Markov Surfaces: A Probabilistic Framework for User-Assisted Three-Dimensional Image Segmentation

Yongsheng Pan[1], Won-Ki Jeong[2], and Ross Whitaker[3]

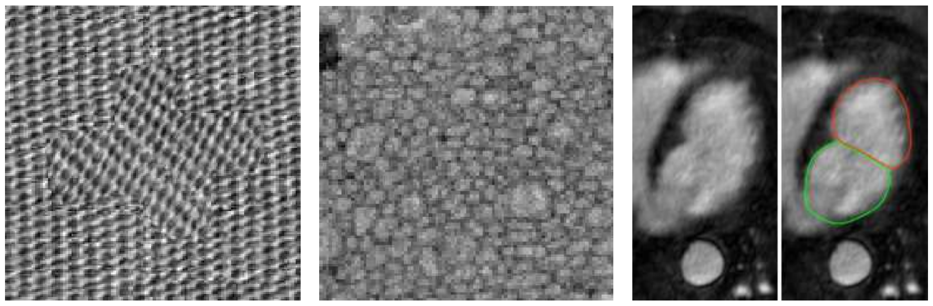
[1] Scientific Computing and Imaging Institute, University of Utah,
ypan@sci.utah.edu
[2] Initiative in Innovative Computing, Harvard University,
wkjeong@seas.harvard.edu
[3] Scientific Computing and Imaging Institute, University of Utah,
whitaker@cs.utah.edu

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. Bezier 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 can not be fully automatic. Several examples, especially when the boundary of the object is not clearly separable using intensity differences, are shown in Figure 1. In Figure 1 (a), 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 Figure 1 (b) is hard to be segmented, even if the regions there can be distinguished by some combination of texture and dark boundaries. Figure 1 (c) 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 usually hard to get good segmentation results without user interaction, especially when the boundary of the object is not clearly separable using intensity differences. Semiautomatic segmentation using user interaction, therefore, seems necessary in these cases.



(a) Texture image (b) TEM data (c) Heart MRI

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

Several methods [1] [2] [3] [4] [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* improved the efficiency of this method using live lane, and Schenk *et al* extended the live wire method to 3D based on shape-based interpolation and optimization. Both methods need users to interactively specify points for segmentation.

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

All these methods, however, rely only on the gradient information. They may have problems when weak edges are present, e.g., in Figure 1 (c). The method proposed in this paper, Markov surfaces, generalizes these methods by using region information in a probabilistic framework. It 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*. This method 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 (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 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 ith *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 [6], so that probability of each subsequent position along the path depends only on the previous position and probability. This gives

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

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:

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

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 Figure 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.

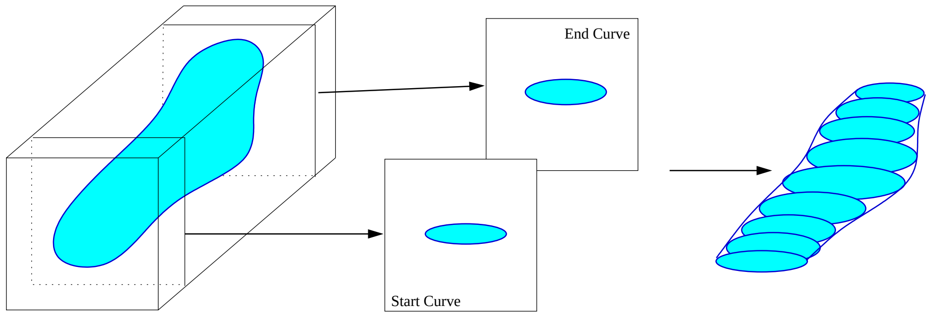


Fig. 2. Overview of the proposed 3D segmentation process using 2D tracking.

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 [7] or feature matching [8].

A deformable image registration method is applied here, which represents the correspondences as a smooth displacement field.

Let I denote a 2D image where $I(\mathbf{x}) : \varOmega \mapsto \mathbb{R}$ and $\mathbf{x} \in \varOmega \subset \mathbb{R}^2$, 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 $\mathbf{v}_{i,i+1}(\mathbf{x}) : \varOmega \mapsto \mathbb{R}^2$ that maps each pixel in I_i to I_{i+1}.

