MIT CSAIL

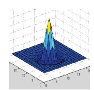
6.869: Advances in Computer Vision

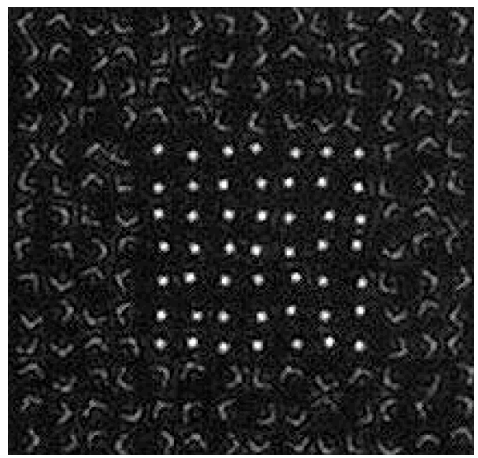
Antonio Torralba, 2013

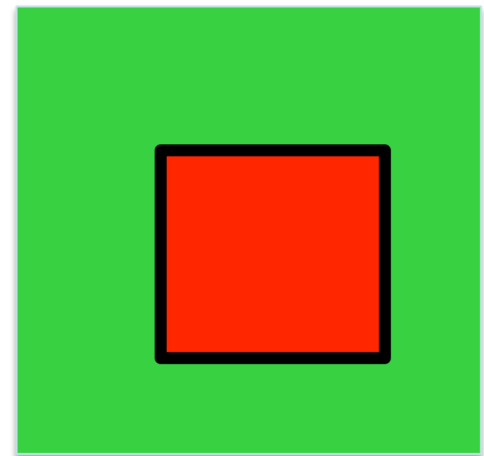
MIT COMPUTER VISION

Lecture 9

Edges and segmentation







Lecture 7

Texture representation

Lecture 8

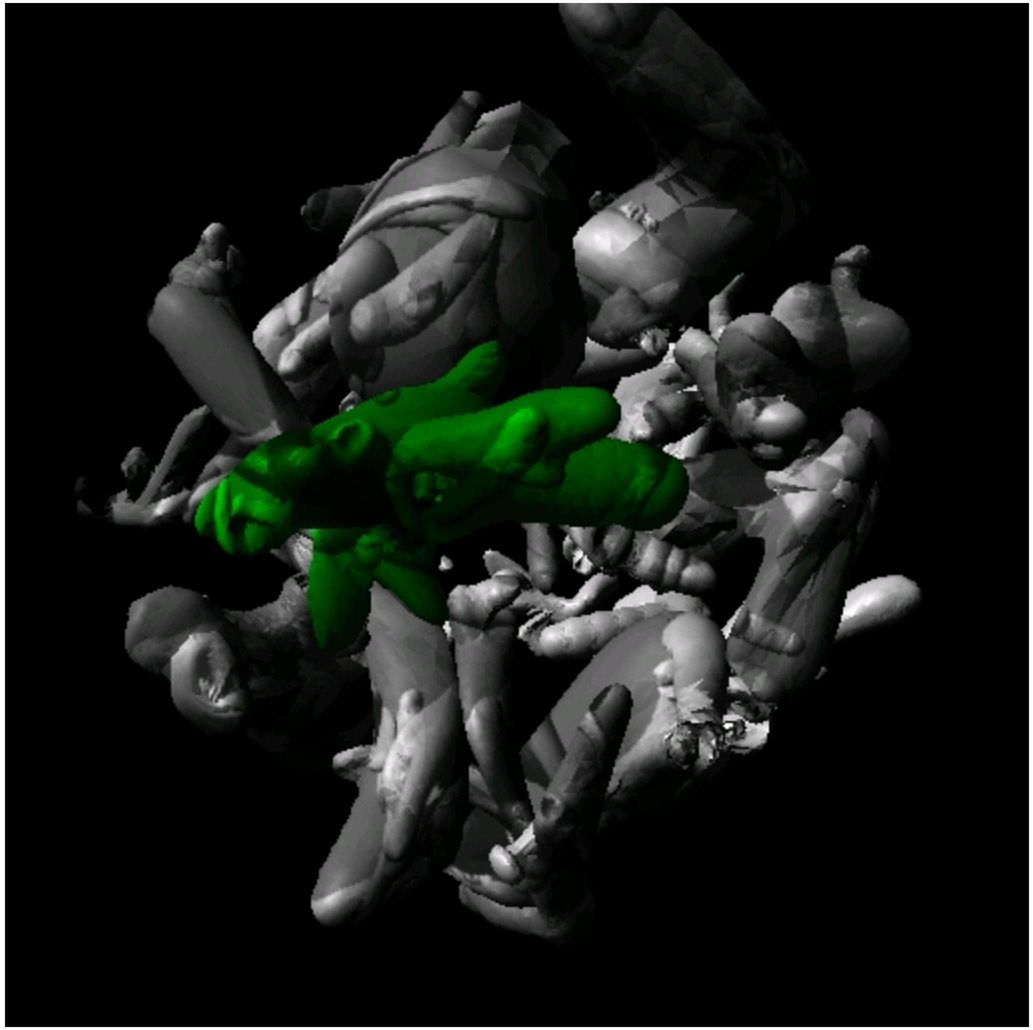
Edges and segmentation

A “simple” segmentation problem

It can get a lot harder

Brady, M. J., & Kersten, D. (2003). Bootstrapped learning of novel objects. J Vis, 3(6), 413-422

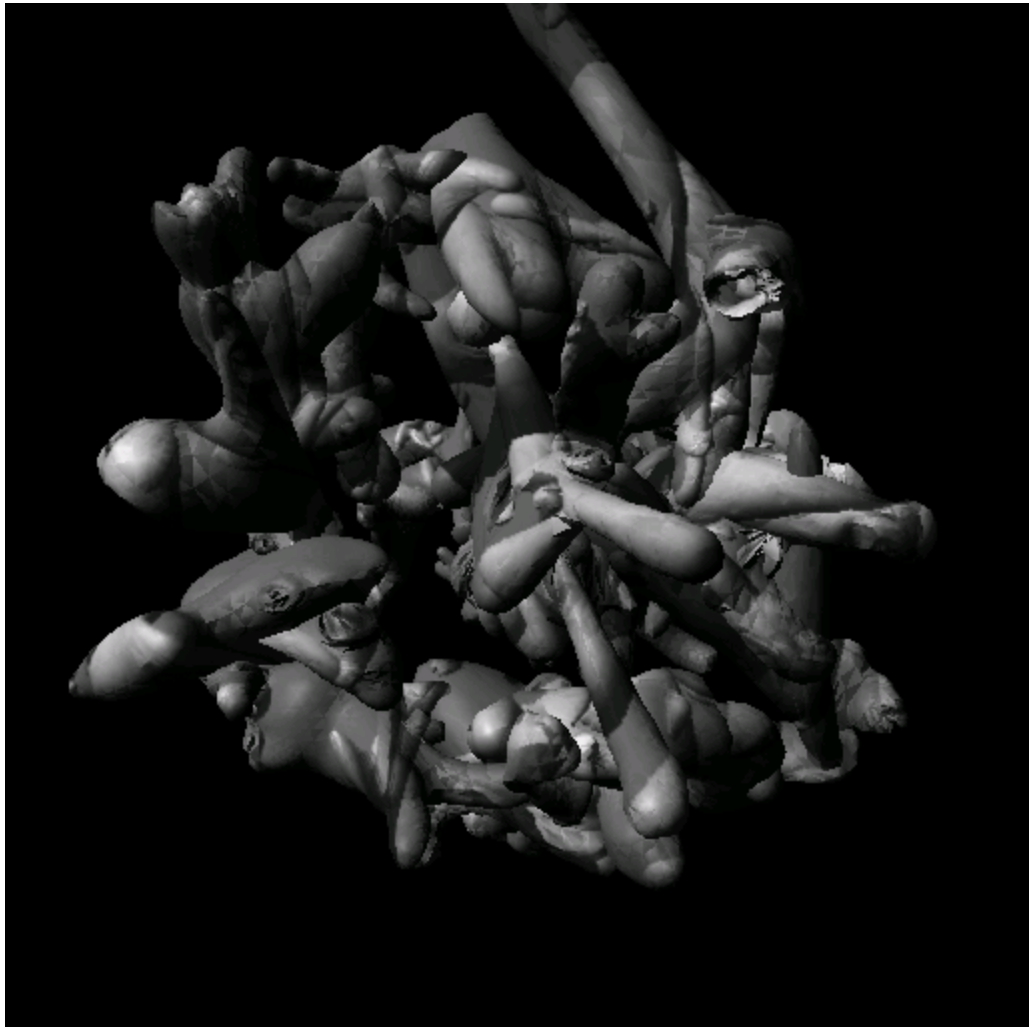
Discover the camouflaged object

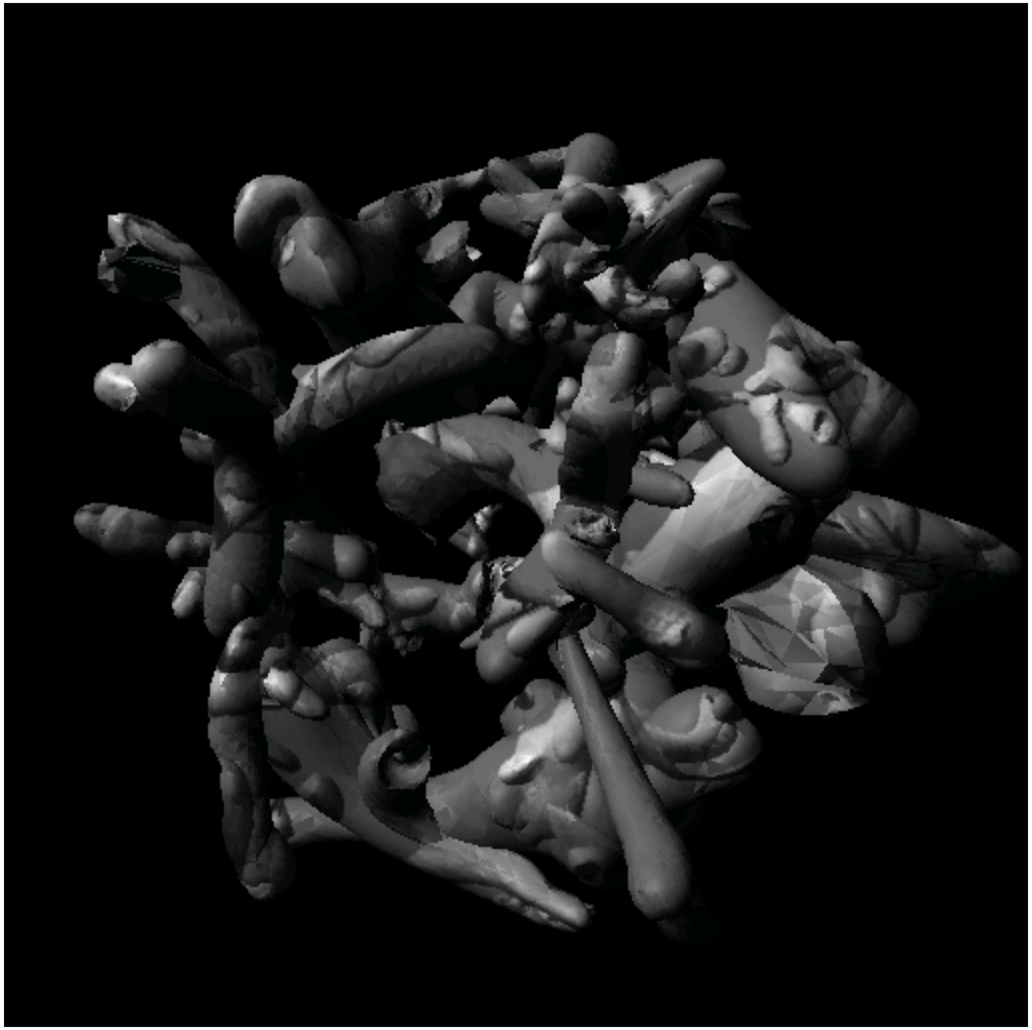


Brady, M. J., & Kersten, D. (2003). Bootstrapped learning of novel objects. J Vis, 3(6), 413-422

Discover the camouflaged object

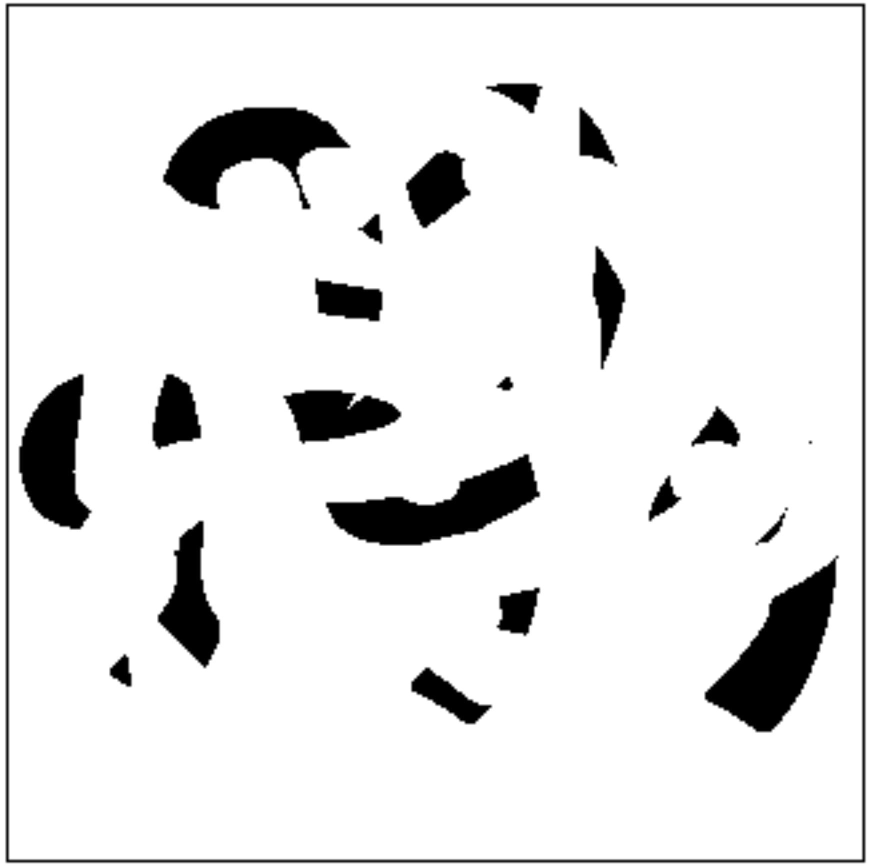
Brady, M. J., & Kersten, D. (2003). Bootstrapped learning of novel objects. J Vis, 3(6), 413-422





Any guesses?

Segmentation is a global process



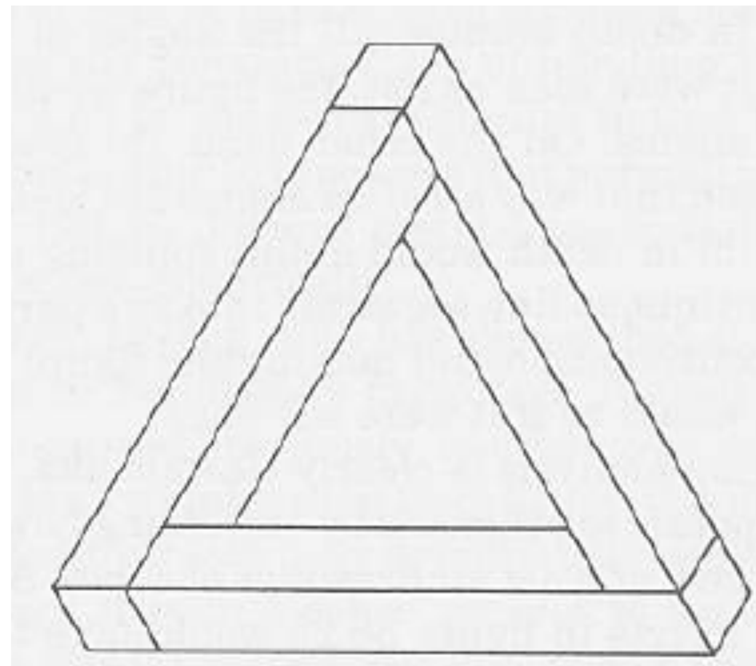
What are the occluded numbers?

Segmentation is a global process

What are the occluded numbers?

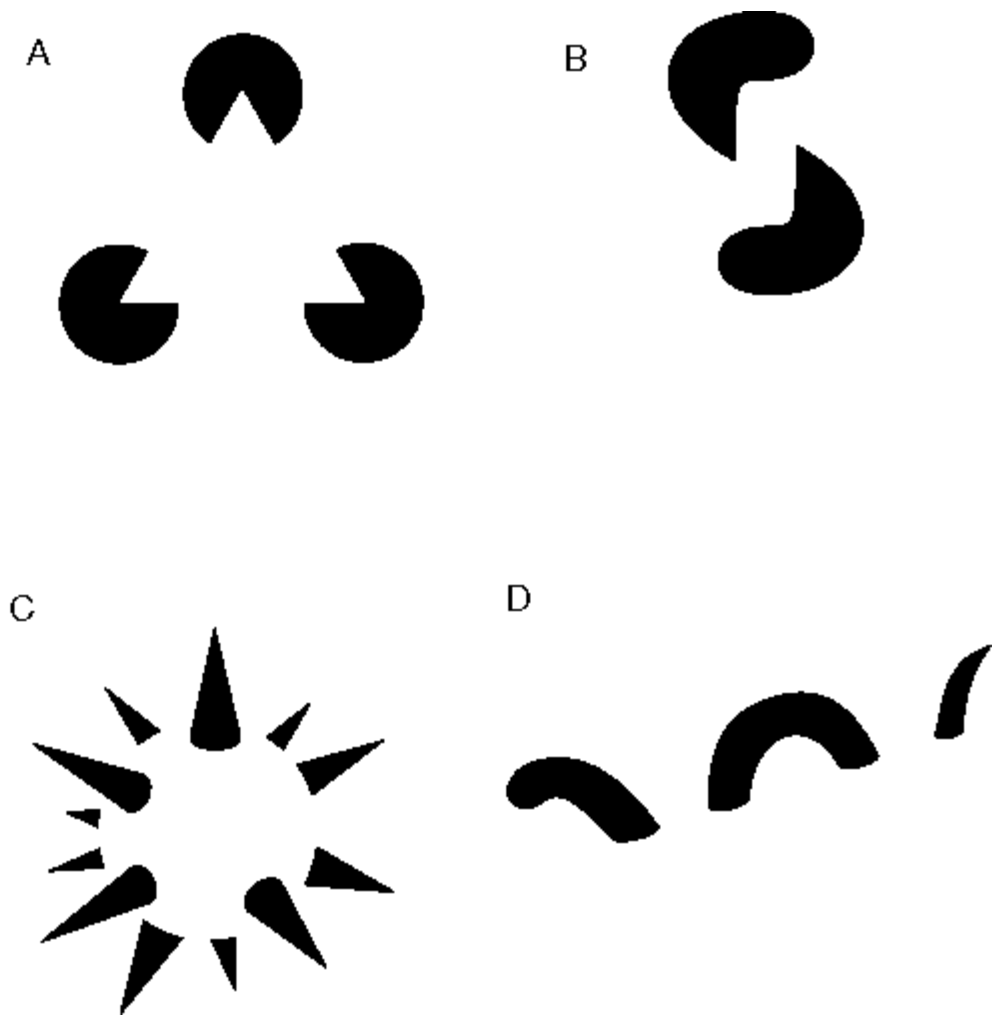
Occlusion is an important cue in grouping.

