ECE5554 – Computer Vision
Lecture 7b– Other Segmentation Methods

Creed Jones, PhD

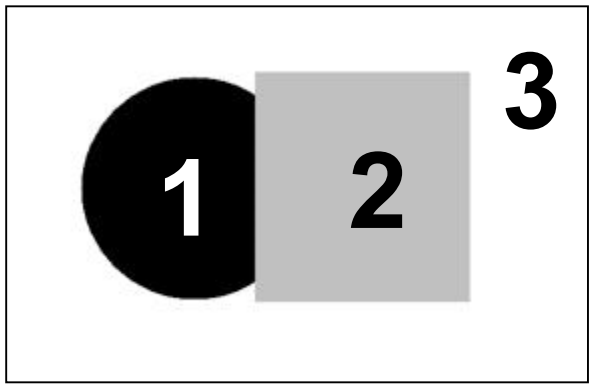
Today's Objectives

- Histogram clustering
  - K-means clustering

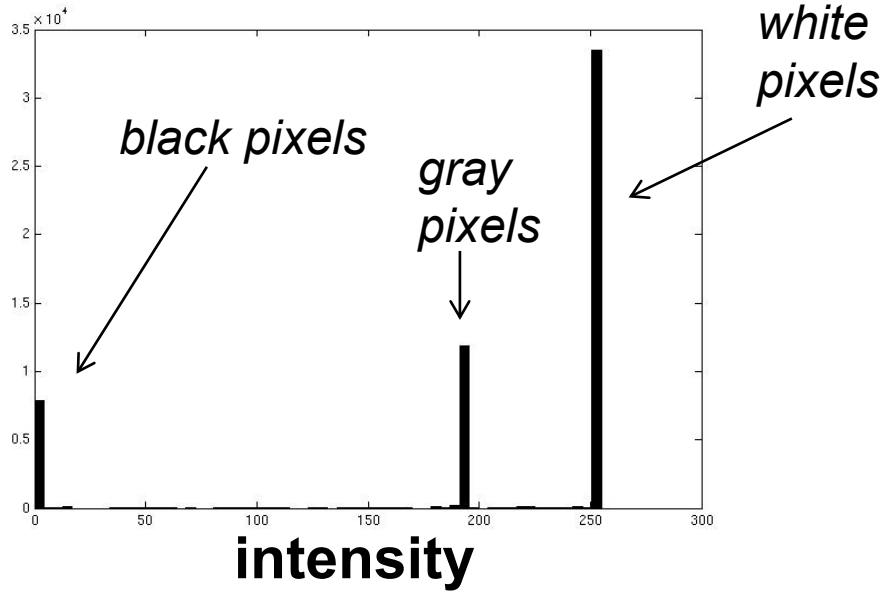
Labeling Regions by Flood Filling

- The recursive flood fill algorithm
- Watershed segmentation

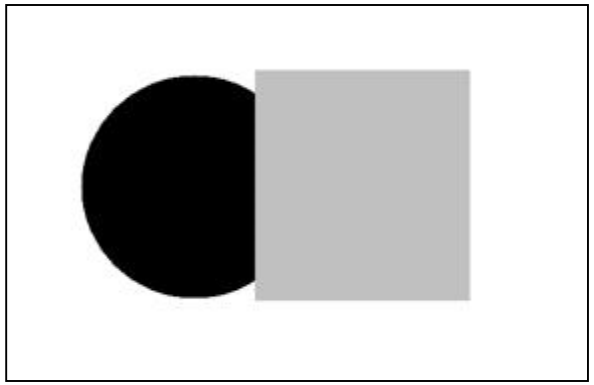
Reconsider the case of a multimodal histogram – can we do more thorough analysis of the histogram itself?



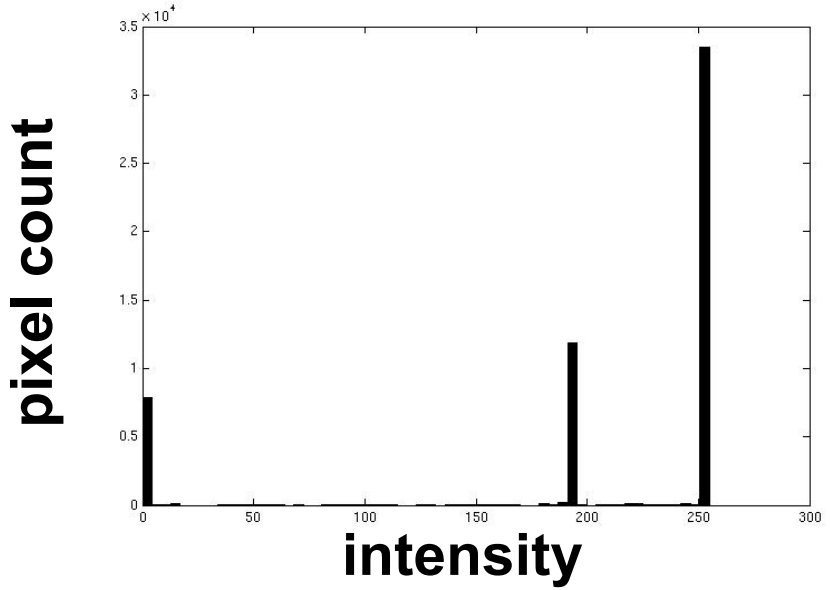
input image

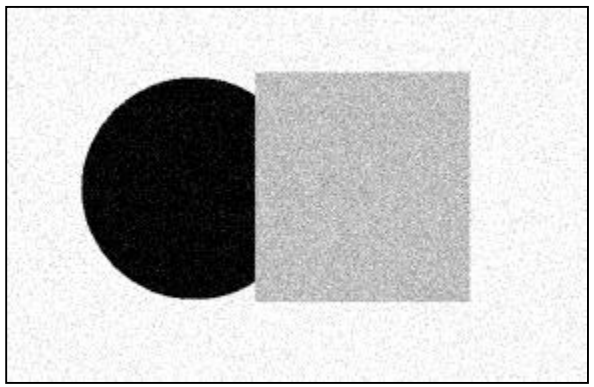


- These intensities define the three groups
- We could label every pixel in the image according to which of these primary intensities it is
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

