COS 429: COMPUTER VISON

Segmentation

- human vision: grouping
- k-means clustering
- graph-theoretic clustering
- Hough transform
- line fitting
- RANSAC

Reading: Chapters 14, 15

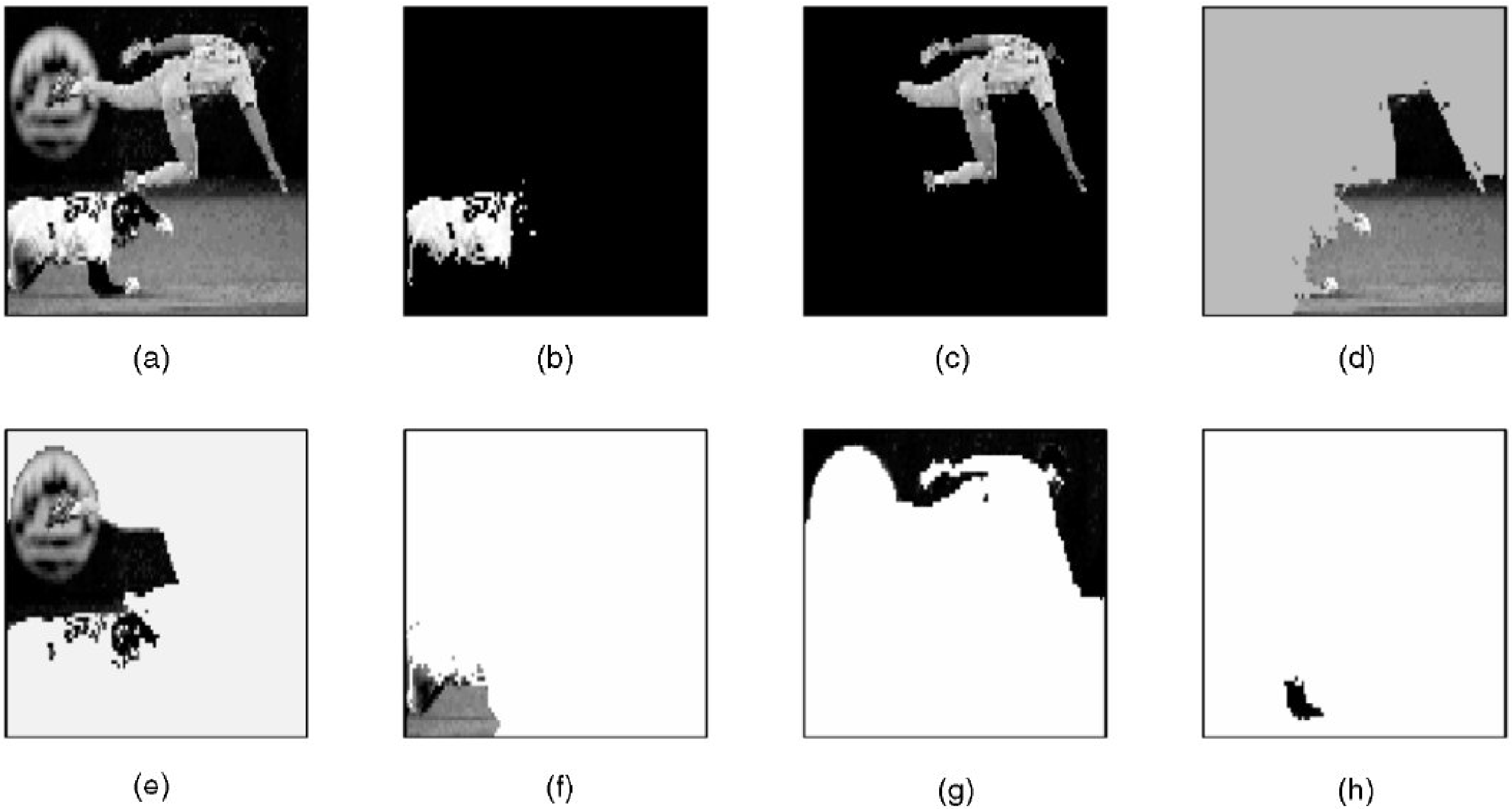
Some of the slides are credited to: David Lowe, David Forsyth, Christopher Rasmussen

Segmentation by Clustering

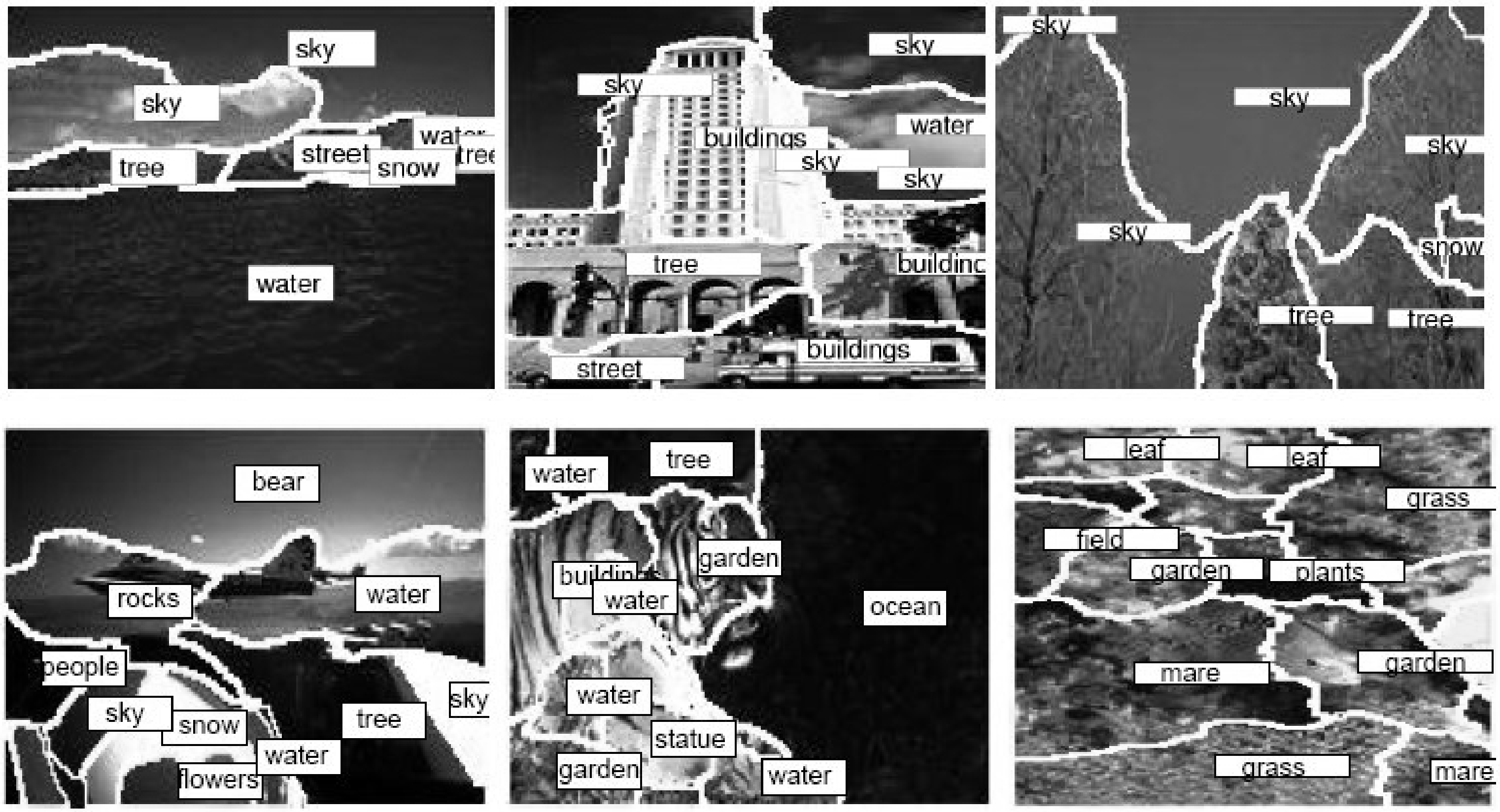
- Data reduction - obtain a compact representation for *interesting* image data in terms of a set of components
- Find components that belong together (form clusters)

Segmentation by Clustering

Segmentation by Clustering



Segmentation by Clustering



From: Object Recognition as Machine Translation, Duygulu, Barnard, **de Freitas**, Forsyth, ECCV02

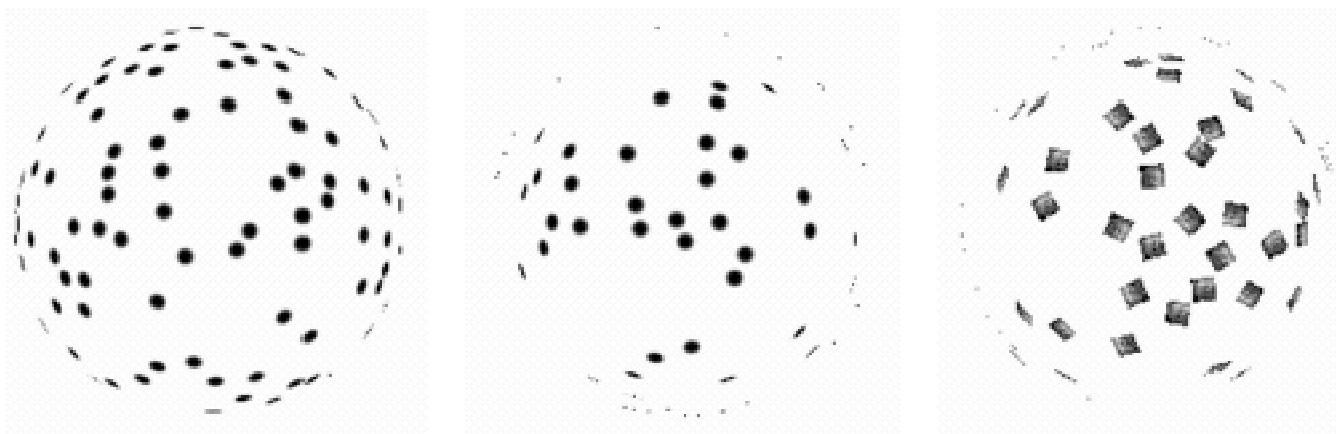
General ideas

- Tokens
  - whatever we need to group (pixels, points, surface elements, etc., etc.)
- Bottom up segmentation
  - tokens belong together because they are locally coherent
- Top down segmentation
  - tokens belong together because they lie on the same object
- These two are not mutually exclusive

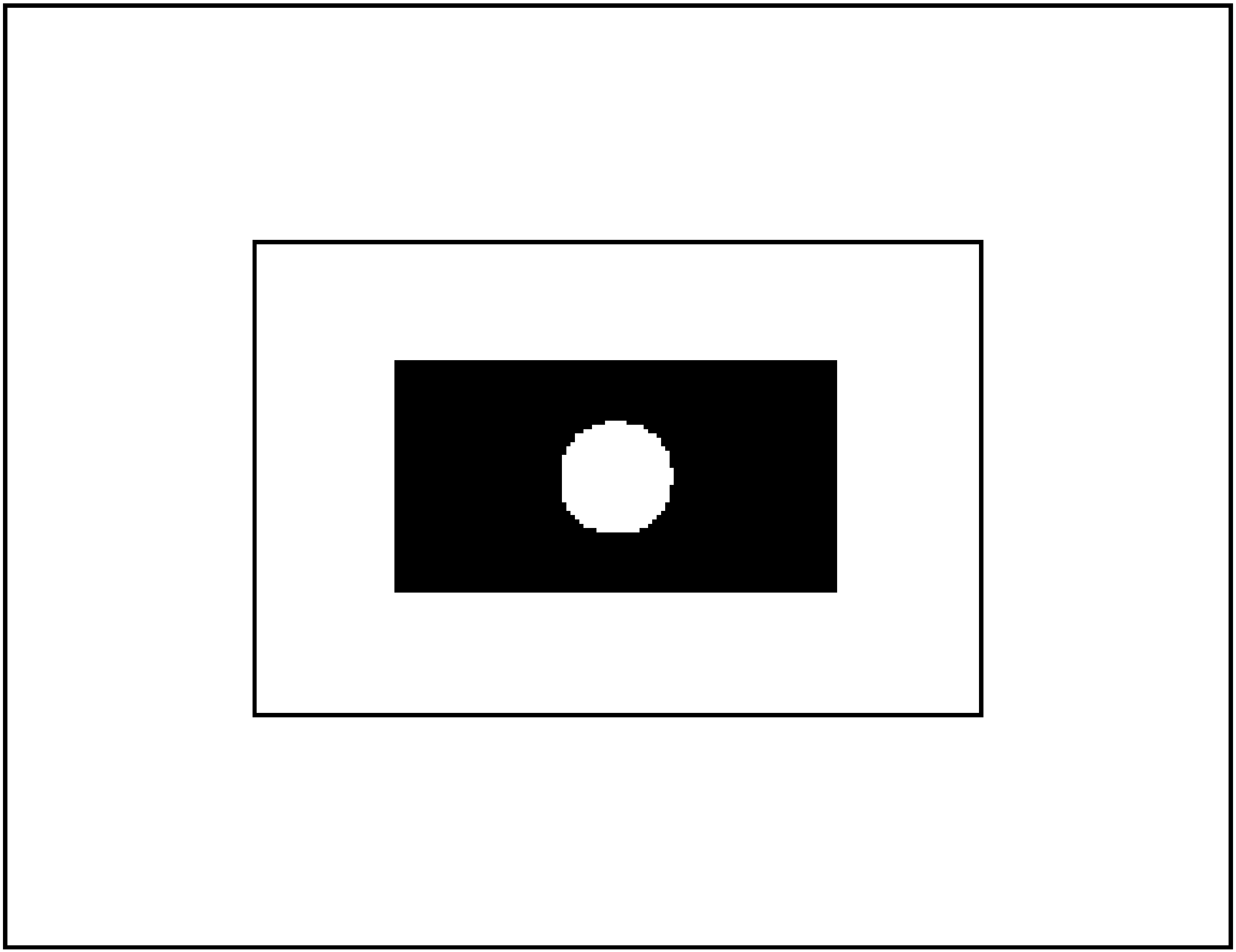
What is Segmentation?

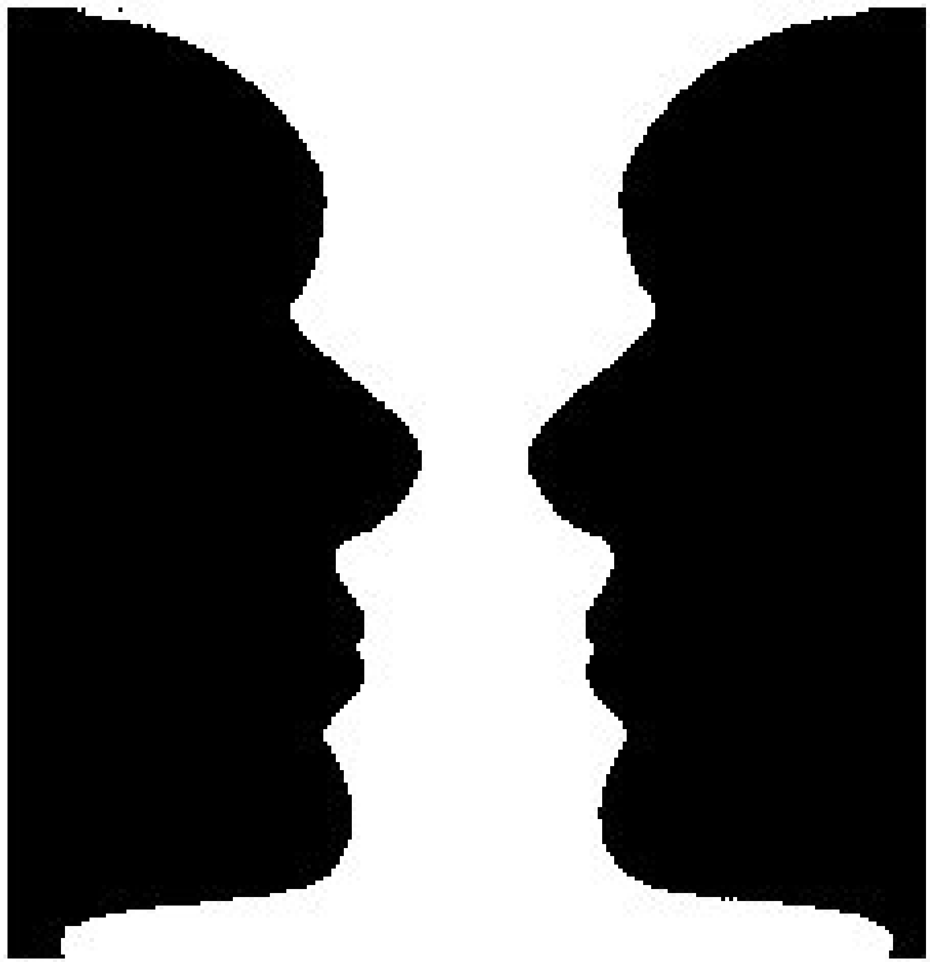
- **Clustering image elements that “belong together”**
  - **Partitioning**
    - Divide into regions/sequences with coherent internal properties
  - **Grouping**
    - Identify sets of coherent tokens in image
- **Tokens:** Whatever we need to group
  - Pixels
  - Features (corners, lines, etc.)
  - Larger regions with uniform colour or texture
  - Discrete objects (e.g., people in a crowd)
  - Etc.

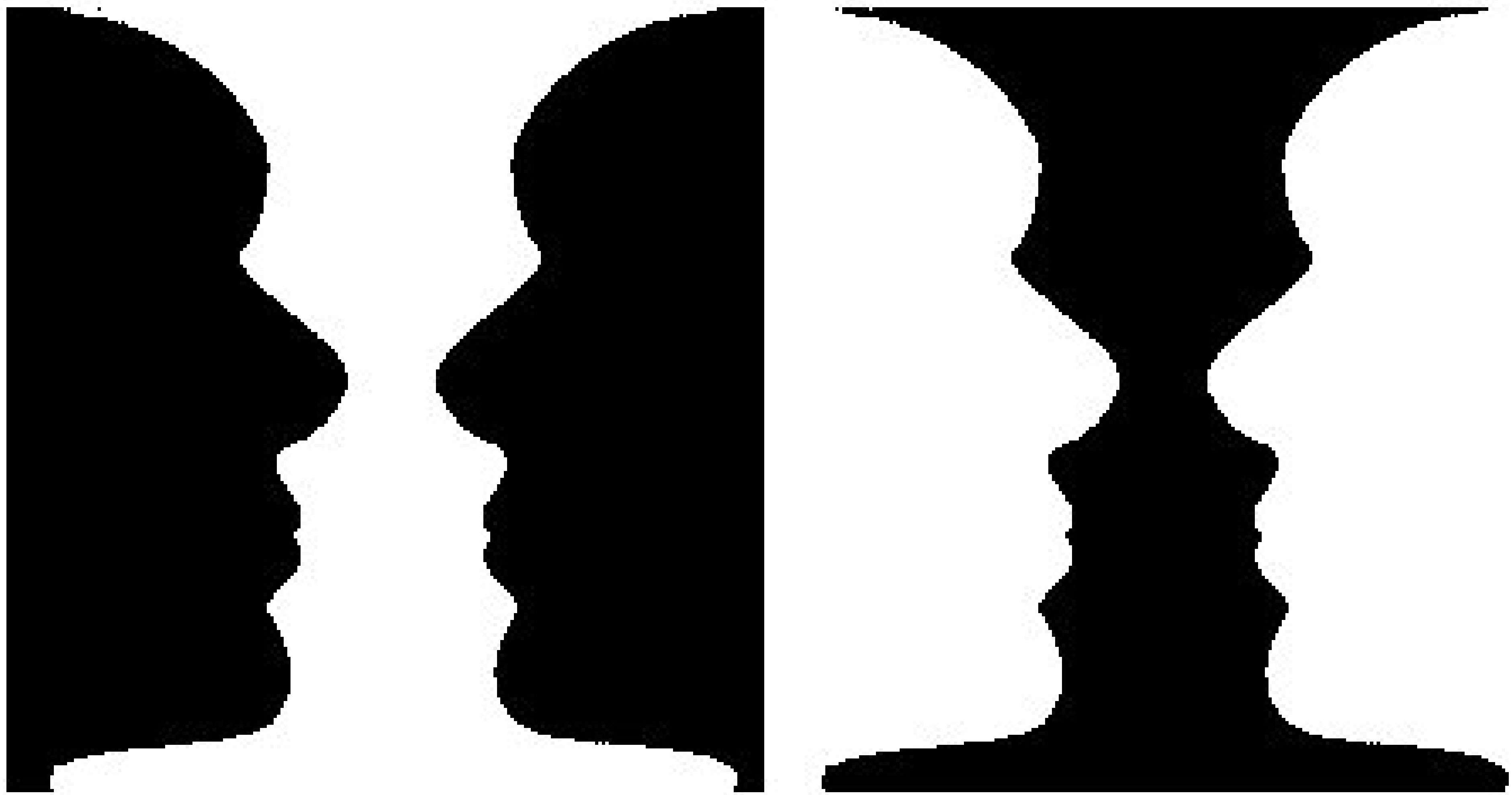
Slide credit: Christopher Rasmussen

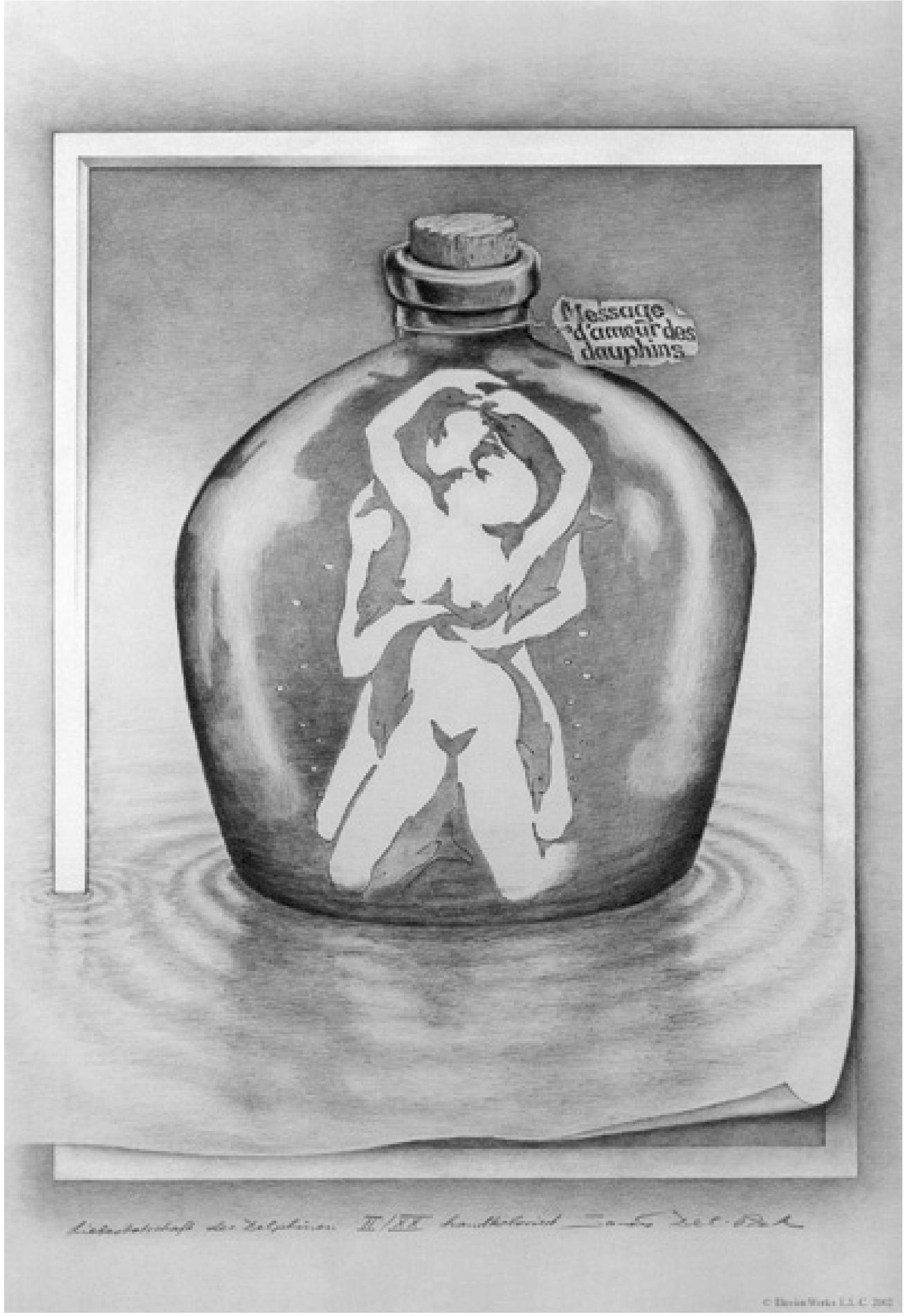


Why do these tokens belong together?



