Fast Generation of multiresolution surfaces from Contours

Andreas Schilling and Reinhard Klein

Universität Tübingen, Auf der Morgenstelle 10 / C9, 72076 Tübingen, Germany.
E-mail: {andreas—reinhard}@gris.uni-tuebingen.de
http://www.gris.uni-tuebingen.de

Abstract. Surface reconstruction from contours is an important problem especially in medical applications. Other uses include reconstruction from topographic data, or isosurface generation in general. The drawback of existing reconstruction algorithms from contours is, that they are relatively complicated and often have numerical problems. Furthermore, algorithms to generate multiresolution surface models do not exploit the special situation having contours.
In this paper we describe a new robust and fast reconstruction algorithm from contours that delivers a multiresolution surface with controlled distance from the original contours. Supporting selective refinement in areas of interest, this multiresolution model can be handled interactively without giving up accuracy.

1 Introduction and previous work

In medical applications very often tomography-techniques are used to acquire the data. These techniques deliver voxel data sets consisting of a staple of slices (images) where the distance of the slices in general is much larger than the pixel distance within one slice. Therefore, the data can also be considered as an unisotropically sampled voxel set (undersampled in one direction). The general approaches to reconstruct the surface are outlined in Fig. 1. There are two main approaches: a direct and an indirect approach. In the direct approach isosurfaces are extracted from the preprocessed voxel data set using the well known and fast Marching Cubes (MC) algorithm [18]. The preprocessing should perform a continuos classification of the original voxel data, so that the desired surface can be described as an isosurface of the voxel data set. Unfortunately, because of the lack of appropriate alternatives very often only very simple preprocessing like simple noise filtering is applied. If the data set is sampled sufficiently and if the classification doesn't pose problems this approach delivers good results,. If however the sampling in one direction is not sufficient, which is very common e.g. in medical applications, the linear interpolation in the MC algorithm depends on wrong assumptions and the resulting surfaces contains staircase artifacts revealing the slice structure. Furthermore, the resulting surfaces often consist of millions of triangles due to the regular space partitioning used in the MC algorithm. It was soon recognized that without mesh simplification techniques, models produced by the MC could not be handled [26]. As an alternative to the costly simplification algorithms several attempts on adaptive MC schemes with different but not fully satisfactory results [2, 28, 22] have been undertaken.

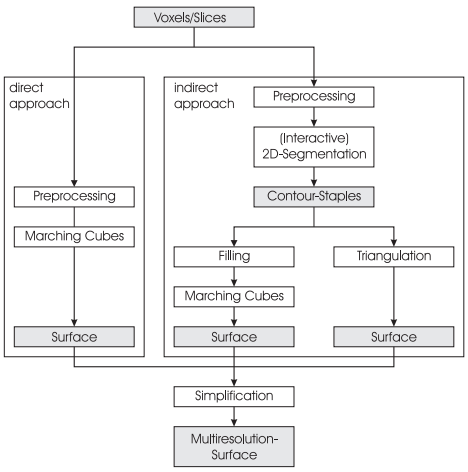

Fig. 1. The main approaches to surface reconstruction from volume data.

In the indirect approach an intermediate step is introduced: contours are extracted from the 2D-slices resulting in a staple of contours. For this purpose 2D-segmentation algorithms are used. In this approach interactive input or manipulation is possible to guide the segmentation. For the reconstruction of the surface from the contour staples again two alternatives are possible, either to voxelize the contours again by filling them and then using a MC algorithm or to construct a triangular facet surface by connecting two neighboring contours. Using the MC algorithm in this setting suffers from the already discussed problems: staircase artifacts (due to the undersampling and 1-bit quantization especially bothersome) and very large surface models. A large number of publications treat the problem of connecting contours, see [14, 8, 4, 17, 3, 20, 1, 23] and [19, 21, 27] for overviews. All these algorithms have to decide which vertices in the neighboring contour must be connected with a given vertex to form triangles. Unfortunately, this correspondence problem cannot be answered uniquely and suffers from the same problem of missing information caused by the undersampling of the original data like the MC algorithm.

