





View dependent decompression for web-based massive triangle meshes visualisation

A. Cecchin¹ , P. Du¹ , M. Pastor¹  and A. Agouzoul¹ 

¹ Arskan Corp, France

Abstract

We introduce a framework extending an existing progressive compression-decompression algorithm for 3D triangular meshes. First, a mesh is partitioned. Each resulting part is compressed, then joined with one of its neighbours. These steps are repeated following a binary tree of operations, until a single compressed mesh remains. Decompressing the mesh involves progressively performing those steps in reverse, per node, and locally, by selecting the branch of the tree to explore. This method creates a compact and lossless representation of the model that allows its progressive and local reconstruction. Previously unprocessable meshes can be visualized on the web and mobile devices using this technique.

CCS Concepts

• *Information systems* → *Data compression*; • *Computing methodologies* → *Mesh geometry models*;

1. Context

With the democratization of photogrammetry, the need to process and visualise larger triangular meshes has significantly increased. For example, the field of cultural heritage faces the challenge of displaying interactively large meshes online, as recently described by [BIS19]. As described by [PCDS18], one of the main challenges to web based visualisation is the huge amount of data to be processed.

Previously used algorithms, based on mesh traversal, can no longer be directly applied to such 3D models due to resource management and large data structures. In particular, progressive decompression was previously used to efficiently transfer large triangle meshes for web content [LCD13]. Its iterative aspect meant that level of details could be generated to visualise a simplified version of a model on performance constrained devices.

However, this algorithm relies on mesh traversal, using a half-edge data structure during the iterative compression and decompression. For a gigantic triangle mesh, containing more than 200 millions vertices, this structure cannot be generated in-core in its entirety, and not even out-of-core for web content on mobile devices. In addition, the decompression itself spreads the details over the whole model, regardless of the user's view. This means that even if only a part of the model is seen, the entire model still has to be decompressed to view the original details, which can be a waste of resources. Thus we want to extend this algorithm and circumvent those two limitations. In this context, we introduce a framework around this algorithm to allow a view dependent decompression for gigantic 3D meshes.

2. Methodology

2.1. Pre-processing

We start our framework by partitioning the original mesh in several submeshes that will fit in-core during the next steps. We achieve this by clustering our vertices and triangles out-of-core in a similar way to [HLK01]. We recursively process the resulting submeshes by compressing them with an algorithm similar to [AD01], and then stitching them two by two with their neighbour. This creates a binary tree of compressing plus stitching operations. At the end of this process, we obtain a basemesh, which is a coarse representation of the complete model assigned as the root of the tree, and a series of decompression and separation properties linked to each node of the graph.

2.2. Transfer

At the end of the pre-processing step, we can separate our data into two categories: global and local. The global information will only be downloaded once and contains the basemesh and the tree structure with a bounding volume and a quality criterion per node. The local information will be downloaded based on the selection at runtime, and only once per node. For each node, it contains the seed edge for the decimation conquests, encoded traversal attributes, and the separation attributes. Previous multiresolution approaches require that each submesh is encoded individually. In our case, each submesh is defined implicitly from the decompression and separation of its parent mesh. Thus we have no intermediary mesh structure to download, only a basemesh and the steps to refine it progressively based on our tree of operations.

	Vertex count	Triangle count	Pre-processing time (H:MM:SS)	Original file size (GB)	Compressed file size (GB)	Compression rate (bit/vertex)
Avignon Cathedral	986,396,965	1,973,010,686	1:37:24	45	3.1	27
Marigny Aqueduct	499,722,772	999,342,292	0:32:35	24	1.8	30
Underground Parking	312,709,538	625,443,996	0:22:54	15	1.0	27

Table 1: Model presentation and their corresponding results using our framework

2.3. Runtime

Our runtime computations can be separated in two processes.