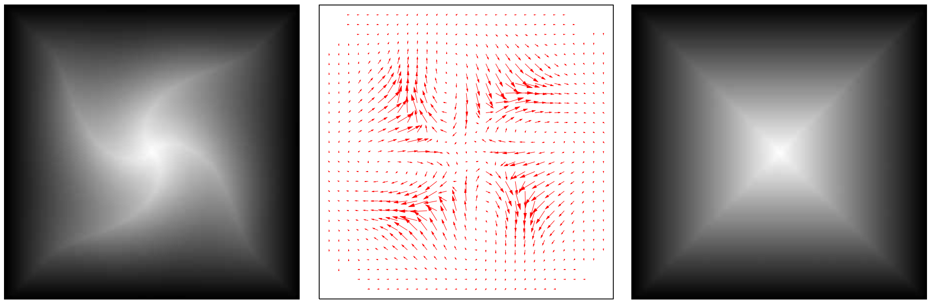

(a) Source image (b) Warping vectors (c) Target image

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

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

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

where $\widetilde{I}_i = I_i(\mathbf{x} + \mathbf{v}_{i,i+1}(\mathbf{x}))$, $I_{i+1} = I_{i+1}(\mathbf{x})$, $\mathbf{v} = \mathbf{v}_{i,i+1}(\mathbf{x})$, $\mathbf{x} \in \varOmega$, 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 [9, 10]. The update equation for $\mathbf{v}$ is written as

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

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

Because the energy function E we want to minimize in this problem is not, generally, convex, the solution of the minimization converges to local minima. 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. Figure 3 shows an example of a vector field for warping between two images.

A cascading multigrid scheme (coarse to fine) is applied in the implementation to handle large deformations. A 4-to-1 averaging combined with a Gaussian smoothing kernel G is utilized to downsample and process the input images, followed by a 1-to-4 projection kernel to upsample the vector images.

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. Thus, the transition function is

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

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:

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

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

2.4 Shortest Path Cost Computation

A variant of Dijkstra's algorithm [11] is applied here to compute the optimal path on a directed graph formulated in Equation 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 $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \mathbf{n}-1)$ of the pixel $(\mathbf{x}, \mathbf{y}, \mathbf{n}-1)$ in the slice $n-1$, the cost C_{new} is computed using $C^{new} = C_{n-1}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \mathbf{n}-1) - \log(F(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \mathbf{n}-1; \mathbf{x}, \mathbf{y}, \mathbf{n}))$, and the minimum cost among all the neighbors is taken as the minimum cost of the path $C_n(\mathbf{x}, \mathbf{y}, \mathbf{n})$.

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. Figure 4 shows an example of backtracking from $\mathbf{w}_n$ on the end slice n through $\mathbf{W} = \{\mathbf{w}_i \in \Omega | i = 1, ..., n\}$ to slice 1, the start slice.

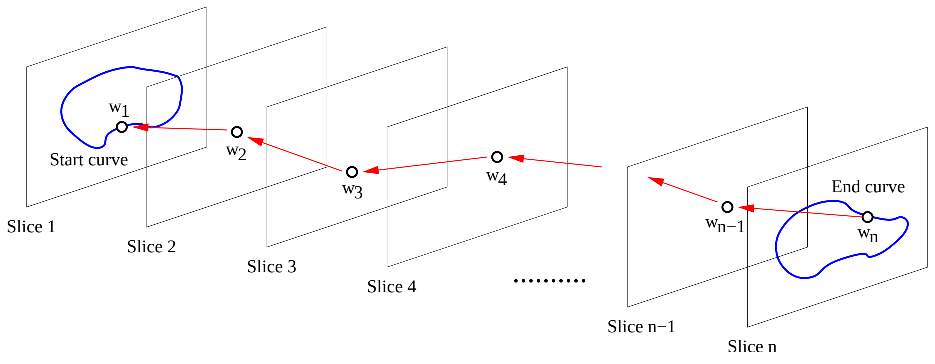


Fig. 4. Backtracking from a point $\mathbf{w}_n$ on the end curve to the start curve.

2.5 Path Interpolation and Forward Update

Paths may merge during backtracking [4]. Path merging generates separate points, which makes surface extraction challenging. A path interpolation method using B*é*zier curve [12] is applied to generate a continuous curve during backtracking. B*é*zier curve interpolation regularizes the extracted surface by dynamically adjusting paths to formulate a smooth curve in each backtracking slice. During regularization new paths are added; existing paths are adjusted; abnormal paths are removed. Curve smoothing using kernel averaging may be necessary before interpolation, especially for closed start curves.

The backtracking paths may only reach a small portion of the start curve [4], especially when the specified start curve is open. In this case we record the unmatched points in the start curve, and add their corresponding points to the backtracking curve in the next slice. Each unmatched point in the start curve may have multiple corresponding points. Only the closest one to the backtracking curve in the next slice is selected. This process is repeated from the start slice to the end slice, and B*é*zier curve interpolation is applied again for continuous curves.

