SketchSnakes: Sketch-Line Initialized Snakes for Efficient Interactive Medical Image Segmentation

T. McInerney

Dept. of Computer Science
Ryerson University
350 Victoria St., Toronto, ON, Canada M5B 2K3 Ryerson University

Abstract

We present an intuitive, fast and accurate 2D interactive segmentation method that combines a general subdivision-curve Snake possessing powerful editing capabilities, with a novel sketch-line user initialization process, and a pen input device. Using the pen (or a mouse), the Snake is quickly and precisely initialized with a few quick sketch lines drawn across the width of the target object. The smooth contour constructed using these lines is extremely close to the position and shape of the object boundary. This makes the Snake's task of snapping to the object boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. We apply our Snake to the segmentation of several 2D medical images to demonstrate it's efficiency, accuracy and robustness. We also compare *Sketch Snakes* to Adobe Photoshop's *Magnetic Lasso* [1] as well as a recent graph-cut based image cutout tool known as *Snap* [2] in order to highlight SketchSnakes effectiveness.

Key words: Image Segmentation, Snakes, User Interaction, Deformable Models

Email addresses: tmcinern@scs.ryerson.ca (T. McInerney), tmcinern@scs.ryerson.ca (T. McInerney).

URLs: www.scs.ryerson.ca/~tmcinern (T. McInerney), www.scs.ryerson.ca/~tmcinern (T. McInerney).

1 Introduction

Robust, fully automatic medical image segmentation systems have proved extremely challenging to develop. Consequently, semiautomatic techniques, such as Snakes [3] and LiveWire [4–8] have become widely popular. Since their inception in the late 1980's, many improvements to Snakes models for medical image applications have been proposed. For example, many Snakes variants have been developed that ameliorate the well-known initialization sensitivity problem, allowing "one-click" initialization and the ability of these active contour models to "flow" into complex shapes [9–11] as well as change their connectivity[12,13]. These improvements have proved effective for some segmentation tasks, especially involving very complex-shaped objects (e.g. blood vessels or the cortical surface of the brain) or numerous small convex-shaped objects (e.g. blood cells). However, these models often fail in noisy images that contain missing object edges and this failure results in significant user intervention. Unfortunately, the mathematical formulations and numerical algorithms that enabled the simple initialization and complex shape extraction were often unable to also provide the capability to perform interactive steering, high-level control, and intuitive editing during the segmentation process, and the ability to dynamically impose shape constraints.

The deficiencies in the "flow"-type Snakes segmentation paradigm have led us to explore an alternative research direction [14–16], creating interactive active contour models for segmenting medical images and image sequences which exhibit a significant amount of noise. These models are designed to complement and integrate with the "flow"-type Snakes. More specifically, our goals are to minimize the time and labor required to interactively initialize a Snake in noisy images, imbue Snakes with precise, intuitive steering and editing capabilities, and create an accurate, noise-insensitive model.

In this paper, we present *SketchSnakes* - a general, robust, highly efficient active contour model with a simple formulation - in an effort to meet our goals. Our new active contour model is constructed using a recently published subdivision curve algorithm, providing the robustness against noise of a finite-element or B-spline Snake while remaining simple enough to easily impose constraints and maintain geometric flexibility. This new subdivision curve Snake can be quickly initialized with a few rough sketch lines drawn across the width of the target object such that the initial curve is extremely close in shape to the object (Figure 1). This novel initialization process makes the Snake's task of snapping to the object boundary much simpler and hence more likely to succeed in noisy images with minimal user editing. SketchSnakes are inherently smooth and naturally multi-scale, allowing coarse to fine and other custom data-fitting "schedules" if desired. We enhance user control over the model by making use of a tablet personal computer with a pen input device to allow easy, fast, precise positioning and sketching of points and lines directly on the tablet screen. We demonstrate our sketch-line based Snake on several 2D medical images and discuss and compare its performance with Adobe Photoshop's Magnetic Lasso as well as a recent graph-cut based image cutout tool known as *Snap* [1] .

[1] Snap is a product of Digital Film Tools and is implemented as an Adobe Photoshop plugin.

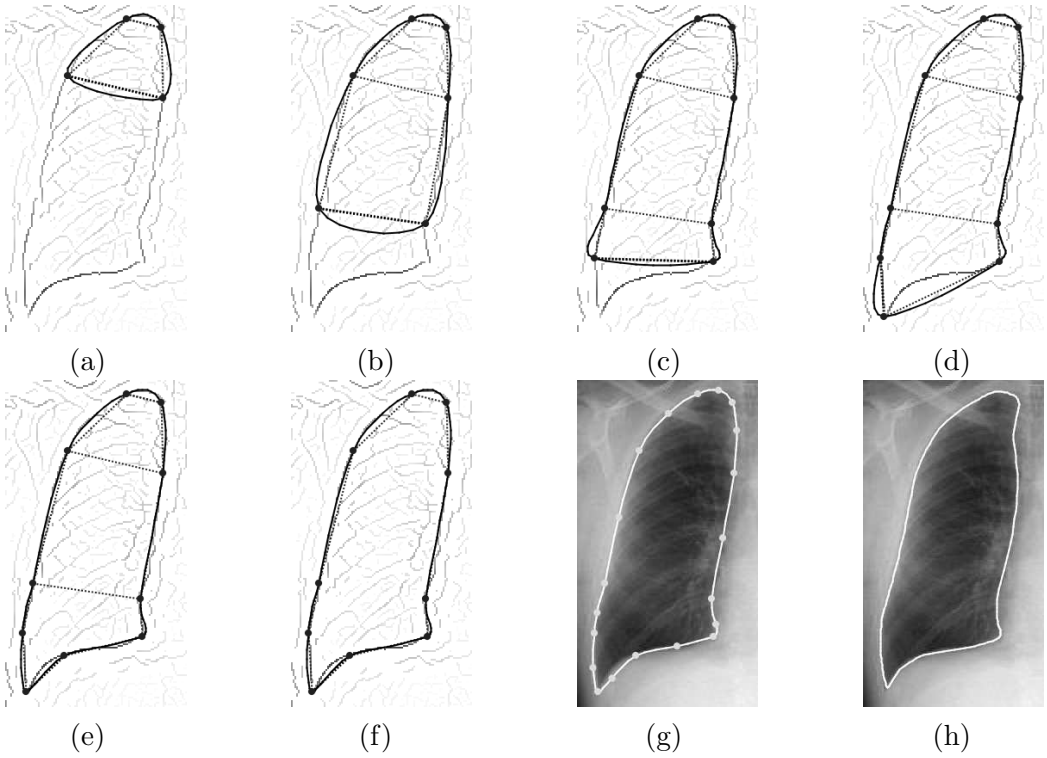


Fig. 1. SketchSnakes segmentation of the lung from a chest x-ray. (a)-(e) A few seconds are required to sketch several lines across the width of the lung. The resulting subdivision curve, its control polygon, and sketch lines are shown in an edge detected image. (f)(g) Note the accuracy of the initial SketchSnake. (h) The Snake is easily able to instantly snap to the lung boundary and requires little or no editing.

2 Interactive Segmentation Techniques

Over the past decade, several interactive segmentation techniques have emerged as popular tools for (medical) image segmentation. In this section, we provide an overview of these techniques as well as a brief comparison with SketchSnakes. A more detailed discussion of SketchSnakes is presented in Section 6.

2.1 Snakes

Snakes are energy-minimizing contours controlled by internal and external energies. The internal energy enforces smoothness of the contour and the external energy attracts the contour towards salient features in the image, such as edges (which correspond to object boundaries). An iterative procedure is typically used to minimize the energy causing the snake to converge (i.e. lock-on or "snap") to the boundary of the target object in the image. Unlike more traditional noise-sensitive techniques such as edge linking, the model-based Snake proposed by Kass, Witkin, and Terzopoulos [3], considered the target object boundary as single connected structure and was designed to be interactive via user-imposed constraint energy terms. Using a mouse or pen input device, users are able to intuitively attract the

Snake via virtual springs, pulling a Snake off spurious or incorrect image edges onto the correct edges (i.e. moving it out of one local energy minima into another). Although Snakes provide a powerful and general approach to image segmentation, the original finite difference formulation in [3] had several limitations that affected repeatability and efficiency for some medical image segmentation tasks:

- *Small capture range*: there is no external force on Snake points in regions of homogeneous image intensity due to the almost zero image gradient magnitude in these areas.
- *Local energy minimization*: local optimization techniques are commonly used to minimize the Snake energy, forcing the user to initialize the Snake close to the target object boundary to ensure convergence to the desired solution. User initialization is typically performed with tedious tracing actions or boundary point selection, often causing the Snake to lock on to neighboring object edges or noisy edges. The user is then forced to spend time pulling the Snake onto the correct edge.
- *Geometric inflexibility*: the original Snakes model had a fixed geometric parameterization and this formulation, in conjunction with the internal deformation energies, made it incapable of deforming to fit shapes with deep concavities or shapes with significant branching or protrusions without significant and tedious user initialization [2].
- *Extraneous degrees of freedom*: the original finite difference formulation and initialization process often resulted in too many degrees of freedom to efficiently segment objects in very noisy images.
- *Inadequate editing capabilities*: user interaction based solely on virtual springs does not provide the degree of control and precision required to efficiently and accurately edit the Snake.

2.1.1 Finite Element and B-Spline Snakes

Snakes have been constructed using higher-order basis functions, such as polynomials of degree 2 or higher [9,14,15] [17–19]. These models are built using finite elements or parametric B-spline curves. They provide a compact, local representation of a curve, converge faster to the solution than the original finite difference Snakes, often are inherently smooth, provide an analytic representation over the whole Snake (rather than only at Snake nodes), and require fewer degrees of freedom for the same level of accuracy as finite difference Snakes. These properties are advantageous for segmentation or tracking tasks involving noisy images where the target object boundaries may exhibit significant gaps in the edge image.

A disadvantage is the mathematical formulation and software implementation of these types of Snakes is more complex and less flexible than the simple finite difference formulation. Furthermore, the smoothness of the finite element Snakes is achieved through the energy minimization process rather than through mathematical construction. The result is that only force-based editing (i.e. via virtual springs) is available for finite element Snakes whereas precise, controllable geometric editing is possible with the mathematically smooth B-spline

[2] The original formulation was also incapable of changing its topology and could not disconnect and reconnect itself when deforming around objects embedded within the target object. We do not address topology adaptation in this paper.

Snake. On the other hand, a B-spline curve is an approximating curve and does not interpolate its control points (the B-spline Snake degrees-of-freedom) which can prove un-intuitive in many interactive segmentation scenarios. In contrast, the "control points" of the finite element Snake (the Snake nodes) are the same as its degrees-of-freedom. This property makes the initialization and (interactive) constraint of Snake nodes more user-friendly for the finite element Snake.

Our subdivision curve-based SketchSnakes combine the properties of the finite element and B-spline Snakes, while maintaining the simple, flexible formulation and implementation of the original finite difference Snake. In fact, in [20] and [21], a subdivision curve (see Section 3.1) has been shown to be equivalent (in the limit) to a quadratic and cubic B-spline, respectively, while avoiding the complex analytical definition of B-splines. Subdivision curve Snakes have been previously proposed in [22] but the interaction capabilities, initialization process, constraint mechanisms, and external force usage were underdeveloped when compared to SketchSnakes. In [22], they opted for a traditional coarse-to-fine model fitting approach, using a boundary point selection initialization procedure. They also did not utilize the finest level curve in the curve subdivision process to gather all available image feature information in the vicinity of the Snake and distribute this information to the coarse level control polygon (which act as the degrees of freedom of in our subdivision curve Snake formulation). Their approach does not take full advantage of the inherent properties of subdivision curves.

2.2 "Flow" Snakes

To create a more automated Snake that is less sensitive to its initial position and to overcome the geometric and topology limitations of standard parametric Snakes, researchers have added various constraint energies, forces, and other algorithmic mechanisms to the parametric Snakes model formulation. Balloon (or inflation) forces were introduced [9] and automatic snake element subdivision mechanisms in [11,23,12,24]. Many researchers have constructed implicit Snakes models by adopting Osher and Sethian's [25] level-set evolution technique to the image segmentation problem (see [13] for a complete review). These models are formulated as evolving contours ("propagating fronts") which define the level set of some higher-dimensional surface over the image domain.

The inflation or propagation forces of the above techniques significantly increased the capture range of a Snake. In this case, the model can be simply initialized using, for example, one mouse click to create a small circular Snake inside the target object. The inflation force expands the model and it automatically subdivides, allowing it to "*flow*" into complex shapes. As the model approaches the boundary of the target object, external image forces (based on edge strength, image region statistics, or area minimization) oppose the inflation, stopping the model on the boundary.

