PODS: PARTIALLY ORDERED DELIVERY FOR 3D SCENES IN RESOURCE-CONSTRAINED ENVIRONMENTS

Dihong Tian and Ghassan AlRegib

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

ABSTRACT

Three-dimensional (3D) graphic scenes require considerable network bandwidth to be transmitted and computing power to be rendered on users' terminals. Toward high-quality display in real time, we propose a sender-driven mechanism for streaming 3D scenes in a resource-constrained environment. By pre-processing the database, objects in the scene are properly weighted upon their rendering importance, and their resolutions are selected accordingly to reduce the bit rate. Partially ordered delivery is then performed using decoding independencies between the objects. Simulation results show the efficacy of the proposed mechanism. For a test benchmark, for example, the proposed algorithm outperforms the comparing heuristic by 4 dB under a 100-KB bit rate.

1. INTRODUCTION

One category of 3D graphics applications, e.g., 3D maps, require online navigation of large virtual environments for which replication of the entire database at the client is not practical. Considering the insufficient bandwidth and computing resource of the client device, sending all 3D objects with large data dimension may not only overload the network, but also overload the user device when the models are rendered. On the other hand, objects in a 3D scene are in general with unequal importance depending on such factors as the screen dimension, the object coordinates, the view distance, etc. For a given view perspective and a certain quality requirement, transmitting all 3D meshes with high tessellation and/or all textures with high resolution to the device may be beyond the client's actual need and therefore become unnecessary. For example, one object may be occluded by another in the scene, or may be outside of the view space and will be culled away in the client's rendering pipeline. A coarse representation for the object would be sufficient while more bandwidth and computation can be spent on objects for which high levels of detail (LODs) are desired. For high-quality display with optimal use of limited resource, transmission and rendering scalability becomes important for networked graphics applications.

Previous research on scalable coding and transmission focused on individual 3D models [1–5] while essences of 3D scenes have not been well addressed. As a new contribution, in this paper we propose a joint application and transport mechanism for streaming 3D scenes under a low bit-rate and low computing-power environment. In doing so, the scene database is pre-processed at the server, which estimates the relative importance of the objects in rendering the scene. Based on the estimated weights, proper resolutions of the objects are selected, and are transmitted over multiple streams using a newly proposed IP transport protocol named SCTP (Stream Control Transmission Protocol) [6]. Performance evaluation shows that, using the properties of 3D scenes, the proposed streaming mechanism greatly improves the transmission efficiency and significantly reduces the amount of 3D data processed by the client while preserving the quality.

The paper is organized as follows. After briefly summarizing the background in Section 2, we present in Section 3 a distortion measure that facilitates evaluating the quality of a resolution-reduced 3D scene. The streaming mechanism is studied in detail in Section 4, and test results in a simulated network environment are given in Section 5. Section 6 concludes the paper.

2. BACKGROUND

Multi-resolution coding of 3D models (e.g. [1–3]) is a partial solution to provide transmission and rendering scalability. Using multi-resolution encoders, the server can select the appropriate resolution for a particular client according to its quality requirement, or initially sends a coarse representation of the 3D model to the client for quick reconstruction and rendering, and then transmits upon necessity refinement layers which allow the client to increase model fidelity toward higher resolutions. Multi-resolution methods are successful in exploring the space and time efficiency of a 3D object. However, when considering 3D scenes which contain multiple objects in the same view space, interactions of the objects with and within the view space need to be taken into account. Depending on the coordinates and the view space, the objects may require different LODs to be displayed with desired quality, or may not need to be rendered at all if, for example, the object falls outside of the view space.

Regarding transmission, a single object requires sequenced delivery of the mesh stream because further decoding higher LODs is based on successfully decoding preceding LODs. Nevertheless, ordered delivery for multiple objects is not a requirement. Transmitting multiple objects in a parallel fashion can accelerate the receiving process as objects can be decoded independently. Unfortunately, the long-existing transmission control protocol (TCP) and the user datagram protocol (UDP) provide either strictly sequenced delivery or unordered delivery at all. A hybrid version of the two protocols [4,5] also does not produce the desired flexibility. Although, having multiple TCP connections can possibly provide partially sequenced delivery, it complicates the design of the transport layer and may violate bandwidth fairness to other applications in the network.

