# Mesh Compression with Random Accessibility

Sungyul Choe[1] * Junho Kim[1,3] † Haeyoung Lee[2] ‡ Seungyong Lee[1,4] § Hans-Peter Seidel[4] ¶

POSTECH[1] Hongik Univ.[2] RWTH-Aachen[3] MPI Informatik[4]

## Abstract

*Previous mesh compression techniques provide nice properties such as high compression ratio, progressive decoding, and out-of-core processing. However, none of them supports the* random accessibility *in decoding, which enables the details of any specific part to be available without decoding other parts. This paper introduces the random accessibility to mesh compression and proposes an effective framework for the property. The key component of the framework is a* wire-net mesh *constructed from a chartification of the given mesh. Experimental results show that random accessibility can be achieved with competent compression ratio, only a little worse than single-rate and comparable to progressive encoding.*

**Keywords:** Mesh compression, Random accessiblity, Wire-net mesh

## 1. Introduction

As exquisite meshes gain popularity in various applications, mesh compression has been an active area of research to reduce storage and transmission time for last 10 years. Single-rate compression [28, 22, 29, 2, 18] has reached a sophisticated level, especially in connectivity encoding. Progressive compression [5, 21, 1, 9] provides decompressed meshes in multi-resolutions, allowing immediate access of global shape during transmission with a little more overhead than single-rate algorithms.

However, none of the techniques provides the property of *random accessibility* in the sense that the details of any specific part in the original mesh can be made available without decoding other parts. The techniques mostly have a symmetric process between encoding and decoding. They tended to concentrate more on how to compress and resulted in a deterministic decompression as merely a reverse of compression. Consequently, only streaming of the decompressed mesh data is provided, prohibiting random order access of compressed mesh parts.

In contrast, in other domains of compression, the random accessibility has been considered as an important property and incorporated into compression schemes. The most immediate example is the MPEG encoding for videos [20]. In a MPEG video, we can play a selected scene without decoding all the preceding frames. In volume compression, a 3D wavelet-based scheme was proposed which allows random access to compressed voxels [4].

In this paper, we introduce the random accessibility to mesh compression and propose a framework to support the property. Similar to other domains, such as videos and volumes, parts of a mesh can be selectively decoded in a non-deterministic order for visualization or processing such as editing while the other parts remain in the compressed form. The key component of our framework is a *wire-net mesh*, which plays the role of an indexing structure for random access to mesh parts (see Fig. 1). By combining the suitable techniques for the framework, we achieve competitive compression ratio with random accessibility, which is only a bit worse than single-rate and comparable to progressive encoding.

## 2. Related Work

### 2.1. Random accessible compression

Random accessibility that allows us to only restore parts on-demand is very common in the compression of multimedia data such as sound and video. MPEG provides a set of compression formats to encode digital audio and video with high compression ratio and random accessibility [20]. Bajaj *et al*. [4] proposed an efficient wavelet-based compression scheme for interactive volume data visualization. The success of random accessibility in other domains, such as multimedia and volume data, motivated our work in this paper with the observation that the property has not been provided in mesh compression.

* ggalssam@postech.ac.kr, http://home.postech.ac.kr/∼ggalssam
† victor@postech.ac.kr, http://home.postech.ac.kr/∼victor
‡ leeh@cs.hongik.ac.kr, http://www.cs.hongik.ac.kr/∼leeh
§ leesy@postech.ac.kr, http://www.postech.ac.kr/∼leesy
¶ hpseidel@mpi-sb.mpg.de, http://www.mpi-sb.mpg.de/∼hpseidel/

### 2.2. Mesh compression

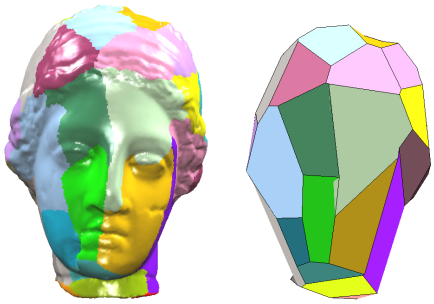
*Single-rate compression* Ever since the introduction of Deering's algorithm [6], single-rate compression of 3D meshes has been an active area of research. "Edgebreaker" [22] was introduced first with a theoretical bound for the compression ratio. The innovative Touma-Gotsman's [29] utilized the entropy of valence distribution in a mesh for better compression ratio. Alliez-Desbrun's [2] improved upon [29] and achieved pseudo-optimal ratios for connectivity encoding. All of these algorithms were initially developed for triangle meshes. Polygon mesh compression techniques have also been proposed [14, 16, 12]. "Angle-Analyzer" [18] for a triangle-quad mesh applied localized intrinsic geometry property for better compression ratio. Recently spectral approaches [15, 26] inspired by image compression and shape compression techniques [17, 11, 3] via remeshing have also been introduced.

