Data-driven Progressive Mesh Compression Using Associated Properties

Gabriel Cirio[1], Guillaume Lavoué[2,3], and Florent Dupont[2,4]

[1] INRIA Rennes, France
[2] Université de Lyon, CNRS
[3] Université Lyon 1, LIRIS, UMR5205, F-69622, France
[4] INSA-Lyon, LIRIS, UMR5205, F-69621, France
gabriel.cirio@irisa.fr {glavoue,fdupont}@liris.cnrs.fr

Abstract. Many 3D models carry associated properties, such as normals or colors, with the geometry and connectivity data. Existing progressive mesh compression techniques usually do not take these properties into account during compression, missing the quality improvement of the intermediate decompression meshes. In this work, we propose a novel progressive compression algorithm that uses the associated properties of the mesh to drive the compression process. Any property can be used as long as a distance function is defined for that particular property. The algorithm builds a kd-tree structure using a voxelisation process, which recursively separates the set of vertices according to the associated properties distances, resulting in an improved quality of the intermediate meshes. We evaluate our method by comparing its compression ratios to recent algorithms, and by conducting a perceptive evaluation. At equal rates, our method provides a better overall visual quality.

Keywords: Progressive mesh compression, Associated properties, Data-driven, Kd-tree

1 Introduction

Nowadays, 3D models are widely used in many fields involving 3D graphics such as scientific visualization, engineering, virtual reality, gaming, e-commerce and education. They are becoming as popular as other media types like audio, image and video. In recent years, due to improvements in three dimensional data-acquisition, storing and display devices, the demand for high resolution models has exploded. This increase in the complexity of 3D models brought the need to use efficient compression techniques in order to satisfy the requirements of graphic applications in terms of compact storage and fast transmission of models. The increasing use of networked applications, low computational power devices (such as mobile phones or PDAs) and huge size meshes calls for progressive visualization of models through intermediate quality meshes.

A 3D mesh is made of geometry data and connectivity data. However, in many cases there are properties associated to the mesh's elements, most commonly to vertices. These properties can be colors, normals, or, such as in the

case of scientific visualization, density or temperature values. Most recent mesh compression algorithms focus only on geometry and connectivity data compression. However, property data can be as large or even larger in size than geometry and connectivity data together, hence dedicated compression algorithms could greatly improve coding efficiency. Moreover, existing techniques are driven by geometry or connectivity, meaning that only the geometry or the connectivity is taken into account to optimize the compression, while associated properties play a secondary role.

In this work, we describe a framework that takes into account vertex-bound associated properties for the driving of the compression (the technique is therefore called data-driven), and that compresses the associated properties together with the geometry and the connectivity data. We believe that a compression process driven by the associated properties will deliver better results in terms of visual quality during decompression, particularly for meshes where geometry and topology play a secondary role compared to associated properties, such as with many scientific visualization models.

In the next section, we survey the existing techniques for progressive mesh compression and associated properties compression, and we describe the few algorithms that take into account associated properties for data-driving. In section 3 we present our algorithm for data-driven compression. In section 4 we compare our results with recent techniques and present a perceptive evaluation of the quality of the compressed meshes, before concluding.

2 Related Work

2.1 Progressive Mesh Compression

In this section, we briefly describe the different existing approaches for progressive mesh compression. For a more comprehensive review, we refer the reader to [3] and [19].

Connectivity-based approaches first process connectivity data, with the vertices positions playing a secondary role. Hoppe [12] reported the first technique, where he simplifies the mesh by performing a sequence of edge collapses. Later techniques focused on improving the compression ratio through implicit indexing [17], through vertex decimation techniques [9] [2], or by extending single-rate methods to achieve progressivity [23] [4].

More recent research has focused on geometry-based approaches, achieving higher compression ratios. Gandoin and Devillers [11] proposed a kd-tree space subdivision technique, where they recursively subdivide the space in two cells and encode the number of vertices contained in one of the child cells. The geometry compression relies on the bit sharing property, since grouping vertices in a common cell makes them share the first high-level bits of their positions. Peng and Kuo [20] improve the technique by using an octree subdivision and efficient prediction schemes, but at the cost of restricting the algorithm to triangular meshes only. bordignon et al. [6] extend [11] algorithm by using a BSP-tree,

in order to obtain child cells with the same number of vertices and therefore maximise the bit sharing property.

Other apporaches for progressive mesh compression techniques exist, but are not well suited for our requirements. Spectral coding [14] does not allow a local control of the compression algorithm, making it inappropriate for data driving. Wavelet based compression [16] [15] requires special mesh properties, such as regularity or piecewise regularity, and often a remeshing pre-processing step. Since many applications cannot afford these constraints, we do not consider these techniques in our work.

