Computer Vision and Image Understanding 115 (2011) 1375–1383

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Markov surfaces: A probabilistic framework for user-assisted three-dimensional image segmentation

Yongsheng Pan [a,*], Won-Ki Jeong [b], Ross Whitaker [a]

[a] Scientific Computing and Imaging Institute, University of Utah, UT 84112, USA
[b] Initiative in Innovative Computing, School of Engineering and Applied Science, Harvard University, MA 02138, USA

ARTICLE INFO

Article history:
Received 29 October 2009
Accepted 8 June 2011
Available online 16 June 2011

Keywords:
Image segmentation
Probabilistic framework
Markov chain
GPU

ABSTRACT

This paper presents *Markov surfaces*, a probabilistic algorithm for user-assisted segmentation of elongated structures in 3D images. The 3D segmentation problem is formulated as a path-finding problem, where path probabilities are described by Markov chains. Users define points, curves, or regions on 2D image slices, and the algorithm connects these user-defined features in a way that respects the underlying elongated structure in data. Transition probabilities in the Markov model are derived from intensity matches and interslice correspondences, which are generated from a slice-to-slice registration algorithm. Bézier interpolations between paths are applied to generate smooth surfaces. Subgrid accuracy is achieved by linear interpolations of image intensities and the interslice correspondences. Experimental results on synthetic and real data demonstrate that Markov surfaces can segment regions that are defined by texture, nearby context, and motion. A parallel implementation on a streaming parallel computer architecture, a graphics processor, makes the method interactive for 3D data.

1. Introduction

Despite many significant advances in machine vision, many 3D image segmentation in radiation oncology, cardiology, and psychiatry still cannot be fully automatic. Several examples, especially when the boundary of the object is not clearly separable using intensity differences, are shown in Fig. 1. In Fig. 1a, the background and the cross-shaped object at the center have the same texture patterns with a slightly different orientation. The transmission electron microscopy (TEM) data of retinal ganglia shown in Fig. 1b is hard to be segmented, even if the regions there can be distinguished by some combination of texture and dark boundaries. Fig. 1c shows an example of a magnetic resonance imaging (MRI) of a heart. The wall between a left atrium and a left ventricle in the left image is usually very thin and fuzzy. It is hard to get good segmentation results without user interaction in these cases. Semiautomatic segmentation using user interaction, therefore, seems necessary in these cases.

Several methods [1–5] have been proposed for semiautomatic segmentation. A live wire algorithm is proposed in [1] to formulate boundary extraction as a graph searching problem. It utilizes the start points and the end points specified by users, and generates paths between these points using local gradient features. Falcao et al. [2] improved the efficiency of this method using live lane, and Schenk et al. [5] extended the live wire method to 3D based on shape-based interpolation and optimization. Both methods need users to interactively specify points for segmentation. A turtle segmentation algorithm based on 3D live wire has been proposed by Poon et al. [6]. This algorithm is able to segment complex objects with arbitrary topology. But it usually requires many user interaction steps to get satisfactory results.

Level-set methods [3,4,7,8] have become popular for image segmentation because they are able to handle topological changes automatically. But these methods are prone to converge to a local minimum. Yushkevich et al. [8] introduced user interactions to level set methods. In their ITK-SNAP tool [9], users are able to initialize the 3D active contour, set up parameters and perform post processing. However, their approach is sensitive to parameter selection and contour initialization. Furthermore, level set methods are computationally intense, and users have no steering control during the curve evolution.

Ardon et al. [3] proposed a surface extraction method based on the start and end curves specified by users. A network of minimal paths between these curves are generated using Fast Marching method, and a 3D surface is acquired by the interpolation between minimal paths. A 3D level set algorithm is performed for segmentation using the acquired 3D surface as initialization. Although this approach may provide good results, the topology of the network is often problematic [4]. An implicit method is proposed in [4] for this issue. It segments the object by finding a 3D real function using

* Corresponding author.
E-mail addresses: ypan@sci.utah.edu (Y. Pan), wkjeong@seas.harvard.edu (W.-K. Jeong), whitaker@cs.utah.edu (R. Whitaker).

1077-3142/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2011.06.003

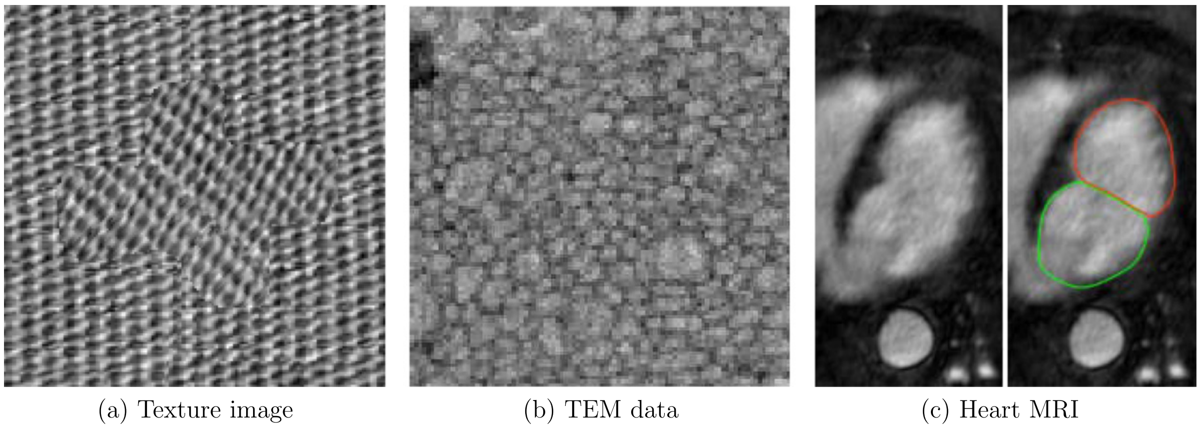

Fig. 1. Examples that conventional intensity-based image segmentation methods fail to work.

transport equations such that the network of minimal paths is contained in its zero level set.

These methods, however, utilize mostly gradient information. Image registration results between slices, which are very important for 3D segmentation, are not utilized. They may have problems when weak edges are present, e.g., in Fig. 1c. These methods are also computationally intense and no parallel computing techniques are applied. The method proposed in this paper, Markov surfaces, generalizes these methods by using region information in a probabilistic framework. It combines a user-assisted approach with an optimal path formulation that addresses a more general class of images and shapes without a predefined model. This method allows users to define surfaces or regions in 3D data and to follow object boundaries in a way that does not require any specific formulation of an *edge*. A parallel implementation on graphics processor is provided for efficient computation. The proposed framework is especially useful for segmentation problems where the objects of interest are elongated in a predefined direction, which occurs regularly in 3D medical images.

The paper is organized as follows. Details of the probabilistic framework are introduced in Section 2. Section 3 discusses implementation issues. Experimental results are shown in Section 4, followed by Section 5, where a summary is presented.

2. Proposed method

The proposed segmentation system consists of two parts: a preprocessing part, which establishes correspondences between slices by 2D image registration, and an interactive part, which finds the Markov surfaces that connect user defined regions. Once the mapping between slices are acquired, the user can select a start curve to compute cost or probability, for the entire input volume, of attaching every point to the initial conditions via a Markov chain. The user then selects an end curve, and the algorithm backtracks through the cost volume to create a set of curves, each of which has the highest probability of connecting the two sets. This process can be repeated until desired segmentation results are obtained.

