Published in final edited form as:

Inf Process Med Imaging. 2011 ; 22: 208–220.

Efficient Algorithms for Segmenting Globally Optimal and Smooth Multi-Surfaces*

Lei Xu¹, Branislav Stojkovic¹, Yongding Zhu¹, Qi Song², Xiaodong Wu², Milan Sonka², and Jinhui Xu¹

Lei Xu: lxu@buffalo.edu; Branislav Stojkovic: bs65@buffalo.edu; Yongding Zhu: yzhu3@buffalo.edu; Qi Song: qi-song@uiowa.edu; Xiaodong Wu: xiaodong-wu@uiowa.edu; Milan Sonka: milan-sonka@uiowa.edu; Jinhui Xu: jinhui@buffalo.edu

¹Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA

²Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242, USA

Abstract

Despite extensive studies in the past, the problem of segmenting globally optimal single and multiple surfaces in 3D volumetric images remains challenging in medical imaging. The problem becomes even harder in highly noisy and edge-weak images. In this paper we present a novel and highly efficient graph-theoretical iterative method with bi-criteria of global optimality and smoothness for both single and multiple surfaces. Our approach is based on a volumetric graph representation of the 3D image that incorporates curvature information. To evaluate the convergence and performance of our method, we test it on a set of 14 3D OCT images. Our experiments suggest that the proposed method yields optimal (or almost optimal) solutions in 3 to 5 iterations. To the best of our knowledge, this is the first algorithm that utilizes curvature in objective function to ensure the smoothness of the generated surfaces while striving for achieving global optimality. Comparing to the best existing approaches, our method has a much improved running time, yields almost the same global optimality but with much better smoothness, which makes it especially suitable for segmenting highly noisy images.

1 Introduction

In this paper, we consider the problem of segmenting single and multiple surfaces in 3D volumetric images. Efficient detection of globally optimal surface in volumetric images is one of the most challenging problems in imaging processing. The problem arises not only in the segmentation problem of biomedical images (e.g., CT, MRI, Ultrasound, Microscopy, Optical Coherence Tomography (OCT)), but also in many other fundamental optimization problems such as surface reconstruction [11], data mining [6], ore mining [5], metric labeling [3], and radiation treatment planning [3]. Many known segmentation methods used today are energy-based approaches, which can be classified into two categories, variational segmentation and combinatorial segmentation. Variational methods include snake [9], region competition, geodesic active contours [4], active appearance models, and other methods based on level-sets [13]. In most cases, variational segmentation uses variational optimization techniques that find only a local minima of the corresponding energy function.

*The research of the first three and the last authors was supported in part by NSF through a CAREER Award CCF-0546509 and a grant IIS-0713489. The research of the other three authors was supported in part by the NSF grants CCF-0844765 and the NIH grants K25 CA123112 and R01 EB004640.

Some versions of active contours compute a globally optimal solution, but only in 2-D. The combinatorial segmentation methods are mainly based on graph algorithms, such as minimum spanning tree [7], shortest paths and their 3D extensions [1], and graph cuts [2]. Each of these approaches has respective strengths and weakness, depending on the optimality of the cost function. In recent years, Wu et al. proposed a new detection method for optimal surfaces [11, 14]. Their optimality is controlled by a cost function using only the voxel weights and some geometric constraints on the connectivity of the surface. Their approach can compute a globally optimal surface in polynomial time. However, since their cost function only considers voxel weights, their surface often contains spikes and jaggedness in noisy regions.

To provide better solution, we study in this paper a new variant of the segmentation problem, called *Optimal Smooth Surfaces (OSS)*. In the OSS problem, the objective function is a linear combination of voxel weights and their smoothness, measured by the curvature of each voxel on the segmented surfaces. Thus our problem aims to simultaneously achieve global optimality of the surfaces and local smoothness. To ensure the local smoothness, OSS uses (mean) curvature as a penalty to avoid spikes (as spikes often have very large curvature). The additional requirement on the smoothness dramatically increases the hardness of the problem. There are evidences suggesting that the OSS problem is NP-hard. To our best knowledge, there is no efficient volumetric image segmentation algorithm which guarantees both global optimality and smoothness.

