Attribute-aware Partitioning for Graph-based Point Cloud Attribute Coding

Thibaut Meyer, Maria Meyer, Dominik Mehlem, and Christian Rohlfing
Institut für Nachrichtentechnik, RWTH Aachen University, Germany
thibaut.meyer@ient.rwth-aachen.de

2022 Picture Coding Symposium (PCS) | 978-1-6654-9257-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/PCS56426.2022.10018065

***Abstract*—The unstructured nature of point cloud data makes compression of their attributes very challenging. In this paper, the known approach of using the Graph Fourier Transform on partitions of the point cloud is improved. It is proposed to make the partitioning process both geometry and attribute-aware, taking all of the point cloud's characteristics into account simultaneously. Additional information, that allows the decoder to reproduce the partitioning of the encoder, is added to the bit-stream. Furthermore, a refinement algorithm which re-estimates the partitioning information at the encoder with the decoder in mind is proposed. Experiments show that the baseline method is outperformed in Bjøntegaard Delta rate reduction by 2.39%, reaching as much as 3.58% at high bitrates.**

***Index Terms*—3D Point cloud compression, cluster-based partitioning, color attributes, Graph Fourier Transform**

I. Introduction

As methods of acquiring point clouds become more and more accessible, they constitute an increasingly popular way of representing 3D objects, which enable diverse applications e.g. free view-point rendering and autonomous driving. The data is initially a set of unrelated, unstructured points, which make point clouds very costly to store and transmit, thus driving the need for compression methods adapted to point cloud data.

The Moving Pictures Expert Group (MPEG) introduced two different point cloud coding standards: on one hand, V-PCC [1] finalized in 2021, which leverages existing video coding standards for compression, and on the other hand, G-PCC [2], which directly exploits the geometry of the point cloud and is still under development. Representing a point cloud as an octree is a popular technique used for partitioning the point cloud, which enables working at a block level. It not only provides an efficient way of signalling the partitioning, but is also very fast. Several schemes, e.g. [3], and [4] which is part of G-PCC, use it as part of their compression pipeline. Octree partitioning is improved by e.g. [5] who adaptively divides the voxels using non-square subdivisions. More recently, neural network-based solutions have been proposed for point cloud attribute compression as well, e.g. [6], [7].

In contrast, originally pioneered in [8], graph representations of point clouds have allowed the use of the Graph Fourier Transform (GFT) for compression of point cloud attributes to remarkable success. This approach has been the basis for several improved methods over the years, such as [9] which introduces the multi-resolution transform RA-GFT, or [10], which predicts the graph from a set of selected representative points. Cluster-based partitioning has been introduced more recently to provide a more flexible alternative to octree partitioning: [11] uses the geometry information of the points as features for clustering thus providing a data-dependent partitioning scheme. This has also been combined with the GFT by [12], [13].

The problem of point cloud attribute compression is tackled in this paper by extending the partitioning method of [12] which clusters the geometry information at both encoder and decoder. In contrast, utilizing both geometry and attribute information for cluster-based partitioning is proposed in this paper. This approach produces partitions that have smoother attributes which can be more efficiently represented in the GFT domain, eventually increasing the compression performance. Since the attributes are not available as input to the decoder, some side information is transmitted to help the decoder reproduce the encoder's partitioning. The code for our approach is available at https://github.com/IENT/AA-PCAC.

This paper is structured as follows: The fundamentals of using the GFT for point cloud attribute coding are summarized in Section II. Based on that, the novel attribute-aware partitioning method is introduced in Section III. Conducted experiments and their results are presented in Section IV and finally conclusions are drawn in Section V.

II. Attribute Coding with Graph Fourier Transform

Point clouds consist of N points, each possessing geometry information $v_i = (x_i, y_i, z_i)^\top \in \mathbb{R}^3$ and corresponding attributes $a_i \in \mathbb{R}^3$. In this paper, it is assumed that the attributes hold 8-bit RGB color information such that $a_i = (r_i, g_i, b_i)^\top \in [0, 255]^3$. It is assumed that the points are voxelized and thus the point cloud has an intrinsic resolution.

