PROGRESSIVE STREAMING OF TEXTURED 3D MODELS OVER BANDWIDTH-LIMITED CHANNELS

Dihong Tian and Ghassan AlRegib

Center for Signal and Image Processing
Georgia Institute of Technology
{dhtian,gregib}@ece.gatech.edu

ABSTRACT

The bitstream of a progressively encoded textured model consists of multiple refinement layers. Decoding each layer produces a model with a simplified mesh and a resolution-reduced texture. The authors in [4] proposed a quality measure that captures the visual fidelity of the multi-resolution textured models. Based on the quality measure, in this paper, we consider the problem of streaming progressively encoded textured models over a bandwidth-limited channel. We develop a bit-allocation algorithm that optimally packetizes the source bits in every transmitted data unit such that the perceptual quality of the model displayed on the client's screen is maximized. The experimental results confirm the effectiveness of the proposed bit-allocation algorithm.

1. INTRODUCTION

A textured 3D model consists of a geometric mesh parameterized by a texture such as a 2D image, which adds realism or surface detail to the geometry. The mesh and the texture can be progressively compressed to generate multi-resolution representations [1, 2], and further provide multiple levels-of-detail (LOD) of the textured model [3, 4]. Every LOD is composed of a simplified mesh and a resolution-reduced texture. Progressive compression facilitates not only the storage of the model in a memory-limited device, but also the transmission of the model over a bandwidth-limited channel, as depicted in Figure 1. The coarsest pair of mesh and texture is first transmitted to quickly start the display of the model on the client's screen. Then, the enhancement data for both the mesh and the texture is delivered to gradually refine the displayed model. Frequently, the refinement process is prone to be terminated by the client before all the enhancement data is downloaded, because 3D models are typically presented by large data sets while the channel bandwidth is limited. In such cases, delivery of the full-resolution model is not of major concern, but the clients desire to *maintain* the best approximated model with currently received data at any instant before the refinement process is possibly terminated.

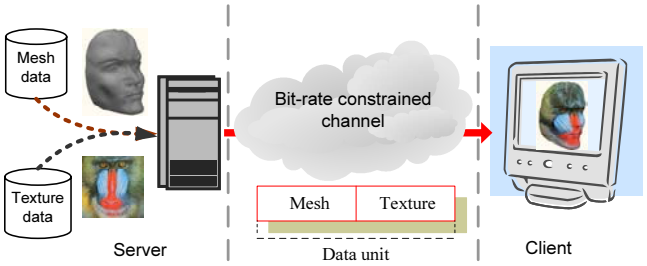


Fig. 1. An illustration of the scenario under consideration.

This paper addresses the above challenge. More explicitly, for a progressively encoded textured model, assuming that the coarsest representation has been decoded and displayed, we study *under the bit-rate constraint of the channel, how the enhancement data should be packetized so that decoding next data unit increases the visual quality of the displayed model to the maximum extent*.

To provide a solution, we propose a bit-allocation algorithm based on a quality measure that properly predicts the visual fidelity of different LODs for a progressively encoded textured model [4]. Under the bit constraint on every data unit, the proposed algorithm searches for the optimal distribution of the bits between mesh and texture data and packetizes the data units accordingly, thus providing the maximum quality increment. The algorithm has a linear computation complexity, and the experimental results show that the proposed algorithm outperforms the heuristical methods both objectively and subjectively.

The rest of the paper is organized as follows. Section 2 explains the progressive compression method. Section 3 introduces the objective measure used to evaluate the visual quality of the textured models with different resolutions. In Section 4, we present the bit-allocation algorithm based on the quality measure. Empirical comparison of the proposed algorithm with several heuristical methods are presented in Section 5 while Section 6 concludes the paper.

Thanks to Peter Lindstrom for providing the test models.

2. PROGRESSIVE COMPRESSION

In this section, the progressive compression methods for both the geometry and the texture are summarized. There are several methods available to code textures and we do not plan to devote effort toward developing new ones. Instead, in the paper we limit the texture to 2D images, and work with published image compression algorithms. Specifically, we adopt the wavelet-based encoder known as SPIHT (Set Partitioning in Hierarchical Trees) [2] to encode the texture image into a progressive bitstream.

