Surface Extraction from Binary Volumes with Higher-Order Smoothness

Victor Lempitsky
University of Oxford*

Abstract

A number of 3D shape reconstruction algorithms, in particular 3D image segmentation methods, produce their results in the form of binary volumes, where a binary value indicates whether a voxel is associated with the interior or the exterior. For visualization purpose, it is often desirable to convert a binary volume into a surface representation. Straightforward extraction of the median isosurfaces for binary volumes using the marching cubes algorithm, however, produces jaggy, visually unrealistic meshes. Therefore, similarly to some previous works, we suggest to precede the isosurface extraction by replacing the original binary volume with a new continuous-valued embedding function, so that the zero-isosurface of the embedding function is smooth but at the same time consistent with the original binary volume.

In contrast to previous work, computing such an embedding function in our case permits imposing a higher-order smoothness on the embedding function and involves solving a convex optimization problem. We demonstrate that the resulting separating surfaces are smoother and of better visual quality than minimal area separating surfaces extracted by previous approaches to the problem. The code of the algorithm is publicly available.

1. Introduction

The problem of surface extraction from binary volumes arises in the post-processing step of several computer vision applications. In particular, binary image segmentation algorithms such as region growing [1], graph cuts [6], or a simple thresholding proceed by labeling image elements as belonging to either foreground or background. The output of such an algorithm is, therefore, a binary-valued segmentation mask. For two-dimensional problems, the segmentation mask can be visualized in several ways, e.g. by superimposing it onto the original image. Over the recent years, there is, however, an ever-increasing demand for 3D image segmentation, where a three-dimensional segmentation mask (a binary volume) needs to be visualized after the segmentation is performed.

In many cases, the user expects the 3D image segmentation result to be presented in the form of a *separating surface*, i.e. a surface that separates the background and the foreground segments of a binary volume. In many applications such as the biomedical imaging, this surface may correspond to the actual physical interface, e.g. the boundary of an organ. The problem of extracting a separating surface from a binary volume also arises within the post-processing step in several other applications such as stereo- or silhouette-based multiview reconstruction [15,16] or shape-from-range data [8], when the underlying algorithms work with voxel representations and make hard decisions about the voxel occupancy.

Given a binary volume, a separating surface can be extracted as an isosurface corresponding to the median value (e.g. the zero-isosurface is taken, if the background label is interpreted as -1 and the foreground label is interpreted as 1). These isosurfaces can be efficiently extracted in a form of triangular meshes using the marching cubes algorithm [18]. Such isosurfaces, however, exhibit distracting aliasing artifacts (Figure 1a, Figure 3a). These artifacts have a regular structure and therefore are perceived as "signal" rather than "noise", which leads to their amplification by the human visual system.

The aliasing problem is caused by the fact that a binary volume does not define the separating surface uniquely. In fact, depending on the interpretation, a binary volume is typically consistent with the entire family of separating surfaces. Thus, in this work we interpret a binary volume as a set of hard constraints imposed on the separating surface. Under these constraints, the separating surface must contain the centers of all foreground voxels inside while having the centers of all background voxels outside; whether or not this interpretation has a "physical" meaning depends on the particular algorithm used to compute the binary volume. It can be demonstrated, for example, that this interpretation has a sound geometric justification for the graph cut (or more pre-

*The author is supported by EU under ERC grant VisRec no. 228180 and Microsoft Research programs in Russia. The work was done while the author was with Microsoft Research Cambridge.

cisely *GeoCut*) framework [7], which became very popular for segmentation as well as other low-level vision tasks over the last years.

In the paper, we discuss a new method for the extraction of smooth separating surfaces based on the constrained convex optimization of the higher-order smoothness criterion. The results of the method for several binary volumes suggest that it yields higher-quality surfaces as compared to the previous approaches that are discussed in the following section.

2. Related Work

The problem of extracting a separating surface can be regarded as the problem of picking one out of an infinitely-large class of surfaces that meet the hard constraints imposed by the binary volume. The zero-isosurface is a choice that is suboptimal from the perceptual point of view, as it lacks smoothness. Some of the initial approaches [25] suggested to overcome the aliasing artifacts (jagginess and terracing) by a local Gaussian pre-filtering of the binary volume. The problem with that approach, however, is that the terracing/jagginess effects often require very large kernel and strong filtration to be diminished (let alone completely eliminated), whereas such a filtration smooths out the fine details and, in general, produces oversmoothed surfaces incompatible with the original binary volumes.

To overcome this essential non-locality of the aliasing effects, strategies based on the optimization of global objectives were suggested. Towards this end, [9] evolves the extracted surface to minimize its area, subject to the constraint that it has to remain compatible with the original binary volumes during the evolution. A body of work has considered similar constrained mesh evolution approaches either based on the constrained Laplacian smoothing [3, 4], which lead to similar surface area minimization effect, or based on the higher-order smoothness criteria [12, 19, 20], which has been demonstrated to improve the smoothness of the marching cube-based meshes considerably in the process of local non-convex optimization.

