Integrated Learning-Based Point Cloud Compression for Geometry and Color with Graph Fourier Transforms

Davi Lazzarotto and Touradj Ebrahimi

Multimedia Signal Processing Group (MMSPG), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

ABSTRACT

Point cloud representation is a popular modality to code immersive 3D contents. Several solutions and standards have been recently proposed in order to efficiently compress the large volume of data that point clouds require, in order to make them feasible for real-life applications. Recent studies adopting learning-based methods for point cloud compression have demonstrated high compression efficiency specially when compared to the conventional compression standards. However, they are mostly evaluated either on geometry or color separately, and few learning-based joint codecs have been proposed. In this paper, we propose an integrated learned coding architecture by joining a previously proposed geometry coding module based on three-dimensional convolutional layers with a color compression method relying on graph Fourier transform (GFT) using a learning-based mean and scale hyperprior to compress the obtained coefficients. Evaluation on a test set with dense point clouds shows that the proposed method outperforms G-PCC and achieves competitive performance with V-PCC when evaluated with state-of-the-art objective quality metrics.

Keywords: Point cloud, compression, graph Fourier transform, geometry coding, color coding

1. INTRODUCTION

Although the majority of digital visual information is conveyed in the form of two-dimensional media formats, new modalities more suitable to immersive applications are becoming increasingly popular. As a result, a great amount of attention has been devoted by the scientific community to techniques that enable the use of such modalities with current network, storage and computing infrastructure. Since the majority of such media formats require a rich spatial representation, the amount of data required for their use can be very large. For that reason, compression algorithms that effectively reduce their size with low impact on perceived visual quality are required. Standardisation committees such as JPEG and MPEG have been working towards interoperable compression standards for immersive contents such as omnidirectional , point clouds and dynamic meshes. Likewise, JPEG is creating the JPEG Pleno framework, focusing on the compression of hologram, light field and point cloud images.

The algorithms and data structures used to compress the geometric coordinates of point clouds can greatly vary in their nature. Among the many data structures employed for their representation, octree are among the most prominent. Octree-based compression methods partition the 3D space recursively into eight cubes representing quadrants, each cube being then represented in a tree by a binary occupancy value. Other algorithms divide the point cloud into patches and project each patch onto a plane, representing the position of each point as its distance to the plane, resulting in sets of two-dimensional depth maps. These maps are then encoded with video compression algorithms. In recent years, learning-based algorithms have emerged as an alternative with high compression performance, representing voxelized point clouds as occupancy maps and encoding them with a convolutional autoencoder architecture.

In addition, points have often associated attributes such as color triplets, normal vectors and reflectance values. The quality of the color values have an important impact on the subjective quality, since they are directly used to render the three dimensional model. Due to the different nature of color information when compared

Further author information: (Send correspondence to the authors)
E-mail: davi.nachtigalllazzarotto@epfl.ch, touradj.ebrahimi@epfl.ch

to geometry, attribute compression methods often employ distinct techniques. In particular, although fewer learning-based compression methods have been proposed for color than for geometry, recent studies have explored innovative methods such as learned volumetric representations[1] and sparse convolutions.[2] However, contrary to conventional algorithms for which several integrated frameworks have been implemented for compressing both geometry and color of point clouds,[3,4] including the MPEG compression standards V-PCC[5] and G-PCC,[6] not many similar studies have been carried out with learned architectures. The first study where such techniques were employed for both geometry and color compression in an integrated architecture was presented by Alexiou et al.[7] However, the distortion level of the color attributes saturates at higher bitrates without reaching transparent quality, and the performance is therefore not comparable to G-PCC and V-PCC. Recently, Guarda et al.[8] implemented a joint codec following a similar approach with proposed improvements and achieved competitive performance to the MPEG standards.

This paper adopts a different approach and combines distinct algorithms for geometry and color compression in a combined architecture competitive with the MPEG standards for the compression of dense point clouds. While the geometry coding is performed with a previously proposed method that uses 3D dense convolutional layers to compress occupancy maps with an enhanced channel-wise autoregressive entropy model, the color coding module employs the Graph Fourier Transform (GFT) to decorrelate the attribute vectors and uses a mean and scale hyperprior to guide the entropy coding process.

2. RELATED WORK

Recent studies on point cloud lossy compression have explored both conventional and learning-based techniques to achieve high compression ratios for geometry and color while maintaining acceptable visual quality.

2.1 Geometry compression

Early works[9,10] have employed the octree to compress point cloud geometry with prediction algorithms to enhance rate-distortion performance. Subsequent studies[3,4] dropped the intra prediction in order to reduce computational complexity and achieve real-time performance. Prediction between frames is however employed in order to ensure higher compression efficiency for dynamic point clouds. This strategy was also used in the popular Point Cloud Library (PCL) framework.[11] Other works[12] pruned the octree only at a certain depth, modeling the leaf nodes as triangular primitives. Recently, the MPEG standardisation committee released the Geometry-based point cloud compression (G-PCC) standard,[6] which is based on the octree for geometry coding and has an additional coding module for coding leaf nodes as triangles.[12] Alternatively, a projection-based algorithm for point cloud compression was introduced in an early study[13] and later employed in the Video-based point cloud compression (V-PCC)[5] standard, achieving very high performance for dense point cloud models.

Learning-based methods for point cloud geometry compression have been first studied by two early works[14,15] that adapted autoencoder architectures previously used for images[16] to compress voxelized point cloud blocks. These algorithms represent blocks as occupancy maps, which can be seen as 3D grids where each position receives a value of 1 if it is occupied by a point and 0 otherwise. The encoder downsamples the input block generating latent values that go through an entropy coder, which assumes a fully factorized probability distribution learned during training. In the decoder, the output probability map is binarized using a threshold value of 0.5 in order to generate an occupancy map that can be converted again to a point cloud block. In a subsequent work,[17] this architecture was improved by the addition of residual blocks, adaptive thresholding and a hyperprior model to model the entropy of the latent features, as proposed in previous works[18] for image compression. Similarly, other authors proposed an adaptive framework[19] to achieve better performance for sparse point clouds and a novel loss function[20] to better capture spatial context during training. Channel-wise autoregressive models were explored by Frank et al.[21] to increase compression efficiency by exploring redundancies between channels in the latent domain. Learning-based versions of methods popular in common video codecs were also studied, such as intra prediction[22] and residual coding.[23] An important development was introduced by Wang et al.,[24] which used sparse convolutions in order to take better advantage of the fact that the majority of voxels in point clouds are empty, operating only on positions that are effectively occupied by points. Sparse convolutions were further explored by the same authors[25] using cross-scale and cross-stage prediction to achieve lossless compression as well as superior lossy rate-distortion performance.

2.2 Color compression

Since the color structure depends on the underlying topology, many conventional color compression algorithms take as input the geometric coordinates assuming they have been encoded. The work from Zhang et al.[26] proposed the use of the Graph Fourier Transform (GFT) to decorrelate the color attributes prior to entropy coding. A graph is generated from the geometry by connecting neighbouring points with edges weighted according to their distance, and the point cloud is partitioned into cubic blocks. A later work[27] improved this work with a K-d Tree guided block partitioning process which avoided the formation of disjoint graphs. A laplacian sparsity optimized mechanism is also added to build the graphs for each block. This method was further leveraged by a hybrid framework[28] which selects either the GFT or the DCT during encoding in an adaptive process. Another work[29] adopted a partition process based on K-means clustering, and used the angular similarity between normal vectors to obtain an underlying graph. A recent study[30] uses the attribute vectors themselves to refine the graphs, transmitting adjacency matrices as penalty function variables in the form of side information.