All of these flow-type Snakes can and do work well in many segmentation scenarios, especially when the image feature map is relatively clean and homogeneous. However, clinical images are noisy, contain many uninteresting edges and regions of low contrast, contain gaps in the object boundary, or exhibit a complex texture. Hence, these more automatic techniques

may not generate the expected result - the added automation does not come without a cost. Some interactive control (steering capability) over the model is lost and the gaps in the object boundaries (especially in noisy images) may allow the model to leak through, requiring user intervention in the form of barriers. Setting up barriers in advance can be problematic - it is often not obvious where barriers are needed. Stopping a leak during the fast flow of the model is also difficult. Another common problem with flow-type Snakes is they may not completely flow into all regions of the target object. In this case, the user is required to plant other Snake seeds, change the Snake parameters and re-run the algorithm, or provide additional constraint information to force the Snake into these regions. All of these problems can lead to segmentation inefficiencies.

In summary, there exist segmentation tasks where the loss of interactive model control in the flow-Snakes may not be offset by the gain in automation, especially when they are applied to noisy images. Our subdivision curve Snake combined with the sketch-line initialization procedure, can be effectively applied in these situations by providing noise insensitivity and superior user steering capabilities. Although designed to be a completely general technique, SketchSnakes are optimized for simple-shaped to moderately complex-shaped objects, and for clean to noisy images. Flow-type Snakes are perhaps best suited for very complex-shaped objects (such as arterial structures or contours of the cortical surface) or numerous small objects, in relatively clean images where a higher degree of automation is desirable.

2.3 Semi-automatic Boundary Tracing

Subsequent to the introduction of Snakes, a related technique, known as LiveWire or Intelligent Scissors [4,26,5–8] has emerged as an effective interactive boundary tracing tool which allows user interaction and control over the segmentation process. Adobe Photoshop's image cutout tool [1] *Magnetic Lasso*, is an example of this type of algorithm. Like Snakes, the idea behind the LiveWire technique is to perform the segmentation with minimal user interaction while at the same time allowing the user to guide or "steer" the segmentation process.

In this technique, the user uses the mouse to initially specify a seed point on the object boundary and then moves the mouse to advance the cursor to a point further along the object boundary. A globally optimum path (*the trace*) from the initial seed point to the current point is computed and displayed in real time. The optimal paths are determined by assigning a set of features and cost functions to boundary elements (such as edge strength), and then finding the minimum cost path. As the user moves the cursor slightly, different paths are computed and displayed in real-time, akin to an electrical arc - hence the name "LiveWire". If the cursor moves close to the boundary, the LiveWire *snaps* to the boundary (assuming the cost functions are set correctly). If the user is satisfied with the computed boundary segment, the user "deposits" the cursor point. This point becomes the new seed point and the recursive process continues.

Although LiveWire methods are much faster and more reproducible than manual tracing of object boundaries and other traditional pixel selection tools and provide the user with good control during the segmentation process, these techniques can still demand a large amount

of concentration from the user. A few problem scenarios are listed below:

- Once the user deposits the curser point when tracing the boundary, the point is collected as a seed point and the trace is frozen and added to the extracted object boundary. The user has no further control over the trace other than return to the previous point and remove it. This correction increases segmentation time and user interaction when dealing with a noisy and low contrast object or complex boundary. There is never a perfect match between the features used by the LiveWire algorithm and the desired object boundary. As a result, the user often must control the mouse carefully. If a mistake is made, the user must "backtrack" and try again.
- Live wire still requires tracing-like actions to position the cursor. Moving the cursor around an entire object under the control of a mouse (or other input device) can be tedious and therefore fatiguing, especially for a complex-shaped object (see Section 6).
- When the desired object boundary has a relatively weak edge close to an insignificant but strong edge, the live wire snaps to the strong edge rather than the desired weak boundary. Falcão and Udupa [8] developed a technique called "live wire on-the-fly" in an attempt to minimize this problem but it assumes that edge characteristics are relatively consistent along the entire object boundary.
- Objects which are complex in shape (contain highly curved regions or protrusions for example) often force the user to deposit many seed points.
- LiveWire is not as amenable to segmenting multiple image slices in a time series (object tracking) or volume image as are Snakes. This is due to its fundamentally tracing-based nature. Snakes on the other hand, can be modified anywhere at any time by the user.
- Unlike Snakes, LiveWire is limited in its ability to be controlled by "intelligent" high-level control algorithms.
- In some LiveWire systems [1], any segmentation errors must be cleaned up with other traditional selection tools.

In [14] we claimed that Snakes and LiveWire are complementary for certain tasks and combine them into one system. Unlike flow-type Snakes which attempt to provide initialization insensitivity, our idea was to increase the efficiency of interactive initialization, enabling the user to instantiate (i.e. construct, initialize, and activate) Snakes quickly and with minimal effort by exploiting the strengths of LiveWire.

In SketchSnakes, we extend this idea of efficient initialization and present an alternative strategy that removes the requirement of tracing by using simple, intuitive, fatigue-free line-stroke actions. Drawing (i.e. "sketching") short line segments is a task well-suited to a pen input device and is performed directly on a pen-enabled screen, such as on a Tablet PC. Furthermore, by using a subdivision curve to represent the Snake, precise control over the initialization, fitting and editing is seamlessly integrated into a single work-flow process and these operations can each be performed at any time.

Recently, researchers have developed interactive region painting methods for image segmentation [27–29] and these graph-cut based algorithms are showing great promise for cutting out complex-shaped objects, especially from color images. An Adobe Photoshop plugin known as *Snap* [2], from Digital Film Tools, is an example of this type of algorithm. The user gives loose hints as to which part of the image are foreground and which are background by simply painting rough strokes on the foreground and several strokes on the background. An optimization algorithm uses these inputs hints to extract the actual object boundary. These region-based methods allow the user to operate at whatever scale they want and they also show partial results. After each hint, the foreground and background specification becomes more and more accurate.

The main problem with region-based methods is that there are often cases where the features used by the region detection algorithms do not match the target object. It is difficult to provide the correct hints in ambiguous areas of low contrast edges or shadows. In medical image segmentation, these problems are especially apparent as the gray-scale images are noisy and contain less information than color images. For example, this technique works well on fairly clean images on bright objects such as bone in a CT scan. For noisier objects the technique can, like the flow-type Snakes, leak through or not flow completely into the object. In this case the user needs to provide more foreground and/or background strokes.

Although this "painting" paradigm has simple interaction with fast feedback, it lacks the precision of shape model-based interaction and editing. The user can become frustrated attempting to draw small strokes in noisy boundary regions to generate the correct segmentation. It is also often difficult to determine where to draw foreground and background strokes. In [28] (and in Snap [2]) the authors attempt to overcome these problems by generating a polygonal boundary representation from the region found in the painting phase and then editing the polygon with several editing modes: clicking and dragging vertices, using the polygon as a soft constraint for a subsequent region optimization, and manually setting polygon vertices as hard constraints for a subsequent region optimization.

However, unlike SketchSnakes this system requires two types of input and interactions and it is often not obvious when to switch between the two input "modes". SketchSnakes is a purely boundary-based technique with seamless and consistent transitions between input and editing or steering. Furthermore, although it is boundary-based, by sketching lines across the width of the target object to construct the initial contour, Sketch-Snakes initialization acts in a region-oriented manner.

3 Subdivision Curve Snakes

The success of interactive shape-model based segmentation techniques, such as Snakes, is still heavily dependent upon the generality, controllability, and simplicity of the underlying shape representation scheme. We propose the use of subdivision curves (and surfaces in 3D)

[30] as a very general shape representation basis for interactive deformable models which have a simple formulation. We have developed a Java-based, open-source, highly extensible Snakes package known as JESS[31] for the rapid construction of 2D segmentation systems, which includes several subdivision curve Snake implementations. This package was extended to build our SketchSnakes system. In the following sections we provide a brief overview of subdivision curves, and then proceed to describe the construction of a SketchSnake using these curves.

3.1 Subdivision Curves

(a) (b) (c) (d)

Fig. 2. Example of corner cutting subdivision to generate a subdivision curve. (a) Control polygon. (b) After one subdivision. (c) Two subdivisions. (d) Three subdivisions

The underlying idea behind subdivision methods [30] is very simple, using geometric algorithms to progressively subdivide a control polygon (or control mesh in 3D). Repeated subdivision leads to a hierarchy of increasingly refined models which approach the *limit curve* (*limit surface* in 3D). For example, one such algorithm in 2D is based upon Chaikin's[20] "corner cutting" method, where a new control polygon is generated by cutting the corners off the original one (Figure 2). Assume the n vertices of a control polygon are $\mathbf{P}^0 = \{\mathbf{p}_0^0, \mathbf{p}_1^0, \ldots, \mathbf{p}_{n-1}^0\}$, where the superscript denotes the level of subdivision. Chaikin's scheme creates two new vertices between each subsequent pair of vertices, say $\mathbf{p}_i^k$ and $\mathbf{p}_{i+1}^k$, of the original polygon as:

$$\begin{aligned} \mathbf{p}_{2i}^{k+1} &= \tfrac{3}{4}\mathbf{p}_i^k + \tfrac{1}{4}\mathbf{p}_{i+1}^k \\ \mathbf{p}_{2i+1}^{k+1} &= \tfrac{1}{4}\mathbf{p}_i^k + \tfrac{3}{4}\mathbf{p}_{i+1}^k. \end{aligned} \tag{1}$$

These two equations may be rewritten in matrix-vector form using a local subdivision operator as $\mathbf{p}^{k+1} = \mathbf{S}\mathbf{p}^k$.

The positions of the vertices at level $k+1$ can be expressed in matrix form as $\mathbf{P}^{k+1} = \mathbf{S}_k^{k+1}\mathbf{P}^k$. The global subdivision operator $\mathbf{S}_k^{k+1}$, which defines level $k+1$ in terms of level k, may be represented as a rectangular (and usually sparse) matrix. The entries and size of the global subdivision operator change at each level.

In another well-known subdivision scheme [21] the curve *interpolates* rather than approximates its control points. In the limit, this subdivision curve is equivalent to a cubic B-spline (Chaikin's scheme has been shown to be equivalent to a quadratic B-spline).

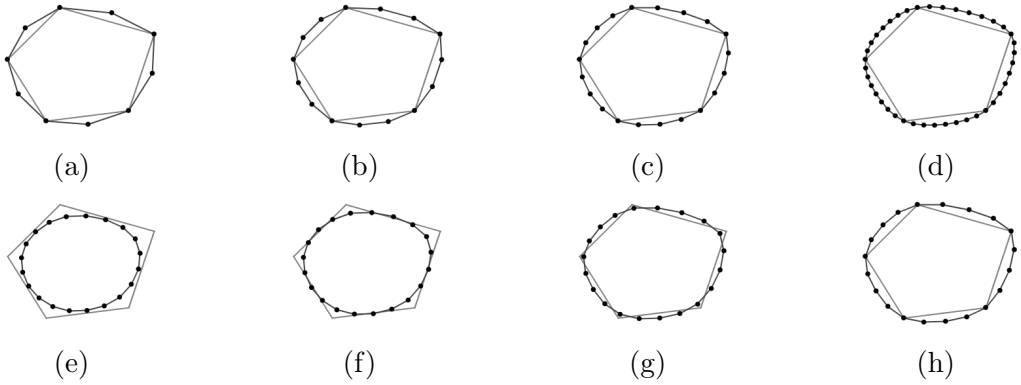


Fig. 3. Maillot-Stam[32] general subdivision curve. (a)(b)(c) Control polygon and curve after one subdivision and subdividing each control edge into 2, 3, and 4 pieces, respectively. (d) after 3 subdivisions, dividing each edge into two pieces. (e)-(h) Control polygon and curve after two subdivisions with varying degree of control point interpolation. From left to right, $\alpha = 0.0, 0.3, 0.7, 1.0$.

Recently, Maillot and Stam [32] introduced a unified subdivision curve (and surface) scheme (Figure 3) that incorporates approximating and interpolating models into one framework. This general model also allows an edge to be subdivided into any number of pieces, creating the ability to tie the number of contour points of a Snake more closely and naturally to the pixel resolution of the image. The original control points are retained at each subdivision level - a very important feature if the control points of a Snake are specified by the user as landmark constraints. They introduced a new parameter α to allow the curve to be pushed back towards the control points, interpolating the control points if desired. This feature allows the user to initialize a Snake that is very close to the target object shape and to utilize the user-entered control points as Snake constraint points. We use Maillot's and Stam's algorithm for SketchSnakes. We also set $\alpha = 1.0$ to create an control point interpolating curve. Other uses of the parameter α are the subject of future research.

