Edges vs Segments
Edges vs Segments

• Edges
  – More low-level
  – Don’t need to be closed

• Segments
  – Ideally one segment for each semantic group/object
  – Should include closed contours
Edge detection

- **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
- **Why?**

  - **Main idea**: look for strong gradients, post-process

Figure from J. Shotton et al., PAMI 2007

Source: K. Grauman
Designing an edge detector

• Criteria for a good edge detector:
  – **Good detection:** find all real edges, ignoring noise or other artifacts
  – **Good localization**
    • detect edges as close as possible to the true edges
    • return one point only for each true edge point

• Cues of edge detection
  – Differences in color, intensity, or texture across the boundary
  – Continuity and closure
  – High-level knowledge

Source: L. Fei-Fei
What causes an edge?

Reflectance change: appearance information, texture

Depth discontinuity: object boundary

Cast shadows

Adapted from K. Grauman
Characterizing edges

• An edge is a place of rapid change in the image intensity function

Source: L. Lazebnik
Intensity profile

Source: D. Hoiem
With a little Gaussian noise

Source: D. Hoiem
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

Where is the edge?

Source: S. Seitz
Effects of noise

• Difference filters respond strongly to noise
  – Image noise results in pixels that look very different from their neighbors
  – Generally, the larger the noise the stronger the response

• What can we do about it?

Source: D. Forsyth
Solution: smooth first

• To find edges, look for peaks in \( \frac{d}{dx}(f \ast g) \)

Source: S. Seitz
Derivative theorem of convolution

• Differentiation is convolution, and convolution is associative:
  \[ \frac{d}{dx} (f * g) = f * \frac{d}{dx} g \]

• This saves us one operation:

Source: S. Seitz
Canny edge detector

• Filter image with derivative of Gaussian
• Find magnitude and orientation of gradient
• Threshold: Determine which local maxima from filter output are actually edges
• Non-maximum suppression:
  – Thin wide “ridges” down to single pixel width
• Linking and thresholding (hysteresis):
  – Define two thresholds: low and high
  – Use the high threshold to start edge curves and the low threshold to continue them

Adapted from K. Grauman, D. Lowe, L. Fei-Fei
Example

input image ("Lena")
Derivative of Gaussian filter

$x$-direction

$y$-direction

Source: L. Lazebnik
Compute Gradients

X-Derivative of Gaussian  Y-Derivative of Gaussian  Gradient Magnitude

Source: D. Hoiem
Thresholding

• Choose a threshold value $t$
• Set any pixels less than $t$ to 0 (off)
• Set any pixels greater than or equal to $t$ to 1 (on)

Source: K. Grauman
The Canny edge detector

norm of the gradient (magnitude)

Source: K. Grauman
The Canny edge detector

thresholding

Source: K. Grauman
Another example: Gradient magnitudes
Thresholding gradient with a lower threshold

Source: K. Grauman
Thresholding gradient with a higher threshold

Source: K. Grauman
The Canny edge detector

How to turn these thick regions of the gradient into curves?

Source: K. Grauman
Non-maximum suppression

• Check if pixel is local maximum along gradient direction, select single max across width of the edge
  – requires checking interpolated pixels p and r

Source: K. Grauman
Bilinear interpolation

\[ f(x, y) \approx [1 - x \ x] \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix}. \]
The Canny edge detector

Problem: pixels along this edge didn’t survive the thresholding

thinning
(non-maximum suppression)

Source: K. Grauman
Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Source: D. Hoiem
Hysteresis thresholding

• Check that maximum value of gradient value is sufficiently large
  – drop-outs? use **hysteresis**
    • use a high threshold to start edge curves and a low threshold to continue them.

Source: S. Seitz, D. Hoiem
Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei
Hysteresis thresholding

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei
Effect of $\sigma$
(Gaussian kernel spread/size)

- original
- Canny with $\sigma = 1$
- Canny with $\sigma = 2$

The choice of $\sigma$ depends on desired behavior
- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Source: S. Seitz
Low-level edges vs. perceived contours

- Berkeley segmentation database:
  
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Source: L. Lazebnik
How can we do better?

