# Automatic Semi-regular Mesh Construction from Adaptive Distance Fields

P.-T. Bremer S. D. Porumbescu B. Hamann K. I. Joy

**Abstract.** This paper describes a method to construct semi-regular meshes for a surface S defined by the zero set of a trivariate function F(x,y,z), representing a distance field definition of S. An adaptive distance field (ADF) definition of S is obtained by constructing, adaptively, an octree decomposition of F's domain. The vertices of the octree-based definition of S lie either on the positive or negative side of S (or on S). Octree cells that are intersected by S are identified, and the faces of these cells that lie on the outside of S are projected onto S. The result is a quadrilateral mesh to which various procedures are applied that lead to an improved mesh containing a much smaller number of extraordinary vertices, i.e., non-valence-four vertices.

## §1. Introduction

Adaptive distance fields (ADFs) are gaining increasing importance as they support adaptive modeling of complicated surfaces [8]. Due to rapid technology advances, their large memory requirements are no longer viewed as being restrictive. ADFs have many advantages: Large point data sets, such as high resolution laser scans, can be converted to ADFs without the need to perform a triangulation step by using, for example, radial basis functions [2]. ADFs can also be used to represent unfavorably triangulated models or discontinuous surfaces.

Modeling an ADF means modeling an implicitly defined surface, the surface defined as the zero set of a trivariate function F(x,y,z). However, an implicit surface definition is not the preferred representation for many application areas. In most industrial settings, a surface-based, bivariate representation is used. We describe an approach that integrates ADF-modeling and two-manifold surface representation.

Curve and Surface Fitting: Saint-Malo 2002
XXX (eds.), pp. 1–10.

ISBN XXX.

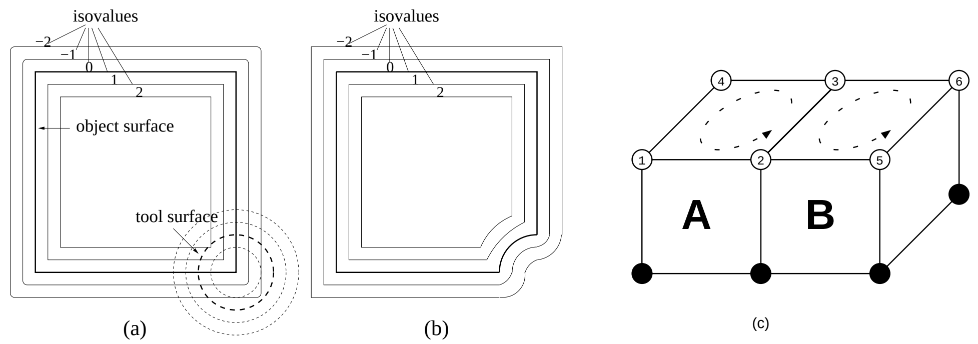

**Fig. 1.** (a) Isolines of a distance field before and (b) after a carving operation; (c) two data leaves showing vertex indices and face orientation.

Recent work concerning surface parameterization is described in [3,7,1]. There has been an increasing number of papers about extracting meshes from volumetric representations [6,5]. The meshes extracted by Ju et al. resemble the meshes we presented in [9], but the underlying methods are quite different, see Section 3. To store and manipulate ADFs we use the methods described in [8].

In an ADF modeling setting, discretized (sampled) distance functions ("fields") are used to represent (implicit) surfaces. Since the surface is implicitly defined, the ADF represents one or multiple closed components of the surface. A topologically and geometrically simplified surface is defined by a coarser representation of the ADF.

A primary advantage of ADFs is flexibility. This flexibility is especially beneficial for the implementation of Boolean modeling 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 using the operator $min(d_O, -d_T)$, see Figure 1. All other Boolean operations (union and intersection) and ADF tools can be implemented in the same fashion [8].

## §2. Mesh Creation

First, we convert an ADF to a surface representation via creation of a polygonal surface mesh. As done in [6], we force all octree cells of the zero set to the same level. (We refer the reader to [9] for a description of mesh creation in an adaptive setting.)

We identify octree leaves ("data leaves") that contain parts of the implicit surface $F(x, y, z) = 0$ (the zero set of the ADF ). The union of the data leaves defines a set that is topologically equivalent to the surface, within some error bound. Topological equivalence refers to preservation of every topological feature, e.g., a hole or a connection between components. The set of faces of the data leaves that lie completely "outside" the surface defines a quadrilateral mesh that is topologically equivalent to the implicit surface. The vertex valences (number of edges emanating from vertices) in this mesh vary between three and six. By connecting all these faces, projecting the vertices onto the implicit surface, and "relaxing" the mesh, as described in [8], we create a mesh whose vertices lie on the surface.