The first step in the subdivision process is to linearly subdivide each segment, defined as in the previous section by a pair of vertices $\mathbf{p}_i^k$ and $\mathbf{p}_{i+1}^k$, into d pieces [32]:

$$\begin{aligned} \mathbf{p}_{di}^{k+1} &= \mathbf{p}_i^k \\ \\ \mathbf{p}_{di+j}^{k+1} &= \tfrac{d-j}{d}\mathbf{p}_i^k + \tfrac{j}{d}\mathbf{p}_{i+1}^k,\ 0 < j < d \end{aligned} \tag{2}$$

A smoothing step is then performed that modifies the vertices $\mathbf{p}_i^{k+1}$:

$$\mathbf{p}_i^{k+1} = \frac{1}{d^2}\left(d\mathbf{p}_i^{k+1} + \sum_{j=1}^{j<d} (d-j)(\mathbf{p}_{i-j}^{k+1} + \mathbf{p}_{i+j}^{k+1}) \right) \tag{3}$$

In the limit this process generates B-spline curves of degree $2m+1$ if the smoothing is applied

m times. Finally, a *push-back* of these new vertices is performed to offset shrinking effects due to smoothing, where the degree of push-back is controlled by parameter α :

$$\Delta_i = \alpha(\mathbf{p}_i^k - \mathbf{p}_{di}^{k+1})$$

$$\mathbf{p}_{di}^{k+1} = \mathbf{p}_{di}^{k+1} + \Delta_i \tag{4}$$

$$\mathbf{p}_{di+j}^{k+1} = p_{di+j}^{k+1} + \tfrac{d-j}{d}\Delta_i + \tfrac{j}{d}\Delta_{i+1},\ 0 < j < d$$

To avoid side effects caused by using neighbor vertex positions which have already been processed, all evaluations are done in "parallel" by using an intermediate array to store vertex positions. The parameter α controls the transition from approximation to interpolation. When $\alpha = 0$ there is no push-back step and the subdivision scheme produces uniform B-splines in the limit. On the other hand, when $\alpha = 1$ the scheme is interpolating. Figure 3e-h shows the influence of α on the subdivided control vertices. When the degree p is greater than three, the smoothing and push-back steps are repeated $\frac{p-1}{2}$ times. In particular, when $p = 3$ and $d = 2$, only one smoothing and one push-back step are performed.

Similar to the interpolating curve defined in [21], a "tension" parameter may also be incorporated into the subdivision scheme. This parameter controls the "tightness" of the curve (i.e. how closely it follows the control polygon) and can be useful to accurately initialize a Snake in objects with flat "end-cap" regions.

3.3 Constructing a Subdivision-Curve Snake

To construct a Snake using a subdivision curve, we use the vertices of the coarsest level control polygon of the curve (the control points) as the degrees of freedom (d.o.f.), and the finest level vertices as "sensors". As mentioned in the previous section, the Maillot-Stam subdivision curve allows for subdivision of a Snake edge into any number of pieces. This property can be used to ensure that there is roughly one sensor point for each object edge pixel, making maximal use of all available image feature information.

External image forces are computed at these sensor points and then distributed, using weights derived from the original subdivision rules, among the control points (Figure 4) (i.e. a control point that is closer to a particular sensor point will receive proportionately more of the computed force than a control point further away). The Snake is then formulated numerically using either the simple explicit scheme [12] or using the well-known and numerically more robust semi-implicit formulation in [3] [3] and the positions of the control points are updated. New external forces are calculated and the process is repeated until an accurate solution is reached.

[3] SketchSnakes uses the original finite difference formulation.

(a) (b)

Fig. 4. Example of external force computation at "sensor" points along the Snake (light gray curve) pulling the Snake towards the object edge (black contour). If a sensor point is within a user-defined range of an object edge point, a force proportional to the distance between the two points is computed (light gray line segments). This force is then divided proportionately, using weights derived from the subdivision rules, and added to the closest control points (black circles). The resulting normalized external force at the control points (thick dark gray lines) pulls the control points towards the object boundary. In (b) the user-defined range has been increased resulting in more activated sensor points.

To distribute forces from sensor points at the finest level of the subdivision curve to the control points [33], we use the transpose of the global subdivision operator $(\mathbf{S}_k^{k+1})^\top \mathbf{F}^{k+1} = \mathbf{F}^k$, where $\mathbf{F}$ is the generalized external force vector. This process can also be carried out explicitly for each sensor point using the transpose of the *local* subdivision operator.

4 SketchSnakes: A Subdivision Curve Snake Using a Sketch-Line Initialization Process

A critical and often overlooked phase of the interactive shape-model based segmentation process is the initialization of the model. Parametric Snakes typically minimize energy using local optimization techniques. Consequently, efficient convergence to the desired segmentation result is heavily dependent upon careful initial Snake placement. Since their inception, many different initialization methods and Snakes formulations have been proposed. The most common initialization methods include drawing a rough polygonal contour around the target object, or using the mouse to specify points on the object boundary and using these points to construct a polygon. This polygon, which can be open or closed [4], is then converted to a Snake. Other methods specify the position (and optionally, radius and orientation) of a circle or ellipse which is then converted to the initial Snake [5].

Each of these initialization methods has advantages and disadvantages. Drawing a rough contour is simple and does not require much user concentration. However, the target object boundary is often close to neighboring and/or spurious edges which attract the Snake, forcing the user to make corrections by performing tedious pushing/pulling operations while the Snake is deforming. In addition, the act of tracing around an object (especially a fairly complex-shaped object) becomes tedious, fatiguing and slow if the user wants to create an accurate initial boundary in order to minimize this steering and editing phase. Clicking points to create an initially polygonal Snake is also problematic as the polygonal approximation to

[4] SketchSnakes support both open and closed Snakes. In this paper, we focus on applications requiring closed Snakes.

[5] JESS provides the ability to initialize a Snake using any of these methods.

the object boundary shape can be quite coarse, again causing the Snake to snap to incorrect edges.

As mentioned in Section 2.2, using a flow-type Snake and initializing a small circular Snake can be very effective for highly-automated segmentation of complex-shaped objects, assuming the image map is homogeneous and relatively noise free. If it is not the user is forced to initialize and run additional seed Snakes, draw barriers to prevent leakage, or drag/force the flowing active contour into narrow cavities.

4.1 Sketch Lines: A Simple, Fast, Accurate Initialization Technique

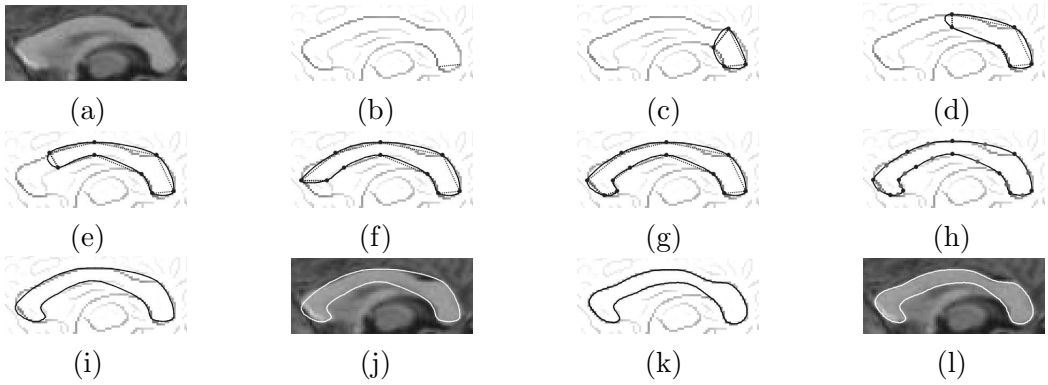


Fig. 5. SketchSnakes segmentation of the corpus callosum from a mid-sagittal MR image (a). (b)-(g) Six short sketch lines are drawn across the width of the corpus callosum in an edge-detected image. (h) A user-defined number of additional control points are automatically added between the user-defined control points. (i)(j) The initial SketchSnake shape is highly similar to the shape of the corpus callosum. (k)(l) The resulting segmentation after snapping to the corpus callosum boundary.

User fatigue is one of the most important considerations in any interactive design and analysis task. For this reason, sketching actions are being actively researched for many of these tasks in an effort to reduce fatigue and to "amplify" user input action and thereby maximize productivity. In the context of interactive segmentation, the idea behind sketching is to allow the user to provide a good initialization for the Snake - which minimizes subsequent steering and editing - with a low degree of concentration and simple mouse movements (except perhaps in noisy regions). Our sketch-line initialization process [15] is a simple but effective technique that realizes this idea. It does not require tedious tracing actions, minimizes fatigue, can be performed with a minimum of concentration and is very flexible.

Our new initialization technique also lends itself well to a pen input device. Short sketch lines are quick, easy and comfortable to draw directly on the screen, with minimal occlusion of the object by the user's hand (a common side effect of tracing actions using the pen). In this paper, we explore the use of a tablet computer. Tablet laptop computers (tablet PCs) have become extremely popular for a wide range of applications and consequently their price is now comparable to a tradition laptop computer. A pen is a more efficient, natural and precise input device for drawing actions than a mouse.

The user begins the initialization process by sketching a few short lines across the width of the target object (Figure 5b-g). Drawing these lines roughly perpendicular to the object boundary creates a contour that is approximately locally aligned with the boundary (Figure 5i,j). If the object boundary exhibits strong edges near the sketch line, the user need not sketch the line carefully - the algorithm will automatically search for the strongest (or user-thresholded) edge close to and perpendicular to the line. However, we have observed through repeated experiments that the cursor can be positioned precisely and quickly using the pen. In almost all cases, the initial Snake can be constructed in seconds with a few simple pen strokes. The number of sketch lines required is dependent upon the shape of the object and the desired degree of accuracy of the initial contour. The user typically performs a few trials, with different numbers and positions of sketch lines, until a good result is obtained (Figure 5k,l) . This pattern of sketch lines can then be used repeatedly for this type of object in subsequent images.

As each line is drawn, the algorithm uses it to automatically update the control polygon and dynamically displays the resulting subdivision curve. In this way, the user is given immediate visual feedback of the Snake construction and can observe the accuracy of the initialization as it progresses. We have found that there are obvious dents and/or bumps in the boundary or other landmark points that are good candidate positions for sketch line placement. For example, in the chest x-ray (Figure 1) we have placed the lines in these types of critical locations and the initial Snake contour is extremely close to the lung boundary. The initialization procedure acts as almost a pre-segmentation.

One point should be noted about the subdivision curve construction based on the sketch lines. Due to the nature of the subdivision algorithm, the initial contour becomes more accurate (closer in shape to the target object) as additional sketch lines (and hence control points) are added. After the first two or three sketch lines, the subdivision curve may appear a small distance away in some parts of the target object boundary (for example, Figures 1a,b versus Figures 1c,d). Consequently, there is a very slight learning curve for the user to adapt to this effect.

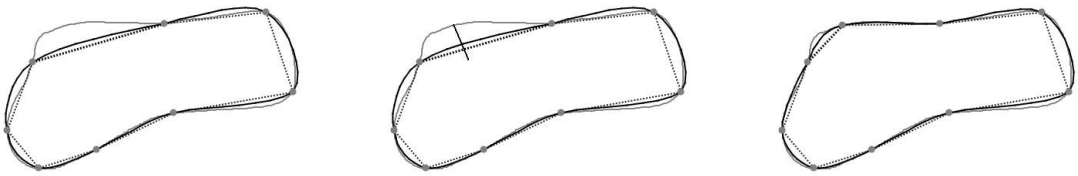
4.2 Editing and Steering

Editing a segmentation result typically implies making corrections to the segmentation after the algorithm has run to completion. Steering, on the other hand, implies guiding the segmentation process toward the correct result while the algorithm is running. The line between editing and steering is blurred when considering the segmentation work-flow using SketchSnakes.

Once the Snake has been initialized, the user presses a key (for example, the space-bar key if the user is using the mouse or a tablet button if the user is using the pen) to begin the fitting step and the Snake *snaps* to the object boundary. We terminate the fitting step after a (user-adjustable) number of iterations has occurred. This snapping to the boundary is essentially instantaneous for all objects we have segmented. Other stopping criteria may be employed, such as when the average distance traveled by all Snake points from one iteration

to the next falls below a threshold. However, because the initialization process creates a Snake very close to its final position, only one fitting step with a small number of fitting iterations is typically required to generate an accurate segmentation. Nevertheless, the user may repeat the fitting step by hitting the space-bar key or tablet button.

