INF 5300 – Flexible shape extraction

Anne Solberg (anne@ifi.uio.no)

Next two lectures:

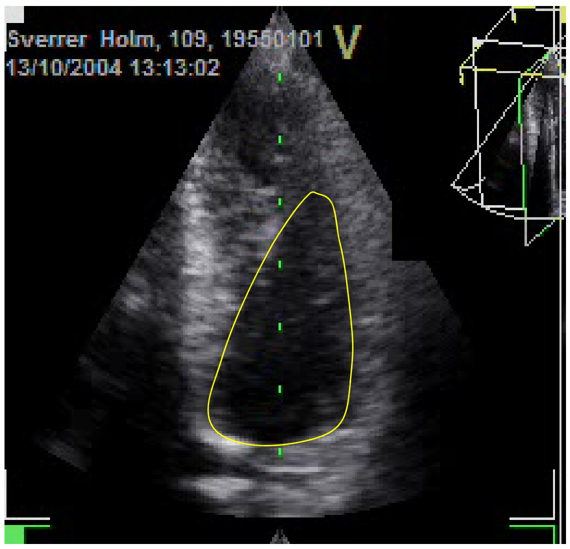
•Example: finding the border of the left ventricle

• Deformable templates

•Snakes

•Active shape models

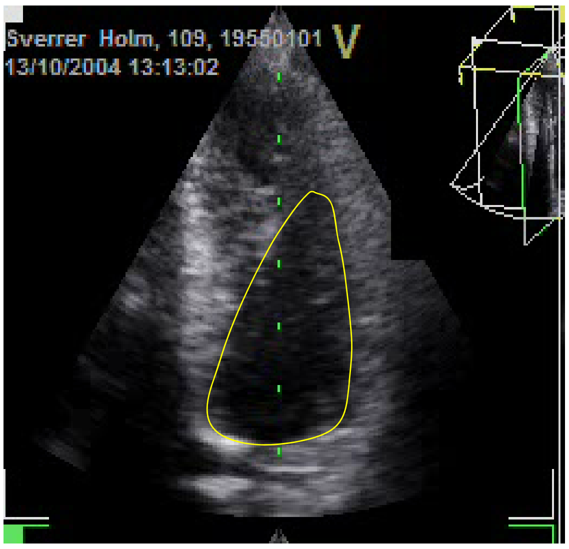
Example – segmenting ultrasound images of the hearth



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?

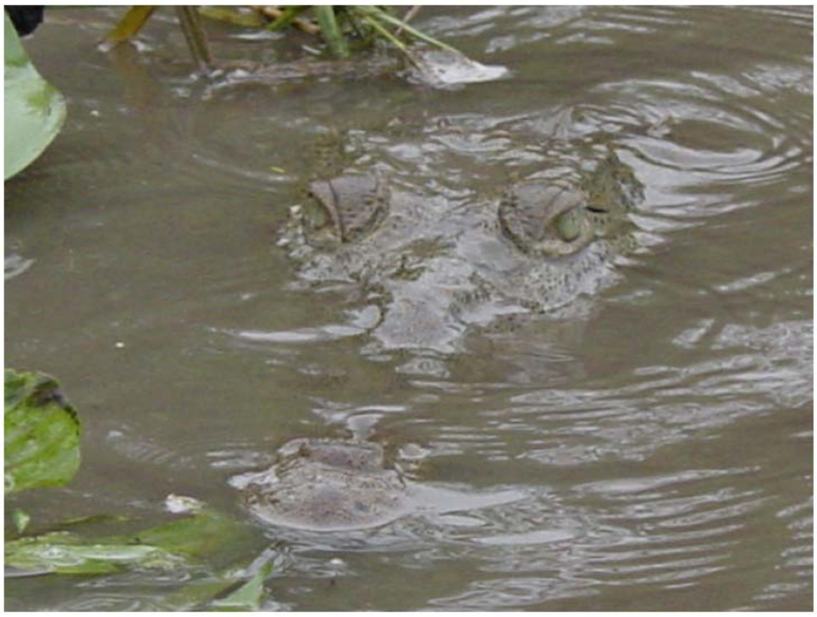


- Thresholding?
- Hit and miss?
- Region growing?
- Edge-based segmentation?
- Watershed?
- Line detection?
- Hough transform?
  - Ellipse?
  - Can be extended to general shapes if the precise mathematical description of the shape is known.

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.

