# Snakes, Shapes and Gradient Vector Flow

Junaed Sattar
School of Computer Science
McGill University
Montreal QC Canada H3A 2A7

**Abstract**

*A novel method for finding active contours, or snakes as developed by Xu and Prince [1] is presented in this paper. The approach uses a regularization based technique and calculus of variations to find what the authors call a Gradient Vector Field or GVF in binary-values or grayscale images. The GVF is in turn applied to 'pull' the snake towards the required feature. The approach presented here differs from other snake algorithms in its ability to extend into object concavities and its robust initialization technique. Although their algorithm works better than existing active contour algorithms, it suffers from computational complexity and associated costs in execution, resulting in slow execution time.*

## 1 Introduction

Active contours or snakes are of great importance in computer vision. Applications like object tracking, shape modelling, segmentation and edge detection rely heavily on finding the contour of the object in question. Although, problems relating to initialization and convergence in object concavities limit their applications. Xu and Prince [1] present a new approach of snake modelling which they claim removes the abovementioned difficulties in active contour generation. This report discusses their algorithm in detail for binary-valued and grayscale images (section 2) and also discusses some results obtained by implementing and testing their algorithm (section 3).

## 2 Description of the algorithm

This section discusses the algorithm by Xu and Prince [1] for active contour modelling in details. The outline of the algorithm is as follows. The image on which the snake is to be computed is taken and is converted into a gray-level or binary-valued representation. If the objects of interest are present like line-like structures in the image, the input image can be directly used as the edge map. If the objects are present as homogeous regions whose boundaries separate the regions from the background of different intensity value, an edge map has to be computed. The edge map can be computed using any type of edge detector in the vision or image processing literature. The edge map is normalized to have all edge intensities fall between 0 and 1, if required This normalized edge map is input into the GVF solver. This will produce the GVF field of the edge map. The active contours can be formed by following the direction of the gradient field vectors over a certain number of iterations until a statistical equilibrium is reached.

### 2.1 Active Contours

A traditional snake or active contour is a curve $\mathbf{x(s)} = [x(s), y(s)], s \in [0, 1]$. that moves through the spatial domain of an image to minimize the energy functional

$$E = \int_0^1 \frac{1}{2}(\alpha|x'(s)|^2 + \beta|x''(s)|^2) + E_{ext}(x(s))ds \quad (1)$$

Here $\alpha$ and $\beta$ are the weighting parameters that control the snake's tension and rigidity, respectively. $x'(s)$ and $x''(s)$ are the first and second derivatives of $x(s)$ with respect to $s$. The external energy function $E_{ext}$ is derived in such a way from the image that it takes on smaller values at the feature of interests; e.g. at boundary positions. An example of $E_{ext}$ for a gray level image $I(x, y)$ leading to step edges could be

$$E_{ext}(x, y) = -|\nabla(G_\sigma(x, y) * I(x, y))| \quad (2)$$

where $G_\sigma(x, y)$ is a two-dimensional Gaussian function with standard deviation $\sigma$ and $\nabla$ is the gradient operator. Larger values of $\sigma$ will cause the boundaries to become blurry. Even though, large values of

$\sigma$ are sometime required to increase the capture range of the active contour, as can been seen from the results(section 3). A snake that minimizes $E$ must satisfy the Euler equation

$$\alpha x''(s) - \beta x'''(s) - \nabla E_{ext} = 0 \tag{3}$$

This is equivalent to a force balance equation

$$F_{int} + F_{ext}^{(p)} = 0 \tag{4}$$

where $F_{int} = \alpha x''(s) - \beta x'''(s)$ and $F_{ext}^{(p)} = -\nabla E_{ext}$. The internal force $F_{int}$ discourages stretching and bending whereas the external potential force $F_{ext}^{(p)}$ pulls the snake towards the desired image edges. To solve for (3), the snake is made dynamic by treating $x$ as a function of time $t$ as well as $s$. Then the partial derivative of $x$ with respect to $t$ is set as follows:

$$x_t(s,t) = \alpha x''(s,t) - \beta x'''(s,t) - \nabla E_{ext} \tag{5}$$

When statistical equilibrium for $x(s,t)$ is reached, the term $x_t(s,t)$ vanishes and a solution to equation (3) is obtained.

