ON BLOCK PREDICTION FOR LEARNING-BASED POINT CLOUD COMPRESSION

Davi Lazzarotto, Evangelos Alexiou, Touradj Ebrahimi

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

2021 IEEE International Conference on Image Processing (ICIP) | 978-1-6654-4115-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIP42928.2021.9506429

ABSTRACT

Point clouds are among popular visual representations for immersive media. However, the vast amount of information generated during their acquisition requires effective compression for practical applications. Although relevant activities from standardization bodies have led to state-of-the-art compression using conventional methods, learning-based encoders have recently emerged as promising solutions with comparable performance while offering additional attractive features. Yet, there is still a large unexplored space for research that can lead to further advances. In this paper, we propose a block prediction module for bit-rate reduction of geometry-only point clouds. Our method exploits spatial redundancies at the decoding stage between block partitions in the point cloud, and predicts a query block using Generative Adversarial Networks. Results show performance improvements of the objective metrics at low bit-rates, after integration in a baseline auto-encoder architecture.

Index Terms— Point cloud compression, auto-encoder

1. INTRODUCTION

Point cloud imaging has emerged as a promising visual data representation for modern 3D communication systems and information technologies. In this type of content, the scene topology is defined by points spanning throughout the three-dimensional space, with optional attributes that describe their appearance, such as color values and normal vectors, among others. Despite their flexible nature, a huge amount of information is required for faithful scene representation, which in turn implies the need for efficient compression solutions in most practical situations.

Anticipating the potential and the wide range of use-cases for this type of visual data, relevant activities have been carried out during the last few years by both JPEG and MPEG standardization committees for the establishment of standards that will enable a common ground and assist an efficient deployment of point cloud technology into the market [1, 2]. These activities have also enabled additional efforts for the development of novel compression technologies, triggering further research by the scientific community and industry.

Compression solutions found in the literature today can be classified as (i) model-based, (ii) projection-based, or (iii) deep learning-based. The first category operates on the point cloud domain, making use of efficient data structures, handcrafted transforms, and prediction algorithms to reduce the data size. The second involves methods making use of 2D encoding algorithms applied to projected views, or patches of a point cloud. The third category relies on artificial neural network architectures that learn data-driven transforms after training, and apply them for encoding the queried models. The latter denotes newly introduced approaches that were inspired by the remarkable results observed by corresponding 2D imaging solutions, and are expected to be widely explored in the near future.

In this study, we propose a neural network architecture for compression of point cloud geometry, introducing a predictive coding module at the decoder stage. To respect limitations of computational resources, a point cloud is partitioned into blocks and each block is fed into the network, similarly to other current auto-encoding solutions. A block is either encoded independently, or predicted by its neighbors, which resembles the underlying working principle of inpainting. This decision is based on the local topology of the model and the quality of the reconstructed block. Results show the proposed prediction module leads to performance gains at low bit-rates.

The next sections are structured as follows: first, we review related works on learning-based point cloud coding; the proposed prediction method is then described followed by presentation and discussion of results. The paper is then concluded with directions for future.

2. RELATED WORK

Examples of model-based compression include [3] and [4], which propose octree-based progressive compression that rely on approximations of the underlying surfaces to predict neighboring occupancy. Denser shape approximations after octree decomposition are enabled by reconstructing the underlying surface using triangular primitives, also known as "Triangle Soup" (TriSoup), as described in [5], or planar surfaces, as proposed in [6]. The MPEG Geometry-based Point

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

Cloud Compression (G-PCC) test model [7], for geometry coding belongs to this category.

Among projection-based algorithms we can find the MPEG Video-based Point Cloud Compression (V-PCC) test model [8]. The latter employs HEVC to encode the two video sequences generated to capture geometry and texture information. In more recent studies, algorithms to improve the encoding efficiency of V-PCC have been proposed, based on padding of projected patches [9], and better predictions of the motion vectors [10].

Deep learning-based approaches commonly exploit auto-encoding architectures that target compression of geometry-only information in a block-by-block basis. Two early studies are presented in [11, 12], using shallow architectures composed of convolution and de-convolution layers for analysis and synthesis. The impact of several parameters added to the initial version of the former network is evaluated in [13]. In [14], a rate-distortion performance analysis is conducted on the latent space of [12], which is enriched with a hyper-prior and the possibility of explicit quantization in [15]. In [16], a deeper auto-encoding architecture is proposed, based on 3D convolution layers stacked with Voxception-ResNet structures and a hyper-prior. In [17], an encoding scheme relying on folding of a 2D grid onto a point cloud is proposed, with the attributes of the latter being mapped on top of it. In [18], geometry and color information is encoded directly in the 3D domain by extracting features from regular grids, using 3D convolutions and capturing spatial redundancies. The objective of this paper is not to assess the performance of the above approaches.

