6.891

Computer Vision and Applications

Prof. Trevor. Darrell

Lecture 14:

- Unsupervised Category Learning
- Gestalt Principles
- Segmentation by Clustering
  - K-Means
  - Graph cuts
- Segmentation by Fitting
  - Hough transform
  - Fitting

Readings: F&P Ch. 14, 15.1-15.2

(Un)Supervised Learning

- Methods in last two lectures presume:
  - Segmentation
  - Labeling
  - Alignment
- What can we do with unsupervised (weakly supervised) data?
- Clustering / Generative Model Approach…

Representation

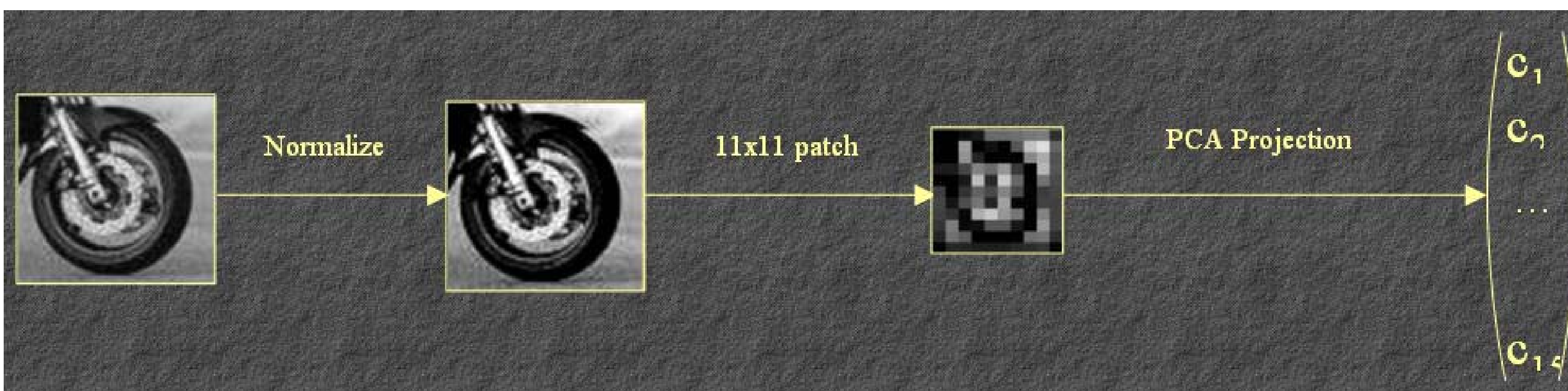
Use a scale invariant, scale sensing feature keypoint detector (like the first steps of Lowe's SIFT).

From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

[Slide from Bradsky & Thrun, Stanford]

Features for Category Learning

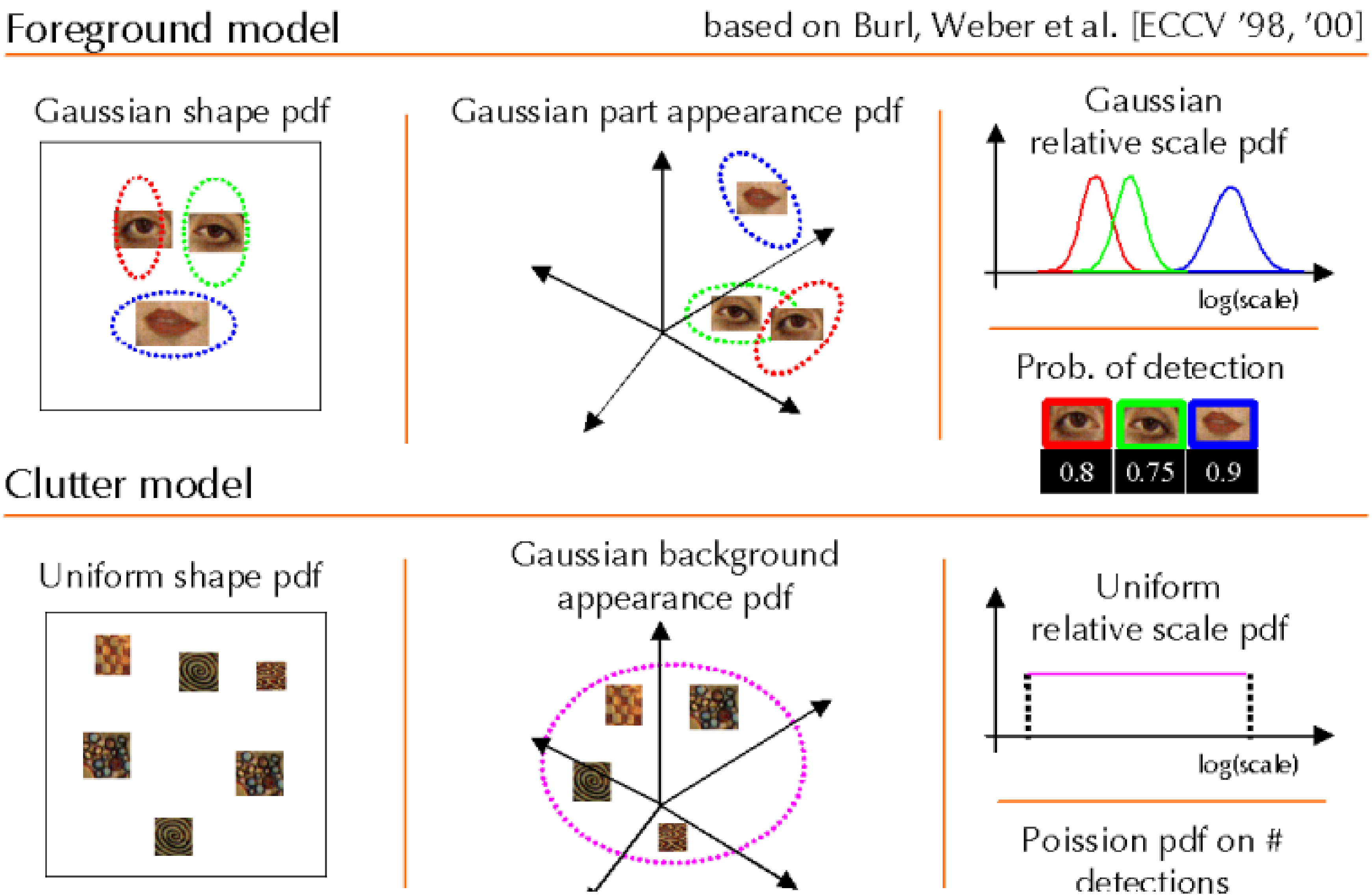
A direct appearance model is taken around each located key. This is then normalized by it's detected scale to an 11x11 window. PCA further reduces these features.



[Slide from Bradsky & Thrun, Stanford]

From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

Generative probabilistic model



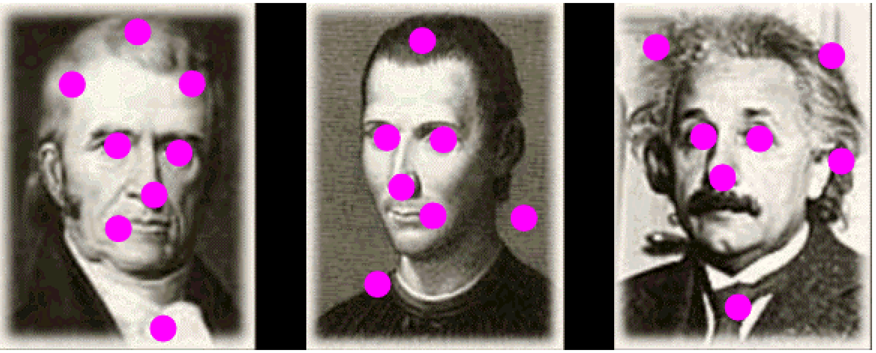
$$p(\mathcal{X},\mathcal{A}|\boldsymbol{\theta}) = \sum_{\mathbf{h}\in H} p(\mathcal{X},\mathcal{A},\mathbf{h}|\boldsymbol{\theta}) = \sum_{\mathbf{h}\in H} \underbrace{p(\mathcal{A}|\mathcal{X},\mathbf{h},\boldsymbol{\theta})}_{Appearance}\,\underbrace{p(\mathcal{X}|\mathbf{h},\boldsymbol{\theta})}_{Shape}$$

From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

[Slide from Bradsky & Thrun, Stanford]

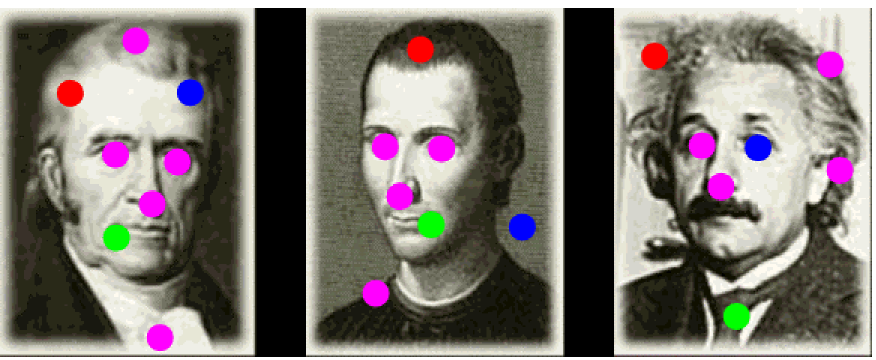
Learning

- Fit with E-M (this example is a 3 part model)
- We start with the dual problem of **what** to fit and **where** to fit it.

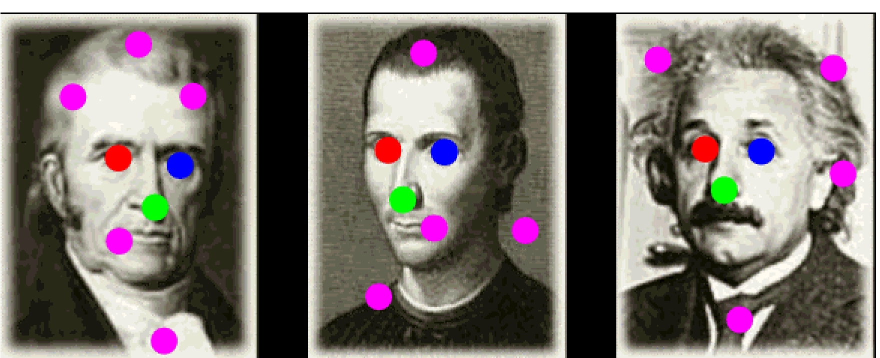


Assume that an object instance is the only consistent thing somewhere in a scene.

We don't know where to start, so we use the initial random parameters.



1. (M) We find the best (consistent across images) assignment given the params.
2. (E) We refit the feature detector params. and repeat until converged.
    - Note that there isn't much consistency

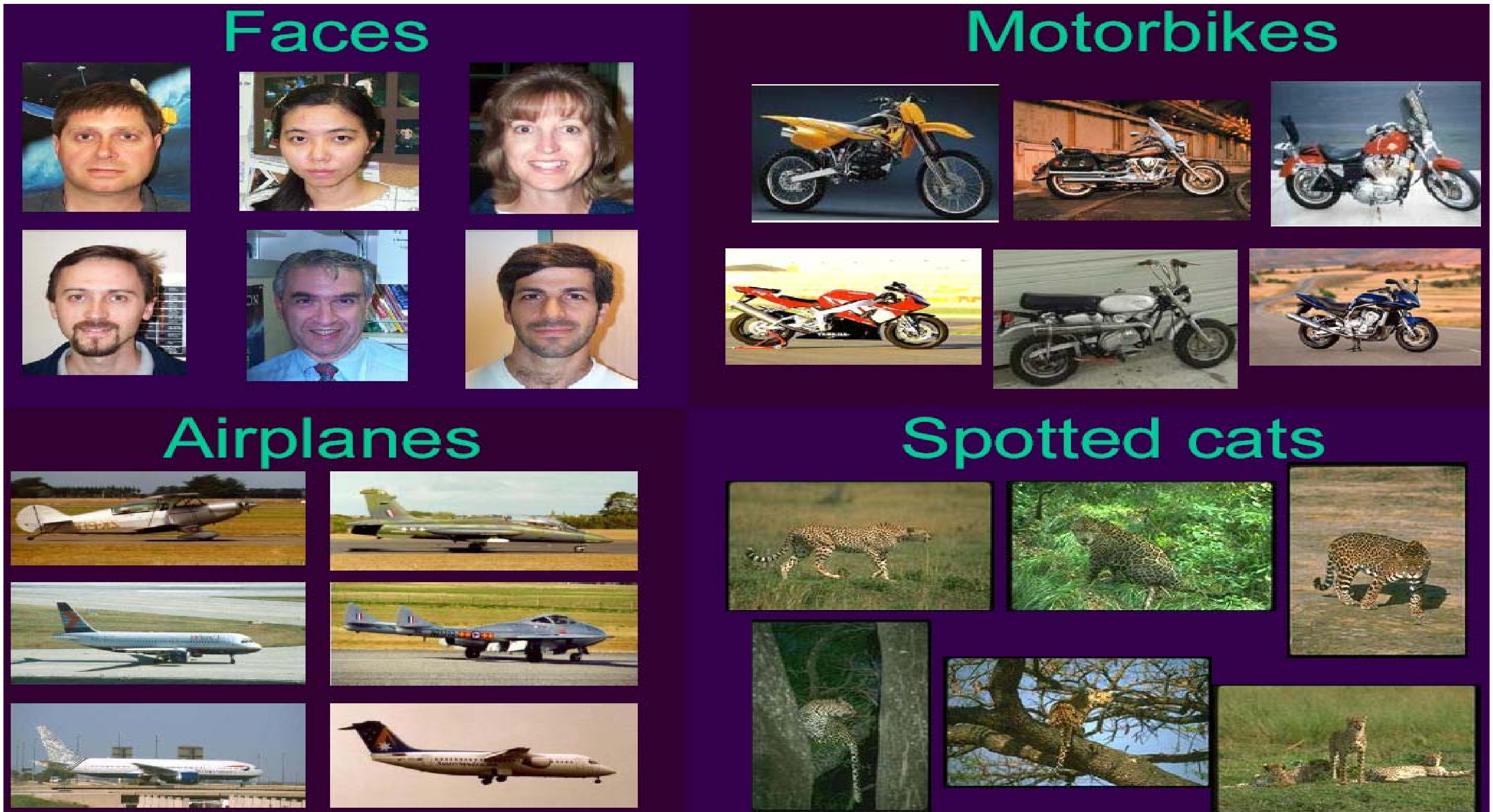


3. This repeats until it converges at the most consistent assignment with maximized parameters across images.

From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

[Slide from Bradsky & Thrun, Stanford]

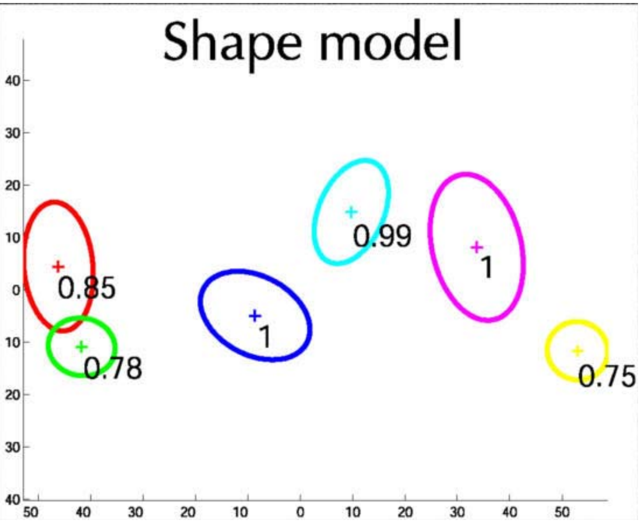
Data

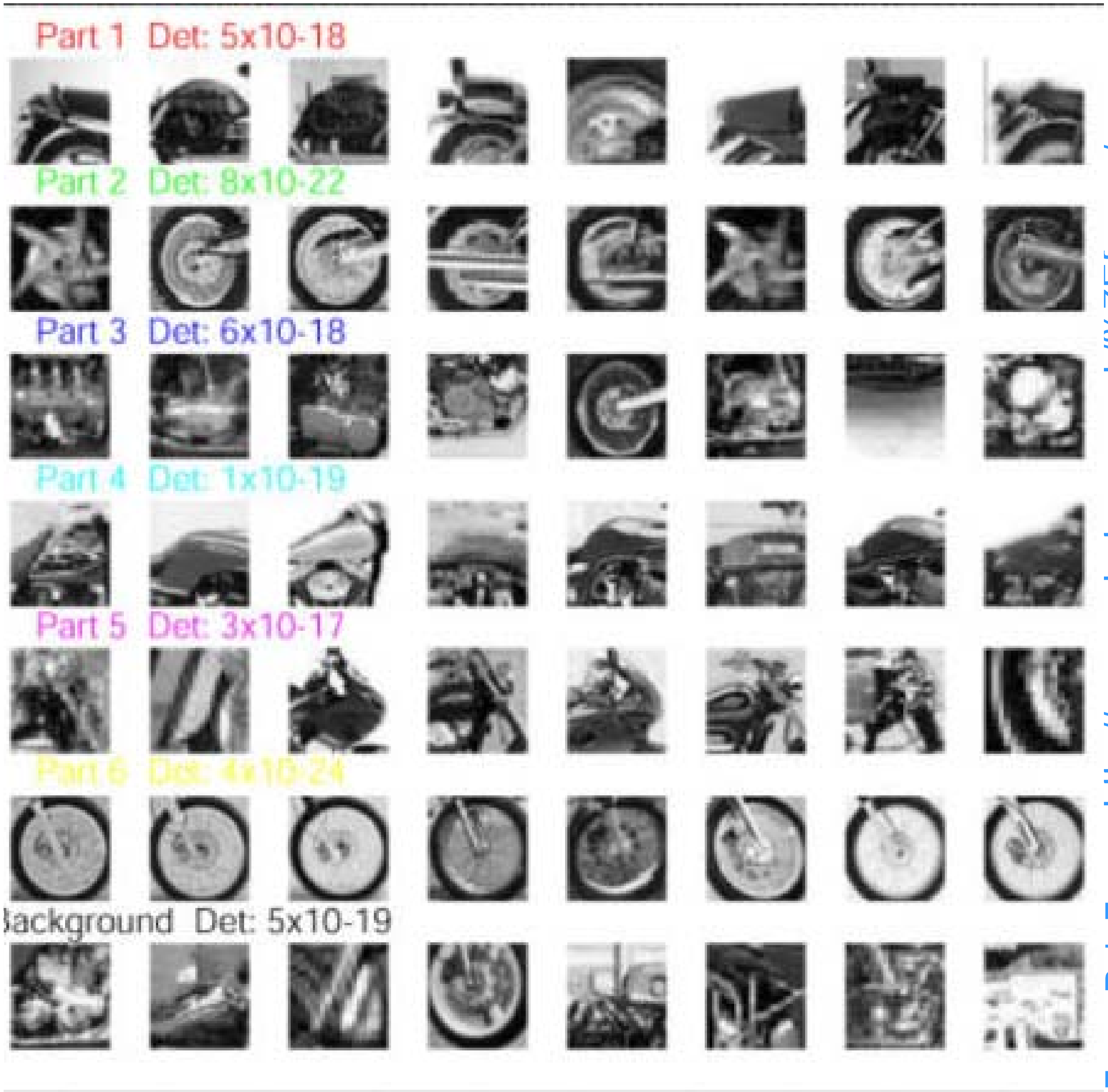


[Slide from Bradsky & Thrun, Stanford]

Slide from Li Fei-Fei http://www.vision.caltech.edu/feifeili/Resume.htm

Learned Model

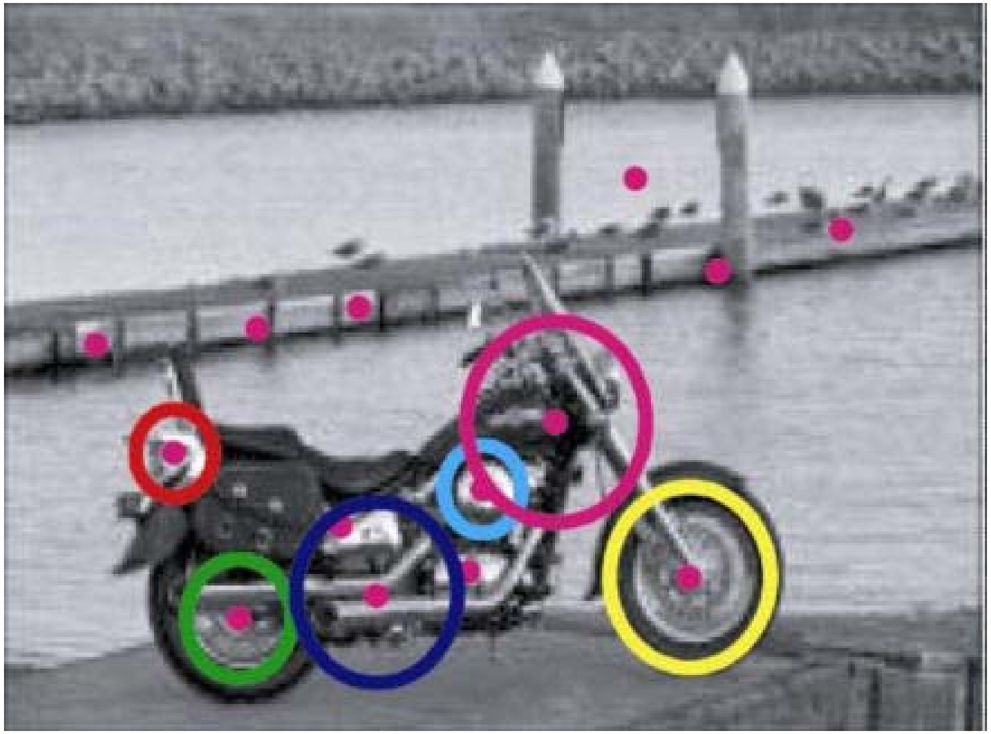




From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

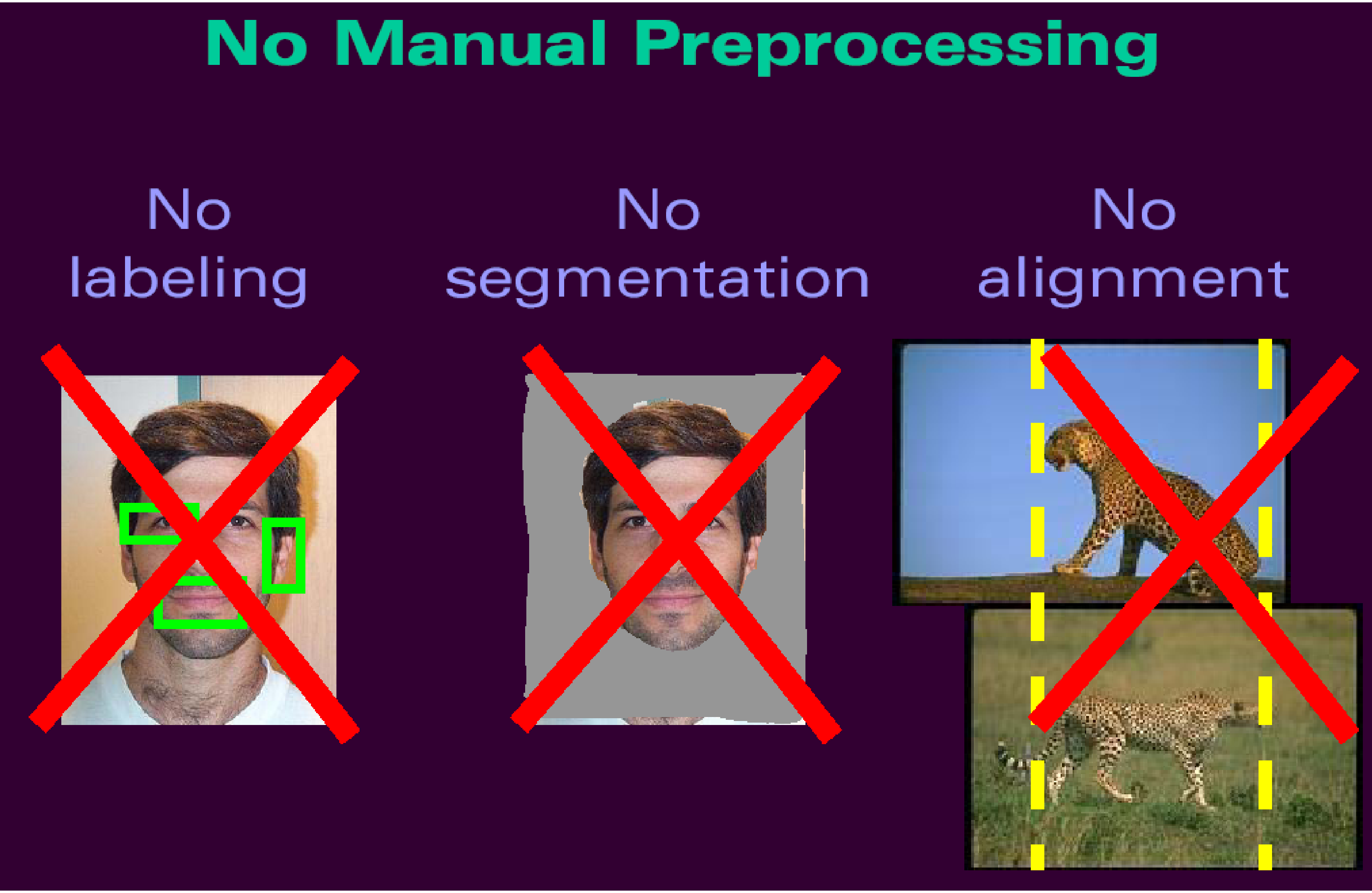
The shape model. The mean location is indicated by the cross, with the ellipse showing the uncertainty in location. The number by each part is the probability of that part being present.

