Reconstruction of Topologically Correct and Adaptive Trilinear Isosurfaces

P. Cignoni  F. Ganovelli  C. Montani  
R. Scopigno

Istituto di Elaborazione dell’Informazione, Consiglio Nazionale delle Ricerche,
Via S. Maria, 46 - 56126 Pisa ITALY

Email: {cignoni, ganovelli, montani}@iei.pi.cnr.it, r.scopigno@cnuce.cnr.it

July 28, 1999

Abstract

The paper goal is to fit trilinear iso-surfaces out of volume data, by adopting an adaptive mesh refinement approach and therefore supporting a higher accuracy with respect to standard MC solutions. In order to be correct, adaptive refinement must be applied to a topologically correct initial mesh patch. For this reason, we designed a new, Exhaustive Look_Up_Table (ELUT) which encodes multi-entry patterns for each ambiguous configuration. Following the solution proposed by Natarajan, for each ambiguous configuration we choose, at run time, the actual pattern by evaluating the corresponding set of saddle points. Once the corresponding starting patch has been read from the ELUT, it is adaptively refined to fulfill a user-selected accuracy. Refinement is adaptive to ensure that the complexity of the fitted mesh will not become excessive. An evaluation of the results produced on some volume dataset is reported, both in terms of accuracy and complexity of the meshes obtained.

Contact author:
Roberto Scopigno, I.E.I.- C.N.R.,
Via S. Maria 46, 56126 PISA (Italy)
Phone: +39 50 593304  FAX: +39 50 904052
EMail: r.scopigno@cnuce.cnr.it
1 Introduction

The Marching Cubes (MC) algorithm [18] is nowadays the most diffuse technique for the extraction of iso-surfaces out of volume datasets. The reasons for the MC success include its simple logical structure, implying a nearly straightforward implementation, and its computational efficiency. MC has been incorporated in many commercial and public domain visualization systems. Many papers appeared on enhancements, optimization, extensions and applications of this technique [35, 34, 24, 22, 4, 2, 17].

One of the few limitations of MC is the linearity of the reconstruction kernel used. MC adopts a local approach, i.e. each cell is tested for a possible iso-surface patch independently from the others. Each patch is computed by adopting a table-driven approach, and is defined by the position of the vertices located on the cell edges.

The iso-surface patch returned is therefore a linear approximation (planar faces), whose vertices are located on cell edges (this ensures iso-surface $C^0$ continuity between cells) and are computed using linear interpolation.

When a very high resolution dataset is used, the simplicity of the reconstruction filter is not easily perceptible, unless a substantial zooming into the mesh is performed. But if the latter case holds, or if dataset resolution is low, the adoption of a more sophisticated interpolation filter might be required to improve smoothness of the fitted surface.

In this paper we focus on volume data applications based on the analysis/visualization of iso-surfaces; preliminary results have been presented in [1]. We look for methods which produce a “surface-based” output (i.e. ray casting solutions are considered not appropriate), to allow hardware-assisted interactive visualization and data distribution/rendering in web environments [33]. Therefore, we present a solution which improves the accuracy in the reconstruction process, with respect to standard MC solution, using mesh refinement and the trilinear reconstruction filter (see Figure 12). Based on the selection, out of a new multi-entry look-up table, of the correct starting configuration for each active cell, the iso-surface reconstruction process is adaptive, to ensure that the complexity of the fitted mesh will not become excessive (thus reducing or preventing interactive visualization).

2 Previous work

The excessive simplicity of the reconstruction filter used by MC has been pointed out firstly by Fruehauf [9]. He compared images produced by rendering MC output meshes with those generated by adopting a ray casting approach (which generally uses a trilinear reconstruction filter) and showed how much these images might differ. An advantage of ray casting is to allow the adoption of whichever reconstruction filter for the interpolation of field values. In fact, a number of different interpolation filters have been proposed and evaluated [19, 20, 21, 8] in order to compute more precisely both field values and gradients.

Unfortunately, using non-linear reconstruction filters with ray casting leads to the computation of an image showing the isosurface we are interested in (a view-dependent process), rather than to extract explicitly the iso-surface (a view-independent process). For many applications, producing an image is not enough. The explicit recon-
struction of surface geometries may be needed, for example, to use these geometries in a virtual simulation environment. Moreover, a shortcoming of the ray casting approach is the non-interactive rendering time (unless high-performance computer are used [28]). For these reasons, iso-surfaces quality cannot be improved in many applications by simply adopting a ray casting solution together with a more sophisticated reconstruction filter.

The technique proposed in the paper adopts a regular mesh refinement approach. The idea of improving the quality of a mesh by [recursively] applying a sequence of local refinements [6] is not new. Many approaches based on the refinement of triangle meshes have been proposed: to construct adaptive piecewise linear representations of implicit surfaces [11, 32]; to reconstruct adaptively the surface of three-dimensional objects from multiple range images, by starting from an initial rough triangular surface and then both adaptively refining and moving the surface vertices towards the interpolated isosurface by minimizing an energy function [25, 10]; to extract a surface out of sampled scalar/vectorial 3D datasets starting from an initial surface seed and then applying an iterative surface inflation process [30]; and to refine a surface under a strict surface curvature approximation constraint [16]. In particular, the refinement rule adopted in this paper is similar to the one adopted in [10], but the focus is here the reconstruction of a very precise piecewise approximation of the trilinear isosurface. Conversely, the focus in [10] is the reconstruction of a compact and good approximation of range scanned surface; the joint use of a vertex geometry optimization and of a mesh refinement process can produce very good results (a compact mesh which represent accurately a given distance field), but does not ensure the topologic correctness of the surface produced (the user has to judge in [10] if the initial mesh gives a sufficiently accurate representation of the object topology).

Iso-surface extraction has also been performed using triangular rational quadratic Bézier patches [12]. A \( C^0 \) continuous surface consisting of rational-quadratic surface patches may be built to interpolate the triangles in the isosurface. The Bézier control points of a single rational-quadratic surface patch are computed on the base of the coordinates of the vertices of the underlying triangle and the gradients and Hessians associated with the vertices.

