INTRA-FRAME CONTEXT-BASED OCTREE CODING FOR POINT-CLOUD GEOMETRY

Diogo C. Garcia and Ricardo L. de Queiroz

Universidade de Brasilia, Brasil

ABSTRACT

3D and free-viewpoint video has been slowly adopting a solid representation, such as using meshes and point-clouds. Among other characteristics, meshes provide direct surface representation, while point-cloud processing requires less computation. 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 and can be further compressed by entropy encoding. This paper presents a lossless intra-frame compression method for point-cloud geometry, which uses the octree structure to provide better contexts for entropy coding. Results show that the proposed solution offers state-of-the-art performance, with an average rate reduction of 29% compared to the octree representation.

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

I. INTRODUCTION

In the past few years, 3D visual communications have developed tremendously, allowing for the real-time capture and transmission of events [1][2][3]. Several space-time representations are then made possible, each with different characteristics. Multiview-plus-depth video is based on multiple 2D projections of 3D scenes, which can then be efficiently compressed, for example, with the multiview extension of the High Efficiency Video Coding standard, MV-HEVC [4][5]. Polygonal meshes, on the other hand, represent 3D scenes with connected polygons [6]. Point-clouds also directly represent 3D scenes (Fig. 1), but by indicating points' colors in 3D space (voxels) [3].

Real-world objects can be rendered with volumetrically sparse point-clouds, which are more efficiently represented by the points' 3D position, or geometry, and corresponding color. The compression of geometry information can be readily obtained with octree scanning [8][9], which is done on a frame basis. In video compression terminology, this is referred as intra-frame coding, or simply intra coding, as opposed to inter-frame coding, where neighboring frame correlation is explored to improve compression ratios.

Work partially supported by CNPq under grant 308150/2014-7.

Fig. 1. Random viewpoints for frame 101 of 3D point-cloud sequence *Man* [7].

Several methods have been developed for intra- and inter-frame point-cloud compression [10][11][12][13][14][15]. Kammerl *et. al.* proposed an inter-coding system [12], while Mekuria *et. al.* developed an octree-based intra- and inter-coding system [13][14]. An open-source library was developed for the coding of point-clouds and meshes [15]. In this paper, a lossless intra-frame compression method is presented for point-cloud geometry, where each octant in the octree is entropy-coded according to its father octant.

II. GEOMETRY CODING FOR POINT-CLOUDS

II-A. Octree scanning and coding

The octree represents the geometry of point-clouds through recursive division of 3D space into fixed-size cubes, or octants (Fig. 2(a)). Each division is regarded as a *level* in the octree. Compression of sparse point-clouds is achieved through the indication of filled octants, instead of representing all possible voxels in 3D space. Furthermore, octrees allow for progressive point-cloud representation, since stopping at a given level is equivalent to downsampling the point-cloud to a correspondent factor [8].

The octree can be encoded through bitwise indication of filled octants at a given level (Fig. 2(b)). The resulting bits can be grouped into bytes, as each octree level subdivides 3D space into eight octants. The corresponding vector of bytes can then be compressed with general entropy-coding methods, such as Huffman codes, arithmetic coding and LZW [16].

For example, consider frame 149 in point-cloud sequence *Man*, which contains 181461 occupied voxels. The positions of the occupied voxels in the corresponding $512 \times 512 \times 512$ space can be represented by 27 bits per occupied voxel

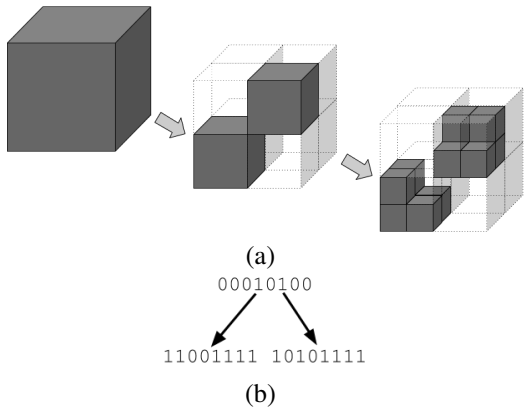


Fig. 2. Point-cloud representation using octrees: (a) three levels of the subdivision of 3D space into octants; (b) binary tree representation of (a). Dark-grey octants indicate the presence of points in further levels of the octree.

(bpov). The equivalent 9-level octree representation contains 70994 bytes, yielding an average rate of $70994 \times 8/181461 = 3.13$ bpov. LZW-based encoding of this octree reduces its representation to 58266 bytes, with an average rate of $58266 \times 8/181461 = 2.57$ bpov. This average-rate reduction from 3.13 to 2.57 bpov indicates that there is still some redundancy in the octree to be removed by entropy coders.

