Fitting:
Deformable contours

Thursday, Sept 24
Kristen Grauman
UT-Austin

Announcements

• Next week: guest lectures
  – Tuesday: Background modeling
  – Thursday: Image formation

• Yong Jae and I are not available for office hours next week. Jaechul is available as usual.

Announcements

• Matlab issues: ask us about Matlab coding problems.
  – e.g., “How do I remove a different pixel from each row? When I try to delete them this way (XYZ), I get a size error…”
  – (but not: “What does the function imfilter do?”)

• Check the functions listed in the psets
  – help <function name>

Some seam carving results from Pset 1
Results from Michael Yao

- Left, "chinese_opera.jpg" [768x600], Original
- Top Right, "chinese_opera-dumb-resize.jpg" [500x768], Regular Resize
- Bottom Right, "chinese_opera-seamcarving-resize.jpg" [500x768], Content Aware Resize

Results from Eunho Yang

Results from Larry Lindsey

Conventional Size History of Seam-Carved Image

Results from Donghyuk Shin

Removal of a marked object

This example shows a hue-based skin detector works well to preserve the face in seam carving. However, we can see the body is largely removed, producing a undesirable artifact in the proportion between face and body.

Results from Donghyuk Shin

Seam carving using a gradient energy [500 by 268]

Results from Michael Fairley

Seam carving in an image with a regular texture pattern.

*This example shows a failure case of seam carving.
Last time: Fitting with “voting”

- Hough transform for fitting lines, circles

\[ y = mx + b \]

where

\((x_0, y_0)\) and \((x_1, y_1)\) are points in image space,
\(m\) and \(b\) are parameters in Hough space.

Grouping and Fitting

Goal: move from array of pixel values (or filter outputs) to a collection of regions, objects, and shapes.

Pixels vs. regions

By grouping pixels based on Gestalt-inspired attributes, we can map the pixels into a set of regions.

Each region is consistent according to the features and similarity metric we used to do the clustering.

Edges vs. boundaries

Edges useful signal to indicate occluding boundaries, shape.

Here the raw edge output is not so bad…

…but quite often boundaries of interest are fragmented, and we have extra “clutter” edge points.
Edges vs. boundaries

Given a model of interest, we can overcome some of the missing and noisy edges using fitting techniques.

With voting methods like the Hough transform, detected points vote on possible model parameters.

Previously, we focused on the case where a line or circle was the model…

Today

• Fitting an arbitrary shape model with Generalized Hough Transform
• Fitting an arbitrary shape with “active” deformable contours

Generalized Hough transform

• What if want to detect arbitrary shapes defined by boundary points and a reference point?

At each boundary point, compute displacement vector: \( r = a - p \). For a given model shape: store these vectors in a table indexed by gradient orientation \( \theta \).

To detect the model shape in a new image:

• For each edge point – Index into table with its gradient orientation \( \theta \)
  – Use retrieved \( r \) vectors to vote for position of reference point
• Peak in this Hough space is reference point with most supporting edges

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

Example

Say we’ve already stored a table of displacement vectors as a function of edge orientation for this model shape.

Example

displacement vectors for model points

Adapted from Lena Lazebnik
Now we want to look at some edge points detected in a new image, and vote on the position of that shape.

Example

votes for points with $\theta = \uparrow$

range of voting locations for test point

Example

Recall: displacement vectors for model points

Example

Example
Application of Generalized Hough for recognition

- 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

Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object
Goal: evolve the contour to fit exact object boundary

Main idea: elastic band is iteratively adjusted so as to
- be near image positions with high gradients, and
- satisfy shape “preferences” or contour priors

Deformable contours: intuition

Like generalized Hough transform, useful for shape fitting; but

Hough
- Rigid model shape
- Single voting pass can detect multiple instances

Deformable contours
- Prior on shape types, but shape iteratively adjusted (deforms)
- Requires initialization nearby
- One optimization “pass” to fit a single contour
Why do we want to fit deformable shapes?

- Some objects have similar basic form but some variety in the contour shape.

Why do we want to fit deformable shapes?

- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands...

Figure from Kass et al. 1987

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

Representation

- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions ("vertices").
- At each iteration, we'll have the option to move each vertex to another nearby location ("state").

\[ \mathbf{v}_i = (x_i, y_i), \quad \text{for } i = 0, 1, \ldots, n-1 \]
Fitting deformable contours

How should we adjust the current contour to form the new contour at each iteration?

