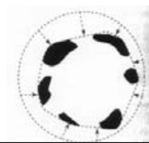
Deformable contours

Tuesday, Oct 7

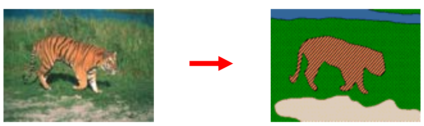
Kristen Grauman
UT-Austin



Midterm

- Tuesday, Oct 14
- Ok to bring one 8.5x11” page of notes

The grouping problem



Goal: move from array of pixel values to a collection of regions, objects, and shapes.

Pixels vs. regions

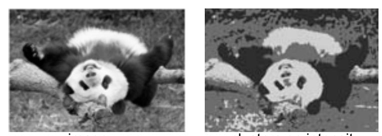


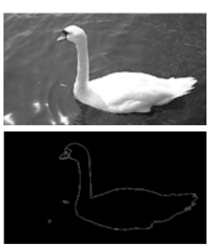
image clusters on intensity

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

image clusters on color

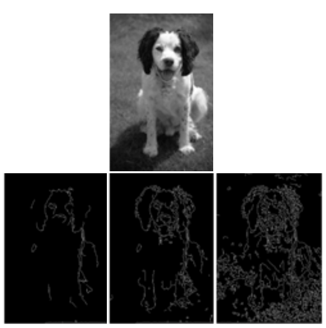
Each region is consistent according to the features and similarity metric we used to do the **clustering**.

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.

Images from D. Jacobs

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

- Generalized Hough transform
- Deformable contours, a.k.a. snakes

Generalized Hough transform

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

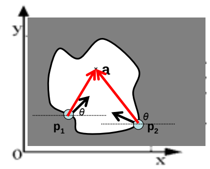


Image space

At each boundary point, compute displacement vector: **r = a – $\mathbf{p_i}$**.

For a given model shape: store these vectors in a table indexed by gradient orientation θ.

[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

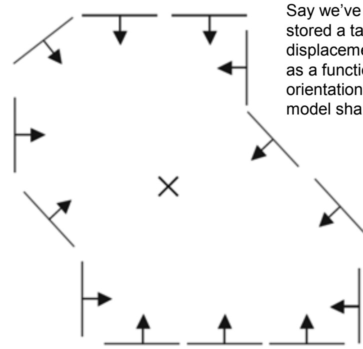
Generalized Hough transform

To *detect* the model shape in a new image:

- For each edge point
  - Index into table with its gradient orientation θ
  - 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

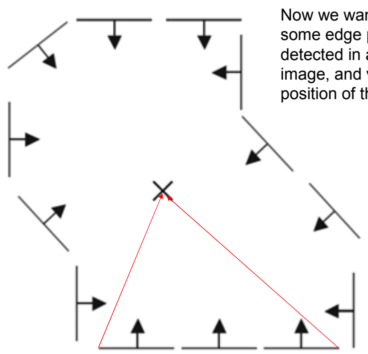


Say we've already stored a table of displacement vectors as a function of edge orientation for this model shape.