2.1. A probabilistic formulation for elongated structures

The goal of this section is to create a method that allows users to quickly and interactively define features (curves or regions) on disparate slices of a 3D dataset and connect these regions to form 3D structures in a way that conforms the data. The strategy is to make it lightweight and general and thereby quickly applicable to a wide range of different applications and data types.

The proposed framework constructs the *most probable* paths between regions using a Markov chain model that incorporates the probability of correspondence between points on two different slices. Here we define the *i*th *slice* f_i of a volume $f(x,y,z)$ to be the 2D function defined by fixing one of the coordinates, so that we have $f_i(x,y) = f(x,y,i)$.

Denote a particular path $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ as having probability $P(\mathbf{W}) = P(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$, where $\mathbf{w}_i$ is a position of the path on the slice i of the input data. We model the path as a Markov chain [10], so that probability of each subsequent position along the path depends only on the previous position and probability. This gives

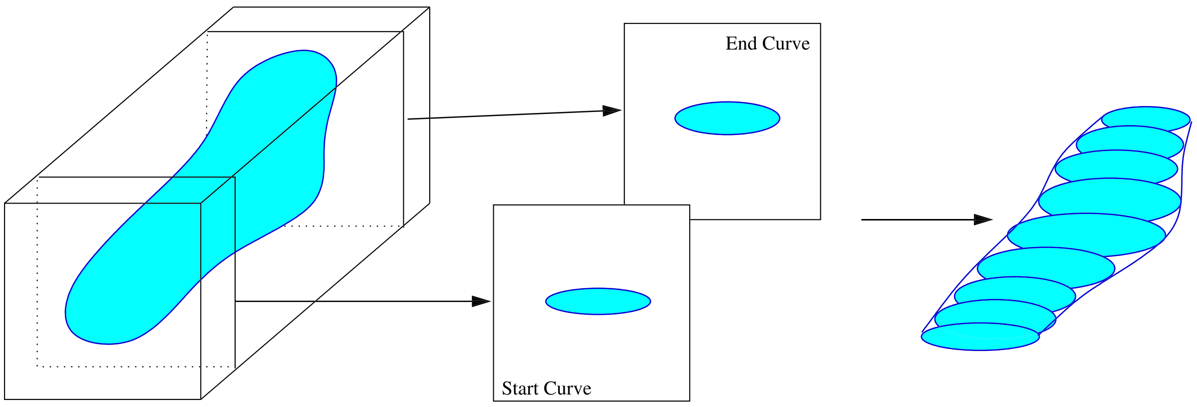

Fig. 2. Overview of the proposed 3D segmentation process using 2D tracking.

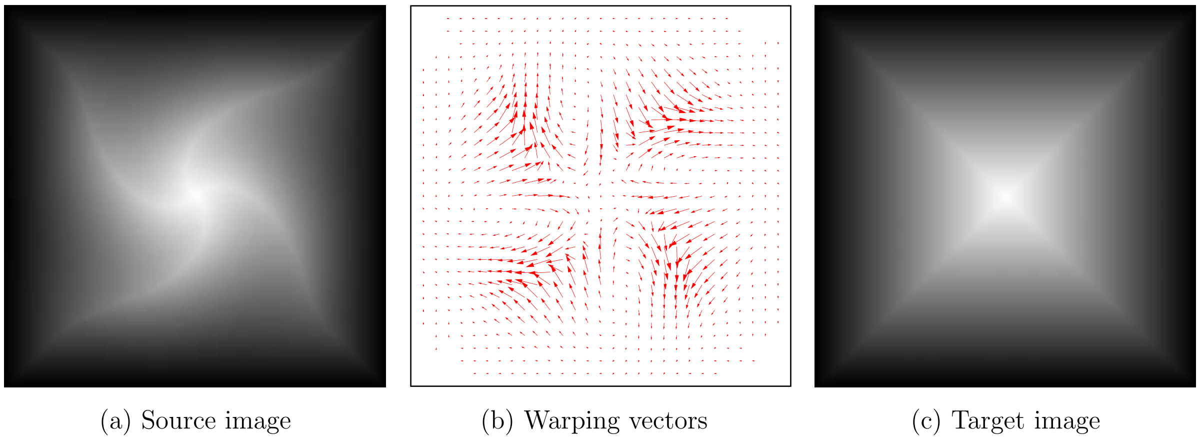

Fig. 3. Vector field for warping between two images. Warping the source image (a) with the vector field (b) gives the target image (c).

$$P(\mathbf{W}) = P(\mathbf{w}_1)\prod_{i=1}^{n-1} P(\mathbf{w}_{i+1}|\mathbf{w}_i) \tag{1}$$

The proposed strategy is to define the conditional probabilities in terms of a transition function from each pixel on one slice to every pixel on the next. Thus

$$P(\mathbf{w}_{i+1}|\mathbf{w}_i) = F(\mathbf{w}_{i+1}, \mathbf{w}_i), \tag{2}$$

where $F(\mathbf{w}_{i+1},\mathbf{w}_i)$ may be considered as weights on a directed graph that connects every pixel in one slice to every pixel in the next.

We define the initial probability $P(\mathbf{w}_1)$ in terms of a user-defined region, A (e.g., a starting curve). These probabilities could include some uncertainty around this curve, or alternatively, as in this paper, a binary mask:

$$P(\mathbf{w}_1) = \begin{cases} a > 0 & \mathbf{w}_1 \in A \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where for curves or points in a continuous domain, this would be, formally, a measure.

The path $\widetilde{\mathbf{W}}$ that maximizes $P(\mathbf{W})$ is defined:

$$\begin{aligned}\widetilde{\mathbf{W}} &= \operatorname{argmax}_{\mathbf{W}}[P(\mathbf{W})] \\ &= \operatorname{argmin}_{\mathbf{W}}\left[-\sum_{i=2}^{n}\log F(\mathbf{w}_i, \mathbf{w}_{i-1}) - \log P(\mathbf{w}_1)\right]\end{aligned} \tag{4}$$

where $-\log P(\mathbf{W})$ is referred as the *path cost* for the path $\mathbf{W}$.

A variation on Dijkstra's algorithm for dynamic programming, described in Section 2.4, is proposed to find the optimal path to every point in the volume. In practice, users define the start and end curves for an object on different slices in a volume, and the method quickly connects these regions using the most probable paths, as shown in Fig. 2. The probability of the paths are derived from a set of automatically determined correspondences, and thus the resulting surfaces follow the structure of the data.

2.2. Slice-to-slice correspondence estimation

The first step of the Markov surfaces is to find a dense set of correspondences between 2D slices in a 3D volume. There are a variety of ways that one could find such correspondences, such as patch correlations [11] or feature matching [12]. A deformable image registration method is applied here, which represents the correspondences as a smooth displacement field.

