6.801/866

Segmentation and Line Fitting

T. Darrell

Segmentation and Line Fitting

- Gestalt grouping
- Background subtraction
- K-Means
- Graph cuts
- Hough transform
- Iterative fitting

(Next time: Probabilistic segmentation)

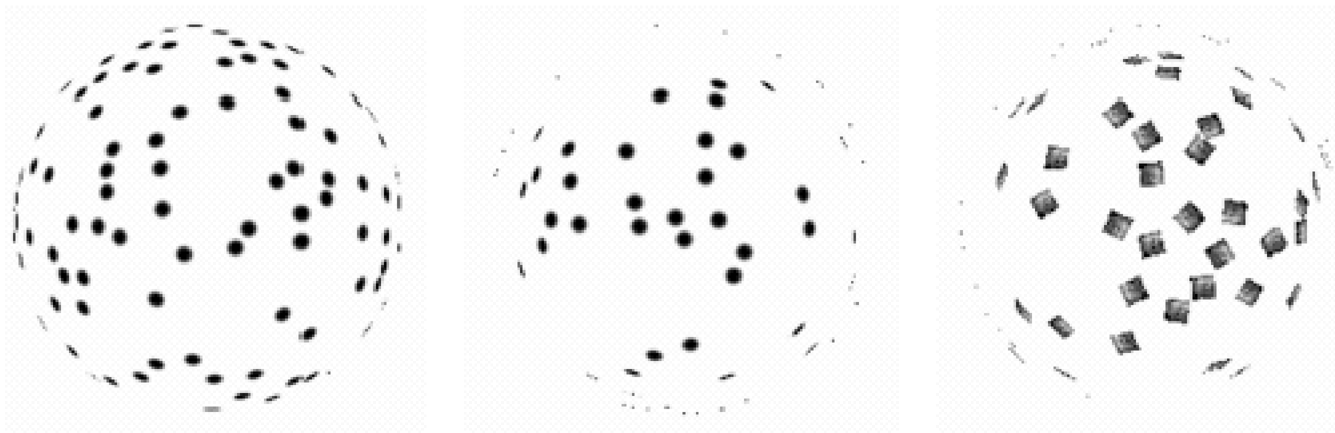
Segmentation and Grouping

- Motivation: vision is often simple inference, but for segmentation
- Obtain a compact representation from an image/motion sequence/set of tokens
- Should support application
- Broad theory is absent at present

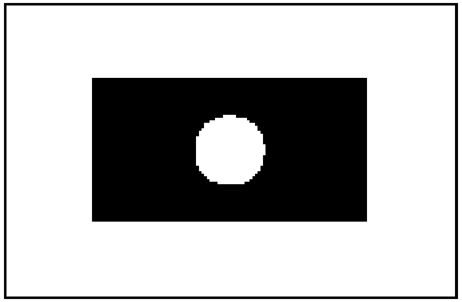
- Grouping (or clustering)
  - collect together tokens that "belong together"
- Fitting
  - associate a model with tokens
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?

General ideas

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



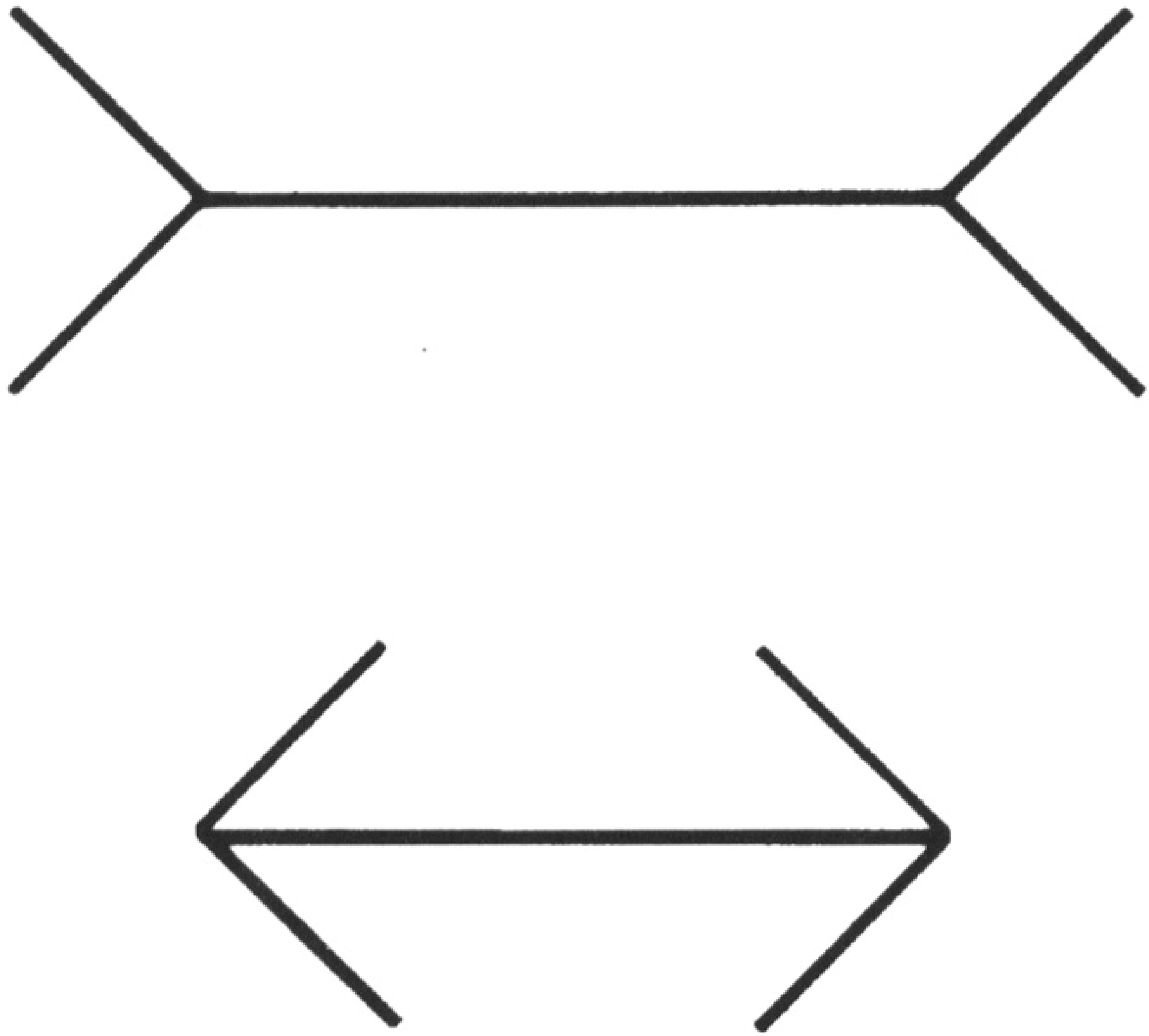
Why do these tokens belong together?

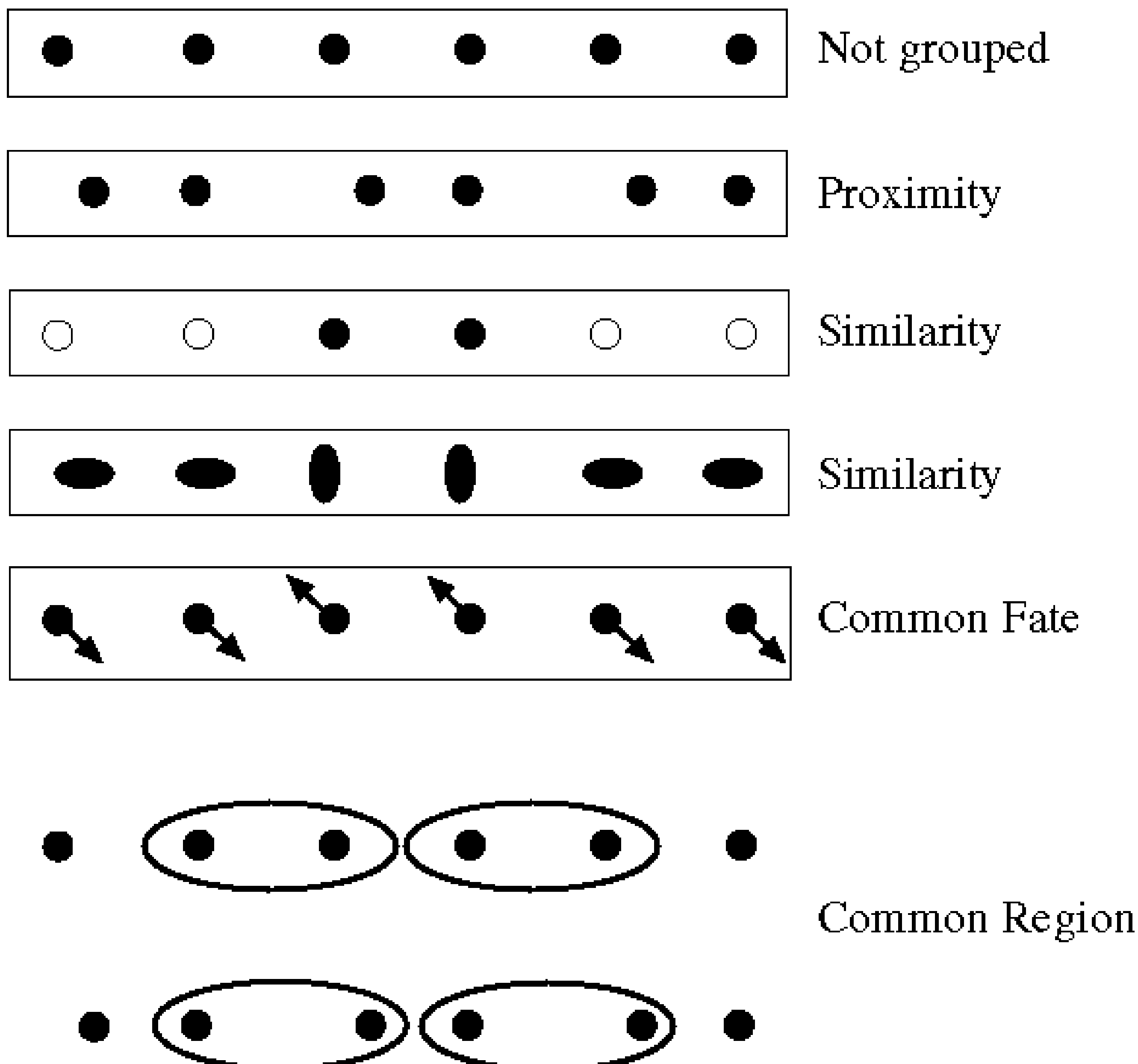


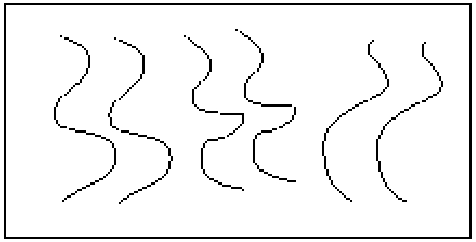
What is the figure?

Basic ideas of grouping in humans

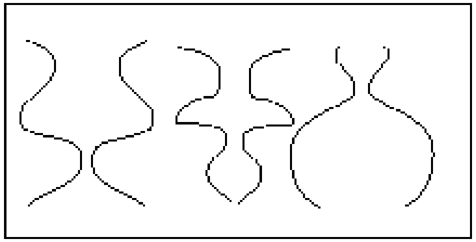
- Figure-ground discrimination
  - grouping can be seen in terms of allocating some elements to a figure, some to ground
  - impoverished theory
- Gestalt properties
  - elements in a collection of elements can have properties that result from relationships (Muller-Lyer effect)
    - gestaltqualitat
  - A series of factors affect whether elements should be grouped together
    - Gestalt factors



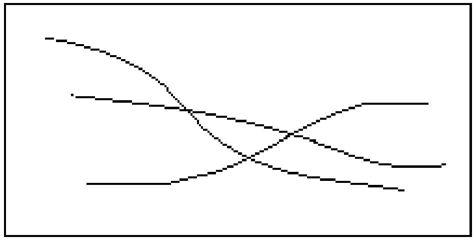

Not grouped
Proximity
Similarity
Similarity
Common Fate
Common Region



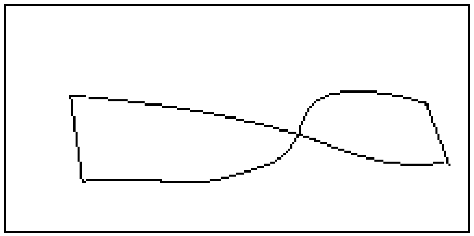
Parallelism



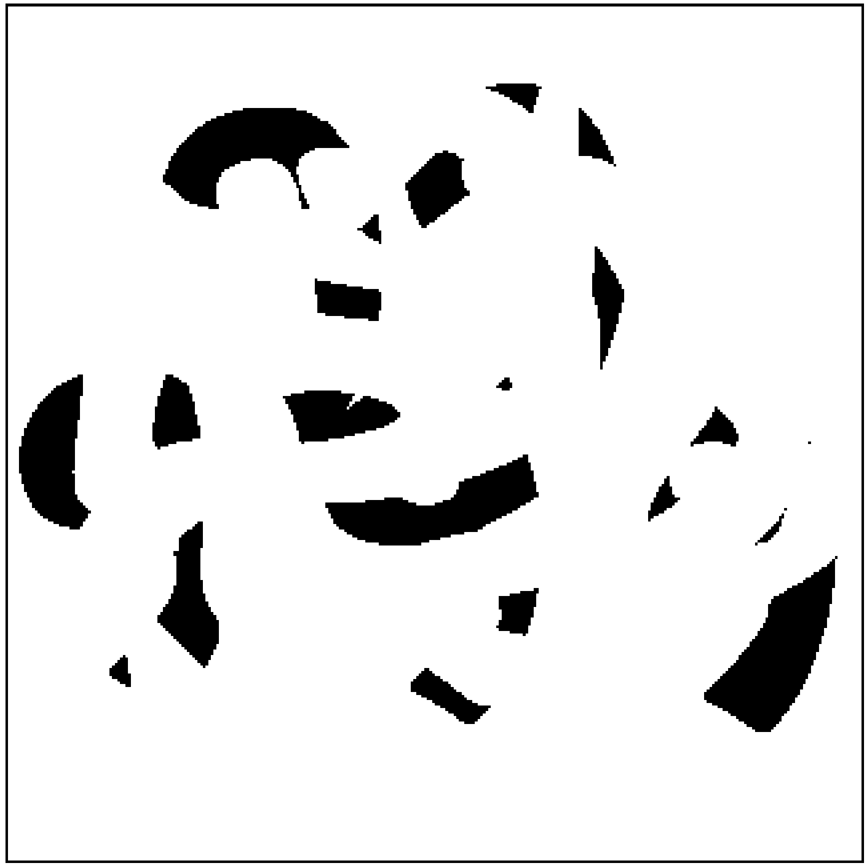
Symmetry



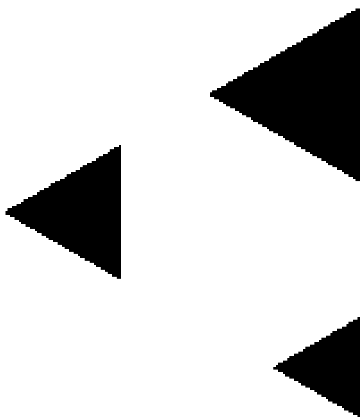
Continuity

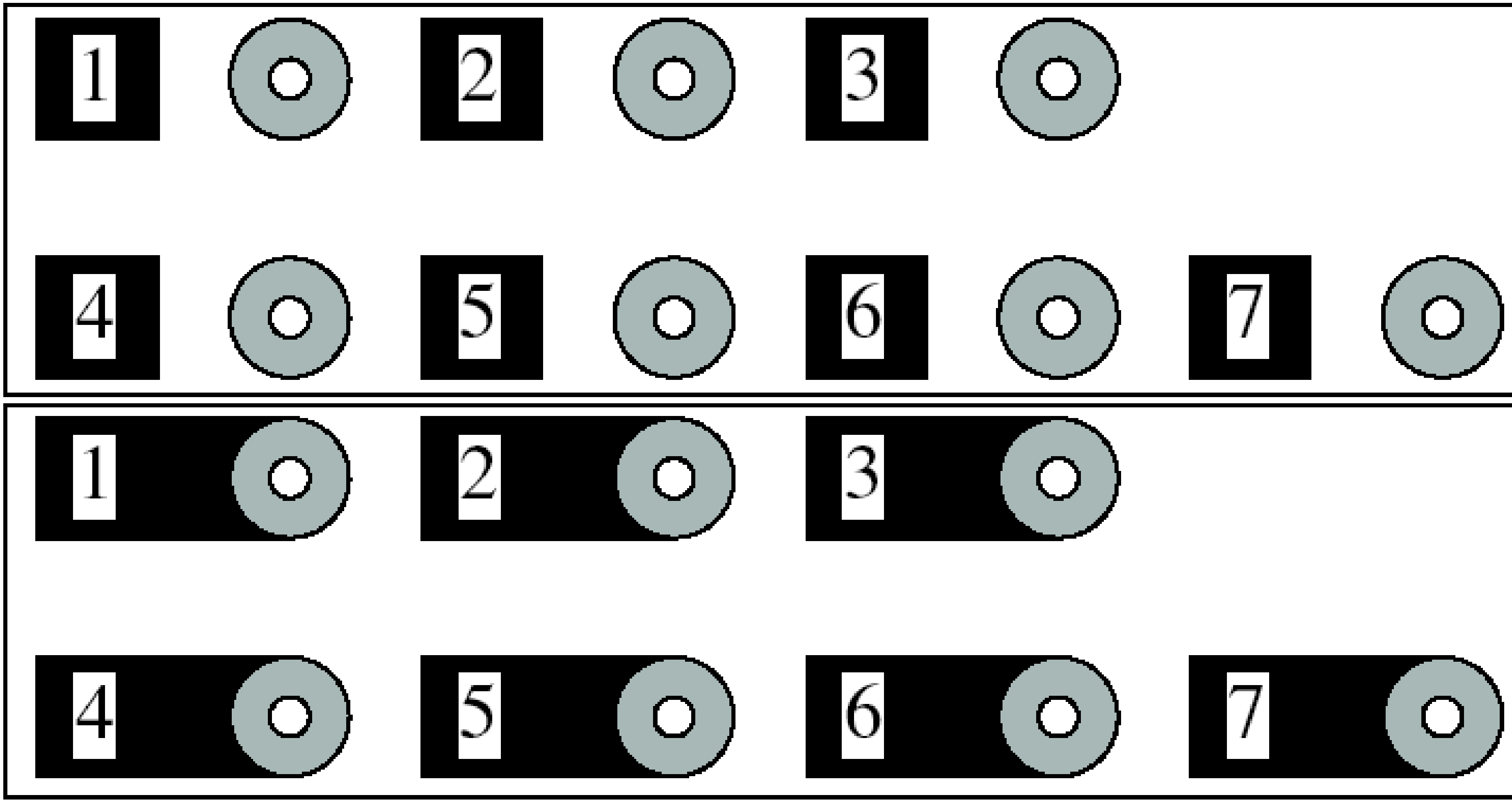


Closure



Occlusion is an important cue in grouping.

