B-spline Snakes and a JAVA interface: An Intuitive Tool for General Contour Outlining

Patrick Brigger[1] and Robert Engel
ImageMinds
Imaging and Internet Technologies
Greenwich , CT, USA

Michael Unser
Biomedical Imaging Group
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland

Abstract

We present a novel formulation for B-spline snakes that can be used as a tool for fast and intuitive contour outlining. The theory is implemented in a platform independent JAVA interface, which allows real time computation of the snake curve. In this paper, our main focus is on two points. First, we propose a novel B-spline snake formulation, where the intrinsic scale of the spline model is adjusted a priori. It leads to a reduction of the number of parameters to be optimized and eliminates the need for internal energies. The approach solves the two main drawbacks of traditional snakes (slow convergence speed and difficult to adjust weighting factors for internal energy terms). Second, we comment on our experience using JAVA for the implementation of the snake and for the design of the graphical user interface. Our technique provides a very intuitive, user-friendly, and platform independent tool for contour outlining, generally applicable to a vast range of images. Several biomedical examples of applications are included to illustrate the versatility of the method.

1. Introduction

Many applications require the extraction of salient image features such as edges, lines, subjective differences in gray-level. Often, users are interested in computer assisted devices that help to detect and outline an image contour, but which still permit input of personal knowledge and experience. Such an approach is especially desirable in medical imaging, for the detection of specific organs or other medical features of interest. Often, two types of problems subsist: a) contour outlining is either too elementary and subjective (manual outlining using a pencil tool) or b) contour outlining is automatic, and specific to a particular problem, usually requiring important computational resources and not allowing a manual intervention.

The snake as an energy minimizing "spline" has found wide acceptance and has proven extremely useful in applications for medical analysis [1], feature tracking in video sequences [2], 3D object recognition[3], stereo matching [4, 5]. The original snake by Kass et. al. [5] is the solution of a functional minimization problem of Euler equations. The snake gives an elegant method to simulate an elastic material, which can dynamically conform to local image features. It is guided by external and internal forces. The former are given as a user supplied function, while the latter is determined by the shape of the curve in terms of first and second order derivatives. While it provides an elegant mathematical solution, it has two main drawbacks: 1) a large number of control points subject to optimization, 2) an explicit formulation of the smoothness constraint, which requires the knowledge of difficult to determine a priori weighting factors. The associated problems of slow convergence speed and difficult user-interaction have been addressed in a number of publications [6, 7, 8].

The recent introduction of B-snakes provides an alternative approach to snakes, which also circumvents many of the inherent problems [4, 9, 10]. The B-snake is mainly characterized by the following points: 1) few parameters, and 2) smoothness implicitly built into the model. In addition, the B-snake approach naturally permits the local control of the curve by controlling individual control points.

The motivation for our work is to extend the basic concept of B-spline snakes in order to improve their efficiency, speed, and to provide a means for intuitive interaction. Our main contributions are as follows: First,

[1] This work was mainly performed while employed with the Bioengineering and Physical Science Program, BEPS/ORS, National Institutes of Health (NIH), Bethesda, MD, USA.

we introduce a scale parameter for the B-splines, which allows us to control the smoothness of the snake implicitly. Second, the proposed formulation eliminates the need for curve-internal energies, making the snake more convenient to handle because no internal energy weight terms need to be determined. Third, we discuss object-oriented implementation issues, and the adequacy of JAVA as programming language.

In the next section, the B-spline snake will be formally derived. Section 3 discusses some implementation issues, and in Section 4, a number of examples are presented.

2. B-spline snakes: Parametric formulation

In this section, the goal is to introduce our parametric B-spline snake formulation, which is characterized by a scale parameter for the splines. A two-dimensional B-spline curve is defined by its control points as:

$$\mathbf{s}(t) = (s_x(t), s_y(t)) = \sum_{k \in Z} \mathbf{c}(k) \cdot \beta^n (t-k) \qquad (0 \le t \le t_{\max} = N-1) \tag{1}$$

where $s_x(t)$ and $s_y(t)$ are the x and y spline components, respectively, both parameterized by the curvilinear variable t. N denotes the number of control points, which corresponds to the number of primary B-spline coefficients, denoted by $\mathbf{c}(k) = (c_x(k), c_y(k))$. As the sum needs to be carried out over the domain of all integers, the remaining coefficients are found through appropriate boundary conditions. Based on (1), B-snake formulations have been proposed, which minimize an energy functional that consists of an external, user supplied term, and an internal, curve specific term [4, 9, 10]. While the formulation allows a continuous representation of the curve, it still is subject to some of the same limitations as in the traditional approach. The idea presented here is to eliminate the term corresponding to the internal energy and to introduce a variable knot spacing between the spline knot points. An increased knot spacing will essentially have the same smoothing effect on the solution. Thus, we consider a B-spline curve with a coarser knot spacing $h > 1$:

$$\mathbf{s}_h(t) = \sum_{k \in Z} \mathbf{c}_h(k) \cdot \beta^n \left(\frac{t}{h} - k\right), \qquad (0 \le t \le t_{\max} = hN-1) \tag{2}$$