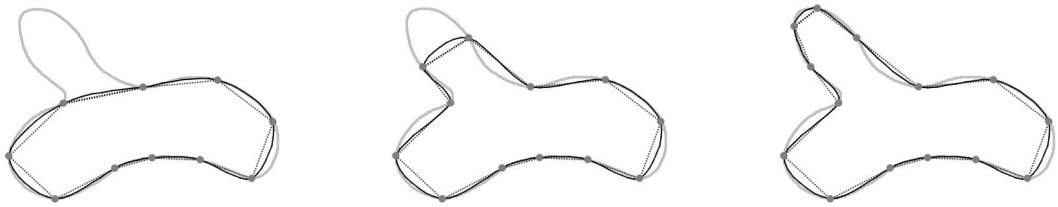
Before, between, or after these fitting steps, SketchSnakes can be precisely and intuitively controlled using various geometric editing actions. The initialization, fitting, editing, steering, and zooming are all performed within a single seamless process, using only simple pen actions [6]. The user is not forced to constantly switch modes or select actions items from a menu or panel. Examples of the main SketchSnakes editing actions are listed below:



(a) (b) (c)

Fig. 6. (a)(b) Adding a control point to an initial SketchSnake by drawing a line that cuts the user-selected control polygon edge. (c) The first point of the cut line is used as the position of the new control point (c).

- For complex shapes the user may need to add to the initial contour. A seamless and effective way of extending the initial contour is to draw a short line segment (a "cut"- line) across an existing edge of the current control polygon. This intuitive action automatically cuts (subdivides) the edge and creates a new control point (Figure 6). The first point of the cut-line is used as the final position of the new control point. This combined cutting-positioning action allows the user to create protrusions in the initial Snake with a single stroke of the pen. Of course, the new control point may also be selected and dragged to fine-tune its position, if desired. The control polygon and curve are automatically updated and drawn as the control point is dragged.



(a) (b) (c)

Fig. 7. (a) The user places the pen near a control polygon edge and taps the screen. This edge is highlighted and becomes the "active" edge. (b)(c) The user sketches additional lines to extend the initial SketchSnake.

- If the pen is placed on or near a control polygon edge and the screen tapped once, it becomes the *active* edge and is highlighted. Consequently, when a new line is sketched by the user, it is connected to the control polygon via this active edge. The previous cutting action and this extending action allow a user to quickly and easily outline complex shapes

[6] Alternatively, the user may use a mouse.

that may have significant protrusions or bumps (Figure 7) while keeping the initialization process flexible and intuitive.

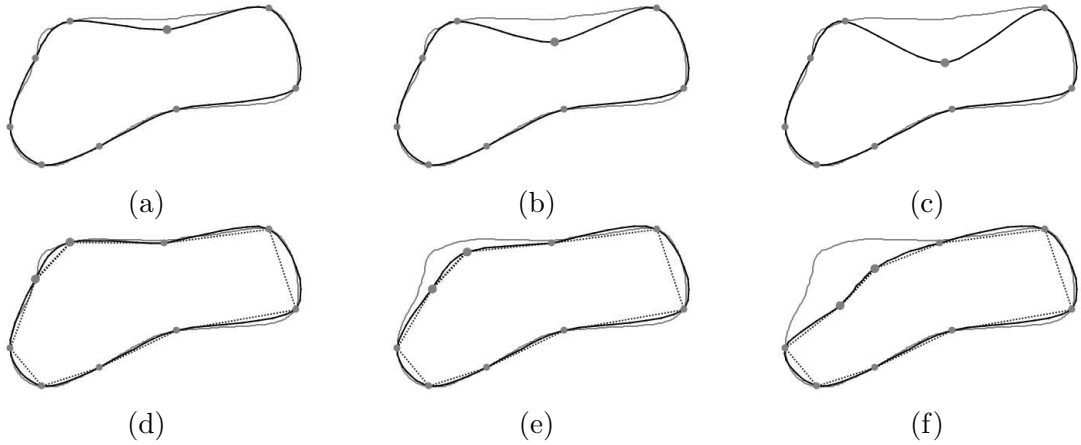


Fig. 8. The SketchSnake can be edited by selecting and dragging: a control point (a)-(c) or a control edge (d)-(f).

- The use of subdivision curves as the underlying representation for a Snake provides a robust and flexible foundation for editing and steering the model. Subdivision curves provide intuitive, precise geometric editing capabilities. Pressing the pen to the screen overtop of a control point or control polygon edge selects the point or edge. The selected point (Figure 8a-c) or edge Figure 8e-f) is dragged with the pen or mouse and the real-time subdivision process automatically updates the smooth interpolating curve. Due to the nature of the initialization process, a slight "nudge" (repositioning by dragging) of one or two control points is often all that is required to correct the object segmentation in noisy image regions.

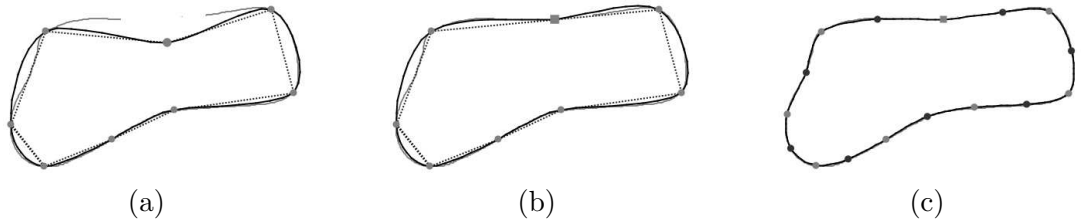


Fig. 9. (a) A control point (or control polygon edge) can be selected and the pen held to the screen for a half-second. (b) This "pinning" action freezes the control point (indicated by its square shape). The point can be repositioned before or after it is frozen. (c) The SketchSnake is now constrained to pass exactly through the frozen control point. Frozen points can be added near noisy object regions as the user constructs the SketchSnake or after the Snake has snapped to the object boundary.

- For regions of the object with missing or noisy edges the user can, if desired, quickly and simply provide the algorithm with additional information. The pen is positioned over a control point and the user presses down and holds for half a second (or so) on the screen. This intuitive "pinning" action generates a hard constraint at the control point (Figure 9b). This control point is frozen and the Snake is constrained to pass through it[14] [7] (Figure 9c). Alternatively the user may press and hold on a control polygon edge of the Snake to freeze both end-points. The user typically takes more care when initially

[7] Frozen control points can still be repositioned at any time with the pen or mouse or unpinned.

positioning these points, ensuring that the control points are placed where the user has interpreted the boundary location. Thus, optimal use of human recognition capabilities are exploited. The user is able to easily and dynamically transfer knowledge of the object boundary to the algorithm *as the Snake is constructed*, maximizing the chance of Snake segmentation success and thereby minimizing or eliminating post user editing/interacting phase. This philosophy is similar to Live Wire [7], where the user-controlled cursor speed is used to indicate weak and strong boundaries.

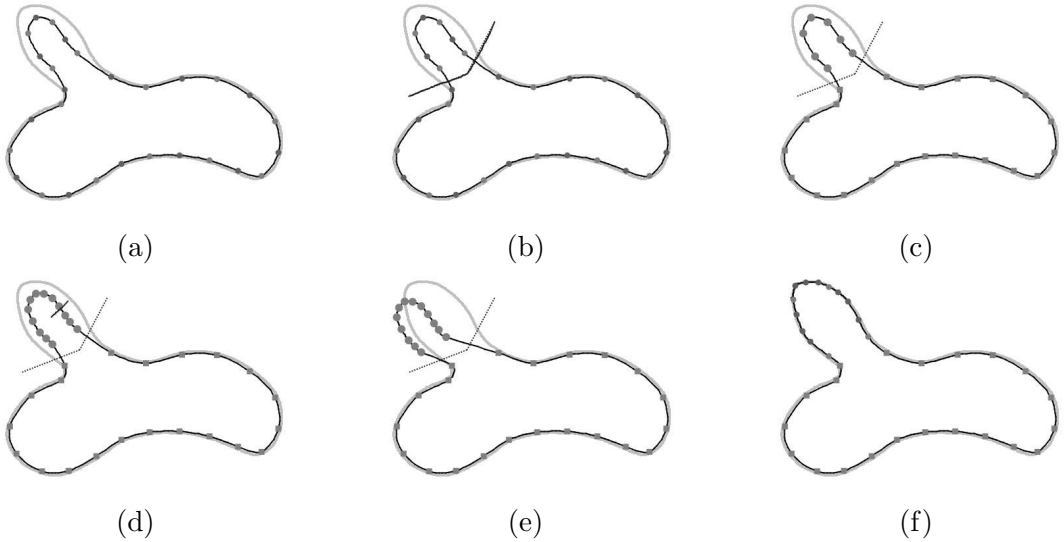


Fig. 10. Regions of a SketchSnake can be selected and edited using a region-selection gesture. (b) The user draws a rough curve, consisting of one to three approximately linear segments, that cuts two control polygon edges of the SketchSnake. (c) This gesture is recognized by the system and the curve is replaced by one to three line segments, dividing the SketchSnake into two regions. (d) The user can select or freeze all control points within either region by selecting/freezing a single control point. Furthermore, a cut-line drawn across any control polygon edge will subdivide the entire region. (e) The entire selected region can be dragged with the pen or (f) fitted to the target object without disturbing the remainder of the SketchSnake.

- We have developed a simple but effective gesture recognition scheme that allows the user to selects regions of the SketchSnake with simple pen strokes. The user draws a rough curve, which consists of one to three straight parts, across two or more edges of the SketchSnake control polygon (Figure 10b). This gesture is recognized and the curve is instantly replaced with one to three line segments, respectively (Figure 10c). The line segments divide the SketchSnake into two regions. If the user selects a control point within either region, all of the region control points are selected (Figure 10c). If the user freezes a control point, all region control points are frozen. Furthermore, if the user draws another line that cuts across any edge of the control polygon within a region, all of the edges are automatically subdivided and new control points are added (Figure 10d). The user may drag the entire region (Figure 10e) or freeze the opposing region and fit only the selected region (Figure 10f). This region editing control allows the user to focus on noisy areas of the target object without disturbing the correctly segmented area. Selecting one of the region delineating line segments deactivates the segments and region processing.
- A zoom gesture has also been developed to seamlessly incorporate zooming into the segmentation work-flow. The user draws a simple v-shaped curve (Figure 11) and this gesture

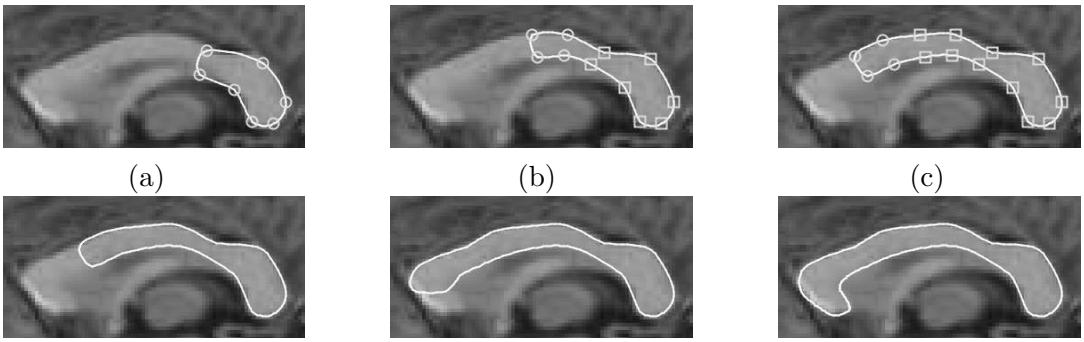
(a) (b)

Fig. 11. Zooming is seamlessly integrated into the SketchSnakes work-flow through the use of a zoom gesture. (a) The user draws an upside-down or right-side up v-shaped curve to indicate a zoom-in or zoom-out action. (b) The point of the v-shape is used as the center of the zoom.

is recognized by the SketchSnakes system as a zooming action. If the v-shape is upside down (Figure 11) a zoom-in action is activated with the center of zoom at the point of the v-shape, conversely a right-side up v-shape curve indicates a zoom-out action.

Finally, one of the defining features of parametric Snakes models has always been the ability to use input-device derived forces to dynamically push or pull on a Snake (i.e. as it is deforming). While not as precise as geometric editing, this force-based editing paradigm can occasionally be useful for moving larger regions of a Snake by pushing/pulling on a single Snake point - the neighboring points will be dragged along due to the internal forces. In our SketchSnakes system, force-based editing is initiated and terminated by depressing a button on the Snakes control panel.

4.2.1 Steering with Progressive Snapping



(a) (b) (c)

(d) (e) (f)

Fig. 12. A SketchSnake fitted to a corpus callosum using "progressive" snapping mode. (a)-(c) As each sketch line is drawn, internal control points are automatically added and the active region of the SketchSnake (circular control points) snaps to the boundary, while the previously fitted region (square control points) are automatically frozen. In (b) the user has manually corrected the ill-fitting region near the fornix before proceeding. (d)-(f) Control points have been turned off and the user draws two more sketch lines to complete the segmentation.

