Smooth Shape-Based Interpolation using the Conjugate Gradient Method

Balázs Csébfalvi, László Neumann*, Armin Kanitsar, Eduard Gröller

Vienna University of Technology
Institute of Computer Graphics and Algorithms
Favoritenstraße 9-11 / E 186, Vienna, Austria, A-1040
Email: {csebfalvi, kanitsar, groeller}@cg.tuwien.ac.at

Abstract

In this paper a novel technique for smooth shape-based interpolation of volume data is introduced. Previously simple linear interpolation of signed distance maps has been used in practice. As it will be shown, this approach results in artifacts, since sharp edges appear along the original slices. In order to obtain a smooth 3D implicit function generated by interpolating 2D distance maps, we use a global interpolation method instead of a higher order local technique. The global curvature of the implicit function representing an isosurface is minimized using an iterative conjugate gradient method. Because of the iterative approach the user can easily control the trade-off between the smoothness of the isosurface and the computational cost of the refinement. As opposed to previous techniques, like variational interpolation, our method can generate a reasonably good approximation of the ideal solution in a significantly shorter time.

1 Introduction

In computer graphics research, it is a recurring problem, how to reconstruct a continuous function from a finite number of samples located at regular grid points. The traditional approach is convolution-based filtering. The *sinc* function is considered to be an ideal reconstruction kernel since it represents an ideal low-pass filter in frequency domain [1, 2]. Theoretically, the original signal can be reconstructed if it is band-limited and the sampling density is higher than the Nyquist limit. In practice, it is not always the case. For instance, in a medical CT scan the resolution within slices is usually much higher than the resolution along the axis perpendicular to the slices (often called z-axis). Typically the sampling distance between the slices is insufficient for reconstructing small details. If the boundary surface of an organ is varying drastically along the z-axis the local interpolation methods result in staircase aliasing. The question is, what interpolation technique can be considered to be optimal in such a case. For example, if it is assumed that the data contains some smooth surfaces of different organs, it is worthwhile to aim at the curvature minimization of the interpolated function rather than using a local reconstruction method.

Many volume rendering algorithms require an isostack of slices, where the inter-slice distance is equal to the inter-pixel distance inside the slices. In order to fulfill this requirement, usually a more sophisticated interpolation method (like cubic B-spline interpolation) is applied for generating an isotropic data set, and a simple trilinear interpolation is used in the rendering pipeline [3] to obtain a continuous representation. Our strategy is similar. We want to select a subvolume of interest and create a higher resolution, nearly isotropic volume out of it using a high-quality shape-based interpolation technique. This higher resolution representation of the selected region can be visualized by the traditional volume-rendering methods. In this sense, our approach can be interpreted as a sharp volume-zooming technique.

We assume that some fine details are contained in the selected subvolume, which cannot be reconstructed by local convolution-based filtering because of undersampling along the z-axis. Therefore, instead of interpolating the original density values, we apply the traditional shape-based approach by interpolating 2D implicit functions generated from

*ICREA Researcher at the University of Girona, Email: lneumann@ima.udg.es

the contours inside the slices. Our goal is to create a higher-resolution representation of the original subvolume. We do this by adding intermediate slices aiming at a minimal global curvature. The newly inserted intermediate samples are free variables, while the samples of the original 2D implicit functions have to be preserved. Although by a simple linear interpolation of implicit functions, like signed distance maps, smoother surfaces can be obtained than by linear density interpolation, some artifacts like sharp edges at the position of the original slices are still produced (see figure 1).

In contrast our interpolation technique provides smooth isosurfaces. All the 2D slices are interpolated at the same time using an iterative method, where the global curvature is minimized.

Section 2 gives an overview on the previous shape-based interpolation techniques discussing their advantages and drawbacks. In section 3 our novel approach based on global curvature minimization is introduced. In section 4 we go into the details of the iterative solution using the conjugate gradient method. In section 5 our method is tested on artificial and medical data sets and finally in section 6 the contribution of this paper is summarized.

2 Previous Work

The previous shape-based interpolation techniques can be classified into two fundamentally different categories: *parametric correspondence methods* and *implicit function interpolation*. The parametric methods try to find corresponding pairs of points between the boundaries of two 2D shapes. Afterwards the intermediate samples are interpolated from these pairs of points. An early application of this approach was a contour interpolation technique published by Fuchs et al. [4]. The idea was to find a minimal-area triangulation connecting two corresponding contours. Further improvements were also published like defining quality measures for the correspondence between contours and optimizing the basic method [5, 6]. The major drawback of the parametric methods is the problem of self-intersecting surfaces and branching which are addressed in recent publications [8, 9].