Let I denote a 2D image where $I(\mathbf{x}) : \Omega \mapsto \mathbb{R}$ and $\mathbf{x} \in \Omega \subset \mathbb{R}^2$, and I_i and I_{i+1} be two consecutive slices in the 3D volume. Define a correspondence from image I_i to I_{i+1} as a 2D transformation vector field $\mathbf{v}_{i,i+1}(\mathbf{x}) : \Omega \mapsto \mathbb{R}^2$ that maps each pixel in I_i to I_{i+1}.

Slice-to-slice correspondence estimation is achieved by minimizing the following the energy E:

$$E = \frac{1}{2}\int_{\mathbf{x}\in\Omega} (\widetilde{I}_i - I_{i+1})^2 + \alpha\|\nabla\mathbf{v}\|^2 \tag{5}$$

where $\widetilde{I}_i = I_i(\mathbf{x} + \mathbf{v}_{i,i+1}(\mathbf{x}))$, $I_{i+1} = I_{i+1}(\mathbf{x})$, $\mathbf{v} = \mathbf{v}_{i,i+1}(\mathbf{x})$, $\mathbf{x} \in \Omega$, and α is a constant parameter. The regularization term $\|\nabla\mathbf{v}\|^2$ helps to produce a smooth vector field and makes the problem well posed.

A gradient flow is used to compute $\mathbf{v}_{i,i+1}$, similar to [13,14]. The update equation for $\mathbf{v}$ is written as

$$\begin{aligned}\mathbf{v}^{k+1} &= \frac{1}{(I + \Delta t\alpha L)}\left[\mathbf{v}^k + \Delta t(I_{i+1} - \widetilde{I}_i^k)\nabla\widetilde{I}_i^k\right] \\ &= G\star\left[\mathbf{v}^k + \Delta t(I_{i+1} - \widetilde{I}_i^k)\nabla\widetilde{I}_i^k\right],\end{aligned} \tag{6}$$

where G is a Gaussian kernel of width $\sigma = \sqrt{2\alpha\Delta t}$.

Because the energy function E we want to minimize in this problem is not, generally, convex, the solution of the minimization converges to local minimum. Therefore, images with large deformations require better optimization strategies. To overcome such problems, we use cascading multigrid scheme (coarse to fine) with a 4-to-1 averaging D combined with a Gaussian smoothing kernel G (to eliminate aliasing effects) for down sampling the input images, and a 1-to-4 projection kernel U for up sampling the vector images. Fig. 3 shows an example of a vector field for warping between two images.

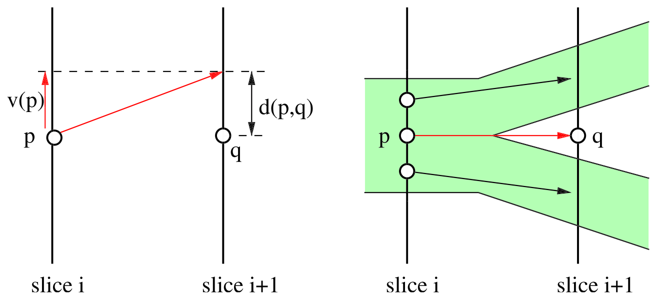

Fig. 4. Distance (left) and matching (right) metrics. $v(p)$ in the left image represents the movement of p in slice i towards slice $i+1$ from image registration. The intensity difference between p and q on the right is illustrated by different colors. By means of cost computation, p may be found to correspond to q with the help of distance metric although their intensities differ.

2.3. Slice-to-slice mapping probability

In this section we define the slice-to-slice mapping probability $F(\mathbf{p},\mathbf{q})$, defined in Eq. (2), which indicates the transition probability for a point $\mathbf{p}$ on slice i to a point $\mathbf{q}$ on slice $i+1$. A bilateral fall-off function F is applied based on two measures: the distance $d(\mathbf{p},\mathbf{q}) = |\mathbf{p} + \mathbf{v}_i(\mathbf{p}) - \mathbf{q}|$ from the correspondence given by the registration and the intensity difference $g(\mathbf{p},\mathbf{q}) = |f_i(\mathbf{p}) - f_{i+1}(\mathbf{q})|$ between the image values on the adjacent slices in the path. An illustration of these measures is provided in Fig. 4. Thus, the transition function is

$$F(\mathbf{p},\mathbf{q}) = \frac{1}{K_p}\exp\left(-\frac{d^2(\mathbf{p},\mathbf{q})^2}{2k_d^2}\right)\exp\left(-\frac{g(\mathbf{p},\mathbf{q})^2}{2k_g^2}\right) \tag{7}$$

where k_d and k_g are user-given parameters, and K_p is the normalization constant. The cost computation based on the probability function (7) is therefore the quadratic expression:

$$C^{\text{new}}(\mathbf{x}) = C_{n-1}(\tilde{\mathbf{x}}) + \log(K_p) + \frac{d^2(\tilde{\mathbf{x}},\mathbf{x})}{2k_d^2} + \frac{g^2(\tilde{\mathbf{x}},\mathbf{x})}{2k_g^2}, \tag{8}$$

where $C^{\text{new}}(\mathbf{x})$ represents the new cost on grid x computed from the grid $\tilde{\mathbf{x}}$ in the previous slice.

The right image in Fig. 4 illustrates the computation of shortest paths between slices by minimizing the cost in Eq. (8). The points above and below the point p in slice i are shown to correspond to points with similar intensities (illustrated by color) in slice $i+1$. The point p, however, may be found to correspond to the point q with different intensity. This is achieved by incorporating the distance metric to the cost function in Eq. (8). This demonstrates the effectiveness of image registration, which will be further explored in Section 4.

2.4. Shortest path cost computation

A variant of Dijkstra's algorithm [15] is applied here to compute the optimal path on a directed graph formulated in Eq. (4). The strategy is to compute a sequence of optimal paths to every pixel on each successive slice. Let $C_n(\mathbf{x},\mathbf{y},\mathbf{n})$ be the cost of the optimal path from the first slice to pixel $(\mathbf{x},\mathbf{y},\mathbf{n})$ on slice n. For every neighbor $(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{n}-1)$ of the pixel $(\mathbf{x},\mathbf{y},\mathbf{n}-1)$ in the slice $n-1$, the cost C^{new} is computed using $C^{\text{new}} = C_{n-1}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{n}-1) - \log(F(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{n}-1;\mathbf{x},\mathbf{y},\mathbf{n}))$, and the minimum cost among all the neighbors is taken as the minimum cost of the path $C_n(\mathbf{x},\mathbf{y},\mathbf{n})$.

This is described in pseudocode in Algorithm 1, where ρ_i represents the corresponding point in slice $i-1$ for the point x in slice i.

Algorithm 1. Algorithm for computing the optimal cost and path to each point in the volume.

```
1:Initialize $C_1$
2: for $i = 2$ to $n$ do
3:   for all $\mathbf{x} \in \Omega$ do
4:     $C_i(\mathbf{x}) = \infty$
5:     for all $\tilde{\mathbf{x}} \in \text{neighbor}(\mathbf{x})$ do
6:       $C^{\text{new}} = C_{i-1}(\tilde{\mathbf{x}}) - \log(F(\tilde{\mathbf{x}},\mathbf{x}))$
7:       if $C_i(\mathbf{x}) > C^{\text{new}}$ then
8:         $C_i(\mathbf{x}) = C^{\text{new}}$
9:       $\rho_i = \tilde{\mathbf{x}}$
10:     end if
11:     end for
12: end for
13: end for
```

The proposed algorithm computes cost values on each slice in sequence, from the starting to the ending slice. Because of the strict causal relationship and the parallel nature of the method, its implementation on parallel architectures, such as graphics processing units, is straightforward and gives a significant speed-up, allowing the Markov surfaces to be generated and visualized at interactive rates, immediately after the user has defined the starting region (and while they are selecting the ending region).

