Adaptive quantization with error bounds for compressed view-dependent multiresolution meshes

Markus Grabner[1] and Christopher Zach[2]

[1]Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria
[2]VRVis research center, Graz, Austria

Abstract

We present a simple method to adaptively determine compression parameters according to the given input data. Our method allows to make better use of the available dynamic range (number of bits per component of vertex locations) than with a given fixed-point representation. Moreover, the number of bits per component is also determined adaptively under a user-specified upper bound for the quantization error.
Our adaptive quantization method can handle models with largely non-uniform geometric resolution. This is typically the case for manually created models or models derived from different sensors. The experiments we performed demonstrate the benefits of the proposed method.

Categories and Subject Descriptors (according to ACM CCS):
I.3.6 [Computer Graphics]: Graphics data structures and data types

1. Introduction

Both view-dependent multiresolution and mesh compression have been active fields of research during the past ten years. These techniques are useful to manage and visualize huge amounts of geometric data, which is especially important if data is to be transmitted over a slow network link. As these methods become mature, they are used for real-life applications. In our case we study the use of compression and view-dependent progressive transmission in the context of digital archaeology[6].

Fusion of data derived from different sources is important for creating complex virtual environments such as a model of an archaeological excavation site. Aerial photos or isolines extracted from analog maps can be used to create a digital elevation model at large scale into which the site of interest is embedded. More information about the surface can be acquired by terrestrial surveying. Objects of particular interest can be scanned at high detail by laser range finders, photogrammetry, or structured light methods. And finally, some objects can be manually modelled based on the knowledge of archaeologists.

Merging data from the abovementioned sources gives a model containing geometry at largely differing resolution. The sampling distance in this example may vary in the range between 100 micrometers up to 10 or even 50 meters. Due to the non-uniform scale of geometric detail we cannot find a reasonable quantization of vertex locations according to an absolute global accuracy.

The main contributions of our work are a method to adaptively scale the input to the encoder such that it optimally fits into the available dynamic range, and an adaptive bit allocation scheme that guarantees to meet a user-defined error threshold. Related work in the fields of multiresolution and mesh compression is briefly reviewed in Section 2. Our method is explained in detail in Section 3, results are given in Section 4. The paper is concluded in Section 5.

2. Related work

Compression of 3D geometry, being a relatively new field of research, has been pioneered by Michael Deering[3]. He introduced the *generalized triangle mesh*, which is similar to the well-known triangle strip concept in its ability to reuse already transmitted vertices. However, the generalized triangle mesh acts as a *mesh buffer* with explicit storage and retrieval capabilities for more than just two vertices (as in the case of triangle strips). Vertex locations are quantized (i.e., truncated to a fixed number of bits) and delta encoded.

Taubin and Rossignac re-order mesh vertices in a way that reflects topology, giving an efficient encoding of topology. They use a *vertex predictor* that takes into account the K most recently encoded vertices[14]. Instead of storing absolute vertex coordinates, only the *correction vector* between the predicted and the actual position of a vertex is encoded. Predictor parameters are estimated to minimize the least square error of all correction vectors.

2.1. Progressive compression

Progressive compression is particularly important for applications involving inefficient transmission of mesh data (e.g., Web3D). Hoppe[10] recognized that progressive mesh simplification supports compact encoding, which was further elaborated by Pajarola and Rossignac[12].

Cohen-Or et al.[2] used vertex decimation and coloring the affected patches for connectivity encoding. Alliez and Desbrun[1] show the importance of regular meshes (i.e., valence close to six for all vertices) and propose a valence-driven progressive compression technique. If only mesh appearance needs to be preserved, the bit rate can be further reduced by removing parameter and connectivity information and storing geometry in a wavelet representation[11].

A geometry-based vertex ordering and encoding scheme has been presented by Devillers and Gandoin[4]. They recursively (and regularly) subdivide the domain of input vertices and store the number of vertices in one of the halfs. The decoder implicitly calculates and refines vertex coordinates during traversal of the hierarchical subdivision. The authors extended their geometry encoding approach by additional encoding of the topology of an arbitrary simplicial complex[7].

