Available online at www.sciencedirect.com

SCIENCE DIRECT®

Medical Image Analysis 10 (2006) 215–233

MEDICAL IMAGE ANALYSIS

www.elsevier.com/locate/media

United Snakes

Jianming Liang [a,*], Tim McInerney [b,c], Demetri Terzopoulos [d,c]

[a] *Computer Aided Diagnosis and Therapy, Siemens Medical Solutions USA, Inc., Malvern, PA 19355, USA*
[b] *Department of Computer Science, Ryerson University, Toronto, Ont., Canada M5B 2K3*
[c] *Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 3H5*
[d] *Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095, USA*

Received 21 September 2005; accepted 27 September 2005
Available online 28 November 2005

Abstract

Since their debut in 1987, snakes (active contour models) have become a standard image analysis technique with several variants now in common use. We present a framework called "United Snakes", which has two key features. First, it unifies the most popular snake variants, including finite difference, B-spline, and Hermite polynomial snakes in a consistent finite element formulation, thus expanding the range of object modeling capabilities within a uniform snake construction process. Second, it embodies the idea that the heretofore presumed competing technique known as "live wire" or "intelligent scissors" is in fact complementary to snakes and that the two techniques can advantageously be combined by introducing an effective hard constraint mechanism. The United Snakes framework amplifies the efficiency and reproducibility of the component techniques, and it offers more flexible interactive control while further minimizing user interactions. We apply United Snakes to several different medical image analysis tasks, including the segmentation of neuronal dendrites in EM images, dynamic chest image analysis, the quantification of growth plates in MR images and the isolation of the breast region in mammograms, demonstrating the generality, accuracy and robustness of the tool.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Snakes; Active contours; Live wire; Intelligent scissors; Finite elements; Interactive image analysis; Neuronal dendrite segmentation; Dynamic chest image analysis; Growth plate quantification; Breast region isolation

1. Introduction

Snakes (active contour models) quickly gained popularity following their debut in 1987 (Kass et al., 1988). They have proven to be especially useful in medical image analysis (McInerney and Terzopoulos, 1996; Singh et al., 1998) and for tracking moving objects in video (Terzopoulos and Szeliski, 1992; Blake and Isard, 1998), among other applications. Variants such as finite element snakes (Cohen and Cohen, 1993), B-snakes (Menet et al., 1990; Blake and Isard, 1998), and Fourier snakes (Staib and Duncan, 1992) have been proposed in an effort to improve aspects of the original finite difference implementation (e.g., to decrease initialization sensitivity, increase robustness against noise, improve selectivity for certain classes of objects, etc.). No formulation has yet emerged as the "gold standard". Rather, the primary variants seem well-suited to different applications with particular image modalities and processing scenarios.

Given the broad array of choices for the user, there is a need for a portable and reusable snakes implementation which unites the best features of the variants while maintaining the simplicity and elegance of the original formulation. To this end, our first contribution in this paper is to unify the most important snakes variants, including finite difference, B-spline, and Hermite polynomial snakes, in a comprehensive finite element formulation, where a particular type of snake can be derived by simply changing the finite element shape functions at the user level.

Subsequent to snakes, a related technique, known as "live wire" or "intelligent scissors" (Mortensen et al.,

* Corresponding author. Tel.: +1 610 448 1460.
E-mail address: jianming.liang@computer.org (J. Liang).

1995; Falcão et al., 1996; Barrett and Mortensen, 1997; Falcão and Udupa, 1997; Mortensen and Barrett, 1998; Falcão et al., 1998, 2000) emerged as an effective interactive boundary tracing tool. Based on dynamic programming (Falcão et al., 1998) or Dijkstra's graph search algorithm (Mortensen and Barrett, 1998), it was originally developed as an interactive 2D extension to earlier optimal boundary tracking methods. Live wire features several similarities with snakes, but it is generally considered in the literature as a competing technique. Our second contribution in this paper is the idea that live wire and snakes are in fact complementary techniques that can be advantageously combined via a simple yet effective method for imposing hard constraints on snakes. An advantage of this combination is the efficient handling of large images – a potential obstacle for live wire alone.

We call our software implementation *United Snakes* (Liang et al., 1999a,b), because it unites several snake variants with live wire to offer a general purpose tool for interactive image segmentation that provides more flexible control while reducing user interaction. United Snakes is implemented in the highly portable Java programming language. We have applied United Snakes to several different medical image analysis tasks including the segmentation of neuronal dendrites in EM images, dynamic chest image analysis, the quantification of growth plates in MR images and the isolation of the breast region in mammograms, demonstrating the generality, accuracy, robustness, and ease of use of the tool.

In the remainder of this paper, we first describe our finite element framework in Section 2 and show how several snake variants can be integrated within it. Section 3 describes the live wire technique. We justify the idea of combining snakes with live wire in Section 4 and develop a hard constraint mechanism in Section 5 that makes this combination possible. Section 6 presents results utilizing the United Snakes system in medical image segmentation scenarios. We conclude in Section 7 and propose future extensions of United Snakes.

2. Finite element unification of snakes

A snake is a time-varying parametric contour $\mathbf{v}(s,t) = (x(s,t), y(s,t))^{\mathrm{T}}$ in the image plane $(x,y) \in \Re^2$, where x and y are coordinate functions of parameter s and time t. The shape of the contour subject to an image $I(x,y)$ is dictated by an energy functional $\mathscr{E}(\mathbf{v}) = \mathscr{S}(\mathbf{v}) + \mathscr{P}(\mathbf{v})$. The first term is the internal deformation energy defined as

$$\mathscr{S}(\mathbf{v}) = \frac{1}{2}\int_0^L \alpha(s)\left|\frac{\partial \mathbf{v}}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 \mathbf{v}}{\partial s^2}\right|^2 \mathrm{d}s, \tag{1}$$

where $\alpha(s)$ controls the "tension" of the contour and $\beta(s)$ regulates its "rigidity". The second term is an external image energy

$$\mathscr{P}(\mathbf{v}) = \int_0^L P_I(\mathbf{v})\, \mathrm{d}s, \tag{2}$$

which couples the snake to the image via a scalar potential function $P_{\mathrm{I}}(x,y)$ typically computed from $I(x,y)$ through image processing. The Euler–Lagrange equations of motion for a dynamic snake are

$$\mu\frac{\partial^2 \mathbf{v}}{\partial t^2} + \gamma\frac{\partial \mathbf{v}}{\partial t} - \frac{\partial}{\partial s}\left(\alpha\frac{\partial \mathbf{v}}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2 \mathbf{v}}{\partial s^2}\right) = \mathbf{q}(\mathbf{v}). \tag{3}$$

The first two terms represent inertial forces due to the mass density $\mu(s)$ and damping forces due to the dissipation density $\gamma(s)$. The next two terms represent the internal stretching and bending deformation forces. On the right-hand side are the external forces $q(\mathbf{v}) = -\nabla P_{\mathrm{I}}(\mathbf{v}) + \mathbf{f}(s,t)$, where the image forces are the negative gradient of the image potential function. The user may guide the dynamic snake via time-varying interaction forces $\mathbf{f}(s,t)$ (usually applied through an input device such as a mouse), driving the snake out of one energy minimizing equilibrium and into another. Viewed as a dynamical system, the snake may also be used to track moving objects in a time-varying (video) image $I(x,y,t)$.

2.1. Finite element formulation

In a finite element formulation (Zienkiewicz and Taylor, 1989), the parametric domain is partitioned into finite subdomains, so that the snake contour is divided into "snake elements". Each element e is represented geometrically using shape functions $\mathbf{N}(s)$ and nodal variables $\mathbf{u}^{\mathrm{e}}(t)$. The nodal variables of all the elements are assembled into the snake nodal variable vector $\mathbf{u}(t)$. This leads to a discrete form of the equations of motion (3) as a system of second-order ordinary differential equations in $\mathbf{u}(t)$:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{g}, \tag{4}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the damping matrix, $\mathbf{K}$ is the stiffness matrix, and $\mathbf{g}$ is the external force vector, which are assembled from corresponding element sub-matrices that depend on the shape functions $\mathbf{N}$ (Appendix A details the finite element formulation).

By using different shape functions and thereby generating different stiffness matrices, the behavior of the resulting snake can be adapted to specific tasks. For example, snakes that use B-spline shape functions are typically characterized by a low number of degrees of freedom, typically use polynomial basis functions of degree 2 or higher, and are inherently very smooth. Therefore, these "B-snakes" (Menet et al., 1990; Blake and Isard, 1998) can be effective in segmentation or tracking tasks involving noisy images where the target object boundaries may exhibit significant gaps in the images. On the other hand, object boundaries with many fine details or rapid curvature variations may best be segmented by a snake that uses simpler shape functions and more degrees of freedom, such as a finite difference snake (Kass et al., 1988). Various contour representations are reviewed in Gavrila (1996). The unification of these different shape functions in a single framework expands the range of object modeling capabilities, and the range of segmentation and tracking scenarios that can be handled by a single tool.

The following sections address Hermitian shape functions, B-spline shape functions, and "shape functions" for finite difference snakes. Since the two coordinate functions $x(s)$ and $y(s)$ of the snake $\mathbf{v}(s)$ are independent, we shall discuss the shape functions in terms of only one component $x(s)$; the shape functions for $y(s)$ assume an identical form.

2.2. Hermitian shape functions

In the case of Hermitian snakes, $x(s)$ ($0 \leqslant s \leqslant l$, where l is the element parametric length) is approximated with a cubic polynomial function, parameterized by position x and slope θ at the endpoints $s = 0$ and $s = l$ of an element. We can show that $x(s) = \mathbf{N}_h \mathbf{u}^{e_i}$, where $\mathbf{u}^{e_i} = [x_i, \theta_i, x_{i+1}, \theta_{i+1}]^{\mathrm{T}}$ are the nodal variables of element e_i and $\mathbf{N}_h = \mathbf{sH}$ are the Hermitian shape functions, with $\mathbf{s} = [1, s, s^2, s^3]$ and the *Hermitian shape matrix* is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3/l^2 & -2/l & 3/l^2 & -1/l \\ 2/l^3 & 1/l^2 & -2/l^3 & 1/l^2 \end{bmatrix}. \tag{5}$$

It is reasonable to assume that the mass density $\mu(s)$, the dissipation density $\gamma(s)$, the tension function $\alpha(s)$ and rigidity function $\beta(s)$ are constant within the element. Hence, for element e_i, the mass matrix is

$$\mathbf{M}^{e_i} = \mu_i 420 l \begin{bmatrix} 156 & 22l & 54 & -13l \\ 22l & 4l^2 & 13l & -3l^2 \\ 54 & 13l & 156 & -22l \\ -13l & -3l^2 & -22l & 4l^2 \end{bmatrix} \tag{6}$$

the damping matrix is

$$\mathbf{C}^{e_i} = \gamma_i 420 l \begin{bmatrix} 156 & 22l & 54 & -13l \\ 22l & 4l^2 & 13l & -3l^2 \\ 54 & 13l & 156 & -22l \\ -13l & -3l^2 & -22l & 4l^2 \end{bmatrix} \tag{7}$$

and the stiffness matrices associated with the tension and rigidity components are, respectively,

$$\mathbf{K}_\alpha^{e_i} = \frac{\alpha_i}{30l} \begin{bmatrix} 36 & 3l & -36 & 3l \\ 3l & 4l^2 & -3l & -l^2 \\ -36 & -3l & 36 & -3l \\ 3l & -l^2 & -3l & 4l^2 \end{bmatrix}, \tag{8}$$

$$\mathbf{K}_\beta^{e_i} = \frac{\beta_i}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix}. \tag{9}$$

An analytic form of the external forces $\mathbf{q}(\mathbf{v})$ in (3) is generally not available. Therefore, Gauss–Legendre quadrature (Kwon and Bang, 1997) may be employed to approximate the value of the integral for the element external force vector $\mathbf{F}^{\mathrm{e}}$. For element e_i we have

$$\mathbf{F}_x^{e_i} = \int_0^l \mathbf{N}_h^{\mathrm{T}} \mathbf{q}_x(\mathbf{v}(s))\,\mathrm{d}s = \sum_j \rho_j \mathbf{N}_h(\xi_j)^{\mathrm{T}} \mathbf{q}_x(\mathbf{v}(\xi_j)), \tag{10}$$

where the subscript x indicates the association with coordinate function $x(s)$, and where ξ_j and ρ_j are the jth Gaussian integration point and its corresponding weighting coefficient, respectively. $\mathbf{F}_y^{e_i}$ is derived in a similar fashion.

To make the global matrix assembly process identical for all shape functions, we introduce *assembling matrices*. Suppose that we have a snake with n elements and N nodes ($N = n$ if the snake is closed and $N = n + 1$ if it is open). For the ith element e_i of the snake ($0 \leqslant i \leqslant n - 1$), the assembling matrices are $\mathbf{G}_{\mathbf{M}}^{e_i} = \mathbf{G}_{\mathbf{C}}^{e_i} = \mathbf{G}_\alpha^{e_i} = \mathbf{G}_\beta^{e_i} = \mathbf{G}_{\mathbf{F}}^{e_i} = \mathbf{G}^{e_i}$, where

$$(\mathbf{G}^{e_i})_{jk} = \begin{cases} 1 & \text{if } (j + di) \bmod (dN) = k, \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

are $(2d) \times (dN)$ matrices, with d the number of degrees of freedom of each node in an element (here $d = 2$). Hence, K_α, $\mathbf{K}_\beta$ and $\mathbf{F}$ may be assembled as follows:

$$\mathbf{M} = \sum_{i=0}^{n-1} (\mathbf{G}_{\mathbf{M}}^{e_i})^{\mathrm{T}} \mathbf{M}^{e_i} (\mathbf{G}_{\mathbf{M}}^{e_i}), \tag{12}$$

$$\mathbf{C} = \sum_{i=0}^{n-1} (\mathbf{G}_{\mathbf{C}}^{e_i})^{\mathrm{T}} \mathbf{C}^{e_i} (\mathbf{G}_{\mathbf{C}}^{e_i}), \tag{13}$$

$$\mathbf{K}_\alpha = \sum_{i=0}^{n-1} (\mathbf{G}_\alpha^{e_i})^{\mathrm{T}} \mathbf{K}_\alpha^{e_i} (\mathbf{G}_\alpha^{e_i}), \tag{14}$$

$$\mathbf{K}_\beta = \sum_{i=0}^{n-1} (\mathbf{G}_\beta^{e_i})^{\mathrm{T}} \mathbf{K}_\beta^{e_i} (\mathbf{G}_\beta^{e_i}), \tag{15}$$

$$\mathbf{F} = \sum_{i=0}^{n-1} (\mathbf{G}_{\mathbf{F}}^{e_i})^{\mathrm{T}} \mathbf{F}^{e_i}. \tag{16}$$

In our implementation, we set the element parametric length l to 1. Only the shape matrix and the assembling matrices are determined by specific shape functions. Therefore, in the following section we shall focus only on the derivation of the shape matrix and the assembling matrices for B-spline shape functions, and briefly mention other kinds of shape functions which are suitable for snakes.