Curves connecting the regions are generated by finding a path for every point on the end curve that connects to the start curve. This is done by *backtracking* from the ending region through each previous pixel on the optimal path to the starting slice. That is, we recursively build the path by referring to each previous transition in the optimal path. Fig. 5 shows an example of backtracking from $\mathbf{w}_n$ on the end slice n through $\mathbf{W} = \{\mathbf{w}_i \in \Omega | i = 1,\ldots,n\}$ to slice 1, the start slice.

2.5. Bézier interpolation

One issue in the proposed framework is that paths may merge during backtracking [4]. In the illustration of Fig. 6a, for example, paths from P_{52} and P_{53} merge at slice S_3 during backtracking, and the path from P_{54} merges the path from P_{52} (and P_{53}) at slice S_1. In this case, paths from three points in the end curve in slice S_5 are backtracked to one point in the start curve in slice S_0. Therefore, backtracking from a smooth end curve (e.g., Fig. 6b) may result in a cluster of spatially disconnected points (e.g., Fig. 6c) seen from one slice. This feature makes surface extraction challenging [4], which means a surface such as the one in Fig. 10e is hard to acquire from the backtracking results without interpolation.

Bézier curve [16] is applied to model smooth curves from disconnected points in this case. A Bézier curve of order $\mathbf{n}$ is generated using $\mathbf{n}$ points $\{\mathbf{P}_i,\ i = 0,\ldots,\mathbf{n}\}$ (Fig. 6c),

$$B(t) = \sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^i\mathbf{P}_i, \quad t \in [0,1] \tag{9}$$

The order n may be specified by users depending on specific applications. A smooth curve (e.g., Fig. 6d) is achieved by applying Bézier curve to all the backtracking points (e.g., Fig. 6c). Every point in the smoothed curve will be used as a start point of a path and be backtracked to the next slice, where Bézier curve is applied again to the backtracking results. This process is repeated until the backtracking process has been done for all the slices.

2.6. Forward update after backtracking

A smooth curve is acquired in each image slice after Bézier interpolation is applied. However, the curve resulting from backtracking may be only a portion of the start curve in the first slice [4]. Fig. 7a illustrates this issue, where the portion of the start curve between $P01$ and $P04$ is not reached by any path from backtracking. However, this portion of the start curve should be reached based on the minimization of the cost function in Eq. (8). This is illustrated in Fig. 7b, where paths inside the area specified by $P01$, $P04$ and $P51$ are expected. A forward update mechanism is proposed to handle this issue, as illustrated in Fig. 7.

Once the backtracking stage is finished, we record the unmatched points between $P01$ and $P04$ in slice $S0$. Then slice $S1$ is searched using Algorithm 1 to find the paths corresponding to those unmatched points in slice $S0$. The magenta curve in Fig. 7c shows the searching results in slice $S1$. Bézier interpolation is then applied in slice $S1$ to generate smooth curves. Fig. 7d is achieved after Bézier interpolation. The same process is applied to all other

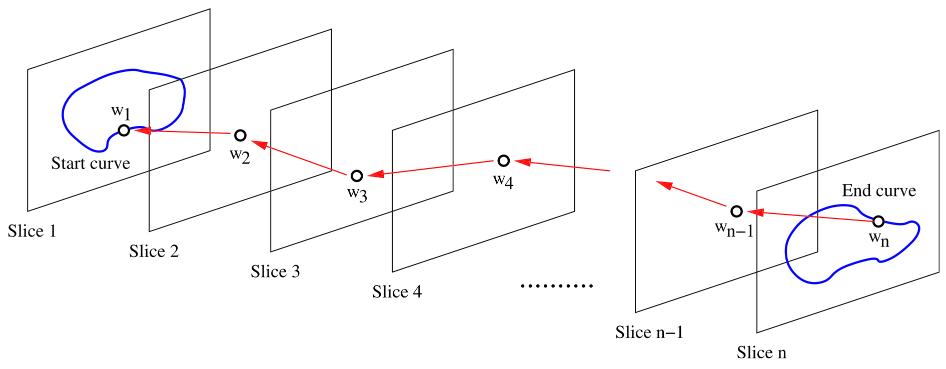

Fig. 5. Backtracking from a point $\mathbf{w}_n$ on the end curve to the start curve.

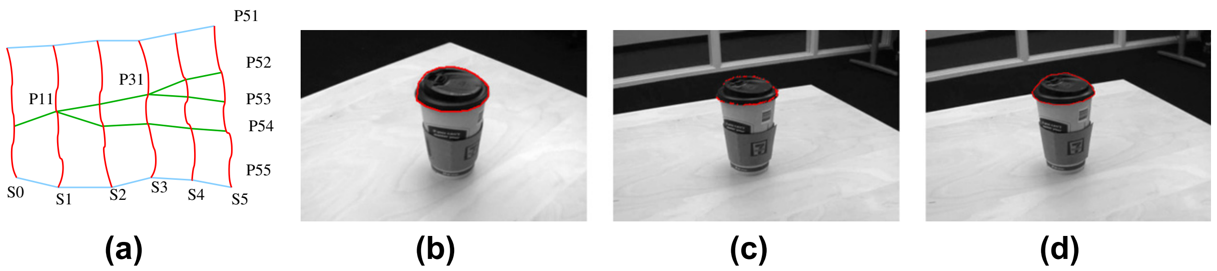

Fig. 6. Illustration of path merging and Bézier interpolation. (a) Path merging during backtracking. (b) Smooth end curve. (c) Spatially disconnected points seen in a slice, which results from path merging during backtracking. (d) Curve acquired from Bézier interpolation in (c).

