Optimal Iso-Surfaces

Carlos Andújar[1], Pere Brunet[2], Antoni Chica[3], Isabel Navazo[4], Jarek Rossignac[5] and Àlvar Vinacua[6]

[1]Universitat Politècnica de Catalunya, andujar@lsi.upc.es
[2]Universitat Politècnica de Catalunya, pere@lsi.upc.es
[3]Universitat Politècnica de Catalunya, achica@iri.upc.es
[4]Universitat Politècnica de Catalunya, isabel@lsi.upc.es
[5]Georgia Institute of Technology, jarek@cc.gatech.edu
[6]Universitat Politècnica de Catalunya, alvar@lsi.upc.es

ABSTRACT

Since the publication of the original Marching Cubes algorithm, numerous variations have been proposed for guaranteeing water-tight constructions of triangulated approximations of iso-surfaces. Most approaches divide the 3D space into cubes that each occupies the space between eight neighboring samples of a regular lattice. The portion of the iso-surface inside a cube may be computed independently of what happens in the other cubes, provided that the constructions for each pair of neighboring cubes agree along their common face. The portion of the iso-surface associated with a cube may consist of one or more connected components, which we call *sheets*. We distinguish three types of decisions in the construction of the iso-surface connectivity: (1) how to split the *X-faces*, which have alternating in/out samples, (2) how many sheets to use in a cube, and (3) how to triangulate each sheet. Previously reported techniques make these decisions based on local criteria, often using pre-computed look-up tables or simple construction rules. Instead, we propose *global* strategies for optimizing several topological and combinatorial measures of the iso-surfaces: triangle count, genus, and number of shells. We describe efficient implementations of these optimizations and the auxiliary data structures developed to support them.

Keywords: Iso-surface extraction, Handle Removal, Topological Ambiguity, Triangle Meshes.

1. INTRODUCTION

Let O be a solid object with one or more connected com-ponents. A discrete representation of O may be obtained by classifying, against O, a set of sample points distributed on the *nodes* of a regular, axis-aligned three-dimensional grid. Nodes lying inside O or on its boundary are labeled as *black* and nodes lying outside O are labeled as *white*. Such a lattice may be constructed in a variety of ways from a polyhedral or curved representation of O through a voxe-lization process. A similar lattice color-coding may be produced by considering the values of a scalar field at each node. If the value is larger than a prescribed threshold, the node is black; otherwise, it is white.

In many application areas, it is useful to convert the dis-crete information stored in the black/white coloring of the grid nodes into a continuous boundary model. Most often, this model is a *triangle mesh* M, which approximates the boundary of the original solid object O. We say that M is a *separating surface* for a black/white grid of nodes when it is a *manifold* triangle mesh that bounds a solid that contains all black nodes and none of the white ones.

Multiple variations of the original Marching Cubes algorithm [8] give a solution to the iso-surface extraction problem. However, current techniques are based on *local criteria* and therefore cannot offer a direct control of topological properties of the extracted mesh.

Sometimes the connectivity, and hence the topology, of M is unambiguously defined by the in/out classification of the samples and therefore all iso-surface extraction algorithms lead to topologically equivalent meshes. But in general, different extraction algorithms may lead to meshes with different topologies. The principal focus of prior art in this area was to guarantee that M is a valid boundary of a solid and possibly to guarantee that it is a two-manifold.

In this paper, we propose an approach for selecting amongst all valid topologies the one that minimizes a desired topological or combinatorial cost, which can be

the total triangle count, the number of connected shells, or the total genus. The proposed algorithm is very efficient, succeeds in optimizing the topological properties of M and does not allow mesh triangles on the faces of the grid cubes (in order to avoid rigid orthogonal mesh triangles).

Our main contributions are:

- The identification and classification of the degrees of freedom in the iso-surface extraction algorithms.
- The design of a new iso-surface extraction algorithm that guarantees a topologically correct iso-surface, by using these degrees of freedom.
- The derivation of two data structures (the X-face graph and the Merge Tree of equivalence classes) that capture the global topological properties of M.
- Efficient algorithms, based on a few atomic operations and on the traversal of the X-face graph, for the optimization of the topology of M.
- The guarantee that M is two-manifold.

After reviewing the previous work in the next section, we present the degrees of freedom that are implicit in standard Iso-surface Extraction algorithms, and propose algorithms and dedicated data structures for iso-surface generation with optimization of the topological properties of the final mesh, M. Finally, we report results on several examples.

