View-Dependent Coding of 3D Mesh Sequences

Semih Çelik[*]
Istanbul Technical University

Uluğ Bayazıt[†]
Istanbul Technical University

ABSTRACT

Visibility computations are commonly used in computer graphics applications. This paper presents a new view-dependent compression technique for 3D mesh sequences. The approach consists of geometry coding of visible parts and region descriptions of changes in visible regions. The proposed view-dependent compression method yields significant improvement in compression performance over compression method without visibility awareness with gains up to 47%.

Index Terms: Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

1 INTRODUCTION

Mesh sequences are popularly used for visualization of moving synthetic objects in computer simulation, film and game industries. View-dependency is used in many application areas for 3D meshes such as, visualization and lightening. The purpose of this paper is adapting view-dependency to compression of 3D mesh sequences. View-dependency has not been applied to 3D mesh sequences in the literature. For static 3D meshes, two types of visibility based compression methods have been applied.

Removal of invisible vertices has been applied to the compression of static meshes in some studies [1, 2, 3]. Additionally, some studies allocate low bits to invisible parts of static meshes [4, 5, 6].

In our study, we assume that the viewpoint is known at the encoder. Another assumption is that the connectivity of the mesh is constant over time.

The proposed method is based on removing invisible vertices in each frame of a 3D mesh sequence. The decoder must have all visible parts in each frame. There are three considerations of bit rate change when visibility detection is incorporated into the proposed system.

1- Bit reduction with removal of invisible vertices.

2- Required bits for region descriptions, which become visible or invisible in current frame

3- Some vertices become visible in the current frame. These vertices have no temporal reference.

The first two factors directly affect the bit rate, while the third factor increases bit rate as much as the difference in efficiency between temporal and spatial compression methods.

2 PROPOSED METHOD

Our proposed system is composed of visibility detection, region description and geometry coding. Edgebreaker ([7]) is applied to first frame of mesh sequences to encode the constant connectivity.

2.1 Visibility Detection and Region Definitions

Ray-triangle intersection [8] is applied to detect which vertices/faces are visible in the current frame. Each frame might be divided into two regions as visible ($R^f{}_1$) or invisible ($R^f{}_0$) and four regions as in Figure 1 based on changes of visibility (from f-1 to f):

$R^f{}_{(0,0)}$: invisible in both previous and current frame
$R^f{}_{(0,1)}$: visible parts, which are invisible previously
$R^f{}_{(1,0)}$: invisible parts, which are visible previously
$R^f{}_{(1,1)}$: visible in both previous and current frame

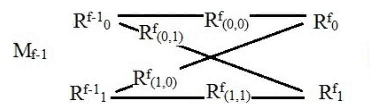

Figure 1: Region definitions for transition from frame f-1 to f.

In coding phase of any frame f, the encoder and the decoder has the information of $R^{f-1}{}_0$ and $R^{f-1}{}_1$. Thus, describing one of $R^f{}_{(0,0)}$ and $R^f{}_{(0,1)}$ also describes the other region. Similarly, describing one of $R^f{}_{(1,0)}$ and $R^f{}_{(1,1)}$ is sufficient to describe the other region.

Regions, which are visible or invisible in both frames f-1 and f, are greatly larger than regions, which become visible or invisible from frame f-1 to frame f. It is beneficial to represent visibility with descriptions of $R^f{}_{(0,1)}$ and $R^f{}_{(1,0)}$.

2.2 Region Description

A modified version of the simple and efficient Edgebreaker algorithm [7, 9, 10] is developed to represent regions. Edgebreaker traverses a mesh by region growing. It encodes one of five symbols {center (C), left (L), right (R), split (S) and end (E)} for each face included into the region. The symbols indicate the direction of traversal for following faces with respect to a reference edge (gate). Edgebreaker algorithm requires less than 2n (guaranteed higher bound) bits and 1.7n bits on the average to encode connectivity of a mesh with n triangles.