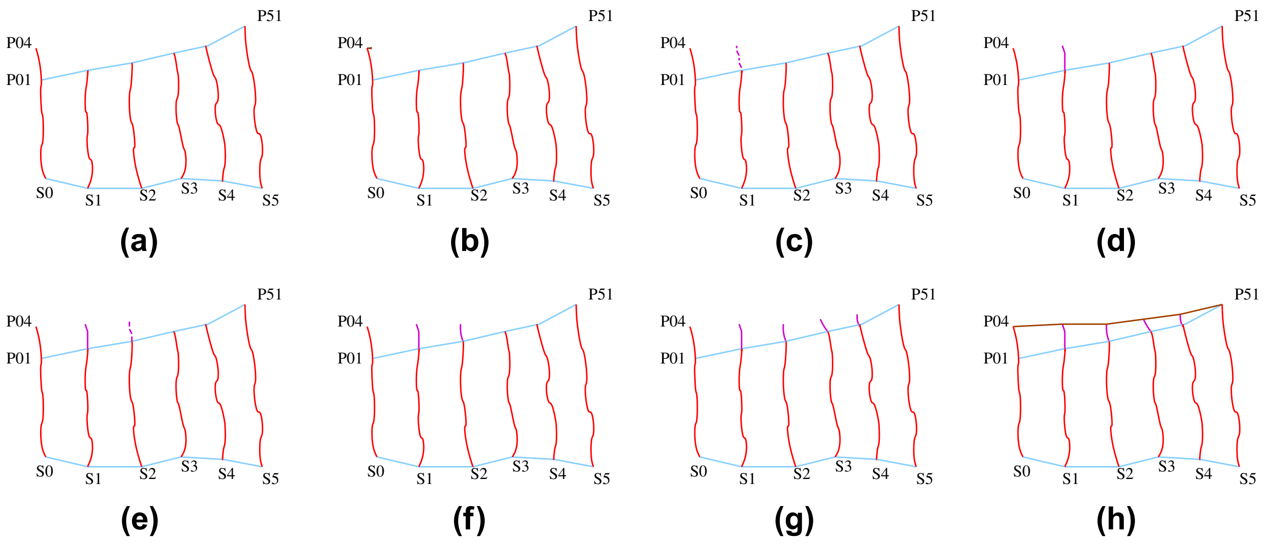

Fig. 7. Illustration of forward update after backtracking. (a) Start curve may be reached partially during backtracking. (b) Results expected from cost minimization. (c) Magenta curve in slice $S1$ corresponds to the unmatched points in slice $S0$. (d) Bézier interpolation is applied in slice $S1$ for smooth curves. (e) Magenta curve in slice $S2$ corresponds to the unmatched points in slice $S1$. (f) Bézier interpolation is applied in slice $S2$ for smooth curves. (g) Magenta curves after applying the forward process on all slices. (h) Paths after the the forward process, which is more reasonable than (a).

slices $S2$, $S3$, and $S4$, as shown in Fig. 7e–g. After the forward update process, we generate the results in Fig. 7h, which are more reasonable than those in Fig. 7a.

The forward update mechanism may be applied whether the user-specified curves are open or closed. It can improve the performance in both cases. From our experiences, this mechanism is required for open curves, while it is not necessary for closed curves. This may be explained by the fact that Bézier interpolation can generate more regularization for closed curves than open curves.

3. Implementation issues

Markov surfaces present two computational problems: nonrigid image registration and the dense optimal cost computation. Because those problems are inherently parallel, we have implemented them on a graphics processing unit (GPU) to make the system interactive. Efficient computation on the GPU entails reusing memory in the access of overlapping neighborhood regions and the reduction of memory latency for random access. For this task, we use texture hardware on the current GPUs because texture memory is cached and interpolation is done for free by hardware. For example, the proposed semi-implicit nonrigid image registration method iteratively updates the vector field and smoothes it until it converges to a steady state.

To optimize the displacement field, we need to sample the pixel values on *arbitrary* locations (unlike, for example, optic flow) in the transformed image. We can do this efficiently by using the GPU's hardware interpolation function. For shortest path cost computation, we need to collect a set of pixels from the previous slice per every pixel on the current slice, and then we find the pixel having the minimum cost among them. This is a memory bound process (access to memory is a bottleneck). However, the searching region for a given pixel is a spatially coherent rectangle region on a 2D slice, and is largely overlapped to the neighbor pixel's search region. Therefore, we can get a high cache hit rate and significantly reduce the running time by using texture memory of the GPU. All of these strategies are part of the implementation and allow us to get the significant computational speed up (relative to CPU implementations) and interactive computation times for this approach.

Table 1
Comparison of running times for cost computation.

	Synthetic texture ($150 \times 150 \times 50$)	Seismic ($301 \times 111 \times 32$)	MRI ($640 \times 460 \times 16$)
CPU time	4.86	21.5	596
GPU time	0.25	0.46	3.8
Search width	1.8	4.3	16
Speedup	19	46	156

4. Experimental results

Experimental results are provided in this section. The proposed segmentation system is implemented on a Windows XP PC equipped with an Intel Core 2 Duo 2.4 GHz CPU, 4 GB main memory, and an NVIDIA Geforce 8800 GT graphics card.

Image registration and cost computation are time consuming processes, and they cannot be done in real-time on a conventional computer. For example, 2D registration of a 300×300 image (e.g., Fig. 3) takes about 28 s on the CPU after 600 iterations. The same registration can be done only in 0.7 s on the GPU. Slice-to-slice registration for a large 3D volume could easily take a few minutes even on the GPU, but it needs to be computed only once for each volume along each direction of interest. Thus, we consider a preprocessing step, done just before the interactive segmentation process.

Table 1 compares the running times (in seconds) of computation on the CPU and the GPU on one synthetic and two real 3D datasets. The MRI volume is about four times larger than the other two volumes. The most important factor in computation time is the size in the cost computation, because the algorithm complexity of cost computation is $O(kN)$ where k is the size of neighbor search and N is the size of input data (i.e., the number of voxels). Also, because of the benefit of using local memory (or texture cache [17]), the neighbor search size k affects the running time less significantly in the GPU version. Thus, the speed gain associated with