For the OSS problem, we present a novel graph-theoretical approach. Our approach first uses shortest paths (or edge-disjoint shortest paths) to compute a set of local segmentations, and then stitches them to form a global surface (or surfaces). The obtained surfaces are then iteratively improved through local computation, with each iteration strictly reducing the objective value. We tested our approach using a set of OCT images (e.g., 14 data sets). Our experiments show that within a small constant (3-5) number of iterations, our approach converges rather quickly to optimal or almost optimal. Compared to existing best approaches ([15]) using OCT images, our approach avoids to perform the rather costly minimum cut computation, and hence significantly improves the running time. Our segmentation has a total weight which is almost equal to that of the weight-only approach, but with much improved smoothness. In all our experiments, our approach almost eliminates all spikes, seemingly suggests that our approach could handle highly noisy images.

Since all computations in our approach are based on local information. Our approach is particularly suitable for parallel implementation on GPUs, which could potentially allow us to segment 3 or 4D volumetric images in real time.

2 Problem Description

In general, the multi-surfaces segmentation problem can be viewed as an optimization problem for finding a set of surfaces with the minimum cost-such cost being edge and/or region based-so that the obtained surface set is feasible. In this section, we define what is meant by a feasible surface set and the cost of a set of surfaces. We first briefly review what is curvature and mean curvature of a surface.

2.1 Curvature and Mean Curvature

Curvature captures the intrinsic geometric properties of surfaces and has been widely used for measuring smoothness [10, 16]. For 3D surfaces, there are two types of commonly used curvatures, mean curvature and Gaussian curvature. Mean curvature is defined as the mean of two 2D principal curvatures measured from two orthogonal directions which have the maximum and minimum bending rates. The Gaussian curvature is defined as the product of

the two principal curvatures. In the OSS problem, we use the mean curvature for two main reasons. (i) Gaussian curvature is not indicative as a shape descriptor in cases where the minimal curvature is zero (e.g., cylinder). (ii) The mean curvature is relatively easier to approximate by two orthogonal directions. Note that when the surface is unknown in advance, it is highly challenging to compute the exact curvature. Thus most of existing techniques compute only approximation curvature [8]. In our proposed approaches, we use the curvatures along the x and y directions to approximate the mean curvature, which gives us the freedom to de-couple the curvature computation in the two directions. We call the curvatures in the two direction as x-curvature and y-curvature respectively.

Each x and y curvature is the curvature of a 2D curve. For a 2D curve given explicitly as a

function of $y = f(x)$, its curvature is $\kappa = \frac{|y''|}{(1+y'^2)^{3/2}}$. For a more general curve with

parametrization $s(t) = (x(t), y(t))$, its curvature at $s(t)$ is $\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{3/2}}$.

2.2 Single Surface Detection

The single surface detection (OSS-1) problem can be defined as follows: Given a 3D image I (see Fig. 1) of size $n_x \times n_y \times n_z$ and with each voxel $I(x, y, z)$ associated with a non-negative weight $w(x, y, z)$ representing the inverse of the probability that $I(x, y, z)$ is a boundary voxel, find a terrain surface S (i.e., each column of I intersects S at one voxel) with the minimum cost, where the cost of S is defined as

$c(S) = \sum_{I(x,y,z) \in S} \alpha w(x, y, z) + (1 - \alpha) |\kappa(x, y, z)|$, where $\kappa(x, y, z)$ is the mean curvature of S at $I(x, y, z)$, $\alpha \in [0, 1]$ is a weighting constant, and the voxel weight is computed from some standard edge-detection technique. Computing the weight of the OCT data sets examined in this paper was described in [15] and [12]. An optimal surface is the one with the minimum total sum of voxel costs among all feasible surfaces that can be defined in the 3D image. The feasibility of a surface is constrained by the smoothness parameter $\kappa_{\max}$ which is the maximum curvature allowed of the surface, such that for each voxel $I(x, y, z)$ on the surface,

$|\kappa(x, y, z)| \leq \kappa_{\max}$. In the OSS problem, $\kappa(x, y, z) = 0$ since we choose $\kappa = \frac{|y''|}{(1+y'^2)^{3/2}}$ to compute it.

