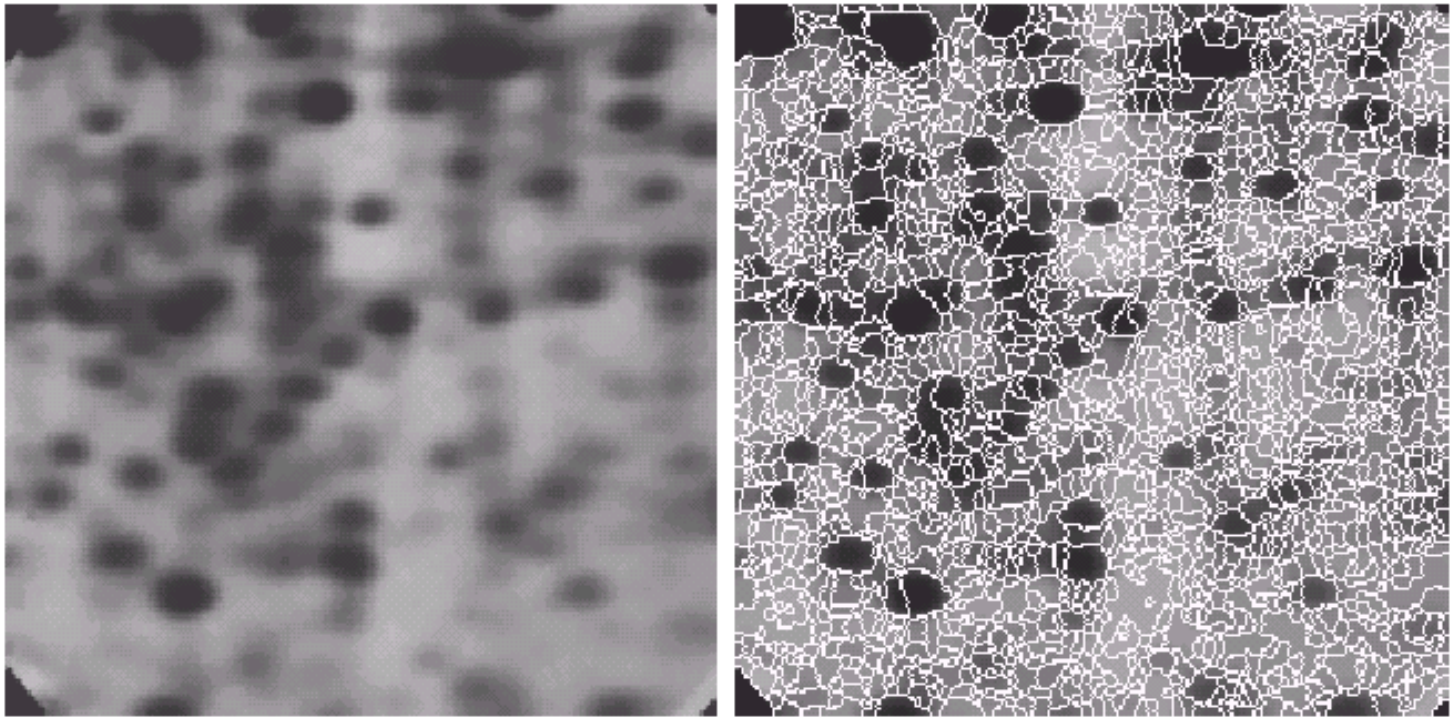


Figure 10.29 An example of non-marker-based immersion watershed on a simple 5 × 5 image with 10 levels (0 through 9). Shown are the results after several steps of the algorithm, followed by the final result. (Note that ties are broken arbitrarily, so other solutions are possible.) At each step, pixel coordinates in the pixel list with a value less than or equal to the current value are colored, and pixels on the frontier are shaded.

Watershed Method (cont.)