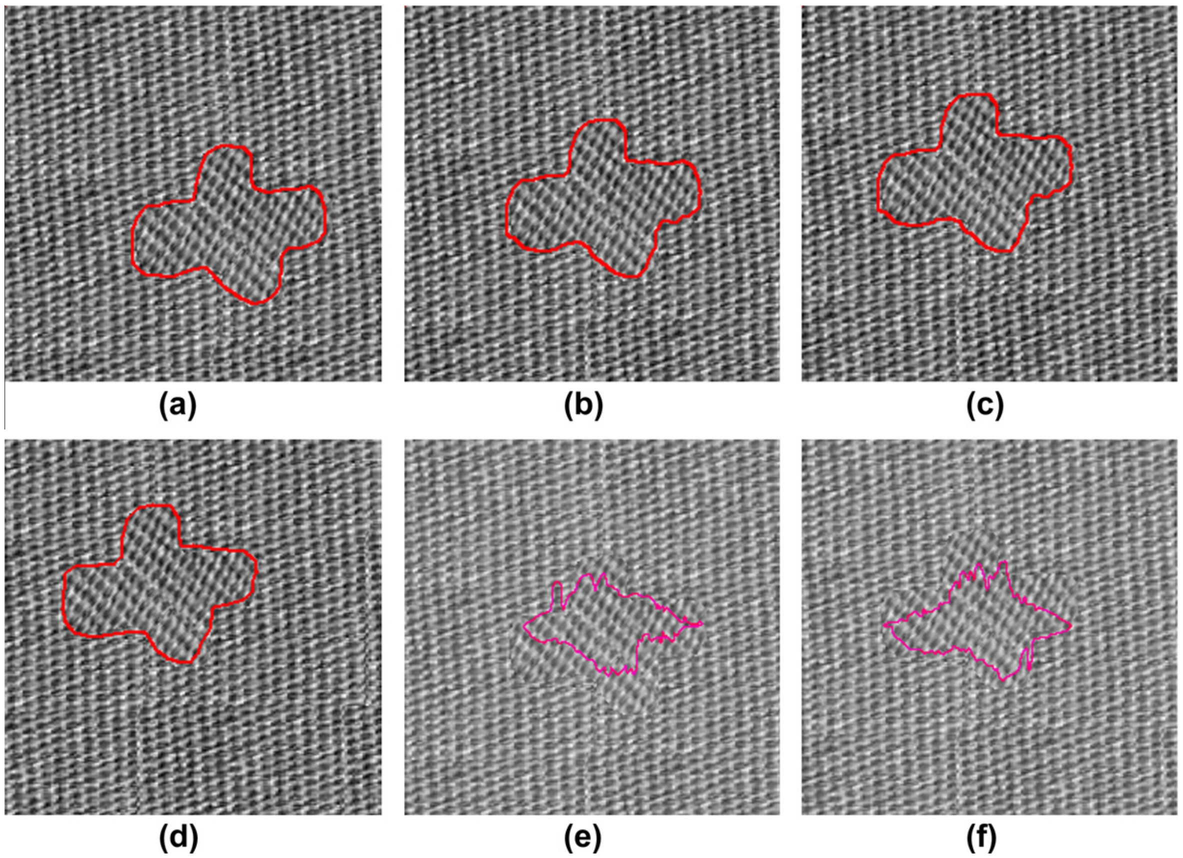

Fig. 8. Segmenting a 3D texture. (a) Start contour in the 1st slice. Contours for the Markov surfaces for slices 17 (b) and 34 (c), and the ending contour (d), on the 50th slice. (e) Segmentation results for slice 17 using the turtle segmentation method in [6]. (f) Segmentation results for slice 34 using the turtle segmentation method in [6]. The texture pattern is not segmented properly.

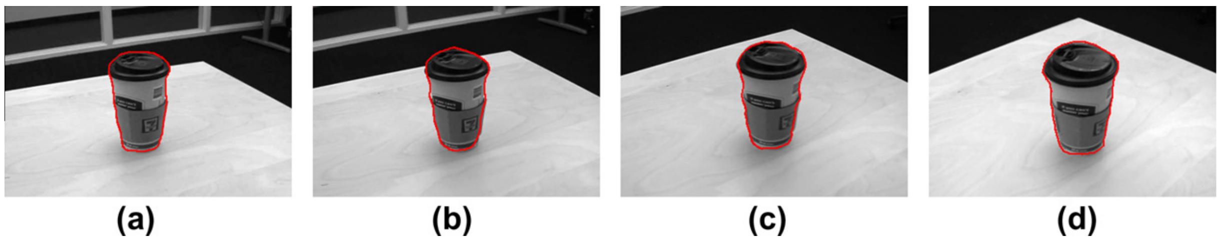
Fig. 9. Segmenting video data with nonstationary objects and background: (a) a user defined initial contour, (b and c) intermediate contours from the Markov surface, and (d) the user-defined ending contour on image 50 from the sequence.

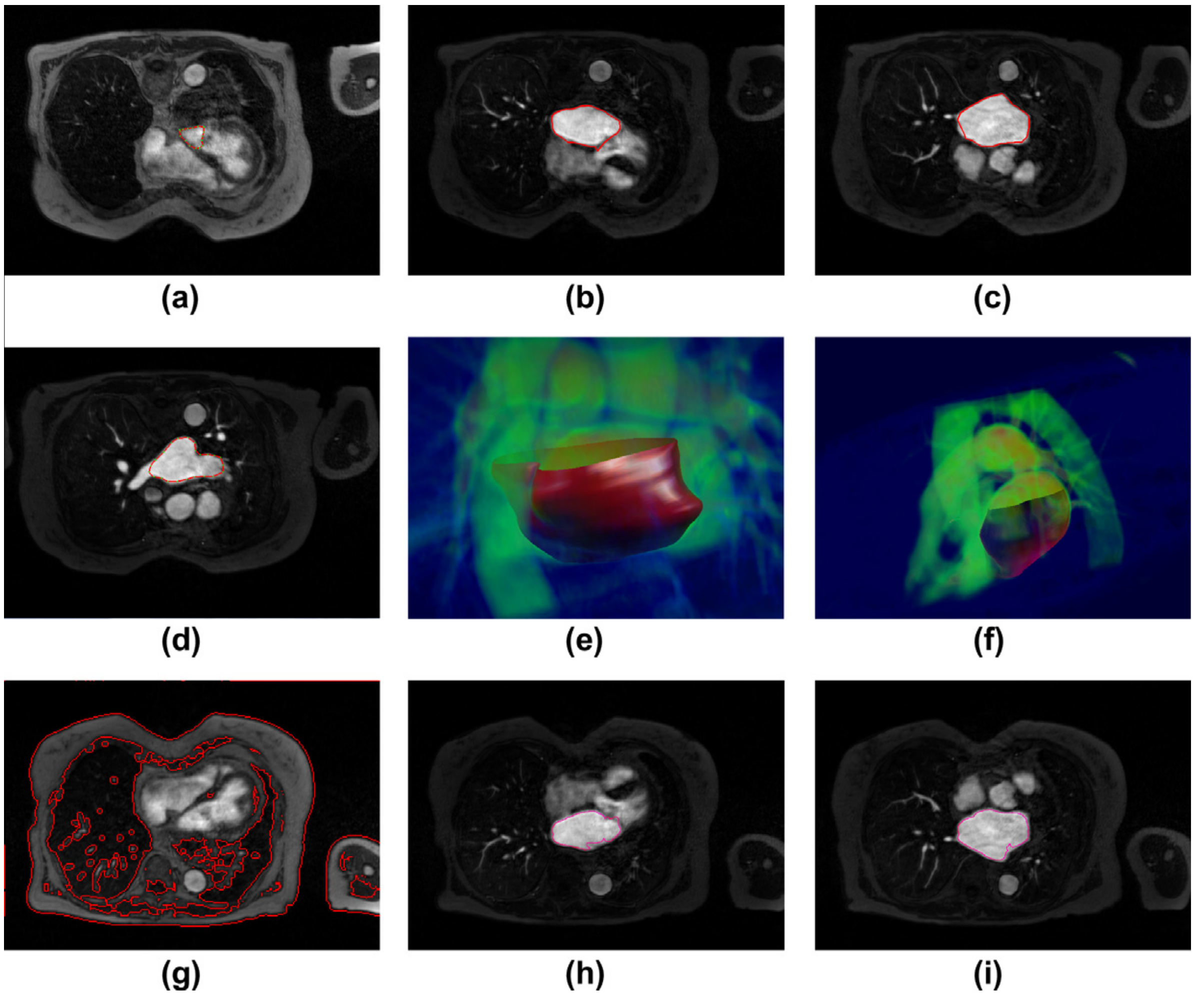
Fig. 10. Segmenting the left atrium from MRI: (a) Starting contour, contours from the Markov surface for slices 6 (b) and 12 (c), the ending contour (d) on slice 18, and (e)–(f) 3D rendering of the Markov surface with a volume rendering of the original MRI data for context. (g) Chan-Vese level set results for slice 2. The area of interest is not segmented. (h) Interactive level set results using ITK-SNAP [9] for slice 6. (i) Interactive level set results using ITK-SNAP [9] for slice 12.

the GPU implementation increases proportionally with neighbor search size.

