# INF 5300 – Regularization, energy functions, and snakes

Anne Solberg (anne@ifi.uio.no)

25.02.15

This and the next lecture will cover:

•Introduction to energy functions

• Snakes

• Mean shift segmentation

• Contextual classification

## This lecture

- Introduction to regularization
- Segmentation by closed contours using snakes
- Snake optimization algorithms
- Practical  use of snakes
- Limitations of snakes
- (Continued next week if needed)

## Curriculum

- Szeliski 3.7.1 Introduction to energy functions

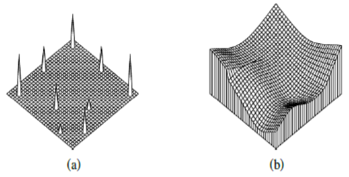
- Lectures notes for snakes are based on 6.1-6.3 in Nixon and Aguado ~inf5300/pensum-artikler/activecontour_kap6.pdf
- Snakes are briefly covered in Szeliski 5.1.1

## Segmentation and energy functions

- Simple segmentation often gives non-continuous or noisy results.
- Smoothness constrains are also used for e.g. image restoration/interpolation etc.
- Regularization is a method to impose smoothness contraints by using an energy function.
- Markov random fields is a related field where we can use prior models for constraining the segmentation.

## Regularization

- Regularization originates from statistics:
  - Finding a smooth surface that fits a set of data points.
- Many surfaces might fit, but the shape of the surface might be sensitive to noise in input data.
- Regularization can be formulated as estiming an unknown function f(x,t) from data points d(x,y).



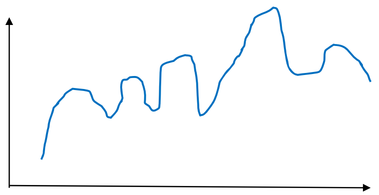
## Producing smooth 1D functions

- The derivative of the function is integrated to get a scalar measure of smoothness.
- Integrated first derivative:

$$\varepsilon_1 = \int f_x^2(x)dx$$

- Integrated second derivative:

$$\varepsilon_2 = \int f_{xx}^2(x)dx$$



•Function f(x)
•First derivative $f_x(x)$
•Second derivative $f_{xx}(x)$

• For a boundary:
Sum the first derivatives – minimize the length of the boundary
Sum the second derivatives – avoid points with high curvature

## Smooth 2D functions

- For images or surfaces, the smoothness functionals are:

$$\varepsilon_1 = \int f_x^2(x,y) + f_y^2(x,y)dxdy = \int \|\nabla f(x,y)\|^2$$

First derivative:
Often called *membrane* because it is similar to a tent-like structure

Second derivative:
Often called *thin-plate spline* because it is similar to how a thin plate bends under small deformation.

$$\varepsilon_2 = \int f_{xx}^2(x,y) + 2f_{xy}^2(x,y) + f_{yy}^2(x,y)dxdy$$

where the mix term $2f_{xy}^2(x,y)$ is needed to make the measure rotationally invariant

- These functions are often combined, and they can be assigned a weight term.

## Data terms

- The energy function must also include a data term considering the gray levels/gradients/feature vectors in the image.
- A simple discrete data term:

$$\varepsilon_d = \sum_i [f(x_i, y_i) - d_i]^2$$

- For continuous data:

$$\varepsilon_d = \int [f(x, y) - d(x,y)]^2 dxdy$$

- The combined energy function is:

$$\varepsilon = \varepsilon_d + \lambda \varepsilon_s$$

$\lambda$ is a parameter that controls the amount of smoothing.

## Minimizing the energy function

- f(x,y) is normally discretized to a regular grid.
- This is done by finite element analysis, and the function is approximated with a piecewise continuous spline (for this we need a discrete derivation operator, which we know well in image analysis).
- Energy function for the first derivative (grid size h):

$$E_1 = \sum_{i,j} s_x(i,j)\left[f(i+1,j) - f(i,j) - g_x(i,j)\right]^2 + s_y(i,j)\left[f(i,j+1) - f(i,j) - g_y(i,j)\right]^2$$

- $s_x$(i,j) and $s_y$(i,j) are the smoothness weights and control the location of horizontal or vertical tears in the data.
- $g_x$(i,j) and $g_y$(i,j) the horizontal and vertical gradient of the data.
- This function is minimized if the gradients of f is similar to the gradients of the image.
-

## Minimizing the energy function

- Energy function for the second derivative
- The resulting discrete forms are simple (grid size h):

$$E_2 = -h^2 \sum_{i,j} c_x(i,j)\left[f(i+1,j) - 2f(i,j) + f(i-1,j)\right]^2 + 2c_m(i,j)\left[f(i+1,j+1) - f(i+1,j) - f(i,j+1) + f(i,j)\right]^2 + \mathrm{c_y(i,j)}\left[f(i,j+1) - 2f(i,j) + f(i,j+1)\right]^2$$

- $c_x$(i,j), $c_m$(i,j) and $c_y$(i,j) are the crease variables.
- *Crease: a line made by pressing, folding or wrinkling (folding a thin-plate)*

## Parameters:

- $s_x$(i,j) and $s_y$(i,j) are the smoothness weights and control the location of horizontal or vertical tears in the data.
- $c_x$(i,j), $c_m$(i,j) and $c_y$(i,j) are the crease variables.
- $g_x$(i,j) and $g_y$(i,j) the horizontal and vertical gradient of the data.
- The discrete data term is:

$$E_d = \sum_{i,j} w(i,j)\left[f(i,j) - d(i,j)\right]^2$$

The data weight w(i,j) controls the influence of the data term.

Estimating/specifying these parameters may be difficult.

