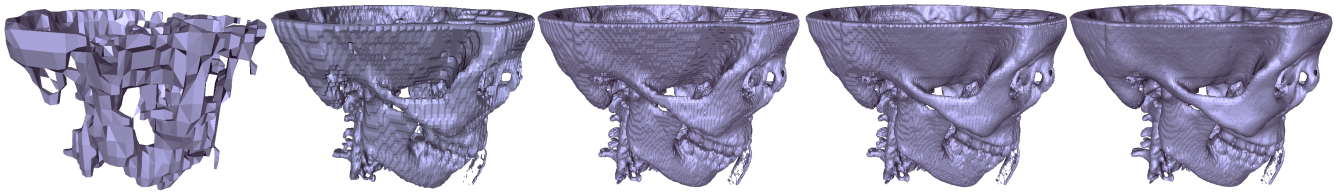
Progressive Encoding of Complex Isosurfaces

Haeyoung Lee
USC

Mathieu Desbrun
USC

Peter Schröder
Caltech



Octree level	5	7	8	8+33%	8+100%
Bytes passed	507	8605	24957	92156	226554
Distortion(10^{-4})	166.18	22.02	6.77	4.06	0.65

Figure 1: *Progressively decoded isosurface (*Headscan *model) at* 257^3 *grid resolution (octree depth of eight). Up to the last octree level, no explicit geometry is encoded. Explicit geometry bits are only associated with the final resolution level. Distortions are relative* L^2 *errors measured by Metro [Cignoni et al. 1998], normalized to the bounding box diagonal length and given in multiples of* 10^{-4}.

Abstract

We present a progressive encoding technique specifically designed for complex isosurfaces. It achieves better rate distortion performance than all standard mesh coders, and even improves on all previous single rate isosurface coders. Our novel algorithm handles isosurfaces with or without sharp features, and deals gracefully with high topologic and geometric complexity. The inside/outside function of the volume data is progressively transmitted through the use of an adaptive octree, while a local frame based encoding is used for the fine level placement of surface samples. Local patterns in topology and local smoothness in geometry are exploited by context-based arithmetic encoding, allowing us to achieve an average of 6.10 bits per vertex (b/v) at very low distortion. Of this rate only 0.65 b/v are dedicated to connectivity data: this improves by 24% over the best previous single rate isosurface encoder.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques–Interaction techniques;

Keywords: Compression, Isosurfaces, Progressive Transmission.

1 Introduction

Many of the very large, highly detailed meshes produced so far are *isosurfaces* [Bloomenthal 1997] extracted from volume data. 3D photography, volumetric imaging, and scientific simulation are frequent sources of such data. In particular, medical scans (CT, MRI) are important examples of the complex surfaces which need to be visualized, stored, or transmitted. The complexity of such models is manifested by the presence of fine geometric details throughout the volume as well as the high genus and number of connected components (see Figures 1 and 2 showing a model with genus 425; see also Table 1).

The sheer size of these meshes was a main motivation for research efforts in mesh simplification [Hoppe 1996; Lindstrom 2000], mesh compression [Deering 1995; Rossignac 1999; Taubin and Rossignac 1998; Touma and Gotsman 1998; Lee et al. 2002], and progressive transmission [Taubin et al. 1998; Bajaj et al. 1999; Pajarola and Rossignac 2000; Alliez and Desbrun 2001; Gandoin and Devillers 2002]. Mesh compression methods have steadily decreased their bit rates, but so far very few take advantage of the special structure of isosurfaces to improve their rate distortion (r/d) performance. Some of the most efficient progressive geometry compression techniques to date [Khodakovsky et al. 2000; Gu et al. 2002] are based on remeshing; unfortunately they are *impractical* for complex isosurfaces with high topologic complexity.

Previous Work To achieve better r/d performance for isosurfaces extracted from regularly sampled volume data one can exploit the special structure of such meshes. Given the voxel grid, a surface sample can be localized by specifying a voxel and a displacement with respect to the origin of that voxel. For example, the Marching Cubes (MC) algorithm [Lorensen and Cline 1987] specifies surface samples along voxel edges which "pierce" the isosurface, *i.e.*, one of the endpoints is on the strict outside while the other is on or inside the isosurface. Knowing these edges and the relative location along each edge completely specify the mesh extracted[1]. Contrast this with arbitrary meshes in which vertex coordinates are entirely unconstrained.

The observation that each vertex produced by MC can be specified by an edge and a displacement has been exploited to lower the rate by a number of authors [Saupe and Kuska 2001; Zhang et al. 2001; Yang and Wu 2002; Saupe and Kuska 2002; Taubin 2002]. These papers differ in the methods by which they localize the piercing edges and the use of context for entropy coding. Overall, this approach results in significantly lower rates: on the horse model (Figure 5) we extracted from scan-converted volumetric data, Taubin's BLIC [2002] achieves 2.4 b/v with a distortion similar to a standard mesh coder [Lee et al. 2002] at 17 b/v.

