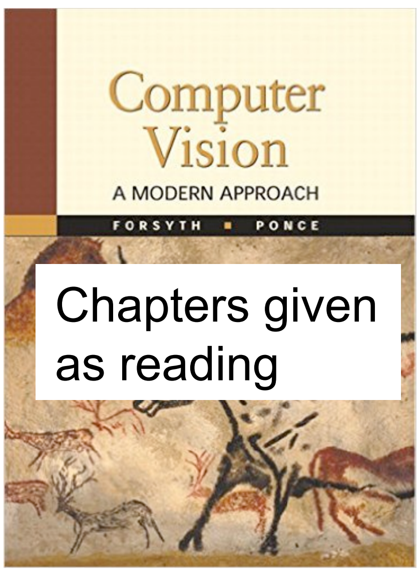
6.819 / 6.869: Advances in Computer Vision

Mid-level vision:

Edges, Segmentation, Grouping & Perceptual Organization

Website:

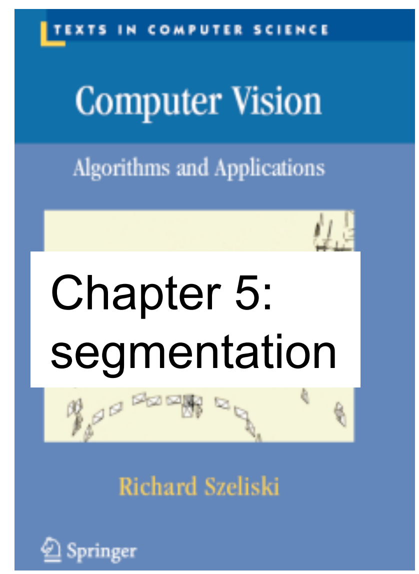
http://6.869.csail.mit.edu/fa15/



Chapters given as reading

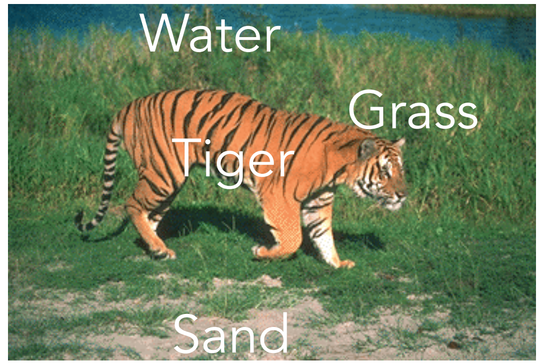
Instructor: Aude Oliva

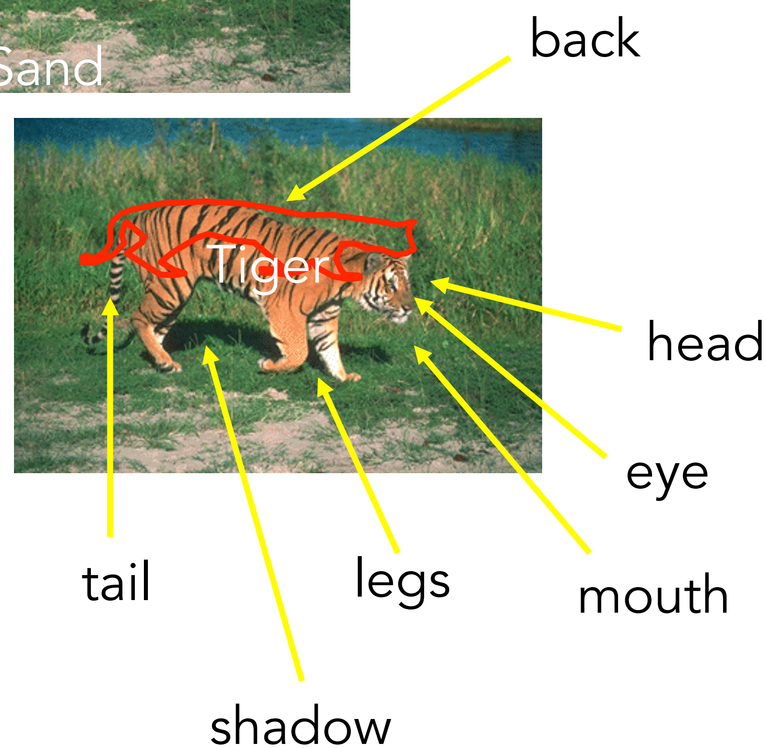
Lecture TR 9:30AM – 11:00AM
(Room 34-101)



Chapter 5: segmentation

From Pixels to Perception:
Mid-level operations of Segmentation and Grouping





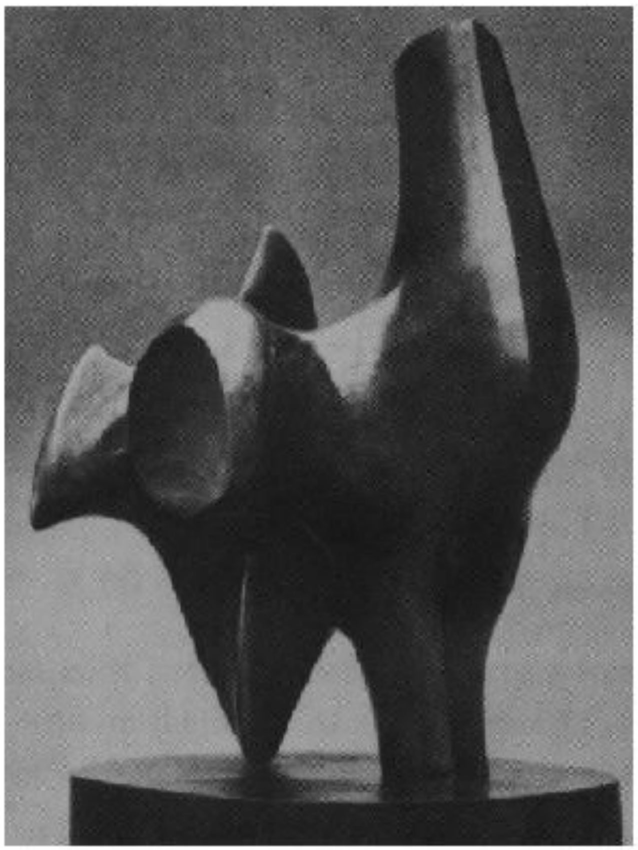
Emergence

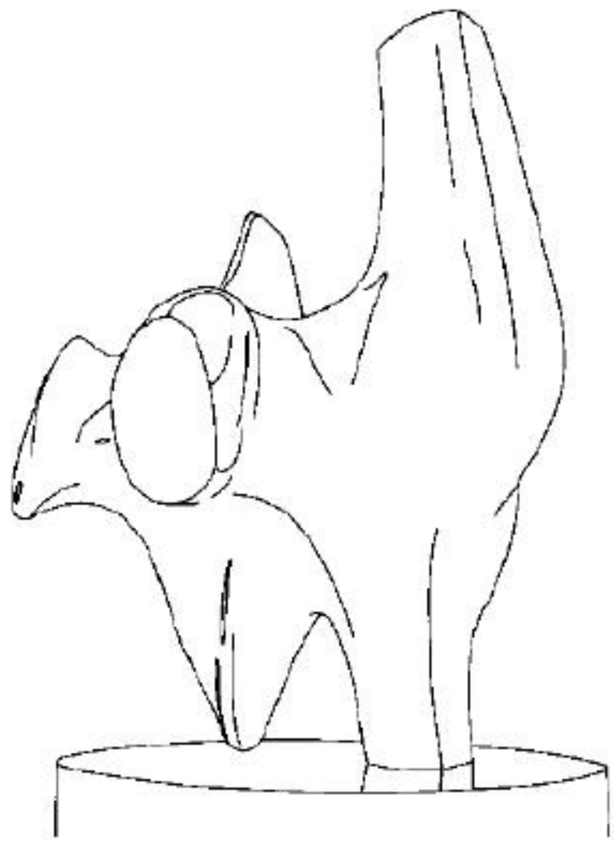
http://en.wikipedia.org/wiki/Gestalt_psychology



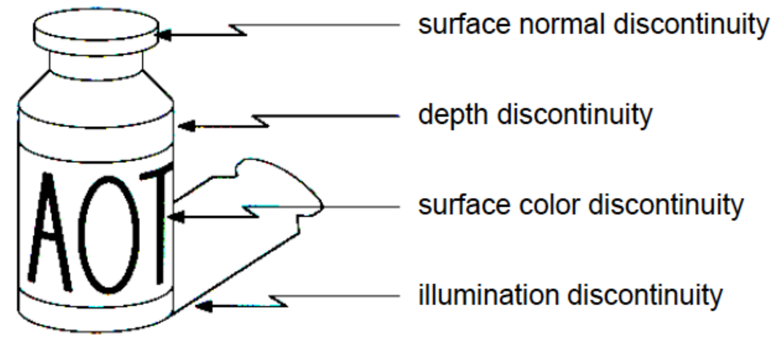


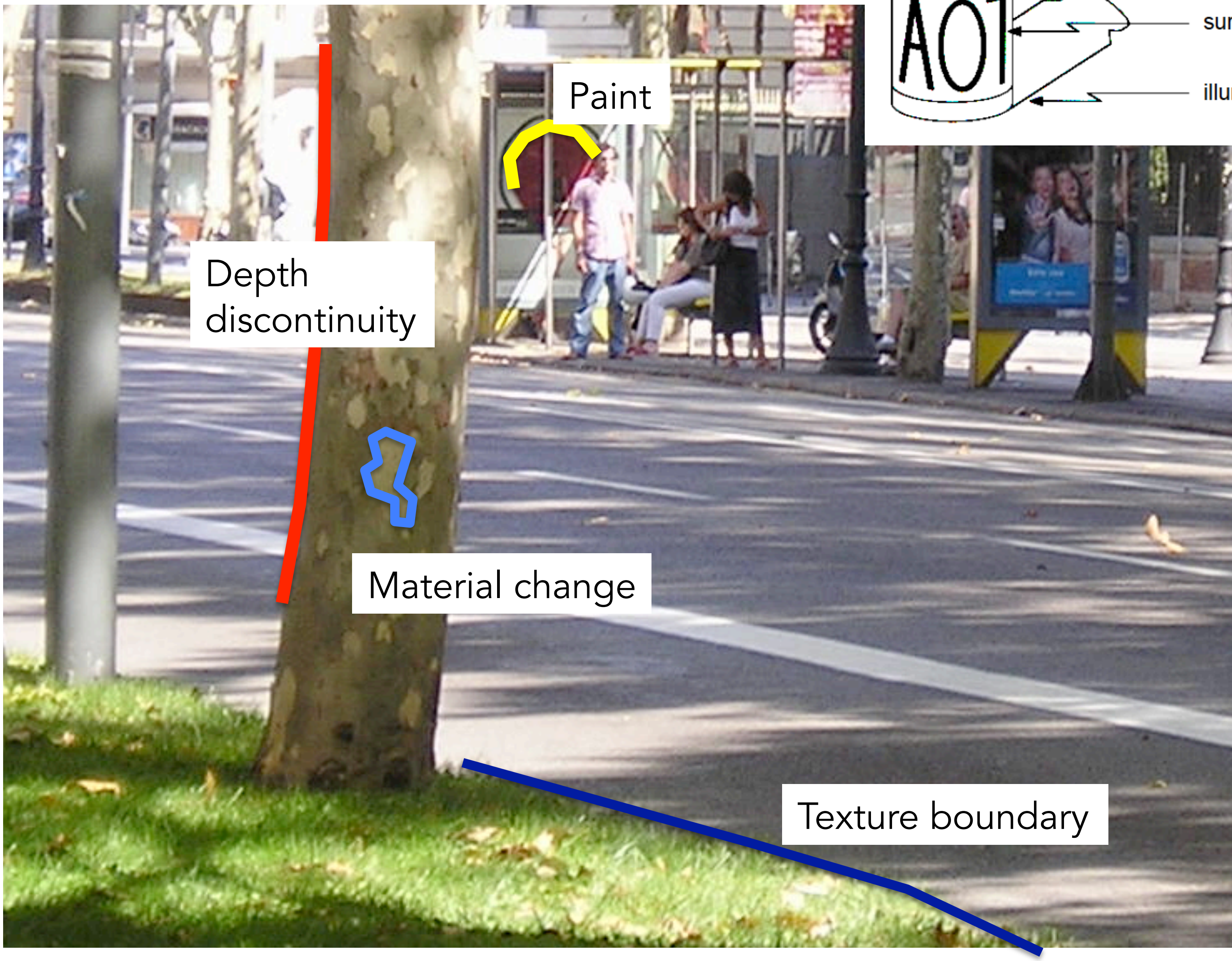
I. Edges



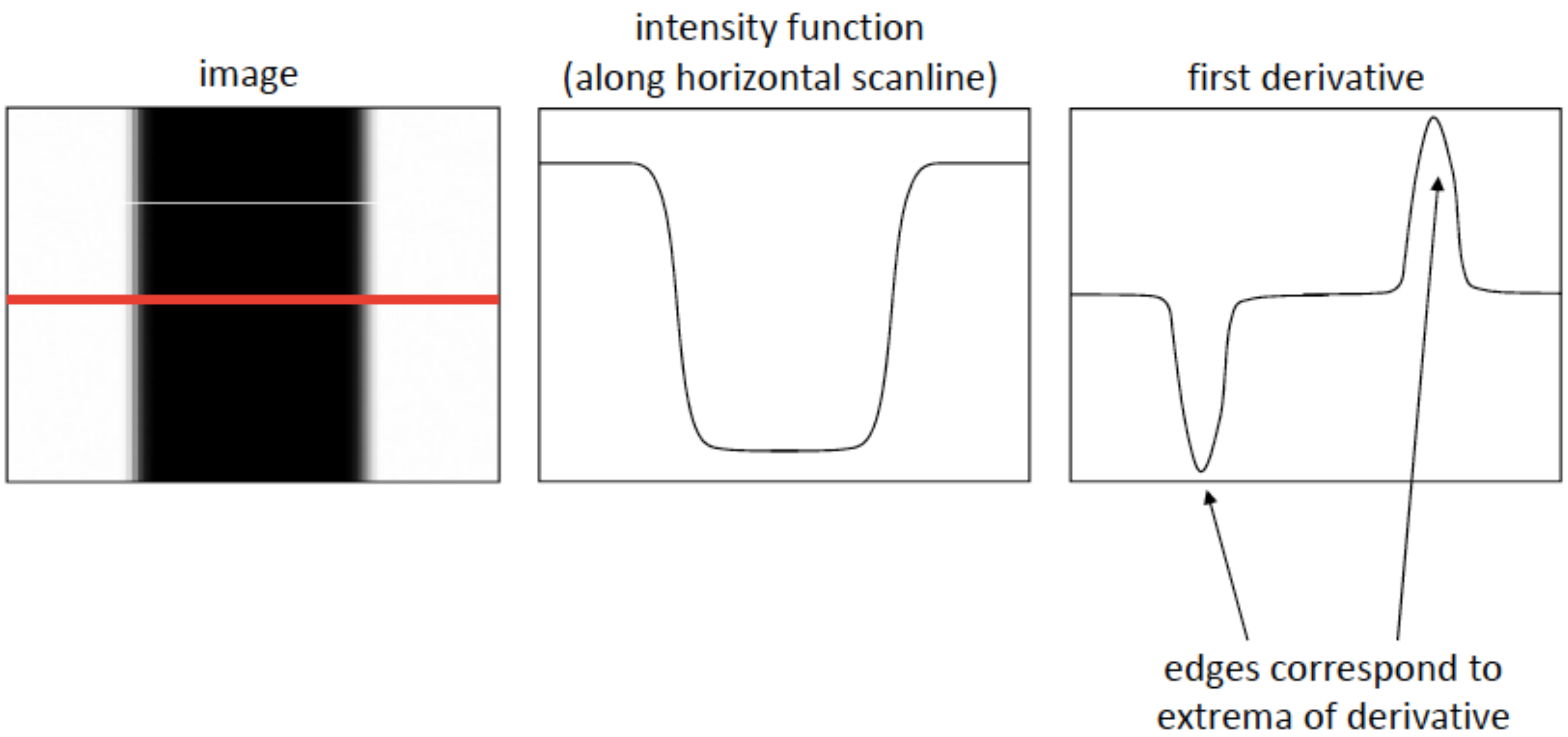


What is an edge?





Finding edges: Computing derivatives



Canny edge detector

edge(image,'canny')

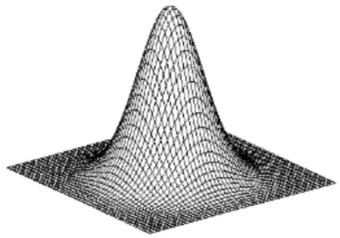
1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

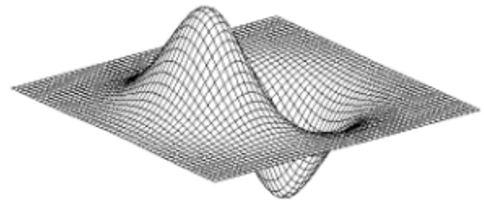
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

1: Filter Image with derivatives of Gaussian
2D edge detection filters



Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



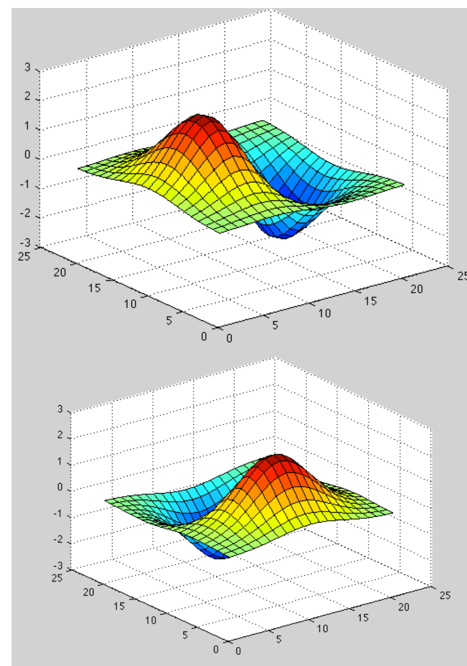
derivative of Gaussian (x)

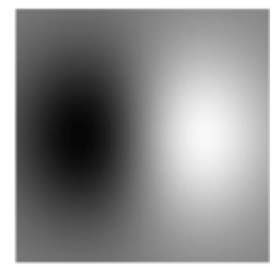
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

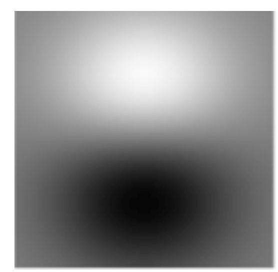
$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$h_y(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

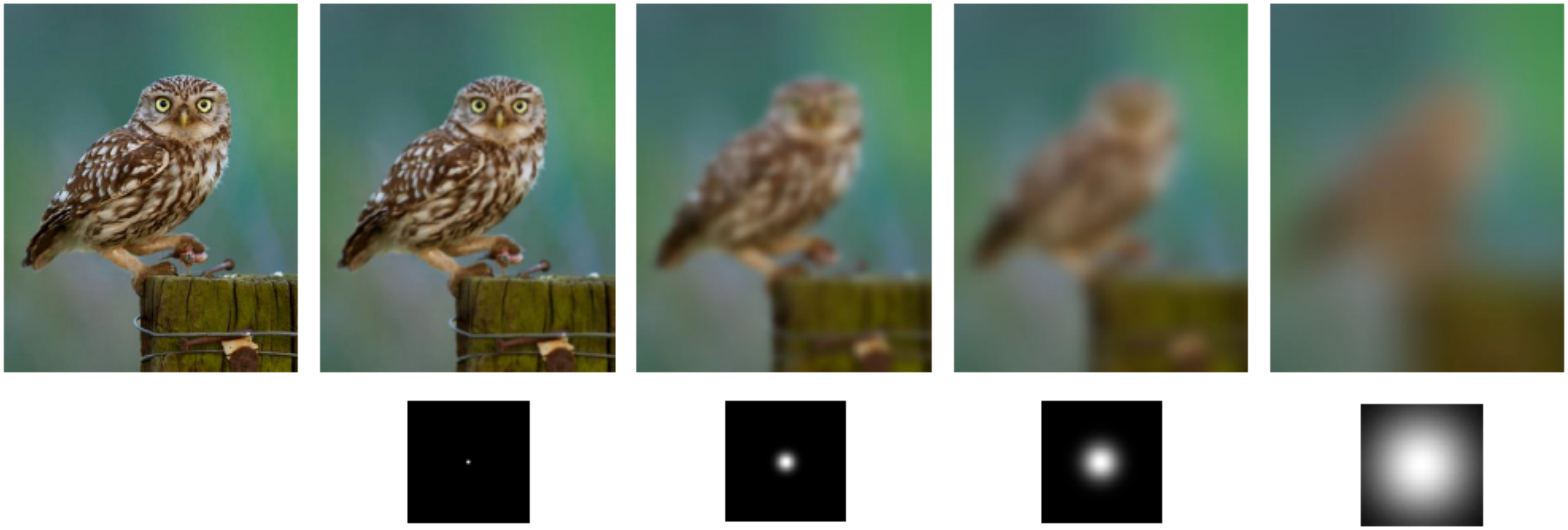
Scale



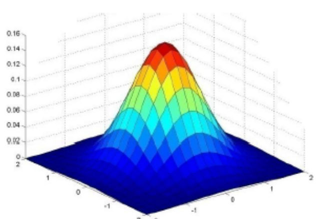


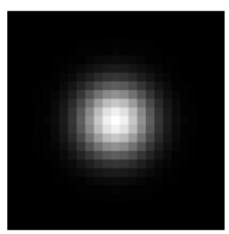


Gaussian filters



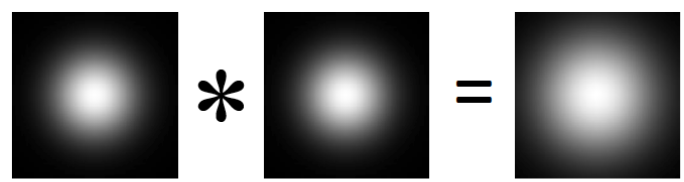
σ = 1 pixel　　σ = 5 pixels　　σ = 10 pixels　　σ = 30 pixels





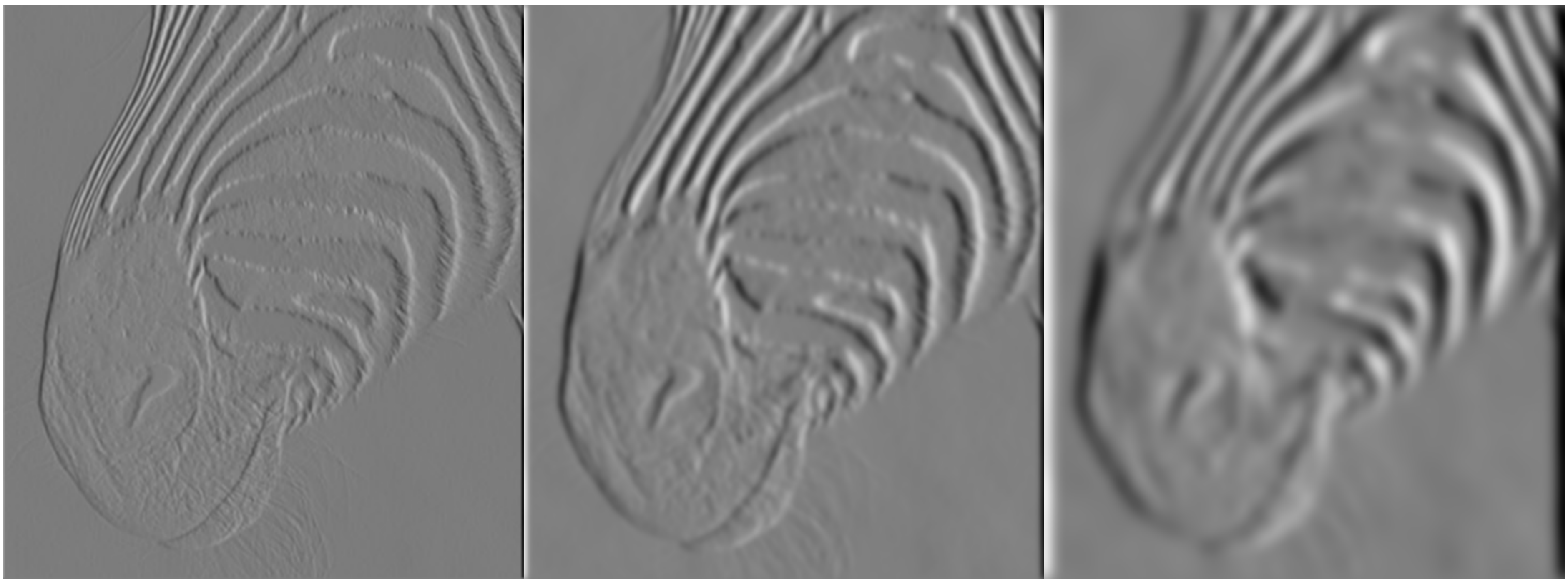
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Convolution with self is another Gaussian



– Convolving two times with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

1: Filter Image with derivatives of Gaussian
2D edge detection filters



1 pixel | 3 pixels | 7 pixels

Smoothing filters with different scales

The Sobel Operator: A common approximation of derivative of gaussian

- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value

Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

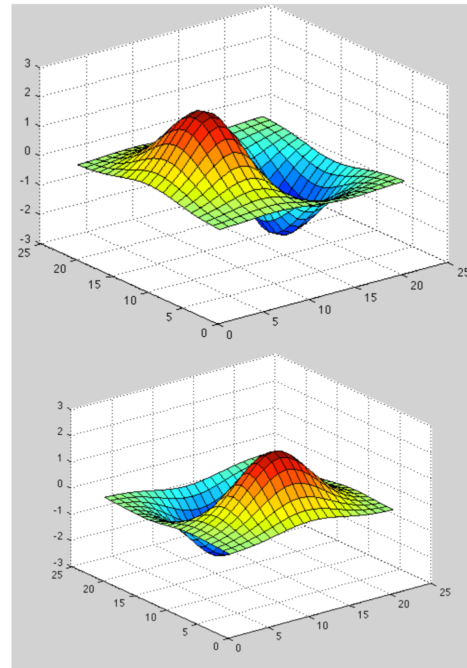
3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

2: Gradient: Find edge strength (magnitude) and direction (angle) of gradient

$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$h_y(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



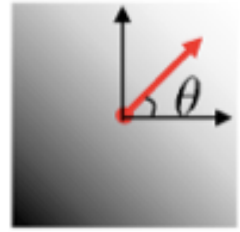
Magnitude: $h_x(x,y)^2 + h_y(x,y)^2$ *Edge strength*

Angle: $\arctan\left(\frac{h_y(x,y)}{h_x(x,y)}\right)$ *Edge normal*

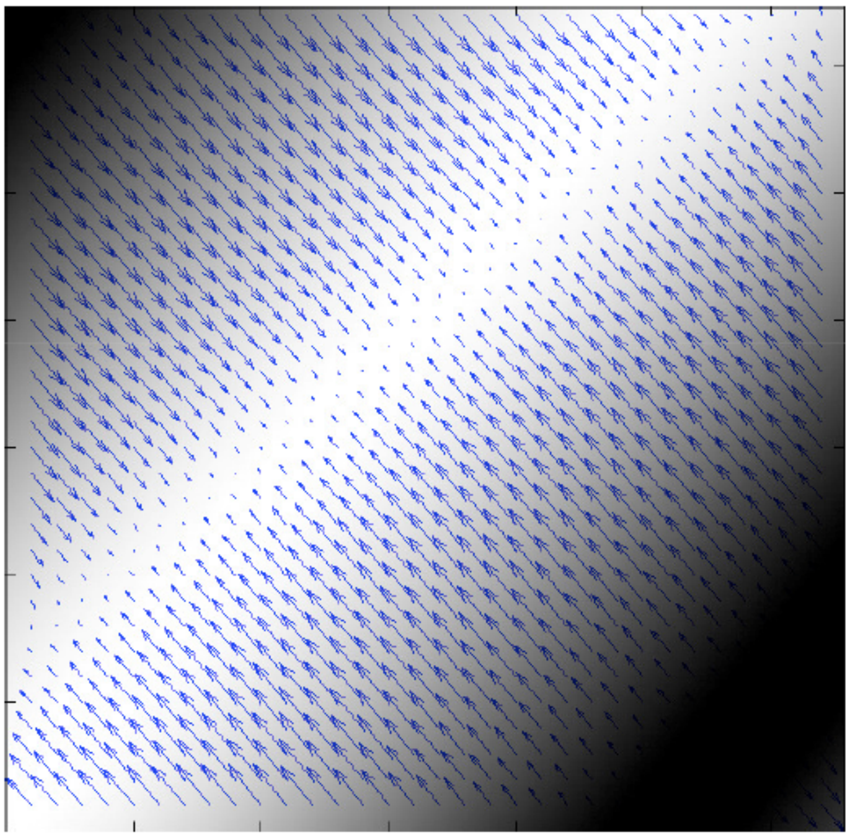
Image Gradient: gradient points in the direction of most rapid increase in intensity