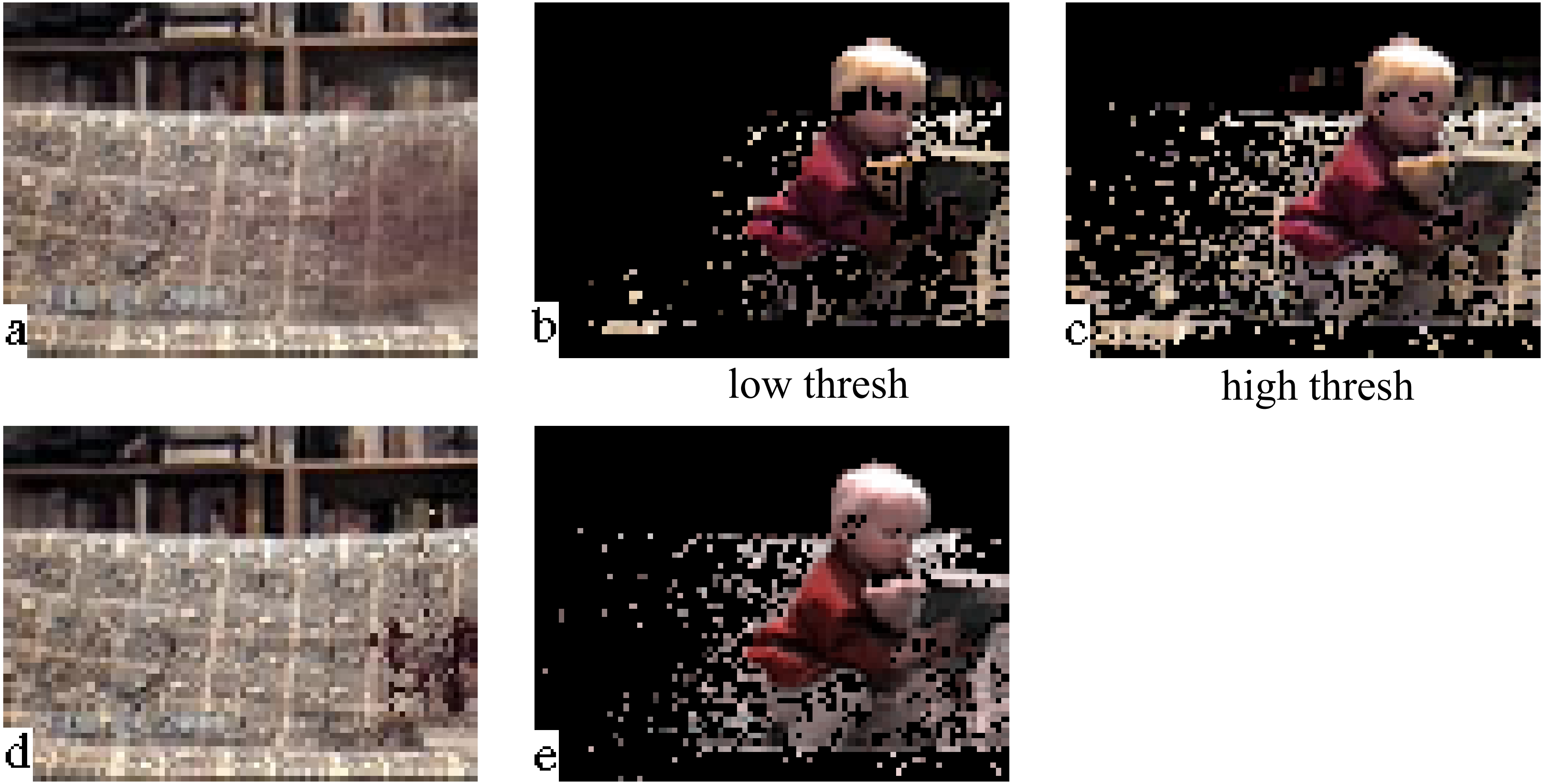



1
2
3
4
5
6
7
1
2
3
4
5
6
7

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

80x60

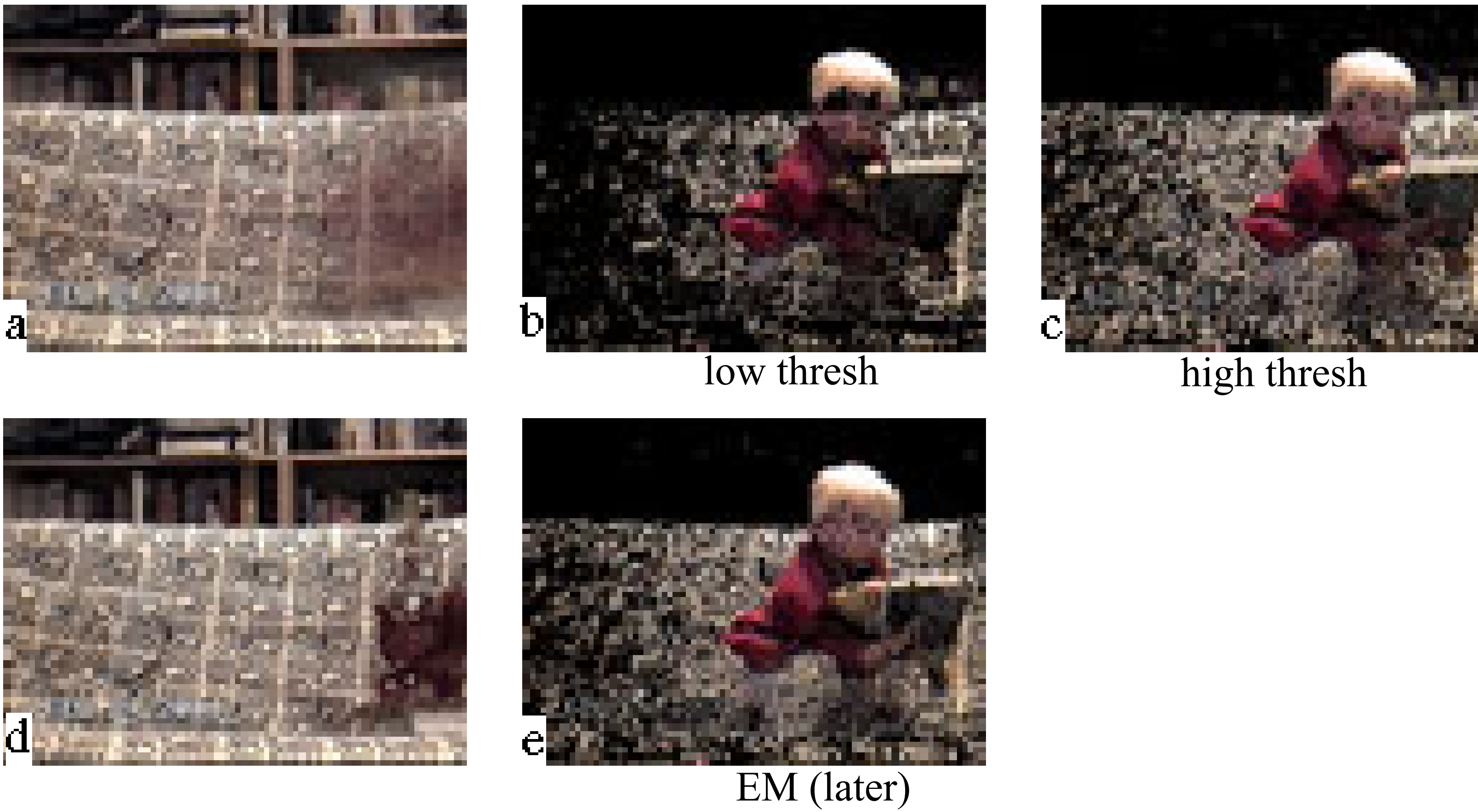


low thresh

high thresh

EM (later)

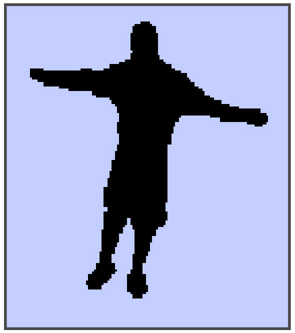
160x120



Classic Background Subtraction model

- Background is assumed to be mostly static
- Each pixel is modeled as by a gaussian distribution in YUV space
- Model mean is usually updated using a recursive low-pass filter

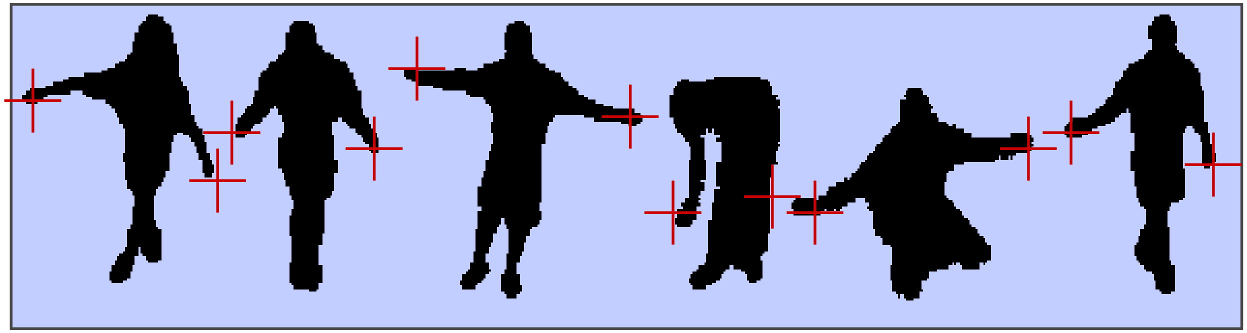
Given new image, generate silhouette
by marking those pixels that are significantly
different from the "background" value.



Finding Features

2D Head / hands localization

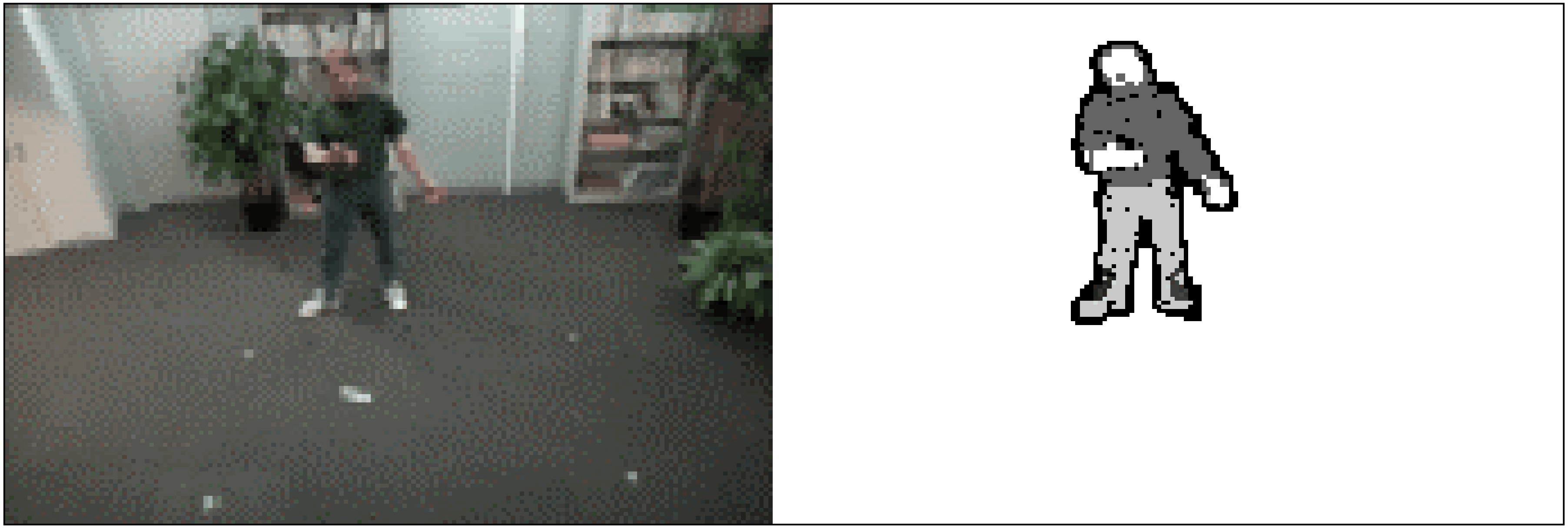
- contour analysis: mark extremal points (highest curvature or distance from center of body) as hand features
- use skin color model when region of hand or face is found (color model is independent of flesh tone intensity)



Static Background Modeling Examples

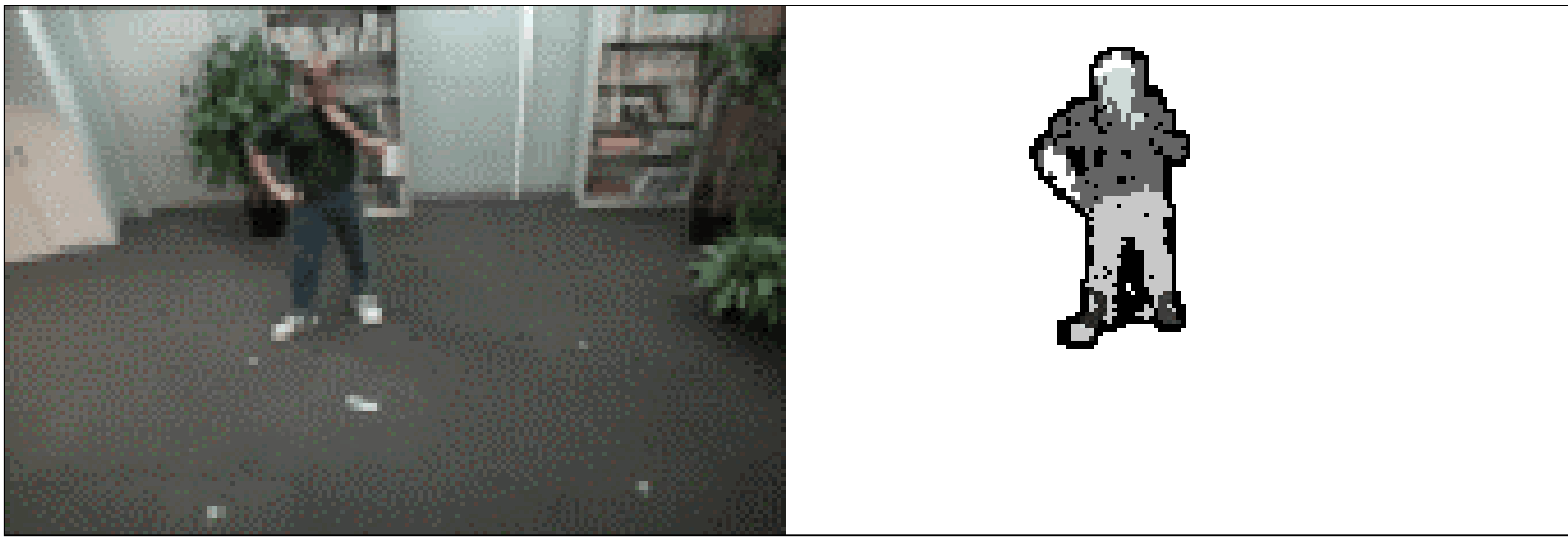
[MIT Media Lab Pfinder / ALIVE System]

Static Background Modeling Examples



[MIT Media Lab Pfinder / ALIVE System]

