Topological reconstruction of oil reservoirs from seismic surfaces

GEOVAN TAVARES[1], ROGERIO SANTOS[2], HÉLIO LOPES[1], THOMAS LEWINER[1,3] AND ANTÔNIO WILSON VIEIRA[1,4]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Petrobras – Brazilian Oil Company — Rio de Janeiro — Brazil
[3] Géométrica Project — INRIA – Sophia Antipolis — France
[4] CCET — Universidade de Montes Claros — Brazil
{tavares, awilson, lopes, tavares, tomlew, awilson}@mat.puc--rio.br.
r_santos@petrobras.com.br.

Abstract. This paper describes the main aspects of Project Geosis. It is an ongoing three year project between the Brazilian oil company Petrobras and the Pontifical Catholic University of Rio de Janeiro, Brazil. Its main objective is to extract information from seismic data through the use of geometric and topological modeling, as well as scientific visualization.
Keywords: ***Topological surface reconstruction. Surface simplification.***

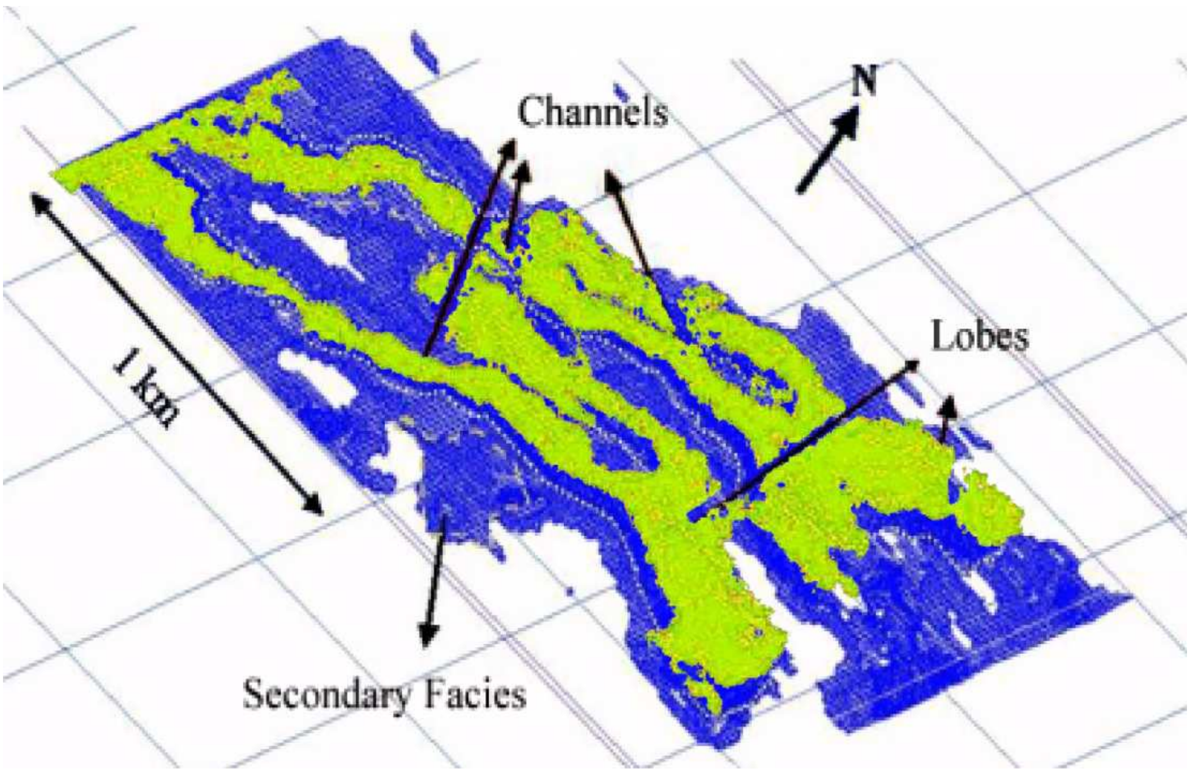

Figure 1: *3D topological reconstruction, from seismic data, of a deep water oil reservoir (2000 meters).*

