Lecture 21: Grouping

Announcements

- Final project guidelines and sign-up sheet out on Piazza

Writeup (70%)

- Due 4/26 midnight
- Grading rubric provided

Presentation (20%)

- Sign-up sheet for 4/27 - 4/28.
- If no times work, send private Piazza post ASAP.
- 5 mins + 2 mins of questions.
- Recommend splitting time between teammates

Announcements

- Final project guidelines and sign-up sheet out

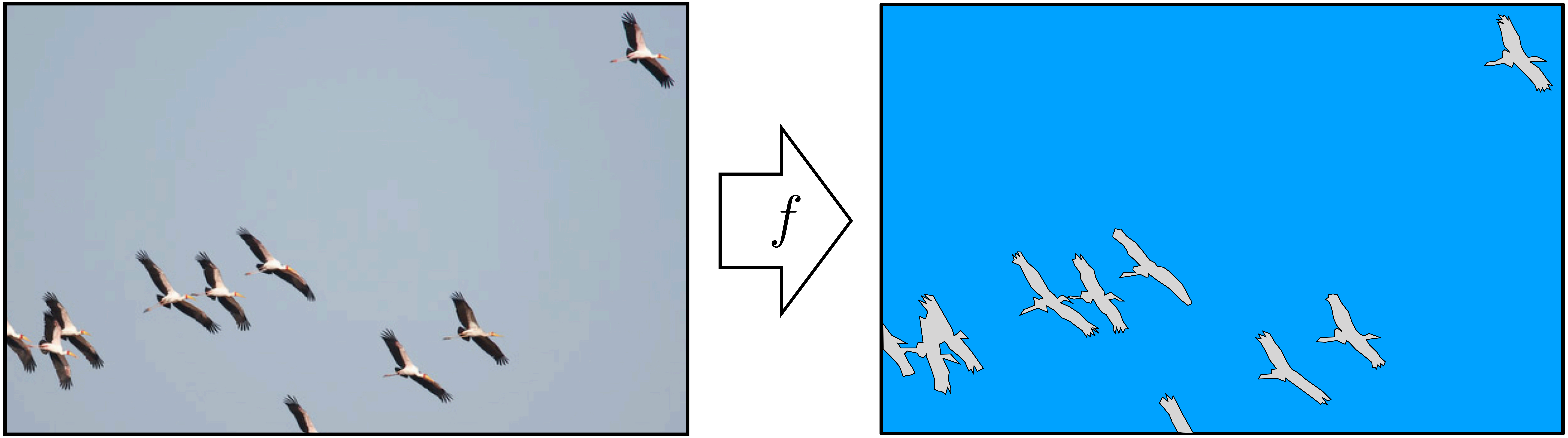
Writeup (70%)

- Should have intro, related work, method, experiments, conclusion.
- At most 6 pages in CVPR format
- Evaluated on:
  - Background: how well do you explain the related work?
  - Completeness: results quality, what worked or didn’t work, thoroughness of evaluation
  - Format and clarity

Today

- Image segmentation
- Edge-aware image processing

Recall: **semantic** segmentation problem



(Colors represent categories)

Source: Torralba, Freeman, Isola

Bottom-up segmentation

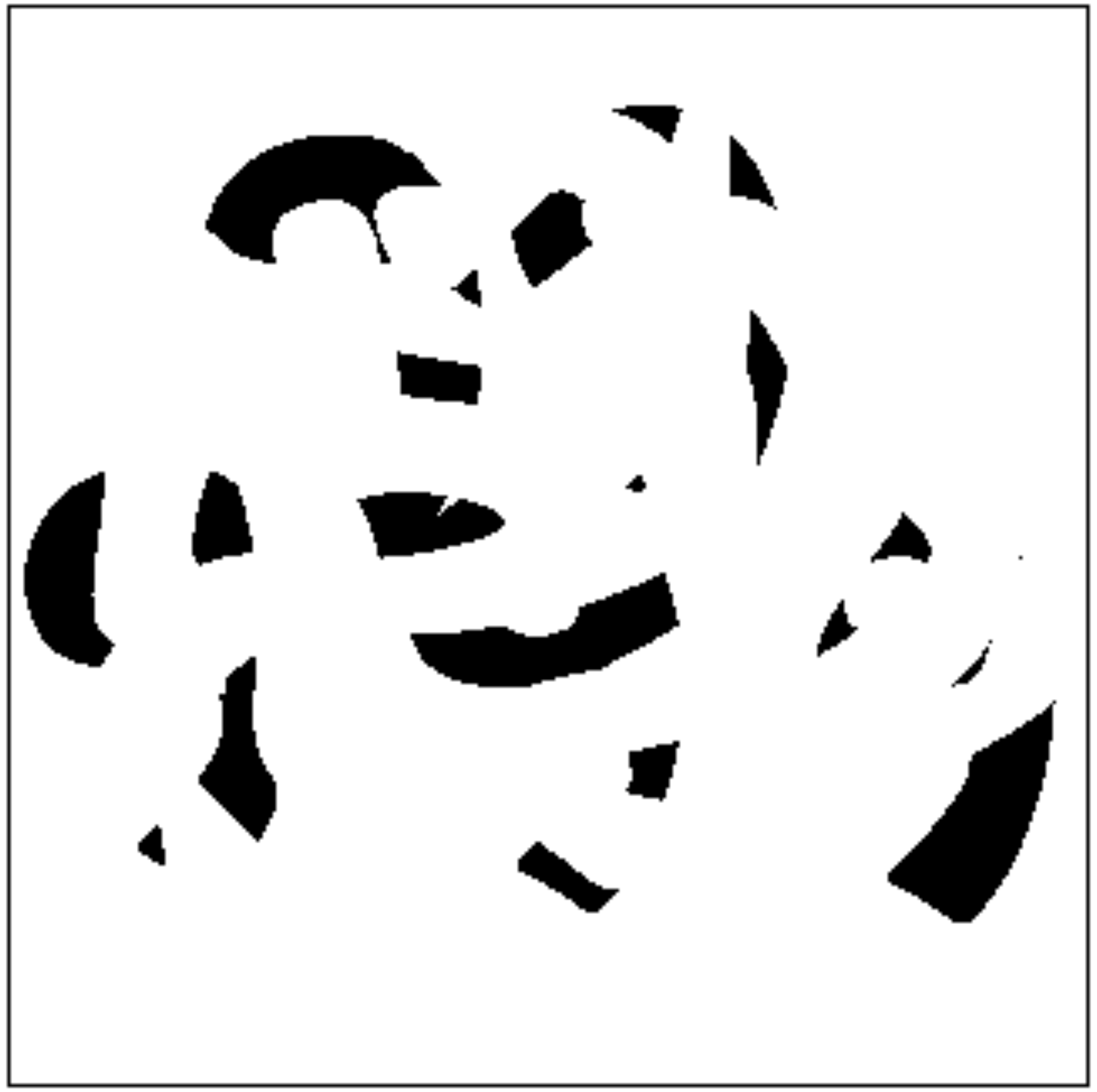
- Group together similar-looking pixels
  - "Bottom-up" process
  - Unsupervised

Source: A. Oliva

A “simple” segmentation problem

Source: Torralba and Freeman

Segmentation is a global process



What are the occluded numbers?

Source: A. Oliva

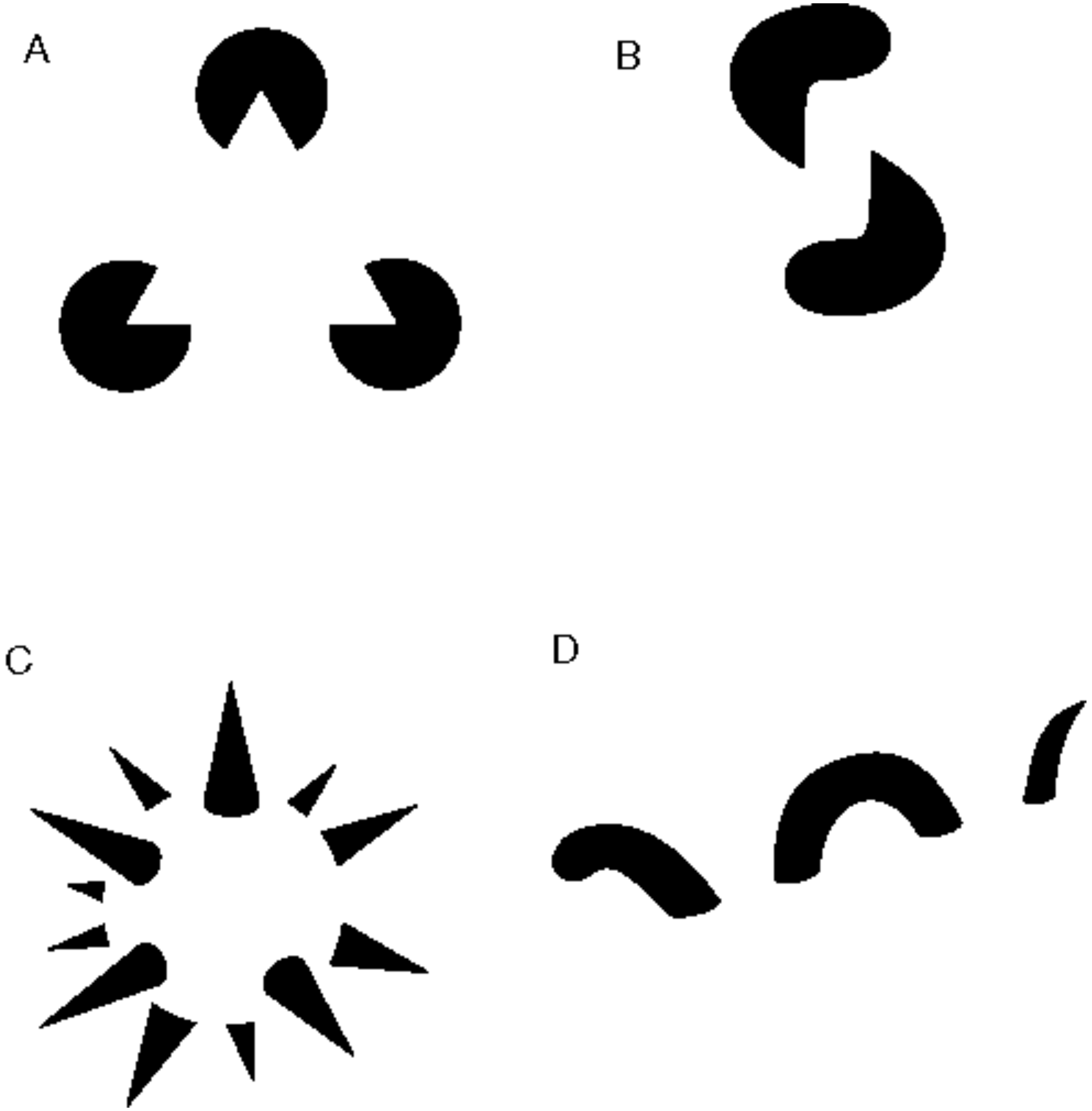
Segmentation is a global process

What are the occluded numbers?

Occlusion is an important cue in grouping.

Source: A. Oliva

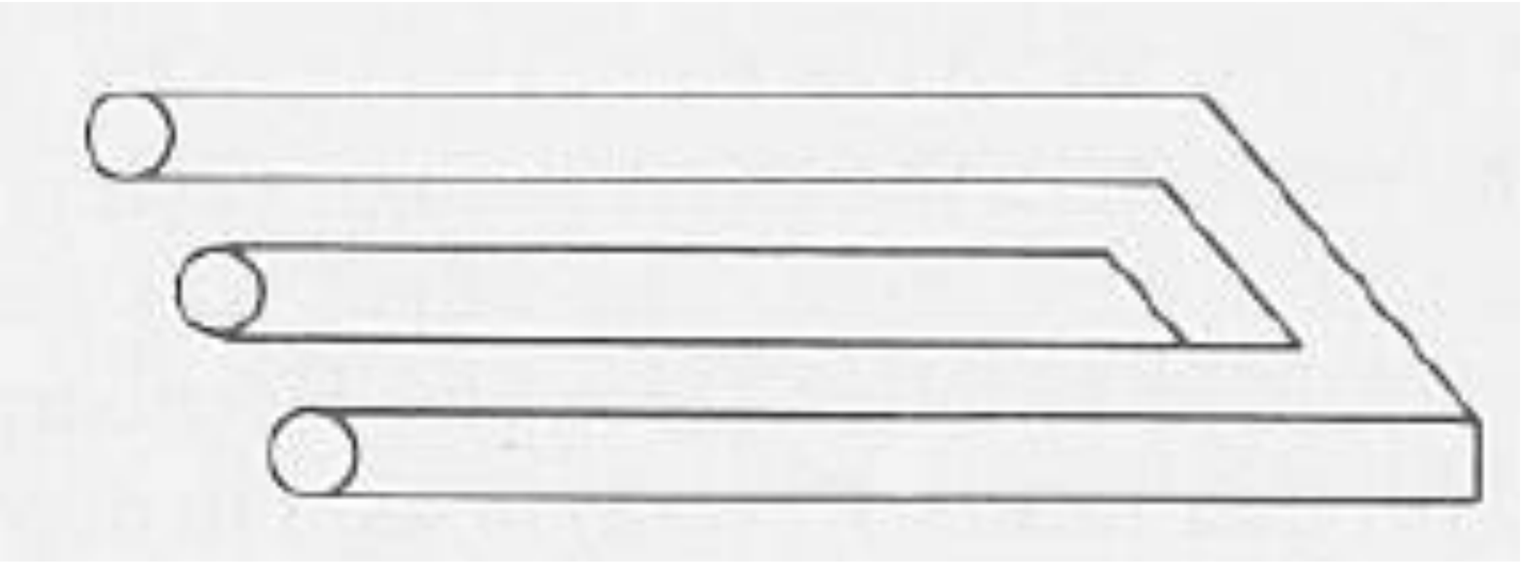
Groupings by Invisible Completions

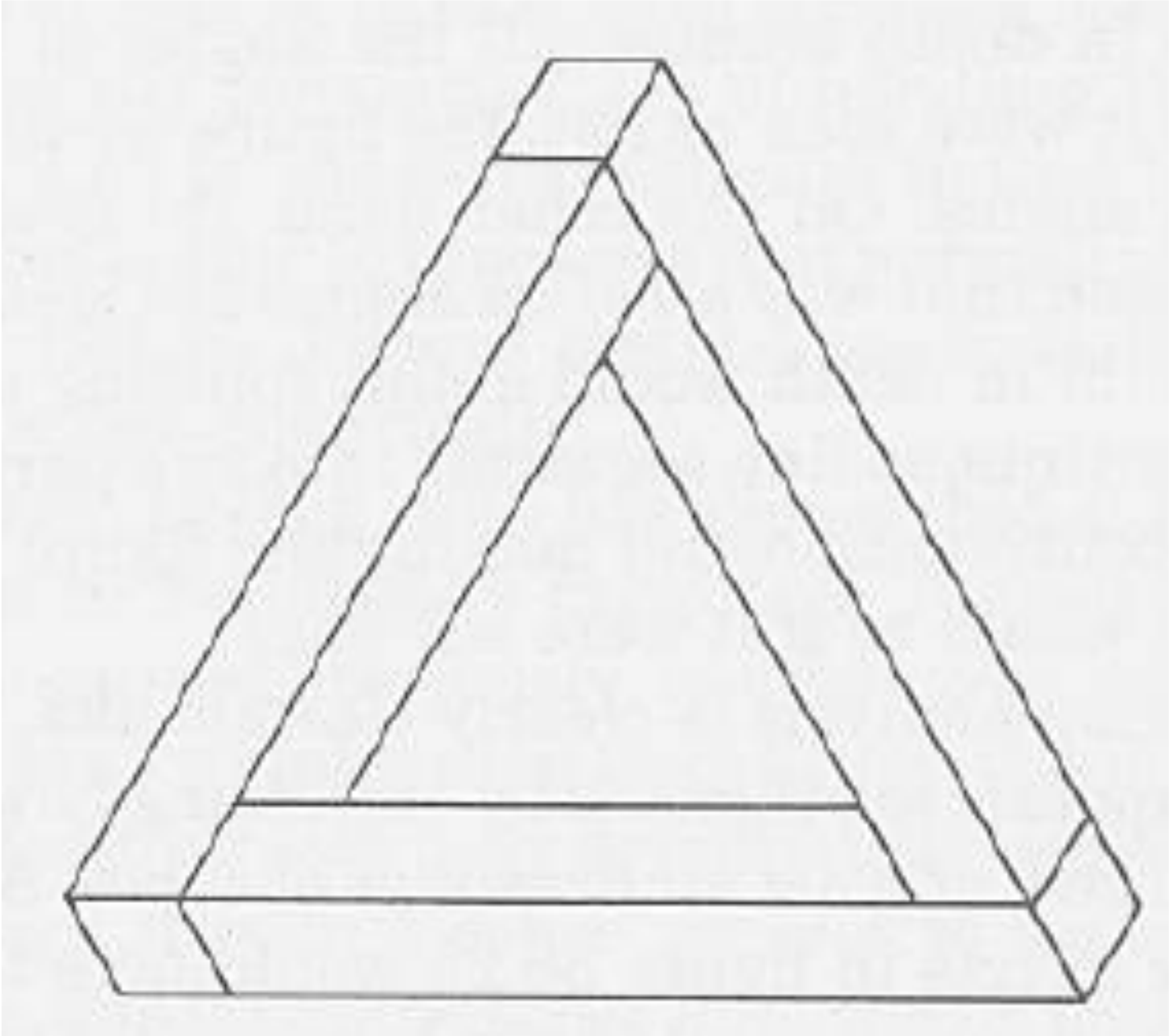


* Images from Steve Lehar’s Gestalt papers

Source: A. Oliva

… but not too global



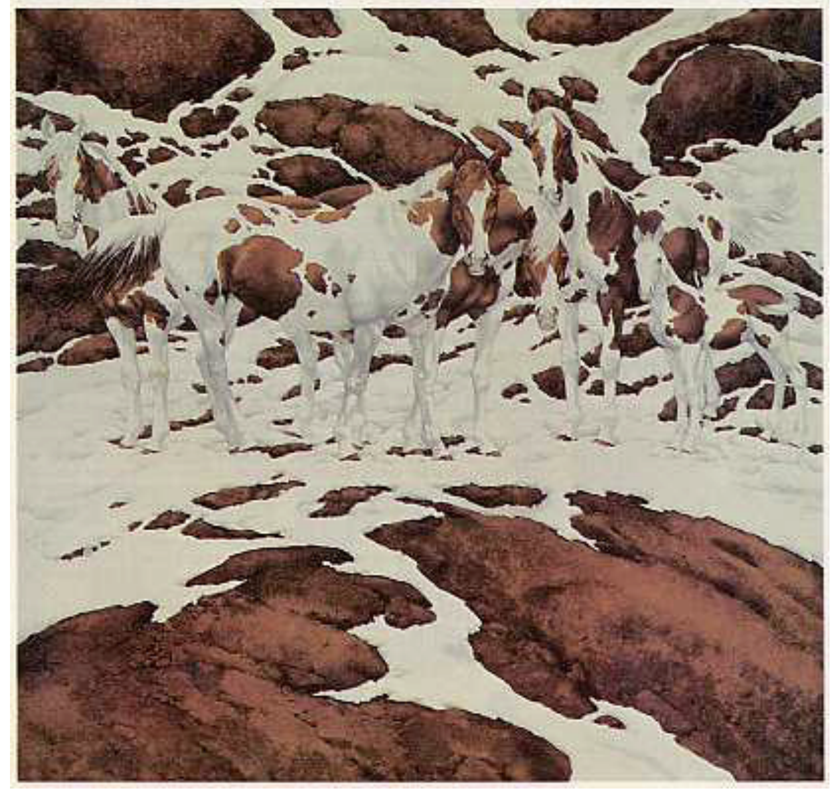


Source: A. Oliva

Photo credit: R. C. James

Perceptual organization

"…the processes by which the bits and pieces of visual information that are available in the retinal image are structured into the larger units of perceived objects and their interrelations"

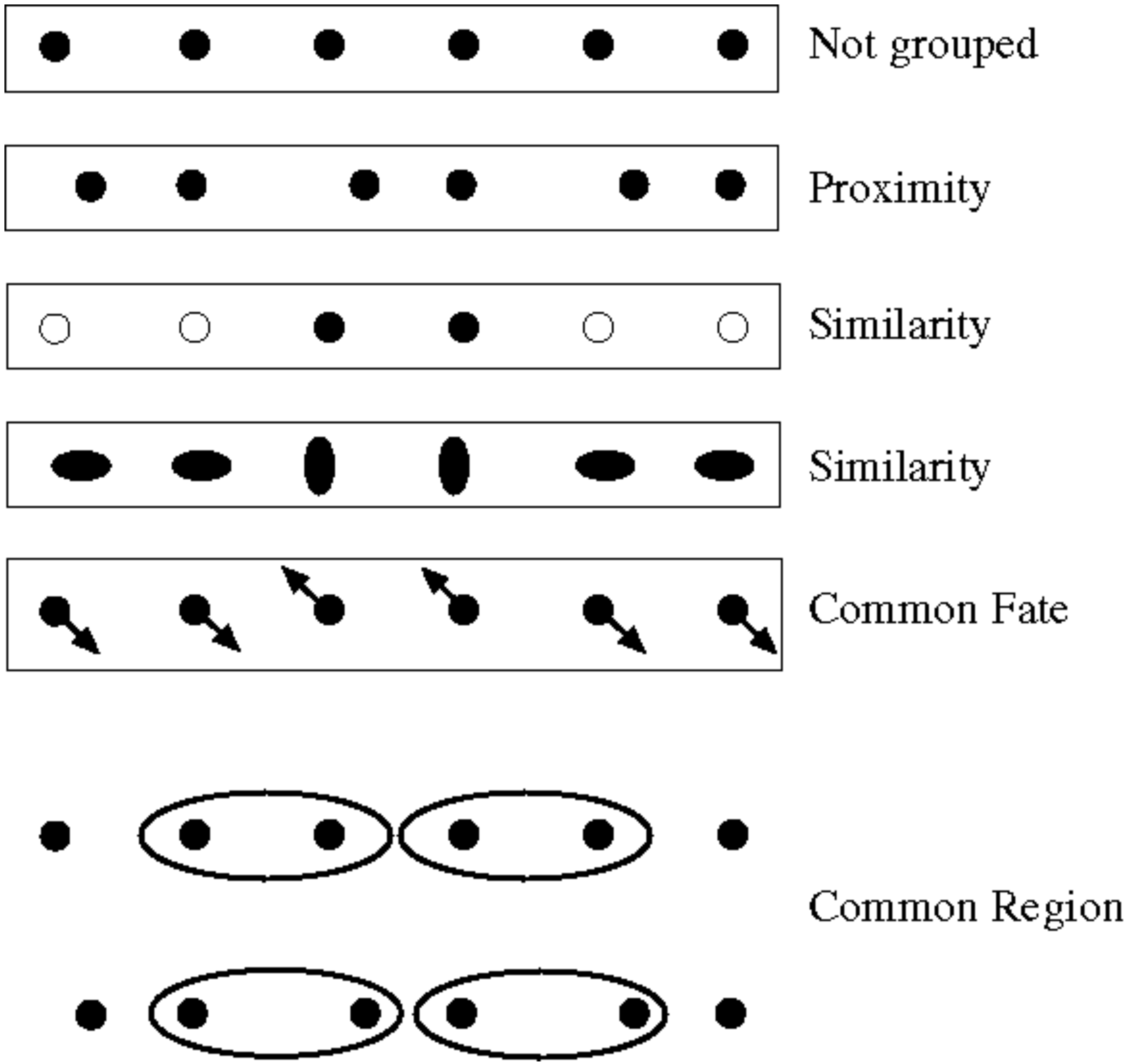


Stephen E. Palmer, *Vision Science, 1999*

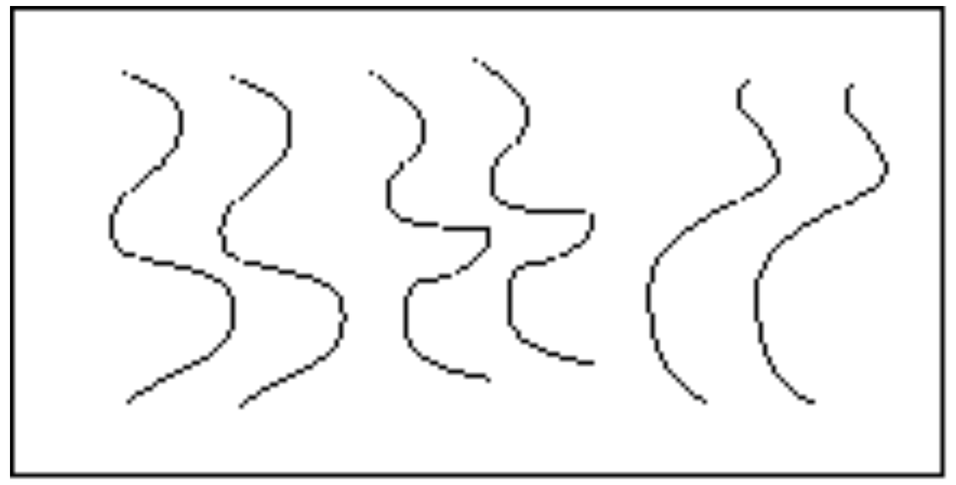
Source: A. Oliva

Gestalt principles

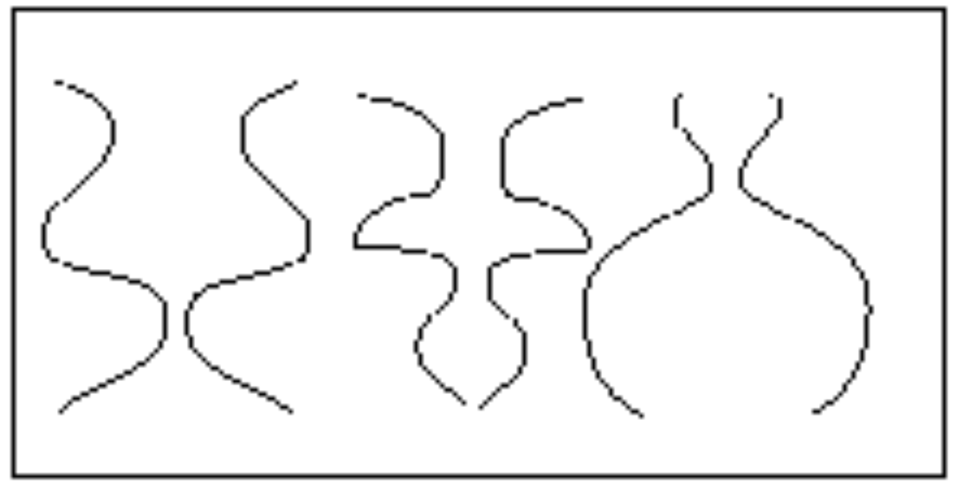
There are hundreds of different grouping laws