Another steering mechanism supported by SketchSnakes is a "progressive" fitting mode (Figure 12). In this mode the Snake automatically snaps to the object boundary after each sketch line has been drawn [8]. This mode is similar to the LiveWire model of segmentation (i.e. as

[8] A minimum of two sketch lines are required to create a closed SketchSnake and activate the fitting process

the mouse is moved along the object boundary, the LiveWire algorithm continuously computes and display the least-cost path). However, unlike LiveWire the user may edit the Snake at any time and location along the Snake (Figure 12b). In some segmentation experiments, progressive snapping is even more efficient than the normal SketchSnake "initialization plus snapping" mode and is under further investigation.

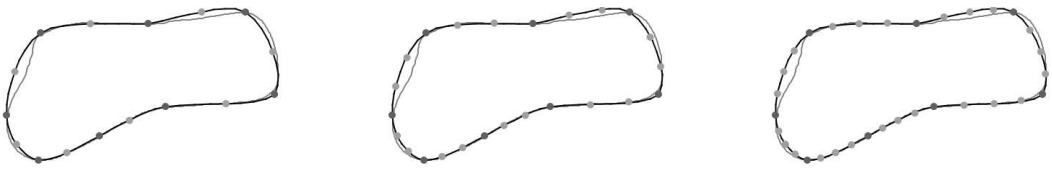
4.3 SketchSnakes Visual User Interface

As with the traditional Snakes, various SketchSnakes parameters can be controlled to improve efficiency and/or robustness to noise. Over the years one of the main criticisms of Snakes has been the difficulty in finding suitable parameters for a specific target object and image modality. This criticism can also be leveled at Adobe Photoshop's Magnetic Lasso. Although default parameter settings for Magnetic Lasso are often adequate, for efficient segmentation parameter adjustments are needed to control the search width, frequency of automatic seed dropping, and edge contrast. SketchSnakes have similar parameters plus additional stiffness and stretchiness controls. However, due to the nature of SketchSnakes initialization these additional controls very rarely need adjusting. To make parameter setting more intuitive, we have developed a visual interface which includes the following controls and displays:

(a) (b) (c)

Fig. 13. (a) Canny edge detected image. Sliders are used to control the minimum edge intensity (b) and the maximum edge intensity (c). The SketchSnake will search for edges within this range.

- *Edge intensity control.* The maximum and minimum edge intensity threshold can be set with two sliders and the resulting edge detected image is displayed in the image display window (Figure 13). The Snake will search for edge intensities within this range.
- *Sliders for stiffness/stretchiness.* The default setting for these parameters are typically adequate for almost all objects and image modalities. In fact, we have used the same settings for all the experiments in this paper. This is primarily due to the proximity of the Snake to the object boundary after initialization, making these parameters less important.
- *Automatic insertion of internal control points.* The number of control points automatically added between the user-defined control points can be controlled with a slider. Internal control points are analogous to the seed points automatically dropped by Magnetic Lasso. The internal control point slider is activated once a SketchSnake has been created by the user and by depressing a button on the control panel. The points are displayed on the Snake in the image window so that the user can visually ascertain if there are sufficient control points to represent the target object shape (Figure 14). The default number of internal control points inserted between user-defined points is set to one. Typically the



(a) (b) (c)

Fig. 14. A slider is used to control the number of internal control points (light gray shaded) automatically inserted between user-defined control points (black) resulting from sketch lines. (a) One internal control point, (b) two internal and (c) three internal points.

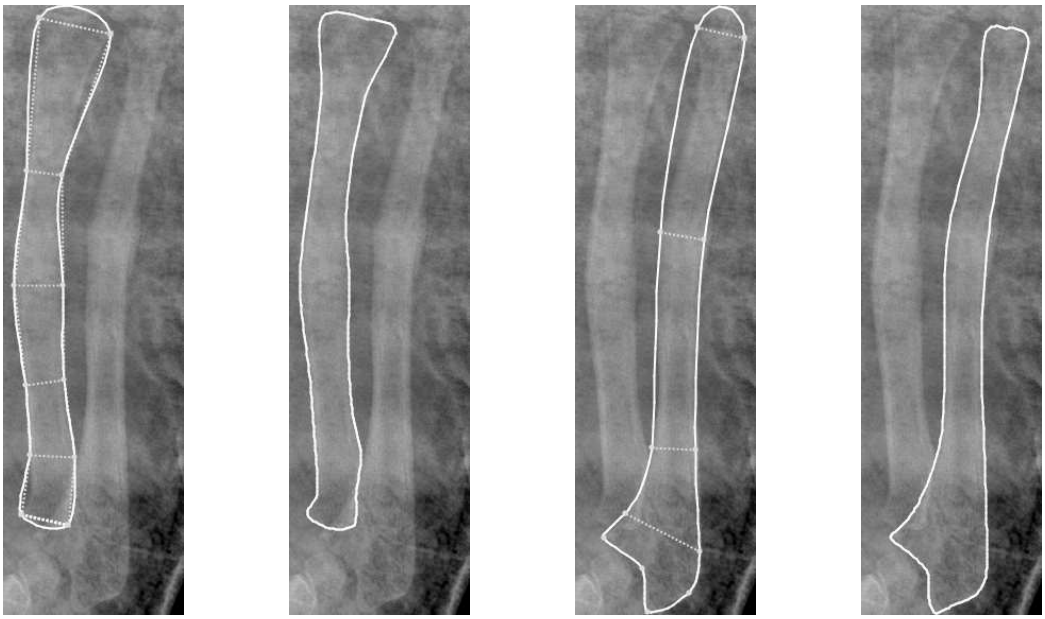
user experiments with this setting and increases the number of points if the default setting proves inadequate. A future version of SketchSnakes will allow for local control of internal points. In Figure 14, note how the internal control points do not change the shape of the subdivision curve constructed from the user sketch lines.

(a) (b) (c) (d)

Fig. 15. Two sliders are used to control edge search width (measured in pixels) and the resulting search band is displayed in the image window. (a)(b) The corpus callosum MR image. (c)(d) The edge search band at different settings surrounding the Snake in a Canny edge detected image.

- *Image edge search width.* For each sensor point, a search along its normal is carried out for a small, user-specified distance (typically only two or three pixels). If a matching edge pixel is found, a spring force is applied to attract the snake point to it. If no matching edge is found (in the case of a boundary gap or noisy edge pixels), this pixel does not contribute to the image forces. Edge search width is controlled by two sliders, one for the negative sensor point normal direction, one for the positive normal direction and the resulting search region is visualized by drawing a transparent band around the Snake in an edge detected image window (Figure 15).
- *Edge transition control.* Buttons are used activate a search for edges with either a bright-to-dark transition in the image or a dark-to-bright transition. This feature allows the Snake to ignore edges with incorrect transitions and results in improved segmentation accuracy and repeatability.

This visual interface allows a user to quickly "tune" the Snake parameters for a specific object and image modality and maximize segmentation performance. This tuning process is typically only performed once. The parameter settings can then be labeled and saved in a separate "Settings" tabbed panel. Upon startup, this tabbed panel can be selected and specific saved parameter settings appear as a radio button. Selecting this button loads the parameters. Of course, the user can change the parameters at any time and save a new set.



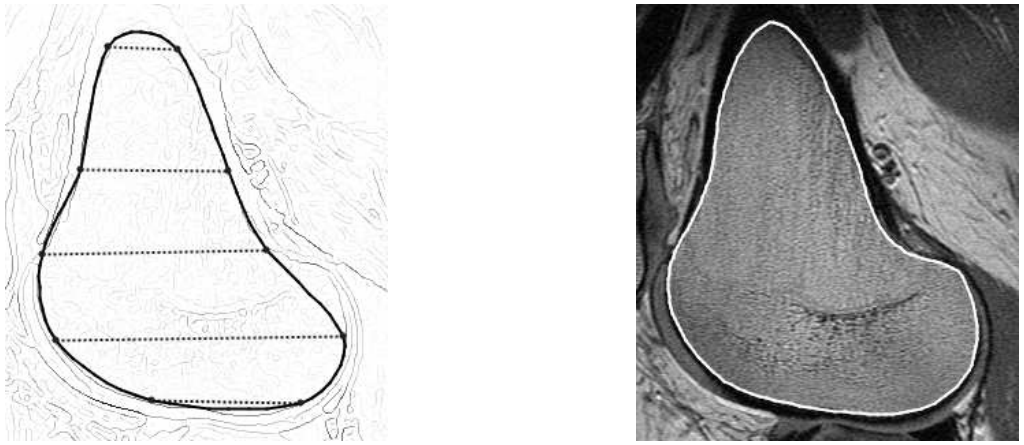
(a) (b) (c) (d)

Fig. 16. Segmenting arm bones from a noisy x-ray image. (a)(c) user input lines and initial SketchSnakes (b)(d) Segmentation results.

5 Experimental Results

We have applied SketchSnakes to various 2D medical images to demonstrate its generality, ease of use, accuracy and efficiency. In Figure 16 we show the six user sketched lines and initial SketchSnakes (Figure 16a,c) and the final segmentation for two forearm bones in a noisy x-ray image (Figure 16b,d). In Figure 16c, the active edge (the control polygon edge at the bottom) was "broken" into two pieces and a new control point inserted by drawing a cut line across the edge. For each arm bone, the sketch lines were drawn in approximately 8 seconds and 3-4 small edits were required to nudge a Snake section into place, taking another 3-4 seconds. This x-ray image is very noisy, especially where two bones overlap. There are many large gaps in the edges of the bone boundary and many spurious edges inside the bone. Since the initial Snakes are almost the same shape as the target bones, they are able to ignore edges that are not of a specific magnitude. In contrast, *Magnetic Lasso* took, on average, approximately 15-20 seconds with 3-4 seeds manually deposited and 2-3 backtracking operations required for accuracy in very noisy regions. After experimentation, we found the Magnetic Lasso parameter settings of Width = 10 pixels, Edge Contrast = 10%, Frequency = 50 maximized efficiency and accuracy. However, if a mistake was made during the long tracing action required and not corrected immediately during backtracking, the segmentation time for Magnetic Lasso is significantly increased as the user is required to select another editing tool. Similarly, when using *Snap*, it takes roughly 20 seconds to draw foreground and background strokes and generate a reasonably accurate initial segmentation. However, there are almost always highly inaccurate regions (especially near regions of high curvature) and several additional foreground strokes are needed to improve the segmentation, taking another 5 seconds. Even with these corrections, further edits are required to generate a more accurate result. Due to the inadequacies of the various boundary editing tools provided

(for example, control point repositioning) and the boundary visualization issues created by the editing tools, we were unable to use Snap to efficiently generate a highly accurate segmentation in many of the experiments involving segmentation of the arm bone .



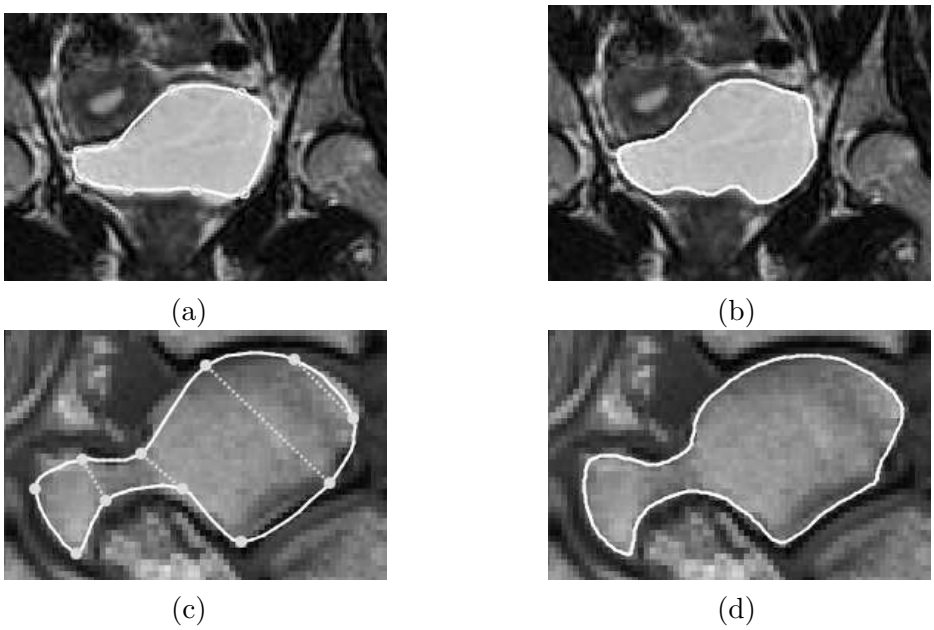
(a) (b)

Fig. 17. Segmentation of knee bone from MR image. (a) initial SketchSnake generated from 5 sketch lines in edge-detected image. (b) segmentation result.