So far, we have only considered change in intensity as a cue for the existence of an edge.
pB Boundary Detector

Figure from Fowlkes

Martin, Fowlkes, Malik 2004: Learning to Detection Natural Boundaries…
pB Boundary Detector

Figure from Fowlkes
Brightness
Color
Texture
Combined
Human
Results

Human (0.95)

Pb (0.88)

Source: D. Hoiem
Results

Human (0.96)  Pb (0.88)

Source: D. Hoiem
Results

Source: D. Hoiem
Results

For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html
The goals of segmentation

- Separate image into coherent “objects”
The goals of segmentation

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

“superpixels”


Source: L. Lazebnik
Types of segmentations

Oversegmentation

Undersegmentation

Multiple Segmentations

Source: D. Hoiem
Major processes for segmentation

- **Bottom-up**: group tokens with similar features
- **Top-down**: group tokens that likely belong to the same object

Source: D. Hoiem

[Levin and Weiss 2006]
Examples of grouping in vision

Determine image regions

Group video frames into shots

Object-level grouping

Figure-ground

Source: K. Grauman
Grouping in vision

• Goals:
  – Gather features that belong together
  – Obtain an intermediate representation that compactly describes key image (video) parts

• Hard to measure success
  – What is interesting depends on the application

Adapted from K. Grauman
Segmentation and grouping

• Inspiration from human perception
  – Gestalt properties

• Bottom-up segmentation via clustering
  – Features: color, texture, ...
  – Algorithms:
    • Mode and density finding: k-means, mean-shift
    • Graph-based: normalized cuts

Adapted from K. Grauman
Gestalt cues for grouping

• Gestalt (psychology):
  – Whole is greater than sum of its parts
  – Relationships among parts yield additional information

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

• Good intuition and basic principles for grouping
  – Some are difficult to implement in practice

Adapted from K. Grauman, D. Hoiem
Gestaltism: We perceive the interpretation

Source: K. Grauman
Gestaltism: We perceive the interpretation

The Muller-Lyer illusion

Source: D. Hoiem
Gestaltism: We perceive the interpretation
Principles of perceptual organization

Not grouped

Proximity

Similarity

Similarity

Common Fate

Common Region

Source: D. Hoiem

From Steve Lehar: The Constructive Aspect of Visual Perception
Similarity
Common fate

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

Source: K. Grauman
Proximity
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?

Source: K. Grauman
The image shows two input images, each with a histogram indicating the distribution of pixel intensity values. The histograms on the right side of the images represent the pixel counts for different intensity levels. The images are credited to K. Grauman.
• 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 $c_i$:

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

Source: 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 per group.

Source: K. Grauman
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:
    \[
    \sum_{i} \sum_{p \text{ in cluster } i} \|p - c_i\|^2
    \]
K-means

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

Source: A. Moore
K-means

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

2. Randomly guess k cluster Center locations

Source: 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)
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

Source: 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!

Source: A. Moore
K-means converges to a local minimum
K-means: pros and cons

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

Cons/issues
• Setting k?
  – One way: silhouette coefficient
• Sensitive to initial centers
  – Use heuristics or output of another method
• Sensitive to outliers
• Detects spherical clusters

Adapted from K. Grauman
An aside: Smoothing out cluster assignments

- Assigning a cluster label per pixel may yield outliers:

  - How to ensure they are spatially smooth?

Source: 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)

Source: K. Grauman
quantization of the feature space; segmentation label map

Source: 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 **color** similarity

Feature space: color value (3-d)

Source: 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.

Source: 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+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.

