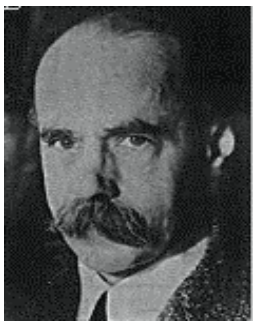
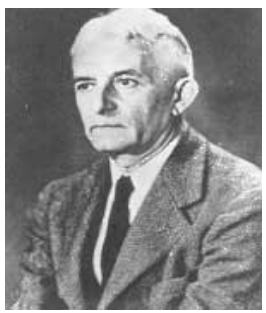
Basic ideas of grouping in humans: The Gestalt school

Wertheimer



Koehler

Koffka



Gestalt properties

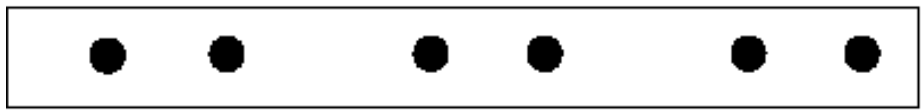
elements in a collection of elements can have properties that result from relationships

- Gestaltqualitat

A series of factors affect whether elements should be grouped together

- Gestalt factors

Not grouped



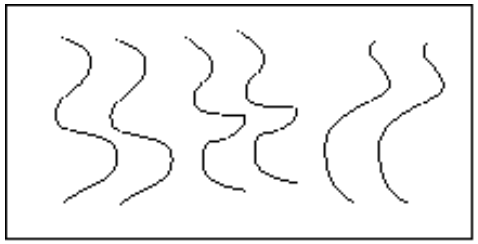
Proximity

Similarity

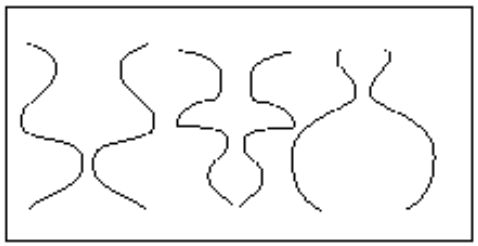
Similarity

Common Fate

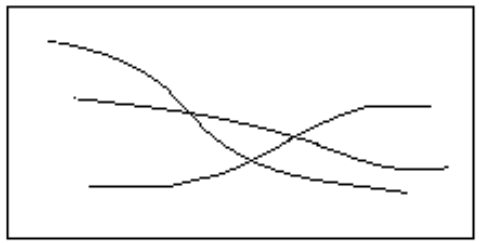
Common Region



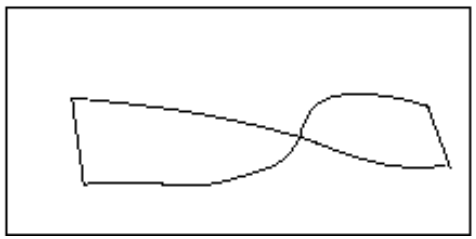
Parallelism



Symmetry



Continuity



Closure

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
- Point-Cluster distance
  - single-link clustering
  - complete-link clustering
  - group-average clustering
- Dendrograms
  - yield a picture of output as clustering process continues

Clustering

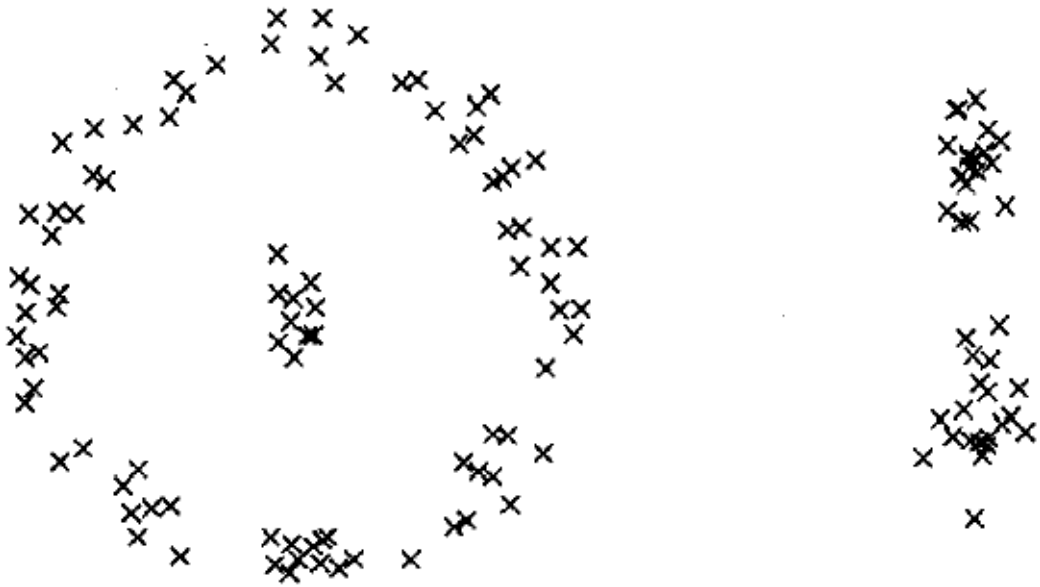
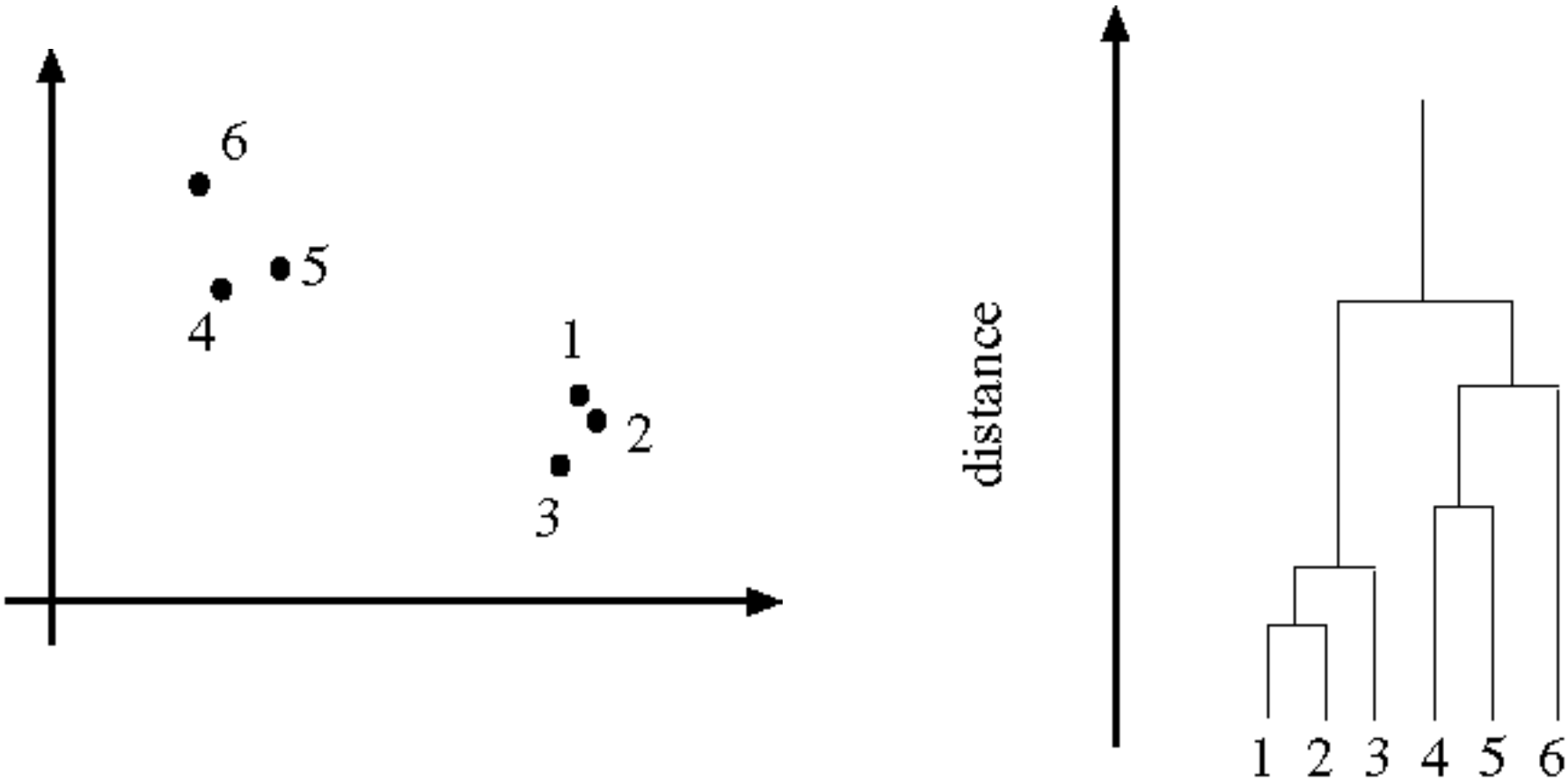