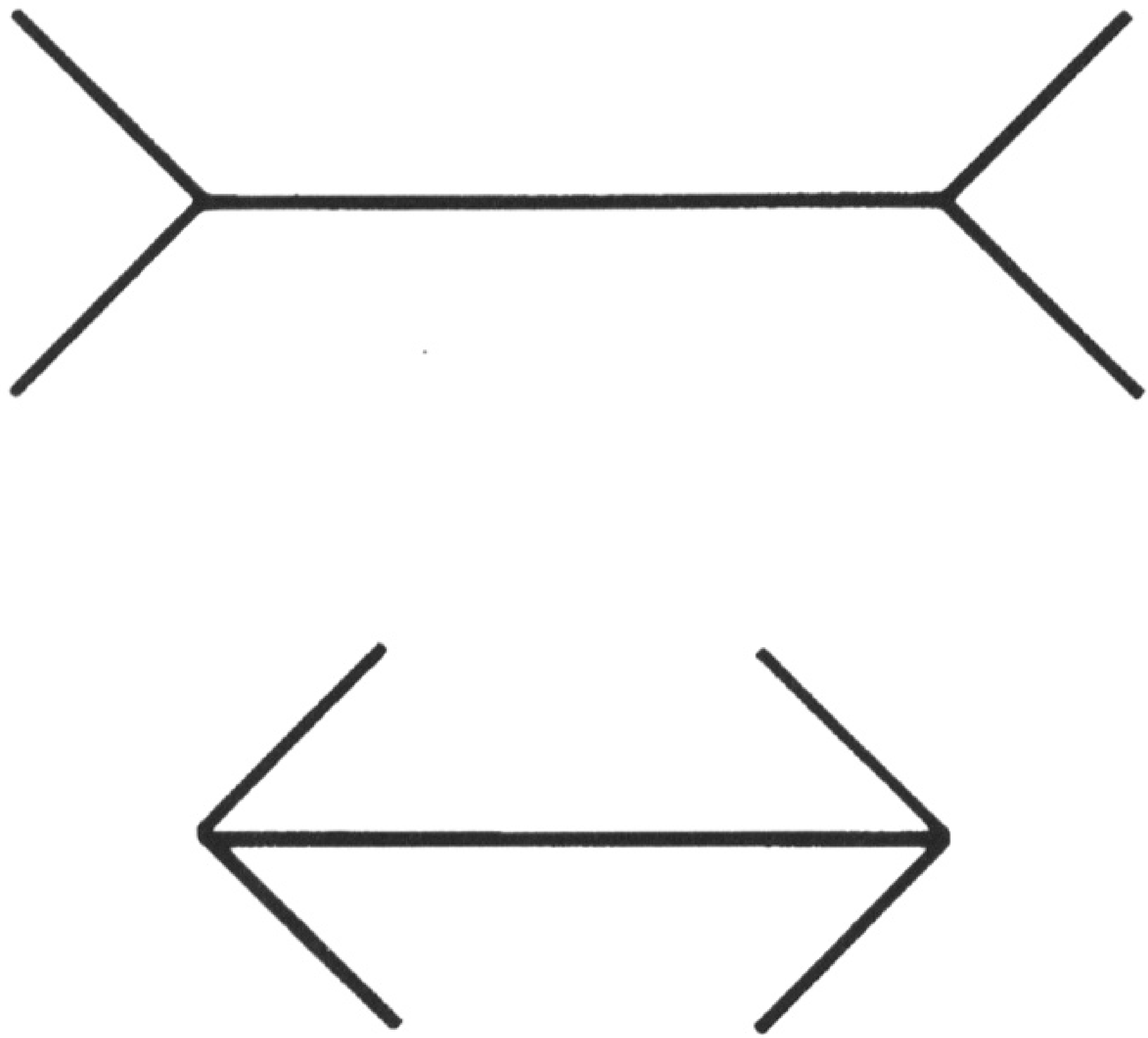


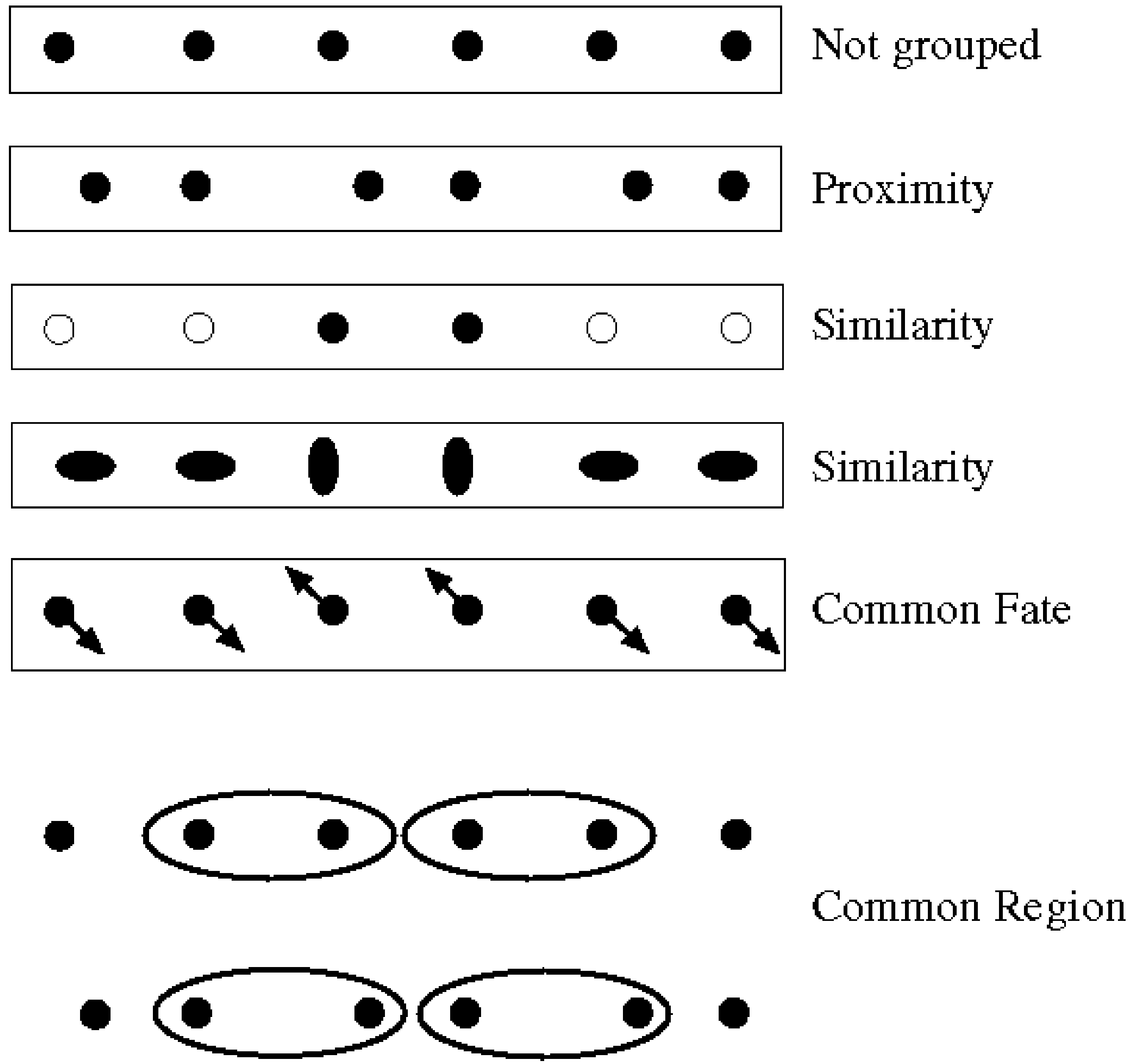


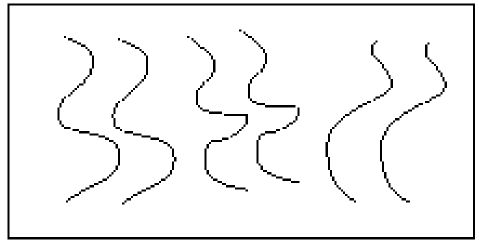

Message d'amour des dauphins

Basic ideas of grouping in human vision

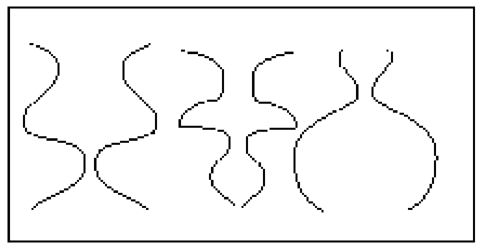
- Figure-ground discrimination
  - grouping can be seen in terms of allocating some elements to a figure, some to ground
  - Can be based on local bottom-up cues or high level recognition
- Gestalt properties
  - elements in a collection of elements can have properties that result from relationships (Muller-Lyer effect)
  - A series of factors affect whether elements should be grouped together
    - Gestalt factors



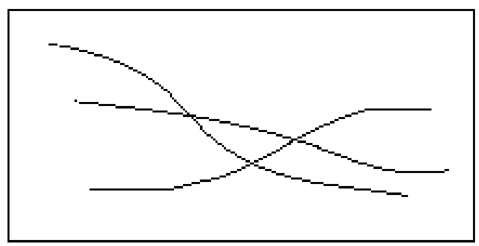

Not grouped
Proximity
Similarity
Similarity
Common Fate
Common Region



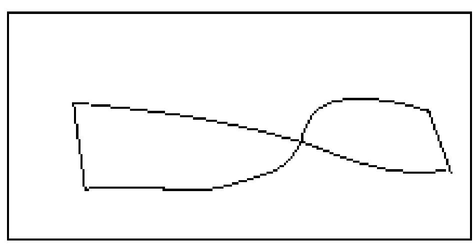
Parallelism



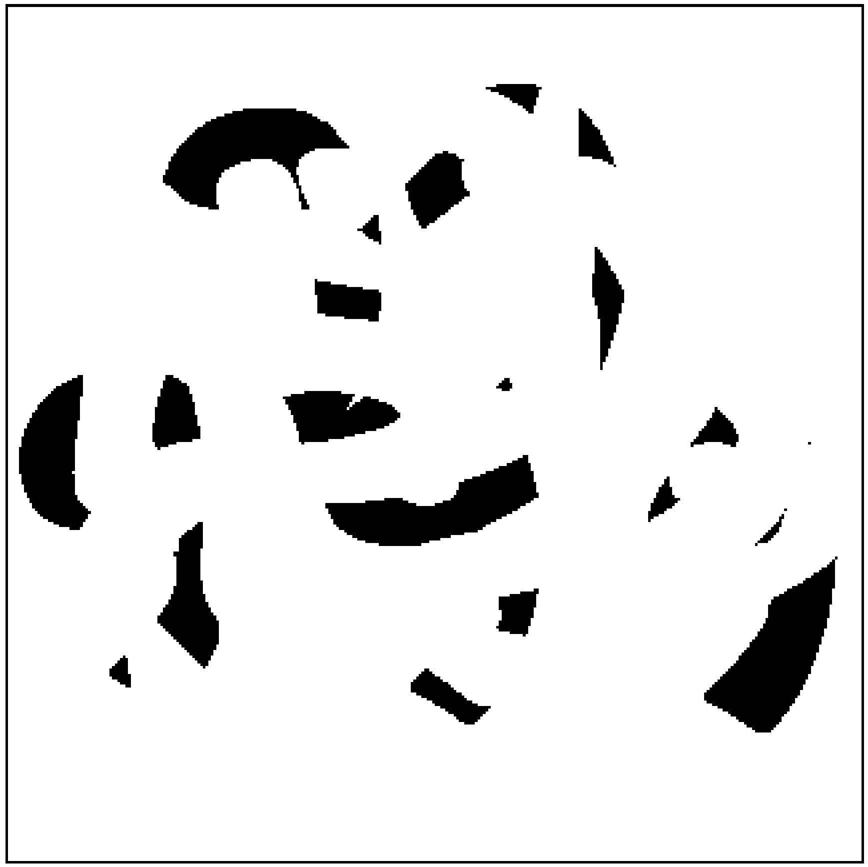
Symmetry

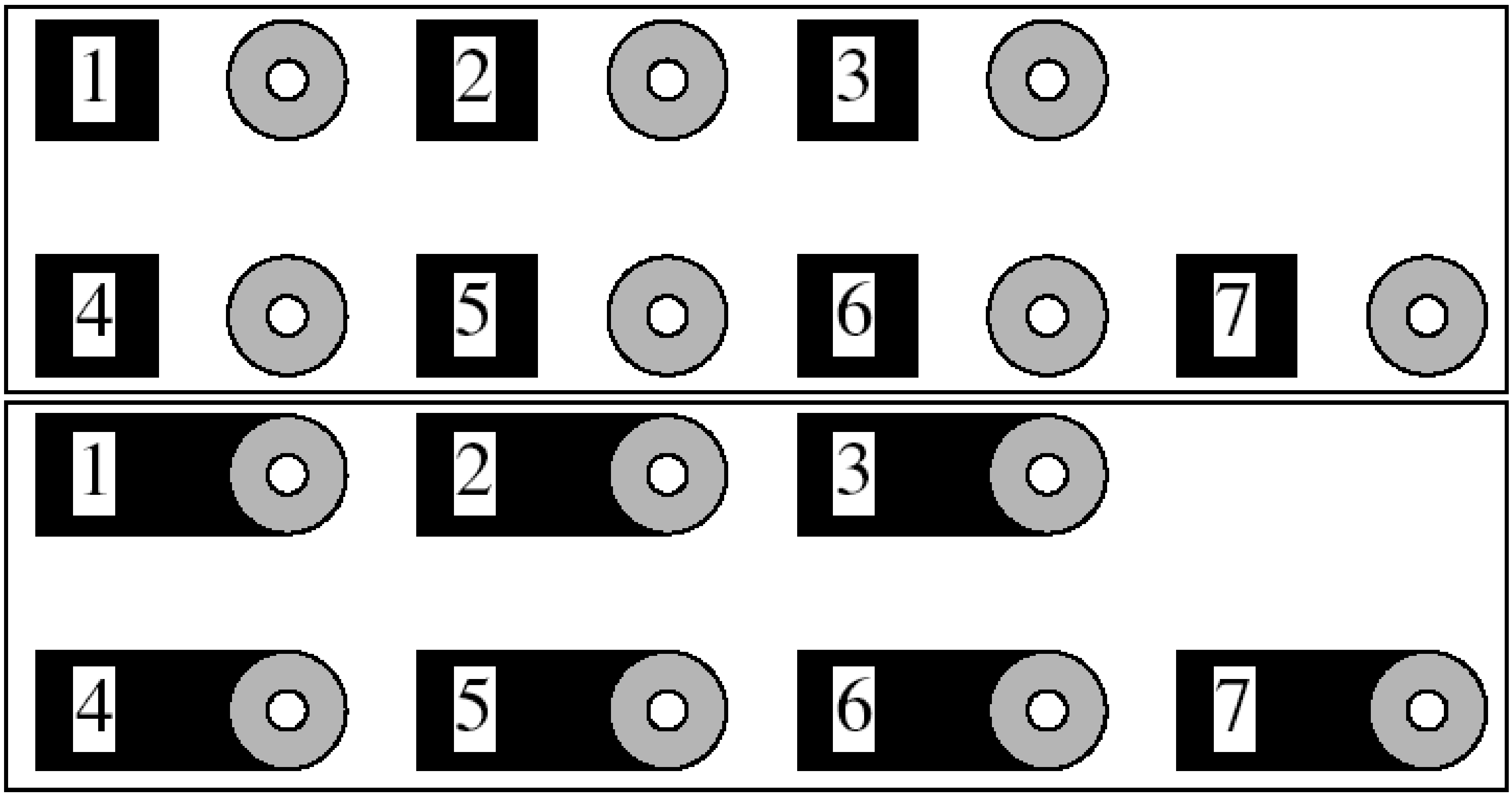


Continuity

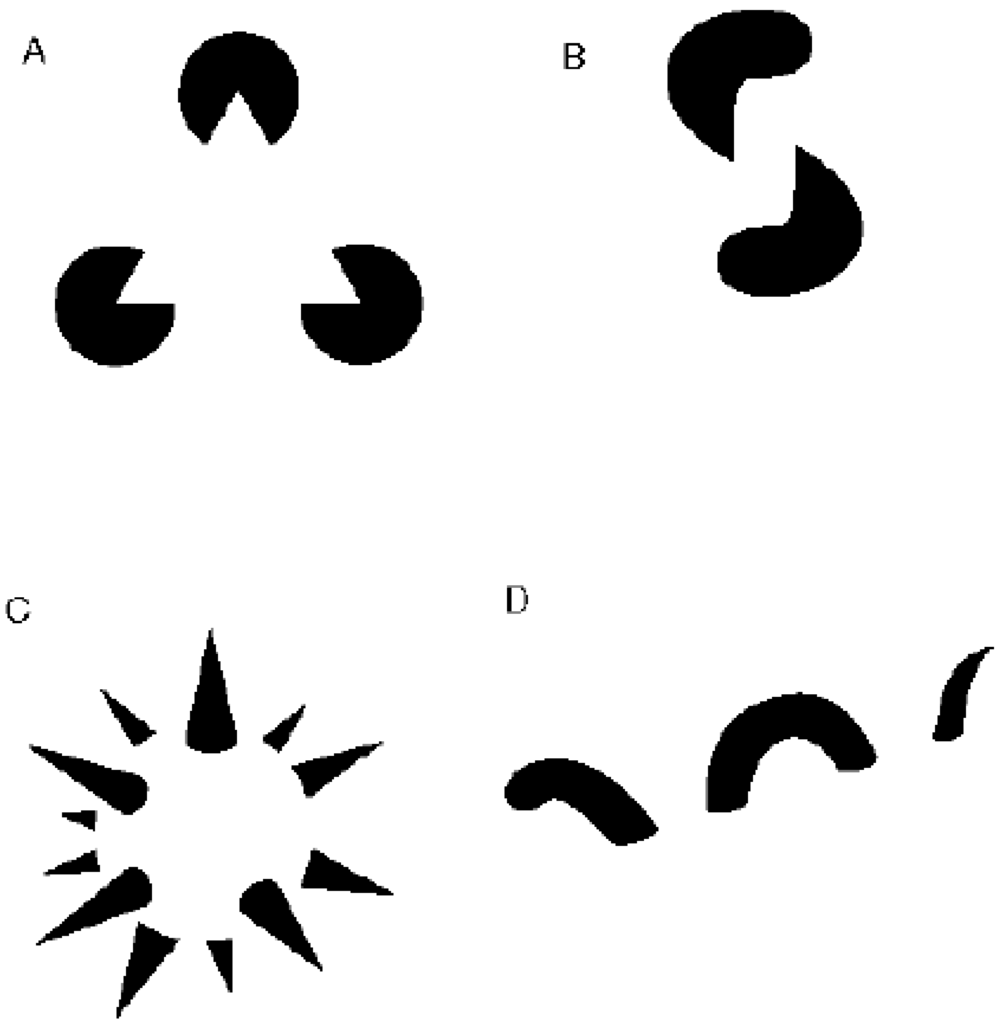


Closure




1
2
3
4
5
6
7
1
2
3
4
5
6
7

Groupings by Invisible Completions



* Images from Steve Lehar's Gestalt papers: http://cns-alumni.bu.edu/pub/slehar/Lehar.html

Application: Background Subtraction

- The problem: Segment moving foreground objects from static background

from C. Stauffer and W. Grimson

Current image

Background image

Foreground pixels

- Applications
  - Traffic monitoring
  - Surveillance/security
  - User interaction

courtesy of C. Wren

Pfinder

Slide credit: Christopher Rasmussen

Technique: Background Subtraction

- If we know what the background looks like, it is easy to identify “interesting bits”
- Applications
  - Person in an office
  - Tracking cars on a road
  - surveillance
- Approach:
  - use a moving average to estimate background image
  - subtract from current frame
  - large absolute values are interesting pixels
    - trick: use morphological operations to clean up pixels

Algorithm

video sequence $I(\mathbf{x}, t)$ background $I_0(\mathbf{x}, t)$

frame difference $d(\mathbf{x}, t)$ thresholded frame diff $d_T(\mathbf{x}, t)$

for t = 1:N

Update background model $I_0(\mathbf{x}, t)$

Compute frame difference $d(\mathbf{x}, t) = |I(\mathbf{x}, t) - I_0(\mathbf{x}, t)|$

