CONTEXT-BASED OCTREE CODING FOR POINT-CLOUD VIDEO

Diogo C. Garcia and Ricardo L. de Queiroz

Universidade de Brasilia, Brasil

ABSTRACT

3D and free-viewpoint video has been moving towards solid-scene representation using meshes and point clouds. Point-cloud processing requires much less computation and the points in the cloud are minimally represented by their geometry (3D position) and color. A common point-cloud geometry compression method is the octree representation, which acts on individual frames. We present a lossless inter-frame compression method for pointcloud geometry, by reordering each octree based on previous frames prior to entropy coding. Results show that compared to the octree representation, the proposed solution offers an average rate reduction of 30%, while entropy-coding of the octree yields an average rate reduction of 24%.

***Index Terms*—** Point-cloud compression, 3D immersive video, free-viewpoint video, octree, real-time point-cloud transmission.

1. INTRODUCTION

3D immersive communications have recently evolved, allowing for the real-time capture and transmission of moving persons and objects [1][2]. According to the space-time representation method chosen for such systems, different features are made possible. Multiview-plus-depth video is a low-level representation of 3D scenes, based on their 2D projection, that can be efficiently compressed, for example, with the multiview extension of the High Efficiency Video Coding standard, MV-HEVC [3][4]. Polygonal meshes represent 3D scenes with connected polygons, and are the most common representation for computer graphics [5]. Point clouds represent 3D scenes by indicating colors of points in a regular grid in 3D space (voxels), and are less computationally intensive than polygonal meshes [2].

Point clouds of sampled real-world objects are usually volumetrically sparse, typically represented by the points' 3D positions or geometry, along with their corresponding colors. The geometry information is commonly encoded through octree scanning [6]–[7], applied to individual frames. In video compression terminology, this is referred as intra-frame coding. Frame correlation, or inter-frame coding, can be used to improve compression ratios. In this paper, a simple lossless inter-frame compression method is presented for point-cloud geometry, where each octree is reordered based on previous frames and entropy coded.

2. GEOMETRY CODING FOR POINT CLOUDS

2.1. Octree scanning and coding

The octree is a tree data structure which represents the geometry of 3D data in a compact manner by recursively dividing 3D space into cubes of equal volume, or octants, as shown in Fig. 1(a). Each division in 3D space is regarded as a *level* in the octree. By indicating which of these octants contain points, compression is achieved, as this eliminates the need for representing the status (occupied or not) of every possible voxel in 3D space. This also allows for the progressive representation of 3D space, since stopping at a given level is equivalent to downsampling the point cloud to a correspondent factor [6].

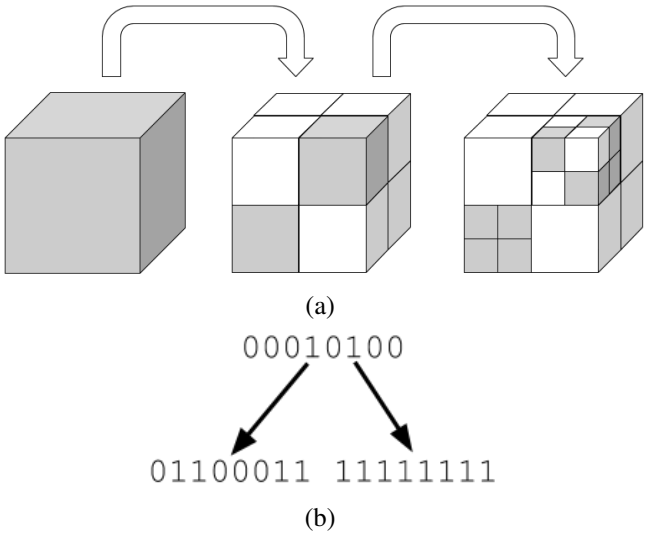

Fig. 1. Point-cloud representation using octrees: (a) three levels of the subdivision of 3D space in octants; (b) binary tree representation of (a). Dark-grey octants indicate the presence of points in further levels of the octree.

Octree encoding can be done by indicating the presence of points in a given level with a binary symbol, as indicated in

Fig. 1(b). Since eight octants divide 3D space in each octree level, these symbols can be grouped in bytes along each level.

Analysis of this byte grouping reveals that the octree can be further compressed by general entropy-coding methods, such as Huffman codes, arithmetic coding and LZW [8]. Consider, for instance, frame 149 in point-cloud sequence *Man* [9], which contains 181461 occupied voxels. The equivalent 9-level octree representation contains 70994 bytes, yielding an average rate of 3.13 bits per occupied voxel. DEFLATE encoding of this octree using GZIP [8] reduces its representation to 58266 bytes, yielding an average rate of 2.57 bits per occupied voxel.

