Towards neural network approaches for point cloud compression

Evangelos Alexiou*, Kuan Tung*, and Touradj Ebrahimi

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

ABSTRACT

Point cloud imaging has emerged as an efficient and popular solution to represent immersive visual information. However, the large volume of data generated in the acquisition process reveals the need of efficient compression solutions in order to store and transmit such contents. Several standardization committees are in the process of finalizing efficient compression schemes to cope with the large volume of information that point clouds require. At the same time, recent efforts on learning-based compression approaches have been shown to exhibit good performance in the coding of conventional image and video contents. It is currently an open question how learning-based coding performs when applied to point cloud data. In this study, we extend recent efforts on the matter by exploring neural network implementations for separate, or joint compression of geometric and textural information from point cloud contents. Two alternative architectures are presented and compared; that is, a unified model that learns to encode point clouds in a holistic way, allowing fine-tuning for quality preservation per attribute, and a second paradigm consisting of two cascading networks that are trained separately to encode geometry and color, individually. A baseline configuration from the best-performing option is compared to the MPEG anchor, showing better performance for geometry and competitive performance for color encoding at low bit-rates. Moreover, the impact of a series of parameters is examined on the network performance, such as the selection of input block resolution for training and testing, the color space, and the loss functions. Results provide guidelines for future efforts in learning-based point cloud compression.

Keywords: point cloud, compression, deep-learning, auto-encoder

1. INTRODUCTION

The increase of depth sensors availability that nowadays equip high-performing handheld devices, as well as the current development of augmented reality (AR) and virtual reality (VR) applications, indicate the increasing demands for flexible 3D content representations that can be efficiently captured, encoded, processed and displayed. Among the solutions, point cloud imaging has lately attracted a strong interest. The past years, the JPEG and MPEG standardization bodies have initiated activities focused on this visual data modality, triggering notable advances in compression technologies. As a result of such efforts, MPEG has developed the first point cloud compression standard,[1] which will facilitate interoperability across devices, and is expected to further assist the integration of this technology in several daily use-cases.

A point cloud can be defined as a set of points that span in the 3D space, defined by their x, y and z coordinates. The geometric information is typically accompanied by attributes that are associated with each point, such as color, normal vectors, curvature values, reflectance, etc. In essence, a point cloud can be considered as an organized or unorganized data structure, which can be obtained from regular or irregular sampling of the external surface of a 3D model. The coordinates indicate the spatial position of each sample, while the associated attributes provide information that better describes the shape (e.g., normal vectors, curvature values), or the texture (e.g., color, reflectivity) of the underlying continuous surface. Using discrete samples to represent a 3D model offers high levels of flexibility, while providing the potential of per-point adjustments. Such qualities,

*Both authors contributed equally to this work.
Further author information: (Send correspondence to authors)
E-mail: firstname.lastname@epfl.ch

though, come at a cost of a vast amount of information that is required in order to faithfully represent a 3D model. Thus, it becomes clear that efficient compression solutions are inevitably needed.

In static point cloud compression, there are different approaches aiming at reducing the data size of geometric or textural information. Notably, the most popular solutions employ tree data structures, graphs, or patches of projected views of a model. The first rely on data structures that can efficiently organize the spatial placement of the points, such as KD-trees and Octrees; the second employ graph arrangements to represent a model with nodes indicating a point, or a neighborhood; the latter approaches are based on plane projections of a model that are typically obtained from different perspectives and can be encoded using conventional 2D imaging compression solutions. Lately, auto-encoding neural network architectures have been proposed to encode point cloud geometry, extending similar efforts that have preceded in 2D imaging. Despite the fact that this type of point cloud coding is still at its infancy, the results are very promising, with the current solutions competing, if not out-performing, state-of-the-art algorithms.

Inspired by the great potentials that neural networks show in learning transforms for compressing visual data representations, in this study we extend previous efforts by learning geometry and color attributes of point cloud models. In particular, we initiate by extending a publicly available geometry-only point cloud auto-encoding solution in learning transforms for a holistic data representation including both geometry and color. We analyse the performance of this unified network, using widely employed objective quality metrics that focus on geometric and color degradations. Moreover, we examine the impact of assigning various weights to geometry and color distortion terms in the loss function, to understand whether an optimal weighting scheme can be found. The performance of this model is compared to a different architecture that is composed of two separately trained networks dedicated to geometry and color. Furthermore, the proposed model is benchmarked against a widely-used coding solution, which denotes the anchor in the recent point cloud compression-related efforts of the MPEG standardization body. A set of meta-analysis studies is also reported, carried to understanding the impact of dataset, color space, and loss function selection, among others, in the network performance. Results demonstrate that the adopted architecture is able to perform competitively with respect to well-established solutions for point cloud compression, both in the geometry and color domain, especially at low bitrates.

2. RELATED WORK

The prevalent approaches that have been widely explored in the literature for point cloud compression can be clustered as model-based and projection-based. The first class can be further sub-divided to geometry and attribute compression, with attribute encoding typically applied on the resulting encoded shape. Geometry codecs rely on efficient data structures and, most commonly Octrees[2] which allow regular grid representations indicating points position through occupancy maps. Octree-based compression was initially introduced in,[3] with a progressive encoding extension described in.[4] After Octree decomposition, denser approximations can be achieved by reconstructing the underlying surface of a model through a “Triangle Soup” (TriSoup) as described in,[5] planar surface models as proposed in[6] while also graph-based enhancements and volumetric functions can be employed as in[7] and,[8] respectively.

Color attribute encoding using Graph Fourier Transform (GFT) was initially presented in[9] and further extended in,[10] by enabling Laplacian sparsity. In,[11] the Gaussian Process Transform (GPT) is employed to exploit geometry correlations, while the Region Adaptive Hierarchical Transform (RAHT) based on the Haar wavelet transform is introduced in[12] offering a high-performance solution with significant complexity reductions. The above algorithms can be clustered as transform-based. Prediction-based solutions include a 3-D intra prediction scheme that is based on neighboring blocks and is described in,[13] while in,[14] a hierarchical structure is proposed with points belonging to a lower layer being used to predict attributes at a higher layer of details.

Projection-based compression solutions exploit the high performance of 2D imaging codecs, applied on projected views of point clouds. In,[15] the JPEG coding engine is used to compress the colour of points that were projected in a depth-first tree traversal order. In,[14] a patch-based point cloud projection on planar surfaces is proposed, where the patches are assembled in a video sequence. This work essentially established the basis of the emerging MPEG Video-based Point Cloud Compression (V-PCC) test model.[16] The latter employs HEVC to encode the two video sequences that are generated to capture geometry and texture information of a point cloud.

Additional metadata to reconstruct the model are compressed separately. For a recent review and taxonomy of compression algorithms, the interested reader can refer to.[17]

Recently, deep learning architectures dedicated to compression of visual data representations have been proposed, showing promising results. The success and efficiency that has been observed in 2D imaging modalities has driven the interest for extending data driven approaches in point cloud imaging, which denotes a higher-dimensionality and irregular content representation. Early works on image-based solutions employ recurrent neural networks enabling iterative residual encoding[18] to achieve higher quality. In[19] and,[20] end-to-end network architectures that jointly optimize rate and distortion introducing differentiable approximations for quantization and entropy rate estimation are proposed. The latter work is extended in[21] by introducing Variational Auto-Encoding (VAE) to improve the entropy coding performance. In,[22] the application of Principal Component Analysis (PCA) on the feature maps obtained after the synthesis stage is proposed in order to exploit redundancies in the latent representation, showing performance improvements.

Regarding deep-learning approaches on point cloud imaging, the majority of current auto-encoding architectures target compressing geometry-only information. In particular, one of the first attempts is reported in,[23] proposing a rather simple, yet efficient architecture composed of convolution and de-convolution layers for analysis and synthesis, respectively. In a more recent work,[24] the impact of several parameters added to the initial network version[23] is evaluated through a series of experiments. Among the additions, a hyper-prior model and deeper transforms, as well as fine tuning of the loss function and adaptive thresholding, were found to improve the performance. Another early study on the field is presented in,[25] which also adopts a small number of convolution and de-convolution layers for analysis and synthesis. Performance evaluation results show that a larger number of filters per layer is only beneficial at larger bitrates. The same authors extend their efforts in[26] and conduct rate-distortion performance analysis on the latent space using the same network.[25] In,[27] the network architecture is enriched with a hyper-prior and the possibility of explicit quantization via down-scaling and up-scaling, before and after feeding the latent representation to the VAE module, respectively.

In,[28] a deeper auto-encoding architecture is proposed, based on 3D convolution layers stacked with Voxception-ResNet structures and a hyper-prior implemented as a VAE. Several pre-processing steps are employed including voxelization, scaling and partition before feeding a model block-by-block to the network, similarly to.[25] The performance of this network shows promising results, achieving comparable, if not better performance when compared to V-PCC. Experimentation with different partition sizes and adaptive thresholding for classification of a voxel as occupied or not, are part of the study. In,[29] a multi-scale hierarchical encoder is proposed based on local features that are extracted at each layer. The aforementioned studies are handling point clouds as 3D occupancy maps on regular grids. In,[30] raw point clouds are fed to the proposed architecture, which makes use of the PointNet[31] to extract features from unorganized coordinates in 3D space, while the synthesis transform is represented by a generative fully-connected network.

To the best of our knowledge, there is only one study focused on compression of point cloud attributes, described in,[32] which is based on folding a 2D grid onto a point cloud and then mapping the attributes on top of it. An advantage of this approach is the application of highly efficient 2D imaging techniques for point cloud compression; yet, a bottleneck is the low accuracy of the folding in geometrically complex parts of a model. In our study, we handle geometry and/or color in the 3D domain by extracting features from regular grids making use of 3D convolutions, which enable capturing of spatial redundancies for both types of information. The study aims to provide useful insights for future references focused on the matter.