Recognition



From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

Result: Unsupervised Learning



[Slide from Bradsky & Thrun, Stanford]

Slide from Li Fei-Fei http://www.vision.caltech.edu/feifeili/Resume.htm

Algorithm	Training Examples	Categories	Results (error)
Burl, et al. Weber, et al. Fergus, et al.	200 ~ 400	Faces, Motorbikes, Spotted cats, Airplanes, Cars	5.6 - 10 %

From: Rob Fergus http://www.robots.ox.ac.uk/%7Efergus/

Segmentation and Line Fitting

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

(Next time: Probabilistic segmentation)

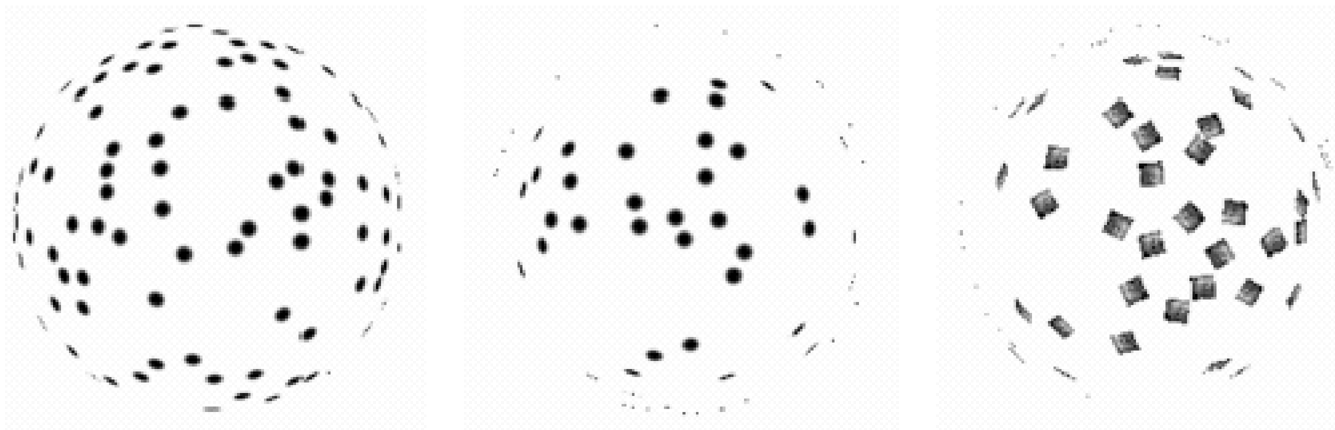
Segmentation and Grouping

- 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

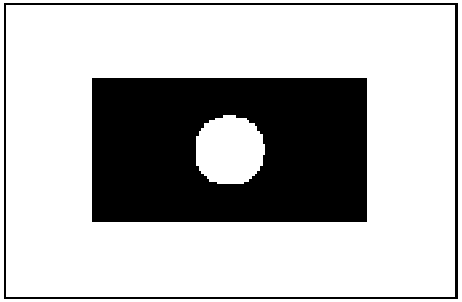
- Grouping (or clustering)
  - collect together tokens that "belong together"
- Fitting
  - associate a model with tokens
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?

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



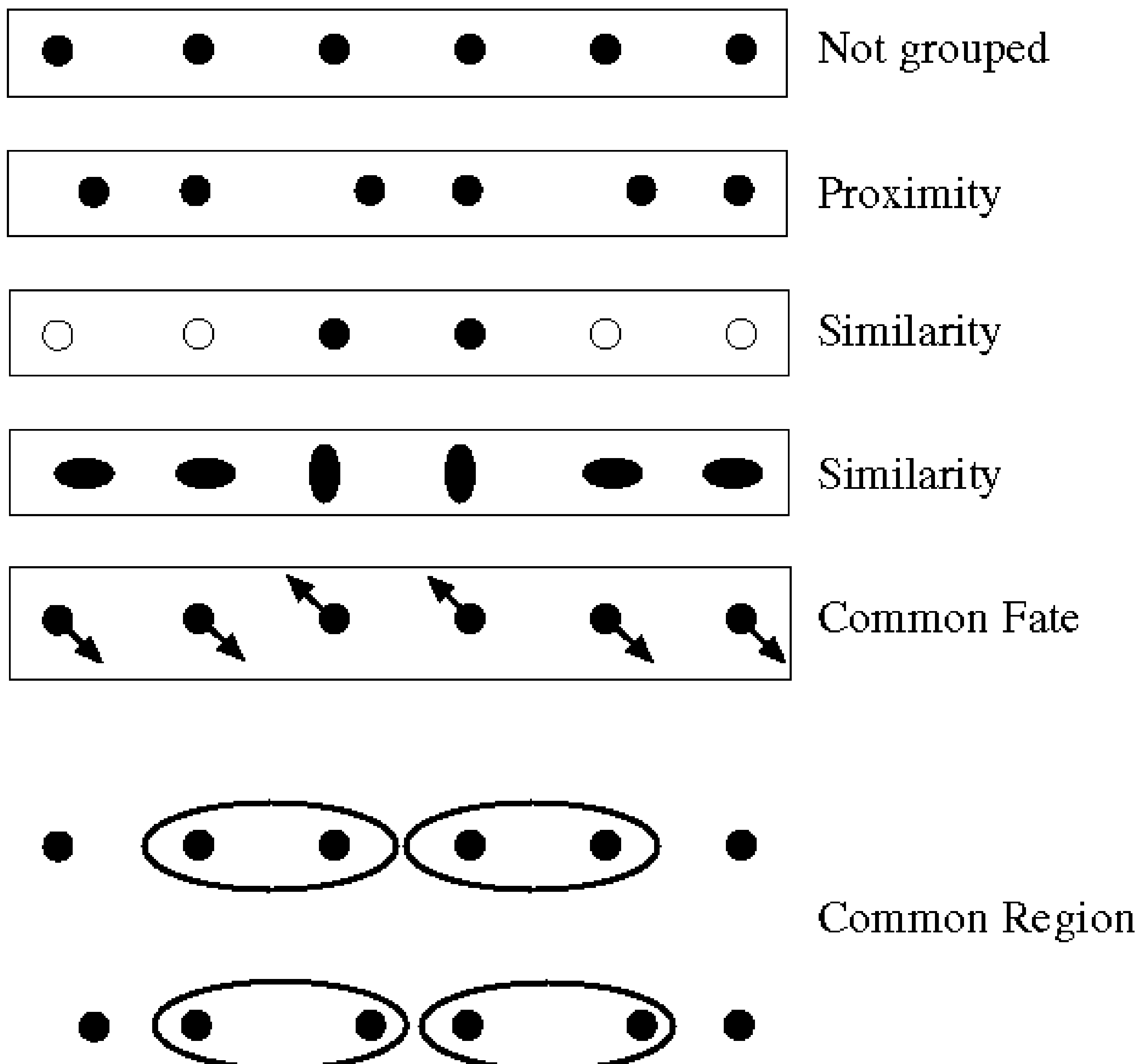
Why do these tokens belong together?

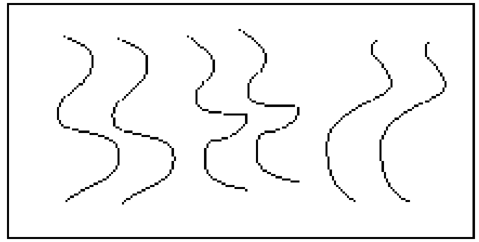


What is the figure?

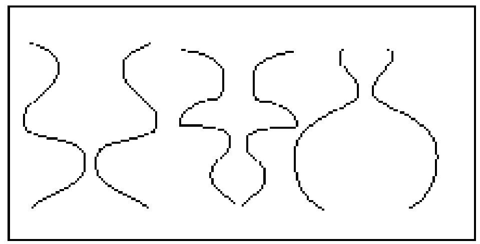
Basic ideas of grouping in humans

- Figure-ground discrimination
  - grouping can be seen in terms of allocating some elements to a figure, some to ground
  - impoverished theory
- Gestalt properties
  - A series of factors affect whether elements should be grouped together

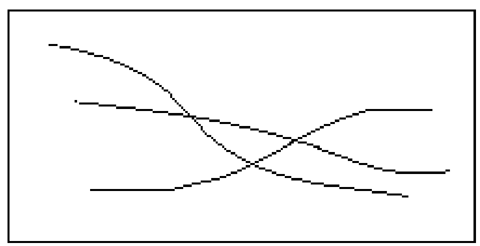

Not grouped
Proximity
Similarity
Similarity
Common Fate
Common Region



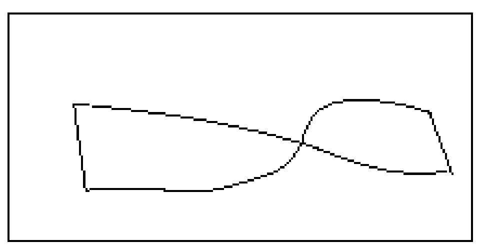
Parallelism



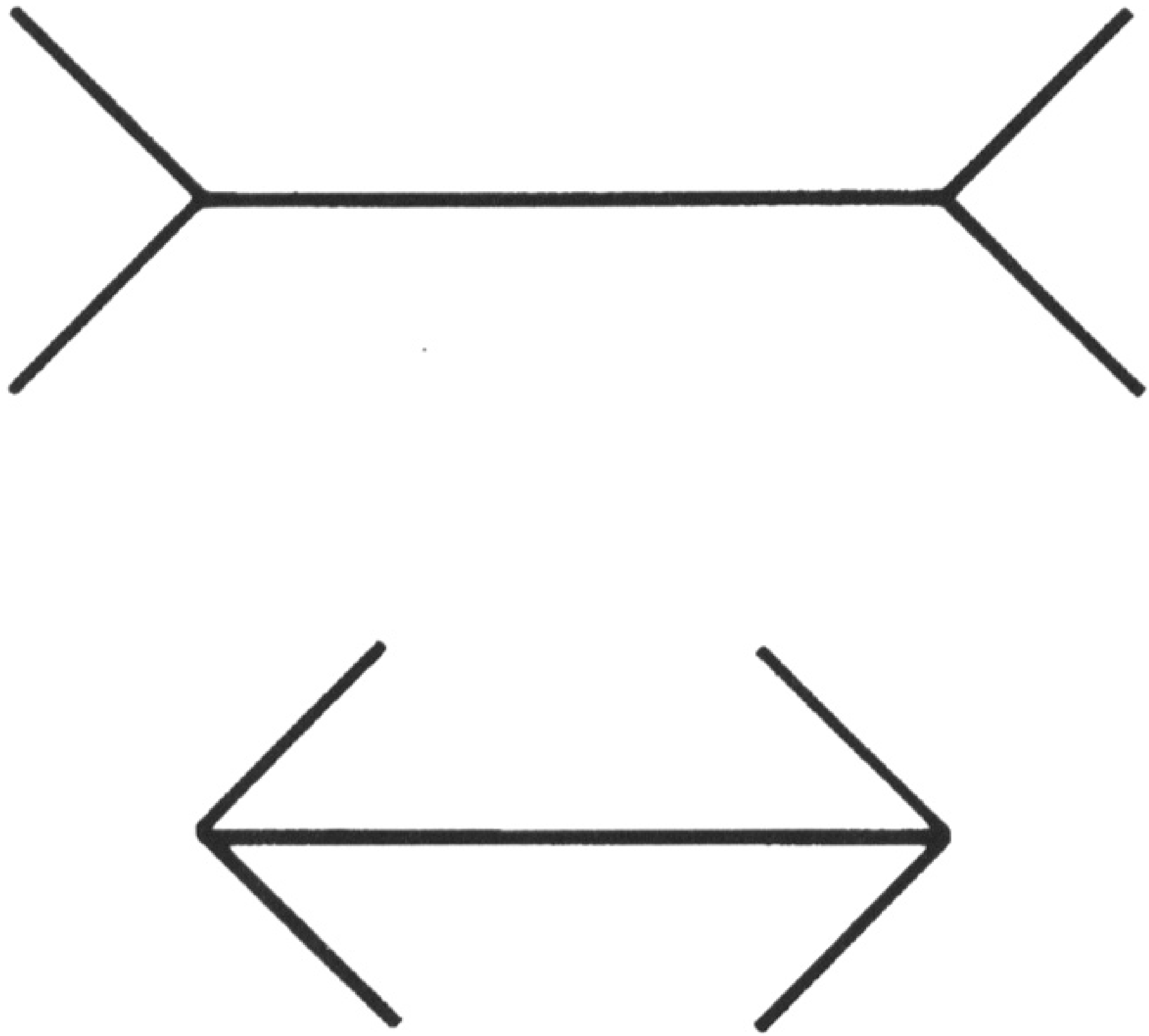
Symmetry

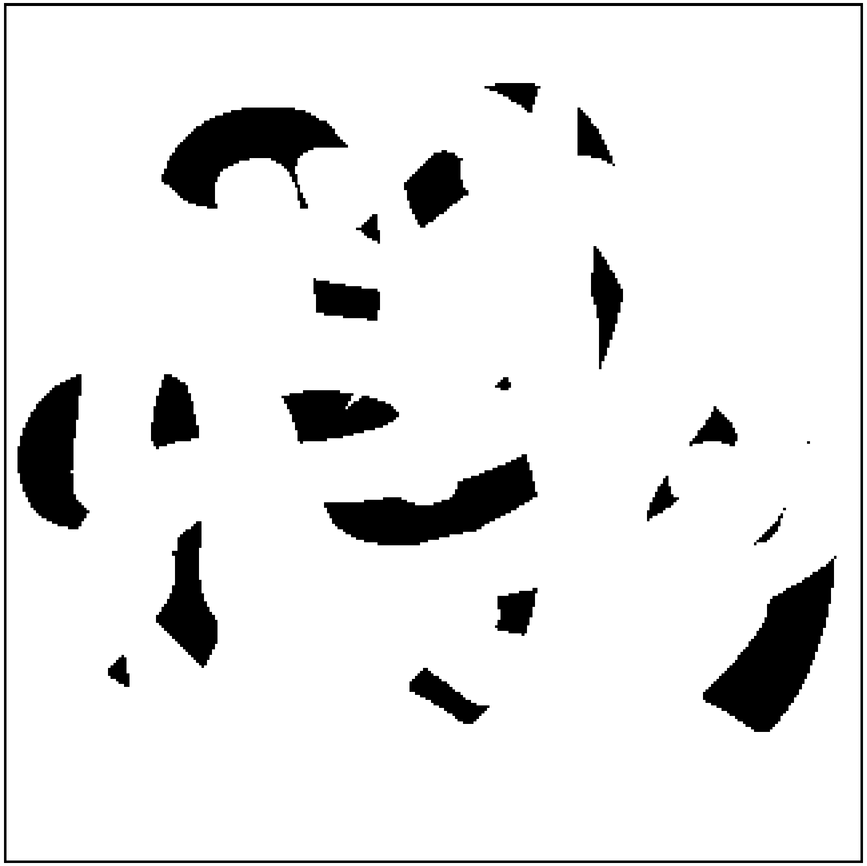


Continuity



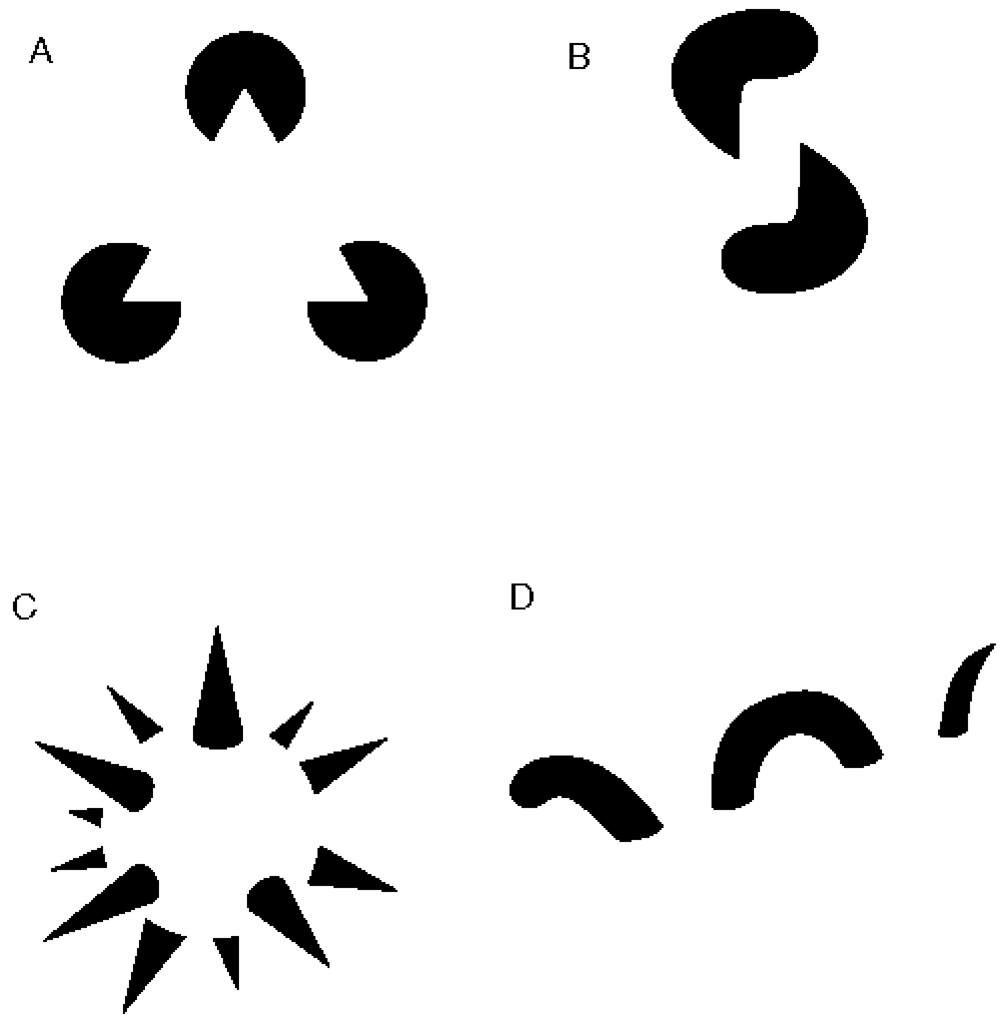
Closure





Occlusion is an important cue in grouping.

Consequence: Groupings by Invisible Completions



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

And the famous…

And the famous invisible dog eating under a tree:

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

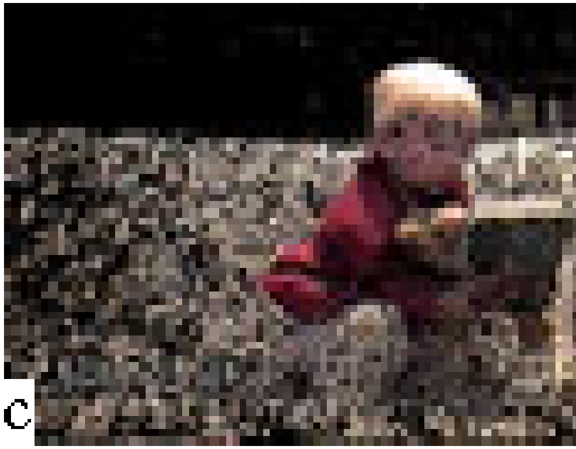
low thresh

high thresh

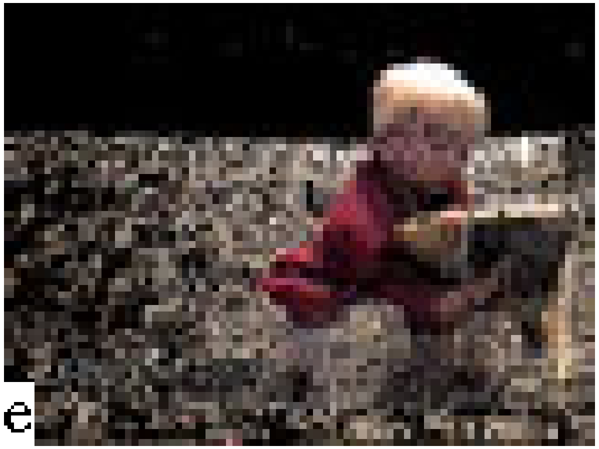
EM (later)

160x120

low thresh



high thresh

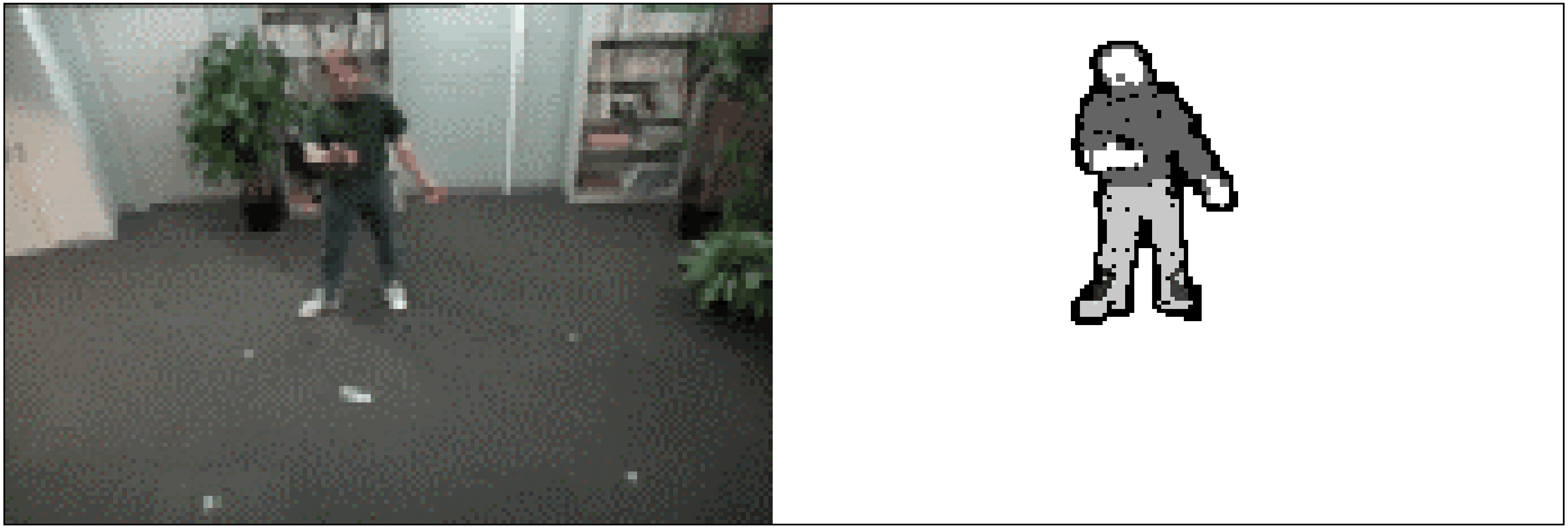


EM (later)

Static Background Modeling Examples

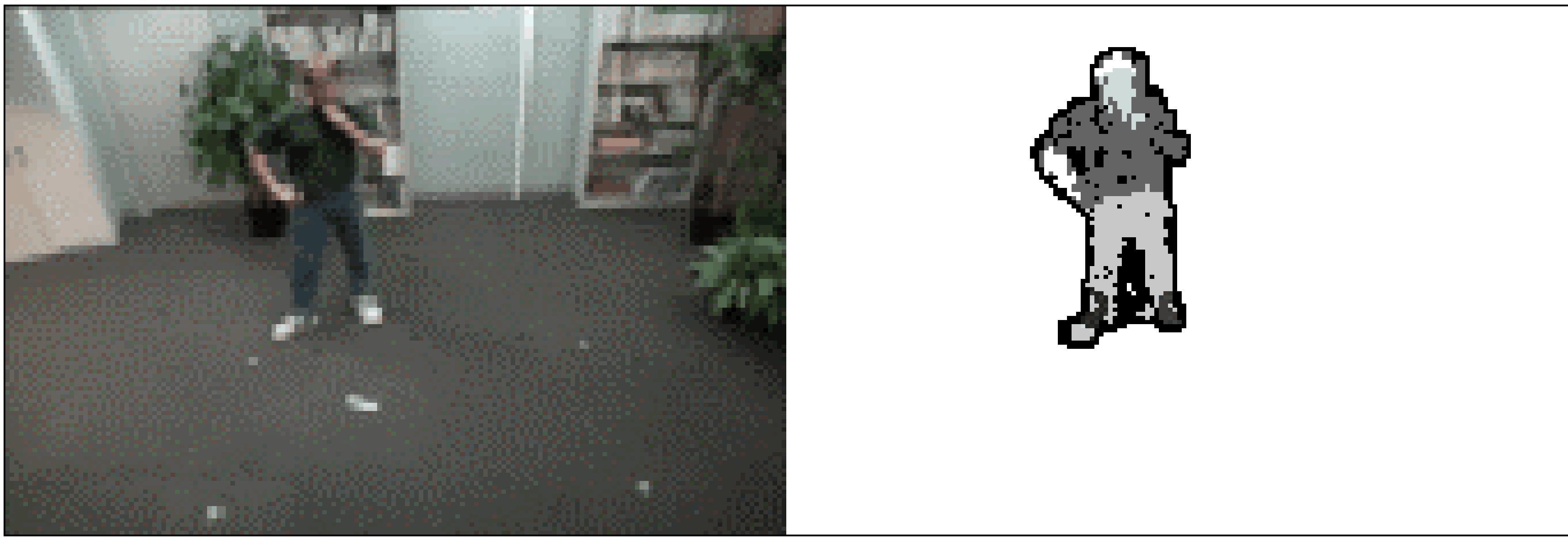
[MIT Media Lab Pfinder / ALIVE System]

Static Background Modeling Examples



[MIT Media Lab Pfinder / ALIVE System]