II-B. Context-based octree intra-frame coding

Besides using general entropy-coding methods, higher compression ratios in point-cloud-geometry coding can be achieved by exploiting contexts within the octree, for example [16]. Except for the first level, every octant in the octree has a parent octant. Octant values can be separated according to their parent's values, yielding different contexts for different entropy-coding methods, such as arithmetic coding and LZW.

In order to better illustrate the proposed method, consider the 2D symbol depicted in Fig. 3(a), as all the concepts can be easily transposed to three dimensions. 3 levels of the corresponding quadtree representation are shown, as well as its binary representation (Fig. 3(b)). Each value in the quadtree sequence can be classified either according to its parent value, yielding 15 contexts, or according to its parent bit position $\in [1, 4]$, yielding 4 contexts. (In 3D, the parent value and bit position yield 255 and 8 contexts, respectively.) Following Fig. 3(b), the quadtree sequence $\mathbf{t}$, its parent values $\mathbf{v}$, and the parents' positions $\mathbf{p}$ in the quadtree are given in hexadecimal representation as

$$\begin{aligned} \mathbf{t} &= [\text{0xF 0x3 0x2 0x4 0x8 0x1 0xC 0x9 0x8 0x8}], \\ \mathbf{v} &= [\text{0x0 0xF 0xF 0xF 0xF 0x3 0x3 0x2 0x4 0x8}], \\ \mathbf{p} &= [\text{0x0 0x1 0x2 0x3 0x4 0x3 0x4 0x3 0x2 0x1}]. \end{aligned} \tag{1}$$

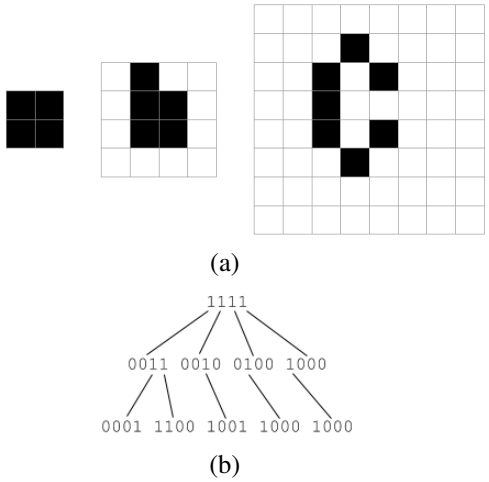


Fig. 3. A 2D symbol and its quadtree representation: (a) three levels of the subdivision of 2D space in quadrants; (b) binary tree representation of (a). Dark quadrants indicate the presence of points in further levels of the quadtree. The analysis can be extended to the 3D case and its octree representation.

For the first positions in $\mathbf{v}$ and $\mathbf{p}$, the null value was chosen in order to indicate the first level, which has no parents. All the other octants have at least one occupied position, so that the null value does not appear again.

$\mathbf{v}$ or $\mathbf{p}$ can be used as context, according to the chosen entropy-coding method. In this work, arithmetic coding and LZW are considered, but the proposed method for arithmetic coding can be easily adapted in order to support Huffman coding. Figures 4(a)-(b) illustrate the proposed methods based on arithmetic and LZW coding.

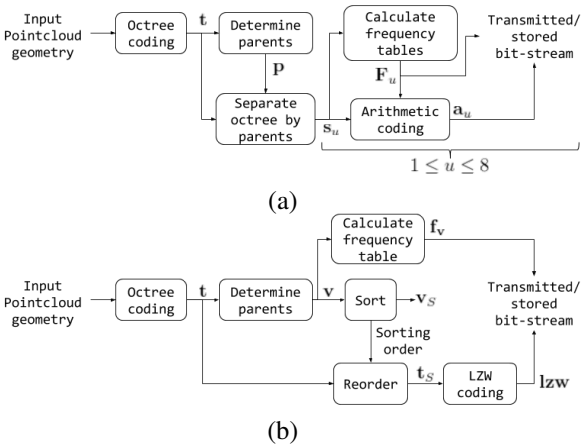


Fig. 4. Proposed methods for context-based intra coding of point-cloud geometry data, according to entropy-coding methods: (a) arithmetic coding; (b) LZW coding.