… but not too global



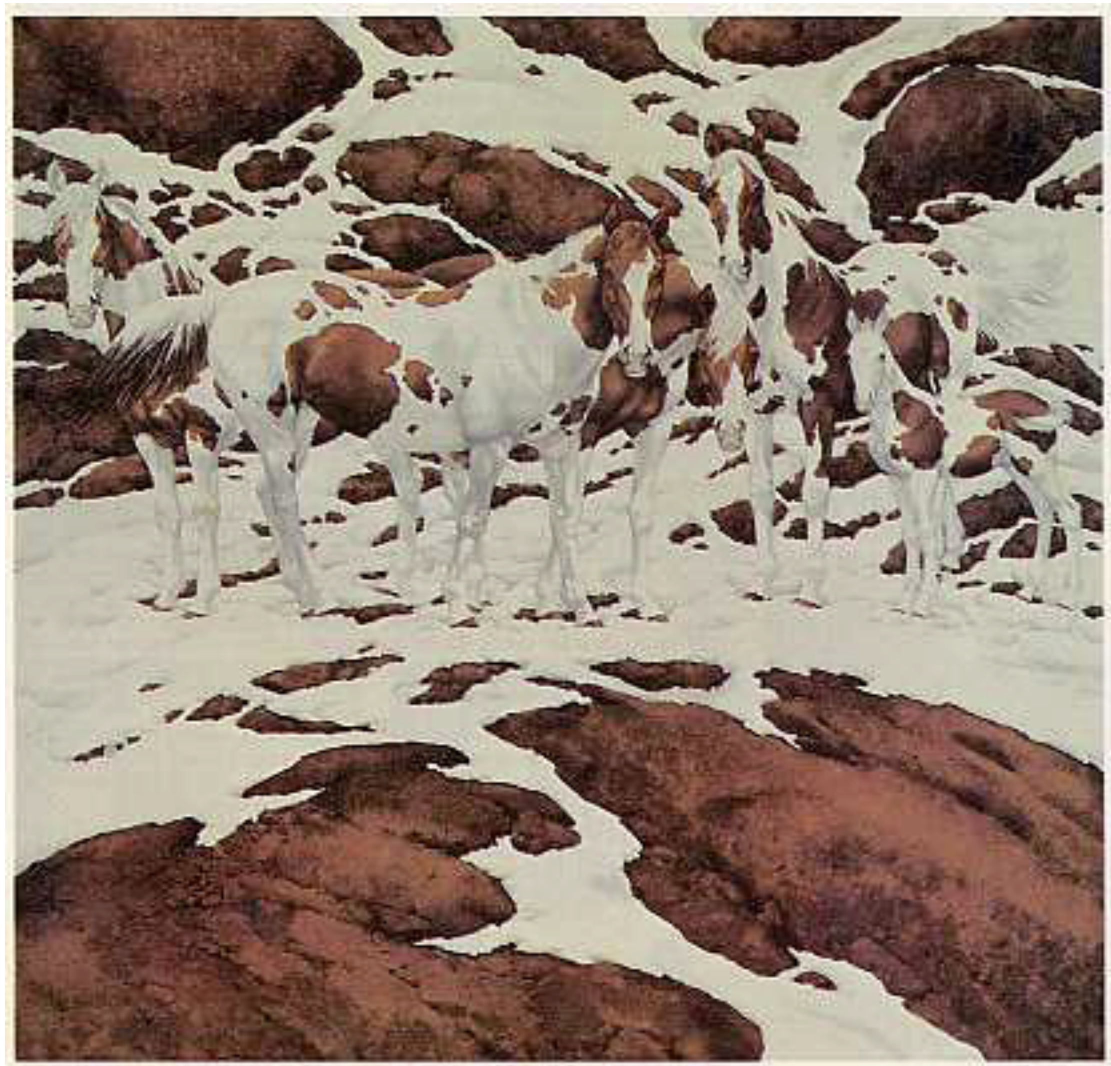
Magritte, 1957

Groupings by Invisible Completions



* Images from Steve Lehar’s Gestalt papers

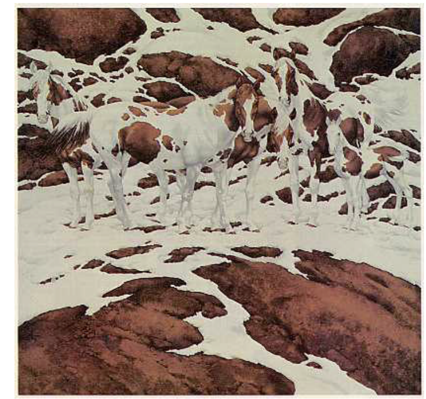
1970s: R. C. James



2000s: Bev Doolittle

Perceptual organization

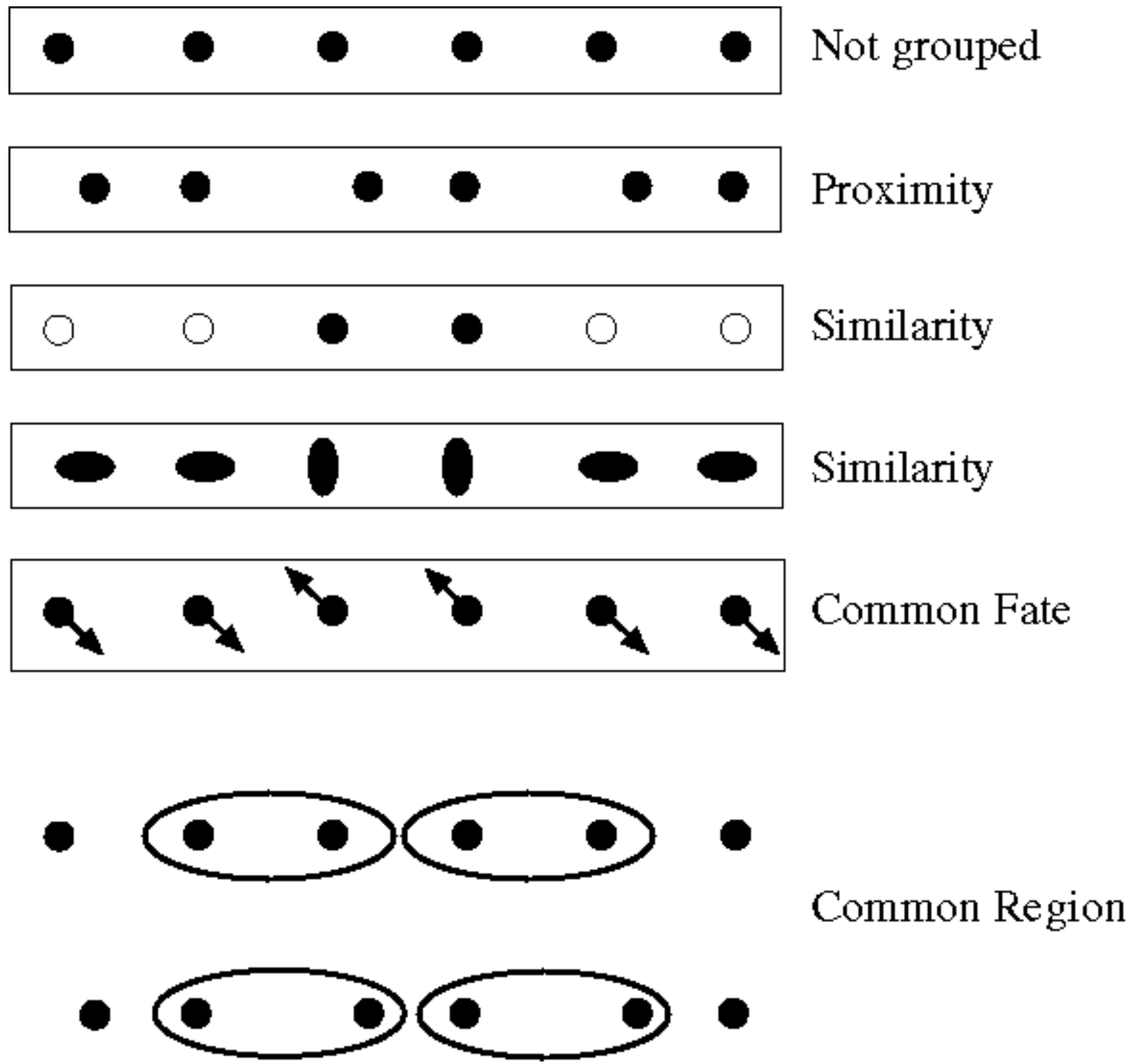
"…the processes by which the bits and pieces of visual information that are available in the retinal image are structured into the larger units of perceived objects and their interrelations"

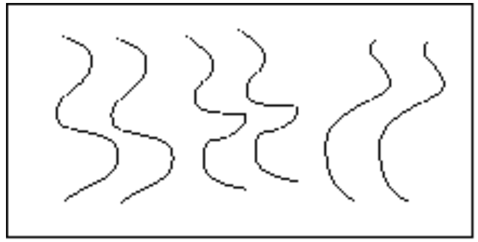


Stephen E. Palmer, *Vision Science, 1999*

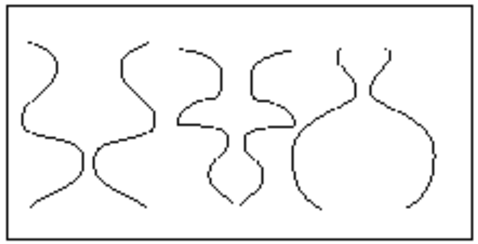
Gestalt principles

There are hundreds of different grouping laws

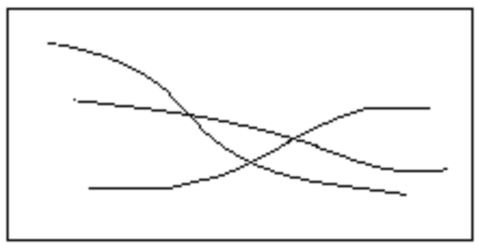

Not grouped
Proximity
Similarity
Similarity
Common Fate
Common Region



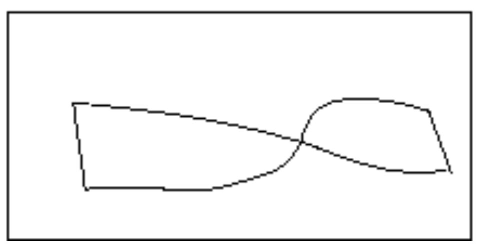
Parallelism



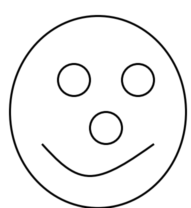
Symmetry



Continuity



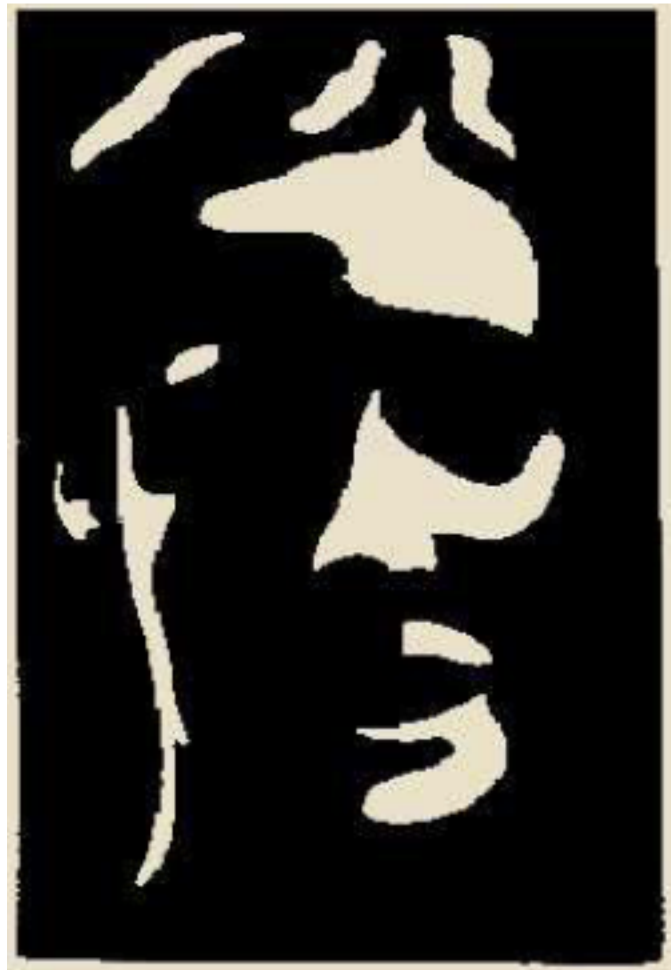
Closure



Familiar configuration

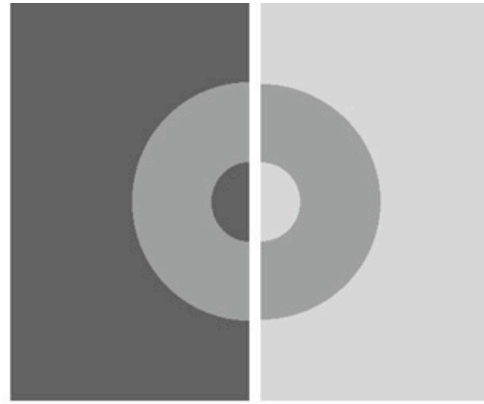
Familiarity

Familiarity



Influences of grouping

a

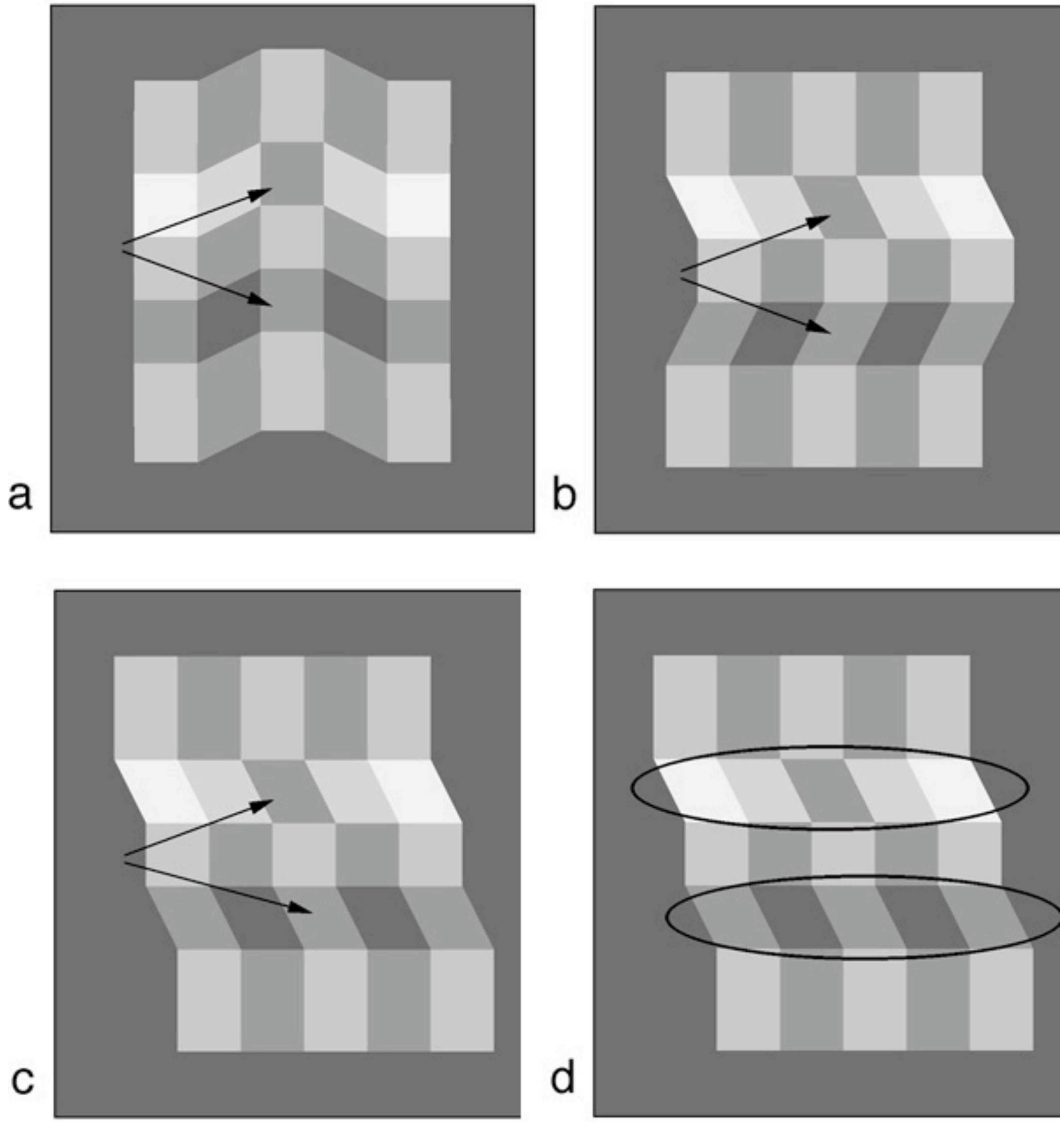


b

c

Grouping influences other perceptual mechanisms such as lightness perception

http://web.mit.edu/persci/people/adelson/publications/gazzan.dir/koffka.html



Variations on the corrugated plaid. (a) The two patches appear nearly the same. (b) The patches appear quite different. (c) The patches appear quite different, but there is no plausible shaded model. (d) Possible grouping induced by junctions.

E. H. Adelson, Lightness Perception and Lightness Illusions

Today

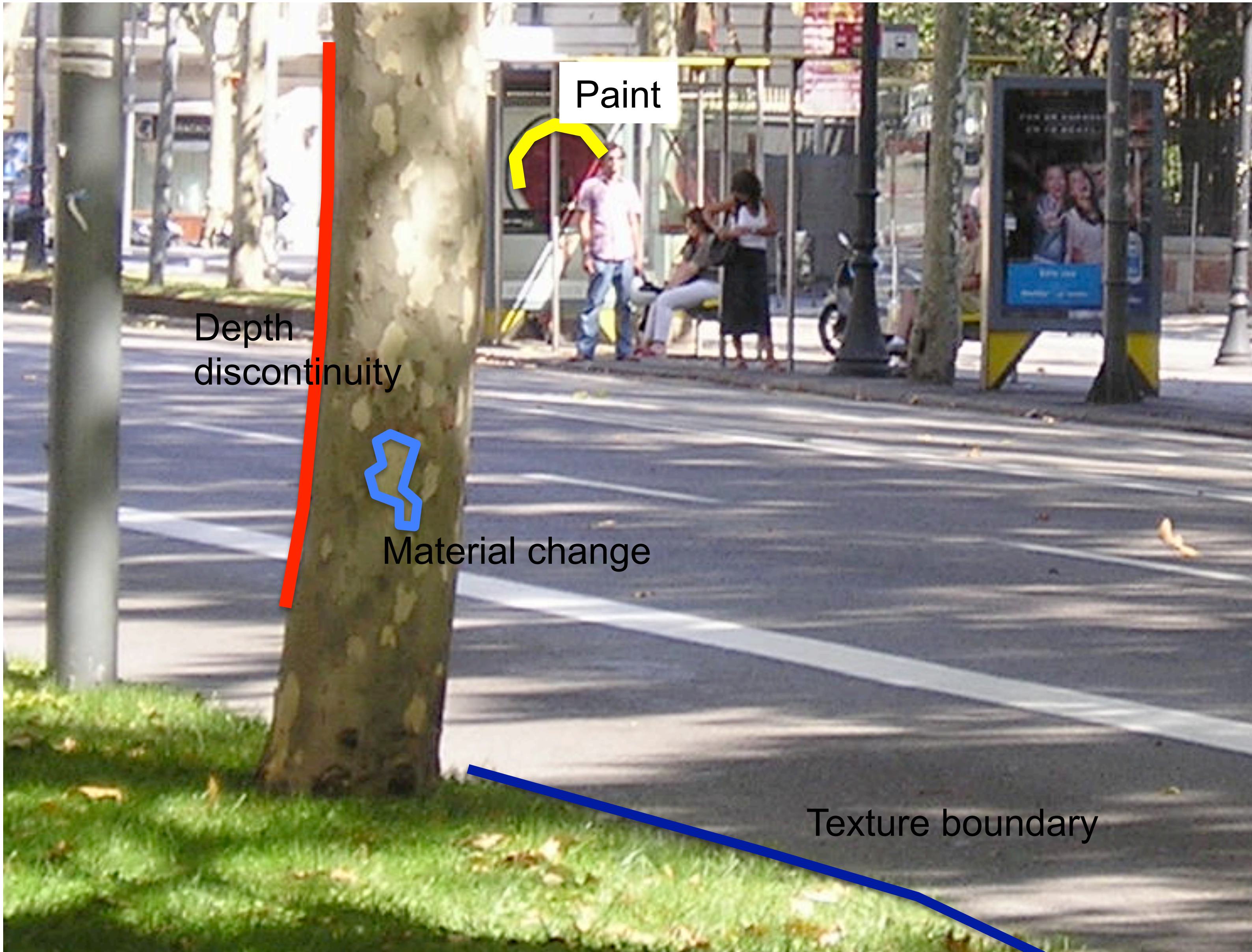
- Edges
  - Canny edge detector
  - Pb

- Segmentation
  - Clustering
  - Spectral methods

Finding edges

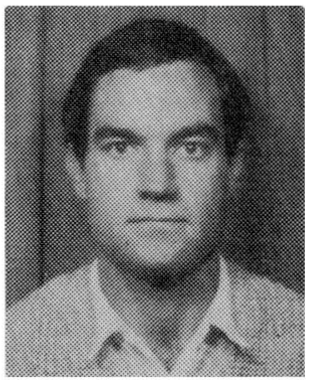
What is an edge?

What is an edge?



A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE



John Canny (S'81–M'82) was born in Adelaide, Australia, in 1958. He received the B.Sc. degree in computer science and the B.E. degree from Adelaide University in 1980 and 1981, respectively, and the S.M. degree from the Massachusetts Institute of Technology, Cambridge, in 1983.

He is with the Artificial Intelligence Laboratory, M.I.T. His research interests include low-level vision, model-based vision, motion planning for robots, and computer algebra.

Mr. Canny is a student member of the Association for Computing Machinery.

From lecture 1

Finding edges in the image

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

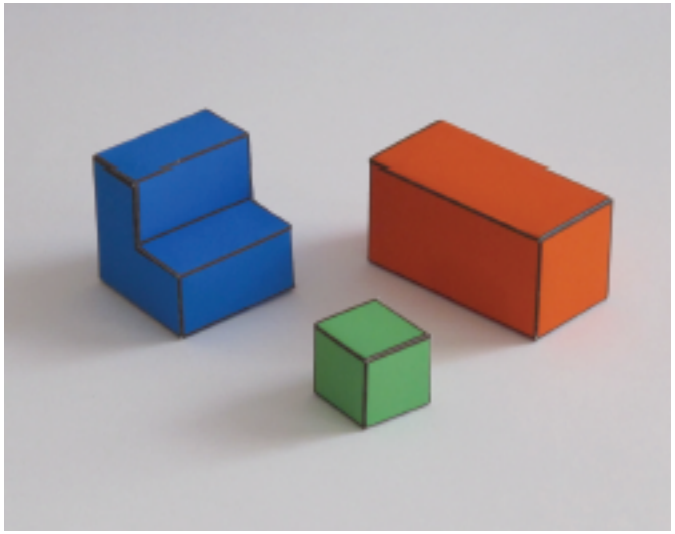
Edge strength $E(x, y) = |\nabla \mathbf{I}(x, y)|$

Edge orientation: $\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I}/\partial y}{\partial \mathbf{I}/\partial x}$

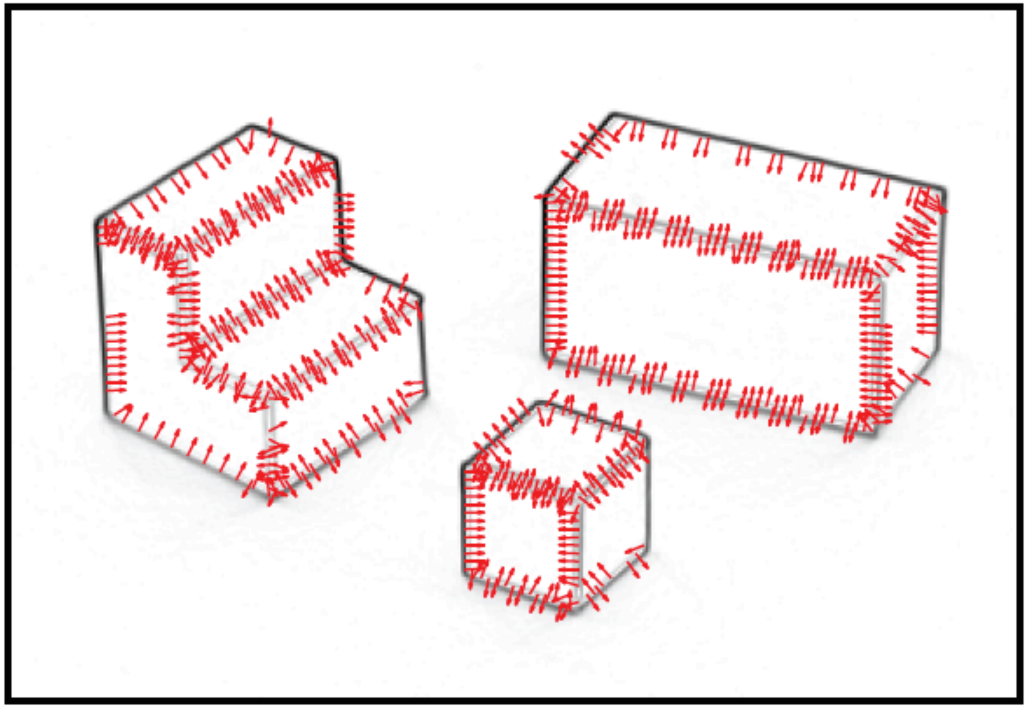
Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

