Simplicial isosurface compression

THOMAS LEWINER[1,2], HÉLIO LOPES[1], LUIZ VELHO[3] AND VINÍCIUS MELLO[3]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Géométrica Project — INRIA – Sophia Antipolis — France
[3] Visgraf Project — IMPA — Rio de Janeiro — Brazil
{tomlew, lopes}@mat.puc--rio.br. {lvelho, vinicius}@visgraf.impa.br.

Abstract. In this work, we introduce a new algorithm for direct and progressive encoding of isosurfaces extracted from volumetric data. A binary multi–triangulation is used to represent and adapt the 3D scalar grid. This simplicial scheme provides geometrical and topological control on the decoded isosurface. The resulting algorithm is an efficient and flexible isosurface compression scheme.
Keywords: ***Isosurfaces. Data Compression. Simplicial Methods. Progressive Transmission.***

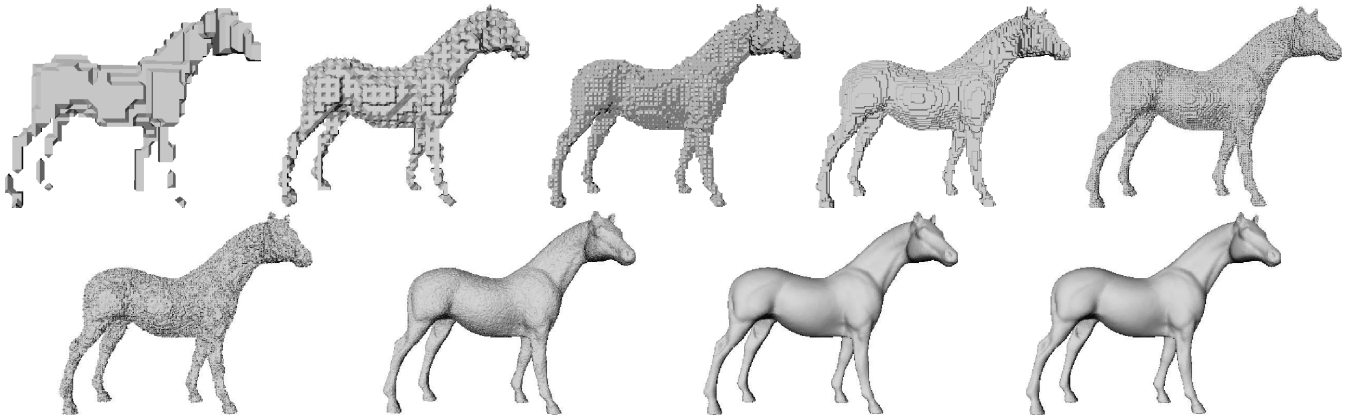


Figure 1: *Decompression of the Horse model as a* 257^3 *regular sampled data. The first images corresponds to the binary multi–triangulation refinement, followed by the geometric refinements.*

1 Introduction

With technological advances, 3D images become more accurate, and thus result in larger meshes to handle. The most common objects produced by 3D data acquisition and simulation are isosurfaces [2]. For example, medical scan techniques such as Computerized Axial Tomography and Magnetic Resonance Imaging measure physical quantities sampled on a semi–regular 3D grid. In addition, Scientific simulations often resolve Partial Differential Equations through level set methods [15] which also result in sampled functions over a 3D grid.

Techniques to manipulate isosurfaces encompass structured extraction such as Marching Cubes [10, 6], simplicial extraction [19], smoothing [16], visualization [12] and others. However, we will focus here on compression (cf Figure 1).

Preprint MAT. 11/04, communicated on January 21^{st}, 2004 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of Vision, Modelling, and Visualization 2004, pp. 299–306. IOS Press, 2004.

Problem statement. In most of isosurface applications, the visual result is only a surface extracted from a 3D grid, which corresponds to the interpolation of the sampled function around a specific *iso*value. For example, the cortex corresponds to only a specific x–ray scintillation inside the scan of the whole head. In that case, when looking only at this cortex, most of the 3D grid is unused, as the reconstructed cortex is contained only in a limited part of the head. This observation motivated us to look for specific techniques to compress an isosurface by encoding only the portion of the 3D grid necessary to reconstruct it.

The polygonized isosurface could be compressed directly using a standard mesh compression method [18, 13, 5]. However, remeshing techniques that enhance the efficiency of the compression algorithm [7] are difficult to apply for complex isosurface. Moreover, they do not take advantage of the specificity of the isosurfaces, and thus achieve lower compression ratio (typically 8–16 bits per vertex), whereas specific techniques such as [17, 8] can achieve less than one bit per vertex.