2. PREVIOUS WORK

As first noted by Dürst [4], the original Marching Cubes algorithm [8] may produce iso-surfaces with holes due to topologically inconsistent decisions on the reconstruction of ambiguous faces, where the borders used by one incident cube do not match the borders of the other incident cube. Several approaches addressing this problem have been published (see [6],[13] for a review).

Disambiguation techniques reported so far have focused on two major concerns: *topological consistency*, i.e. producing closed surfaces by proper cube polygonalization, and *topological correctness*, i.e. extracting a surface faithful to the geometry of the real surface.

Consistency can be guaranteed by just considering the inside/outside node classification, regardless of the actual data values. *Tetrahedra decomposition* techniques [11],[18] split each cube into five or six tetrahedra, which always exhibit an unambiguous polygonalization. *Preferred polarity* methods decide how to slash an X-face using a uniform criterion: always join black nodes or always join white nodes. This

decision can be implemented either algorithmically [2] or by using a single-entry lookup table [7],[9]. All these techniques are generally simple to implement although they do not solve the correctness problem.

Techniques addressing the topological *correctness* problem infer the proper polygonalization of an ambiguous cube by analyzing its actual data values. As a consequence, these methods are required to provide different polygonalization schemes for each ambiguous cube. Most methods only attempt to assure the correctness of the returned surface on the *boundary* of ambiguous faces. The analysis can be based on *face center resampling* [15],[17], *bilinear interpolation* [12] or *gradient disambiguation* [16]. Only a few methods attempt to recover the original topology also *inside* the ambiguous cubes either by using *critical point analysis* [14] or *trilinear interpolation* [3],[10]. Note that all these techniques are data-dependent and therefore are noise-sensitive and cannot be applied to binary grids.

All the disambiguation techniques discussed so far are based on local decisions and do not offer any explicit control over the global properties of the extracted surface such as genus, triangle count, or number of shells.

Besides Marching Cubes disambiguation, a number of techniques have been proposed for guaranteeing the topological correctness of the resulting surface. When the desired topology and an approximating shape are known beforehand, one can start with an initial estimate of M and then adjust it to match a given shape by applying topology-preserving operations (see e.g. [1]). An alternative approach was proposed for removing small handles or tunnels [5].

In this paper we identify the choices that may be used to control the topology of M while guaranteeing its topological consistency.

3. DEFINITIONS

The space surrounding the solid O may be decomposed into cubic cells in two different ways. Cubes centered around the nodes are usually referred to as the *voxels* of a volumetric model. Each voxel inherits the color of the node located at its center. The union of the black voxels may be used as a coarse approximation of O. In contrast to voxels, the *cubes* considered here span the interstice between 8 nodes of the lattice, which are its *corners*. A cube has 12 edges. Some of them may join white and black nodes and contain the vertices of the mesh $V = \{V_i\}$.

3.1 Sheets and Border Edges

The intersection between the triangle mesh M and the boundary of a given cube forms one or several polygonal cycles. Like most iso-surface generation schemes, we require that the edges of these intersection

curves form a subset of the edges of M. This assumption is fundamental to the Marching Cubes algorithm and to most of its variations, because it ensures that *each triangle of M is contained in a single cube*. Consequently, the triangles of M may be gene-rated by considering one cube at a time.

The set of the triangles of M that lie in a given cube may be empty (when all corners of the cube have the same color) or may form one or more connected components called *sheets*. Each one of these components is a manifold with boundary. It is bounded by one or more simple cycles of *border edges* (contained in the faces of the cube). Let C_L and C_R be the two face-connected cubes sharing a common face f_{LR}. To ensure that M is a manifold without boundary, the border edges of the portion of M in the cube C_L must match the border edges of the portion of M in the neighbor cube C_R. This way, each triangle of M has one neighbor across each one if its edges. This requirement has lead to several publications that disambiguate the MC algorithm, as discussed in Section 2.

3.2 X-faces, Loops and X-cubes

When the four corners of a face have the same color, the face contains no edges of M. When a face f has two vertices (see Fig. 1(a)., Fig. 1(b).), it contains a *single border edge* of M, joining them. This edge will be used as a border edge by the two cubes incident upon f.