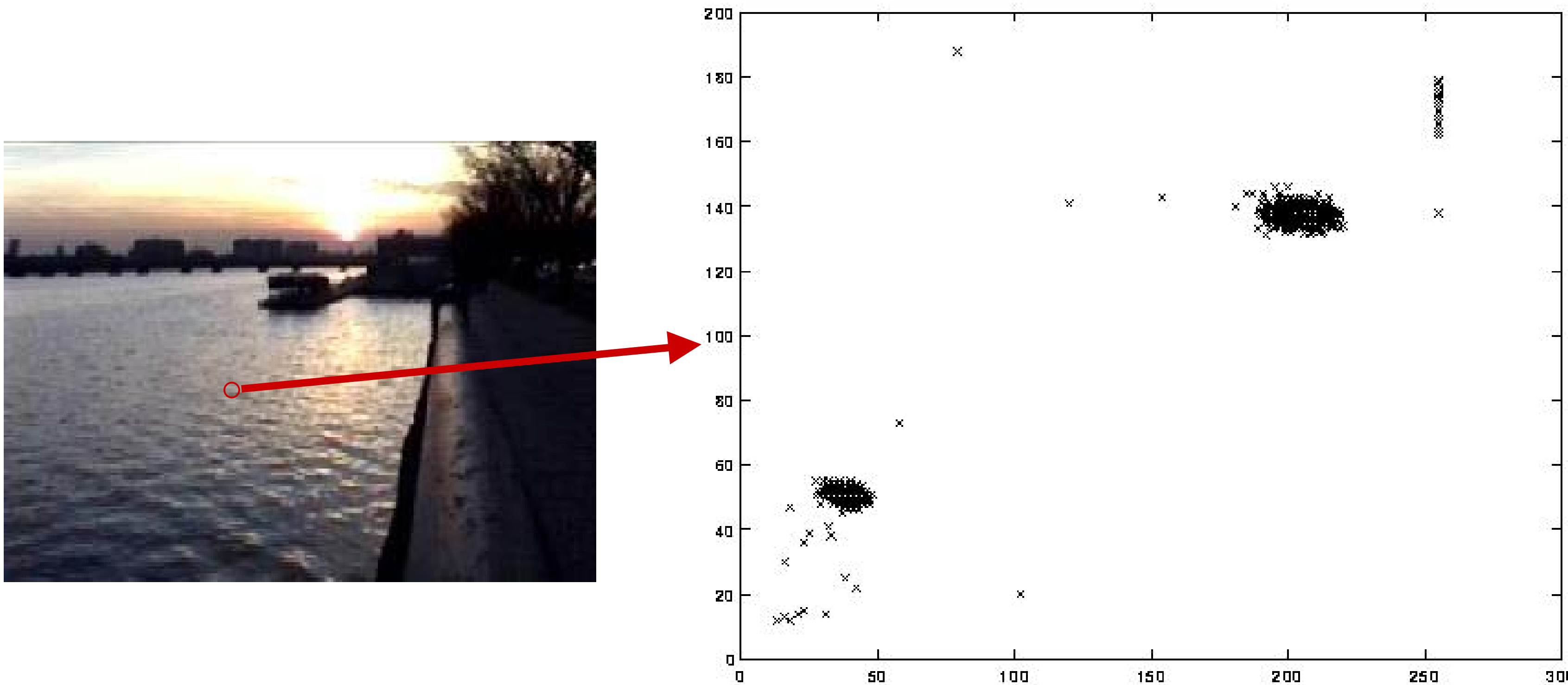
Static Background Modeling Examples



[MIT Media Lab Pfinder / ALIVE System]

Dynamic Background

BG Pixel distribution is non-stationary:

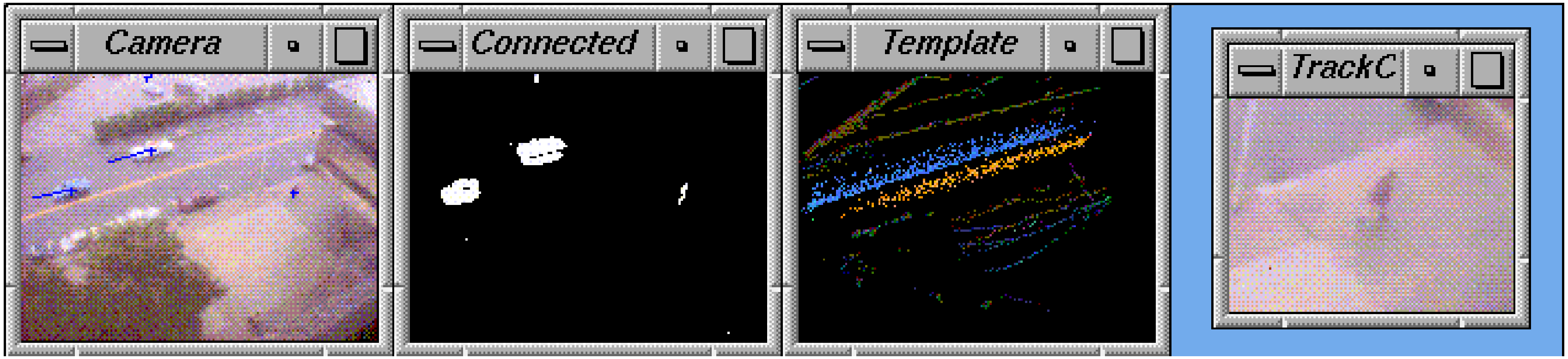


[MIT AI Lab VSAM]

Mixture of Gaussian BG model

Staufer and Grimson tracker:

Fit per-pixel mixture model to observed distrubution.



[MIT AI Lab VSAM]

Background Subtraction Principles

Wallflower: Principles and Practice of Background Maintenance, by Kentaro Toyama, John Krumm, Barry Brumitt, Brian Meyers.

P1: Semantic differentiation of objects should not be handled by the background maintenance module.

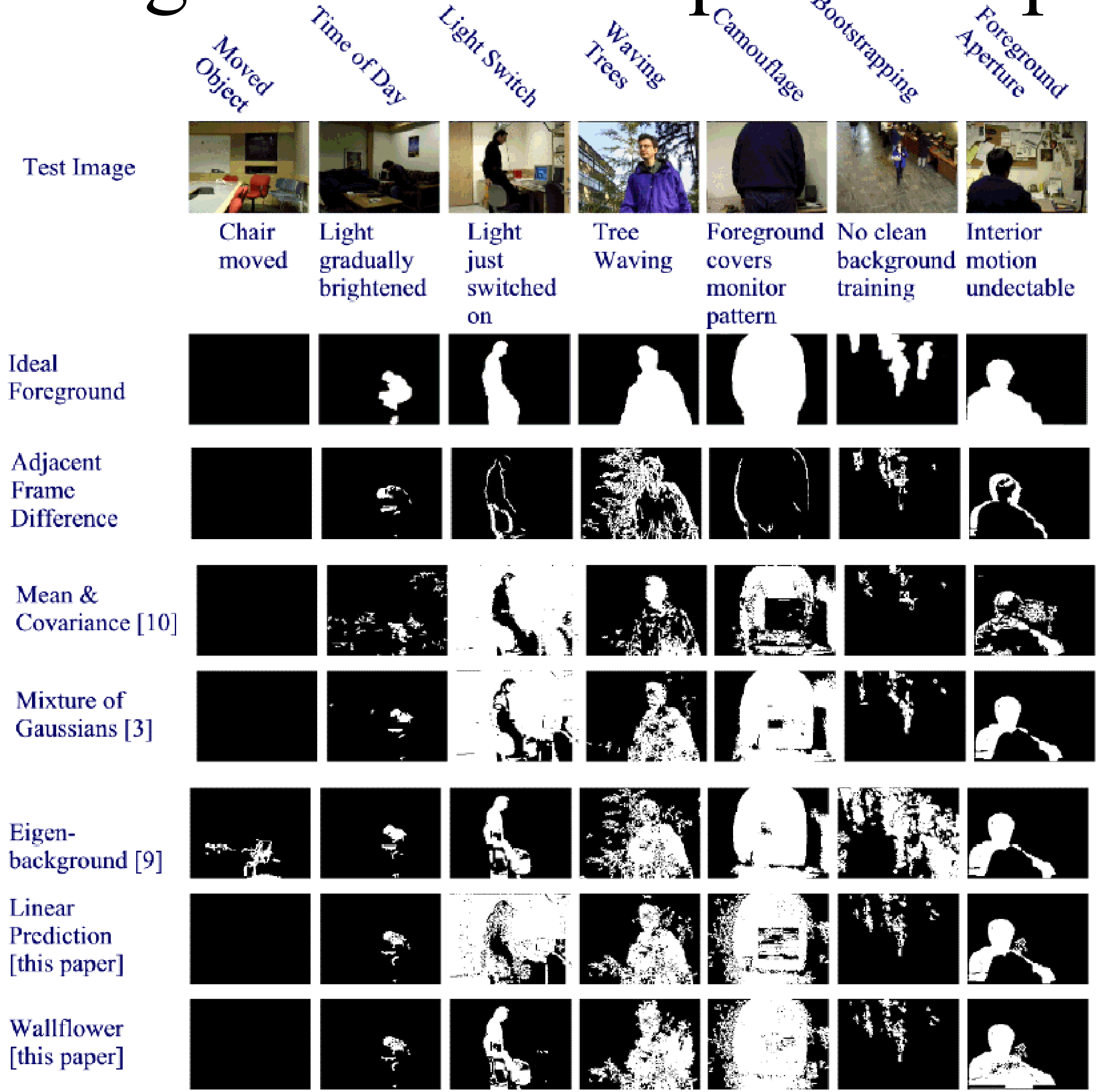
P2: Background subtraction should segment objects of interest when they first appear (or reappear) in a scene.

P3: An appropriate pixel-level stationarity criterion should be defined. Pixels that satisfy this criterion are declared background and ignored.

P4: The background model must adapt to both sudden and gradual changes in the background.

P5: Background models should take into account changes at differing spatial scales.

Background Techniques Compared



From the Wallflower Paper

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

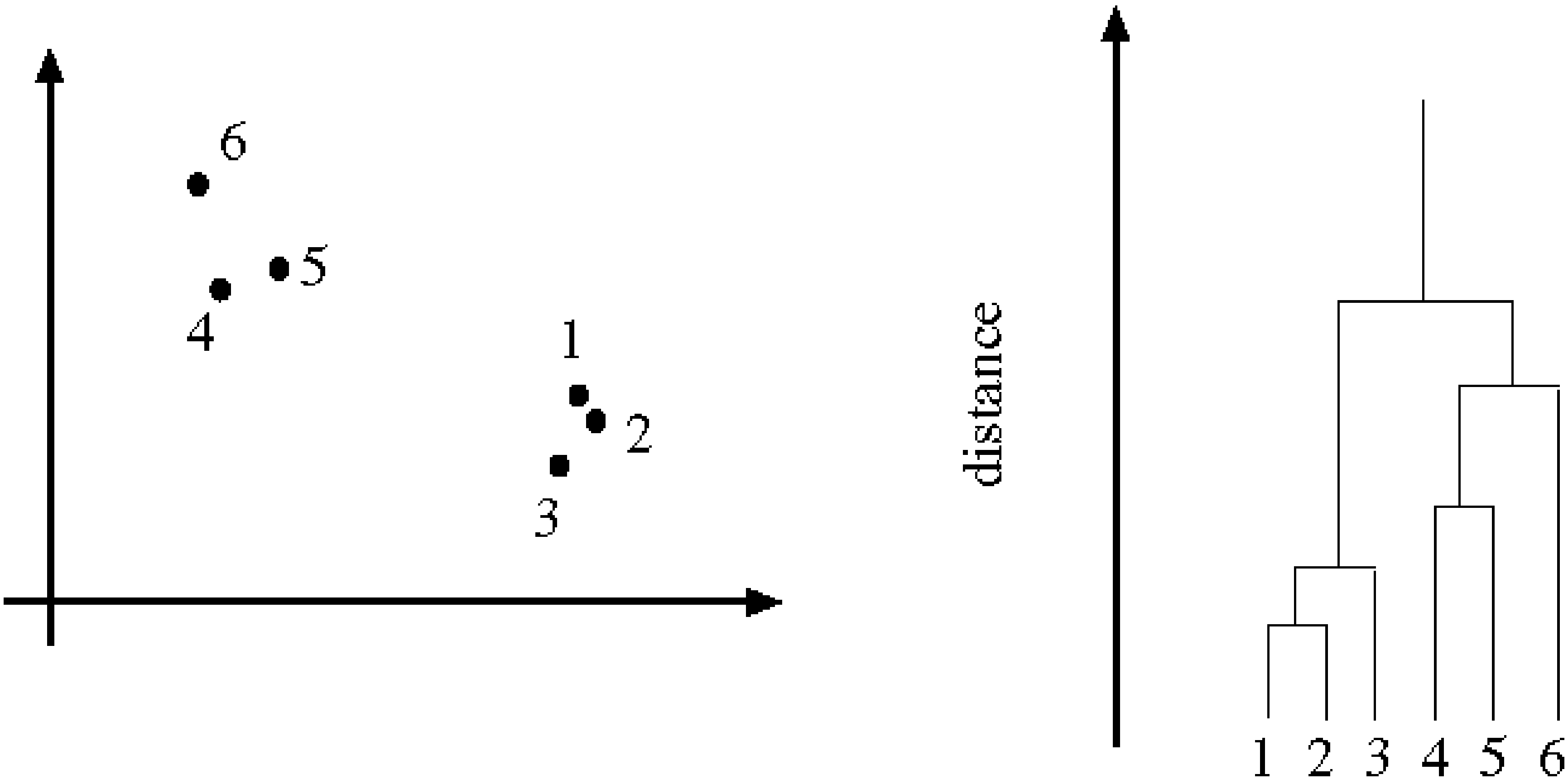
Clustering Algorithms

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)

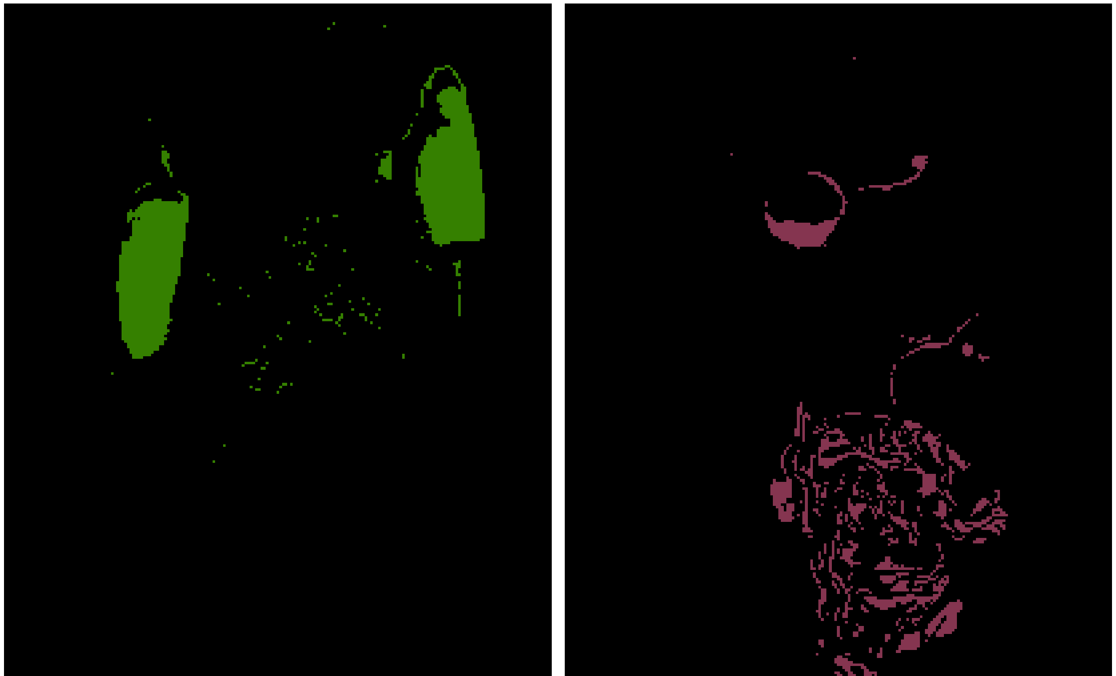
Clusters on color (K=5)

K-means clustering using intensity alone and color alone

Image

Clusters on color

K-means using color alone, 11 segments



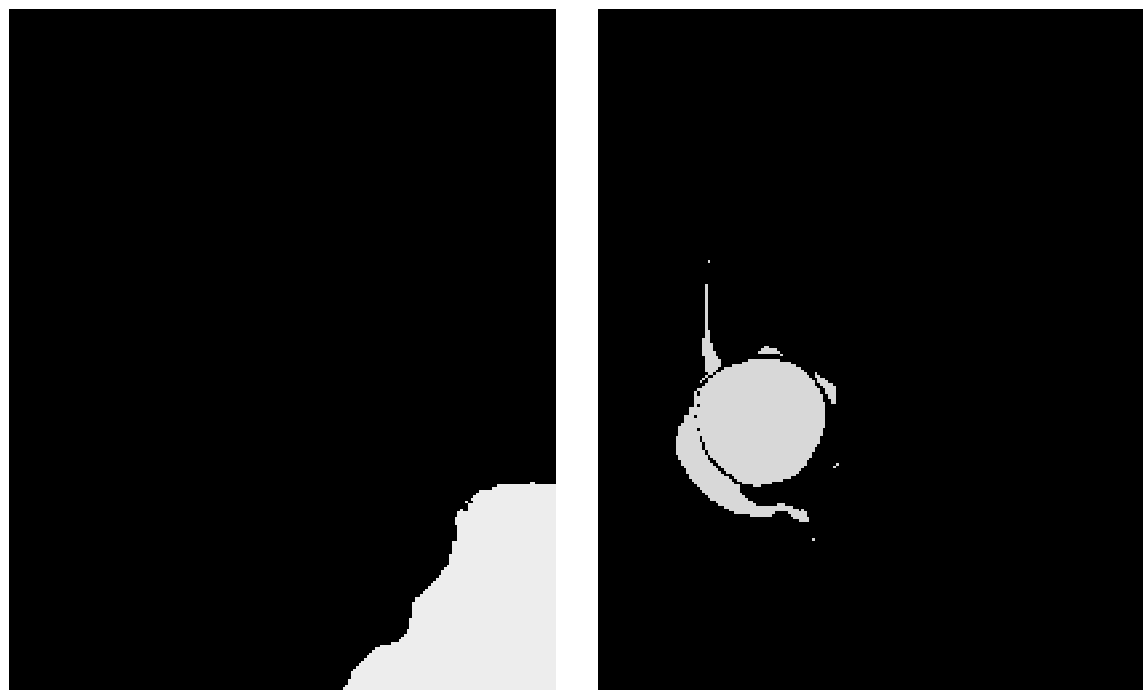
K-means using
color alone,
11 segments.

***Color alone
often will not
yeild salient segments!***

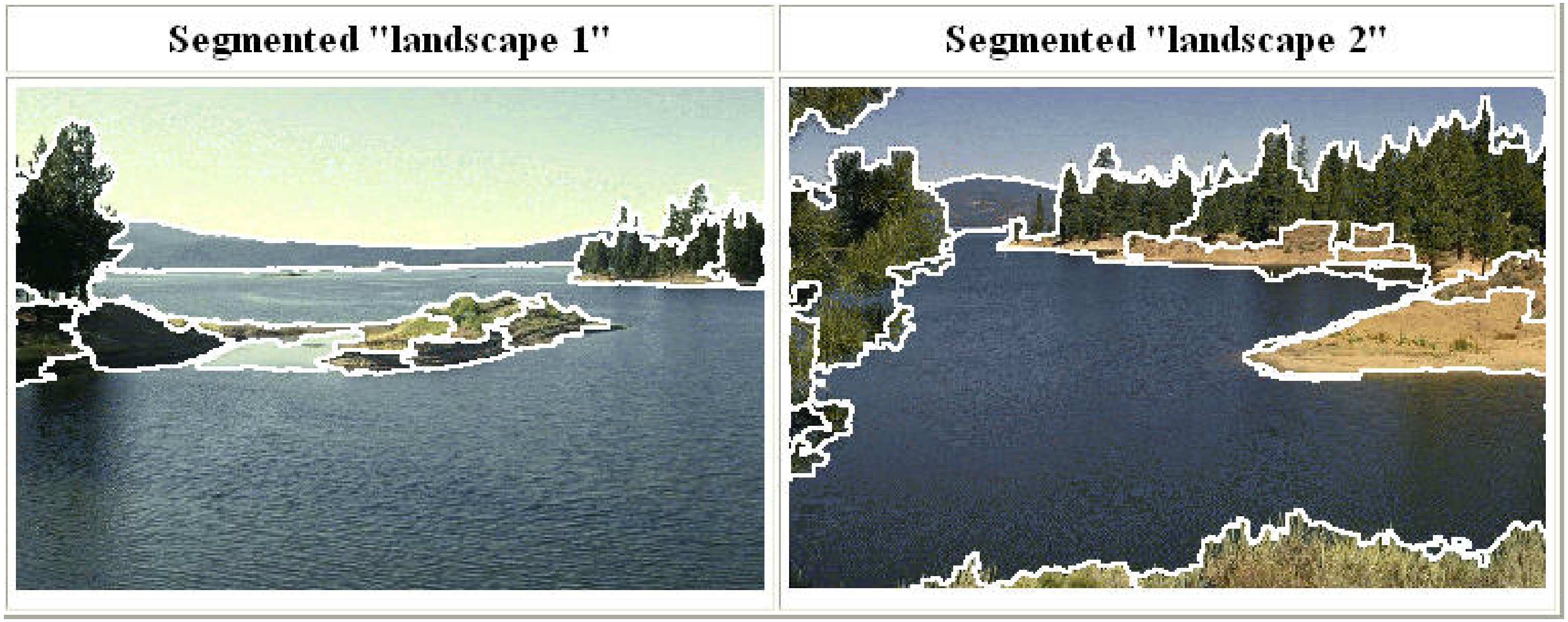
K-means using colour and position, 20 segments

Still misses goal of perceptually pleasing segmentation!

Hard to pick K…



Mean Shift Segmentation



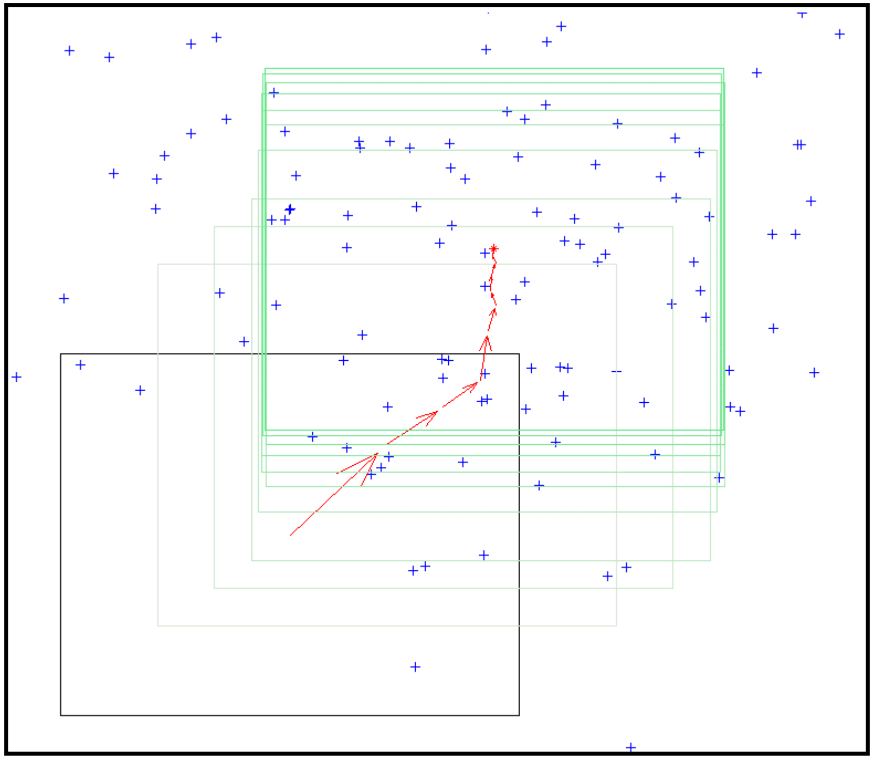
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

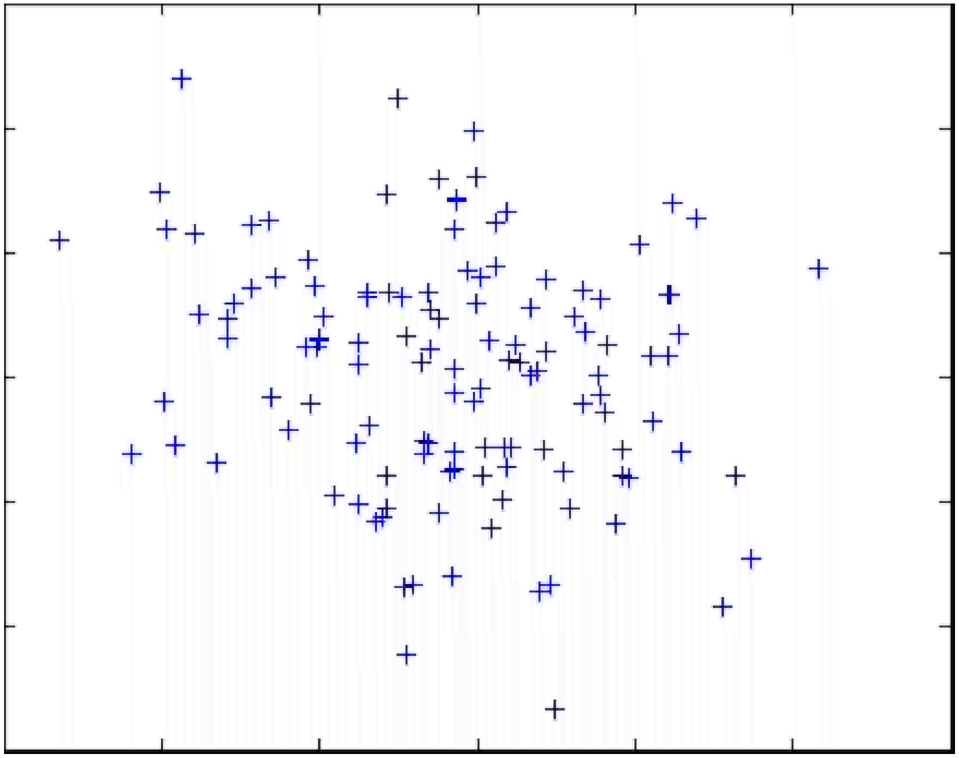
Mean Shift Algorithm

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:

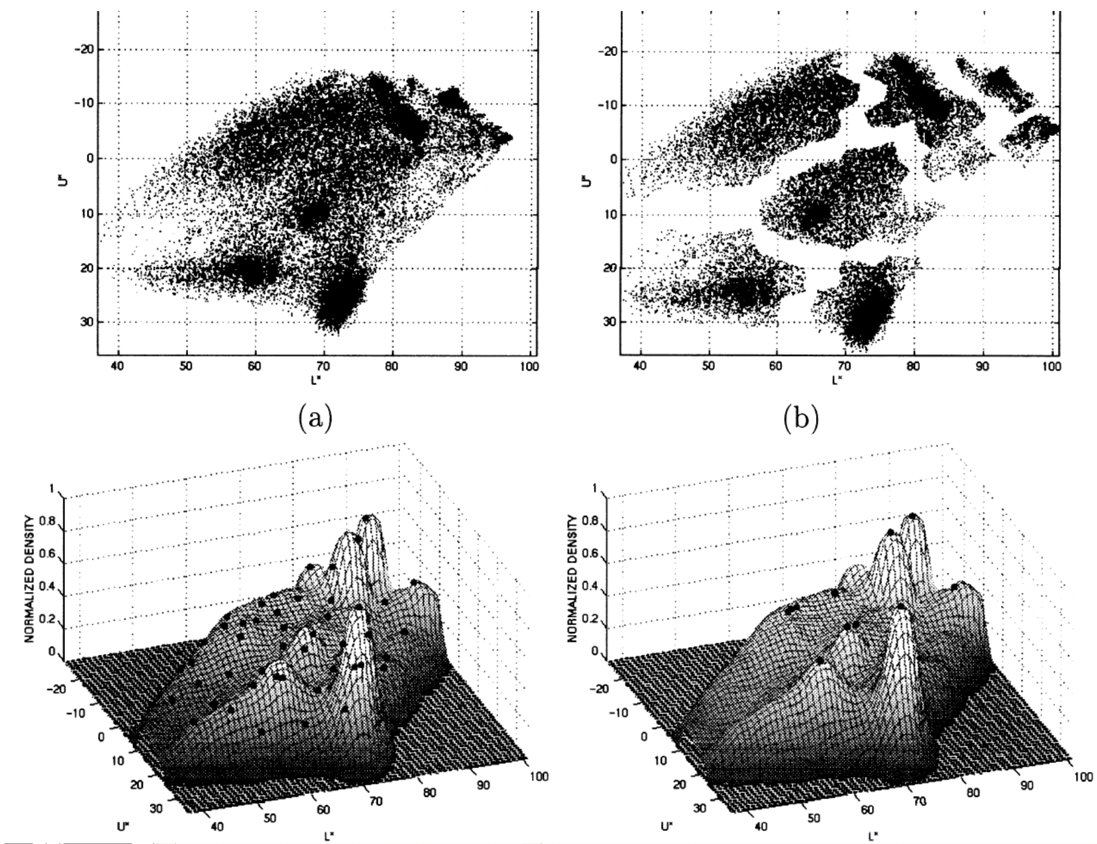




Mean Shift Segmentation

Mean Shift Setmentation Algorithm

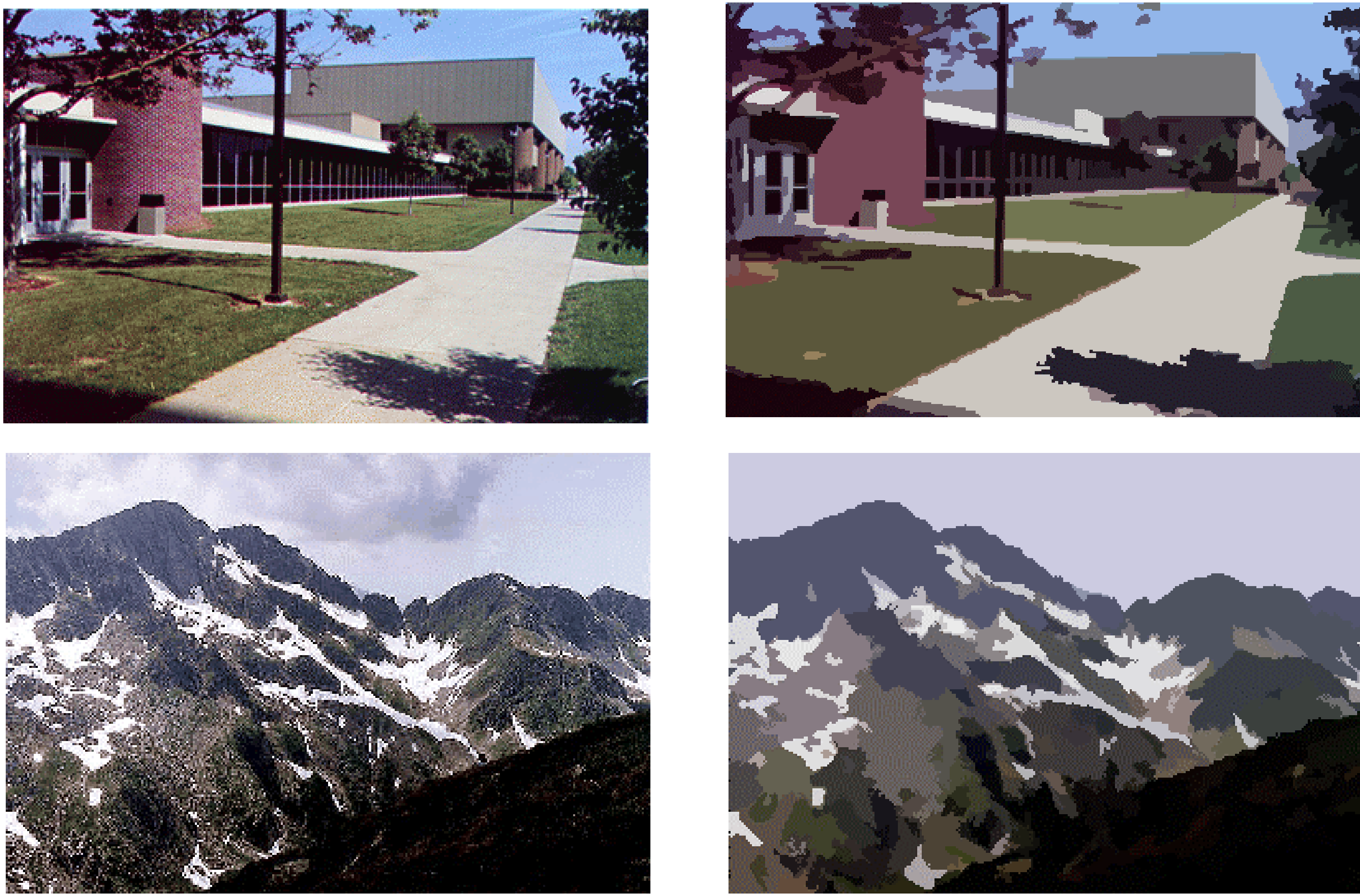
1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same “peak” or mode.
5. The data these merged windows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



(a) (b)

*Image From: Dorin Comaniciu and Peter Meer, Distribution Free Decomposition of Multivariate Data, Pattern Analysis & Applications (1999)2:22–30

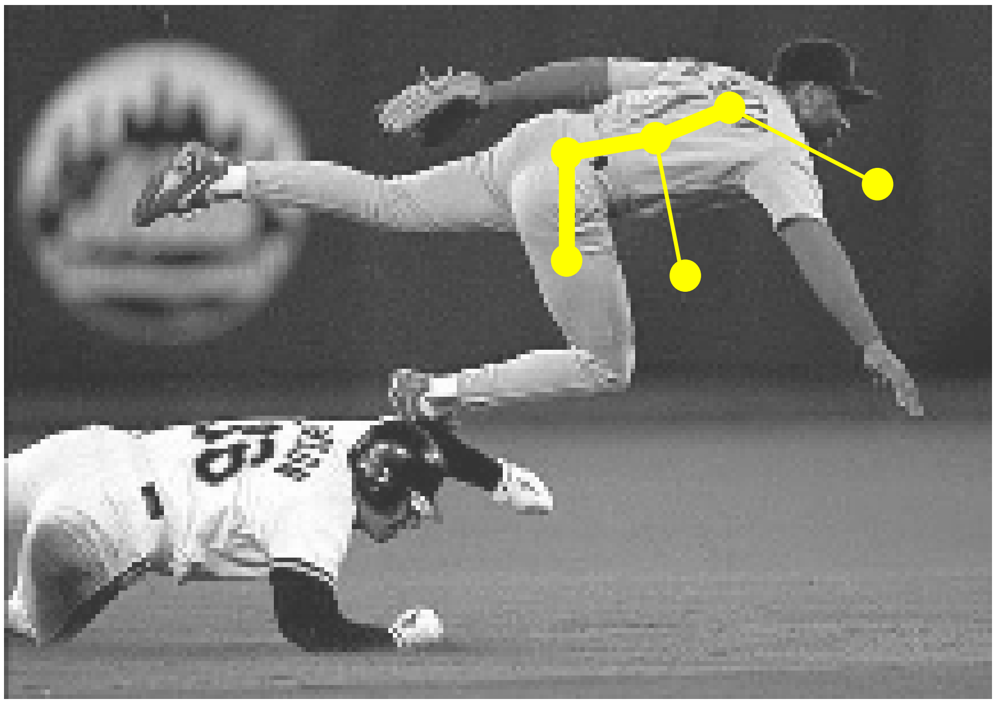
Mean Shift Segmentation Results:



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

Graph-Theoretic Image Segmentation

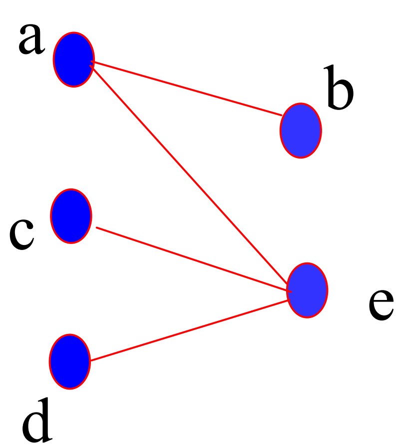
Build a weighted graph G=(V,E) from image



V: image pixels

E: connections between pairs of nearby pixels

Graphs Representations

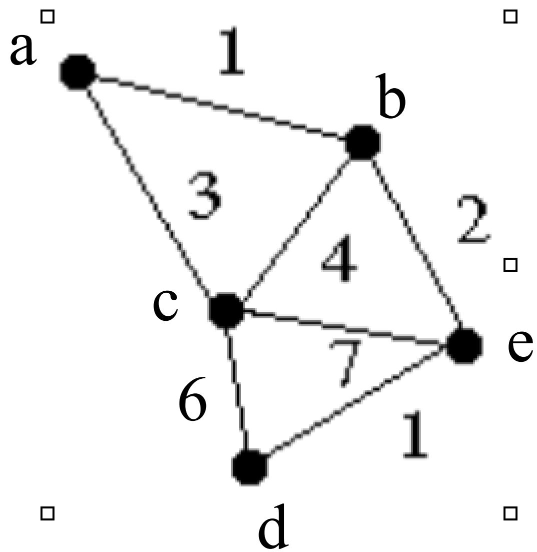


$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Weighted Graphs and Their Representations



$$\begin{bmatrix} 0 & 1 & 3 & \infty & \infty \\ 1 & 0 & 4 & \infty & 2 \\ 3 & 4 & 0 & 6 & 7 \\ \infty & \infty & 6 & 0 & 1 \\ \infty & 2 & 7 & 1 & 0 \end{bmatrix}$$