Contributions. In this paper we introduce a new method for compressing of isosurfaces based on a hierarchical simplicial decomposition of volume data. Our method improves upon the state–of–the–art in many aspects:

We improve the rate/distortion ratio by encoding the isosurface preserving its local adjacencies, creating more correlations for the final arithmetic coder and improving the prediction.

We provide a very flexible scheme allowing non–regular sampled data and giving easy control on the topology and/or the geometric features of the decoded isosurface.

The progressive version of our algorithm allows smooth progression of the encoding and alternated geometry/connectivity–improved resolution levels.

Our algorithm allows usage of already compressed isosurfaces to encode close–by ones on the same 3D grid.

Our algorithm produces rate/distortion curves very close to octree–based compression when used in a similar way, but offers other ways of encoding the isosurface.

However, our technique requires more memory than octree–based methods at execution time.

2 Related work

Isosurface compression state–of–the–art includes different techniques based mainly on octrees for the encoding of the isosurface localization and connectivity. The isosurface geometry is generally encoded as a displacement, and then quantized. All of them are restricted to regular sampled data and differs mainly in the compression strategy for the octree. All of those which encode the geometry do it as the last step of the compression.

[14] encodes the voxels containing the isosurface by their position index inside the octree. Then the geometry of the isosurface vertices is encoded as a vertex displacement along one of 6 predefined axis.

[21] encodes only the information used by the Marching Cubes method for polygonizing the isosurface, i.e., the indices of each edge of the grid crossing the isosurface, and the vertex displacement along those edges. The isosurface connectivity is encoded by the cube class in the original 256–lookup table of the Marching Cubes algorithm.

[17] encodes directly the 3D image through an arithmetic encoder using context in all 3 dimensions of the image. The geometry is coded as a displacement with predictor. This technique improves its efficiency when encoding of the isosurface corresponding to a second isovalue, as what remains to encode is mainly the geometry. This feature can be very useful in medical applications, where the contrast of measure planes varies inside the grid.

[3] encodes the voxels containing the isosurface as a global octree and encodes the sign of each grid point that appears in the octree. This allows a progressive transmission, which ends when the vowel size guarantees a small enough geometric distortion. There is no specific geometry encoding.

[8] encodes the connectivity in a similar way, improving the compression with an arithmetic encoder acting on the whole grid (instead of plane separated encoding for [3]). When the octree progression ends, the geometry can be further refined by quantizing the isosurface vertices displacement. To maintain a small amount of vertices to encode, they use the Dual Contouring method [6].

3 Overview

Paper outline. We chose a simplicial representation of the 3D data: the Binary Multi–Triangulation (BMT) [11] (cf section 4 *Adaptive Simplicial Decompositions*). This structure can represent regularly sampled data with the same amount of grid points as the classical octree representation, but it can also adapt to non–regular data. It also provides simple controls on the isosurface topology and geometry that can be used for adaptive encoding of the surface. To enhance the rate/distortion curves and separate our results from the initial 3D data, our encoding strategy accompanies the isosurface, maintaining the neighbourhood relation during the compression (cf section 5 *Method and design*). This enhances the performance of the arithmetic coder used for the final compression.

Definitions. The *3D scalar data* is given as a tetrahedral mesh, where each vertex or *sample point* is associated with a sample of the scalar function called its *isovalue*. In particular, when the sample points are regularly spaced on a 3D grid, this mesh can be automatically generated by the subdivision of a cube.

Assuming that we want to obtain the isosurface corresponding to the isovalue α, we say that a vertex of the 3D data is qualified of *positive* if its isovalue is greater than α, and of *negative* otherwise. An edge of the tetrahedral mesh is *crossing* when it has one of its vertices if positive, and the other one is negative. A triangle or a tetrahedron is crossing when it contains a crossing edge. The *tubular neighbourhood* of the isosurface is the set of all the crossing tetrahedrons.

The isosurface is polygonized by computing a triangle or quadrangle for each crossing tetrahedron, according to the look–up table of [19].

4 Adaptive Simplicial Decompositions

The multiresolution structure used in our method is a Regular Binary Multi–Triangulation (RBMT) [11]. This structure is constructed using Stellar operators on edges and has very good adaptation properties. Using this structure we are able to compute adaptive simplicial decompositions of isosurface embeddings defined by a scalar field. The resulting decomposition is a hierarchy of conforming tetrahedral meshes. In this section we will give a brief description of the Stellar operators on edges and the binary multi–triangulations.