Mesh compression is a more complex process as it has the geometric vertices, the edges, and the texture coordinates that parameterize the mesh surface to the texture plane. Generally, a progressive mesh is generated by successively performing *edge-collapse* operations in the order of increasing geometric error [1]. On the decoding side, an inverse operation called *vertex-split* is conducted to recover the simplified mesh.

The mesh compression algorithm implemented in the paper similarly follows that in [5] while having two major differences: (i) The cost of an edge collapse is measured by the *texture deviation* [6], i.e., the maximum distance from points on the simplified mesh to their correspondents on the input mesh with the same texture coordinates; (ii) Each vertex of the mesh is treated as a vector $V \in \mathbf{R}^5$: three spatial coordinates and two texture coordinates. We then compress the edge-collapses using vertex prediction followed by entropy coding for the prediction error [5]. For each edge-collapse operation, five variable length codewords (VLC) are appended after the collapse status, the split direction, and the cut edges[1] are respectively coded.

3. QUALITY MEASURE

The aforementioned algorithms encode the textured model into a base mesh with a number of edge-collapses that refine the base mesh to higher levels-of-detail, and similarly a multi-resolution representation for the texture. Mapping a lower-resolution texture to a simplified mesh gives an approximated representation of the original textured model with certain distortion. It is crucial to properly measure such distortion so as to reflect the visual difference of the displayed models.

An effective computational measure has been proposed in [4] where the quality degradation incurred by the simplified mesh and the lower-resolution texture are combined through an *equalization* factor. Mathematically, if the textured model is composed of a simplified mesh, M_i, and a lower-resolution texture T_j, then the quality of the approximated model is given by

$$\mathcal{Q}(M_i, T_j) = \lambda \mathcal{Q}_{\mathcal{G}}(M_i) + (1-\lambda)\mathcal{Q}_{\mathcal{T}}(T_j), \qquad (1)$$

where $\lambda \in [0,1]$ is the equalization factor, and $\mathcal{Q}_{\mathcal{G}}$, $\mathcal{Q}_{\mathcal{T}}$ are computed using errors measured in the mesh and texture compression processes, respectively:

$$\mathcal{Q}_{\mathcal{G}}(M_i) = norm[\log_{10}(1 - \frac{\text{MSD}(M_i)}{L^2})], \qquad (2)$$

$$\mathcal{Q}_{\mathcal{T}}(T_j) = norm[\log_{10}(1 - \frac{\text{MSE}(T_j)}{255^2})]. \qquad (3)$$

L in (2) is the diagonal of the bounding box of the mesh; MSD is the mean squared texture deviation over all collapsed edges, and MSE is the mean squared pixel error for the texture image. Function $norm[\cdot]$ is defined as a *normalization* operator that normalizes the scales of $\mathcal{Q}_{\mathcal{G}}$ and $\mathcal{Q}_{\mathcal{T}}$ into $[0,1]$, respectively[2]. Conceptually, (2) and (3) quantify the quality of the coarser mesh and the lower-resolution texture in their own domains, and (1) measures the visual quality of the textured model by accounting for the inter-effect of mesh and texture through the factor λ. Depending on the features of the mesh geometry and the mapped texture, the equalization factor for a particular model is estimated using error samples measured in the rendering space. Because of the space limitation, we skip the details of the algorithm. Interested readers are referred to [4] for further details. As an example, Figure 2 presents sampled mesh and texture pairs of the MAP-SPHERE and the MARBLE-BALL models.

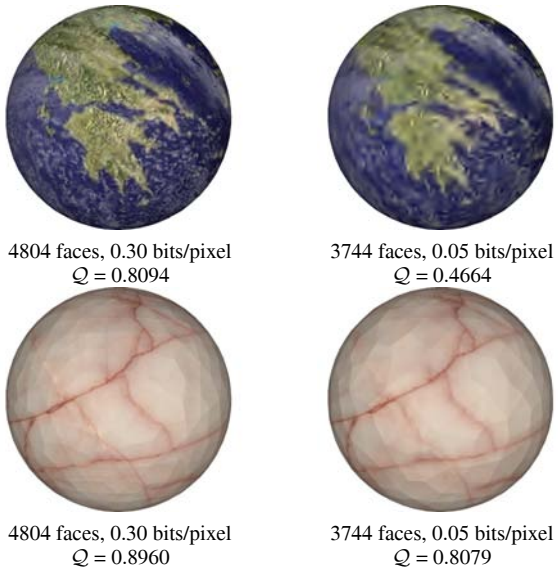


Fig. 2. Captured views of the MAP-SPHERE ($\lambda = 0.5269$) and the MARBLE-BALL ($\lambda = 0.9127$) models.

