On the stability of point cloud machine learning based coding

João Prazeres, Rafael Rodrigues, Manuela Pereira, and Antonio M. G. Pinheiro
Instituto de Telecomunicações & Universidade da Beira Interior, Covilhã, Portugal
joao.prazeres@ubi.pt, rafael.rodrigues@ubi.pt, mpereira@di.ubi.pt, pinheiro@ubi.pt

***Abstract*—This paper analyses the performance of two of the most well known deep learning-based point cloud coding solutions, considering the training conditions. Several works have recently been published on point cloud machine learning-based coding, following the recent tendency on image coding. These codecs are typically seen as a set of predefined trained machines. However, the performance of such models is usually very dependent of their training, and little work has been considered on the stability of the codecs' performance, as well as the possible influence of the loss function parameters, and the increasing number of training epochs. The evaluation experiments are supported in a generic test set with point clouds representing objects and also more complex scenes, using the point to point metric (PSNR D1), as several studies revealed the good quality representation of this geometry-only point cloud metric.**

***Index Terms*—Point cloud coding, machine learning-based codecs**

I. Introduction

The usage of 3D data formats has been increasing recently, most notably in virtual (VR), augmented (AR) and mixed reality applications, but also in a wide variety of other fields, such as computer graphics, 3D printing, construction, manufacturing, robotics, automation, medical applications, retail, cultural heritage, remote sensing, and geographical information systems.

Point cloud technology is one of the most popular solutions for 3D data representation, which maps surfaces on a Cartesian coordinate system (x, y, z). Each mapped point might have a list of associated attributes, including RGB components, reflectance, physical sensor information, or normal vectors. Point clouds can provide accurate representations of both objects and scenes, from any viewing position or distance.

The models of data representation and their associated quality play an essential role in point cloud applicability, as 3D content often creates huge amounts of information. Accurate point cloud representations of landscapes, buildings or artefacts typically contain several millions of points, each with one or more associated attributes. Thus, depending on the represented object or scene, the size of raw point clouds may become impractical, especially in real-time applications. For that reason, efficient point cloud compression and decompression solutions are needed.

Research funded by the Portuguese FCT-Fundação para a Ciência e Tecnologia under the project UIDB/50008/2020, PLive X-0017-LX-20, and by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competencias em Cloud Computing.

Two of the most recognized and used coding solutions were developed by MPEG, i.e., the Video-based Point Cloud Compression (V-PCC) [1] and the Geometry-based Point Cloud Compression (G-PCC) [2]. Authors in [3] provide a quality study of both codecs. Google developed DRACO[1], which provides both lossless and lossy point cloud encoding. As shown in [4], the MPEG codecs perform better than DRACO, in terms of quality vs bit rate.

Recently, machine learning-based point cloud coding solutions have been emerging, following this trend on image and video coding. The Multiscale Point Cloud Geometry Compression (PCGC) was proposed by Wang *et al.* [5], and further developed in [6]. In [7] the Deep Point Cloud Geometry Compression (PCC_GEO_CNN) was presented, with an improved version proposed in [8]. Adaptive Deep Learning Point Cloud Compression was presented in [9]. In [10], a point cloud lossy attribute auto encoder is proposed, directly encoding and decoding attributes with the help of geometry. In [11], a deep convolutional autoencoder is proposed that directly operating on the points. It also considers a deconvolution operator in order to upsample point clouds, allowing decompression to an arbitrary density.

When dealing with learning-based methods, it is well known that the performance of the final learned model may vary, even with similar training conditions, due to the stochastic nature of the learning process. However, most research efforts in deep learning-based coding report their results as unique. In this work, the performance of two learning-based codecs is assessed based on the robustness of the compression performance from different training processes, under similar conditions. More specifically, the stability of PCGC [6] and PCC_GEO_CNN [8] is assessed, based on the resulting coding performances. For each step of the learning progression, a set of six point clouds is encoded and the point to point metric (PSNR D1) [12] is computed. The PSNR D1 metric was chosen, as the codecs only encode geometry. The obtained results are analysed to verify if there is a convergence to a stable operating point. The point cloud point to point metric (PSNR D1) is used in this work as it revealed to be always one of the best that uses the geometry only [3], [4].