Although the GFT is able to achieve higher coding performance for color attributes, the eigendecomposition of the laplacian matrix during the GFT is a computationally expensive process. A low-complexity method was proposed by Queiroz et al.,[31] which decomposes the point cloud into a binary tree and applies the region adaptive hierarchical transform (RAHT) on the attributes and encodes only high frequency components. Another popular method is the lifting transform, which separates the points into level of details and uses lower level points to predict the attributes of the higher levels. Both methods are included in the G-PCC[6] compression standard for attribute coding.

In order to employ learned transforms for point cloud color coding, the work from Alexiou et al.[7] extended a previous architecture[17] to receive as input color attributes as well. Quach et al.[32] later proposed a learned transform that mapped points from the three-dimensional domain to a 2D map through folding operations. The color is then projected to this map and encoded using a standard image codec. The work from Isik et al.[1] learns a volumetric function that maps geometric coordinates into color attributes through a lightweight neural network that is overfitted to the encoded point cloud and transmitted to the decoder. The work from Wang et al.[2] employs sparse convolutions to encode color attributes, also adopting a hyperprior and autoregressive context models to entropy code the obtained latent features. Following a similar approach as a previous work,[7] Guarda et al.[8] takes geometry and color together as input channels and applies 3D convolutions, proposing improvements such as a hyperprior for entropy coding and binarization optimization.

3. PROPOSED METHOD

3.1 Integration between geometry and color coding

The proposed codec for compression of both geometry and color attributes employs two separate coding methods. While the geometry compression algorithm is based on an autoencoder architecture with 3D convolutional layers, the proposed method for color coding applies the GFT on color attributes and uses a learning-based entropy coding module to compress the coefficients. For that reason, during encoding, the geometry is first compressed and the generated bitstream is kept. This bitstream is decoded, generating a geometry-only version of the reconstructed model. Each point of the reconstructed model then receives the color value of its closest neighbour from the original model, and the transferred color values are encoded by the color algorithm, which also takes as input the decoded geometry. Both geometry and color bitstreams are concatenated and sent to the decoder. At the decoder side, the geometry bitstream is decoded and the reconstructed geometry is obtained, while the color decoder takes both the color bitstream and the reconstructed geometry as input to finally generate the decoded point cloud with both geometry and color. This process is illustrated in the block diagram of Figure 1.

3.2 Geometry coding

The employed geometry coding algorithm is the same as that presented by Frank et al.[21] It is based on a convolutional autoencoder that operates on block partitions of a voxelized point cloud, which are converted into occupancy maps prior to compression. Each occupancy map block, denominated $x_{\text{g-occ}}$, is fed to the autoencoder architecture, where it goes through the analysis transform. This transform is composed of downsampling layers and residual blocks, all based on learned three-dimensional convolution operations which reduce the spatial

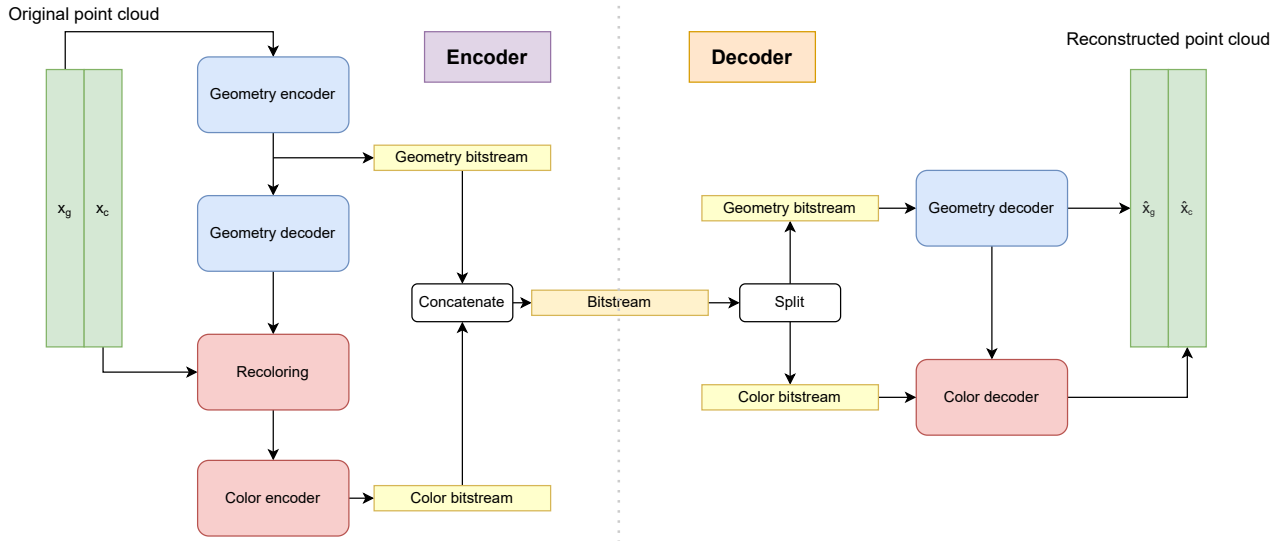


Figure 1: Block diagram of the integrated geometry and color compression.

dimensions of the block by a factor of 8. The analysis transform yields the latent representation y_g, which is composed by K channels. This latent representation is used to feed the hyper-analysis transform, yielding the tensor z_g, which is quantized into $\hat{z_g}$, entropy encoded assuming a fully factorized learned probability distribution and added to the bitstream as side information. The tensor $\hat{z_g}$ goes then through the hyper-synthesis transforms, yielding the tensors μ_g and σ_g which are later used to entropy code the values from y_g.

The latents y_g are partitioned channel-wise prior to entropy coding into N slices with K/N channels $\{y_{g1}, y_{g2}, ..., y_{gN}\}$. Each slice y_{gi} is quantized into $\hat{y}_{gi}$ and entropy encoded assuming a normal distribution with mean μ_{gi} scale σ_{gi}. The tensors μ_{gi} and σ_{gi} have the same dimensions as $\hat{y}_{gi}$ and are obtained as the output of a block of convolutional layers denominated the slice transform. The slice transform that computes μ_{gi} takes as input a concatenation of the tensor μ and all previously encoded slices $\tilde{y}_{gj} \forall j \in \{1, ..., i-1\}$. Note that $\tilde{y}_{gj} \neq \hat{y}_{gj}$, as explained in the next paragraph. Similarly, the slice transform that computes σ_{gj} takes the same inputs, except for μ_g, which is replaced by σ_g.

The channel-wise partition of the latent features y_g and entropy modeling based on previously encoded slice partitions allow to leverage redundancies between channels in order to reduce the bitrate. However, since the slice partitions have to be encoded and decoded sequentially, the capacity of the model to parallelize operations is limited. A trade-off therefore emerges between computation time and compression performance controlled by the number of slices, especially in situations when devices that enable for highly efficient parallel computing such as graphical processing units (GPU) are available.

Since the quantization of the latent features invariably induces an error $\hat{y}_{gi} - y_{gi}$, this model also attempts to compensate this error with the latent residual prediction. More specifically, a transform block outputs values between -0.5 and 0.5 by using $g(x) = 0.5 \tanh(x)$ as the activation function for its last layer. These values are then added to $\hat{y}_{gi}$ yielding $\tilde{y}_{gi}$.