1 Introduction

The characterization of oil reservoirs has as one of its components the study of their geometry and topology. While the geometry deals with spatial distribution of points and its associated properties, e.g. permeability and porosity, the topology handles its connectivity. To determine the topology of the reservoir is then an essential ingredient for its characterization. One of the main characteristics of seismic processing is the generation of large data sets. Computationally intensive powerful techniques are needed to extract mean-

Preprint MAT. 22/03, communicated on May 7^{th}, 2003 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of the International Association for Mathematical Geology 2003.

ingful information from these data. The main objective of the paper is to indicate how to reconstruct geological objects (channels, lobes, ...) directly from seismic data. The data is piled up in offset groups in such a way as to get information from the seismic aiming at the geological modeling.

This paper focuses on: 1. Topological surface reconstruction from seismic data and its corresponding visualization; 2. Surface simplification in order to deal with large data sets. Techniques for surfaces simplification for handling large data sets will be discussed. Several techniques have been implemented and will be presented.

This paper describes the main aspects of Project Geosis. It is an ongoing three year project between the Brazilian oil company Petrobras and the Pontifical Catholic University of Rio de Janeiro, Brazil. Its main objective is to extract information from seismic data through the use of geometric and topological modeling, as well scientific visualization.

2 Isosurface extraction

(a) Overview

The seismic data obtained by physical measures can be interpreted, after the signal processing step, as a sampling of a continuous function $f : (x, y, z) \in \mathbb{R}^3 \mapsto f(x, y, z) \in \mathbb{R}$ at some points in the space $\{(x_i, y_j, z_k); i = 1..m, j = 1..n, k = 1..p\}$. The function f describe a geophysical property, such as the permeability, the porosity, ... The reservoir corresponds to the portion of space R where the value of the property is included in a certain range $[v, w] : R = f^{-1}([v, w])$.

Any method that computes this volume R needs to extrapolate from the sampled data. For example, we could induce the function f from the sampled data, and then compute analytically the pre–image R. This extrapolation would involve a specific and complete geological model of the terrain. However, this method is computationally expensive, renders interpreted data and does not guarantee the topology of the reservoir in general. We will use a different but classical strategy, and enhance its robustness.

We will compute the reservoir surface directly from the sampled data. We aim at controlling and minimizing the artifacts induced by the extrapolation. In particular, we are concerned with the topology of the resulting surface, i.e. preserving the connections between or inside the reservoirs. Moreover, in order to improve the quality of the result, we will include in our process the global geological information we already know, such as the data representation described in the next section.

(b) Incorporation of geological models

We will suppose from now on that the points where the property f is estimated are arranged in a cuberille manner. This means that we are able to connect those points in order to form adjacent convex solid polytopes with 8 faces. Actually, if we do not look for the boundary region, there are several ways of connecting the measure points into a cuberille grid. For example on an infinite cubic grid, we could connect the cube with diagonal strips.

Ideally, the cubes should follow the geological structure of the reservoir. This can be partially achieved by a preprocessing step, identifying the main curves of the geological points. In particular, submarine reservoir are often locates in fracture zone, and the reservoir itself often follows the fracture lines. During this preprocessing step, we connect the measure points in order to form a cuberille grid which follows those main curves.

This preprocessing offers different advantages. First, the resulting reservoir surface is more accurate: the isosurface extracted from the measures f is tiled within each cube. If an edge of the grid has one vertex inside the reservoir and one outside, then the isosurface will intersect this edge. If the grid were simply parallel to the axes, those edges crossed by the surface can be arranged in a very steep manner, leading to less accurate surface. Second, we can rectify the grid, i.e. moving the grid points to form a cubic grid parallel to the axis, without modifying the value of the property f at that point. Then, we can work in the original grid, which gives a realistic view of the reservoir, or in the rectified grid, which often gives a clearer view of the reservoir.

