

# An Introduction to Geometric Snakes in Image Processing

YingLiang Ma

System Biology Research Group

University of Ulster

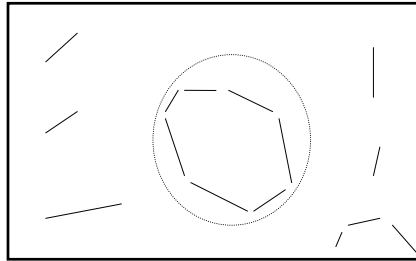
May, 2006

## Why Snakes?

1. They are popular methods to extract noisy object boundaries from biomedical images.
2. They create smooth and optimised boundaries without too much user interaction.
3. They can embed many statistical image processing methods such as edge enhance, region probability statistics, filters and more.

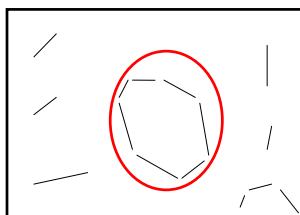
## Introduction

1. Suppose we wish to find the boundary of an object in an image.
2. One approach is to find edge segments and links them together.

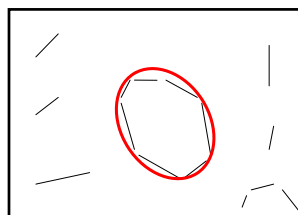


## Introduction (Cont.)

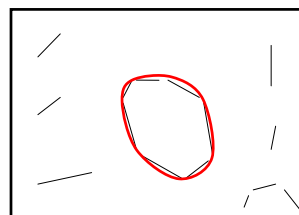
- Alternatively use a snake (Active Contour Model).
  - A smooth curve which matches to image data
  - Curve is initialised near the target
  - Iteratively refined



Initial position



Intermediate position



Final position

## Formulation of Snakes

$$E_{snakes} = E_{internal} + E_{img} + E_{constraints}$$

Internal energy smooths the shape of curve.

External energy encourage matching to image features.

- e.g. strong edges
- region homogeneity.

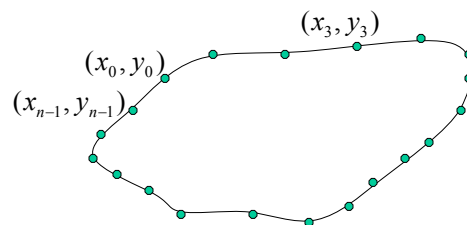
Constraint energy give the opportunities of user interactions.

## Point Representation

- Snake can be represented by a list of points.

$$P_i = (x_i, y_i)$$

$$i = 0, \dots, n-1$$



## Internal Energy

For a curve represented as a set of points.

- The elastic (length) energy can be expressed as

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

This energy encourages the curve to shrink to a single point.

## Internal Energy (Cont.)

For a curve represented as a set of points

- The stiffness (curvature) energy can be expressed as

$$\begin{aligned} E_{in\_stiffness} &= \beta \sum_{i=1}^{n-2} |P_{i+1} - 2P_i + P_{i-1}|^2 \\ &= \beta \sum_{i=1}^{n-2} (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2 \end{aligned}$$

Minimizing this energy can smooth the curve. (reduce bends)

## Equation Explanation (1)

$$E_{in\_stiffness} = \beta \int_0^N |P''(t)|^2 dt = \beta \int_0^N (|x''(t)|^2 + |y''(t)|^2) dt$$

Express in discrete representation:

$$E_{in\_stiffness} = \beta \sum_{i=0}^{n-1} |P_{i+1} - 2P_i + P_{i-1}|^2$$

$$\begin{array}{l} \frac{dP_i(t)}{dt} \approx P_{i+1} - P_i \\ \frac{dP_{i-1}(t)}{dt} \approx P_i - P_{i-1} \end{array} \quad \Rightarrow \quad \frac{d^2 P_i(t)}{dt^2} \approx \frac{dP_i(t)}{dt} - \frac{dP_{i-1}(t)}{dt}$$

## External Energy

- The external energy describes how well the curve matches the image data locally.
- Numerous forms can be used, attracting the curve toward different image features.