3. PROPOSED METHOD

3.1. Overview

Our work consists in a predictive coding module that can be integrated into any block-based point cloud compression algorithm that works with voxel blocks of fixed size. In this paper, we combine it with the geometry-only version of the learning-based solution proposed in [18]. One disadvantage of the baseline implementation is that it does not exploit spatial redundancies between adjacent blocks. The proposed method enhances the baseline by allowing prediction of blocks at the decoding stage using already decoded neighbors. The predicted blocks are therefore excluded from the encoding process and corresponding bits are not added to the bitstream, thus, reducing the bit-rate. To limit distortions within an acceptable range, a labeling process is implemented at the encoder stage to decide which blocks are encoded and which are predicted. At the decoder stage, a learning-based architecture is employed to predict the non-encoded blocks. The high level architecture of our compression algorithm is presented in Figure 1.

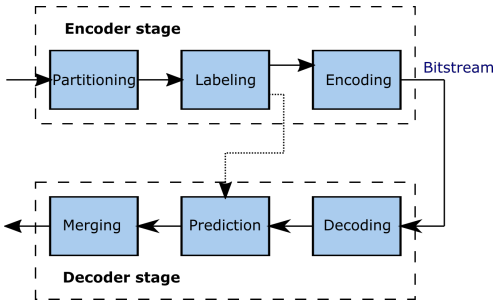


Fig. 1. Block diagram of the proposed method.

3.2. Block labeling module

The block labeling module consists of three criteria that are employed to classify voxel blocks as encoded or predicted. All criteria must be met for a block to be labeled as predicted.

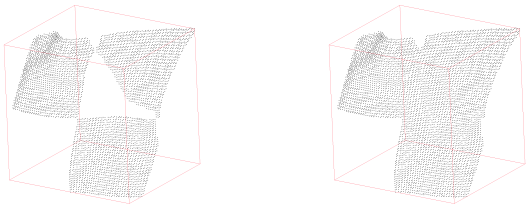
During the partitioning process of the input point cloud, each block receives an integer index (i_b, j_b, k_b) representing its position in the 3D space. The first criterion establishes that the sum of these three indexes has to be an odd number, ensuring that if a block with index (i_b, j_b, k_b) is set to be predicted, all of its six direct neighbours with indexes $(i_b \pm 1, j_b, k_b)$, $(i_b, j_b \pm 1, k_b)$ and $(i_b, j_b, k_b \pm 1)$ will be encoded.

The second criterion specifies that one predicted block must have at least three occupied neighbours. Otherwise, it is considered that the available information is not sufficient for a block to be predicted from its surroundings.

Finally, the distortion between a predicted block and the corresponding block from the original point cloud should not exceed a pre-set threshold. The distortion is quantified by the symmetric mean squared point-to-point error [19], while the threshold is manually specified. The prediction module generates all blocks selected by the first two criteria at the encoder side, and blocks with error lower than the threshold are labeled as predicted.

3.3. Block prediction module

The prediction module operates at the decoder stage and generates the blocks labeled as predicted from their neighbours. It is based on a Generative Adversarial Network that takes as input a macro-block with empty voxels in the center and is trained to generate that same macro-block with the central region filled. The missing region at the center of the macro-block corresponds to the block to be predicted, while the neighbouring occupied voxels belong to decoded blocks that were labeled as encoded. Note that only the direct neighbours of a block are used for prediction, i.e. blocks that have a shared face; thus, the macro-block side length is three times larger than that of a block. After prediction, only the central region of the output macro-block is kept, which corresponds to the predicted block. An example of an input and a target for prediction are illustrated in Figure 2.



(a) Input macro-block. (b) Target macro-block.

Fig. 2. Input (a) and target (b) macro-blocks for the prediction module.

The selected architecture is similar to [20]. A generator is trained to produce a macro-block with the missing region filled, while a conditional discriminator can receive as input either the generated macro-block or the reference, being trained to determine whether the input is real or not. This value is also used in the loss function of the generator, which is trained to fool the discriminator. During testing, only the generator is used.

The generator is based on an auto-encoder architecture with skipped connections between layers of same size at the encoder and the decoder. The encoder is made of four 3D convolutional layers that progressively reduce the size of the input cubic macro-block from 96 to a feature space with size 4, using strides of 3 in the first layer and 2 in the following, and an increasing number of filters of 32, 64, 128 and 256. The decoder has a symmetric architecture using transposed 3D convolutions that re-scale the macro-block back to its input shape, with 128, 64, 32 and 1 filters, respectively. At the input of each layer of the decoder, the output with same dimensions from the encoder is concatenated. On the bottleneck, two fully connected layers with 5000 and 8192 nodes and ReLu activations are applied on the flattened output of the encoder, followed by a reshaping layer that builds a cubic block of size 4 and 128 channels that is fed to the decoder of the generator. The encoder and decoder use LeakyReLu activations, except for the last layer of the decoder that uses a sigmoid.