## A quadratic optimization problem

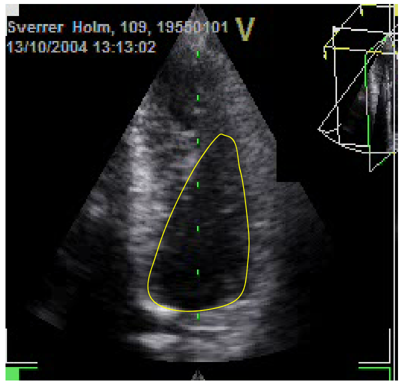
- The total energy can be written as a quadratic form:

$$E = Ed + \lambda Es = \mathbf{x}^T \mathbf{A}\mathbf{x} - 2\mathbf{x}^T \mathbf{b} + c$$
$$\text{where } \mathbf{x} = \left[\mathrm{f(0,0),.....f(m-1,n-1)}\right]$$

- We minimize this by solving the equation we get when the dervative is zero:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

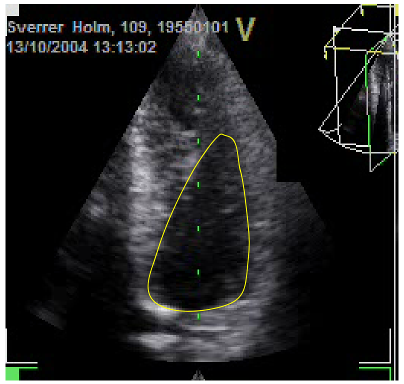
## Example – segmenting ultrasound images of the hearth



Find the border of the left ventricle

- 3D object with a closed border
- 2D views have partly discontinuous border
- Noisy image

## Can previous segmentation methods work?

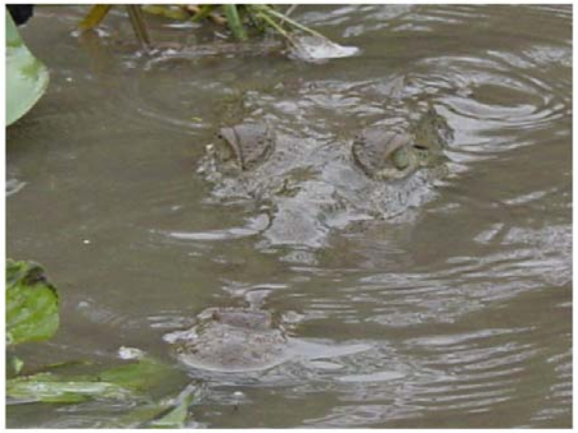


- Thresholding?
- Hit and miss?
- Region growing?
- Edge-based segmentation?
- Watershed?
- Line detection?
- Hough transform?
  - Ellipse?
  - Can be extended to general shapes if the precise mathematical description of the shape is known.

## Motivation

- Common assumption for many segmentation methods:
  - Digital images will show real world objects as well-defined regions with unique gray levels and a clear border against a uniform background.
    - There are many applications where this assumption does not hold.
      - Textured images.
      - Noisy images (ultrasound, SAR (syntetic aperture radar)) images.
      - Images with partly occluded borders
        - » 2D images of 3D objects

## Motivation



Beware of extreme case of blending and occlusion

## Motivation

- We have seen several cases where prior knowledge is used:
  - Thresholding: knowledge about distribution of gray levels can be used.
  - Adaptive thresholding: Window size should be determined in relation to the size of the objects we want to find.
  - Character recognition: size (and shape) of the typical characters useful for both segmentation and feature extraction.
  - Hough transform: a precise model for the shape is used.

## Motivation

1. The images we will look at now are just another example of segmentation methods using models external to the image in order to obtain the best possible segmentation.
2. A typical application where these methods are useful is segmentation of medical ultrasound images
   - Much noise and blurred edges
   - Much knowledge about the shape of the objects.

## The initial idea: Snakes

- An active contour (snake) is a set of points which aims to enclose a target feature.
- Snakes are model-based methods for localization and tracking of image structures.
- The snake is defined as an energy minimizing contour (often defined using splines).
- The <u>energy</u> of the snake depends on its shape and location within the image.
- Snakes are attracted to image boundaries through forces.

## The initial idea: Snakes

- The approach is iterative:
  1. The user draws an initial approximate contour.
  2. A dynamic simulation is started.
  3. The contour is deformed until it reaches equilibrium.
- Snakes depend on:
  - Interaction with the user
  - Interaction with a high-level description.
  - Interaction with image data adjacent in space and time.

## The initial idea: Snakes

- The energy of the snakes is affected by different types of forces:
  1. Internal forces:
     - Tension/elasticity forces that make the snake act like a membrane.
     - Rigidity forces that make the snake act like a thin plate that resists bending.
  2. Image forces.
  3. Constraint forces
     - User-supplied forces that come from higher-level image understanding processes.

## Representation of the contour

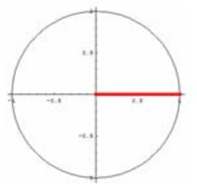
- The contour is represented as:

  $v(s) = (x(s), y(s))^T$
- This is a parametric representation of the contour.
- The vector describing the position of every point on the contour makes one pass over the entire contour as *s* varies from its mimimum to its maximum value.
- Typically, *s* is normalized

  $s \in [0,1]$
- We only need coordinates (x(s),y(s)) of the points on the contour, not a mathematical equation for the contour.

## What is a parametric contour?

- Let $x(s)=\cos(2\pi s)$ and $y(s)=\sin(2\pi s)$
- Let $s \in [0,1]$
- Then v(s) describes a circle as *s* varies from 0 to 1.



- See also contour representation in INF 3300
  http://www.ifi.uio.no/~inf3300/2007H/object-representation.pdf

## Energy functions

- Finding the contour is described as an energy minimization problem.
- The energy function consists of several terms:
  - The snakes own properties (bending, stretching)
  - Image energy (edge magnitude along the snake)
  - Constraints making the contour smooth etc.
- The energy function is also called a <u>functional</u>.
- The final position of the contour will correspond to a minimum of this energy function.
- Typically, the energy function is minimized in a iterative algorithm.

## The energy function

$$E_{snake} = \int_{s=0}^{1} E_{\text{int}}(f(s)) + E_{image}(f(s)) + E_{con}(f(s))ds$$

Internal deformation energy of the snake itself. How it can bend and stretch.

A term that relates to gray levels in the image, e.g. attracts the snake to points with high gradient magnitude.

Constraints on the shape of the snake. Enchourages the contour to be smooth. (Often omitted)

- Let the contour points v(s)=f(s) be the function we want to find
- The minimum value is found by derivation:

$$\frac{dE_{snake}}{df} = 0$$

## The internal deformation term

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

First derivative
Measures how stretched the contour is.
Keyword: point spacing.
Imposes tension.
The curve should be short if possible.
Physical analogy: v acts like a membrane.

Second derivative
Measures the curvature or bending energy.
Keyword: point variation.
Imposes rigidity.
Changes in direction should be smooth.
Physical analogy: v acts like a thin plate.

- $\alpha$ and $\beta$ are penalty parameters that control the weight of the two terms.
- Low $\alpha$ values: the snake can stretch much.
- Low $\beta$ values: the snake can have high curvature.