2.3. B-spline shape functions

For B-spline shape functions, the $x(s)$ coordinate function of $v(s)$ is constructed as a weighted sum of N_B basis functions $B_n(s)$, for $n = 0, \ldots, N_B - 1$, as follows: $x(s) = B(s)^{\mathrm{T}} \mathbf{Q}^x$, where $\mathbf{B}(s) = [\mathbf{B}_0(s), \ldots, \mathbf{B}_{N_B-1}(s)]^{\mathrm{T}}$, $\mathbf{Q}^x = [x_0, \ldots, x_{N_B-1}]^{\mathrm{T}}$ and x_i are the weights applied to the respective basis functions $B_n(s)$.

A B-spline span serves as an element in our finite element formulation (hence "span" and "element" are interchangeable terms). Consequently, we shall determine the nodal variables, the shape matrix, and the assembling matrix associated with a span. When all spans are of unit length, the knot multiplicities at the breakpoints are $m_0, \ldots, m_L$ (L is the number of spans and the total number of knots $N_B = \sum_{i=0}^{L} m_i$), the knot values k_i are determined by $k_i = l$, such that $0 \leqslant (i - \sum_{j=0}^{l} m_j) < m_{l+1}$. Furthermore, the nth polynomial $B_{n,d}^{\sigma}$ in span σ can be computed as follows:

$$B_{n,1}^{\sigma}(s) = \begin{cases} 1 & \text{if } k_n \leqslant \sigma < k_{n+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

$$B_{n,d}^{\sigma}(s) = \frac{(s+\sigma-k_n)B_{n,d-1}^{\sigma}(s)}{k_{n+d-1}-k_n} + \frac{(k_{n+d}-s-\sigma)B_{n+1,d-1}^{\sigma}(s)}{k_{n+d}-k_{n+1}}. \tag{18}$$

For span σ, the index b_σ for the first basis function whose support includes the span can be determined as $b_\sigma = [(\sum_{i=0}^{\sigma} m_i) - d] \bmod N_B$. Therefore,

$I = [b_\sigma, (b_\sigma + 1) \bmod N_B, \ldots, (b_\sigma + d - 1) \bmod N_B]$

are the indices of the nodal variables and also those of the d polynomials $B_{n,d}^{\sigma}$.[1] Now, the shape matrix for span σ can be constructed by collecting the coefficients of each of the d polynomials $B_{n,d}^{\sigma}$ as its columns. For example, the shape matrix of a regular cubic B-spline is

$$\mathbf{H} = \begin{bmatrix} 1/6 & 2/3 & 1/6 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1/2 & -1 & 1/2 & 0 \\ -1/6 & 1/2 & -1/2 & 1/6 \end{bmatrix} \tag{19}$$

and the element matrices for element e_i are

$$\mathbf{M}^{e_i} = \frac{\mu_i}{5040}\begin{bmatrix} 20 & 129 & 60 & 1 \\ 129 & 1188 & 933 & 60 \\ 60 & 933 & 1188 & 129 \\ 1 & 60 & 129 & 20 \end{bmatrix}, \tag{20}$$

$$\mathbf{C}^{e_i} = \frac{\gamma_i}{5040}\begin{bmatrix} 20 & 129 & 60 & 1 \\ 129 & 1188 & 933 & 60 \\ 60 & 933 & 1188 & 129 \\ 1 & 60 & 129 & 20 \end{bmatrix}, \tag{21}$$

$$\mathbf{K}_{\alpha}^{e_i} = \frac{\alpha_i}{120}\begin{bmatrix} 6 & 7 & -12 & -1 \\ 7 & 34 & -29 & -12 \\ -12 & -29 & 34 & 7 \\ -1 & -12 & 7 & 6 \end{bmatrix}, \tag{22}$$

$$\mathbf{K}_{\beta}^{e_i} = \frac{\beta_i}{6}\begin{bmatrix} 2 & -3 & 0 & 1 \\ -3 & 6 & -3 & 0 \\ 0 & -3 & 6 & -3 \\ 1 & 0 & -3 & 2 \end{bmatrix}. \tag{23}$$

[1] In an open B-spline snake, d knots are introduced at the two ends. As a result, the index for the first basis function in the first span is zero (i.e., $b_0 = 0$) and the index of the last basis function in the last span is $N_B - 1$. For a closed B-spline snake, the index needs to be wrapped properly (Blake and Isard, 1998).

The assembling matrix $\mathbf{G}^{e_i}$ can be defined as

$$(\mathbf{G}^{e_i})_{jk} = \begin{cases} 1 & \text{if } (j + b_\sigma) \bmod N_B = k, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

In a similar fashion as above, we may construct other kinds of shape functions; for instance, NURBS shape functions (Terzopoulos and Qin, 1994), Catmull-Rom shape functions, Bézier shape functions, and Fourier shape functions (Staib and Duncan, 1992). The latter are global shape functions over the whole snake, thus the associated assembling matrix becomes an identity matrix.

2.4. Finite difference snakes in element form

Despite the differences between finite element snakes and finite difference snakes, the finite difference snakes can also be constructed in the finite element fashion, using the Dirac delta function $\delta(s)$ as the shape function. The construction primitives are as follows. For a snake with n nodes, $\mathbf{M}^{e_i}$ is a 1×1 matrix and its corresponding assembling matrix $\mathbf{G}_{\mathbf{M}}^{e_i}$ is a $1 \times n$ matrix:

$$\mathbf{M}^{e_i} = \mu_i[1]^{\mathrm{T}}[1] = \mu_i[1], \tag{25}$$

$$(\mathbf{G}_{\mathbf{M}}^{e_i})_{0,k} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

where $0 \leqslant i \leqslant n-1$ for both open and closed snakes. $\mathbf{C}^{e_i}$ is also a 1×1 matrix with a $1 \times n$ assembling matrix $\mathbf{G}_{\mathbf{C}}^{e_i}$:

$$\mathbf{C}^{e_i} = \gamma_i[1]^{\mathrm{T}}[1] = \gamma_i[1], \tag{27}$$

$$(\mathbf{G}_{\mathbf{C}}^{e_i})_{0,k} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{otherwise,} \end{cases} \tag{28}$$

where $0 \leqslant i \leqslant n-1$ for both open and closed snakes. $\mathbf{K}_{\alpha}^{e_i}$ is a 2×2 matrix and its corresponding assembling matrix $\mathbf{G}_{\alpha}^{e_i}$ is a $2 \times n$ matrix:

$$\mathbf{K}_{\alpha}^{e_i} = \alpha_i[-1 \quad 1]^{\mathrm{T}}[-1 \quad 1] = \alpha_i\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \tag{29}$$

$$(\mathbf{G}_{\alpha}^{e_i})_{jk} = \begin{cases} 1 & \text{if } (j+i) \bmod n = k, \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

where $0 \leqslant i \leqslant n-2$ for an open snake and $0 \leqslant i \leqslant n-1$ for a closed snake. $\mathbf{K}_{\beta}^{e_i}$ is a 3×3 matrix and with it is associated a $3 \times n$ assembling matrix $\mathbf{G}_{\beta}^{e_i}$:

$$\mathbf{K}_{\beta}^{e_i} = \beta_i[1 \quad -2 \quad 1]^{\mathrm{T}}[1 \quad -2 \quad 1] = \beta_i\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}, \tag{31}$$

$$\left(\mathbf{G}_{\beta}^{e_i}\right)_{jk} = \begin{cases} 1 & \text{if } (j+i) \bmod n = k, \\ 0 & \text{otherwise,} \end{cases} \tag{32}$$

where $0 \leqslant i \leqslant n-3$ for an open snake and $0 \leqslant i \leqslant n-1$ for a closed snake. The $1 \times n$ assembling matrix $\mathbf{G}_{\mathbf{F}}^{e_i}$ is defined as

$$(\mathbf{G}_{\mathbf{F}}^{e_i})_{0,k} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{otherwise,} \end{cases} \tag{33}$$

where $0 \leqslant i \leqslant n-1$ for both open and closed snakes.

With the above formulation of finite difference snakes, we have a uniform finite element construction for a variety of snake representations, which leads to a relatively straightforward United Snakes implementation in an object-oriented programming language, such as Java.

3. Live wire

Live wire (or intelligent scissors) is a recently proposed interactive boundary tracing technique (Mortensen et al., 1995; Falcão et al., 1996, 1997; Falcão and Udupa, 1997; Mortensen and Barrett, 1998; Falcão et al., 1998; Falcão et al., 2000). Although it shares some similarities with snakes – it was originally developed as an interactive 2-D extension to previous stage-wise optimal boundary tracking methods – it is generally considered in the literature as a competing technique to snakes. Like snakes, the idea behind the live wire technique is to allow image segmentation to occur with minimal user interaction, while at the same time allowing the user to exercise control over the segmentation process. However, live wire realizes the idea differently from snakes.

Live wire is very easy to use. The user begins by placing an initial *seed* point near the boundary of the object of interest. As the cursor, or *free* point is moved around the image, the current calculated boundary, called the *live wire* or *trace*, from the seed point to the free point is dynamically displayed. If the displayed trace is acceptable and the user clicks the mouse, the free point is collected as an additional seed point, and this trace will be frozen and will become part of the extracted object boundary. The resulting live wire boundaries are piecewise optimal (i.e., optimal between seed points), while the snake gives an optimal solution over the entire contour.

The genesis of live wire has its origin in the early collaboration between Udupa (University of Pennsylvania) and Barrett (Brigham Young University) (Mortensen and Barrett, 1998; Falcão et al., 1998). Their two research groups have since independently developed different live wire systems. They share two essential components: a local cost function that assigns lower cost to image features of interest, such as edges, and an expansion process that forms optimal boundaries for objects of interest based on the cost function and seed points provided interactively by the user. However, they employ different underlying graph models with different local cost functions. In (Mortensen and Barrett, 1998), each pixel represents a graph node, and directed, weighted edges are created between each pixel and its eight adjacent neighbors. In (Falcão et al., 1998), the graph nodes are pixel corners and they are connected by oriented, weighted edge cracks, called boundary elements (*bels* for short). In both cases, when the image is large, a corresponding large underlying graph may have to be maintained and live wire performance will be compromised. To improve the efficiency of live wire, the two groups have developed extensions known as live lane (Falcão et al., 1998) and toboggan-based intelligent scissors (Mortensen and Barrett, 1998; Mortensen, 2000), respectively.

Live-wire-like user interaction techniques have been proposed in the snakes literature. In (Cohen and Kimmel, 1997), Cohen and Kimmel compute the global minimal path between two points using Sethian's fast marching algorithm (Sethian, 1997), which has sub-pixel accuracy[2] and may eliminate metrication errors of graph search algorithms. A minimal path between two points is also obtained in (Grzeszczuk and Levin, 1994) based on simulated annealing. In a technique called "static" snakes, proposed in (Neuenschwander et al., 1994), the user initially specifies two end snake points and then the snake takes image information into account progressively from the two end points to its center, resulting in a minimal path between the two points. A similar technique has also been proposed in (Hyche et al., 1992). Dubuisson-Jolly and Gupta have formulated tracking an active contour with shape constraints in an image sequence as a shortest path problem (Dubuisson-Jolly and Gupta, 2001).

3.1. Trace formation

Boundary finding in live wire can be formulated as a directed graph search for an optimal (minimum cost) path using Dijkstra's algorithm in the underlying graph model. First, the graph is initialized with the local costs as described in the next section. Once the user selects a seed point (node), it will be used as the starting point for a recursive expansion process. In the expansion process, the local cost at the seed point is summed into its neighboring nodes. The neighboring node with the minimum cumulative cost is then further expanded and the process produces a dynamic "wavefront". The wavefront expands in the order of minimum cumulative cost. Consequently, it propagates preferentially in directions of highest interest (i.e., along image edges).

For any dynamically selected goal node (i.e., the free point) within the wavefront, the optimal path back to the seed point which forms a live wire trace can be displayed in real time. When the cursor (the free point) moves, the old live wire trace is erased and a new one computed and displayed in real time. The expansion process aims to compute an optimal path from a selected seed point to *every* other point in the image and lets the user choose among paths interactively, based on the current cursor position.

Live wire may be implemented very efficiently in multi-threaded programming languages, such as Java, because the expansion process and the user interface can execute in separate, parallel threads. Since the free point is generally near the target object boundary, the expansion process will most likely have already advanced beyond that point and the live wire trace can be displayed immediately. That

[2] Toboggan-based live wire (Mortensen and Barrett, 1999) obtains sub-pixel localization by fitting an edge model to the tobogganed region boundaries.

is, the live wire trace can typically be displayed before the expansion process has finished sweeping over the entire image. Therefore, our implementation (Liang et al., 1999a,b) is equivalent to the interleaved computation proposed in (Mortensen et al., 1995; Mortensen and Barrett, 1998) or live wire-on-the-fly introduced in (Falcão et al., 2000) in terms of computation cost, and the multi-threaded Java implementation is more elegant in software design and in supporting user interactions.

