Segmentation & Grouping

[ Slides mostly from Kristen Grauman ]
Outline

• What are grouping problems in vision?

• Inspiration from human perception
  – Gestalt properties

• Bottom-up segmentation via clustering
  – Algorithms:
    • Mode finding and mean shift: k-means, mean-shift
    • Graph-based: normalized cuts
  – Features: color, texture, …
    • Quantization for texture summaries
Grouping in vision

• Goals:
  – Gather features that belong together
  – Obtain an intermediate representation that compactly describes key image or video parts
Examples of grouping in vision

Determine image regions

Group video frames into shots

Object-level grouping

Figure-ground

[Figure by J. Shi]

Grouping video frames into shots

Object-level grouping

[Figure by Wang & Suter]

Figure-ground

[Figure by Grauman & Darrell]
What things should be grouped?
What cues indicate groups?
Gestalt

• 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)
Similarity

Symmetry

Common fate

Image credit: Arthus-Bertrand (via F. Durand)
Proximity
Illusory/subjective contours

Interesting tendency to explain by occlusion

In *Vision*, D. Marr, 1982
Continuity, explanation by occlusion
Figure-ground
Grouping phenomena in real life

Forsyth & Ponce, Figure 14.7
Grouping phenomena in real life

Forsyth & Ponce, Figure 14.7
Gestalt

• Inspiring observations/explanations; challenge remains how to best map to algorithms.
Outline

• What are grouping problems in vision?

• Inspiration from human perception
  – Gestalt properties

• Bottom-up segmentation via clustering
  – Algorithms:
    • Mode finding and mean shift: k-means, mean-shift
    • Graph-based: normalized cuts
  – Features: color, texture, …
    • Quantization for texture summaries
The goals of segmentation

Separate image into coherent “objects”

Source: Lana Lazebnik
The goals of segmentation

Separate image into coherent “objects”

Group together similar-looking pixels for efficiency of further processing

“superpixels”


Source: Lana Lazebnik
Image segmentation: toy example

- Intensities define the three groups.
- What if the image isn’t quite so simple?
input image

pixel count

input image

pixel count
• 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 $c_i$:

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

Source: Steve Seitz
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

Source: Steve Seitz
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. *(e.g. \( k=5 \))*

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
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 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:

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

Source: Steve Seitz
K-means clustering

• Javascript demo:
  http://syskall.com/kmeans.js/

  http://util.io/k-means
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 the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)
quantization of the feature space; segmentation label map
Segmentation as clustering

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

Grouping pixels based on *color* similarity

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

Depending on 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...
Segmentation as clustering

Depending on 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)
Recall: texture representation example

Windows with primarily horizontal edges

Windows with primarily vertical edges

Both

Windows with small gradient in both directions

Windows with primarily vertical edges

Kristen Grauman
Image segmentation example

Texture-based regions

Color-based regions
Pixel properties vs. Texture properties

These look very similar in terms of their color distributions (histograms).

Very different texture

Kristen Grauman
Material classification application

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 application

*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
Outline

• What are grouping problems in vision?

• Inspiration from human perception
  – Gestalt properties

• Bottom-up segmentation via clustering
  – Algorithms:
    • Mode finding and mean shift: k-means, mean-shift
    • Graph-based: normalized cuts
  – Features: color, texture, …
    • Quantization for texture summaries
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
Mean shift algorithm

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

Search window

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass
Mean Shift vector
Mean shift
Mean shift

Search window

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Search window

Slide by Y. Ukrainitz & B. Sarel
Mean shift
Mean shift

Search window
Mean shift

Search window

Slide by 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
Mean shift segmentation results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html
Mean shift segmentation results
Mean shift

• **Pros:**
  – Does not assume shape on clusters
  – One parameter choice (window size)
  – Generic technique
  – Find multiple modes

• **Cons:**
  – Does not scale well with dimension of feature space
Outline

• What are grouping problems in vision?

• Inspiration from human perception
  – Gestalt properties

• Bottom-up segmentation via clustering
  – Algorithms:
    • Mode finding and mean shift: k-means, mean-shift
    • Graph-based: normalized cuts
  – Features: color, texture, …
    • Quantization for texture summaries
Images as graphs

**Fully-connected graph**

- node (vertex) for every pixel
- link between every pair of pixels, $p,q$
- affinity weight $w_{pq}$ for each link (edge)
  
  - $w_{pq}$ measures *similarity*
    
    » similarity is *inversely proportional* to difference (in color and position…)

Source: Steve Seitz
Segmentation by Graph Cuts

Break Graph into Segments

• Want to delete links that cross between segments
• Easiest to break links that have low similarity (low weight)
  – similar pixels should be in the same segments
  – dissimilar pixels should be in different segments

Source: Steve Seitz
Cuts in a graph: Min cut

Link Cut
- set of links whose removal makes a graph disconnected
- cost of a cut:
  \[
  \text{cut}(A, B) = \sum_{p \in A, q \in B} w_{p,q}
  \]

Find minimum cut
- gives you a segmentation
- fast algorithms exist for doing this

Source: Steve Seitz
Minimum cut

- Problem with minimum cut:
  Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]
Cuts in a graph: Normalized cut

Normalized Cut

- fix bias of Min Cut by **normalizing** for size of segments:

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

\[
assoc(A, V) = \text{sum of weights of all edges that touch A}
\]
Cuts in a graph: Normalized cut

Normalized Cut

- fix bias of Min Cut by normalizing for size of segments:

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

\[
\text{assoc}(A, V) = \text{sum of weights of all edges that touch } A
\]

- Prefer two clusters with many strong edges and few weak edges in-between
- Solution: Generalized eigenvalue problem.

Source: Steve Seitz

Example results
Results: Berkeley Segmentation Engine

http://www.cs.berkeley.edu/~fowlkes/BSE/
Normalized cuts: pros and cons

Pros:
• Generic framework, flexible to choice of function that computes weights (“affinities”) between nodes
• Does not require model of the data distribution

Cons:
• Time complexity can be high
  – Dense, highly connected graphs → many affinity computations
  – Solving eigenvalue problem
• Preference for balanced partitions
Top-down segmentation

Top-down segmentation


Slide credit: Lana Lazebnik
Summary

• Segmentation to find object boundaries or mid-level regions, tokens.
• Bottom-up segmentation via clustering
  – General choices -- features, affinity functions, and clustering algorithms
  – Texton histograms for texture distance
• Example clustering methods
  – K-means
  – Mean shift
  – Graph cut, normalized cuts
• Top down with higher level information