2.2. Selective refinement

Selective refinement is important for applications not restricted to present a single object to the user. If a scene containing a large collection of separate objects is provided, the user often wants to explore a few of them initially before downloading all objects in full detail. This has been intensively studied for non-compressed data[16, 9, 5]. However, traditional progressive compression is not sufficient for this type of application since the whole scene is decoded at uniform detail (unless separate objects are individually encoded at the cost of larger overhead).

Wavelet-based methods can be easily adapted to this requirement due to the regular hierarchical structure of the coefficients describing the mesh at successively higher detail. At the same time, some flexibility with respect to the representable data is lost.

A completely different approach, avoiding any explicit representation of a continuous surface, uses a hierarchical description of a point cloud, typically sampled from the original surface[13]. Arbitrary triangle meshes can be compressed for a selective refinement framework by identifying vertices by their paths through the simplification hierarchy[8].

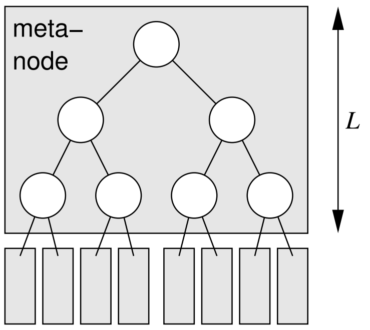


Figure 1: *Meta-node consisting of L levels of the simplification hierarchy*

3. Adaptive compression

Our method makes use of the meta-node concept and a relative geometry coding scheme as reviewed in Sections 3.1 and 3.2, respectively. We improve these techniques by two contributions: first, the quality of the decoded model can be improved by scaling the correction vectors (see Section 3.3). After the appropriate scale has been found, the minimum number of bits per component is allocated for representing geometry that still satisfies a user-defined error threshold (Section 3.4).

3.1. Meta-node concept

ElSana and Chiang introduced the concept of *meta-nodes*[5] to group a fixed number L of levels in the simplification hierarchy into a unit that is atomic unit with respect to storage and transmission (see Figure 1). This concept is also well suited to adapt to changing resolution within the scene. In a highly complex model we often find regions with more geometric detail embedded into parts with less detail, but there is a good chance that within one meta node the resolution is homogeneous.

We make use of this fact by optimizing the representation for a homogeneous resolution (Section 3.3). If this is not possible, we increase the number of bits to accurately represent geometry (Section 3.4).

3.2. Relative geometry encoding

Several methods have been developed to encode geometry relatively to previously encoded vertices (e.g., the *parallelogram rule*[15]). Since we want to apply our method in the

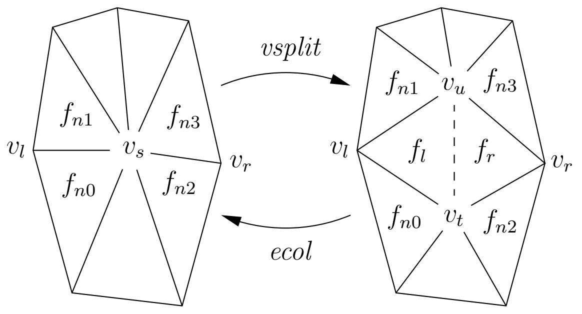


Figure 2: *Mesh transformations*

view-dependent multiresolution framework, we encode locations of vertices (v_t and v_u in Figure 2) relative to their parent node (v_s) in the simplification hierarchy[8].