1.1 The distance field interpolation

One possibility to overcome the problem of missing data between the slices is to interpolate the so called *distance field* between the contours [17, 24, 10, 7]. Using distance field interpolation, a triangulation of the isosurface can be obtained by applying the MC-algorithm to the unmodified distance-field. In such a way the stair-case artifacts normally produced by the MC-algorithm are avoided. In [13] this idea is realized without mentioning distance-field interpolation. Since our approach is also based on this principle we briefly review this method.

Let Ω be the 3D object and

$$\Omega_i = \{(x, y) | (x, y, z_i) \in \Omega\}$$

a finite set of cross sections. Then the distance fields at the levels $z_0, \ldots, z_n$ are defined by

$$D_i(x,y) = \begin{cases} -dist((x,y)\partial\Omega_i) & \text{if}(x,y) \in \Omega_i \\ dist((x,y)\partial\Omega_i) & \text{otherwise} \end{cases}$$

where $\partial\Omega_i$ denotes the boundary of Ω_i which is described by the the contours and $dist$ denotes the Euclidean distance within the slices. Now an interpolation of the distance values in z-direction is used to find intermediate contours (where the interpolated distance is zero). Another approach that can be considered as a special case of the distance field interpolation is the use of the medial axis between contours from neighboring slices [23]. The medial axis between the contours of slice i and $i-1$ is identical to the contour resulting from a distance field interpolation in the plane $z = 0.5 \cdot (z_{i-1} + z_i)$. In [23] an approximation of the medial axis is used as the basis of an elaborate reconstruction algorithm that allows to use multiple intermediate levels in cases where the geometry of subsequent contours is too different. Although in the paper nice results of this algorithm are shown, a fixed size model is produced. The number of triangles in the model depends on the number of vertices in the original contours and therefore, on the accuracy used in the approximation of the contours. The size of the resulting surface models is smaller than the size of a model produced by the MC algorithm if the contours are approximated with the same accuracy, but not sufficiently small for interactive rendering. The use of simplification techniques is therefore still necessary.

1.2 Simplification algorithms and multiresolution models

A large number of simplification algorithms for triangle meshes have been developed, but only a part of them can guarantee a certain geometric approximation error between simplified and original mesh [16, 6]. However, this error must be known to guarantee a certain quality for the rendered images of the simplified model. Unfortunately, the simplification algorithms with this feature are very slow (see [5] for a comparison) or produce over-estimations of the error [25, 9] that make the results useless for multiresolution models aiming for view-dependent refinement.

In the rest of the paper we describe a new robust and fast reconstruction algorithm combined with a simplification technique, that exploits the special situation we are faced with when reconstructing from contours. In this way a very efficient technique to measure the errors during the simplification process can be used, see 7.1. Section 2 gives a brief overview of the algorithm. Sections 3, 4, and 5 contain the reconstruction part of the algorithm and section 7 the simplification part.

2 Overview of the algorithm

In principle the algorithm consists of two main steps: the reconstruction of the surface and the simplification of this surface. However both steps are closely related since the distance field is used for the reconstruction as well as for the simplification step. The outline of the algorithm is as follows:

I. Reconstruction **1. Extraction of 2D-contours.** We start with a staple of segmented images, where for each pixel it is known if it belongs to the object or not. In the first step the boundary of the object is determined and the boundary pixels are marked and numbered sequentially. Each contour is identified with a unique number. To avoid topological problems, this step is performed in a grid containing not only the midpoints of the pixels but also their corners. Second, during the extraction of the contours adjacent slices are checked for overlapping areas to identify the connectivity of contours in different slices. **2. Simplification of contours.** After this, each contour is simplified up to a certain approximation error (we use half a pixel). In this way to every edge of a simplified contour the maximum geometric approximation error between the edge itself and the corresponding part of the original contour is known.

3. Computation of medial axes including correspondence. If we consider the simplified contour at the lowest resolution guaranteeing a maximum error of half a pixel, the pixels of the original contour can be regarded as one possible representation in image space and the original pixels can be classified as vertices or inner points of edges. Now, the distance field is computed and afterwards the medial axes between contours from different slices are extracted. To each pixel of the medial axes two pointers to the closest pixels on both contours are stored, see Fig. 2.