• Define a cost function ("energy" function) that says how good a candidate configuration is.
• Seek next configuration that minimizes that cost function.

Initial | Intermediate | Final

Energy function

The total energy (cost) of the current snake is defined as:

\[ E_{total} = E_{internal} + E_{external} \]

**Internal** energy: encourage prior shape preferences: e.g., smoothness, elasticity, particular known shape.

**External** energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour and the target shape in the image will yield a low value for this cost function.

External energy: intuition

• Measure how well the curve matches the image data
• "Attract" the curve toward different image features – Edges, lines, texture gradient, etc.

External image energy

• Measure how well the curve matches the image data
• "Attract" the curve toward different image features – Edges, lines, texture gradient, etc.

External image energy

• Gradient images \( G_x(x, y) \) and \( G_y(x, y) \)

• External energy at a point on the curve is:

\[ E_{external}(v) = -(G_x(v))^2 + (G_y(v))^2 \]

• External energy for the whole curve:

\[ E_{external} = \sum_{i=0}^{n-1} \left[ (G_x(x_i, y_i))^2 + (G_y(x_i, y_i))^2 \right] \]
Internal energy: intuition

A priori, we want to favor smooth shapes, contours with low curvature, contours similar to a known shape, etc. to balance what is actually observed (i.e., in the gradient image).

Internal energy

For a continuous curve, a common internal energy term is the "bending energy".

At some point $v(s)$ on the curve, this is:

$$E_{\text{internal}}(v(s)) = \alpha \left( \frac{dv}{ds} \right)^2 + \beta \left( \frac{d^2v}{ds^2} \right)^2$$

Internal energy

• For our discrete representation,

$$v_i = (x_i, y_i) \quad i = 0, ..., n - 1$$

$$\frac{d^i v}{ds^i} = v_{i+1} - v_i \quad \frac{d^2 v}{ds^2} = (v_{i+1} - v_i) - (v_i - v_{i-1}) = v_{i+1} - 2v_i + v_{i-1}$$

• Internal energy for the whole curve:

$$E_{\text{internal}} = \sum_{i=0}^{n-1} \alpha \|v_{i+1} - v_i\|^2 + \beta \|v_{i+1} - 2v_i + v_{i-1}\|^2$$

Why do these reflect tension and curvature?

Penalizing elasticity

• Current elastic energy definition uses a discrete estimate of the derivative:

$$E_{\text{elastic}} = \sum_{i=0}^{n-1} \alpha \|v_{i+1} - v_i\|^2$$

$$= \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$

What is the possible problem with this definition?

Penalizing elasticity

• Current elastic energy definition uses a discrete estimate of the derivative:

$$E_{\text{elastic}} = \sum_{i=0}^{n-1} \alpha \|v_{i+1} - v_i\|^2$$

Instead:

$$= \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 - d^2$$

where $d$ is the average distance between pairs of points – updated at each iteration.

Dealing with missing data

• The preferences for low-curvature, smoothness help deal with missing data:

Illusory contours found!
Extending the internal energy: capture shape prior

- If object is some smooth variation on a known shape, we can use a term that will penalize deviation from that shape:
  \[ E_{\text{internal}} + = \alpha \cdot \sum_{i=0}^{n-1} (v_i - \hat{v}_i)^2 \]
  where \( \{\hat{v}_i\} \) are the points of the known shape.

Total energy

\[ E_{\text{total}} = E_{\text{internal}} + \gamma E_{\text{external}} \]

\[ E_{\text{external}} = -\sum_{i=0}^{n-1} G_x(x_i, y_i) + \sqrt{G_y(x_i, y_i)} \]

\[ E_{\text{internal}} = \sum_{i=0}^{n-1} \alpha \left( \|v_i - v_{i+1}\|^2 + \beta \|v_i - 2v_j + v_{i-1}\|^2 \right) \]

Function of the weights

- e.g., \( \alpha \) weight controls the penalty for internal elasticity

   large \( \alpha \)  medium \( \alpha \)  small \( \alpha \)

Recap: deformable contour

- A simple elastic snake is defined by:
  - A set of \( n \) points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)

- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy

Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We’ll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

Energy minimization: greedy

- For each point, search window around it and move to where energy function is minimal
  - Typical window size, e.g., 5 x 5 pixels

- Stop when predefined number of points have not changed in last iteration, or after max number of iterations

- Note:
  - Convergence not guaranteed
  - Need decent initialization
Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We’ll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

Energy minimization: dynamic programming

With this form of the energy function, we can minimize using dynamic programming, with the Viterbi algorithm.