3 MC with a trilinear reconstruction filter

The goal is to support a trilinear reconstruction filter in a surface fitting context. The proposed solution has been designed as an extension to the classical MC approach and follows the following list of requirements:

- surface fitting is performed with a local approach (each cell is processed independently from the others);
- given a generic reconstruction filter, the simplicial surface mesh produced must approximate the ideal iso-surface defined by the given reconstruction kernel at a user-selected approximation;
- \( C^0 \) continuity has to be ensured.

The idea is therefore to enhance the MC algorithm by giving the possibility to progressively refine each surface patch until the requested accuracy is fulfilled (Figure 12). The only limitation of the proposed approach is that the reconstruction filter must fulfill a crucial MC constraint: for each cell, the topology of the associated
iso-surface patch must be univocally determined on the basis of local considerations. This is to preserve the simple and efficient look-up based MC approach.

The overall pipeline is as follows (with $V$ the volume dataset, $q$ the iso-surface threshold, $\varepsilon$ the user selected accuracy, and $\text{maxRec}$ the maximum level of recursion which may be produced).

**PreciseMC**($V, q, \varepsilon, \text{maxRec}$):

FOR EACH cell $c_{i,j,k} \in V$ DO

fit an iso-surface patch $S$ on $c_{i,j,k}$ (using an MC approach);

FOR EACH face $f \in S$ DO

TryToRefine($f, V, q, \varepsilon, \text{maxRec}, 1$);

**TryToRefine** ($f, V, q, \varepsilon, \text{maxRec}, \text{lev}$):

FOR EACH sampling point $p_i$ on $f$ DO

evaluate the approximation $\varepsilon_i$ of $f$ in $p_i$ with respect to the trilinear filter;

IF $\varepsilon_i > \varepsilon$ THEN $\text{Split\_points} := \text{Split\_points} + p_i$;

IF $\text{Split\_points} = {}$ THEN output($f$)
ELSE refine $f$ in $\{f_j\}$ (using $\text{Split\_points}$);

FOR EACH $f_j$ DO

IF $\text{lev} \leq \text{maxRec}$

THEN TryToRefine($f_j, V, q, \varepsilon, \text{maxRec}, \text{lev} + 1$)
ELSE output($f_j$);

Mesh refinement is adaptive, because we subdivide only those faces which do not approximate sufficiently the ideal iso-surface. The recursive refinement is halted either if a simplicial approximation which satisfies the given accuracy is found, or a maximum recursive level is reached. The user may therefore drive the fitting process by selecting two parameters: the accuracy $\varepsilon$ and the maximum number of refinement steps $\text{maxRec}$.

### 3.1 Evaluation of the approximation and refinement rules

The accuracy of each face can be evaluated at least in two different manners. A first possibility is to measure a field-based difference (i.e. given a simplicial mesh $S$ which approximates the reconstruction filter $K$, compute the difference between the given threshold value $q$ and the value of the field in the points of $S$). A second approach is to measure a geometric difference between the current iso-surface $S$ and the ideal iso-surface $S_K$ (e.g. evaluate the Hausdorff distance between $S$ and $S_K$). Both these evaluations may be performed in a precise or in an approximate manner.

We evaluate an approximate geometric difference by computing the distance between each face of $S$ and the
ideal iso-surface on a discrete number of sampling points. There are many different criteria to select the set of sampling points. A possible choice may be to select the midpoints of the edges (which leads to quaternary subdivision [6]). For each of these points $p_i$, we evaluate the distance between $p_i$ and a corresponding point $p'_i$ on the ideal iso-surface $S_K$. If this distance is greater than the selected accuracy threshold $\epsilon$, we classify point $p_i$ as a splitting point. The current face is then refined by inserting the splitting points and the new local triangulation is simply determined by an ad hoc table, see Figure 1. We adopt therefore an heuristic refinement approach [6], to allow refinement of only a subset of edges. In this case, four different configurations are possible: the three ones represented in Figure 1 plus the one with no splitting vertices. Let us call this refinement criterion Rule A. Other rules are also possible. A first variation of Rule A is to evaluate four splitting points, adding the baricenter to the three edge midpoints. This leads to a slightly more complex pattern of different triangulations, which can be coded analogously by the definition of a table of different configurations (see Figure 2). Let us call it Rule B. But rule B has a disadvantage: because we evaluate all the four split candidates at once, we may decide to split on the central point also when its insertion does not really improve mesh approximation. For example, look at the case when the distance between the central point $p_c$ and the ideal iso-surface $S_K$ is greater than $\epsilon$, and thus, according to Rule B, we use $p_c$ to split face $f$. But, if we consider the refinement obtained by using only the other three split points, then in many cases the actual difference between the two refined meshes could be much smaller than $\epsilon$ (see Figure 3). In those cases we create three new faces that are not really needed to obtain the required approximation. To prevent an excessive increase in the number of faces due to the above reason, we introduce two alternative criteria based on four sampling points. The first one, called Rule A1, extends criterion
A by sampling the central point in a second step, only when none of the three edge midpoints is classified as a splitting one (see Figure 4).

The second one, Rule A2, always evaluates a fourth sampling point in a second phase. In this case, the initial location of this candidate point is not on the plane of the face to be split, but it depends directly on the current splitting points locations (after relocation on the ideal surface $S_K$, see Figure 5).

The three different criteria result into different meshes; see for example the evaluation of the results produced with the two refinement patterns reported in Section 5.

### 3.2 Splitting point displacement

In the previous discussion we have not specified how do we find the point $p'$ on the ideal iso-surface $S_K$ which corresponds to the potential splitting point $p$ we are evaluating.

The solution we choose is a sort of ray-tracing: we start a sampling process on the line which originates from the current point $p$ and is parallel to the field gradient in $p$ itself. As far as we generate points on this line, we compute the corresponding field value and gradient using the trilinear reconstruction filter. Sampling terminates as soon as we reach the searched value $q$ (i.e. point $p'$). Once we have found this point, we compute the Euclidean distance between $p$ and $p'$.