## 2.2 Problems with Snake Formation

Applications of snakes in computer vision and image processing has had their limitations. There are two main reasons for that. One, the poor convergence property of the active contours. Specifically, concavities in the object of interest are rarely covered; i.e. the snake does not extend to the concave regions of the object. An example of that is shown in figure 1. The snake was obtained by following the general algorithm outlined in the previous subsection. As can be seen, the steady state active contour fails to reach inside the concave region of the U-shaped object. The second problem with snakes is the limited capture range, which is related to the initialization of the snake around the object of interest. The magnitude of external forces die out far away from the boundaries. This phenomenon might fail convergence of the snake. Increasing the value of $\sigma$ of the Gaussian might improve the range, but at higher values, the edge will get blurry and eventually the concavitiy will be obliterated. Cohen and Cohen [2] proposed a distance potential force model to get around this problem. The general idea behind this model is to have large external forces far away from the boundaries of the object, thus increasing the capture range of the snake. Even then, snakes based on this model fails to converge to concavities, due to horizontal forces pulling the snake

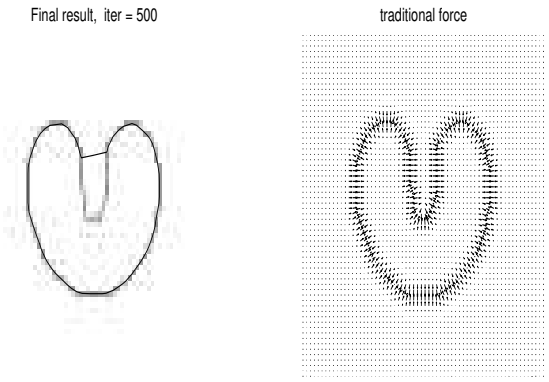


Figure 1: Traditional snake and force field after 500 iterations

apart but not downward into the concavities. The technique presented by Xu and Prince [1] addresses these issues and presents a new formulation for active contour modelling.

## 2.3 Gradient Vector Fields

The poor convergence of the snake can be shown to happen due to convergence of the solution to a local minimum. The authors present a solution to the problem by replacing the standard external force $F_{ext}$ in the force balance equation (4) with a static external force which does not change with time or depend on the position of the snake itself. This new static external external force field $F_{ext}^{(g)} = \mathbf{v}(x,y)$ is called the Gradient Vector Field or GVF. Replacing the external potential force $-\nabla E_{ext}$ in (5) with $\mathbf{v}$ yields the following equation:

$$x_t(s,t) = \alpha x''(s,t) - \beta x'''(s,t) + \mathbf{v} \tag{6}$$

The parametric curve solving the above dynamic equation is termed as a *GVF snake*. Standard numerical methods can be used to solve for this equation and yield the GVF snake.

## 2.4 GVF Snake Formation

The process starts by calculating the edge map of the given image, using any edge finding algorithm from the image processing literature. The edge map has three imporant features realting to snake formation. One, the gradient of this edge map has vectors pointing towards the edges which is a desirable property for snakes. Two, these vectors have large magnitude at the vicinity of the edges. Three, in homogenous regions (regions with little variation in image intensity)

$\nabla f$ is almost zero. The second and third features can be problematic when constructing an active contour. To keep the first feature and nullify the effect of the latter two, the gradient map is extended farther away from the edges and into homogenous regions using a computational diffusion process.