In the following sections, two previous methods [8], [12] for coding point cloud attributes are summarized. As shown in Fig. 1, the point cloud is first divided into partitions which are then each turned into a graph, enabling the usage of the Graph Fourier Transform (GFT) for transforming the l-th partition's attributes A_l as the second step. The resulting coefficients $\hat{A}_l$ are then quantized, coded and send to the decoder which inverts these steps to obtain the coded attributes $\tilde{A}$.

A. Partitioning

Applying the GFT to a graph representing the entire input point cloud is not computationally feasible, thus it is necessary to divide the point cloud into blocks or partitions. Traditional

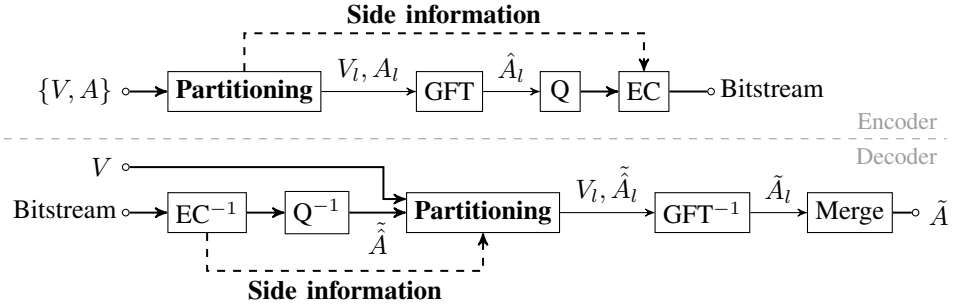


Fig. 1: Point cloud attribute compression scheme [8] with proposed modifications (in bold).

methods, e.g. [8], use octree partitioning for dividing the 3D space evenly into eight voxels called octants. Octants not containing any points are discarded, while the others are recursively divided a given number of times in the same way. The resulting partitions of the point cloud can then be represented using an octree where each node has a maximum of eight children.

However, this paper focuses on data-driven partitioning instead, as in [12], where geometry-based clustering yields non-uniform partitions. There, the k-means algorithm is used to partition the point cloud by using the points' geometry information v_i as features for clustering at both encoder and decoder. The number of clusters k is set to be dependent on the number of points N. This method creates partitions that are more similarly sized, and which further account for the geometry characteristics of the point cloud compared to the partitions represented by an octree.

B. Graph Fourier Transform

The sub-point cloud contained in each partition is turned into a graph $\mathcal{G} = (\mathcal{N}, W)$ where $\mathcal{N} = \{n_1, n_2, \ldots, n_{|\mathcal{N}|}\}$ is a set of nodes and W is the weighted adjacency matrix. The nodes are chosen as the points of the sub-point cloud and $(W)_{ij} = w_{ij}$, if non-zero, is the weight of the edge connecting the points n_i and n_j. Several methods were proposed to determine the graph's edges and their weights w_{ij}: In [8], neighboring nodes are connected if their positions vary by at most 1 along any axis and sets w_{ij} to the inverse of the Euclidean distance between the two nodes. For the Normal Weighted Graph Fourier Transform (NWGFT) [12], the weights are computed using local spatial features of the nodes. While the geometric similarity between nodes is more accurately represented by the resulting graph, it also introduces several additional parameters and overhead. In this work the method of [8] is used to compute the adjacency of the graphs.

Let $L = D - W$ be the combinatorial graph Laplacian matrix of a given graph $\mathcal{G}$ where $D = \mathrm{diag}(d_i)$ and $d_i = \sum_j w_{ij}$. Since L is positive semidefinite, it has a full set of orthogonal eigenvectors U that can be obtained using the eigen-decomposition of L:

$$L = U \Lambda U^\top \ . \tag{1}$$

$U^\top$ is used as the Graph Fourier Transform matrix to project a signal defined on the graph's nodes y onto the GFT domain i.e. $\hat{y} = U^\top y$. The signal can be retrieved with $y = U\hat{y}$.