Threshold frame difference $d_T(\mathbf{x}, t) = d(\mathbf{x}, t) > thresh$

Noise removal $d_T(\mathbf{x}, t) = imerode(d_T(\mathbf{x}, t))$

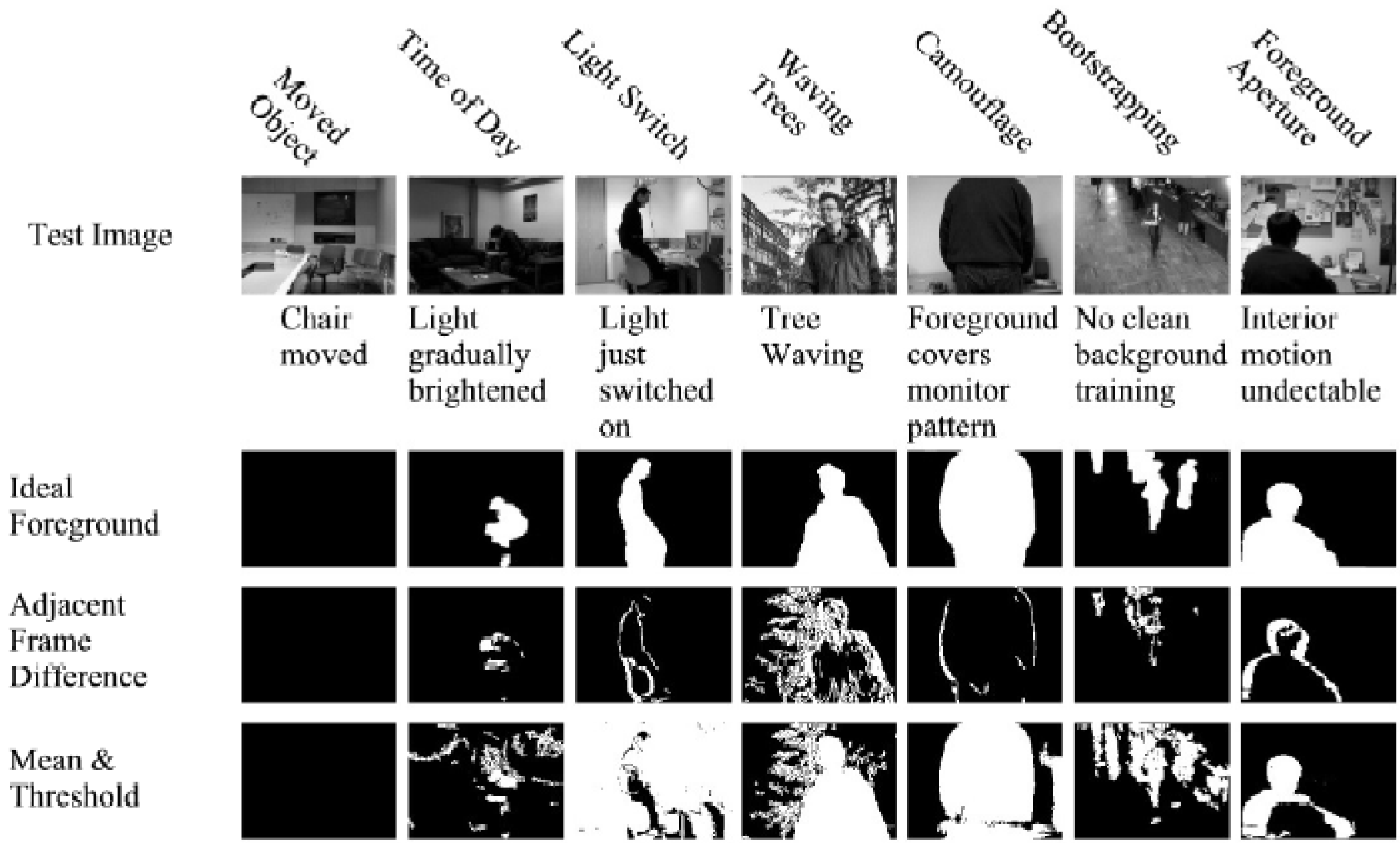
end

Objects are detected where $d_T(\mathbf{x}, t)$ is non-zero

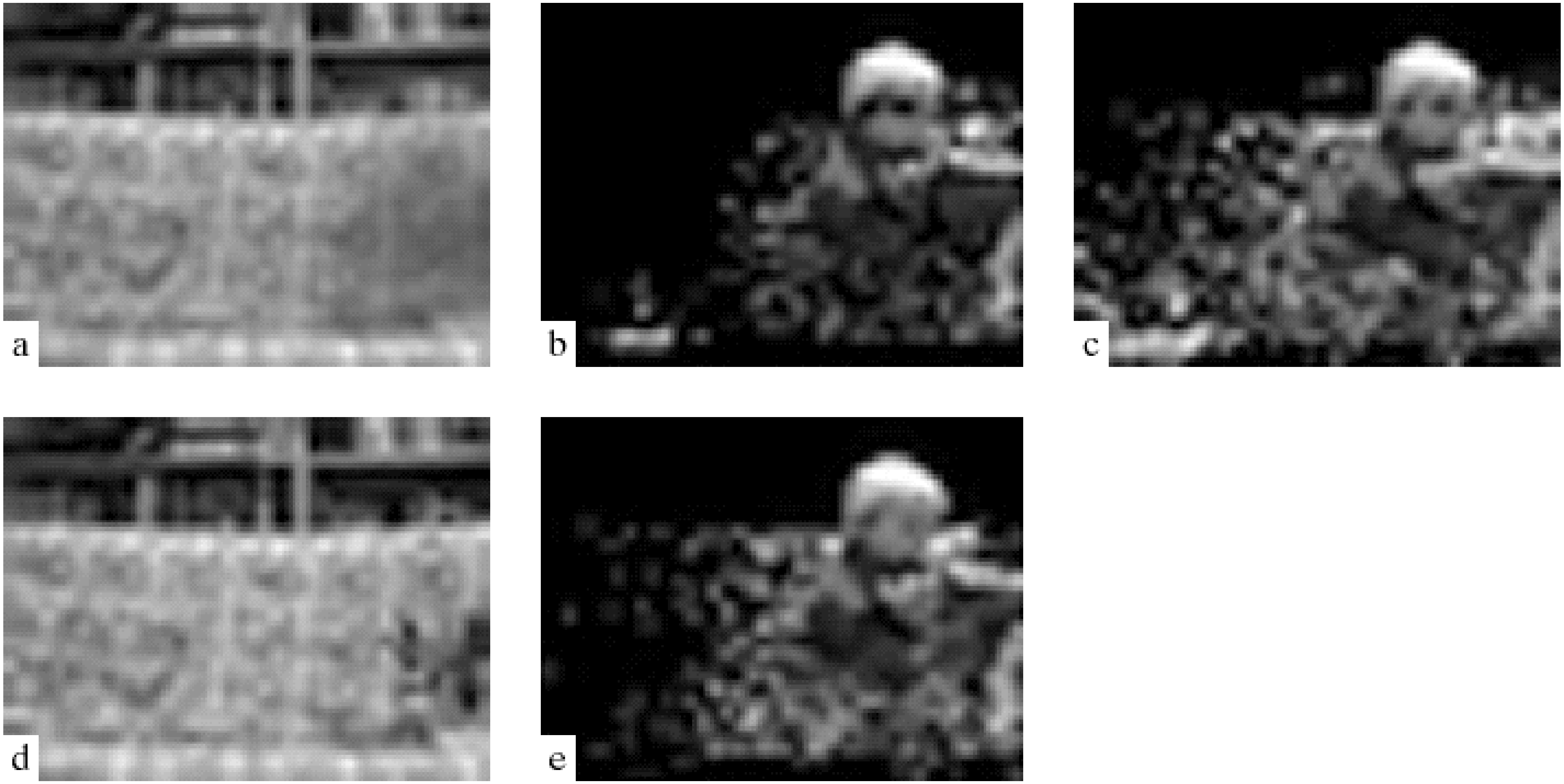
Background Modelling

- **Offline average** $$I_0(\mathbf{x},t) = \frac{1}{T}\sum_{t=1}^{T} I(\mathbf{x},t)$$
  - Pixel-wise mean values are computed during training phase (also called Mean and Threshold)
- **Adjacent Frame Difference** $$I_0(\mathbf{x},t) = I(\mathbf{x},t-1)$$
  - Each image is subtracted from previous image in sequence
- **Moving average** $$I_0(\mathbf{x},t) = \frac{w_a I(\mathbf{x},t) + \sum_{i=1}^{N} w_i I(\mathbf{x},t-i)}{w_c}$$
  - Background model is linear weighted sum of previous frames
- **Multi-Modal** $$p(I_0(\mathbf{x},t)) = \sum_{i=1}^{n_c} \pi_i N(\mathbf{x};\mathbf{m}_{\mathbf{x}},\sigma_{\mathbf{x}}^2)$$
  - Background model is Gaussian mixture model learnt from training data

Results & Problems for Simple Approaches



from K. Toyama et al.


a
b
c
d
e

Background Subtraction: Issues

- Noise models
  - **Unimodal**: Pixel values vary over time even for static scenes
  - **Multimodal**: Features in background can "oscillate", requiring models which can represent disjoint sets of pixel values (e.g., waving trees against sky)
- Gross illumination changes
  - **Continuous:** Gradual illumination changes alter the appearance of the background (e.g., time of day)
  - **Discontinuous:** Sudden changes in illumination and other scene parameters alter the appearance of the background (e.g., flipping a light switch)
- Bootstrapping
  - Is a training phase with "no foreground" necessary, or can the system learn what's static vs. dynamic online?

Slide credit: Christopher Rasmussen

Technique: Shot Boundary Detection

- Find the shots in a sequence of video
  - shot boundaries usually result in big differences between succeeding frames
- Strategy:
  - compute interframe distances
  - declare a boundary where these are big
- Possible distances
  - frame differences
  - histogram differences
  - block comparisons
  - edge differences
- Applications:
  - representation for movies, or video sequences
    - find shot boundaries
    - obtain “most representative” frame
  - supports search

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

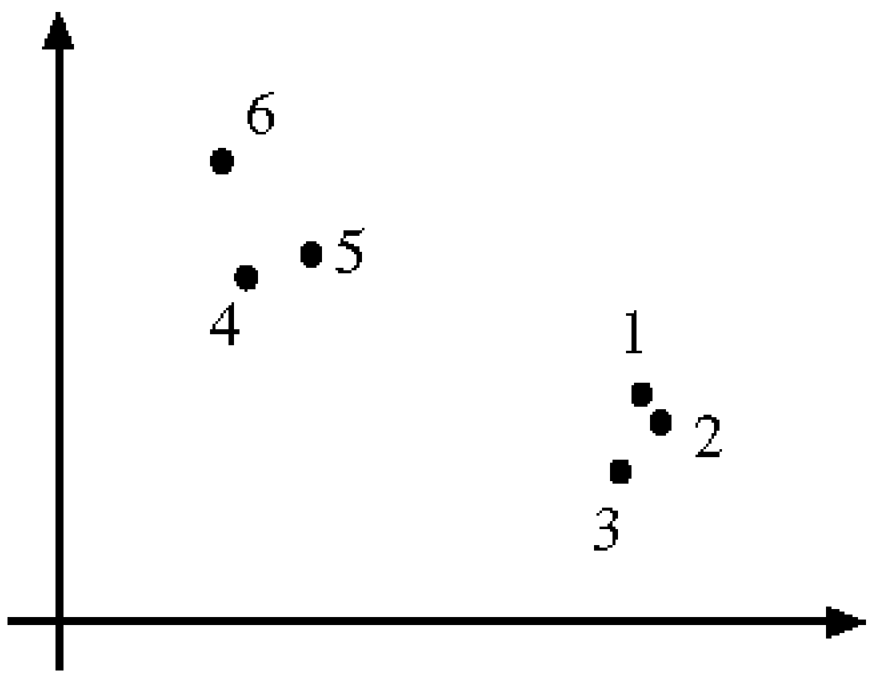
Feature Space

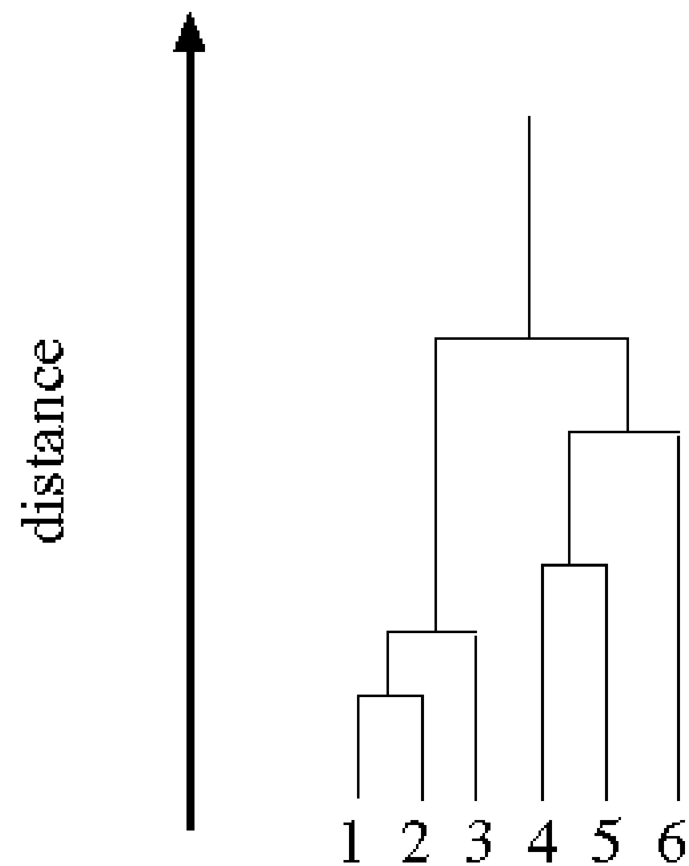
- Every token is identified by a set of salient visual characteristics called *features*. For example:
  - Position
  - Color
  - Texture
  - Motion vector
  - Size, orientation (if token is larger than a pixel)

- The choice of features and how they are quantified implies a *feature space* in which each token is represented by a point

- Token similarity is thus measured by distance between points ("feature vectors") in feature space

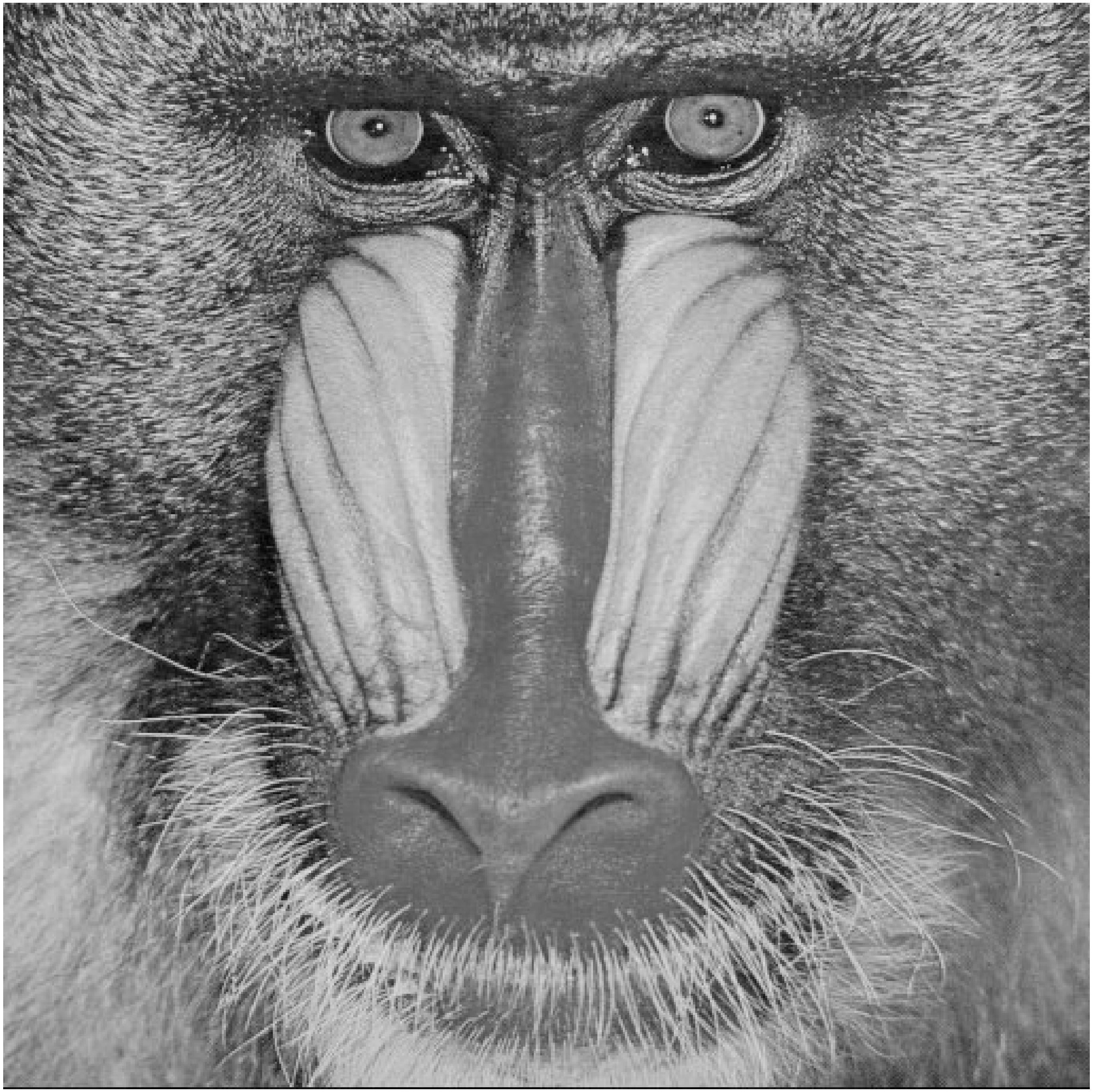
Slide credit: Christopher Rasmussen

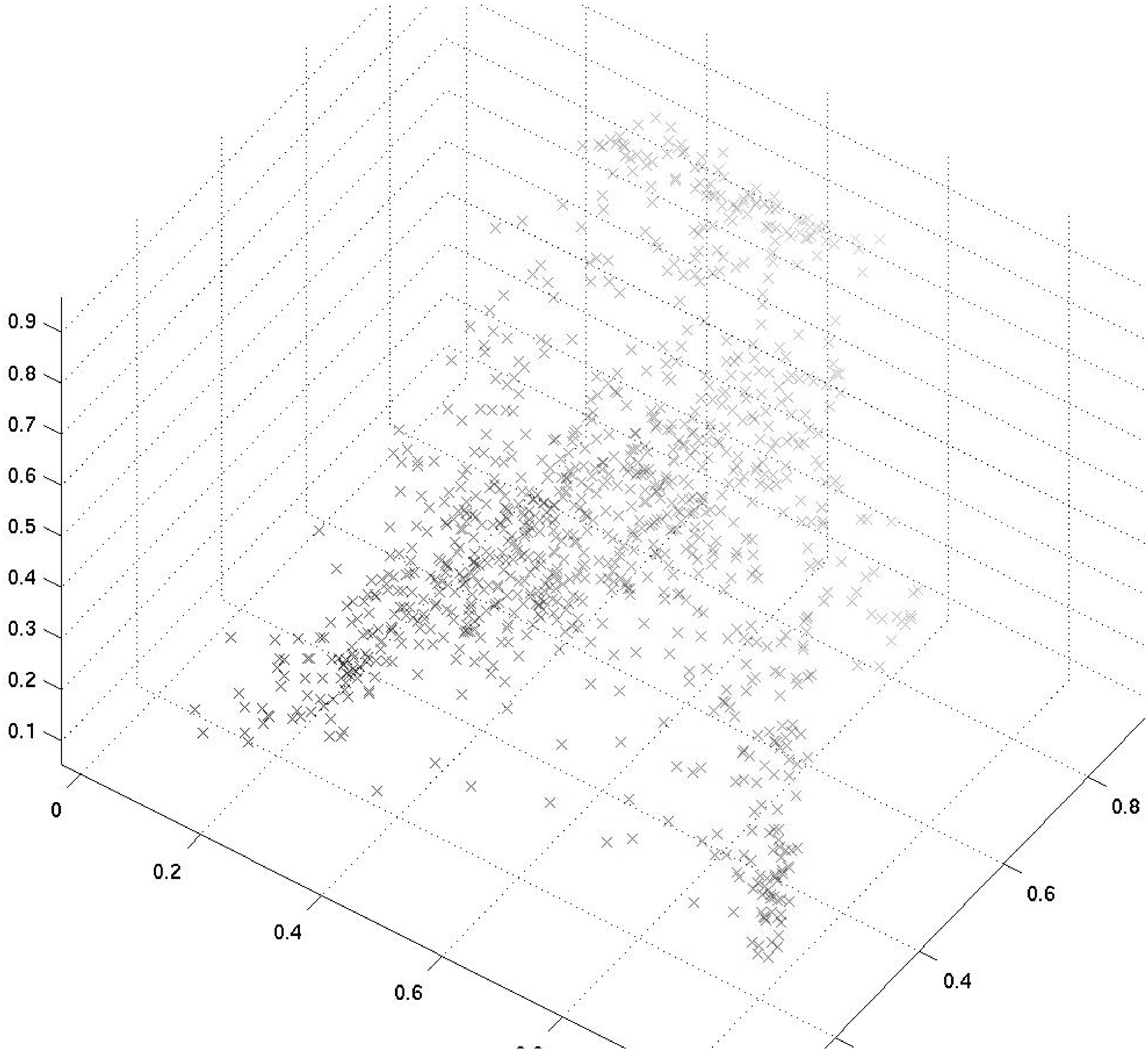

6
5
4
1
2
3

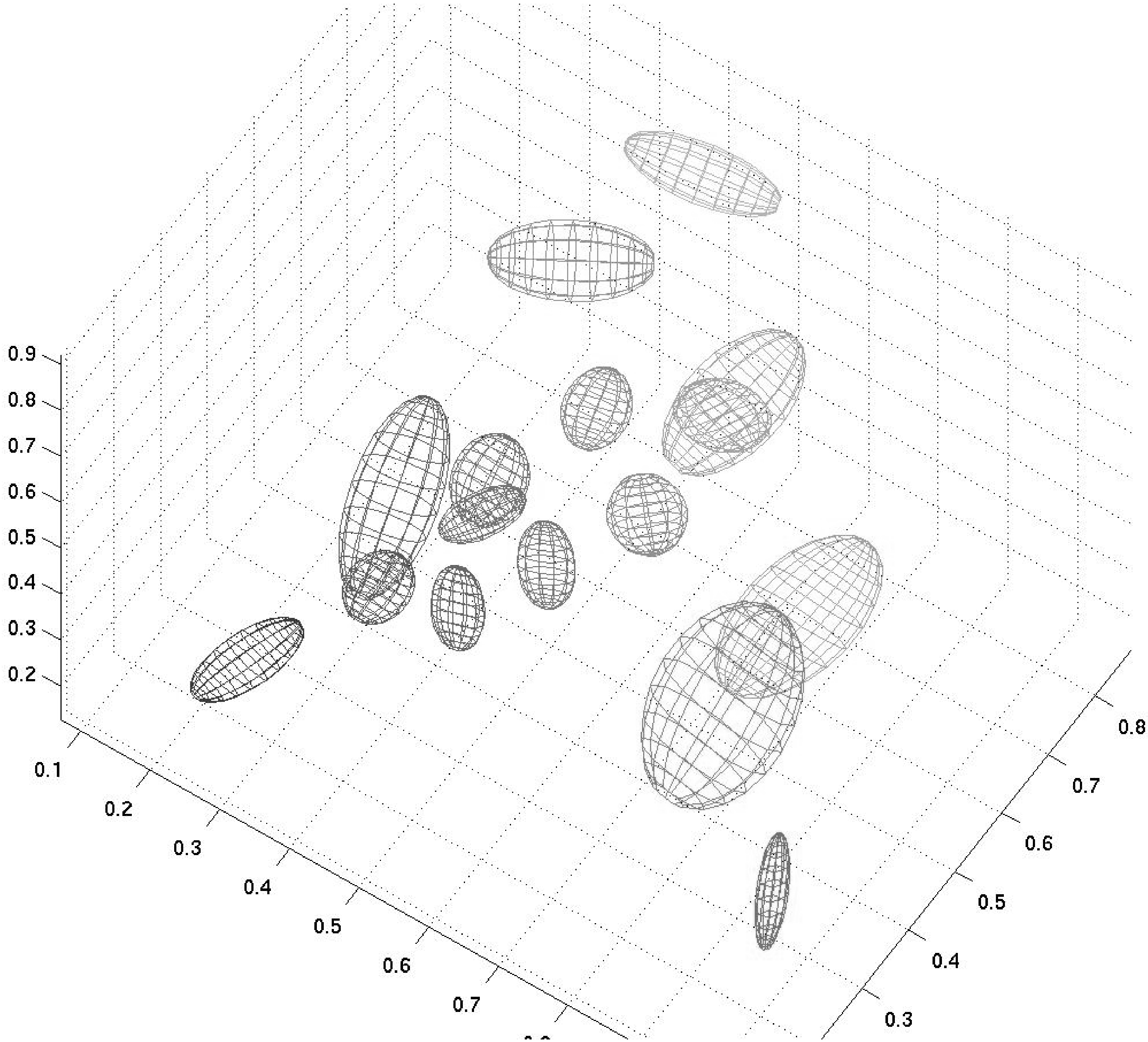

distance
1 2 3 4 5 6

Matlab Code

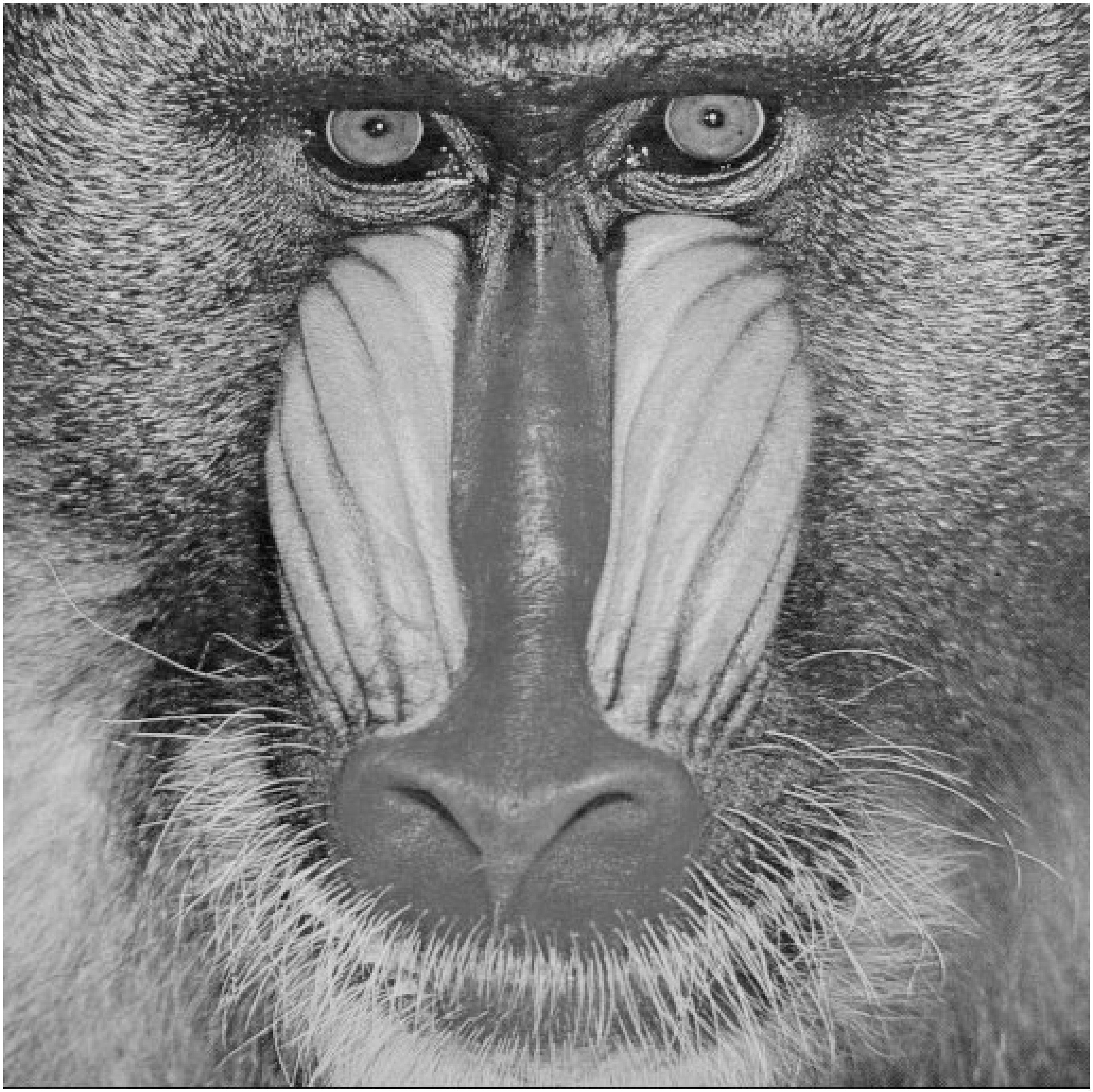
- rand('seed',12);
- X = rand(100,2);
- Y = pdist(X,‘euclidean');
- Z = linkage(Y,‘single');
- [H, T] = dendrogram(Z);