The new smoothness parameter is thus h. Typically, we will take h to be an integer m, which will reduce the number of degrees of freedom (B-spline coefficients) in the same proportion. The end of the curve is given at $t = t_{\max}$, which is dictated by the desired resolution of the final curve. In the discrete case, we only render the curve for t integer, in which case we associate $M = t_{\max} + 1$. Clearly, if we specify N (i.e. the number of snake control points) and M, the curve resolution, then the knot spacing is $h = M / N$ and therefore the smoothness constraint for the curve is defined. The freedom of the curve has been reduced by the same amount, resulting in a smoothing and stiffening of the curve. Increasing the number N of control points will reduce the knot spacing, and consequently it will reduce the smoothing effect of the curve. The energy term may now be formulated in the following way:

$$\xi(\mathbf{c}(k)) = \sum_{i=0}^{M} g(s_x(i), s_y(i)) \tag{3}$$

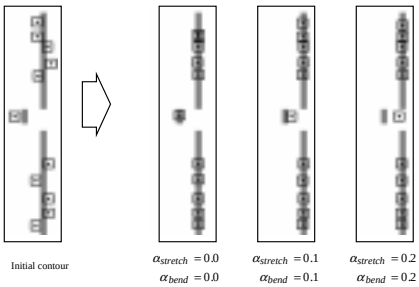
where $g(x, y) = L[f(x, y)]$. L is an image processing operator (for example, gradient magnitude) that enhances the contours of interest in the image $f(x,y)$. The operator may include a smoothing component to reduce the likelihood of the snake getting trapped in a local minimum. Optimization of (3) is carried out using a conjugate gradient algorithm [11]. With this algorithm, the direction of search for each independent variable is in an *A-orthogonal* direction with respect to the other variables. In the case of a quadratic potential function, the procedure leads to a scheme where exactly one step is done in every search direction. In other words, the principal idea is that once a parameter is optimized, the remaining parameters are moved in directions conjugate to the previously optimized parameters, such that those are not modified any more. An iterative scheme is used for non-quadratic potential function.

In Fig. 1, we demonstrate the similar optimization performances of the proposed B-spline snake formulation without internal energies, versus the traditional snake formulation [5] with internal energies. The comparison is based on a binary test image consisting of a vertical line, of which a small part has been displaced to the left (see Fig. 1, initial contour). In order to obtain a smooth force function, the binary image is smoothed by a two dimensional Gaussian, with $\sigma = 5$. Optimization is formulated as a minimization problem, and hence the optimal snake position is on the line. Eleven control points that have been set manually at unequal length intervals characterize the initial snake curve. Depending on the smoothness requirements of the final curve, two different results can be anticipated from the optimization. a) The resulting curve is vertically centered on the longer line, being unaffected by the small displaced part. Such

an outcome corresponds to an important smoothing constraint. b) The resulting curve has a "bump" and is attracted towards the small displacement on the left. This outcome reflects a less severe smoothness requirement.

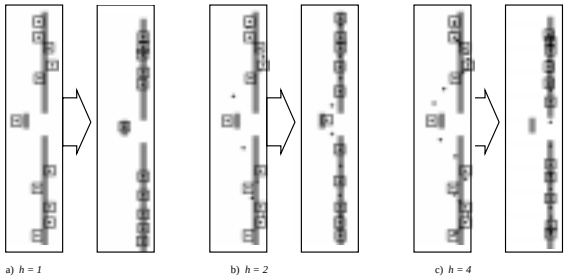
First, the traditional snake is computed with various weights for the internal energies. Each discrete curve point is independently optimized and attracted to the closest minimum by setting weights for the stretching and bending energy to zero (Fig. 1a). A weight of $\alpha_{stretch} = 0.1$ and $\beta_{stretch} = 0.1$ tends to pull the "bump" towards the right (Fig. 1b), however, does not produce a straight curve yet. A weight of $\alpha_{stretch} = 0.2$ and $\beta_{stretch} = 0.2$ produces an almost flat curve (Fig. 1c). This type of snake proves to be very flexible in that the user can choose among a large number of smoothness requirements by adjusting $\alpha_{stretch}$ and $\beta_{stretch}$. The feature may also represent a drawback for certain applications, because of the associated difficulties in choosing the correct weighting factors, by either empirical of automatic means.