*Progressive compression* Progressive compression algorithms are to provide the overview of the complete mesh in a coarse to fine fashion with the best trade-off between rate and distortion. Connectivity-driven [27, 5, 21, 1], geometry-driven [7, 9], and remeshing-based [17, 11] algorithms have been proposed. Progressive compression is similar to our random accessible compression in a way to give an immediate access to the decompressed shape but different in the levels of access. In progressive compression, to access a part of the mesh in the finest level, still we have to wait for finishing the whole decompression. Our algorithm provides a more interactive and direct way to access arbitrary parts of the decompressed mesh.

## 3. Overview

We can define the random accessibility of a compression scheme as the property that allows the decoding of desirable parts in an arbitrary order without decoding the other non-interesting parts. To support such a property, a compression scheme should have two components: an indexing structure for the parts of the compressed object and an encoding/decoding technique for separately handling the parts.

We decompose the given mesh into separate segments, called *charts*, and handle the charts independently from each other in the encoding/decoding process. Although a progressive scheme can be applied to encode the charts, we use a single-rate algorithm to achieve better compression ratio. In addition, in the setting of random accessible compression, the benefits of progressive decoding are much reduced. Since any desired part can be accessed without decoding other parts, the latency can be minimized by transmitting only the necessary data.



(a) chartification (b) wire-net mesh

**Figure 1. Chartification and its corresponding wire-net mesh: A wire-net mesh is a manifold structure that abstractly represents a chartification. Each vertex/edge/face of a wire-net mesh corresponds to a corner vertex/wire/chart of a chartification, respectively.**

With compressed charts, we have no immediate information on the shape of the original mesh and due to the irregularity, an underlying structure for indexing, such as a time line in a video or a 3D index for a volume, is also not available. To resolve this problem, we use a polygonal mesh, called a *wire-net mesh*, constructed from the chartification of the original mesh (see Fig. 1). Each chart is mapped to a face of the wire-net mesh and the common boundary between two adjacent charts, called a *wire*, is mapped to an edge. The vertices of a wire-net mesh come from the corner vertices of the chartification. By designating the faces of a wire-net mesh, we can specify the desirable parts of the original mesh in the unit of charts.

The wires of the chartification are compressed separately from the chart interiors. When the selected charts are to be reconstructed, we first decompress the wires surrounding the charts and then decompress the chart interiors from the boundaries. With this approach, the common boundary is reconstructed only once and shared for adjacent charts,providing an easy stitching of arbitrarily selected charts.

## 4. Encoding/Decoding Steps

### 4.1. Mesh chartification

The first step of the encoding process is the chartification of the given mesh. Since the chartification determines the structures of the wire-net mesh, wires, and charts, it has influences on the compression ratios of the components. Among the three components, the wire-net mesh has the simplest structure, whose vertices come from the corner vertices of the chartification. In contrast, wires consist of the boundary vertices of the charts and contain a

much larger number of vertices than the wire-net mesh. Obviously, charts contains most of the vertices of the given mesh. Hence, the bits used for the wire-net mesh in the compressed file is rather small and the wires and charts determines the majority of the compression performance.

In this paper, we consider compactness and planarity of charts as the requirements of the chartification. The compactness of charts is important to reduce the length of wires, which implies that the wires are straightened as much as possible. This property helps to increase the compression ratio when we encode the geometry of a wire. The planarity of a chart is related with the geometry compression ratio when we encode the chart. When a chart is planar, we can encode the geometry of the chart with high compression ratio.

