Semiautomatic Video Object Segmentation Using VSnakes

Shijun Sun, *Member, IEEE*, David R. Haynor, and Yongmin Kim, *Fellow, IEEE*

***Abstract*—Video object segmentation and tracking are essential for content-based video processing. This paper presents a framework for a semiautomatic approach to this problem. A semantic video object is initialized with human assistance in a key frame. The video object is then tracked and segmented automatically in the following frames. A new active contour model, VSnakes, is introduced as a segmentation method in this framework. The active contour energy is defined so as to reflect the energy difference between two contours instead of the energy of a single contour. Multiple-resolution wavelet decomposition is applied in generating the edge energy of the image frame. Contour relaxation is used to deal with the object deformation frame by frame, and the Viterbi algorithm is used to update the contour path during contour relaxation. Compared to the original snakes algorithm, semiautomatic video object segmentation with the VSnakes algorithm resulted in improved performance in terms of video object shape distortion (1.4% versus 2.9% in one experiment), which suggests that it could be a useful tool in many content-based video applications, e.g., MPEG-4 video object generation and medical imaging.**

***Index Terms*—Contour relaxation, segmentation, tracking, video object, VSnakes.**

I. Introduction

BUILDING on recent advances in computing technologies, new digital video applications and services are rapidly becoming reality. Block-based compression techniques, such as H.261, H.263, MPEG-1, and MPEG-2 [1], have been widely used in digital TV, DVD, desktop video conferencing, and telemedicine. These techniques are simple and economical in terms of computation and required hardware complexity. However, most existing digital video standards are built on the single-layer frame-based coding techniques, it is difficult to extract any meaningful contents from MPEG-1 or MPEG-2 [1] bitstream without decompression and further analysis.

To obtain improved visual quality and content-based functionality, more advanced coding techniques have been explored during the past decade. Content or object-based video coding standards are the principal contenders for the next generation of video coding. For example, MPEG-4 [2] enables video streams represented as semantic visual information or meaningful entities, which are called video objects (VOs). The introduction of semantic video objects into bit streams could improve coding efficiency for both storage and transmission tasks, and could provide flexible video query capability.

Extensive research has gone into general-purpose video object extraction systems. In the Core Experiment N2 on Automatic Segmentation Techniques during the MPEG-4 investigation [2], [3], various automatic segmentation techniques for moving objects were proposed. However, these techniques often do not obtain desirable segmentation results because the mathematical model required for the extraction of a video object is difficult to define. In a segmentation algorithm, the definition of homogeneity, or object coherence, is a crucial factor. Different definitions of homogeneity can lead to totally different segmentations given the same input data. Automatic image-segmentation algorithms typically segment an image into regions that are homogeneous with respect to a certain characteristic, e.g., color, texture, or motion. However, a semantic video object may contain multiple colors and multiple motions. A single homogeneous color, texture, or motion criterion does not lead to the extraction of complete semantic visual information. Attempts (such as [3]–[5]) have been made in combining color, texture, and motion information for automatic foreground/background segmentation. Since control parameters for color and motion have to be manually set for each sequence, the techniques still cannot perform exactly in an automatic fashion. Given the current status of automatic segmentation technology, semiautomatic methods currently are more feasible for generating meaningful video objects. If a user can define a video object at the time of its first appearance, semantics can then be used as the segmentation criterion, and we can obtain meaningful segmentation results through the sequence.

A semiautomatic segmentation process consists of two steps: supervised initial segmentation and unsupervised tracking [6]. The segmentation starts with a user-assisted initialization of the desired object contour in an initial key frame and is then followed by automatic segmentation in the following frames. The simplest method for key frame initialization is to have the user approximate the contour of the video object with a polygon. Some image segmentation techniques, such as graph search approaches [7], [8], can also be used to reduce the amount of user assistance.

We have developed an integrated system for semiautomatic video object segmentation, using Modified Continuous-Valued Adaptive Resonant Theory (M-ART2) [9] for scene-change detection and two-dimensional (2-D) Correlation-based Adaptive Predictive Search (CAPS) [10] and the M-ART2 tracker [9] for video object tracking. After object contour initialization in a key

Manuscript received June 1, 2000; revised April 28, 2002. This paper was recommended by Associate Editor Q. Tian.

S. Sun was with the Image Computing Systems Laboratory, University of Washington, Seattle, WA 98195 USA. He is now with Sharp Laboratories of America, Camas, WA 98607 USA (e-mail: ssun@sharplabs.com).

D. R. Haynor is with the Department of Radiology, University of Washington, Seattle, WA 98195 USA (e-mail: haynor@u.washington.edu).

Y. Kim is with the Image Computing Systems Laboratory, Departments of Bioengineering and Electrical Engineering, University of Washington, Seattle, WA 98195-2500 USA (e-mail: ykim@u.washington.edu).

Digital Object Identifier 10.1109/TCSVT.2002.808089

frame, a segmentation module in our system refines the contour and defines it as the object template. The tracking algorithms predict the global translational motion of the object from the current frame to the next frame. The segmentation module then refines and updates the object boundary in the next frame based on the prediction. The process continues one frame at a time through the whole sequence until a new scene occurs or the automatically generated contour is no longer judged by the operator to be satisfactory. Our segmentation algorithm, an active contour model, is the main focus in this paper. CAPS and M-ART2 have been presented elsewhere [9], [10].