Weight Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

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 affinity clusters

- 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

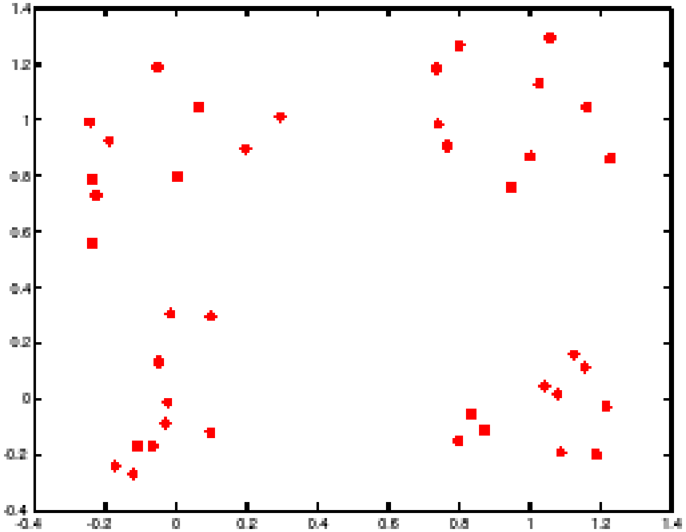
$$a^T A a$$

- But need the constraint

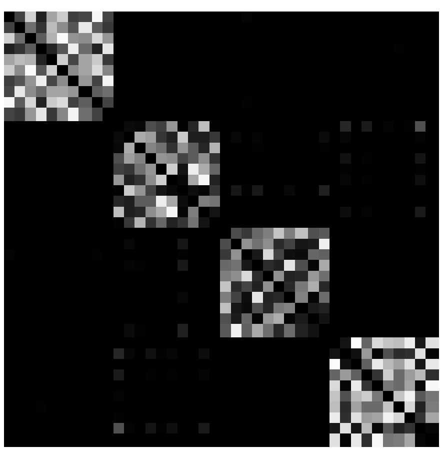
$$a^T a = 1$$

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

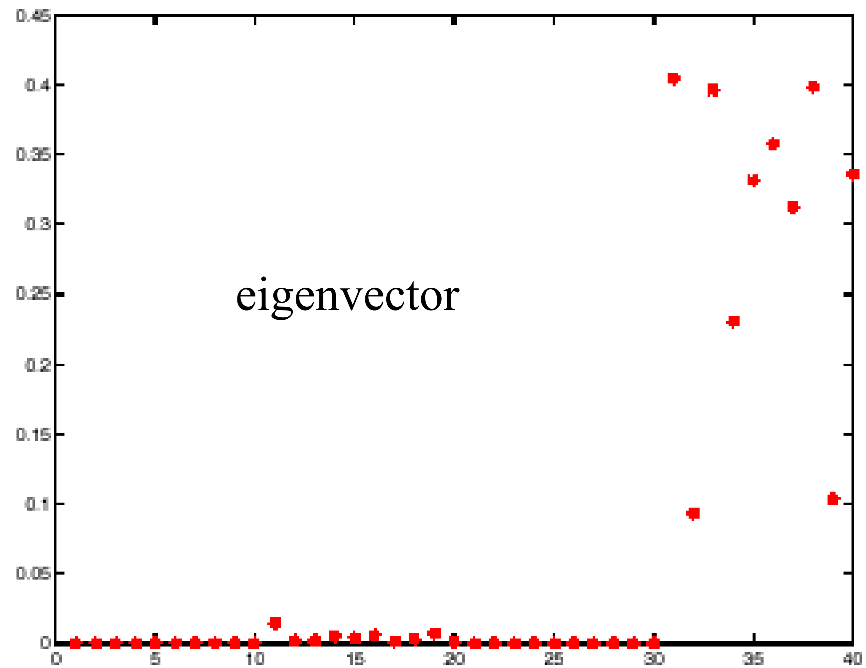
Example eigenvector



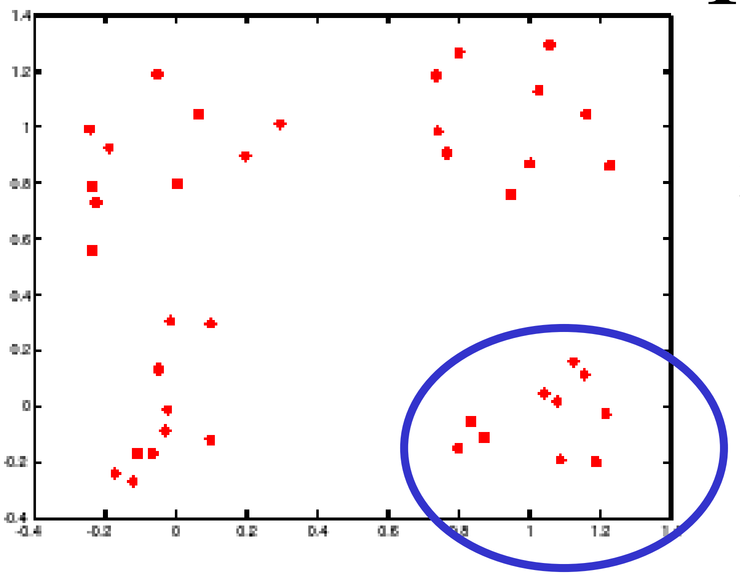
points



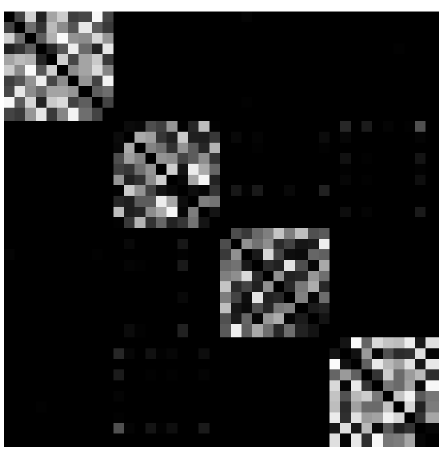
matrix



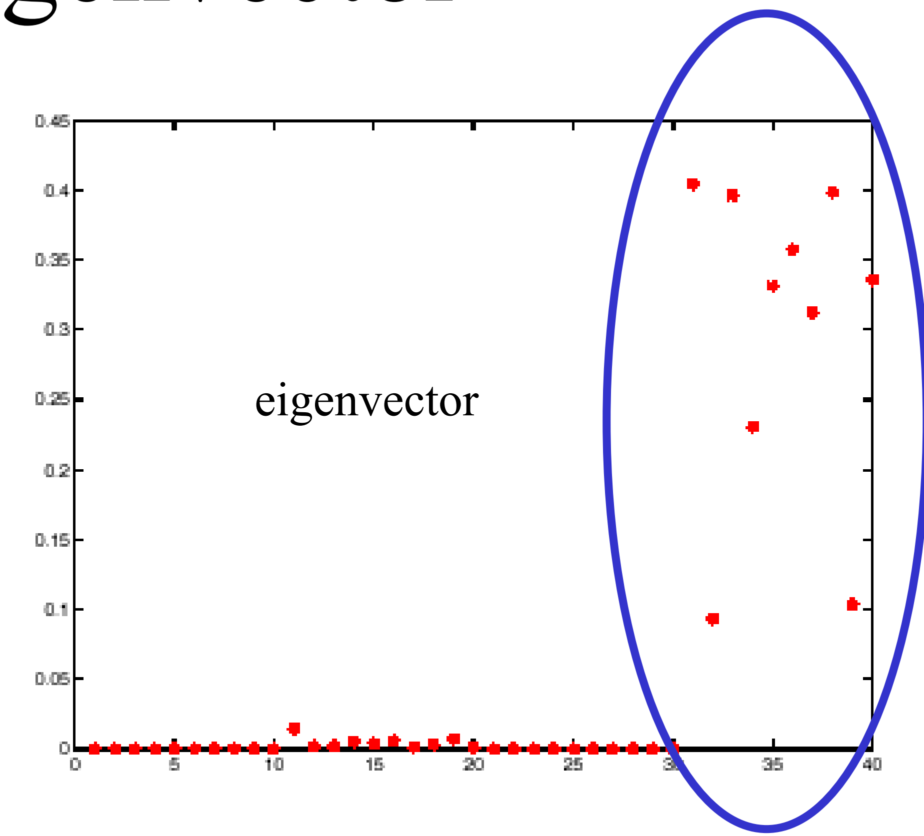
Example eigenvector



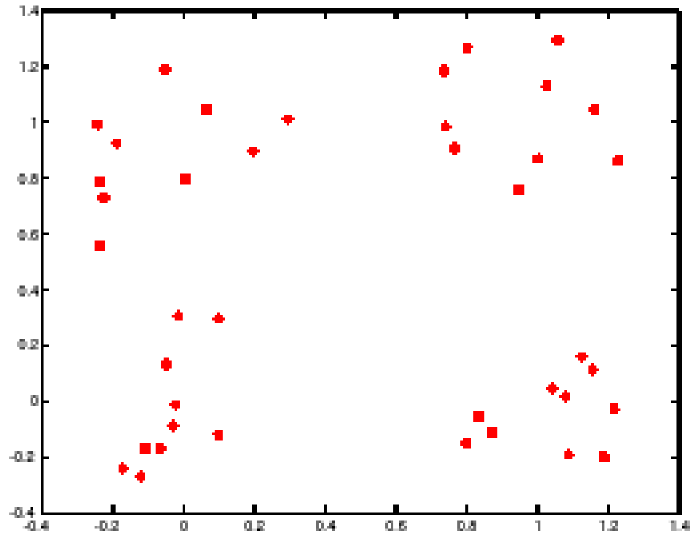
points



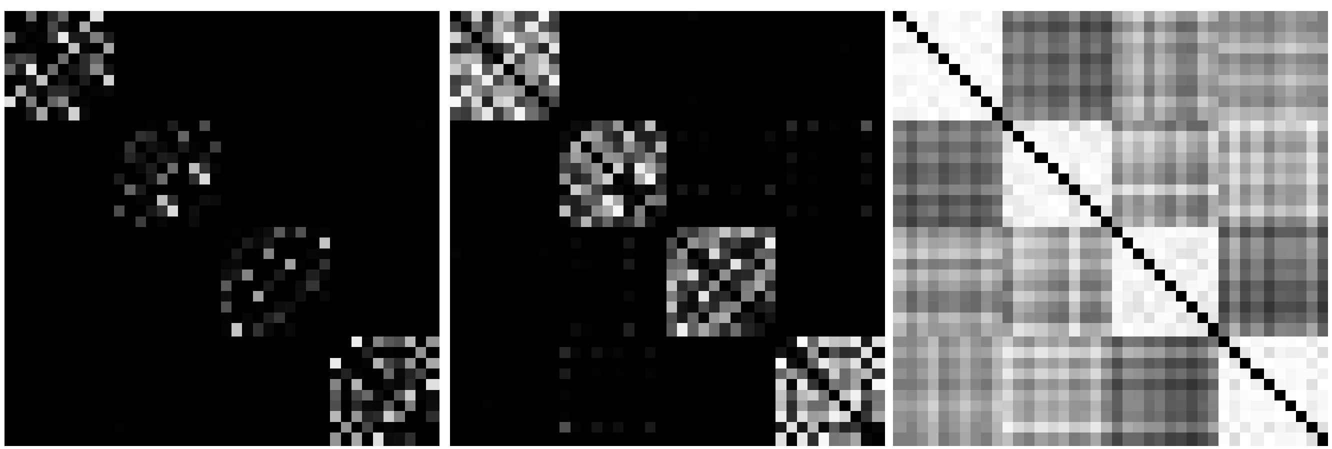
matrix



Scale affects affinity

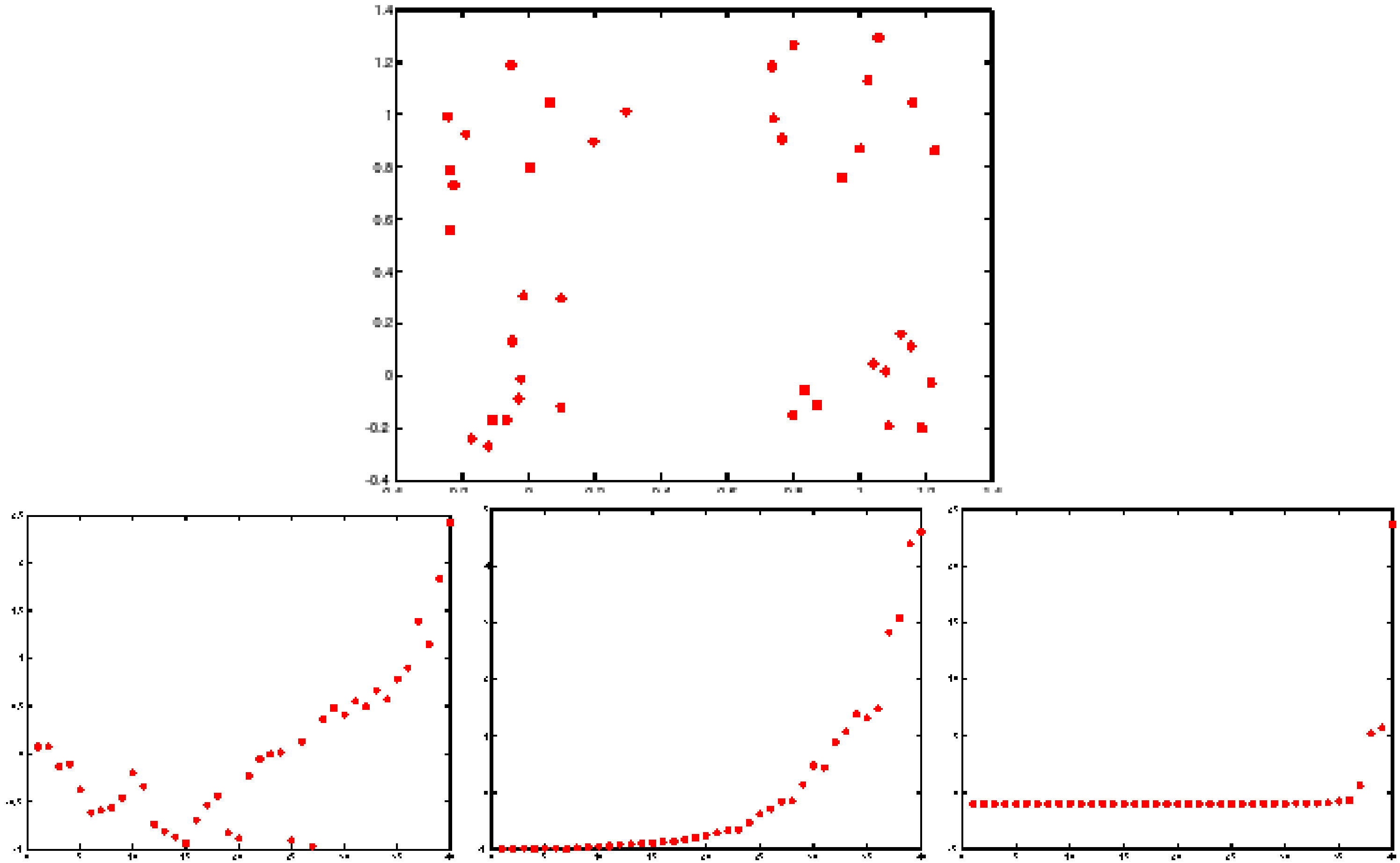


σ=.2



σ=.1 σ=.2 σ=1

Scale affects affinity

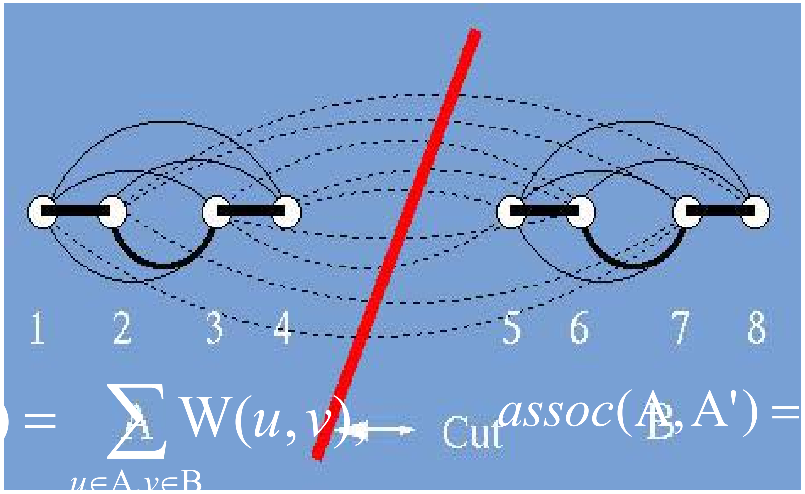


σ=.1 σ=.2 σ=1

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

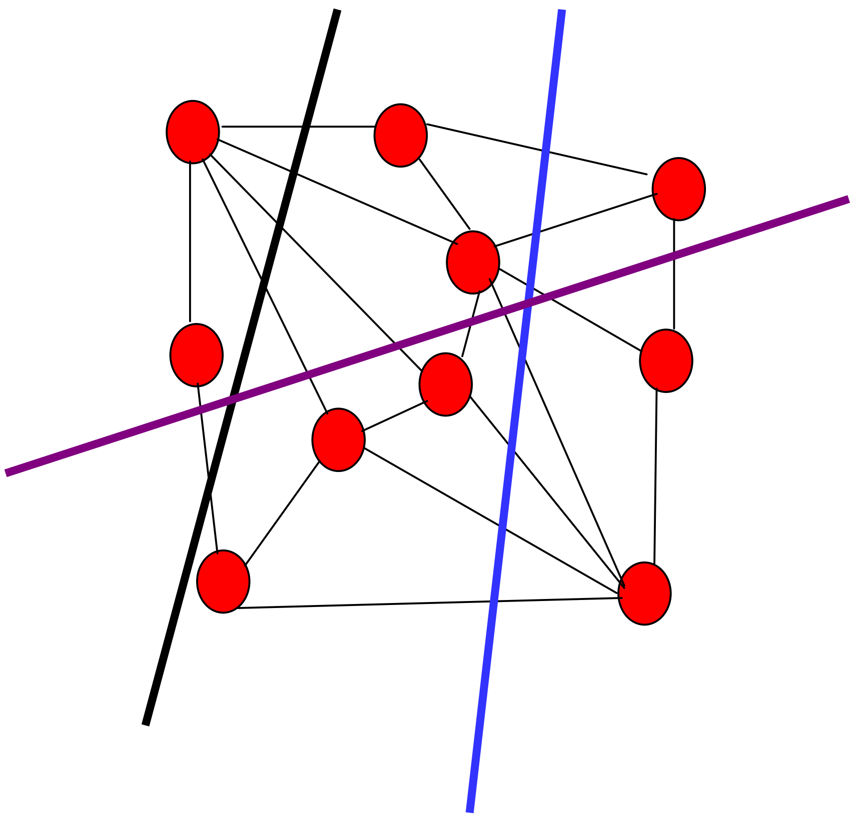
Some Terminology for Graph Partitioning

- How do we bipartition a graph:



[Malik]

Minimum Cut



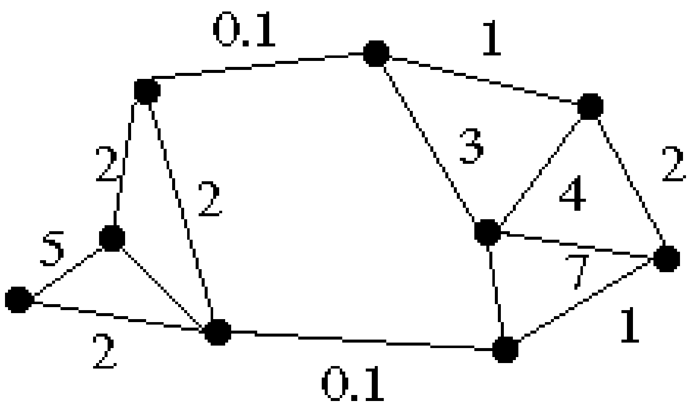
A cut of a graph G is the set of edges S such that removal of S from G disconnects G.

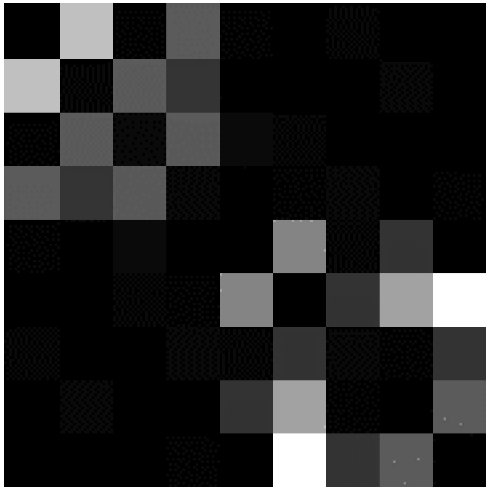
Minimum cut is the cut of minimum weight, where weight of cut <A,B> is given as

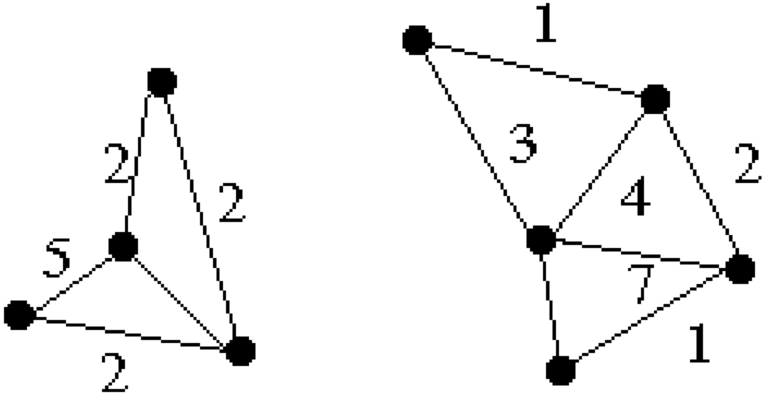
$$w(\langle A, B \rangle) = \sum_{x \in A, y \in B} w(x, y)$$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Minimum Cut and Clustering



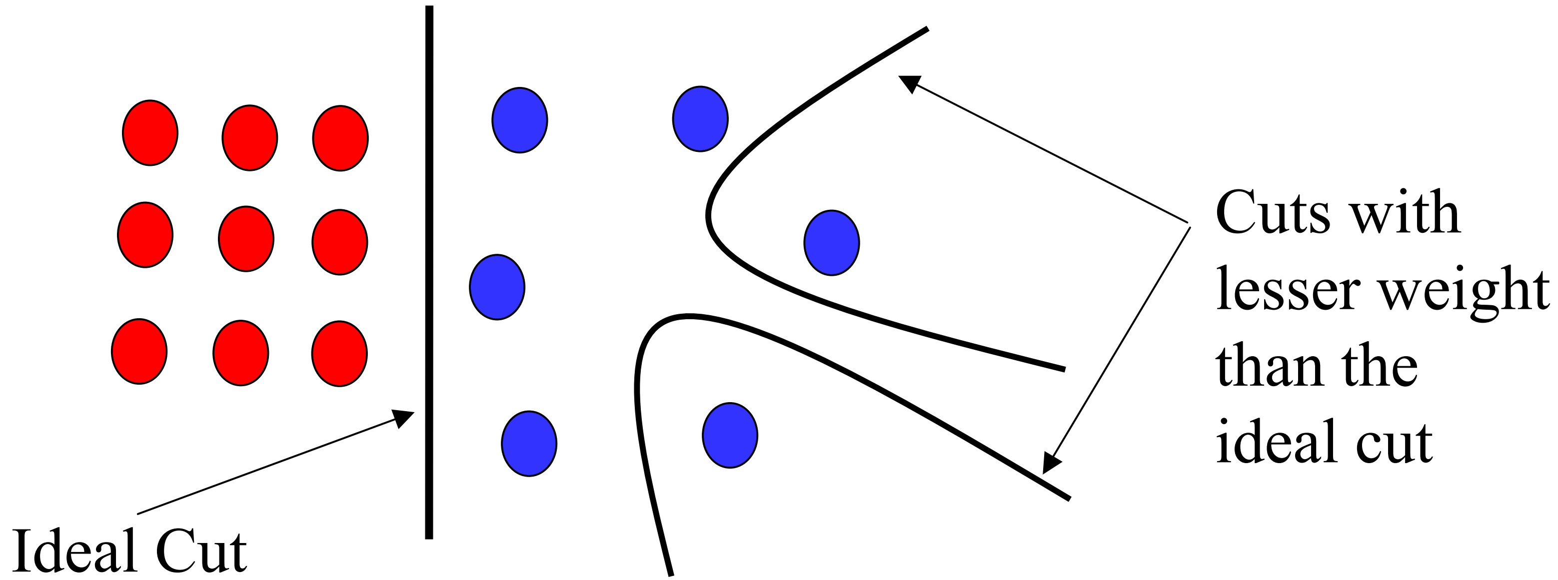




* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Drawbacks of Minimum Cut

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



Normalized cuts

- First eigenvector of affinity matrix captures within cluster similarity, but not across cluster difference
- Min-cut can find degenerate clusters
- 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

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

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

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 Cut As Generalized Eigenvalue problem

$$
\begin{aligned}
Ncut(\mathrm{A},\mathrm{B}) &= \frac{cut(\mathrm{A},\mathrm{B})}{assoc(\mathrm{A},\mathrm{V})} + \frac{cut(\mathrm{A},\mathrm{B})}{assoc(\mathrm{B},\mathrm{V})} \\
&= \frac{(1+x)^T(D-W)(1+x)}{k1^T D1} + \frac{(1-x)^T(D-W)(1-x)}{(1-k)1^T D1};\ k = \frac{\sum_{x_i>0} D(i,i)}{\sum_i D(i,i)} \\
&= \ldots
\end{aligned}
$$

- 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

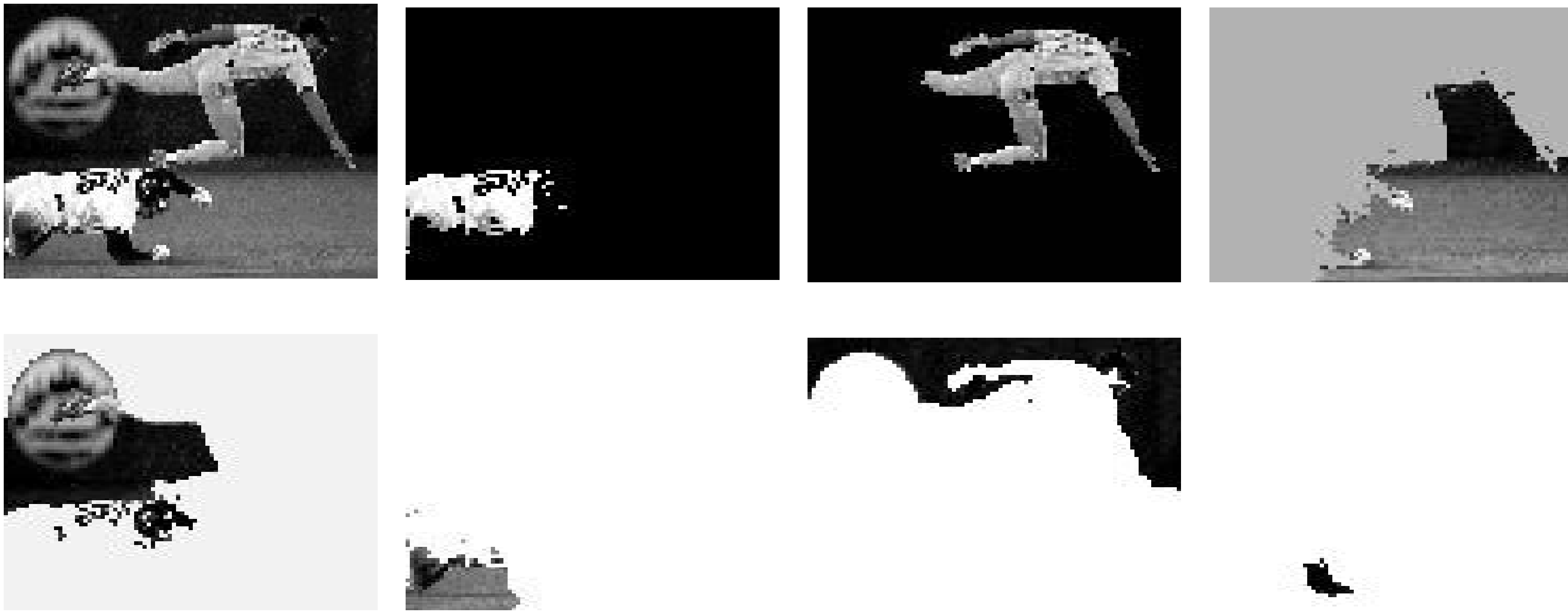
$$\max_y \left(y^T (D - W) y\right) \text{ subject to } \left(y^T D y = 1\right)$$

- which gives

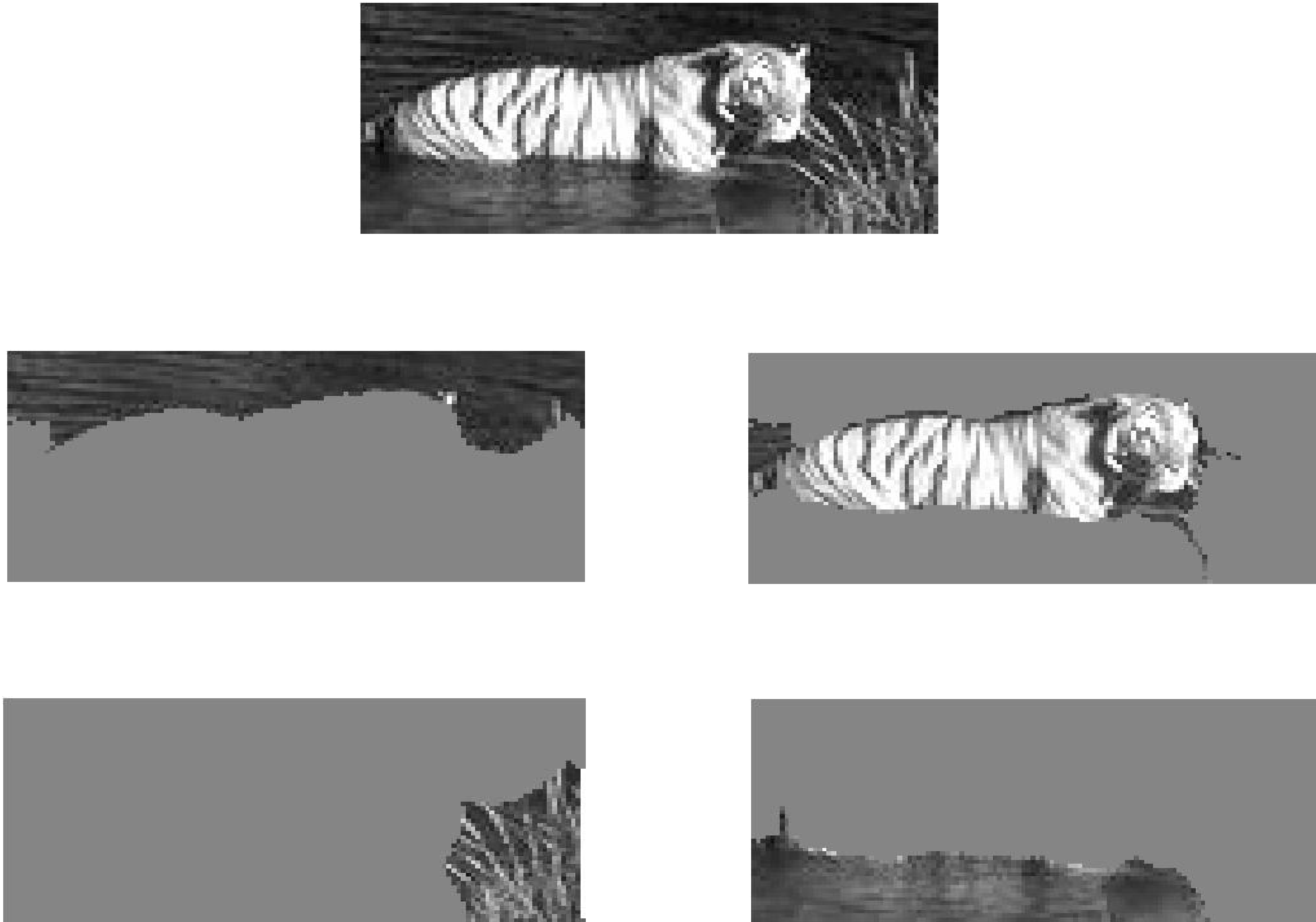
$$(D - W)y = \lambda D y$$

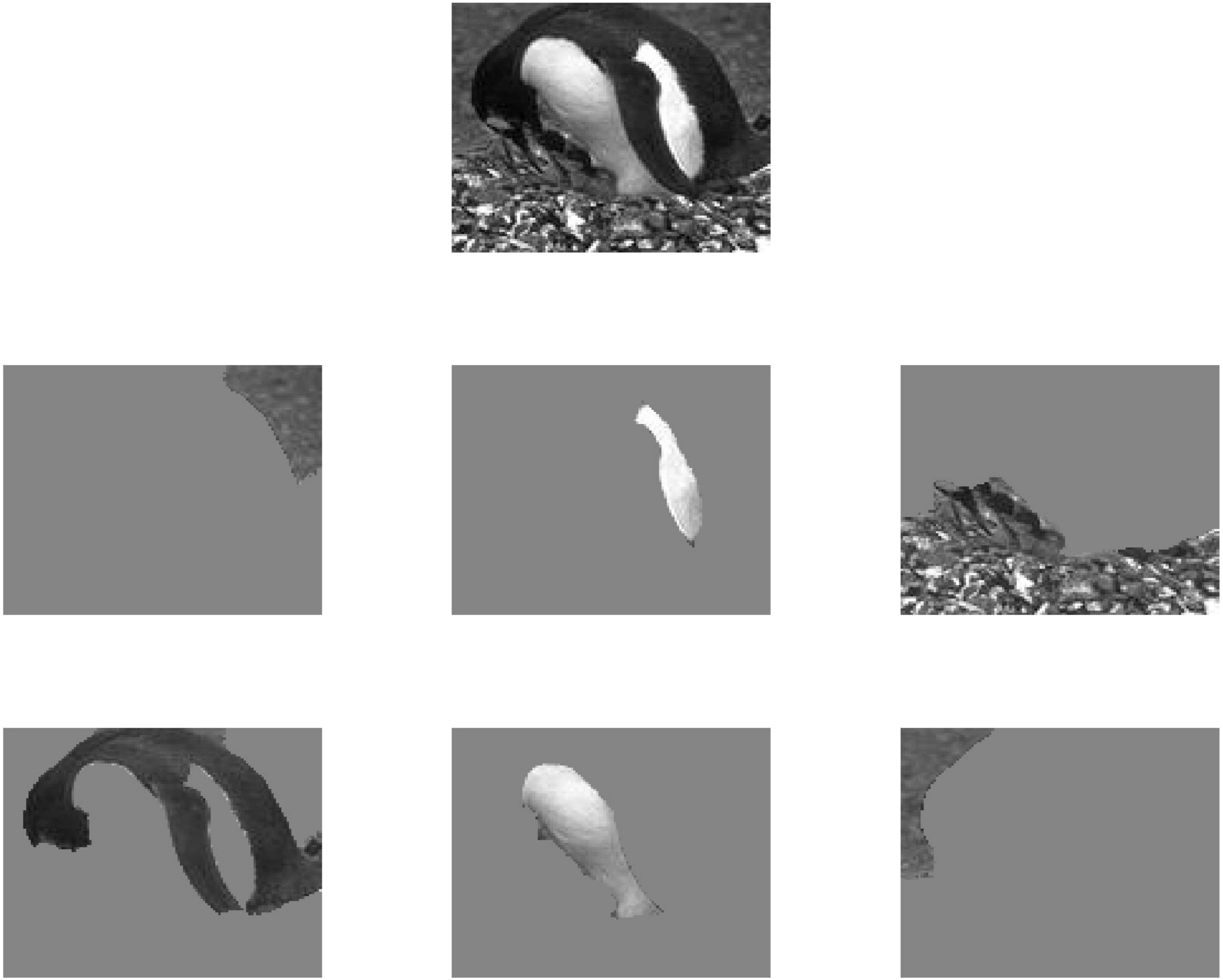
- Now look for a quantization threshold that maximizes the criterion --- i.e all components of y above that threshold go to one, all below go to -b