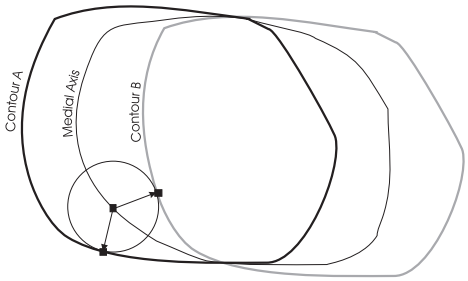

Fig. 2. The arrows from a pixel on the medial axis to the two contours A and B respectively indicate the correspondences to the closest pixels on the contours.

4. Surface triangulation. After the computation of the medial axes and the correspondences above, a triangulation of the resulting surface is computed. The main idea is to trace the medial axes and use the correspondences to pixels on the two neighboring contours to connect the vertices by edges.

II. Simplification **1. Edge collapse.** In the simplification algorithm simple edge collapse operations are performed [12]. The order of the edge collapse operations is determined by a priority queue based on the error that would be introduced if the edge was collapsed. **2. Error measurement.** To measure the error introduced by an edge collapse, the distance field computed above is exploited. The intersection lines of the newly created triangles with the respective slices are rendered (using Bresenham algorithm) into the images of the slices. While rendering the distance values are read from the distance map of the corresponding slice, see Fig. 8.

3 Simplification of contours

To simplify the contours a modified version of the Douglas-Peucker-algorithm is used [11]. This algorithm starts with one arbitrary vertex of the original contour. In each subsequent step a further point (with greatest distance to the current polygon) of the original

contour is inserted. To find this point with greatest distance a convex hull technique is used, that reduces the complexity of the algorithm from $O(n^2)$ to $O(n \log n)$.

4 Computation of the medial axes and correspondences

4.1 The distance field

For our purpose the application of a simple distance transform to compute the distance field and the medial axes is not sufficient, since we also want to know for each pixel which is (are) the closest pixels on the border. Therefore, we use a simple filling technique based on the following observations: Starting from a point on a contour there exist only four basic configurations of neighboring contour pixels from which all others can be deduced using symmetries, see Fig. 3. For each configuration the area of pixels that are closer to the pixel than to its two neighbors can easily be determined. Therefore, for each contour pixel only certain image pixels are visited and filled with the distance to the contour pixel.

During the filling a pointer to the border pixel from which the filling started is stored in each image pixel. The filling stops if the border of the image is reached or lower distance values are already present in a pixel, that indicate that further pixels to be filled are closer to another part of the contours of the same slice.

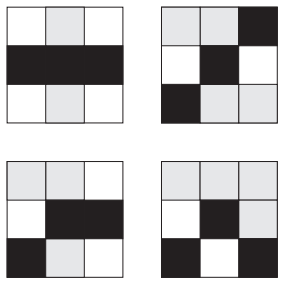

Fig. 3. Four basic configuratuions of contour pixels. Black: Contour pixels, gray: pixels to be filled with distance values, white: pixels, that are filled starting from other contour pixels.

To speed up the filling process and reduce the number of updates of distance values in a pixel we first fill horizontal and vertical lines and only then the more complicated areas. A further speed up is achieved by starting the filling on border pixels that belong to a coarse rectangular grid, thereby considerably reducing the areas to be filled.

4.2 The medial axis

After the computation of the distance field it is easy to extract the medial axes between contours of neighboring slices. First the two distance fields are added. The resulting image contains connected regions i of positive (including zero) or negative values, respectively, see Figure 4. The borders between these areas constitute the medial axes. These borders are extracted with an algorithm which scans the image for a change of the sign and then immediately traces and marks the pixels of encountered medial axes. For each new medial axis a pointer to one of its pixels is stored.

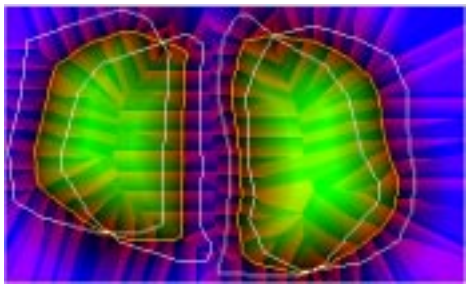

Fig. 4. Adding the two distance fields of consecutive slices delivers the medial axes at pixels where the sign changes. Each slice contains two contours. The gray values in this image encode the distance to the contours as well as the correspondences to the different contour pixels. For a color version of this image see Fig. 10 in the Appendix.