3. EXPERIMENTAL SETUP

3.1 Data set

For the purposes of this study, a selection of high resolution point clouds from several repositories was pursued in order to form a collection of training and testing models with diverse characteristics in geometry and color. In particular, a total of 50 models were selected from the MPEG[*], JPEG Pleno[†], PointXR,[33] VSENSE,[34] and M-PCCD[35] datasets, forming the so-called High Resolution Geometry and Color (HighResGC) dataset. The

[*]http://mpegfs.int-evry.fr/MPEG/PCC/DataSets/pointCloud/CfP/
[†]https://jpeg.org/plenodb/

(a) amphoriskos (b) andrew (c) biplane (d) egyptianmask (e) matis (f) nefertiti
(g) queen (h) redandblack (i) thaidancer (j) tiki (k) ulliwegner (l) zeus

Figure 1: Sample models used for training.

(a) bumbameuboi (b) guanyin (c) longdress (d) phil (e) rhetorician (f) romanoillamp

Figure 2: Models used for testing.

JPEG Pleno and MPEG repositories consist of colored models that were assembled in the context of relevant standardization activities, containing representative sets of real-life acquired and synthetic point clouds that span across a variety of categories, such as, inanimate models, cultural heritage, human bodies, etc. The PointXR data set[33] consists of low-noise, high quality point clouds that represent cultural heritage models, obtained after conversion from their original mesh content representations. The VSENSE data set[34] consists of two dynamic sequences of human bodies, thus, including several frames of the same figures at different poses. From this repository, only a representative, low-noise frame was selected per sequence. Finally, a content coming with the M-PCCD data set[35] was recruited, to further enhance the data.

The majority of the models were voxelized at a bit depth of 10, independently of their original content representation (i.e., raw or voxelized at a higher grid resolution). Sparser point clouds were voxelized at a bit depth of 9, as in,[25] while models with geometry originally lying at a grid of lower resolution (e.g., 9), remained as such (e.g., Microsoft Upper Bodies). Moreover, the color attributes were normalized in a range between 0 and 1. Following previous efforts,[25–28] the collected point clouds were partitioned into blocks, with the latter denoting the input data that are fed into the network. This approach mimics the block partition in classical image compression and is coming with two main advantages; that is, lower computational demands in handling input units, and data augmentation, provided that every block is interpreted as an independent sample. However, spatial redundancies cannot be largely exploited when blocks of low resolution are selected. Note that a higher performance is expected when considering larger block sizes[28] (see Section 5.1).

3.1.1 Training data

The training data consists of the entire set of point clouds that were collected, excluding 6 models that comprise our testing set. A part of the selected models is illustrated in Figure 1. The training models were partitioned into non-overlapping blocks of size K, with $K = 32$ or 64 depending on the task at hand, with each block being handled independently in our network. Following,[25] blocks that contain less than 500 occupied points were discarded, as they carry limited relevant information. From the remaining blocks, a total of 10'000 samples were randomly picked to form our training set.

3.1.2 Testing data

The testing data consists of the models that have been specified in the Common Test Conditions document authored by the JPEG standardization committee as a result of its latest efforts.[36] The employed models denote a representative set of inanimate objects and human figures with a relatively wide range of geometric arrangement and color distribution, as illustrated in Figure 2. For the testing data, block sizes of $K = 128$ are used, except if otherwise mentioned. Note that it is a rather common approach[23] to use different resolutions for training and testing blocks, whose influence is investigated in Section 5.2.

3.2 Network architecture

3.2.1 Input

The geometry and texture of every input unit is initially provided in a typical format, which resembles a 6-tuple list, with each entry denoting a point that is defined by its x, y and z coordinates followed by the r, g and b color values. The point cloud data are already voxelized, thus, the original format can be easily converted to a 3D voxel grid. This data representation allows us to exploit 3D convolution kernels to capture spatial redundancies in the output feature maps. The 3D voxel grid is then partitioned into blocks of a specified resolution, and each block is associated with a number of input channels that carry topological and potentially textural information, depending on the task. In particular, the blocks are of resolution $K \times K \times K \times \hat{C}$, with $\hat{C} = 1$ for geometry-only compression and $\hat{C} = 4$ for color-only or geometry-plus-color encoding. In all cases, the first channel contains values of 0 or 1 to indicate occupied voxels. The optionally enabled additional color channels contain values between 0 and 1, obtained after a scaling step.

3.2.2 Auto-encoder

The network architecture adopts as a baseline the model proposed in.[23] As the majority of the current auto-encoding solutions, the processing pipeline can be decomposed in three main parts; that is, an analysis stage consisting of convolution layers, a synthesis stage that is composed of de-convolution layers, and a bottleneck in the middle that corresponds to the latent representation. Our selection for this baseline is motivated by the fact that it denotes a publicly available, efficient implementation of an end-to-end auto-encoder with good performance on geometry compression. Moreover, similar core architectures have been employed in 2D image-based paradigms, revealing high-performance in terms of compression efficiency.

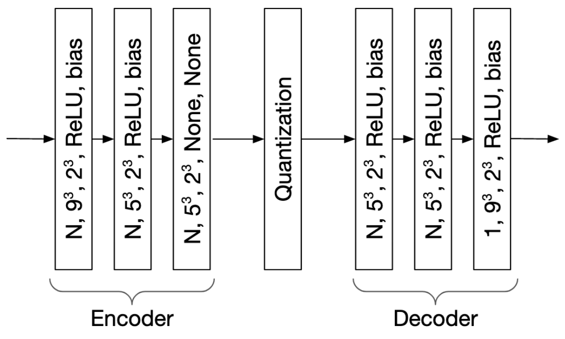


Figure 3: The *base model*'s structure.[28]

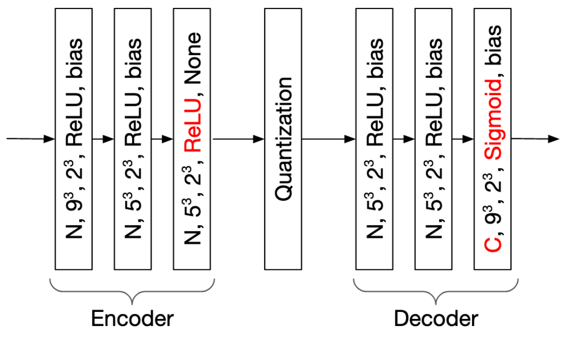


Figure 4: The *unified model*'s structure.

The base model can encode only point cloud geometry.[23] It is composed of three 3D convolution layers at the encoder and their symmetric transposed convolution (i.e., de-convolution) counterparts at the decoder side, as illustrated in Figure 3. The first term of each block of the diagram denotes the number of filters (i.e., N), the second term denotes the size of the filters (i.e., 9^3), the third term is the size of the stride (i.e., 2^3), the forth term is the type of activation function, and the fifth term is whether or not bias is applied.

At the encoding stage, the point cloud is partitioned into 3D blocks, as described in Section 3.1.1. A selection of stride size higher than 1 implies down-sampling of the input representation. In our case, a stride size of 2^3, denotes down-sampling of the input unit by a factor of 0.5 in each dimension at the output of each layer. Quantization is applied on the latent representation, which is obtained at the output of the encoder. During training, the quantization is replaced by additive uniform noise,[20] in order to ensure that the gradient is defined for the back-propagation operation. Moreover, the rate is estimated using differential entropies,[20] provided that the values at the output of the quantization step are continuous. During testing, the floating point latent representation is quantized with trained probability tables, and the bitstream is obtained by entropy coding. At the decoding stage, the bitstream is received and passes from a set of de-convolution layers with stride size equal to 2^3, which implies up-sampling by a factor of 2. Through a series of symmetric de-convolution layers, the compact feature maps are decoded and the point cloud geometry can be recovered in the form of 3D blocks. A loss function is employed to quantify the reconstruction distortion and train the model in an end-to-end manner performing joint optimization of both rate and distortion. For this purpose, a multiplier is employed to steer the trade-off at will. In particular, the loss is composed of this multiplier (weight term), λ_g, a distortion term, D_g, and a rate term, R that represents bits per input occupied voxel (bpp) as follows:

$$L = R + \lambda_g D_g \tag{1}$$

Note that by modifying the λ_g term, the bitrates and the reconstructed quality can be tuned; that is, by setting a higher weight, the model will focus more on learning how to preserve geometry information and less on compressing, thus, resulting in higher reconstruction quality at the expense of higher bitrate. The distortion term is computed by comparing the original with the recovered point cloud. This task can be interpreted as a binary classification problem, hence, the focal loss is employed to assess the reconstruction error, defined as in[37] and given in Equation 2

$$\begin{aligned} FL(p_z^t) &= -\alpha_z (1 - p_z^t)^\gamma log(p_z^t), \\ FL(x) &= \sum_{z \in S} FL(p_z^t), \end{aligned} \tag{2}$$

where p_z^t is defined as p_z if the voxel is occupied and $1 - p_z$ if the voxel is unoccupied, p_z is the output value of the voxel indicating probability of whether the voxel is occupied or not. α_z is defined as α if the voxel is occupied and $1 - \alpha$ otherwise.

The unified model is able to encode point cloud geometry and/or color attributes. It follows the same architecture as the *base model*, with some necessary modifications to support the enhanced functionality, as illustrated in Figure 4. In this diagram, red color is used to highlight differences with respect to the original version. Specifically, the number of channels for the last layer of the decoder, C, is set to either 1, 3 or 4 depending on the task. For geometry compression, C is equal to 1, for color compression C equals 3, while for geometry-plus-color compression, C is equal to 4. Notice that a ReLU activation function is added at the final layer of the encoder, while at the final layer of the decoder, the activation function is switched to sigmoid in order to ensure that the output values lie in the range $[0, 1]$.

