Lecture 9: Fitting, Contours

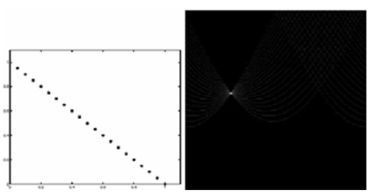
Thursday, Sept 27

Announcements

- Midterm review:
  next Wed Oct 4, 12-1 pm, ENS 31NQ

Last time

- Fitting shape patterns with the Hough transform and generalized Hough transform

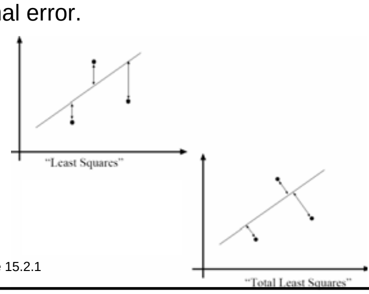


Today

- Fitting lines (brief)
  - Least squares
  - Incremental fitting, k-means allocation
- RANSAC, robust fitting
- Deformable contours

Line fitting: what is the line?

- Assuming all the points that belong to a particular line are known, solve for line parameters that yield minimal error.



Forsyth & Ponce 15.2.1

Line fitting: which point is on which line?

Two possible strategies:

- Incremental line fitting
- K-means

Incremental line fitting

- Take connected curves of edge points and fit lines to runs of points (use gradient directions)

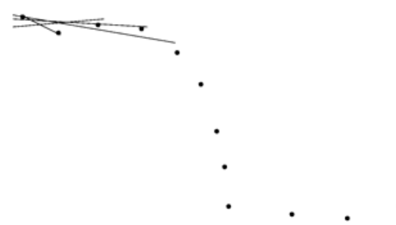
Incremental line fitting

Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

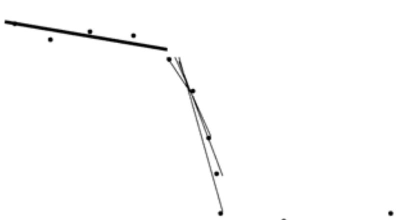
```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
  Transfer first few points on the curve to the line point list
  Fit line to line point list
  While fitted line is good enough
    Transfer the next point on the curve
      to the line point list and refit the line
  end
  Transfer last point(s) back to curve
  Refit line
  Attach line to line list
end
```

Incremental line fitting

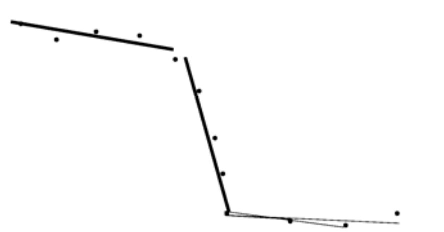
Incremental line fitting



Incremental line fitting



Incremental line fitting



Incremental line fitting

If we have occluded edges, will often result in more than one fitted line

Allocating points with k-means

- Believe there are k lines, each of which generates some subset of the data points
- Best solution would minimize the sum of the squared distances from points to their assigned lines
- Use k-means algorithm
- Convergence based on size of change in lines, whether labels have been flipped.

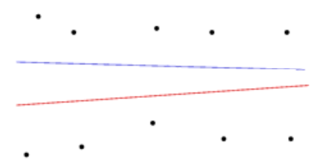
Allocating points with k-means

Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

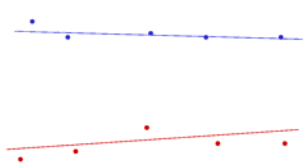
```
Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
   and then fit lines using this assignment

Until convergence
   Allocate each point to the closest line
   Refit lines
end
```