Following up on the area minimization approach, [26] introduced the surface extraction method, which solve essentially the same optimization problem (constrained minimal area separating surface) but in the implicit level-set framework [22]. The smoothness is thus introduced prior to the isosurface extraction by the modification of the underlying volume function. In this way, the function values are no longer restricted to be binary (1 or -1). Such an implicit smoothing strategy has several advantages over the explicit smoothing of the isosurface mesh including the ease of handling of topology changes during smoothing as well as the simplicity of imposing the hard constraints.

In this paper, we suggest a new simple criterion that similarly to [26] can be used to extract the separating surfaces from binary volumes within the implicit framework, yielding surfaces with higher-order smoothness rather than the minimal area property. Such criterion has a nice property of leading to convex optimization problems (as opposed to non-convex problems in previous frameworks). Importantly, we demonstrate that the higher-order smoothness imposed by the resulting algorithm allows to obtain separating surfaces with much fewer aliasing artifacts as compared to the area minimization method [26].

3. Surface Extraction with Higher-order Smoothness

Problem setting. Assume that a binary volume function $V : \mathcal{G} \rightarrow \{-1, +1\}$ is given, where $\mathcal{G}$ is the discrete grid domain $\mathcal{G} = \{1, 2, \ldots, L\} \times \{1, 2, \ldots, M\} \times \{1, 2, \ldots, N\}$. Then, denote with $v_{ijk} \in \{-1, +1\}$ the value of V on the respective node of the grid.

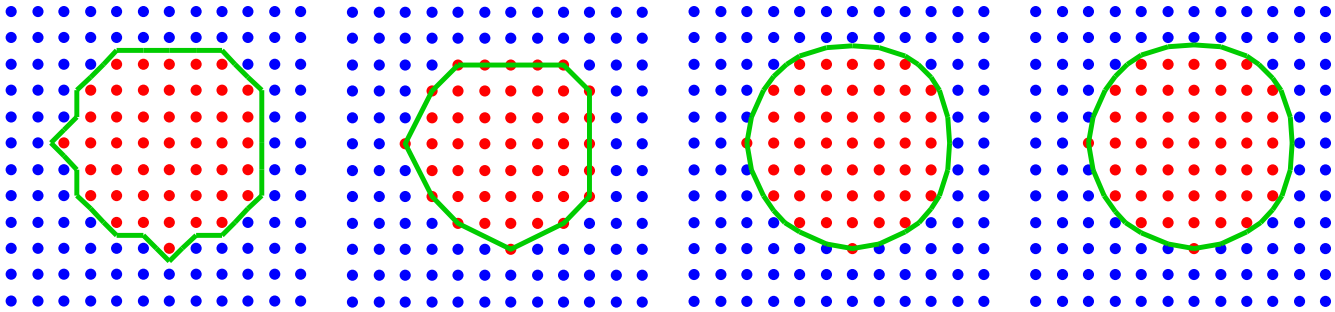
As a result of the smoothing, we are going to obtain a non-binary *embedding function* $F : \mathcal{G} \rightarrow \mathbf{R}$, where $f_{ijk} \in \mathbf{R}$ will again denote the value of F on the node of the grid. The obtained function F must be consistent with the binary volume V, so that the zero-isosurface extracted from (the continuous interpolation of) F contains all foreground nodes $\{i, j, k |\, v_{ijk} = +1\}$ inside (or on the boundary) and all background nodes $\{i, j, k |\, v_{ijk} = -1\}$ outside (or on the boundary). This requirement is equivalent to the following set of hard constraints imposed on F:

$$\forall i, j, k \qquad v_{ijk} \cdot f_{ijk} \geq 0\,. \tag{1}$$

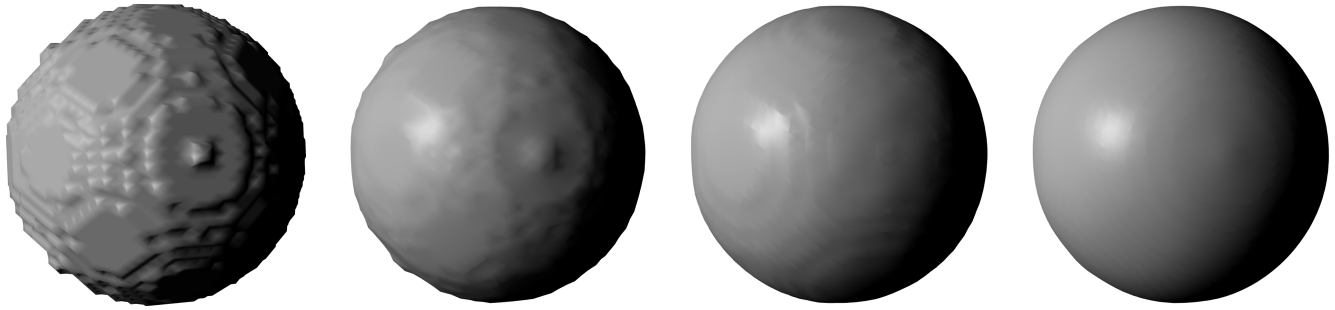
We seek to obtain F meeting constraints (1) so that its zero isosurface is smooth (in the sense that we discuss below). An idea that we use in our method is to impose smoothness on F directly, so that its isosurfaces also possess a certain degree of smoothness.