2.2 Property compression

The compression of mesh associated properties plays a secondary role in recent compression schemes. However, mesh properties can often be of greater size than other mesh components: if we consider a color 3 floating-point value property attached to vertices, both geometry and color data are equally important in size. Here, we survey existing techniques that deal with the progressive compression of these associated properties.

Many algorithms on progressive compression of associated properties can be found in the area of progressive compression of point clouds. Waschbusch et al. [24], while recursively contracting pairs of neighboring vertices in point-clouds, also contract their YUV color data. They encode the color of the resulting point by delta coding with one of the contracted points. The same method is applied for the coding of normals, using the normal's spherical coordinates. Huang et al. [13], with their octree-based geometry compression data structure, further improve properties compression ratios by using progressive normal quantization technique based on [22] and improved in [7]. The normal unit sphere, first represented by an octahedron, is subdivided along the mesh, and each normal of the intermediate mesh is mapped to a facet of the subdivided octahedron, which leads to a progressive increase in precision. The normal of a cell is the average of the normals of the vertices inside the cell. For color compression, they use an optimized custom color frame and adaptive quantization. Each component is encoded through delta coding.

Some research has focused on the compression of a single property, like the progressive compression of normals from Bass and Been [5]. Their algorithm can be applied in many progressive compression techniques. They use the recursive subdivision of cones to progressively represent the normals. The coarsest normals have an attached cone containing the normal. The initial cone is then subdivided into 7 cones set up in a way that minimizes overlapping while covering the entire parent cone. The refined normals fall into one of the 7 sub-cones, and the process is repeated recursively until the highest precision is reached.

2.3 Data-driven compression

Little research has been conducted to exploit associated property data to drive the compression of the geometry, in order to achieve a higher quality of the

intermediate meshes. The aim has been improving the compression ratios and not necessarily the decompression quality. To the best of our knowledge, there are few papers which make the first steps into data-driven progressive mesh compression.

Peng et al. [18] recursively split the initial set of vertices of the mesh into several child sets, through the use of a distance metric applied on the geometry data and the normal property data. However, no compression ratio is given. Waschbusch et al. [24], as previously mentioned, recursively contract pairs of neighboring points. In order to decide which pairs will be contracted, and in order to obtain the best set of pairs, they solve a minimum weight perfect matching graph problem. The weight of each edge of the graph is the sum of the distance between the two vertices for each associated property, together with their Euclidean distance. Each distance is given by a distance function that returns the difference between two instances of the same property. In the examples given, they use the normals as driving associated property, and the distance between two normals is the cosine of their enclosing angle. By taking into account normal data, both techniques lead to an increased quality of the intermediate levels of detail.

These algorithms show an initial concern for data-driving the progressive compression of meshes. However, they do not provide a generic approach for any kind of property or they exploit these properties in a limited way. Inspired from [24] distance functions and based on [11] kd-tree compression technique, we propose a framework for the data-driven progressive compression of arbitrary meshes.

3 Our Data-driven Framework

The space subdivision techniques [11] [20] [6], while achieving high compression ratios, use data structures that allow a certain flexibility in the splitting of a cell: the position of the cutting plane. Our algorithm exploits this degree of freedom by choosing, similarly to the BSP-tree algorithm from [6], the position of the cutting plane for each cell subdivision. The position is selected according to a set of variables which are, in our work, the associated properties. Due to the orthogonality constraint in the kd-tree cutting planes, a kd-tree structure is cheaper to code than a BSP-tree. We chose the kd-tree over the BSP-tree in a trade-off between degrees of freedom and data overhead, and over the octree for flexibility reasons, since the octree requires cutting along each dimension iteratively, while the kd-tree allows two or more consecutive cuts along the same dimension.

The kd-tree structure is built by recursively splitting the cells in two, starting at the bounding box of the mesh. The data-driven splitting algorithm, the main contribution of our work, is described in section 3.1. Then, in a bottom to top pass, connectivity data is updated as described in [11]. During the same pass, the associated properties data are also updated, with each property of the parent cell being the average of the corresponding properties of the child cells. The

property average is computed through an average function associated to each property type, as described in section 3.2. In a third pass, from top to bottom, the data is encoded. Along with the geometry and connectivity data enconding achieved in [11], we encode the cutting plane as described in section 3.3. The associated properties data are encoded using delta encoding.

3.1 Data-driven cell splitting

At each cell split we want to subdivide a set of vertices into two sets through a plane orthogonal to an axis of the reference frame. We define a discrete set of equidistant planes orthogonal to each dimension of the cell, and we compute a saliency value for each of them. The saliency value represents the overall average weighted distance between the properties associated to the vertices on each side of the plane. We pick the plane with the highest saliency value.