To train the network for point cloud geometry compression, we employ a slight variation of the loss function defined in[23] and provided in Equation 1. In particular, the distortion term is normalized by dividing with the total number of voxels, such that it represents a measurement of distortion per voxel. To train the network for point cloud color compression, a similar formulation is adopted. In this case, the focal loss is replaced by a simple l_2 norm, which is computed between the original and the reconstructed color values across the occupied voxels of the input block. The color loss is normalized by dividing with the number of occupied voxels of a block to reflect the distortion per occupied voxel. Note that both geometry and color distortion terms are normalized by the number of points that effectively contribute to the loss. For color degradation, a logarithmic function of the l_2 norm is computed to obtain scores in the same range with the geometry term. In Equation 3, the updated loss function used for for color-only compression is provided.

$$L = R + \lambda_c D_c \tag{3}$$

To train the network for point cloud geometry-plus-color compression, both metrics are employed and both distortion terms are included in the loss function, as indicated in Equation 4. Notice that the overall quality of the restored model as well as a different quality preservation scheme can be enabled for the two attribute types by selecting different λ values. Note that subscripts g and c indicate geometry and color, respectively.

$$L = R + (\lambda_g D_g + \lambda_c D_c), \tag{4}$$

.

3.2.3 Output

For each input block, a bitstream representing the encoded latent representation is received at the decoder side. After de-compression, an equally sized degraded version of the block is obtained. When geometry-only compression is required, the model outputs one channel that indicates occupancy. In color-only compression, three color channels are obtained. Notice that, in this case, the receiver knows the point cloud topology; thus, the compressed attributes per point are found at the corresponding voxel position at the output blocks. For geometry-plus-color compression, four channels are obtained combining occupancy and color information. In all cases, the output blocks that are extracted from the same point cloud are merged together following a particular order, to restore the de-compressed point cloud. Finally, the optionally compressed color values are converted back to the original range $[0, 255]$.

3.2.4 Configurations

To train a network, a number of filters $N = 32$ per layer is employed, a batch size of 16, and a number of output channels $C = 4$, to involve both geometry and color information. The Adam optimizer[38] is set with learning rate equal to 10^{-4} and $\beta 1 = 0.9$ and $\beta 2 = 0.999$. The loss function given in Equation 4 is employed, using $\alpha = 0.9$ and $\gamma = 2.0$ for the focal loss computation given by Equation 2. The experiments are conducted using Python 3.6 and Tensorflow 1.13.1. As mentioned earlier, training blocks of size $K = 32$ and testing blocks of size $K = 128$ are in principle employed, except if otherwise declared.

3.3 Evaluation methodology

The majority of quality evaluation metrics in point cloud imaging are capturing either geometric- or textural-only distortions.[35] Lately, a number of new approaches has been reported in the literature that consider both sources of distortion to provide a total estimation of the visual quality. Most notably, projection-based metrics,[39] which are based on conventional 2D imaging metrics and are applied on projected views of the model, while also some model-based metrics,[40,41] which combine topology and texture quality levels in a single score. Yet, the former are sensitive to the rendering approach and the camera parameters to obtain projected views, whereas the latter are sensitive to the selection of weights for geometric and textural attributes, since they do not capture the visual quality of a point cloud holistically. In this study, we opt two well-established objective quality metrics to evaluate the quality of geometry and color information for the compressed models, separately.

For evaluation of geometric distortions, we choose the symmetric point-to-point metric using the PSNR variant, noted as D2-PSNR. The D2-PSNR captures topological distortions in a point cloud model by measuring the deviation of the coordinates of a distorted point from a linear approximation of the reference surface. To compute the PSNR variant, the resolution of the voxel grid that the content lies is employed as the peak value. Two error values are obtained by setting both the compressed and the original model as a reference, and the symmetric error is obtained by choosing the maximum out of the two error values. For evaluation of color distortions, the symmetric color PSNR is adopted. The well-known formula from 2D imaging is employed, using the nearest neighbors algorithm to establish associations between the reference and the content under evaluation. To compute a quality score, the color values of the point cloud models are converted from the original RGB to the

Table 1: Selected values of λ_g and λ_c for the computation of the loss function as in Equation 4, to achieve various distortion allocation schemes with ratios $\lambda_g : \lambda_c$, for different bitrate values (from smallest to largest, R1 to R4).

	1:1		0:1		1:4		1:9		1:19		1:0		4:1		9:1		19:1	
	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c	λ_g	λ_c
R1	20	20	0	40	8	32	4	36	2	38	40	0	32	8	36	4	38	2
R2	100	100	0	200	40	160	20	180	10	190	200	0	160	40	180	20	190	10
R3	500	500	0	1000	200	800	100	900	50	950	1000	0	800	200	900	100	950	50
R4	2500	2500	0	5000	1000	4000	500	4500	150	4750	5000	0	4000	1000	4500	500	4750	150

YUV colorspace using the ITU-R Recommendation BT.709-6;[42] thus, this metric is referred to as YUV-PSNR. To compute a total score, a weighted average between the luma and the two chrominance channels is obtained using weights 6, 1 and 1, as in.[43] This procedure is repeated setting both the original and the distorted models as the reference and the maximum error is kept to account for the symmetric YUV-PSNR score.

To compute both metrics the MPEG software version 0.13.5 is used.[44] For the D2-PSNR, normal vectors are required to be associated with the coordinates of the testing models. In this case, we used a plane fitting algorithm with 10 nearest neighbors as implemented in MeshLab v2020.06.[45]

4. EXPERIMENTAL RESULTS

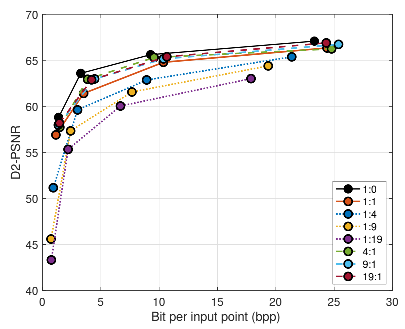
When compressing both the geometric structure and the color attributes of point cloud contents using neural networks, two main approaches can be identified. The first approach relies on creating a holistic representation of both dimensions, feeding both geometry and color information to a network designed to compress both simultaneously. The second approach relies on designing two separate networks to be used sequentially: one that handles geometry, and another that deals with compressing the color attributes. The first approach is advantageous in terms of computational and time resources. Moreover, it allows for an holistic evaluation of point cloud distortions, given a loss function that can reliably detect artifacts in both geometry and color domains at the same time. In the second approach, networks dedicated on a particular type of information are employed and, thus, a better performance is expected provided the usage of the same network hyper-parameters (i.e., number and size of filters, size of strides, etc). Furthermore, the rate allocation for each component can be manipulated independently, thus leading to higher flexibility in the encoding process.

In this section, we describe and provide performance evaluation results for a series of experiments conducted using the unified model as a baseline, which compresses geometry and color attributes simultaneously. In particular, we analyse how the performance of the network is affected when different weights are given to either geometry or color distortions. Then, we compare the performance of our unified model with respect to using separate networks to encode geometry and color information. Finally, benchmarking results against the MPEG anchor are depicted to indicate the performance of the network against a well-established encoding solution.

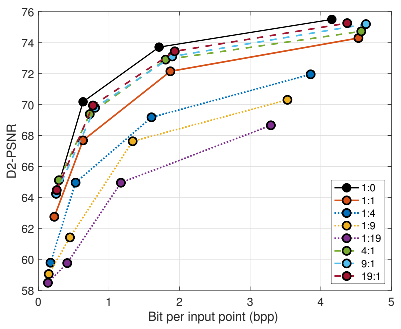
4.1 Distortion allocation for geometry and color compression using the unified network

Figures 5 and 6 depict the performance evaluation of using the unified model to compress both geometry and color, according to geometry metric D2-PSNR and color metric YUV-PSNR, respectively, for all testing contents. To obtain the curves, parameters λ_g and λ_c in the loss function are weighted in order to obtain different allocation schemes, indicated by $\lambda_g : \lambda_c$. For these experiments, the unified model described in Section 3.2.2 and illustrated in Figure 4 is employed.

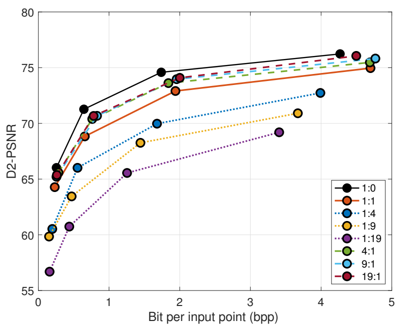
In Table 1, the values of λ_g and λ_c that were selected to achieve the desired weighting for geometry and color distortions, respectively, are reported. Figure 5 indicates how different weighting schemes for geometry and color distortions affect the quality of the reconstructed point cloud in the geometry domain, expressed through the D2-PSNR metric. In particular, the solid black line shows the performance when the color distortion is not considered in the computation of the loss function ($\lambda_c = 0$). As such, it represents an upper limit on the performance in terms of geometrical distortions. The solid red line indicates the performance when equal weights are assigned to both color and geometry distortions, which we consider as the baseline. As expected,



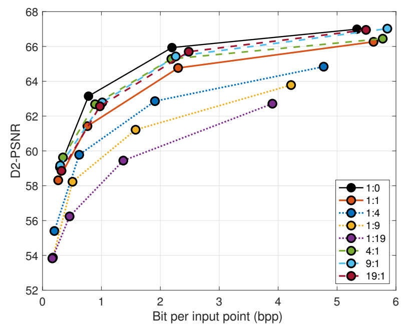
(a) bumbameuboi



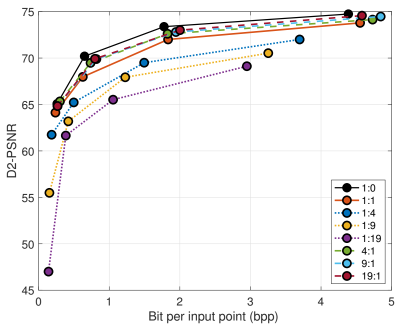
(b) guanyin



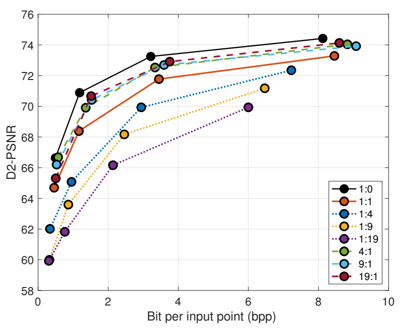
(c) longdress



(d) phil



(e) rhetorician



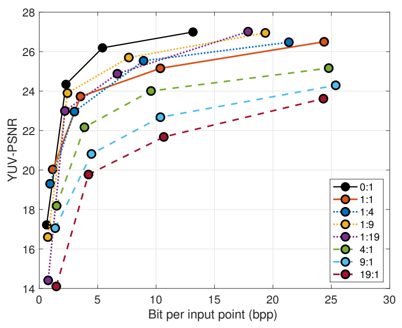
(f) romanoillamp

Figure 5: Rate-distortion performance of the unified network architecture, according to geometry metric D2-PSNR, with different λ allocations to geometry and color ($\lambda_g : \lambda_c$). Solid black represents pure geometry compression ($\lambda_c = 0$), solid red represents 1:1 allocation. Dashed lines represent allocations for which $\lambda_g > \lambda_c$, whereas for dotted lines, $\lambda_g < \lambda_c$.