Imposing smoothness. The idea of imposing the smoothness directly on the embedding function is used extensively in image segmentation. Thus, the level set frameworks modify the embedding function locally, so that the area of the zero isosurface is also minimized. The TV-minimization framework [21] as well as the GeoCut framework [7] minimize the integral of the absolute value of variation of the embedding function. Again, although the smoothness is imposed on the embedding function rather than on its zero-isosurface[1], the area of the zero-isosurface is provably minimized in this case. These, however, lead to so-called "shrinking bias", that is bias towards smaller surfaces, which have smaller areas, and ultimately towards an empty surface, which is a unique global optimizer of such regularization.

[1]The term "0.5-isosurface" would be more consistent with the notation of papers concerned with these frameworks. We still use "zero-isosurface" here to be consistent with the rest of our paper.



(a) the binary volume and its zero-isoline | *(b)* minimal-length separating curve | *(c)* separating curve with regularization (2) | *(d)* separating curve with regularization (3)



(a) zero-isosurface | *(b)* minimal-area separating surface | *(c)* separating surface with regularization (2) | *(d)* separating surface with regularization (3)

Figure 1. (*better viewed in color*) **Top row:** a 2D example highlighting the performance of different approaches. The input binary volume is obtained by sampling a circle on a very coarse (12×12) grid, so that red nodes correspond to the interior (+1) and blue nodes correspond to the exterior (-1). Both the zero-isoline *(a)* and the minimal length separating curve *(b)* (as sought by [3, 9, 26]) suffer heavily from the aliasing artifacts, while our method computes smooth separating curves *(c,d)* with the shape close to being circular. **Bottom row:** analogous surfaces in the 3D case for a ball sampled on a discrete 3D grid. Same pattern is observed: zero-isosurface and minimal area separating surface suffer from jagginess, while higher order order regularization in our method leads to smoother surfaces. In 3D case, regularization (3) leads to smoother surface than regularization (2).

More recently, it has been argued (see e.g. [11]) that one can impose smoothness by minimizing the square (or other powers) of the variation of the embedding function. Unlike the case of the absolute value of the variation, minimizing such quantity for the embedding function does not translate into minimizing some clearly understood functional of the zero-isosurface (and the same holds for the regularization used in our approach). Yet, it has been demonstrated to achieve the desired effect of imposing smoothness on this isosurface [11].

The methods discussed above (except level-sets) minimize some function of the first-order variation $|\nabla F|$ or related quantities. As a result, these methods are biased towards the constant embedding functions, as this is the only class of the embedding functions, which are considered absolutely smooth under this definition of smoothness. To avoid this and achieve the higher-order smoothness, we suggest to regularize the higher-order derivatives of the embedding function. Thus, in the continuous limit, one may regularize the deviation of the laplacian of the embedding function from zero, by imposing the following regularization penalty:

$$\int (\Delta F)^2 \, dV \to \min, \tag{2}$$

where $\Delta F = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \frac{\partial^2 F}{\partial z^2}$ is a laplacian of F. The global minimizers of (2) are the embedding functions that are harmonic inside the bounding volume. As any plane may be expressed as a zero-isosurface of a linear (hence, harmonic) embedding function, planes (as well as other isosurfaces of harmonic functions) are not penalized by this regularization. This is in sharp contrast to the methods that use the first-order variation on the embedding function for the regularization, and thus penalize all surfaces except the empty ones.

Another very similar way to impose higher-order smoothness on the embedding function is to penalize the deviations of the second-order non-mixed derivatives from zero:

$$\int \left(\frac{\partial^2 F}{\partial x^2}\right)^2 + \left(\frac{\partial^2 F}{\partial y^2}\right)^2 + \left(\frac{\partial^2 F}{\partial z^2}\right)^2 dV \to \min. \tag{3}$$

This regularization essentially differs from (2) by the absense of cross-products between derivatives in the square

of the laplacian. Unlike (2), the functional (3) is not rotationally-invariant. The set of global minimizers of (3) is a subset of global minimizers of (2) that still includes all the linear functions (hence, planar surfaces are still not penalized).

Our framework can use any of the regulariztions (2) and (3), and both of them lead to the surfaces that are considerably smoother and less prone to jagginess artifacts than the minimal area isosurfaces (Figure 1). Between them, the functional (3) leads to smaller computational burden, faster convergence of our numeric optimization scheme, and often produce smoother and more visually pleasant results compared to (2), despite not being rotationally invariant. We, therefore, used the regularization (3) throughout the rest of the experiments.