(c) Missing data

The main problem we encountered with the seismic data is its incompleteness: some of the grid points (x_i, y_j, z_k) do not have an associated property value $f(x_i, y_j, z_k)$. Those points can be isolated, or form entire volumes inside the grid. We implemented three strategies to handle this deficiency, which corresponds to different quality/reliability and quality/computational costs trade–off:

– No interpolation: the grid point is discarded, and none of the triangles of the final surface will intersect a cube containing a discarded point. This ensures a more reliable output, but leads to many holes in the surface.

– Linear interpolation: the value of a missing grid point is computed as the barycenter of its nearest valid points.

– Radial–basis interpolation: all the valid point of the grid contributes to the missing value proportionally to their distance. This gives nice results, but induces a model of the data which is not always accurate. Radial–basis methods have been extensively studied, and would offer many possibilities of including accurate geo-physics models [1].

(d) Surface reconstruction

We will use an extension of the Marching–Cubes' algorithm [8] to extract the surface of the reservoir from the preprocessed seismic data. The Marching Cubes method produces a triangular mesh of the preimage $g^{-1}(0)$, given by samples over a cuberille grid. To convert the test $f(x_i, y_j, z_k) \in [v, w]$ into $f(x_i, y_j, z_k) \geq 0$, we will test a grid point with $g = (f - v) \cdot (w - f)$. The original method

The corresponding work was published in the proceedings of the International Association for Mathematical Geology 2003.

sweeps the grid, and tiles the surface cube per cube. Each point of the grid is classified into positive and negative vertices. Thus, there are $2^8 = 256$ possible configurations of a cube. The usual implementation stores those 256 in a lookup table that encodes the tiling of the cube in each case (see Figure 7).

However, the original implementation can lead to cracks (see Figure 2) and could not respect the topology of the tri–linear interpolation of f. We thus need to add further test on ambiguous cubes. This distinction has been done by Chernyaev in its Marching Cubes' 33 algorithm [2], and lead to the completed lookup table 8 of 730 cases.

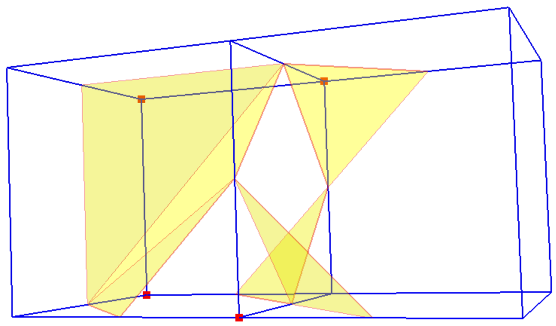

Figure 2: *A crack occurring on an ambiguous face in-between cases 12 and 3 of table 7.*

This enhanced algorithm has been implemented in [7]. However, their implementation needs a re–computation when tiling an ambiguous cube, which is computationally expensive. Their computation allows a more accurate geometry inside each cube by the addition of extra vertices inside some of the cubes. For our purpose, we needed a faster algorithm which would not produce too many vertices. Therefore, we implemented the 730 cases of Chernyaev's lookup table 8 [5]. The tiling of a cube is then done by only a consultation of the lookup table.

The Marching Cubes' algorithm offer many extensions, in particular in previsualization, view–dependant rendering, hardware acceleration . . . Some of those enhancements need an expensive pre-computation ($n \cdot \log(n)$ for topology pre–visualization for example), other pre–suppose parts of the results (the ability of having one seed per connected component in order to avoid parsing all cubes). All those extensions are possible to implement with our enhancements. We will not introduce them here as they are not always adapted to our problem.

(e) Visualization of seismic surfaces

Given a seismic cube, parallel slices are extracted and piled up in such a way as to allow the 3D reconstruction of the reservoir using the amplitude of the seismic wave in a given interval. In the figures below domains on parallel slices were chosen. In Figure 3 and Figure 1 isosurfaces were extracted and geological structures like channels and lobes, were identified in a deep water reservoir.