Local Voxelisation The splitting process is represented in Figure 1 for a 2D cell. We first voxelise the entire cell as a pre-processing step. The boundaries of the voxels are defined by the discrete set of planes for the cell, as detailed in section 3.1. The properties of the vertices inside each voxel are averaged so that each voxel is represented by only one instance of each property. Then, for each plane, each voxel in the plane neighborhood is evaluated with its symmetric voxel with respect to the plane, hence computing the distance of every property for every pair of voxels. The saliency value s of a plane is given by the average of the distance values d_{i,p_j} for P properties p_0 to p_P of the N voxels on each side of its neighborhood:

$$s = \frac{1}{P}\sum_{j=0}^{P}(\frac{1}{N}\sum_{i=0}^{N} d_{i,p_j}) \tag{1}$$

This technique allows a local analysis of data, which leads to a local vertex segmentation driven by the associated properties. The plane with the highest saliency value is the one that captures the most visible property difference according to the distance criterion. Figure 2 compares the first two cuts of the data-driven and the non data-driven compression of an example model with a simple color distribution.

Adaptive Precision When a cell is voxelised, the number of planes of the discrete set on each dimension defines the number of voxels in the cell. The size of a voxel and the efficiency of the split are directly related to the number of planes. This number is computed for each cell, and for each dimension, following 2 criteria:

The cell size. For a constant number of planes, the bigger the cell the larger the separation between planes, and hence the lower the accuracy of the split.

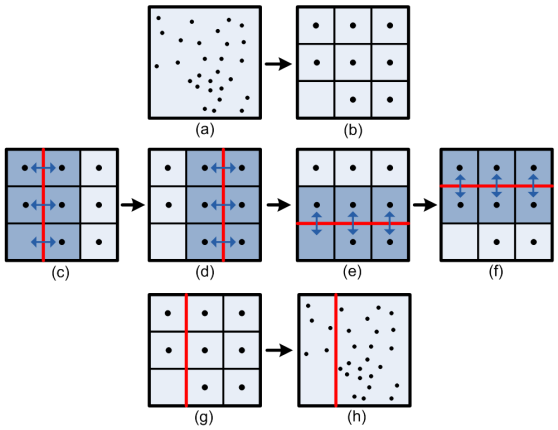


Fig. 1. The cell (a) is voxelised (b) according to the discrete planes, and each voxel averages the properties of the vertices it contains. The algorithm iterates through the discrete planes ((c), (d), (e), (f)), computing the property distances for every pair of symmetric voxels. The saliency value of the plane is the weighted average of the average property distances. The plane with the highest saliency value is selected (g), and the cell is split according to the selected plane (h).

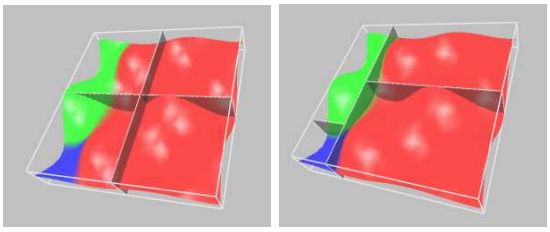


Fig. 2. Illustration of the first two cuts (4 cells) of a simple model for the non data-driven (left) and the data-driven (right) versions of the algorithm. The non data-driven cuts subdvide the parent cell into 2 equal children cells, while the data-driven cuts follow the color borders.

A bigger cell requires a larger discrete set of planes. If l_c is the length of the longest side of a cell C and l_{bb} is the length of the corresponding side of the mesh bounding box, the size criterion S is given by $S = \frac{l_c}{l_{bb}}$.

The number of vertices of the cell. The higher the number of vertices in a cell, the larger the number of planes should be in order to accurately separate the set of vertices in two. Hence, we need to quantize the surplus or the lack of vertices for a given cell volume. For that, we normalize the number of vertices with the expected number of vertices of the cell, which is the number of vertices the cell would have if the vertices were homogeneously distributed inside the bounding box. If V_c is the volume of C, V_{bb} is the volume of the bounding box,

P_c is the number of vertices of C and P_{bb} is the total number of vertices, the ratio criteron R is given by $R = \frac{P_c * V_{bb}}{V_c * P_{bb}}$.

The number of discrete planes N_c of a cell C is therefore given by the following formula:

$$N_c = N_{max} * S^{\alpha} * R^{\beta}$$

where N_{max} is the maximum number of planes set by the user. We choose $\alpha = \beta = \frac{1}{2}$ to obtain a stagnating behavior of N_c. N_c is bounded by 1 and N_{max} in order to avoid particular cases of extreme out-of-range values.

Special Cases Some special cases deserve special attention and fall back to a different subdivision algorithm: the middle cut. The middle cut of a cell is the cut closest to the center of the cell, along the dimension with the largest cell size. This cut is not data-driven. The method falls back to this cut when a data-driven cut is not justified or not necessary. A cut through the middle of the cell is then a good compromise, since it usually does not need to be encoded, as we will see in the next section.