The B-spline snake incorporates smoothness through different knot spacings. The knot spacing, $h = M / N$, where *M* is the number of interpolated points and *N* is the number of control points, can be changed by either varying *M* or *N*. For this example, we have decided to employ the same number of control points as for the traditional snake, and hence *h* is changed by changing the number of interpolated points *M*. A knot spacing of *h=1* signifies that no points are interpolated between control points. A B-spline of degree one corresponds exactly to the above experiment with zero weights, and the result is identical (Fig. 2a). Using a B-spline of higher degree, control points are no longer completely independent. For all remaining experiments, we have used a B-spline of degree three, because it leads to visually pleasant curve



a) b) c)

Figure 1: Traditional snake: Initial contour and optimization with different internal energies.



a) b) c)

Figure 2: B-spline snake: Initial contour and optimization with different knot spacings.

representations. The optimized result with *h=2* is shown in Fig. 2b. Note that there is one interpolated point between two control points. The point helps attracting the curve towards the longer line. An increased knot-spacing with *h=3* uses two interpolated values between two control points for computation of energy (Fig. 2c). In this configuration, these points manage to fully attract the curve towards the longer line, and the "bump" disappears. The experiment demonstrates the similar effect of a variable knot spacing and of internal energies on the smoothness of the final snake curve.

3. Object-oriented implementation using The JAVA programming language

Java was born three years ago. Every since, it has attracted programmers' attention, mainly because of its ability to run through web browsers, providing sophisticated and animated web pages. The power of Java, however, goes much further than that. It is a language that naturally supports an object-oriented implementation approach, being more convenient and less error prone than C++, for example. Furthermore, its ability to run on different computer platforms makes it attractive to stand-alone applications as well. For image processing, it has not been widely used so far, because of the slow performance capabilities of the interpreter. New improved just-in-time compilers and native compilers, however, have increased execution speeds. We have successfully implemented the B-spline snake in a Java framework, with real-time curve rendering speeds. Curve optimization requires less than 10 sec. on a 166MHz Pentium II Processor for a snake curve with 4 control points.

The B-spline snake algorithm design is object-oriented. In other words, the basic algorithm is designed to optimize an object of type *Function* with some given boundary conditions of type *Boundary*. Given such a framework, it is then possible to define an object

FunctionSpline derived from *Function*, as well as different boundary objects (*BoundaryPeriodic, BoundaryMirror, etc.)* derived from *Boundary*. Such a design is very general and allows the introduction of new functions and boundary conditions at any time, without having to change the main program. From our experience, the Java language lends itself extremely well for such an implementation, and usually allows a development in a shorter time than C++, for instance.

4. Application results

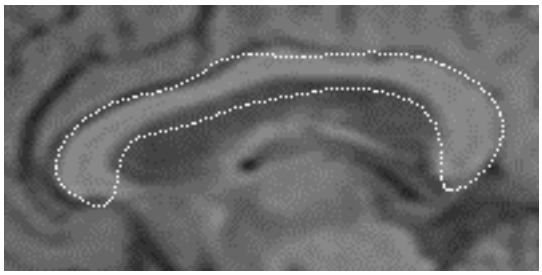
Typically, it is routine medical practice for a technician to outline the boundaries of the liver, spleen, kidneys, etc. in CT scans. Often, the standard imaging tools consist of an interface that allows manual border tracing using the computer mouse. Technicians face several problems related to that task. First, the hand drawn boundary is subject to small, uncontrollable hand movements that result in a noisy boundary. Second, often the technician does not have the possibility to pause in the outlining process. Also, a curve that is not considered satisfactory often has to be redrawn in its entirety.

We have asked an experienced technician to roughly outline the boundaries of the corpus callosum using the B-spline snake concept (Fig. 3a). The initial curve is then optimized and the final segmentation result is shown in Fig. 3b.