Applying implicit function interpolation these problems do not arise, and shapes of different topologies can be easily transformed. Implicit function methods operate on a higher dimensional representation of objects than the parametric methods do. Therefore, they have higher computational costs and memory requirements.

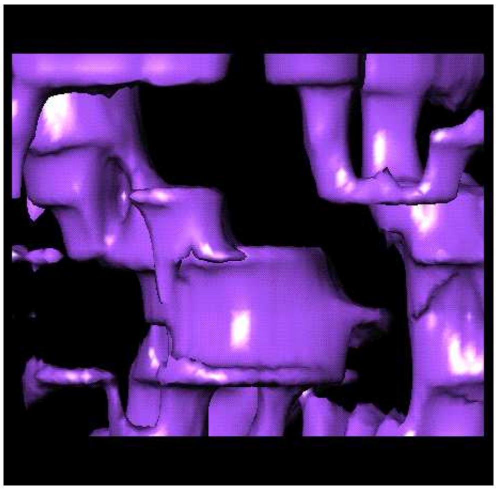
A contour inside a slice can be represented by a 2D implicit function $f(\mathbf{x})$, as a set of all points $\mathbf{x}$ such that $f(\mathbf{x}) = 0$. A 2D implicit function can be interpreted as a height field, where the contour is defined as a level line intersecting the height field at the zero level. Intermediate contours can be generated by interpolating 2D implicit functions and searching for the contour points at the zero level.

One possible implicit function is a binary *inside/outside function* or *characteristic function*. Such implicit functions are rather simple. They can only be used for contour transformation if a more sophisticated interpolation method is applied. An alternative solution is to perform the interpolation of characteristic functions in the frequency domain [7]. Smooth intermediate shapes can also be generated from characteristic functions by using weighted Minkowski sums [10, 11].

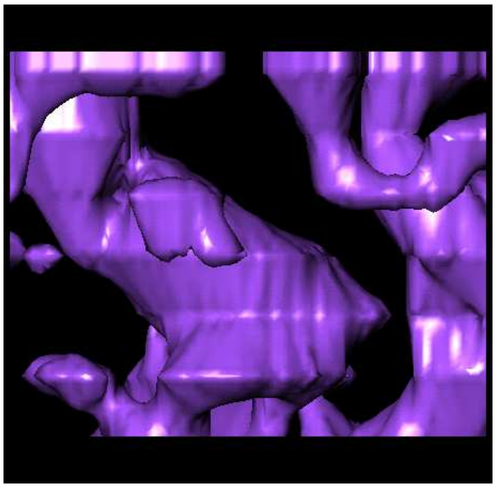
More sophisticated implicit functions like *signed distance maps* provide smooth contour transitions even if a simple interpolation technique is used [12, 13, 14, 15]. In order to interpolate contours, the 2D distance maps are evaluated at regular grid points. Therefore, each pixel contains the signed Euclidean distance to the nearest contour point in the given slice. The pixels with negative distance are outside the object and the pixels with positive distance are inside the object. An intermediate implicit function is created by a simple linear interpolation and each intermediate contour is determined as a zero isocontour in an interpolated implicit function. This method results in quite good results. Because of the local interpolation, however, some second order discontinuities are produced.

An alternative solution to this problem is the application of *variational interpolation* [16]. This approach simultaneously builds implicit functions and interpolates between them. Although this method generates smooth surfaces from contours, it is computationally rather expensive. Furthermore, it provides an analytic solution which has to be evaluated at regular grid points to create an input data for volume rendering.

In contrast our method is based on an iterative global smoothing of the initial 3D implicit function which is produced by linear interpolation. Therefore, the user can easily control the trade-off be-



(a) Linear density interpolation.



(b) Linear interpolation of 2D distance maps.

Figure 1: Density- (a) and shape-based (b) interpolation in a CT scan of a human colon.

tween the smoothness of the interpolated surfaces and the computational cost of the refinement. On the other hand the generated result is a volumetric representation, therefore it is directly available for a volume-rendering application.