These special cases occur when either:

- the saliency value of the selected discrete plane is lower than a minimum threshold. The algorithm falls back to the middle cut since no discrete plane is good enough.
- the number of discrete planes on each dimension of the cell is equal to 1. The voxelisation of the cell produces a single voxel, and no cutting plane can be chosen with this technique.
- the number of vertices inside the cell is equal to one. Although we could select the cutting plane closest to the vertex in order to give the non-empty child cell a higher precision, the cost of coding the cutting plane will be superior to letting a series of middle cuts increase the precision by dichotomy.

3.2 Distance and average functions

For each associated property, we need to define a distance and an average function. The distance function is used in the subdivision process, while the average function is used in the encoding process.

Distance function The distance function compares two instances of a property and returns a positive value representing a difference between them. It is used in Equation 1 to compute the plane saliency. For the color property, we can define the distance between two colors as their Euclidean distance in the Luv space, as proposed in [24]. The Luv space is better suited for human perception of colors than the RGB space, since distances between colors in the Luv space are closer to distances perceived by a human eye than their corresponding RGB space distances. Moreover, the Luv space decorrelates color information redundant in the

RGB space, which improves coding efficiency.

Average function The average function should return a property instance that closely represents both input instances. For the color property, an appropriate average function is the 3-tuple of the averages of each Luv space component.

3.3 Plane Coding

Two extra values need to be encoded for each non-empty cell in order to recover the cutting planes during the decompression process: the dimension to which the cutting plane is orthogonal to, and the index of the cutting plane among the discrete set of planes of that dimension of the cell. The dimension is arithmetically encoded, using the parent and the grand-parent cutting planes dimension as context. This reduces the cost of coding the dimension by around 60%. Since the number of discrete planes N_c is cell dependent and can be computed at decompression time, the index of the cutting plane is arithmetically encoded with N_c as context.

The middle cuts usually do not need to be encoded, since the middle of the cell and the dimension of its largest side can be computed at decompression time. However, for the special case where the saliency value of the plane is lower than the minimum threshold, the plane data does need to be encoded since saliency values are not known during decompression.

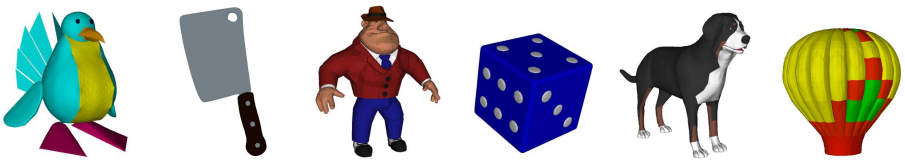
4 Results

In the following section we quantitatively and qualitatively evaluate the compression ratios and decompression qualities of sets of colored meshes. We use the color mesh property to drive the compression process with the color distance and average functions previously described.

We quantitatively evaluate the geometric quality of the intermediate levels of detail of several 3D models using the METRO [8] rate/distortion evaluation tool. However, this geometric distance does not reflect the perceptual visual quality of the meshes, which depends in part on the color property. Therefore, we also carried perceptive evaluations through subjective experiments with human subjects (experts and non experts) in order to compare the results of our algorithm with its non data-driven version. The non data-driven version splits cells at their center, as in [11].

A first set of colored meshes was chosen among the original meshes of the Princeton Shape Benchmark repository [21]. We selected a set of 8 heterogeneous models that we used throughout our quantitative tests and subjective experiments. Some of the models are shown in Figure 3. The number of vertices ranged from 4000 to 17000.

A second set of 8 colored meshes was provided by EDF R&D (Energie de France Research and Development) division. The meshes are examples of CAD



(a) (b) (c) (d) (e) (f)

Fig. 3. Some models from the Princeton Shape Benchmark repository data set: birdy (a), cleaver (b), gangster (c), dice (d), dog (e), montgolf (f)

machinery parts and scientific computation outputs of thermal flows. Unfortunately, this data set is confidential, with the exception of the radiator sample shown in Figure 4. The number of vertices of the meshes of this data set ranges from 26000 to 322000. This dataset was used during a perceptive evaluation performed by EDF R&D experts.

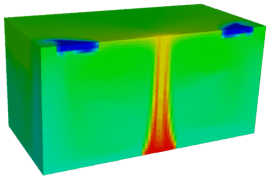


Fig. 4. The radiator, from the EDF R&D data set

4.1 Rate/Distortion

In order to evaluate the geometry quality of the intermediate meshes, we computed distortion values at several bitrates. The geometry distortion value for a particular bitrate is given by the Hausdorff distance between the original (lossless) mesh and the intermediate mesh at that particular bitrate. We used the METRO [8] tool for the computations. Results are given in Figure 5, which show the rate-distortion (R-D) curves of the data-driven and the non data-driven compression of 3 different models of the first set of meshes.