We denote these relative locations *correction vectors* $\mathbf{c}_i^j$ at level i, $j = 1 \ldots 2^{i+1}$, and their average magnitudes at each level

$$\overline{d}_i = 2^{-(i+1)} \sum_{j=1}^{2^{(i+1)}} ||\mathbf{c}_i^j||. \qquad (1)$$

Due to increasing resolution at lower levels of the simplification hierarchy, the $\overline{d}_i$ tend to decrease with increasing hierarchy depth i. For a sufficiently regular mesh, the average *shortening factor* $s = \frac{\overline{d}_i}{\overline{d}_{i+1}}$ between two levels is $\sqrt{2}$ as previously suggested[8]. To compensate for this, the vectors

$$\tilde{\mathbf{c}}_i^j = \frac{s^i}{\max_{i,j}(||\mathbf{c}_i^j||)} \mathbf{c}_i^j, \qquad (2)$$

are stored and transmitted, where all $\tilde{\mathbf{c}}_i^j$ can be stored in the same fixed point encoding in the interval $[-1;+1]$.

If the mesh is irregular with respect to distribution of geometric detail, $s \approx \sqrt{2}$ does not hold any longer, and we need to calculate the optimal s as explained in the next section.

3.3. Adaptive scaling

To minimize quantization error, we want at each level of the hierarchy to optimally use the possible range of correction vectors under a given bit count (see Section 3.4). Therefore our goal is to find a shortening factor s for each meta-node suitable for use in Equation (2). We proceed as follows:

1. Determine the average magnitude $\overline{d}_i$ of correction vectors at level i according to Equation (1).
2. Fit a line $y = ax + b$ to the point set $(x_i, y_i) = (i, \log \overline{d}_i)$ in the Euclidean plane (i.e., minimize the sum of squared errors between the line and the points).
3. Use $s = \exp(a)$ as the average shortening factor for the given meta-node.

If the magnitudes $||\mathbf{c}_i^j||$ of all correction vectors would enter the line fitting process instead of their per-level averages $\overline{d}_i$, the values at lower levels would dominate the line fit due to their large number. This could lead to suboptimal range usage at levels close to the meta-node's root, requiring to increase the bit count for the whole meta-node.

3.4. Adaptive bit allocation

Many applications require that the quantization error is below a user-defined threshold. If the mesh is sufficiently regular such that a value for s consistent among all levels within the meta-node can be found, as few as three bits per component give reasonable results (see Figure 6(c)). If this is not the case, additional data needs to be stored to fulfill the error criterion. Starting at six bits per component in our current implementation, the number of bits for each component is tentatively increased or decreased until the smallest number satisfying the error threshold is found.

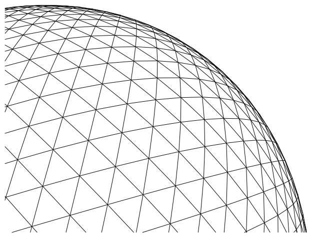
We use the maximum distance between the vertex locations input to the encoder and the corresponding locations output by the decoder as our error measure. This approach has been chosen on one hand for its simplicity, and on the other hand due to the difficulties in evaluating more advanced measures (e.g., Hausdorff distance) in view-dependent meshes. Moreover, we decided to define the desired error bound relative to the extents of the meta-nodes to account for varying geometric detail across the mesh. This has the following consequences:

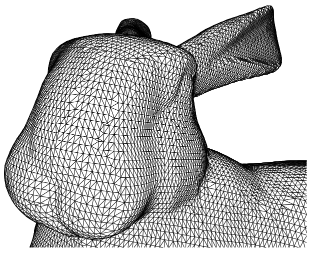
1. Vertex locations in densely sampled regions will be more accurately represented than those in less densely sampled regions in terms of absolute error.
2. Vertex locations in "inner" meta-nodes (i.e., meta-nodes not containing vertices of the input mesh) will be less accurately represented than those of the original mesh. While the input model needs to be preserved accurately, vertices at inner nodes do not provide additional information. Instead, they are typically derived from some optimization step performed for each edge collapse. Hence we do not lose the ability to restore the model at full resolution within the desired error bounds even if we do not represent inner vertices at the same (absolute) accuracy as input vertices.

There is an obvious need to avoid accumulation of quantization errors. This can easily be achieved by synchronizing encoder and decoder, i.e., replacing the already processed parts of the mesh by the corresponding quantized mesh during the encoding stage.