Finally, in order to reconstruct the point cloud block, all slice partitions $\tilde{y}_{gi}$ are concatenated and fed to the synthesis block, where they are upsampled to the original dimension of $x_{\text{g-occ}}$ and reduced to only one channel. This reconstructed block $\hat{x}_{\text{g-prob}}$ contains the probability of occupancy for each voxel. In order to obtain a tensor with binary occupancy values $\hat{x}_{\text{g-occ}}$, these values are rounded using a threshold t. While it is possible to set a fix threshold equal to 0.5 for all blocks, this algorithm searches for the threshold that minimizes the point-to-point MSE (mean squared error) between the reconstructed and the input block is performed during encoding time. The optimal threshold is then encoded with the bitstream and used during decoding. More details on this geometry compression model can be found in its original paper.[21]

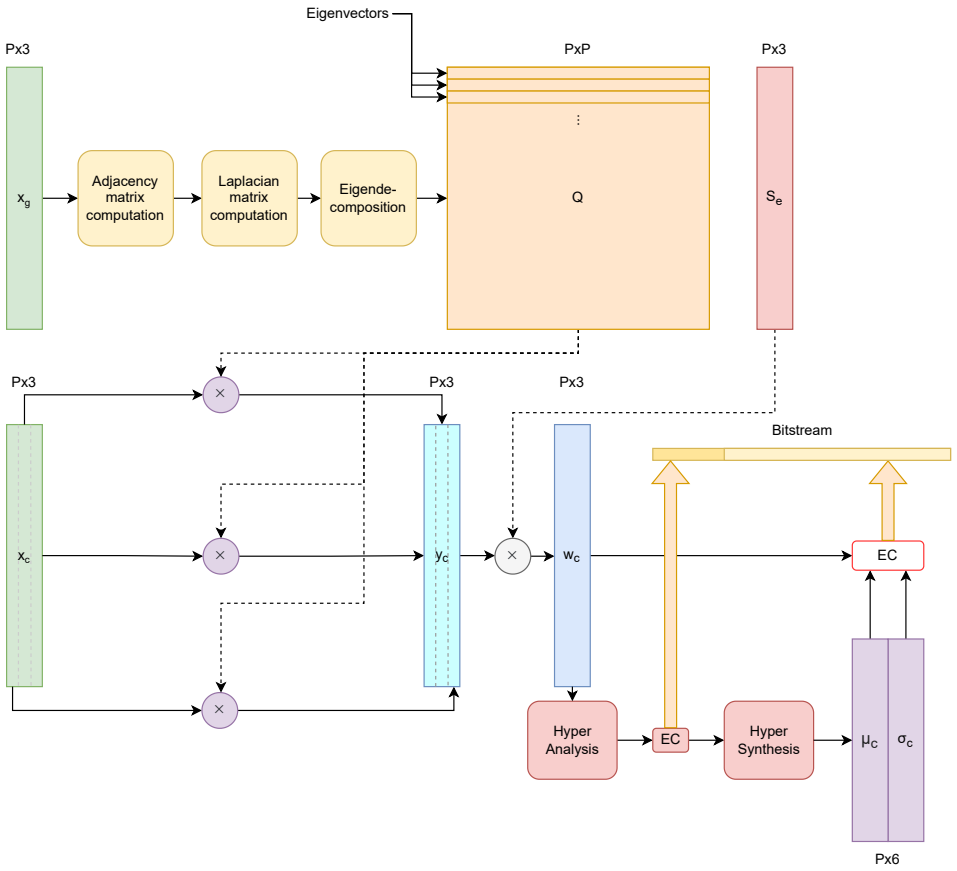


Figure 2: Block diagram of the color encoder.

3.3 Color coding

The color attributes of the point cloud are compressed using the GFT, which is applied to the color attributes generating coefficients that are then scaled, quantized and entropy coded using a learning-based hyperprior. Similarly to the geometry compression algorithm, the point cloud has to be partitioned into blocks prior to the encoding of the color. However, this partition process follows a process based on the K-d tree. First, the geometric axis (x, y or z) for which the point cloud has the largest variation is detected, which is determined by the difference between the minimum and maximum values of the geometric coordinates. The point cloud is then split in half, with the first portion containing the points with coordinates higher than the identified median value for the largest variation axis, and the second half containing the points with lower coordinate values for the same axis. These steps are recursively repeated for the two subsets of points generated after the splitting operation, and it is stopped when the number of points in all point cloud blocks is equal or lower than a threshold M. When all blocks are obtained, they are translated to have their geometric center at the origin and scaled to fit a bounding box of size 2. This block partition is similar to the algorithm described by Shao et al.[27]

While the obtained blocks contain both geometry and color matrices x_g and x_c, the geometry will be already available in the decoder side and only the color matrix x_c needs to be encoded. The color encoding process is represented in the block diagram of Figure 2. In order to apply the GFT to the color components, a graph is generated from the geometric coordinates x_g, being fully represented by its adjacency matrix A. In the proposed method, two points i and j are connected by an edge if the distance d_{ij} between them is smaller than τ.

The weight given to each edge is equal to $e^{-\frac{d_{ij}^2}{2\sigma^2}}$, with τ and σ^2 are assined values of 0.2 and 0.005. The weights are used to build the adjacency matrix A of the graph, where each element a_{ij} is equal to the weight of the edge connecting the point i to the point j. Since the graph is undirected, a_{ij} is equal to a_{ji} and A is a

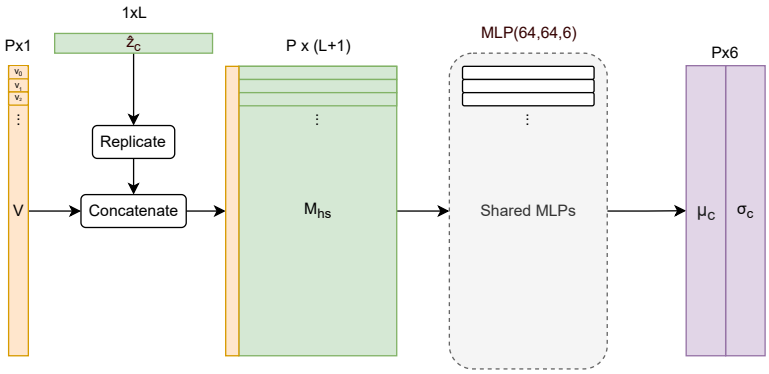


Figure 3: Block diagram of the hyper synthesis transform of the color compression method.

symmetric matrix of dimension $P \times P$, with P being the number of points in the block. The laplacian matrix L is then obtained by the equation $L = D - A$, with D being a diagonal matrix where each element $d_{ii} = \sum_{j=1}^{P} a_{ij}$. Finally, the eigendecomposition is applied to L in order to obtain its eigenvectors. Since L is symmetric and real, all of the eigenvalues of L are real and the eigenvectors form an orthonormal basis. This process results in the matrix Q where each column of Q contains an eigenvector of L.

The attribute vectors from x_c are represented in the YUV colorspace obtained following ITU-R Recommendation BT.709-6.[33] In order to generate the GFT coefficients, each vector is multiplied by the matrix Q. The obtained coefficient vectors are then rearranged into a matrix denominated y_c. Prior to quantization and entropy coding, the GFT coefficients are scaled through an element-wise product with a learned matrix S_e. The purpose of this scaling operation is to adapt the range of the coefficients prior to the quantization, which allows to implicitly define the error induced by quantization as well as the amount of distinct values after quantization, directly affecting the size of the compressed bitstream. The matrix S_e with size $P \times 3$ is obtained from a primitive matrix S with size $M/2 \times 3$, which is extended by replicating the last row $P - M/2$ times. The matrix S is learned during training, and is therefore able to learn from data a combination of scaling factors for each color component and for each coefficient index.