At low bitrates, the data-driven method performs worse than the non data-driven version, in terms of geometric distortion. This behaviour was expected, since the algorithm is optimized for higher associated properties quality, while the METRO tool judges geometry quality. At higher bitrates, we can notice that both versions of the meshes are roughly equivalent in geometry. It is interesting to point out that the distortion difference remains small for the tested dataset, meaning that the gain on the associated properties side is achieved at a small cost. Since a purely geometric distance is not well suited to evaluate the visual

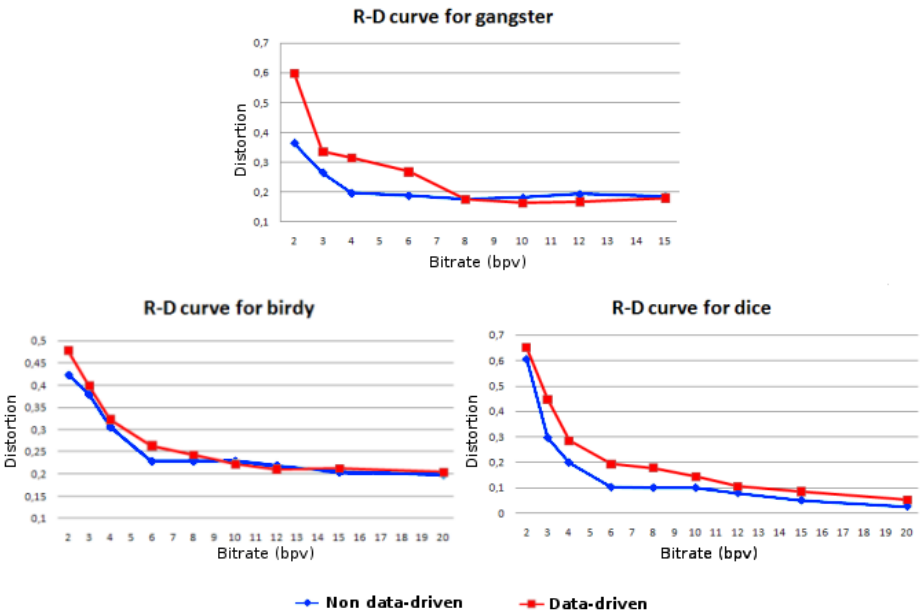


Fig. 5. R-D curves for 3 of the evaluated meshes (birdy, dice and gangster). Abscissa: bitrates (in bits per vertex). Ordinates: Hausdorff distances.

quality of a mesh, particularly for colored meshes where color may carry an important part of the visual information, we proceeded to human perceptive evaluations through subjective experiments.

4.2 Perceptive Evaluation

In order to demonstrate that our data-driven method increases the visual quality of the intermediate meshes compared to its non data-driven counterpart, we have conducted two subjective experiments using human observers. These two experiments share the same evaluation protocol. However, the data sets and the expertise of the human observers are different.

Evaluation protocol The evaluation protocol was as follows: we presented to each evaluator 4 intermediate meshes at a time, together with the original lossless model. The intermediate meshes were composed of 2 data-driven intermediate meshes and their corresponding non data-driven version of the same bitrates, in random positions. The evaluator could rotate, scale and translate the models. A transformation applied to one of the meshes was also automatically applied to all the other meshes on the screen, in order to keep a consistent point of view. The evaluator was asked to provide a score for each object, reflecting the degree of perceived visual similarity with the original, from 1 (no similarity) to 6

(highest similarity). The Mean Opinion Score (MOS) is then computed for each intermediate mesh of each model of each corpus:

$$MOS_i = \frac{1}{n} \sum_{j=1}^{n} m_{ij}$$

where MOS_i is the mean opinion score of the i^{th} object, n is the number of evaluators, and m_{ij} is the score ($\in [1, 6]$) given by the j^{th} evaluator to the i^{th} object.

For each model of the data sets, we evaluated 4 different bitrates, which required 8 intermediate meshes (4 data-driven, 4 non data-driven). Since only 4 intermediate meshes could be shown on the screen at a time for a confortable visualization, each model was fully evaluated in two independent steps: the low step, with the 2 lower bitrates, and the high step, with the other 2 bitrates. The high step immediately followed the low step.

Since we noticed that, at high resolutions, our algorithm performs roughly equally than the non data-driven version, we believed it was more interesting to compare the two algorithms at low bitrates. The color property was compressed with 9, 6 and 6 bits per color component (L, u and v) respectively. Geometry was compressed with 14 bits per coordinate.