3.2. Local cost functions

Many local cost functions can be defined. In (Mortensen et al., 1995), the local cost $l(\mathbf{p},\mathbf{q})$ on the directed link from $\mathbf{p}$ to a neighboring pixel $\mathbf{q}$ is defined as a weighted sum of six local component costs created from various edge features:

$$l(\mathbf{p},\mathbf{q}) = \omega_Z f_Z(\mathbf{q}) + \omega_G f_G(\mathbf{q}) + \omega_D f_D(\mathbf{p},\mathbf{q}) + \omega_P f_P(\mathbf{q}) + \omega_I f_I(\mathbf{q}) + \omega_O f_O(\mathbf{q}), \tag{34}$$

where $f_Z(\mathbf{q})$ is the Laplacian zero-crossing function at $\mathbf{q}$,$f_G(\mathbf{q})$ is the gradient magnitude at $\mathbf{q}$,$f_D(\mathbf{p},\mathbf{q})$ is the gradient direction from $\mathbf{p}$ to $\mathbf{q}$,$f_P(\mathbf{q})$ is the edge pixel value at $\mathbf{q}$,$f_I(\mathbf{q})$ and $f_O(\mathbf{q})$ are the "inside" and "outside" pixel values at $\mathbf{q}$, respectively, while ω_Z, ω_G, ω_D, ω_P, ω_I and ω_O are their corresponding weights.

The Laplacian zero-crossing function $f_Z(\mathbf{q})$ is a binary function defined as

$$f_Z(\mathbf{q}) = \begin{cases} 0 & \text{if } I_L(\mathbf{q}) = 0, \\ 1 & \text{otherwise,} \end{cases} \tag{35}$$

where $I_L(\mathbf{q})$ is the Laplacian of the image I at pixel $\mathbf{q}$. The gradient magnitude serves to establish a direct connection between edge strength and cost. The function f_G is defined as an inverse linear ramp function of the gradient magnitude G

$$f_G = \frac{\max(G') - G'}{\max(G')} = 1 - \frac{G'}{\max(G')}, \tag{36}$$

where $G' = G - \min(G)$. When calculating $l(\mathbf{p},\mathbf{q})$, the function $f_G(\mathbf{q})$ is further scaled by 1 if $\mathbf{q}$ is a diagonal neighbor to $\mathbf{p}$ and by $1/\sqrt{2}$ if $\mathbf{q}$ is a horizontal or vertical neighbor.

The gradient direction $f_D(\mathbf{p},\mathbf{q})$ adds a smoothness constraint to the boundary by associating a higher cost for sharp changes in boundary direction. With $\mathbf{D}'(\mathbf{p})$ defined as the unit vector normal to the gradient direction $\mathbf{D}(\mathbf{p})$ at pixel $\mathbf{p}$ (i.e., $\mathbf{D}(\mathbf{p}) = [I_x(\mathbf{p}), I_y(\mathbf{p})]$ and $\mathbf{D}'(p) = [I_y(\mathbf{p}), -I_x(\mathbf{p})]$), the formulation of the gradient direction cost is

$$f_D(\mathbf{p},\mathbf{q}) = \frac{2}{3\pi}\{\arccos[d_{\mathbf{p}}(\mathbf{p},\mathbf{q})] + \arccos[d_{\mathbf{q}}(\mathbf{p},\mathbf{q})]\}, \tag{37}$$

where $d_{\mathbf{p}}(\mathbf{p},\mathbf{q}) = \mathbf{D}'(\mathbf{p}) \cdot \mathbf{L}(\mathbf{p},\mathbf{q})$ and $d_{\mathbf{q}}(\mathbf{p},q) = \mathbf{L}(\mathbf{p},\mathbf{q}) \cdot \mathbf{D}'(\mathbf{q})$ are vector dot products and

$$\mathbf{L}(\mathbf{p},\mathbf{q}) = \frac{1}{\|\mathbf{p}-\mathbf{q}\|}\begin{cases} \mathbf{q}-\mathbf{p} & \text{if } \mathbf{D}'(\mathbf{p})\cdot(\mathbf{q}-\mathbf{p}) \geqslant 0, \\ \mathbf{p}-\mathbf{q} & \text{if } \mathbf{D}'(\mathbf{p})\cdot(\mathbf{q}-\mathbf{p}) < 0 \end{cases} \tag{38}$$

is the normalized bidirectional link or unit edge vector between pixels $\mathbf{p}$ and $\mathbf{q}$.

Along with the gradient magnitude f_G, pixel value features (f_P, f_I and f_O) are used in on-the-fly training to increase the live wire dynamic adaptation (Mortensen and Barrett, 1998). With the typical gray-scale image pixel value range [0, 255], they are defined as

$$f_P(\mathbf{q}) = \frac{1}{255} I(\mathbf{p}), \tag{39}$$

$$f_I(\mathbf{q}) = \frac{1}{255} I(\mathbf{p} + k \cdot \mathbf{D}(\mathbf{p})), \tag{40}$$

$$f_O(\mathbf{q}) = \frac{1}{255} I(\mathbf{p} - k \cdot \mathbf{D}(\mathbf{p})), \tag{41}$$

where $\mathbf{D}(\mathbf{p})$ is the unit vector of the gradient direction as defined above, and k is a constant distance value for determining the inside and outside features.

In (Falcão et al., 1998), the local cost assigned to each boundary element (bel) $\mathbf{b}$ is a linear combination of the costs with its eight possible features f_i:

$$l(\mathbf{b}) = \frac{\sum_{i=1}^{8} w_i c_{f_i}(f_i(\mathbf{b}))}{\sum_{i=1}^{8} w_i}, \tag{42}$$

where w_i is the associated weight with feature f_i, and where c_{f_i}, called the *feature transform* function of feature f_i, converts feature value $f_i(\mathbf{b})$ into a cost value. The eight features of a bel $\mathbf{b}$ include the intensity values on positive and negative sides of $\mathbf{b}$ (f_1 and f_2), four different gradient magnitude approximations (f_3,f_4,f_5,f_6), orientation-sensitive gradient magnitude (f_7) and boundary distance (f_8). Each feature value ($f_i, 1 \leqslant i \leqslant 8$) may be converted into a cost value with any of the following six feature transforms: linear (c_1), inverted linear (c_2), Gaussian (c_3), inverted Gaussian (c_4), modified hyperbolic (c_5), and inverted modified hyperbolic (c_6). Training methods have been developed for optimum selection of the bel features and automatic selection of the parameters with their feature transforms, based on the typical segments painted by the user along the desired object boundary.

4. Combining snakes and live wire

Excluding user interaction, an accurate initialization is generally needed in order for a snake to lock onto image features of interest in all but the simplest images. Therefore, researchers have been actively investigating techniques to mitigate the sensitivity of snakes to their initialization. Among these techniques are the use of an inflation force (Terzopoulos et al., 1988; Cohen and Cohen, 1993), a chamfer distance map (Cohen and Cohen, 1993) and gradient vector flow (Xu and Prince, 1998). These techniques can work well if the image feature map is relatively clean. However, most clinical images are noisy, contain many uninteresting edges, or texture is present. Hence, these more automatic techniques can fail. For this reason, we explore an alternative direction – instead of attempting to mitigate initialization sensitivity, we seek to increase the efficiency of interactive initialization. In particular, we enable the user to instantiate (i.e., construct, initialize, and activate) snakes quickly and with minimal effort by exploiting the strengths of the live wire technique.

In this section, we first justify the complementarity of snakes and live wire, and then we show that the combination of snakes and live wire also provides an efficient mechanism for handling large images.

4.1. Snakes and live wire are complementary

There are numerous ways to define the local cost functions in live wire, as long as sufficiently low cost values are assigned to the desired object boundaries. Therefore, various techniques developed for computing snake potentials (Kass et al., 1988; Blake and Isard, 1998) can be used for the generation of local cost maps in live wire. For instance, the chamfer distance map (Cohen and Cohen, 1993) and gradient vector flow (Xu and Prince, 1998) are readily usable. Similarly, the local cost map computed for live wire may be treated as an image potential in snakes. Therefore, in United Snakes, snakes and live wire may share the same image potential (local cost map).

In general, live wire provides user-friendly control during the segmentation process. The user may freely move the free point on the image plane, and the corresponding live wire trace is interactively displayed in real time. However, once the free point is collected as an additional seed point, the trace is frozen and it becomes a part of the extracted object boundary. At this point, the user has no further control over the trace between seed points other than *backtracking*. Therefore, when the shape of the object boundary is complex or when object boundaries are noisy and unclear, the user may need to backtrack to produce acceptable traces. Consequently, many seed points may be needed to guide the live wire to an accurate result. Furthermore, it is not uncommon for the user to make small errors when placing seed points using a mouse or other input device, forcing the user to repeat the placement. By contrast, a snake may be dynamically adjusted or refined as it deforms at any time and at any point on the snake via intuitive mouse-controlled forces. However, the best performance of the snake is often achieved when user-specified constraint points are utilized. The constraint points do not "lock in" the solution – they too may be changed dynamically, allowing further refinement of the extracted object boundary.

Live wire seeks a global minimal path between two points. Therefore, when a section of the desired object boundary has a weak edge relative to a nearby strong but uninteresting edge, the live wire snaps to the strong edge rather than the desired weaker boundary. In order to mitigate this problem, Falcão et al. (1998) have developed training techniques for optimum feature selection and automatic parameter selection, and Mortensen and Barrett have proposed *on-the-fly training* (Mortensen and Barrett, 1998). Basically, these techniques (dynamically) update the cost map to filter out the image features which do not have edge characteristics similar to the sample boundary specified by the user. In other words, these methods rely on the assumption that the edge property is relatively consistent along the object boundary. Training is most effective for those objects with relatively consistent boundary properties and may be counter-productive for objects with sudden and/or dramatic changes in their boundary properties (Mortensen and Barrett, 1998). For example, in the lung image of Fig. 1(a), the live wire snaps to the strong edges of the elliptical viewport rather than the desired lung boundary. In this case, training is ineffective since the edge property of the lung boundary varies considerably over its extent and is also disturbed by the ribs (not obvious to the eye). Consequently, it is difficult to specify a typical segment of the lung boundary. Nevertheless, in situations where training can be effective, snakes can also take advantage of it by utilizing the trained cost map. Moreover, in United Snakes, the user has more control, using spring forces to pull the snake out of one minimum into another without training, as shown in Fig. 1(b).

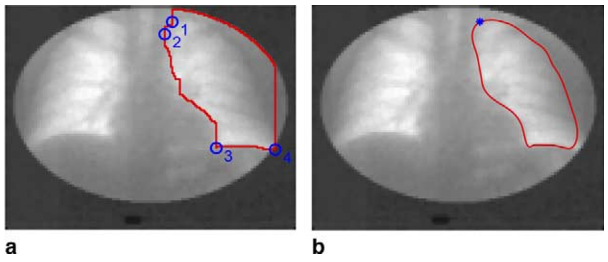

Fig. 1. (a) Delineation of the lung in X-ray fluoroscopy images using live wire (seed points are shown). (b) A Hermite snake instantiated from live wire traces with the first seed point imposed as a hard constraint. It is interactively pulled out of the strong edge with spring forces and then locks onto the lung boundary.

The underlying graph search makes it possible for live wire to bridge boundary gaps and pass through noisy areas. For instance, in MR wrist images from Falcão et al. (2000) (Fig. 2), live wire bridges the gap along the vessel boundary in Fig. 2(a) and passes through a noisy region in Fig. 2(b). Even if there are no image features at all between two points, live wire can still provide a minimal path – a straight line. However, live wire is inherently

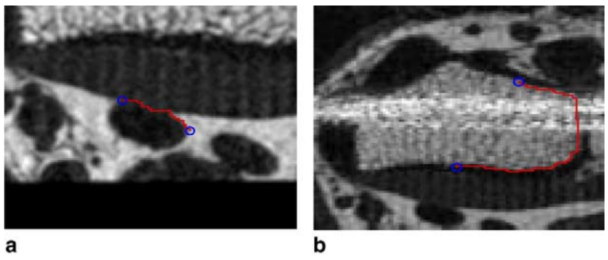

Fig. 2. Live wire bridges the gap along the vessel boundary (a) and passes through a noisy region (b) in an MR wrist image.