The resulting matrix $w_c = y_c \odot S_e$ goes through a learning-based entropy coding module that assumes a normal distribution with mean μ_c and scale σ_c estimated by a hyperprior, which is composed of a hyper analysis and a hyper synthesis transform. The hyper analysis takes as input w_c and produces a fixed-length codeword z_c with size L with a PointNet[34] network. Although for the most part the analysis transform follows the architecture described in the original PointNet paper, the last MLP (Multi-Layer-Perceptron) is only composed of two layers with L nodes. The obtained codeword z_c is then quantized into $\hat{z}_c$, entropy encoded following a learned fully factorized probability distribution, and added to the bitstream. The quantized codeword $\hat{z}_c$ serves as input to the hyper synthesis transform, which estimates the tensors μ_c and σ_c that are used during in the entropy coder. A block diagram representing the hyper synthesis is given in Figure 3.

In this transform, a column vector V with P elements is first created, the value of each element v_i being given by $e^{-\sigma_{hs}\sqrt{i}}$. In this expression, σ_{hs} is given by $\sigma_{hs} = -\ln(\epsilon)/\sqrt{P-1}$, where ϵ is the value of the last element of V, manually set to 0.01. Once V is obtained, the quantized codeword $\hat{z}_c$ is replicated vertically P times into a matrix of size $P \times L$ and concatenated to the vector V horizontally to generate M_{hs}. The goal of creating M_{hs} is to obtain a matrix where every row contains information of the GFT coefficients, represented by $\hat{z}_c$, and is also uniquely different from the others, ensured by the distinct values of v_i. The values assigned for v_i are designed to decrease with respect to i since the first GFT coefficients represent the lowest frequencies, and are therefore likely to be larger for smooth signals.[35] Each row of M_{hs} goes then through a Multi-Layer Perceptron with six nodes in the final layer, resulting in a $P \times 6$ matrix that models the probability distribution of the elements w_c. The first three columns of this matrix are used to represent the mean μ_c of the scaled GFT coefficients, while the last three columns represent their scales σ_c. These values are finally fed to the entropy bottleneck which quantizes and entropy codes w_c assuming a normal distribution with mean μ_c and scale σ_c.

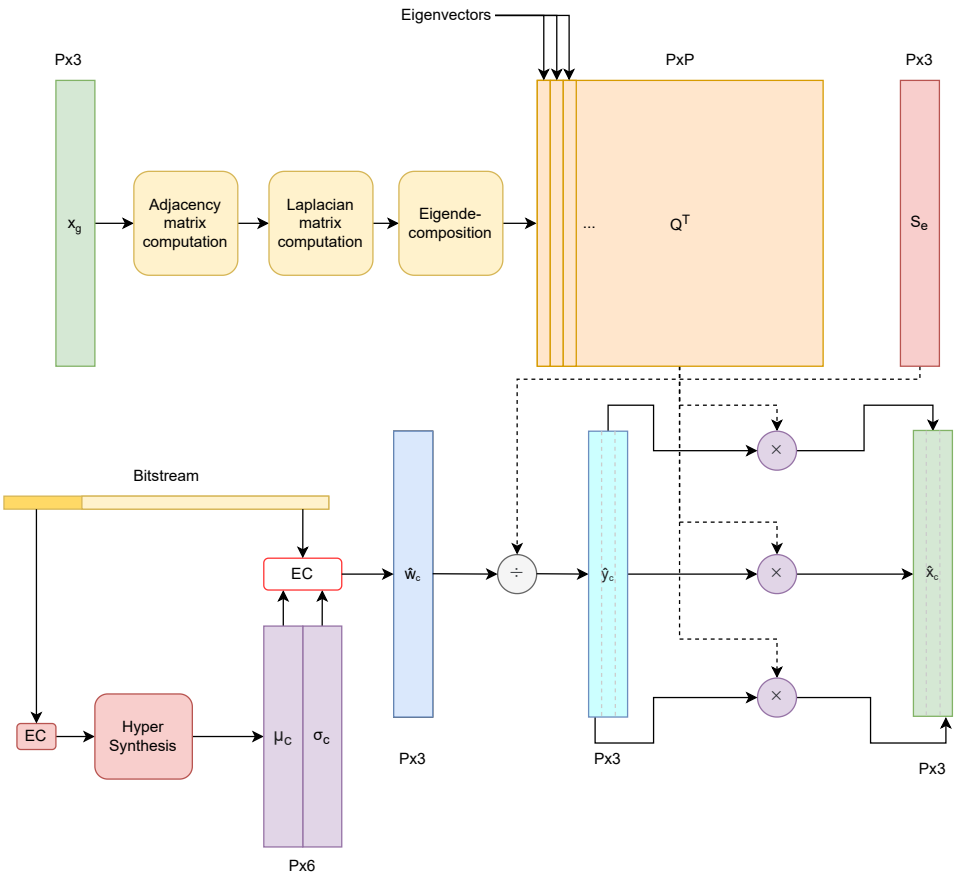


Figure 4: Block diagram of the color decoder.

The decoding process is initiated by the entropy decoding of the bitstream portion related to the codeword $\hat{z}_c$, following the factorized distribution shared with the decoder. The hyper synthesis transform is then applied to estimate mean μ_c and scale σ_c that are used to entropy decode the main portion of the bitstream. The obtained quantized scaled GFT coefficients $\hat{w}_c$ are divided element-wise by the extended scaling matrix S_e to generate $\hat{y}_c$. The decoded geometric coordinates are leveraged to obtain Q^T with the eigenvectors of the laplacian matrix contained in its columns. Each column of $\hat{y}_c$ is then multiplied by Q^T, and the reconstructed color values $\hat{x}_c$ in the YUV colorspace are finally obtained. The decoding process is illustrated in the Figure 4.

3.4 Training configuration

The geometry and color coding algorithm are based on different networks, and are therefore trained separately. For geometry compression, the algorithm was trained with blocks of size 64, extracted from a total of 44 point clouds. Blocks with less than 500 points were discarded, the training set being composed of a total of 13'084 blocks. During compression, the block resolution of 128 is adopted. The point clouds included in the training set were recruited from both the JPEG Pleno[36] and MPEG repositories, but also from other repositories such as PointXR,[37] VSENSE[38] and M-PCCD.[39]

The loss function used during training is defined as a trade-off between the rate of compressed bitstream and the distortion of the output point cloud, according to the Equation 1.

$$L_g = R(\hat{y}_g) + R(\hat{z}_g) + \lambda_g D(x_{\text{g-occ}}, \hat{x}_{\text{g-prob}}) \tag{1}$$

The hyperparameter λ_g controls the importance given to minimize the distortion when compared to the rate. $R(\hat{y}_g)$ and $R(\hat{z}_g)$ are given by an upper-bound estimation of the rate based on the entropy of $\hat{y}_g$ and $\hat{z}_g$, while

$D(x_{\text{g-occ}}, \hat{x}_{\text{g-prob}})$ is computed using the focal loss between the reconstructed and the input block. The focal loss value for each voxel is given by the Equation 2.

$$\text{FL}(x_j, \hat{x}_{gj}) = \begin{cases} -\alpha(1-\hat{x}_{gj})^{\gamma}\log(\hat{x}_{gj}), & \text{if } x_{gj}=1 \\ -(1-\alpha)\hat{x}_{gj}^{\gamma}\log(1-\hat{x}_{gj}), & \text{if } x_{gj}=0 \end{cases} \tag{2}$$

