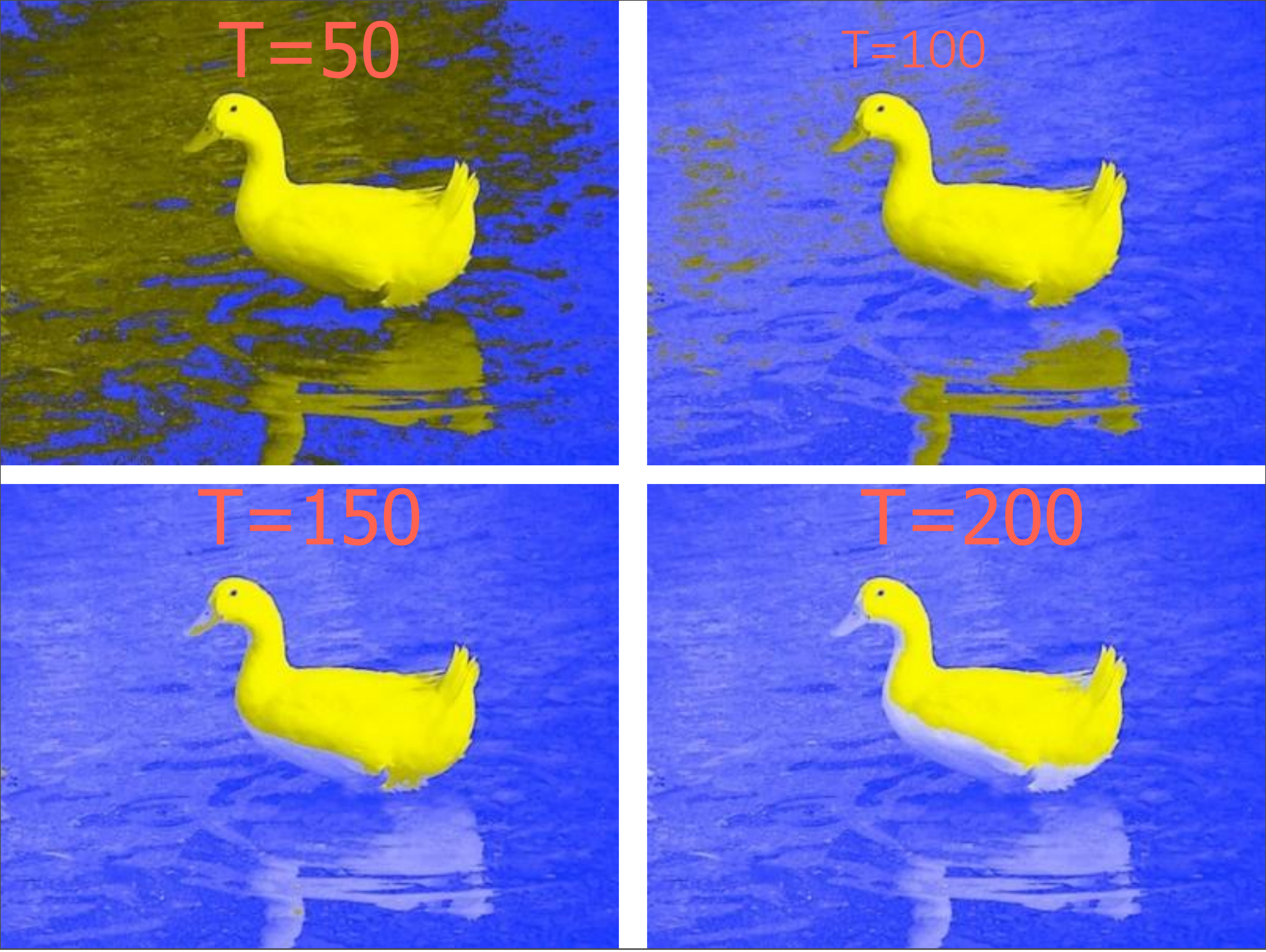
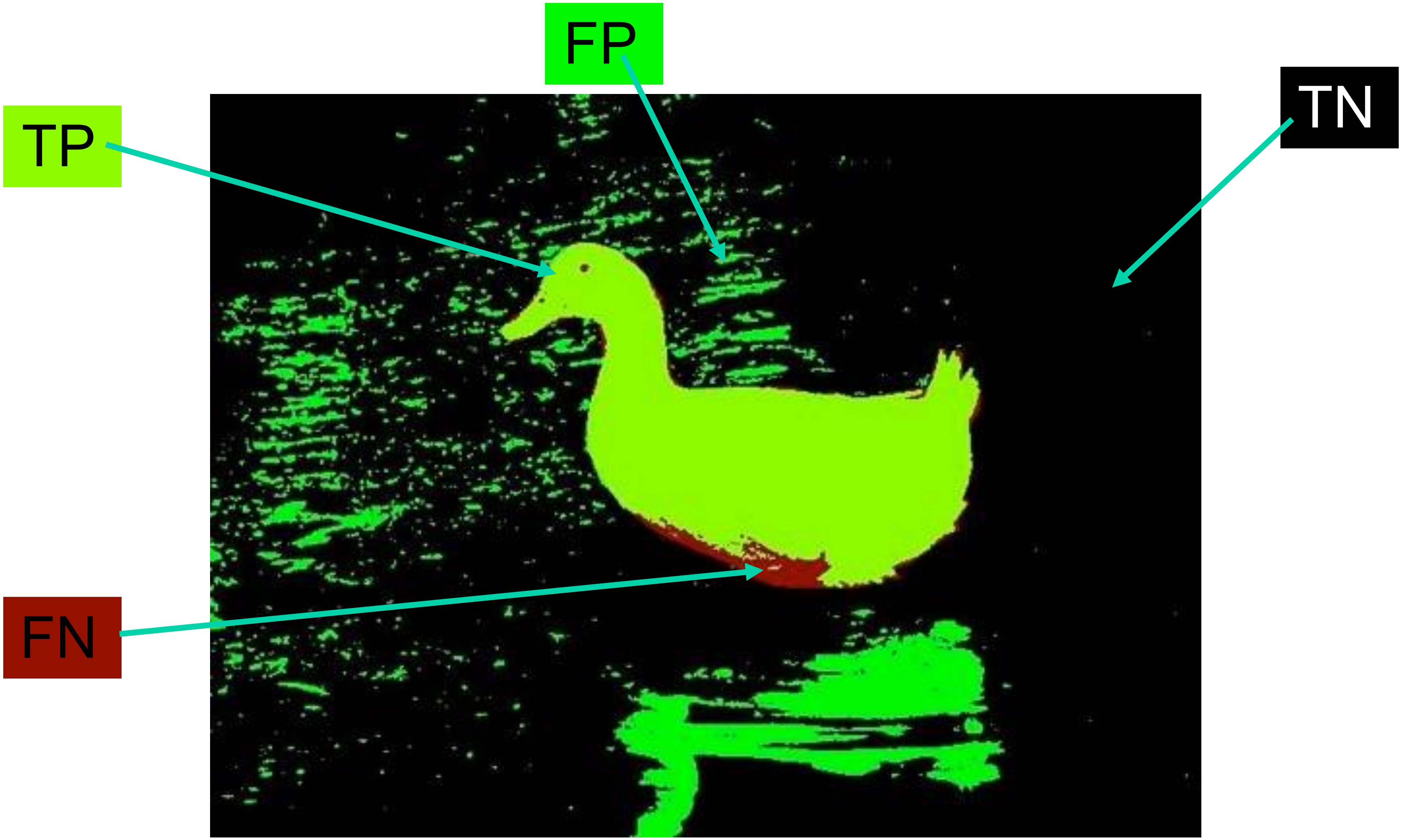

T=50
T=100
T=150
T=200

Classification outcomes



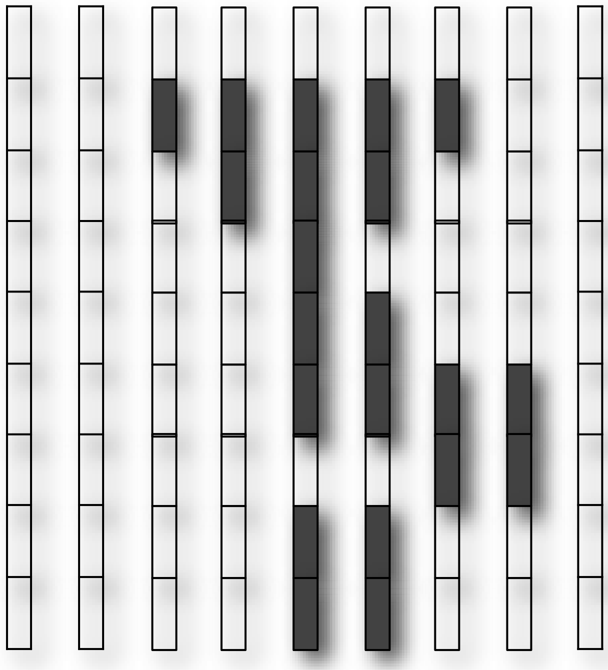
Limitations of Thresholding

- Why can we segment images much better by eye than through thresholding processes?

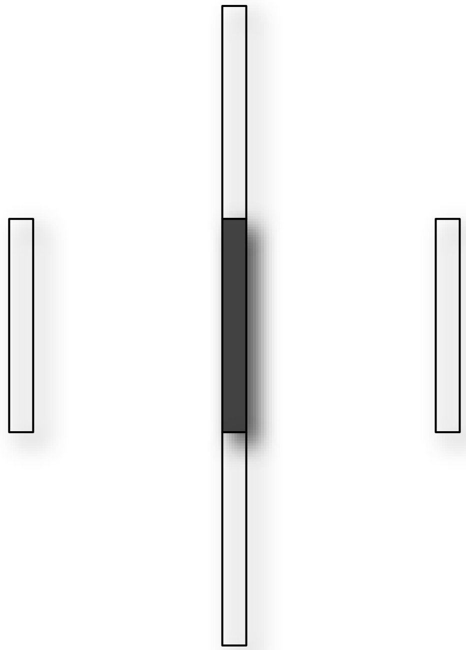
- We might improve results by considering image **context:** Surface Coherence

Pixel Connectivity

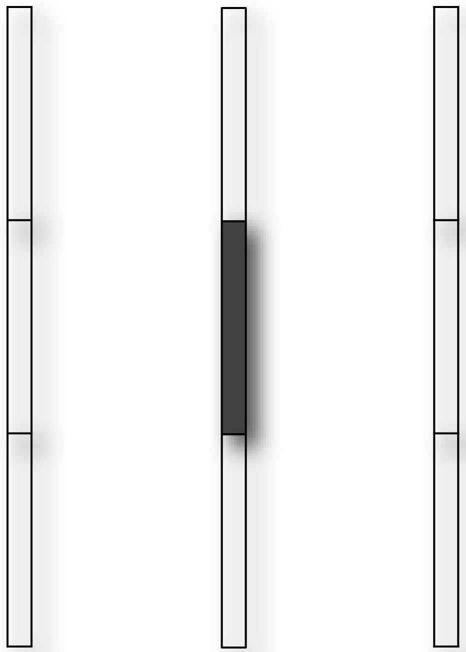
- We need to define which pixels are neighbours.
- Are the dark pixels in this array connected?



Pixel Neighborhoods



4-neighbourhood



8-neighbourhood

Pixel Paths

- A 4-connected path between pixels p_1 and p_n is a set of pixels $\{p_1, p_2, \ldots, p_n\}$ such that p_i is a 4-neighbour of p_{i+1}, i=1,…,n-1.

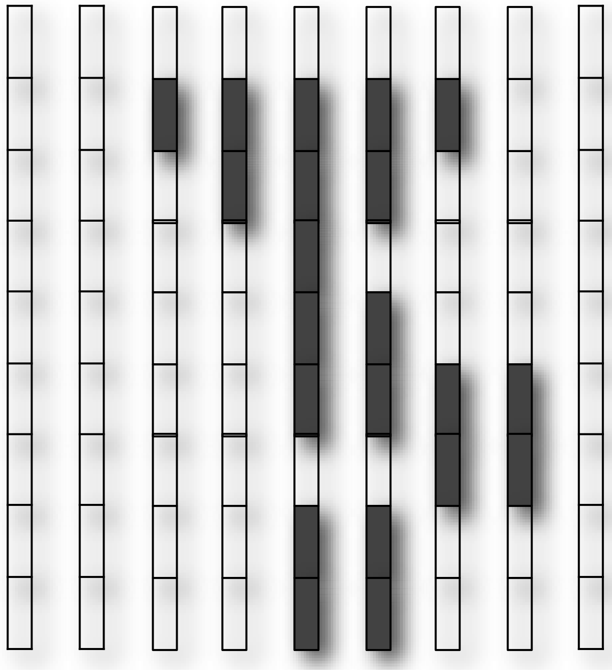
- In an 8-connected path, p_i is an 8-neighbour of p_{i+1}.

Connected Regions

- A region is 4-connected if it contains a 4-connected path between every pair of its pixels.
- A region is 8-connected if it contains an 8-connected path between every pair of its pixels.

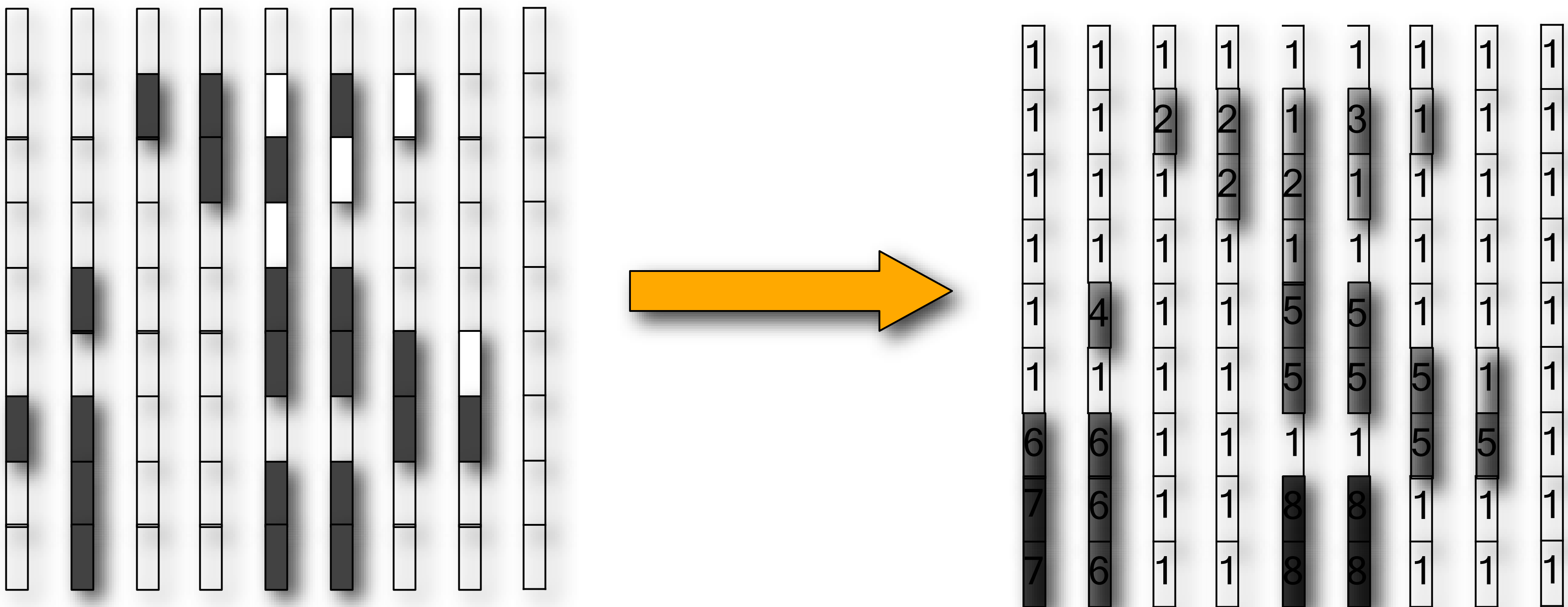
Connected Regions

- Now what can we say about the dark pixels in this array?
- What about the light pixels?



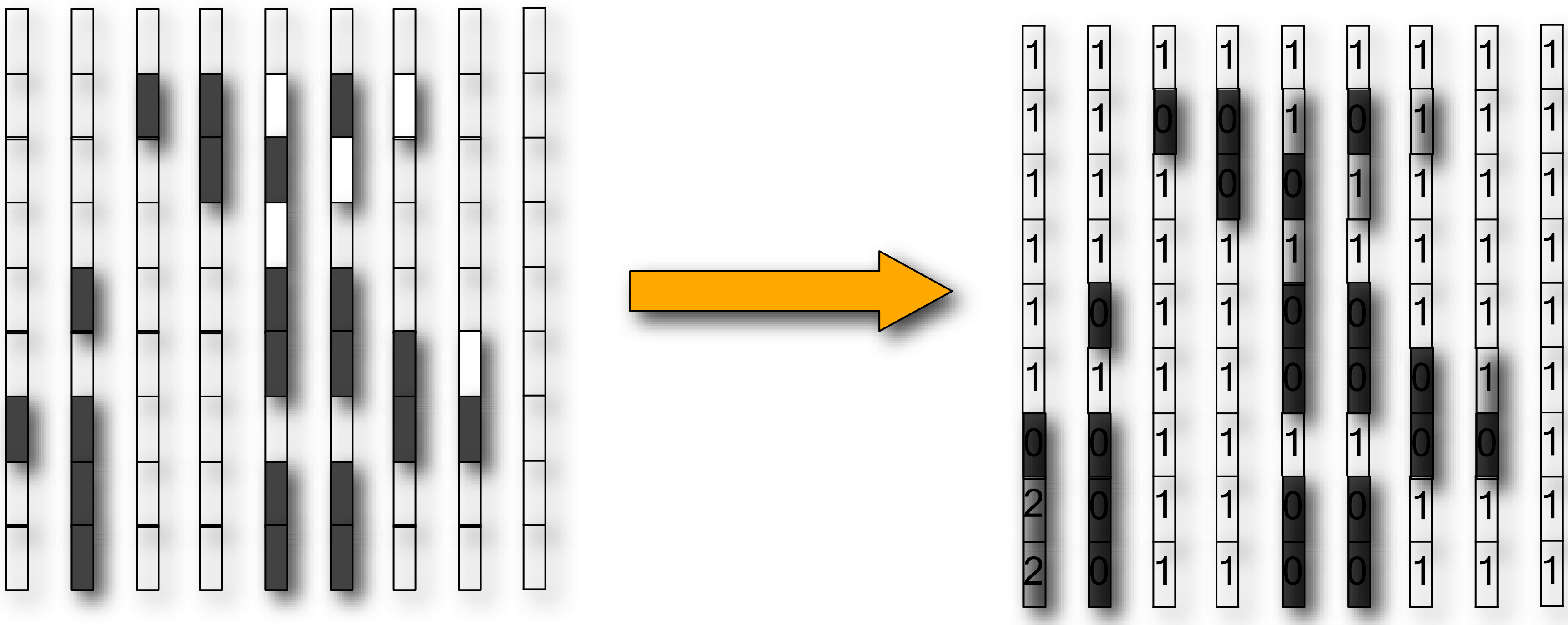
Connected Components Labelling

- Labels each connected component of a binary image with a separate number.



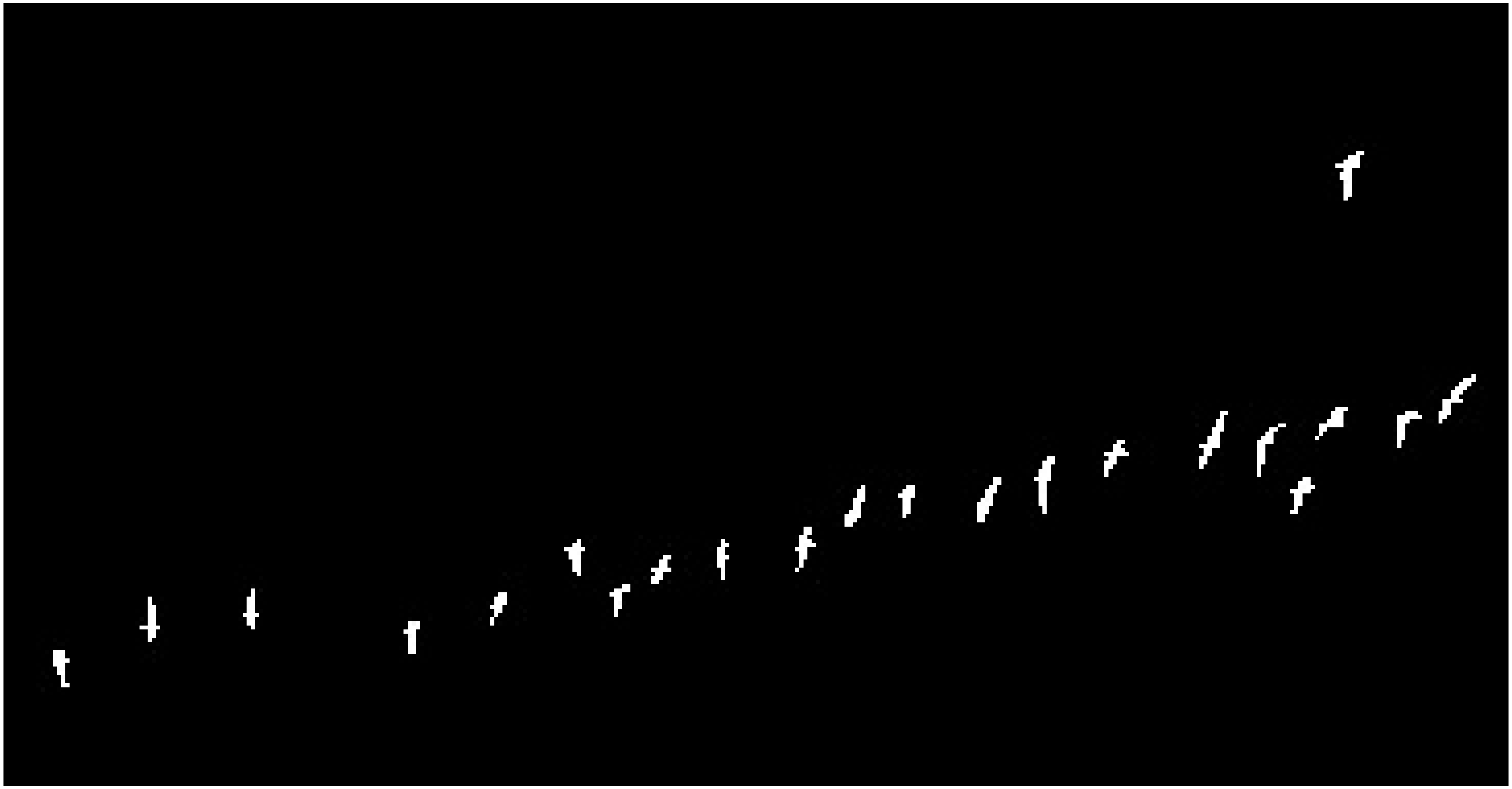
Foreground Labelling

- Only extract the connected components of the foreground

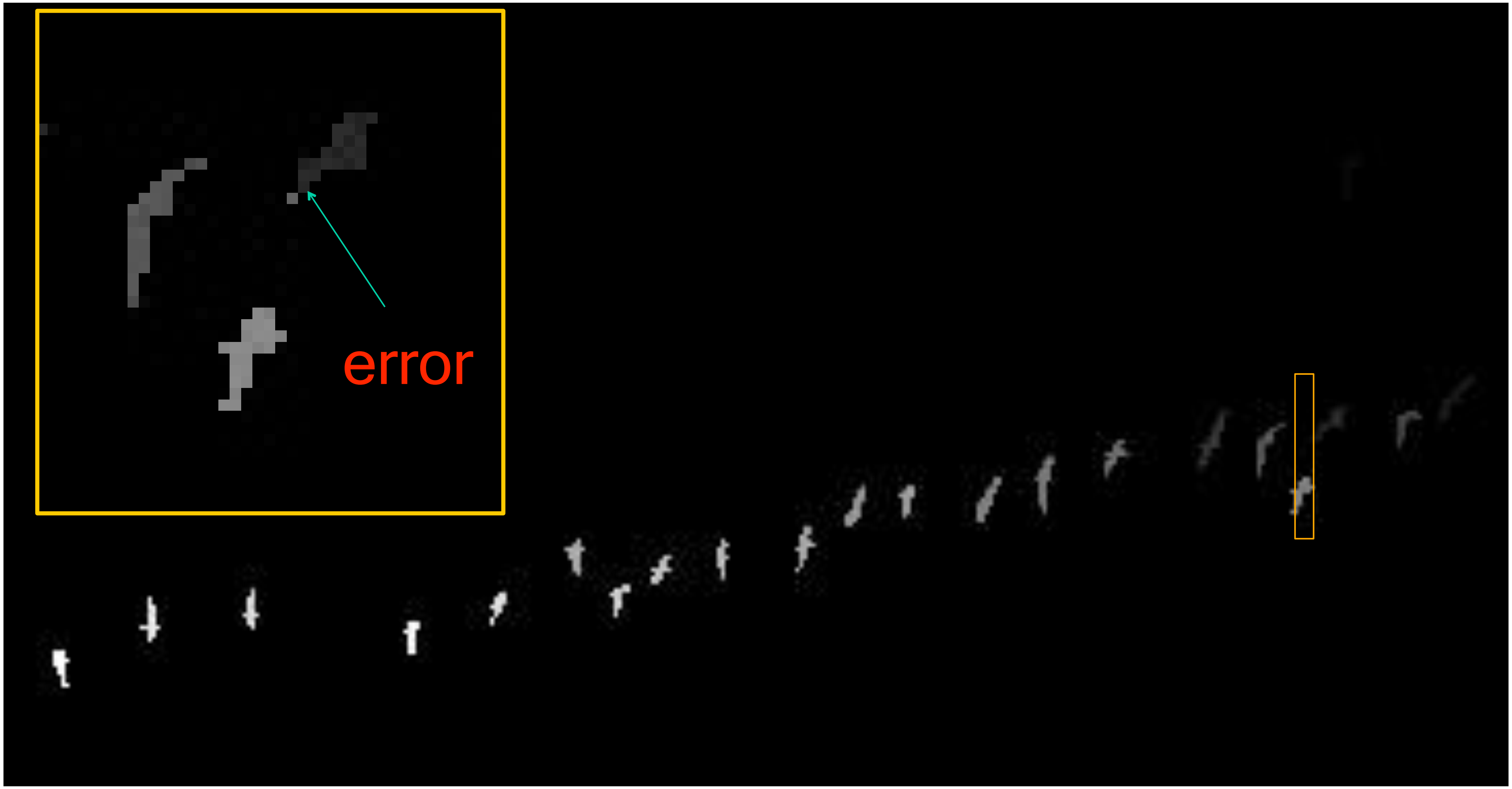


Goose Detector

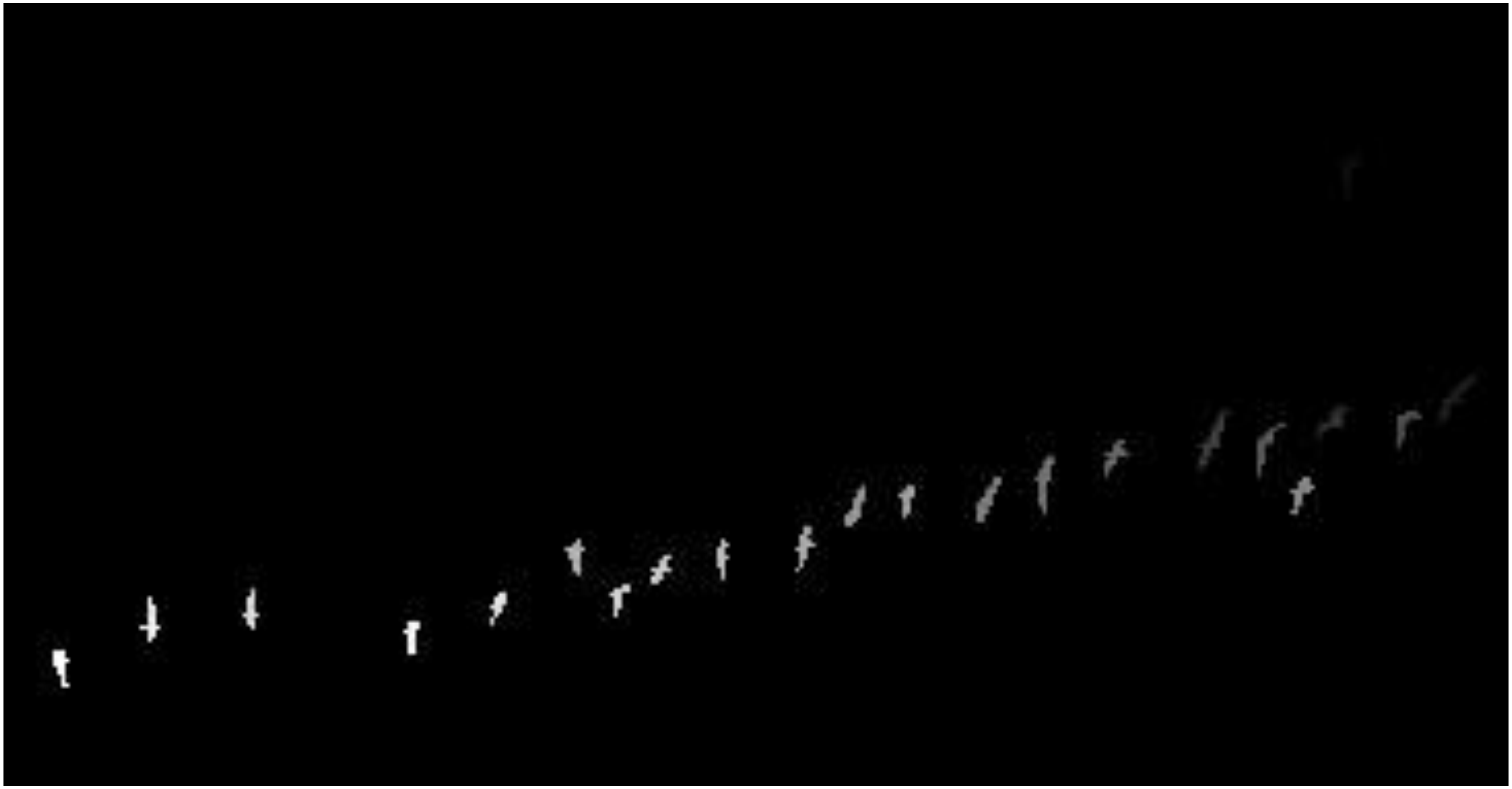
Goose Detector



Goose 4--components (26)

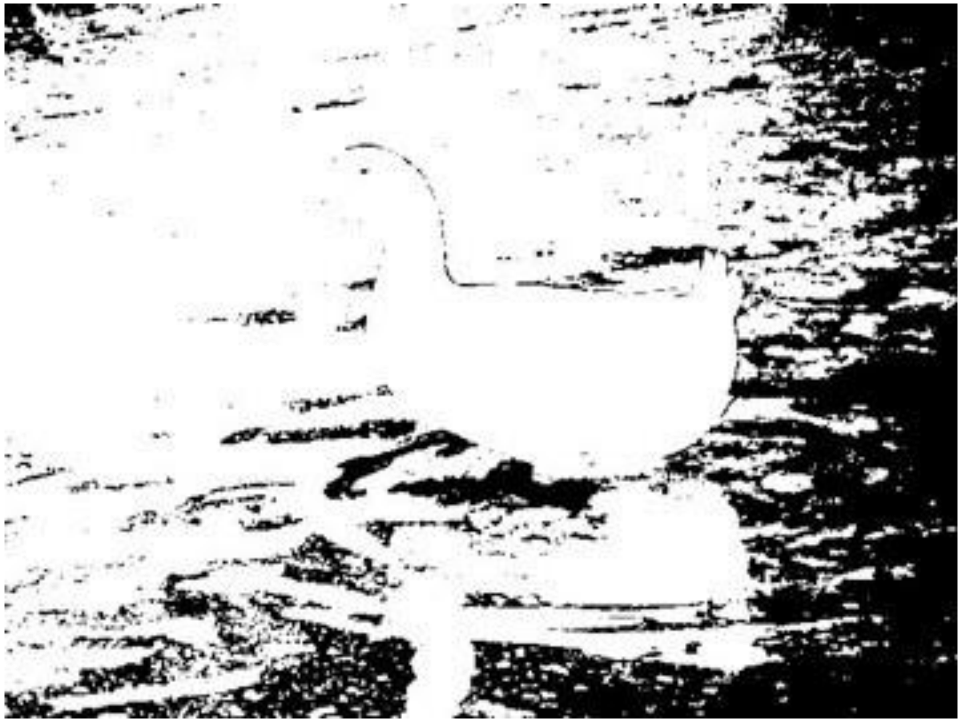


Goose 8--components (22)



Connected Components

- What happens if we use the connected components algorithm on this image?



- How might we improve our implementation?