1. Gradient energy form.

$$E_{ext\_grad} = -\lambda_1 \sum_{i=0}^{n-1} (|G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2)$$

## External Energy (Cont.)

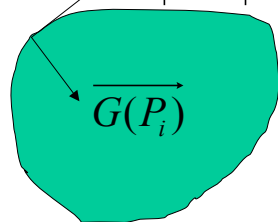
### 2. Directed gradient energy

$$E_{ext\_dirgrad} = -\lambda_1 \sum_{i=0}^{n-1} |u_{x,i} G_x(P_i) + u_{y,i} G_y(P_i)|$$

$$\vec{u}_i = \frac{\vec{P(t_i)}}{|\vec{P(t_i)}|} \quad E_{ext\_dirgrad} = -\lambda_1 \sum_{i=0}^{n-1} \vec{u}_i \bullet \vec{G(P_i)}$$

• is dot product

Directed gradient energy uses the direction of the gradient.



## Constraint energy

### 1. Shape constraints

To approximate the object boundaries by using prior knowledge.

In general, constraint energy can be expressed as

$$E_{constraints} = \eta \sum_{i=0}^{N_c-1} \min_{t \in [0, N]} |P(t) - P_{c,i}|^2$$

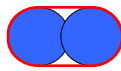
Where  $P_{c,i}; i = 0, \dots, N_c - 1$  are the point constraints.

## Find the Right Parameters

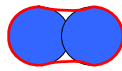
For internal energy

$$E_{internal} = \alpha E_{in\_elasticity} + \beta E_{in\_stiffness}$$

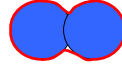
keeping  $\beta$  constant and decreasing  $\alpha$  will increase the stiffness of the snake curve.



large  $\alpha$



medium  $\alpha$

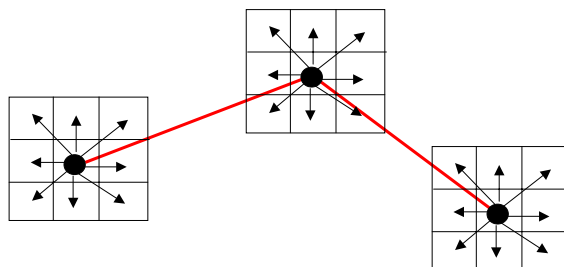


small  $\alpha$

## Optimization

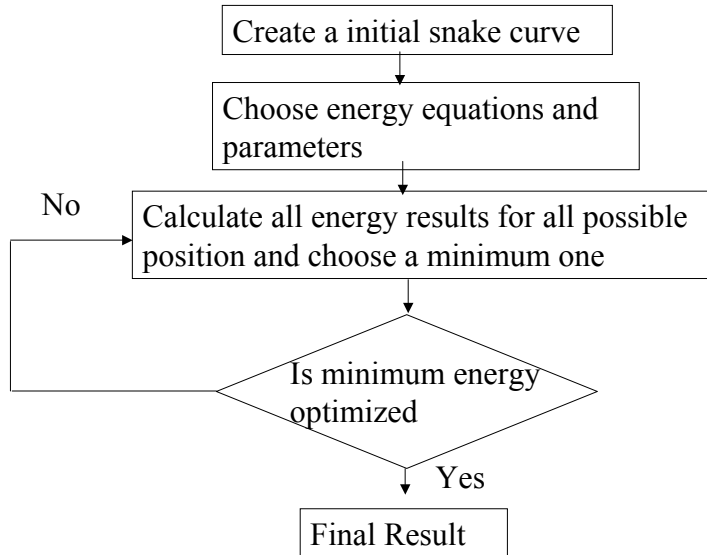
Goal: To find a smooth curve with minimum energy.

Solution: The easiest way is the heuristic approach.



Complexity:  $O(nm^2)$    
 n is number of point   
 m is number of position   
 directions in one axis.