$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



Can think of it as the slope of a 3D surface

Gradient at a single point (x,y) is a vector:

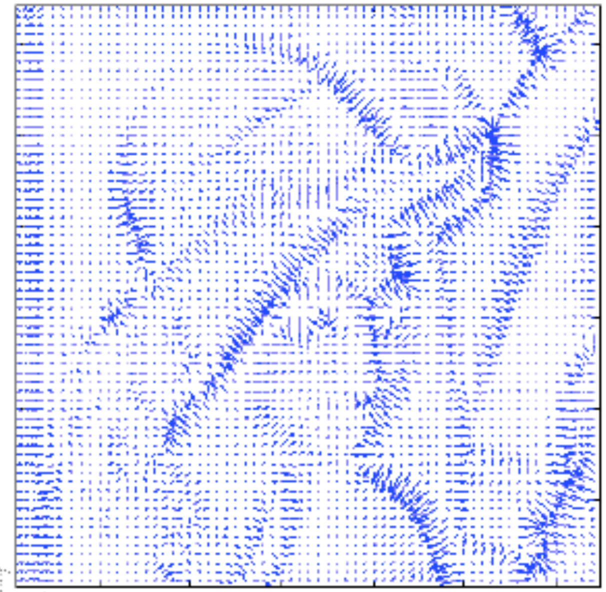
- Direction is the direction of maximum slope:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Length is the magnitude (steepness) of the slope

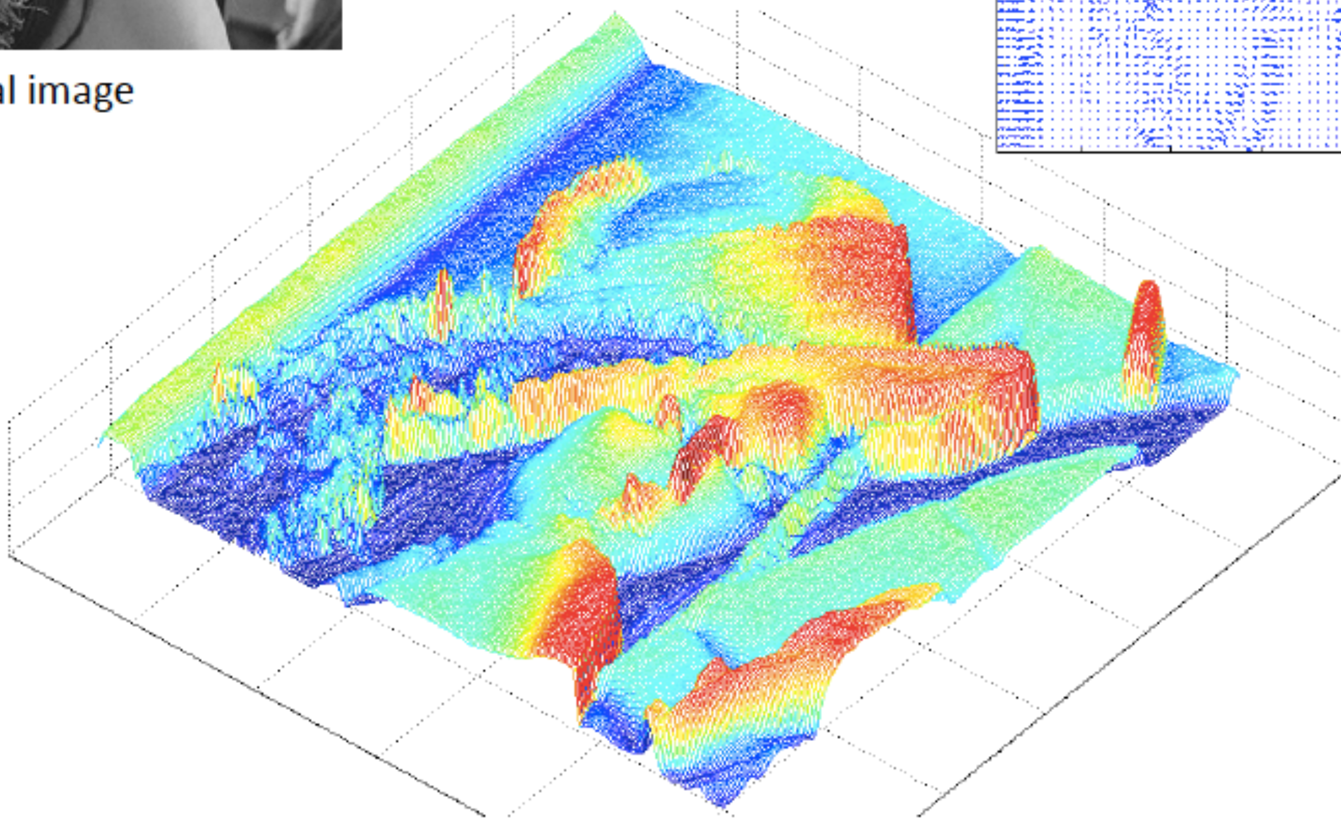
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

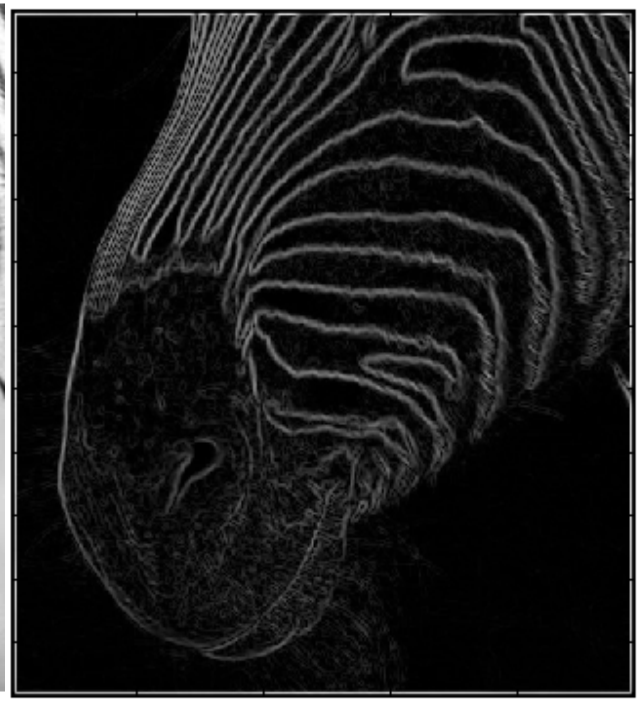
Original image



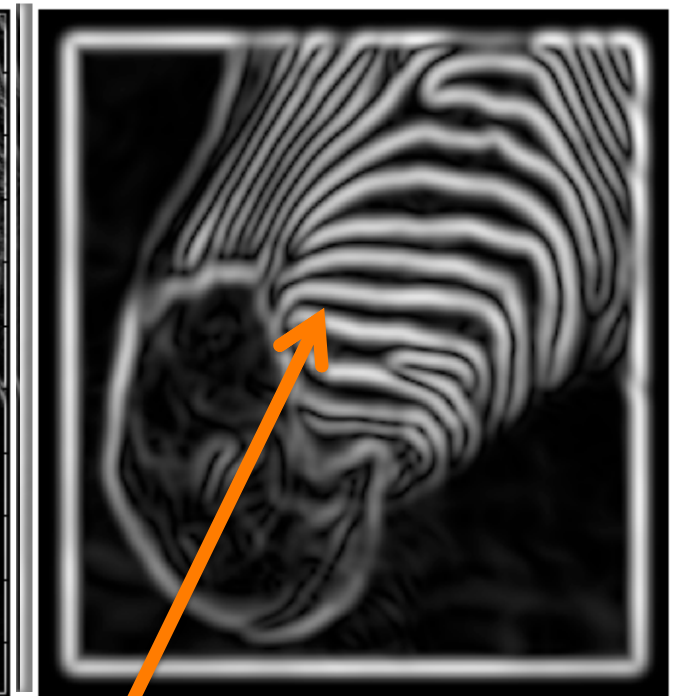
Gradient

3D plot of luminance





Gradient magnitudes at scale 1



Gradient magnitudes at scale 2

Issues:

1) The gradient magnitude at different scales is different; which should we choose?
2) **The gradient magnitude is large along thick trails; how do we identify the significant points?**
3) How do we link the relevant points up into curves?
4) Noise.

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Canny edge detector

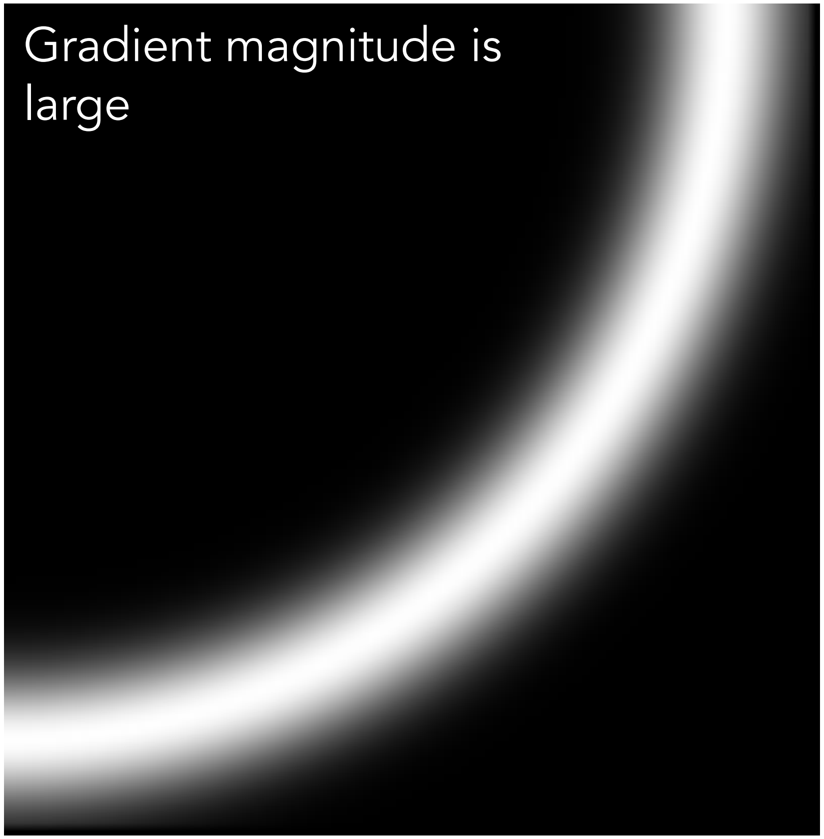
edge(image,'canny')

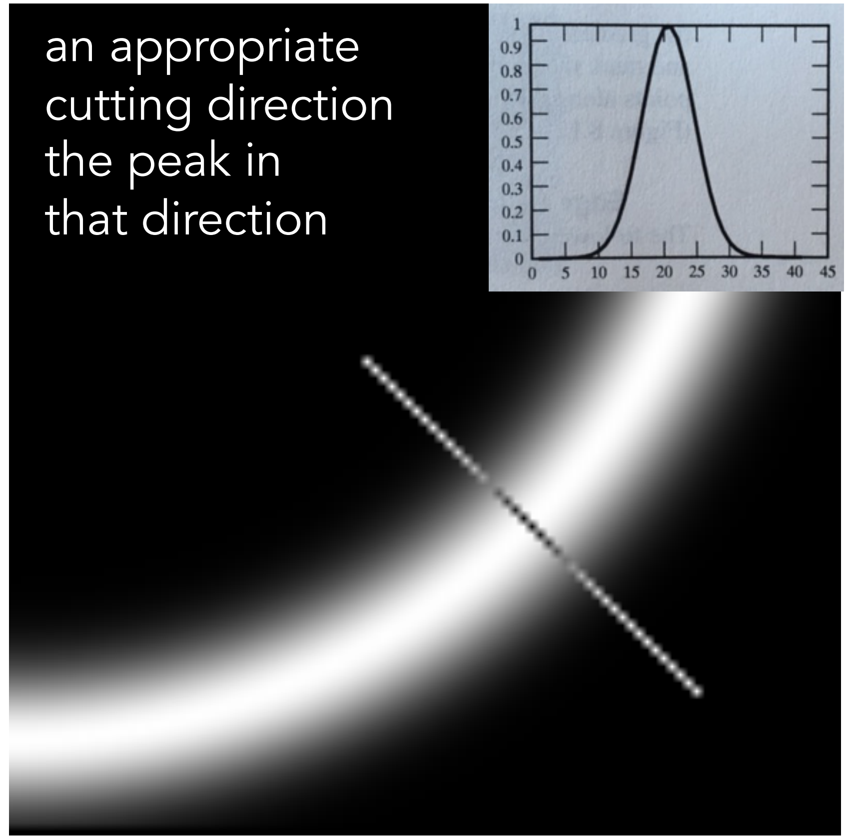
1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

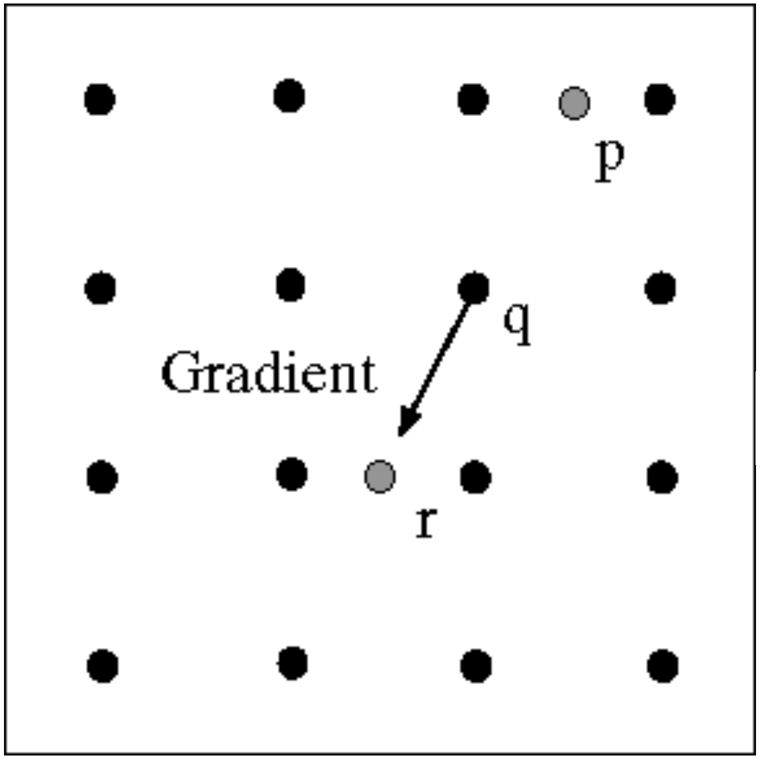


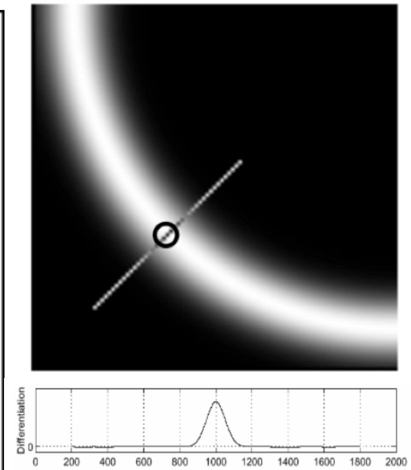


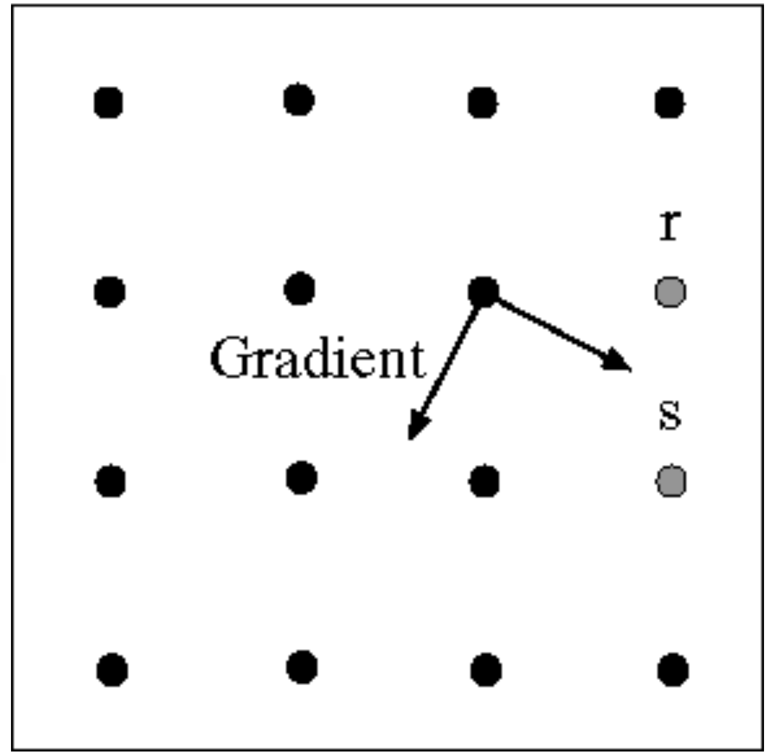
Goal: mark points along the curve where the magnitude is biggest.
How? looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve.
There are then two algorithmic issues:
-at which point is the maximum
-where is the next one?

Forsyth, 2002

Non maximum suppression: check if pixel is local maximum along gradient direction







At q, we have a maximum if the value is larger than those at both *p* and at *r*.
Interpolate between *p* and *r* to get these values.

<u>Predicting the next edge point:</u> Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either *r* or *s*).

Examples: Non-Maximum Suppression

Original image

Gradient magnitude

Non-maxima suppressed

But some edges are broken

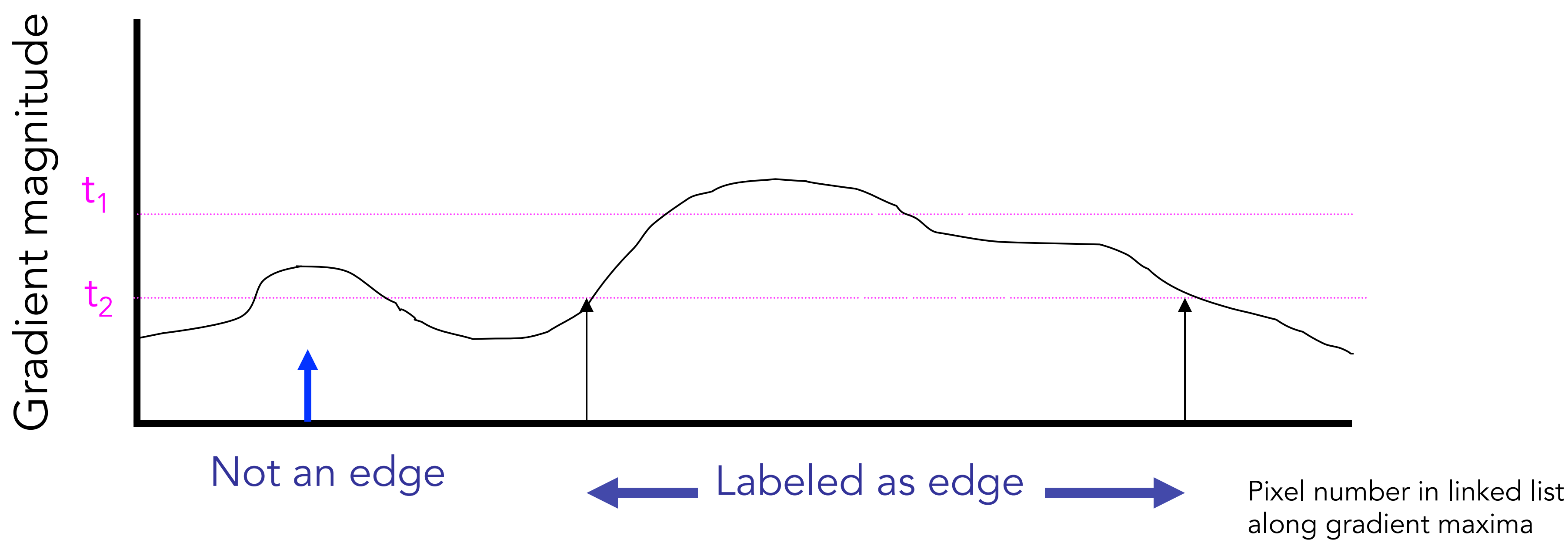
Canny edge detector

edge(image,'canny')

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Closing edge gaps

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Example: Canny Edge Detection

Original image

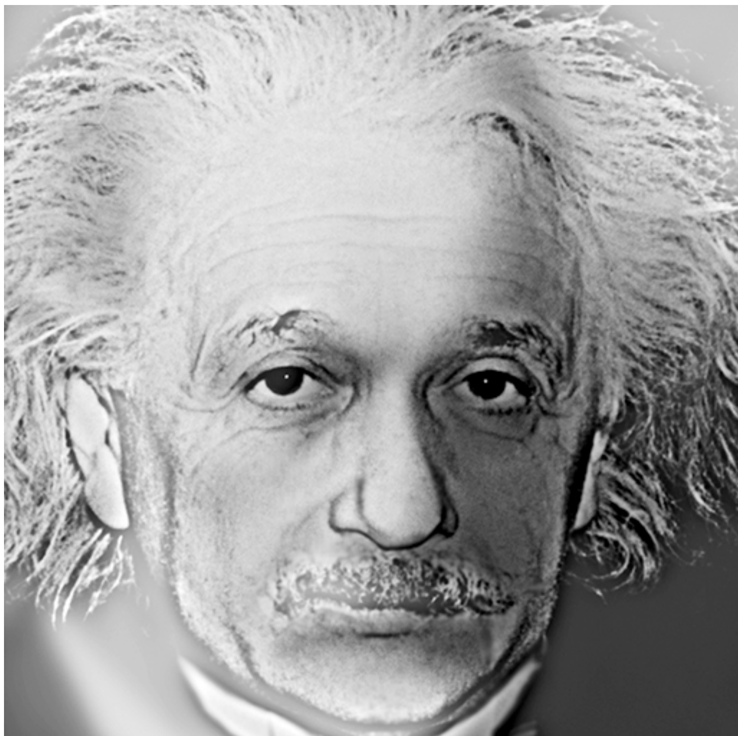
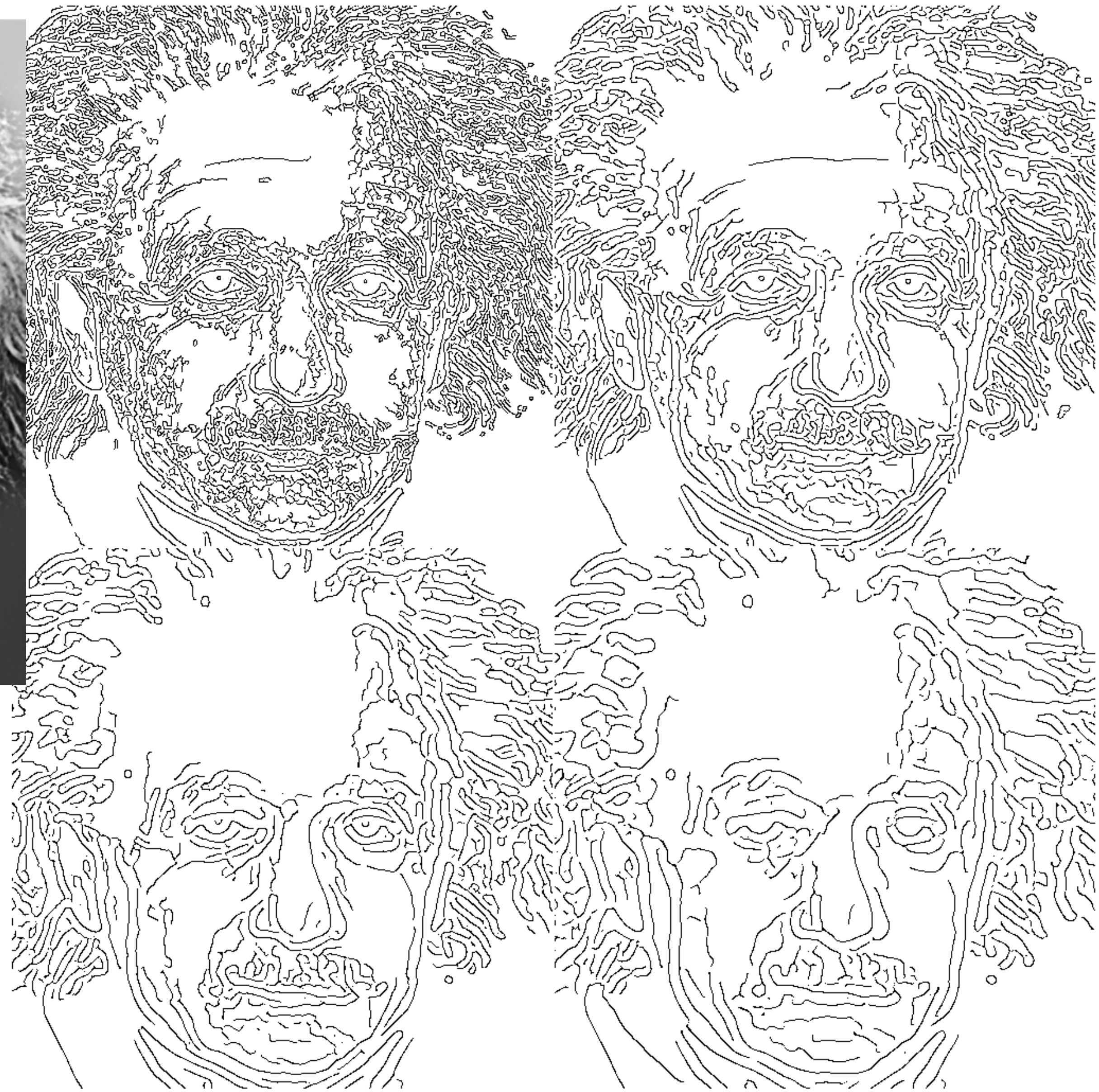
gap is gone