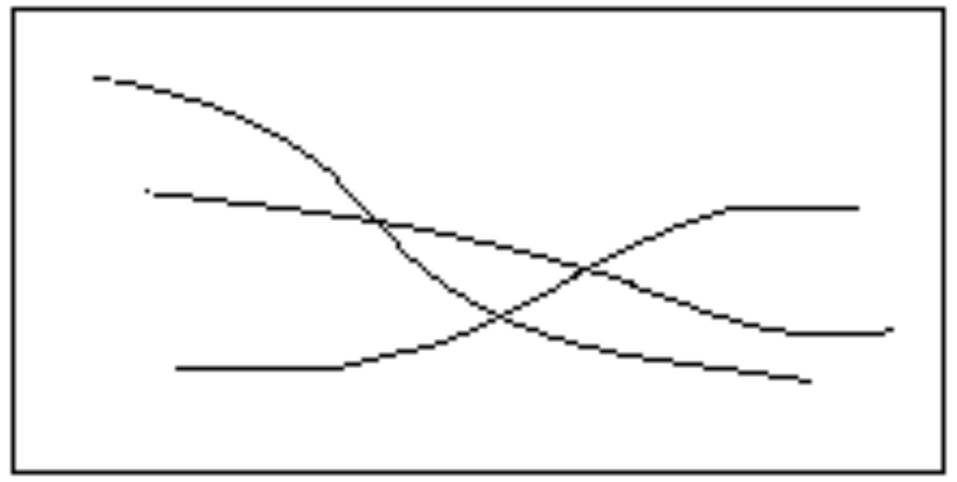
Source: A. Oliva



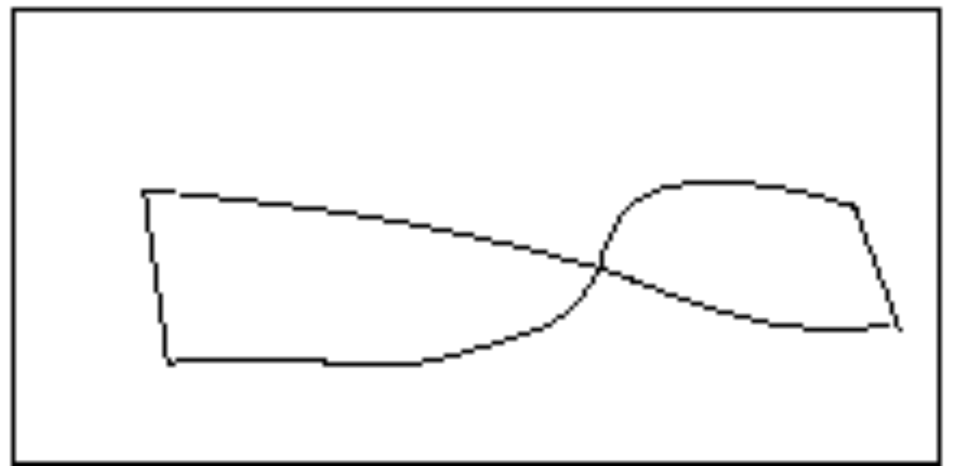
Parallelism



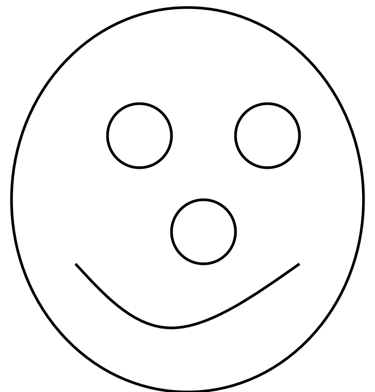
Symmetry



Continuity



Closure

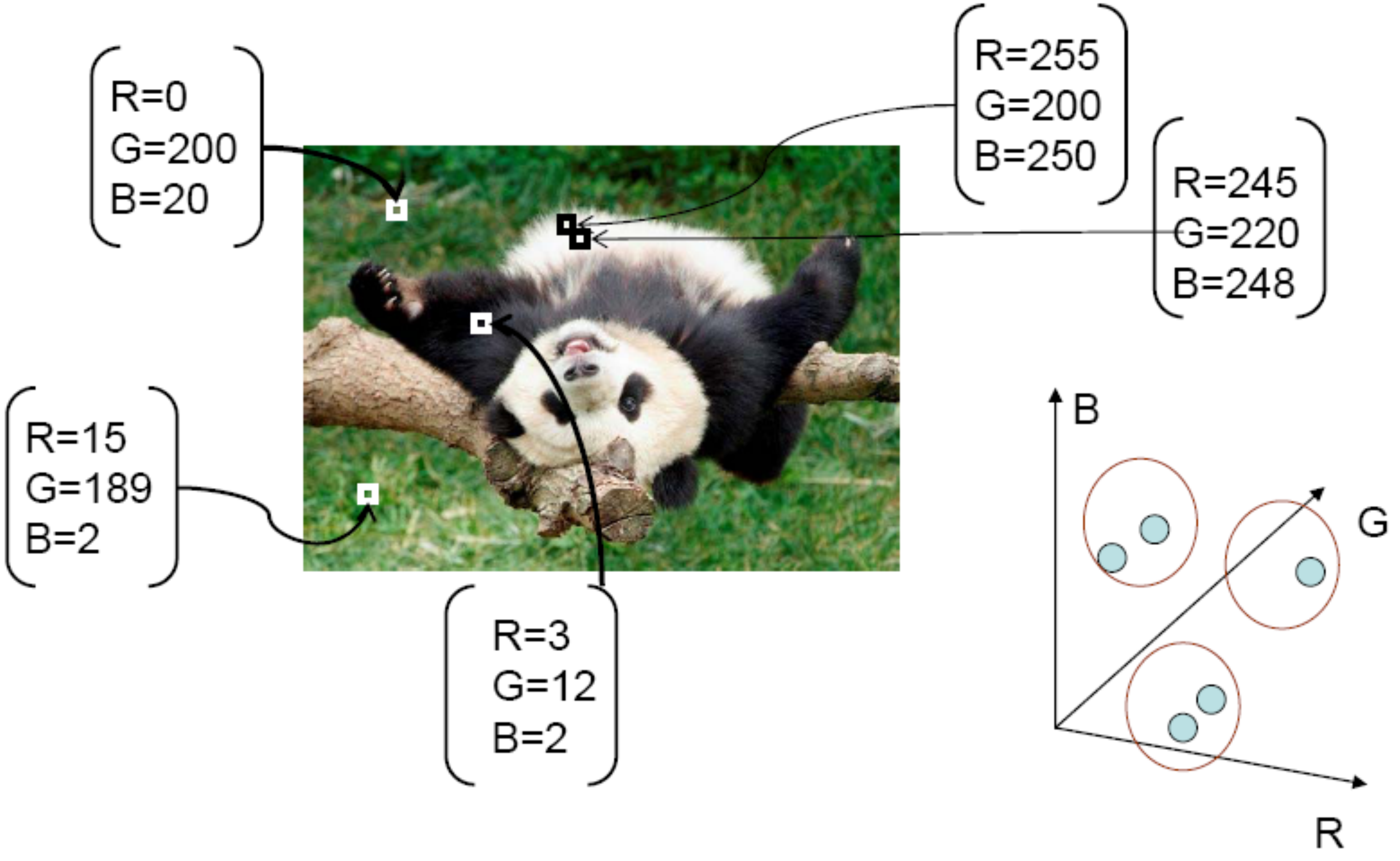


Familiar configuration

Source: A. Oliva

Clustering

- Cluster similar pixels (features) together



Source: K. Grauman

A simple segmentation algorithm

- Each pixel is described by a vector

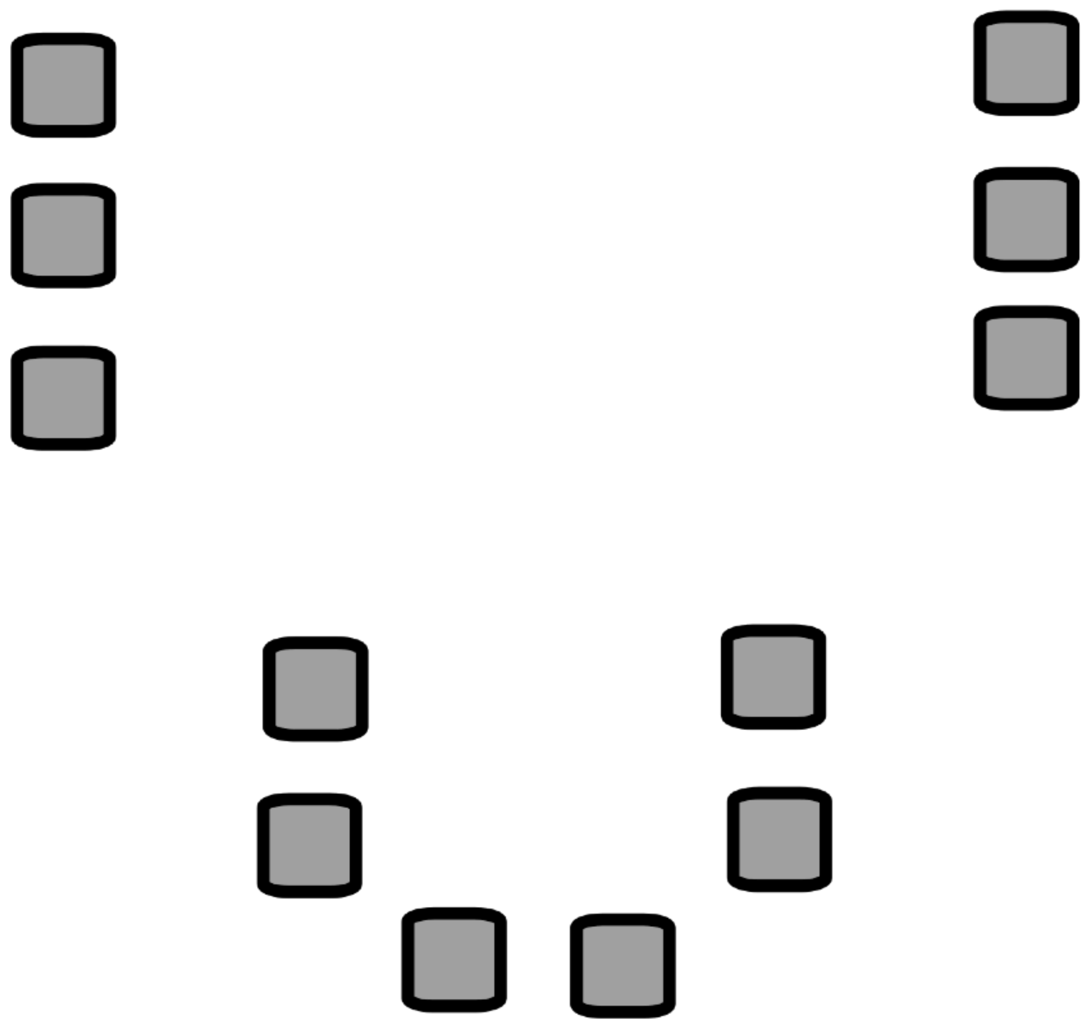
$$z = [r, g, b] \text{ or } [Y\ u\ v], \ldots$$

- Run a clustering algorithm (e.g. k-means) using some distance between pixels:

$$D(\text{pixel}_i, \text{pixel}_j) = || z_i - z_j ||^2$$

Source: A. Oliva

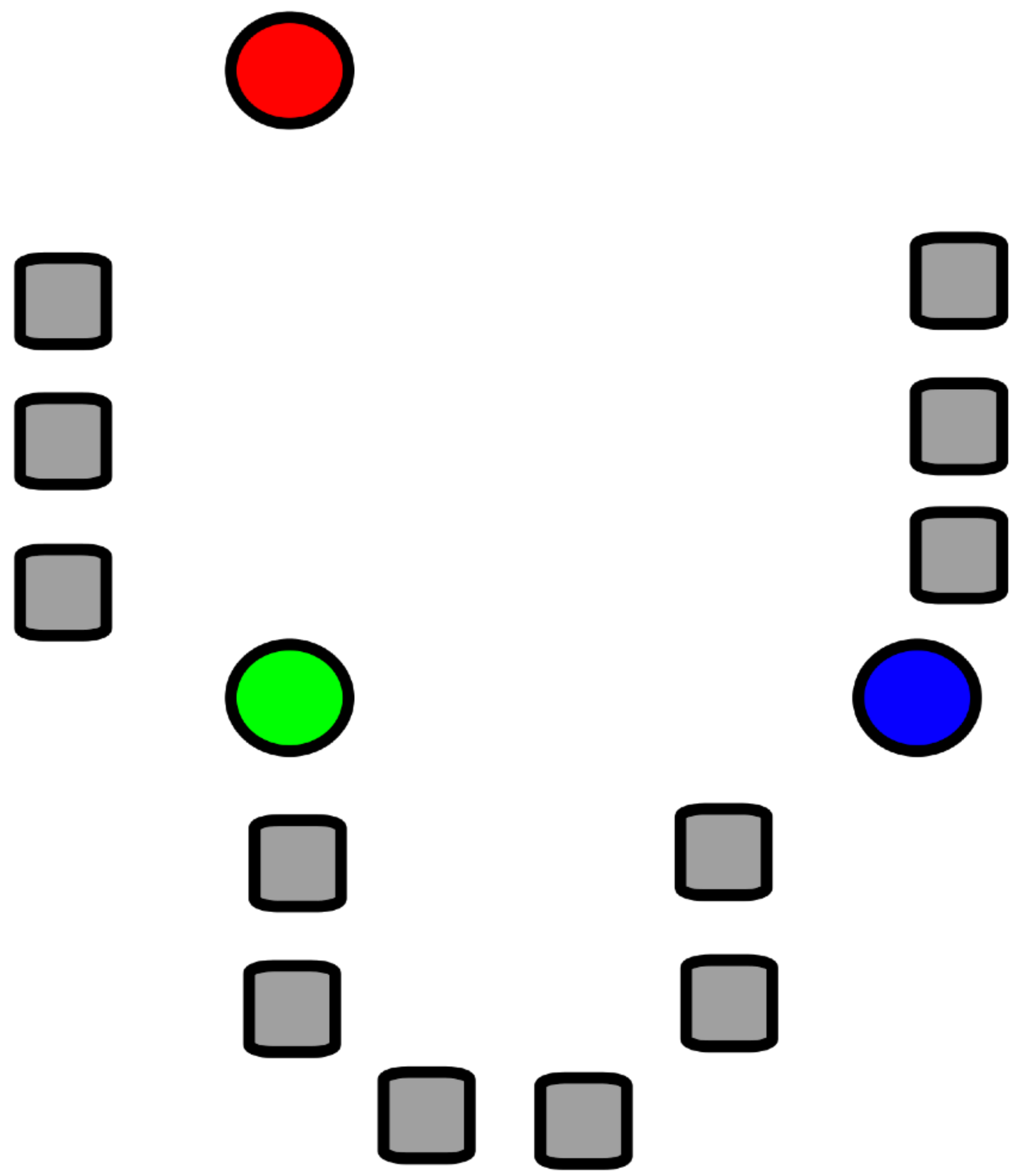
Method 1: K-Means



Idea: find K centroids that cover the points

Example from: https://en.wikipedia.org/wiki/K-means_clustering

K-Means



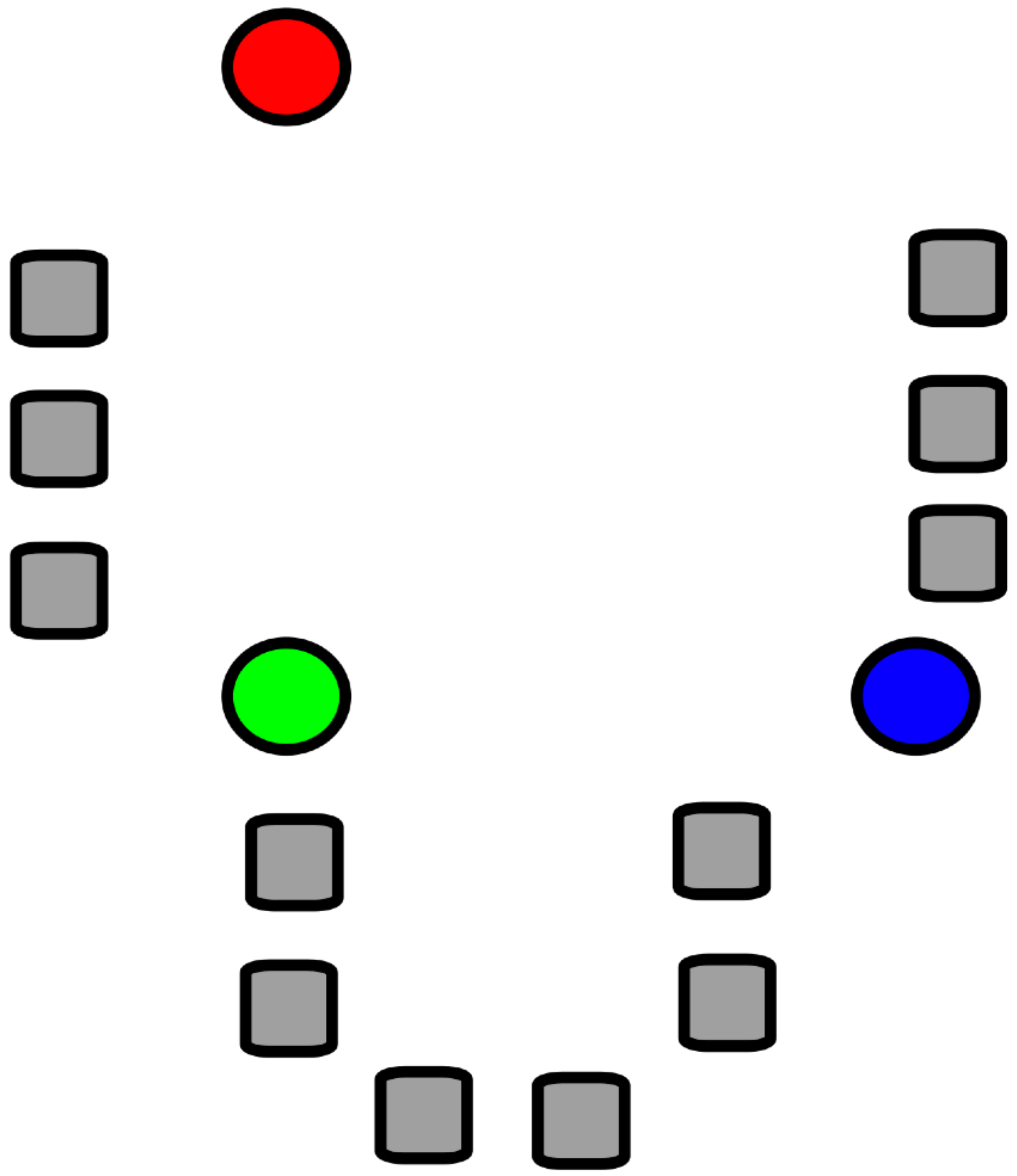
- Points are assigned to nearest centroid
- Find centroids μ_i that minimize distance to points assigned to it

Minimize:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2$$

Example from: https://en.wikipedia.org/wiki/K-means_clustering

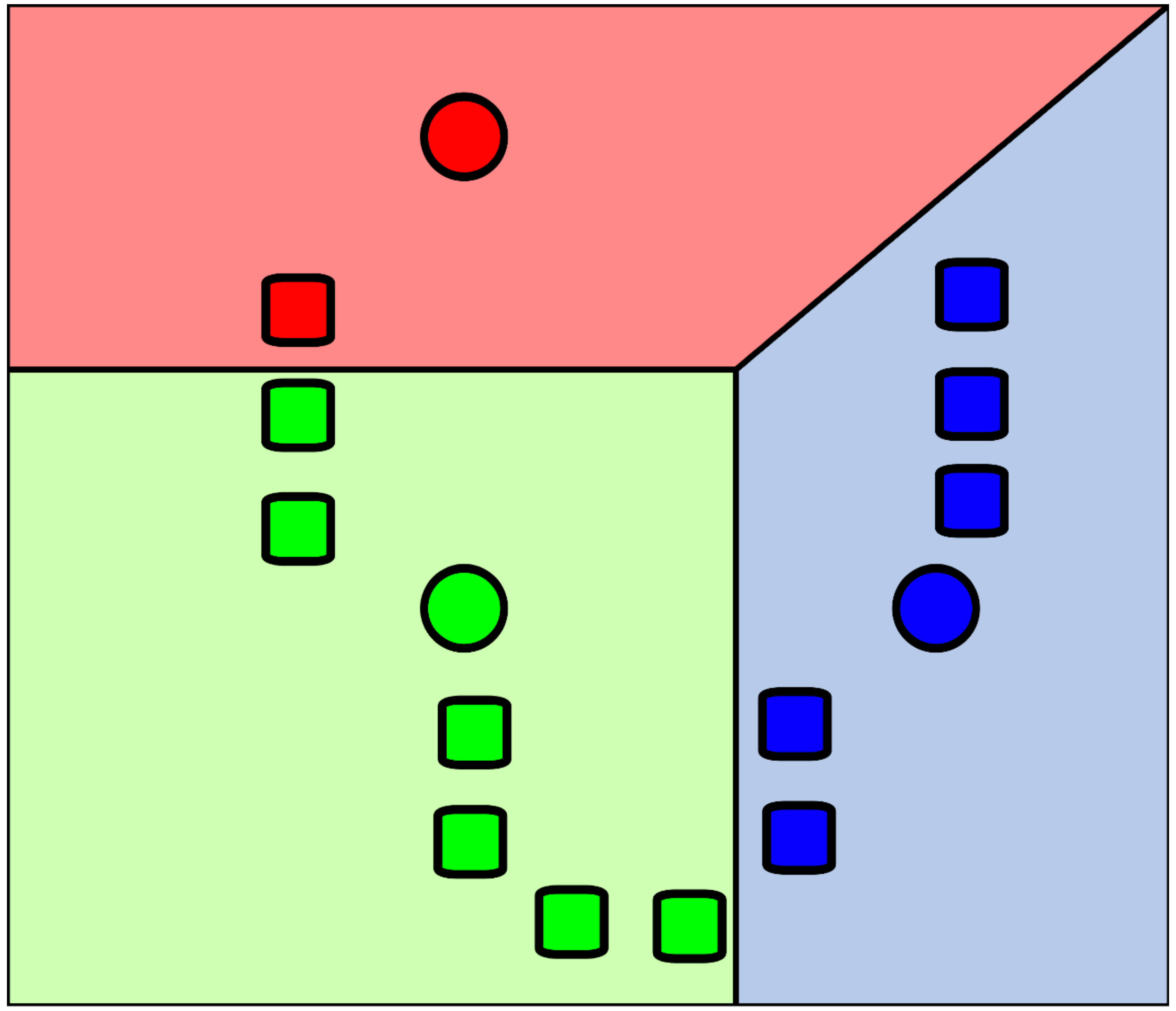
K-Means



1. Initialize centroids randomly

Example from: https://en.wikipedia.org/wiki/K-means_clustering

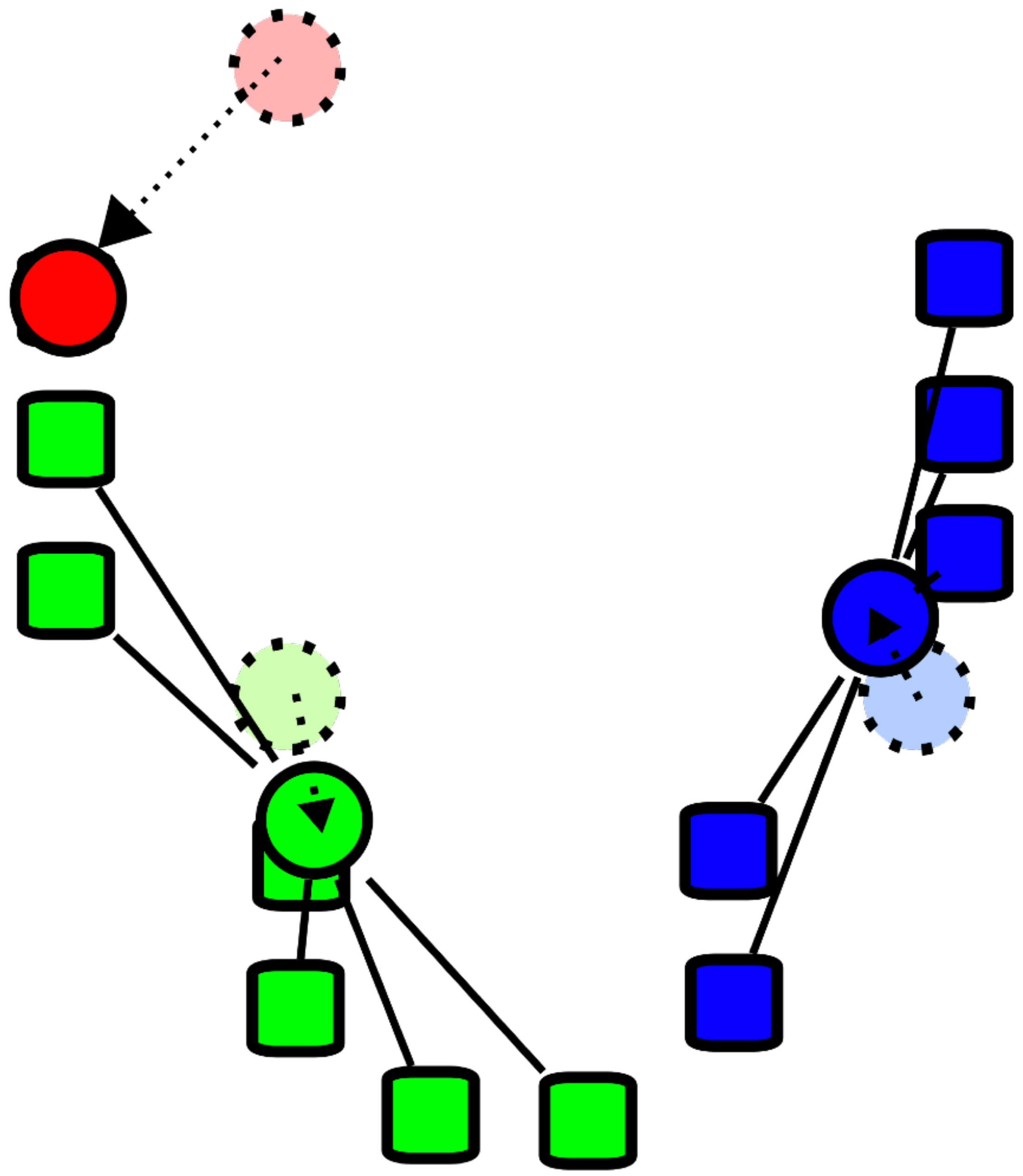
K-Means



2. Assign points to nearest cluster centroid

Example adapted from: https://en.wikipedia.org/wiki/K-means_clustering

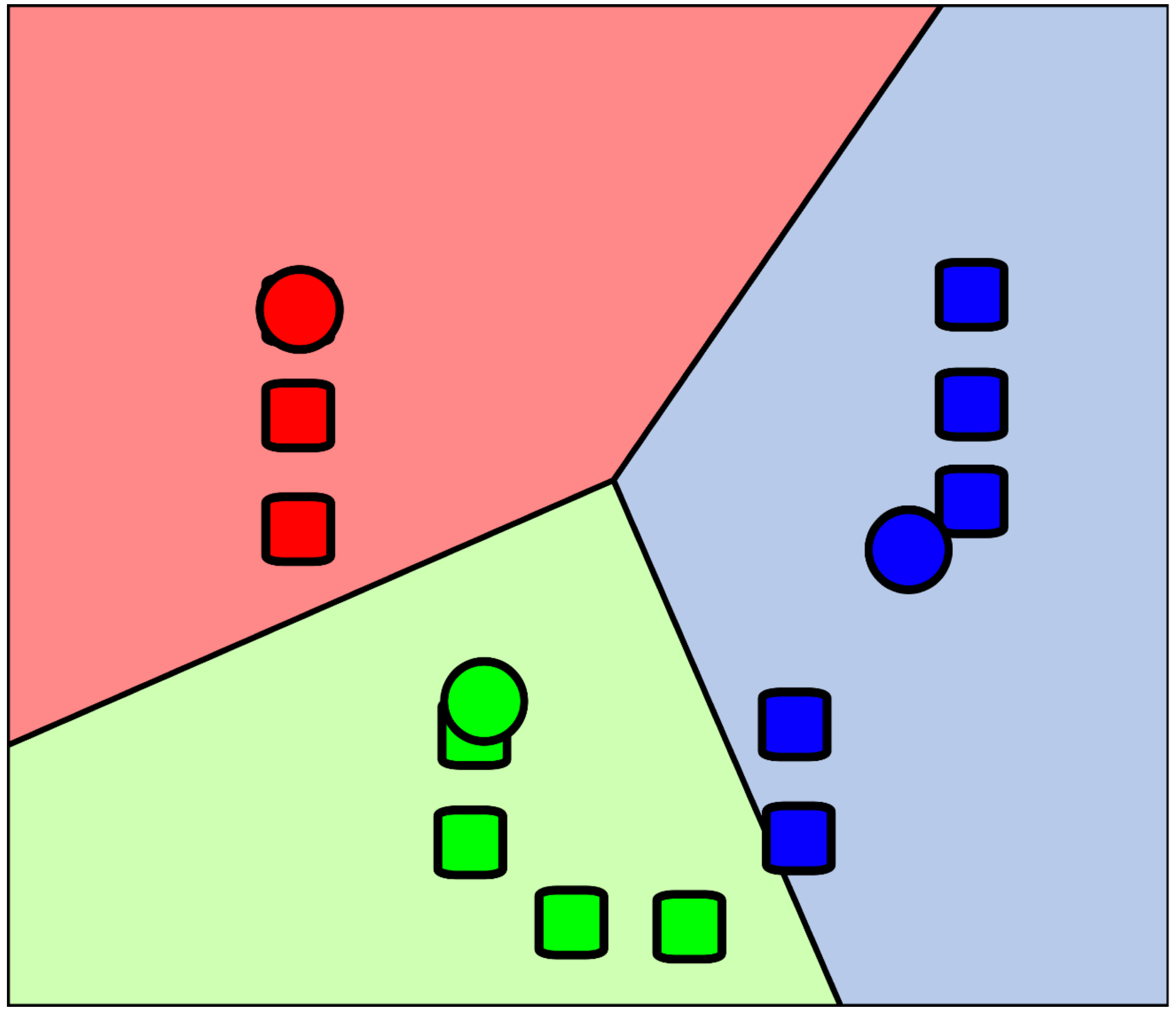
K-Means



3. Update centroids

Example adapted from: https://en.wikipedia.org/wiki/K-means_clustering

K-Means

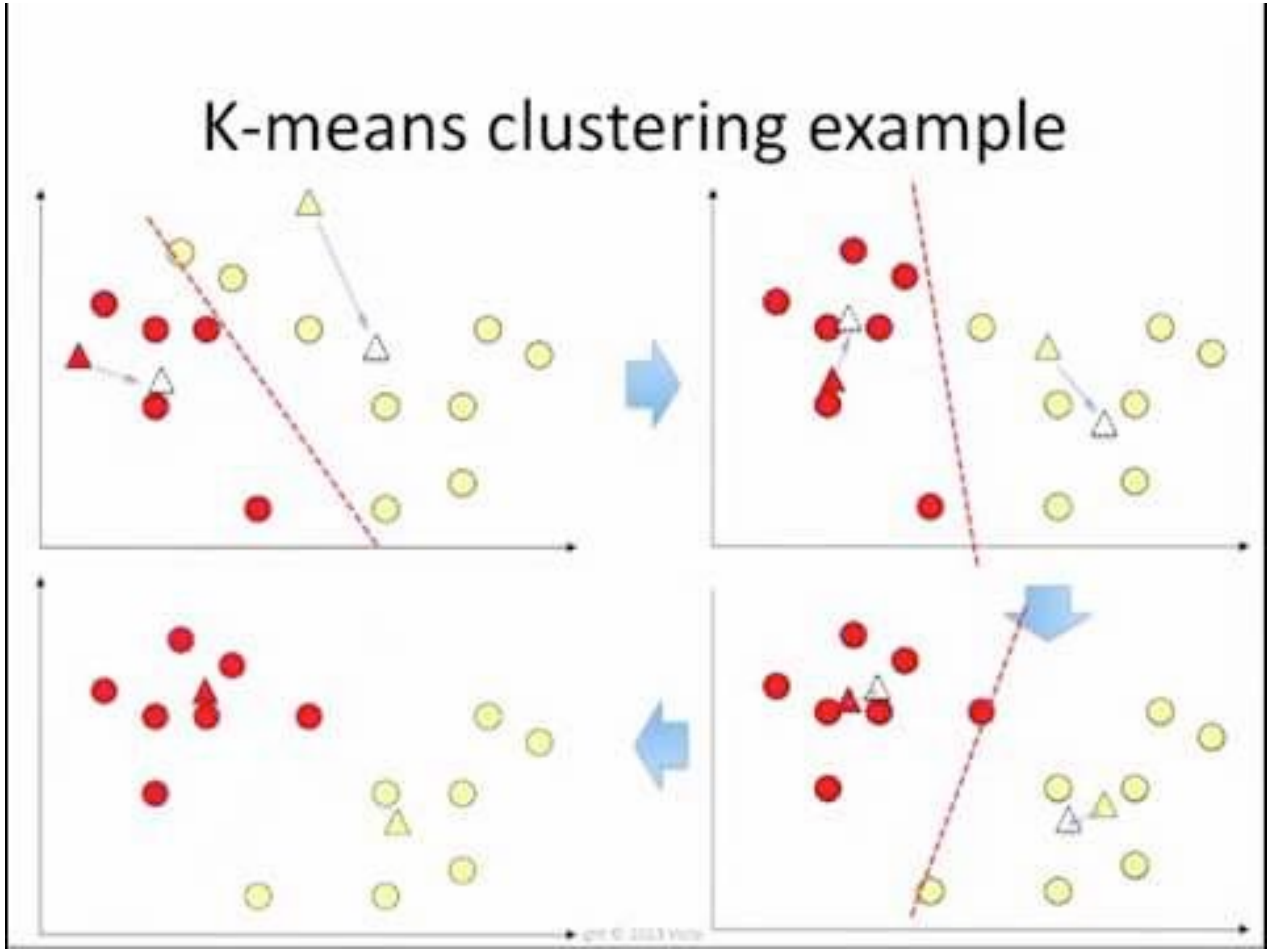


Repeat until change is very small

Example adapted from: https://en.wikipedia.org/wiki/K-means_clustering

K-Means Clustering

- Given k, the k-means algorithm consists of four steps:
  - Select initial centroids at random.
  - Assign each point to the cluster with the nearest centroid.
  - Compute each centroid as the mean of the points assigned to it.
  - Repeat previous 2 steps until no change.

