LOSSY POINT CLOUD GEOMETRY COMPRESSION VIA DYADIC DECOMPOSITION

Davi R. Freitas, Eduardo Peixoto, Ricardo L. de Queiroz[*] *and Edil Medeiros*

Universidade de Brasília
Brasília, Brasil

rabbouni.davi@image.unb.br, eduardopeixoto@ieee.org, queiroz@ieee.org and j.edil@ene.unb.br

ABSTRACT

This paper proposes a lossy intra-frame coder of the geometry information of voxelized point clouds. Using an alternative approach to the widespread octree representation, this method represents the point cloud as an array of binary images. This algorithm works recursively using a dyadic decomposition that splits an interval of slices in two smaller intervals, depicting a binary tree traversal, and transmitting the occupancy information of each interval. The sequence of bi-level images are encoded in a lossless fashion until a fixed point in the tree, from where the algorithm "skips" the dyadic slicing and transmits all the k remaining slices as leaves of the tree, which are then encoded in a lossy fashion. The performance assessment shows that the proposed method outperforms state-of-the-art intra coders of lossy geometry for medium to higher bitrates on the public point cloud datasets tested.

Index Terms— Point clouds, lossy coding, geometry compression, intra coder

1. INTRODUCTION

Point clouds are important data structures in the context of virtual and augmented reality applications, as it provides a means of representation of objects with complex geometry in a flexible manner. A point cloud can be defined as a set of points in the 3D space, represented by coordinates (x, y, z), and optional attributes - generally color. Although the data volume of a point cloud is smaller than that observed in polygonal meshes [1], it is still significant and, thus, urges the development of efficient compression algorithms. While there are works involving the compression of non-voxelized (continuous) point clouds [2, 3, 4], the scope of this work focus on voxelized point clouds.

A great number of codecs use the octree [5] representation as means to structure point clouds and exploit it to compress the data, both for intra [6] and inter frame coding [7]. Lossless compression methods achieve significant bitrate reductions, but further savings can be accomplished by introducing lossy mechanisms in the compression scheme. Kammerl et al. [8] propose a lossy geometry approach based on a double buffering scheme for inter-frame coding. Mekuria et al. [9] approaches the problem of lossy geometry compression by constructing the octree up to only a certain level of detail. Oliveira et al. [10] set the octree representation as a base layer and a graph-based transform as an enhancement layer to achieve lossy compression. Quach et al. [11] propose a lossy geometry method based on a convolutional auto-encoder.

We note that representing point clouds by octrees is a popular approach. However, solutions using different methods of representation of point clouds have also been proposed. Daribo et al. [12] model the point cloud as a 3D space curve and compresses the geometry in a lossy fashion with arithmetic encoding. Zhu et al. [13] propose a lossless intra-frame geometry coding scheme by using binary tree partitioning. Milani et al. [14] introduces a lossy intra-frame algorithm for point cloud geometry based on a reversible cellular automata block transform. Peixoto [15] represents a point cloud as a sequence of silhouette images in a 3D cube and encode them using dyadic decomposition in a lossless algorithm.

The MPEG G-PCC *TMC13* v7.0 [16] is a geometry based codec that originated from the current proposals by the MPEG group. In addition to its lossless geometry compression capability, the *TMC13* coder has two methods for compressing lossy geometry [17]. The first is the direct geometry quantization, which will be referred in this work as *TMC13 Octree*. It works by quantizing the differences between each point of the point cloud and the minimum coordinates along its three axis. The other is the triangle surface approximation, referred here as *TMC13 Trisoup*. It is implemented by representing each voxel as a surface that intersects each of its edges. On the decoder side, this method constructs triangles from a set of vertices, which are the intersections between the surface and the voxel, and the lost voxels are estimated by extracting refined vertices from the constructed triangles.

While the lossless algorithm proposed by Peixoto [15] outperforms the lossless mode from *TMC13*, it does not feature a lossy compression technique. For this reason, we modify this previous work to propose a lossy intra-frame coder of the point cloud geometry in an effort to achieve further bitrate savings. Another contribution of this paper is a set of post-processing reconstruction techniques that improve the quality of the reconstructed images.