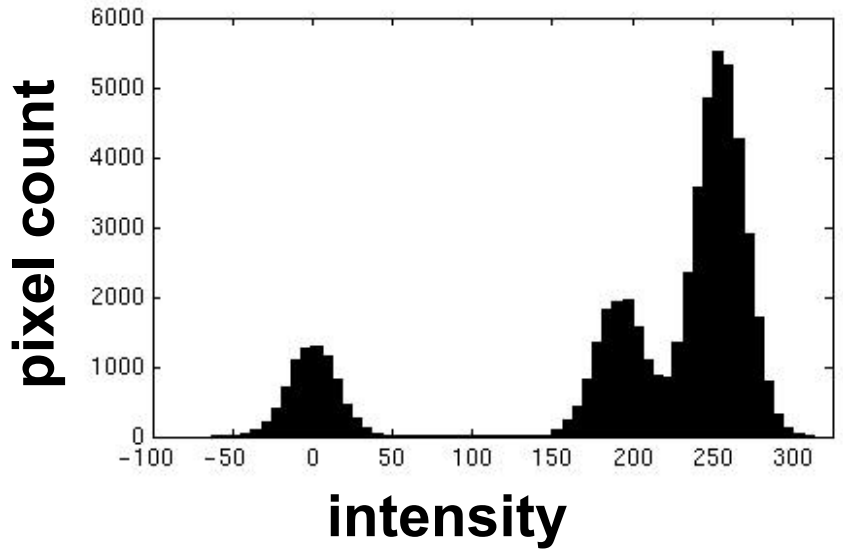


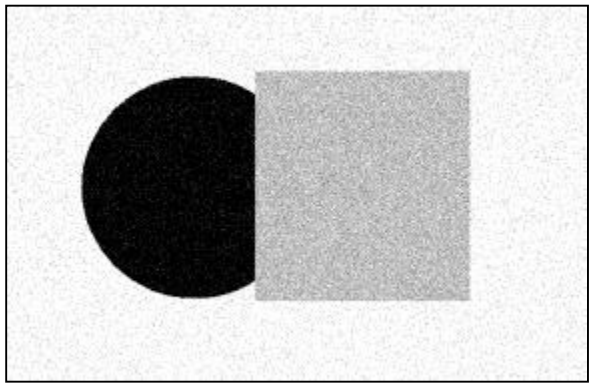
input image



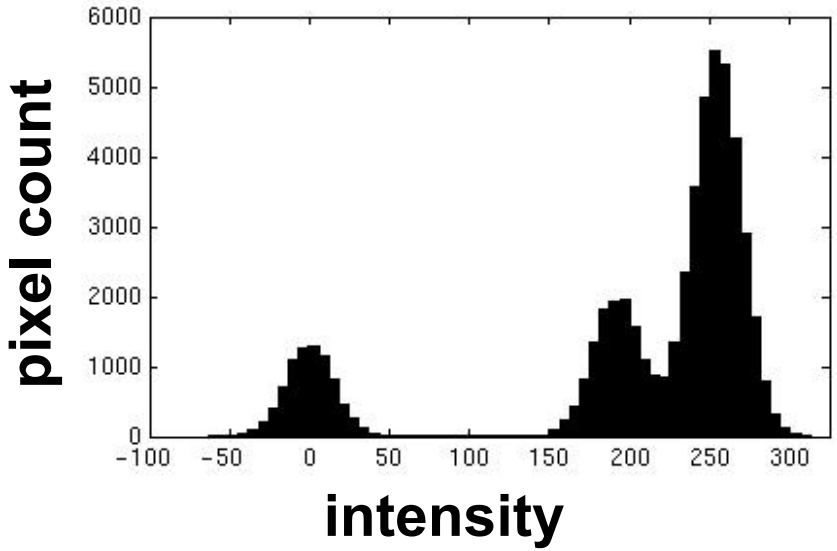


input image

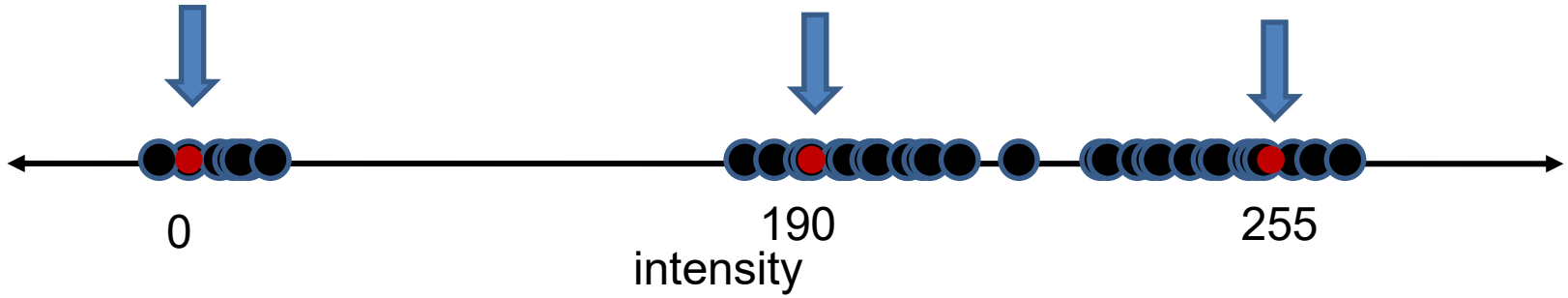


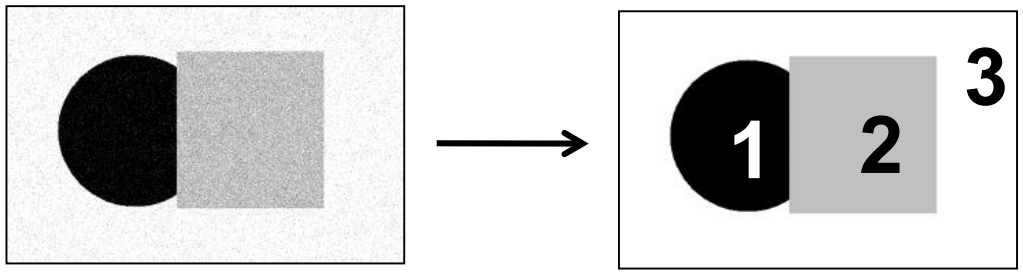


input image



- Now, how to determine the three main intensities that define our groups?
- We could perform ***clustering***



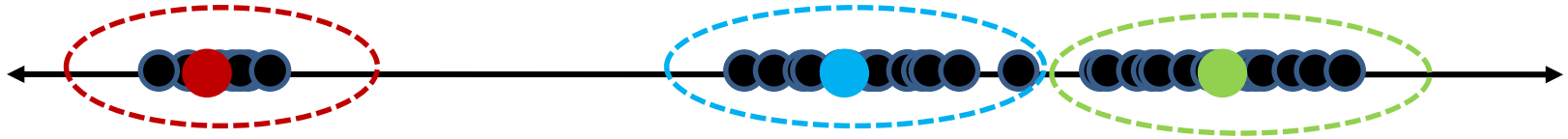


- Goal: choose three "centers" as the representative intensities, and label every pixel based on which center is nearest in intensity (not nec. nearest in image space)
- Best cluster centers? Possibly those that minimize SSD between all points and their nearest cluster center c_i:

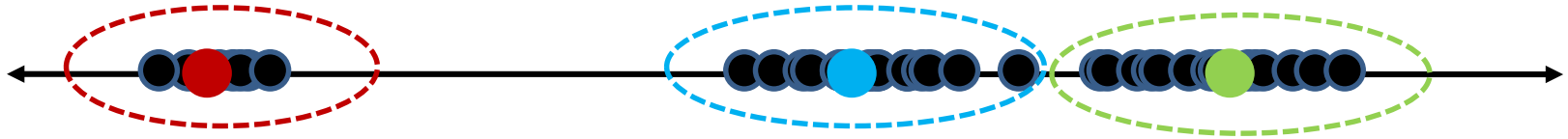
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Histogram analysis, considered as a clustering problem

- With this objective, it is a "chicken and egg" problem:
  - If we knew the cluster centers, we could allocate points to groups by assigning each to its closest center.



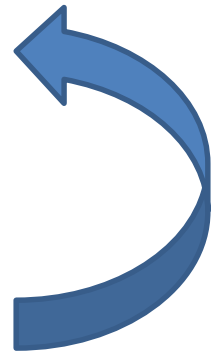
  - If we knew the group memberships, we could get the centers by computing the mean per group.



K-means clustering

- Basic idea: randomly initialize the *K* cluster centers, and iterate between the two steps we just saw.

  1. Randomly initialize the cluster centers, c_1, ..., c_K
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat from Step 2