978-1-6654-6623-3/22/$31.00 ©2022 IEEE

[1]https://github.com/google/draco

II. Codecs Architecture description

A. Multiscale Point Cloud Geometry Compression

The Multiscale Point Cloud Geometry Compression (PCGC) model [6] features a mirrored encoder-decoder architecture, which performs consecutive downsampling to multiple scales. The encoder consists of three sparse convolution modules in sequence, each containing two convolutional layers with 3x3x3 kernels and ReLU activations. In the second convolution layer, a stride of 2x2x2 is used for downsampling. After the sparse convolution modules, a residual feature extraction step is added, consisting of three instances of the Inception Residual Network [13]. The final latent representation Y is given by an additional sparse convolution layer, with 8 3x3x3 filters.

At the bottleneck, the two components of Y, i.e., the geometry coordinates (C_Y) and feature attributes (F_Y) are encoded separately, using the lossless octree codec [2] and entropy coding, respectively. In the decoding process, C_Y and F_Y are merged and upsampled through a convolutional branch that mirrors the encoder. In each module, transposed convolutions with stride 2x2x2 are used to upsample the sparse point clouds. After each upsampling step, a single 3x3x3 convolution with sigmoid activation is used to obtain the voxel occupation probability (p_i), which is in turn used to compute a scale-specific binary cross-entropy (BCE) loss:

$$L_{BCE} = -\frac{1}{N}\sum_{i}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right) \quad (1)$$

where y_i is i^{th}-voxel true label (1 if occupied, 0 otherwise).

The model training aims at optimizing the following Lagrangian loss function:

$$L = R + \lambda D \quad (2)$$

where D refers to the geometrical distortion and R is the resulting bit rate of feature attribute encoding ($\hat{F}_y$). D is obtained from the multi-scale BCE loss, which is the average of the BCE losses computed at each scale. Finally, the λ parameter sets the rate-distortion trade-off. Each time the λ value decreases, the training model will define a new working point where the bit rate is lower but the distortion measure will also decrease, as it has lost weight in the loss function.

B. Deep Point Cloud Geometry Compression

The Deep Point Cloud Geometry Compression (PCC_GEO_CNN) model [7] has a fairly straightforward learning-based approach, which proposes to reduce the blocking effect typically introduced by other learning-based codecs.

The model architecture features an encoder part (f_a) with three sequential convolutional layers, each with 32 filters. The first layer uses a 9x9x9 kernel with stride 2x2x2, whereas the other two use a 5x5x5 kernel with stride 2x2x2. A ReLU activation is used in the first two layers. The latent representation $y = f_a(x)$ is given by the linear output of the third layer. This latent representation is then quantized ($\hat{y} = Q(y)$) through element-wise integer rounding. $\hat{y}$ compression is performed using the Deflate algorithm, which is a combination of LZ77 and Huffman coding [14].

The decoder branch f_s consists of three transposed convolutional layers, which mirror the encoder in terms of number of filters (except for the last layer), kernel size and stride. All layers in the decoder use a ReLU activation function. The last layer uses a single filter and provides the distorted point cloud $\tilde{x}$, using element-wise minimum, maximum and rounding functions. In the decoding process, p_z^t is the probability of a point z being occupied. The global loss function of the PCC_GEO_CNN model is the same as the Eq. 2. The distortion component is given by the overall of the focal loss (Eq. 3),

$$L_{Focal}(x,\tilde{x}) = \alpha_z(1-p_z^t)^\gamma log(p_z^t) \quad (3)$$

which allows to compensate for the imbalanced ratio between occupied and non-occupied voxels. Given the balance parameter α, α_z is 1 if the corresponding voxel in the original point cloud is occupied, and 1-α otherwise.

III. Experimental setup and results

The experiments carried out will study the evolution of the codecs through the learning process. For that, the test data set will be coded/decoded after each epoch and the PSNR D1 is computed. The process is repeated three independent times to observe the stability of the codecs. The complete test will allow to understand the stability of the codecs with the same training conditions.

A. Test data selection

In this work, the performance of the tested point cloud codecs was assessed on a test set comprised of six point clouds, available at the JPEG Pleno database[2]. Three of these point cloud depict objects and the other three depict landscapes. The chosen object point clouds were the *Romanoillamp*, from the University of Sao Paulo Database[3], the *Guanyin* from the EPFL dataset, and frame 1300 of the *Longdress* dynamic point cloud. The selected landscapes are three point clouds from the University of Sao Paulo Database, namely *Citiusp*, *IpanemaCut* and *Ramos*. The six selected point clouds are shown in Figure 1.