The surface $S$ is defined by three simplicies: vertices $v = \{i\}$, edges $e = \{i, j\}$, and faces $f = \{i, j, k, l\}$ where $i$, $j$, $k$, and $l$ are indicies into a set of $N$ geometric point positions such that $1 \leq i, j, k, l \leq N$. Furthermore, we define a key for an edge $e$ (defined by indicies $i$ and $j$) as the index pair $k = (i, j)$.

Figure 1 (c) shows two data leaves $A$ and $B$. The numbered vertices lie on the exterior of the zero surface and the dark vertices lie on the interior. Based on a consistent orientation (counterclockwise) imposed by the octree, voxel $A$ contributes face $f_A = \{1, 2, 3, 4\}$ and voxel $B$ face $f_B = \{3, 2, 5, 6\}$ to the surface. Edge $\{2, 3\}$ from $f_A$ has as its key $(2, 3)$. Likewise, edge $\{3, 2\}$ from $f_B$ has key $(3, 2)$. To connect edge $\{2, 3\}$ we search for key $(3, 2)$ and connect the associated faces in the surface $S$. In general, given a data leaf we create all its outside faces and the keys associated with those faces' edges. The construction of a "reversed" key signifies that a neighboring face has been found.

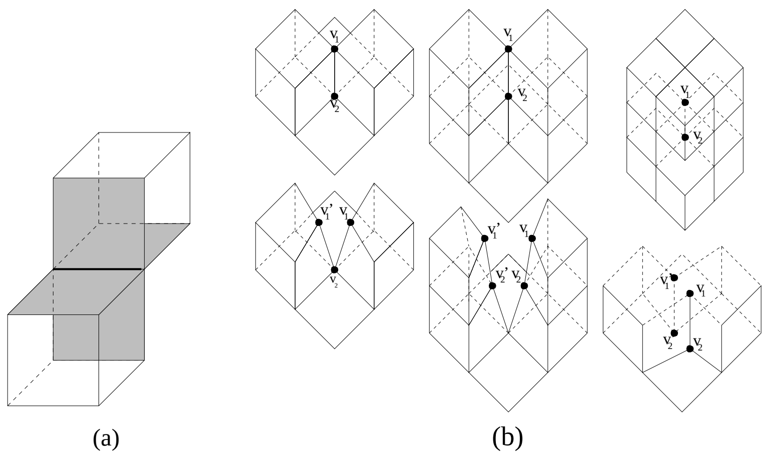

**Fig. 2.** (a) Four faces sharing edge; (b) vertex duplication.

Figure 2(a) shows a special case that arises when dealing with more complex models. In this example, four faces share a common edge, and the index pair of an edge is no longer unique. The topologically correct decision is to connect the faces that originate from the same cell. (Since the other two cells sharing this edge are not data leaves, the implicit surface cannot cross this edge.) To avoid the problem of duplicate edge keys we create all outside faces of a data leaf at once. Immediatley after creation we start connecting faces. Once an edge key is used to find a connection, it is discarded. This strategy ensures that keys not used so far are unique, and situations as the one in Figure 2(a) are handled correctly.

We have described how to correctly connect faces of independent components. Since each mesh vertex is only created once, these faces still share the same corner. In certain cases we must "split" mesh vertices into multiple vertices. There exist several configurations for these cases, some of which are shown in Figure 2(b). In the leftmost configuration shown in Figure 2(b) the vertex v1 must be duplicated, but v2 does not need to be duplicated. In the middle configuration, both v1 and v2 must be duplicated. (In general, a vertex should be duplicated when there exist two or more independent sets of faces that share this vertex.) However, the right configuration shown in Figure 2(b) depicts the singular case when this is not true: The familiar situation of two leaves sharing an edge occurs between two planes of leaves. Both v1 and v2 are part of only one set of faces, but the usual connection creates a configuration where the platelet of v1 (i.e., the set of faces that share v1) contains v2 twice, effectively creating a one-dimensional "hole" in the surface. In this case, we duplicate both v1 and v2 and change the connections artificially, thus closing the existing hole. This is the only case where our algorithm creates a topologically incorrect surface mesh with respect to the zero set of the distance field. However, this theoretically possible case is rare and it has never occurred in all our experiments.