Active contour models, or snakes [11], have been extensively studied and applied as image segmentation methods during the past decade. A snake is an elastic curve placed on an image that deforms from an initial shape to adjust to certain image features, e.g., the boundary of a video object. The solution is found by minimizing the snake's total energy, which is typically composed of energy terms for internal tension (stretching) and stiffness (bending) as well as a potential term that is derived from certain image features, e.g., edges or corners. A pressure force can also be added so that closed contours inflate like balloons [12]

$$E_{\text{snake}} = \alpha \cdot E_{\text{tension}} + \beta \cdot E_{\text{stiffness}} + \gamma \cdot E_{\text{image}} + \rho \cdot E_{\text{pressure}}. \quad (1)$$

The behavior of a snake is strongly influenced by the weighting parameters α, β, γ, and ρ in (1). Choosing the optimal values for each application is difficult. Traditional snakes also face the problem of avoiding attraction to spurious local features, e.g., spurious edge points. Variational approaches have been proposed for solving active contour models [11]. However, variational approaches do not guarantee global optimality of the solution, and the method may fail when the features to be followed, e.g., edges, are not connected. Therefore, for the active contour model to succeed, the initial active contour may have to be placed very close to the real object boundary.

Approaches have been developed to overcome the drawbacks of the original snake method. Amini *et al.* [13] introduced dynamic programming as an optimization approach to derive the optimal active contour by minimizing the contour energy. At each iteration, dynamic programming finds the minimum energy deformation among all possible local deformations, reducing the susceptibility to trapping by spurious edges. In this paper, we present a new class of active contour approaches, VSnakes. The VSnakes algorithm defines a differential contour energy, which reflects the difference between successive contours, rather than directly defining the energy of a contour. Multiresolution wavelet decomposition is used to generate edge potential fields. The minimization process is similar to dynamic programming [13] and uses the Viterbi algorithm [14] to derive the global optimal contour during relaxation. Section II describes the VSnakes algorithm. The experimental results of VSnakes are presented in Section III, and a discussion is given in Section IV.

II. VSnakes

In each image frame, video object segmentation starts with an initialization, i.e., either a user-assisted initialized contour for the key frame or a contour prediction by a tracking algorithm [9], [10] for subsequent frames. The contour is represented as a set of *contour points*, which is a set of 8-connected image pixels. Although a spline representation was employed in the original snakes [11], the results are not acceptable for high-quality visual information and the error might propagate during video object tracking [15].

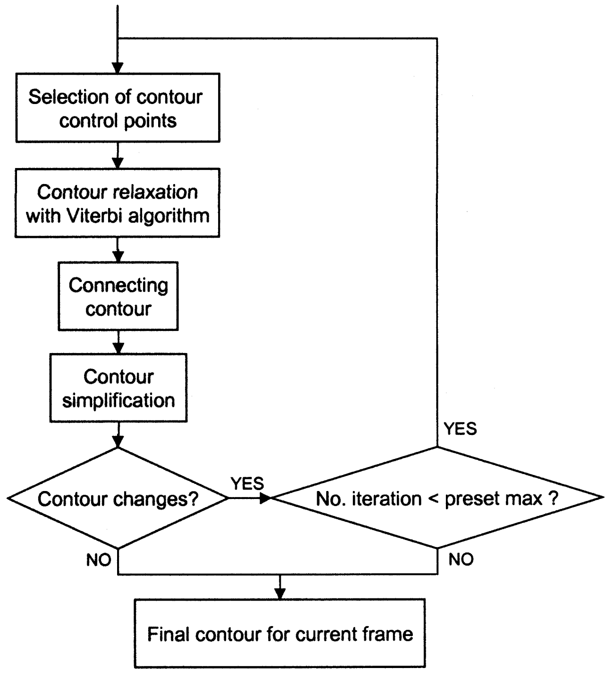
Selection of contour control points
Contour relaxation with Viterbi algorithm
Connecting contour
Contour simplification
Contour changes?
YES
No. iteration < preset max ?
YES
NO
NO
Final contour for current frame

Fig. 1. Flowchart for the iterative VSnakes.

After initialization, the contour of the targeted video object is updated iteratively. Fig. 1 shows the flowchart for the VSnakes iteration process. During each iteration, a set of contour *control points* is first generated along the current video object boundary by subsampling the contour points. The control points are chosen so that adjacent points are between 3 and 7 pixels apart. The contour is then relaxed using the Viterbi algorithm. The third step is a fine-tuning step, in which a continuous contour is formed based on the updated control points. Finally, the contour is simplified to smooth the contour and reduce the number of contour points, completing one iteration. The iterative process stops when the active contour no longer moves significantly between successive iterations or when the number of iterations exceeds a user-specified maximum.

Section II-A introduces our method of calculating the edge potential energy in an image frame. In Section II-B, we describe our definition of the differential energy function for our active contour model. In Section II-C, we present the VSnakes minimization process: the relaxation of an active contour using the Viterbi algorithm. Finally, the contour fine-tuning and simplification procedures are described in Sections II-D and E.

A. Edge Potential Based on Multiple-Resolution Wavelet Decomposition

Multiple-resolution wavelet edge detection has been studied by Singh *et al.* [16]. They concluded that the Haar wavelet (D1) has the best performance among various wavelets for edge detection and that five to ten levels of decomposition are normally sufficient to yield clear edges in both noiseless and noisy images. As they point out, a basic problem is how to properly combine the edge information from multiple levels. A similar question that arises in processing color images is how to combine the information from multiple color components, e.g., (Y, C_b, C_r) or (R, G, B).