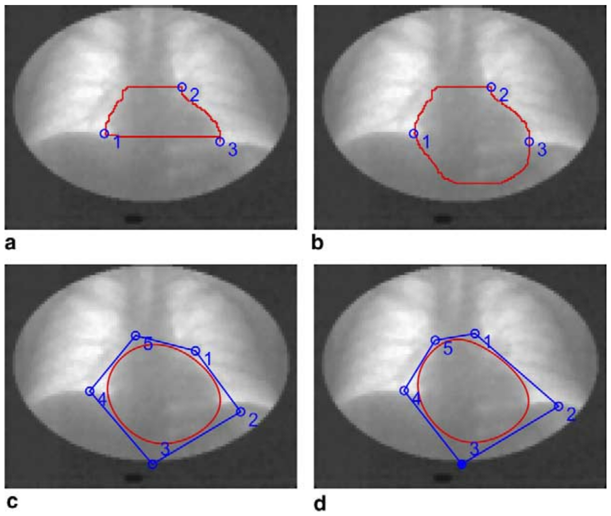

Fig. 3. (a) Delineation of the heart in X-ray fluoroscopy images using live wire (seed points are shown). (b) The unacceptable segment replaced by manual drawing. Alternately, the user may place multiple seed points and let live wire generate a piecewise linear path between adjacent seed points to approximate the missing cardiac boundary. (c) Initial B-spline snake and control polygon instantiated from live wire traces in (b). (d) Resulting segmentation after a few iterations with control point 3 as a hard constraint, which effectively bridges the gaps along the cardiac boundary.

image-based, rather than model-based. Fundamentally, it is not designed to bridge gaps in a manner that is consistent with the image features bordering the gaps and the smoothness of the traces cannot be guaranteed. For instance, in Fig. 3(a), part of the live wire trace from seed point 1 to seed point 2 is a straight line where the cardiac boundary is missing, and the live wire technique does not generate an acceptable cardiac boundary from seed point 3 to seed point 1. The user may place multiple seed points and let live wire generate a piecewise linear path between adjacent seed points to approximate the missing cardiac boundary. In this case, we have found that it is convenient to draw a rough curve manually between the points (Fig. 3(b)). Snakes, on the other hand, are model-based and were designed to adhere to image edges and interpolate between edge features in regions of sparse and noisy data (i.e., fill in the gaps). For example, a B-snake instantiated from the live wire traces is more effective in bridging the gaps along the cardiac boundary, as shown in Fig. 3(d).

In summary, it is desirable to enable the user to exercise more control over the live wire traces between seed points, impose smoothness on live wire traces, and bridge complicated gaps along object boundaries. This is what snakes are good at doing. Snakes adhere to edges with sub-pixel accuracy and they may also be adjusted interactively as parametric curves with intuitively familiar physical behaviors. Furthermore, snakes have the power to track moving objects, while live wire does not.

However, the efficient performance of interactive snakes is linked to fast, reasonably accurate initialization and user-specified constraints. Even with a few seed points, live wire can quickly give much better results than casual manual tracing. Hence, the resulting live wire boundary can serve well to instantiate a snake. The live wire seed points reflect the user's prior knowledge of the object boundary. They can therefore serve as either hard or soft point constraints for the snake, depending on the user's confidence in the accuracies of these seed points.

Because a live wire-traced initial object boundary is more accurate than a hand-drawn boundary, and with the further incorporation of the seed points as snake constraints, the snake will very quickly lock onto the desired object boundary. If necessary, the user may then correct mistakes inherited from the live wire-generated boundary by applying mouse-controlled spring forces to the snake. Because the user still has the opportunity to correct the deficiencies of the trace as the snake is evolving, the number of seed points needed by live wire to generate the object boundary can be further reduced. Consequently, a satisfactory initial object boundary can be generated very quickly using live wire. Other hard or soft constraints may be added during the snake deformation process as well. Because constrained values may be changed dynamically, the user may adjust the seed points to further refine the object boundary as the snake deforms.

To illustrate their performance, we apply United Snakes to an angiogram (Fig. 4) and a vertebra image (Fig. 5), to which Mortensen and Barrett applied their live wire algorithm in (Mortensen and Barrett, 1998). With only a few seed points, United Snakes generate the boundaries shown in Figs. 4 and 5(c), which are comparable to the ideal boundaries used as references in (Mortensen and Barrett, 1998).

As further evidence that the United Snakes framework improves upon the robustness and accuracy of its component techniques, Fig. 6 shows a synthetic image of a known curve degraded by strong Gaussian white noise (variance

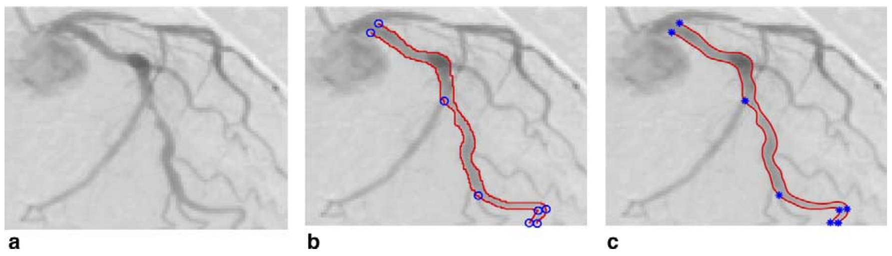

Fig. 4. Segmenting a vessel in an angiogram. (a) The image used in (Mortensen and Barrett, 1998). (b) Live wire segmentation. (c) United Snakes generate boundaries comparable to *ideal* boundaries in (Mortensen and Barrett, 1998).

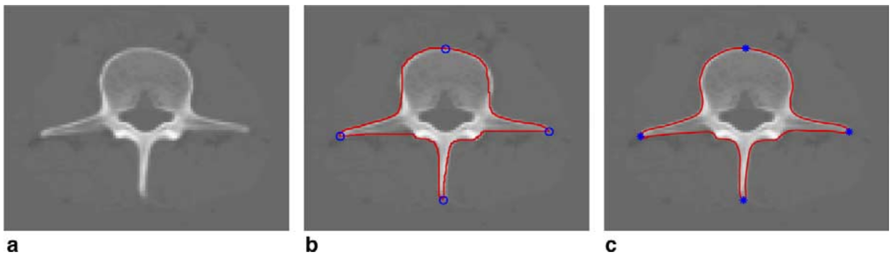
Fig. 5. Segmenting the outer boundary of a vertebra. (a) The image used in (Mortensen and Barrett, 1998). In United Snakes, we only expect a coarse object boundary from live wire. To illustrate this point, referring to Eq. (35), we have set $\omega_G = 0.50$, $\omega_Z = 0.5$, and turned all the other parameters off (i.e., $\omega_D = \omega_P = \omega_I = \omega_O = 0$), resulting in the live wire segmentation (b). From it, United Snakes generate a boundary (c) which is comparable to the *ideal* boundary in (Mortensen and Barrett, 1998).

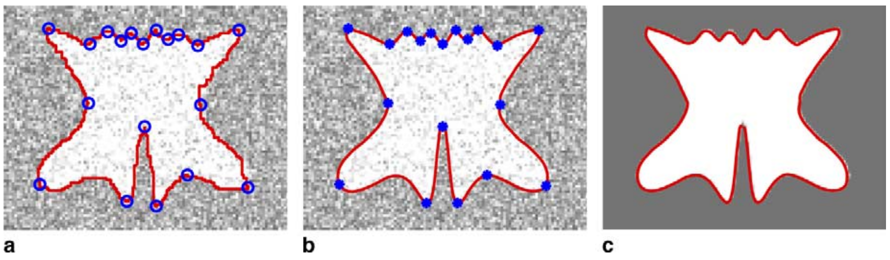
Fig. 6. Performance of United Snakes demonstrated using a noisy synthetic image. This image was designed to challenge the snakes and live wire with the high curvature points as well as the small wave details. (a) A live wire is sensitive to noise (the required seed points are shown). (b) United Snakes are robust against noise. (c) The segmented boundary accurately conforms to the ideal boundary.

0.25). Given its image-based nature, the live wire is sensitive to noise as shown in Fig. 6(a). A snake instantiated by the live wire gives a better result (Fig. 6(b)). Fig. 6(c) shows that the United Snakes result is very close to the boundary in the ideal image, despite the strong noise. This performance is a consequence of the imposed hard constraints, without which the snake would slip away from high curvature points.

4.2. Handling large images

Large images are now common in clinical settings because high resolution is often needed to make accurate diagnoses. For instance, in the mammogram analysis task (see Section 6.4), we need to handle images with a typical resolution of 3500×6500 pixels. However, due to the nature of its underlying graph-based algorithm, the basic live wire algorithm is unable to handle large images efficiently. To support user interaction, live wire aims to compute an optimal path from the last seed to *every* other point in the image. Even with our efficient multi-threaded Java implementation of Dijkstra's algorithm (e.g., with bucket sort (Mortensen and Barrett, 1998) or Dia's method (Falcão et al., 2000)), the performance of live wire will be significantly compromised when working with large images. The reason is that the lower bound of its computational complexity is O(m), where m is the number of image pixels involved in the computation of an optimal path from the seed to the free point; that is, all the pixels within the wavefront (i.e., the expansion process). In the worst case, m is the total number of pixels in the image.

For effective user interaction, the thread responsible for computing an optimal path from the seed point to every other point in the image should not stop until the user has selected the current free point as a seed. This ensures the user may move with more freedom in the image plane to select optimal paths and quickly generate an acceptable object boundary with a minimal number of seed points. That is, the user should be able to place two neighboring seed points as far apart as desired. However, when the desired object boundary is not clear/sharp (e.g., chest images, mammograms, etc.) or has many branches (e.g., a retinal angiogram), the wavefront will spread too widely and include many pixels for any path of reasonable length. As a result, the memory required to maintain the auxiliary information in Dijkstra's algorithm will increase dramatically for large images. Furthermore, as demonstrated also in (Falcão et al., 2000), when the image size changes from 128×128 to 1024×1024, the live wire performance will be reduced by a factor of 400, and the ultra fast live wire on the fly may still be 40 times slower.

The combination of live wire and snakes in United Snakes provides a new mechanism for handling large images. The computational complexity of snakes is O(n) in each iteration, where n is the number of snake nodal variables. In United Snakes, we typically require live wire to generate only a coarse boundary with a few seed points. Therefore, we can construct a *truncated* pyramid of images, and let the live wire work at the top of the pyramid with a small image size (for example, 128×128 or 256×256), thus efficiently supporting user interaction. The snake "descends" the image pyramid from coarse to fine levels of resolution, tolerating any live wire errors introduced at the top of the pyramid, and accurately locks onto the desired object boundary. The original large image is still displayed to the user and thus the seed points can generally be accurately specified or dynamically adjusted if necessary. The extra memory needed to maintain the pyramid is offset by the reduced memory necessary for the auxiliary information in Dijkstra's algorithm. In practice, we do not have to maintain a pyramid for the original image, but only for the image potential. Assuming the pyramid has n levels and the original image occupies M amount of memory, the extra memory required for the pyramid then is

$$\left(\frac{1}{4}+\frac{1}{4^2}+\frac{1}{4^3}+\cdots+\frac{1}{4^{(n-1)}}\right)M=\left(1-\frac{1}{4^{(n-1)}}\right)\frac{M}{3}, \tag{43}$$

while the reduced auxiliary memory (e.g., only for the cumulative cost map) in the live wire implementation is $(1-\frac{1}{4^{(n-1)}})M$.

As a demonstration, Fig. 7(a) shows a retinal angiogram with pixel resolution of 256×256 obtained by down-sampling the original large image with resolution 1024×1024. Suppose we would like to trace the vessel starting from point S to one of the target points 0–9 (see Fig. 7(b)). Once point S is selected as the first seed, the corresponding branch should ideally be instantly available once the user points to any of the 10 branch end-points. This real-time user interaction is achievable by live wire when the image size is under 300×300 on modest PCs (Fig. 7(c)). For larger images, however, real-time user interaction becomes increasingly difficult to achieve using live wire alone. Table 1 shows the time needed for the wavefront to reach the 10 targets as well as the time needed to sweep over the entire image at different resolutions on an 866 MHz Pentium PC with SUN JDK1.3. From the table, we can see that the time required at 256×256 resolution is approximately 1/4 of

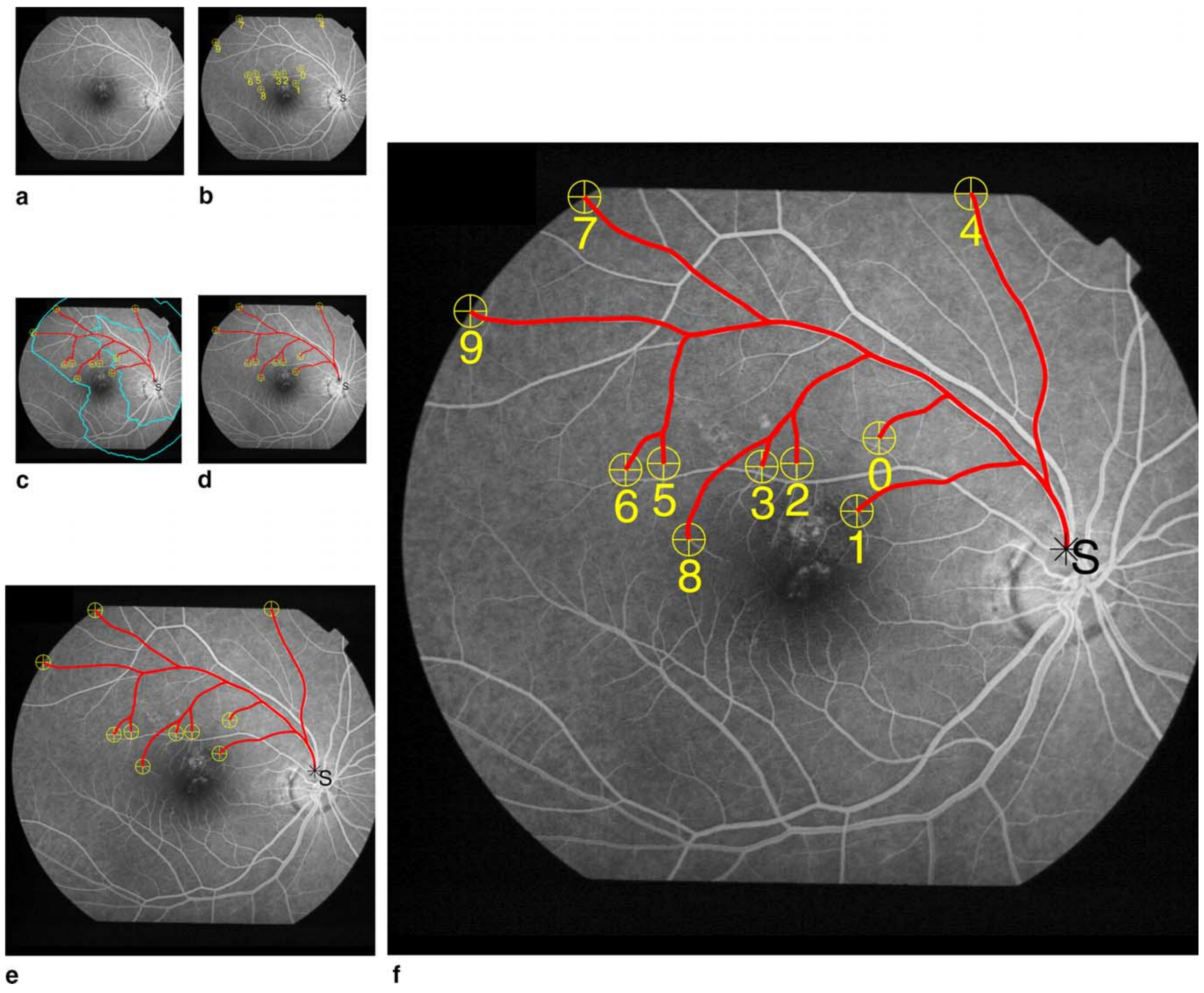

Fig. 7. (a) A retinal angiogram with pixel resolution 256×256 obtained by down-sampling the original 1024×1024 image. (b) The seed point S and 10 marked target points. (c) The superimposed live wire traces shown only for the first and last target points. (d) The superimposed snakes dynamically instantiated from the live wire traces in (c) descend the truncated pyramid reaching the intermediate level with resolution 512×512 (e) and the original large image (f). This mechanism supports real-time user interaction: once point "S" is selected as a seed, the corresponding vessel branch is instantly available when the user points to a new position (such as, the 10 targets) on the original large image.