## The image term

- Attracts the snake to features in the image, like edge pixels or bright pixels.
- Originally, it consisted of a term for lines, edges (and maybe also terminations): $E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}$
- $w_{line}$, $w_{edge}$, and $w_{term}$ are weights that control the influence of each term.
- $E_{line}$ can be set to image intenstity values. If $w_{line}$ is positive, it will attract the snake to dark regions, and to bright regions if $w_{line}$ is negative.
- $E_{edge}$ can be computed using an edge detector.
- $E_{term}$ is not commonly used.

## The image term

- We choose the image term in such a way that is coincides with special features in the image, e.g. bright or dark areas, or edges.
- If I(x,y) is the image intensity for point (x,y), what kind of structure does this function attract the snake to?

$$E_{\text{Image}} = -I(x,y)$$

- A common way of defining $E_{\text{Image}}$ is:

$$E_{\text{Image}} = -c\left|\nabla G_\sigma * I(x,y)\right|,$$

where $\nabla G_\sigma$ is a smoothed gradient filter

- If we want to attract the snake to user-defined points d(i) we add the term:

$$E_{\text{spring}} = k_i \left\| v(i) - d(i) \right\|^2$$

## The energy function

- Simple snake with only two terms (no termination energy):

$$E_{snake}(s) = E_{\text{int}}(v_s) + E_{image}(v_s)$$

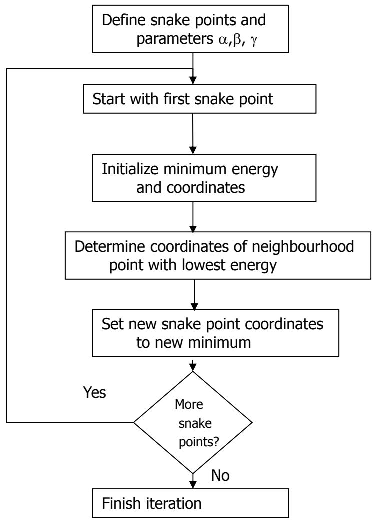
$$= \alpha\left|\frac{dv_s}{ds}\right|^2 + \beta\left|\frac{d^2v_s}{ds^2}\right|^2 + \gamma E_{edge}$$

- We need to approximate both the first derivative and the second derivative of $v_s$, and specify how $E_{edge}$ will be computed.
- How should the snake iterate from its initial position?

## How do we implement this?

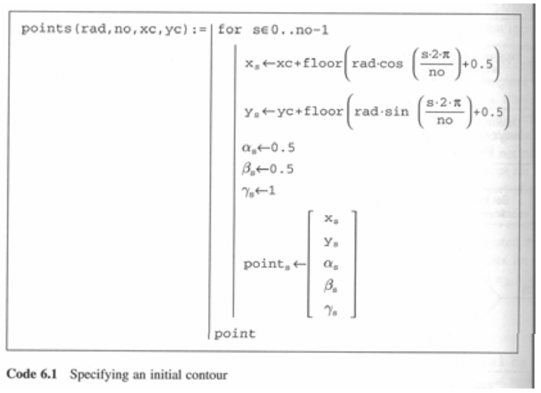
- The energy function involves finding the new location of S new coordinates $(x_s, y_s)$, $0 \leq s \leq 1$ for one iteration.
- Which algorithm can we use to find the new coordinate locations?
  1. Greedy algorithm
     - Simple, suboptimal, easier to understand
  2. Complete Kass algorithm
     - Optimizes all points on the countour simultaneously by solving a set of differential equations.
- These two algorithms will now be presented.

## The greedy algorithm for snakes


Define snake points and parameters α,β, γ
Start with first snake point
Initialize minimum energy and coordinates
Determine coordinates of neighbourhood point with lowest energy
Set new snake point coordinates to new minimum
More snake points?
Yes
No
Finish iteration

## Coordinates of the initial contour

- The starting point of the snake is the initial contour. It can e.g. be *no* (number of points) on a circle with radius *r*:



```
points(rad,no,xc,yc):= for s∈0..no-1
                           $x_s \leftarrow xc + \text{floor}\left(rad \cdot \cos\left(\frac{s \cdot 2 \cdot \pi}{no}\right) + 0.5\right)$
                           $y_s \leftarrow yc + \text{floor}\left(rad \cdot \sin\left(\frac{s \cdot 2 \cdot \pi}{no}\right) + 0.5\right)$
                           $\alpha_s \leftarrow 0.5$
                           $\beta_s \leftarrow 0.5$
                           $\gamma_s \leftarrow 1$
                           $point_s \leftarrow \begin{bmatrix} x_s \\ y_s \\ \alpha_s \\ \beta_s \\ \gamma_s \end{bmatrix}$
                        point
```

**Code 6.1** Specifying an initial contour

## Approximating the first derivative of $v_s$

Average distance between points on the contour

$$\left|\frac{dv_s}{ds}\right| = \left|\sum_{i=0}^{S-1} \|v_i - v_{i+1}\| / S\right|$$
$$= \left|\sum_{i=0}^{S-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} / S\right|$$

- When will this function be a minimum?
- Why is this a problem?

## Approximating the second derivative of $v_s$

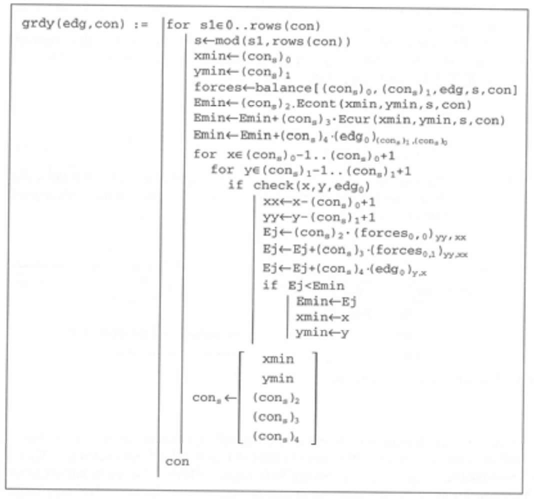
Why is this correct?
Hint: Check the derivation of the Laplace operator

$$\left|\frac{d^2 v_s}{ds^2}\right|^2 = \left|(v_{s+1} - 2v_s + v_{s-1})\right|^2$$
$$= (x_{s+1} - 2x_s + x_{s-1})^2 + (y_{s+1} - 2y_s + y_{s-1})^2$$

## Computing $E_{edge}$

- $E_{edge}$ can be implemented as the magnitude of the Sobel operator at point (x,y).
- The energy should be minimized, so we invert the edge image (maximizing a function f is equvalent to minimizing –f).
- Normalize all energy terms so that they have an output in the interval [0,1].