All these previous techniques are, alas, single-rate: they do not support progressive decoding. However, the size and complexity of isosurfaces in many applications is such that an embedded coder is all but required. This motivated Laney *et al.* [2002] to use wavelet encodings in the special case of distance functions, while Samet and Kochut [2002] conducted experiments to evaluate the entropy

[1] Some topologic ambiguities have to be resolved (see [Lachaud 1996]).

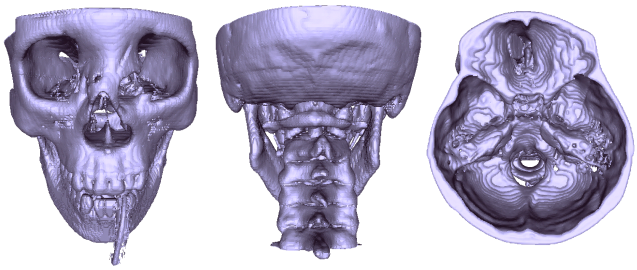


Figure 2: *An isosurface extracted from a medical dataset often exhibits high topologic complexity with many small handles and connected components. In this head scan example, there are* 183 *components and a total genus of* 425.

of adaptive octrees which approximate an isosurface. Unfortunately in both cases the results are much worse than those achieved by single rate isosurface coders.

Contributions In this paper, we present a novel algorithm for progressive encoding of isosurfaces extracted from volumetric data sampled on a regular grid. The algorithm deals naturally with high geometric and topologic complexity and produces successive refinements of the 3D model in a coarse-to-fine fashion. We exploit correlation in space and scale using an adaptive octree structure and context based entropy coding. Despite its progressive nature, the r/d performance of our algorithm surpasses all standard mesh compression algorithms as well as the isosurface-specific single-rate coders as we demonstrate on a variety of test cases.

2 Setup and Design Choices

Before going into a detailed description of our algorithm, we begin with a discussion of desirable features to motivate our subsequent design choices. We also use this section to fix notation.

Desiderata The features we desire in our coder are straightforward. We wish to:

- accept regularly sampled scalar (*e.g.*, from MRI and CT, level set simulation, and 3D photography sources), or Hermite volume data as it allows for the description of sharp features [Kobbelt et al. 2001].
- produce significantly better r/d performance than general purpose mesh coders;
- exploit the special structure of isosurface meshes;
- produce an embedded bitstream suitable for progressive reconstruction of successively refined geometry and topology while producing crack free reconstructions at all times.

Design Choices Given these requirements, we chose

1. an adaptive octree based approach for the localization of piercing edges. It takes advantage of the special nature of isosurface meshes providing progressive localization a bit plane at a time. It also provides contexts for entropy coding which can exploit correlation both spatially and across scale. To support adaptive reconstruction we need the inside/outside status for all vertices of the grid, the transmission of which must be performed with care to avoid obvious redundancies.
2. the Dual Contouring method [Ju et al. 2002; Schaefer and Warren 2002] for isosurface extraction. Its dual nature produces *watertight* meshes for *any* adaptive description of the original grid. It can also handle Hermite data, which enables the encoding of surfaces with sharp features (see Figure 5).

Definitions The *volume data* is given as a regular 3D grid of samples of a scalar function (possibly with hermite data) at some resolution N: $\{d_{ijk}\}$ for $i, j, k \in [0, N-1]$. The spatial location of d_{ijk} is denoted $\mathbf{x}_{ijk}$. We assume $N = 2^j + 1$ for some $j > 0$, as any other grid size can be extended and padded with outside values at small entropy cost. The *sign* bit at $\mathbf{x}_{ijk}$ is the predicate $d_{ijk} \leq c$, *i.e.*, zero iff $\mathbf{x}_{ijk}$ is on the strict exterior of the isosurface. The type of a cell is *homogeneous* if its 8 corners carry the same sign bit and *inhomogeneous* otherwise (following the convention of [Ju et al. 2002]). Edges are homogeneous if the sign bits on either end agree, and inhomogeneous otherwise. The latter corresponds to piercing edges mentioned earlier. An inhomogeneous cell always has at least one inhomogenous edge. During octree traversal we enumerate children cells in lexicographic order, which in turn dictates the order of sign bit transmission. Finally, the *distortion* of a given reconstruction is evaluated as the L^2 norm of the distance from the reconstruction to the original surface using Metro [Cignoni et al. 1998]. The measure is symmetrized by exchanging the roles of original and reconstruction and the result is normalized by the bounding box diagonal.

