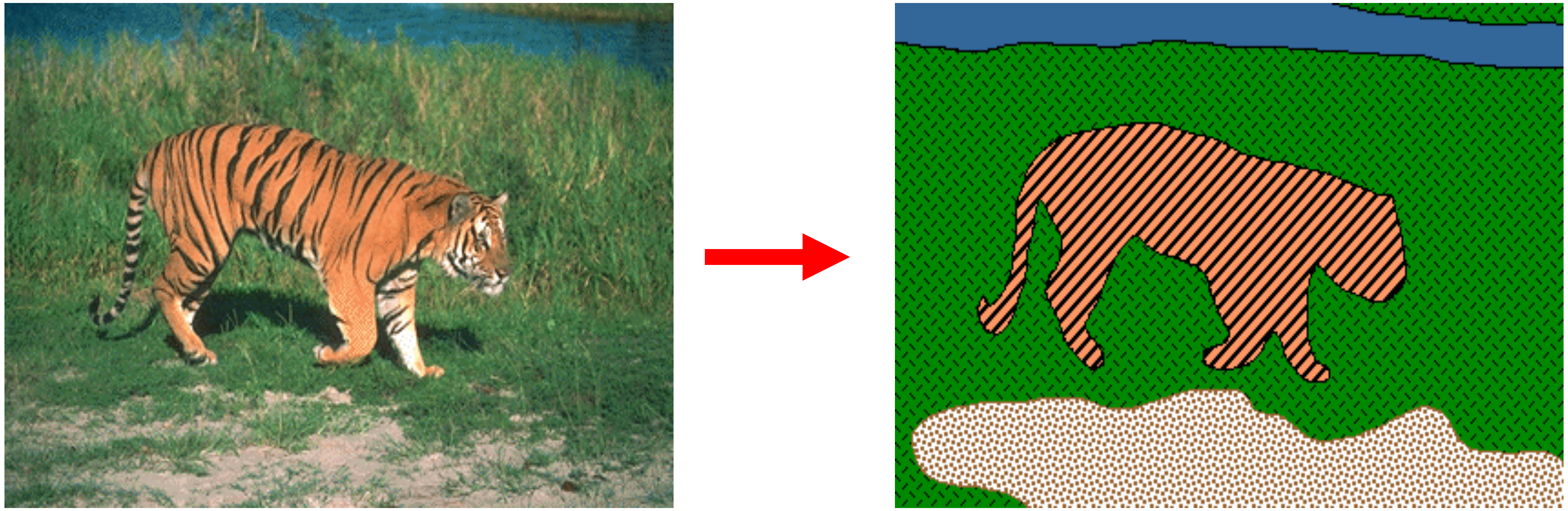
BIL 717
Image Processing
Feb. 22, 2016

Image Segmentation
Boundary Detection

Erkut Erdem
Hacettepe University
Computer Vision Lab (HUCVL)

Image segmentation

- Goal: identify groups of pixels that go together

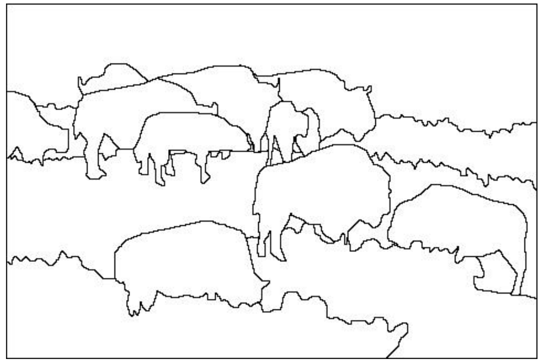


Slide credit: S. Seitz, K. Grauman

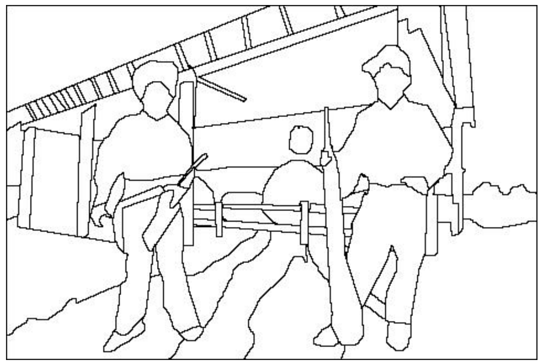
The goals of segmentation

- Separate image into coherent "objects"

image



human segmentation



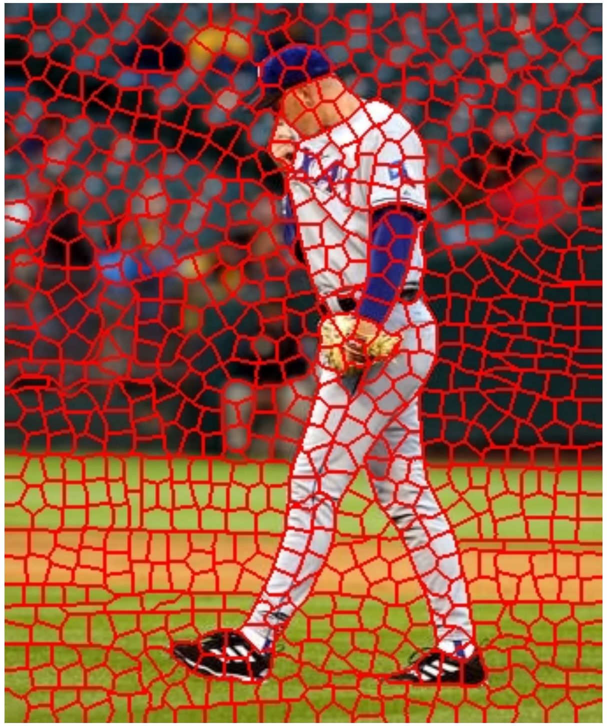
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Slide credit: S. Lazebnik

The goals of segmentation

- Separate image into coherent "objects"
- Group together similar-looking pixels for efficiency of further processing

"superpixels"



X. Ren and J. Malik. [Learning a classification model for segmentation.] ICCV 2003.

Slide credit: S. Lazebnik

Segmentation

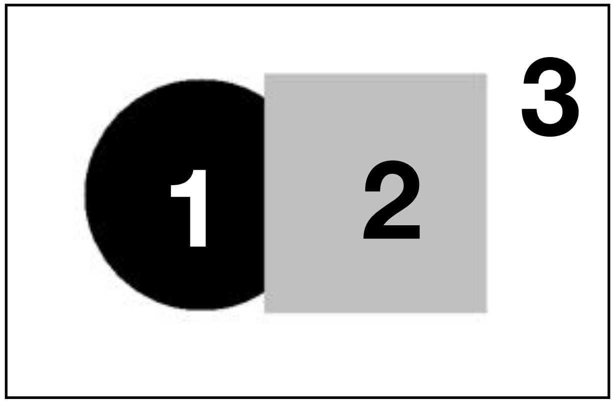
- Compact representation for image data in terms of a set of components
- Components share “common” visual properties
- Properties can be defined at different level of abstractions

Slide credit: Fei-Fei Li

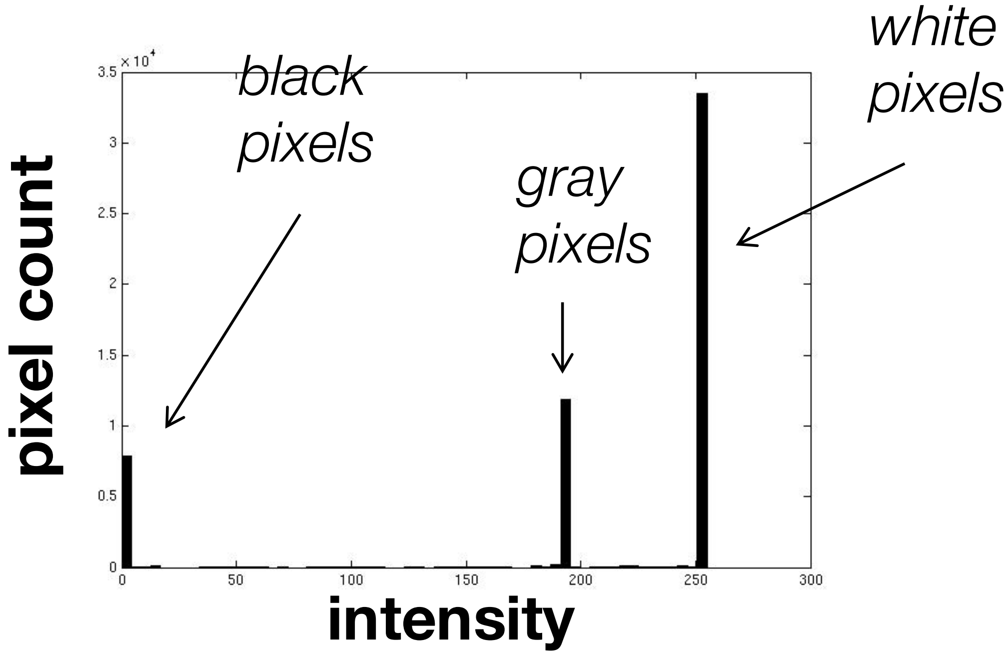
Segmentation methods

- K-means clustering
- Graph-theoretic segmentation
- Boundary Detection

Image segmentation: toy example

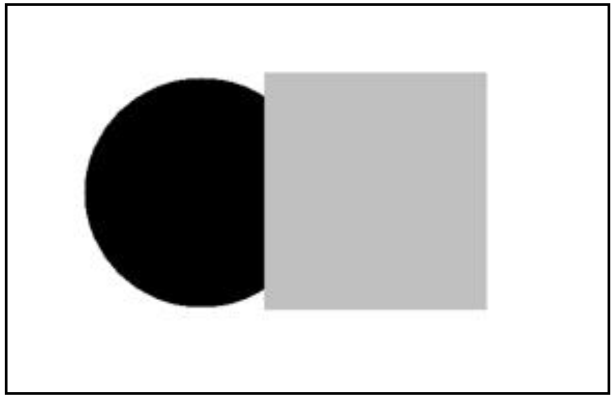


input image

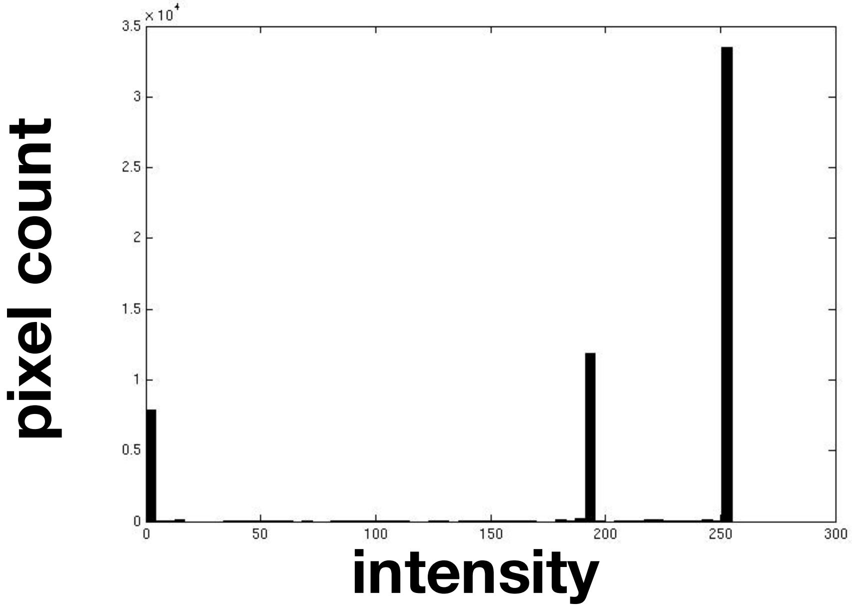


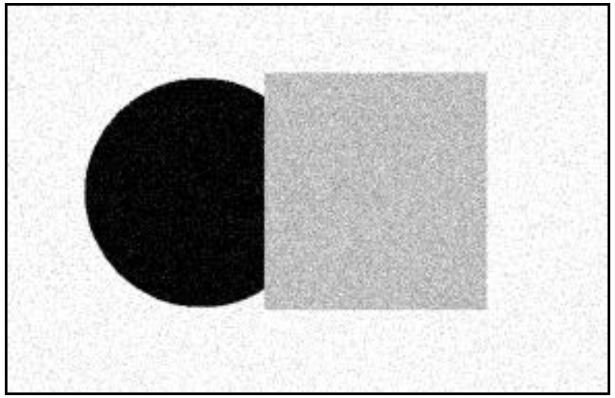
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Slide credit: K. Grauman

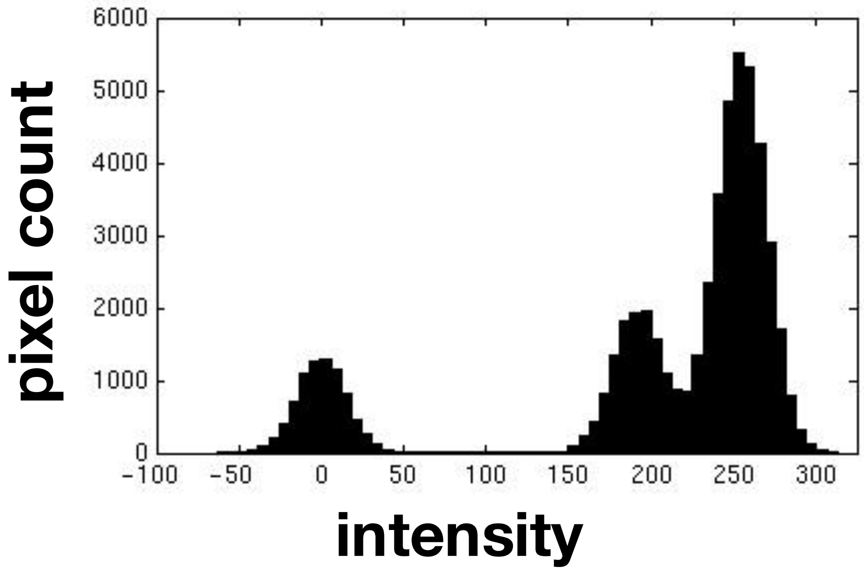


input image

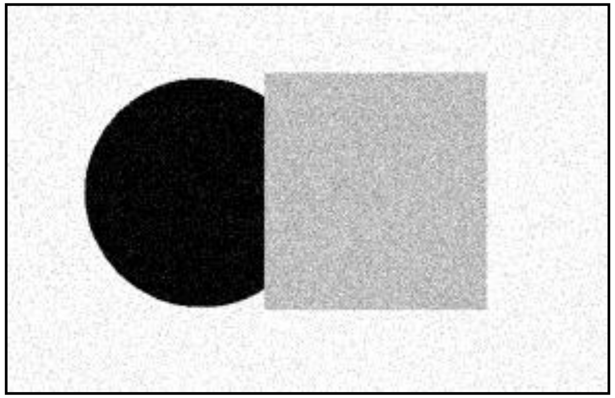




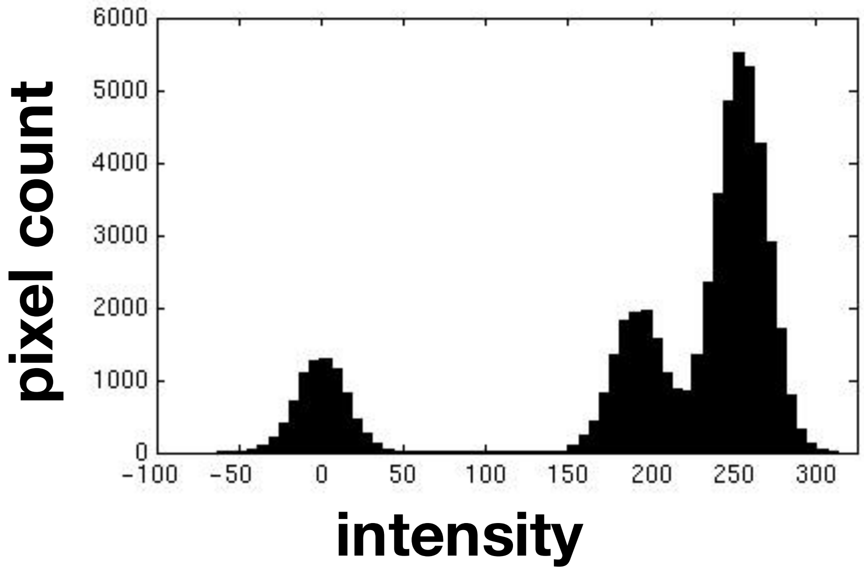
input image



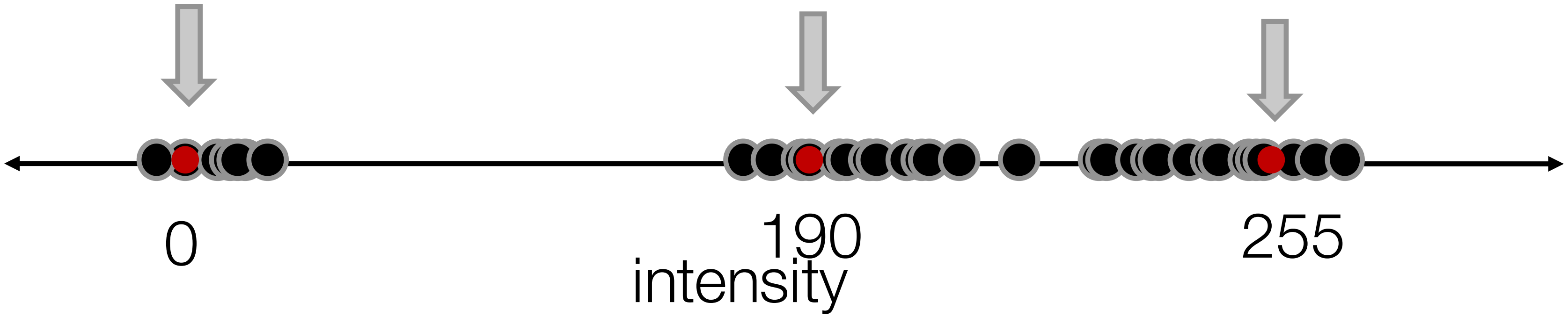
Slide credit: K. Grauman

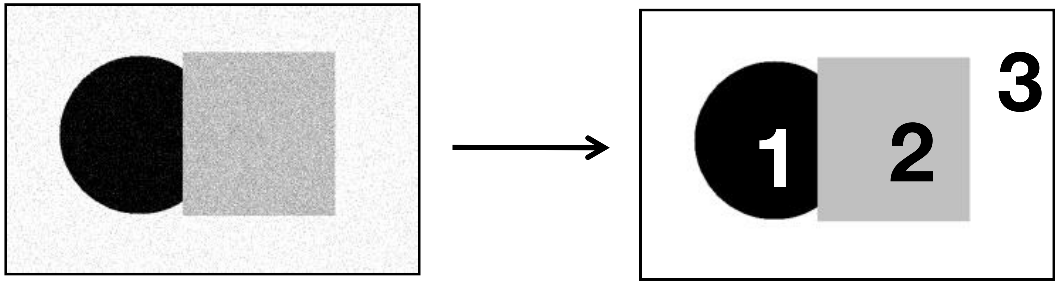


input image



- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***





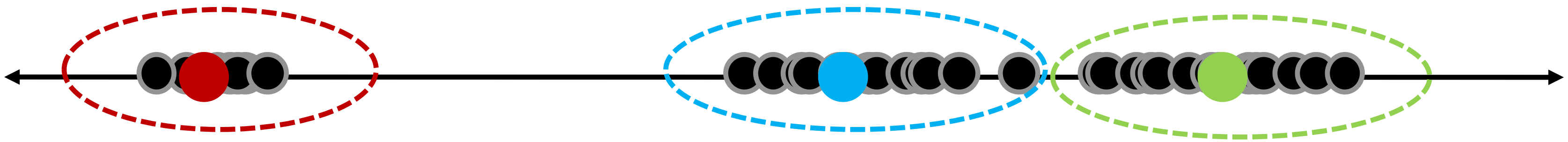
- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center ci:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

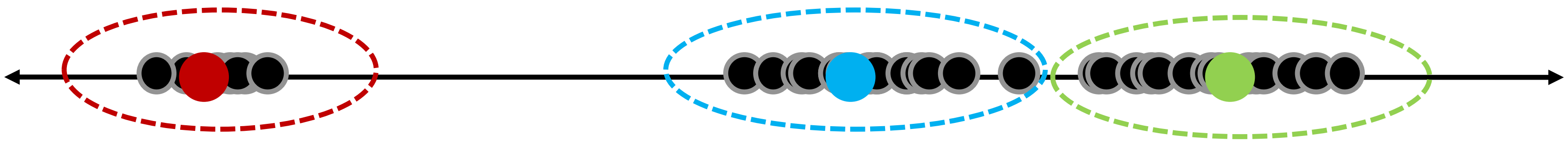
Slide credit: K. Grauman

Clustering

- With this objective, it is a “chicken and egg” problem:
  - If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



  - If we knew the **group memberships**, we could get the centers by computing the mean per group.