[*]Work partially supported by CNPq under grant 301647/2018-6

2. DYADIC DECOMPOSITION

Consider a voxelized point cloud of dimensions $N \times N \times N$ represented as a 3D boolean occupancy array $\mathbf{G}(x, y, z)$. The central idea of the dyadic decomposition method [15] is to recursively slice the 3D array $\mathbf{G}$ along a chosen axis of interest. At each step, an $N \times N$ *silhouette* image is obtained by projecting along the chosen axis all the occupied voxels in the region being processed. These images are then compressed by an arithmetic coder.

The recursive nature of the algorithm works by dividing an interval of slices in two smaller equal intervals, thus, implying a binary tree traversal. Each node of the dyadic binary tree holds the silhouette for the whole 3D region it represents. The rationale behind this dyadic decomposition is that from the silhouette image for an arbitrary node one can infer a great number of unoccupied voxels: all blank pixels in the silhouette image are empty because they were empty in all slices through that axis meaning that all subtrees from that node will present at least those same unoccupied voxels. The algorithm takes advantage of the similarities between the current node silhouette and its parent to mask information that can be inferred at the decoder. It also uses information of neighbor nodes to build contexts to a JBIG [18] inspired binary arithmetic coder. The slicing process continues until each leaf of the binary tree holds an atomic interval slice.

Experiments show that slices close to the leaves of the dyadic binary tree are not compressed as efficiently as slices closer to the root. Thus, as well as the dyadic decomposition method described above, another encoding approach called *single mode* encoding is also proposed in the previous work. The algorithm adds a parameter $P \in [1, N]$ and follows with the dyadic decomposition until the processing region comprise a volume of $k <= P$ slices. From this point on, the *single mode* approach "skips" the dyadic slicing and transmits all the k remaining slices as leaves of the tree, using the corresponding silhouette of this interval as a mask for the encoding of each leaf. For the implementation in [15], both single mode encoding and the dyadic decomposition methods are tested and the method yielding the lowest bitrate is chosen to compose the output bitstream.

2.1. Losing information to improve compression

In this paper we propose to introduce lossy mechanisms on the dyadic decomposition encoder to achieve lower bitrates without presenting significant geometry degradation. We perform the single mode encoding only using a fixed value of $P = 64$ slices. That is, we proceed with the dyadic decomposition with lossless encoding of the silhouettes until and including depth $\log_2 N - 5$. The algorithm then stops the dyadic slicing of the tree and jumps to the leaf images that will then be compressed in a lossy fashion. Fig. 1 shows an schematic representation of the tree construction. The specific choice of the parameter $P = 64$ was empirically determined and we regard further tuning of this parameter as a future work.

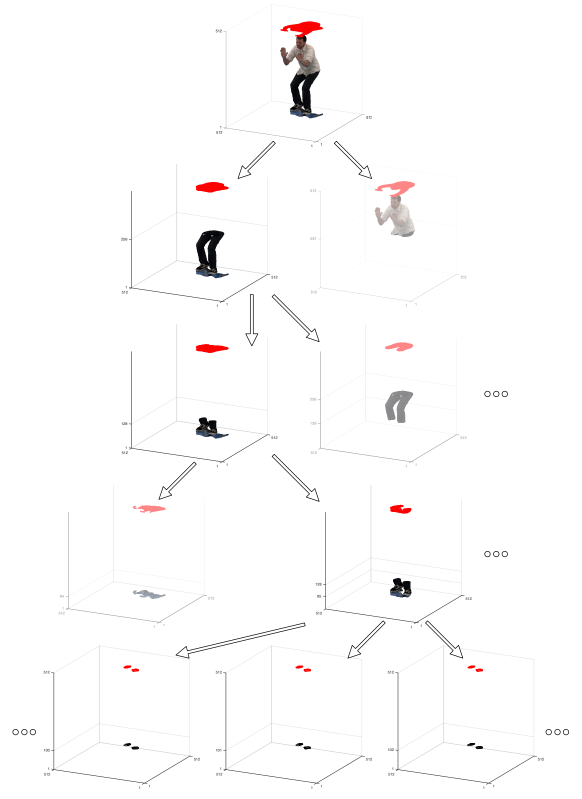

Fig. 1. Diagram showing dyadic decomposition and fixed single mode encoding with $P = 64$.