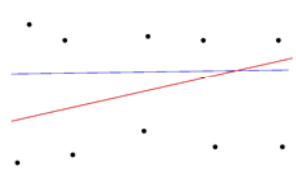
K-means line fitting



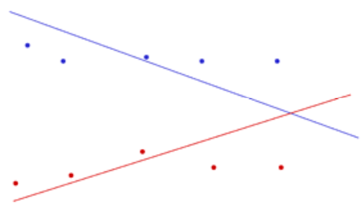
K-means line fitting



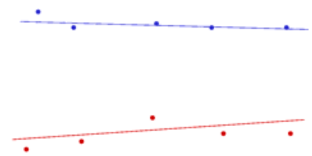
K-means line fitting



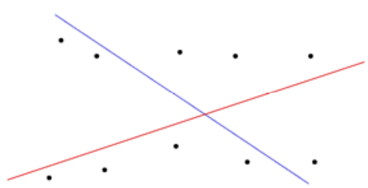
K-means line fitting



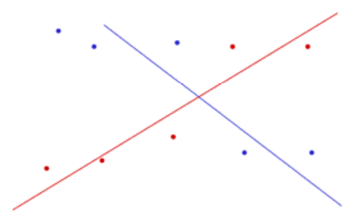
K-means line fitting



K-means line fitting



K-means line fitting

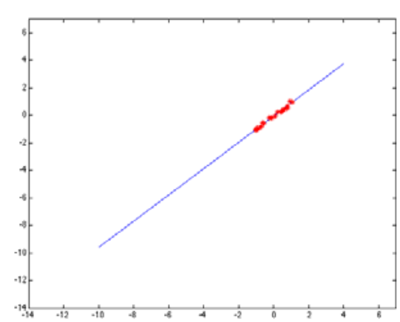


Sensitivity to starting point

Outliers

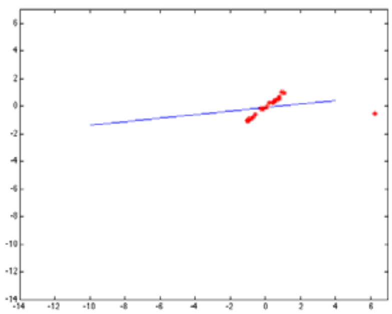
- **Outliers** can result from
  - Data collection error
  - Overlooked case for the model chosen
- Squared error terms mean big penalty for large errors, can lead to significant bias

Outliers affect least squares fit

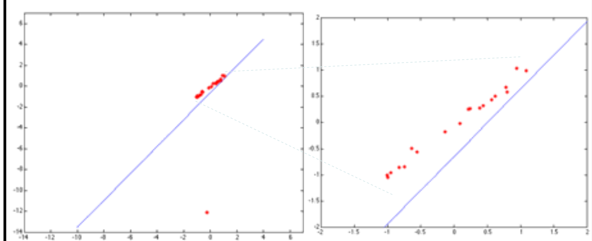


Forsyth & Ponce, Fig 15.7

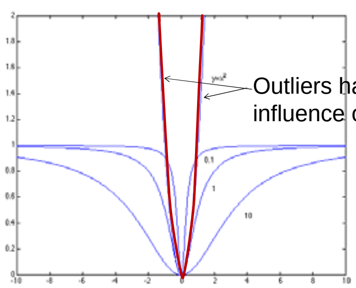
Outliers affect least squares fit



Outliers affect least squares fit



Least squares and error



Best model minimizes residual error:

$$\sum_i r_i(x_i, \theta)$$

data point model parameters

Least squares and error

- If we are assuming Gaussian additive noise corrupts the data points
  – Probability of noisy point being within distance *d* of corresponding true point decreases rapidly with *d*
  – So, points that are way off are not really consistent with Gaussian noise hypothesis, model wants to fit to them…

Robustness

- A couple possibilities to handle outliers:
  – Give the noise heavier tails
  – Search for "inliers"

M-estimators

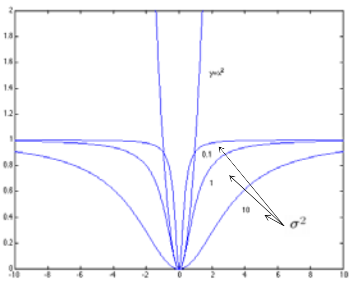
- Estimate parameters by minimizing modified residual expression

$$\sum_i \rho(\underbrace{r_i(x_i, \theta)}_{\text{residual error}}; \sigma)$$

parameter determining when function **flattens out**

- Reflects a noise distribution that does not vanish as quickly as Gaussian, i.e., consider outliers more likely to occur
- De-emphasizes contribution of distant points