2.2. Context-based octree inter-frame coding

In the same way that video compression achieves better compression ratios than image compression by exploiting inter-frame correlation, point-cloud video compression can benefit from similar techniques. However, point-cloud frames and 2D-video frames have a fundamental difference: the amount of occupied voxels varies along the frames in a point cloud, while 2D video always offers the same amount of pixels along its frames. This characteristic makes it difficult to apply motion estimation and compensation to point-cloud frames without some extra efforts.

Since the amount of occupied voxels varies, the length of the corresponding octree also varies. Also, nodes that were not filled in a previous frame could suddenly be occupied in a new frame, or vice-versa, making it very hard to use one octree to predict another, for example.

To illustrate these concerns, consider the first two levels of the octree representations of frames 149 and 150 for point-cloud sequence *Man* [9], shown in Fig. 2. Frame 150 has all eight octants filled in the first level, while frame 149 has one less octant, so that the second levels in frames 149 and 150 have 7 and 8 bytes, respectively. As for the octants filled in both frames, they may constitute good predictions from one frame to another. In the example, five of the seven common octants are identical in both frames.

Considering these characteristics, a prediction octree $\mathbf{O}_P$ for the current octree $\mathbf{O}_C$ is composed using a reference octree $\mathbf{O}_R$ in the following manner. We wish to build a good prediction $\mathbf{O}_P$ from $\mathbf{O}_R$, having as many bytes as $\mathbf{O}_C$. In order to do so, we copy the levels in $\mathbf{O}_R$ to $\mathbf{O}_P$ if $\mathbf{O}_R$'s parent levels are also occupied in $\mathbf{O}_C$. Parent levels present in $\mathbf{O}_C$ but not in $\mathbf{O}_R$ are filled with zeros (00000000) in $\mathbf{O}_P$. In this manner, $\mathbf{O}_P$ will contain the exact amount of bytes as $\mathbf{O}_C$, and $\mathbf{O}_R$ will be used as prediction just for the levels in common with $\mathbf{O}_C$. The null value inserted in other levels was chosen because this value only occurs in the octree representation if all 3D space is unoccupied, which is rarely the case. Since the first level has no parent level, the first level of $\mathbf{O}_P$ is equal to the first level of $\mathbf{O}_R$.

This method for creating a prediction octree $\mathbf{O}_P$ was developed by Kammerl *et. al.* [10], who then applied the exclusive disjunction operation between $\mathbf{O}_P$ and $\mathbf{O}_C$ (XOR($\mathbf{O}_P, \mathbf{O}_C$)) before entropy encoding. In this paper, $\mathbf{O}_P$ is used in another manner, which proved itself more effective in the tests performed in Section 3. Instead of entropy-encoding XOR($\mathbf{O}_P, \mathbf{O}_C$), $\mathbf{O}_P$ is sorted in ascending order, creating $\mathbf{O}_{SP}$. The sorting order found ($\mathbf{O}_P \Rightarrow \mathbf{O}_{SP}$) is applied to $\mathbf{O}_C$, yielding the octree $\mathbf{O}_{SC}$, which is entropy coded. The reason for this sorting is that the sorted version of $\mathbf{O}_P$ has long sequences of repeated symbols, which may have higher compression ratios using some compressor, and if $\mathbf{O}_P$ constitutes a good prediction of $\mathbf{O}_C$, $\mathbf{O}_{SC}$ will also have similar long sequences of repeated symbols. If the prediction is not good, $\mathbf{O}_{SC}$ becomes a scrambled version of $\mathbf{O}_C$, with similar statistics, entropy and compression ratio.

If more than one frame is to be used as reference, the reference $\mathbf{O}_R$ is composed by all unique 3D positions in these reference frames. After that, the algorithm continues as explained in the previous paragraphs. For example, for the following reference frames

$$\mathbf{V}_{R1} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 12 \\ 11 & 12 & 10 \\ 12 & 13 & 5 \end{bmatrix} \text{ and } \mathbf{V}_{R2} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 12 \\ 11 & 11 & 11 \\ 12 & 13 & 5 \end{bmatrix}, \quad (1)$$

where the columns in matrices $\mathbf{V}_{R1}$ and $\mathbf{V}_{R2}$ represent X, Y and Z positions of the points in those frames, the corresponding reference used by the algorithm is

$$\mathbf{V}_R = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 12 \\ 11 & 11 & 11 \\ 11 & 12 & 10 \\ 12 & 13 & 5 \end{bmatrix}, \quad (2)$$

where $\mathbf{V}_R$ contains the unique 3D positions of $\mathbf{V}_{R1}$ and $\mathbf{V}_{R2}$. $\mathbf{V}_R$ is then used to compose $\mathbf{O}_R$.