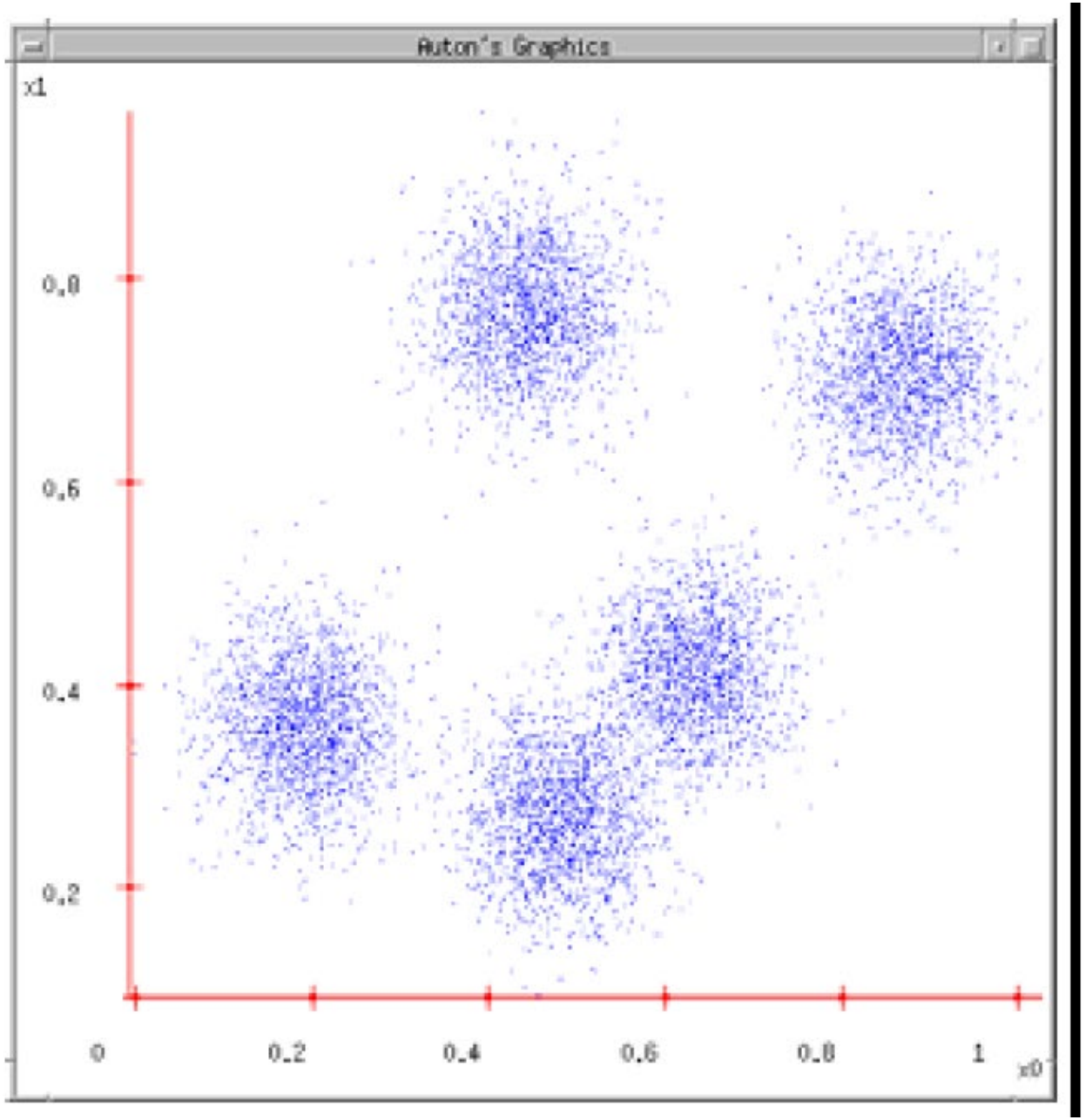
Properties

- Will always converge to *some* solution
- Can be a local minimum: does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

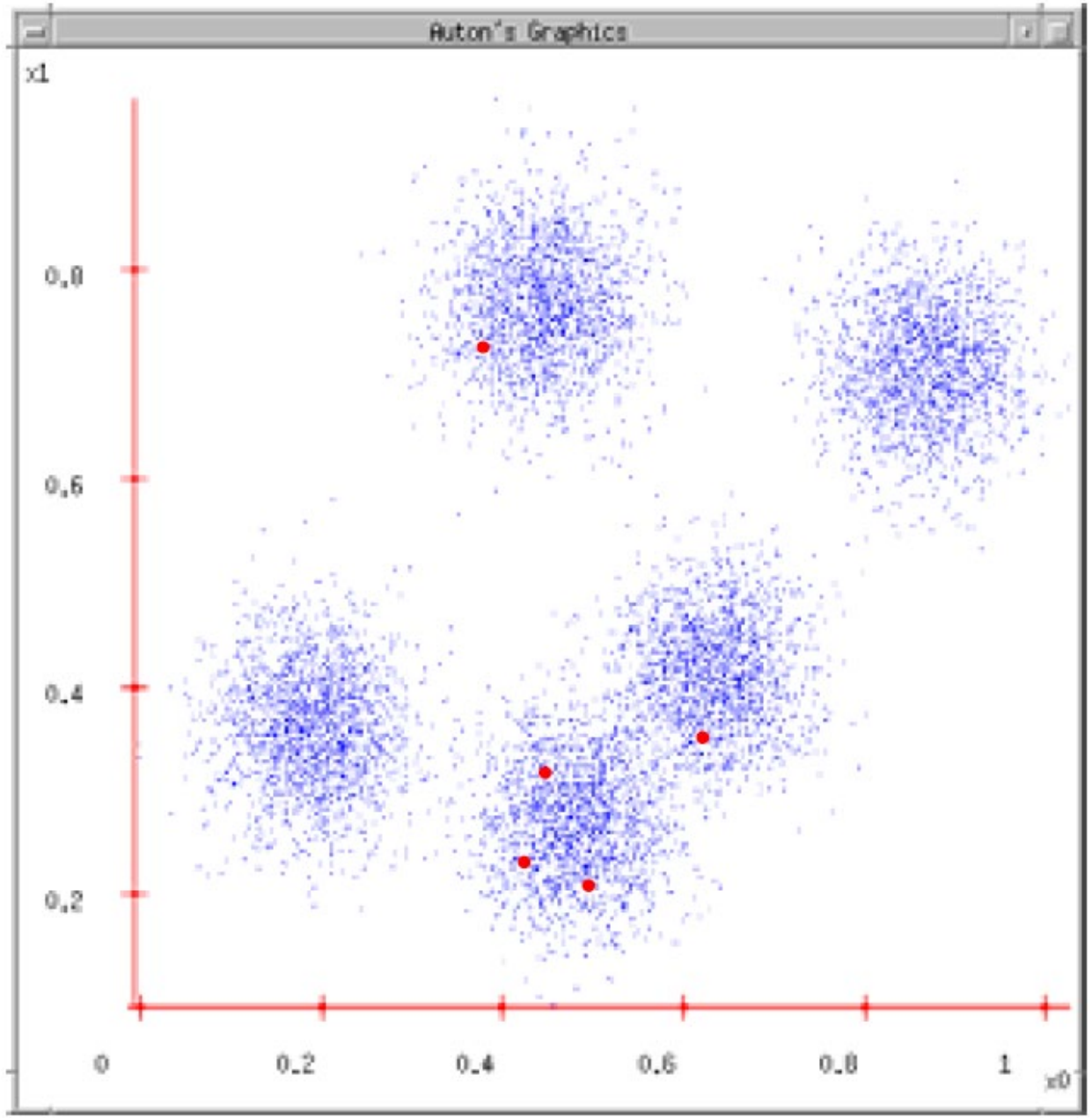
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*



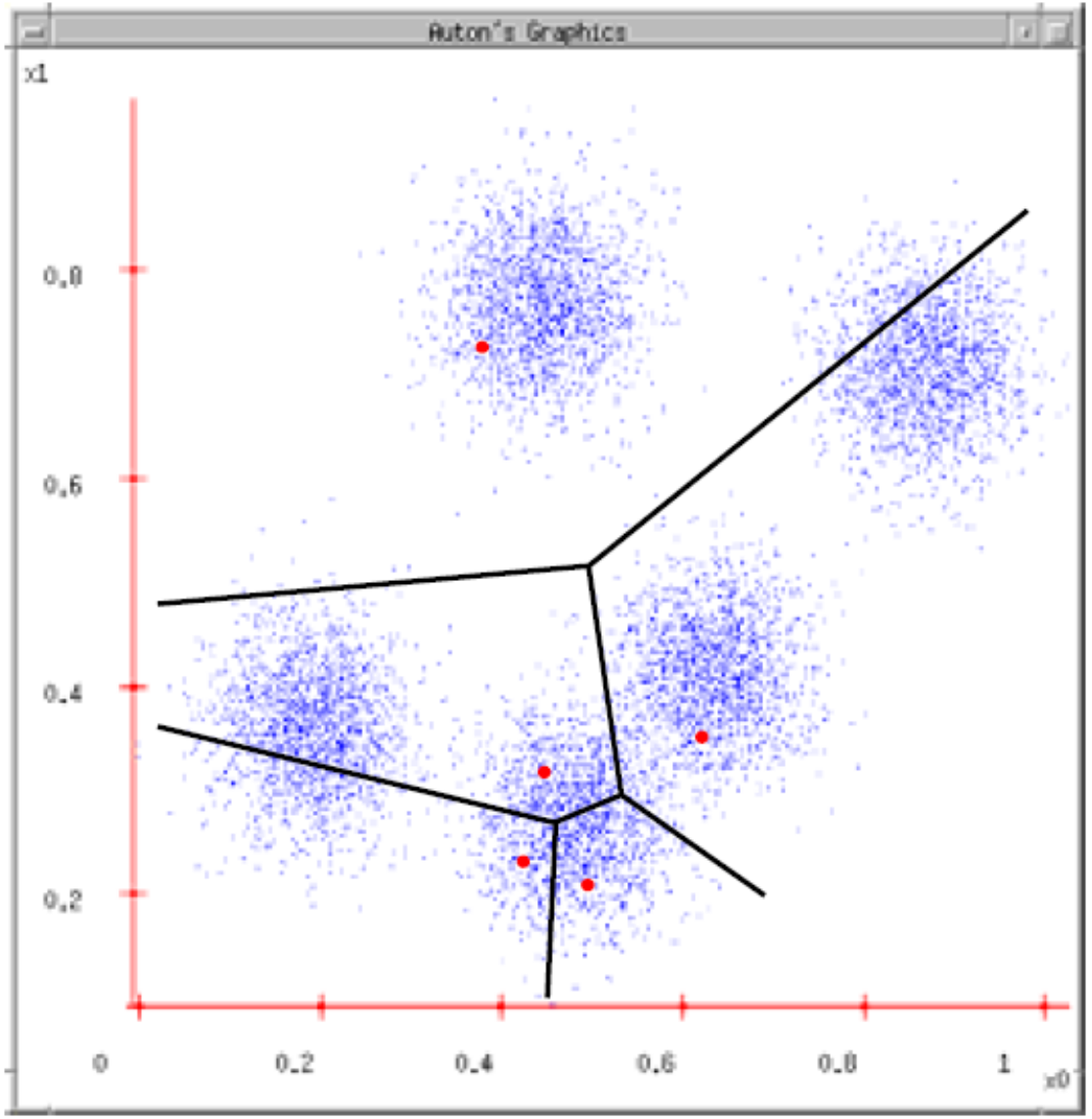
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