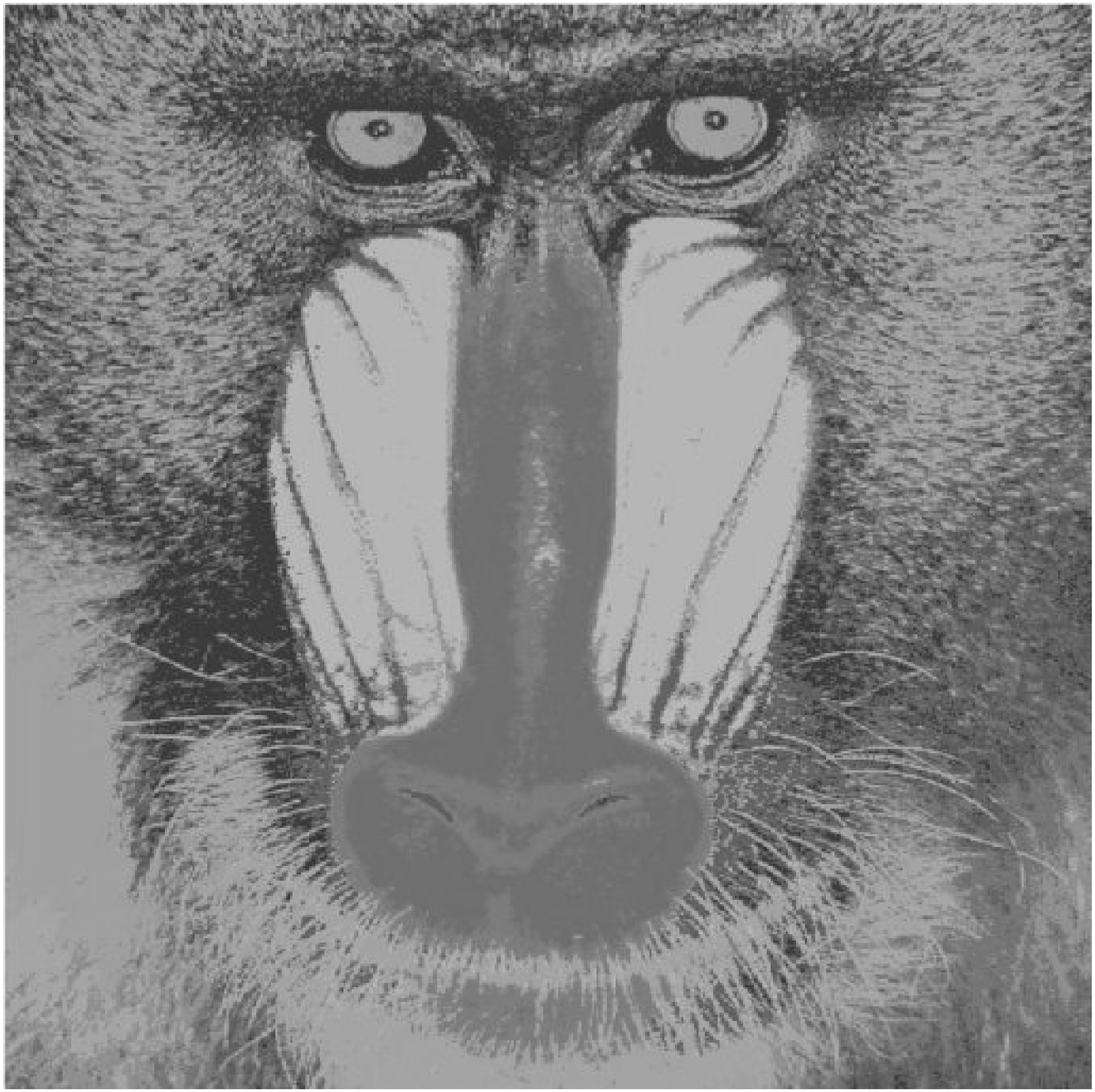


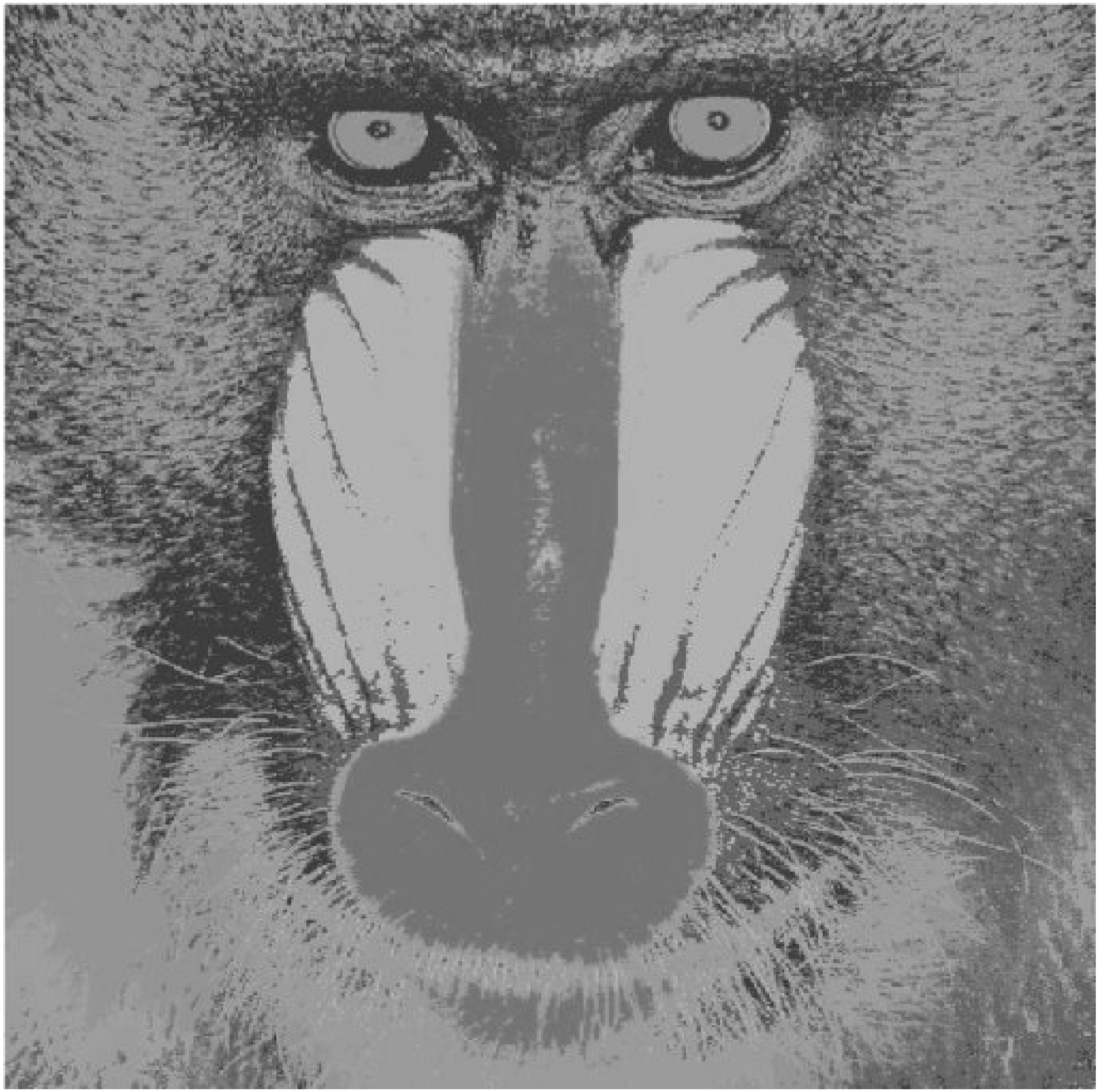

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.2
0.4
0.6
0.8
0.6
0.4

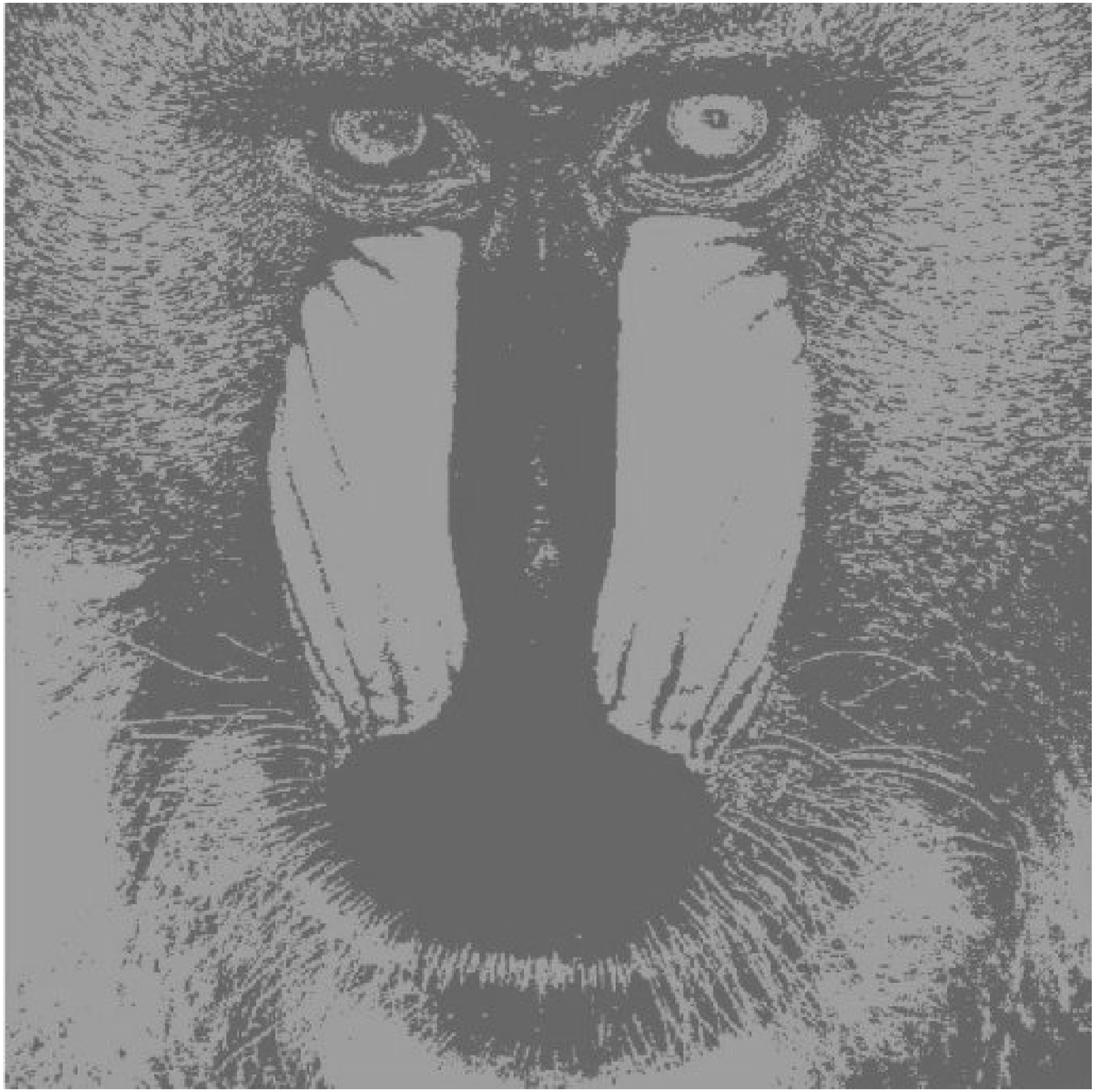


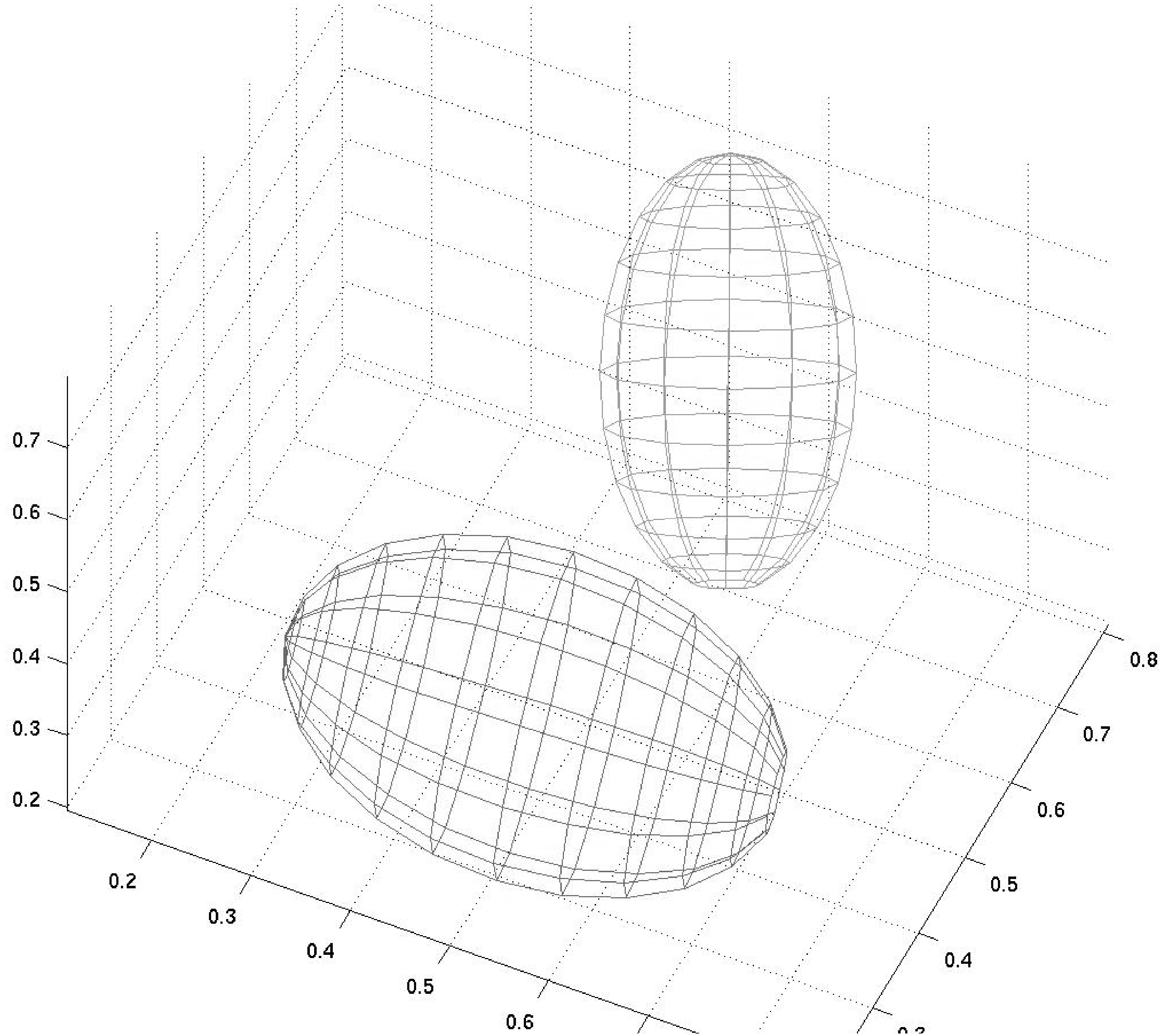
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.7
0.6
0.5
0.4
0.3










0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.8
0.7
0.6
0.5
0.4

K-Means Clustering

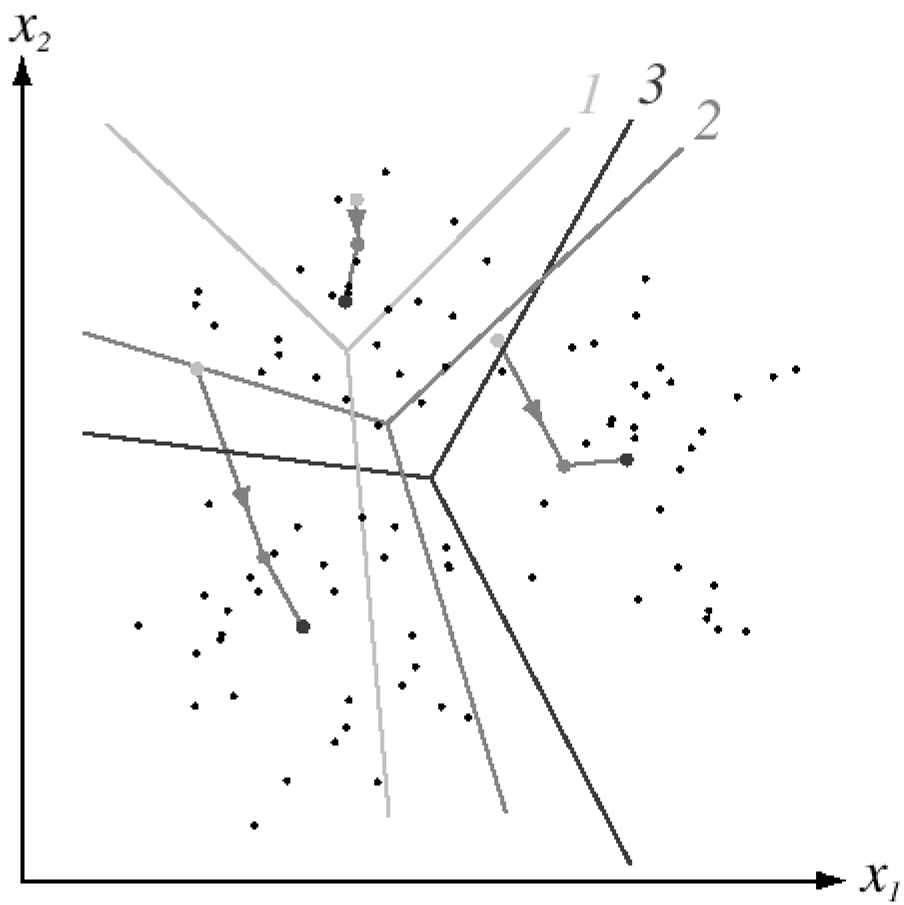
- Initialization: Given K categories, N points in feature space. Pick K points randomly; these are initial cluster centers (means) $m_1, \ldots, m_K$. Repeat the following:
  1. Assign each of the N points, $\boldsymbol{x}_j$, to clusters by nearest m_i (make sure no cluster is empty)
  2. Recompute mean m_i of each cluster from its member points
  3. If no mean has changed more than some ε, stop

- Effectively carries out gradient descent to minimize:

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

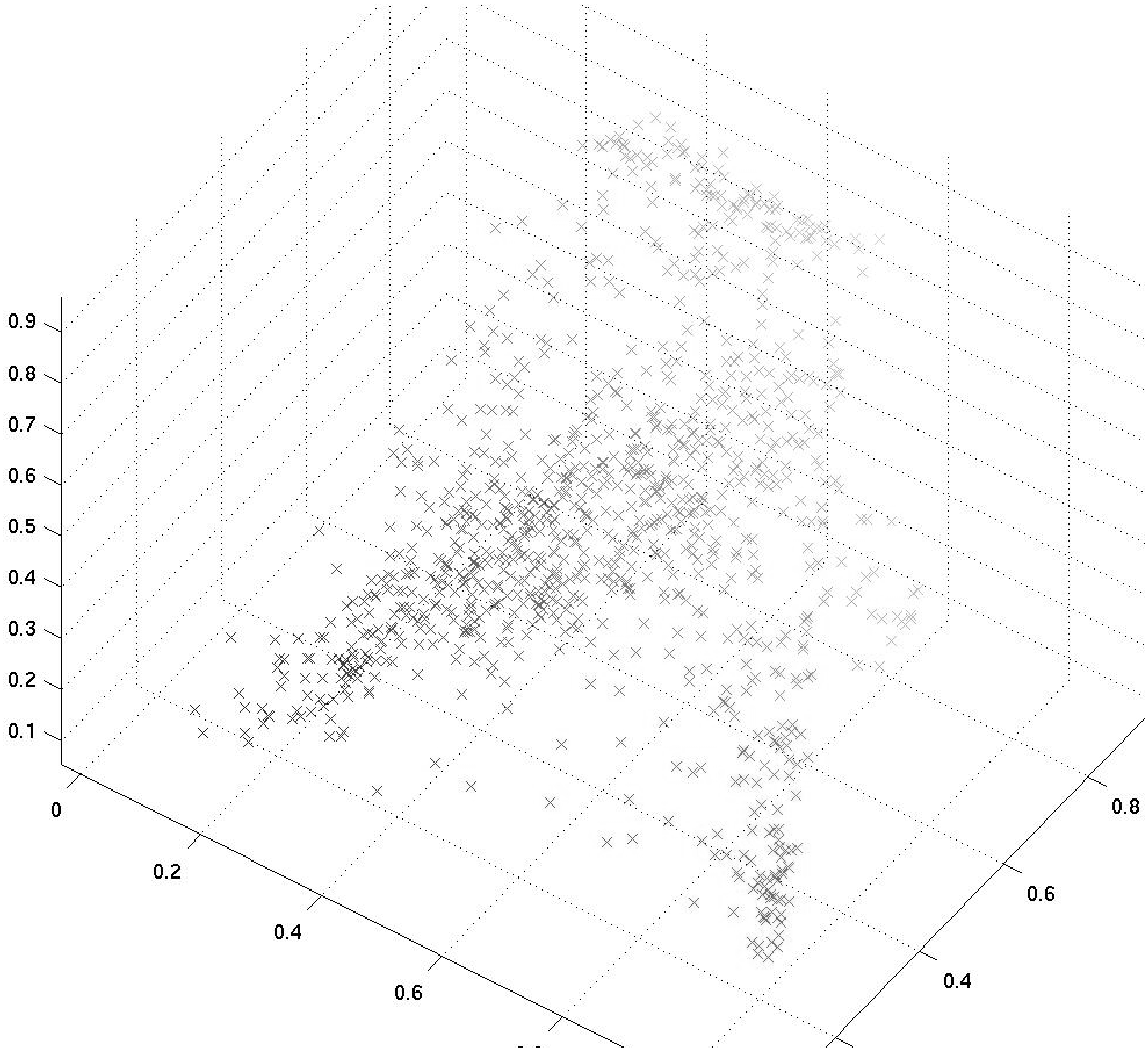
Slide credit: Christopher Rasmussen

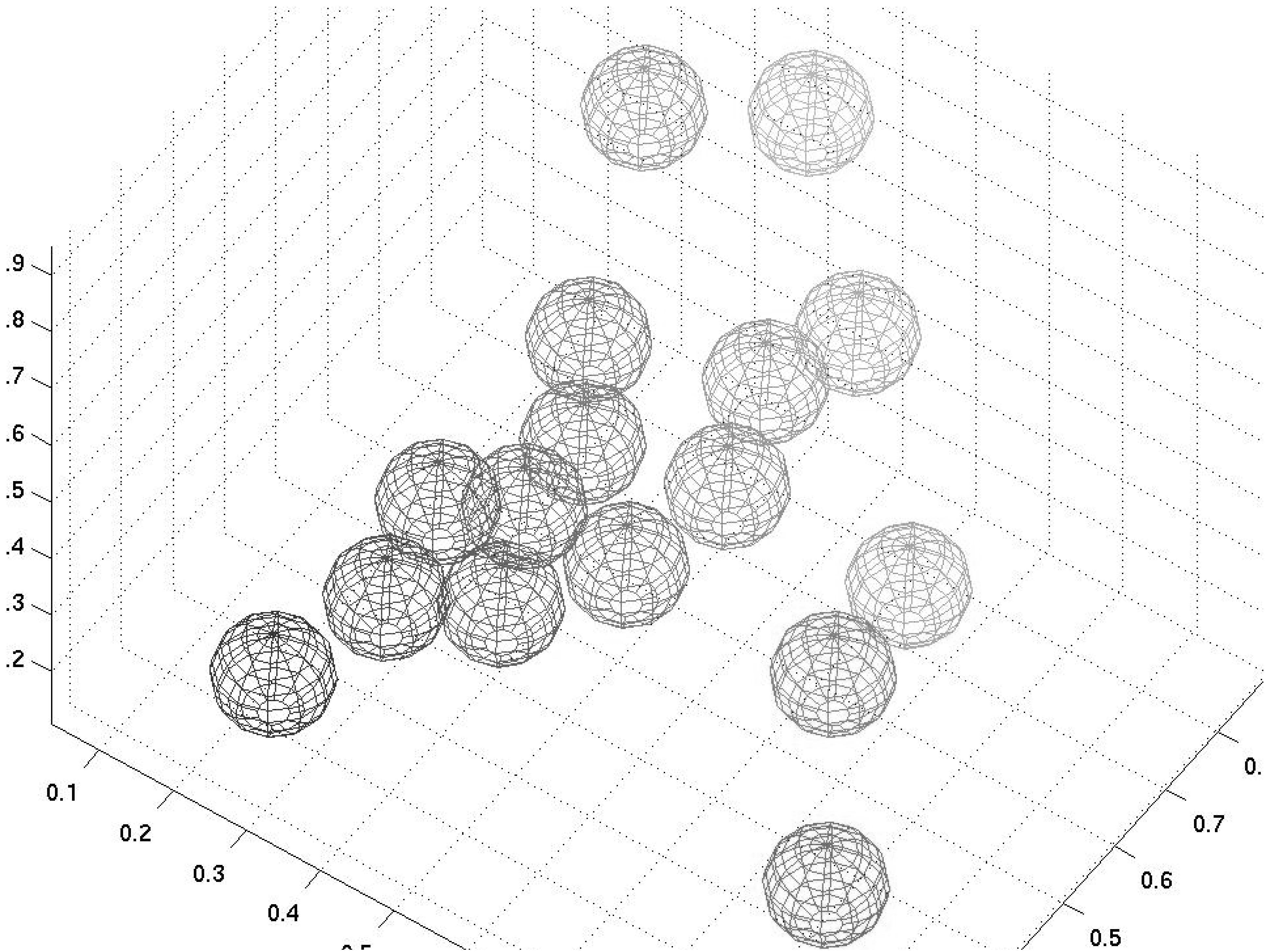
Example: 3-means Clustering



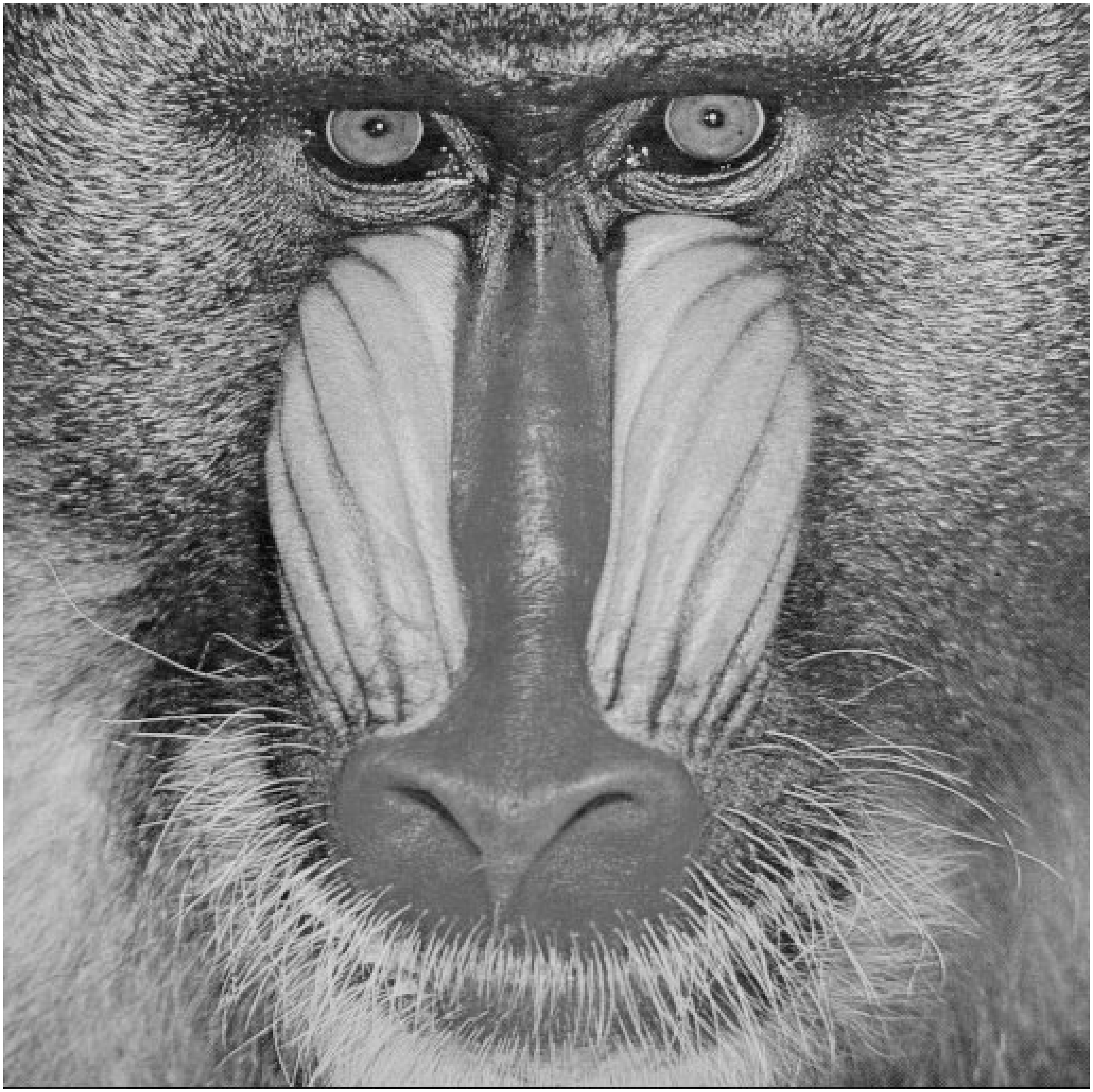
from
Duda et al.

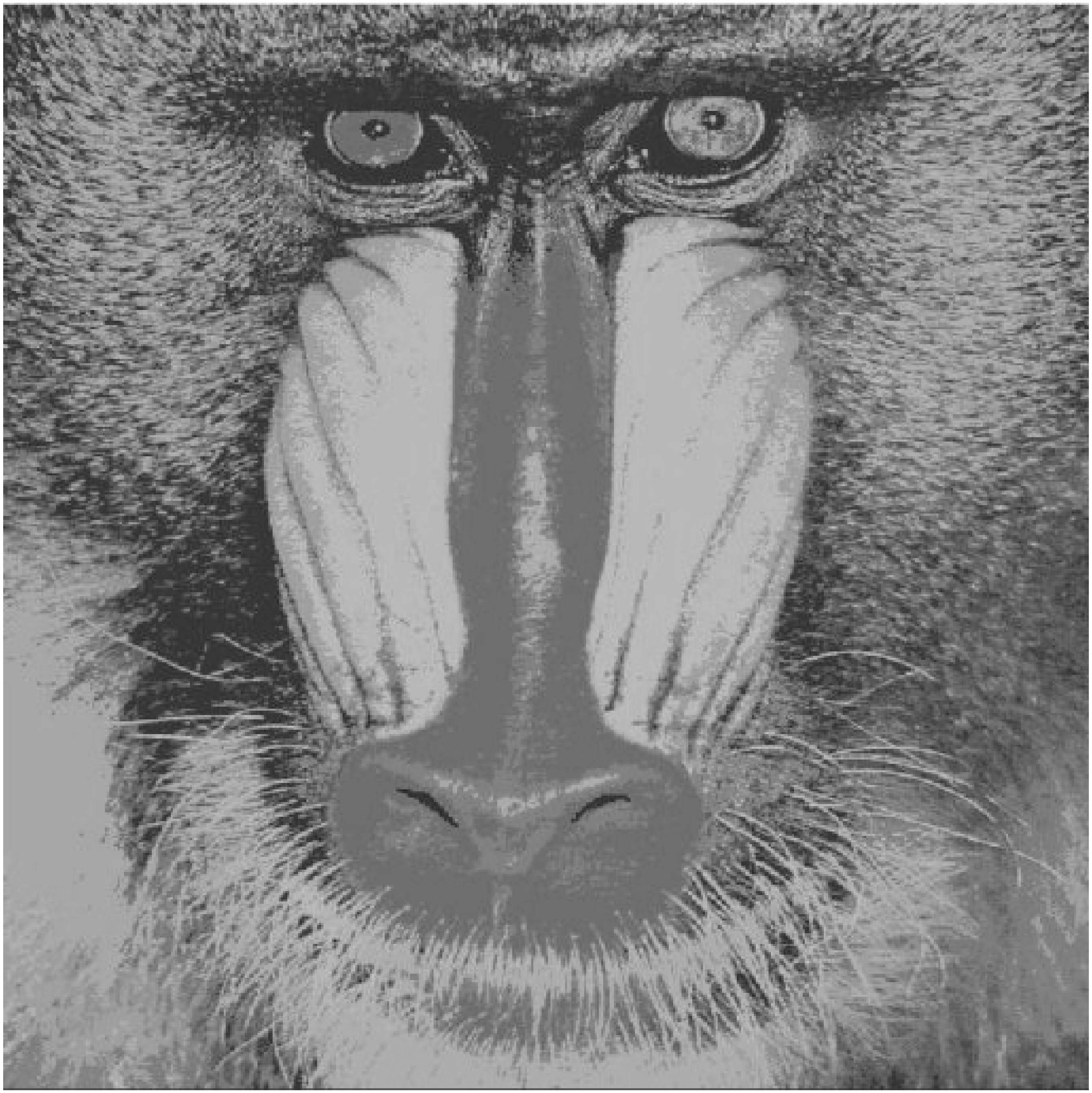
Convergence in 3 steps

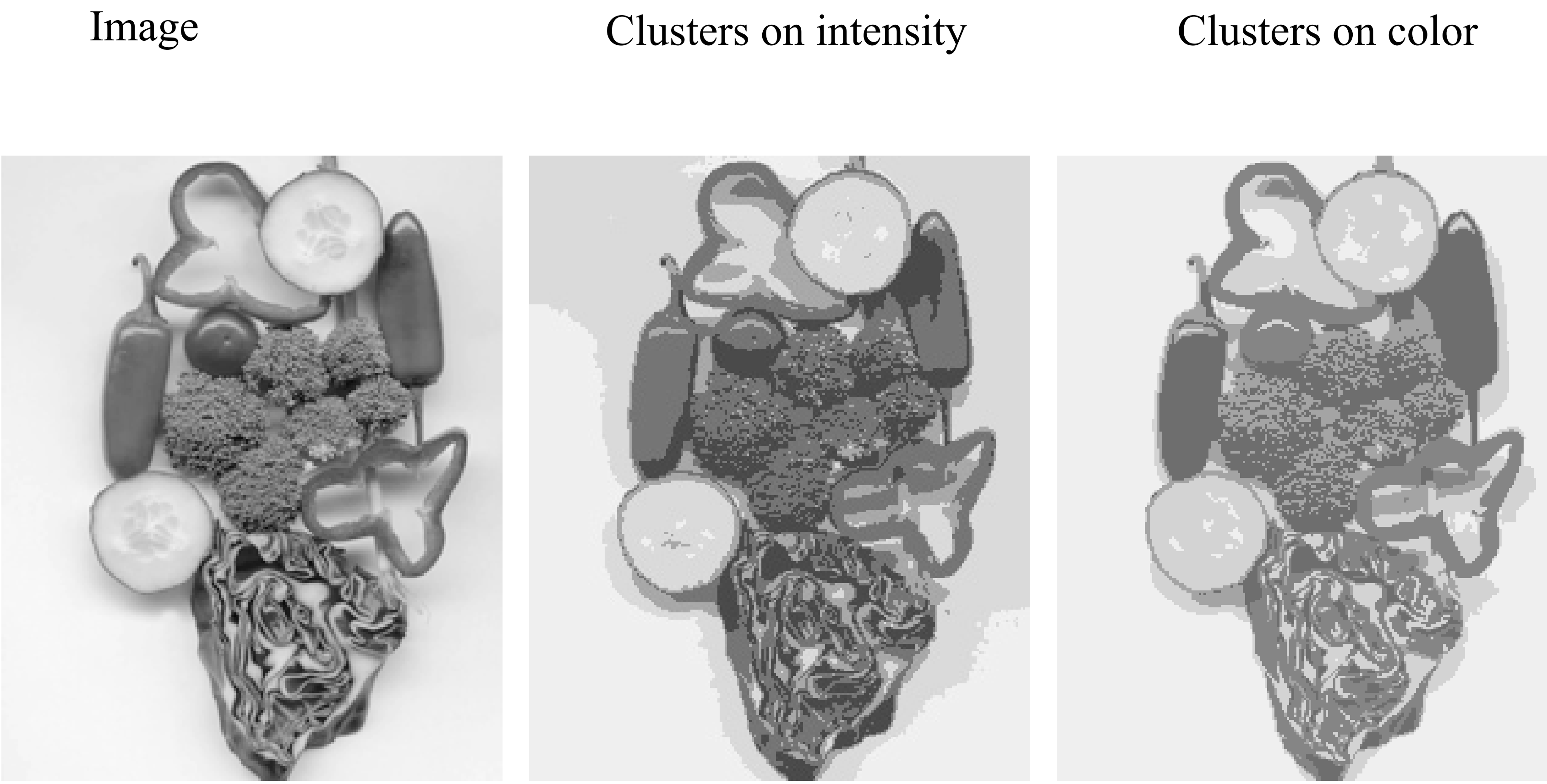

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.2
0.4
0.6
0.8
0.6
0.4



.9
.8
.7
.6
.5
.4
.3
.2
0.1
0.2
0.3
0.4
0.
0.7
0.6
0.5







K-means clustering using intensity alone and color alone

K-Means

$$e(\mathbf{m}_i) = \sum_{i=1}^{n_c} \sum_{j;c_j=i} |\mathbf{x}_j - \mathbf{m}_i|^2$$

$$\frac{\partial e}{\partial \mathbf{m}_k} = \sum_{j;c_j=k} -2(\mathbf{x}_j - \mathbf{m}_k) = 0$$

$$\mathbf{m}_k = \frac{\sum_{j;c_j=k} \mathbf{x}_j}{\sum_{j;c_j=k} 1} = \frac{1}{n_k} \sum_{j;c_j=k} \mathbf{x}_j$$

EM

$$p(\mathbf{x}|\mathbf{m}) = \prod_{j=1}^{n_d} \sum_{i=1}^{n_c} p(\mathbf{x}_j|c_j = i)p(c_j = i)$$

assume uniform priors, and gaussian likelihood

$$p(c_j = i) = 1/n_c, p(\mathbf{x}_j|c_j = i) \propto \exp^{-\frac{1}{2}(\mathbf{x}_j - \mathbf{m}_i)^2}$$

$$e(\mathbf{m}) = \ln p(\mathbf{x}|\mathbf{m}) = \sum_{j=1}^{n_d} \ln \sum_{i=1}^{n_c} \exp^{-\frac{1}{2}(\mathbf{x}_j - \mathbf{m}_i)^2}$$

for $\frac{\partial e}{\partial \mathbf{m}} = 0$, we have

$$\mathbf{m}_k = \frac{\sum_j p(c_j = k|\mathbf{x}_j)\mathbf{x}_j}{\sum_j p(c_j = k|\mathbf{x}_j)} = \frac{\sum_j \xi_{kj}\mathbf{x}_j}{\sum_j \xi_{kj}}$$

Compare K-means VS EM

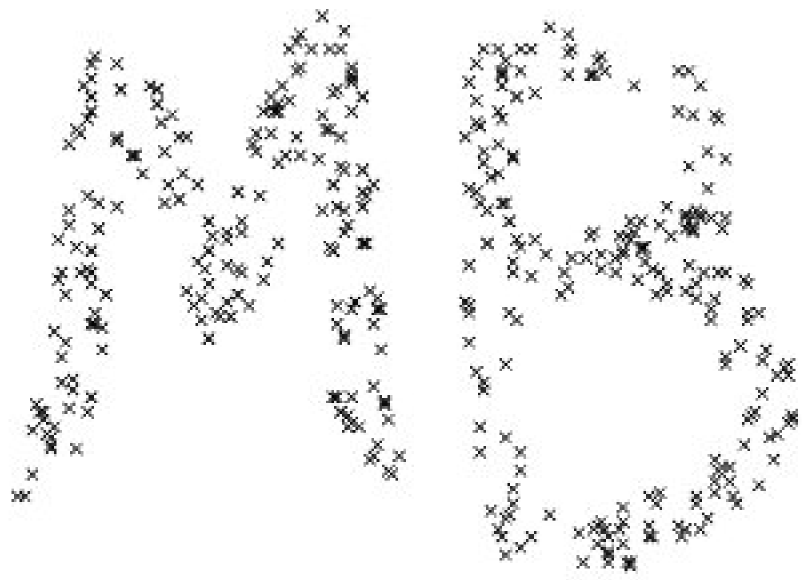
- K-means

$$\mathbf{m}_k = \frac{\sum_{j;c_j=k} \mathbf{x}_j}{\sum_{j;c_j=k} 1}$$

- EM

$$\mathbf{m}_k = \frac{\sum_j \xi_{kj}\mathbf{x}_j}{\sum_j \xi_{kj}}$$

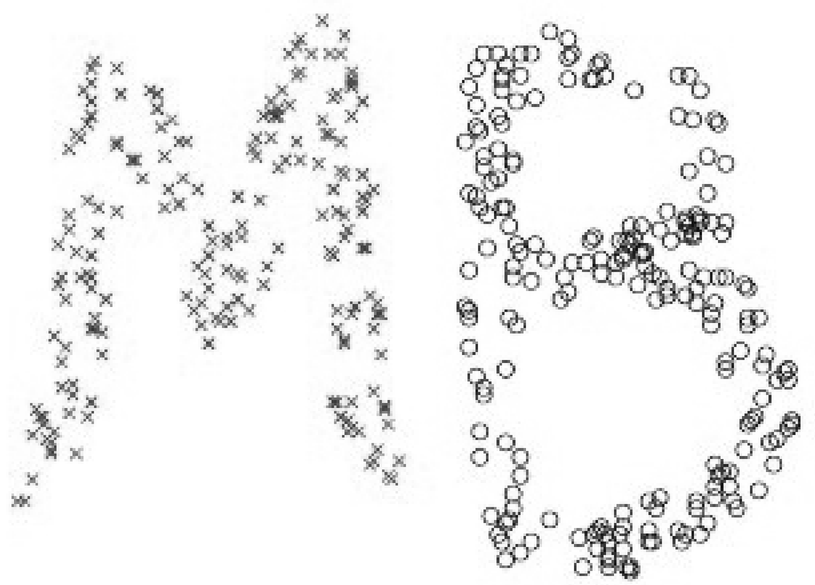
Graph theoretic clustering

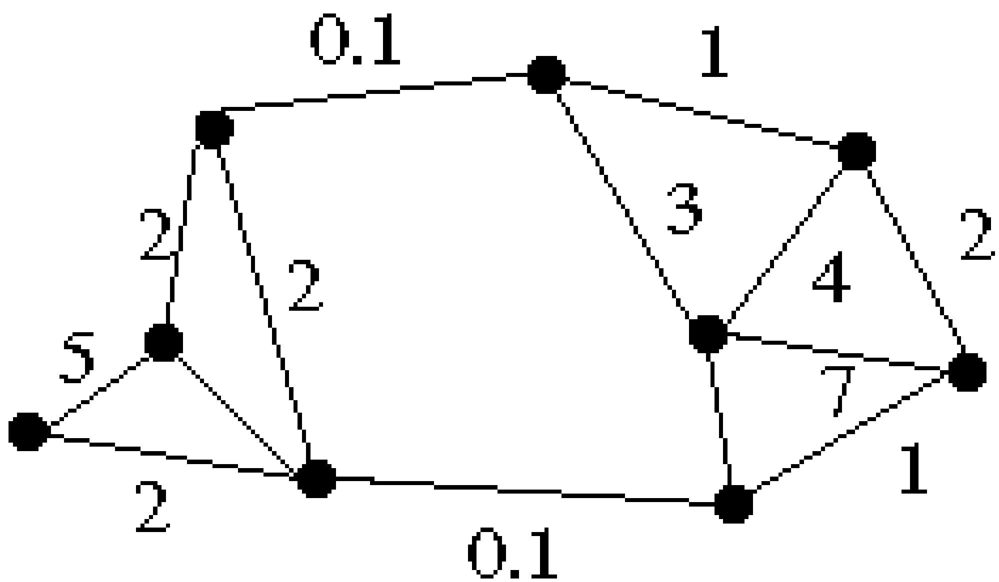


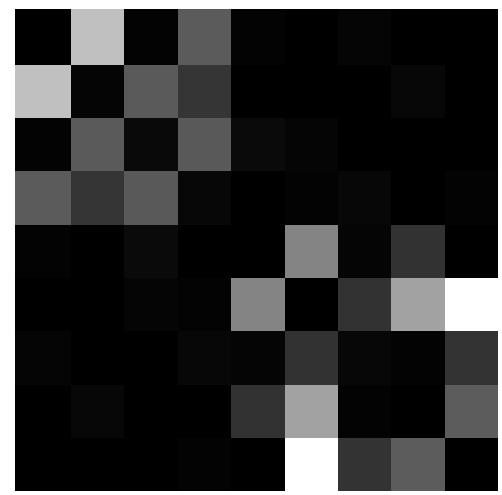
Graph theoretic clustering

- Sometimes, clusters have unusual shapes and K-means fails.
- Alternative approach: encode similarity of tokens instead of absolute properties
- Represent similarity of tokens using a weighted graph/affinity matrix
- Cut up this graph to get subgraphs with strong interior links