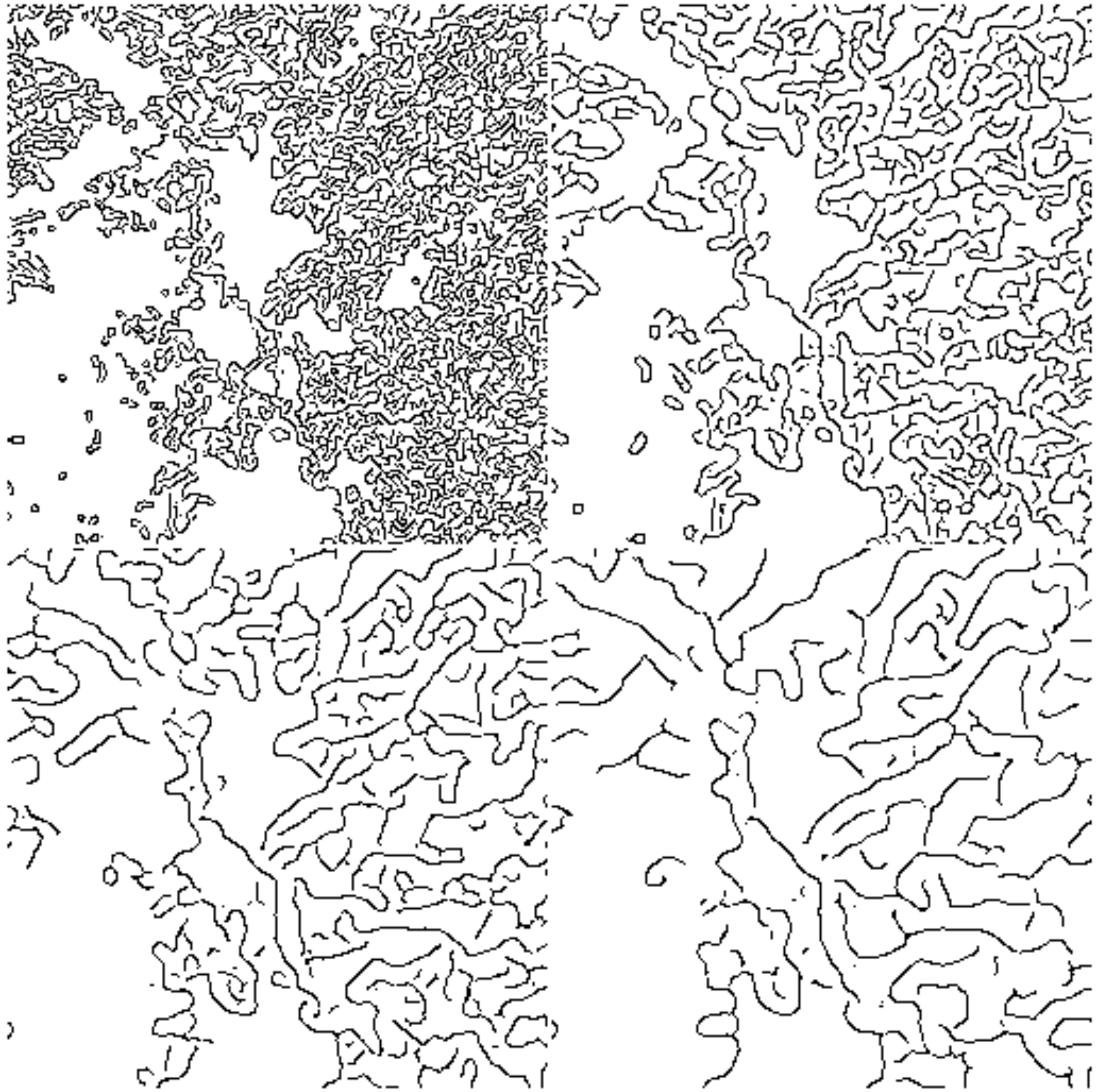
Strong + connected weak edges

Strong edges only

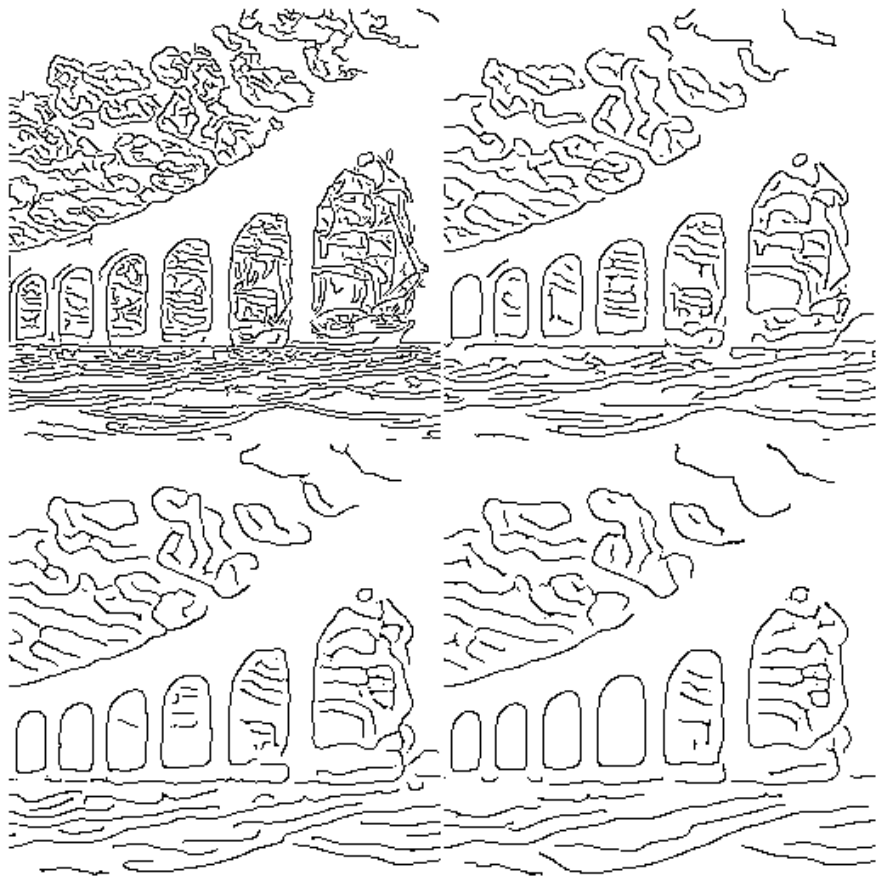
Weak edges

courtesy of G. Loy



II. Segmentation



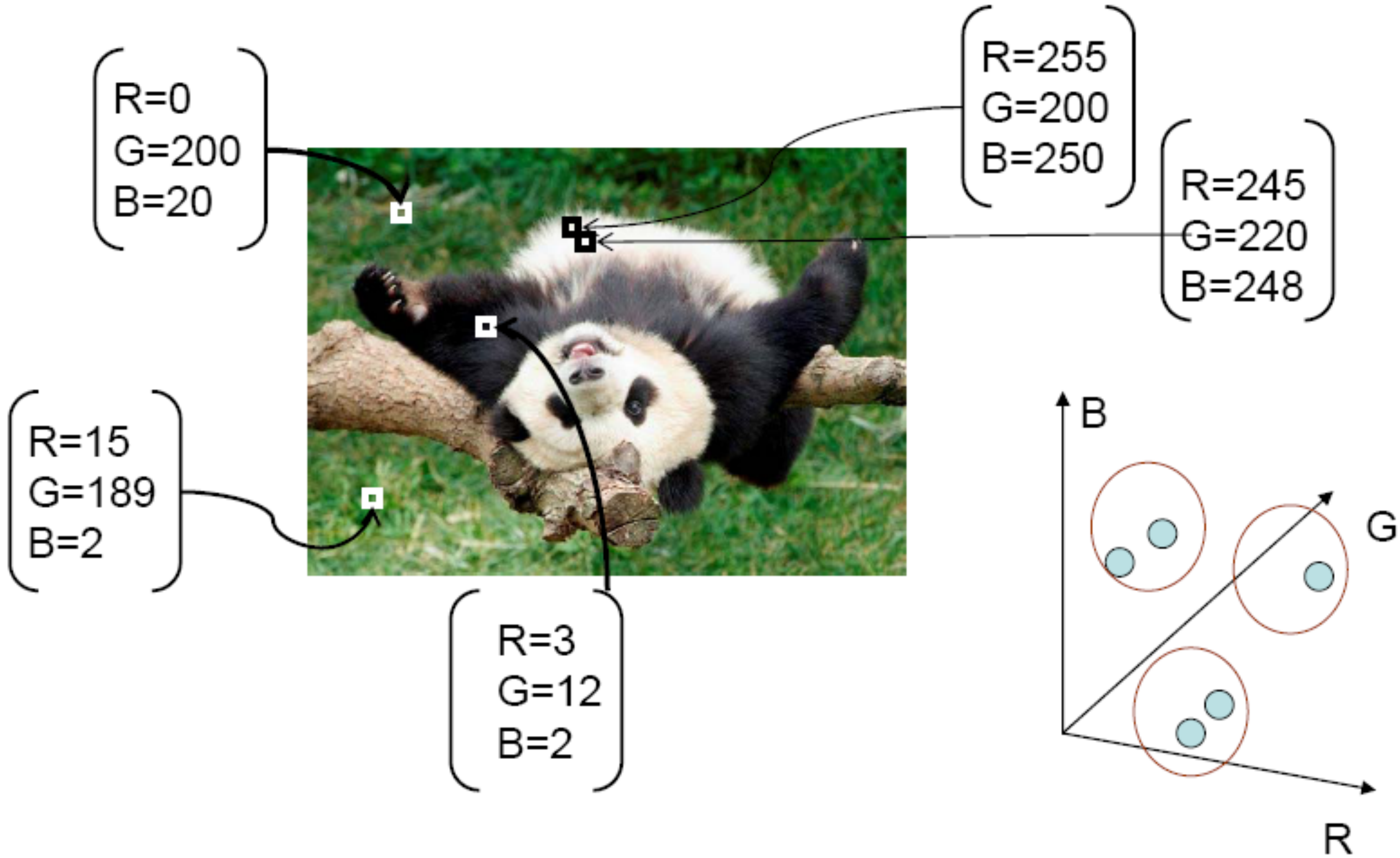
II.1 Bottom-up segmentation

- Group together similar-looking pixels
  - "Bottom-up" process
  - Unsupervised
- Bottom-up segmentation
  - Clustering
  - Mean shift
  - Graph-based

"superpixels"

Method 1: Clustering

- Cluster similar pixels (features) together



Source: K. Grauman

Segmentation as clustering

- Cluster together (pixels, tokens, etc.) that belong together…
- Agglomerative clustering
  - attach closest to cluster it is closest to
  - repeat
- Divisive clustering
  - split cluster along best boundary
  - repeat
- Dendrograms
  - yield a picture of output as clustering process continues

Chapter – Forsyth & Ponce

A simple segmentation algorithm

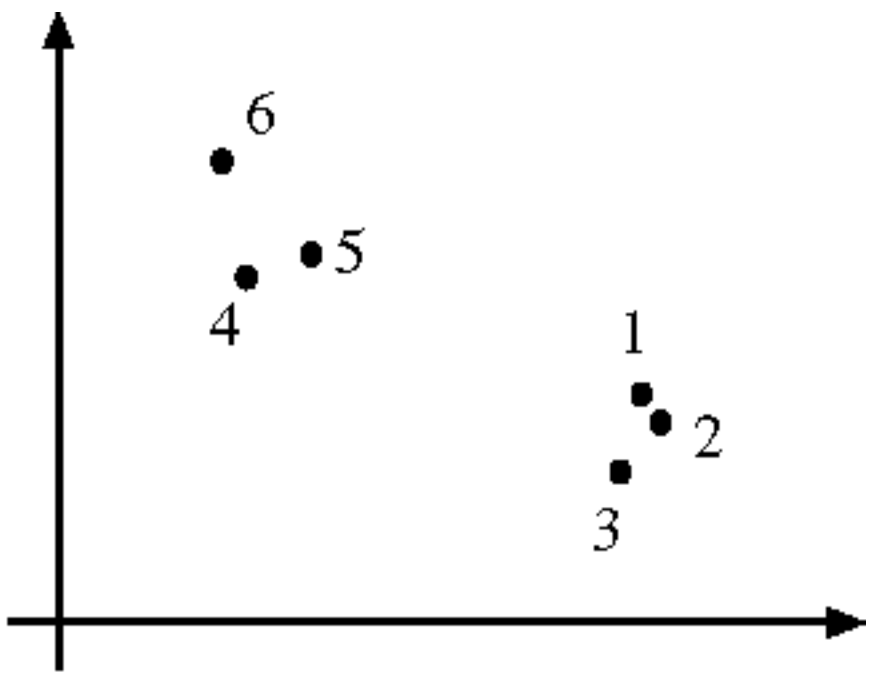
- Each pixel is described by a vector

  $z = [r, g, b]$ or $[Y\ u\ v]$, …

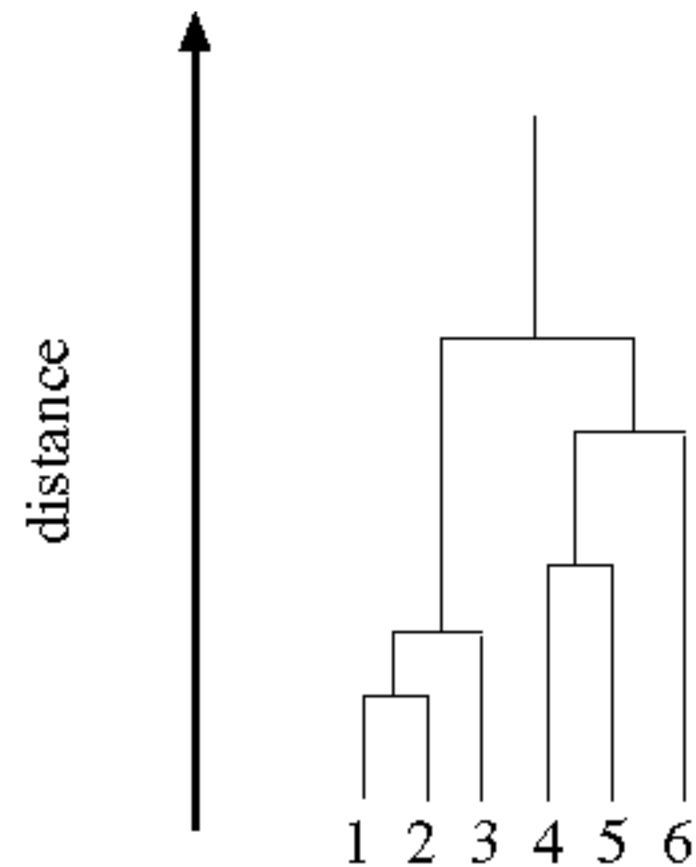
- Run a clustering algorithm (e.g. Kmeans) using some distance between pixels:

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

Data set



Dendrogram obtained by agglomerative clustering

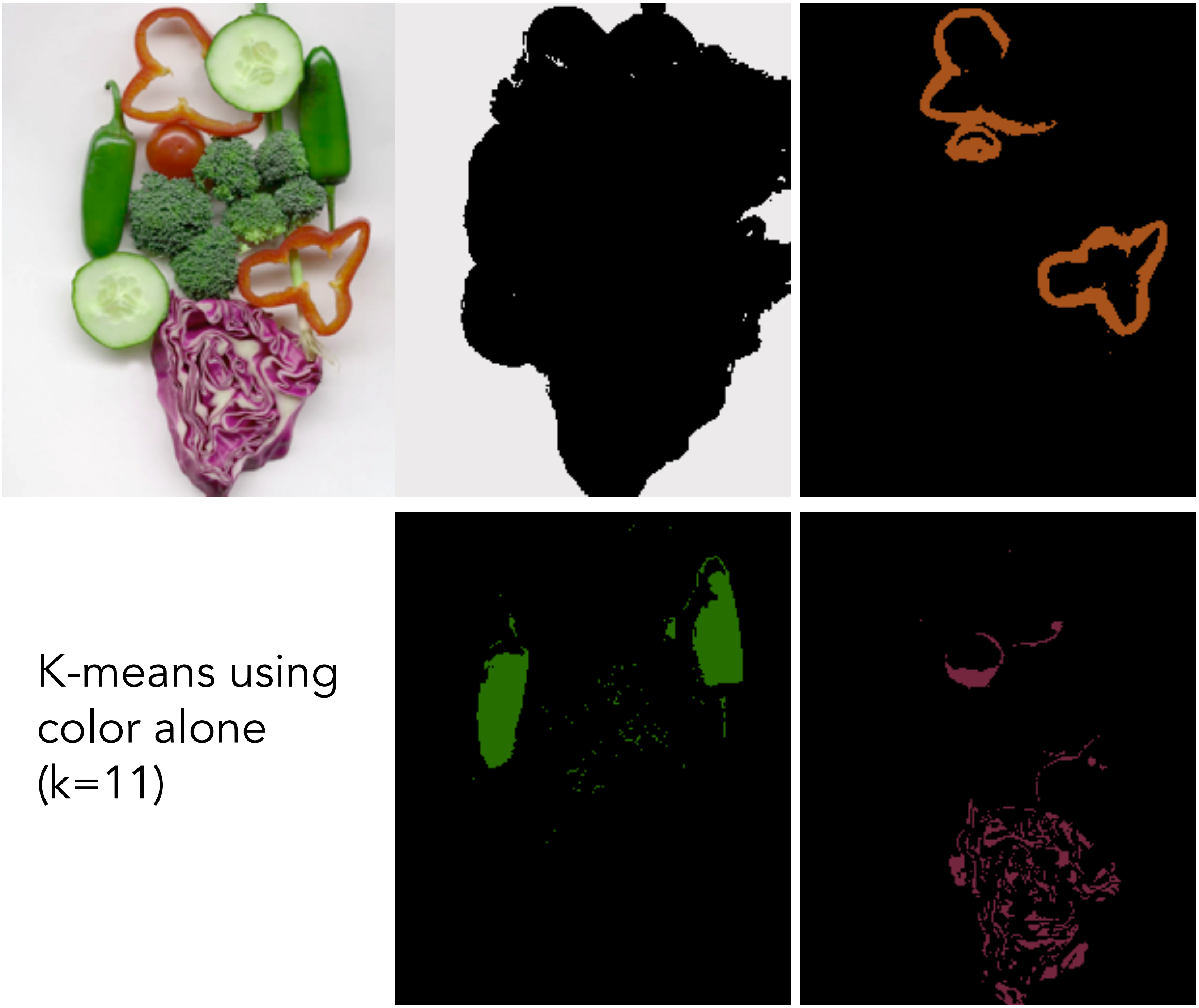


- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent

Image

Intensity-based clusters

Color-based clusters



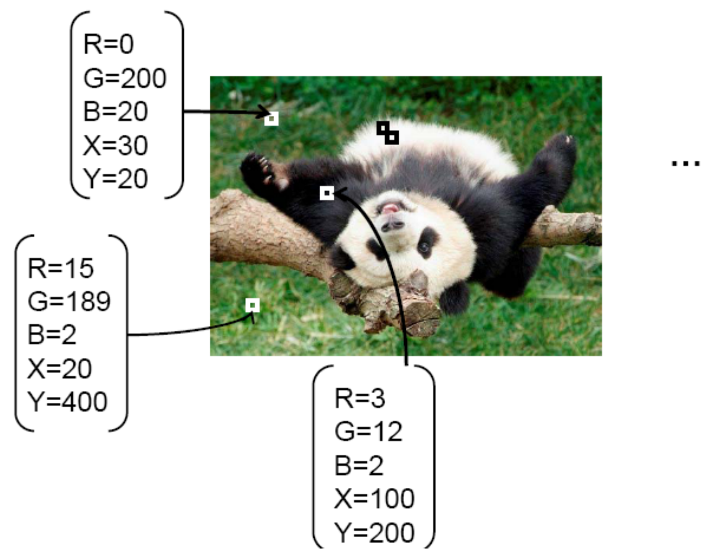
K-means using color alone (k=11)

Including spatial relationships

Augment data to be clustered with spatial coordinates.

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

- Cluster similar pixels (features) together



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

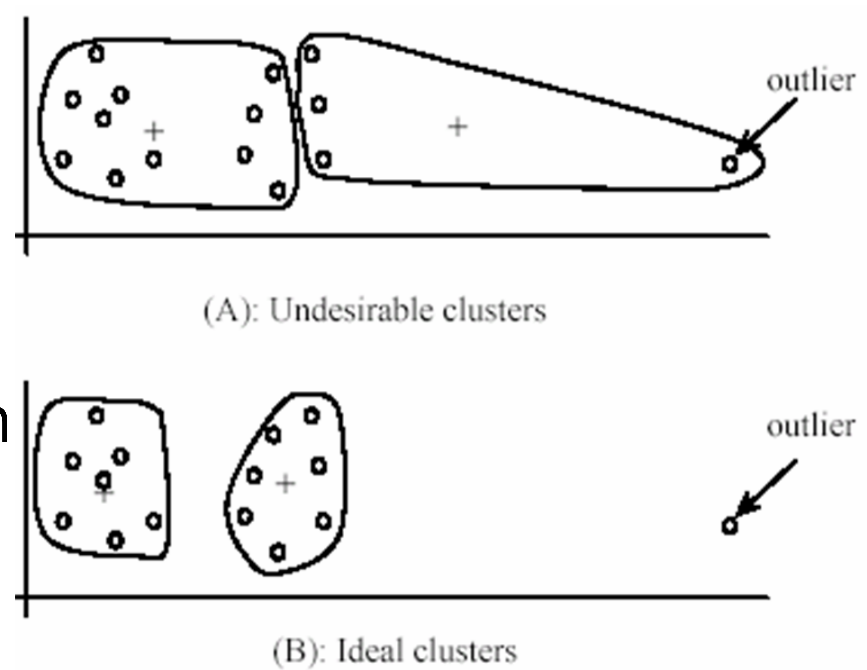
K-means using colour and position, 20 segments

Still misses goal of perceptually pleasing or useful segmentation

Hard to pick K…

K-Means for segmentation

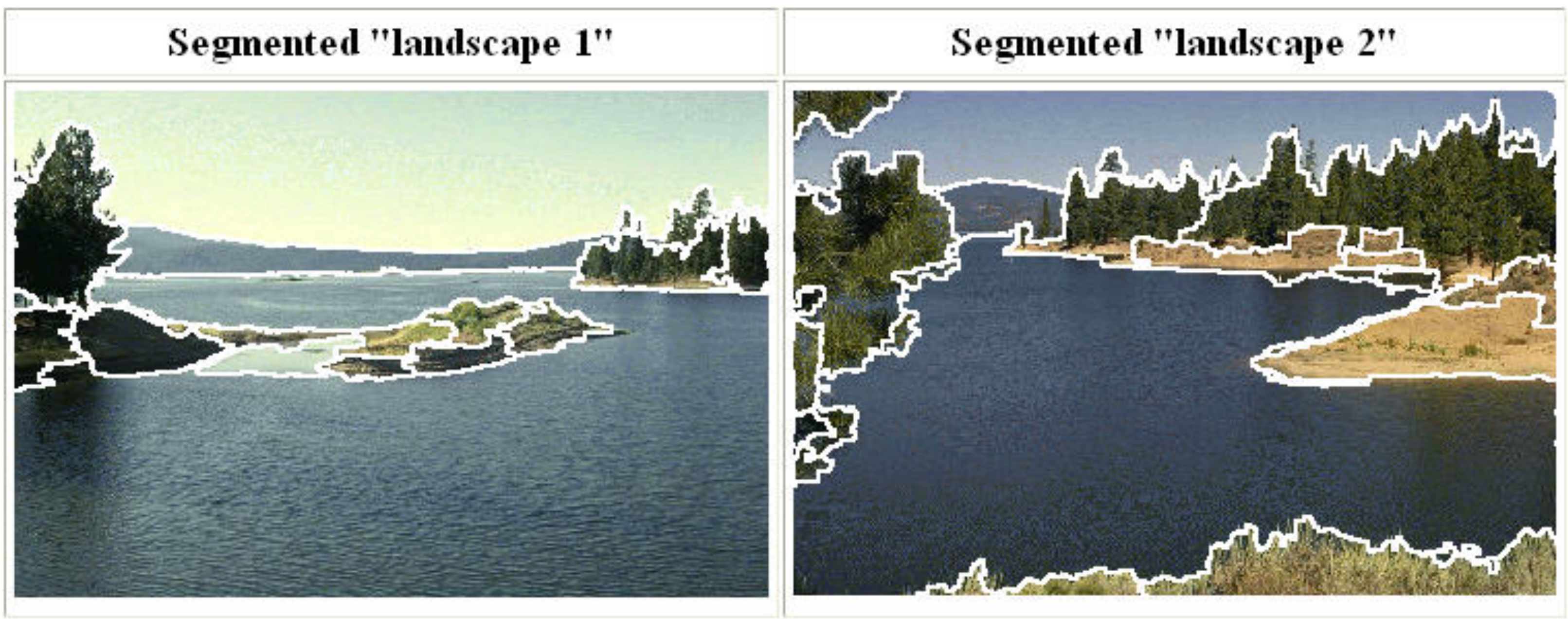
- Pros
  - Very simple method
  - Converges to a local minimum of the error function
- Cons
  - Memory-intensive
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds "spherical" clusters



Slide credit: S. Lazebnik

Method 2: Mean shift clustering

- An advanced and versatile technique for clustering-based segmentation



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

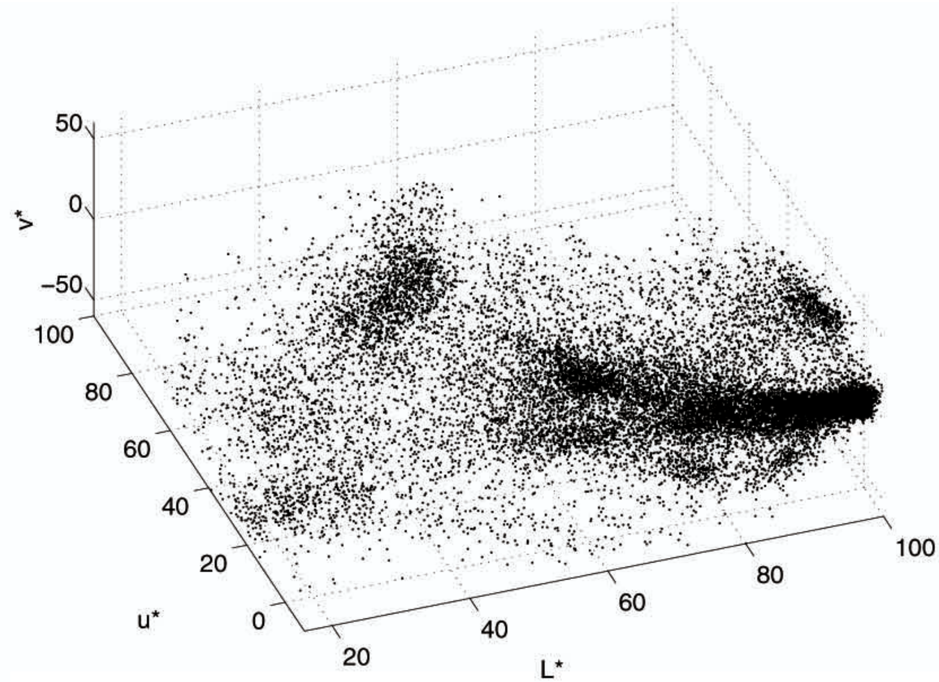
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean shift algorithm