Figures 17 demonstrates the ability to create an initial SketchSnake that is very close in shape to the target object (the knee bone in this MR image slice). The user may further improve this initial contour by "breaking" an edge near the right bulge of the knee bone. The 5 lines can be sketched in approximately 7 seconds and typically only 1 edit is required, adding another 2 seconds. Magnetic Lasso requires about 15 seconds (Width = 5 pixels, Edge Contrast = 30%, Frequency = 50) with 1 to 2 additional seeds and 1 backtracking operation, on average, to generate an accurate result. Tracing more quickly results in many more backtracking operations and a significant increase in segmentation time. Snap can generate an accurate result with only two well-placed strokes, requiring about 5 seconds. However, the region near the top requires considerable "fine-tuning", best achieved by directly manipulating spline control points. This editing action typically takes an additional 10 seconds.

In Figure 18a we have drawn 4 sketch lines and broken a sketch line at the bulge near the lower right part of the bladder in this MR image. The lines are consistently sketched in approximately 6 seconds and 2 small edits are typically required adding an additional 2 seconds to the overall segmentation (Figure 18b). Magnetic Lasso requires about 15 seconds (Width = 5 pixels, Edge Contrast = 20%, Frequency = 50) with 5 to 6 additional seeds for an accurate result. This object is difficult to segment with Magnetic Lasso due to the high curvature regions and additional edits using a separate tool in Adobe Photoshop is sometimes needed. Snap can segment the bladder using a single foreground and background stroke in about 5 seconds. However, the segmentation is not of sufficient accuracy in several regions and requires significant editing by dragging control points.

In Figure 18c we segment the talus bone. We use 5 sketch lines with an additional break of one line. On average, 7 seconds are required for line sketching and an additional 2 - 3 seconds for 2 small edits. Magnetic Lasso requires about 15-20 seconds (Width = 3 pixels, Edge Contrast = 20%, Frequency = 100) with 5 to 6 additional seeds for an accurate result. We found this

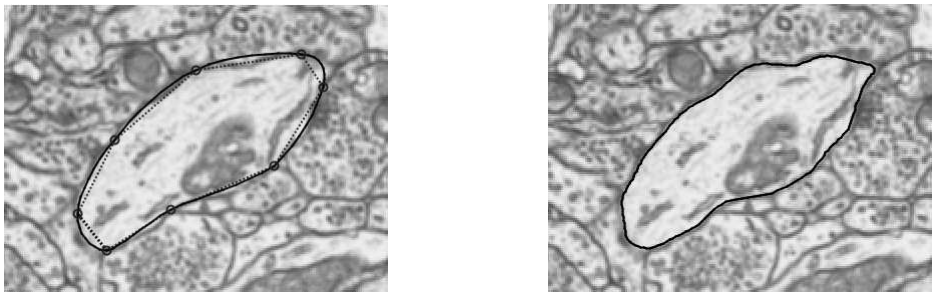


(a) (b)

(c) (d)

Fig. 18. (a)(b) Initial SketchSnake and final segmentation result for a bladder in an MR image. (c)(d) Initial SketchSnake and final segmentation result for an ankle bone (talus) in an MR image.

object also quite difficult to segment with Magnetic Lasso due to the many "corner" points. We experimented with several parameter combinations and eventually set the auto-seed frequency to it's highest value (100) to reduce the number of manually set seed points. Snap requires approximately 15-20 seconds to draw several foreground and background strokes to generate a reasonable result. However, once again a considerable amount of manual editing is required adding significantly to the overall segmentation time. In fact, we found it difficult to generate an accurate segmentation with the current editing tools included with Snap.



(a) (b)

Fig. 19. Segmenting a neuronal cell from EM image. (a) Only four sketch lines are used and the initial SketchSnake is very close in shape to the desired result (b).

In Figure 19 we segment a neuronal cell from an EM image. Note how close the initial SketchSnake is to the desired shape using only 4 sketch lines. The lines are sketched in approximately 6 seconds and occasionally 1 small edit is required. Magnetic Lasso requires about 12 seconds (Width = 3 pixels, Edge Contrast = 20%, Frequency = 51) with 1 additional seed to consistently generate an accurate result. Snap requires about 15 seconds for the initial segmentation. Additional editing of well over one minute is then required to obtain a reasonably accurate solution.

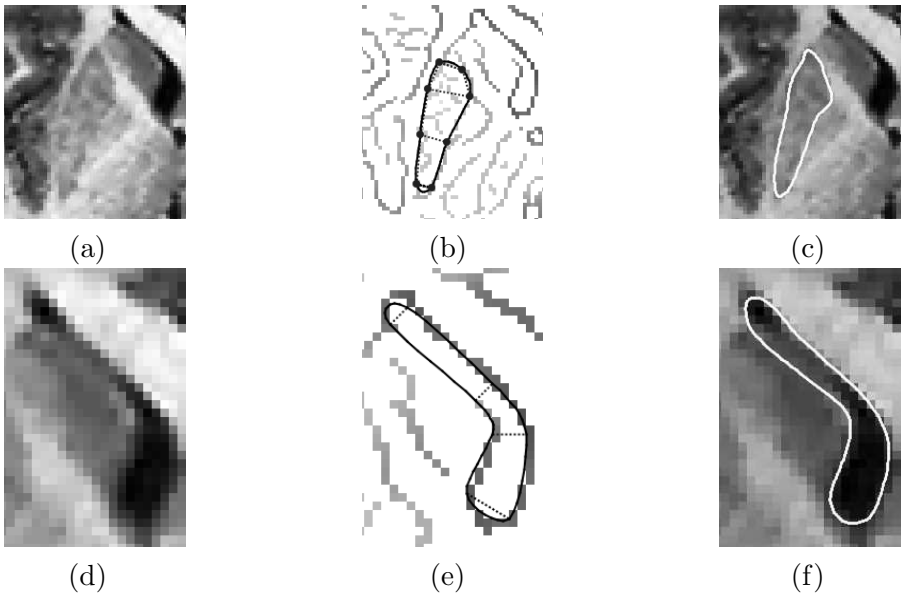


Fig. 20. Segmenting the putamen (a)-(c) and the lateral ventricle (d)-(f) from a slice of an MR brain volume image. Both of these objects are difficult to segment due to texture, noise and neighboring structures with similar intensity characteristics. (b)(e) Initial SketchSnake in edge detected image for each structure showing sketch lines. (c)(f) Final segmentation result.

We segmented the putamen and the lateral ventricle from a slice of an MR brain volume image. Both of these structures present a segmentation challenge due to noise and the similar intensity characteristics of neighboring structures. For the lateral ventricle, there is a large gap in the edge map where the ventricle meets the caudate nucleus and the ventricle is also very narrow in this region. With a few sketch lines (Figure 20e) the initial SketchSnake is also the exact required shape. The sensor points along the gap in the edge map will not find an edge to snap to and consequently they will not contribute any external force in this region. Rather, internal forces will maintain (approximately) the same initial position for the single Snake control point in this region. A similar situation occurs for the left side of the putamen that is very close to the cortical surface. For both the putamen and the lateral ventricle, one internal control point is automatically added between the Snake control points created from the sketch lines. In terms of efficiency, the four sketch lines require approximately 6 seconds for the putamen and 3 seconds for the lateral ventricle. The putamen typically requires 2 to 3 small edits requiring an additional 3 to 4 seconds while the ventricle requires 3 small nudges for an additional 3 seconds. After averaging repeated trials, Magnetic Lasso requires about 10 seconds (Width = 3 pixels, Edge Contrast = 20%, Frequency = 51) with 5 additional seed to generate an accurate result for the ventricle and about 9 seconds (Width = 3 pixels, Edge Contrast = 10%, Frequency = 51) with 5 additional seeds for the putamen. However, a great degree of concentration is required to trace both of these structures using Magnetic Lasso in order to ensure a repeatable result is obtained; the result is sensitive to slight deviations in mouse/pen positioning. Using the Snap tool, we found it very difficult to generate a reasonable initial segmentation for the putamen and this structure needed to be essentially hand-segmented. Similarly, we were unable to efficiently use Snap for the lateral ventricle and resorted to complete manual placement of spline control points.

In Figure 21 we demonstrate the ability to quickly segment multiple structures from an

(a) (b) (c)

Fig. 21. Segmenting vertebra from a MR image slice. Only two sketch lines are needed to generate the initial SketchSnake for each vertebrae.

image. In this example, only two quick sketch lines are needed to segment each vertebra (two internal control points are automatically added between each pair of control points created from the sketch lines). Because of the sharp corners of the vertebra, we have set the SketchSnake into a mode where the four control points generated from the two sketch lines are initially frozen. Figure 21 (b) shows an initial SketchSnake for the seventh vertebra. When this image is zoomed, the pen can be placed very precisely when drawing the two sketch lines as it is simple to place your hand in a position that does not occlude the corner points. SketchSnakes requires approximately 50 seconds to draw sketch lines and fit to all of the vertebra. However, there are typically 4 to 5 control points that require manual repositioning adding an additional 10 to 12 seconds. Magnetic Lasso (Width = 3 pixels, Edge Contrast = 20%, Frequency = 100) requires an average of approximately 78 seconds to segment all the vertebra (ignoring time taken to save the segmentation results of each vertebra). We set the frequency of seed dropping to the most automatic setting as this maximized efficiency for these fairly simple square-shaped objects. Similar to the SketchSnakes case, four seeds are manually dropped at each corner point of a vertebra. The user must take some care when tracing these structures as tracing too quickly leads to errors and subsequent manual editing. It is also difficult to see the corner points as you are tracing due to occlusion from the user's own hand.

Snap can segment each vertebra often using a single foreground and background stroke in about 5 to 8 seconds. However, the segmentation is almost always not sufficiently accurate and requires an additional 20 to 30 seconds of editing for each vertebra.

Finally, in Figure 22 we demonstrate the generality of SketchSnakes by segmenting the skin and all of the finger and wrist bones from this x-ray image of a hand. The initial SketchSnake for these elongated structures is typically very close in shape to the target object and therefore rarely requires an edit. To construct the initial SketchSnake for the skin, for each finger we use a single pen stroke to cut the top sketch line (Figure 22a) and position the inserted control point at the point between two fingers. We then select the control polygon edge at the base of each finger and add three additional sketch lines (Figure 22b). The resulting initial SketchSnake for the skin is very close in shape to the skin shape (Figure 22c) and the SketchSnake correctly segments the skin with little or no editing. In Figure 22e,f we show the final segmentation of the skin and all the bones in the hand. Each finger bone requires

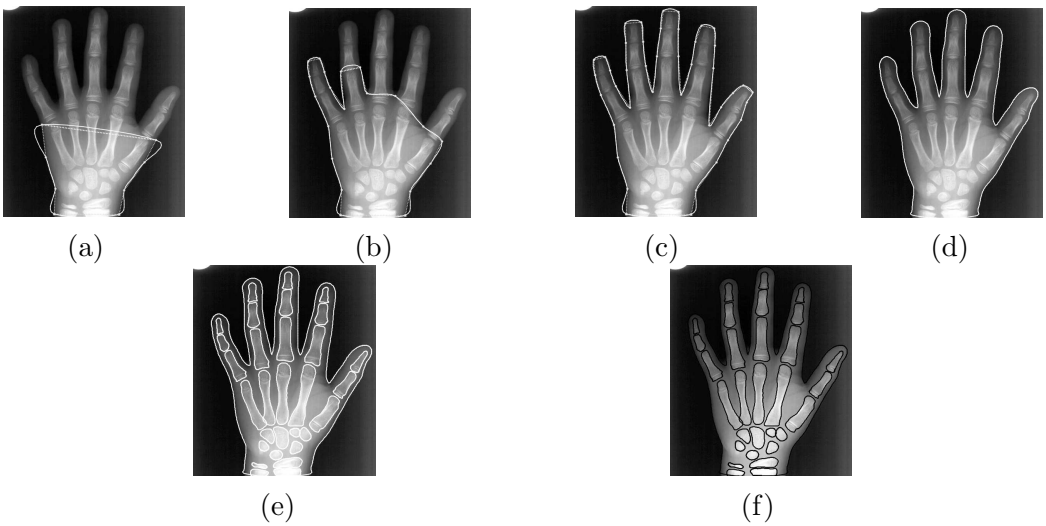


Fig. 22. Segmenting multiple structures from an x-ray image of a hand (the skin, finger and wrist bones).(a)-(d) Generation of the initial SketchSnake for the skin by continuously breaking and positioning a sketch line using a single pen stroke. (e)(f) Each bone requires 2 - 4 sketch lines (depending on its length) and occasionally a small edit is required.

only a few seconds to segment using 2 to 4 sketch lines and minimal editing. Both Magnetic Lasso and Snap require significantly more time and editing than SketchSnakes to generate an accurate result.