In our segmentation approach, we used principal component analysis to guide the combination of edge energies. Multiple levels of wavelet-transformed edge information were generated for each color component separately. At each level, the horizontal and vertical details were taken as the horizontal and vertical image gradients respectively, and the edge energy at that level was calculated as the magnitude of the image gradients. To avoid half-pixel shifts in edge location, edge energies were extracted only from the even-number levels (levels 2, 4, 6, 8, and 10) and are denoted by E_2, E_4, E_6, E_8, and E_{10}, respectively. A scalar edge energy for each color component can then be obtained by taking a weighted sum of corresponding multiple-level edge energies

$$E_{\text{color}} = l_2 \cdot E_2 + l_4 \cdot E_4 + l_6 \cdot E_6 + l_8 \cdot E_8 + l_{10} \cdot E_{10} \quad (2)$$

where the weighting factors (l_2, l_4, l_6, l_8, l_{10}) are proportional to the elements of the dominant eigenvector of the covariance matrix of (E_2, E_4, E_6, E_8, E_{10}). Similarly, the final edge energy can be defined as a weighted sum of the edge energies from each individual color component, such as for (Y, C_b, C_r)

$$E_{\text{edge}} = C_Y \cdot E_Y + C_{C_b} \cdot E_{C_b} + C_{C_r} \cdot E_{C_r}. \quad (3)$$

The first 100 frames of three test video sequences, "table-tennis," "flower-garden," and "mobile-train," were studied. Based on the statistics in the 300 sample frames and with a view to simplifying the computation of the final edge energy in fixed-point arithmetic, we chose the weighting factors (l_2, l_4, l_6, l_8, l_{10}) for the multiple-level edge energies to be 1, 2, 4, 8, and 16, respectively, and equal weighting factors (C_Y, C_{C_b}, C_{C_r}) for the three edge energies, E_Y, E_{C_b}, E_{C_r}. Theoretically, the weighting factors could be different for different video objects and sequences with the tradeoff of computational complexity.

B. Active Contour Energy

Our definition of the energy for the active contour model differs from the traditional active contour models in which the contour energy is defined as an integral of an energy function over the contour. Instead, we define a differential energy, which represents the energy difference between a candidate contour and the current contour.

If (x_i^0, y_i^0) with $i = 1 \cdots N$ are the current contour control points, and (x_i^1, y_i^1) with $i = 1 \cdots N$ is a set of candidate contour control points (see Fig. 2), the energy difference ΔE between these two contours is defined as

$$\Delta E = \sum_{i=1}^{N} \delta E_i \quad (4)$$

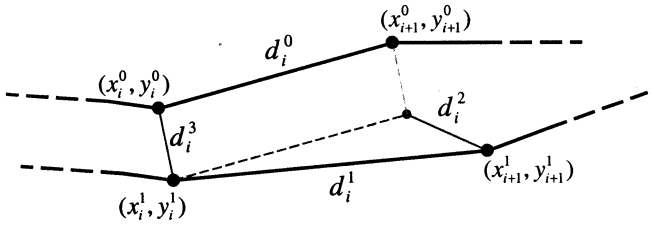

Fig. 2. Current and candidate contour points.

where the energy difference for contour segment i is given by

$$\delta E_i = f_i^0 \cdot \frac{\left|d_i^1 - d_i^0\right|}{d_i^0} + f_i^0 \cdot \frac{d_i^2}{d_i^0} + f_i^0 \cdot \frac{d_i^3}{d_i^0} + \left(f_i^0 - f_i^1\right) \quad (5)$$

where we define

$$f_i^0 = \int_{(x_i^0,\, y_i^0)}^{(x_{i+1}^0,\, y_{i+1}^0)} E_{\text{edge}} \cdot ds \quad (6)$$

$$f_i^1 = \int_{(x_i^1,\, y_i^1)}^{(x_{i+1}^1,\, y_{i+1}^1)} E_{\text{edge}} \cdot ds \quad (7)$$

$$\begin{aligned} d_i^0 &= \left|(x_i^0, y_i^0) - (x_{i+1}^0, y_{i+1}^0)\right| \\ d_i^1 &= \left|(x_i^1, y_i^1) - (x_{i+1}^1, y_{i+1}^1)\right| \\ d_i^2 &= \left|\left[(x_i^1, y_i^1) - (x_{i+1}^1, y_{i+1}^1)\right] - \left[(x_i^0, y_i^0) - (x_{i+1}^0, y_{i+1}^0)\right]\right| \\ d_i^3 &= \left|(x_i^1, y_i^1) - (x_i^0, y_i^0)\right| \end{aligned} \quad (8)$$

where f_i^0 and f_i^1 represent an integration of the edge energy along the ith segment of the current contour and the candidate contour, respectively, d_i^0 and d_i^1 are the ith segment length of the current contour and the candidate contour, respectively, d_i^2 is the length of the (vector) difference between the two segments, and d_i^3 is the distance between the ith current contour control point and the ith candidate contour control point. In discrete image domain, f_i^0 and f_i^1 are calculated as the sum of the edge energy of the pixels along the corresponding contour segments.

In terms of the traditional active contour models, the first term in (5) is related to the "tension." The difference is that the "tension" in the original snakes formulation tends to make the contour shrink completely while the "tension" we have defined here keeps the length of the contour consistent with the initial contour. The term related to d_i^2 is responsible for the "stiffness" that keeps the shape of the candidate contour similar to the current contour while the "stiffness" defined in the original snakes produces only smooth contours without sharp "bending." The term related to d_i^3 acts like the "pressure" in the traditional snakes and is responsible for keeping the candidate contour close to the current contour while the "pressure" term in the original snakes represents either an inflation or deflation force. Finally, the difference between f_i^0 and f_i^1 corresponds to the "image" term in (1). With the active contour energy defined in (5), the optimal contour is the one with the minimum ΔE among all candidate contours.