Another manner to find point $p'$ can be to analytically compute the nearest intersection between the gradient
half-line and the local section of ideal isosurface $S_K$. This is surely possible in the case of a tri-linear reconstruction filter [28], but we preferred to adopt the previous solution, based on ray sampling, for the following two reasons. First, in order to be more general: given a reconstruction filter $K$, we only need to know how to compute $K$ in a generic point $p$. Second, because in some cases the intersection point between the trilinear filter and a given ray can be located in the exterior space with respect to the current cell (more details on that problem are in the next subsection). This is the main reason for choosing a point-sampling strategy.

The robustness of geometrical computations is obviously a fundamental issue. All of the splitting points are shared between pairs of incident faces. To prevent the occurrence of different values in the replicated evaluation of a candidate splitting point (and potential topological inconsistencies), we must avoid redundant evaluations. All of the evaluated splitting point coordinates (plus accessory information, such as the local geometrical approximation and the gradients computed on such points) are therefore stored in a hash table, to prevent redundant evaluations.

### 3.3 Local vs. non-local output

A characteristic of MC is that all fitted patches are contained in the corresponding cells. This characteristic is no more valid with the refinement approach described above, unless we determine sampling direction in a rather different manner. Because we refine a mesh by moving points in the direction of the gradient, the resulting mesh section may be partially contained in an adjacent cell or, conversely, it may be curved towards the interior of the current cell. See Figure 6 for an example (leftmost image). This is because in proximity of a cell face, following the gradient, we may exit from the current cell and look for the ideal iso-surface $S_K$ in the adjacent one.

If we want to maintain the locality of the iso-surface fitting process, than we must modify the method used to
relocate splitting points. During the search for $S_K$, for each sampling point $p$ we compute the field value and the new gradient (that is used to move to the following sample). All points are relocated exactly in the interior of the current cell if we replace the gradient $G$ with $G'/0$:

$$G'(p) = \left( 2 \ast f(\Delta_x) \ast G_x(p), 2 \ast f(\Delta_y) \ast G_y(p), 2 \ast f(\Delta_z) \ast G_z(p) \right)$$

where $(\Delta_x, \Delta_y, \Delta_z)$ are the coordinates of the sampling point $p$ with respect to the current cell $(0 \leq \Delta_x, \Delta_y, \Delta_z \leq 1)$ and $f(\Delta_w)$ is defined as follows:

$$f(\Delta_w) = \begin{cases} 
\Delta_w & \text{if } \Delta_w \leq 0.5 \\
1 - \Delta_w & \text{if } \Delta_w > 0.5
\end{cases}$$

In particular, all of the splitting points that are located on a cell face, are moved along such face (because on each cell face, $G'$ is exactly the projection of $G$ on that face). Two meshes obtained by relocating splitting points using standard gradient (leftmost image) or the $G'$ direction (rightmost image) are shown in Figure 6.

### 4 Management of topological anomalies

The proposed approach produces an adjustable-precision approximation of the ideal iso-surface by simply refining the standard MC linear patch. This implies that, given a cell and its configuration, we need the initial mesh patch to be topologically correct, otherwise the refinement process can produce erroneous results.

As first noted by Dürst[7], the MC algorithm can produce topological inconsistencies (i.e., holes) in the extracted isosurface(s). An example for this is shown in Figure 4: the polygonization adopted by the algorithm for the two cells of the example implies the generation of a hole on the boundary. This problem occurs in the polygonization of ambiguous cells, i.e. cells which have one or more ambiguous faces. A face is called ambiguous if the scalar values of two diagonally opposite vertices are greater than the selected threshold and the other two
Figure 7: A hole (gray) in the surface generated by the MC algorithm.

vertices are lower than the threshold. In these cases, different triangulations patterns can be selected for the same cell.

Actually, the problem of the MC topological ambiguity, as noted by Ning and Bloomenthal [27], has to be more properly splitted into two aspects:

- consistency: the polygonization of each cell do not result in improperly closed surfaces (i.e., surfaces with holes);
- correctness: the extracted isosurface is faithful to the geometry of the real surface.

Topological consistency is very important because it prevents the appearance of easily detected artifacts. Topological correctness is also desirable but it becomes indispensable in our application in which the real underlying function is (or is assumed to be) known and magnification of the ambiguous regions and progressive refinements of the cells’ polygonization are required. In fact, in these cases, the topological decisions have to strictly correspond to the true functional variations.

Numerous solutions have been proposed in the last ten years for the solution of the MC topological problems; they can be classified on the basis of the strategy they adopt. In general:

- the techniques proposed for the consistency problem are the simplest to implement and the fastest in execution. They do not solve the correctness problem;
- the solution of the correctness problem implies also the solution of the consistency problem;
- most of the algorithms for the correctness problem assure the correctness of the returned isosurface(s) only on the boundary of the ambiguous cell, not in the inner part.

We will pay particular attention to the last statement because we are interested in applications in which the isosurface has to be refined inside the cells on the base of a given trilinear interpolant: the topological decisions we have to take must be consistent with the trilinear functional everywhere in the examined cell.

A review [27, 13] of the proposed solutions includes (but it is certainly not limited to):
- **Preferred Polarity**: in order to determine in a consistent way the direction of contours onto the ambiguous faces of an ambiguous cell, these methods are based on the polarity of the four face vertices: they always separate the positive vertices (with respect to the threshold) and join the negative ones, or vice versa. The choice to join or separate vertices is not based on the values of the field; for this reason these methods assure consistency but not correctness. Preferred polarity methods are very fast and simple to implement: they can be based on contour tracking algorithms in order to take care of the polarity during the surface reconstruction [3], or on the definition of ad hoc configurations’ look-up tables [22].

- **Face Adjacency**: in case of topological ambiguity, an analysis of the cell(s) which is (are) adjacent on the ambiguous face(s) to the cell under examination is performed [7, 38]. This analysis implies the local choice of a preferred polarity and therefore it assures consistency but not correctness.