Table 1
The time (in milliseconds) needed for the wavefront to reach the 10 targets shown in Fig. 7, as well as the time needed to sweep over the entire image at different resolutions on an 866 MHz Pentium PC with SUN JDK1.3

Targets	0	1	2	3	4	5	6	7	8	9	Entire image
256×256	65	109	118	131	131	170	205	211	225	240	405
512×512	374	526	545	604	805	875	979	1025	1081	1248	2178
1024×1024	1562	2293	2393	2703	3479	3683	4425	4543	4688	5481	9802

The time required at 256×256 resolution is approximately 1/4 of that at 512×512 resolution, which is roughly 1/4 of that at 1024×1024 resolution. The time needed for a snake sliding down is $O(n)$ in each iteration, where n is the number of snake nodal variables. We use five iterations at each level of the pyramid. The longest snake (from point S to point 9 in Fig. 7) in this experiment has 100 nodal variables. When the live wire trace is available at the top level (256×256), it takes 15 iterations or 77 ms for the snake to descend the pyramid. So, the total time needed is 317 (77 + 240) ms, which is much less than 5481 ms when working directly at the resolution of 1024×1024.

that at 512×512 resolution, which is roughly 1/4 of that at 1024×1024 resolution. This can be justified by the observation that reducing an image by a factor of 2 in linear dimension while maintaining its aspect ratio reduces its area in pixels to 1/4, and that the complexity of the wavefront computation is proportional to the latter. Thus, with a three-level pyramid, we can make the algorithm approximately 16 times faster and, with four levels, it becomes approximately 64 times faster.

In United Snakes, snakes that are dynamically instantiated from live wire traces at the top of the truncated image pyramid can easily descend the pyramid, reaching the original large image (Fig. 7(f)) via intermediate level(s) (Fig. 7(e)), resulting in real-time user interaction on the original large image. Thus, United Snakes with the image pyramid scheme yields real-time response – a critical factor in any interactive segmentation scheme – with sub-pixel accuracy in original large images.

5. Hard constraints

Our combination of snakes and live wire relies on an efficient constraint mechanism. A constraint on a snake may be either soft or hard. Hard constraints generally compel the snake to pass through certain positions or take certain shapes, whereas soft constraints merely encourage a snake to do so. Two kinds of soft constraints, springs and volcanos, were described in the original snakes paper (Kass et al., 1988) and they are incorporated into our finite element formulation. Hard constraints have been used to prevent snake nodes from clustering in dynamic programming snakes (Amini et al., 1990). Generic hard constraints are discussed in (Fua and Brechbühler, 1997,). In this section, we propose a convenient mechanism, called *pins*, as a simple yet effective way to impose hard constraints on snakes for the integration of snakes and live wire.

Suppose that we wish to guarantee that the snake node i sticks at position (x_i^c, y_i^c) in the Hermitian parameterization. Recall that in the Hermitian parameterization, the polynomial shape of each element is parameterized by the position and slope of $x(s)$ and $y(s)$ at the two nodes (position and slope variables occupy alternating positions in the nodal variable vector $\mathbf{u}$). Therefore, the snake stiffness matrix $\mathbf{K}$ may be updated with

$$\mathbf{K}_{2i,j} = \begin{cases} 1 & \text{if } 2i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{44}$$

where $0 \leqslant j \leqslant 2(N-1)$ and N is the number of snake nodes. The system force vector $\mathbf{F}$ is updated as

$$\mathbf{F}_{2i}^x = x_i^c, \quad \mathbf{F}_{2i}^y = y_i^c, \tag{45}$$

where x and y indicate coordinate function $x(s)$ and $y(s)$, respectively. It is then guaranteed that the snake node i is always at position (x_i^c, y_i^c).

A drawback of this simple technique, however, is that the updated system stiffness matrix is no longer symmetric. Consequently, we are unable to store the stiffness matrix economically using skyline storage, nor factorize it into LDL^T form (see Appendix A). Nevertheless, since the position of node i is given, a constant force may be derived from the stiffness matrix for each degree of freedom and subtracted from its corresponding position in the system force vector so that we can restore the symmetry of the stiffness matrix while keeping the system in balance. In our implementation, we store column $2i$ of $\mathbf{K}$ into a vector k^{2i}; i.e., $\mathbf{k}_j^{2i} = \mathbf{K}_{j,2i}$, for $0 \leqslant j \leqslant 2(N-1)$, before K is made symmetric with

$$\mathbf{K}_{j,2i} = \begin{cases} 1 & \text{if } 2i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{46}$$

To keep the system in balance, the system force vector $\mathbf{F}$ is updated with

$$\mathbf{F}_j^x = \mathbf{F}_j^x - x_i^c \mathbf{k}_j^{2i}, \quad \mathbf{F}_j^y = \mathbf{F}_j^y - y_i^c \mathbf{k}_j^{2i} \tag{47}$$

for $0 \leqslant j \neq 2i \leqslant 2(N-1)$. We can constrain the slope in the same way. If we constrain two node variables of an element in both position and slope, this element will be frozen. Its two neighboring elements will also be influenced by the constraint. The constraints on a B-snake are imposed on the nodes of its control polygon. Imposing hard constraints in this manner also lessens computational cost, in terms of both memory and time, since the number of entries in the skyline storage of the stiffness matrix is reduced. Consequently, the LDL^T factorization and forward/backward substitutions can be performed more efficiently (see Appendix A). It is also possible to apply more general constraints to any point on the snake as is described in (Terzopoulos and Qin, 1994).

In the formulation above, the updated stiffness matrix only indicates which degrees of freedom of the snake are constrained, it does not contain any constraint values. These are recorded in the system force vector. As a result, the constraint values may be updated dynamically during snake deformation. Hence, the user can move the constraint points around the image plane to refine the object boundary as the snake is deforming. This property is very useful when integrating snakes with live wire. While a snake is deforming, additional hard constraints may be imposed on the snake to restrict its deformation. Because these constraints are unknown before the snake is instantiated, they may be incorporated on-the-fly using reaction forces on the system force vector without changing the stiffness matrix. However, small time steps are required to ensure the stability of the snake. In our implementation, we create a new snake from the current snake plus the hard constraints, since the LDL^T factorization is fast.

Hard constraints play very important roles in capturing the intricate details and bridging gaps along object boundaries in image segmentation. For instance, to segment the bladder from an MR image of a female abdomen shown in Fig. 8, neither the live wire (Fig. 8(a)) nor its corresponding, instantiated dynamic snake (Fig. 8(b)) would be able to capture the intricate details indicated by the rectangle, which require additional seed points (Fig. 8(c)). With all the seed points imposed as hard constraints, the corresponding snake accurately captures the details without any further user intervention as shown in Fig. 8(d). Hard constraint points may be adjusted to refine the object boundary. For illustration purposes, hard constraint point 1 has been deliberately moved far away from the desired bladder boundary in Fig. 8(e). Releasing the hard constraints will lose the details as shown in Fig. 8(f).

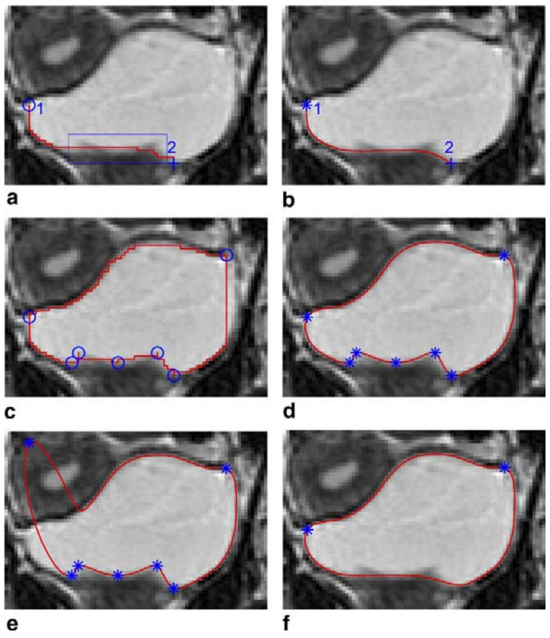

Fig. 8. Segmenting the bladder in an MR image of a female abdomen. Neither the live wire (a) nor its corresponding dynamic snake (b) would be able to capture the intricate details indicated by the rectangle without additional live wire seed points (c). (d) With all the seed points naturally imposed as hard constraints, the corresponding snake accurately captures the fine, intricate details without any further user intervention. (e) Hard constraint point 1 is deliberately moved far away from the desired bladder boundary to illustrate the adjustment capability. (f) Releasing the hard constraints will lose the details.

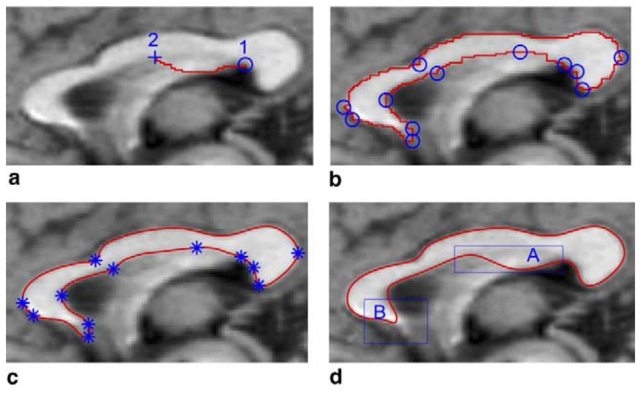

Fig. 9. Segmenting the corpus callosum in an MR head image of a human volunteer. (a) The live wire snaps to the nearby strong edge. (b) An additional seed bridges the missing boundary. (c) The corresponding snake with the seed points naturally imposed as hard constraints nicely captures the desired object. (d) When releasing the hard constraints, the strong image force in the region of region A will gradually drive the snake away from the desired boundary, while, in region B, the snake will become insufficiently peaked due to its internal energy.

The desired object boundary might be unclear or even missing in many clinical images. For example, in segmenting the corpus callosum in an MR head image of a human volunteer, the live wire snaps to the nearby strong edge 9(a), and additional seed points are required to bridge the missing boundary 9(b). These seed points can be naturally imposed as hard constraints on the corresponding snake, which nicely captures the desired object in Fig. 9(c). When releasing the hard constraints, the strong image force in region A (see Fig. 9(d)) will gradually drive the snake away from the desired location, while the snake will become insufficiently peaked in region B due to its internal energy.

5.1. Static vs. dynamic constraint integration

We have argued that a hard constraint mechanism is crucial in practical image segmentation. Live wire generally requires seed points at the critical, complicated locations where the desired boundary is twisted, unclear, weak or even missing. These seed points, interactively provided by the user to guide the live wire, capture the user's expert prior knowledge about the desired object boundary, and they can naturally be imposed as hard constraints on the snake that is then instantiated from the complete live wire trace. We refer to this form of livewire-snake integration as *static* integration – once the live wire result is used to instantiate a snake, the segmentation process continues using only the constrained, user-controlled snake.

A more *dynamic* constraint integration "mode" is often useful – once the live wire trace between the last seed point

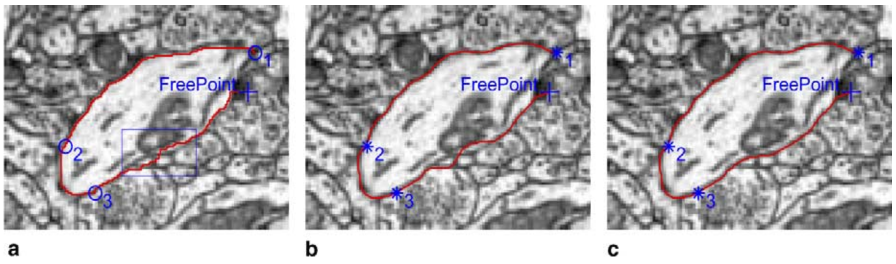

Fig. 10. Using United Snakes in dynamic mode to segment neuronal EM images. (a) Live wire boundary showing three seed points and free point (rectangle indicates a problem area). (b) Open snakes dynamically generated from the live wire traces and constrained by seed and free points. (c) Third snake corrected in the problem area using the mouse.

and the free point is formed, a corresponding open snake with constraints at the seed point and the free point is instantiated and set in motion. When the free point is chosen and collected as a seed point, this open snake is merged with the snake (if any) instantiated from previous live wire traces. All seed points are automatically applied as constraints. Fig. 10 illustrates this process where "+" indicates the current free point. The live wire and snake results are shown separately in the neuronal EM images in Figs. 10(a) and (b), respectively. Since the snake is automatically set in motion, the user may use the mouse-controlled springs to adjust it in any problematic areas along the snake trace (Fig. 10(c)).