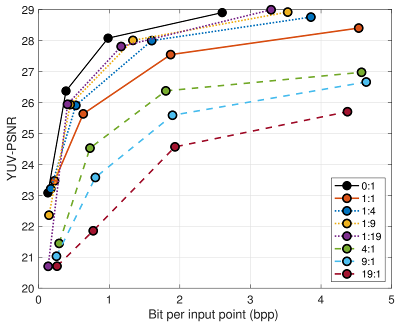
an increase in performance can be observed when more relative weight is assigned to the geometry distortion in the loss function (dashed lines). However, the increase in performance is not as remarkable as the dB losses that are observed when more relative weight is assigned to the color distortion term (dotted lines). In fact, the performance for weight ratios 4:1, 9:1, and 19:1 is approximately equivalent for all contents.

A similar trend can be observed in Figure 6, which presents the performance of the same weighting schemes in terms of color distortion, represented by the YUV-PSNR metric. As in Figure 5, the solid black line indicates the performance when the color distortion is only considered in the loss function ($\lambda_g = 0$). It is noteworthy that, certain allocation schemes mark an increase in performance with respect to the theoretical upper limit 0:1 at low bitrates. This is due to the fact that the computation of the color metric depends on the underlying geometry. Thus, in a geometry-plus-color compression scheme, the reconstructed error is measured on a different than the input topology, which might lead to such behaviours, especially in such low color quality levels. As expected, allocation schemes which favor color distortions (dotted lines) achieve better performance with respect to the 1:1 baseline (depicted in solid red). However, sharp loss in performance can be observed when more weight is assigned to geometry distortions, at the expense of color information (dashed lines).

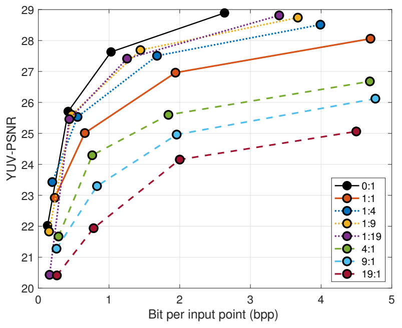
In order to better analyse the impact of varying the relative importance of color or geometry information in the loss function calculation, we computed the Bjontegaard dB gains obtained by each allocation scheme under exam, with respect to the 1:1 baseline. Results are depicted in Figure 7, separately for each test content. Blue color indicates dB gains computed with respect to the color metric YUV-PSNR, whereas red color indicates gains with respect to geometry metric D2-PSNR. Dashed lines represent the theoretical upper limit, i.e., the gains obtained when using only geometry (1:0, red dashed) or only color (0:1, blue dashed) allocations. As we observed before, the gains with respect to the baseline (bars above the 0 line) are quite modest, and tend to saturate between the 1:9 and 1:19 allocation schemes in the case of color gains, and between 9:1 and 19:1



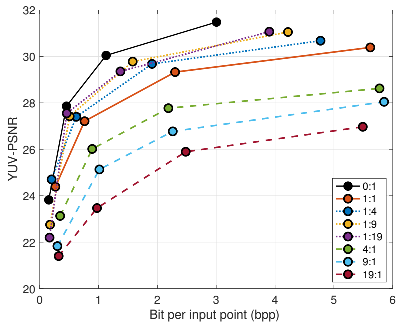
(a) bumbameuboi



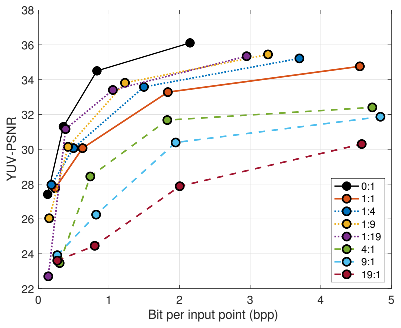
(b) guanyin



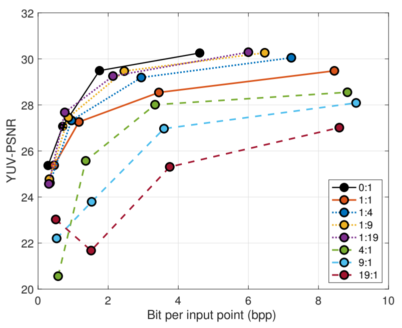
(c) longdress



(d) phil



(e) rhetorician

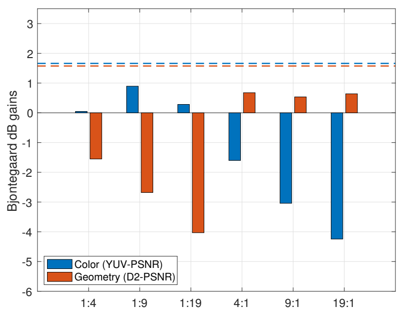


(f) romanoillamp

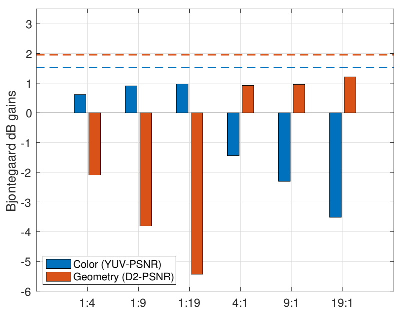
Figure 6: Rate-distortion performance of the unified network architecture, according to geometry metric YUV-PSNR, with different λ allocations to geometry and color ($\lambda_g : \lambda_c$). Solid black represents pure color compression ($\lambda_g = 0$), solid red represents 1:1 allocation. Dashed lines represent allocations for which $\lambda_g > \lambda_c$, whereas for dotted lines, $\lambda_g < \lambda_c$.

in the case of geometry gains. However, steep losses in dB are observed as the distortion allocation schemes become more unbalanced. For content longdress, for instance, we observe a loss of -5.96 dB in the geometry domain when the 1:19 weighting ratio is selected, whereas the corresponding gains in terms of color distortions are limited to 1.08 dB (see Figure 7 (c)). A visual comparison for the weight ratios 1:1, 1:19, and 19:1 at the highest bitrate under consideration is shown in Figure 8 for the content longdress. It can be observed that the geometry distortion introduced by changing λ_g from 2500 to 250, is not heavily influencing the visual perception of the content, despite the reported loss of 2.5 dB. However, in the case of distortion allocation of 19:1, the artifacts in the color domain heavily degrade its appearance, effectively masking any improvements brought in the geometry domain. Figure 9 shows a visual comparison for the same allocation ratios, for content guanyin, at the second lowest bitrate under exam. It can be seen that for a weight ratio of 1:19, geometric artifacts in the form of holes appear (see Figure 9 (c)), whereas assigning larger weight to geometry distortion term brings a very poor performance in color compression. The 1:1 allocation, in this case, represents a compromise between geometry and color distortions.

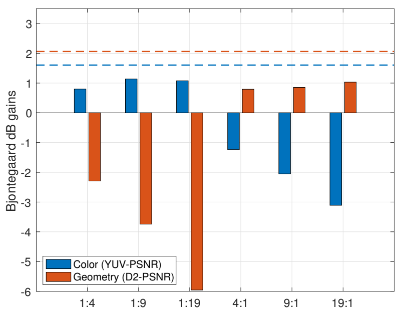
Results show that, while performance gains can be achieved in either geometry or color domain by assigning larger weight to the corresponding type of distortion, they come at the cost of a loss in the other domain. Moreover, losses are generally more pronounced, whereas gains remain modest even when remarkably imbalanced allocation schemes are employed. The selection of the best allocation scheme must be conducted by examining which domain leads to perceptually more pleasant results, and by carefully considering whether the gains in one domain outweigh the costs in the other.



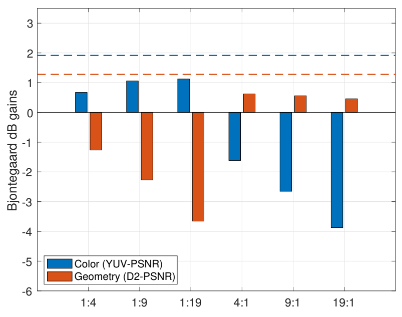
(a) bumbameuboi



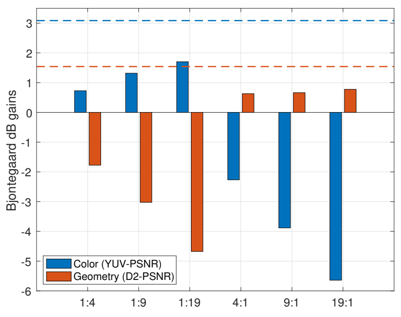
(b) guanyin



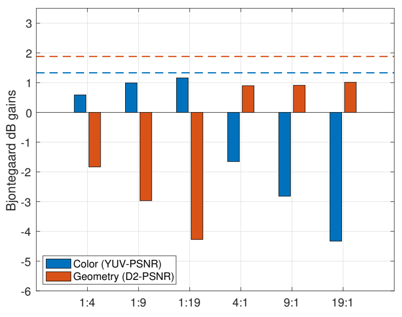
(c) longdress



(d) phil



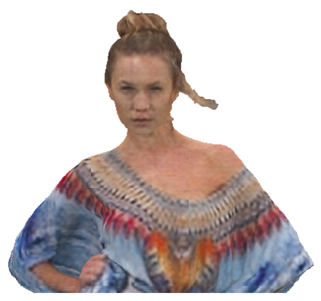
(e) rhetorician

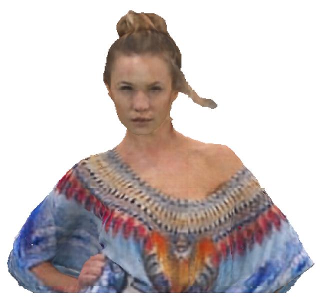


(f) romanoillamp

Figure 7: Bjontegaard dB gains for each allocation $\lambda_g : \lambda_c$ with respect to allocation 1:1, for color metric YUV-PSNR (blue) and geometry metric D2-PSNR (red). Dashed lines represent dB gains when using pure color compression (blue) or pure geometry compression (red), with respect to 1:1 baseline.