The corresponding work was published in the proceedings of Vision, Modelling, and Visualization 2004, pp. 299–306. IOS Press, 2004.

(a) Stellar Operators

Stellar theory studies the equivalencies between simplicial complexes and defines topological operators that change a manifold structure while maintaining the integrity of its combinatorial description [9].

The *stellar subdivision* operation inserts a vertex into an r–simplex of an n–dimensional simplicial complex. This type of operation is also called *split*. Its inverse is called *weld*.

One important result from Stellar theory states that stellar operators *on edges* are sufficient to map any two equivalent combinatorial manifolds [1]. Edge–split and Edge–weld will be the basic operators to construct Binary Multi–Triangulations.

(b) Binary Multi–Triangulation

The Binary Multi–Triangulation (BMT) is a multiresolution structure based on stellar subdivision on edges. Whenever a stellar subdivision happens on an edge e, all simplices containing e are split in two. Accordingly, a sequence of stellar subdivision induces a binary tree structure in the simplices; and binary trees often leads to simpler algorithms.

In order to define a BMT, we need some preliminary definitions. We follow closely the definitions of [4]. A *partially ordered set* (poset) $(C, <)$ is a set C with an antisymmetric and transitive relation $<$ defined on its elements. Given $c, c' \in C$, notation $c \prec c'$ means $c < c'$ and there in no $c'' \in C$ such that $c < c'' < c'$. An element $c \in C$, such that for all $c' \in C$, $c \leq c'$, is called a *minimal* element in C. If there is a unique minimal element $c \in C$, then c is called the *minimum* of C. Analogously are defined *maximal* and *maximum* elements.

Definition 1 *A* binary multi–triangulation *is a poset $(\mathcal{T}, <)$, where $\mathcal{T}$ is a finite set of combinatorial k–manifolds (named* triangulations*) and the order $<$ satisfies:*

1. *For all $T, T' \in \mathcal{T}$: $T \prec T'$ if, and only if, T' is obtained from T by splitting one edge of T, called a* refinement edge *of T.*

2. *There is minimum and maximum k–manifolds in $\mathcal{T}$, called* base triangulation *and* full triangulation*, respectively.*

Every two triangulations in $\mathcal{T}$ are stellar equivalent. In fact, a BMT is a *lattice* and thus can be thought of as a directed acyclic graph (DAG), with one source (T^0) and one drain (T^N) whose arrows corresponds to stellar subdivisions of refinement edges. From an algorithmic perspective the key idea is to use the above mentioned binary tree structure in the simplices to encode the DAG.

(c) Adaptation Properties and Regularity

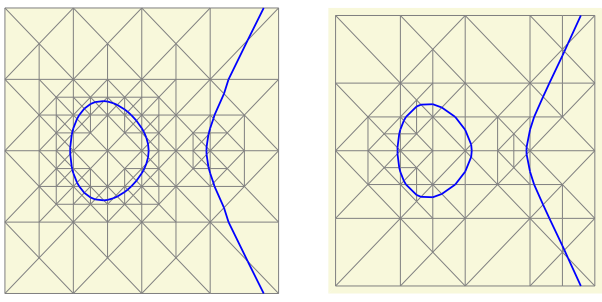
The adaptability of the BMT comes from the possibility to refine and coarsify the simplicial complex locally and, at the same time, maintain the dependencies inside the partial order relation.

Non–local transitions. The stellar operators *split* and *weld* implement just "local" transitions in the DAG, that is, if T and T' are the triangulations before and after split be performed, respectively, then $T \prec T'$. In our application, we want to impose regularity to the multiresolution structure, i.e. to ensure that adjacent triangles differ at most by one resolution level. To do so, it is necessary to make non–local transitions in the DAG, i.e. to propagate the local stellar operations to neighbouring elements. This corresponds to groups of transition in the DAG.

For example, Figure 2 illustrates a sequence of refinements are used to adapt the triangulation to the blue line. In order to preserve gradual transition between resolution levels, local refinements around the blue line propagates inside the mesh (in this example, far away from the blue line), as what happens to the bottom left part when we move from T^3 to T^8.