Region Growing

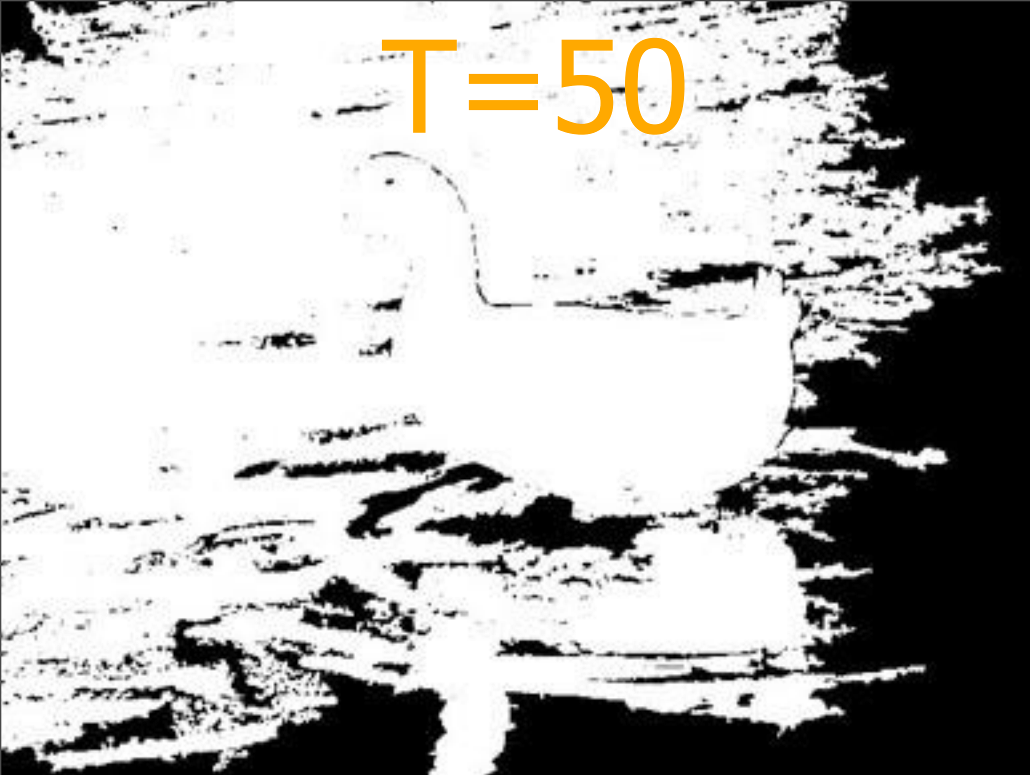
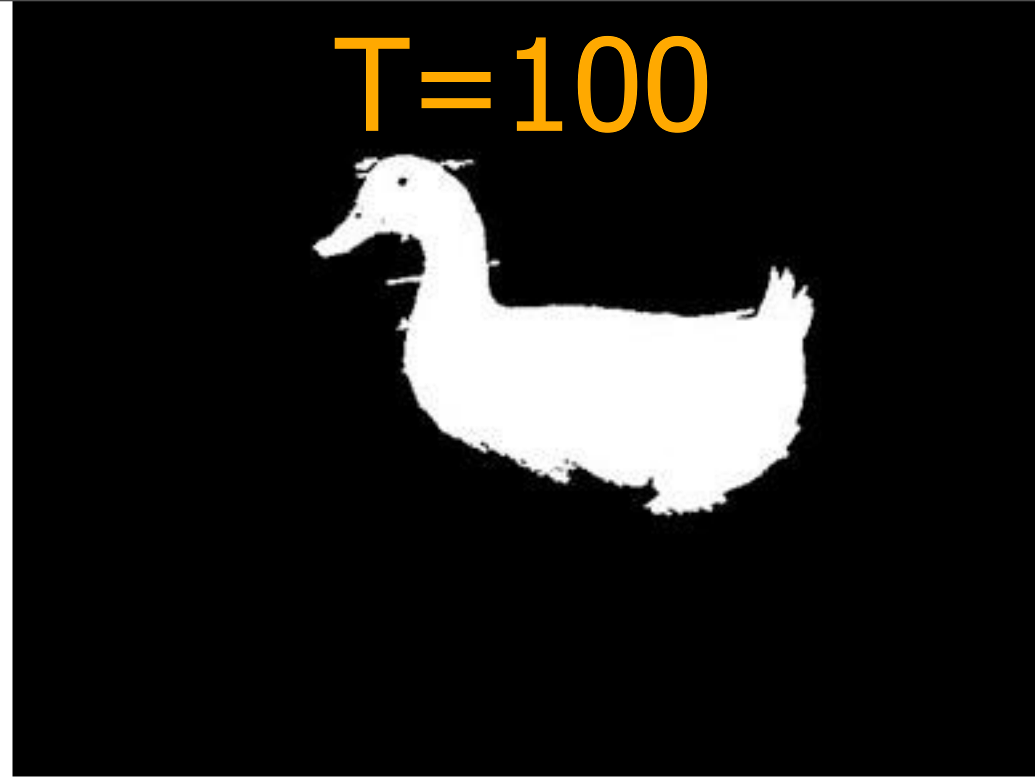
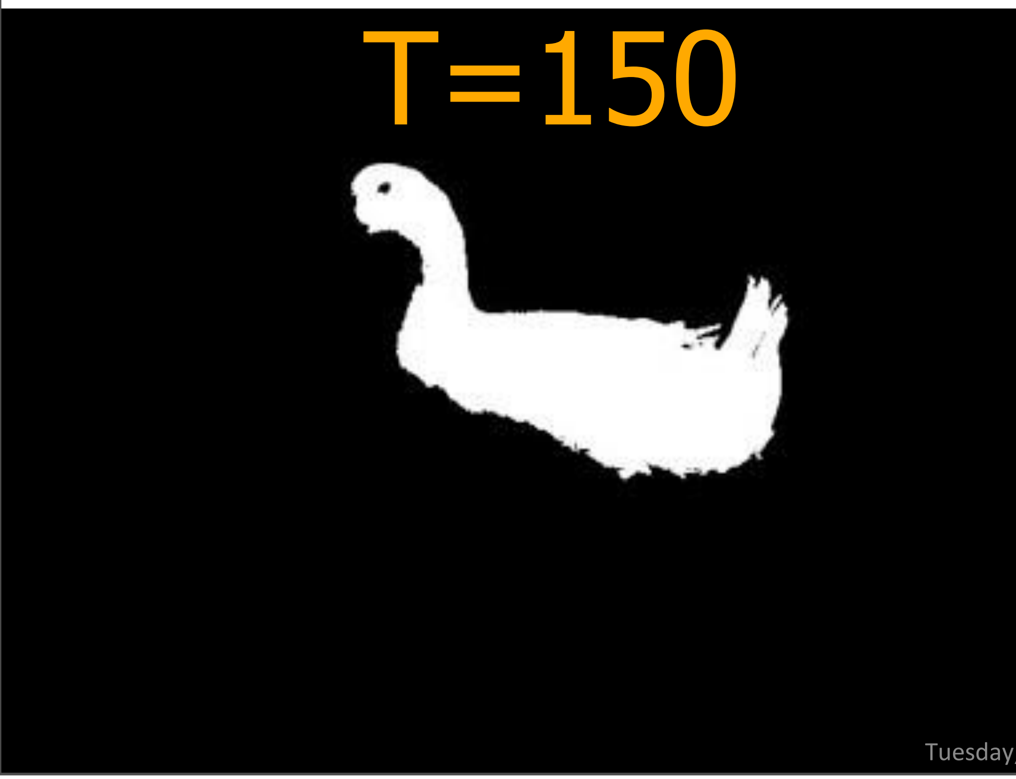
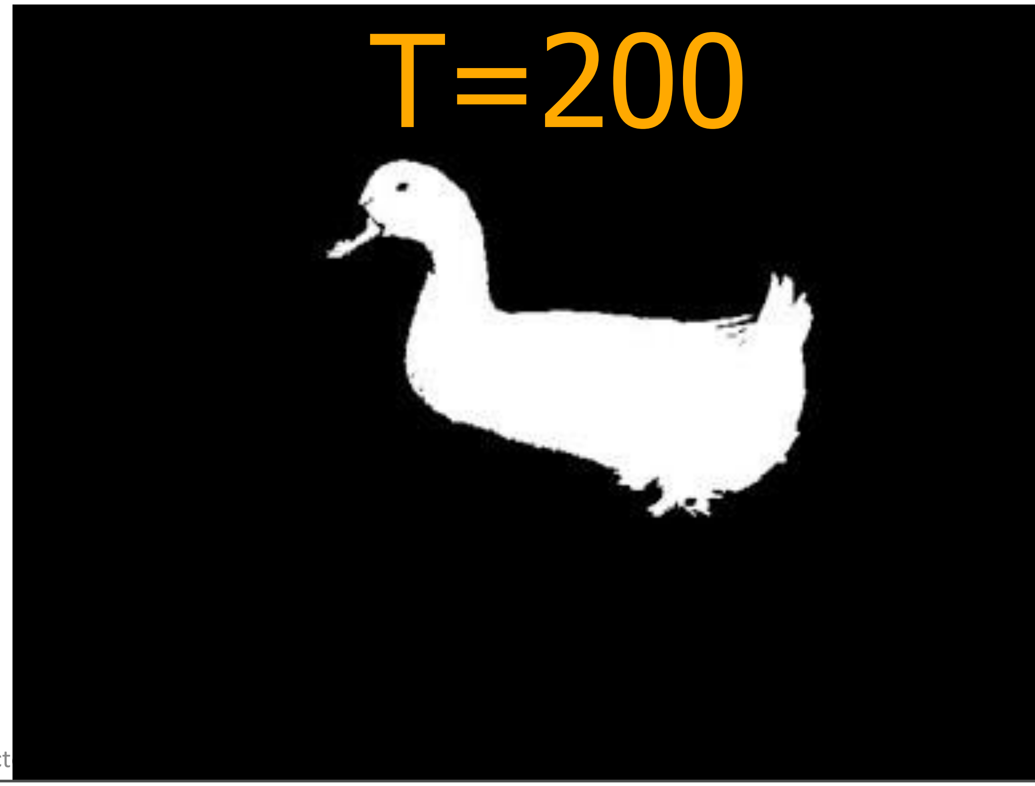
- Start from a seed point or region.
- Add neighbouring pixels that satisfy the criteria defining a region.
- Repeat until we can include no more pixels.

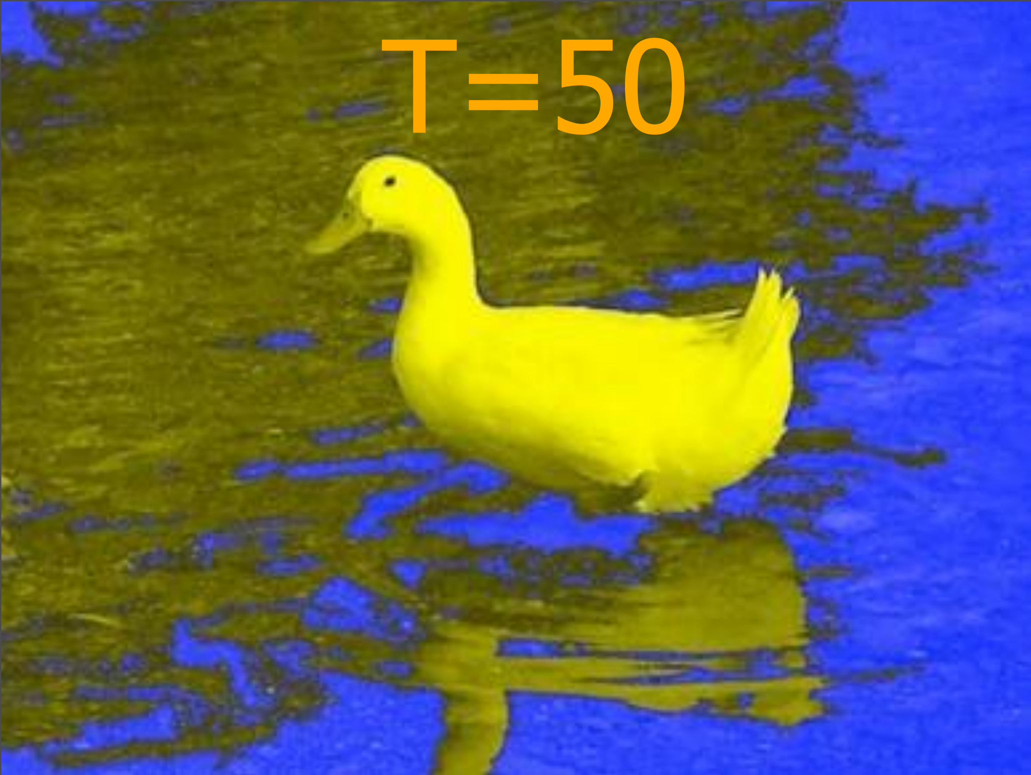
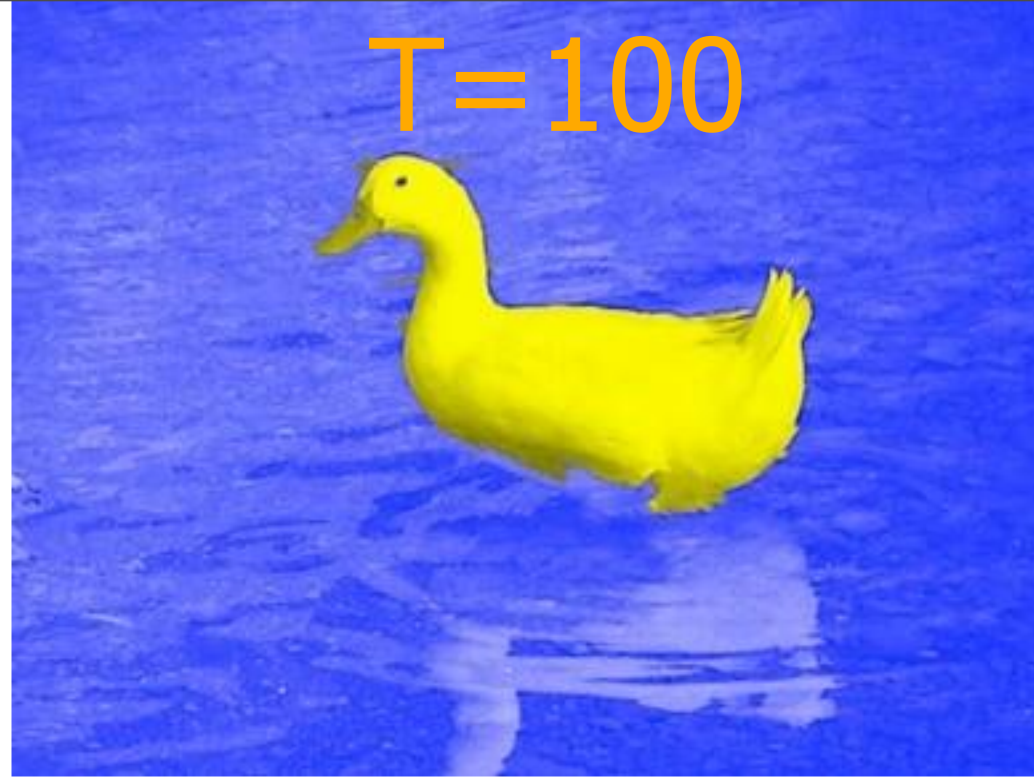
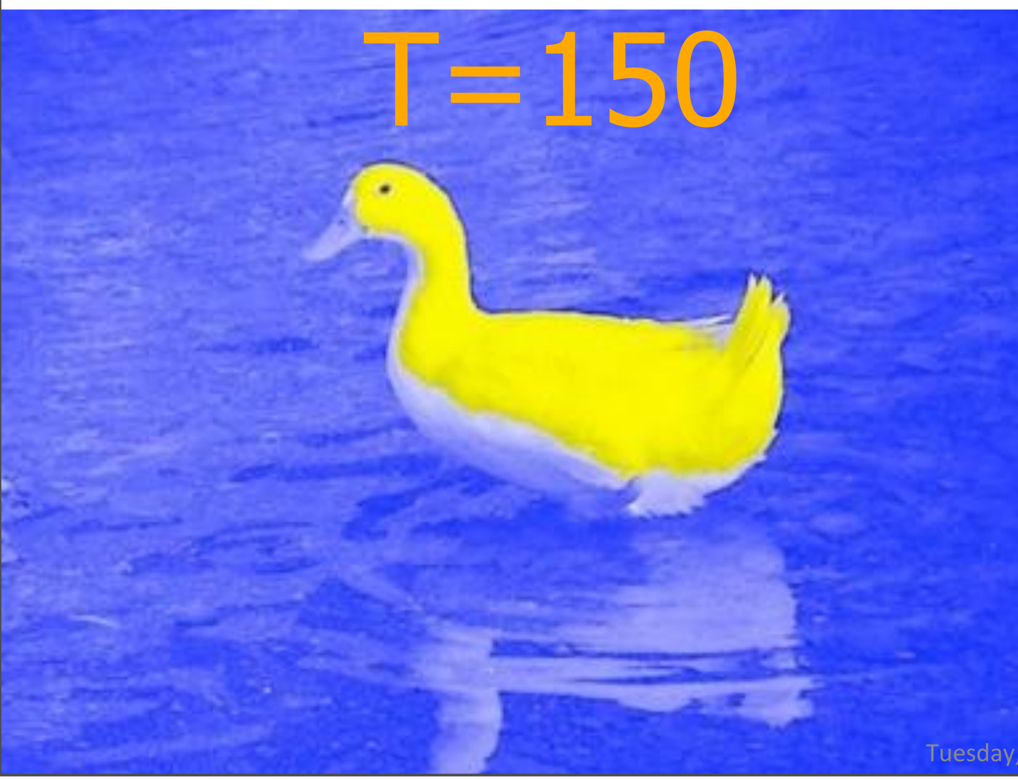
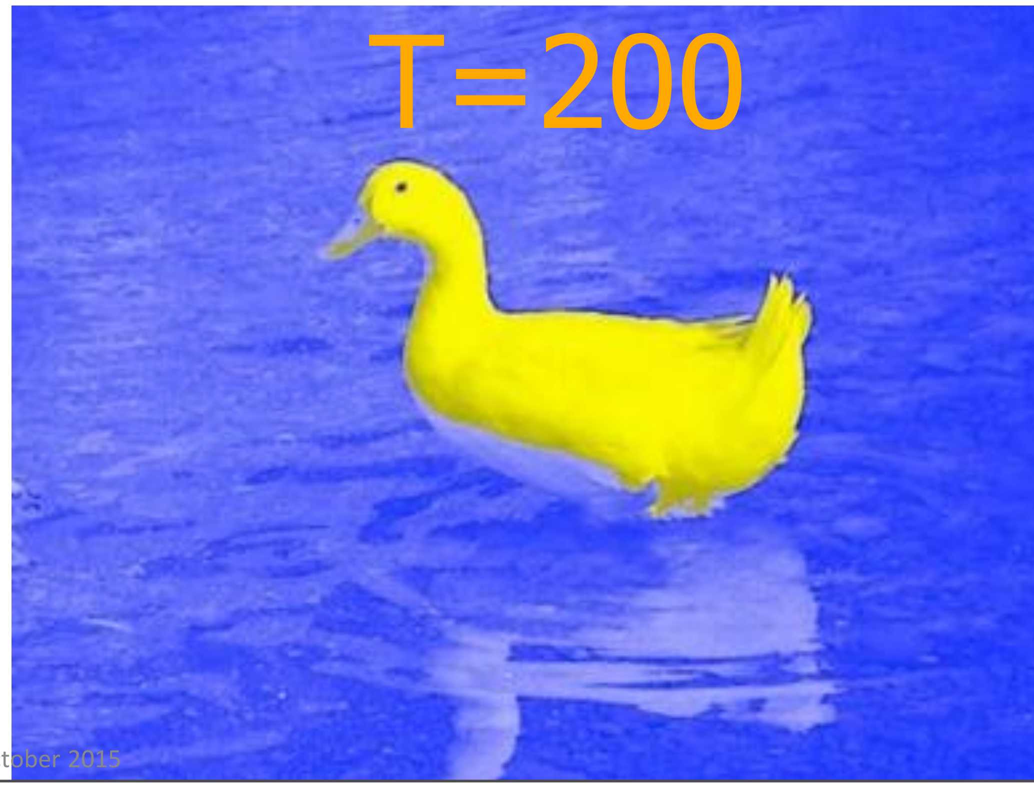
Region Growing example

- Pick a single seed pixel.
- Inclusion test is a simple grey level threshold:

```
function test = include(p)
    test = (p>=T);
```

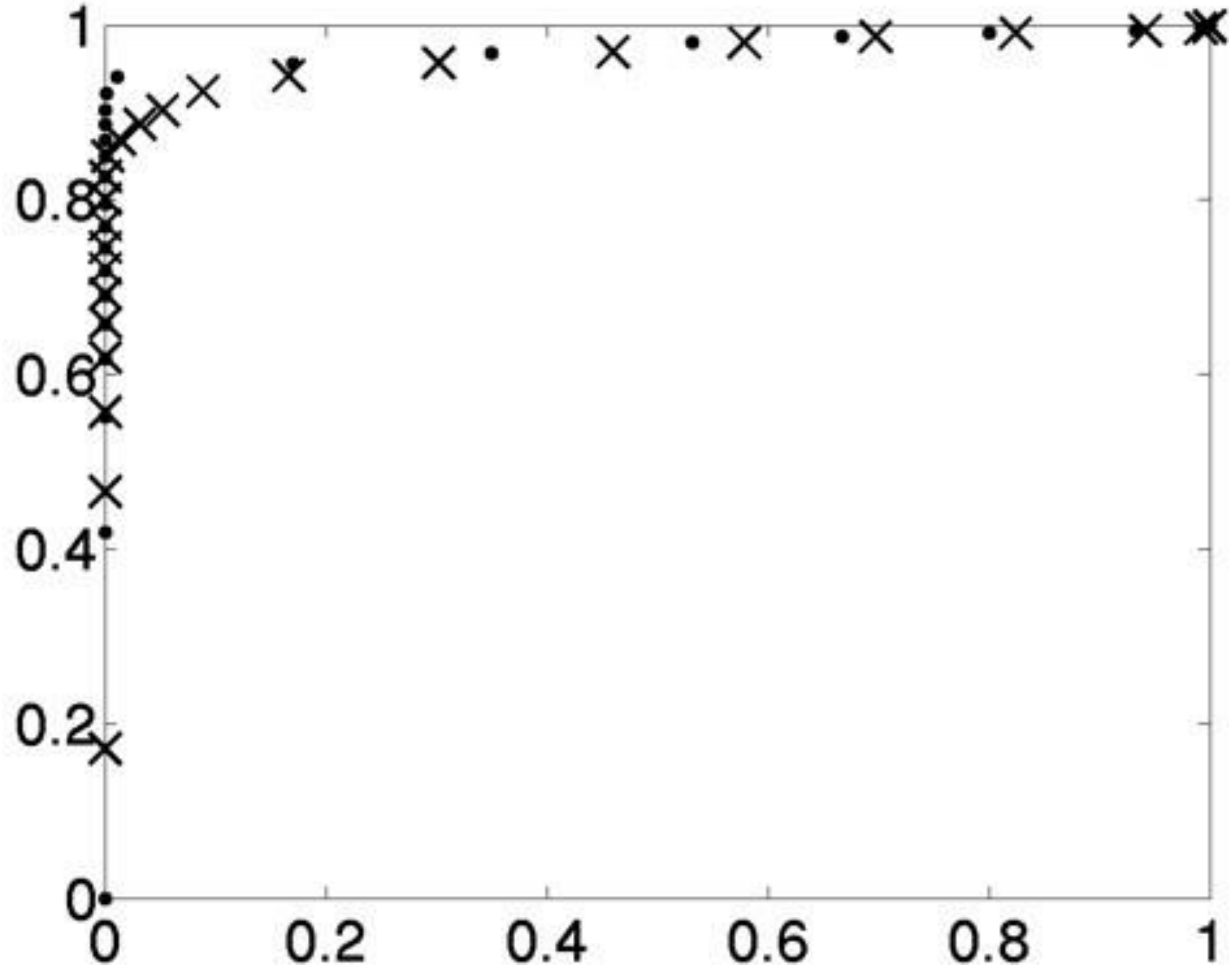
Seed pixel


T=50

T=100

T=150

T=200


T=50

T=100

T=150

T=200

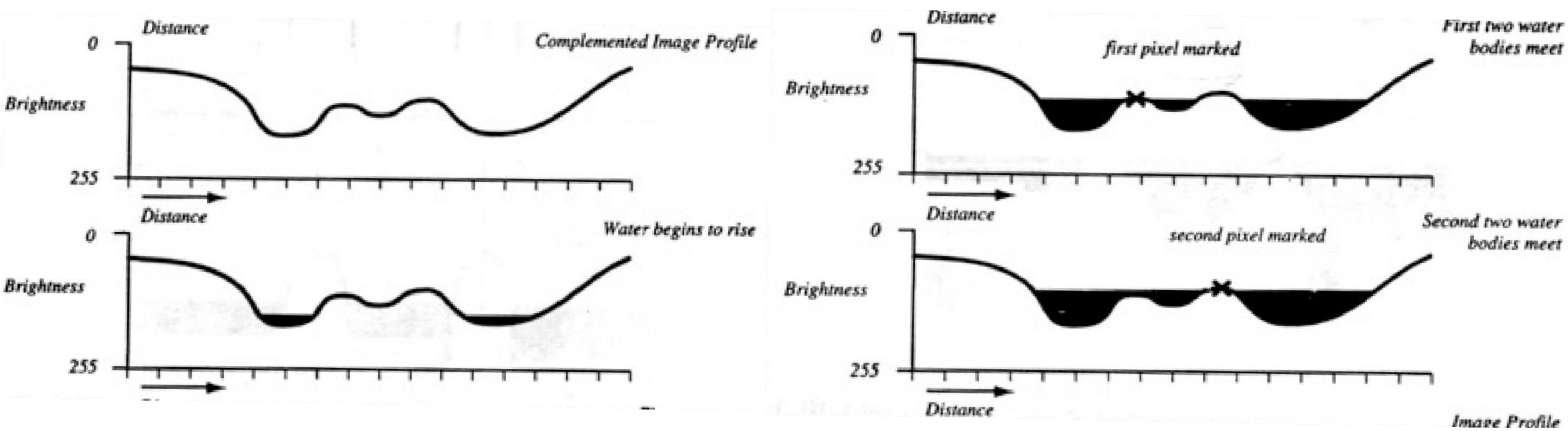
ROC Curve

- • Region growing
- × Thresholding

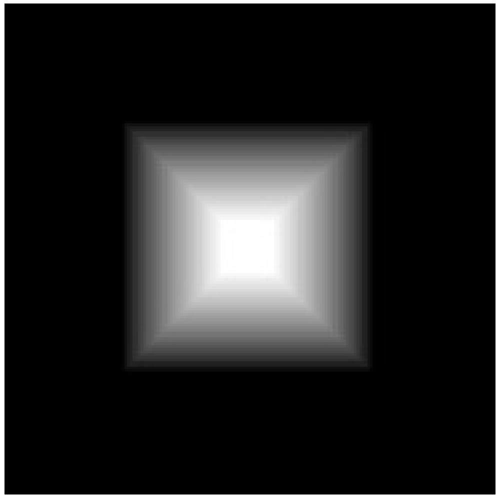


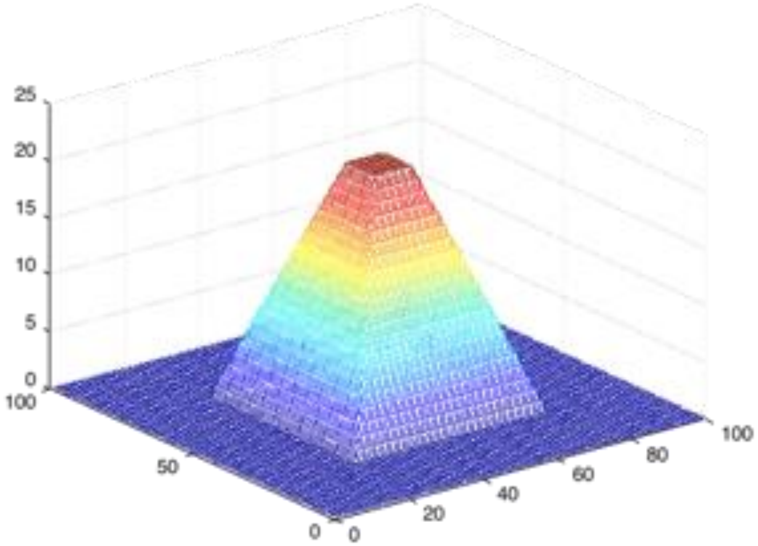
Watershed Algorithm

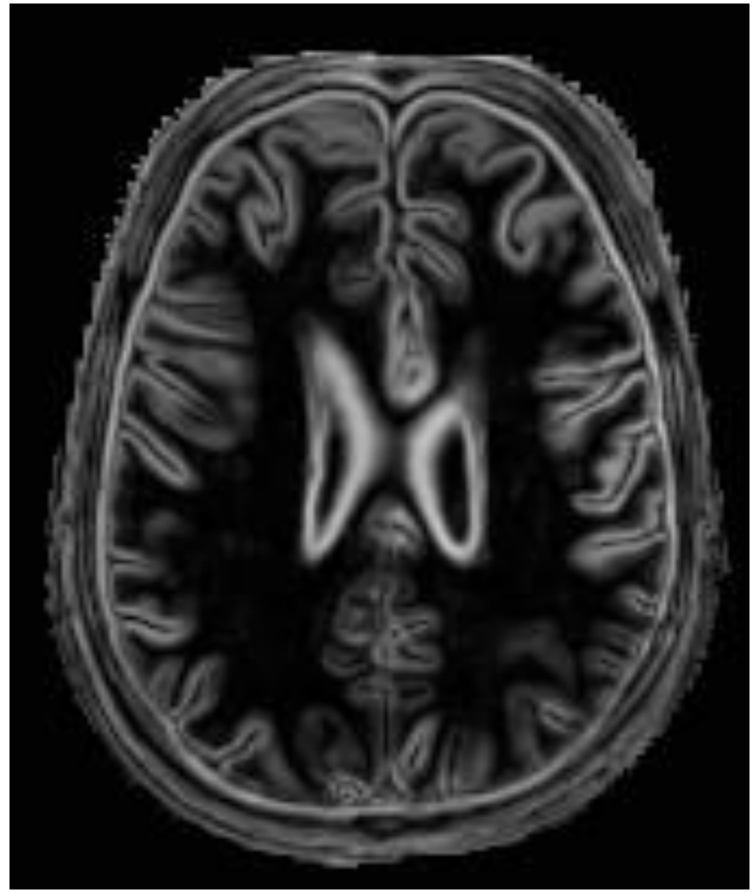
- The objective is to find watershed lines.
- Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate.
- When rising water in distinct catchment basins is about to merge, a dam is built to prevent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

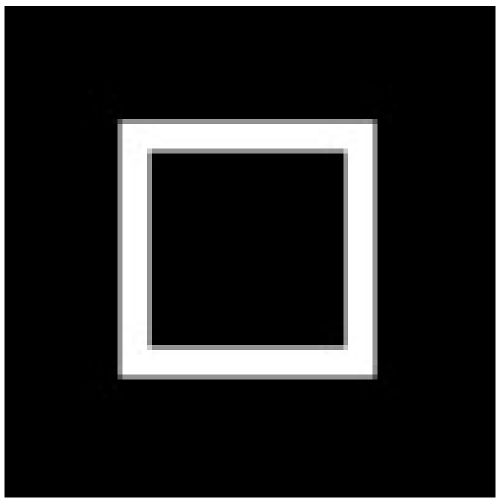


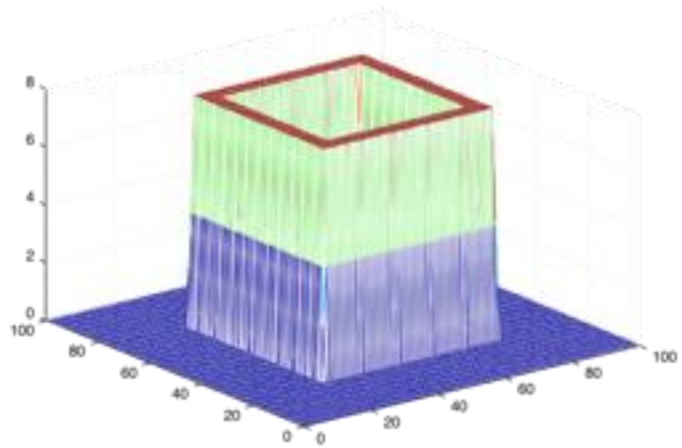
Watershed Segmentation











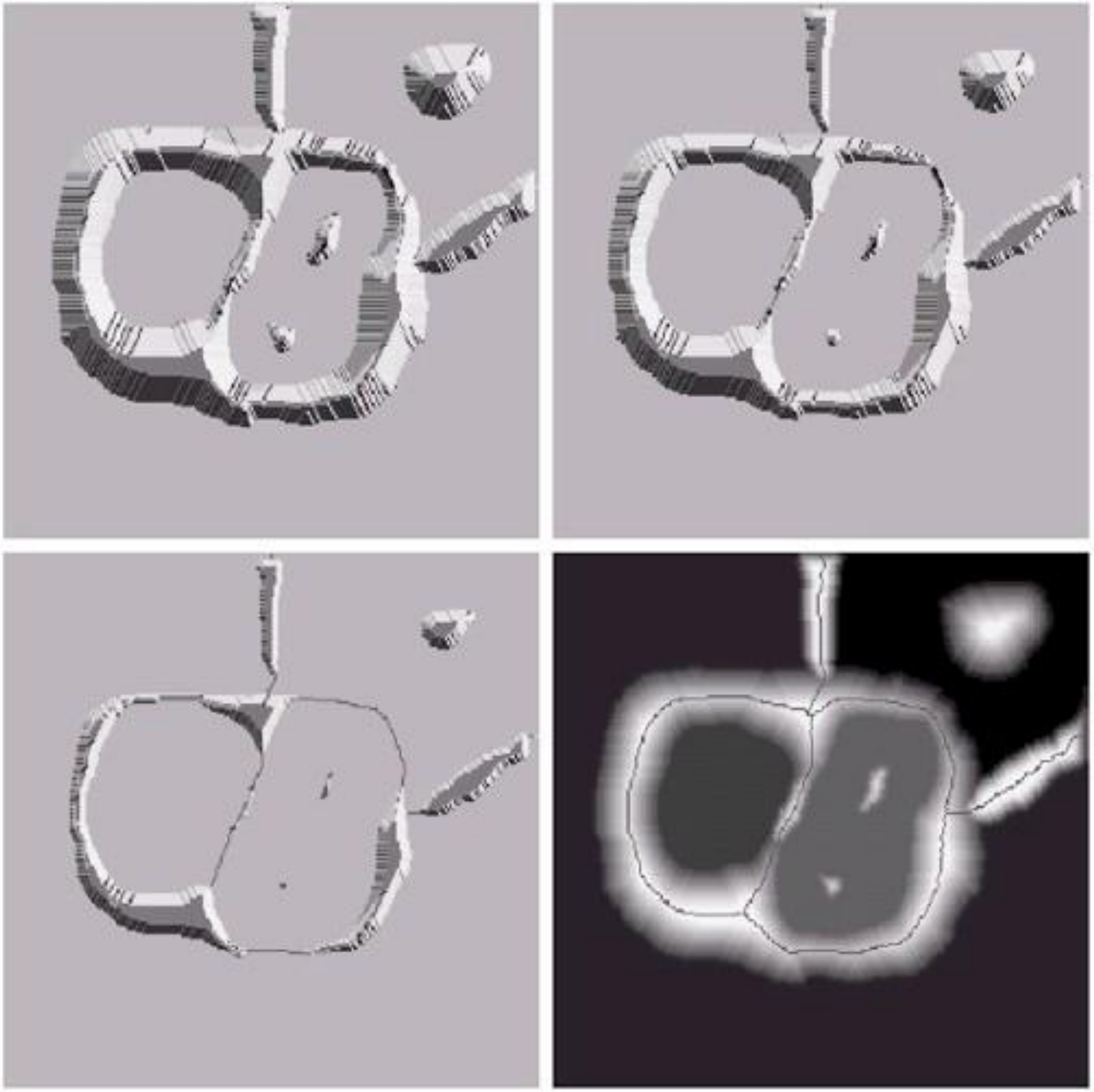
In this case, each object is distinguished from the background by its raised edges

Basic Steps

1. Piercing holes in each regional minimum of I

2. The 3D topography is flooded from below gradually

3. When the rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging



c 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.)