B. PCGC Model Training

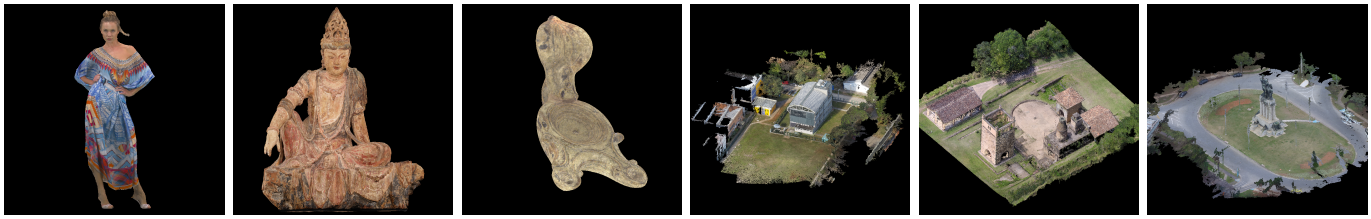
A study of the influence of model training on the performance of the PCGC codec [6] is established. This model was selected because information is given on the training procedures and even the training datasets are shared[4].

In [5], different coding bit rates are targeted, by varying the rate-distortion trade-off parameter λ between 0.75 and 16. In the code made available, the global loss function J depends on two parameters, α and β, such that $J = \alpha D + \beta R$. In this experiment, β was fixed at 1, so that α becomes equivalent to λ in Eq. 2.