- Highly susceptible to noise in the segmentation function
- Any tiny wrinkle in the function will cause a local minimum, which will generate a unique label for a possibly small catchment basin containing the local minimum
- This leads to over-segmentation, and the errors from the standard algorithms are severe.
- The most common approach is to use marker-based watershed segmentation.
- First use an independent procedure to generate a binary image where ON pixels indicates locations (**markers**) where the local minimum is significant.

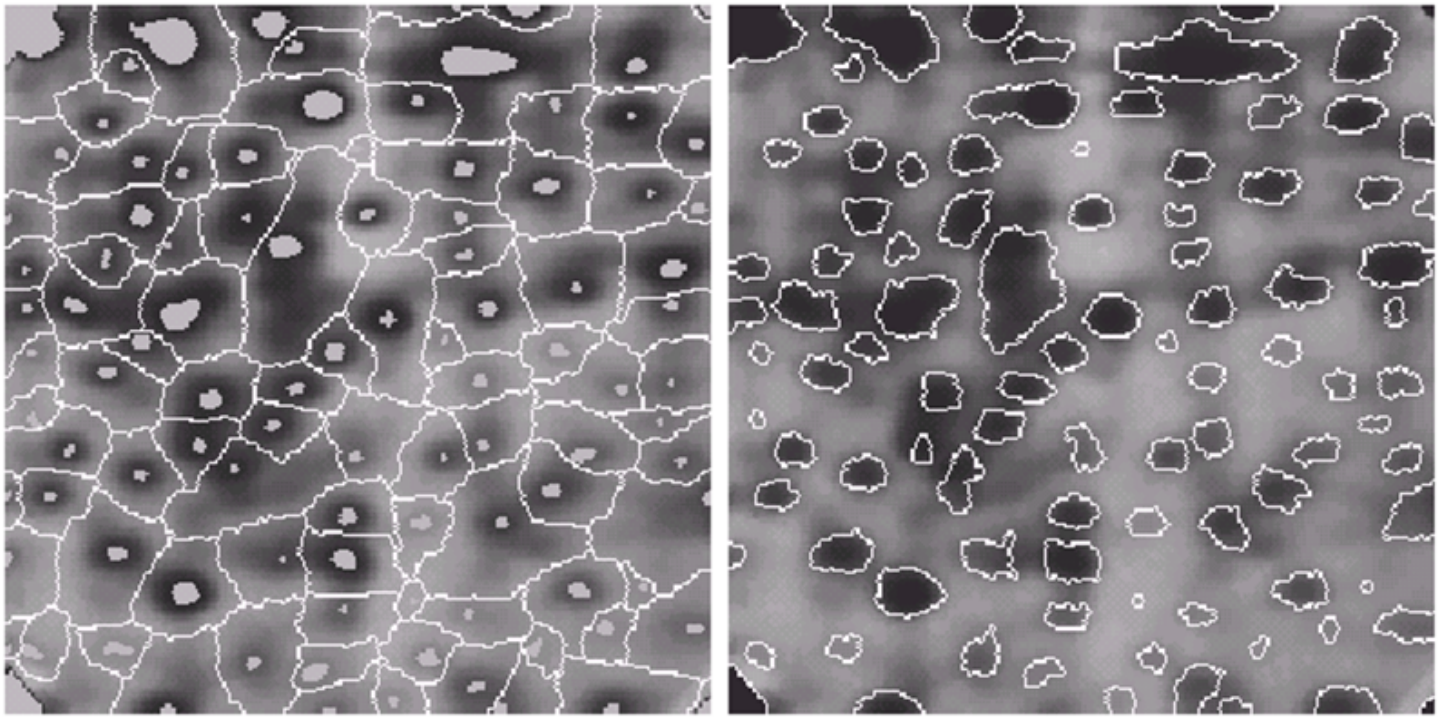
Watershed leads to over-segmentation



a b

FIGURE 10.47
(a) Electrophoresis image. (b) Result of applying the watershed segmentation algorithm to the gradient image. Oversegmentation is evident. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

Markers solve this problem



a b

FIGURE 10.48
(a) Image showing internal markers (light gray regions) and external markers (watershed lines). (b) Result of segmentation. Note the improvement over Fig. 10.47(b). (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