4. The dam boundaries correspond to the watershed lines to be extracted by a watershed segmentation algorithm

Watershed Algorithm

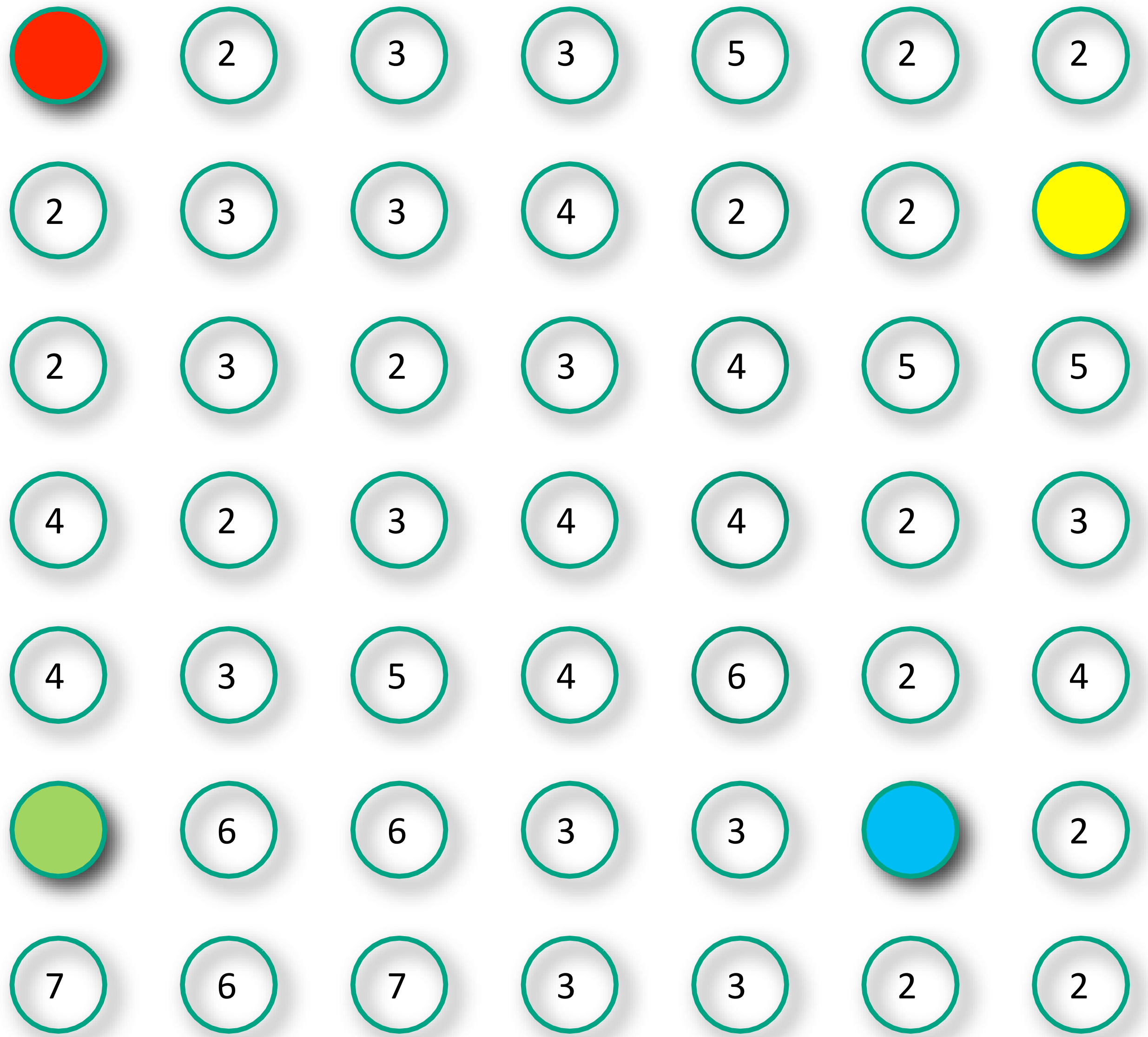
```
Set label at every pixel to -1

For each seed point u
For each neighbour v of u
    Label[v] = label [u]
Add v to priority queue: priority by gray level

while(elements in queue)
  u = popMinimum(Queue);
  For each neighbour v of u
    If (label[v]!==-1)
      Label[v] = label [u]
Add v to priority queue: priority by gray level
```

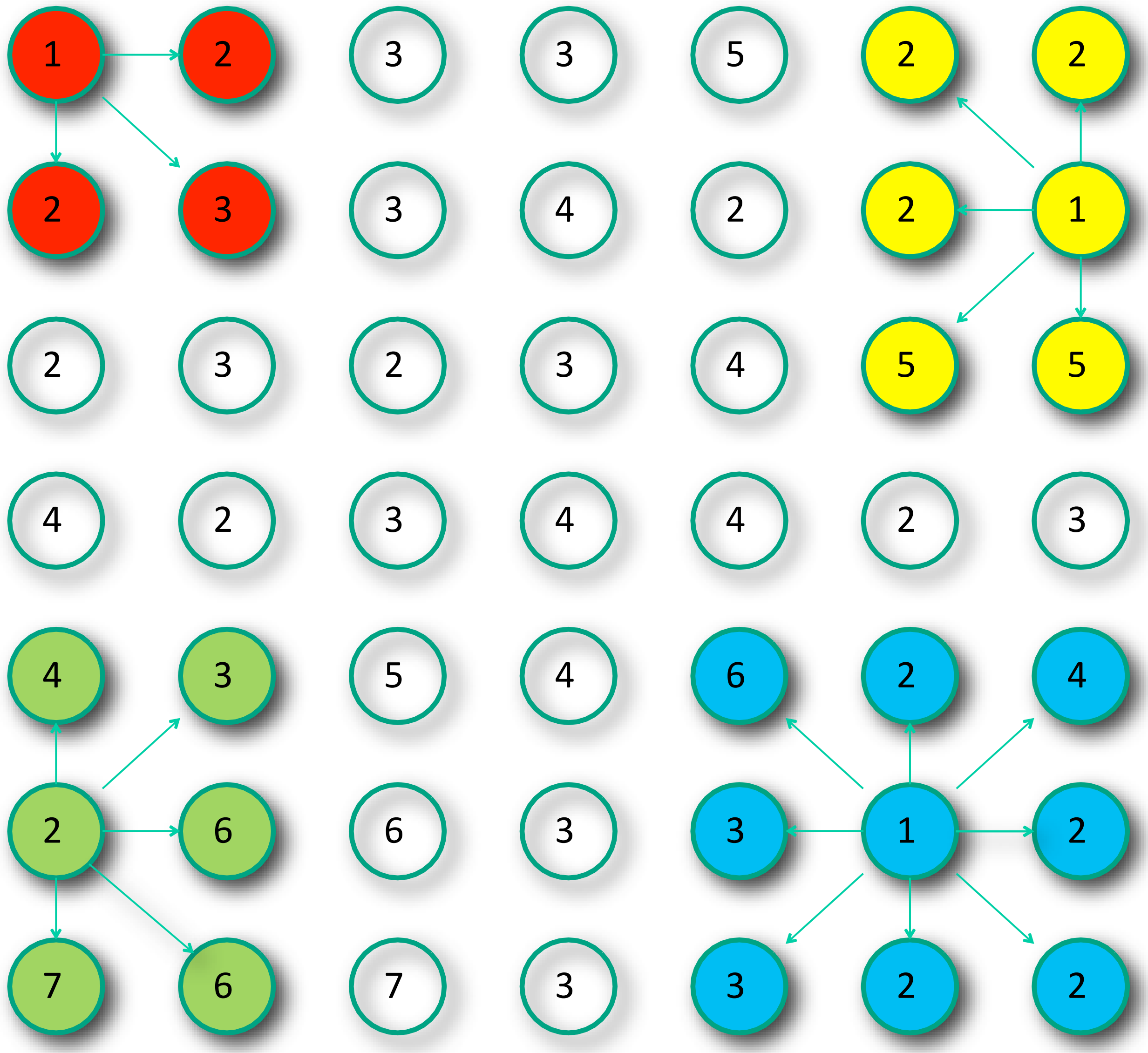
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW



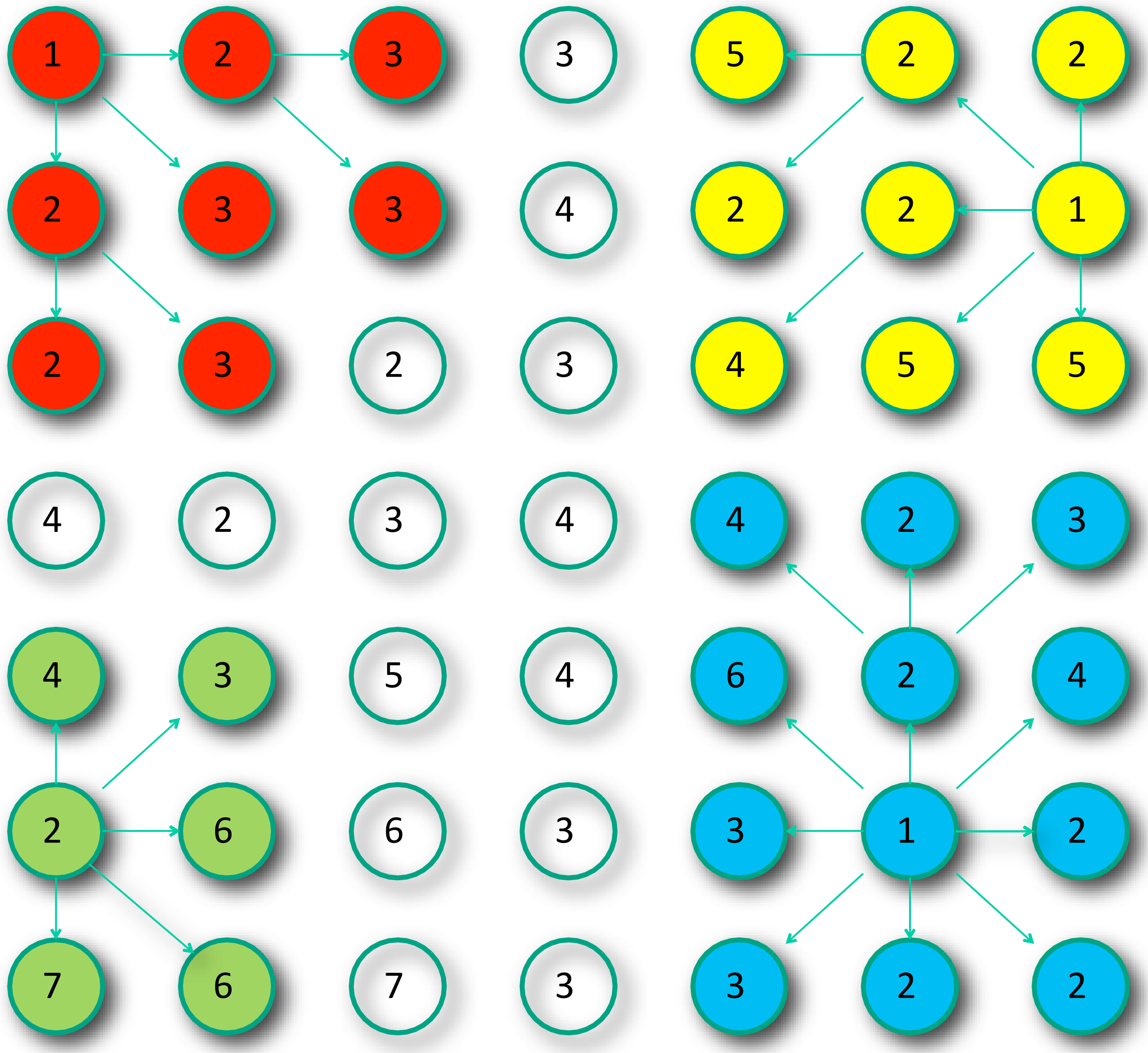
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW



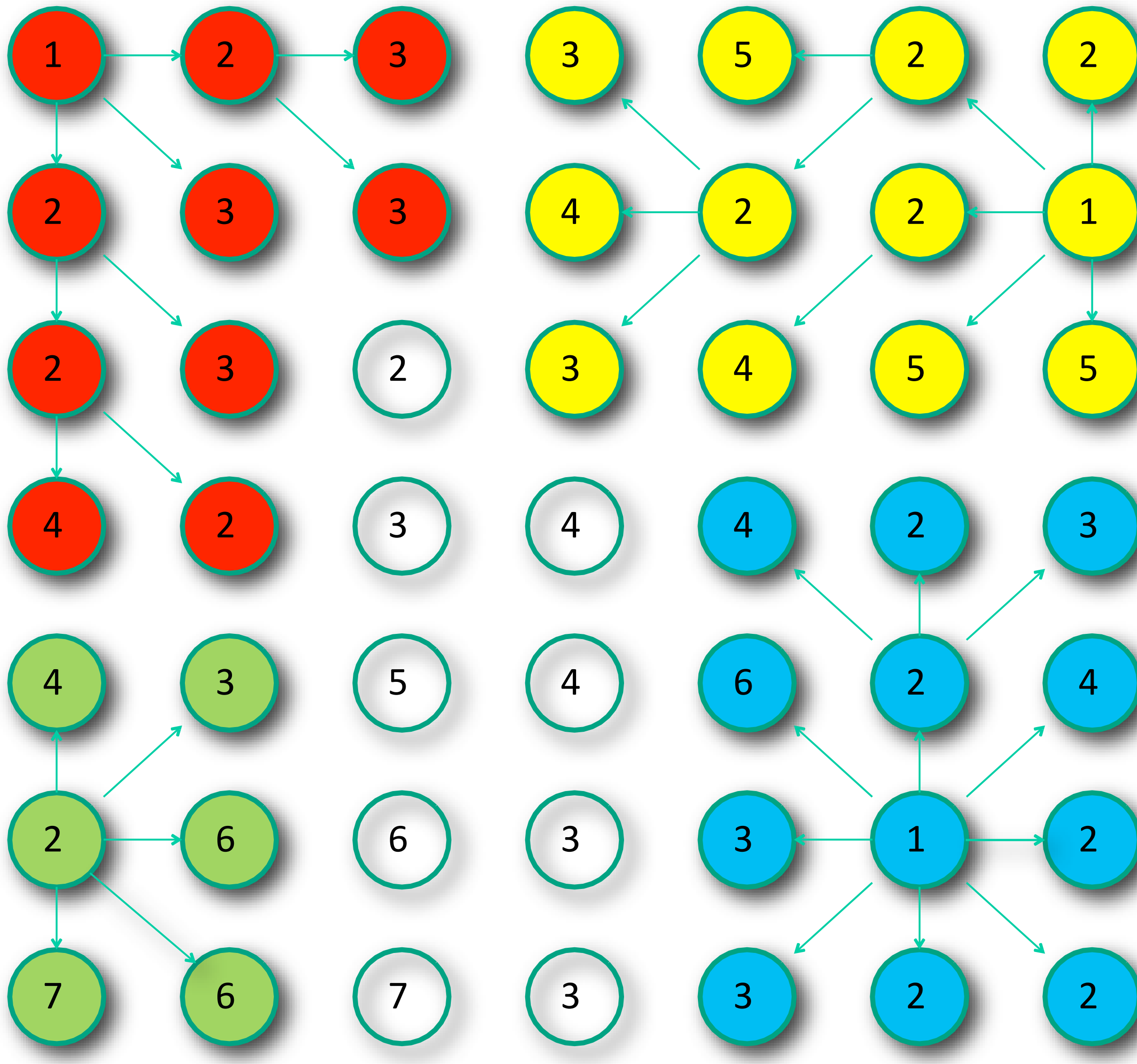
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW



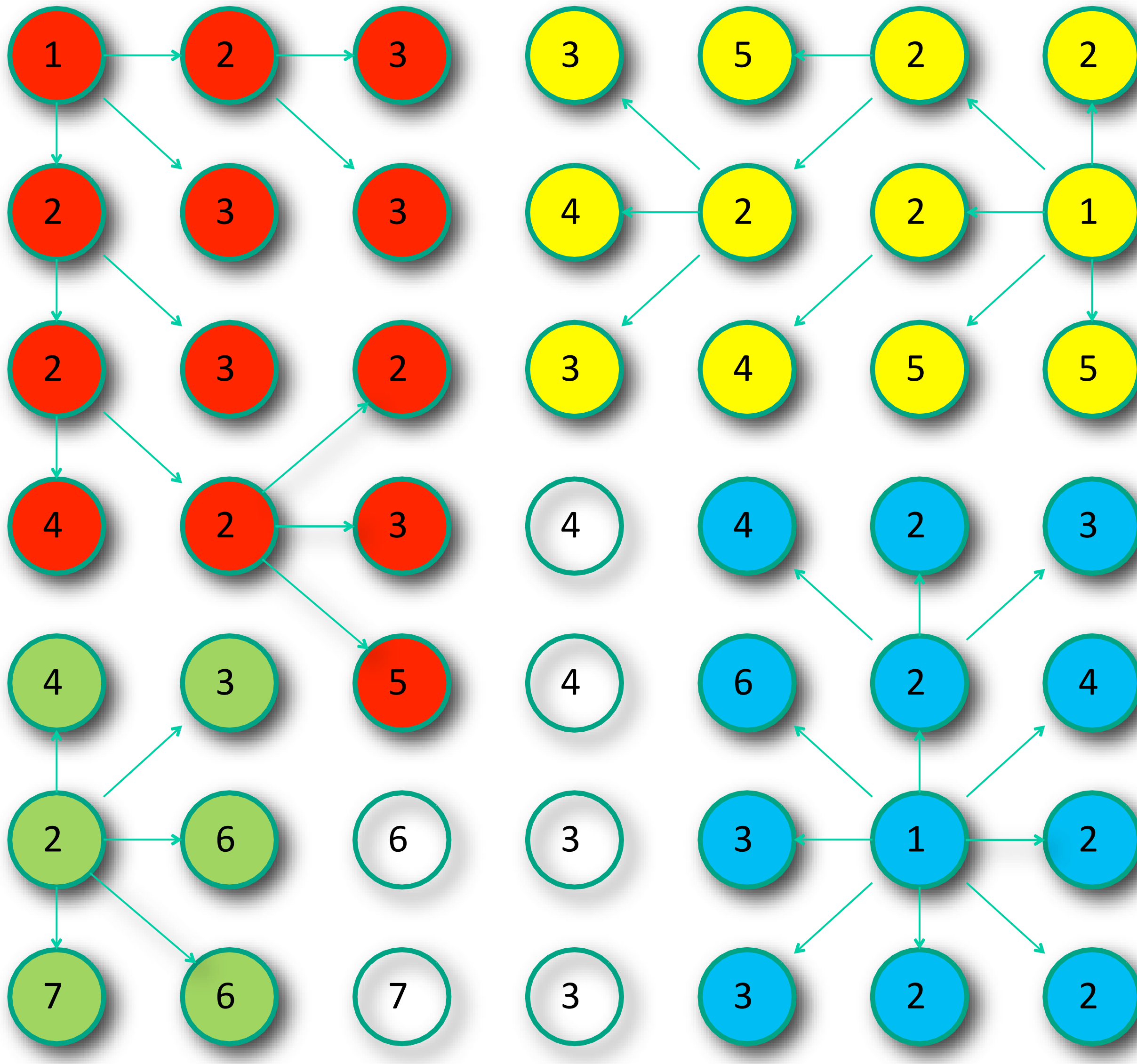
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW



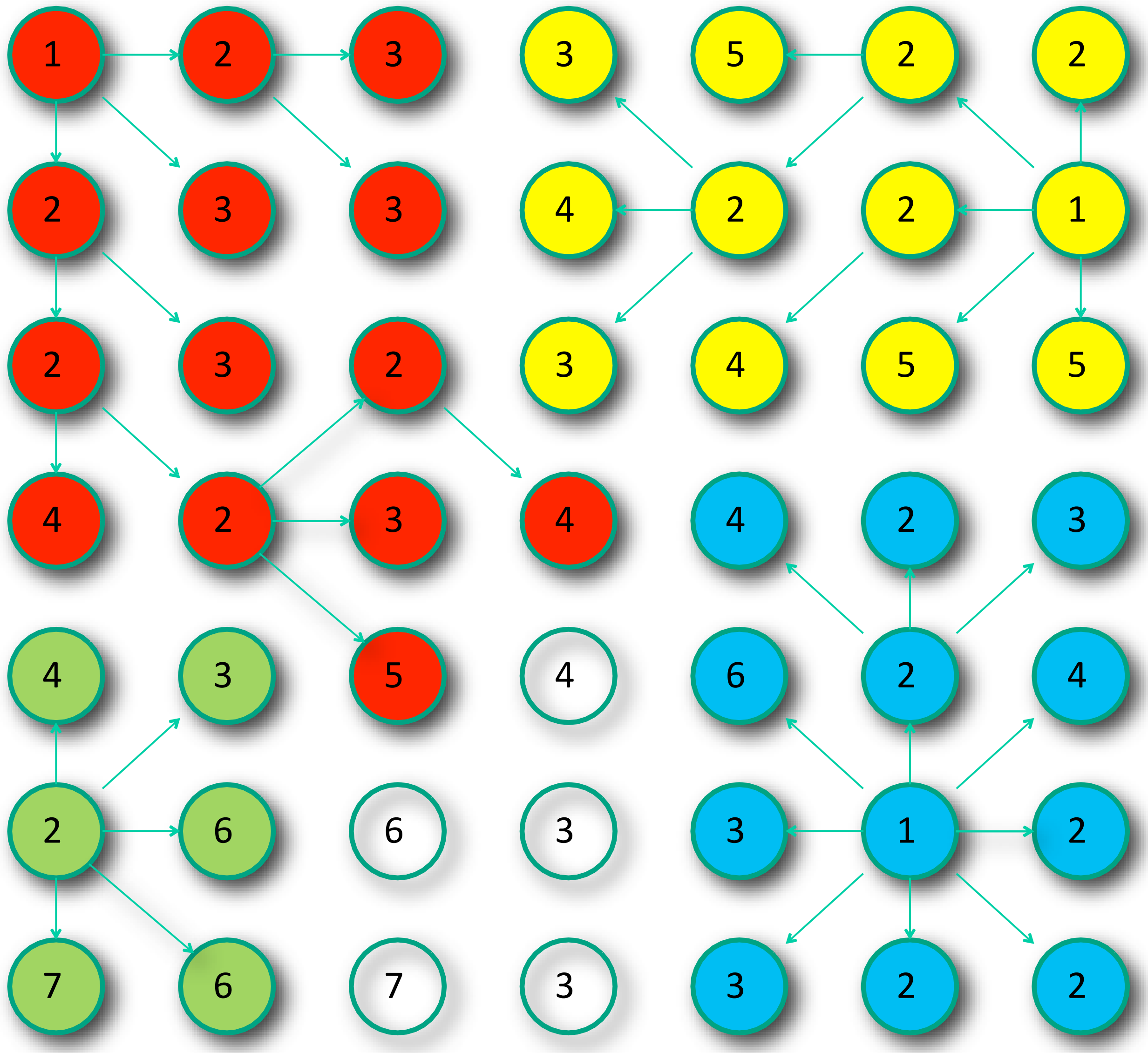
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW



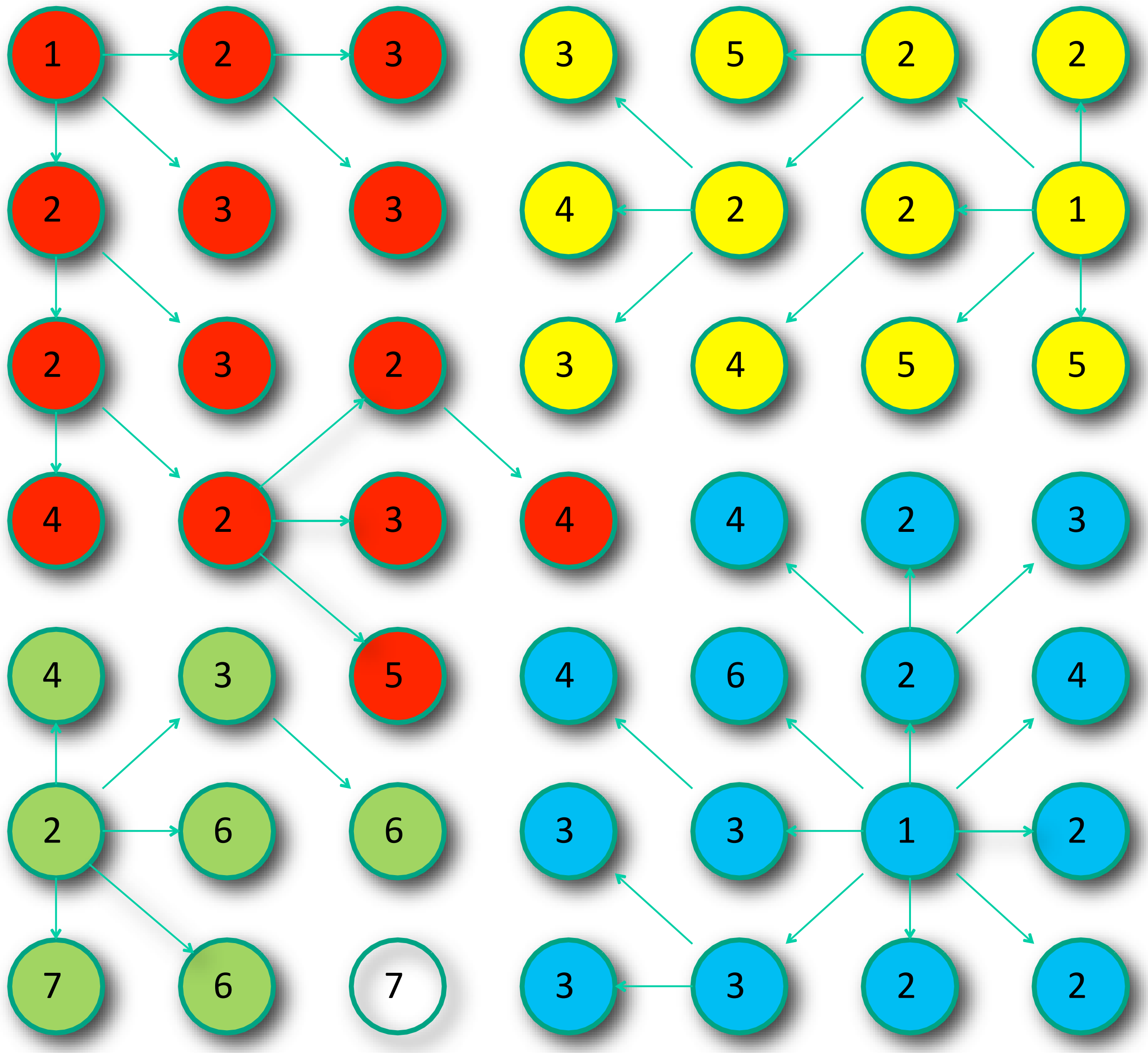
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW



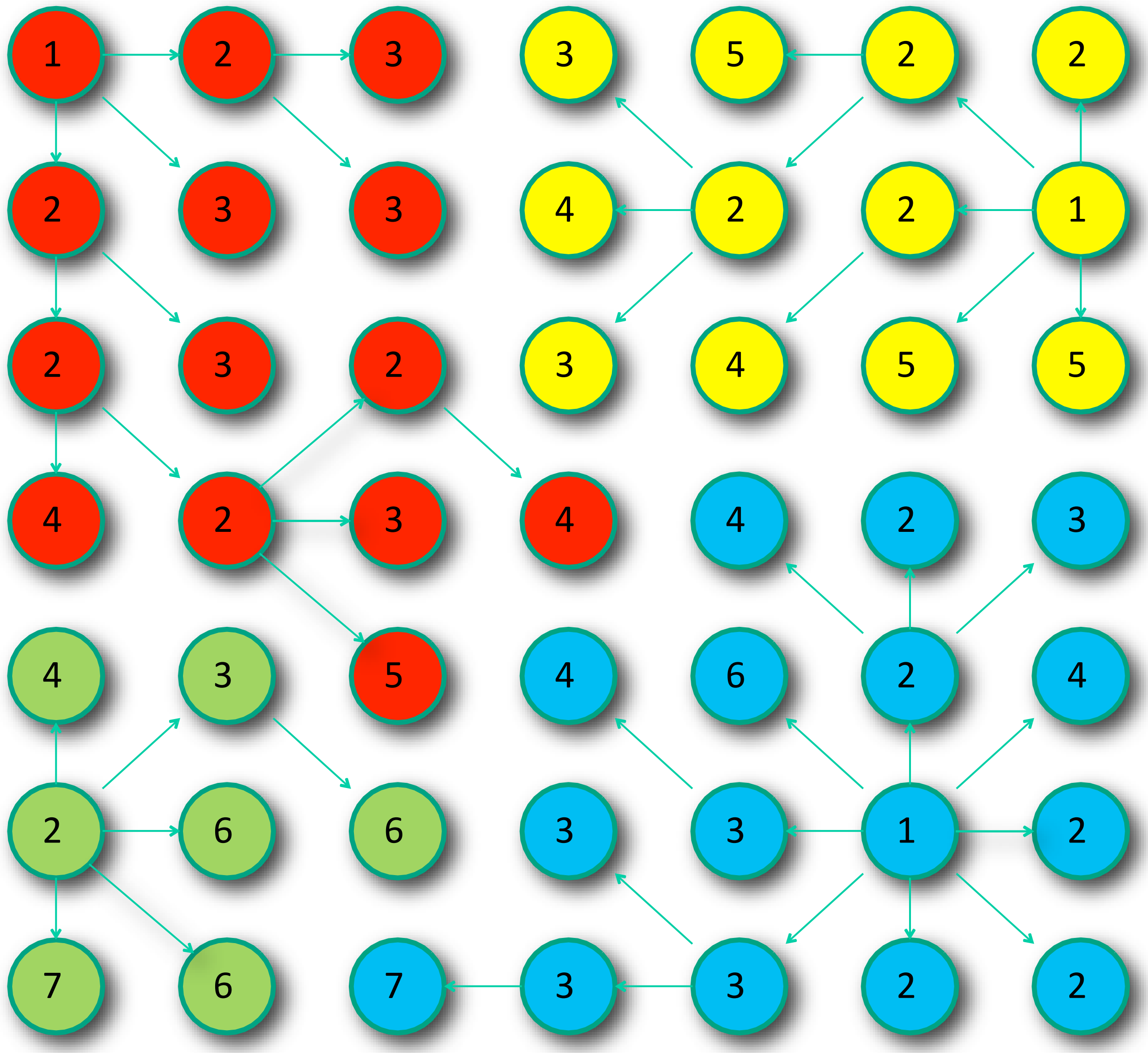
Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

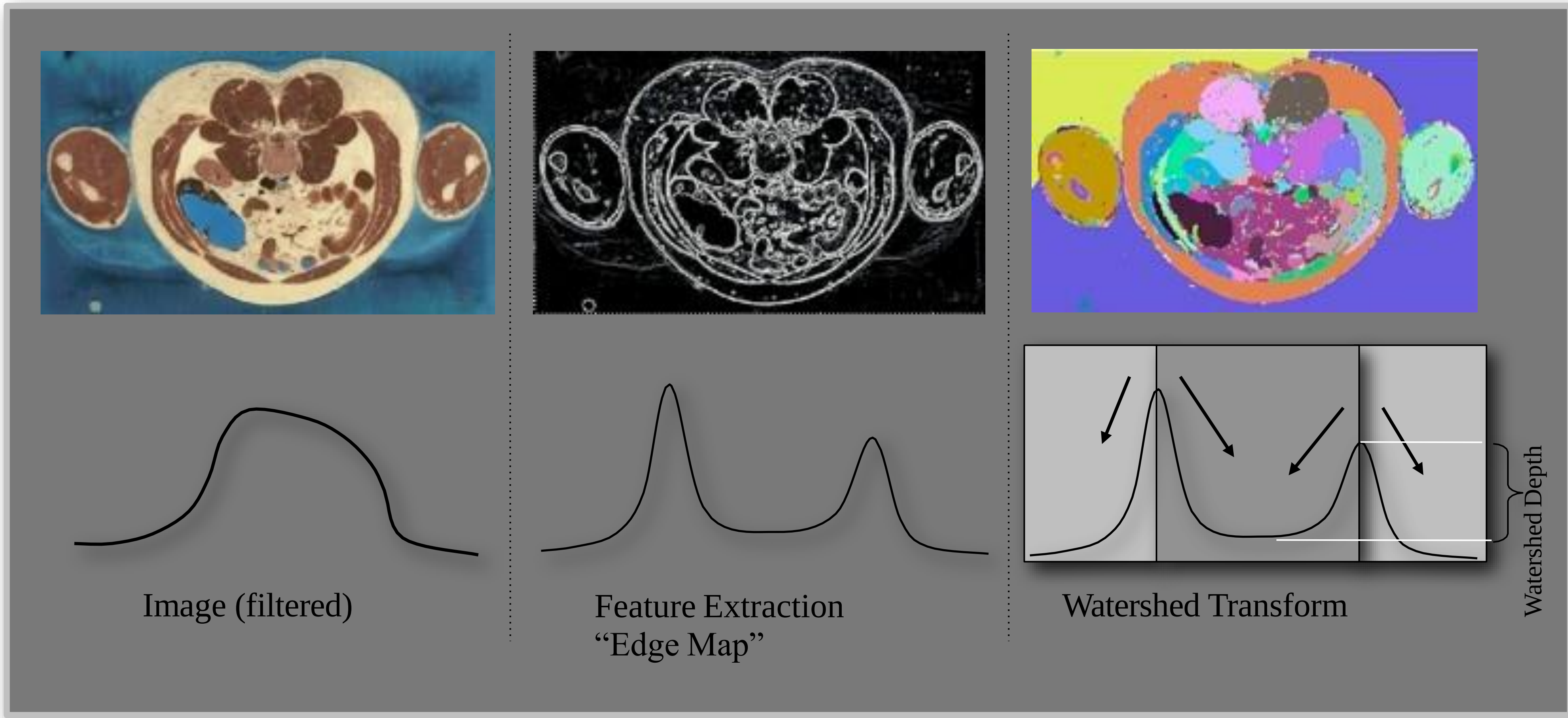
Direction Order: N,NE,E,SE, S,SW,W,NW

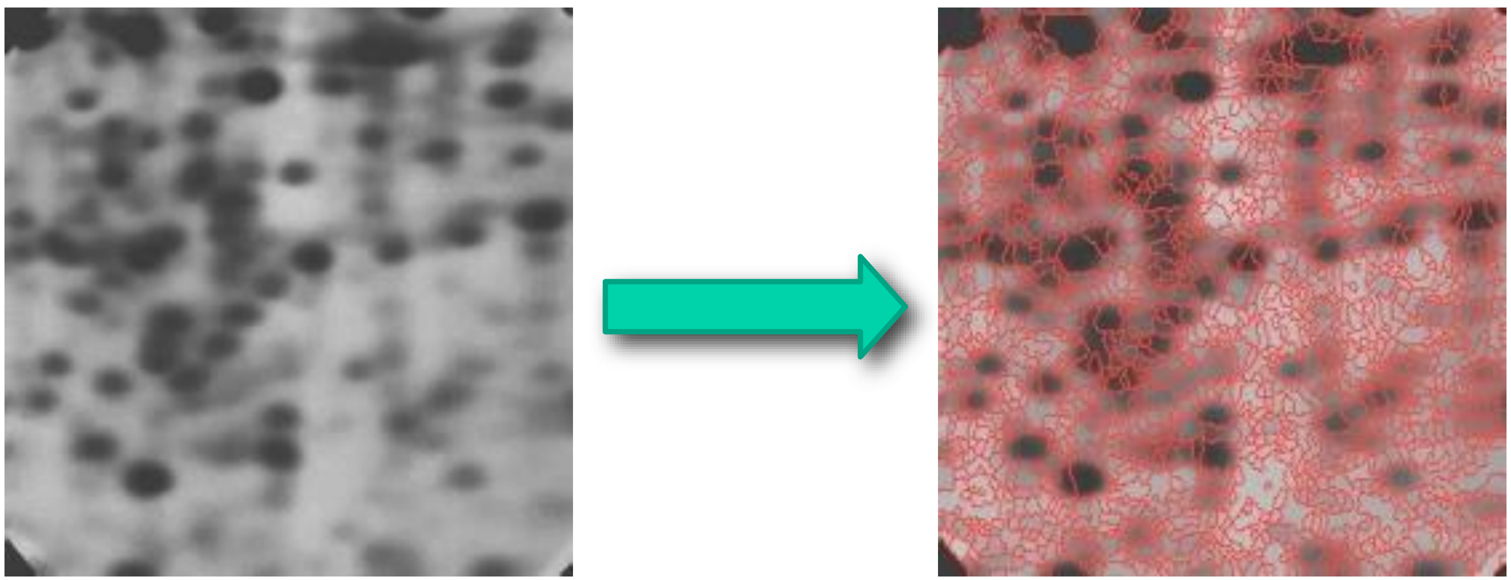


Watershed algorithm. Search simultaneously from seeds in order determined by gray level.