[2]http://plenodb.jpeg.org/pc/8ilabs

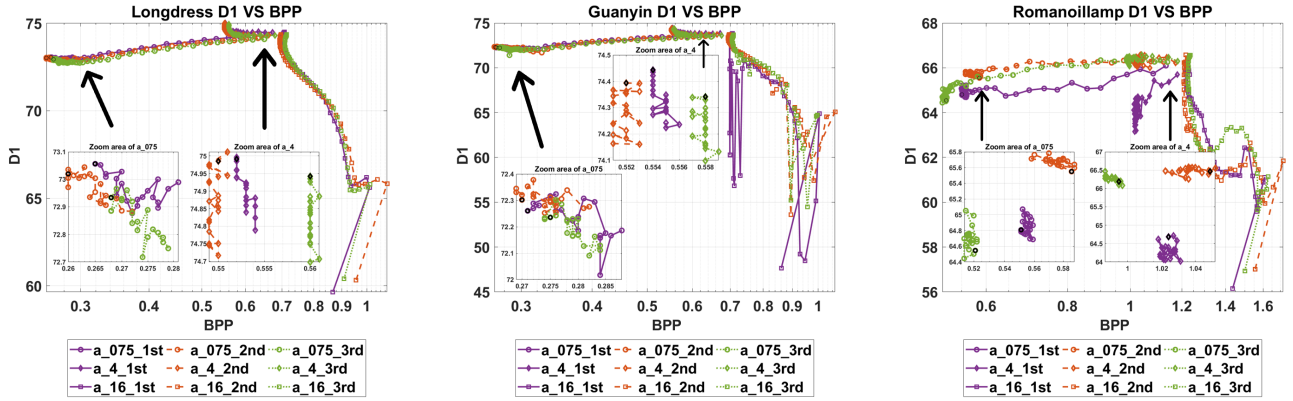
[3]http://uspaulopc.di.ubi.pt

[4]available at https://github.com/NJUVISION/PCGCv2

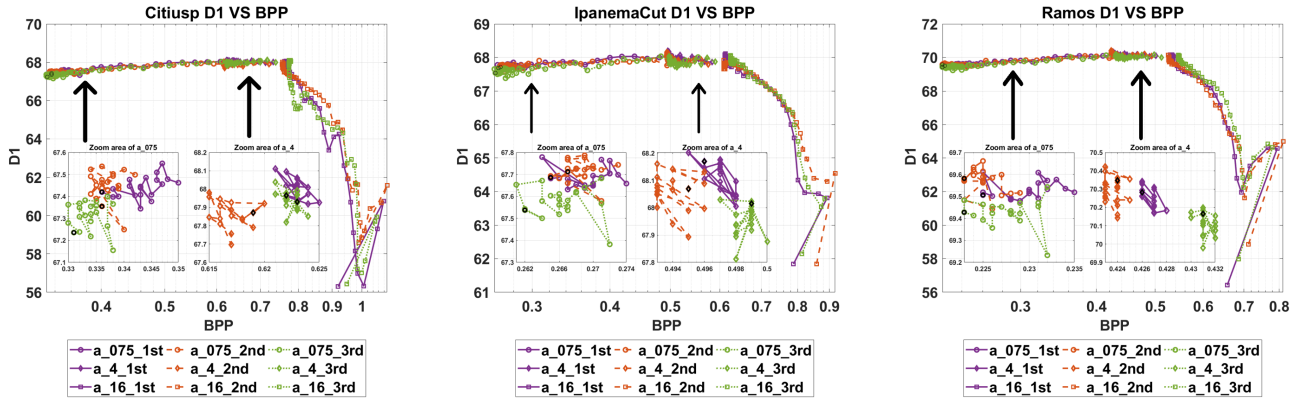


(a) *Longdress* (b) *Guanyin* (c) *Romanoillamp* (d) *Citiusp* (e) *IpanemaCut* (f) *Ramos*

Fig. 1: Point Cloud test set.



