Progressive compression of generic surface meshes

Florian Caillaud, Vincent Vidal, Florent Dupont, Guillaume Lavoué

CGI2015 manuscript No.
(will be inserted by the editor)

Progressive compression of generic surface meshes

F. Caillaud[1,2] · **V. Vidal**[1,3] · **F.Dupont**[1,3] · **G. Lavoué**[1,2]

Abstract This paper presents a progressive compression method for generic surface meshes (non-manifold and/or polygonal). Two major contributions are proposed : (1) generic edge collapse and vertex split operators allowing surface simplification and refinement of a mesh, whatever its connectivity; (2) a distortion-aware collapse clustering strategy that adapts the decimation granularity in order to optimize the rate-distortion tradeoff.

Keywords Progressive compression, surface mesh, non-manifold, polygonal.

1 Introduction

The amount of data describing 3D models does not stop increasing even though several applications (e.g. mobile device visualization, video games) are constrained by memory and/or processing speed. This situation may lead to communication or visualization latencies. The progressive compression of 3D models is a possible solution to this latency problem. The principle is to compress the 3D mesh while enabling its progressive decompression in the form of several Levels of Detail (LoD). In contrast of the so-called single-rate compression, progressive compression allows to quickly visualize a draft of the 3D model and then to refine it until the original mesh is retrieved. This offers more comfort to users and, sometimes, a quality control of the intermediate meshes.

Existing methods for progressive compression have mostly focused on 2-manifold and triangular meshes. Only few methods are able to progressively compress non-manifold surface meshes and none provide a generic compression of surface meshes, whatever their manifoldness or their faces degree. In this context, we present a lossless progressive compression method suited for non-manifold and/or polygonal meshes. Our approach is inspired by the seminal work from Popović and Hoppe [9] and introduce generic edge collapse and vertex split operators. The obvious additional cost of this genericity is balanced by an efficient local control of the distortion, which leads to excellent results at low and medium bitrates.

The rest of this paper is organized as follows : we introduce the previous work in Section 2. Then, our approach and its features are described in Section 3. Finally, Section 4 presents our results as well as comparison with the state of the art.

2 Previous work

Progressive compression of 3D models has been introduced by Hoppe [3]. He proposed a simplification operator : the edge collapse. The main drawback of his method is its very fine granularity (difference between two consecutive LoDs) which allows a strong control of the distortion but penalizes the compression rate. Moreover, like most of further progressive compression approaches [1,6,4,10], it only deals with *triangular manifold* meshes.

Only few methods are able to compress *polygonal* manifold meshes. Maglo *et al.* [5] presents a generalization of the valence-based algorithm from Alliez and Desbrun [1] for this task.

Toward the progressive compression of *non-manifold* meshes, almost nothing exists in the literature. Popović and Hoppe [9] adapted [3] for this purpose. We particularly set apart tree-based methods from Gandoin and

[1]Université de Lyon, CNRS
[2]INSA-Lyon, LIRIS UMR 5205
[3]Université Lyon 1

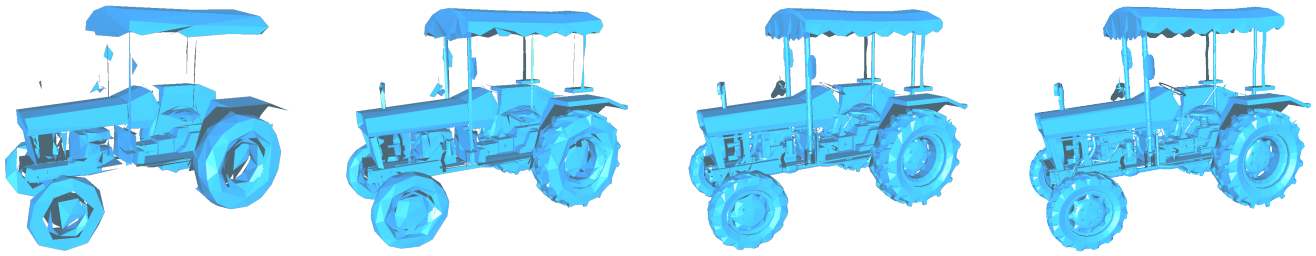

Fig. 1 Progressive decompression of the *Tractor* model (polygonal non-manifold) at 3, 5, 10 bits per vertex and lossless.