Brightness Image Segmentation

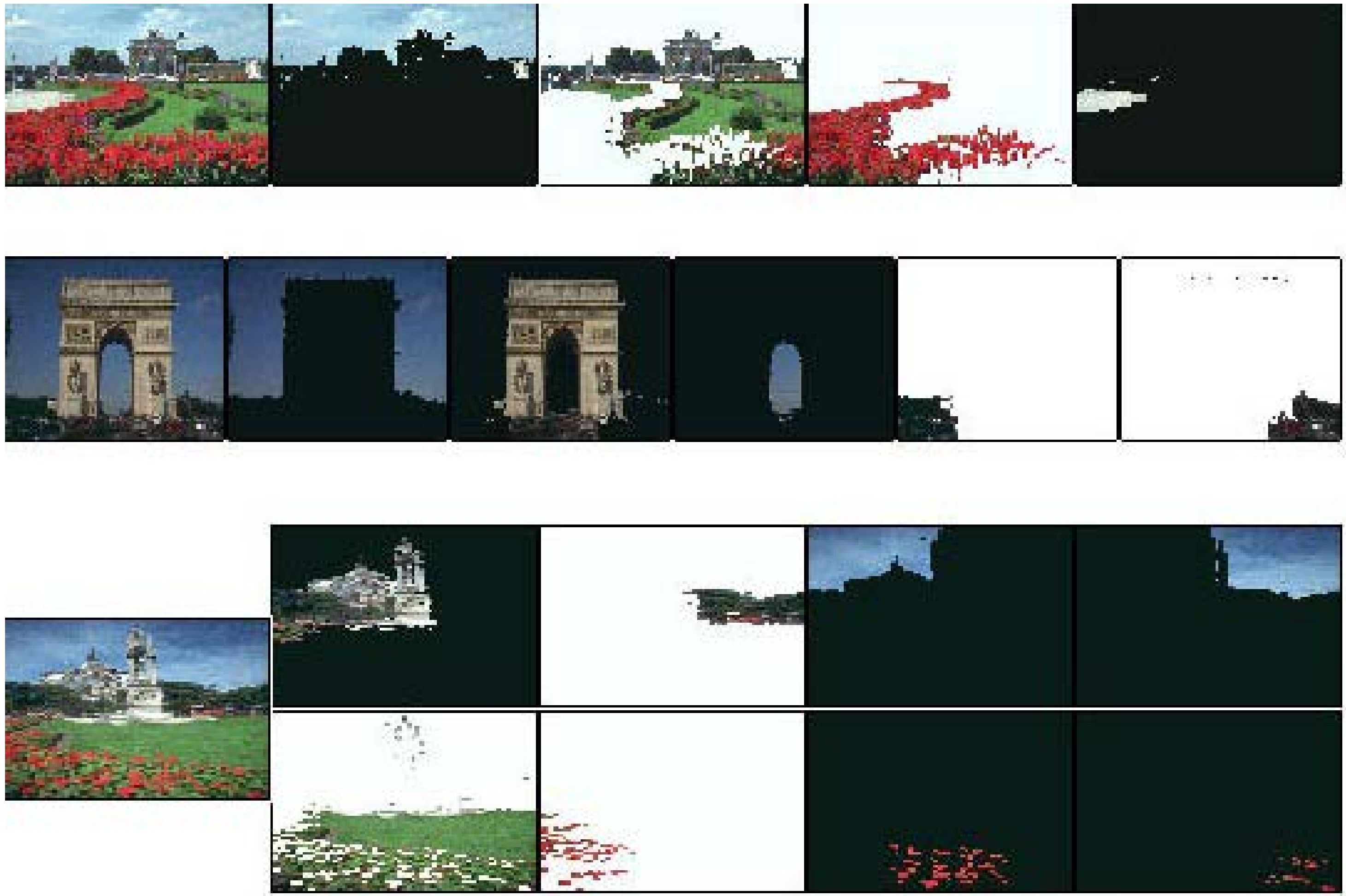


Brightness Image Segmentation



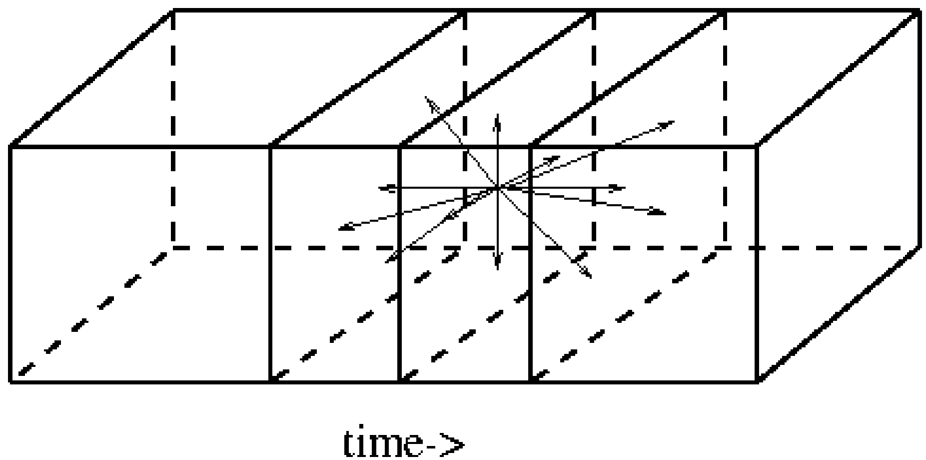


Results on color segmentation

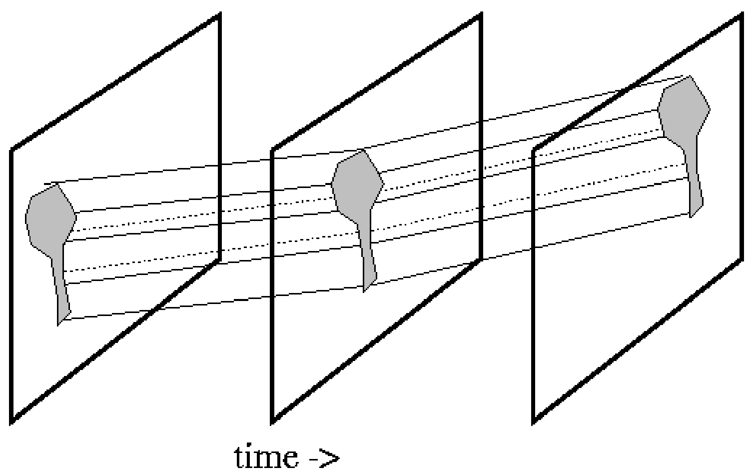


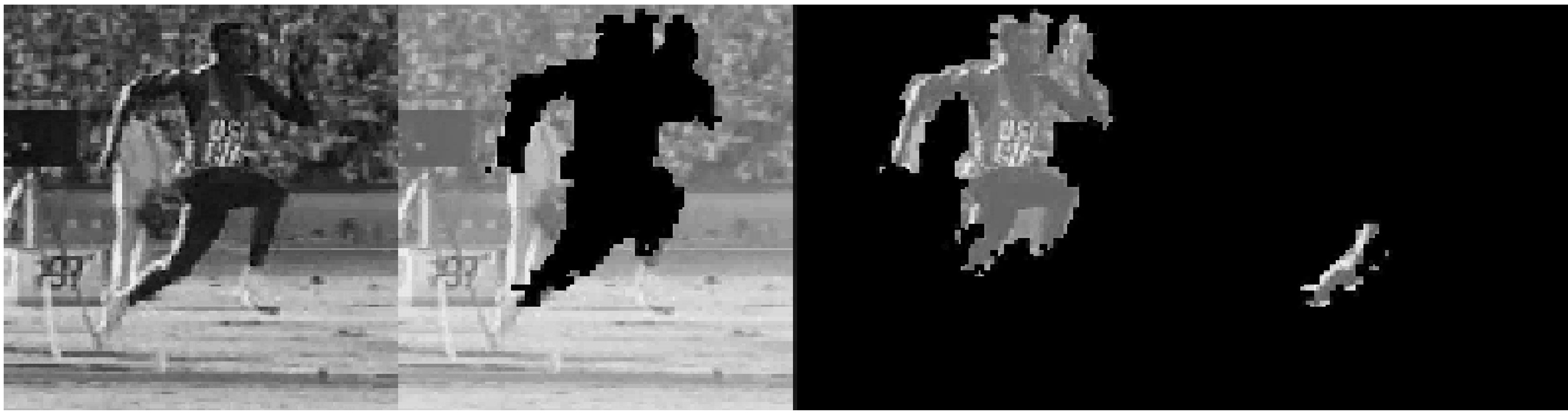
Motion Segmentation with Normalized Cuts

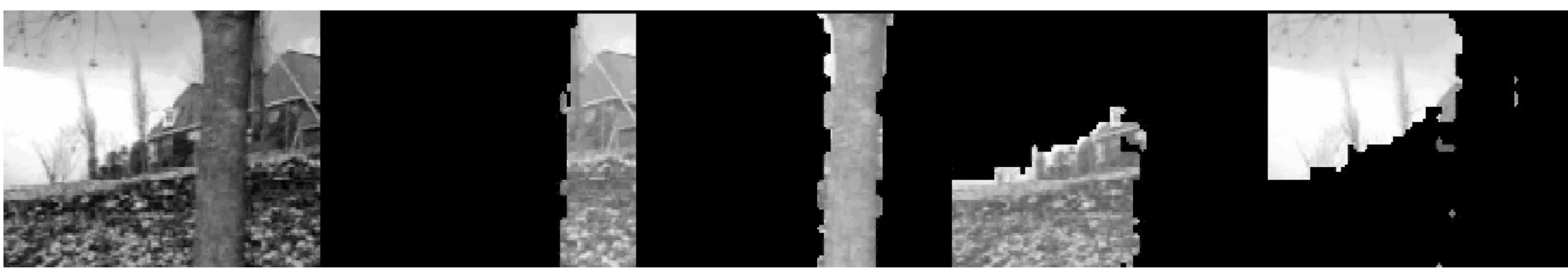
- Networks of spatial-temporal connections:



- Motion "proto-volume" in space-time







Comparison of Methods

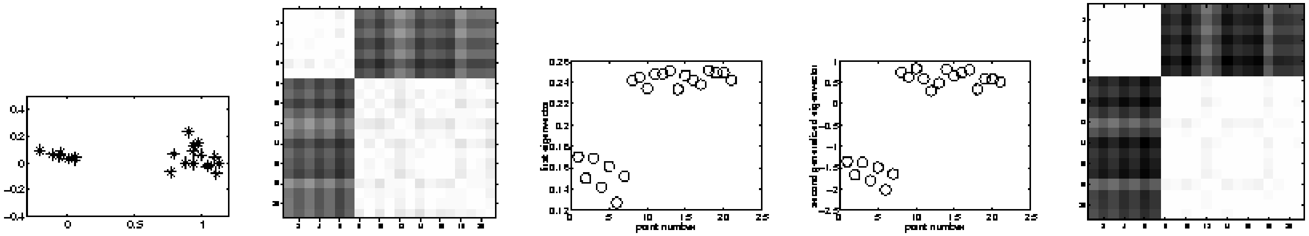
Authors	Matrix used	Procedure/Eigenvectors used
Perona/ Freeman	Affinity A	1st x: $Ax = \lambda x$ Recursive procedure
Shi/Malik	D-A with D a degree matrix $D(i,i) = \sum_j A(i,j)$	2nd smallest *generalized* eigenvector $(D - A)x = \lambda Dx$ Also recursive
Scott/ Longuet-Higgins	Affinity A, User inputs k	Finds k eigenvectors of A, forms V. Normalizes rows of V. Forms Q = VV'. Segments by Q. Q(i,j)=1 -> same cluster
Ng, Jordan, Weiss	Affinity A, User inputs k	Normalizes A. Finds k eigenvectors, forms X. Normalizes X, clusters rows

Advantages/Disadvantages

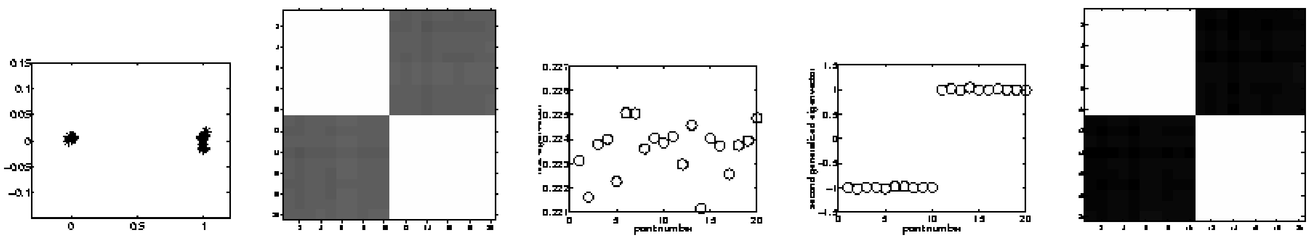
- Perona/Freeman
  - For block diagonal affinity matrices, the first eigenvector finds points in the "dominant"cluster; not very consistent
- Shi/Malik
  - 2^{nd} generalized eigenvector minimizes affinity between groups by affinity within each group; no guarantee, constraints

Advantages/Disadvantages

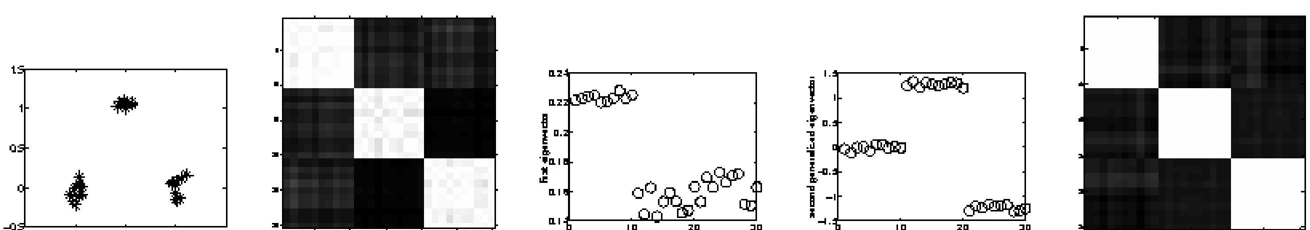
- Scott/Longuet-Higgins
  - Depends largely on choice of k
  - Good results
- Ng, Jordan, Weiss
  - Again depends on choice of k
  - Claim: effectively handles clusters whose overlap or connectedness varies across clusters



Affinity Matrix | Perona/Freeman 1st eigenv. | Shi/Malik 2nd gen. eigenv. | Scott/Lon.Higg Q matrix



Affinity Matrix | Perona/Freeman 1st eigenv. | Shi/Malik 2nd gen. eigenv. | Scott/Lon.Higg Q matrix



Affinity Matrix | Perona/Freeman 1st eigenv. | Shi/Malik 2nd gen. eigenv. | Scott/Lon.Higg Q matrix

Segmentation and Line Fitting

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

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

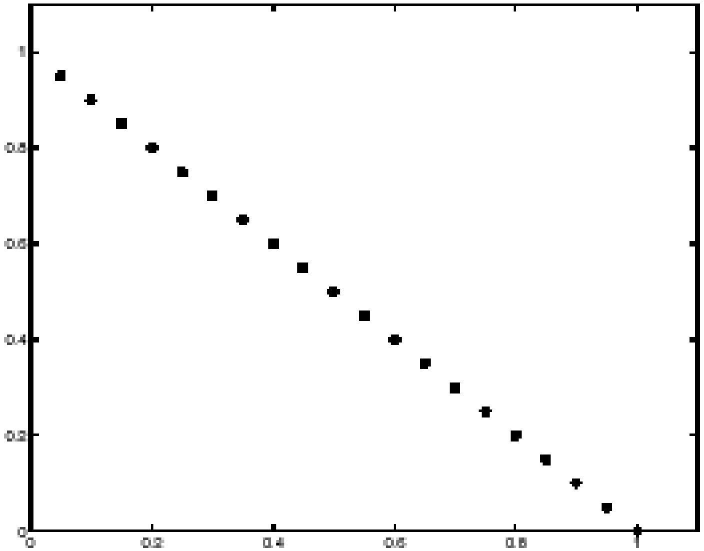
- Purports to answer all three questions
  - in practice, answer isn't usually all that much help
- We do for lines only
- A line is the set of points (x, y) such that

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

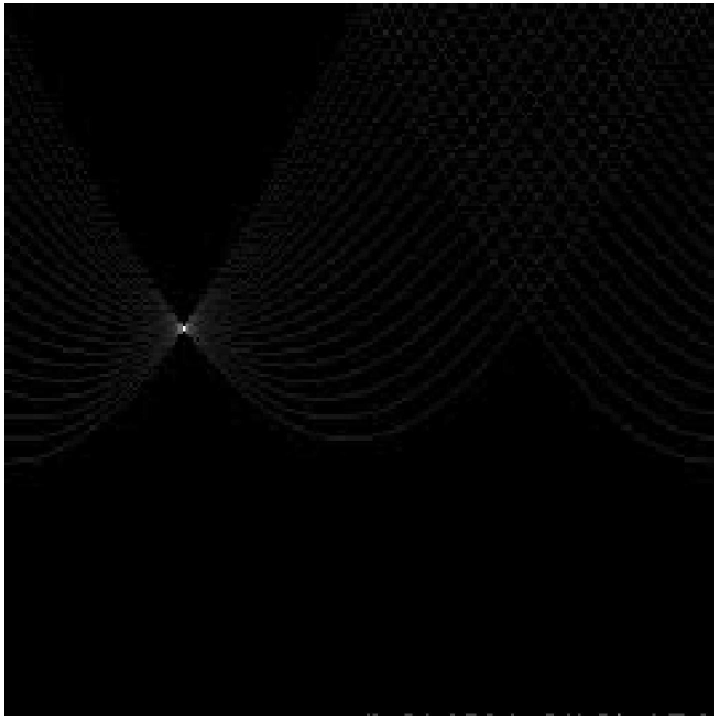
- Different choices of θ, d>0 give different lines
- For any (x, y) there is a one parameter family of lines through this point, given by

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

- 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



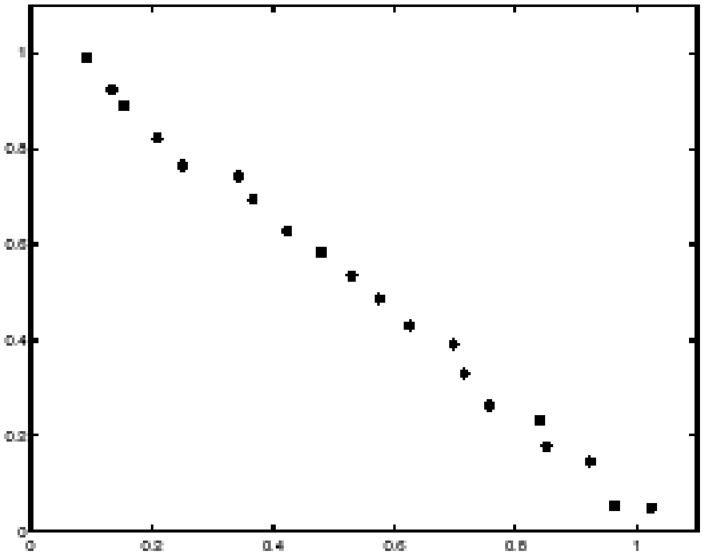
votes

Mechanics of the Hough transform

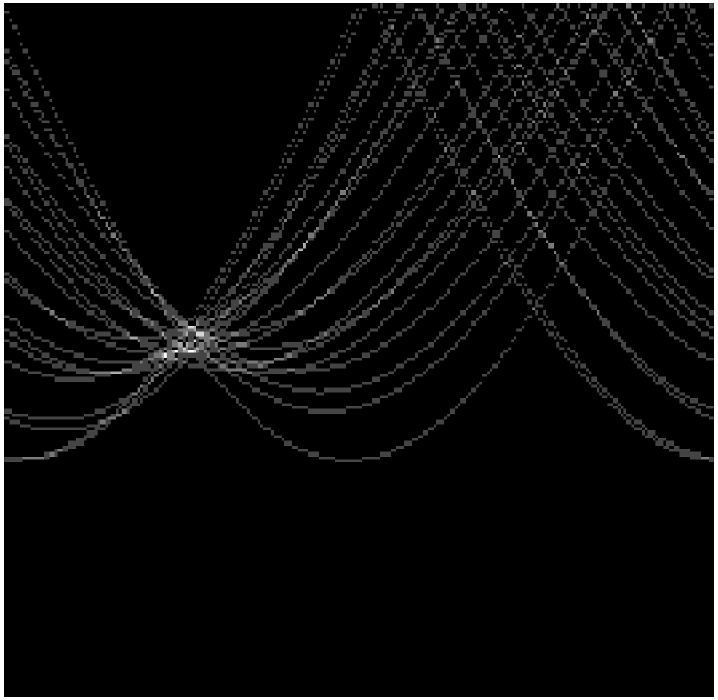
- Construct an array representing θ, d
- For each point, render the curve (θ, d) into this array, adding one at each cell
- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)

- How many lines?
  - count the peaks in the Hough array
- Who belongs to which line?
  - tag the votes

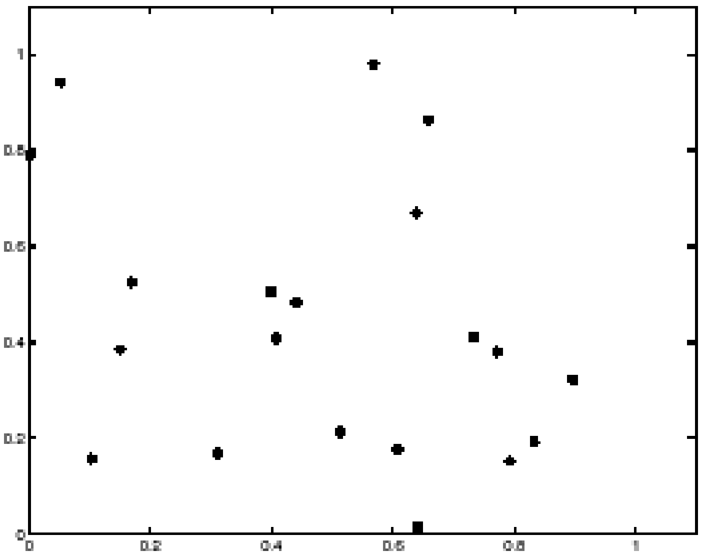
- Hardly ever satisfactory in practice, because problems with noise and cell size defeat it