Example M-estimator



original

$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$

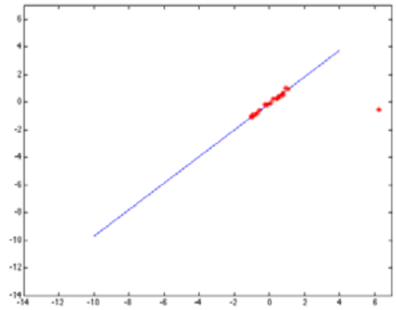
Looks like distance for small values,
Like a constant for large values

Non-linear optimization, must be solved iteratively

σ^2

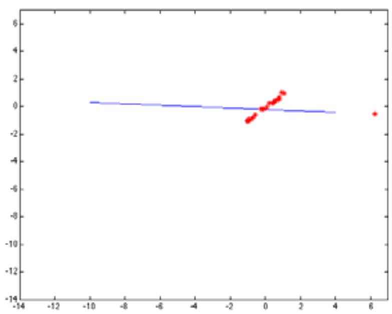
Impact of sigma on fitting quality?

Applying the M-estimator



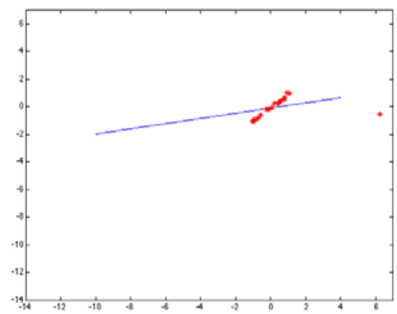
Fit with good choice of σ^2

Applying the M-estimator



σ^2 too small: error for all points similar

Applying the M-estimator



σ^2 too large: error about same as least squares

Scale selection

- Popular choice: at iteration n during minimization

$$\sigma^{(n)} = 1.4826 \; \text{median}_i \, |r_i^{(n)}(x_i; \theta^{(n-1)})|$$

RANSAC

- RANdom Sample Consensus
- Approach: we don’t like the impact of outliers, so let’s look for “inliers”, and use those only.

RANSAC

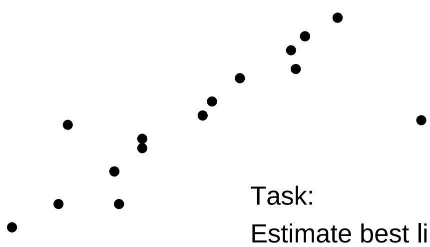
- Choose a *small subset* uniformly at random
- Fit to that
- Anything that is *close* to result is signal; all others are noise
- Refit
- Do this *many times* and choose the best (best = lowest fitting error)

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

```
Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion
```