The advantage of the contour energy definition in (5) over the traditional snake's energy in (1) is that no weighting parameters are necessary in (5) for different energy terms, i.e., *tension*, *stiffness*, *image*, and *pressure*. Mathematically, weighting parameters, if inserted into (5), would give favor to certain *distances*

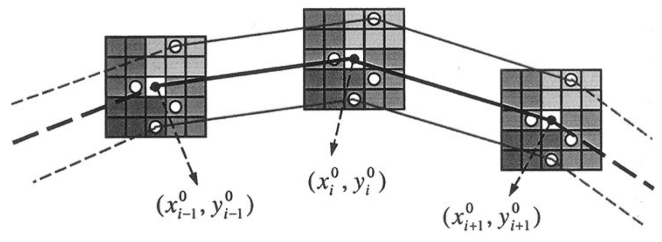

Fig. 3. Illustration of contour relaxation. Open circles represent four points that are candidates to replace the original (solid) point.

defined in (8). In general, there is no such a reason to do so; therefore, we formulated (5) without any weighting parameters.

C. Contour Relaxation With Viterbi Algorithm

Traditional active contour models, as discussed above, use only local information, such as the image gradients, and therefore may not converge to the globally optimal contour. This is the so-called "local minima" problem in the original snakes. We use a dynamic programming technique for contour relaxation that reduces this problem.

Fig. 3 shows the contour-relaxation process. For every current contour control point, we choose four candidate control points from its 5×5 neighborhood. The 5×5 box is divided into four 6-pixel regions as shown in Fig. 3, and the point with the maximum edge energy in each region is chosen as a candidate control point. Thus, for each current control point, there are five candidate replacement points, including the current point itself. If we have N control points along the current contour, the number of possible contours would be 5^N. An exhaustive search is not feasible, and so a more efficient way to solve the problem is needed.

The problem is similar to a 1-D travel problem, which can be solved using the Viterbi algorithm [14]. The algorithm applied in our active contour model can be summarized as follows. : *At each step and for each candidate point, preserve the optimal path and its distance* (ΔE) *from the current path.* In this way, we only need to store five contour energy ΔE values and the equivalent contour points in Fig. 3. At the final (Nth) step, we simply select the contour with the minimum ΔE from the five preserved contours as the optimal contour, which can be expressed as

$$\Delta E^{\min} = \min_{u=1\cdots 5} \left(\Delta E_{N,u}^{\min}\right) \tag{9}$$

where

$$\Delta E_{i,u}^{\min} = \min_{v=1\cdots 5} \left(\delta E_i^{v,u} + \Delta E_{i-1,v}^{\min}\right) \tag{10}$$

represents the minimum ΔE up to the candidate point u at the step i along the contour where $u = 1, \cdots, 5$, $i = 1, \cdots, N$, and

$$\Delta E_{0,u}^{\min} = \min_{v=1\cdots 5} \left(\delta E_0^{v,u}\right) \tag{11}$$

with the term $\delta E_i^{v,u}$ representing the corresponding energy difference (5) for the contour segment between the candidate point v of step i and candidate point u of step $(i+1)$.

For a closed contour, we require the replacement control points of the first and last points to be the same image pixel. Since the Viterbi algorithm [14] is applied in the optimization

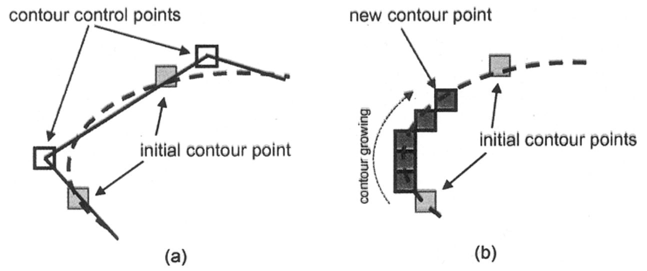

Fig. 4. Illustration of the method for contour connection. (a) Set of initial contour points. (b) Contour growth from one initial contour point to the next.

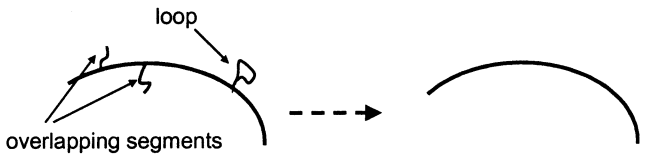

Fig. 5. Illustration of elimination of overlapping contour segments and small loops.

process, we call our active contour model Viterbi Snakes or VSnakes.

D. Contour Connection

After the Viterbi algorithm has been applied to obtain an optimal set of discrete contour control points, we form a continuous contour consisting of a set of 8-connected contour points. First, the pixel with the maximum edge energy along the segment between each pair of adjacent control points is chosen to form a set of *initial contour points* [Fig. 4(a)]. Then, contour points are added between each pair of adjacent initial contour points. The contour grows from one initial contour point to the next [Fig. 4(b)] using the following criteria: 1) each new contour point should be 8-connected to the previous contour point; 2) the distance between the new contour point and the next initial contour point should be less than the distance between the immediately previous contour point and the same initial contour point; and 3) the new contour point should have the highest edge energy among the points satisfying conditions 1) and 2). Note that the contour control points, as defined above, do not necessarily lie on the contour itself.

E. Contour Simplification

The purposes of contour simplification are to reduce the number of contour points and to smooth the contour, and as a result to reduce video object tracking and coding costs. Morphological filters have been used to simplify 2-D segmentation boundaries for coding with low bit rates [17], [18]. Here, we applied a simpler approach to eliminate the redundant contour points.

Overlapping segments and small loops frequently remain after the first three steps of the VSnakes iteration (Fig. 5), especially when the image is very noisy. As illustrated in Fig. 5, we first eliminate every overlapping contour segment and loop that does not occupy a large fraction, e.g., one quarter of the entire contour.