## The full greedy algorithm



```
grdy(edg,con) := for s1∈0..rows(con)
                     s←mod(s1,rows(con))
                     xmin←(con_s)_0
                     ymin←(con_s)_1
                     forces←balance[(con_s)_0,(con_s)_1,edg,s,con]
                     Emin←(con_s)_2.Econt(xmin,ymin,s,con)
                     Emin←Emin+(con_s)_3·Ecur(xmin,ymin,s,con)
                     Emin←Emin+(con_s)_4·(edg_0)_(con_s)_1,(con_s)_0
                     for x∈(con_s)_0-1..(con_s)_0+1
                        for y∈(con_s)_1-1..(con_s)_1+1
                           if check(x,y,edg_0)
                              xx←x-(con_s)_0+1
                              yy←y-(con_s)_1+1
                              Ej←(con_s)_2·(forces_0,0)_yy,xx
                              Ej←Ej+(con_s)_3·(forces_0,1)_yy,xx
                              Ej←Ej+(con_s)_4·(edg_0)_y,x
                              if Ej<Emin
                                 Emin←Ej
                                 xmin←x
                                 ymin←y
                                    ⎡ xmin     ⎤
                                    ⎢ ymin     ⎥
                     con_s ←        ⎢ (con_s)_2 ⎥
                                    ⎢ (con_s)_3 ⎥
                                    ⎣ (con_s)_4 ⎦
                  con
```

## Comments on the greedy algorithm

- Edge points can be allowed to form corners if points with large gradient magnitude and large change in direction (above a threshold) are not included in the summations.
- A threshold on the number of changes done in a single iteration can be used to avoid oscillations between two contours with very similar energy.
- If $\alpha$=0, contour points can have very different spacing.
- If $\beta$=0, points with high curvature can be allowed (this can be allowed locally if $\beta$ varies with s).
- If $\gamma$=0, we ignore the image and the position of the contour can be far from the real edge in the image.

## From the greedy algorithm to a full snake

- The greedy algorithm only finds the minimum energy for one point (x,y) on the snake at the time, and only points that are neighbors of current snake points are checked at a given iteration.
- A full algorithm should minimize the energy for <u>all</u> snake points $v_s$, s=1,S.

## The complete snake algorithm

- Derivation of the complete snake algorithm is only for those interested in mathematical details.
- The derivation is given, but not part of the curriculum.
- What we end up with, is a set of differential equations controlling how v(s) = (x(s),y(s)) change with time.

## The full snake equations

- Simple snake with only two terms (no termination energy):

$$E_{snake} = \int_{s=0}^{1} E_{\text{int}}(v(s)) + E_{edge}(v(s))ds$$

- We want to minimize this energy.
- Without $E_{edge}$, it will be minimum if v(s) has length 0.
- To avoid this, the internal forces must balance with $E_{edge}$

## The complete snake

- Assume that we seek an iterative solution.
- Assume that we have one solution

$$\hat{v}(s) = (\hat{x}(s), \hat{y}(s))$$

- If this solution is perturbated slightly by $\varepsilon\delta v(s)$, the solution that has minimum energy must satisfy:

$$\frac{dE_{snake}\left(\hat{v}(s) + \varepsilon\delta v(s)\right)}{d\varepsilon} = 0$$

The new solution should be a minimum, so the derivative must be 0.

- The slight spatial perturbation is defined as $\delta v(s)=(\delta_x(s), \delta_y(s))$.
- The perturbed snake solution is:

$$\hat{v}(s) + \varepsilon\delta v(s) = (\hat{x}(s) + \varepsilon\delta_x(s), \hat{y}(s) + \varepsilon\delta_y(s))$$

- After some derivation (see last slides) we get the following equation:

$$\int_{s=0}^{1} \left\{ -\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\} + \frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\} + \frac{1}{2}\int_{s=0}^{1} \left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}} \right\} \delta_x(s)ds = 0$$

- Because this must be true for all $\delta_x(s)$ the term within the outer {} must be zero:

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\} + \frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\} + \frac{1}{2}\int_{s=0}^{1} \left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}} = 0$$

- A similar derviation can be done for y(s). Thus, we have a pair of differential equations.
- A complete snake must solve these two equations.

## Snake differential equations

- The full snake consists of two differential equations, one for x(s) and one for y(s).
- We approximate the first order derivatives: $dx(s)/ds \cong x_{s+1}-x_s$
- And the second order derivatives:
  $d^2x(s)/ds^2 \cong x_{s+1}-2x_s+x_{s-1}$
- We discretize the contour into S (s=1,..,S) points with spacing h.
- The discrete equation is then:

$$-\frac{1}{h}\left\{\alpha_{s+1}\frac{x_{s+1}-x_s}{h} - \alpha_s\frac{x_s - x_{s-1}}{h}\right\}$$
$$+\frac{1}{h^2}\left\{\beta_{s+1}\frac{x_{s+2}-2x_{s+1}+x_s}{h^2} - 2\beta_s\frac{x_{s+1}-2x_s+x_{s-1}}{h^2} + \beta_{s-1}\frac{x_s-2x_{s-1}+x_{s-2}}{h^2}\right\}$$
$$+\frac{1}{2}\left.\frac{\partial E_{edge}}{dx}\right|_{x_s,y_s} = 0$$

- We can write this on the form

$$f_s = a_s x_{s-2} + b_s x_{s-1} + c_s x_s + d_s x_{s+1} + e_s x_{s+2}$$

where

$$f_s = -\frac{1}{2}\left.\frac{\partial E_{edge}}{dx}\right|_{x_s,y_s} \quad a_s = \frac{\beta_{s-1}}{h^4} \quad b_s = -\frac{2(\beta_s+\beta_{s-1})}{h^4} - \frac{\alpha_s}{h^2}$$

$$c_s = \frac{\beta_{s+1}+4\beta_s+\beta_{s-1}}{h^4} + \frac{\alpha_{s+1}+\alpha_s}{h^2} \quad d_s = -\frac{2(\beta_{s+1}+\beta_s)}{h^4} - \frac{\alpha_{s+1}}{h^2} \quad e_s = \frac{\beta_{s+1}}{h_4}$$