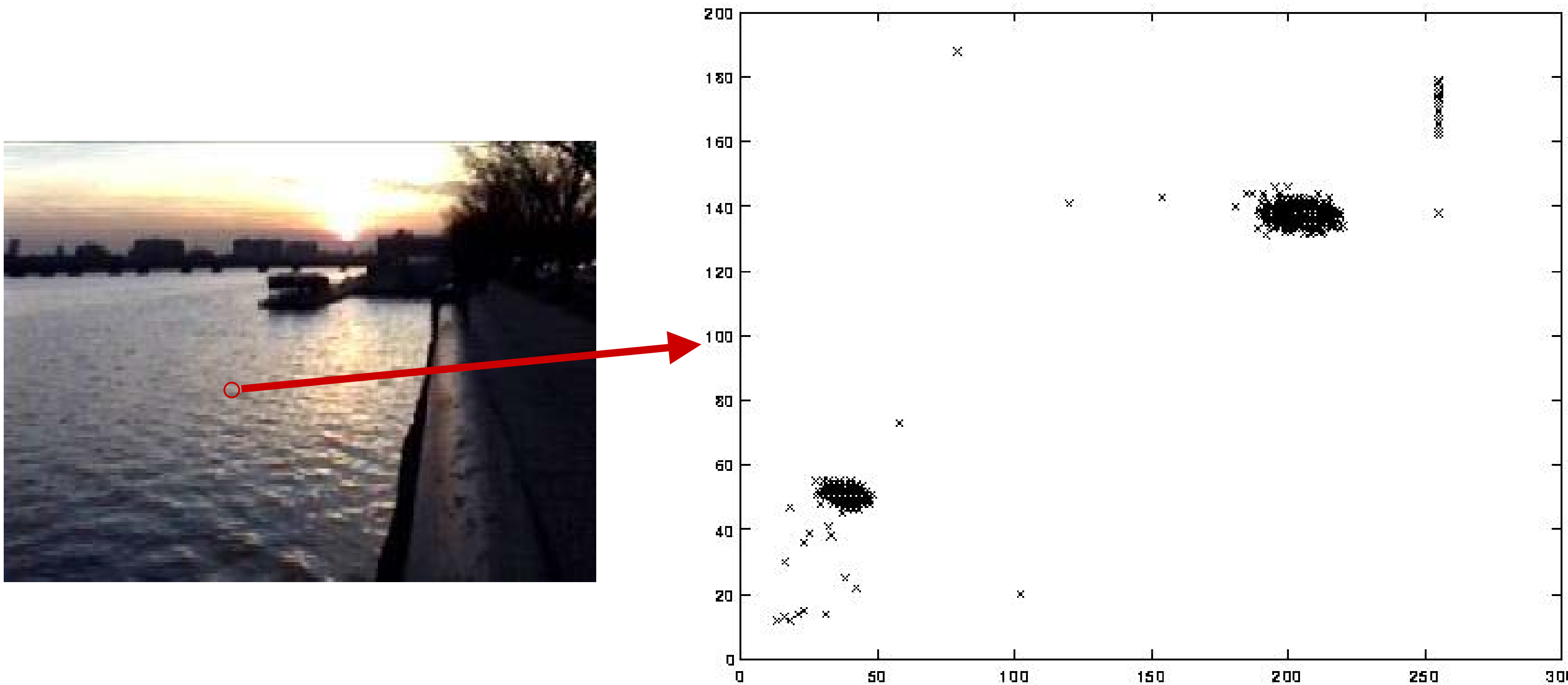
Static Background Modeling Examples



[MIT Media Lab Pfinder / ALIVE System]

Dynamic Background

BG Pixel distribution is non-stationary:

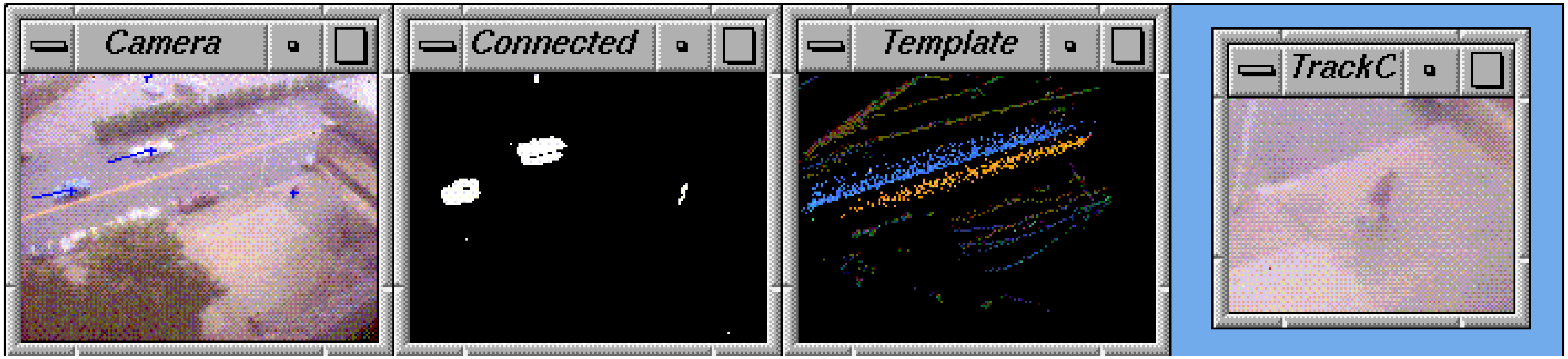


[MIT AI Lab VSAM]

Mixture of Gaussian BG model

Staufer and Grimson tracker:

Fit per-pixel mixture model to observed distrubution.



[MIT AI Lab VSAM]

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

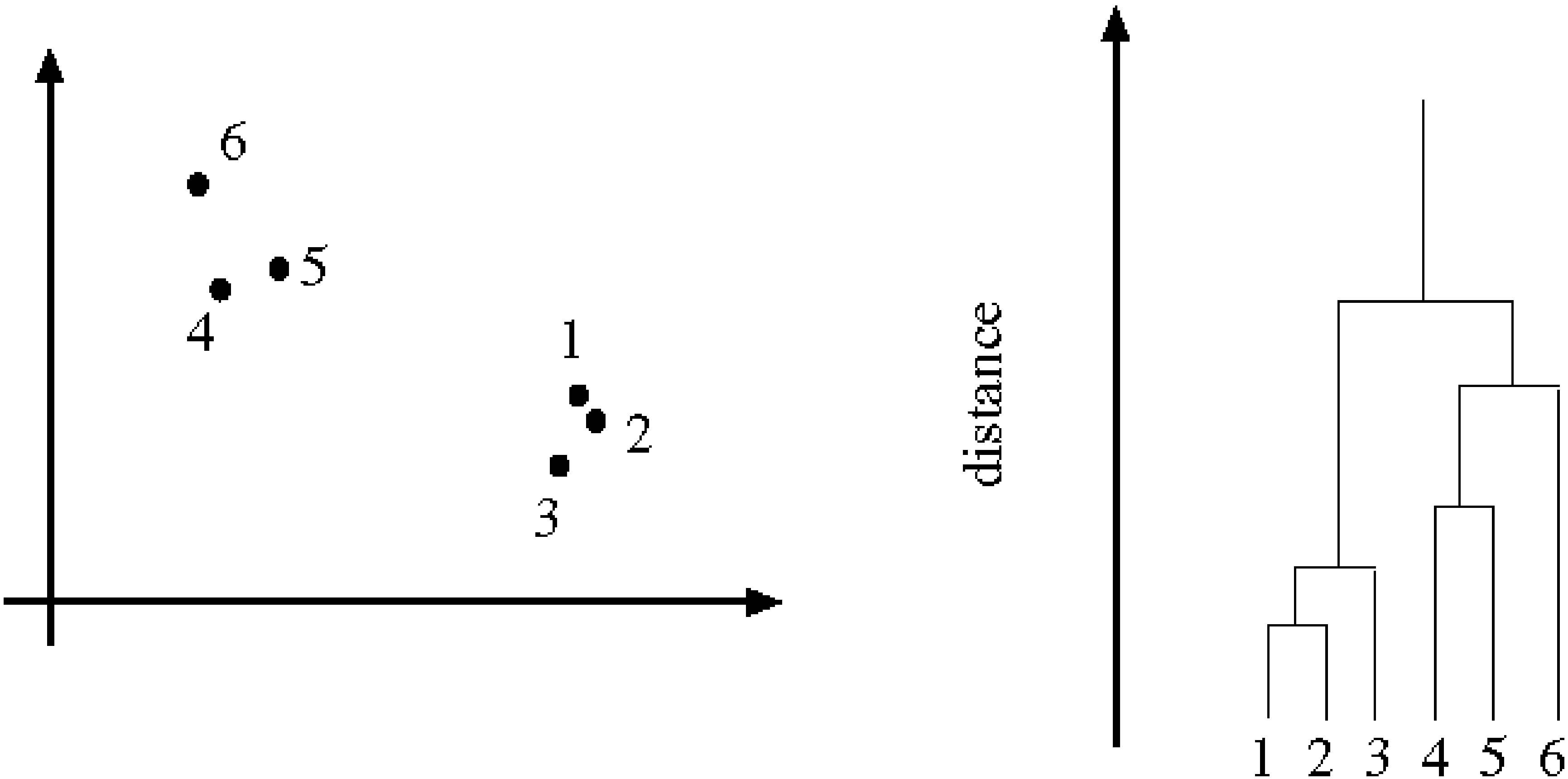
Clustering Algorithms

Algorithm 15.3: Agglomerative clustering, or clustering by merging

```
Make each point a separate cluster
Until the clustering is satisfactory
     Merge the two clusters with the
        smallest inter-cluster distance
end
```

Algorithm 15.4: Divisive clustering, or clustering by splitting

```
Construct a single cluster containing all points
Until the clustering is satisfactory
     Split the cluster that yields the two
        components with the largest inter-cluster distance
end
```


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

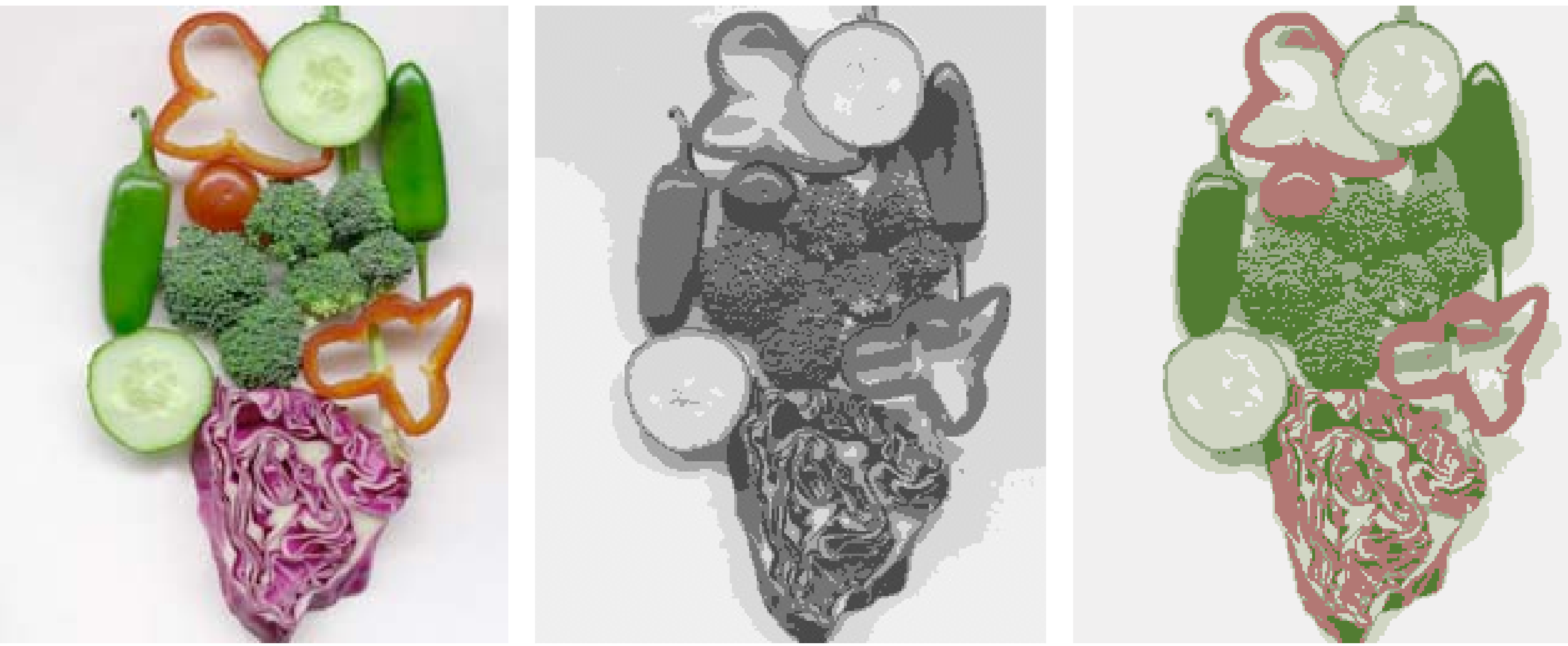
Algorithm 15.5: Clustering by K-Means

Choose k data points to act as cluster centers
Until the cluster centers are unchanged
    Allocate each data point to cluster whose center is nearest
    Now 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 center.
    Replace the cluster centers with the mean of the elements
        in their clusters.
end

Image

Clusters on intensity (K=5)

Clusters on color (K=5)



K-means clustering using intensity alone and color alone

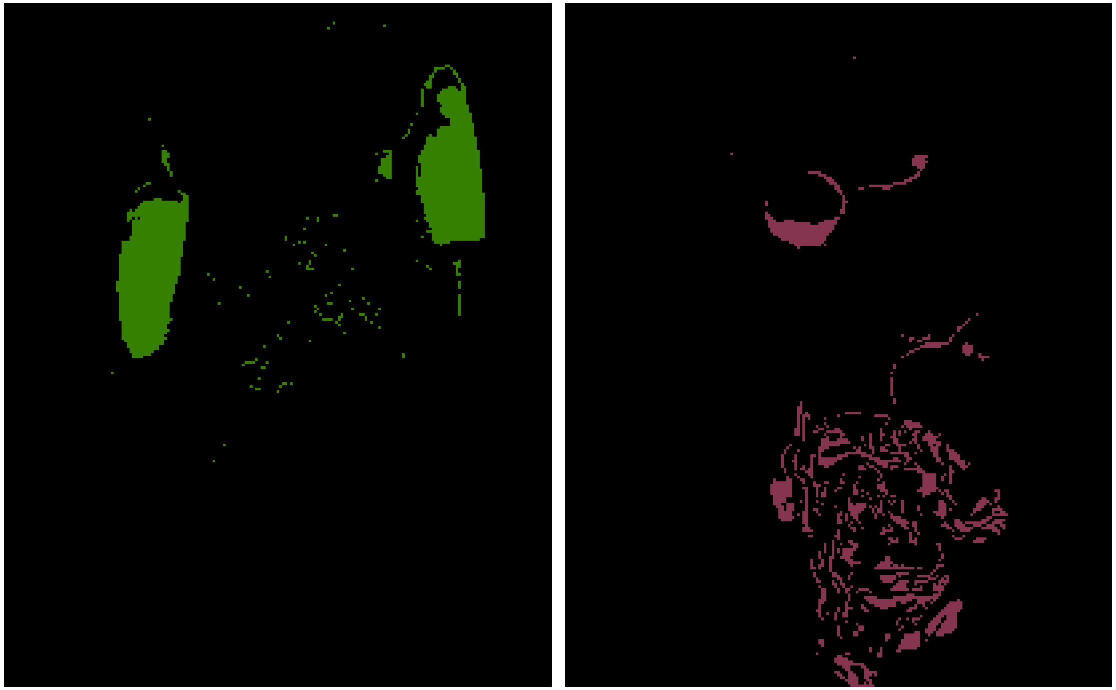
Image

Clusters on color

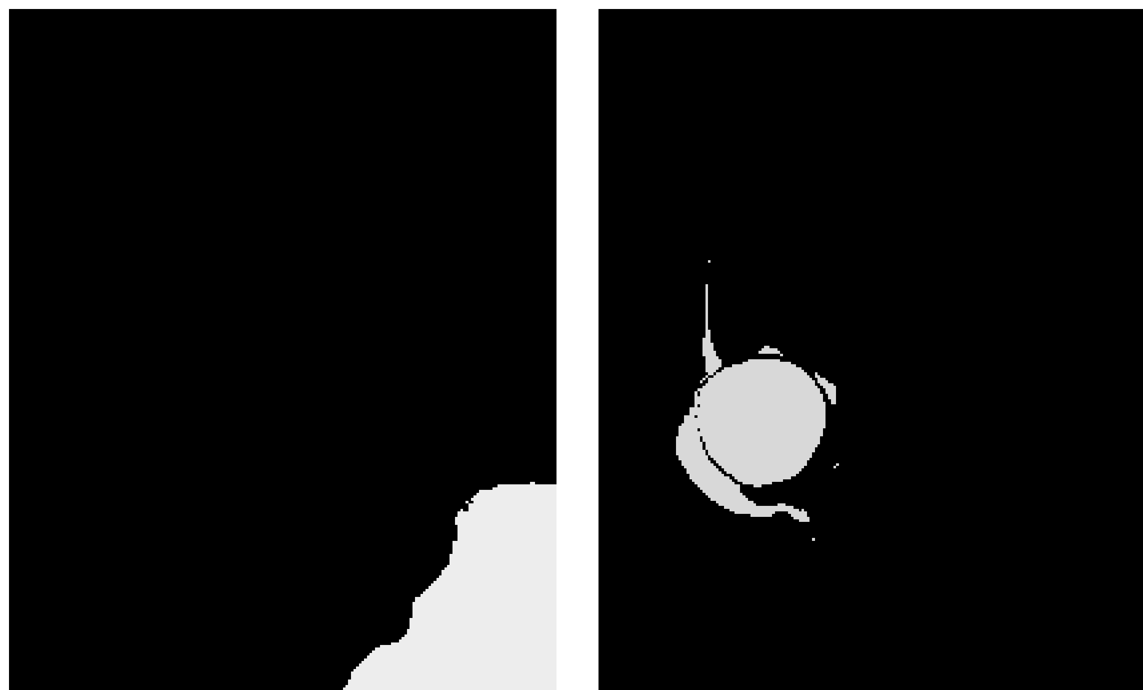
K-means using color alone, 11 segments

K-means using
color alone,
11 segments.

Oversegmentation!



