Virtual Clay Modeling Using Adaptive Distance Fields

Peer-Timo Bremer*, Serban D. Porumbescu*, Falko Kuester†, Bernd Hamann*
Kenneth I. Joy* Kwan-Liu Ma*

*Visualization and Graphics Research Group
Center for Image Processing and Integrated Computing
Department of Computer Science, University of California
One Shields Avenue Davis, CA 95616-8562

†Visualization and Interactive Systems Group
Department of Electrical and Computer Engineering
University of California, Irvine, CA 92697-2625

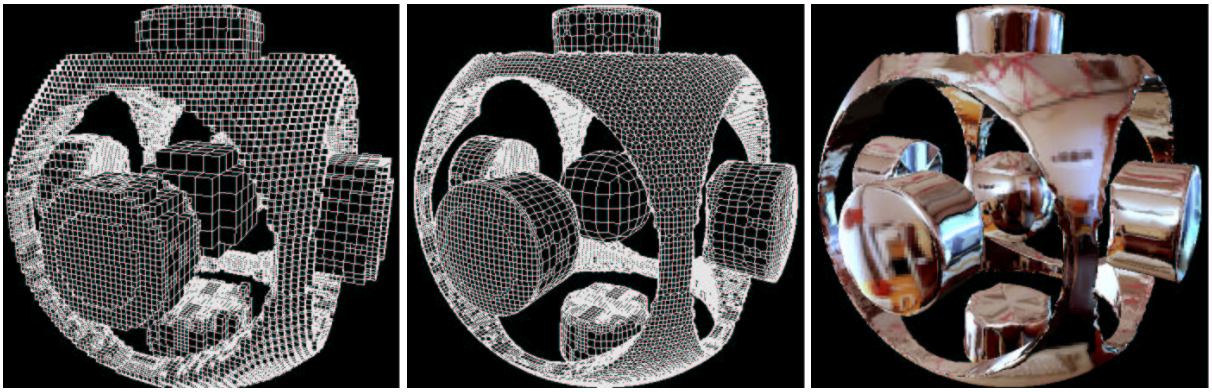

Fig. 1. *Topology changes of sphere created by sweeping tori through it: octree cells (left), resulting mesh (middle) and high-quality rendering using environment mapping.*

***Abstract*—*This paper describes an approach for the parametrization and modeling of objects represented by adaptive distance fields (ADFs). ADFs support the construction of powerful solid modeling tools. They can represent surfaces of arbitrary and even changing topology, while providing a more intuitive user interface than control-point based structures such as B-splines. Using the octree structure, an adaptively refined quadrilateral mesh is constructed that is topologically equivalent to the surface. The mesh is then projected onto the surface using multiple projection and smoothing steps. The resulting mesh serves as the "interface" for interactive modeling operations and high-quality rendering.*

***Keywords*—Virtual Reality, Modeling, Adaptive Distance Field**

I. Introduction

In recent years, the industrial design process has moved toward computer-aided virtual reality based design. The use of computer-aided geometric design (CAGD) techniques dramatically decreases the development time for product prototypes. However, due to several factors, the design process is still not fully virtual. Creating intricate (spline) models from scratch is still complicated. Therefore, designers often first create a model in clay or a similar material that is scanned and converted to analytical form. ADFs, as described by Perry and Frisken [1] promise to advance this process. For example, using the ADF approach a very powerful virtual toolkit can be constructed to model clay quite intuitively and accurately in a fully virtual environment. Compared to other volumetric modeling techniques, such as a voxel-based [2] or constructive solid geometry (CSG) [3] representation, ADFs can provide improved image quality. However, this ADF-based volume modeling is only useful if modeling can be performed interactively and if the result can be converted to traditional CAD model representations, such as B-splines.

We describe a fast and simple approach that creates a quadrilateral mesh of a surface in ADF representation. This mesh can be used for rendering and subsequent approximation of the surface by splines. After discussing related work in Section II, advanced modeling operations will be presented in Section III, along with methods guaranteeing user-defined error bounds. In Section IV, a fast method for model rendering is described. The method provides a quadrilateral mesh on the surface of the model that is a starting point for conversion to B-spline representation.