## §3. Mesh Simplification

The quadrilateral mesh resulting from the mesh creation process has special properties due to the connectivity of the octree leaves containing the surface. Exploiting these properties, we decimate the mesh by applying three specific mesh decimation operators. These operators are designed to remove irregular vertices (vertices with valence different from four). The algorithm eliminates (collapses) entire closed strips of quadrilaterals, guided by the correspondence between strips and octree. We define a ring and the ring condition in Section 3.1 to facilitate the discussion of collapse operators in Section 3.2. Section 3.3 describes the algorithm in full and provides a specific sequence for application of the decimation operators.

### 3.1 Rings and the Ring Condition

A quadrilateral strip is a set of quadrilaterals that are only connected via opposite edges. A **ring** is a closed quadrilateral strip that does not contain the same quadrilateral twice. Considering the surface defined by a voxel model, there exist three types of rings. Each ring type is defined by octree faces being parallel to one coordinate plane. For example, a ring parallel to the xz-plane is defined by octree faces with the same y coordinate. This means that a ring cannot cross itself (at least not on an orientable two-manifold surface). Two rings of a quadrilateral mesh of a machine part

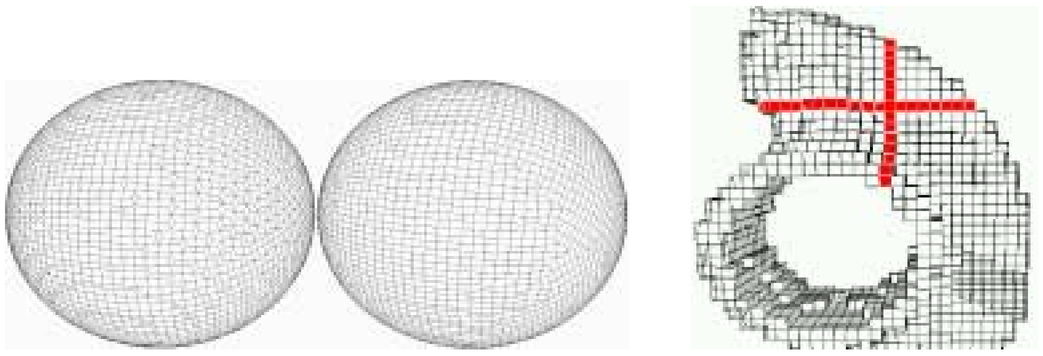

**Fig. 3.** Original mesh of sphere (left) and mesh after local clean-up step (middle); data leaves of machine part with two rings (right).

are shown in Figure 3. A quadrilateral mesh satisfies the **ring condition** when the mesh is comprised entirely of rings.

## 3.2 Collapse Operators

We describe three decimation operators that, when used together, transform an unstructured quadrilateral mesh (extracted from an octree) into a semi-regular mesh. The three operators are: Local Collapse, Semi-local Collapse, and Ring Collapse. They are defined as follows:

**Local Collapse.** *A quadrilateral comprised of vertices with valences* $(3, m, 3, n)$ *where* $4 \leq m, n \leq 6$*, is collapsed to a single point having valence four. An example of several local collapses can be seen in Figure 4.*

**Semi-local Collapse.** *This operator involves collapsing two quadrilaterals comprised of vertices with valences* $(3, m, 3, n)$ *where* $4 \leq m, n \leq 6$ *simultaneously along their valence-three diagonal. The two quadrilaterals to be collapsed must have a sequence of valence four vertices between their valence-three vertices. By collapsing both end faces at the same time, one "unwinds" the two rings involved. The valid configuration and associated collapse can be seen in Figure 5(b).*

**Ring Collapse.** *This operator removes an entire ring of quadrilaterals simultaneously. Figure 5(a) shows an example of a Ring Collapse where four pairs of irregular vertices, shown by black dots, can be removed simultaneously by collapsing the eight faces that surround the center region. Considering the octree structure, the Ring Collapse is equivalent to "pushing" the extruded four leaves inward to create a locally regular mesh.*

## 3.3 Algorithm

Given any voxelized model, one realizes that all details of a surface result from "in-" or "extrusion" of certain octree cells, starting with some kind

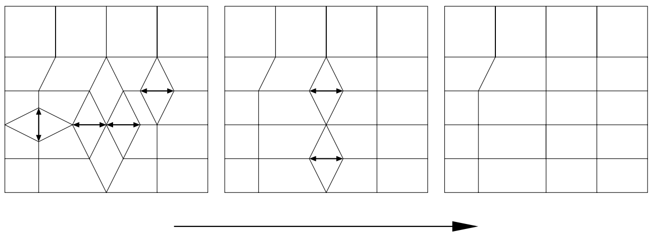

**Fig. 4.** Local face collapse creating regularity.

of box shape. If one identifies all these operations and reverses them in the correct order, one would be able to construct a mesh that is the mesh associated with the original box shape. Our algorithm is based on this insight.