(a) *Longdress* (b) *Guanyin* (c) *Romanoillamp*

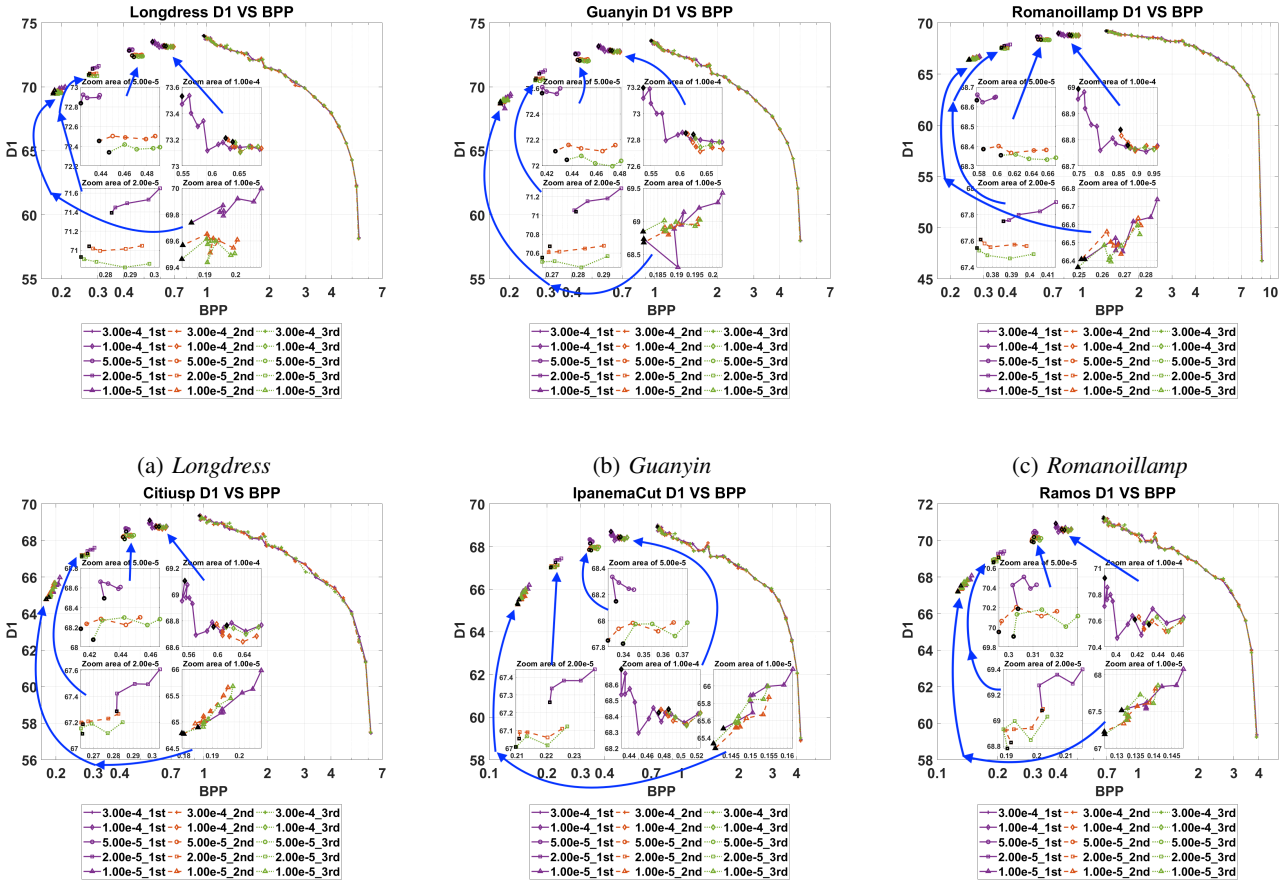


(d) *Citiusp* (e) *IpanemaCut* (f) *Ramos*

Fig. 2: PSNR D1 vs. bpp plots for PCGC, trained with $\alpha = \{16, 4, 0.75\}$.

The model was trained with densely sampled data from the ShapeNet [15], a database containing $\approx$ 51.300 3D surface models. The final training set was obtained by random rotation and quantization with 7-bit precision. Also, the number of points in each point cloud was also randomized. In this paper, PCGC was trained with $\alpha = \{16, 4, 0.75\}$, each for 50 epochs, with a constant learning rate of 10^{-5}. For faster convergence, the learned weights with $\alpha = 16$ were used to initialize the training with both $\alpha = 4$ and $\alpha = 0.75$, as recommended in [5]. This training routine was ran three times with similar conditions. The result of the training sessions is shown in figure 2, with zoomed areas of interest.

The PCGC codec shows an acceptable level of stability for the *Citiusp*, *Longdress*, *IpanemaCut* and *Ramos*. In all of these cases, the D1 metric shows a similar behaviour across the training process. For the *Citusp* case, when $\alpha = 16$, some instability is shown between different epochs, across the different training sessions. For the point cloud *Guanyin*, a high level of instability is observed when $\alpha = 16$ for the first training session. In this case, D1 shows an inconsistent behaviour across the first training session, while the second and third sessions, a different, more stable behavior is observed,



(a) *Longdress* (b) *Guanyin* (c) *Romanoillamp*

(d) *Citiusp* (e) *IpanemaCut* (f) *Ramos*

Fig. 3: PSNR D1 vs. bpp plots for PCC_GEO_CNN, trained with $\lambda = \{3 \times 10^{-4}, 10^{-4}, 5 \times 10^{-5}, 2 \times 10^{-5}, 10^{-5}\}$.

although some instability is found in higher bit rates. This is not observed for $\alpha = 4, 0.75$, where all the training sessions show very little D1 variation. In the encoding process of the *Romanoillamp* point cloud, the codec shows a high amount of instability. The D1 metric shows inconsistent behaviour across different training sessions. Contrary to what is observed for the other contents, the bit rate does not converge to a stable operating point, except for $\alpha = 16$.

In most cases, the codec reveals a good level on the encoding performance stability. Across all training sessions epochs, the bit rates converge to similar operating points. However, some content might show some undesirable change in the encoding performance, depending on the training process. Moreover, there is the assumption that none of the point clouds used in this test is present in the training data as they are not included in the Shape Net database, up to the authors knowledge.

C. PCC_GEO_CNN model training

In [8], the authors train four individual models for each Rate-Distortion tradeoff. They chose four values for λ (eq. 2), notably 3×10^{-4}, 10^{-4}, 5×10^{-5}, 2×10^{-5}). In the software provided by the authors[5], an additional value is considered, $\lambda = 10^{-5}$. This experiment followed the sequential training approach described in [8], where each training using λ_i is initialized with the trained weights of the previous model (i.e., using λ_{i-1}). The first training uses $\lambda_1 = 3 \times 10^{-4}$, and targets a low distortion, high bit rate model. Subsequent training uses λ values in descending order, which results in a progressive reduction of the target bit rates, while attempting to minimize the increase in distortion. The α and γ parameters of the focal loss function were set to 0.9 and 2, respectively, which were the default values in the provided code.

