Report 4

Snake: Active Contour Models

MICHAEL KASS, ANDREW WITKIN, and DEMETRI TERZOPOULOS

Schlumberger Palo Alto Research, 3340 Hillview Ave, Palo Alto, CA 94304

As I mentioned in my last report that Snake are active contour models that help to determine features like edges, lines and subjective contours while in motion of an image. Snake achieves this goal by locking onto nearby edges while decreasing the energy functional associated to them. In this paper, Snakes have been implemented interactively by applying user-imposed forces which can guide them to the nearby features in the image. We will discuss about these in more detail in the below sections.

**How Energy Functional Works:**

A snake is initialized around an image boundary either with the help of a user or a high level process. To guide a snake to a nearby feature in an image, an energy functional is designed which tries to minimize itself continuously and thus locking the snake onto nearby image. The energy functional of a snake looks like:



Where,

**Esnake** = Total Energy of the snake.

**Einternal** = Internal energy of the spline due to bending.

**Eexternal** = External energy of the image forces acting on spline.

**Econstraint** = External constraint energy imposed by user.



Internal Energy of the snake is in-turn comprised of two terms,

**Econt** = Energy of the snake contour.

**Ecurv** = Energy of the due to curvature of spline.

Also, Internal Energy can be written in mathematical form as below:

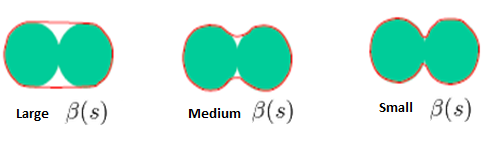


Stiffness, increases if the curve bends more

Elasticity, increases if the snake stretches more

Where, large value of α(s) increases the energy of snake as it stretches more and more and β(s) increases the energy of snake as it develops more curves.

These energy functions try to find local minima instead of global minima which comprises of a set of alternative solutions to be used for high level interpretation. This is done because we don’t want to make rigid and irreversible low level decisions which can in turn affect the high level interpretation.



**How image forces works:**

There are three different energy function of an image which attracts a snake towards itself.



Weights associated with different energy function determine which feature of the image will be more dominant to attract the snake towards itself i.e.

Line: Attracts towards either light or dark lines

Edge: Attracts towards contours with large image gradient

Term: Attracts towards corners and terminations.

**Snake Pit: ()**

|  |  |
| --- | --- |
|  | **One of the major novelties of this paper** After all as the system is interactive, even if there is a lack of high level mechanism, it is possible for the user to push the model out of local minima towards to desired result interactively. This is implemented using Snake Pit which is user interface designed on symbolic LISP machine that allows a user to initialize a snake and exert external forces on them. Once User pushes the snake near a feature, the energy minimization pulls the snake towards the image. Constraint forces of the energy functional can be applied using this interface. The Volcano can be used to move snake from one local minima to another one. |

**Applications of Snakes- Stereo and Motion:**

One of the other contributions or novelties of this paper is stereo-matching and motion detection.

Stereo matching is attained by using two contour models where energy of the snake is the square of the difference of the two contours. If the two contours are similar to each other the energy function minimizes. 3D model can be constructed from two different contours if the stereo matching is successful.

For motion detection, Snakes gets locked to an image by attaining the local minima and even if the image moves, Snake tries to track the once attained local minima, hence tracking the image movement.

This paper opens up many possible opportunities of research in image processing like segmentation, motion detection, shape recognition, stereo matching and may be many more. All these can be implemented by “snakes” by modifying the approach in slightly different way. Thus one gets a lot of opportunities to implement it different areas. Also these implementations are of real world usage and hence this paper is worth understanding.

**Algorithm formulation**

To formulate all this into an algorithm, we have to do a lot of mathematics. Our major area of concerns would be:

1. How to represent a snake shape, size etc.

2. How to represent energy functionals to work with them.

3. How to minimize the total energy.

We can represent a snake mathematically. Considering a snake a spline, we can represent points of that spline as V(s).

V(s) = (x(s), y(s)) 0 ≤s≤1

If the curvature is represented by n points

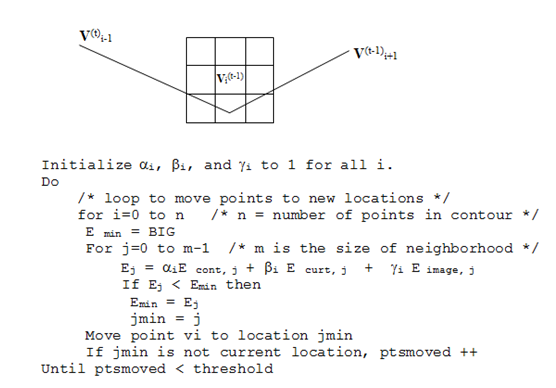
Where i=0….n-1

We can then write first order and second order derivatives of V(s) with respect to S, and as seen in the internal energy functional of snake above,