5 Triangulation of the surface

Using the medial axes and the known correspondences to the closest points on the contours the triangulation of the resulting surface is straightforward.

The basic step is to trace the medial axes, which are closed polygons. The tracing starts at an arbitrary pixel of the medial axis. Let M be the medial axis and $M_i, i = 1, \ldots, m$ its pixels. Let P and Q denote the contours corresponding to M and let $P_{i,0}, Q_{k,0}$ be the vertices of the polygon approximating P and Q respectively with a maximum approximation error of $1/2$ pixel. The pixels of the original contours between the vertices are numbered with the second index e.g. the p_i pixels $P_{i,1}, P_{i,2}, \ldots, P_{i,p_i}$ between $P_{i,0}$ and $P_{i+1,0}$, where $P_{i,p_{i+1}} = P_{i+1,0}$, see Fig. 5.

When the pixels of the medial axis are traced in a sequential way, for each of its pixels the numbers i, k of the two corresponding contour edges $P_{i,0}P_{i+1,0}$ and $Q_{k,0}Q_{k+1,0}$ are recorded. If, while tracing the medial axis the corresponding pixel on one of the contours jumps to the next edge, e.g. from edge i to edge $i+1$, the vertex $P_{i+1,0}$ is put

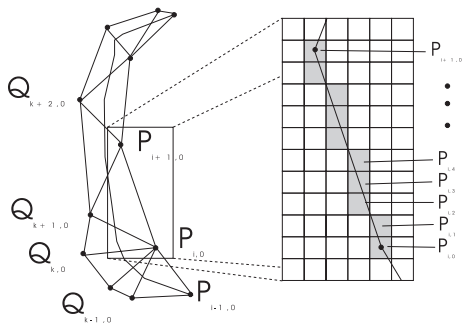

Fig. 5. Triangulation of the contour (see text for details).

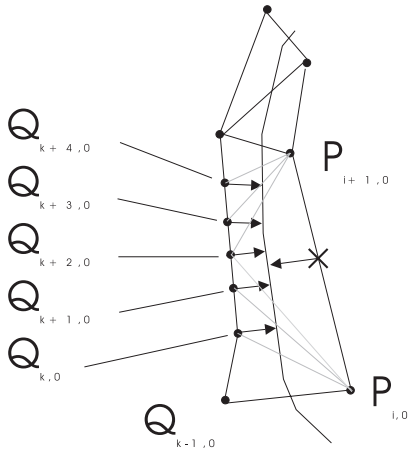

Fig. 6. Vertices '$Q_{k,0}$ through $Q_{k+4,0}$ are connected to $P_{i,0}$ or $P_{i+1,0}$ resp. due to their correspondences on the medial axis.

onto a stack S [1]. At the beginning we trace until in the sequence of recorded vertices on contour P is followed by a vertex on contour Q or reverse. For simplicity we assume for the following that we had a change from P to Q. Then the two vertices are connected by an edge and all other vertices are removed from the stack and the tracing proceeds until again a change in the recorded vertices, now from contour Q to P occurs. Again these two vertices are immediately connected and all remaining vertices that are already on the stack are processed in the following way. The stack contains now a sequence of the following form $PQ \ldots QP$, with one or more vertices from contour Q between two vertices of contour P. The polygon defined by these vertices can be triangulated in different ways. Nice triangulations can be achieved exploiting the correspondences in the following way. We connect all vertices on Q that correspond to pixels on the medial axis that are closer to $P_{i,0}$ than to $P_{i+1,0}$ with $P_{i,0}$ and the others with $P_{i+1,0}$, see Fig. 6.

Until now we have assumed that we always found consecutive edges while tracing the contours, but sometimes this may not happen, see Fig. 7. In these cases new vertices are inserted into the contours. Let us assume that the correspondence changes as shown in Fig. 7. Then we insert the vertices $P_{i,\alpha}, P_{m,\beta}, Q_{j\delta}$ into the contours P and Q, respectively and we introduce one of the two pixels on the medial axis M_i or M_{i+1} (we choose M_i). The corresponding stripe is triangulated as shown in Fig. 7 and the vertices $P_{m,\beta}$ and $Q_{j,\delta}$ are put on the stack [2]. Now the algorithm can proceed as described above until all medial axes have been processed.