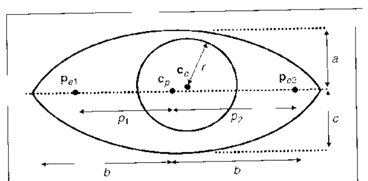
Introduction to deformable templates and energy functions

- Consider detection of the eye.
- An eye template consists a circle (the pupil) inside a closed contour of two parabolas.
- Parabola:

$$y = a - \frac{a}{b^2}x^2$$

- Find the values of the parameters {c_p, a, b, c, c_c, r} that best fits the image, in the sense that they maximize an energy function

$$\sum_{(x,y)\in circle.perimeter, parabola.perimeter} E_{x,y}$$



(a) Eye template

Introduction to deformable templates

- First, define an edge energy. The sum is over all edge points that are either on the parabolas or on the circle, and normalized by the nof. points on the contour.

$$Ee = \frac{\sum_{(x,y)\in circle.perimeter} E_{x,y}}{circle.perimeter} + \frac{\sum_{(x,y)\in parabola.perimeter} E_{x,y}}{parabola.perimeter}$$

- Assume also that the iris is darker than the sclera (the white area). Let $P_{x,y}$ be the gray levels.
- Consider backscatter energy for the iris:

$$Ev = -\frac{\sum_{(x,y)\in circle} P_{x,y}}{circle.area}$$

- Since the iris is dark, a minus sign is used to create a function that has large values for pixels with small gray levels.

- For the white regions, we compute the average pixels value of pixels inside the parabola, but outside the circle:

$$Ep = \frac{\sum_{(x,y)\in parabola-circle} P_{x,y}}{parabola.area - circle.area}$$

Combining the energy functions:

- Combine the three terms into:

$$E = c_e E_e + c_v E_v + c_p E_p$$

- c_p, c_v, and c_e are weights that influence the weighting of the different energy terms.
- Parameters to estimate: 8 shape parameters, 3 weights
  - How do we optimize all 11 parameters?
  - Suboptimal solutions can be found using genetic algorithms, but simpler models with fewer parameters are more popular.

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 energy 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:

$$v(s) = (x(s), y(s))^T$$

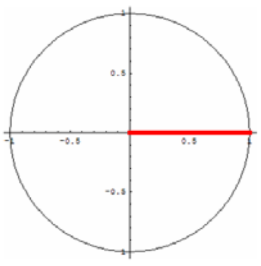
- This is a parametric representation of the contour.
- The vector describing the position of every point on the contour makes one pass over the entire contour as *s* varies from its mimimum to its maximum value.
- Typically, *s* is normalized

$$s \in [0,1]$$

- 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(2πs) and y(s)=sin(2πs)
- 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

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

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)

- The minimum values is found by derivation:

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

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.

- α and β are penalty parameters that control the weight of the two terms.
- Low α values: the snake can stretch much.
- Low β 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.
- In general, E_{image} is an integral over the curve (we will later discretize the curve)

$$E_{image} = \int_0^1 P(v(s))ds$$

The image term

$$E_{image} = \int_0^1 P(v(s))ds$$

- P(v(s)) denotes a scalar potential function defined on the image plane.
- We choose P(x,y) 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 intensity for point (x,y), what kind of structure does this function attract the snake to?

$$P(x,y) = -I(x,y)$$

- A common way of defining P(x,y) is:

$$P(x,y) = -c\left|\nabla G_\sigma * I(x,y)\right|$$

The energy function

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

$$E_{snake}(s) = E_{\text{int}}(v_s) + E_{image}(v_s)$$
$$= \alpha\left|\frac{dv_s}{ds}\right|^2 + \beta\left|\frac{d^2 v_s}{ds^2}\right|^2 + \gamma E_{edge}$$

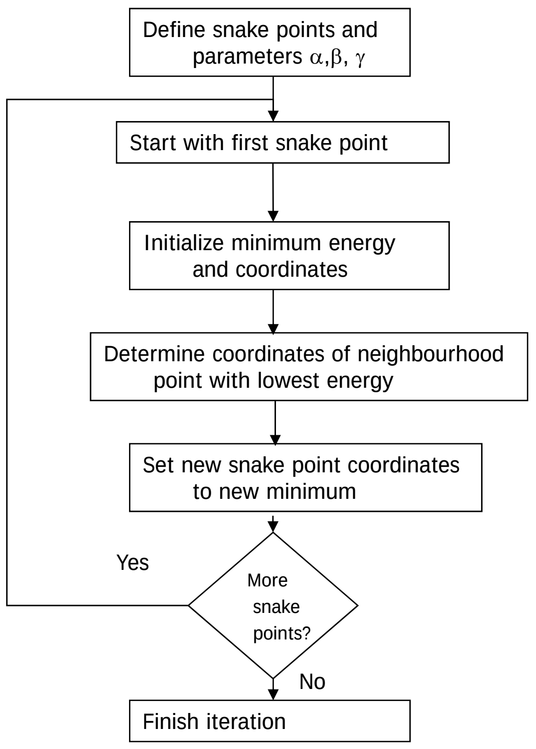
- We need to approximate both the first derivative and the second derivative of v_s, and specify how E_{edge} will be computed.
- How should the snake iterate from its initial position?

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.