II-C. Arithmetic coding

For this entropy-coding method, decoding can only be achieved at the receiver side with knowledge of the table for the frequency of occurrence of each symbol. Although $\mathbf{v}$ offers a larger number of contexts, it also requires conveying a higher amount of side information for the frequency tables, rendering it impractical for rate reduction. $\mathbf{p}$ is then used as the context, so that 4 and 8 separate codes are generated in 2D and 3D space, respectively.

Following the example in Eq. 1, the 4×15 frequency table $\mathbf{F}$ is given as

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{2}$$

where the position (u, v) in $\mathbf{F}$ indicates the frequency of occurrence for symbol v under context u, $1 \leq u \leq 4$ and $1 \leq v \leq 15$. In 3D space, $\mathbf{F}$ is an 8×255 matrix, $1 \leq u \leq 8$ and $1 \leq v \leq 255$.

The quadtree sequence $\mathbf{t}$ is coded according to each parent position $\mathbf{p}$ and the corresponding frequency table $\mathbf{F}$. In order to simplify the implementation, $\mathbf{t}$ can be encoded according to each position separately, which avoids the need to switch between frequency tables on a constant basis. For instance, the sequences $\mathbf{s}_u$ for positions $1 \leq u \leq 4$ are $\mathbf{s}_1$ = [0x3 0x8], $\mathbf{s}_2$ = [0x2 0x8], $\mathbf{s}_3$ = [0x4 0x1 0x9] and $\mathbf{s}_4$ = [0x8 0xC], respectively. Each sequence $\mathbf{s}_u$ is then arithmetically coded using line u in $\mathbf{F}$ as its frequency table, rendering sequences $\mathbf{a}_u$. Also, the quadtree's first level $\boldsymbol{\lambda}_1$ and $\mathbf{F}$ must be conveyed to the decoder (for example, by using a variable bitdepth for each row u).

Reconstruction of $\mathbf{t}$ is performed by first decoding each sequence $\mathbf{a}_u$, recovering $\mathbf{s}_u$ back. The decoder then reconstructs each level $\boldsymbol{\lambda}_L$ based on level $\boldsymbol{\lambda}_{L-1}$. Since the first level $\boldsymbol{\lambda}_1$ was transmitted separately, the second level can then be accordingly reconstructed, and so on all the way to the last level.

In the example, $\boldsymbol{\lambda}_1$ = [0xF] = [0b1111], which indicates that positions 1, 2, 3 and 4 are filled. The decoder then reads the first position in sequences $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$ and $\mathbf{s}_4$, in that order, and the second level is reconstructed as $\boldsymbol{\lambda}_2 = [s_1(1)\, s_2(1)\, s_3(1)\, s_4(1)]$ = [0x3 0x2 0x4 0x8]. Now, $\boldsymbol{\lambda}_2$ can be interpreted exactly as $\boldsymbol{\lambda}_1$, allowing for the reconstruction of $\boldsymbol{\lambda}_3$, and so on until the last level $\boldsymbol{\lambda}_{L_{MAX}}$ is reconstructed.

II-D. LZW coding

This entropy-coding method is backward-adaptive, so that frequency tables are not needed. Since $\mathbf{v}$ offers a larger number of contexts, it is used in order to build the conditional probabilities, so that 15 or 255 separate codes are generated in 2D and 3D space, respectively.

The vector $\mathbf{v}$ is sorted in ascending lexicographical order, creating $\mathbf{v}_S$. The sorting order found is applied to $\mathbf{t}$, yielding the tree $\mathbf{t}_S$, which is then entropy-coded with the LZW algorithm, generating $\boldsymbol{\omega}$. The frequency table $\mathbf{f_v}$ of $\mathbf{v}$ is also conveyed to the decoder (for example, represented in ASCII and separated by commas). In 2D space, $\mathbf{f_v}$ is a 1×15 vector, and in 3D space, it is a 1×255 vector.

For the example in Eq. (1), $\mathbf{v}_S$, $\mathbf{t}_S$ and $\mathbf{f_v}$ are

$$\begin{aligned} \mathbf{v}_S &= [0x0\ 0x2\ 0x3\ 0x3\ 0x4\ 0x8\ 0xF\ 0xF\ 0xF\ 0xF], \\ \mathbf{t}_S &= [0xF\ 0x9\ 0x1\ 0xC\ 0x8\ 0x8\ 0x3\ 0x2\ 0x4\ 0x8], \\ \mathbf{f_v} &= [0\ 1\ 2\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 4]. \end{aligned} \tag{3}$$