From lecture 1

Finding edges in the image



$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right) \qquad \mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$$

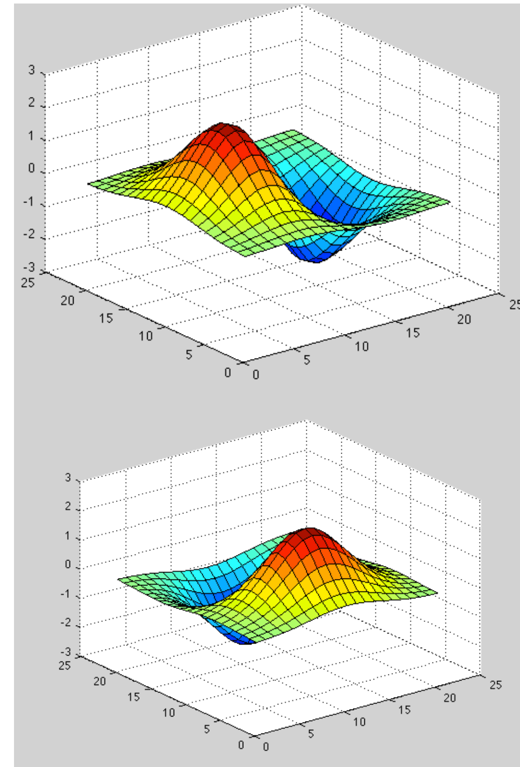


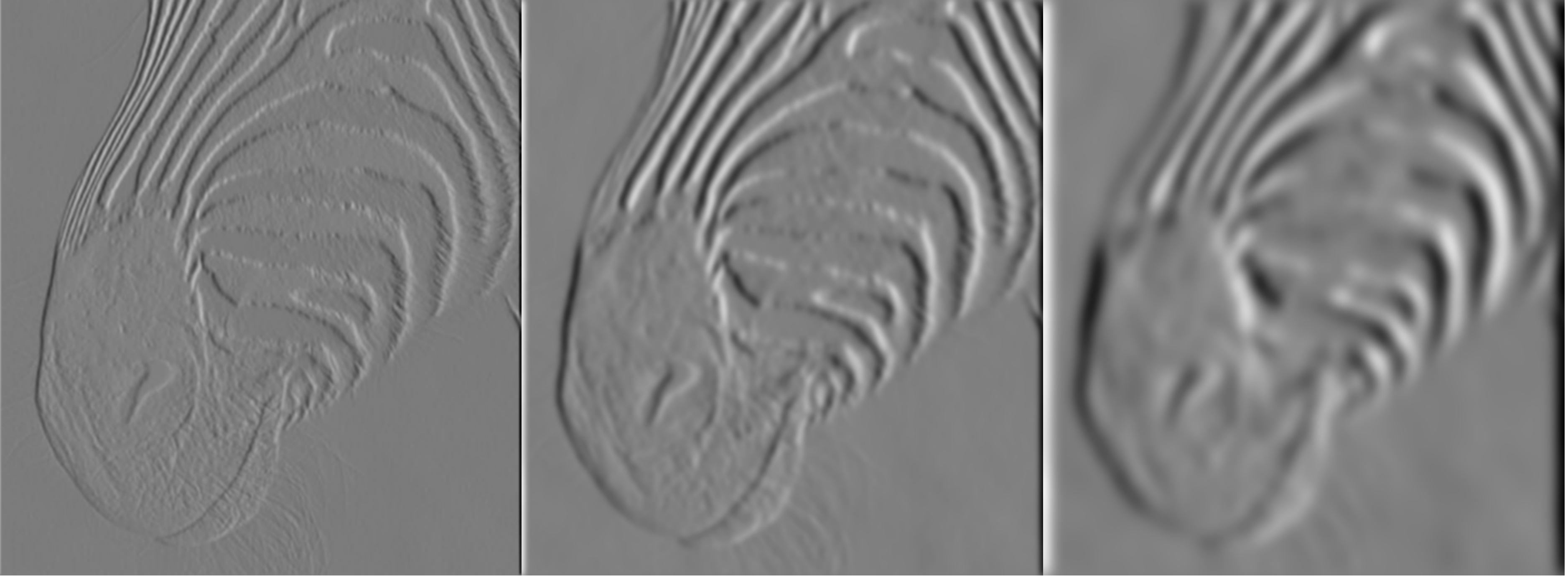
A better way of computing derivatives:

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

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

Scale





1 pixel

3 pixels

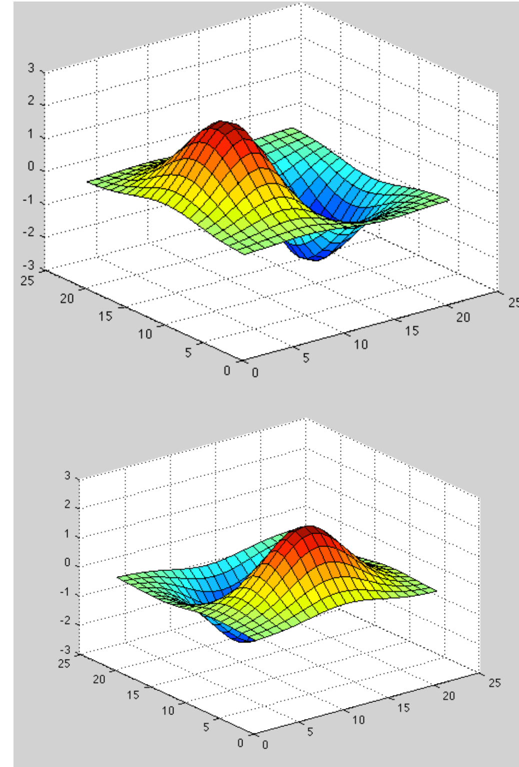
7 pixels

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

Forsyth, 2002

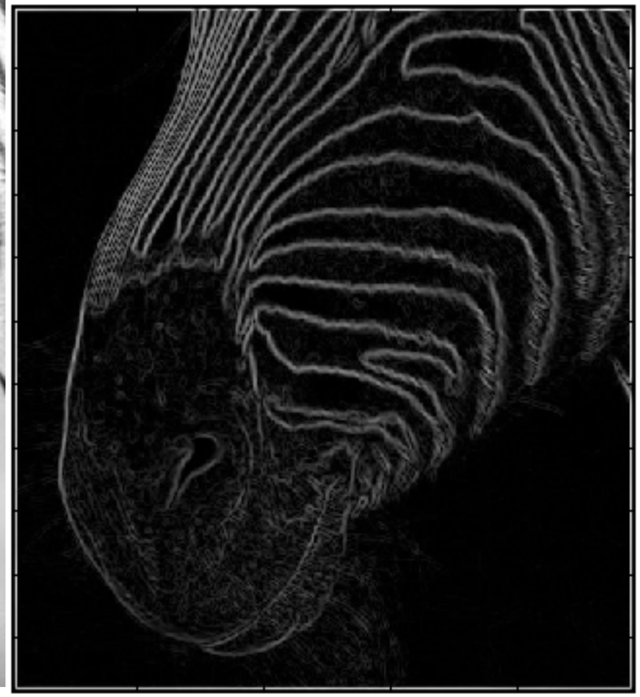
$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

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



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

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



Gradient magnitudes at scale 1

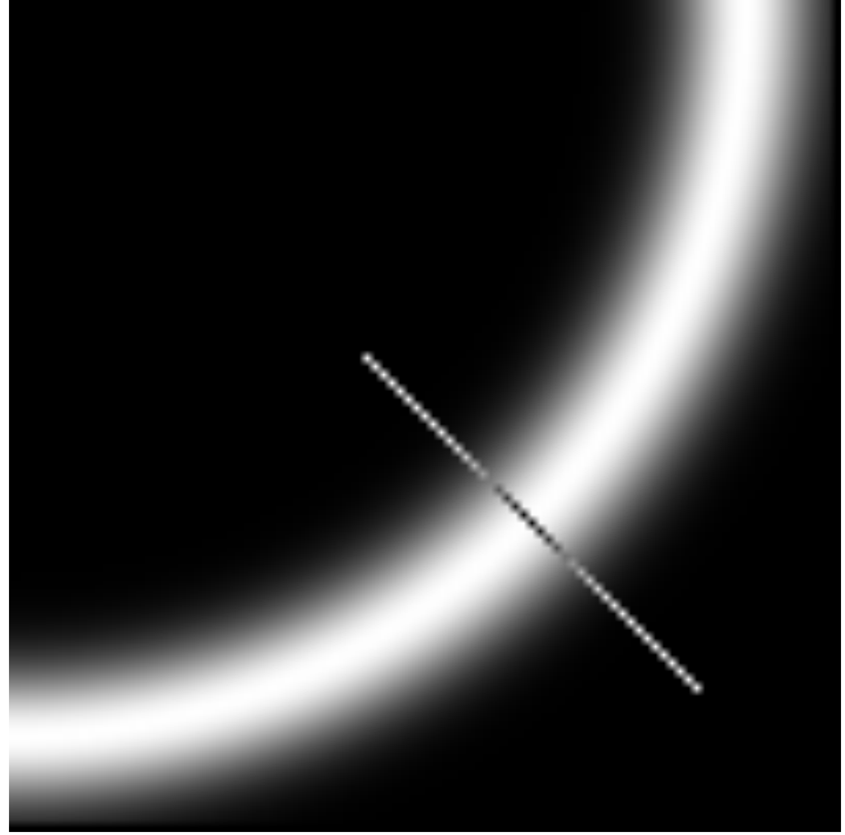
Gradient magnitudes at scale 2

Issues:

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

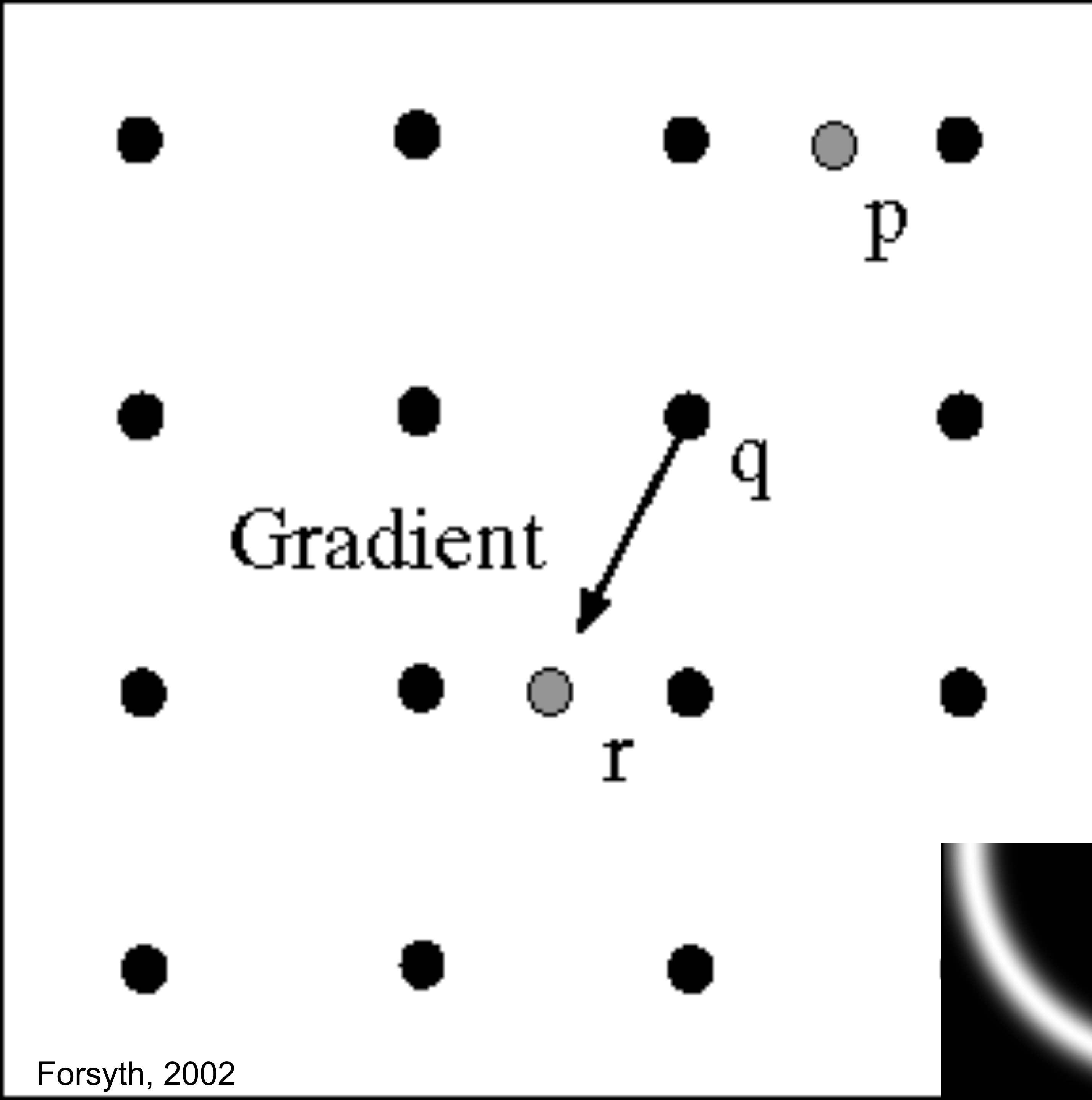
The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Forsyth, 2002



We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

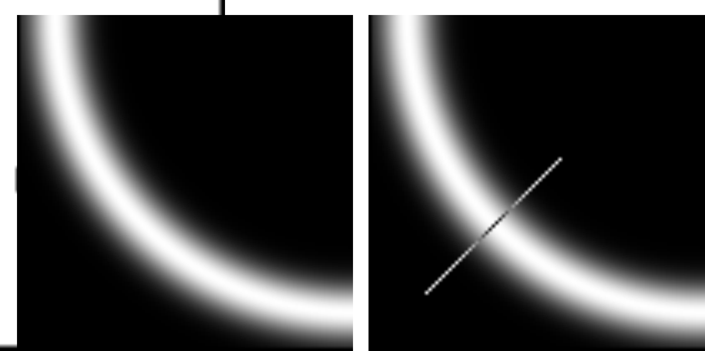
Forsyth, 2002



Forsyth, 2002

Non-maximum suppression

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



Examples: Non-Maximum Suppression

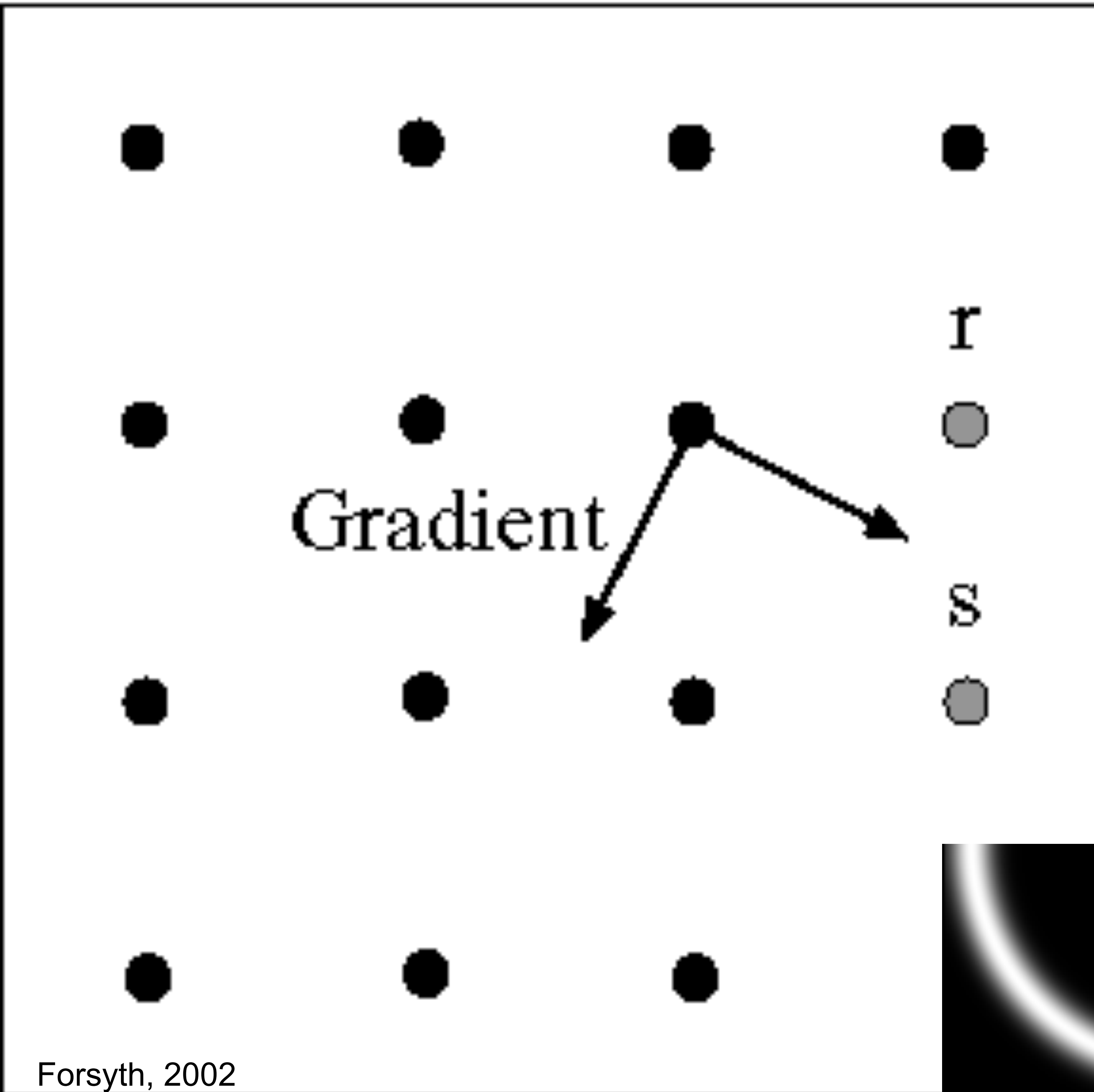
courtesy of G. Loy

Original image

Gradient magnitude

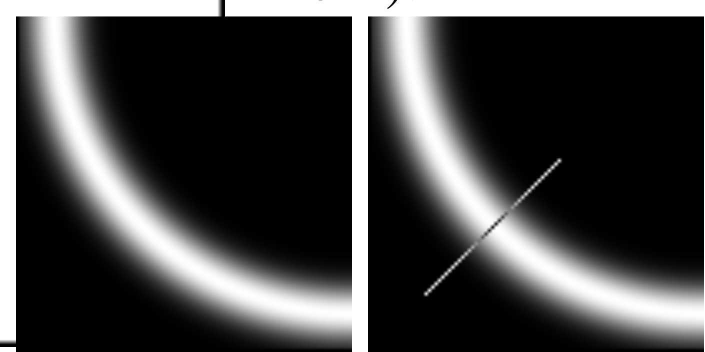
Non-maxima suppressed

Slide credit: Christopher Rasmussen



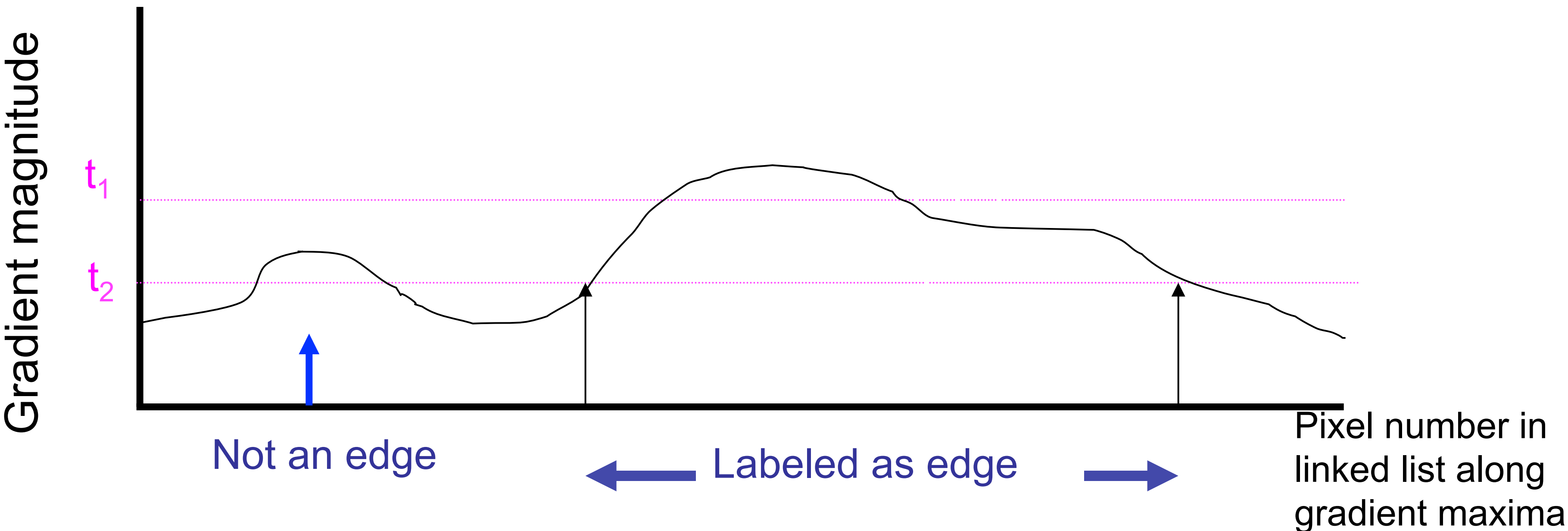
Predicting the next edge point

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



Closing edge gaps

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



Example: Canny Edge Detection

Original image

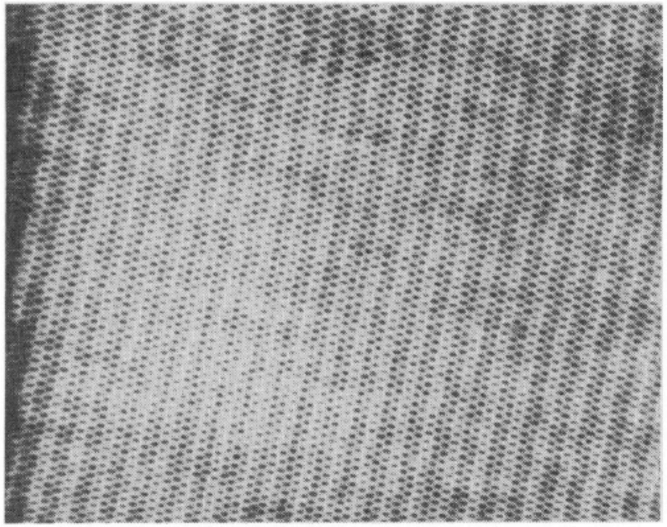
gap is gone

Strong + connected weak edges

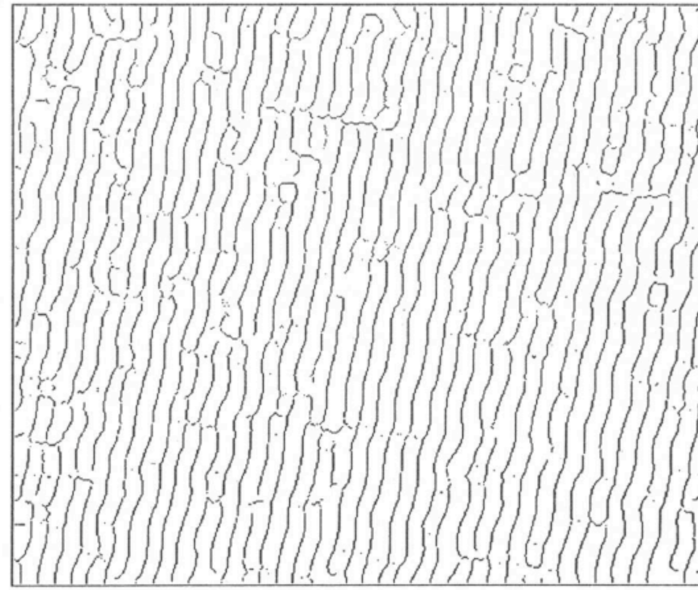
Strong edges only

Weak edges

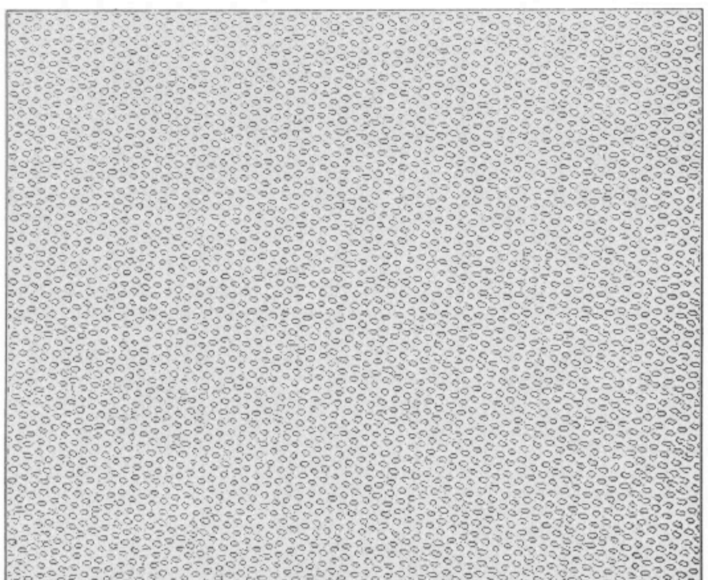
courtesy of G. Loy



(a)