model shape

Source: L. Lazebnik

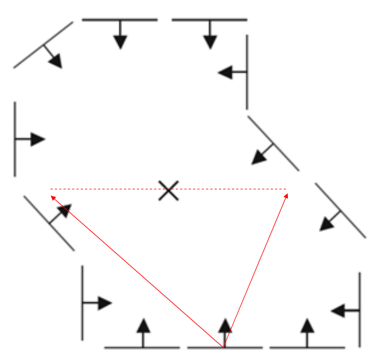
Example



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

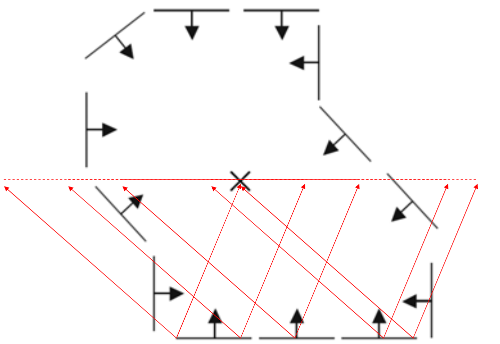
displacement vectors for model points

Example



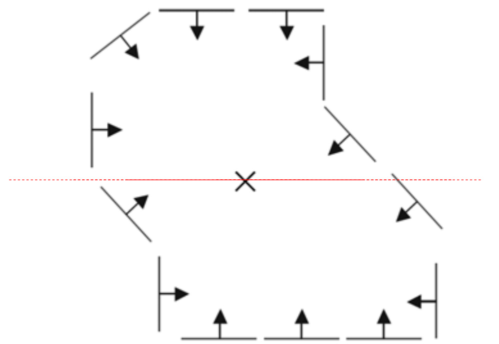
range of voting locations for test point

Example



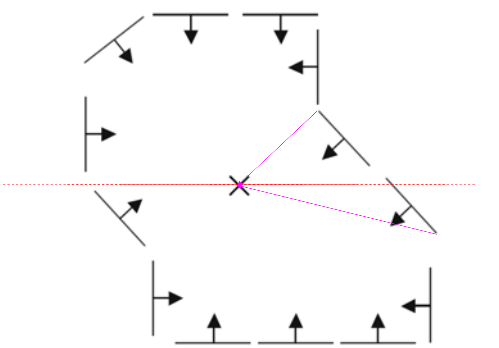
range of voting locations for test point

Example



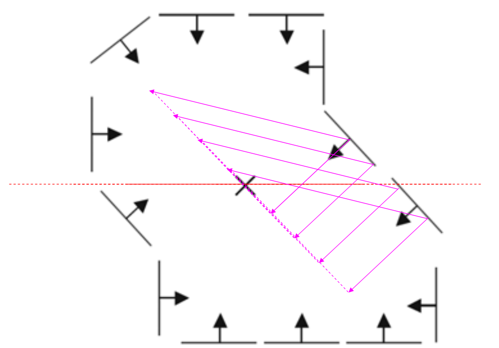
votes for points with $\theta = \uparrow$

Example



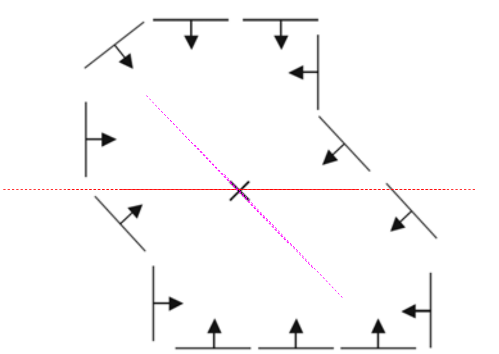
displacement vectors for model points

Example



range of voting locations for test point

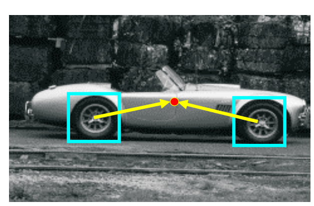
Example



votes for points with $\theta = \swarrow$

Application in recognition

- Instead of indexing displacements by gradient orientation, index by “visual codeword”



training image

visual codeword with displacement vectors

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

Application in recognition

- Instead of indexing displacements by gradient orientation, index by "visual codeword"

test image

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

Features → shapes, boundaries

- **Segment regions** (lecture 8)
  - cluster pixel-level features, like color, texture, position
  - leverage Gestalt properties
- **Fitting models** (lecture 9)
  - explicit 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
- **Background models & foreground detection** (lecture 10)
- **Detection of *deformable* contours, and *semi-automatic* segmentation methods** (today)
  - provide rough initialization nearby true boundary, or
  - interactive, iterative process where user guides the boundary placement

Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

(Single frame)

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

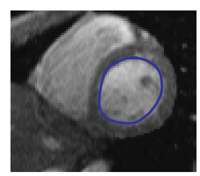
Fig: Y. Boykov

Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



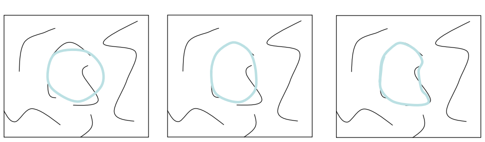
(Single frame)