3 Surface Simplification

The surface reconstruction discussed in the previous section will generate a triangular mesh that represents the boundary of the oil reservoir. According to the sampled data resolution, the data set for handling such mesh may be extremely large. Than, in order to represent the same surface with lower data cost, we have developed a simplification process to obtain a simplified mesh with lower number of triangles that preserves the topology type of the original surface and uses an *Error Metric* in order to control its geometrical proprieties, as volume, by minimizing the geometrical distortion from the original surface.

Our simplification algorithm is based on local topology preserving operators so that the simplified mesh has the same topology type to the original mesh. The main objective of these operators is removing elements as vertices, edges and faces from the mesh without changing its topologic type. Each element to be removed from the mesh is chosen as the one with lower geometrical cost to the original surface. This cost is computed using the *Quadric Error Metric* by Garland and Heckbert [4]. We will point out how does this operators and metric works before discussing the simplification algorithm.

(a) Mesh simplification operators

The operators we will present were implemented using a data structure called *Corner-Table* [6] that is a compact version of the *Half-Edge* representation of triangular meshes. Using this data structure we implemented several local operators and techniques for mesh simplifications [9] as follow.

Edge–Flip: This operation does not remove any element from the mesh and consists in transforming a *two-face* cluster into another *two-face* cluster by swapping its common edge.

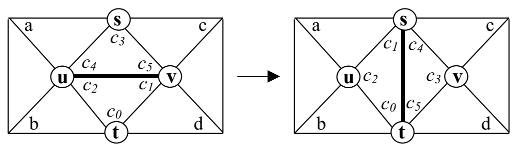

Figure 4: *Edge–Flip.*

Let the edge $e = (u, v)$ and s and t the two vertices opposed to e (see Figure 4). The *Edge-Flip* operation will replace e by (s, t), and replace the 2 triangles incident to e by (u, s, t) and (v, t, s).

Edge–Collapse: This operator consists on removing an edge $e = (u, v)$ from the mesh, identifying its vertices to a unique vertex w. From a combinatorial viewpoint, this operator will remove 1 vertex, 3 edges and 2 faces from original