After processing all medial axes there remain parts of contours that had no correspondence to a medial axis and are therefore not fully integrated in the triangulation.

[1] The handling of the special (simpler) case, where P and Q corresponds to the same pixel on the medial axis is handled is not described here, but is straight forward

[2] In some cases better triangulations can achieved with a more elaborated algorithm that inserts two additional vertices on the medial axis into the triangulation.

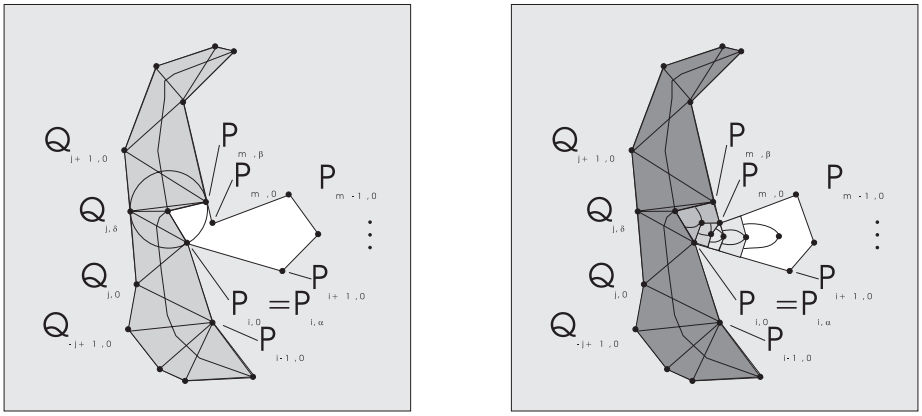

Fig. 7. Left: When edges without correspondence on the medial axis are present, additional vertices are introduced. Then the dark gray area can be triangulated. The white area remains to be handled as shown on the right side: Various shades of gray show areas gained by multiple recursive computations of medial axes between the vertices inserted on the medial axes and the remaining part of the contours. In the Figure six recursion step are performed. The last inserted vertex on the midline lies on a level of only $\frac{1}{32}$ of the distance between the slices above the lower slice. Therefore, the recursion could as well be terminated earlier without giving up much of the quality.

These areas could be triangulated resulting in flat areas parallel to the slices (with the exception of the vertices on the medial axis in between the two slices) [1]. But to improve the results the newly introduced vertices on the medial axes can be considered as new contours (in between the original contours) and the described algorithm (including the calculation of new distance fields in the respective areas) is applied recursively, see Fig. 7. Of course, contours without a neighboring contour in the next slice (end of the object, end of the contour staple) are closed by a plane triangulation in the slice of this contour. During simplification these triangles have to be treated separately in such a way that the distance between the original position and the simplified triangulation can be controlled.

Note, that the results achieved with our simple and fast triangulation algorithm are similar to the ones achieved with the algorithm of Oliva et al [23].

6 Review of multiresolution models

6.1 Generating the multiresolution model

The generation of a MRM of an object generally involves a sequence of local simplification operations like vertex removal, edge collapse, triangle collapse or vertex clustering. The sequence of local simplification operations defines a sequence of coarser and coarser approximations of the original model, the MRM. How this sequence is generated depends on the various simplification algorithms. In general a mesh simplification algorithm starts with the finest triangulation in 3D space approximating the

original model. Then it simplifies the starting triangulation by clustering vertices, by collapsing edges or triangles or by removing vertices from the current triangulation and retriangulating the resulting holes. This is done until no further simplification step can be performed. In many algorithms the order in which the simplification steps are performed is determined by a priority queue. A cost function is evaluated for each possible simplification operation and the one with the lowest cost is performed. In general the cost function represents the error (geometric distance) between original and simplified mesh.