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Fig: Y. Boykov

Deformable contours

a.k.a. active contours, snakes



initial intermediate final

Initialize near contour of interest

Iteratively refine: elastic band is adjusted so as to

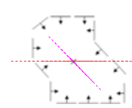
- be near image positions with high gradients, and
- satisfy shape "preferences" or contour priors

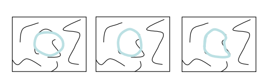
Fig: Y. Boykov

Deformable contours

a.k.a. active contours, snakes

Like generalized Hough transform, useful for shape fitting; but





initial intermediate final

Hough

Fixed model shape

Single voting pass can detect multiple instances

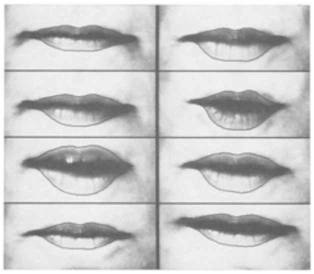
Snakes

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?



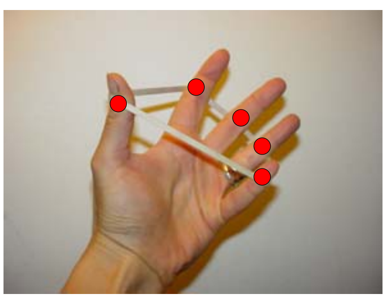
- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands.

Figure from Kass et al. 1987

Why do we want to fit deformable shapes?

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

Deformable contours: intuition



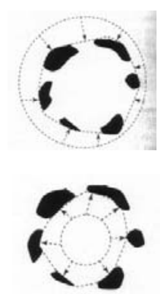


Image from http://www.healthline.com/blogs/exercise_fitness/uploaded_images/HandBand2-795868.JPG

Figure from Shapiro & Stockman

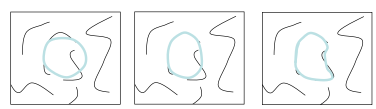
Deformable contours

a.k.a. active contours, snakes

How is the current contour adjusted to find the new contour at each iteration?

•Define a cost function ("energy" function) that says how good a possible configuration is.

•Seek next configuration that minimizes that cost function.



What are examples of problems with energy functions that we have seen previously?

Snakes 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.

Parametric curve representation

(continuous case)

$$v(s) = (x(s), y(s)) \qquad 0 \le s \le 1$$

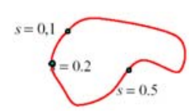


Fig from Y. Boykov

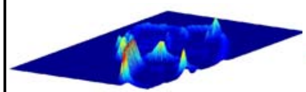
External energy: intuition

- Measure how well the curve matches the image data
- "Attract" the curve toward different image features
  - Edges, lines, etc.

External image energy

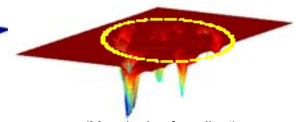
How do edges affect "snap" of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$



- (Magnitude of gradient)

$$-\left(G_x(I)^2 + G_y(I)^2\right)$$

External image energy

- Image *I(x,y)*
- Gradient images $G_x(x,y)$ and $G_y(x,y)$
- External energy at a point *v(s)* on the curve is

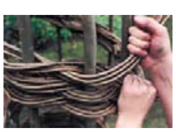
$$E_{external}(\nu(s)) = -(\,|G_x(\nu(s))|^2 + |G_y(\nu(s))|^2\,)$$

- External energy for the whole curve:

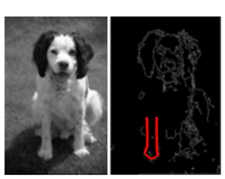
$$E_{external} = \int_0^1 E_{external}(\nu(s))\,ds$$

Internal energy: intuition

Internal energy: intuition

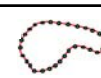


A priori, we want to favor smooth shapes, contours with low curvature, contours similar to a known shape, etc. to balance what is actually observed (i.e., in the gradient image).