Even if an isosurface is reconstructed from the volume using the marching cubes algorithm [18], the vertex normals can be estimated from the neighboring voxels, therefore a smooth Gouraud-shaded surface can be rendered. In contrast, using the recent methods which generate triangular meshes from contours [8, 9] the vertex normals have to be derived from the orientation of the triangles.

3 Curvature Minimization

The input data of our method is a selected subvolume of resolution $X \times Y \times Z$, which contains some fine details of interest. It is assumed that along the z-axis the object to be visualized is undersampled. Our interpolation algorithm consists of the following steps:

1. contours specified by a threshold are generated within 2D slices
2. 2D signed distance maps are created for each slice
3. intermediate slices are added through linear interpolation of the distance maps
4. iterative refinement using the conjugate gradient method
5. extracting an isosurface crossing the zero level

First of all, contours specified by an isovalue are generated in each slice using the 2D version of the marching cubes algorithm [18]. The next step is the generation of an implicit function for each slice by calculating signed distance maps of the contours. The marching cubes algorithm provides a piecewise linear approximation of a contour. Therefore, for each single segment its Euclidean distance from an arbitrary pixel position can be easily determined. It is sufficient to evaluate the signed distances just for a certain pixel neighborhood of each segment, where the radius depends on the number of intermediate slices to be interpolated. After processing all the segments, each pixel near the contours contains the signed distance to the nearest contour segment. Figure 2 shows an example of such a 2D implicit function, where blue pixels (negative distance) are outside the object and red pixels (positive distance) are inside the object.

In the third step the resolution along the z-axis is increased by a factor of N by linearly interpolating $N-1$ slices between each neighboring pair of the 2D distance maps. The obtained distance volume of resolution $X \times Y \times ((Z-1) \cdot N + 1)$ is used in the following steps as a discrete 3D implicit function

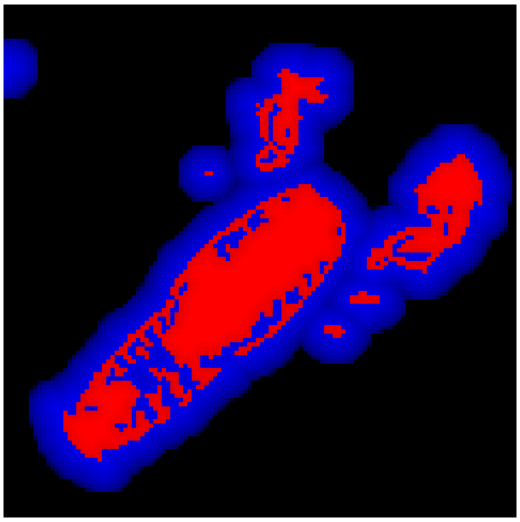


Figure 2: An example of a 2D signed distance map.

denoted by $f(i, j, k)$.

The fourth step is an iterative refinement minimizing the global curvature of function f. At each single voxel position (i, j, k), the local error is defined as the square of the curvature $c(i, j, k)$, which can be calculated by finite differences approximating the following formula:

$$c(i,j,k) = f_{xx}^2(i,j,k) + f_{yy}^2(i,j,k) + f_{zz}^2(i,j,k) + 2 \cdot f_{xy}^2(i,j,k) + 2 \cdot f_{xz}^2(i,j,k) + 2 \cdot f_{yz}^2(i,j,k). \tag{1}$$

The global error E is defined as the sum of the local errors:

$$E = \sum_{k=1}^{Z'} \sum_{j=1}^{Y} \sum_{i=1}^{X} c(i,j,k)^2, \tag{2}$$

where $Z' = (Z - 1) \cdot N + 1$. At the minimum location of the error function E, its partial derivatives according to all the distance values $f(i, j, k)$ have to be equal to zero, where $i \in [1, 2, ..., X]$, $j \in [1, 2, ..., Y]$, $k \in [1, 2, ..., Z']$. This is equivalent to the gradient vector ∇E) being equal to zero:

$$\partial E / \partial f(i,j,k) = 0 \Rightarrow \nabla E = 0. \tag{3}$$

The global error function can be minimized by applying an iterative gradient method. Note that, the distance maps of the original slices ($k \in [1, N+1, 2N+1, ..., Z']$) should not be modified during the iteration. This requirement can be fulfilled if the iteration is performed in a lower dimensional subspace. Let us denote the vector of all the distance values by $\mathbf{f}$. Using the classical gradient method, in the ith iteration step vector $\mathbf{f}^{i+1}$ is calculated as follows:

$$\mathbf{f}^{i+1} = \mathbf{f}^i - \alpha \cdot \nabla E^i. \tag{4}$$

In order to leave original slices unchanged we are not using the gradient vector ∇E but a projection $\nabla E'$ onto a subspace. $\nabla E'$ results from ∇E by setting all those components of ∇E to zero which correspond to original slices. Thus, during the iterative minimization, vector $\mathbf{f}$ converges to the smoothest solution which fits to the original slices.