In order to allow for decoding, the histogram $\mathbf{H}_{SP}$ of $\mathbf{O}_{SP}$ needs to be known, allowing for the decoder to unscramble $\mathbf{O}_{SP}$ ($\mathbf{O}_{SP} \Rightarrow \mathbf{O}_P$) and $\mathbf{O}_{SC}$ ($\mathbf{O}_{SC} \Rightarrow \mathbf{O}_C$). First, the histogram $\mathbf{H}_R$ of $\mathbf{O}_R$ is calculated, and the difference $\mathbf{H}_D = \mathbf{H}_R - \mathbf{H}_{SP}$ is sent in L-bit representation, $L = \lceil \log_2(\arg\max(\mathbf{H}_D)) \rceil$. L is also conveyed to the decoder.

At the decoder, $\mathbf{O}_P$ is constructed sequentially from $\mathbf{O}_R$, $\mathbf{O}_{SC}$ and $\mathbf{H}_{SP}$, which can be calculated as $\mathbf{H}_{SP} = \mathbf{H}_R - \mathbf{H}_D$. The first level of $\mathbf{O}_P$ is equal to the first level of $\mathbf{O}_R$, and the corresponding first level in $\mathbf{O}_C$ is found based on $\mathbf{H}_{SP}$ and $\mathbf{O}_{SC}$. With knowledge of the first level, the second level of $\mathbf{O}_P$ can be replicated, and the process can be repeated for all the other levels.

Fig. 2. First two levels of the octree representation of frames 149 and 150 in point-cloud sequence *Man* [9], and octree prediction for frame 150 using frame 149 as reference. Corresponding levels in the frames' octrees which are different are indicated in bold typeface.

3. EXPERIMENTAL RESULTS

Tests for the proposed method were carried out with the first 200 frames of six publicly available point-cloud sequences: *Andrew*, *David*, *Man*, *Phil*, *Ricardo* and *Sarah* [9][11]. The average rate was calculated for six scenarios:

- **FO**: using the full octree;
- **EO**: entropy-encoding the full octree;
- **KA**: entropy-encoding Kammerl *et. al.*'s method ($\text{XOR}(\mathbf{O}_P, \mathbf{O}_C)$) [10];
- **P1**, **P2**, **P3**: applying the proposed method using 1, 2 or 3 references, respectively.

In order to make a fair comparison, the same entropy coder was employed for all methods, except **FO**, which is *deflate* (GZIP in version 1.0.7 of the Keka archiver for OSX, with the maximum-compression setting *-mx9*).

Table 3 presents the average rate in bits per occupied voxel for the six proposed scenarios, and Table 3 presents the average percentage rate gain over the full octree for the other five scenarios (**EO**, **KA**, **P1**, **P2** and **P3**). It can be seen that the proposed method using one reference outperforms all the other scenarios, gaining an average 5% over entropy-encoding the full octree and an average 8% over Kammerl *et. al.*'s method. Adding extra references in the proposed manner, however, did not improve those gains.

Table 3 shows that, inrelation to entropy-encoding the octree, the proposed method's gains fluctuate according to the sequence. For instance, there is a 9% rate gain for sequence *Andrew*, but only a 2% rate gain for sequence *David*. As the former sequence presents higher movement than the latter,

Table 1. Average rate in bits per occupied voxel. **FO** stands for the full octree, **EO** stands for the entropy-encoded octree, **KA** stands for Kammerl *et. al.*'s method, and **P3**, **P2** and **P1** stand for the proposed method using three, two and one references, respectively.

Sequence	FO	EO	KA	P3	P2	P1
Andrew	2.58	1.94	1.83	1.70	1.69	**1.69**
David	2.62	1.89	2.09	1.86	1.85	**1.83**
Man	3.16	2.51	2.48	2.35	2.33	**2.29**
Phil	2.64	2.00	2.13	1.93	1.91	**1.88**
Ricardo	2.92	2.37	2.39	2.24	2.23	**2.22**
Sarah	2.61	1.89	1.92	1.74	1.72	**1.70**
Average	2.76	2.10	2.14	1.97	1.96	**1.94**

Table 2. Average rate gain over full octree. **EO** stands for the entropy-encoded octree, **KA** stands for Kammerl *et. al.*'s method, and **P3**, **P2** and **P1** stand for the proposed method using three, two and one references, respectively.