In order to compress the geometry in the single mode encoding, two lossy techniques were implemented. First, a *downsampling* technique is achieved by applying the non-adaptive, nearest neighbor interpolation method [19] to each single slice. Second, we sought to skip a certain number of consecutive slices, interpolating the intervening ones on the decoder side. For the rest of this paper, this second technique will be referred simply as *step*, where a step value of k means that $k - 1$ successive slices were skipped. These techniques can be combined to achieve different bitrates. Fig. 2 illustrates the lossy techniques combinations.

When the downsampling is used, the decoder can reconstruct the downsampled images with the nearest neighbor interpolation in order to estimate the voxels in that slice. When the step is used, a 3D linear interpolation to reconstruct the intervening skipped slices can be used.

We propose two reconstruction techniques to improve the quality of the reconstructed voxels without transmitting additional data. To recover some of the occupied pixels lost in the downsampling, a morphological operation of closing [20] is performed on the upsampled image at the decoder side. Then, the parent silhouette Y_S (Fig. 2) is used as a mask to constraint the region in which occupied voxels can be recovered. For

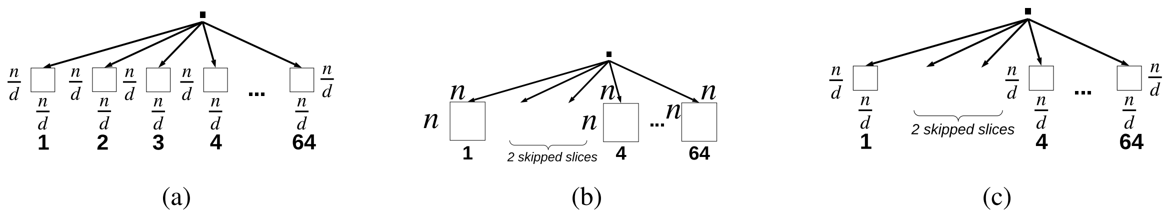

Fig. 2. Examples of application of lossy techniques in the single mode encoding: (a) downsampling; (b) *step* = 3; and (c) downsampling and *step* = 3

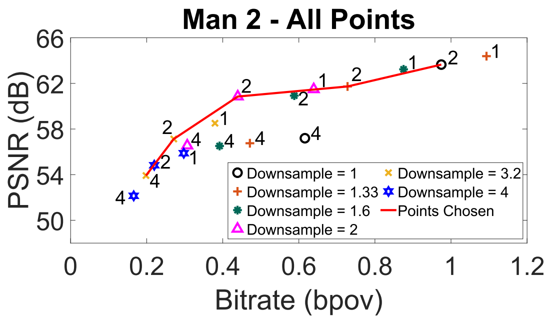

Fig. 3. Operation points for sequence *Man2*. The values near each point indicates its corresponding step value.

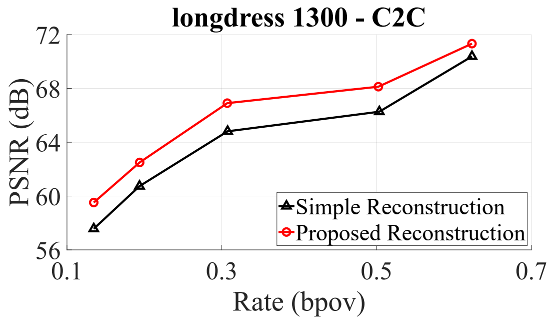

Fig. 4. Comparison of the proposed reconstruction methods.

the step technique, a basic interpolation algorithm between the non skipped frames (referred here as *antecedent* and *subsequent* frames) is performed to find the best fit between them - referred here as t. Once this value t is found, the intermediate slices are generated by translating the antecedent frame by fractions of t. As an example, suppose the best value $t = 1$ between two frames. For a step value of 4, the three intermediate frames are generated by translating the antecedent frame by values by $\frac{1}{4}$, $\frac{2}{4}$ and $\frac{3}{4}$. In addition, operations of morphological closing and the use of Y_S as a mask are performed to further enhance the interpolated slices.