Models	#V	#F	#E	#CC	genus
MRI-hd.	1415850	1450574	2867855	5793	6508
Engine	304384	301005	605193	265	167
Hd.scan	280039	281154	561677	183	425
Sphere	213544	213542	427084	1	0
Feline	192848	196698	389544	12	11
Dragon	191154	191318	382418	21	-6
Bonsai	165467	166396	330677	510	-83
Buddha	80755	81004	161755	23	21
Horse	69311	69318	138629	1	1
Eight	67214	67218	134434	1	2
Tricera.	33696	33708	67403	1	1
Temple	219950	219958	439916	1	5

Table 1: *Statistics for sample datasets: number of vertices, faces, edges, connected components, and genus. Each model is extracted from a* 257^3 *grid, except for Feline and Dragon that come from a* 513^3 *grid. Sphere, Horse, Eight, and Triceratops are scan-converted from meshes to volume data to compare with previous mesh compression methods. The temple model is distinguished by its synthetic origin and sharp edges.*

3 Isosurface Encoding Algorithm

The encoder is structured into three main phases. In the first phase a pass over the original volume data produces an adaptive octree representation of the sign bitmap (*i.e.*, the inside/outside function), as well as the surface samples of the Dual Contouring extraction. In a second phase this sign bit octree, as well as the type of each cell, are turned into a bitstream during a coarse to fine traversal. Hierarchical prediction and context coding exploit correlations in space and scale. This completes what we will refer to as the **connectivity encoding** stage of the algorithm, since it defines the topology and connectivity of the Dual Contouring mesh. Notice that, because of the peculiar nature of isosurfaces, the bitstream produced at this point contains connectivity *and* geometry information since the surface samples have effectively been localized to within their final voxels. The third phase performs final **geometry encoding** by progressively sub-localizing the vertices of the Dual Contouring mesh. To further reduce bit rate we decompose the geometry residuals into tangential and normal components.

Since the decoder performs the exact symmetric set of operations we will give algorithms for the encoder only.

3.1 Adaptive Octree Construction

The input of the algorithm is the set $\{d_{ijk}\}$ possibly with associated Hermite data. In either case we need to locate the inhomogeneous edges to specify the isosurface. These edges can be derived from a knowledge of the sign bitmap. For example, Taubin [2002] used a sequential traversal of each volume slice of the sign bitmap to enumerate these edges. Since we want a progressive bitstream we perform the transmission coarse to fine. For this we need a sign bit

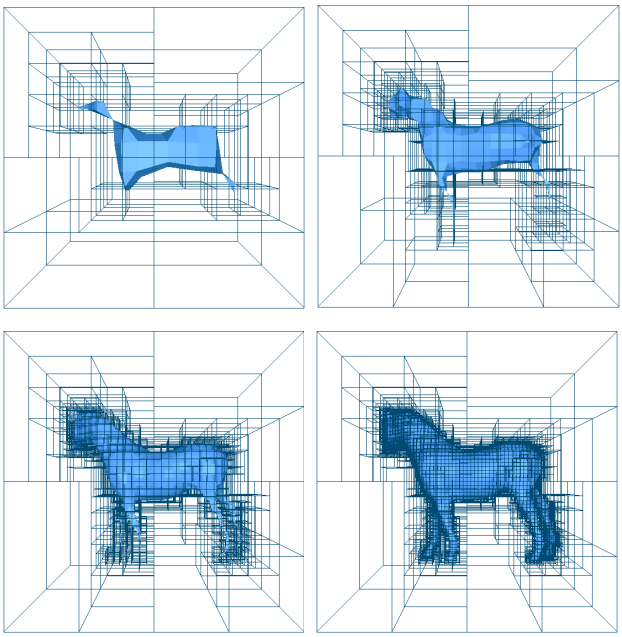


Figure 3: *Visualization of the adaptive octree (and associated reconstruction) for the Horse model for levels three through six.*

octree which is constructed in a fine to coarse fashion. Whenever all eight children are homogeneous they are deleted. This leads to coarse cells in homogeneous regions and refinement of the tree near the isosurface (see Figure 3). The inhomogeneous leaves of this tree contain vertices of the Dual Contouring mesh. The structure of this tree fully specifies the connectivity of the isosurface while the vertices stored with the leaves recover the geometry of the Dual Contouring mesh.

If Hermite data is available we use the Dual Contouring algorithm to find the optimal position within the leaf cell. In the case of sharp edges, this will ensure that the vertices are snapped to these features. For scalar volume data we compute the isosurface intersection points along each inhomogeneous edge of a leaf cell and use their barycenter as the leaf vertex. Note that hybrid techniques to position the leaf vertices can also be used. For example, normals can be estimated by finite differences if the data is clean enough.

At this point we have extracted all topology and geometry information from the original volume data and will need it no further. Notice that normals are only used initially to find the optimal vertex positions, so we do not transmit the normals in our technique.