- This is also a matrix equation: $Ax=f_x(x,y)$ where $f_x(x,y)$ is the first order differential edge magnitude along the x-axis (horisontal gradient magnitude) and

$$A = \begin{bmatrix} c_1 & d_1 & e_1 & 0 & .. & a_1 & b_1 \\ b_2 & c_2 & d_2 & e_2 & 0 & .. & a_2 \\ a_3 & b_3 & c_3 & d_3 & e_3 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & & \\ e_{s-1} & 0 & .. & a_{s-1} & b_{s-1} & c_{s-1} & d_{s-1} \\ d_s & e_s & 0 & .. & a_s & b_s & c_s \end{bmatrix}$$

- The equivalent holds for y(s). So we have two equations

  $Ax=f_x(x,y)$

  $Ay=f_y(x,y)$
- These means that the snake energy should be balanced by the edge energy.
- We need an iterative approach to get a solution that is globally optimal (one single iteration by computing $A^{-1}$ gives a local optimal solution).
- An iterative solution must have snake points that depend on time, a snake that can move.
- Let $x^{<i>},y^{<i>}$ denote the solution at time i.

## Manipulating the equation

- We have:

$$Ax^{<i>} - fx(x^{<i>}, y^{<i>}) = 0$$
$$Ay^{<i>} - fy(x^{<i>}, y^{<i>}) = 0$$

- To solve these equations, set them equal to a small step size $\lambda$ times the negative time derivatives of the coordinates (also assume for simplicity that $f_x$ and $f_y$ are constant during one time step):

$$Ax^{<i+1>} - fx(x^{<i>}, y^{<i>}) = -\lambda(x^{<i+1>} - x^{<i>})$$
$$Ay^{<i+1>} - fy(x^{<i>}, y^{<i>}) = -\lambda(y^{<i+1>} - y^{<i>})$$

- If the solution is at an equilibrium, the right hand side will equal 0 and the original equation be fullfilled.
- Rewrite this as:

$$(A+\lambda I)x^{<i+1>} = f_x(x^{<i>}, y^{<i>}) + \lambda x^{<i>}$$
$$(A+\lambda I)y^{<i+1>} = f_y(x^{<i>}, y^{<i>}) + \lambda y^{<i>}$$

$$(A+\lambda I)x^{<i+1>} = f_x(x^{<i>}, y^{<i>}) + \lambda x^{<i>}$$
$$(A+\lambda I)y^{<i+1>} = f_y(x^{<i>}, y^{<i>}) + \lambda y^{<i>}$$
$$\Updownarrow$$
$$x^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda x^{<i>} + f_x(x^{<i>}, y^{<i>})\right)$$
$$y^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda y^{<i>} + f_y(x^{<i>}, y^{<i>})\right)$$

- The matrix $A+\lambda I$ is pentadiagonal banded and can be inverted fast using LU-decomposition.
- A whole set of contour points is found for each solution.

## Derivation of the snake equations

- The following slides are intended for those who like derivations.
- Deriving this is not part of the curriculum.

## Derivation of the snake equations

- Assume that we seek an iterative solution.
- Assume that we have one solution

$$\hat{v}(s) = (\hat{x}(s), \hat{y}(s))$$

- If this solution is perturbated slightly by εδv(s), the solution that has minimum energy must satisfy:

$$\frac{dE_{snake}\left(\hat{v}(s)+\varepsilon\delta v(s)\right)}{d\varepsilon} = 0$$

The new solution should be a minimum, so the derivative must be 0.

- The slight spatial perturbation is defined as $\delta v(s)=(\delta_x(s), \delta_y(s))$.
- The perturbed snake solution is:

$$\hat{v}(s)+\varepsilon\delta v(s) = (\hat{x}(s)+\varepsilon\delta_x(s), \hat{y}(s)+\varepsilon\delta_y(s))$$

## Derivation of the snake equations

- The snake equation is:

$$E_{snake} = \int_{s=0}^{1} E_{\text{int}}(v(s)) + E_{edge}(v(s))ds$$

- With a slight perturbation:

$$E_{snake}(\hat{v}(s)+\varepsilon\delta v(s)) = \int_{s=0}^{1} E_{\text{int}}(\hat{v}(s)+\varepsilon\delta v(s)) + E_{edge}(\hat{v}(s)+\varepsilon\delta v(s))ds$$

Insert the perturbated solution

- Insert the values derived for $E_{\text{int}}$ and $E_{edge}$:

$$E_{snake}(\hat{v}(s)+\varepsilon\delta v(s)) = \int_{s=0}^{1}\left\{\alpha(s)\left|\frac{d(\hat{v}(s)+\varepsilon\delta v(s))}{ds}\right|^2 + \beta(s)\left|\frac{d^2(\hat{v}(s)+\varepsilon\delta v(s))}{ds^2}\right|^2 + E_{edge}(\hat{v}(s)+\varepsilon\delta v(s))\right\}ds$$

## Derivation of the snake equations

- Separate into x(s) and y(s):

$$E_{snake}(\hat{v}(s)+\varepsilon\delta v(s)) = \int_{s=0}^{1}\left\{\begin{matrix} \alpha(s)\left\{\begin{matrix}\left(\frac{d\hat{x}(s)}{ds}\right)^2 + 2\varepsilon\frac{d\hat{x}(s)}{ds}\frac{d\delta_x(s)}{ds} + \left(\varepsilon\frac{d\delta_x(s)}{ds}\right)^2 \\ +\left(\frac{d\hat{y}(s)}{ds}\right)^2 + 2\varepsilon\frac{d\hat{y}(s)}{ds}\frac{d\delta_y(s)}{ds} + \left(\varepsilon\frac{d\delta_y(s)}{ds}\right)^2\end{matrix}\right\} \\ +\beta(s)\left\{\begin{matrix}\left(\frac{d^2\hat{x}(s)}{ds^2}\right)^2 + 2\varepsilon\frac{d^2\hat{x}(s)}{ds^2}\frac{d^2\delta_x(s)}{ds^2} + \left(\varepsilon\frac{d^2\delta_x(s)}{ds^2}\right)^2 \\ +\left(\frac{d^2\hat{y}(s)}{ds^2}\right)^2 + 2\varepsilon\frac{d^2\hat{y}(s)}{ds^2}\frac{d^2\delta_y(s)}{ds^2} + \left(\varepsilon\frac{d^2\delta_y(s)}{ds^2}\right)^2\end{matrix}\right\} \\ + Eedge(\hat{v}(s)+\varepsilon\delta v(s))\end{matrix}\right\}ds$$

## Derivation of the snake equations

- Use Taylor series expansion on $E_{edge}$:

$$E_{edge}(\hat{v}(s)+\varepsilon\delta v(s)) = E_{edge}(\hat{x}(s)+\varepsilon\delta_x(s), \hat{y}(s)+\varepsilon\delta_y(s))$$

$$= E_{edge}(\hat{x}(s), \hat{y}(s)) + \varepsilon\delta_x(s)\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}} + \varepsilon\delta_y(s)\left.\frac{\partial E_{edge}}{\partial y}\right|_{\hat{x},\hat{y}} + O(\varepsilon^2)$$

- $E_{edge}$ must be twice differentiable, which holds for edge information.