3 Implementation Issues

3.1 Subgrid Cost Computation

The cost computation above assumes that the search on each previous slice is limited to pixel values, and thus paths are limited to the grid, which can result in aliasing and inaccuracies. Here we propose a more accurate subgrid method that solves for continuous locations at each slice using a linear interpolation of the intensity, cost, and correspondence positions from the previous slice. For this interpolation we divide each quadrangle of pixels into four triangles (by adding a new vertex with a value that is the average of its neighbors, as in Figure 5), and interpolate the necessary quantities on these triangles using barycentric

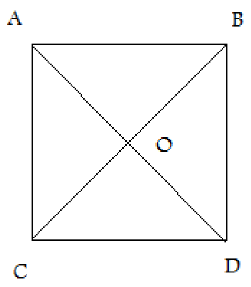


Fig. 5. Illustration for subgrid computation: the algorithm relies on linear interpolation on a set of 4 triangles for each set of 4 adjacent pixels on a slice.

coordinates. The optimal cost is computed for this analytic quantity over all four triangles on every quadrilateral contained in the corresponding neighborhood on the previous slice. The optimal cost and position for each pixel are those associated with the minimum over all these triangles in the previous slice. If the *log* transition probabilities are quadratic in position, cost, and intensity (as they are in this paper), the optimal cost on each triangle has a closed form solution, otherwise optimal costs must be obtained through some root trapping method.

3.2 Acceleration using Graphics Processors

Nonrigid image registration and optimal cost computation in the proposed method are highly parallelizable. They are therefore implemented on a graphics processing unit (GPU). 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. In this way we can get a high cache hit rate and significantly reduce the running time by using texture memory of the GPU.

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., Figure 3) takes about 28 seconds on the CPU after 600 iterations. The same registration can be done only in 0.7 second 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 [13]), the neighbor search size k affects the running time less significantly in the GPU version. Thus, the speed gain associated with the GPU implementation increases proportionally with neighbor search size.

	Synthetic ($150 \times 150 \times 50$)	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

Table 1. Comparison of running times for cost computation.

Fig. 6 illustrates the effects of B*é*zier interpolation and forward update. Backtracking results in separate points in Fig. 6(b) from the end curve in Fig. 6(a). B*é*zier interpolation of these points provides continuous contour in Fig. 6(c). Forward update is performed when the start curve in Fig. 6(d) is partly reached in Fig. 6(e). Fig. 6(f) shows the results from forward update in the next slice by adding those points corresponding to the unmatched points in the start curve. Fig. 6(g) shows the final results by applying B*é*zier interpolation.

Results on real and synthetic images demonstrate the effectiveness of the method. These results are better understood by referring to the accompanying video supplements. Each dataset has 30-50 slices, and intermediate results shows the results in the middle slices. Fig. 7 shows the segmentation results of a synthetic 3D texture and the video tracking results of a cup in real application. Fig. 8 displays the effects of the proposed method on real medical images. Volume renderings of the Markov surfaces are provided to show the segmented path.

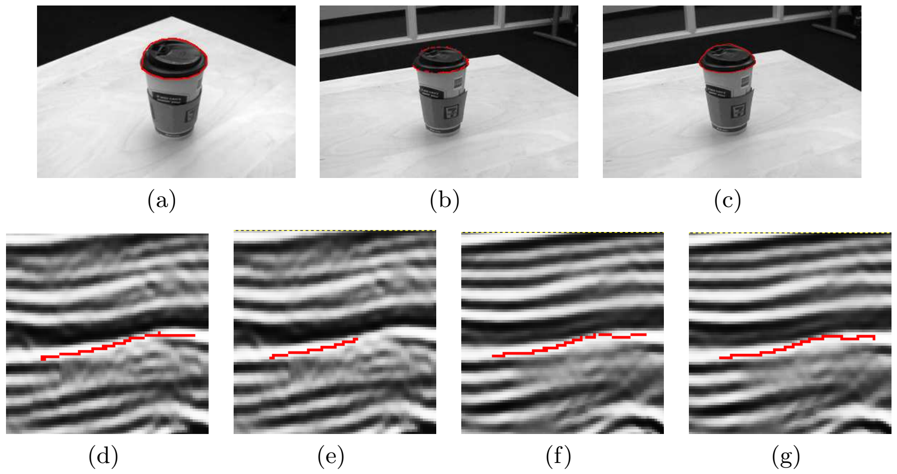


Fig. 6. Illustration of Bezier interpolation and forward update.

5 Summary

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 nonrigid image registration and cost/path computation, are implemented on the GPU, resulting in an interactive technique.