Devillers [2] and Peng and Kuo [8], which are very efficient at lossless rates but generate undesired quantization effects at low and medium bitrates. These techniques [2,8] handle non-manifold meshes but do not allow any local control of the decimation and thus of the distortion. Using a different bottom-up clustering strategy, Peng et al. [7] propose a progressive approach which provides nice low bitrate results however it does not guarantee to retrieve the lossless version. Furthermore, these techniques [9,2,8,7] are limited to triangular meshes.

In comparison, our method (1) handles both polygonal and non-manifold meshes, (2) allows a lossless retrieval of the mesh and (3) allows a precise local control of the decimation leading to excellent rate-distortion performance at low and medium bitrates.

3 Generic progressive compression

3.1 Compression pipeline

We describe, in this section, our progressive compression pipeline handling polygonal non-manifold meshes, i.e. every surface meshes.

We start by quantizing the mesh coordinates into Q bits. Then the mesh is simplified iteratively and the removed geometry and connectivity data are encoded into a compressed stream. This stream is structured so as to permit the progressive decompression and reconstruction of the original 3D model.

The simplification step is realized with a set of edge collapses (cf. Section 3.3.1) grouped in waves (cf. Section 3.3.2) similarly to [10]. Each wave generates a LoD of the 3D model. The best set of edges to be removed at each iteration is selected using a metric (cf. Section 3.2) and its size (i.e. the number of edges to collapse) is automatically chosen. During this simplification, different types of data will be generated. Each type is encoded using a specific arithmetic coder.

3.2 Edge selection

As explained above, each simplification step decimates the mesh by a set of edge collapses. The edges to be collapsed are selected using a priority queue ordered by a weight w_e associated to each edge e. The priority queue is updated after each collapse. Our idea here is to (1) select edge collapses that produce the less geometric error, (2) minimize the creation of dangling edges (i.e. edges without incident faces) because they cause losses of areas and (3) minimize the collapses of dangling edges because they may lead to a drastic reduction of the bounding box. For that purpose, if e is a dangling edge (case 3 above), $w_e = diag + length(e)$, if the collapse of e creates a dangling edge (case 2), $w_e = 2 \times diag + area(e)$, otherwise (case 1), $w_e = d_H(e)$. $diag$ is the bounding box diagonal length, $area(e)$ is the area loss ratio caused by the hypothetical collapse of e and $d_H(e)$ is the symmetric Hausdorff distance between the mesh before and after the hypothetical collapse. This way, we give priority to case 1, then 2, then 3.

The number of edges to be collapsed in each wave is critical. Indeed, if we select a very high number, then a large distortion could be introduced. To adapt this wave size, at the beginning of a wave we compute the weights of all edges. Then we determine a threshold as the average weight of these edges. Edges with weights superior than this threshold are not collapsed in this wave. This simple yet efficient rule provides excellent rate-distortion tradeoffs.

In order to avoid conflict between two collapses during the same wave, edges included in the patch (set of incident faces) of an already selected edge, during the current wave, cannot be selected afterwards.

3.3 Progressive encoding

3.3.1 Generic Edge Collapse

The mesh simplification (resp. refinement) is realized by our edge collapse (resp. vertex split) operator on each selected edge e. We introduce a generalisation of these

operators to handle polygonal non-manifold local configurations. The edge collapse starts by removing the triangular incident faces of e (Figure 2-a). It removes

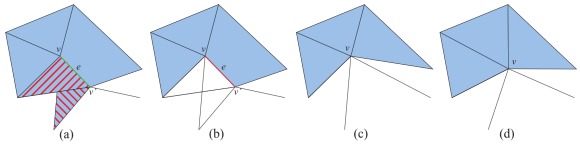

Fig. 2 Different steps of the generic edge collapse of an edge e.

e (Figure 2-b), which reduces the degree of polygonal incident faces of e. The operator then merges v and v' (incident vertices of the removed edge e) in v without forgetting to solve the now duplicated edges and faces around v (Figure 2-c). Finally, v is displaced to the middle of the removed edge e.This new position is chosen as we only need one displacement vector to deduct the previous position of v and v' (Figure 2-c).

3.3.2 Walk on mesh

Each simplification wave is decomposed in two parts. Firstly, edges are collapsed, in the order of the priority queue, until the weight threshold is reached. After each collapse, the priority queue is updated. Secondly, after having collapsed all the edges of the current wave, a spanning tree is build over each connected component of the mesh, using a lexicographic order. Each spanning tree starts on a fixed vertex. This vertex is randomly chosen once on the original mesh since this choice does not affect the simplification. Each fixed vertex stays during all the simplification and, therefore, cannot be removed (its incident edges can be collapsed anyway). This walk on the mesh provides an ordering of the edge collapses, thus an ordering of the vertex splits during refinement, for each simplification wave.