## Flow Chart



## C++ Implementation

```
num_states=9; //3x3 states
MoveXY[0][0]=-1; MoveXY[0][1]=-1;
MoveXY[1][0]=-1; MoveXY[1][1]=0;
MoveXY[2][0]=-1; MoveXY[2][1]=1;
MoveXY[3][0]=0; MoveXY[3][1]=-1;
MoveXY[4][0]=0; MoveXY[4][1]=0;
MoveXY[5][0]=0; MoveXY[5][1]=1;
MoveXY[6][0]=1; MoveXY[6][1]=-1;
MoveXY[7][0]=1; MoveXY[7][1]=0;
MoveXY[8][0]=1; MoveXY[8][1]=1;
num_points=myPolyLine->GetPointCount();
en_arr=(int *)malloc(num_points*num_states*sizeof(int));
```

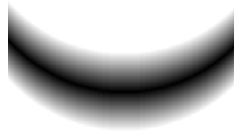


## Calculation Internal Energy

```
in_energy=0;
for(i=1;i<(num_points-1);i++)
{
    myPolyLine->GetPoint(i-1,Pt1);
    myPolyLine->GetPoint(i,Pt2);
    myPolyLine->GetPoint(i+1,Pt3);
    if((i-1)==j) {Pt1[0]+=MoveXY[k][0]; Pt1[1]+=MoveXY[k][1]; }
    if(i==j) {Pt2[0]+=MoveXY[k][0]; Pt2[1]+=MoveXY[k][1]; }
    if((i+1)==j) {Pt3[0]+=MoveXY[k][0]; Pt3[1]+=MoveXY[k][1]; }
    in_energy+=((Pt3[0]-Pt1[0])*(Pt3[0]-Pt1[0])+(Pt3[1]-Pt1[1])*(Pt3[1]-Pt1[1]))/2;
    in_energy+=((Pt3[0]-2*Pt2[0]+Pt1[0])*(Pt3[0]-2*Pt2[0]+Pt1[0]));
    in_energy+=((Pt3[1]-2*Pt2[1]+Pt1[1])*(Pt3[1]-2*Pt2[1]+Pt1[1]));
}
```

## Calculation External Energy

```
img_energy=0;
for(i=0;i<num_points;i++)
{
    myPolyLine->GetPoint(i,Pt1);
    if(i==j) {Pt1[0]+=MoveXY[k][0]; Pt1[1]+=MoveXY[k][1]; }
    pixel=pGImage->data+3*((PWidth*Pt1[1])+Pt1[0]);
    img_energy+=pixel[0];
}
en_arr[j*num_states+k]=(int)(0.01*in_energy+img_energy);
```



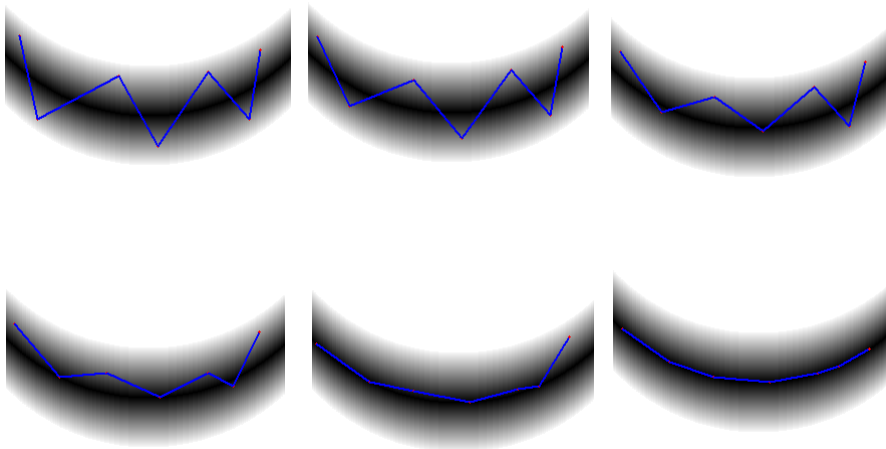
Demo  
Image

Demo only, have to be replaced by real image energy.