In the Equation 2, x_{gj} represents the binary occupancy value of voxel j, while $\hat{x}_{gj}$ stands for the predicted probability for that same voxel. α and γ are hyperparameter values used to tune the behaviour of the network according to the sparsity of the point cloud blocks. The model was trained with λ_g values of 40, 200, 400, 700, 1000 and 1750. Since the quantization operation has zero gradient almost everywhere, it cannot be used as such during backpropagation. Instead, added uniform noise was used during training as a proxy function only for the learning of the parameters for the entropy coding. For operations performed after entropy coding, a straight-through estimator was employed, i.e. replacing the gradient of the quantizer by an identity function. The model was trained using $\alpha = 0.6$ and $\gamma = 2$, and the Adam Optimizer was employed to update the weights of the model with learning-rate equal to 5×10^{-5} and batch size of 16. The epoch was defined with constant number of steps of 1000, and the early stopping callback function was used to detect model convergence and stop the training, with patience of 20 epochs. While the training for the model with the highest λ_g value was started with randomly initialized weights, the remaining were trained in sequential order, taking previous weights as a starting point. An Nvdia RTX 3090 GPU and an 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz CPU were employed for training.

The color compression algorithm was trained with point cloud blocks with maximum number of points M set to 512 during partition. A total of 50 point clouds were selected from the JPEG Pleno,[36] MPEG, PointXR,[37] VSENSE[38] and ShapeNet,[40] with 181'248 blocks being extracted from them. Only a set of randomly selected 10'000 blocks were actually employed during the training process. Similarly to the geometry coding method, a loss function defining a trade-off between rate and distortion is employed, as illustrated in Equation 3.

$$L_c = R(\hat{w}_c) + R(\hat{z}_c) + \lambda_c D(\hat{x}_c, x_c) \tag{3}$$

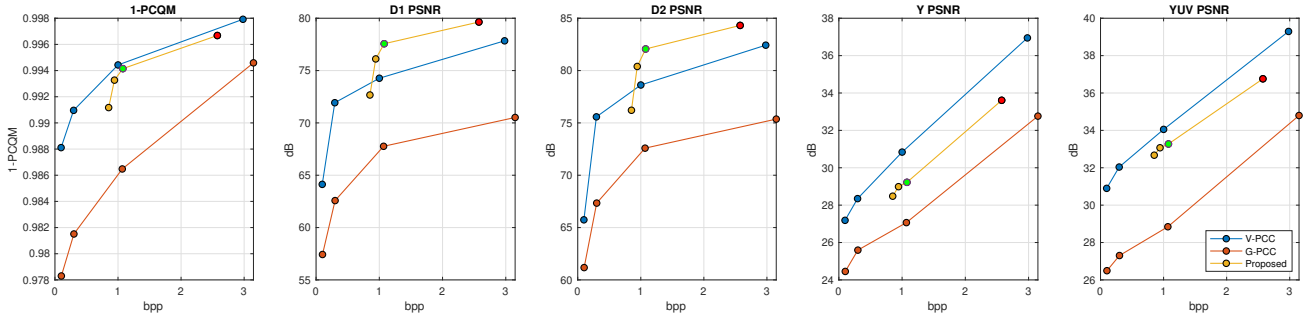
The terms $R(\hat{w}_c)$ and $R(\hat{z}_c)$ correspond to the estimated rate of both the scaled quantized GFT coefficients and the quantized codeword, while $D(\hat{x}_c, x_c)$ is measured by the mean squared error between the color values of the input and decompressed blocks. Since the human perception is highly correlated with distortions applied in the luminance channel, a weighted average between the three channels is adopted, according to the Equation 4.

$$D = \frac{6MSE_Y + MSE_U + MSE_V}{8} \tag{4}$$

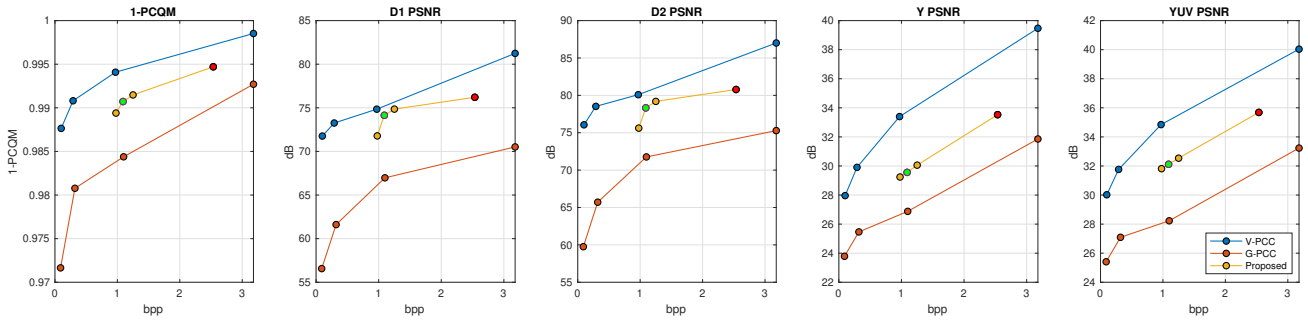
The model was trained with λ_c values of 100, 250, 500, 1000, 2500 and 5000. Since the quantization operation has zero gradient almost everywhere, it cannot be used as such during backpropagation. Instead, added uniform noise was used during training as a proxy function. The Adam Optimizer was employed with learning-rate equal to 10^{-4} and batch size of 16. The epoch was defined with constant number of steps of 1000, and the early stopping callback function was used to detect model convergence and stop the training, with patience of 10 epochs. The same CPU and GPU devices used for training of the geometry compression models were also employed for color compression.

4. RESULTS AND DISCUSSION

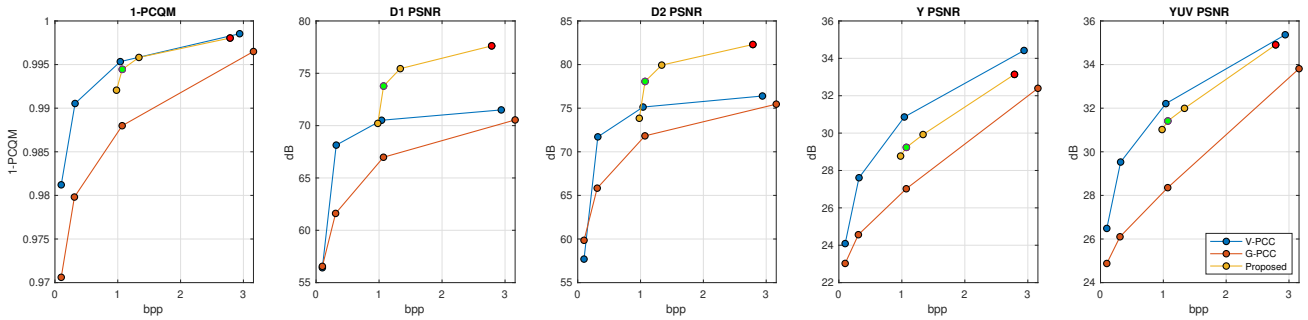
The integrated compression model was employed to compress and decompress point cloud models from the Real-World Textured Things dataset,[41] which have also been used for the recent JPEG Pleno Point Cloud Coding Call for Proposals.[42] Four point clouds used for the subjective experiment of the Call are selected here for evaluation. Since the geometry and color models can be chosen independently and different training parameters are used to achieve distinct bitrate values, the results depicted in this paper were generated with combinations of geometry and color models selected to minimize the PCQM metric for a given rate. These test models were