RANSAC Reference: M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

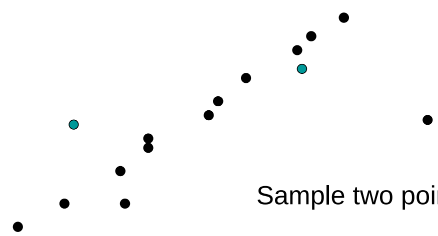
RANSAC Line Fitting Example



Task:
Estimate best line

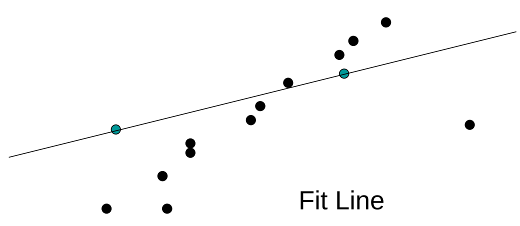
Slide credit: Jinxiang Chai, CMU

RANSAC Line Fitting Example



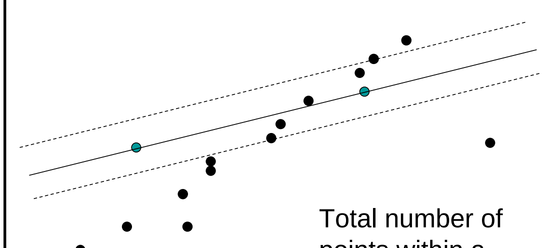
Sample two points

RANSAC Line Fitting Example



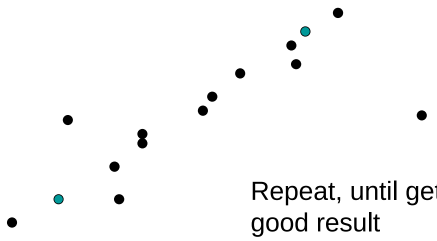
Fit Line

RANSAC Line Fitting Example



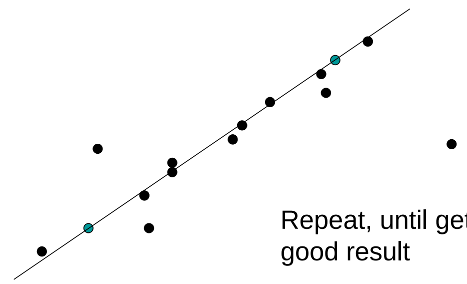
Total number of points within a threshold of line.

RANSAC Line Fitting Example



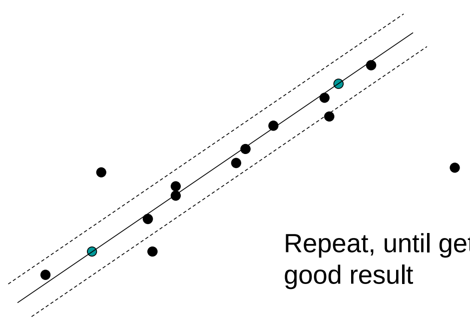
Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

RANSAC Line Fitting Example



Repeat, until get a good result

RANSAC application: robust computation

Interest points (Harris corners) in left and right images

about 500 pts / image

640x480 resolution

Putative correspondences (268)

(Best match,SSD<20)

Outliers (117)

(*t*=1.25 pixel; 43 iterations)

Inliers (151)

Final inliers (262)

Hartley & Zisserman p. 126

RANSAC parameters

- Number of samples required (*n*)
  - Absolute minimum will depending on model being fit (lines -> 2, circles -> 3, etc)
- Number of trials (*k*)
  - Need a guess at probability of a random point being “good”
  - Choose so that we have high probability of getting one sample free from outliers
- Threshold on good fits (*t*)
  - Often trial and error: look at some data fits and estimate average deviations
- Number of points that must agree (*d*)
  - Again, use guess of probability of being an outlier; choose *d* so that unlikely to have one in the group

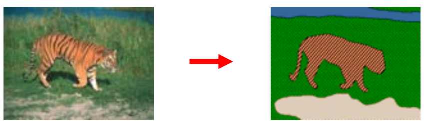
Grouping and fitting

- Grouping, segmentation: make a compact representation that merges similar features
  - Relevant algorithms: K-means, hierarchical clustering, Mean Shift, Graph cuts
- Fitting: fit a model to your observed features
  - Relevant algorithms: Hough transform for lines, circles (parameterized curves), generalized Hough transform for arbitrary boundaries; least squares; assigning points to lines incrementally or with k-means; robust fitting

Today

- Fitting lines (brief)
  - Least squares
  - Incremental fitting, k-means allocation
- RANSAC, robust fitting
- Deformable contours

Towards object level grouping

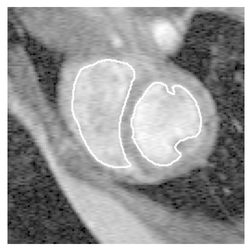


Low-level segmentation cannot go this far…
How do we get these kinds of boundaries?

