Multiple-Description Geometry Compression for Networked Interactive 3D Graphics

Pavel Jaromersky jpavel@cis.poly.edu
Xiaolin Wu xwu@poly.edu
Yi-Jen Chiang yjc@poly.edu
Nasir Memon memon@poly.edu

Polytechnic University
Brooklyn, New York, U.S.A.

1. Introduction

In web-based visualization applications, such as internet computer games, tele-medicine and so on, 3D geometric data are typically communicated in a distributed and networked environment. The commonly used TCP/IP network protocols can provide reliable transmission, but they often introduce unpredictable delays. On the other hand, users in these applications can tolerate degradation of rendering quality when network conditions deteriorate, but not excessive delay or stoppage of animation sequences.

An existing technique for robust streaming of 3D graphics contents over lossy networks is multi-resolution coding of 3D geometry [8, 7, 2, 4]. An advantage of this approach is that it uses refinement layers and therefore multiple clients with different bandwidths can be served by a single unified code stream. However, there is a dependency between refinement layers, called *prefix condition*. Decoding of a given layer requires the knowledge of all the previous layers. A problem in the base layer reception interrupts the streaming all together and voids the remaining layers even though they are received perfectly.

To overcome this drawback, we propose an alternative approach to multi-resolution geometry coding, called *Multiple-Description Coding (MDC)* of 3D geometry. Instead of organizing code stream into embedded layers, MDC generates several separate descriptions of a geometric object, called *co-descriptions*. Each co-description of MDC can be independently decoded without any knowledge of other co-descriptions. Each extra successfully received co-description improves the fidelity of reconstructed geometry regardless of what has been received so far or in what order.

The MDC coding of geometric data is also desirable for distributed and networked storage systems. For robustness against hardware and connection failures, geometric datasets can reside on different disk drives or even on distant sites. If the datasets are MDC coded, then an application can simultaneously retrieve data from different storage devices. If any subset of the multiple requests are satisfied, then the rendering process can proceed with the received MDC coded data, whereas a multi-resolution code will impose a particular order of receiving different layers of data due to the prefix condition. The latter approach is clearly at disadvantage because the arrival order of data packets cannot be controlled by the receiver in these systems.

2. Problem Formulation

MDC geometry compression poses a hard combinatorial optimization problem even for two co-descriptions. Let us examine a bipartition of a 3D polygonal mesh M into two sub-meshes M_1 and M_2. Each sub-mesh M_i, $i = 1, 2$ portraits an approximation of the input mesh M. Given a distortion measure D, let $D(M_i)$ be the distortion between M and M_i. Let P_i be the probability of receiving i^{th} co-description successfully, which is independent of the reception of the other co-description. Then given M_1 and M_2 the expected distortion of the MDC compression is

$$\begin{aligned}\bar{D}(M_1, M_2) &= P_1P_2D(M) + P_1(1-P_2)D(M_1) \\ &\quad +(1-P_1)P_2D(M_2) \\ &\quad +(1-P_1)(1-P_2)D(\phi) \end{aligned} \tag{1}$$

where $D(M) = 0$ since the union of M_1 and M_2 will reconstruct the original 3D mesh M precisely; $D(\phi)$ is a constant independent of the bipartition assuming that a fixed approximation of M is reproduced if no co-description is received. Therefore, we can formulate the optimal two-description geometry compression problem as to minimize

$$\bar{D}(M_1, M_2) = P_1(1-P_2)D(M_1) + (1-P_1)P_2D(M_2) \tag{2}$$

over all possible bipartitions of M. This is a three-dimensional clustering problem under a complicated distortion measure, which is well known to be NP-complete [5]. Therefore we necessarily resort to heuristic algorithms to solve the problem. One heuristics is to produce two

sub-meshes M_1 and M_2 by a down-sampling of the original mesh M as uniformly as possible into an interleaved mosaic (see Section 4). The allocation of vertices of M into M_1 and M_2 is given by

$$\frac{m_1}{m_2} = \frac{P_1(1-P_2)}{(1-P_1)P_2}, \quad m_1 + m_2 = m \tag{3}$$

where m_1, m_2, and m are the numbers of vertices in the meshes M_1, M_2, and M respectively. The motivation is to make the quality of a co-description proportional to the probability of successful transmission of the corresponding channel. If more than two co-descriptions are desired, then one can always apply a two-description compression algorithm recursively to provide a tree-structured solution.