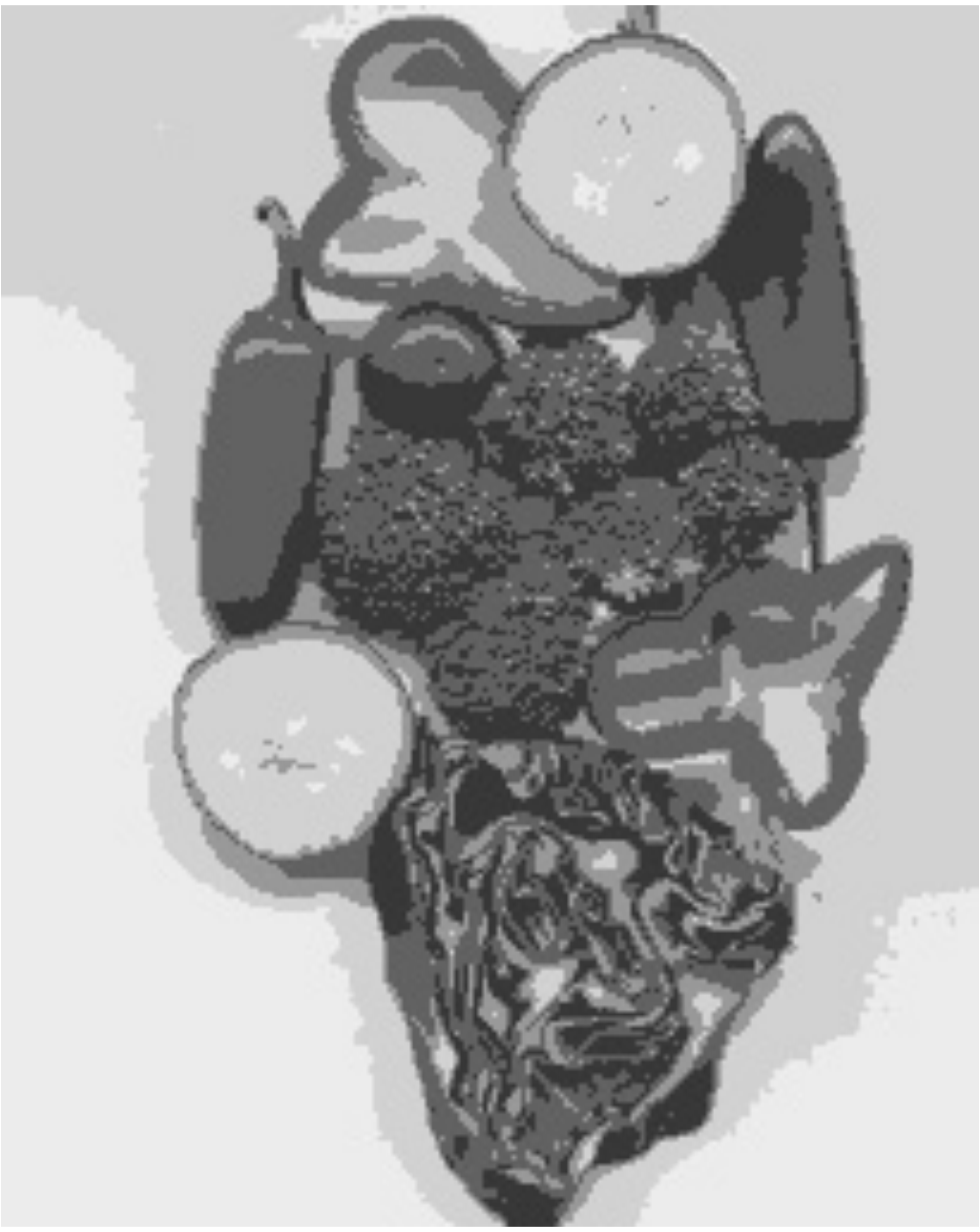


Source: A. Oliva

- K-means (k=5) clustering based on intensity (middle) or color (right) is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent

Image

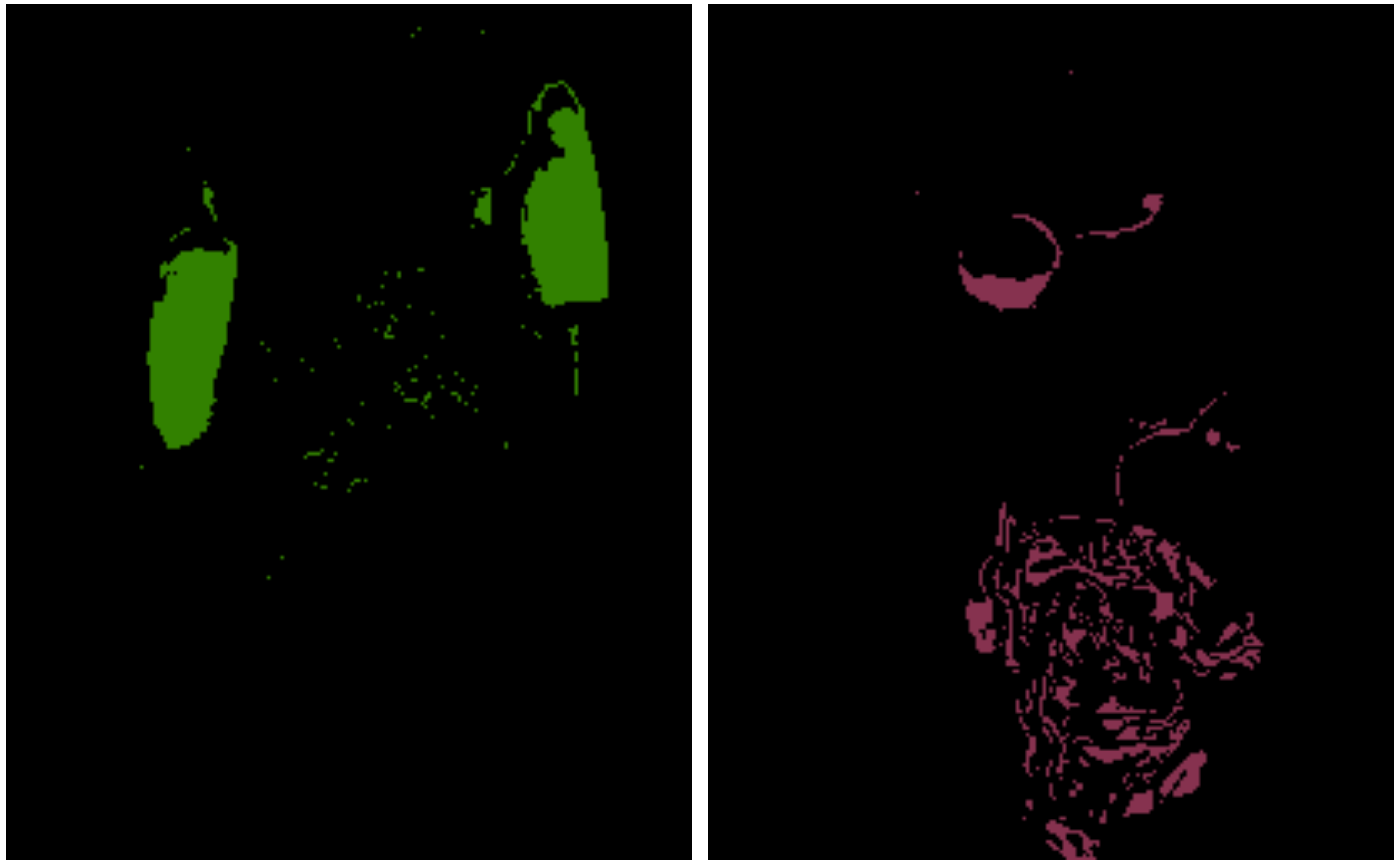
Intensity-based clusters



Color-based clusters

each pixel is replaced with the mean value of its cluster

Source: A. Oliva



K-means using
color alone
(k=11 clusters)
Showing 4 of the
segments, (not
necessarily connected)
Some are good, some
meaningless

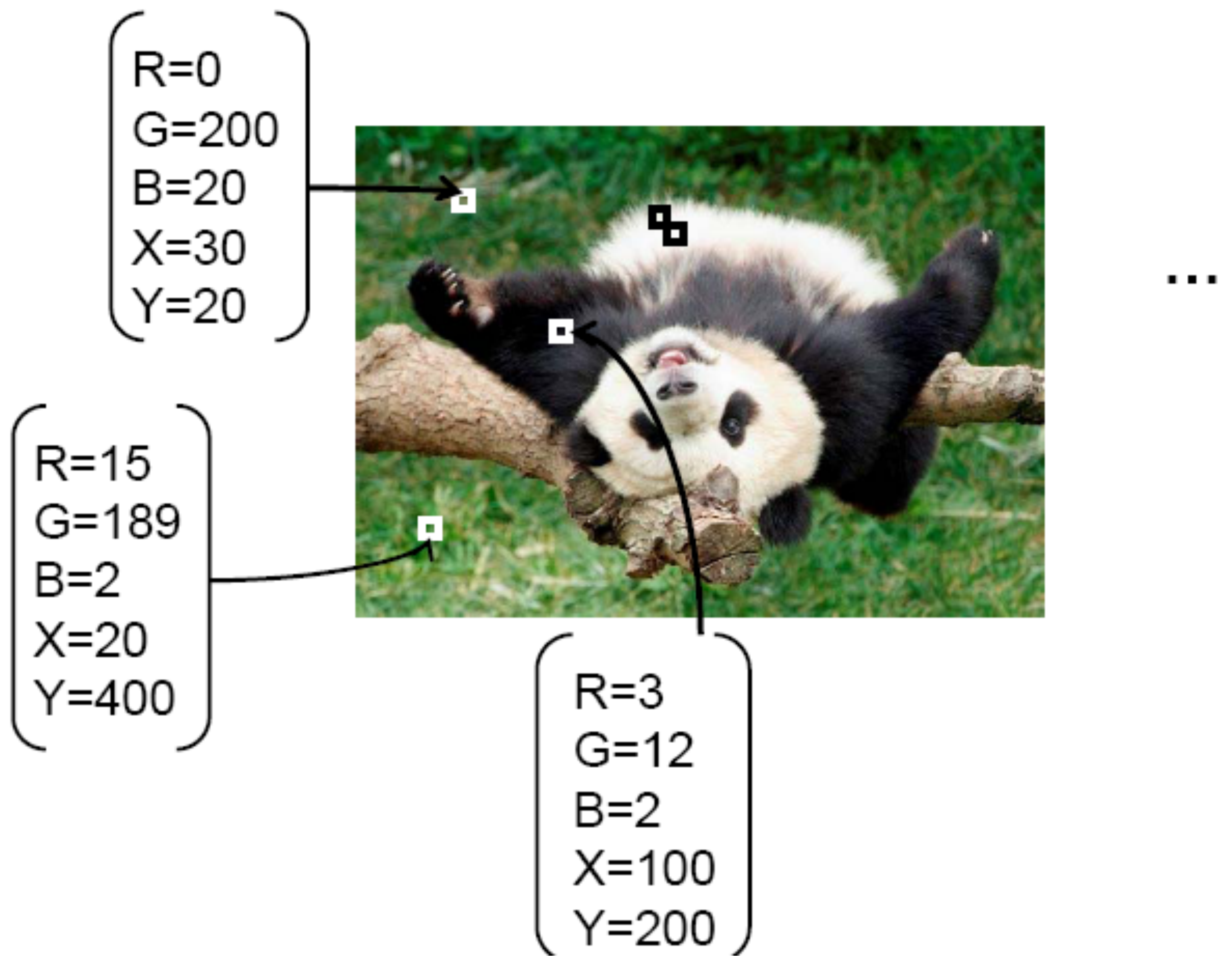
Source: A. Oliva

Including spatial relationships

Augment data to be clustered with spatial coordinates.

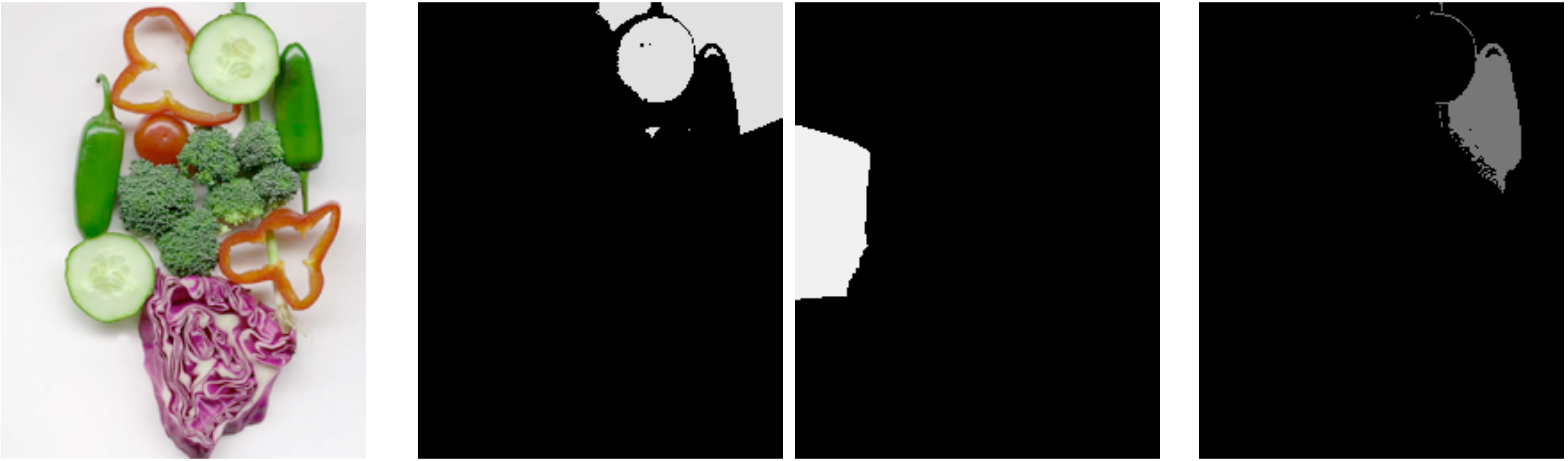
$$z = \begin{pmatrix} Y \\ u \\ v \\ x \\ y \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{color coordinates (or r,g,b)} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{spatial coordinates} \end{matrix}$$

- Cluster similar pixels (features) together



Source: A. Oliva

- Clustering based on (r,g,b,x,y) values enforces more spatial coherence



K-means using color and position, 20 segments

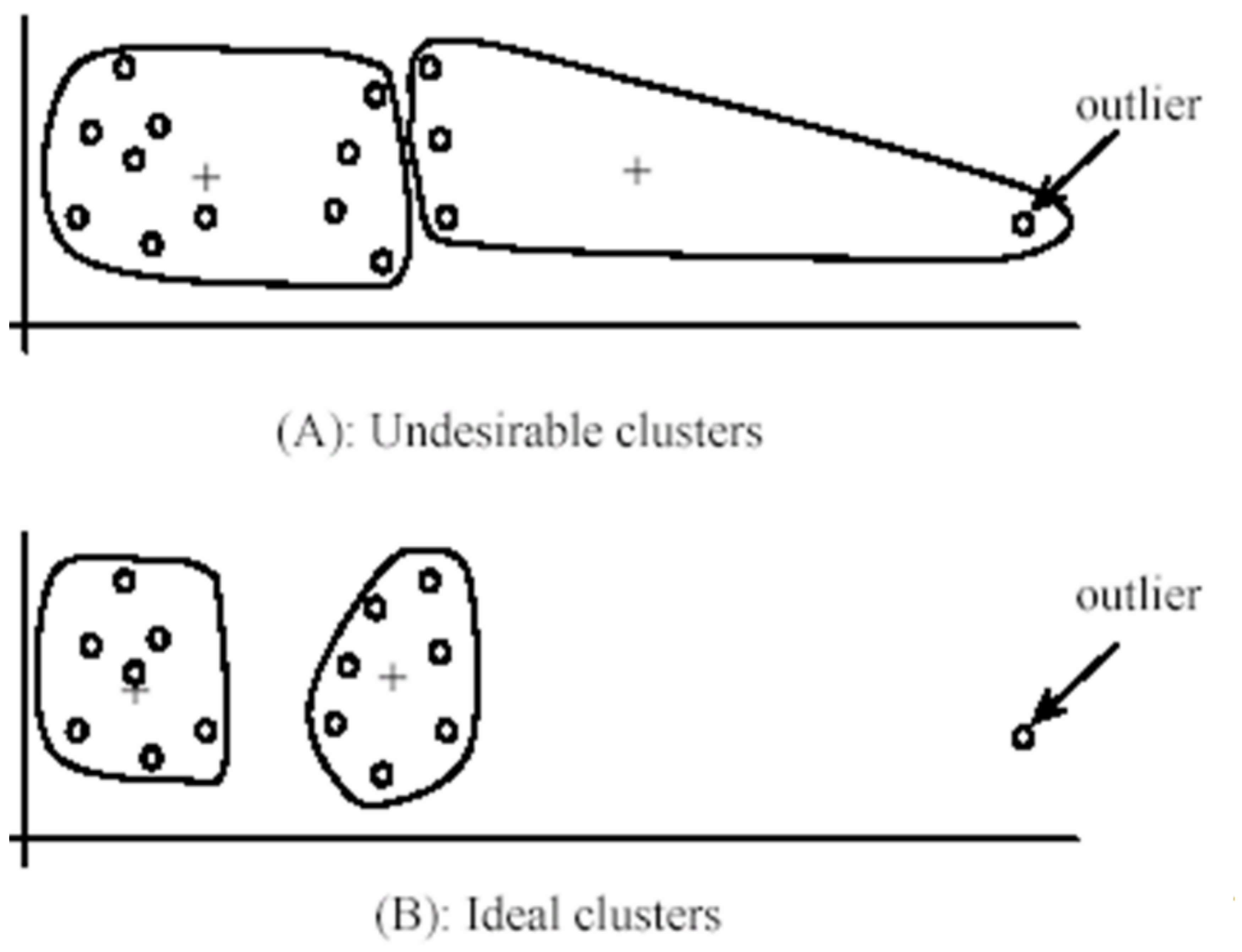
Still misses goal of perceptually pleasing or useful segmentation
No measure of texture

Hard to pick K…

Source: A. Oliva

K-Means for segmentation

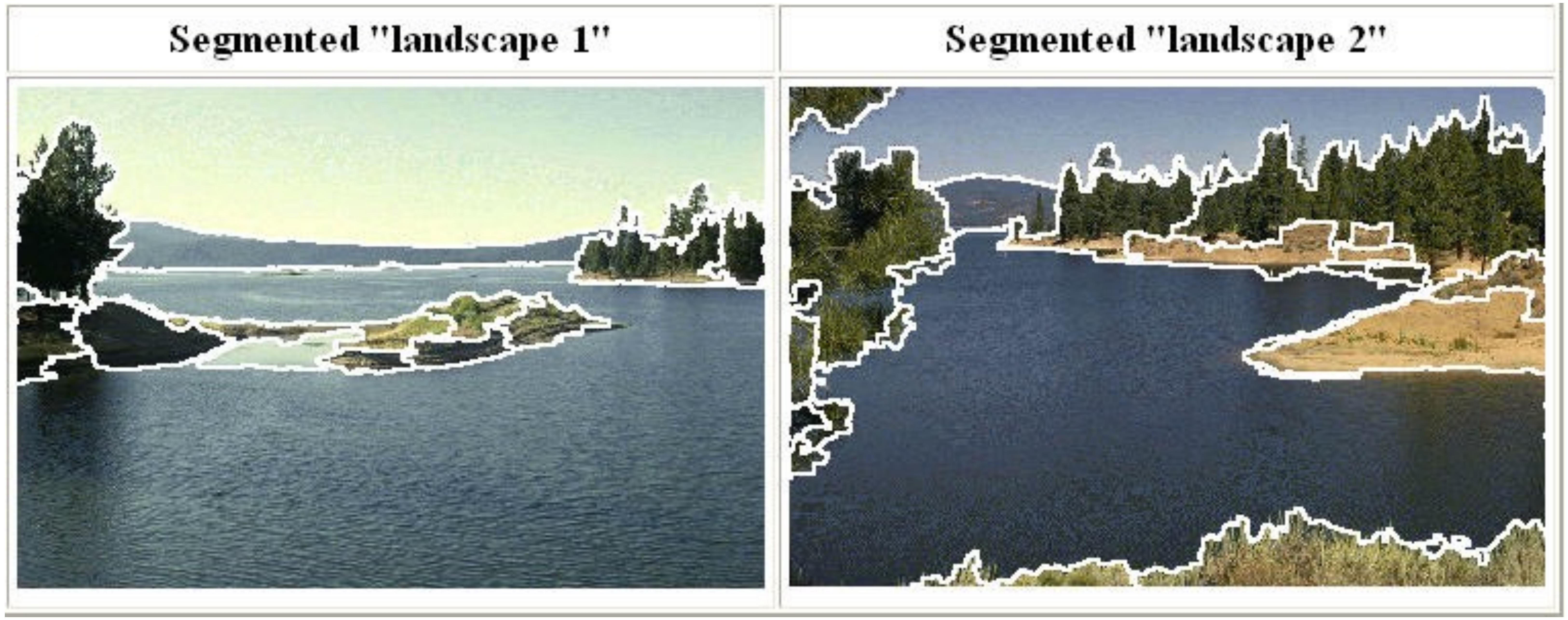
- Pros
  - Very simple method
  - Converges to a local minimum of the error function
- Cons
  - Memory-intensive
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Parametric



Source: S. Lazebnik

Method 2: Mean shift clustering

- A versatile technique for clustering-based segmentation



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

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

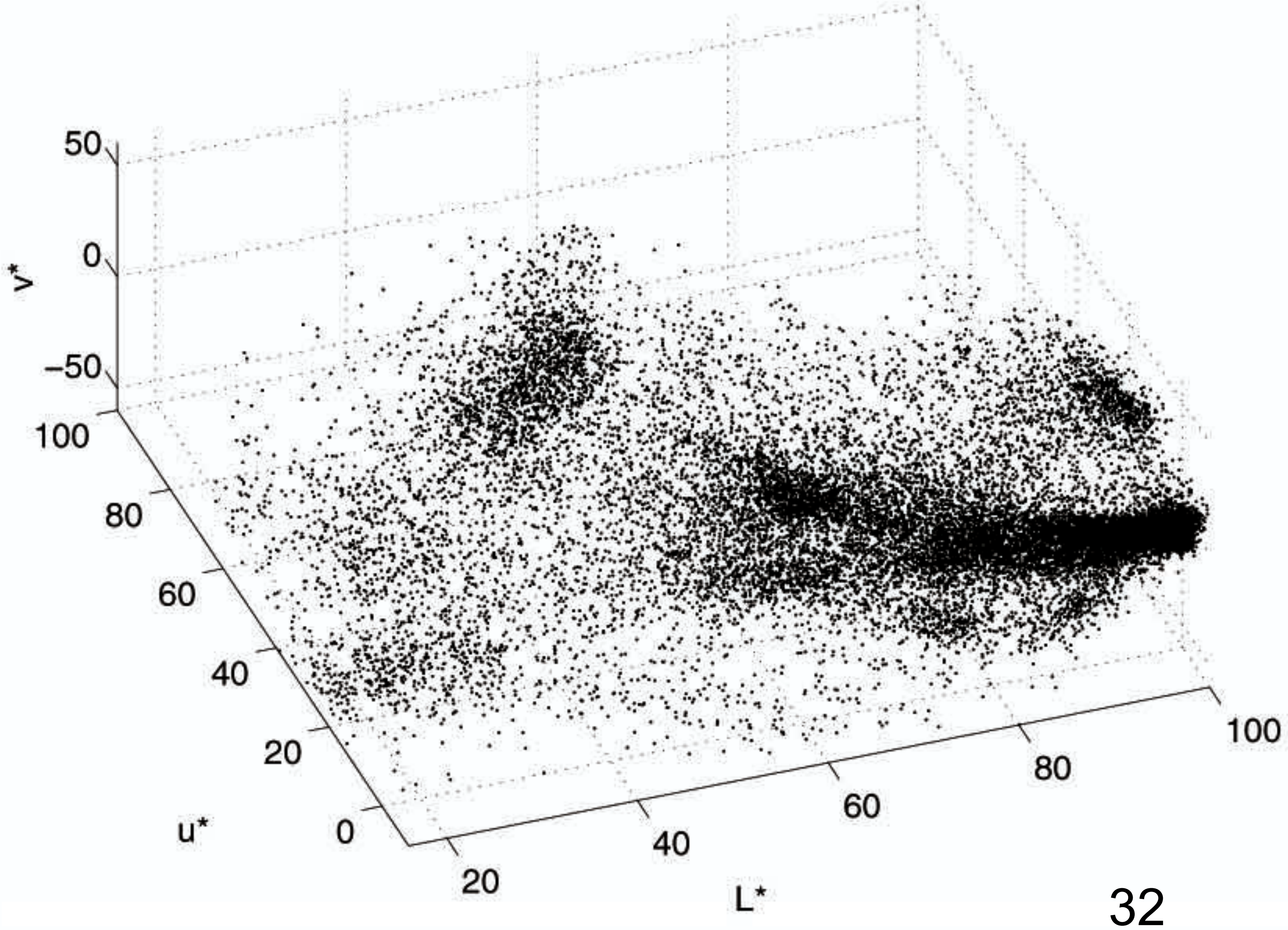
Source: A. Oliva

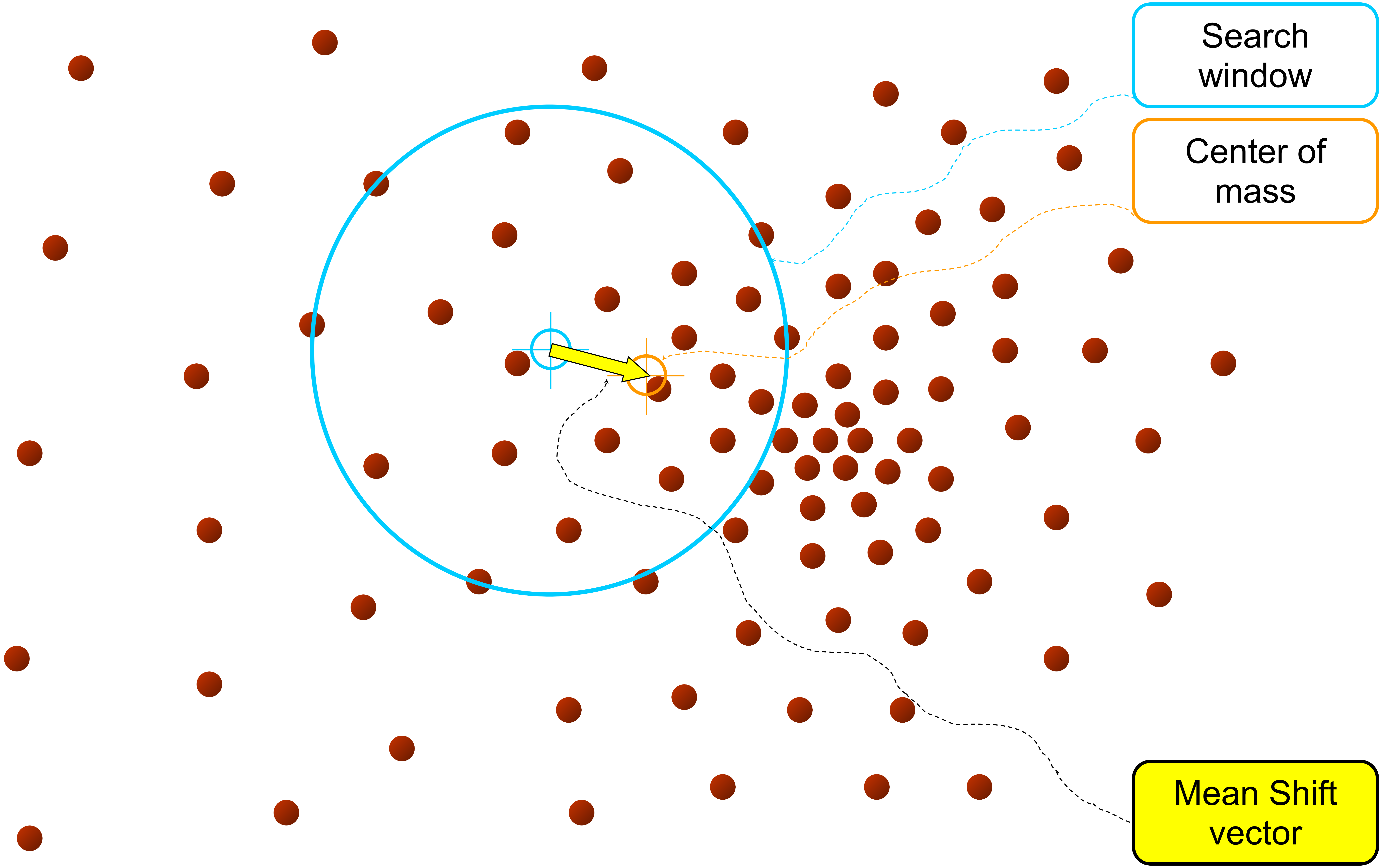
Mean shift mode-finding algorithm

The mean shift algorithm seeks modes or local maxima of density in the feature space

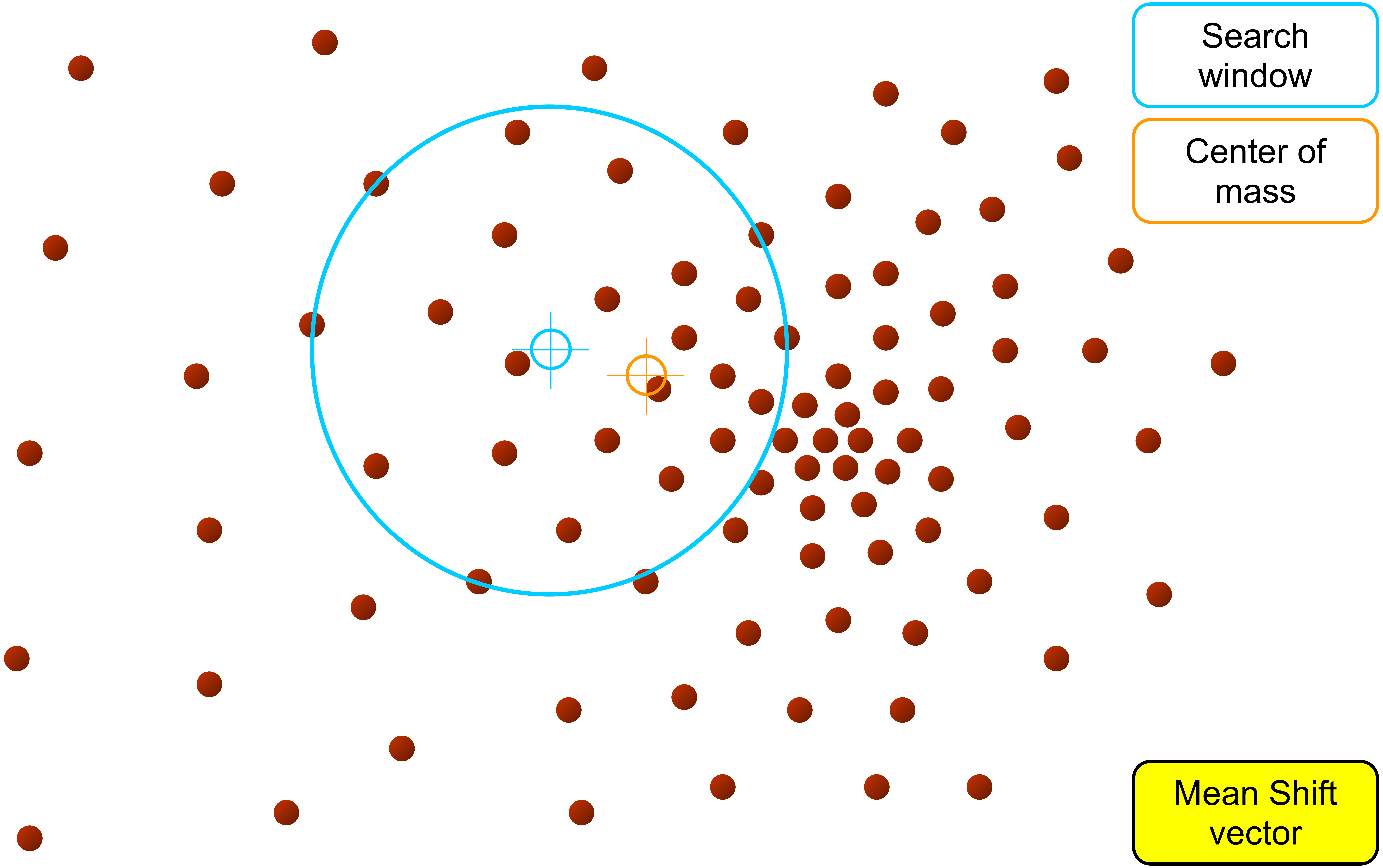
image

Feature space (L*u*v* color values)