One direction: semi-automatic methods
- Give a good but rough initial boundary
- Interactively guide boundary placement

Still use image analysis techniques in concert.

Deformable contours



Tracking Heart Ventricles
(multiple frames)

Deformable contours

a.k.a. active contours, snakes

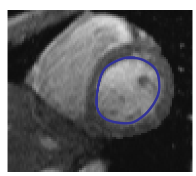
Given: initial contour (model) near desired object

(Single frame)

Deformable contours

a.k.a. active contours, snakes

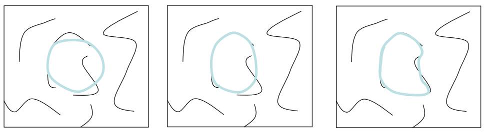
Goal: evolve the contour to fit exact object boundary



[Kass, Witkin, Terzopoulos 1987]

Deformable contours

a.k.a. active contours, snakes



initial intermediate final

Deformable contours

a.k.a. active contours, snakes

- Elastic band of arbitrary shape, initially located near image contour of interest
- Attracted towards target contour depending on intensity gradient
- Iteratively refined

Comparison: shape-related methods

- **Chamfer matching**: given two shapes defined by points, measure average distance from one to the other
- **(Generalized) Hough transform**: given pattern/model shape, use oriented edge points to vote for likely position of that pattern in new image
- **Deformable contours**: given initial starting boundary and priors on preferred shape types, iteratively adjust boundary to also fit observed image

Snake Energy

The total energy of the current snake defined as

$$E_{total} = E_{in} + E_{ex}$$

Internal energy encourages smoothness or any particular shape

External energy encourages curve onto image structures (e.g. image edges)

Internal energy incorporates **prior** knowledge about object boundary, which allows a boundary to be extracted even if some image data is missing

We will want to iteratively *minimize* this energy for a good fit between the deformable contour and the target shape in the image

Many of the snakes slides are adapted from Yuri Boykov

Parametric curve representation

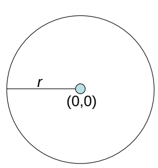
- Coordinates given as functions of a parameter that varies along the curve
- For example, for a circle with center (0,0):

parametric form:

x = r sin(s)
y = r cos(s)

parameters:

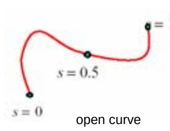
radius r
angle 0 <= s < 2pi



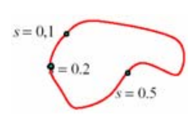
(continuous case)

Parametric curve representation

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



open curve



closed curve

Curves parameterized by arc length, the length along the curve

(continuous case)

Internal energy

- Bending energy of a continuous curve

$$E_{in}(\nu(s)) = \boxed{\alpha(s)\left|\frac{d\nu}{ds}\right|^2} + \boxed{\beta(s)\left|\frac{d^2\nu}{d^2s}\right|^2}$$

Elasticity, Tension

Stiffness, Curvature

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

Internal energy for a curve:

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

External energy

- Measures how well the curve matches the image data, locally
- Attracts the curve toward different image features
  – Edges, lines, etc.

External energy: edge strength

- Image *I(x,y)*
- Gradient images $G_x(x,y)$ & $G_y(x,y)$
- External energy at a point is

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

(**Negative** so that minimizing it forces the curve toward **strong** edges)

- External energy for the curve:

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

Snake Energy (continuous form)

$$E_{total} = E_{in} + E_{ex}$$

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

e.g. bending energy

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

e.g. total edge strength under curve

Discrete approach

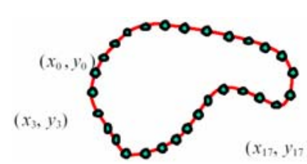
discrete image ➡ discrete snake representation ➡ discrete optimization (dynamic programming)

Parametric curve representation

(discrete case)