The test point clouds were encoded at each 500 training steps, which is the validation interval defined in the provided code. In the original code, the model checkpoint was saved at a given validation point, only if there is an improvement in the loss from the last validation point. However, for this experiment, the code was adapted to bypass this definition and save the checkpoints at every 500 training steps, to encode the test point clouds. An early stopping condition was also implemented in the original code, which interrupts the training

[5]avaliable at https://github.com/mauriceqch/pcc_geo_cnn_v2

(a) *Guanyin* cropped area of the original

(b) 1st training session, epoch=50, $\alpha = 0.75$, PSNR D1=64.808

(c) 2nd training session, epoch=50, $\alpha = 0.75$, PSNR D1=65.55

(d) 3rd training session, epoch=50, $\alpha = 0.75$, PSNR D1=64.5461

(e) 1st training session, step 31500, $\lambda = 10^{-5}$, PSNR D1=66.4068

(f) 2nd training session, step 28000,$\lambda = 10^{-5}$, PSNR D1=66.4068

(g) 3rd training session, step 27000,$\lambda = 10^{-5}$, PSNR D1=66.3627

Fig. 4: Decompressed *Guanyin* (Cropped area) for the low bit rate (final epoch) of each codec training. The first row shows the decoded point clouds for PCGC, and the second row for PCC_GEO_CNN.

(a) *Romanoillamp* cropped area of the original

(b) 1st training session, epoch=50, $\alpha = 0.75$, PSNR D1=64.808

(c) 2nd training session, epoch=50, $\alpha = 0.75$, PSNR D1=65.55

(d) 3rd training session, epoch=50, $\alpha = 0.75$, PSNR D1=64.5461

(e) 1st training session, step 31500, $\lambda = 10^{-5}$, PSNR D1=66.4068

(f) 2nd training session, step 28000, $\lambda = 10^{-5}$, PSNR D1=66.4068

(g) 3rd training session, step 27000, $\lambda = 10^{-5}$, PSNR D1=66.3627

Fig. 5: Decompressed *Romanoillamp* (Cropped area) for the low bit rate (final epoch) of each codec training. The first row shows the decoded point clouds for PCGC, and the second row for PCC_GEO_CNN.

process if the loss does not improve for more than 4 validation steps.

The models are trained on a subset of the ModelNet40 [16] dataset. First, the mesh data is voxelized with resolution $512 \times 512 \times 512$ and the 200 largest point clouds are selected. Then, the point clouds are divided into blocks with resolution $64 \times 64 \times 64$ and the 4000 largest blocks are selected.

Three different training sessions were carried out, using the five λ values described above. Figure 3 shows the results of the three training sessions, with zoomed areas of interest. The codec shows a high level of stability across the two additional training sessions, although in most cases, the D1 metric has a slight variation in the intermediate λ values. When analysing the training plots for *Citiusp*, the second and third training are highly similar when $\lambda = 3 \times 10^{-4}$ and 10^{-5}. For $\lambda = 5 \times 10^{-5}$, the D1 value for the first training session is above the achieved in the other two training sessions. The same can be observed when $\lambda = 10^{-4}$. This behaviour is found across the tested point clouds. Some small degrees of instability can also be found in the point clouds *Ramos* when $\lambda = 10^{-4}$ and in the point cloud *Guanyin* when $\lambda = 10^{-5}$.

Overall, this codec shows a higher degree of stability than PCGC. When encoding the *Guanyin*, the training sessions with $\alpha = 16$ reveal a small degree of instability. Other points where different training processes reach different Rate-Distortion relations are observed mostly to *Guanyin* and *Romanoillamp*.

However, these differences are very small and this codec always find working points that reveal a higher level of stability than PCGC.

D. Visual Examples

It is also important to visualize some examples to understand the variation of each training session in the decoded point clouds. Figure 4 shows a cropped area of the decoded *Guanyin* point clouds for each training session for both codecs. All training sessions of PCGC show some artifacts in the torso area. Small parts seem to be missing, always in different locations on the torso of the point cloud. In the face area, the second training session produced almost no artifacts, while the first and third training produced some holes. PCC_GEO_CNN reveal much more noticeable artifacts for the lower bit rate. All sessions produced a number of distortions in the face, especially training session two, where cracks can be seen in the nose area. In the torso area, some artefacts can be identified, like the ones found in PCGC, but in a larger scale.