Figure 1: How many groups?


6
5
4
1
2
3
distance
1 2 3 4 5 6

K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error
- can't do this by search, because there are too many possible allocations.

- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

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

K-Means Algorithm

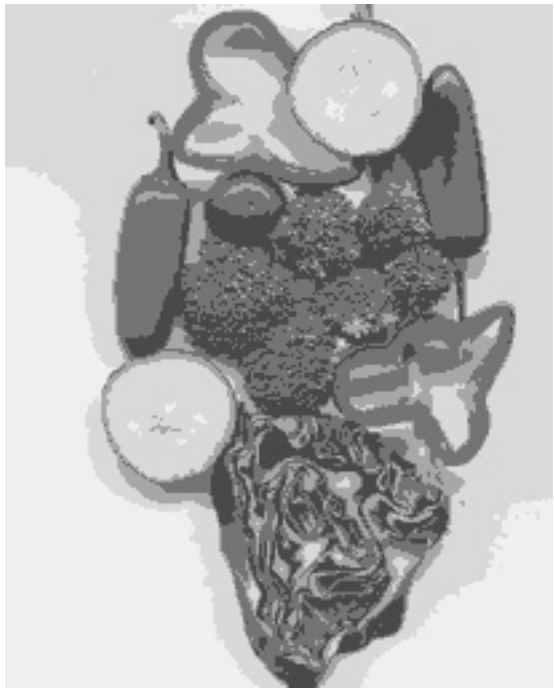
1. Choose *k* data points to act as cluster centers
2. Repeat
3. Allocate each data point to cluster whose center is nearest
4. Replace the cluster centers with the centroid of the elements in their clusters
5. Until the cluster centers are unchanged

NOTE: In step 2), ensure that every cluster has at least one data point.
Possible techniques for doing this include supplying empty clusters with a point chosen at random from points far from their cluster centers.

Image

Clusters on intensity

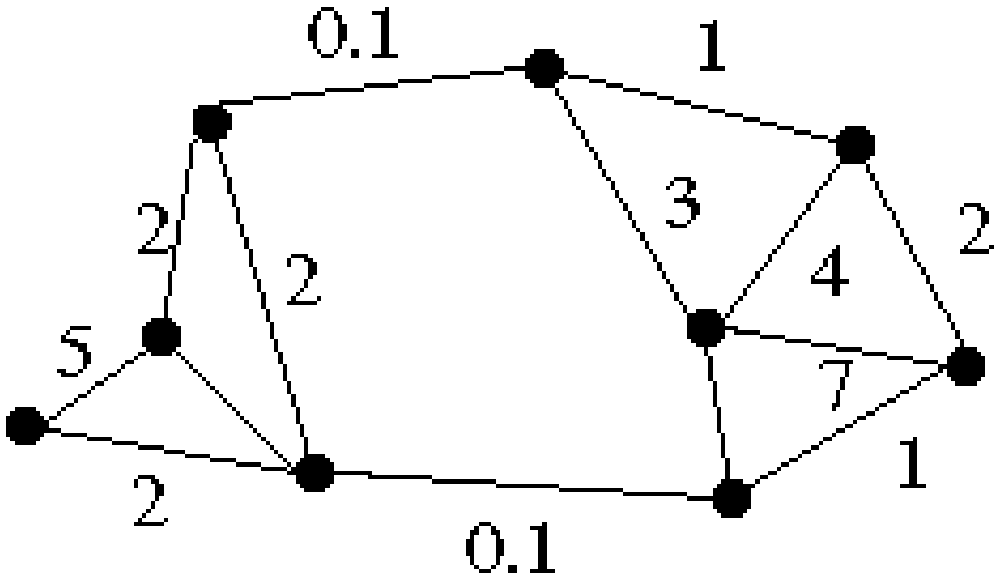
Clusters on color

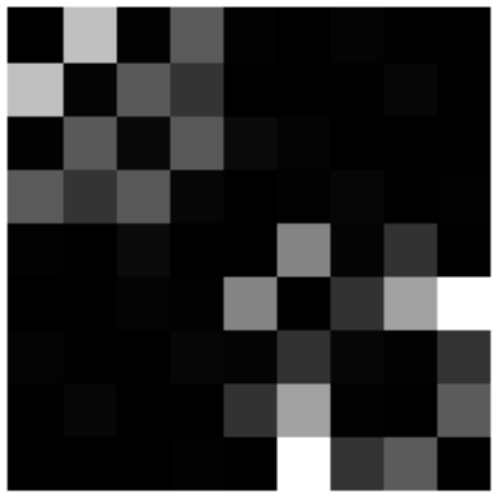


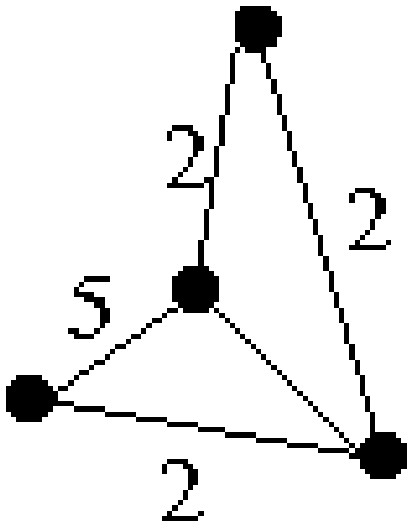
K-means clustering using intensity alone and color alone