III. Geometry-and-Attribute-aware Partitioning

In Fig. 1, the classic compression scheme from [8] that codes the point cloud's attributes A using the GFT on partitions of the point cloud is shown. The proposed modifications are highlighted and consist of an attribute-aware partitioning method, not only exploiting the geometry V but also the attributes A, extending the approach of [12]. Since the attributes are not available at the decoder the necessary information that will enable the decoder to reproduce the same partitions as the encoder has to be send additionally.

A. Clustering Scheme

The main goal of the proposed method is to generate partitions that have smooth attributes, in order to enhance the compression of the subsequent GFT. Here, a clustering-based partitioning method is utilized. In contrast to [12], not only the geometry V but also the attributes A are used as features for clustering at the encoder. Prior to clustering, the attributes are converted from RGB to the CIE L*a*b* color space [14] and denoted again as a_i for sake of simplicity. Geometry and attributes are combined as

$$x_{VAi} = \begin{pmatrix} \bar{v}_i \\ \lambda \bar{a}_i \end{pmatrix} \in \mathbb{R}^6 \tag{2}$$

where v_i, a_i are scaled to zero mean and unit variance yielding $\bar{v}_i, \bar{a}_i$. The steering parameter λ controls the importance of the attributes during clustering.

Using k-means on the concatenated data given by (2) to generate the cluster labels L_{VA} can be alternatively expressed with the following label update rule

$$l_{VAi} = \arg\min_j \|\bar{v}_i - c_{Vj}\|_2 + \lambda \|\bar{a}_i - c_{Aj}\|_2 \tag{3}$$

where c_{Vj}, c_{Aj} denote the j-th center in either the geometry or the attribute domain, respectively. Note that for $\lambda = 0$, the partitioning becomes equivalent to the method of [12]. The value of λ could be chosen using a rate-distortion-based decision.

$C_{VA} \in \mathbb{R}^{k \times 6}$ are cluster centers that comprise both geometry and attributes, and are thus obtained by combining c_{Vj}

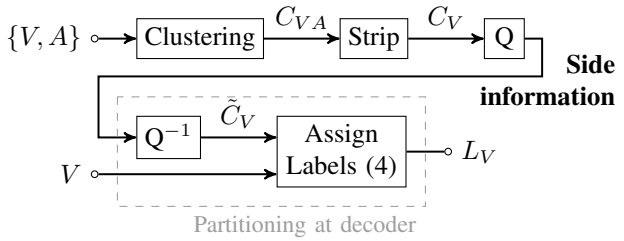


Fig. 2: Proposed encoder partitioning scheme containing the corresponding decoder steps.

and $c_{A\,j}$ akin to (2). Since only the geometry information V is available at the decoder, the last three dimensions of C_{VA}, containing the centers w.r.t. the attributes, are dropped yielding the geometry centers $C_V \in \mathbb{R}^{k\times 3}$. These centers, which constitute the necessary side information, are quantized, coded, and sent to the decoder.

At the decoder, only the geometry V and the quantized centers $\tilde{C_V}$ are at hand. The partition labels L_V are thus obtained by setting $\lambda = 0$ in (3) with

$$l_{V\,i} = \arg\min_j \|\bar{v}_i - \tilde{c_{V\,j}}\|_2 \ . \tag{4}$$

Note that this label assignment step is executed only once, which makes the computational overhead for the decoder very low, compared to [12]. In order to ensure consistent partitions between encoder and decoder, the encoder uses the final labels L_V as well by simulating the decoder at the encoder i.e. decoding C_V and applying (4) once subsequently. The encoder's partitioning algorithm is summarized in Fig. 2, where the part that simulates the decoder is outlined accordingly. Please note that the Mini-batch k-Means algorithm [15] is used throughout this work for clustering[1].