The stream control transmission protocol (SCTP) [6] is a state-of-the-art IP transport protocol which embeds ordered and unordered delivery in one connection (called *association* in SCTP) and always retains TCP-friendly congestion control and congestion avoidance. SCTP allows separated data streams from the upper-layer application to be multiplexed into one association, and maintains respec-

tively the order of data units on different streams. Thus, if one data unit belonging to a certain stream is lost, succeeding data units from that stream will be stored in the receiver's stream buffer until the lost data is retransmitted from the source. In the meantime, data from other streams can still be passed to upper-layer applications for prompt processing. Motivated by this property, our streaming mechanism proposed in this paper is built atop an SCTP-based transport layer, which transmits 3D objects over multiple streams with properly selected resolutions. We present the detailed design of the mechanism and a performance evaluation in the next sections.

3. MEASURING SCENE QUALITY

One main objective of the streaming mechanism is to best preserve the scene quality with minimized amount of 3D data processed by the client device. To accomplish this goal, we first apply multi-resolution coding [3] to generate, for each object, a sequence of LODs that gradually improve the model fidelity with increased bit rates. To evaluate the quality of the scene with multi-resolution objects, we introduce a metric derived from a weighted summation of individual distortion measured for the objects. In the following, we present several definitions toward the development of this novel quality metric.

Definition 1 For two given surfaces, S and S', the root-mean-square (RMS) surface distance and the maximum surface distance from S' to S are respectively defined as

$$\mathcal{E}_{rms}(S', S) = \left(\frac{1}{S'} \int_{S'} e^2(v, S) ds \right)^{1/2} \tag{1}$$

and

$$\mathcal{E}_{max}(S', S) = \max_{v \in S'} e(v, S), \tag{2}$$

where $e(v, S)$ is the point-to-surface distance given by

$$e(v, S) = \min_{v' \in S} d(v, v'),$$

and $d()$ is the Euclidean distance between two points in $\mathbb{E}^3$.

Definition 2 For a geometric mesh with multiple resolutions $\{M^i\}$, $i = 0, ..., L$, where M^0 is the base mesh and M^L is the full resolution, we define the normalized distortion of an LOD M^i as

$$\mathcal{D}(M^i) = \frac{\mathcal{E}_{rms}(M^i, M^L)}{\mathcal{E}_{max}(M^0, M^L)}. \tag{3}$$

The maximum and root-mean-square surface distances between the corresponding pairs of meshes can be calculated using the fast Metro tool [7] in practice. The metric in (3) is further extended to measure the distortion of a 3D scene which has multiple meshes with potential difference in rendering.

Definition 3 Given a 3D scene $\mathcal{S}$ with N multi-resolution objects, $\{M_k^i\}$, $k = 1...N, i = 1...L_k$, we define the distortion of the scene $\mathcal{S}$ as

$$\mathcal{D}_s = \sum_{M_k^{i_k} \in \mathcal{S}} \omega_k \cdot \mathcal{D}(M_k^{i_k}), \quad 0 \le \omega_k \le 1, \tag{4}$$

and

$$PSNR = -20 \log_{10} \mathcal{D}_s \text{ (dB)}, \tag{5}$$

where $\mathcal{D}(M_k^{i_k})$ denotes the measured distortion for mesh M_k with resolution i_k; $\{\omega_k\}_{k=1...N}$ are defined as normalized weight factors to reflect the relative importance of the objects, $\sum_{k=1}^{N} \omega_k = 1$.

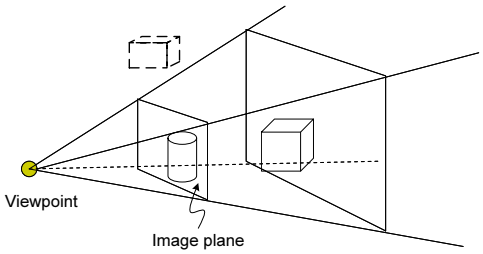


Fig. 1. A view frustum is defined by six planes.

4. THE STREAMING MECHANISM

A remaining challenge in the development of (4) is the determination of the weight factors. As the first component in the streaming mechanism, we describe in the following an object weighting process. After that, we study the proper selection of object resolutions using a rate-distortion optimization framework. Upon transmission, an SCTP-based transport layer is used and the selected LODs are transmitted over multiple streams in a partially ordered manner.