This iteration is repeated until each connected component is compound by only one vertex. This vertex is the fixed vertex of the corresponding connected component. This wave strategy avoid us to specify explicitly the edges to collapse (in contrast to [9]) , which drastically reduce the generated information.

3.3.3 Connectivity and geometry encoding

In order to provides the needed information for the mesh reconstruction, during each edge collapse, connectivity and geometry information will be generated.

Firstly, the connectivity around e (i.e. its adjacent faces and edges) is represented according to a generalization of the connectivity cases from [9] (Figure 3). Actually, the different configurations are coded the same way (except for polygonal case 2.3) using the same topological constraints to reduce redundancy.

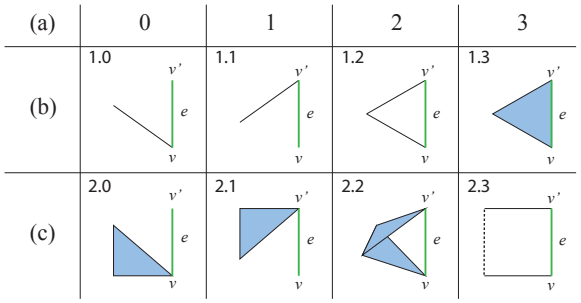

Fig. 3 Possible configurations around an edge e. (a) shows the generated codes, (b) edges configurations and (c) faces configurations.

Secondly, in order to preserve the face orientations at reconstruction, if they cannot be determined, we encode the orientation of the removed faces using one bit.

Thirdly, in order to efficiently encode the displacement vector of v, we use its representation into a local Frenet frame. This representation is not especially optimal for local non-manifold configurations, but considering that most configurations are locally manifold, it still remains efficient.

Finally, during each simplification step, the spanning tree(s) allows to order this information so that the decompression has the corresponding data for each vertex to split. Obviously, we also need to specify which vertices will be split along the tree(s). For this purpose, during the construction, we encode a 1 if the visited vertex is resulting of an edge collapse, otherwise we encode a 0. As we forbid edge collapse conflict, we do not encode a 0 if, at this state, we already encode a 1 for an adjacent vertex.

4 Results

We compare our generic progressive compression algorithm (GPC) with [8] (PK05) and [7] (PHK10).

Table 1 shows the mean error given by METRO for different meshes of various complexity (both manifold and non-manifold). The data given by GPC are compared with the available data of PK05 and PHK10 (computed from the models provided by the authors). "N/A" values are due to early termination.

Figure 4 describes rate-distortion curves obtained with GPC, PK05 and PHK10 for the Horse and Rabbit models (triangular manifold) using MRMS metric. Despite a bit overhead due to the genericity of our method,

Mesh(#v)	Meth.	1.0	2.0	4.0	8.0	12.0	16.0
Horse (19,851)	GPC	20.2	9.3	4.1	1.6	1.0	0.5
	PK05	31.9	19.4	10.2	3.3	0.9	N/A
	PHK10	19.0	12.8	5.1	3.3	1.6	1.1
Rabbit (67,039)	GPC	5.3	2.4	1.1	0.6	0.3	0.2
	PK05	18.9	10.2	8.3	2.2	0.6	N/A
	PHK10	5.5	4.2	2.1	1.7	1.0	0.5
Tractor (27,251)	GPC	31.2	26.3	14.5	6.2	3.8	2.0
Skeleton (6,308)	GPC	19.0	11.9	10.6	7.3	5.5	3.8

Table 1 Table of comparison between GPC, PK05 and PHK10 for different bitrates. The mean error values are scaled by 10^4.

the strong distortion control of GPC allows to obtain very competitive results at low and medium bitrates.