Since the error function E is quadratic, its partial derivatives are calculated as a linear function of the distance values $f(i, j, k)$. The curvature formula (1) is approximated by finite differences, therefore the partial derivatives, which are the components of the gradient vector ∇E, can be determined by a computationally efficient convolution with a $5 \times 5 \times 5$ kernel (see Appendix). Practically, each iteration step is a simple filtering operation.

Nevertheless, applying the classical gradient method the iteration converges slowly. In order to speed up the convergence we rather use an optimized conjugate gradient method [17] although its memory requirements are higher.

4 Iteration using the Conjugate Gradient Method

In each iteration step, the conjugate gradient method translates vector $\mathbf{f}$ into a direction $\mathbf{d}$ so that one component of the error becomes zero and stays that way in the further iterations. Thus, if the number of components is N then after N iterations ∇E would be zero for sure. In an iteration step, we are not going into the $-\nabla E$ direction but in a modified direction depending on the previous step:

$$\mathbf{f}^{i+1} = \mathbf{f}^i + \alpha^i \cdot \mathbf{d}^i, \tag{5}$$

where $\mathbf{d}^i = \beta^i \cdot \mathbf{d}^{i-1} - \nabla E^i$. The initial value of vector $\mathbf{d}$ is defined as $\mathbf{d}^0 = -\nabla E^0$. The optimal scaling factor β^i in the ith iteration step is calculated from the current and the previous gradient vector in the following way:

$$\beta^i = \frac{(\nabla E^i - \nabla E^{i-1})^T \cdot \nabla E^i}{(\nabla E^{i-1})^T \cdot \nabla E^{i-1}}. \qquad (6)$$

The formula for the optimal calculation of the scaling factor α is derived from the current values of $\mathbf{f}$ and $\mathbf{d}$ (see Appendix). Thus, the major steps of the conjugate gradient method are the following:

1. calculate $\mathbf{f}^0$ by linear interpolation
2. $\mathbf{d}^0 = -\nabla E^0$
3. calculate an optimal α^i
4. $\mathbf{f}^{i+1} = \mathbf{f}^i + \alpha^i \cdot \mathbf{d}^i$
5. calculate an optimal β^{i+1}
6. $\mathbf{d}^{i+1} = \beta^{i+1} \cdot \mathbf{d}^i - \nabla E^{i+1}$
7. if $|\mathbf{d}| > \epsilon$ go to 3.

In order to keep the original slices unchanged, we slightly modify this classical conjugate gradient method by substituting ∇E in steps 2 and 6 by its projection $\nabla E'$ as it is defined in the previous section. According to our experience, the conjugate gradient method converges much faster (the number of necessary iterations is less with an order of magnitude) than the simple "steepest slope" gradient method. In the conjugate gradient method one iteration step is more complicated and an additional floating point field has to be allocated for each voxel in order to calculate the optimal α_i coefficients. Because of the lower number of necessary iterations the computational cost is, however, significantly reduced.

5 Implementation

The presented iterative smoothing technique has been implemented in C++ and tested on a *Silicon Graphics O2* workstation. First we used two artificial data sets for testing, where the 2D signed distance maps were generated by calculating accurate Euclidean distances. One of these data sets was a simple sphere and the other one was a cross-like shape of tubular objects. In both cases an isotropic volume was created and we tried to reconstruct the original volume from a lower resolution representation taking every eighth slice as an original slice. Figure 3b and 3d show the results after 15 iterations. Compared to the linear interpolation of the distance maps (figure 3a and 3c) the surfaces are much smoother and the second-order discontinuities are significantly reduced.

We also tested our method on real medical data. From a CT scan of a colon a $64 \times 64 \times 64$ subvolume

	$\nabla E'$	β^i	α^i	$\mathbf{f}^{i+1}$
time	0.69 s	0.09 s	0.81 s	0.14 s

Table 1: Running time measurements for the different steps of the conjugate gradient method.