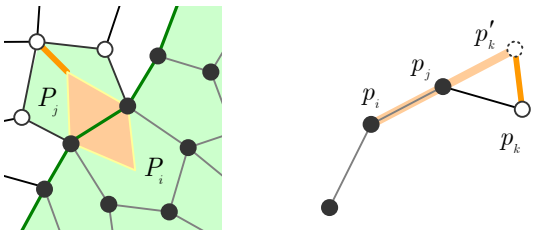
Chartification methods that consider the compactness and planarity of charts have been developed for texture mapping and fast global illumination [10, 24, 25]. We have experimented with these methods and found that the method proposed by Sander *et al*. [25] for multi-chart geometry images achieves the best compression performance. We have tested variations of [25] with different cost functions but the original formulation in [25] gave comparable compression ratio in most cases of our experiments.

The basic idea of the chartification method in [25], inspired by Lloyd quantization [19], is to repeatedly update the current charts by moving the seeds to the chart centers. The iteration results in a centroidal Voronoi tessellation [8] of the mesh surface. By including the normal variation of the faces in the distance function, the final charts capture the planar regions on the mesh surface where the chart boundaries are mapped to the features with large normal variations. Although the compactness of charts are not directly considered in the distance function, a centroidal Voronoi tessellation usually contain short chart boundaries.

In [25], the initial chartification for iterative updates is obtained by selecting the seeds one by one. In this paper, we use the chartification method based on face clustering [10] for the initialization. When the desired number of charts $k$ is given, we first generate $k$ charts by face clustering and the result is initial chartification for iterative updates. In the face clustering, we maintain each chart as a topological disk.

### 4.2. Wire-net mesh compression

To compress a wire-net mesh, we adopt the technique proposed by Khodakovsky *et al*. [16] for connectivity encoding. For geometry encoding, the parallelogram prediction for polygons [13] can be used. However, a wire-net mesh has a more irregular distribution of vertex positions in a face than a usual polygonal mesh. This irregularity makes the parallelogram prediction [13] less efficient for a wire-net mesh.




(a) vertices on a wire-net mesh (b) vertices on a wire

**Figure 2. Vertex position encoding: (a) Each vertex position in the polygon $P_j$ is encoded from the predicted center of $P_j$, which is estimated from the center of polygon $P_i$ with the parallelogram rule. (b) Each vertex position $p_k$ in a wire is encoded from $p'_k$, which is estimated by the 1D version of the parallelogram rule with the two previous vertices, $p_i$ and $p_j$.**

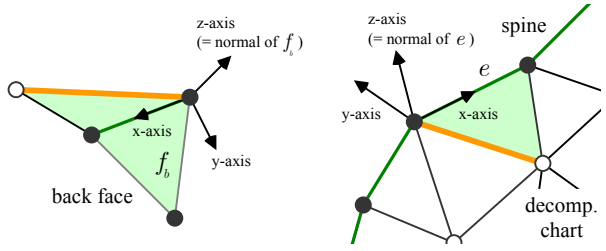
To encode the geometry of a wire-net mesh, we apply the parallelogram prediction [29] to the centers of polygons (see Fig. 2(a)). That is, when a new polygon $P_j$ is visited from $P_i$ through a gate, the center of $P_j$ is estimated from the center of $P_i$ by the parallelogram prediction. Then, the geometry of vertices in $P_j$ are encoded by using the differences from the estimated center.

### 4.3. Wire compression

Since a wire is a sequence of vertices, the connectivity of a wire is obvious. For connectivity encoding of a wire, it is sufficient to store the number vertices on the wire. To encode the geometry, we use a linear prediction for the vertex sequence, which is a 1D version of the parallelogram prediction (see Fig. 2(b)). When $p_i$, $p_j$, and $p_k$ are three consecutive vertices on a wire, we predict the position $p'_k$ for $p_k$ by reflecting $p_i$ with respect to $p_j$; that is, $p'_k = p_j + (p_j - p_i)$.