II. Related Work

Several papers were published on volume-based modeling. Galyean and Hughes [2] described a voxel-based approach to volume sculpting that uses marching cubes [4], [5] to display the model. Later work includes [6], [7], [8]. The approaches described in these papers are limited by low resolution due to the data size of the volume and exhibit high triangle counts for the displayed surface. Adaptively sampled distance fields can solve some of these difficulties.

ADFs were described by Frisken et al. [9], based on earlier work by Gibson [10]. In an ADF approach, discretize distance functions ("fields") are used to represent surfaces. A distance field is a discrete volumetric data set, where each sample point has an associated field value, the minimal signed distance to the surface. This representation offers the same flexibility as a voxel-based representation, but

requires, in the refined adaptive version, much less storage space. Since distance values inside voxels are computed using trilinear approximation, smooth features can be represented using a lower resolution of the modeling space than in a voxel-based method. Frisken et al. store the distance field values in an adaptively refined octree to reduce data size and increase local accuracy of the distance field. With these refinements, ADFs can provide the framework for a powerful modeling engine. In [1], Perry and Frisken describe a method for creating a surface mesh; this will be discussed and compared to our approach in Section IV.

In principle, geometric modeling is done using a solid modeling or surface modeling approach. Solid modeling is mostly based on a direct voxel representation [2]. The modeling space is divided into voxels that are segmented into two categories: Voxels lie inside or outside the model. The advantage of this method is its flexibility. Arbitrary topology can be modeled, and topological changes require no additional work. However, since the manageable data size, and therefore resolution, is limited, image quality remains poor and precision limited.

On the other hand, surface-based modeling techniques are used extensively in industry. Extensive research describing virtual modeling using surface deformations has been conducted [11], [12], [13]. However, most of these approaches share two fundamental problems: (1) Surface models are control-point based, and the modeling process is in its nature indirect. Even though much work on direct free-form deformations (FDDs) [14] has been done, these methods still modify control points. (2) The initial topology of control-point-based structures remains a problem. For all practical purposes, this structure cannot be changed easily, and therefore the topology of the model itself cannot change.

Our work combines solid modeling with surface modeling approaches. ADFs are used to model objects without any topological constrains. A fast and simple method to generate a surface mesh for any given object is provided. This approach allows us to achieve interactive rendering rates and high image quality. A benefit of a mesh-based representation is that the resulting mesh provides a good starting point for converting a quadrilateral mesh to a (tensor-product) spline-based representation.

III. Modeling

A primary advantage of ADFs is their flexibility. This is especially true for the implementation of Boolean operations. For example, the difference between the distance field of an object d_O and the distance field of a tool d_T can be computed by $min\{d_O, -d_T\}$, see Figure 2. All other Boolean operations (union, intersection) and ADF tools can be computed in the same fashion [9]. In general, modeling is done as follows: Depending on the tool size, a small local neighborhood of the octree around the tool is updated. The specific update rules are defined by the operation, e. g., carving, adding, etc. The following criterion is used: *An octree cell*

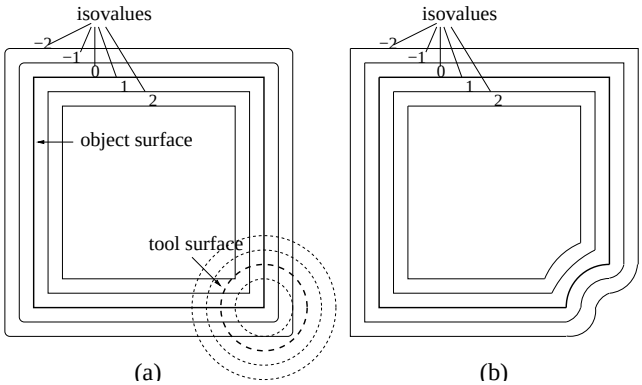

Fig. 2. *Isolines of a distance field before (a) and after carving operation (b).*