2.3 Multiple Surface Detection

The multiple surface detection (OSS-M) problem is an extension of OSS-1 problem discussed in the above section. In simultaneously detecting $k(k \geq 2)$ distinct but interrelated surfaces, the optimality is not only determined by the inherent costs and smoothness properties of the individual surfaces, but also confined by their interrelations. Thus additional constraints are added to model the desired relations between the surfaces. Let each surface of interest S_i denoted by a function $f_i(x, y)$ mapping (x, y) pairs to z-values. For each pair of surfaces S_i and S_j , a constraint will be added to require that $|f_i(x, y) - f_j(x, y)| > \delta$ for all (x, y) , where δ is surface interaction parameter. While in general a pair of surfaces may be allowed to cross, having crossing surfaces does not make sense for the 3D images in many practical problems where the surface are expected not to intersect or overlap, and thus we will also assume $\delta > 0$. Then a set of surfaces are considered feasible if each individual surface in the set satisfies the given smoothness constraints for the surface and each pair of surfaces does not cross. The multiple surface detection problem is to find a set of feasible surfaces with minimum cost.

3 Algorithms

To solve the OSS-1 and OSS-M problem, we propose a graph-theoretical based iterative approach. In this approach, our main idea is to reduce the problem to a sequence of shortest path problem in 2D spaces. To illustrate the idea more clearly, we present the algorithm for the OSS-1 and OSS-M problem respectively in the following sections.

3.1 Algorithm for Single Surface Detection

Instead of solving the problem in one shot, our strategy to detect the surface with global optimum (including smoothness) in the OSS-1 problem is iterative local improvement. More specifically, the algorithm for solving the OSS-1 problem contains two main steps. First, a feasible surface S with less cost is constructed as the initial solution. Second, the surface S is iteratively improved to global optimum by local adjustment. We call the first step as *initialization* and second step as *stitching*.

Initialization of OSS-1—In the initialization step, our goal is to construct a feasible surface S with less cost. To achieve this, our approach uses the following two observations: (a) The curvature can be computed independently in the x and y directions (e.g., the x-curvature of $I(x, y, z)$ depends only on voxels in the neighboring columns $Col(x-1, y)$, $Col(x, y)$ and $Col(x+1, y)$); (b) The curvature in one direction can be computed by using only a constant number (e.g., 3) of neighboring voxels. With these observations, we show (1) how to find $S(1, :, :)$ in the first x-slice $I(1, :, :)$ and (2) how to expand it to the rest x-slices.

In (1), $S(1, :, :)$ is actually the curve which is the intersection between the first x-slice $I(1, :, :)$ and S . To find this curve, our idea is to construct a weighted directed graph $G_1 = (V_1, E_1)$ of $I(1, :, :)$ and reduce it to a shortest path problem in G_1 which could be solved optimally. G_1 is constructed according to $I(1, :, :)$ as follows. Every vertex $V(u, c, b) \in V_1$ is a 3-tuple of voxels $u = I(x_u, y_u, z_u)$, $c = I(x_c, y_c, z_c)$ and $b = I(x_b, y_b, z_b)$ in $I(1, :, :)$, where the three voxels satisfy the following three conditions,

$$\begin{cases} x_u = x_c = x_b = 1, & (c1) \\ y_u = y_c + 1 \text{ and } y_b = y_c - 1, & (c2) \\ \kappa[V(u, c, b)] \leq \kappa_{\max} & (c3) \end{cases}$$

where (c1) is obvious, (c2) means that the three voxels should be from three consecutive neighboring columns $Col(x, y-1)$, $Col(x, y)$ and $Col(x, y+1)$ respectively, and (c3) is for guaranteeing the smoothness. More precisely, let $\kappa[V(u, c, b)]$ be the curvature of voxel c when voxel u and voxel b as its y-direction neighbors. We have

$$\kappa[V(u, c, b)] = \frac{1}{2}(\kappa_x[V(u, c, b)] + \kappa_y[V(u, c, b)]) \quad (1)$$

where $\kappa_x[V(u, c, b)]$ and $\kappa_y[V(u, c, b)]$ denote the x-direction and y-direction curvature of voxel c respectively. Since we do not consider x-direction curvature in this case,