Finally, when a face f has alternating black and white corners, and hence four vertices in its boundary, it contains *two border edges* of M. Note that we have a choice in selecting these two edges (Fig. 1(c). and Fig. 1(d).). We use the term *X-face* to refer to such ambiguous faces.
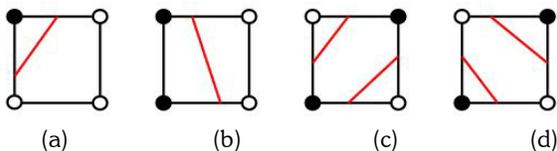

Fig. 1. Faces with two vertices (a, b) generating a single border edge of M. In (c, d), faces generating two border edges of the triangular mesh M.

Once the edges of M have been defined for all the faces, the corresponding borders for any given cube may always be uniquely chained into cycles, which we call *loops*. These loops are the boundary of the portion of M associated with the cube. Note that we can have at most four loops inside a cube (Fig. 2.).
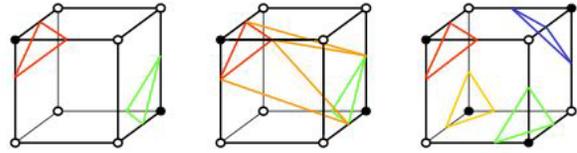

Fig. 2. Configurations with two and with four loops. With two loops (left and center), we have two possibilities: two sheets, each homeomorphic to a disk, or a single sheet homeomorphic to an annulus (tunnel or handle).

X-cubes are defined as cubes having no X-faces but having more than one loop. The only MC configuration leading to an X-cube is the one depicted in Fig. 2-left. The loops in an X-cube can be connected or not, and this does not affect any X-face decision (observe that this is not the case in Fig. 2 right). X-cubes represent quasi-non-manifold parts of O that are not producing X-faces.

4. DEGREES OF FREEDOM IN ISO-SURFACE EXTRACTION

We have identified two tools for controlling the topology and the connectivity of the final mesh: we must decide how to slash X-faces and we must decide (in all X-cubes) whether to have a different sheet per loop or to connect the two loops. Decisions on X-cubes are obviously independent from decisions on X-faces, as they do not affect the X-face slashing decisions. Deciding how to slash X-faces and deciding if loops must be connected or not in X-cubes, gives us a number of degrees of freedom that can be used to optimize the topological properties of the final mesh M.

Moreover, after having decided which way to slash each X-face and which way we connect loops in each X-cube, individual sheets must be triangulated. Although the available choices may impact the total area of M, they do not affect -in our approach- the topology, and hence they are not relevant for the optimization of the number of shells, cells, or the genus. In short,

- Once the choice for all of the X-faces is made, the border edges and hence the loops of the final mesh are completely determined.
- The choice on having separate sheets or connecting loops in individual X-cubes, affects the total number of triangles in the mesh and the topology of the mesh (Fig. 2.).
- The decision on how to select a triangulation for each sheet among the valid ones has no impact on the topology of the mesh, but it does have an effect on the total surface area of the mesh. Therefore we will not discuss these choices in the present paper.

5. TOOLS FOR A GLOBAL APPROACH

The objective of the rest of the paper is to propose several algorithms that use the degrees of freedom shown in the last section in order to optimize the topological properties of the final mesh. In the rest of this section we will show that this is not possible if we are only based on local decisions, and two efficient data structures (representing global information) for the topological optimization in Marching Cubes will be proposed.

The Euler-Poincarè formula for a closed manifold triangle mesh without borders consisting of V vertices, S shells (connected parts of M), and H handles indicates that the total number T of triangles is:

$$T = 2V+4(H-S) \qquad (1)$$

The total number of loops over all cubes will be noted as L, while the total number of *half* border edges over all the faces of the cubes will be noted as B. For a particular cube C, its number of loops and border edges will be noted as L_c and B_c. Note that L is the sum of all L_c and that B equals the sum of all B_c.

