Multi-way Geometry Encoding

Daniel Cohen-Or, Rami Cohen and Revital Irony

The School of Computer Science, Tel-Aviv University

Abstract

The problem of representing a 3D model in a more compact manner was extensively studied. While much attention has been paid to the compression of the connectivity, compressing the geometry remains challenging. We introduce a novel geometry-encoding algorithm, which is based on traversing the vertices in an order that improves the performance of the geometry prediction. We analyze the prediction scheme and show how a multi-directional prediction can improve the encoding of the geometry. We suggest an approximated multi-directional prediction and show that it leads to the best geometry compression ratios published so far.

1 Introduction

Large 3D models require deep compression for their efficient transmission over the network and for archiving them. As a consequence, interest in 3D encoding techniques has increased in recent years.

As in other visual media, some compression techniques are lossy while others are lossless. Lossy methods for 3D models either use mesh simplification to reduce the number of faces (triangles) that represent the model, or remeshing techniques [9], where the original mesh is replaced by another mesh that well approximates the original one and can be efficiently encoded. See [5] for a survey of recent developments in mesh compression.

Lossless mesh compression techniques need to encode both the geometry (coordinates) and the topology (connectivity) of the mesh. Although these are not entirely independent, their encoding is often treated as two separate processes, which are usually applied interleaved. Much attention has been paid to the compression of the connectivity [13] [14] [11] [6] [7] [1]; state-of-the-art connectivity compression is extremely effective [14] [1]. Compressing the geometry remains challenging, since the encoded geometry is on average five times larger than the encoded connectivity, even at 10–12 bits per uncompressed

coordinate. Higher precision even further amplifies the importance and need for effective geometry encoding.

Karni and Gotsman [8] introduced a lossy geometry compression based on a spectral method [12], which computes the eigenvalues and eigenvectors of a large sparse matrix associated with the mesh topology. Computing the eigenvectors requires large amounts of computation and memory. Karni and Gotsman addressed this issue by partitioning the mesh into relatively small patches and compressing each patch separately. They compute low-frequency eigenvectors for each patch. Thus, the partitioning approach requires the computation of eigenvectors of many small matrices instead of those of one large matrix. In essence, partitioning trades encoding and decoding time and space for compression rates. Alliez and Isenburg [3] presented a generalization of the geometry coder by Touma and Gotsman to polygon meshes. They let the polygon information dictate where to apply the parallelogram predictor, and made predictions within a polygon rather than across polygons.

This paper presents a lossless geometry encoding technique based on traversing the vertices in an order that improves the performance of the geometry prediction, which is multi-directional. We analyze the prediction scheme and show in Section 2, how a multi-directional prediction can improve the geometry encoding. In Section 3 we introduce a novel geometry encoding algorithm, which is based on the special traversing order of the vertices, that improves the geometry prediction. As will be demonstrated in the results (Section 5) this leads to the best compression ratios known so far.

2 Multi-way Geometry Predictors

The geometry encoding process begins by quantizing the coordinates if they are initially expressed as floating-point numbers. Hereafter, we assume that the coordinates are integers, and that both the encoder and the decoder know the connectivity of the mesh.

The absolute coordinates of vertices are encoded by relative coordinates, which are essentially displacements from a predicted location. The relative coordinates are integers and most of them are small. Such distribution yields lower entropy, which enables encoding with fewer bits. Better prediction schemes yield smaller displacements with lower entropy.

The parallelogram-prediction rule [14] uses a traversal of the triangles of the mesh. The predicted location of the unknown vertex p of the next triangle is based on completing the current triangle into a parallelogram (see Figure 1(a)). We call this completing operation *folding*. The vector (displacement)

between the predicted location and the exact location of p is then encoded. This method produces small displacements, and is the best predictor known. However, the parallelogram prediction of a point is based on a single neighbor in only one direction (backwards in the traversal), while a better prediction would be based on all neighboring vertices in all directions. We regard the parallelogram scheme as a 1-way predictor, as opposed to a k-way predictor that uses a multiplicity of directions to predict the location of a vertex. Figure 1(b) shows the folding of a number of triangles rather than one. In the figure, only three triangles are used. Each triangle predicts a point, and their average is usually closer to p than the point predicted by a single triangle.

Let M be a given triangulated mesh with n vertices. Each vertex $v \in M$ is represented using absolute cartesian coordinates, denoted by $v_i = [x_i, y_i, z_i]$. We define the *F coordinates* of v_i to be the differences between the absolute coordinates of v_i and the average folding of its immediate neighbors in the mesh (see Figure 1(b)).

