BBM 413
Fundamentals of Image Processing

Erkut Erdem
Dept. of Computer Engineering
Hacettepe University

Image Segmentation
Image segmentation

- Goal: identify groups of pixels that go together
The goals of segmentation

- Separate image into coherent “objects”

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”

The goals of segmentation

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

“superpixels”

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
What is segmentation?

• Clustering image elements that “belong together”

  – **Partitioning**
    • Divide into regions/sequences with coherent internal properties

  – **Grouping**
    • Identify sets of coherent tokens in image

Slide credit: Fei-Fei Li
Segmentation is a global process

What are the occluded numbers?

Slide credit: B. Freeman and A. Torralba
Segmentation is a global process.

What are the occluded numbers?

Occlusion is an important cue in grouping.

Slide credit: B. Freeman and A. Torralba
... but not too global
Groupings by Invisible Completions

A

B

C

D

* Images from Steve Lehar’s Gestalt papers

Slide credit: B. Freeman and A. Torralba
Groupings by Invisible Completions

1970s: R. C. James

Slide credit: B. Freeman and A. Torralba
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 Psychology

- German: Gestalt - "form" or "whole"
- Berlin School, early 20th century
  - Kurt Koffka, Max Wertheimer, and Wolfgang Köhler

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features

- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have “327”? No. I have sky, house, and trees."

Max Wertheimer (1880-1943)
Not grouped

Proximity

Similarity

Similarity

Common Fate

Common Region

Slide credit: B. Freeman and A. Torralba
Parallelism

Symmetry

Continuity

Closure

Familiarity

Slide credit: B. Freeman and A. Torralba
Similarity


Slide credit: K. Grauman
Symmetry


Slide credit: K. Grauman
Common fate

Image credit: Arthus-Bertrand (via F. Durand)

Slide credit: K. Grauman
Proximity
Familiarity

Slide credit: B. Freeman and A. Torralba
Emergence

http://en.wikipedia.org/wiki/Gestalt_psychology

Slide credit: S. Lazebnik
Gestalt cues

- Good intuition and basic principles for grouping
- Basis for many ideas in segmentation and occlusion reasoning
- Some (e.g., symmetry) are difficult to implement in practice
Segmentation methods

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-theoretic segmentation
  – Min cut
  – Normalized cuts

• Boundary detection
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 $c_i$:

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

1. Say “Every point is its own cluster”
Agglomerative clustering

1. Say "Every point is its own cluster"

2. Find "most similar" pair of clusters
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster

Copyright © 2001, 2004, Andrew W. Moore

K-means and Hierarchical Clustering: Slide 42

Slide credit: D. Hoiem
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Common similarity/distance measures

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

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

\[
\|x\|_1 := \sum_{i=1}^{n} |x_i|
\]

\[
\|x\| := \sqrt{x_1^2 + \cdots + x_n^2}
\]

\[
\|x\|_\infty := \max(|x_1|, \ldots, |x_n|)
\]

- **Mahalanobis**
  - Scaled Euclidean

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

- **Cosine distance**

\[
similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}
\]

Here \(x_i\) is the distance btw. two points

Slide credit: D. Hoiem
Dendograms

Dendogram formed by agglomerative clustering using single-link clustering.

Slide credit: B. Freeman
Agglomerative clustering

How to define cluster similarity?
- Average distance between points, maximum distance, minimum distance
- Distance between means or medoids

How many clusters?
- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges
Agglomerative clustering

Good
- Simple to implement, widespread application
- Clusters have adaptive shapes
- Provides a hierarchy of clusters

Bad
- May have imbalanced clusters
- Still have to choose number of clusters or threshold
- Need to use an “ultrametric” to get a meaningful hierarchy
Segmentation methods

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-Theoretic Segmentation
  – Min cut
  – Normalized cuts

• Interactive segmentation

• Boundary detection
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, $c_1, \ldots, c_K$
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:

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

1. Ask user how many clusters they’d like. *(e.g. k=5)*
K-means

1. Ask user how many clusters they’d like.  
   \textit{(e.g. } k=5 \text{)}

2. Randomly guess k cluster Center locations
**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)
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!
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
An aside: Smoothing out cluster assignments

- Assigning a cluster label per pixel may yield outliers:

  - How to ensure they are spatially smooth?
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 (I-d)
quantization of the feature space; segmentation label map
Segmentation as clustering

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

Grouping pixels based on color 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.
Segmentation as clustering

K-means clustering using intensity alone and color alone

Slide credit: B. Freeman
Segmentation as clustering

K-means using color alone, 11 segments

Slide credit: B. Freeman
Segmentation as clustering

K-means using color alone, 11 segments.

Color alone often will not yield salient segments!

Slide credit: B. Freeman
Segmentation as clustering

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

Grouping pixels based on intensity + position similarity

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

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

Slide credit: K Grauman
Texture representation example

Windows with primarily horizontal edges

Windows with small gradient in both directions

Windows with primarily vertical edges

Both

Dimension 1 (mean $d/dx$ value)

Dimension 2 (mean $d/dy$ value)

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


Slide credit: K Grauman, L. Lazebnik
Image segmentation example

Texture-based regions

Color-based regions

Slide credit: K Grauman
These look very similar in terms of their color distributions (histograms).

How would their texture distributions compare?
Material classification example

For an image of a single texture, we can classify it according to its global (image-wide) texton histogram.