In the discrete setting, the finite-difference approximation is used to express (3):

$$\sum_{ijk} \big[(f_{i+1\,jk} + f_{i-1\,jk} - 2f_{ijk})^2 + (f_{i\,j+1\,k} + f_{i\,j-1\,k} - 2f_{ijk})^2 + (f_{ij\,k+1} + f_{ij\,k-1} - 2f_{ijk})^2 \big] \rightarrow \min . \quad (4)$$

Similar finite-difference approximation can be used for the laplacian regularization (2).

Adding a margin. We now seek to obtain an embedding function that is smooth in the sense of (3) and meets the conditions (1). While the regularization (3) on its own does not bias the embedding function to be constant, one may notice however, that combining it with the hard constraints (1) would lead back to the unique and trivial optimal solution $F \equiv 0$. This can be avoided if the hard constraints (1) are made more stringent, ensuring some margin separating the resulting embedding function from the zero solution:

$$\forall i,j,k \qquad v_{ijk} \cdot f_{ijk} \geq m_{ijk} . \quad (5)$$

Here, m_{ijk} are non-negative values, which are strictly positive for some i,j,k, ensuring that the embeding function deviates from zero somewhere. There exist different reasonable choices of margin values that lead to perceptually plausible and similar separating surfaces. Thus, if we denote with B the set of boundary nodes in V, i.e. nodes adjacent (in 26-connectivity) to the nodes of the different values in V, the simple choice for m_{ijk} would be:

$$m_{ijk} = \begin{cases} 0, & \text{if } (i,j,k) \in B, \\ 1, & \text{otherwise.} \end{cases} \quad (6)$$

A marginally better results in our experiments were produced by the margin equal to the (unsigned) Euclidean distance to the set B:

$$m_{ijk} = dist\big((i,j,k), B\big) = \min_{(\alpha,\beta,\gamma)\in B} \sqrt{(i-\alpha)^2 + (j-\beta)^2 + (k-\gamma)^2} . \quad (7)$$

Algorithm 1 Surface Extraction

Require: Binary volume V
1: D = Signed Distance Function (V)
2: Compute narrow band NB from D
3: Compute margin (5) from D
4: F = Solve Quadratic Programming (8) in NB
5: S = Marching Cubes (F, 0-isosurface)
6: **return** mesh surface S

Figure 2. Our surface extraction algorithm.

Extracting the surface. To extract the separating surface from the binary volume, our method simply solves the following convex quadratic optimization problem:

$$SOLVE \quad (4) \qquad s.t. \quad (5). \quad (8)$$

The separating surface is then extracted as the zero-isosurface of the optimal embedding function using the marching cubes algorithm [18]. Alternatively, the recovered embedding function can be rendered directly using the volume rendering techniques such as [10].

4. Implementation details

Narrow band implementation. As we are interested in the values of F near its zero isosurface, we can restrict our computations to the nodes within the narrow band defined as $\{(i,j,k) |\ dist\big((i,j,k), B\big) < C\}$, where C is the constant defining the half-width of the band which can be set to a small value (e.g. 4) without visually affecting the resulting isosurface (as compared to the computations on the full grid). The overall pseudocode of our method is then given in Figure 2.

Solving the quadratic program. The convex quadratic program (8) can be solved using a large variety of the convex optimization algorithms [5], with the final result being invariant to the particular choice of the algorithm. For our experiments, we devised a simple special-purpose optimization scheme based on the projected Jacobi iterations, in the spirit of the scheme used in [14]. Experimentally, we have confirmed the convergence of the scheme to global optima.

To describe the scheme, we detail (8) as:

$$\mathbf{f}^T Q\, \mathbf{f} \rightarrow \min, \quad s.t.\ \mathbf{l} \leq \mathbf{f} \leq \mathbf{u}, \quad (9)$$

where $\mathbf{f}$ is a vector of the embedding function values in the voxels of the narrow band, $\mathbf{l}$ and $\mathbf{u}$ are the lower and upper bounds on the values of $\mathbf{f}$ derived from the margin inequality (5), and Q is a sparse, positive-semidefinite matrix of the quadratic form derived from the finite-difference approximation (4). Then, the kth iteration of the method starts with the current solution $\mathbf{x}^k$, computes its Jacobi update, makes

a step towards it, and then projects the resulting point onto the optimization domain by clamping the current solution to be between the lower and the upper bounds:

$$\mathbf{x}_{Jacobi}^{k+1} = -D^{-1}\, R\mathbf{x}^k \tag{10a}$$

$$\mathbf{x}_{step}^{k+1} = \omega \cdot \mathbf{x}_{Jacobi}^{k+1} + (1-\omega)\cdot \mathbf{x}^k \tag{10b}$$

$$\mathbf{x}^{k+1} = \min\left\{\mathbf{u}, \max\{\mathbf{l}, \mathbf{x}_{step}^{k+1}\}\right\} \tag{10c}$$

Here, $\mathbf{x}^{k+1}$ is the solution at the $k+1$ iteration of the algorithm, D and R are the diagonal and the off-diagonal parts of Q, the scalar ω is the step-length parameter, which in our experiments was set to 0.5, and the minima and the maxima are taken element-wise. The attractive properties of the update (10) are its small memory footprint as well as easiness and efficiency, in particular for GPU architectures.

