Active Contours (SNAKES)

- Back to boundary detection
  - This time using perceptual grouping.
- This is non-parametric
  - We’re not looking for a contour of a specific shape.
  - Just a good contour.

For Information on SNAKES

- Not in Forsyth and Ponce.
- See Text by Trucco and Verri, or Shapiro and Stockman.
- Kass, Witkin and Terzopoulos, IJCV.

Sometimes edge detectors find the boundary pretty well.

Sometimes it’s not enough.

Improve Boundary Detection

- Integrate information over distance.
- Use Gestalt cues
  - Smoothness
  - Closure
- Get User to Help.

Humans integrate contour information.
Strategy of Class

- What is a good path?
- Given endpoints, how do we find a good path?
- What if we don’t know the end points?

Note that like all vision this is *modeling* and *optimization*.

We’ll do something easier than finding the whole boundary. Finding the best path between two boundary points.

Discrete Grid

- Contour should be near edge.
- Strength of gradient.
- Contour should be smooth (good continuation).
- Low curvature
- Low change of direction of gradient.

Review Gradient

*Blackboard: See notes on Class 6 also.*

Smoothness

- Discrete Curvature: if you go from \( p(j-1) \) to \( p(j) \) to \( p(j+1) \) how much did direction change?
  - Be careful with discrete distances.
- Change of direction of gradient from \( p(j-1) \) to \( p(j) \)
Combine into a cost function

- Path: \( p(1), p(2), \ldots, p(n) \).

\[
\sum_{j=1}^{n} d(p(j), p(j+1)) \cdot g(p(j) + \lambda f(p(j-1), p(j))
\]

Where
- \( d(p(j), p(j+1)) \) is distance between consecutive grid points (i.e., 1 or \( \sqrt{2} \)).
- \( g(p(j)) \) measures strength of gradient
- \( \lambda \) is some parameter
- \( f \) measures smoothness, curvature.

One Example cost function

\[
\sum_{j=1}^{n} d(p(j), p(j+1)) \cdot [g(p(j) + \lambda f(p(j-1), p(j))]
\]

\[
g(p(j)) = \frac{1}{l_j + \rho}
\]

\[
l_j = \frac{|\nabla I_j|}{\max|\nabla I|}
\]

\( \rho \) is some constant


Example

So How do we find the best Path? Computer Science at last.

A Curve is a path through the grid.
Cost depends on each step of the path.
We want to minimize cost.

Map problem to Graph

Weight represents cost of going from one pixel to another. Next term in sum.

Dijkstra’s shortest path algorithm

- **Algorithm**
  1. Init node costs to \( \infty \), set \( p = \text{seed point}, \text{cost}(p) = 0 \)
  2. Expand \( p \) as follows:
     - for each of \( p \)'s neighbors \( q \) that are not expanded
       - set \( \text{cost}(q) = \min(\text{cost}(p) + c_{pq}, \text{cost}(q)) \)

(Seitz)
Dijkstra's shortest path algorithm

- Algorithm
  1. init node costs to ∞, set p = seed point, cost(p) = 0
  2. expand p as follows:
     - for each of p's neighbors q that are not expanded
       - set cost(q) = min(cost(p) + c_{pq}, cost(q))
       - if q's cost changed, make q point back to p
     - put q on the ACTIVE list (if not already there)
  3. set r = node with minimum cost on the ACTIVE list
  4. repeat Step 2 for p = r

Application: Intelligent Scissors

Results

Continuous versions

- Can express cost function as continuous.
- Use continuous optimization like gradient descent.
- Level Set methods.
- These lead to local optima near a starting point.
Why do we need user help?

- Why not run all points shortest path and find best closed curve?

Lessons

- Perceptual organization, middle level knowledge, needed for boundary detection.
- Fully automatic methods not good enough yet.
- Formulate desired solution then optimize it.