More formally, let us denote by R_j the triangles incident upon v_i, and by T_j the triangles that are not in R_j but share an edge with a triangle in R_j. Each of the triangles in T_j can be "mirrored" about its common edge with R_j to yield a point that can be used as a prediction to v_i (the mirroring predictor is similar to the parallelogram predictor, but seems to be a somewhat better predictor in our experiments). Now, F is defined as the average of all the prediction points of T_i.

In general, it is rather easy to transform absolute coordinates to relative coordinates. The transformation back from relative to absolute is not trivial since to find the location of a vertex, requires the surrounding vertices to already be decoded. A simultaneous solving of all the vertices might be possible using relaxation techniques. However, to guarantee the convergence of the relaxation to the exact absolute coordinates, the relative coordinates need to be represented accurately with many precision bits, which hampers the compression. Applying the relaxation process to the quantized coordinates prevents the recovery of all the precision bits of the original geometry.

A k-way prediction is applied by progressive geometry-encoders [4] [10] [2], which use a hierarchy of progressively coarse-to-fine meshes. They encode the displacements of a subset of vertices in a mesh with respect to a small set of surrounding vertices from a coarser mesh. In these methods, the displacements are encoded as relative coordinates based on a k-way predictor. However, since the compression is hierarchical, the displacements tend to grow in magnitude as the mesh becomes coarser and coarser.

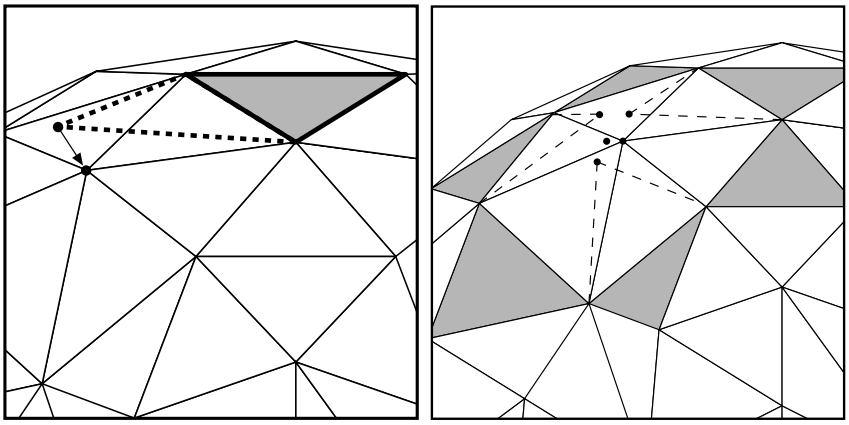

Fig. 1. The folding schemes. In (a) a 1-way prediction is used to predict the location of the bold vertex; the vector displacement is encoded. In (b) a k-way prediction is used. An average of two foldings is used as the prediction. Typically, this yields a better prediction than a 1-way prediction.

3 k-way Geometry Encoding

In the following we show how the F coordinates can be approximated using an iterative reconstruction process. The approximation predicts the location of a given vertex based on k triangles that are already decoded. This requires traversing the mesh in a special order such that in each step the prediction of a vertex is based on more than one triangle. To achieve this, the traversal of the triangle is not sequential, but in each step, the encoded triangle is selected according to some prediction criterion.

Let A be the set of triangles that has been decoded so far during the traversal (the dark triangles in Figure 2). The next vertex to be encoded, denoted by q, must be on a triangle T connected to the border of A. That is, one of the edges of T is already decoded, and adjacent to a decoded triangle $W \in A$. By folding W, the location of q can be predicted. As discussed above, if q is on k triangles connected to A the prediction is likely to be better than a 1-way prediction. In Figure 2(a) the bold vertex has a 4-way prediction, since it is on four triangles connected to a set of triangles already encoded. Thus, during the traversal, each vertex not yet decoded has its *prediction degree*.

A possible traversal order can be such that in each step the vertex with the highest prediction degree is selected. In the case of two vertices with the same prediction degree, a secondary sort is made on the basis of the variance between the predictions. Such a greedy method has an average of a 2-way prediction. This is because on an average mesh with n vertices there are $2n$ triangles only. Since each folding recovers one triangle, and no triangle is recovered twice, no more than $2n$ foldings are possible. Since the expected average is 2, it is enough to select in each step a vertex with a 2-way prediction. The traversal