Computing distance function. The distance function required to construct the band and to compute the margin values can be efficiently computed even for large volumes using the algorithm [23] among others.

5. Results

Qualitative assessment. The suggested algorithm has been evaluated on several synthetic and real binary volumes. In Figure 3a, we present the zero-isosurfaces of the input volumes, while the minimal area separating surfaces computed using the method [26] are shown in Figure 3b. Finally, the separating surfaces extracted with our method are presented in Figure 3c (the margin as defined by (7) was used). It can be observed that while the minimial area approach yields the surfaces that look much better than the original zero-isosurfaces, they still suffer from aliasing (terracing) artifacts. These artifacts are removed and smoother, more naturally-looking surfaces are obtained with our method.

The top row in Figure 3 correspond to a binary volume obtained by rasterizing a cube on low-resolution ($64 \times 64 \times 64$) grid (the cube axes were rotated relative to the grid axes). The second row corresponds to the binary segmentation result of an MRI hip joint dataset ($128 \times 128 \times 119$) obtained using the graph cut method [6]. The third row corresponds to the binary volume ($100 \times 100 \times 79$) obtained from the set of range scans using the method [17]. While for the illustration purposes the volumes in Figure 3 have low resolution, our method scales to much larger volumes due to its banded nature.

For example, Figure 4 shows the closeups of the surface extracted from the $256 \times 256 \times 288$ binary volume. It also demonstrates the main failure mode for our method, which are thin objects (protrusions). Although, our method (Figure 4b) still does marginally better than the minimal area surface approach (in Figure 4a), it can be seen that very thin objects cannot be processed with the method described above in a satisfactory way. The problem occurs because the functional (4) drives the value of the embedding function towards zero, while the margin value (7) does not prevent that, as it is equal to zero for the voxels belonging to thin parts.

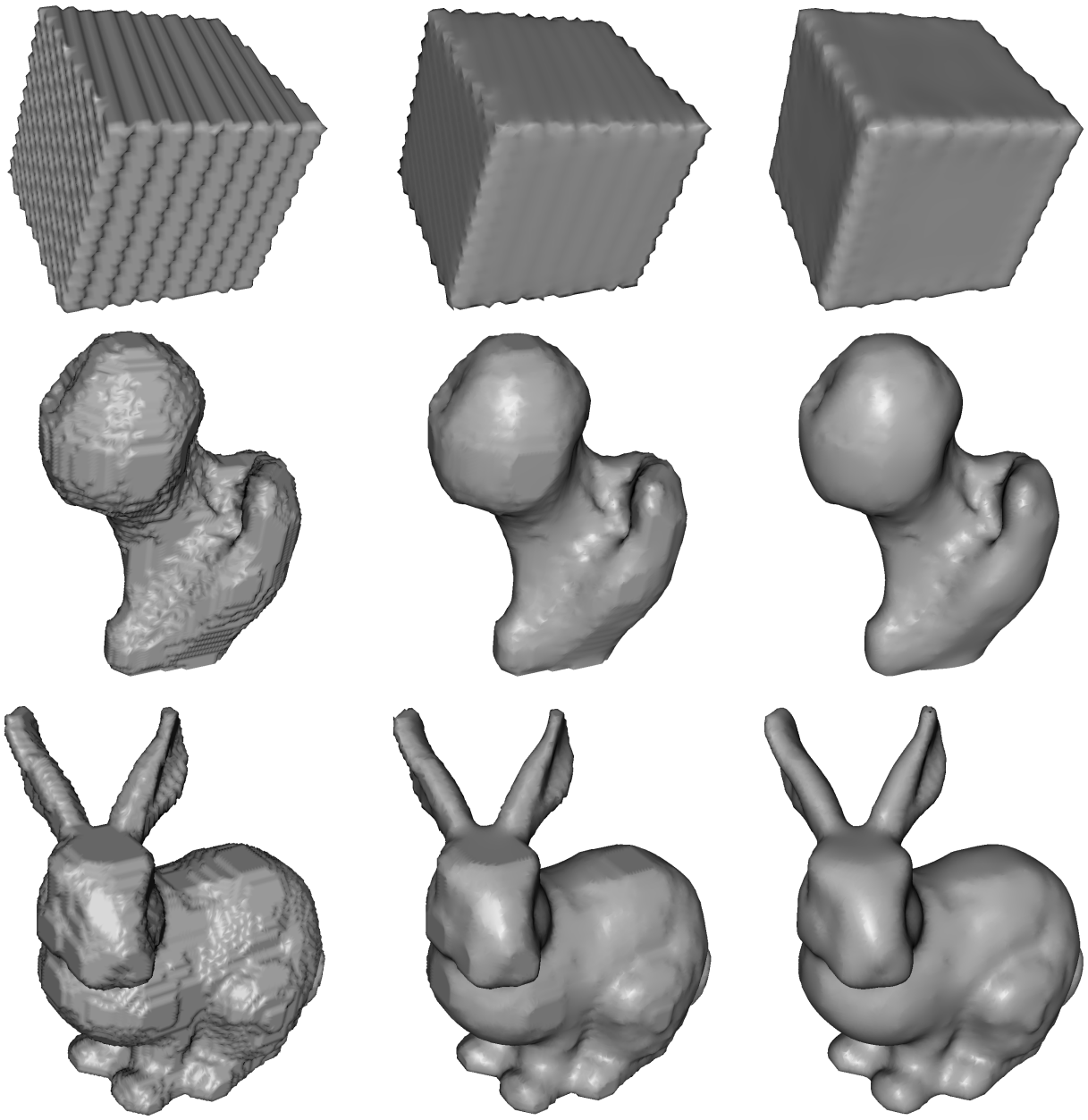
Handling thin parts. One easy way to improve the performance of the method on thin objects is first to identify the thin parts to be preserved and then to increase the margin values for these voxels. The thin parts can be identified using simple morphological operations. Thus, we perform morphological *opening* of the binary volume by eroding it in a 6-connected neighborhood, and subsequently dilating it in a 26-connected neighborhood. We then consider all voxels that are inside the object in the original volume but not inside the opening as protrusions that require further processing and raise their margin values m_{ijk} (which the rule (7) always set to zero in these parts) to $\epsilon = 0.25$. As shown in (Figure 4c), this improves the performance of the method in protrusion areas considerably (although, admittedly, the aliasing artifacts can be observed there). Importantly, the surface remains virtually unchanged in other parts.

