Loss Tolerance Scheme for 3D Progressive Meshes Streaming over Networks

Hui Li[1], Ziying Tang[2], Xiaohu Guo[2], B.Prabhakaran[2]
[1]Ask.com, USA Email: huili.michael@gmail.com
[2]Department of Computer Science, University of Texas at Dallas, USA
Emails: {zxt061000, xguo, praba}@utdallas.edu

***Abstract* – Nowadays, the Internet provides a convenient medium for sharing complex 3D models online. However, transmitting 3D progressive meshes over networks may encounter the problem of packets loss that can lead to connectivity inconsistency and distortion of the reconstructed meshes. In this paper, we combine reliable and unreliable channels to reduce both time delay and mesh distortion, and we propose an error-concealment scheme for tolerating packet loss when the meshes are transmitted over unreliable network channels. When the loss of connectivity data occurs, the decoder can predict the geometry data and mesh connectivity information, and construct an approximation of the original mesh. Therefore, the proposed error-concealment scheme can significantly reduce the data size required to be transmitted over reliable channels. The results show that both the computational cost of our error-concealment scheme and the distortion introduced by our scheme are small.**

1. INTRODUCTION

The advent and development of the Internet have transformed computer networks to a convenient medium for sharing different kinds of information, including complex 3D models. Although there are many shape representations, we only consider 3D models represented as triangle meshes in this paper. Before the progressive compression techniques like [1, 2] are proposed, long-time delay is a common problem for transmitting detailed 3D meshes with large data sizes as the single-resolution techniques [3] need to download complete meshes before rendering starts.

Progressive compression techniques solve this problem by sending a coarse mesh followed by a sequence of refinements that are a series of vertex splits operations as mentioned in [4]. Most of the existing progressive compression techniques use TCP, and some use UDP but with ineffective data loss handling. Transmitting using TCP is reliable but leads longer delay due to the packet retransmissions and congestion control mechanisms; hence, it is not suitable for real-time applications like 3D games. On the other hand, using UDP for streaming is faster but the distortion due to the packet loss cannot be avoided. Although hybrid transmission techniques have been proposed, there is no efficient solution to handle packet loss and balance high band load and time delay.

In this paper, we introduce a recovering approach which focuses on dealing with irreversible packet loss during streaming on hybrid transmissions to take the advantages of both TCP and UDP. We send split bits, triangulation bits and geometric data, which are all included in the refinement information, differently. The base mesh and the split bits are encoded and transmitted over reliable channels, while the triangulation and the geometric data are delivered over unreliable channels. Each packet transmitted over the unreliable channel contains a vertex index field indicating the index of the first vertex in this packet. The decoder uses it to detect the vertices when triangulation and geometry data are lost. When packet loss occurs on the unreliable channel, we propose a prediction method that predicates the geometry data and connectivity information. We have included two methods, the reliability bit method and the partition method, to limit the prediction errors propagation, and restrict the errors in a particular partition only.

The proposed approach is robust against the connectivity data loss during the streaming, while effectively reducing the transmission delay. Our decoding with triangulation prediction is able to estimate the geometry data and connectivity information of new vertices when the triangulation bits and geometric data are lost. Our results show that the distortion of decoded mesh is small and limited.

2. RELATED WORKS

Our work is closely related to the progressive compression techniques. Progressive Meshes (PM) [1] uses the simplified base mesh and the refinement technique of vertex splitting [4]. However, it is limited on the compression ratio as the vertices are removed one at a time. Compressed Progressive Meshes (CPM) [2], which applies Butterfly subdivision scheme to predict the displacement of the new vertex, groups vertex splits into batches.

Al-Regib and Altunbasak suggest a hybrid 3TP protocol [5] to control the errors during transmission. Their approach reduces the distortion and delay by transmitting only the selected 3D parts by TCP and the rest by UDP. Other methods [6, 7] use redundant data to tolerate packet loss during transmission over unreliable channels. Redundant bits are decided in terms of the importance of the transmitted data to the receivers. Obviously, the use of redundant data leads to more workload for data transmission. In addition, because the data loss during transmission is unpredictable, it is hard to guarantee the correctness of decoding.

MPEG-4 [8] divides the 3D Mesh into disjoint partitions to localize the error propagation, which is a similar idea used in our method.

In order to know the reliability of transmission channel, some works [9, 10] propose methods to detect network conditions first and bandwidth available before streaming. In our algorithm, we focus on developing a loss data recovering technique which is applicable to different compression methods and lossy networks conditions.

3. METHODOLOGY

Our goal is to reduce distortion as much as possible while at the same time reducing transmission delay. In this loss tolerance

scheme, we send encoded data using the hybrid protocols, and we propose a mesh prediction method to handle packet loss in decompression.