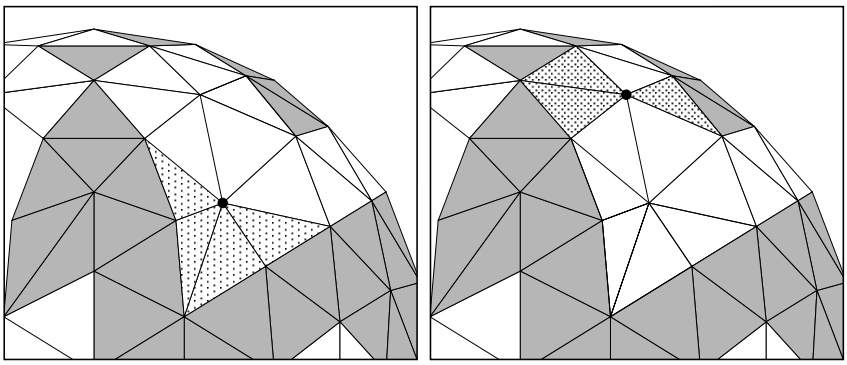

Fig. 2. The prediction degree. In (a) the bold vertex has a prediction degree of 4, while in (b) only three.

is implemented by stepping among the candidate vertices and selecting the first one to have a prediction degree greater than one.

The traversal is always applied to meshes whose boundaries are already decoded. It requires explicitly encoding one triangle connected to the boundary to start the traversal. As we shall see in Section 5, this k-way prediction consistently yield a better prediction than a 1-way prediction.

4 A Two-step Method

Based on the k-way prediction scheme, the surface can be reconstructed in two steps. First we use only the connectivity information to compute a rough approximation to the geometry. The geometry is then corrected by using displacements, to result with the exact geometry. The computation of the approximated geometry in the first step starts with known boundaries. Based on the geometry along boundaries, the k-way predictions are computed, assuming the displacements are zero. Relaxation and smoothing operations are applied to achieve a rough approximation to the mesh.

To generate the initial boundaries, the mesh is split into disk-like patches. Better predictions can be achieved if that the boundaries appear along ridge and ravines of the surface. The vertices along boundaries serve as a geometric anchor to the reconstruction of the geometry of other vertices, that are positioned solely based on the known geometry of the boundary and some expected mean values. The amount of data required for the boundary is typically rather small.

In Figure 3 the boundaries were first transmitted and reconstructed. We can see that the approximate location of vertices can be quite successful. The views in figure 3 are better approximations than typical initial meshes used by progressive mesh techniques since they contain all the vertices of the original

Fig. 3. Models reconstructed at step 1 from an initial stream consisting of the boundary data plus the connectivity data of the patches. They require only about 9 bpv. The rest of the data, which consists of the geometry corrections requires about 20 bpv.

model, but not in their exact location.

The k-way prediction scheme is applied in a different order to the above. Here the vertices are sorted according to their distance from the mesh boundary to give priority to the closer vertices. Once the triangles are folded into an initial position, an edge relaxation is applied. The algorithm iterates on all vertices, except for those on the boundary, and scans their coincident edges, trying to reposition the vertex so that the edge length approaches the average edge length. After every five iterations, a smoothing operation is applied, by displacing each vertex towards the average position of its adjacent vertices.

Model	quant.	#v	#T	Virtu3D	Our
Horse	12 bits	19k	39k	19.79	17.54
	14 bits			25.89	23.64
David	12 bits	24k	47k	22.37	19.02
	14 bits			28.92	25.16
Dinosaure	12 bits	50	100k	20.42	19.12
Venus	14 bit	14k	28k	25.17	21.57

Table 1
A comparison between Virtue3D compression and our compression with a vertex quantization of 12 and 14 bits. The results are given by the number of bits per vertex.

Table 2
Geometry entropy using 1-way prediction vs. k-way prediction. As the predication is based on more ways, consistently the entropy gets lower.

Models	1-way	2-way	3-way	4-way	5-way	6-way	All-ways
Crocodile (12b)	15.54	15.63	15.09	14.47	14.11	13.85	13.69
Venus (12b)	19.24	17.91	17.16	16.54	16.06	15.76	15.48
Dinosaure (12b)	17.47	16.33	15.57	15.08	14.68	14.47	14.34
David (14b)	23.50	22.34	21.58	20.97	20.56	20.33	20.20
Horse(12b)	15.95	15.12	14.41	13.88	13.55	13.34	13.23
Bunny(14b)	25.98	23.52	23.28	20.98	19.85	20.16	19.87
Venus(14b)	20.67	19.48	18.76	18.19	17.78	17.55	17.46

5 Results

To evaluate the performance of our geometry coder, we tested it on several meshes. The compression results obtained from some typical meshes are summarized in Tables 1–4. We compared the bits per vertex ratio achieved by our geometry encoder with the ratio obtained by the parallelogram rule of Touma and Gotsman (TG) [14], using a commercial version of their algorithm, implemented in the "virtue 3D optimizer" product of Virtue Ltd. For that purpose, we used a connectivity encoding which is competitive with that of TG.