3.2 Connectivity Encoding

Principle The octree must now be transmitted in a coarse to fine fashion to support progressivity. Together with the sign bits of the corners of a given cell we must also indicate whether the cell is a leaf node. The leaf/non-leaf status of a partially reconstructed octree at the receiver will guide interpretation of subsequent bits: leaves will not be explored further. At the finest level of the octree, any cell is a leaf node by definition. At all but the finest level, the algorithm uses the sign bits at cell corners to *predict* the leaf/non-leaf status: an inhomogeneous cell is not a leaf; a homogeneous cell may *not* be a leaf. For example, if a small connected component at a fine level of the hierarchy is eventually contained in a coarse cell with uniform sign bits at its corners, it is homogeneous yet not a leaf cell. Consequently we send a *leaf bit* for each homogeneous cell, indicating whether it is a leaf or not.

To ensure synchronized decoding, we start by sending the grid size N from which the decoder deduces the number of levels ($\lceil \log_2 N \rceil$), followed by the eight sign bits of the root cell of the octree. The algorithm then performs a breadth first traversal of the adaptive octree in lexicographic order. For each cell it emits the sign bits of the eight children unless they have already been transmitted—both encoder and decoder can do so in lock step—and the leaf bit (yes/no), only if the cell is homogeneous and not at the finest level. Sign and leaf bits are emitted in two different streams since their entropy encoding will use different contexts. We experimented with different orders and different marking strategies but found this approach to consistently yield the best results.

Implementation Although the algorithm is best described using an octree representation, our implementation does not use a pointer-based octree data structure: it would become very memory intensive for large volume data. Instead, we use a light bitmap structure, with leaf and sign bits stored in bit arrays of size N^3. Surface samples are stored in a hash table. For performance reasons recursion on the data structures is replaced by a simple FIFO queue. Such structures make for a small memory footprint, and enable us to encode very large datasets (1024^3 and larger).

Context-based Arithmetic Coding As an entropy coder we use Wheeler's multi-symbol arithmetic coder [1996], since it outperformed other encoders we tried. To further improve compression rates we use context modeling [Pennebaker and Mitchell 1993] which automatically updates context based probability tables to learn patterns in the bitstream and reduce the entropy accordingly. We found the performance of this context based arithmetic encoding highly dependent on the choice of context, with good contexts drastically reducing bitrate as already demonstrated in [Taubin 2002].

After many trials, we chose the following contexts for their consistently good performance:

- the sign bitstream uses a JBIG-style 15-bit context: the seven neighboring sign bits from the corners of the same cell (in lexicographic order) followed by the eight sign bits of the parent cell. For signs not yet transmitted we use zeros in the context.
- the leaf bitstream uses only a one-bit context consisting of the previous leaf bit sent. Zeros are used for bootstrapping.

Geometry Although this first part of our compression technique can be seen as a pure hierarchical topology description, geometric information is implicitly present. The grid nature of an isosurface intertwines topology and geometry at all levels of the octree. Inhomogeneous edges, independent of their level in the hierarchy, intersect the isosurface, implying that we have *approximate* positions for the vertices throughout the connectivity transmission process. The shorter the edges, *i.e.*, the finer the level of the octree, the better the approximation. The *final* vertex positions will need extra geometry bits.

In practice, we update the display each time a new inhomogeneous cell has been received. The Dual Contouring algorithm is called locally to rebuild the mesh for display. It requires a vertex for each inhomogeneous cell. We use the *barycenter* of the midpoints of each inhomogeneous edge. Figures 1 and 5, as well as the r/d curves (Figure 4), demonstrate the quality of progressive visual increments throughout the connectivity transmission. For some isosurfaces, stopping at the end of the connectivity decoding phase (or even earlier) may provide enough geometric accuracy (smoothing techniques can be used to improve the visual aspect): for single rate coding this was already suggested by Taubin [2002]. When higher geometric fidelity is desired, explicit geometry bits are sent in the final phase.

3.3 Geometry Encoding

At the end of the octree decoding phase the final topology has been recovered, but the geometric positions can still be further refined. In contrast to all previous methods, we do not choose to localize vertex positions on edges (as in MC), but within cells. At first sight this appears more costly since there are now three coordinate residuals

rather than one. In running experiments for a large number of isosurfaces we found that locations along edges are distributed more or less uniformly, *i.e.*, these locations have high entropy. Interestingly, the entropy, and thus the ultimate bit budget, is much lower for Dual Contouring vertex residuals. There is another advantage to using the Dual Contouring method: it accommodates surfaces with *sharp features* and is thus more general than MC. Previous isosurface coders could not handle this more general case. To reduce entropy, we send the final position as prediction residuals, *i.e.*, the difference between a predicted position and the actual position. This residual is expressed in a local frame, which further reduces entropy.