[1] *collapse status*: one-bit flag specifying whether a vertex is to be split; *split direction*: one-bit flag specifying the vertex-split direction; *cut edges*: $\left\lceil \log_2 \binom{e}{2} \right\rceil$ bits specifying the two cut edges among the e incident edges for a vertex that needs to be split.

[2] After normalization, the lowest resolution mesh (or texture) corresponds to $\mathcal{Q}_{\mathcal{G}}$ (or $\mathcal{Q}_{\mathcal{T}}$) $= 0$, and the full resolution version is measured as $\mathcal{Q}_{\mathcal{G}}$ (or $\mathcal{Q}_{\mathcal{T}}$) $= 1$.

The quality measures are calculated using (1) with the corresponding values of λ where the larger the value of λ the more fidelity information of the model is conveyed by the mesh geometry and vice versa.

4. THE BIT-ALLOCATION ALGORITHM

Recall the scenario demonstrated in Section 1, where the coarsest version of a textured 3D model is displayed at the client, and is being progressively refined every time when a new data unit is decoded. Although the particular size of the data unit varies among different environments, it is generally constrained by a *maximum data unit* (MDU) in a bit-rate constrained channel. Our objective is to optimally packetize the size-limited data units so that the greatest increment of model quality is obtained at each instant when the model is refined. In the remainder of this section, we present a solution to such a process based on the quality measure introduced in Section 3.

We assume in general that the size of MDU is $\mathcal{K}$ bits. Also, we use $(\chi_{\mathcal{G}}^{(i)}, \chi_{\mathcal{T}}^{(i)})$ to denote the data bits that have been decoded by the client for the mesh and the texture, respectively, after the i-th refinement. Specifically, $(\chi_{\mathcal{G}}^{(0)}, \chi_{\mathcal{T}}^{(0)})$ represents the initial model that has been viewed by the client. The bit allocation solution for the $(i+1)$-th refinement is given by

$$\begin{cases} (\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{opt} = \underset{\Delta\chi_{\mathcal{G}}+\Delta\chi_{\mathcal{T}} \le \mathcal{K}}{\arg\max}\, \mathcal{Q}(\chi_{\mathcal{G}}^{(i)} + \Delta\chi_{\mathcal{G}}, \chi_{\mathcal{T}}^{(i)} + \Delta\chi_{\mathcal{T}}; \lambda), \\ (\chi_{\mathcal{G}}^{(i+1)}, \chi_{\mathcal{T}}^{(i+1)}) = (\chi_{\mathcal{G}}^{(i)}, \chi_{\mathcal{T}}^{(i)}) + (\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{opt}, \end{cases} \tag{4}$$

where $\mathcal{Q}$ is the quality measured using (1), and λ is the equalization factor associated with the particular textured model. $(\Delta\chi_{\mathcal{G}}, \Delta\chi_{\mathcal{T}})_{opt}$ denote the numbers of bits for the mesh and the texture, respectively, according to which a new data unit will be packetized for next transmission. Starting from $(\chi_{\mathcal{G}}^{(0)}, \chi_{\mathcal{T}}^{(0)})$, the computation in (4) is repeated until all the refinement data is transmitted or the transmission is terminated by the client.

The proposed algorithm in (4) is referred in the paper as the *maximum quality bit-allocation* (MxQ-BA) algorithm. For each decoding opportunity, the optimal bit-allocation solution of (4) is searched over the solution space. Exhaustive search is not necessary because the quality measure, $\mathcal{Q}$, has a decoupled structure and is monotone as a function of the resolutions of mesh and texture. Henceforth, the optimal operating point can be found on the boundary of the feasible region determined by the bit-rate constraint ($\mathcal{K}$ bits/unit), which results in linear computation complexity.