Iterate until optimal position for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.

Snake energy: pair-wise interactions

$$E_{total}(v_1, \ldots, v_n) = -\sum_{i=1}^{n-1} E_i(v_i, v_{i+1})$$

Or sum of triple-interaction potentials.

$$E_{total}(v_1, \ldots, v_n) = \sum_{i=1}^{n-1} E_i(v_{i-1}, v_i, v_{i+1})$$

Viterbi algorithm

Main idea: determine optimal position (state) of predecessor, for each possible position of self. Then backtrack from best state for last vertex.

Energy minimization: dynamic programming

Complexity: $O(nm^2)$ vs. brute force search ____?
Energy minimization: dynamic programming

DP can be applied to optimize an open ended snake

\[ E_1(v_1, v_2) + E_2(v_2, v_3) + ... + E_{n-1}(v_{n-1}, v_n) \]

For a closed snake, a “loop” is introduced into the total energy.

\[ E_s(v_s, v_i) + E_1(v_1, v'_2) + ... + E_{n-2}(v_{n-1}, v'_n) + E_n(v'_n, v_s) \]

Work around:
1) Fix \( v_s \) and solve for rest.
2) Fix an intermediate node at its position found in (1), solve for rest.

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

Tracking via deformable contours

1. Use final contour/model extracted at frame \( t \) as an initial solution for frame \( t+1 \)
2. Evolve initial contour to fit exact object boundary at frame \( t+1 \)
3. Repeat, initializing with most recent frame.

Limitations

- May over-smooth the boundary
- Cannot follow topological changes of objects
Limitations

• External energy: snake does not really "see" object boundaries in the image unless it gets very close to it.

Distance transform

• External image can instead be taken from the distance transform of the edge image.

Deformable contours: pros and cons

Pros:
• Useful to track and fit non-rigid shapes
• Contour remains connected
• Possible to fill in "subjective" contours
• Flexibility in how energy function is defined, weighted.

Cons:
• Must have decent initialization near true boundary, may get stuck in local minimum
• Parameters of energy function must be set well based on prior information

Interactive forces

• An energy function can be altered online based on user input – use the cursor to push or pull the initial snake away from a point.
• Modify external energy term to include:

\[ E_{push} = \sum_{i=0}^{n} \left| v_i - p \right|^2 \]

Nearby points get pushed hardest

Interactive forces

• An energy function can be altered online based on user input – use the cursor to push or pull the initial snake away from a point.
• Modify external energy term to include:

\[ E_{push} = \sum_{i=0}^{n} \left| v_i - p \right|^2 \]

Nearby points get pushed hardest

What expression could we use to pull points towards the cursor position?

Intelligent scissors

Another form of interactive segmentation:
Use dynamic programming to compute optimal paths from every point to the seed based on edge-related costs.

What expression could we use to pull points towards the cursor position?
Intelligent scissors

Summary

• Deformable shapes and active contours are useful for
  – Segmentation: fit or “snap” to boundary in image
  – Tracking: previous frame’s estimate serves to initialize the next

• Fitting active contours:
  – Define terms to encourage certain shapes, smoothness, low curvature, push/pulls, …
  – Use weights to control relative influence of each component cost
  – Can optimize 2d snakes with Viterbi algorithm.

• Image structure (esp. gradients) can act as attraction force for interactive segmentation methods.

Recap: mid-level vision
Features → regions, shapes, boundaries

• Segment regions (last Thursday)
  – cluster pixel-level features, like color, texture, position
  – leverage Gestalt properties

• Fitting models (Tuesday)
  – explicit rigid parametric models such as lines and circles, or arbitrary shapes defined by boundary points and reference point
  – voting methods useful to combine grouping of tokens and fitting of parameters; e.g. Hough transform

• Detection of deformable contours, and interactive segmentation (today)
  – provide rough initialization nearby true boundary, or
  – interactive, iterative process where user guides the boundary placement

Coming up

• Tues: Background modeling
  – Read F&P 14.3
  – Stauffer & Grimson paper

• Thurs: Image formation
  – Read F&P Chapter 1

• Pset 1 due Mon 10/5