Close inspection of mesh extracted from the octree reveals that the mesh has a specific structure that is defined by the octree from which it was extracted. This specific type of quadrilateral mesh has vertices with valences that range between three and six only. Additionally, only a restricted set of local configurations occurs based on the types of local block structures of the octree. In fact, an extracted quadrilateral mesh clearly reveals the original underlying octree structure, see Figure 3 (left and middle). We exploit this structure in devising our collapse strategy. A typical configuration is shown in Figure 5(a).

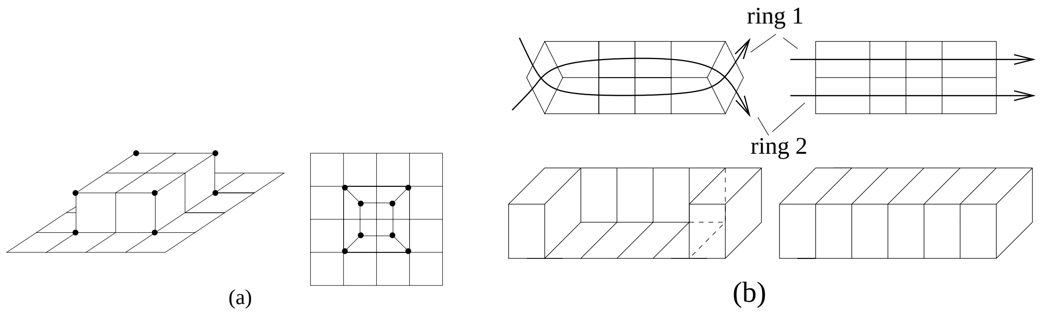

**Fig. 5.** (a) Outside faces of octree leaves (left) and corresponding mesh (right); (b) Semi-local collapse.

To convert the quadrilateral mesh into a semi-regular mesh we need to apply the collapse operators in a specific order. To motivate the sequence in which we apply these operators we first consider the case of applying the Local Collapse operator to the mesh. An example of this approach is depicted in Figure 3 (left and middle). After application of the local collapse operator, the mesh is more regular. However, close inspection reveals that in nearly each row and column of the mesh some irregular vertices remain. Even though large regions of the mesh are regular, there exist no large regions consisting only of valence-four vertices. Since only

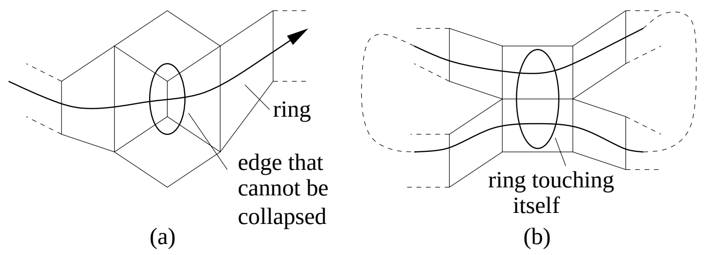

**Fig. 6.** Situations where rings cannot be collapsed: (a) illegal valence and (b) ring that touches itself.

local information is used when applying a Local Collapse, collapses are performed at random. One can impose certain heuristics (e.g., collapsing certain configurations before others), but this is difficult in our experience.

Clearly, the Local Collapse is not sufficient. We propose the application of a sequence of operators that utilizes more information to drive the conversion to a semi-regular mesh. The application of operators is as follows: Semi-local Collapse, Ring Collapse, and Local Collapse. This sequence of operators is applied in a way that preserves the ring condition for as long as possible.

Application of the Semi-local Collapse is straightforward. We identify in the mesh configurations that are amenable to the Semi-local Collapse. The Ring Collapse is a simple and powerful operation. Unfortunately, selecting which ring to collapse first is not a simple task.