is split if the local trilinear approximation of the distance field in the cell differs by more than a user-defined error bound from the distance field given by the tool. In the same manner, eight children are merged when the parent node approximates the distance field within the error bound. During modeling operations, we are interested in the isosurface of a distance field f(x,y,z). The isosurface f=0 represents the surface to be modeled. Therefore, the error introduced by only updating the octree locally is not significant in our application. Boolean operations also introduce an inherent error, as described by Breen et al. [15], for points not close to the surface. However, techniques exist that can correct this error if necessary, see [16], [15]. For surface modeling and rendering only values "close to" the surface need to be correct. Since greater errors are introduced where one moves away from the surface, it is reasonable to apply an adaptive error bound: If the user-defined error bound is ϵ, we want this bound to be effective for cells whose centers are less than $k\epsilon$ away from the surface (where k is a user-defined constant). From there on, we want the error bound to decline quadratically with increasing distance to the surface, shown in Figure 3. According to the desired properties, we

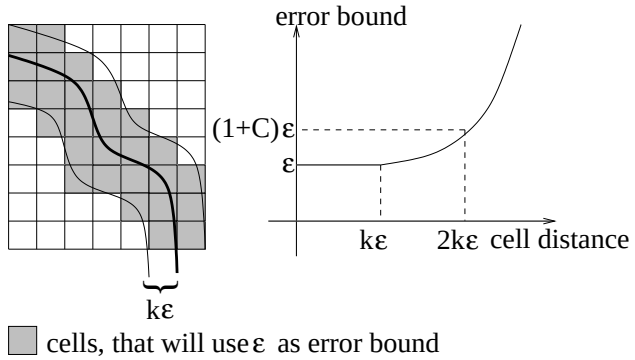

Fig. 3. *Adaptive error bound.*

define the adaptive error bound $\hat{\epsilon}$ as: $\hat{\epsilon} = \epsilon$, if $d_c < k\epsilon$, or $\hat{\epsilon} = \epsilon + (d_c - k\epsilon)^2 * \frac{C}{k^2\epsilon}$, otherwise. The parameter d_c denotes the distance of a cell center to the surface, and C is another user-defined constant. For an even less precise but faster-to-calculate error bound, one can ignore all cells with distance values d_c greater than some constant, see [1]. An

adaptive error bound has two advantages: (1) It reduces the number of octree cells.(Depending on k, C and the size of the octree, we have observed savings by more than an order of magnitude.) (2) It solves the problem of critical points in the distance field.

Our method of enforcing the error bound depends on the local derivatives of the distance field. Distance fields can contain critical points where the gradient is ill-defined. An example is shown in Figure 4. These critical points define the

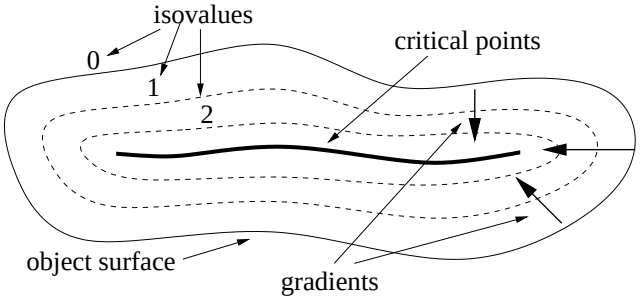

Fig. 4. *Part of medial axis of an object.*

medial axis, see [17], [18] for details. Such critical points are generally far away from the surface and are not of interest. However, due to ill-defined gradients, cells around the critical points are usually subdivided further than cells close to the surface. An adaptive error bound solves this problem.

IV. Mesh Creation

To render an ADF at interactive rates, polygon rendering techniques must be applied. Perry and Frisken [1] use the centers of what they call "boundary leaf cells" (called "data-leaf cells" by us) as vertices for a triangulation. Based on the local neighborhood of these leaf cells, certain edges are chosen and triangles are constructed around them. The vertices are then projected onto the surface and the triangles are relaxed to improve triangle quality. The disadvantage of this approach is that it can create cracks. However, cracks can be detected and eliminated by locally refining the octree.