Direction Order: N,NE,E,SE, S,SW,W,NW

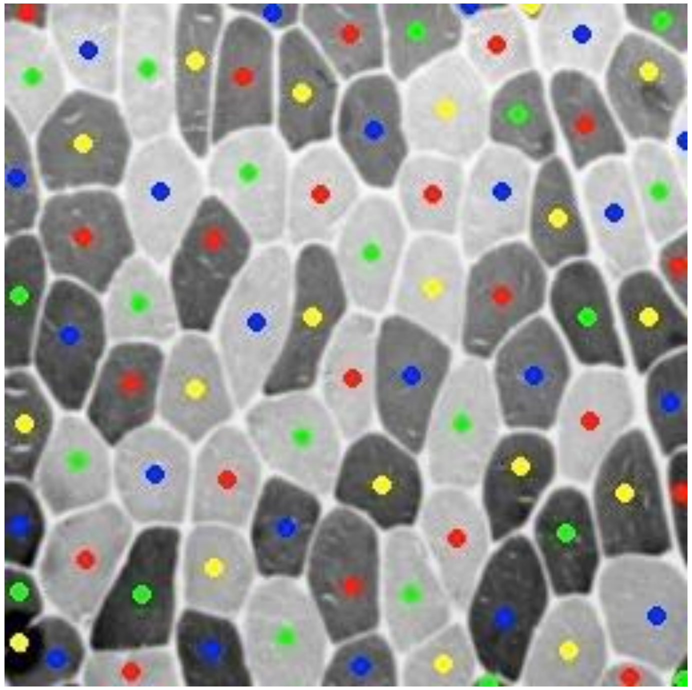




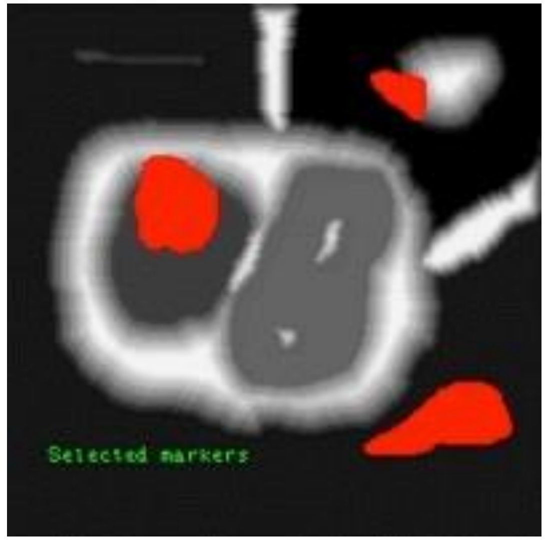


Problem: Watershed algorithm typically over–segments

Initialization



IDEA: Initialize regions using other information (cell nuclei) or by hand--marking. Pre-defines number of segments



Analysis (pros & cons)

Thresholding

- Relies on a single global threshold
- Produces separate regions (may prefer connected components)

Region growing

- Supervised, requires 1+ seed point
- Relies on a single threshold
- Produces one region

Watershed

- May be supervised or unsupervised
- No need for threshold parameter
- Partitions image into many regions
- To get good results best to specify number and positions of seeds

Overview

What is image segmentation?

Types of segmentation algorithms

1. Thresholding, region labelling and growing algorithms
   - (connected components, region growing, watershed)
2. Statistical Segmentation
   - (k-means, mean shift)
3. Graph based methods
   - (Merging algorithms, splitting algorithms, split/merge)
4. Edge based methods
   - (Intelligent Scissors, Snakes)

Overview

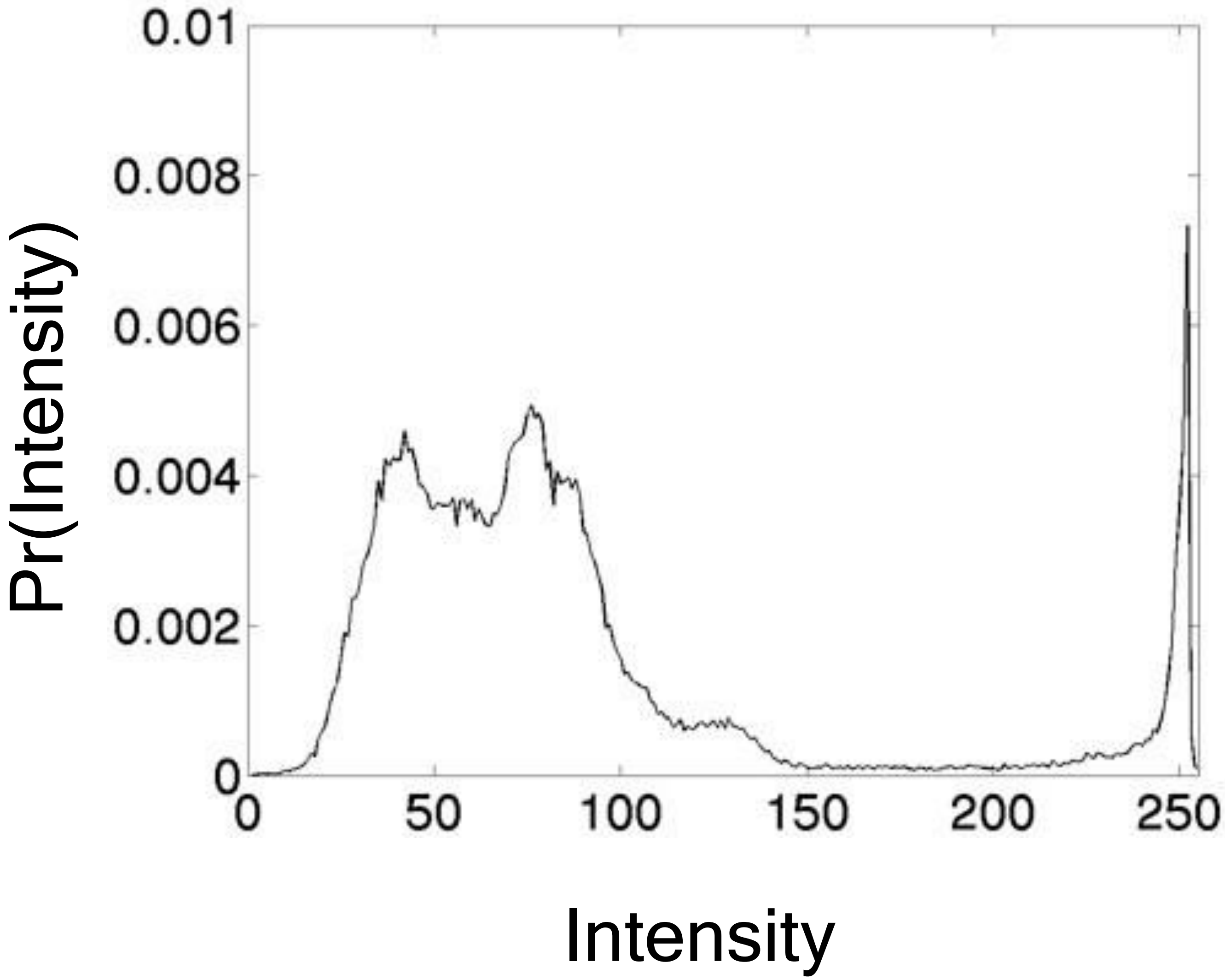
What is image segmentation?

Types of segmentation algorithms

1. Thresholding, region labelling and growing algorithms
   – (connected components, region growing, watershed)
2. Statistical Segmentation
   – (k-means, mean shift)
3. Graph based methods
   – (Merging algorithms, splitting algorithms, split/merge)
4. Edge based methods
   – (Intelligent Scissors, Snakes)

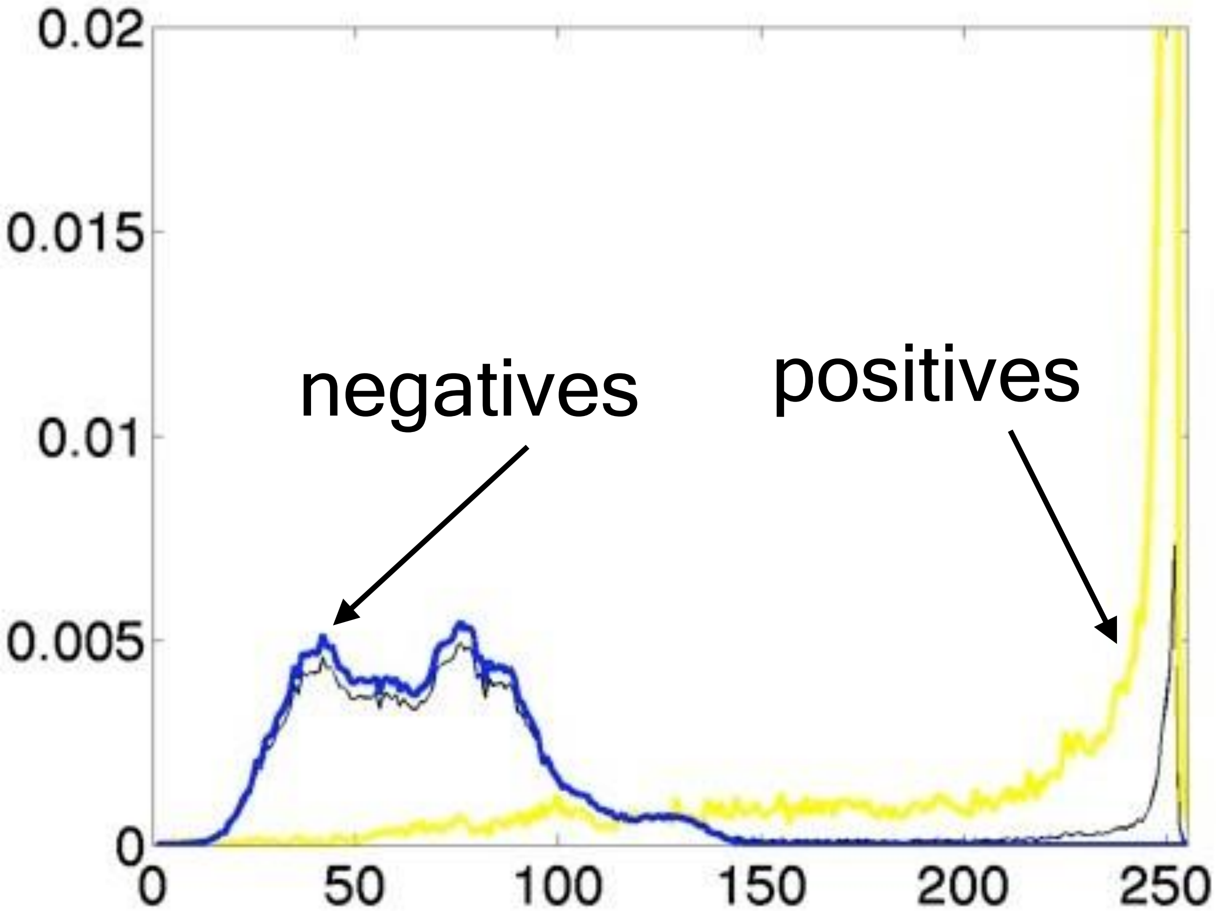
Choosing Threshold Automatically

Q. For binary thresholding, how can we choose the threshold automatically?



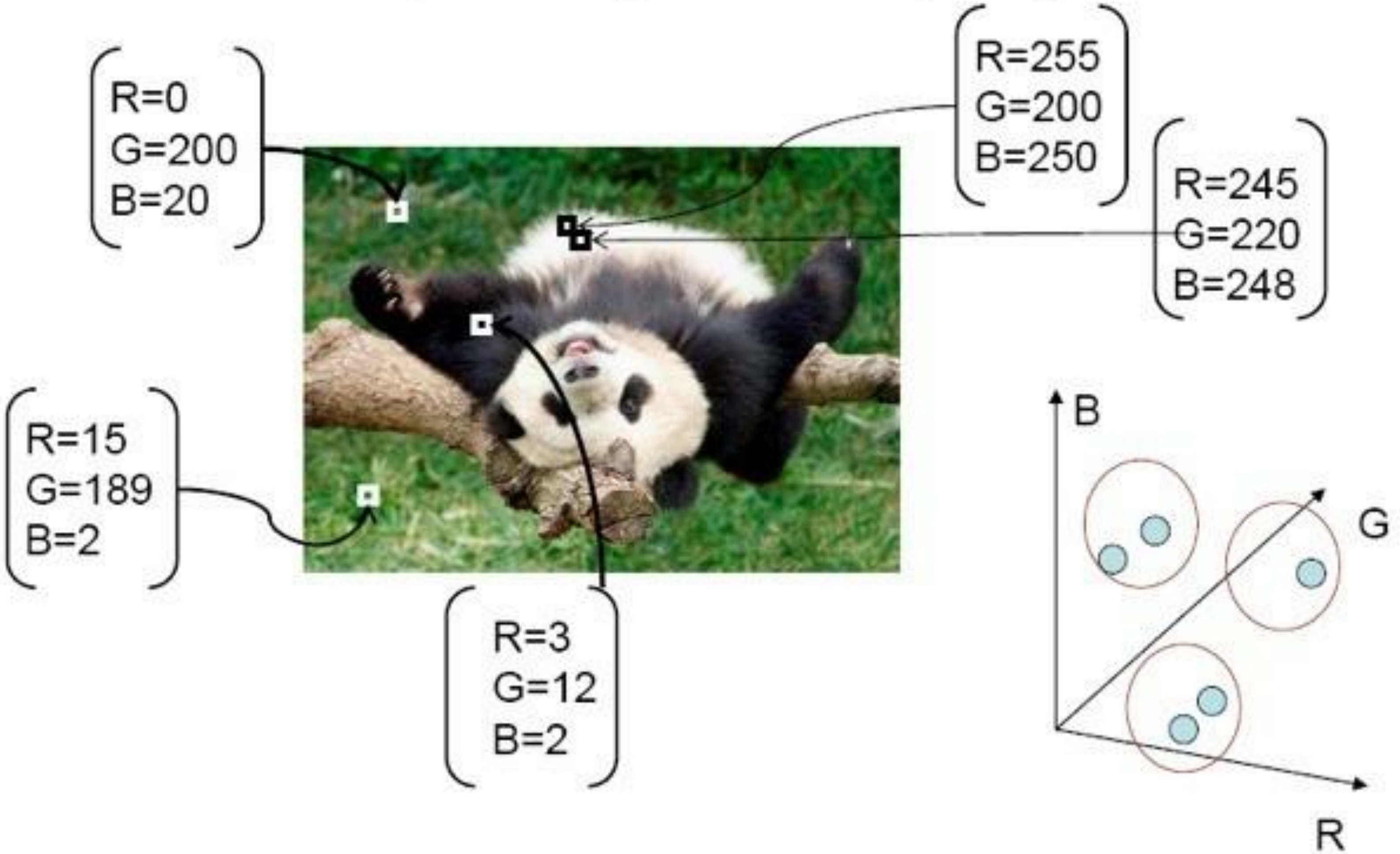
Choosing Threshold Automatically

Q. For binary thresholding, how can we choose the threshold automatically?



Segmentation as clustering

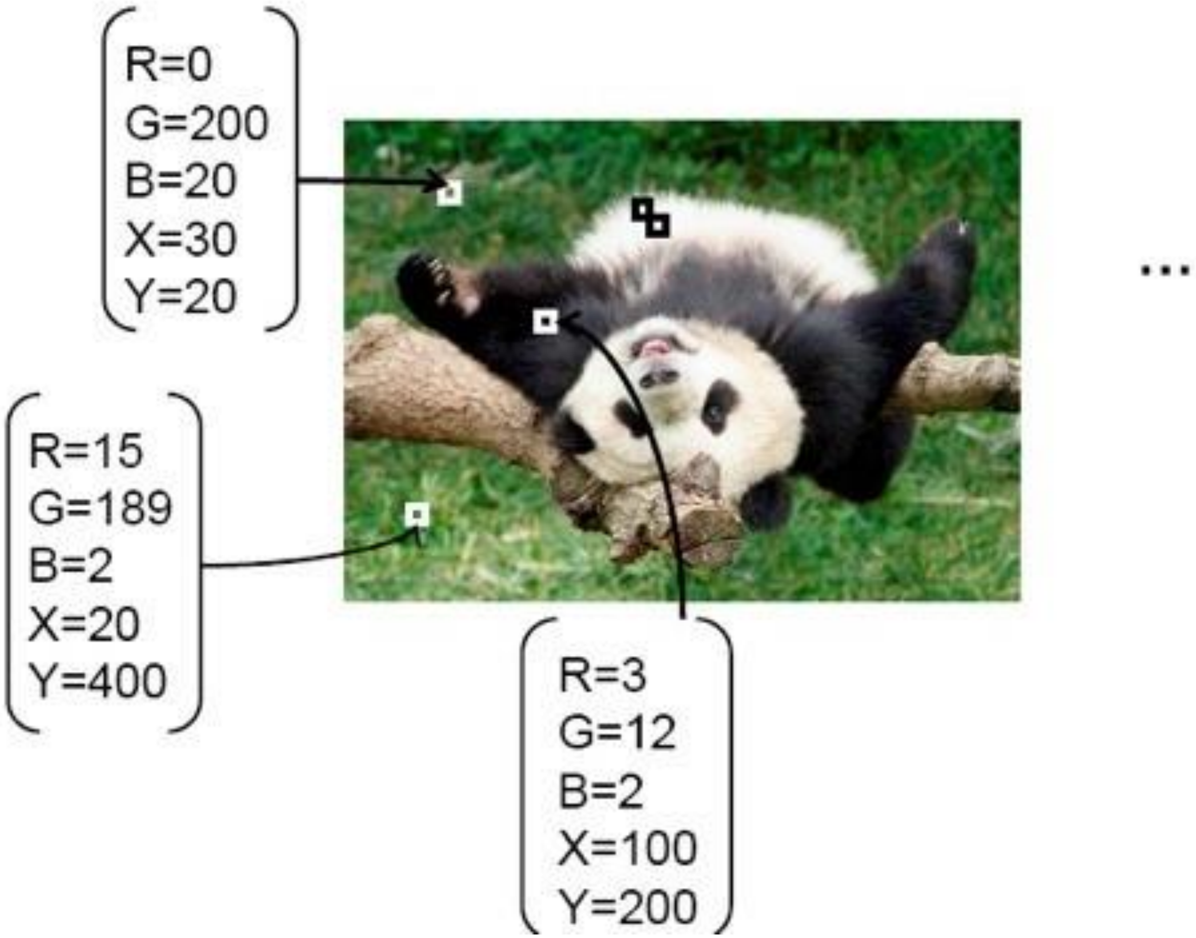
- Cluster similar pixels (features) together



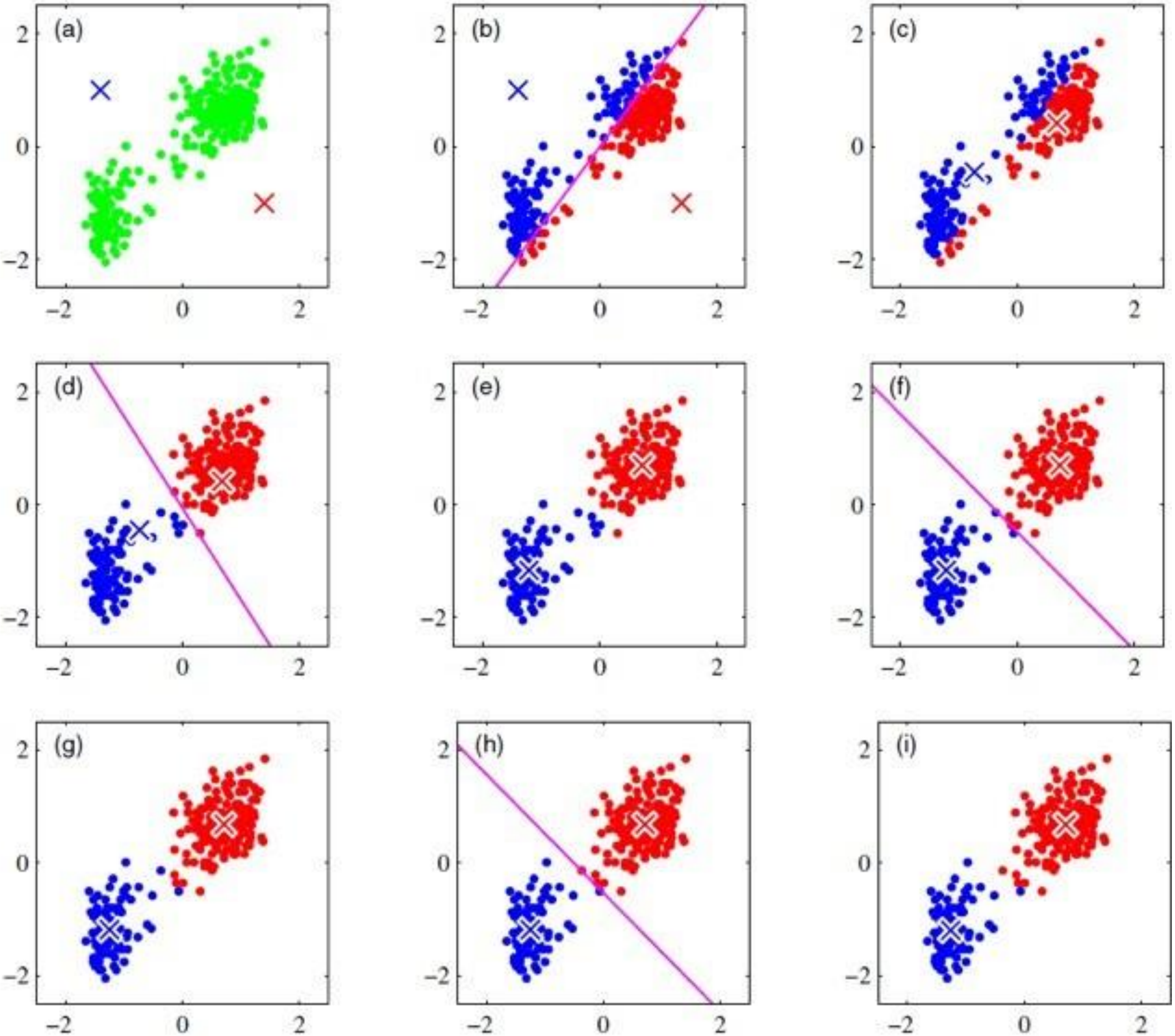
Source: K. Grauman

Segmentation as clustering

- Cluster similar pixels (features) together



Source: K. Grauman



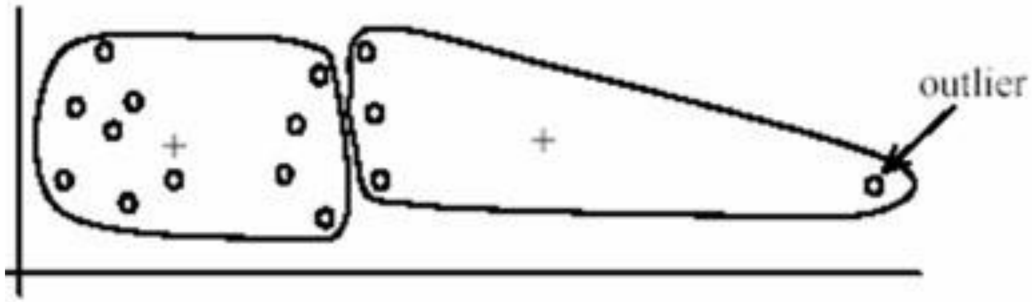
From Bishop (2006)

Figure 9.3 Two examples of the application of the K-means clustering algorithm to image segmentation showing the initial images together with their K-means segmentations obtained using various values of K. This also illustrates of the use of vector quantization for data compression, in which smaller values of K give higher compression at the expense of poorer image quality.

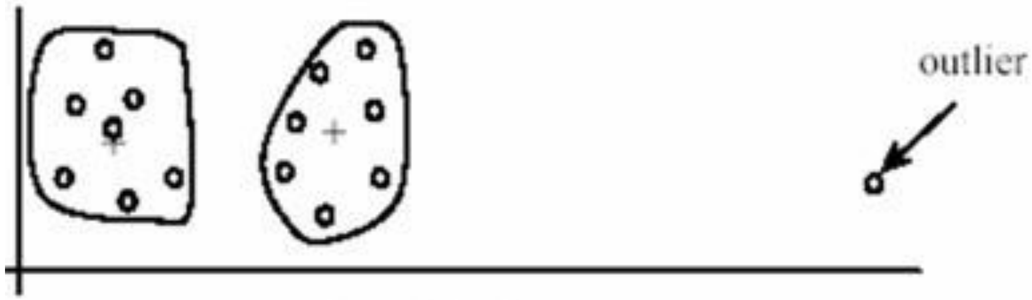
From Bishop (2006)

K–Means

- Pros
  - Simple and fast
  - Converges to a local minimum of the distance function
- Cons
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds “spherical” clusters



(A): Undesirable clusters



(B): Ideal clusters

Comparison

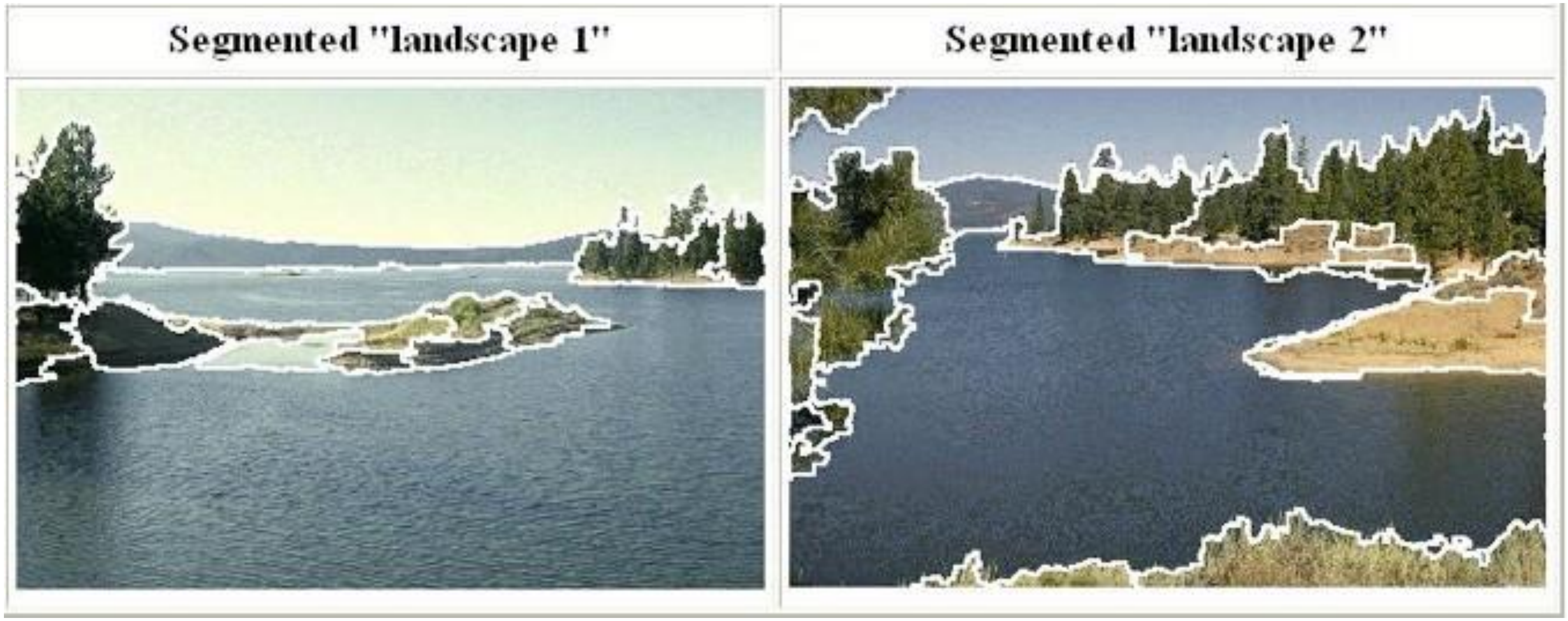
Watershed

- May be supervised or unsupervised
- No need for threshold parameter
- Partitions image into many regions
- To get good results best to specify number and positions of seeds