K-means using colour and position, 20 segments

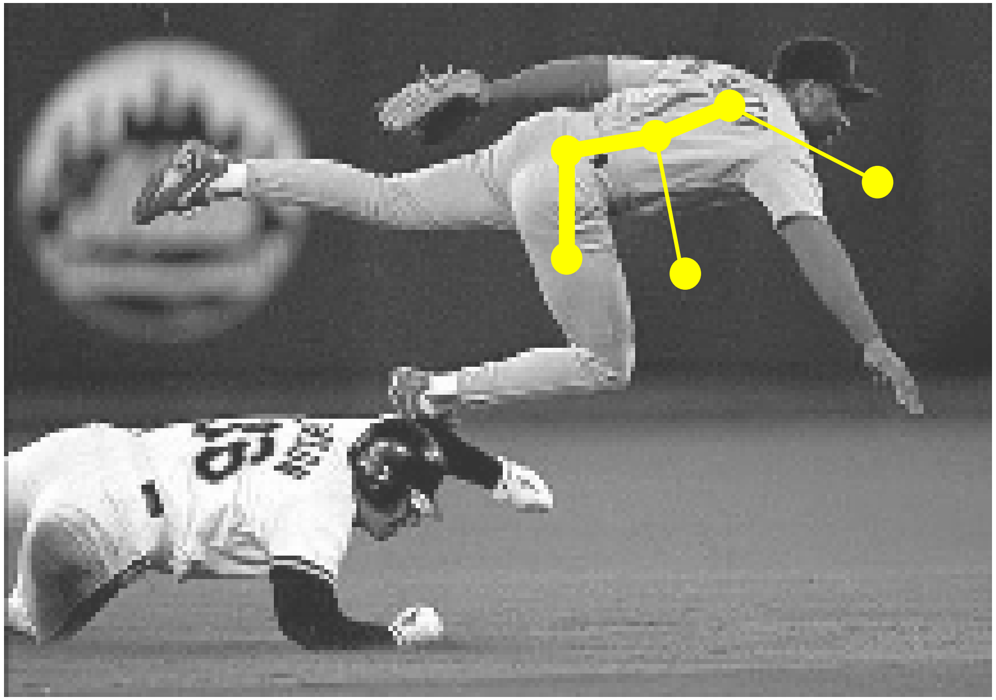


Still misses goal of perceptually pleasing segmentation!

Graph theoretic clustering

- Avoid local minima; use global criteria
- Represent tokens using a weighted graph.
  - affinity matrix
- Cut up this graph to get subgraphs with strong interior links

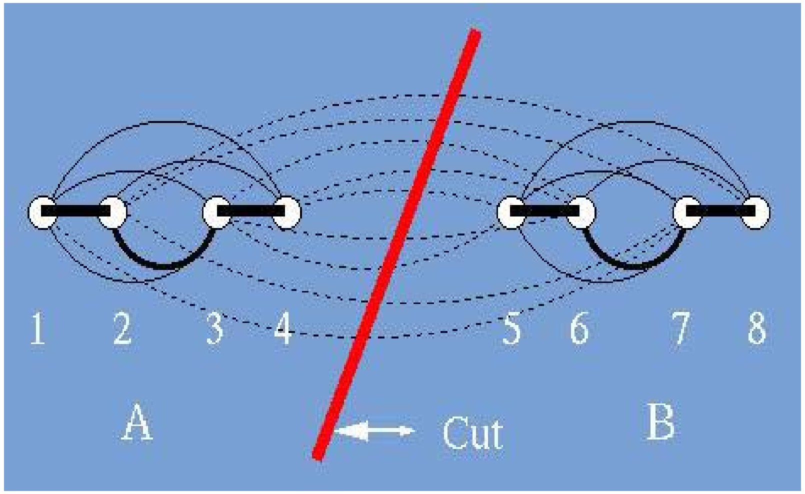
Image Segmentation as Graph Partitioning



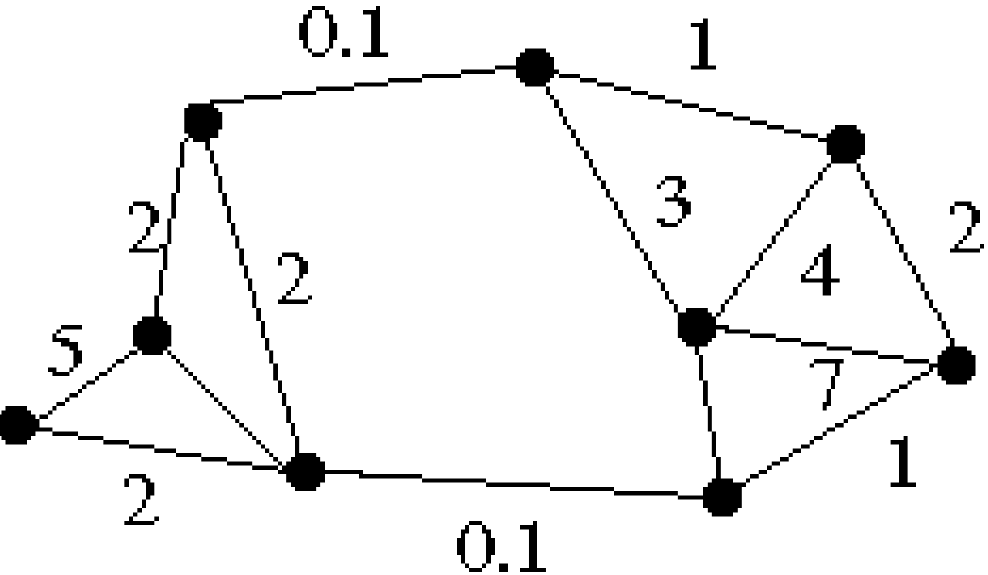
[Malik]

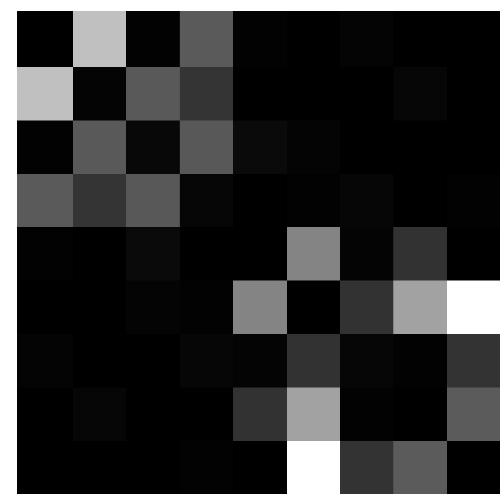
Some Terminology for Graph Partitioning

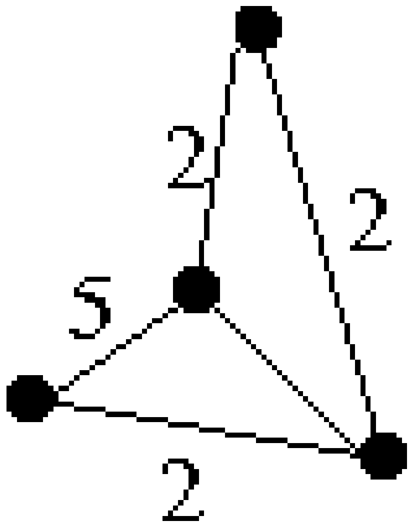
- How do we bipartition a graph:

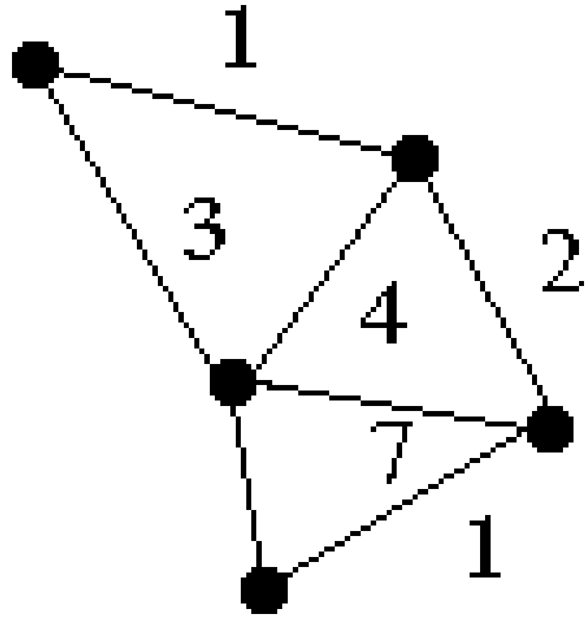


[Malik]


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




2
2
5
2


1
3
4
2
7
1

Boundaries of image regions defined by a number of attributes

- Brightness/color
- Texture
- Motion
- Stereoscopic depth
- Familiar configuration

[Malik]

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

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

Color

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

Eigenvectors and cuts

- 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

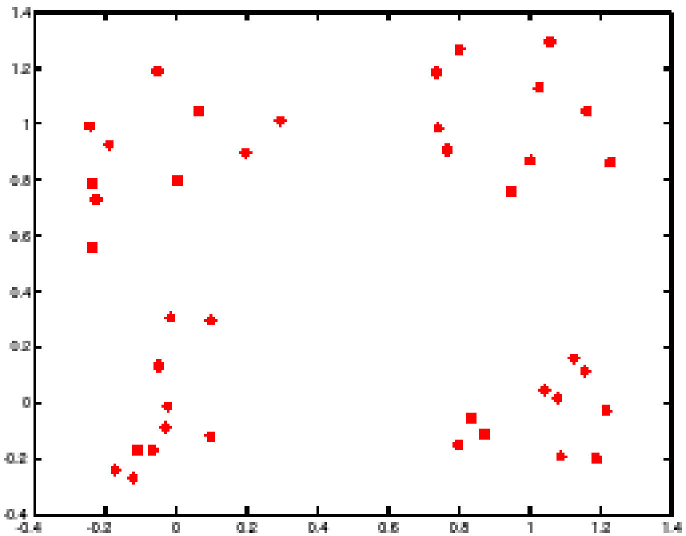
$$a^T A a$$

- But need the constraint

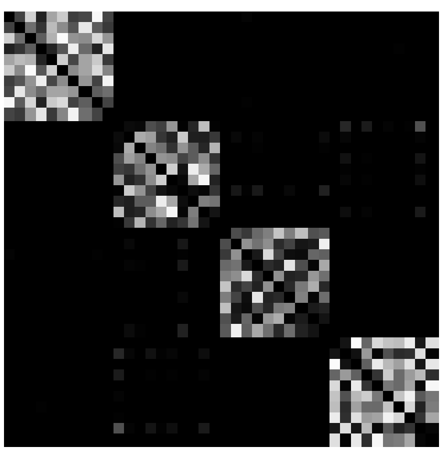
$$a^T a = 1$$

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

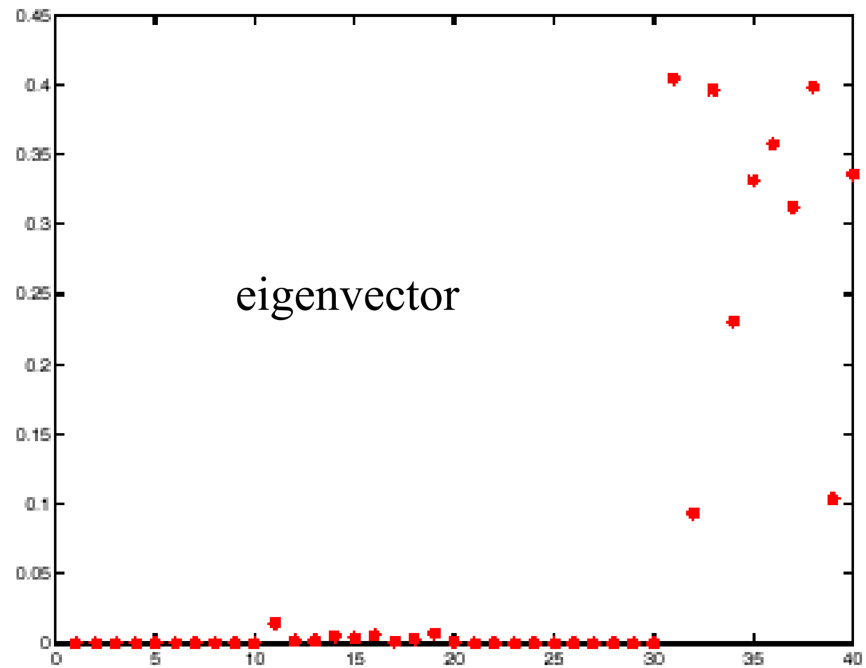
Example eigenvector



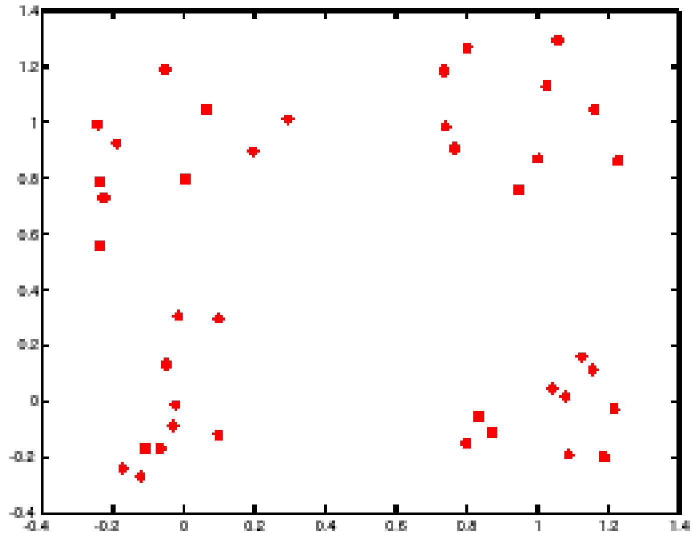
points



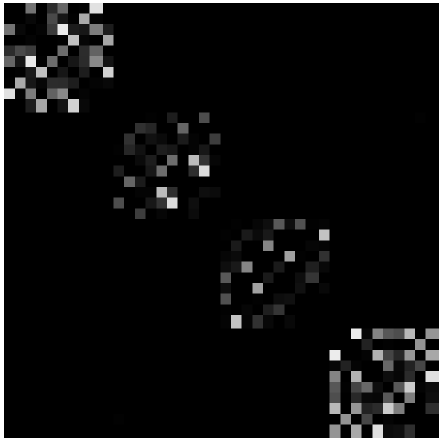
matrix



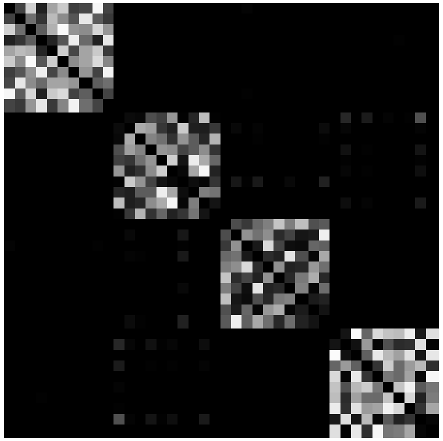
Scale affects affinity



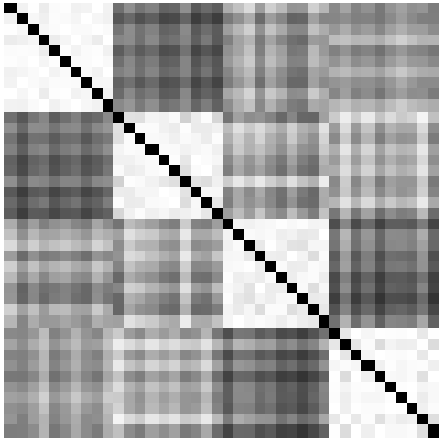
$\sigma = .2$



$\sigma = .1$

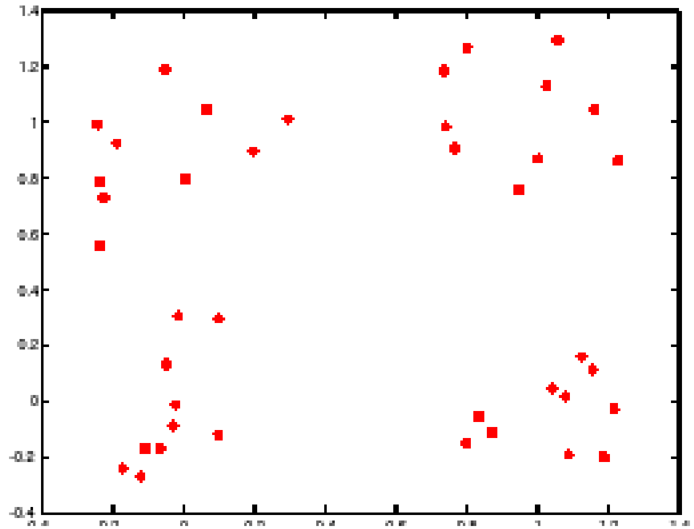


$\sigma = .2$

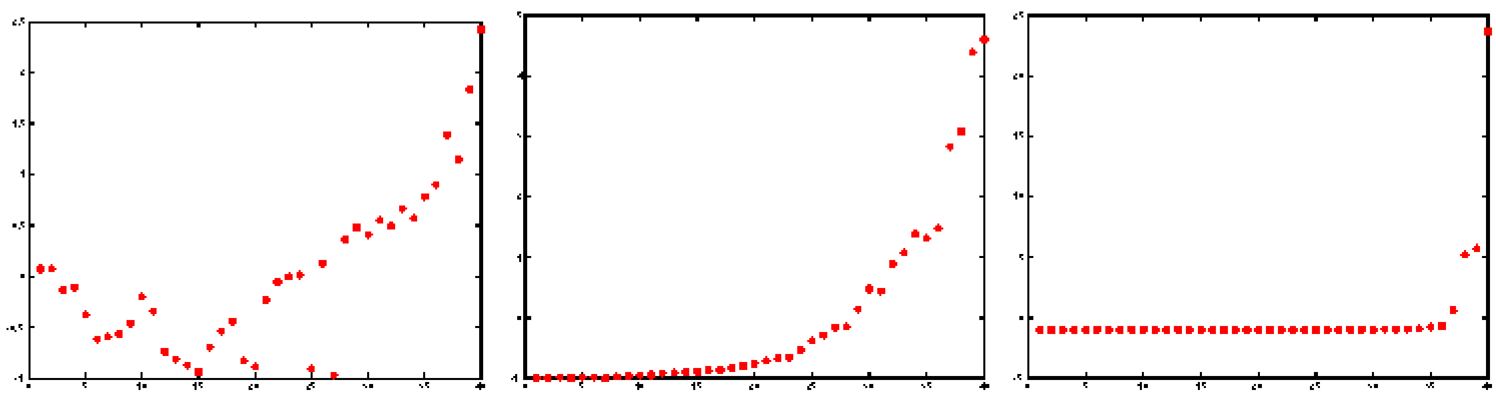


$\sigma = 1$

Scale affects affinity



σ=.2



σ=.1 σ=.2 σ=1

FIGURE 15.21: The number of clusters is reflected in the eigenvalues of the affinity 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