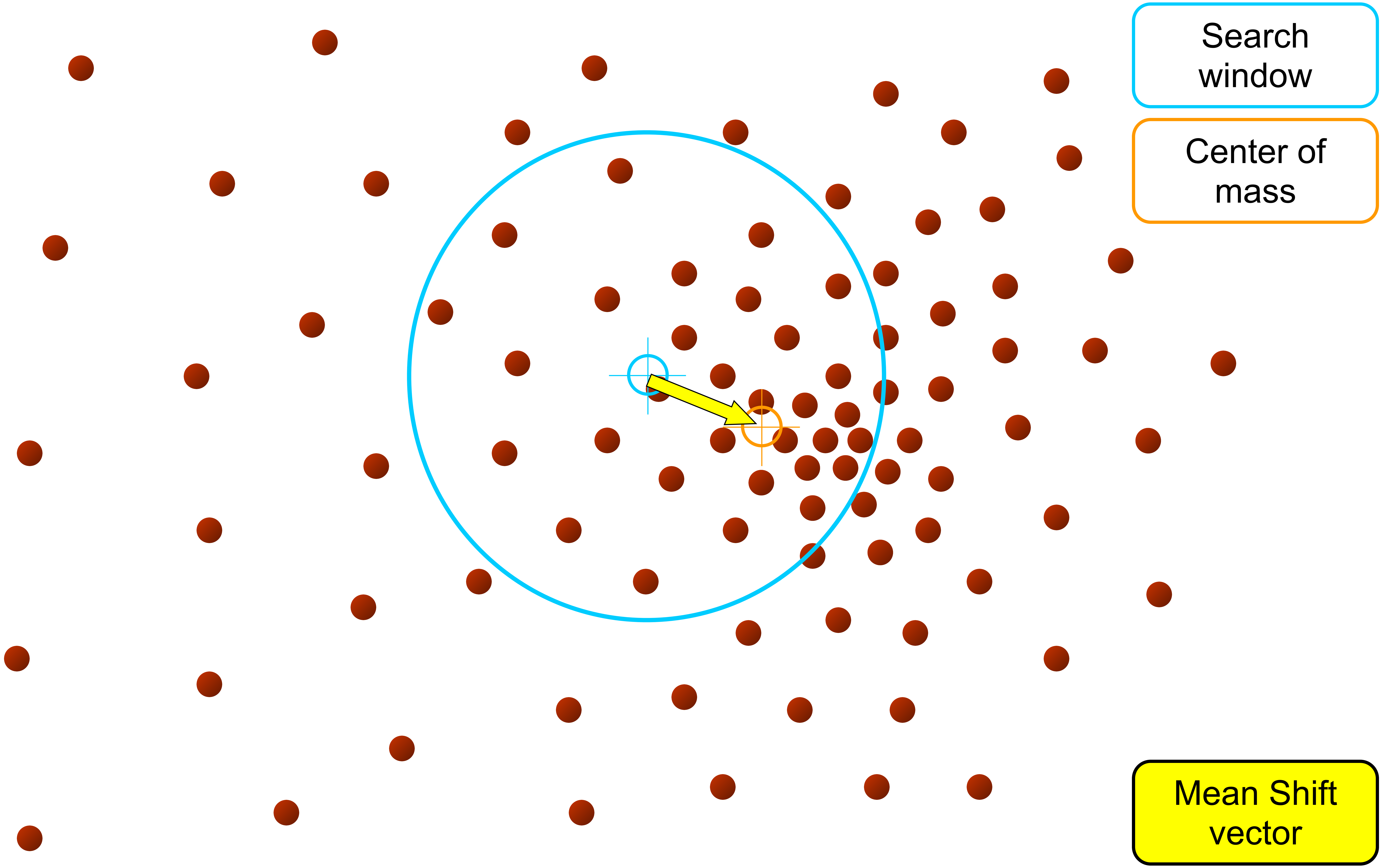




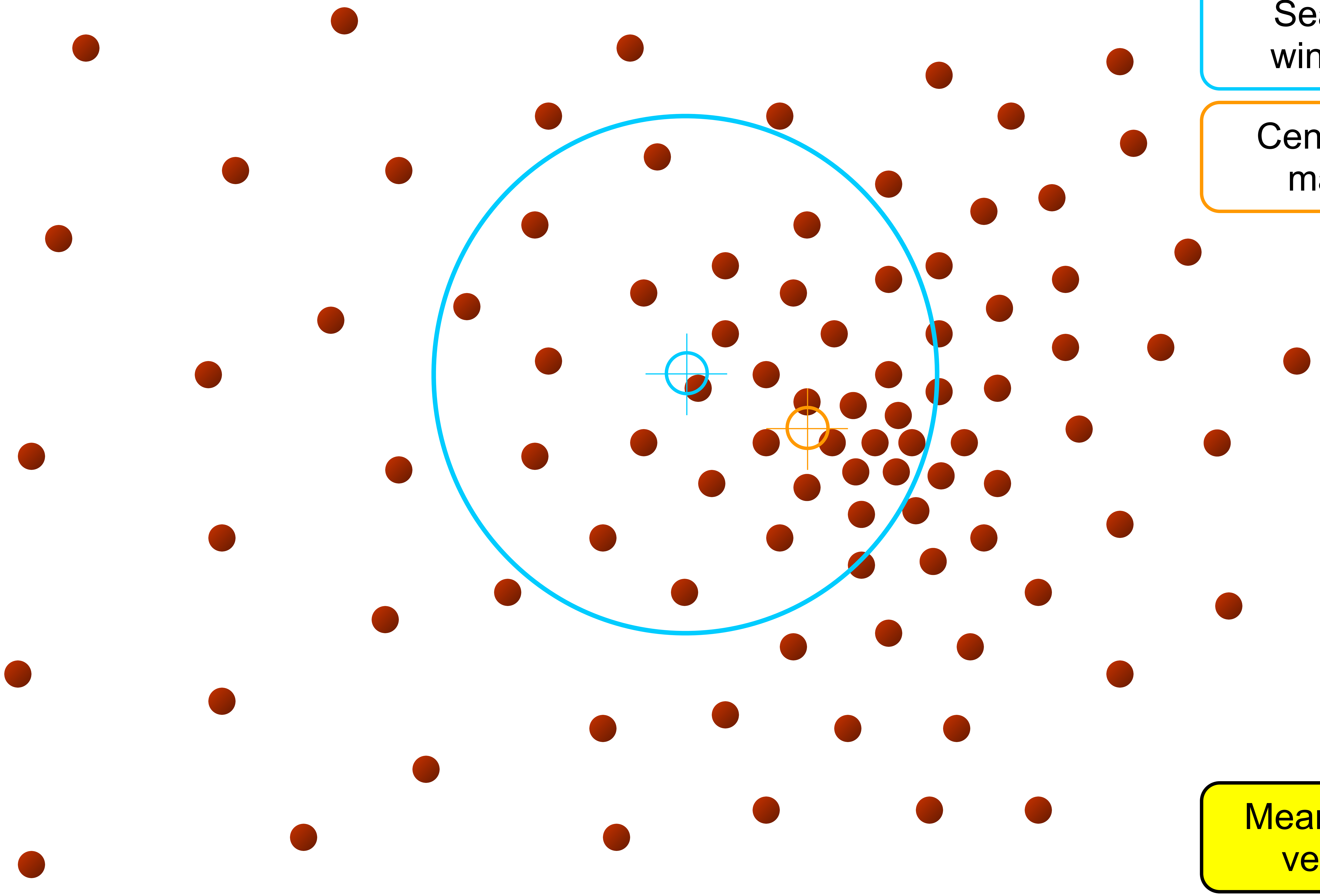
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel

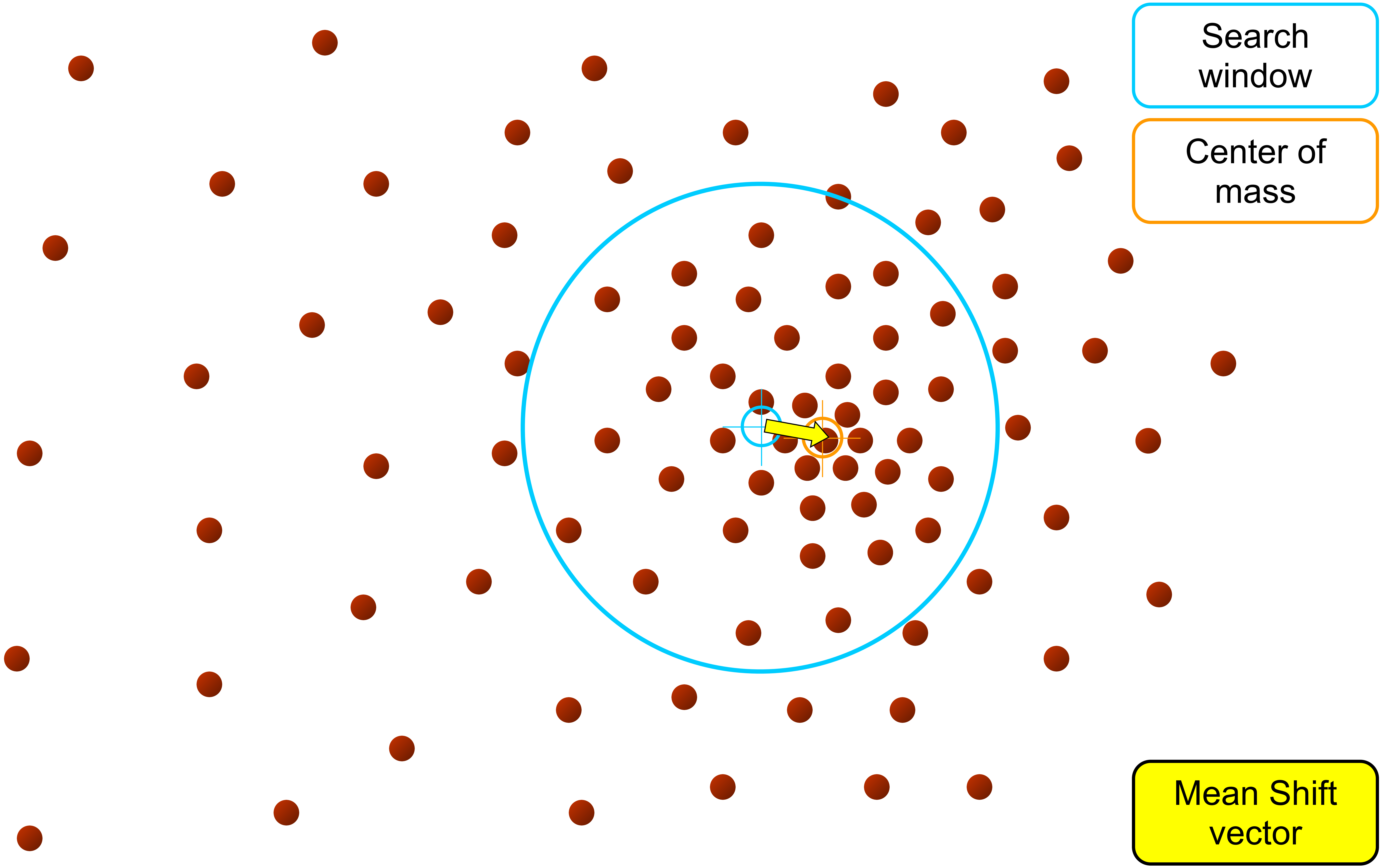


Search window

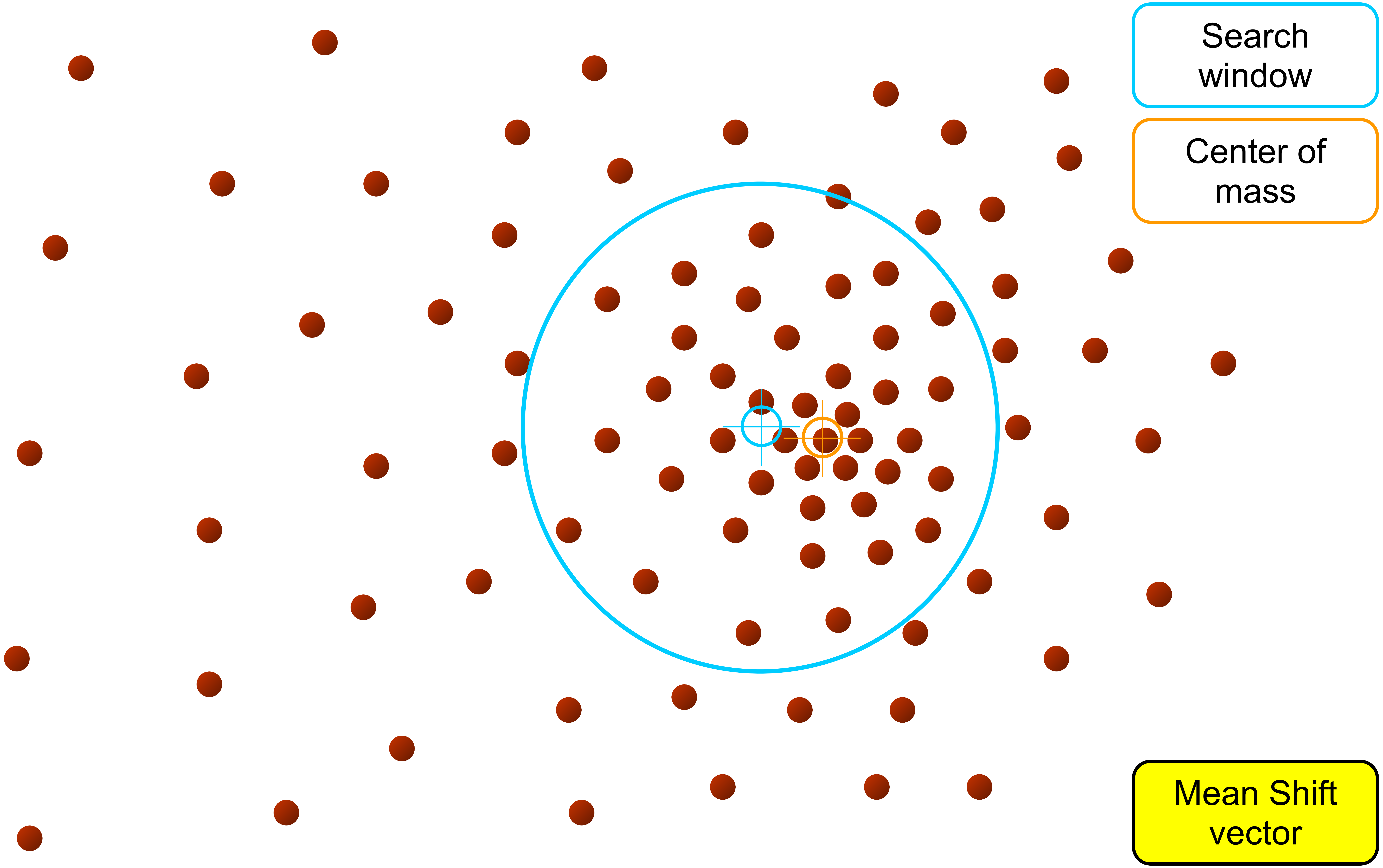
Center of mass

Mean Shift vector

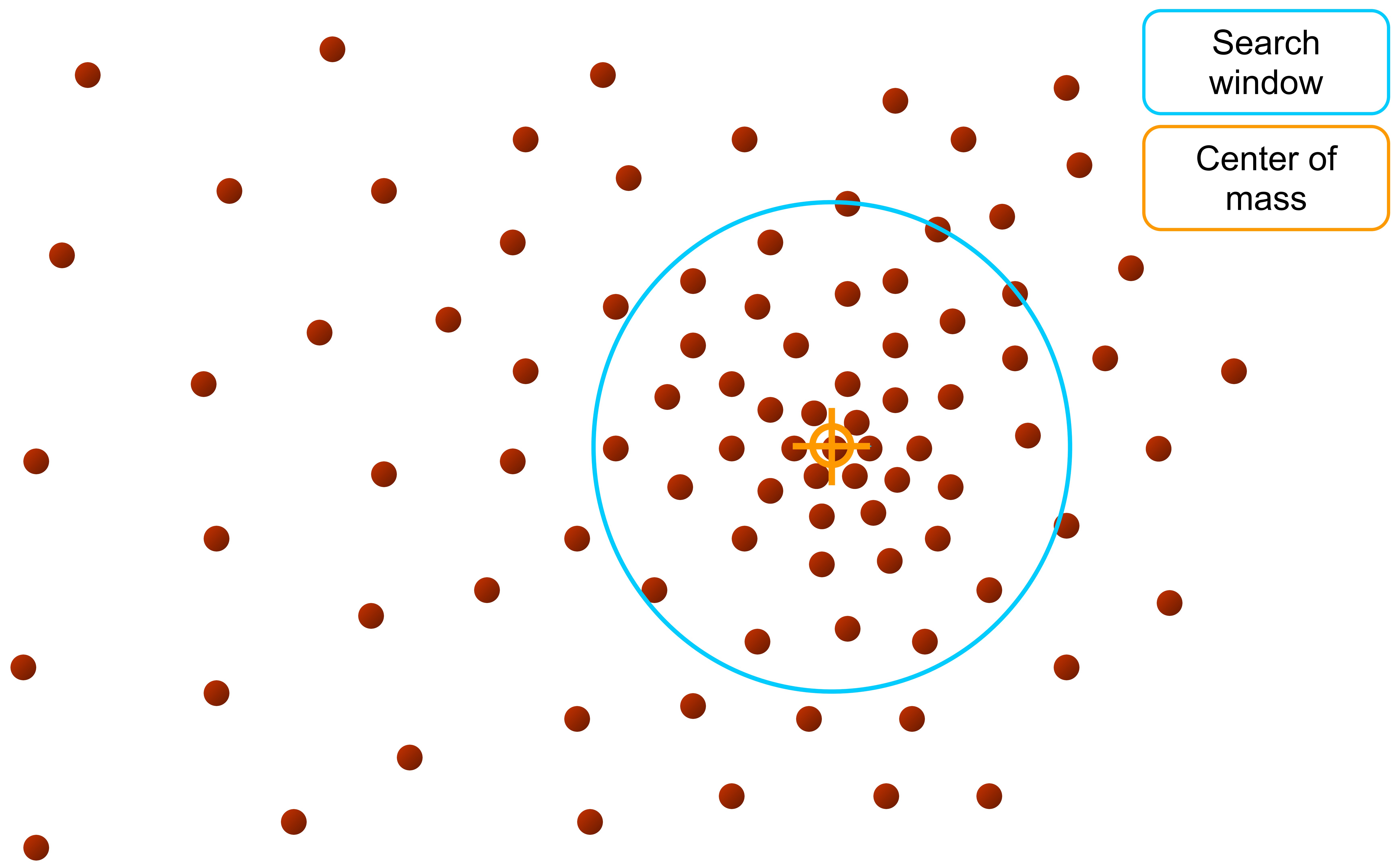
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel



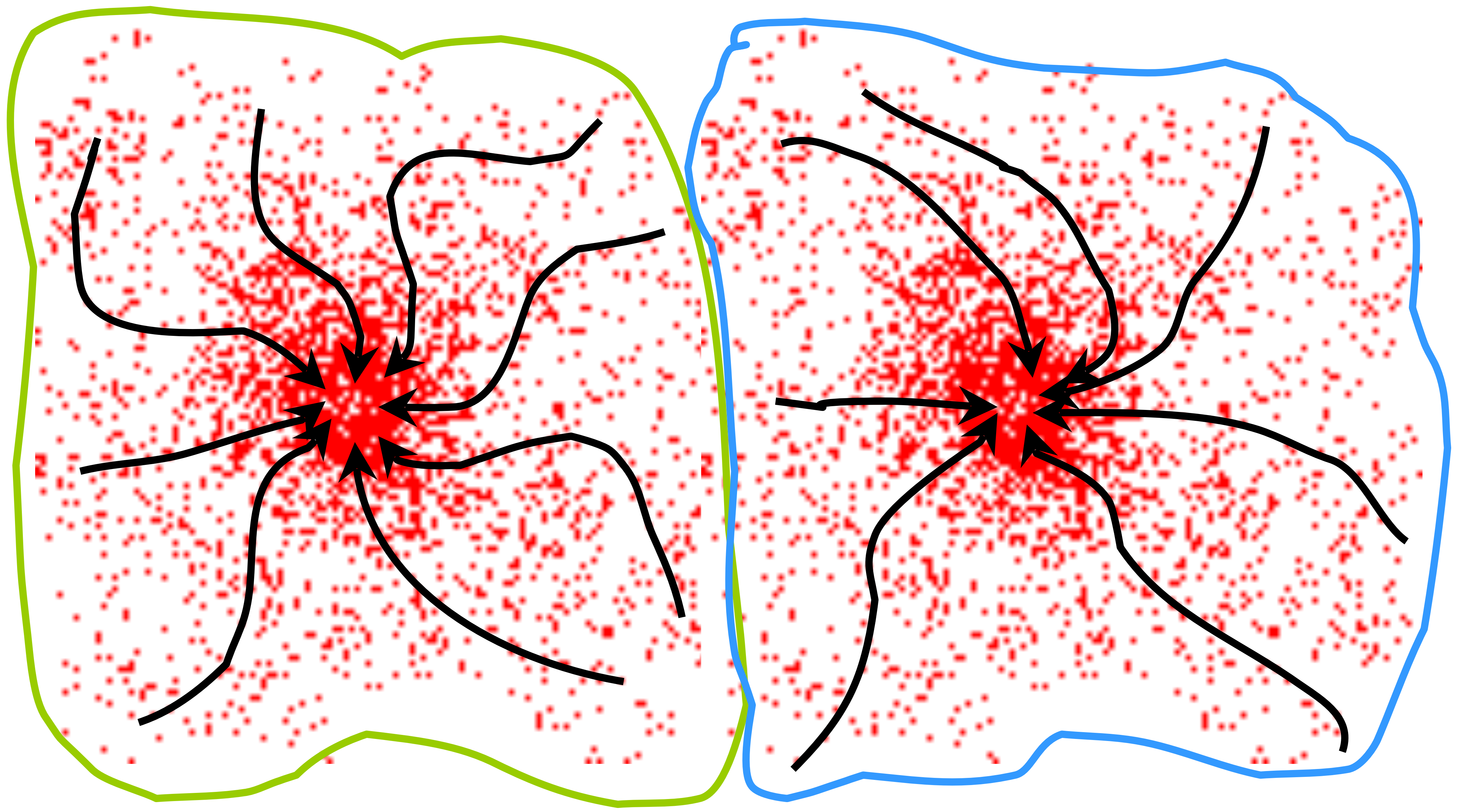
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

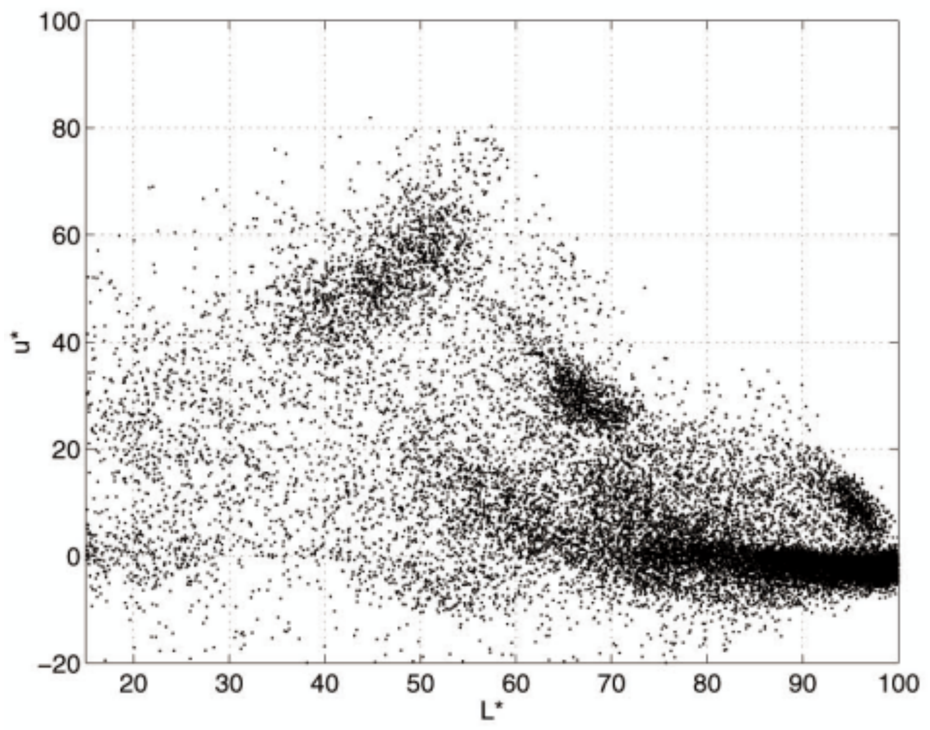
- Cluster: all data points in the attraction basin of a mode



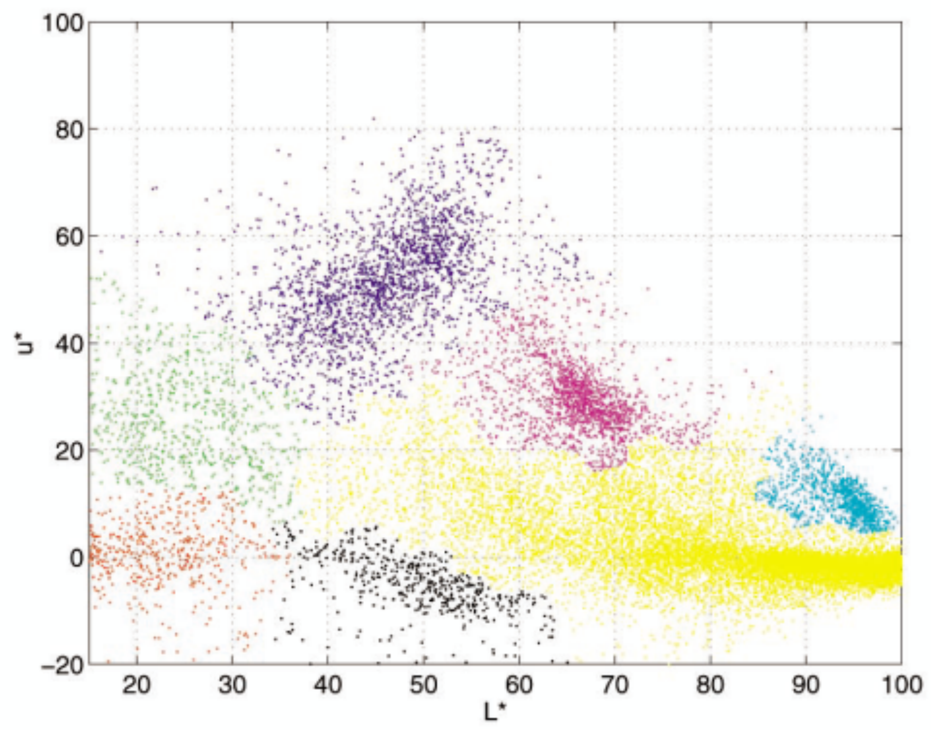
Slide by Y. Ukrainitz & B. Sarel

Mean Shift Segmentation

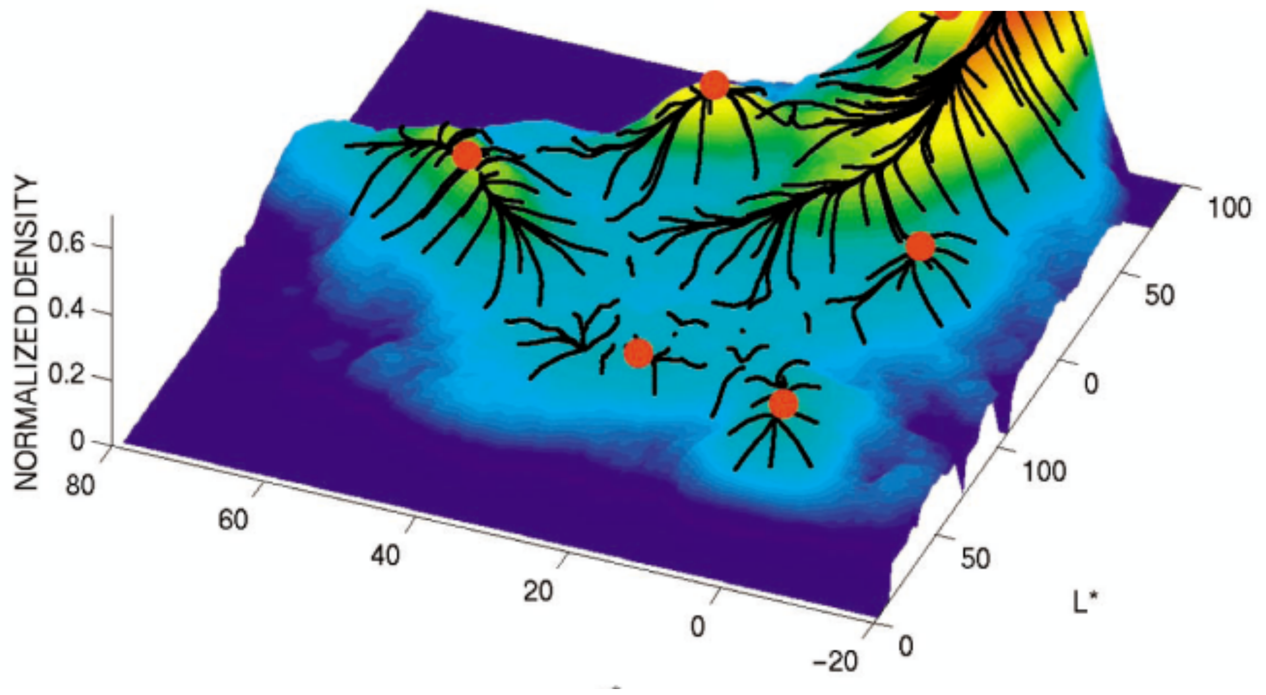
1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same "peak" or mode.
5. The data these merged windows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



Pixels in L*u* space



Clustering results after
~160 mean shift procedures

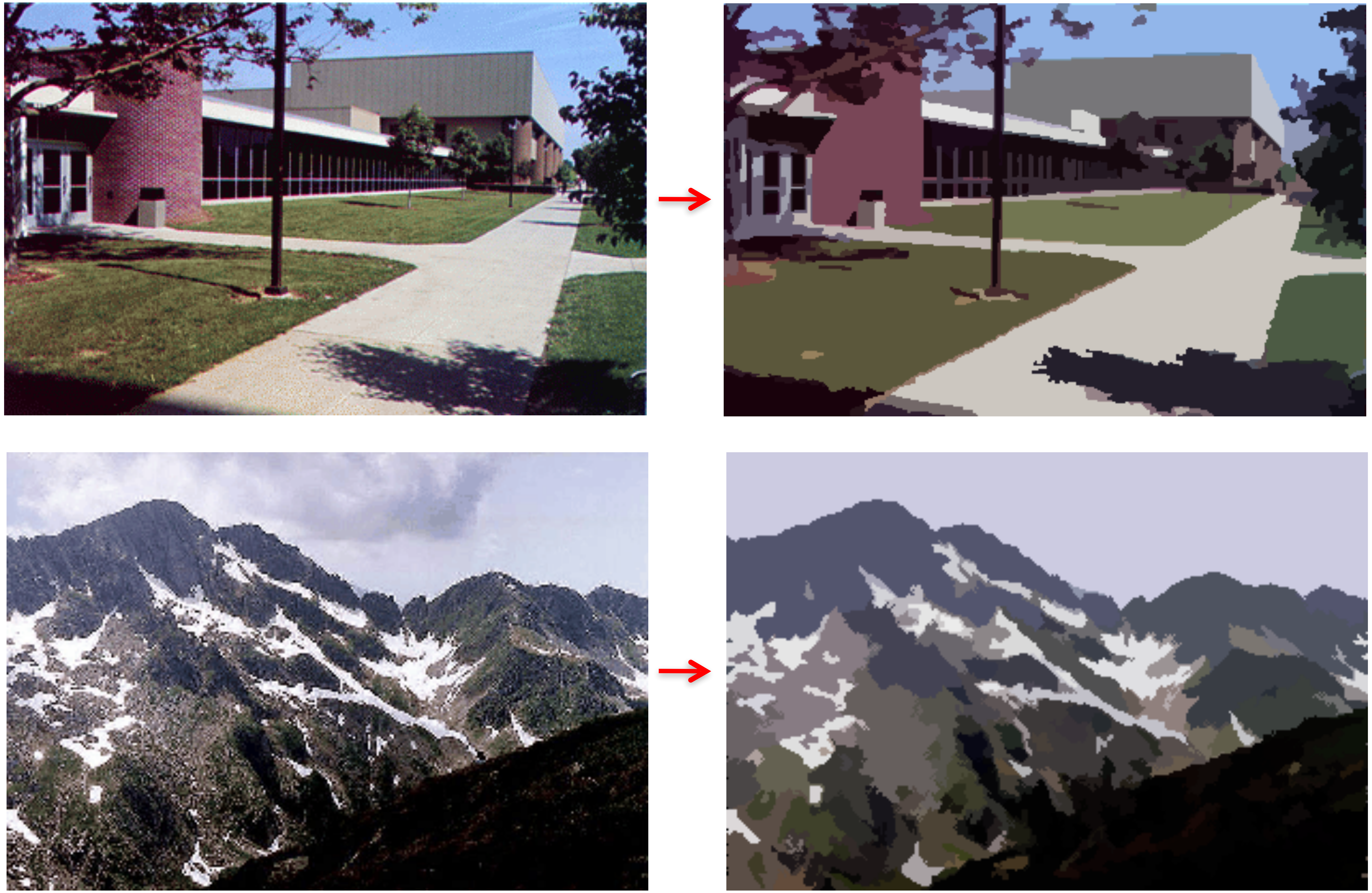


Corresponding trajectories with peaks marked as red dots

Szeliski – Chapter 5 – Segmentation, 5.3 Mean shift

Source: A. Oliva

Mean Shift Segmentation Results



Source: A. Oliva

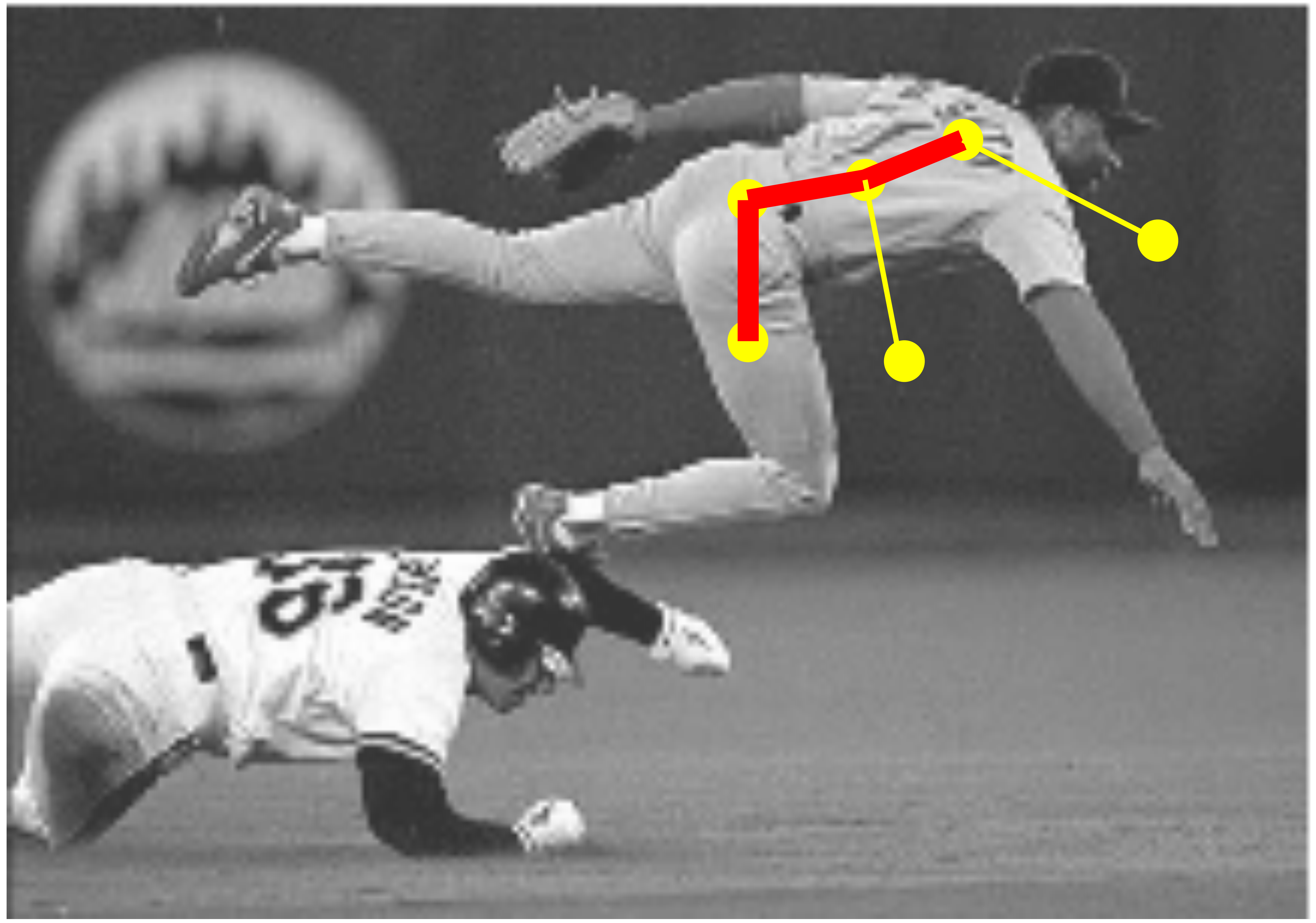
Mean shift pros and cons

- Pros
  - Nonparametric (e.g. K-means has a parametric model for cluster shape)
  - Clusters are places where data points tend to be close together
  - 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