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

image

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

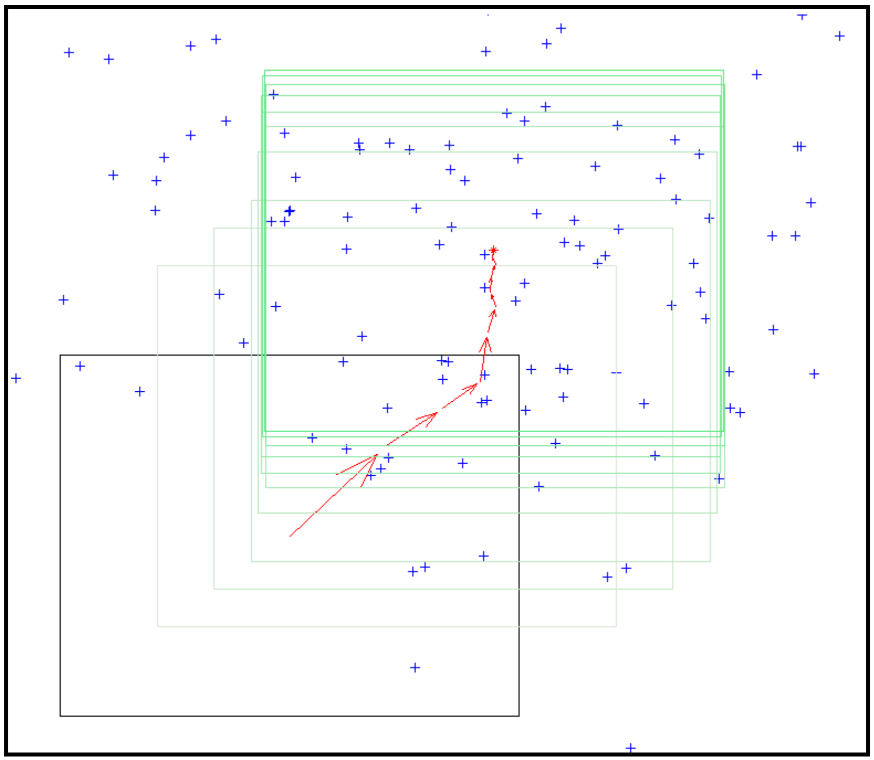


Mean Shift Algorithm

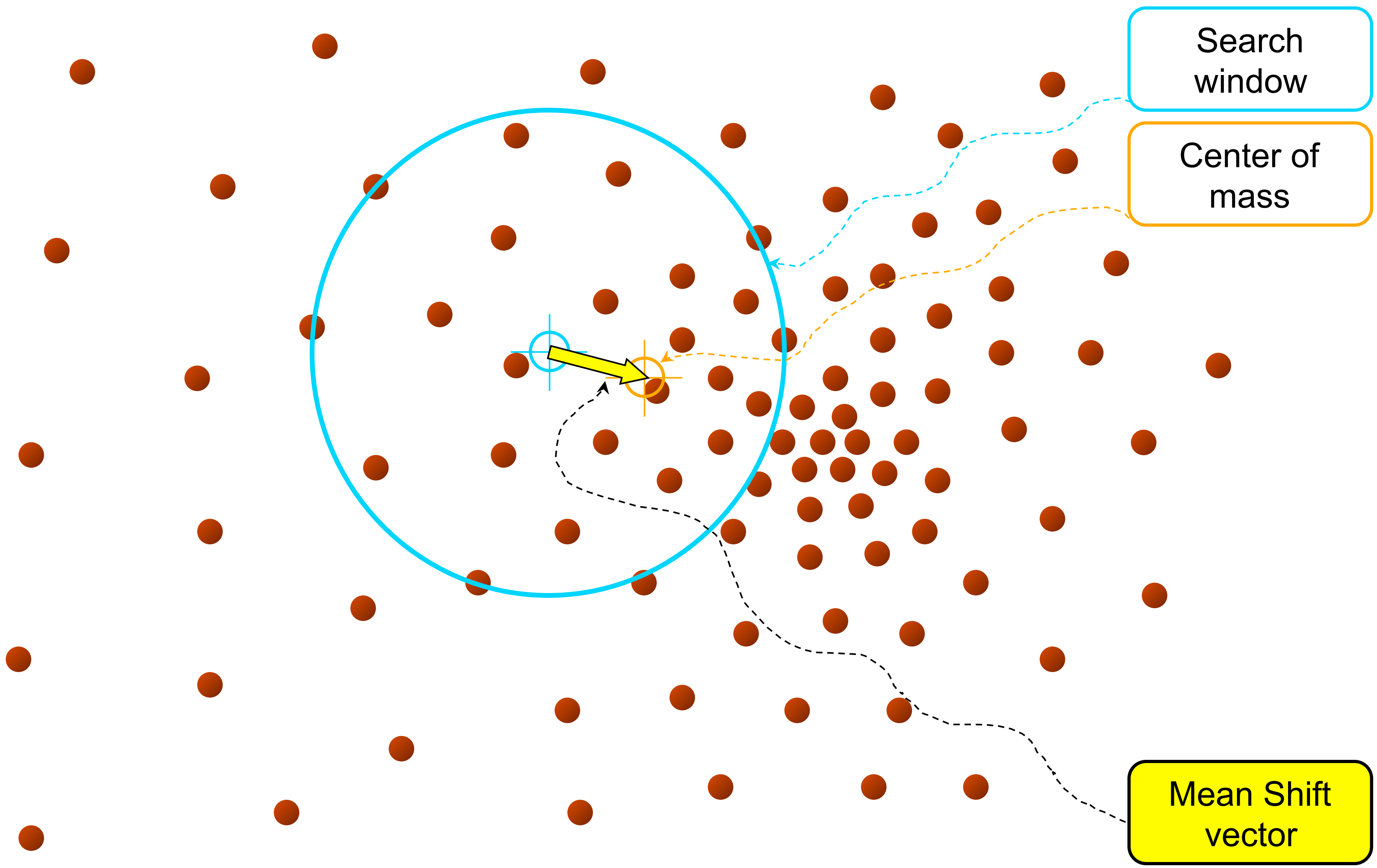
Mean Shift Algorithm

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

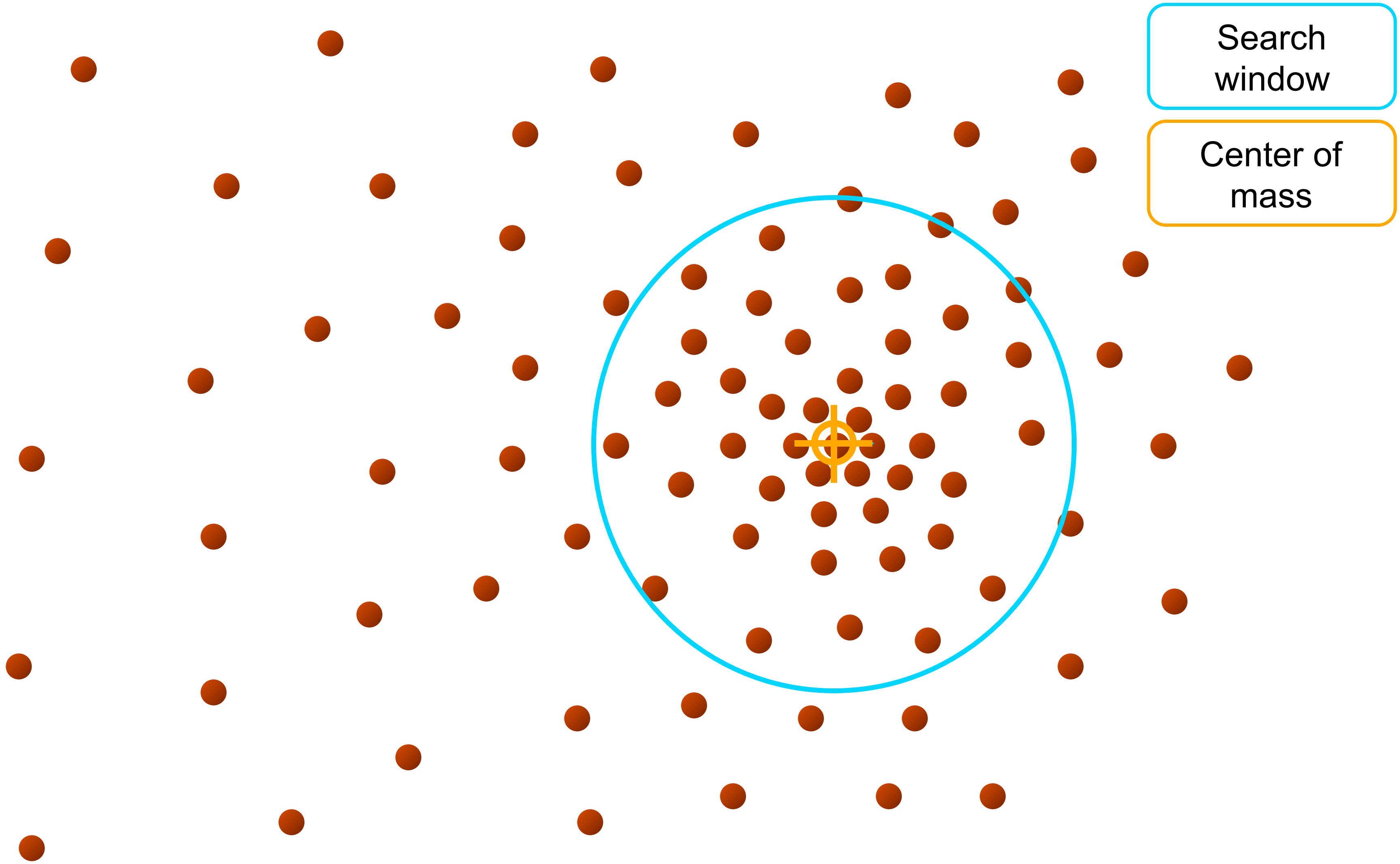
The mean shift algorithm seeks the "mode" or point of highest density of a data distribution:



Two issues:
(1) Kernel to interpolate density based on sample positions.
(2) Gradient ascent to mode.



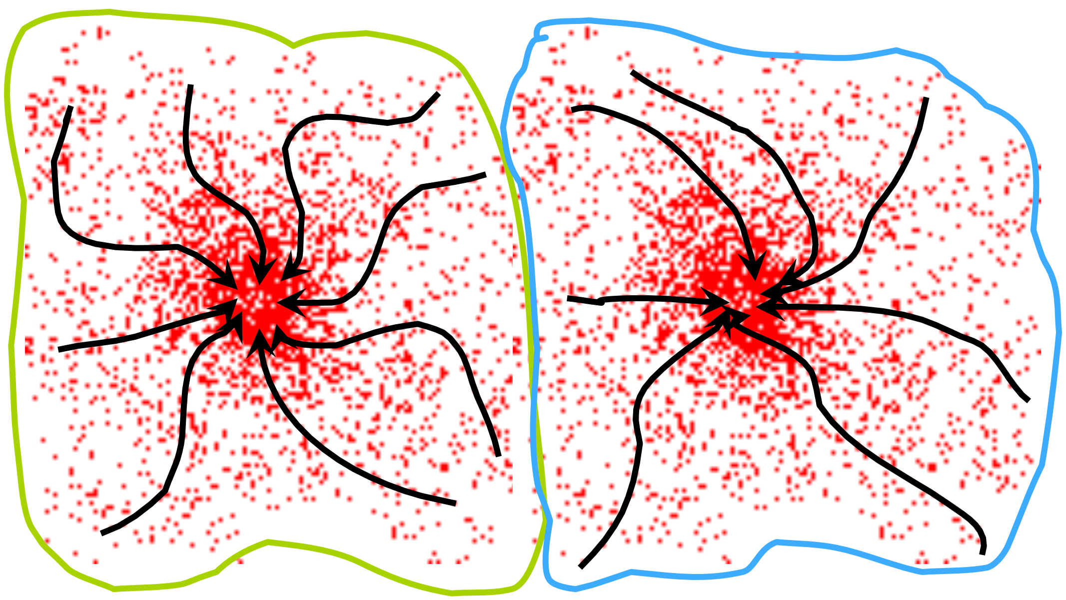
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

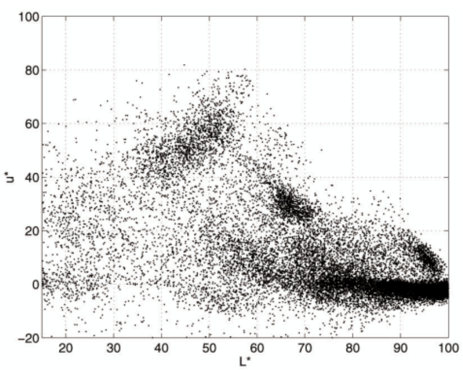
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



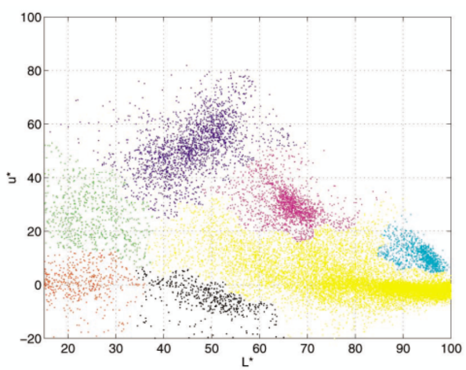
Slide by Y. Ukrainitz & B. Sarel

Mean Shift Segmentation

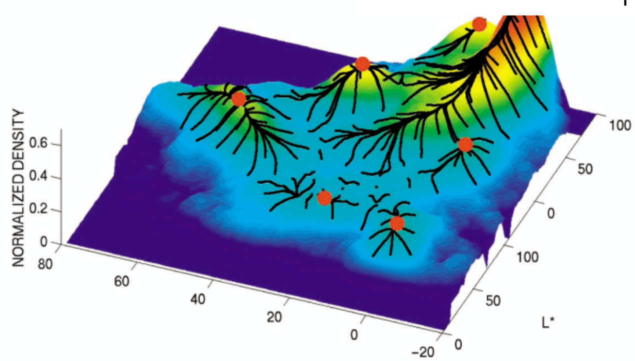
1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same "peak" or mode.
5. The data these merged windows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



Pixels in L*u* space



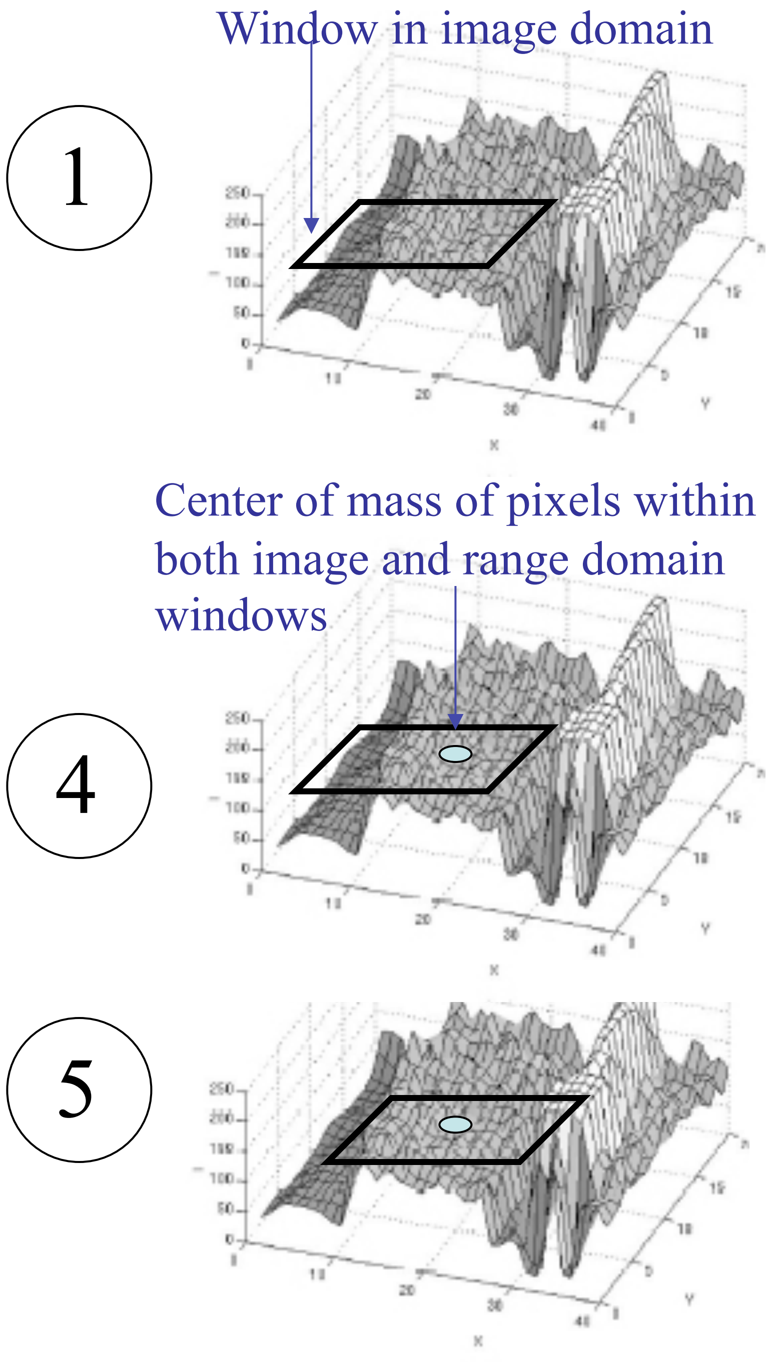
Clustering results after
~160 mean shift procedures

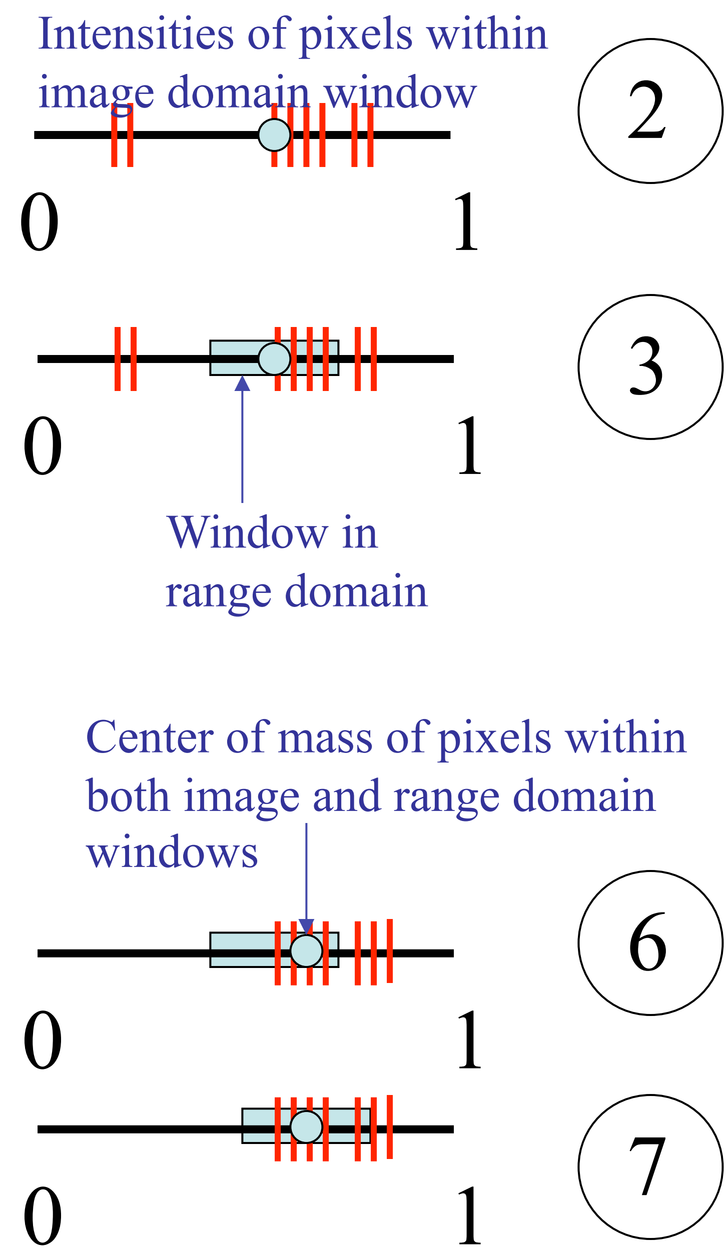


Corresponding trajectories with peaks marked as red dots

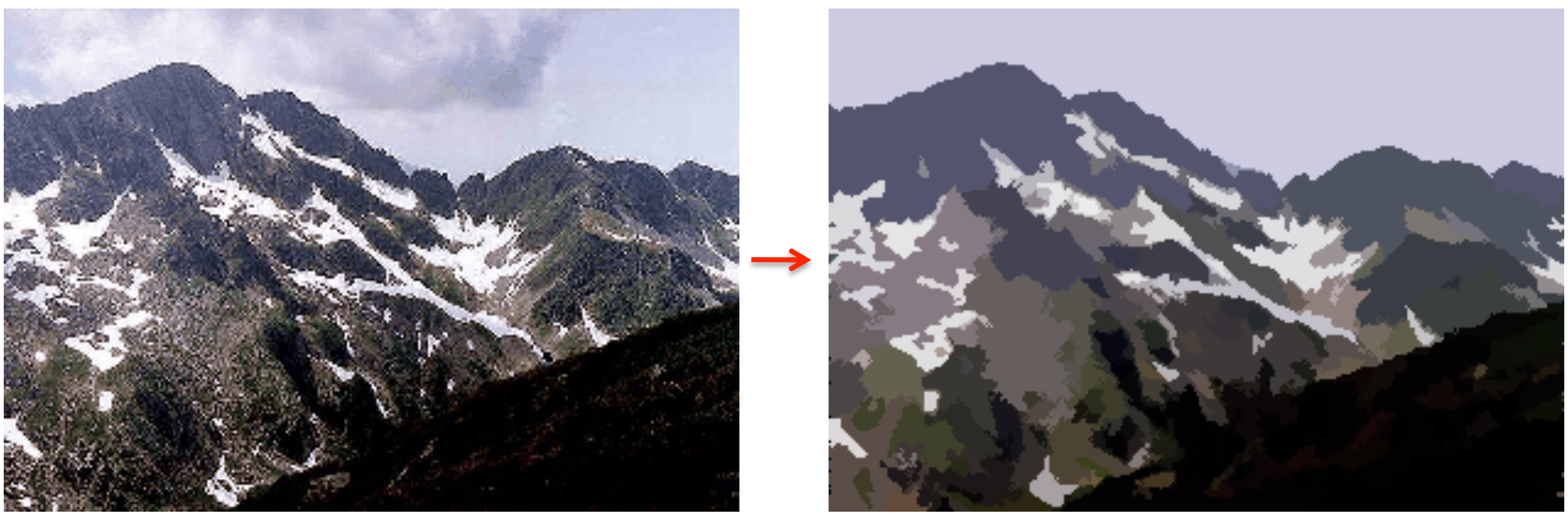
Szeliski – Chapter 5 – Segmentation, 5.3 Mean shift

Apply mean shift jointly in the image (left col.) and range (right col.) domains





Mean Shift color & spatial Segmentation Results:



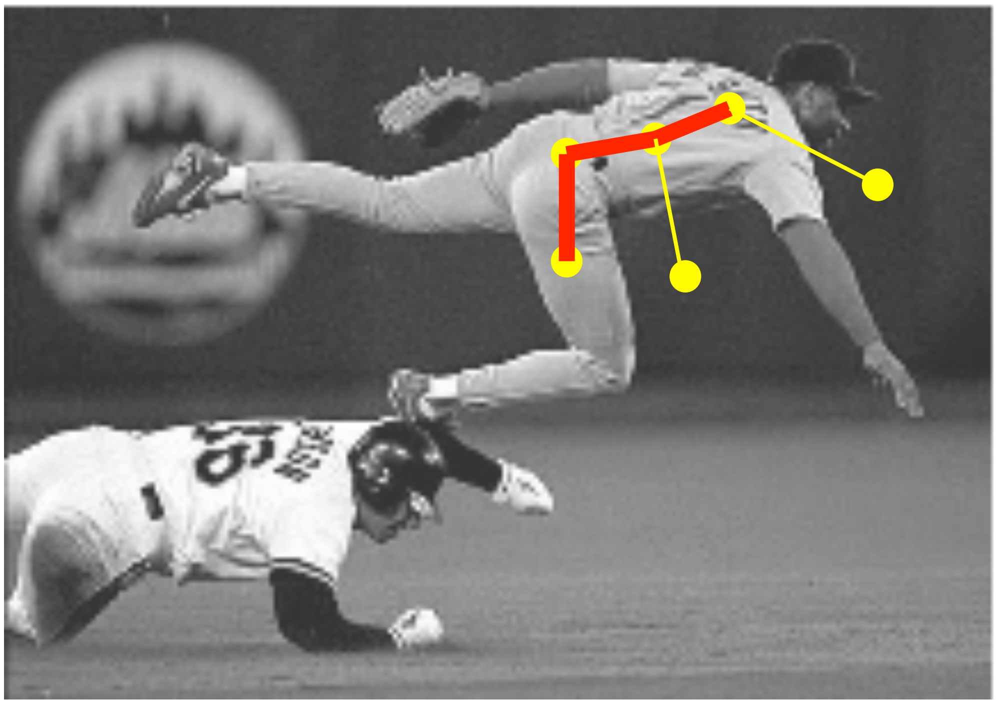
Mean shift pros and cons

- Pros
  - Clusters are places where data points tend to be close together
  - 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

Slide credit: S. Lazebnik

Method 3: Graph-Theoretic Image Segmentation

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



A different way of thinking about segmentation…

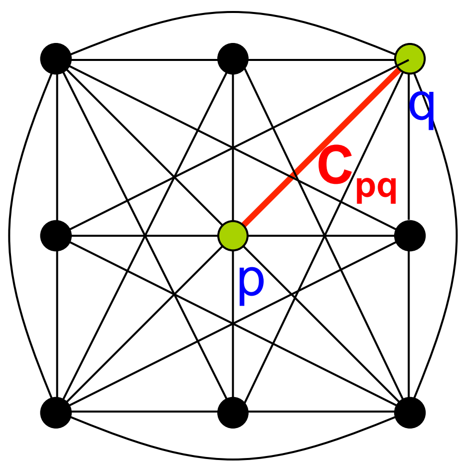
V: image pixels

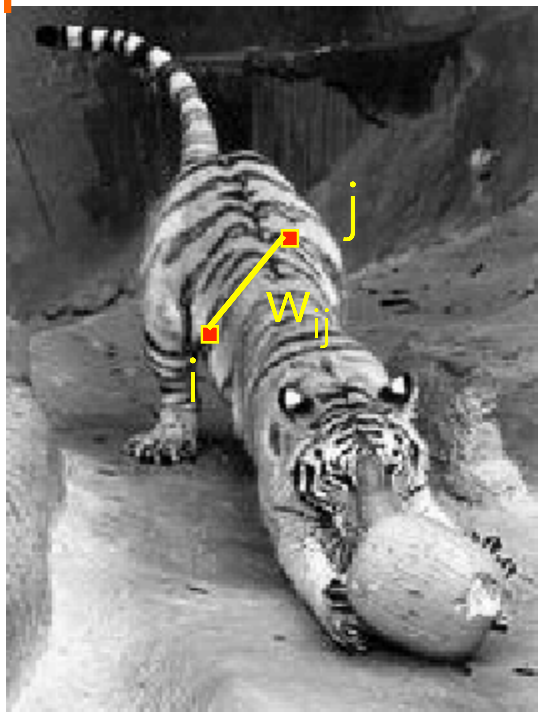
E: connections between pairs of nearby pixels

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

Segmentation = graph partition

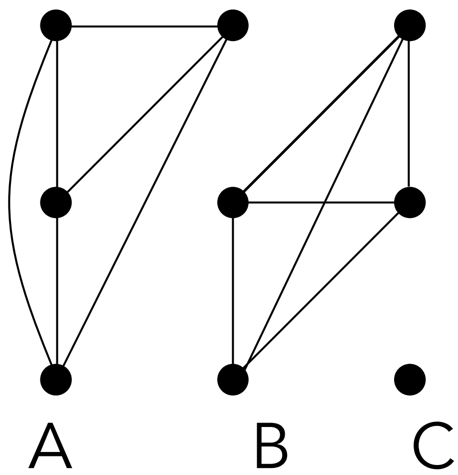
Segmentation by graph cut





- 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

Segmentation by graph cut



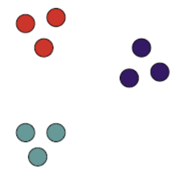
- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low cost (similarity or affinity)
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

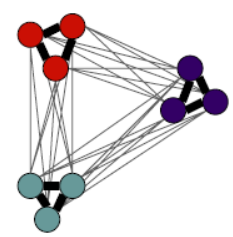
Source: S. Seitz

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.