## Find the Optimized One

```
min_en=en_arr[0]; min_p=0; min_st=0;
for(j=0;j<num_points;j++)
    for(k=0;k<num_states;k++)
    {
        if(en_arr[j*num_states+k]<min_en)
        {
            min_p=j; min_st=k; min_en=en_arr[j*num_states+k];
        }
    }
```

## Demo Pictures



# B-Spline Snake

Why B-Spline Snake?

Advantages:

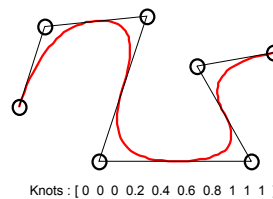
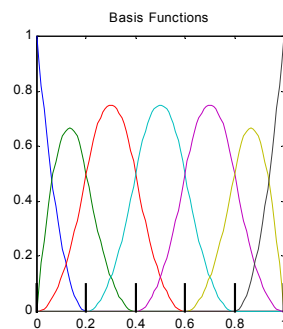
1. Creating a smooth snake.
2. Reducing the computation cost for optimization.
3. Local shape control.

Disadvantage:

Have to design the control points which are not on the curve.

## B-Spline Curve Definition

$$C(t) = \sum_{i=0}^{n-1} P_i N_{i,p}(t)$$



## B-Spline Blending Functions

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t)$$

Knot vector  $T = \{t_0, t_1, \dots, t_m\}$   $m = n + p + 1$

## Derivative of B-Spline Curve

$$C'(t) = \sum_{i=0}^{n-1} P_i N'_{i,p}(t)$$

$$N'_{i,p}(t) = \frac{p}{t_{i+p} - t_i} N_{i,p-1}(t) - \frac{p}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t)$$

## Another Form for Cubic B-Spline

$$C_i(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{2} & -1 & \frac{1}{2} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

where  $i = 1, \dots, n-3$   $0 \leq t \leq 1$

N control points have N-3 curve segments.

## Another Form (Cont.)

$$C(t) = \sum_{i=0}^{n-1} P_i B_4(t-i+3)$$

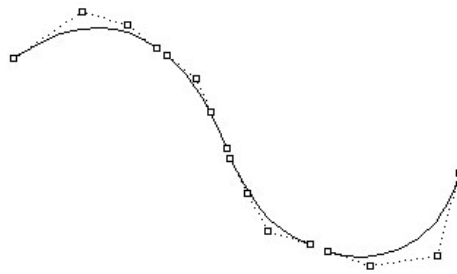
$$B_4(t) = \frac{1}{6} \begin{cases} t^3 & 0 \leq t < 1 \\ v(2-t) & 1 \leq t < 2 \\ v(t-2) & 2 \leq t < 3 \\ (4-t)^3 & 3 \leq t < 4 \end{cases}$$

where  $v(s) = (3s^3 - 6s^2 + 4)$

## B-Spline Snake Energy

$$E_{in\_elasticity} = \alpha \int_0^1 C(t) dt$$

$$E_{in\_stiffness} = \beta \int_0^1 |C''(t)|^2 dt$$



## Demos and Codes

- Snake demo
  - <http://www.isbe.man.ac.uk/~bim/software>
- Matlab snake demo (tested)
  - <http://www.slaney.org/malcolm/pubs.html>
- ImageJ Plugin (Java)  
<http://ip.beckman.uiuc.edu/Software/SplineSnake/>
- B-Spline Curve Library (C++)  
<http://research.bioinformatics.ulster.ac.uk/~yma/Software.html>

## Suggested Reading

Mathews Jacob, Thierry Blu and Michael Unser, “Efficient Energies and Algorithms for Parametric Snakes”, IEEE Transactions on Image Processing, Vol 13 (9) 2004.

Amir Amini, Terry Weymouth and Ramesh Jain, “Using Dynamic Programming for Solving Variational Problems in vision.

For more, just use “active contour model snake” or “B-Spline Snake” to search in Google.