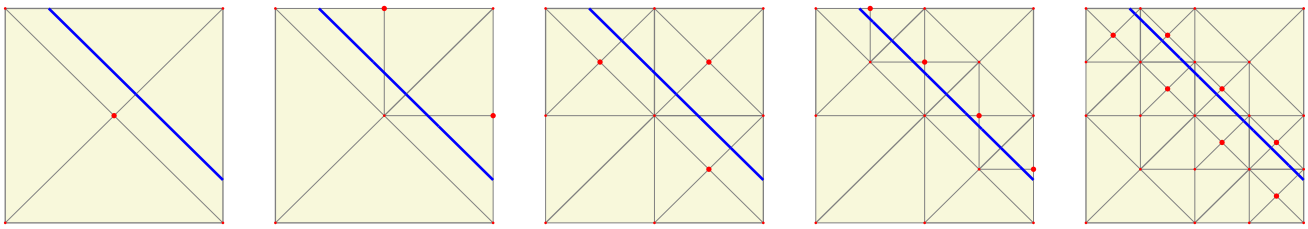
Regular BMT. With this condition, the resulting structure is called a *regular* binary multi–triangulation (RBMT). The RBMT uses non–local transitions in such a way to enforce that adjacent simplices with higher dimension differ at most by one resolution level. This type of structure is also called a restricted hierarchy [20]. The non–local transitions are implemented by restricted refinement and simplification procedures that use the split and weld operators, respectively.

Geometry control. Adaptive decomposition is achieved by performing restricted refinement and simplification based on some adaptation criteria. In the case of isosurface representation the adaptation criteria aims at refining around a tubular neighbourhood of the isocurve. For example on Figure 2, the RBMT is created by non–local split of each edge crossing the blue line. The decision to split a crossing refinement edge can depend on the local geometry of the curve, for example on the curvature as on Figure 3.



(a) curvature (b) distortion

Figure 3: *A RBMT adapted according to (a) the curvature and (b) the distortion.*



(a) T^1 (b) T^3 (c) T^8 (d) T^{12} (e) T^{19}

Figure 2: *RBMT adapted to the blue line: refinements in the upper–right part of the square propagates to the lower–left part.*

Topology control. Similarly, the topology of the isosurface can be easily controlled during simplification of a vertex w (welds): if the refined edge e is crossed at most once (below or above the welded vertex w), the topology of the isosurface will not change during the weld. This is a sufficient condition. To obtain a necessary one, we compute, when the surface crosses e twice, the Euler characteristic χ of the isosurface in the star of w. The simplification is allowed only if $\chi = 1$.

5 Method and design

The techniques we are now to introduce allow both progressive and direct encoding. Moreover, the encoding of this neighbourhood can be done uniformly (i.e., the tetrahedrons forming the tubular neighbourhood of the isosurface have constant size) or adaptively (i.e. the size of the tetrahedrons varies according to the surface).

This section is organized as follows. We will introduce our method for encoding the tubular neighbourhood of the isosurface first uniformly and then adaptively. These methods can be used to encode the base level for the progression, or to encode the entire isosurface at single rate. Then, we will detail our techniques to encode the refinements, used for the progressive compression. Again, this process can be done uniformly or adaptively. Finally, we will explain the geometry encoding, and how to reuse the compressed data of one isosurface to encode other near–by ones on the same scalar data.

(a) Base level

Our main idea is to encode the tubular neighbourhood going along with the isosurface, from one crossing tetrahedron to an adjacent one in the tubular neighbourhood. The algorithm encodes first the localization of an unvisited crossing tetrahedron t_0, and the signs of its vertices. From this initial tetrahedron, it traverses the isosurface in a depth–first–search (DFS) way, encoding the sign of each unvisited vertex encountered. When the traversal is done, it continues on the next connected component. Such traversal defines naturally an ordering on the sample points of the tubular neighbourhood. Notice that only the points of the tubular neighbourhood are encoded, leaving our algorithm almost independent of the initial 3D data size.

Figure 4 illustrates the idea by the use of an isocurve. We have the RBMT of the 2D grid, spatially adapted to the isocurve. Once the initial triangle is detected, we encode in a certain sequence only the scalar value of the vertices on the tubular neighbourhood of the curve.

Localization. The location of a tetrahedron can be encoded using the hierarchy of the RBMT, which acts as Binary Space Partition. The top of the hierarchy, i.e., the minimum triangulation, contains only a few tetrahedrons: 6 for a regular grid. Then knowing the level l of the tetrahedron to encode, it can be localized using a sequence of Left and Right symbols. The signs of its vertices are then encoded and all of them are marked as visited; the initial tetrahedron t_0 is also marked as visited.