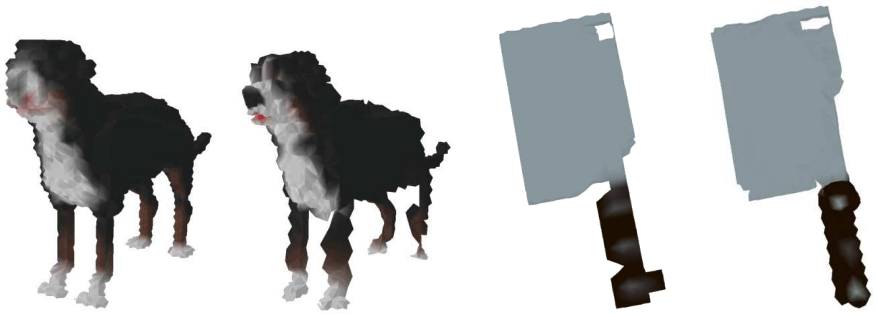
First experiment: Princeton Shape Benchmark data set. Eight color models from the Princeton Shape Benchmark repository data set (Figure 3) were evaluated by 12 human subjects, from the research staff of the LIRIS laboratory in Lyon, France, with different degrees of knowledge in the computational geometry and computer graphics field. For this experiment, we chose 4 fixed bitrates: 2, 3, 4 and 6 bits per vertex (bpv). Hence our corpus was composed of 64 objects (8 models * (4 data driven version + 4 non-data-driven version)). Table 1 details the Mean Opinion Scores (MOS) for the models of the dataset, according to the bitrate and the method (data-driven vs. non data-driven).

The results show a positive evaluation for most of the meshes of the set, where the data-driven algorithm is evaluated from roughly equal (no MOS difference) to considerably better (more than 1 point in MOS difference) than its non data-driven counterpart. Clear examples are the dog and cleaver (fig. 6) models, where different color regions can be distinguished from an early stage in the data-driven version, hence the overall visual quality is clearly improved even if the geometric distorsion is higher than for the non-data-driven version. However, for some models (for instance montgolf) the geometric distortion breaks the overall shape of the model, and therefore the data-driven version has lower MOS than its non data-driven couterpart, regardless of the color visual quality of the data-driven model.

For many of the models, the MOS at a particular bitrate with the data-driven algorithm was equal or higher than the MOS of the next (higher) bitrate with the non data-driven algorithm, like with bitrates 4 and 6 of the dog model, for instance. In such cases, both the compression ratio and the visual quality are improved by our method.

	Non data-driven				Data-driven			
	2	3	4	6	2	3	4	6
dog	2.8	4.1	4	4.4	3	4.2	4.6	5.2
cleaver	4.1	4.5	4.1	4.3	4.5	4.6	4.6	5
montgolf	3.5	4	3.6	4	2.3	3	2.8	3.7
tuna	3.7	4.1	4.4	5.1	4.3	4.9	5	5.2
dice	2.3	2.7	3	3.5	2.9	3.3	3.4	3.8
gangster	3	3.5	3.4	3.9	3.3	3.4	3.7	4.1
chair	3.3	3.6	4	4.2	3.1	3.5	3.8	4
birdy	3.5	3.6	4.3	4.5	4	4.2	4.9	5
Average	**3.28**	**3.76**	**3.85**	**4.24**	**3.43**	**3.89**	**4.1**	**4.5**

Table 1. Mean Opinion Scores for the models of the Princeton Shape Benchmark data set, at all tested bitrates, for the data-driven and the non data-driven algorithms



(a) (b)

Fig. 6. The non data-driven (left) and the data-driven (right) intermediate meshes for the dog model at 3 bpv (a) and cleaver model at 2 bpv (b)

It is interesting to see that, even if our data-driving algorithm exhibits a rate-distorsion compromise slightly worse than the non data-driven one, on the contrary the rate-quality compromise is greatly improved.

Second experiment: EDF data set. Five color models from EDF R&D Clamart were evaluated by 7 EDF physicists and 3D experts. There were considerable differences in the number of polygons among the models of this dataset. Therefore, instead of using fixed bitrates for all the models, we chose bitrates corresponding to 5, 10, 15 and 20% of the file size of the non data-driven lossless compressed model. Resolutions under 5% were usually too coarse for any visual evaluation, and resolutions higher than 20% were less interesting due to the close ressemblance between the results of both algorithms. For this experiment, we included the normal data in the compression process, but only as encoded data, not as a driving property. This produced intermediate meshes of a higher rendering quality, for both algorithms.

The data-driven algorithm produced higher quality intermediate meshes for 3 of the 5 models (higher MOS), as shown in Table 2. The models where the data-driven algorithm was rated lowerer than its non data-driven counterpart were the radiator and a second model where color regions were hard to distinguish even in its lossless version. For the case of the radiator, the compressed version presented an important visual distortion, due to its rough parallelepipedal shape, with any break in the continuity of the contours being easily noticeable, decreasing distortion tolerance. Figure 7 compares the result of the algorithms for a bitrate where the data-driven algorithm was given a higher grade.