$\kappa[V(u, c, b)] = \frac{1}{2}\kappa_y[V(u, c, b)]$. The cost assigned to $V(u, c, b)$ is

$$\text{cost}[V(u, c, b)] = \alpha w(x_c, y_c, z_c) + (1 - \alpha)\kappa[V(u, c, b)] \quad (2)$$

An edge is added for a pair of vertices $V(u, c, b)$ and $V(u', c', b')$ if $c = u'$ and $b = c'$ which indicate that the pair of vertices share two common voxels. After adding two dummy vertices in G_1 (one as source and the other as sink), the problem in the first x-slice $I(1, :, :)$

can be reduced to a shortest path problem in G_1 . Since G_1 is a DAG (directed acyclic graph), it could be solved optimally by topological sort algorithm in linear time $O(n_y n_z \kappa_{\max}^3)$. The cost of the resulting path includes the weight and the y-curvature of each voxel on the path. Thus $S(1, :, :)$ is formed by the voxels appeared on the path.

In (2), we generalize the above idea and expand $S(1, :, :)$ on the rest x-slice one by one to form a feasible surface S . There are two differences which could be illustrated by example of generating $S(i, :, :)$ from partial surface $S(1 : i-1, :, :)$. First, in addition to the above three conditions (c1), (c2) and (c3), every vertex $V(u, c, b) \in V_i$ in $G_i = (V_i, E_i)$ has to satisfy one more condition (c4) which restricts the y-coordinate y_c of voxel c by

$$\kappa[I(i-1, y_c, z_l)] \leq \kappa_{\max}, \text{ (c4)}$$

where $I(i-1, y_c, z_l)$ is the voxel of S on column $Col(i-1, y_c)$ and $\kappa[I(i-1, y_c, z_l)]$ is the curvature of that voxel. By adding (c4), the smoothness property of voxel $I(i-1, y_c, z_l)$ is preserved because its three neighboring voxels are fixed and the smoothness is not violated before we compute $S(i, :, :)$. The second difference is on the cost assigned to the vertex $V(u, c, b)$. More specifically, the equation (2) is replaced by the following one,

$$\text{cost}[V(u, c, b)] = \alpha(2w(x_c, y_c, z_c) + w(i-1, y_c, z_l)) + (1-\alpha)(\kappa[V(u, c, b)] + \Delta\kappa[I(i-1, y_c, z_l)]) \quad (3)$$

where $w(i-1, y_c, z_l)$ is the cost of voxel $I(i-1, y_c, z_l)$ on surface S and $\Delta\kappa[I(i-1, y_c, z_l)]$ is the change of curvature of voxel $I(i-1, y_c, z_l)$ since it does not have voxel c as its x-direction neighbor previously. In addition to the weight of voxel $I(u, c, b)$ itself, we need to include the weight adjustment for $I(i-1, y_c, z_l)$ caused by voxel $I(u, c, b)$ and thus the coefficient 2 before $w(x_c, y_c, z_c)$ follows. The new cost formula (3) reflects the overall change of cost for chosen voxel c on frame $S(i, :, :)$. The size of G_i is reduced from $O(n_y n_z \kappa_{\max}^3)$ to $O(n_y \kappa_{\max}^4)$. Finally, $S(i, :, :)$ is formed by the voxels appeared on the shortest path of G_i .

Stitching of OSS-1—To form an optimal surface from S constructed in the above section, our idea is to iteratively improve S by local adjustment called stitching. In this section, we first interpret what is stitching and then show how to iteratively apply it to obtain an optimal surface.