3. EXPERIMENTAL RESULTS

In 2017, the Moving Picture Experts Group (MPEG) issued a Call for Proposals [21], where an objective performance assessment for compression solutions to point clouds was outlined. This evaluation is based on rate-distortion performance, where the rate is reported as bits per occupied voxel (bpov) and the distortion can be measured in terms of peak signal to noise ratio (PSNR) by two different metrics [22]: the point-to-point error (C2C) is obtained by calculating the Mean Squared Error (MSE) between the reconstructed points and their nearest neighbors in the reference point cloud. The second metric is the point-to-plane error (C2P), which is computed by taking into account the surface planes instead of the nearest neighbors [23].

A Matlab implementation of the proposed coder is available [24]. We ran the experiments in the publicly available JPEG Pleno Database [25], in particular in the 8i Voxelized Full Bodies dataset [26]. It consists of 4 sequences with 10 bit resolution: *LongDress*, *Loot*, *RedAndBlack* and *Soldier*.

3.1. Selection of the best coding parameters

The combination of different values of downsampling and step can provide distinct levels of distortion and bit rate on the point cloud. As a result, it is necessary to obtain a definite set of parameters to provide a rate-distortion (RD) optimized curve. Taking that into account, 18 different combinations of values were tested: for downsampling, values of 1, 1.33, 1.6, 2, 3.2 and 4 were analyzed; for step, values of 1, 2 and 4 were used. This test was performed over ten randomly selected frames of the point cloud sequence *Man 2* [27], resulting in the values observed in Fig. 3. We have chosen operation points that better cover the possible bitrates. The chosen values are: $(downsampling, step) = [(1, 2), (1.33, 2), (2, 2), (3.2, 2), (3.2, 4)]$.

3.2. Post-processing reconstruction improvements

Post-processing techniques were presented in Section 2.1 in an effort to improve the results obtained from simple reconstruction for both *step* and downsampling methods. In order to assess these techniques we ran a small experiment comparing the simple reconstruction with the proposed one, which is shown in Fig. 4. It can be seen in the figure that the proposed algorithms significantly improve the reconstruction quality.

3.3. RD performance evaluation

We evaluate our solution against the *TMC13* v7.0 codec to compare the results obtained from the proposed algorithm