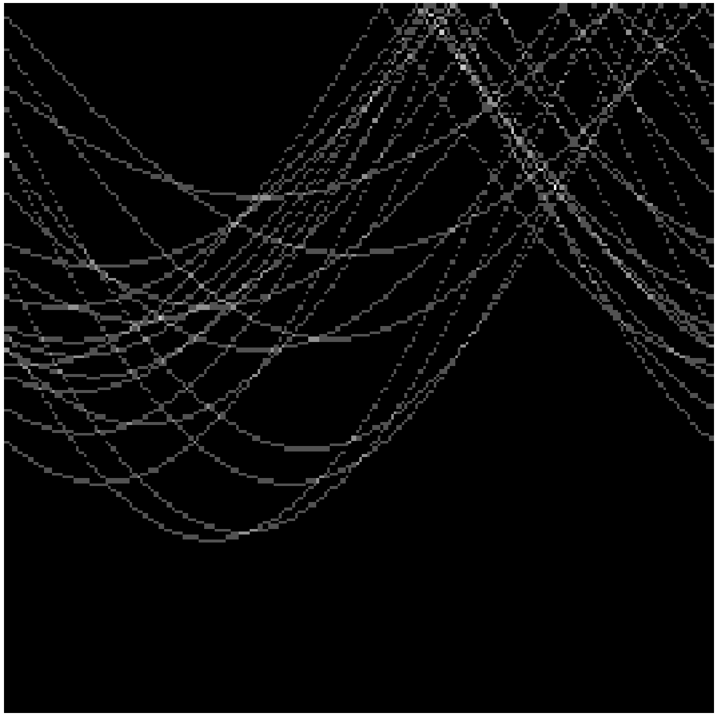
tokens

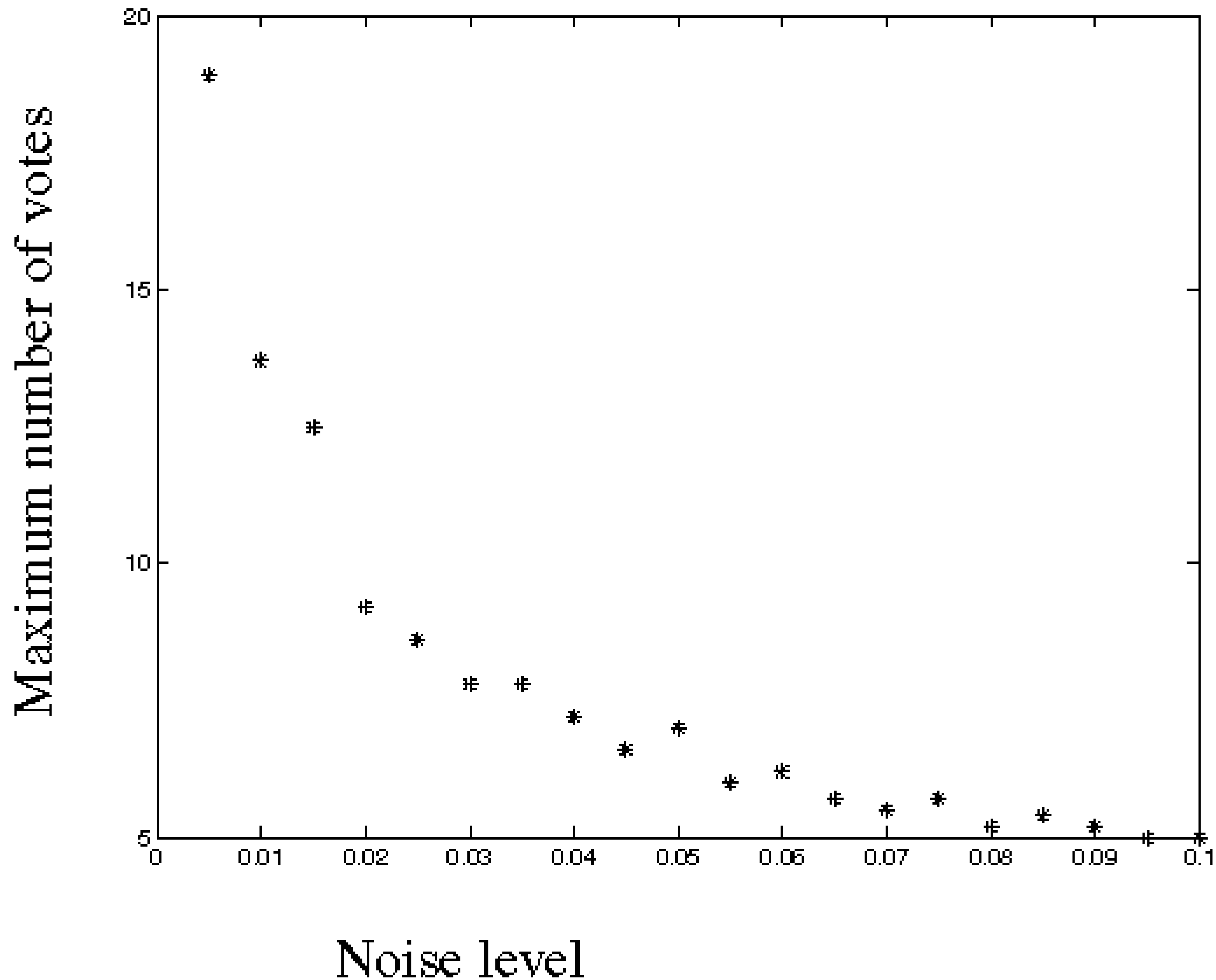


votes

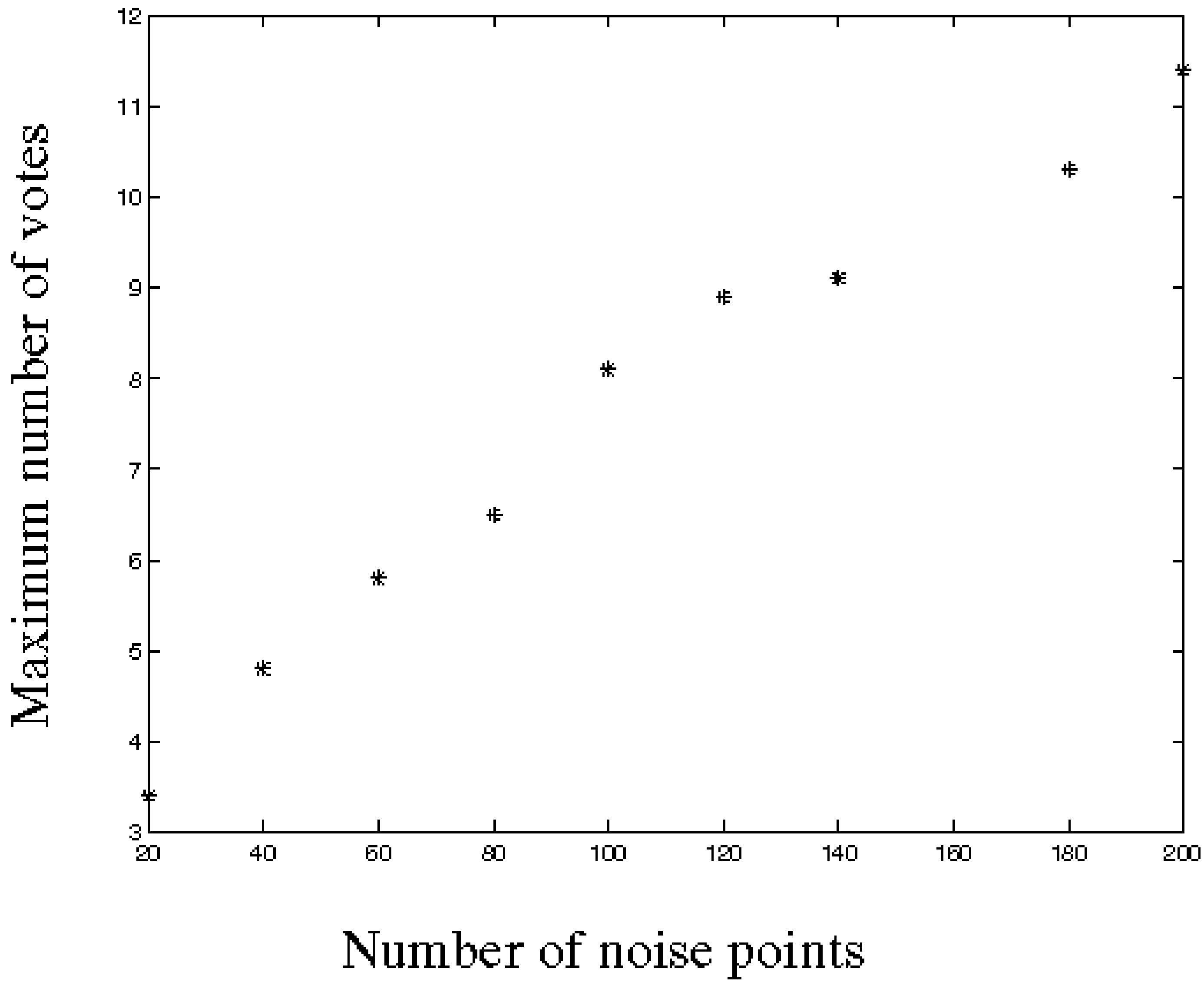


1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1



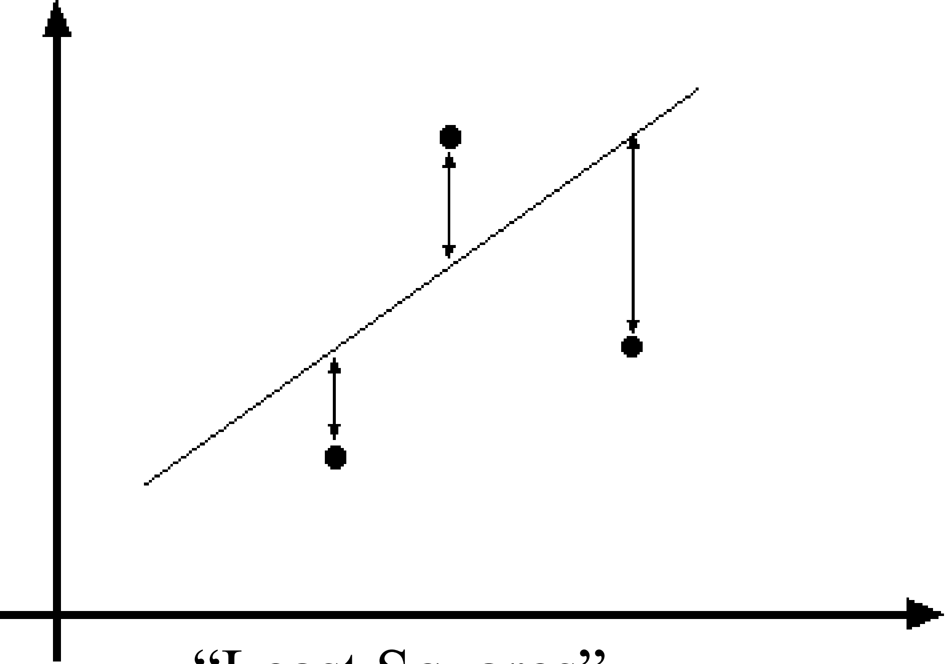


Maximum number of votes
20
15
10
5
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Noise level


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

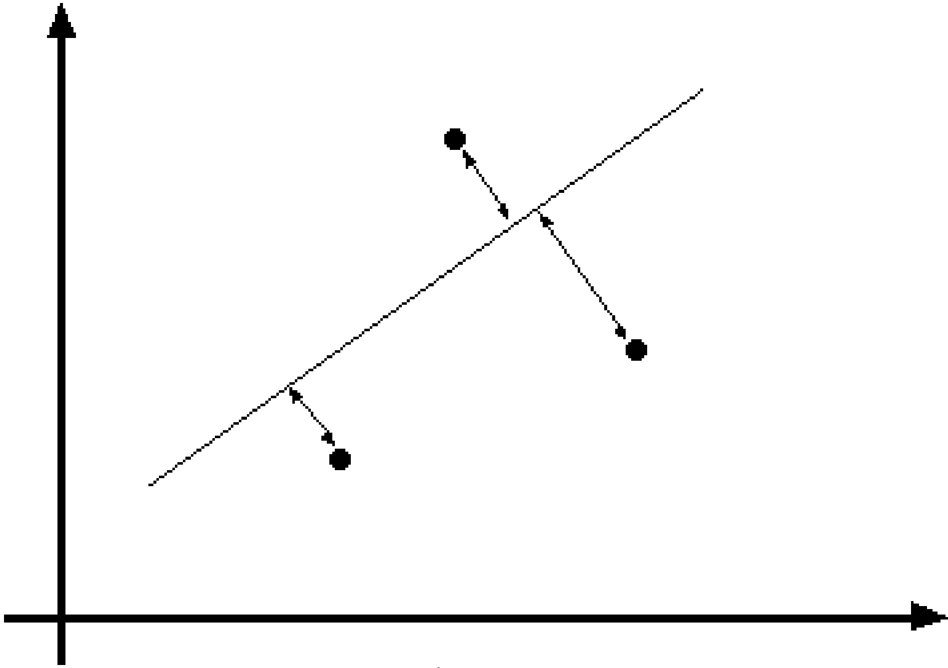
Line fitting

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”

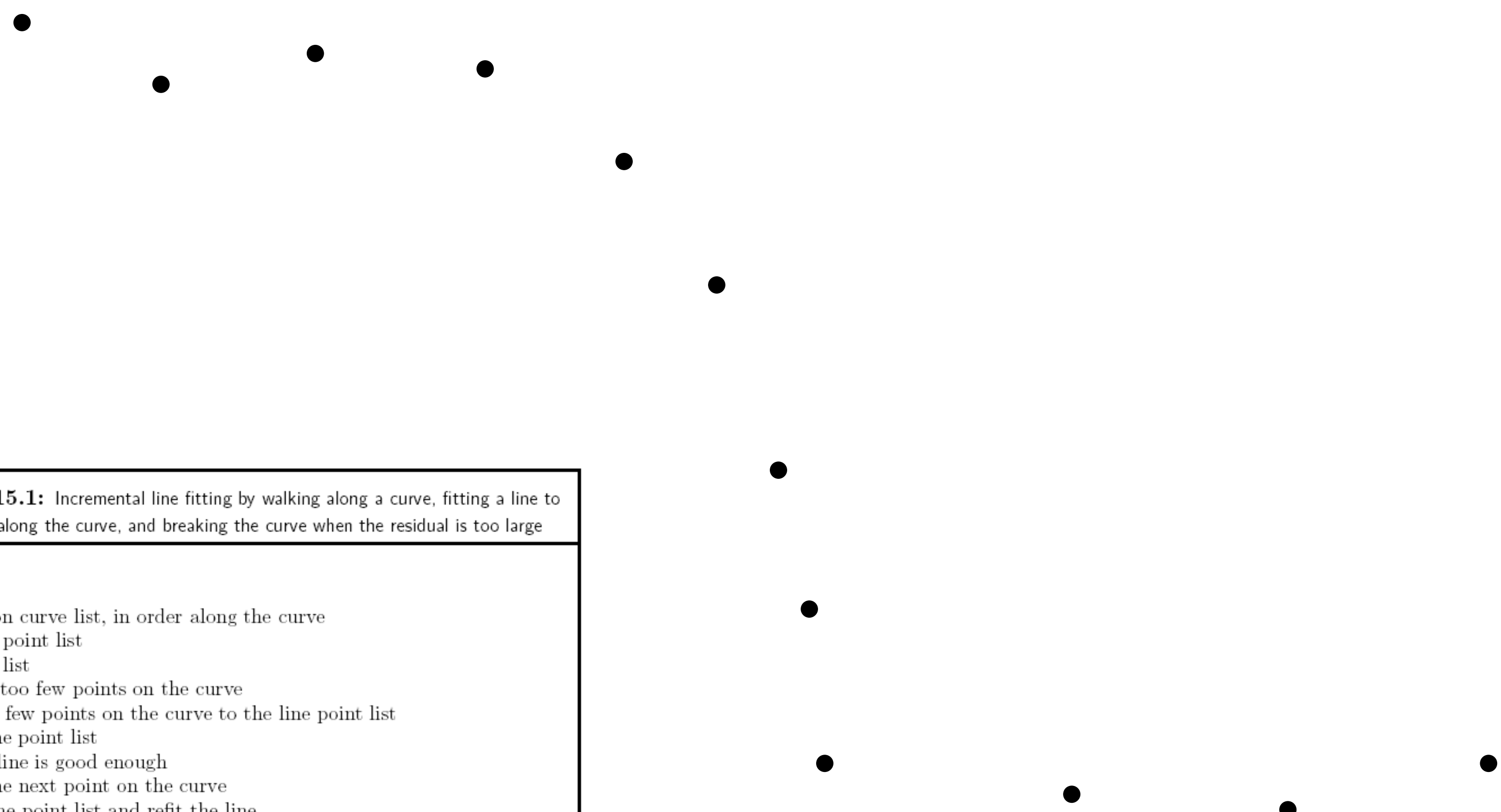
Who came from which line?

- 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
```

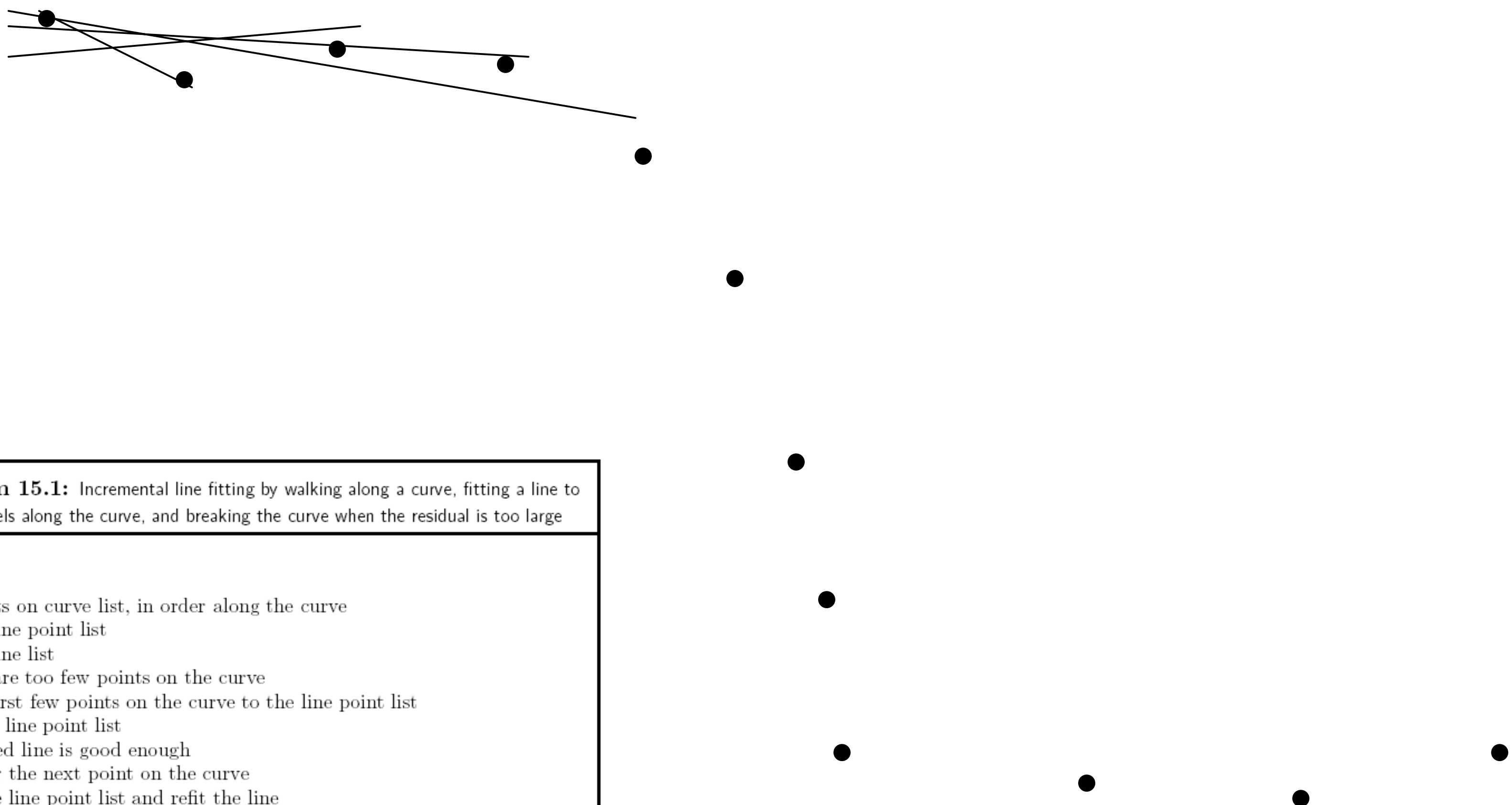
Incremental line fitting



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
```

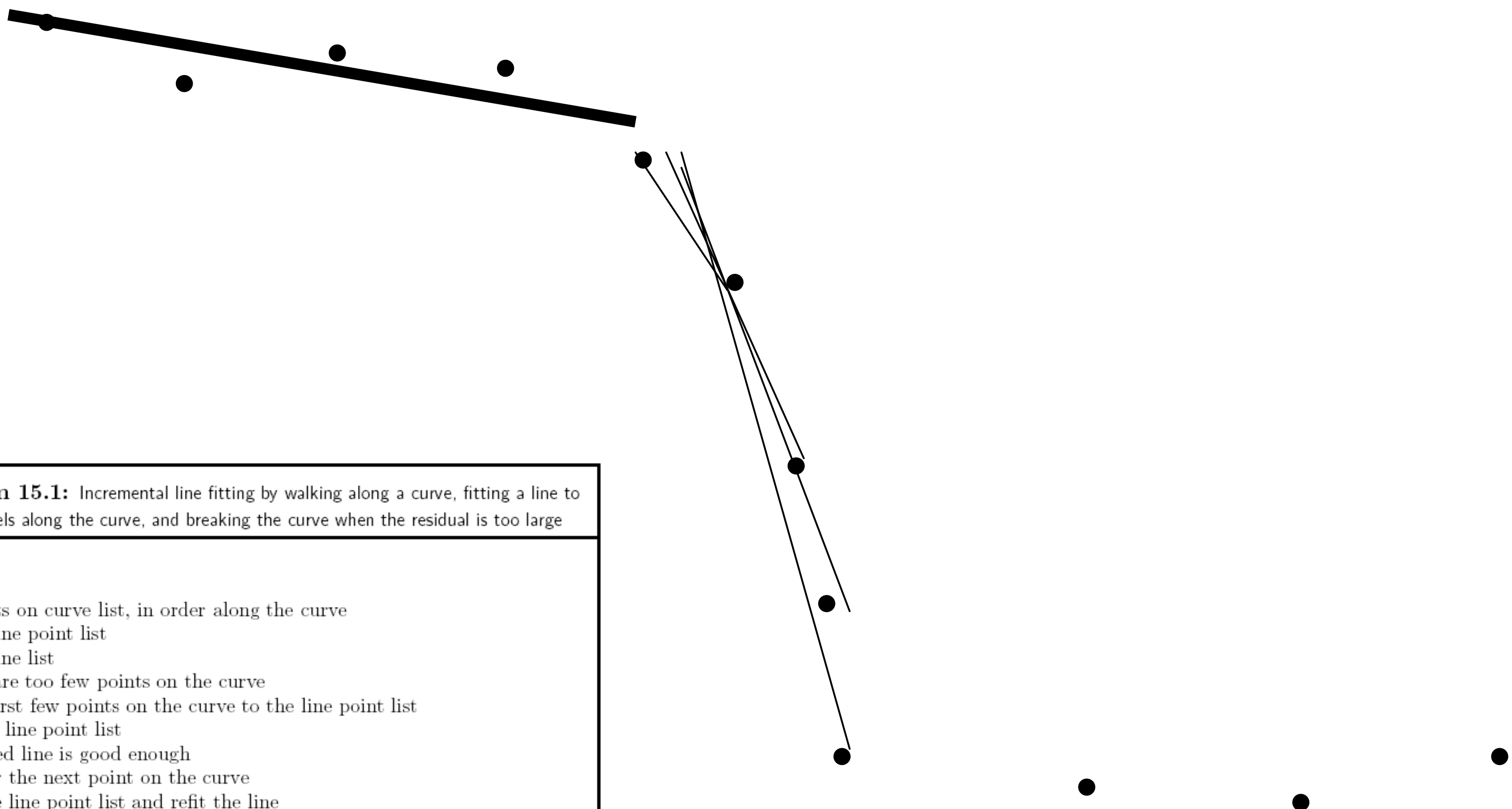
Incremental line fitting



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
```

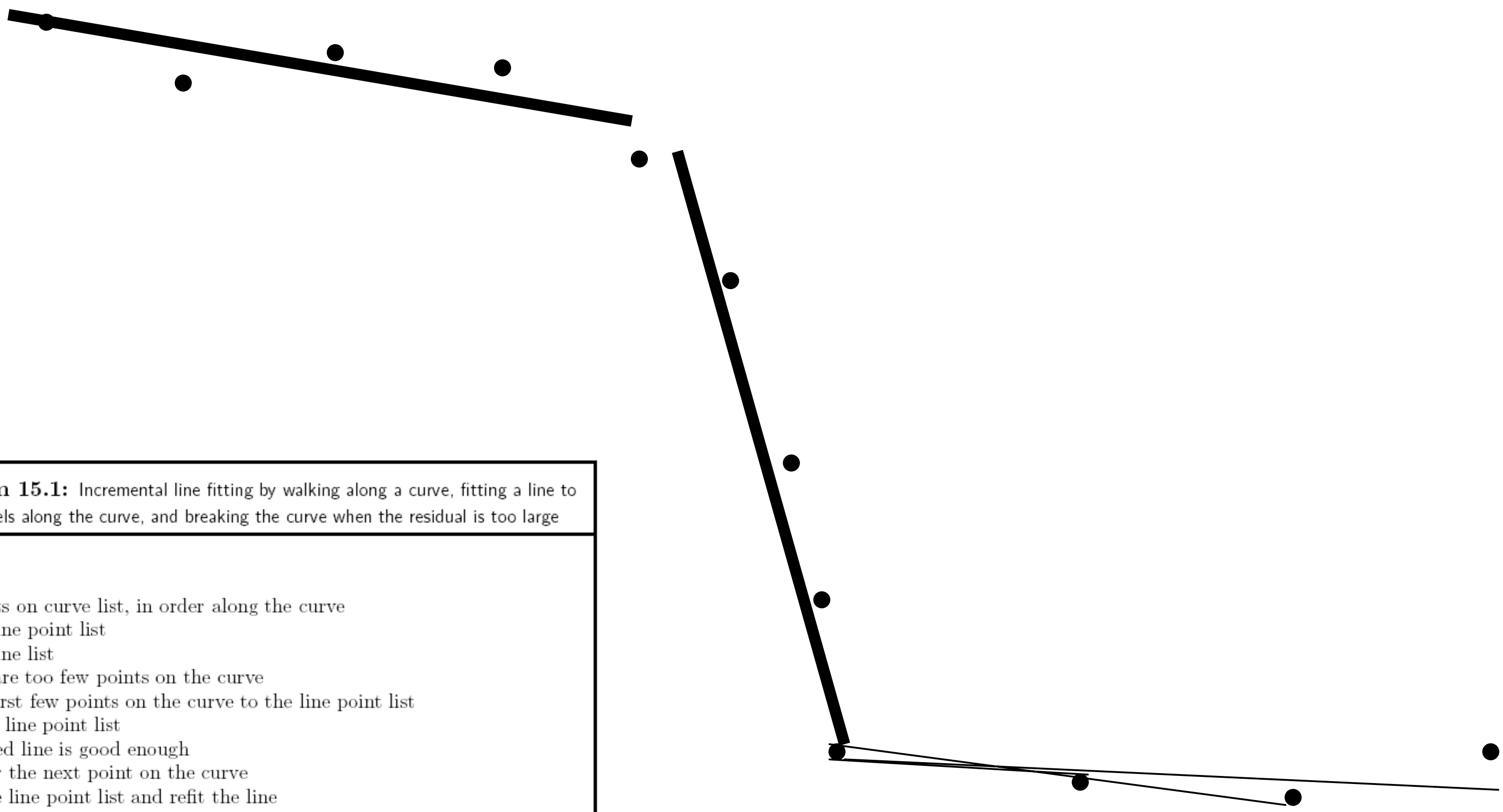
Incremental line fitting



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
```

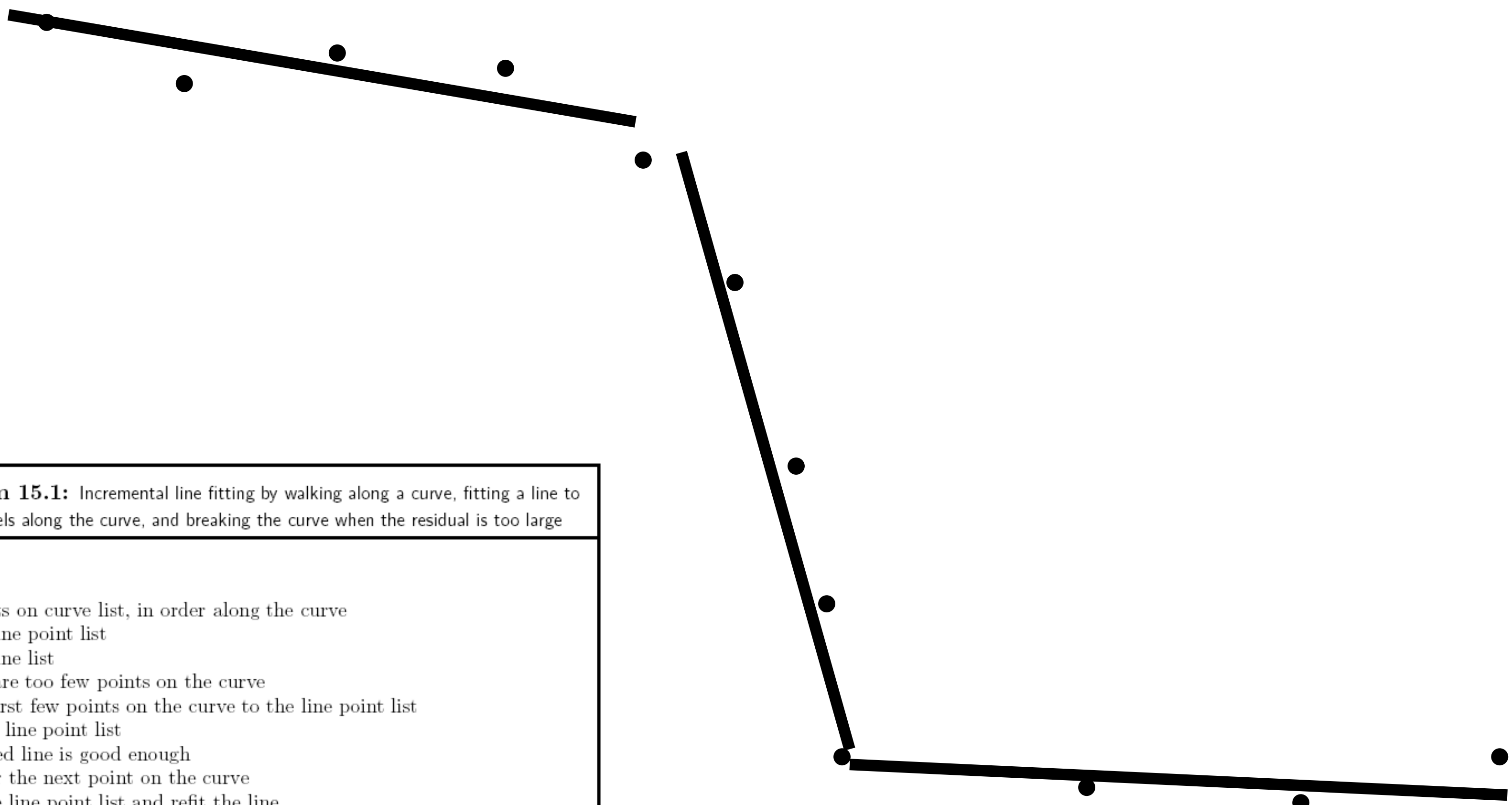
Incremental line fitting



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



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

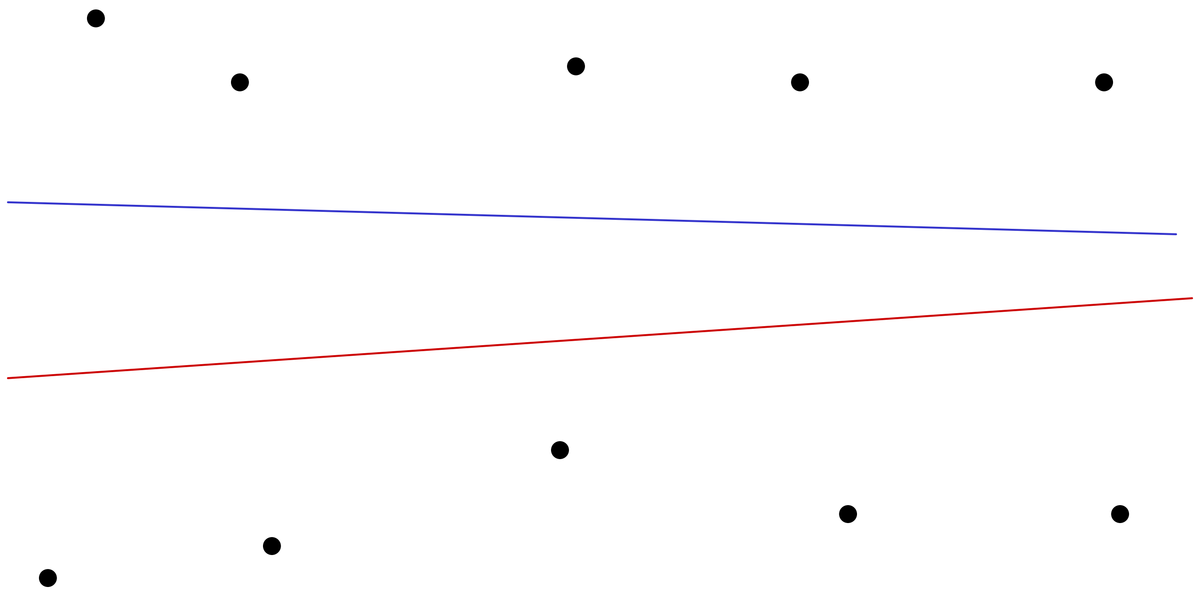
```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next 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
```