To further reduce the contour points and smooth the contour, we eliminate contour points whose two neighboring contour

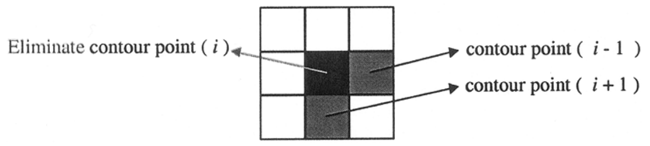

Fig. 6. Contour point elimination when the neighboring two points are 8-connected.

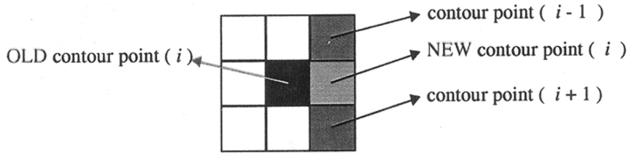

Fig. 7. Contour point translation used to correct fine wiggles.

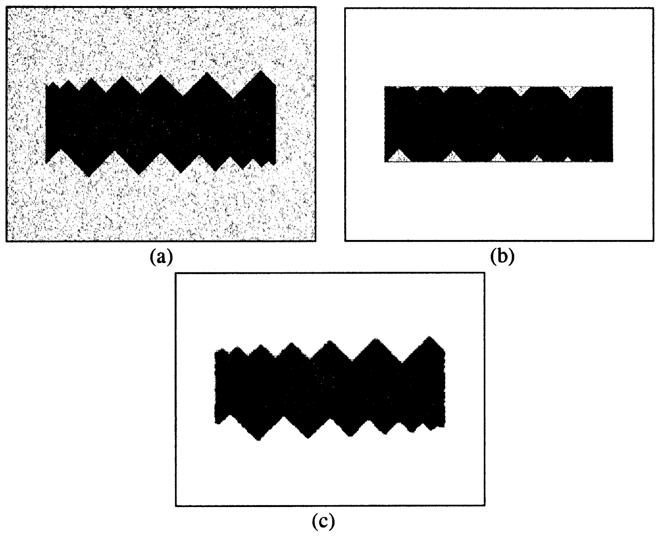

Fig. 8. (a) A 320 × 240 test image contaminated with Gaussian noise. (b) Initial segmentation of the zigzag object. (c) Segmentation of the zigzag object after four VSnakes iterations.

points are 8-connected, as shown in Fig. 6, where the contour point i can be eliminated. Finally, as shown in Fig. 7, when a contour point causes a single-pixel "wiggle" in the contour, it is moved to the middle of the neighboring two points.

Our contour-simplification method processes the contour pixels only and is essentially a one-dimensional (1-D) approach, rendering the computation efficiently compared to 2-D morphological methods [17], [18].

III. Experiments

All the experiments described here were conducted on a Pentium II machine running at 300 MHz with 128 MB of memory and the Windows NT 4.0 operating system.

First, a computer-generated test image, shown in Fig. 8(a), was used to test the convergence of the VSnakes algorithm. The image was formed by a black foreground object with zigzag boundaries on a white background. The whole image was then contaminated by white Gaussian noise with standard deviation of 32. The pixel value was either truncated at 0 or saturated at 255. A rectangle was used to initialize this zigzag object, as shown in Fig. 8(b). Fig. 8(c) shows the object segmentation after four VSnakes iterations. The relative VO shape distortion

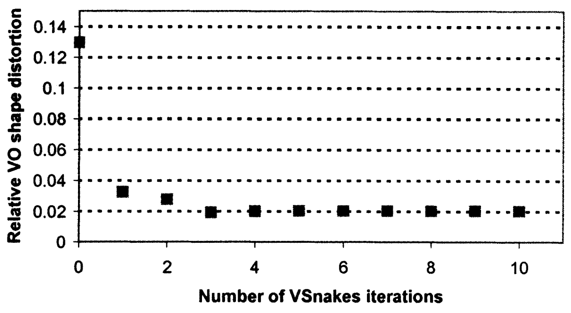

Fig. 9. Relative VO shape distortion of the zigzag object in the test image (Fig. 8) with respect to different number of VSnakes iterations.

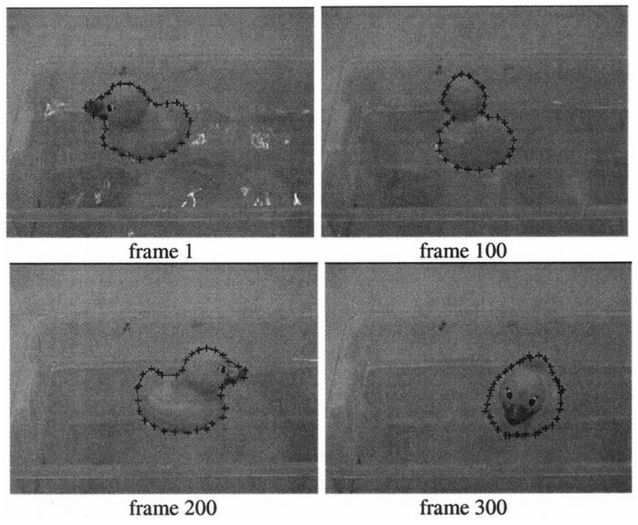

Fig. 10. Frames 1, 100, 200, and 300 of the "duck" sequence and the subsampled contour points for each frame, respectively.

(shown in Fig. 9), which is defined as the number of pixels with discordant labels between the experimental and true segmentations divided by the number of pixels within the true segmentation, decreased to 1.9% after three iterations and stabilized at 2.0% on the fourth and subsequent iterations.

A simple video sequence, "duck," was produced and used to test our algorithm in video object segmentation. In this sequence, a yellow toy duck moves around in a water tank. The color space used by the sequence is (Y, C_b, C_r) in $4:2:0$ format with a luminance resolution of 320 × 240 and a chrominance resolution of 160 × 120.