4.1. Object Weighting

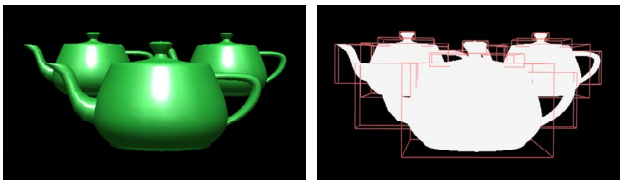
The weight factor of an object is expected to reflect the relative importance of the object in rendering the scene. As shown in Figure 1, given a view perspective, a *view frustum* (VF) is defined by six planes where two are parallel to each other. Intuitively, determined upon the coordinates, objects falling outside of the view frustum will not be visible and hence have no contribution to the display quality. Also, some objects may be partially or completely occluded by other objects that are "in front of" them. In general, an object closer to the viewpoint and with a smaller view angle is subject to a higher weight of importance than an object located further away or with a larger angle to the viewpoint.

Driven by the above discussion, our object weighting algorithm is composed of two passes. The first step of the algorithm is to perform a fast view-frustum testing and its output are two sets of objects: $\mathcal{S}_1 \triangleq$ {Objects outside of VF}, and $\mathcal{S}_2 \triangleq$ {Objects within or intersected with VF}. Apparently, we have $\omega(\mathcal{O}_j) = 0$ for $\mathcal{O}_j \in \mathcal{S}_1$. For the objects in $\mathcal{S}_2$, three actions are taken prior to the estimate of the weight factors: First, the bounding volume (BV) of each object is constructed and the center of the BV is calculated. Then, the distance between the viewpoint and the center of the BV is calculated and used as an estimate of the distance from the viewpoint to the object. Finally, these distances are sorted with the nearest objects first and the centering objects first (for objects that have equal distances to the viewpoint).

In the second pass, the image plane is uniformly divided into $h \times w$ grids, which will be used to evaluate the coverage of an object's projection on the image plane. Each grid is weighted by the reciprocal of its distance to the viewpoint, i.e.,

$$\omega_{\mathbf{v}} \propto \frac{1}{\| \mathbf{v} - \mathbf{v}_0 \|} \tag{6}$$

where $\mathbf{v}$ and $\mathbf{v}_0$ denote the center coordinates of the grid and the coordinates of the viewpoint, respectively. Initially, all the grids are marked as "unused for weighting". Then, for each object in the sorted list, a *bounding volume projection map* (BVPM) is generated which, as the name implies, is the projection of the object's bounding volume on the image plane. The overall weight of the grids covered



(a) (b)

Fig. 2. An illustration on the projection-map based object weighting.

by the BVPM is used to evaluate the object weight. Once a grid is counted, it is marked as "used for weighting" and becomes no longer usable for the succeeding objects. At the end of the process, normalization is performed so the summation of the object weights is equal to 1. Figure 2 presents a demonstration on the projection maps. Note that in Figure 2, each object has several partitions and the bounding box for each partition is created, which formulates an overall bounding volume for the entire object. When computing the object weights, the bounding boxes for the partitions may be processed separately in order to obtain more accurate results, and their summarized weight will be used as the weight for the original object.

The BVPM-based method estimates the object weights with approximated coverage of the visible portions in the scene. Although, an occlusion map [8] of the object can provide the most accurate coverage of the visible portions, as also shown in Figure 2(b), generating the occlusion map requires processing every geometry primitive, which significantly increases the computation complexity. In contrast, the BVPM-method has light weight computation, and is therefore preferable for timely streaming applications.

4.2. LOD Selection

The object weighting process differentiates objects into three categories: (i) objects rejected by the view-frustum test (hence $\omega = 0$), (ii) objects in the view frustum but assigned $\omega = 0$, and (iii) objects with $0 < \omega \leq 1$. Transmission strategies are different for the three categories. In particular, objects in the first category will not be transmitted while for objects in the second category only the base representations will be sent. For objects in the third category, proper resolutions will be selected with regard to the bit-rate constraint.

Without loss of generality, we assume a total set of N objects, $\{\mathcal{O}_k\}_{k=1...N}$, that have non-zero weights, denoted by $\{\omega_k\}$. Each object $\mathcal{O}_k$ has a number of L_k LODs and their respective distortion-rate performance are given by $\{\mathcal{D}_{kj}, \mathcal{R}_{kj}\}_{k=1...N, j=1...L_k}$, where $\mathcal{R}_{kj}$ is the number of bits needed for decoding LOD M_k^j and $\mathcal{D}_{kj}$ denotes the corresponding distortion, measured using (3).