The gradient vector flow field is defined as the vector field $\mathbf{v}(x,y) = (u(x,y), v(x,y))$ that minimizes the following enerygy functional

$$\varepsilon = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2 dxdy \tag{7}$$

As can be seen, this is an example of variational formulation of regularization. The parameter $\mu$ is a regularizing parameter which adjusts the tradeoff between the first and second terms of the integrand and is set according to the level of noise present in the image. Also, where the value of the edge gradient is small, energy is dominated by the sum of the partial derivatives of the gradient field. When the gardient is large, the second term dominates. In this case, setting $\mathbf{v} = \nabla f$ minimizes the energy. Overall, this formulation transforms the gradient vector flow field by keeping it equal to the edge gradient at the boundaries; it also keeps $\mathbf{v}$ slowly varying at the homogenous regions of the image. Using the calculus of variations, it can be shown that the GVF field can be found by solving the pair of Euler equations stated below:

$$\mu\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) = 0 \tag{8}$$

$$\mu\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) = 0 \tag{9}$$

Here, $\nabla^2$ is the Laplacian operator. Equations (8) and (9) can be solved numerically by treating $u$ and $v$ as a function of time. The resulting equations are:

$$\begin{aligned} u_t(x,y,t) &= \mu\nabla^2 u_t(x,y,t) - (u(x,y,t) - f_x(x,y)) \\ &\quad \cdot(f_x(x,y)^2 + f_y(x,y)^2) \end{aligned} \tag{10}$$

$$\begin{aligned} v_t(x,y,t) &= \mu\nabla^2 v_t(x,y,t) - (v(x,y,t) - f_y(x,y)) \\ &\quad \cdot(f_x(x,y)^2 + f_y(x,y)^2) \end{aligned} \tag{11}$$

The steady-state solution of equation (10) and (11) yields the solution to the Euler equations (8) and (9). An iterative solution can be set up for solving the equations above.

## 3 Experiments and Results

The algorithm for snake generation and active contour modelling was implemented in Matlab and tested on images obtained from the author's websites and also on images obtained by myself. The images contained both binary valued and grayscale images. All edge detection was performed using the Canny edge detection algorithm.

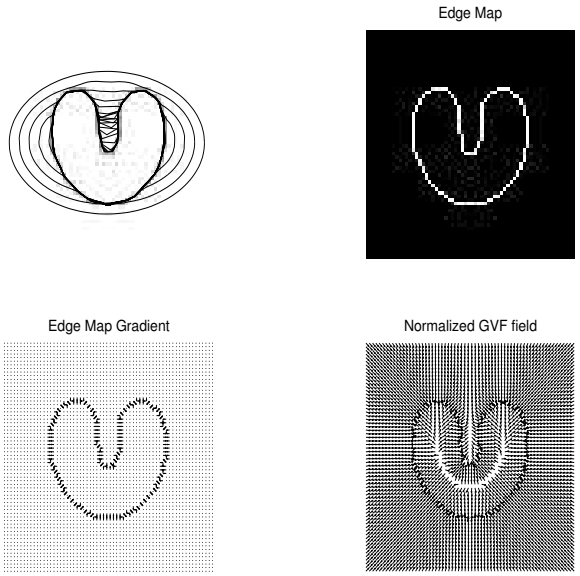


Figure 2: GVF Snake Formation

### 3.1 Convergence to Boundary Concavity

An implementation of the GVF snake algorithm was first run on a binary valued 64x64 pixel image containing a U-shaped object as shown in figure 2. The gradient maps, the GVF of the flow vector and the final GVF snake is also shown in the figure. As can be seen from the outputs, the algorithm correctly converges to the object boundary, even in the concavity in the given shape. The algoritm was next run on a binary valued image with dimensions of 64x64 pixels containing a non-convex polygon. The results can be seen in figure 3. Initialization of the snake, the gradient maps and the gradient vector fields for the given curve is also shown. The snake succeeded in converging to the boundary positions in this example too. The next run of the implementation was performed on a gray level image. The edge map was first computed using the Canny edge operator as mentioned before. No threshholding was performed on the image to binarize the edge map. The results are shown in figure 4. The results show that the snake was able to latch on to the correct object boundaries on this test image as well.