As we described earlier, the video object segmentation was started with a user-assisted initialization in an initial key frame. VSnakes then refined and updated the object boundary in each of the following frames based on the CAPS [10] or M-ART2 [9] prediction.

The VSnakes algorithm successfully tracked the duck through the entire 300-frame sequence. Several frames of the segmentation results of the duck sequence are shown in Fig. 10. The maximum number of VSnakes iterations was set to 4 for this sequence.

The VSnakes segmentation results were compared to the segmentations with an implementation of the original snakes algorithm [11], where the weighting parameters, α and β in (1) were set to 1.0 and 0.5, respectively. The video object segmentation with the original snakes algorithm followed the same procedure

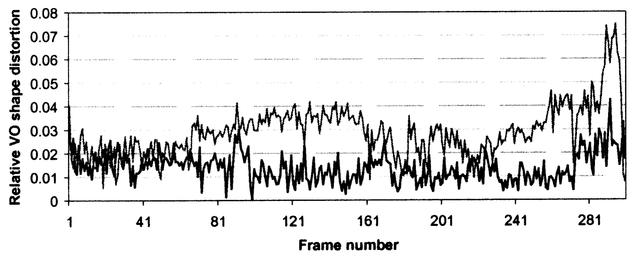

Fig. 11. Relative VO shape distortions of the VSnakes (black curve) and original snakes (gray curve) algorithms in segmenting the duck object.

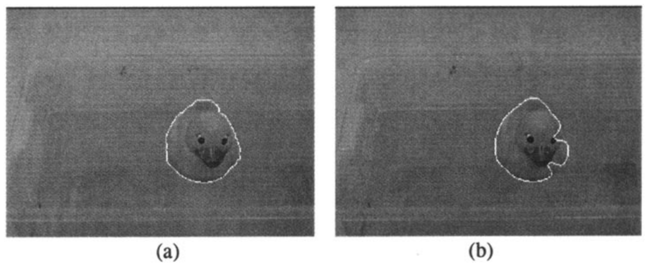

Fig. 12. Segmentations of the duck in frame 280 with (a) VSnakes and (b) the original snakes.

as that of VSnakes. It started with a user-assisted initialization in an initial key frame, and then the original snakes algorithm refined and updated the object contour frame by frame based on the CAPS [10] or M-ART2 [9] prediction.

A "true" segmentation of the duck through the sequence was generated manually frame-by-frame using a segmentation tool based on the graph search method [7]. Fig. 11 compares the relative VO shape distortions of the VSnakes and original snakes algorithms. The VSnakes algorithm shows more consistent and reliable performance. On average, the relative VO shape distortion is 1.4% for VSnakes, while it is 2.9% for the original snakes algorithm. As an example, Fig. 12 shows the segmentations in frame 280 with the VSnakes and original snakes algorithm, respectively. As the duck moves around, the contour derived by the original snakes is attracted to some strong image edges (i.e., around the eye and beak) rather than the duck boundary because only local information has been considered in the snakes algorithm, whereas VSnakes tries to preserve global contour property between frames.

In our VSnakes experiment, it took about 251 milliseconds to generate the edge energy for each frame with the multiple-resolution wavelet decomposition, and it then took another 69 milliseconds on average for VSnakes to segment the duck in each frame. Implementations using new-generation mediaprocessors with a high level of parallelism [19] would greatly speed up edge energy generation.

Additional experiments were conducted with the VSnakes algorithm. Fig. 13 (the left column) shows two video objects, the hand paddle and the ping-pong ball, segmented from the "table-tennis" sequence using VSnakes. Up to three VSnakes iterations were applied for each object in each frame. The algorithm tracked the ping-pong ball well through the sequence. However, the thumb of the player's hand was gradually cut off the hand-paddle object. As a comparison, Fig. 13 (the right column) also shows segmentation examples of the hand paddle