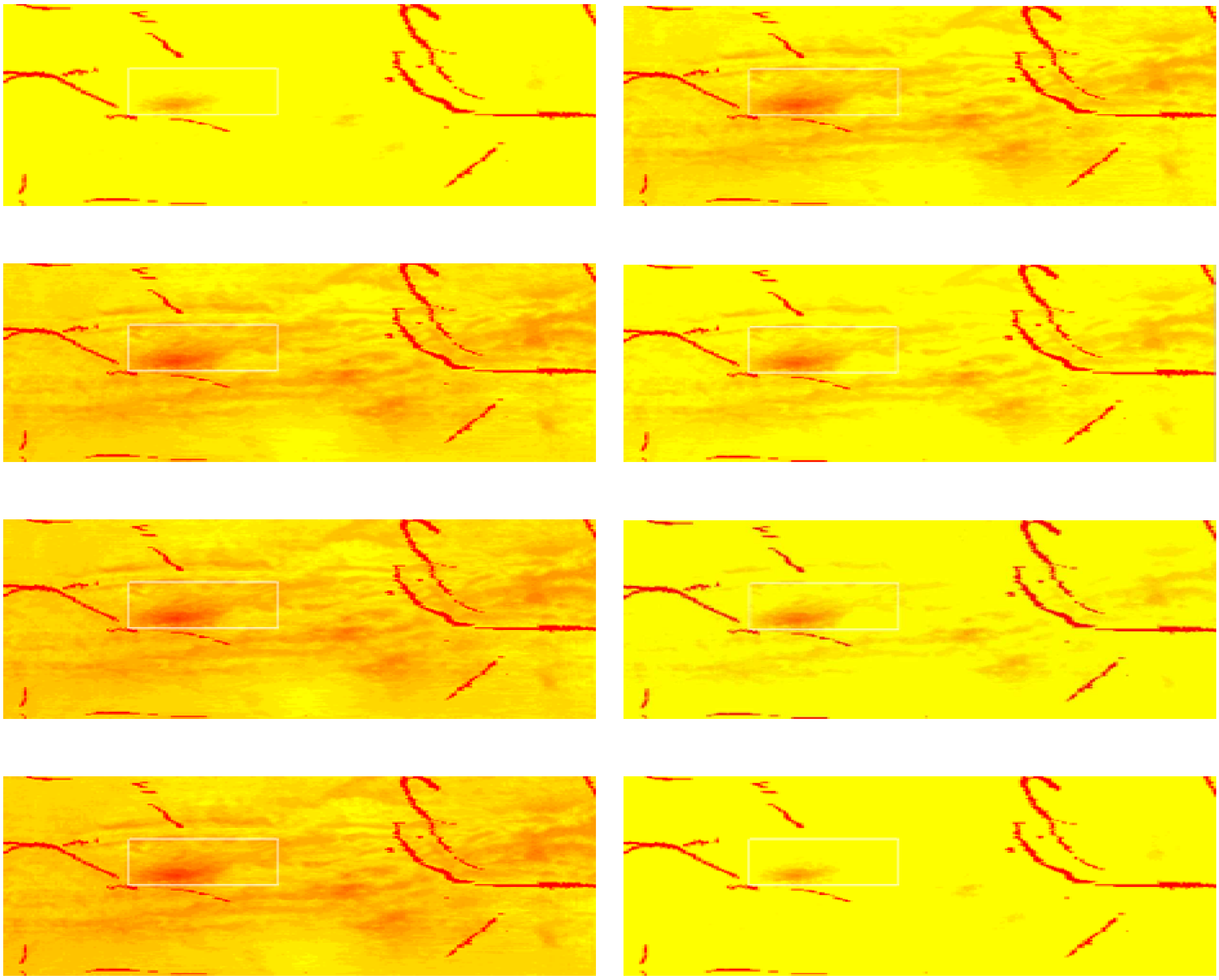

Figure 3: *Slices of seismic data of a deep water oil reservoir (2000 meters).*

mesh, without changing its Euler characteristic. From a geometric viewpoint, the new position of the vertex w can be computed with the geometry around u and v.

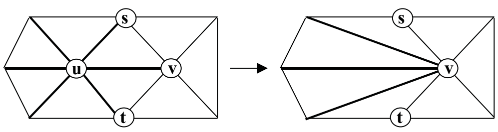

Figure 5: *Edge–Collapse.*

Let s and t be the vertices opposite to $e = (u, v)$, which is the edge to be collapsed (see Figure 5). By collapsing the edge e we will remove the faces (u, v, s) and (u, t, v) from the mesh. The topological consistency of this operations is guaranteed by the following *link condition* [3].

(Link Condition) Let S be a combinatorial 2–manifold. The contraction of an edge $e = (u, v) \in S$ preserves the

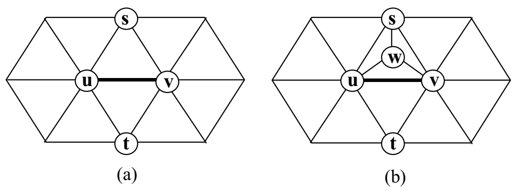

Figure 6: *Link Condition.*

topology of S if and only if $\mathrm{link}(u) \cap \mathrm{link}(v) = \mathrm{link}(e)$.

Figure 6 shows, in (a), an edge (u, v) whose contraction is topology preserving while in (b) it is not.

Other operators as *edge-weld*, *face-collapse* and *star-collapse* were implemented as a combination of these two first operators.

The corresponding work was published in the proceedings of the International Association for Mathematical Geology 2003.

(b) Geometric cost evaluation

For evaluation of the geometrical cost of performing simplification operators we used the *Quadric Error Metric* from Garland and Heckbert [4] as fallow.

Consider the problem of finding the distance of a point w to the plane p_f support of a face $f = (v_1, v_2, v_3)$ whose unitary normal vector is $\vec{n}$.