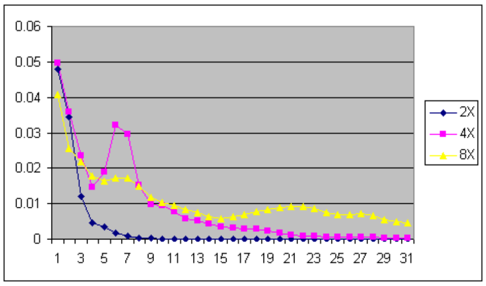


Figure 5: The magnitude of vector $\mathbf{d}$ depending on the number of iterations, where the z-resolution was increased by a factor of 2, 4, and 8.

of interest was selected. Figure 4 shows the test results for this subvolume. Similarly to the previous example, we tried to reconstruct the original subvolume by taking every fourth then every eighth slice as an original slice. Linear interpolation of the distance maps (4a and 4c) results in sharp edges along the original slices while using our method much smoother surfaces are produced after 15 iterations. These 15 iterations took 26 seconds on an *SGI O2* workstation. Table 1 shows the running time measurements in seconds for the different phases of one iteration step.

In order to illustrate the nature of the convergence, figure 5 shows the magnitude of vector $\mathbf{d}$ which is used to slightly modify the distance values $f(i, j, k)$ in each iteration step. The different graphs belong to different ratios of the original and the new resolution along the z-axis. For instance, if this ratio is 4 then every fourth slice is considered to be an original one, therefore 3 intermediate slices are generated between each pair of neighboring original slices. Note that, the more intermediate slices have to be generated the slower the convergence is, because of the increased number of free variables. For a typical medical data set it is usually sufficient to increase the z-resolution by a factor of 2 in order to generate a nearly isotropic volume. In such a case after about 10 iterations the magnitude of vector $\mathbf{d}$ will be practically zero.