Assuming that the input macro-block is x, the target is y, and the output is y', the discriminator takes as input a macro-block with two channels, formed by the concatenation of x and y or y'. The first is considered as real while the latter is considered as fake. Its architecture is similar to the encoder of the generator, with four 3D convolutional layers having twice the amount of filters than the generator. It outputs a cubic block of size 4 and 512 channels which is then reshaped to a vector, similarly as in [20].

The loss function of the generator is composed of three terms with weights controlled by the hyper-parameters β and λ_l, as depicted in (1).

$$loss_g = (1-\beta)\cdot FL(\alpha_{FL},\gamma_{FL}) + \beta\cdot loss_{adv} + \lambda_l \cdot loss_{lapl} \quad (1)$$

The $FL(\cdot)$ is the same focal loss used in the baseline codec [18], while the adversarial loss term $loss_{adv}$ is given by (2),

$$loss_{adv} = -\mathbb{E}[D(y'|x)] \quad (2)$$

with $\mathbb{E}[\cdot]$ indicating expectation, D the discriminator, and $y'|x$ the concatenation of the output and the input of the generator. Finally, the laplacian loss term $loss_{lapl}$ is given in (3)

$$loss_{lapl} = \frac{1}{N^3}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} L(y'(i,j,k))\cdot y(i,j,k) \quad (3)$$

with N set equal to 96 corresponding to the size of the macro-block, $(i,j,k) \in [0,95]$ being the coordinates of a voxel in the macro-block and $L(\cdot)$ depicting the discrete laplacian operation formulated as per (4).

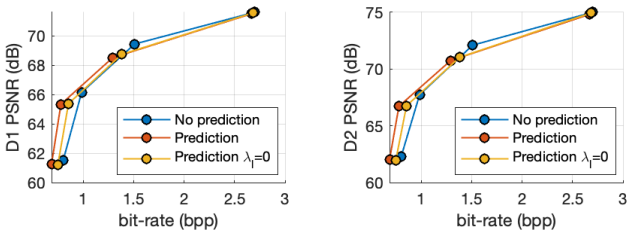
$$\begin{aligned} L(y'(i,j,k)) = -6\cdot y'(i,j,k) + y'(i-1,j,k) + \\ y'(i+1,j,k) + y'(i,j-1,k) + y'(i,j+1,k) + \\ y'(i,j,k-1) + y'(i,j,k+1) \end{aligned} \quad (4)$$

The laplacian loss term is proposed in order to take into account the spatial distribution of the output of the generator. Its introduction is motivated by the fact that the focal loss has a limited capacity of capturing spatial context, since it is calculated on each voxel separately without considering its neighbours. The underlying assumption for the proposed term is that the generated point cloud block should represent a thin surface. Therefore, for voxels that are occupied at the target, the generated value should decrease fast on the direction of the normal of the surface. This condition is met if the value of the second derivative on these voxels is negative and has a large absolute value, which is achieved by the minimization of the proposed term.

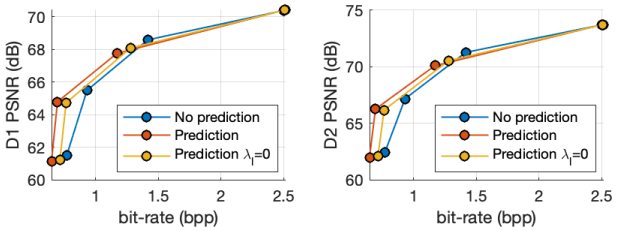
4. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed coding method, the HighResGC dataset assembled in [18] was employed, which consists of 44 point clouds for training and 6 for testing. Since the prediction scheme was designed for geometry-only point clouds, the color attributes present in the models of the dataset were ignored. For training purposes, the hyper-parameter values of the generator loss function were set as follows: $\beta = 0.05$, $\lambda_l = 0.1$, $\alpha_{FL} = 0.95$, $\gamma_{FL} = 2$. Moreover, the loss function of the discriminator was the same as in [20]. Finally, the Adam Optimizer [21] was used with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate of $5\cdot 10^{-4}$ for the generator and of 10^{-4} for the discriminator.

The baseline codec was trained using 10000 randomly chosen blocks with size 32 from the training set. Blocks with less than 500 occupied voxels were discarded. Four values of λ were defined in the loss function to obtain different rate-distortion trade-offs: $\{20, 100, 500, 2500\}$. The prediction



(a) *longdress D1 metric* (b) *longdress D2 metric*



(c) *rhetorician D1 metric* (d) *rhetorician D2 metric*

Fig. 3. Rate-distortion plots for two contents using metrics D1 and D2.