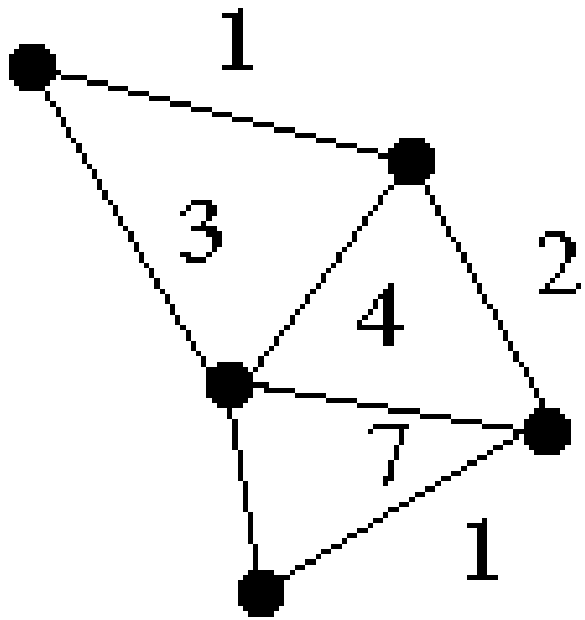
Graph-theoretic (pairwise) clustering

- Represent tokens using a weighted graph.
  - affinity matrix
- Cut up this graph to get subgraphs with strong interior links


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




2
2
5
2


1
3
2
4
7
1

Measuring Affinity

Intensity

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

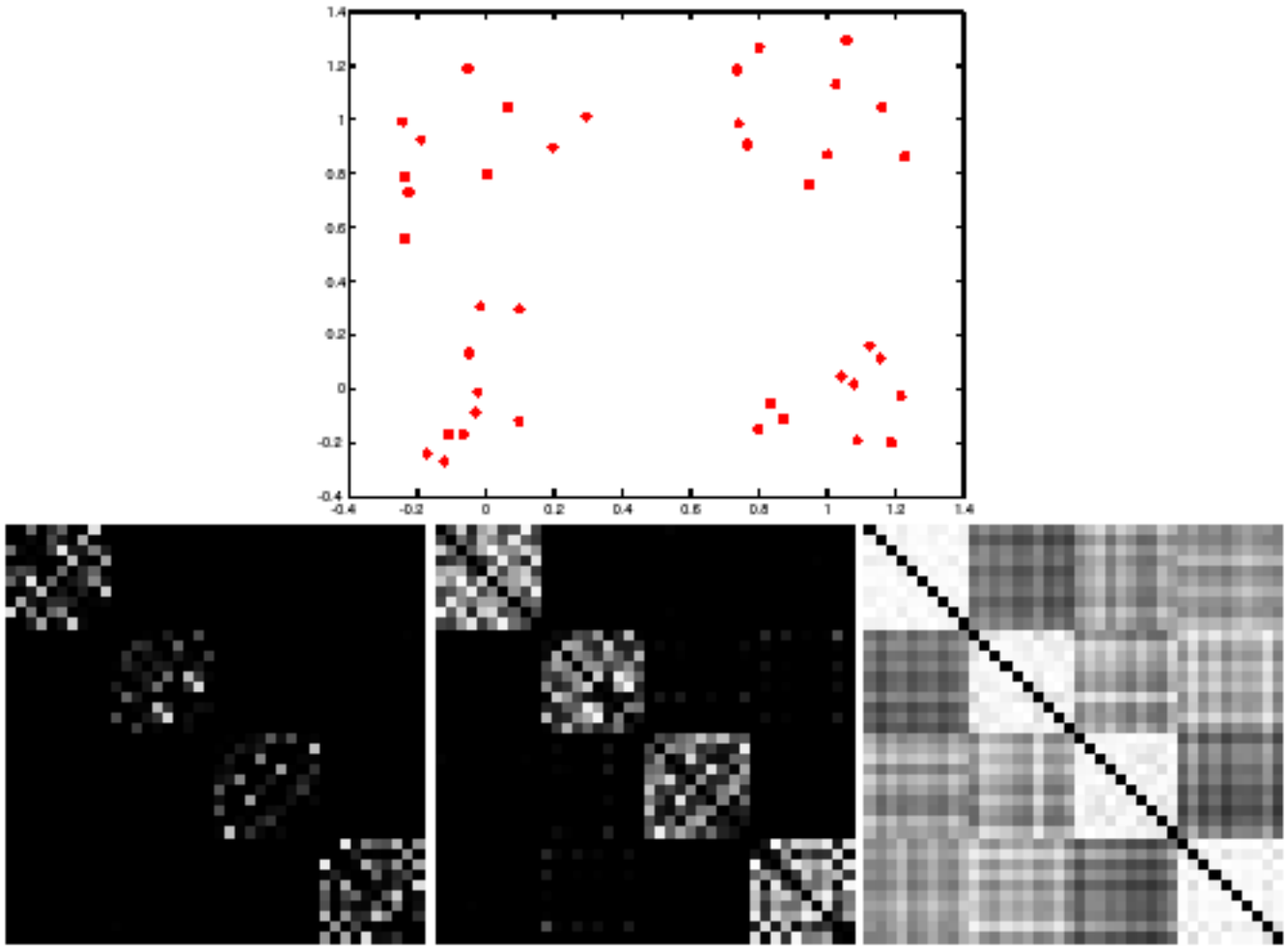
Distance

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

Texture

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

Scale affects affinity



Eigenvector-based clustering

- Simplest idea: we want a vector a giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize

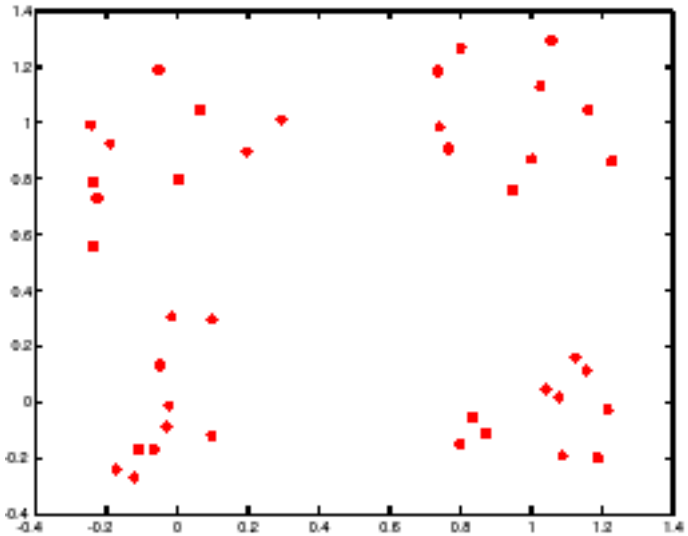
$$a^T A a$$

- But need the constraint

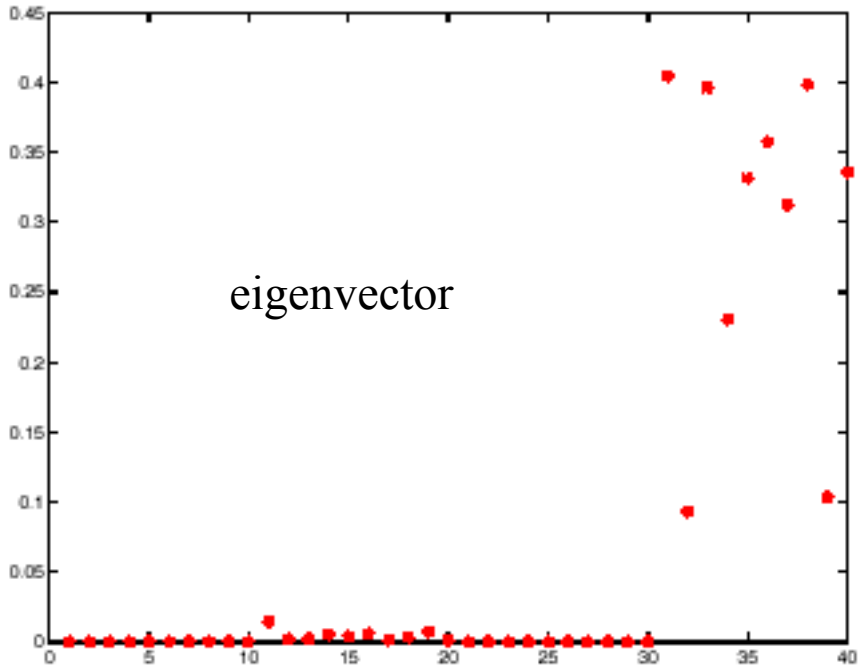
$$a^T a = 1$$

- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue

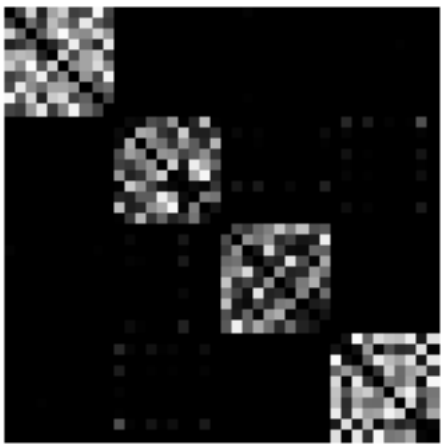
Example eigenvector



points



eigenvector



matrix

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

Clustering by eigenvectors: Algorithm

1. Construct (or take as input) the affinity matrix A
2. Compute the eigenvalues and eigenvectors of A
3. Repeat
4. Take the eigenvector corresponding to the largest unprocessed eigenvalue
5. Zero all components corresponding to elements that have already been clustered
6. Threshold the remaining components to determine which elements belong to this cluster
7. If all elements have been accounted for, there are sufficient clusters
8. Until there are sufficient clusters

Clustering as graph partitioning

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

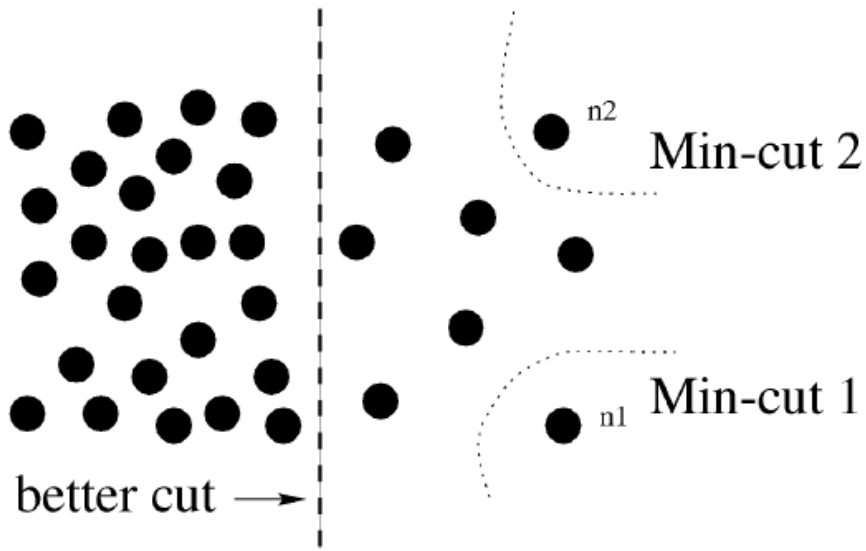


Fig. 1. A case where minimum cut gives a bad partition.

Normalized Cut

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

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

Cut vs Association

$$
\begin{aligned}
Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\
&= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} \\
&\quad + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\
&= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \right) \\
&= 2 - Nassoc(A,B).
\end{aligned}
$$

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

- Maximize

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

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

- Let **y** be a P = |V| dimensional vector where $y_i = \begin{cases} 1, & \text{if node } i \in A \\ -1, & \text{otherwise} \end{cases}$
- Let $d(i) = \sum_j w_{ij}$

  define the affinity of node *i* with all other nodes
- Let **D** = P x P diagonal matrix:

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ & & \dots & \\ 0 & 0 & \dots & d_P \end{bmatrix}$$

"degree matrix"

- Let **A** = P x P symmetric matrix:

"affinity matrix"

$$\mathbf{A} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1P} \\ w_{21} & w_{22} & \cdots & w_{2P} \\ & & \cdots & \\ w_{P1} & w_{P2} & \cdots & w_{PP} \end{bmatrix}$$

- It can be shown that

  $\mathbf{y} = \arg\min_{\mathbf{x}} ncut(\mathbf{x})$

  $$= \arg\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{A})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} \text{ subject to } \mathbf{y}^T\mathbf{D}\mathbf{1} = 0$$

- Relaxing the constraint on **y** so as to allow it to have real values means that we can **approximate the solution** by solving an equation of the form: $(\mathbf{D} - \mathbf{A})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$

- The solution, **y**, is an eigenvector of (**D** – **A**)

- An eigenvector is a characteristic vector of a matrix and specifies a segmentation based on the values of its components; similar points will hopefully have similar eigenvector components.