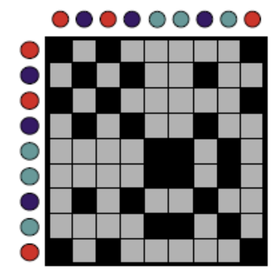
1- Get vectors of **data**





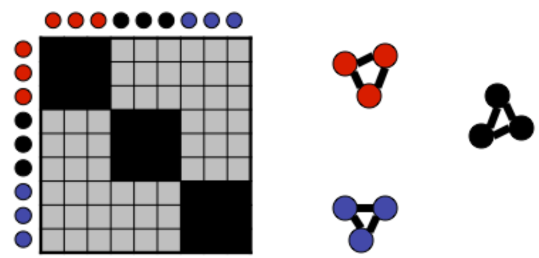
2a- Build a **similarity** graph

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

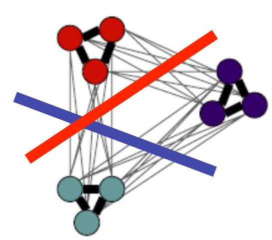


Similarities

3- Calculate **eigenvectors**



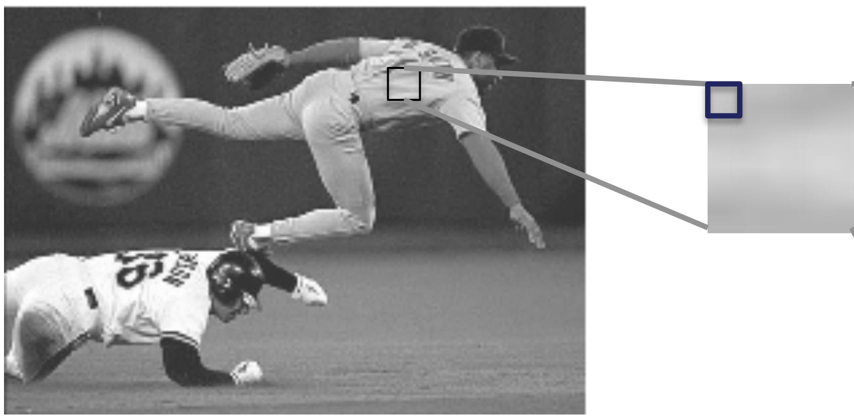
4- Cut the graph:
apply **threshold** to eigenvectors



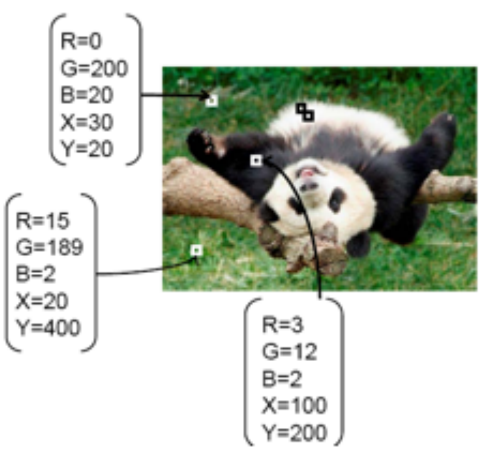
1- Vectors of data

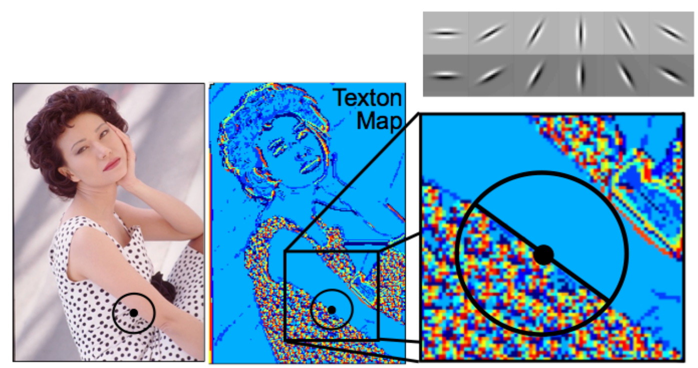
We represent each pixel by a feature vector **x**, and define a distance function appropriate for this feature representation (e.g. euclidean distance).

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

Measuring 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 with the help of a generalized Gaussian kernel:

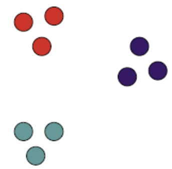
$$\exp\left(-\frac{1}{2\sigma^2}\operatorname{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$

Slide credit: S. Lazebnik

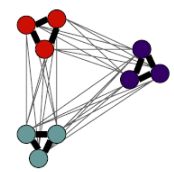
Graph-based Image Segmentation

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.

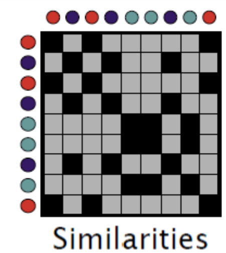
1- Get vectors of **data**



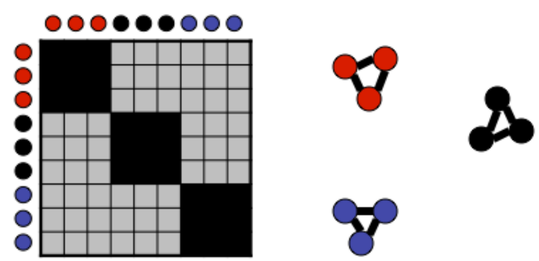
2a- Build a **similarity** graph



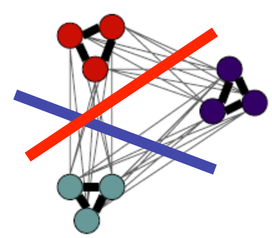
2b- Build a similarity/affinity **matrix**



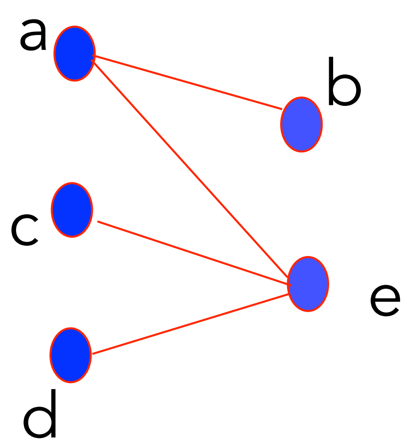
3- Calculate **eigenvectors**



4- Cut the graph:
apply **threshold** to eigenvectors



2a- What is a graph?



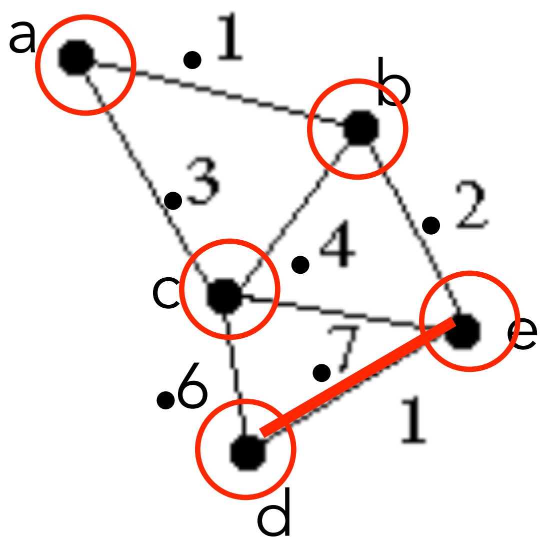
$$
\begin{array}{c c}
 & \begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0
\end{bmatrix}
\end{array}
$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

2a- What is a weighted graph?

Affinity Matrix represents the weighted links



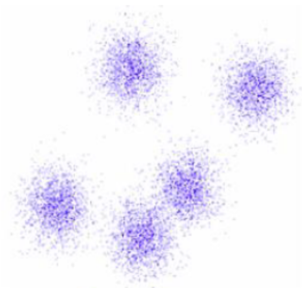
$$W = \begin{array}{c} \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{bmatrix} 1 & .1 & .3 & 0 & 0 \\ 0 & 1 & .4 & 0 & .2 \\ .3 & .4 & 1 & .6 & .7 \\ 0 & 0 & .6 & 1 & 1 \\ 0 & .2 & .7 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}$$

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

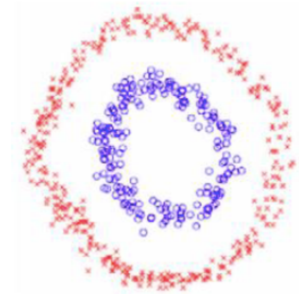
i,j are the pixels in the image

See Forsyth-Ponce chapter



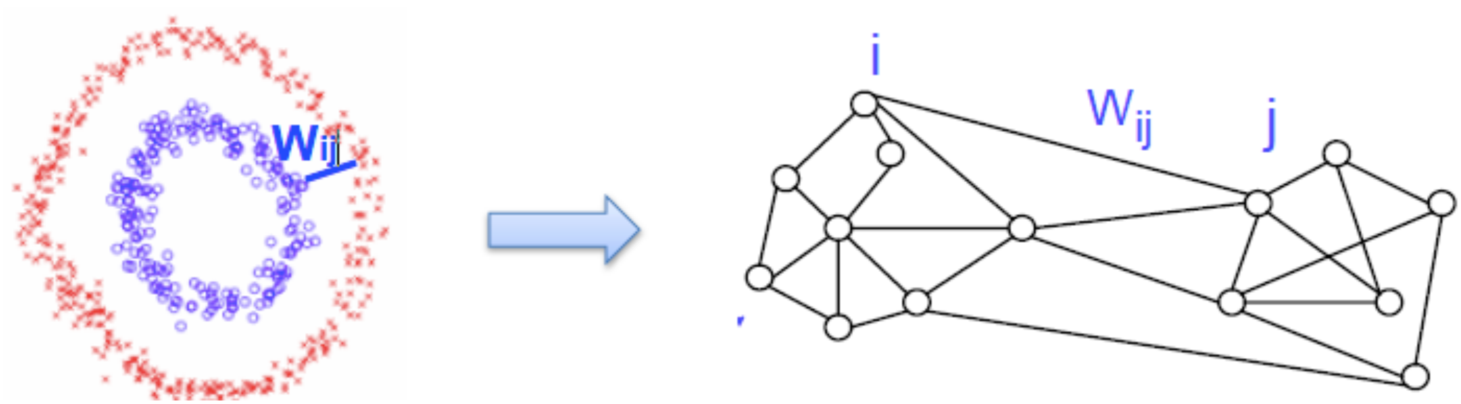
Compactness

Similarity graph construction



Connectivity

Similarity Graphs: Model local neighborhood relations between data points



E.g. Gaussian kernel similarity function

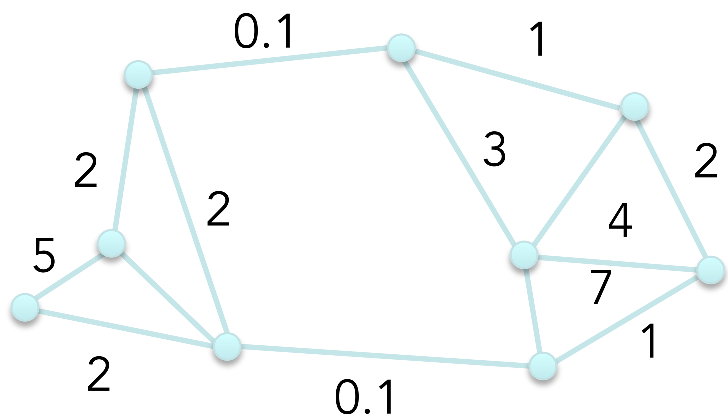
$$W_{ij} = e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

Controls size of neighborhood

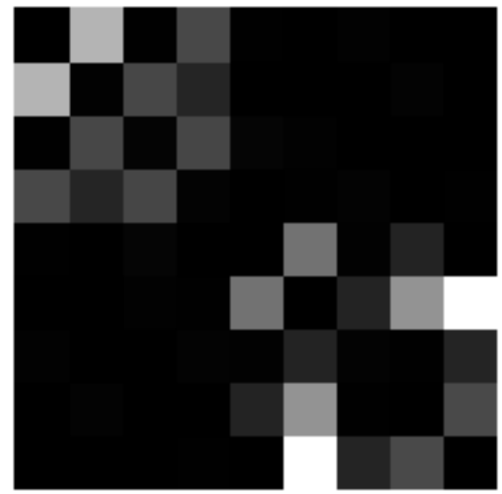
See Forsyth-Ponce chapter

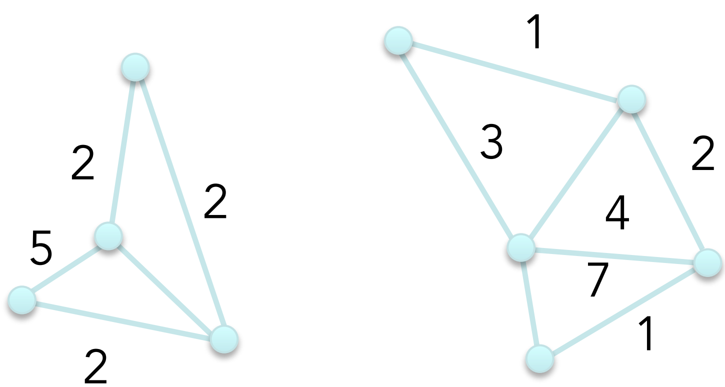
2b- Building Affinity Matrix

A weighted graph



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



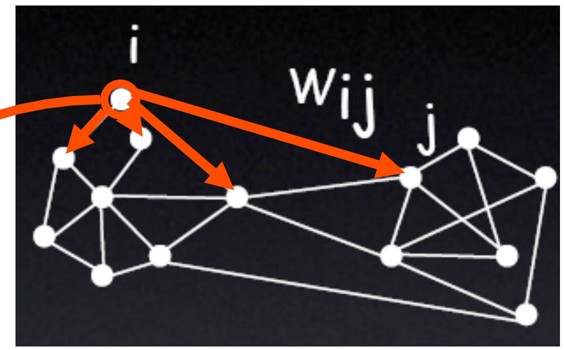


A cut of the graph: two tightly linked components. This cut decomposes the graph’s matrix into two main blocks on the diagonal

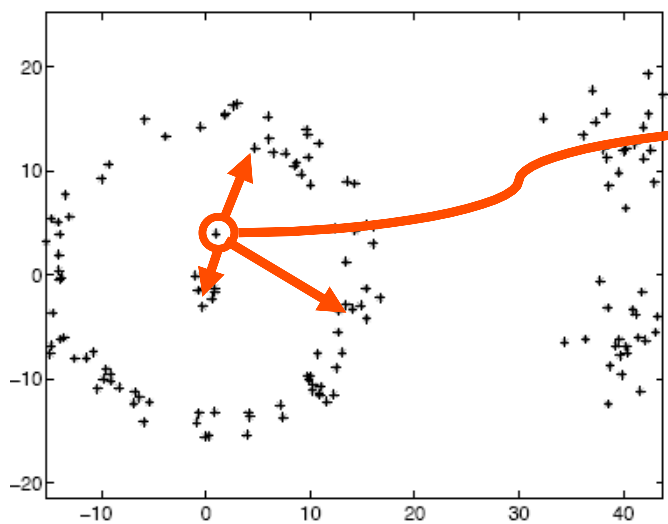
We can do segmentation by finding the *minimum cut* in a graph.

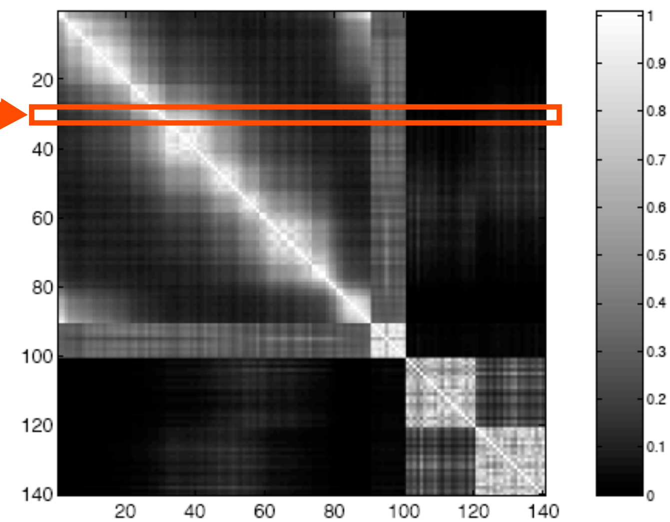
Graph terminology

- Similarity matrix: $W = \left[w_{i,j} \right]$



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

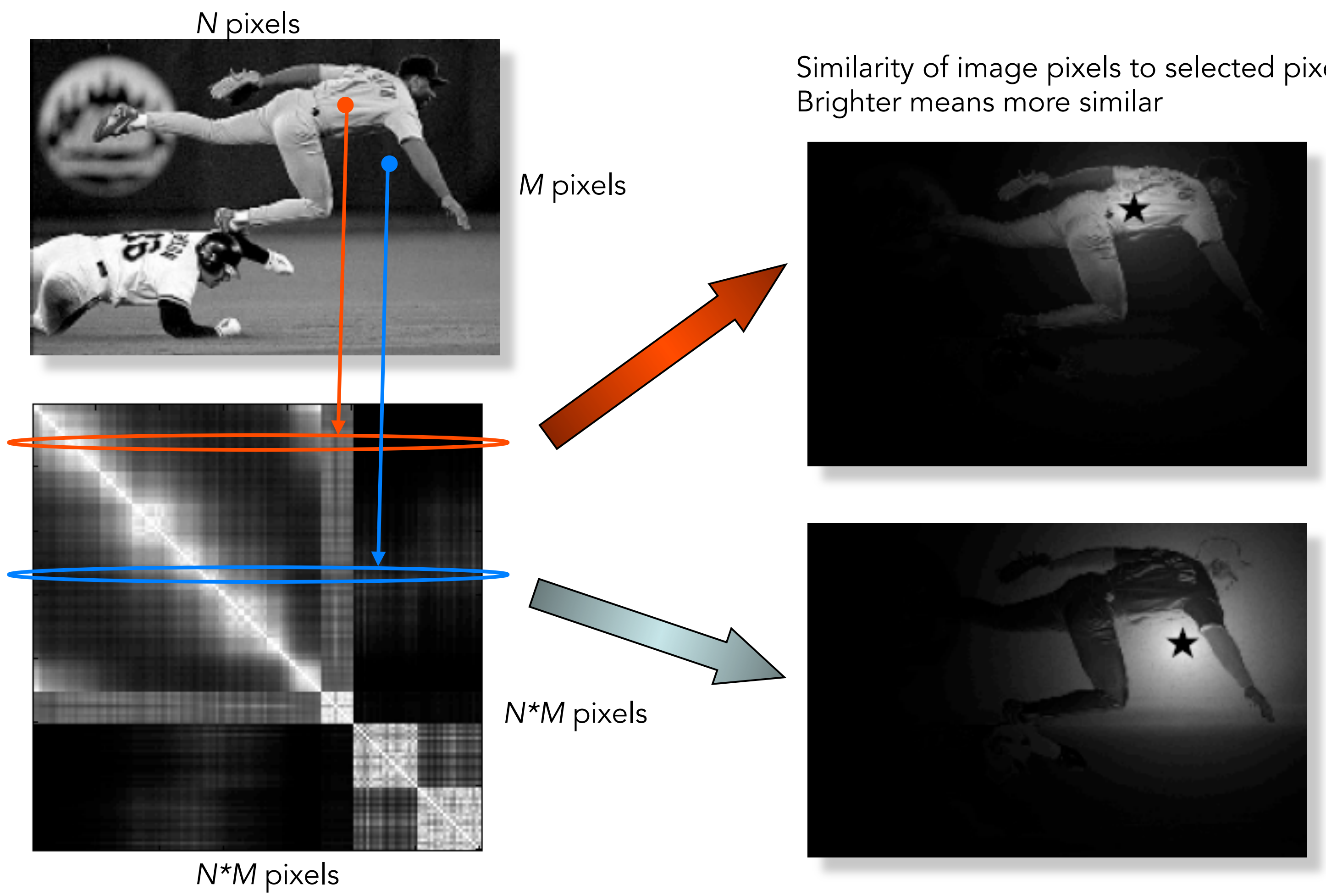




Weight matrix associated with the graph
(larger values are lighter)

Slides from Jianbo Shi

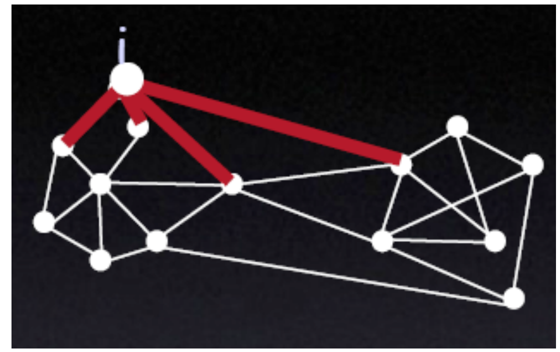
Affinity matrix of a natural image

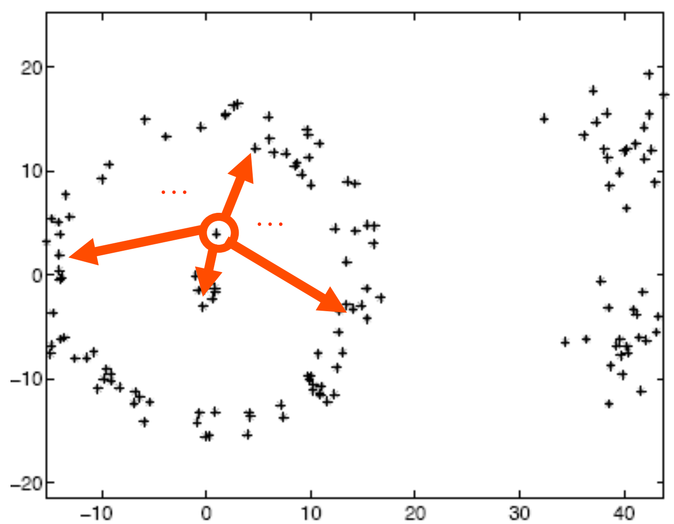


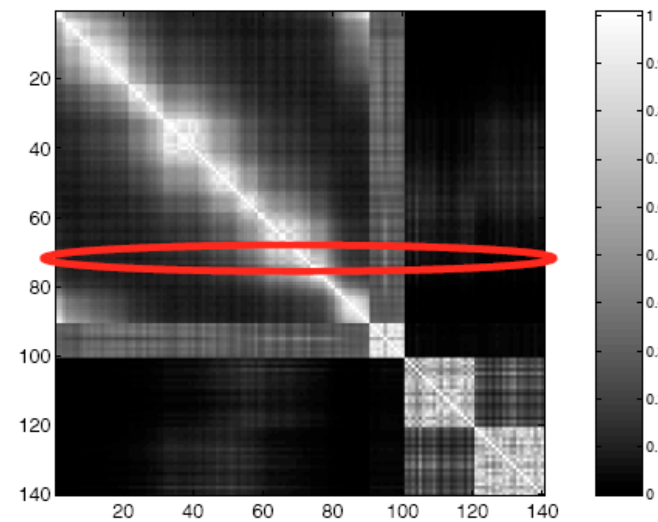
Graph terminology

- Degree of node:

$$d_i = \sum_j w_{i,j}$$





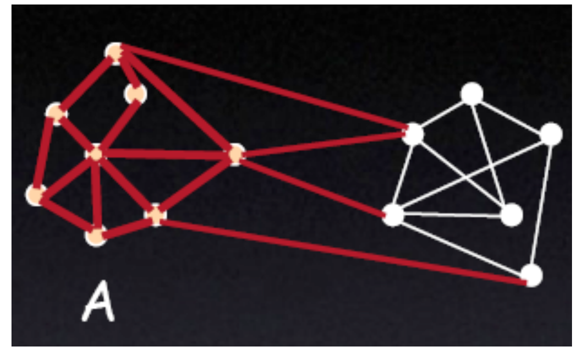


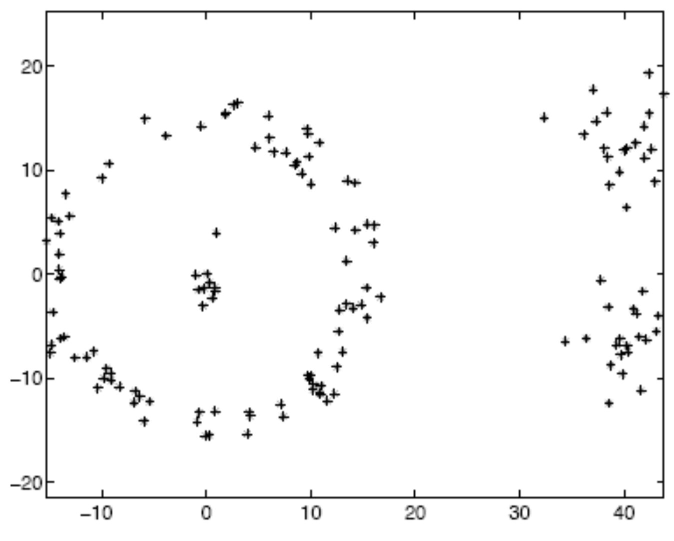
Slides from Jianbo Shi

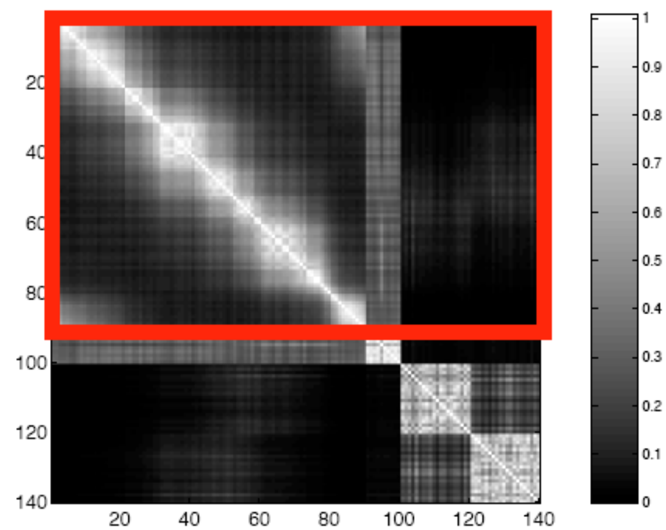
Graph terminology

- Volume of set:

$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





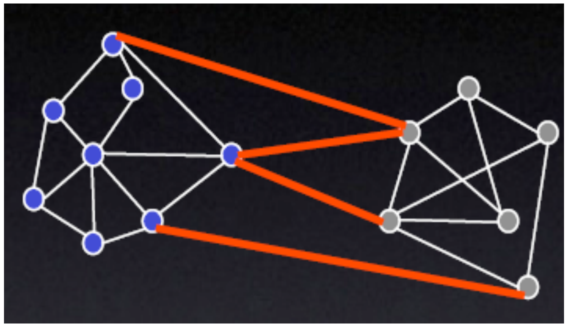


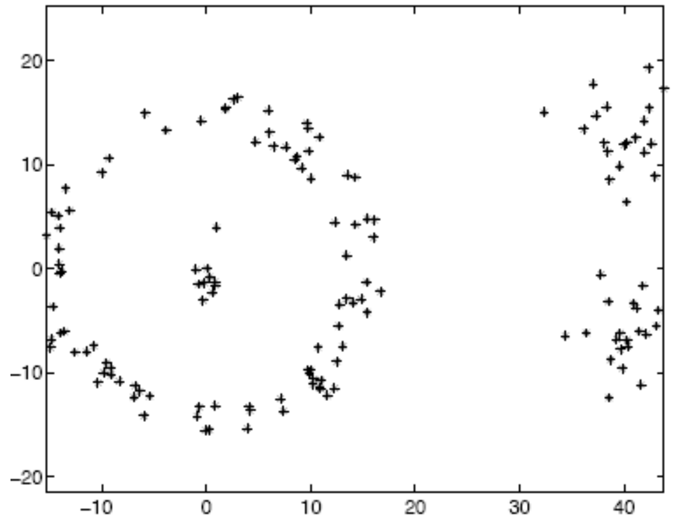
Slides from Jianbo Shi

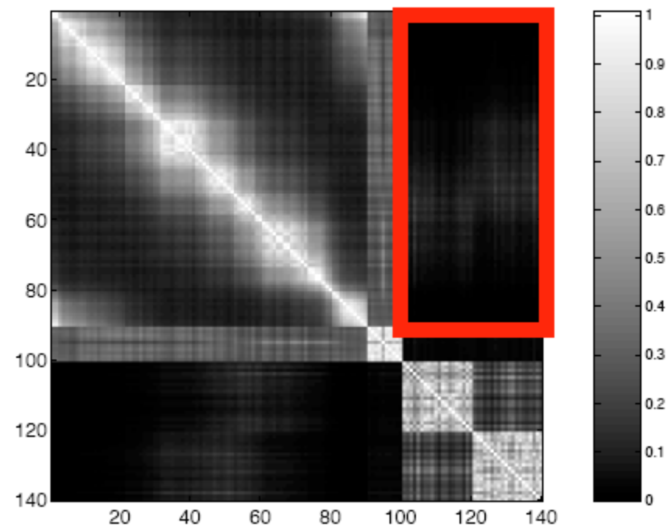
Graph terminology

Cuts in a graph:

$$cut(A,\overline{A}) = \sum_{i\in A,\, j\in \overline{A}} w_{i,j}$$







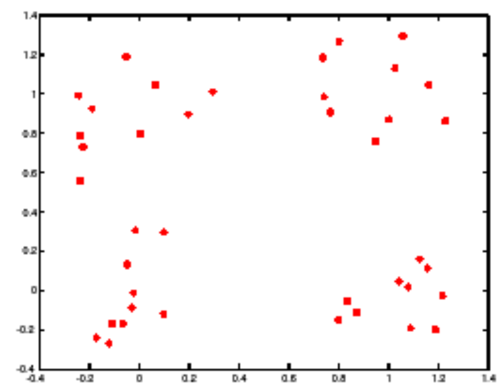
Slides from Jianbo Shi

Scale affects affinity

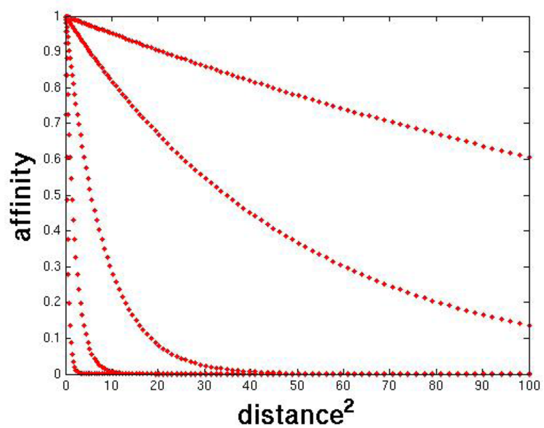
$$W_{ij} = \exp(-\| z_i - z_j \|^2 / \sigma^2)$$

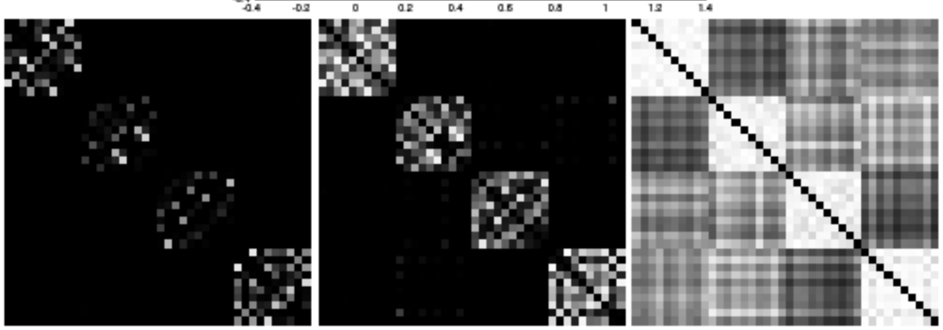
- Small σ: group only nearby points
- Large σ: group far-away points

Dataset of 4 groups of 10 points drawn from a normal distribution with four different means



σ=.2





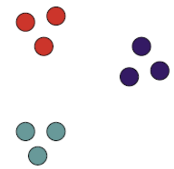
σ=.1 σ=.2 σ=1

See Forsyth-Ponce chapter

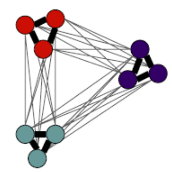
Graph-based Image Segmentation

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.