6. Applying United Snakes

In this section, we apply United Snakes to several different medical image analysis tasks, demonstrating the generality, accuracy, robustness, and ease of use of the tool.

6.1. Segmenting neuronal dendrites in EM images

A neuronal dendrite is the receiving unit of a nerve cell. The area of contact between the dendrites of different cells is called a synapse and is located on the dendritic spines. In humans, morphological changes in dendritic spines are seen with aging and with diseases that affect the nervous system, such as dementia, brain tumors and epilepsy (Carlbom et al., 1994). Detailed anatomical models of dendritic spines and their synapses will provide new insights into their function, thus providing better opportunities to understand the underlying causes and effects of these diseases. To build such models, the dendrite must be segmented from the surrounding tissue in positive electron micrography (see Carlbom et al., 1994 for a detailed description of how snakes are used in reconstruction of 3D nerve cell models from serial microscopy). Here, we are interested in localizing nerve cell membranes, which appear dark in positive micrography.

In the United Snakes system, the user begins an image segmentation task using a live wire. An initial seed point is placed near the boundary of the object of interest. As the cursor, or free point, is moved around, the live wire, or trace, is interactively displayed from the seed point to

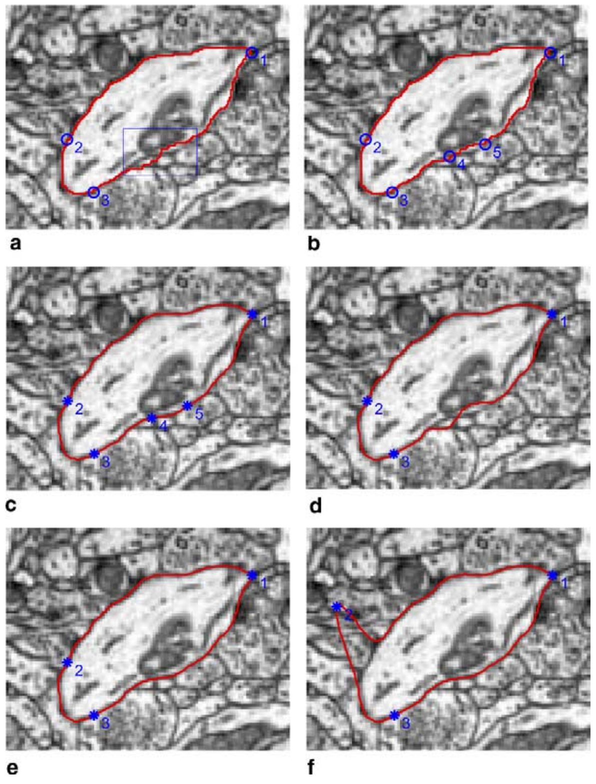

Fig. 11. Using United Snakes in static mode to segment neuronal EM images. (a) Approximate live wire boundary using just three seed points (rectangle indicates a problem area). (b) Additional seed points can improve live wire's accuracy. (c) Instantiated from the live wire traces in (b), the snake tolerates live wire errors and locks on cell boundary without further user interaction. (d) Instantiated from the live wire traces in (a), the snake "sticks" in the problem area, but it is easily adjusted (e) using the mouse. (f) Snake adjustment capability illustrated by moving constraint point 2.

the free point. If the displayed trace is acceptable, the free point is collected as an additional seed point. For example, we can capture an approximate cell boundary in Fig. 11(a) with just three seed points.

The live wire tends to stick to the object boundary using the seed points as a guide. The trace between the two adjacent seed points is frozen. The user has no further control over these traces other than backtracking. In order to generate a more accurate result in the area indicated by a rectangle, more seed points may be placed as in Fig. 11(b).

Although the live wire boundary is somewhat jagged and exhibits some small errors, it is in general as accurate as manual tracing, but more efficient and reproducible.

Next, we instantiate a snake from the live wire-generated boundary and use the seed points to constrain it. The user may select a shape function for the snake which is suitable for the object boundary. In our cell segmentation example, if the live wire result with five seed points is used to instantiate a finite difference snake, it is able to tolerate the live wire errors and very quickly and accurately lock onto the cell boundary without any further user interaction (Fig. 11(c)). Using the live wire result with three seed points, the snake becomes "stuck" in the problematic area (Fig. 11(d)) due to the live wire-generated boundary errors. However, this situation can be easily remedied using the mouse spring (Fig. 11(e)). Furthermore, as the snake is deforming, the hard constraints may be adjusted to refine the snake boundary. In Fig. 11(f), for example, constraint point 2 is moved to illustrate this snake boundary adjustment capability. By contrast, it is not nearly as easy to adjust a seed point in the live wire algorithm.

In summary, the information from live wire including the user guidance and expert prior knowledge is fully utilized by the snake; the snake very quickly locks onto the image features of interest with reasonable tolerance to mistakes in the live wire traces.

6.2. Dynamic chest image analysis

The aim of the dynamic chest image analysis task is to show focal and general abnormalities of lung ventilation and perfusion based on a sequence of digital chest fluoroscopy frames collected over a short time period (typically about 4 s) (Liang, 2000; Liang et al., 1997a,b, 1998, 2001, 2003). The project uses only plain X-ray fluoroscopy for the ventilation and perfusion studies; the radiation dose to patients is low and, unlike a nuclear medicine scan, no preparation is required before the examination and radioactive isotopes are unnecessary. The information gleaned from these images is helpful in several aspects of cardiothoracic radiology. Diseases directly related to the parameters being measured include pulmonary embolism, pulmonary emphysema, cardiac failure, congenital heart disease and other diseases (tumors, obstructive lesions or infections) which may change pulmonary ventilation and/or perfusion. The shapes and motions of the lung and heart give an indispensable clue to ventilation and perfusion examinations. Therefore, an essential first step for ventilation and perfusion analysis is the delineation of the lungs and the heart from each frame in a chest image sequence. The United Snakes system has been used for this purpose. Typically most of the user interactions to initialize and edit the snake are applied to the first image of the sequence only. The resulting snake is then propagated and deformed through the remaining frames of the image sequence.

We employ the dynamic integration mode, which was described in Section 5.1, to delineate the lung boundaries

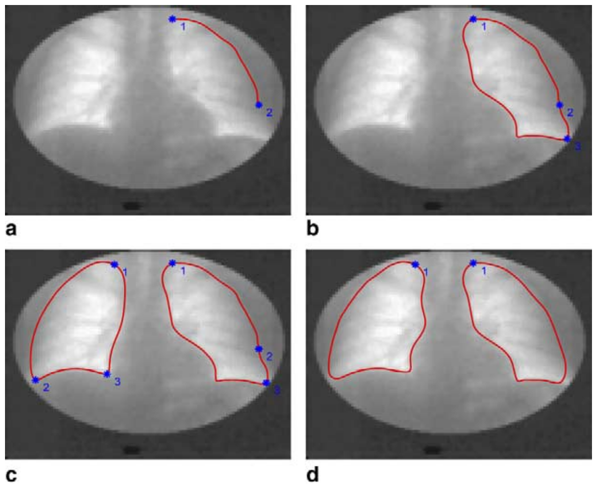

Fig. 12. Dynamic lung delineation with United Snakes. (a) An automatically instantiated Hermite snake. (b) United Snakes result for the left lung with three seed points. (c) The final result for both lungs. (d) Releasing hard constraints except those at both lung apices for lung motion tracking.

interactively for the first image in the sequence. Fig. 12(a) shows the first dynamically instantiated Hermite snake with two end points (seeds) applied as hard constraints. Three seed points are sufficient for delineating the left lung (Fig. 12(b)), similarly, for the right lung shown in Fig. 12(c).

In the dynamic integration method, all seed points are automatically applied as hard constraints. Although hard constraints can be dynamically adjusted, for motion tracking it is not convenient to perform hard constraint adjustments in each frame. Therefore, the United Snakes system allows the user to add or release hard constraints dynamically. The edge information at the lung apex is very weak and there is no observable motion in quiet breath. Consequently, it is desirable to maintain a hard constraint there. All other hard constraints are released for lung motion tracking. Fig. 12(d) shows a Hermite snake with the first seed imposed as hard constraint for each lung.

We apply the snake motion tracking mechanism on the entire image sequence, resulting in the registration of the lung from one frame to another. Since the first seed is applied as hard constraint, the snake can firmly stick at the apex, although the edge information there is rather weak. Fig. 13 illustrates the tracking result for every fifth image.

In the case of the heart, we first employ the static integration method for heart boundary tracing with the B-spline shape function in the first image. A least squares approximation to the initial curve shown in Fig. 3(b) with a cubic B-spline with 5 knots can be used as an initialization to a B-snake in Fig. 3(c). A hard constraint may be further imposed on control polygon node 3 to effectively bridge the gap along the heart boundary. The resulting B-snake for the first frame (Fig. 3(d)) is then used to track the heart motion through the whole image sequence (see

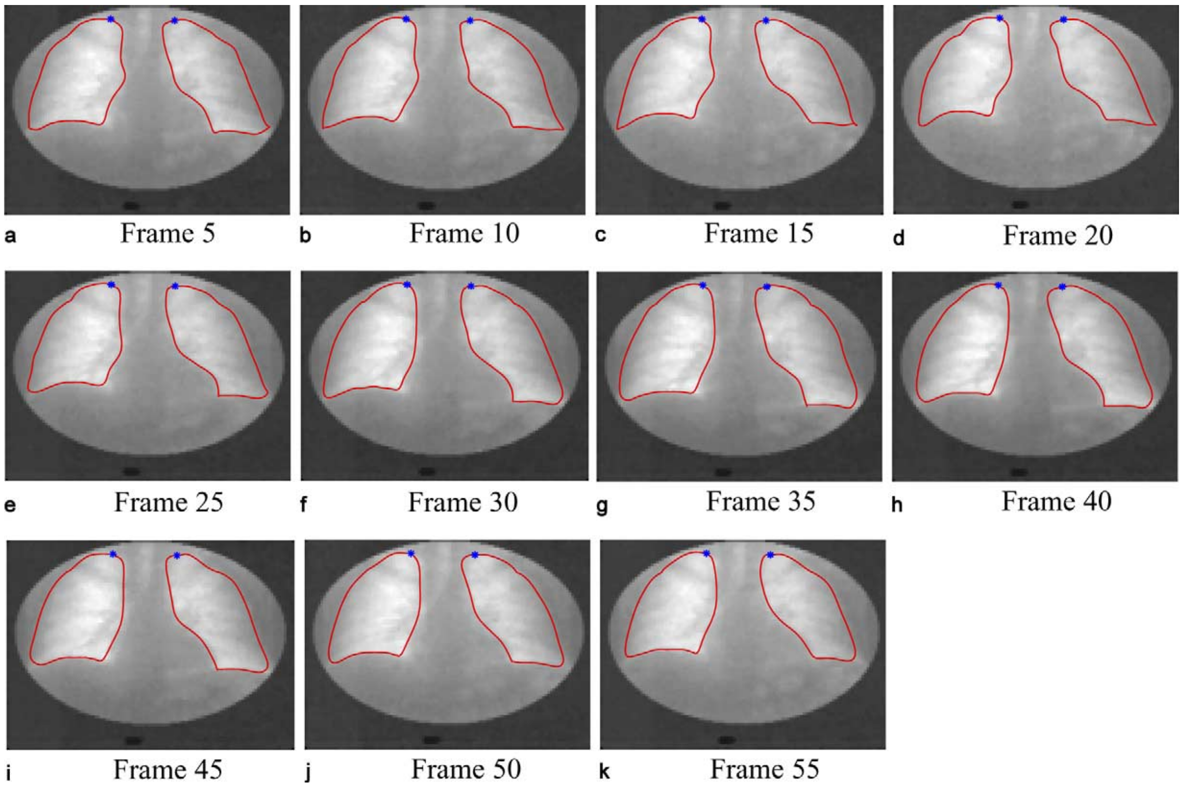

Fig. 13. Lung motion tracking result for every fifth frame.

Fig. 14). Since, in this patient orientation, there is no significant motion with the missing cardiac boundary, it is desirable to apply a hard constraint on the control polygon node 3. In the case of cardiac motion tracking, the hard constraint is not only effective for single images but also for the entire image sequence.