http://www3.imperial.ac.uk/pls/portallive/docs/1/52679.JPG

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:

$$E_{internal}(\nu(s)) = \alpha \underbrace{\boxed{\left|\frac{d\nu}{ds}\right|^2}}_{\text{Elasticity, Tension}} + \beta \underbrace{\boxed{\left|\frac{d^2\nu}{d^2 s}\right|^2}}_{\text{Stiffness, Curvature}}$$

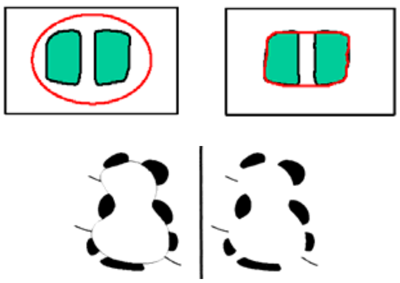
The more the curve bends → the larger this energy value is.

The weights α and β dictate how much influence each component has.

Internal energy for whole curve: $E_{internal} = \int_0^1 E_{internal}(\nu(s))\,ds$

Dealing with missing data

- The smoothness constraint can deal with missing data:



[Figure from Kass et al. 1987]

Total energy
(continuous form)

$$E_{total} = E_{internal} + \gamma \quad E_{external}$$

$$E_{internal} = \int_0^1 E_{\text{internal}}(\nu(s))\,ds$$ // bending energy

$$E_{external} = \int_0^1 E_{external}(\nu(s))\,ds$$ // total edge strength under curve

Parametric curve representation
(discrete form)

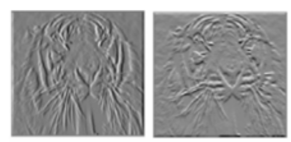
- Represent the curve with a set of *n* points

$$\nu_i = (x_i, y_i) \qquad i = 0 \ldots, n-1$$

Discrete energy function:
external term

- If the curve is represented by *n* points

$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$



$G_x(x,y)$ $G_y(x,y)$

Discrete image gradients

Discrete energy function:
internal term

- If the curve is represented by *n* points

$$\nu_i = (x_i, y_i) \qquad i = 0 \;\ldots.\; n-1$$

$$\frac{d\nu}{ds} \approx \nu_{i+1} - \nu_i \qquad \frac{d^2\nu}{ds^2} \approx (\nu_{i+1} - \nu_i) - (\nu_i - \nu_{i-1}) = \nu_{i+1} - 2\nu_i + \nu_{i-1}$$

$$E_{internal} = \sum_{i=0}^{n-1} \boxed{\alpha \left\| \nu_{i+1} - \nu_i \right\|^2} + \boxed{\beta \left\| \nu_{i+1} - 2\nu_i + \nu_{i-1} \right\|^2}$$

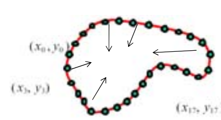
Elasticity, Tension

Stiffness Curvature

Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative, and can be re-written as:

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



Possible problem with this definition?

This encourages a closed curve to shrink to a cluster.

Penalizing elasticity

- To stop the curve from shrinking to a cluster of points, we can adjust the energy function to be:

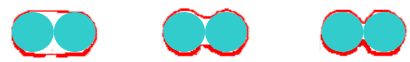
$$E_{internal} = \alpha \cdot \sum_{i=0}^{n-1} \left((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 - \bar{d} \right)^2$$

Average distance between pairs of points – updated at each iteration

- This encourages chains of equally spaced points.

Function of the weights

- α weight controls the penalty for internal elasticity



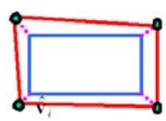
large α medium α small α

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

Fig from Y. Boykov

Optional: specify shape prior

- If object is some smooth variation on a known shape, we can use a term that will penalize deviation from that shape:



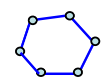
$$E_{internal} = \alpha \cdot \sum_{i=0}^{n-1} (\nu_i - \hat{\nu}_i)^2$$

where $\{\hat{\nu}_i\}$ are the points of the known shape.

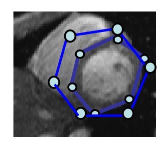
Fig from Y. Boykov

Summary: elastic snake

- A simple elastic snake is defined by
  - A set of *n* points,
  - An internal elastic energy term
  - An external edge based energy term



- To use this to locate the outline of an object
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy



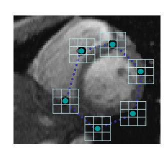
How should the weights in the energy function be chosen?