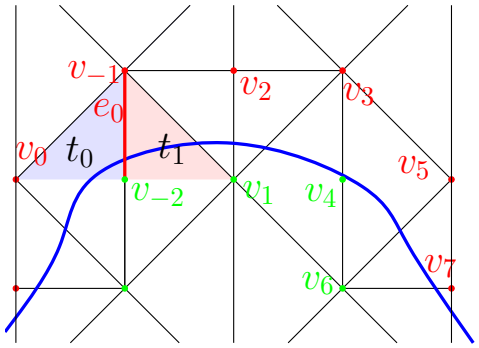


Figure 4: *Base level compression: the traversal goes from t_0 to t_1 through gate f_0, and encode the sample point v_1.*

Uniform encoding. Once the signs of the vertices of the initial tetrahedron t_0 have been encoded, its crossing faces are known to the decoder. One of them is chosen to be the first gate f_0. Such choice is done by the use of the canonical face ordering of t_0 induced by the RBMT [11] (cf Figure 4). Then, the tetrahedron t_1 ($\neq t_0$) incident to the gate f_0 is also crossing. The sign of its vertex v_1 opposite to f_0 is encoded, and both t_1 and v_1 are marked as visited. The algorithm then continues the same way starting as t_1.

Actually, a gate is valid only when t_1 has not been visited, and the sign of the vertex v_1 is encoded only when it has not been marked. When more than one gate is valid, then the algorithm pushes t_0 onto a stack. When there are no

The corresponding work was published in the proceedings of Vision, Modelling, and Visualization 2004, pp. 299–306. IOS Press, 2004.

valid gates starting at t_0, the stack is popped. The traversal stops when there is no valid gate and the stack is empty. The algorithm will then look for other connected components.

This guarantees to encode exactly one sign bit per sample point, with an overhead of a few bits per connected components, used for the localization procedure.

Adaptive encoding. The above algorithm can be easily modified to encode the isosurface described with crossing tetrahedrons of different levels. In that case, the algorithm also encodes the level of the current tetrahedron (t_1) during the traversal. The decoder will read the required level for t_1 and refine or simplify it if necessary. The RBMT actually allows only a difference of one level between adjacent tetrahedrons (cf section 4 *Adaptive Simplicial Decompositions*). Therefore, the decoder will have to refine or simplify an unvisited tetrahedron at most once, and the encoder will manage only 3 symbols: $=$ when the levels match, $>$ or $<$ when the levels differ. The encoding of the signs is done similarly to the uniform one.

This guarantees to encode exactly one bit per sample point of the tubular neighbourhood, plus one symbol of $\{=, <, >\}$ per tetrahedron of the tubular neighbourhood, with an overhead of a few bits per connected components, used for the localization procedure.

(b) Progressive refinement

The refinement methods allow to progressively adapt the tubular neighbourhood to the isosurface, using each time smaller tetrahedrons. The algorithm can simply encode the signs of the new vertices created by the refinements for all tetrahedrons on the tubular neighbourhood, or it can first specify which tetrahedrons to refine and then send the sign of their corresponding new vertices.

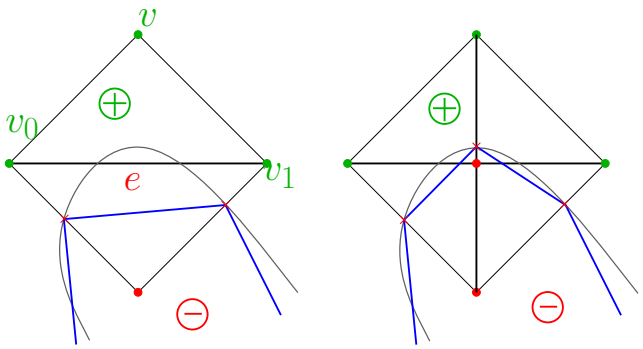


Figure 5: *A subdivision can extend the tubular neighbourhood of the isocontour. The sign of v is the one of v_1 and v_2.*

Uniform refinement. The uniform refinement simply refine first all the tetrahedrons of the tubular neighbourhood, and then en/decode the signs of the vertices created by the subdivision inside the former tubular neighbourhood. The order of such vertices are induced by the DFS traversal used on the base level en/decoding.