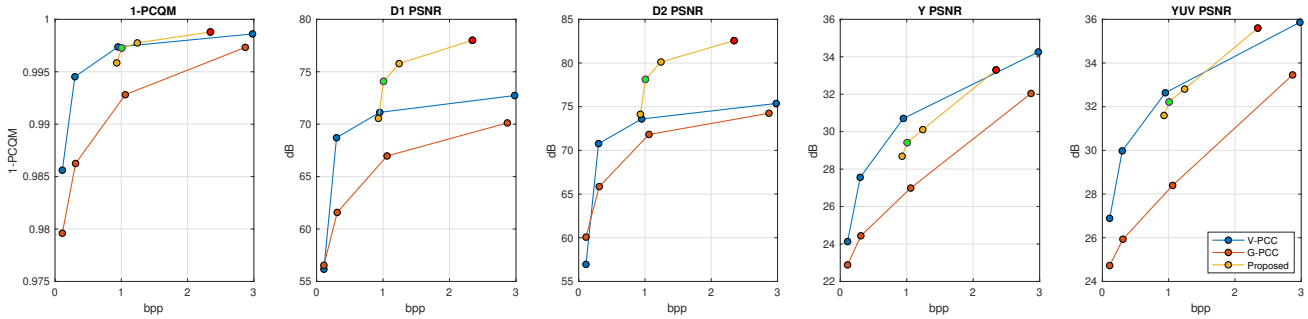
(a) *RWT136*



(b) *RWT246*



(c) *RWT395*



(d) *RWT503b*

Figure 5: Rate-distortion plots for four point clouds of the test dataset. In the curve of the proposed method, the points highlighted in green and in red correspond to R_A and R_B, respectively, as referenced in Figure 6.

also encoded with the MPEG compression standards V-PCC and G-PCC, according to the bitrates defined in the JPEG Pleno Common Training and Test Conditions.[43] The distorted point clouds were then evaluated with different objective quality metrics. The PSNR values of the point-to-point (D1) and point-to-plane (D2)[44] metrics computed between the distorted and original models were used to estimate the geometry distortion. The color attributes were also leveraged to compute the PSNR metric over the luminance channel, and a combination between the luminance and chrominance channels was also employed, according to the Equation 5.

$$\text{YUV PSNR} = \frac{6 \cdot \text{Y PSNR} + \text{U PSNR} + \text{V PSNR}}{8} \tag{5}$$

In order to compute the color metrics, the correspondence between points in the two evaluated point clouds is established by selecting the nearest neighbour. For all the above metrics, the symmetric value is obtained by setting both the original and the distorted model as reference, and selecting the lowest metric value between the two cases. Finally, PCQM[45] is selected as a joint metric for both geometry and color, mainly because previous studies observed its higher correlation with human perception than previous metrics.[46] Since PCQM computes the distance between two point clouds, the value of 1-PCQM is plotted as a monotonically increasing metric. The software for the first four metrics was obtained from the MPEG repository*, while the source code released by the authors of PCQM† was employed as is.

4.1 Objective results

The rate distortion plots for the four selected point clouds are shown in Figure 5. The objective quality evaluation reveals that the proposed method outperforms the anchor G-PCC across the entire evaluated test set for all employed metrics at the selected bitrates. However, the comparison with V-PCC reveals that the performance depends on the employed content and specially on the objective metric. The proposed method achieves better results when evaluated using the geometry-only metrics D1 and D2 PSNR for the majority of the point clouds with the exception of RWT246. These findings reflect the high performance achieved by the method proposed by Frank et al.,[21] which explored redundancies in the latent domain to achieve high compression ratios. However, similar results are not observed with the color metrics, for which the proposed method is still not able to achieve lower distortion levels than V-PCC. PCQM indicates that the performance of the proposed method and V-PCC are similar at higher bitrates, again with the exception of RWT246. Since this metric has been reported to achieve higher correlation with subjective perception,[46] this result suggest that the proposed method would perform similarly to V-PCC in a subjective experiment. The next subsection presents renderings of the distorted test set in order to evaluate this observation.

It must also be noted that the proposed method is not able to achieve bitrates as low as the anchors. This is mainly caused by the color bitstream, which consumes the majority of the bitrate for lower quality levels. Experiments showed that using lower λ values during training was not enough to achieve arbitrarily low rates. Instead, using bigger block sizes during compression or higher values for the quantization step could become viable alternatives. Moreover, compressing downsampled versions of the point cloud and performing upsampling as post-processing, similarly to the method proposed by Ruivo et al.,[47] could be another way of achieving higher compression rates for both geometry and color.

Considering that the performance of the proposed color compression method is lower than the geometry compression when comparing to V-PCC, it is reasonable to assume that future works should concentrate on that module in order to increase the overall performance of the integrated codec. Recent research on learning-based compression of both images and point clouds indicates that better rate distortion performance can be achieved by improving the efficiency of the entropy coder, which corresponds to having a predicted latent probability distribution that approximates the real distribution of the data. Since in the proposed architecture the GFT coefficients are modeled as a normal distribution, optimal compression efficiency is obtained when the mean μ is the closest to the coefficient values and when the scale σ is higher only for coefficients with greater uncertainty. During experimentation, it was observed that the values of μ and σ vary smoothly in relation to the index i,