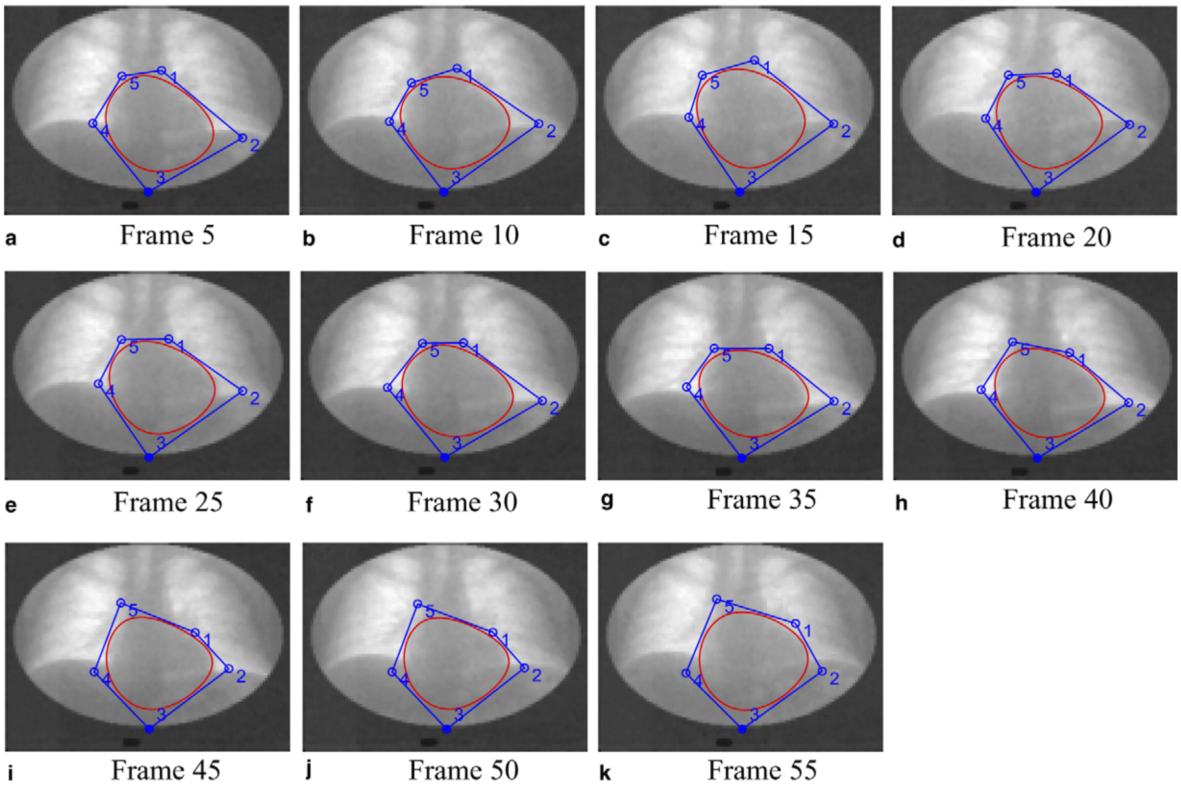

Fig. 14. Cardiac motion tracking result for every fifth frame.

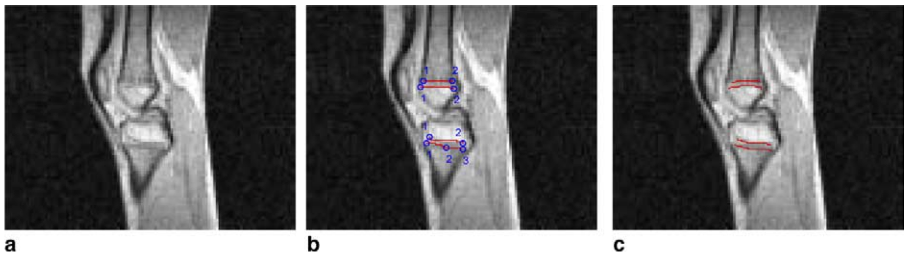

Fig. 15. Quantifying growth plates in MR images. (a) An MR growth plate image. (b) The live wire results. (c) The United Snakes results.

6.3. Quantifying growth plates in MR images

The aim of the growth plate image analysis task is to determine the right time for surgery for patients with abnormal growth of the legs. To this end, the four tiny (essentially horizontal) lines in the image (Fig. 15(a)) must be detected to quantify the growth plate.

In this scenario, it is difficult for the user to trace an initial contour for a snake manually because of the small size of the lines and the small distance between each pair of lines. However, live wire can be used to generate quickly an acceptable snake initialization with just two or three seed points as shown in Fig. 15(b). In the final results shown in Fig. 15(c), two hard boundary conditions are applied on each of four finite difference snakes.

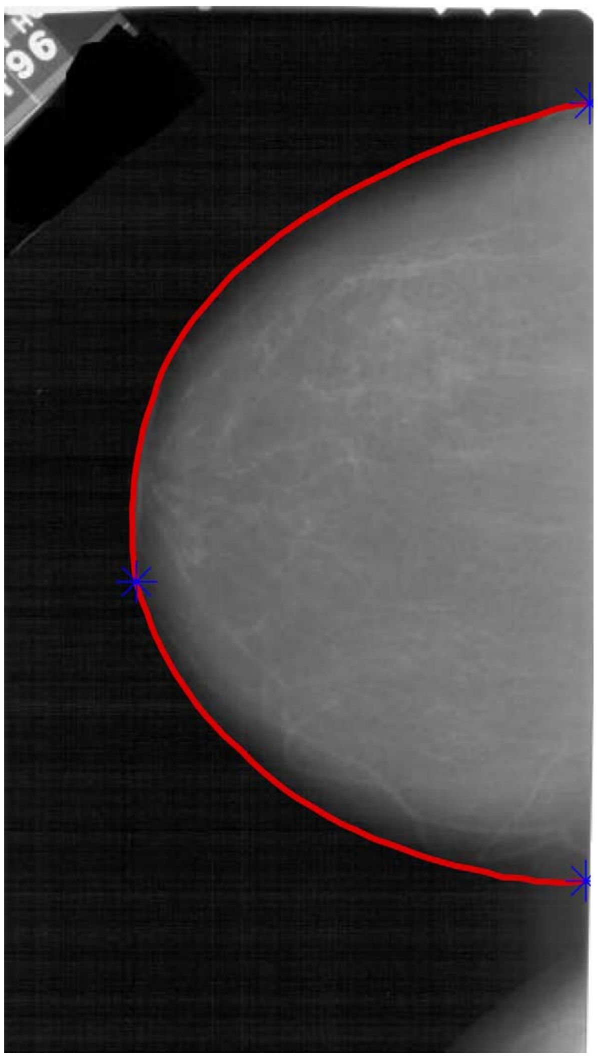

Fig. 16. Real-time isolation of the breast region in mammograms from a 3691 × 6466 pixel image using only three seed points.

6.4. Isolating the breast region in mammograms

The goal of the mammogram project is to use pattern recognition techniques to detect abnormalities in the breast tissue. The mammograms we are handling are very large with a typical resolution of 3500 × 6500 pixels, requiring about 30 MB of disk space. For effective and efficient abnormality detection, it is essential to isolate the breast region from the background (Ojala et al., 2000, 2001). For instance, the original mammogram in Fig. 16 has a resolution of 3691 × 6466. In United Snakes, we can achieve the real-time interactive segmentation of the breast region on the original mammogram with only two or three seed points using the truncated image pyramid technique proposed in Section 4.2.

7. Discussion and conclusion

It is concluded in (Falcão et al., 1998) that the main goals of research in interactive segmentation methods are (i) to provide as complete control as possible to the user of the segmentation process while it is being executed and (ii) to minimize the user's intervention and the total user time required for segmentation. The entire segmentation process may be thought of as consisting of two tasks: *recognition* and *delineation*. Recognition determines roughly where the object (boundary) is, while delineation defines precisely the spatial extent of the object region/boundary in the image. For practical applications, we have found that an additional task – *refinement* – is essential. The errors in reproducibility occur mostly in the vicinity of seed points (Mortensen and Barrett, 1999). In United Snakes, both live wire traces and hard constraint points can be interactively adjusted for refinement. Furthermore, dynamically instantiated snakes can tolerate the live wire errors and thus reduce the number of the seed points which are interactively given by the user. In other words, United Snakes provides more complete control to the user while further minimizing the user's intervention in the interactive segmentation process.

In summary, our United Snakes framework unites several snake variants with live wire to provide a general purpose tool for interactive medical image segmentation and tracking. The union of these techniques amplifies the efficiency, flexibility and reproducibility of the component techniques. The United Snakes technique offers more control for relatively less user interaction. As it quickly locks onto the image features of interest with reasonable tolerance to errors in live wire, the snake fully exploits the user guidance and expert prior knowledge captured by the initial live wire trace and the seed points. We have demonstrated the generality, accuracy and robustness of United Snakes in applications ranging from the segmentation of neuronal dendrites in EM images, to the analysis of dynamic chest images, to the quantification of growth plates, to the isolation of the breast region in mammograms, among other examples. We believe that United Snakes are in several ways superior to live wire or snakes alone.

We the creators of the United Snakes, in order to form a more perfect union of snake technologies, plan to incorporate within our framework, affine cell image decomposition methods for snake topological adaptability (McInerney and Terzopoulos, 2000), advanced snake motion tracking mechanisms (Terzopoulos and Szeliski, 1992; Blake and Isard, 1998), generic hard constraint mechanisms (Fua and Brechbühler, 1997; Fua, 1997), automatic learning and adaptation of shape functions to specific images, and other snake techniques. We anticipate that such efforts will further enhance the effectiveness of this image segmentation tool.

Acknowledgments

This work was carried out at the Turku Centre for Computer Science, Turku, Finland, and in the Department of Computer Science, University of Toronto, Toronto, Canada. J. Liang thanks the Turku Centre for Computer Science and the Science Foundation of Instrumentarium Corporation (now GE Healthcare) for their financial support, as well as the Faculty of Mathematics and Natural Sciences of the University of Turku for a faculty research award that supported the dynamic chest image analysis project. The authors gratefully acknowledge the insightful comments and suggestions of Prof. Alexandre X. Falcão, Prof. Pascal Fua, Prof. Dariu M. Gavrila, Prof. Timo Järvi, Prof. Ron Kimmel, Dr. Aaro Kiuru, Prof. Martti Kormano, Prof. Eric Mortensen, Dr. Erkki Svedström, Prof. Jayaram K. Udupa, and the anonymous referees. The chest images were acquired by Dr. Raimo Virkki and provided by Dr. Aaro Kiuru. The growth plate image was provided by Prof. Martti Kormano and Dr. Matti Sauna-aho. The mammogram was provided by Prof. Olli Navalainen. The cell image was obtained from Dr. Kristen Harris of the Harvard Medical School. The retinal angiogram was obtained from Dr. Piotr Jasiobedzki. The angiogram and spine images were provided courtesy of Eric Mortensen of Brigham Young University. The bladder image and the corpus callosum images were provided courtesy of Johannes Hug of Swiss Federal Institute of Technology, Switzerland. The MR wrist images were provided courtesy of Dr. Alexandre X. Falcão of State University of Campinas, Brazil.

Appendix A. Finite element Snakes formulation

The two coordinate functions $x(s,t)$ and $y(s,t)$ of the snake $v(s,t)$ are independent, we shall develop the finite element formulation and the corresponding matrix equations in terms of only one component $x(s,t)$. An identical form will be assumed for component $y(s,t)$. We apply Galerkin's method to the Euler–Lagrange equation for $x(s,t)$:

$$\mu\frac{\partial^2 x}{\partial t^2}+\gamma\frac{\partial x}{\partial t}-\frac{\partial}{\partial s}\left(\alpha\frac{\partial x}{\partial s}\right)+\frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2 x}{\partial s^2}\right)-q(x)=0, \quad (A.1)$$

which expresses the necessary condition for the snake at equilibrium. The average weighted residual is

$$I=\int_0^L\left(\mu\frac{\partial^2 x}{\partial t^2}+\gamma\frac{\partial x}{\partial t}-\frac{\partial}{\partial s}\left(\alpha\frac{\partial x}{\partial s}\right)+\frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2 x}{\partial s^2}\right)-q(x)\right)\times w(s)\,\mathrm{d}s=0, \quad (A.2)$$

where $w(s)$ is an arbitrary test function. By performing integrations by parts once for the third term and twice for the fourth term of (A.2), we arrive at the weak formulation of the snake model:

$$\int_0^L w\mu\frac{\partial^2 x}{\partial t^2}\,\mathrm{d}s+\int_0^L w\gamma\frac{\partial x}{\partial t}\,\mathrm{d}s+\int_0^L\left(\frac{\partial w}{\partial s}\alpha\frac{\partial x}{\partial s}\right)\mathrm{d}s+\int_0^L\left(\frac{\partial^2 w}{\partial s^2}\beta\frac{\partial^2 x}{\partial s^2}\right)\mathrm{d}s-\int_0^L wq\,\mathrm{d}s+b=0, \quad (A.3)$$

where

$$b=\left[-w\alpha\frac{\partial x}{\partial s}+w\frac{\partial}{\partial s}\left(\beta\frac{\partial^2 x}{\partial s^2}\right)-\frac{\partial w}{\partial s}\beta\frac{\partial^2 x}{\partial s^2}\right]_0^L \quad (A.4)$$

are the boundary conditions at the two boundary points, $s=0$ and $s=L$. We approximate $x(s,t)$ as

$$x(s,t)=\mathbf{N}(s)\mathbf{u}(t), \quad (A.5)$$

where $\mathbf{N}(s)=[N_1(s),N_2(s),\ldots,N_n(s)]$ are the shape functions and $\mathbf{u}(t)=[u_1(t),u_2(t),\ldots,u_n(t)]^{\mathrm{T}}$ are the n nodal variables (degrees of freedom) of the snake model, implying the derivatives of $x(s,t)$ are

$$\frac{\partial^2 x}{\partial t^2}=\mathbf{N}\ddot{\mathbf{u}},\quad \frac{\partial x}{\partial t}=\mathbf{N}\dot{\mathbf{u}},\quad \frac{\partial x}{\partial s}=\frac{\partial \mathbf{N}}{\partial s}\mathbf{u},\quad \frac{\partial^2 x}{\partial s^2}=\frac{\partial^2 \mathbf{N}}{\partial s^2}\mathbf{u}. \quad (A.6)$$