B. Refinement Algorithm

In Section III-A the partition centers C_V are calculated at the encoder with both geometry and attributes at hand. However, the fact that the decoder has solely access to the geometry for obtaining the partition labels given by (4) was neglected in the encoder clustering step until now. In this section, a refinement algorithm which re-calculates the centers with the decoder in mind is proposed: For initialization, it is assumed that k-means was conducted as described in the previous section.

To refine the centers, first a weighting factor is calculated for each point i in partition j

$$\alpha_i = p_{\text{norm}}\left(a_i, \mu_{a,j}, \Sigma_{a,j}\right) \tag{5}$$

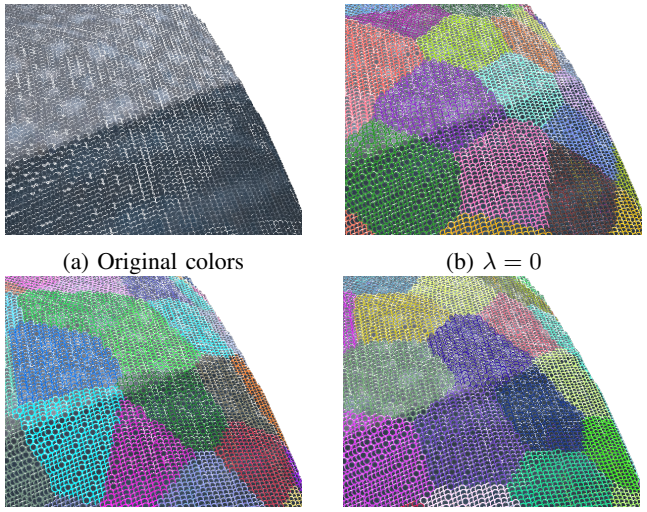
with $p_{\text{norm}}(x, \mu, \Sigma)$ being the multivariate normal distribution and $\mu_{a,j}, \Sigma_{a,j}$ mean vector and covariance matrix estimated from the corresponding attributes of points belonging to partition j. Note that this ensures that points which color is close to the partition's average color are assigned with a large weighting factor.

Second, the weights are then used to update the centers as a weighted mean

$$c_{V,\text{ref}\,j} = \frac{1}{\sum_i \alpha_i} \sum_{i\in\mathcal{P}_j} \alpha_i v_i \tag{6}$$

with $\mathcal{P}_j$ containing all points belonging to partition j (all points for which $l_i = j$ holds). The labels are again updated with the decoder update rule (4). This resembles a weighted k-means procedure [16], with the important difference being that the label update step always assumes uniform weights, since the weights cannot be calculated at the decoder. Note that this refinement step is iteratively applied I times.

Fig. 3 displays an exemplary point cloud with different partitions: In Fig. 3b, the partitions obtained by [12] by setting $\lambda = 0$ in (3) are shown. Enabling $\lambda > 0$ yields partitions which take the color information into account, as can be seen in Fig. 3c . Fig. 3d illustrates that additionally enabling the proposed refinement algorithm yields even better partitions compared to Fig. 3c. This becomes especially evident at the boundary between light and dark blue colored points.