(a) Reference


(b) 1:1, $\lambda_g = 2500, \lambda_c = 2500$


(c) 1:19, $\lambda_g = 250, \lambda_c = 4750$

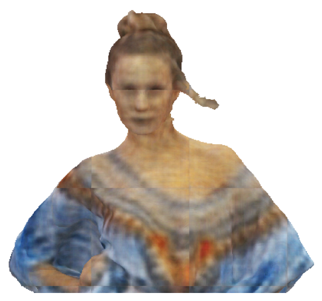
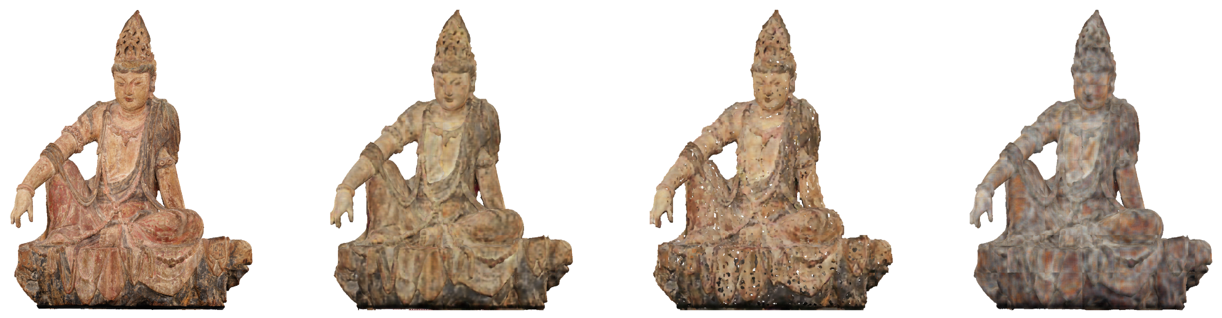

(d) 19:1, $\lambda_g = 4750, \lambda_c = 250$

Figure 8: Visual comparison for content longdress, for different distortion allocation ratios.

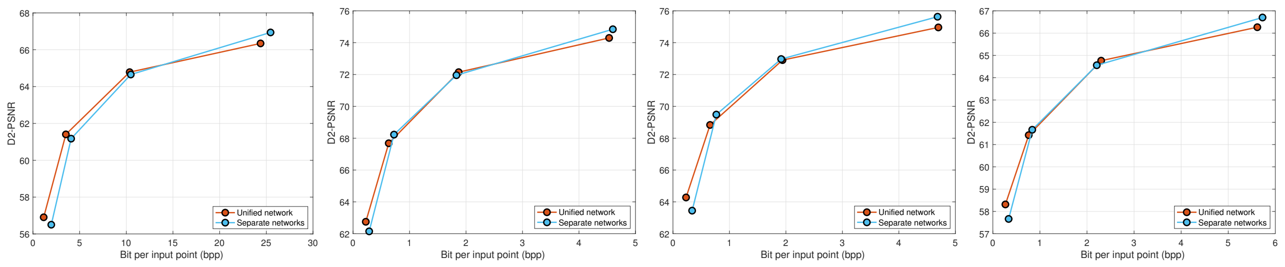
4.2 Comparison of unified network against separately trained networks for color and geometry

For the separately trained networks architecture, two models are employed, each dedicated to compress a particular type of attribute. In our context, we train a model on geometry-only compression and a second model on color-only compression. The testing point clouds are compressed by initially feeding the geometric information of the point cloud data into the geometry-only encoding network, in the form of individual blocks, as described in Section 3.2.1 using $C = 1$. The de-compressed blocks are reassembled to restore the encoded point cloud topology. Then, a re-coloring step is applied by associating the original color values to the de-compressed coordinates using the nearest neighbor algorithm. The resulting point cloud is partitioned again into blocks (input channels $C = 4$) and fed to the color-only encoding network. The output blocks are eventually stitched together, forming the final decoded point cloud. This implementation results in two bitstreams, each corresponding to a different type of attribute, which are both required at the received side in order to restore the encoded model. It should



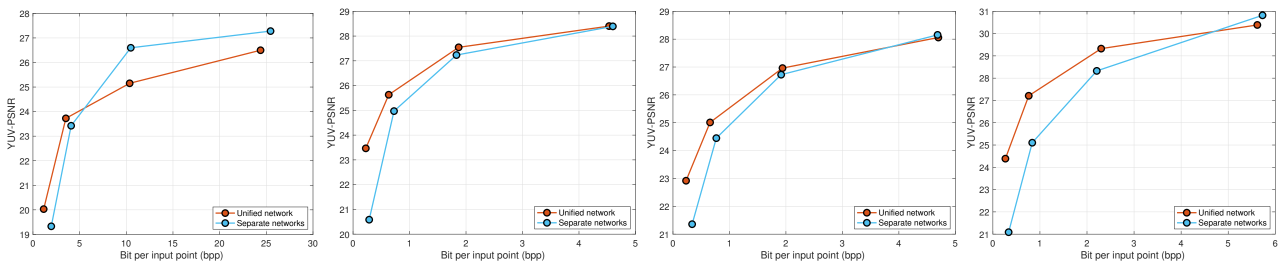
(a) Reference (b) 1:1, $\lambda_g = 100, \lambda_c = 100$ (c) 1:19, $\lambda_g = 10, \lambda_c = 190$ (d) 19:1, $\lambda_g = 190, \lambda_c = 10$

Figure 9: Visual comparison for content guanyin, for different distortion allocation ratios.



(a) bumbameuboi (b) guanyin (c) longdress (d) phil

Figure 10: Rate-distortion performance of the unified model and the separately trained networks, according to geometry metric D2-PSNR.



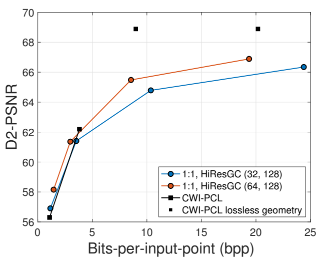
(a) bumbameuboi (b) guanyin (c) longdress (d) phil

Figure 11: Rate-distortion performance of the unified model and the separately trained networks, according to color metric YUV-PSNR.

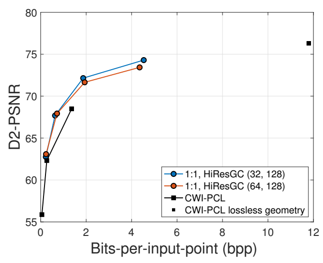
be noted that for the training of both networks, the same data and the same hyper-parameters adopted for the unified version and described in Section 3.2.2 were applied. Moreover, a training and a testing block size of 32 and 128 were used, respectively.

Figures 10 and 11 report the performance evaluation results obtained with the unified network, with 1:1 allocation among geometry and color distortion terms, together with the results obtained from the separately trained networks on geometry and color. Performance is shown using the geometry metric D2-PSNR and the color metric YUV-PSNR, respectively. Due to space limitations, we only report the results for test contents bumbameuboi, guanyin, longdress, and phil. For the unified network, the parameters for distortion allocation 1:1 were used, according to Table 1. For the separately trained networks, parameter λ was set independently for geometry and color; curves are obtained by using (from smallest to highest bitrate), $\lambda_g = \lambda_c = 20, 100, 500, 2500$.

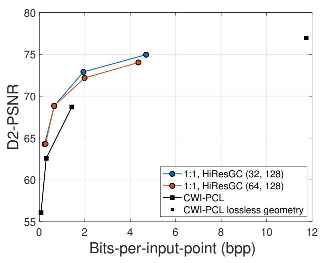
Based on our results illustrated in Figure 10, similar performance is obtained when using the unified model to compress geometry information, with respect to employing an ad-hoc network which is trained on geometry-only data. The two solutions are interchangeable in terms of geometric distortions. In the color domain, however,



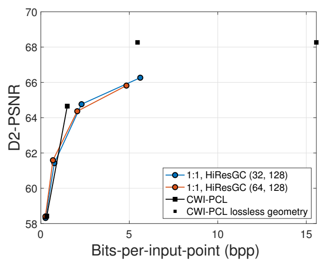
(a) bumbameuboi



(b) guanyin

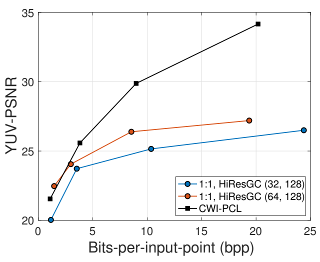


(c) longdress

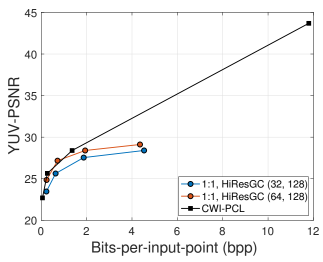


(d) phil

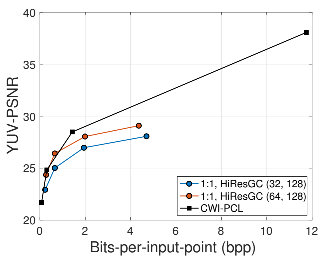
Figure 12: Rate-distortion performance of the of the unified model, trained with block resolution of 32 and 64, against the MPEG anchor, according to geometry metric D2-PSNR.