- Theorem: If **M** is any real, symmetric matrix and **x** is orthogonal to the j-1 smallest eigenvectors $\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}$, then $\mathbf{x}^T\mathbf{M}\mathbf{x} / \mathbf{x}^T\mathbf{x}$ is minimized by the next smallest eigenvector $\mathbf{x}_j$ and its minimum value is the eigenvalue λ_j

- Smallest eigenvector is always 0

  because $A=V$, $B=\{\}$ means $ncut(A,B)=0$
- Second smallest eigenvector is the real-valued **y** that minimizes *ncut*
- Third smallest eigenvector is the real-valued **y** that optimally sub-partitions the first two regions
- Etc.
- Note: Converting from the real-valued **y** to a binary-valued **y** introduces errors that will propagate to each sub-partition

Comments on the Algorithm

- Recursively bi-partitions the graph instead of using the 3^{rd}, 4^{th}, etc. eigenvectors for robustness reasons (due to errors caused by the binarization of the real-valued eigenvectors)
- Solving standard eigenvalue problems takes $O(P^3)$ time
- Can speed up algorithm by exploiting the "locality" of affinity measures, which implies that **A** is sparse (non-zero values only near the diagonal) and (**D** – **A**) is sparse. This leads to a $O(P\sqrt{P})$ time algorithm

Normalized cuts algorithm

1. Given an image or image sequence, set up a weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.
2. Solve $(\mathbf{D} - \mathbf{W})x = \lambda \mathbf{D}x$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
4. Decide if the current partition should be subdivided and recursively repartition the segmented parts if necessary.

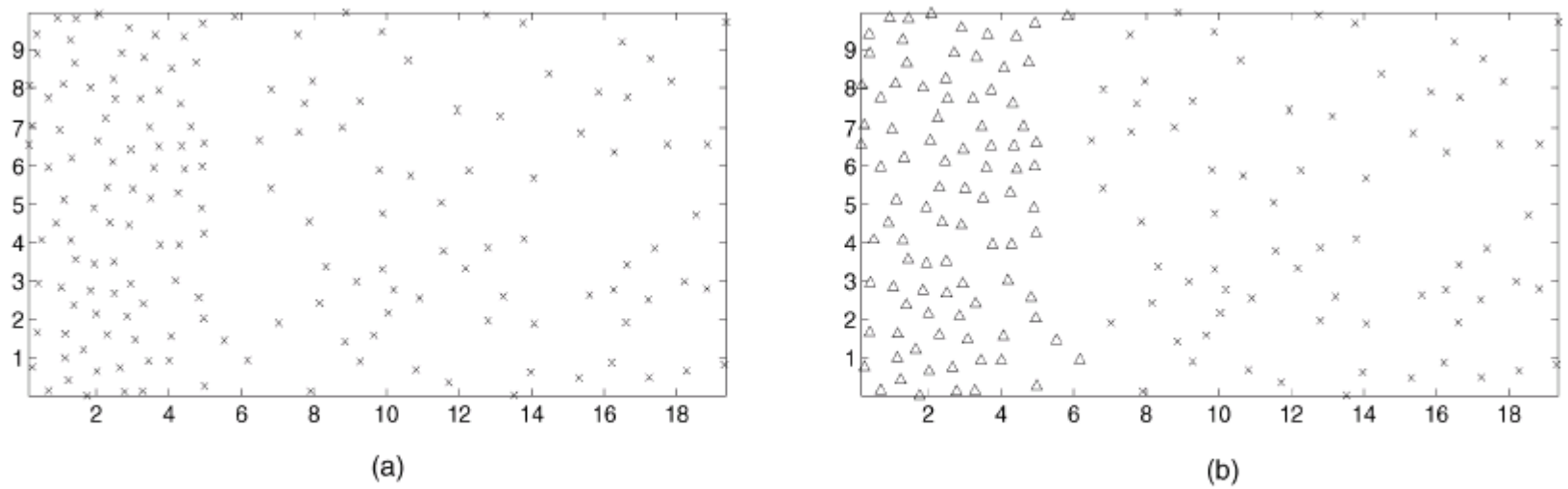


Fig. 5. (a) Point set generated by two Poisson processes, with densities of 2.5 and 1.0 on the left and right clusters respectively, (b) $\triangle$ and $\times$ indicate the partition of point set in (a). Parameter settings: $\sigma_X = 5$, $r = 3$.

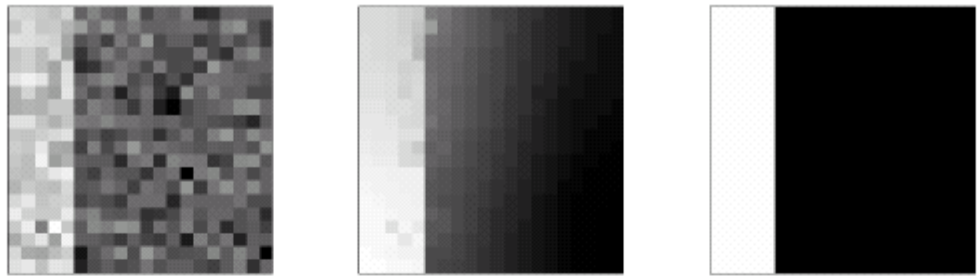


Fig. 6. (a) A synthetic image showing a noisy "step" image. Intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.2$ is added. Subplot (b) shows the eigenvector with the second smallest eigenvalue and subplot (c) shows the resulting partition.

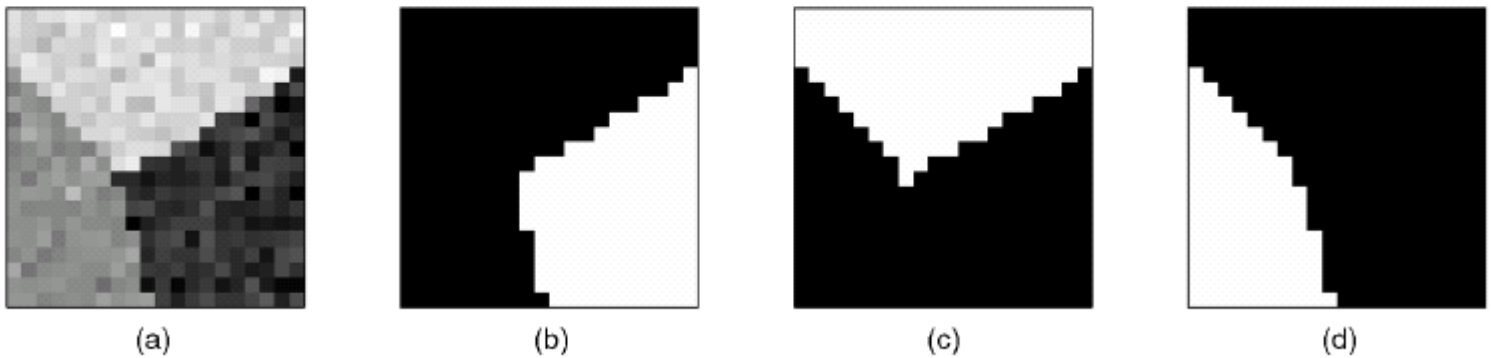


Fig. 7. (a) A synthetic image showing three image patches forming a junction. Image intensity varies from 0 to 1 and Gaussian noise with $\sigma = 0.1$ is added. (b)-(d) show the top three components of the partition.