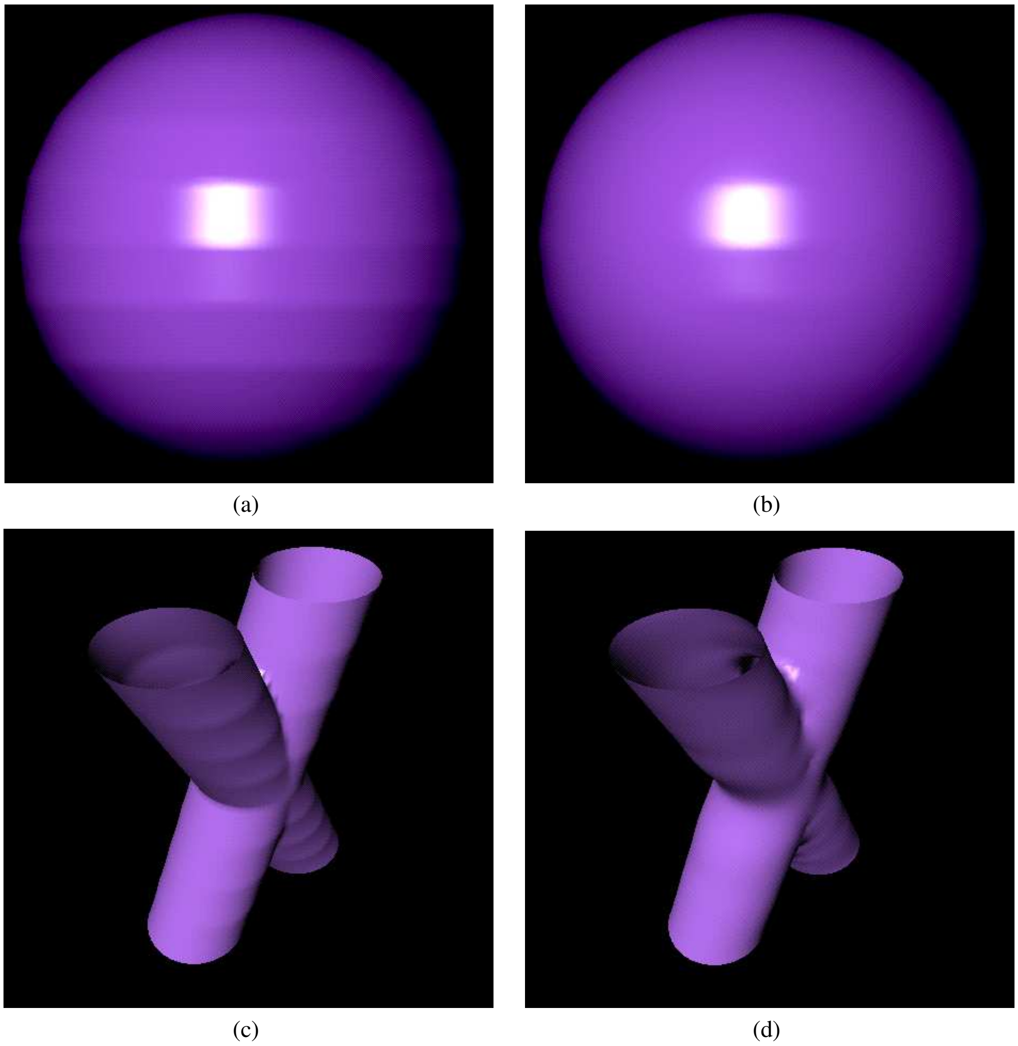


Figure 3: Artificial test data generated by linear interpolation of 2D signed distance maps (a, c) and by using our iterative smoothing method (b, d).

6 Conclusion

In this paper a smooth shape-based interpolation technique based on an iterative refinement approach has been presented. We have shown that the traditional linear interpolation of signed distance maps can result in sharp edges along the original slices if the surfaces are varying drastically in the z-direction. These artifacts can be eliminated by our method which minimizes the global curvature of the generated 3D implicit function. Similarly to the variational interpolation our technique produces all the intermediate slices simultaneously instead of interpolating between pairs of 2D implicit functions. Therefore, second-order discontinuities do not appear along the original slices and the smoothness of the surfaces is comparable to the results of variational interpolation.