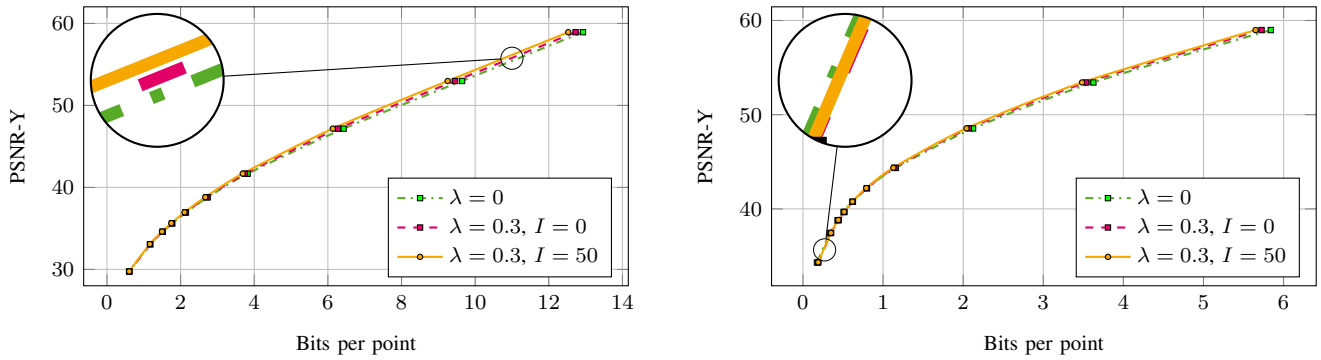
(a) Original colors (b) $\lambda = 0$ (c) $\lambda = 0.3$ (d) $\lambda = 0.3$ with refined centers

Fig. 3: Comparison of different partitioning schemes for exemplary extract from "Loot" sequence: Baseline method [12] with $\lambda = 0$ compared to proposed method with $\lambda = 0.3$ without and with refined centers.

IV. Experimental Results

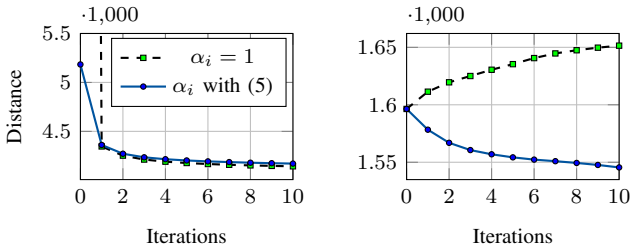
The point cloud sequences "Loot", "Longdress", "Soldier" and "Redandblack" from the "8iVFBv2" data set [17] are used. They represent full human bodies with different levels of attribute complexity and possess from 700K to 1M points. Here, $T = 30$ frames uniformly distributed over time are evaluated. For partitioning, the number of centers is set to $k = 1500$. The attributes are converted to the Y, U, V color space and are then processed independently by the

[1]Note that the proposed method is not restricted to this particular clustering scheme and that any other center-based algorithm could be used instead.



(a) Longdress (b) Loot

Fig. 4: Rate-quality curves comparing baseline [12] with $\lambda = 0$ to proposed partitioning scheme with $\lambda > 0$ with either deactivated ($I = 0$) or activated ($I = 50$) proposed refinement algorithm.



(a) Distances $\|v_i - c_{Vj}\|_2$ (b) Distances $\|a_i - c_{Aj}\|_2$

Fig. 5: Convergence of proposed refinement algorithm on "Loot" sequence, frame 1000 with $\lambda = 0.3$.

GFT. The resulting coefficients are uniformly quantized using quantization step sizes $q \in [1, 64]$, and entropy-coded using adaptive run-length Golomb-Rice (RLGR) coding [18]. The partitioning centers are also uniformly quantized with a fixed step size $q_{\text{centers}} = 10$ and entropy-coded using RLGR as well. Note that Y, U, V are coded independently. To measure the quality of the decoded point clouds, the peak signal-to-noise ratio (PSNR) of the Y component is calculated as

$$\text{PSNR-Y} = -10 \log_{10} \left(\frac{1}{T} \sum_{t=1}^{T} \frac{\|Y_t - \tilde{Y}_t\|_2^2}{255^2 N_t} \right) \tag{7}$$

where N_t is the number of points present in the t-th frame, Y_t and $\tilde{Y}_t$ are the original and decoded Y component of frame t respectively.