Models	s_1	s_2	s_3
bumbameuboi	-0.0105	-0.0412	5.1128
guanyin	-0.1518	-0.6002	4.9281
longdress	-0.2833	-1.0154	4.4292
rhetorician	-0.4183	-0.9071	3.8445
romanoillamp	-0.2324	-1.0789	4.5224
phil	-0.4433	-0.557	3.034

Table 1. Variation of the D2 PSNR in dB.

Models	s_1	s_2	s_3
bumbameuboi	-4.52	-6.7	10.61
guanyin	-11.67	-18.34	1.1
longdress	-14.41	-20.86	-2.78
rhetorician	-15.87	-25.41	-9.74
romanoillamp	-14.79	-25.1	-11.1
phil	-12.21	-14.08	6.49

Table 2. Variation of the bit-rate in (%).

module was trained with a dataset composed by macro-blocks assembled using all blocks from the training set with their corresponding direct neighbours, excluding instances with less than 500 points and less than three occupied neighbours. An ablation study for the proposed laplacian loss term was also performed. All point clouds from the test set were then compressed and reconstructed using the baseline method, as well as with the proposed prediction module trained with and without the laplacian loss term. The values of the threshold for the point-to-point metric used on the block labeling module were set to 4, 2, 1 and 0.5 for each λ, respectively.

For each decoded point cloud, the PSNR values of the D1 (point-to-point) and D2 (point-to-plane) metrics were computed using the MPEG software version 0.13.5. In order to compute D2, the normals of the reference point clouds were estimated with plane fitting using 10 nearest neighbours and MeshLab v2020.06. The rate-distortion plots for two contents of the dataset are reported in Figure 3.

Based on our results, we can observe that the prediction method is performance-wise beneficial, mainly at low bit-rates. For the highest bit-rates, almost no change is observed when including the prediction module into the encoding process. This can be explained by the lower threshold value used for the point-to-point criterion on the block labeling process at higher values of λ, which substantially reduced the number of predicted blocks. Indeed, the percentage of blocks labeled as predicted is 16%, 22%, 13% and 1% for λ equal to 20, 100, 500 and 2500, respectively. We also observe that the models obtained with the prediction module trained with the laplacian loss consistently achieve lower bit-rates for equivalent distortion values.

The Bjontergaard-Delta bit-rate was computed using the D2 PSNR metric in two distinct scenarios. The prediction module was compared both to the baseline codec and to itself trained without the proposed laplacian loss term. We observe an average bit-rate saving of -8.95% in the first case and of -4.83% in the second, illustrating both the performance gain of our method and of the laplacian loss.

In order to better evaluate the performance of the prediction module at low bit-rates, Tables 1 and 2 present the bit-rate and D2 PSNR gains, respectively, in three scenarios s_1, s_2 and s_3. The first two compare the algorithm with and without prediction for the two lower bit-rate levels. The third scenario compares the lowest compression level of the baseline codec to the second level of the codec with the proposed method. Based on Table 2, we observe that for s_1 and s_2, we achieve a bit-rate reduction always greater than 10% except for *bumbameuboi*, which has dissonant behaviour. This reduction is accompanied by a drop on D2 PSNR, according to Table 1, that doesn't surpass 1.1 dB in any case. Moreover, we observe in s_3 that for half of the test set, bit-rate reductions are accompanied by quality improvements.

5. CONCLUSION

This paper proposes a predictive coding approach for geometry-only point clouds. By using a learning-based module to predict blocks using information from its decoded neighbours, the method allows a notable size reduction of the bitstream without quality compromises at low bit-rates. Future work can focus on improving the current method by adding to the bitstream a residual representation of the difference between predicted and original blocks, or by implementing partitioning schemes better suited for predictive coding.

6. REFERENCES

[1] S. Schwarz *et al*., "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019.

[2] WG1, "First Call for Evidence on JPEG Pleno Point Cloud Coding," ISO/IEC JTC1/SC29/WG1 Doc. N86013, Jan. 2020.

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

[4] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "Octree-based progressive geometry coding of point clouds," in *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, 2006, pp. 103–110.

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

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

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

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

[9] L. Li, Z. Li, S. Liu, and H. Li, "Efficient projected frame padding for video-based point cloud compression," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[10] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li, "Advanced 3D motion prediction for video-based dynamic point cloud compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 289–302, 2020.

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

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

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

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

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

[16] J. Wang *et al.*, "Learned point cloud geometry compression," *arXiv:1909.12037*, 2019.

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

[18] E. Alexiou, K. Tung, and T. Ebrahimi, "Towards neural network approaches for point cloud compression," in *Applications of Digital Image Processing XLIII*. 2020, vol. 11510, pp. 18 – 37, SPIE.

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

[20] B. Yang *et al.*, "3D Object Reconstruction from a Single Depth View with Adversarial Learning," *arXiv:1708.07969*, 2017.

[21] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.