K–Means

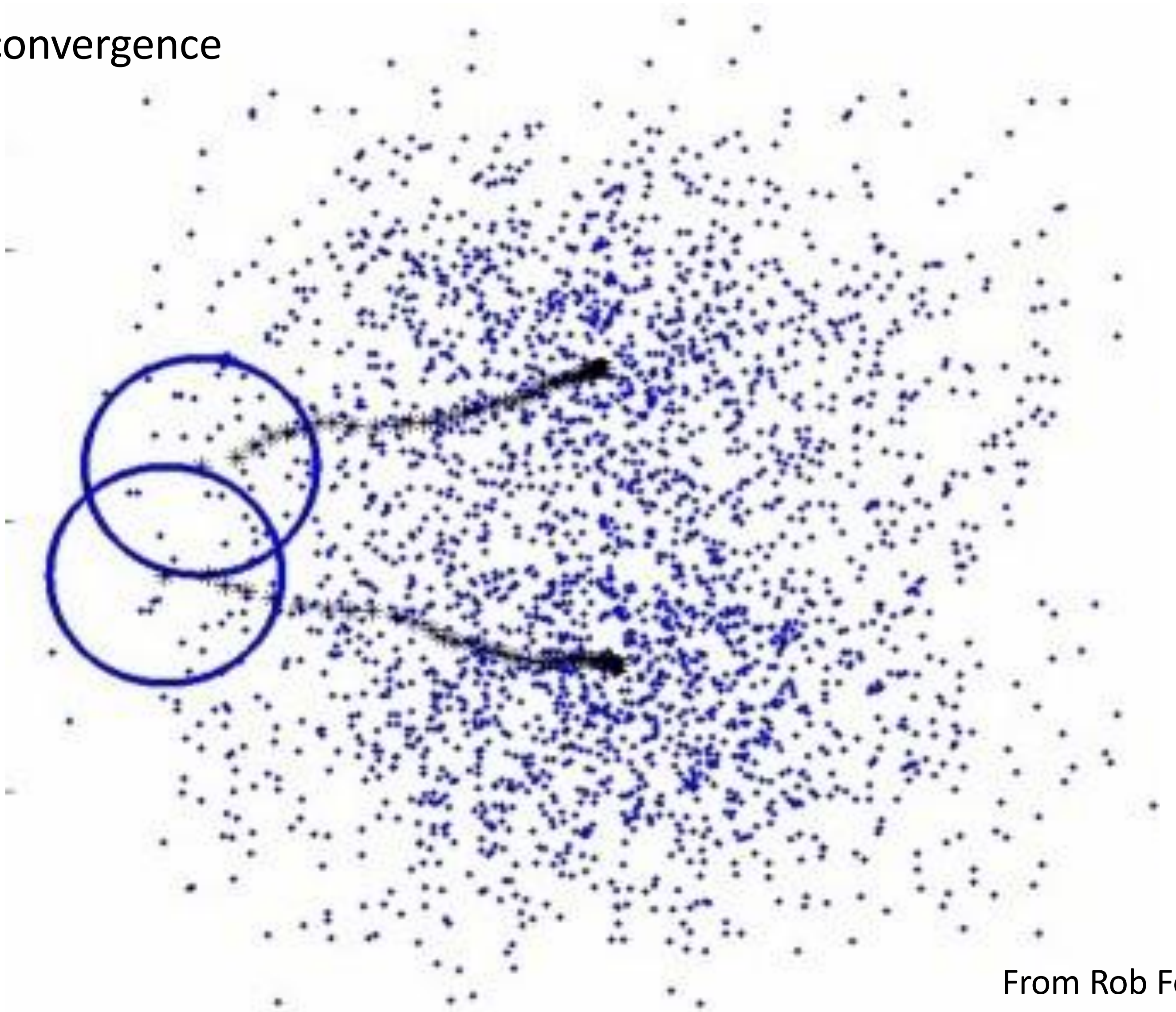
- Unsupervised
- Works in high--dimensions
- May produce disjoint segments
- Must specify number of clusters but not position

Mean shift Segmentation

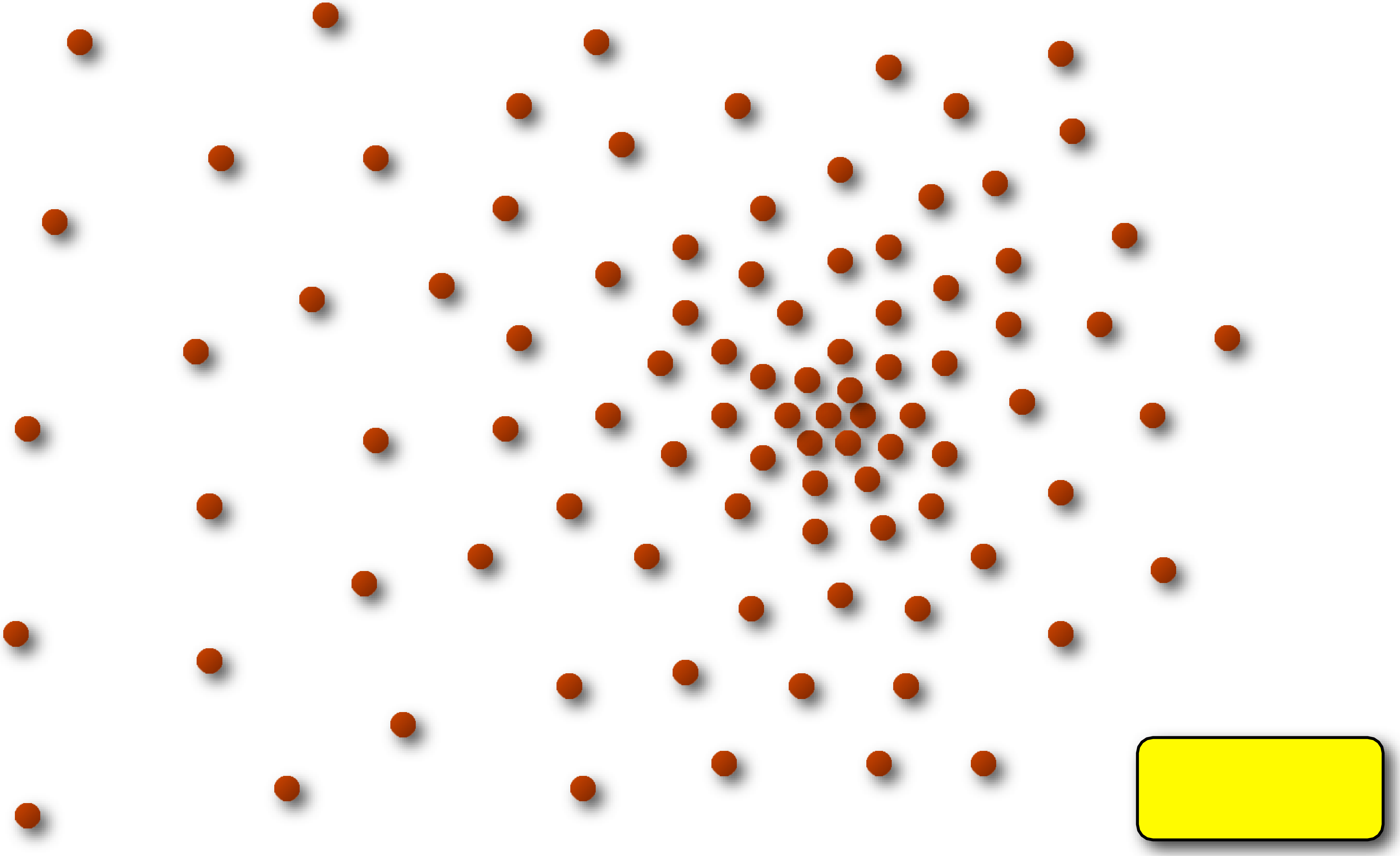


Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.
D. Comaniciu and P. Meer

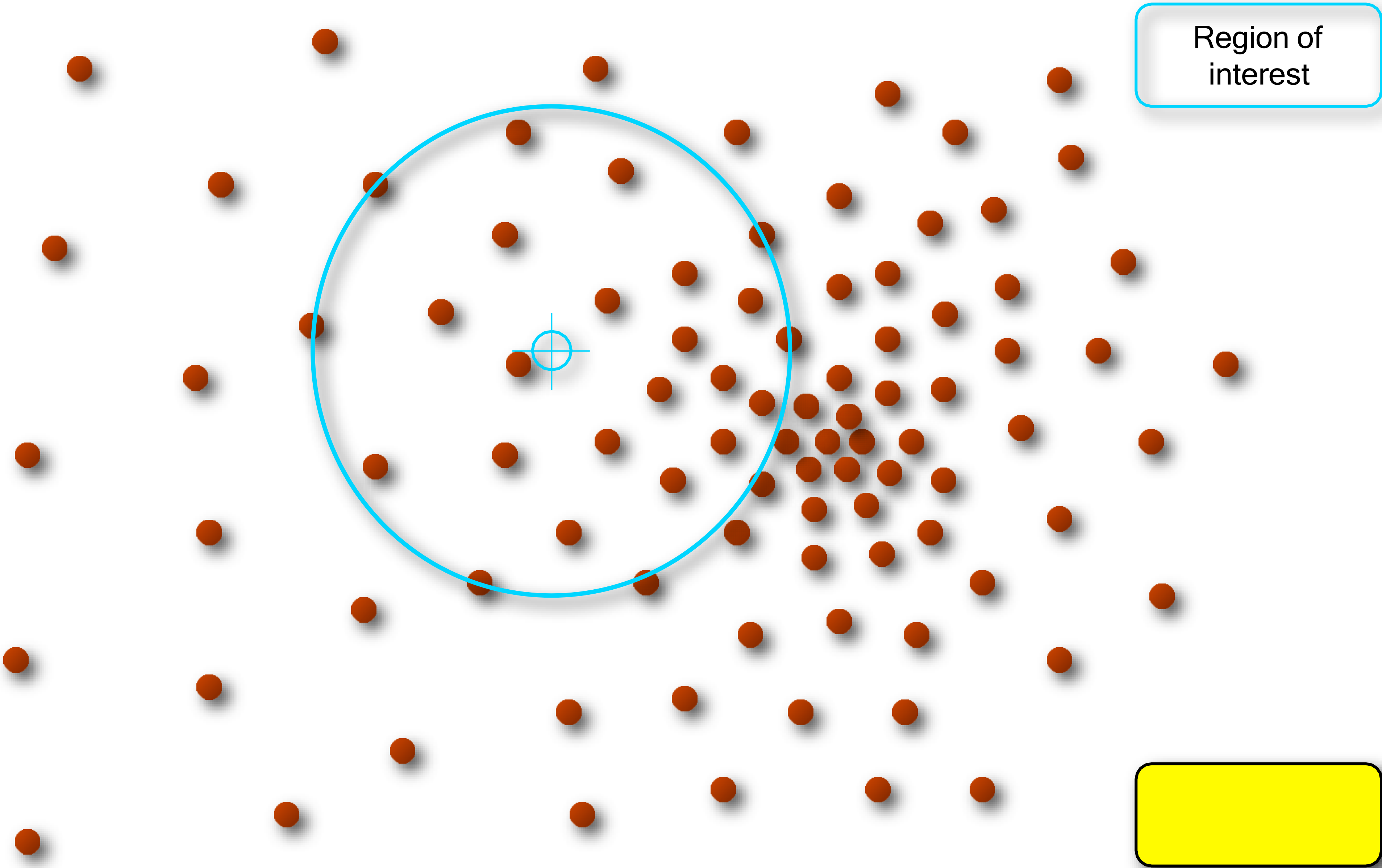
- The mean shift algorithm seeks a *mode* or local maximum of density of a given distribution
  - Choose a search window (width and location)
  - Compute the mean of the data in the search window
  - Center the search window at the new mean location
  - Repeat until convergence



