Grouping in vision

- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image or video parts

- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they look similar

- Hard to measure success
  - What is interesting depends on the application
**Image segmentation**

- Goal: identify groups of pixels that go together

**The goals of segmentation**

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

---


---

**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

---
Segmentation is a global process

What are the occluded numbers?

... but not too global

Magritte, 1957

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

* Images from Steve Lehar’s Gestalt papers

---

**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)
Symmetry


Slide credit: K. Grauman

Proximity

http://www.capitol.edu/Resources/Images/outside6_035.jpg

Slide credit: K. Grauman

Common fate

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

Familiarity

Slide credit: B. Freeman and A. Torralba
Familiarity

Slide credit: B. Freeman and A. Torralba

Emergence

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

Slide credit: S. Lazebnik

Grouping phenomena in real life

The buttons on an elevator in the computer science building at U.C. Berkeley

Images: Forsyth and Ponce, Computer Vision: A Modern Approach

Slide credit: K. Grauman

Grouping phenomena in real life

The buttons on an elevator in the computer science building at U.C. Berkeley

Images: Forsyth and Ponce, Computer Vision: A Modern Approach

Slide credit: K. Grauman
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

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts

A simple segmentation technique: Background Subtraction

- If we know what the background looks like, it is easy to identify “interesting bits
- Applications
  - Person in an office
  - Tracking cars on a road
  - Surveillance
- Approach:
  - Use a moving average to estimate background image
  - Subtract from current frame
  - Large absolute values are interesting pixels
  - Trick use morphological operations to clean up pixels

Movie frames from which we want to extract the foreground subject

Images: Forsyth and Ponce, Computer Vision: A Modern Approach
2 different background removal models

Background estimate
Average over frames

Foreground estimate

Foreground estimate

EM background estimate
low thresh
high thresh

Images: Forsyth and Ponce, Computer Vision: A Modern Approach

Slide credit: B. Freeman

Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts

Image segmentation: toy example

- 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?
Now how to determine the three main intensities that define our groups?

• We need to *cluster*.

---

**Segmentation methods**

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts

---

**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

<table>
<thead>
<tr>
<th>Algorithm 15.3: Agglomerative clustering or clustering by merging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make each point a separate cluster</td>
</tr>
<tr>
<td>Until the clustering is satisfactory</td>
</tr>
<tr>
<td>Merge the two clusters with the smallest inter-cluster distance</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 15.4: Divisive clustering or clustering by splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct a single cluster containing all points</td>
</tr>
<tr>
<td>Until the clustering is satisfactory</td>
</tr>
<tr>
<td>Split the cluster that yields the two components with the largest inter-cluster distance</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

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
3. Merge it into a parent cluster

Dendograms

Dendogram formed by agglomerative clustering using single-link clustering.
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

Segmentation methods
- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-Theoretic Segmentation
  - Shortest path
  - Min cut
  - Normalized cuts

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:

\[
\sum_{i} \sum_{\text{points } p \text{ in cluster } i} ||p - c_i||^2
\]
K-means
1. Ask user how many clusters they’d like. (e.g. $k=5$)
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)
4. Each cluster finds the centroid of the points it owns

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
K-means clustering

- Java demo:
  - http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

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 (1-d)

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)

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

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...

Recall: texture representation example

Windows with primarily horizontal edges

Windows with primarily vertical edges

Both

Statistics to summarize patterns in small windows
Segmentation with texture features

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


Image segmentation example


Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts

Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation

Finding Modes in a Histogram

- How Many Modes Are There?
  - Easy to see, hard to compute

Mean shift algorithm
- The mean shift algorithm seeks modes or local maxima of density in the feature space

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 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
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

Mean shift segmentation results

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

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

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts

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

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

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

\( \sigma \) - Scale factor...

it will hunt us later

Affinity Matrix

\[
\begin{bmatrix}
1 & .1 & .3 & 0 & 0 \\
.1 & 1 & .4 & 0 & .2 \\
.3 & .4 & 1 & .6 & .7 \\
0 & 0 & .6 & 1 & 1 \\
0 & .2 & .7 & 1 & 1 \\
\end{bmatrix}
\]

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

Slide credits: B. Freeman and A. Torralba

* From Khurram Hassan-Shafique CAP5455 Computer Vision 2003
Affinity between pixels

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

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

\( \sigma \) - Scale factor...

it will hunt us later

Scale affects affinity
- Small \( \sigma \): group only nearby points
- Large \( \sigma \): group far-away points

Example eigenvector

Three points in feature space

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>0.63</td>
<td>0.69</td>
</tr>
<tr>
<td>0.68</td>
<td>1.00</td>
<td>0.69</td>
</tr>
<tr>
<td>0.63</td>
<td>0.69</td>
<td>1.00</td>
</tr>
</tbody>
</table>

The first 2 eigenvectors group the points as desired...
Example eigenvector

Segmentation methods

• Segment foreground from background
• Histogram-based segmentation
• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation
• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts

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 $A \cap B = \emptyset$.
Minimum cut

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

Minimum cut example

---

Drawbacks of Minimum cut

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

Ideal Cut vs. Cuts with lesser weight than the ideal cut

---

Segmentation methods

• Segment foreground from background
• Histogram-based segmentation
• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation
• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts
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 \( A \).

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

\( A \) and \( B \) not necessarily disjoint


Normalized cut

- Let \( W \) be the adjacency matrix of the graph
- Let \( D \) be the diagonal matrix with diagonal entries
  \[ D(i, i) = \sum_j W(i, j) \]
- Then the normalized cut cost can be written as
  \[ y^T (D - W) y \]
  \[ y^T D y \]

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 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

Example

Affinity:

\[ w_{ij} = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-x_j)^2}{2\sigma^2}} & \text{if } \|X(i)-X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \]

Location

N pixels = ncols * nrows

Figure 12: Subplot (1) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplot (2) - (9) shows the eigenvectors corresponding to 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 200MHz machines. Furthermore, the bottleneck of the computation, a sparse matrix-vector


Slide credit: B. Freeman and A. Torralba

Brightness Image Segmentation


Slide credit: B. Freeman and A. Torralba

Results on color segmentation


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

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

Results: Berkeley Segmentation Engine
http://www.cs.berkeley.edu/~fowlkes/BSE/

Do we need recognition to take the next step in performance?
Top-down segmentation


Motion segmentation