Slide credit: K. Grauman

Common similarity/distance measures

- P-norms
  - City Block (L1)
  - Euclidean (L2)
  - L-infinity

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

$$\|\mathbf{x}\|_\infty := \max\left(|x_1|, \ldots, |x_n|\right)$$

Here x_i is the distance btw. two points

- Mahalanobis
  - Scaled Euclidean

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2}}$$

- Cosine distance

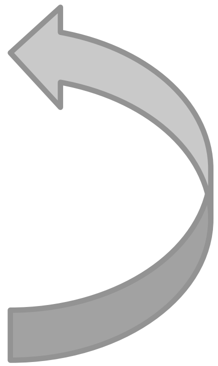
$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Slide credit: D. Hoiem

K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

1. Randomly initialize the cluster centers, $c_1, ..., c_K$
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2



Properties

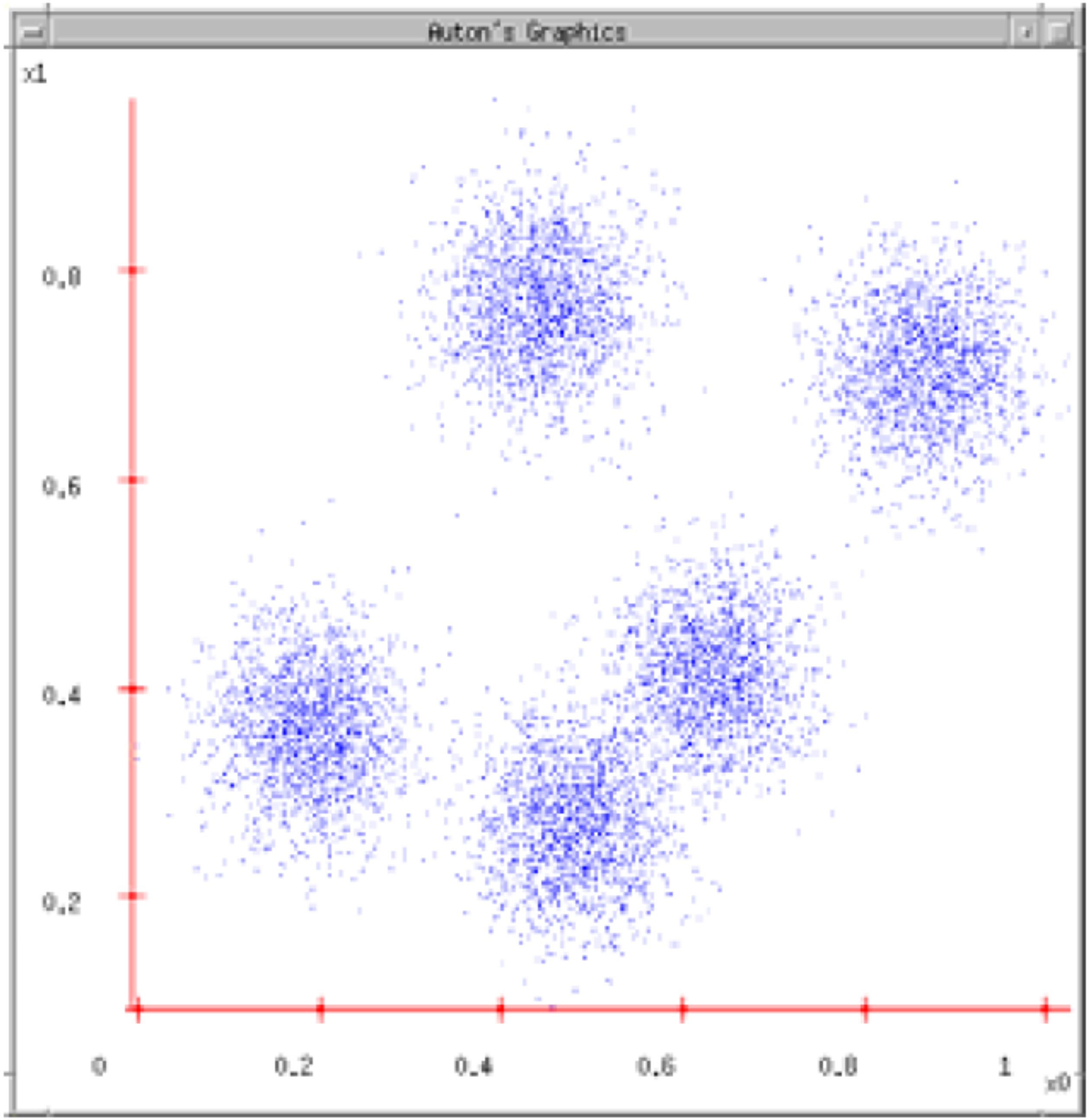
- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Slide credit: S. Seitz

K-means

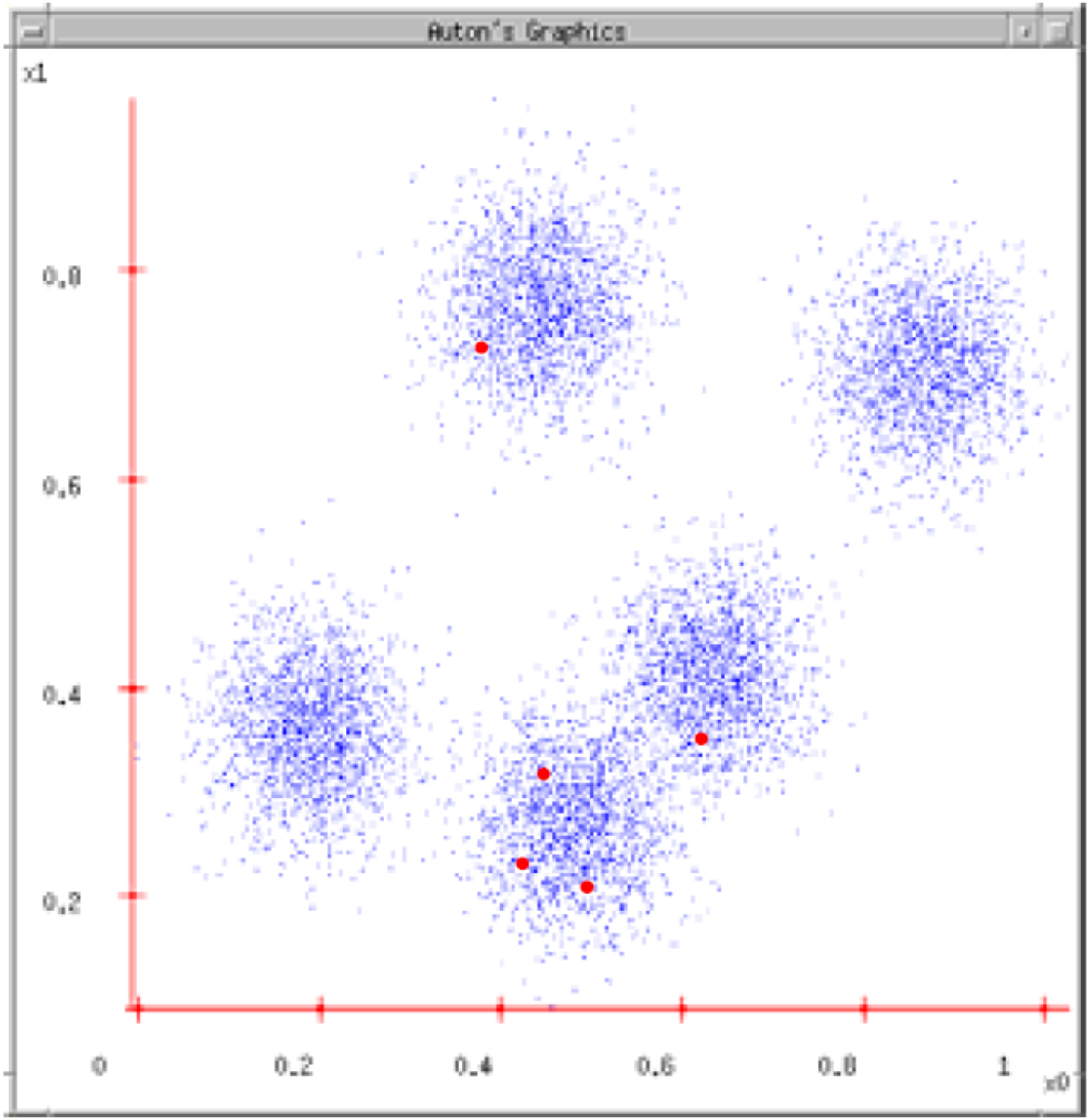
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Slide credit: K Grauman, A. Moore

K-means

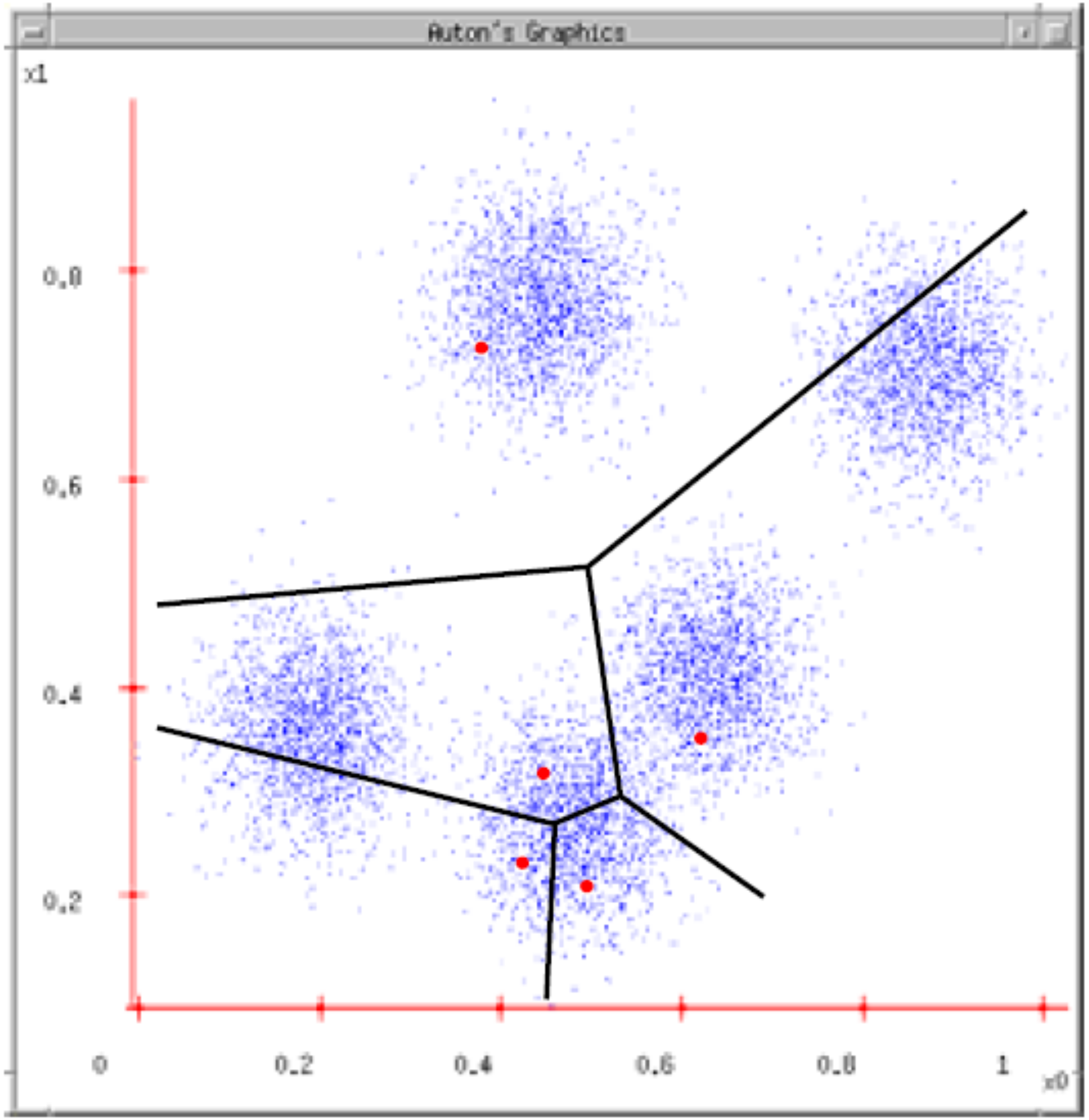
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



Slide credit: K Grauman, A. Moore

K-means

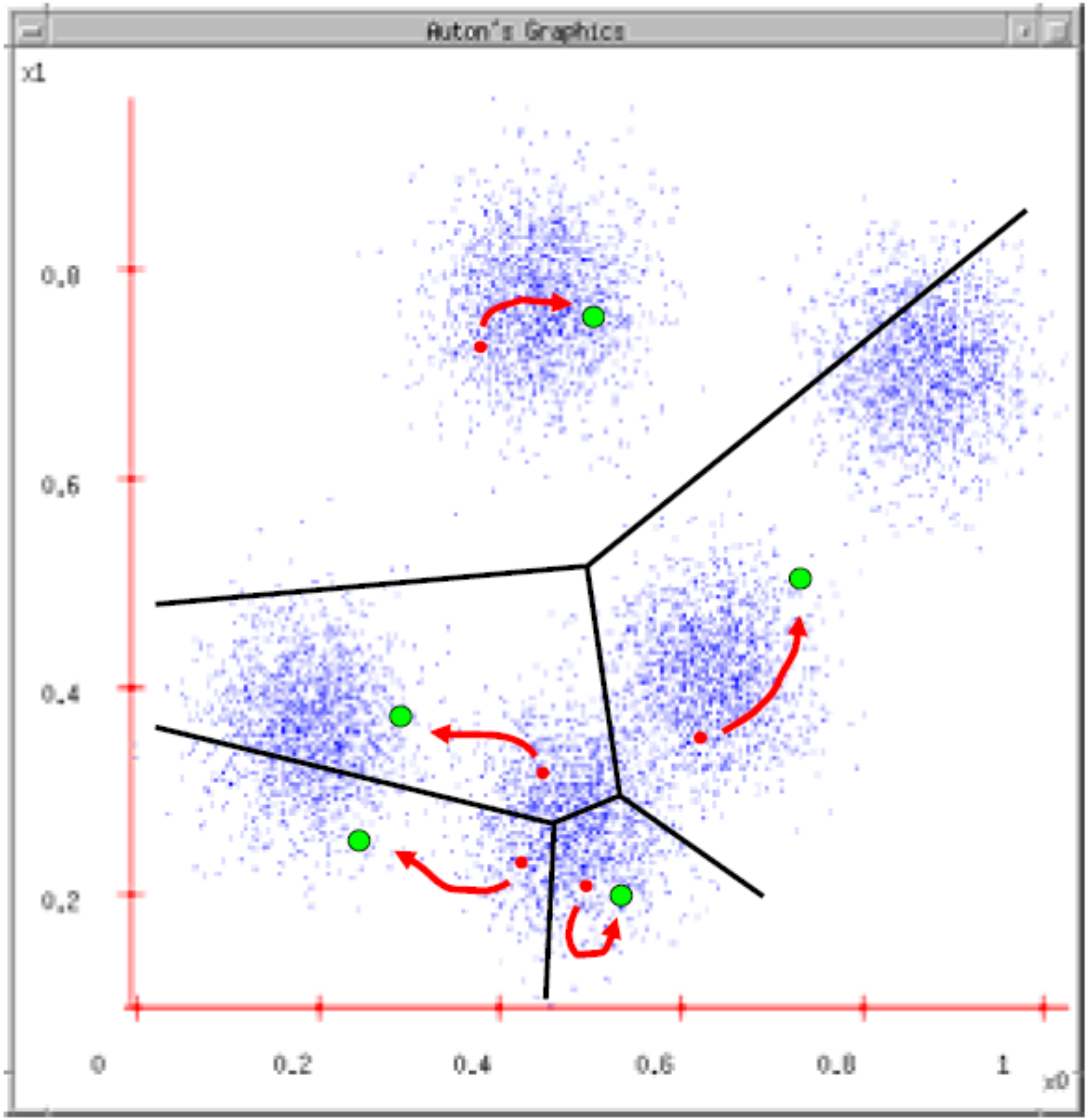
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Slide credit: K Grauman, A. Moore

K-means

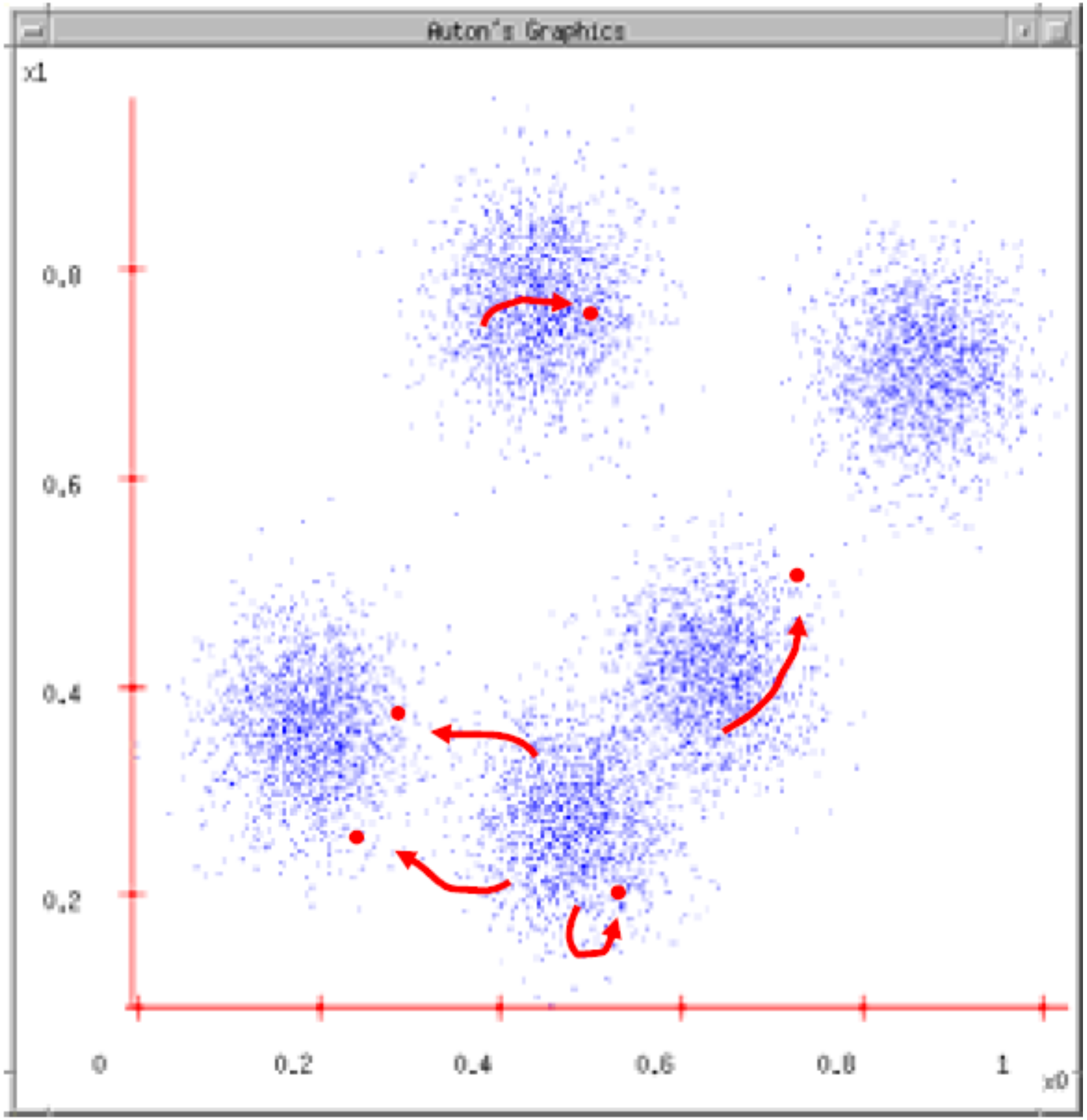
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Slide credit: K Grauman, A. Moore

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Slide credit: K Grauman, A. Moore

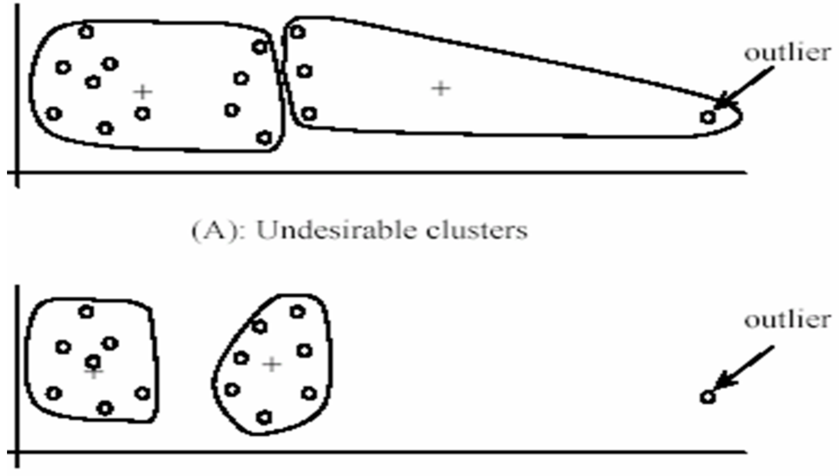
K-means: pros and cons

Pros

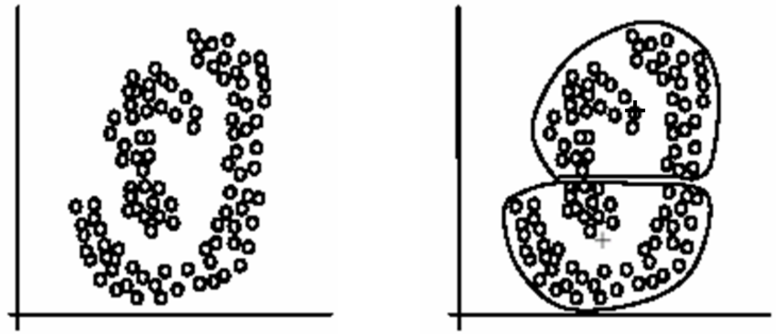
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



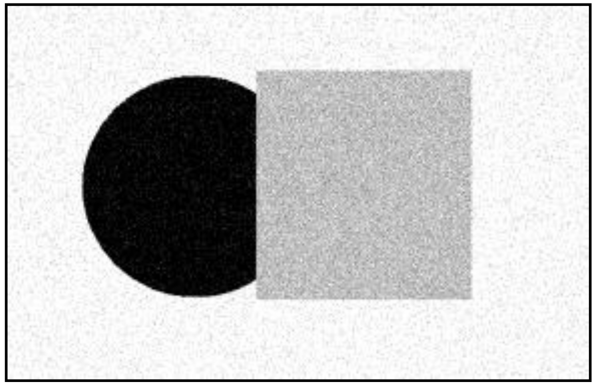
(A): Undesirable clusters
(B): Ideal clusters


(A): Two natural clusters
(B): *k*-means clusters

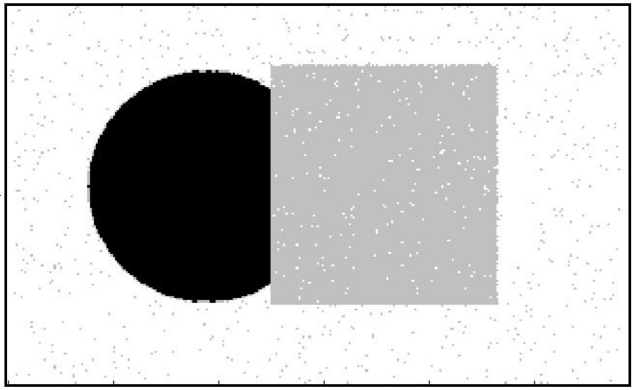
Slide credit: K Grauman

An aside: Smoothing out cluster assignments

- Assigning a cluster label per pixel may yield outliers:



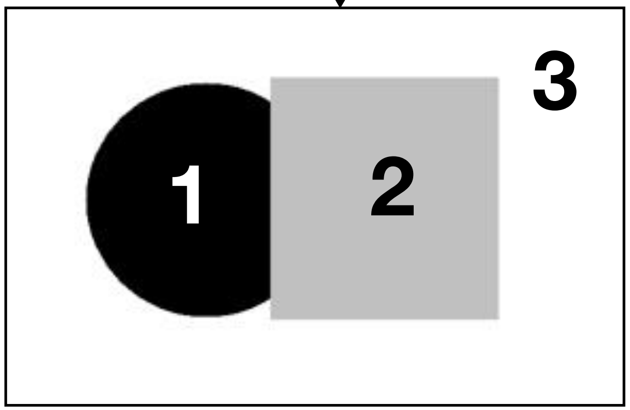
original



labeled by cluster center's intensity

?

- How to ensure they are spatially smooth?