Each vertex of M is bounding eight border edges because it lies on an edge of the grid which has four incident square faces of the adjacent cubes. Each face f contains two coincident border edges, one per cube incident upon f. Furthermore, each border edge is bounded by two vertices. Hence, the total number of border edges in all cubes is constant:

$$B = 4V \qquad (2)$$

We also have a relationship among the number of loops L, the number of triangles T and the total number of sheets (denoted by s) inside all cubes. The relation is given by the formula:

$$T = 4V + 2L - 4s \qquad (3)$$

To prove this relationship, let us first look at a single cube C, and let us first assume that we keep the loops separated without connecting them. Then, for a loop having b border edges we generate $t=b-2$ triangles of M. If C has b border edges and l loops, we can sum the number of triangles for each of the loops and obtain $t=b-2l$ as the total number of triangles generated for C. By summing this expression for all cubes and taking into account Eqn. (2) and that $s=L$ (since we do not have connected loops) we can write:

$T=B-2L = 4V-2L = 4V-2L-4(s-L) = 4V+2L-4s$, which is the above equation. To prove it in the general case where we connect some of the loops, we can simply observe that the above equation is invariant under the connecting loops operation: for every connection between two loops in any of the cubes, s decreases in one while T increases by four (Fig. 2.).

Also observe that once X-faces have been fixed, since V is constant, L is also constant and we can conclude that

T and s are always varying in opposite ways: s decreases as T increases, and vice-versa.

Given that V is fixed, the only available variables for our optimization are T, H and S. Unfortunately, we do not have a local control on H and S, which depend on the global structure of the mesh. In the rest of this section we will present the two new data structures that supply the necessary global information to the mesh generation algorithm: the X-face propagation graph and the merge tree of equivalence classes of vertices.

5.1 X-face propagation graph

The *X-face propagation graph* is a convenient tool for deciding on X-face slashing. Consider the abstract graph $G=(V, E)$ where configurations with at least one X-face correspond to graph nodes and where the X-faces correspond to links between graph nodes that represent their incident cubes. For this graph to represent a possible choice of how to slash the X-faces, each graph edge is assigned a binary value indicating its slashing choice.

The X-face graph can be constructed in linear time by a single traversal of the volumetric model. During the traversal, a graph node with label (i,j,k) is inserted into V if the cube (i,j,k) has at least one X-face. An edge connecting node (i,j,k) with any of its three face-neighbors along directions $\{X+, Y+, Z+\}$ is inserted into E if the shared face is an X-face. Since this graph is not oriented, only three faces of each cube are considered, so as to avoid edge repetition.

A simple examination of the 256 cube configurations reveals that cubes can have 0, 1, 2, 3 or 6 X-faces (frequencies over the 256 configurations are shown in Tab. 1.). Since the degree of the graph vertices is bounded by 6, X-face graphs are sparse, with a small number of edges connecting nodes and only a few high-order nodes.

# X-faces	# Configurations	Percentage
0	135	52.7%
1	72	28.1%
2	30	11.7%
3	16	6.2%
4	0	0.0%
5	0	0.0%
6	2	0.7%

Tab. 1. Frequencies of X-faces over the 256 cube configurations. Note that cubes with 4 or 5 X-faces do not exist.

Dataset	Ship (Fig. 8.)	Random 1	Random 2
Resolution	128x128x128	8x8x8	8x8x8
Non-Empty cubes	35,020	312	299
Deg. 1	54	49	34
Deg. 2	21	9	12
Deg. 3	0	9	8
Deg. 6	0	2	2
# arcs	48	53	47
# cycles	0	0	8
# components	27	16	13

Tab. 2. Properties of the X-face graph on the test datasets. *Deg n* stands for the number of graph nodes with degree *n*.

Tab. 2. shows the number of vertices, edges, components and cycles of the X-face graph on a ship engine's room model (Fig. 8.) and on two random datasets. Note that on all test models the X-face graph has many connected com-ponents and few high-ordernodes. Our experiments show that graph cycles might appear, although very rarely in practice. Hence, most of the connected components of the X-face graph are trees.
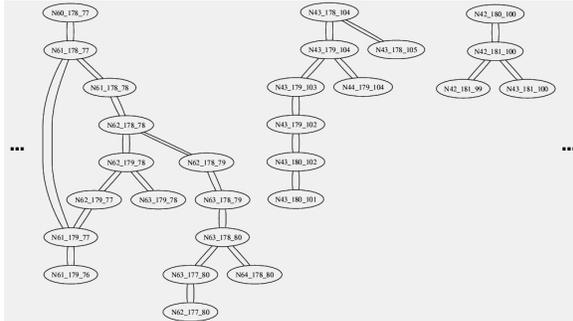

Fig. 3. Main components of the X-face graph on the test model.