	Non data-driven				Data-driven			
	5%	10%	15%	20%	5%	10%	15%	20%
1	2.8	3.4	3.9	4.3	2.3	3.2	3.8	4.3
radiator	2.2	3.7	2.2	4	1.3	2.5	2.5	3.5
3	2.8	3.1	3	3.4	3.3	3.7	3.8	4
4	2.6	2.7	3	3.3	2.3	2.5	3	3.6
5	3.1	3.4	3.5	4.2	3.4	3.6	3.6	4.2
Average	**2.7**	**3.26**	**3.12**	**3.84**	**2.52**	**3.1**	**3.34**	**3.92**

Table 2. Mean Opinion Scores for the models of the EDF data set, at all tested bitrates, for the data-driven and the non data-driven algorithm

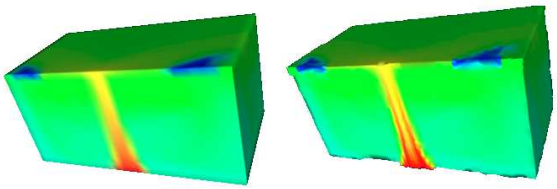


Fig. 7. The non data-driven (left) and the data-driven (right) intermediate meshes for the radiator model at 6 bpv (15%)

An interesting remark can be made based on the results of the evaluation. Two of the evaluators systematically gave a lower grade to the data-driven algorithm, for every model at every evaluated bitrate. Even in situations where all of the other evaluators gave a significant advantage to the data-driven algorithm, these 2 evaluators graded it lower. We could therefore think that some individuals are or become more sensitive to geometry distortion than others.

These two subjective experiments have shown that driving the compression with attributes like color can improve the rate-quality compromise, whereas producing a slighty higher geometric distorsion. Of course this improvement depends on the geometric structure of the object and on the importance of the attributes

in its visual appearance: we have noticed that it is more interesting to use our compression technique on objects where the important data is color, instead of meshes where the geometry-color tradeoff can dramatically deteriorate the overall visual quality.

4.3 Other properties

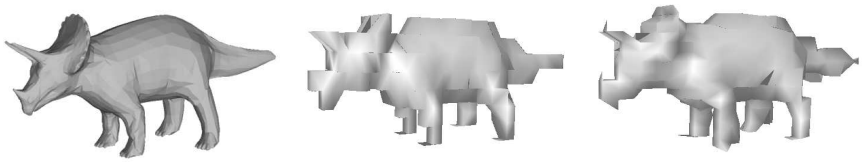
Due to the generic nature of the framework, other vertex-bound properties can be used to drive the compression process. However, one can use properties that are not necessarily explicitely defined, but rather computed from the mesh data. These intrinsic properties, such as curvature values, can then provide relevant information on the mesh structure for the compression process. Typically, these properties can be used to drive the algorithm, but are not encoded as an associated property.

To demonstrate this genericity, we have conducted a short experiment of data-driving with curvature values. The curvature tensors are computed for each vertex using Cohen-Steiner and Morvan [10] algorithm with a geodesic neighborhood [1]. The principal curvature value Kmax is then extracted, and used as a property value for each vertex. The information provided by curvature tensors can be used to separate regions whose borders show a highly different shape, increasing the geometric visual quality of the mesh during decompression.

Figure 8 shows an example of data-driving with curvature values. The distance value d for the curvature property, where C_1 and C_2 are two curvature values, is defined as

$$d(C_1, C_2) = |C_1 - C_2|$$

At 2 bpv, the data-driven version of the triceratops shows a closer ressemblance to the lossless version than its non data-driven couterpart, mainly on the head region (mouth, horns).



(a) (b) (c)

Fig. 8. The lossless version of the triceratops model (a), with its non data-driven (b) and data-driven (c) intermediate meshes at 2 bpv

5 Conclusion

We have presented a framework for the data-driven progressive compression of meshes. It allows the use of any associated property, and the behavior of the

compression process can be customized through a single distance function per property. We showed an example of color data-driving and compression of the color property. At equal bitrates, it improves the visual quality of the decompressed meshes compared to the non data-driven version. This algorithm is well suited for the compression of meshes where geometry and topology play a secondary role compared to associated properties, such as with scientific visualization models.

This work sets the bases for further research around the concept of data-driven compression. The performance of the algorithm relies in part on efficient distance functions. Other properties will be considered for testing, such as normals, as well as more complex and efficient distance functions based on intrinsic properties such as connectivity data, in order to improve the geometric shape of the intermediate meshes. Moreover, a BSP-tree could be used to allow more flexible cuts. However, special attention would have to be given to plane coding and overall processing time.