However, the decoder needs the connectivity information that associates vertices in 3D sub-meshes from different co-descriptions. The connectivity is a vital side information for the decoder to form joint descriptions. Special care is needed on how to split the vertex set and how to code the connectivity between the vertices in different sub-meshes.

3. Connectivity Side Information

Most 3D meshes consist of two types of data: connectivity and geometry. A key issue in MDC geometry compression is whether it is advantageous, and if so, how to split connectivity into several co-descriptions.

For simplicity consider only two co-descriptions. Denote the vertices in the first co-description by $1, 2, \ldots m$ and the vertices in the second co-description $m+1, m+2, \ldots, n$. Simplified connectivity of the mesh can be described by an adjacency matrix A: $A_{i,j} = 1$ if i and j are connected by an edge, and $A_{i,j} = 0$ if i and j are not connected. The adjacency matrix is symmetric. The original adjacency matrix A can be split into four sub-matrices as follows:

$$A = \left(\begin{array}{c|c} A_{1,1} & B \\ \hline B & A_{2,2} \end{array}\right).$$

From the structure of matrix A it is clear that some part of original connectivity information (matrix B) will be shared by the two descriptions. The decoder has to know the shared connectivity matrix B to merge the two descriptions into one. The matrix B represents the necessary merge side information in MDC coding of 3D meshes.

Fortunately, in practice the size of connectivity (less than 2 bits per triangle [10, 9, 3]) is rather small compared to the size of geometry (20–30 bits/vertex for coordinates quantized to a 16-bit integer [6]). Furthermore, many efficient methods for encoding the connectivity of a mesh have been developed [10, 11, 9, 3]. Therefore the merge side information B, does not have a big impact on the overall compression ratio.

Moreover, in many practical scenarios, the connectivity of a 3D dataset remains constant for the life time of the application, whereas the geometry data change constantly. For instance, for on-line interactive computer games, it is well justified in terms of geometry compression to distribute the global connectivity information of a large model to all clients ahead of time. Only the interactive geometry data will be streamed via the Internet in MDC code to improve the quality of network service. This method would be especially useful for the visualization of sophisticated, nonlinear interactions between the users and the model, such as warping and morphing.

4. Mesh Partition for Multiple Description Coding

As justified in Section 3, in MDC compression scheme we only split the geometry into subsets, one for each submesh, and code the connectivity of the entire mesh as the common side information for all co-descriptions. This global connectivity side information is later used during decompression to interpolate/estimate the missing vertices if not all co-descriptions are available, as well as to combine different co-descriptions to improve the quality of the reconstructed model if more than one co-description is received.

Our task of splitting the vertex set into subsets for the co-descriptions is to decide which vertices belong to which co-description. Intuitively, vertices included in each of the co-descriptions should be spaced evenly in the mesh, so that the missing vertices can be interpolated to the positions that are close to the original ones. In addition, the description of each vertex subset and the description of the global connectivity must be tightly coupled and efficiently encoded, in order to make use of the connectivity information to estimate the missing vertices and to combine different co-descriptions.

For densely sampled 3D objects, one way to divide the vertices into subsets whose vertices are spaced evenly is to take the graph whose nodes and edges are respectively the vertices and edges of the 3D mesh, and construct a *vertex spanning tree* T of this graph. To divide the vertices into p subsets of roughly the same size, we can fix some degree-1 node of T as the root, defining for each node its level in the tree (where the root has level 0), and assign each vertex at level ℓ to the i-th subset, where $i = \ell \bmod p$. In this way, if a vertex v is missing from one co-description, at least the near neighbors are included and can be used to estimate v. Moreover, some efficient triangle mesh compression algorithms, such as the *topological surgery* approach of Taubin and Rossignac [10], already use a vertex spanning tree to compress the connectivity information. Therefore, the connectivity information can be efficiently encoded and tightly coupled with the encoding of the vertex spanning tree. This will serve our purpose of MDC compression, with only a

very small additional overhead.