Slide credit: K Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on intensity similarity

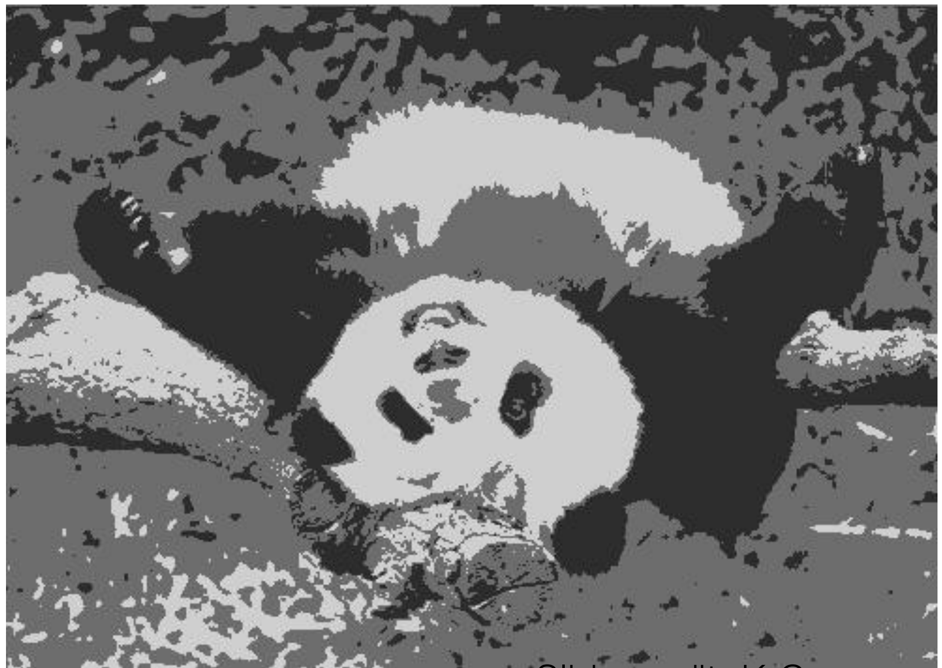
Feature space: intensity value (1-d)

Slide credit: K Grauman

K=2

K=3

quantization of the feature space; segmentation label map

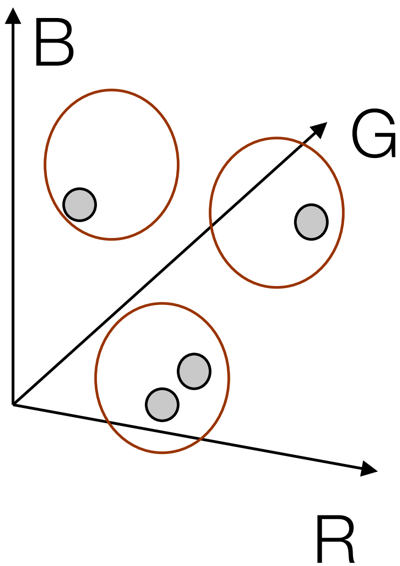


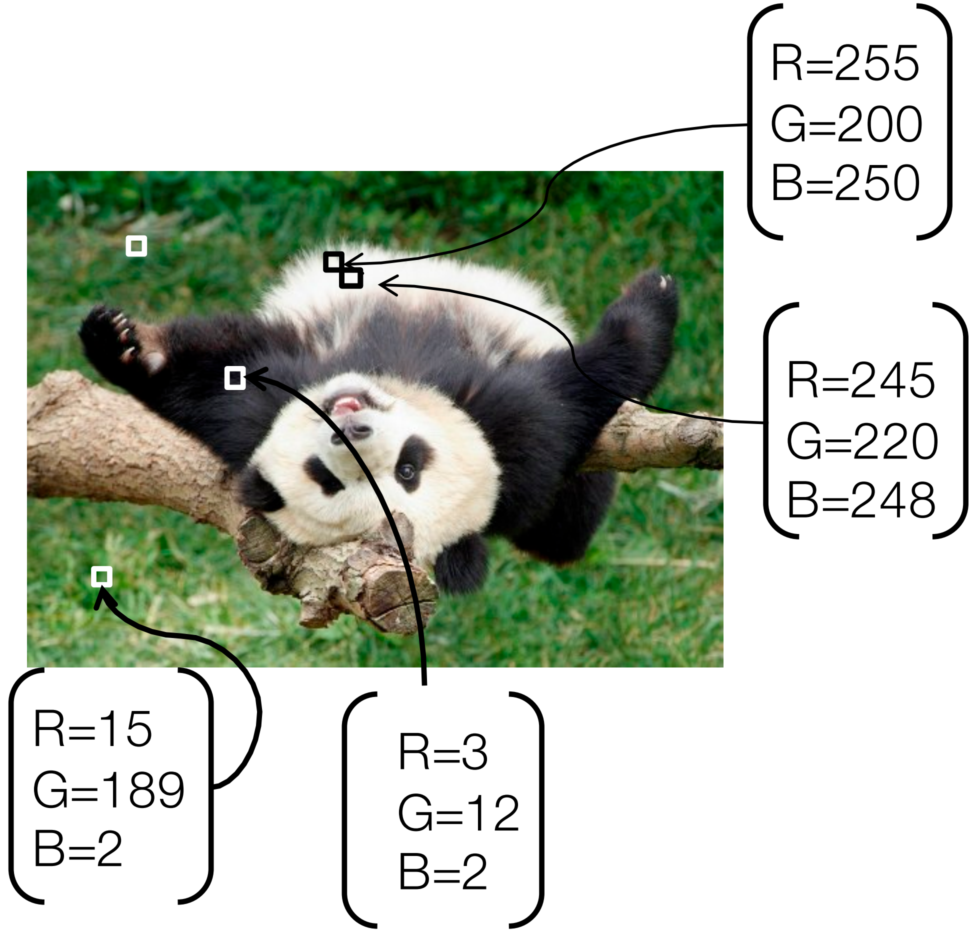
Slide credit: K Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on <u>color</u> similarity





Feature space: color value (3-d)

Slide credit: K Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

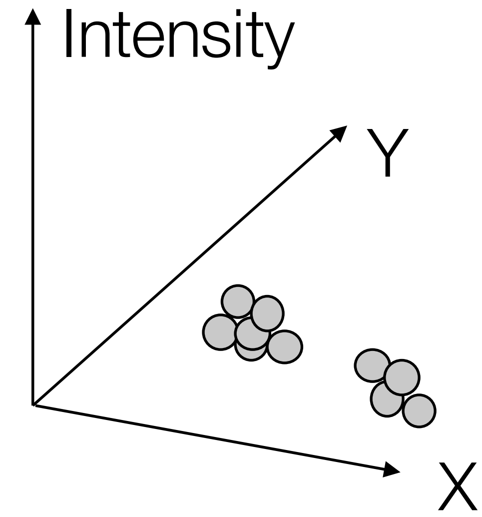
Clusters based on intensity similarity don’t have to be spatially coherent.

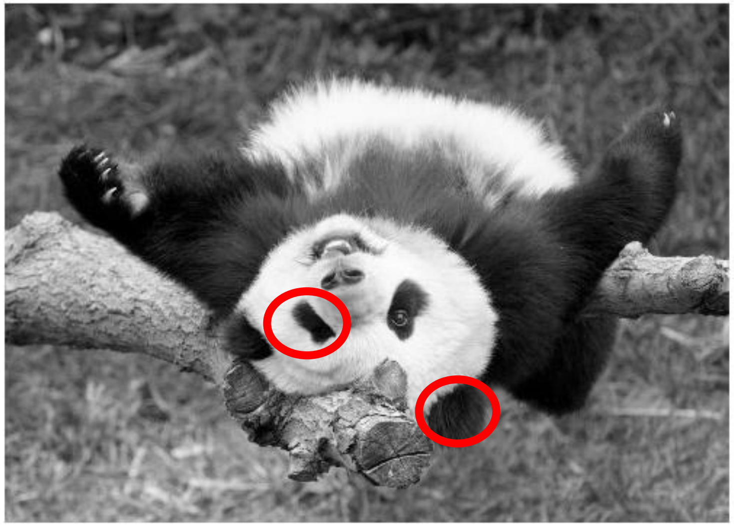
Slide credit: K Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on <u>intensity+position</u> similarity



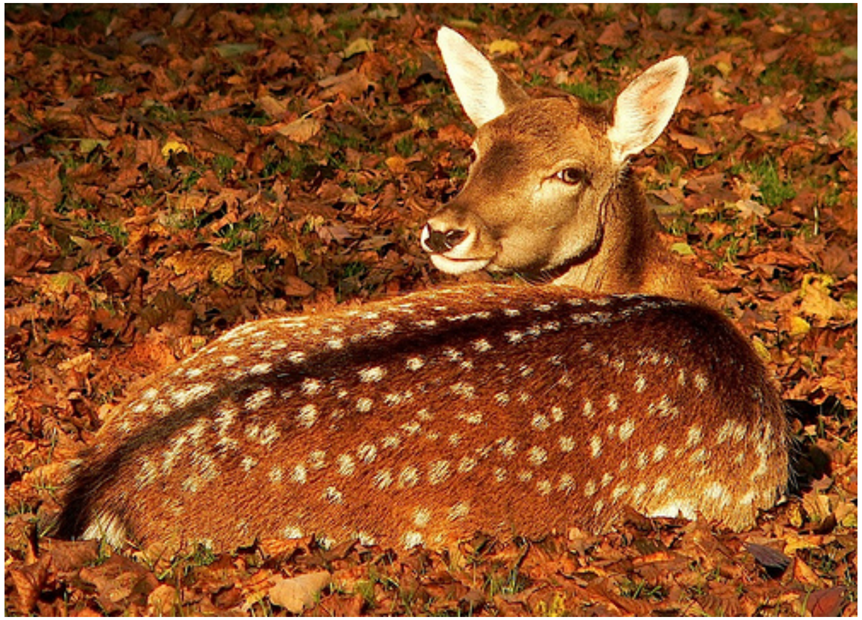


Both regions are black, but if we also include <u>position (x,y)</u>**,** then we could group the two into distinct segments; way to encode both similarity & proximity.

Slide credit: K Grauman

Segmentation as clustering

- Color, brightness, position alone are not enough to distinguish all regions…

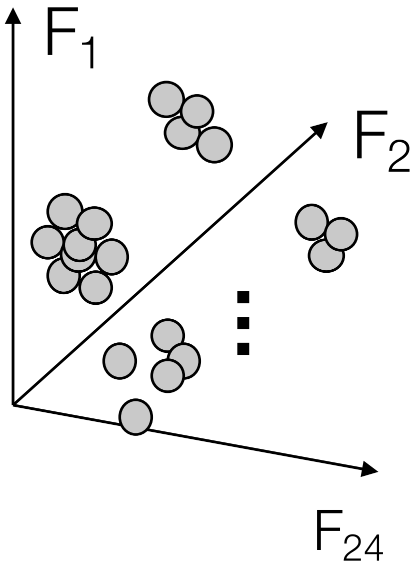


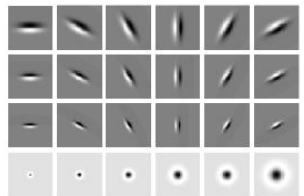
Slide credit: K Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on <u>texture</u> similarity





Filter bank of 24 filters

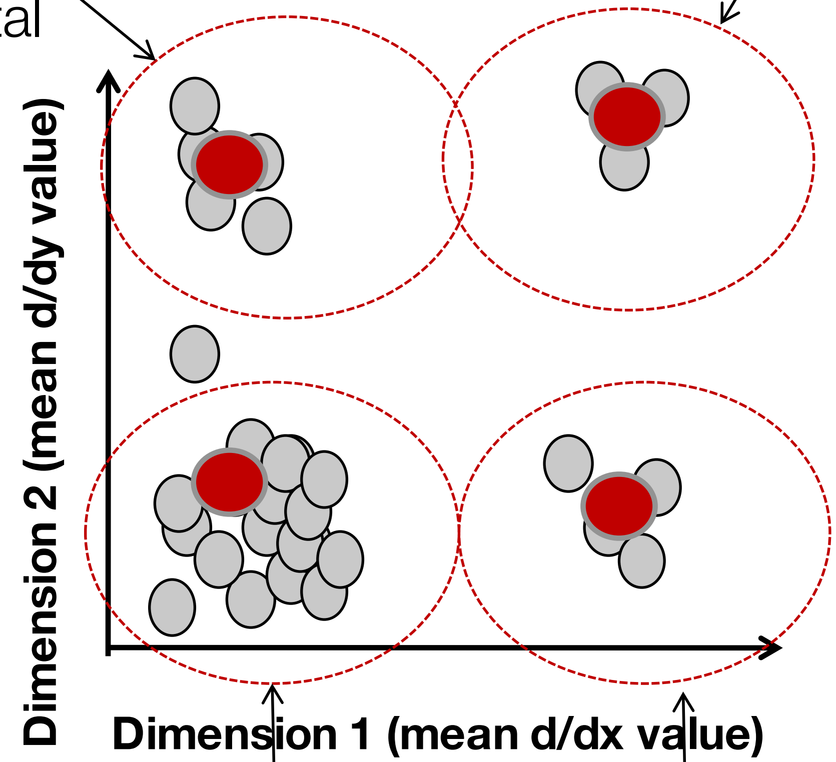
Feature space: filter bank responses (e.g., 24-d)

Slide credit: K Grauman

Texture representation example

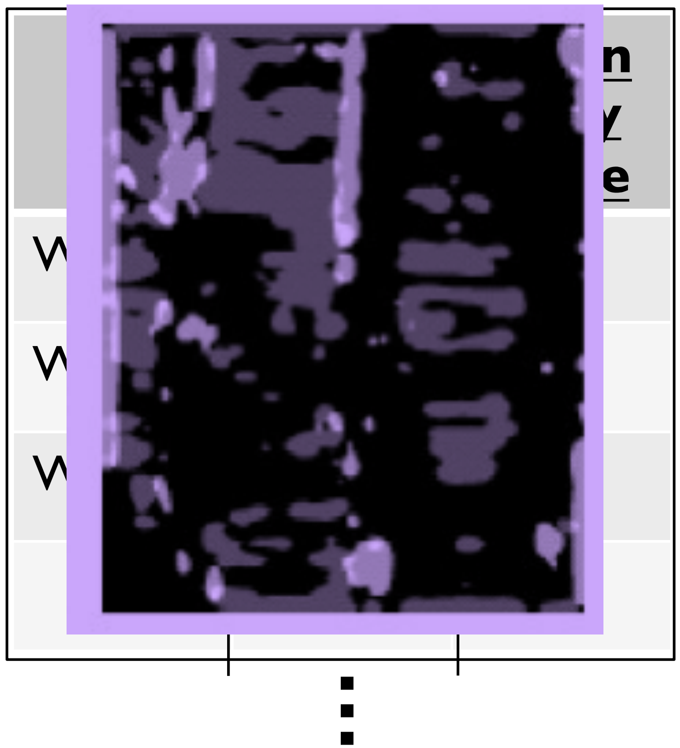
Windows with primarily horizontal edges

Both



Windows with small gradient in both directions

Windows with primarily vertical edges



statistics to summarize patterns in small windows

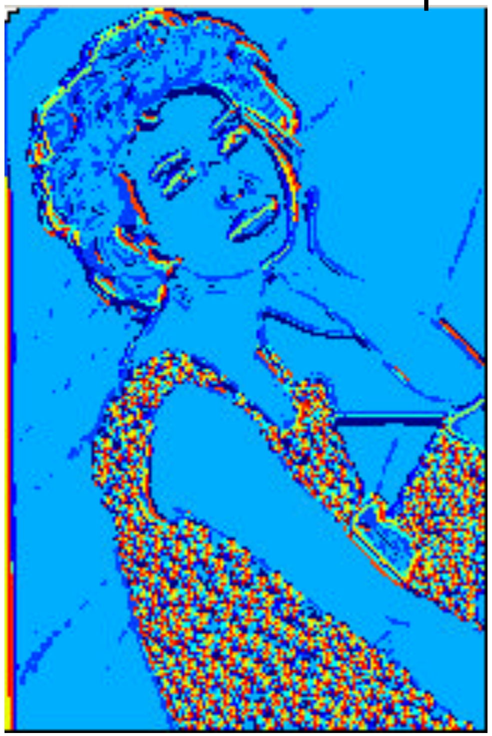
Slide credit: K Grauman

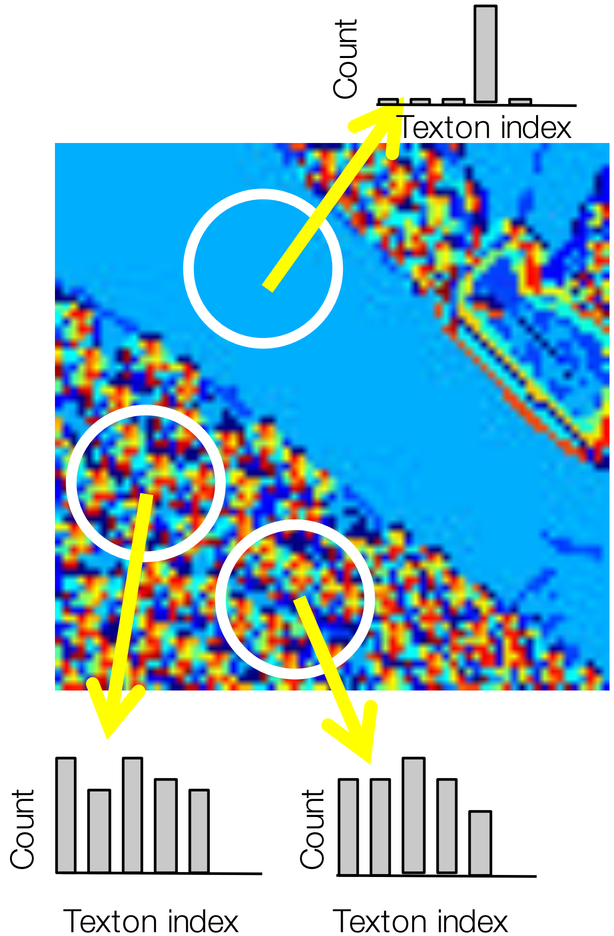
Segmentation with texture features

- Find "textons" by **clustering** vectors of filter bank outputs
- Describe texture in a window based on *texton histogram*

Image

Texton map

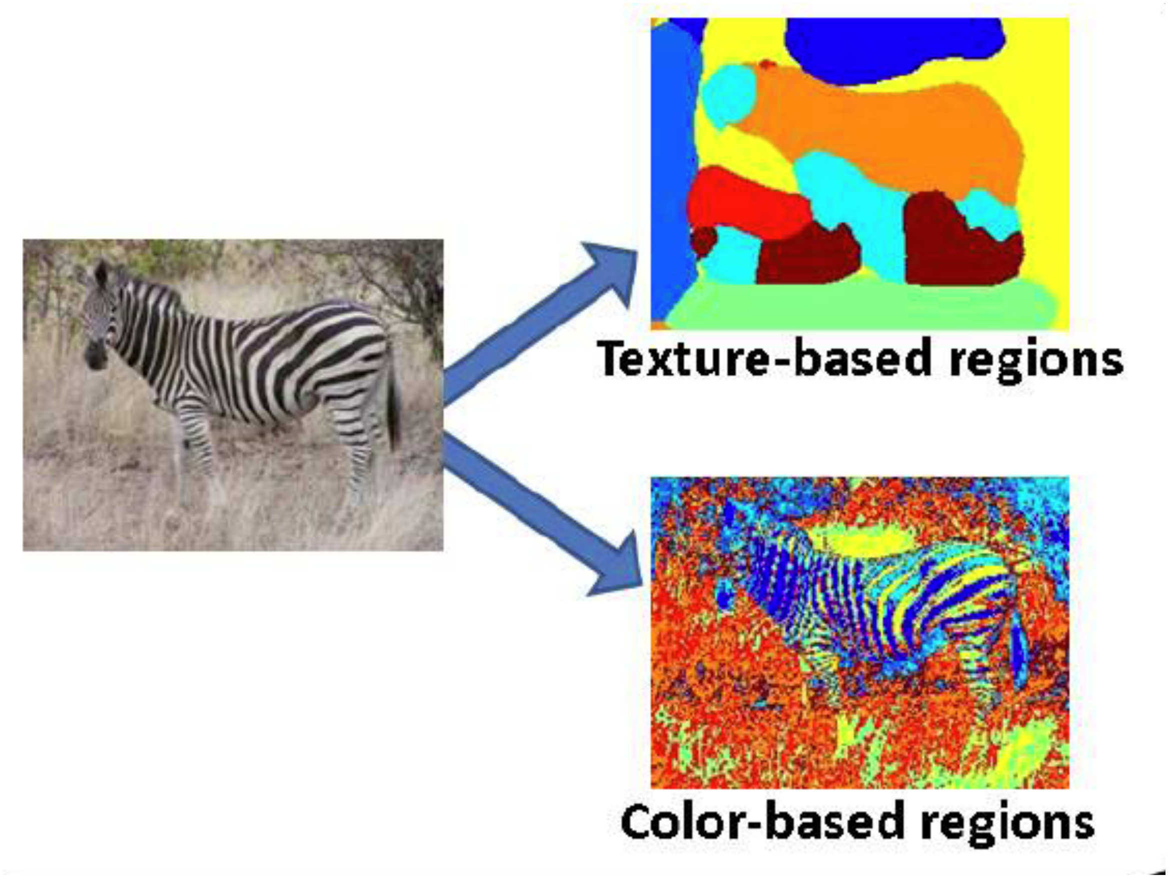




Malik, Belongie, Leung and Shi. IJCV 2001.