Given a point $p = (p_x, p_y, p_z) \in p_f$, the distance d from the origin to the plane p_f is

$$d = p^t.n = p_x n_x + p_y n_y + p_z n_z$$

Then, the distance from a point $w \in \mathbb{R}^3$ to p_f, is the distance d_w from the origin to the plane parallel to p_f touching w minus d.

$$d_w = w^t.n - d = w_x n_x + w_y n_y + w_z n_z - d$$

Making $w = (w_x, w_y, w_z, 1)$ and $\vec{n} = (n_x, n_y, n_z, -d)$, d_w can be computed as a dot product in dimension 4.

$$d_w = w^t.n = w_x n_x + w_y n_y + w_z n_z + 1(-d)$$

The quadratic distance $d(w)$ from a point $w \in \mathbb{R}^3$ to the plane p_f is, than, given by

$$d(w) = \left(w^t.n\right)^2 = \left(w^t.n\right)\left(n^t.w\right) = w^t\left(n.n^t\right)w$$

The product $n.n^t$ above origins a 4x4 symmetric matrix, which Garland and Heckbert called the fundamental quadric Q_f associated to the face f.

$$Q_f = n.n^t = \begin{pmatrix} n_x^2 & n_x n_y & n_x n_z & n_x d \\ n_x n_y & n_y^2 & n_y n_z & n_y d \\ n_x n_z & n_y n_z & n_z^2 & n_z d \\ n_x d & n_y d & n_z d & d^2 \end{pmatrix}$$

Let v a vertex on a mesh M, and $f_i \in \mathrm{star}(v)$ the faces incidents to v. The distance $d(w, v)$ from a point w to the vertex v is given, by the *Quadric Error Metric*, as the sum of the quadratic distances $d_i(w)$ from w to the plane support of each face f_i. Using the quadrics Q_i associated with each face f_i we have

$$d(w, v) = \sum w^t (Q_i) w = w^t \left(\sum Q_i\right) w$$

The sum $\sum Q_i$ origins a new 4x4 matrix called the fundamental quadric associated with the vertex v, which is noted as Q_v.

On performing the *edge-collapse* operation for an edge $e = (u, v)$ to a resultant vertex w we have the cost C (geometric distortion) given by the sum of the distances $d(w, v)$ and $d(w, u)$

```
c=0;
assign quadrics;
while (c < C) do
  for (i = 1 to 8) do
    e_i =random edge
    if Cost(e_i) < Cost(e) then e = e_i
  c+=Cost(e)
  perform EdgeCollapse(e)
```

Table 1: ***Algorithm 1****: Simplify*(M, C)

$$\begin{aligned} C &= d(w, v) + d(w, u) \\ &= w^t Q_v w + w^t Q_u w \\ &= w^t (Q_v + Q_u) w \end{aligned}$$

The computation of C, as defined above, for each edge candidate to the *edge-collapse* allow us to choose the edge that minimizes the geometrical cost.

(c) Simplification Algorithm

The first step of our simplification algorithm is to assign Q_v, as previously defined, for all $v \in M$, where M is the mesh to be simplified. For each edge $e = (u, v) \in M$ we choose a vertex w in order to minimize the cost $C_e = d(w, v) + d(w, u)$. The cost C_e is than inserted into a priority queue so that we can perform the *edge-collapse* operator in a increasing order of cost.

In the Garland and Heckbert algorithm, the cost for edges incidents to the resultant vertex w is recomputed and its priority in the queue updated after each simplification operation. This cost re-computation and queue updating leads to an increasing amount of time processing that we avoid in our implementation by using a probabilistic optimization strategy [10] that does not use a priority queue.

The sum of cost introduced for each *edge-collapse* operation is than used to control the simplification process till the desired level. The algorithm below obtain a simplification for the mesh M with geometric cost less than C.

The algorithm above naturally constructs an hierarchy of meshes $\left(M^0, M^1, .., M^n\right)$ with decreasing number of elements (vertices, edges and faces). The surface M^0 is the original surface M, and each surface M^j, $j = 1..n$, corresponds to the level j of detail of M. Such levels of detail M^j are than obtained by the parallel application of such algorithm. Each level of detail accumulates a geometrical cost that estimates its fairness to the original mesh.