- Represent the curve with a set of n points

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



Discrete representation

- If the curve is represented by *n* points

$$v_i = (x_i, y_i) \qquad i = 0 \;\cdots\; n-1$$

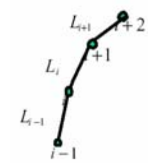
$$\frac{dv}{ds} \approx \frac{v_{i+1} - v_i}{2} \qquad \frac{d^2 v}{ds^2} \approx (v_{i+1} - v_i) - (v_i - v_{i-1}) = v_{i+1} - 2v_i + v_{i-1}$$

$$E_{in} = \sum_{i=0}^{n-1} \underbrace{\boxed{\alpha |v_{i+1} - v_i|^2}}_{\text{Elasticity, Tension}} + \underbrace{\boxed{\beta |v_{i+1} - 2v_i + v_{i-1}|^2}}_{\text{Stiffness Curvature}}$$

Simple elastic curve

- For a curve represented as a set of points a simple elastic energy term is

$$\begin{aligned} E_{in} &= \alpha \cdot \sum_{i=0}^{n-1} L_i^2 \\ &= \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \end{aligned}$$



This encourages the closed curve to shrink to a point (like a very small elastic band)

Encouraging point spacing

- To stop the curve from shrinking to a point

$$E_{in} = \alpha \cdot \sum_{i=0}^{n-1} (L_i - \hat{L}_i)^2$$

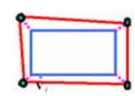
Average distance between pairs of points – updated at each iteration

  – encourages formation of equally spaced chains of points

Optional: specify shape prior

- If object is some smooth variation on a known shape, use

$$E_{in} = \alpha \cdot \sum_{i=0}^{n-1} (v_i - \hat{v}_i)^2$$



  - where $\{\hat{v}_i\}$ give points of the basic shape

Edge strength for external energy

- An external energy term for a (discrete) snake based on image edge

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


Summary: simple elastic snake

- A simple elastic snake is thus 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

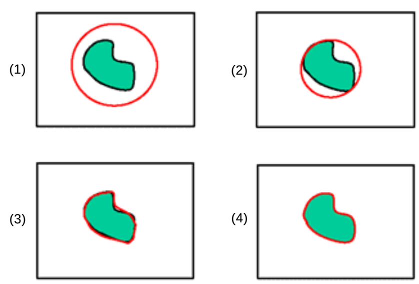
Energy minimization

- Many algorithms proposed to fit deformable contours
  - Greedy search
  - Gradient descent
  - Dynamic programming (for 2d snakes)

Greedy minimization

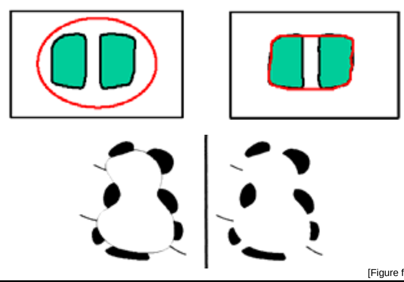
- For each point, search window around it and move to where energy function is minimal
- Stop when predefined number of points have not changed in last iteration
- Local minimum

Synthetic example



Dealing with missing data

- The smoothness constraint can deal with missing data:



[Figure from Kass et al. 1987]

Relative weighting

- α weight controls internal elasticity

large α medium α small α

Dynamic programming (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

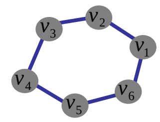
$$E_{total}(x_0, \ldots, x_{n-1}, y_0 \ \ldots, y_{n-1}) = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$
$$+ \quad \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$

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

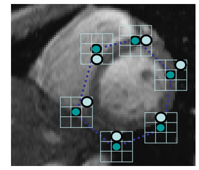
$$\boxed{E_{total}(v_0, \ldots, v_{n-1}) = \sum_{i=0}^{n-2} E_i(v_i, v_{i+1})}$$

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

DP Snakes [Amini, Weymouth, Jain, 1990]



control points

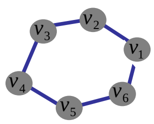


First-order interactions (elasticity)