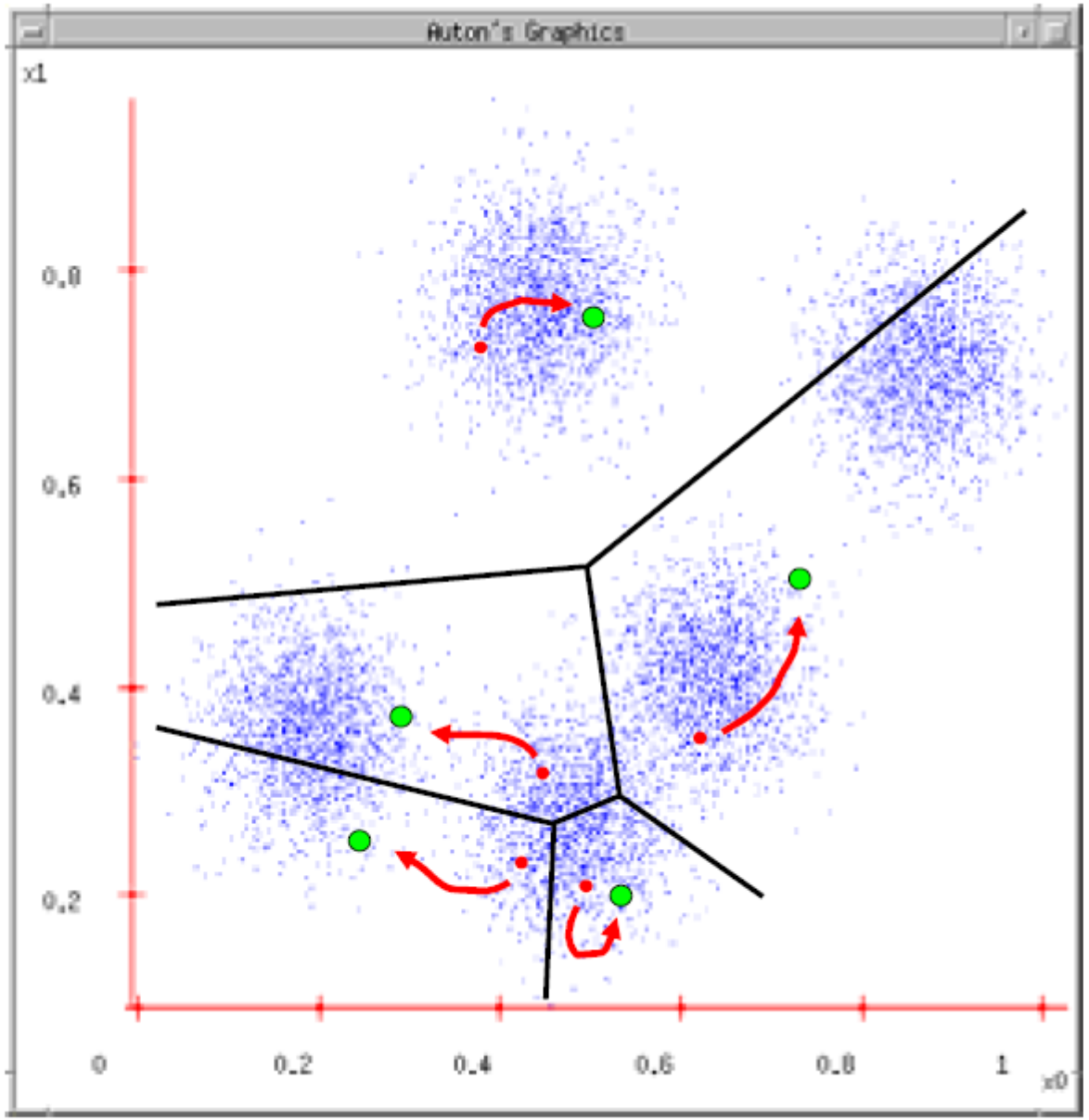
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



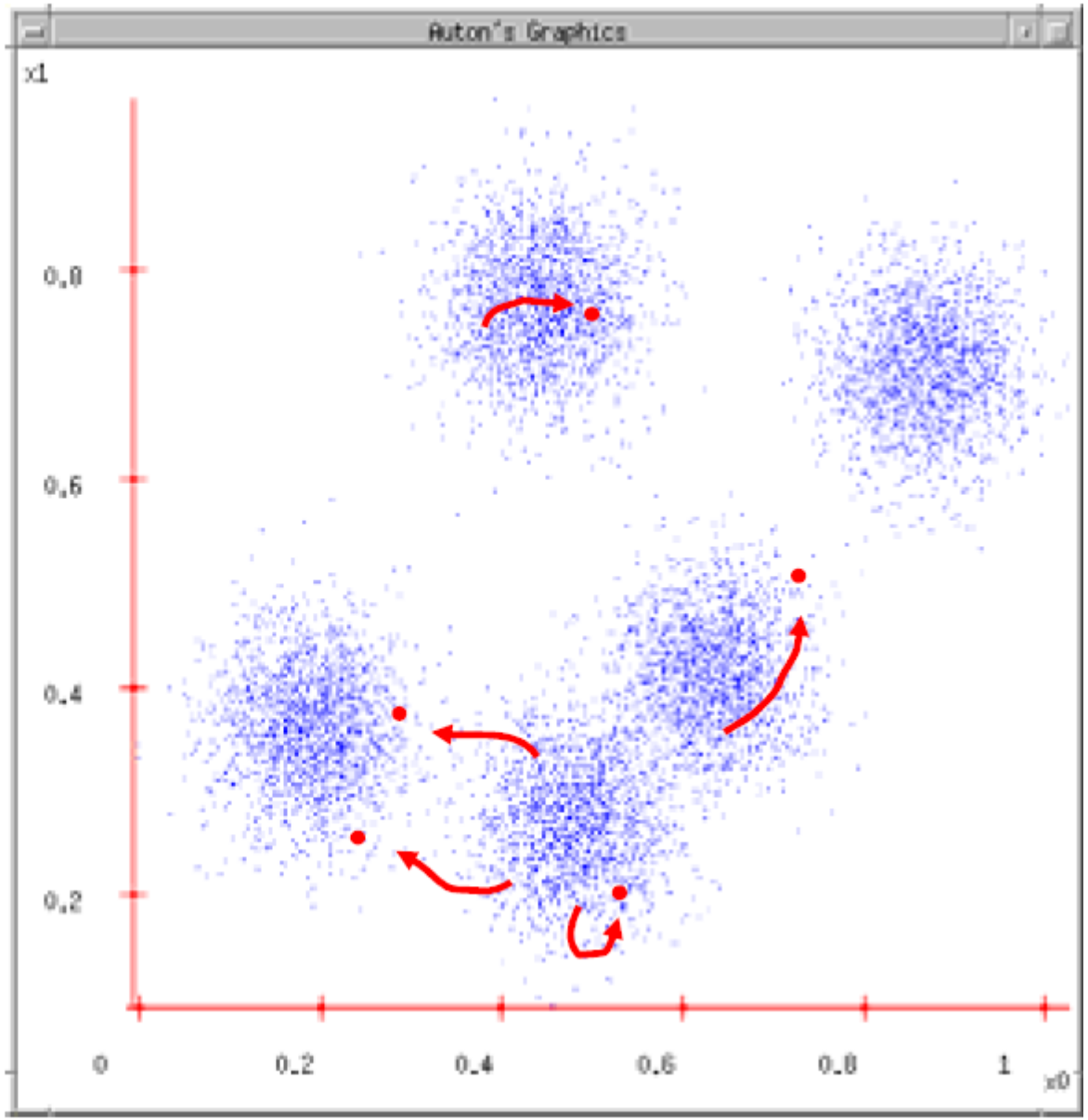
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns

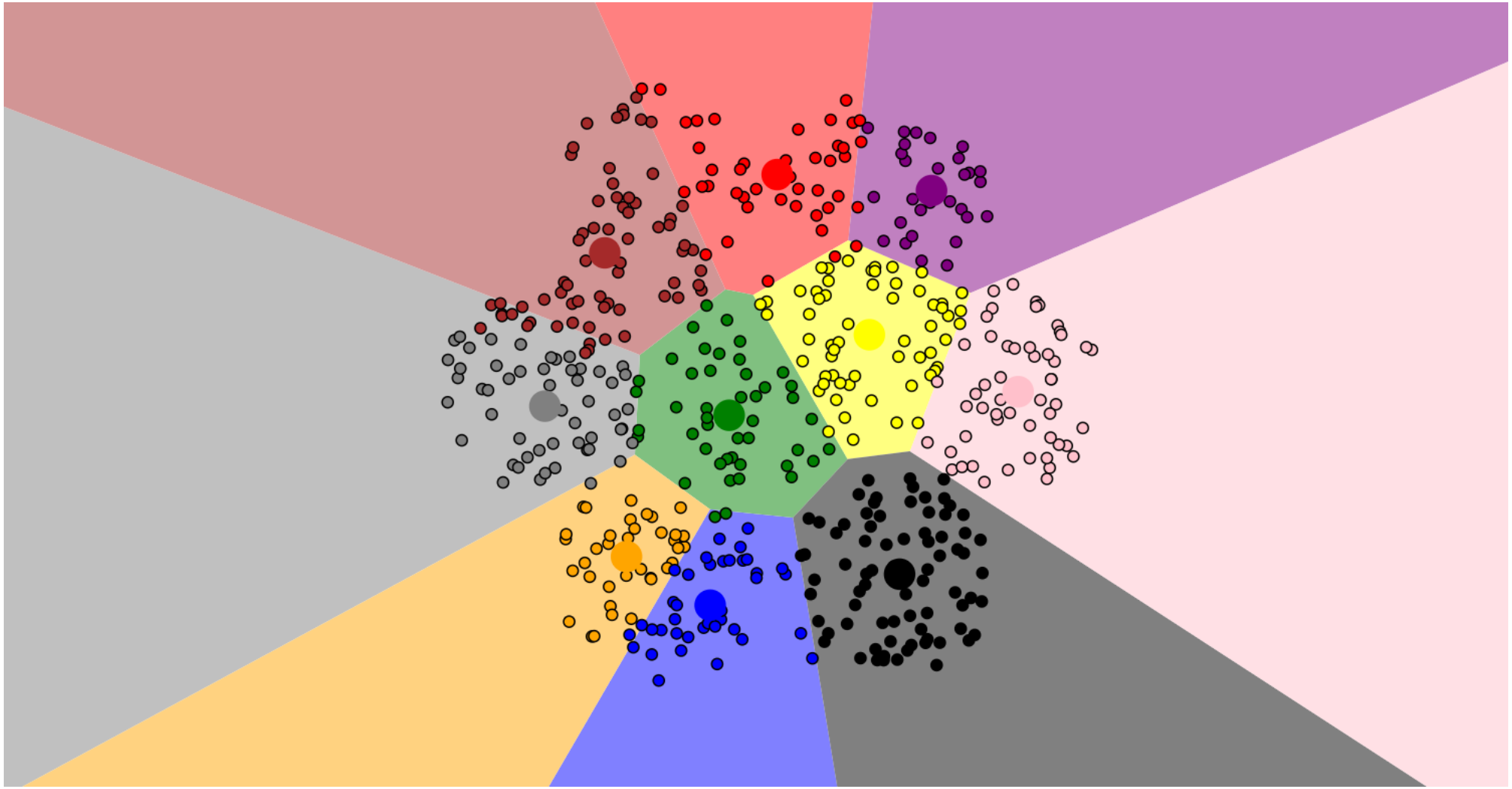


K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



A nice demo of K-means clustering is at https://www.naftaliharris.com/blog/visualizing-k-means-clustering/



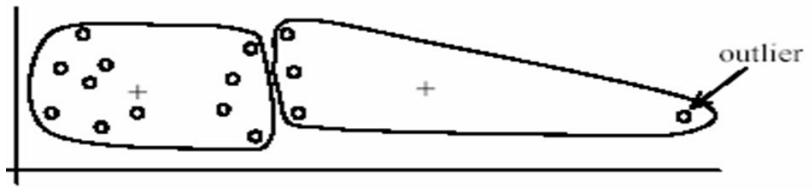
K-means: pros and cons

Pros

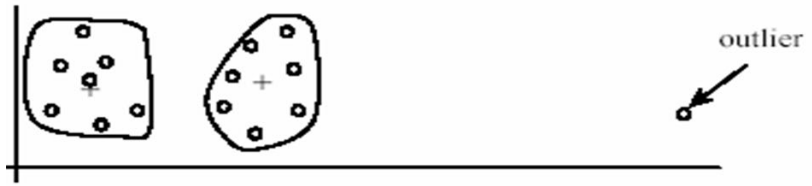
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

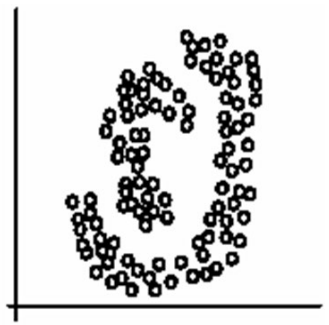
- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



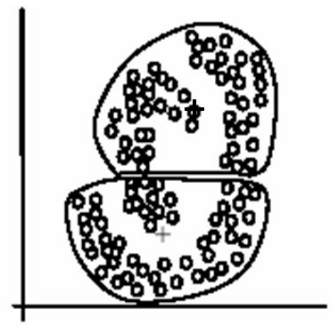
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters



(B): *k*-means clusters

Image segmentation as clustering

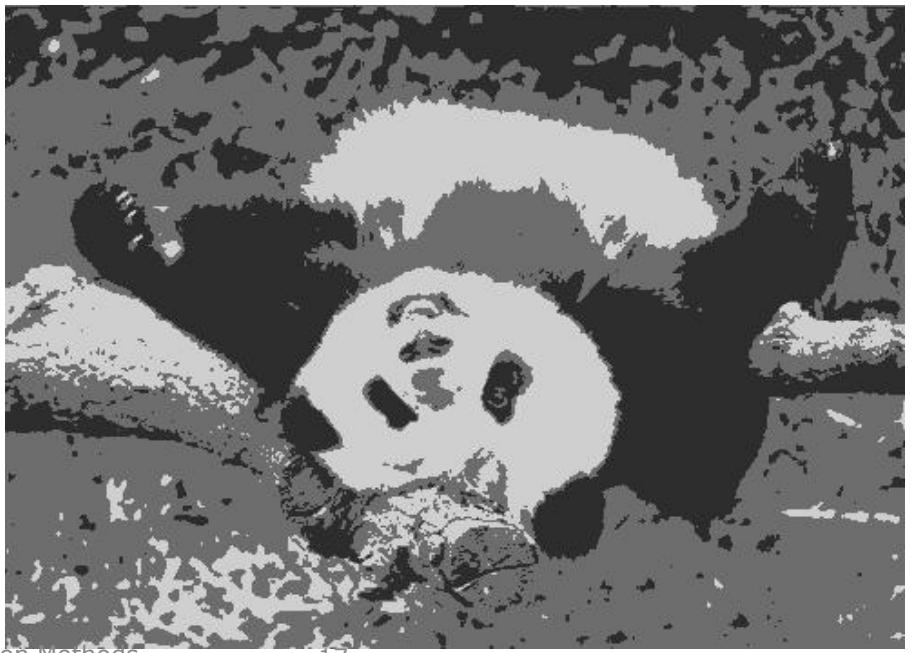
Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-dimensional)