Results on real and synthetic images demonstrate the effectiveness of the method. Each dataset has 30–50 slices, and intermediate results shows the results in the middle slices.

Fig. 8 shows an example of nontrivial feature segmentation, i.e., tracking of the boundary of an object in a complex texture background. Because the inside of the moving object and the background regions are filled with the same texture patterns, it is difficult to extract the boundary of the object using conventional edge detectors. As shown in the images, the object is segmented reasonably well through a series of slices using the proposed method.

For comparison, the turtle segmentation method [6,18] is applied to the same texture data. Four contours are specified by users for 3D segmentation in this experiment. These contours include the start contour in the 1st slice as in Fig. 8a, the end contour in the 50th slice as in Fig. 8d, an intermediate contour in the 25th slice, and a contour in the plane orthogonal to the planes of the first three contours. Fig. 8e and f show the segmentation results on slice 17 and slice 34. It can be seen that these results are not appropriate enough, and more user interactions and more time

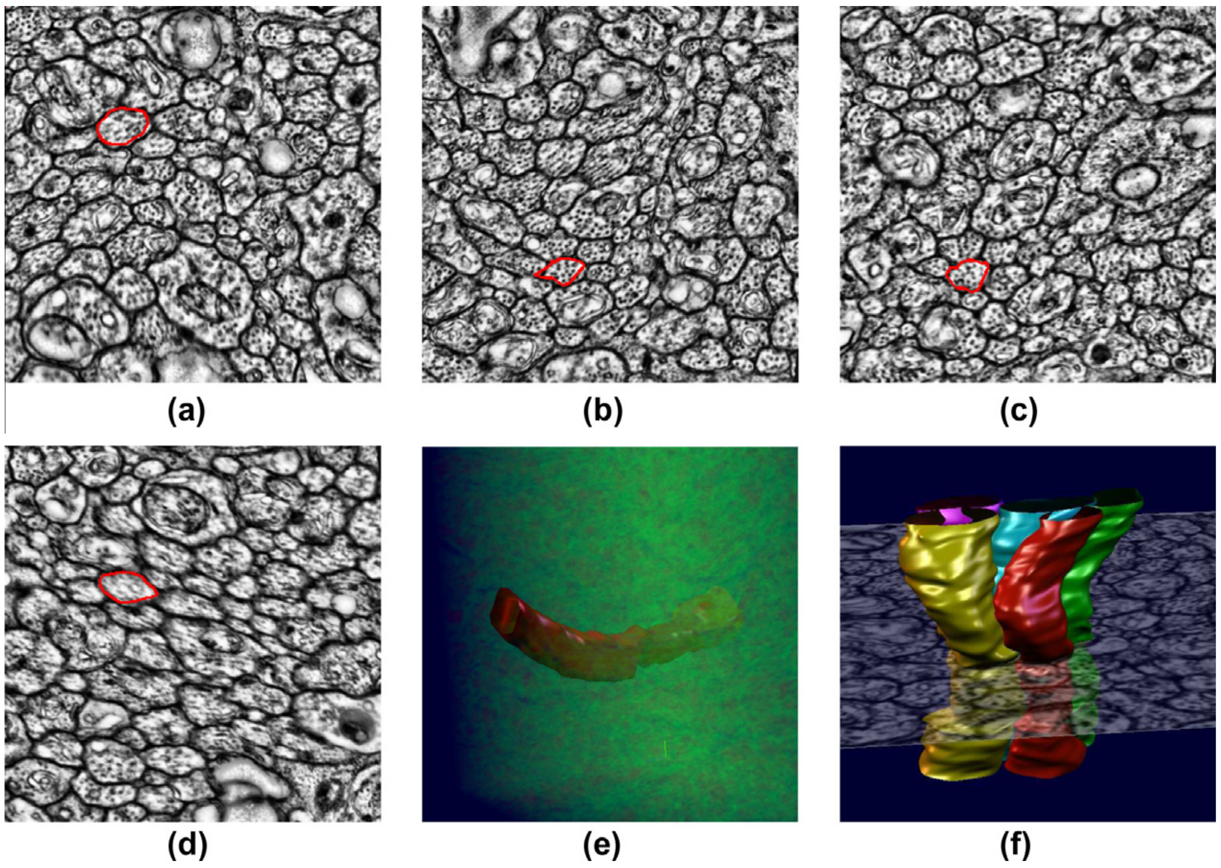

Fig. 11. Segmenting serial section microscopy data: (a) User-defined contour on the first slice, (b and c) intermediate contours from the Markov surface, (d) the user-defined ending contour, (e) 3D surface rendering of the Markov surface with volume rendering of the original data for context, and (f) multiple cells are segmented with less than 2 minutes of user interaction.

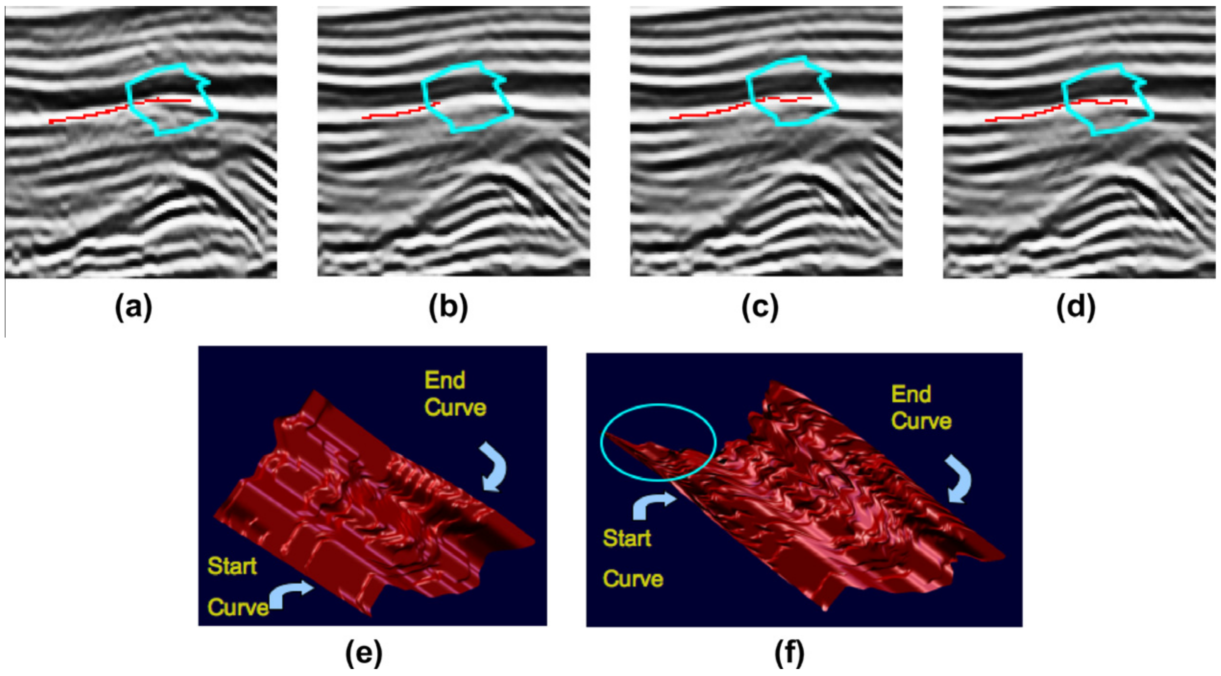

Fig. 12. Illustration of forward update after backtracking. (a) Start curve in slice 1 of the seismic data. (b) Backtracking results for slice 1. Only part of the start curve is reached. (c) Add in slice 2 the corresponding points to the unmatched points in slice 1. (d) Results in slice 2 after Bézier interpolation. (e) Volume rendering of the paths with no forward update. (f) Volume rendering of the paths after forward update.

are required to refine the segmentation. In comparison, our proposed method requires less user interaction and generates better results, because of the utilization of both the distance metric from image registration and the intensity metric.