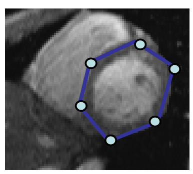
$$E(v_1, v_2, \ldots, v_n) = E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$$

Energy E is minimized via Dynamic Programming

DP Snakes [Amini, Weymouth, Jain, 1990]



control points



First-order interactions (elasticity)

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

Energy E is minimized via Dynamic Programming

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

DP Viterbi Algorithm

- Reuse solutions to subproblems
- Introduce intermediate variables

$$s_2(v_2) = \min_{v_1} E_1(v_1, v_2)$$

$$s_3(v_3) = \min_{v_2}(s_2(v_2) + E_2(v_2, v_3))$$

$$s_4(v_4) = \min_{v_3}(s_3(v_3) + E_3(v_3, v_4))$$

$$\vdots$$

$$s_n(v_n) = \min_{v_{n-1}}(s_{n-1}(v_{n-1}) + E_{n-1}(v_{n-1}, v_n))$$

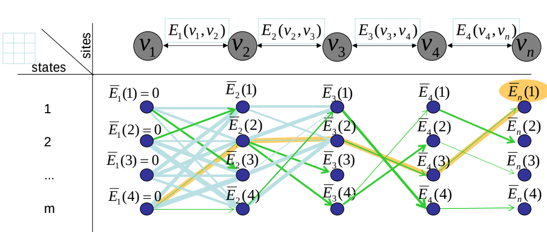
determine optimal position of predecessor, for each possible position of self

$s_k(v_k)$: lowest total energy for the first k-1 vertices of the snake for a given value of v_k

DP Viterbi Algorithm

Considering first-order interactions (elasticity), one minimization iteration

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

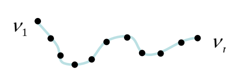


Complexity: $O(nm^2)$ vs. brute force search ____?

Dynamic Programming for a closed snake?

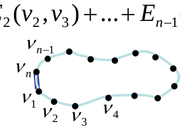
DP can be applied to optimize an open ended snake

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



What about "looped" energy, in the case of a closed snake?

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



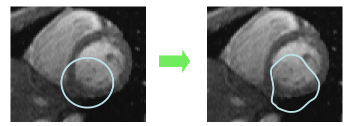
Problems with snakes

- Depends on number and spacing of control points
- Snake may oversmooth the boundary
- Not trivial to prevent curve self intersecting

- Cannot follow topological changes of objects

Problems with snakes

- May be sensitive to initialization, get stuck in local minimum



- Accuracy (and computation time) depends on the convergence criteria used in the energy minimization technique

Problems with snakes

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

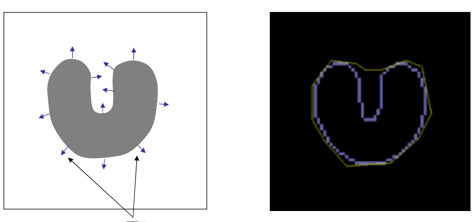


image gradients ∇I
are large only directly on the boundary

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 steps 1 and 2 for *t* = *t+1*

Tracking via deformable models

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

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

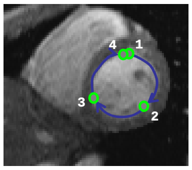
Intelligent scissors

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

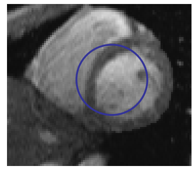
[Mortensen & Barrett, SIGGRAPH 1995, CVPR 1999]

Snakes vs. scissors



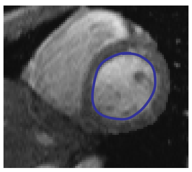
Shortest paths on image-based graph connect seeds placed on object boundary

Snakes vs. scissors



Given: initial contour (model) near desirable object

Snakes vs. scissors



Given: initial contour (model) near desirable object

Goal: evolve the contour to fit exact object boundary

Coming up

- Stereo
- F&P 10.1, 11
- Trucco & Verri handout