6.2 Selective refinement of multiresolution models

If the inverse local simplification operations are known (e.g. vertex split as the inverse of edge collapse operation), we are able to refine a coarse approximation of the model by reversing the whole simplification process. However, if we want to perform only selective refinement we have to find a way to skip parts of the inverse simplification process and thereby change the sequence of refinement operations. Of course this is not arbitrarily possible (e.g. we cannot split a vertex which is not present in the current mesh). The dependencies between the different simplification steps define a hierarchy that can be described by a directed acyclic graph of modification operations or the associated triangles. Therefore, a general selective refinement algorithm starts with a crude approximation of the model and checks for each triangle if refinement is needed. If yes, the algorithm has to take care that all predecessor operations of the needed refinement operation have already been performed. The next section describes the measure that can be used to decide about the need of further refinement of a certain triangle.

7 Simplification

The most expensive part of the simplification algorithm is the evaluation of the cost function. In our case we need a geometric distance between the original contours and the simplified model. According to our experience the quality of the resulting triangulation does mainly depend on the order of the different simplification steps and not on the special topological operations like vertex removal, edge- or triangle collapse [15]. Therefore, we use a simple edge collapse technique, where no new vertices are introduced.

7.1 Measuring the error

To evaluate the cost function, that is to measure the error between original and simplified surface model we use the fact that in each slice the distance field is already calculated and delivers automatically a set of envelopes of the original contour. Based on this observation the measurement of an error that would be introduced if an edge was collapsed can easily be performed in the following way: Let $\Delta = \Delta(p_j, p_k, p_l)$ be a triangle generated if the edge was collapsed spanning the slices $m, m+1, \ldots, n-1, n$. Consider the line segments $l_i,\ m \leq i \leq n$ defined as the intersection between Δ and the planes $z = z_i$. For each slice we want to find the maximum distance between the

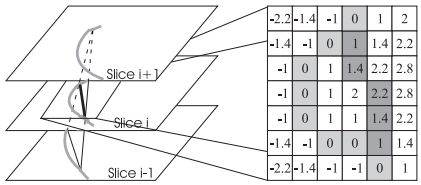

Fig. 8. Measuring the approximation error by tracing Bresenham-lines in the distance fields of the slices.

points of l_i and the contour Ω_i. To find this distance we first read the distance field at the pixels intersected by the triangle Δ. These pixels can easily be determined by 'drawing' l_i using a Bresenham algorithm, see Figure 8. [3]

Note, that like in the simplification envelopes algorithm [6] also in our approach a one-sided Hausdorff- distance between original and simplified triangulation is measured. Therefore, if this distance is smaller than a certain ϵ, we guarantee that for every point p of the simplified triangulation we can find a point q on the original triangulation with $d(p,q) \leq \epsilon$, but the inverse relation does not hold.

7.2 Acceleration of distance computation

For errors larger than 2 pixels in image space the read out of the distance values can be accelerated by skipping $\lfloor \epsilon - d - 1 \rfloor$ pixels, where d is the distance readout at the current position and ϵ is the already reached error between original and simplified model, as from pixel to pixel in image space the distance can grow at most by 1. In this way the drawback of the simplification envelopes algorithm [6] of having a fixed envelop and therefore not being able to build up a reasonable multiresolution model is avoided.

[3] To obtain a smaller (but still conservative) estimation for the deviation from the original contour, the maximum of the values read out from the distance field could be multiplied with $\sin(\alpha)$, where α is the angle between the normal of Δ and the z-direction, see Fig. 9. However, in this case it is important to take care of the bounds of the triangles.

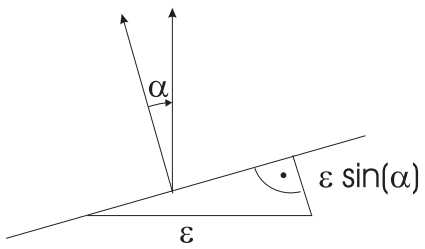

Fig. 9. The error ϵ read out from the distance field is multiplied by $\sin(\alpha)$ to get the real approximation error.

8 Conclusion and further work

The contribution to the problem of reconstruction from contours presented in this paper is a twofold. On one hand the distance field is used for a robust reconstruction algorithm based on the medial axes. In this algorithm the distance field is not only used to calculate the medial axes, but also delivers correspondences used for an excellent triangulation. On the other hand, the second big problem of current reconstruction algorithms, the huge number of resulting triangles, is solved with a new fast simplification algorithm that exploits the (already calculated) distance field to guarantee a certain approximation error between the simplified surface models and the original contours.