5.2 Connectivity merge tree

The second data structure is related with the *equivalence classes* of vertices. These equivalence classes initially encode clusters of vertices connected by border edges that are not contained in X-faces (obviously, internal edges of the cube triangulations do not affect these clusters). In other words, two vertices initially belong to the same class iff they will belong to the same shell regardless of the X-face and X-cube decisions. We will use this additional tool to decide on the individual effect on H and S of a certain choice on the connection of loops of an X-cube or the slashing of an X-face, and we

will update the equivalence classes at each decision. The interest of this data structure relies in the fact that the number of shells S depends on the global connectivity of the mesh, and cannot be determined locally: if a particular slashing choice in a certain X-face connects vertices that were not previously in the same cluster (equivalence class), we are decreasing the total number of shells S. This set of classes gives a strict upper bound to the number of resulting shells, as all vertices in a single equivalence class must lie on the same shell. At the end, when all decisions have been taken, the number of clusters in this data structure is exactly the number S of shells.

To implement it, we use a *merge tree* of the vertices V_i that is initially constructed in a one-sweep process. In this way we store equivalence classes of vertices, modulo the equivalence relation given by the connectivity along a series of border edges that do not belong to an X-face. That is: initially, two vertices V_i and V_j are in the same class if there exists a sequence of vertices $V_i=V_0, V_1, ... V_n =V_j$ such that for $k=0 ... n-1$, the segment $V_i V_{i+1}$ is a border edge that does not belong to an X-face. Notice that these border edges will always remain in the final triangulation.

Using standard data structures we can construct this set of equivalence classes in a single pass of the model, merging classes as we visit the boundary edges of non-X-faces. Finding the canonical representative of a class has a cost of $O(log\ m)$ where m is the number of vertices in the class. Merging two classes can be done in constant time. Therefore the whole data structure is initialized in time bounded by $O(n\ log\ n)+m$, where n is the total number of vertices in the model and m is the number of voxels.

Furthermore, this data structure supports the dynamic computation of the impact of any choice on any given X-face or X-cube. If the end vertices of the chosen border edges on an X-face or on the two loops of an X-cube belong to the same class, the choice does not affect the number of shells. If however they belong to different classes, then the choice of connecting the classes will reduce in one unit the maximum number of shells attainable.

6. THE TOPOLOGICAL OPTIMIZATION ALGORITHM

Our algorithm consists of four main steps, and optimizes the mesh topology by traversing the X-face graph while taking some atomic decisions on how to slash the individual X-faces and on how to connect (or not) the loops inside X-cubes:

InitializeXfaceGraph(G)
InitializeMergeTree(T)
{convert the X-face graph into a tree}

if there are cycles in the X-face graph **then**
   for each graph cycle C **do**
      choose a random X-face of the cycle
      cut the cycle C by choosing a random slash on
      the X-face
   end
end
{*traverse the X-face tree and fix all X-faces*}
while not all X-faces have been fixed **do**
   ChoseOneTreeLeaf(c,f) {leaves correspond to cubes
   c with only one X-face *f*}
   FixLeaf(f,SlashingCriterion)
   UpdateMergeTree(T)
   PruneLeaf(c)
end
{*decide on connecting loops within X-cubes*}
for each cube *c* **do**
   FixXcube(ConnectingCriterion)
   UpdateMergeTree(T)
end
{*final triangulation within cubes*}
for each cube *c* **do**
   Triangulate its border edges with triangles inside *c*
end

6.1 Slashing criteria on a X-face f

Observe that the particular choice on how X-faces are slashed affects the total number of loops in the mesh. Slashing an X-face *f* from one slashing choice to the other will always change by exactly one the number of loops in each one of the two cubes adjacent to *f* (if, before the slash, the two borders of *f* were part of the same loop in one of the cubes, the slash will split that loop and hence increase the number of loops for that cube; if, before the slash, the two borders of f were part of two different loops of the cube, the slash will merge these two loops and decrease by one the loops count for this cube). Therefore, depending on the situation, an X-face slash may either leave *L* unchanged -when the loop count was increased in one of the cubes and decreased in the other one- or increase or decrease it by 2.

We propose the following four criteria (the last two are supported by the Merge Tree encoding the equivalence classes of vertices):

Criterion 1. Take the option that maximizes L_c in every cube *c* sharing the face *f*. We have seen that if L_c increases by one, the loop count in the neighbor cube cannot decrease by more than one. Then, the total count *L* can never decrease. This is used in our greedy algorithm for maximizing *L*.