Marker-based watershed Algorithm

- Threshold image
- Compute chamfer distance
- Run watershed to get lines between objects
- Set these lines (skeletons) and blobs to zero in the gradient magnitude image
  Only allow new basins where the value is zero

Watershed Method (cont'd)

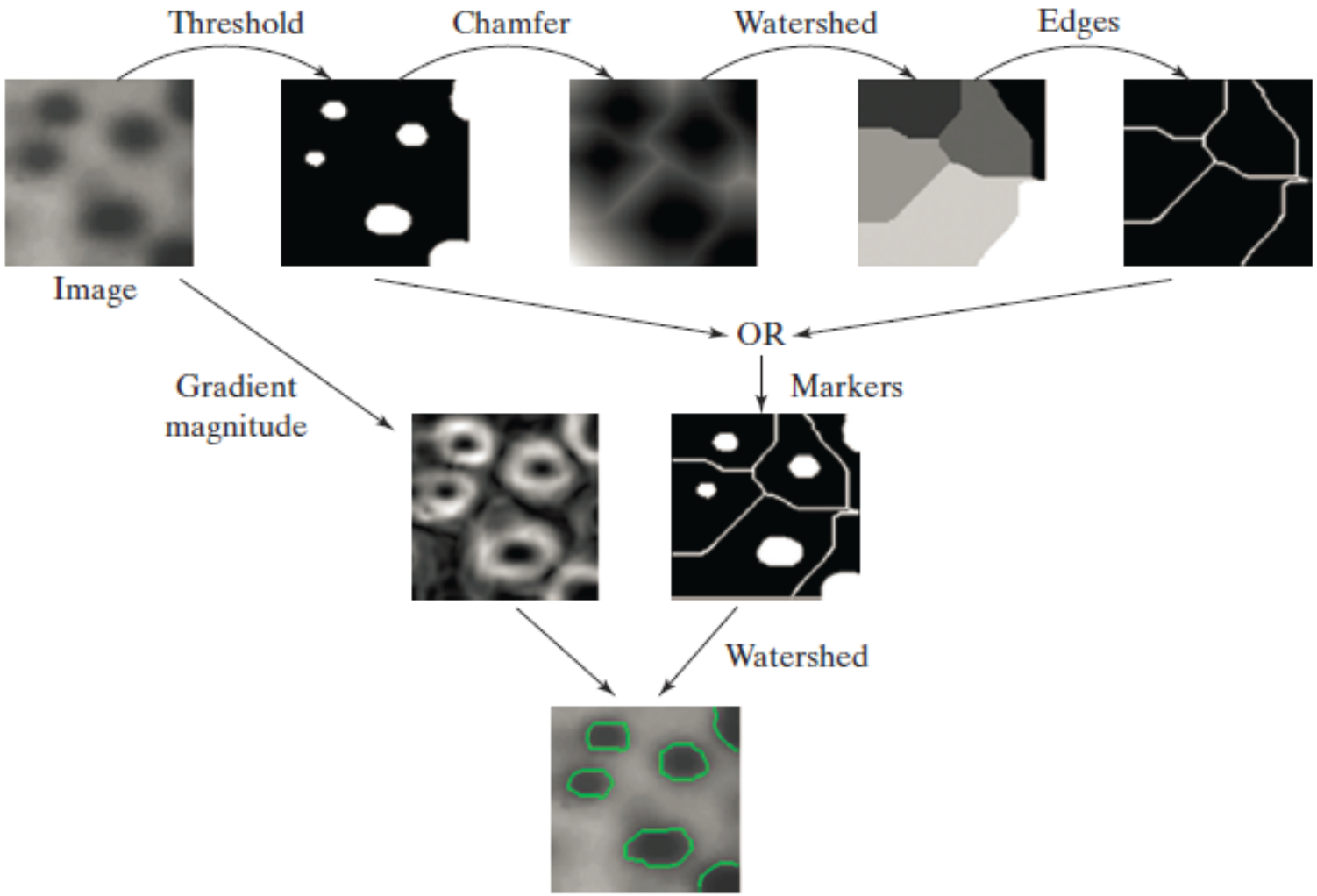
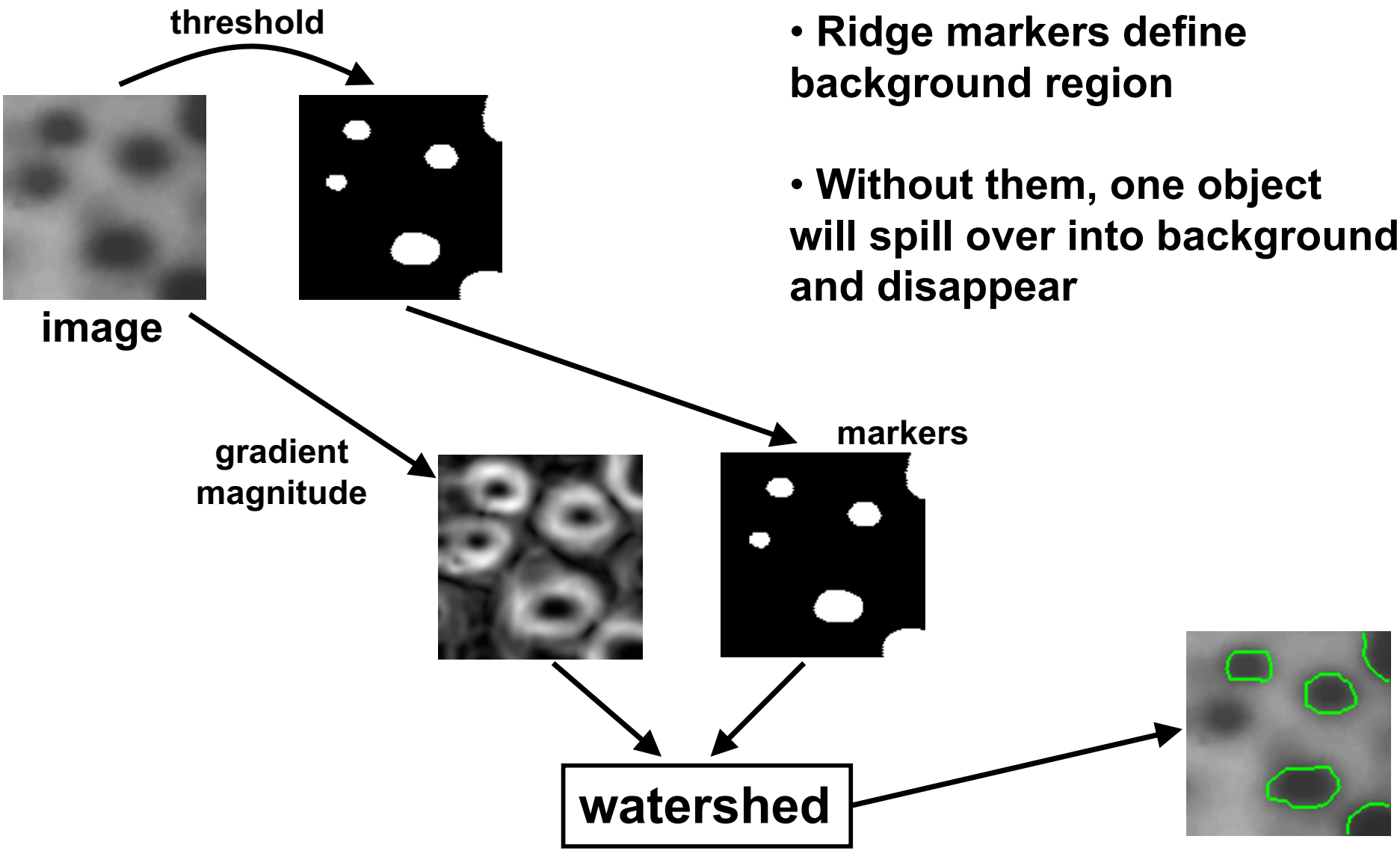


Figure 10.30 Flowchart of the end-to-end watershed procedure.