Slide credit: K Grauman, L. Lazebnik

Image segmentation example



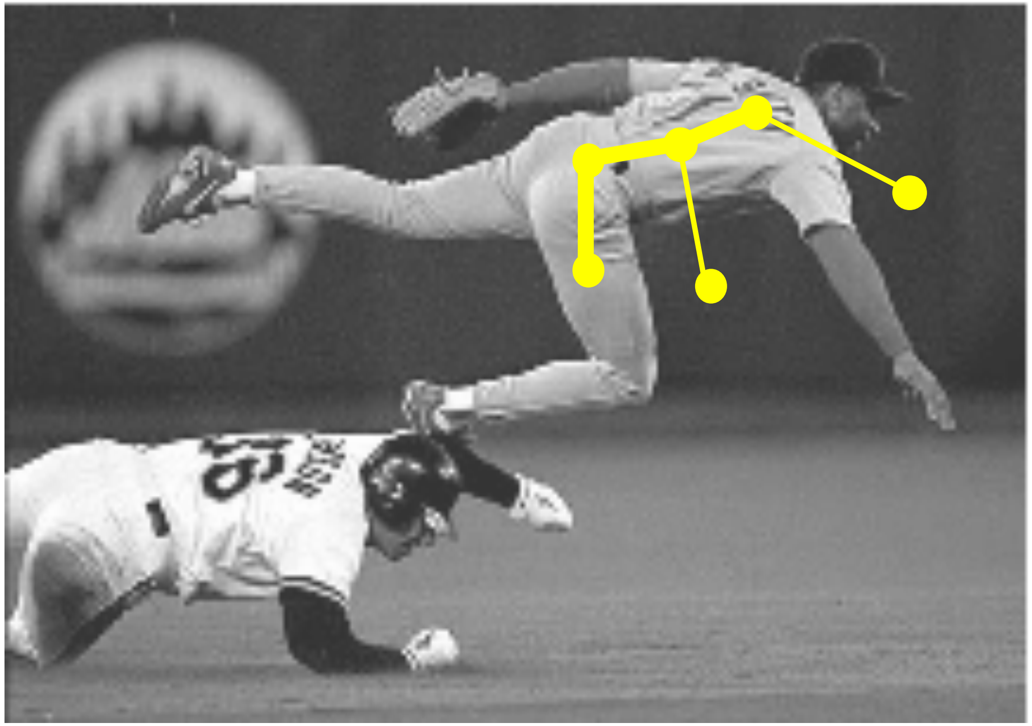
Slide credit: K Grauman

Segmentation methods

- K-means clustering
- Graph-theoretic segmentation
- Boundary Detection

Graph-Theoretic Image Segmentation

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



V: image pixels

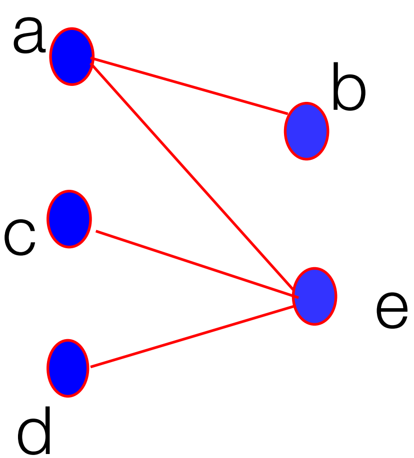
E: connections between pairs of nearby pixels

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

Segmentation = graph partition

Slide credit: B. Freeman and A. Torralba

Graphs Representations



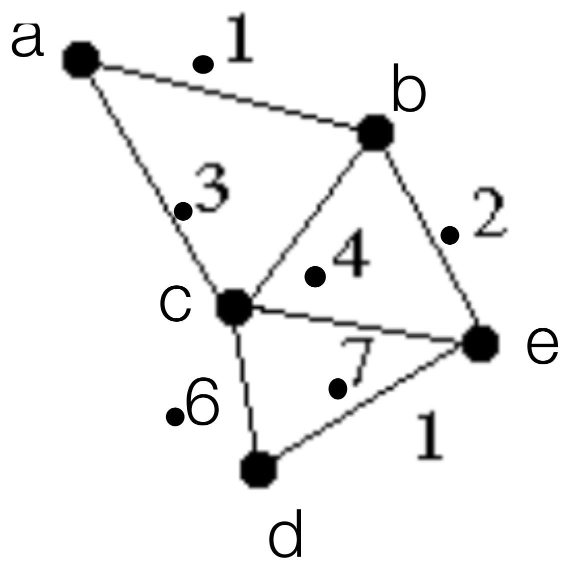
	a	b	c	d	e
a	0	1	0	0	1
b	1	0	0	0	0
c	0	0	0	0	1
d	0	0	0	0	1
e	1	0	1	1	0

Adjacency Matrix

Slide credit: B. Freeman and A. Torralba

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

A Weighted Graph and its Representation



Affinity Matrix

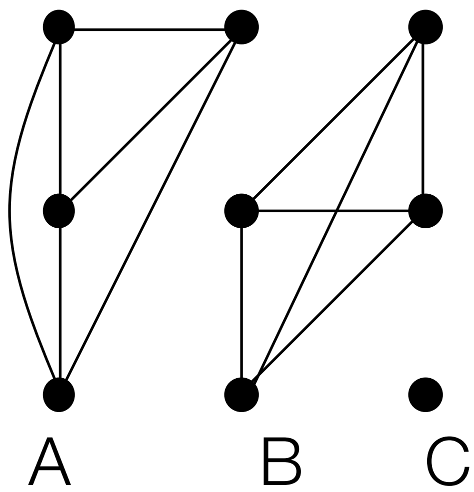
$$W = \begin{bmatrix} 1 & .1 & .3 & 0 & 0 \\ .1 & 1 & .4 & 0 & .2 \\ .3 & .4 & 1 & .6 & .7 \\ 0 & 0 & .6 & 1 & 1 \\ 0 & .2 & .7 & 1 & 1 \end{bmatrix}$$

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

Slide credit: B. Freeman and A. Torralba

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Segmentation by graph partitioning



- Break graph into segments
  - Delete links that cross between segments
  - Easiest to break links that have low affinity
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

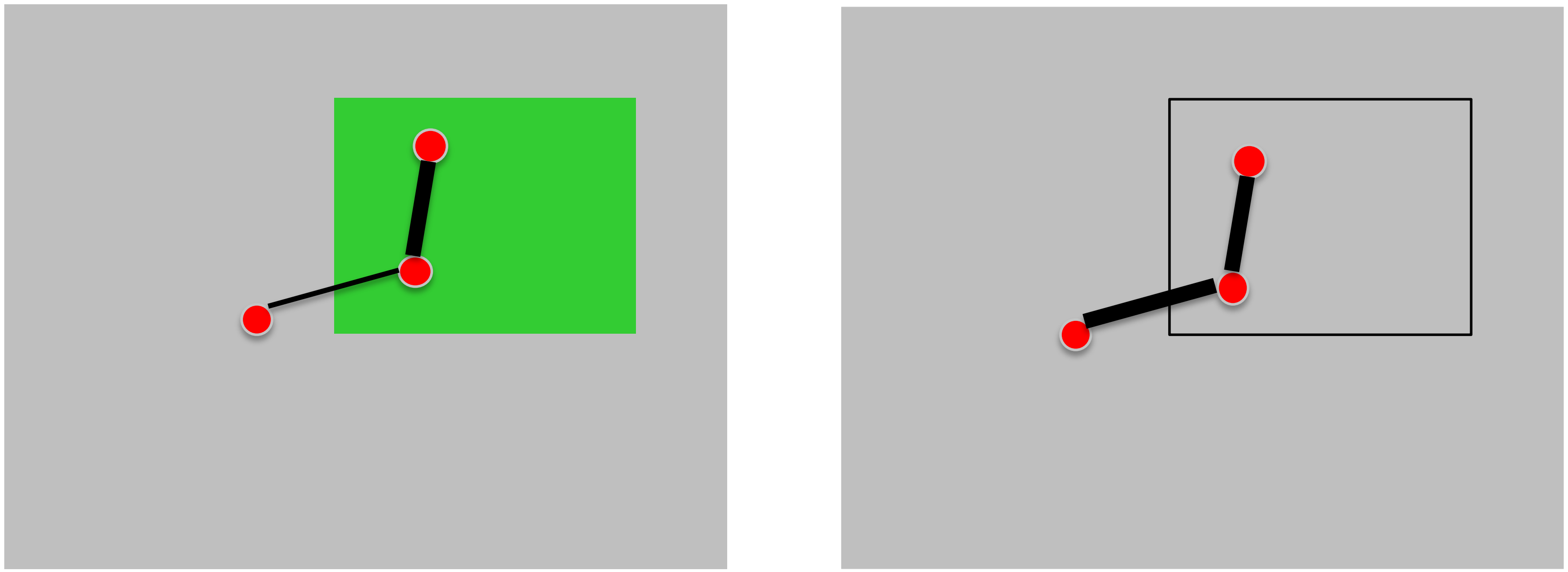
Slide credit: S. Seitz

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



Slide credit: B. Freeman and A. Torralba

Affinity between pixels

Similarities among pixel descriptors

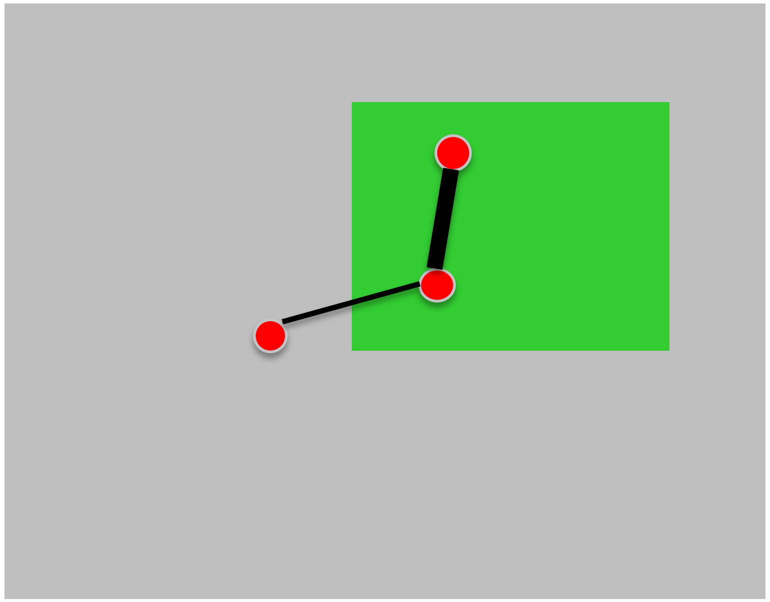
$$W_{ij} = \exp(-|| z_i - z_j ||^2 / \sigma^2)$$

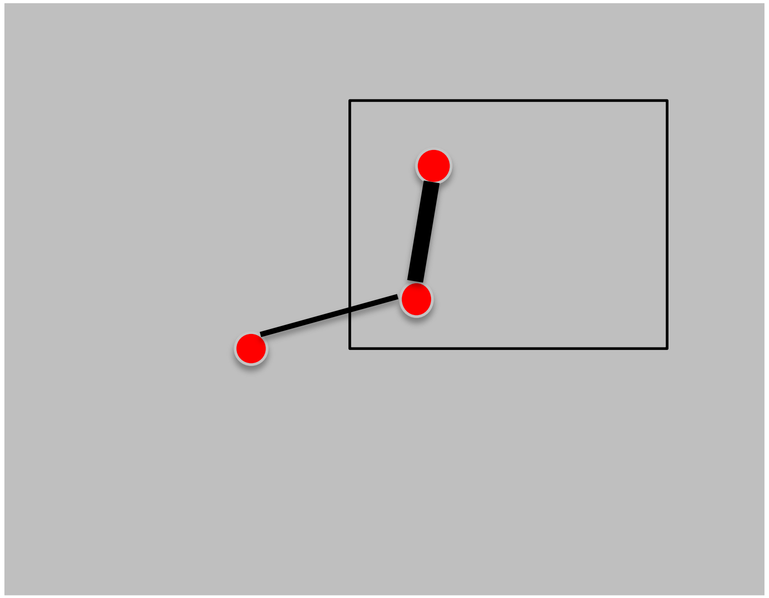
↖ σ = Scale factor… it will hunt us later

Interleaving edges

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

With Pb = probability of boundary

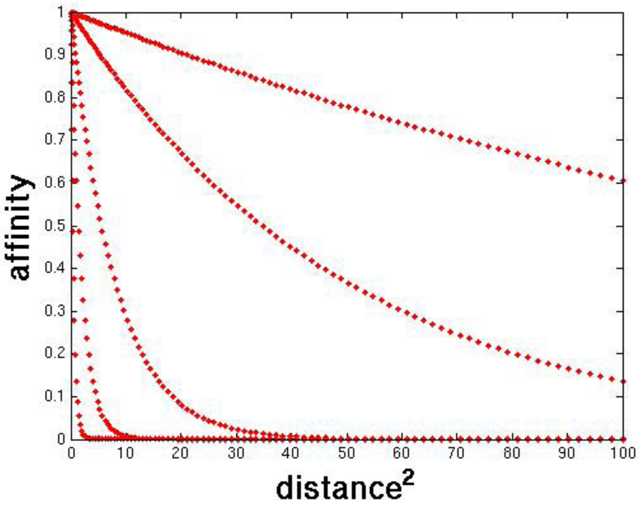


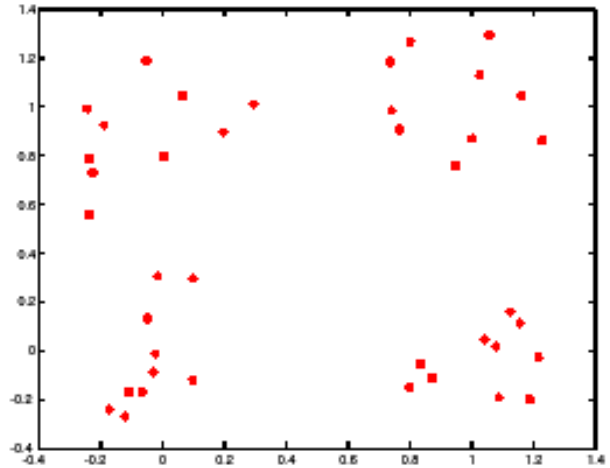


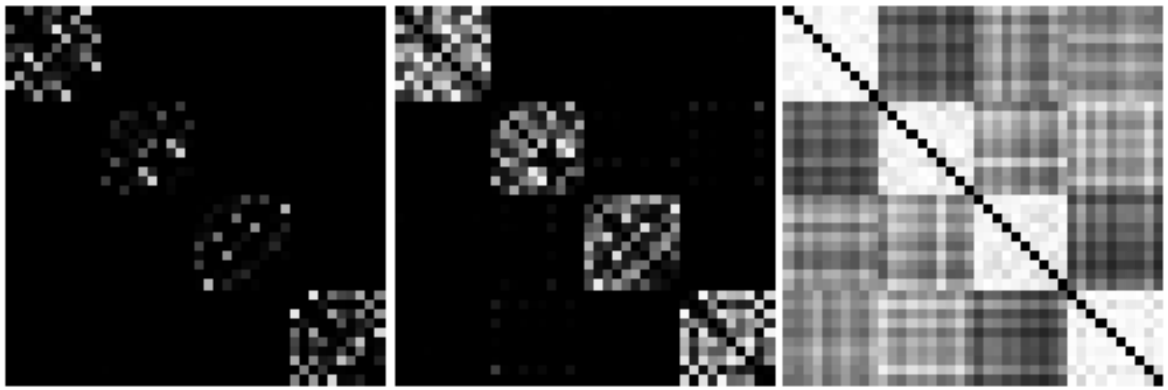
Slide credit: B. Freeman and A. Torralba

Scale affects affinity

- Small σ: group only nearby points
- Large σ: group far-away points







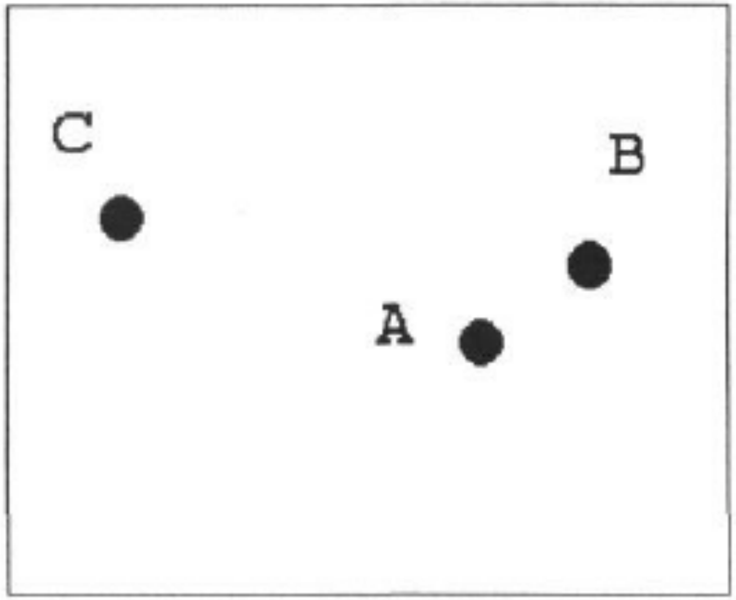
Slide credit: S. Lazebnik

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

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

With an appropriate s

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

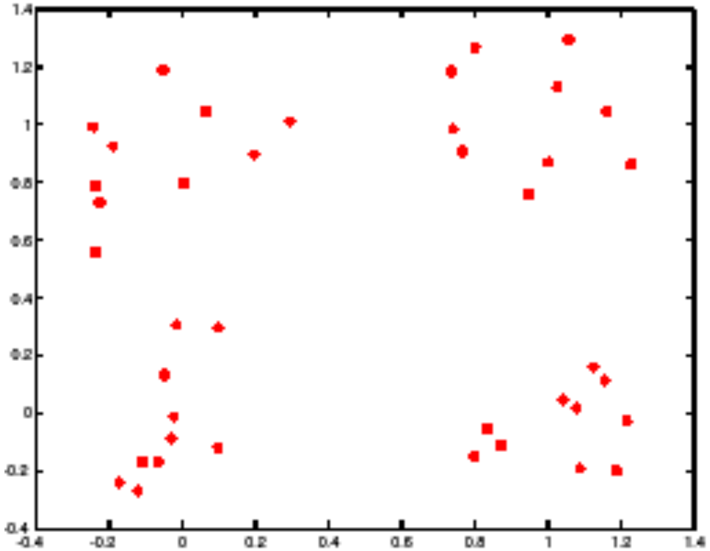
The eigenvectors of W are:

	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

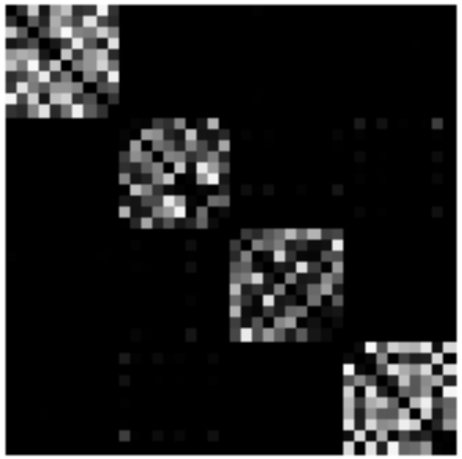
The first 2 eigenvectors group the points as desired…

Slide credit: B. Freeman and A. Torralba

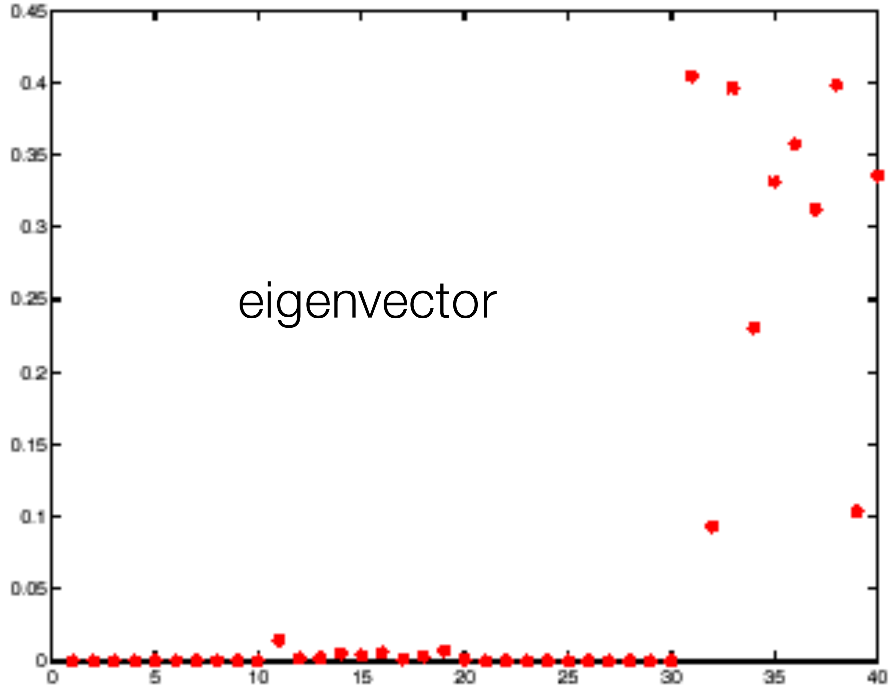
Example eigenvector



points

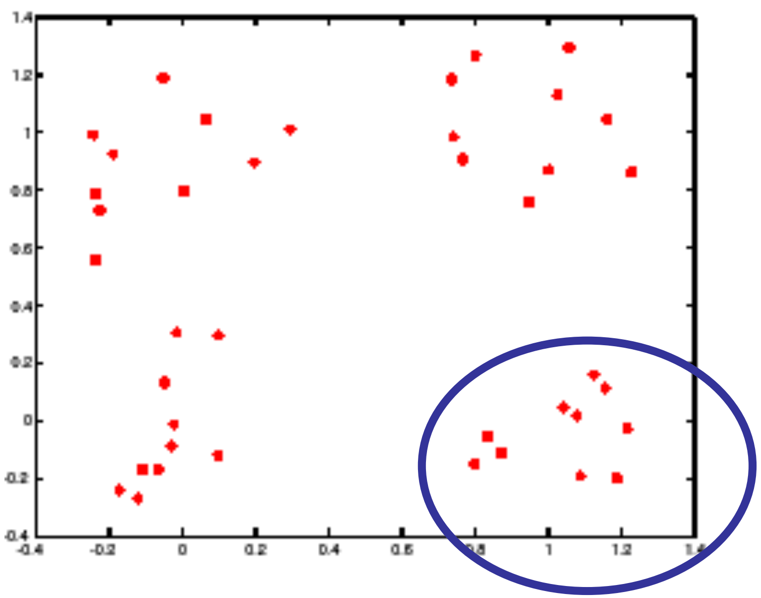


Affinity matrix

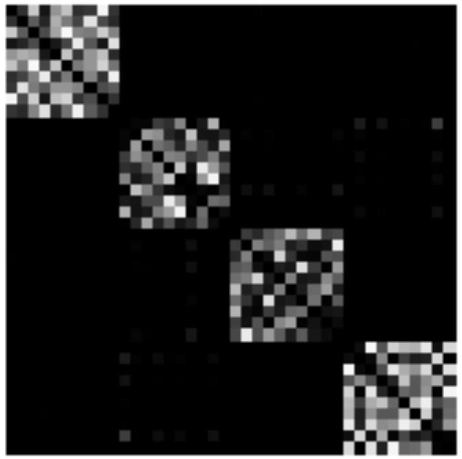


Slide credit: B. Freeman and A. Torralba

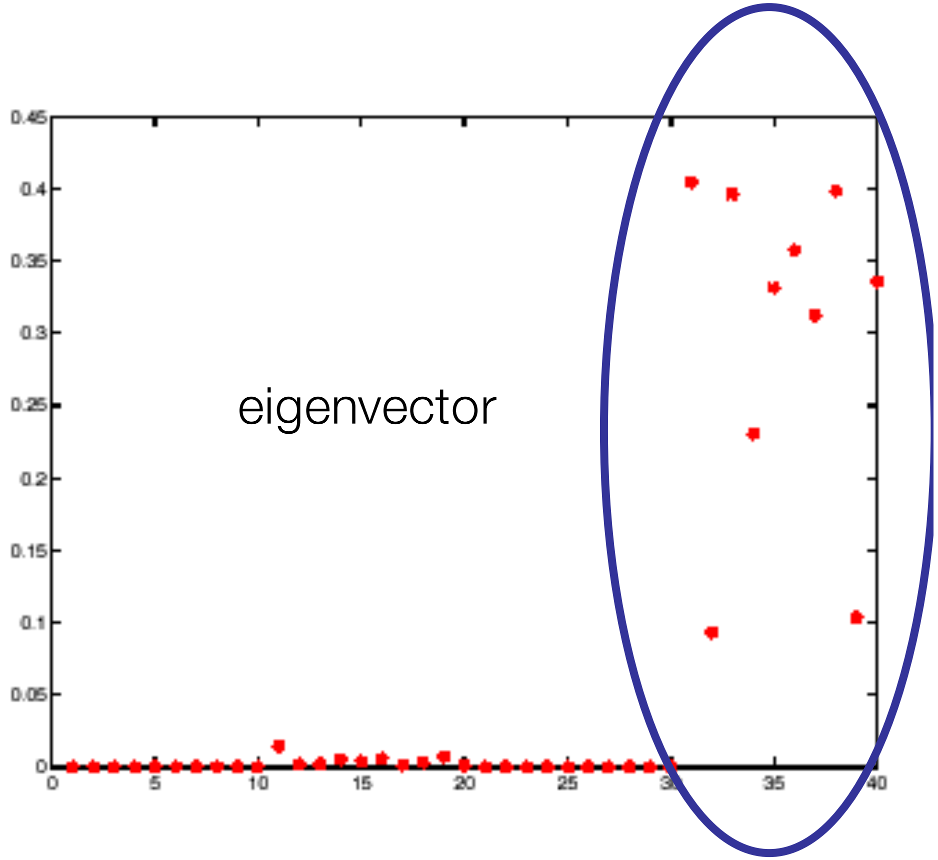
Example eigenvector



points

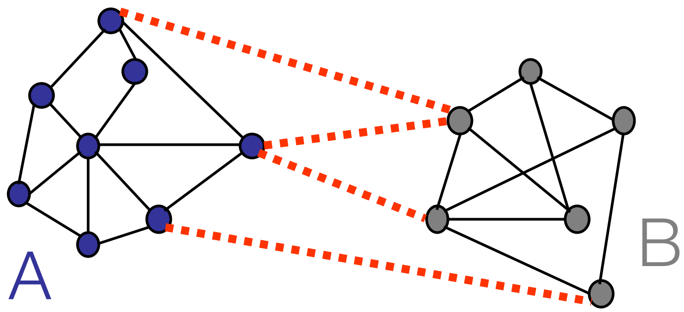


Affinity matrix



Slide credit: B. Freeman and A. Torralba

Graph cut

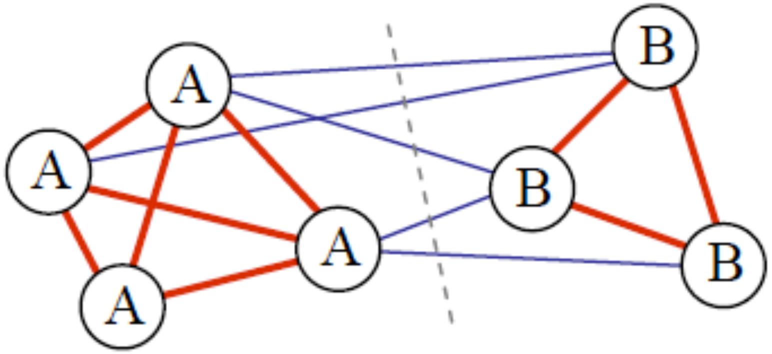


- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

Slide credit: S. Seitz

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:

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

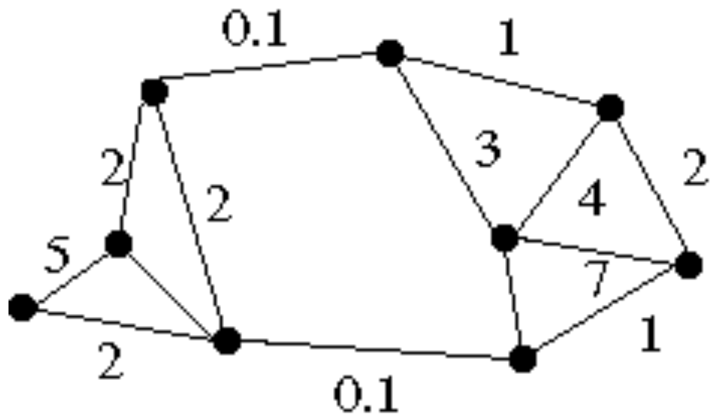
with $\mathrm{A} \cap \mathrm{B} = \varnothing$

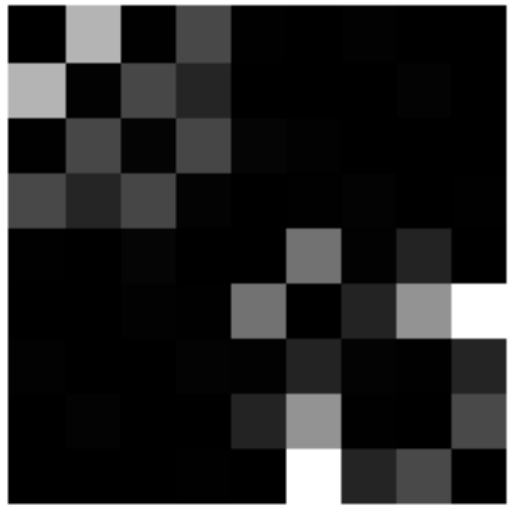
Slide credit: B. Freeman and A. Torralba

Minimum cut

- We can do segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

Minimum cut example



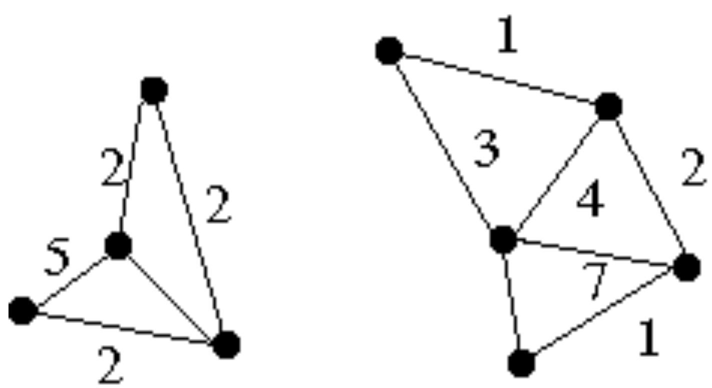


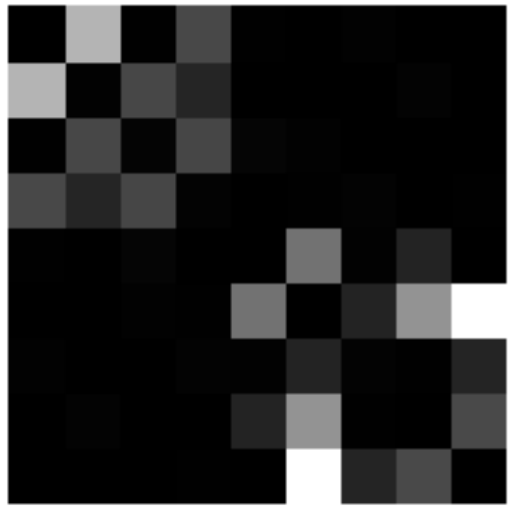
Slide credit: S. Lazebnik

Minimum cut

- We can do segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

Minimum cut example

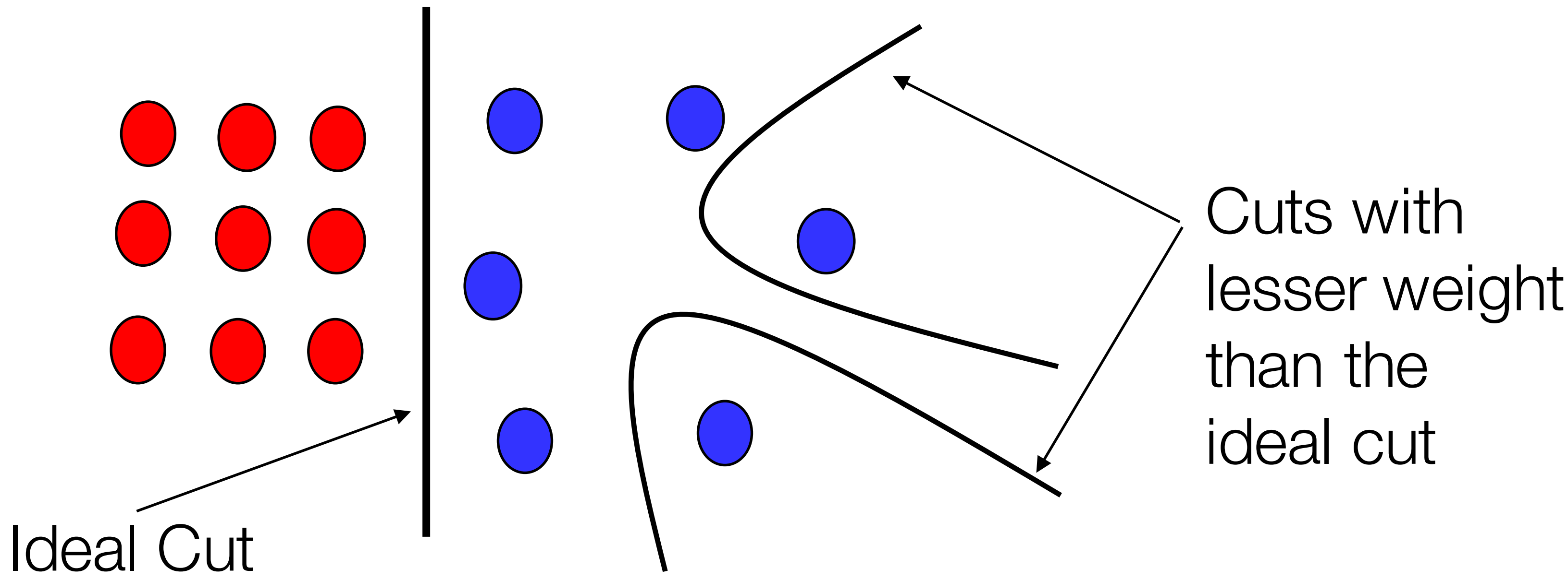




Slide credit: S. Lazebnik

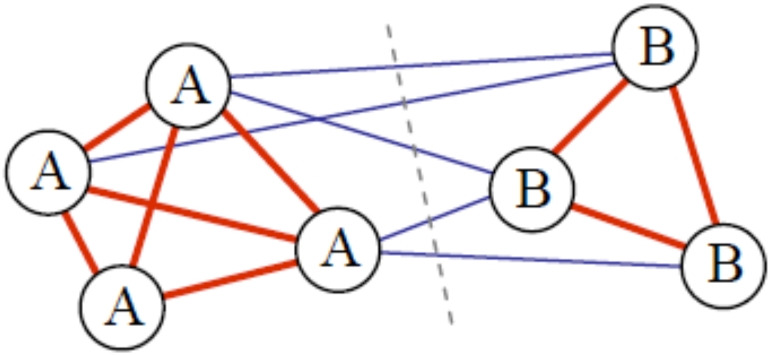
Drawbacks of Minimum cut

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



Normalized cuts

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



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

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

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

with $A \cap B = \varnothing$

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

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

A and B not necessarily disjoint

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Slide credit: B. Freeman and A. Torralba

Normalized cut

- Let W be the adjacency matrix of the graph
- Let D be the diagonal matrix with diagonal entries
  $D(i, i) = \Sigma_j W(i, j)$
- Then the normalized cut cost can be written as

$$\frac{y^T(D-W)y}{y^T Dy}$$

where y is an indicator vector whose value should be 1 in the ith position if the ith feature point belongs to A and a negative constant otherwise

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Slide credit: S. Lazebnik

Normalized cut

- Finding the exact minimum of the normalized cut cost is NP-complete, but if we *relax* y to take on arbitrary values, then we can minimize the relaxed cost by solving the *generalized eigenvalue problem* $(D - W)y = \lambda Dy$
- The solution y is given by the generalized eigenvector corresponding to the second smallest eigenvalue
- Intitutively, the ith entry of y can be viewed as a "soft" indication of the component membership of the ith feature
  - Can use 0 or median value of the entries as the splitting point (threshold), or find threshold that minimizes the Ncut cost

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Slide credit: S. Lazebnik

Normalized cut 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.

Slide credit: B. Freeman and A. Torralba

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)

Slide credit: B. Freeman and A. Torralba

Boundaries of image regions defined by a number of attributes

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

[Malik]

Slide credit: B. Freeman and A. Torralba

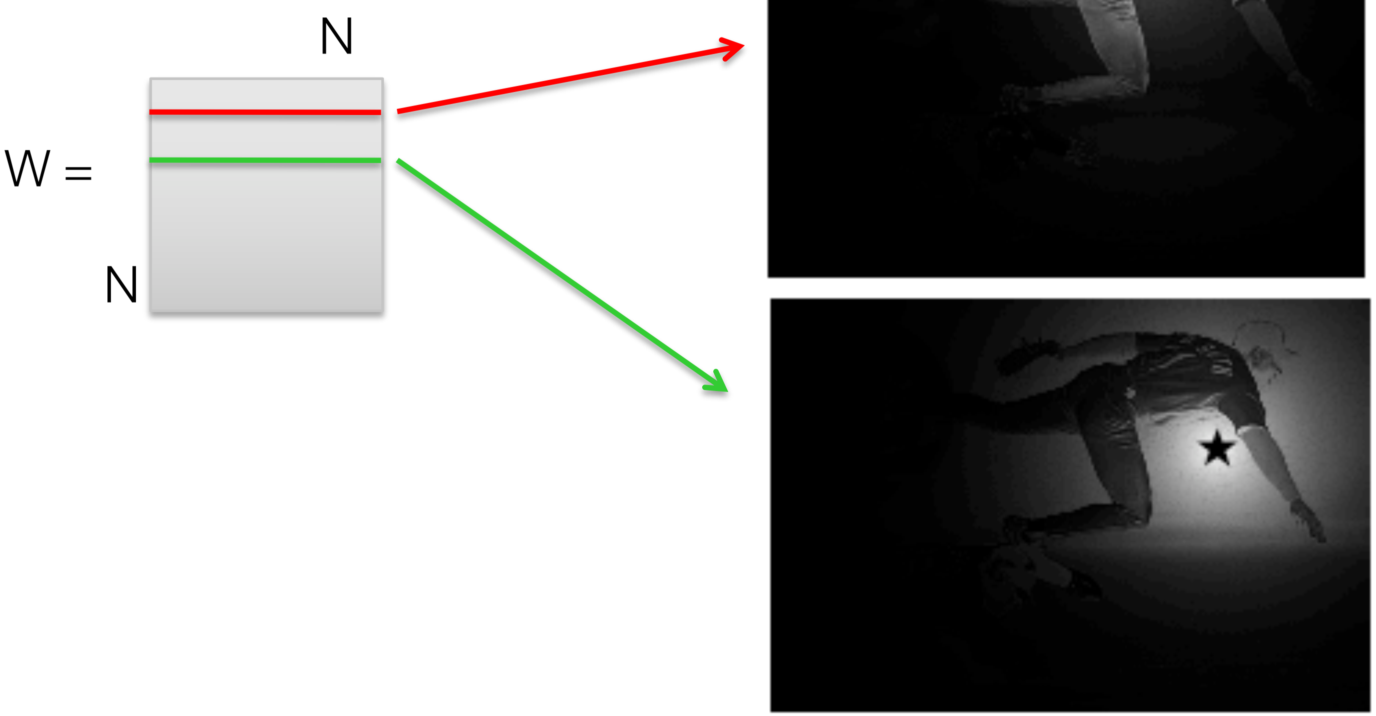
Example

Affinity:

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

brightness　　　　Location

N pixels = ncols * nrows



Slide credit: B. Freeman and A. Torralba

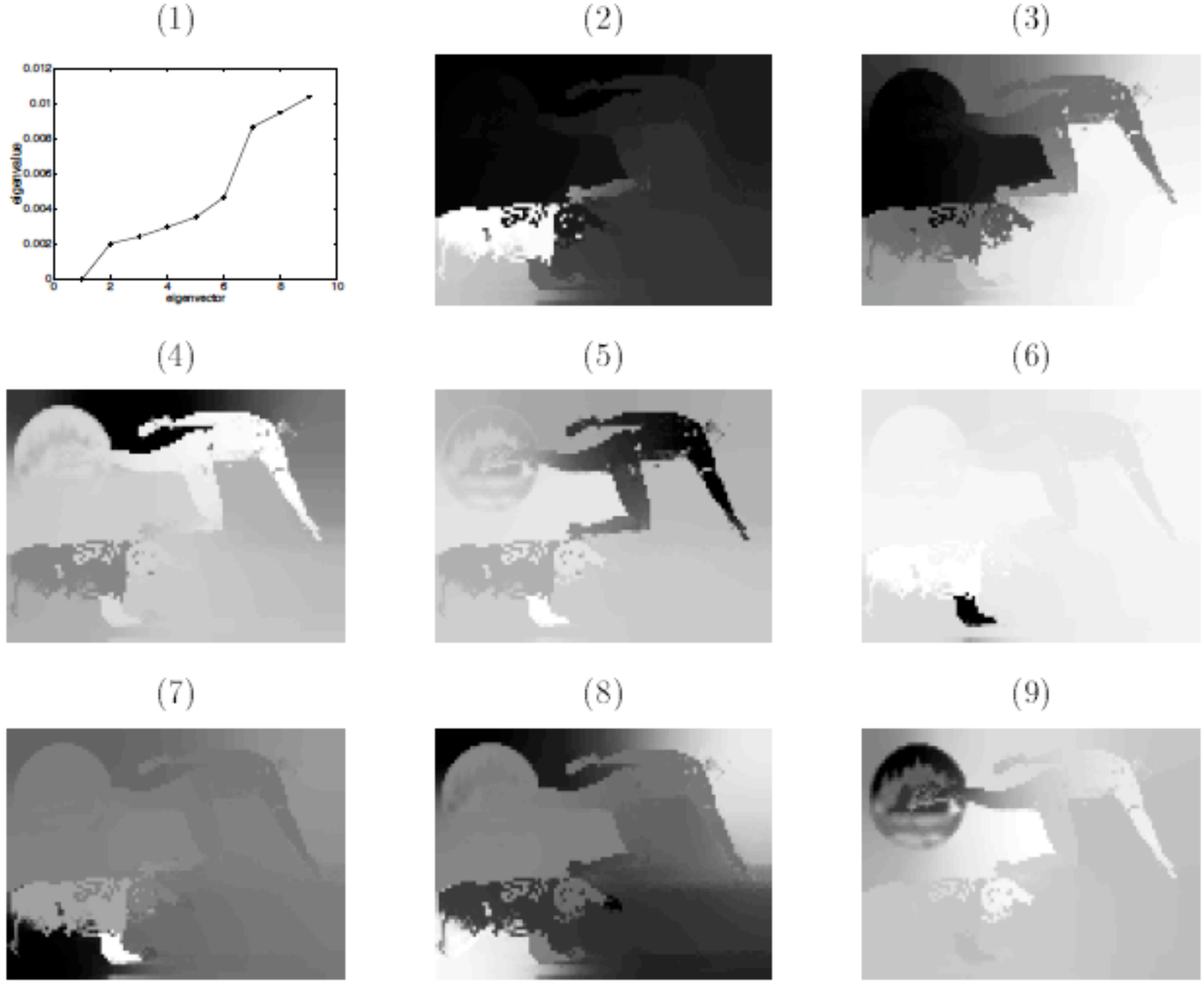
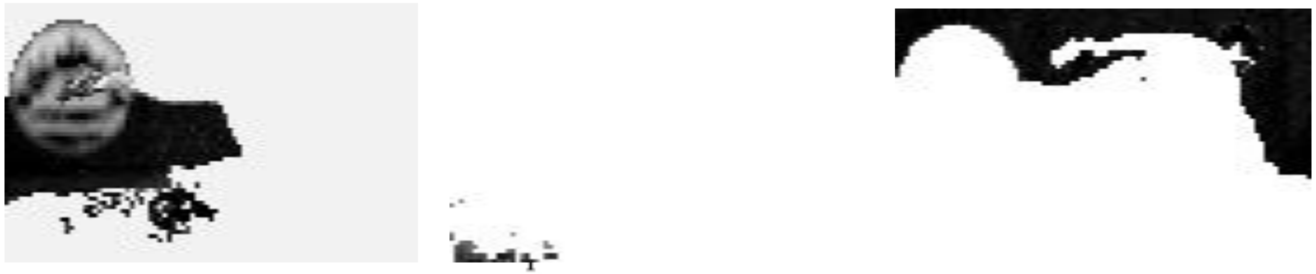


Figure 12: Subplot (1) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplot (2) - (9) shows the eigenvectors corresponding the 2nd smallest to the 9th smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

Slide credit: B. Freeman and A. Torralba

Brightness Image Segmentation



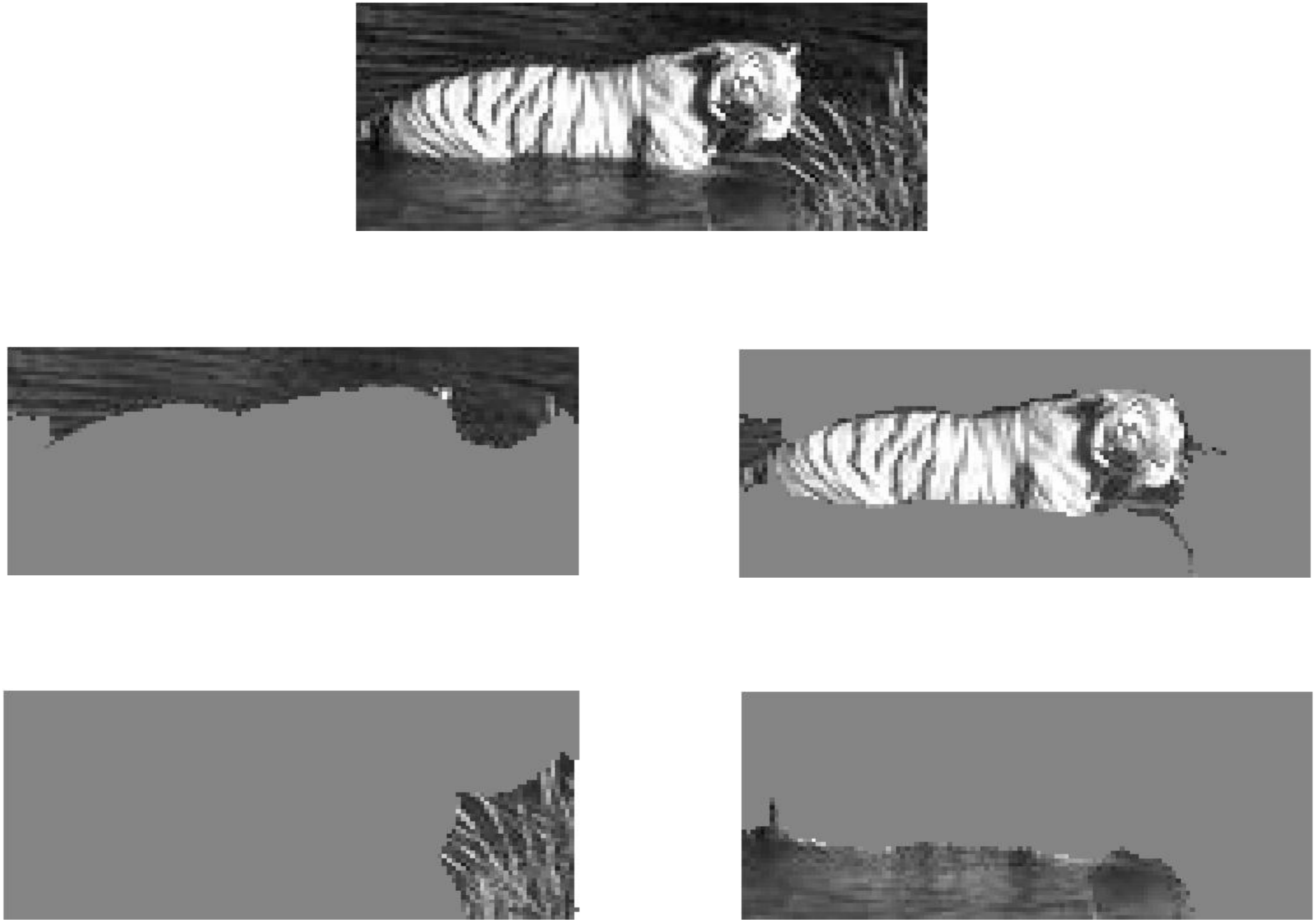
converge. On the 100×120 test images shown here, the normalized cut algorithm takes about 2 minutes on Intel Pentium 200MHz machines.