(c)



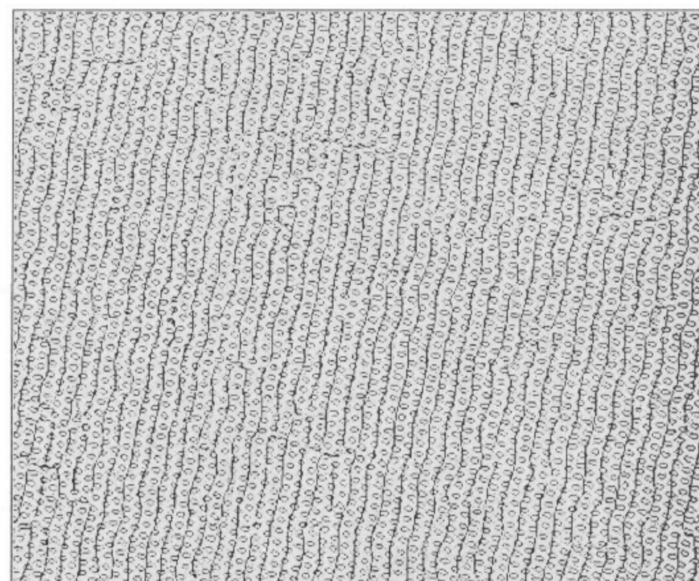


Fig. 9. (a) Handywipe image 576 by 454 pixels. (b) Edges from handywipe image at $\sigma = 1.0$. (c) $\sigma = 5.0$. (d) Superposition of the edges. (e)

edges

Isn’t it way to early to be thresholding, based on local, low-level pixel information alone?

Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues

David R. Martin, *Member*, *IEEE*, Charless C. Fowlkes, and Jitendra Malik, *Member*, *IEEE*

Abstract—The goal of this work is to accurately detect and localize boundaries in natural scenes using local image measurements. We formulate features that respond to characteristic changes in brightness, color, and texture associated with natural boundaries. In order to combine the information from these features in an optimal way, we train a classifier using human labeled images as ground truth. The output of this classifier provides the posterior probability of a boundary at each image location and orientation. We present precision-recall curves showing that the resulting detector significantly outperforms existing approaches. Our two main results are 1) that cue combination can be performed adequately with a simple linear model and 2) that a proper, explicit treatment of texture is required to detect boundaries in natural images.

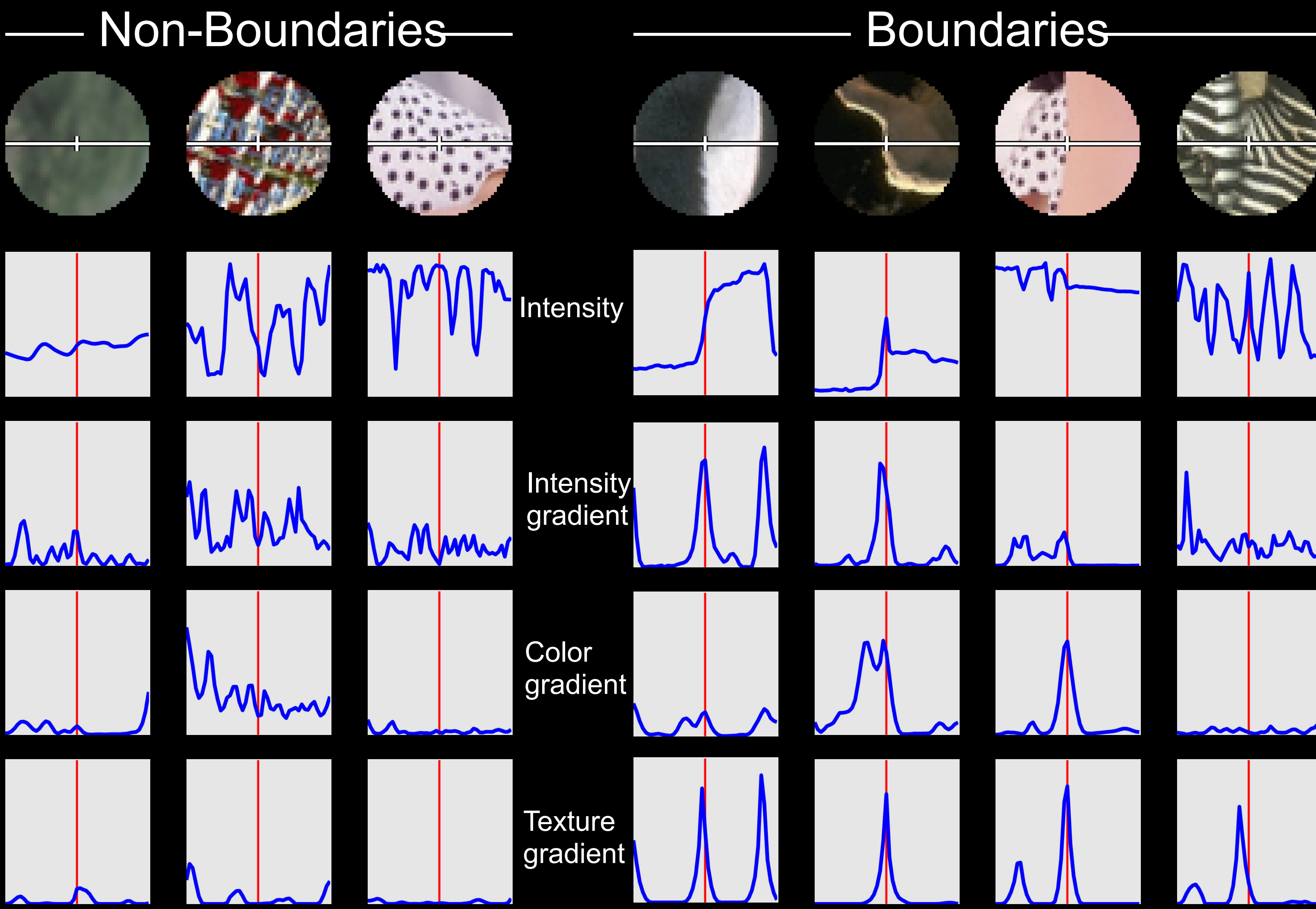
Slides credit: Jitendra Malik

Slides credit: Jitendra Malik

Slides credit: Jitendra Malik

Color Gray InvNeg

Slides credit: Jitendra Malik



Slides credit: Jitendra Malik

Dataflow

Image

Boundary Cues

Brightness

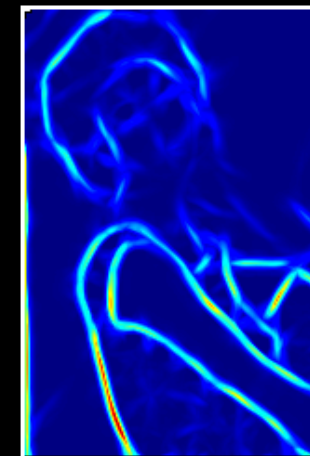
Color

Texture

Cue Combination

Model

P_b



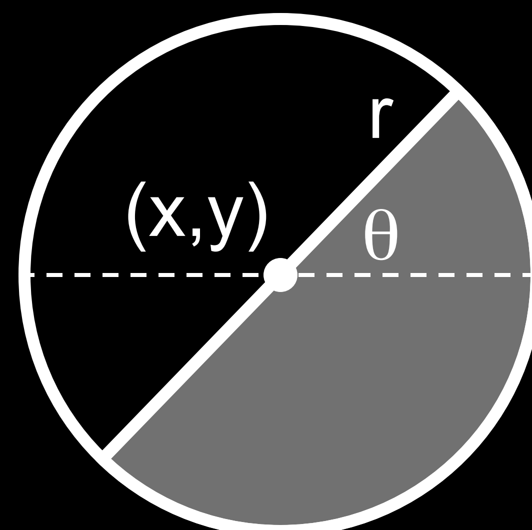
<u>Challenges</u>: texture cue, cue combination

<u>Goal</u>: learn the posterior probability of a boundary $P_b(x,y,\theta)$ from <u>local</u> information only

Slides credit: Jitendra Malik

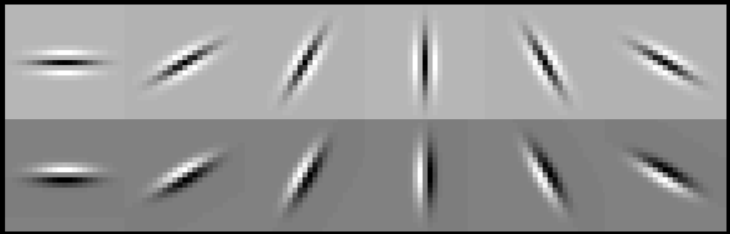
Brightness and Color Features

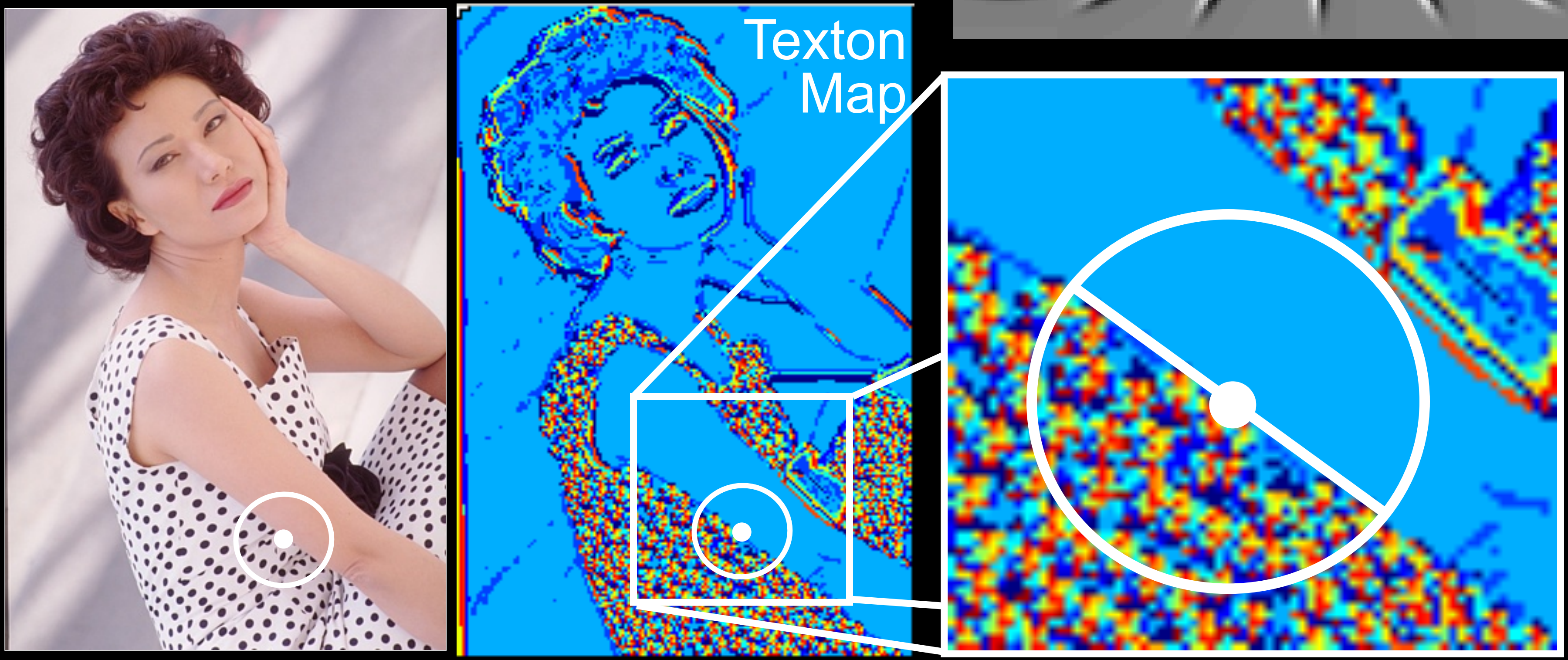
- 1976 CIE L*a*b* colorspace
- Brightness Gradient BG(x,y,r,θ)
  - χ^2 difference in L* distribution
- Color Gradient CG(x,y,r,θ)
  - χ^2 difference in a* and b* distributions



Slides credit: Jitendra Malik

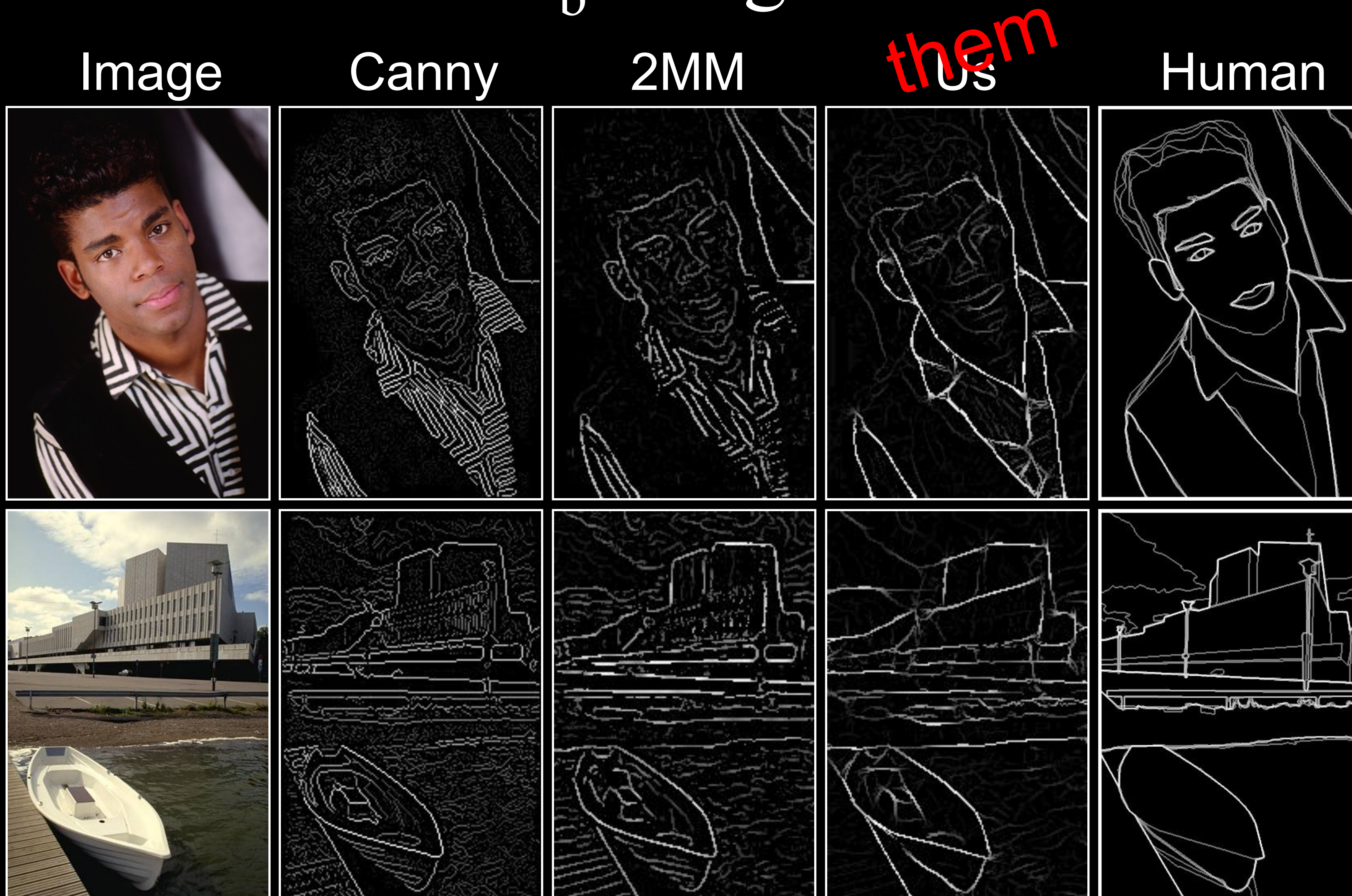
Texture Feature



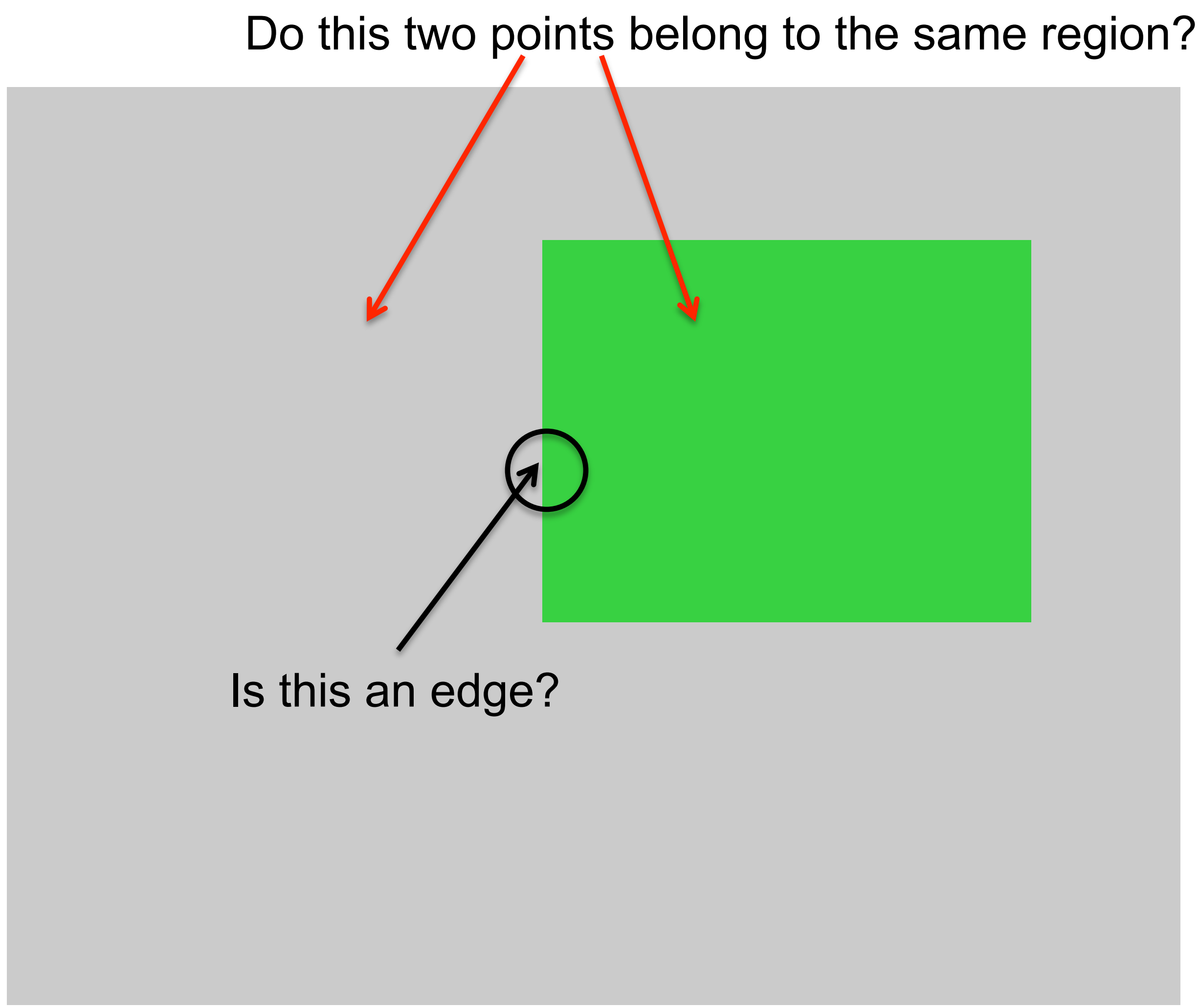


- Texture Gradient TG(x,y,r,θ)
  - χ^2 difference of texton histograms
  - Textons are vector-quantized filter outputs

P_b Images



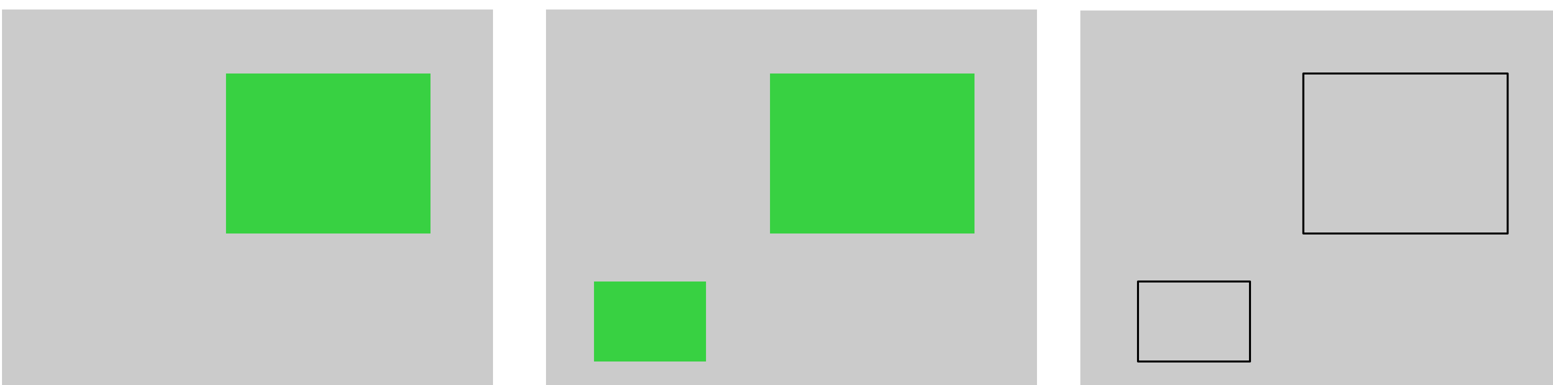
Slides credit: Jitendra Malik


Do this two points belong to the same region?
Is this an edge?