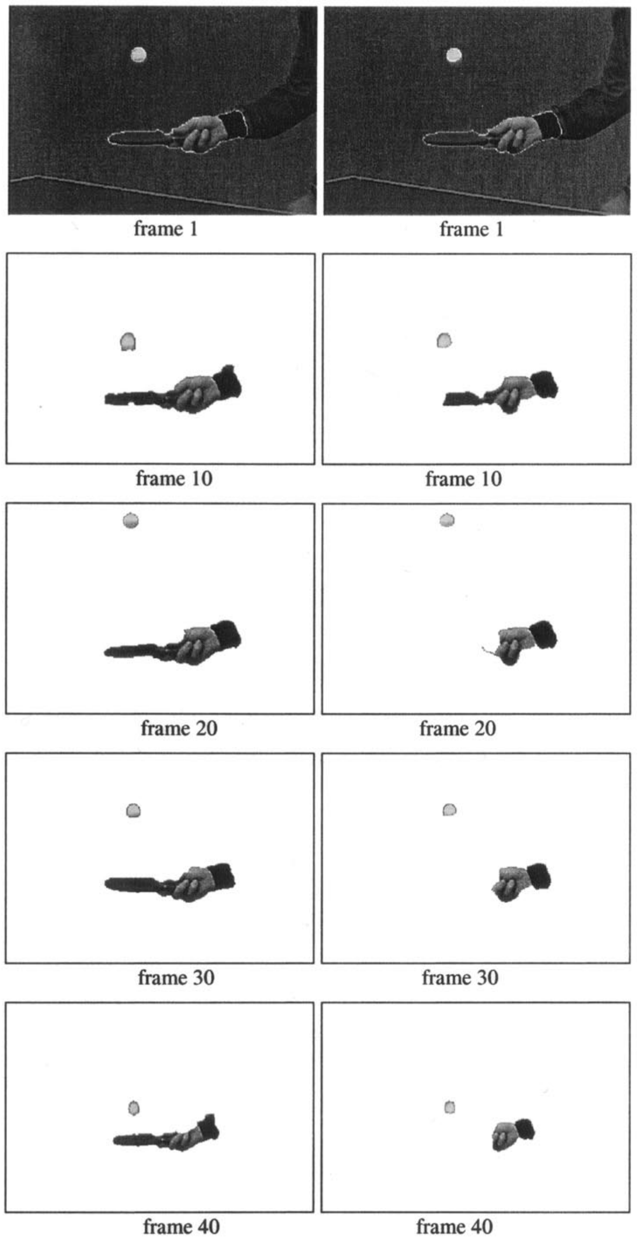

Fig. 13. Segmentations of the hand paddle and ping-pong ball in the "table-tennis" sequence with the VSnakes (the left column) and original snakes (the right column) algorithms.

and the ping-pong ball using the original snakes algorithm. We can see clearly that the paddle and most of the hand were cut off during the process. The relative shape distortion of the hand paddle is shown in Fig. 14 (the black curve) together with that of the original snakes algorithm (the gray curve), showing that VSnakes has better performance in this case than the original snakes algorithm. Fig. 15 shows the segmentation of the house in the 115-frame "flower-garden" sequence using the VSnakes algorithm. The house was successfully recovered despite the occlusion by the foreground tree. Only one VSnakes iteration was applied for each frame to segment this rigid video object. Here, the number of segmentation iterations is applied to control the "rigidness" of the video object, since fewer iterations allows less shape deformation of a video object from frame to frame.

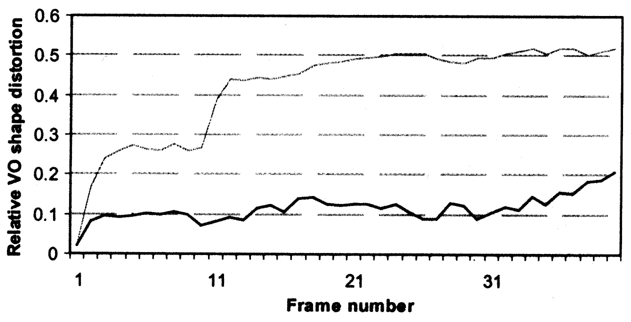

Fig. 14. Relative VO shape distortions of the VSnakes (black curve) and original snakes (gray curve) algorithms in segmenting the hand-paddle object.

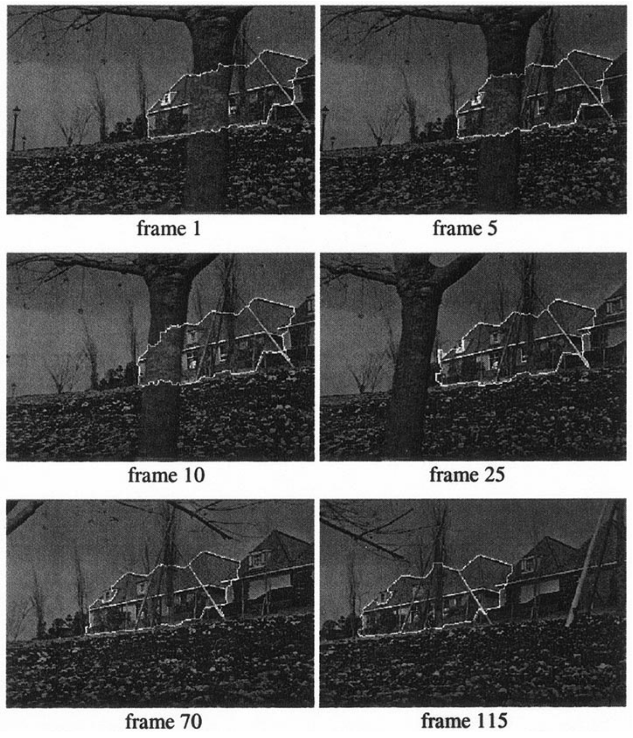

Fig. 15. Segmentations of the house in the "flower-garden" sequence.

IV. Discussion

The VSnakes algorithm has shown promising early results in our semiautomatic video object segmentation and tracking. If a video object is not subject to large deformations from one frame to the next, VSnakes can successfully segment the object in a video sequence with improved accuracy and stability compared to the original snakes algorithm. We introduced a new definition of the differential contour energy. Relaxation with Viterbi algorithm was applied in an active contour model to segment video object undergoing moderate deformations and occlusions. Multiresolution wavelet decomposition was used to generate the image edge energy. Finally, a 1-D boundary simplification technique was developed to smooth the contour and control the error propagation.

The issue of choosing the proper number of VSnakes iterations per frame during video object segmentation remains. Currently, the maximum number of iterations is set manually for each video object to control the "rigidness" of the video object. Usually, fewer iterations allows less shape deformation of a video object from frame to frame. If an initial contour is very close to the actual object boundary, one iteration will suffice. Similarly, a video object, such as the house in the "flower garden" sequence, which does not undergo significant deformation through a sequence, can be segmented using one iteration in each frame. If an initial contour is not somewhat close to the actual object boundary or if the object being segmented undergoes significant deformations or motions, three or more iterations will be necessary. However, more iterations will increase the possibility of the contour to be trapped by other image edges, especially when occlusion is present in the scene. An automated method still needs to be developed to dynamically control the iterative process through video sequences.