This guarantees that in each level of detail, the contours are approximated with a certain approximation error, since the intersections between the slices and the simplified surfaces are within a certain envelop. A problem not yet recognized in the literature is that in general it is not sufficient to guarantee only the distance of the simplified models to certain points or vertices, since it may happen that between two consecutive simplification steps although the distance to the points or vertices remains small or even constant the distance between the two consecutive triangulations can be arbitrarily large. This leads to artifacts in animations where the level of detail is changed. Currently we are working on this problem. Our present research includes also the generalization by using 3D-distance fields to control the approximation error.

9 Acknowledgement

We would like to thank B. Kreher for all the programming efforts which were necessary to gain the described results.

References

1. Gill Barequet and Micha Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding: CVIU*, 63(2):251–272, March 1996.
2. Jules Bloomenthal. Polygonizatino of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, November 1988.
3. Jean-Daniel Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, October 1988.
4. H. N. Christiansen and T. W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *Computer Graphics*, 12(3):187–192, August 1978.
5. P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22, 1998.
6. J. Cohen, A. Varshney, D. Manocha, and G. Turk. Simplification envelopes. *Computer Graphics*, 30(Annual Conference Series):119–128, 1996.
7. Daniel Cohen-Or, David Levin, and Amira Solomovici. Contour blending using warp-guided distance field interpolation. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.
8. H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface re-construction from planar contours. *Comm. of the ACM*, 20:693–702, 1977.

9. Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
10. Gabor T. Herman, Jingsheng Zheng, and Carolyn A. Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, 12(3):69–79, May 1992.
11. John Hershberger and Jack Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In P. Bresnahan et al., editors, *Proc. 5th Intl. Symp. on Spatial Data Handling*, volume 1, pages 134–143, Charleston, SC, August 1992.
12. Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.
13. M. W. Jones and Min Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):C/75–C/84, ???? 1994.
14. E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. Dev.*, 19:2–11, 1975.
15. R. Klein and J. Krämer. Multiresolution representations for surface meshes. In *Proceedings of the SCCG (Spring Conference on Computer Graphics), Budmerice, Slovakia*, pages 57–66, 1997.
16. Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh reduction with error control. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.
17. D. Levin. Multidimensional reconstruction by set-valued approximation. *IMA J.Numerical Analysis*, (6):173–184, 1986.
18. W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In M. C. Stone, editor, *SIGGRAPH '87 Conference Proceedings (Anaheim, CA, July 27–31, 1987)*, pages 163–170. Computer Graphics, Volume 21, Number 4, July 1987.
19. Michael Lounsbery, Charles Loop, Stephen Mann, David Meyers, James Painter, Tony DeRose, and Kenneth Sloan. Testbed for the comparison of parametric surface methods. In L. A. Ferrari and R. J. P. de Figueiredo, editors, *Curves and Surfaces in Computer Vision and Graphics (Proceedings of SPIE)*, volume 1251, pages 94–105, 1990.
20. David Meyers, Shelley Skinner, and Kenneth Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, July 1992.
21. Heinrich Müller and A. J. Klingert. Surface interpolation from cross sections. In H. Hagen, H. Mueller, and G. Nielsen, editors, *Focus on Scientific Visualization*, pages 139–190. Springer-Verlag, 1993.
22. Heinrich Müller and Michael Stark. Adaptive generation of surfaces in volume data. *The Visual Computer*, 9:182–199, 1993.
23. J. M. Oliva, M. Perrin, and S. Coquillart. 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram. *Computer Graphics Forum*, 15(3):C397–C408, September 1996.
24. Bradley A. Payne and Arthur W. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, January 1992.
25. R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C67–C76, C462, September 1996.
26. William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, July 1992.
27. L.L. Schumaker. Reconstructing 3d objects from cross-sections. In W. Dahmen, M. Gasca, and C.A. Micchelli, editors, *Computation of Curves and Surfaces*, pages 275–309. Kluwer Academic, Dordrecht/Norwell, MA, 1989.
28. R. Shu, C. Zhou, and M. S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11(4):202–217, 1995. ISSN 0178-2789.