Furthermore, in our case the trade-off between

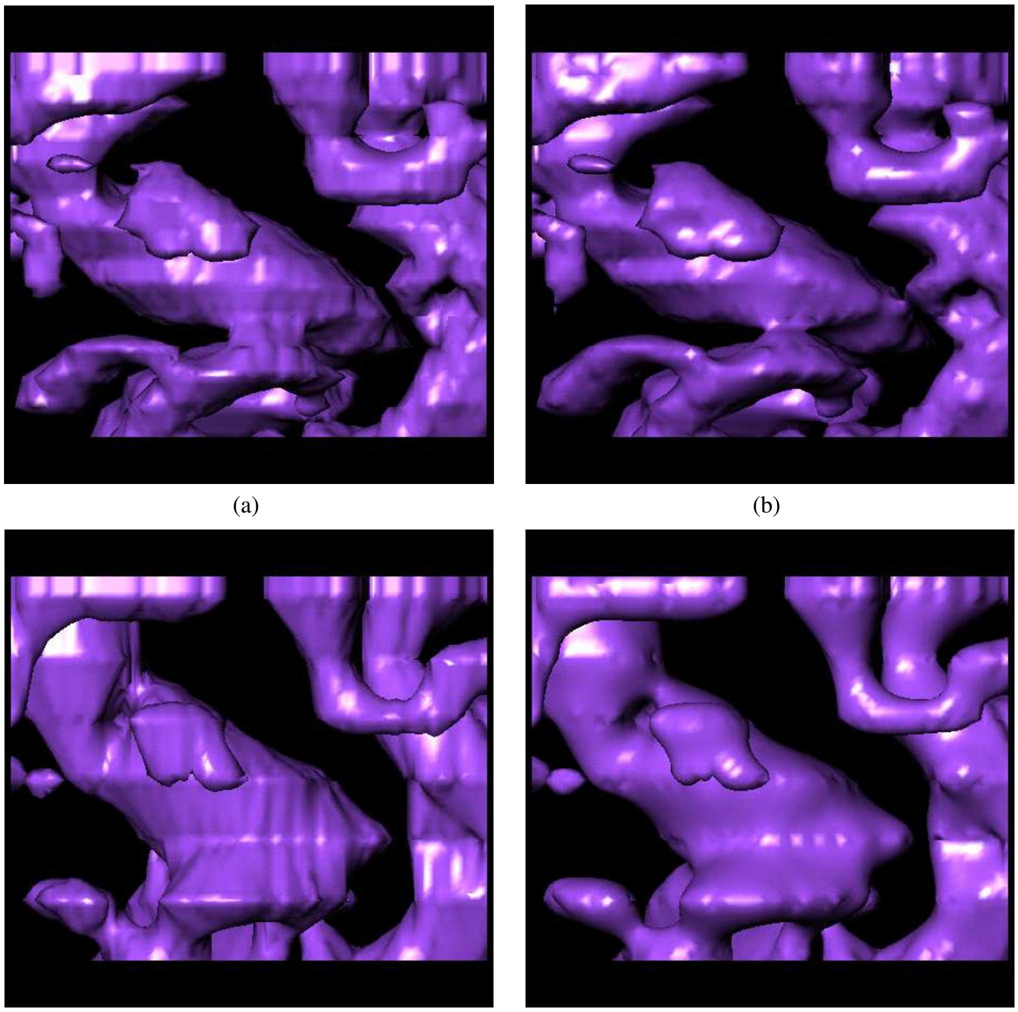


Figure 4: Interpolation of a human colon by taking every fourth (a, b) and every eighth slice (c, d) as an original slice. For the volume reconstruction linear interpolation of approximate signed distance maps (a, c) and our iterative smoothing method (b, d) were used.

the image quality and the computational cost can be easily controlled by the user because of the iterative approach. Thus a good approximation can be obtained in a significantly shorter processing time. In contrast, variational interpolation calculates an analytical solution which is computationally rather expensive. Additionally this analytical solution has to be evaluated at regular grid points to build a volumetric data for an interactive volume-rendering application.

Acknowledgements

The work presented in this publication has been funded by the ADAPT project (FFF-804544). ADAPT is supported by *Tiani*

Medgraph, Vienna (http://www.tiani.com), and the *Forschungsförderungsfonds für die gewerbliche Wirtschaft*, Austria. See http://www.cg.tuwien.ac.at/research/vis/adapt for further information on this project.

References

[1] M. J. Bentum, B. B. A. Lichtenbelt, T. Malzbender, Frequency Analysis of Gradient Estimators in Volume Rendering, *IEEE Transactions on Visualization and Computer Graphics*, 2(3), pages 242–254, September, 1996.

[2] S. C. Dutta Roy, B. Kumar, Digital Differentiators, in *Handbook of Statistics*, (N. K. Bise and C. R. Rao eds.), vol. 10, pages 159–205, 1993.

[3] M. Levoy, Display of surfaces from CT data, *IEEE Computer Graphics and Application*, 8(5), pages 29–37, 1988.

[4] H. Fuchs, Z. M. Kedem, S. P. Uselton, Optimal Surface Reconstruction from Planar Contours, *Communications of the ACM*, 20(10), pages 693-702, October, 1977.

[5] D. Meyers, S. Skinner, Surfaces From Contours: The Correspondence and Branching Problems, *Proceedings of Graphics Interface '91*, pages 246–254, 1991.

[6] T. W. Sederberg, E. Greenwood, A Physically Based Approach to 2D Shape Blending, *Computer Graphics (Proceedings of SIGGRAPH '92)*, 26(2), pages 25–34, 1992.

[7] H. F. Hugues, Scheduled Fourier Volume Morphing, *Computer Graphics (Proceedings of SIGGRAPH '92)*, 26(2), pages 43–46, 1992.

[8] C. L. Bajaj, E. J. Coyle, K. N. Lin, Surface and 3D Triangular Meshes from Planar Cross Sections, *5th International Meshing Roundtable*, Sandia National Laboratories, pages 169–178, 1996.

[9] J. M. Olivia, M. Perrin, S. Coquillart, 3D Reconstruction of Complex Polyhedral Shapes from Contours using a Simplified Generalized Voronoi Diagram, *Computer Graphics Forum (Proceedings of EUROGRAPHICS '96)*, pages 397–408, 1996.

[10] A. Kaul, J. Rossignac, Solid- Interpolating Deformations: Construction and Animation of PIPs, *Computer Graphics Forum (Proceedings of EUROGRAPHICS '91)*, pages 493–505, 1991.

[11] J. Rossignac, A. Kaul, AGRELs and BIPs: Metamorphosis as a Bezier Curve in the Space of Polyhedra, *Computer Graphics Forum (Proceedings of EUROGRAPHICS '94)*, pages 179–184, 1994.

[12] G. T. Herman, J. Zheng, C. A. Bucholtz, Shap-Based Interpolation, *IEEE Computer Graphics and Applications*, 12(3), pages 69–79, 1992.

[13] L. David, Multidimensional Reconstruction by Set-valued Approximation, *IMA J. Numerical Analysis*, 6, pages 173–184, 1986.