Slide credit: S. Lazebnik

Method 3: Graph-based Image Segmentation

Build a **weighted graph G** = (V,E) from image



A different way of thinking about segmentation…

V: image pixels

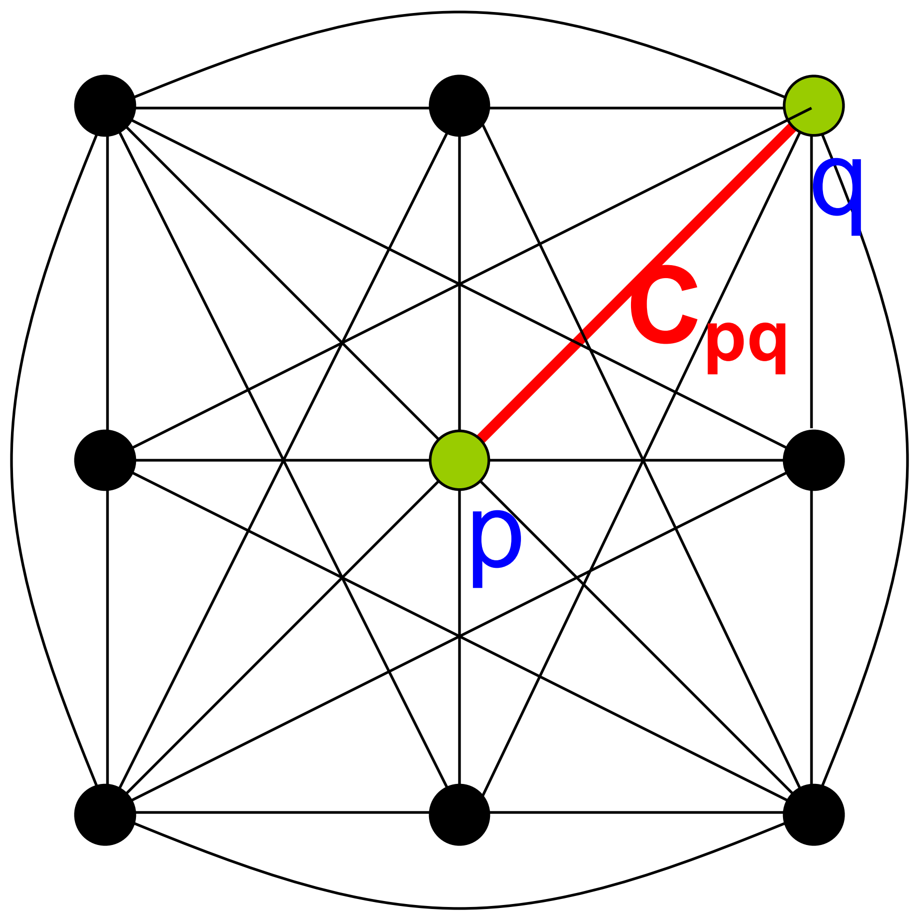
E: connections between pairs of nearby pixels

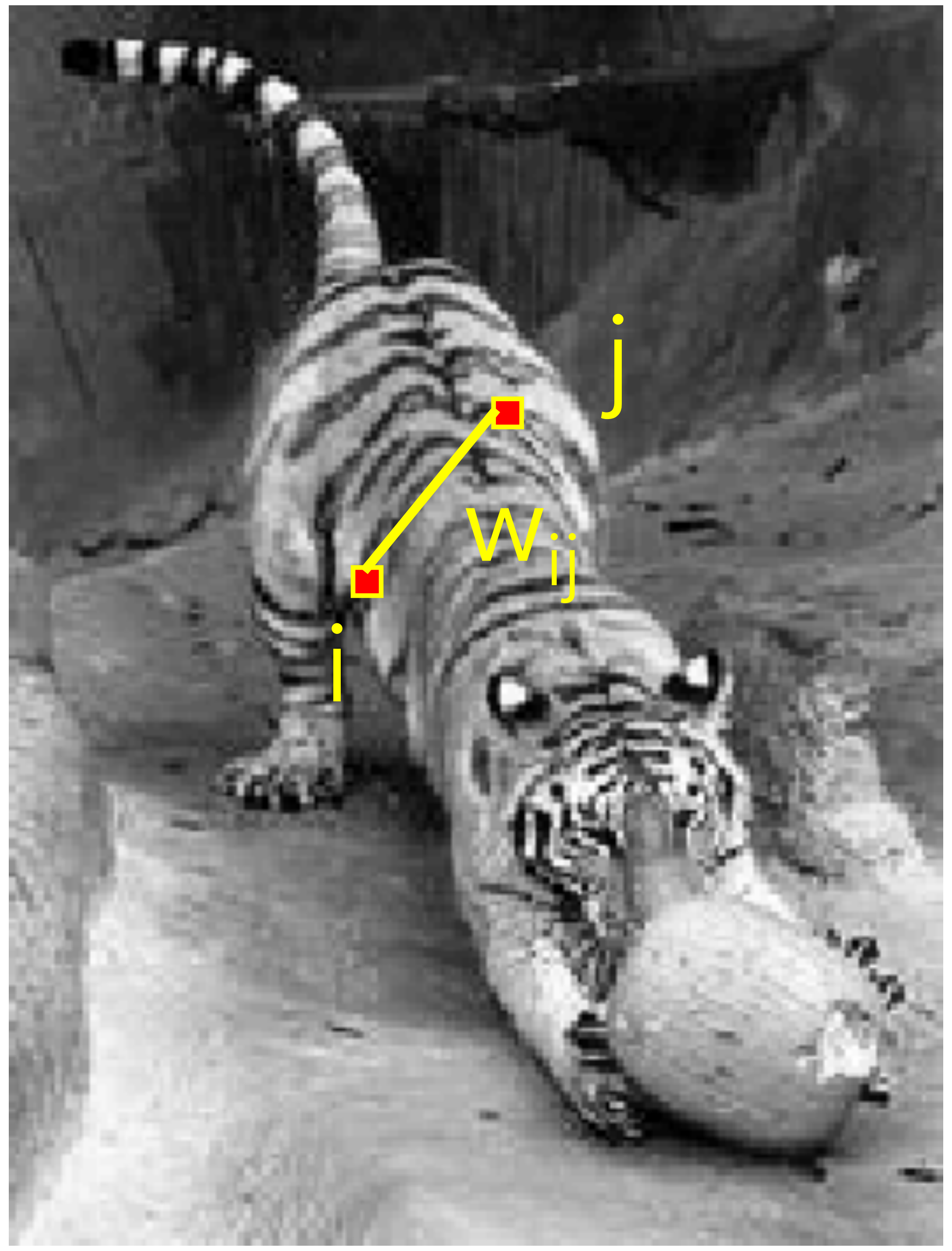
W_{ij} : probability that i &j belong to the same region

Segmentation = graph partition

Source: A. Oliva

Graph formulation

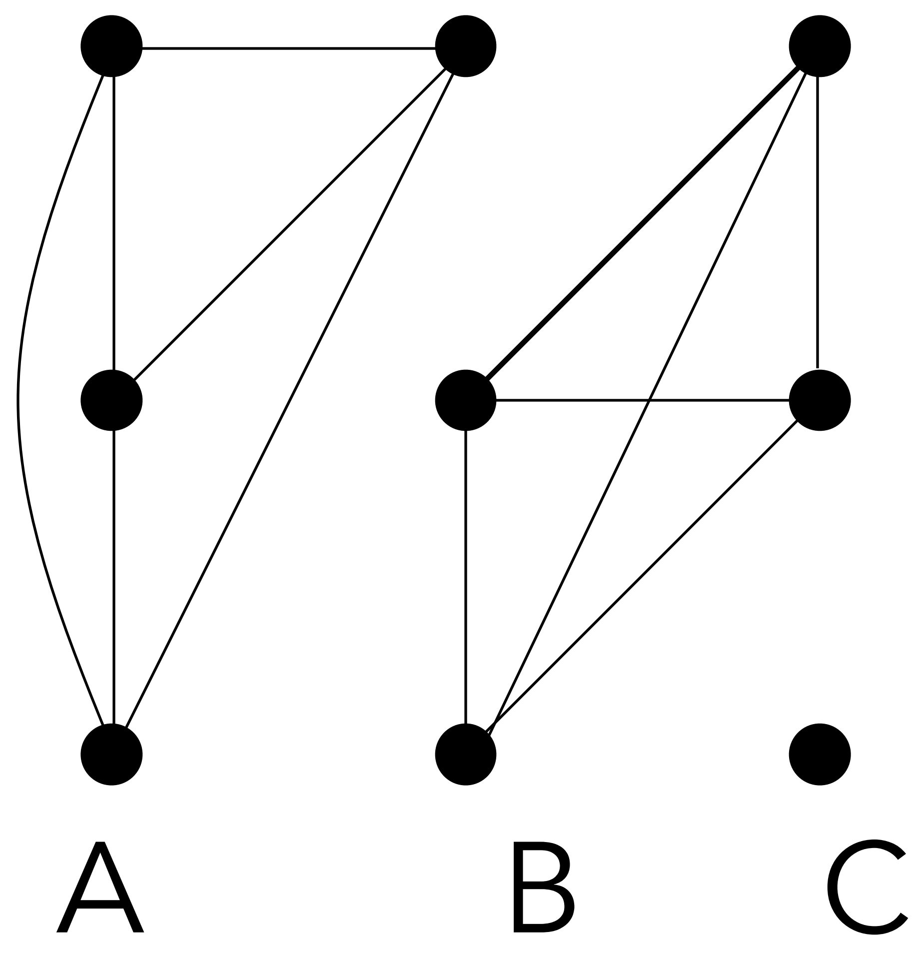




- Fully connected graph (node for every pixel i,j)
- Edge/link between every pair of pixels: **p,q**
- Each edge is weighted by the affinity or similarity of the two nodes:
  - cost c_{pq} for each link: c_{pq} measures similarity (or affinity)
  - similarity is inversely proportional to difference in color and position

Source: A. Oliva

Segmentation by graph cut



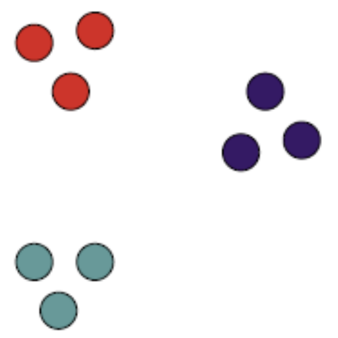
- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (similarity or **affinity)**
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: S. Seitz

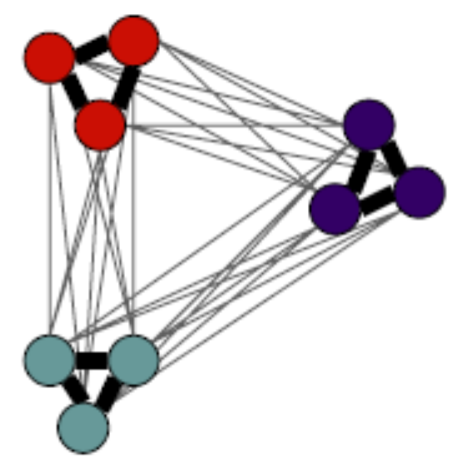
Graph-based Image Segmentation

Goal: Given data points X1, …, Xn and similarities w(Xi,Xj), partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

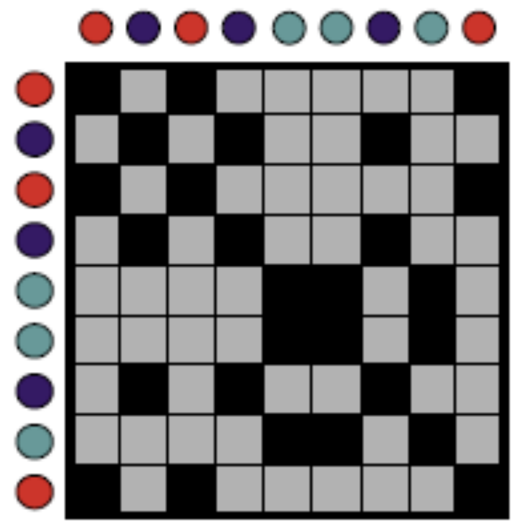
1- Get vectors of **data**



2a- Build a **similarity** graph

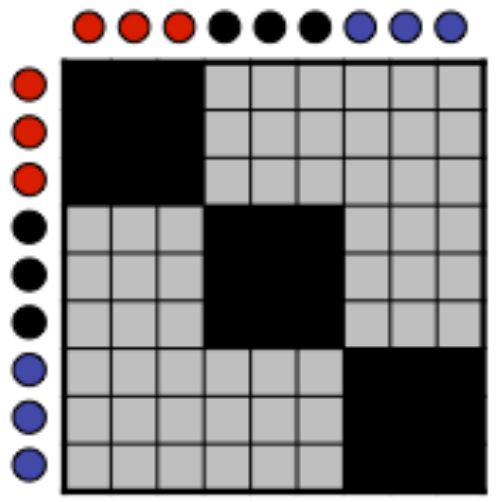


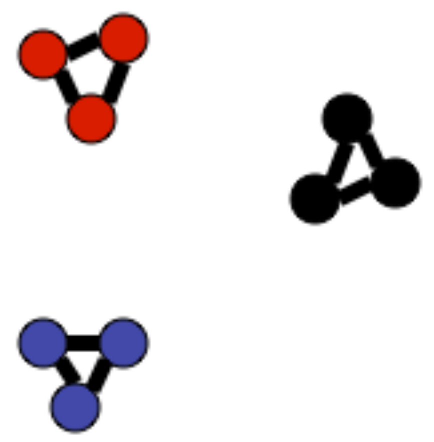
2b- Build a similarity/affinity **matrix**



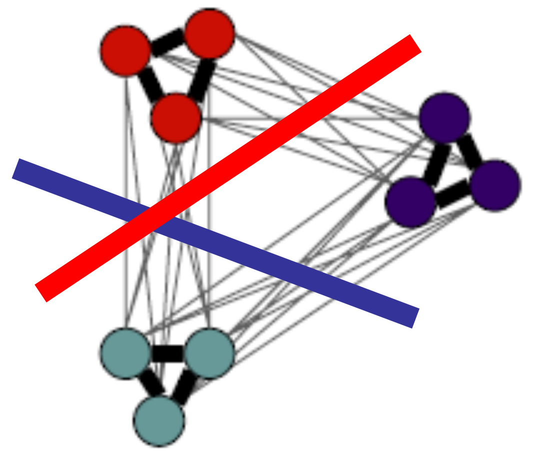
Similarities

3- Calculate eigenvectors





4- Cut the graph:
apply threshold to eigenvectors

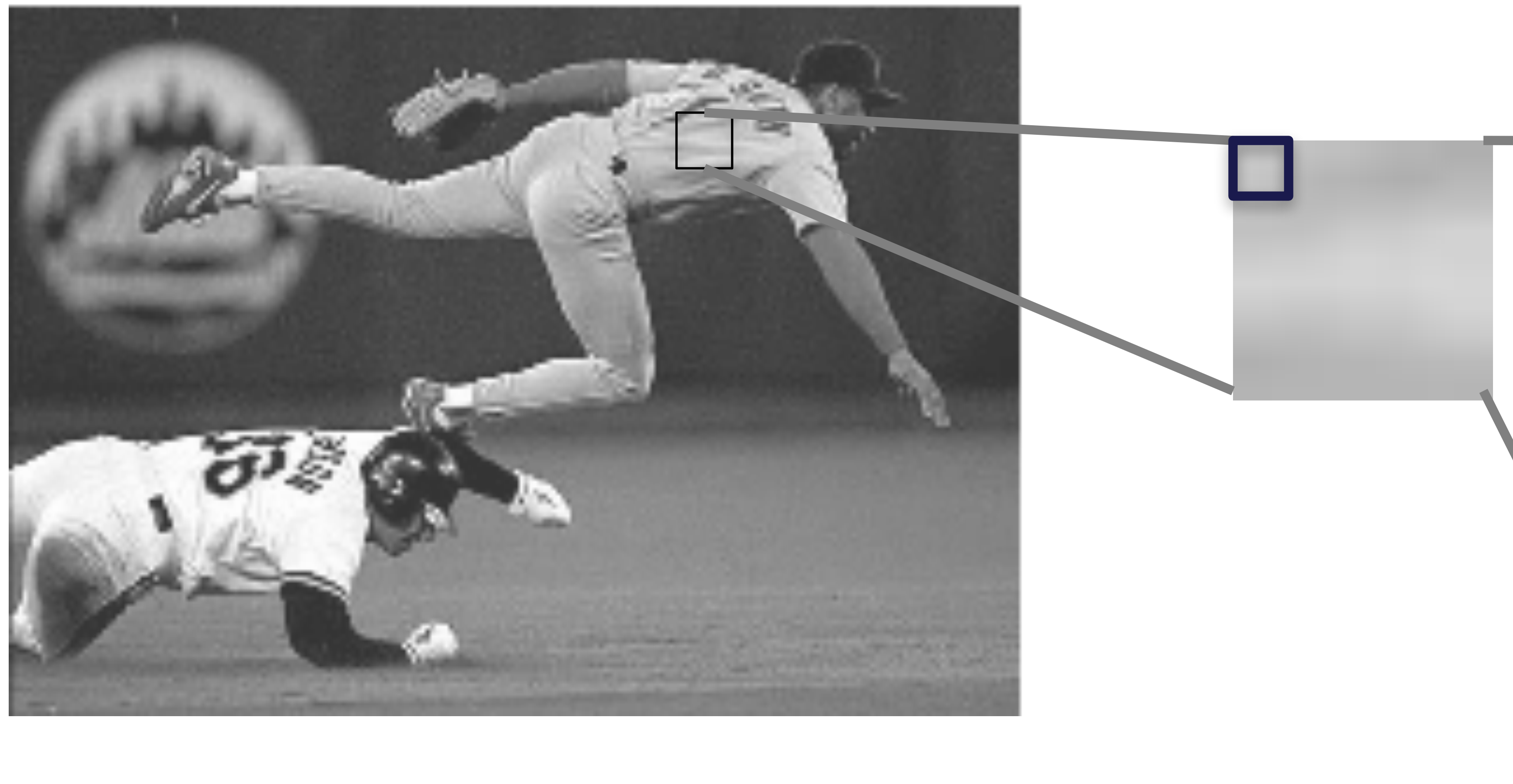


Source: A. Oliva

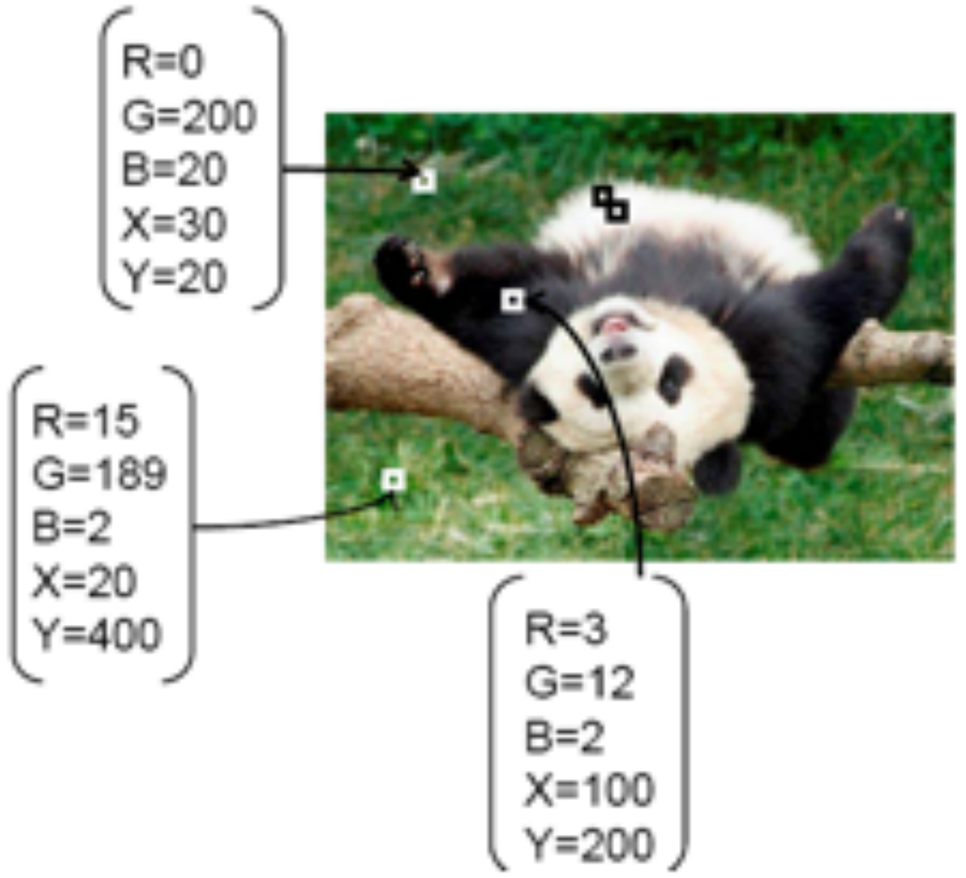
1- Vectors of data

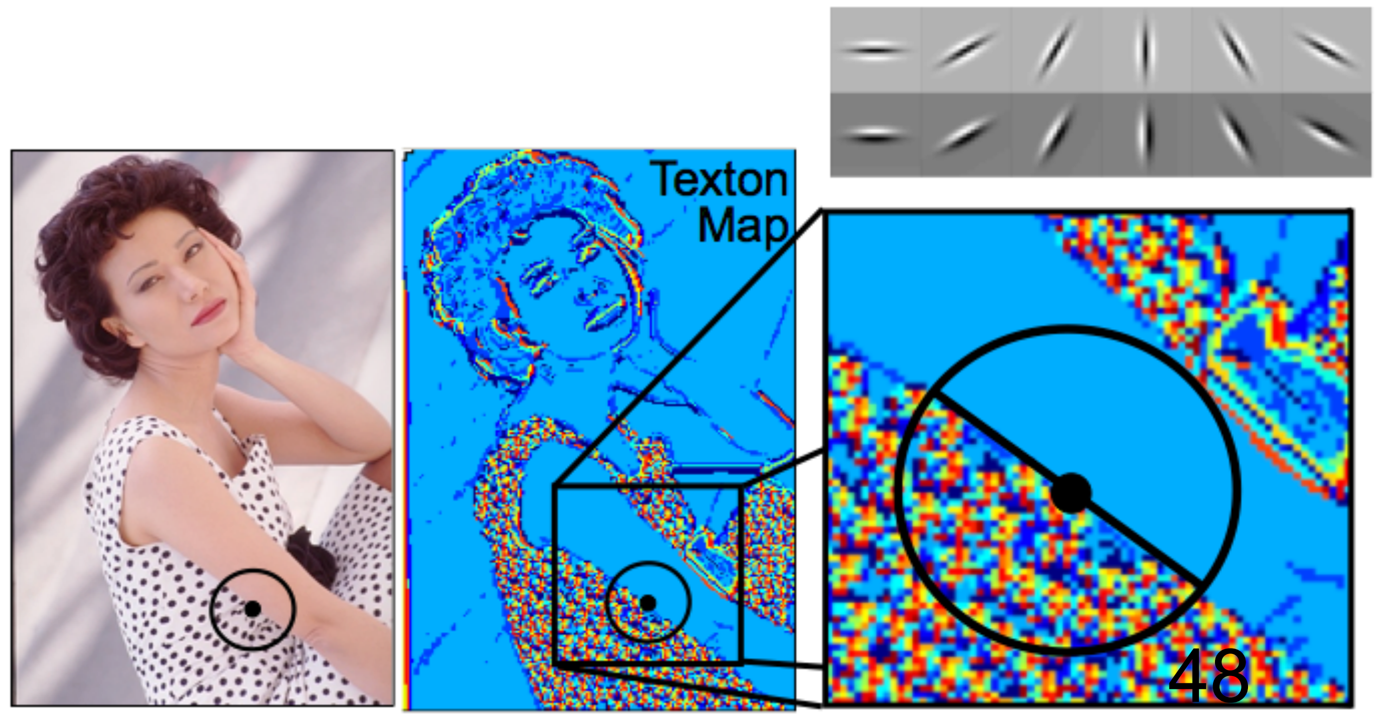
We represent each pixel by a feature vector **x**, and define a distance function appropriate for this feature representation (e.g. euclidean distance).