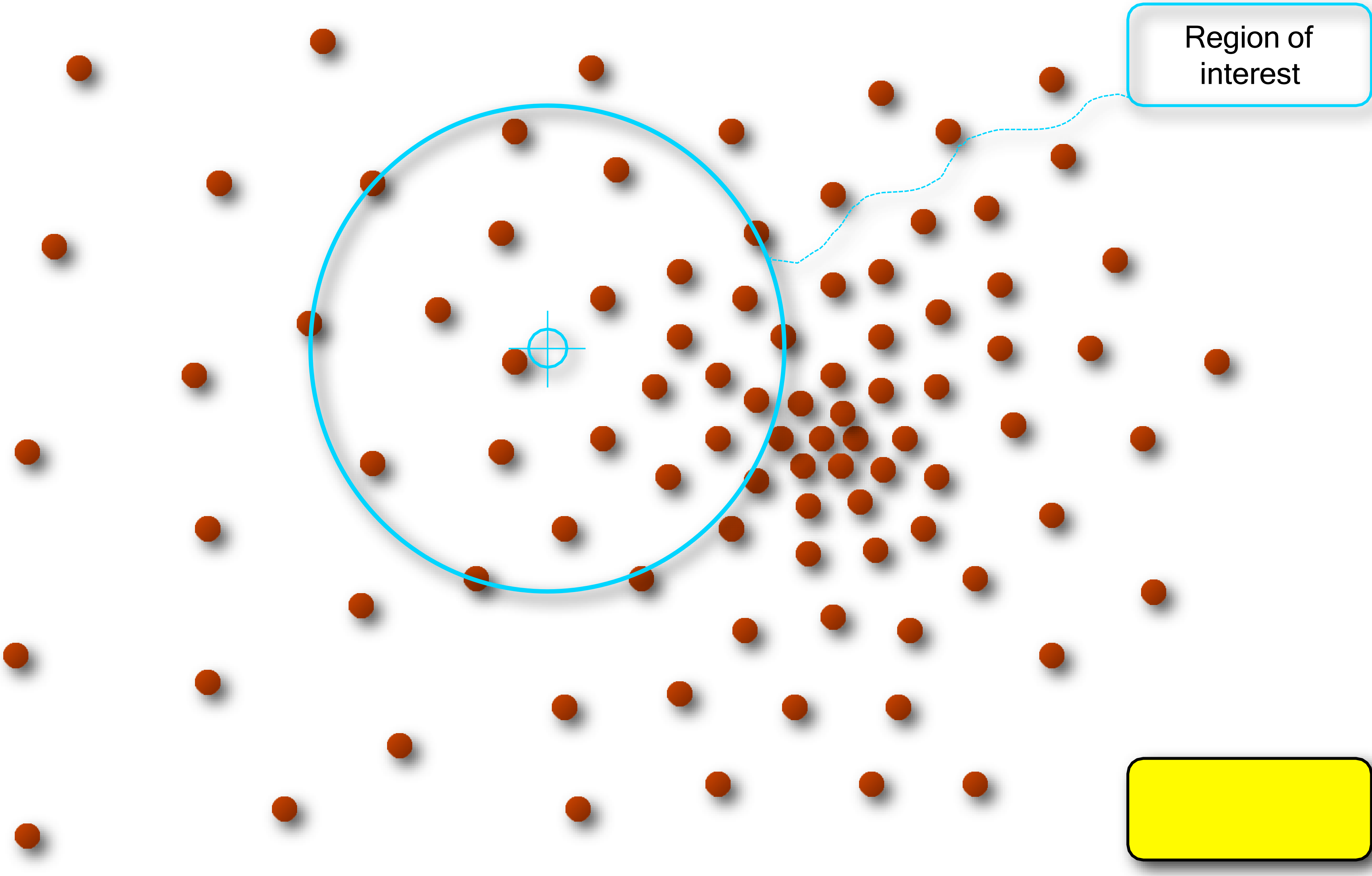
From Rob Fergus/ CV Online



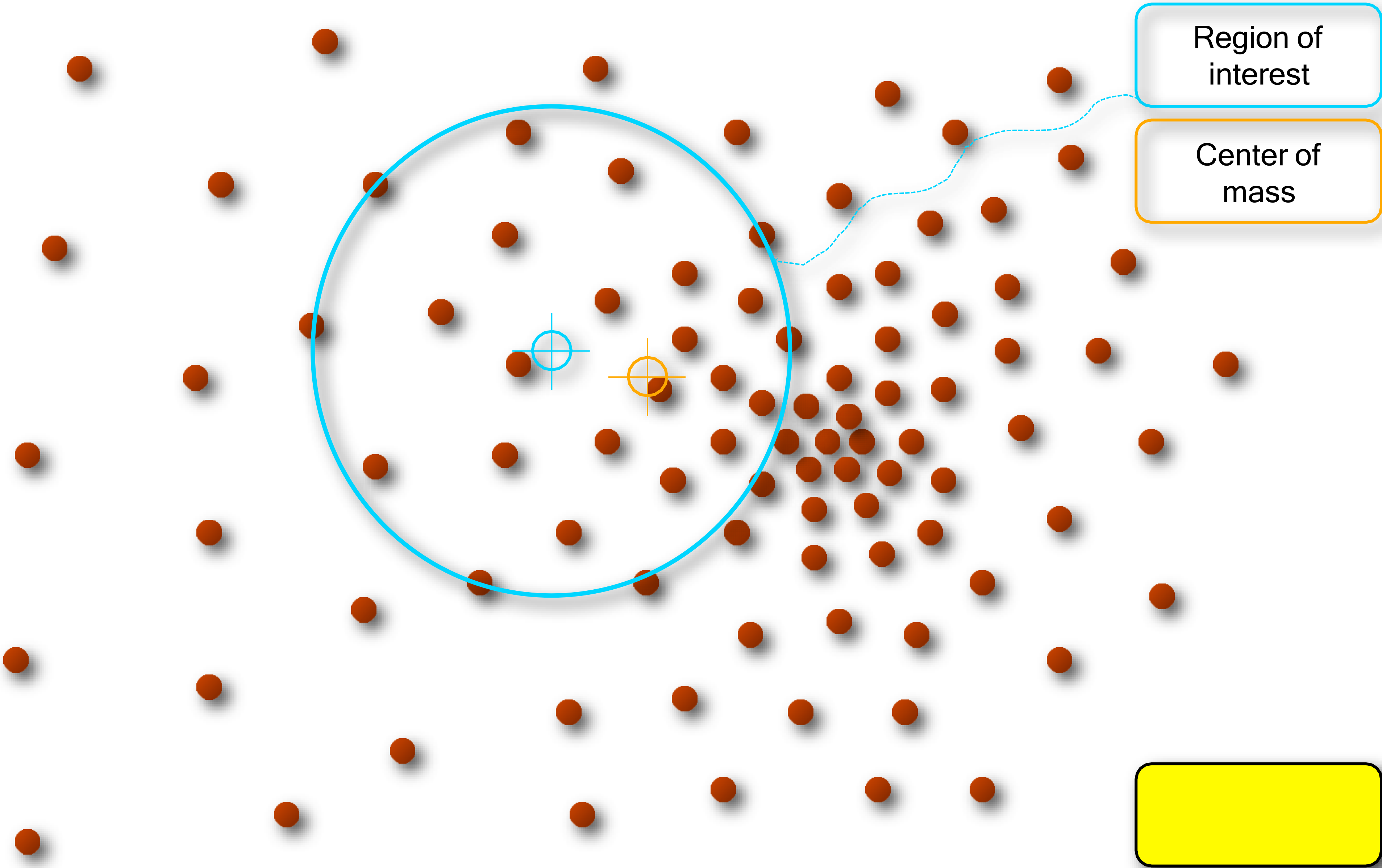
Slide by Y. Ukrainitz & B. Sarel



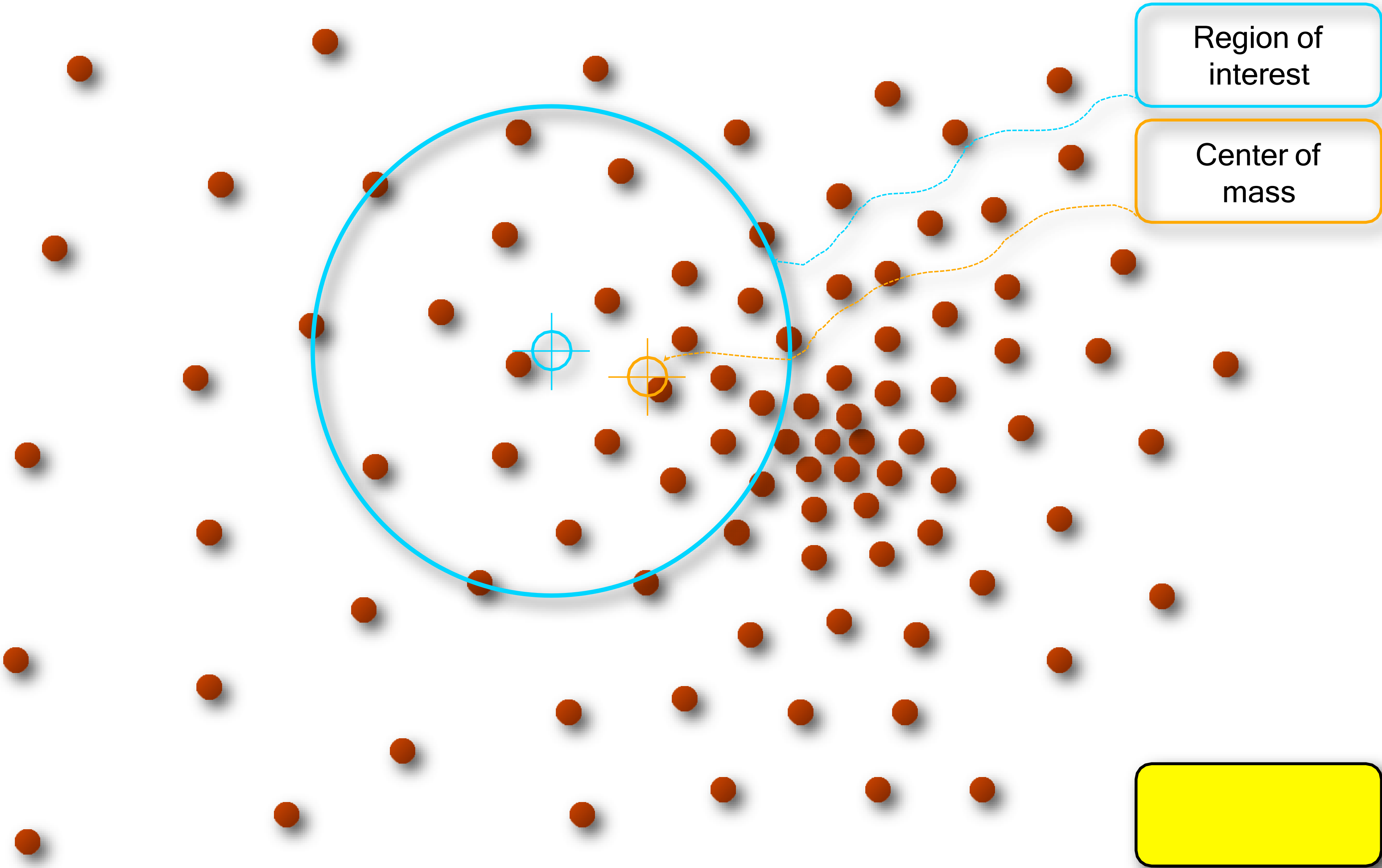
Slide by Y. Ukrainitz & B. Sarel



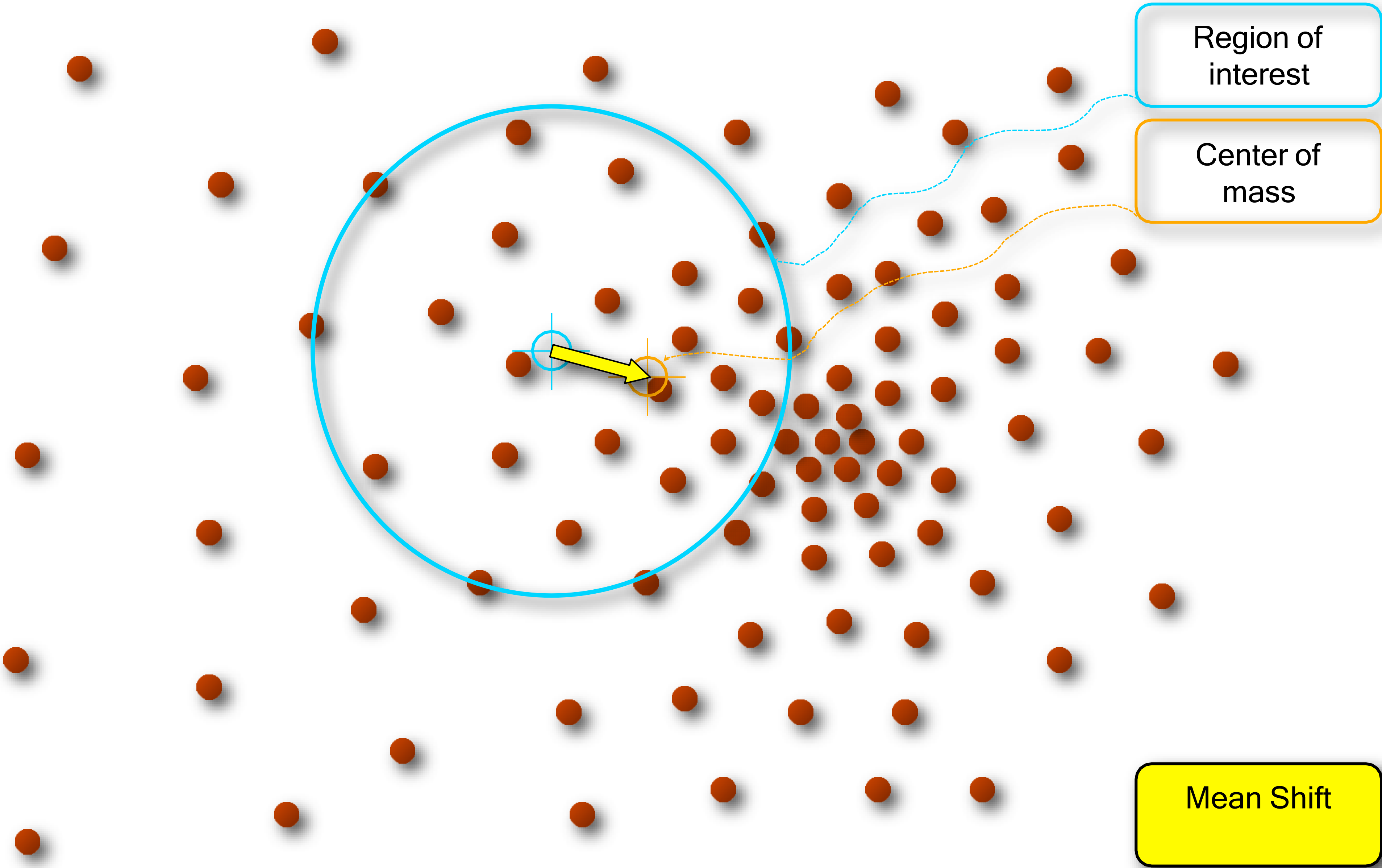
Slide by Y. Ukrainitz & B. Sarel



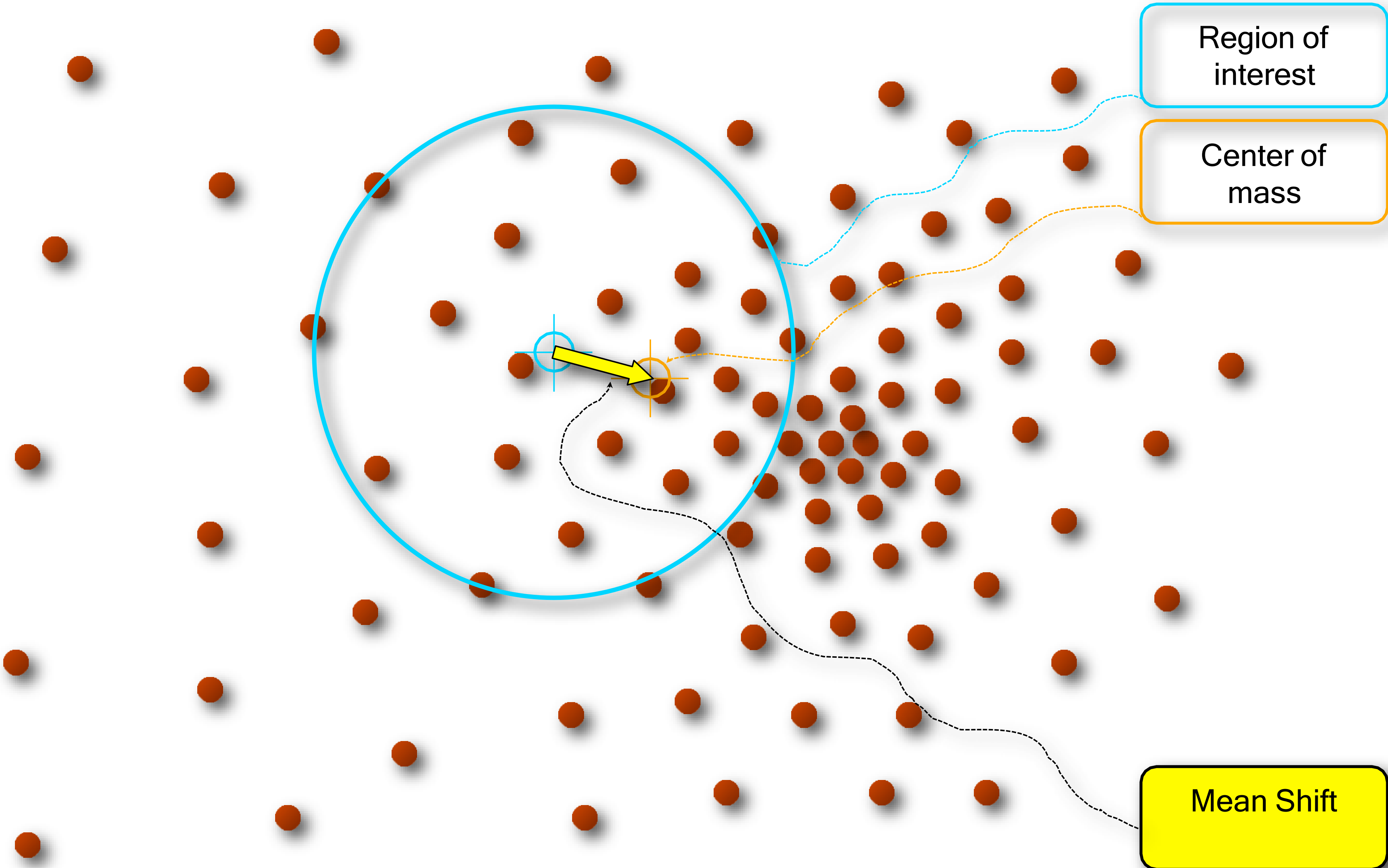
Slide by Y. Ukrainitz & B. Sarel



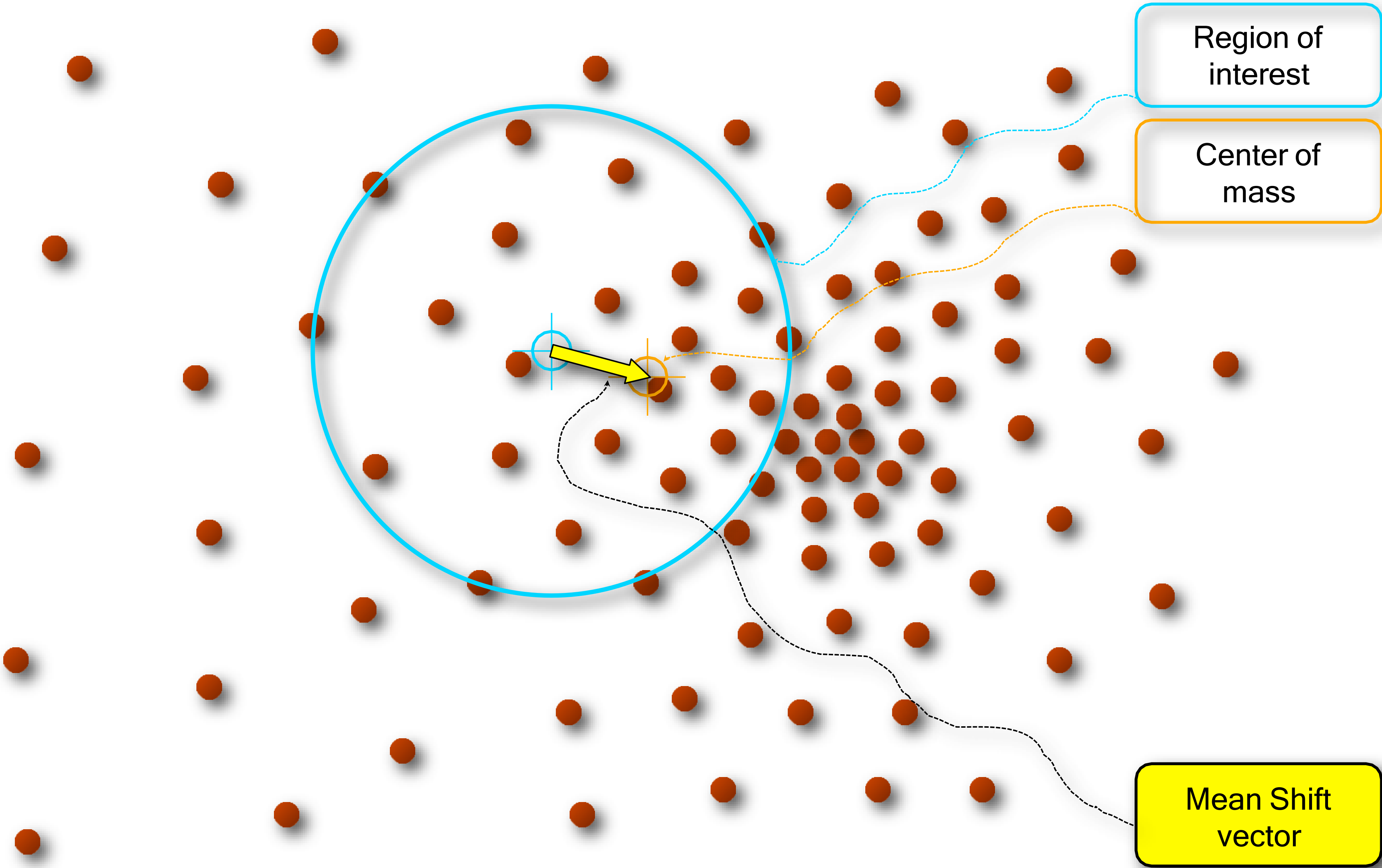
Slide by Y. Ukrainitz & B. Sarel



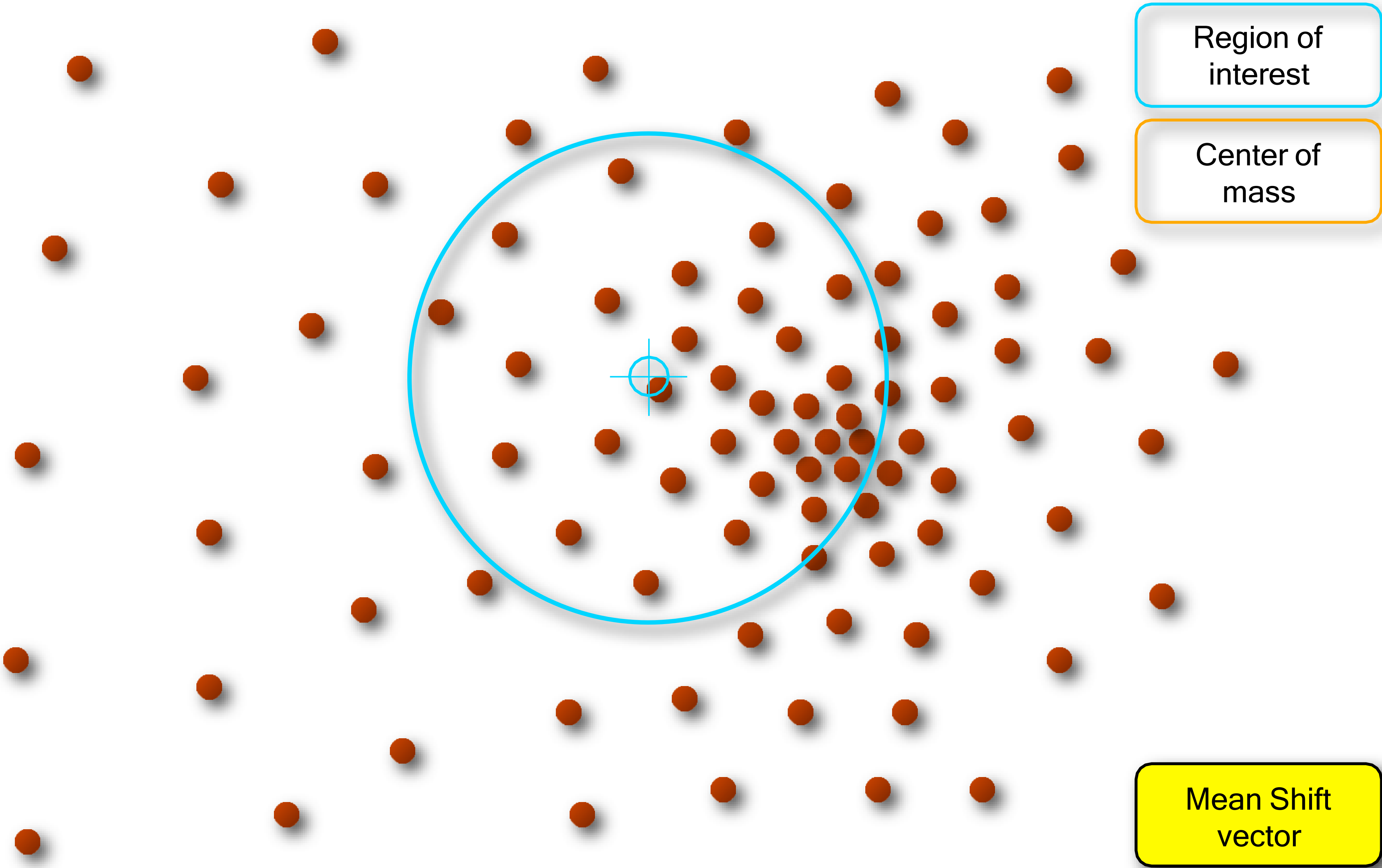
Slide by Y. Ukrainitz & B. Sarel



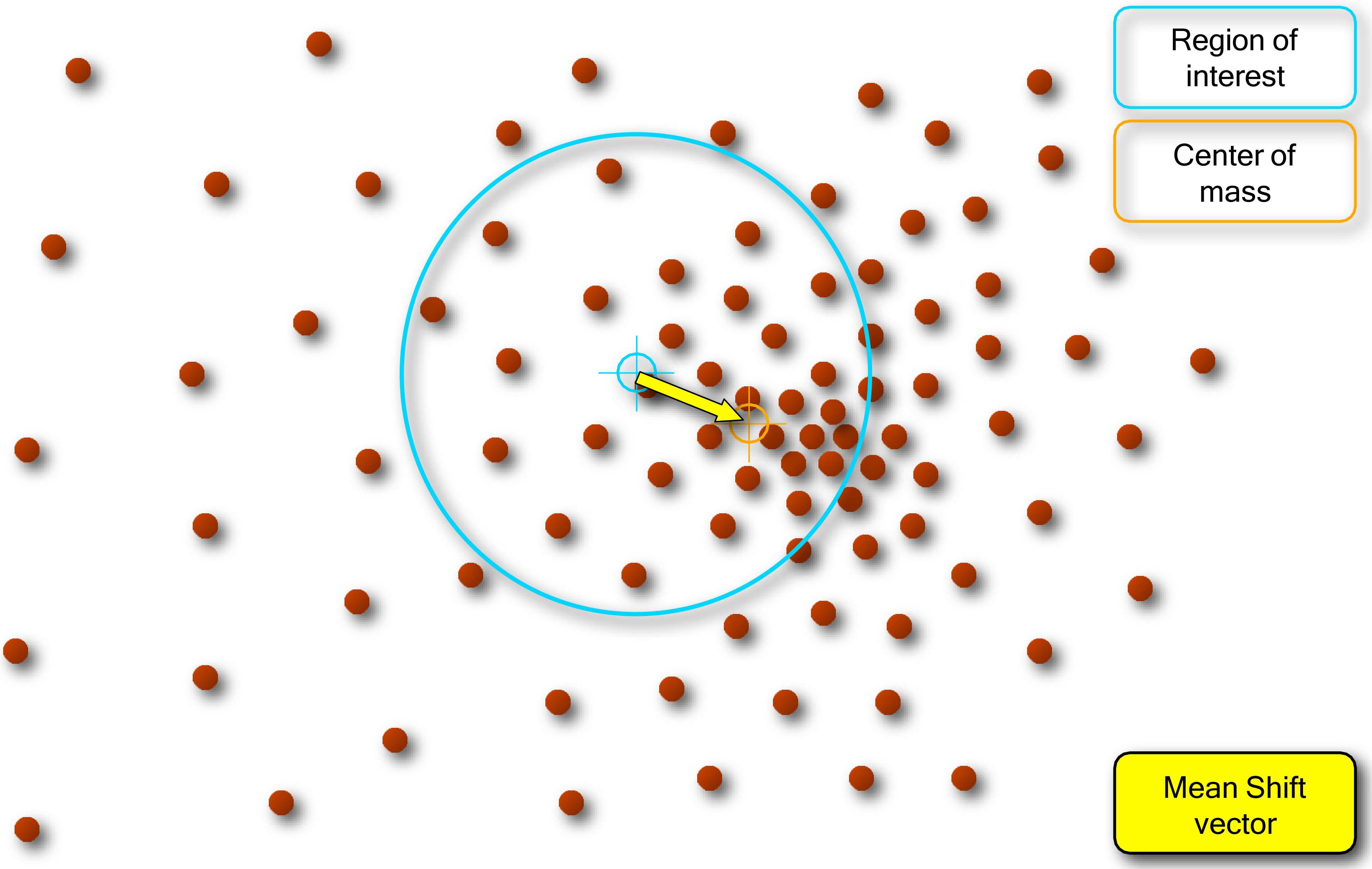
Slide by Y. Ukrainitz & B. Sarel



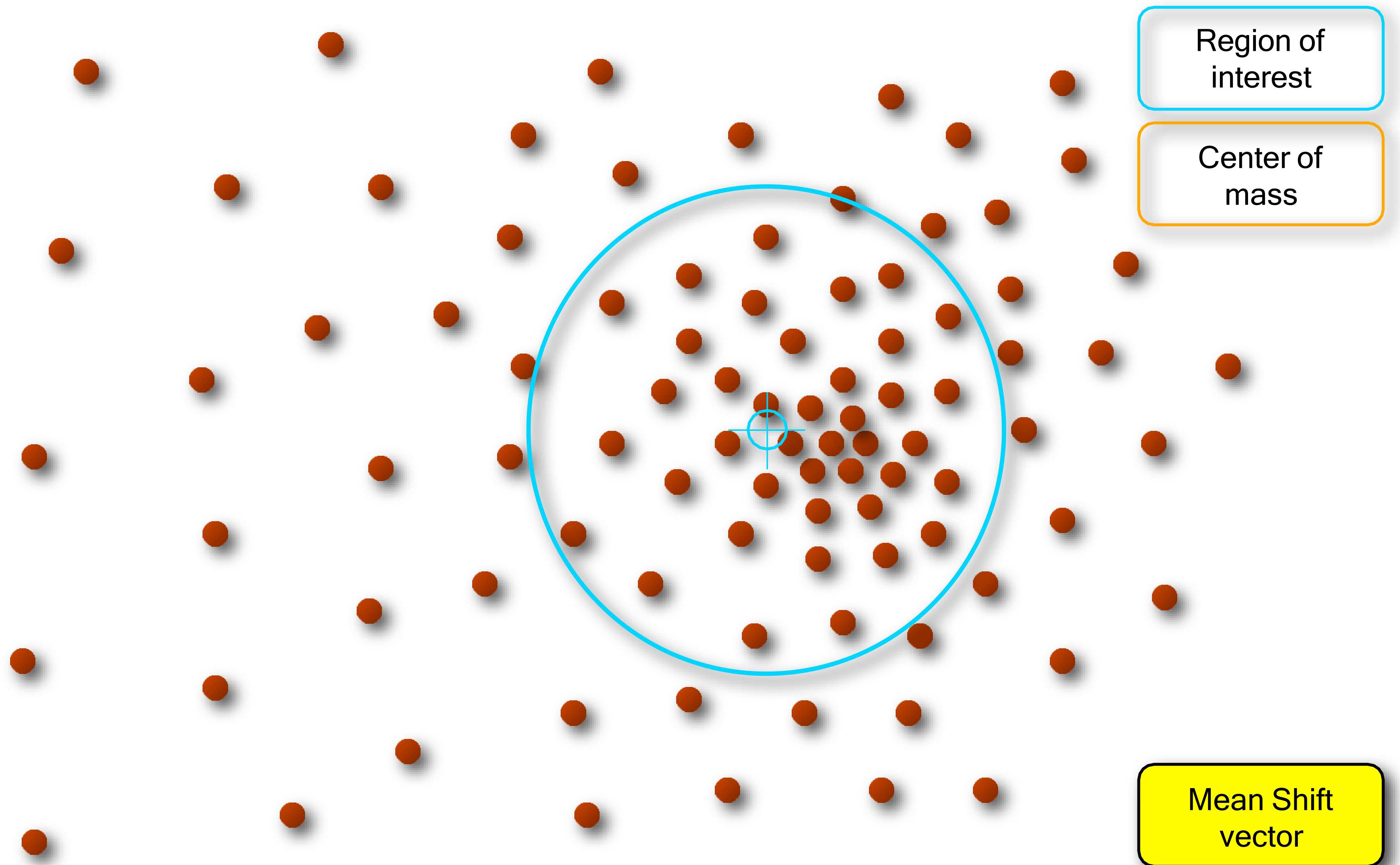
Slide by Y. Ukrainitz & B. Sarel



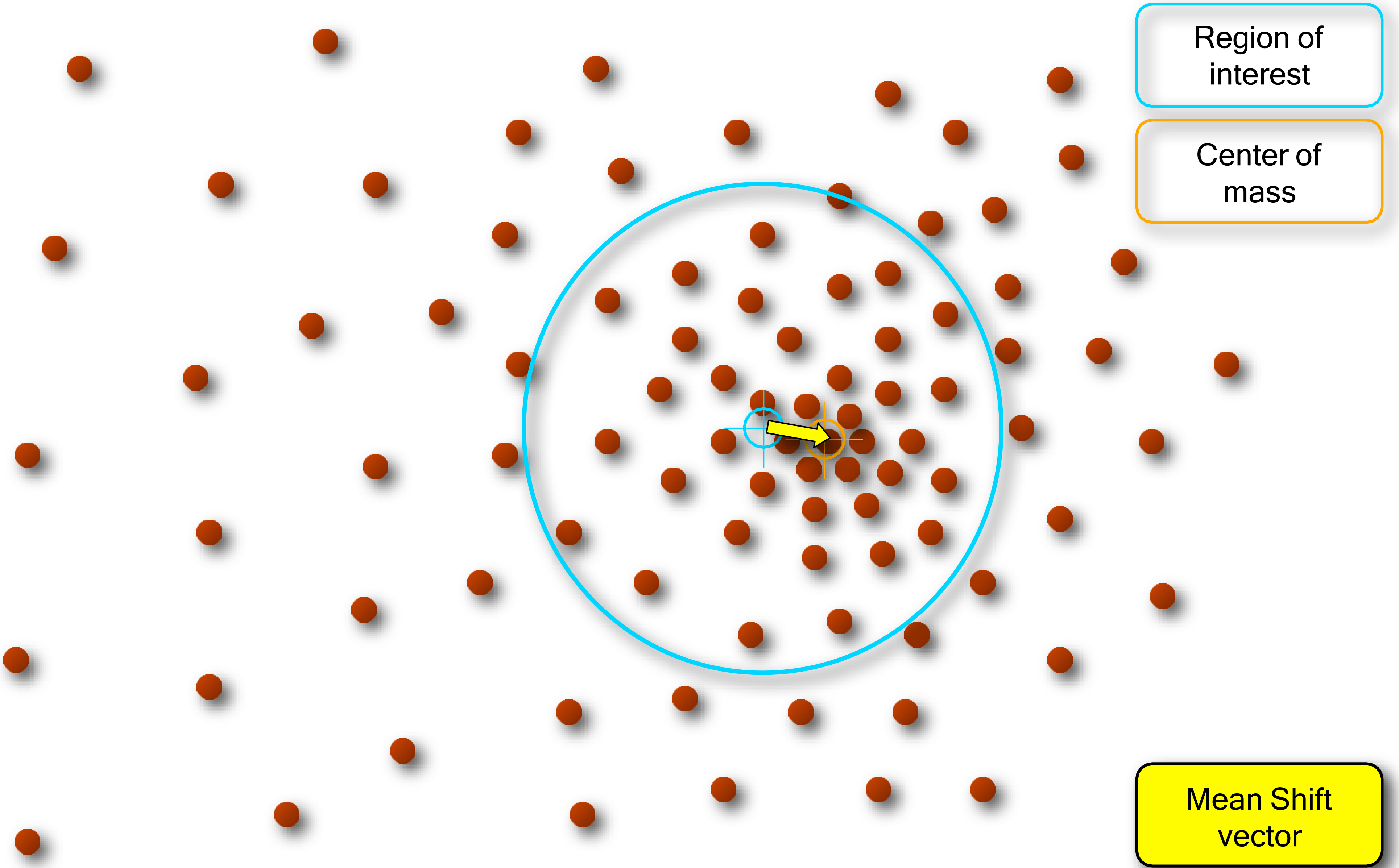
Slide by Y. Ukrainitz & B. Sarel



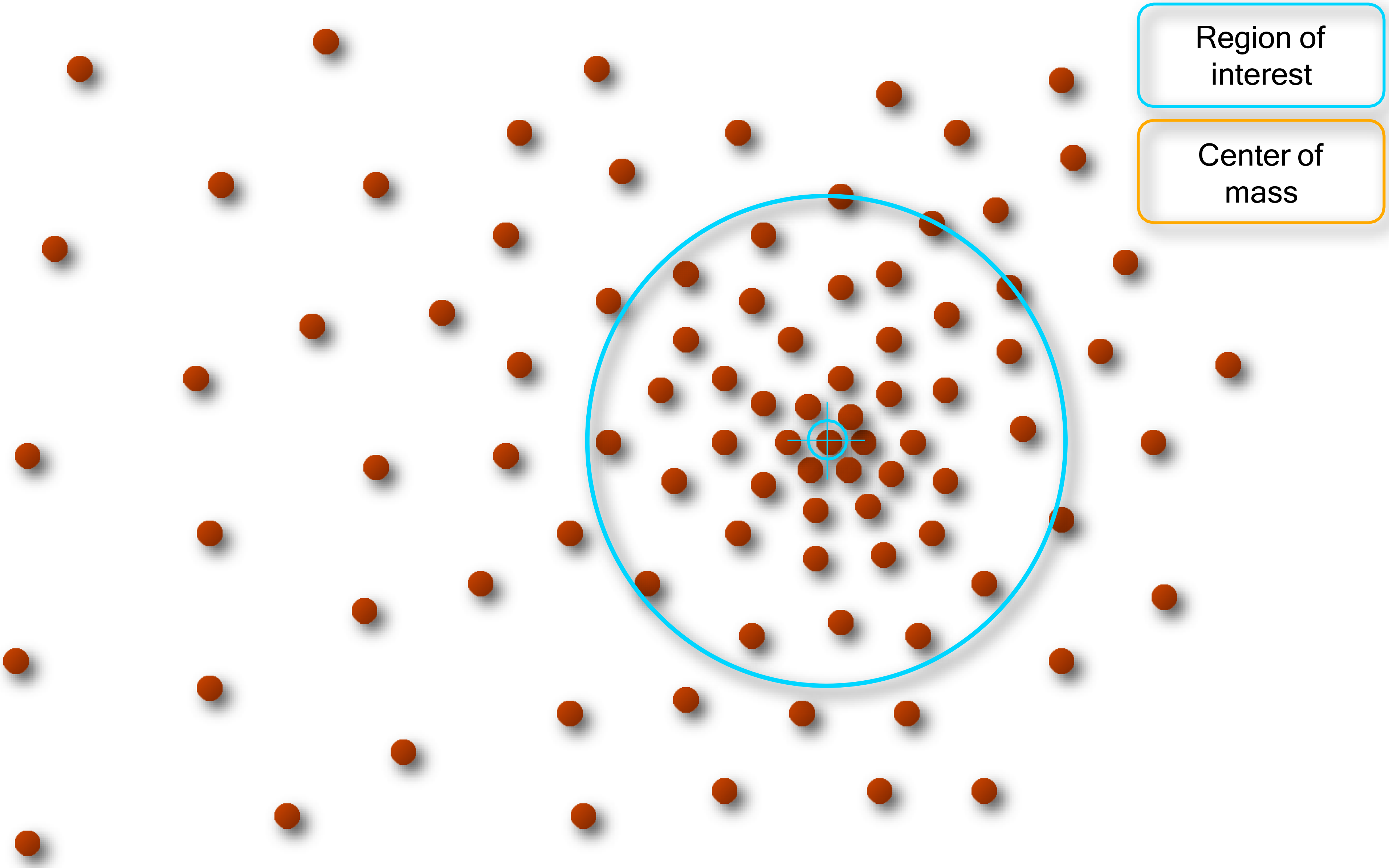
Slide by Y. Ukrainitz & B. Sarel



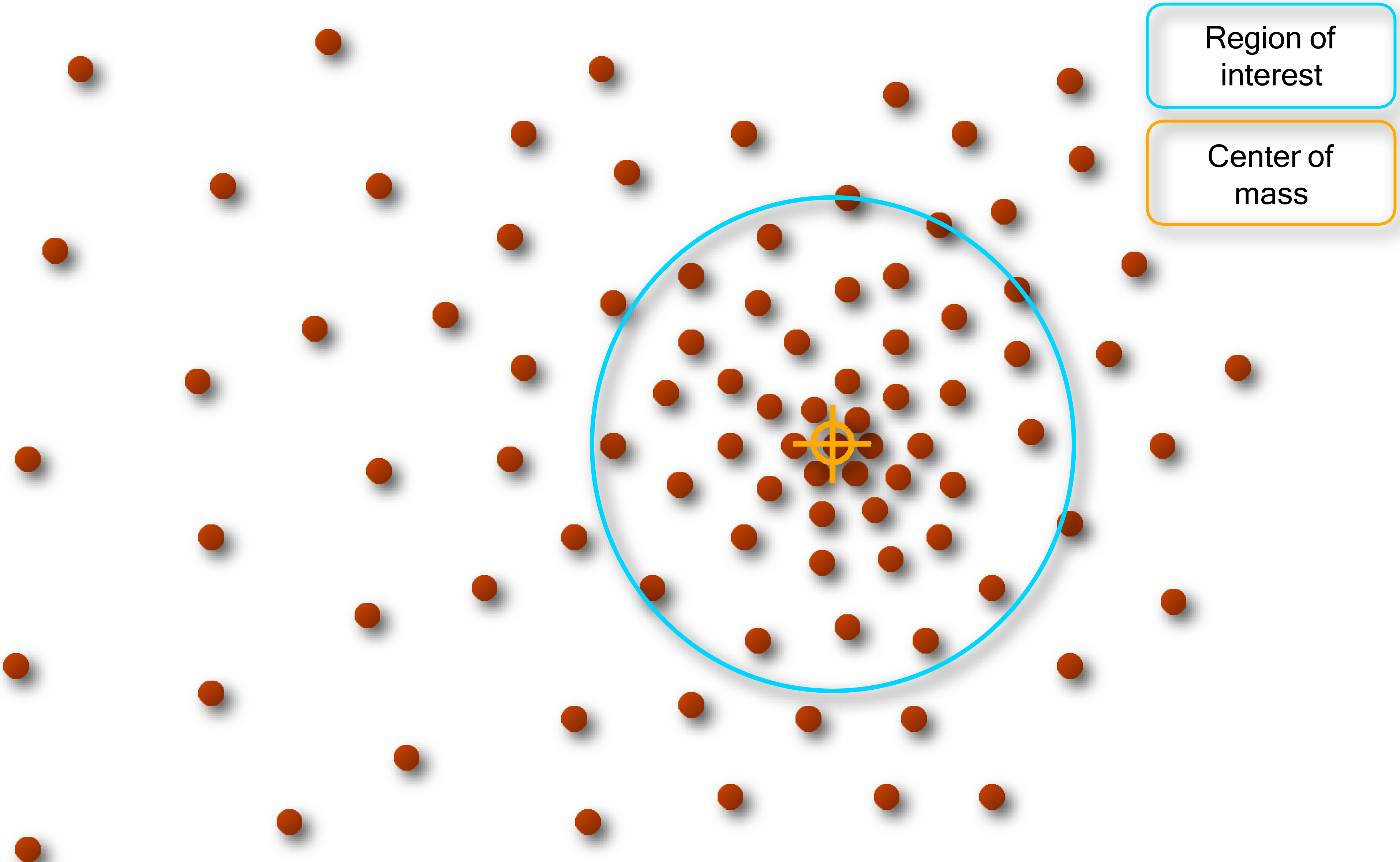
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel



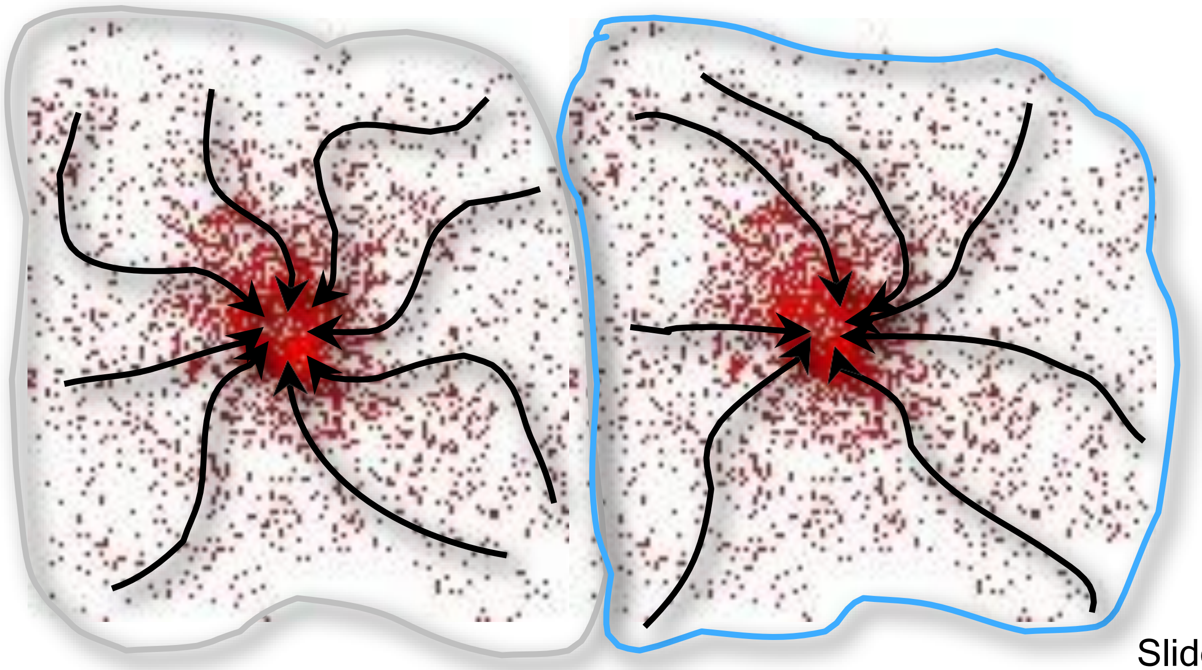
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel

Mean shift Clustering

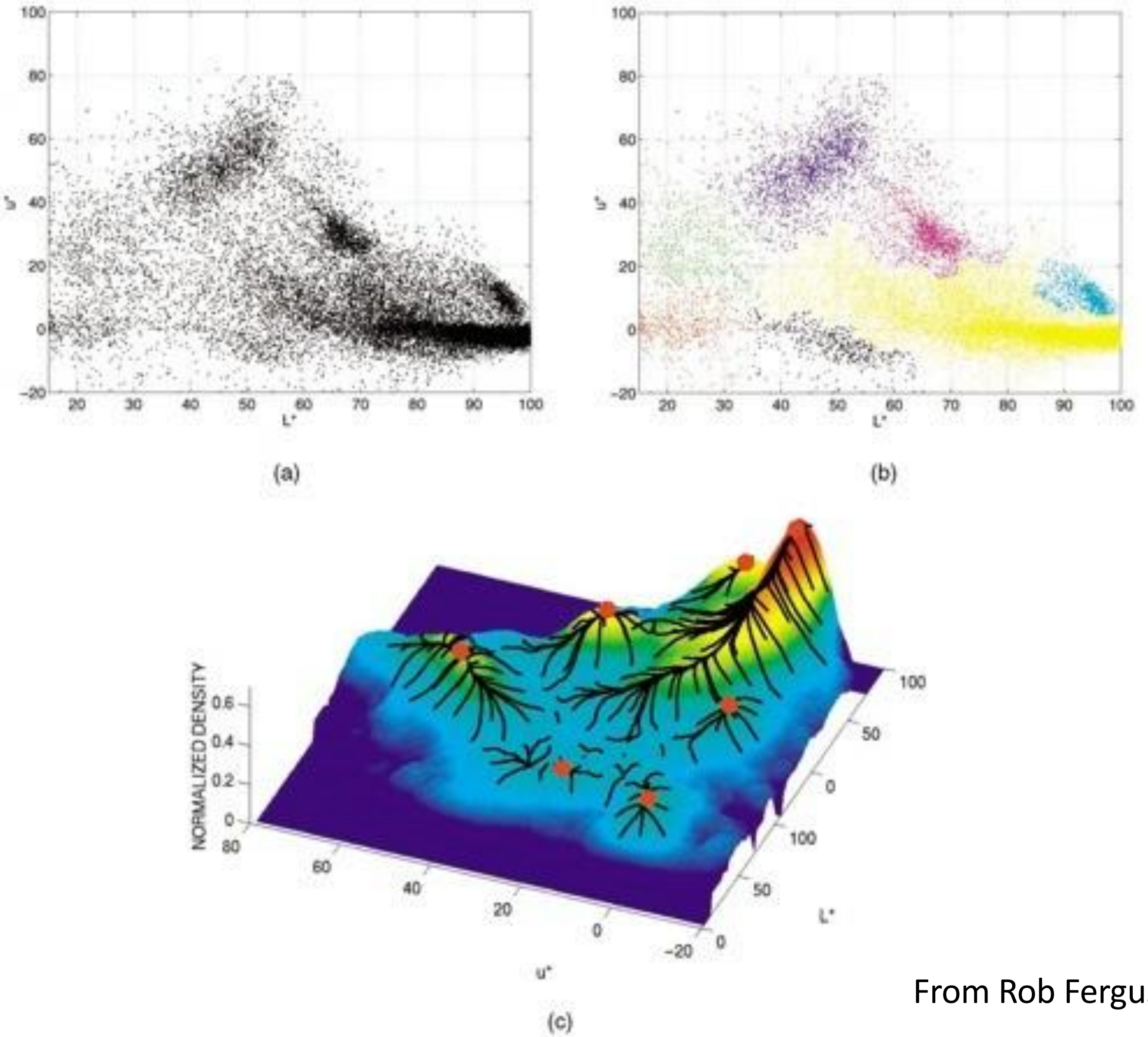
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

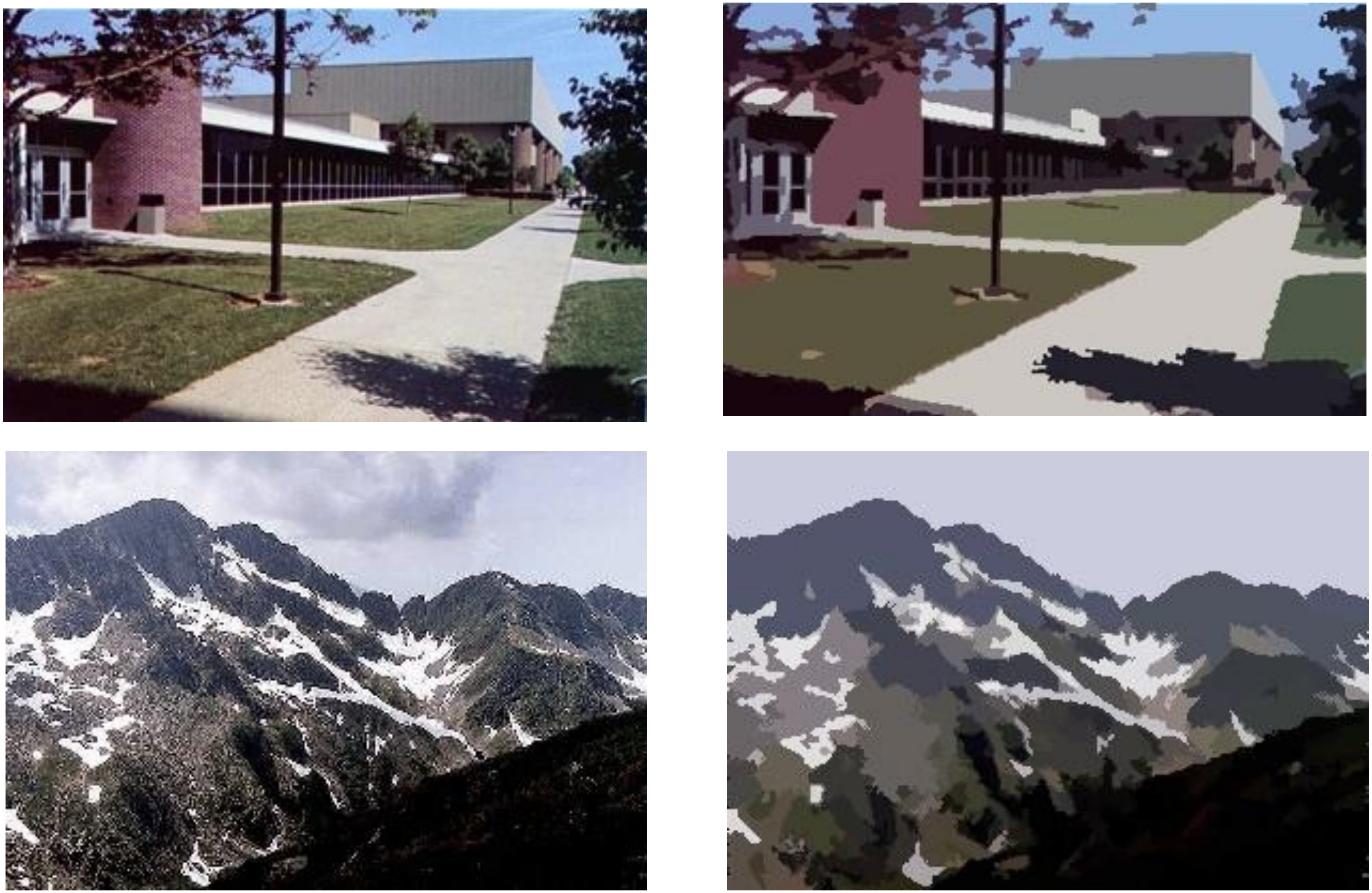
Mean shift Clustering

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same "peak" or mode



From Rob Fergus

Mean-shift Clustering Result






More Results

Mean shift Clustering

- Pros
  - Does not assume spherical clusters
  - Just a single parameter (window size)
  - Finds variable number of modes
  - Robust to outliers
- Cons
  - Output depends on window size
  - Computationally expensive
  - Does not scale well with dimension of feature space

Comparison

Watershed

- May be supervised or unsupervised
- No need for threshold parameter
- Partitions image into many regions
- To get good results best to specify number and positions of seeds

K--Means

- Unsupervised
- Works in high--dimensions
- May produce disjoint segments
- Must specify number of clusters but not position

Mean--Shift

- Unsupervised
- Works in high--dimensions (kind of)
- May produce disjoint segments
- "Only" specify windows size

Overview

What is image segmentation?