[14] P. Bradley, A. W. Toga, Distance Field Manipulation of Surface Models, *IEEE Computer Graphics and Applications*, 12(1), pages 65–71, 1992.

[15] D. Cohen-Or, D. Levin, A. Solomovici, Three Dimensional Distance Field Metamorphosis, *ACM Transactions on Graphics*, 1997.

[16] G. Turk, J. F. O'Brien, Shape Transformation Using Variational Implicit Functions, *Computer Graphics (Proceedings of SIGGRAPH '99)*, pages 335–342, 1999.

[17] J. R. Shewchuk, An Introduction to the Conjugate Gradient Method, `http://www.csc.fi/math_topics/Mail/NANET94/msg00543.html`, 1994.

[18] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics*, 21(4), pages 163–169, 1987.

Appendix

The components of vector ∇E can be calculated by a simple $5 \times 5 \times 5$ convolution kernel. Let us denote the layers of this kernel by $l_1, l_2, ..., l_5$. Because of symmetry reasons $l_1 = l_5$ and $l_2 = l_4$. The layers are defined by the following matrices:

$$l_1 = \begin{bmatrix} 0 & 0 & 0.125 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.125 & 0 & 0.5 & 0 & 0.125 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.125 & 0 & 0 \end{bmatrix},$$

$$l_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$l_3 = \begin{bmatrix} 0.125 & 0 & 0.5 & 0 & 0.125 \\ 0 & 0 & -4 & 0 & 0 \\ 0.5 & -4 & 19.5 & -4 & 0.5 \\ 0 & 0 & -4 & 0 & 0 \\ 0.125 & 0 & 0.5 & 0 & 0.125 \end{bmatrix}.$$

In the third step of the conjugate gradient method the optimal α can be calculated by the following formula:

$$\begin{gathered}
\alpha = \sum_{k=1}^{Z'} \sum_{j=1}^{Y} \sum_{i=1}^{X} [(f_{i+1,j,k} + f_{i-1,j,k} - 2f_{i,j,k}) \cdot \\
(d_{i+1,j,k} + d_{i-1,j,k} - 2d_{i,j,k}) + \\
(f_{i,j+1,k} + f_{i,j-1,k} - 2f_{i,j,k}) \cdot \\
(d_{i,j+1,k} + d_{i,j-1,k} - 2d_{i,j,k}) + \\
(f_{i,j,k+1} + f_{i,j,k-1} - 2f_{i,j,k}) \cdot \\
(d_{i,j,k+1} + d_{i,j,k-1} - 2d_{i,j,k}) + 0.125 \cdot \\
((f_{i,j+1,k+1} + f_{i,j-1,k-1} - f_{i,j-1,k+1} - f_{i,j+1,k-1}) \cdot \\
(d_{i,j+1,k+1} + d_{i,j-1,k-1} - d_{i,j-1,k+1} - d_{i,j+1,k-1}) + \\
(f_{i+1,j,k+1} + f_{i-1,j,k-1} - f_{i+1,j,k-1} - f_{i-1,j,k-1}) \cdot \\
(d_{i+1,j,k+1} + d_{i-1,j,k-1} - d_{i+1,j,k-1} - d_{i-1,j,k-1}) + \\
(f_{i+1,j+1,k} + f_{i-1,j-1,k} - f_{i+1,j-1,k} - f_{i-1,j+1,k}) \cdot \\
(d_{i+1,j+1,k} + d_{i-1,j-1,k} - d_{i+1,j-1,k} - d_{i-1,j+1,k}))] / \\
[(f_{i+1,j,k} + f_{i-1,j,k} - 2f_{i,j,k})^2 + \\
(f_{i,j+1,k} + f_{i,j-1,k} - 2f_{i,j,k})^2 + \\
(f_{i,j,k+1} + f_{i,j,k-1} - 2f_{i,j,k})^2 + 0.125 \cdot \\
((d_{i+1,j,k+1} + d_{i-1,j,k-1} - d_{i+1,j,k-1} - d_{i-1,j,k-1})^2 + \\
(d_{i,j+1,k+1} + d_{i,j-1,k-1} - d_{i,j-1,k+1} - d_{i,j+1,k-1})^2 + \\
(d_{i+1,j+1,k} + d_{i-1,j-1,k} - d_{i+1,j-1,k} - d_{i-1,j+1,k})^2)].
\end{gathered}$$