Equation (3) does not accurately illustrate the advantage of applying the LZW algorithm to $\mathbf{t}_S$ instead of $\mathbf{t}$, since the former vector does not present larger sequences of repeated symbols than the latter. In order to show the potential advantages of the proposed algorithm, we calculated the first-order difference $D()$ of vectors $\mathbf{t}$ and $\mathbf{t}_S$ for frames 145 to 149 of sequence *Man2*, such that the counts of the occurrence of zeros $C_0()$ in $D(\mathbf{t})$ and $D(\mathbf{t}_S)$ for each frame are

$$\begin{aligned} C_0(D(\mathbf{t})) &= [5565\ 5175\ 4857\ 4833\ 4798], \\ C_0(D(\mathbf{t}_s)) &= [6349\ 5966\ 5777\ 5663\ 5652]. \end{aligned} \tag{4}$$

Higher values in $C_0(D(\mathbf{t}_s))$ mean that $\mathbf{t}_s$ contains more repeated values than $\mathbf{t}$. Even though the first-order difference within vectors is not the only measure of rate-reduction effectiveness for the LZW algorithm, Eq. (4) hints at potential gains.

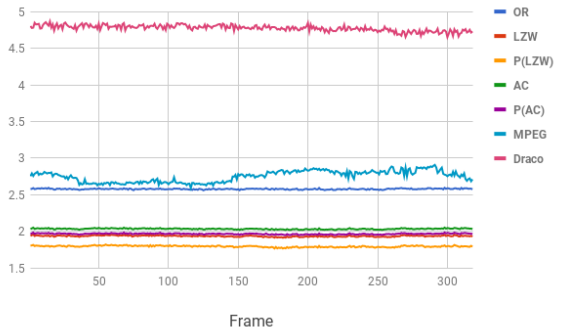
Reconstruction of $\mathbf{t}$ is performed by first decoding $\boldsymbol{\omega}$, which recovers $\mathbf{t}_S$. As in the arithmetic-coded version, the decoder reconstructs level $\boldsymbol{\lambda}_L$ based on level $\boldsymbol{\lambda}_{L-1}$, starting with the first level $\boldsymbol{\lambda}_1$, which was separately transmitted.

In the example, $\boldsymbol{\lambda}_1 = t_S(1)$ = [0xF]. $\mathbf{f_v}$ indicates that $t_S(2)$'s parent has value 0x2, $t_S(3)$ and $t_S(4)$'s parents have value 0x3, $t_S(5)$'s parent has value 0x4, $t_S(6)$'s parent has value 0x8, and $t_S(7)$, $t_S(8)$, $t_S(9)$ and $t_S(10)$'s parents have value 0xF. In this manner, $\mathbf{v}_S$ is recovered at the decoder side from $\mathbf{f_v}$, and does not need to be transmitted.

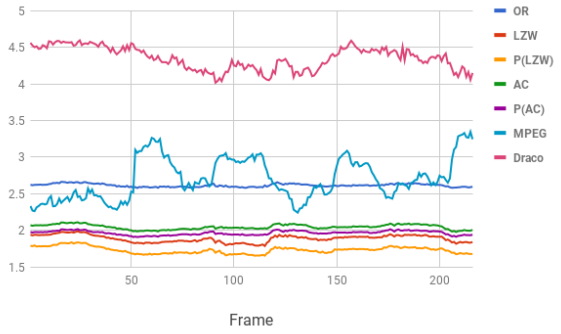
$\boldsymbol{\lambda}_1$ = [0xF] = [0b1111] indicates that the second level $\boldsymbol{\lambda}_2$ has 4 entries. With knowledge of $\mathbf{f_v}$, $\mathbf{t}_S$ and $\mathbf{v}_S$, the decoder infers that the entries in $\boldsymbol{\lambda}_2$ all have parent value 0xF, and that they correspond to the last 4 entries in $\mathbf{t}_S$: $\boldsymbol{\lambda}_2$ = [0x3 0x2 0x4 0x8]. $\boldsymbol{\lambda}_2$ is once again interpreted in the same manner, indicating two entries with parent value 0x3, one entry with parent value 0x2, one entry with parent value 0x4 and one entry with parent value 0x8. The end result is that $\boldsymbol{\lambda}_3$ = [0x1 0xC 0x9 0x8 0x8]. Further levels for deeper quadtrees/octrees would be reconstructed in the same manner.

III. EXPERIMENTAL RESULTS

Tests for the proposed methods were carried with six point-cloud sequences: *Andrew9*, *David9*, *Man*, *Phil9*, *Ri-*



(a)



(b)

Fig. 5. Rate on a frame basis, in bpov, for the scenarios proposed in Table I. Sequences [17]: (a) *Andrew9*; (b) *David9*.