More than two segments

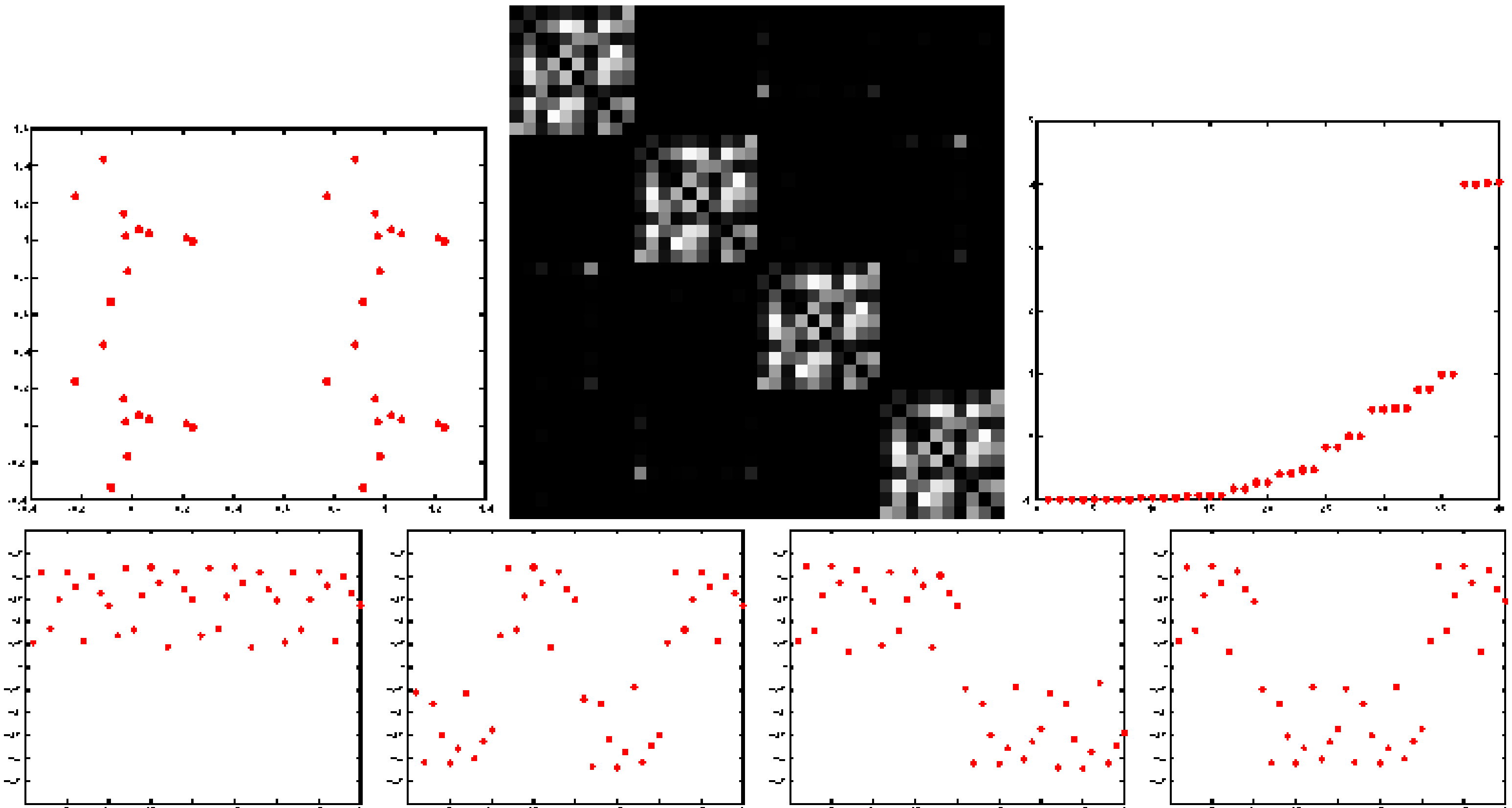


FIGURE 15.22: Eigenvectors of an affinity matrix can be a misleading guide to clusters.

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

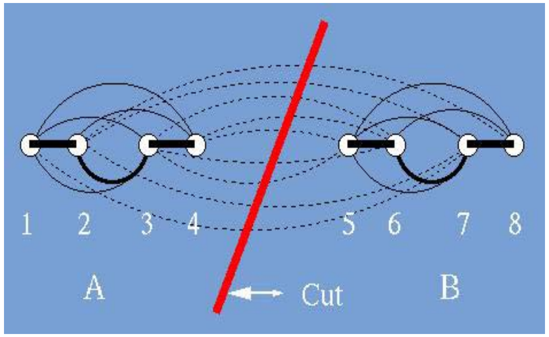
- Maximize

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

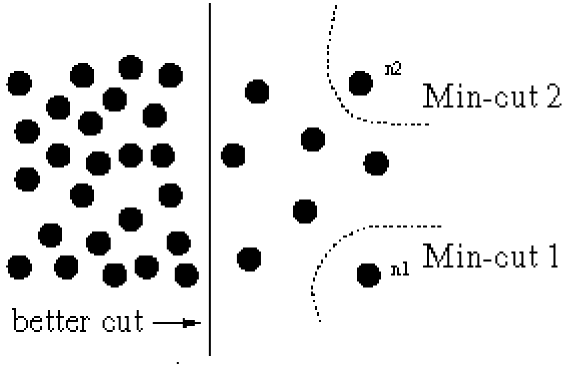
where cut(A,B) is sum of weights that straddle A,B; assoc(A,V) is sum of all edges with one end in A.

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

Normalized Cut



- Minimum cut (total weight of edges removed) is not appropriate since it favors cutting small pieces.



- Normalized Cut, *Ncut*:

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

[Malik]

Solving the Normalized Cut problem

- Exact discrete solution to Ncut is NP-complete even on regular grid,
  - [Papadimitriou'97]
- Drawing on spectral graph theory, good approximation can be obtained by solving a generalized eigenvalue problem.

[Malik]

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.

$$D_{ii} = \sum_{i} W_{ij}$$

- Criterion becomes

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

and we have a constraint

$$y^T D 1 = 0$$

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

Normalized Cut As Generalized Eigenvalue problem

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

$$= \frac{(1+x)^T(D-W)(1+x)}{k1^T D1} + \frac{(1-x)^T(D-W)(1-x)}{(1-k)1^T D1};\ k = \frac{\sum_{x_i>0} D(i,i)}{\sum_i D(i,i)}$$

$$= \ldots$$

- after simplification, we get

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

[Malik]

Normalized cuts

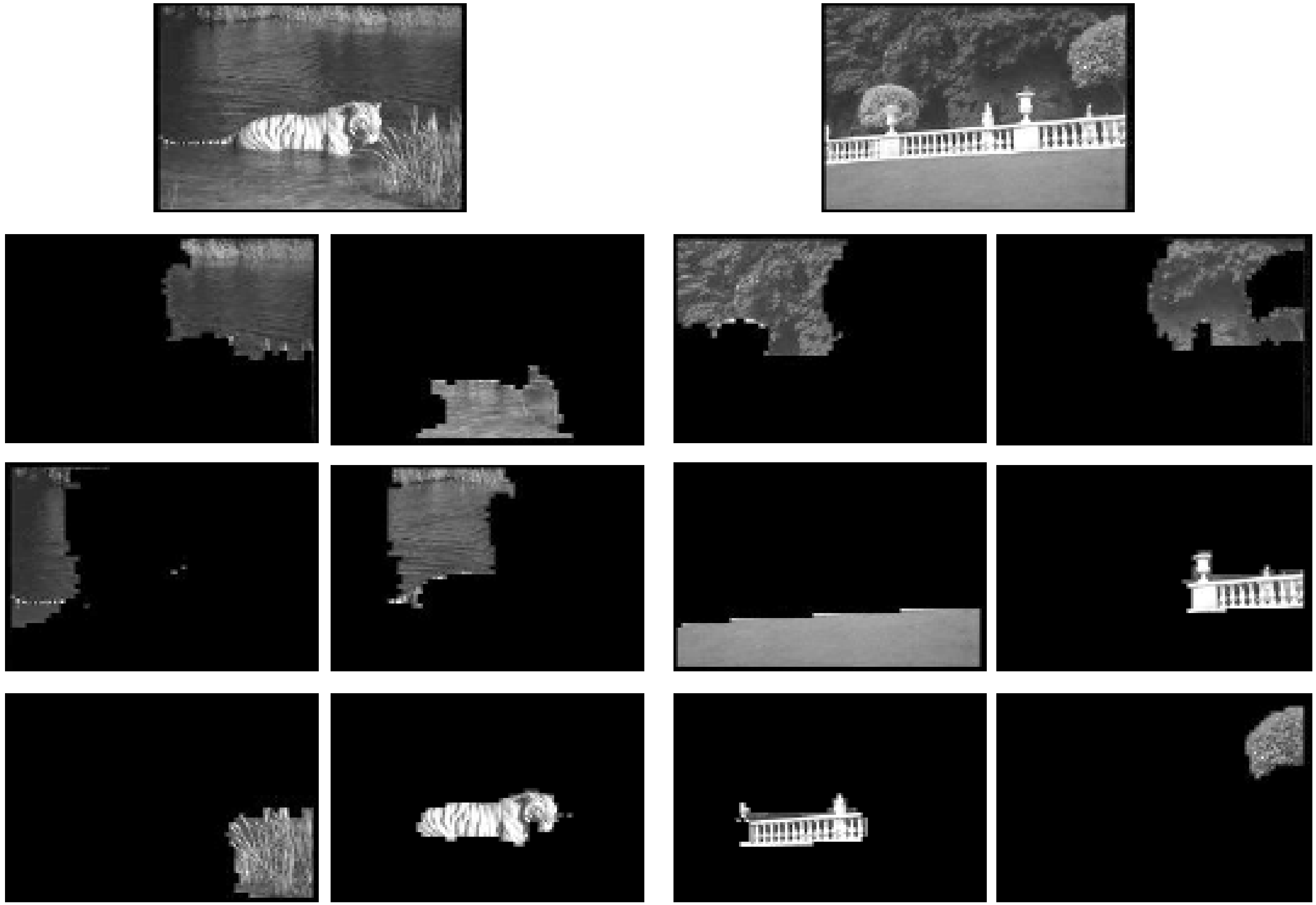
- Instead, solve the generalized eigenvalue problem

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

- which gives

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

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



(using intensity and texture affinity)

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

(using motion / spatio-temporal affinity)

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

Fitting

- Choose a parametric object/some objects to represent a set of tokens
- Most interesting case is when criterion is not local
  - can't tell whether a set of points lies on a line by looking only at each point and the next.

- Three main questions:
  - what object represents this set of tokens best?
  - which of several objects gets which token?
  - how many objects are there?

(you could read line for object here, or circle, or ellipse or...)

Fitting and the Hough Transform

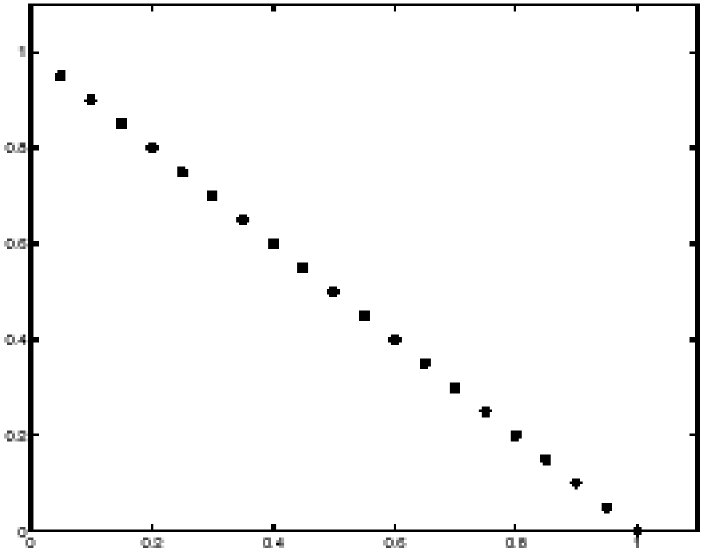
- Purports to answer all three questions
  - in practice, answer isn't usually all that much help
- We do for lines only
- A line is the set of points (x, y) such that

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

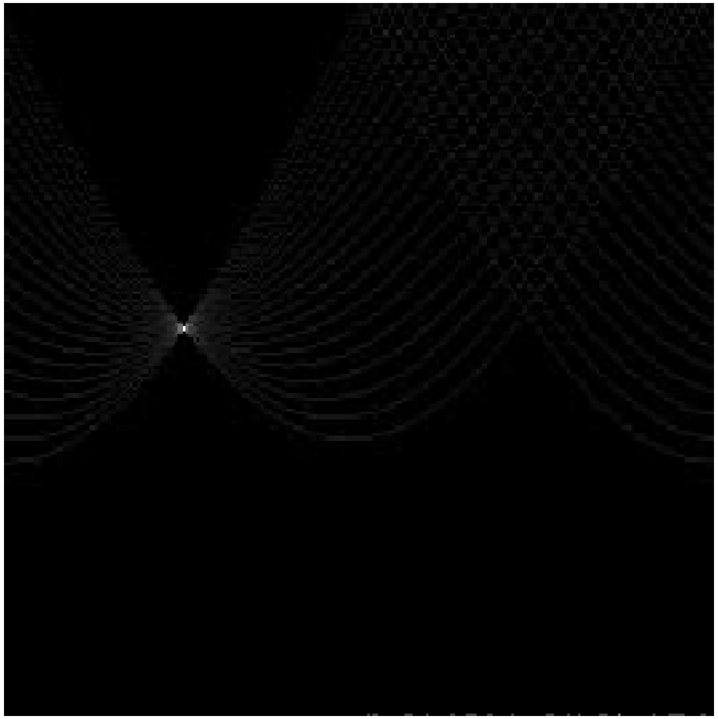
- Different choices of θ, d>0 give different lines
- For any (x, y) there is a one parameter family of lines through this point, given by

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

- Each point gets to vote for each line in the family; if there is a line that has lots of votes, that should be the line passing through the points



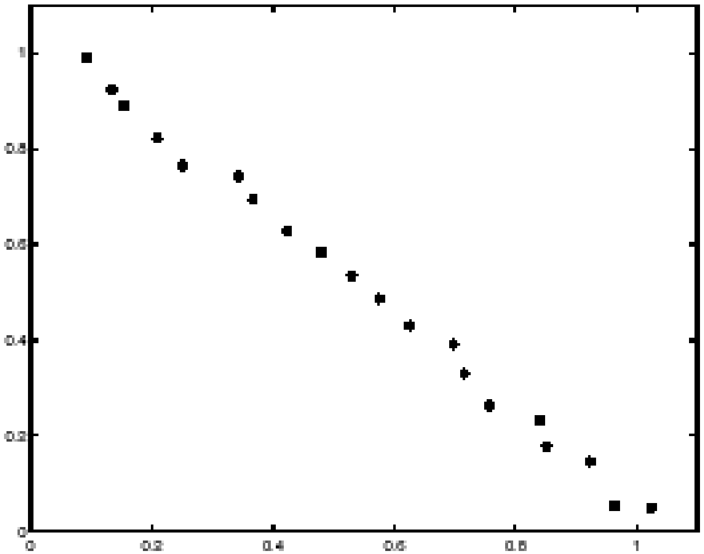
tokens



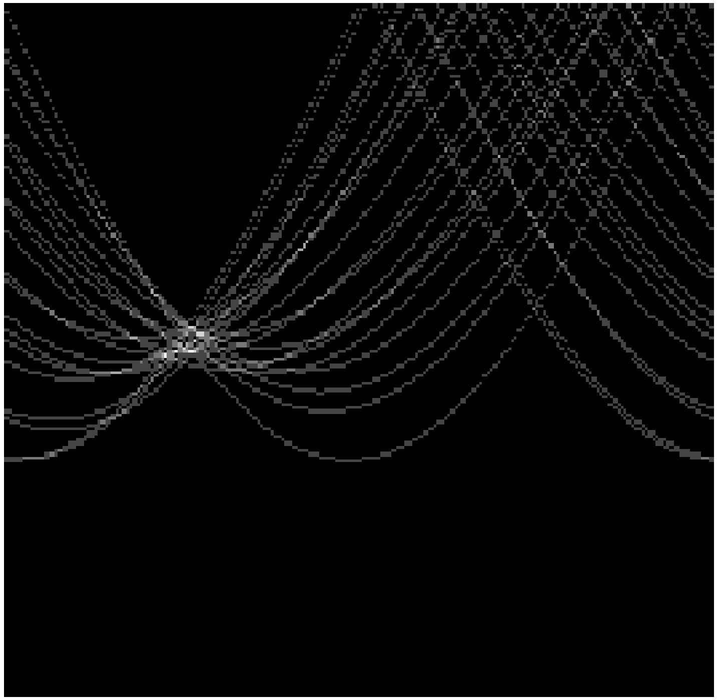
votes

Mechanics of the Hough transform

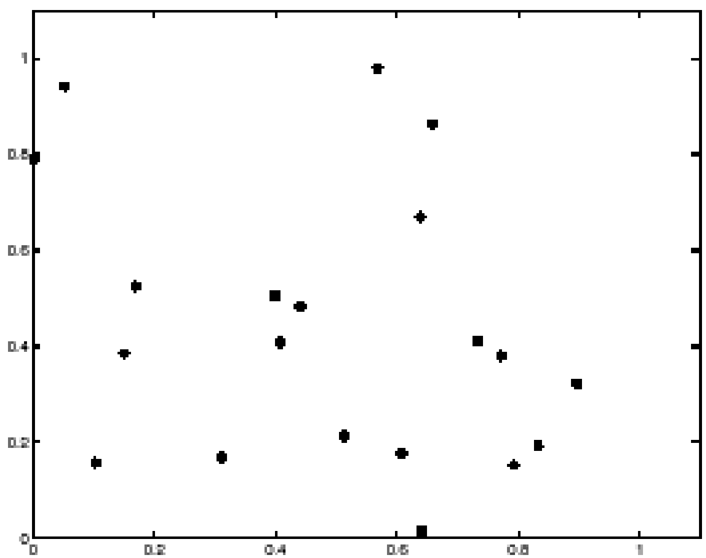
- Construct an array representing θ, d
- For each point, render the curve (θ, d) into this array, adding one at each cell
- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - count the peaks in the Hough array
- Who belongs to which line?
  - tag the votes
- Hardly ever satisfactory in practice, because problems with noise and cell size defeat it