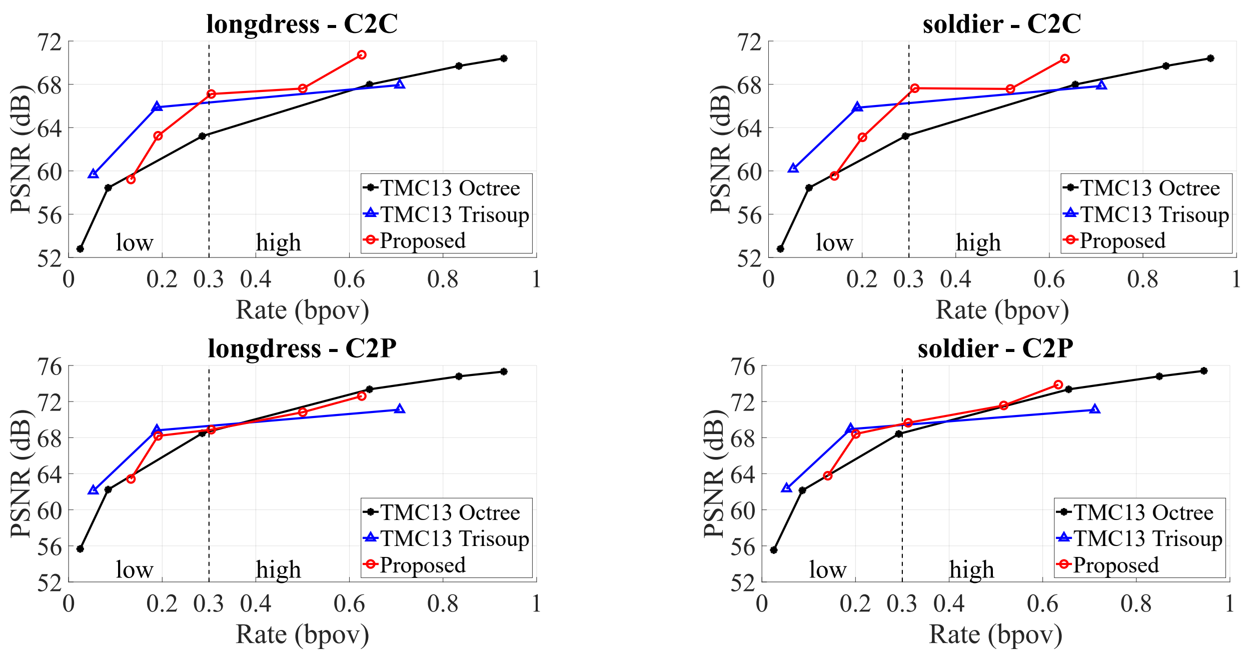

Fig. 5. RD performance results for (left) *LongDress* and (right) *Soldier*: (top) C2C and (bottom) C2P metrics.

Sequence	***C2C***						***C2P***					
	Proposed			**TMC 13 - Trisoup**			**Proposed**			**TMC 13 - Trisoup**		
	Low	High	Total	Low	High	Total	Low	High	Total	Low	High	Total
LongDress	2.10	**2.49**	2.10	4.28	0.99	3.07	1.58	**-0.31**	0.11	2.51	-1.27	1.48
Loot	0.34	**1.81**	0.88	4.47	0.81	3.37	1.67	**0.42**	0.57	2.78	-1.45	1.74
RedAndBlack	1.85	**2.53**	2.04	4.05	0.99	2.93	1.95	**0.08**	0.39	2.86	-0.96	1.71
Soldier	2.06	**2.67**	2.21	4.39	1.03	3.26	2.11	**0.55**	0.70	2.86	-1.09	1.84
Average	1.59	**2.38**	1.81	4.30	0.96	3.14	1.83	**0.19**	0.44	2.75	-1.19	1.60

Table 1. BD-PSNR comparisons of lossy geometry compression with state-of-the-art algorithms. All values are in dB. Results of the MPEG G-PCC TMC13 v7.0 with direct geometry quantization method are used as the benchmark.

with state-of-the-art solutions. The RD performance for C2C and C2P metrics averaged over the first 100 frames for the *LongDress* and *Soldier* sequences is shown in Fig. 5.

In order to better assess the proposed method we have divided the bitrate range in two ranges: a low range, up to 0.3 bpov, and a high range, from 0.3 bpov. In the figures, this is shown as a vertical dotted line. The experimental results are summarized in Table 1. The results were obtained by taking the average of the BD-PSNR values [28] - using TMC13 *Octree* as anchor - for both C2C and C2P metrics over the first 100 frames of each sequence. The BD-PSNR over the whole bitrate interval is also shown.

We can observe that both TMC13 *Trisoup* and our proposal perform better in the C2C metrics than the TMC13 *Octree*, with our proposal presenting better results than both algorithms for higher bitrates (an average of 2.4dB advantage over TMC13 *Octree* and 1.4dB over TMC13 *Trisoup*), while TMC13 *Trisoup* performs better for lower bitrates. For the C2P metrics, both algorithms again present better results than TMC13 *Octree*. However, TMC13 *Trisoup* performs worse for higher bitrates (average of -1.19dB), while our solution performs only slightly better with an average of 0.19dB.

4. CONCLUSIONS

This work presents a new approach for lossy intra compression of point cloud geometry using an alternative representation to the popular octree structure. It also proposes post-processing techniques that improve the quality of the reconstructed voxels. For bitrates greater than 0.3 bpov, the algorithm outperforms both lossy geometry compression methods from the state-of-the-art intra coder TMC13 v7.0 in the C2C metrics, while slightly outperforming them in the C2P metrics. Even for lower bitrates, the proposed algorithm still outperforms the *TMC13 Octree*, although the *TMC13 Trisoup* performs better in this range. For future work we plan to improve the proposed reconstruction algorithms, especially for lower bitrates, as well as expand the operation points of the codec to even lower bitrates.