Acknowledgments. This work has been supported by French National Research Agency (ANR) through SCOS project (ANR-06-TLOG-029) and COSI-NUS program (project COLLAVIZ ANR-08-COSI-003). The authors would like to thank R. Marc and C. Mouton from EDF for their help regarding the perceptive evaluation.

References

1. Alliez, P., Cohen-Steiner, D., Devillers, O., Lvy, B., Desbrun, M.: Anisotropic polygonal remeshing. ACM Transactions on Graphics 22(3), 485–493 (2003)
2. Alliez, P., Desbrun, M.: Valence-Driven connectivity encoding for 3D meshes. Computer Graphics Forum 20, 480–489 (2001)
3. Alliez, P., Gotsman, C.: Recent advances in compression of 3D meshes. In: Proceedings of the Symposium on Multiresolution in Geometric Modeling. pp. 26, 3 (2003)
4. Bajaj, C.L., Pascucci, V., Zhuang, G.: Progressive compression and transmission of arbitrary triangular meshes. In: Proceedings of the IEEE Visualization Conference. IEEE Computer Society (1999)
5. Bass, A., Been, K.: Progressive compression of normal vectors. In: Proceedings of the International Symposium on 3D Data Processing, Visualization, and Transmission. pp. 1010–1017. IEEE Computer Society (2006)
6. Bordignon, A., Lewiner, T., Lopes, H., Tavares, G., Castro, R.: Point set compression through BSP quantization. In: Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing. pp. 229–238 (2006)
7. Botsch, M., Wiratanaya, A., Kobbelt, L.: Efficient high quality rendering of point sampled geometry. In: Proceedings of the Eurographics workshop on Rendering. pp. 53–64. Eurographics Association, Pisa, Italy (2002)
8. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Computer Graphics Forum 17, 167–174 (1998)
9. Cohen-Or, D., Levin, D., Remez, O.: Progressive compression of arbitrary triangular meshes. In: Proceedings of the conference on Visualization. pp. 67–72. IEEE Computer Society Press, San Francisco, California, United States (1999)

10. Cohen-Steiner, D., Morvan, J.: Restricted delaunay triangulations and normal cycle. In: Proceedings of the annual symposium on Computational geometry. pp. 312–321. ACM, San Diego, California, USA (2003)
11. Gandoin, P., Devillers, O.: Progressive lossless compression of arbitrary simplicial complexes. In: SIGGRAPH. pp. 372–379. ACM, San Antonio, Texas (2002)
12. Hoppe, H.: Progressive meshes. In: SIGGRAPH. pp. 99–108. ACM (1996)
13. Huang, Y., Peng, J., Kuo, C.C.J., Gopi, M.: A generic scheme for progressive point cloud coding. IEEE Transactions on Visualization and Computer Graphics 14(2), 440–453 (2008)
14. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: SIGGRAPH. pp. 279–286. ACM Press/Addison-Wesley Publishing Co. (2000)
15. Khodakovsky, A., Guskov, I.: Normal mesh compression. In: Geometric Modeling for Scientific Visualization (2002)
16. Khodakovsky, A., Schröder, P., Sweldens, W.: Progressive geometry compression. In: SIGGRAPH. pp. 271–278. ACM Press/Addison-Wesley Publishing Co. (2000)
17. Pajarola, R., Rossignac, J.: Compressed progressive meshes. IEEE Transactions on Visualization and Computer Graphics 6(1), 79–93 (2000)
18. Peng, J., Eckstein, I., Kuo, C.J.: A novel and efficient progressive lossless mesh coder. In: SIGGRAPH Sketches. p. 180. ACM, Boston, Massachusetts (2006)
19. Peng, J., Kim, C., Kuo, C.J.: Technologies for 3D mesh compression: A survey. Journal of Visual Communication and Image Representation 16(6), 688–733 (Dec 2005)
20. Peng, J., Kuo, C.J.: Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. ACM Transactions on Graphics 24(3), 609–616 (2005)
21. Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The princeton shape benchmark. In: Proceedings of the Shape Modeling International. pp. 167–178. IEEE Computer Society (2004)
22. Taubin, G., Horn, W., Lazarus, F., Rossignac, J.: Geometry coding and VRML. Proceedings of the IEEE 86(6), 1228–1243 (1998)
23. Taubin, G., Rossignac, J.: Geometric compression through topological surgery. ACM Transactions on Graphics 17(2), 84–115 (1998)
24. Waschbusch, M., Gross, M., Eberhard, F., Lamboray, E., Wurmlin, S.: Progressive compression of Point-Sampled models. Proceedings of the Eurographics Symposium on Point Based Graphics pp. 95–102 (2004)