Taylor expansion of f(x+h)=f(x)+hf'(x)+h/2f''(x)+....
If ε is small, $\varepsilon^2$ can be neglected.

## Derivation of the snake equations

- Since $\varepsilon$ is small, ignore alle second order terms in $\varepsilon$ and reformulate $E_{snake}$:

$$
\begin{aligned}
&E_{snake}(\hat{v}(s)+\varepsilon\delta v(s)) = E_{snake}(\hat{v}(s)) \\
&+2\varepsilon\int_{s=0}^{1} \alpha(s)\frac{d\hat{x}(s)}{ds}\frac{d\delta_x(s)}{ds} + \beta(s)\frac{d^2\hat{x}(s)}{ds^2}\frac{d^2\delta_x(s)}{ds^2} + \frac{\delta_x(s)}{2}\frac{\partial E_{edge}}{\partial x}\bigg|_{\hat{x},\hat{y}} ds \\
&+2\varepsilon\int_{s=0}^{1} \alpha(s)\frac{d\hat{y}(s)}{ds}\frac{d\delta_y(s)}{ds} + \beta(s)\frac{d^2\hat{y}(s)}{ds^2}\frac{d^2\delta_y(s)}{ds^2} + \frac{\delta_y(s)}{2}\frac{\partial E_{edge}}{\partial y}\bigg|_{\hat{x},\hat{y}} ds
\end{aligned}
$$

## Derivation of the snake equations

- Since $\hat{v}(s)$ is a valid solution, it must be a local minimum and the two intergral terms must be zero:

$$
\int_{s=0}^{1} \alpha(s)\frac{d\hat{x}(s)}{ds}\frac{d\delta_x(s)}{ds} + \beta(s)\frac{d^2\hat{x}(s)}{ds^2}\frac{d^2\delta_x(s)}{ds^2} + \frac{\delta_x(s)}{2}\frac{\partial E_{edge}}{\partial x}\bigg|_{\hat{x},\hat{y}} ds = 0
$$

$$
\int_{s=0}^{1} \alpha(s)\frac{d\hat{y}(s)}{ds}\frac{d\delta_y(s)}{ds} + \beta(s)\frac{d^2\hat{y}(s)}{ds^2}\frac{d^2\delta_y(s)}{ds^2} + \frac{\delta_y(s)}{2}\frac{\partial E_{edge}}{\partial y}\bigg|_{\hat{x},\hat{y}} ds = 0
$$

To solve this integral, use the rule for partial integration

$$\int f(x)G(x)dx = F(x)G(x) - \int F(x)g(x)dx$$

with

$$G(x) = \alpha(s)\frac{d\hat{x}(s)}{ds} \quad \text{and} \quad f(x) = \frac{d\partial_x(s)}{ds}$$

## Derivation of the snake equations

- By integration we get:

$$
\cancel{\left[\alpha(s)\frac{d\hat{x}(s)}{ds}\delta_x(s)\right]_{s=0}^{1}} - \int_{s=0}^{1}\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\}\delta_x(s)ds
$$

$$
\cancel{\left[\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\frac{d\ \delta_x(s)}{ds}\right]_{s=0}^{1}} - \cancel{\left[\frac{d}{ds}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\}\delta_x(s)\right]_{s=0}^{1}}
$$

$$
+\int_{s=0}^{1}\frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\}\delta_x(s)ds + \frac{1}{2}\int_{s=0}^{1}\frac{\partial E_{edge}}{\partial x}\bigg|_{\hat{x},\hat{y}}\delta_x(s)ds = 0
$$

- As s goes from 0 to 1, we tranverse one full contour and end up at precisely the same point. Thus $\delta_x(1)-\delta_x(0)=0$ and $\delta_y(1)-\delta_y(0)$
- Because of this, the first, third and fourth term is zero.

## Derivation of the snake equations

- So we get:

$$
\int_{s=0}^{1}\left\{-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\} + \frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\} + \frac{1}{2}\int_{s=0}^{1}\frac{\partial E_{edge}}{\partial x}\bigg|_{\hat{x},\hat{y}}\right\}\delta_x(s)ds = 0
$$

- Because this must be true for all $\delta_x$(s) the term within the outer {} must be zero:

$$
-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\} + \frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\} + \frac{1}{2}\int_{s=0}^{1}\frac{\partial E_{edge}}{\partial x}\bigg|_{\hat{x},\hat{y}}
$$

- A similar derviation can be done for y(s). Thus, we have a pair of differential equations.
- A complete snake must solve these two equations.

## The practical part of using snakes

- The pratical part of the Kass snake algorithm
- Matlab implementation
- The capture range problem
  - Distance measure based solutions
  - Gradient vector flow field based solutions
- Briefly on alternative models:
  - Active shape models

## The Kass snake algorithm

- Initialize the snake by selecting an initial countour
- Compute the initial energy terms and the gradient.
- Select parameters $\lambda$, $\alpha$, $\beta$, h
  - $\alpha$ and $\beta$ can be functions of s, but this is difficult to estimate. In practice we ofter use scalar values for $\alpha$ and $\beta$
- Given the solution at iteration i $x^{<i>}$,$y^{<i>}$, compute $x^{<i+1>}$,$y^{<i+1>}$:

$$x^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda x^{<i>} + f_x(x^{<i>}, y^{<i>})\right)$$
$$y^{<i+1>} = (A+\lambda I)^{-1}\left(\lambda y^{<i>} + f_y(x^{<i>}, y^{<i>})\right)$$

## Capture range problems

- The snake must be initialized fairly close to the final target in order to get convergence.
- To make a really good initialization we need to have a very good estimate of the solution before starting the iterative process of adapting the snake.
- So we can find a good solution if we already know the solution. Obviously not very intersting...