In our MDC compression algorithm, we use the topological surgery approach [10] to encode the connectivity information. To compress the geometry information in cases where code length is very important, we develop our own *surface-based predictor*. We also deal with the missing vertices, in both encoding and decoding of co-descriptions. These techniques will be discussed in Section 5.

5. Surface-Based Predictive Coding of Geometry

In accompany with connectivity coding with the topological surgery approach [10], we carry out predictive coding of the coordinates of the vertices. The prediction can be done fast by the *parallelogram rule* [11], or - if extra processing power is available - better yet by our own *surface-based predictor*.

The existing prediction methods used in compression of 3D meshes are planar in nature [10, 11, 6], assuming a flatness (small curvature) in local geometry. On a second reflection, if we want to make the code length shorter, higher order predictors should work better than linear type of predictors. To exploit the spatial coherence of a coarse mesh in MDC geometry compression, the predictor needs to capture the global trend of the underlying surface. To this end we fit the six previously coded vertices that are topologically closest to the current vertex into a quadratic surface

$$z' = a_1 x'^2 + a_2 y'^2 + a_3 x' y' + a_4 x' + a_5 y' + a_6. \quad (4)$$

Given the six closest encoded vertices, first their interpolating plane is computed and then the coordinates are transformed such that the interpolated plane becomes $x'y'$ plane of the new coordinate system. The coefficients a_i, $1 \leq i \leq 6$, are determined from the chosen six neighboring vertices in transformed coordinates (x'_i, y'_i, z'_i), $1 \leq i \leq 6$. The transformation tries to minimize variance in the z' coordinate for better fit as well as to remove degenerate cases, where the neighboring vertices would not define a surface (one z value for every (x, y) pair) in the original coordinate system. Under the assumption that the next vertex is close to the resulting quadratic surface, we want to select a point $(\hat{x}', \hat{y}', \hat{z}')$ on this surface as the prediction, and then transform the point back to original coordinate system - $(\hat{x}, \hat{y}, \hat{z})$.

Using the quadratic surface fitting the neighborhood data as one constraint, two more constraints are needed to uniquely determine the prediction point $(\hat{x}', \hat{y}', \hat{z}')$. These additional constraints should be selected according to some domain knowledge about the 3D triangle mesh. Typical triangulations of 3D meshes consists of roughly equilateral and/or isosceles triangles. This observation suggests a way of designing the predictor.

Referring to Fig. 1, let $\mathbf{v}_t = (x_t, y_t, z_t)$, $1 \leq t \leq 3$, be the three vertices of the triangle that is opposite to the vertex $\mathbf{v}$ to be predicted. Then we can set up the two additional constraints by assuming that $\mathbf{v}$ has equal distance to $\mathbf{v}_1$ and $\mathbf{v}_2$, and that $\mathbf{v}_1$ has equal distance to $\mathbf{v}$ and $\mathbf{v}_3$. Namely,

$$\begin{aligned} (\hat{x} - x_1)^2 + (\hat{y} - y_1)^2 + (\hat{z} - z_1)^2 &= \\ (\hat{x} - x_2)^2 + (\hat{y} - y_2)^2 + (\hat{z} - z_2)^2 & \end{aligned}$$

$$\begin{aligned} (\hat{x} - x_1)^2 + (\hat{y} - y_1)^2 + (\hat{z} - z_1)^2 &= \\ (x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2 & \end{aligned} \quad (5)$$

Solving the system of three equations in (4) and (5) for $(\hat{x}, \hat{y}, \hat{z})$, we obtain a prediction of the location of the next vertex.