- **Cell Decomposition**: these methods solve the MC consistency problem by decomposing the ambiguous cell into tetrahedral subcells and polygonizing these latter [23], or by recursively subdividing the cell into eight smaller cubes [31] and trying, in this way, to eliminate ambiguous situations. The main problem of these techniques is that cell subdivision implies smaller triangles and, therefore, larger meshes. Again, these methods solve the consistency problem and not the correctness one.

- **Topology Inference**: with these methods, the correct (and consistent) topology of an ambiguous face is inferred by the analysis of the cell data values. This analysis can be performed in different modes: face center resampling [36], bilinear [26] or trilinear interpolation [24], or gradient heuristics [31]. If the interpolant over the cell is well known, the methods based on interpolation or gradient heuristics are assured to solve the correctness problem. These technique generally imply a greater number of tests and computations.

Topology Inference techniques are the only methods able to solve both the MC correctness and consistency problem. Most of these methods limit themselves to the correctness of the isosurface on the boundary faces of the cell: no assumption is made for the cell interior. Natarajan method [24] assures a complete correctness over the whole cell. Our solution is therefore based on the Natarajan method.

According to the Natarajan’s solution, the topological correctness over the cell can be obtained by analysing the value the interpolant assumes on the saddle points of the ambiguous faces and, eventually, on the saddle point internal to the cell (body saddle point). From a geometric point of view, each face saddle point represents the origin of the asymptotes of the hyperbola which the bilinear interpolant describes on that face. Depending on the value of the saddle point with respect to the threshold it is possible to decide which opposite vertices are to be geometrically connected. A similar discussion holds for the internal (body) saddle point. Face and body saddle point values and positions can be computed as described in Appendix A. In its original description, the Natarajan solution is specified by a rather intricate procedure.

On the basis of Natarajan’s analysis, we designed a new multi-entry MC look-up table (called Exhaustive Look-Up Table - ELUT) which provides, for each cell configuration and for each combination of the values of the saddle points of the ambiguous faces and/or of the body saddle point, the correct isosurface patch contained
into the cell. In Appendix B the new ELUT is presented and its creation is discussed. As noted in the Appendix B, our ELUT does not provide, for certain ambiguous configurations, the geometric patterns corresponding to all of the possible combinations of the saddle point value with respect the threshold. Some of these combinations are missing because they can not take place in real cases. This is demonstrated in Appendix C.

Using the correct initial isosurface patch given by our ELUT, we are sure that the refinement process will converge, and that the results are sound.

An example of the different topologies which may be generated from the same configuration of the cell vertices is shown in Figure 17.

5 Experimental results

The current prototypal implementation of the approach presented, PreciseMC, has been coded in C++. Its GUI has been designed to be used in a distributed client/server environment, using the Java language.

The MC Exhaustive LookUpTable (ELUT) we used to decide our initial geometry is available free on our web site\(^1\). This ELUT can also be used in regular MC algorithms where correctness, besides consistency, is mandatory.

In our experiments the overhead due to the computation of one or more saddle points resulted very small if compared to the processing time spent for the determination of the entire initial geometry. Our ”worse” dataset presented only 0.1 % ambiguous cells with respect to the active ones, and therefore only a few dozens of floating points operation were needed.

All the times reported have been obtained on a PentiumPro 200Mhz pc (64 MB RAM). The adoption of an hash table to store splitting points and vertices data (coordinates, approximation error, interpolated normals) has proved highly effective both to improve robustness and to reduce computation times by avoiding redundant computations (in average, times are halved).

\(^1\)For a free downloading of the ELUT see at: http://vcg.iei.pi.cnr.it
The codes has been evaluated on many datasets. We report here results on some iso-surfaces fitted on an 11x14x14 section of the SOD dataset\(^2\), and on two synthetic datasets: “tube” (resolution 9x9x9)\(^3\), and “F” (resolution 30x30x4).

Two isosurfaces fitted on the “tube” dataset are shown in Figure 8 (MC on the left, PreciseMC on the right).

The recursive refinement process is shown in Figure 13, where we show the meshes produced at each level of recursion on the “effe” dataset; in each image the faces generated in the current recursive step are colored in red, while the gray levels denote the refinement step (darker gray means earlier refinement).

Figure 14 shows two meshes extracted from the SOD dataset with threshold 50. The one on the left has been fitted using standard MC, and is composed of 654 faces. The mesh in the two rightmost images (top and side view) has been fitted with PreciseMC, using at most five levels of recursion. It is composed of 14,244 faces. Note the difference in the section where the iso-surfaces bifurcate: the mesh fitted with PreciseMC is much more smoother and thinner. Another mesh extracted from a section of the SOD dataset is in Figure 16.

To evaluate the different results produced with the three refinement rules introduced in Section 3.1, we show in Table 1 the different complexity (number of faces, \# faces) of the meshes produced by PreciseMC, under the same accuracy \(\varepsilon\). The table reports also the maximum (maxL) and mean (meanL) recursive depth required to reach the user-selected approximation \(\varepsilon\). Accuracy is given using cell edge size units (e.g. \(\varepsilon=1/100\) means accuracy not less than 1/100 of the cell edge). In Table 2 some running times for the SOD dataset with different settings for the approximation accuracy and the maximum number of recursive decomposition are given.

From the analysis of these results, we can say that Rule A1 has to be preferred, because it is sufficiently fast (mean running time of A1 are 1.05 vs Rule A and 0.71 vs Rule A2), it produces a compact output with a slightly deeper recursion than A2 and, obviously, it allows more precise refinement than A (if we compare meshes of approximately the same size). In particular, Rule A2 shows an excessive increase in the size of the mesh produced (nearly the double than the meshes produced with Rules A and A1).