**≈ ( – ()) =**

Esnake = ∫0 1 (α(s)Econt + β(s)Ecurv + γEimage) ds

**Algorithm**

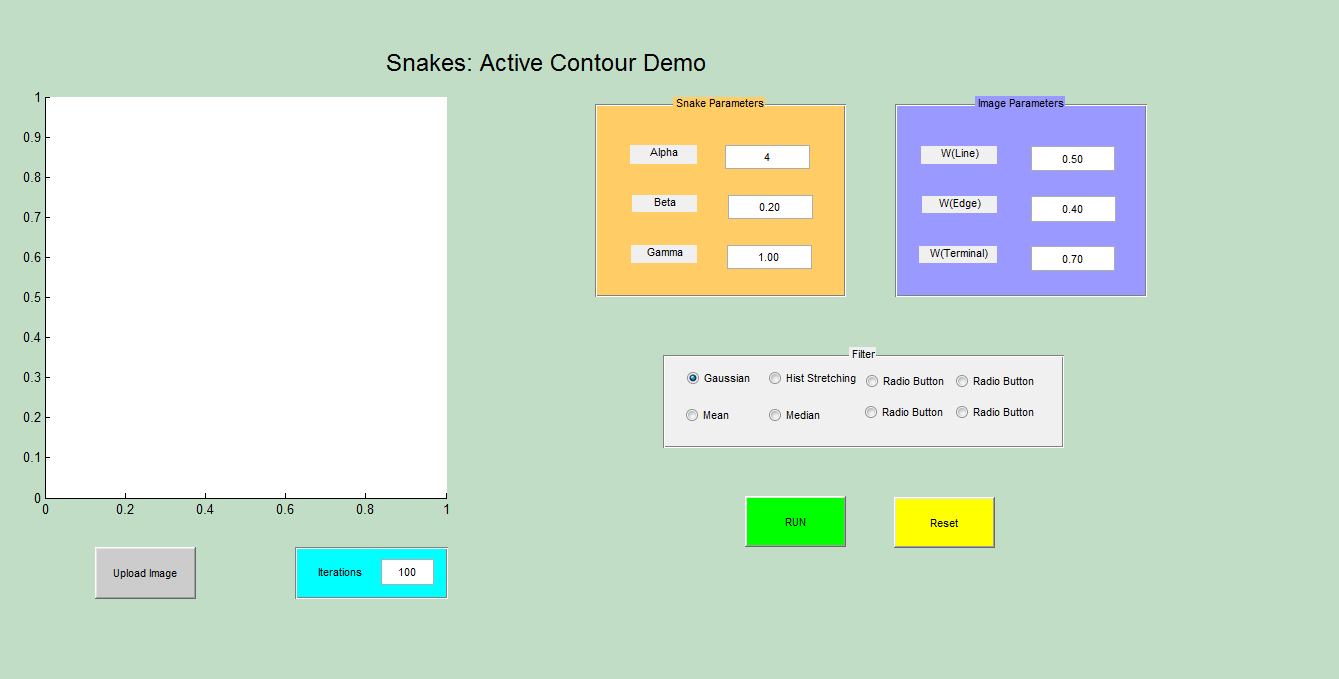


As we can see that above is a pseudo code of a greedy algorithm where in each step we try to get a new value for V(s) and not only move the snake but minimizing the energy too simultaneously. As long as we minimize the energy we iterate and when we find the best result we stop. We set the values of α, β and ƴ depending upon the type of image we are working on.

**Implementation**

Being most comfortable with MATLAB, I have tried to implement in the same. Below is the step by step approach I took:

1. First Create a user interface to make the application interactive
   1. Upload an image
   2. Apply different types of filters to improve the image quality to give better result
   3. Default the values of alpha, beta, gamma, W(line), W(edge), W(Term), number of iterations etc. which can even be modified by the user at any time.



1. Upload an image using the upload button and create a snake around it by input points.

global xy;

axis tight;

hold on;

[x, y] = getpts(axes1);

x=x';y=y';

temp=[x(1);y(1)];

xy=[x;y];

xy=[xy,temp];

n=length(xy);

t = 1:n;

ts = 1: 0.1: n;

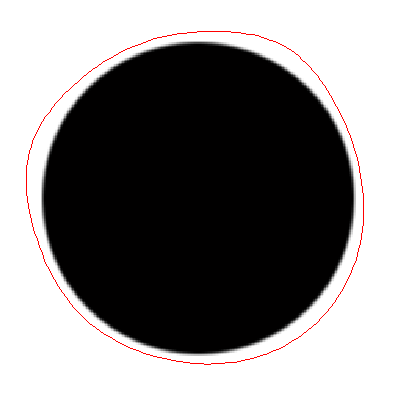
xys = spline(t,xy,ts);

xs = xys(1,:);

ys = xys(2,:);

xs=xs';

ys=ys';



1. Apply filters to enhance the image quality for better results.
2. Making the Snake shrink by using only snake energy to minimize the energy functional.