Fig. 2. A gray level image of a baseball game.

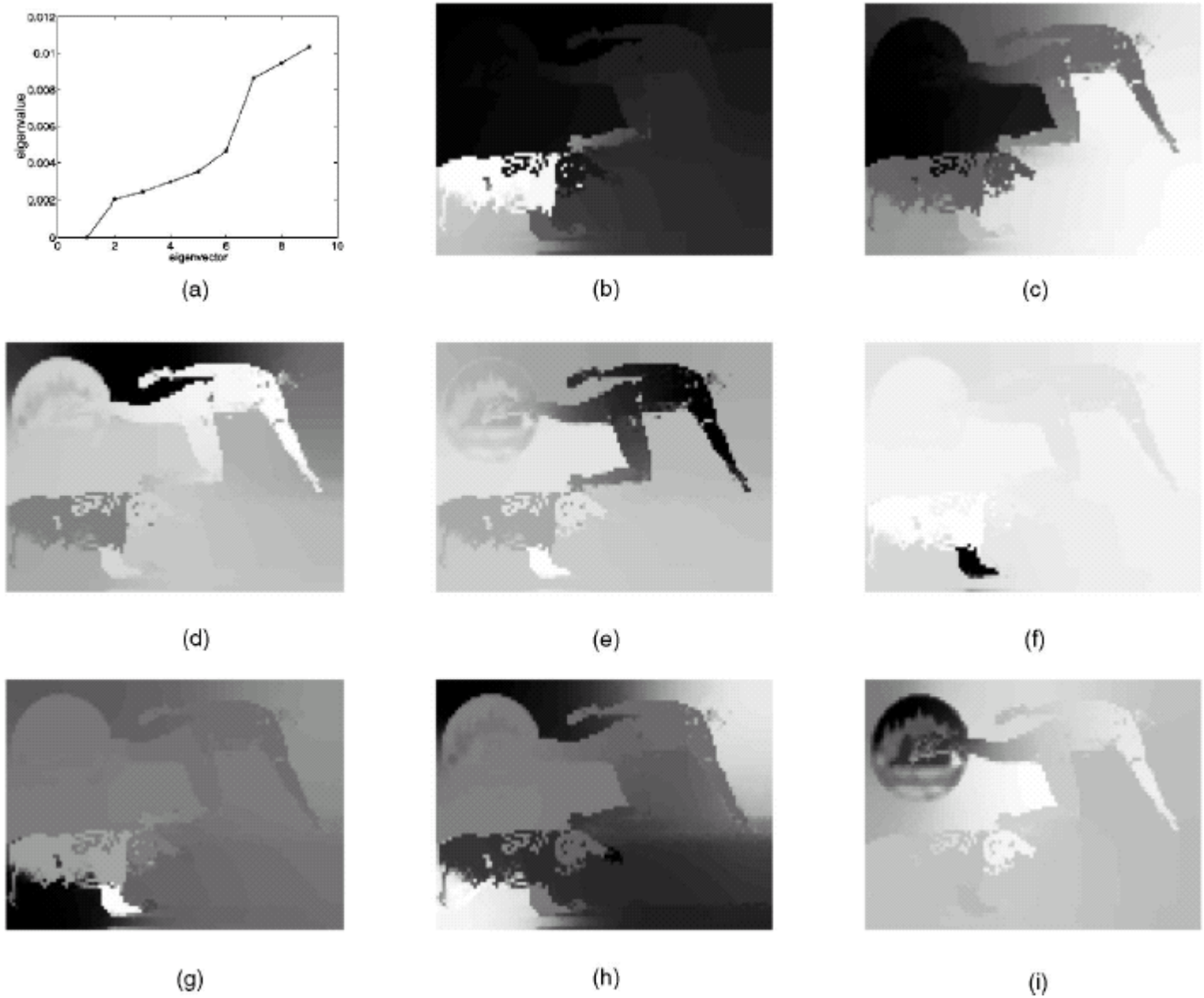


Fig. 3. Subplot (a) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplots (b)-(i) show the eigenvectors corresponding the second smallest to the ninth smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