Types of segmentation algorithms

1. Thresholding, region labelling and growing algorithms
   – (connected components, region growing, watershed)
2. Statistical Segmentation
   – (k-means, mean shift)
3. Graph based methods
   – (Merging algorithms, splitting algorithms, split/merge)
4. Edge based methods
   – (Intelligent Scissors, Snakes)

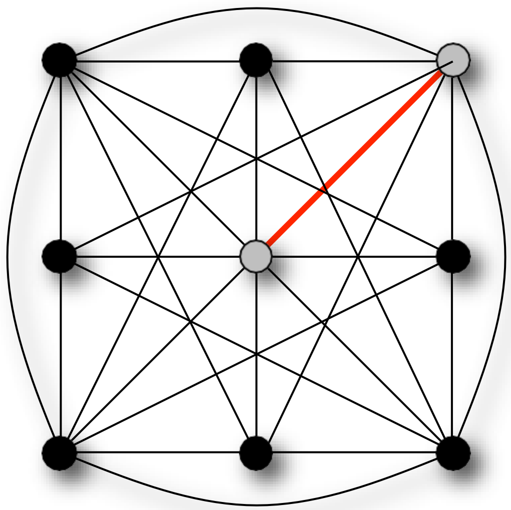
Overview

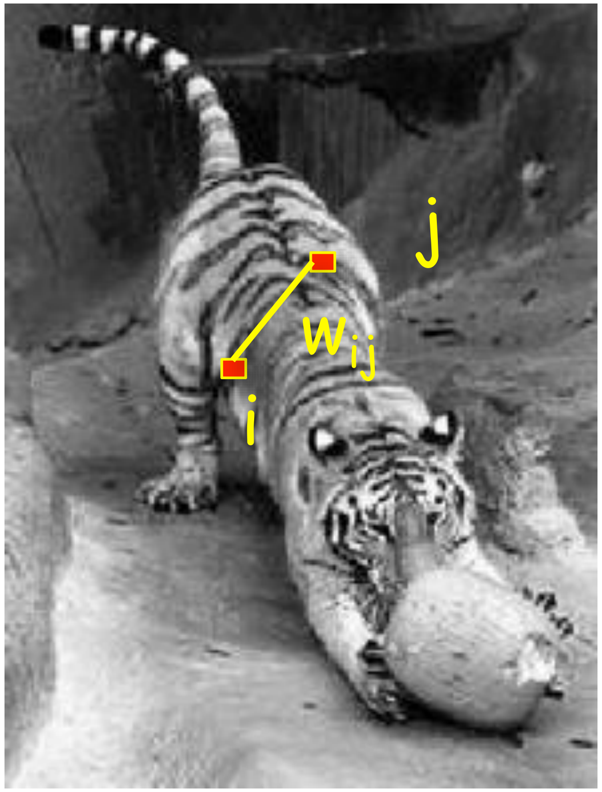
What is image segmentation?

Types of segmentation algorithms

1. Thresholding, region labelling and growing algorithms
   - (connected components, region growing, watershed)
2. Statistical Segmentation
   - (k-means, mean shift)
3. Graph based methods
   - (Merging algorithms, splitting algorithms, split/merge)
4. Edge based methods
   - (Intelligent Scissors, Snakes)

Images as Graphs





- Node for every pixel
- Edge between every pair of pixels
  (or every pair of "sufficiently close" pixels)
- Each edge is weighted by the *affinity* or similarity of the two nodes

Source: S. Seitz

Greedy Merging Algorithm

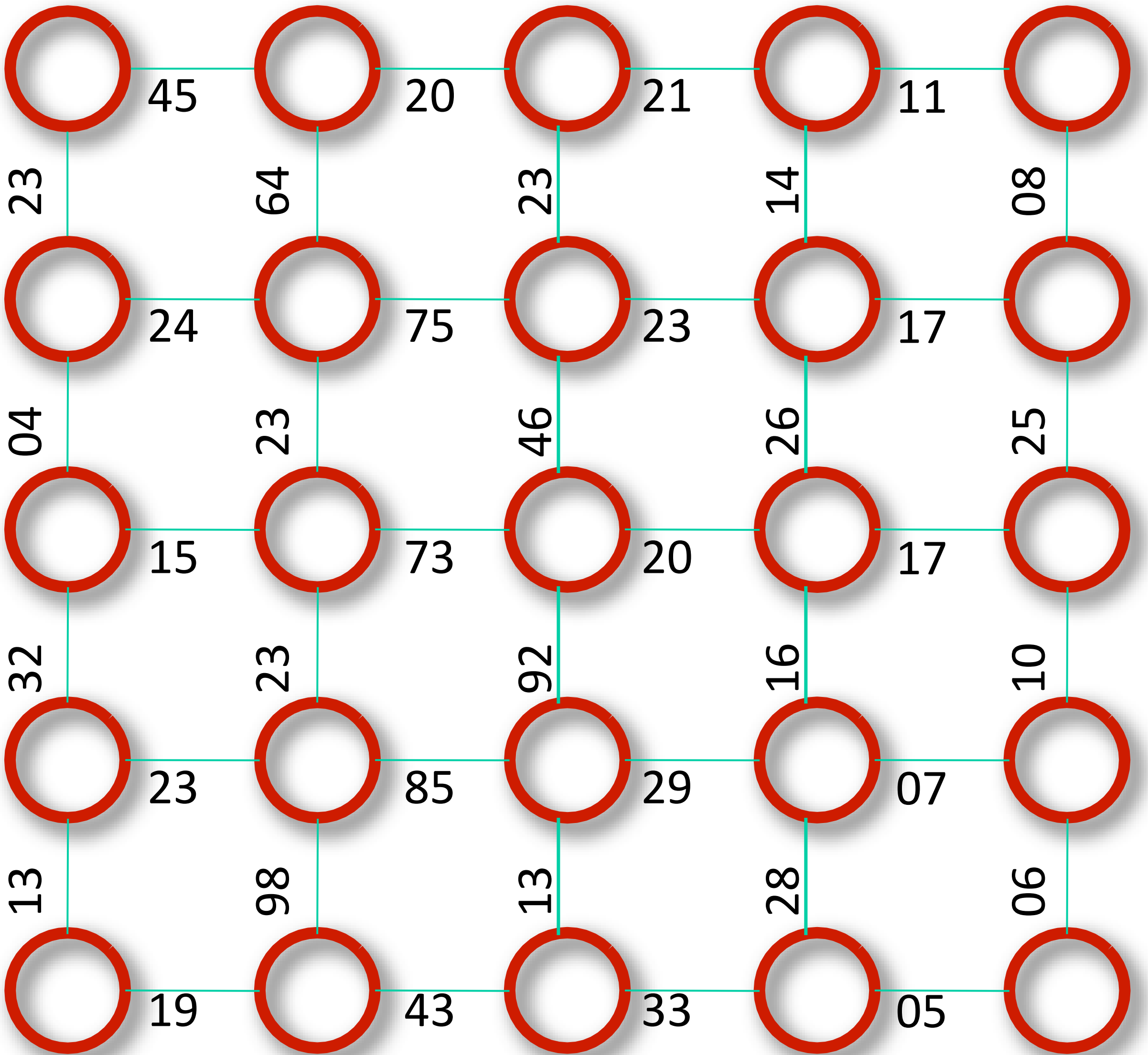
1. Initialize each pixel to a separate segment

2.Assign a cost, c_{ij} to each edge
(possibly based on pixel difference)

3. Sort all edges by edge strength

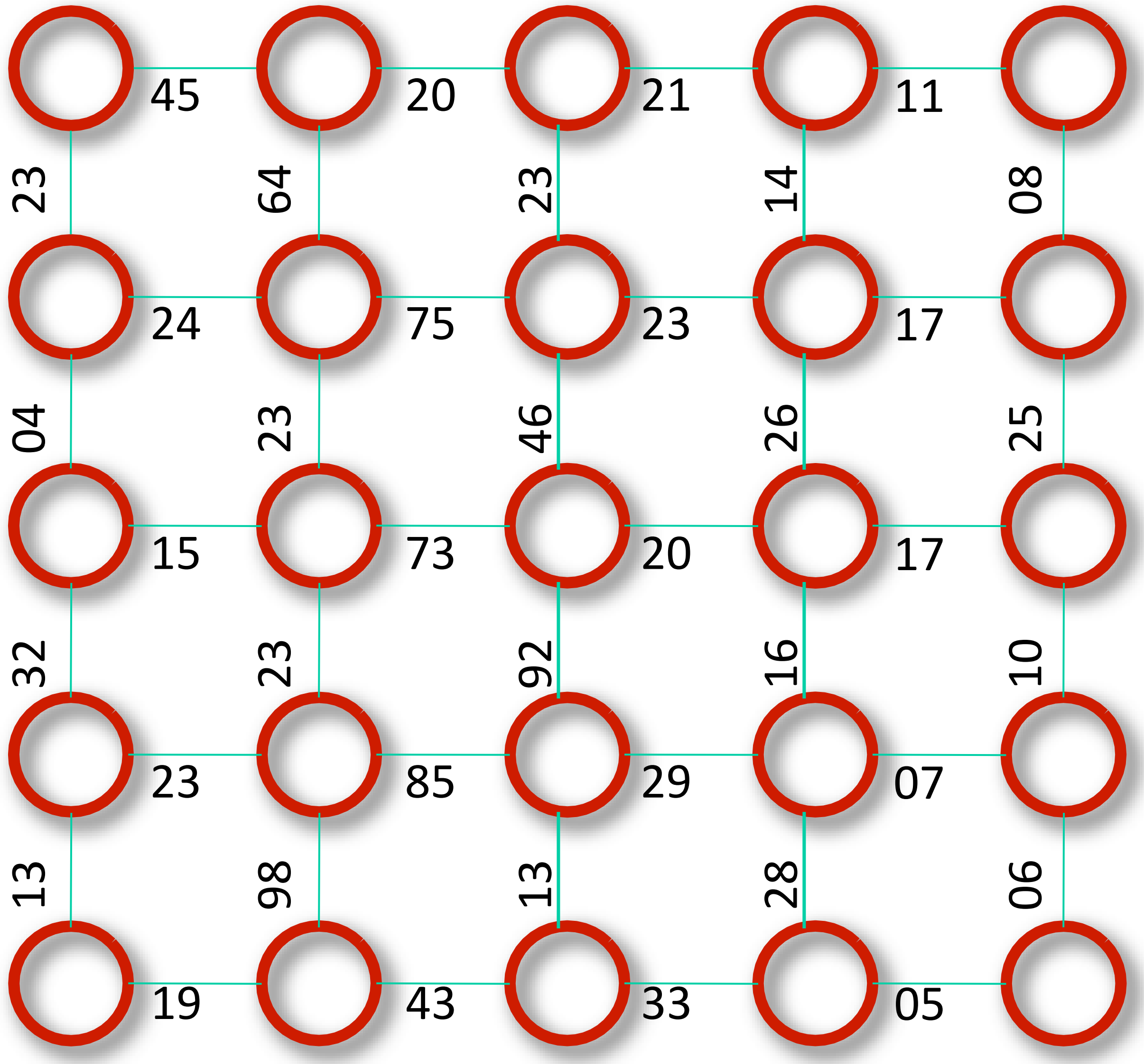
4.Connect edges in order of strength, merging segments as we go

5.Stop when we have reached some threshold edge strength



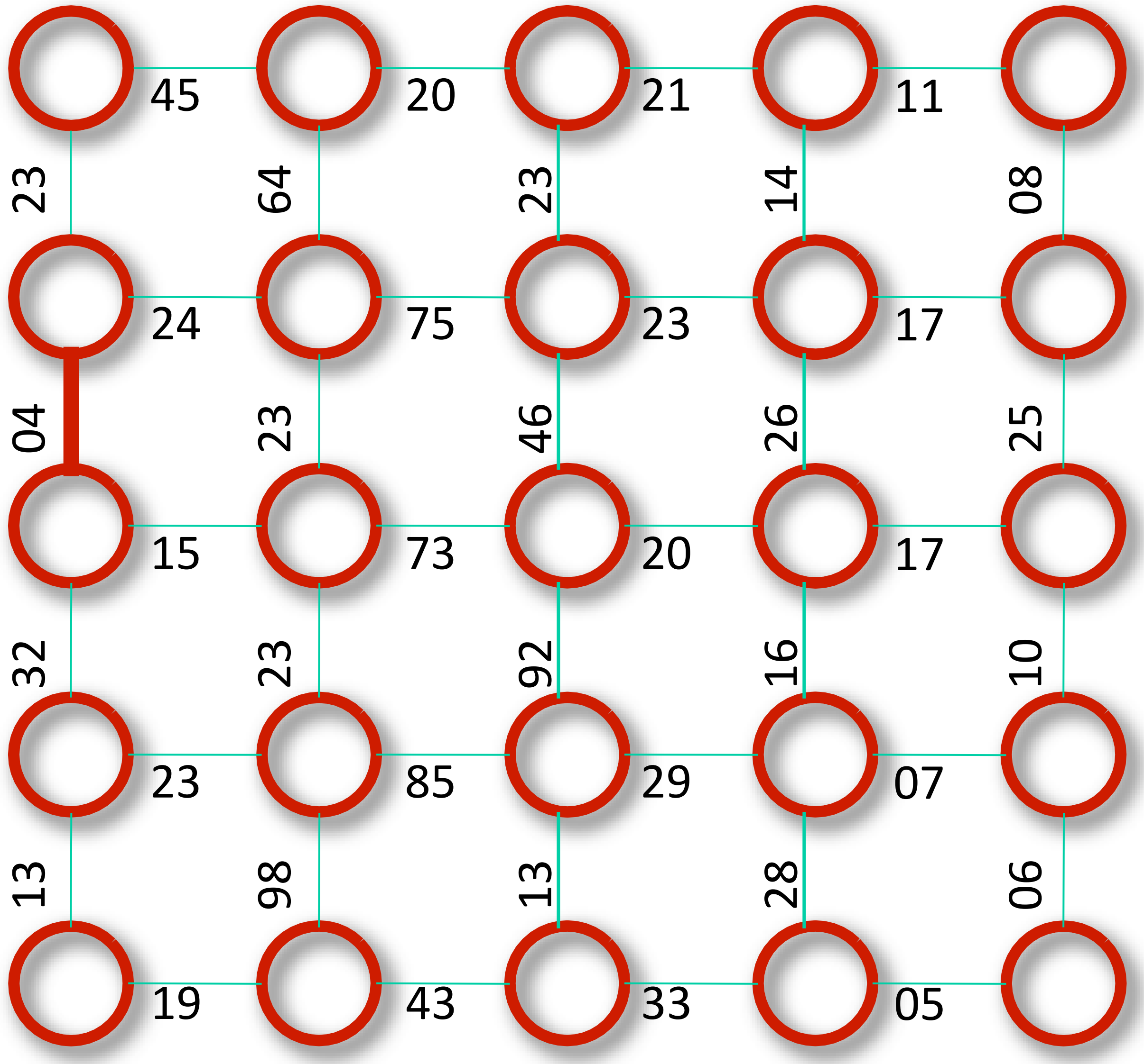
Extract all edge strengths

45, 20, 21, 11 ,24, 75, 23, 17, 15, 73, 20 ,17, 23, 85, 29, 07, 19,43,33, 05, 13, 22, 04,23, 98, 23, 23, 64, 13, 92, 46, 23, 28, 16, 26, 14, 06, 10, 25, 08



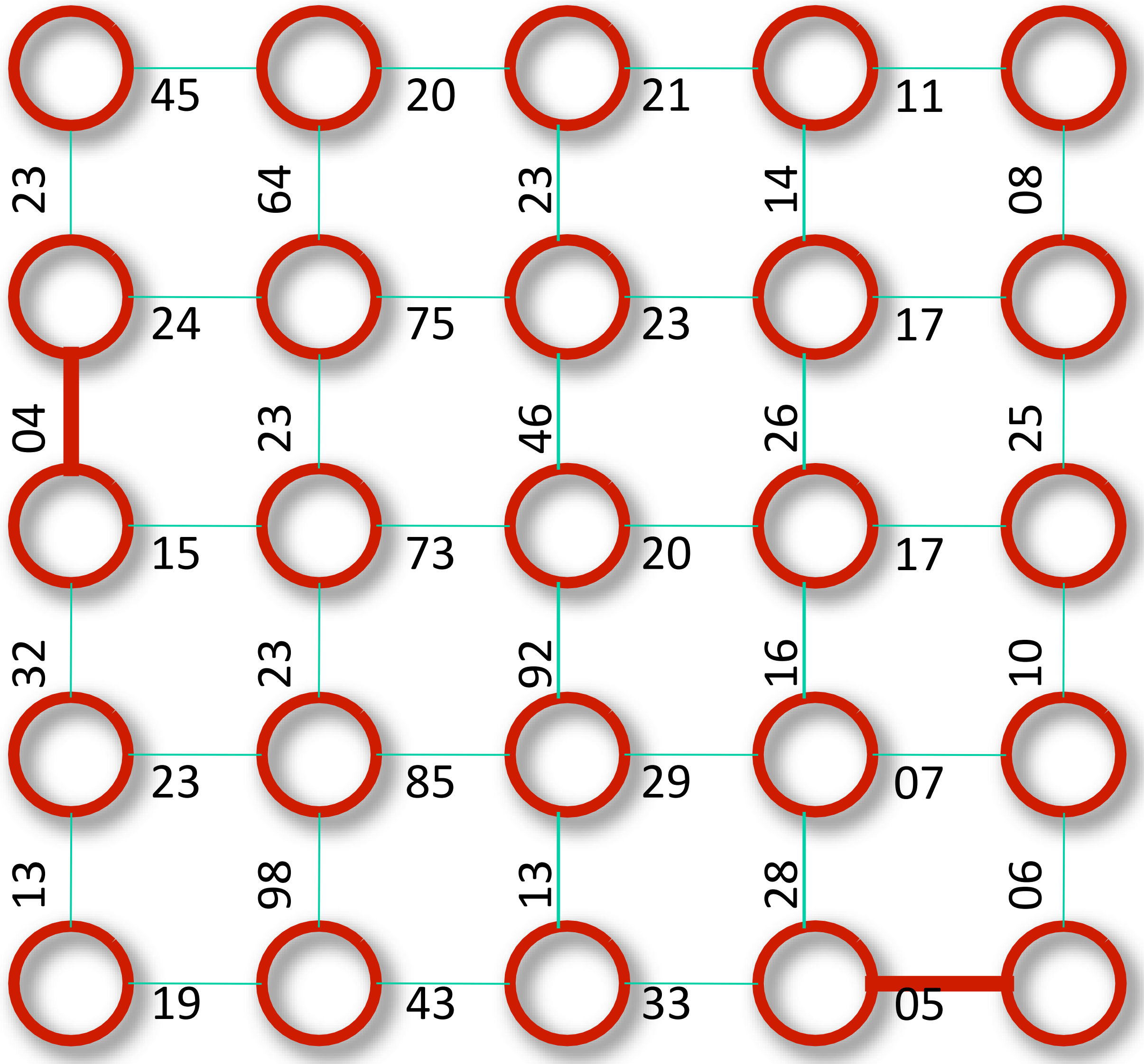
Sort edge strengths (remembering where they came from)

04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98



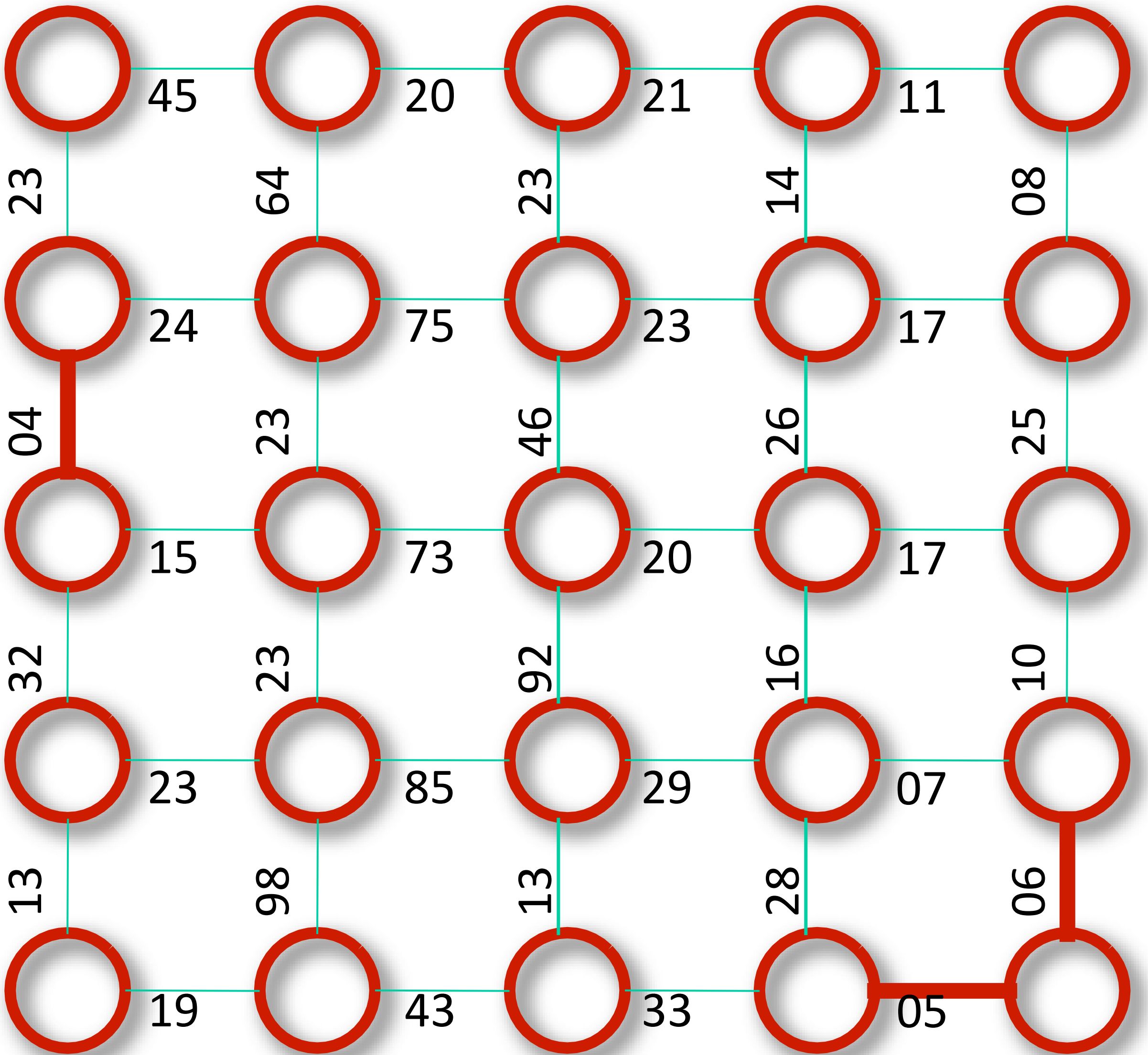
Connect together pixels, joining weakest first (until reach some threshold)

04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98



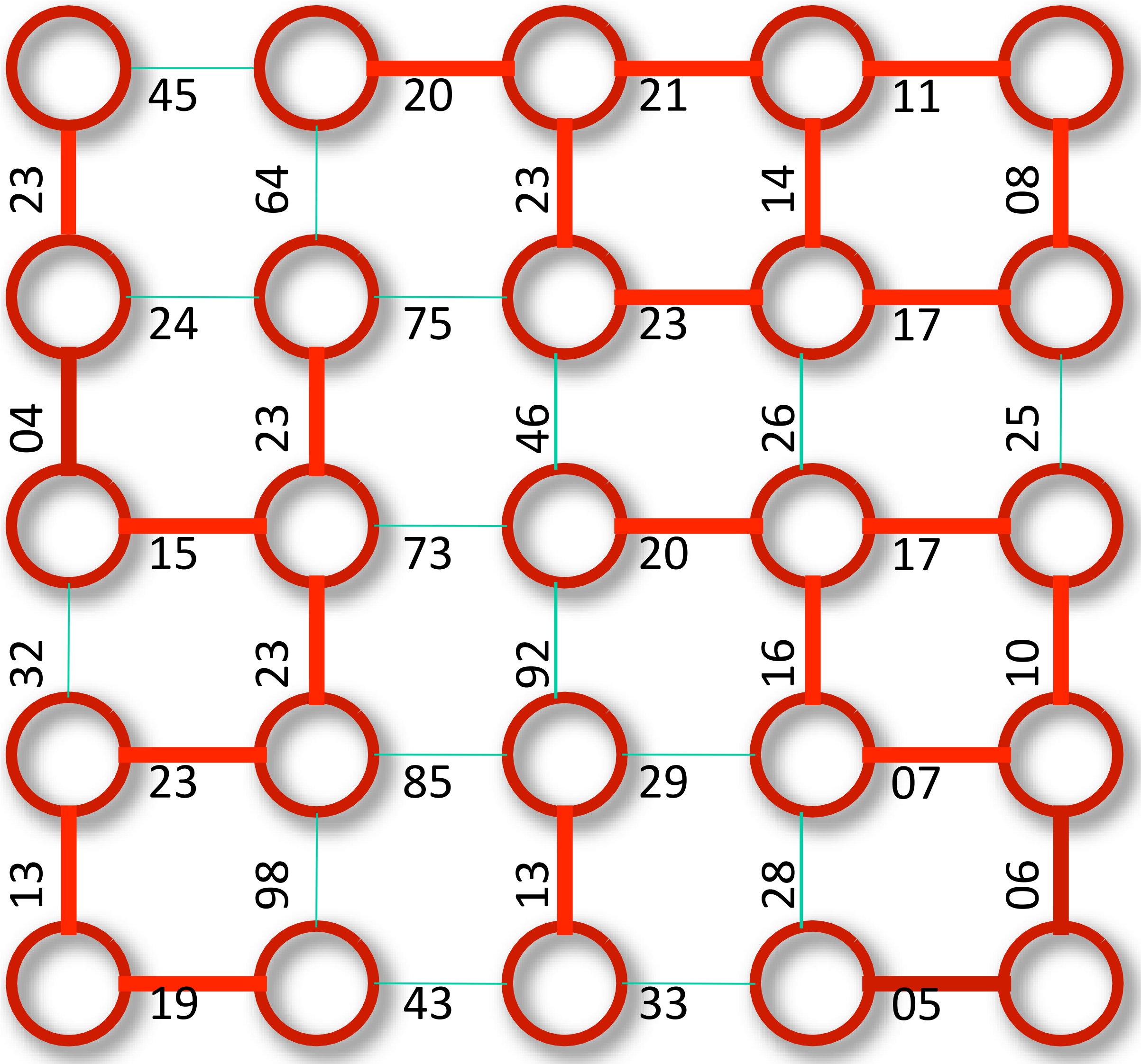
Connect together pixels, joining weakest first (until reach some threshold)

04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98



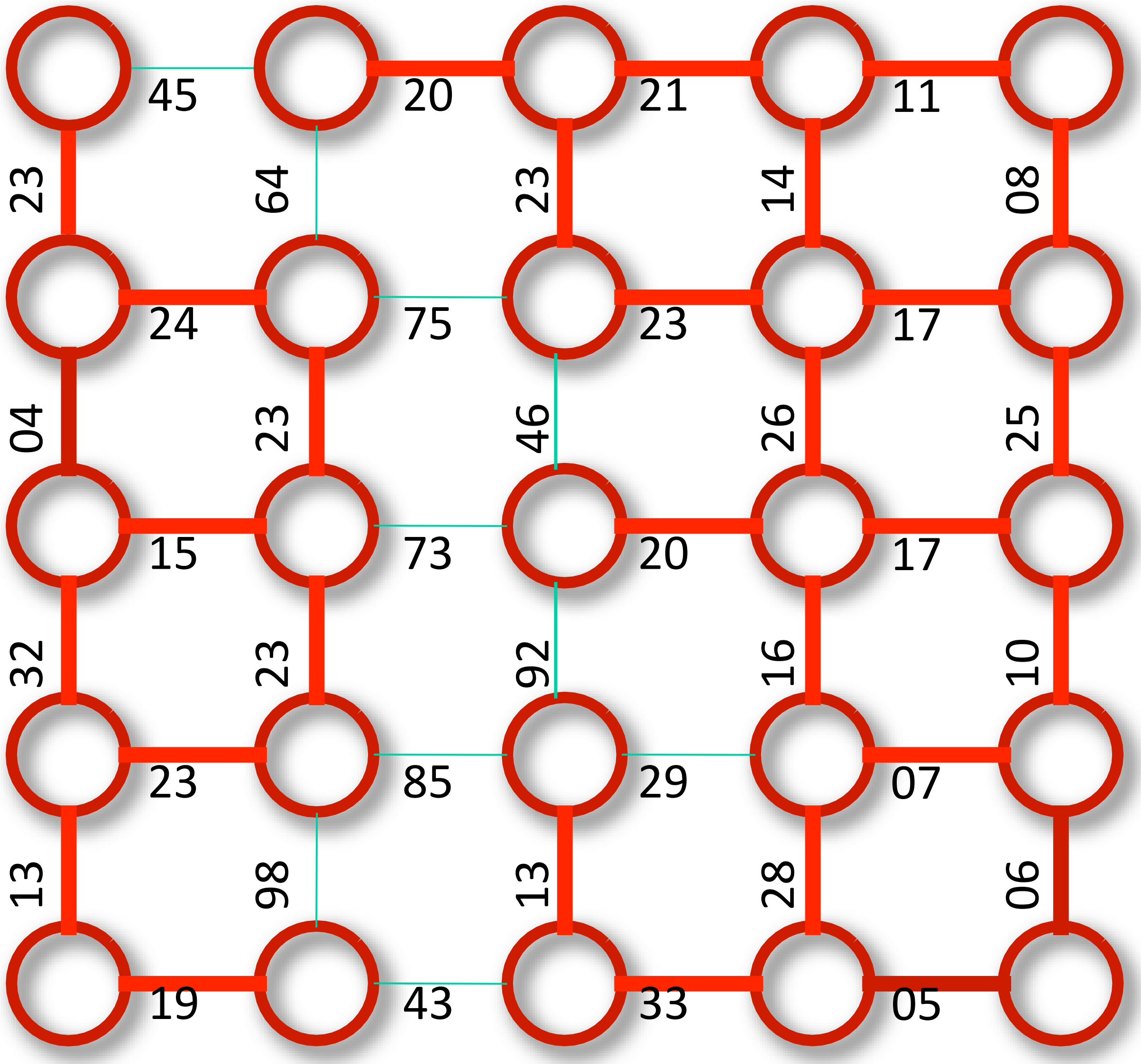
Connect together pixels, joining weakest first (until reach some threshold)

04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98



Connect together pixels, joining weakest first (until reach some threshold)

04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98



Connect together pixels, joining weakest first (until reach some threshold)

04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98

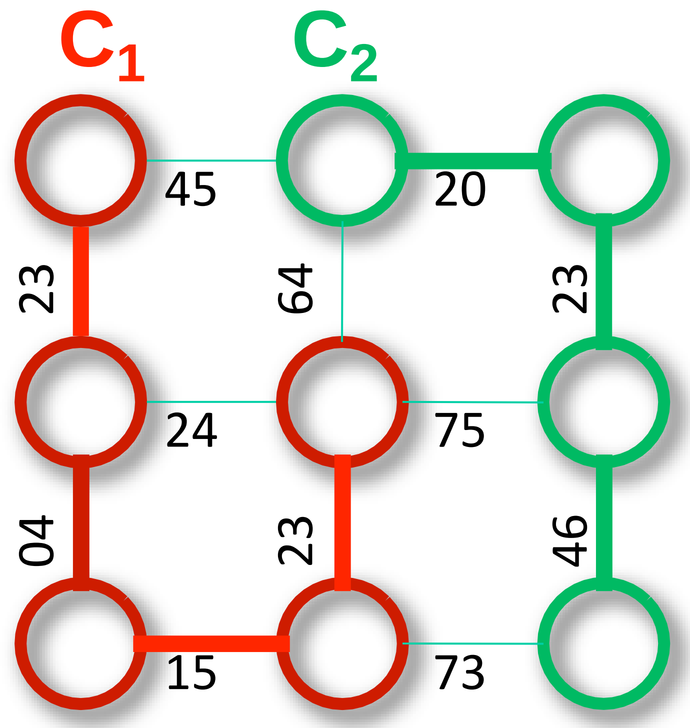
A Better Merging Algorithm

Efficient Graph--Based Image Segmentation **(Felzenswalb, Huttenlocher [IJCV 2004])**

- Assign weight $w(e_{ij})$ to each edge e_{ij}
- Sort edges by weight and run through
- Define Int(C) to be the maximum cost weight in the minimum spanning tree of cluster C
- Define Dif(C1,C2) to be lowest cost link joining components
- Merge components if

$$\mathrm{Dif}(C_{1,}C_2) < \min(\mathrm{Int}(C_1) + \tau(C_1), \mathrm{Int}(C_2) + \tau(C_2))$$

where $\tau(C) = k / |C|$



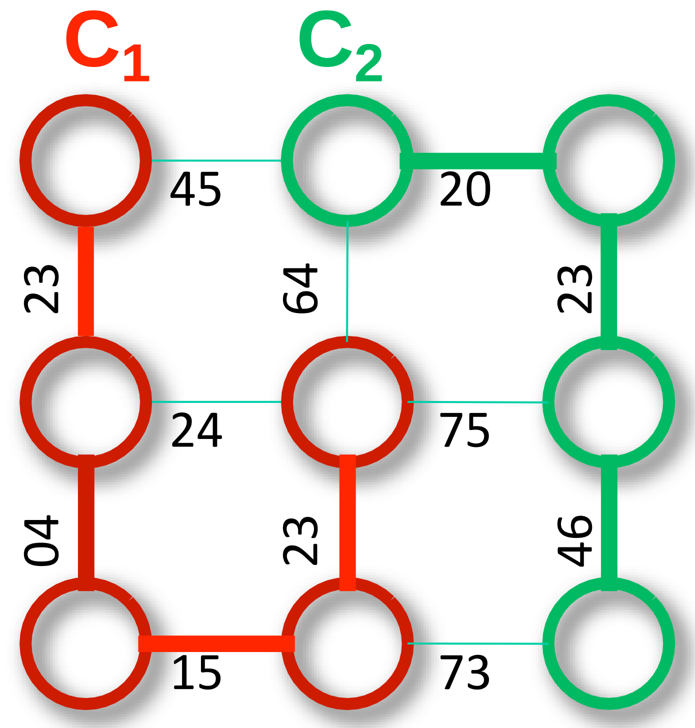
A Better Merging Algorithm

Efficient Graph--Based Image Segmentation (Felzenswalb, Huttenlocher [IJCV 2004])

