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

Group video frames into shots

Determine image regions

Object-level grouping

Figure-ground

Slide credit: Kristen Grauman
Grouping in vision

• Goals:
  – Gather features that belong together
  – Obtain an intermediate representation that compactly describes key image (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 app.
What are meta-cues for grouping?
A Few General Principles
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

Slide credit: Kristen Grauman
Symmetry

Slide credit: Kristen Grauman
Common fate

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

Slide credit: Kristen Grauman
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)

• 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, EM, 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
Clustering

- Clustering algorithms:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. group pixels into regions
  - Useful when don’t know what you’re looking for
  - Requires data, but no labels

Slide credit: Dan Klein
• 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*. 

Slide credit: Kristen Grauman
• 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 $c_i$:

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

Slide credit: Kristen 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.
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:
    $$\sum_{i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$

Source: Steve Seitz
K-Means

- An iterative clustering algorithm

  - Pick K random points as cluster centers (means)
  
  - Alternate:
    
      - Assign data instances to closest mean
    
      - Assign each mean to the average of its assigned points
  
  - Stop when no points’ assignments change
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!
Initialization

- **K-means is non-deterministic**
  - Requires initial means
  - It does matter what you pick!

- What can go wrong?

- Various schemes for preventing this kind of thing
K-Means Getting Stuck

- A local optimum:
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
K-Means Questions

- Will K-means converge?
  - To a global optimum?

- Will it always find the true patterns in the data?
  - If the patterns are very very clear?

- Will it find something interesting?

- How many clusters to pick?

- Do people ever use it?
Probabilistic clustering

Basic questions

- what’s the probability that a point $x$ is in cluster $m$?
- what’s the shape of each cluster?

K-means doesn’t answer these questions

Probabilistic clustering (basic idea)

- Treat each cluster as a Gaussian density function

Slide credit: Steve Seitz
Expectation Maximization (EM)

A probabilistic variant of K-means:

- **E step:** “soft assignment” of points to clusters
  - estimate probability that a point is in a cluster
- **M step:** update cluster parameters
  - mean and variance info (covariance matrix)
- maximizes the likelihood of the points given the clusters
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)
quantization of the feature space; segmentation label map

```matlab
img_as_col = double(im(:));
cluster_membs = kmeans(img_as_col, K);

labelim = zeros(size(im));
for i=1:k
    inds = find(cluster_membs==i);
    meanval = mean(img_as_column(inds));
    labelim(inds) = meanval;
end
```

Slide credit: Kristen Grauman
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: Kristen 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: Kristen 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+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.

Slide credit: Kristen Grauman
Segmentation as clustering

• Color, brightness, position alone are not enough to distinguish all regions…
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)
Recall: 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: Kristen Grauman
Segmentation with texture features

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

Adapted from Lana Lazebnik
Image segmentation example

Texture-based regions

Color-based regions

Slide credit: Kristen Grauman
Pixel properties vs. neighborhood properties

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

How would their *texture* distributions compare?

Slide credit: 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
Recall: 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 clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation

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

Mean shift algorithm

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

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass
Mean Shift vector

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

Search window

Center of mass
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
Mean shift segmentation results
Mean shift segmentation results
Mean shift

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

• **Cons:**
  – Selection of window size
  – 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
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
Measuring affinity

• One possibility:

\[ \text{aff}(x, y) = \exp\left\{-\left(\frac{1}{2}\sigma^2_d\right)(\|x - y\|^2)\right\} \]

Small sigma: group only nearby points

Large sigma: group distant points
Measuring affinity

Data points

Affinity matrices

σ=.1  σ=.2  σ=1

Slide credit: Kristen Grauman
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:

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

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

- \( N\text{cut} \) value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the \( N\text{cut} \) value: generalized eigenvalue problem.

Source: Steve Seitz

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997
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
Summary

• Segmentation to find object boundaries or mid-level regions, tokens.

• Bottom-up segmentation via clustering
  – General choices -- features, affinity functions, and clustering algorithms

• Grouping also useful for quantization, can create new feature summaries
  – Texton histograms for texture within local region

• Example clustering methods
  – K-means
  – Mean shift
  – Graph cut, normalized cuts
Segments as primitives for recognition

B. Russell et al., “Using Multiple Segmentations to Discover Objects and their Extent in Image Collections,” CVPR 2006
Top-down segmentation


Slide credit: Lana Lazebnik
Top-down segmentation


Slide credit: Lana Lazebnik
Motion segmentation

Image grouping