### 4.4. Chart compression

In this paper, to compress each chart, we use "Angle-Analyzer" [18], which is a single-rate compression algorithm with the best performance reported so far. Since the chart boundary has already been encoded as wires, only the chart interior is encoded with the algorithm. The encoding starts from the chart boundary (i.e., wires) by initializing the gate list with the chart boundary edges and continues until every face in the chart is traversed. For geometry encoding, among the two techniques proposed in [18], we use the local coordinate based approach. In the approach, the normal vector of the back face of a gate is used to define the local coordinates, as shown in Fig. 3(a). However, there exists no back face for a gate generated from the chart bound-




(a) interior case (b) nearby-wire case

**Figure 3. Local frames for geometry (de)coding of a chart: (a) The geometry of the opposite vertex is (de)coded with a local frame, defined with the normal of the back face $f_b$ and the direction of the current gate edge. (b) The normal stored at edge $e$ is used to define a local frame for a vertex adjacent to the chart boundary.**

ary. To resolve this problem, we store a normal vector for each edge of chart boundaries, as shown in Fig. 3(b). The normal vector is computed as the average normal vector of the two faces adjacent to the boundary edge and encoded by spherical quantization [23].

### 4.5. Compressed file structure

To enable the random access to the charts through the wire-net mesh, we have to link the edges and faces of the wire-net mesh to the encoded wires and charts, respectively. When the wire-net mesh is recovered at the beginning of the decoding stage, each edge is assigned an *edge-id*. This *edge-id* is used as the pointer to access the corresponding wire in the compressed file. The mapping from a face to a chart is handled with a *face-id* in the same way.

Fig. 4 illustrates our file structure for random accessible mesh compression. Each shaded region in Fig. 4 represents the compressed data of a wire-net mesh, wires, or charts. In the mesh information part, we store the auxiliary information for decoding in the raw format, such as a bounding box, quantization levels, and so on. The pointer table keeps track of the positions in the file at which the compressed data of wires and charts are stored. In the decoding phase, the pointer table resides in the main memory.

### 4.6. Chart decompression

Suppose that a face $f$ of the wire-net mesh is selected for decoding the corresponding chart. From the *edge-id*s of the edges of the face $f$, we first obtain the pointers to the wires to be decoded in the compressed file. The connectivity and geometry of the chart boundary are then recovered from the wires and the vertices of the face $f$. The reconstructed chart

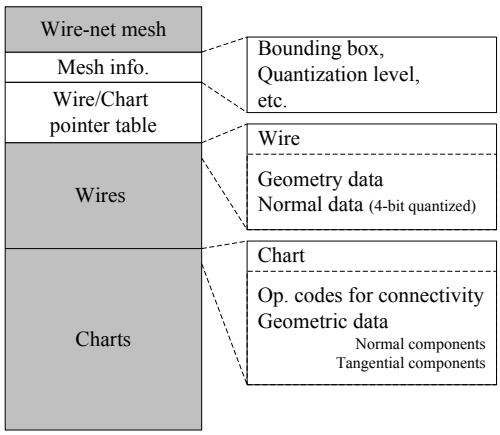



**Figure 4. File structure for random accessible mesh compression: The shaded regions contains the compressed data described in Sec. 4.**

boundary gives the initial gate list. Also, the *face-id* of $f$ is used to access the encoded chart data. The chart interior is recovered by decoding the chart with the initial gate list obtained from wires. When two or more charts are decoded simultaneously, the common wires are decoded only once and used as the shared chart boundaries.

## 5. Experimental results

We experimented with our random accessible mesh compression technique on several models and different numbers of charts. We also compared the compression results with the current state-of-the-art mesh compression techniques. In this paper, for entropy encoding, we use an order-1 adaptive arithmetic coder [30].

Table 1 shows the comparison of compression ratios between our technique, single-rate algorithms [29, 18], and a progressive algorithm [1]. For geometry encoding of [29], we used 12-bit quantization. For [18], the quantization levels were determined to have almost same distortion errors with 12-bit quantization of [29]. We also used 12-bit quantization for [1]. For our technique, we used the same quantization level with [18] for geometry encoding of chart vertices.