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

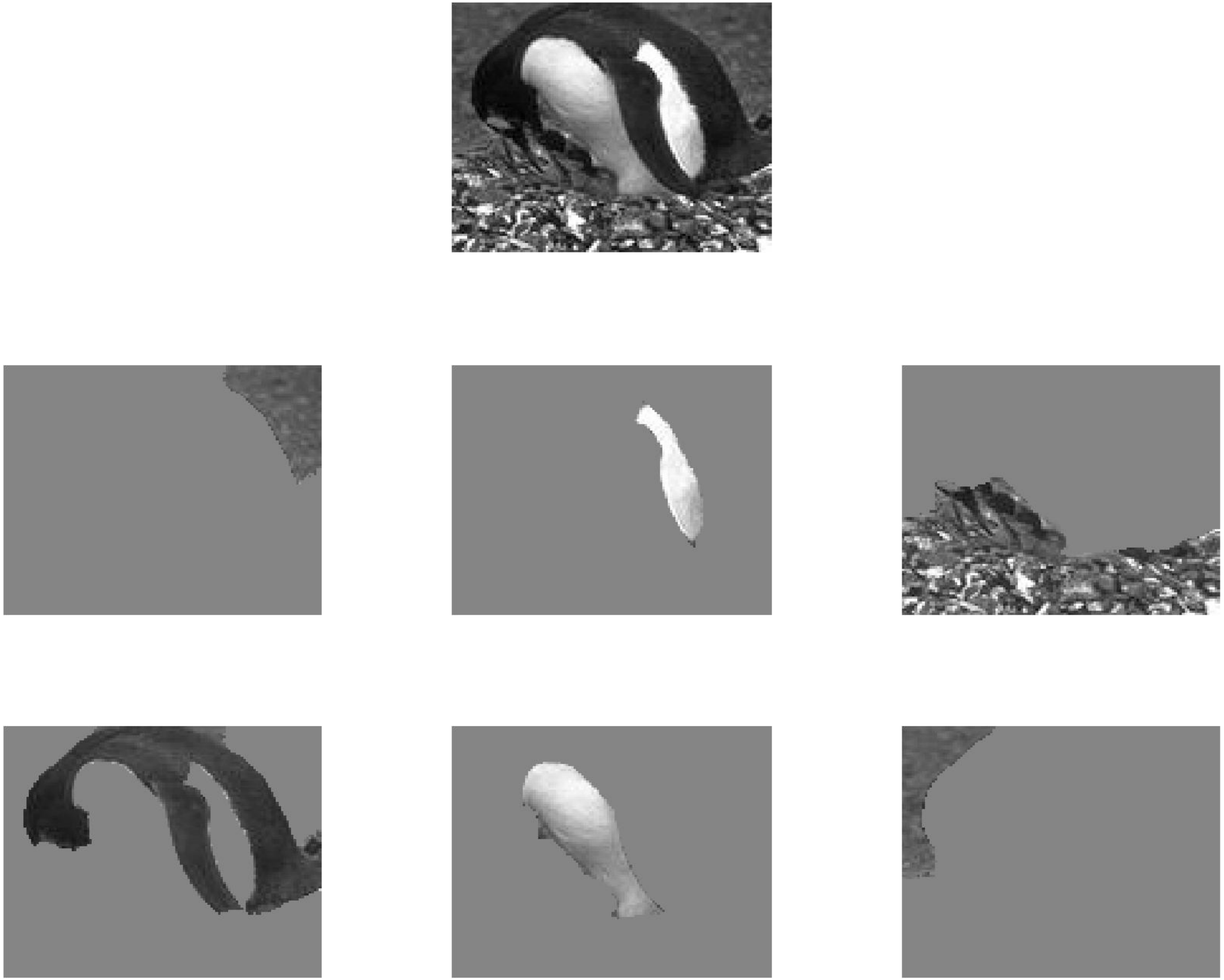
Slide credit: B. Freeman and A. Torralba

Brightness Image Segmentation



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

Slide credit: B. Freeman and A. Torralba



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

Slide credit: B. Freeman and A. Torralba

Example results

Slide credit: S. Lazebnik

Results: Berkeley Segmentation Engine

http://www.cs.berkeley.edu/~fowlkes/BSE/

Slide credit: S. Lazebnik

Normalized cuts: Pro and con

- Pros
  - Generic framework, can be used with many different features and affinity formulations
- Cons
  - High storage requirement and time complexity
  - Bias towards partitioning into equal segments

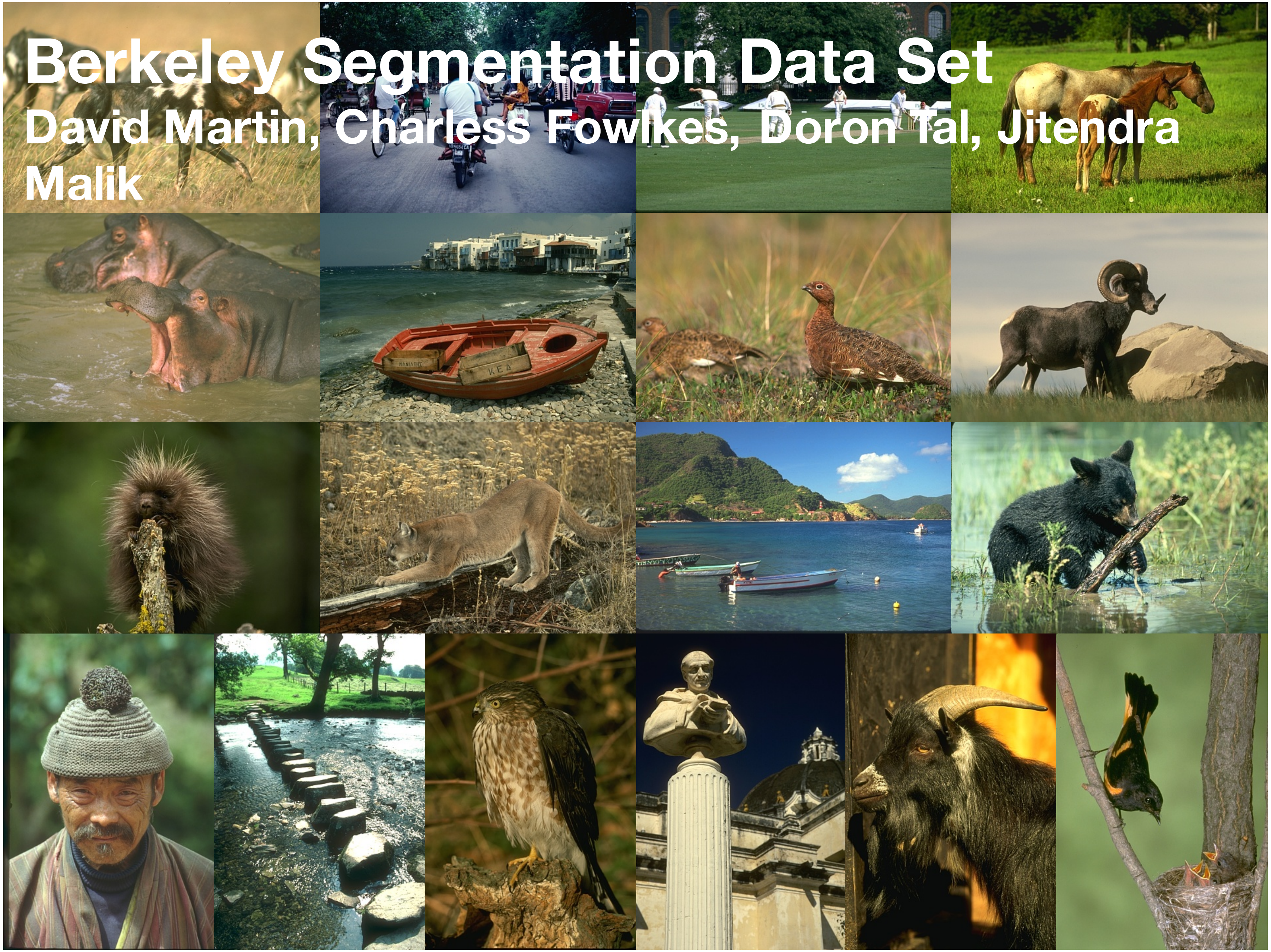
Slide credit: S. Lazebnik

Segmentation methods

- K-means clustering
- Graph-theoretic segmentation
- Boundary Detection

Berkeley Segmentation Data Set

David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik

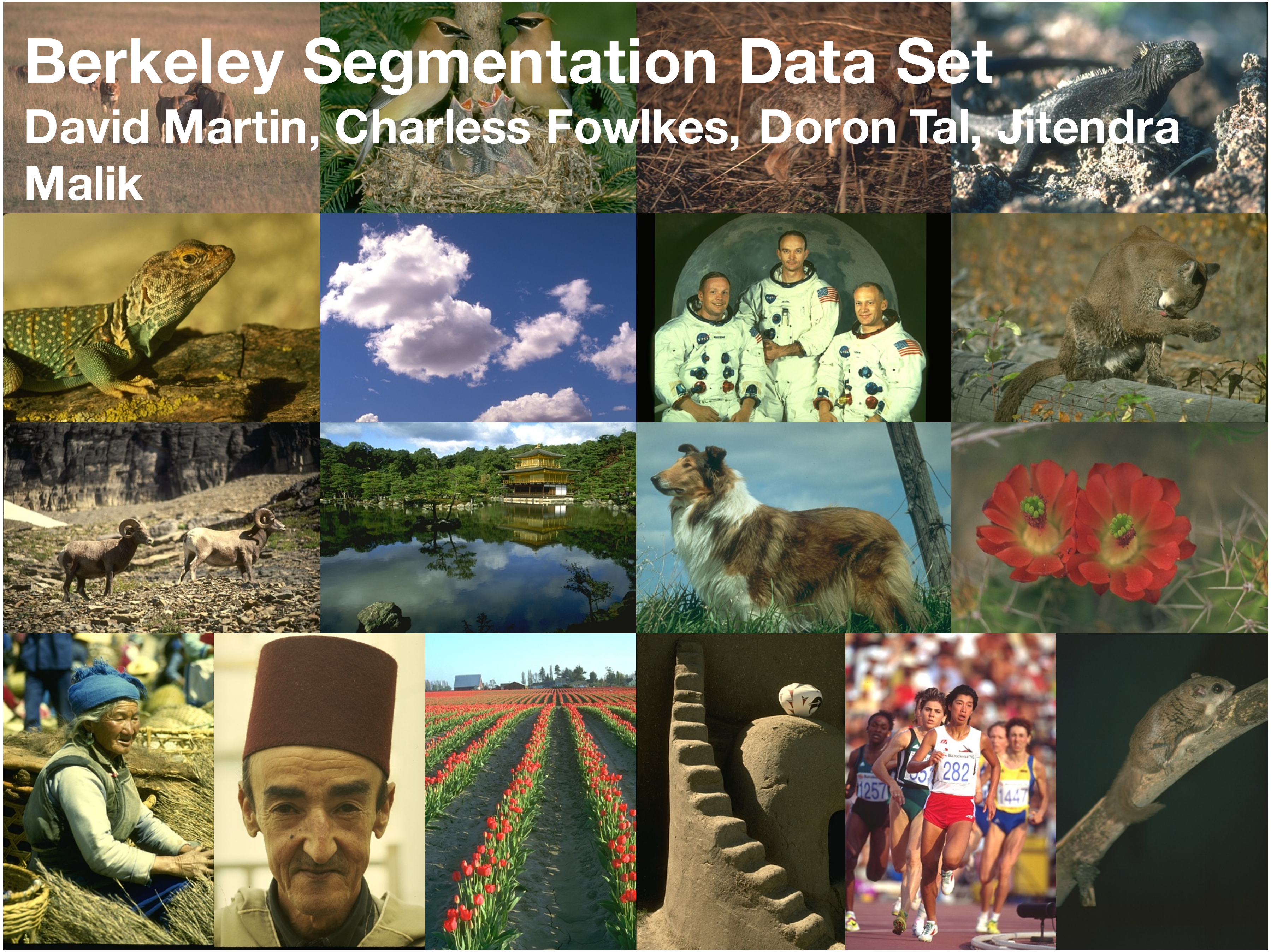


Slide credit: J. Hays

Berkeley Segmentation Data Set

David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik

Slide credit: J. Hays



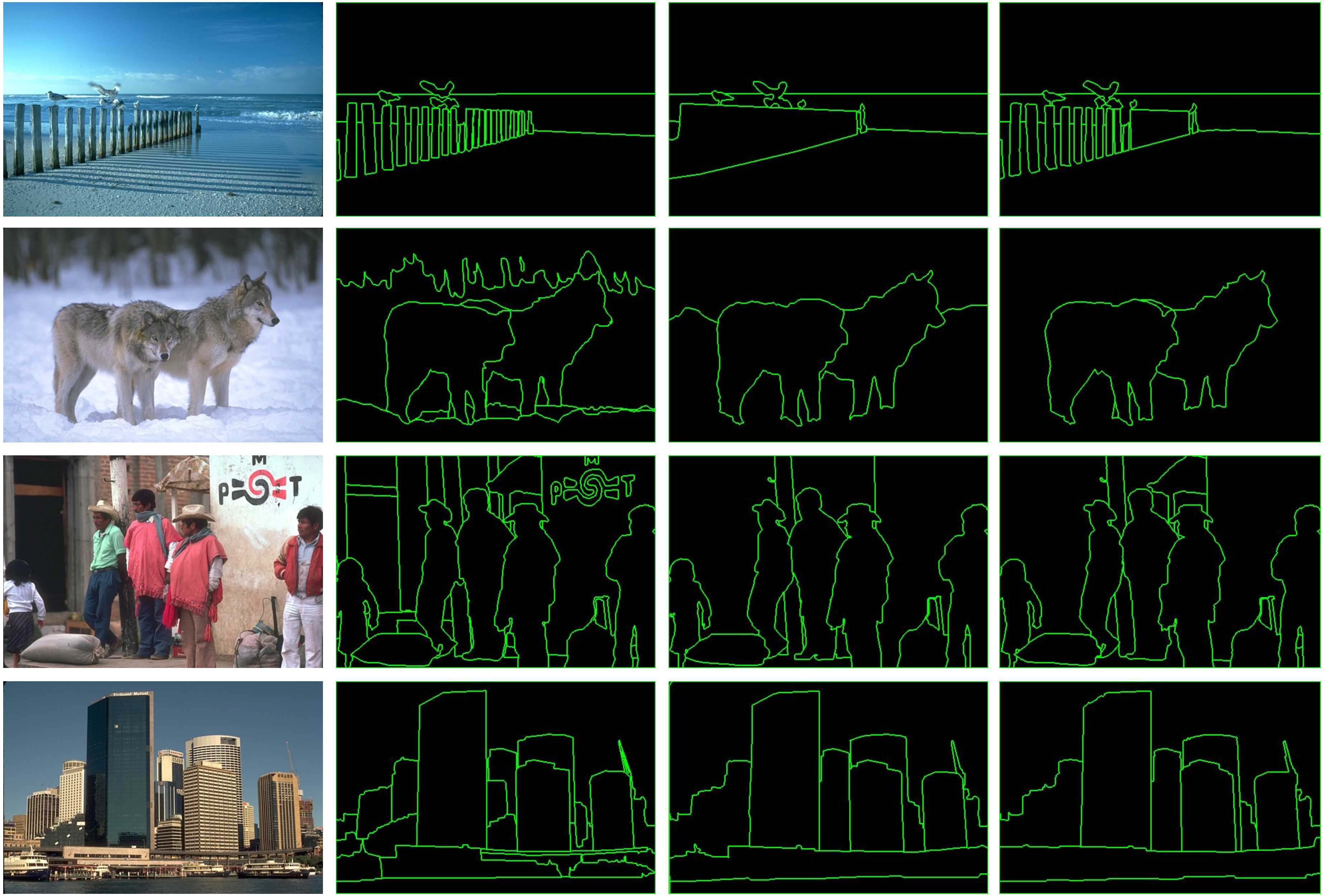
Slide credit: J. Hays

Protocol

You will be presented a photographic image. Divide the image into some number of segments, where the segments represent "things" or "parts of things" in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance.

- Custom segmentation tool
- Subjects obtained from work-study program (UC Berkeley undergraduates)

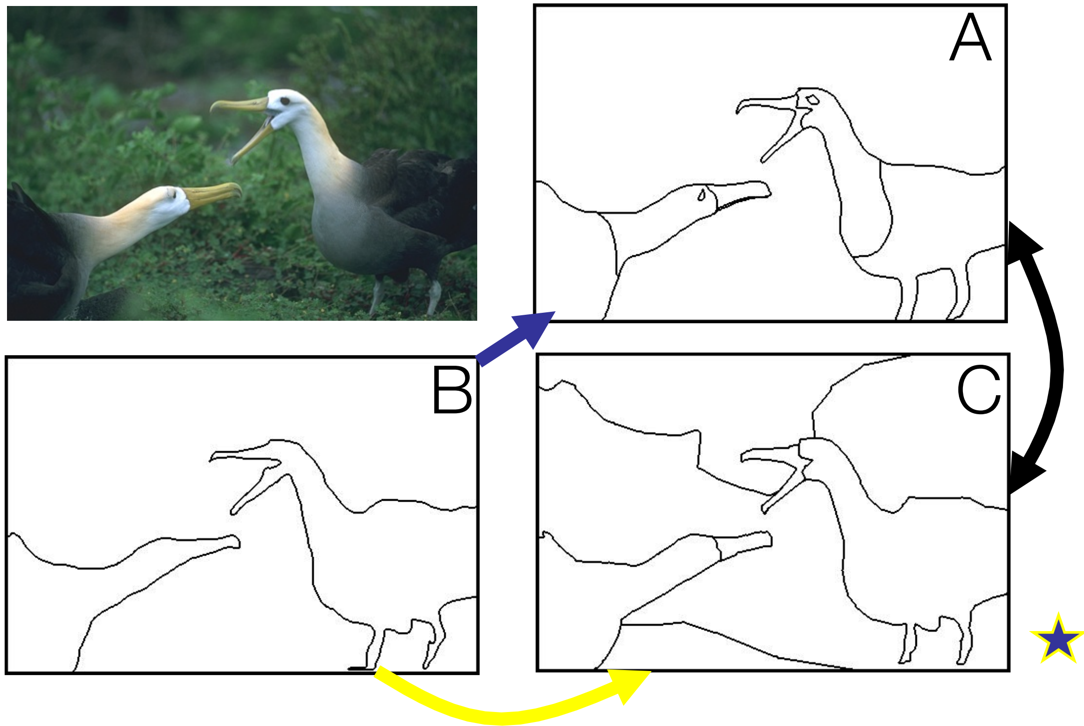
Slide credit: J. Hays



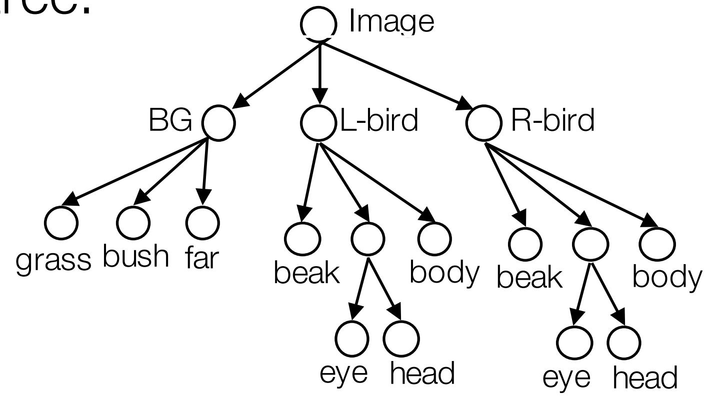
Slide credit: J. Hays

Slide credit: J. Hays

Segmentations are Consistent



Perceptual organization forms a tree:



- A,C are refinements of B
- A,C are mutual refinements
- A,B,C represent the same percept
  - Attention accounts for differences

★ Two segmentations are consistent when they can be explained by the same segmentation tree (i.e. they could be derived from a single perceptual organization).

Slide credit: J. Hays

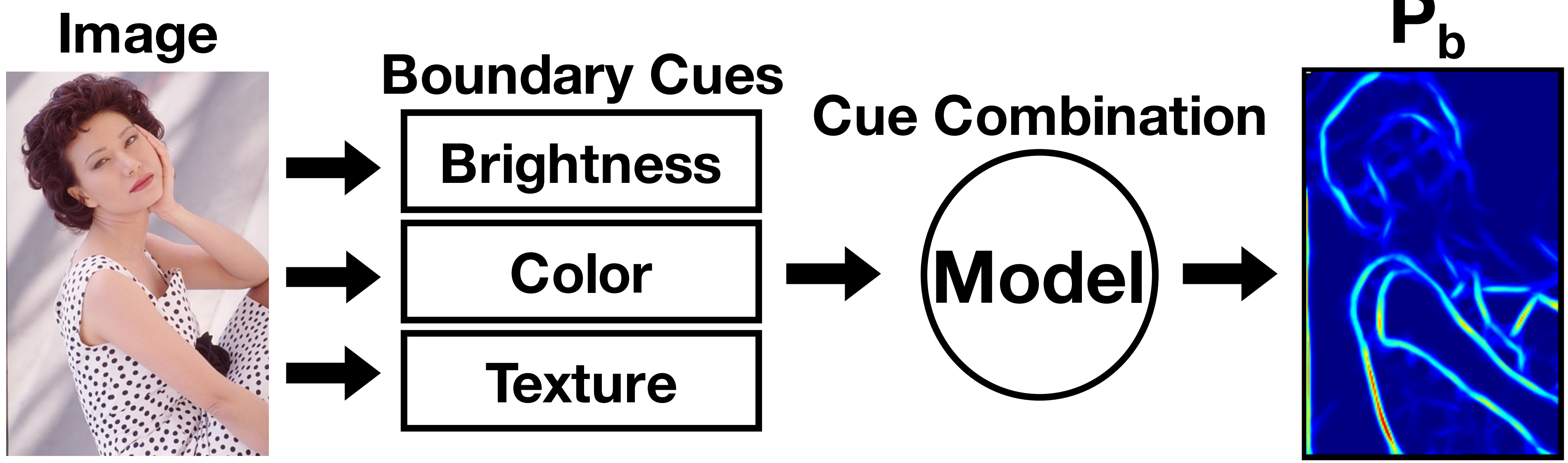
Slide credit: J. Hays

Dataset Summary

- 30 subjects, age 19-23
  - 17 men, 13 women
  - 9 with artistic training
- 8 months
- 1,458 person hours
- 1,020 Corel images
- 11,595 Segmentations
  - 5,555 color, 5,554 gray, 486 inverted/negated

Slide credit: J. Hays

Pb Detector



Challenges: texture cue, cue combination

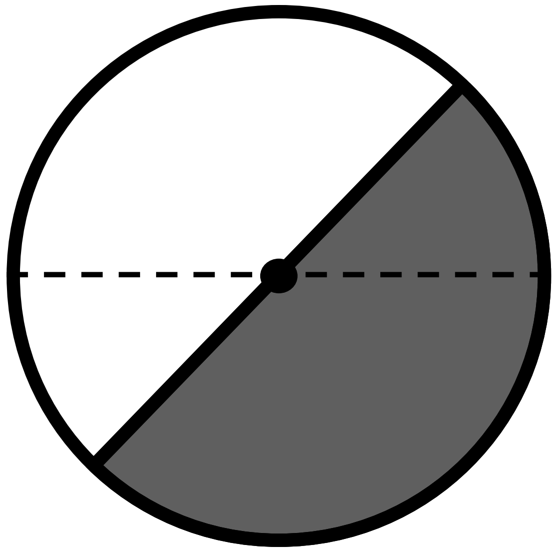
Goal: learn the posterior probability of a boundary P_b from local information only