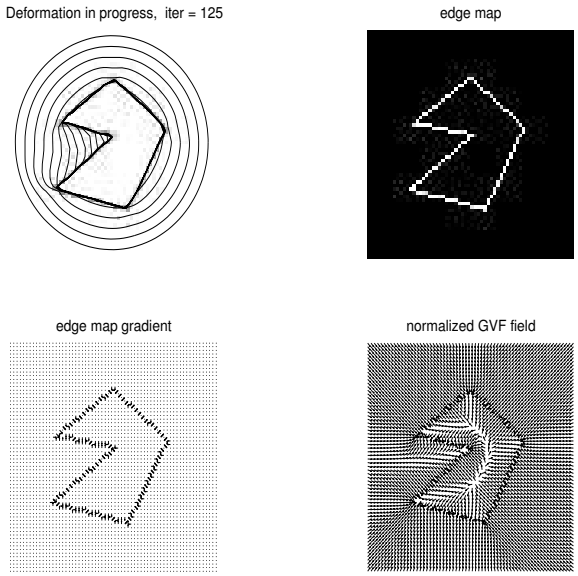


Figure 3: GVF Snake Formation

### 3.2 Insensitivity to Initialization

The GVF implementation was tested alongside an implementation of a traditional snake algorithm. The traditional snake algorithm, as can be seen from figure 5, has a limited capture range and is also very sensitive to the way the snake contour is initialized. Particularly, in cases where the snake is initialized across the object, part of the snake falls inside and part outside the object of interest. In such cases, the algorithm gets confused about the direction along which it should progress the snake, and converges to a highly undesirable configuration.

The GVF snake, on the other hand, follows the gradient vector flow field to reach a point of steady state configuration. The flow vectors both inside and outside of the object point to the object boundaries. The GVF snake thus eventually converges correctly along the object boundary. The tests show that GVF has insensitivity to initialization as well as ability to progress into bounary concavities.

### 3.3 Noise Sensitivity

The binary valued images on which the snake algorithm was tested came out to be quite robust upto a certain level of noise. After being corrupted by Gaussian white noise of zero mean and a small variance, the GVF snake algorithm was run on the images. The result can be shown in figure 6. The grayscale images took a larger value of sigma to reduce the effect of