3.1. Scheme Architecture

The architecture of our loss tolerance scheme is depicted in Figure 1. Similar to most progressive compression approaches, it consists of two parts: encoder and decoder. The details of each part will be explained as follows.

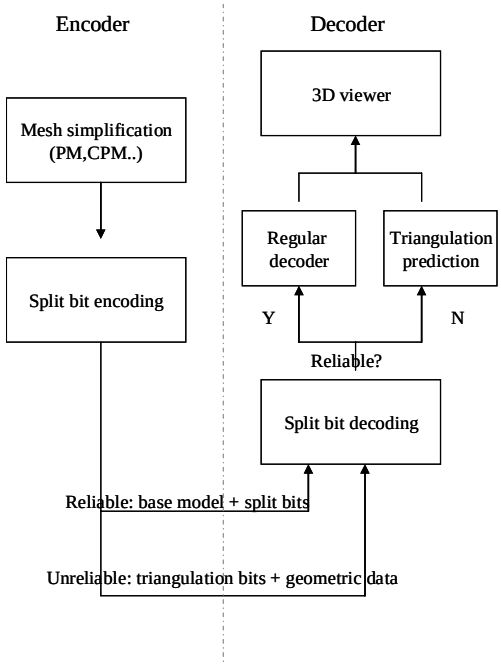

Figure 1. Architecture of the proposed scheme.

3.2. Compression

In the encoding process, a 3D mesh is simplified by a compression method similar to [2] to get a base mesh and a series of refinements. In order to guarantee correct vertex/edge references, the encoder encodes the split bits in "decoder indexing" order.

The refinements include split bits, triangulation bits and geometry data. Split bits indicate whether vertices will split during decoding or not. Triangulation bits provide the connectivity information between the new vertices and their neighbors. And geometric data includes information that represents the 3D coordinates of vertices. In our approach, while the base mesh and the split bits are transmitted over the reliable channel, the triangulation bits and the geometric data are delivered over the unreliable channel.

Split bits have to work together with triangulation bits when vertex-splitting operations are processed in decompression. Therefore, whenever packet loss occurs on an unreliable channel, we may face connectivity inconsistencies. The loss of the geometric data will affect the quality of the decoded mesh and introduce distortion.

3.3. Decompression

At the decoder, the crude model is refined progressively through a series of vertex splits, which are the inverse operations of the edge collapse described in [4]. See Figure 2 for an example.

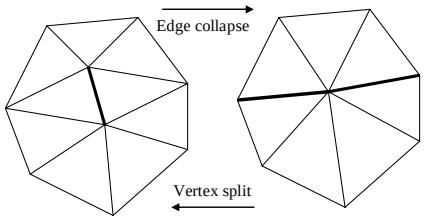

Figure 2. Edge collapse and vertex split for triangle mesh simplification and reconstruction.

After decoding split bits, the regular decoding can be applied if the triangulation bits are correctly received at the end user. Otherwise, we predict the triangulation. The triangulation prediction consists of the following steps: predict the coordinates of the new vertex iteratively by its neighbor vertices; find the connectivity of new vertex with the neighbors; mark all new vertices as unreliable for further refinements.

3.3.1 Iterative Prediction of New Vertex Coordinates

We employ a point-based approach to estimate the coordinates of the new vertex. The advantages of our method are: 1) By calculating the coordinates of the new vertex iteratively, we can obtain the "best-effort" guess based on the neighbor vertices. 2) Since the vertices closer to the new vertex play a more important role in evaluating the coordinates of the new vertex, we propose a weight assignment operator that gives higher weights to the vertices closer to the new vertex. The summation of the weights remains unity.

From the split bit, we can get the split vertex and its neighbors. Initially, we set the coordinates of the new vertex the same as the split vertex. After that, we start the iteration process:

- Find the least square best fitting plane of the split vertex, the new vertex and the neighbors; project those corresponding vertices onto this plane.
- In the best fitting plane, find the nearest neighbors of the new vertex. We start with the split vertex and find the next nearest neighbor clockwise.
- For each of the nearest neighbor, assign a weight using the equation (1). Our method satisfies: 1) The summation of the weights is unity and 2) The neighbors closer to the new vertex have higher weights.

$$w(i) = \frac{1 - w'(i)}{n} \tag{1}$$

where $w'(i) = \frac{d_i}{D}$ and $D = \sum_{i=0}^{n} d_i$

Note d_i is the distance between the neighbor vertex *i* and the new vertex, and $\sum_{i=0}^{n} w'_i = 1$

- Compute the weighted center of those nearest neighbors after assigning different weights, and assign it to the new vertex. The equation (2) explains how to get the weighted center.

$$V = \sum_{i=0}^{n} w(i) V_i \tag{2}$$

- Iteratively repeat the above processes until get a threshold.