As shown in Table 1, our encoder improves the bit per vertex (bpv) ratio by 2.3 bits on average for models that were pre-quantized to 12 bits per coordinate, and by 3.2 bits for models that were quantized to 14 bits. The Venus model is very detailed and the dinosaur is very sparse, so these models were processed only with 14 and 12 bits respectively.

We achieve better ratios than TG on the geometry. Since the geometry information is more dominant than the connectivity, we get better compression ratios. Moreover, as the dynamic range of the mesh (bits per coordinate) increases, the benefit from our algorithm increases. This includes the compression of other attributes, like UV coordinates, normals, and colors.

The performance of the k-way geometry prediction with respect to the 1-way prediction is quantified in Table 4 in terms of the number of bits of their entropy. The results of this comparison imply that we can save up to approximately 1 bit per vertex by using k-way prediction instead of 1-way. There are several ways to compute the 1-way prediction of each vertex. In our table we chose the best folding of 1-way prediction for each vertex. However, the prediction of a typical 1-way encoder is usually restricted to the order in which triangles are reconstructed. Hence the actual 1-way prediction is

somewhat worse than detected in the table, so the gap between 1-way and k-way is even bigger, and exhibits itself with more bits (see Table 1).

The vertices on the boundaries are efficiently encoded by a delta encoding, where the next vertex along the path is predicted based on the previous edge.

6 Conclusions and Future Work

We have introduced a lossless geometry compression technique for meshes. The technique is based on a k-way predictor with which we improve the geometry encoding. Our results show a significant improvement in the geometry encoding, that outperform the results given by the best-known method.

In the future we would like to improve the relaxation performed at the first step by using, for example, the average curvature value. This may further improve the first approximation that leads to a better compression of the patch geometry data. Another direction is to accompany the geometry encoding method by a connectivity-encoding scheme, such that the geometry predictions to place the vertices would match the connectivity traversal.

References

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

[2] Alliez P. and Desbrun M., *Progressive Compression for Lossless Transmission of Triangle Meshes*, SIGGRAPH '2001 Conference Proceedings, (2001) 198–205

[3] Alliez P. and Isenburg M., *Compressing Polygon Mesh Geometry with Parallelogram Prediction*, to appear in Proceedings of Visualization 2002 (2002)

[4] Cohen-Or D., Levin D. and Remez O., *Progressive Compression of Arbitrary Triangular Meshes*, IEEE Visualization '99, (1999) 67–72

[5] Gotsman C., Gumhold S. and Kobbelt L., *Simplification and compression of 3D meshes*, Proceedings of the European summer school on Principles of Multiresolution in Geometric Modeling (PRIMUS), Munich (2001)

[6] Gumhold S. and Stra W.ßer", *Real Time Compression of Triangle Mesh Connectivity*, SIGGRAPH 98 Conference Proceedings, Annual Conference Series, (1998) 133–140

[7] Isenburg M. and Snoeylink J., *Face Fixer: Compressing Polygon Meshes With Properties*, Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, (2000) 263–270

Fig. 4. Large triangular mesh used in our tables of results.

[8] Karni Z. and Gotsman C., *Spectral Compression of Mesh Geometry*, Siggraoh 2000, Computer Graphics Proceedings, (2000) 279–286

[9] Khodakovsky A., Schröder P. and Sweldens W", *Progressive Geometry Compression*, Siggraph 2000, Computer Graphics Proceedings, (2000) 271–278

[10] Pajarola R. and Rossignac J., *Compressed Progressive Meshes*, IEEE Transactions on Visualization and Computer Graphics, **6(1)** (2000) 79–93

[11] Rossignac J., *Edgebreaker: Connectivity Compression for Triangle Meshes*, IEEE Transactions on Visualization and Computer Graphics, **5(1)** (1999)

[12] Taubin G., *A Signal Processing Approach to Fair Surface Design*, Computer Graphics, **29** Annual Conference Series, (1995) 351–358

[13] Taubin G. and Rossignac J., *Geometric Compression Through Topological Surgery*, ACM Transactions on Graphics, **17(2)** (1998) 84–115

[14] Touma C. and Gotsman C., *Triangle Mesh Compression*, Graphics Interface'98, (1998) 26–34