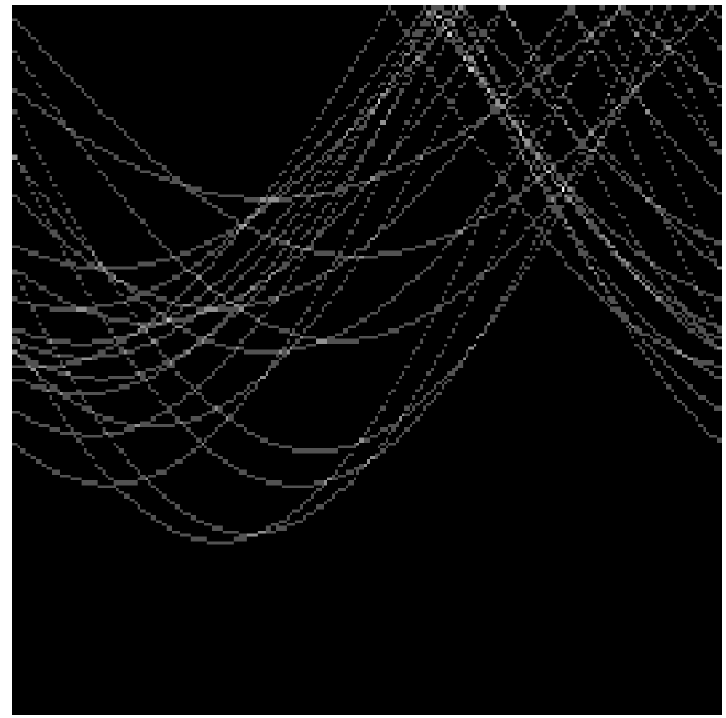
tokens

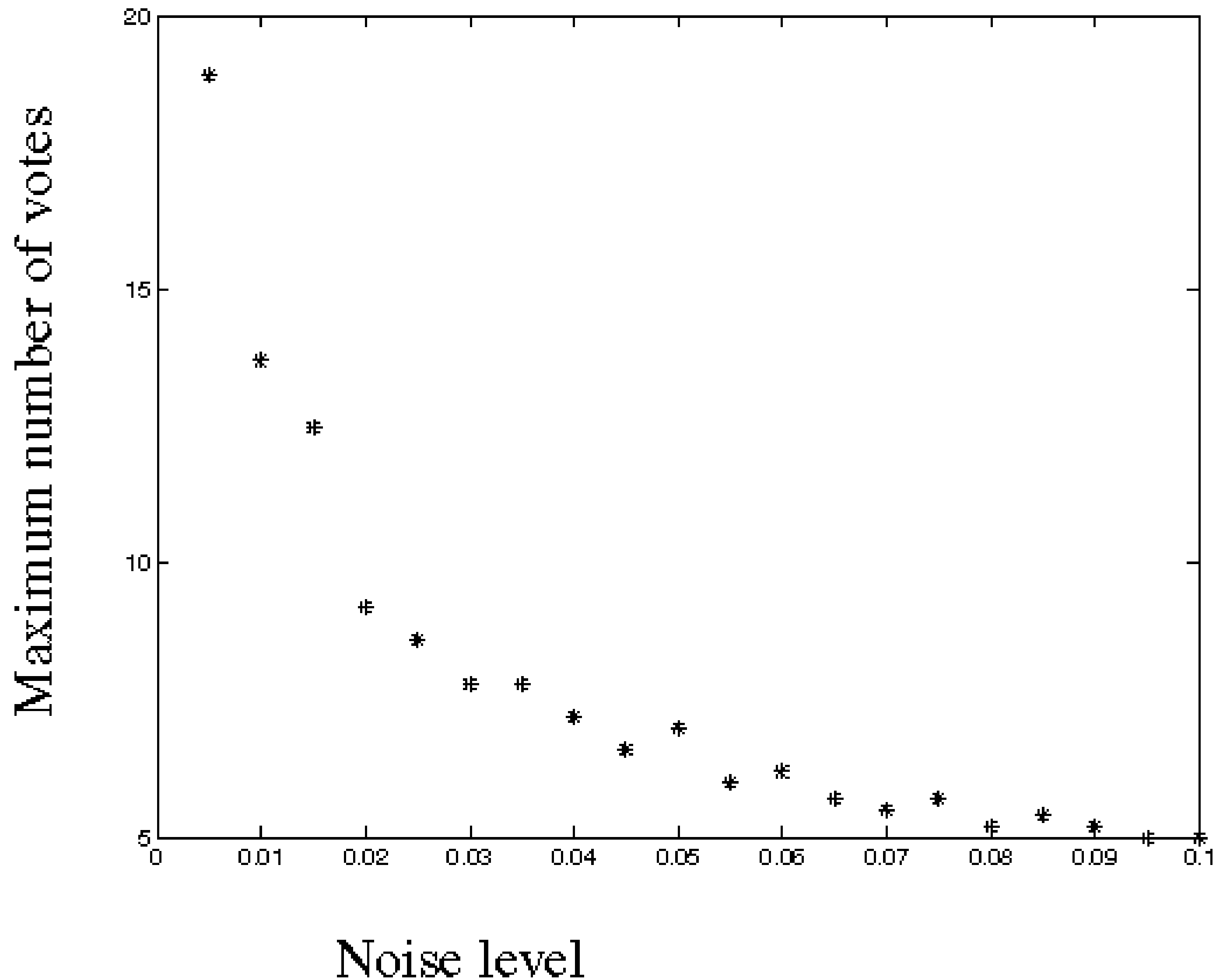


votes

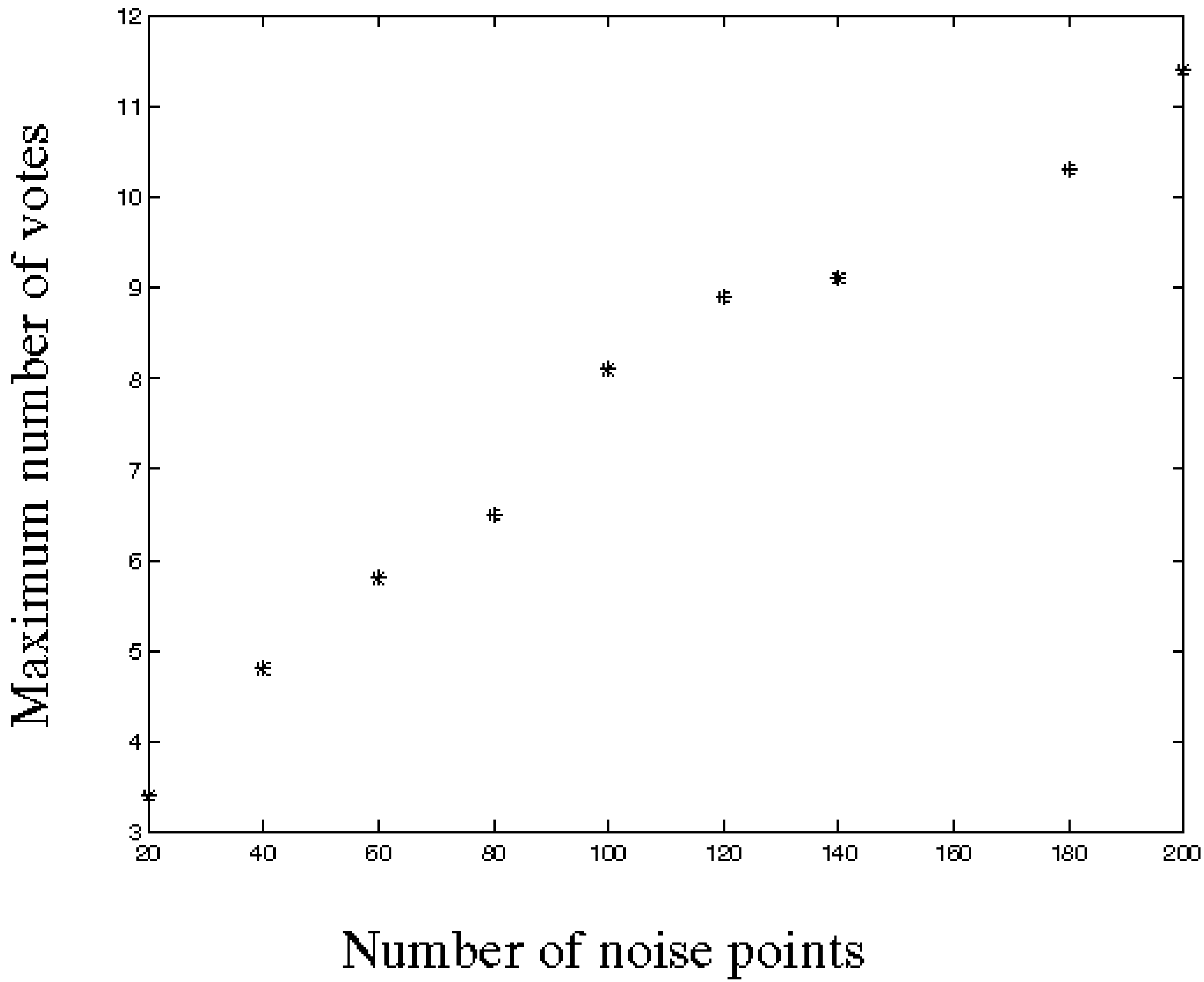


0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1





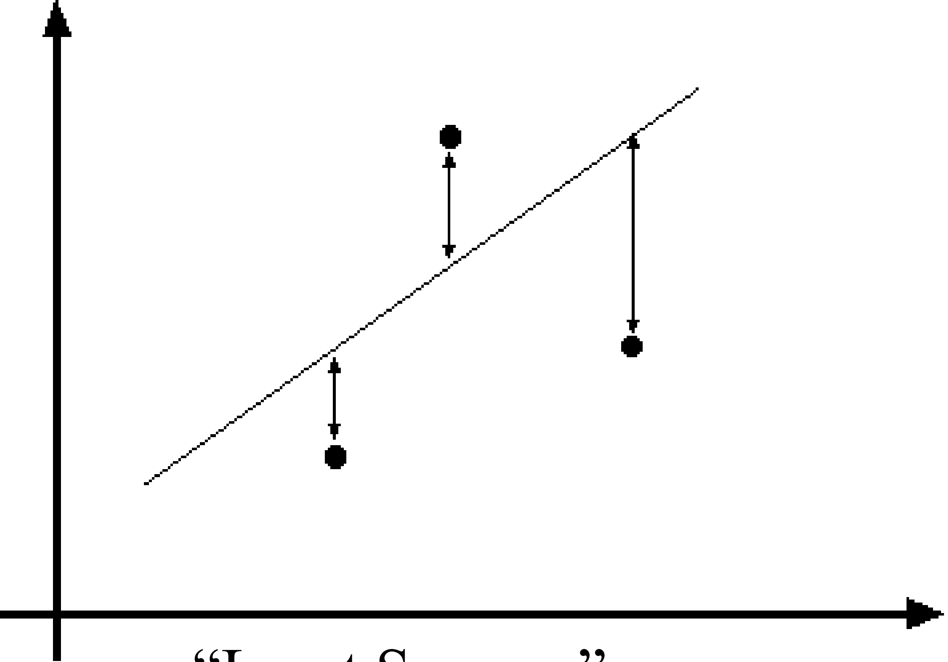
Maximum number of votes
20
15
10
5
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Noise level



Maximum number of votes
Number of noise points
12
11
10
9
8
7
6
5
4
3
20
40
60
80
100
120
140
160
180
200

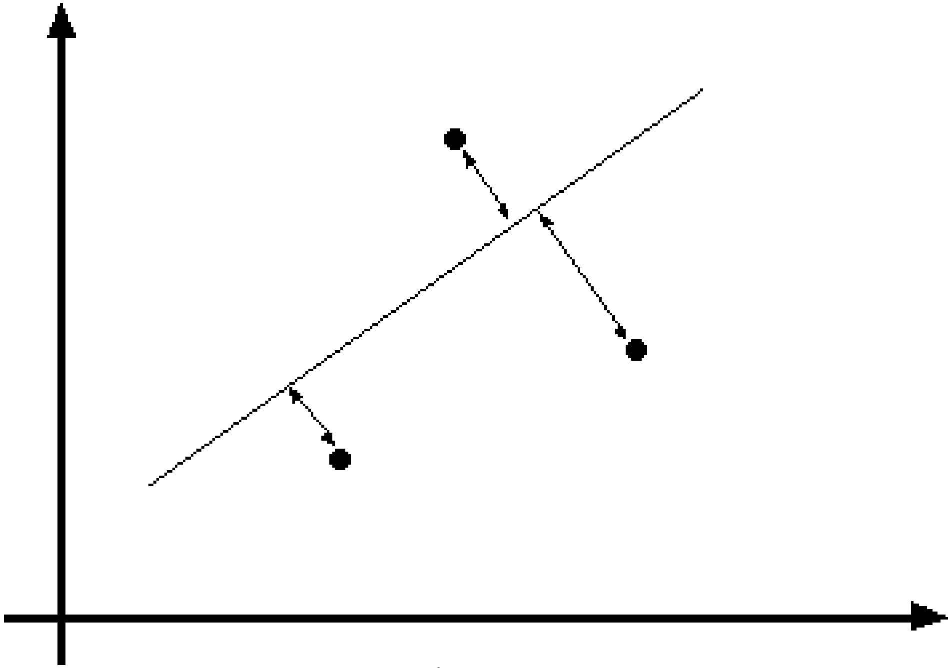
Line fitting

What criteria to optimize when fitting a line to a set of points?



"Least Squares"

Line fitting can be max. likelihood - but choice of model is important



"Total Least Squares"

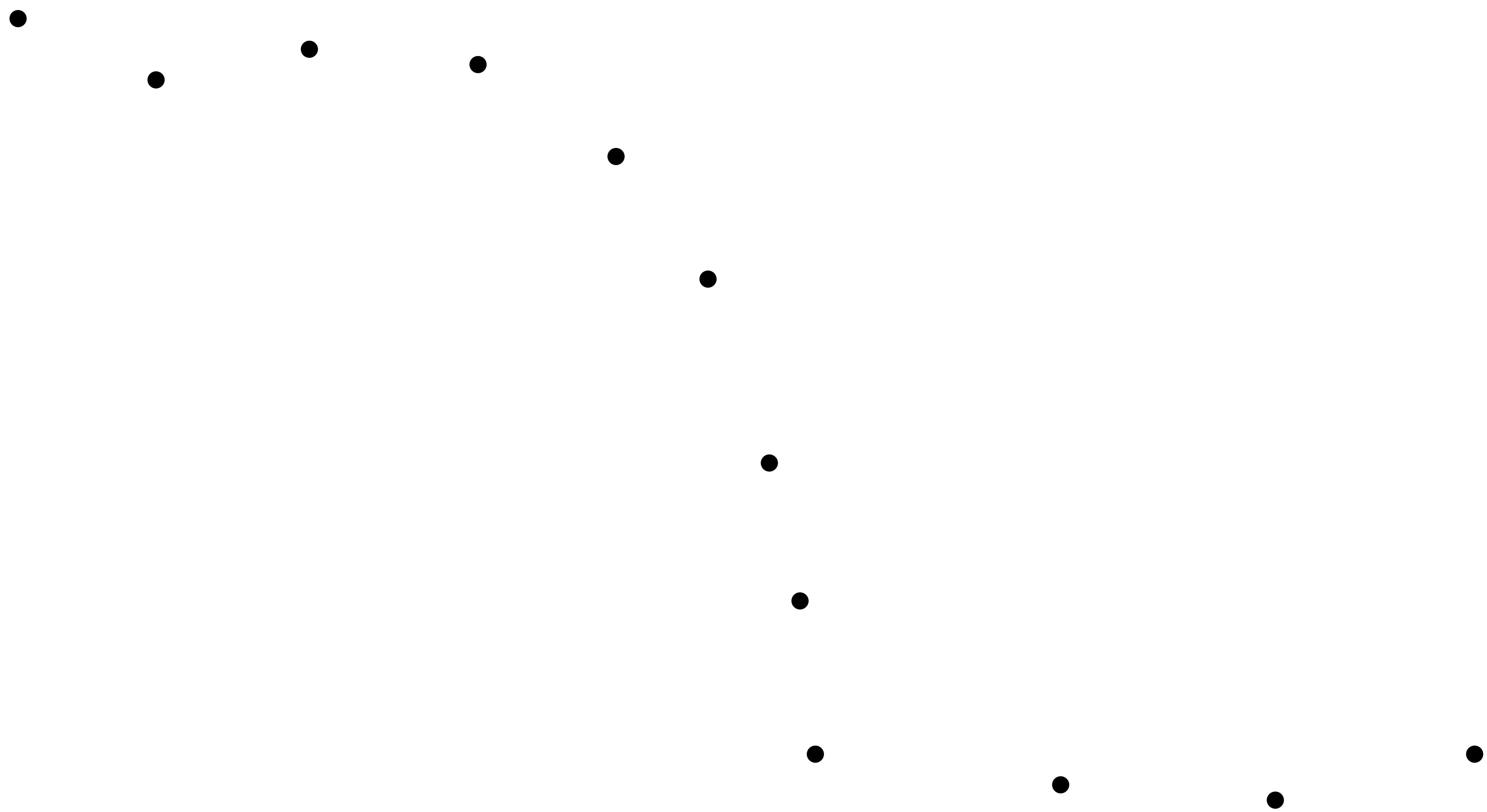
Who came from which line?