With carefully chosen weighting parameters, the original snakes algorithm can successfully segment an object with uniform color or texture when there is no spurious edge nearby. For the simple test image in Fig. 8, there is no obvious performance difference between the VSnakes algorithm and the original snakes algorithm, which converged with a relative VO shape distortion of 2.2%. However, when the content of an object is complicated and there are spurious edges near the object boundary, such as the hand-paddle object, the VSnakes algorithm outperforms the original snakes algorithm. Since the VSnakes algorithm is an edge-based method, the complexity of the object content, such as color or texture, has no significant contribution to the segmentation output. Due to the application of the Viterbi algorithm during contour relaxation, the active contour is less likely to be trapped by spurious edges.

Although VSnakes provides a solution for the minimization problem during contour relaxation, it still shares certain limitation with the original snakes algorithm. The algorithm at this stage can only track video objects without major deformation. Currently, only edge information is considered in the VSnakes energy. Temporal coherency of the object interior will be considered and occlusion modeling will be integrated into the energy function in our future research.

In conclusion, we believe that with further improvement, VSnakes could be potentially useful in many applications, such as machine vision, medical imaging, and MPEG-4 video object generation.

References

[1] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video: Compression Standard*. New York: Chapman & Hall, 1996.
[2] *MPEG-4 Overview—(Seoul Version)*, ISO/IEC JTC1/SC29/WG11 N2725, Mar. 1999.
[3] M. Wollborn, R. Mech, S. Colonnese, U. Mascia, G. Russo, P. Talone, J. G. Choi, M. Kim, M. H. Lee, and C. Ahn, "Description of automatic segmentation techniques developed and tested for MPEG-4 version 1,", ISO/IEC JTC1/SC29/WG11 MPEG97/2702, Oct. 1997.
[4] J. Pan, S. Li, and Y. Zhang, "Automatic extraction of moving objects using multiple features and multiple frames," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS) 2000*, Geneva, May 2000.
[5] T. Meier and K. N. Ngan, "Automatic segmentation of moving objects for video object plan generation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 525–538, Sept. 1998.
[6] C. Gu and M. C. Lee, "Semiautomatic segmentation and tracking of semantic video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 572–584, Sept. 1998.
[7] L. D. Cohen and R. Kimmel, "Global minimum for active contour models: A minimal path approach," *Int. J. Comput. Vis.*, vol. 24, pp. 57–78, Aug. 1997.

[8] E. N. Mortensen and W. A. Barrett, "Toboggan-based intelligent scissors with a four-parameter edge model," in *Proc. 18th IEEE Conf. CVPR*, vol. 2, June 1999, pp. 452–458.

[9] S. Sun and Y. Kim, "Color clustering for scene change detection and object tracking in video sequences," U.S. Patent 6 272 250.

[10] S. Sun, H. W. Park, and Y. Kim, "Template matching using correlative auto-predictive search," U.S. Patent 6 301 387.

[11] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, pp. 321–331, Jan. 1988.

[12] L. D. Cohen, "Note: On active contour models and balloons," in *CVGIP: Image Understanding*, vol. 53, Mar. 1991, pp. 211–218.

[13] A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving various problems in vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 855–867, Sept. 1990.

[14] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[15] A. Blake and M. Isard, *Active Contours: The Application of Techniques From Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. New York: Springer, 1998.

[16] R. Singh, R. Vasquez, and R. Singh, "Comparison of daubechies, coiflet, and symlet for edge detection," *Proc. SPIE*, vol. 3074, pp. 151–159, Apr. 1997.

[17] V. A. Christopoulos, P. D. Muunck, and J. Cornelis, "Contour simplification for segmented still image and video coding: Algorithms and experimental results," *Signal Processing: Image Commun.*, vol. 14, pp. 335–357, Feb. 1999.

[18] C. Gu and M. Kunt, "Contour simplification and motion compensated coding," *Signal Processing: Image Commun.*, vol. 7, pp. 279–296, Nov. 1995.

[19] C. Basoglu, D. Kim, R. J. Gove, and Y. Kim, "High-performance image computing with modern microprocessors," *Int. J. Imaging Syst. Technol.*, vol. 9, pp. 407–415, Dec. 1998.

Shijun Sun (S'99–M'01) received the B.S. degree in physics from the University of Science and Technology of China, Hefei, China, in 1992, the M.S. degree in physics and the Ph.D. degree in electrical engineering, both from University of Washington, Seattle, in 1997 and 2000, respectively.

He joined Sharp Laboratories of America, Camas, WA, in 2000 as a member of technical staff of the Digital Video Department. His research interests are in the area of image and video processing, with emphasis on compression, streaming, real-time computing, medical imaging, segmentation, and computer vision.

David R. Haynor received the B.A. degree in mathematics from Harvard University, Cambridge, MA, in 1968, the Ph.D. degree in mathematics from the University of California, Berkeley, in 1971, and the M.D. degree from Harvard Medical School, Cambridge, MA, in 1979.

He is currently a Professor of Radiology and an Adjunct Professor of Bioengineering at the University of Washington, Seattle. His research interests include image segmentation, functional MRI, and Bayesian statistics.

Yongmin Kim (S'79–M'82–SM'87–F'96) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Wisconsin, Madison.

He is Professor and Chair of Bioengineering, Professor of Electrical Engineering, and Adjunct Professor of Radiology and of Computer Science and Engineering at the University of Washington, Seattle. His research interests include algorithms and systems for multimedia, image processing, computer graphics, medical imaging, high-performance programmable processor architecture, and modeling and simulation. His group has filed 72 invention disclosures and 69 granted patents, and 21 commercial licenses have been signed.

Dr. Kim is a member of the Editorial Board of Proceedings of IEEE, the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, the IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, and the IEEE Press series in Biomedical Engineering. He received the Early Career Achievement Award from the IEEE Engineering in Medicine and Biology Society in 1988.