Features can be brightness value, color– RGB, L*u*v; texton histogram, etc- and calculate distances between vectors (e.g. Euclidean distance)



191	187	179	176	176
186	180	169	165	165
181	177	171	172	172
178	178	178	184	188
194	193	191	195	200
201	201	199	201	205
188	186	183	185	191
172	171	172	174	175
157	158	163	164	162
163	163	165	165	163





Textons (texture features)

Source: A. Oliva

From distance to affinity

- We represent each pixel by a feature vector x, and define a distance function appropriate for this feature representation
- Then we can convert the distance between two feature vectors into an affinity/similarity measure with the help of a generalized Gaussian kernel:

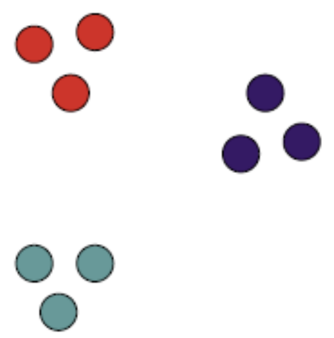
$$\exp\left(-\frac{1}{2\sigma^2}\mathrm{dist}(\mathbf{x}_i,\mathbf{x}_j)^2\right)$$

Source: S. Lazebnik

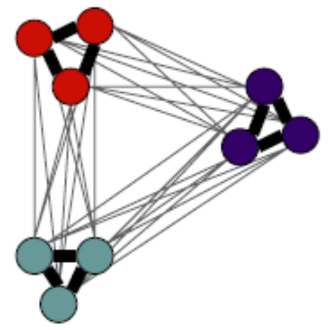
Graph-based Image Segmentation

Goal: Given data points X1, …, Xn and similarities w(Xi,Xj), partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

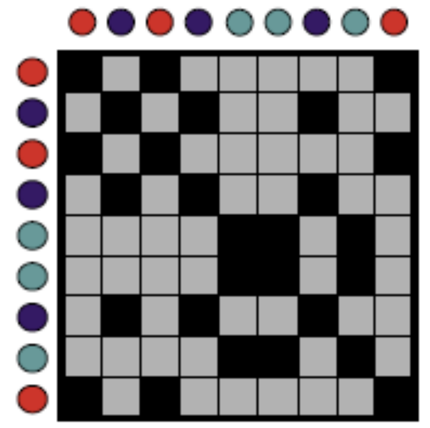
1- Get vectors of **data**



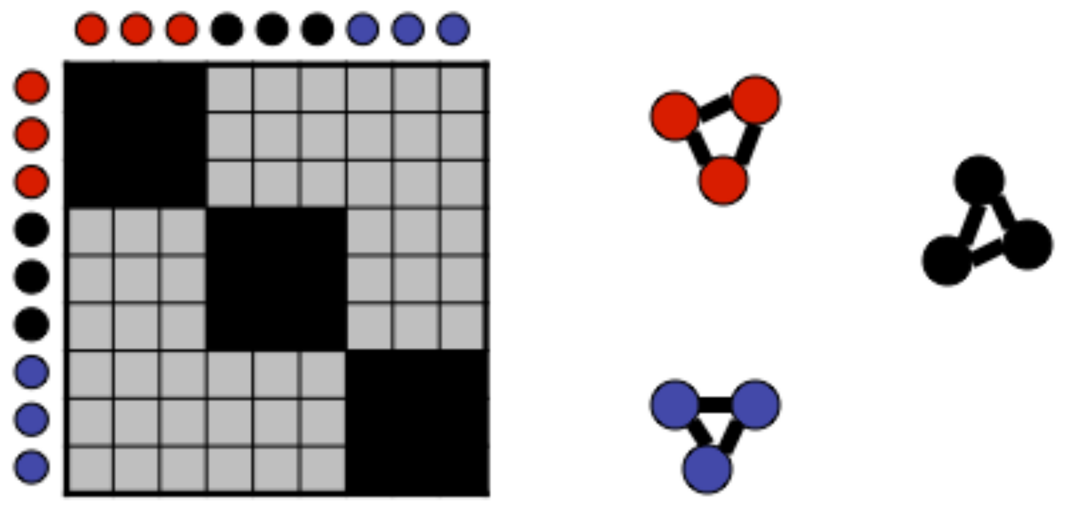
2a- Build a **similarity** graph



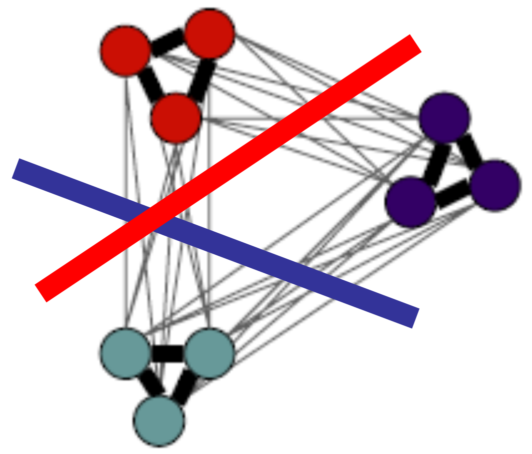
2b- Build a similarity/affinity **matrix**



3- Calculate eigenvectors

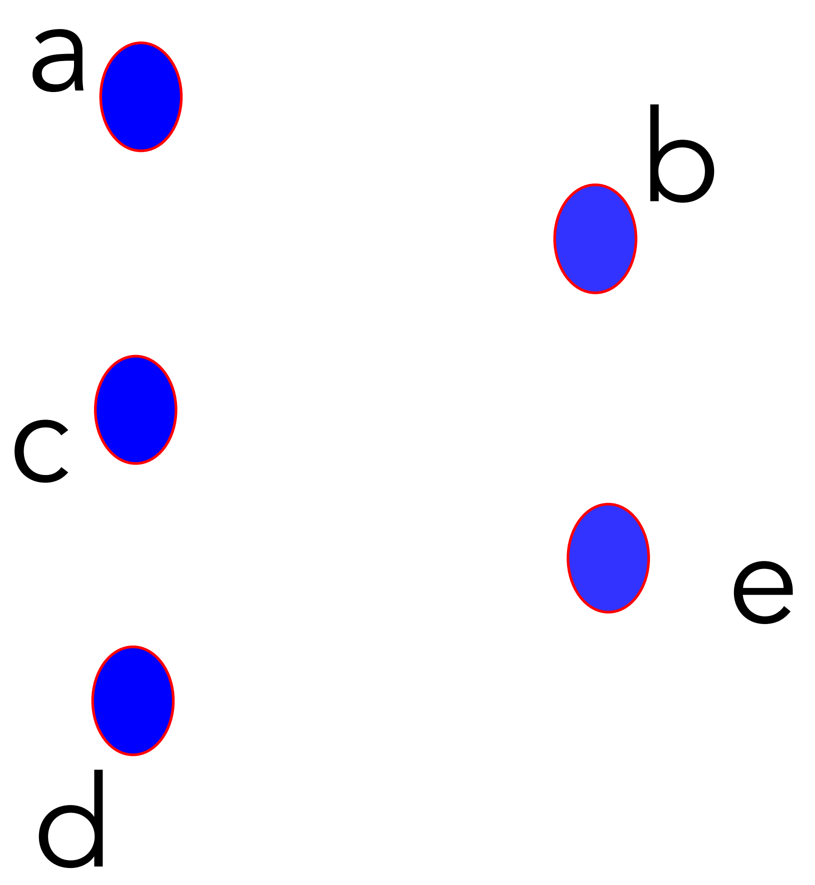


4- Cut the graph:
apply threshold to eigenvectors



Source: A. Oliva

2a- Graph

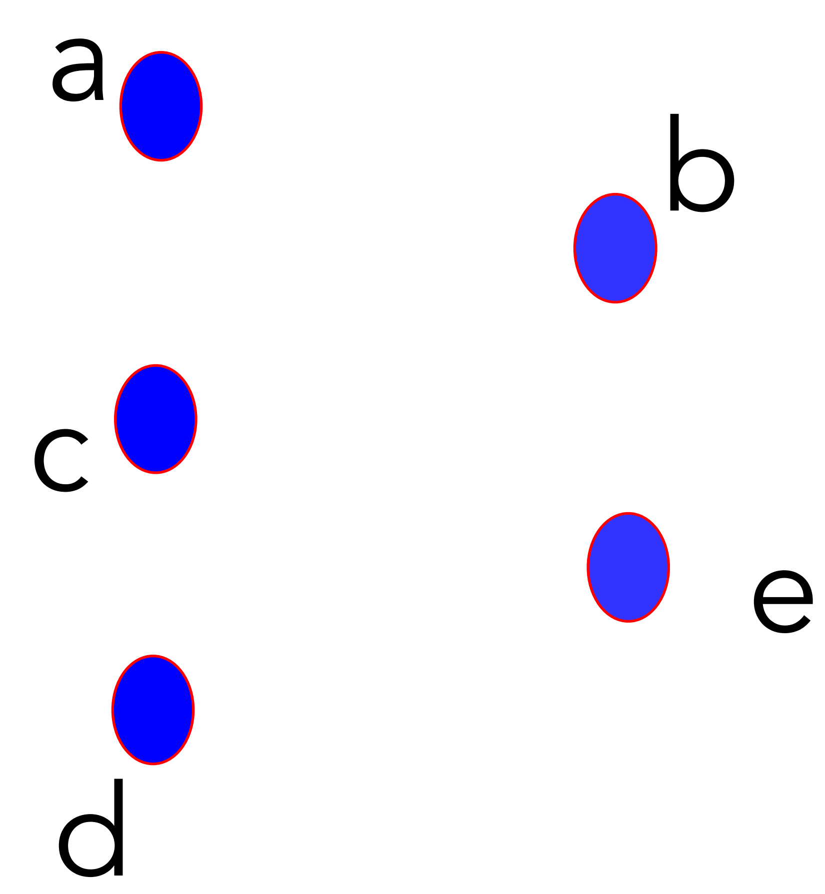


$$
\begin{array}{c}
\\ a \\ b \\ c \\ d \\ e
\end{array}
\begin{array}{c}
\begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{bmatrix}
- & - & - & - & - \\
- & - & - & - & - \\
- & - & - & - & - \\
- & - & - & - & - \\
- & - & - & - & -
\end{bmatrix}
\end{array}
$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

2a- Graph

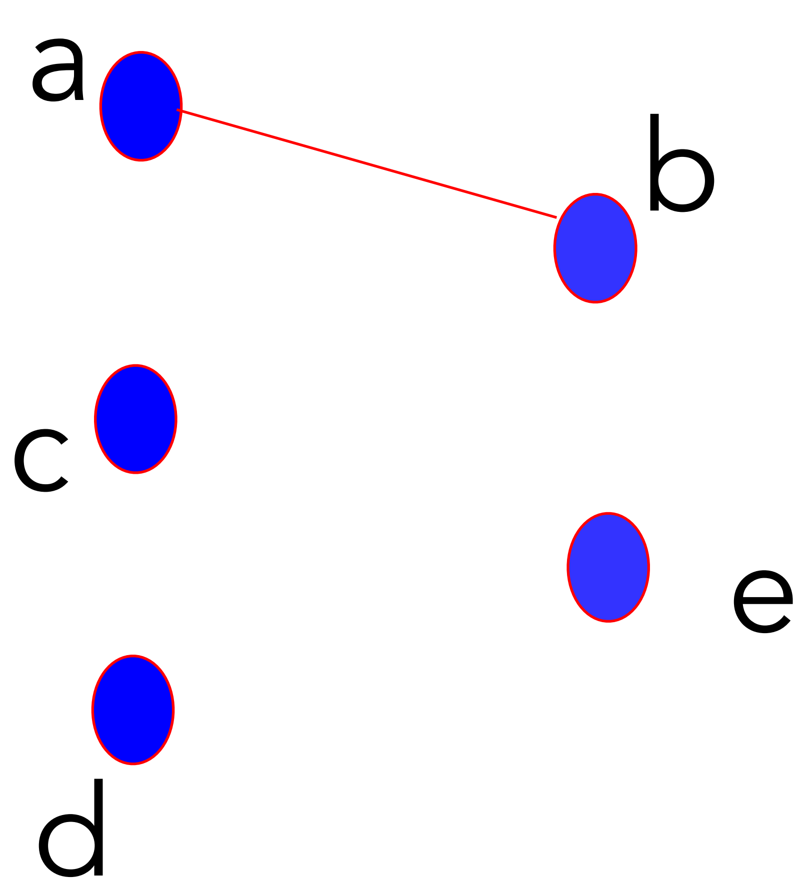


$$
\begin{array}{c}
 \\ a \\ b \\ c \\ d \\ e
\end{array}
\begin{array}{c}
\begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{bmatrix}
0 & - & - & - & - \\
- & 0 & - & - & - \\
- & - & 0 & - & - \\
- & - & - & 0 & - \\
- & - & - & - & 0
\end{bmatrix}
\end{array}
$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

2a- Graph

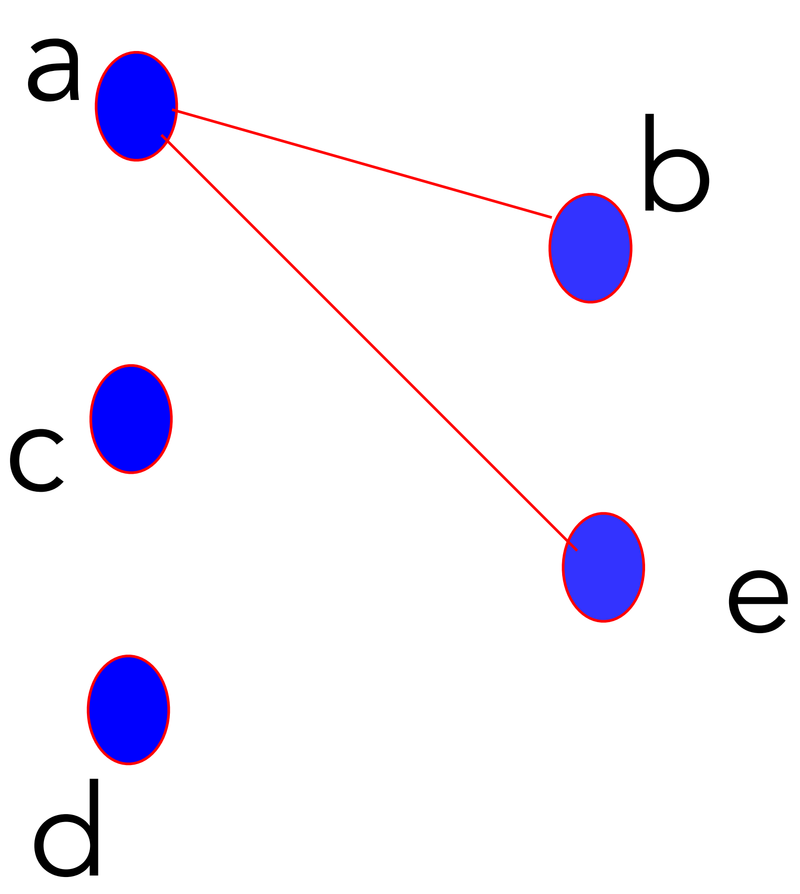


$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \left[\begin{array}{ccccc} 0 & 1 & - & - & - \\ 1 & 0 & - & - & - \\ - & - & 0 & - & - \\ - & - & - & 0 & - \\ - & - & - & - & 0 \end{array}\right] \end{array}$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

2a- Graph

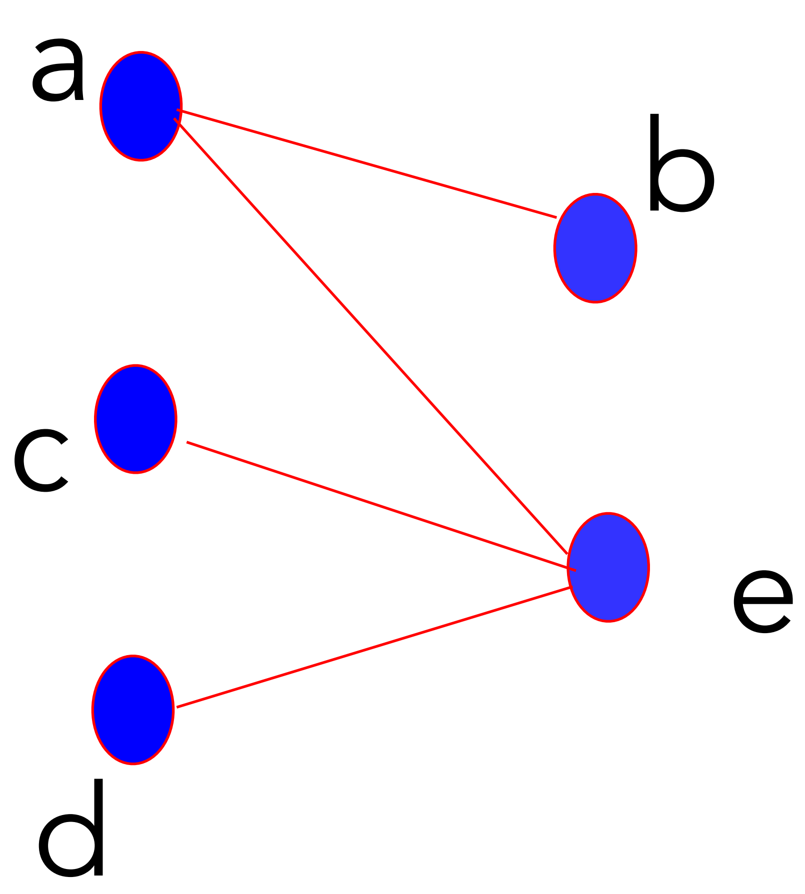


$$
\begin{array}{c c}
 & \begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} &
\begin{bmatrix}
0 & 1 & - & - & 1 \\
1 & 0 & - & - & - \\
- & - & 0 & - & - \\
- & - & - & 0 & - \\
1 & - & - & - & 0
\end{bmatrix}
\end{array}
$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

2a- Graph



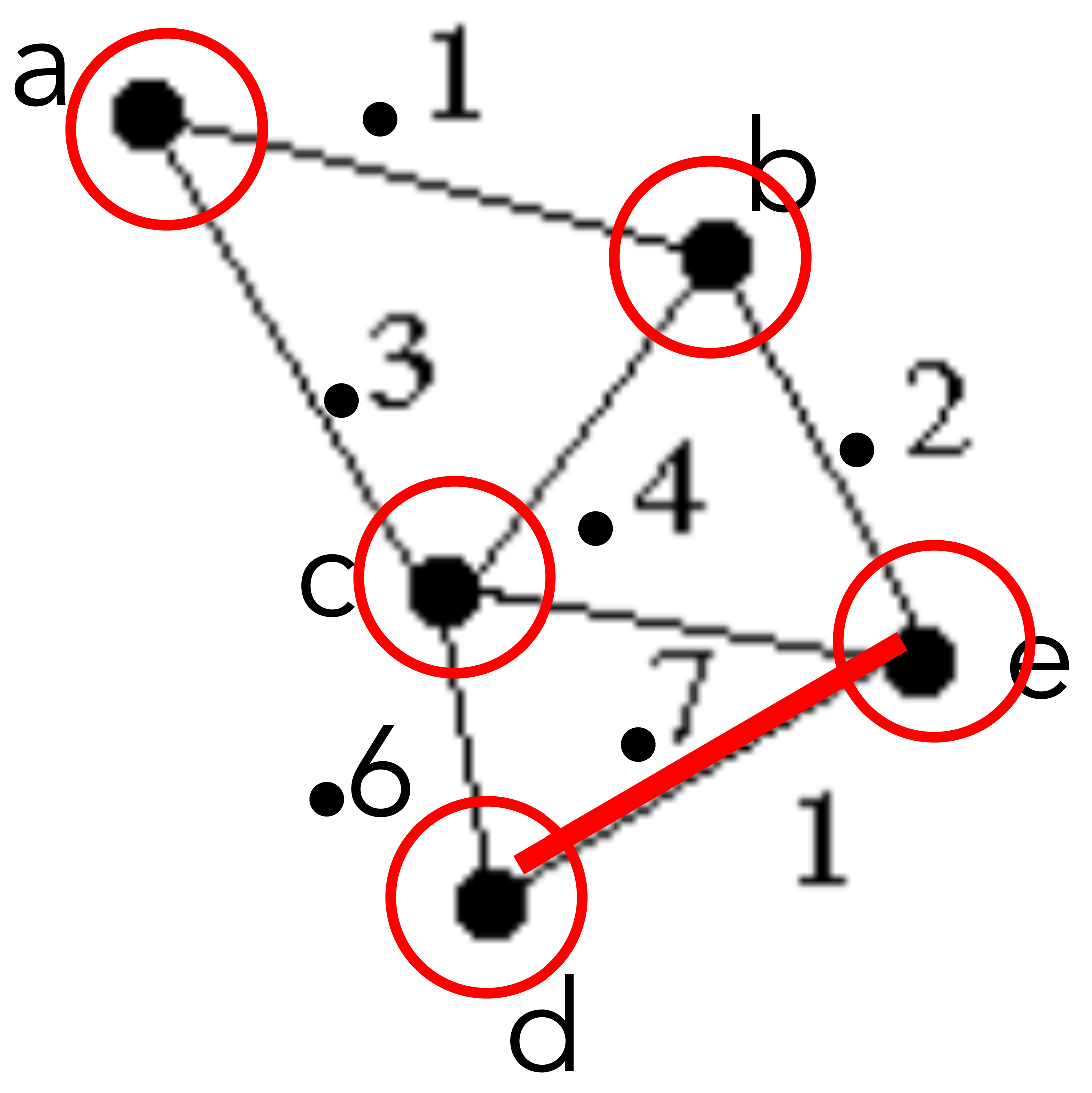
$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & 1 & 0 & 0 & 1 \\ b & 1 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 0 & 0 & 1 \\ d & 0 & 0 & 0 & 0 & 1 \\ e & 1 & 0 & 1 & 1 & 0 \end{array}$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

2a- What is a weighted graph?

Affinity Matrix represents the weighted links



$$
W = \begin{array}{c|ccccc|c}
 & a & b & c & d & e & \\
\hline
 & 1 & .1 & .3 & 0 & 0 & a \\
 & 0 & 1 & .4 & 0 & .2 & b \\
 & .3 & .4 & 1 & .6 & .7 & c \\
 & 0 & 0 & .6 & 1 & 1 & d \\
 & 0 & .2 & .7 & 1 & 1 & e
\end{array}
$$

Diagonal: each point with itself is 1

Strong links/edges

Weak links/edges

No links/edges connected

W_{ij} : probability that i &j belong to the same region

i,j are the pixels in the image

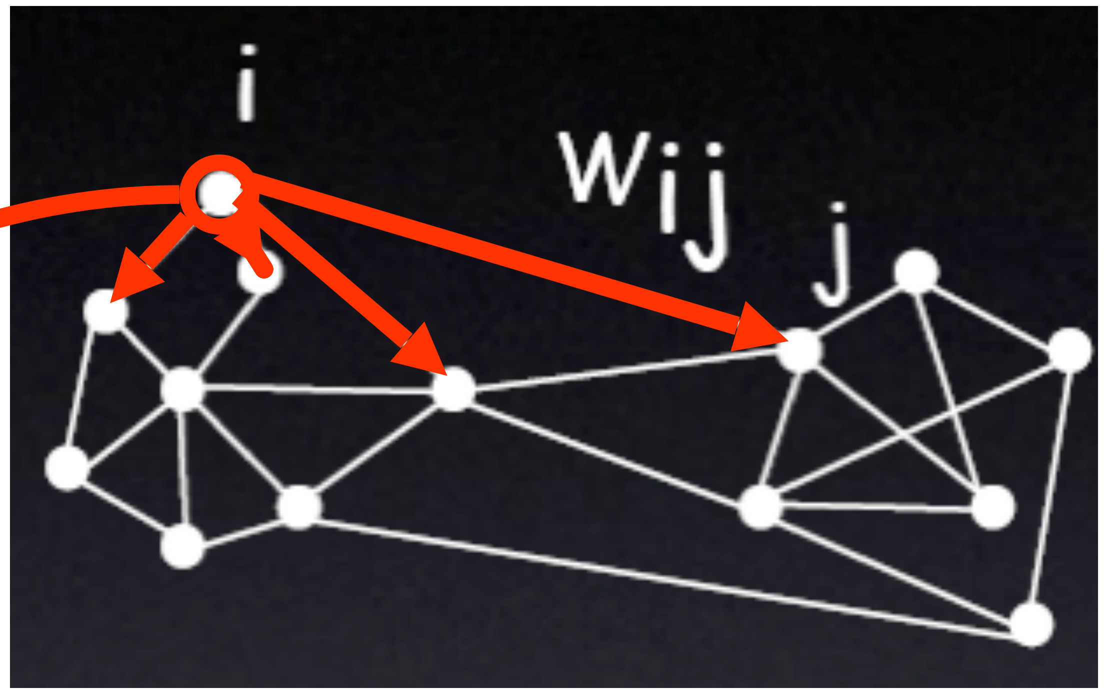
See Forsyth-Ponce chapter

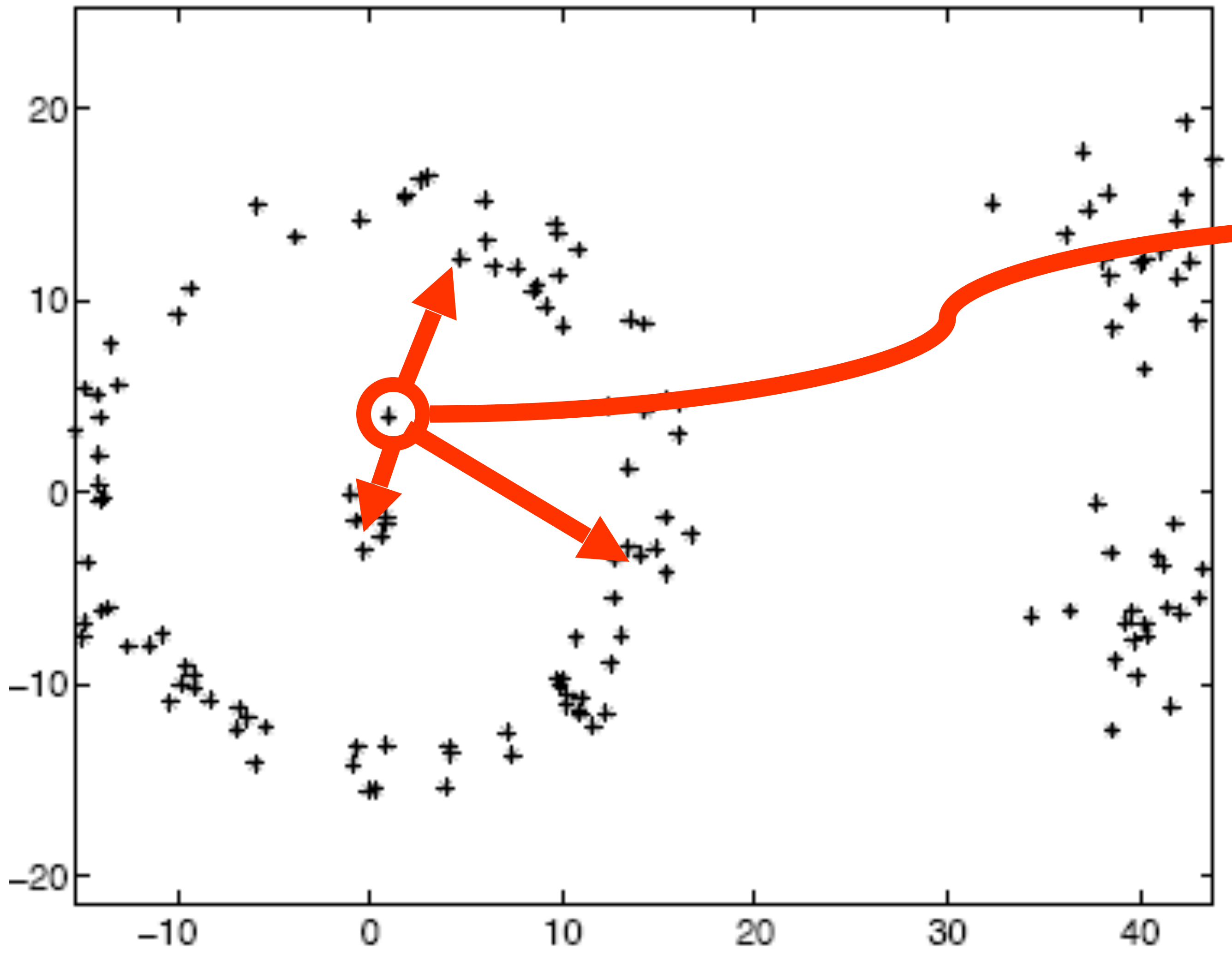
Source: A. Oliva

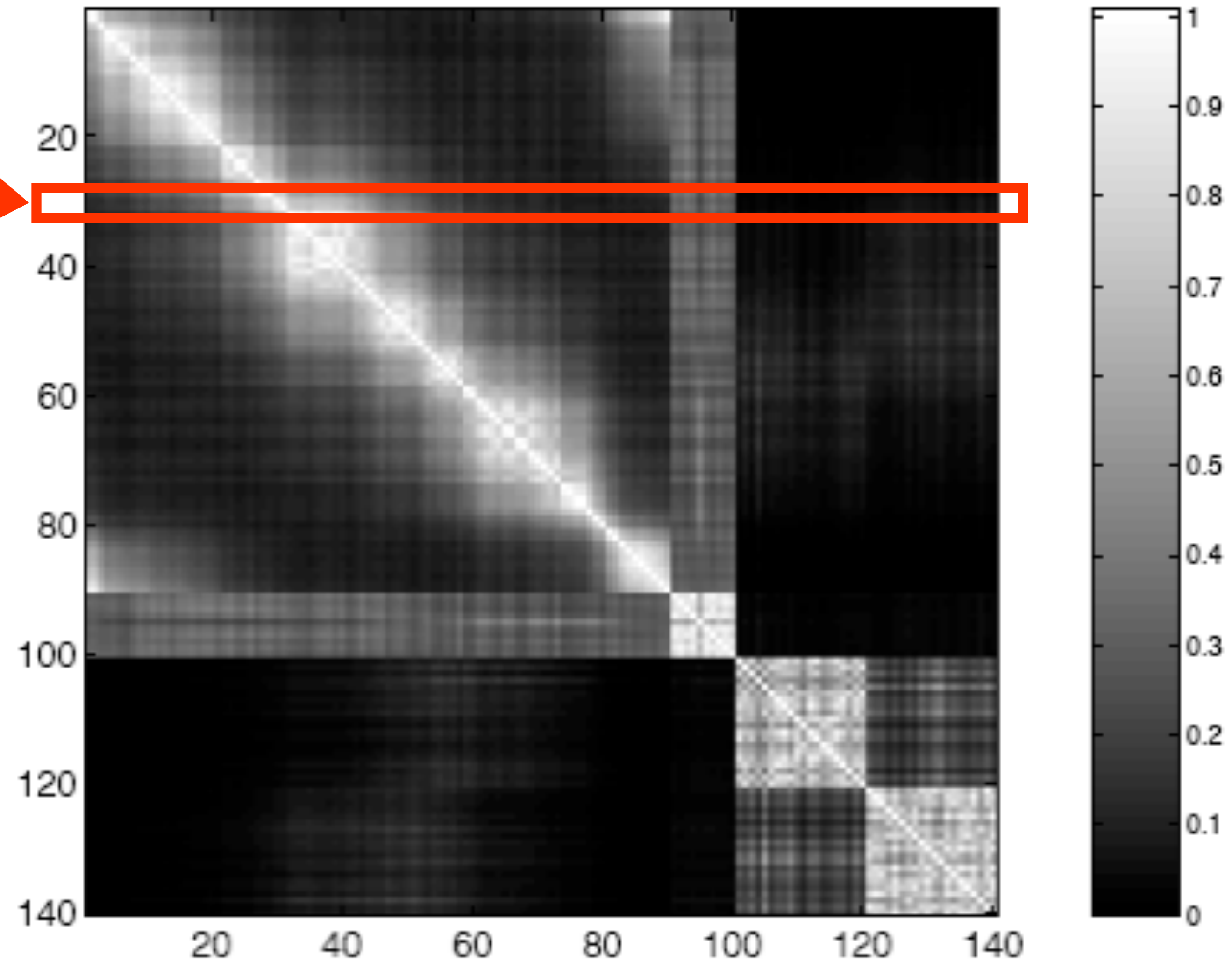
2b- Building Affinity Matrix

- Similarity matrix:

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



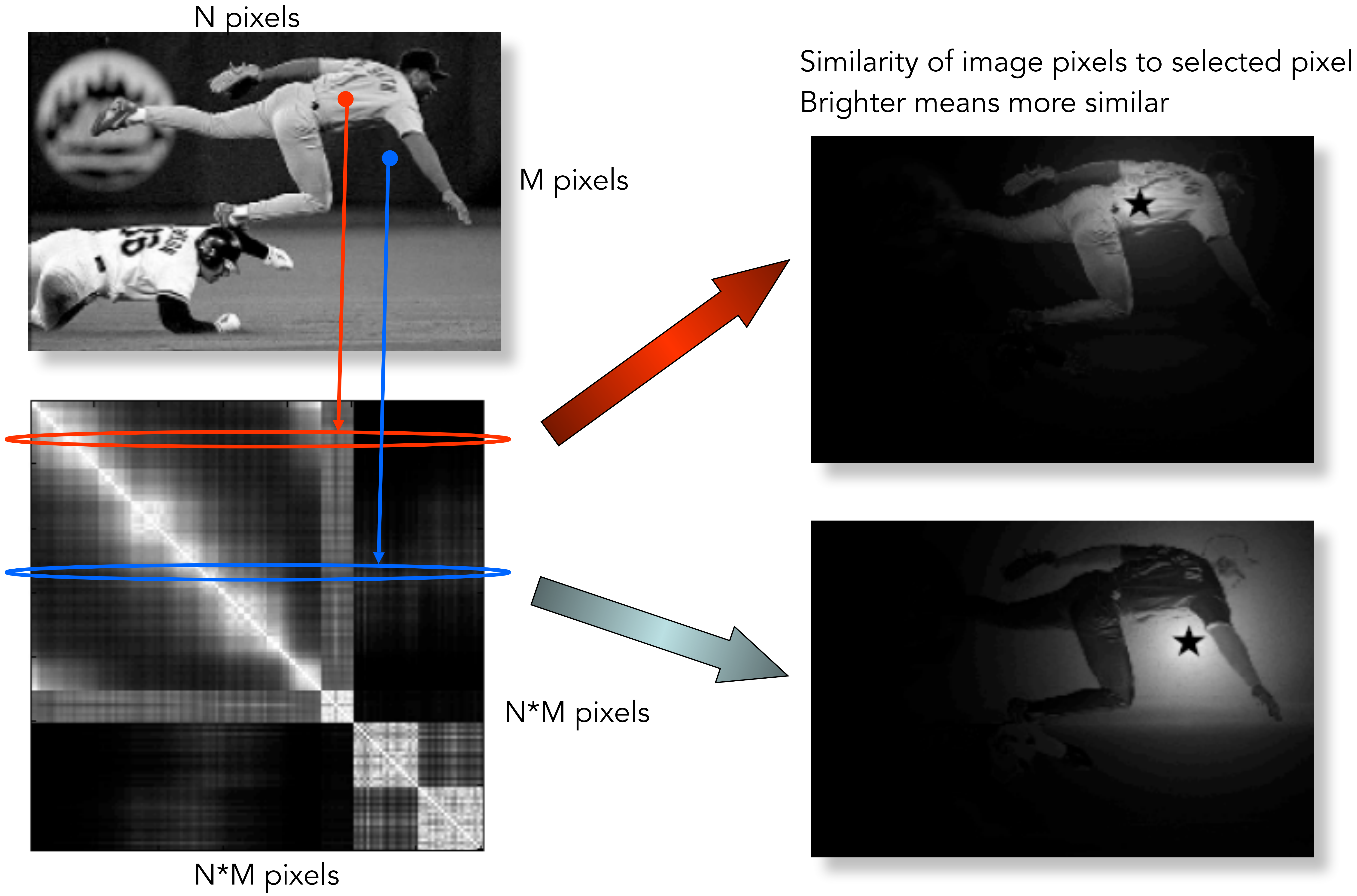




Weight matrix associated with the graph
(larger values are lighter)

Slides from Jianbo Shi

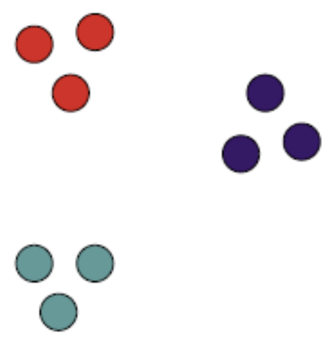
Affinity matrix of a natural image



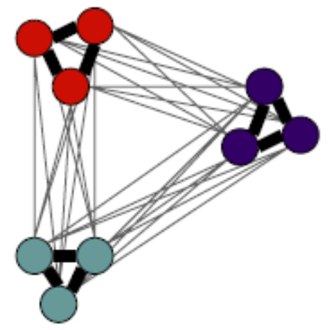
Graph-based Image Segmentation

Goal: Given data points X1, …, Xn and similarities w(Xi,Xj), partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