5. REFERENCES

[1] S. Gebhardt, E. Payzer, L. Salemann, A. Fettinger, E. Rotenberg, and C. Seher, "Polygons, point-clouds and voxels: A comparison of high-fidelity terrain representations," in *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation—Processes, Standards, and Lessons Learned*, 2009, p. 9.

[2] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Continuous Point Cloud Data Compression Using SLAM Based Prediction," *2017 IEEE Intelligent Vehicles Symposium (IV)*, , no. Iv, pp. 1744–1751, 2017.

[3] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing Continuous Point Cloud Data Using Image Compression Methods," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1712–1719, 2016.

[4] X. Sun, H. Ma, Y. Sun, and M. Liu, "A Novel Point Cloud Compression Algorithm Based on Clustering," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2132–2139, 2019.

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

[6] D. C. Garcia and R. L. de Queiroz, "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, pp. 1807–1811.

[7] 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)*, Sep. 2017, pp. 1412–1416.

[8] 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.

[9] 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.

[10] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-Based Static 3D Point Clouds Geometry Coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 284–299, Feb 2019.

[11] M. Quach, G. Valenzise, and F. Dufaux, "Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression," *CoRR*, vol. abs/1903.08548, 2019.

[12] I. Daribo, R. Furukawa, R. Sagawa, and H. Kawasaki, "Adaptive arithmetic coding for point cloud compression," in *2012 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, Oct 2012, pp. 1–4.

[13] W. Zhu, Y. Xu, L. Li, and Z. Li, "Lossless point cloud geometry compression via binary tree partition and intra prediction," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct 2017, pp. 1–6.

[14] S. Milani, "Fast point cloud compression via reversible cellular automata block transform," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 4013–4017.

[15] E. Peixoto, "Intra-Frame Compression of Point Cloud Geometry using Dyadic Decomposition," *IEEE Signal Processing Letters*, pp. 1–1, 2020.

[16] 3DG, "G-PCC codec description v4," Tech. Rep., ISO/IEC JTC1/SC29/WG11/W18673, 2019.

[17] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A Comprehensive Study and Comparison of Core Technologies for MPEG 3-D Point Cloud Compression," *IEEE Transactions on Broadcasting*, vol. PP, pp. 1–17, 2019.

[18] ISO/IEC JTC 1 / SC 29 /WG 1 (ITU–T SG8), "Coding of Still Pictures," Tech. Rep. N 1359, ISO/IEC, July 1999.

[19] O. Rukundo, K. Wu, and H. Cao, "Image interpolation based on the pixel value corresponding to the smallest absolute difference," in *The Fourth International Workshop on Advanced Computational Intelligence*, Oct 2011, pp. 432–435.

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, Inc., USA, 2006.

[21] 3DG, "Call for Proposals for Point Cloud Compression (v2)," Tech. Rep., ISO/IEC JTC1/SC29/WG11/N16763, Hobart, Australia, Apr. 2017.

[22] S. Schwarz, G. Cocher, D. Flynn, and M. Budagavi, "Common test conditions for point cloud compression," Tech. Rep., ISO/IEC JTC1/SC29/WG11/N17766 ISO/IEC JTC1/SC29/WG1 M72012, Jul. 2018.

[23] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3460–3464.

[24] Davi R. Freitas, Eduardo Peixoto, Ricardo L. de Queiroz, and Edil Medeiros, "Geometry Coder Implementation.," https://github.com/pointcloud-unb/GeometryCoder/releases/tag/v1.1-lossy-ICIP2020, Online, published on 28-May-2020.

[25] "JPEG Pleno Database," `https://jpeg.org/plenodb/`, [Online].

[26] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft Voxelized Upper Bodies – A Voxelized Point Cloud Dataset," Tech. Rep., ISO/IEC JTC1/SC29/WG11 m38673 ISO/IEC JTC1/SC29/WG1 M72012, Geneva, Switzerland, Jan. 2017.

[27] 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.

[28] Gisle Bjøntegaard, "Improvements of the BD-PSNR Model," Tech. Rep., VCEG-AI11, ITU-T SG16/Q6, Berlin, Germany, Jul. 2008.