Last time: Grouping

- Bottom-up segmentation via clustering
  - To find mid-level regions, tokens
  - General choices -- features, affinity functions, and clustering algorithms
  - Example clustering algorithms
    - Mean shift and mode finding: K-means, Mean shift
    - Graph theoretic: Graph cut, normalized cuts
- Grouping also useful for quantization
  - Texton histograms for texture within local region

Recall: Images as graphs

- Fully-connected graph
  - node for every pixel
  - link between every pair of pixels, \( p, q \)
  - similarity \( w_{pq} \) for each link
  
Slide credits:
- Kristen Grauman
- Steve Seitz
Last time: Measuring affinity

1. What do the blocks signify?
2. What does the symmetry of the matrix signify?
3. How would the matrix change with larger value of $\sigma$?

Example: weighted graphs

- Suppose we have a 4-pixel image (i.e., a 2 x 2 matrix)
- Each pixel described by 2 features

Computing the distance matrix:

```matlab
d = 2
N = 4
D = zeros(N, N);
for i = 1:N
    D(i,:) = vecnorm(x(i,:) - x,:2);
end
D
```

Example: weighted graphs

Dimension of data points : $d = 2$
Number of data points : $N = 4$
for \( i = 1 : N \)
for \( j = 1 : N \)
\[ D(i,j) = ||x_i - x_j||^2 \]
end
end

\[ D(1,:) = \begin{bmatrix} 0.24 & 0.01 & 0.47 \end{bmatrix} \]
\[ D(:,1) = \begin{bmatrix} 0.15 & 0.24 & 0.29 \end{bmatrix} \]

Example: weighted graphs

Computing the distance matrix:

\[ A(i,j) = \exp(-1/(2\sigma^2)||x_i - x_j||^2); \]
\[ A(j,i) = A(i,j); \]
end
end

Example: weighted graphs

Distances \( \rightarrow \) affinities
Scale parameter $\sigma$ affects affinity

Distance matrix $D$:

Affinity matrix with increasing $\sigma$:

Visualizing a shuffled affinity matrix

If we permute the order of the vertices as they are referred to in the affinity matrix, we see different patterns:

Putting these two aspects together

$$A(i,j) = \exp\left(-\frac{1}{2\sigma^2}||x_i - x_j||^2\right)$$
Goal: Segmentation by Graph Cuts

Break graph into segments
- Delete links that cross between segments
  - Easiest to break links that have low similarity
    - Similar pixels should be in the same segments
    - Dissimilar pixels should be in different segments

Now: Fitting

- Want to associate a model with multiple observed features
  - For example, the model could be a line, a circle, or an arbitrary shape.

Fitting: Main idea

- Choose a parametric model that best represents a set of features
- Membership criterion is not local
  - Can’t tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features
Example: Line fitting

• Why fit lines?
  Many objects characterized by presence of straight lines

• Wait, why aren’t we done just by running edge detection?

Difficulty of line fitting

• Extra edge points (clutter), multiple models:
  – which points go with which line, if any?

• Only some parts of each line detected, and some parts are missing:
  – how to find a line that bridges missing evidence?

• Noise in measured edge points, orientations:
  – how to detect true underlying parameters?

Voting

• It’s not feasible to check all combinations of features by fitting a model to each possible subset.

• Voting is a general technique where we let each feature vote for all models that are compatible with it.
  – Cycle through features, cast votes for model parameters.
  – Look for model parameters that receive a lot of votes.

• Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.
Fitting lines: Hough transform

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

Hough Transform is a voting technique that can be used to answer all of these questions.

Main idea:
1. Record vote for each possible line on which each edge point lies.
2. Look for lines that get many votes.

Finding lines in an image: Hough space

Equation of a line:
\[ y = mx + b \]

Connection between image \((x,y)\) and Hough \((m,b)\) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points \((x,y)\), find all \((m,b)\) such that \(y = mx + b\)
- What does a point \((x_0, y_0)\) in the image space map to?
- Answer: the solutions of \(b = -x_0m + y_0\)
  - this is a line in Hough space
Finding lines in an image: Hough space

What are the line parameters for the line that contains both $(x_0, y_0)$ and $(x_1, y_1)$?

- It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$.

Finding lines in an image: Hough algorithm

How can we use this to find the most likely parameters $(m, b)$ for the most prominent line in the image space?

- Let each edge point in image space vote for a set of possible parameters in Hough space.
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Polar representation for lines

Issues with usual $(m, b)$ parameter space: can take on infinite values, undefined for vertical lines.

$$d : \text{perpendicular distance from line to origin}$$

$$\theta : \text{angle the perpendicular makes with the x-axis}$$

$$x \cos \theta + y \sin \theta = d$$

Point in image space $\rightarrow$ sinusoid segment in Hough space.
Hough transform algorithm

Using the polar parameterization:

\[ x \cos \theta + y \sin \theta = d \]

Basic Hough transform algorithm

1. Initialize \( H[d, \theta] = 0 \)
2. for each edge point \((x, y)\) in the image
   for \( \theta = [\theta_{\text{min}} \text{ to } \theta_{\text{max}} ] \) // some quantization
     \[ d = x \cos \theta + y \sin \theta \]
     \( H[d, \theta] += 1 \)
3. Find the value(s) of \((d, \theta)\) where \( H[d, \theta]\) is maximum
4. The detected line in the image is given by \( d = x \cos \theta + y \sin \theta \)

Time complexity (in terms of number of votes per pt)?