At the encoder, there are two widely used metrics for selecting the edge for collapsing: shortest edge and minimum geometry change. For both of the metrics, the neighbors that are closer to the removed vertex play a much more important role in evaluating the coordinates of the new vertex. Therefore, by assigning larger weights to these neighbors during the iterations, we can obtain the "best-effort" guess on the new vertex coordinates.

3.3.2. New Vertex Connectivity Construction

After guessing the coordinates of the new vertices, we need to predict how to construct new triangles among new vertices, split vertices and their neighbors.

For each neighbor vertex, we need to decide if it should connect with the split vertex, or the new vertex, or both the split and the new vertices, after splitting operation. We first order the neighbors clockwise, then for each neighbor, we calculate the distance to the split vertex and to the new vertex separately. If the neighbor is closer to the split vertex, we mark it "old". Otherwise, we mark it "new". The marking results have three possibilities:

- Two connected neighbor vertices are marked "old". They form a triangle with the split vertex after the split.
- Two connected neighbor vertices are marked "new". They form a triangle with the new vertex after the split.
- Two connected neighbor vertices, one is marked "old" and the other is marked as "new".

In case three, the two neighbor vertices, the split vertex and the new vertex form a quadrilateral which is a hole on the refined mesh. To get ride of the hole, we need to determine which one is the vertex on the cut-edge. To keep degrees between the new vertex and the split vertex close, if the current degree of the new vertex is less than that of the split vertex, we change both vertices to "new"; otherwise, we change them to "old". The vertex whose mark is changed is called "tip vertex" which is the vertex on the cut-edge. For each tip vertex found, we insert a new triangle consisting of the split vertex, the new vertex and the tip vertex into the mesh. An example of our new vertex connectivity construction is described in Figure 3. *s* is the split vertex, and *nv* is the new vertex. *o* vertices are those neighbors marked "old" and *n* vertices represent neighbors marked "new". *t* vertices are tip vertices mentioned in case three.

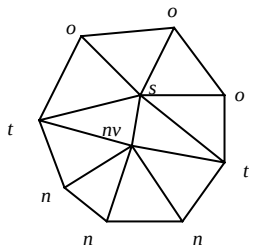

Figure 3. New vertex connectivity reconstruction.

3.3.3. Control Error Propagation

In the last step of our prediction method, we mark those predicted new vertices as unreliable. The prediction on geometric data and connectivity is an approximation of the original mesh. It becomes necessary to control the propagation of errors which are brought by prediction. Since it is very difficult to ensure that the prediction error will not be propagated to other regions of the mesh, we have included two simple methods to control the error propagation.

- Tracking vertices with predicted coordinates so that the subsequent triangulation information is not used.
- Partitioning the 3D mesh before encoding to localize the prediction errors within the partition.

In order to prevent the decoder from using the transmitted triangulation bits on the predicted vertices, we use one bit (called "reliability bit") per vertex to indicate whether this vertex is predicted or not. After each prediction, we mark all the new vertices as unreliable and any subsequent vertex split having incidence on the unreliable vertices do not use the transmitted triangulation bits. Instead, our scheme is applied. Doing so improves the robustness of the decoder since the loss-sensitive connectivity data is only used for the part of the mesh that is consistent with the encoder.

Besides using the reliability bit, a mesh partition method is also applied to localize the propagation of prediction errors. Before streaming, the original 3D mesh is divided into some disjoint partitions. Each partition is simplified individually. Note that the vertices on the partition boundary remain in the base mesh and are not simplified during the encoding. In this way, we guarantee that both encoder and decoder have the same view of the partitions and partition boundaries. It seems that a large number of partitions can help localize the prediction error. However, a large number of partitions may reduce the compression ratio since the number of vertices that can be simplified becomes smaller. Therefore, there is a tradeoff between the compression ratio and robustness of the error-concealment scheme. For the details of the partition method, please refer to [10].

4. RESULTS

We assume a network with random packet loss ratio *l*, which is between 0.04 and 0.12 based on the real networks. Each refinement reduces 5% triangles from the current mesh. We choose to implement the proposed error-concealment mechanism based on CPM [2], although our error-concealment scheme is generically applicable to different progressive representations. To explain our results, we use the Stanford "Bunny" model and the Cyberware "Horse" models with size and vertex number shown in Table 1.

	Bunny	**Horse**
Vertices	34834	48485
Size(MB)	3.25	4.74

Table 1. Uncompressed model information.

In our experiments, we record the accuracy and the computational cost (overhead) of our prediction method. The decoded mesh quality is represented by the distortion values. The *Hausdorff Distance* measured with Metro [11] is used here to represent the distortion between the decoded meshes with and without loss.