We define an order according to which to collapse rings. Since the goal is to remove as many irregular vertices as possible, we use the number of irregular vertices as a deciding criterion. For each ring, we determine how many irregular vertices are removed when this ring is collapsed. Surprisingly, collapsing rings that remove the maximal number of irregular vertices first does, in general, not work well. Each face is part of two rings, and, therefore, each irregular vertex can be removed (or at least be changed) by two different ring collapses. Often, there exist rings with many irregular vertices, and all of them should be removed using some other ring. Since one also wants the mesh to stay geometrically close to the original surface, collapses of smaller rings (rings with smaller numbers of faces) are preferable.

We order rings by the numbers of faces they contain, and smaller rings are processed first. If two rings have the same number of faces, we collapse the one that removes more irregular vertices. We never collapse rings that remove less than four irregular vertices. (Considering the voxel model, any in-/extrusion operation creates at least four pairs of irregular vertices, unless existing irregular vertices distort the configuration.) Some additional special cases must be considered. Some rings, for topological reasons, cannot be collapsed, see Figure 6.

If a Ring Collapse were to merge two valence-three vertices, a valence-two vertex would result, which is not allowed, see Figure 6(a). For implementation reasons, we do not collapse rings that touch themselves, see Figure 6(b). Sometimes it is useful to perform multiple passes of Semi-local and Ring Collapsing. In our experience, two passes produce configurations that our algorithm can no longer improve. Once such a configuration is obtained, one can start applying operators that destroy the ring condition. We apply the Local Collapse operator as shown in Figure 4, to improve the mesh further. However, there are usually only few places where improvements are possible with the operators we consider.

We have only described the topological aspect of our algorithm. Whenever we collapse a ring, we also change the mesh geometrically by creating elongated quadrilaterals. We also loose geometrical accuracy by removing large numbers of mesh elements. However, the original ADF representation can be used to keep the quadrilateral mesh reasonably accurate. The ADF also allows us to smooth the mesh, remove local distortions, or re-project the mesh vertices to the exact surface. Such operations can be done repeatedly, for example, after a certain number of collapses. To reintroduce detail that was removed during the simplification process we apply several steps of bilinear subdivision (commonly referred to as polygonal subdivision). Between subdivision steps we project newly introduced vertices onto the zero set of the original ADF. Some stages of the algorithm are shown on the model of an Isis statue in Figure 7 through 11.

## §4. Conclusions

We have presented a technique to extract a semi-regular mesh from an ADF-based surface definition. This mesh could be used to construct B-spline or NURBS representations, necessary for the use of distance fields in industrial modeling systems. The technique presented relies heavily on the structure of the octree from which the quadrilateral mesh is extracted. As a result, the technique does not apply to general quadrilateral meshes. We plan to further improve the technique by introducing more powerful collapse operators and by investigating a new technique that can avoid collapses entirely.

**Acknowledgments.** This work was supported, in part, by the National Science Foundation, the National Institute of Mental Health, the Lawrence Livermore National Laboratory and the Lawrence Berkeley National Laboratory. We thank Mark Duchaineau and Daniel Laney for the use of thier distance field generation tools and Cyberware for the use of thier models.

## References

1. Alliez, P., M. Meyer, and M. Desbrun, Interactive geometry remeshing, *Computer Graphics (Proc. SIGGRAPH '01)*, 36(4):347–354, 2002.
2. Carr, J. C., R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, Reconstruction and representation of 3d objects with radial basis functions, *Computer Graphics (Proc. SIGGRAPH '01)*, 35(4):67–76, 2001.
3. Gu, X., S. Gortler, and H.Hoppe, Geometry Images, *Computer Graphics (Proc. SIGGRAPH '02)*, 36(4):355–361, 2002.
4. Gibson, S. F., Using distance maps for smooth surface representation in sampled volumes, in *IEEE Symposium on Volume Visualization*, pages 23–30, Los Alamitos, California, Oct. 1998. IEEE, IEEE Computer Society Press.
5. Ju, T., F. Losasso, S. Schaefer, and J. Warren, Dual contouring of hermite data, *Computer Graphics (Proc. SIGGRAPH '02)*, 36(4):339–346, 2002.
6. Kobbelt, L. P., M. Botsch, U. Schwanecke, and H.-P. Seidel, Feature sensitive surface extraction from volume data, *Computer Graphics (Proc. SIGGRAPH' 01)*, 35(4):57–66, 2001.
7. Lee, A. W. F., W. Sweldens, P. Schroeder, L. Cowsar, and D. Dobkin, Maps: Multiresolution adaptive parameterization of surfaces, *Computer Graphics (Proc. SIGGRAPH '98)*, 32(4):95–104, 1998.
8. Perry, R. N., and S. F. Frisken, Kizamu: A system for sculpting digital characters, *Computer Graphics (Proc. SIGGRAPH '01)*, 35(4):47–56, 2001.
9. Bremer, P.-T., S. D. Porumbescu, F. Kuester, B. Hamann, K. I. Joy, K.-L. Ma: Virtual clay modeling using adaptive distance fields, in *Proceedings of the 2002 International Conference on Imaging Science, Systems, and Technology (CISST 2002)*, Volume 2, pages 627–632.