Accuracy. The main criterion of the isosurface extraction method is the visual consistency of the produced meshes. This, however, is hard to quantify, and, therefore, we performed quantitative comparisons of the produced meshes in the following way. We took two binary volumes at high resolution (rasterized ball at 400^3 resolution, and "bunny" volume at approximately 512^3 resolution). We then downsampled them by a factor of 5 using nearest neighbor interpolation, and reconstructed the meshes from the downsampled volumes. We then looked at the distribution of the signed distances from the zero isosurface for the vertices of the produced meshes, whereas the distances were computed from the zero-isosurface of the *high*-resolution volume and measured in voxels of that volume.

Table 5 reveals that mean squared distance for the three methods (extracting zero-isosurface from the low-resolution volume, performing minimum area reconstruction, and using our method) was very close for both methods, our method being only marginally better. At the same time, the mean *signed* distance reveal the characteristic shrinking bias of the minimal area approach towards positive signed distance (positive = "interior").

For the ball dataset, we also computed the normals at vertices (by averaging the normals of the adjacent faces), and looked at the squared angles between these normals and the normals of the perfect sphere. Here in the orientation domain, the accuracy advantage of our method was very significant, namely one and two order of magnitude over minimal area and zero-isosurface approaches, as the bottom line of Table 5 (as well as the bottom row of Figure 1) demonstrates.

Runtimes. We give the runtimes for the main stages of



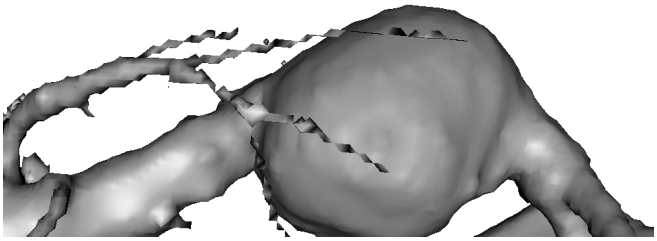
(a) Zero-isosurfaces of binary volumes *(b)* Minimal area separating surfaces *(c)* Our separating surfaces

Figure 3. For a set of synthetic and real-data binary volumes, our method extracts separating surfaces with less aliasing artifacts, as compared to the minimal area approach or the straightforward application of the marching cubes. See the text for the description of the binary volumes.

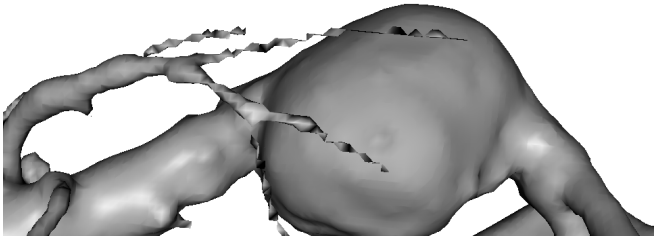
our algorithm for several datasets. The run-times are given for the narrow band half-width $C = 4$; the Jacobi update (10) was performed 1000 times.