We use a different approach. Since a primary goal is the eventual conversion to a spline representation, we prefer dealing with quadrilateral elements. If a "nice" quadrilateral mesh can be constructed, existing methods can be used to convert the meshes to splines, e.g., hierarchical B-splines [19]. A method that automatically displays the correct topology of the distance field is also preferable.

Our method works as follows: Throughout the modeling process, each octree node updates a data flag. This flag is true when the node or one of its children contains or touches parts of the surface. An octree node "touches" a surface when the distance values at each of its corners have the same sign but one or more distance values are zero (or close to zero within a user-defined error bound.) A node "contains" parts of a surface when at least two of the distance values at each corner have a different sign, see Figure 5. All leaves

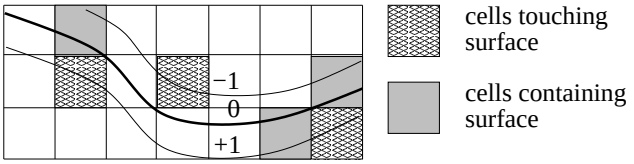

Fig. 5. *Cells touching/containing surface.*

that contain or touch the surface are called "data leaves." Figure 6 shows the data leaves of a bowl.

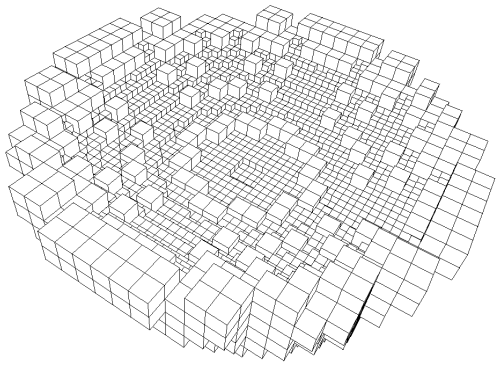

Fig. 6. *Data leaves of a bowl.*

The union of all data leaves defines a set that is topologically equivalent to the surface, (within a user-defined error bound.) Considering an orientable two-manifold surface, each data leaf has at least one face neighbor that is also a data leaf. A leaf is a data-leaf when it contains the surface or touches it. If the leaf contains the surface, the data at corners of at least one face will have a different sign. However, since there is an octree node that shares this face, it would also be chosen as a data leaf. Therefore, leaf A in Figure 7 would have a data leaf as face neighbor. If a data leaf touches the surface, there is at least one corner that has a distance value of zero. Since this corner is shared by eight octree nodes, all eight nodes are data leaves. Therefore, the first leaf has at least three face neighbors that are also data leaves, see Figure 7 (b).

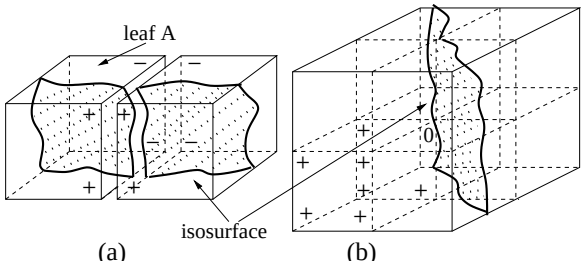

Fig. 7. *Face neighbors for node containing surface (a) and node touching surface (b). Vertices marked "+" are outside the object, the ones marked "-" are inside.*

We now consider the set **F** of all faces of all data leaves: Data leaves are divided into three categories: *(1) outside*

faces; (2) inside faces; and *(3) partner faces*. A face $f_1 \in \mathbf{F}$ is called partner face when a face $f_2 \in \mathbf{F}$ exists such that f_1 and f_2 contain the same vertices in opposite orientation. All other faces lie either on the inside or the outside of the surface and are named accordingly. The subset of all outside faces define a quadrilateral mesh that is topologically equivalent to the surface. By connecting all faces, projecting the vertices onto the surface and relaxing the mesh, we can create a mesh whose vertices lie exactly on the surface. The special case where faces of different levels share an edge can be detected easily. In the octree, neighboring cells are allowed to be one level apart. Therefore, the only special case that can occur is depicted in Figure 8. For the