Why are ridges needed?



- Ridge markers define background region
- Without them, one object will spill over into background and disappear

Graph-Based Segmentation

Issues with Region Growing

- **While effective, it leaves several important questons unanswered.**
  - What merge criterion should be used?
  - How should we select the starting pixels?
  - Among the several pixels adjacent to the region, which one should be considered next?

- **The merge criterion, which defines what we mean when we say pixels look similar, involves three aspects:**
  - The value used (intensity, color, texture, etc.)
  - The distance metric (Euclidean, Manhattan, etc.)
  - The Clustering type (Single Link vs. Complete Link)

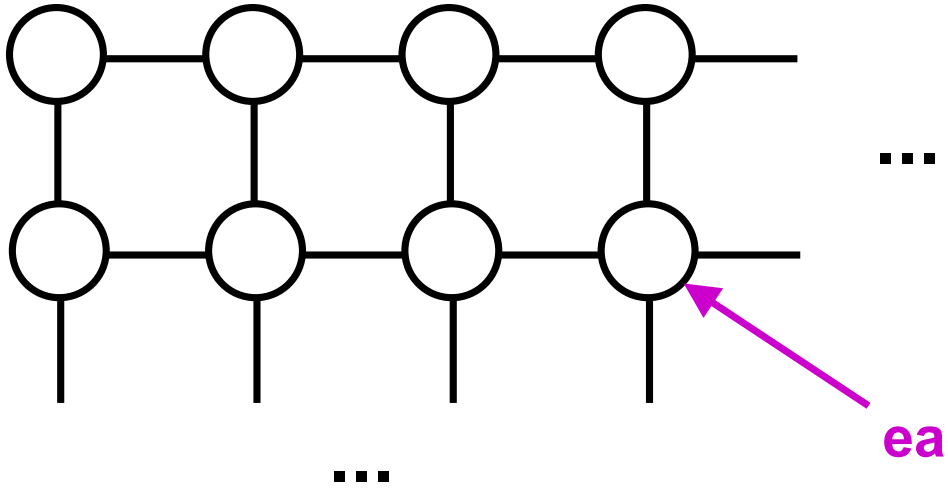
- **Decisions are application specific**

Minimum spanning tree

- Agglomerative clustering can be implemented by building graph using pixels as nodes
- Repeated merging becomes finding a minimum spanning tree in a graph

from Pedro F. Felzenszwalb and Daniel P. Huttenlocher, Efficient Graph-Based Image Segmentation, IJCV, 59(2), 2004
http://people.cs.uchicago.edu/~pff/segment/

Image as a graph



Graph G=(V,E)

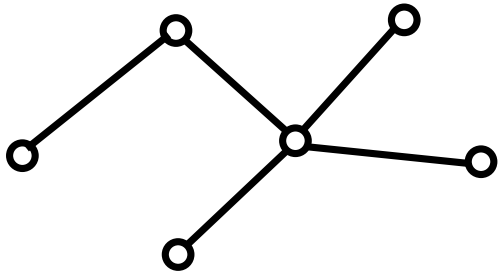
vertices

edges

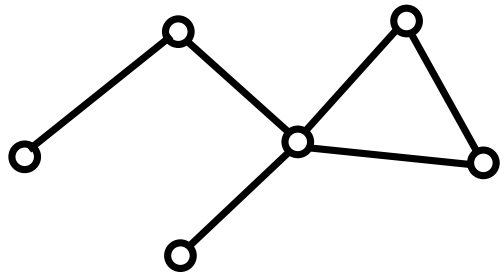
- vertex *v* is just a number *v ≥ 0*
- edge *e=(u,v)* is a pair of vertices
- If undirected graph, then *(u,v)* ⬅➡ *(v,u)*
- If weighted graph, then *w(e)* is weight of edge