Firstly, it is easy to see that before S achieves optimum, there must exist an x-frame $S(i, :, :)$ on which the voxels are not entirely located at optimal positions. Otherwise S must be optimal. Thus in order to improve S , one way is to adjust $S(i, :, :)$ locally when we fix the rest of S other than $S(i, :, :)$. To illustrate the idea of stitching more clearly, let $S(1 : i-1, :, :)$ and $S(i+1 : n_x, :, :)$ be two partial surfaces. To stitch $S(1 : i-1, :, :)$ and $S(i+1 : n_x, :, :)$ in slice $I(i, :, :)$, our main idea is to fix the selected voxels in slices $I(i-1, :, :)$ and $I(i+1, :, :)$, and convert the stitching problem into a 2D shortest path problem in graph $G_s(i)$ which is similar to the graph G_i discussed in the above section. However, there are two differences from the initialization step. Firstly, in addition to the above four conditions (c1)-(c4), every vertex $V(u, c, b) \in V_s(i)$ in $G_s(i) = (V_s(i), E_s(i))$ has to satisfy one more condition (c5) which restricts the y-coordinate y_c of voxel c to preserve the smoothness property of S . More precisely,

$$\kappa[I(i+1, y_c, z_r)] \leq \kappa_{\max}, \text{ (c5)}$$

where $I(i+1, y_c, z_r)$ is the voxel of S on column $Col(i+1, y_c)$ and $\kappa[I(i+1, y_c, z_r)]$ is the curvature of that voxel. Secondly, different from the initialization step, the cost function in this case includes the weight and the mean curvature of voxels in slice $I(i, :, :)$, as well as the adjustment of the x-curvatures for those selected voxels in slices $I(i-1, :, :)$ and $I(i+1, :, :)$. We replace equation (3) by

$$\text{cost}[V(u, c, b)] = \alpha \text{cost}_w[V(u, c, b)] + (1 - \alpha) \text{cost}_k[V(u, c, b)], \quad (4)$$

where

$$\text{cost}_w[V(u, c, b)] = w(x_c, y_c, z_c) + w(i-1, y_c, z_l) + w(i+1, y_c, z_r) + 2(w(x_c, y_c, z_c) - w(x'_c, y'_c, z'_c)) \quad (5)$$

and

$$\text{cost}_k[V(u, c, b)] = \kappa[V(u, c, b)] + \Delta\kappa[I(i-1, y_c, z_l)] + \Delta\kappa[I(i+1, y_c, z_r)], \quad (6)$$

and $w(x'_c, y'_c, z'_c)$ is the weight cost of voxel on column $Col(I, y_c)$ of unadjusted $S(i, :, :)$. We set coefficient 2 before $(w(x_c, y_c, z_c) - w(x'_c, y'_c, z'_c))$ in equation (5) to reflect the weight adjustment for both selected voxels in slices $I(i-1, :, :)$ and $I(i+1, :, :)$. In equation (6), we include the curvature of voxels in slice $I(i, :, :)$ as well as adjustment of the x-curvatures for both selected voxels in slices $I(i-1, :, :)$ and $I(i+1, :, :)$. Thus an locally improved $S(i, :, :)$ can be obtained by shortest path algorithm on graph $G_s(i)$ within $O(n_y \kappa_{\max}^3)$ time.

Since stitching on one x-frame could locally improve S , our strategy is repeatedly apply the stitching algorithm on the obtained surface with each time shifting the to-be-stitched slices by one slice. For instance, if the current iteration of stitching is on slices $I(1, :, :), I(3, :, :), \dots, I(2i-1, :, :), \dots$ (see Fig. 2 (a)-(d)), then in the next iteration, the stitching is on slices $I(2, :, :), I(4, :, :), \dots, I(2i, :, :), \dots$. It can be shown (through mathematical proof and experiments) that the cost of the surface will monotonically decrease in each iteration and an optimal solution can be finally obtained. In all our experiments with the OCT images, our approach converges to the optimal (or almost optimal) within a small constant (3-5) number of iterations.

3.2 Algorithm for Multiple Surface Detection

To solve the OSS-M problem, we extend the algorithm of OSS-1 problem discussed in the above section. The algorithm for the OSS-M problem also contains two main steps, initialization and stitching. Thus we only focus on the extension of algorithm to avoid repetition.