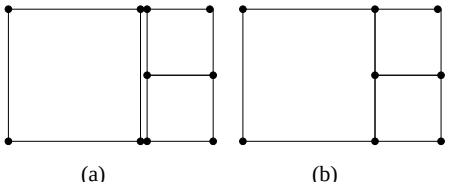

Fig. 8. *Avoiding cracks between levels: before connection (a) and after connection (b).*

projection of vertices we first calculate an approximate surface normal. (One can either use the volumetric information contained in the octree of the surface normals of the current mesh.) The ADF provides a distance-to-closest-surface value for each vertex amd we move the vertex by this distance along its normal. The resulting mesh is smoothed (relaxed) to distribute the vertices more evenly, using for example a Laplacian smoothing. If necessary, the procedure is repeated. Ususally at most three project/smooth steps are necessary to move all mesh vertices onto the surface. Our approach avoids cracks and maintains a quadrilateral structure. To selecte all outside faces one must considere for each data leaf which of its six faces must be created. This decision is based on the distance values at the corners and the leaf's resolution level relative to that of its neighbors.

Severall special cases can occur for which the connection of faces is not straightforward. In Figure 9, four faces share an edge, and the connectivity is not necessarily obvious. Many more complicated special cases exist, but will not be described in this paper. For display purposes, it does not matter how these edges are connected.

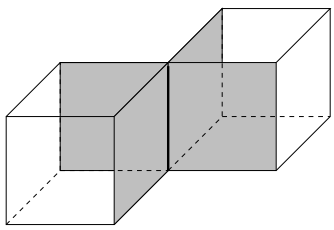

Fig. 9. *Four faces sharing edge (bold).*

V. Results

Figure 1 shows three views of the same model. The model was created by sweeping a torus successively along the three coordinate axes through a sphere. This results in multiple topology changes. Not only can our algorithm create the final mesh but it can also create topologically correct meshes at any time during the carving process. Figure 10 shows the UC Davis mascot as a 3D model. One notices that the model is represented as a closed $2\frac{1}{2}$D surface. Figure 12 shows a sphere with four rectangles carved out and two holes drilled into it. In the close-up view of the mesh (Figure 13) one can see the highly adapted mesh that captures the sharp edges. In Figure 11 we show a cube rotated around one diagonal. Both edges and corners are preserved.

The smaller models, like the cube, consist of about 10000 faces with a level-7 octree, while the mascot has about 155000 faces and a level 11 octree. The smaller meshes required less than one second to generate, only the Davis mascot mesh required three to four seconds. However, we are using our prototype implementation which is not optimized.

Fig. 10. *Three-dimensional model of the UC Davis mascot.*

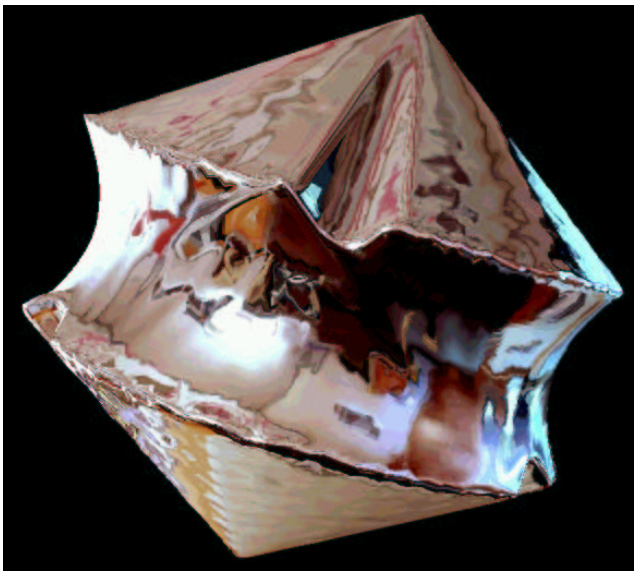

Fig. 11. *Cube partially "rotated" around its diagonal.*

VI. Conclusions

We have described an algorithm that converts objects in ADF representation to a two-dimensional surface representation consisting of quadrilaterals. Within a user-defined error bound, this quadrilateral mesh is topologically equivalent to the object's surface. The creation of the mesh is numerically stable and can be used for efficient rendering. This approach combines the modeling techniques provided by ADFs with fast polygon-based rendering. We are able to model objects of any topology, as long as their surface is an orientable two-manifold. More importantly, changes in topology are easily obtained as "byproducts" of simple operations on an underlying ADF representation.