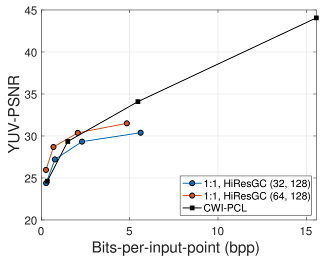
(a) bumbameuboi



(b) guanyin



(c) longdress



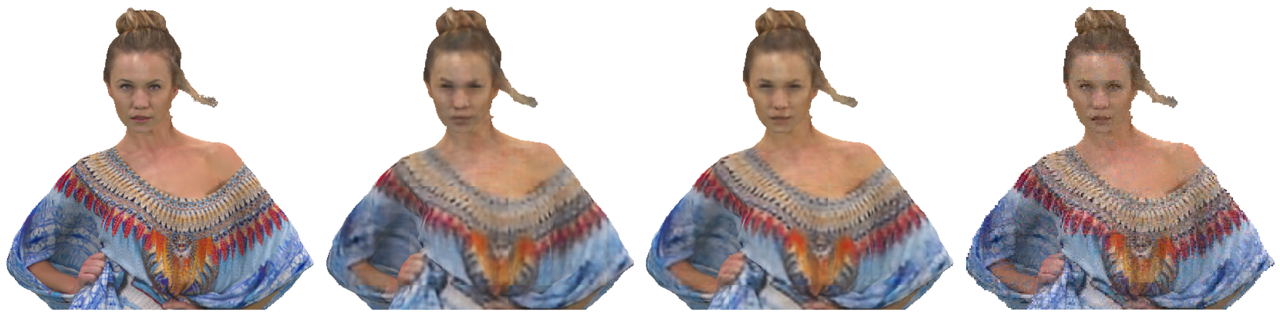
(d) phil

Figure 13: Rate-distortion performance of the unified model, trained with block resolution of 32 and 64, against the MPEG anchor, according to color metric YUV-PSNR.

a difference in performance can be observed between the two solutions, as shown in Figure 11). In particular, for three out of the four contents, i.e., guanyin, longdress, and phil, the two networks have similar performance for high bitrates, whereas for low bitrates, the unified model provides better performance. For bumbameuboi, though, notable gains can be observed for high bitrates, when a separate network is used to compress the color information. This might be due to the complexity of the model, both in the geometric and color domain, which might lead to diminished performance when the two types of information are considered simultaneously. Note that this constitutes a particularly sparse point cloud, which in general behaves as an outlier.

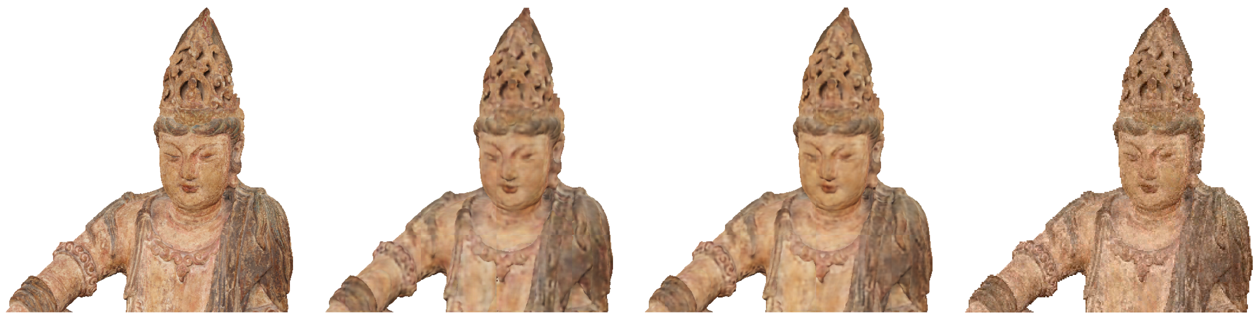
4.3 Benchmarking of unified network

In this section, we examine the performance of the unified network, which is selected as a superior approach based on the results of the previous section, against the anchor codec that was used in the MPEG point cloud compression-related activities. In Figure 12 and 13, rate-distortion curves indicate the performance of the network using a training block size of 32 and 64, which is found to better exploit spatial redundancies[28] (see also Section 5.1) and, thus, leading to lower bit rates for the same visual quality. For block resolution of 32, the λ values for geometry and color distortion were chosen according to the 1:1 ratio in Table 1, whereas for block resolution of 64, $\lambda_g = \lambda_c = \{80, 400, 2000, 10000\}$. For the MPEG anchor, namely, CWI-PCL,[15] we opt for geometry compression Octree bit depths of 7, 8, 9 and 10 and for color compression JPEG Quality Parameter (QP) of 10, 50, 80 and 100, respectively, to obtain scalable visual quality levels by degrading both attributes simultaneously. Note that when the Octree bit depth is equal or higher than the corresponding voxel resolution of a content, lossless geometry compression is essentially applied; thus, leading to a PSNR value of infinity for geometry distortion. These cases are noted with simple markers on the figures to allow indicating the corresponding achieved bitrates (see Figure 12, black squares). It can be observed that for low bitrates, the network achieves comparable or higher performance with respect to the CWI-PCL in terms of geometrical distortions. Similar performance can be observed when considering color distortions, as depicted in Figure 13. In particular, training the network with blocks of resolution 64 leads to better performance with respect to resolution 32, and achieves comparable performance with respect to the CWI-PCL for low bitrates. A quality



(a) Reference (b) HiResGC(32,128), $\lambda_g = \lambda_c = 500$ (c) HiResGC(64,128), $\lambda_g = \lambda_c = 2000$ (d) CWI-PLC, bit depth 9, QP 80

Figure 14: Visual comparison for content longdress, compressed using the proposed network and the MPEG Anchor.



(a) Reference (b) HiResGC(32,128), $\lambda_g = \lambda_c = 500$ (c) HiResGC(64,128), $\lambda_g = \lambda_c = 2000$ (d) CWI-PLC, bit depth 9, QP 80

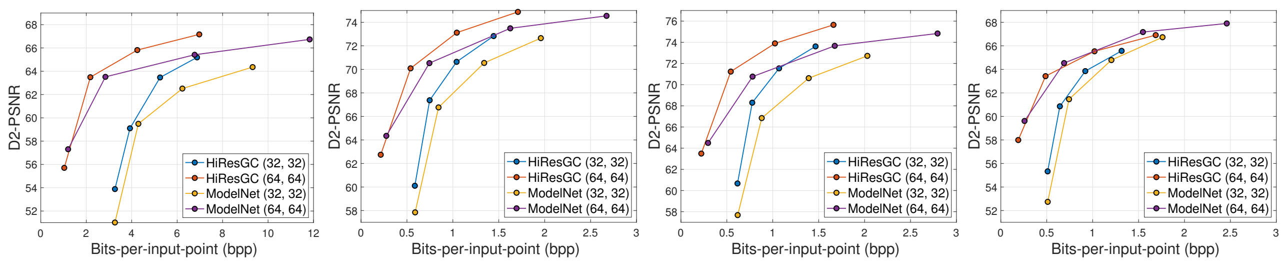
Figure 15: Visual comparison for content guanyin, compressed using the proposed network and the MPEG Anchor.

saturation is shown for the network performance as the bitrate is increasing, indicating the need for more efficient architectures for compression at high fidelity.

Despite the similar quality values that are observed when considering the quality metric YUV-PSNR, visual comparison between the results obtained with the proposed model and the CWI-PCL show that markedly different distortions are introduced by the two compression solutions. Figure 14 shows a zoomed-in region of the content longdress, for the second-highest bitrate. It can be observed that, whereas the CWI-PCL codec contains artifacts in the form of high frequency noise in the color domain, the network tends to have a smoother appearance, at the cost of a loss of detail. It can also be observed that increasing the block resolution from 32 to 64 leads to sharper results and more preserved details. A similar behavior can be seen for the content guanyin, as depicted in Figure 15. In particular, smoother texture is obtained when encoding with the network architecture with respect to CWI-PCL, as the former introduces artifacts in form of low-pass filtering, whereas false contours are present using the latter.

5. META-ANALYSIS

Neural networks represent a powerful tool to learn a compact representation of given data. As such, they have been largely employed to tackle compression for 2D visual data representations, and have recently been extended in point cloud data formats. However, a number of issues remains to be faced when considering compression of point clouds through neural network architectures, both when considering the distribution of the points in 3D space, and when trying to encode the accompanying attributes. In this section, we aim to shed some light regarding the influence and the selection for a number of hyper-parameters that affect the learning efficiency of a given network architecture. Note that the same network parameters and configurations specified in Section



(a) bumbameuboi (b) guanyin (c) longdress (d) phil

Figure 16: Rate-distortion performance of the geometry-only network, using different datasets and training data resolutions, according to geometry metric D2-PSNR.

5.1 Selection of training data for geometry compression

Inspired from the different approaches,[23, 25] in the generation of training data for point cloud geometry compression, in this experiment, we aim to evaluate the impact of using different datasets and grid resolutions. In general, there are two main lines that have been reported in the literature for the generation of relevant training data. In the first approach,[23] a mesh repository is employed and point cloud models are generated through sampling, and potentially voxelizing at a desirable grid resolution. Typically, the original mesh models are artificially generated, and represent full-shaped colorless objects. In the second approach,[25] which is adopted in our experimental setup, high-resolution point cloud contents are collected from available repositories. Such contents typically consist of either real-life acquired and synthetic point clouds that span across a variety of categories.

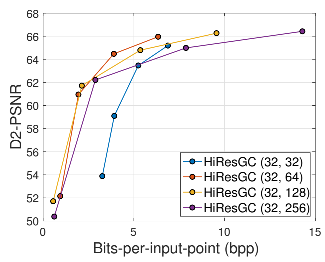
Provided that point clouds are generally comprised of a considerable amount of points, whose sheer size and irregular structure make them unsuitable for being directly handled by neural networks, a common choice is to apply voxelization and block partitioning at a low resolution. Nonetheless, setting a specific block size against another influences the performance of the network, as has been shown in previous studies.[28] Adding attribute encoding increases the complexity, as they will necessarily depend on the underlying 3D structure to be encoded.