Initialization of OSS-M—In the initialization step, our goal is to construct a set of feasible surfaces S_1 to S_k with less cost, where k is the number of surfaces in the 3D image. To achieve this, we show (1) how to find $S_1(1, :, :)$ to $S_k(1, :, :)$ in the first x-slice $I(1, :, :)$ and (2) how to expand $S_1(1, :, :)$ to $S_k(1, :, :)$ to the rest x-slices. In (1), we first construct G'_1 similar with G_1 in the above section. In order to find $S_1(1, :, :)$ to $S_k(1, :, :)$, we need to compute k shortest edge-disjoint paths. It could result in a wrong solution if we try to solve the problem by executing k times the following two steps: compute a shortest path and remove its arcs from the graph G'_1 . To see this, let us consider the example in Fig. 3a. In this example, we intend to determine two edge-disjoint shortest paths from 1 to 4. The resulting network is the one shown in Fig. 3b, where the two shortest path from 1 to 4 is $\{1, (1, 2), 2, (2, 4), 4\}$ and $\{1, (1, 3), 3, (3, 4), 4\}$. Note that the shortest path in Fig. 3a $\{1, (1, 2), 2, (2, 3), 3, (3, 4), 4\}$ is not one of the two shortest disjoint paths. To overcome this difficulty, one way is using the Suurballe's algorithm. In this algorithm, the k shortest arc-disjoint paths

problem is considered to be equivalent to a minimum cost flow problem in which a feasible flow of value k is sought from an initial vertex to a terminal vertex such that the total cost is minimized. The flow in each arc can not exceed the unity. Obviously, if such a flow does not exist, the same can be concluded about the k disjoint paths. In order to obtain the k paths, we have to construct a new graph G''_1 , whose set of arcs is the set of arcs for which the flow is one. To determine the shortest disjoint path, the second disjoint shortest, the third disjoint shortest, and so up to the k -th shortest disjoint path, we have to execute k times the following two steps (1) Compute a shortest path in the graph G''_1 and (2) Remove its arcs. However, it is difficult to directly apply this algorithm to our problem because (1) the surface interaction parameter δ is not easily encoded as a constraint and (2) the running time of minimum cost flow algorithm is $O(E \log(E + V \log V))$. Since our goal is to find a set of feasible solutions, our strategy is to compute the shortest path one by one, and remove δ vertices above and below for each vertex on the shortest path. To avoid the effect caused by the deletion of vertices in the previous step of computing shortest path, we relax $\kappa_{\max}$ when we compute the second to k shortest paths. Finally, we sort the voxels on each column by their z -coordinates. If any voxel of $S_1(1, :, :)$ to $S_k(1, :, :)$ violates the smoothness requirement, we replace it by its neighbor within the same column and thus make the result feasible. In such a way, since k is a small constant number (3 or 4), the running time is still linear to the size of graph G'_1 . Thus in the next round, we expand $S_1(1, :, :)$ to $S_k(1, :, :)$ on x -frame $I(2, :, :)$ one by one. By repeating the same procedure $(n_x - 1)$ times, we expand $S_1(1, :, :)$ to $S_k(1, :, :)$ to a set of feasible surfaces.

Stitching of OSS-M—To obtain a set of surfaces with global optimum, we follow the same idea of algorithm for OSS-1 problem. For instance, if the current iteration of stitching is on slices $I(1, :, :), I(3, :, :), \dots, I(2i-1, :, :), \dots$, then in the next iteration, the stitching is on slices $I(2, :, :), I(4, :, :), \dots, I(2i, :, :), \dots$. For one iteration, the set of k surfaces is stitched one by one similar to OSS-1's. However, to avoid intersection of surfaces, we add one more restriction (c6) of the y -coordinate y_c of voxel c on the to-be-stitched x -frame of surface S ,

$$y_l < y_c < y_u, (c6)$$

where $y_l(y_u)$ is the y -coordinate of the voxels on the same column of y_c and belonged to the surface below (above) to S .