We also measured the actual difference between meshes extracted from the SOD dataset using the Metro tool [5]. Metro numerically compares two triangle meshes \(S_1\) and \(S_2\). It performs a surface sampling process on the first mesh, and for each elementary surface parcel it computes a point–to–surface distance with the other mesh. At the end of the sampling process, Metro switches the meshes and execute sampling again, to get a symmetric evaluation of the error. Metro returns both numerical and visual evaluations of surface meshes “likeness”. We have compared the two meshes extracted from the SOD dataset (see Figure 14); the MC mesh is composed of 654 faces, and the PreciseMC one of 14,244. The Metro test gave a maximal distance between the two meshes of 0.39 units (i.e. cell edge length), and a mean distance of 0.12 units. A snapshot of the Metro visual output is shown in Figure 15; it is zoomed to view in detail the mesh section which describes the thin bifurcation.

---

\(^2\)SOD is a regular rectilinear dataset (electron density map of an enzyme), produced by D. McRee, Scripps Clinic, La Jolla (CA), and kindly distributed by the University of North Carolina at Chapel Hill.

\(^3\)Tube is the same sample dataset defined and used in [12].
6 Conclusions

A new iso-surface fitting solution has been presented, PreciseMC. Given a trilinear reconstruction filter, it improves the accuracy of the reconstruction process, with respect to standard MC solutions, using an approach based on mesh refinement. The iso-surface reconstruction process is adaptive, to ensure that the complexity of the fitted mesh will not become excessive.

Three different refinement rules have been evaluated. Rule A1 (based on the evaluation of three split points, plus the centroid in case none of the previous is classified as a split point) gave the best results; it requires short processing times and better accuracy.

From a qualitative point of view, the results obtained with PreciseMC are much smoother, more regular and, in some cases, also thinner than those produced with standard MC. PreciseMC shows therefore great potential in medical applications, where it may be selectively adopted to improve the quality of those surfaces which correspond to very thin specimens, such as blood vessels or other internal small cavities. This may improve either the measures taken on the extracted mesh (e.g. to evaluate the occurrence of stenosis or aneurysms in the vessel [29]) or the quality of virtual navigation [14].

The presented solution might lead to a rapid growth of the surface size (number of faces). For most applications this is correct because the user needs to obtain a precise and smooth isosurface. When the number of triangles becomes a critical issue, a possible solution is to speedup rendering by adopting one of the view-dependent LOD approaches [15, 37].

To ensure correctness and convergence of the recursive refinement process, each initial isosurface patch has to be topologically correct. The strategy proposed by Natarajan has been adopted to guarantee topological correctness. With respect to the original proposal, we converted Natarajan procedural-based solution into an Exhaustive LUT, which is much simpler to use and to include into existing codes. The definition of the ELUT was instrumental to the implementation of our adaptive trilinear isosurface fitter, but it has obviously a much wider applicability. Consequently, the ELUT was made available on our web, to allow other people in the field to experiment easily with topological correct MC.

Acknowledgements

This work was partially supported by the Progetto Coordinato “Modelli Multirisoluzione per la Visualizzazione di Campi Scalari Multidimensionali” of the Italian National Research Council.

A Saddle Points Computation

In this Appendix we recall the simple mathematical operations to be performed in order to compute the exact location and value of the cell face and body saddle points; this is based on the assumption of a trilinear interpolant
over the cell. The reader may refer to the Natarajan’s paper [24] for a deeper discussion on the correctness of the method.

Given a generic cell \( V \) with scalar values at the vertices \( v_0, v_1, \cdots, v_7 \) as indicated in Figure 9, the trilinear interpolant in the cell can be written as:

\[
T(x, y, z) = axyz + bxy + cyz + dxz + ex + fy + gz + h
\]

where

\[
\begin{align*}
a &= v_1 + v_3 + v_4 + v_6 - v_0 - v_7 - v_5 - v_2, \\
b &= v_0 + v_2 - v_1 - v_3, \\
c &= v_0 + v_7 - v_4 - v_3, \\
d &= v_0 + v_5 - v_1 - v_4, \\
e &= v_1 - v_0, \\
f &= v_3 - v_0, \\
g &= v_4 - v_0, \\
h &= v_0.
\end{align*}
\]

For each face of the cube, \( T(x, y, z) \) reduces to the bilinear interpolant for that face. For instance, for the face \( x = 0 \), we have:

\[
T(0, y, z) = cyz + fy + gz + h
\]

and the saddle point coordinates can be determined by solving the system of equations:

\[
\begin{align*}
\delta T/\delta y &= cz + f = 0 \\
\delta T/\delta z &= cy + g = 0
\end{align*}
\]

which locates the saddle point at \( SP_{x=0} \equiv [0, -g/c, -f/c] \), with a saddle value (by substituting \( SP_{x=0} \) in \( T(x, y, z) \)) of \( SV_{x=0} = h - fg/c \).
The face saddle values for the other faces of the cube can be computed in the same way, and we obtain:

\[
\begin{align*}
SV_{x=0} &= h - \frac{fg}{c} = \frac{v_0 v_4 - v_0 v_7}{v_3 + v_4 - v_0 - v_7}, \\
SV_{x=1} &= \frac{ae + ce + ah + ch - bd - df - bg - fg}{a + c} = \frac{v_1 v_0 - v_2 v_5}{v_1 + v_0 - v_2 - v_5}, \\
SV_{y=0} &= h - \frac{eg}{d} = \frac{v_1 v_4 - v_0 v_5}{v_1 + v_4 - v_0 - v_5}, \\
SV_{y=1} &= \frac{af + df + ah + dh - bc - ce - bg - eg}{a + d} = \frac{v_3 v_0 - v_2 v_7}{v_3 + v_0 - v_2 - v_7}, \\
SV_{z=0} &= h - \frac{ef}{b} = \frac{v_1 v_3 - v_0 v_2}{v_1 + v_3 - v_0 - v_2}, \\
SV_{z=1} &= \frac{ag + bg + ah + bh - cd - ce - df - ef}{a + b} = \frac{v_4 v_0 - v_3 v_5}{v_4 + v_0 - v_3 - v_5}.
\end{align*}
\]