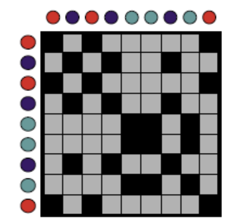
1- Get vectors of **data**



2a- Build a **similarity** graph

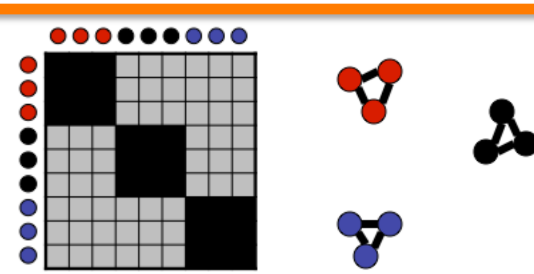


2b- Build a similarity **matrix**

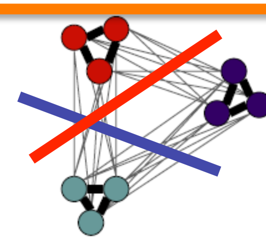


Similarities

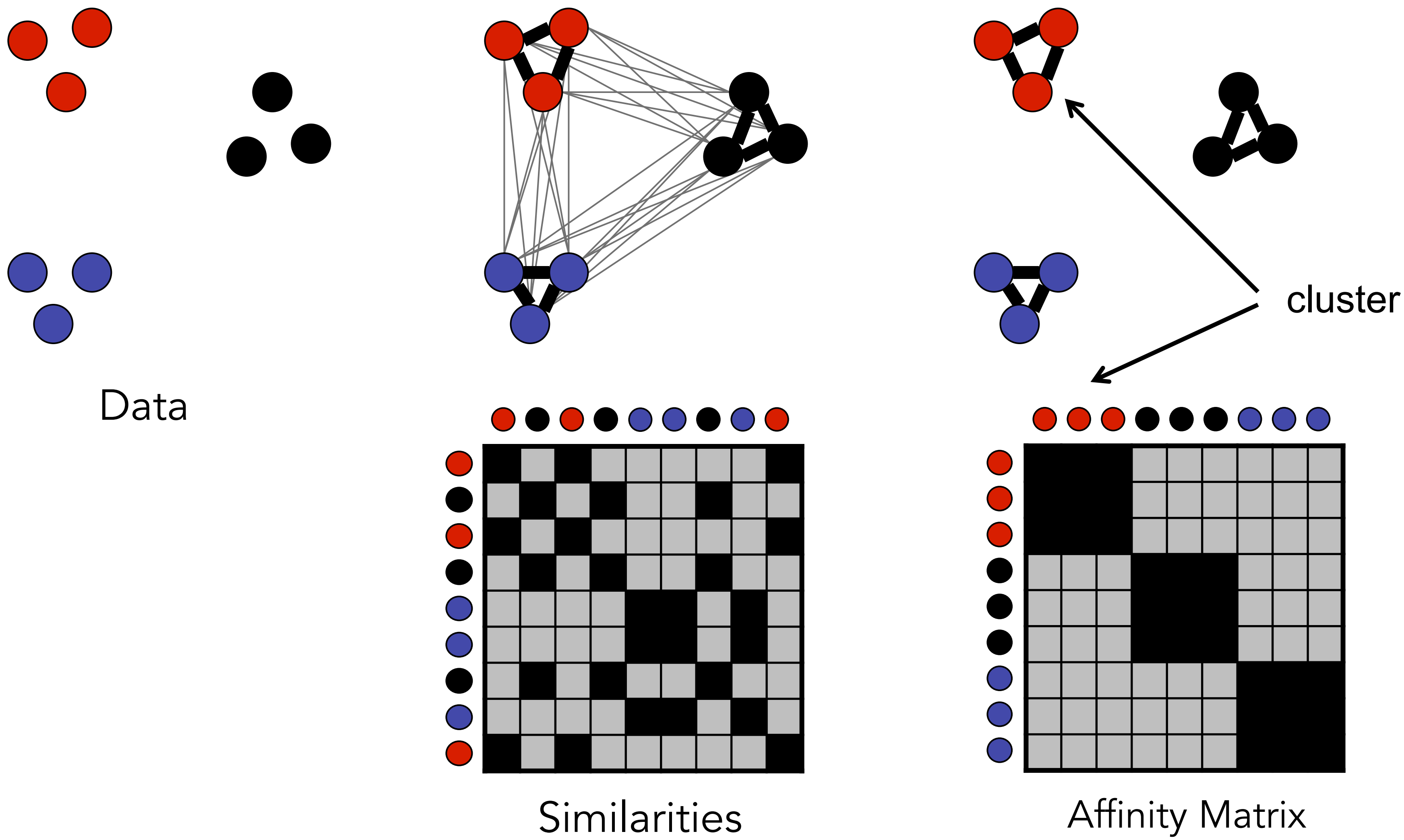
3- Calculate **eigenvectors**



4- Cut the graph:
apply **threshold** to eigenvectors



Spectral Clustering



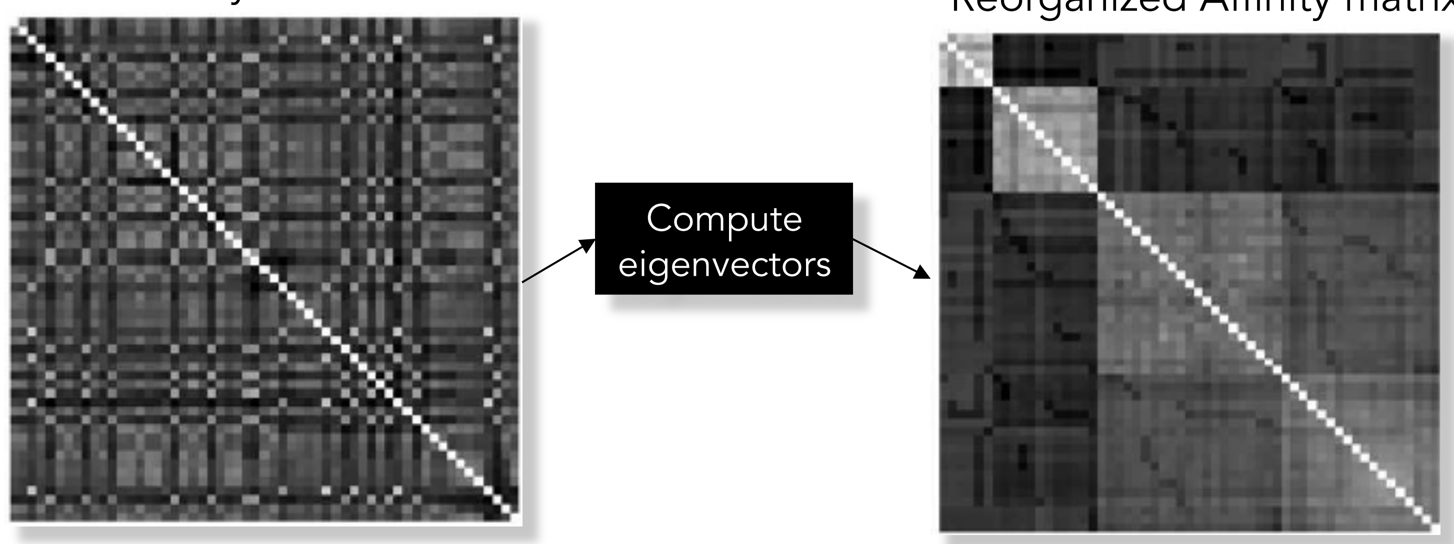
* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Spectral clustering: Using Eigenvalues of the matrix

- **spectral clustering** uses the eigenvalues of the similarity/affinity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions

Affinity matrix

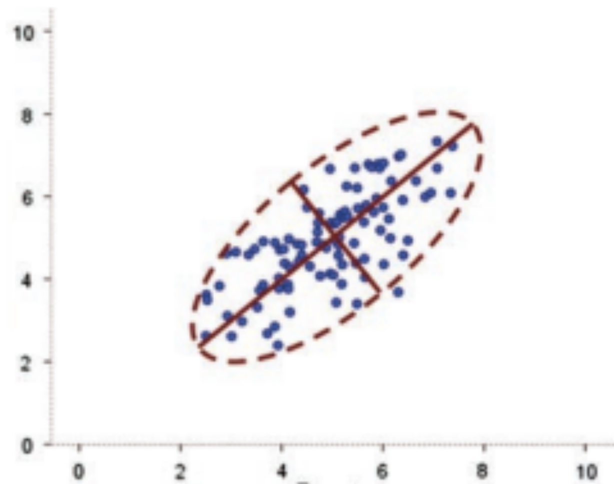
Reorganized Affinity matrix



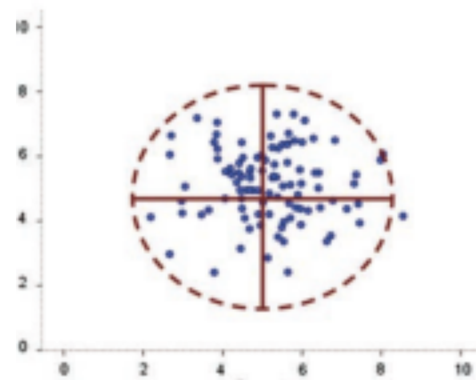
w(i,j)→ distance node i to node j

What are eigenvectors?

Eigenvectors represent the dimensions of data
Eigenvalues are the length of eigenvectors

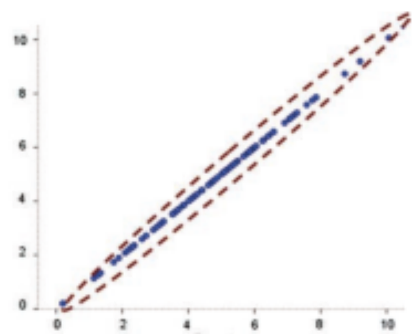


In a case of two variables,
Eigenvectors are the two lines
drawn in the scatterplot



$$\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}} = 1$$

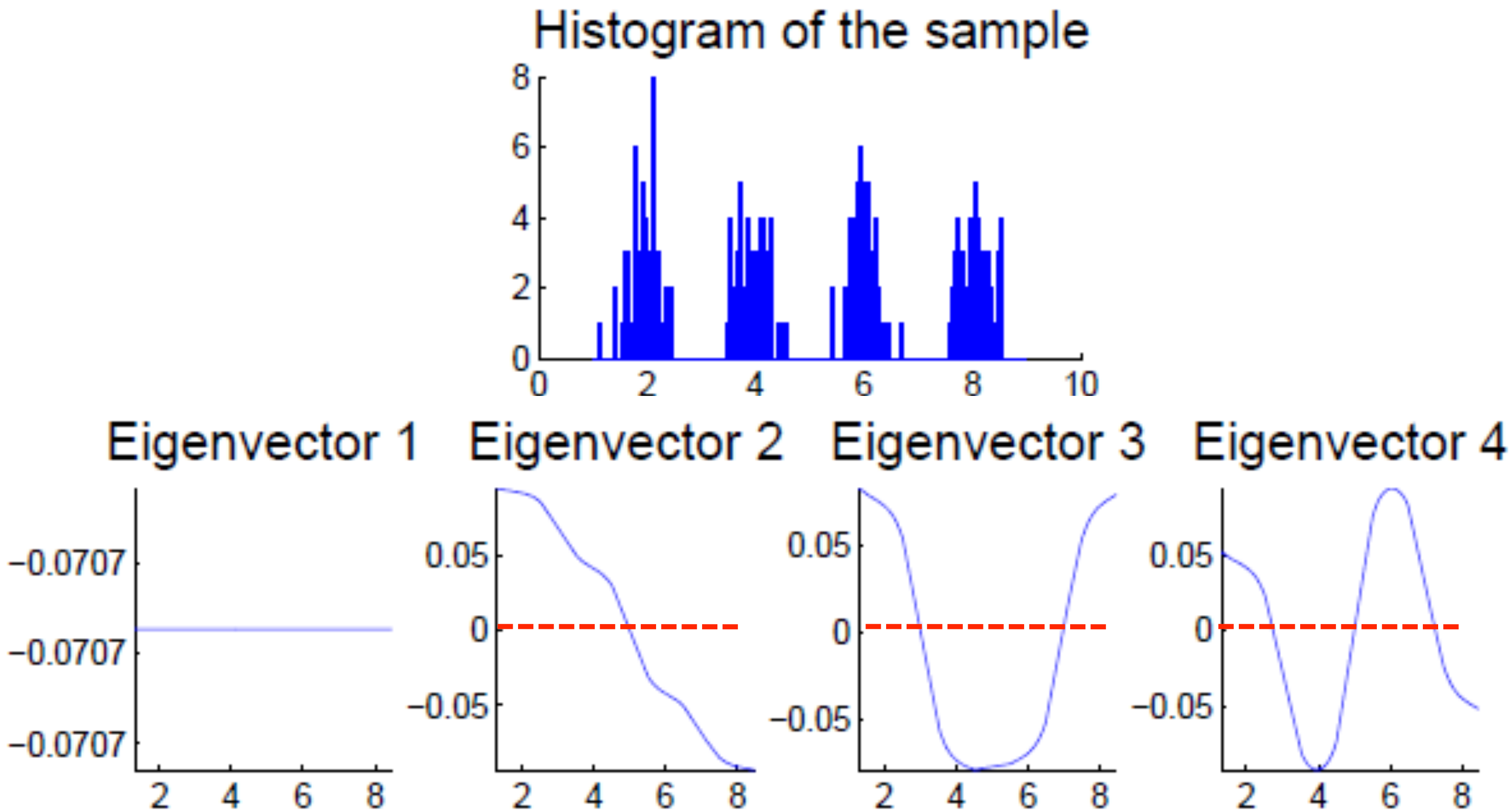
No relationship between variables



$$\frac{\text{largest eigenvalue}}{\text{smallest eigenvalue}} = \infty$$

A linear relationship between variables

Eigenvectors example



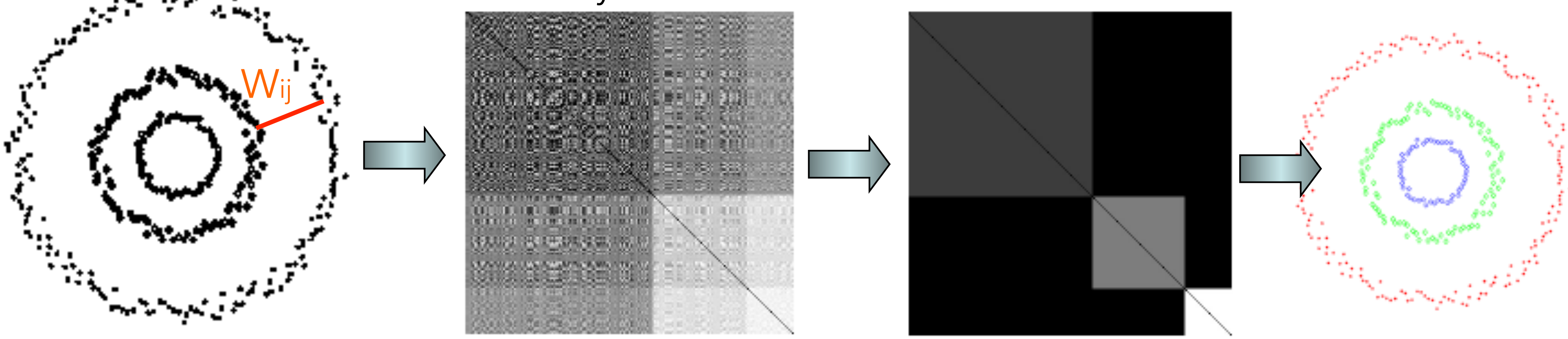
1st Eigenvector is the all ones vector *1 (if graph is connected)*
•2nd Eigenvector thresholded at 0 separates first two clusters from last two
• k-means clustering of the 4 eigenvectors identifies all 4 clusters

Spectral Clustering pipeline

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

Affinity matrix M

Matrix V after Spectral Clustering



Data are projected into a lower-dimensional space (spectral/eigenvector domain) where they are easily separable

Given number k of clusters, compute the first k eigenvectors, $V_1, \dots, V_k$ of the affinity matrix M

Build the matrix V with the eigenvectors as columns

Interpret the rows of V as new data points Z_i

Cluster the points Z_i with the k-means algorithms

	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Z_n	v_{n1}	v_{n2}	v_{n3}

Dimensionality reduction
n x n → n x k

Eigenvectors and blocks

- Block weight matrices have block eigenvectors:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \quad \overset{\lambda_1 = 2}{\begin{bmatrix} .71 \\ .71 \\ 0 \\ 0 \end{bmatrix}} \quad \overset{\lambda_2 = 2}{\begin{bmatrix} 0 \\ 0 \\ .71 \\ .71 \end{bmatrix}} \qquad \lambda_3 = 0 \quad \lambda_4 = 0$$

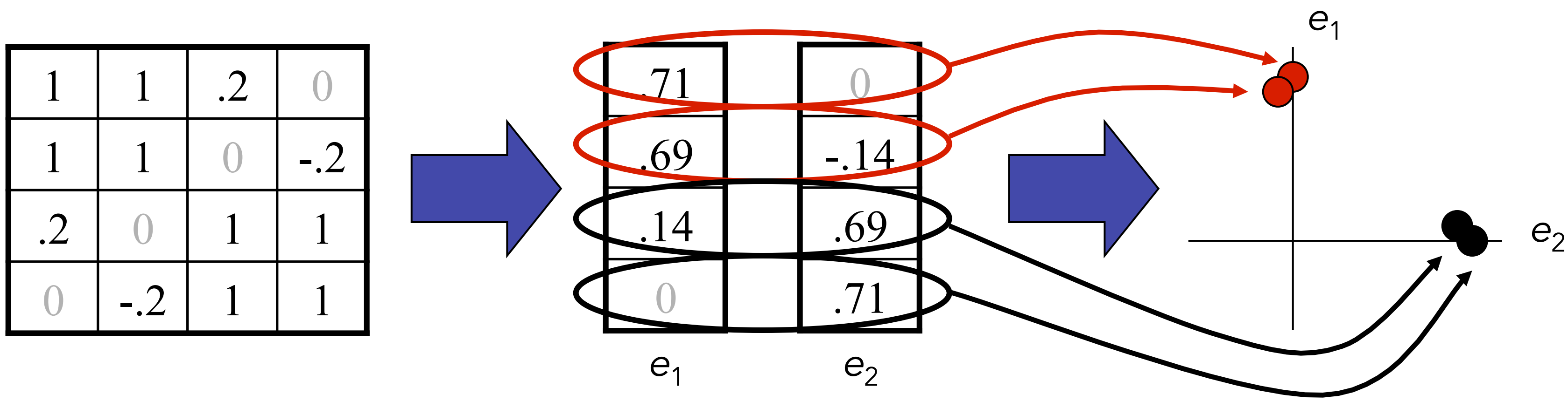
- Near-block matrices have near-block eigenvectors:

$$\begin{bmatrix} 1 & 1 & .2 & 0 \\ 1 & 1 & 0 & -.2 \\ .2 & 0 & 1 & 1 \\ 0 & -.2 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \quad \underset{e1}{\overset{\lambda_1 = 2.02}{\begin{bmatrix} .71 \\ .69 \\ .14 \\ 0 \end{bmatrix}}} \quad \underset{e2}{\overset{\lambda_2 = 2.02}{\begin{bmatrix} 0 \\ -.14 \\ .69 \\ .71 \end{bmatrix}}} \qquad \lambda_3 = -0.02 \quad \lambda_4 = -0.02$$

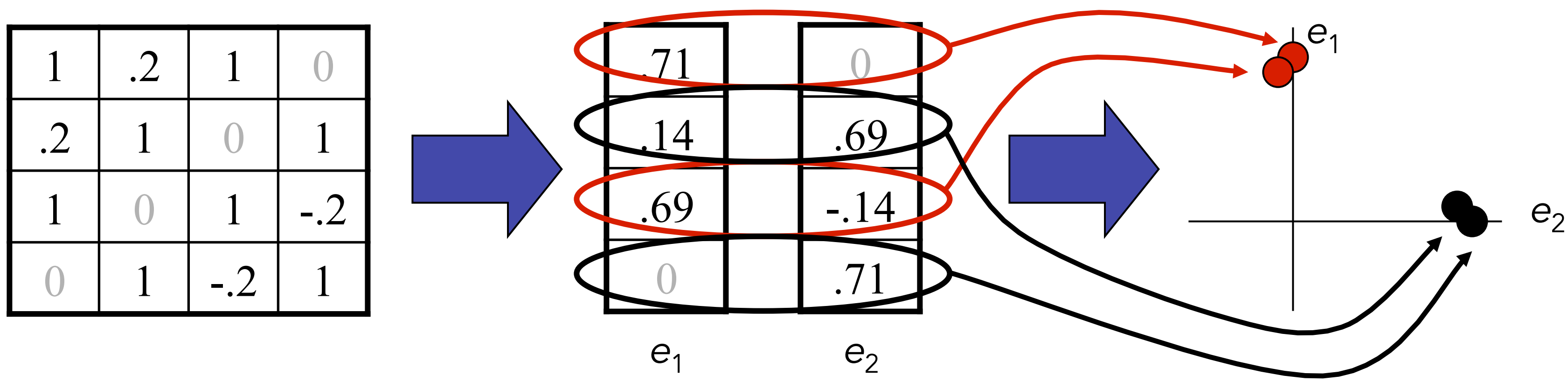
* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Spectral Space

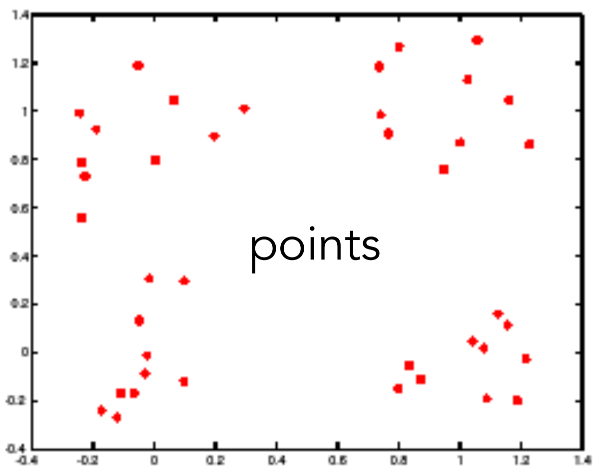
Can put items into blocks by eigenvectors:

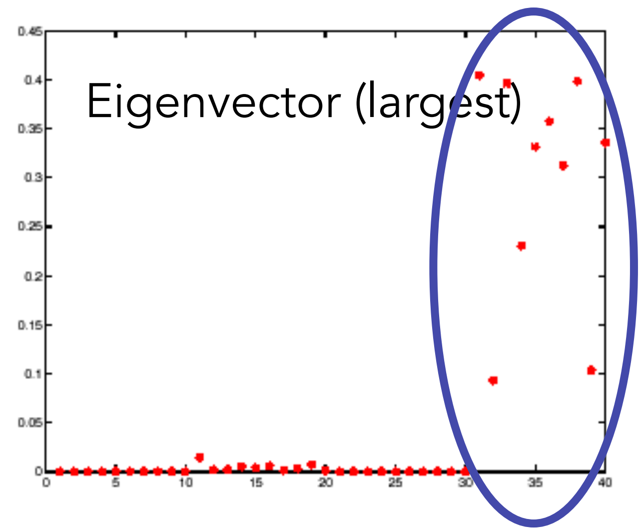


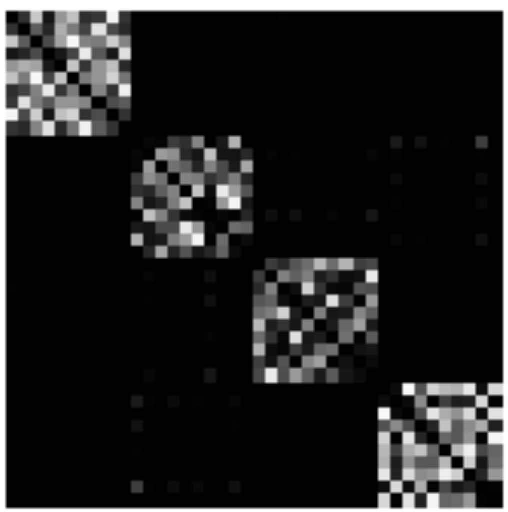
Clusters clear regardless of row ordering:



Example eigenvector

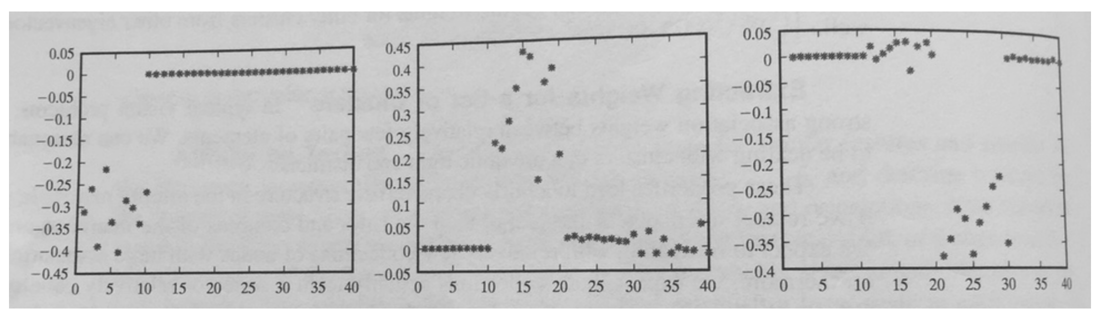






Affinity matrix

The eigenvector corresponding to the largest eigenvalue of the affinity matrix. Most values are small, but some, corresponding to the elements of the main cluster, are large



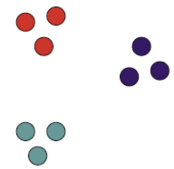
The 3 next eigenvectors corresponding to the next 3 largest eigenvalues of the affinity matrix. Most values are small but for (disjoint) sets of elements the values are large. This follows from the block structure of the affinity matrix

See Forsyth Ponce, Chapter 14

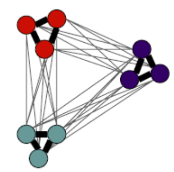
Graph-based Image Segmentation

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.