- What do you think happens if you initialize the snake inside the structure you want to find?

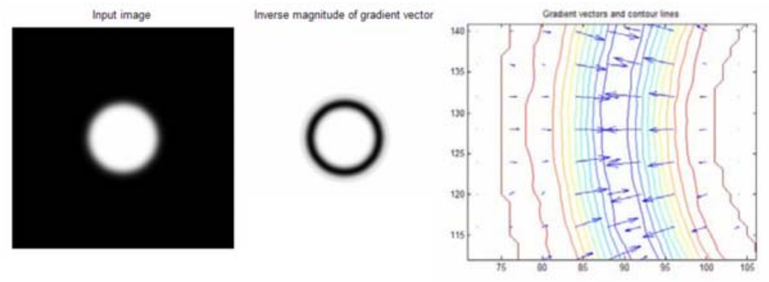
## Capture range problems

- The problem stems from the short ”range” of the external forces.
- The inverse magnitude of the gradient will have significant values only in the vicinity of the salient edges.
- This basically forces us to initialize the snake very close to the target contour.
- This problem is know as the **capture range problem.**

## Capture range problems

- One good way of visualizing this is by looking at the negative of the gradient of the external force field.
- These are the fources that pull the snake.
- The next slide shows this for a circle.
- Notice that outside the area in the immediate vicinity of the circle, these forces are negligible.

## Capture range problems

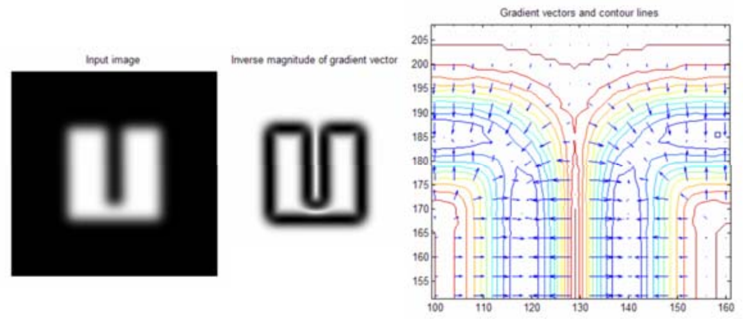


- Zoomed version of the gradient.
- Arrow lenght: gradient magnitude
- Arrow direction: gradient angle

## Capture range problems

- This phenomenon is also the reason why you will not get convergence into concavities, there are simply no forces to ”drag” the snake into the concavity.

## Capture range problems



## Capture range problems

- Many authors have suggested different methods for increasing the capture range of the external gradient vector field.
- We will look at a method suggested by Xu and Prince in: Snakes, Shapes and Gradient Vector Flow, IEEE Tr. Image Processing, vol. 7, no. 3, pp. 359-369, 1998.

## Capture range problems

- Xu and Prince observed that smoothing will increase the capture range, but will NOT provide convergence into concavities.
- Other methods, for instance those based on distance maps are even better at increasing capture range, but the concavity problem remains the same.
- Xu and Prince's solution is based on diffusing the gradient information.

## Capture range problems

- Remember the snake differential equations:

$$-\frac{d}{ds}\left\{\alpha(s)\frac{d\hat{x}(s)}{ds}\right\}+\frac{d^2}{ds^2}\left\{\beta(s)\frac{d^2\hat{x}(s)}{ds^2}\right\}+\frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}}=0$$

$$\Updownarrow$$

$$-\alpha(s)\frac{d^2\hat{x}(s)}{ds^2}+\beta(s)\frac{d^4\hat{x}(s)}{ds^4}=-\frac{1}{2}\int_{s=0}^{1}\left.\frac{\partial E_{edge}}{\partial x}\right|_{\hat{x},\hat{y}}$$

- The right hand side is the negative gradient of the external force field. The limited reach of this field is our problem.
- Xu and Prince's method consists of replacing this vector field with another one – a diffused version of the gradient field.

## Principle for gradient vector flow

- Xu and Prince considered an edge map ($-\nabla f$).
- The gradient of this edge map would point towards the edges.
- The idea of Xu and Prince is that this gradient of the edge map should be diffused to all other parts of the image in a smooth manner.
- In areas where $\nabla f$ is high, the gradient edge map should be close to $\nabla f$.
- In homogeneous areas in the image, the gradient edge map should be smooth (the flow should be small and without «curl» or turbulence in form of rapid directional changes.

## Capture range problems

- Xu and Prince define the vector field:

$$\mathbf{v}(x,y) = (u(x,y), v(x,y))^T$$

- It is **v** that will be the GVF.
- The field **v** is the field that minimizes the following functional:

$$G = \iint \mu\left(u_x{}^2 + u_y^2 + v_x^2 + v_y^2\right) + \left|\nabla f\right|^2 \left|v - \nabla f\right|^2 dxdy$$

- v(x,y) is found by solving this equation.
- μ is a parameter that controls the amount of smoothing.

## Capture range problems

$$G = \iint \mu\left(u_x{}^2 + u_y^2 + v_x^2 + v_y^2\right) + \left|\nabla f\right|^2 \left|v - \nabla f\right|^2 dxdy$$

- The goal is to minimize G.
- The second term will have a minimum if v=∇f.
- If ∇f is small, the first term will dominate.
- From calculus of variation, this can be written as

$$\mu\nabla^2 u - \left(u - f_x\right)\left(f_x^2 + f_y^2\right) = 0$$
$$\mu\nabla^2 v - \left(v - f_y\right)\left(f_x^2 + f_y^2\right) = 0$$

- If ∇f is small, what remains is Lagrange's equation:

$$\mu\nabla^2 u = 0$$
$$\mu\nabla^2 v = 0$$

## Capture range problems

$$\mu\nabla^2 u = 0$$
$$\mu\nabla^2 v = 0$$

- The first term is Lagrange's equation which appear in often in physics, e.g. in heat flow or fluid flow.
- Imaging the a set of heaters is initialized at certain boundary conditions. As time evolves, the heat will redistribute/diffuse until we reach an equilibrium.
- In our setting, the gradient term act as the starting conditions.
- As the differential equation iterate, the gradient will diffuse gradually to other parts of the image in a smooth manner.