References

1. Barrett, W.A., Mortensen, E.N.: Interactive live-wire boundary extraction. Medical Image Analysis **1** (1997) 331–341
2. Falcao, A.X., Udupa, J.K., Samarasekera, S., Sharma, S.: User-steered image segmentation paradigms: Live wire and live lane. Graphical Models and Image Processing **60**(4) (July 1998) 233–260
3. Ardon, R., Cohen, L.D.: Fast constrained surface extraction by minimal paths. International Journal of Computer Vision **69**(1) (2006) 127–136
4. Ardon, R., Cohen, L.D., Yezzi, A.: Fast surface segmentation guided by user input using implicit extension of minimal paths. Journal of Mathematical Imaging and Vision **25**(3) (2006) 289–305
5.
6. Meyn, S.P., L.Tweedie, R.: Markov Chains and Stochastic Stability. Springer-Verlag (1993)
7. Engin, Z., Lim, M., Bharath, A.: Gradient field correlation for keypoint correspondence. (2007) II: 481–484
8. Wen, G., Lv, J., Yu, W.: A high-performance feature-matching method for image registration by combining spatial and similarity information. IEEE Trans. on Geoscience and remote sensing **46** (2008) 1266–1277

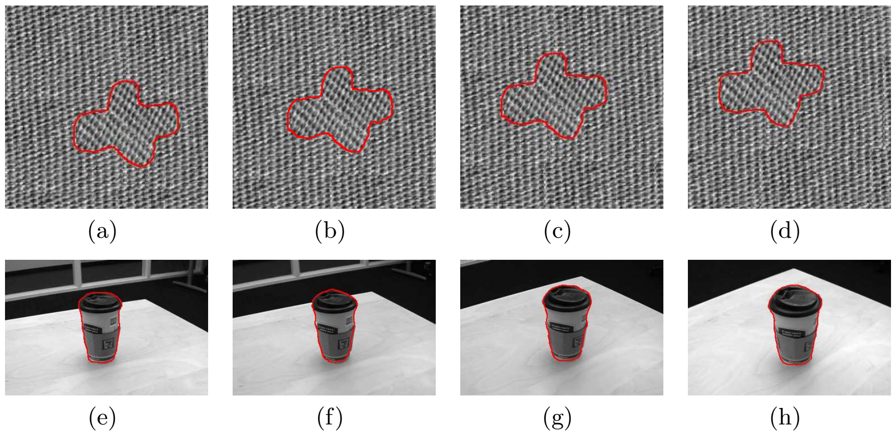


Fig. 7. Segmenting synthetic and real data. (a)(e) Start curves. (b)(c)(f)(g) Intermediate results. (d)(h) End curves.

9. Anandan, P.: A computational framework and an algorithm for the measurement of visual motion. Journal on Computer Vision **2** (1989) 283–310
10. Clarenz, U., Droske, M., Rumpf, M.: Towards fast non–rigid registration. In: Inverse Problems, Image Analysis and Medical Imaging, AMS Special Session Interaction of Inverse Problems and Image Analysis. Volume 313., AMS (2002) 67–84
11. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
12. Foley, J.D., et al: Computer Graphics: Principles and Practice in C. 2 edn. Addison Wesley (1992)
13. Nguyen, H.: Gpu gems 3. Addison-Wesley Professional (2007)

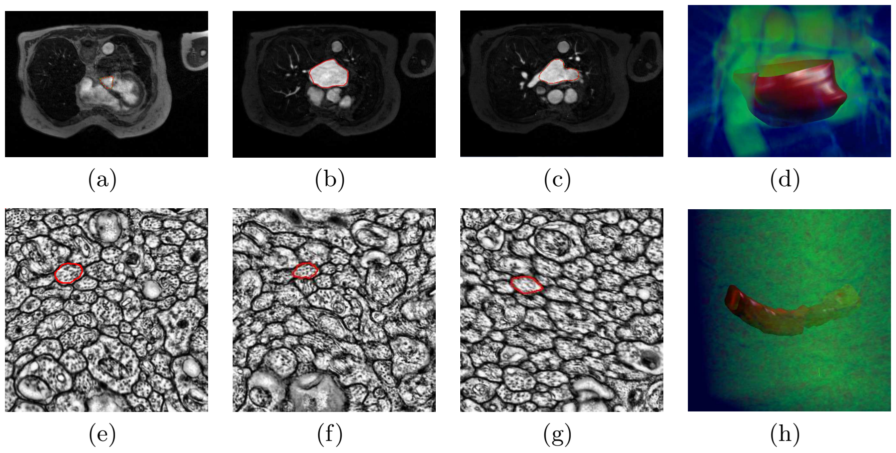


Fig. 8. Segmenting real medical data. (a)(e) Start curves. (b)(c) Intermediate results. (c)(g) End curves. (d)(h) Volume rendering of the Markov surfaces.