Fig. 9 shows results for video segmentation, where the time series is treated as a set of slices in the proposed framework. In this example, the camera moves and the cup, during the sequence, occludes objects with similar intensities. Thus, this is not a problem which is amenable to simple motion tracking, intensity thresholding, or edge detection.

Fig. 10 shows the start and end curves for the MRI cardiac data set (top and bottom of heart, respectively) and several intermediate curves that are found as Markov surfaces. Note that the boundary of the heart wall expands abruptly at the beginning (top row), but the proposed cost method can capture the movement reasonably well. The right images in Fig. 10 show surface renderings of the Markov surface for the left atrium combined with a volume rendering, with user-defined transfer functions, of the original MRI data for context. A comparison with the Chan-Vese level set method [7] is provided for slice 2, with Fig. 10g showing the segmentation results. It can be seen that the automatic level set method with no user control is not able to segment the object of interested in this case. Fig. 10h and Fig. 10i show the segmentation results using the interactive level set method proposed in [8] by means of the ITK-SNAP implementation [9]. These results segment most of the heart, but the boundaries are irregular compared to the results in Fig. 10b and c. Moreover, during the segmentation process using ITK-SNAP, Users need to carefully choose initial curves, specify proper parameters, and stop the curve evolution. These requirements make it hard for user interactions. In comparison, the Markov surfaces are more straightforward for user interaction.

Fig. 11 shows similar results for tracing cells in serial section microscopy data. In this example, multiple axons are traced using Markov surfaces. Tracking axons is difficult; some axons move abruptly from one slice to another, new axons appear, or some axons suddenly disappear. Fig. 11 upper image shows tracking of a single axon through multiple slices, and lower image is the 3D rendering of the surface of five axons created using the proposed method.

Fig. 12 shows the effects for the forward update process on real seismic data. Fig. 12b shows that the curve from backtracking in slice 1 corresponds to only a small portion of the start curve in Fig. 12a. The points enclosed in cyan curve in Fig. 12c are found in slice 2 via cost minimization, which correspond to the unmatched points of the start curve in slice 1. A smooth curve is acquired after Bézier interpolation is applied, as shown in Fig. 12d. This forward update process is repeated from the start slice to the end slice for final segmentation results. Fig. 12e shows the volume rendering of paths with no forward update, while Fig. 12f show the volume rendering of paths after forward update. The portion enclosed in the blue cycle in Fig. 12f shows the effects of forward update. By comparison we can see that the forward update process generates more reasonable results with the paths reaching all of the start curve.

5. Summary

Despite many significant advances in machine vision, there remains a need for simple, general tools that allow users to quickly segment 3D data sets. This paper addresses a user-assisted segmentation method, Markov surfaces, for elongated structures in 3D images. Markov surfaces are based on a probabilistic framework that finds the optimal paths that connect user-defined regions. Computationally demanding components, such as non-rigid image registration and path computation, are implemented on the GPU, resulting in an interactive technique. Experimental results show that the proposed Markov surfaces may be able to segment elongated structures accurately and efficiently with straightforward user interactions.

One limitation of the Markov surfaces is that the elongated object must be aligned along a specific direction. Applying the proposed method to arbitrarily curved structures will be the future work. Another shortcoming is that the individual, one-dimensional, paths are optimized independently between slices. Future work will consider the joint optimization of a set of interacting paths in the 3D volume, in order to guarantee complete surfaces that connect the user-defined boundary conditions. Quantitative performance evaluation in terms of repeatability, accuracy and efficiency will be explored in the future too.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.cviu.2011.06.003.

References

[1] W.A. Barrett, E.N. Mortensen, Interactive live-wire boundary extraction, Medical Image Analysis 1 (1997) 331–341.
[2] A.X. Falcao, J.K. Udupa, S. Samarasekera, S. Sharma, User-steered image segmentation paradigms: live wire and live lane, Graphical Models and Image Processing 60 (4) (1998) 233–260.
[3] R. Ardon, L.D. Cohen, Fast constrained surface extraction by minimal paths, International Journal of Computer Vision 69 (1) (2006) 127–136.
[4] R. Ardon, L.D. Cohen, A. Yezzi, Fast surface segmentation guided by user input using implicit extension of minimal paths, Journal of Mathematical Imaging and Vision 25 (3) (2006) 289–305. doi:dx.doi.org/10.1007/s10851-006-9641-9.
[5] A. Schenk, G. Prause, H.-O. Peitgen, Efficient semiautomatic segmentation of 3D objects in medical images, in: MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer-Verlag, London, UK, 2000, pp. 186–195.
[6] M. Poon, G. Hamarneh, R. Abugharbieh, Efficient interactive 3D live wire segmentation of complex objects with arbitrary topology, Computerized Medical Imaging and Graphics 32 (2008) 639–650.
[7] T. Chan, L. Vese, Active contour without edges, IEEE Transactions on Image Processing 10 (2) (2001) 266–277.
[8] P. Yushkevich, J. Piven, H. Hazlett, R. Smith, S. Ho, J. Gee, G. Gerig, User-guided 3D active contour segmentation of anatomical structures: significantly improved efficiency and reliability, NeuroImage 31 (2006) 1116–1128.
[9] http://www.itksnap.org/pmwiki/pmwiki.php.
[10] S.P. Meyn, R.L. Tweedie, Markov Chains and Stochastic Stability, Springer-Verlag, 1993.
[11] Z. Engin, M. Lim, A. Bharath, Gradient field correlation for keypoint correspondence, in: Proceedings of International Conference on Image Processing, 2007, pp. II: 481–484.
[12] G. Wen, J. Lv, W. Yu, A high-performance feature-matching method for image registration by combining spatial and similarity information, IEEE Transactions on Geoscience and Remote Sensing 46 (2008) 1266–1277.
[13] P. Anandan, A computational framework and an algorithm for the measurement of visual motion, Journal on Computer Vision 2 (1989) 283–310.
[14] U. Clarenz, M. Droske, M. Rumpf, Towards fast non–rigid registration, in: Inverse Problems, Image Analysis and Medical Imaging, AMS Special Session Interaction of Inverse Problems and Image Analysis, vol. 313, AMS, 2002, pp. 67–84.
[15] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.
[16] J.D. Foley et al., Computer Graphics: Principles and Practice in C, second ed., Addison Wesley, 1992.
[17] H. Nguyen, GPU Gems 3, Addison-Wesley Professional, 2007.
[18] http://www.turtleseg.org.