Our data structure uses only two arrays of integers for storing the mesh connectivity and an array of floats for storing the geometry (coordinates) of vertices. At each level of detail we have a new connectivity among a subset of the same initial set of vertices so that the same array of vertices is used in all levels of detail. Hence, for saving n levels of detail form a mesh we just need store $2n$ arrays of integers for representing the connectivity at each level. This connectivity can also be encoded in a compressed manner (with less than 2 bits per triangle) using the Edgebreaker compression scheme [6].

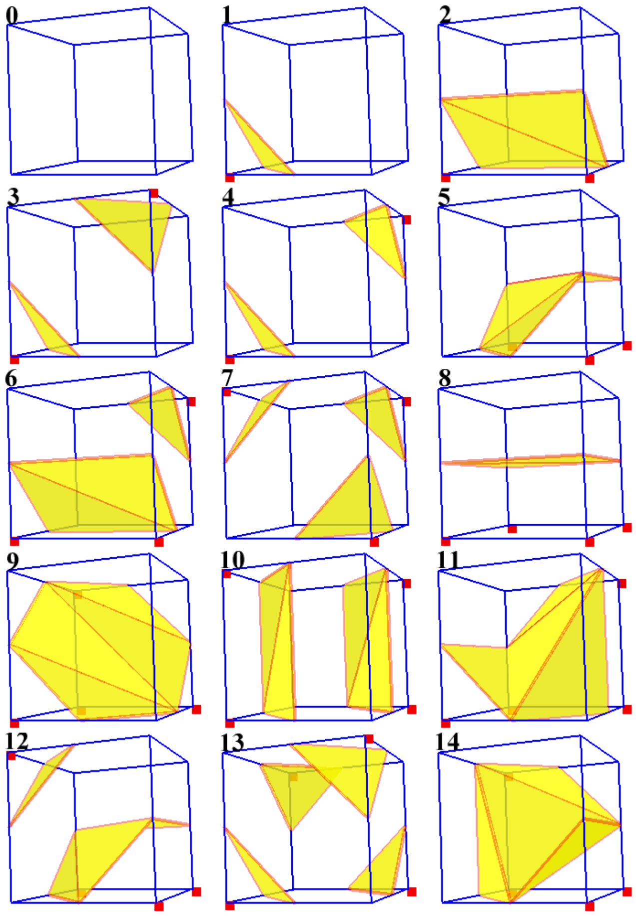

Figure 7: *Original Marching Cubes lookup table.*

References

[1] R. K. Beatson and J. B. Cherrie. Reconstruction and representation of 3D objects with radial basis functions. In *Siggraph*, pages 67–76. ACM, 2001.

[2] E. V. Chernyaev. Marching Cubes 33: construction of topologically correct isosurfaces. Technical Report CERN CN 95–17, CERN, 1995.

[3] H. Edelsbrunner. *Geometry and topology for mesh generation*. Cambridge Monographs on Applied and Computational Mathematics, 2002.

[4] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Computers & Graphics*, 31:209–216, 1997.

[5] T. Lewiner, H. Lopes, A. W. Vieira and G. Tavares. Efficient implementation of Marching Cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[6] H. Lopes, J. Rossignac, A. Safonova, A. Szymczak and G. Tavares. Edgebreaker: a simple compression for surfaces with handles. In C. Hoffman and W. Bronsvort, editors, *Solid Modeling and Applications*, pages 289–296, Saarbrücken, Germany, 2002. ACM.

[7] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the Marching Cubes algorithms for isosurfacing. *Transactions on Visualization and Computer Graphics*, 9:16–29, 2003.

[8] W. E. Lorensen and H. E. Cline. Marching Cubes: a high resolution 3D surface construction algorithm. In *Siggraph*, volume 21, pages 163–169. ACM, 1987.

[9] A. W. Vieira, L. Velho, H. Lopes, G. Tavares and T. Lewiner. Fast stellar mesh simplification. In *Sibgrapi*, pages 27–34. IEEE, 2003.

[10] J. Wu and L. Kobbelt. Fast mesh decimation by multiple–choice techniques. In *Vision, Modeling and Visualization*, pages 241–248. IOS Press, 2002.