Criterion 2. Minimize L_c in every cube *c* sharing the face *f*. For the same reasons as in (1), the algorithm will tend to minimize *L*.

Criterion 3. Take the option that does not decrease the number of equivalence classes (if one of the

possible choices does so). The algorithm will tend to maximize *S*, as the final number of equivalence classes equals *S*.

Criterion 4. Take the option that decreases the number of equivalence classes (if one of the possible choices decreases it). The algorithm will tend to minimize *S* (the final number of equivalence classes equals *S*).

6.2 Criteria on how to connect loops within X-cubes

After having fixed the X-faces of the model, we must decide how to connect the remaining free loops (the loops in X-cubes). We must first observe that, when we connect two loops, we have a net increase of *T* in four (due to Eqn. (3)). Therefore, and due to *T=2V+4(H-S)*, we have a net increase of *(H-S)* in one. Taking into account this property, we have the following four options:

Criterion a. Never connect loops. In this case, *(H-S)* is decreased. This decision tends to generate many small blobs -disconnected shells-. In case of noisy models, irrelevant small features can be easily identified and removed.

Criterion b. Always connect loops. Now, *(H-S)* is increased and *S* is decreased. At the end we will have a lower number of equivalence classes and a small *S*.

Criterion c. Two loops are connected when they belong to the same equivalence class. In this case, *S* remains constant. But, as we have an increase in *H-S*, the final consequence is an increase of *H* in one.

Criterion d. Two loops are connected when they do not belong to the same equivalence class. In this case, *S* is reduced by one. As we have an increase in *H-S*, the final consequence is that *H* remains constant.

7. COMBINED DECISIONS

We have sixteen possible combined decisions that can be taken during the traversal of the X-face graph and the visit of the X-cubes (from 1-a to 4-d). In what follows, the notation *min(W)* stands for the subset σ of all meshes *M* such that *W(m)* has its minimum value for all *m* in σ The same applies to *max(W)*. We use the notations *High* and *Low* in those cases where we cannot guarantee a maximum or a minimum.

The four implemented algorithms correspond to 1-a, 2-b, 3-c and 4-d. In these cases, both atomic decisions are complementary:

- In case 1-a we maximize *L* and, since we do not connect loops, we have *s=L*. Then, *T=4V+2L-4s = 4V - 2L* and the maximization of *L* leads to *min(T)*. On the other hand, as *T=2V+4(H-S)*,

we will have *min(H-S)*, with *Low(H)* and *High(S).*

- In case 2-b, *L* is minimized and *s* has its maximum conditioned to the value of *L*. Then *s≥L* and *4(H-S)=2V+2L-4s≤2V-2L\$*. The consequence is a *High(H-S)* with *High(H)* and a *Low(S).*

- In case 3-c, the number of shells is maximized, due to the arguments presented in Section 6. The consequence is *max(S)* with *High(H).*

- Case 4-d tends to *min(S)* with *Low(H)* since we always connect disconnected equivalence classes. We can guarantee that, if the initial solid (before voxelization) had *R* shells, the final mesh *M* will have *R'≤ R* shells.

These four algorithms have been implemented, and their results will be discussed in the next section. The Max/Min global optimal values are always reached provided that the X-face graph has no cycles. In the next section we will see that this is the case in most practical cases.

Our current implementation triangulates each sheet so that no triangle lies on the faces of the cubes. These faces would produce undesirable artefacts when actual data values are used for interpolating the vertex position along the grid edges. It has also to be noted that when using criteria 2 and 3, in order to represent the intersection of two (or more) tunnels, a few cases require the introduction of internal vertices [3].

The remaining twelve combined decisions (1-b, 1-c, .., 4-c) take conflicting decisions on the X-faces and the X-cube loops, their application being less clear. They will be investigated as part of our future work.

On the other hand, tie situations might occur when the two slashing choices for an X-face fulfill a given criterion. This is simply a consequence of the fact that in most cases the mesh having the optimal values for *T, H* or *S* is not unique. A simple random selection can be used to solve the tie whenever we only target a single magnitude. If this is the case, the output mesh will be a random choice from the set of meshes having the optimal value of the target. A much more interesting approach for solving slashing ties consists in using an ordered pair of combined criteria. This fact enables us to optimize a magnitude while trying to keep small another one.