Slide credit: J. Hays

Brightness and Color Features

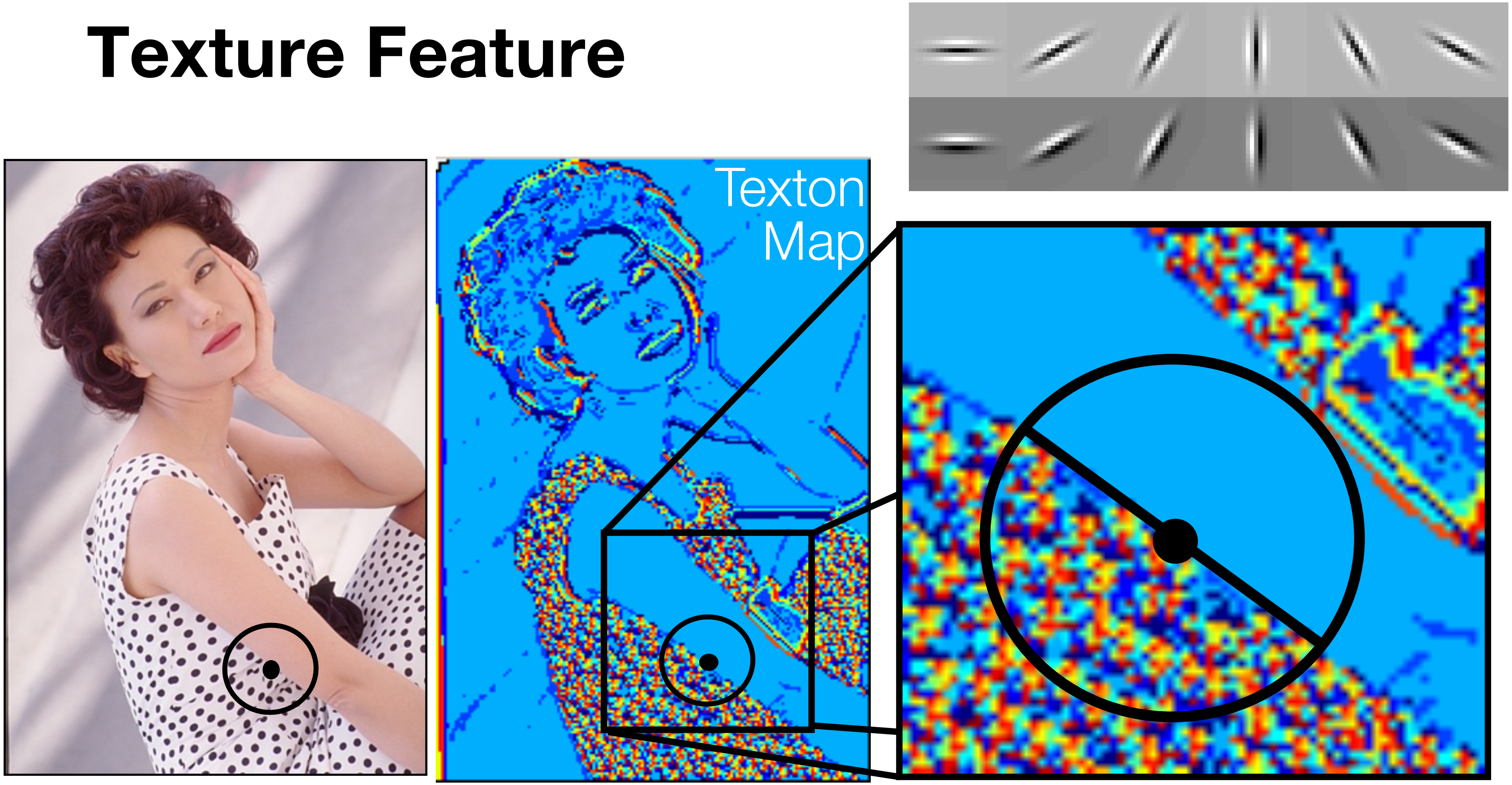
- 1976 CIE L*a*b* colorspace
- Brightness Gradient (B)
  - Chi[2] difference in L* distribution
- Color Gradient (C)
  - Chi[2] difference in a* and b* distributions

$$\chi^2(g,h) = \frac{1}{2}\sum_i \frac{(g_i - h_i)^2}{g_i + h_i}$$



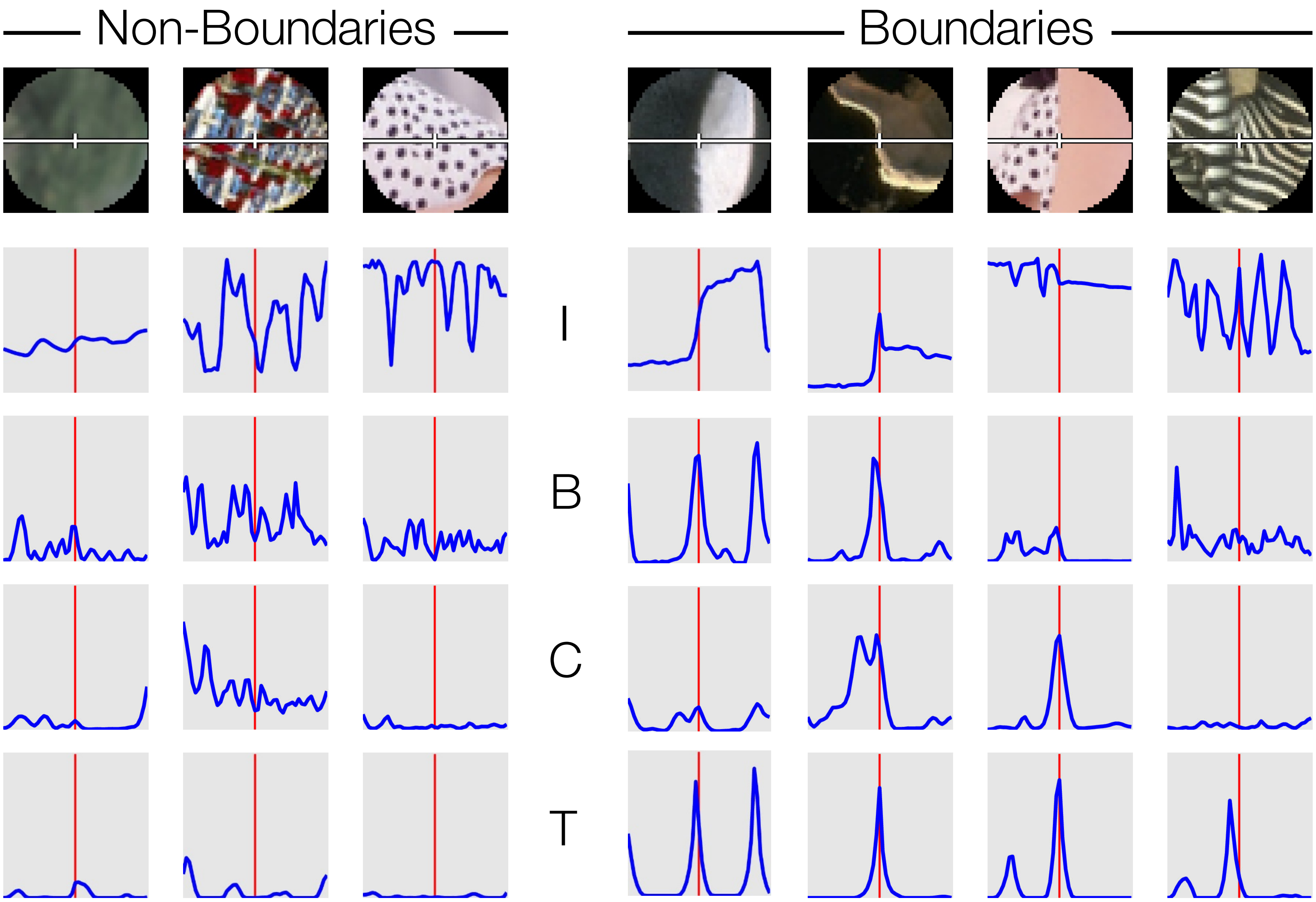
Slide credit: J. Hays

Texture Feature



- Texture Gradient (T)
- Chi^2 difference of texton histograms
  - Textons are vector-quantized filter outputs

Slide credit: J. Hays



Slide credit: J. Hays

Cue Combination Models

- Classification Trees
  - Top-down splits to maximize entropy, error bounded
- Density Estimation
  - Adaptive bins using k-means
- Logistic Regression, 3 variants
  - Linear and quadratic terms
  - Confidence-rated generalization of AdaBoost (Schapire&Singer)
- Hierarchical Mixtures of Experts (Jordan&Jacobs)
  - Up to 8 experts, initialized top-down, fit with EM
- Support Vector Machines (libsvm, Chang&Lin)

- Range over bias, complexity, parametric/non-parametric

Slide credit: J. Hays

Computing Precision/Recall

Recall = Pr(signal|truth) = fraction of ground truth found by the signal

Precision = Pr(truth|signal) = fraction of signal that is correct

- Always a trade-off between the two
- Standard measures in information retrieval (van Rijsbergen XX)
- ROC from standard signal detection the wrong approach

Strategy

- Detector output (Pb) is a soft boundary map
- Compute precision/recall curve:
  - Threshold Pb at many points t in [0,1]
  - Recall = Pr(Pb>t|seg=1)
  - Precision = Pr(seg=1|Pb>t)

Slide credit: J. Hays

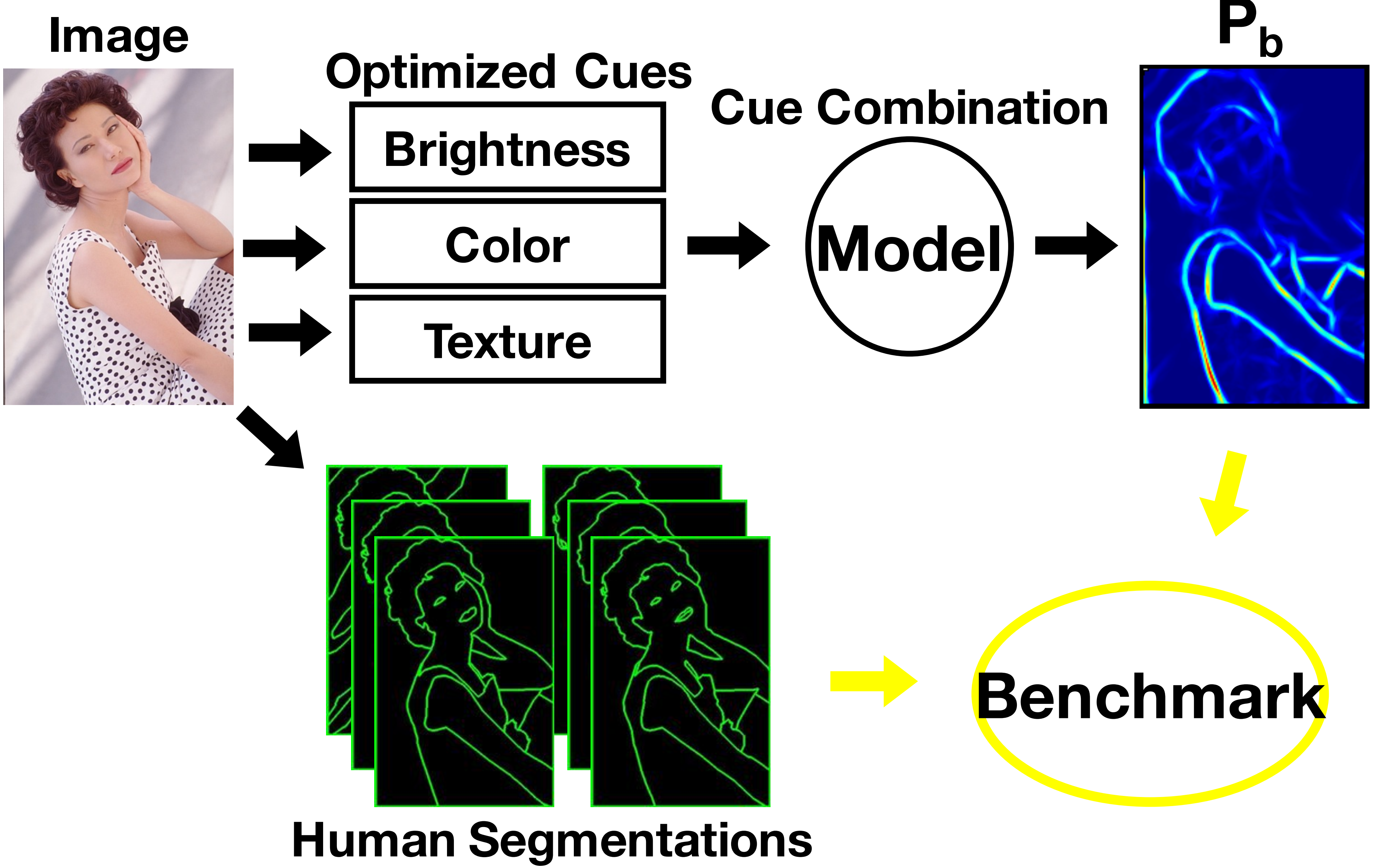
Cue Calibration

- All free parameters optimized on training data
- All algorithmic alternatives evaluated by experiment

- Brightness Gradient
  - Scale, bin/kernel sizes for KDE
- Color Gradient
  - Scale, bin/kernel sizes for KDE, joint vs. marginals
- Texture Gradient
  - Filter bank: scale, multiscale?
  - Histogram comparison
  - Number of textons, Image-specific vs. universal textons
- Localization parameters for each cue

Slide credit: J. Hays

Dataflow

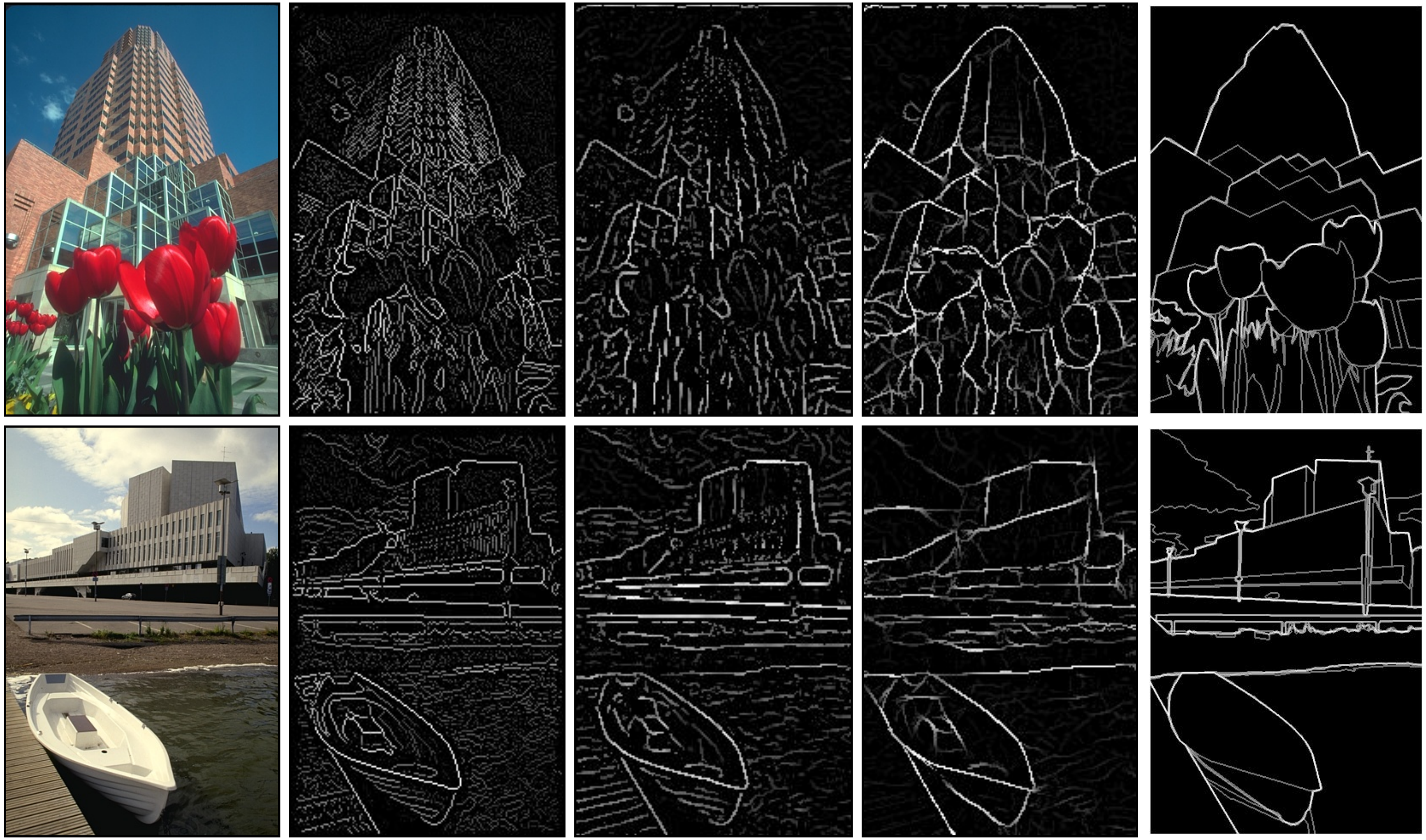


Slide credit: J. Hays

P_b Images

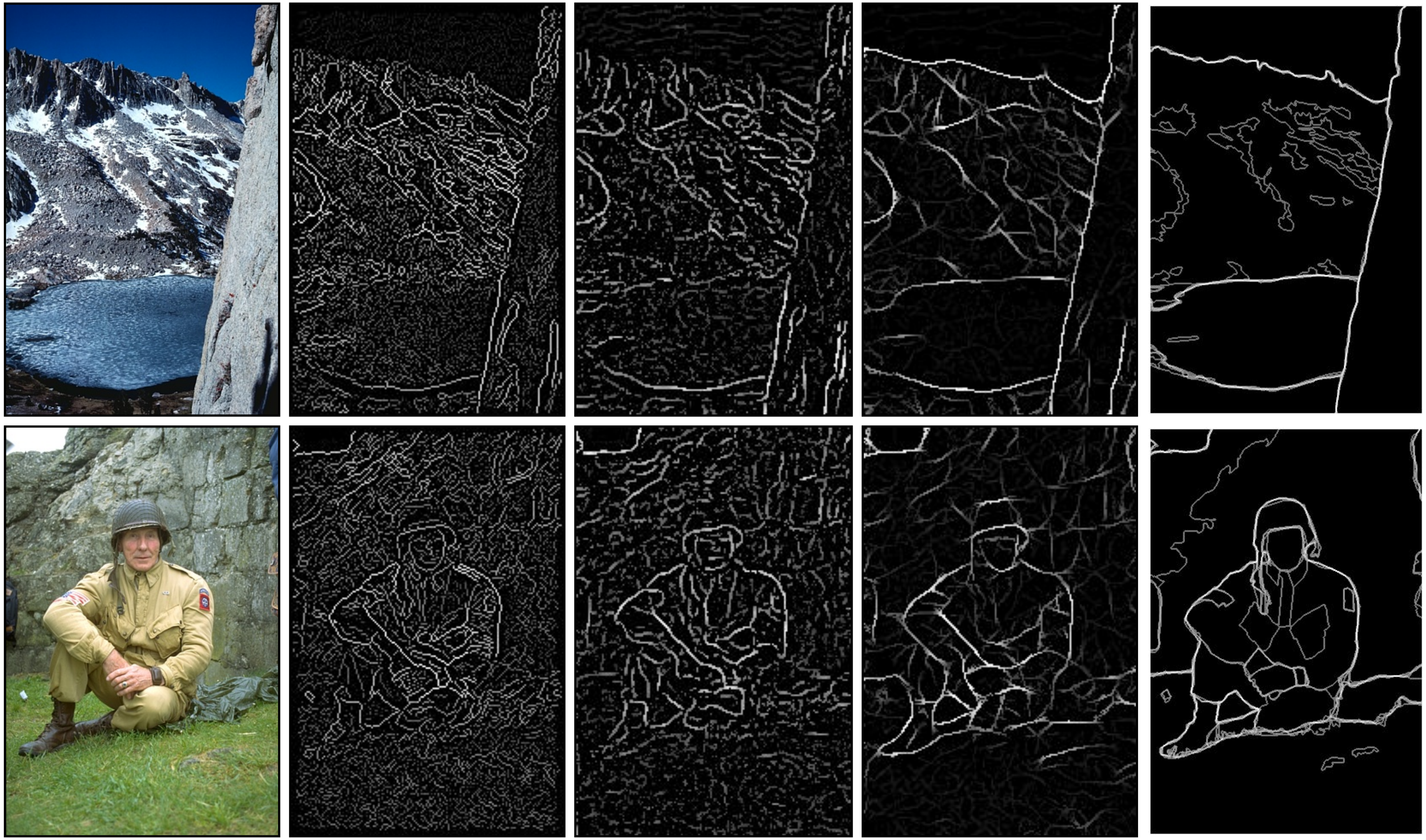
Slide credit: J. Hays

P_b Images II



Slide credit: J. Hays

P_b Images III



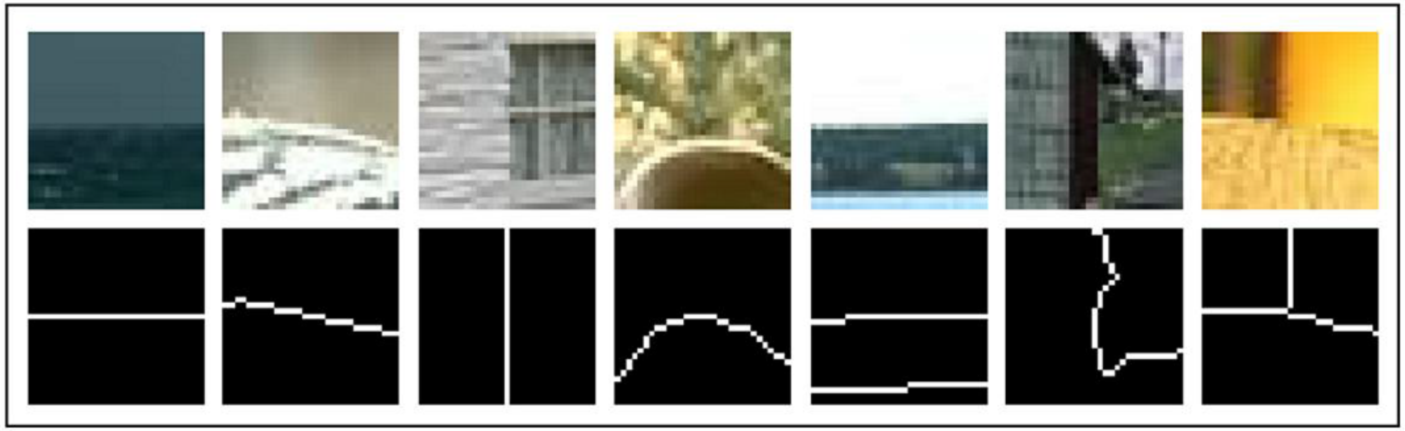
Slide credit: J. Hays

Findings

1. A simple linear model is sufficient for cue combination
   - All cues weighted approximately equally in logistic
2. Proper texture edge model is not optional for complex natural images
   - Texture suppression is not sufficient!
3. Significant improvement over state-of-the-art in boundary detection
4. Empirical approach critical for both cue calibration and cue combination