In general, the solution space should be constructed by all possible combinations of a certain number of edge collapses and a portion of enhancement bits of texture. Practically, it is neither efficient nor necessary to investigate the problem for every single edge-collapse or few texture bits as the difference in both the bit-rate and the quality will be trivial. In our experimental study presented next, we organize the refinement data of the mesh and the texture into batches with bit-rate 100 bytes/batch. The size of MDU is chosen to be 6KB or 8KB. At each transmission opportunity, we decide under the bit constraint how many batches of the remaining mesh data should be packed into the to-be-sent data unit and how many batches should be allocated to the remaining texture data.

5. EXPERIMENTAL RESULTS

We have tested the proposed algorithm on various models and in this section we report the results for the ZEBRA and the SALAMANDER models with equalization factors found to be $\lambda_z = 0.6276$ and $\lambda_s = 0.8990$, respectively.

We first compare the proposed MxQ-BA algorithm with the simplest heuristics that transmit first all the mesh (texture) data and then the texture (mesh). These simple heuristics are called *M-First* and *T-First*, respectively. For the ZEBRA model, Figure 3 plots the quality increments achieved by each data unit, where the (black) points-curve denotes the results by MxQ-BA while the (red) triangles-curve and (green) diamonds-curve denote the results of the heuristics, respectively. As can be seen from the plots, the MxQ-BA algorithm improves the model quality more quickly than simply streaming the geometry (texture) data first and then the other until the delivered model approaches close-original quality ($\mathcal{Q} > 0.90$). In Figure 3, for example, the quality obtained by M-First after sending 8 data units is $\mathcal{Q} = 0.4842$ whereas $\mathcal{Q} = 0.7818$ is achieved by MxQ-BA. Subjective comparison of the two models are shown in Figure 4. It is apparent that the model given by the MxQ-BA algorithm (Figure 4(a)) has higher visual quality than that of M-First (Figure 4(b)).

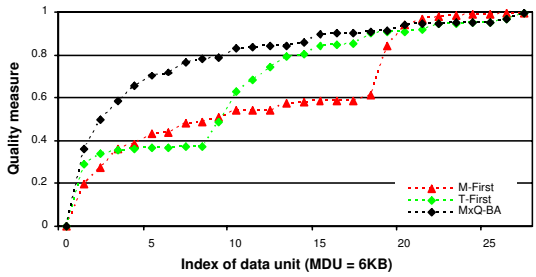
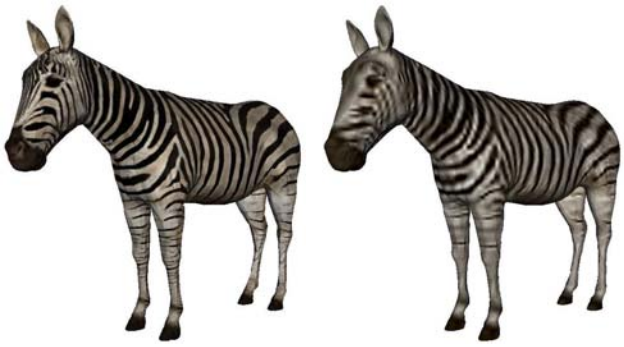


Fig. 3. MxQ-BA vs. M-First and T-First heuristics for the ZEBRA model.

Two more sophisticated heuristical methods are also investigated and the results are presented in Figure 5. The *Interleaved* scheme organizes the mesh and the texture data in an interleaving fashion. The *Half-BA* algorithm is a simple bit-allocation approach that equally distributes bits for the mesh and the texture within a data unit. As shown in



(a) MxQ-BA: $\mathcal{Q} = 0.7818$ (b) M-First: $\mathcal{Q} = 0.4842$

Fig. 4. Subjective comparison of MxQ-BA with M-First after decoding 8 enhancement data units for the ZEBRA.

Figure 5, the proposed MxQ-BA algorithm outperforms the Half-BA heuristic as well as the Interleaved scheme. The enhanced performance is more substantial for the SALAMANDER model and/or in the cases where the data unit has larger size limit. This observation is anticipated because the solution space becomes smaller when the size of data units decreases, hence lowering the possible gain that can be achieved by finding the optimal solution. Figure 6 shows for Figure 5(c) the rendered results of the above methods after decoding 3 data units, where Figure 6(b-d) corresponding to the models obtained by MxQ-BA, Half-BA and the Interleaved scheme, respectively. Notice, for example, how the claws become substantially distorted from Figure 6(b) to Figure 6(d). These subjective comparison confirms the effectiveness of the proposed MxQ-BA algorithm.