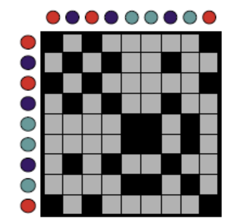
1- Get vectors of **data**



2a- Build a **similarity** graph

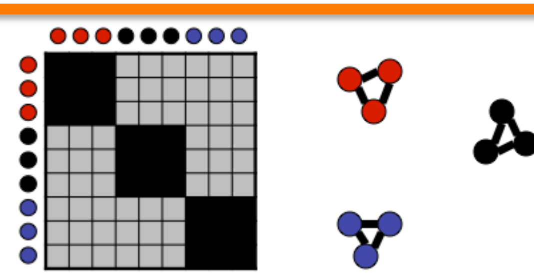


2b- Build a similarity **matrix**

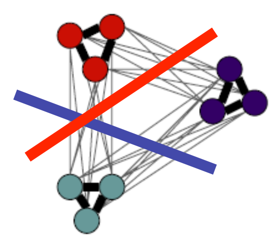


Similarities

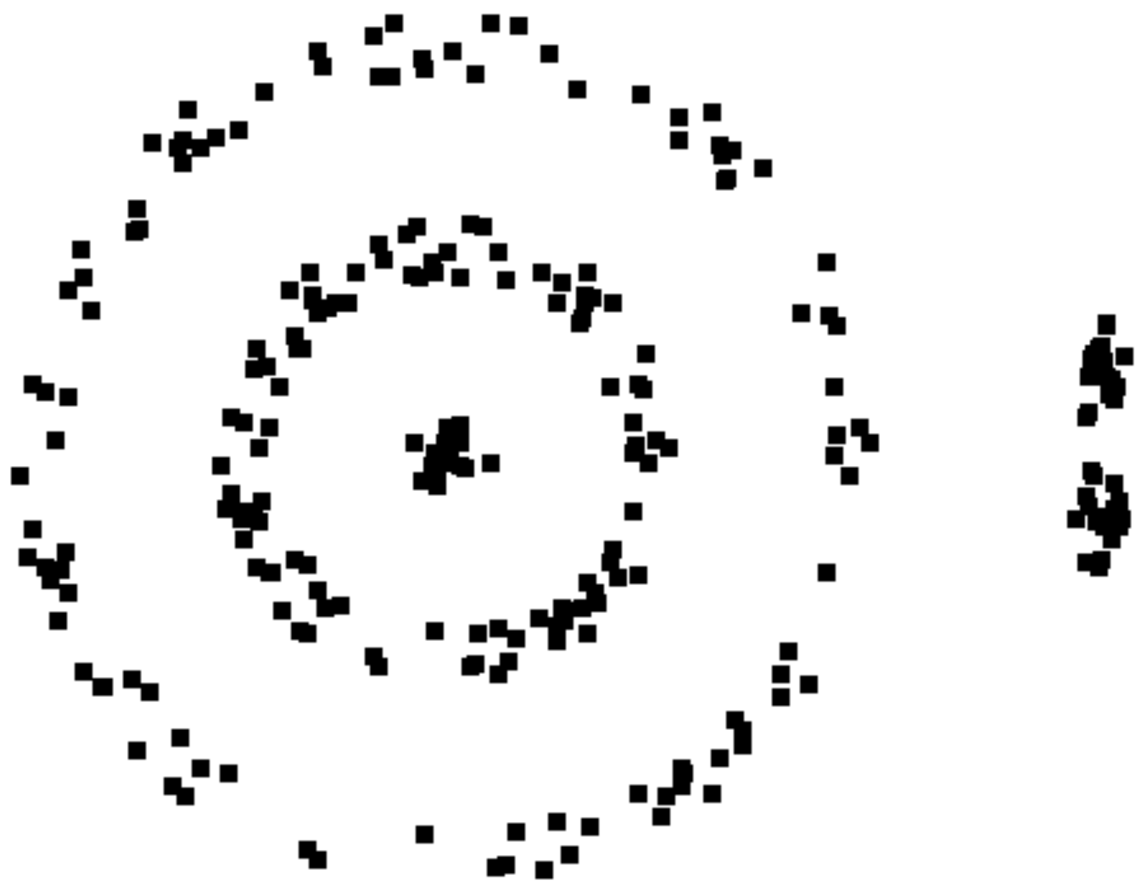
3- Calculate **eigenvectors**



4- Cut the graph:
apply **threshold** to eigenvectors

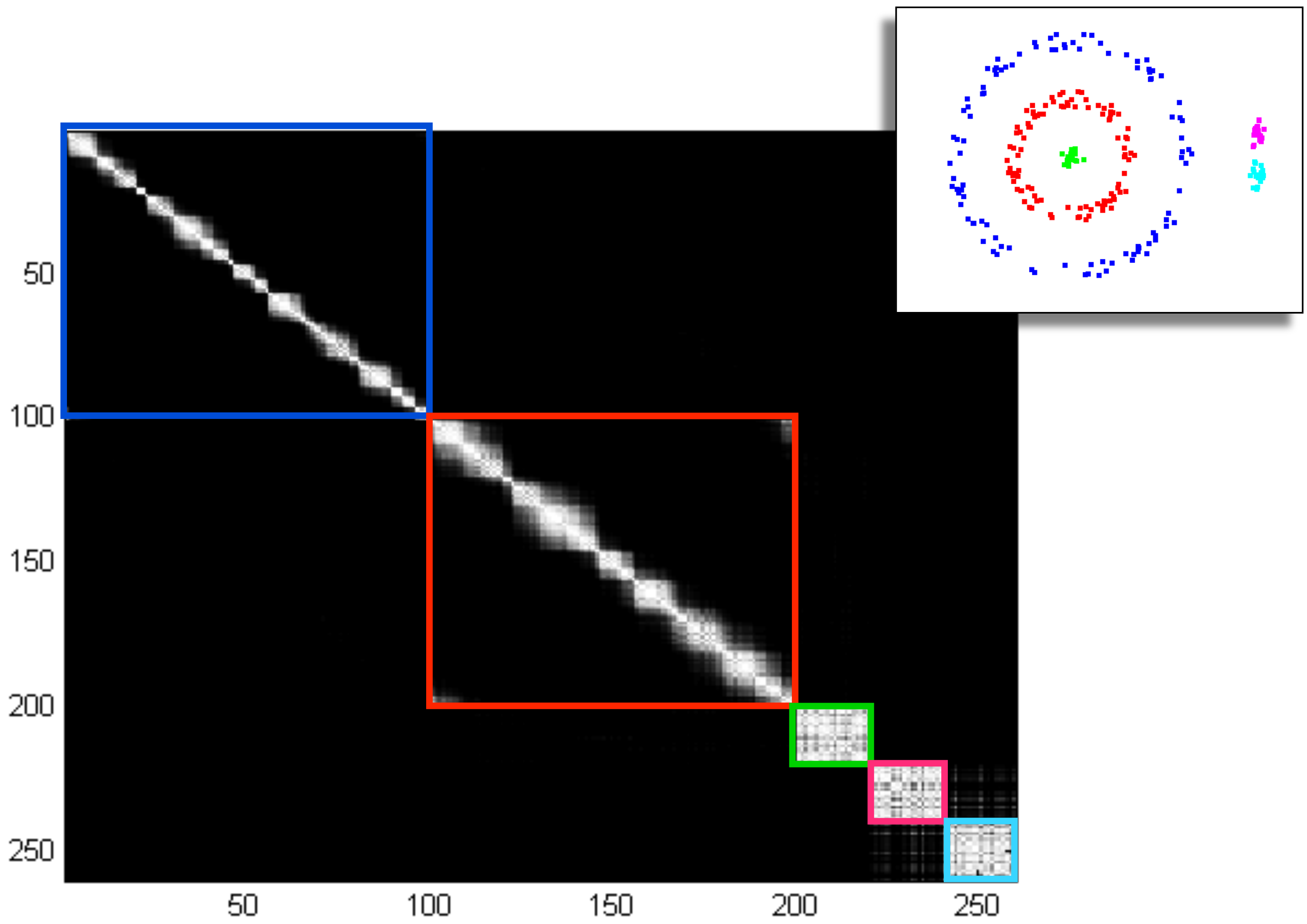


Clustering – How many groups are there?

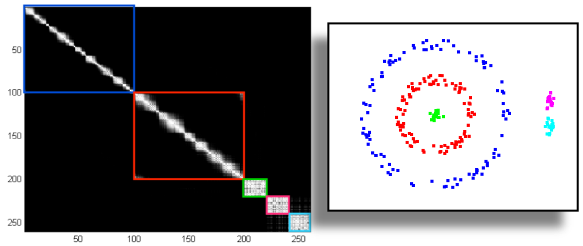


Out of the various possible partitions, which is the correct one?

What does the Affinity Matrix Look Like?

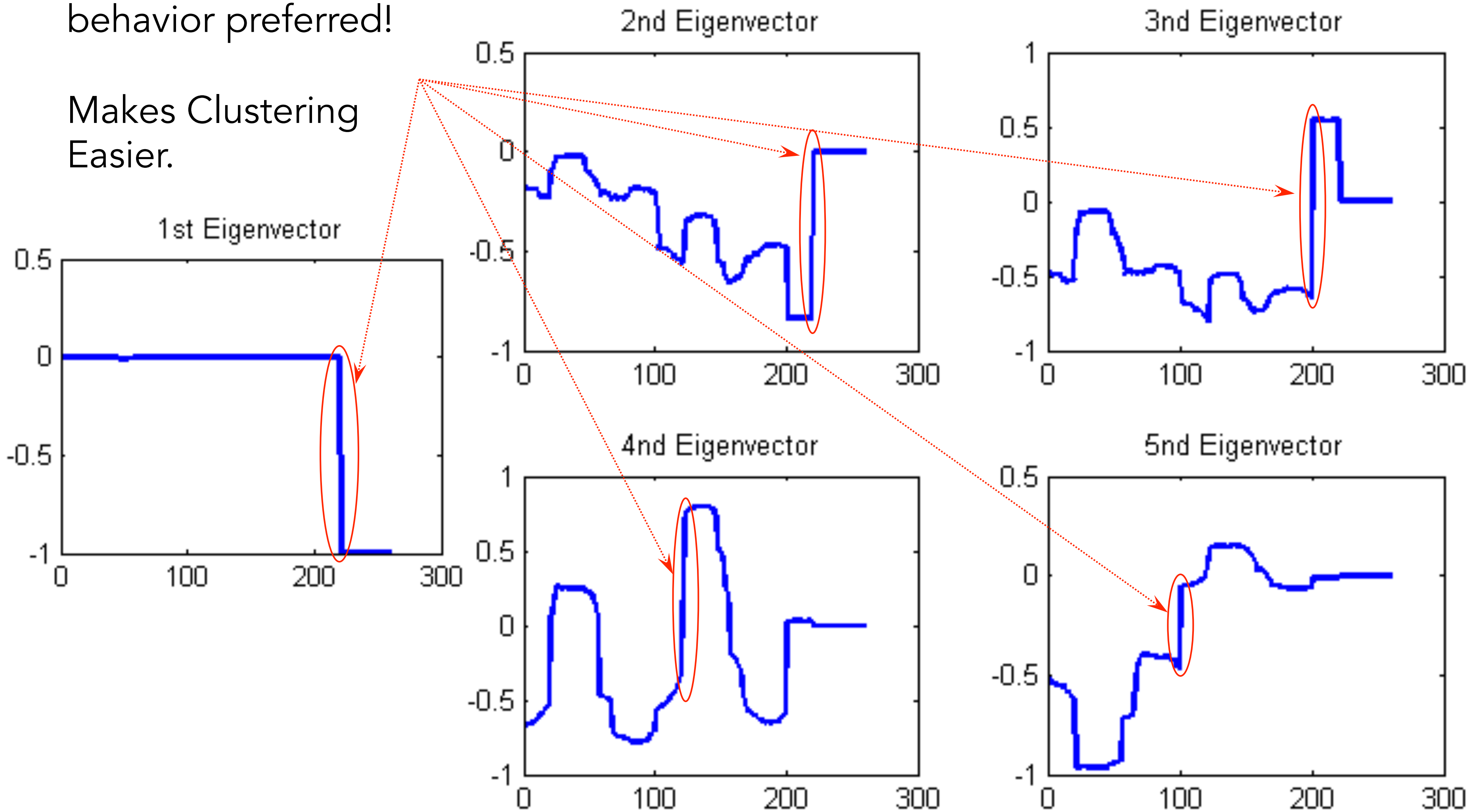


The Eigenvectors and the Clusters

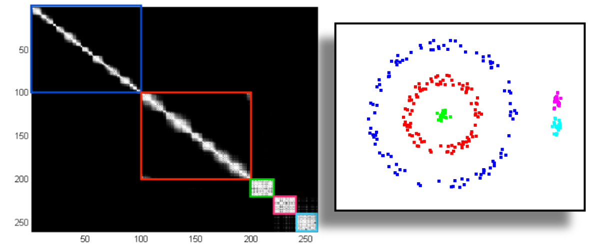


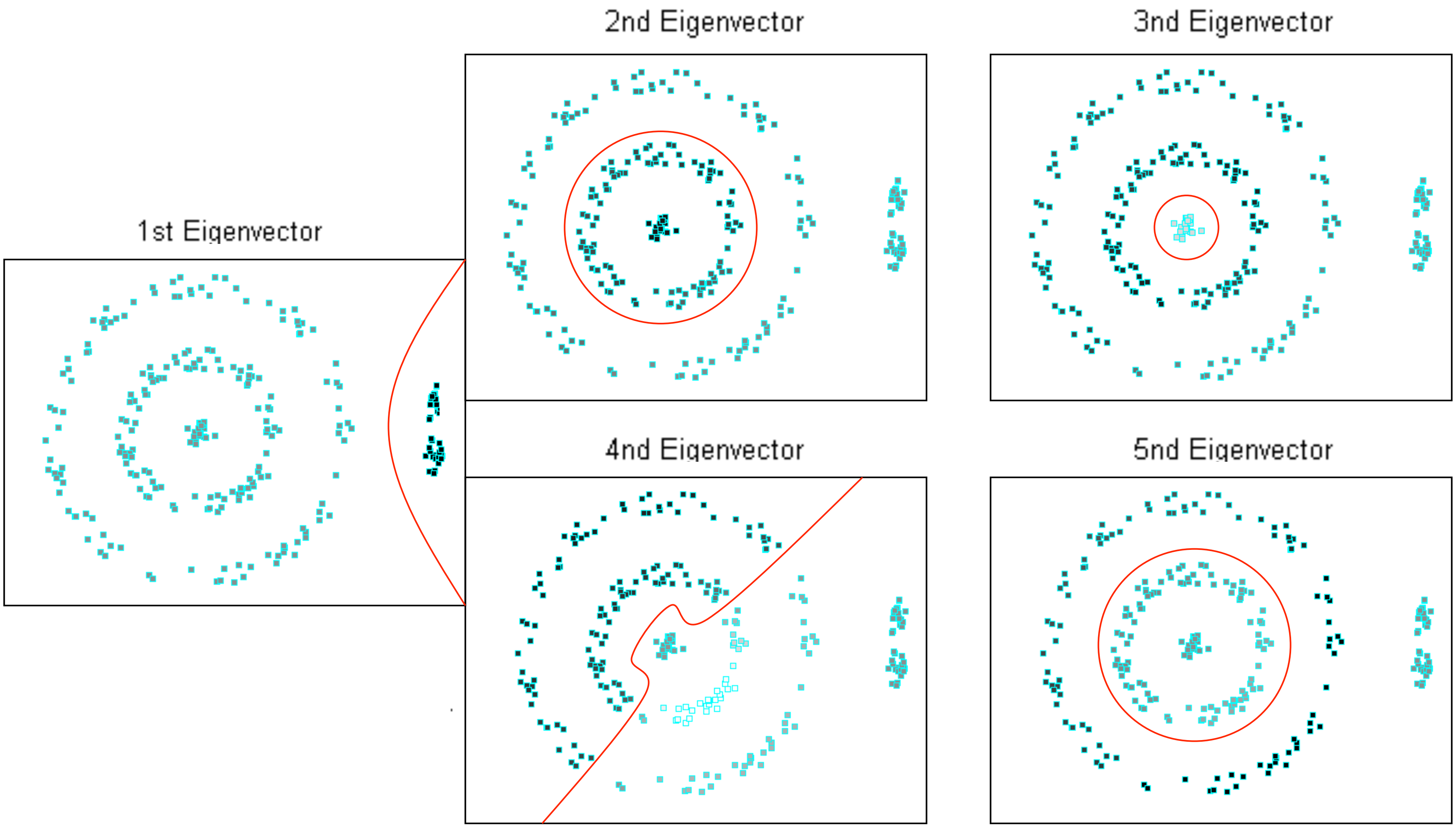
Step-Function like behavior preferred!

Makes Clustering Easier.