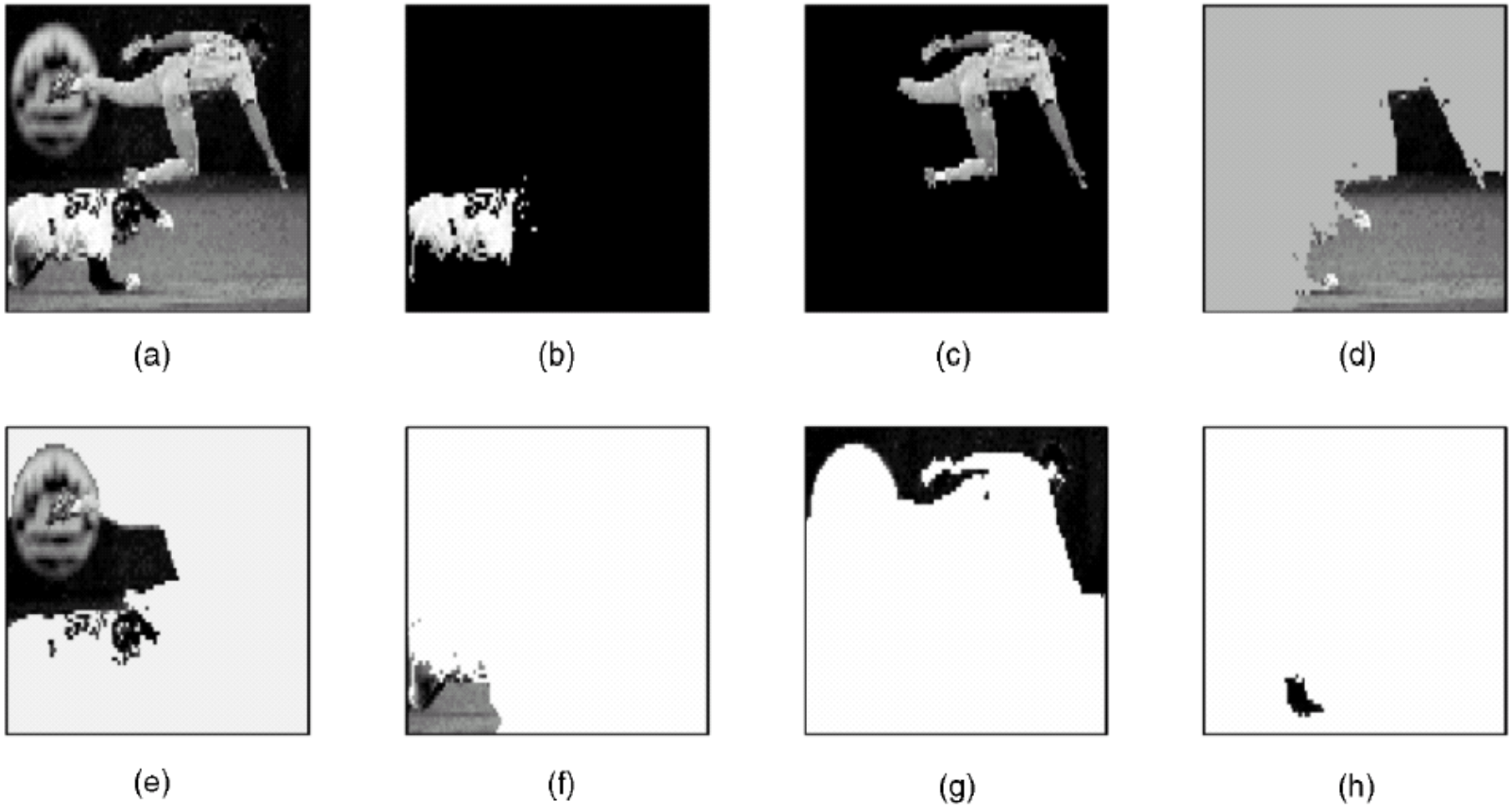


Fig. 4. (a) shows the original image of size 80×100. Image intensity is normalized to lie within 0 and 1. Subplots (b)-(h) show the components of the partition with $Ncut$ value less than 0.04. Parameter setting: $\sigma_I = 0.1$, $\sigma_X = 4.0$, $r = 5$.

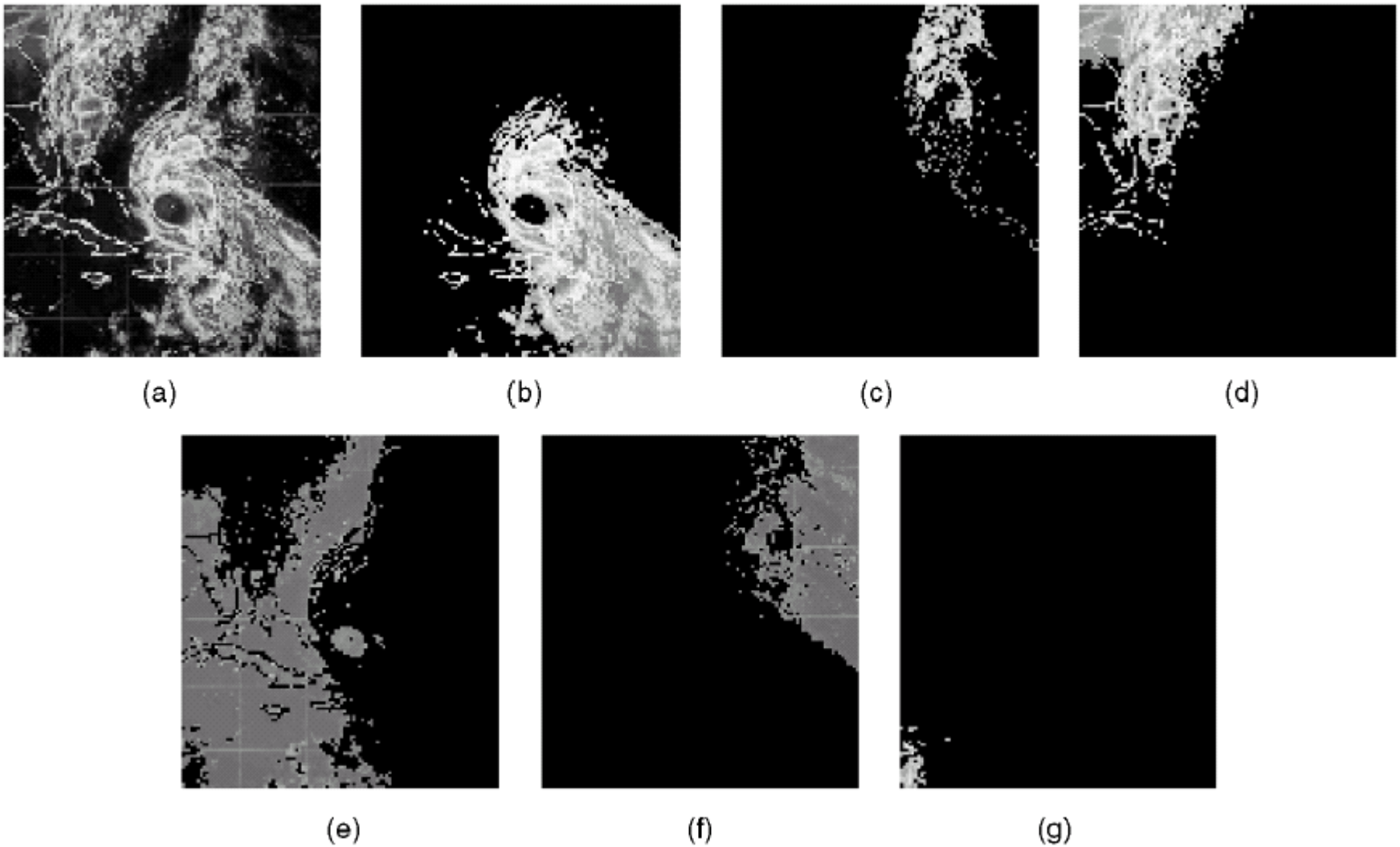


Fig. 8. (a) shows a 126×106 weather radar image. (b)-(g) show the components of the partition with $Ncut$ value less than 0.08. Parameter setting: $\sigma_I = 0.007$, $\sigma_x = 15.0$, $r = 10$.