When subdividing a tetrahedron crossing the isosurface, the configuration of its sub–tetrahedrons is determined by the sign of the new vertex introduced on the subdivision edge. This sign is encoded during the compression. But this subdivision can locally extend the tubular neighbourhood (cf Figure 5). This case occurs when a non–crossing subdivision edge gives rise to two crossing sub–edges. In that case, the vertices opposite to this edge will be included in the tubular neighbourhood, but their signs need not to be encoded as they can be deduced by the decoder from the border of the tubular neighbourhood they belong to.

Adaptive refinement. Our method allows an adaptive refinement progression, creating smaller tetrahedrons where the isosurface is more complex, and leaving bigger tetrahedrons where it is simple. For each refinement edge in the tubular neighbourhood, we encode the Refine and Keep code. When a Keep symbol is encoded, the refinement edge will not be considered in future refinements, keeping it as a leaf on the hierarchy.

The refinement edges of the tubular neighbourhood are collected in the order of the base mesh encoding. As a subdivision propagates through the RBMT in order to maintain its smooth transitions, a subdivision of one refinement edge e of the tubular neighbourhood implies subdivisions of other refinement edges, in particular when e has a high level of subdivision. Therefore, for adaptive refinement, we sort the collected refinement edges according to their level in the RBMT, in order to encode fewer subdivision codes.

(c) Final geometry encoding

Once the complete tubular neighbourhood is encoded, the position of each vertex v of the isosurface as polygonized by [19] is *a priori* the middle point of a crossing edge of e of the RBMT. This can be improved by sending the scalar value at the endpoints of e. This is cheaper than sending the coordinate of the vertex v along the edge e, as there are more crossing edges than useful RBMT grid points (the situation is different in [8] as there are fewer crossing cubes than vertices in the octree). Moreover, this scalar value can be used to encode other isosurface generated with a different isovalue.

(d) Close–by isosurface encoding

Once an isosurface S corresponding to an isovalue α_S has been decoded, the decoder knows its whole tubular neighbourhood. It can be useful to encode also isosurfaces S' corresponding to isovalues $\alpha_{S'}$ close to α_S, especially for medical applications where the contrast of the 3D data is not well defined. S' can be easily encoded with the refinement techniques by sending first the isovalue $\alpha_{S'}$ and considering all tetrahedrons crossing S as leaves.

6 Results

[8] use the dual marching method [6] to reduce the amount of geometry to encode (only one vertex per crossing cube), which force the geometry to be sent at the end. However, our method can be used to compress meshes with a

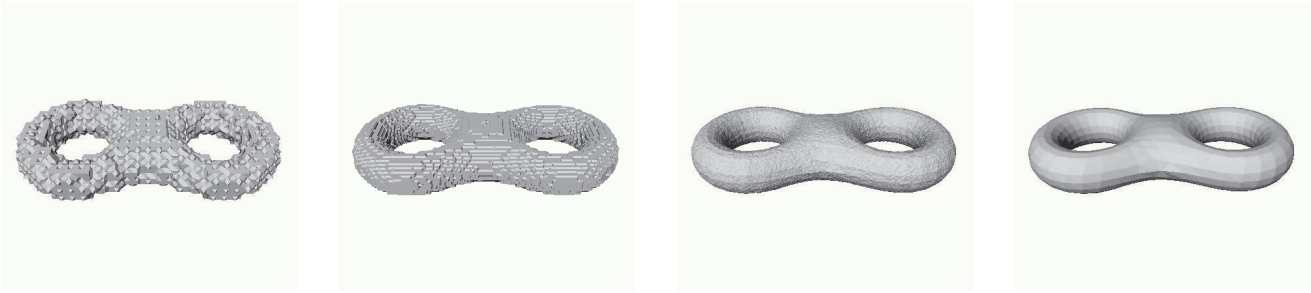


Figure 6: *Decompression of the eight model as a 257^3 regular sampled data. The first images corresponds to the binary multi–triangulation refinement, followed by the geometric refinements.*

controlled distortion at single rate. The initial mesh is simplified by successive weld operations when those welds do not increase the Hausdorff distance with the initial mesh more than a given threshold. Then the resulting mesh can be encoded adaptively at single rate.