VII. Future Research

Our method has produced promising results. Some issues need to be studied in more detail. We are currently working on several methods to improve mesh creation and appearance. To further increase rendering speed the mesh should only change in a certain neighborhood for local modeling operations. Additional work also needs be done concerning the user interface, especially for virtual environments.

Several research directions are possible. As of now, our mesh still contains many extraordinary vertices (vertices with valences unequal four). We are investigating algorithms to topologically "clean up" the mesh to reduce the number of such extraordinary vertices substantially, without distorting the mesh too much. Furthermore, user studies are needed to determine what type of modeling tools are suitable for which types of modeling applications. Especially, it should be clearly defined what kind of modeling will be performed best using standard CAGD techniques and what type could benefit most from ADF techniques.

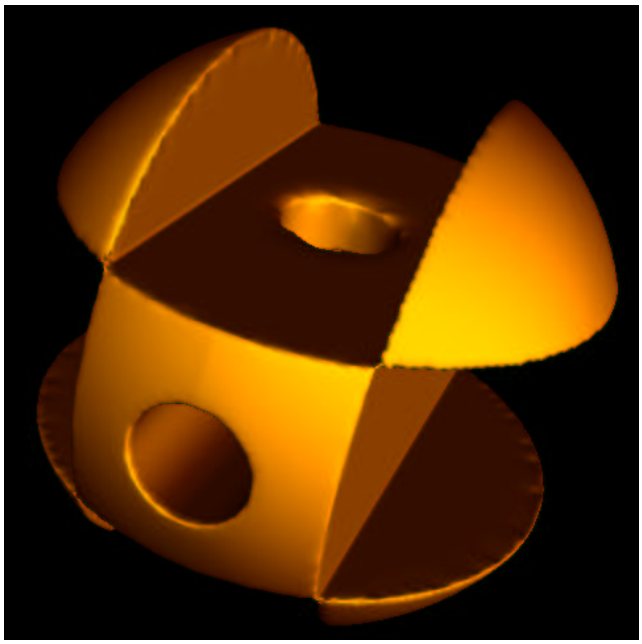

Fig. 12. *Spherical solid manipulated by multiple carving operations.*

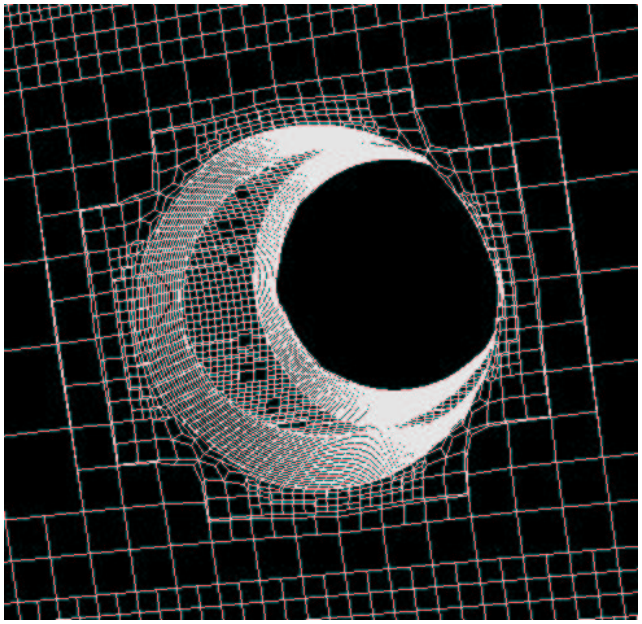

Fig. 13. *Magnified view of a hole of geometry from Figure 12.*

Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; the Lawrence Berkeley National Laboratory; the Los Alamos National Laboratory; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628. We also acknowledge the support of ALSTOM Schilling Robotics and SGI. We thank the members of the Visualization and Graphics Research Group of the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

References

[1] R. N. Perry and S. F. Frisken, "Kizamu: A system for sculpting digital characters," *Computer Graphics (Proc. SIGGRAPH' 01)*, vol. 35, no. 4, pp. 47–56, 2001.