4 Experiments

The proposed algorithms were examined on 14 datasets of 3D OCT images ($200 \times 200 \times 256$). To prove the effectiveness of our algorithms, we mainly focused on segmenting surface 1, 6 and 7 since they were computed by the same weight function [12]. The average of the two tracings from two human experts were used as the reference standard. The unsigned surface positioning errors were computed for each surface as a distance between the computed results and the reference standard. In our experiments, we apply our algorithm to the down-sampled 3D images ($100 \times 100 \times 128$). The results were up-sampled to the size of original 3D images before we compare them to the reference standard. Our algorithm is implemented by C++ and LEDA (Library of EFFicient Data types and Algorithms)-5.2. The experiments were conducted on a Linux workstation (2.4 GHz, 4GB memory).

5 Results

To test the performance of our algorithms, we mainly focus on the results of OSS-M for the case of $k = 3$ (i.e., detecting surfaces 1, 6, and 7), since OSS-1 very well detects surface 6.

The computed unsigned position errors are summarized in the table to the right where $\alpha = 0.75$ and $\kappa_{\max} = 5$. Compared to previous result reported in the literature [15] in which all 7 surfaces were detected simultaneously, our segmentation results are more accurate for surface 1 and 6 (i.e., 4.0 ± 1.2 and 3.5 ± 2.0 in [15]), and comparable for surface 7 (i.e., 7.8 ± 2.5 in [15]). An example result with $\alpha = 0.75$ and $\kappa_{\max} = 5$ shown in Fig. 4(a) is compared to the reference standard in Fig. 4(b). Our result (shown in green line) is consistent with the reference standard (shown in red line).

Table 1

Mean $\pm$ SD in μm		
Surface 1	Surface 6	Surface 7
3.65 ± 2.70	2.35 ± 1.56	7.9 ± 4.89

To analyze the convergence of OSS-M, we measure the cost deduction in each iteration. Fig. 5(a) shows that for each iteration, the deduction of the surfaces' cost is positive before it goes down to 0. This means that (1) the cost of all surfaces is monotonically decreased, (2) the proposed method yields optimal solutions in at most 5 iterations, and (3) the largest cost deduction occurs in the first two iterations.

To evaluate the running time of our algorithm, we measure the average running time (in seconds) t_i for a stitching iteration (obtained from dividing the total time by the number of iterations; see Fig. 5 (b)) and the total running time T (see Fig. 5 (c)) for different $\kappa_{\max}$ values. It is interesting to see that even though the searching space increases super linearly with respect to $\kappa_{\max}$, the running time is roughly a linear function of $\kappa_{\max}$.

To examine the influence of $\kappa_{\max}$ on the segmentations, we intentionally choose some small $\kappa_{\max}$ (e.g., $\kappa_{\max} = 2$). Fig. 6 shows that for too small $\kappa_{\max}$ value, the algorithm may generate wrong result and could easily correct it when increasing $\kappa_{\max}$. Our experiments show that $\kappa_{\max} = 5$ is sufficient for all tested images. This also means that the average running time of our algorithm is around 4 minutes.

To visualize the improved smoothness by our algorithm, we measure the total curvature and total weight of a particular surface for different α values. Fig. 7(a) shows the surface for $\alpha = 1$, meaning that the cost function uses only weight. Fig. 7(b) is the same surface with $\alpha = 0.75$. From Fig. 7, we can see that our algorithm could improve the smoothness of the surface and remove all spikes by incorporating curvature into the cost function. The total curvature improves about 5% while the total weight increases only 0.2%.

6 Discussion and Conclusion

We have presented efficient algorithms for segmenting globally optimal and smooth multi-surfaces. These are the first algorithms that utilize curvature in the objective function to ensure smoothness. Comparing to the best existing approaches using graph cut algorithms [15], our method can detect more smooth surfaces with much improved running time. The proposed method is also suitable for parallel implementation on GPUs, which could potentially allow us to segment highly noisy volumetric images in real time.