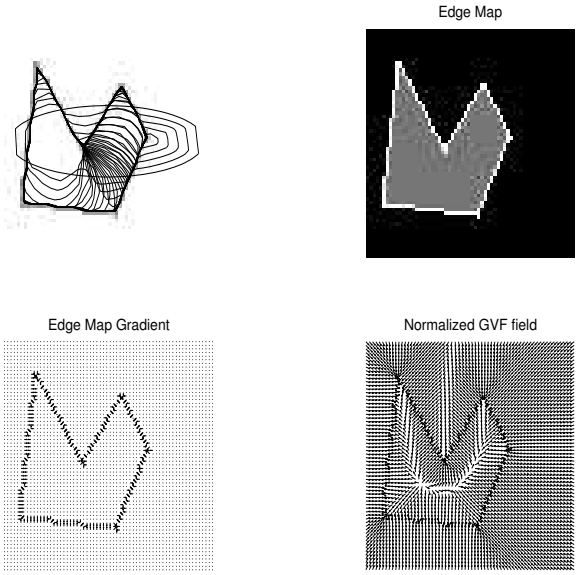


Figure 4: Graylevel GVF Snake with across-the-object initialization

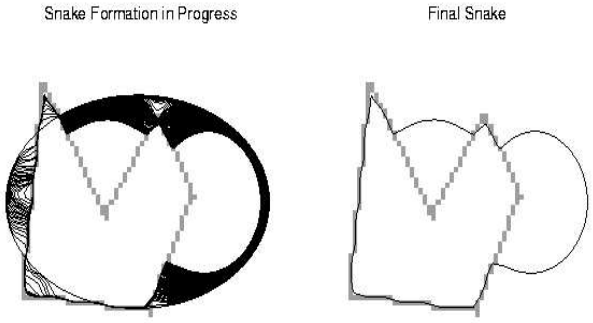


Figure 5: Traditional Snake with across-the-object initialization

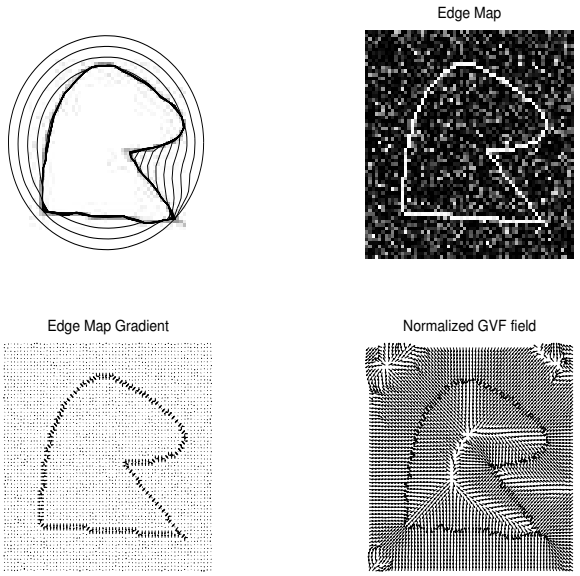


Figure 6: GVF Snake on Noisy Image

noise on the snake algorithm.

### 3.4 Performance

The single biggest drawback of the GVF snake algorithm was the speed of exeucution. The algorithm was implemented in Matlab for Linux (kernel 2.4.22) running on an AMP Athlon XP 1.4 GHz processor with 512 megabytes of RAM. For the binary-valued images (i.e. the image containing the U-shaped object) the algorithm took somewhere around the 50 second mark to complete the execution. The standard snake algorithm, on the other hand took around 11 seconds, although it failed to correctly converge to the boundary concavities. For grayscale images, the time taken was a lot more (almost around 90 seconds). The slowness could be due to the lack of optimization in the Matlab execution engine.Even then, this drawback makes the algorithm unsuitable for real time applications like tracking, inspite of its robustness and felxibility. The authors nevertheless claim that the speed of the execution could be improved by using native optimized C code and by using an algorithm optimization method as the multigrid method.

### 3.5 Higher-Dimensional GVF

The GVF snake algorithm can be extended to higher dimension and applied to 3-D objects. The paper presents an example of computing the contour of a 3-D object by using the GVF snake, although this particular case was not tested in the implementation.

## 4 Discussion

The gradient vector flow based active contour generation algorithm by Xu and Prince [1] was presented. The algorithm, along with a traditional snake generation algorithm was implemented in Matlab and tested on various images, both grayscale and binary valued. It was found that the algorithm succeeds in convergin the active contour to boundary concavitie in both types of images, even with the presence of noise. The drawback of the methods is its execution speed. Inspite of its robustness to initialization and increased capture range, the algorithm takes a long time to converge to object contours. Also, the value of the regulrization parameter $\mu$ and the snake parameters $\alpha$ and $\beta$ are set manually. The paper does not adress the issue of setting these parameters according to a particular image. The technique presented by Xu and Prince is attractive in terms of accuracy and robustness. It has found applications in medical imaging and offline tracking. With these issues in mind, the approach may not be suitable for real time tracking or contour modelling in unconstrained (e.g. excessively noisy) environments.

## References

[1] Chengyang Xu and Jerry L. Prince. “Snakes, Shapes and Gradient Vector Flow.” *IEEE Transactions on Image Prccessing*, Vol. 7, No. 3, March 1998.

[2] L. D. Cohen and I. Cohen. “Finite-element methods for active contour models and balloons for 2-D and 3-D images.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131-1147, November 1993.

[3] Chengyang Xu and Jerry L. Prince. “Gradient Vector Flow: A New External Force for Snakes.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 66-71, March 1997.

[4] B. K. P. Horn and B. G. Schunck. “Determining Optical Flow.” *Artificial Intelligence*, 17:185-203, 1981.