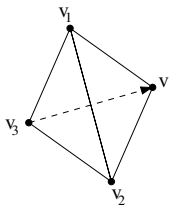

Figure 1. Surface-based prediction

When we encode an individual co-description, we avoid using any missing vertices in the prediction to prevent cumulative errors. When we decode, we first recover the available vertex positions (possibly from more than one co-description), and then estimate the positions of the missing vertices from local neighbors. This can be done by using the vertex spanning tree (interpolating from both ancestors and descendants), or by using the parallelogram rule or our surface-based predictor. The estimation can be viewed as a prediction without correcting the predictive error, and thus any prediction scheme can be used; the better the prediction, the better the quality of the reconstructed model.

6. Experimental Results

In order to evaluate our MDC algorithm for both symmetric channels ($P_1 = P_2$) and asymmetric channels ($P_1 \neq P_2$), we ran it for both even and uneven partitions of the input 3D mesh. The allocation of vertices of input mesh M to the two sub-meshes M_1 and M_2 is governed by Equation (3).

Table 1 lists the results of two-description compression in comparison with the single-description algorithms. The table contains file sizes in bytes of four 3D triangle meshes compressed by: (a) the single-description compression algorithm of topological surgery [10] using the linear predictor (based on 4 ancestors in the vertex spanning tree) and the

Figure 2. Reconstruction results of the Bunny and Triceratops models.

parallelogram rule; (b) the proposed MDC compression algorithm in both co-descriptions and for both even (1:1) and uneven (2:1) partition of the input meshes; (c) the k-d tree-based multi-resolution compression algorithm [4];

Dataset	Bunny	Horse	Dino	Tricer.
Vertices	34834	48485	56194	2832
Triangles	69451	96966	112384	5660
Single Desc.				
Connectivity	14191	15047	18464	1312
Linear pred.	76717	92626	104766	7865
Parallelogram	54690	70507	83540	7431
MDC 1:1				
Co-desc. 1	38693	47304	55915	4322
Co-desc. 2	38510	47212	56071	4355
Total	77203	94516	111986	8677
MDC 2:1				
Co-desc. 1	52547	64851	74370	5612
Co-desc. 2	29343	35879	41791	3179
Total	81890	100730	116161	8791
Multiresolution				
Connectivity	17380	24219	31369	2278
Geometry	61327	71491	70694	6378

Table 1. Results

For multiple-description geometry compression, if we sum up the sizes of three compressed files: connectivity, co-description 1 and co-description 2, the total code length is only slightly larger than the total code length of the multi-resolution method.

Finally, we show in Figure 2 reconstructed models of Bunny (MDC 2:1, from left: co-desc. 1, co-desc. 2, both co-descriptions) and Triceratops (MDC 1:1, from top: co-desc. 1, co-desc. 2, both co-descriptions). Comparing the image quality, we can clearly see that when only one co-description is received, no matter which one, our approach can still reconstruct an approximate model that is very close to the original one.

References

[1] G. Al-Regib, Y. Altunbasak, and J. Rossignac. An unequal error protection method for progressively compressed 3-d meshes. In *Proceedings of IEEE INFOCOM*, 2002.

[2] P. Alliez and M. Desbrun. Progressive encoding for lossless transmission of 3d meshes. In *Proceedings of ACM SIGGRAPH*, pages 198–205, 2001.

[3] P. Alliez and M. Desbrun. Valence-driven connectivity encoding for 3D meshes. *Proc. Eurographics 2001*, pages 480–489, 2001.

[4] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. Graphics*, 212(3):372–379, 2002. Special Issue for SIGGRAPH '02.

[5] M. Garey, D. S. Johnson, and H. S. Witsenhausen. The complexity of the generalized lloyd-max problem. *IEEE Trans. on Inform. Theory*, 28:255–266, 1982.

[6] M. Isenburg and P. Alliez. Compressing polygon mesh geometry with parallelogram prediction. In *Proc. Visualization*, pages 141–146, 2002.

[7] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of ACM SIGGRAPH*, pages 271–278, 2000.

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

[9] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Visualization Computer Graphics*, 5(1):47–61, 1999.

[10] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Trans. Graphics*, 17(2):84–115, 1998.

[11] C. Touma and C. Gotsman. Triangle mesh compression. In *Proc. Graphics Interface*, pages 26–34, 1998.