Minimum spanning tree

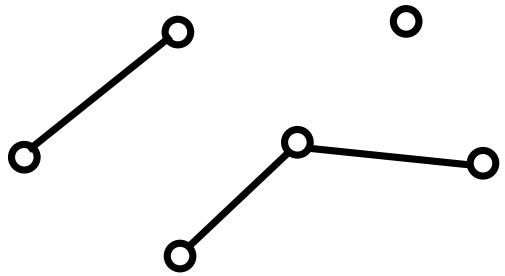
path is sequence of vertices: $v_0, v_1, v_2, \ldots, v_k$
such that (v_i, v_{i+1}) is edge for all i
graph is connected if there exists a path b/w each pair of vertices
graph is tree if connected and acyclic (no cycles)
Examples:



tree



not a tree
(contains a cycle)



not a tree
(not connected)

Given a graph G=(V,E), the minimum spanning tree is a set of edges such that

- resulting graph is a tree
- the sum of all the edge weights is minimal

Kruskal's MST algorithm

1. Initialize each vertex as separate set (or component or region)
2. Sort edges of E by weight (non-decreasing order)
3. T = ϕ (empty set)
4. for each edge (u,v)
    1. if FindSet(u) ≠ FindSet(v)
        1. T = T U {(u,v)}
        2. Merge(u, v)
5. return T

greedy algorithm yet optimal

Kruskal's Algorithm

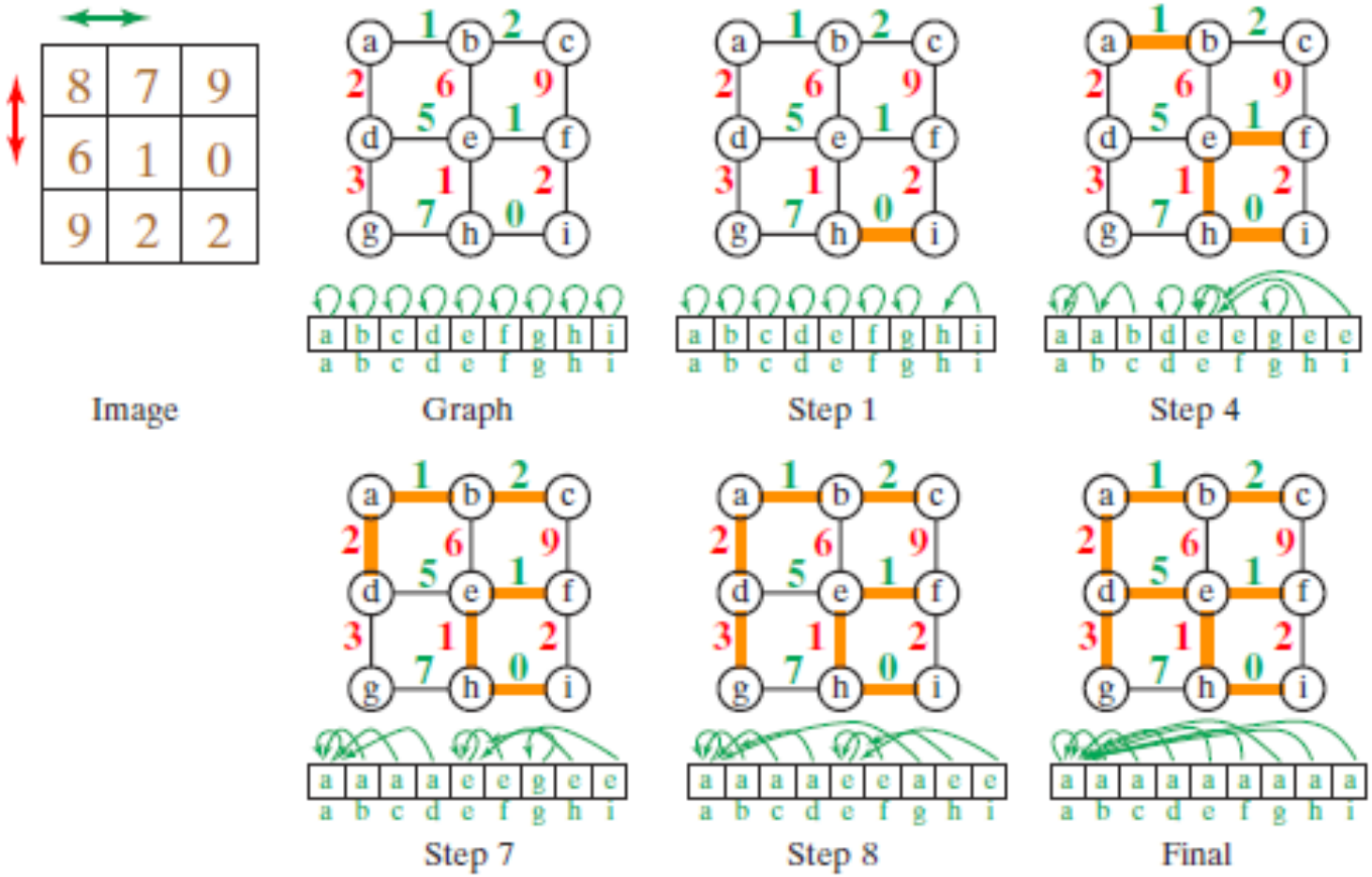
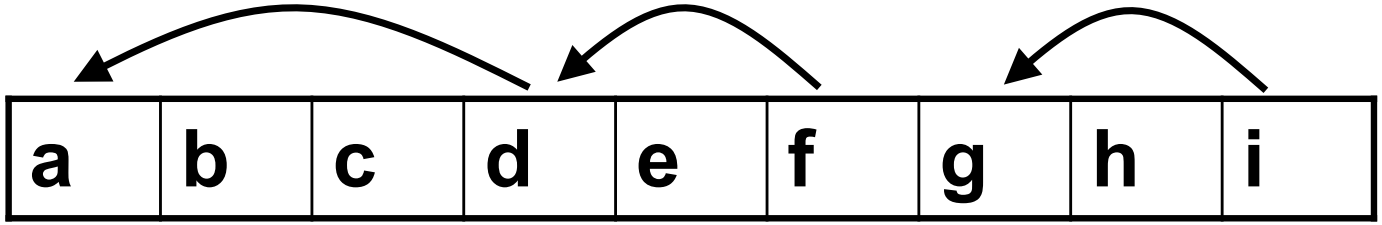