In Table 1, our technique gives little worse compression ratios than a single rate algorithm because it has to pay some overhead to provide the good property of random accessibility. However, with up to certain number of charts, the compression ratio of our technique is better than progressive compression, regardless of mesh models. This implies that our technique is a good alternative to progressive com-

| model (# of V) | TG [29] | LAD [18] | AD [1] | ours (#charts) |
|---|---|---|---|---|
| Igea (67,180) | 17.24 | 14.36 | 19.15 | 17.72 (50)<br>18.46 (60)<br>18.52 (70)<br>19.08 (80)<br>19.36 (90)<br>19.56 (100) |
| Skull (98,306) | 12.35 | 11.41 | 16.80 | 14.98 (50)<br>15.59 (70)<br>16.47 (100) |
| Iphigenie (371,750) | 9.47 | 13.78 | 15.94 | 16.01 (70)<br>16.37 (100) |
| Feline (49,864) | 16.56 | 15.62 | 20.30 | 19.33 (50)<br>20.77 (70)<br>21.40 (100) |
| Dino (14,070) | 19.80 | 16.53 | 25.29 | 22.80 (50)<br>23.62 (70)<br>24.84 (100) |

**Table 1. Comparison of compression ratios (bpv) for single-rate [29, 18], progressive [1], and our techniques: With up to 70 charts, our technique outperforms progressive compression.**

pression when a prompt access of mesh shape is necessary without decoding the whole compressed data.

Table 1 also shows a trade-off between random accessibility and compression ratio. When the number of charts is small, we obtain a high compression ratio although random accessibility is not fully supported. As the number of charts increases, we can provide more random accessibility with overhead in the compression ratio. Fig. 5 shows the rate/distortion curves with different numbers of charts for the Igea model. We can see that when the number of charts is larger, more bits are needed to achieve the same distortion error.

In Fig. 6, we visualized a large mesh with two different ways of decoding charts. One is simply decoding and rendering all charts and the other is processing only view-dependently selected charts. For view-dependent selection of charts, we test the visibility of charts using the face normals of the wire-net mesh. As shown in Fig. 6, when we render a mesh view-dependently, the rendering time is reduced with the number of visible charts, which is made possible with the random accessibility of our framework.

## 6. Discussion and Future Work

In this paper, we proposed a novel framework for mesh compression which provides the random accessibility similar to MPEG encoding for sound and video. By carefully adopting the component techniques, we can achieve high compression ratio, which is slightly worse than single-rate compression and in most cases, better than progressive compression.

An interesting future work would be providing the multiresolution granularity for the random accessible compression framework.

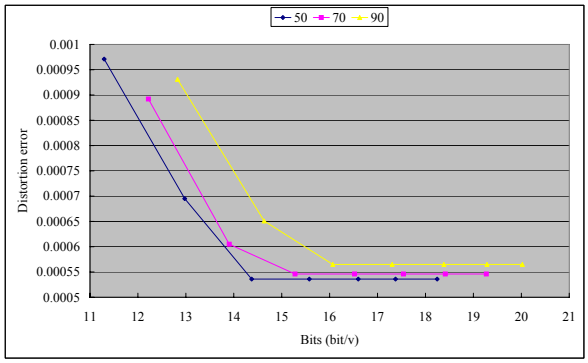
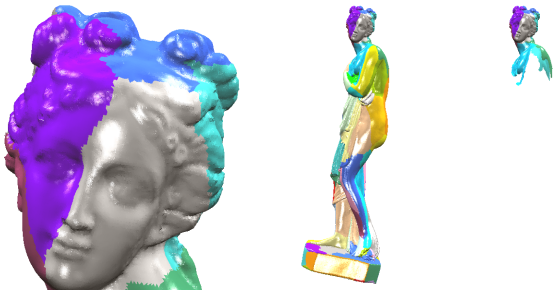



**Figure 5. Rate/distortion curves with different numbers of charts for the Igea model**



(a) screen image (b) all charts (c) face part

**Figure 6. View-dependent rendering example: The Iphigenie model has been compressed with 100 charts. (a) Image shown on the screen; (b) Restoring all charts one by one (440 milliseconds); (c) View-dependent restoring of only the face part (50 milliseconds, around 10% of the total charts).**

## Acknowledgements

The authors would like to thank Christian Rössl for helpful discussion and Sanghun Park for help at the early stage of this work. The Igea and dinosaur sculpture models are courtesy of the Cyberware (www.cyberware.com). The

skull model is courtesy of Headus (www.headus.com.au) and Phil Dench. The feline model is courtesy of Multi-Res Modeling Group at Caltech. This work was supported in part by the Korea Ministry of Education through the Brain Korea 21 program, the Game Animation Research Center, and Korea Science and Engineering Foundation (KOSEF-042010702).