References

1. Ardon, Roberto; Cohen, Laurent D.; Yezzi, Anthony J. A new implicit method for surface segmentation by minimal paths: Applications in 3d medical images. *Energy Minimization Methods in Computer Vision and Pattern Recognition*. 2005:520–535.
2. Boykov, Yuri; Kolmogorov, Vladimir. Computing geodesics and minimal surfaces via graph cuts. *International Conference on Computer Vision*. 2003:26–33.
3. Chekuri, Chandra; Khanna, Sanjeev; Naor, Joseph; Zosin, Leonid. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM J. Discret. Math.* 2005; 18(3):608–625.
4. Caselles, Vicent; Kimmel, Ron; Sapiro, Guillermo. Geodesic active contours. *International Journal of Computer Vision*. 1995; 22:61–79.
5. Schofield, Denby. Genetic algorithms: A new approach to pit optimization. *Proc. 24th Int.Symp. on the Application of Computers and Operations Research in the Mineral Industry*. 1993:126–133.
6. Fukuda, Takeshi; Morimoto, Yasuhiko; Morishita, Shimichi; Tokuyama, Takeshi. Data mining with optimized two-dimensional association rules. *ACM Trans. Database Syst.* 2001; 26(2):179–213.
7. Felzenszwalb, Pedro F.; Huttenlocher, Daniel P. Efficient graph-based image segmentation. *Int. J. Comput. Vision*. 2004; 59(2):167–181.
8. Flynn, Patrick J.; Jain, Anil K. On reliable curvature estimation. *CVPR '89: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1989:110–116.
9. Kass, Michael; Witkin, Andrew P.; Terzopoulos, Demetri. Snakes: Active contour models. *International Journal of Computer Vision*. 1988; 1(4):321–331.
10. Leventon, Michael E.; Grimson, W Eric L.; Faugeras, Olivier; Wells, William M, III. Level Set Based Segmentation with Intensity and Curvature Priors. *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*. 2000
11. Li, Kang; Wu, Xiaodong; Chen, Danny Z.; Sonka, Milan. Optimal surface segmentation in volumetric images-a graph-theoretic approach. *IEEE Trans. Pattern Anal. Mach. Intell.* 2006; 28(1):119–134. [PubMed: 16402624]
12. Garvin, Mona K.; Abràmoff, Michael D.; kardon, Randy; Russell, Stephen R.; Wu, Xiaodong; Sonka, Milan. Intraretinal Layer Segmentation of Macular Optical Coherence Tomography Images Using Optimal 3-D Graph Search In em *IEEE Trans Med Imaging*. 2008; vol. 27(10):1495–1505.
13. Osher, Stanley; Paragios, Nikos. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Secaucus NJ, USA: Springer-Verlag New York, Inc; 2003.
14. Wu, Xiaodong; Chen, Danny Z. Optimal net surface problems with applications. *International Colloquium on Automata, Languages and Programming*. 2002:1029–1042.
15. Haeker, Mona; Wu, Xiaodong; Abràmoff, Michael D.; Kardon, Randy; Sonka, Milan. Incorporation of Regional Information in Optimal 3-D Graph Search with Application for Intraretinal Layer Segmentation of Optical Coherence Tomography Images. *Information Processing in Medical Imaging*. 2007:607–618. [PubMed: 17633733]
16. Wyatt, Christopher L.; Ge, Yaorong; Vining, David J. Segmentation in virtual colonoscopy using a geometric deformable model. *Computerized Medical Imaging and Graphics*. 2006; vol. 30(1):17–30. [PubMed: 16413746]

Fig. 1.
Orientation of surface

Fig. 2.
One iteration of stitching. (a) Before stitching; (b) Fixed the red voxels; (c) Stitching; (d) The new surface.

Fig. 3.
Counter-example of applying shortest path algorithms k times

(a)

(b)

Fig. 4.

(a) Example result shown on a 2-D scan from one of the 3D images; (b) Red: reference standard; Green: our result.

Fig. 5.
 (a) Cost deduction for all surfaces in each iteration; (b) Average running time for an iteration; (c) Total running time.

(a)

(b)

Fig. 6.
(a) $\kappa_{max} = 2$; (b) $\kappa_{max} = 5$

Fig. 7.

(a) $\alpha = 1$ (weight only); (b) $\alpha = 0.75$ ($0.75 \times \text{weight} + 0.25 \times \text{curvature}$).