Graph theoretic clustering




0.1
1
2
2
3
2
4
5
7
2
1
0.1



Measuring Affinity

Intensity

$$aff(x,y) = \exp\left\{ -\left(\frac{1}{2\sigma_i^2} \right) \left(\left\| I(x) - I(y) \right\|^2 \right) \right\}$$

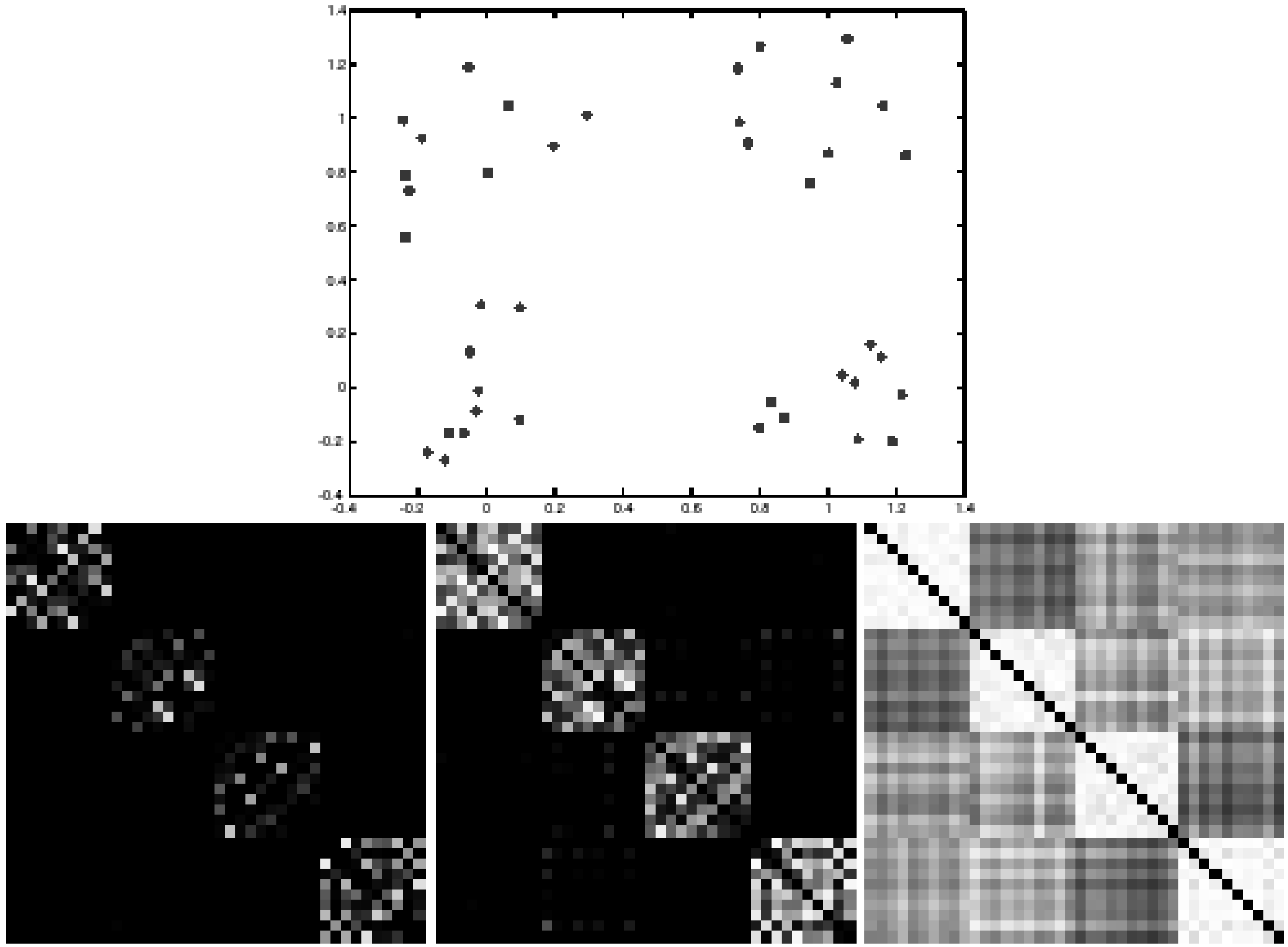
Distance

$$aff(x,y) = \exp\left\{ -\left(\frac{1}{2\sigma_d^2} \right) \left(\left\| x - y \right\|^2 \right) \right\}$$

Texture

$$aff(x,y) = \exp\left\{ -\left(\frac{1}{2\sigma_t^2} \right) \left(\left\| c(x) - c(y) \right\|^2 \right) \right\}$$

Scale affects affinity



Eigenvectors and Segmentation

- Extract a single good cluster
  - Where elements have high affinity values with each other

$$\boldsymbol{w}_n^T \mathcal{A} \boldsymbol{w}_n$$

$$\begin{aligned}&\{\text{association of element } i \text{ with cluster } n\}\times\\&\{\text{affinity between } i \text{ and } j\}\times\\&\{\text{association of element } j \text{ with cluster } n\}\end{aligned}$$

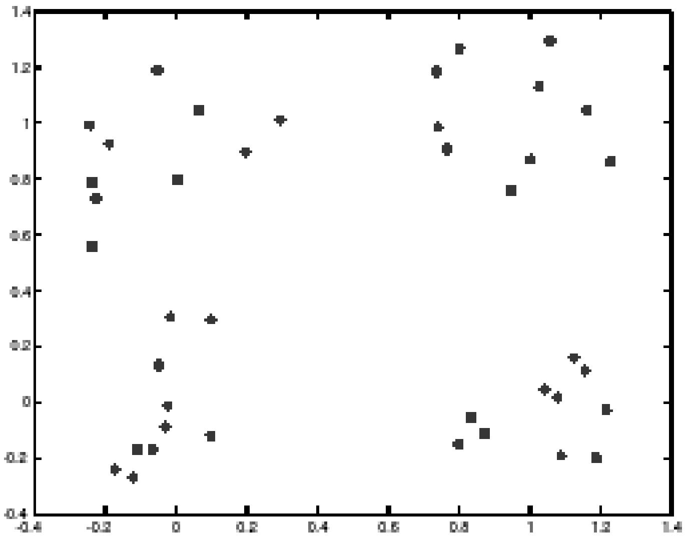
$$\Downarrow$$

$$\boldsymbol{w}_n^T \mathcal{A} \boldsymbol{w}_n + \lambda \left(\boldsymbol{w}_n^T \boldsymbol{w}_n - 1 \right)$$

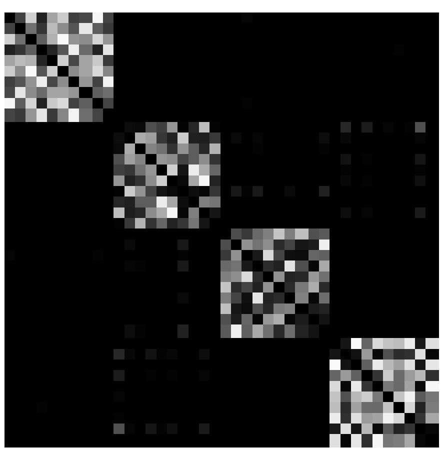
$$\Downarrow$$

$$\mathcal{A} \boldsymbol{w}_n = \lambda \boldsymbol{w}_n$$

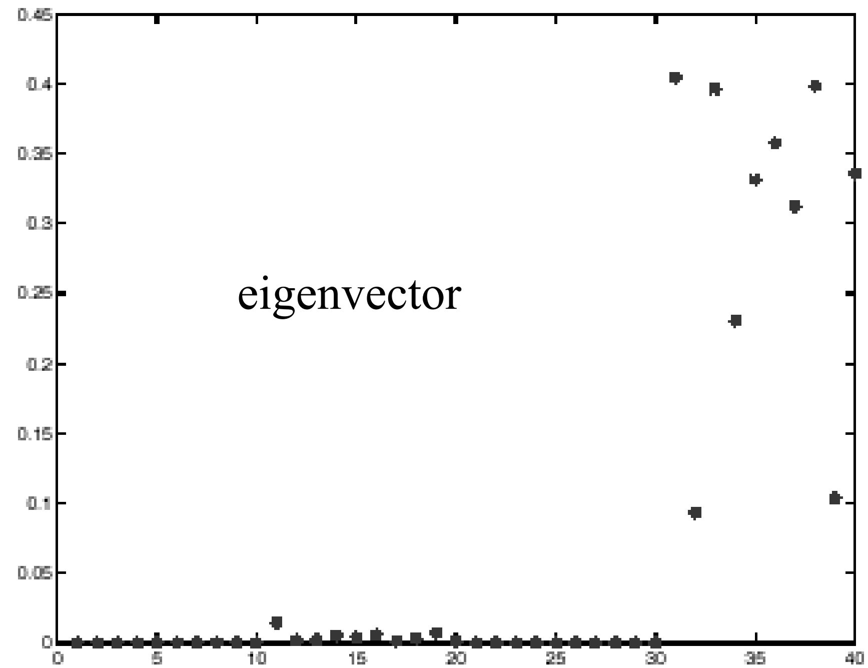
Example eigenvector



points



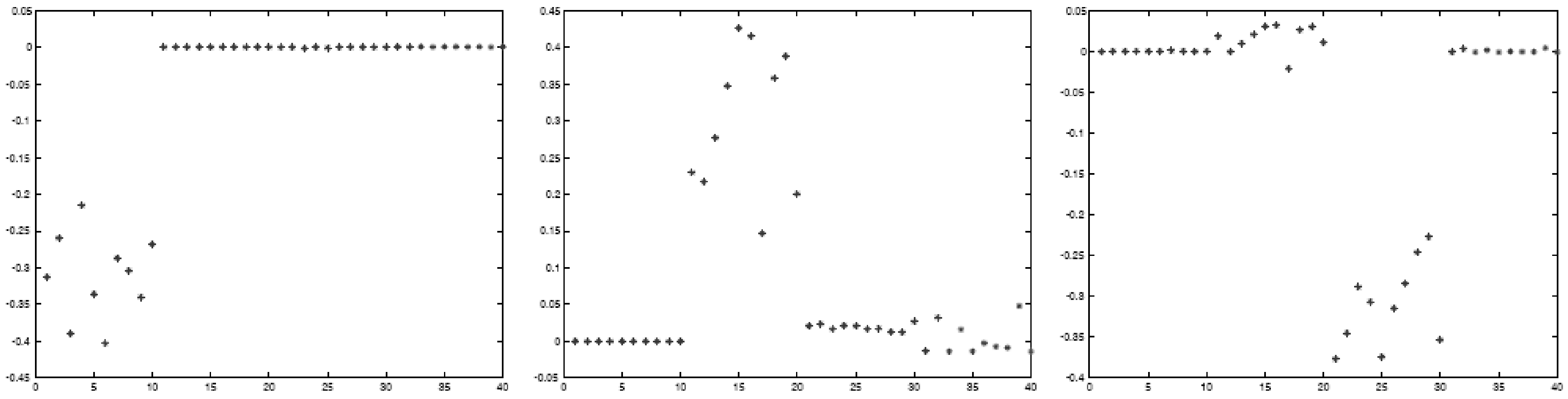
matrix

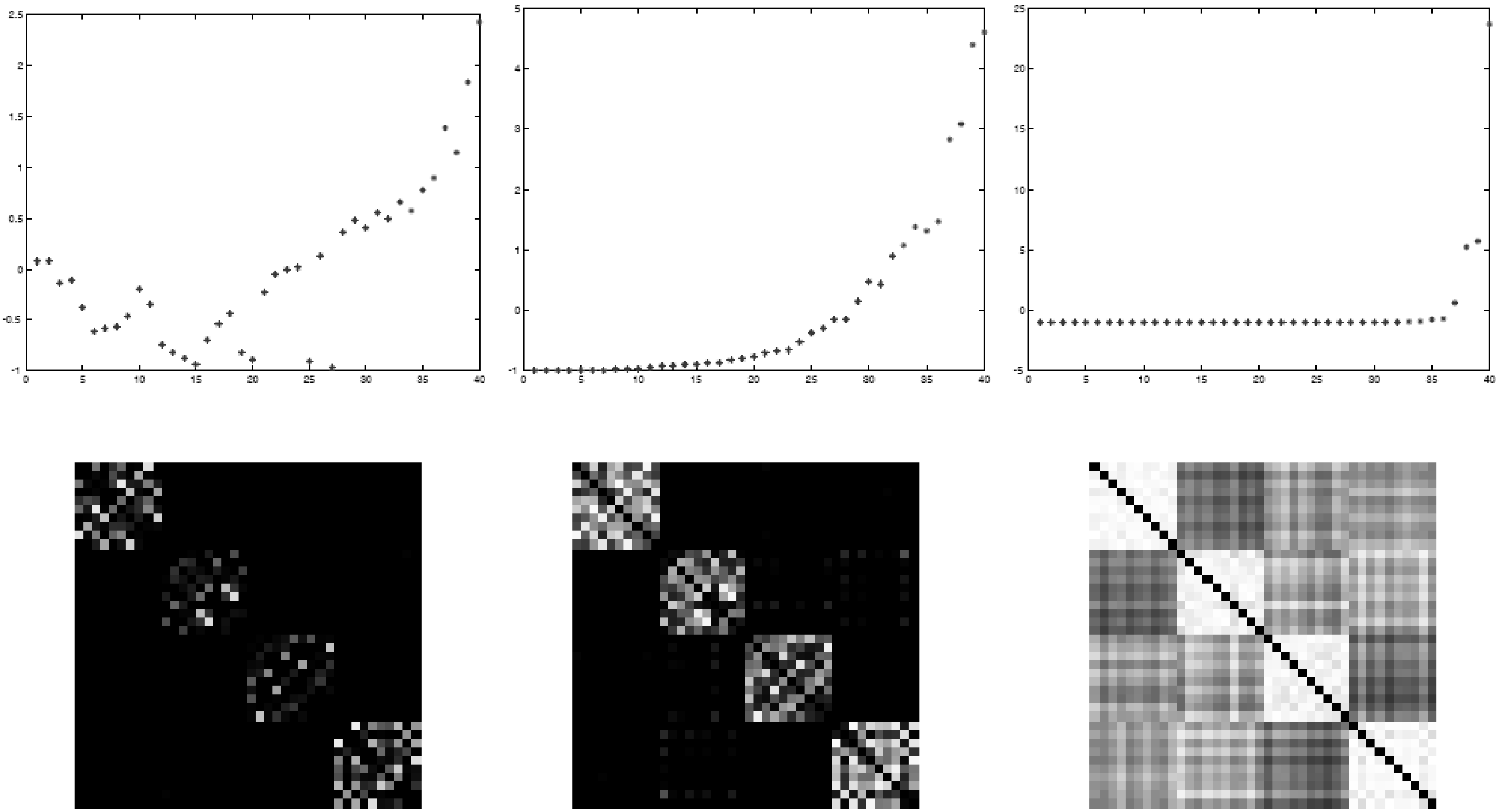


eigenvector

Eigenvectors and Segmentation

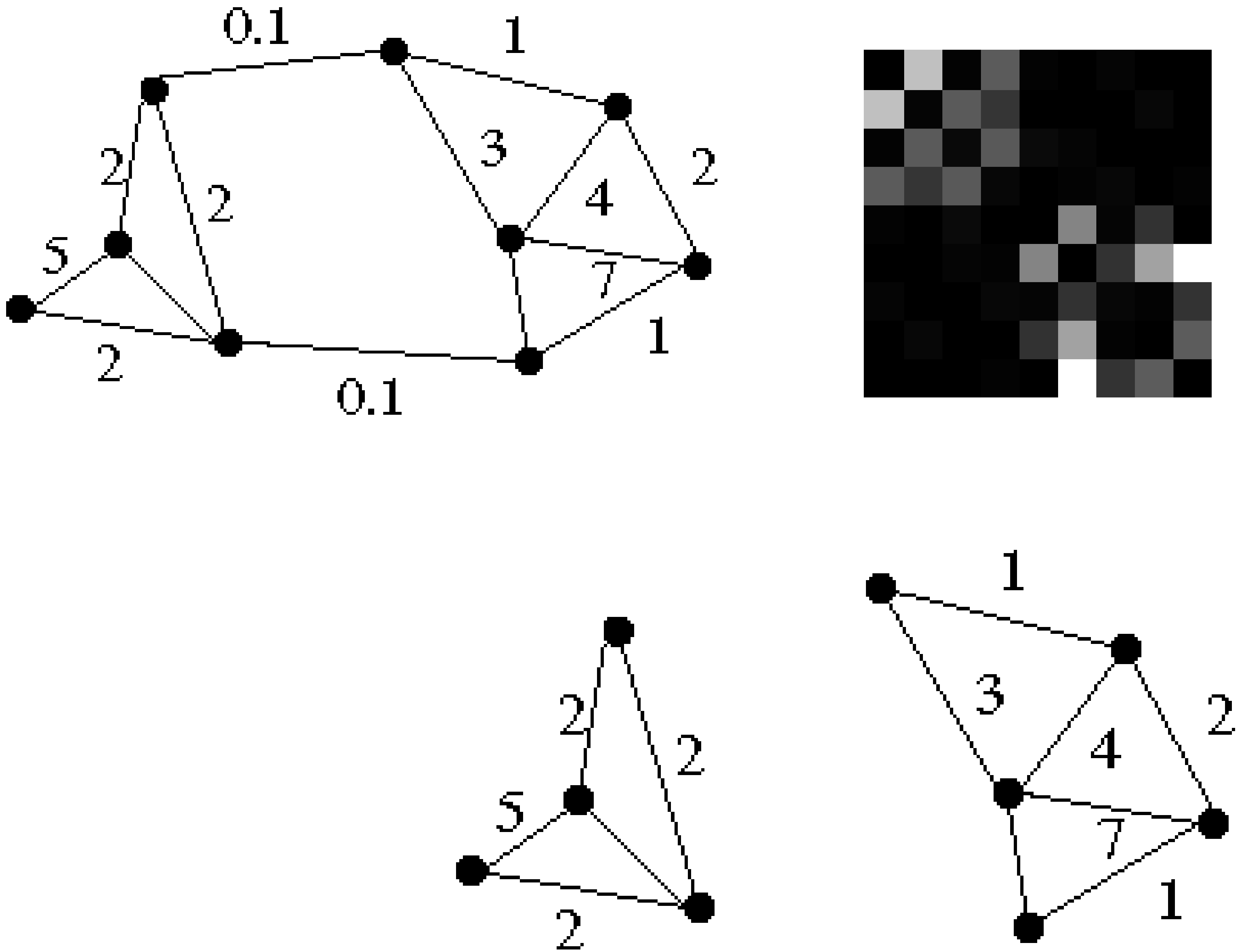
- Extract a single good cluster
- Extract weights for a set of clusters




2.5 2 1.5 1 0.5 0 -0.5 -1
0 5 10 15 20 25 30 35 40
5 4 3 2 1 0 -1
0 5 10 15 20 25 30 35 40
25 20 15 10 5 0 -5
0 5 10 15 20 25 30 35 40

Algorithm 14.6: Clustering by Graph Eigenvectors

Construct an affinity matrix
Compute the eigenvalues and eigenvectors of the affinity matrix
Until there are sufficient clusters
    Take the eigenvector corresponding to the
        largest unprocessed eigenvalue; zero all components corresponding
        to elements that have already been clustered, and threshold the
        remaining components to determine which element
        belongs to this cluster, choosing a threshold by
        clustering the components, or
        using a threshold fixed in advance.
    If all elements have been accounted for, there are
        sufficient clusters
end


0.1
1
2
3
2
4
5
7
2
1
0.1
2
5
2
2
1
3
2
4
7
1

Normalized cuts

- Current criterion evaluates within cluster similarity, but not across cluster difference
- Instead, we'd like to maximize the within cluster similarity compared to the across cluster difference
- Write graph as V, one cluster as A and the other as B

- Minimize

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

- *(or equivalently)* Maximize

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

- i.e. construct A, B such that their within cluster similarity is high compared to their association with the rest of the graph

Normalized cuts

- Write a vector y whose elements are 1 if item is in A, -b if it's in B
- Write the matrix of the graph as W, and the matrix which has the row sums of W on its diagonal as D, 1 is the vector with all ones.
- Criterion becomes

$$\min_{\mathrm{y}}\left(\frac{y^T(D-W)y}{y^T Dy}\right)$$

- and we have a constraint

$$y^T D1 = 0$$

- This is hard to do, because y's values are quantized

Normalized cuts

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} = 1 - \frac{\mathbf{y}^T\mathbf{W}\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

$$\text{let } \mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$$

$$\begin{aligned} \mathbf{z}^* &= \arg\min_{\mathbf{z}} 1 - \frac{\mathbf{z}^T\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{1/2}\mathbf{z}}{\mathbf{z}^T\mathbf{z}} \\ &= \arg\max_{\mathbf{z}} \frac{\mathbf{z}^T\mathbf{A}\mathbf{z}}{\mathbf{z}^T\mathbf{z}} \end{aligned}$$

$\Longrightarrow$ $\mathbf{z}$ is the eigenvector of $\mathbf{A} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{1/2}$ corresponding to the largest eigenvalue.