We use an LOD vector $\boldsymbol{\pi} = \{\pi_1, ..., \pi_N\}$ to denote a set of resolutions for the N objects where a resolution π_k is selected for object $\mathcal{O}_k$. Apparently, each selection of $\boldsymbol{\pi}$ has a corresponding rate $\mathcal{R}(\boldsymbol{\pi})$ with a resulting distortion $\mathcal{D}(\boldsymbol{\pi})$. The objective of the streaming mechanism is to find the proper LOD vector that minimizes distortion while satisfying a given rate constraint $\mathcal{R}_c$, i.e.,

$$\boldsymbol{\pi}_{opt} = \underset{\mathcal{R}(\boldsymbol{\pi}) \leq \mathcal{R}_c}{\arg\min}\ \mathcal{D}(\boldsymbol{\pi}), \tag{7}$$

where $\mathcal{R}(\boldsymbol{\pi}) = \sum_{k=1}^{N} \mathcal{R}_{k,\pi_k}$, and $\mathcal{D}(\boldsymbol{\pi}) = \sum_{k=1}^{N} \omega_k \mathcal{D}_{k,\pi_k}$.

To find the solution for (7), the most straightforward method is a brute force algorithm that searches over the entire solution space. Such an exhaustive search, however, only works under situations where the solution space is with small or modest size, as its complexity increases exponentially when the number of objects in the scene increases. For this reason, we employ in this paper a steepest decent algorithm as a fast heuristic, which requires only linear computation time. Starting from the lowest resolution for each object, the steepest decent search is performed in an incremental fashion. At each step, the resolution of each object is increased to a higher LOD; the ratio of resulting distortion reduction over rate increment is calculated and multiplied by the corresponding object weight. Then, the object with the maximum ratio of distortion reduction over rate increment is selected and its resolution is increased. This procedure is repeated until the rate constraint is reached.

Object index (k)	1	2	3	4	5
Object weight (ω_k)	0.52	0.14	0.29	0.00	0.05

Fig. 3. The objects within the view frustum and the estimated object weights. The 4^{th} object is allocated a zero weight, meaning that a base resolution will be transmitted for that object.

4.3. Multi-Streaming

The selected LODs for different objects are transmitted over a number of n ($n \leq N$) streams using an SCTP-based transport layer. While reliable transmission is guaranteed for all the streams, sequenced delivery only resides within each of the n streams. Thus, reception of a particular object will not be delayed by packet losses occurred in other streams, and the decoding will be performed promptly once an LOD is received. A round-robin scheduling scheme is employed to regulate the transmission of multiple streams.

5. PERFORMANCE EVALUATION

In this section we present a preliminary evaluation on the performance of the proposed framework, which is referred to as PODS (Partially Ordered Delivery for 3D Scenes) hereafter. We compare PODS with a heuristic that transmits all the objects with approximately the same LOD and in a strictly ordered machinery.

A simple benchmark containing seven objects is used in the evaluation, five of which are accepted by the view-frustum test. Each object is encoded to generate ten enhancement batches. The full-resolution objects have 39698 polygons for the HORSE model and 56192 polygons for the DINOSAUR while their lowest-resolution counterparts have 710 and 1022 polygons, respectively. The remaining objects and their estimated weights are shown in Figure 3, where for simplicity object partitioning is not performed and a single bounding box is used for each object.

The simulation is performed using the *ns-2* network simulator [9]. We consider a topology with two SCTP endpoints connected by a bottleneck link that has a low transmission rate. Particularly, we

consider a link with transmission rate $r = 384$ kbps, propagation delay $d = 120$ msec, packet size $s = 1000$ bytes, and packet loss rate $p = 0.01$. Without loss of generality, the sending buffer is assumed to be sufficiently large while the receiver buffer has a limited size of χ bytes. Unless otherwise noted, we use $\chi = 10$ KB in the simulation. We conduct the simulation for a long period and present the averaged results.

A comparison of the receiving quality with respect to different bit rate constraints is first presented in Figure 4(a). One can see that PODS significantly outperforms the comparing heuristic through the presented range of bit-rates. For lower bit rates ($\mathcal{R}_c \leq 100$ KB), PODs improves the receiving quality over the heuristic by 3.5–5 dB. As the bit rate increases, PODS quickly approaches the quality close to the full resolution ($PSNR > 45$ dB), and achieves a gain larger than 10 dB compared to the heuristic.

Figure 4(b) evaluates the performance of PODS from another perspective, where we plot for the same quality the number of polygons that are processed by the client. Such an evaluation is important as the larger number of polygons processed, the more computation power will be consumed by the rendering operations at the client. In Figure 4(b), it is shown that with the same receiving quality, using PODS drastically decreases the number of polygons that need to be processed. For example, for the quality of $PSNR \approx 38$ dB, the heuristic method transmits 92900 polygons to the client while PODS sends solely 56794 polygons, which implies a saving of 39% on the rendering process.