4. Results

To demonstrate the ideas of Section 3, we apply our method to the models shown in Figure 3. The wireframe rendering reveals the regular structure of the sphere and bunny meshes


(a) sphere (7200 triangles)


(b) Stanford bunny (69451 triangles)

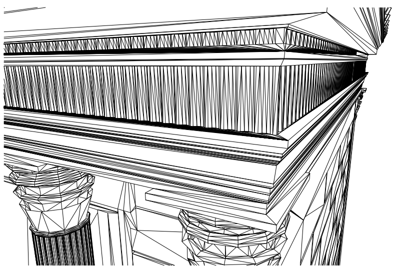
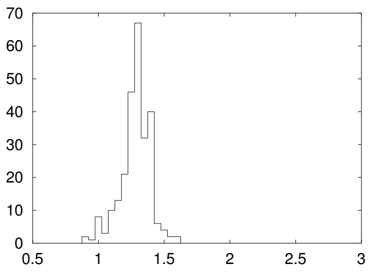
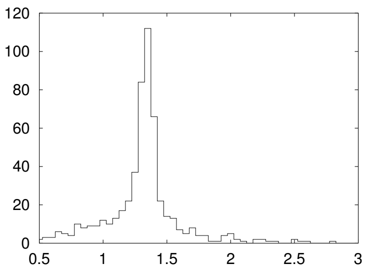

(c) northwest Heroon at Sagalassos/Turkey (31406 triangles)

Figure 3: *Detail views of models used in this paper*


(a) sphere


(b) bunny

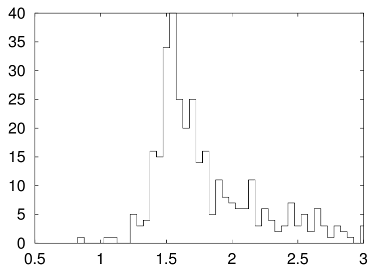

(c) Heroon

Figure 4: *Histograms of shortening factor s (discretized at intervals of 0.05) for different models*

in Figures 3(a) and 3(b), respectively, while the scale of geometric detail largely varies across the Heroon surface (Figure 3(c)).

First we evaluate the regularity of the three models by the histograms of the shortening factors s (Section 3.2) in Figure 4. We find the peak of the histogram to be close to $\sqrt{2}$ as expected for the regular meshes. Values below 1 are due to near-degenerate cases in meta-nodes consisting of only few levels. However, the Heroon model shows a significantly broader distribution of s due to its different scales of geometric detail (see Figure 4(c)).

The adaptive scaling procedure explained in Section 3.3 is illustrated in Figure 5. The steepest line (i.e., most variation in detail scale) is again observed for the Heroon mesh (Figure 5(c), note the scaling of the y-axis).

The benefits of using a proper shortening factor are demonstrated in Figure 6 by overriding the adaptive procedure and selecting fixed values. Obviously, choosing $s = 1$ leads to severe quantization artefacts in Figure 6(a). Setting $s = \sqrt{2}$ in Figure 6(c) gives considerably better results without increasing code size. Note that each correction vector is represented by only 9 bits.

Finally, results of the adaptive bit count selection are presented in Figure 7. With an error tolerance of 5% in Figure 7(a) we observe significant deviation from the original shape in the roof area. However, the fine details below the roof are still accurately represented since they belong to meta-nodes at lower levels of the meta-node hierarchy. Under a 1% error threshold some minor errors are still present (Figure 7(b)), which are no longer visible at an error threshold of 0.3% (Figure 7(c)). Note that the size of he encoded geometry increases with decreasing error threshold.

5. Conclusions and future work

We presented a method designed for compression and view-dependent access of triangle meshes containing geometry at largely differing scale. Some encoder parameters (i.e., shortening factor between simplification hierarchy levels, bit count for geometry representation) are chosen based on this scale evaluation to compensate for highly non-uniform resolution of the data. This gives the user the ability to adjust the encoding process by an intuitive error tolerance parameter. We conducted several experiments to demonstrate the benefits of the new method.