Figure 10.32 Kruskal's algorithm applied to a 3 × 3 image, whose gray levels are shown in brown. Next to the image is shown the graph whose edge weights are the absolute difference in gray level between neighboring pixels, using 4-neighbor connectedness. The remaining graphs show the edges (in orange) added to compute the minimum spanning tree of the graph, as the algorithm proceeds step by step. Note that the penultimate step (step 8) results in a compelling segmentation of the image.

Kruskal implementation

Kruskal's algorithm is implemented similar to connected components that we saw before

disjoint set data structure (equivalence table):



FindSet(u) recursively traces links
Merge(u,v) simply adds link b/w u and v

How does this relate to image segmentation?
This procedure relates to a single image region; it finds the MST of each region in the image.

Graph-Based Methods - Felzenszwalb-Huttenlocher (FH) Algorithm

- The FH algorithm is closely related to computing the **minimum spanning tree (MST)** of a graph, which is the set of edges with minimum weight that connects all the vertices in the graph such that the set contains no cycles.

- Instead of merging automatically whenever two pixels are in different regions, two pixels are merged only if they are in different regions and they look similar.

- As a result, this modified algorithm does not find a single MST but rather a forest of MSTs, one tree per region.

MST image segmentation

- **First, smooth image by convolving with Gaussian**
  Small variance (e.g., σ^2=0.5) is fine

- **Build graph from image:**
  - vertices are pixels
  - edges connected adjacent vertices (e.g., 4-adjacency)
  - edge weights are absolute intensity differences:
    $w(u,v) = | I(u) – I(v) |$

- **Run Kruskal's algorithm on the graph, but only merge two regions if certain criteria are met**

- **While running the algorithm,**
  - we have a *forest* (set of trees)
  - properties are maintained for each tree
  - trees are merged based on criteria
  - (but we don't care about the trees themselves, so we only need to store the regions, i.e., the set of pixels not edges)

- **When the algorithm finishes,**
  the individual trees are the image regions

FH Algorithm (cont.)