To further illustrate the performance of PODS, we present in Figure 4(c) the transmission delay for achieving 38 dB quality with the two approaches, with respect to different sizes of the receiving buffer. As one can expect, the saving on bit rate for PODS results in a constant reduction in the transmission delay. In addition to the constant reduction, a larger improvement is observed for PODS when the size constraint on the receiving buffer is stringent ($\chi \leq 20$ KB). This improvement is contributed by the partially ordered transport which results in a more efficient use on the receiving buffer. Overall, the performance evaluation verifies PODS a successful mechanism for the studied environment which has a low bit rate and highly limited computation resource.

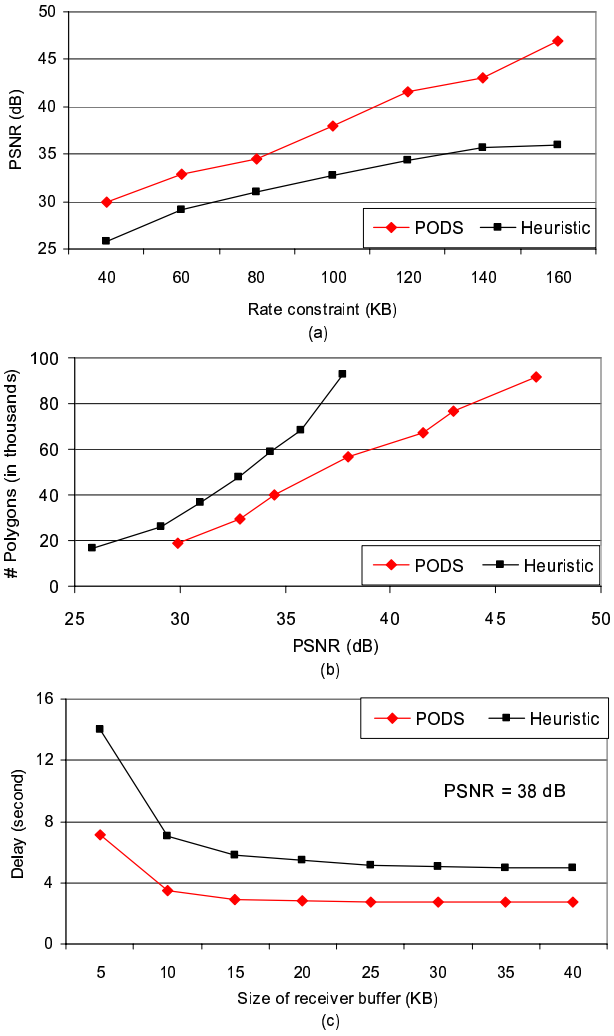


Fig. 4. Comparisons between PODS and the heuristic: (a) quality versus bit rate, (b) processing cost versus quality, and (c) transmission delay versus quality.

6. CONCLUSIONS

A streaming mechanism (PODS) has been proposed for 3D graphical scenes. Taking in account both interactions and independencies of the 3D objects, the proposed mechanism performs bit allocation on the server side by pre-processing the scene database, and transmits the selected resolutions of the objects in a partially ordered fashion. As the result, PODS greatly improves the transmission efficiency and reduces the processing requirement on the client, which makes it preferable for networked graphics applications.

7. REFERENCES

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

[2] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus, "Progressive forest split compression," in *Proceedings of ACM SIGGRAPH*, 1998, pp. 123–132.

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

[4] Z. Chen, B. Bodenheimer, and F. J. Barnes, "Robust transmission of 3D geometry over lossy networks," in *Proceedings of Web3D*, 2003, pp. 161–172.

[5] G. Al-Regib and Y. Altunbasak, "3TP: An application-layer protocol for streaming 3-D graphics," *IEEE Transactions on Multimedia*, 2005, to appear.

[6] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, *Stream control transmission protocol*, Request for Comments (RFC) 2960, The Internet Society, October 2000.

[7] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," in *Proc. Eurographics*, June 1998, vol. 17(2), pp. 167–174.

[8] H. Zhang, "Effective occlusion culling for the interactive display of arbitrary models," Ph.D. dissertation, Department of Computer Science, UNC-Chapel Hill, 1998.

[9] UCB/LBNL/VINT, *network simulator ns (version 2)*, available on http://www.isi.edu/nsnam/ns/.