In order to describe any region R, we developed a modified version of Edgebreaker. The modified version employs a reduced alphabet of symbols split (S), right (R), left (L) and end (E). In our coding scheme, symbol R replaces symbol C in Edgebreaker. Starting with a gate g and a triangle X, coding decisions in possible cases are shown in Figure 2.

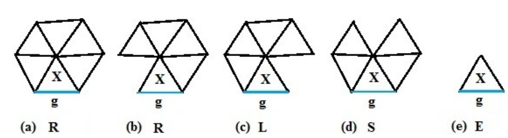

Figure 2: Decision in different cases of boundary.

In Figure 2(a), symbol R replaces symbol C of the original Edgebreaker. In Figure 2(b), 2(c), 2(d) and 2(e), the modified version uses the same symbols of the original Edgebreaker.

Any face that lies in $R^{f-1}{}_0$ cannot be included into $R^f{}_{(1,0)}$. Likewise, faces in $R^{f-1}{}_1$ are excluded from $R^f{}_{(0,1)}$.

Traversal algorithms visit all faces of a region once. In the traversal of $R^f_{(1,0)}$, the faces of $R^{f-1}{}_0$ and the already visited regions of

[*]e-mail: semcelik@itu.edu.tr
[†]e-mail:ulugbayazit@itu.edu.tr

$R^f_{(1,0)}$ restrict the symbol set at each coded face of $R^f_{(1,0)}$. Similarly, in the traversal of $R^f_{(0,1)}$, the faces of $R^{f-1}{}_1$ and the already visited regions of $R^f_{(0,1)}$ as well as $R^f_{(1,0)}$ restrict the symbol set at each coded face of $R^f_{(0,1)}$.

Based on borders around the gate known at the decoder, there are five different cases that are shown by Figure 3.

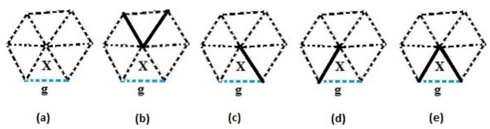

Figure 3: Different cases of borders known at decoder.

In Figure 3, bold edges represent the borders known at the decoder. Blue edges (g) are gates, and other dashed lines are unknown at the decoder or irrelevant to the decision.

For the cases (a) and (b), possible symbols are S, L, R and E, for the case (c), possible symbols are L and E, for the case (d), possible symbols are R and E and for the case (e), possible symbol is E. Decision for E-model is deterministically known at the decoder and no encoding required.

For our specific purpose of descibing a region R, the entropy of the the proposed modification to the Edgebreaker is 1.62, which is lower than Edgebreaker with 4 symbols (1.68) and absolutely lower than the original Edgebreaker (with 5 symbols).

2.3 Geometry Coding

In our coding scheme, we encode two kinds of vertices, which are inside of $R^f{}_{(0,1)}$ or $R^f{}_{(1,1)}$. Both regions are possibly composed of smaller, unconnected regions. Vertices inside $R^f{}_{(0,1)}$ only have local spatial reference. Vertices inside $R^f{}_{(1,1)}$ have both spatial reference and temporal reference.

For spatial prediction, parallelogram prediction is used. For temporal prediction, motion vector averaging prediction ([11]) is used.

2.4 Entropy Coding

We quantize residuals by a dead zone quantizer with quantization bin width 2Δ around zero and Δ everywhere else. The value of Δ determines the amount of data loss.

Quantized residuals are encoded with an adaptive arithmetic coder as discussed in [12]. Probabilities in all previous frames are used to compose the model for the coding of current frame.

3 Experiments

As a benchmark, we implemented a view-independent method, which has same quantization and prediction methods with the proposed view-dependent compression system.

Rate-distortion results of experiments are given in Figure 4. We used the distortion metric defined in [13] as KG-error. Bit rates are given in units of bit per vertex per frame (bpvf). In the experiments, chicken crossing mesh sequence is used. The chicken crossing sequence has 3030 vertices, 5664 faces and 400 frames.