## Capture range problems

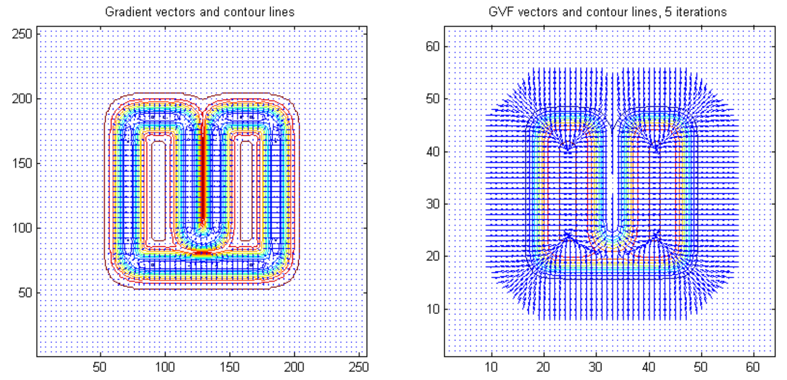
- The first term will smooth the data, that is, far from edges the field will be kept as smooth as possible by imposing that the spatial derivatives be as small as possible.
- When |∇f| is small, the vector field will be dominated by the partial derivatives of the vector field, yielding a smooth field.
- Close to edges (where |∇f| is large) the field is forced to resemble the gradient of f itself.
- So **v** is smooth far from edges and nearly equal to the gradient of f close to edges.
- The term μ just defines the weight we give the different terms in the functional.
- The field **v** is computed iteratively

## Capture range problems

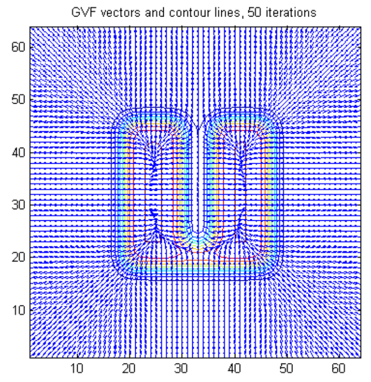
- This equation has a similar solution to the original differential equation.
- We treat u and v as functions of time and solve the equations iteratively.
  - Comparable to how we iteratively computed $x^{<i+1>}, y^{<i+1>}$ from $x^{<i>}, y^{<i>}$
- The solution is obviously a numerical one, we use two sets of iterations, one for u and one for v.
- After we have computed v(x,y), we replace $E_{ext}$ (the edge magnitude term) by v(x,y)
- So an interative algorithm is first used to compute v(x,y)

## Now back to the U-shape

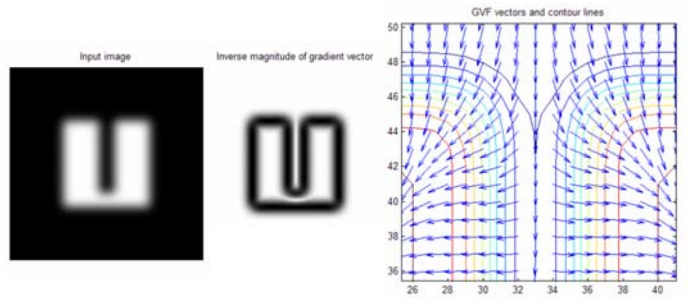
- Start with the gradient vector field and diffuse it over the image as we iterate



## **v** after 50 iterations



## Capture range problems



## Capture range problems

- Xu and Prince provide extensive material on their web address. This is an excellent site for further exploration of the GVF approach to solving the capture range problem.
- http://iacl.ece.jhu.edu/projects/gvf/
- In order to run the GVF examples you must place all GVF matlab files provided at this address in a directory where matlab will find them.

## Alternatives to the snake algorithm

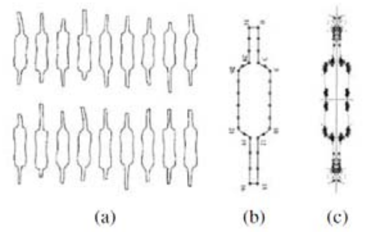
- Snakes have many degrees of freedom in their shape and can sometimes be trapped in local minima.
- One solution: control the snake more strict by using a B-spline for the contour. This is called a B-snake:

$$f(s)=\sum_k B_k(s)x_k$$
$$F=BK$$
$$F=\begin{bmatrix} f^T(0)\\ \vdots \\ f^T(N)\end{bmatrix}\quad B=\begin{bmatrix} B_0(s_0) & \cdots & B_K(s_0)\\ \vdots & \ddots & \vdots \\ B_0(s_N) & \cdots & B_K(s_N)\end{bmatrix},\text{ and }\quad X=\begin{bmatrix} x^T(0)\\ \vdots \\ x^T(K)\end{bmatrix}$$

- If we have additional information about local variations in location, scale or orientation, these can be modelled as additional tranformations of x.

## Shape priors

- Shape prior is an approach to model the statistical variance of certain boundary points on a contour based on a set of training sample.
- Consider the objects in a). Control points are selected on the contour.
- In c) we visualize the variance of each control point over the training set.



- Now, add a prior model for the location $x_k$ and its covariance $C_k$.
- A penalty function (or prior in the Bayesian terminology) would be:

$$E_{loc}(x_k)=\frac{1}{2}\left(x_k-\bar{x}_k\right)^T C_k^{1}\left(x_k-\bar{x}_k\right)$$

- Neighboring points often have correlated motion.
- How do we estimate the covariance matrices???

## Active shape models

- Concatenate all control points into a long vector x.
- The distribution of x can be described by its mean and covariance matrix of the P training samples:

$$\bar{x} \quad C = \frac{1}{P}\sum_p (x_p - \bar{x})(x_p - \bar{x})^T$$

- The main components of C can be found using eigenvector decomposition, by Principal component analysis where $\Phi$ is the largest eigenvectors and b is a shape parameter vector (they can also be restricted).

$$x = \bar{x} + \Phi b$$

## From a single image to a sequence of images…

- So far, we estimate the boundary in one single image.
- In a later lecture, we will study tracking of objects.
- Similar principles can then be applied to regularize the motion of the contour in time.

## Learning goals - snakes

- Understand simple energy functions
- Understand the snake energy function
- Know the greedy algorithm
- Know that the full snake is solved using iterative differential equations
- Know the limitations of snakes and how to solve them

## Next lecture

- Regularized models for segmentation/classification of ALL pixels in a scene.
- Example: contextual classification using Markov random field models.
- Material partly from:
  - 3.7.2 in Szeliski
  - 5.3-5.5 in Szelisk