Sequence	EO	KA	P3	P2	P1
Andrew	25%	29%	34%	34%	**34%**
David	28%	20%	29%	29%	**30%**
Man	21%	22%	26%	26%	**28%**
Phil	24%	19%	27%	28%	**29%**
Ricardo	19%	18%	23%	24%	**24%**
Sarah	28%	26%	33%	34%	**35%**
Average	24%	22%	29%	29%	**30%**

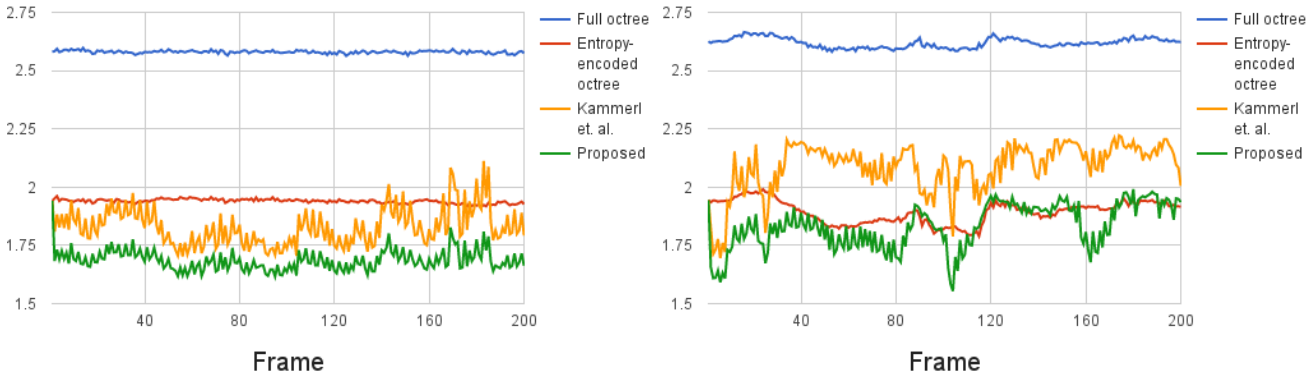

Fig. 3. Rate on a frame basis, in bits per occupied voxel, for the full octree, for entropy-encoding the full octree, for Kammerl *et. al.*'s method and for the proposed method with one reference. Sequences: (a) *Andrew*; (b) *David* [11].

these results indicate that the proposed method could benefit from motion estimation and compensation.

Figures 3(a) and (b) present the rate on a frame-by-frame basis, in bits per occupied voxel, for the full octree, for entropy-encoding the full octree, for Kammerl *et. al.*'s method and for the proposed method with one reference for sequences *Andrew* and *David*, respectively. Figure 3(a) shows the relative lack of movement in sequence *Andrew*, and Fig. 3(b) shows how the proposed algorithm's performance fluctuates with the movement in sequence *David9*.

4. CONCLUSION

In this paper, a lossless inter-frame compression method for point-cloud geometry was presented, which reorders each octree based on previous frames, prior to entropy encoding. Experiments were carried with six point-cloud sequences publicly available, with results showing that the proposed method offered an average rate reduction of 30% when compared to the octree representation, while entropy-encoding the octree and Kammerl *et. al.*'s method yielded average rate reductions of 24 and 22%, respectively, when compared to the octree representation. Frame-by-frame analysis of the results show that further improvements can be achieved if we incorporate motion estimation and compensation to the system. Other entropy-coding methods such as Huffman and arithmetic coding can also be tested.

5. REFERENCES

[1] C. Zhang, Q. Cai, P. Chou, Z. Zhang, and R. Martin-Brualla, "Viewport: A distributed, immersive teleconferencing system with infrared dot pattern," *IEEE MultiMedia*, vol. 20, no. 1, pp. 17–27, Jan. 2013.

[2] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proceedings of the 5th High-Performance Graphics Conference*. 2013, pp. 73–79, ACM.

[3] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *2007 IEEE International Conference on Image Processing*, Sept 2007, vol. 1, pp. I – 201–I – 204.

[4] G. Tech, Y. Chen, K. Mller, J. R. Ohm, A. Vetro, and Y. K. Wang, "Overview of the multiview and 3d extensions of high efficiency video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 35–49, Jan 2016.

[5] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff, and C. Rössl, "Geometric modeling based on polygonal meshes," in *ACM SIGGRAPH 2007 Courses*, New York, NY, USA, 2007, SIGGRAPH '07, ACM.

[6] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, June 1982.

[7] Y. Huang, J. Peng, C. C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440–453, March 2008.

[8] Khalid Sayood, "Introduction to data compression," The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann, Burlington, 3rd edition, 2006.

[9] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 69:1–69:13, July 2015.

[10] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *IEEE International Conference on Robotics and Automation (ICRA)*, Minnesota, USA, May 2012.

[11] C. Loop, Q. Cai, S.O. Escolano, and P.A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012*, May 2016.