In this experiment, to account for the first approach, we use point clouds extracted from the ModelNet data set, as described in.[23] The models are scaled and regularly sampled, before being voxelized at a specific geometric resolution. To analyse the impact of the geometric resolution on the performance efficiency, voxel grid resolutions of 32 and 64 are employed for every model. To account for the second approach, we use the HighResGC dataset that has been defined for our experimental setup (see Section 3), using block resolutions of 32 and 64. In both cases, point cloud units that contain less than 500 occupied voxels are discarded, and from the remaining data, a number of 10,000 is randomly sampled. In summary, we use four different training sets of 10,000 colorless samples: two are extracted from the ModelNet data set and the other two from our generated data set, with grid resolution of 32 and 64 each. In this experiment, the testing models are partitioned in blocks of the same resolution as the one that was used for the training data (32 and 64).

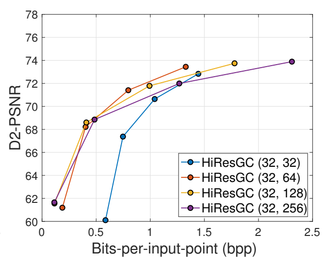
In Figure 16, performance evaluation for 4 out of the 6 testing models is illustrated, using both data sets for learning, at both grid resolutions. It can be observed that better compression efficiency is achieved by the network when trained with the HighResGC, when compared to the ModelNet counterpart. Moreover, there is a clear trend of reaching higher performance when using a block resolution of 64, under both training sets. It is worth noting that the gains in compression efficiency come at the cost of higher demands in terms of resources and time, as blocks of resolution 64 require more computational power.

5.2 Resolution of testing data

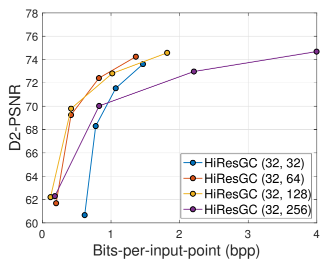
The choice of a given block resolution for training data does not imply that the same grid size must be selected for the testing data. In fact, larger testing blocks can be chosen for compression, denoting another parameter that can potentially affect the reconstruction quality of point clouds. In this experiment, as a first step, we quantify the performance of our network in geometry compression by using different grid resolutions for the testing data. For this purpose, we use 4 different variations of the network, trained with the HiResGC and the ModelNet data sets and training blocks of size 32 and 64. The selected resolutions for the testing blocks under evaluation were



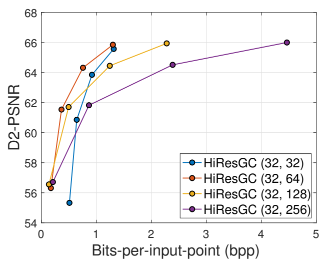
(a) bumbameuboi



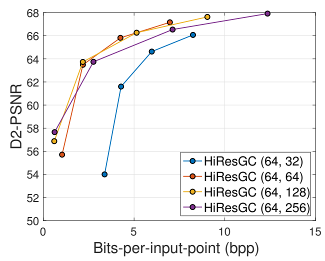
(b) guanyin



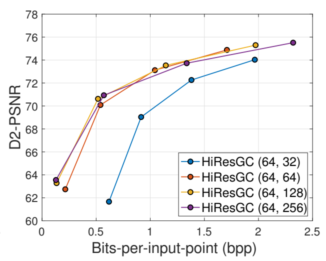
(c) longdress



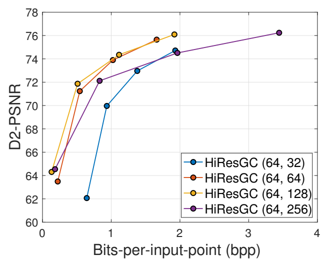
(d) phil



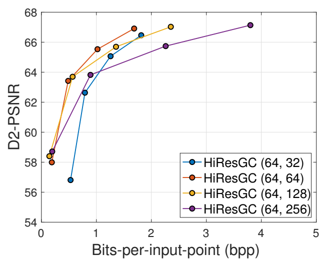
(e) bumbameuboi



(f) guanyin

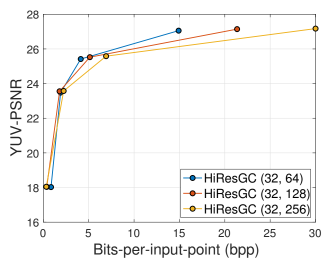


(g) longdress

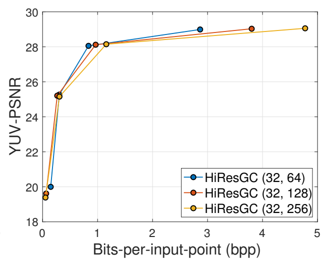


(h) phil

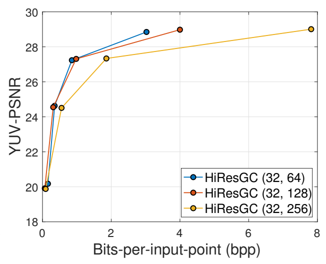
Figure 17: Rate-distortion performance of the geometry-only network, for different testing grid resolutions, according to geometry metric D2-PSNR. First row represents results obtained with a training block resolution of 32, whereas the second row depicts results with training block resolution of 64.



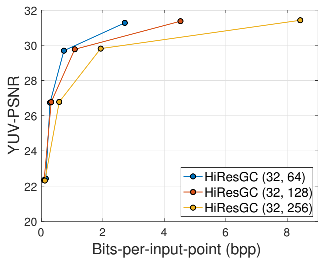
(a) bumbameuboi



(b) guanyin



(c) longdress



(d) phil

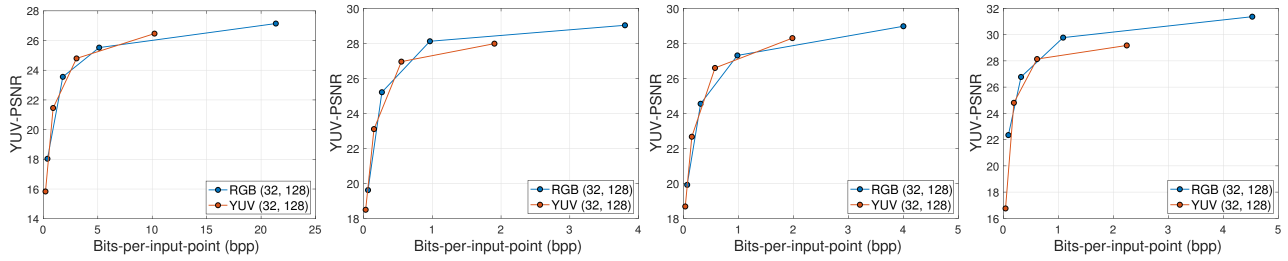
Figure 18: Rate-distortion performance of the color-only network, for different testing grid resolutions, according to color metric YUV-PSNR. In parenthesis, the training data resolution that was used for the learned model.

set to: 32, 64, 128, and 256. In a second step, the HiResGC dataset and a training block size of 32 is employed to examine the quality levels of the reconstructed color using testing block sizes of 64, 128, and 256.

In Figure 17, performance evaluation results for the geometry-only network are illustrated, showing rate-distortion curves for 4 out of the 6 testing models; very similar results are obtained for the rest of the contents. First row represents results obtained with networks trained with a block resolution of 32, whereas the second row depicts results with training block resolution of 64. As can be observed, in both cases testing grid resolutions of 64 and 128 achieve the best results. Similar conclusions are obtained when using the ModelNet dataset to train the networks, at a generally more modest overall performance.

In Figure 18, we present the performance evaluation results for the color-only network, for the same 4 contents. As can be seen, increasing the testing resolution leads to performance saturation, as equivalent quality levels are obtained among the models at higher bitrates. It is worth noting, however, that the influence of border artifacts, which appear due to the block partitioning step, is not necessarily captured by the objective quality metrics. Moreover, the independent encoding/decoding of blocks might lead to different color distributions exhibiting among neighboring regions, which is a quite visible and annoying visual degradation for colored point clouds. Naturally, smaller block resolutions would lead to a more evident appearance of this effect, despite the fact that identical quality scores are obtained at the different testing resolutions.

5.3 Color space



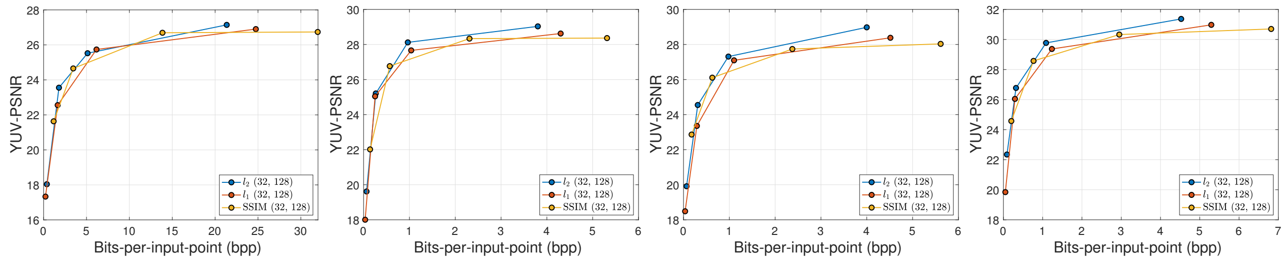
(a) bumbameuboi (b) guanyin (c) longdress (d) phil

Figure 19: Rate-distortion performance of the color-only network, for different input color spaces, according to color metric YUV-PSNR.