[2] T. A. Galyean and J. F. Hughes, "Sculpting: An interactive volumetric modeling technique," *ACM Trans. Computer Graphics*, vol. 25, no. 4, pp. 267–274, 1991.

[3] H. Chen and S. Fang, "A volumetric approach to CSG modeling," in *Symposium on Solid Modeling and Applications*, New York, New York, Jun. 1999, ACM, pp. 318–319, ACM Press.

[4] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *Computer Graphics (Proc. SIGGRAGPH '87)*, vol. 21, no. 4, pp. 163–169, July 1987.

[5] G. M. Nielson and B. Hamann, "The asymptotic decider: Resolving the ambiguity in marching cubes," in *Visualiztion '91*, G. M. Nielson and L. J. Rosenblum, Eds., Los Alamitos, California, Oct. 1991, IEEE, pp. 83–91, IEEE Computer Society Press.

[6] S. Wang and A. E. Kaufman, "Volume sculpting," in *Symposium on Interactive 3D Graphics*, Monterey, CA, Apr. 1995, ACM, pp. 151–156.

[7] A. Raviv and G. Elber, "Three dimensional freeform sculpting via zero sets os scalar trivariate functions," in *Symposium on Solid Modeling and Applications*, New York, New York, 1999, ACM, pp. 246–257, ACM Press.

[8] J. Baerentz, "Octree-based volume sculpting," in *Proc. IEEE Visualization '98*, D. Ebert, H. Hagen, and H. Rushmeier, Eds., Los Alamitos California, 1998, IEEE, pp. 9–12, IEEE Computer Society Press.

[9] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," *Computer Graphics (Proc. SIGGRAPH '00)*, vol. 34, no. 4, pp. 249–254, July 2000.

[10] S. F. Gibson, "Using distance maps for smooth surface representation in sampled volumes," in *IEEE Symposium on Volume Visualization*, Los Alamitos California, Oct. 1998, IEEE, pp. 23–30, IEEE Computer Society Press.

[11] Y.-H. Chai, G. R. Luecke, and J. C. Edwards, "Virtual clay modeling using the ISU exoskeleton," in *Proc. IEEE Virtual Reality Annual Symposium (VRIAS'98)*, Los Alamitos California, March 1998, IEEE, pp. 76–80, IEEE Computer Society Press.

[12] J. P. Y. Wong, R. W. H. Lau, and L. Ma, "Virtual 3d sculpting," *Visualization and Computer Animation*, vol. 11, pp. 155–166, 2000.

[13] K. Kameyama, "Virtual clay modeling system," in *Symposium on Virtual Reality Software and Technologyy, VRST '97*, New York, New York, Sep. 1997, ACM, pp. 197–200, ACM Press.

[14] W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformations," *ACM Trans. Computer Graphics*, vol. 26, pp. 177–184, 1992.

[15] D.Breen, S. Mauch, and R. Whitaker, "3D scan conversion of CSG models into distance volumes," in *IEEE Symposium on Volume Visualization*, Los Alamitos California, Oct. 1998, IEEE, pp. 7–14, IEEE Computer Society Press.

[16] O. Cuisenaire, *Distance Transformations: Fast Algorithm and Applications to Medical Image Processing*, Ph.D. thesis, Universite Catholique de Louvain, Dept. of Engineering, Louvain-la-Neuve, France, 1999.

[17] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, New York, 1976.

[18] F.-E. Wolter, *Cut Loci on Bordered and Unbordered Riemannian Manifolds*, Ph.D. thesis, Technical University Berlin, Dept. of Mathematics, Berlin, Germany, 1985.

[19] D. Forsey and R. Bartels, "Hierarchical b-spline refinement," *Computer Graphics (Proc. SIGGRAPH '88)*, vol. 22, no. 4, pp. 205–212, 1988.