Figure 5 also shows a cropped area of the decoded *Romanoillamp* point clouds for each training session for both codecs. PCGC creates an enormous amount of artifacts across the point cloud. The figure shows that it was impossible for the codec to find a good rate/distortion trade-off. All training sessions produced the same type of artifacts, PCC_GEO_CNN shows a similar behaviour to PCGC, albeit in a smaller scale. All training sessions created similar artifacts in the handle area of the point cloud. In the central area, some variation in the location of the artifacts can be observed.

IV. Discussion and conclusions

A stability analysis of the training process of two machine-learning based codecs, namely PCGC and PCC_GEO_CNN is reported. According to our analysis, both codecs show a high level of stability in the coding performance. PCC_GEO_CNN reveals the most stable across the training sessions. In the training of PCGC, the training dataset is randomly selected from the provided database, while in PCC_GEO_CNN, a static database is used. Given that, it was expected that PCC_GEO_CNN to be the most stable across the conducted training sessions.

When the desired rate-distortion points are reached for each codec, some instability is observed for both codecs. This effect is more visible for the PCGC. However, PCC_GEO_CNN has a stopping mechanism that prevents this effect to become visible. While PCGC has training session with a fixed number of 50 epochs, PCC_GEO_CNN never reaches that value. The number of training epochs is dynamically computed and the training session is stopped.

The observed instability near the limits of the training sessions might be due to some over-fitting mechanism, that does not allow to improve anymore the rate-distortion relation. This causes that different training sessions might have variable performance for the last training epochs of a training process, causing the observed instabilities for some content.

Moreover, it was also observed that different training sessions create different artifacts for the selected working points, although the PSNR D1 metric has similar values. The point to point metric is used for optimization, in both codecs leading to this result. However, it is likely to happen that different metrics can have different values. It is also important to understand if different training can lead to different perceptual quality. That requires to perform subjective evaluation, that has several problems that need to be considered. For instance, how to render the color in the point clouds, as that will have tremendous impact in the perceived quality and might mask any other quality analysis. The authors plan to consider this analysis in future studies.

Nevertheless, both codecs revealed a very reasonable stability on the performance for different training sessions, showing a high reliability. This is mostly observed for the PCC_GEO_CNN where the reached Rate-Distortion working points have only slight variations for different training processes. Nevertheless, PCGC also reveals a high level of stability, and can also be considered a reliable codec. It is expected that different training sessions will tend to create different artifacts, although the PSNR D1 metric is kept similar as it was used in the cost function for the codec optimisation.

References

[1] V. Zakharchenko, ""Algorithm description of mpeg-pcc-tmc2"," *ISO/IEC JTC1/SC29/WG11 MPEG2018/N17767*, Jul 2018.

[2] K. Mammou, P. A. Chou, D. Flynn, and M. Krivokuća, "G-PCC codec description v2," *ISO/IEC JTC1/SC29/WG11 N18189*, Jan 2019.

[3] S. Perry et al, "Quality evaluation of static point clouds encoded using mpeg codecs," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020.

[4] J. Prazeres, M. Pereira, and A. M. G. Pinheiro, "Quality analysis of point cloud coding solutions," in *2022 Electronic Imaging Symposium*, 2022.

[5] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Learned point cloud geometry compression," *CoRR*, 2019.

[6] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," 2020.

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

[8] ——, "Improved deep point cloud geometry compression," *CoRR*, vol. abs/2006.09043, 2020. [Online]. Available: https://arxiv.org/abs/2006.09043

[9] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, 2021.

[10] X. Sheng et al, "Deep-pcac: An end-to-end deep lossy compression framework for point cloud attributes," *IEEE Transactions on Multimedia*, 2021.

[11] L. Wiesmann et al, "Deep compression for dense point cloud maps," *IEEE Robotics and Automation Letters*, 2021.

[12] D. Tian et al, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017.

[13] C. S. et al, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, 2016.

[14] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[15] A. X. Chang et al, ": An information-rich 3D model repository," 2015. [Online]. Available: https://arxiv.org/abs/1512.03012

[16] Z. Wu et al, "3D shapenets: A deep representation for volumetric shapes," 2014. [Online]. Available: https://arxiv.org/abs/1406.5670