Source: Steve Seitz

1. Image \( \rightarrow \) Canny

Derek Hoiem
2. Canny $\rightarrow$ Hough votes

Derek Hoiem

3. Hough votes $\rightarrow$ Edges

Find peaks

Derek Hoiem

Hough transform example

Derek Hoiem

http://ostatic.com/files/images/ss_hough.jpg
Here, everything appears to be "noise", or random edge points, but we still see peaks in the vote space. What difficulty does this present for an implementation?
Extensions

Recall: when we detect an edge point, we also know its gradient direction.

Extension 1: Use the image gradient
1. same
2. For each edge point $[x,y]$ in the image:
   \[ d = x \cos \theta + y \sin \theta \]
   \[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
   \[ h[d, \theta] += 1 \]
3. same
4. same
(Reduces degrees of freedom)

Extension 2
• give more votes for stronger edges (use magnitude of gradient)

Extension 3
• change the sampling of $(d, \theta)$ to give more/less resolution

Extension 4
• The same procedure can be used with circles, squares, or any other shape...

Source: Steve Seitz

Hough transform for circles

• Circle: center $(a, b)$ and radius $r$
  \[(x - a)^2 + (y - b)^2 = r^2\]

• For a fixed radius $r$

Equation of circle?

Equation of set of circles that all pass through a point?
Hough transform for circles

- Circle: center (a, b) and radius r
  \[(x - a)^2 + (y - b)^2 = r^2\]

- For a fixed radius r

![Diagram of Hough space and image space showing circles and votes]

Intersection: most votes for center occur here.

Kristen Grauman

Hough transform for circles

- Circle: center (a, b) and radius r
  \[(x - a)^2 + (y - b)^2 = r^2\]

- For an unknown radius r

![Diagram of Hough space and image space showing circles and votes]

Kristen Grauman

Hough transform for circles

- Circle: center (a, b) and radius r
  \[(x - a)^2 + (y - b)^2 = r^2\]

- For an unknown radius r

![Diagram of Hough space and image space showing circles and votes]

Kristen Grauman
Hough transform for circles

- Circle: center \((a,b)\) and radius \(r\)
  
  \[
  (x_{i} - a)^2 + (y_{i} - b)^2 = r^2
  \]

- For an unknown radius \(r\), known gradient direction

```
Hough transform for circles

For every edge pixel \((x,y)\):
  For each possible radius value \(r\):
    For each possible gradient direction \(\theta\):
      \[
      \begin{align*}
      a &= x - r \cos(\theta) \quad // \text{column} \\
      b &= y + r \sin(\theta) \quad // \text{row} \\
      H[a,b,r] &=+1
      \end{align*}
      \]
      Time complexity per edge pixel?
  
  • Check out online demo: [http://www.markschulze.net/java/hough/](http://www.markschulze.net/java/hough/)

Example: detecting circles with Hough

Original | Edges | Votes: Penny
---|---|---

Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Coin finding sample images from: Vivek Kwatra
Example: detecting circles with Hough

- Combined detections
- Edges
- Votes: Quarter

Coin finding sample images from: Vivek Kwatra

Example: iris detection

- Gradient+threshold
- Hough space (fixed radius)
- Max detections


Example: iris detection

Voting: practical tips

• Minimize irrelevant tokens first
• Choose a good grid / discretization
  ![Too fine](Too fine) ![Too coarse](Too coarse)
• Vote for neighbors, also (smoothing in accumulator array)
• Use direction of edge to reduce parameters by 1

Hough transform: pros and cons

Pros
• All points are processed independently, so can cope with occlusion, gaps
• Some robustness to noise: noise points unlikely to contribute consistently to any single bin
• Can detect multiple instances of a model in a single pass

Cons
• Complexity of search time increases exponentially with the number of model parameters
• Non-target shapes can produce spurious peaks in parameter space
• Quantization: can be tricky to pick a good grid size

Generalized Hough Transform

• What if we want to detect arbitrary shapes?

Intuition:

Now suppose those colors encode gradient directions…
Define a model shape by its boundary points and a reference point.

**Offline procedure:**
At each boundary point, compute displacement vector: \( \mathbf{r} = \mathbf{a} - \mathbf{p}_i \).

Store these vectors in a table indexed by gradient orientation \( \theta \).

**Detection procedure:**
For each edge point:
- Use its gradient orientation \( \theta \) to index into stored table
- Use retrieved \( \mathbf{r} \) vectors to vote for reference point

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

**Generalized Hough for object detection**
- Instead of indexing displacements by gradient orientation, index by matched local patterns.

"visual codeword" with displacement vectors

Source: L. Lazebnik
Instead of indexing displacements by gradient orientation, index by "visual codeword".

B. Leibe, A. Leonardis, and B. Schiele, 
Combined Object Categorization and Segmentation with an Implicit Shape Model. 
ECCV Workshop on Statistical Learning in Computer Vision 2004.

---

Summary

- **Fitting** problems require finding any supporting evidence for a model, even within clutter and missing features
  - associate features with an explicit model

- **Voting** approaches, such as the **Hough transform**, make it possible to find likely model parameters without searching all combinations of features
  - Hough transform approach for lines, circles, ..., arbitrary shapes defined by a set of boundary points, recognition from patches

---

Questions?

See you Tuesday!