- Assume we know how many lines there are - but which lines are they?
  - easy, if we know who came from which line
- Three strategies
  - Incremental line fitting
  - K-means
  - Probabilistic (later!)

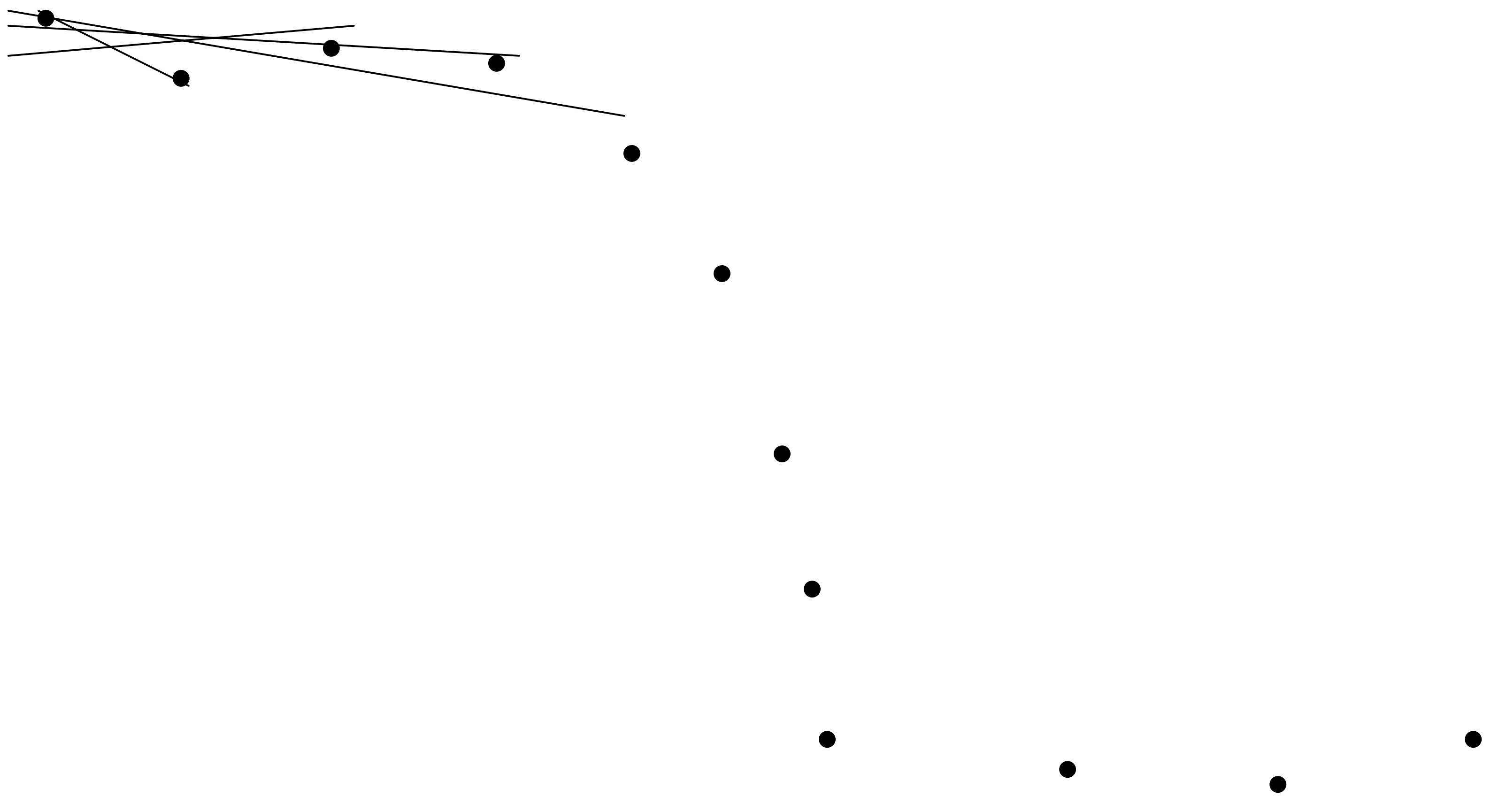
Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next point on the curve
         to the line point list and refit the line
   end
   Transfer last point(s) back to curve
   Refit line
   Attach line to line list
end
```

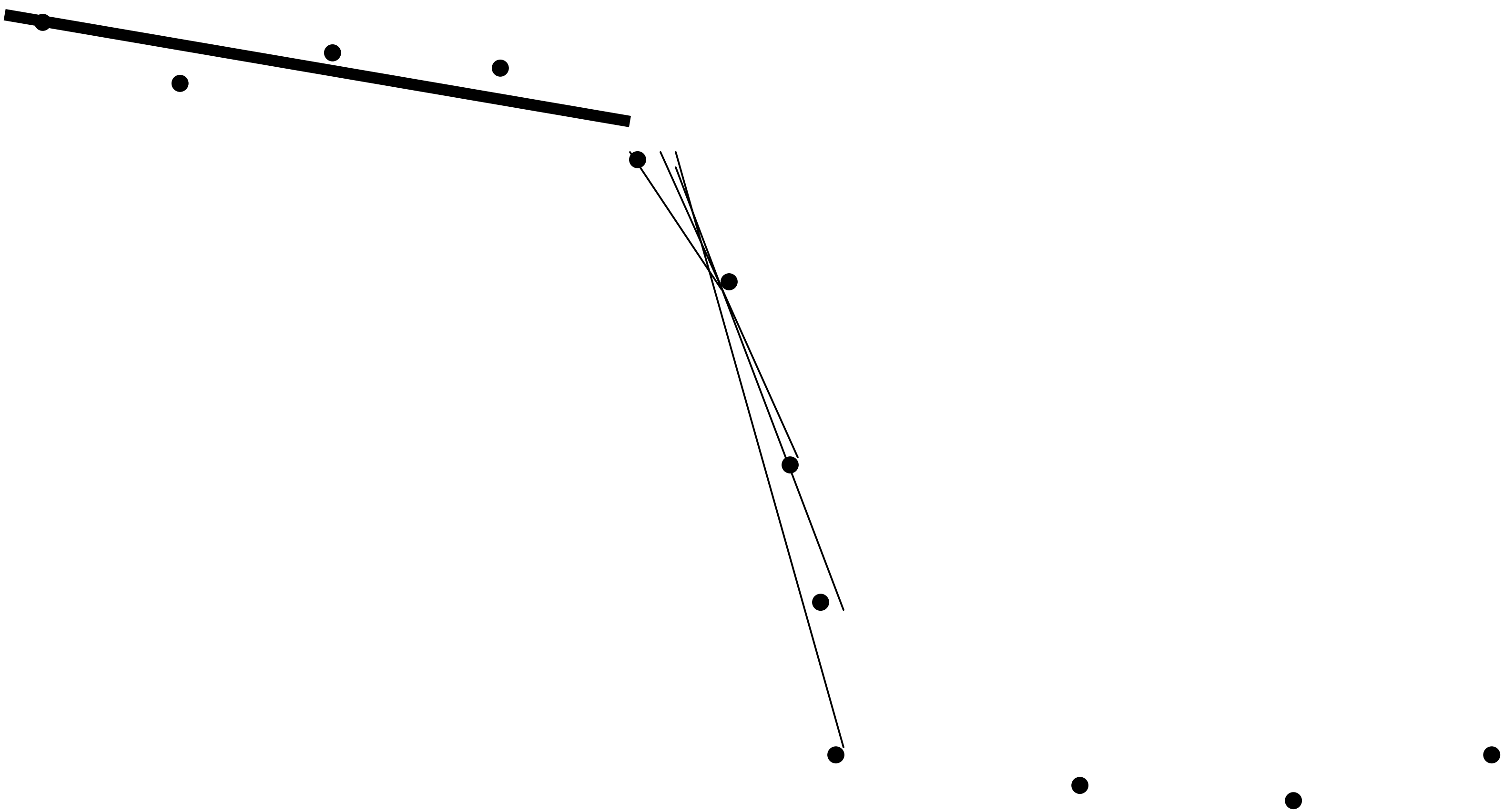
Incremental line fitting



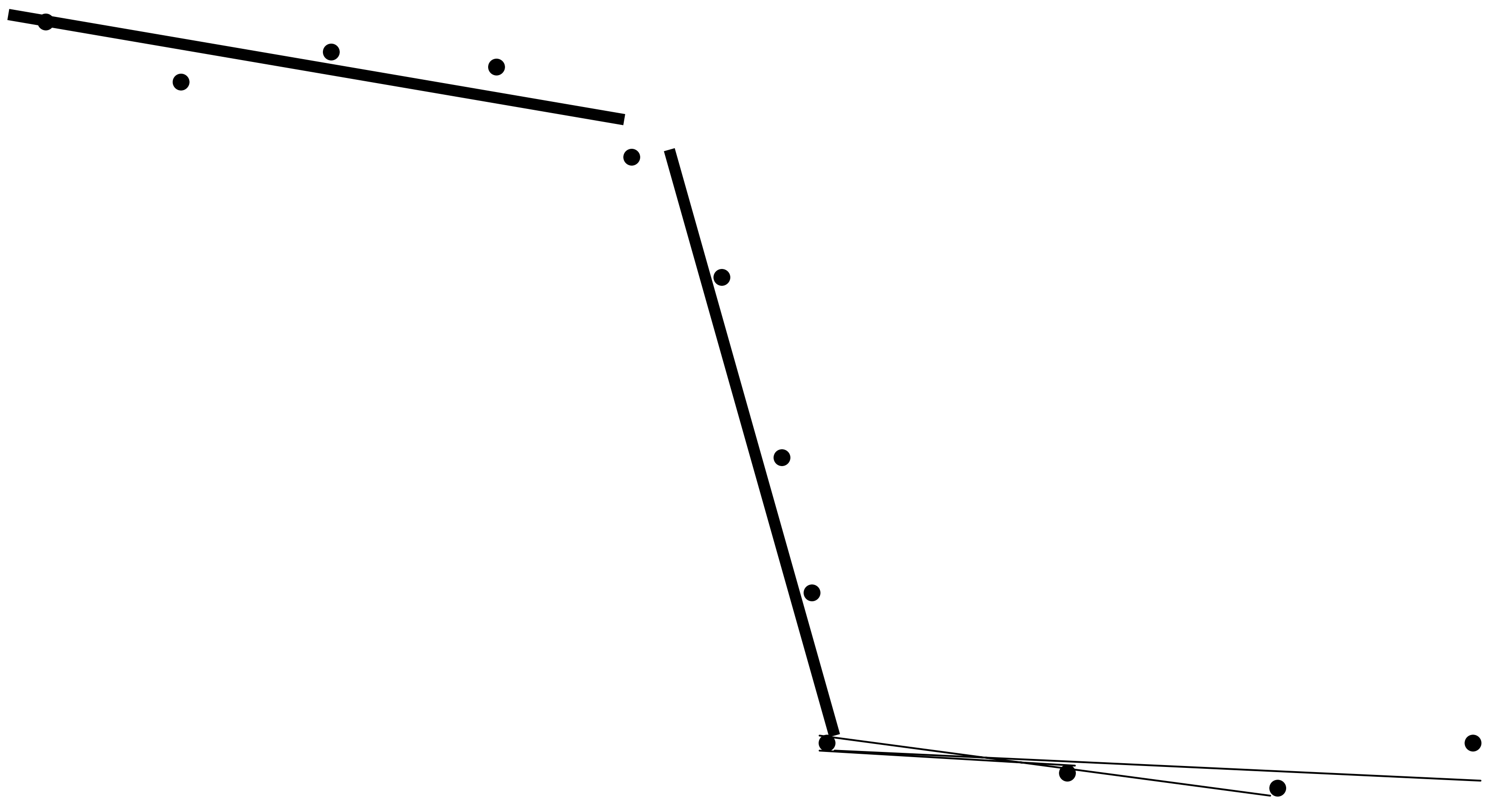
Incremental line fitting



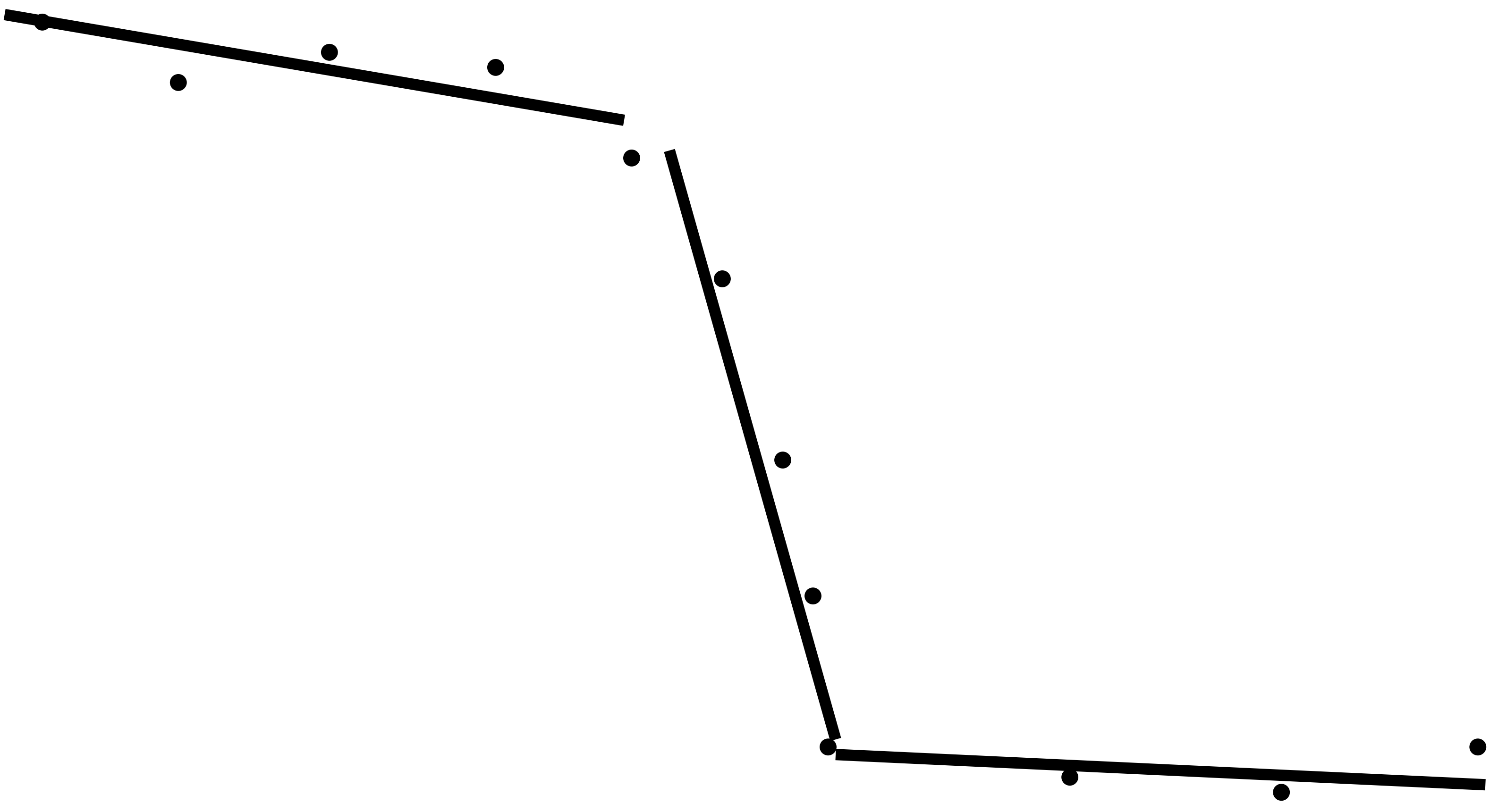
Incremental line fitting



Incremental line fitting



Incremental line fitting

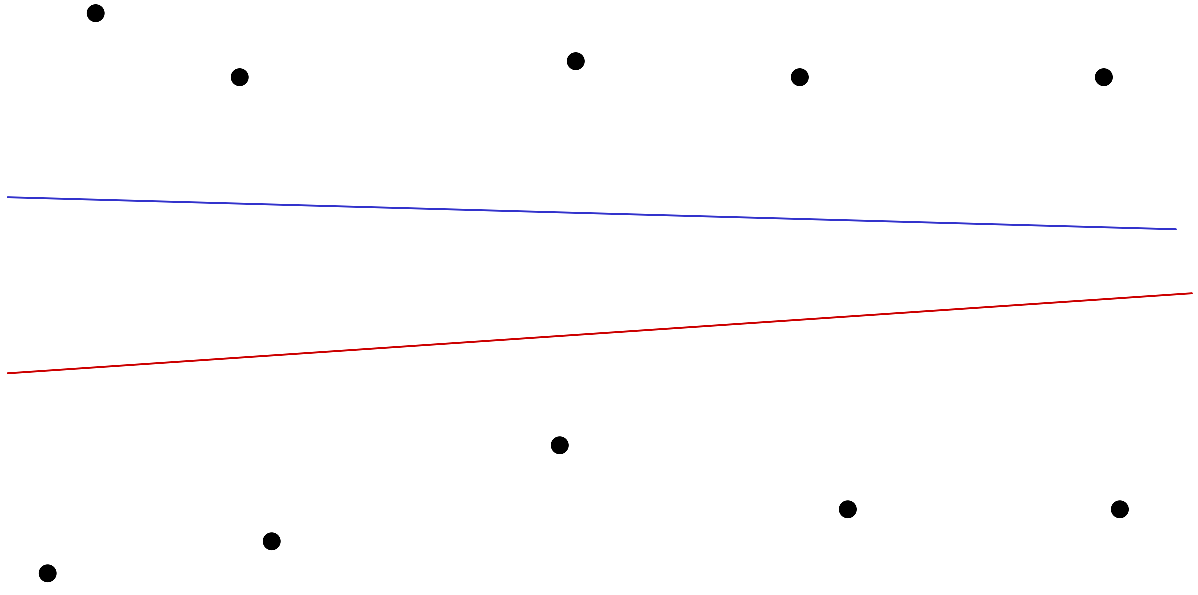


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

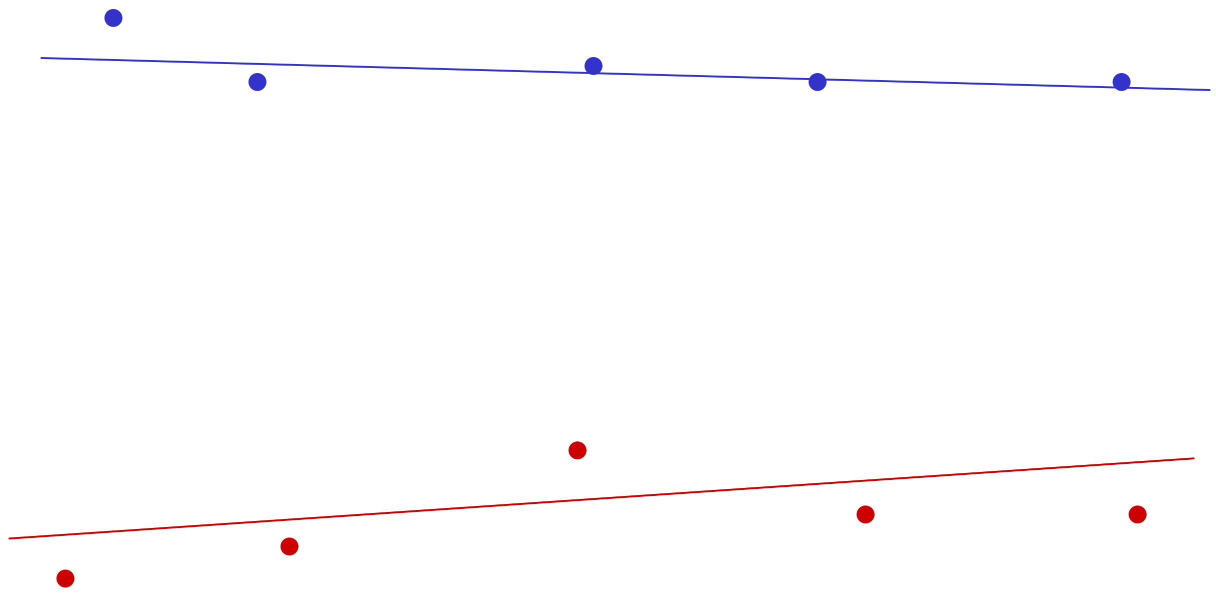
Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
  and then fit lines using this assignment

Until convergence