Segmentation

Issues

- How do we decide that two pixels are likely to belong to the same region?



- How many regions are there?

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

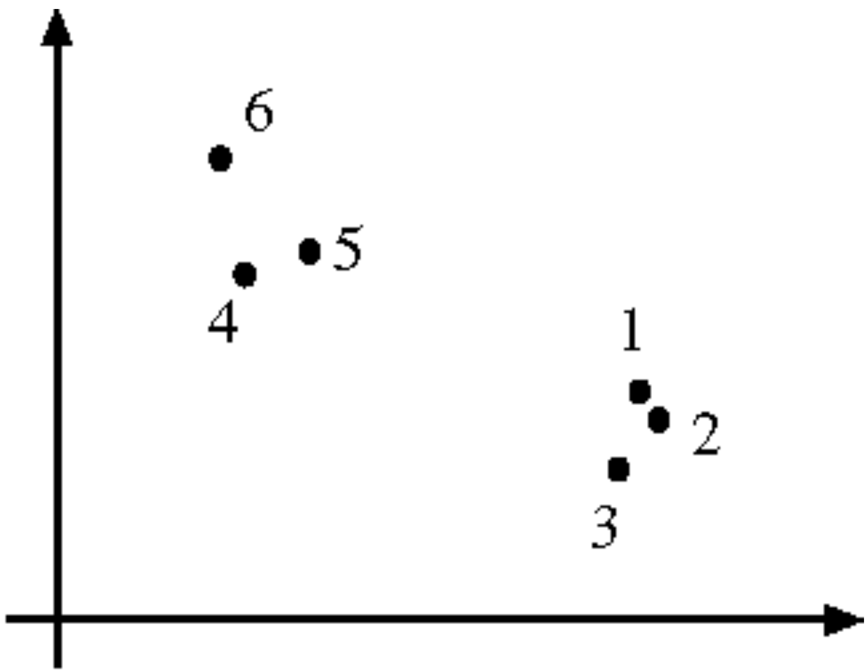
Algorithm 15.3: Agglomerative clustering, or clustering by merging

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

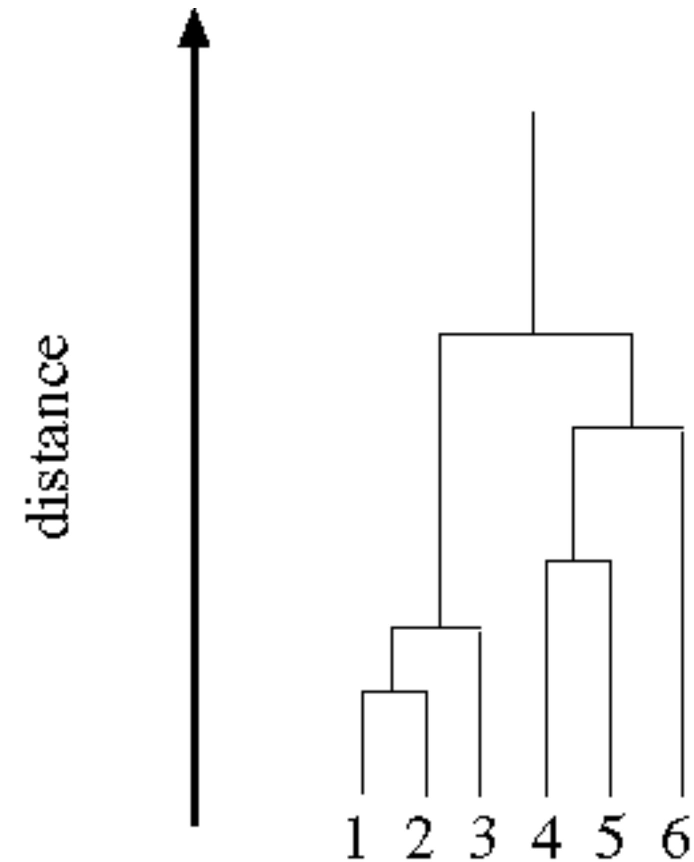
Algorithm 15.4: Divisive clustering, or clustering by splitting

```
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
```

Data set



Dendrogram obtained by agglomerative clustering



A simple segmentation algorithm

- Each pixel is described by a vector

  $z = [r, g, b]$ or $[Y\ u\ v]$, …

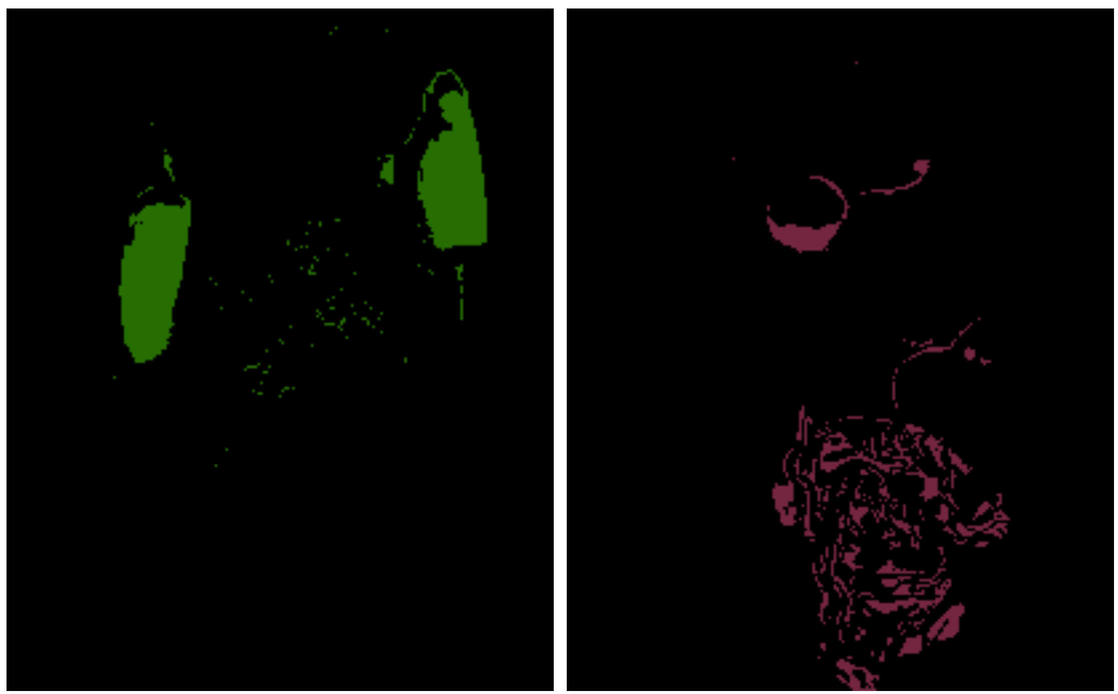
- Run a clustering algorithm (e.g. Kmeans) using some distance between pixels:

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

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



K-means using color alone, 11 segments.

Including spatial relationships

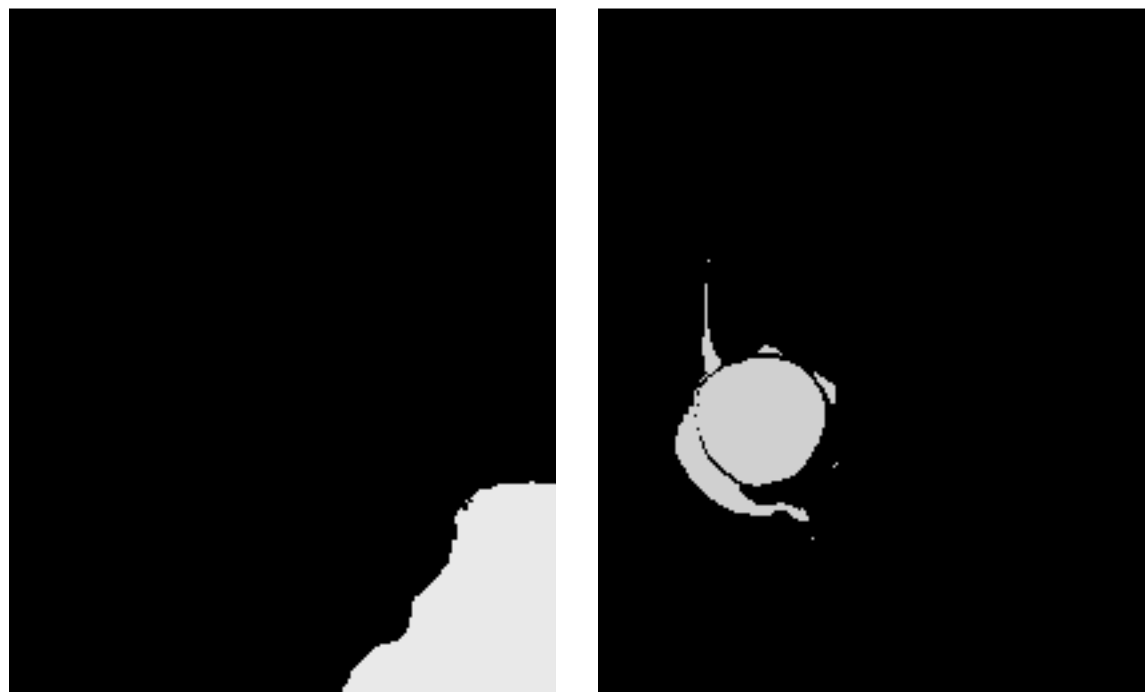
Augment data to be clustered with spatial coordinates.

$$
z = \begin{pmatrix} Y \\ u \\ v \\ x \\ y \end{pmatrix}
\begin{array}{l}
\left.\begin{array}{l} \\ \\ \\ \end{array}\right\} \text{color coordinates} \\
\left.\begin{array}{l} \\ \\ \end{array}\right\} \text{spatial coordinates}
\end{array}
$$

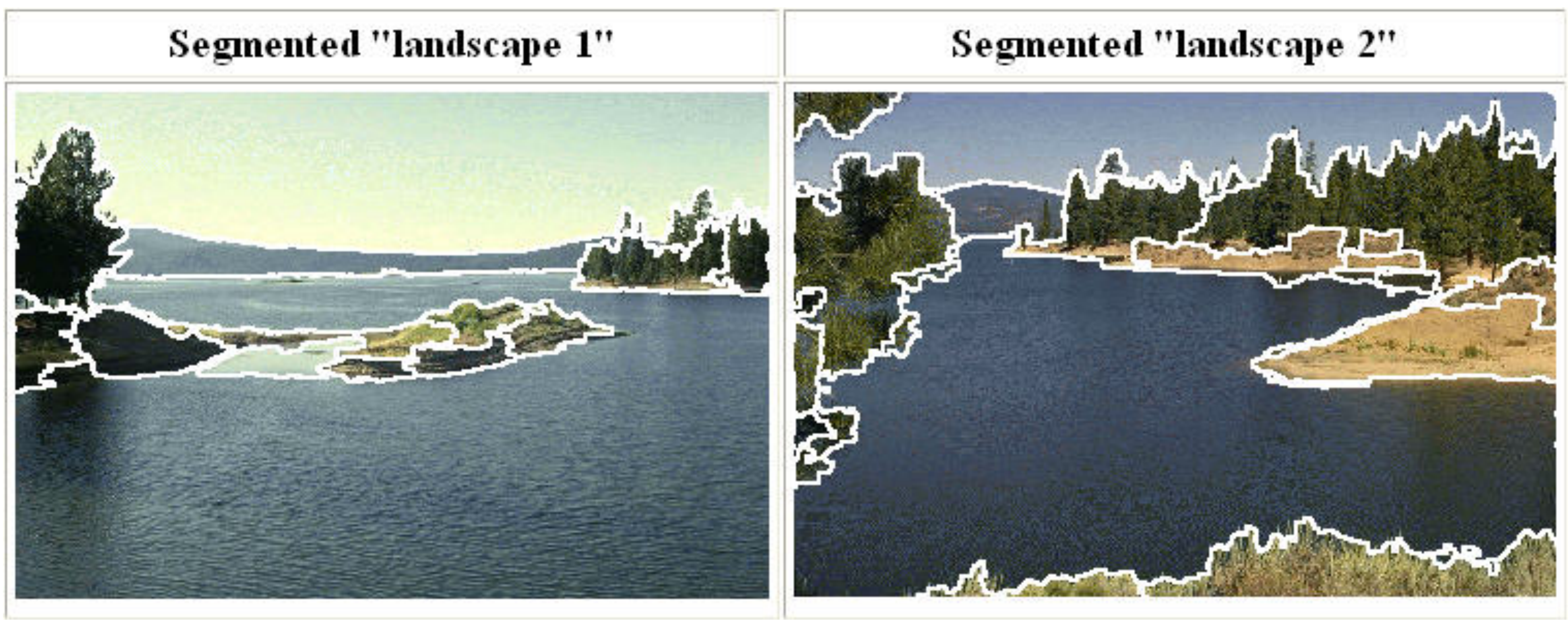
K-means using colour and position, 20 segments

Still misses goal of perceptually pleasing segmentation!

Hard to pick K…



Mean Shift Segmentation



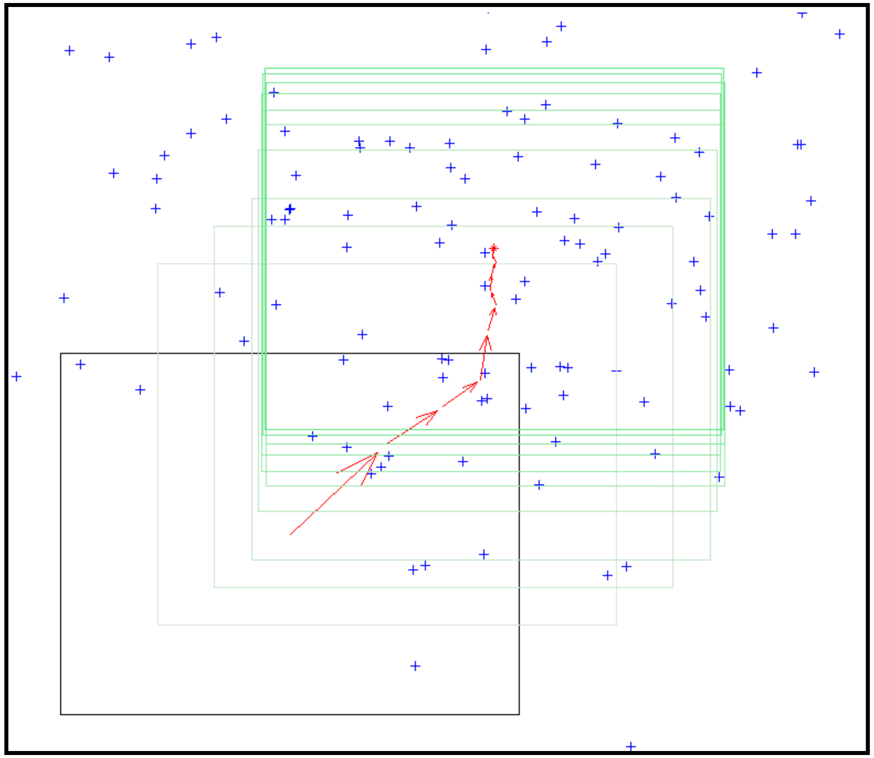
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

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:

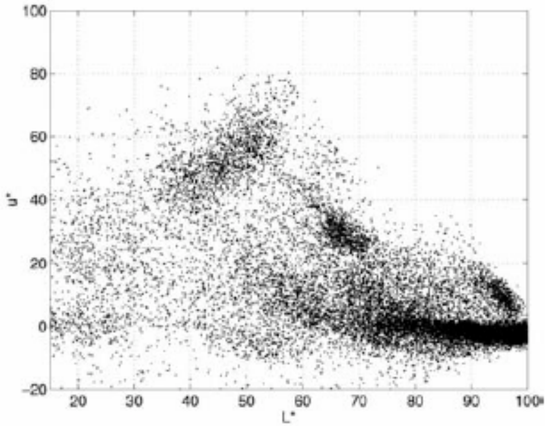


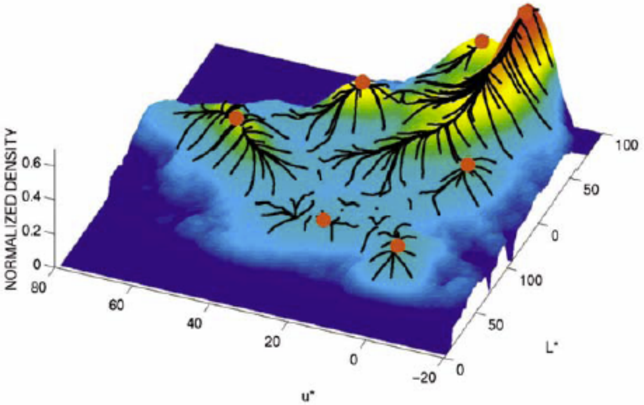
Two issues:
(1) Kernel to interpolate density based on sample positions.
(2) Gradient ascent to mode.

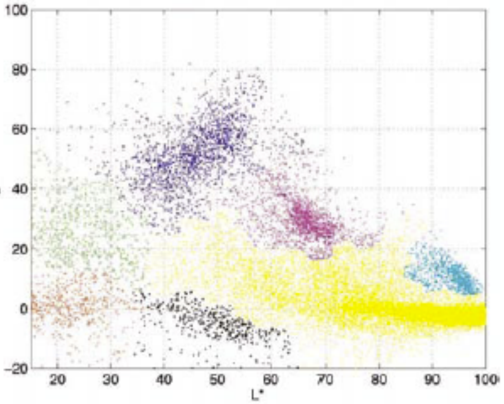
Mean Shift Segmentation

Mean Shift Segmentation Algorithm

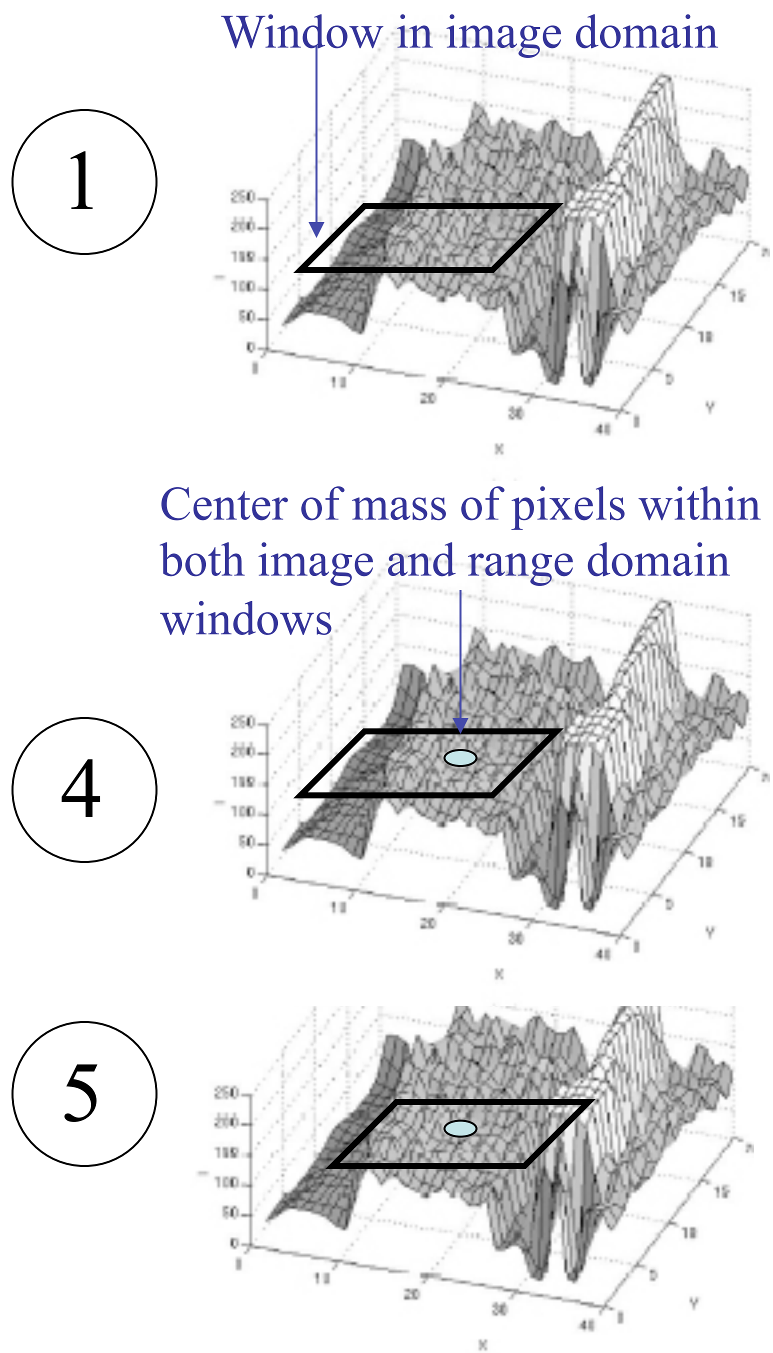
1. Convert the image into tokens (via color, gradients, texture measures etc).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same "peak" or mode.
5. The data these merged windows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