Local Coordinate Frame We begin by visiting all octree cells at the finest level. For each inhomogeneous cell (*i.e.*, a cell containing a surface sample) the barycenter of all inhomogeneous edge midpoints is computed. Call this the *predicted* vertex location. The decoder can perform the same computation without any further information. Recall that the actual Dual Contouring sample location was computed by the encoder during the initial octree construction. To build the local coordinate frame we compute a least squares fitting plane based on all inhomogeneous edge midpoints. There are only 256 different cases possible, so a simple lookup table for the least-square plane is sufficient. The predicted vertex location is on this plane and may be taken to be a local origin. The local frame now follows from the normal direction and an arbitrary but fixed choice of orthogonal basis vectors in the plane. In our implementation, we pick x to be the normalized projection of the global unit vector i onto the plane; if the plane is normal to this direction, we pick j instead. The y-axis follows as the cross product of the normal and x.

Coordinate Encoding Once the prediction vectors have been expressed in their respective local frames we determine the three—two tangent (x, y) and one normal (z)—global ranges of each coordinate and send these three ranges to the decoder at the beginning of the geometry encoding phase. A quantization of the coordinates of all prediction vectors is then performed, with more bins used for the normal direction than for the tangent directions [Khodakovsky et al. 2000; Alliez and Desbrun 2001; Lee et al. 2002]. We found that using 6 to 10 bins for x and y, and 12 to 20 bins for z lead to the best r/d performance in all our tests. Finally, in order to achieve a fine-grain transmission, we send the geometry bits in two passes: first the z coordinates, then the x and y coordinates.

Context Modeling The geometric residuals described above are encoded in a separate geometry bitstream, using a context-based arithmetic encoder as before (Section 3.2). The eight sign bits of the containing cell are used for context.

4 Results

We have run tests on a number of isosurface datasets and experimented with a variety of encoding methods. Here we report quantitative and qualitative results of these experiments. Figures 1 and 5 show progressive reconstructions of a variety of surfaces together with rates and distortions. All images are flat shaded to more clearly show the surfaces. No postsmoothing has been performed. These images qualitatively indicate that the early reconstructions give useful overall views of the isosurface while finer level detail, both geometric and topologic, appears gracefully. Note that all tessellations throughout the reconstruction process are watertight because of the use of the Dual Contouring method. More detailed quantitative data is recorded in Tables 2 and 3, while r/d performance throughout the reconstruction process is plotted in Figure 4. On average (using ten isosurfaces) we find bitrates of 6.10 b/v of which the octree encoding consumes 0.65 b/v. Explicit geometry (prediction residuals at the finest level) use up the remainder of 5.45 b/v. This implies that most of the rate is used at the leaf level. The r/d curves show that this additional rate is well spent if high fidelity reconstructions are called for. Note finally that both the encoding and decoding phases of a 257^3 volume take about 20 seconds on a 1GHz PIV 512Mb computer with our current implementation.

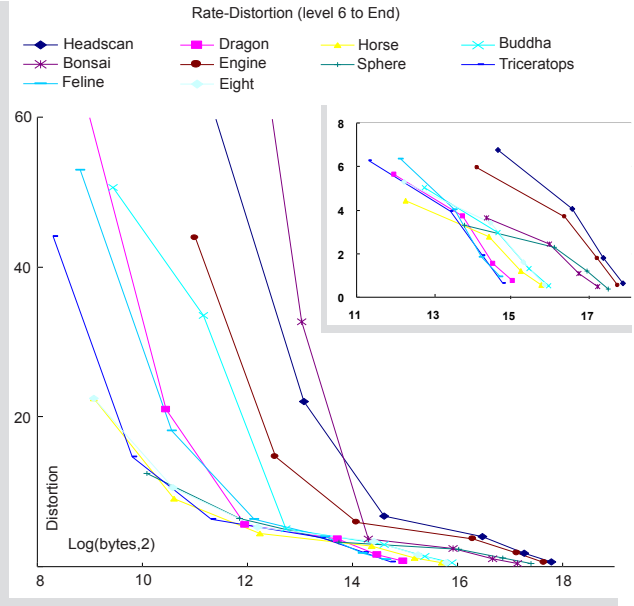


Figure 4: *R/d curves (relative L^2 error in multiples of 10^{-4} as a function of total bytes transmitted on a semi-log scale) for ten example isosurfaces (all from 257^3 volumes). The bottom graph shows results throughout the transmission while the inset graph gives a closeup of the explicit geometry phase of transmission (with a steeper slope).*