K=2

K=3



Quantization of the feature space; segmentation label map

Segmentation as clustering

- Color, brightness, position alone are not enough to distinguish all regions of interest …



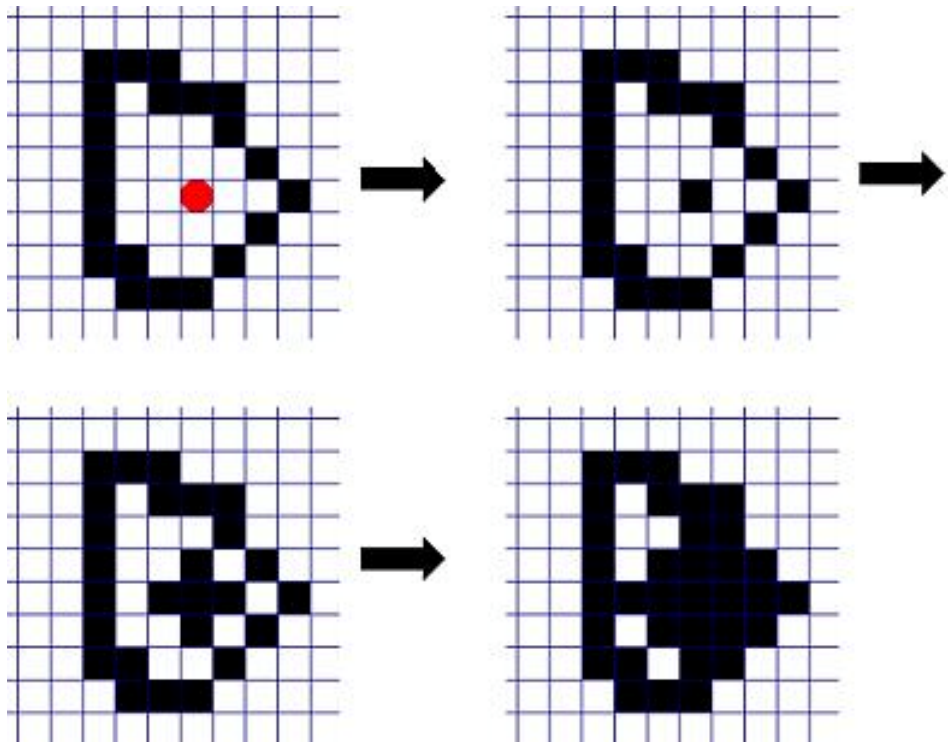
We can also examine pixels individually & accumulate the region statistics using a *flood fill* algorithm – this is a simple recursive example (other, more efficient versions exist)

Flood-fill (node, target-color, replacement-color):

1. If target-color is equal to replacement-color, return.
2. If the color of node is not equal to target-color, return.
3. Set the color of node to replacement-color.
4. Perform Flood-fill (one step to the south of node, target-color, replacement-color).
   Perform Flood-fill (one step to the north of node, target-color, replacement-color).
   Perform Flood-fill (one step to the west of node, target-color, replacement-color).
   Perform Flood-fill (one step to the east of node, target-color, replacement-color).
5. Return.

The flood fill algorithm ensures connectivity of the pixels included, since it works on the basis of the local 8- or 4-neighborhood

- We must start with a single pixel in the interior – or more commonly on the perimeter of – the object
- Repeated steps will agglomerate the connected pixels up to the boundary



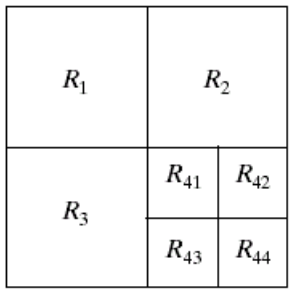
Region-Oriented Segmentation

- Region Growing
  - Region growing is a procedure that groups pixels or subregions into larger regions.
  - The simplest of these approaches is *pixel aggregation*, which starts with a set of "seed" points and from these grows regions by appending to each seed points those neighboring pixels that have similar properties (such as gray level, texture, color, shape).
  - Region growing based techniques are better than the edge-based techniques in noisy images where edges are difficult to detect.

Slide: thanks to Bahadir K. Gunturk, LSU

Region-Oriented Segmentation

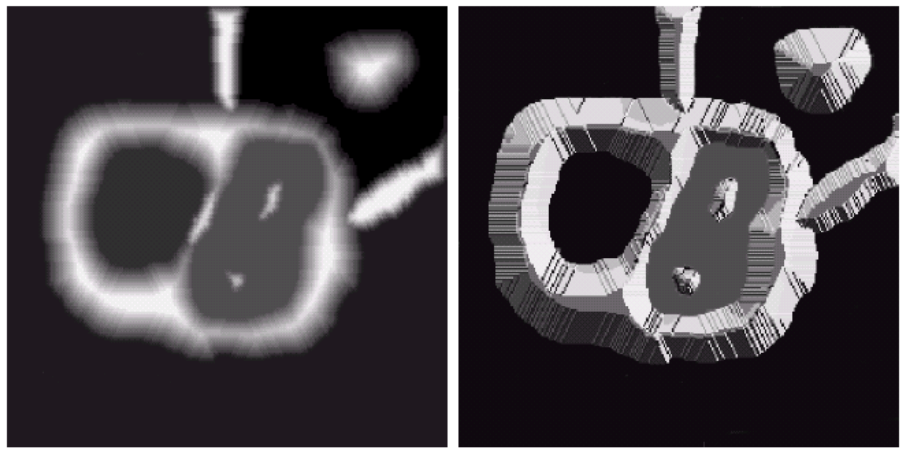
- Region Splitting
  - Region growing starts from a set of seed points.
  - An alternative is to start with the whole image as a single region and subdivide the regions that do not satisfy a condition of homogeneity.
- Region Merging
  - Region merging is the opposite of region splitting.
  - Start with small regions (e.g. 2x2 or 4x4 regions) and merge the regions that have similar characteristics (such as gray level, variance).
  - Typically, splitting and merging approaches are used iteratively.

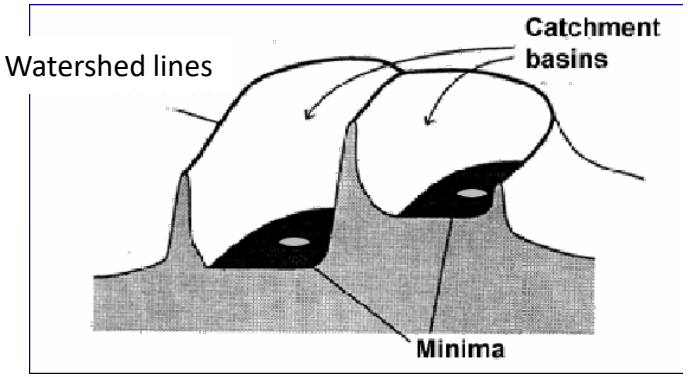


Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm

- Visualize an image in 3D: spatial coordinates and gray levels.
- In such a topographic interpretation, there are 3 types of points:
  - Points belonging to a regional minimum
  - Points at which a drop of water would fall to a single minimum. (→The *catchment basin* or *watershed* of that minimum.)
  - Points at which a drop of water would be equally likely to fall to more than one minimum. (→The *divide lines* or *watershed lines*.)