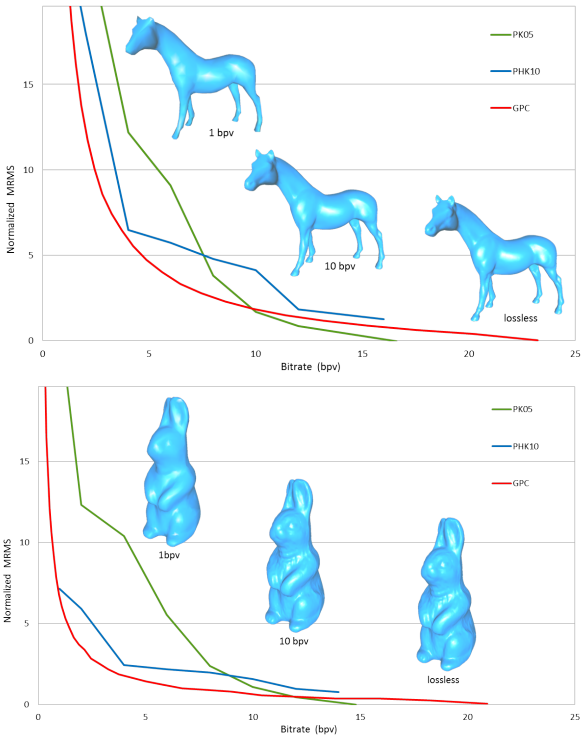

Fig. 4 Rate-distortion curves of Horse and Rabbit measured with MRMS metric. The MRMS values are scaled by 10^4.

We show, in Figure 1, different LoDs of the Tractor model to demonstrate the benefit of our edge selection strategy. Furthermore, we visually compare three LoDs of the Skeleton model obtained with GPC, PK05 and PHK10 at the same bitrate (Figure 5).

5 Conclusion

We have proposed a generic method for the progressive compression of arbitrary surface meshes. This genericity involves an additional cost in the case of lossless compression. However, the strong distortion control allowed by our edge selection strategy makes our algorithm competitive at low and medium bitrates. As future works, we plan to investigate a more *perceptual* distortion control. We also plan to integrate progressive texture encoding in our pipeline.

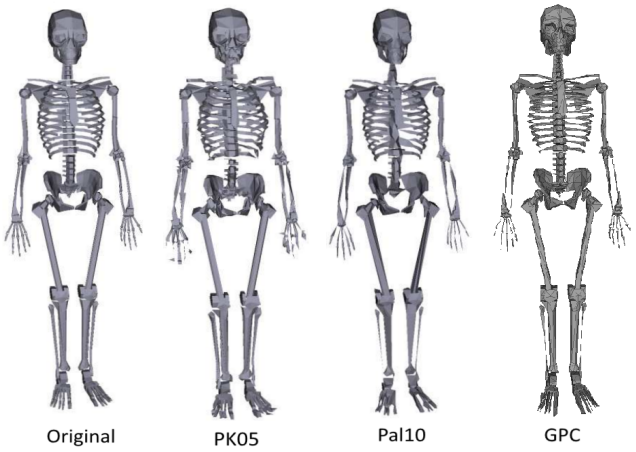

Fig. 5 LoDs of the *Skeleton* model at 8 bpv using PK05, PHK10 and GPC compared to the original.

References

1. Alliez, P., Desbrun, M.: Progressive compression for lossless transmission of triangle meshes. In: ACM SIGGRAPH, pp. 195–202 (2001)
2. Gandoin, P.M., Devillers, O.: Progressive lossless compression of arbitrary simplicial complexes. In: ACM SIGGRAPH, pp. 372–379 (2002)
3. Hoppe, H.: Progressive meshes. In: ACM SIGGRAPH, pp. 79–93 (1996)
4. Karni, Z., Bogomjakov, A., Gotsman, C.: Efficient compression and rendering of multi-resolution meshes. In: IEEE Visualization, pp. 347–354 (2002)
5. Maglo, A., Courbet, C., Alliez, P., Hudelot, C.: Progressive compression of manifold polygon meshes. Computers & Graphics **36**(5), 349–359 (2012)
6. Pajarola, R., Rossignac, J.: Compressed progressive meshes. IEEE Transactions on Visualization and Computer Graphics **6**(1), 79–93 (2000)
7. Peng, J., Huang, Y., Kuo, C.C.J., Eckstein, I., Gopi, M.: Feature oriented progressive lossless mesh coding. In: Computer Graphics Forum, vol. 29, pp. 2029–2038 (2010)
8. Peng, J., Kuo, C.C.J.: Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition. In: ACM SIGGRAPH, pp. 609–616 (2005)
9. Popović, J., Hoppe, H.: Progressive simplicial complexes. In: ACM SIGGRAPH, pp. 217–224 (1997)
10. Taubin, G., Guéziec, A., Horn, W., Lazarus, F.: Progressive forest split compression. In: ACM SIGGRAPH, pp. 123–132 (1998)