A. Centers Refinement

Fig. 5 shows the convergence behaviour of the refinement algorithm depending on the weights chosen for updating the centers. The weights are either computed with respect to (5) (—•—) or set to $\alpha_j = 1$ (- ▪ -). Distance functions between data points and cluster centers on both geometry and attributes are shown, whereas intermediate centers for the attributes were calculated as $c_{Aj} = \frac{1}{|\mathcal{P}_j|} \sum_{i \in \mathcal{P}_j} a_i$ with $\mathcal{P}_j$ containing all points belonging to partition j given by the decoder labels $l_i = j$. Both variants yield in a decreasing geometry distance as shown in Fig. 5a. However, the attribute distance diverges when weights $\alpha_j = 1$ are used, whereas the weight update (5) is able to decrease the attribute distances as demonstrated in Fig. 5b. In summary, calculating the geometry centers as a weighted mean, with attribute-dependent weights, results in partitions at the decoder that have smoother attributes.

B. Attribute-aware partitioning

By setting $\lambda = 0$, the attributes are ignored when partitioning the pointcloud, and no additional information is sent to the decoder, using only the geometry for clustering. This makes the partioning comparable to [12] and provides us with a baseline method (- ▪ - on Fig. 4).

Since the centers are provided to the decoder as side information in our approach, and because their size are independent from the quantization step size q used to compress the attributes coefficients, they constitute, for a given q_{centers}, a constant overhead at any rate range. This makes them represent a bigger proportion of the total bitstream size especially at lower bitrates. This can be seen in Fig. 4, where the curve - ▪ - for $\lambda = 0$ shows better performance for low bitrate ranges. However, once the mid and high-bitrate points are reached, having $\lambda \neq 0$ - ▪ - provides gains that are able to offset the induced cost of the side information. Additionally using the refinement algorithm —•— improves the compression even further.

These results are summarized again in Table I in terms of Bjøntegaard Delta (BD) rate changes [19][2] with respect to $\lambda = 0$. For all sequences, values are given for low, mid and high rate intervals using $\lambda = 0.3$ with either the refinement algorithm disabled ($I = 0$) or enabled with $I \in \{10, 50\}$. It can be seen, that almost no improvement could be achieved at lower bitrate ranges. However, compression gains ranging from 1.16% to 3.97% can be observed for the more complex "Longdress" sequence in the mid and higher bitrates. On the more simple "Loot" sequence, compression gains from 0.71% up to 3.82% are attained for mid and higher bitrates. For the

[2]The two rate-PSNR curves to be compared are interpolated with a piecewise cubic interpolation scheme.

TABLE I: BD rate changes in % with respect to $\lambda = 0$. Low: $q \in [24, 32, 64]$, Mid: $q \in [8, 12, 16, 20]$, High: $q \in [1, 2, 4]$.

	Longdress				Loot				Soldier				Redandblack				Average
I	Low	Mid	High	All	Low	Mid	High	All	Low	Mid	High	All	Low	Mid	High	All	All
0	0.81	-1.16	-1.94	-1.32	6.07	0.10	-2.40	-0.26	1.97	-1.88	-1.99	-1.39	3.29	-0.05	-1.47	-0.33	-0.83
10	-0.42	-2.18	-3.40	-2.62	5.13	-0.71	-3.27	-1.10	1.09	-2.23	-2.47	-1.89	1.56	-1.77	-3.24	-2.12	-1.93
50	-0.75	-2.63	-3.97	-3.14	4.83	-1.16	-3.82	-1.59	1.00	-2.43	-2.73	-2.11	0.98	-2.41	-3.81	-2.74	-2.39

other two sequences, similar results can be obtained. Note that increasing the number of refinement iterations improves the performance in all rate intervals for all sequences, yielding in an average of 3.58% savings for high bitrates. Finally, it is important to mention that the proposed algorithm is a direct extension of [12]. Deciding wich λ to use can be easily made dependent on the quantization step size q: For lower bitrates, $\lambda = 0$ should be favored, whereas $\lambda > 0$ yields better performance for higher bitrates. This could be also employed as a rate-distortion-based decision.