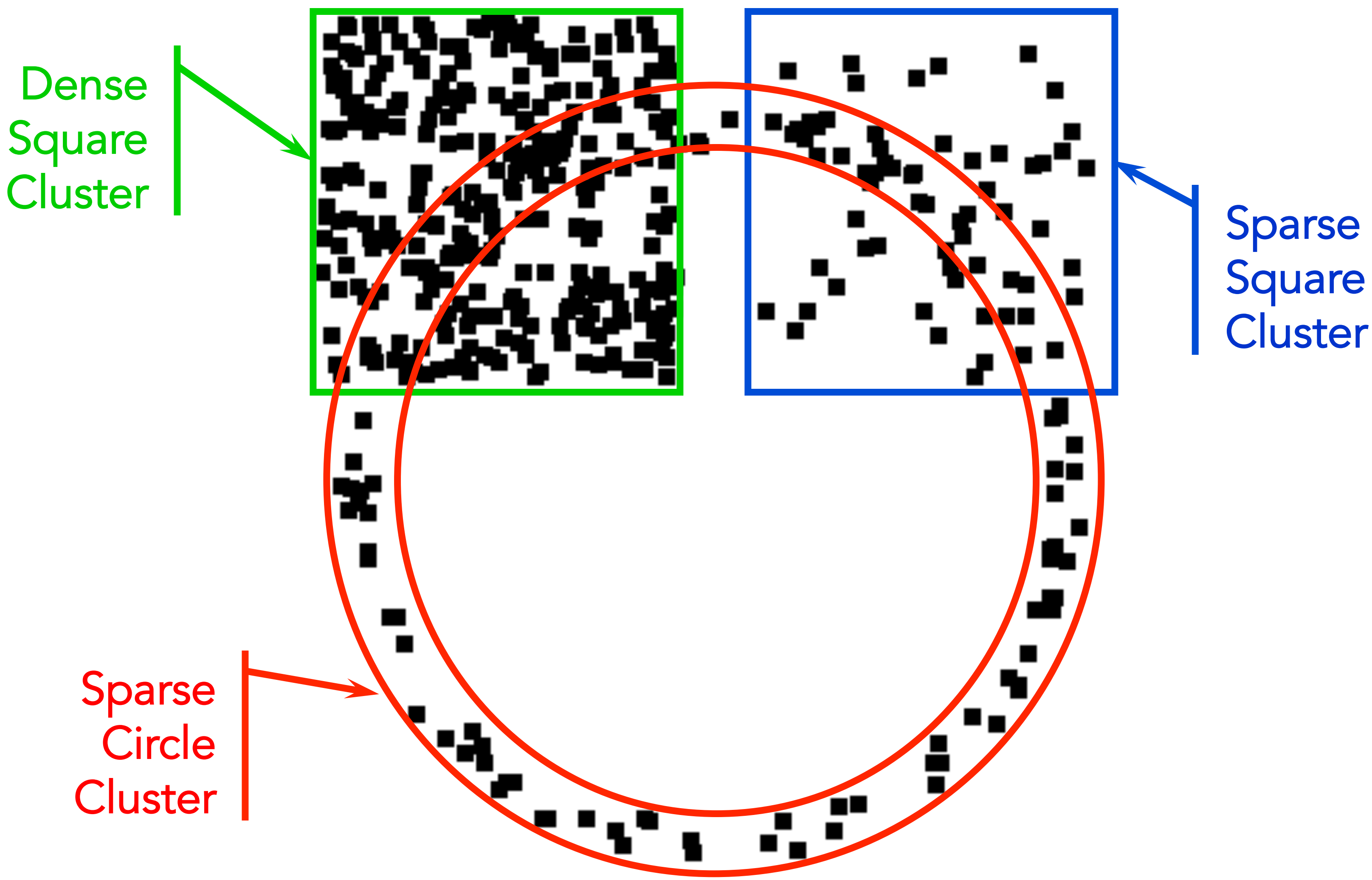


The Eigenvectors and the Clusters

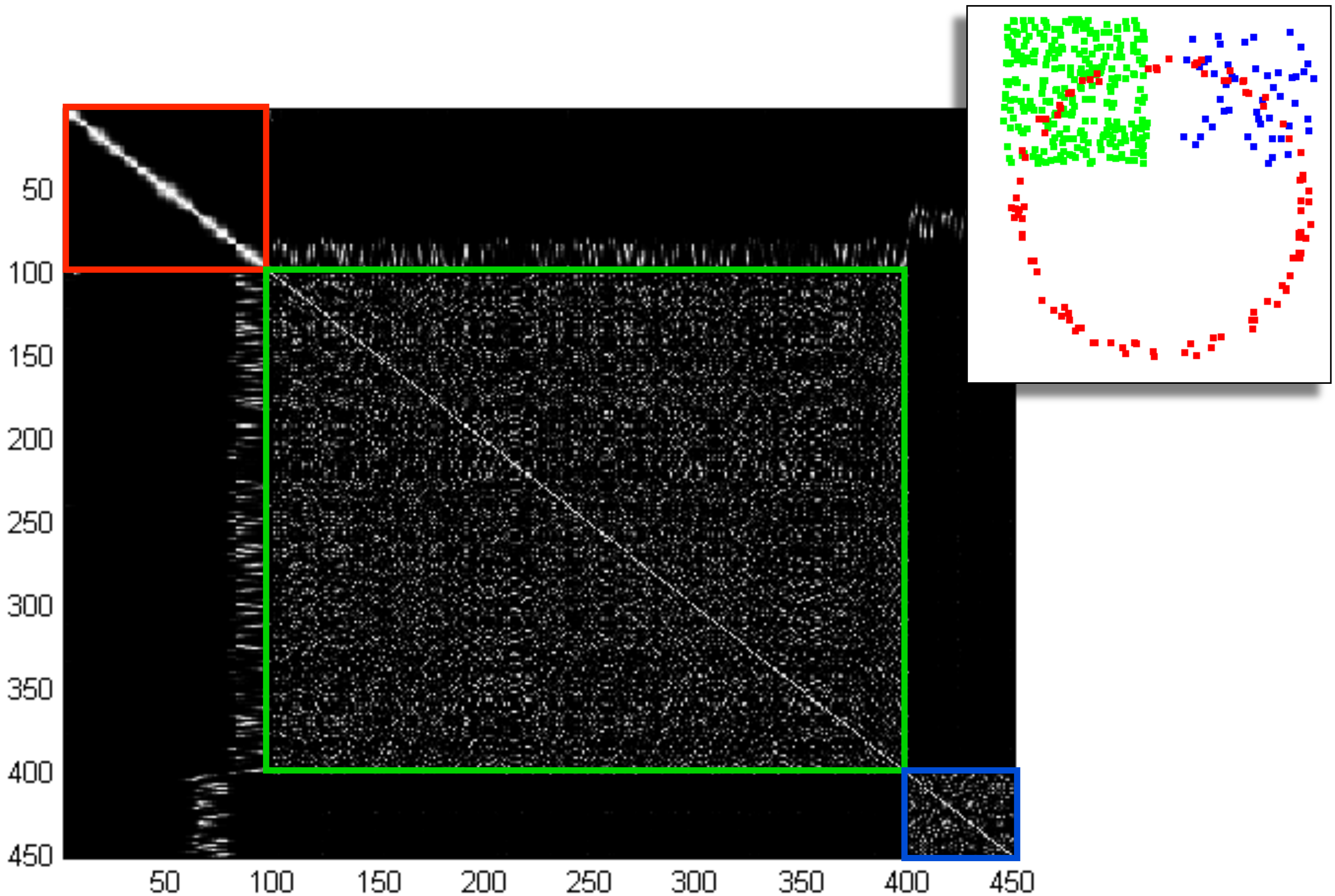


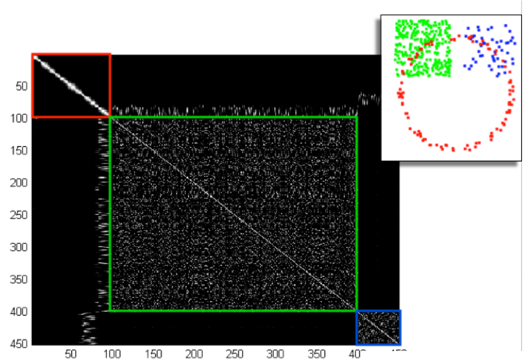


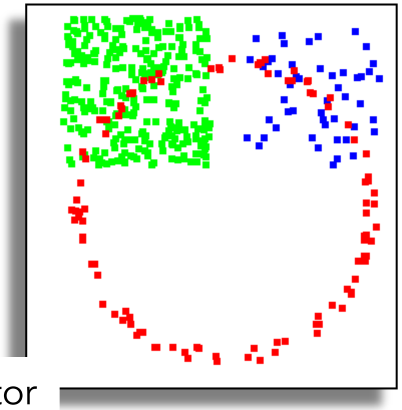
Clustering – Example 2



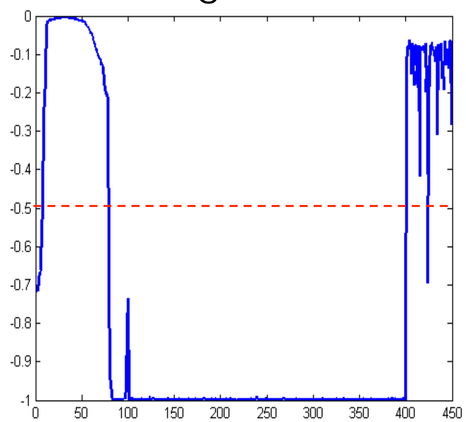
The Affinity Matrix



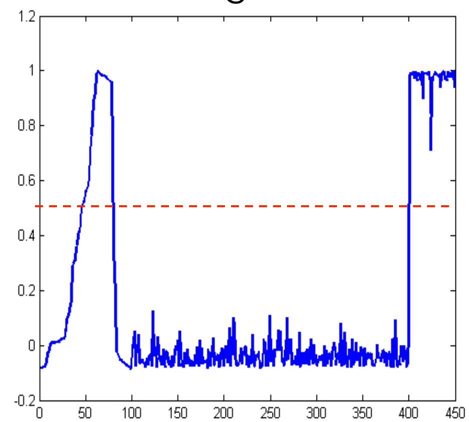




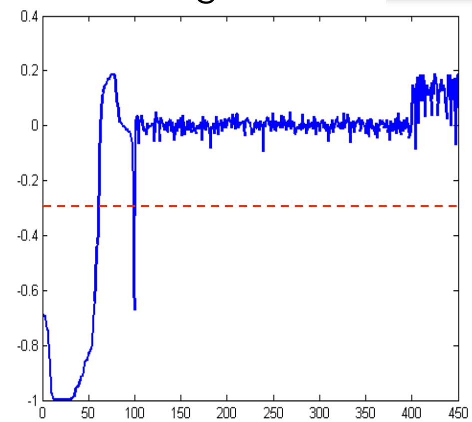
1st Eigenvector

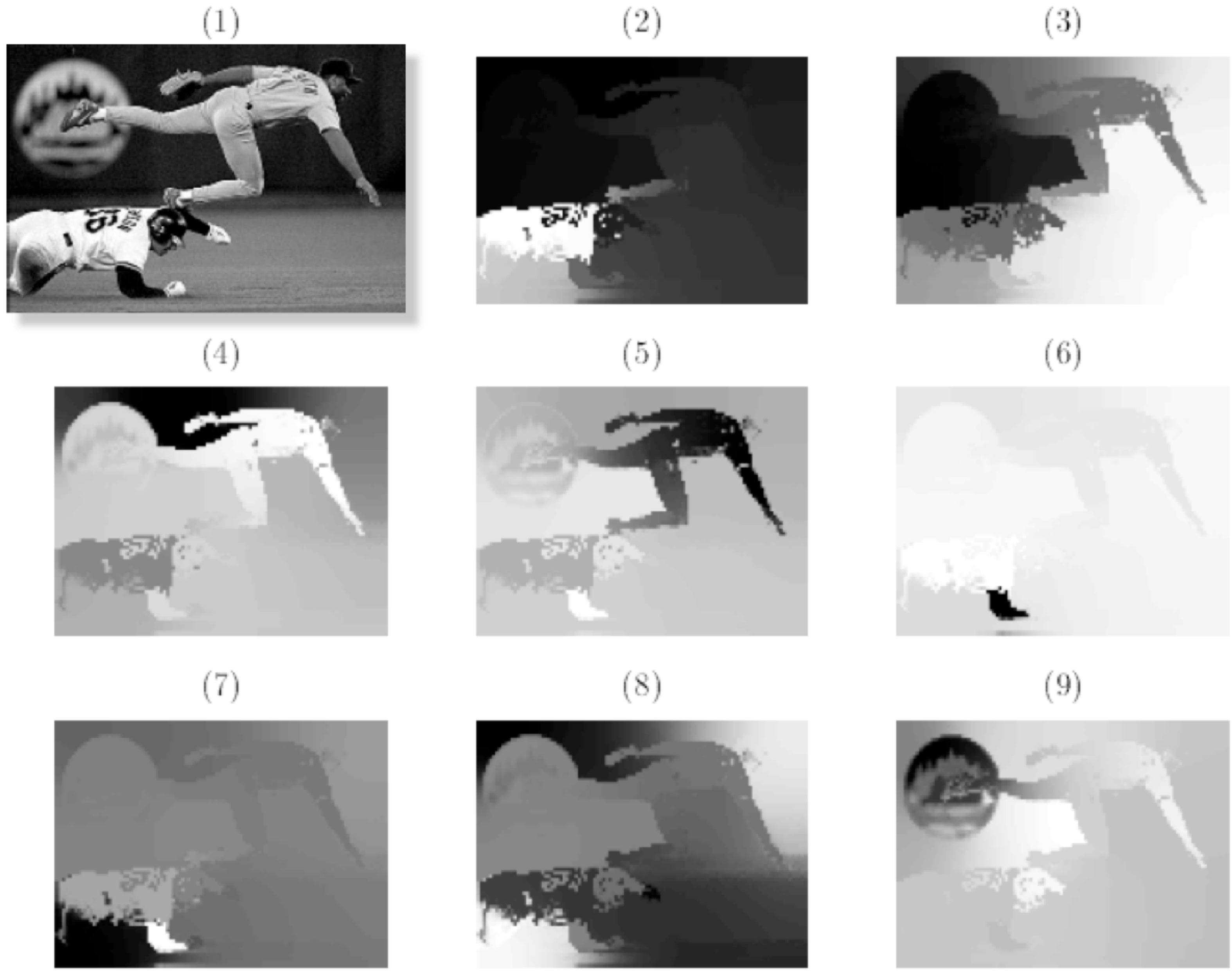


2nd Eigenvector



3rd Eigenvector

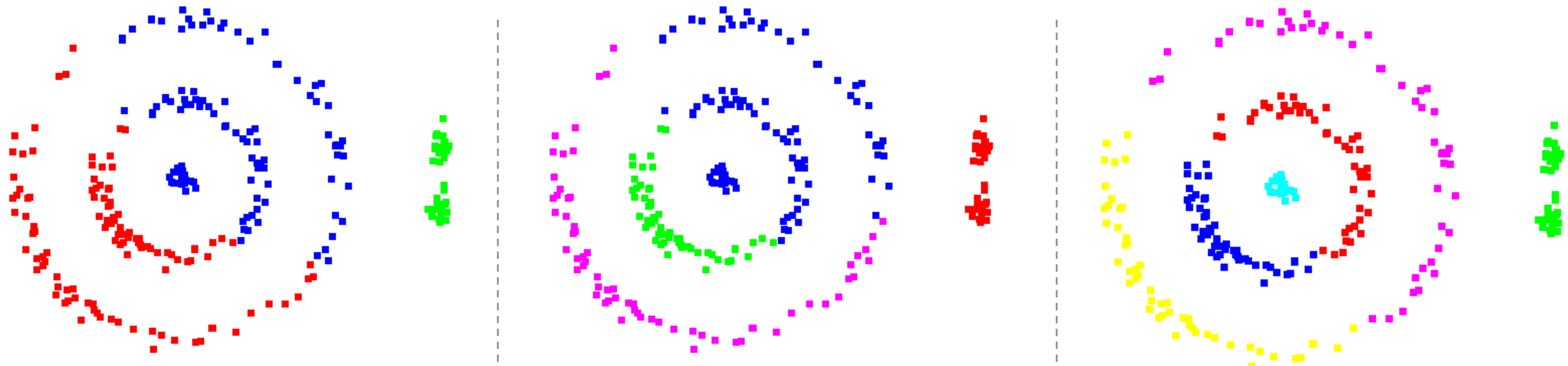




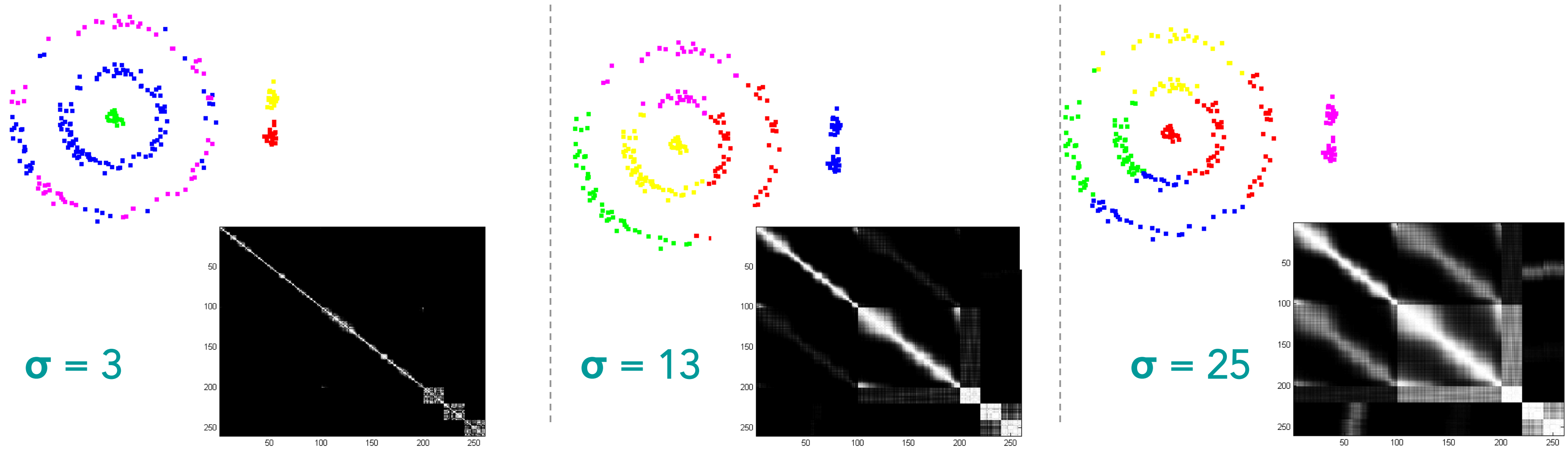
The eigenvectors correspond the 2nd smallest to the 9th smallest eigenvalues

Issue: Number of Clusters ?

k = 3 k = 4 k = 6



Issue: choice of kernel, for Gaussian kernels, choice of $\boldsymbol{\sigma}$

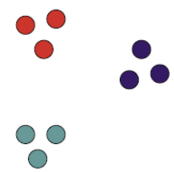


$\sigma = 3$ $\sigma = 13$ $\sigma = 25$

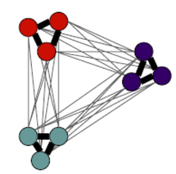
Graph-based Image Segmentation

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.

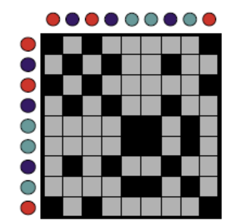
1- Get vectors of **data**



2a- Build a **similarity** graph

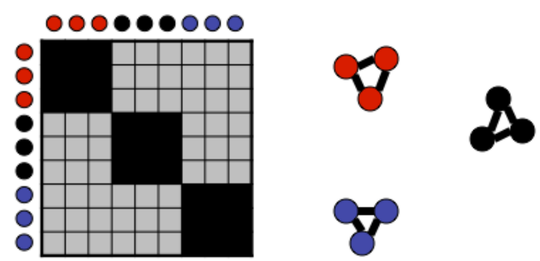


2b- Build a similarity **matrix**

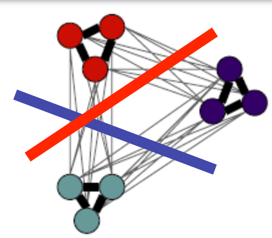


Similarities

3- Calculate **eigenvectors**



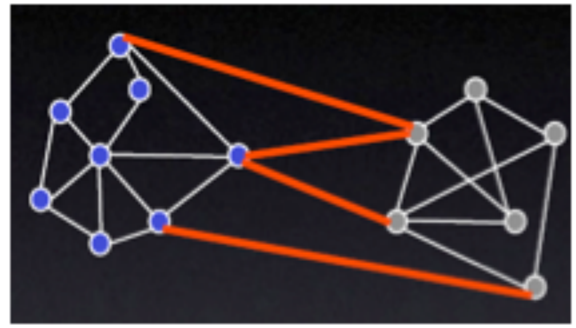
4- Cut the graph:
apply **threshold** to eigenvectors

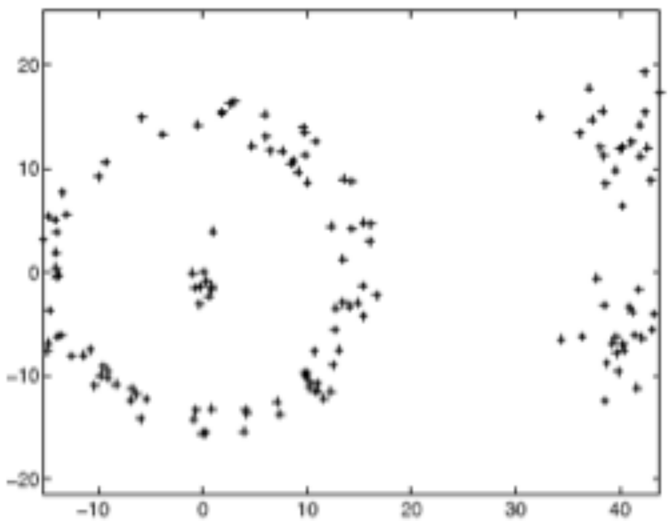


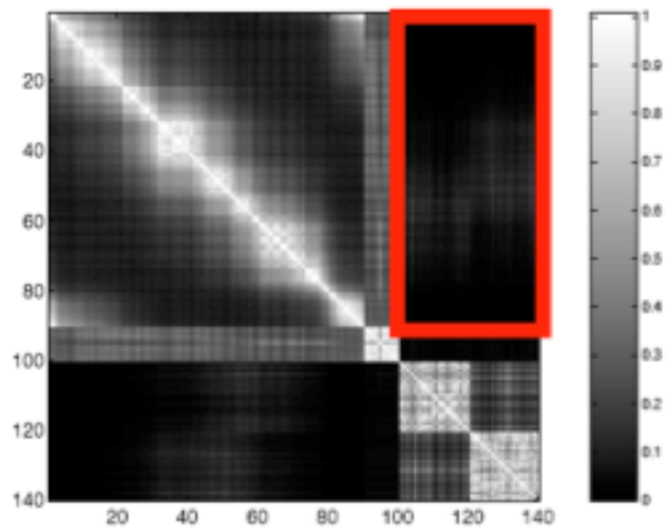
Graph cut

Cuts in a graph:

$$cut(A,\overline{A}) = \sum_{i\in A,\, j\in \overline{A}} w_{i,j}$$

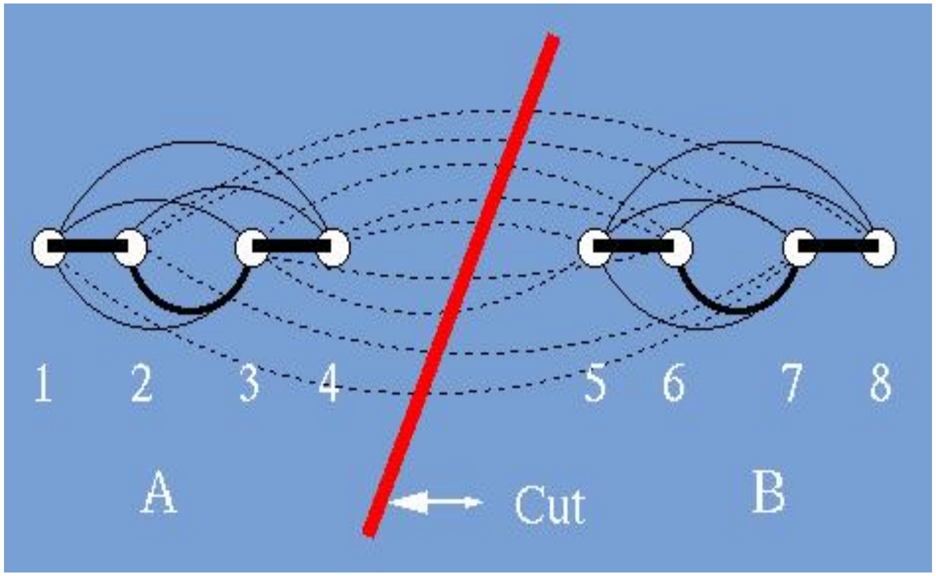






- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation

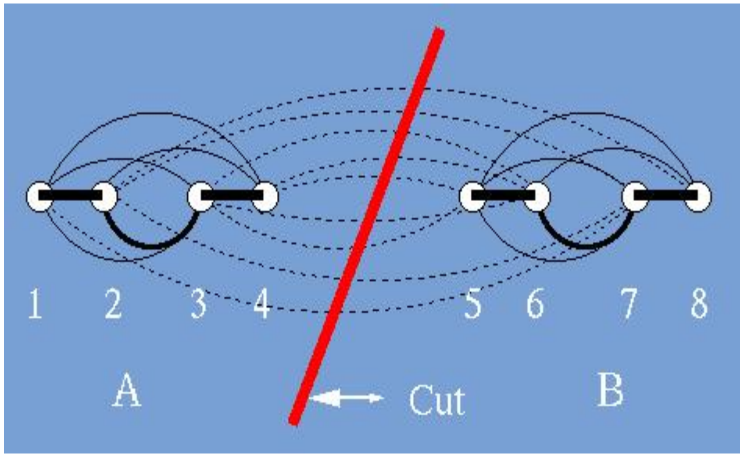
Partition a graph with minimum cut



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

- **Cut**: sum of the weight of the cut edges:
- Minimum cut is the cut of minimum weight

Normalized Cut is a better measure ..



- We normalize by the total volume of connections

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

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

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$

Many different 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

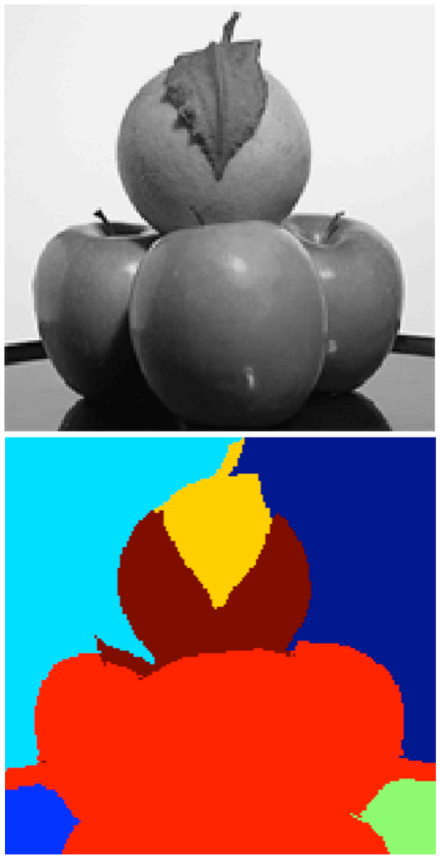
3- Get eigenvectors with smallest eigenvalues

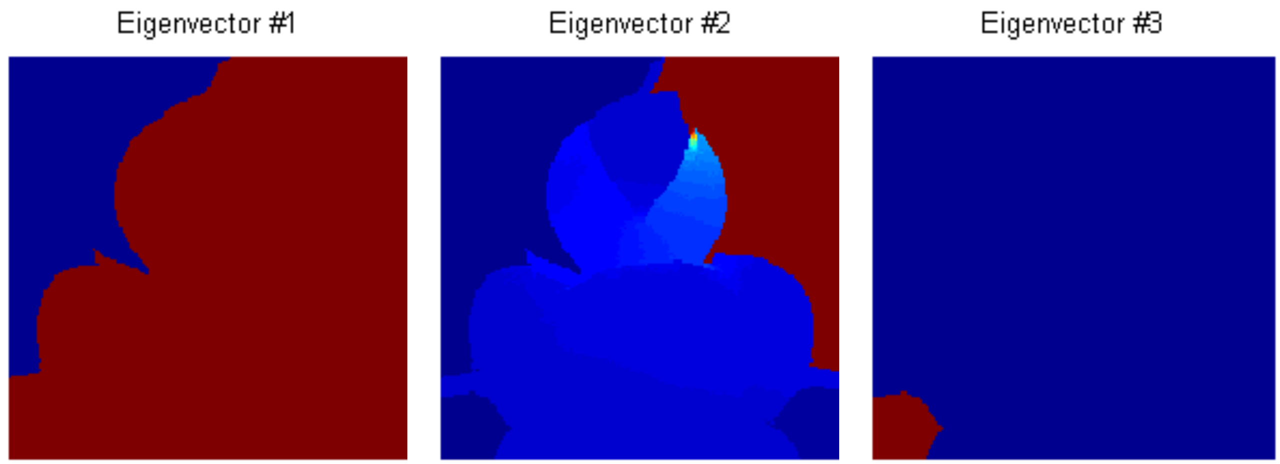
4- Apply threshold

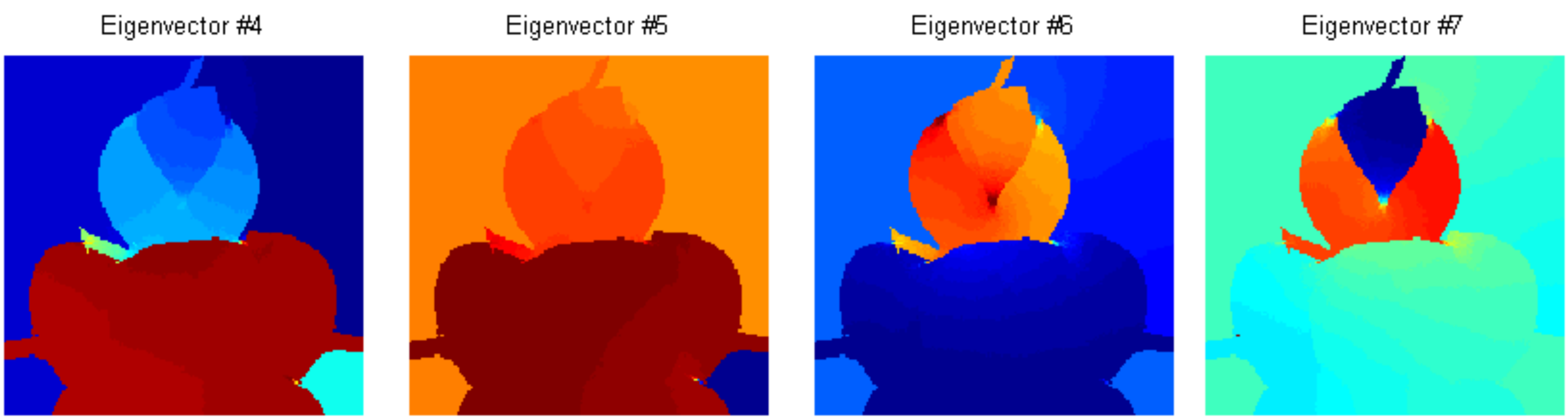
Shi & Malik

… etc

Normalized cut

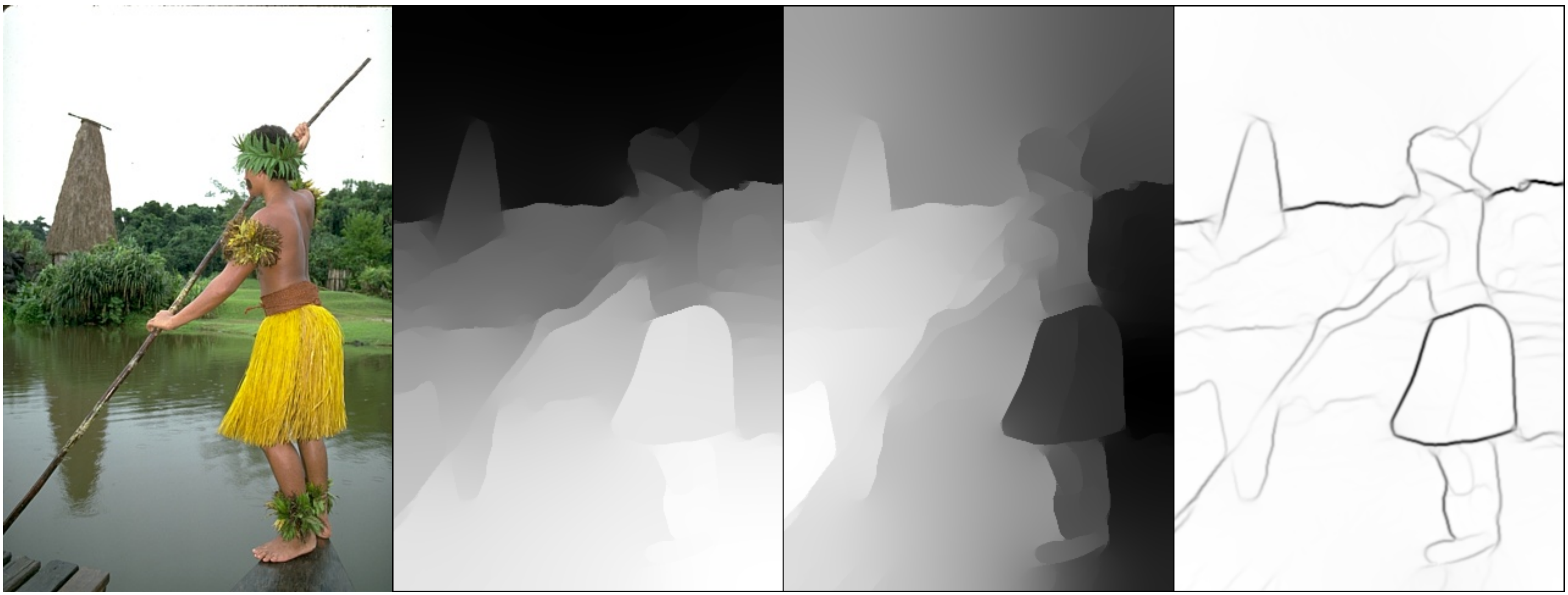






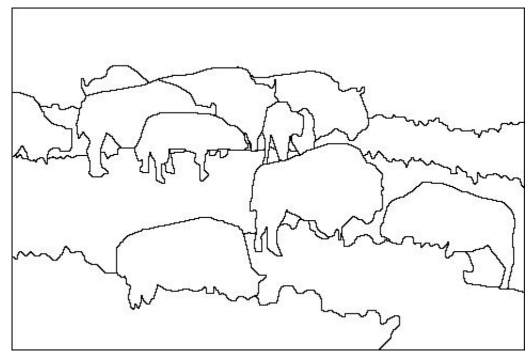
Normalized cut

Normalized cut



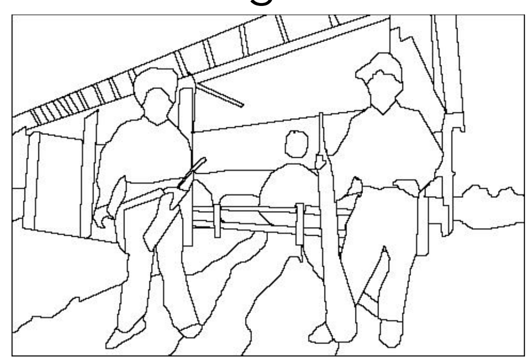
II.2 Top-down Segmentation

- Separate image into coherent “things”: combining object recognition with segmentation
  - Supervised or unsupervised



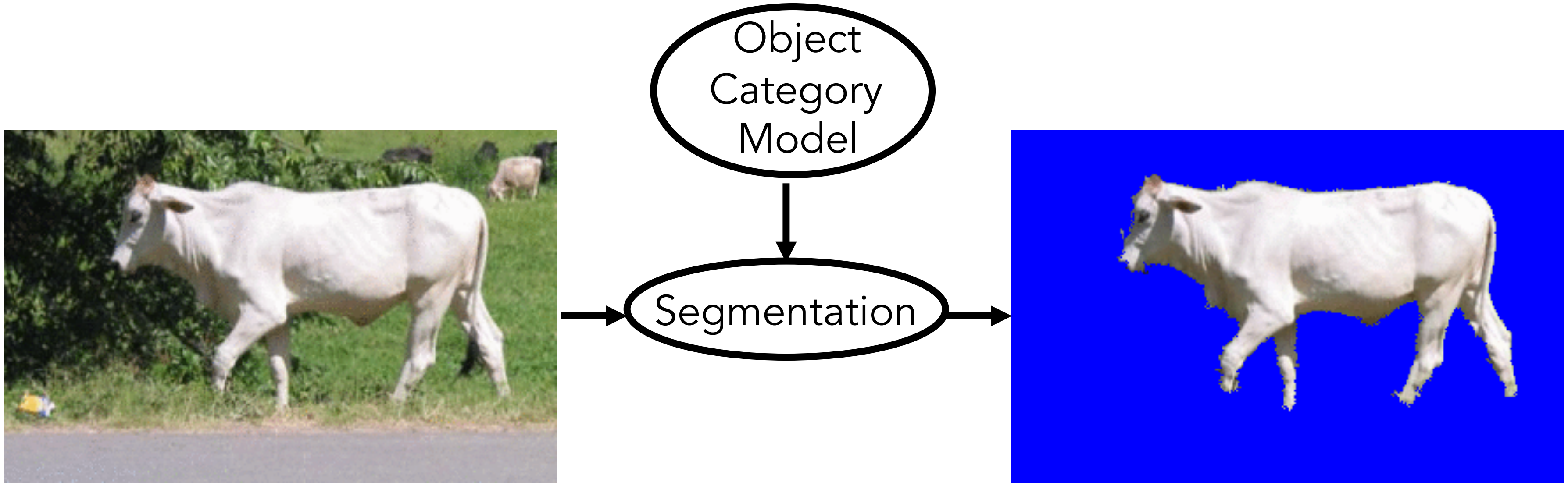
image

human segmentation



Berkeley segmentation database

Aim: Given an image and object category, to segment the object



Cow Image

Segmented Cow

Segmentation should (ideally) be

- shaped like the object e.g. cow-like
- obtained efficiently in an unsupervised manner
- able to handle self-occlusion

Slide from Kumar ‘05

Examples of bottom-up segmentation

- Using Normalized Cuts, Shi & Malik, 1997