Isosurf. models	Sign stream (byte)	Leaf stream (byte)	Con. rate (b/v)	Ours/BLIC (%)
MRI-head	132891	19055	0.86	92
Engine	16431	1014	0.46	78
Headscan	22733	2224	0.71	86
Sphere	13718	10	0.51	58
Feline513	17679	2581	0.84	73
Dragon513	12753	642	0.56	65
Bonsai	16058	4266	0.98	93
Buddha	6324	515	0.68	82
Horse	4703	151	0.56	67
Eight	4618	101	0.56	65
Triceratops	2419	142	0.61	75
Average			0.67	76
Temple	1348	241	0.06	81

Table 2: *Connectivity Encoding of sample datasets: sign bitstream, leaf bitstream, and total connectivity rate in b/v. Temple (synthetic data) is given separately and is not included in the average. The ratio of improvement over the current best single rate encoder [Taubin 2002] is also indicated.*

4.1 Discussion

Ideally one would like to compare different algorithms against the same dataset computing rates and distortions on the same scale. At present the results in the literature are few and reported in inconsistent ways, making real comparisons hard. For example, some authors give compression ratios against formats such as ASCII VRML. Unfortunately such numbers are very soft since ASCII files can be arbitrarily verbose, leading to ill-defined ratios. In any case, without specifying the distortion any ratio or file size is of limited utility. Nonetheless, we have attempted to perform at least some informal comparisons.

Isosurf. models	Geo. stream (byte)	Geo. rate (b/v)	Tot. streams (byte)	Tot. rate (b/v)
Engine	190264	5.00	208199	5.46
Headscan	201597	5.76	226554	6.47
Sphere	160217	6.00	173945	6.52
Feline513	99110	4.11	119370	4.95
Dragon513	109103	4.57	122498	5.13
Bonsai	125384	6.06	145708	7.04
Buddha	54325	5.38	61164	6.06
Horse	48852	5.64	53706	6.20
Eight	51694	6.15	56413	6.71
Triceratops	24700	5.86	27261	6.47
average		5.45		6.10
Temple	93246	3.39	94956	3.45

Table 3: *Encoding of sample datasets: Number of bits sent during the geometry encoding phase, and corresponding rates; total number of bits (connectivity + geometry encoding) for each dataset, along with achieved rates per vertex.*

Laney *et al.* [2002] introduced a progressive isosurface compression method based on encoding an associated distance function. They reported results for two models, one of them being the horse on a grid of size $96 \times 208 \times 173$. Their final size was 66 KB with 0.8 as reported distortion. We achieved a final size of 52.4KB at a distortion of 0.56. Samet and Kochut [2002] used an adaptive octree and provided measured entropies of their symbol sequences at 0.847 bits per symbol. Such entropies appear fairly high compared to our *achieved* bit rates for the octree description.

There are somewhat more results available for single rate isosurface coders. The current leading coder of this type appears to be the one proposed by Taubin [2002]. Our algorithm improves on his results by an average of 24% (see Table 2), even though our algorithm is progressive. To evaluate the geometry coding performance distortion measures are required, which, unfortunately, are not provided by any previous work. Saupe and Kuska [2001; 2002] give compression ratios for 12-bit quantization and reported better ratios than Touma and Gotsman's [1998]. We found that such quantization leads to distortions on the order of $0.5(10^{-4})$ to $1.0(10^{-4})$ and use this as a basis for comparison. They tested four simple isosurfaces of one component each and several isosurfaces generated from a CT scan using different isovalues. Their best result was 1.2% of its ASCII VRML representation [2002]. While their models are different we computed an average ratio for our, more complex, isosurfaces of 0.71%, which represents an additional reduction of 41%. Note that the finer the grid the more our method improves over previous techniques.

5 Summary and Future Work

To summarize, we have introduced a simple and efficient progressive encoding technique specifically designed for isosurfaces. Our algorithm supports more general isosurfaces than previous algorithms due to the use of Dual Contouring. We also found that this lowers the entropy of prediction residuals significantly, leading to significantly better bit rates when compared to any previously published compression results, be they single rate isosurface specific, or achieved by a general purpose mesh compression method.

Supporting view dependent refinement in the decoder is an interesting problem for future work. Finally, we wish to explore other avenues to decorrelate the bits describing the octree hierarchy. Our current approach of using hierarchical parents for prediction at children can be likened to a Haar transform on the sign bits. More sophisticated binary valued wavelet transforms are possible and may lead to further improvements and insights.

Acknowledgment This work was supported in part by NSF (DMS-0220905, DMS-0138458, DMS-0221666, DMS-0221669, CCR-0133983, EEC-9529152, ACI-0219979), the DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Intel, Alias|Wavefront, Pixar, and the Packard Foundation. Special thanks to Pierre Alliez, Scott Schaefer, Joe Warren, and C.-C. Jay Kuo. Datasets are courtesy of Rezk-Salama *et al.* (http://www9.informatik.uni-erlangen.de/Persons/Rezk/Research/VolRen), Zoë Wood, Stanford Graphics Group, Scott Schaefer, and Sylvain Jaume.