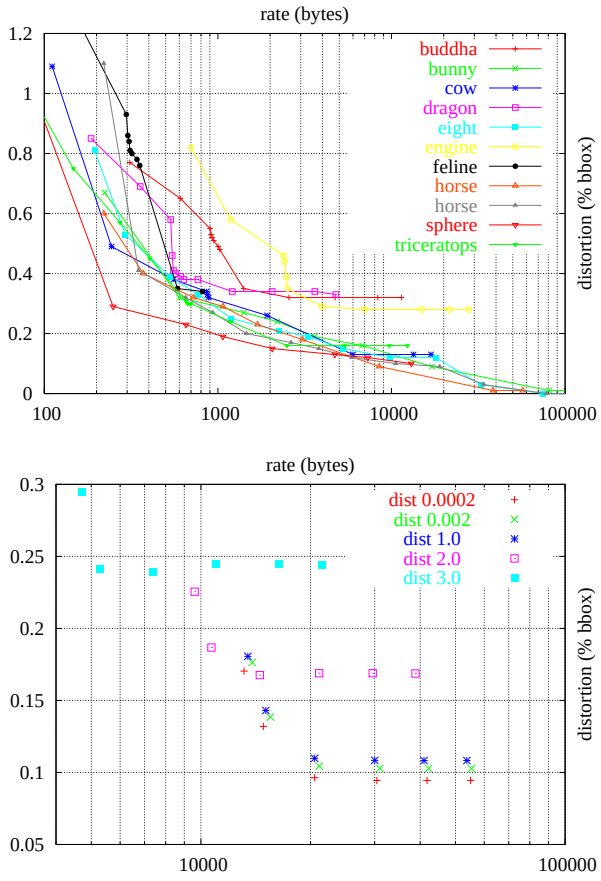


Figure 7: *Rate/distortion curves: (left) on various models, (right) of single rate encoding of the happy buddha model, with different distortion thresholds and geometry quantizations*

Figure 7(left) shows our rate/distortion curves for other models with the same strategy, and Figures 1 and 6 illustrate it.

Figure 7right shows the rate/distortion ratios for different distortion threshold and different geometry quantization. The threshold is well respected and the rate/distortion distributions converges to an optimum.

Finally, we can use this distortion control for multiresolution encoding, by alternating refinement and geometry encoding pass. This is illustrated on Figure 8, where the base mesh is encoded as in the previous single rate application, and the refinements are sent adaptatively. At each level, we encode the full geometry of the new vertices of the tubular neighbourhood (8 bits).

7 Next steps

We introduced a complete isosurface compression scheme base on simplicial and hierarchical space partition. This structure allowed us to achieve both direct and progressive compression, and to encode the isosurface uniformly or adaptively. Moreover, it provides a full control on the progression, permitting any sequence of connectivity or geometry refinements, and granting topological and geometrical control on the decoded mesh. Finally, it achieves rate/distortion ratios close to octree–based compression when the progression is done first with the connectivity and then with the geometry, and opens on other progression schemes.

We used binary multi–triangulations for the scalar data representation, implemented using dimension–independent generic programming. This offers a very simple extension of our algorithm to compress level sets in any dimension with a similar efficiency. We also intend to enhance the probability models for the arithmetic encoding, and to combine the decoding with progressive rendering techniques.

References

[1] J. W. Alexander. The combinatorial theory of complexes. *Annals of Mathematics*, 31:219–320, 1930.

[2] J. Bloomenthal, C. L. Bajaj, J. Blinn, M.-P. Cani, A. Rockwood, B. Wyvill and G. Wyvill. *Introduction to implicit surfaces*. Morgan Kaufmann, San Francisco, 1997.

[3] I. Boada and I. Navazo. An octree isosurface codification based on discrete planes. In T. L. Kunii, editor, *Spring Conference*

The corresponding work was published in the proceedings of Vision, Modelling, and Visualization 2004, pp. 299–306. IOS Press, 2004.

on Computer Graphics, pages 187–194, Comenius University, Bratislava, 2001.

[4] L. de Floriani, E. Puppo and P. Magillo. A formal approach to multiresolution modeling. In W. Strašer, R. Klein and R. Rau, editors, *Theory and Practice of Geometric Modeling*, pages 302–323. Springer, 1997.

[5] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. In *Siggraph*, volume 21, pages 372–379. ACM, 2002.

[6] T. Ju, F. Losasso, S. Schaefer and J. Warren. Dual contouring of hermite data. In *Siggraph*, pages 339–346. ACM, 2002.

[7] A. Khodakovsky, P. Schröder and W. Sweldens. Progressive geometry compression. In *Siggraph*, pages 271–278. ACM, 2000.

[8] H. Lee, M. Desbrun and P. Schröder. Progressive encoding of complex isosurfaces. *Transactions on Graphics*, 22(3):471–476, 2003.