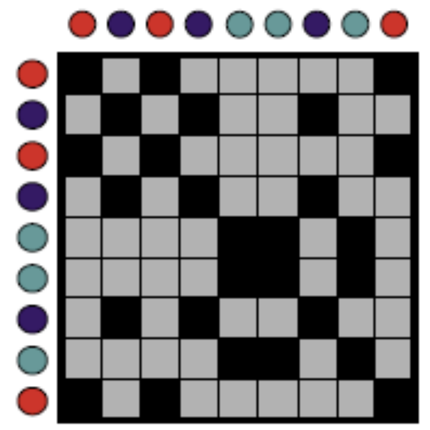
1- Get vectors of **data**



2a- Build a **similarity** graph

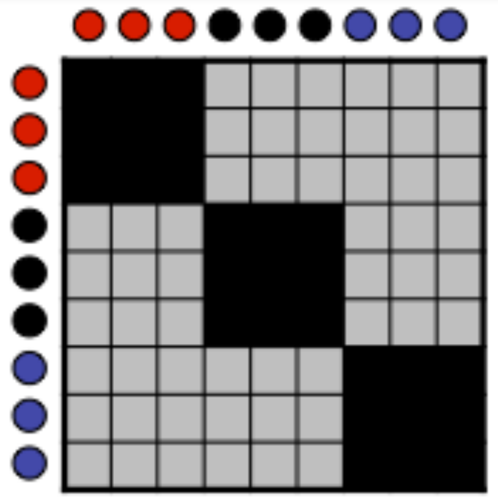


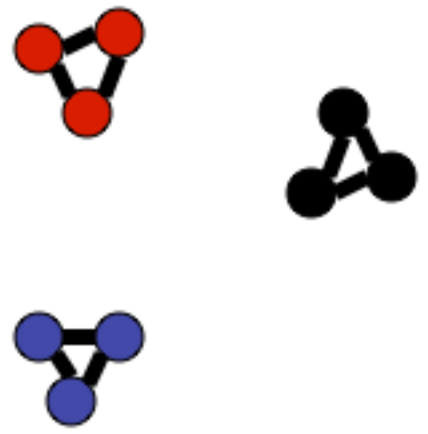
2b- Build a similarity **matrix**



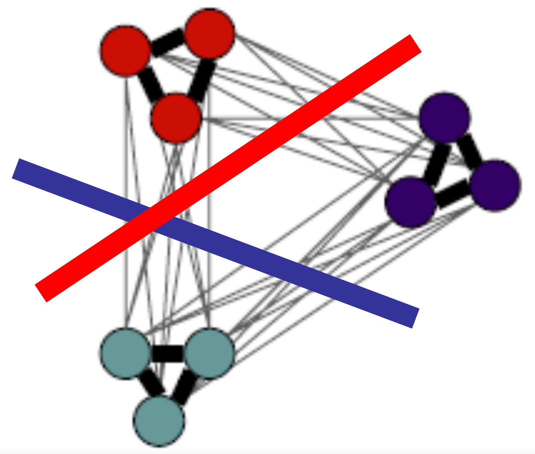
Similarities

3- Calculate eigenvectors



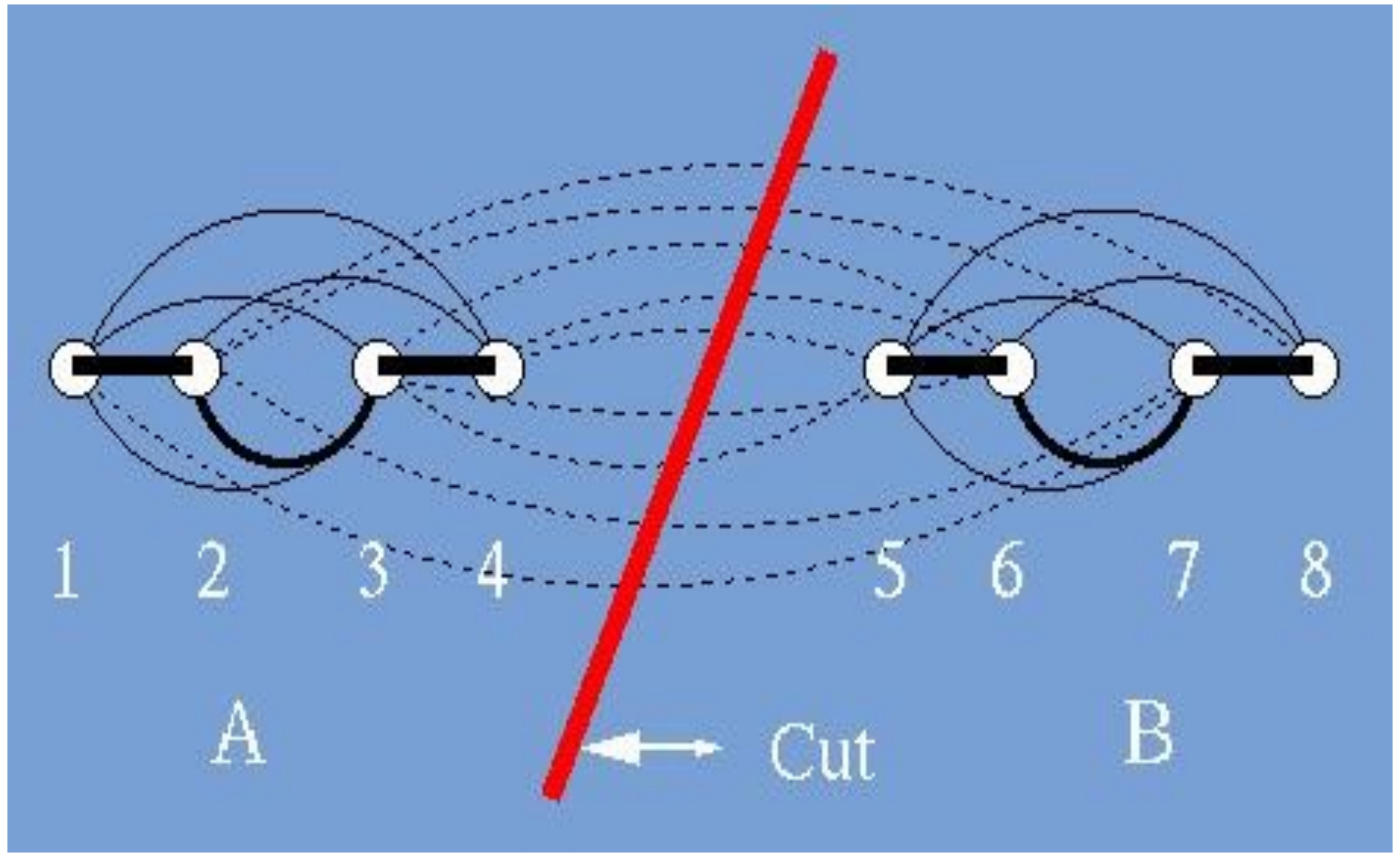


4- Cut the graph:
apply threshold to eigenvectors



Source: A. Oliva

Partition a graph with minimum cut



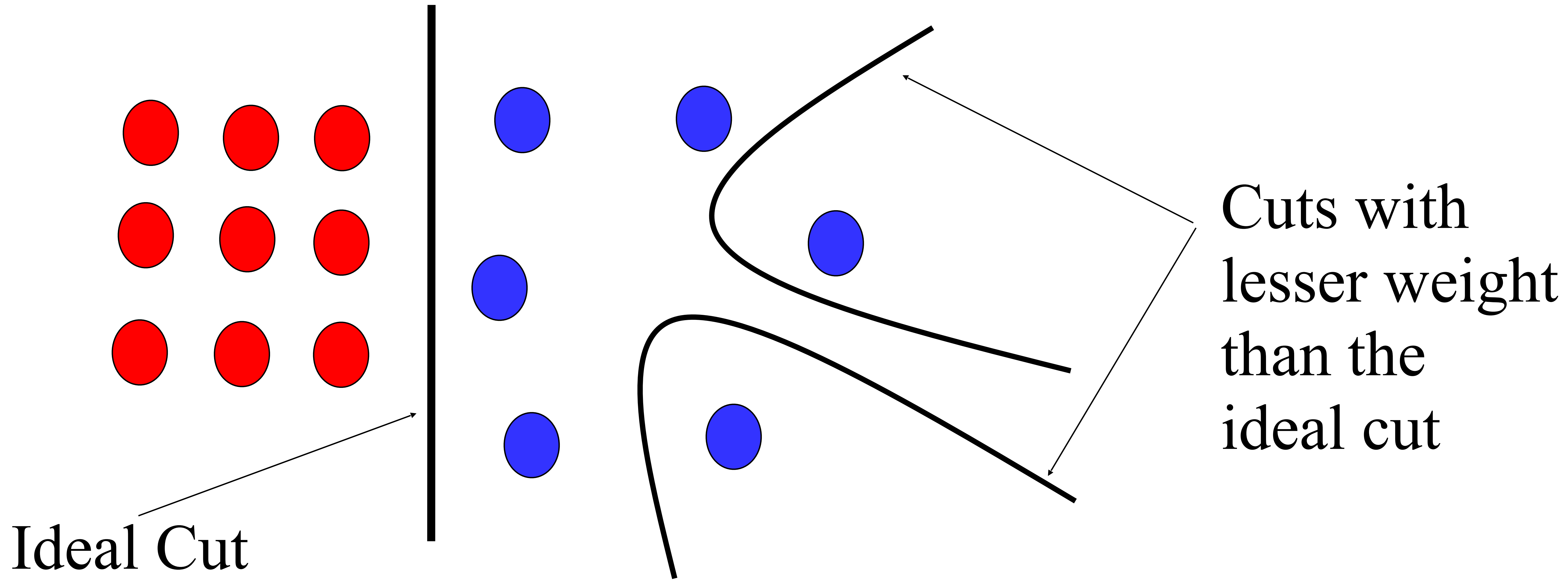
$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

- **Cut**: sum of the weight of the cut edges:
- Natural idea: **minimum cut**

Source: Jianbo Shi

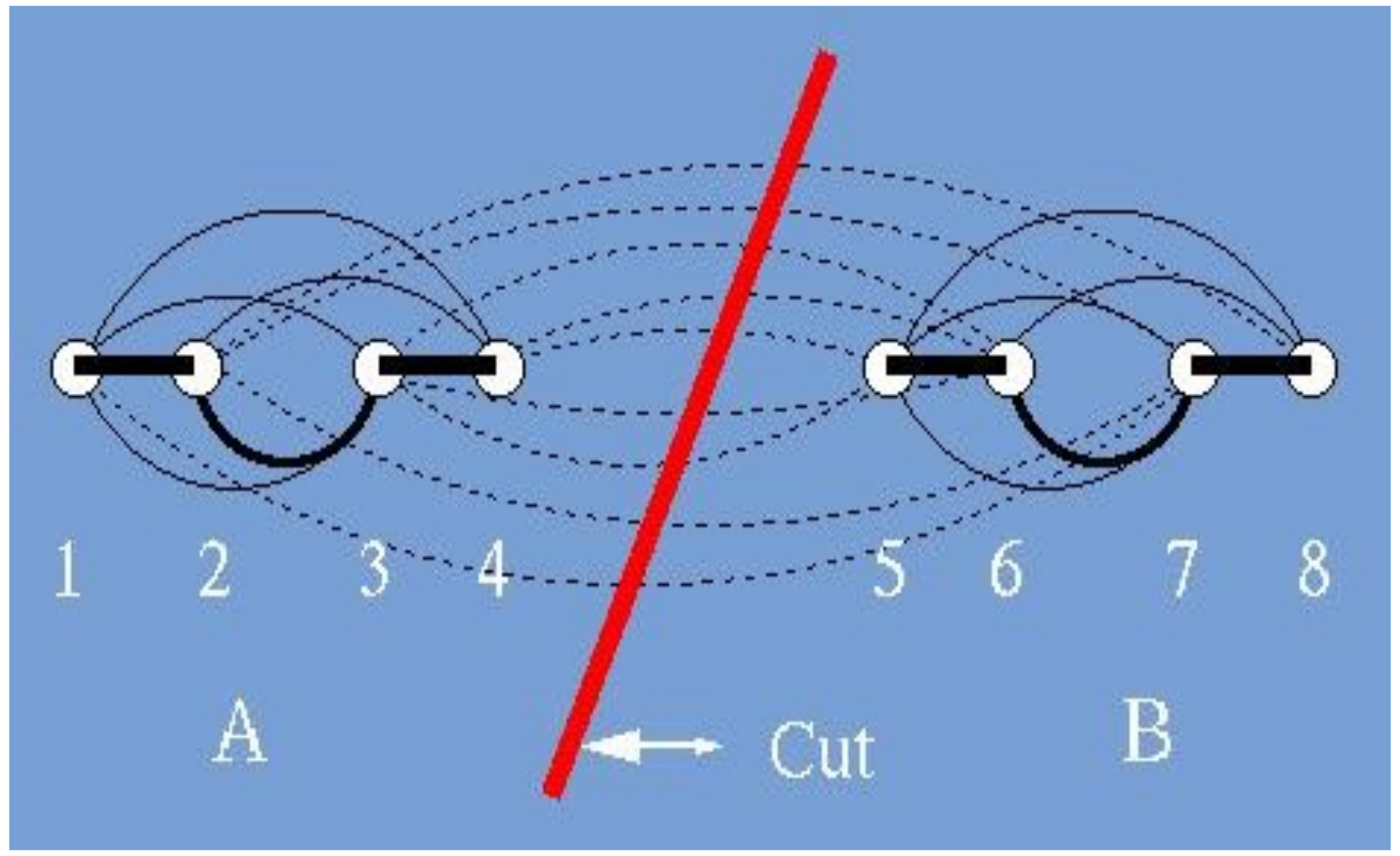
Drawbacks of Minimum Cut

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



* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Normalized Cut is often better



Normalize by the total volume of connections [Shi and Malik 2000]

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$ is the sum of all weights "associated" with A

Source: A. Oliva

Normalized Cut As Generalized Eigenvalue problem

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

After lots of simplification, can show that this is:

$$Ncut(A,B) = \frac{y^T(D-W)y}{y^T D y}, \quad \text{with } y_i \in \{1,-b\},\ y^T D1 = 0.$$

where W is the affinity matrix and D is a diagonal degree matrix.

For detailed derivation: http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Source: A. Oliva

Normalized cuts

Want to solve integer program:

Binary

Binary

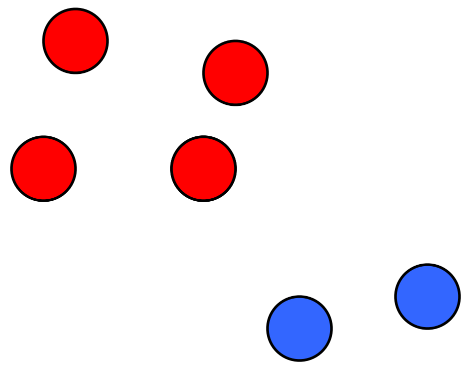
$$Ncut(A,B) = \frac{y^T(D-W)y}{y^T D y}, \quad \text{with } y_i \in \{1,-b\}, y^T D 1 = 0.$$

- Use a continuous relaxation!
- They show that the 2nd smallest eigenvector solution y is an approximate solution to the original normalized cuts problem.
- Choose a quantization threshold that maximizes the criterion — i.e all components of y above that threshold go to one, all below go to -b

http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Source: A. Oliva

An ideal case



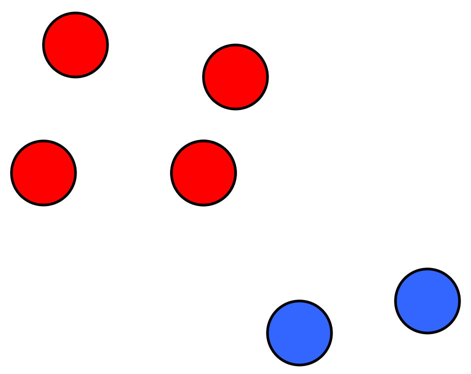
$$W=\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Affinity Matrix

What are the eigenvectors of this matrix?

On Spectral Clustering: Analysis and an algorithm. Andrew Y. Ng, Michael I. Jordan, Yair Weiss, NIPS 2001

An ideal case



$$W=\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow$$

Affinity Matrix

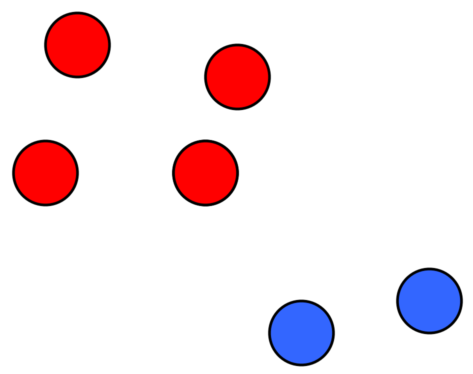
Eigenvectors:

$$\begin{array}{ccc} 1 & 0 & \\ 1 & 0 & \\ 1 & 0 & \\ 1 & 0 & \dots \\ 0 & 1 & \\ 0 & 1 & \\ \lambda=4 & \lambda=2 & \end{array}$$

On Spectral Clustering: Analysis and an algorithm. Andrew Y. Ng, Michael I. Jordan, Yair Weiss, NIPS 2001

An ideal case

But we do not know the ordering, so W with have some random permutation:



$$W=\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \longrightarrow$$

Affinity Matrix

Eigenvectors:

1	0
0	1
1	0
0	1
1	0
1	0
$\lambda=4$	$\lambda=2$

On Spectral Clustering: Analysis and an algorithm. Andrew Y. Ng, Michael I. Jordan, Yair Weiss, NIPS 2001

Eigenvectors and blocks

- Block weight matrices have block eigenvectors:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \underset{\lambda_1 = 2}{\begin{bmatrix} .71 \\ .71 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{\lambda_2 = 2}{\begin{bmatrix} 0 \\ 0 \\ .71 \\ .71 \end{bmatrix}}$$

$\lambda_1 = 2$ $\lambda_2 = 2$ $\lambda_3 = 0$

$\lambda_4 = 0$

- Near-block matrices have near-block eigenvectors:

$$\begin{bmatrix} 1 & 1 & .2 & 0 \\ 1 & 1 & 0 & -.2 \\ .2 & 0 & 1 & 1 \\ 0 & -.2 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \underset{e1}{\begin{bmatrix} .71 \\ .69 \\ .14 \\ 0 \end{bmatrix}} \quad \underset{e2}{\begin{bmatrix} 0 \\ -.14 \\ .69 \\ .71 \end{bmatrix}}$$

$\lambda_1 = 2.02$ $\lambda_2 = 2.02$ $\lambda_3 = -0.02$

$\lambda_4 = -0.02$

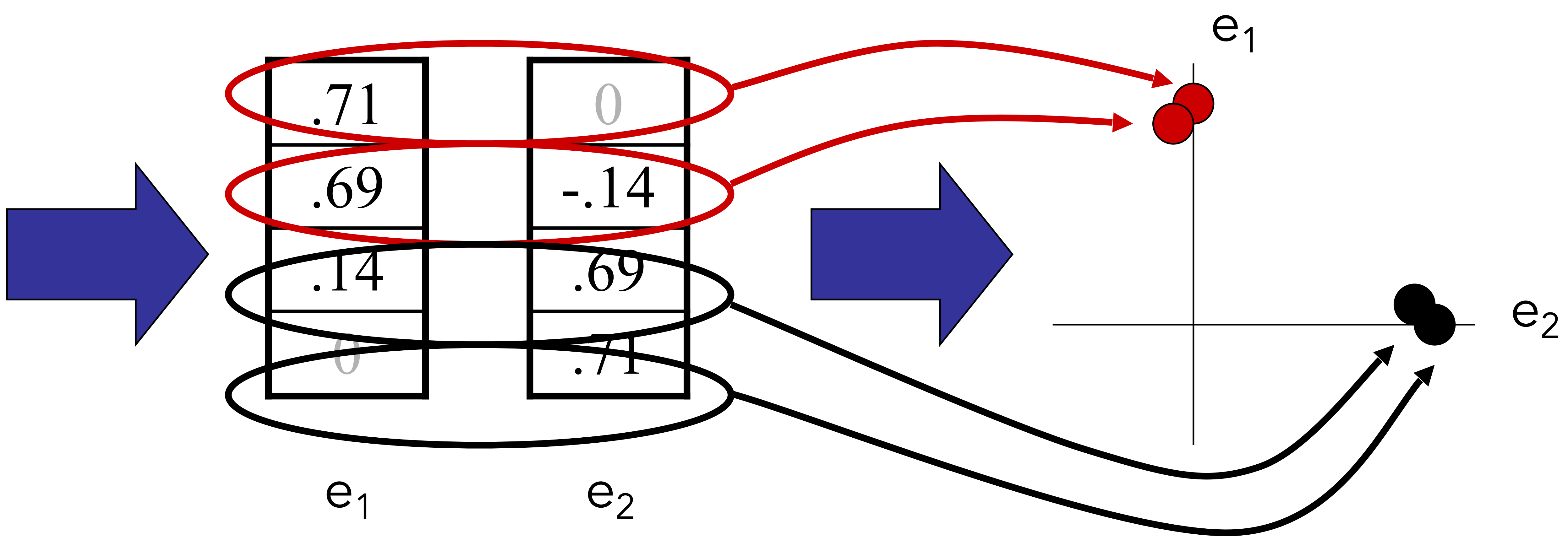
e1 e2

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Spectral Space

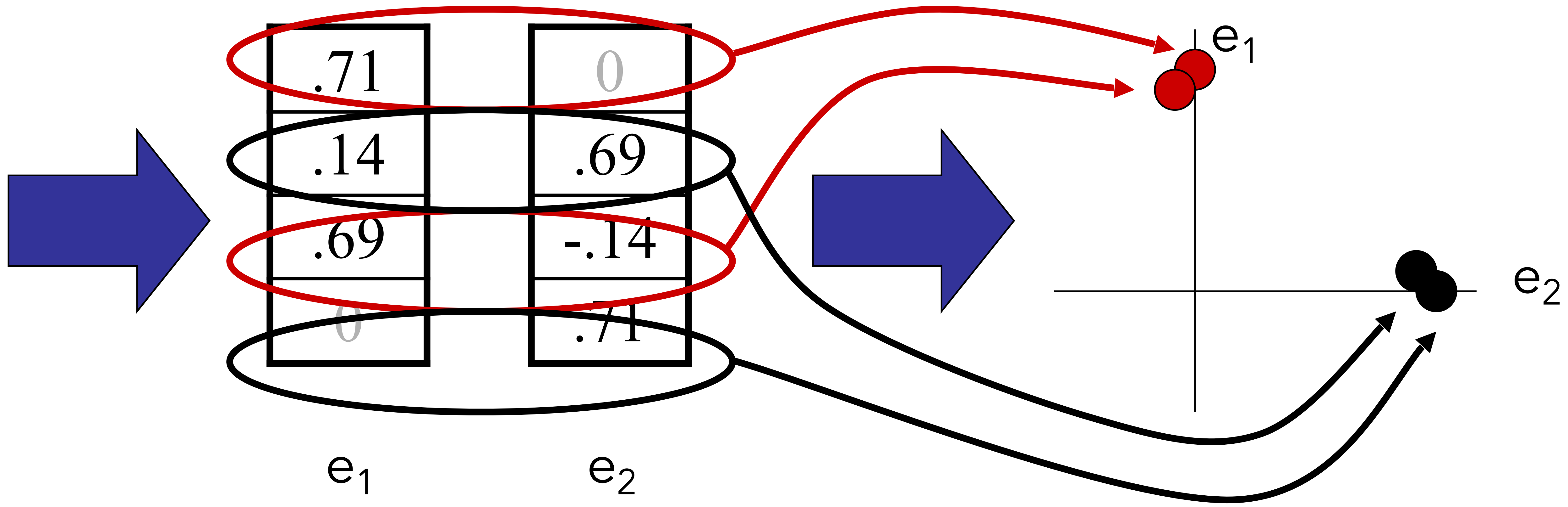
Can put items into blocks by eigenvectors:

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1

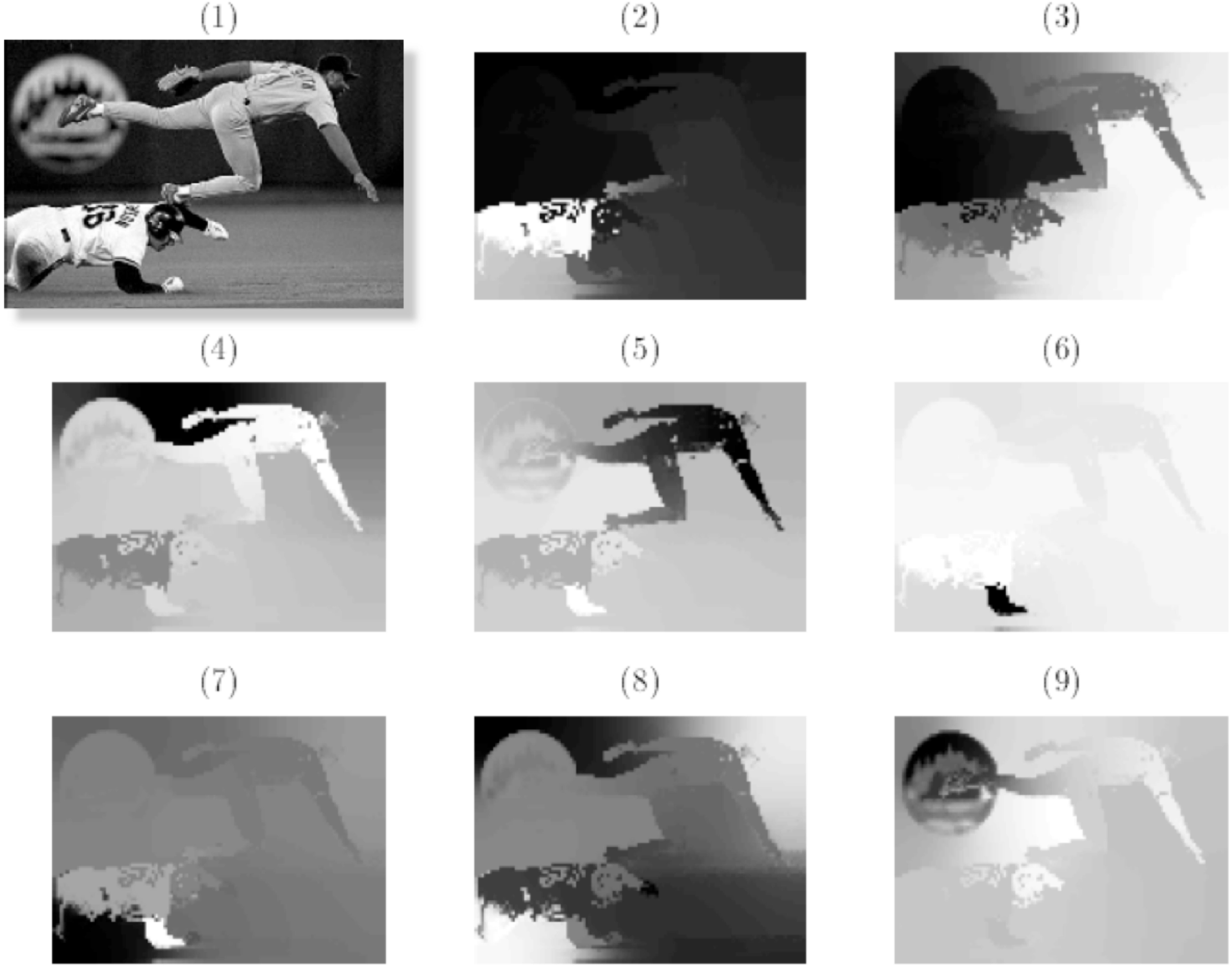


Clusters clear regardless of row ordering:

1	.2	1	0
.2	1	0	1
1	0	1	-.2
0	1	-.2	1



* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University



The eigenvectors correspond the 2nd smallest to the 9th smallest eigenvalues

Many different spectral clustering methods…

Goal: Given data points X1, …, Xn and similarities w(Xi,Xj), partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