References

ALLIEZ, P., AND DESBRUN, M. 2001. Progressive Encoding for Lossless Transmission of 3D Meshes. In *Proc. of SIGGRAPH*, 198–205.

BAJAJ, C. L., PASCUCCI, V., AND ZHUANG, G. 1999. Progressive Compression and Transmission of Arbitrary Triangular Meshes. In *Proc. of IEEE Visualization*, 307–316.

BLOOMENTHAL, J. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum 17*, 2, 167–174.

DEERING, M. 1995. Geometry Compression. In *Proc. of SIGGRAPH*, 13–20.

GANDOIN, P.-M., AND DEVILLERS, O. 2002. Progressive Lossless Compression of Arbitrary Simplicial Complexes. *ACM Trans. on Graphics 21*, 3, 372–379.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry Images. *ACM Trans. on Graphics 21*, 3, 355–361.

HOPPE, H. 1996. Progressive Meshes. In *Proc. of SIGGRAPH*, 99–108.

JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual Contouring of Hermite Data. *ACM Trans. on Graphics 21*, 3, 339–346.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive Geometry Compression. In *Proc. of SIGGRAPH*, 271–278.

KOBBELT, L. P., BOTSCH, M., SCHWANECKE, U., AND SEIDEL, H.-P. 2001. Feature Sensitive Surface Extraction from Volume Data. In *Proc. of SIGGRAPH*, 57–66.

LACHAUD, J.-O. 1996. Topologically Defined Iso-surfaces. In *Proc. 6th Discrete Geometry for Computer Imagery*, Springer-Verlag, Berlin, vol. 1176, 245–256.

LANEY, D., BERTRAM, M., DUCHAINEAU, M., AND MAX, N. 2002. Multiresolution Distance Volumes for Progressive Surface Compression. In *Proc. of the 1st Intl. Symp. on 3D Data Processing Visualization and Transmission*, 470–479.

LEE, H., ALLIEZ, P., AND DESBRUN, M. 2002. Angle-Analyzer: A Triangle-Quad Mesh Codec. *Computer Graphics Forum (Proc. of Eurographics) 21*, 3, 383–392.

LINDSTROM, P. 2000. Out-of-Core Simplification of Large Polygonal Models. In *Proc. of SIGGRAPH*, 259–262.

LORENSEN, W., AND CLINE, H. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (Proc. of SIGGRAPH) 21*, 4, 163–169.

PAJAROLA, R., AND ROSSIGNAC, J. 2000. Compressed Progressive Meshes. *IEEE Trans. on Visualization and Computer Graphics 6*, 1, 79–93.

PENNEBAKER, W. B., AND MITCHELL, J. L. 1993. *JPEG: Still Image Date Data Compression Standard*. Van Nostrand Reinhold.

ROSSIGNAC, J. 1999. EdgeBreaker : Connectivity Compression for Triangle Meshes. *IEEE Trans. on Visualization and Computer Graphics 5*, 1, 47–61.

SAMET, H., AND KOCHUT, A. 2002. Octree Approximation and Compression Methods. In *Proc. of the 1st Intl. Symp. on 3D Data Processing Visualization and Transmission*, 460–469.

SAUPE, D., AND KUSKA, J.-P. 2001. Compression of Isosurfaces for Structured Volumes. In *Proc. of Vision, Modeling and Visualization*, 333–340.

SAUPE, D., AND KUSKA, J.-P. 2002. Compression of Isosurfaces for Structured Volumes with Context Modelling. In *Proc. of the 1st Intl. Symp. on 3D Data Processing Visualization and Transmission*, 384–390.

SCHAEFER, S., AND WARREN, J. 2002. Dual Contouring: "The Secret Sauce". Tech. rep., Rice University.

TAUBIN, G., AND ROSSIGNAC, J. 1998. Geometry Compression Through Topological Surgery. *ACM Trans. on Graphics 17*, 2, 84–115.

TAUBIN, G., GUEZIEC, A., HORN, W., AND LAZARUS, F. 1998. Progressive Forest Split Compression. In *Proc. of SIGGRAPH*, 123–132.

TAUBIN, G. 2002. BLIC: Bi-Level Isosurface Compression. In *Proc. of IEEE Visualization*, 451–458.

TOUMA, C., AND GOTSMAN, C. 1998. Triangle Mesh Compression. In *Proc. of Graphics Interface*, 26–34.

WHEELER, F., 1996. Adaptive Arithmetic Coding Source Code. http://www.cipr.rpi.edu/~wheeler/ac.

YANG, S.-N., AND WU, T.-S. 2002. Compressing Isosurfaces Generated with Marching Cubes. *The Visual Computer 18*, 1, 54–67.