BUT the constraint $\mathbf{y}^T\mathbf{D}\mathbf{1} = 0$ means that the solution is actually the **second largest eigenvector**

Normalized cuts

- Instead, solve the generalized eigenvalue problem

$$\max_y \left(y^T (D - W) y \right) \text{ subject to } \left(y^T D y = 1 \right)$$

- which gives

$$(D - W) y = \lambda D y$$

- Now look for a quantization threshold that maximises the criterion --- i.e all components of y above that threshold go to one, all below go to -b

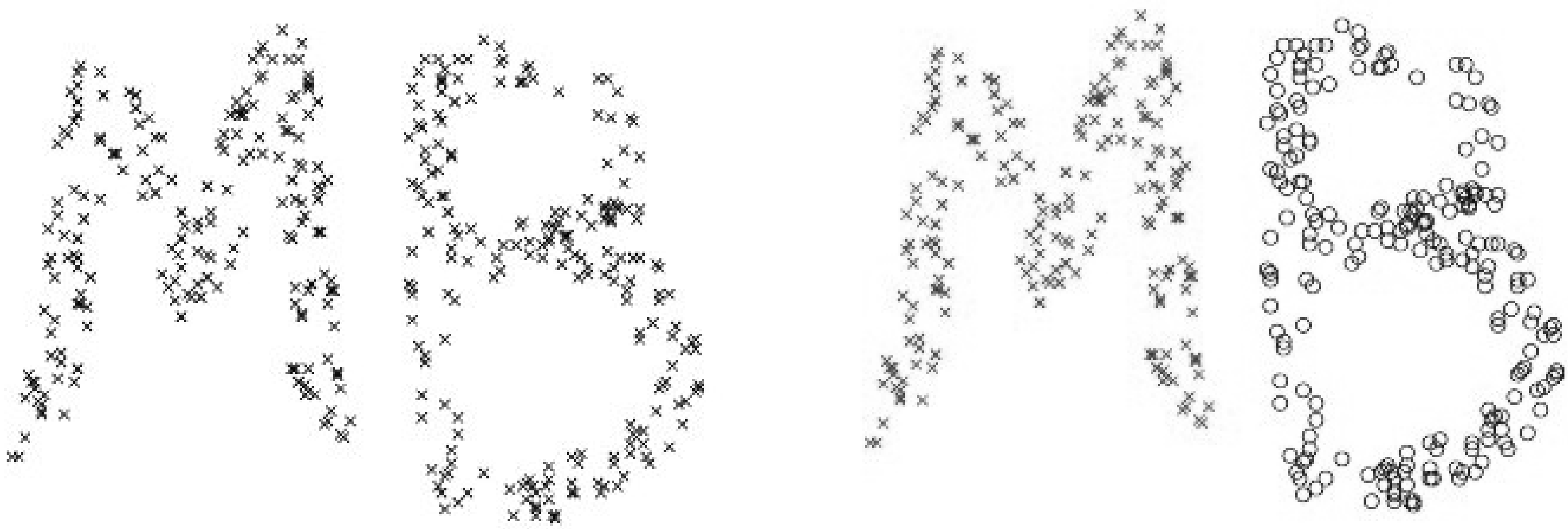
More than two segments

- Two options
  - Recursively split each side to get a tree, continuing till the eigenvalues are too small
  - Use the other eigenvectors

Normalized cuts: Summary

1. Compute affinity matrix $\mathbf{W}(i,j) = \exp^{-\frac{1}{2}(\frac{|\mathbf{x}_i - \mathbf{x}_j|}{\sigma})^2}$
2. Normalise affinity matrix $\mathbf{N} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{\frac{1}{2}}$, $\mathbf{D}(i,i) = \sum_j \mathbf{W}(i,j)$
3. Calculate top k eigenvectors $\mathbf{V} = eig(\mathbf{N})$
4. Map $\mathbf{x}_i$ to vector in $\mathbb{R}^k$, $\mathbf{x}_i \rightarrow \mathbf{V}(i, 1:k)$
5. Partition vectors in $\mathbb{R}^k$

Normalized cuts: Results



Normalized cuts: Results

(a) Input image (b) 2nd Eigenvector (c) Segmentation

Figure 3: Unsupervised segmentation of donkey image using Normalised Cuts

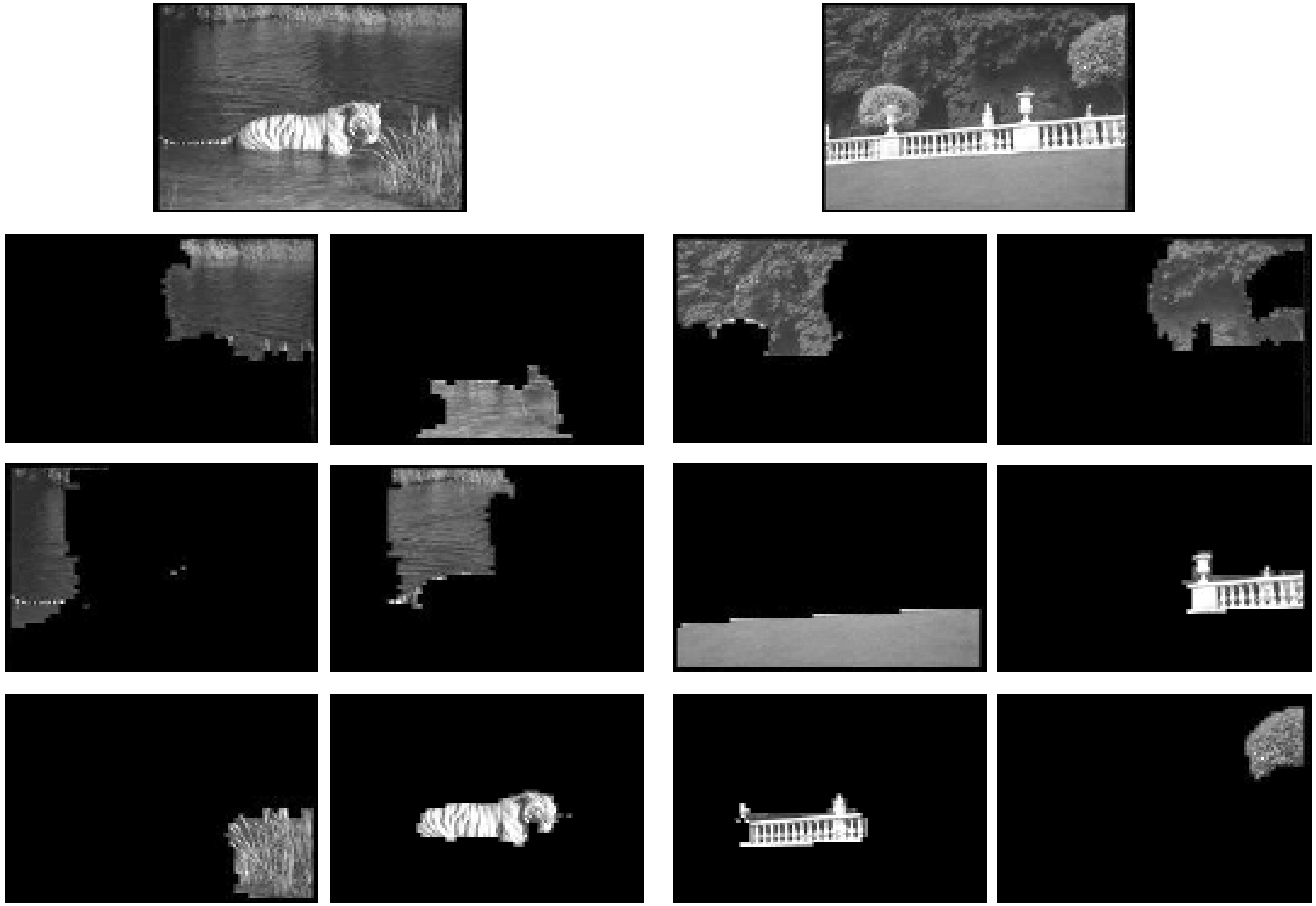
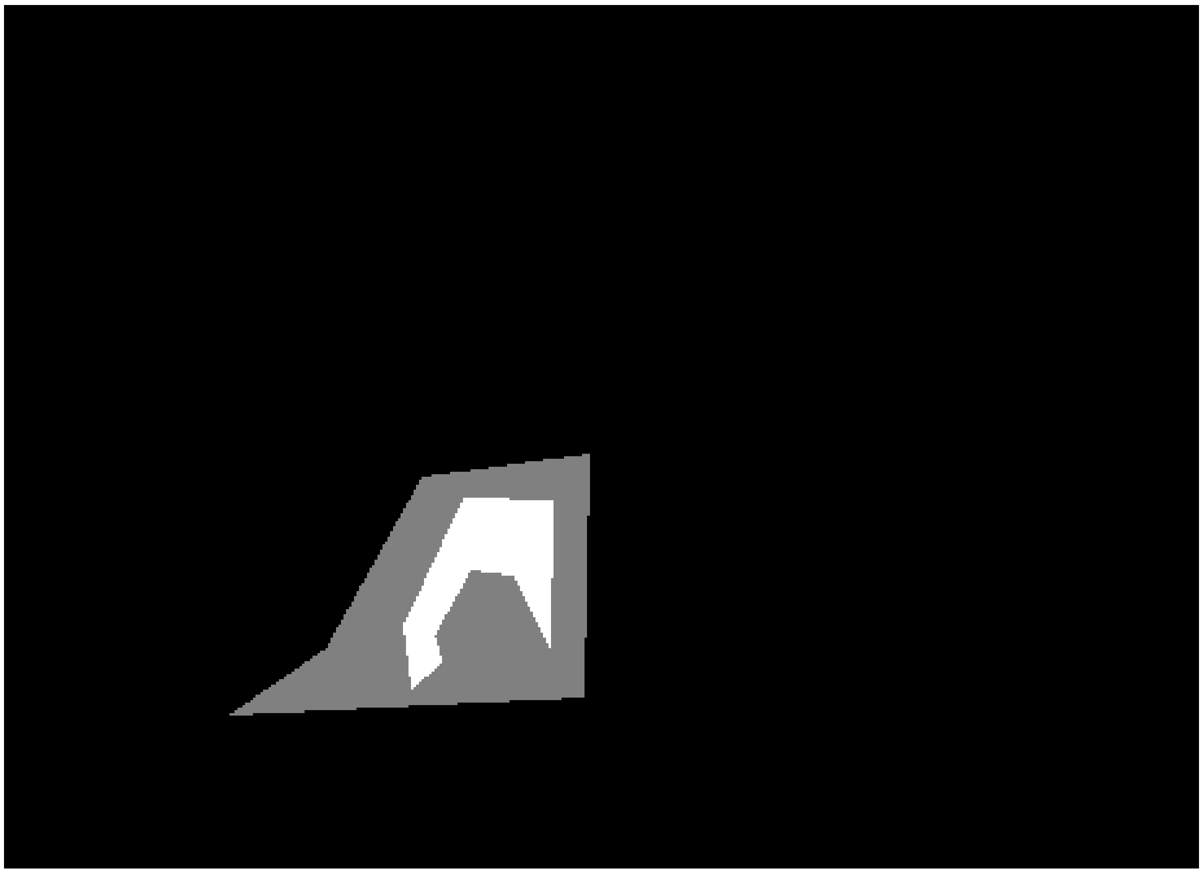


Figure from “Image and video segmentation: the normalised cut framework”, by Shi and Malik, copyright IEEE, 1998

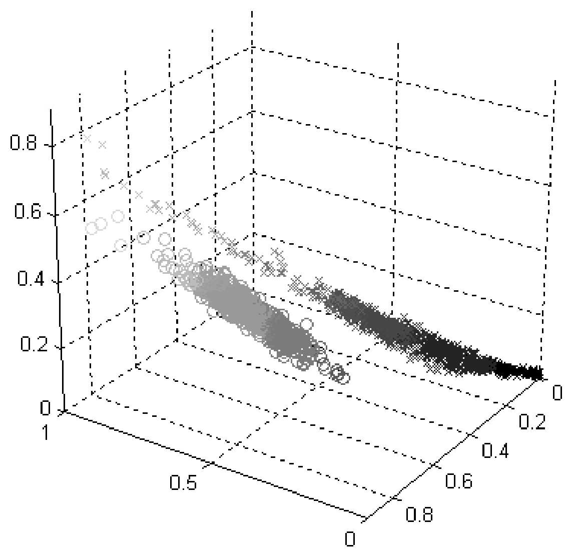
F igure from “Normalized cuts and image segmentation,” Shi and Malik, copyright IEEE, 2000

Application: Supervised Segmentation I

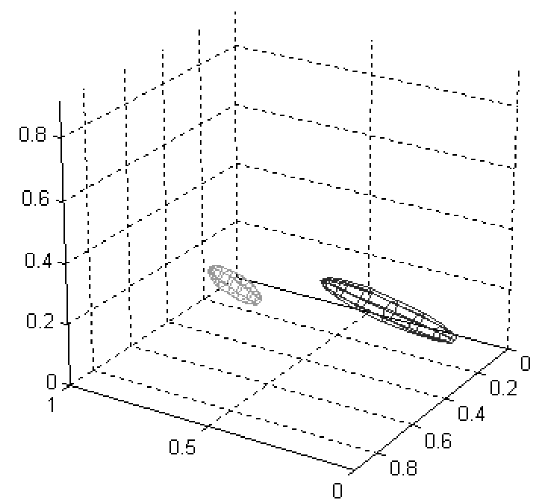


Gaussian Colour Distributions

Data in Colour Space



Gaussian Models

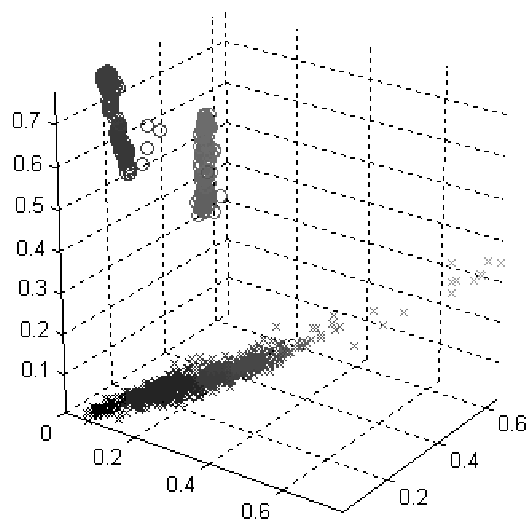


Gaussian Colour Distributions

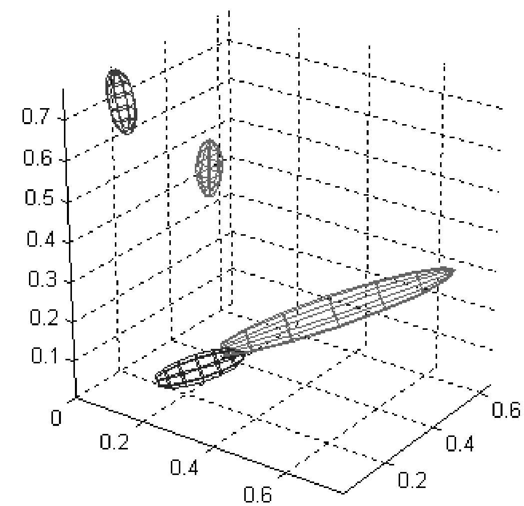
Application: Supervised Segmentation II

Mixture of Gaussian Colour Distributions

Data in Colour Space

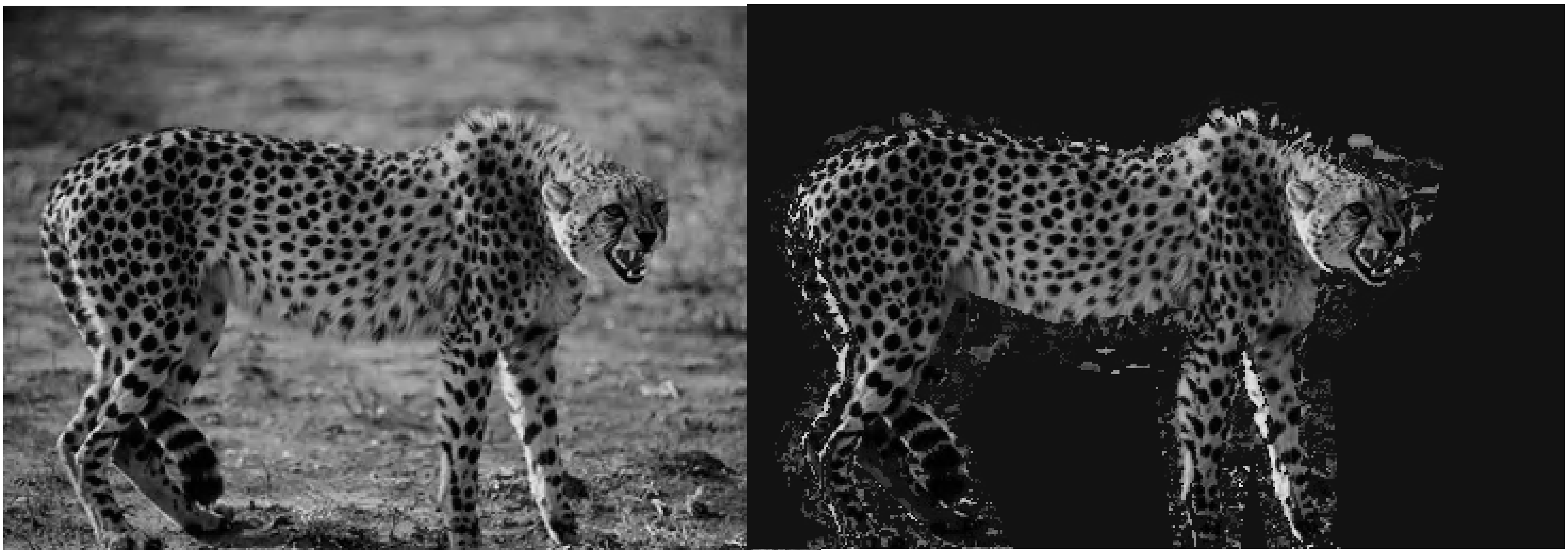


Mixture of Gaussian Models



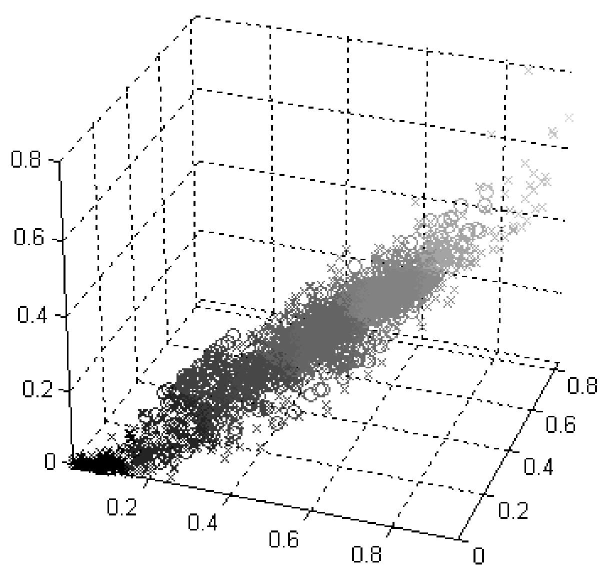
Mixture of Gaussian Colour Distributions

Colour is not always sufficient…

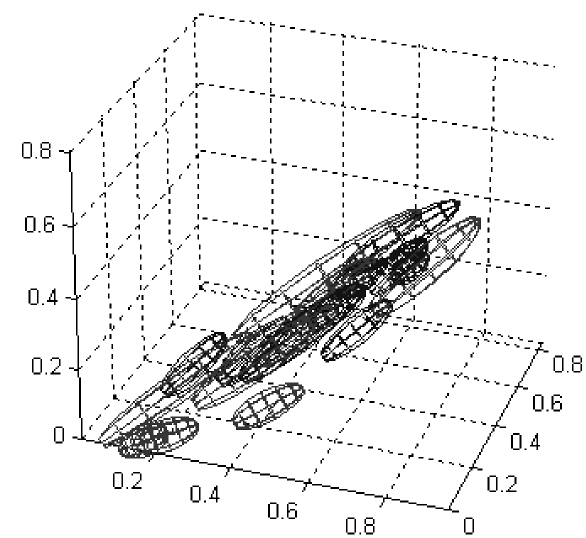


Colour is not always sufficient…

Data in Colour Space

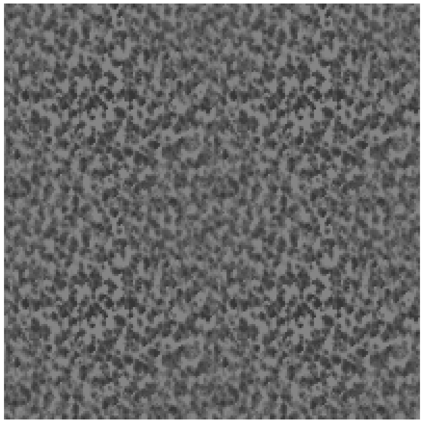
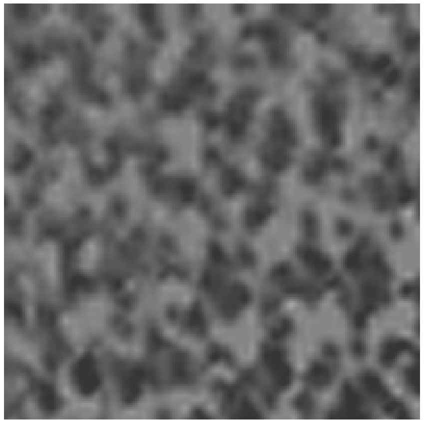


Mixture of Gaussian Models



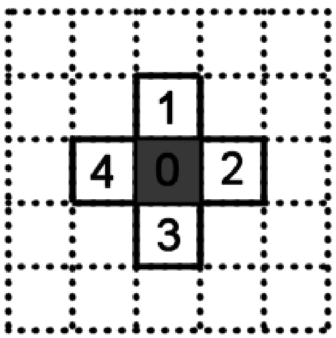
Texture = Colour in context

- Independent pixel assumption fails when foreground and background share colours
- Need to look at **context**
  - ? Over what area should we look ?

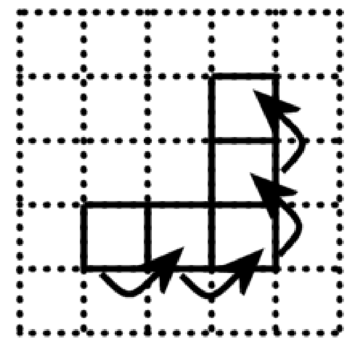
 

Markov Random Fields

- Markov Random Fields (MRFs) express spatial dependence in terms of neighbours



$$p(x_0|\mathbf{x}) = p(x_0|x_1, x_2, x_3, x_4)$$



- BUT this does not mean that non-neighbours are independent!

Example: Markov Random Fields

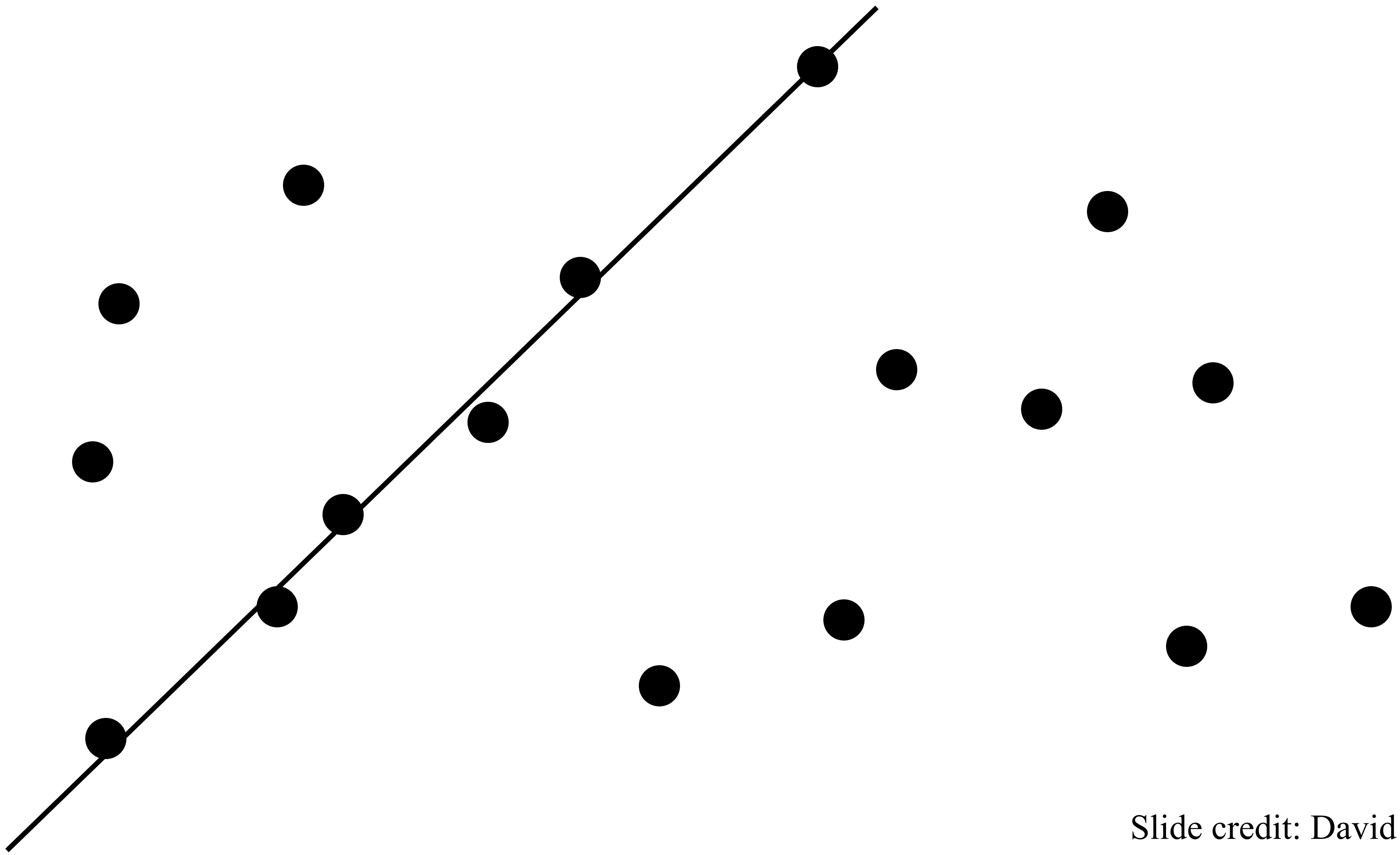
A. Blake, C. Rother, M. Brown, P. Perez and P. Torr. Interactive image segmentation using an adaptive Gaussian Mixture MRF model. ECCV2004

Fitting a Model to Data
Reading: 15.1, 15.5.2

- Cluster image parts together by fitting a model to some selected parts

- **Examples:**
  - A line fits well to a set of points. This is unlikely to be due to chance, so we represent the points as a line.
  - A 3D model can be rotated and translated to closely fit a set of points or line segments. It it fits well, the object is recognized.