N = iterations;

imageobject = image;

[row col] = size(image);

Eimage=zeros(row,col);

[fx, fy] = gradient(Eimage);

[m n] = size(xs);

A = zeros(m,m);

b = [(2\*alpha + 6 \*beta) -(alpha + 4\*beta) beta];

temp = zeros(1,m);

temp(1,1:3) = temp(1,1:3) + b;

temp(1,m-1:m) = temp(1,m-1:m) + [beta -(alpha + 4\*beta)];

for i=1:m

A(i,:) = temp;

temp = circshift(temp',1)';

end

[L U] = lu(A + gamma .\* eye(m,m));

Ainv = inv(U) \* inv(L);

for i=1:N;

hold all;

newx = gamma\*xs - .1\*interp2(fx,xs,ys);

newy = gamma\*ys - .1\*interp2(fy,xs,ys);

xs = Ainv \* newx ;

ys = Ainv \* newy ;

im=imshow(image,[]);

set(im,'buttondownfcn',@clicky1) %Apply spring forces

set(im,'HitTest','on');

hold on;

plot([xs; xs(1)], [ys; ys(1)], 'r-');

hold off;

pause(0.05)

end;

1. Apply the spring forces by detecting a mouse click during minimisation.

global xs ys

global xy

min\_dist = 500;

point = get(gca,'Currentpoint');

x\_new=point(1);

y\_new=point(3);

len=length(xs);

x1=[];

y1=[];

for i = 1:10:len

x1=[x1,xs(i)];

y1=[y1,ys(i)];

end

xy1=[x1;y1];

len=length(xy1);

for i = 1:len

X=[xy1(1,i),xy1(2,i);x\_new,y\_new];

%X=[xs(i),ys(i);x\_new,y\_new];

distance=pdist(X,'euclidean');

if distance<min\_dist

min\_dist=distance;

location=i;

end

end

xy1(1,location)=x\_new;

xy1(2,location)=y\_new;

%new\_xy=[xs;ys];

n=length(xy);

t = 1:n;

ts = 1: 0.1: n;

xys = spline(t,xy1,ts);

xs = xys(1,:);

ys = xys(2,:);

xs=xs';

ys=ys';

return

1. Introduce the image forces.

imageobject = image;

[row col] = size(image);

image\_intensities = imageobject;

[grady,gradx] = gradient(imageobject);

eedge = -1 \* sqrt ((gradx .\* gradx + grady .\* grady));

mask1 = [-1 1];

mask2 = [-1;1];

mask3 = [1 -2 1];

mask4 = [1;-2;1];

mask5 = [1 -1;-1 1];

cx = conv2(imageobject,mask1,'same');

cy = conv2(imageobject,mask2,'same');

cxx = conv2(imageobject,mask3,'same');

cyy = conv2(imageobject,mask4,'same');

cxy = conv2(imageobject,mask5,'same');

for i = 1:row

for j= 1:col

eterm(i,j) = (cyy(i,j)\*cx(i,j)\*cx(i,j) -2 \*cxy(i,j)\*cx(i,j)\*cy(i,j) + cxx(i,j)\*cy(i,j)\*cy(i,j))/((1+cx(i,j)\*cx(i,j) + cy(i,j)\*cy(i,j))^1.5);

end

end

Eimage = (Wline\*image\_intensities + Wedge\*eedge -Wterm \* eterm);

**Conclusion**

In my opinion, this method is really good as it can be enhanced to achieve many different tasks like motion detection and stereo-matching etc. Also the result obtained by this implementation is quite satisfactory. Energy minimization technique makes it a very intelligent system which I guess can be improved a lot. Instead of working on the original image I have tried to introduce many type of filtering (image processing) which can be used to do a pre-processing on the image to make it suitable for the snakes to get the desired result. I also working on it to combine this with Canny edge detector to increase its efficiency. To use this for more complex images it need to be made more intelligent to detect edges which I think can be done by applying advanced machine learning to increase its efficiency even better.

**Reference**

[1] M. Kass, A, Witkin, and D, Terzopoulos, “*Snake: Active Contour Models*”, International Journal of Computer Vision, vol. 1, no.4, pp321-331, 1988

[2] <http://en.wikipedia.org/wiki/Active_contour_model>

[3] Active contours Models by Ritwik Kumar

Nitesh Kedia

ASU ID: 1206342939