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

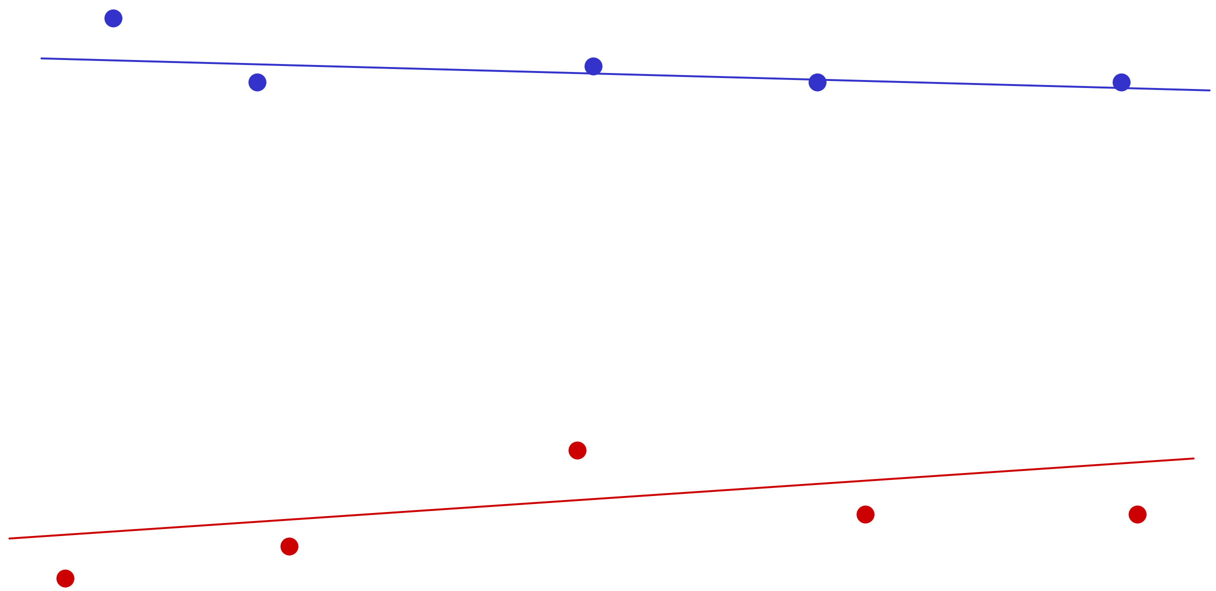


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

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

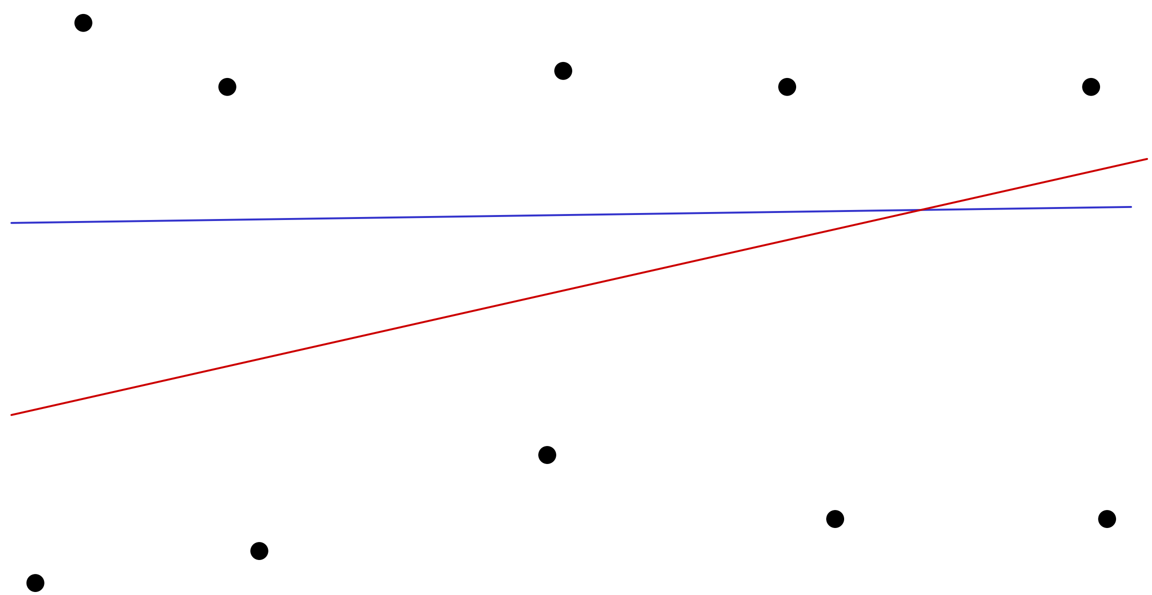


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

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

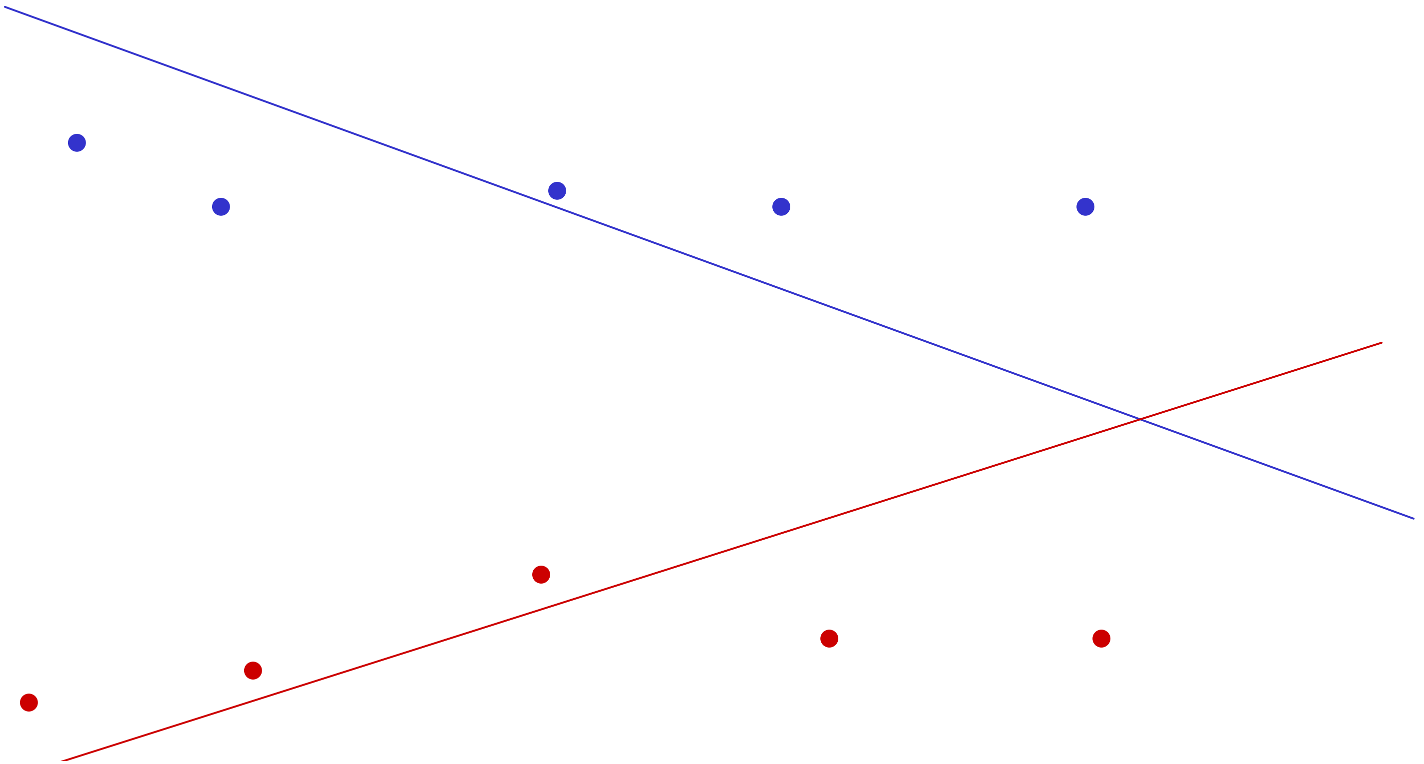


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

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

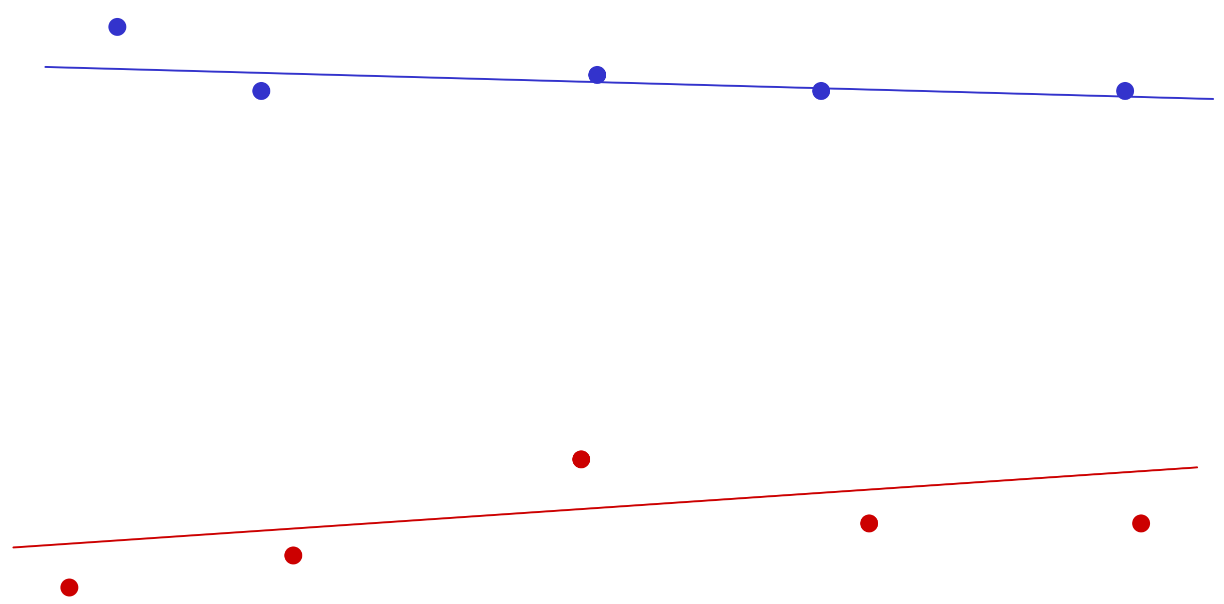


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

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

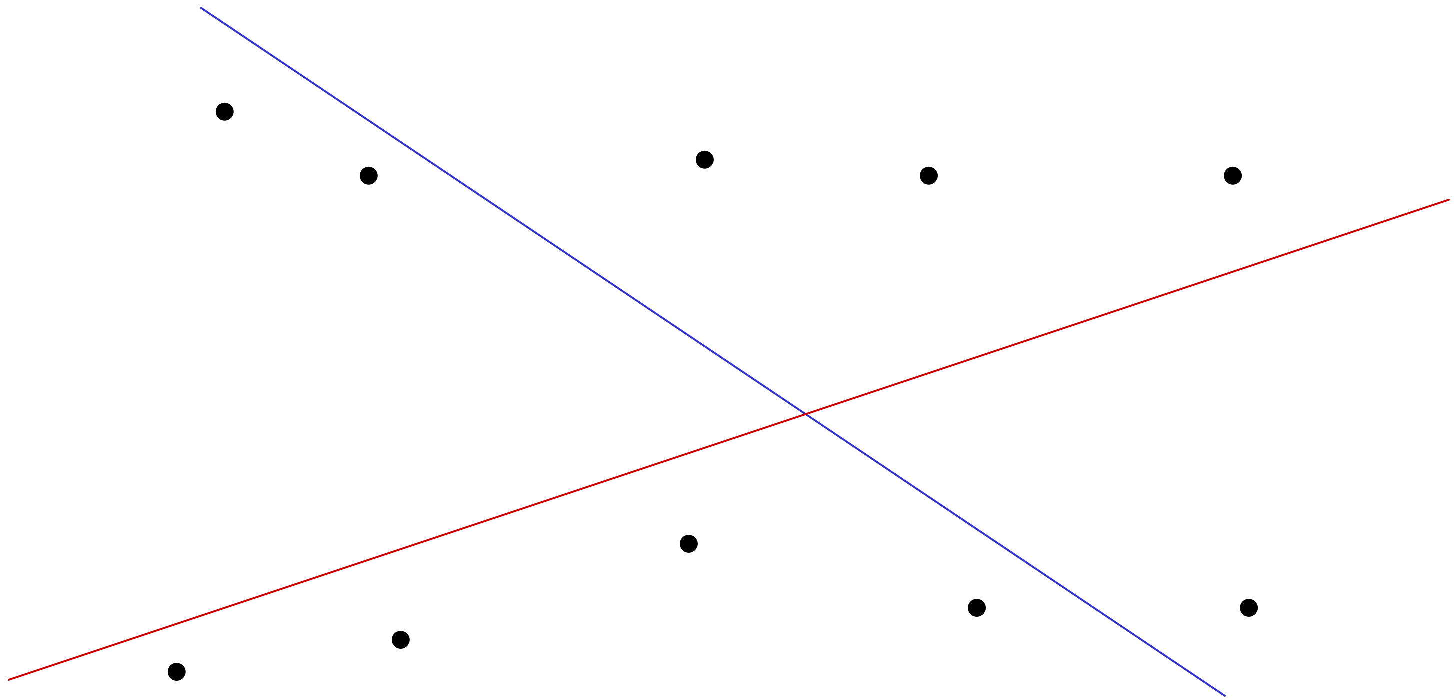


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

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

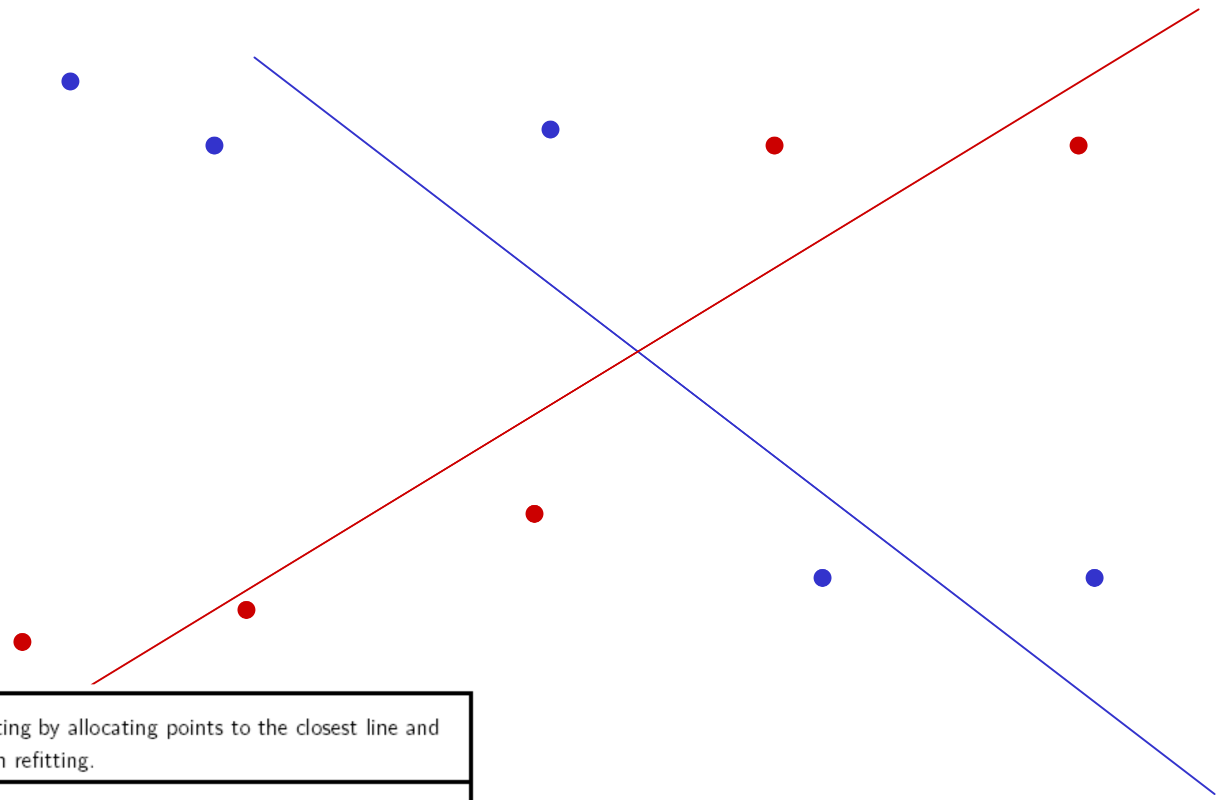


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

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



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

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

Lecture 14:

- Unsupervised Category Learning
- Gestalt Principles
- Segmentation by Clustering
  - K-Means
  - Graph cuts
- Segmentation by Fitting
  - Hough transform
  - Fitting

Readings: F&P Ch. 14, 15.1-15.2

(Next time: Finish fitting, Probabilistic segmentation;
FP 15.4-5, 16)

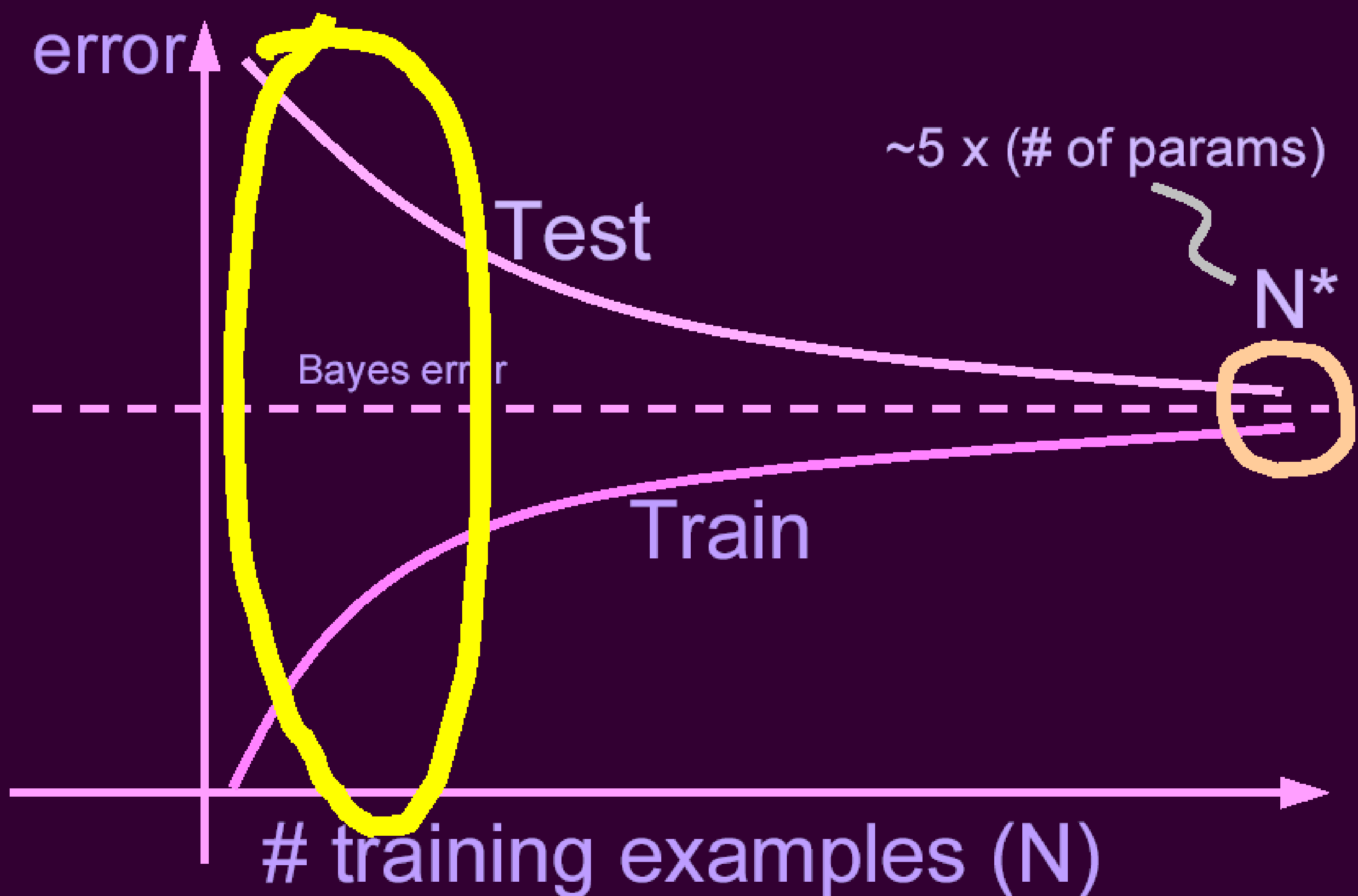
Visual learning is inefficient

Algorithm	Training Examples	Categories
Rowley et al.	~500	Faces
Schneiderman, et al.	~2,000	Faces, Cars
Viola et al.	~10,000	Faces
Burl, et al. Weber, et al. Fergus, et al.	200 ~ 400	Faces, Motorbikes, Spotted cats, Airplanes, Cars

[Slide from Bradsky & Thrun, Stanford]

Slide from Li Fei-Fei http://www.vision.caltech.edu/feifeili/Resume.htm

Words of wisdom from statisticians



[Slide from Bradsky & Thrun, Stanford]


This guy is wearing a haircut called a “Mullet”
[Slide from Bradsky & Thrun, Stanford]

Find the Mullets

One-Shot Learning

[Slide from Bradsky & Thrun, Stanford]

One-Shot Learning

"The appearance of the categories we know and … the variability in their appearance, gives us important information on what to expect in a new category"

1. *L. Fei-Fei, R. Fergus and P. Perona, "A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories" ICCV 03.*
2. *R. Fergus, P. Perona and A.Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning", CVPR 03.*

- *http://www.vision.caltech.edu/html-files/publications.html*

[Slide from Bradsky & Thrun, Stanford]

Learn meta-parameters

$$\frac{p(\text{Object})}{p(\text{No Object})}$$ set to 1.0

$$R = \frac{p(\text{Object}|\mathcal{X}, \mathcal{A}, \mathcal{X}_t, \mathcal{A}_t)}{p(\text{No Object}|\mathcal{X}, \mathcal{A}, \mathcal{X}_t, \mathcal{A}_t)} \quad (1)$$

$$= \frac{p(\mathcal{X}, \mathcal{A}|\mathcal{X}_t, \mathcal{A}_t, \text{Object})\ p(\text{Object})}{p(\mathcal{X}, \mathcal{A}|\mathcal{X}_t, \mathcal{A}_t, \text{No object})\ p(\text{No object})} \quad (2)$$

$$\approx \frac{\int p(\mathcal{X}, \mathcal{A}|\boldsymbol{\theta}, \text{Object}) p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t, \text{Object})\ d\boldsymbol{\theta}}{\int p(\mathcal{X}, \mathcal{A}|\boldsymbol{\theta}_{bg}, \text{No Object}) p(\boldsymbol{\theta}_{bg}|\mathcal{X}_t, \mathcal{A}_t, \text{No Object})\ d\boldsymbol{\theta}_{bg}} \quad (3)$$

Shape

Training set Appearance

Appearance

Training set Shape

Model Params $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}^{\mathcal{X}}, \boldsymbol{\mu}^{\mathcal{A}}, \boldsymbol{\Gamma}^{\mathcal{X}}, \boldsymbol{\Gamma}^{\mathcal{A}}\}$

Learn $p(\boldsymbol{\theta}|\mathcal{X}_t, \mathcal{A}_t)$ [Fei Fei et al. 2003]

[Slide from Bradsky & Thrun, Stanford]