Our testing results in Table 2 show that the proposed triangulation prediction has a much higher accuracy than "random pick", which is *O(1/m)*, where *m* is the valence of the split vertex. There is no much difference as loss ratio changes because the geometry prediction depends on the local mesh geometry characteristics.

l	Bunny	Horse
0.04	0.59	0.57
0.08	0.59	0.56
0.12	0.59	0.56

Table 2. The accuracy of the proposed scheme over Bunny and Horse models with different loss ratios *l*.

Mesh	*l*	Overhead(ms)
Bunny	0.04	2.5
	0.08	3.5
	0.12	4.05
Horse	0.04	2.78
	0.08	3.60
	0.12	3.89

Table 3. The decoding time (ms) of the proposed scheme over Bunny and Horse models with different loss ratios *l* on Pentium IV machines.

Table 3 shows the computational cost of two models. As the loss ratio increases, the decoding time increases as well. The overhead can be ignored compared with the RTT(s) saved on reliable channel transmission on Pentium IV machines. In addition, this overhead can be significantly reduced if the hardware is more powerful. Distortions of the two models under different loss ratios are listed in Table 4 and Table 5 separately; besides, we compare decoded model without loss and with loss and prediction. The distortion is denoted as the percentage of whole model size. Note that the distortion is small indicating that the distortion is dominated by the simplification error. Therefore, the prediction error is localized and not propagated to other parts of the mesh.

Refinement	*l*=0.04	*l*=0.08	*l*=0.12
2	0.00333	0.00333	0.00333
4	0.003442	0.003442	0.003442
6	0.004758	0.004758	0.004758
8	0.004758	0.004758	0.004758

Table 4. Distortion (in percentage) of the Bunny over different loss ratios *l*.

Refinement	*l*=0.04	*l*=0.08	*l*=0.12
10	0.003136	0.003136	0.003136
20	0.004767	0.004767	0.004767
30	0.004767	0.004767	0.004767

Table 5. Distortion (in percentage) of the Horse over different loss ratios *l*.

5. CONCLUSION

In this paper, we propose a loss tolerance scheme that deals with packet loss on the unreliable channel. We transmit 3D triangle meshes under hybrid protocols by sending the base model and the split bits over reliable channels, and sending triangular bits and the geometry data over unreliable channels. Whenever the triangulation bits are lost, we apply the triangulation prediction method to estimate the coordinates of the new vertices and re-construct their connectivity with their neighbors.

Compared with [1, 2], our scheme is capable of dealing with the loss of connectivity data, and the transmission time is reduced due to the usage of hybrid protocol. Therefore, it is suitable for real-time 3D streaming applications like online games. This scheme is a generic solution to different compression methods. Moreover, as shown in the result, the accuracy is higher than that by random pick, and the computational cost for predicting new vertices is small and can be ignored. The distortion is caused mainly by the simplification operations, not by the triangulation prediction.

The connectivity prediction of decompression session does not always generate correct results in some degenerate cases. In the future, we will investigate more connectivity prediction techniques. The connectivity data is important for vertex splitting operations, so its streaming over reliable channels and the statistical performance analysis by balancing the higher band load and time delay is a research direction in our future work. In addition, our algorithm assumes that reliable channel is available which is maybe not suitable for general cases. Therefore, using only unreliable channels will be considered.

6. REFERENCES

[1] H. Hoppe, "Progressive meshes," in *Proceedings of SIGGRAPH'96*, 1996.

[2] R.Pajarola and J.Rossignac, "Compressed progressive meshes," in *IEEE Trans. Visual. Comput. Graphics*, vol. 6, pp. 79–93, 2000.

[3] M.Deering, "Geometry compression," in *Proceedings of SIGGRAPH'95*, 1995.

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proceedings of. SIGGRAPH '93*, pp. 19-26, 1993.

[5] G.Al-Regib and Y. Altunbasak, "3TP: An application-layer protocol for streaming 3D graphics," in *Proceedings of ICME*, 2003.

[6] Z. Yan, S. Kumar, J. Li, and C. Kuo, "Robust encoding of 3D mesh using data partitioning," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, vol. 4, 1999, pp. 495–498.

[7] G. Al-Regib and Y. Altunbasak, "An unequal error protection method for packet loss resilient 3D mesh transmission," in *Proceedings of INFOCOM*, 2002.

[8] E. Jang, S. Kim, M.Song, M.Han, S. Jung, and Y. Seo, "Results of CE M5 error resilient 3d mesh coding," ISO/IEC JTC 1/SC 29/WG 11, 1998.

[9] AL-REGIB, G., ALTUNBASAK, Y., AND ROSSIGNAC, J. 2005. Error-resilient transmission of 3D models. *ACM Trans. Graph. 24*, 182–208.

[10] H. Li, "Streaming 3D progressive meshes over lossy Networks," PhD Thesis, Fall 2007, University of Texas at Dallas.

[11] P.Cignoni, C.Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," in *Computer Graphics Forum*, vol. 17, 1998.