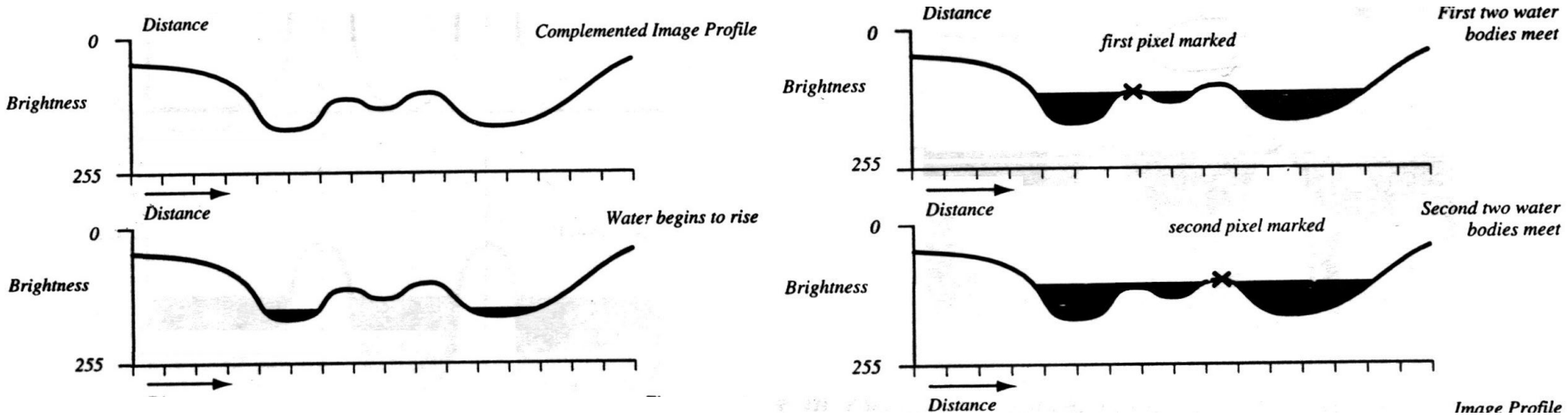




Slide: thanks to Bahadir K. Gunturk, LSU

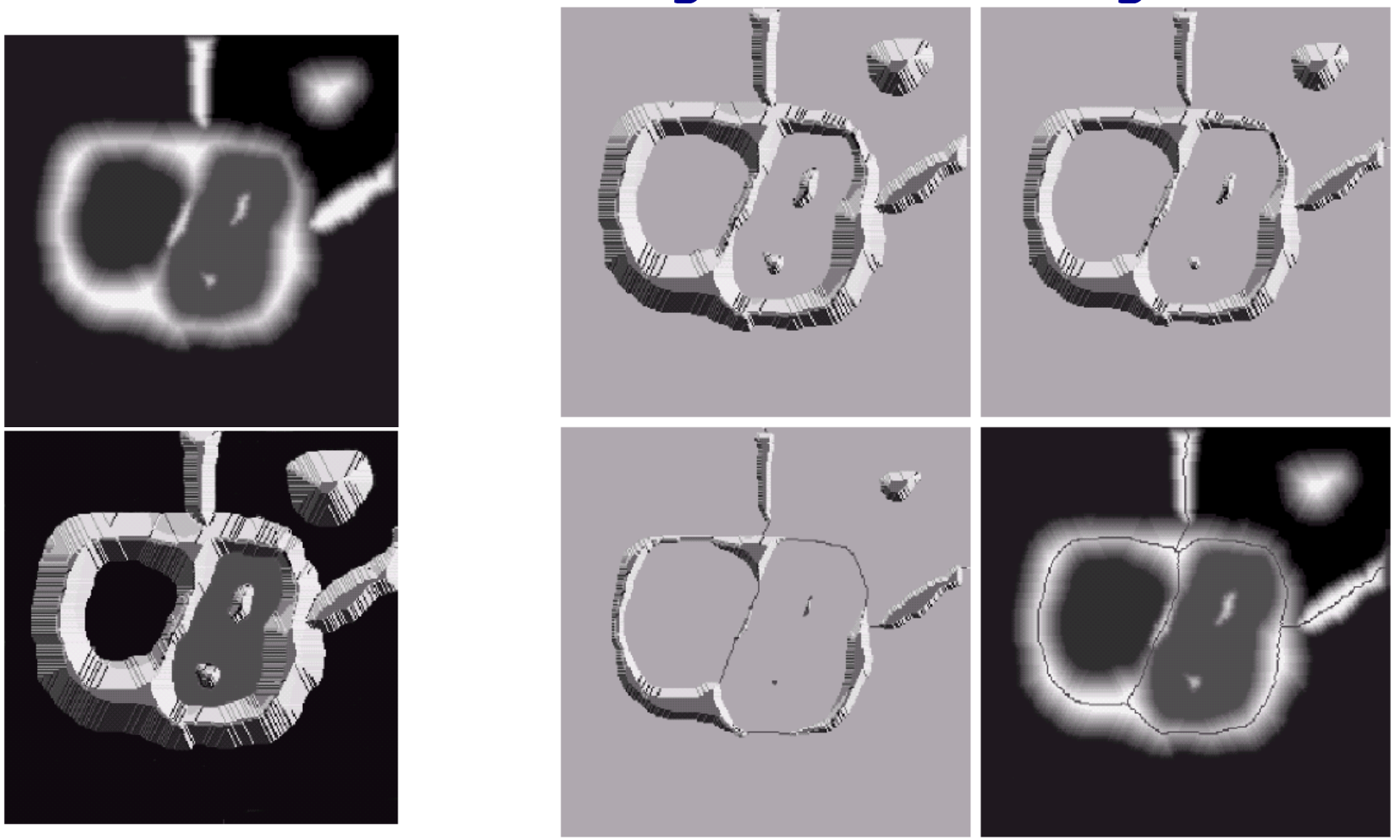
Watershed Segmentation Algorithm

- The objective is to find watershed lines.
- The idea is simple:
  - Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate.
  - When rising water in distinct catchment basins is about the merge, a dam is built to prevent merging. These dam boundaries correspond to the watershed lines.



Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm



e f
g h

FIGURE 10.44
(Continued)
(e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm

- Start with all pixels with the lowest possible value.
  - These form the basis for initial watersheds
- For each intensity level k:
  - For each group of pixels of intensity k
    - If adjacent to exactly one existing region, add these pixels to that region
    - Else if adjacent to more than one existing regions, mark as boundary
    - Else start a new region

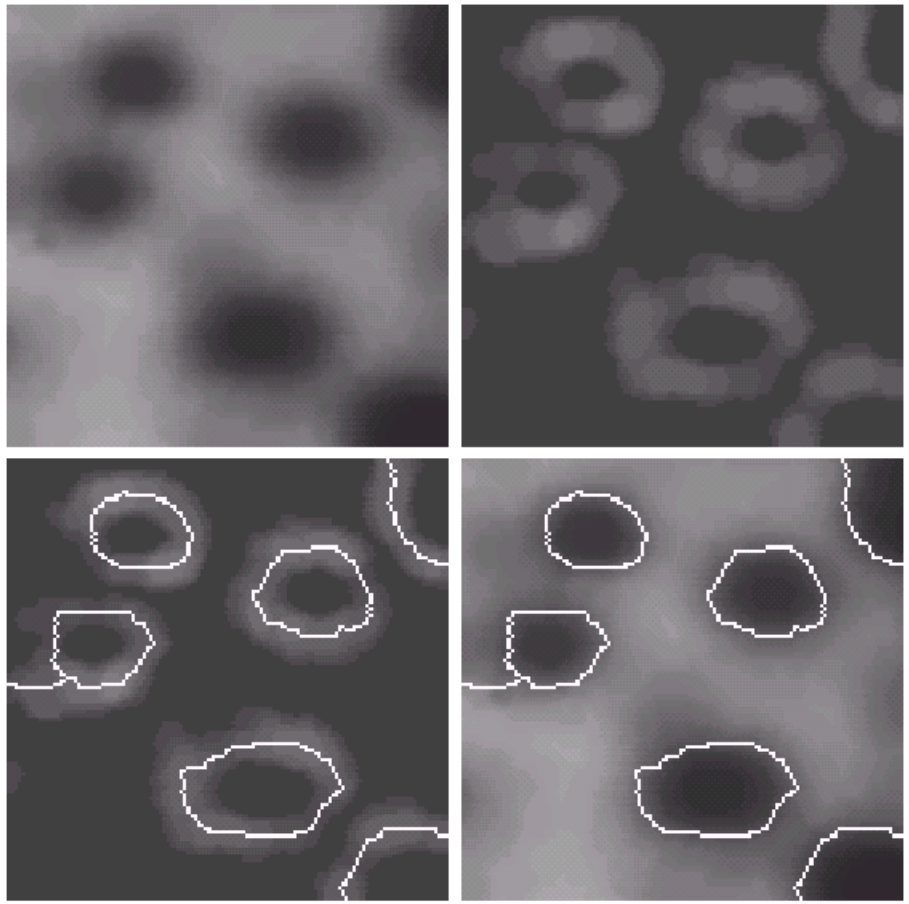
Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm

Watershed algorithm might be used on the gradient image instead of the original image.

a b
c d

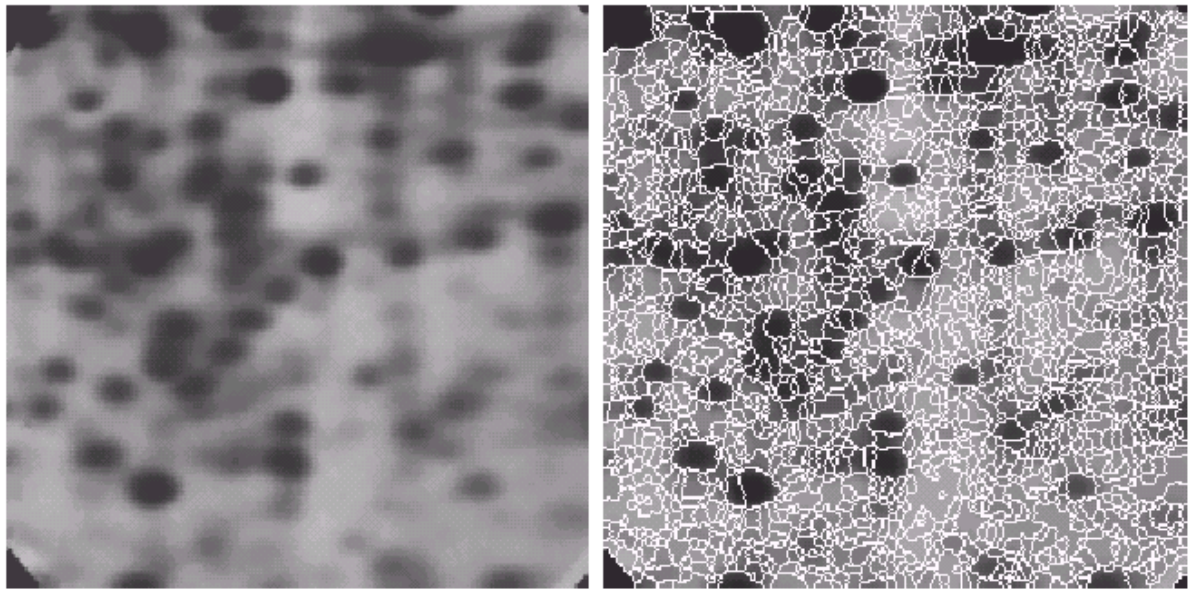
FIGURE 10.46
(a) Image of blobs. (b) Image gradient. (c) Watershed lines. (d) Watershed lines superimposed on original image. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)



Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm

Due to noise and other local irregularities of the gradient, oversegmentation might occur.



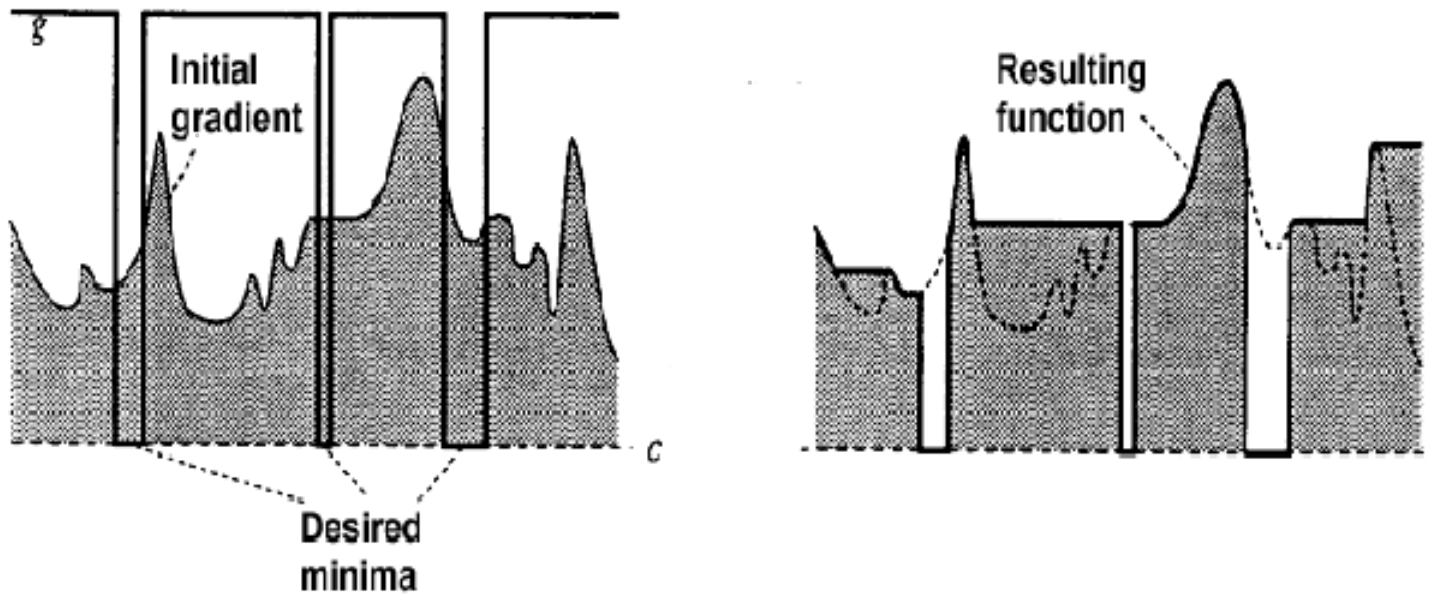
a b

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.)

Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm

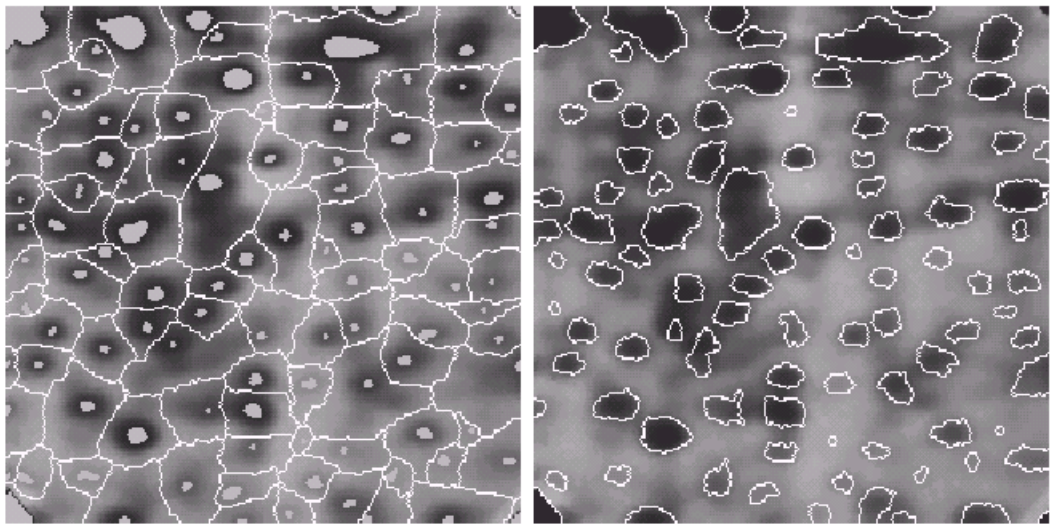
A solution is to limit the number of regional minima. Use markers to specify the only allowed regional minima.



Slide: thanks to Bahadir K. Gunturk, LSU

Watershed Segmentation Algorithm

A solution is to limit the number of regional minima. Use markers to specify the only allowed regional minima. (For example, gray-level values might be used as a marker.)



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.)

Slide: thanks to Bahadir K. Gunturk, LSU

Today's Objectives

- Histogram clustering
  - K-means clustering

Labeling Regions by Flood Filling

- The recursive flood fill algorithm
- Watershed segmentation