Source: K. Grauman
Segmentation as clustering

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

Source: L. Lazebnik
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)
Segmentation w/ texture features

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

Source: L. Lazebnik

Malik, Belongie, Leung and Shi, IJCV 2001
Image segmentation example

Texture-based regions

Color-based regions

Source: K. Grauman
K-means: pros and cons

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

**Cons/Issues**
- Setting k?
  - One way: silhouette coefficient
- Sensitive to initial centers
  - Use heuristics or output of another method
- Sensitive to outliers
- Detects spherical clusters

Adapted from K. Grauman
Mean shift algorithm

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

Source: K. Grauman
Kernel density estimation

Kernel

Estimated density

Data (1-D)

Source: D. Hoiem
Mean shift
Mean shift

Search window
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

- Search window
- Center of mass
- Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass
Mean Shift vector
Mean shift

- Search window
- Center of mass
- Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass

Slide by Y. Ukrainitz & B. Sarel
Computing the mean shift

Simple Mean Shift procedure:
• Compute mean shift vector
• Translate the Kernel window by $m(x)$

$m(x) = \frac{\sum_{i=1}^{n} x_i K(x_i - x)}{\sum_{i=1}^{n} K(x_i - x)} - x$

$K(x_i - x) = e^{-\frac{||x_i-x||^2}{\sigma}}$

Adapted from Y. Ukrainitz & B. Sarel
Points in same cluster converge
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

- Compute features for each point (color, 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

Source: D. Hoiem
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
- Robust to outliers

**Cons/Issues:**
- Need to choose window size
- Does not scale well with dimension of feature space
  - Search for neighbors could be sped up in lower dimensions *

* http://lear.inrialpes.fr/people/triggs/events/iccv03/cdrom/iccv03/0456_georgescu.pdf
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
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
Cuts in a graph: Min 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.
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) = sum of weights of all edges that touch A

- Ncut 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 Ncut value: generalized eigenvalue problem

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

Source: Steve Seitz
So far, we discussed detecting low-level edges and contours, and grouping similar content via clustering.

Finally, a few segmentation methods from recent conferences...
Hypercolumns for Object Segmentation and Fine-grained Localization (Hariharan et al., CVPR 2015)

- Create a CNN-based representation for each pixel
- Which CNN level to use? All of them!
  - Stack together activations from all layers above that pixel
  - Upsample feature maps as needed
Hypercolumns for Object Segmentation and Fine-grained Localization (Hariharan et al., CVPR 2015)

- Classify each pixel via a linear classifier (logistic regression)
- Train a grid of location-specific classifiers
- The prediction at a pixel is a linear combination of the nearby classifiers:

\[ h_i(\cdot) = \sum_k \alpha_{ik} g_k(\cdot) \]
Hypercolumns for Object Segmentation and Fine-grained Localization (Hariharan et al., CVPR 2015)

• Results:

<table>
<thead>
<tr>
<th>Metric</th>
<th>T-Net Only fc7</th>
<th>T-Net Hyp</th>
<th>O-Net Only fc7</th>
<th>O-Net Hyp</th>
<th>O-Net Hyp+ Rescore</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP$r$ at 0.5</td>
<td>44.0</td>
<td>49.1</td>
<td>52.6</td>
<td>56.5</td>
<td>60.0</td>
</tr>
<tr>
<td>mAP$r$ at 0.7</td>
<td>16.3</td>
<td>29.1</td>
<td>22.4</td>
<td>37.0</td>
<td>40.4</td>
</tr>
</tbody>
</table>

Figure 4. Figure ground segmentations starting from bounding box detections. Top row: baseline using fc7, bottom row: Ours.
Fully Convolutional Networks for Semantic Segmentation (Long et al., CVPR 2015)

- Adapt classification nets to produce dense outputs
- Combine info from different layers

<table>
<thead>
<tr>
<th></th>
<th>pixel acc.</th>
<th>mean acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-32s-fixed</td>
<td>83.0</td>
<td>59.7</td>
</tr>
<tr>
<td>FCN-32s</td>
<td>89.1</td>
<td>73.3</td>
</tr>
<tr>
<td>FCN-16s</td>
<td>90.0</td>
<td>75.7</td>
</tr>
<tr>
<td>FCN-8s</td>
<td><strong>90.3</strong></td>
<td><strong>75.9</strong></td>
</tr>
</tbody>
</table>

- FCN-8s
- SDS [15]
- Ground Truth
- Image
Segmentation from Natural Language Expressions
(Hu et al., ECCV 2016)

- Segment out regions requested via language (not just any regions for a particular object class)