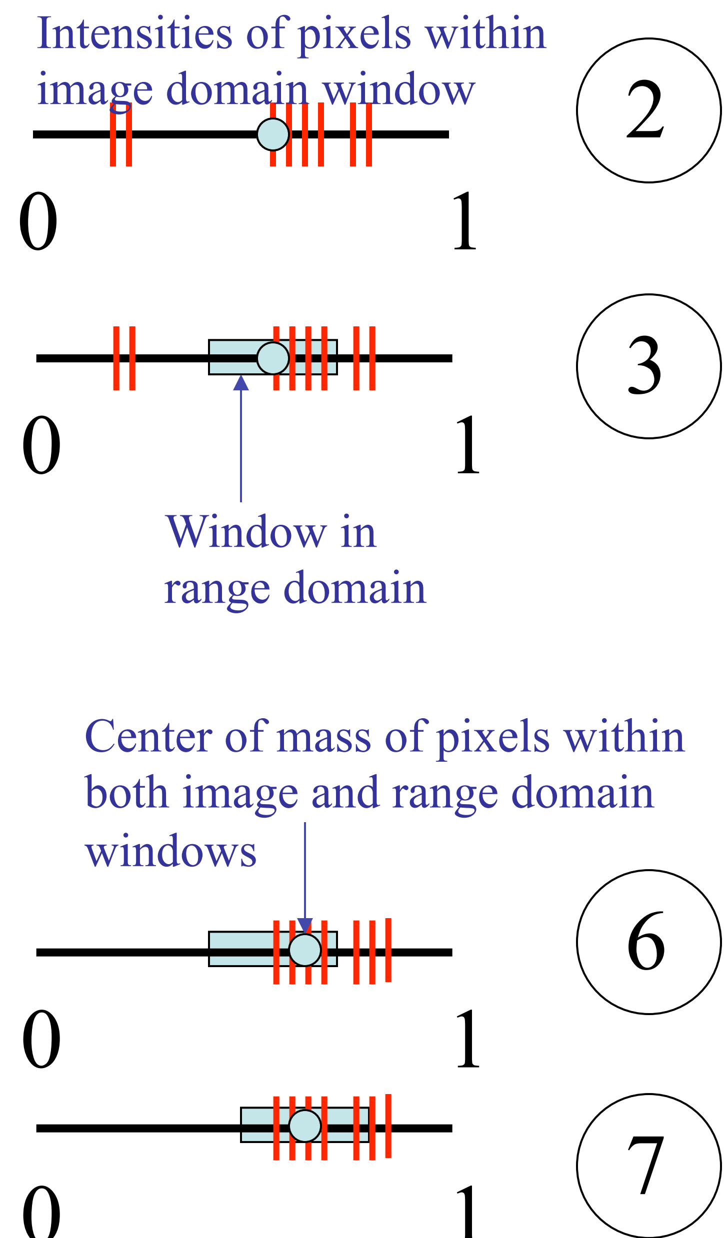






Apply mean shift jointly in the image (left col.) and range (right col.) domains





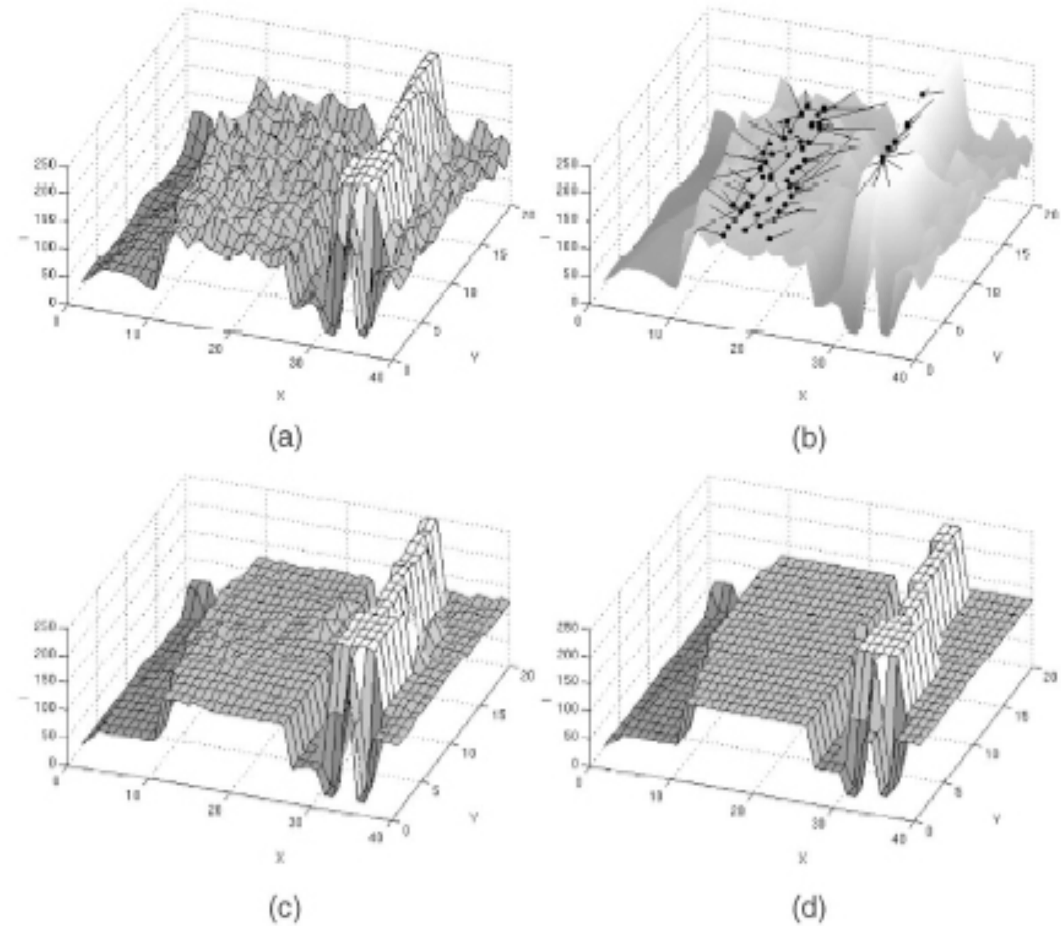
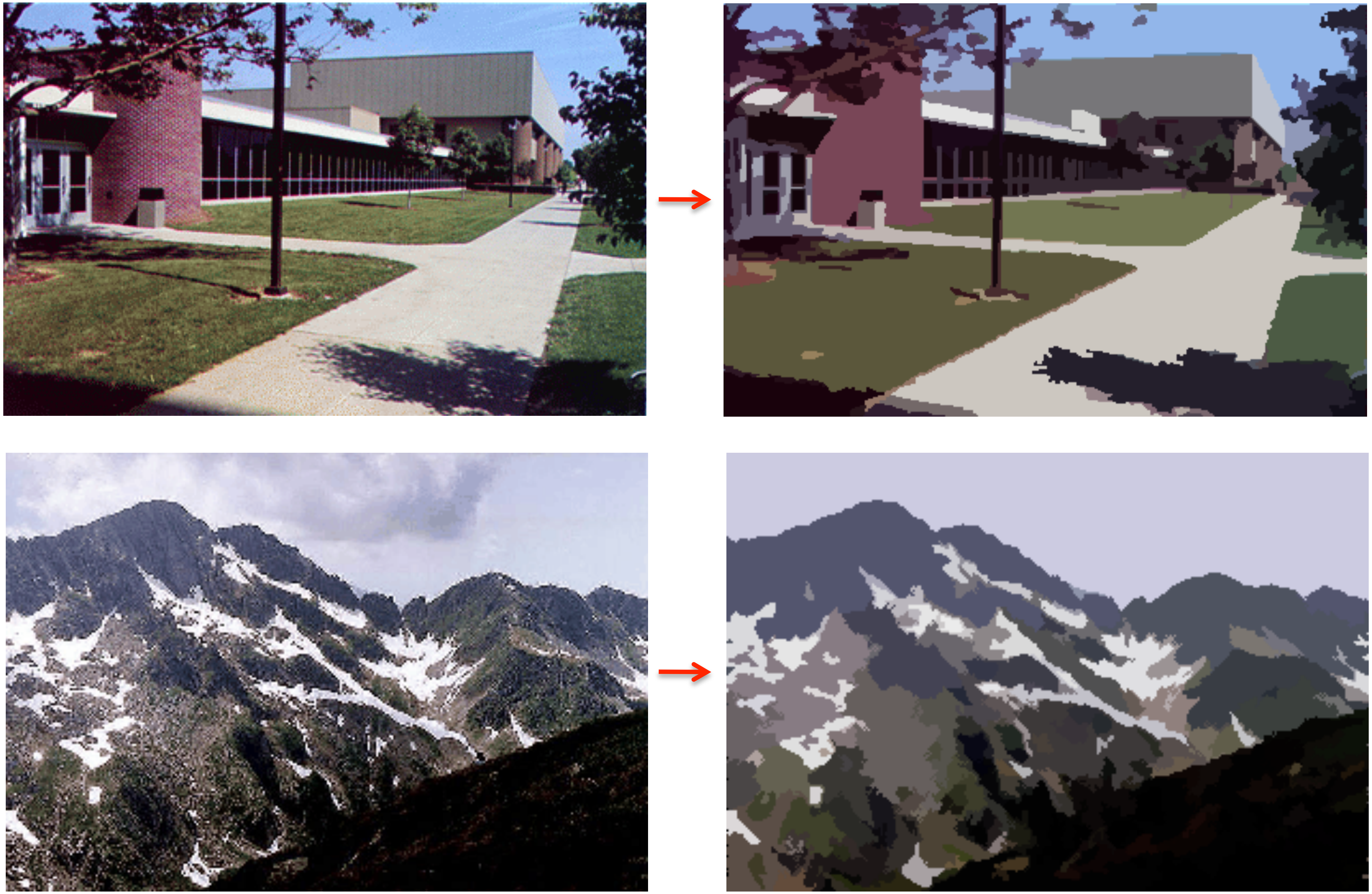


Fig. 4. Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixels on the plateau and on the line. The black dots are the points of convergence. (c) Filtering result $(h_s, h_r) = (8, 4)$. (d) Segmentation result.

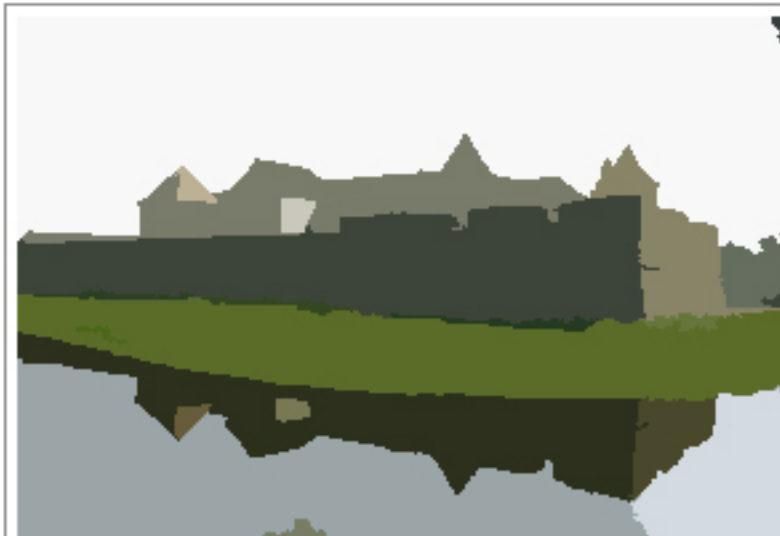
Comaniciu and Meer, IEEE PAMI vol. 24, no. 5, 2002

Mean Shift color&spatial Segmentation Results:



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

Mean Shift color&spatial Segmentation Results:

Original "fagaras"	Segmented
	

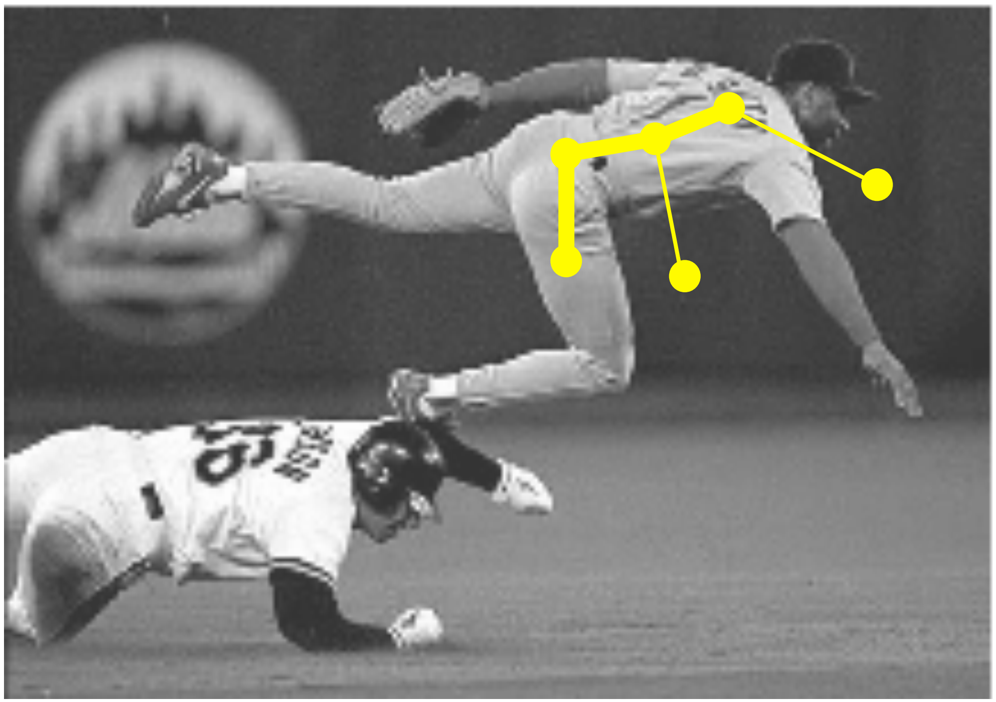
Original "building"	Segmented
	

Graph-Theoretic Image Segmentation

A different way of thinking about segmentation…

Graph-Theoretic Image Segmentation

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



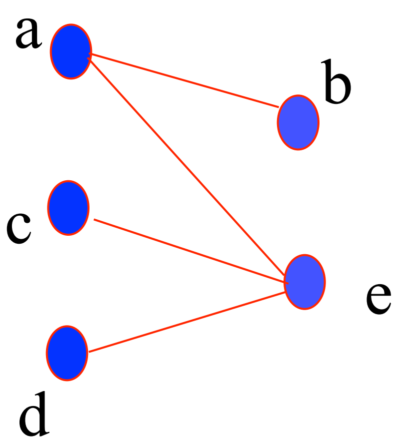
V: image pixels

E: connections between pairs of nearby pixels

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

Segmentation = graph partition

Graphs Representations

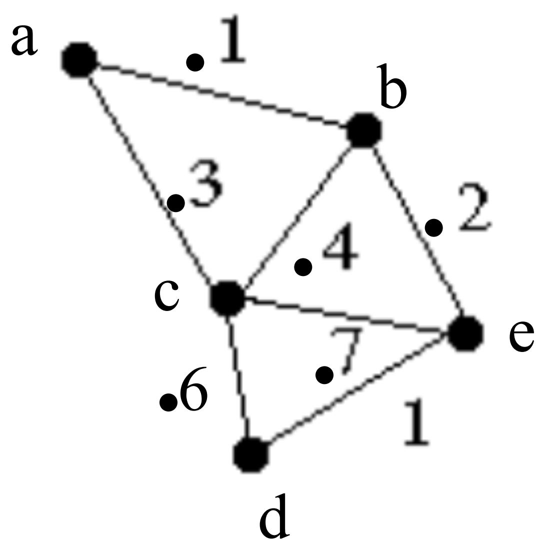


$$
\begin{array}{c c}
 & \begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}
\end{array}
$$

Adjacency Matrix

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

A Weighted Graph and its Representation



Affinity Matrix

$$W = \begin{bmatrix} 1 & .1 & .3 & 0 & 0 \\ .1 & 1 & .4 & 0 & .2 \\ .3 & .4 & 1 & .6 & .7 \\ 0 & 0 & .6 & 1 & 1 \\ 0 & .2 & .7 & 1 & 1 \end{bmatrix}$$

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

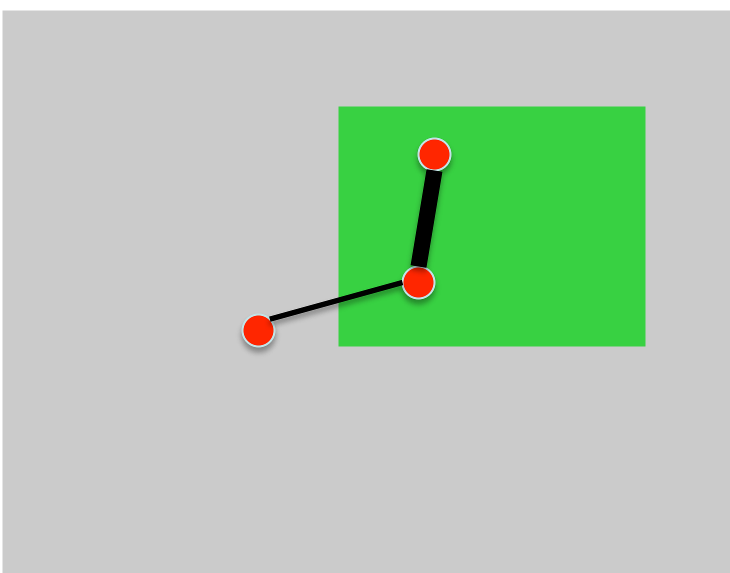
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

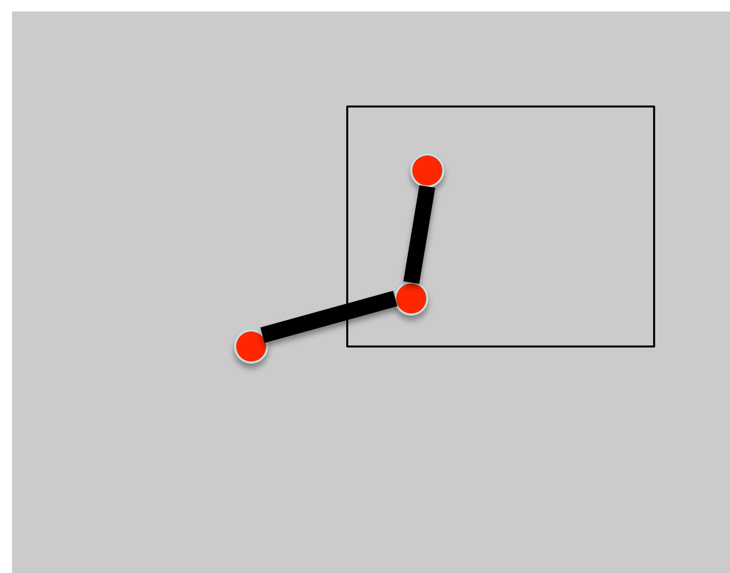
Affinity between pixels

Similarities among pixel descriptors

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

σ = Scale factor…
it will hunt us later





Affinity between pixels

Similarities among pixel descriptors

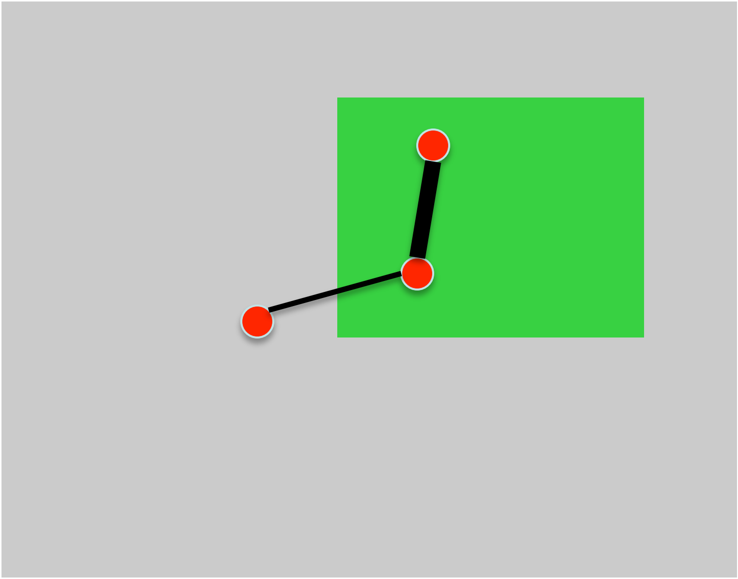
$$W_{ij} = \exp(-\| z_i - z_j \|^2 / \sigma^2)$$

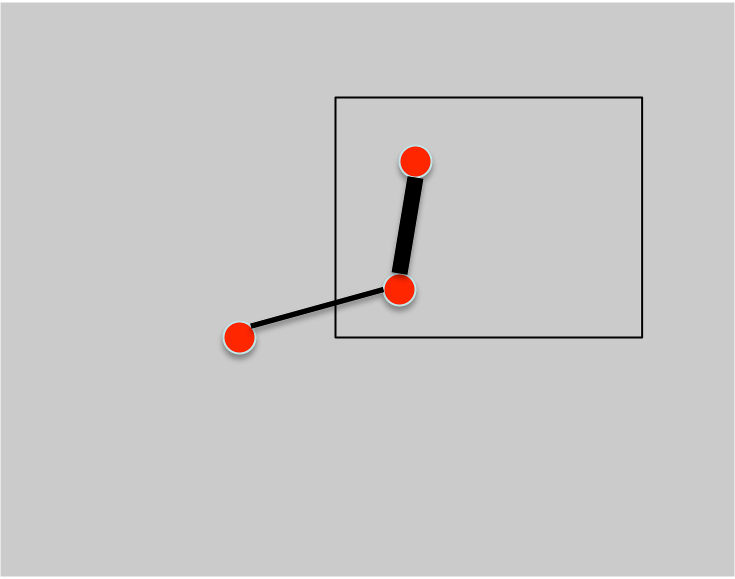
σ = Scale factor…
it will hunt us later

Interleaving edges

$$W_{ij} = 1 - \max_{\text{Line between i and j}} Pb$$

With Pb = probability of boundary



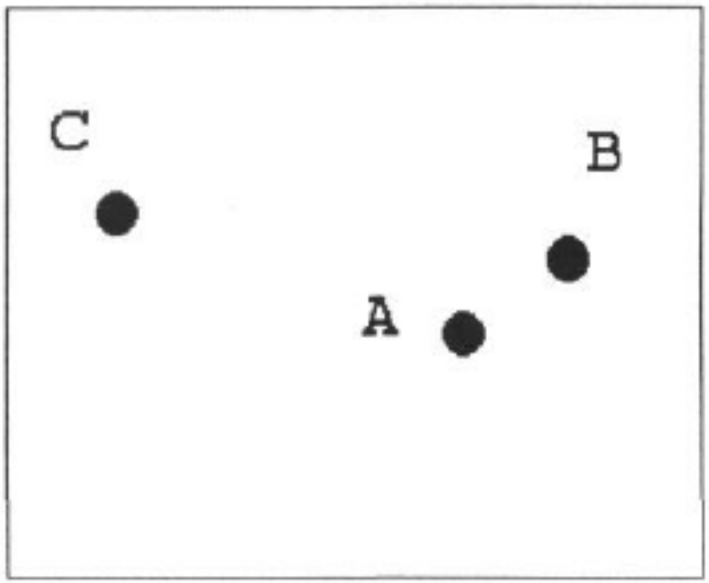


Feature grouping by "relocalisation" of eigenvectors of the proximity matrix

British Machine Vision Conference, pp. 103-108, 1990

Guy L. Scott
Robotics Research Group
Department of Engineering Science
University of Oxford