  Allocate each point to the closest line
  Refit lines
end

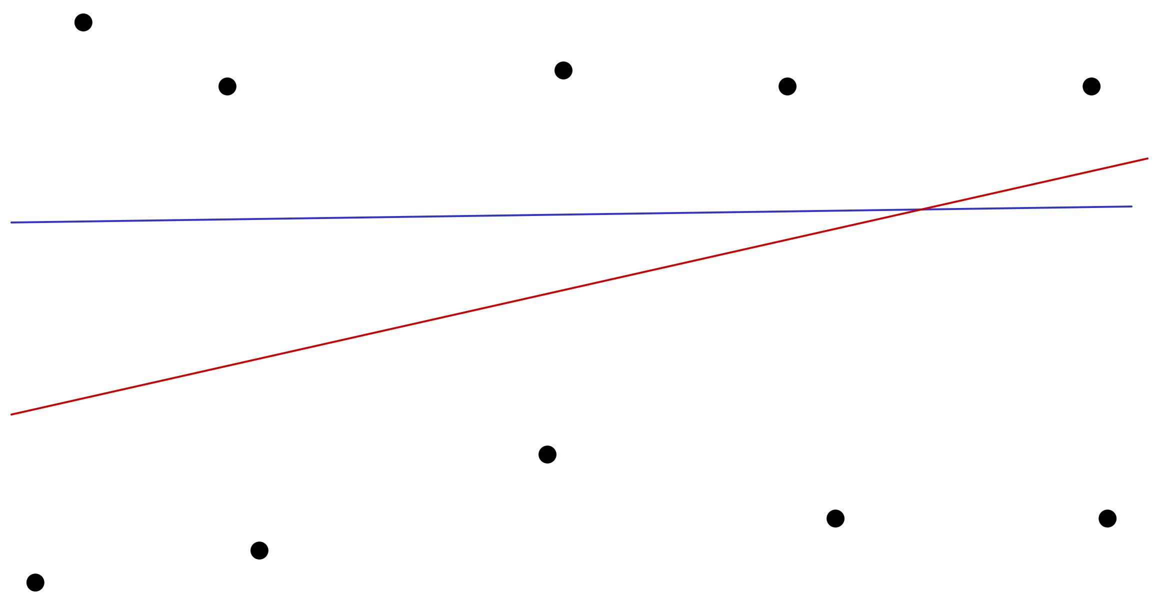
K-means line fitting



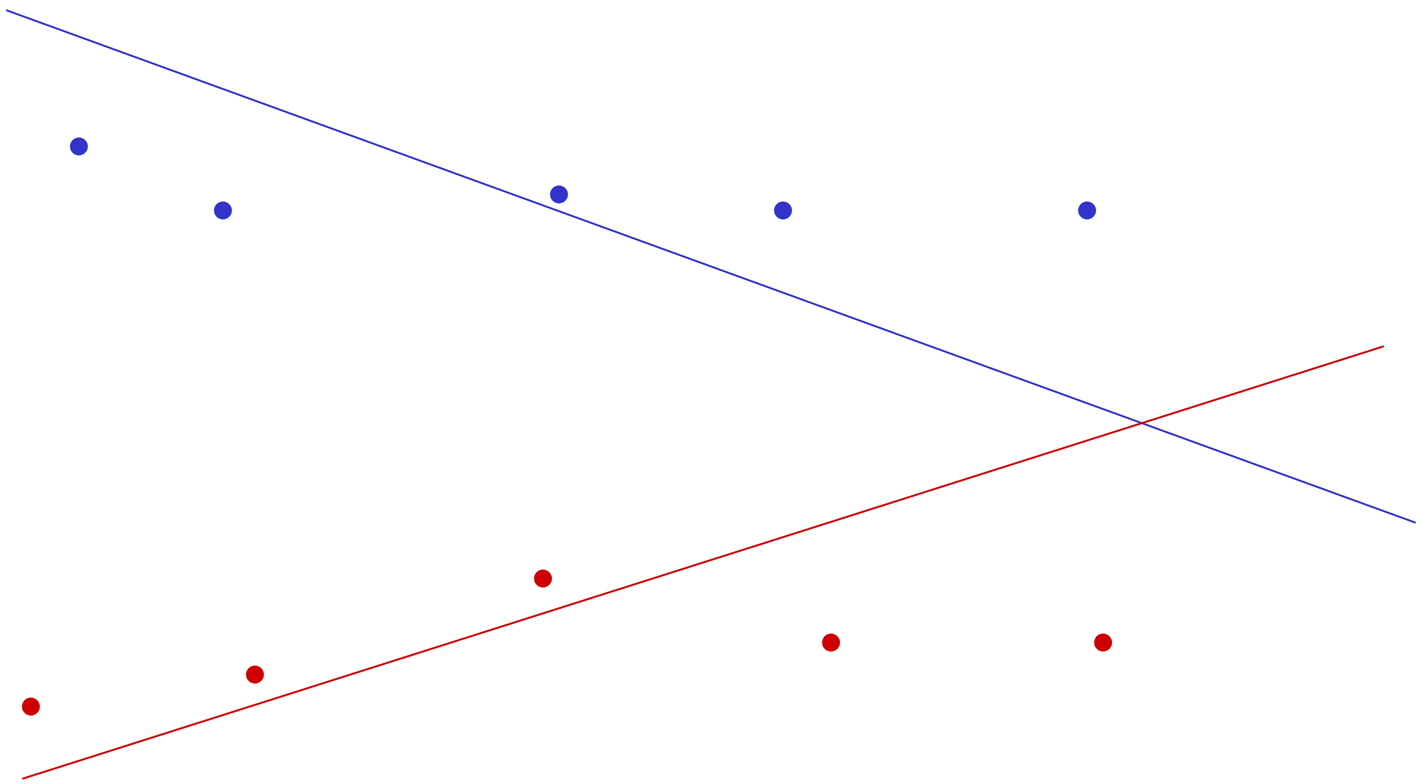
K-means line fitting



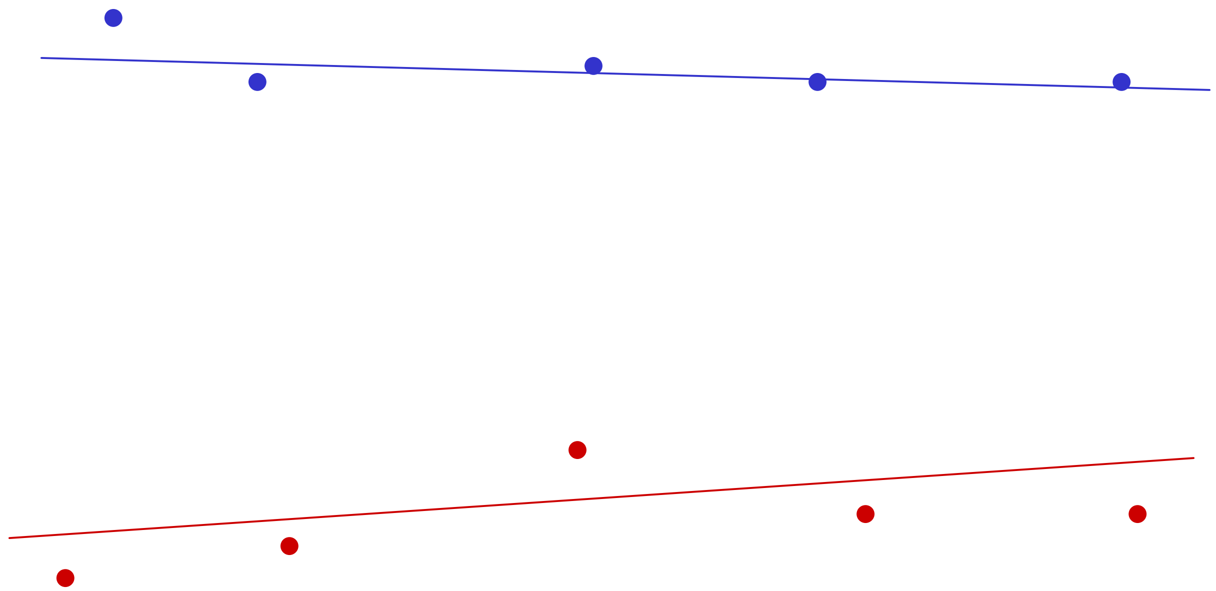
K-means line fitting



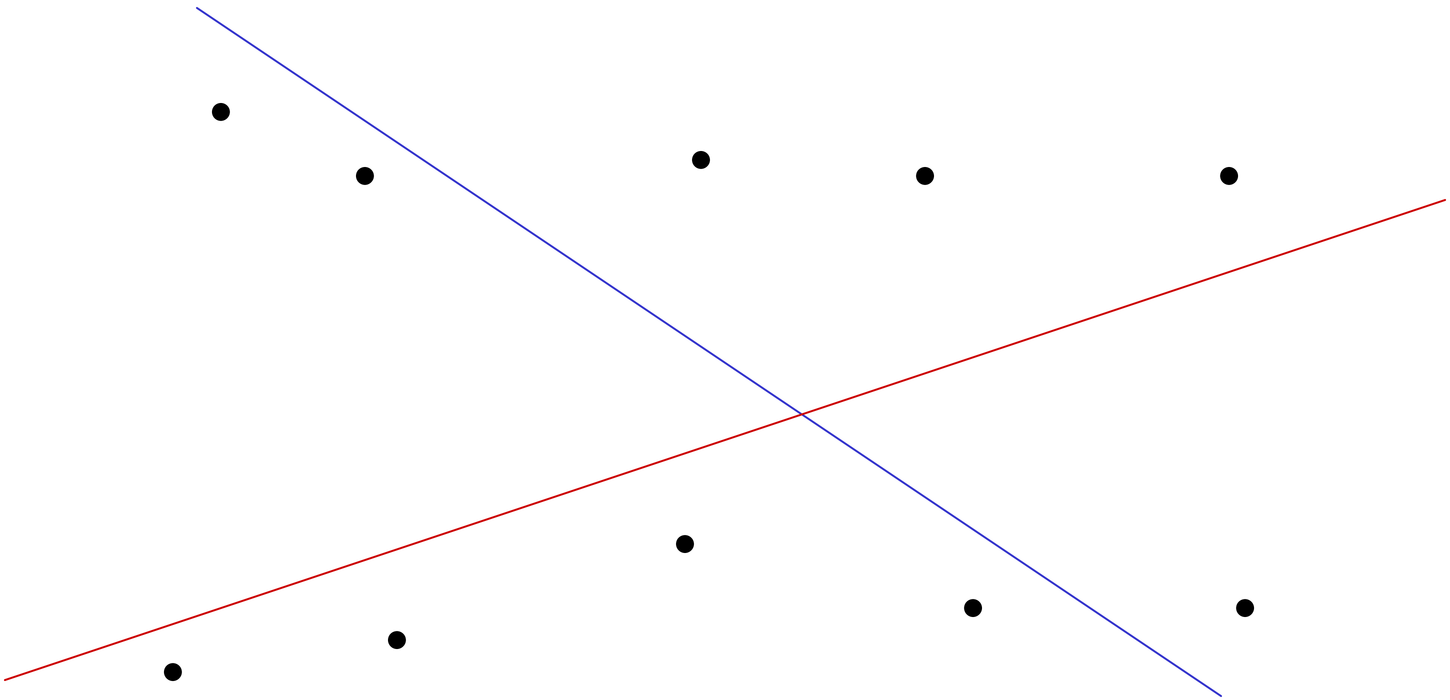
K-means line fitting



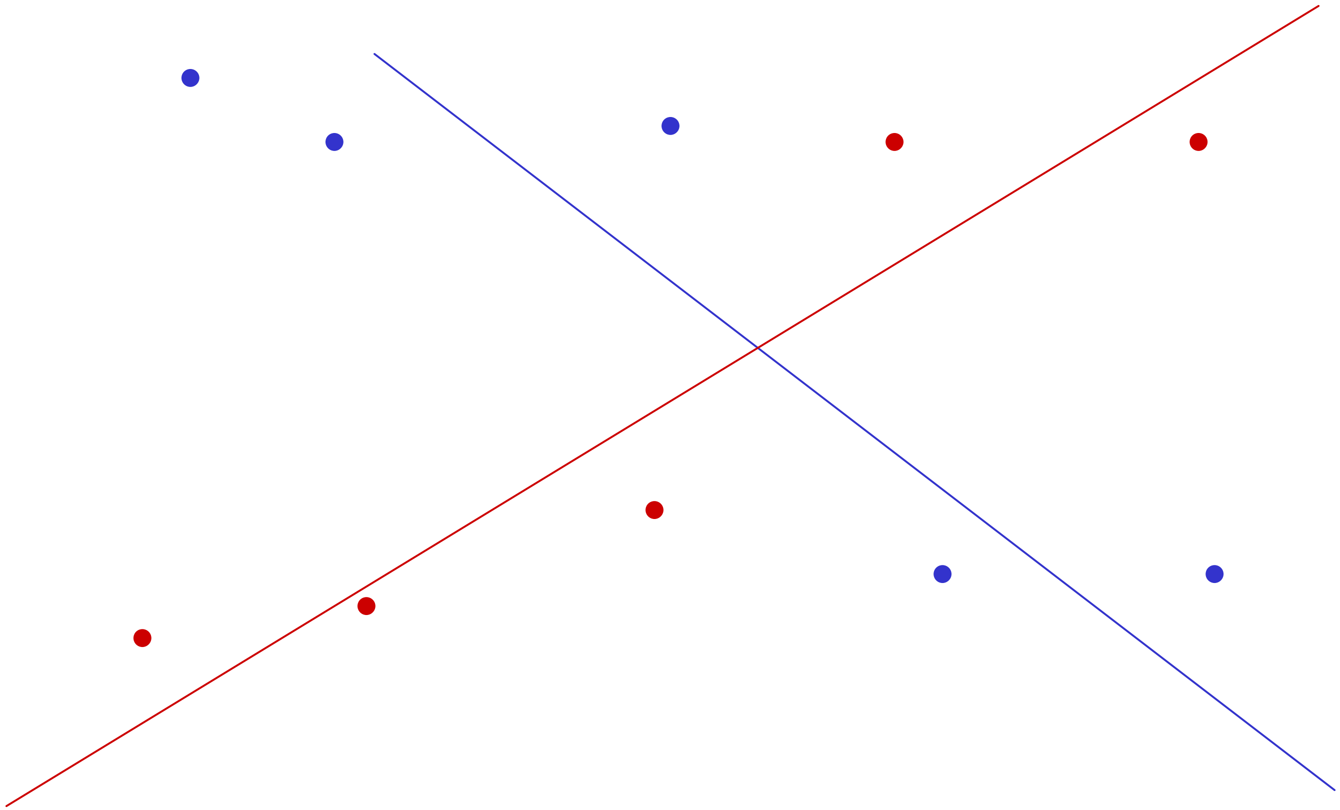
K-means line fitting



K-means line fitting



K-means line fitting



Robustness

- As we have seen, squared error can be a source of bias in the presence of noise points
  - One fix is EM - we'll do this shortly
  - Another is an M-estimator
    - Square nearby, threshold far away
  - A third is RANSAC
    - Search for good points

(Next lecture….)

Segmentation and Line Fitting

- Gestalt grouping
- Background subtraction
- K-Means
- Graph cuts
- Hough transform
- Iterative fitting

(Next time: Probabilistic segmentation)

[Most figures from F&P or
http://iram.cs.berkeley.edu/RetreatJul2000/malik.ppt]