Line Grouping Problem



Slide credit: David Jacobs

This is difficult because of:

- Extraneous data: clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- Missing data: only some parts of model are present
- Noise

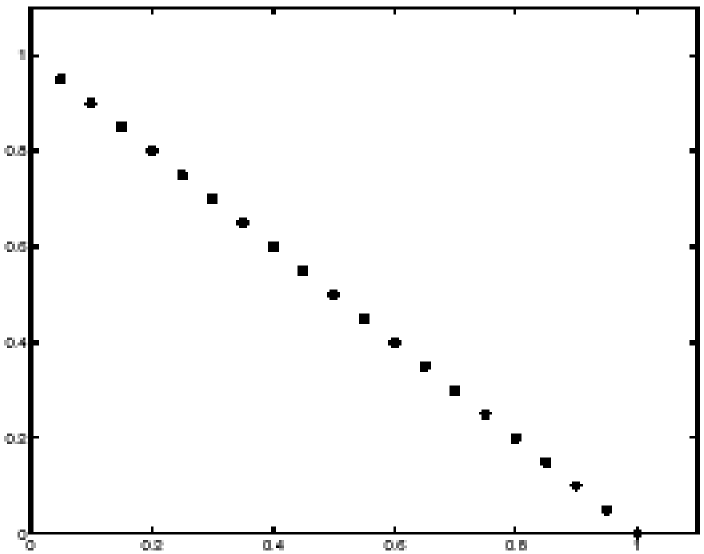
- **Cost:**
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

The Hough Transform for Lines

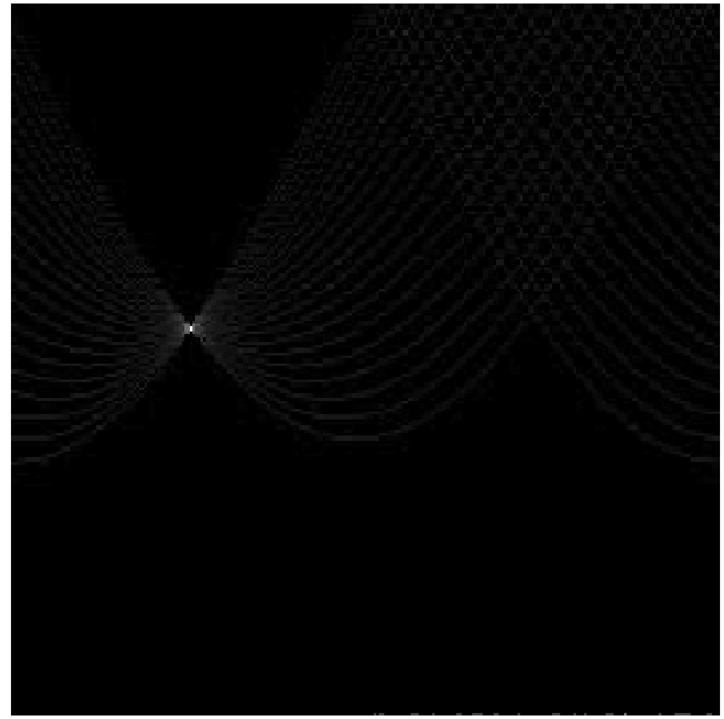
- **Idea:** Each point votes for the lines that pass through it.
- A line is the set of points (x, y) such that

$$(\sin\theta)x + (\cos\theta)y + d = 0$$

- Different choices of θ, d give different lines
- For any (x, y) there is a one parameter family of lines through this point. Just let (x,y) be constants and for each value of θ the value of d will be determined.
- Each point enters votes for each line in the family
- If there is a line that has lots of votes, that will be the line passing near the points that voted for it.

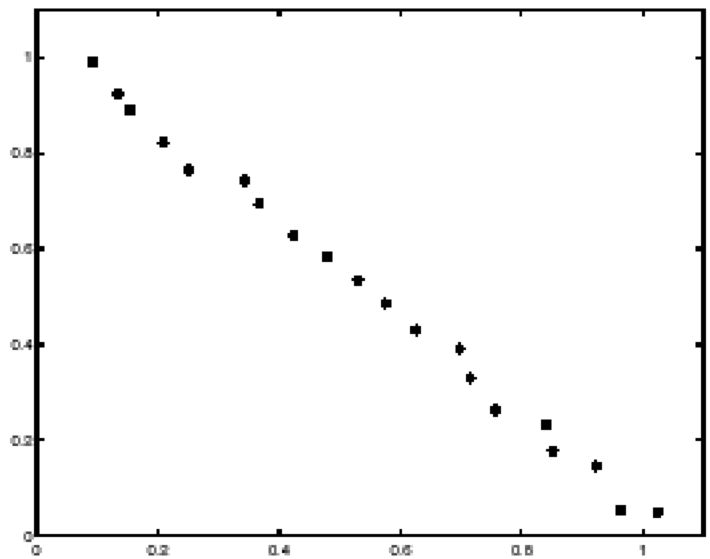


tokens

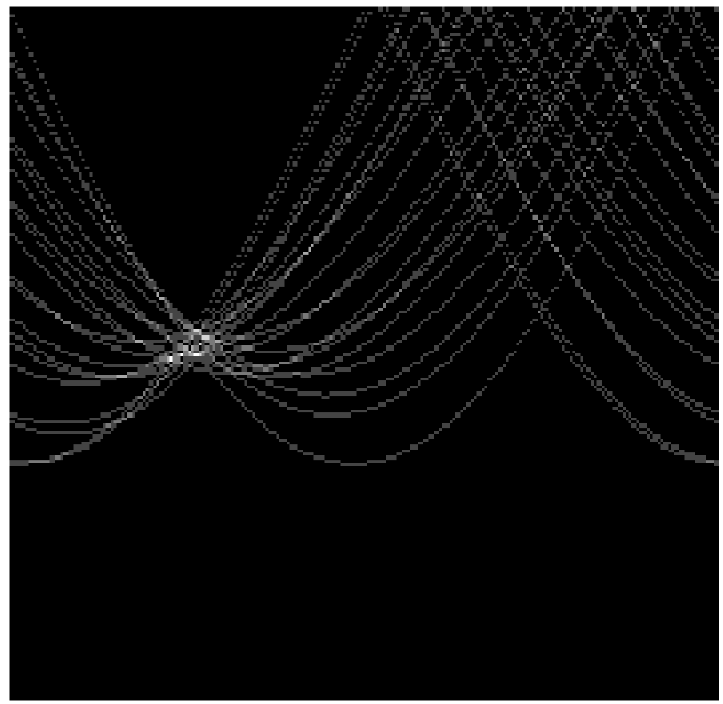


Votes

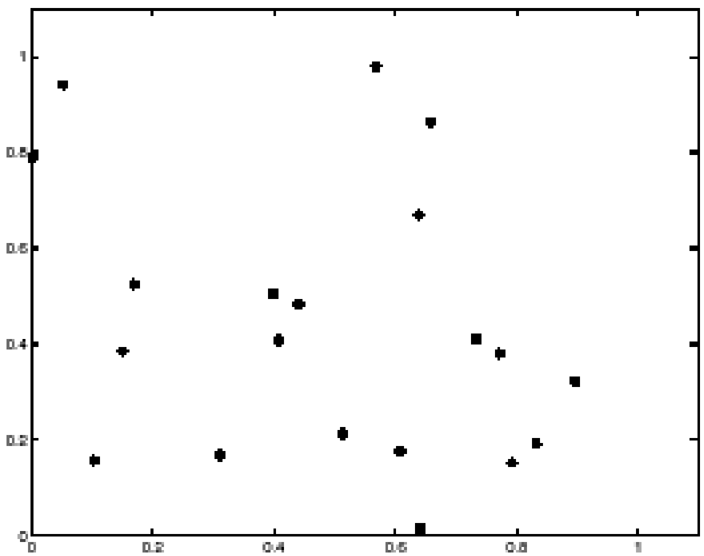
Horizontal axis is θ, vertical is d.



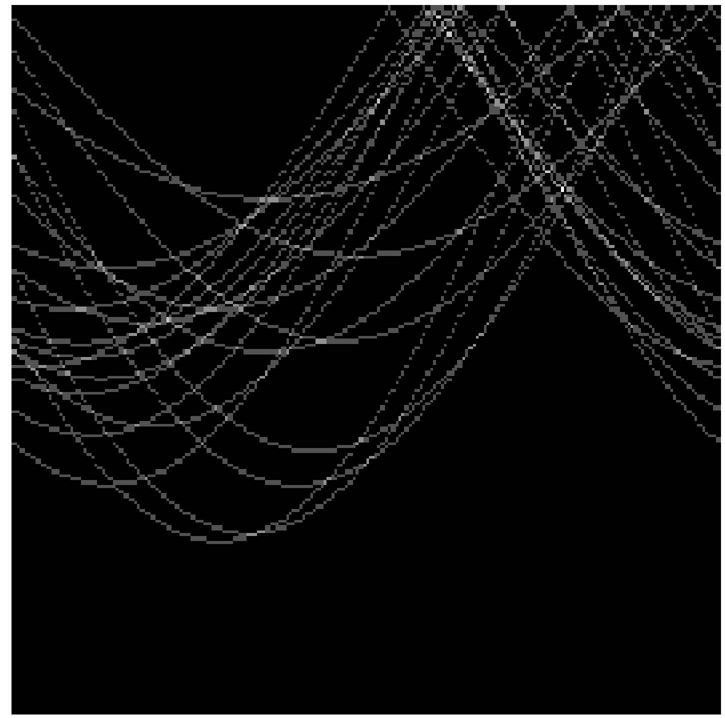
tokens

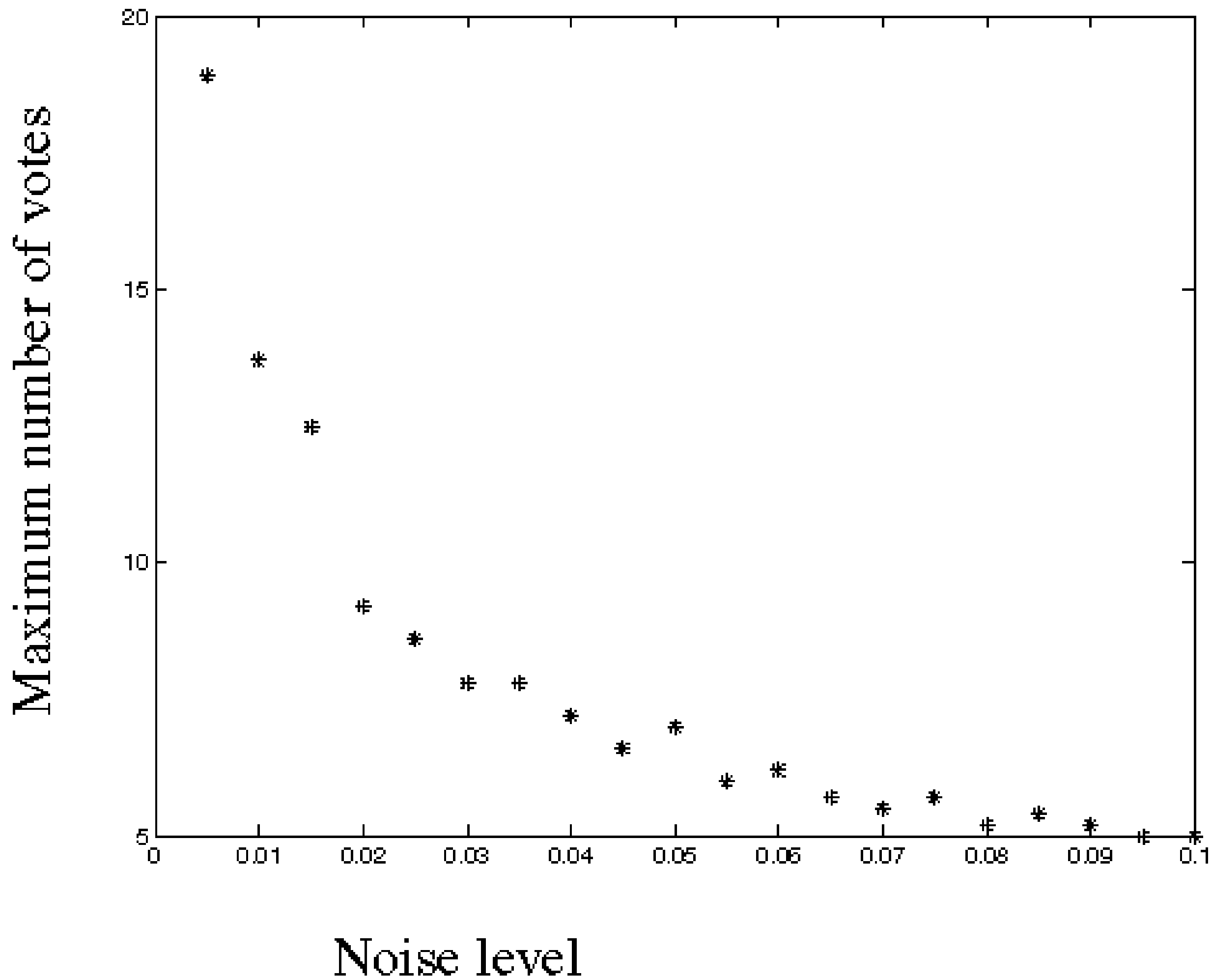


votes

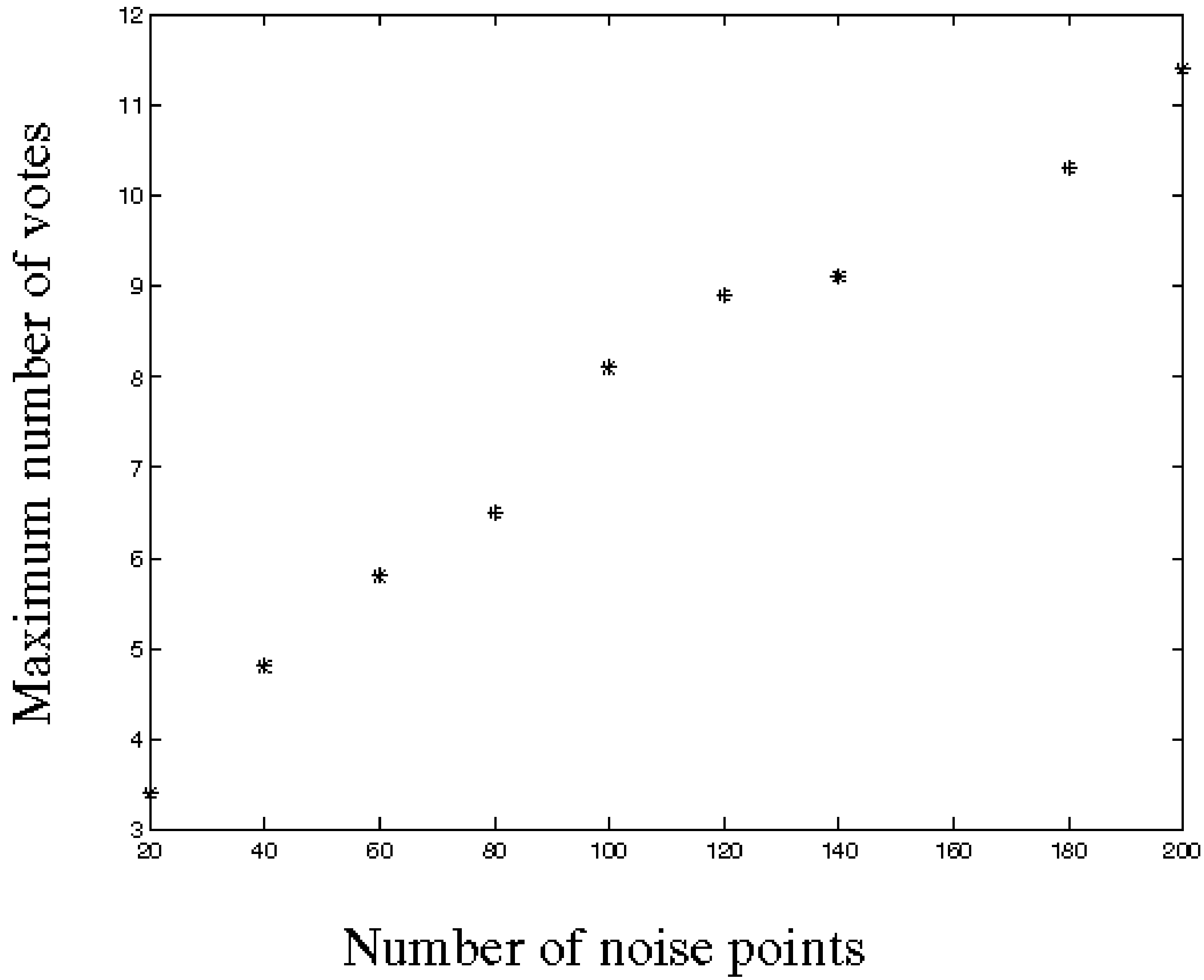


0
0.2
0.4
0.6
0.8
1





Fewer votes land in a single bin when noise increases.



Adding more clutter increases number of bins with false peaks.

Mechanics of the Hough transform

- Construct an array representing θ, d
- For each point, render the curve (θ, d) into this array, adding one vote at each cell
- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak
- Which points belong to each line?
  - Search for points close to the line
  - Solve again for line and iterate

More details on Hough transform

- It is best to vote for the two closest bins in each dimension, as the locations of the bin boundaries is arbitrary.
  - By “bin” we mean an array location in which votes are accumulated
  - This means that peaks are “blurred” and noise will not cause similar votes to fall into separate bins
- Can use a hash table rather than an array to store the votes
  - This means that no effort is wasted on initializing and checking empty bins
  - It avoids the need to predict the maximum size of the array, which can be non-rectangular

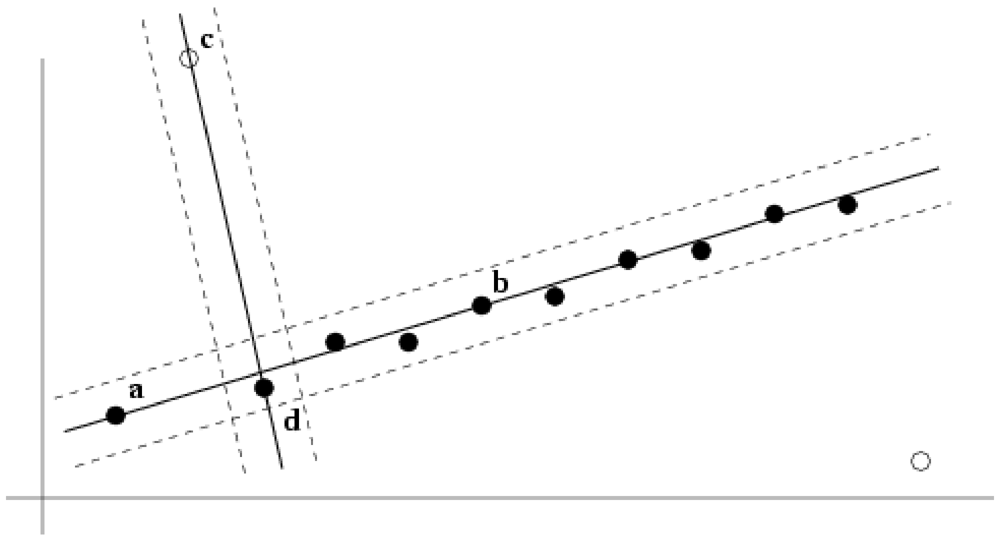
When is the Hough transform useful?

- The textbook wrongly implies that it is useful mostly for finding lines
  - In fact, it can be very effective for recognizing arbitrary shapes or objects
- The key to efficiency is to have each feature (token) determine as many parameters as possible
  - For example, lines can be detected much more efficiently from small edge elements (or points with local gradients) than from just points
  - For object recognition, each token should predict scale, orientation, and location (4D array)
- **Bottom line:** The Hough transform can extract feature groupings from clutter in linear time!

RANSAC
(RANdom SAmple Consensus)

1. Randomly choose minimal subset of data points necessary to fit model (a *sample*)
2. Points within some distance threshold t of model are a *consensus set*. Size of consensus set is model's *support*
3. Repeat for N samples; model with biggest support is most robust fit
   – Points within distance t of best model are inliers
   – Fit final model to all inliers

Two samples
and their supports
for line-fitting



from Hartley & Zisserman

Slide: Christopher Rasmussen

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
        to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
        uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
        Test the distance from the point to the line
            against t; if the distance from the point to the line
            is less than t, the point is close
    end
    If there are d or more points close to the line
        then there is a good fit. Refit the line using all
        these points.
end
Use the best fit from this collection, using the
    fitting error as a criterion

RANSAC: How many samples?

How many samples are needed?

Suppose *w* is fraction of inliers (points from line).

n points needed to define hypothesis (2 for lines)

k number of samples.

Probability that a single sample of n points is correct:

$$w^n$$

Probability that all samples fail is:

$$(1-w^n)^k$$

Choose *k* high enough to keep this below desired failure rate.

RANSAC: Computed k (p = 0.99)

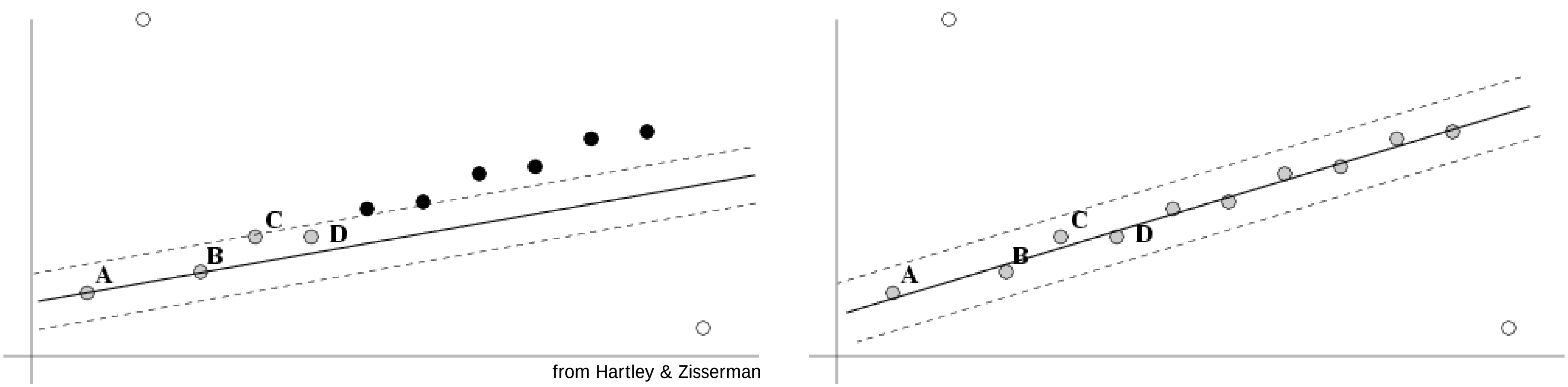
Sample size	Proportion of outliers						
n	**5%**	**10%**	**20%**	**25%**	**30%**	**40%**	**50%**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

adapted from Hartley & Zisserman

Slide credit: Christopher Rasmussen

After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers
- Improve this initial estimate with estimation over all inliers (e.g., with standard least-squares minimization)
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier



from Hartley & Zisserman

Slide credit: Christopher Rasmussen

Discussion of RANSAC

- **Advantages:**
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Disadvantages:**
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- The Hough transform can handle high percentage of outliers