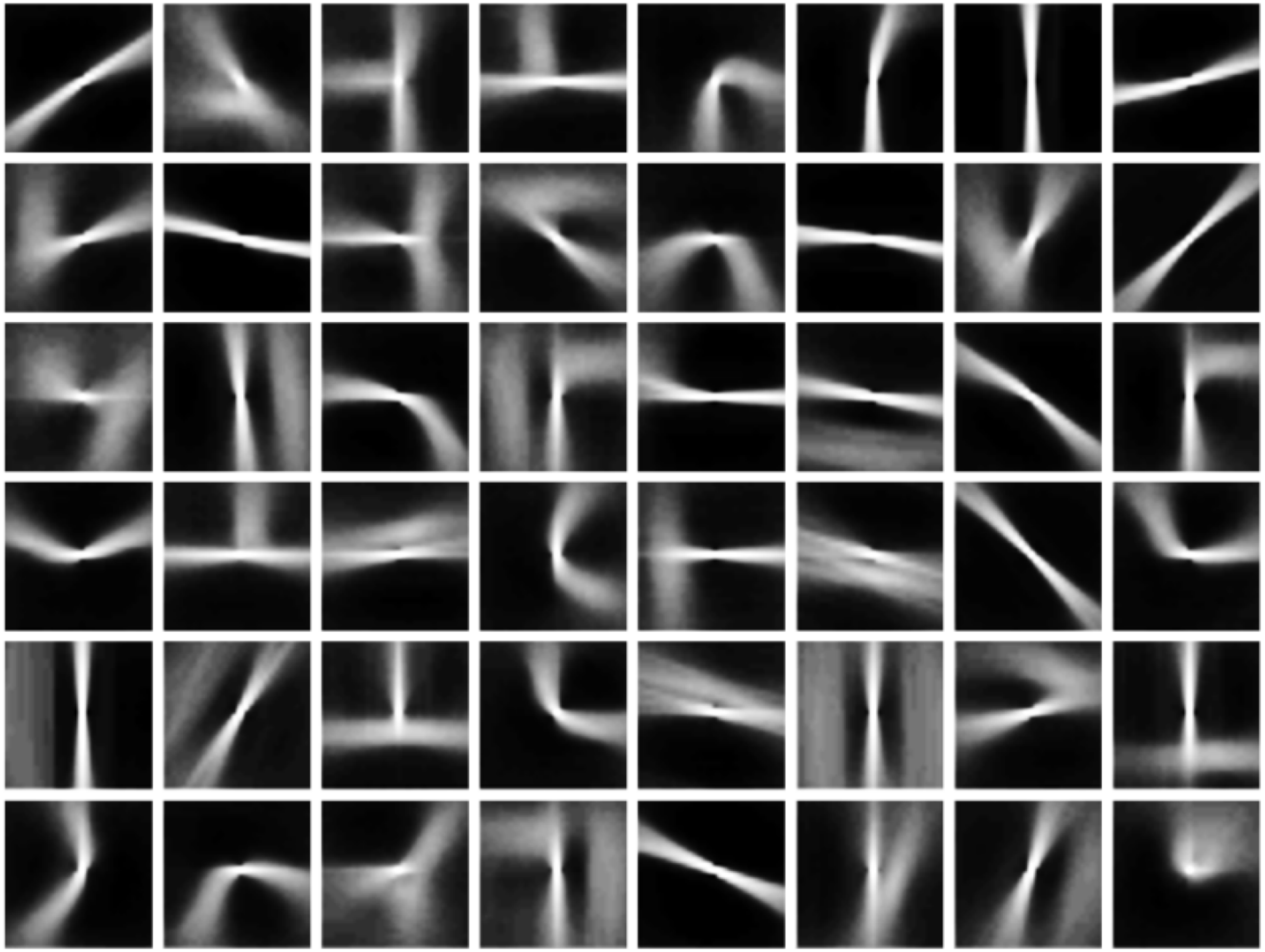
Slide credit: J. Hays

Sketch Tokens (J. Lim et al., CVPR 2013)



Slide credit: J. Hays

Sketch Tokens (J. Lim et al., CVPR 2013)



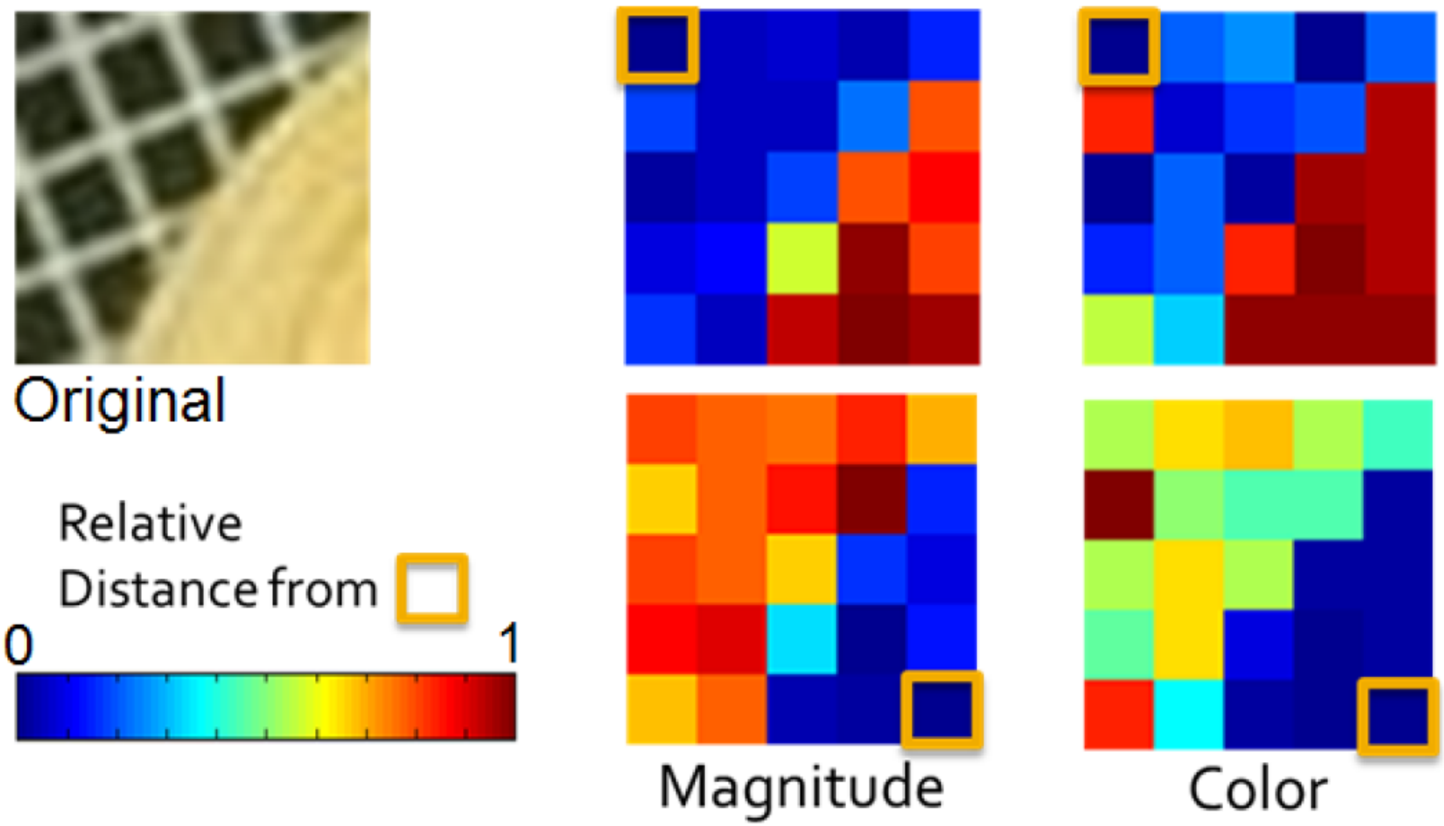
Slide credit: J. Hays

Image Features – 21350 dimensions!

- 35x35 patches centered at every pixel
- 35x35 "channels" of many types:
  - Color (3 channels)
  - Gradients (3 unoriented + 8 oriented channels)
    - Sigma = 0, T heta = 0, pi/2, pi, 3pi/2
    - Sigma = 1.5, Theta = 0, pi/2, pi, 3pi/2
    - Sigma = 5
  - Self Similarity
    - 5x5 maps of self similarity within the above channels for a particular anchor point.

Slide credit: J. Hays

Self-similarity features



Self-similarity features: The L1 distance from the anchor cell (yellow box) to the other 5 x 5 cells are shown for color and gradient magnitude channels. The original patch is shown to the left.

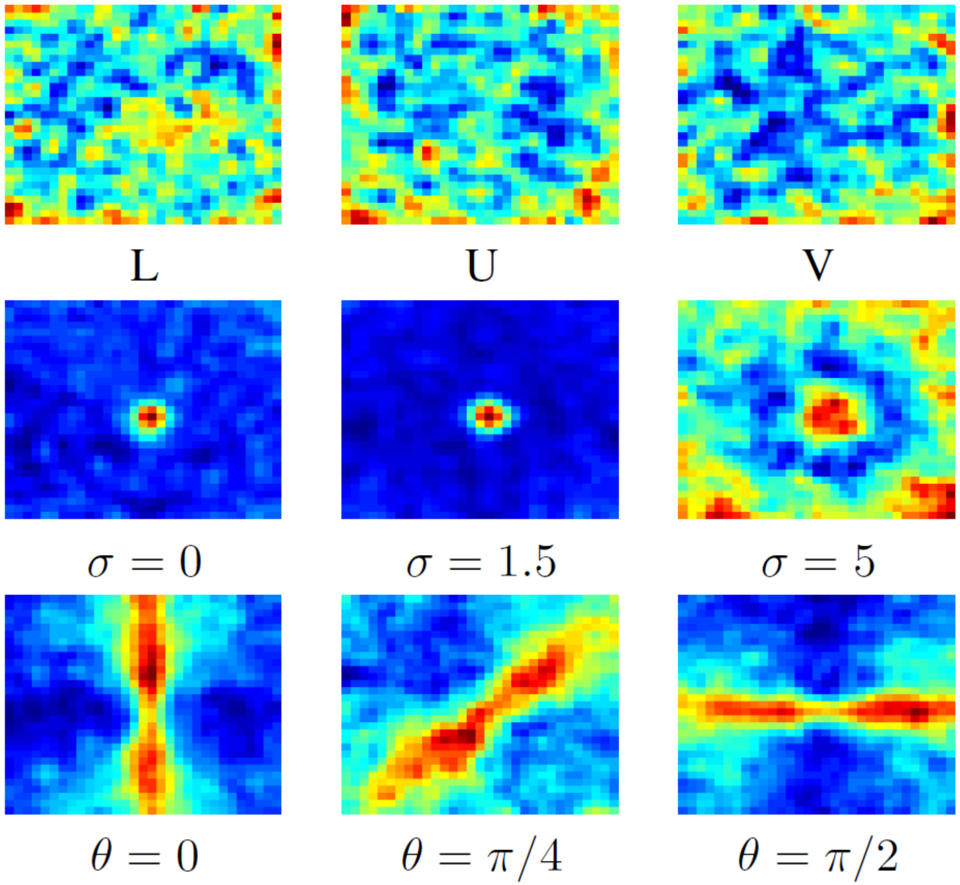
Slide credit: J. Hays

Learning

- Random Forest Classifiers, one for each sketch token + background, trained 1-vs-all
- Advantages:
  - Fast at test time, especially for a non-linear classifier.
  - Don't have to explicitly compute independent descriptors for every patch. Just look up what the decision tree wants to know at each branch.

Slide credit: J. Hays

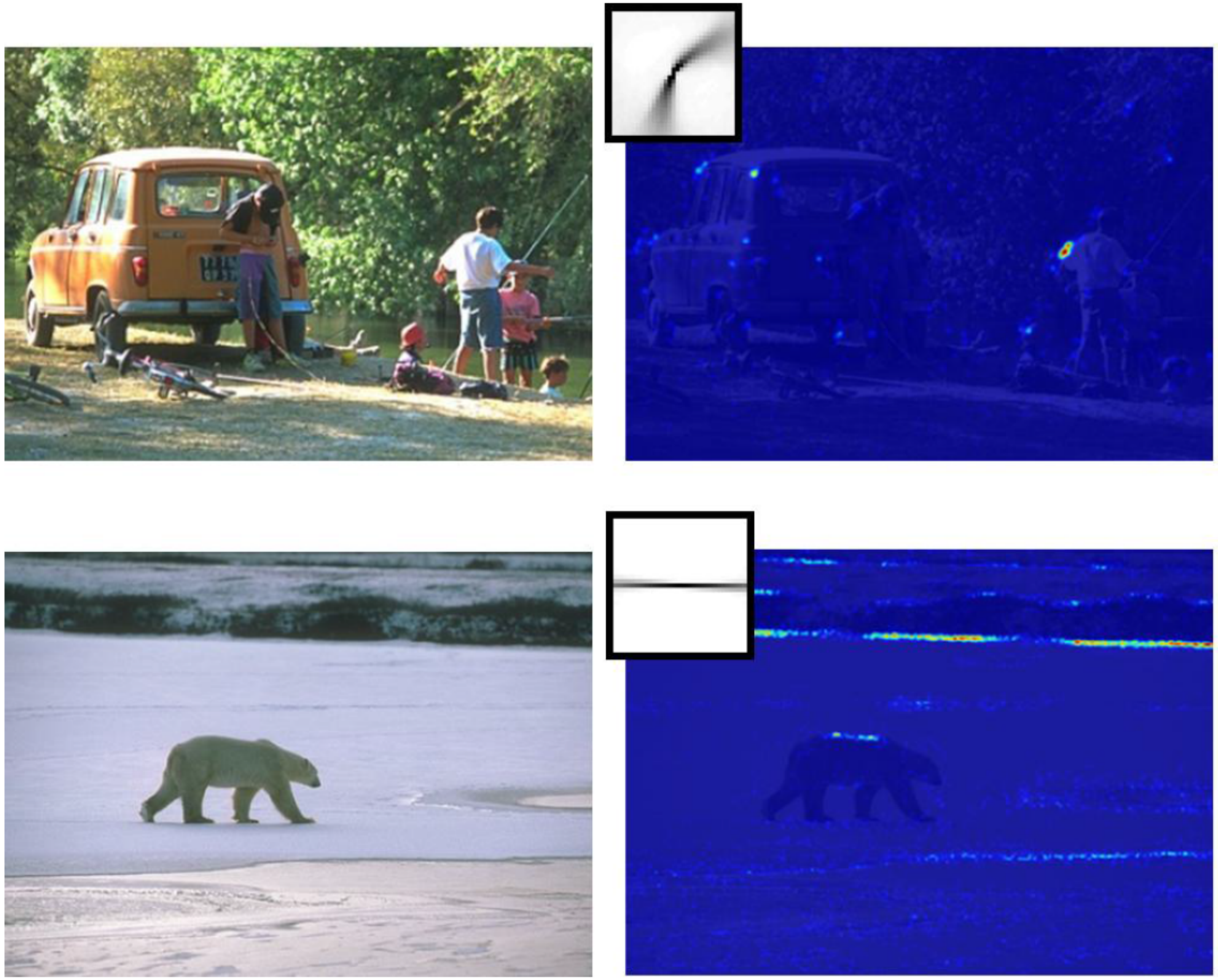
Learning



Frequency of example features being selected by the random forest: (first row) color channels, (second row) gradient magnitude channels, (third row) selected orientation channels.

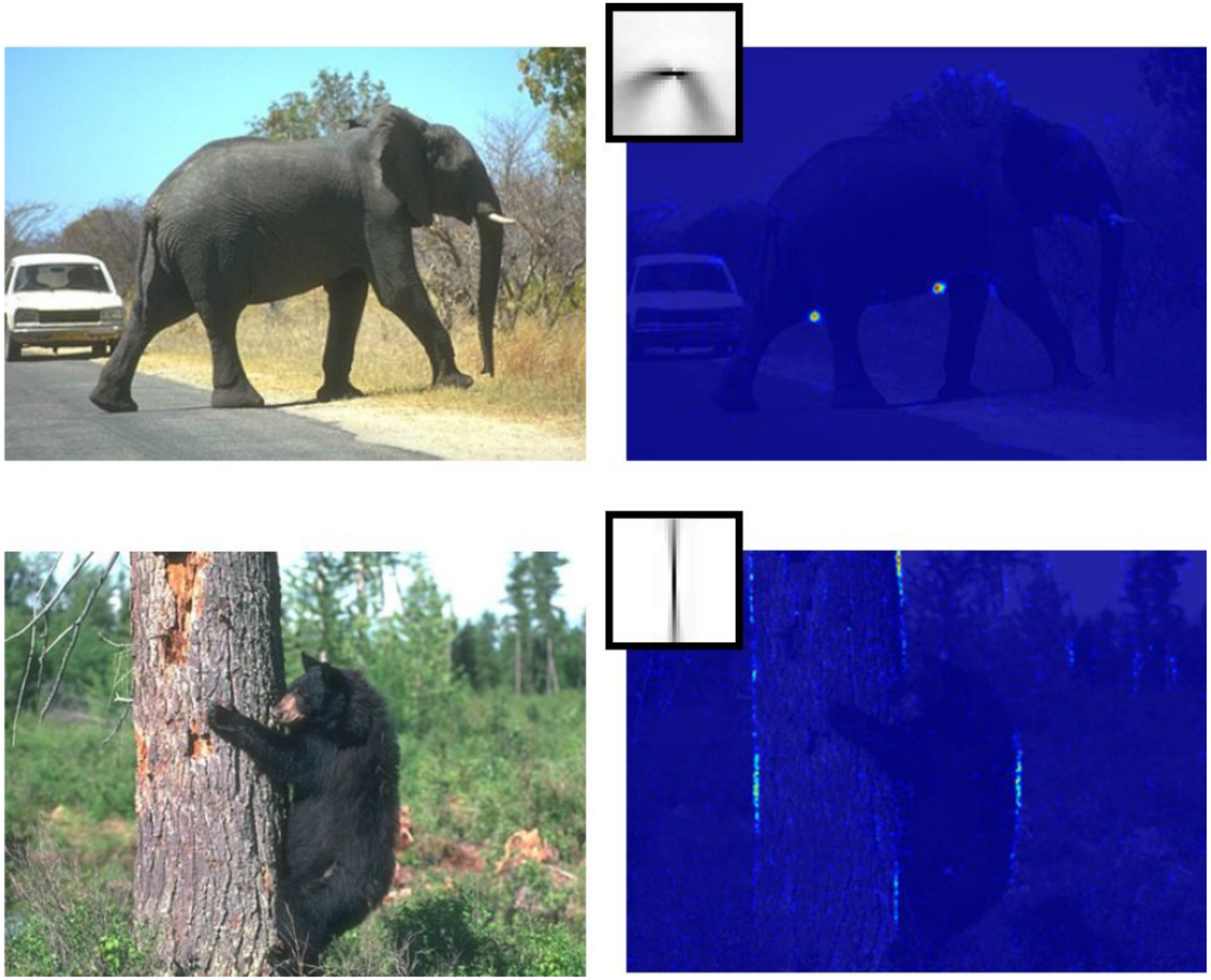
Slide credit: J. Hays

Detections of individual sketch tokens



Slide credit: J. Hays

Detections of individual sketch tokens



Slide credit: J. Hays

Combining sketch token detections

- Simply add the probability of all non-background sketch tokens

- Free parameter: number of sketch tokens
  - k = 1 works poorly, k = 16 and above work OK.

Slide credit: J. Hays

Input Image Ground Truth Sketch Tokens

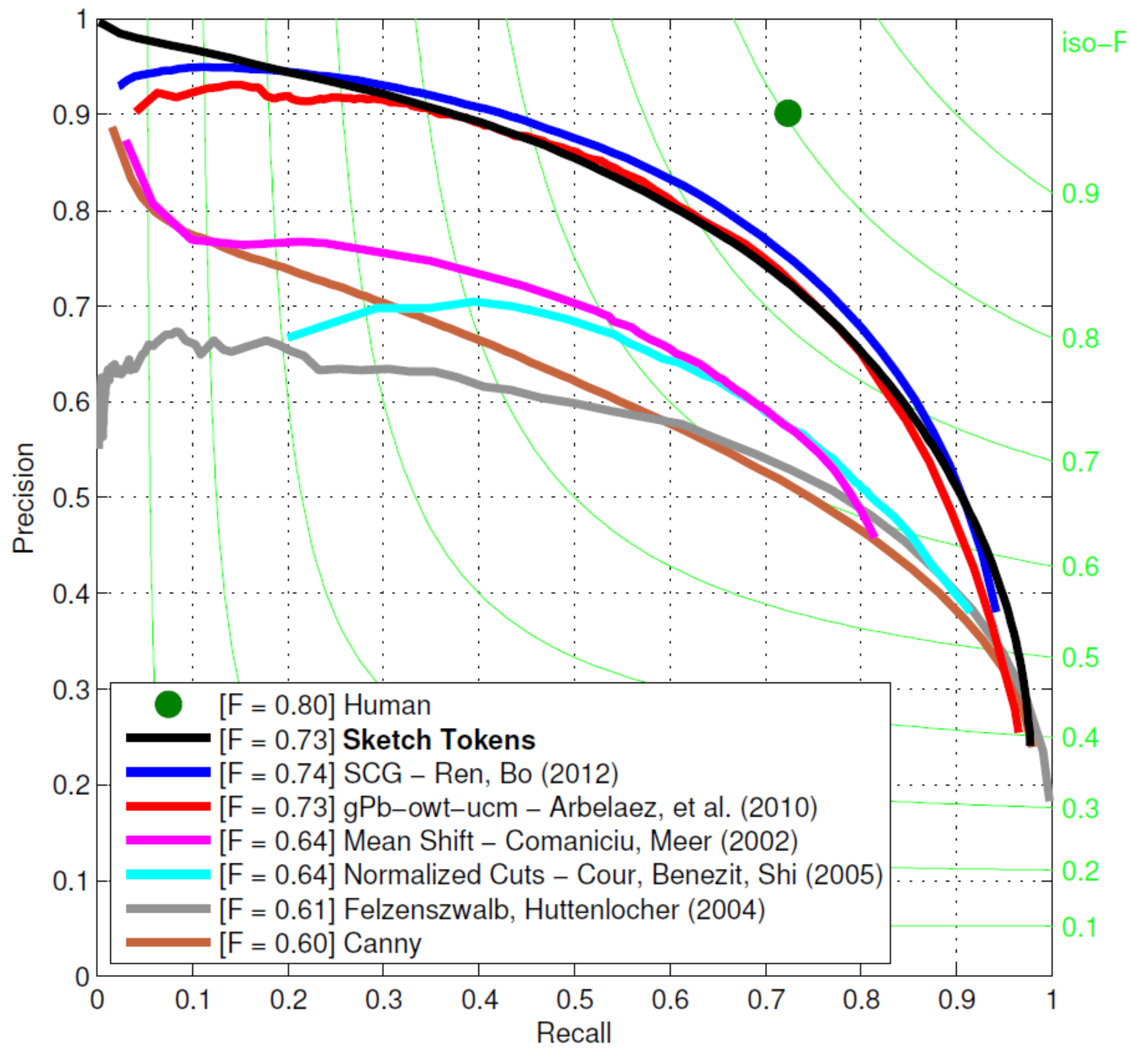
Slide credit: J. Hays

Evaluation on BSDS

Method	ODS	OIS	AP	Speed
Human	.80	.80	-	-
Canny	.60	.64	.58	1/15 s
Felz-Hutt [12]	.61	.64	.56	1/10 s
gPb (local) [1]	.71	.74	.65	60 s
SCG (local) [24]	.72	.74	.75	100 s
Sketch tokens	**.73**	**.75**	**.78**	**1 s**
gPb (global) [1]	.73	.76	.73	240 s
SCG (global) [24]	.74	.76	.77	280 s

Slide credit: J. Hays

Evaluation on BSDS



Slide credit: J. Hays

Summary

- Distinct from previous work, cluster the *human annotations* to discover the mid-level structures that you want to detect.
- Train a classifier for every sketch token.
- Is as accurate as any other method while being 200 times faster and using no global information.

Slide credit: J. Hays