The timings for the three main stages and for a number of datasets are given in Table 5. The set-up stage involved computing the distance transform, identifying narrow band, and preparing the matrix for the quadratic programming. The main stage (quadratic programming via Jacobi iterations) was performed either on a desktop Intel 2.4 GHz CPU, or on the entry-level[2] NVIDIA Quadro FX 580 GPU card (high-end GPU cards are thus expected to bring further speed-up). The timings are given for an excessive number of iterations (1000), while much smaller number may already produce a surface, that is sufficiently smooth (Figure 5). For the last stage (marching cubes), we used the

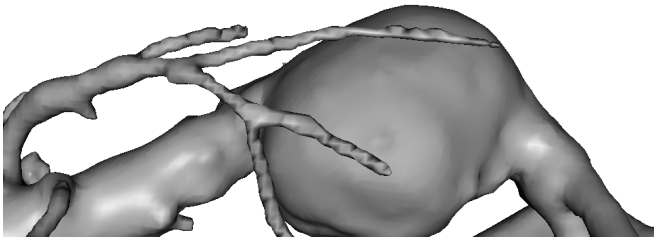
[2] According to NVIDIA web-site.



(a) – Minimal area separating surface



(b) – Our separating surface



(c) – Our separating surface, modified margin

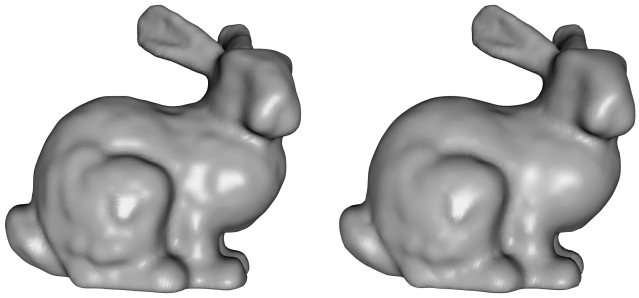
Figure 4. Close-ups of the surfaces extracted from the high-resolution binary volumes with different methods. Both the minimal area approach and our approach struggle to obtain visually consistent surfaces for very thin objects such as vessels. The performance of our method in these regions may be improved by automatically increasing the margin values locally; in other parts, the surface remains unchanged.

Volume – Measurement	*0-isosurf.*	*Minim.area*	*Ours*
Ball – Mean Sq. Dist.	15.05	14.86	14.06
Bunny – Mean Sq. Dist.	16.73	15.96	15.61
Ball – Mean Signed Dist.	-0.13	**0.59**	0.04
Bunny – Mean Signed Dist.	-0.23	0.25	-0.2
Ball – Mean Sq. Angle	**0.06**	**0.003**	**0.0003**

Table 1. Comparative distance and angle measurements for the isosurface reconstruction from downsampled volumes. Mean squared and mean signed distances to the zero-isosurface of the volumes before downsampling are given for the three methods (zero-isosurface, minimal area, our method). The last line correspond to angles between vertex normals and vertices of the perfect sphere (in radians).

MATLAB's `isosurface` function.

As for the methods performing minimal area extraction, [3] reported 1.3–2.0 sec on a CPU presumably similar to ours for a very similar geometry to 'Sphere-100' in our table, [26] reported the runtimes of 1–5 minutes on an SGI 185 MHz workstation for the volume sizes in the same ballpark as ours.



after 100 iterations after 1000 iterations

Figure 5. Even when our computation scheme is not run until convergence, the result may be a sufficiently smooth surface (left).

Dataset	Sphere-100	Sphere-200	Bunny	Bone
Volume Size	100^3	200^3	100x100x80	128x128x60
Mesh Size(tri)	31,496	128,600i	54,892	19,010
Setting up	1.05 sec	6.18 sec	1.55 sec	0.77 sec
QP on GPU*	0.85 sec	3.33 sec	7.85 sec	3.05 sec
(QP on CPU)	(5.42 sec)	(21.87 sec)	(10.54 sec)	(3.68 sec)
March. Cubes	0.13 sec	0.93 sec	0.23 sec	0.09 sec

Table 2. Sample runtimes of the main stages of our algorithm. *Note that an entry-level GPU card was used.

6. Discussion

We have presented a simple algorithm allowing to extract smooth isosurfaces from binary volumes, which is a common post-processing task within a range of shape reconstruction applications, in particular 3D image segmentation. Unlike previous methods, our approach operates within the implicit framework and minimizes a higher-order smoothness criterion imposed on the embedding function. Such minimization can be achieved via convex quadratic programming and yields smooth isosurfaces with fewer aliasing artifacts. It remains an interesting question whether similar kind of higher-order smoothness can be used for other tasks, e.g. whether it can be applied to image segmentation directly.