Table I. Proposed testing scenarios.

MPEG	MPEG anchor code [13]
Draco	Draco open-source library [15]
OR	Octree representation
AC	Arithmetically-coded octree representation
P(AC)	Arithmetic-code-based proposed method
LZW	LZW-coded octree representation
P(LZW)	LZW-based proposed method

cardo and *Sarah9* [7][17]. The average rate was calculated for seven scenarios, as indicated in Table I. For arithmetic coding, an implementation for byte-sized symbols was used, and for LZW compression, GZIP was applied, using version 1.0.7 of the Keka application with the maximum-compression setting (*-mx9*) [18].

Table II presents the average rate in bpov for the seven proposed scenarios, and Table III presents the average percentage rate gain over the octree representation for the other six scenarios. It can be seen that: (a) the proposed methods **P(AC)** and **P(LZW)** outperform the **MPEG**, **Draco** methods in all sequences; (b) each proposed method outperforms the scenario using the same entropy coder, gaining an average 2% over arithmetically coding the octree representation and an average 5% over applying LZW to the octree representation. The arithmetically-coded-based method obtains a smaller average gain than the LZW-based method because it uses a smaller number of contexts, since $\mathbf{p}$ and $\mathbf{v}$ offer 8 and 255 contexts, respectively.

Table II. Average rate in bits per occupied voxel for the scenarios proposed in Table I.

Sequence	MPEG	Draco	OR	AC	P(AC)	LZW	P(LZW)
Andrew9	2.75	4.77	2.58	2.04	1.97	1.94	1.80
David9	2.71	4.35	2.62	2.05	1.97	1.89	1.73
Man	4.49	5.04	3.17	2.66	2.60	2.52	2.32
Phil9	2.89	4.54	2.64	2.10	2.03	2.00	1.84
Ricardo	3.53	4.82	2.92	2.43	2.37	2.37	2.27
Sarah9	3.10	4.86	2.61	2.04	1.97	1.89	1.75
Average	3.25	4.73	2.76	2.22	2.15	2.10	1.95

Table III. Average rate gain over the octree representation for the scenarios proposed in Table I.

Sequence	MPEG	Draco	AC	P(AC)	LZW	P(LZW)
Andrew9	-7%	-85%	+21%	+24%	+25%	+30%
David9	-3%	-66%	+22%	+25%	+28%	+34%
Man	-42%	-59%	+16%	+18%	+21%	+27%
Phil9	-9%	-72%	+21%	+23%	+24%	+30%
Ricardo	-21%	-65%	+17%	+19%	+19%	+22%
Sarah9	-19%	-87%	+22%	+25%	+28%	+33%
Average	-17%	-72%	+20%	+22%	+24%	+29%

Figures 5(a) and (b) present the rate on a frame basis, in bits per occupied voxel, for the proposed scenarios, considering sequences *Andrew9* and *David9*. The relatively low variance of rates for all scenarios in sequence *Andrew9* (Fig. 5(a)) indicates that it does not contain much movement. The opposite is true for sequence *David9* (Fig. 5(b)). Regardless of the movement characteristics of the sequences, these Figures show consistent gains offered by the proposed method, both for arithmetic coding and LZW compression.

IV. CONCLUSION

In this paper, a lossless intra-frame compression method for point-cloud geometry was presented, where different contexts for entropy coding were employed based on the octree structure. Experiments were carried with six point-cloud sequences, and results showed that the proposed method offers an average rate reduction of 29%, when compared to the octree representation, yielding a 5% gain over entropy encoding the octree.We believe the proposed method is the new-state-of-the-art in intra-frame geometry compression for point-clouds.

V. REFERENCES

[1] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin, and S. Izadi, "Holoportation: Virtual 3d teleportation in

real-time," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, New York, NY, USA, 2016, UIST '16, pp. 741–754, ACM.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] R. L. de Queiroz and P. A. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug 2016.

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

[9] 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.

[10] D. C. Garcia and R. L. de Queiroz, "Context-based octree coding for point-cloud video," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 1412–1416.

[11] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of point cloud video," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 1417–1421.

[12] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 778–785.

[13] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, April 2017.

[14] MPEG working group 3DG-PCC on PointCloud Compression, "Draft call for proposals for point cloud compression," *IEEE Transactions on Circuits and Systems for Video Technology*, October 2016.

[15] "Draco 3D graphics compression," https://google.github.io/draco/, Accessed: 2018-01-10.

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

[17] 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.

[18] "Keka - the macOS file archiver," http://www.kekaosx.com, Accessed: 2018-01-10.