8. EXAMPLES AND DISCUSSION

Fig. 4. shows the results of our algorithm with strategies 1-a, 2-b, and 4-d. In this example 3-c yields the same result as 1-a, and is not shown. The test model consists of the edges of a cube plus all of its diagonals. The model has a resolution of 16x16x16; of the 4096 cells, only 240 have X-faces (but only one per cube), and there are a total of 40 X-cubes. The edges of the cube

are thicker, so they are stable throughout. The diagonals, instead, are thinner, and result in very different reconstructions. In agreement with the expected behavior, we get the triangle, shell and hole-counts summarized in Tab. 3.
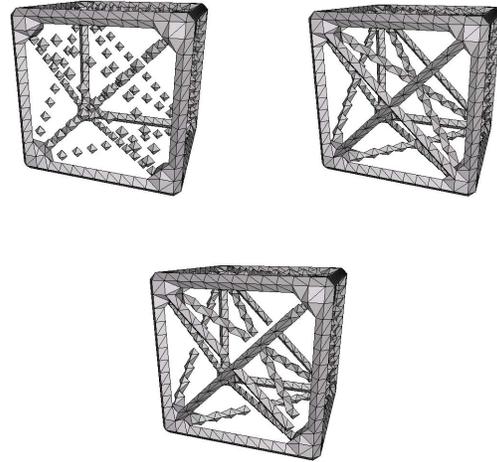

Fig. 4. Results of strategies 1-a, 2-b and 4-d on the test cube.

	T	S	H	T	S	H
1-a	2296	136	5	575,776	1,182	223
2-b	2936	1	30	583,932	339	1,419
3-c	2296	136	5	577,232	1,185	590
4-d	2836	1	5	580,076	340	456

Tab. 3. Number of Triangles, Sheets and Holes in the resulting mesh for the test cube (left, all meshes have 1410 vertices) and the model on Fig. 8. (all meshes have 289,806 vertices).

One can observe how 1-a and 3-c minimize the number of triangles, while 2-b and 4-d minimize the number of shells, but 2-b maximizes genus, whereas 4-d minimizes it.

Although this example is academic, it displays well the behavior of the different variants of the algorithm. For a more realistic, albeit less pedagogical example, Fig. 5. shows a portion of a ship's engine room, sampled at a resolution of 256x256x256. The images at the right of Fig. 5. are enlarged views of a not-so crowded area (highlighted in blue in the leftmost figure), where differences in the results of strategies 2-b and 3-c are readily seen. The models have been rendered here with

510

all vertices fixed at the midpoints of their cell-edges, which accounts for the irregular appearance, but displays clearly the topology of the result (additional images can be found at http://www.lsi.upc.es/\~virtual/CAD.html.
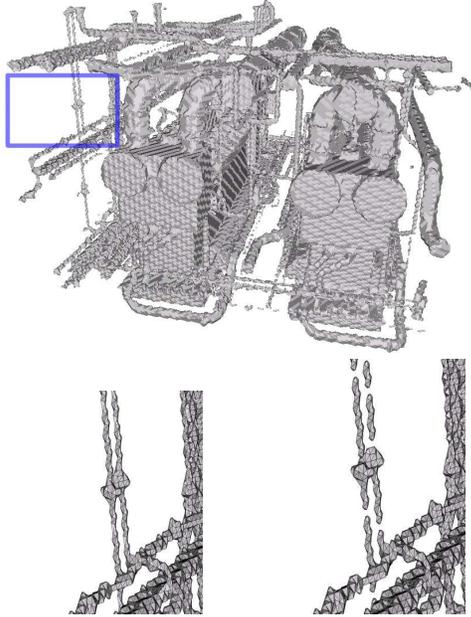

Fig. 5. The result of two different strategies on a complex model. The right-hand side figures are blow-ups of the region marked with a blue rectangle.

The processing of this model on a Pentium-4 running at 1.7GHz with 256Mb of ram took: 0.98 s to build the X-faces graph, 4.23 s to build the connectivity merge-tree, and 0.41 s to solve the graph, for a total running-time overhead of 5.62 s (above the time necessary for the ordinary marching cubes). There are a total of 286,631 non-empty voxels, with 5,739 X-faces; 4,492 cells have only one X-face, 854 have two X-faces, 392 have three X-faces and one has six X-faces. The X-cubes total 289. The merge tree initially has 2,358 components. Tab. 3. summarizes the effect on the topology of the different variants of our algorithm. Notice how these numbers are in agreement with the properties enumerated in Section 7.