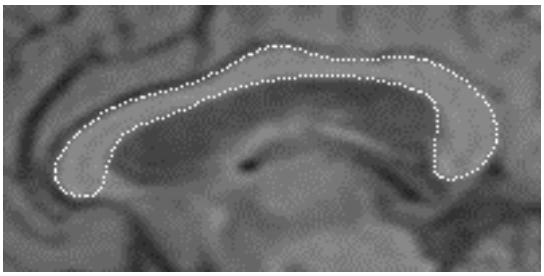
Another example is given in Fig. 3. The goal here is to detect the endothelial wall in an intravascular ultrasound image sequence that has been obtained by constant pull back of the ultrasound device. The initial contour is placed manually in the first frame (Fig. 4a). The snake is optimized and attracted to the endothelial wall (Fig. 4b). The result is then automatically propagated to the next frame, where it is again optimized. This process is iterated through the entire image sequence, and yields a 3D segmentation of the coronary wall. The segmentation result can be displayed as a 3D body using surface rendering techniques. The visualization program is also implemented in Java, and is shown in Fig. 5.

5. Conclusions

We have presented a novel implementation of the B-spline snake, which is characterized by a variable knot spacing. The approach eliminates the need for internal energies. Hence, this type of snake gives an intuitive way of user-interaction, because it does not require adjustment of weighting factors. Also, it features fast convergence speeds, because of the reduced number of control points. We have discussed the possibility of an implementation using Java. Our experiments show that the language is fast enough for real-time computation of the curve, fast optimization, as well as 3D surface rendering of the segmentation result. Various bio-medical examples have illustrated the versatility of the method.

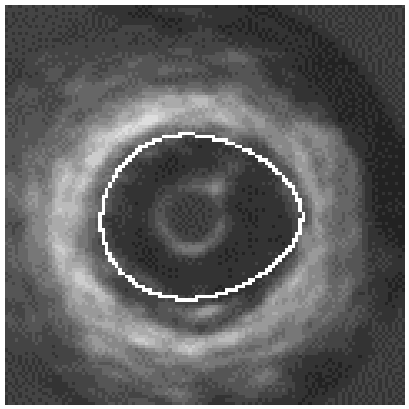


a)

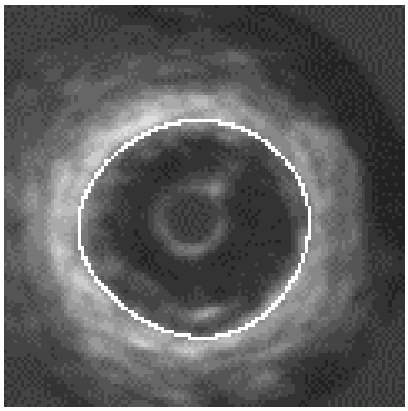


b)

Figure. 3: Outlining of the corpus callosum: a) initial curve and b) automatically optimized curve.



a)



b)

Figure 4: Example of an ultrasound sequence of the coronary artery. a) initial, manually placed contour, b) automatically optimized contour. For this example, only four control points were necessary.

Figure 5: 3D representation of a segment of the coronary artery.

References:

[1] A. Klein, T.K. Egglin, J.S. Pollak, F. Lee and A. Amini, "Identifying vascular features with orientation specific filters and B-spline snakes", *Computers in Cardiology*, pp. 113-116, 1994.

[2] M. Hoch and P. Litwinowicz, "A semi-automatic system for edge tracking with snakes", *The Visual Computer*, vol. 12, pp. 75-83, 1996.

[3] J. Wang and F. Cohen, "Part II: 3-D object recognition and shape estimation from image contours using B-splines, shape invariant matching, and neural networks", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 13-23, January, 1994.

[4] S. Menet, P. Saint-Marc and G. Medioni, "B-snakes: Implementation and application to stereo", in *Image Understanding Workshop*, pp. 720-726, Darpa, September, 1990.

[5] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active contour models", *Internat. J. of Computer Vision*, pp. 321-331, 1988.

[6] A. Amini, T. Weymouth and R. Jain, "Using dynamic programming for solving variational problems in vision", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855-867, September, 1990.

[7] K. Lam and H. Yan, "Fast greedy algorithm for active contours", *Electronics Letters*, vol. 30, no. 1, pp. 21-23, 1994.

[8] G. Xu, E. Segawa and S. Tsuji, "Robust active contours with insensitive parameters", *Pattern Recognition*, vol. 27, no. 7, pp. 879-884, 1994.

[9] M. Flickner, H. Sawhney, D. Pryor and J. Lotspiech, "Intelligent interactive image outlining using spline snakes", in *28th Asilomar Conference on Signals, Systems, and Computers*, vol. 1, pp. 731-735, 1994.

[10] M. Wang, J. Evans, L. Hassebrook and C. Knapp, "A multistage, optimal active contour model", *IEEE Trans. on Image Processing*, vol. 5, no. 11, pp. 1586-1591, November, 1996.

[11] E. Polak, *Computational methods in optimization*. New York: Academic Press, 1971.