The corresponding work was published in the proceedings of the International Association for Mathematical Geology 2003.

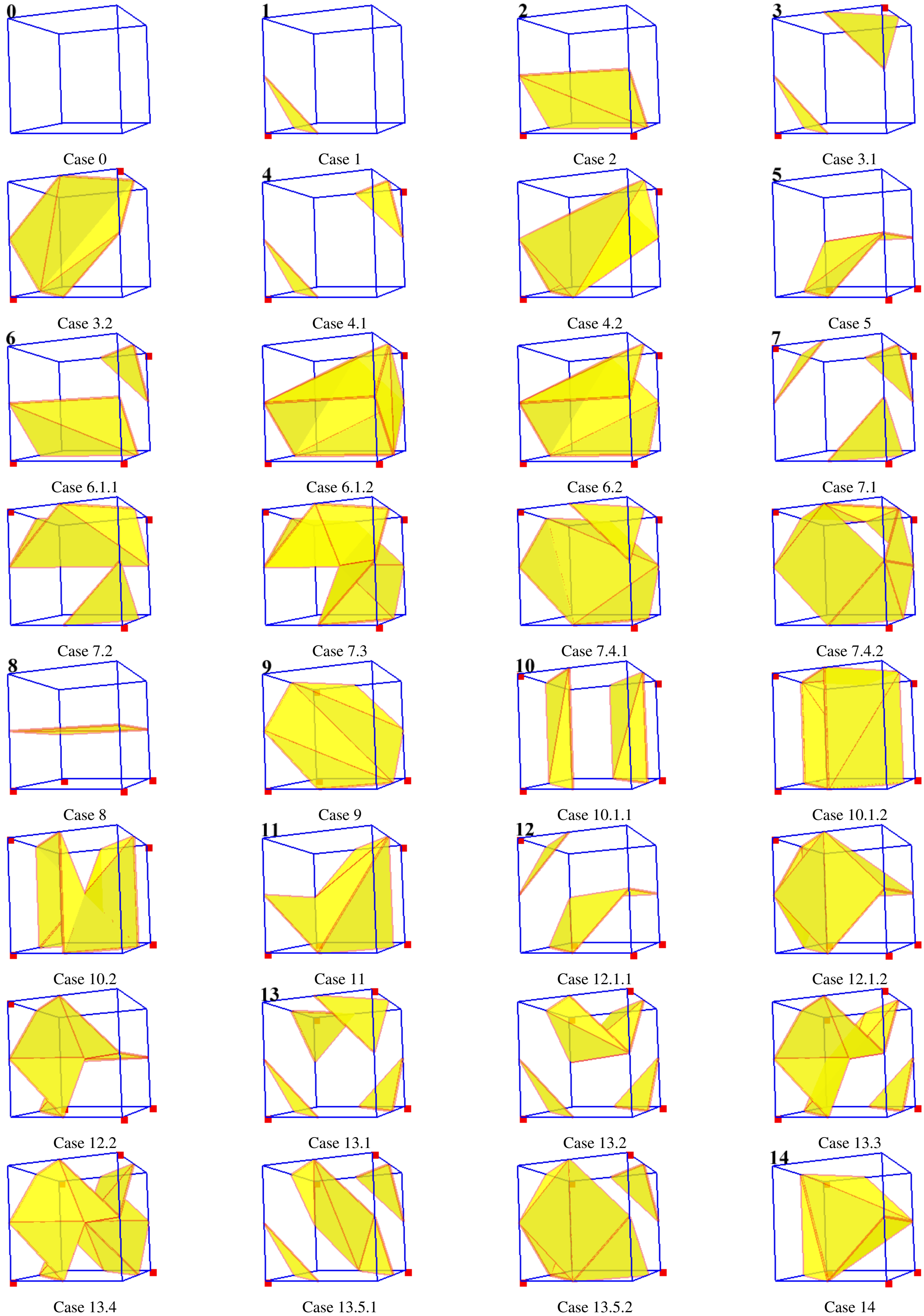

Figure 8: *Chernyaev's lookup table.*