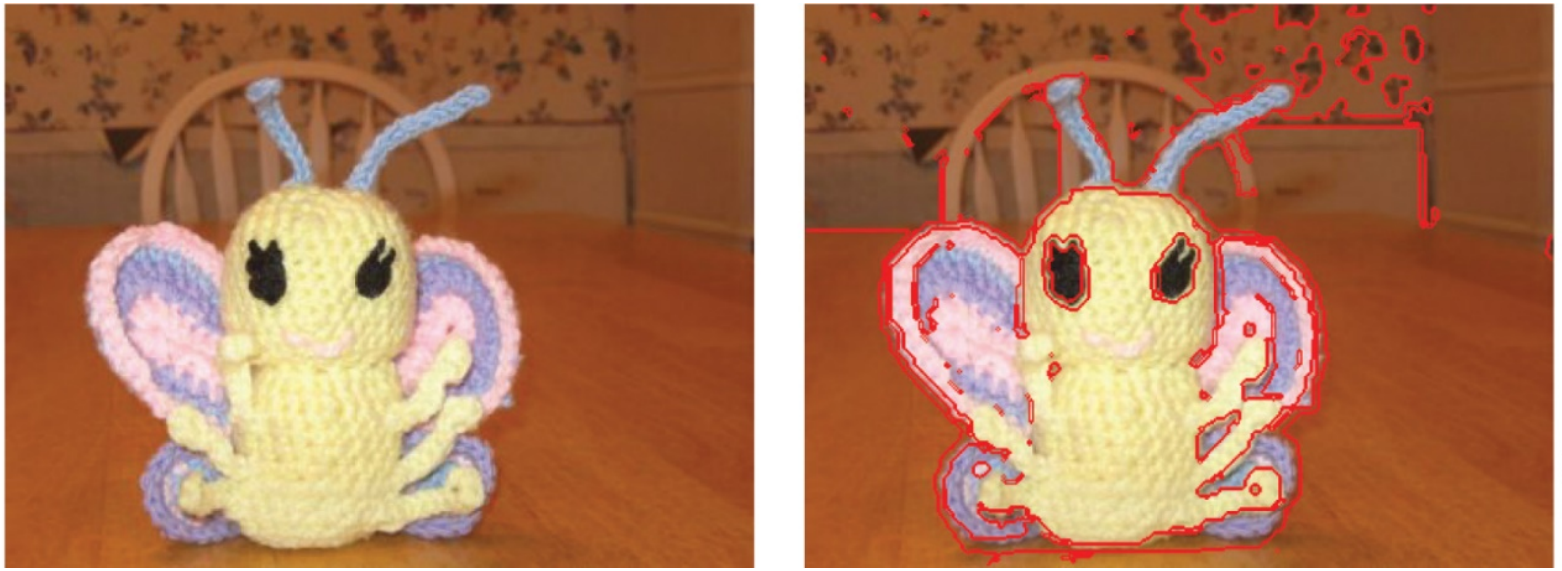


Figure 10.33 Result of the Felzenszwalb-Huttenlocher segmentation algorithm on an image of a knitted butterfly, shown as boundaries (red).

Jessica Birchfield

FH Algorithm (cont.)

- The FH algorithm can be seen as elegant version of region growing that overcomes some of the limitations of the standard approach.

- The FH algorithm is limited to smallest neighbor clustering. In contrast, region growing can use a variety of options but at a cost of increased computation.

- Although a greedy approach, grows separate regions simultaneously in a fair and balanced way.

Recap

- Thresholding
- Gestalt Psychology
- Image Segmentation
- Graph-Based Methods

Questions?

Slide Credits

Some slides from Dr. Stanley Birchfield ECE-847