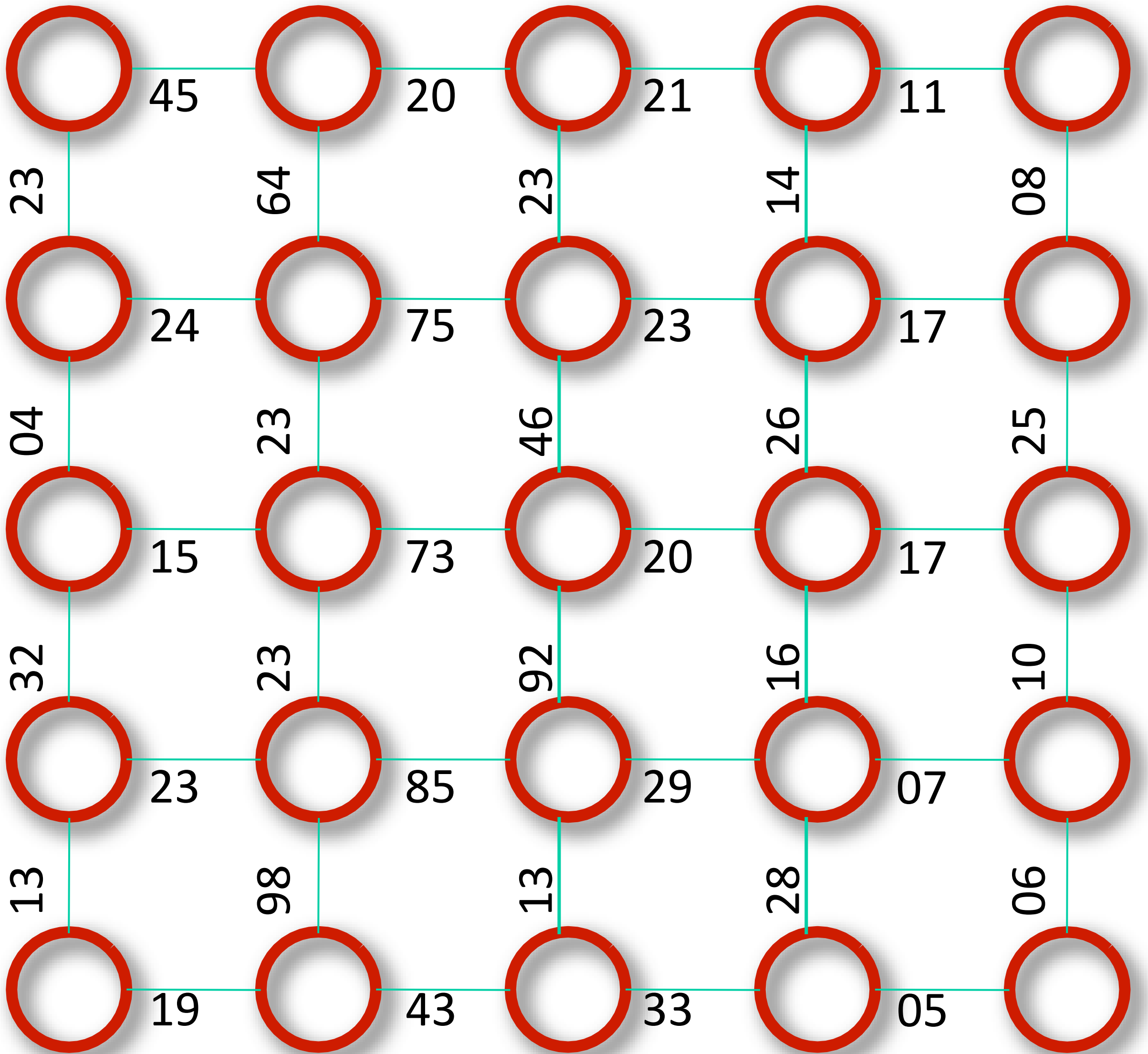
- Assign weight $w(e_{ij})$ to each edge e_{ij}
- Sort edges by weight and run through
- Define Int(C) to be the maximum cost weight in the minimum spanning tree of cluster C
- Define Dif(C1,C2) to be lowest cost link joining components
- Merge components if

$$\mathrm{Dif}(C_1, C_2) < \min(\mathrm{Int}(C_1) + \tau(C_1), \mathrm{Int}(C_2) + \tau(C_2))$$

where $\tau(C) = k / |C|$



Here: Int(C1) = 23
Int(C2) = 46
Dif(C1,C2) = min(45,64,73,75)=45



Sort edge strengths (remembering where they came from)

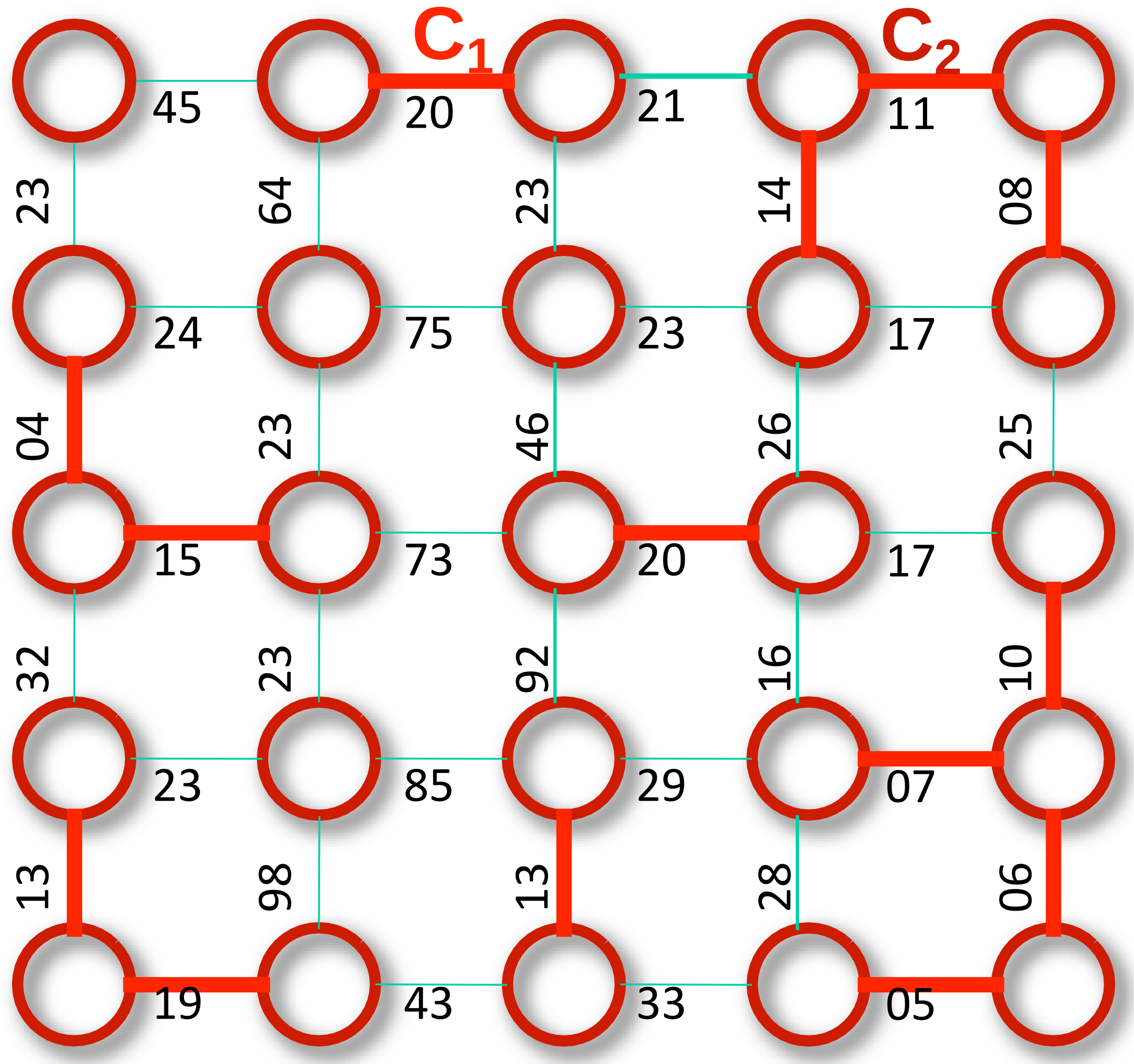
04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98

Let k = 50

Int(C1) = 20
Int(C2) = 14
Dif(C1,C2) = 21

Mint = min(20+50/2,14+50/4)
= 26.5

Dif(C1,C2)<Mint
so merge



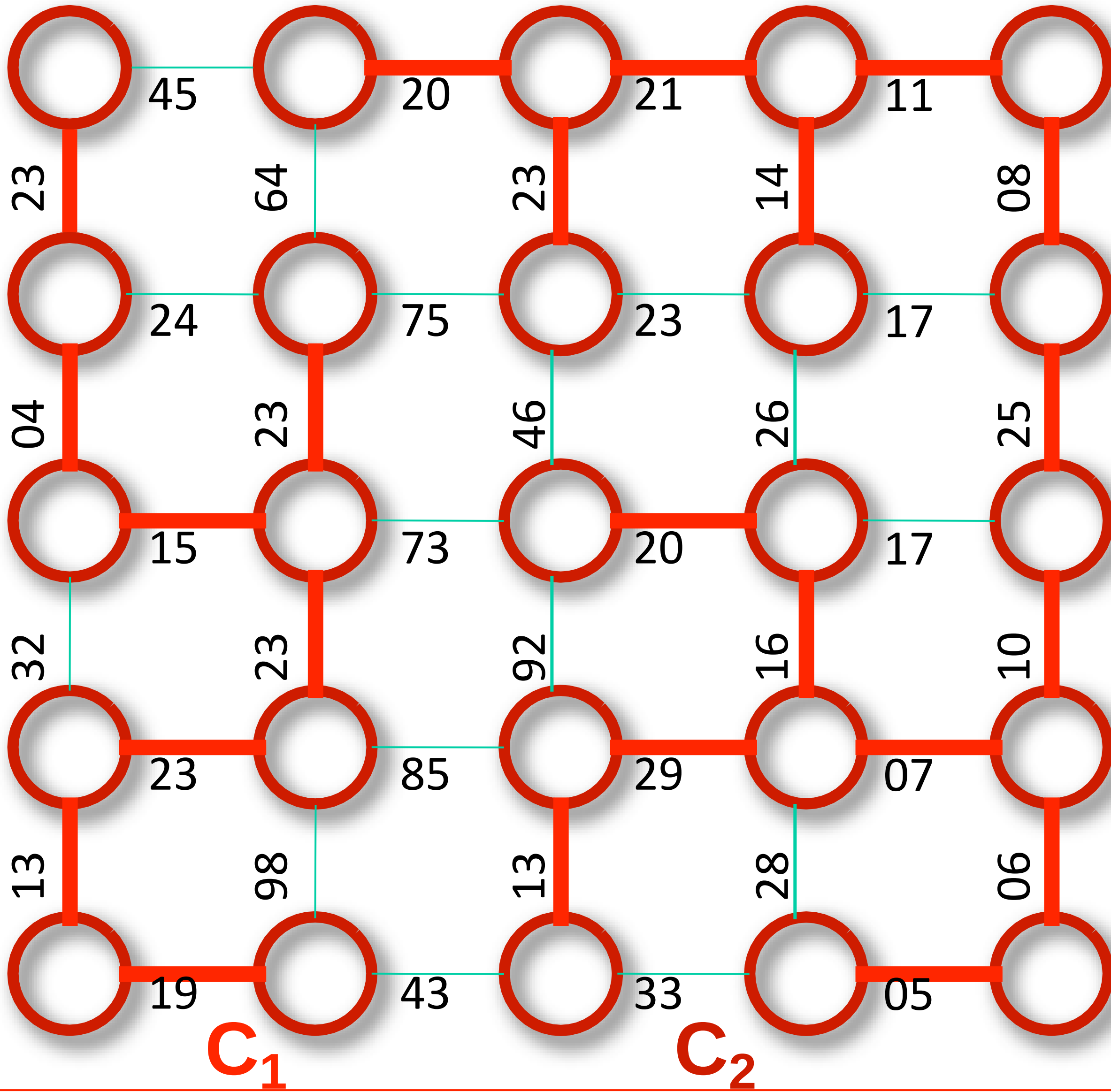
Sorted Edge Strengths: 04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98

Let k = 50

Int(C1) = 23
Int(C2) = 29
Dif(C1,C2) = 43

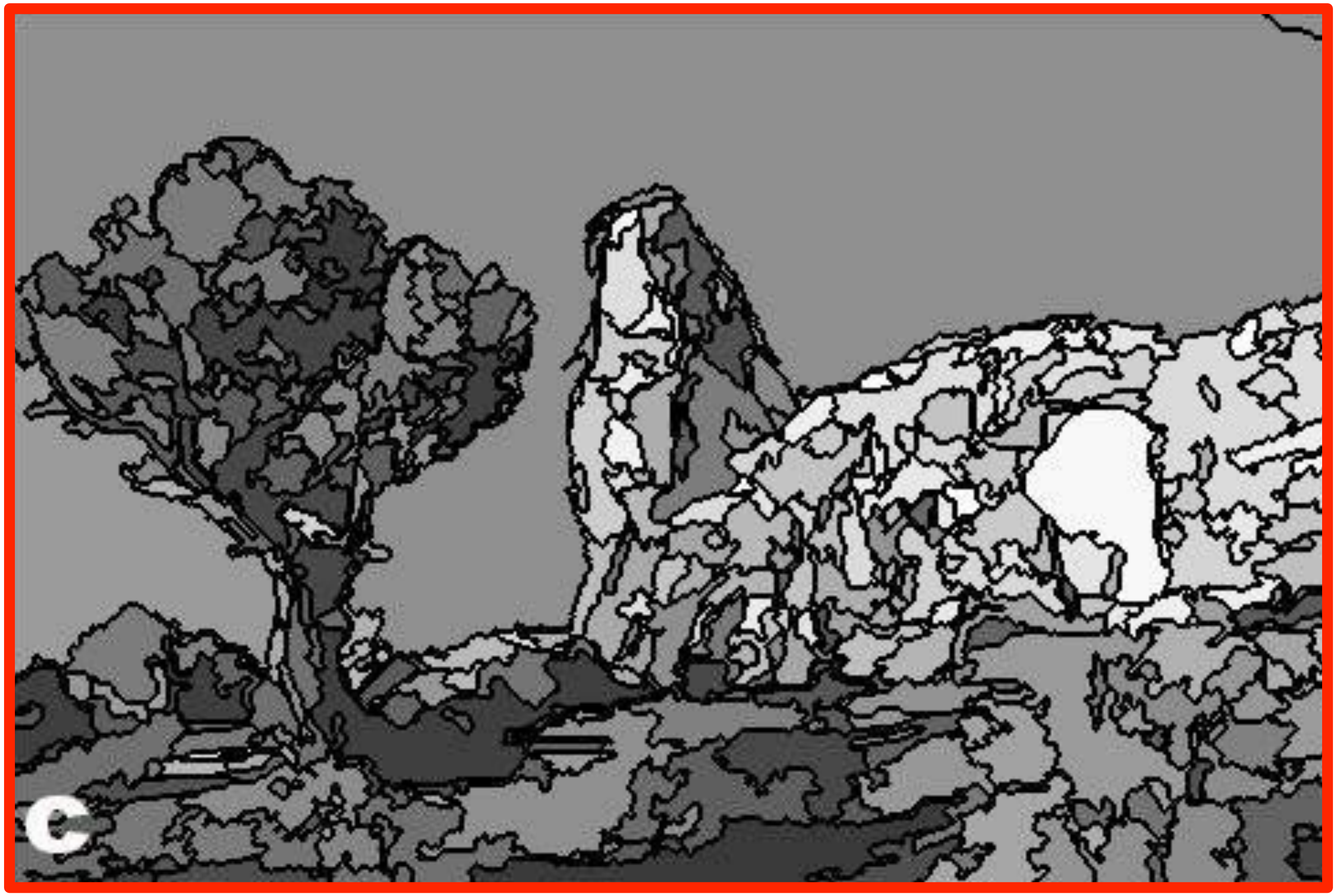
Mint = min(23+50/9,19+50/16)
= 28.566

Dif(C1,C2)>Mint
so don't merge



Sorted Edge Strengths: 04, 05, 06, 07, 08, 10, 11, 13, 13, 14, 15, 16, 17, 17, 19, 20, 20, 21, 22, 23, 23, 23, 23, 23, 23, 24, 25, 26, 28, 29, 33, 43, 45, 46, 64, 73, 75, 85, 92, 98

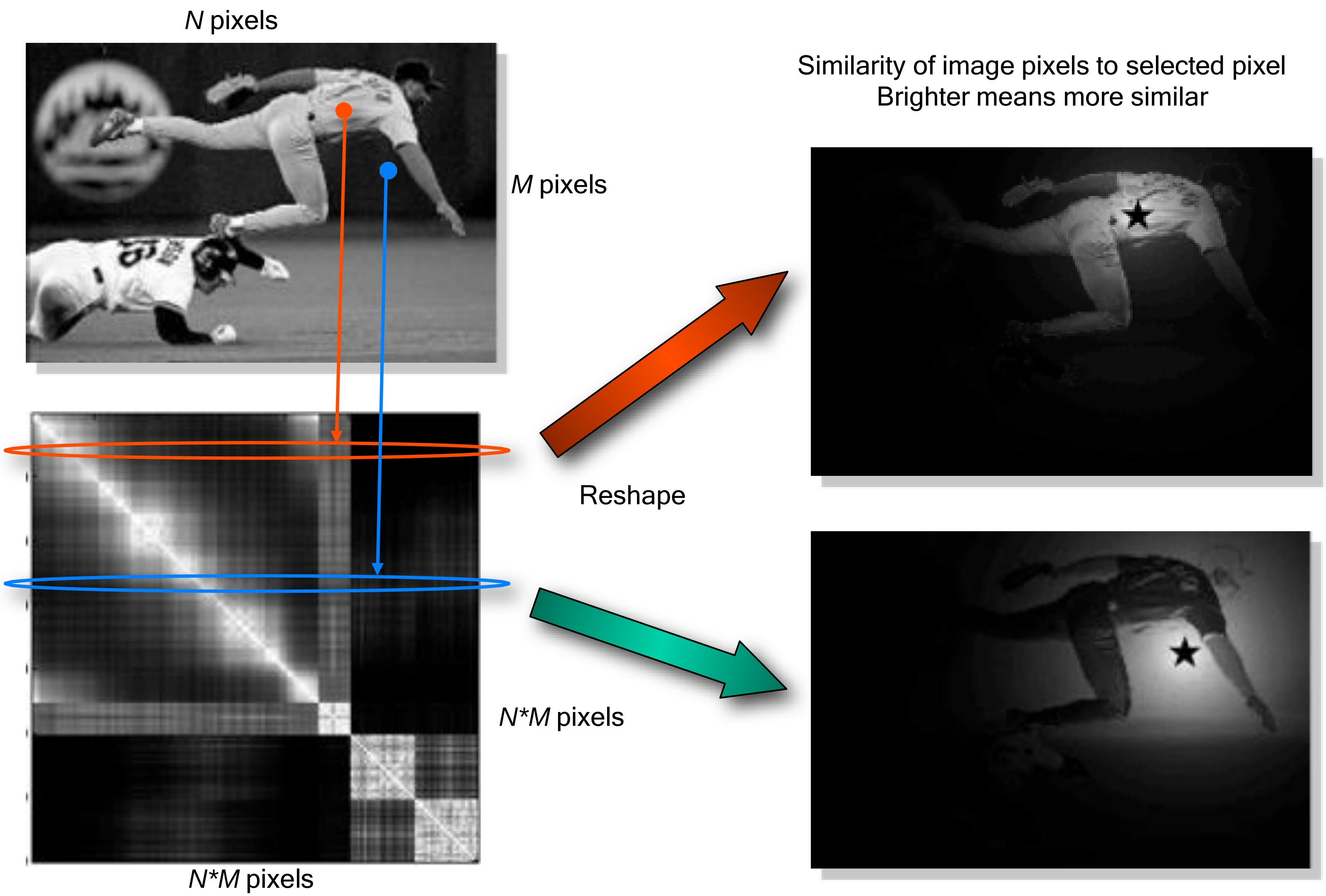
Example Results



Example Results

- Very fast
- But sensitive to noise
- Greedily chooses (too) large regions

Affinity Matrix



Pixel Similarity Functions

Intensity

$$W(i,j) = e^{\frac{-\left\| I_{(i)} - I_{(j)} \right\|_2^2}{\sigma_I^2}}$$

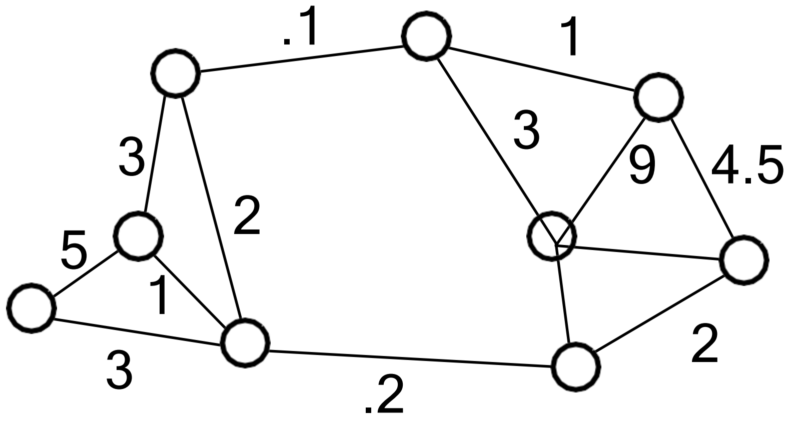
Distance

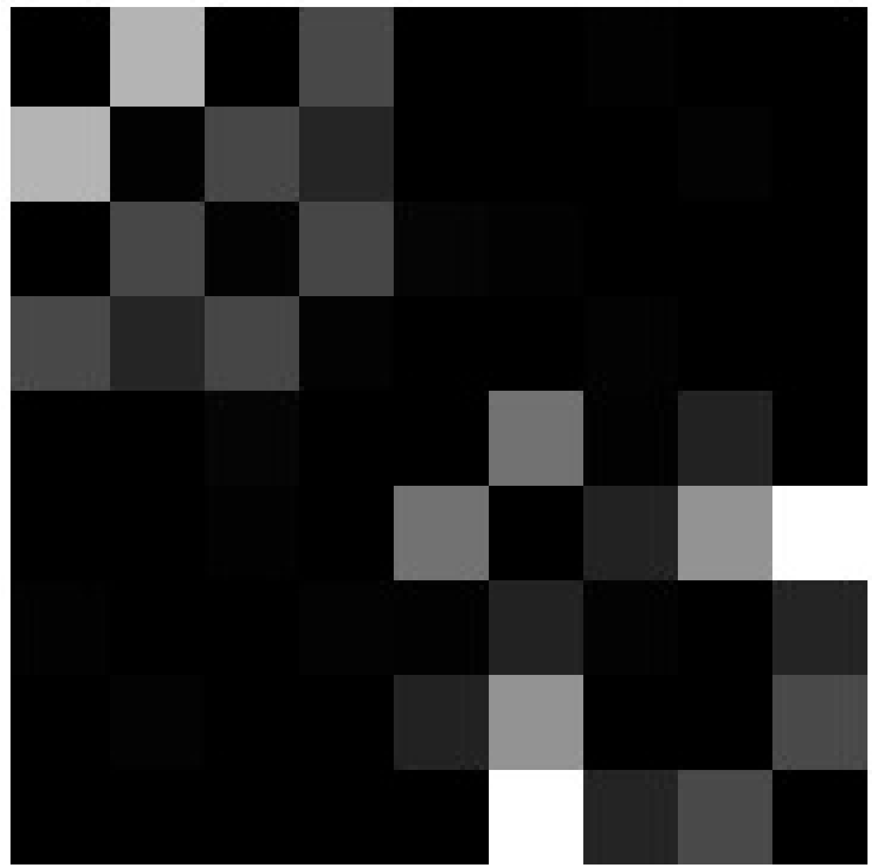
$$W(i,j) = e^{\frac{-\left\| X_{(i)} - X_{(j)} \right\|_2^2}{\sigma_X^2}}$$

Texture

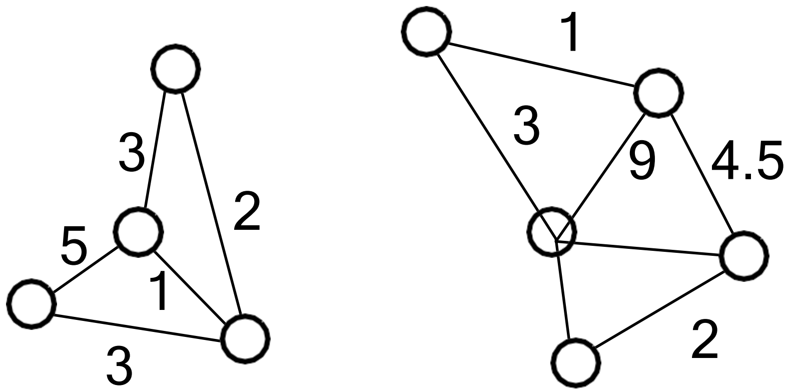
$$W(i,j) = e^{\frac{-\left\| c_{(i)} - c_{(j)} \right\|_2^2}{\sigma_c^2}}$$

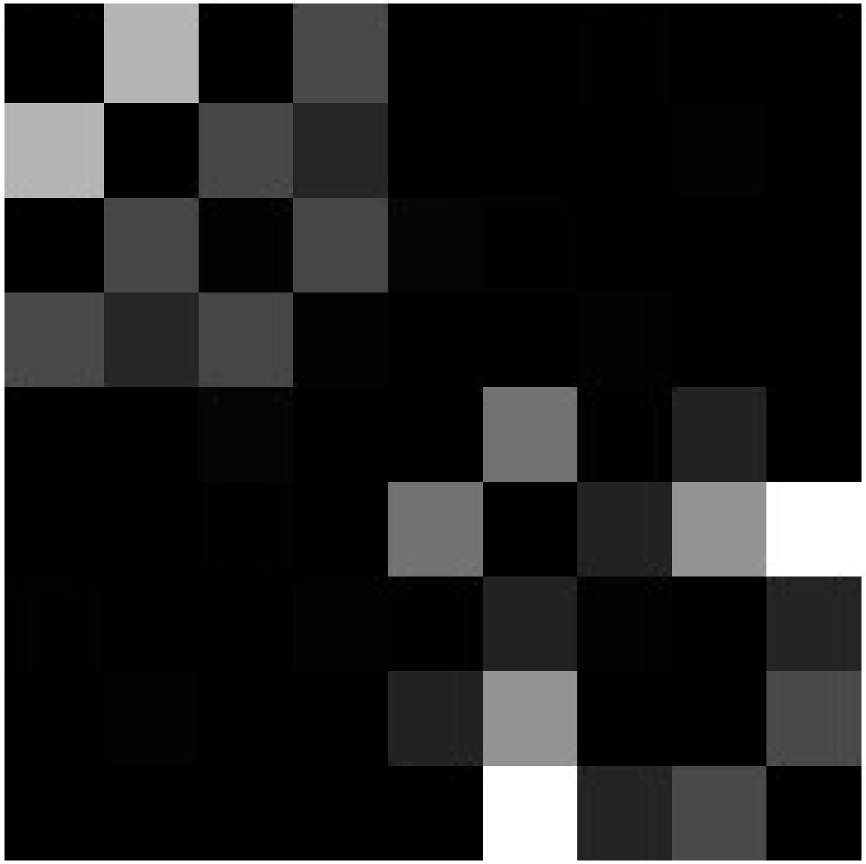
Minimum Cut and Clustering





Minimum Cut and Clustering

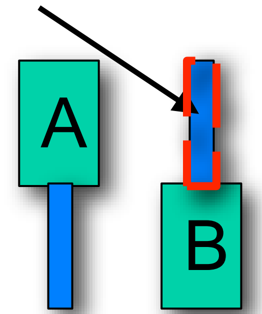




Minimum cut

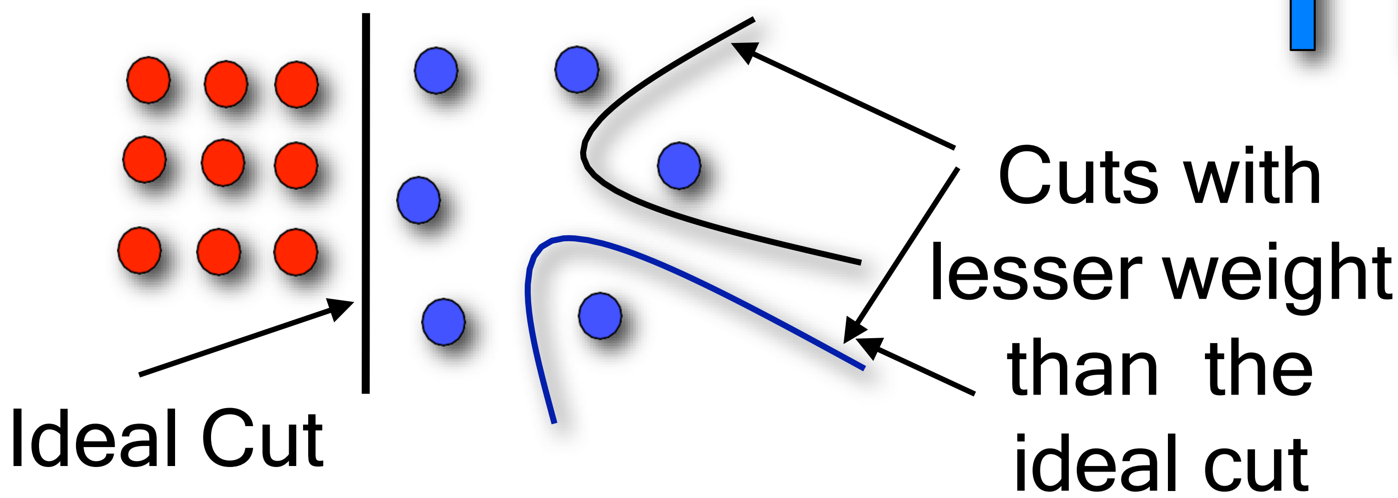
Criterion for partition:

$$\min cut(A, B) = \min_{A,B} \underset{u \in A, v \in B}{} w(u, v)$$



Problem

Weight of cut is directly proportional to the number of edges in the cut.



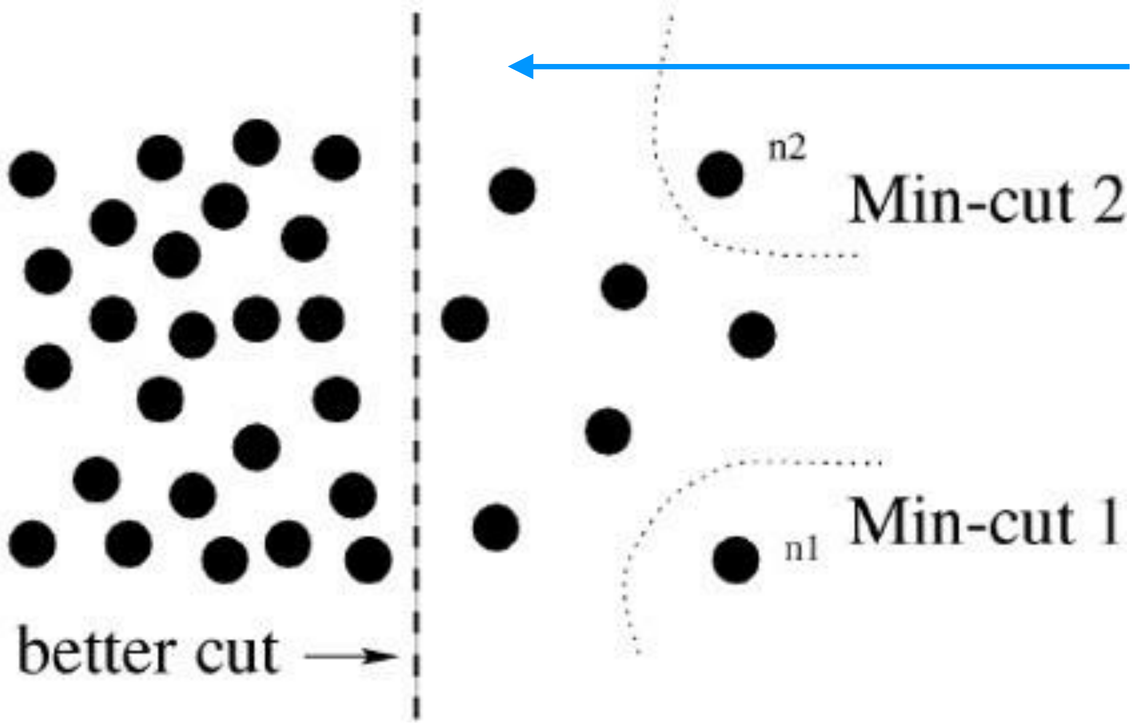
First proposed by Wu and Leahy

Normalized Cut

Normalized cut or balanced cut:

$$N\,cut(A,B) := cut(A,B)\ \frac{1}{vol(A)} + \frac{1}{vol(B)}$$

finds better cut



Analyzing Normalized Cuts

- State of the art results
- Huge storage requirement and time complexity
- Bias towards partitioning into equal segments
- Has problems with textured backgrounds

* Slide from Khurram Hassan-Shafique