Figure from Varma & Zisserman, IJCV 2005
Material classification example

*Nearest neighbor* classification: label the input according to the nearest known example’s label.

\[
\chi^2(h_i, h_j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}
\]

Manik Varma
http://www.robots.ox.ac.uk/~vgg/research/texclass/with.html

Slide credit: K Grauman
Segmentation methods

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-theoretic segmentation
  – Min cut
  – Normalized cuts

• Boundary detection
Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation


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

Slide credit: S. Lazebnik
Finding Modes in a Histogram

• How Many Modes Are There?
  – Easy to see, hard to compute
The mean shift algorithm seeks *modes* or local maxima of density in the feature space.
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
Mean shift

Search window

Center of mass

Mean Shift vector

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift

Search window

Center of mass

Mean Shift vector

Slide credit: Y. Ukrainity & B. Sarel
Mean shift

Search window
Center of mass
Mean Shift vector

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass

Mean Shift vector

Slide credit: Y. Ukainitz & B. Sareli
Mean shift

Search window

Center of mass

Mean Shift vector

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift

Search window

Center of mass

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode
Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

Slide credit: S. Lazebnik
Apply mean shift jointly in the image (left col.) and range (right col.) domains

1. Window in image domain
2. Intensities of pixels within image domain window
3. Window in range domain
4. Center of mass of pixels within both image and range domain windows
5. Center of mass of pixels within both image and range domain windows
6. Window in range domain
7. Center of mass of pixels within both image and range domain windows

Slide credit: B. Freeman and A. Torralba
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 \((h_x, h_y) = (8,4)\). (d) Segmentation result.
Mean shift segmentation results

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

Slide credit: S. Lazebnik
More results
More results
Mean shift pros and cons

• Pros
  – Does not assume spherical clusters
  – Just a single parameter (window size)
  – Finds variable number of modes
  – Robust to outliers

• Cons
  – Output depends on window size
  – Computationally expensive
  – Does not scale well with dimension of feature space
Segmentation methods

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts

• 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

Adjacency Matrix

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}
\]

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
Affinity between pixels

Similarities among pixel descriptors

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

\[ \sigma = \text{Scale factor… it will hunt us later} \]
Affinity between pixels

Similarities among pixel descriptors

\[ W_{ij} = \exp\left(-\frac{|| z_i - z_j ||^2}{\sigma^2}\right) \]

Interleaving edges

\[ W_{ij} = 1 - \max \text{ Pb} \]

Line between i and j

With Pb = probability of boundary

\[ \sigma = \text{Scale factor... it will hunt us later} \]

Slide credit: B. Freeman and A. Torralba
Scale affects affinity

- Small $\sigma$: group only nearby points
- Large $\sigma$: group far-away points

Slide credit: S. Lazebnik
Feature grouping by “relocalisation” of eigenvectors of the proximity matrix

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

With an appropriate $\sigma$

$W = \begin{bmatrix}
A & B & C \\
A & 1.00 & 0.63 & 0.03 \\
B & 0.63 & 1.00 & 0.0 \\
C & 0.03 & 0.0 & 1.00
\end{bmatrix}$

The eigenvectors of $W$ are:

<table>
<thead>
<tr>
<th>Eigenvalues</th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.71</td>
<td>-0.01</td>
<td>0.71</td>
</tr>
<tr>
<td>B</td>
<td>-0.71</td>
<td>-0.05</td>
<td>-0.71</td>
</tr>
<tr>
<td>C</td>
<td>-0.04</td>
<td>1.00</td>
<td>-0.03</td>
</tr>
</tbody>
</table>

The first 2 eigenvectors group the points as desired…

Slide credit: B. Freeman and A. Torralba
Example eigenvector

Affinity matrix

Slide credit: B. Freeman and A. Torralba
Example eigenvector
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?
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:

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

with $A \cap B = \emptyset$.

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
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

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

\text{cut}(A,B) is sum of weights with one end in A and one end in B

\[
\text{cut}(A,B) = \sum_{u \in A, v \in B} W(u,v),
\]

with A \cap B = \emptyset

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

\[
\text{assoc}(A,B) = \sum_{u \in A, v \in B} W(u,v)
\]

A and B not necessarily disjoint

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

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

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

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

- Intuitively, the $i$th entry of $y$ can be viewed as a “soft” indication of the component membership of the $i$th feature
  - Can use 0 or median value of the entries as the splitting point (threshold), or find threshold that minimizes the Ncut cost

Normalized cut algorithm

1. Given an image or image sequence, set up a weighted graph \( G = (V, E) \), and set the weight on the edge connecting two nodes being a measure of the similarity between the two nodes.

2. Solve \((D - W)x = \lambda Dx\) 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:

\[ w_{ij} = e^{-\frac{\|F(i) - F(j)\|^2}{\sigma^2}} \times \begin{cases} \frac{-\|X(i) - X(j)\|^2}{\sigma_X^2} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \]

brightness

Location

\[ N \text{ pixels} = ncols \times nrows \]

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.
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

Brightness Image Segmentation


Slide credit: B. Freeman and A. Torralba
Results on color segmentation

Example results
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
Segmentation methods

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts

• 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
Berkeley Segmentation Data Set
David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik

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)
Segmentations are Consistent

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

Perceptual organization forms a tree:

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
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

Image

Boundary Cues

Brightness

Color

Texture

Cue Combination

Model

Pb

Challenges: texture cue, cue combination

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

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

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

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>\( t \))
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
Dataflow

Image → Optimized Cues

Brightness → Color → Texture → Model → Cue Combination → Benchmark

Human Segmentations → Pb

Slide credit: J. Hays
$P_b$ Images
$P_b$ 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