## References

[1] P. Alliez and M. Desbrun. Progressive Encoding for Lossless Transmission of 3D Meshes. *Siggraph 2001 Conference Proceedings*, pages 198–205, 2001.

[2] P. Alliez and M. Desbrun. Valence-Driven Connectivity Encoding of 3D Meshes. *Eurographics Conference Proceedings*, pages 480–489, 2001.

[3] M. Attene, B. Falcidieno, M. Spagnuolo, and J. Rossignac. Swingwrapper: Retiling triangle meshes for better edgebreaker compression. *ACM Transactions on Graphics*, 22(4):982–996, 2003.

[4] C. Bajaj, I. Ihm, and S. Park. 3D RGB image compression for interactive applications. *ACM Transactions on Graphics*, 20(1):10–38, 2000.

[5] D. Cohen-Or, D. Levin, and O. Remez. Progressive compression of arbitrary triangular meshes. *IEEE Visualization Conference Proceedings 1999*, pages 67–72, 1999.

[6] M. Deering. Geometry Compression. *Siggraph 95 Conference Proceedings*, pages 13–20, 1995.

[7] O. Devillers and P.-M. Gandoin. Geometric Compression for Interactive Transmission. *Proc. of IEEE Visualization 2000*, pages 319–326, 2000.

[8] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tesselations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[9] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. on Graphics*, 21(3):372–379, 2002.

[10] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. 2001 ACM Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

[11] X. Gu, S. J. Gortler, and H. Hoppe. Geometry Images. *ACM Trans. on Graphics*, 21(3):355–361, 2002.

[12] M. Isenburg. Compressing polygon mesh connectivity with degree duality prediction. *Proceedings of Graphics Interface*, pages 161–170, May 2002.

[13] M. Isenburg and P. Alliez. Compressing polygon mesh geometry with parallelogram prediction. In *Proceedings of the IEEE Visualization 2002*, pages 141–146, 2002.

[14] M. Isenburg and J. Snoeyink. Face fixer: Compressing polygon meshes with properties. In *ACM SIGGRAPH 2000 Conference Proceedings*, pages 263–270, 2000.

[15] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. *ACM Computer Graphics (Proc. SIGGRAPH '98)*, pages 279–286, 2000.

[16] A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schröder. Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes. *Graphical Models 64(3-4)*, pages 147–168, 2002.

[17] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive Geometry Compression. *Siggraph 2000 Conference Proceedings*, pages 271–278, 2000.

[18] H. Lee, P. Alliez, and M. Desbrun. Angle-analyzer: A triangle-quad mesh codec. *Eurographics 2002 Conference Proceedings*, pages 383–392, 2002.

[19] S. Lloyd. Least square quantization in PCM. *IEEE Transaction on Information Theory*, 28:129–137, 1982.

[20] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Legall. *Mpeg Video: Compression Standard (Digital Multimedia Standards Series)*. Kluwer Academic Publishers, 1996.

[21] R. Pajarola and J. Rossignac. Compressed progressive meshes. *IEEE Trans. on Visualization and Computer Graphics*, 6(1):79–93, 2000.

[22] J. Rossignac. EdgeBreaker : Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[23] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. *ACM Computer Graphics (Proc. SIGGRAPH 2000)*, pages 343–352, 2000.

[24] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. *ACM Computer Graphics (Proc. SIGGRAPH 2001)*, pages 409–416, 2001.

[25] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proc. 2003 Eurographics Symposium on Geometry Processing*, pages 157–166, 2003.

[26] O. Sorkine, D. Cohen-Or, and S. Toledo. High-pass quantization for mesh encoding. In *Proc. 2003 Eurographics Symposium on Geometry Processing*, pages 41–51, 2003.

[27] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *Proc. of SIGGRAPH'98*, pages 123–132, 1998.

[28] G. Taubin and J. Rossignac. Geometry Compression Through Topological Surgery. *ACM Transactions on Graphics, Vol. 17, No.2*, pages 84–115, 1998.

[29] C. Touma and C. Gotsman. Triangle Mesh Compression. *Graphics Interface 98 Conference Proceedings*, pages 26–34, 1998.

[30] F. Wheeler. Adaptive arithmetic coding source code, 1996. http://www.cipr.rpi.edu/˜wheeler/ac.