9. CONCLUSIONS

In this paper, four algorithms to control and to optimize the topological properties of iso-surfaces have been presented and discussed. Several minimality criteria have been considered: total triangle count, genus, number of shells and combined criteria. The remaining degrees of freedom in iso-surface extraction algorithms have been identified, two data structures (the X-face graph and the Merge Tree of equivalence classes) that retain global topological properties of the final mesh M have been proposed, and several efficient algorithms (based on a few atomic operations and on the traversal of the X-face graph) for the topological control and optimization of the final triangular mesh have been presented and discussed.

Future work will focus on a deeper characterization of the algorithms based on the presented atomic decisions and on the development of area and volume minimization algorithms. Another topic for future work is how to use the proposed atomic decisions for reducing the total number of triangles counting once each group of adjacent, coplanar triangles.

10. ACKNOWLEDGEMENTS

11. REFERENCES

[1] S. Bischoff and L. Kobbelt, Isosurface Reconstruction with Topology Control, in *Proc. Pacific Graphics 2002*, 2002, pp 246-255.

[2] J. Bloomenthal, An implicit surface polygonizer, in *Graphics Gems IV,* Academic Press, 1994, pp 324-349.

[3] P. Cignoni and F. Ganovelli and C. Montani and R. Scopigno, Reconstruction of Topologically Correct and Adaptive Trilinear Isosurfaces, *Computers and Graphics*, Vol. 24, No 3, 2000, pp 399-418.

[4] M. J. Dürst, Letters: Additional Reference to Marching Cubes, *Computer Graphics*, Vol. 22, No. 2, 1988, pp 72-73.

[5] I. Guskov and Z. Wood, Topological noise removal, in *Proc. Graphics Interface 2001*, Canada, 2001, pp 19-26.

[6] S. Hill and J. C. Roberts, Surface models and the resolution of n-dimensional cell ambiguity, in *Graphics Gems V*, Academic Press, 1995, pp 98-106.

[7] J.-O. Lachaud, Topologically Defined Iso-surfaces, in *Proc. 6th Discrete Geometry for Computer Imagery (DGCI'96),* Lyon, France, Springer Verlag, 1996, pp 245-256.

[8] W. Lorensen and H. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphic*, Vol. 21, No. 4, 1987, pp 163-169.

[9] C. Montani and R. Scateni and R. Scopigno, A modified look-up table for implicit disambiguation of Marching Cubes, The Visual Computer, Vol. 10, No 6, 1994, pp 353-355

[10] G. Nielson, On Marching Cubes, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 3, 2003, pp 283—297.

[11] M. Nielson and T.A. Foley and B. Hamann and D. Lane, Visualizing and Modeling Scattered Multivariate Data, IEEE Computer Graphics and Applications, Vol. 11, No. 3, 1991, pp 47-55.

[12] G.M. Nielson and B. Hamann, The Asymptotic Decider : Resolving the Ambiguity in Marching Cubes, in *Proc. of IEEE Visualization 91*, 1991, pp 83-91.

[13] P. Ning and J. Bloomenthal, An evaluation of implicit surface tilers, *IEEE Computer Graphics and Applications*, Vol. 13, No. 6, 1993, pp 33-41.

[14] B. T. Stander and J. C. Hart, Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling, *Computer Graphics (in SIGGRAPH 97 Proceedings)*, Vol. 31, No 1, 1997, pp 279-286.

[15] A. Wallin, Constructing Isosurfaces from CT Data, IEEE Computer Graphics and Applications, Vol. 11, No. 6, 1991, pp 28-33.

[16] J. Wilhelms and A. Van Gelder, Topological Considerations in Isosurface Generation, Computer Graphics, Vol. 24, No. 5, 1990, pp 79-86.

[17] G. Wyvill and C. McPheeters and B. Wyvill, Data structures for soft objects, *The Visual Computer*, Vol. 2, No. 4, 1986, pp 227-234.

[18] C. Zahlten, Piecewise linear approximation of isovalued surfaces, in *Advances in Scientific Visualization*, Spinger-Verlag, 1992, pp 105-118.