6 Discussion

In [34], Olabarriaga and Smeulders suggested several guiding design principles should be followed to produce efficient interactive segmentation methods that generate accurate and repeatable results:

- *Integrating the computational and user interaction components of the segmentation system into one process.* This guiding principle is essentially referring to the role of interactive steering and we interpret it as providing the user with sufficient control of the entire segmentation process such that manual editing in the end is not necessary.
- *Using pictorial input to the computational part of the system;* LiveWire, Snakes, and GraphCut methods all adhere to this principle.
- *Allowing the user to initialize the segmentation process with key information which will lead the method to an accurate result more quickly.* This was the primary design goal behind the sketch line initialization process. LiveWire and GraphCut methods also provide key information at initialization.
- *Allowing the user to control and steer the method throughout the whole process to generate accurate results.* We claim that SketchSnakes provides more flexible and precise steering and editing control over than does LiveWire and GraphCut. This issue is discussed in Section 6.4.
- *Allowing the user to intuitively predict the impact of the interactions on the segmentation result.* In our opinion, boundary-based techniques such as Snakes and LiveWire adhere to

this principle better than region-oriented techniques such as GraphCut. With boundary-based techniques one is explicitly manipulating and repositioning the boundary shape model to lay on top of the target object boundary.
- *Providing visual feedback of the effect of the interaction in real-time.* All pen/mouse-based interactions and fitting steps update the Snake in real-time.
- *Adding intelligent behavior to elevate the abstraction level of the interaction.* We claim that *shape model* based techniques such as Snakes, including SketchSnakes (see [15]), have much more potential for adding intelligent behavior than do the *pixel*-based LiveWire and GraphCut.

We have attempted to design SketchSnakes with these guiding principles in mind. In the following sections, we provide a discussion and validation of our design with respect to the key criteria for evaluating segmentation techniques: accuracy, efficiency, and repeatability. Since control over editing and steering also impacts each of these criterion, we include a separate section on this topic.

6.1 Accuracy

Accuracy can be defined as the degree to which the delineation of the target object agrees with the ground truth. The ground truth can be measured subjectively by a human expert, or objectively by comparing the segmentation result against the ground truth using various metrics. Examples of these metrics include the difference in the area or the average distance between points on the segmentation result and closest points on the ground truth boundary.

Accuracy as an evaluation criterion makes more sense in the context of fully automatic segmentation algorithms [34]. In interactive methods, the user may improve the accuracy to the desired degree unless the method does not provide sufficient user control. In Section 4.2, we demonstrated that SketchSnakes provides both local and semi-global subdivision, steering and editing facilities, providing the user with complete and precise control over the final contour shape. All of these accuracy control mechanisms are available at any time during the segmentation process. Of course, these mechanisms may impact the efficiency of the algorithm although we have attempted to design them to fit seamlessly within the segmentation work-flow.

Magnetic Lasso can also be made accurate by dropping more seed points as the user traces around the object. However, if additional points are required in some section of the trace to improve accuracy, the user must enter a separate editing mode. Furthermore, there is no control of the trace between seed points - the algorithm is always "on". To create a precisely traced section, the user must drop seed points very close together.

We have performed a series of experiments comparing the accuracy of LiveWire with SketchSnakes for twenty-four corpus callosum (CC) mid-sagittal MR images (Table 2). Note that SketchSnakes achieves sub-pixel accuracy using only 24 control points to represent the subdivision curve whereas LiveWire uses approximately 180 pixels. Furthermore, the smooth subdivision curve provides an analytical description over the whole curve.

Controlling the accuracy of region painting algorithms, at least for medical image segmentation, is problematic. The user is forced to paint over smaller sections of the foreground region with the hope that the correct region is obtained. For this reason, other boundary-based tools [28] are necessary to more precisely control accuracy. For example, Snap provides several accuracy control mechanisms. Nevertheless, which mechanism to use in a given situation is difficult to determine and the control and interface of the mechanisms is, in our experience, often inadequate.

6.2 Repeatability

Repeatability may be measured as the difference between the segmentation result over different segmentation sessions with the same target object; the smaller the difference, the higher is the degree of repeatability. As is the case with with accuracy, repeatability is more of an issue in automatic segmentation. In interactive segmentation, the user can edit or steer the initial result to the desired degree of accuracy. The difference in the segmentation results can be due to the difference in the operation of the segmentation tool (for example, the user clicks the mouse at different positions in the image in different sessions) or a difference in user judgement of the correct object delineation. Nothing can be done about the latter but a good segmentation method must minimize the effects of the former. This can generally be achieved when the segmentation result is generated mostly by the computational part of the method.

SketchSnakes supports repeatability in several ways. Firstly, the subdivision-curve based Snake is a low degree of freedom model, equivalent to a B-spline curve. Furthermore, control is provided to the user allowing them to set the minimum number of degrees of freedom necessary for achieving the desired level of accuracy. Fewer degrees of freedom translates into better noise insensitivity, thereby improving repeatability. Secondly, the sketch-line initialization process generates an initial contour very close to the shape of the target object. Through a few quick experiments and with the aid of the visual interface, a user can discover the optimal number, position, and input order of the sketch lines and internal control points for a specific object. Adhering to this initialization protocol ensures an accurate result and supports respeatability.

(a) (b)

Fig. 23. SketchSnake initialization in MR image of CC. (a) Only 4 sketch lines are used instead of the optimal number of six sketch lines. (b) The SketchSnake is still able to snap to the correct boundary with little or no editing.

Finally, SketchSnakes are fairly robust to the number and location of sketch lines. To demonstrate this robustness, we have initialized a Snake using different locations and different number of sketch lines for several corpus callosum images (Figure 23). The resulting segmentation accuracy is very consistent. We performed six trials for five different corpus callosum

	Trial					
Case no.	1 mn : mx	2 mn : mx	3 mn : mx	4 mn: mx	5 mn : mx	6 mn : mx
11	0.77 : 2.2	0.72 : 2.2	0.72 : 2.0	0.75 : 2.2	0.76 : 2.2	0.74 : 2.2
12	0.67 : 2.0	0.58 : 2.0	0.63 : 2.0	0.65 : 2.0	0.61 : 2.2	0.67 : 2.0
13	0.54 : 2.0	0.59 : 2.2	0.62 : 2.0	0.59 : 2.0	0.67 : 2.0	0.68 : 2.0
14	0.68 : 2.0	0.69 : 2.0	0.70 : 2.2	0.70 : 1.4	0.74 : 1.4	0.70 : 1.4
15	0.72 : 2.0	0.71 : 2.0	0.75 : 2.2	0.74 : 2.0	0.78 : 1.4	0.75 : 2.0

Table 1
Table demonstrating the insensitivity of SketchSnakes with respect to sketch line placement. The mean (mn) and maximum (mx) errors are shown (i.e. distance between the extracted and expert-segmented CC boundary, in pixels) for several trials using five CC images. Trial 1 used four sketch lines to initialize the SketchSnake, trial 2 used five sketch lines, while trials 3-6 used six sketch lines drawn in different (and slightly sub-optimal) positions along the CC. For each trial, the number of SketchSnake control points (user-defined and automatically added internal control points) was approximately the same.

cases (Table 1). In the first two trials we used only four and five sketch lines, respectively. However, we increased the number of internal control points so that the total number of control points is approximately equal for all trials. In the last four trials used six sketch lines (through experiment we have determined that six sketch lines is optimal for the CC) and these lines were purposely chosen to be in slightly different and sub-optimal locations compared to the accuracy experiments outlined in the previous section. A slightly increased number of user edits were required to achieve good segmentation accuracy for the four sketch line and five sketch line cases (which is not unexpected). The number of user edits for the six sketch line cases were approximately the same as in the accuracy/efficiency experiments. Although the accuracy achieved is consistent, we noted a slightly larger segmentation error which we attributed to the purposefully sub-optimal sketch line placement.

Repeatability in Magnetic Lasso is supported by dropping seed points in similar locations around an object thus ensuring the algorithm will compute a similar minimum cost path. However, in noisy images we have observed that even a slight change in position of a seed point can result in an erroneous trace section forcing the user to backtrack, often repeatedly. A similar situation occurs when using Snap - a slight change in the foreground and/or background strokes can generate a completely different result.

6.3 Efficiency

The total elapsed time to perform the segmentation is one way to measure efficiency, especially if the efficiency of the computational part of the method allows for real-time interaction. Another definition measures the amount and nature of user interaction (for example, the degree of interaction might be the number of mouse clicks required). Evaluating user effort

(and hence efficiency) in terms of the *nature* of the interaction requires the measuring of task complexity and involves such issues as the type of input device operations required by the segmentation method. A formal evaluation of the nature of interaction is beyond the scope of this paper. Based on our own observations however, we believe the effort to control the pen or mouse increases with the requirement to *trace* the object boundary carefully, resulting in reduced efficiency and user fatigue.

To verify our observations, we have performed a series of experiments to compare the efficiency of SketchSnakes to LiveWire [9] (Table 2), using the total elapsed time and number of user interactions as our efficiency measure. Livewire was chosen in part because there is a highly optimized version commercially available in Adobe Photoshop (Magnetic Lasso). In the case of LiveWire, a user interaction is defined as planting a seed point or undoing a seed point (backtracking). In the case of SketchSnakes, a user interaction is defined as nudging a control point with the pen/mouse after the Snake has snapped to the boundary. In other words, in both cases user interactions are defined as "extra" user actions and do not include the act of tracing around the object with LiveWire or the act of drawing sketch lines with SketchSnakes (both of which we consider as "normal" user actions).

The Adobe Photoshop Magnetic Lasso tool allows the user to specify the automatic dropping of seed points. We have found, for our corpus callosum experiments, that a setting of 100 (on a scale of 0 (no seeds automatically dropped) to 100 (the most automatic setting)) consistently produced the most efficient and accurate result, with the maximum efficiency. If we disable the auto-seeding facility completely, the user is forced to plant more seeds and this adds considerably to the time taken as well as to the degree of concentration required. On the other hand, a setting of 100 can occasionally result in erroneous traces, especially if the the user traces too quickly, requiring the use of a separate editing tool to make corrections. Another factor that affected Magnetic Lasso efficiency is the requirement to trace around the object boundary results in situations where the user's hand is occluding the boundary, making it difficult to determine whether the trace is accurate. Users are forced to spend time re-orienting their hand while keeping the pen on the tablet screen in order to gain a better viewpoint. The use of a mouse eliminates this problem but introduces new problems of control and accuracy thus efficiency is noticeably worse with a mouse.

The sketch line initialization process of SketchSnakes does not suffer nearly as much from this occlusion problem since the user is drawing across the width of the object with the pen using short strokes. This sketch line action is also fairly easy to perform with the mouse as well and therefore we did not observe as significant a difference in efficiency with SketchSnakes as observed with Magnetic Lasso. The insensitivity to object shape of SketchSnakes and sketch line initialization can be inferred from the table. The initialization time is very consistent across all twenty-five data sets and in the experiments performed in Section 5. In general, it requires less than one second to draw a good sketch line for most objects.

The occlusion problem mentioned above raises another related and potentially important

[9] We decided against performing the efficiency experiments using Snap. In many of the images used in the experiments, we were unable to successfully generate an accurate segmentation in a time comparable to SketchSnakes and Magnetic Lasso due to excessive editing requirements.

	LiveWire : SketchSnakes			
Case no.	Time (seconds)	Interactions	Mean error	Max error
2	15 : 10	3 : 1	0.96 : 0.86	3.2 : 2.2
3	17 : 8	5 : 0	0.50 : 0.55	2.0 : 2.0
4	15 : 14	6 : 4	0.55: 0.55	2.2 : 1.4
5	10 : 7	0 : 0	0.7 : 0.8	2.0 : 2.2
6	14 : 7	3 : 0	0.55 : 0.6	1.4 : 1.4
7	14 : 14	3 : 4	0.48 : 0.58	2.0 : 1.4
8	21 : 13	6 : 3	0.67 : 0.56	2.2 : 2.0
9	22 : 19	6 : 7	0.50 : 0.56	2.2 : 2.0
10	22 : 12	8 : 3	0.61 : 0.78	2.2 : 2.2
11	14 : 8	0 : 1	0.52 : 0.50	2.0 : 2.0
12	24 : 19	9 : 8	0.52 : 0.50	2.2 : 2.2
13	18 : 14	7 : 4	0.56 : 0.53	2.2 : 1.4
14	19 : 15	7 : 5	0.77 : 0.65	2.2 : 1.4
15	15 : 10	4 : 4	0.81 : 0.67	2.8 : 1.4
16	19 : 10	7 : 2	0.78 : 0.62	3.2 : 2.2
17	10 : 14	1 : 4	0.86 : 0.76	2.8 : 2.2
18	15 : 13	4 : 3	0.69 : 0.78	2.0 : 1.4
19	20 : 13	9 : 5	0.93 : 0.60	5.0 : 2.0
20	17 : 10	7 : 3	1.0: 0.72	2.2 : 2.0
21	19 : 10	4 : 3	0.55 : 0.65	2.0 : 2.0
22	20 : 9	5 : 4	0.53 : 0.52	2.2 : 1.4
23	21 : 10	8 : 3	0.68 : 0.55	2.2 : 1.4
24	23 : 10	8 : 3	0.65 : 0.66	2.0 : 1.4
25	17 : 13	3 : 4	0.54 : 0.61	2.2 : 2.0

Table 2
Comparison of efficiency between Magnetic Lasso and SketchSnakes for extracting the corpus callosum from 24 2D mid-sagittal MR brain images. The total elapsed time and number of user interactions are shown as well as the mean shortest distance between the extracted and expert segmented CC boundaries (measured in pixels) and the maximum distance between the extracted and expert boundaries.