1- Get vectors of **data**

2- Build a **similarity** graph

3- Calculate eigenvectors

4- Apply threshold to largest eigenvectors

1- Get vectors of **data**

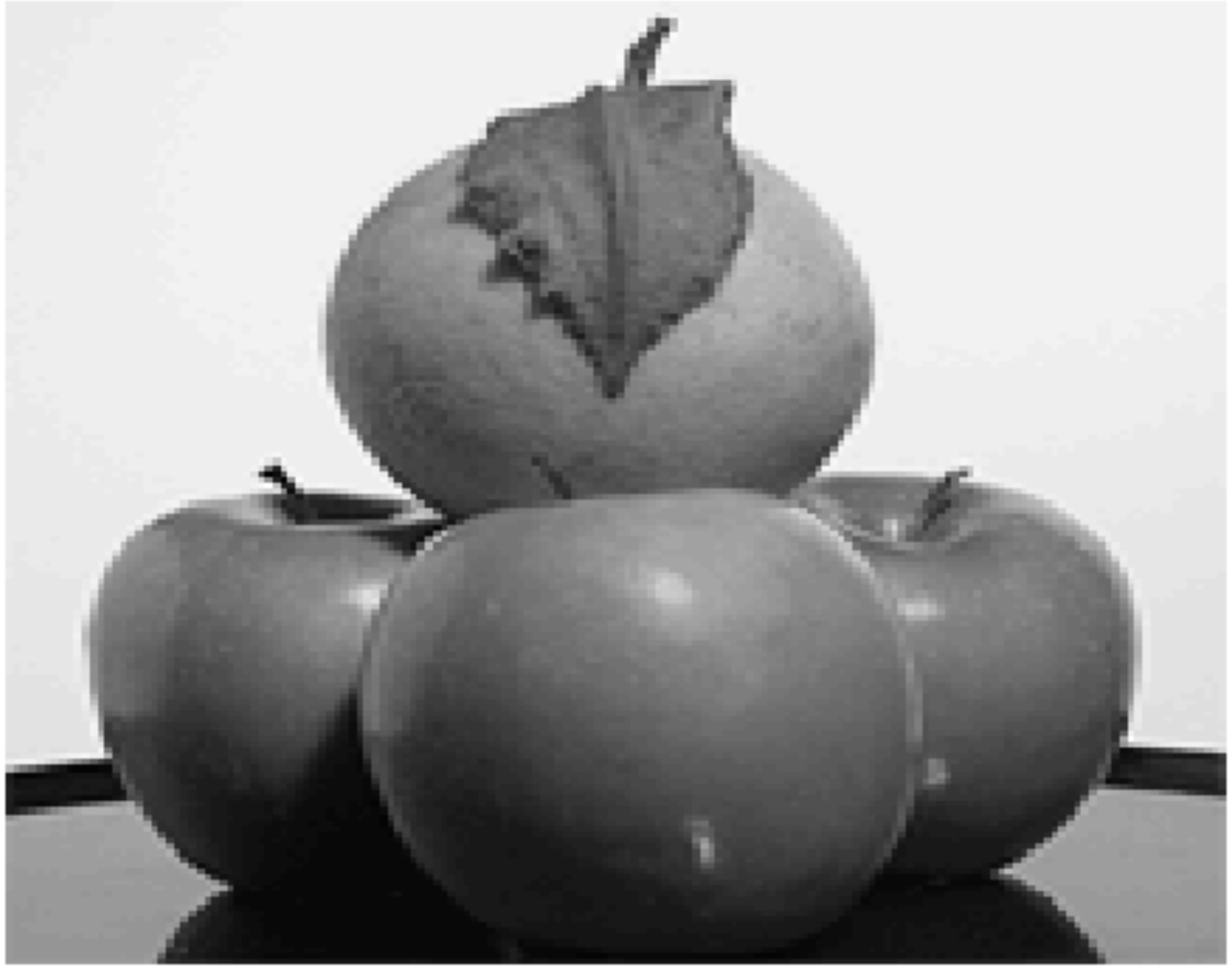
2- Build normalized cost matrix

3- Get eigenvectors with smallest eigenvalues

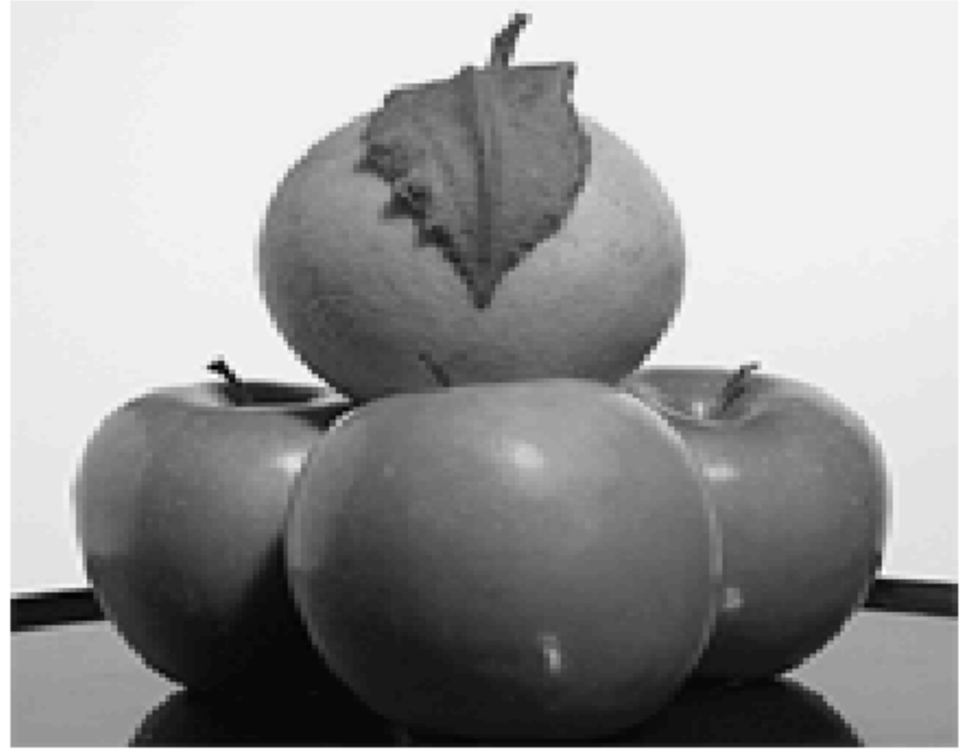
4- Apply threshold

Shi & Malik

Source: A. Oliva



Source: A. Oliva

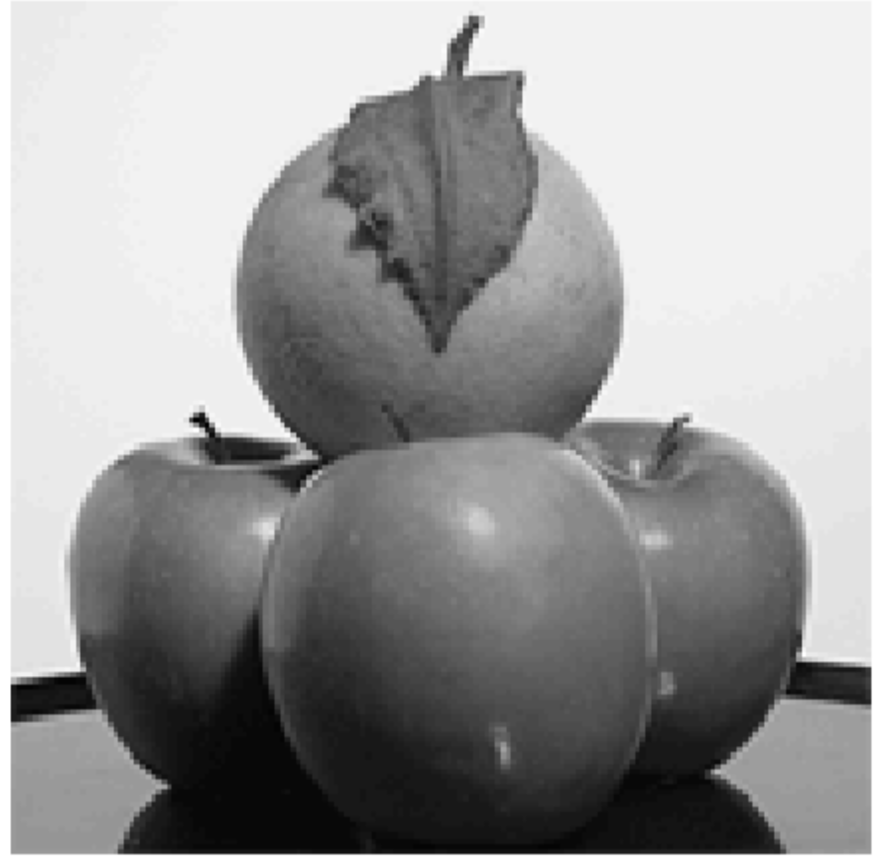


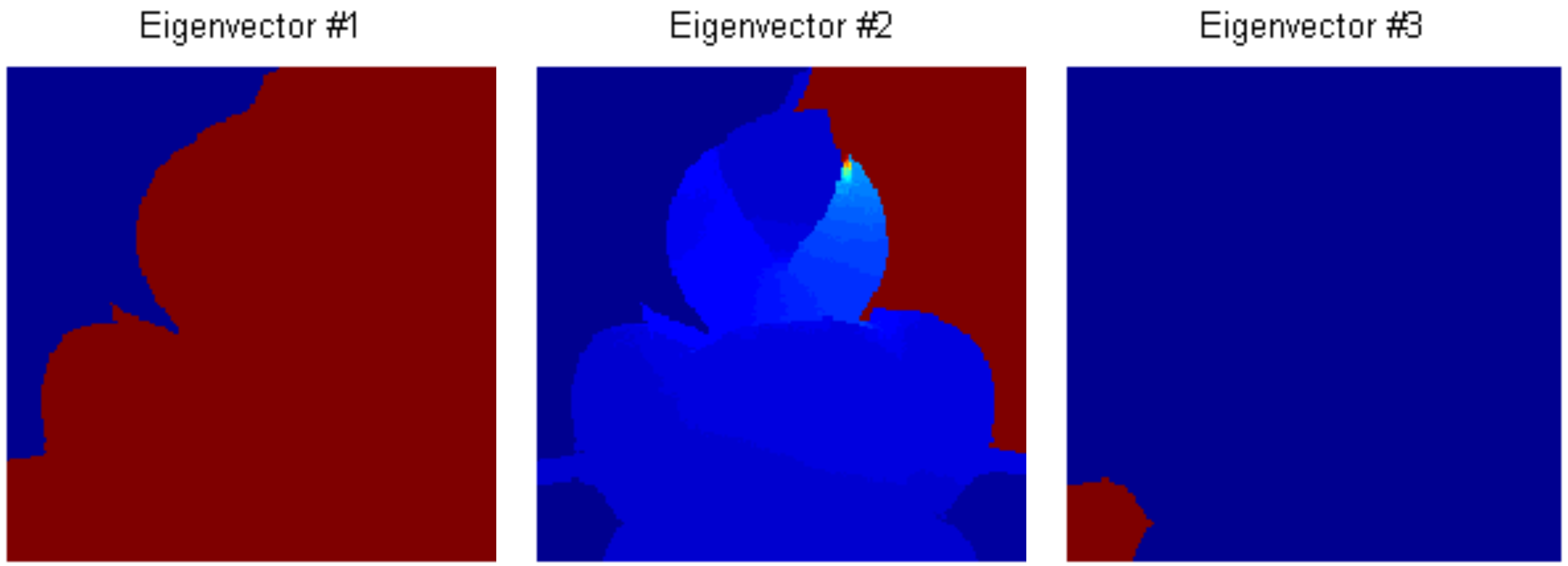
The Eigenvectors

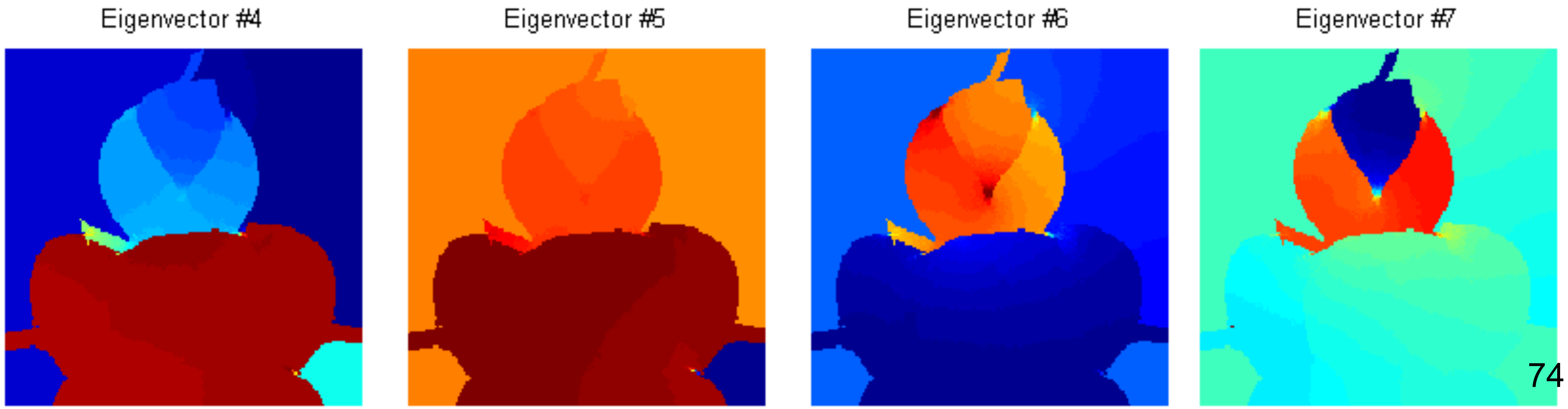
Eigenvector #7

Source: A. Oliva

Normalized cut







Source: A. Oliva

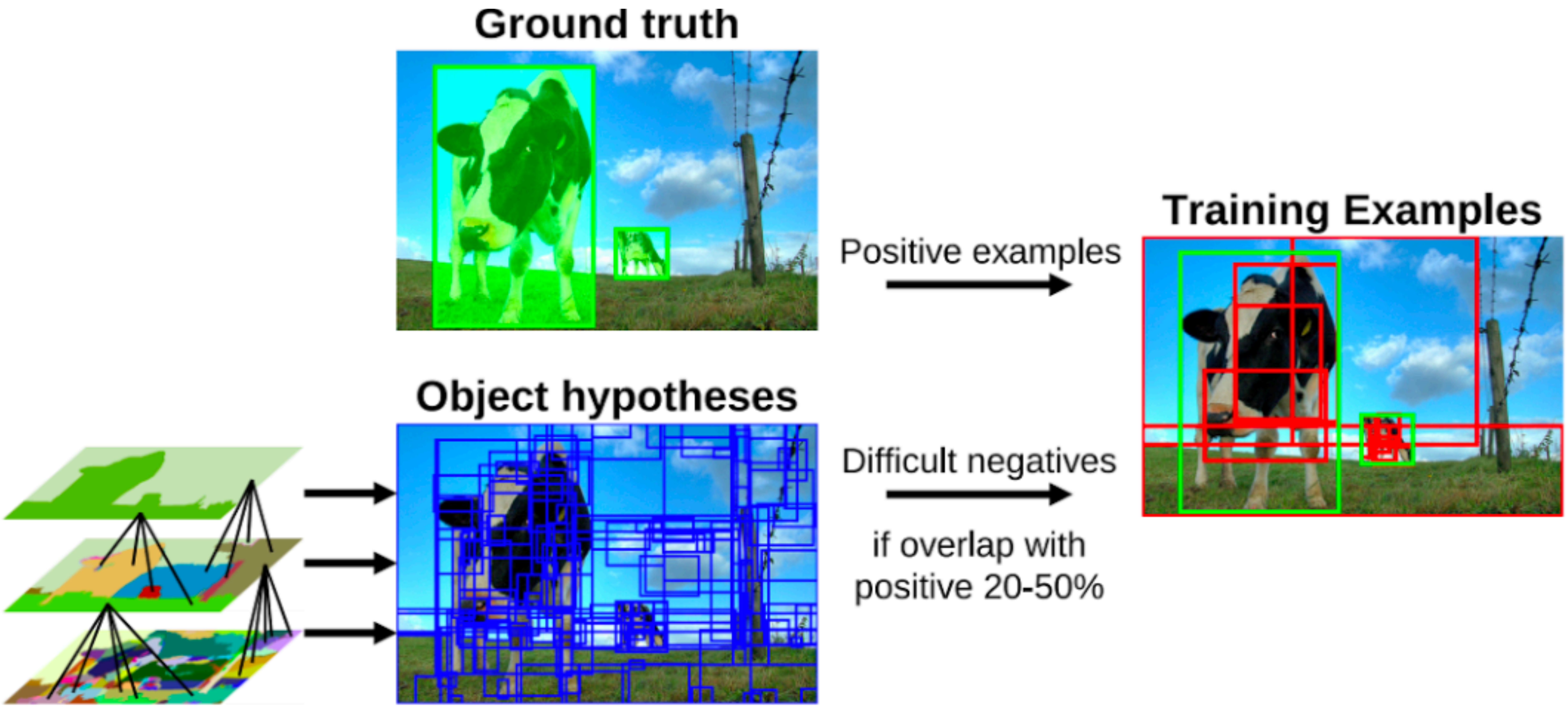
Normalized cut

Source: A. Oliva

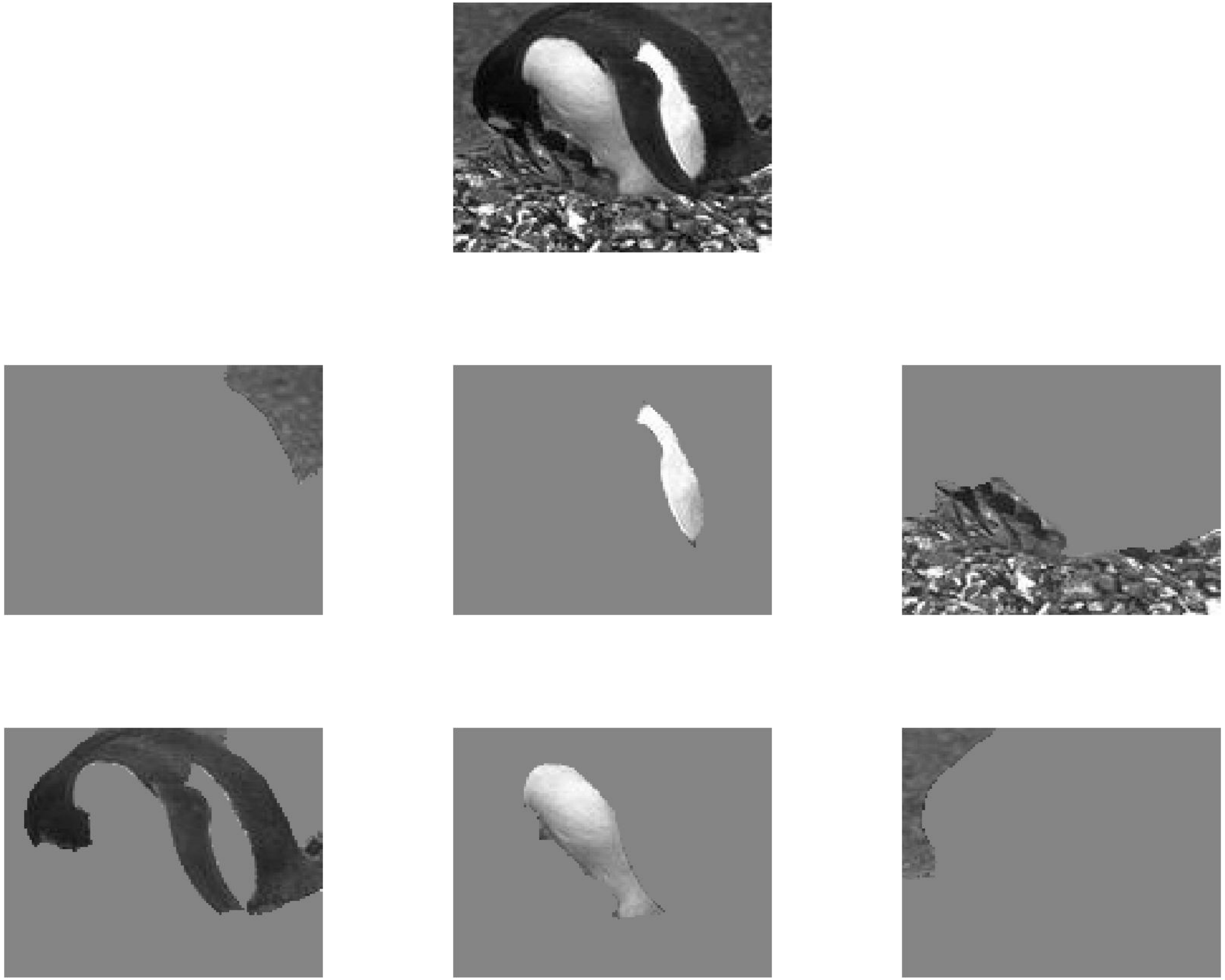
Normalized cut

Source: A. Oliva

Application: selective search



[Uijlings et al., "Selective Search for Object Recognition", 2013]



Min-cut for user-in-the-loop segmentation

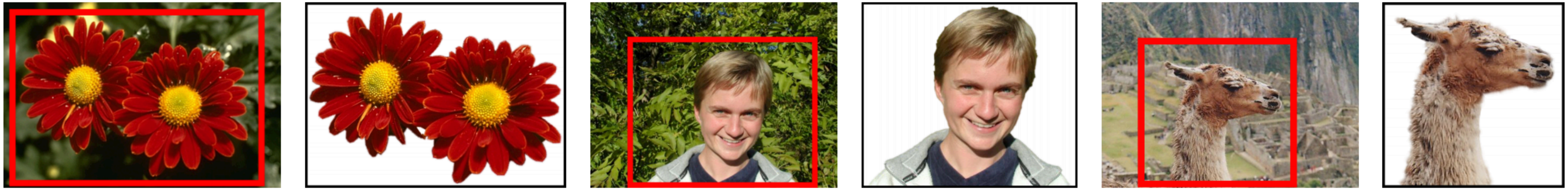
User input　　Min-cut segmentation　　Solution after iteration

- Formulation makes it easy to add local evidence
  - How much does this pixel look like the foreground?

- Formulate as probabilistic graphical model.

[Rother et al., “GrabCut”, 2004]

Min-cut for user-in-the-loop segmentation

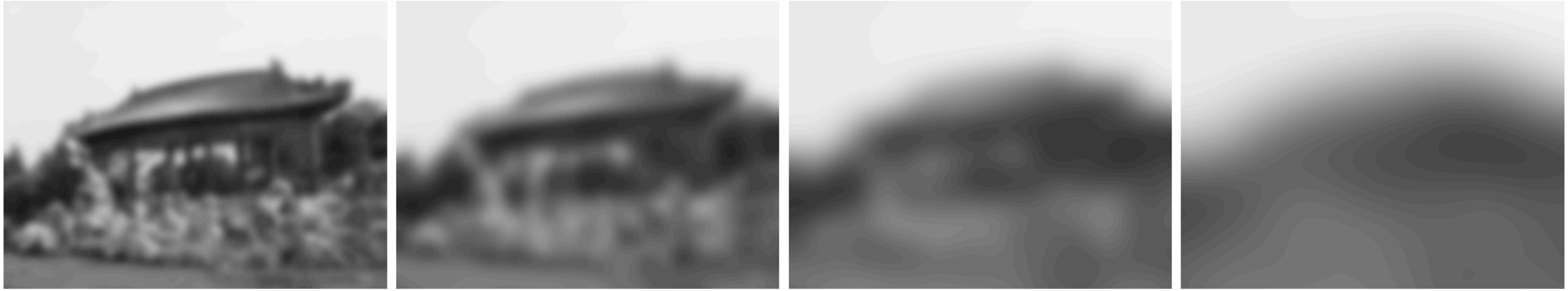


[Rother et al., “GrabCut”, 2004]

What about edges?

Image filters not "edge-aware"

Gaussian filter



$\sigma = 4$ $\sigma = 8$ $\sigma = 16$ $\sigma = 32$

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|)\ I_{\mathbf{q}}$$

[Paris et al. "A Gentle Introduction to Bilateral Filtering and its Applications", 2008]

Bilateral filter

Gaussian filter

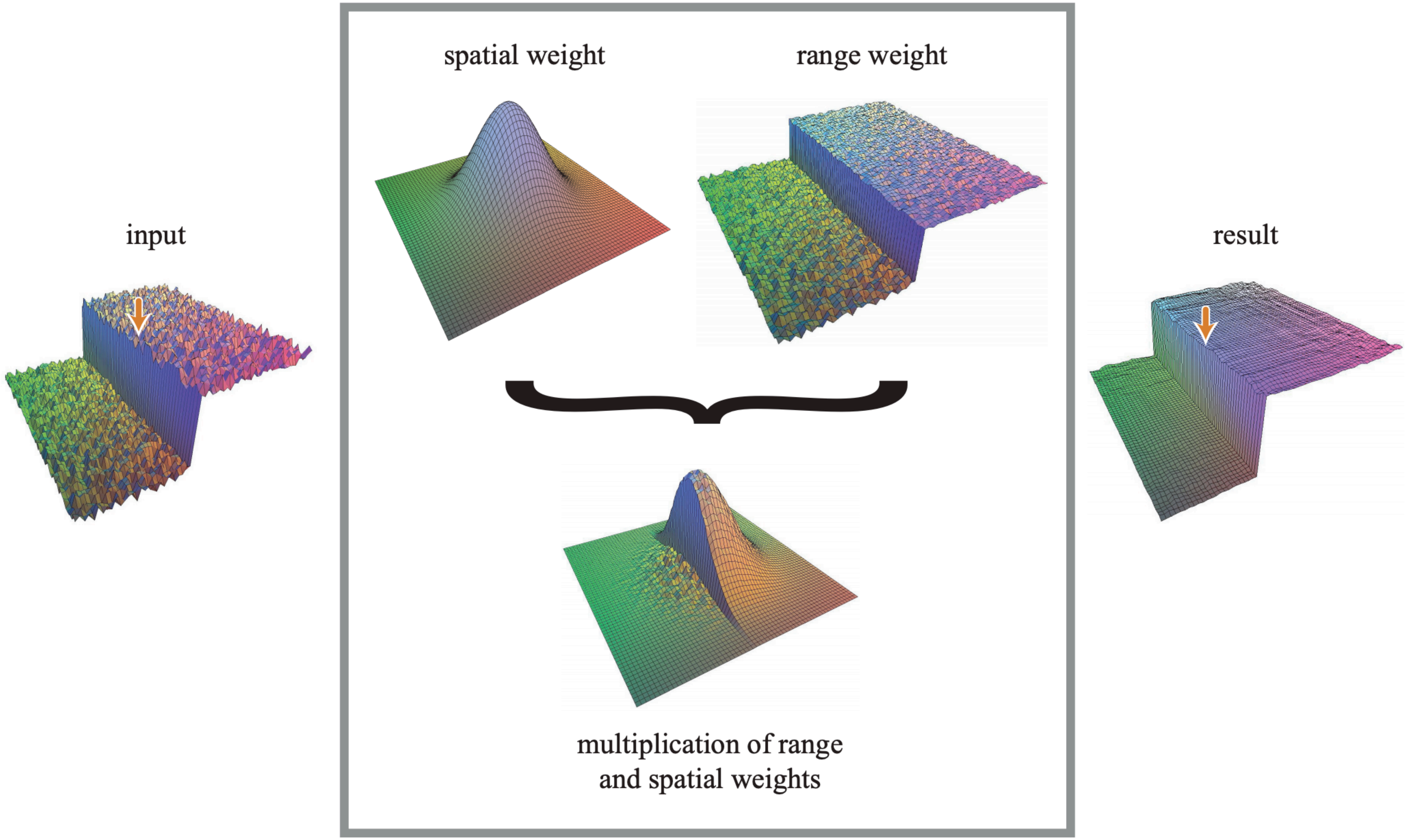
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|)\, I_{\mathbf{q}}$$

Bilateral filter

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})\, I_{\mathbf{q}}$$

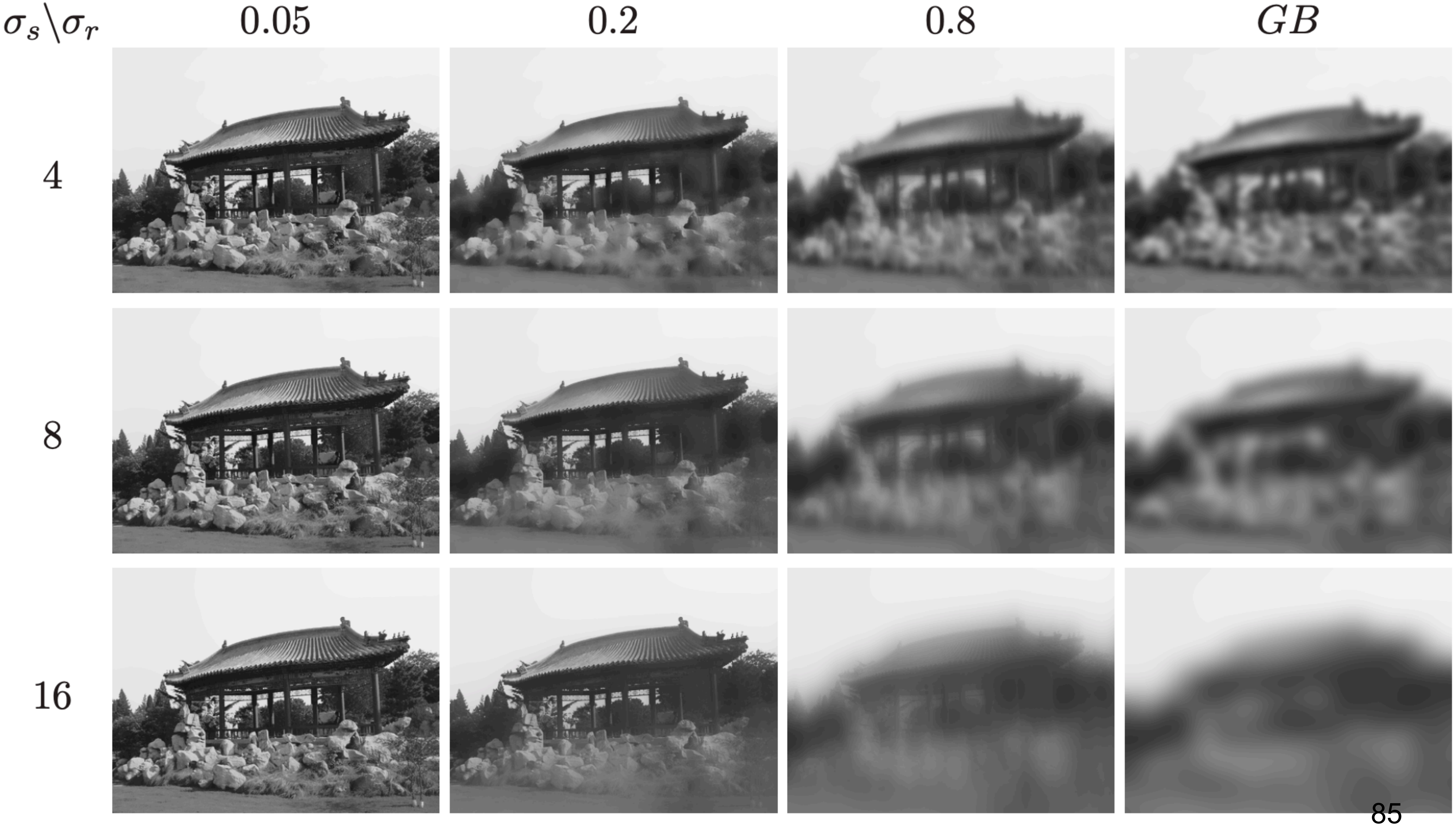
[Paris et al. “A Gentle Introduction to Bilateral Filtering and its Applications”, 2008]

Bilateral filter



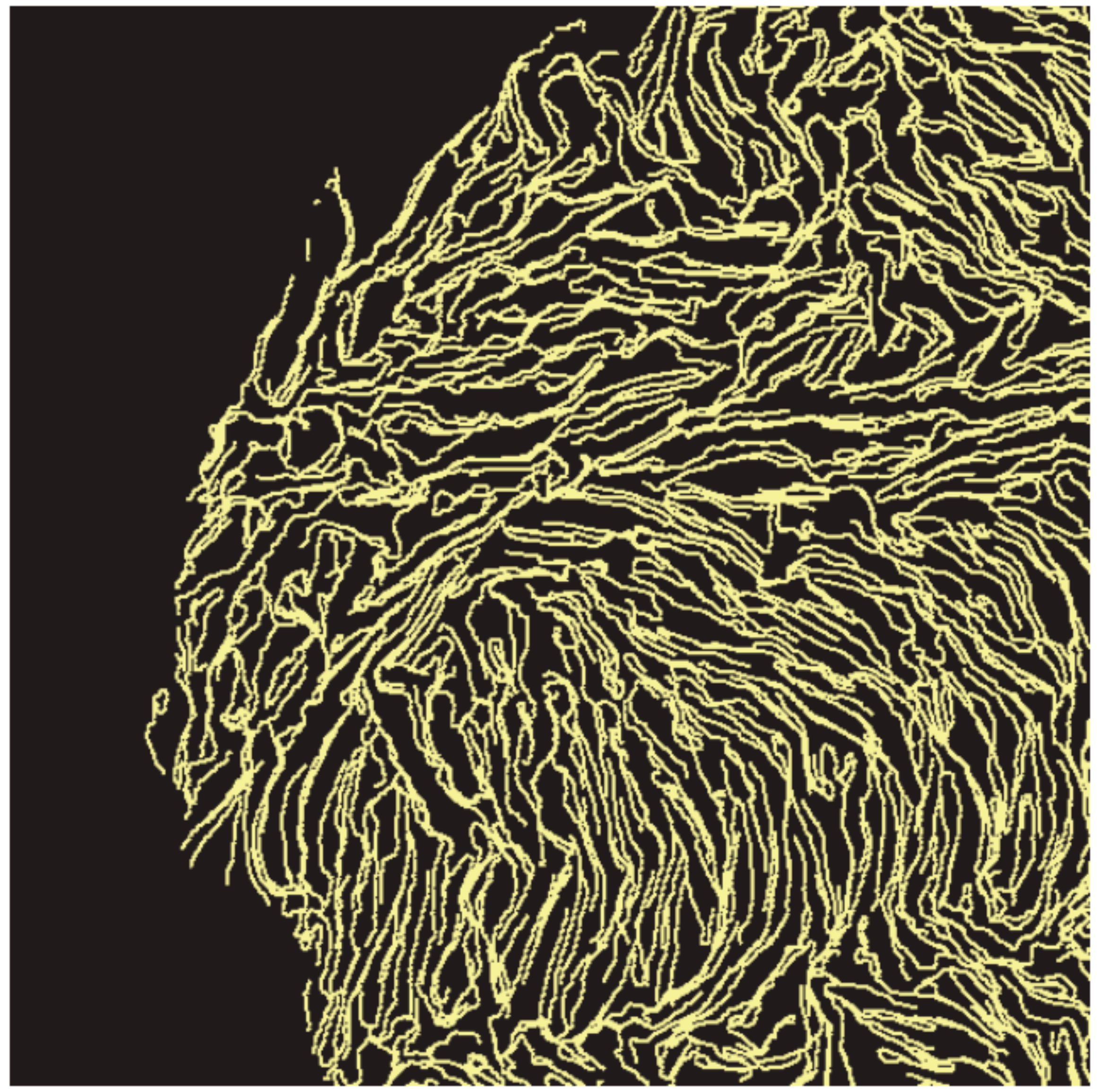
[Paris et al. “A Gentle Introduction to Bilateral Filtering and its Applications”, 2008]

Bilateral filter



[Paris et al. “A Gentle Introduction to Bilateral Filtering and its Applications”, 2008]

Bilateral filter + edge detection



Without bilateral

With bilateral

[Paris et al. “A Gentle Introduction to Bilateral Filtering and its Applications”, 2008]

Neural nets that operate in "bilateral space"

12 megapixel 16-bit linear input
(tone-mapped for visualization)

tone-mapped with HDR+
400 – 600 ms

processed with our algorithm
61 ms, PSNR = **28.4 dB**

[Gharbi et al. Deep Bilateral Learning for Real-Time Image Enhancement", 2017]

Next lecture: recent advances