Our approach, thus, presents a viable alternative to the approaches that impose the higher-order smoothness within the evolution process after the mesh is extracted [12,19,20]. One advantage of imposing the higher-order smoothness within our framework is the convexity of the resulting optimization problem. Furthermore, unlike [12, 19, 20] the optimized higher-order smoothness functional is independent from the mesh structure defined by marching cubes. Finally, our method seems to be more suitable for the class of techniques that avoid extracting the mesh isosurface and

work with the signed distance field representation directly [10, 13]. On the downside, as our method relies on marching cubes to extract the final isosurface, it has no control on the quality of the produced triangles, and as a large number of triangles with poor aspect ratio may appear further remeshing operation [2] may be needed. One remedy for that may be using the dual marching cube algorithm [19] rather than the primal one for the isosurface extraction.

While we have focused on the surface extraction from the binary volumes, it is highly likely that higher-order smoothness will be useful for the segmentation results produced by the methods working with continuous representations, as the criteria within these methods are not designed to extract isosurfaces that are smooth at *subvoxel* levels. E.g., the level-set frameworks [22] typically optimize minimal area-related objectives and therefore isosurfaces extracted from the resulting continuous-valued volumes will be very similar to the minimal area separating surfaces computed with the level set method [26]. Similarly, the random walker algorithm [11] permits subvoxel isosurface extraction, yet enforce the first-order smoothness on the embedding function. Therefore, we believe that imposing higher-order smoothness may be as useful for the extraction of the surfaces from the segmentation results of all these methods as it is for the segmentation methods with binary outputs.

It can be argued, at the same time, that for such tasks as multiview reconstruction or shape-from-points the input data may be reused at the surface extraction stage to achieve the subvoxel accuracy (since unlike image segmentation, the initial data for these problems are not sampled on a grid and typically have higher effective resolution). Still, our method may be useful for the approaches solving these problems within the increasingly popular graph-cut [17, 24] and TV-minimization [14] frameworks as a fast, "ready-to-use" solution for the surface extraction.

The code of the approach is available at the webpage of the author[3].

7. Acknowledgements

The author gratefully acknowledges the discussions with Yuri Boykov, Carsten Rother, Toby Sharp, and Leo Grady. The 3D visualizations were produced using the the *Scanalyze* renderer available at http://graphics.stanford.edu/software/scanalyze/. For the implementation of the distance transform algorithm [23], the implementation code published by Dr. David Coeurjolly on his webpage was used. The hip joint dataset was taken from the UWO Computer Vision group repository at http://vision.csd.uwo.ca/viki/Datasets. Range scans for the Bunny model were taken from the Stanford 3D Scanning Repository.

[3] At the time of the publication: http://www.robots.ox.ac.uk/%7evilem/

References

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6), 1994.

[2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In L. D. Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*. Springer, 2007.

[3] R. Bade, O. Konrad, and B. Preim. Reducing artifacts in surface meshes extracted from binary volumes. v. 15, 2007.

[4] M. Bertram, G. Reis, R. H. van Lengen, S. Köhn, and H. Hagen. Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart. *EuroVis*, 2005.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *ICCV*, 2001.

[7] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. *ICCV*, 2003.

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH*, 1996.

[9] S. F. F. Gibson. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. *MICCAI*, 1998.

[10] S. F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. *IEEE Visualization*, 1998.

[11] L. Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11), 2006.

[12] K. Hildebrandt and K. Polthier. Constraint-based fairing of surface meshes. *Symposium on Geometry Processing*, 2007.

[13] M. W. Jones, J. A. Bærentzen, and M. Srámek. 3d distance fields: A survey of techniques and applications. *IEEE Trans. Vis. Comput. Graph.*, 12(4), 2006.

[14] K. Kolev and D. Cremers. Integration of multiview stereo and silhouettes via convex functionals on convex domains. *ECCV (1)*, 2008.

[15] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *ICCV*, 1999.

[16] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2), 1994.

[17] V. Lempitsky and Y. Boykov. Global optimization for shape fitting. *CVPR*, 2007.

[18] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 1987.

[19] G. M. Nielson. Dual marching cubes. *IEEE Visualization*, 2004.

[20] G. M. Nielson, G. Graf, R. Holmes, A. Huang, and M. Phielipp. Shrouds: Optimal separating surfaces for enumerated volumes. *VisSym*, 2003.

[21] M. Nikolova, S. Esedoglu, and T. F. Chan. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal of Applied Mathematics*, 66(5), 2006.

[22] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79, 1988.

[23] T. Saito and J.-I. Toriwaki. New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition*, 27(11), 1994.

[24] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. *CVPR (2)*, 2005.

[25] S. W. Wang and A. E. Kaufman. Volume-sampled 3d modeling. *IEEE Comput. Graph. Appl.*, 14(5), 1994.

[26] R. T. Whitaker. Reducing aliasing artifacts in iso-surfaces of binary volumes. *Volume Vizualization*, 2000.