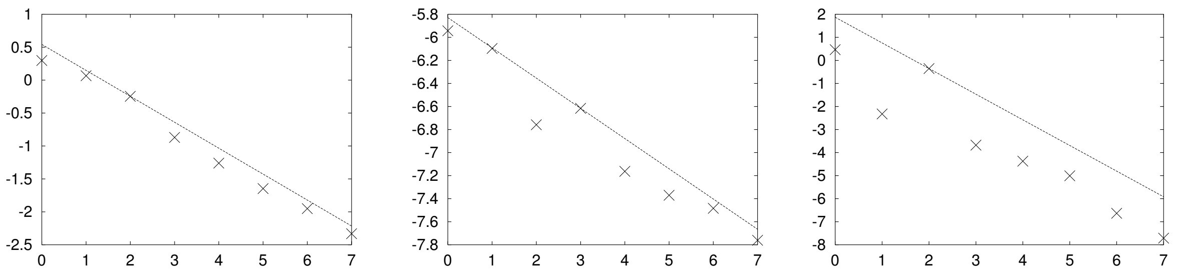
Our relative error evaluation may still fail in case of a significant non-uniformity of resolution within a single meta-node. This problem could be solved by evaluating the dominant scale within a meta-node and taking this into account for relative error estimation.

6. Acknowledgments

Many thanks to Ivana Kolingerova and Andrej Ferko for their comments. This work has been partially supported by the European Commission under contract no. IST-1999-20273.