V. Conclusions and Future Work

In this paper, a novel attribute-aware partitioning scheme for point cloud attribute coding is presented. Better compression is achieved by using both geometry and attribute information for partitioning. This yields a more efficient representation of the attributes in the Graph Fourier Transform domain, thus increasing the overall compression performance. The side information needed to assist the decoder with partitioning is optimized at the encoder. The proposed method meets the state of the art in terms of compression performance: On average, the baseline method is outpeformed by 2.39% for all bitrates in terms of Bjøntegaard Delta rate savings. At high bitrates, the savings even reach 3.58%. Since the proposed work extends the baseline, its gains at lower bitrates can be obtained by adapting the steering parameter depending on the quantization step size.

Several ideas could be explored in the future in order to improve this work: Only k-means has been investigated as a clustering algorithm where other methods could potentially offer better partitioning. Regarding the centers refinement algorithm, it would be worthwhile experimenting with other distributions for obtaining the weights. And finally it would be interesting to try to compress the centers themselves by quantizing them coarser before transmission, which might be advantageous at lower bitrate ranges.

Acknowledgment

The authors would like to thank the reviewers for their thoughtful comments that helped to improve this manuscript.

References

[1] "Information technology – Coded representation of immersive media – Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)," International Organization for Standardization, Geneva, CH, Standard, 2021.

[2] "Final Draft: Information technology – Coded representation of immersive media – Part 9: Geometry-based point cloud compression," International Organization for Standardization, Geneva, CH, Standard, 2022.

[3] S. Gu, J. Hou, H. Zeng, H. Yuan, and K.-K. Ma, "3d point cloud attribute compression using geometry-guided sparse representation," *IEEE Transactions on Image Processing*, vol. 29, pp. 796–808, 2020.

[4] 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, 2016.

[5] X. Zhang and W. Gao, "Adaptive geometry partition for point cloud compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4561–4574, 2021.

[6] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu, "Deep-pcac: An end-to-end deep lossy compression framework for point cloud attributes," *IEEE Transactions on Multimedia*, vol. 24, pp. 2617–2632, 2022.

[7] G. Fang, Q. Hu, H. Wang, Y. Xu, and Y. Guo, "3dac: Learning attribute compression for point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2022, pp. 14 819–14 828.

[8] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2066–2070.

[9] E. Pavez, B. Girault, A. Ortega, and P. A. Chou, "Region adaptive graph fourier transform for 3d point clouds," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2726–2730.

[10] S. Gu, J. Hou, H. Zeng, and H. Yuan, "3d point cloud attribute compression via graph prediction," *IEEE Signal Processing Letters*, vol. 27, pp. 176–180, 2020.

[11] K. Zhang, W. Zhu, and Y. Xu, "Hierarchical segmentation based point cloud attribute compression," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 3131–3135.

[12] Y. Xu, W. Hu, S. Wang, X. Zhang, S. Wang, S. Ma, and W. Gao, "Cluster-based point cloud coding with normal weighted graph fourier transform," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 1753–1757.

[13] Y. Xu, W. Hu, S. Wang, X. Zhang, S. Wang, S. Ma, Z. Guo, and W. Gao, "Predictive generalized graph fourier transform for attribute compression of dynamic point clouds," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 5, pp. 1968–1982, 2021.

[14] "Colorimetry — Part 4: CIE 1976 L*a*b* colour space," International Organization for Standardization, Geneva, CH, Standard, 2019.

[15] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 1177–1178.

[16] H. Liu, J. Wu, T. Liu, D. Tao, and Y. Fu, "Spectral ensemble clustering via weighted k-means: Theoretical and practical evidence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1129–1143, 2017.

[17] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies – A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29/WG1 input document M74006 and ISO/IEC JTC1/SC29/WG11 input document M40059, Geneva, Doc., Jan. 2017.

[18] H. Malvar, "Adaptive run-length/golomb-rice encoding of quantized generalized gaussian sources with unknown statistics," in *Data Compression Conference (DCC'06)*, 2006, pp. 23–32.

[19] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," ITU-T SG16/Q6 VCEG, Austin, Texas, USA, Doc. VCEG-M33, Apr. 2001.