Fitting: Voting and the Hough Transform

Thurs Sept 24
Kristen Grauman
UT Austin

Last time

• 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...)
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:
  \[ \text{cut}(A, B) = \sum_{p \in A, q \in B} w_{pq} \]

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

Source: Steve Seitz

Measuring affinity for edge weights

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

Small sigma: group only nearby points
Large sigma: group distant points
Measuring affinity for edge weights

Data points

Affinity matrices

Cuts in a graph: Min cut

Link Cut
• set of links whose removal makes a graph disconnected
• cost of a cut: $cut(A, B) = \sum_{p=1, q=0}^{P} w_{p,q}$

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

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

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

- 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.

Example results

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 \( \rightarrow \) many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions
Segments as primitives for recognition

Top-down segmentation

E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," ECCV 2002

E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," ECCV 2002
Motion segmentation


Image grouping


Recap on grouping

- 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 (and EM)
  - Mean shift
  - Graph cut, normalized cuts
Now: Fitting

• Want to associate a model with observed features

For example, the model could be a line, a circle, or an arbitrary shape.

Fitting: Main idea

• Choose a parametric model to represent 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 the features 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

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

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.

- \(d\) : perpendicular distance from line to origin
- \(\theta\) : 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 \(\theta = \theta_{min}\) to \(\theta_{max}\) // some quantization
   - \(d = x \cos \theta - y \sin \theta\)
   - \(H[d,\theta]++\)
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)?
Impact of noise on Hough

What difficulty does this present for an implementation?
Impact of noise on Hough

Here, everything appears to be "noise", or random edge points, but we still see peaks in the vote space.

Extensions

Extension 1: Use the image gradient
1. same
2. for each edge point \((x,y)\) in the image
   \[ d = \text{gradient at } (x,y) \]
   \[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial x} / \frac{\partial f}{\partial y} \right) \]
   \[ H[d, \theta] += 1 \]
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 \(r\)
  
  \[ (x_i - a)^2 + (y_i - b)^2 = r^2 \]

• For a fixed radius \(r\), unknown gradient direction

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 \(r\), unknown gradient direction

Intersection: most votes for center occur here.
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 \(r\), unknown gradient direction

\[
(x_i - a)^2 + (y_i - b)^2 = r^2
\]

\[
(x - r \cos(\theta))^2 + (y + r \sin(\theta))^2 = r^2
\]

Hough space

Image space

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

\[
(x - r \cos(\theta))^2 + (y + r \sin(\theta))^2 = r^2
\]

Hough space

Image space

Hough transform for circles

For every edge pixel \((x, y)\):
  For each possible radius value \(r\):
    For each possible gradient direction \(\theta\):
      // or use estimated gradient at \((x, y)\)
      \[a = x - r \cos(\theta)\] // column
      \[b = y + r \sin(\theta)\] // row
      \[H[a, b, r] += 1\]
    end
  end
end

Time complexity per edge?

- Check out online demo: http://www.markschulze.net/java/hough/
Example: detecting circles with Hough

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

Example: detecting circles with Hough

Coin finding sample images from Vivek Kwatra

Example: iris detection

• Hemerson Pistori and Eduardo Rocha Costa
Example: iris detection


Voting: practical tips

- Minimize irrelevant tokens first
- Choose a good grid / discretization
- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes.

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…

Generalized Hough Transform

• Define a model shape by its boundary points and a reference point.

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

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

Generalized Hough Transform

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.

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

Source: L. Lazebnik

Summary

• Grouping/segmentation useful to make a compact representation and merge similar features
  – associate features based on defined similarity measure and clustering objective

• 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.
Coming up

Fitting with deformable contours
A2 is out, due in two weeks