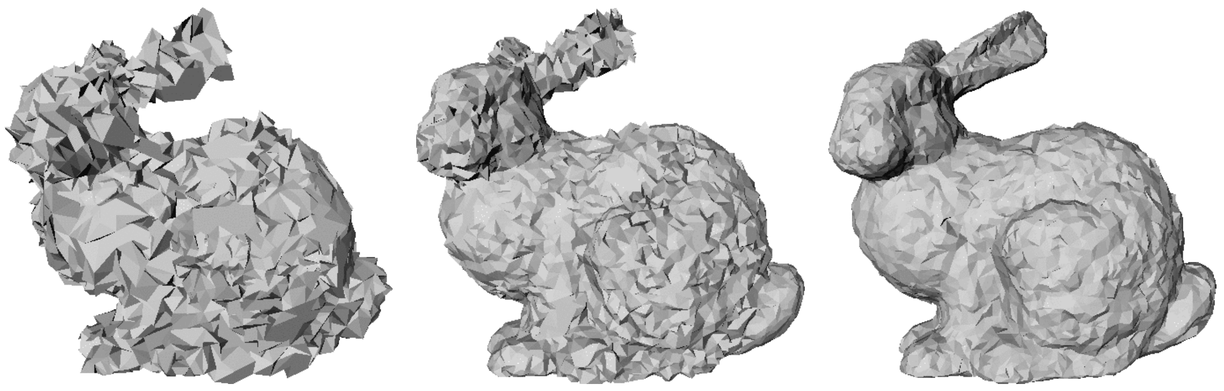
References

1. Pierre Alliez and Mathieu Desbrun. Progressive compression for lossless transmission of triangle meshes. In Eugene Fiume, editor, *SIGGRAPH 2001 Conference Proceedings*, Annual Conference Series. ACM SIGGRAPH, ACM Press / ACM SIGGRAPH, August 2001.
2. Daniel Cohen-Or, David Levin, and Offir Remez. Progressive compression of arbitrary triangular meshes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 67–72, San Francisco, 1999. IEEE.
3. Michael F. Deering. Geometry compression. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, volume 29 of *Annual Conference Series*, pages 13–20. ACM SIGGRAPH, Addison Wesley, August 1995.
4. Olivier Devillers and Pierre-Marie Gandoin. Geometric compression for interactive transmission. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings Visualization 2000*, pages 319–326. IEEE Computer Society Technical Committee on Computer Graphics, 2000.
5. Jihad El-Sana and Yi-Jen Chiang. External memory view-dependent simplification. In *Proceedings Eurographics*, volume 19 of *Computer Graphics Forum*, pages 139–150. Blackwell Publishers, August 2000. ISSN 1067-7055.
6. John Cosmas et al. 3D MURALE: A multimedia system for archaeology. In Stephen N. Spencer, editor, *Proceedings VAST 2001: Virtual Relity, Archaeology, and Cultural Heritage*, pages 297–305. ACM SIGGRAPH, November 2001. ISBN 1-58113-447-9.
7. Pierre-Marie Gandoin and Olivier Devillers. Progressive lossless compression of arbitrary simplicial complexes. In John Hughes, editor, *SIGGRAPH 2002 Conference Proceedings*, Annual Conference Series, pages 372–379. ACM SIGGRAPH, ACM Press / ACM SIGGRAPH, August 2002.
8. Markus Grabner. Compressed adaptive multiresolution encoding. *Journal of WSCG*, 10(1):195–202, February 2002. ISSN 1213-6972.
9. Hugues Hoppe. View-dependent refinement of progressive meshes. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 189–198. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
10. Hugues Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, February 1998. ISSN 0097-8493.
11. Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. In Kurt Akeley, editor, *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, New York, July 2000. ACM SIGGRAPH, ACM Press / ACM SIGGRAPH / Addison Wesley Longman.
12. Renato Pajarola and Jarek Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, January/March 2000.
13. Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In Kurt Akeley, editor, *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, New York, July 2000. ACM SIGGRAPH, ACM Press / ACM SIGGRAPH / Addison Wesley Longman.
14. Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.
15. Costa Touma and Craig Gotsman. Triangle mesh compression. In Wayne Davis, Kellogg Booth, and Alain Fournier, editors, *Proceedings of the 24th Conference on Graphics Interface (GI-98)*, pages 26–34, San Francisco, June 18–20 1998. Morgan Kaufmann Publishers. ISBN 1-55860-550-9.
16. Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In *IEEE Visualization '96*. IEEE, October 1996. ISBN 0-89791-864-9.



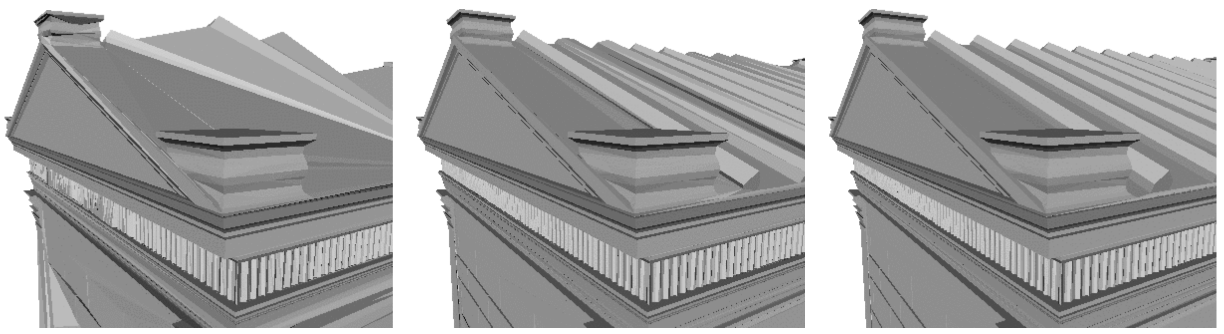
(a) root meta-node of sphere (b) inner meta-node of bunny (c) inner meta-node of Heroon

Figure 5: *Average correction vector magnitudes (y-axis, logarithmic) for each level within a meta-node (x-axis) and lines fit to point set for different models (note that the line is shifted such that all samples are below it as explained in Section 3.2)*



(a) $s = 1$ (b) $s = 1.1$ (c) $s = \sqrt{2}$

Figure 6: *Bunny model with 3 bits per component and different values for s*



(a) $e = 0.05$ (289kbit) (b) $e = 0.01$ (560kbit) (c) $e = 0.003$ (719kbit)

Figure 7: *Roof of Heroon model with different values for the error bound e, the size of the encoded geometry is given in kilobits*