The body saddle point is obtained by solving the system of equations:

\[
\begin{align*}
\delta T/\delta x &= ayz + by + dz + e = 0 \\
\delta T/\delta y &= axz + bx + cz + f = 0 \\
\delta T/\delta z &= ayz + cy + dx + g = 0
\end{align*}
\]

which gives the solutions:

\[
SP^1_b \equiv \left[ \frac{bcd - ace + R}{a(ae - bd)}, \frac{bcd - adf + R}{a(af - bc)}, \frac{bcd - abg + R}{a(ag - cd)} \right],
\]

and

\[
SP^2_b \equiv \left[ \frac{bcd - ace - R}{a(ae - bd)}, \frac{bcd - adf - R}{a(af - bc)}, \frac{bcd - abg - R}{a(ag - cd)} \right],
\]

(where \( R = \sqrt{(bd - ae)(bc - af)(cd - ag)} \)), and the corresponding saddle values of:

\[
SV^1_b = \frac{a^2 h - abg - adf - ace + 2bcd + 2R}{a^2},
\]

and

\[
SV^2_b = \frac{a^2 h - abg - adf - ace + 2bcd - 2R}{a^2}.
\]

We discard the solutions which are outside of the unit cube.

B The Exhaustive Look-Up Table

Based on the Natarajan [24] work, in this Appendix we propose an extension of the MC look-up table, called Exhaustive Look-Up Table (ELUT). The ELUT permits to the user to simply select the triangular patches to be adopted for each cell on the basis of the cell’s configuration and, in the cases of ambiguous faces, on the basis of the sign (with respect to the threshold) of the corresponding face and, eventually, body saddle points.
In our proposal, each entry of the ELUT contains now the number $k$ of different isosurface patches provided for that cell configuration and a pointer to the corresponding list of $k$ different patterns. Each pattern reports the sign of the saddle points (with respect to the threshold) and the list of triangles to be generated.

For example, the two different patterns of the ELUT’s entry for the configuration no. 160 (case no. 2) are shown in Figure 10. The entries $0, 1, \ldots, 5$ in the figure refer to the saddle points lying on faces $x = 0, x = 1, y = 0, y = 1, z = 0, \text{ and } z = 1$, respectively; entry $b$ stores the sign of the body saddle point. In the case of the example, if the saddle point on face $z = 1$ assumes a value lower than the selected threshold (coded with a $\circ$ sign), the upper-most geometric pattern of Figure 10 is adopted; if, conversely, the saddle value is greater than the threshold (coded with a $\bullet$ sign), than the pattern represented below in Figure 10 is adopted. A $\times$ sign in the table means that the corresponding saddle value does not need to be computed.

Table 3 shows the cells’ isosurface patches (or patterns) provided for each MC cell configuration. For the unambiguous cases, an analysis of the saddle points is not required. Tables 4-10 show the possible patches associated to each ambiguous configuration of the Marching Cubes algorithm; the proposed triangular patches are based on the analysis of the saddle point values of the critical faces. In Tables 4-10, the saddle point $0 [1, 2, 3, 4, 5]$ is the saddle point lying onto the face $x = 0 [x = 1, y = 0, y = 1, z = 0, z = 1]$; the saddle point $b$ represents the body saddle point. Again, a $\circ$ sign means that the corresponding saddle point value has to be lower than the threshold value; it has to be greater in the case of the $\bullet$ sign; and, finally, all the saddle points marked with a $\times$ sign can be ignored. In the case of symmetric configurations of the saddle points, the tables show just one geometric pattern and, between horizontal lines, the corresponding combination of face and body saddle points.

Fortunately, the construction of the table does not require the coding of all the configurations. Two positive aspects contribute to considerably reduce this job:

- for each MC case (ambiguous or not) only one configuration needs to be coded. All of the symmetric configurations are obtained by rotating, reflecting, mirroring or complementing the basic one;
- for some MC case, not all the geometric patterns due to the different combinations of the values of the face and body saddle points can take place: in the MC case 13, for example, only 46 out of 128 combinations
Figure 11: The Exhaustive LUT uses tunnels (right) instead of faces lying on the faces (left)

can happen. This is demonstrated in Appendix C.

For this reasons our multy-entry MC ELUT contains 798 different polygonizations, but only 88 of them were
manually encoded. It has also to be noted that the isosurface patches adopted in the actual look-up table do not
present triangles lying on the faces of the cell. This is done in order to avoid ambiguity in the refinement process,
in particular in the possible refinement of faces lying on the faces of a cell. In the actual look-up table, all these
flat faces are represented by means of open tunnels. An example is shown in Figure 11.

In order to represent the intersection of two (or more) tunnels, see Figure 11, it is necessary to have one (or more)
internal vertex(-ices). We approximate these points with opportune points on the cell’s diagonal corresponding to
the intersection we are searching for.

C Reducing the number of entries of the MC Exhaustive Look-Up Table

For some topological ambiguous configurations the MC Exhaustive Look-Up Table in Appendix B does not
contain all the possible combinations of face and body saddle points. Table 10, for example, holds just 46
different combinations out of the 128 possible ones (we do not consider the degenerate cases).

These configurations are missing simply because they cannot take place. In order to demonstrate this we will
show that for one possible configuration, o o o o • • x in Table 11, the corresponding system of inequalities (being
\( t \) the threshold):

\[
\begin{align*}
SV_{x=0} &= \frac{v_0 v_1 - v_3 v_4}{v_0 + v_1 - v_3 - v_4} < t \\
SV_{x=1} &= \frac{v_2 v_5 - v_1 v_6}{v_2 + v_5 - v_1 - v_6} < t
\end{align*}
\]
has no solutions.