*http://mpegx.int-evry.fr/software/MPEG/PCC/mpeg-pcc-dmetric/tree/master

†https://github.com/MEPP-team/PCQM

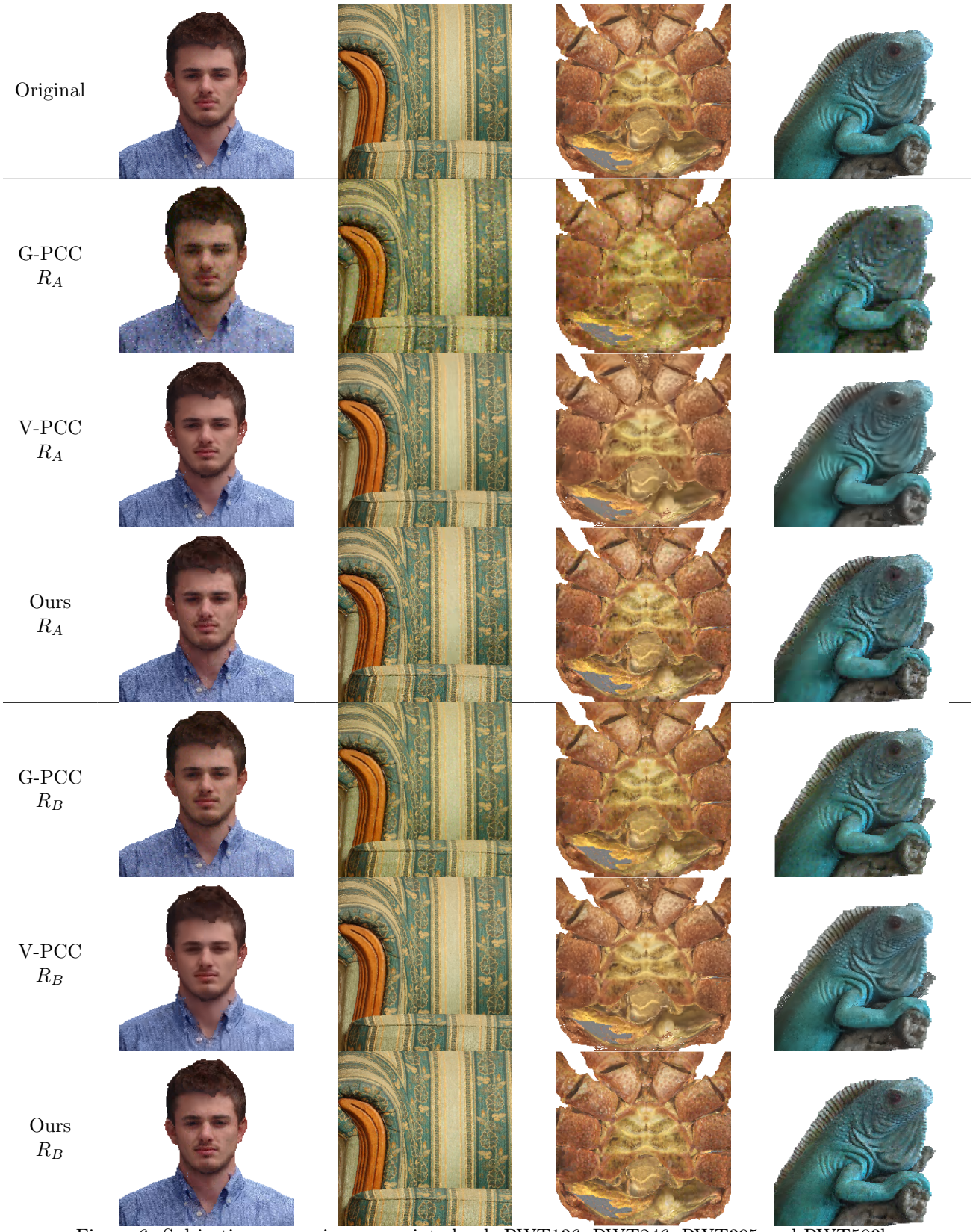


Figure 6: Subjective comparison on point clouds RWT136, RWT246, RWT395 and RWT503b.

similarly to the vector V. However, although the true GFT coefficients tend to have higher value for lower indices and decrease according to i, there can often be abrupt variations between two consecutive coefficients. The employed method of prediction of probability parameters seems therefore to present limitations, which could be exploited to achieve higher compression performance. Given that the employed hyper synthesis has considerably less learned parameters than the hyper analysis, future work could study increasing the complexity of the hyper synthesis transform in order to enable it to model more complex relations between the predicted coefficients. Moreover, the employed vector V has only one scalar element v_i for each GFT coefficient. As already suggested in a previous study,[48] using a seed with higher dimensionality, where each row v_i is a vector rather than one scalar value, could help improve the performance of the network.

The process that generates the adjacency matrix for the point cloud block could also be improved. In the current architecture, A is computed with handcrafted operations with two fixed parameters σ and τ, and the weight of the edge connecting two points depends only on the distance between them. Although experiments were conducted in order to implement learned transforms to generate A, the training process was found to be unstable. The most likely reason is the fact that backpropagation through the eigendecomposition operation is unstable when performed analytically.[49] Instead, replacing the gradients of the eigendeomposition by a proxy function[49] could stabilize training and allow to define learned transforms that capture more complex relations in the point cloud geometry to produce the underlying graph. Regardless, using only geometry information to define the graph will inevitably lead to assigning high weights to texture boundaries[30] in some cases. Defining a learning-based method to leverage color information in order to control the computation of the adjacency matrix and transmit it as side information could also potentially lead to an increase of the compression ratio.

4.2 Subjective results

A subjective evaluation of the performance of the proposed method in comparison to the anchors can be conducted by observing Figure 6. The distorted point clouds were selected from two quality ranges R_A and R_B. While for the anchors the two higher bitrates were selected, the employed rate values from the proposed method are highlighted in the plots of Figure 5. The presented renderings were generated with CloudCompare,[50] and the camera was zoomed in to observe the introduced artifacts with detail.

Careful observation demonstrates that G-PCC is outperformed by the proposed method and V-PCC at R_A, yielding noticeable color shifting and resolution loss. V-PCC tends to smooth the texture of the objects and may lose fine color detail at R_A, as well as cause small holes around intricate surfaces such as the ear of the model at RWT136 and the lower portion of the crab's shell at RWT395. Although it is hard to discern geometric distortions in the proposed method in the color, clear blocking artifacts are noticed, specially around texture-wise smooth areas as the face of RWT136 and the iguana's back at R503b. At the higher rate level R_B, these added distortions are less pronounced, but similar tendencies can be observed. In particular, G-PCC still induces some level of color shifting, which can be specially seen in the shades of green of RWT246 when compared to the other metdhos. The point clouds distorted with V-PCC still have a small number of missing points, but most of the detail is kept when compared to R_A. Finally, while the proposed method still adds blocking artifacts to the face of RWT136, the distortions are almost invisible in the remaining point clouds. The presented renderings suggest that the method achieves competitive subjective performance at similar bitrates as the anchors.

5. CONCLUSION

This paper presents an integrated learning-based point cloud compression method for both geometry and color. While the geometry coding method is based on an autoencoder with 3D convolutional layers, the proposed algorithm for color compression is based on the GFT and uses a mean and scale hyperprior to model the entropy of the obtained coefficients. The results show that the proposed method has competitive performance with the MPEG compression standards G-PCC and V-PCC. Future work could aim at allowing the integrated codec to achieve lower bitrates and further improve its rate-distortion performance, specially for color coding. For the first task, applying downsampling prior to compression and upsampling as a post-processing step, similarly to what is proposed by Ruivo et al.[47] for the geometric coordinates, could be a viable solution. Since the most part of the bitstream is associated with color information, an adaptation of the previous method to our color coding method could be envisaged. In order to improve the compression ratio while maintaing visual quality, studies could also

be conducted by adding complexity to hyper synthesis transform as well as adapting the eigendecomposition gradient to allow for learning-based transforms to define the adjacency matrix of the underlying graph.

ACKNOWLEDGMENTS

This work was supported by the Swiss National Foundation for Scientific Research (SNSF) under the grant number 200021-178854.

We also thank Stuart Perry (University of Technology of Sidney), Luis Cruz (University of Coimbra) and João Prazeres (University of Beira Interior) for providing the test set encoded with the anchors G-PCC and V-PCC as used in the JPEG Pleno Point Cloud Call for Proposals.

REFERENCES

[1] Isik, B., Chou, P. A., Hwang, S. J., Johnston, N., and Toderici, G., "Lvac: Learned volumetric attribute compression for point clouds using coordinate based networks," *arXiv preprint arXiv:2111.08988* (2021).

[2] Wang, J. and Ma, Z., "Sparse tensor-based point cloud attribute compression," *arXiv preprint arXiv:2204.01023* (2022).

[3] Mekuria, R., Blom, K., and Cesar, P., "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Transactions on Circuits and Systems for Video Technology* **27**, 828–842 (Apr. 2017).

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

[5] MPEG 3D Graphics Coding, "Text of ISO/IEC CD 23090-5 Visual Volumetric Video-based Coding and Video-based Point Cloud Compression 2nd Edition." ISO/IEC JTC1/SC29/WG07 Doc. N0003 (Nov. 2020).

[6] MPEG Systems, "Text of ISO/IEC DIS 23090-18 Carriage of Geometry-based Point Cloud Compression Data." ISO/IEC JTC1/SC29/WG03 Doc. N0075 (Nov. 2020).