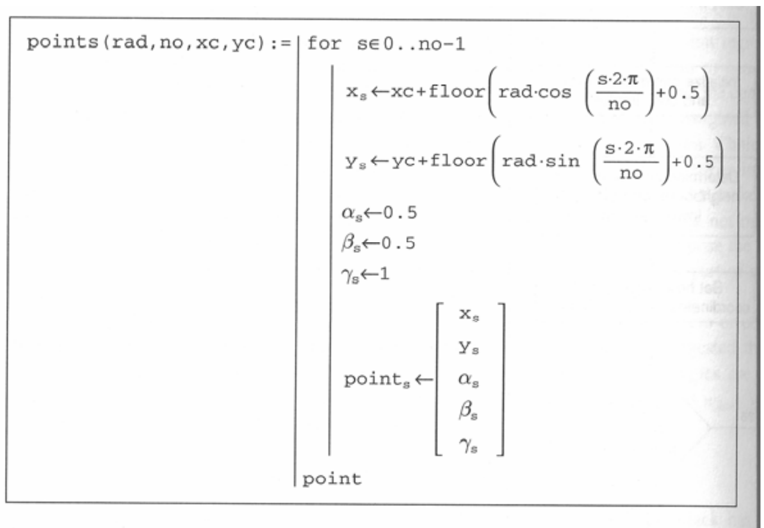
– These two algorithms will now be presented.

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 ← xc+floor(rad·cos(s·2·π / no)+0.5)
                       |   | y_s ← yc+floor(rad·sin(s·2·π / no)+0.5)
                       |   | α_s ← 0.5
                       |   | β_s ← 0.5
                       |   | γ_s ← 1
                       |   | point_s ← [x_s, y_s, α_s, β_s, γ_s]
                       | point
```

Code 6.1 Specifying an initial contour

Approximating the first derivative of v_s

Average distance between points on the contour

Distance between this point and the next point

$$\left|\frac{dv_s}{ds}\right|^2 = \left|\sum_{i=0}^{S-1}\|v_i - v_{i+1}\| / S - \|v_s - v_{s+1}\|\right|$$

$$= \left|\sum_{i=0}^{S-1}\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} / S - \sqrt{(x_s - x_{s+1})^2 + (y_s - y_{s+1})^2}\right|$$

Approximating the second derivative of v_s

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

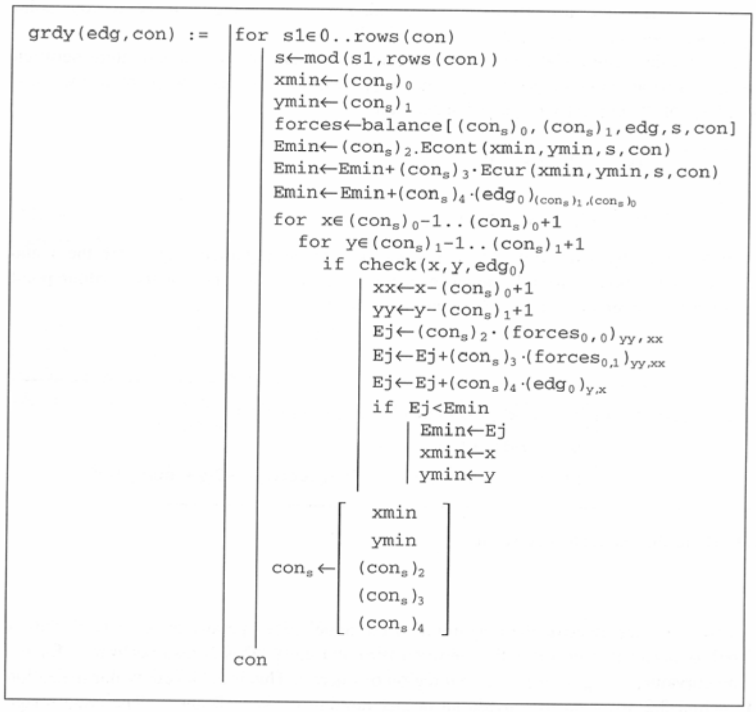
$$\left|\frac{d^2 v_s}{ds^2}\right|^2 = \left|(v_{s+1} - 2v_s + v_{s-1})\right|^2$$

$$= (x_{s+1} - 2x_s + x_{s-1})^2 + (y_{s+1} - 2y_s - y_{s-1})^2$$

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 inteval [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
                 |   con_s←[ xmin, ymin, (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 β 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 - derivation

- 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:

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

- 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_{int} and E_{edge}:

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

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

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

- 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, ε^2 can be neglected.

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

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

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

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

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

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

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

- By integration we get:

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

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

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

- 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.

- So we get:

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

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

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

- 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.

Solving the differential equations

- So, we have two differential equations.
- We approximate the first order derivatives: dx(s)/ds≅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:

$$
\begin{aligned}
&-\frac{1}{h}\left\{\alpha_{s+1}\frac{x_{s+1}-x_s}{h}-\alpha_s\frac{x_s-x_{s-1}}{h}\right\}\\
&+\frac{1}{h^2}\left\{\beta_{s+1}\frac{x_{s+2}-2x_{s+1}+x_s}{h^2}-2\beta_s\frac{x_{s+1}-2x_s+x_{s-1}}{h^2}+\beta_{s-1}\frac{x_s-2x_{s-1}+x_{s-2}}{h^2}\right\}\\
&+\frac{1}{2}\frac{\partial E_{edge}}{dx}\bigg|_{x_s,y_s}
\end{aligned}
$$

- 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

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

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

- 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 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 l 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:

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

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

$$\Updownarrow$$

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

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