The first one is a tree traversal inspired by [YSGM04] and [PD15]. The operation tree created during the preprocessing step is used as a multiresolution representation of the model and a spatial partition. The optimal nodes to decompress and their decompression level are selected according to their visibility depending on the camera frustum and occlusion state, a vertex limit, and the resulting decompressed mesh quality. This selection is computed whenever the camera moves. The vertex limit is chosen based on hardware limitation: lower on mobile devices and higher on high-end computers. The aim is a fluid visualisation, independently from the model.

The selected nodes are decompressed in a separate web worker with an implementation compiled in WebAssembly. Here, the tree is used as a decompression dependency. If a node was not seen previously, we check its parent status. If the parent is fully processed, we download the new node data, progressively decompress it, then if necessary separate it. Otherwise, we process the parent node.

Intermediary meshes are created during the progressive decompression and are re-used during visualisation. For every frame, the closest available decompressed nodes to the optimal view are rendered using WebGL. The progressive decompression allows us to see the model even if not fully decompressed, and the mesh gets refined overtime.

3. Results

We applied this methodology to three gigantic models, containing a color per point attribute. We present the processing time and compression rate in Table 1.



Figure 1: Render of the Avignon Cathedral with a 5M vertex limit.

Once the basemesh is downloaded, the visualisation is continuous and crackless. Even if a targeted node has not been decompressed yet, we can still render its parent or closest available node.

We measured the decompression speed of our framework on three different devices: it takes a mean time of 3.3 seconds on PC with an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 7.7 seconds on a smartphone with a Snapdragon 765G (7nm) chipset and 17.2 seconds on a smartphone with an Exynos 7420 Octa (14nm) chipset.

4. Conclusions

We have introduced a framework for the interactive visualisation of gigantic triangular meshes. Our representation is compact by transferring only a simplified basemesh and the necessary steps to refine it hierarchically. No intermediate mesh is downloaded during the visualisation: each node data is downloaded only once. This allows for a lightweight transfer with no redundancy of even gigantic models. Thanks to the view dependent hierarchy traversal, only targeted nodes are downloaded and decompressed. The visualisation is maintained at a constant framerate of 60 frames per second thanks to our device dependant criteria, independently from the model. The progressive decompression provides a high granularity of levels of details to adapt as best to the view as possible.

5. Thanks

We are deeply grateful to Air Marine, Art Graphique et Patrimoine and Q-Park for kindly providing their datasets. This work was partly supported by the PIA «Programme d’investissements d’avenir» operated by Bpifrance.

References

- [AD01] ALLIEZ P., DESBRUN M.: Progressive compression for lossless transmission of triangle meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH ’01, Association for Computing Machinery, p. 195–202. doi:10.1145/383259.383281. 1
- [BIS19] BOUTSI A.-M., IOANNIDIS C., SOILE S.: An integrated approach to 3d web visualization of cultural heritage heterogeneous datasets. *Remote Sensing* 11, 21 (2019). URL: <https://www.mdpi.com/2072-4292/11/21/2508>, doi:10.3390/rs11212508. 1
- [HLK01] HO J., LEE K.-C., KRIEGMAN D.: Compressing large polygonal models. *Visualization ’01* (10 2001). doi:10.1109/VISUAL.2001.964532. 1
- [LCD13] LAVOUÉ G., CHEVALIER L., DUPONT F.: Streaming compressed 3d data on the web using javascript and webgl. pp. 19–27. doi:10.1145/2466533.2466539. 1
- [PCDS18] POTENZIANI M., CALLIERI M., DELLEPIANE M., SCOPIGNO R.: Publishing and consuming 3d content on the web: A survey. *Foundations and Trends® in Computer Graphics and Vision* 10 (12 2018), 244–333. doi:10.1561/06000000083. 1
- [PD15] PONCHIO F., DELLEPIANE M.: Fast decompression for web-based view-dependent 3d rendering. pp. 199–207. doi:10.1145/2775292.2775308. 2
- [YSGM04] YOON S.-E., SALOMON B., GAYLE R., MANOCHA D.: Quick-VDR: Interactive View-dependent Rendering of Massive Models. *IEEE Visualization* (2004), 131–138. 2