ZHANG, X., BAJAJ, C., BLANKE, Q., AND FUSSELL, D. 2001. Scalable Isosurface Visualization of Massive Datasets on COTS Clusters. In *Proc. of IEEE Symp. on Parallel and Large Data Visualization and Graphics*, 51–58.

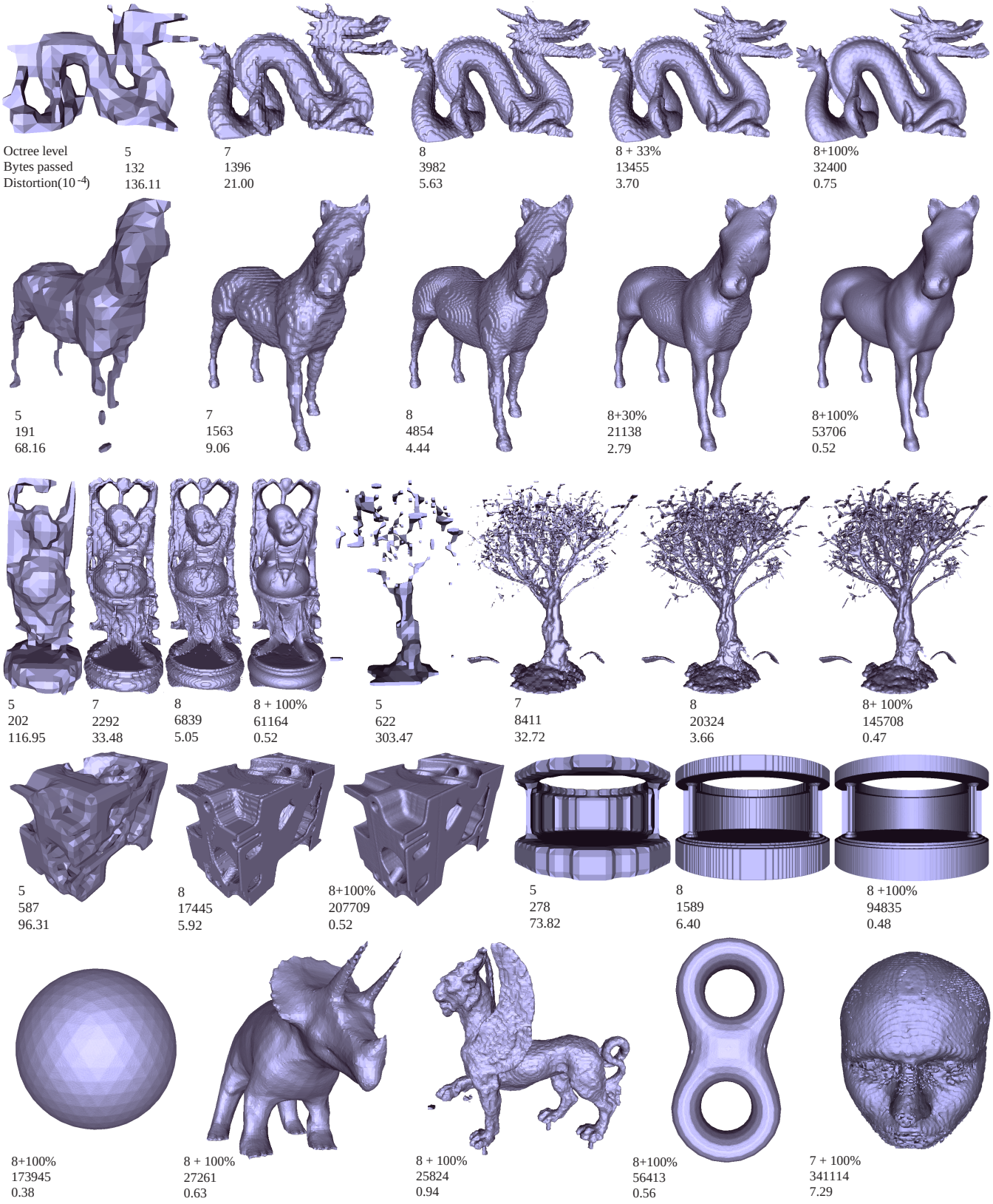


Figure 5: *Examples of progressively decoded isosurfaces from octree hierarchies eight levels deep, shown at different stages during the transmission. Rates are given in bytes. Distortions are relative L^2 errors measured by Metro [1998], normalized to the bounding box diagonal length and scaled to 10^{-4}. From top to bottom: Dragon, Horse, Buddha and Bonsai, Engine and Temple, Sphere, Triceratops, Feline, Eight, and MRI-Head. Sphere, Horse, Eight, and Triceratops are scan-converted from meshes to volume data to compare with previous mesh compression methods.*