It’s not easy, in the general case, to find a feasible solution for a non-linear system of inequalities or simply to prove that the system hasn’t solutions. Fortunately, in this particular case we can show that the system does not have solutions with few algebraic steps. Let’s consider, for example, the inequality representing the saddle point value $SV_{z=0}$:

$$\frac{v_0v_2 - v_1v_3}{v_0 + v_2 - v_1 - v_3} > t$$

(2)

Applying the following substitutions:

$$\tilde{v}_i = v_i - t, \quad i = 0, 2 \quad \tilde{v}_i = -v_i - t, \quad i = 1, 3$$

we can rewrite 2 as:

$$\frac{(\tilde{v}_0 + t)(\tilde{v}_2 + t) - (\tilde{v}_1 + t)(\tilde{v}_3 + t)}{(\tilde{v}_0 + t) + (\tilde{v}_2 + t) - (\tilde{v}_1 + t) - (\tilde{v}_3 + t)} > t \implies$$

$$\frac{\tilde{v}_0\tilde{v}_2 - \tilde{v}_1\tilde{v}_3 + t(\tilde{v}_0 + \tilde{v}_2 + \tilde{v}_1 + \tilde{v}_3) + t^2 - t^3}{\tilde{v}_0 + \tilde{v}_2 + \tilde{v}_1 + \tilde{v}_3 + 2t - 2t} > t \implies$$

$$\frac{\tilde{v}_0\tilde{v}_2 - \tilde{v}_1\tilde{v}_3}{\tilde{v}_0 + \tilde{v}_2 + \tilde{v}_1 + \tilde{v}_3} > t - t\frac{\tilde{v}_0 + \tilde{v}_2 + \tilde{v}_1 + \tilde{v}_3}{\tilde{v}_0 + \tilde{v}_2 + \tilde{v}_1 + \tilde{v}_3} = 0$$

(3)

Furthermore, since $\tilde{v}_i > 0 \quad i = 0, 1, 2, 3$ the term $\tilde{v}_0 + \tilde{v}_2 + \tilde{v}_1 + \tilde{v}_3$ is strictly positive, hence eq. 3 has solutions iff the following holds:

$$\tilde{v}_0\tilde{v}_2 - \tilde{v}_1\tilde{v}_3 > 0, \quad \tilde{v}_i > 0 \quad i = 0, 1, 2, 3$$

Doing these steps for all of the inequalities, the system can be rewritten as:

$$\tilde{v}_0\tilde{v}_2 - \tilde{v}_1\tilde{v}_3 < 0$$

(4)

$$\tilde{v}_2\tilde{v}_5 - \tilde{v}_1\tilde{v}_6 < 0$$

(5)

$$\tilde{v}_0\tilde{v}_5 - \tilde{v}_1\tilde{v}_4 < 0$$

(6)

$$\tilde{v}_2\tilde{v}_7 - \tilde{v}_3\tilde{v}_6 < 0$$

(7)

$$\tilde{v}_0\tilde{v}_2 - \tilde{v}_1\tilde{v}_3 > 0$$

(8)
From inequalities 6, 7, 8, and 9 we can easily find an absurd. In fact, from 8 and 6 we have:

\[
\frac{\hat{v}_1 \hat{v}_3}{\hat{v}_2} < \frac{\hat{v}_0}{\hat{v}_5} \Rightarrow \frac{\hat{v}_1 \hat{v}_4}{\hat{v}_5} < \frac{\hat{v}_4}{\hat{v}_5}
\]

and, using 7,

\[
\frac{\hat{v}_7}{\hat{v}_6} < \frac{\hat{v}_3}{\hat{v}_5} < \frac{\hat{v}_4}{\hat{v}_5}
\]

hence

\[
\hat{v}_2 \hat{v}_7 - \hat{v}_4 \hat{v}_6 < 0
\]

which contradicts inequality 9. Since we have found an absurd only using four out of six inequalities, we can exclude the four configurations in Table 11 and, for symmetry, all of the sixteen missing configurations of MC Case no. 13 (note that only 42 out of 46 in Table 10 are distinct in the face saddle points).

References


Figure 12: Enhancing reconstruction accuracy: different patches are extracted from the same cell, using increased accuracy and deeper refinement.

Figure 13: Recursive refinement process on the effe dataset; red faces are those generated in the current refinement step, and gray levels are associated to previous refinement steps.
Table 1: Time and complexity of the iso-surfaces fitted on the “Tube” and “F” datasets using the three different splitting points evaluation rules (with 10 the max. number of recursive subdivisions).

**“Tube” Dataset** (9x9x9, thres.=130.5, MC times=0.01 sec.)

<table>
<thead>
<tr>
<th>Accuracy (ε)</th>
<th>1/1000</th>
<th>1/2000</th>
<th>1/4000</th>
<th>1/5000</th>
<th>1/10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#faces</td>
<td>732</td>
<td>1,256</td>
<td>1,932</td>
<td>2,308</td>
<td>4,412</td>
</tr>
<tr>
<td>maxL</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>meanL</td>
<td>0.322</td>
<td>0.782</td>
<td>1.056</td>
<td>1.258</td>
<td>1.968</td>
</tr>
<tr>
<td>time</td>
<td>0.050</td>
<td>0.090</td>
<td>0.160</td>
<td>0.190</td>
<td>0.410</td>
</tr>
<tr>
<td>Rule A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#faces</td>
<td>732</td>
<td>1,256</td>
<td>1,996</td>
<td>2,420</td>
<td>4,460</td>
</tr>
<tr>
<td>maxL</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>meanL</td>
<td>0.322</td>
<td>0.782</td>
<td>1.185</td>
<td>1.484</td>
<td>2.064</td>
</tr>
<tr>
<td>time</td>
<td>0.070</td>
<td>0.130</td>
<td>0.220</td>
<td>0.270</td>
<td>0.550</td>
</tr>
<tr>
<td>Rule A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#faces</td>
<td>784</td>
<td>1,486</td>
<td>2,732</td>
<td>3,380</td>
<td>7,204</td>
</tr>
<tr>
<td>maxL</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>meanL</td>
<td>0.322</td>
<td>0.782</td>
<td>1.084</td>
<td>1.250</td>
<td>1.669</td>
</tr>
<tr>
<td>time</td>
<td>0.080</td>
<td>0.170</td>
<td>0.340</td>
<td>0.420</td>
<td>0.990</td>
</tr>
</tbody>
</table>