Energy minimization

- Several algorithms proposed to fit deformable contours, including methods based on
  - Greedy search
  - Dynamic programming (for 2d snakes)
  - (Gradient descent)

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: dynamic programming (for 2d snakes)

- Often snake energy can be rewritten as a sum of pair-wise interaction potentials

$$E_{total}(v_0,\ldots,v_{n-1}) = \sum_{i=0}^{n-1} E_i(v_i,v_{i+1})$$

- Or sum of triple-interaction potentials.

$$E_{total}(v_0,\ldots,v_{n-1}) = \sum_{i=0}^{n-1} E_i(v_{i-1},v_i,v_{i+1})$$

Snake energy: pair-wise interactions

$$E_{total}(x_0,\ldots,x_{n-1},y_0,\ldots,y_{n-1}) = -\sum_{i=0}^{n-1} |G_x(x_i,y_i)|^2 + |G_y(x_i,y_i)|^2 + \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2$$

Re-writing the above with $v_i = (x_i,y_i)$ *:*

$$E_{total}(v_0,\ldots,v_{n-1}) = -\sum_{i=0}^{n-1} \| G(v_i) \|^2 + \alpha \cdot \sum_{i=0}^{n-1} \| v_{i+1} - v_i \|^2$$

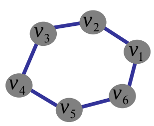
$$E_{total}(v_0,\ldots,v_{n-1}) = \sum_{i=0}^{n-1} E_i(v_i,v_{i+1})$$

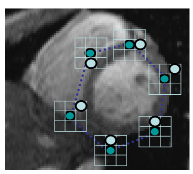
where $E_i(v_i,v_{i+1}) = -\| G(v_i) \|^2 + \alpha \| v_{i+1} - v_i \|^2$

We are defining this function E_i

What terms of this sum will a vertex v_i affect?

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.

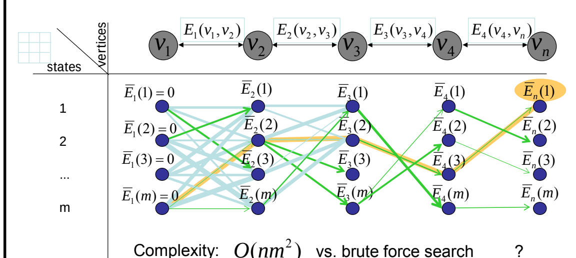
Fig from Y. Boykov
[Amini, Weymouth, Jain, 1990]

Energy minimization: dynamic programming

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

This example: considering first-order interactions, one iteration.

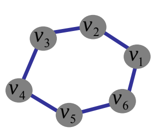
$$E_{total} = E_1(v_1,v_2) + E_2(v_2,v_3) + \ldots + E_{n-1}(v_{n-1},v_n)$$

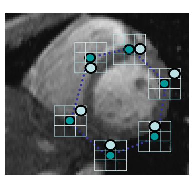


Complexity: $O(nm^2)$ vs. brute force search ____?

Example adapted from Y. Boykov

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.

Fig from Y. Boykov
[Amini, Weymouth, Jain, 1990]

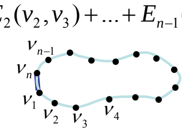
Energy minimization: dynamic programming

DP can be applied to optimize an open ended snake

$$E_1(v_1,v_2) + E_2(v_2,v_3) + \ldots + E_{n-1}(v_{n-1},v_n)$$

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

$$E_1(v_1,v_2) + E_2(v_2,v_3) + \ldots + E_{n-1}(v_{n-1},v_n) + \boxed{E_n(v_n,v_1)}$$



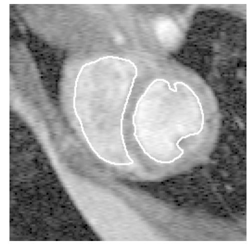
Work around:

1) Fix v_1 and solve for rest .
2) Fix an intermediate node at its position found in (1), solve for rest.