Another parameter that could potentially affect the results of color learning is the type of representation the textural information is provided to the network. Convolution neural networks typically learn local features and optimize filter weights in order to achieve data-driven compact representations. However, it is unclear whether using different bases in the network can effectively influence the results. In this experiment, we opt to examine the performance of the network when using the RGB and the YCbCr/YUV color spaces. The latter has effectively been used in classical image and video compression, while the first depicts the most widely-used color format that has been used in machine learning applications.

For this experiment, we used the ITU-Recommendation BT.709-6[42] for conversion between RGB and YUV. The RGB color values for both training and testing datasets were converted to YUV, and then normalized between 0 and 1. Note that no color conversion is applied at any layer of the network. Thus, the loss function is always computed in the corresponding input color space. Results of the comparison between RGB and YUV are depicted, for 4 out of the 6 contents, in Figure 19. It can be observed that in general, both color spaces have similar performance. Slight gains can be observed at high bitrates when the RGB color space is employed, for the contents guanyun and, more remarkably, phil. Thus, it appears that color space selection does not have a large impact on the compression performance of color attributes.

5.4 Loss function



(a) bumbameuboi (b) guanyin (c) longdress (d) phil

Figure 20: Rate-distortion performance of the color-only network, for different loss functions, according to color metric YUV-PSNR.

The performance of neural network architectures is affected by the choice of the loss function that is used to train a model. In order to assess whether performance gains could be obtained by using a different loss function for computing distortions in the color domain, we tested three different objective quality metrics, namely, l_1, l_2, and SSIM, with the former two denoting the most popular approaches that are used in similar network tasks. To obtain the loss value, the corresponding distance (l_1, or l_2) is computed between the color channels of the original point cloud and the recovered point cloud across the input point coordinates. For the computation of the SSIM, which denotes a more perceptually-relevant metric, the same equation as in[46] is used. However, instead of computing the metric on the Y channel, we decided to use it for all RGB channels. In,[47] it was shown that using SSIM on RGB channels could reflect the quality of the recovered images. Moreover, a filter size of 6, instead of the default 11, to reduce computational costs. To be consistent with the other losses, we applied some simple manipulations to make the range of the loss be within 0, indicating no error, and 1, indicating the largest possible error. As a result, the SSIM loss is defined as follows:

$$L_{SSIM} = \frac{1 - SSIM}{2} \tag{5}$$

For all loss functions under exam, the logarithm function is applied at the output value.

Results of the evaluation of different loss functions for color attributes are depicted in Figure 20, for 4 out of 6 contents in exam. It can be observed that all loss functions show similar performances. Slight gains can be observed when using l_2 at high bitrates. Thus, it can be concluded that in our setup, the choice of the loss function does not seem to have a significant impact on the performance of the network under exam. However, the l_1 or l_2 would be the most compelling choices, considering the reduced costs with respect to SSIM.

6. CONCLUSIONS

In this paper, we present a novel neural network architecture to simultaneously handle the encoding of geometry and color attributes of point cloud contents. In principle, this study can be interpreted as a first attempt to compress both geometry and texture of point clouds using neural networks. Several parameters are examined, and conclusions are drawn regarding their efficiency, paving the way for next efforts. Our network competes with the anchor encoder that was employed in the MPEG activities; however, there is a large, unexplored space that can lead to further improvements. For instance, provided that a point cloud model is split into a series of blocks that is handled independently, due to memory and computational limitations, there is no effort in learning redundancies between neighboring blocks, by enabling for instance intra or inter prediction techniques. Moreover, it has been seen that variational auto-encoders applied on the feature space can remarkably assist by improving the learning efficiency for the entropy model; such an addition is not tested in our network. Finally, it is well-known that high-quality training data are required for better performance; the availability of well-established training data sets with a representative range of geometric and textural complexities are of crucial importance, and would facilitate future efforts.

ACKNOWLEDGMENTS

This work has been conducted in the framework of the Swiss National Foundation for Scientific Research project Advanced Visual Representation and Coding in Augmented and Virtual Reality (SNSF grant 178854).

REFERENCES

[1] Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P. A., Cohen, R. A., Krivokuća, M., Lasserre, S., Li, Z., Llach, J., Mammou, K., Mekuria, R., Nakagami, O., Siahaan, E., Tabatabai, A., Tourapis, A. M., and Zakharchenko, V., "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **9**, 133–148 (March 2019).

[2] Jackins, C. L. and Tanimoto, S. L., "Oct-trees and their use in representing three-dimensional objects," *Computer Graphics and Image Processing* **14**(3), 249 – 270 (1980).

[3] Schnabel, R. and Klein, R., "Octree-based point-cloud compression," in [*Symposium on Point-Based Graphics 2006*], Botsch, M. and Chen, B., eds., Eurographics (July 2006).

[4] 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).

[5] 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).

[6] Dricot, A. and Ascenso, J., "Hybrid octree-plane point cloud geometry coding," in [*2019 27th European Signal Processing Conference (EUSIPCO)*], 1–5 (2019).

[7] de Oliveira Rente, P., Brites, C., Ascenso, J., and Pereira, F., "Graph-based static 3d point clouds geometry coding," *IEEE Transactions on Multimedia* **21**(2), 284–299 (2019).

[8] Krivokuća, M., Koroteev, M., and Chou, P. A., "A volumetric approach to point cloud compression," (2018).

[9] Zhang, C., Florencio, D., and Loop, C., "Point cloud attribute compression with graph transform," IEEE - Institute of Electrical and Electronics Engineers (October 2014).

[10] 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)*], 1–4 (2017).

[11] de Queiroz, R. L. and Chou, P. A., "Transform coding for point clouds using a Gaussian process model," *IEEE Transactions on Image Processing* **26**(7), 3507–3517 (2017).

[12] 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).

[13] Cohen, R. A., Tian, D., and Vetro, A., "Point cloud attribute compression using 3-D intra prediction and shape-adaptive transforms," in [*2016 Data Compression Conference (DCC)*], 141–150 (2016).

[14] 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).

[15] 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**(4), 828–842 (2017).

[16] MPEg 3DG, "Video-based and hierarchical approaches point cloud compression." ISO/IEC JTC1/SC29/WG11 Doc. N19092 (Mar. 2020).

[17] Pereira, F., Dricot, A., Ascenso, J., and Brites, C., "Point cloud coding: A privileged view driven by a classification taxonomy," *Signal Processing: Image Communication* **85**, 115862 (2020).

[18] Toderici, G., O'Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M., and Sukthankar, R., "Variable rate image compression with recurrent neural networks," in [*4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*], Bengio, Y. and LeCun, Y., eds. (2016).

[19] Theis, L., Shi, W., Cunningham, A., and Huszár, F., "Lossy image compression with compressive autoencoders," (2017).

[20] Ballé, J., Laparra, V., and Simoncelli, E. P., "End-to-end optimized image compression," (2016).

[21] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N., "Variational image compression with a scale hyperprior," (2018).

[22] Cheng, Z., Sun, H., Takeuchi, M., and Katto, J., "Deep convolutional autoencoder-based lossy image compression," in [*2018 Picture Coding Symposium, PCS 2018 - Proceedings*], 253–257 (Sept. 2018).

[23] 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).

[24] Quach, M., Valenzise, G., and Dufaux, F., "Improved deep point cloud geometry compression," (2020).

[25] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., "Point cloud coding: Adopting a deep learning-based approach," in [*2019 Picture Coding Symposium (PCS)*], 1–5 (2019).

[26] 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).

[27] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., "Deep learning-based point cloud geometry coding: Rd control through implicit and explicit quantization," in [*2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*], 1–6 (2020).

[28] Wang, J., Zhu, H., Ma, Z., Chen, T., Liu, H., and Shen, Q., "Learned point cloud geometry compression," (2019).

[29] Huang, T. and Liu, Y., "3d point cloud geometry compression on deep learning," in [*Proceedings of the 27th ACM International Conference on Multimedia*], 890–898 (2019).

[30] Yan, W., shao, Y., Liu, S., Li, T. H., Li, Z., and Li, G., "Deep autoencoder-based lossy geometry compression for point clouds," (2019).

[31] 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).

[32] Quach, M., Valenzise, G., and Dufaux, F., "Folding-based compression of point cloud attributes," (2020).

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

[34] Zerman, E., Gao, P., Ozcinar, C., and Smolic, A., "Subjective and objective quality assessment for volumetric video compression," in [*IS&T Electronic Imaging, Image Quality and System Performance XVI*], (2019).

[35] 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).

[36] Perry, S., "JPEG Pleno Point Cloud Coding Common Test Conditions v3.1." ISO/IEC JTC1/SC29/WG1 N86044 (Jan 2020).

[37] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P., "Focal loss for dense object detection," in [*Proceedings of the IEEE international conference on computer vision*], 2980–2988 (2017).

[38] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980* (2014).

[39] Torlig, E. M., Alexiou, E., Fonseca, T. A., de Queiroz, R. L., and Ebrahimi, T., "A novel methodology for quality assessment of voxelized point clouds," in [*Applications of Digital Image Processing XLI*], **10752**, 107520I, International Society for Optics and Photonics (2018).

[40] Viola, I., Subramanyam, S., and César, P., "A color-based objective quality metric for point cloud contents," in [*2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*], 1–6, IEEE (2020).

[41] 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).

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

[43] Ohm, J.-R., Sullivan, G. J., Schwarz, H., Tan, T. K., and Wiegand, T., "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology* **22**(12), 1669–1684 (2012).

[44] Tian, D., Ochimizu, H., Feng, C., Cohen, R., and Vetro, A., "Updates and Integration of Evaluation Metric Software for PCC." ISO/IEC JTC1/SC29/WG11 Doc. MPEG2017/M40522 (Apr 2017).

[45] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G., "MeshLab: an Open-Source Mesh Processing Tool," in [*Eurographics Italian Chapter Conference*], Scarano, V., Chiara, R. D., and Erra, U., eds., The Eurographics Association (2008).

[46] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing* **13**(4), 600–612 (2004).

[47] Zhao, H., Gallo, O., Frosio, I., and Kautz, J., "Loss functions for image restoration with neural networks," *IEEE Transactions on computational imaging* **3**(1), 47–57 (2016).