6. CONCLUSIONS

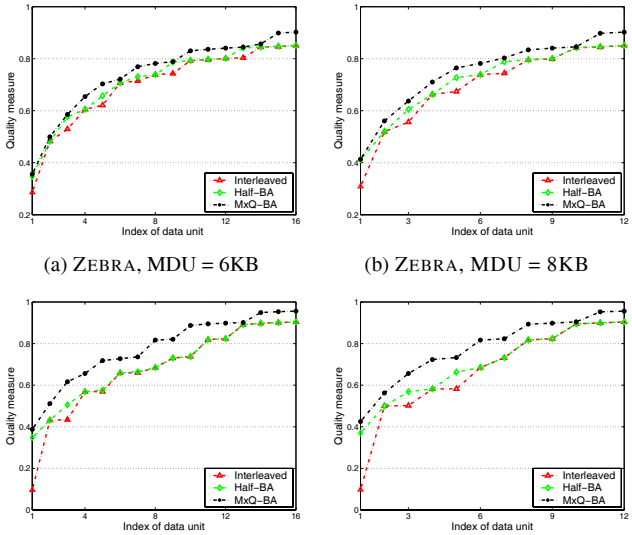
In this paper, we presented a streaming algorithm for textured models based on a novel bit-allocation algorithm that optimally distributes source bits between mesh and texture so that every size-limited data unit conveys the most information of the model fidelity to the client. We solve the optimization problem at each transmission opportunity using boundary analysis, which has linear complexity. The experimental results show the effectiveness of the proposed algorithm both objectively and subjectively.

7. REFERENCES

[1] H. Hoppe, "Progressive meshes," in *Proc. of ACM SIGGRAPH 1996*, pp. 99–108.

[2] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 6, no. 3, pp. 243–250, 1996.

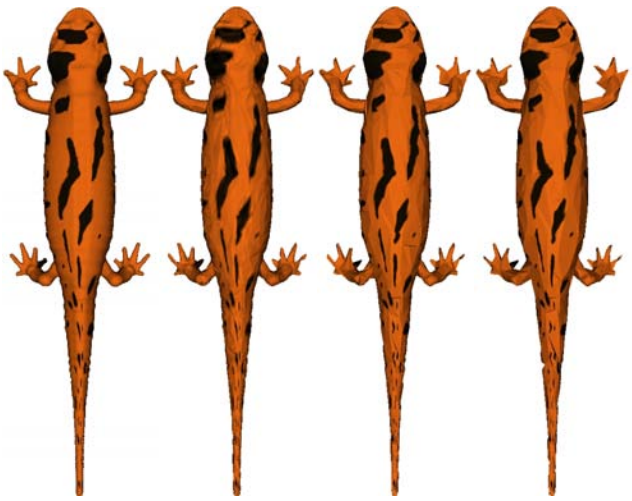
[3] Laurent Balmelli, "Rate-distortion optimal mesh simplification for communications," Ph.D. dissertation No 2260, Ecole Polytechnique Federale de Lausanne, Switzerland, 2001.



(a) ZEBRA, MDU = 6KB (b) ZEBRA, MDU = 8KB

(c) SALAMANDER, MDU = 6KB (d) SALAMANDER, MDU = 8KB

Fig. 5. Comparison of the MxQ-BA algorithm with the Interleaved scheme and the Half-BA heuristic.



Original (a) $\mathcal{Q} = 0.6164$ (b) $\mathcal{Q} = 0.5057$ (c) $\mathcal{Q} = 0.4343$ (d)

Fig. 6. Sampled views of SALAMANDER model from Figure 5(c) after decoding 3 enhancement data units.

[4] D.H. Tian and G. AlRegib, "FQM: a fast quality measure for efficient transmission of textured 3D models," in *Proc. of ACM Multimedia 2004*, pp. 684–691.

[5] R. Pajarola and J. Rossignac, "Compressed progressive meshes," *IEEE Trans. Visualization and Computer Graphics*, vol. 6, no. 1, pp. 79–93, 2000.

[6] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *Proc. of ACM SIGGRAPH 1998*, pp. 115–122.