H. Christopher Longuet-Higgins
University of Sussex
Falmer
Brighton



Three points in feature space

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

With an appropriate σ

W=

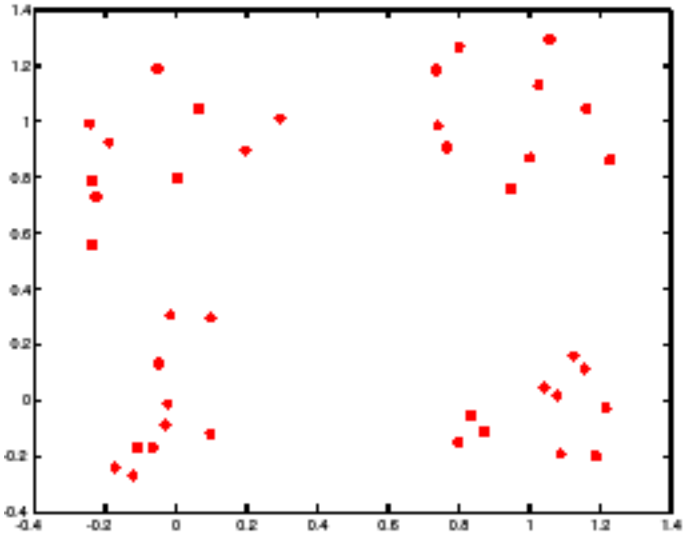
	A	B	C
A	1.00	0.63	0.03
B	0.63	1.00	0.0
C	0.03	0.0	1.00

The eigenvectors of W are:

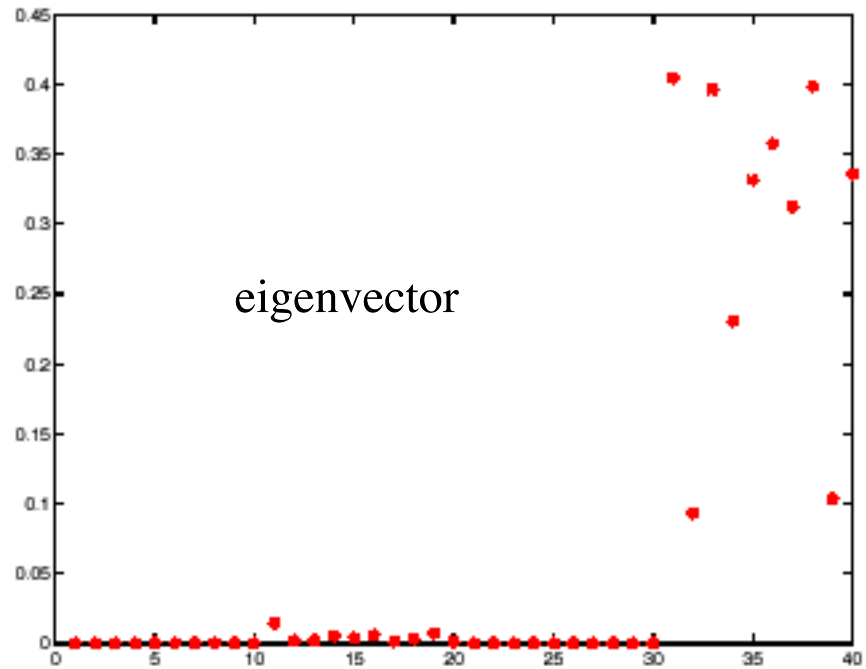
	E_1	E_2	E_3
Eigenvalues	1.63	1.00	0.37
A	-0.71	-0.01	0.71
B	-0.71	-0.05	-0.71
C	-0.04	1.00	-0.03

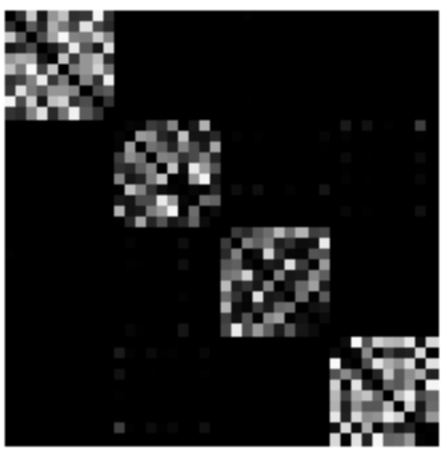
The first 2 eigenvectors group the points as desired…

Example eigenvector



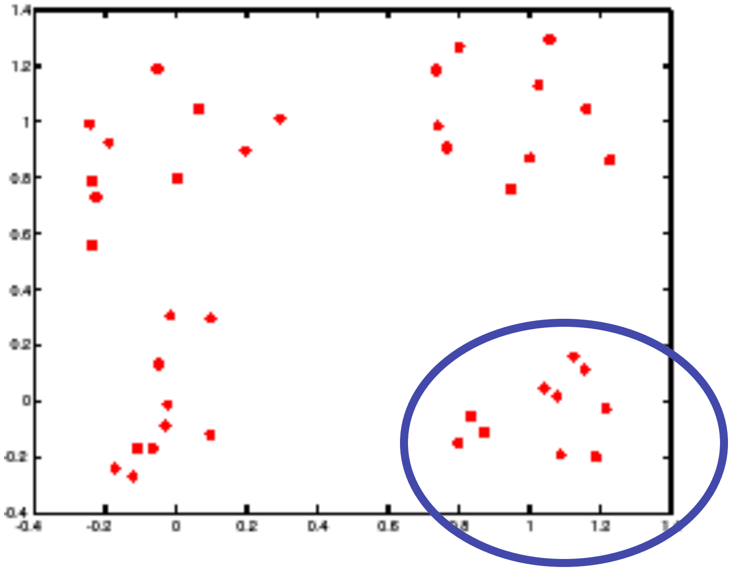
points



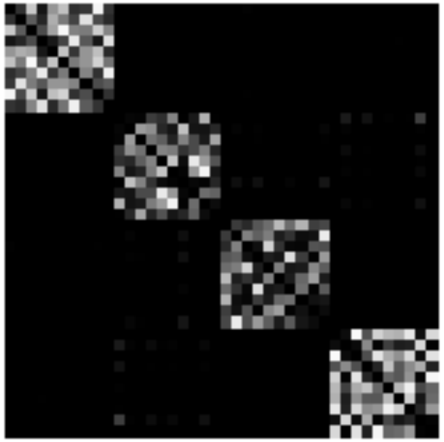


Affinity matrix

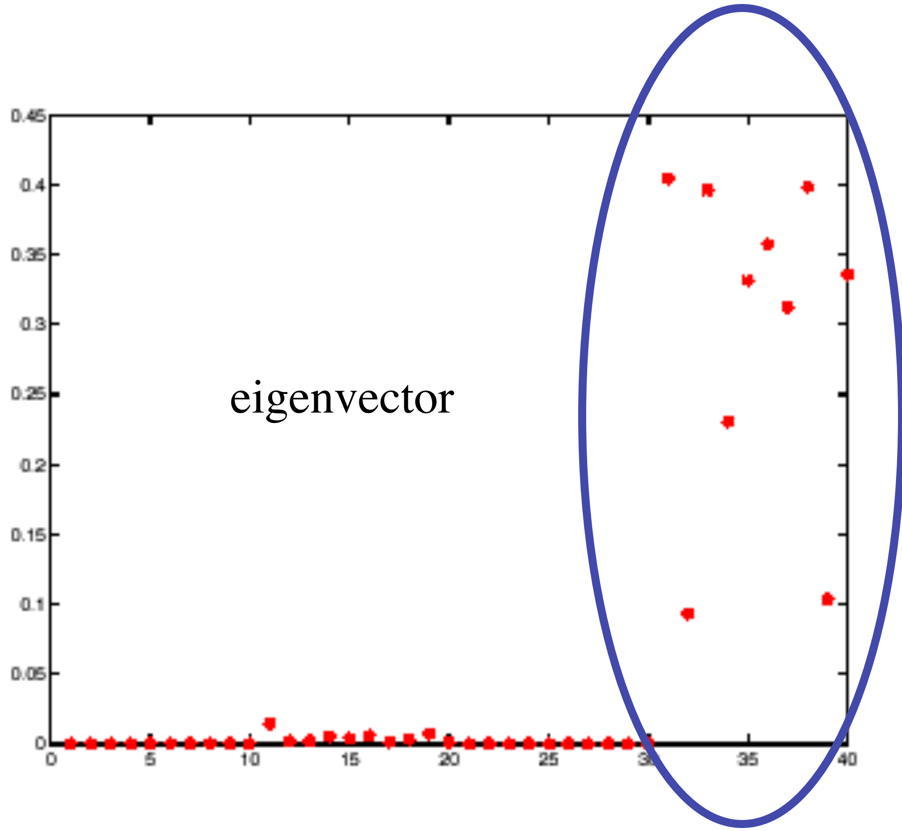
Example eigenvector



points

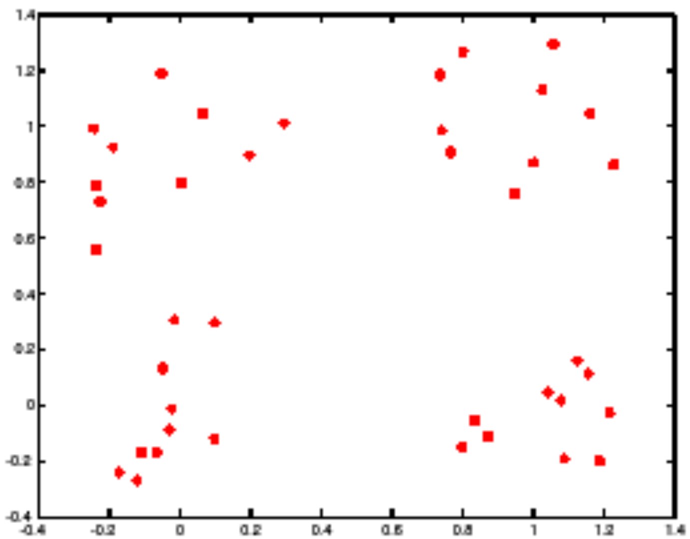


Affinity matrix

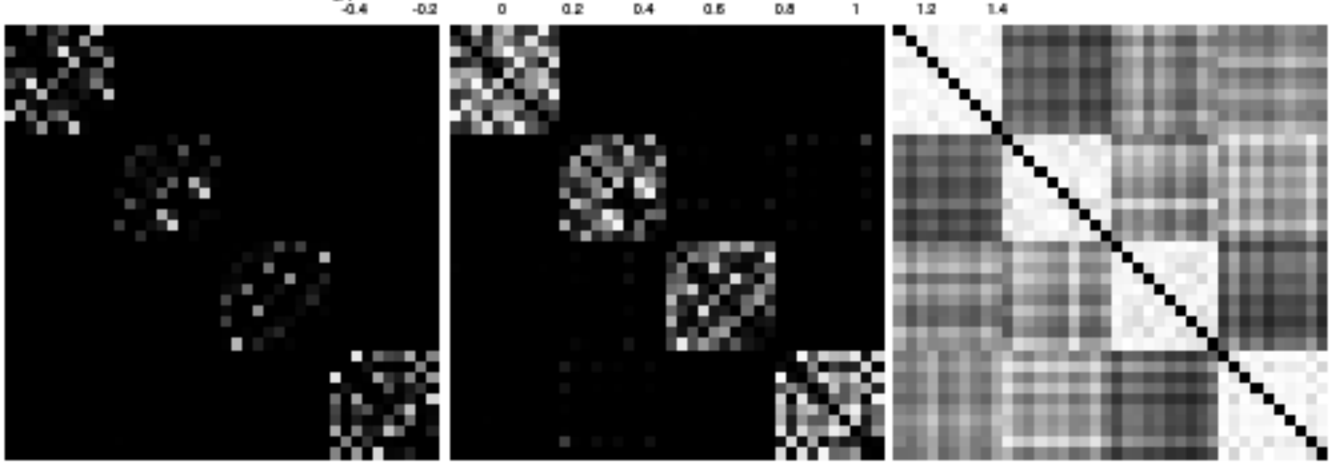


Scale affects affinity

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



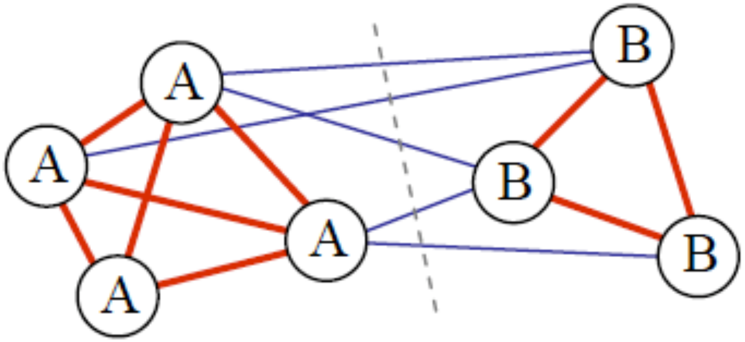
σ=.2



σ=.1 σ=.2 σ=1

Minimum Cut

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

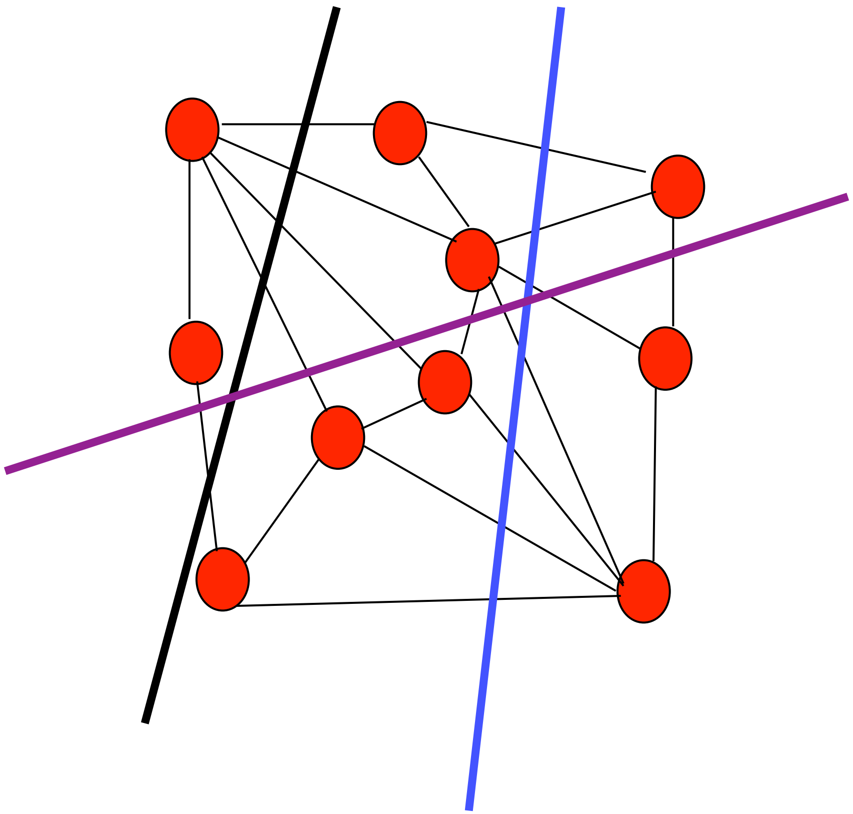


Cut: sum of the weight of the cut edges:

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

with $\mathrm{A} \cap \mathrm{B} = \varnothing$

Minimum Cut



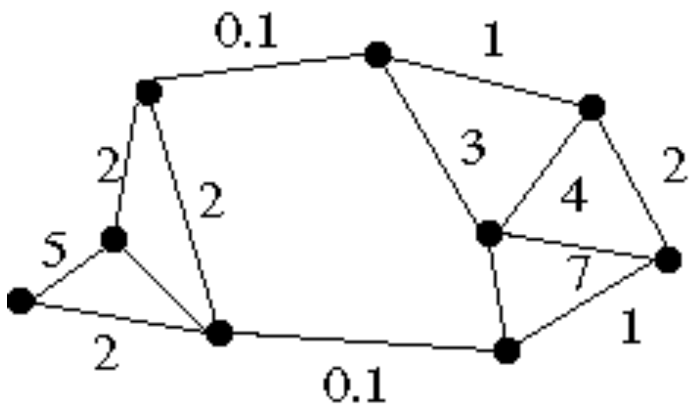
Minimum cut is the cut of minimum weight

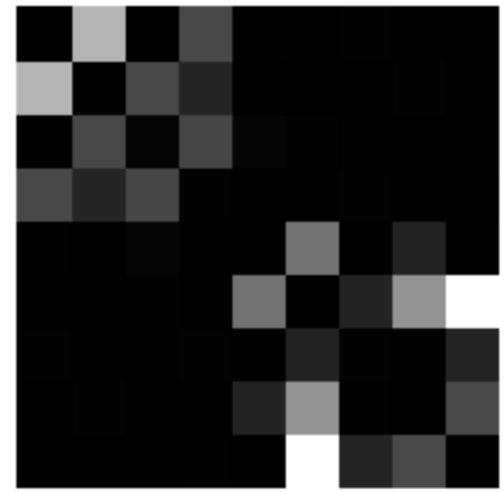
$$cut(A,B) = \sum_{u \in A, v \in B} W(u,v),$$

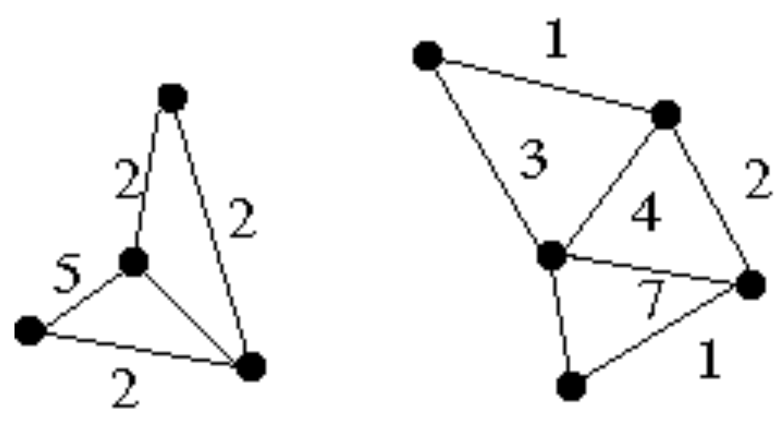
with $A \cap B = \varnothing$

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Minimum Cut and Clustering



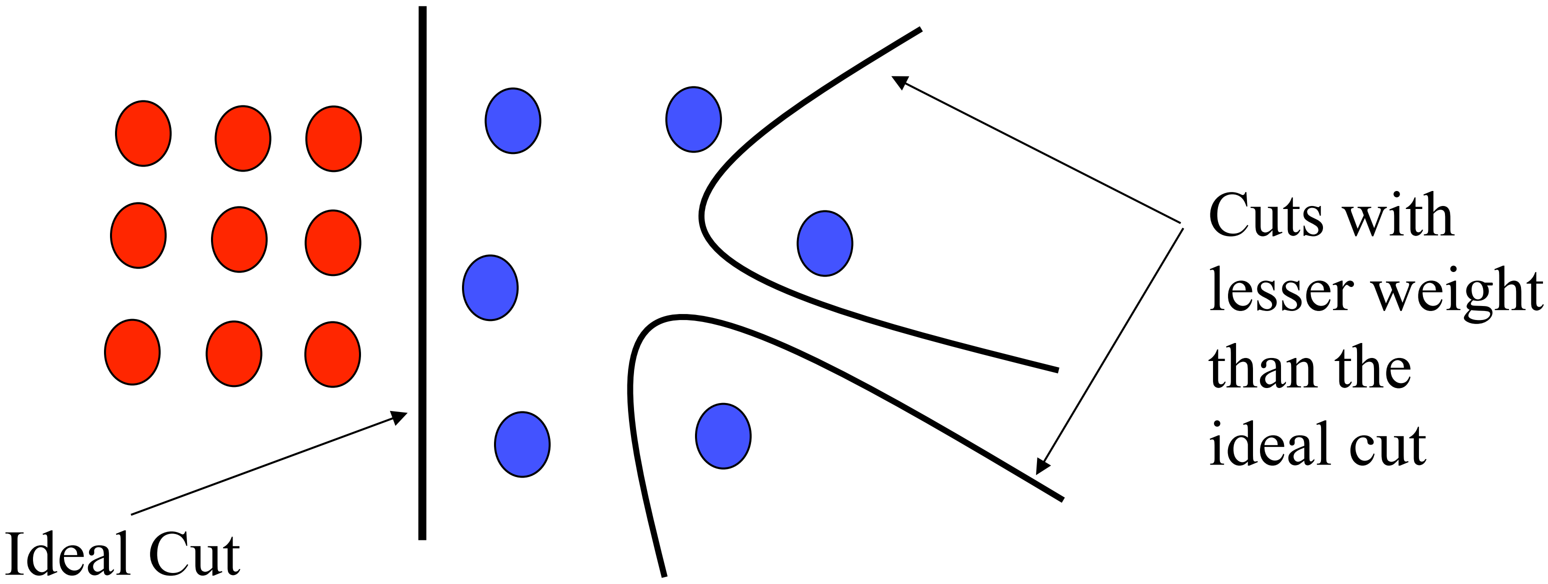




* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Drawbacks of Minimum Cut

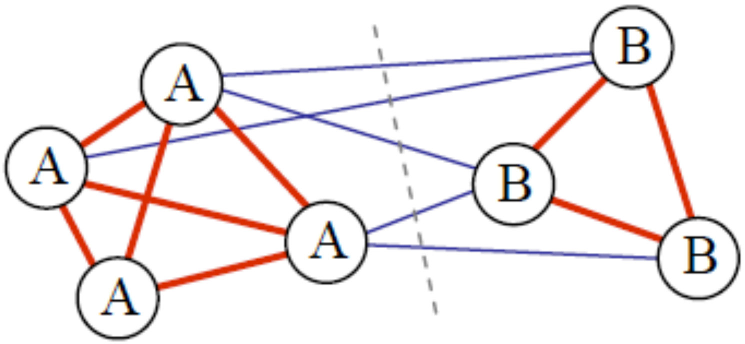
- Weight of cut is directly proportional to the number of edges in the cut.



* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Normalized cuts

Write graph as V, one cluster as A and the other as B



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

cut(A,B) is sum of weights with one end in A and one end in B

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

with $\mathrm{A} \cap \mathrm{B} = \varnothing$

assoc(A,V) is sum of all edges with one end in A.

$$assoc(\mathrm{A},\mathrm{B}) = \sum_{u \in \mathrm{A}, v \in \mathrm{B}} \mathrm{W}(u,v)$$

A and B not necessarily disjoint

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

$$D_{ii} = \sum_j W_{ij}$$

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

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

$$= \ldots$$

after simplification, Shi and Malik derive

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

W = affinity matrix

[Malik]

Normalized cuts

Minimize:

Binary

Binary

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

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

- Instead, solve the generalized eigenvalue problem

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

- They show that the 2nd smallest eigenvector solution y is a good real-valued approx to the original normalized cuts problem. Then you 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

http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Grouping 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 being a measure of the similarity between the two nodes.

2. Solve $(\mathbf{D} - \mathbf{W})\boldsymbol{x} = \lambda \mathbf{D} \boldsymbol{x}$ for eigenvectors with the smallest eigenvalues.

3. Use the eigenvector with second smallest eigenvalue to bipartition the graph.

4. Decide if the current partition should be sub-divided, and recursively repartition the segmented parts if necessary.

Global optimization

- In this formulation, the segmentation becomes a global process.
- Decisions about what is a boundary are not local (as in Canny edge detector)

Boundaries of image regions defined by a number of attributes

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

[Malik]

Example

Affinity:

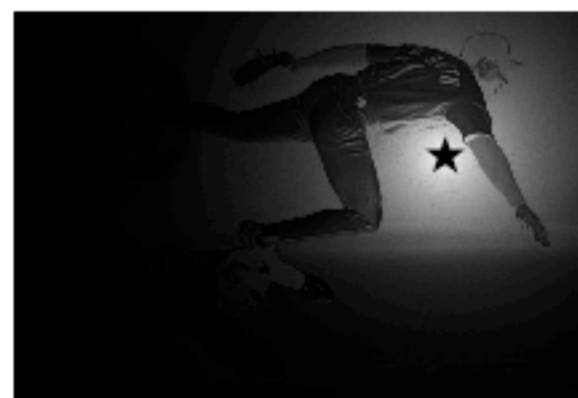
$$w_{ij} = \underbrace{e^{\frac{-\|\boldsymbol{F}_{(i)}-\boldsymbol{F}_{(j)}\|_2^2}{\sigma_I}}}_{\text{brightness}} * \underbrace{\begin{cases} e^{\frac{-\|\boldsymbol{X}_{(i)}-\boldsymbol{X}_{(j)}\|_2^2}{\sigma_X}} & \text{if } \|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases}}_{\text{Location}}$$

N pixels = ncols * nrows

N

W =

N



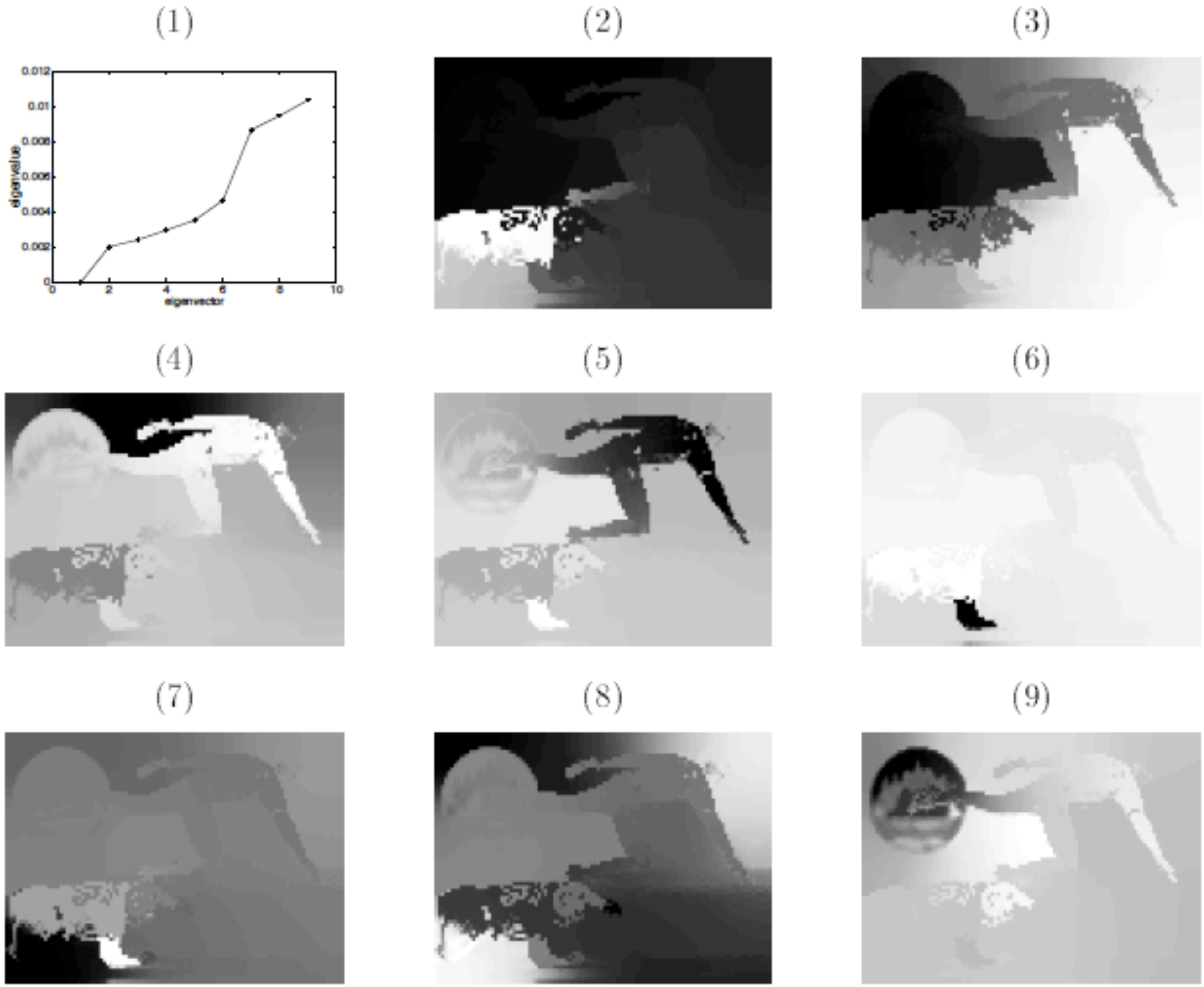


Figure 12: Subplot (1) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplot (2) - (9) shows the eigenvectors corresponding the 2nd smallest to the 9th smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

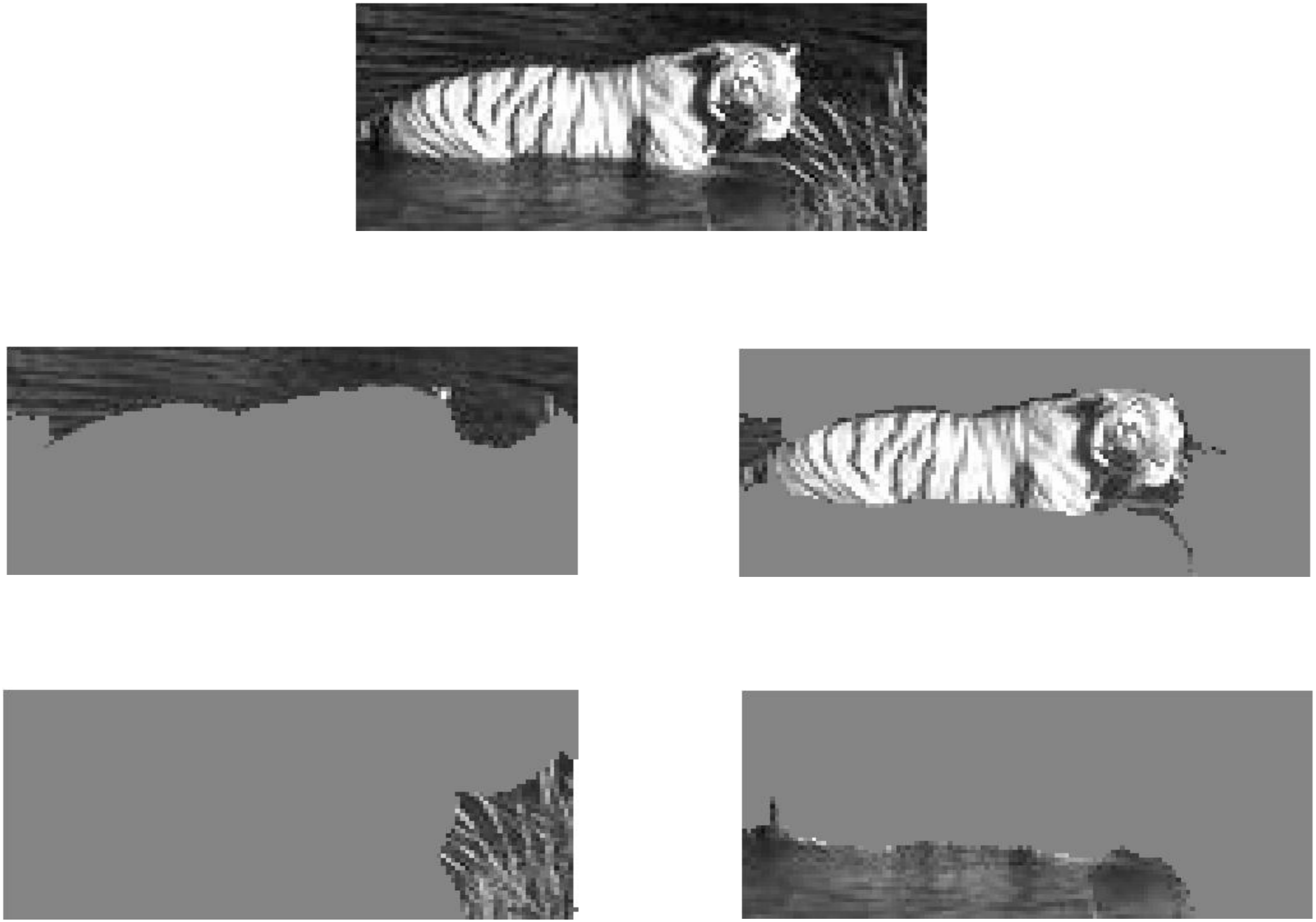
Brightness Image Segmentation

converge. On the 100×120 test images shown here, the normalized cut algorithm takes about 2 minutes on Intel Pentium 200MHz machines.

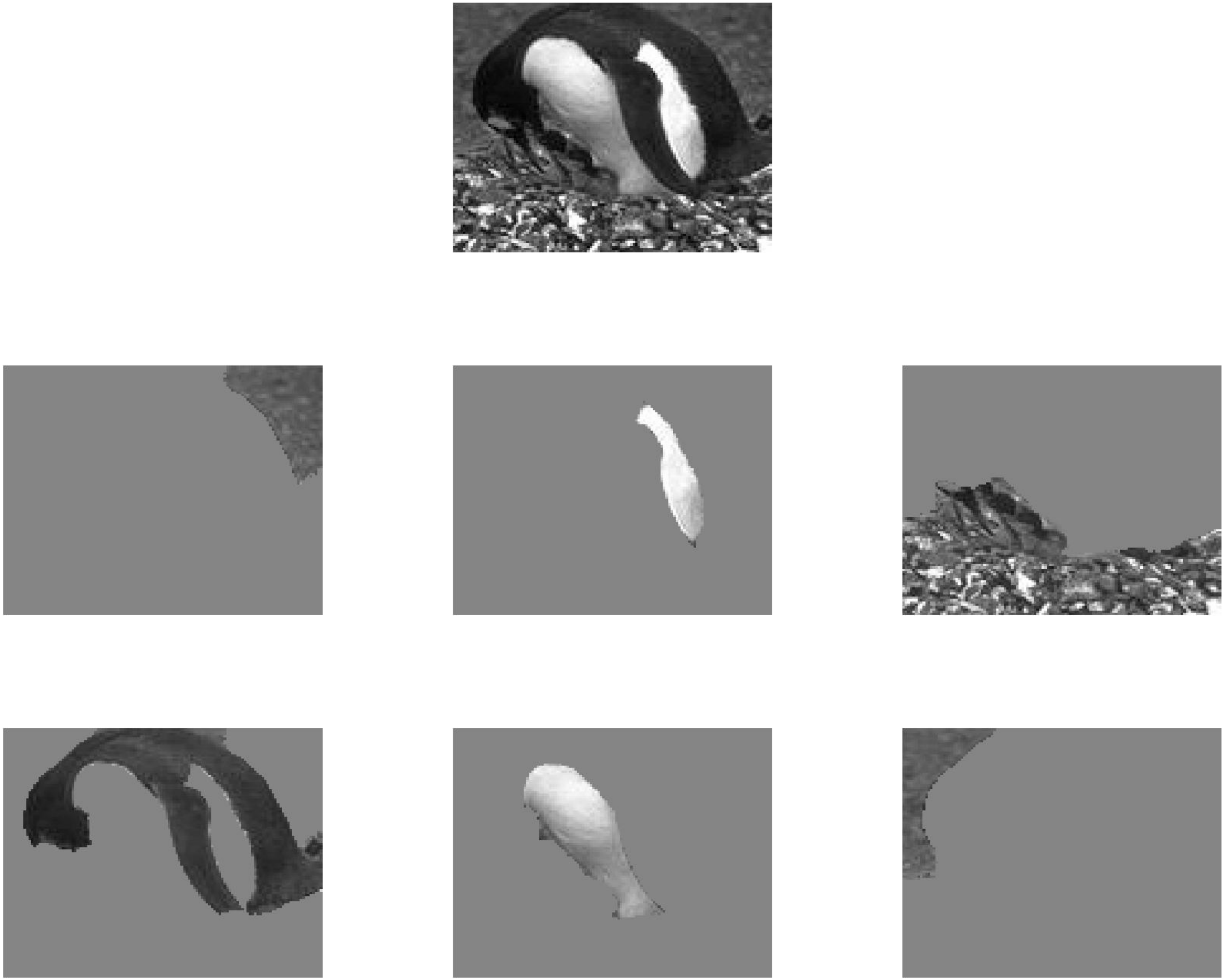
A multiresolution implementation can be used to reduce this running time further on larger images. In our current experiments, with this implementation, the running time on a 300×400 image can be reduced to about 20 seconds on Intel Pentium 300MHz machines. Furthermore, the bottleneck of the computation, a sparse matrix-vector

http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Brightness Image Segmentation

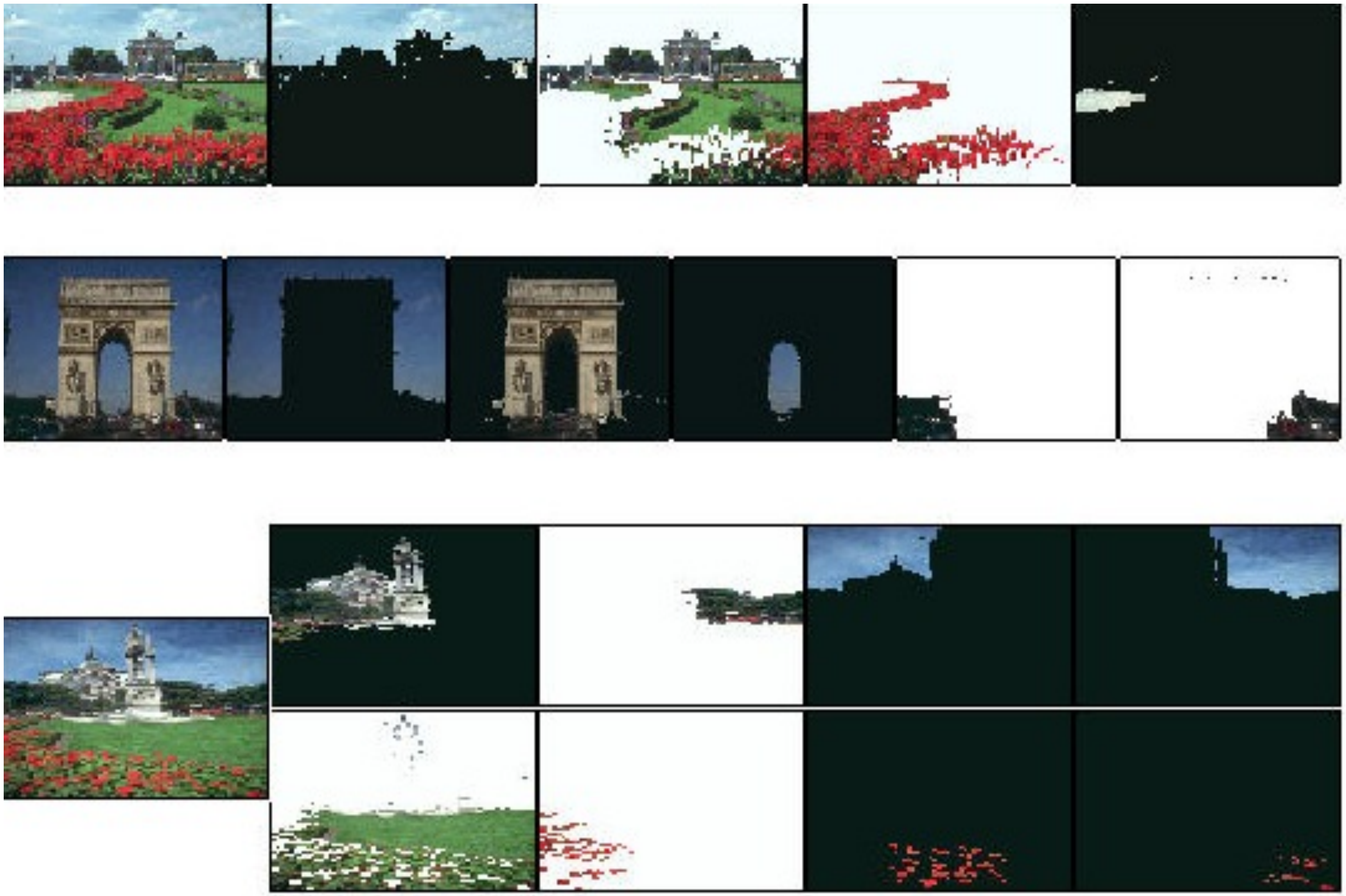


http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf



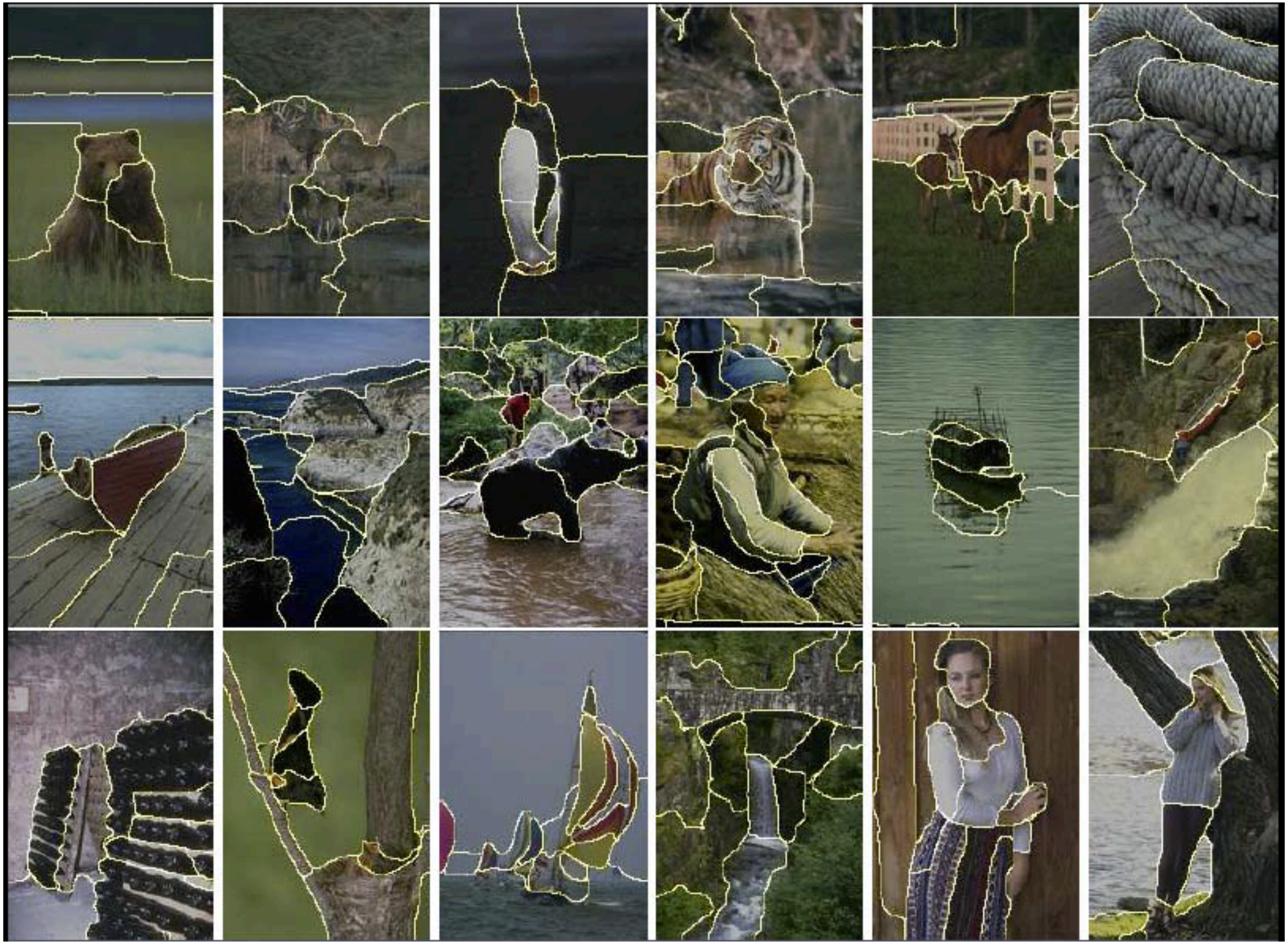
http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Results on color segmentation



http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf

Do we need recognition to take the next step in performance?



Contains a large dataset of images with human "ground truth" labeling.