Tracking via deformable models

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.

Deformable contours



Tracking Heart Ventricles
(multiple frames)

Tracking via deformable models

Visual Dynamics Group, Dept. Engineering Science, University of Oxford.

Applications: Traffic monitoring
Human-computer interaction
Animation
Surveillance
Computer Assisted Diagnosis in medical imaging

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.

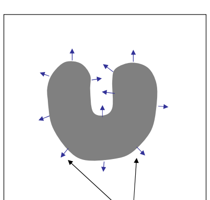


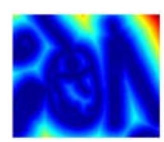
image gradients ∇I
are large only directly on the boundary

Distance transform

- External image can also be taken from the **distance transform** of the edge image.

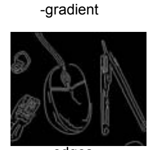
original

-gradient



distance transform

Value at (x,y) tells how far that position is from the nearest edge point (or other binary mage structure)

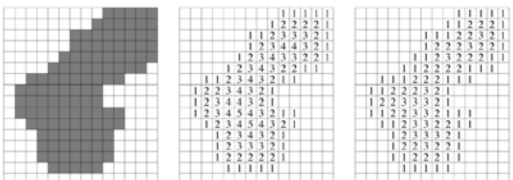


edges

>> help bwdist

Distance transform

- Image reflecting distance to nearest point in point set (e.g., edge pixels, or foreground pixels).

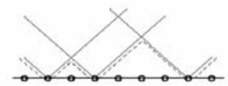


4-connected adjacency **8-connected adjacency**

Distance transform (1D)

Two pass O(n) algorithm for 1D L_1 norm

1. Initialize: For all j
   $D[j] \leftarrow 1_P[j]$ // 0 if j is in **P**, infinity otherwise
2. Forward: For j from 1 up to n-1
   $D[j] \leftarrow \min(D[j], D[j-1]+1)$
3. Backward: For j from n-2 down to 0
   $D[j] \leftarrow \min(D[j], D[j+1]+1)$

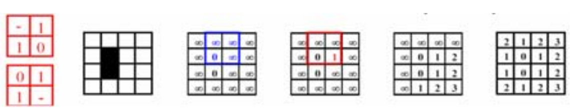


∞	0	∞	0	∞	∞	∞	0	∞
∞	0	1	0	1	2	3	0	1
1	0	1	0	1	2	1	0	1

Adapted from D. Huttenlocher

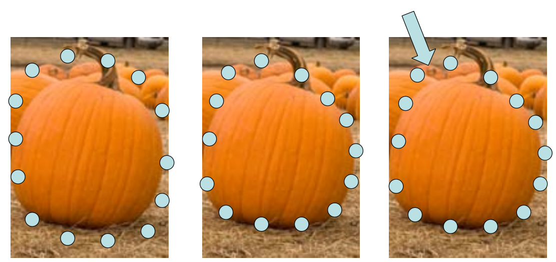
Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right



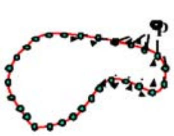
Adapted from D. Huttenlocher

Interactive forces



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:



$$E_{push} = +\sum_{i=0}^{n-1} \frac{r^2}{|\nu_i - p|^2}$$

Nearby points get pushed hardest

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

Intelligent scissors

Use dynamic programming to compute optimal paths from every point to the seed based on edge-related costs

User interactively selects most suitable boundary from set of all optimal boundaries emanating from a seed point

[Mortensen & Barrett, SIGGRAPH 1995, CVPR 1999]

Snakes: 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

Summary: main points

- Deformable shapes and active contours are useful for
  - Segmentation: fit or "settle" to boundary in image
  - Tracking: previous frame's estimate serves to initialize the next
- Optimization for snakes: general idea of minimizing a cost/energy function
  - Can define terms to encourage certain shapes, smoothness, low curvature, push/pulls, …
  - And can use weights to control relative influence of each component cost term.
- Edges / optima in gradients can act as "attraction" force for interactive segmentation methods.
- Distance transform definition: efficient map of distances to nearest feature of interest.