[9] W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Kirbyfest*, volume 2 of *Geometry and Topology Monographs*, pages 299–320, 1999.

[10] 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.

[11] V. Mello, L. Velho, P. Roma Cavalcanti and C. Silva. A generic programming approach to multiresolution spatial decompositions. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 337–360. Springer, Heidelberg, 2003.

[12] S. Parker, P. Shirley, Y. Livnat, C. Hansen and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In D. Ebert, H. Hagen and H. Rushmeier, editors, *Visualization*, pages 233–238. IEEE, 1998.

[13] J. Rossignac, A. Safonova and A. Szymczak. 3D compression made simple: Edgebreaker on a corner–table. In *Shape Modeling International*, pages 278–283. IEEE, 2001.

[14] D. Saupe and J.-P. Kuska. Compression of isosurfaces for structured volumes with context modelling. In *Symposium on 3D Data Processing, Visualization, and Transmission*, pages 384–390, 2002.

[15] J. A. Sethian. *Fast marching methods and level set methods. Cambridge Monograph on Applied and Computational Mathematics*. Cambridge University Press, 1999.

[16] T. Tasdizen, R. Whitaker, P. Burchard and S. Osher. Geometric surface processing via normal maps. *Transactions on Graphics*, 22(4):1012–1033, 2003.

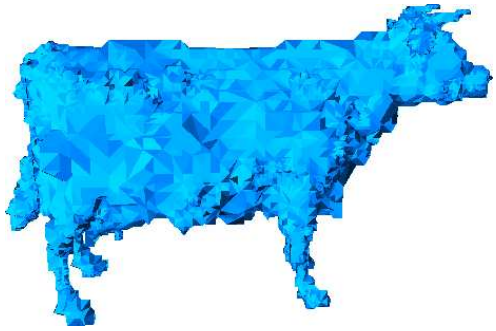
[17] G. Taubin. Blic: bi–level isosurface compression. In *Visualization*, pages 451–458, Boston, Massachusetts, 2002. IEEE.

[18] C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface*, pages 26–34, 1998.

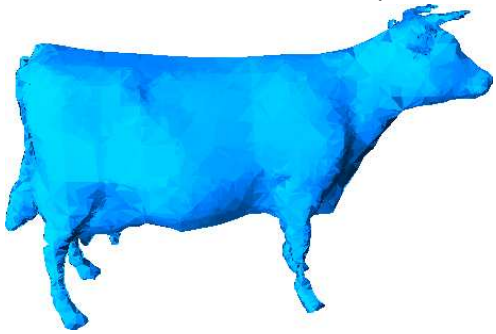
[19] L. Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2):5–25, 1996.

[20] B. von Herzen and A. H. Barr. Accurate triangulations of deformed, intersecting surfaces. In *Siggraph*, pages 103–110. IEEE, 1987.

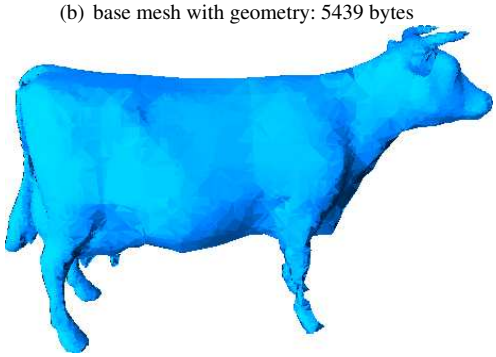
[21] S.-N. Yang and T.-S. Wu. Compressing isosurfaces generated with Marching Cubes. *The Visual Computer*, 18(1):54–67, 2002.



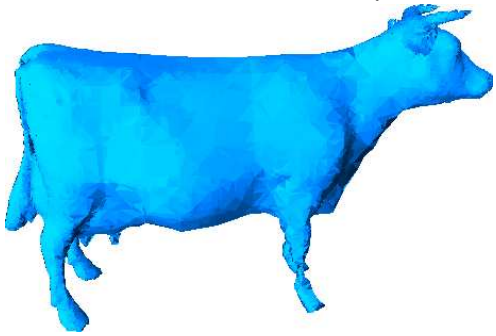
(a) base mesh : threshold 3.0, 2258 bytes



(b) base mesh with geometry: 5439 bytes



(c) first refinement: threshold 0.2, 7353 bytes



(d) geometry of the first refinement: 1135 bytes

Figure 8: *Progressive encoding according to the distortion, with alternate refinement/geometry levels.*