In Galerkin's method, the arbitrary test function w takes the form

$$w=\mathbf{Nc}, \quad (A.7)$$

where $\mathbf{N}$ are the same shape functions as in (A.5), and $\mathbf{c}$ is an arbitrary vector. As w is a scalar, we have

$$w=w^{\mathrm{T}}=\mathbf{c}^{\mathrm{T}}\mathbf{N}^{\mathrm{T}}. \quad (A.8)$$

Substituting (A.5)–(A.8) into (A.3) yields the snake equations of motion

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} - \mathbf{F} + \mathbf{P} = \mathbf{0}, \tag{A.9}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the damping matrix, $\mathbf{K}$ is the stiffness matrix, $\mathbf{F}$ is the force vector, and $\mathbf{P}$ is the boundary forces, defined as follows:

$$\mathbf{M} = \int_0^L \mathbf{N}^{\mathrm{T}}\mu\mathbf{N}\,\mathrm{d}s, \tag{A.10}$$

$$\mathbf{C} = \int_0^L \mathbf{N}^{\mathrm{T}}\gamma\mathbf{N}\,\mathrm{d}s, \tag{A.11}$$

$$\mathbf{K} = \mathbf{K}_\alpha + \mathbf{K}_\beta, \tag{A.12}$$

$$\mathbf{K}_\alpha = \int_0^L \left(\frac{\partial\mathbf{N}}{\partial s}\right)^{\mathrm{T}} \alpha \left(\frac{\partial\mathbf{N}}{\partial s}\right) \mathrm{d}s, \tag{A.13}$$

$$\mathbf{K}_\beta = \int_0^L \left(\frac{\partial^2\mathbf{N}}{\partial s^2}\right)^{\mathrm{T}} \beta \left(\frac{\partial^2\mathbf{N}}{\partial s^2}\right) \mathrm{d}s, \tag{A.14}$$

$$\mathbf{F} = \int_0^L \mathbf{N}^{\mathrm{T}} q\,\mathrm{d}s, \tag{A.15}$$

$$\mathbf{P} = \left[-\mathbf{N}^{\mathrm{T}}\alpha\frac{\partial\mathbf{N}}{\partial s} + \mathbf{N}^{\mathrm{T}}\frac{\partial}{\partial s}\left(\beta\frac{\partial^2\mathbf{N}}{\partial s^2}\right) - \left(\frac{\partial\mathbf{N}}{\partial s}\right)^{\mathrm{T}}\beta\frac{\partial^2\mathbf{N}}{\partial s^2}\right]_0^L \mathbf{u}. \tag{A.16}$$

Eq. (A.9) gives the finite element formulation for the whole snake. To achieve acceptable accuracy in the finite element approximation, the integration domain should be discretized into a number of small subdomains, resulting in the finite element mesh. That is, the snake contour is divided into small segments (elements), each of which can still be considered a snake. Applying (A.9) to an element e, we have $\mathbf{M}^e\ddot{\mathbf{u}}^e + \mathbf{C}\dot{\mathbf{u}}^e + \mathbf{K}^e\mathbf{u}^e - \mathbf{F}^e + \mathbf{P}^e = \mathbf{0}$, where $\mathbf{M}^e$ is the element mass matrix, $\mathbf{C}^e$ is the element damping matrix, $\mathbf{K}^e$ is the element stiffness matrix, $\mathbf{F}^e$ the element force vector, and $\mathbf{P}^e$ the element boundary forces applied to the boundary points of the element. Assembling the element matrices results in the system matrix motion equation (4). In a closed snake, the boundary forces will cancel each other. In an open snake, the boundary conditions may be assumed to be zero at the two ends. However, for generality and clarity, we introduce $\mathbf{g}$ for the external force vector.

To solve the motion equation (4), we replace the time derivatives of $\mathbf{u}$ with the backward finite differences

$$\ddot{\mathbf{u}} = (\mathbf{u}^{(t+\Delta t)} - 2\mathbf{u}^{(t)} + \mathbf{u}^{(t-\Delta t)})/(\Delta t)^2, \quad \dot{\mathbf{u}} = (\mathbf{u}^{(t+\Delta t)} - \mathbf{u}^{(t)})/\Delta t,$$

where the superscripts denote the quantity evaluated at the time given in the parentheses and the time step is Δt. This yields the update formula

$$\mathbf{A}\mathbf{u}^{(t+\Delta t)} = \mathbf{b}\mathbf{u}^{(t)} + \mathbf{c}\mathbf{u}^{(t-\Delta t)} + \mathbf{g}, \tag{A.17}$$

where $\mathbf{A} = \mathbf{M}/(\Delta t)^2 + \mathbf{C}/\Delta t + \mathbf{K}$ and $b = 2\mathbf{M}/(\Delta t)^2 + \mathbf{C}/\Delta t$ and $\mathbf{c} = -\mathbf{M}/(\Delta t)^2$. Because $\mathbf{A}$ is symmetric and banded, it can be economically saved in skyline storage, and efficiently factorized uniquely into the form $\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$, where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{D}$ is a diagonal matrix (Bathe and Wilson, 1976). The solution $\mathbf{u}^{(t+\Delta t)}$ to (A.17) is obtained by first solving $\mathbf{Ls} = \mathbf{bu}^{(t)} + \mathbf{cu}^{(t-\Delta t)}$ with forward substitution, then $\mathbf{L}^{\mathrm{T}}\mathbf{u} = \mathbf{D}^{-1}\mathbf{s}$ with backward substitution. Since $\mathbf{A}$ is constant, only a single factorization is necessary. Therefore, at each time step only the forward/backward substitutions are performed to integrate the snake equations of motion forward through time.

References

Amini, A., Weymouth, T., Jain, R., 1990. Using dynamic programming for solving variational problems in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (9), 855–867.

Barrett, W., Mortensen, E., 1997. Interactive live-wire boundary extraction. Medical Image Analysis 1 (4), 331–341.

Bathe, K.-J., Wilson, E.L., 1976. Numerical Methods in Finite Element Analysis. Prentice-Hall, Englewood Cliffs, NJ.

Blake, A., Isard, M., 1998. Active Contours. Springer, Berlin.

Carlbom, I., Terzopoulos, D., Harris, K., 1994. Computer-assisted registration, segmentation, and 3D reconstruction from images of neuronal tissue sections. IEEE Transactions on Medical Imaging 13 (2), 351–362.

Cohen, L., Cohen, I., 1993. Finite element methods for active contour models and balloons for 2D and 3D images. IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (11), 1131–1147.

Cohen, L., Kimmel, R., 1997. Global minimum for active contour models: a minimal path approach. International Journal of Computer Vision 24 (1), 57–78.

Dubuisson-Jolly, M.P., Gupta, A., 2001. Tracking deformable templates using a shortest path algorithm. Computer Vision and Image Understanding 81 (1), 26–45.

Falcão, A.X., Udupa, J.K., 1997. Segmentation of 3D objects using live wire. In: SPIE on Medical Imaging 1997, vol. 3034, Newport Beach, CA, pp. 228–239.

Falcão, A.X., Udupa, J.K., Miyazawa, F.K., 2000. An ultra-fast user-steered segmentation paradigm: live-wire-on-the-fly. IEEE Transactions on Medical Imaging 19 (1), 55–62.

Falcão, A.X., Udupa, J.K., Samarasekera, S., Hirsch, B.E., 1996. User-steered image boundary segmentation. In: Proceedings of SPIE on Medical Imaging, vol. 2710, Newport Beach, CA, pp. 278–288.

Falcão, A.X., Udupa, J.K., Samarasekera, S., Sharma, S., 1998. User-steered image segmentation paradigms: live wire and live lane. Graphical Models and Image Processing 60, 233–260.

Fua, P., 1997. Model-based optimization: an approach to fast, accurate, and consistent site modeling from imagery. In: Firschein, O., Strat, T.M. (Eds.), RADIUS: Image Understanding for Intelligence Imagery. Morgan Kaufmann, Los Altos, CA.

Fua, P., Brechbühler, C., 1997. Imposing hard constraints on deformable models through optimization in orthogonal subspaces. Computer Vision and Image Understanding 65, 148–162.

Gavrila, D.M., 1996. Hermite deformable contours. In: Proceedings of the International Conference on Pattern Recognition, Vienna, Austria, pp. 130–135.

Grzeszczuk, R., Levin, D., 1994. Brownian strings: segmenting images with stochastically deformable contours. In: Robb, R. (Ed.), Proceedings of the Third Conference on Visualization in Biomedical Computing (VBC'94), SPIE Proceedings, vol. 2359. SPIE, pp. 72–89.

Hyche, M.E., Ezquerra, N.F., Mullick, R., 1992. Spatiotemporal detection of arterial structure using active contours. In: Proceedings of the Second Conference on Visualization in Biomedical Computing (SPIE vol. 1808), Chapel Hill, NC, October, pp. 52–62.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: active contour models. International Journal of Computer Vision 1 (4), 321–331.

Kwon, Y.W., Bang, H., 1997. The Finite Element Method Using MatlabCRC Mechanical Engineering Series. CRC Press, Boca Raton, FL.

Liang, J., 2000. Dynamic chest image analysis: new model-based methods for dynamic pulmonary imaging and other applications. Turku Centre for Computer Science, Turku, Finland, December [TUCS Dissertation No. 31] (available at <http://www.cs.toronto.edu/~liang/phddissertation.pdf>).

Liang, J., Haapanen, A., Järvi, T., Kiuru, A., Kormano, M., Svedström, E., Virkki, R., 1998. Dynamic chest image analysis: model-based pulmonary perfusion analysis with pyramid images. In: Hoffman, E.A., (Ed.), Medical Imaging 1998: Physiology and Function from Multidimensional Images, San Diego, CA, pp. 63–72.

Liang, J., Järvi, T., Kiuru, A., Kormano, M., Svedström, E., 2001. Dynamic chest image analysis: evaluation of model-based perfusion analysis with pyramid images. In: Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Istanbul, Turkey, October, pp. 415–420 (invited paper).

Liang, J., Järvi, T., Kiuru, A., Kormano, M., Svedström, E., 2003. Dynamic chest image analysis: model-based perfusion analysis in dynamic pulmonary imaging. EURASIP Journal on Applied Signal Processing (5), 437–448 (special issue on Advances in Modality-Oriented Medical Image Processing).

Liang, J., Järvi, T., Kiuru, A., Kormano, M., Svedström, E., Virkki, R., 1997. Dynamic chest image analysis: model-based ventilation study with pyramid images. In: Hoffman, E.A. (Ed.), Medical Imaging 1997: Physiology and Function from Multidimensional Images, Newport Beach, CA, pp. 81–92.

Liang, J., McInerney, T., Terzopoulos, D., 1999a. United snakes. In: Proceedings of the Seventh International Conference on Computer Vision (ICCV'99), Kerkyra (Corfu), Greece, September. IEEE Computer Society Press, Silver Spring, MD, pp. 933–940.

Liang, J., McInerney, T., Terzopoulos, D., 1999b. Interactive medical image segmentation with united snakes. In: Proceedings of the Second International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 99), Cambridge, England, September. Springer, Berlin, pp. 116–127.

Liang, J., Virkki, R., Järvi, T., Kiuru, A., Kormano, M., Svedström, E., 1997. Dynamic chest image analysis: evaluation of model-based ventilation study with pyramid images. In: Zurawski, R., Liu, Z.-Q. (Eds.), IEEE First International Conference on Intelligent Processing Systems, Beijing, China, pp. 989–993.

McInerney, T., Terzopoulos, D., 2000. Topology adaptive snakes. Medical Image Analysis 4, 73–91.

McInerney, T., Terzopoulos, D., 1996. Deformable models in medical image analysis: a survey. Medical Image Analysis 1 (2), 91–108.

Menet, S., Saint-Marc, P., Medioni, G., 1990. B-snakes: implementation and application to stereo. In: Proceedings DARPA, pp. 720–726.

Mortensen, E.N., 2000. Simultaneous multi-frame subpixel boundary definition using toboggan-based intelligent scissors for image and movie editing. Ph.D. Thesis, Department of Computer Science, Brigham Young University, Provo, UT.

Mortensen, E.N., Barrett, W.A., 1995. Intelligent scissors for image composition. In: Proceedings of Computer Graphics (SIGGRAPH'95), Los Angeles, CA, August, pp. 191–198.

Mortensen, E.N., Barrett, W.A., 1998. Interactive segmentation with intelligent scissors. Graphical Models and Image Processing 60, 349–384.

Mortensen, E.N., Barrett, W.A., 1999. Toboggan-based intelligent scissors with a four parameter edge model. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, June, pp. 452–458.

Neuenschwander, W., Fua, P., Székely, G., Kübler, O., 1994. Initializing snakes. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'94). IEEE Computer Society Press, Silver Spring, MD, pp. 658–663.

Ojala, T., Liang, J., Näppi, J., Nevalainen, O., 2000. Interactive segmentation of the breast region from digitized mammograms. In: Proceedings of the IASTED International Conference on Signal Processing and Communications (SPC 2000), Marbella, Spain, September, pp. 132–136.

Ojala, T., Näppi, J., Nevalainen, O., 2001. Accurate segmentation of the breast region from digitized mammograms. Computerized Medical Imaging and Graphics 25 (1).

Sethian, J.A., 1997. A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences of the United States of America 93 (4), 1591–1595.

Singh, A., Goldgof, D., Terzopoulos, D., 1998. Deformable Models in Medical Image Analysis. IEEE Computer Society Press, Silver Spring, MD.

Staib, L., Duncan, J., 1992. Boundary finding with parametrically deformable models. IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (11), 1061–1075.

Terzopoulos, D., Qin, H., 1994. Dynamic NURBS with geometric constraints for interactive sculpting. ACM Transactions on Graphics 13 (2), 103–136.

Terzopoulos, D., Szeliski, R., 1992. Tracking with Kalman snakes. In: Blake, A., Yuille, A. (Eds.), Active Vision. MIT Press, Cambridge, MA, pp. 3–20.

Terzopoulos, D., Witkin, A., Kass, M., 1988. Constraints on deformable models: recovering 3D shape and nonrigid motion. Artificial Intelligence 36 (1), 91–123.

Xu, C., Prince, J.L., 1998. Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing 7 (3), 359–369.

Zienkiewicz, O., Taylor, R., 1989. The Finite Element Method. McGraw-Hill, New York, NY.