[7] Alexiou, E., Tung, K., and Ebrahimi, T., "Towards neural network approaches for point cloud compression," in [*Applications of Digital Image Processing XLIII*], 4 (08 2020).

[8] Guarda, A. F., Rodrigues, N. M., Ruivo, M., Coelho, L., Seleem, A., and Pereira, F., "It/ist/ipleiria response to the call for proposals on jpeg pleno point cloud coding," *arXiv preprint arXiv:2208.02716* (2022).

[9] Schnabel, R. and Klein, R., "Octree-based point-cloud compression," in [*Symposium on Point-Based Graphics 2006*], (Jul. 2006).

[10] Huang, Y., Peng, J., Kuo, C. . J., and Gopi, M., "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics* **14**(2), 440–453 (2008).

[11] Rusu, R. B. and Cousins, S., "3d is here: Point cloud library (pcl)," in [*2011 IEEE International Conference on Robotics and Automation*], 1–4 (2011).

[12] Pavez, E., Chou, P. A., de Queiroz, R. L., and Ortega, A., "Dynamic polygon clouds: representation and compression for VR/AR," *APSIPA Transactions on Signal and Information Processing* **7**, e15 (2018).

[13] Mammou, K., Tourapis, A. M., Singer, D., and Su, Y., "Video-based and hierarchical approaches point cloud compression." ISO/IEC JTC1/SC29/WG11 Doc. M41649 (Oct. 2017).

[14] Quach, M., Valenzise, G., and Dufaux, F., "Learning convolutional transforms for lossy point cloud geometry compression," in [*2019 IEEE International Conference on Image Processing (ICIP)*], 4320–4324 (2019).

[15] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., "Deep learning-based point cloud coding: A behavior and performance study," in [*2019 8th European Workshop on Visual Information Processing (EUVIP)*], 34–39 (2019).

[16] Ballé, J., Laparra, V., and Simoncelli, E. P., "End-to-end optimized image compression," in [*5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*], OpenReview.net (2017).

[17] Quach, M., Valenzise, G., and Dufaux, F., "Improved Deep Point Cloud Geometry Compression," in [*IEEE International Workshop on Multimedia Signal Processing (MMSP'2020)*], (Sep. 2020).

[18] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N., "Variational image compression with a scale hyperprior," in [*6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*], OpenReview.net (2018).

[19] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing* **15**(2), 415–430 (2021).

[20] Guarda, A., Rodrigues, N., and Pereira, F., "Neighborhood adaptive loss function for deep learning-based point cloud coding with implicit and explicit quantization," *IEEE MultiMedia* , 1–1 (2020).

[21] Frank, N., Lazzarotto, D., and Ebrahimi, T., "Latent space slicing for enhanced entropy modeling in learning-based point cloud geometry compression," in [*2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], (2022).

[22] Lazzarotto, D., Alexiou, E., and Ebrahimi, T., "On block prediction for learning-based point cloud compression," in [*2021 IEEE International Conference on Image Processing (ICIP)*], (2021).

[23] Lazzarotto, D. and Ebrahimi, T., "Learning residual coding for point clouds," in [*Applications of Digital Image Processing XLIV*], **11842**, 223–235, SPIE (2021).

[24] Wang, J., Ding, D., Li, Z., and Ma, Z., "Multiscale point cloud geometry compression," in [*2021 Data Compression Conference (DCC)*], 73–82, IEEE (2021).

[25] Wang, J., Ding, D., Li, Z., Feng, X., Cao, C., and Ma, Z., "Sparse tensor-based multiscale representation for point cloud geometry compression," *arXiv preprint arXiv:2111.10633* (2021).

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

[27] Shao, Y., Zhang, Z., Li, Z., Fan, K., and Li, G., "Attribute compression of 3d point clouds using laplacian sparsity optimized graph transform," in [*2017 IEEE Visual Communications and Image Processing (VCIP)*], (10 2017).

[28] Shao, Y., Zhang, Q., Li, G., Li, Z., and Li, L., "Hybrid point cloud attribute compression using slice-based layered structure and block-based intra prediction," in [*Proceedings of the 26th ACM international conference on Multimedia*], 1199–1207 (2018).

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

[30] Song, F., Li, G., Gao, W., and Li, T. H., "Rate-distortion optimized graph for point cloud attribute coding," *IEEE Signal Processing Letters* **29**, 922–926 (2022).

[31] de Queiroz, R. L. and Chou, P. A., "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing* **25**(8), 3947–3956 (2016).

[32] Quach, M., Valenzise, G., and Dufaux, F., "Folding-based compression of point cloud attributes," in [*2020 IEEE International Conference on Image Processing (ICIP)*], 3309–3313 (2020).

[33] ITU-R BT.709-6, "Parameter values for the HDTV standards for production and international programme exchange." International Telecommunication Unionn (Jun. 2015).

[34] Qi, C. R., Su, H., Mo, K., and Guibas, L. J., "Pointnet: Deep learning on point sets for 3d classification and segmentation," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 652–660 (2017).

[35] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P., "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine* **30**(3), 83–98 (2013).

[36] "JPEG Pleno Database, https://jpeg.org/plenodb/.,"

[37] Alexiou, E., Yang, N., and Ebrahimi, T., "Pointxr: A toolbox for visualization and subjective evaluation of point clouds in virtual reality," in [*2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*], 1–6, IEEE (2020).

[38] Zerman, E., Gao, P., Ozcinar, C., and Smolic, A., "Subjective and objective quality assessment for volumetric video compression," *Electronic Imaging* **2019**(10), 323–1 (2019).

[39] Alexiou, E., Viola, I., Borges, T. M., Fonseca, T. A., De Queiroz, R. L., and Ebrahimi, T., "A comprehensive study of the rate-distortion performance in mpeg point cloud compression," *APSIPA Transactions on Signal and Information Processing* **8** (2019).

[40] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012* (2015).

[41] Maggiordomo, A., Ponchio, F., Cignoni, P., and Tarini, M., "Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools," *Computer Aided Geometric Design* **83**, 101943 (2020).

[42] WG1, "Final Call for Proposals on JPEG Pleno Point Cloud Coding." ISO/IEC JTC1/SC29/WG1 Doc. N100097 (Jan 2022).

[43] Cruz, L., "JPEG Pleno Point Cloud Coding Common Training and Test Conditions v1.3 ." ISO/IEC JTC1/SC29/WG1 Doc. N100276 (July 2022).

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

[45] Meynet, G., Nehmé, Y., Digne, J., and Lavoué, G., "PCQM: a full-reference quality metric for colored 3d point clouds," in [*2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*], 1–6 (2020).

[46] Lazzarotto, D., Alexiou, E., and Ebrahimi, T., "Benchmarking of objective quality metrics for point cloud compression," in [*2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*], 1–6, IEEE (2021).

[47] Ruivo, M., Guarda, A., and Pereira, F., "Double deep learning-based point cloud geometry coding with adaptive super-resolution," in [*2022 10th European Workshop on Visual Information Processing (EUVIP)*], (Accepted in 2022).

[48] Yang, J., Ahn, P., Kim, D., Lee, H., and Kim, J., "Progressive seed generation auto-encoder for unsupervised point cloud learning," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision*], 6413–6422 (2021).

[49] Wang, W., Dang, Z., Hu, Y., Fua, P., and Salzmann, M., "Backpropagation-friendly eigendecomposition," *Advances in Neural Information Processing Systems* **32** (2019).

[50] "Cloudcompare (version 2.11) [gpl software]," (2022).