According to tests in three viewpoints, invisible parts have average of 1498.4 vertices, which is not encoded. A significant compression gain (25% to 47% experimentally) is achieved by using proposed view-dependent compression method.

4 Conclusion

In this study, we presented a predictive compression of 3D mesh sequences with encoding only visible vertices. Additionally, we developed our region description algorithm based on Edgebreaker to represent changes in visibility. We showed that view-dependent compression of mesh sequences exploits visibility of meshes to reduce bit rate significantly (up to 47% experimentally).

In the future, our system could be modified with prediction of next viewpoints and extending the visible area to cover future visibility of 3D mesh sequences.

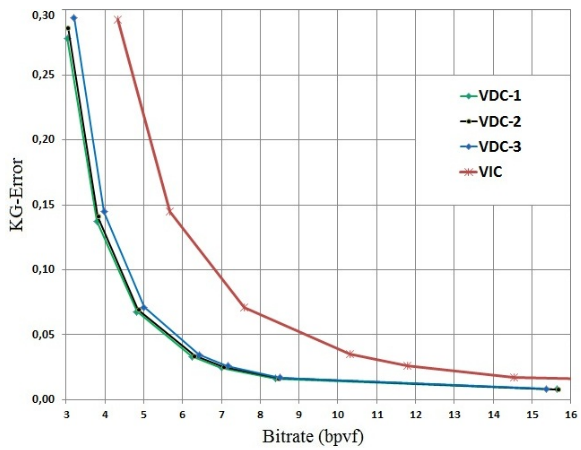

Figure 4: Compression of chicken sequence.VIC represents the view-independent compression. VDC-1, VDC-2 and VDC-3 represent the proposed view-dependent compression (three viewpoints).

References

[1] Sheng Yang, Chang-Su Kim, and C.-C. Jay Kuo. View-dependent progressive mesh coding for graphic streaming. In *Multimedia Systems and Applications IV*, volume 154, August 2001.

[2] Wei Guan, Jianfei Cai, Jianmin Zheng, and Chang Wen Chen. Segmentation-based view-dependent 3-d graphics model transmission. *IEEE Transactions on Multimedia*, 10(5):724–734, August 2008.

[3] F Payan, M Annonini, and F Meriaux. View-dependent geometry coding of 3d scenes. In *IEEE International Conference on Image Processing (ICIP)*, pages 729–732, November 2009.

[4] Pierre Alliez, Nathalie Laurent, Henri Sanson, and Francis Schmitt. Efficient view-dependent refinement of 3d meshes using sqrt3-subdivision. *The Visual Computer*, 19(4):205–221, 2003.

[5] Sheng Yang, Chang-Su Kim, and C.-C. Jay Kuo. View-dependent progressive mesh coding based on partitioning. In *Visual Communications and Image Processing*, volume 268, January 2002.

[6] Wei Guan, Jianfei Cai, Juyong Zhang, and Jianmin Zheng. Progressive coding and illumination and view dependent transmission of 3d meshes using r-d optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(4):575–586, apr 2010.

[7] Jarek Rossignac. Edgebreaker connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[8] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersections. *Journal of Graphics Tools*, 2(1):21–28, 1997.

[9] Jarek Rossignac, Alla Safonova, and Andrzej Szymczak. 3d compression made simple: Edgebreaker on a corner-table. In *Shape Modeling International Conference*, pages 278–283, 2001.

[10] Jarek Rossignac, Helio Lopes, Alla Safanova, Geovan Tavares, and Andrzej Szymczak. Edgebreaker: A simple compression for surfaces with handles. In *In Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 289–296. ACM Press, 2002.

[11] N Stefanoski and J. Ostermann. Connectivity-guided predictive compression of dynamic 3d meshes. In *IEEE International Conference on Image Processing*, pages 2973–2976, 2006.

[12] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. In *Communications of the ACM*, volume 30, pages 520–540, 1987.

[13] Zachi Karni and Craig Gotsman. Compression of soft-body animation sequences. *Computers and Graphics*, 28(1):25–34, 2004.