"comfort" issue - an issue which may also affect efficiency. In our experience, tracing completely around the object with the pen in one continuous tracing action is fatiguing, especially for elongated objects; the tracing action requires the user to concentrate for a considerable

period of time. The short stroke actions of the sketch line initialization, on the other hand, allow the user to rest between strokes (for as long as desired) and/ or edit the current initial contour. These user fatigue issues may become increasingly important when they are considered over longer time scales.

6.4 Editing and Steering

Although editing of the segmentation result enhances its accuracy, it can reduce efficiency and produce non-repeatable interaction, especially if the editing is performed at the pixel level. Therefore, the first goal of an interactive segmentation method is to minimize the need for editing. Nevertheless, some editing is typically always required, especially for noisy images. Steering involves dynamically providing local information to guide the computational part of the algorithm in a process of progressive refinement toward the desired segmentation outcome. When properly designed, interactive steering, which includes initialization, minimizes (or in some cases eliminates) the need for manual editing.

In our opinion, the fundamental problem with Livewire-type algorithms is the inability to seamlessly switch modes and precisely edit the trace at any time and any location along a previous part of the trace. The trace often does not match the desired boundary and it can be difficult to plant seeds to construct a precise boundary segment. Although backtracking (deleting seeds) and redoing a boundary segment is often effective it can also be an exercise in frustration. One would like to "turn off" the automatic trace (without switching modes or tools) and manually generate a smooth segment before continuing. Furthermore, in Magnetic Lasso once the entire object has been traced, if any mistakes remain they must be fixed using separate editing tools. Another difficulty is the inability to get an overview of the segmentation as the user is tracing around the boundary.

As mentioned previously, in addition to its foreground/background stroke painting input mode, Snap has several boundary editing facilities available. In regions of low contrast - a common occurrence in medical images - it is difficult to draw additional foreground strokes that will generate the desired result. This forces the user to switch input modes and use the boundary editing tools. However, when to enter this editing mode and which boundary editing tool to use is a critical issue. We have found that converting the Snap output to a spline curve and simply dragging control points is often the most effective boundary editing operation and is similar to SketchSnakes editing. However, if this input mode is entered too soon (i.e. before sufficient foreground/background strokes have been painted to generate a good initial segmentation) then the user is faced with the tedious task of manually manipulating many control points.

Livewire is admittedly quite fast for some objects when only a few seeds are required and the object is not elongated and/or does not exhibit any regions of high curvature. For example, the lung boundary (Figure 1) can be traced quite easily. However, in our experience if the user traces too quickly, inaccurate trace segments will occur more frequently and the user is never quite sure where these inaccuracies will occur (although experience ameliorates this problem to some degree). Another issue is the setting of the frequency of automatic seed

dropping. At the most automatic setting the user needs to slow down, especially in high-curvature regions or noisy regions, or else mistakes are made. If no automatic seed dropping is used, many seed points are often required for objects with highly-curved regions. For many objects we have often found it difficult to discover an optimal setting that maximizes both efficiency and accuracy.

With SketchSnakes, initialization, fitting, steering, and editing are all performed in a single process as part of the of the segmentation work-flow. The user is able to stop at any time, examine the current initial contour or partial segmentation result, zoom in and out, and make precise corrections before continuing, all with simple pen actions. One side effect of this flexibility is users must freeze manually-edited Snake segments if they want to perform another fitting step. As mentioned in Section 4.2, individual control points, control polygon edges, or entire regions of the Snake can be frozen with simple selection actions and/or sketching actions.

Editing with SketchSnakes can occasionally suffer from the same problems that all spline-based curves exhibit. If there are many control points to reposition, especially if several are clustered together (for example, around a very sharp/thin region or protrusion of an object), the user spends time nudging them all into place. We are currently extending SketchSnakes to detect control point clusters and automatically remove extraneous control points. Another problem is the movement of one control point can occasionally also slightly (and detrimentally) affect the shape of a curve region that had been previously corrected by moving a neighboring control point. The remedy in this situation is the insertion of a new control point to provide more precise shape control. A final editing issue is how to visually represent the control points, the subdivision curve, and the cursor so that the user can see the underlying object boundary and easily distinguish and select clustered control points. We find that sometimes the edge detected image is a useful guide and perform editing using this image. At other times, the original image is more useful to guide curve placement. Zooming can be effective for precise repositioning but overall object boundary context is reduced, especially for noisy regions. These seemingly small interaction and visualization issues can actually have a significant impact on the user's overall experience with an image processing tool and we are currently investigating several promising approaches.

7 Conclusion

Optimizing the performance of interactive medical image segmentation methods for noisy images (and image sequences) requires a minimal number of fast, simple and fatigue-free interactions, intuitive, flexible, and precise user control and editing capabilities, and the ability to dynamically transfer the high-level image interpretation of the human expert to the algorithm. Furthermore, all of these capabilities must be seamlessly integrated into a single work-flow and presented to the user using a consistent, intuitive interaction model. We believe the combination of sketching input lines across an object, a pen input device that can be used directly on the screen, and a powerful subdivision-curve based Snake results in a tool that meets these requirements and is effective for many segmentation tasks that cannot

be processed as efficiently with other techniques.

We have recently extended the SketchSnakes paradigm to 3D and used deformable subdivision surfaces to create *SketchSurfaces* [35]. The sketch lines are drawn across the cross-section of the target object in several key image slices of a volume image spanning the length of the object. Our initial results are extremely promising, allowing the user to extract and reconstruct a highly-accurate analytical surface model of several brain structures in under sixty seconds.

References

[1] Adobe Systems Inc., *Adobe Photoshop User Guide*, 2002.

[2] Digital Film Tools LLC, *Snap User Guide*, 2007.

[3] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[4] Eric N. Mortensen and William A. Barrett, "Intelligent scissors for image composition," in *Proceedings of Computer Graphics (SIGGRAPH'95)*, Los Angeles, CA, August 1995, pp. 191–198.

[5] W.A. Barrett and E.N. Mortensen, "Interactive live-wire boundary extraction," *Medical Image Analysis*, vol. 1, no. 4, pp. 331–341, 1997.

[6] Eric N. Mortensen and William A. Barrett, "Interactive segmentation with intelligent scissors," *Graphical Models and Image Processing*, vol. 60, pp. 349–384, 1998.

[7] A.X. Falcão, J.K. Udupa, S. Samarasekera, and S. Sharma, "User-steered image segmentation paradigms: Live wire and live lane," *Graphical Models and Image Processing*, vol. 60, 1998.

[8] A.X. Falcão, J.K. Udupa, and F.K. Miyazawa, "An ultra-fast usersteered segmentation paradigm: Live-wire-on-the-fly.," *IEEE Transactions on Medical Imaging*, vol. 19, no. 1, pp. 55–62, 2000.

[9] L.D. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2D and 3D images," *IEEE Trans. on PAMI*, vol. 15, no. 11, pp. 1131–1147, November 1993.

[10] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[11] S. Lobregt and M.A. Viergever, "A discrete dynamic contour model," *IEEE Trans. on Medical Imaging*, vol. 14, no. 1, pp. 12–24, March 1995.

[12] Tim McInerney and Demetri Terzopoulos, "T-snakes: Topology adaptive snakes," *Medical Image Analysis*, vol. 4, pp. 73–91, 2000.

[13] J. Suri, K. Liu, S. Singh, S. Laxminarayana, and L. Reden, "Shape recovery algorithms using level sets in 2-d/3-d medical imagery: A state-of-the-art review.," *IEEE Trans. Inform. Technol. Biomed.*, vol. 6, pp. 8–28, 2002.

[14] J. Liang, T. McInerney, and D. Terzopoulos, "United snakes," *Medical Image Analysis*, vol. 10, no. 2, pp. 213–233, 2006.

[15] T. McInerney and H. Dehmeshki, "User-defined B-spline template-snakes," in *Proc. Sixth International Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI 99)*, Montreal, Canada, November 2003, Springer.

[16] T. McInerney and M.R. Akhavan Sharif, "Sketch initialized snakes for rapid, accurate, and repeatable interactive medical image segmentation," in *IEEE International Symposium on Biomedical Imaging (ISBI'06), Arlington, Virginia, April 2006*, 2006.

[17] P. Brigger and M. Unser, "B-Spline Snakes: A Flexible Tool for Parametric Contour Detection," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1484–1496, 2000.

[18] S. Menet, P. Saint-Marc, and G. Medioni, "B-snakes: Implementation and application to stereo," in *Proceedings DARPA*, 1990, pp. 720–726.

[19] Andrew Blake and Michael Isard, *Active Contours*, Springer-Verlag, 1998.

[20] G. Chaikin, "An algorithm for high speed curve generation," *Computer Graphics and Image Processing*, vol. 3, pp. 346–349, 1974.

[21] N. Dyn, J. Gregory, and D. Levin, "A four-point interpolatory subdivision scheme for curve design," *Computer-Aided Geometric Design*, vol. 4, pp. 257–268, 1987.

[22] Johannes Hug, Christian Brechbühler, and Gábor Székély, "Tamed snake: A particle system for robust semi-automatic segmentation," in *Proc. Second International Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI 99)*, Cambridge, England, September 1999, Springer.

[23] J. Ivins and J. Porrill, "Statistical snakes: Active region models," in *Proc. 5th British Machine Conf. (BMVC'94)*. 1994, pp. 377–386, BMVA Press.

[24] Stephan Bischoff and Leif P. Kobbelt, "Snakes with topology control," *The Visual Computer*, 2003.

[25] S.J. Osher and J.A. Sethian, "Fronts propogating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[26] A.X. Falcão, J.K. Udupa, S. Samarasekera, and B.F. Hirsch, "User-steered image boundary segmentation," in *Proceedings of SPIE on Medical Imaging, vol 2710*, Newport Beach, CA, 1996, pp. 278–288.

[27] Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images," in *Proc. Eighth International Conf. on Computer Vision (ICCV'01), Vancouver, Canada, July, 2001*, 2001.

[28] Y. Li, J. Sun, C.K. Tang, and H.Y. Shum, "Lazy snapping," in *Proc. of ACM Siggraph*, 2004, pp. 303–313.

[29] C. Rother, A. Blake, and V. Kolmogorov, "Grabcut - interactive foreground extraction using iterated graph cuts," in *Proc. of ACM Siggraph*, 2004, pp. 309–314.

[30] D. Zorin, P. Schröder, A. DeRose, L. Kobbelt, A. Levin, and W. Sweldens, "Subdivision for modeling and animation," in *SIGGRAPH 2000 Course Notes*. 2000.

[31] T. McInerney, M.R. Akhavan Sharif, and N. Pashotanizadeh, "Java extensible snakes system," in *SPIE Int. Symposium, Medical Imaging*, San Diego, CA, February 2005.

[32] J. Maillot and J. Stam, "A unified subdivision scheme for polygonal modeling," in *Proceedings of Eurographics*. 2001, vol. 20, Computer Graphics Forum.

[33] Seth Green, George Turkiyyah, and Duane Storti, "Subdivisionbased multilevel methods for large scale engineering simulation of thin shells," in *7th ACM Symposium on Solid Modeling and Applications 2002*, Saarbrucken, Germany, 2002.

[34] S.D. Olabarriaga and A.W.M. Smeulders, "Interaction in the segmentation of medical images: A survey," *Medical Image Analysis*, vol. 5, pp. 127–142, 2001.

[35] M. Aliroteh and T. McInerney, "Sketchsurfaces: Sketch-line initialized deformable surfaces for efficient and controllable interactive 3d medical image segmentation," in *3rd International Symposium on Visual Computing (ISVC07)*, 2007, pp. 542–553.