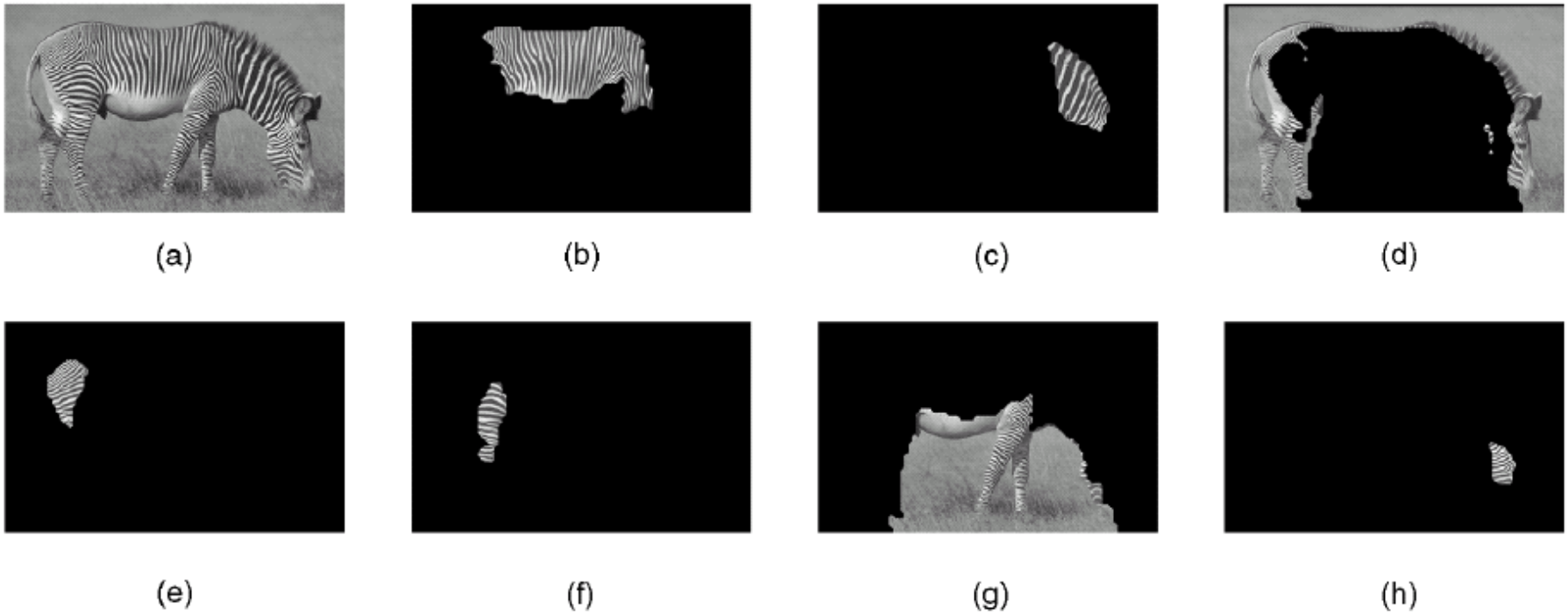


Fig. 10. (a) shows an image of a zebra. The remaining images show the major components of the partition. The texture features used correspond to convolutions with DOOG filters [16] at six orientations and five scales.

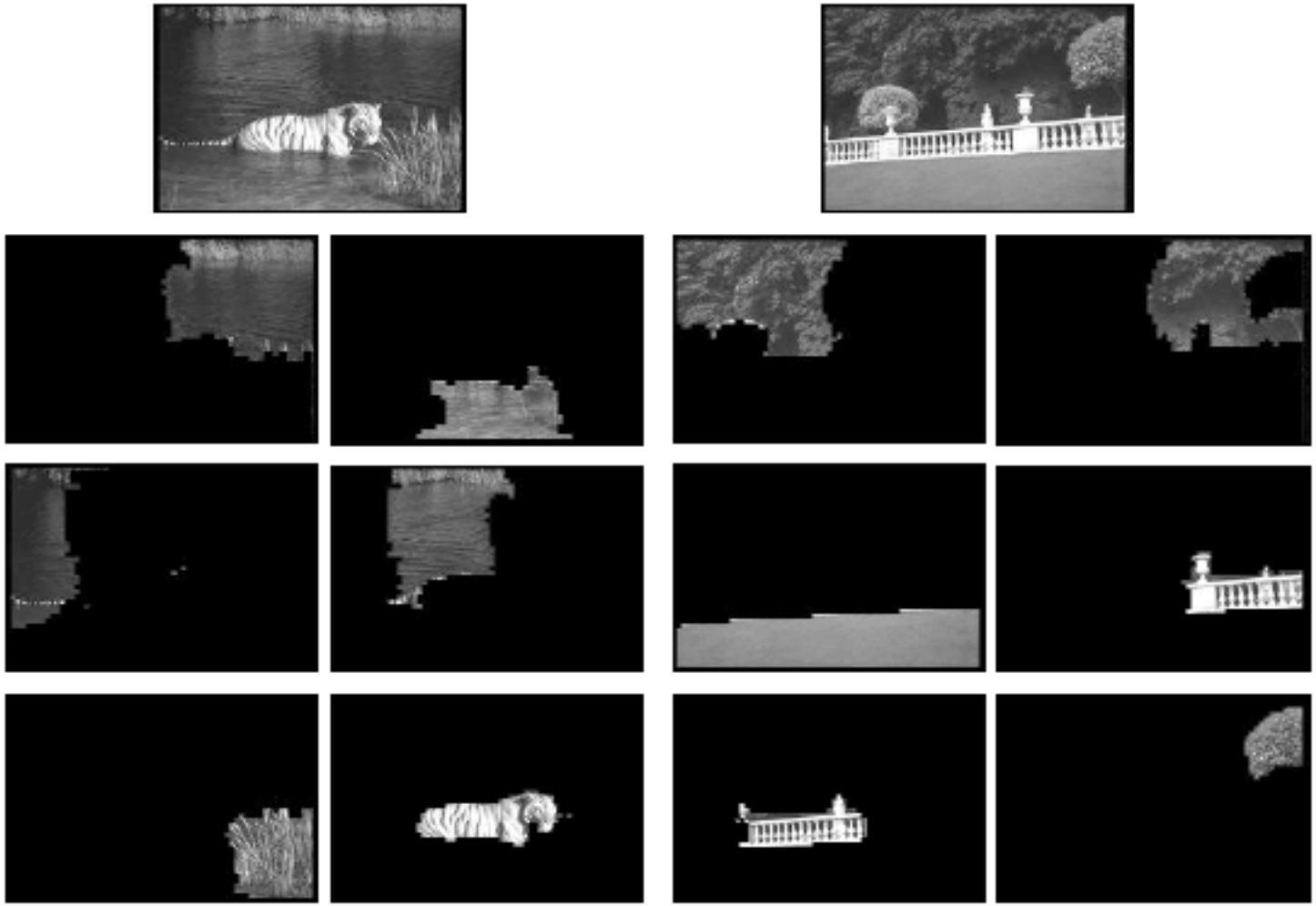


Figure from “Image and video segmentation: the normalised cut framework”, by Shi and Malik, copyright IEEE, 1998

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