**“F” Dataset** (30x30x4, threshold=73.5, MC times=0.01 sec.)

<table>
<thead>
<tr>
<th>Accuracy (ε)</th>
<th>1/100</th>
<th>1/200</th>
<th>1/400</th>
<th>1/800</th>
<th>1/1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#faces</td>
<td>2,708</td>
<td>4,456</td>
<td>8,350</td>
<td>14,238</td>
<td>18,058</td>
</tr>
<tr>
<td>maxL</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>meanL</td>
<td>0.801</td>
<td>1.188</td>
<td>1.558</td>
<td>2.035</td>
<td>2.186</td>
</tr>
<tr>
<td>time</td>
<td>0.210</td>
<td>0.411</td>
<td>0.881</td>
<td>1.803</td>
<td>2.573</td>
</tr>
<tr>
<td>Rule A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#faces</td>
<td>2,726</td>
<td>4,638</td>
<td>8,814</td>
<td>14,450</td>
<td>18,350</td>
</tr>
<tr>
<td>maxL</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>meanL</td>
<td>0.805</td>
<td>1.201</td>
<td>1.579</td>
<td>2.040</td>
<td>2.194</td>
</tr>
<tr>
<td>time</td>
<td>0.300</td>
<td>0.570</td>
<td>1.142</td>
<td>2.273</td>
<td>3.095</td>
</tr>
<tr>
<td>Rule A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#faces</td>
<td>3,914</td>
<td>7,726</td>
<td>16,742</td>
<td>32,350</td>
<td>43,488</td>
</tr>
<tr>
<td>maxL</td>
<td>4</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>meanL</td>
<td>0.784</td>
<td>1.205</td>
<td>1.548</td>
<td>2.016</td>
<td>2.155</td>
</tr>
<tr>
<td>time</td>
<td>0.471</td>
<td>1.041</td>
<td>2.864</td>
<td>6.830</td>
<td>10.265</td>
</tr>
</tbody>
</table>
Table 2: Time and complexity of the iso-surfaces fitted on the “SOD” dataset with different settings for the approximation accuracy and the maximum number of recursive decompositions.

<table>
<thead>
<tr>
<th>SOD dataset</th>
<th>(11x14x14, thres.=50.5, MC times=0.01sec.)</th>
<th>accuracy</th>
<th>#faces</th>
<th>maxL</th>
<th>meanL</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1/100</td>
<td>1,141</td>
<td>1</td>
<td>0.492</td>
<td>0.050</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,449</td>
<td>1,449</td>
<td>2</td>
<td>0.674</td>
<td>0.090</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,581</td>
<td>1,581</td>
<td>3</td>
<td>0.748</td>
<td>0.110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,603</td>
<td>1,603</td>
<td>4</td>
<td>0.772</td>
<td>0.121</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/500</td>
<td>1,856</td>
<td>1</td>
<td>0.843</td>
<td>0.100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,953</td>
<td>3,953</td>
<td>2</td>
<td>1.516</td>
<td>0.240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,925</td>
<td>5,925</td>
<td>3</td>
<td>1.948</td>
<td>0.451</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6,855</td>
<td>6,855</td>
<td>4</td>
<td>2.138</td>
<td>0.631</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7,125</td>
<td>7,125</td>
<td>5</td>
<td>2.212</td>
<td>0.701</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/1000</td>
<td>2,074</td>
<td>1</td>
<td>0.902</td>
<td>0.130</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,364</td>
<td>5,364</td>
<td>2</td>
<td>1.689</td>
<td>0.381</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9,800</td>
<td>9,800</td>
<td>3</td>
<td>2.292</td>
<td>0.841</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13,080</td>
<td>13,080</td>
<td>4</td>
<td>2.552</td>
<td>1.372</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14,244</td>
<td>14,244</td>
<td>5</td>
<td>2.843</td>
<td>1.703</td>
</tr>
<tr>
<td>case no.</td>
<td>pattern</td>
<td>case no.</td>
<td>pattern</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>![Image]</td>
<td>1</td>
<td>![Image]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>![Image]</td>
<td>3</td>
<td>Table 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Table 5</td>
<td>5</td>
<td>![Image]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Table 6</td>
<td>7</td>
<td>Table 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>![Image]</td>
<td>9</td>
<td>![Image]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Table 8</td>
<td>11</td>
<td>![Image]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Table 9</td>
<td>13</td>
<td>Table 10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: The MC Exhaustive Look-Up Table. Ambiguous configurations are presented in separated Tables.

<table>
<thead>
<tr>
<th>pattern</th>
<th>saddle points</th>
<th>pattern</th>
<th>saddle points</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image]</td>
<td>× × × ⬝ × ×</td>
<td>![Image]</td>
<td>× × × ⬝ × ×</td>
</tr>
</tbody>
</table>

Table 4: MC case no. 3 leads to 2 different geometric patterns.
Table 5: MC case no. 4 leads to 2 different geometric patterns.

<table>
<thead>
<tr>
<th>pattern</th>
<th>saddle points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 3 4 5 b</td>
</tr>
</tbody>
</table>

Table 6: MC case no. 6 leads to 4 different geometric patterns.

<table>
<thead>
<tr>
<th>pattern</th>
<th>saddle points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 3 4 5 b</td>
</tr>
</tbody>
</table>

Table 7: MC case no. 7 leads to 9 different geometric patterns.

Table 8: MC case no. 10 leads to 6 different geometric patterns.

27
Table 9: MC case no. 12 leads to 6 different geometric patterns.

Figure 14: Two meshes extracted from a section of the SOD dataset, using standard MC on the left and the PreciseMC method on the right (max recursion level = 4).
Table 10: MC case no. 13 leads to 46 different geometric patterns.
Table 11: MC case no. 13: 4 of the 16 missing geometric patterns.

Figure 15: Comparing iso-surfaces extracted (SOD dataset) with the Metro tool; a section of the MC mesh (left image) is colored according to the distance from the corresponding mesh section extracted with PreciseMC (right image).

Figure 16: Another mesh extracted from a section of the SOD dataset, using standard MC on the left and the PreciseMC method on the right.
Figure 17: Examples of the different mesh patches which can be produced from the same configuration of the original MC binary code, but different data values on the vertices; the six topologically different meshes are shown under two different view direction (front view and side view).