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
`{bremerp, porumbes, hamann, joy}@cs.ucdavis.edu`

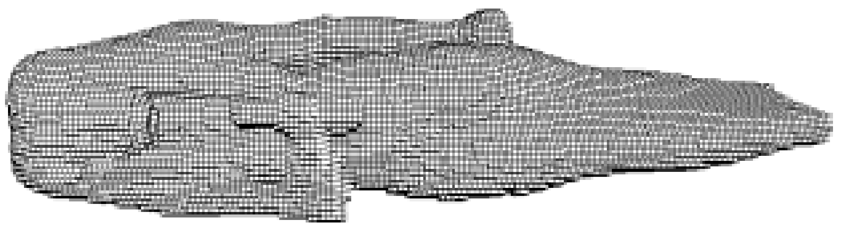

**Fig. 7.** Data leaves of the Isis model.

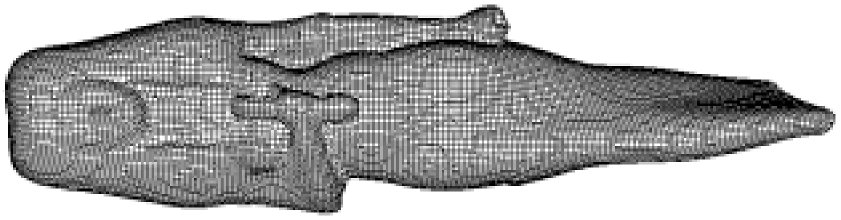

**Fig. 8.** Quadrilateral mesh projected onto the zero set of the ADF..

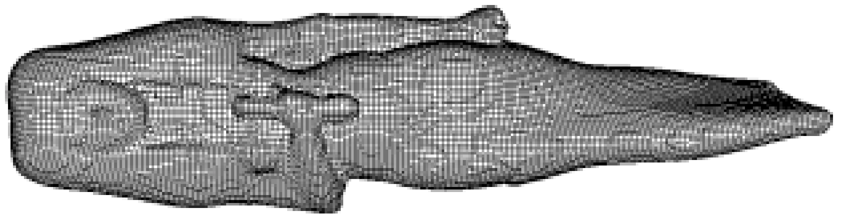

**Fig. 9.** Isis mesh after semi-local collapses.

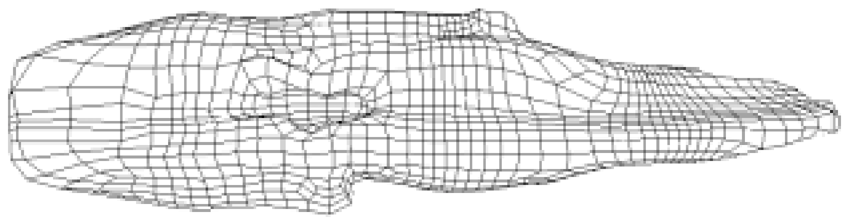

**Fig. 10.** Isis mesh after ring collapses.

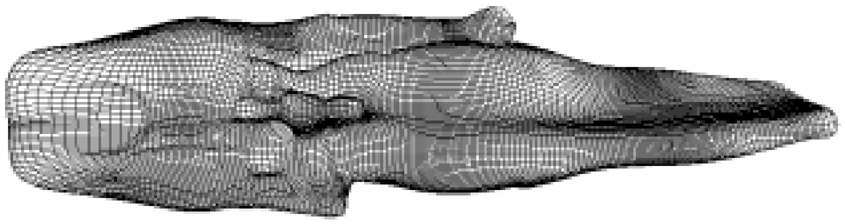

**Fig. 11.** Final mesh, after local collapses and 2 steps of linear subdivision.