

Learning residual coding for point clouds

Davi Lazzarotto and Touradj Ebrahimi

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

ABSTRACT

Recent advancements in acquisition of three-dimensional models have been increasingly drawing attention to imaging modalities based on the plenoptic representations, such as light fields and point clouds. Since point cloud models can often contain millions of points, each including both geometric positions and associated attributes, efficient compression schemes are needed to enable transmission and storage of this type of media. In this paper, we present a detachable learning-based residual module for point cloud compression that allows for efficient scalable coding. Our proposed method is able to learn the encoding of residuals in any layered architecture, and is here implemented in a hybrid approach using both TriSoup and Octree modules from the G-PCC standard as its base layer. Results indicate that the proposed method can achieve performance gains in terms of rate-distortion when compared to both base layer alone, which is demonstrated both through objective metrics and subjective perception of quality in a rate-distortion framework. The source code of the proposed codec can be found at <https://github.com/mmospg/learned-residual-pcc>.

Keywords: Point cloud, Residual coding, Learning-based compression

1. INTRODUCTION

The use of digital media is rapidly growing, mainly due to the advent of the Internet and more recently the popularization of social network platforms. Since the majority of users access the data from devices with flat screens such as computer monitors or smartphones, two-dimensional media have been the norm for representing visual information. However, the recent introduction and development of devices such as the Head Mounted Displays (HMD) and of applications such as Virtual, Augmented and Mixed Reality (VR/AR/MR) have opened the possibility for new ways of visualization of virtual objects. This scenario is stimulating the research and development of imaging modalities based on the plenoptic function, such as light fields and point clouds. The interest in the latter is also increased by the easier access to acquisition technologies such as LiDAR and depth sensors, which are now becoming available in consumer devices such as smart phones.

Point clouds are a media format that represent three-dimensional objects by a set of points spanning throughout the space. Each point is defined by its coordinates (x, y, z) and can contain multiple associated attributes such as color, normal vectors, reflectance and gloss. Since point clouds can represent anything from small objects to large outdoor scenes, the total number of points can become as high as millions or even billions. For many applications, storing and transmitting such a huge amount of raw data would be infeasible, thus exposing the need for effective solutions for compression. In the last years, research in the field of point cloud compression has been on the rise, with standardization committees such as JPEG and MPEG following this trend by creating initiatives in order to develop interoperable coding standards.

Among the multiple solutions for point cloud compression found in the literature, algorithms capable of compressing geometry generally employ different mechanisms than those encoding associated attributes. Octrees constitute a very common data structure employed by the former category, which recursively partitions the space into eight non overlapping cubes indicating its occupancy state and finally storing this information in a tree structure. Other solutions project the patches of a point cloud model into multiple views and encode the projected depth and color maps as two-dimensional images, which are later used to reconstruct the geometry and attributes, respectively.

Further author information: (Send correspondence to the authors)

E-mail: davi.nachtigalllazzarotto@epfl.ch, touradj.ebrahimi@epfl.ch

Lately, algorithms based on deep-learning have been rapidly gaining attention, already achieving performances comparable, if not even superior, to the state of the art. Such technological innovations follow the tendency previously initiated on the field of image compression, which are reporting greater performance when compared to well established codecs resulted from decades of research and development. These learning-based point cloud coding algorithms are mostly based on autoencoder architectures that operate either directly on the point domain or on occupancy maps obtained after voxelization.

However, the majority of compression methods belonging to this category do not allow for progressive coding, i.e. partially decoding the bitstream in order to get a lower quality version of the point cloud, which is then refined as the rest of the compressed bitstream is parsed. Given that scalability is an important requirement in many use cases for point clouds,¹ developing a scalable solution is an important step for standardisation activities aiming to adopt learning-based compression algorithms. Layered frameworks are a solution that enable to achieve this goal, being composed by one base layer that generates a compressed bitstream with lower bitrate and one or more refining layers that enhance the quality of the decoded model by adding a residual representation of the difference between the original and the model decoded by the previous layer.

In this paper, we propose a learning-based residual coding algorithm for point cloud geometry compression. Our method generates, at the encoder stage, a compressed residual representation of the difference between the original point cloud and its reconstructed version after being distorted by another compression algorithm. This representation is then used at decoder stage to refine the quality of the distorted point cloud. The algorithm described in this study can be associated with any point cloud compression method in a two-layer framework. Moreover, it can also be combined with other residual coding methods to provide multiple layers of scalability. In this study, we implement it combined with both geometry coding modules from the G-PCC standard, namely Octree and TriSoup, in order to exploit the advantages of conventional and learning-based methods in a hybrid framework. Results demonstrate that our method is able to achieve better rate-distortion performance than the base layer codec employed in standalone mode.

The next sections of this paper are organized with the following structure: first a selection of works related to point cloud compression and residual coding is presented and discussed. Then, our proposed method for residual coding is thoroughly described. We later present the obtained results in terms of objective quality metrics and display renderings of the test models for subjective evaluation, discussing the advantages of our approach. We finish by drawing final conclusions about our study and pointing to possible direction for future works.

2. RELATED WORK

Several different approaches for point cloud compression have been discussed in the literature. Octrees are an example of commonly used data structure for encoding point cloud geometry, as explored by Schnabel et al.² and Huang et al.³ in early studies that incorporated prediction algorithms as an attempt to achieve high compression performance. Later works, such as the studies from Kammerl et al.⁴ and Mekuria et al.,⁵ preferred a low-complexity version by dismissing predictive coding in order to enable faster computation for real-time performance. The former compression algorithm became popular after being incorporated by the widely used Point Cloud Library (PCL).⁶ Mekuria et al.⁵ also mapped the color attributes in a two-dimensional grid by traversing the octree in depth-first order, which were then compressed using JPEG image coding.

Taking these geometry encoding algorithms as baseline, Dricot et al.⁷ proposed a method to encode leaf nodes with planar primitives. Similarly, Oliveira et al.⁸ built a two-layer framework using the PCL implementation⁶ as the base layer and performed graph-based enhancement to refine the quality and allow for scalable coding. Krivokuća et al.⁹ proposed a new approach by modeling the point cloud geometry as a volumetric function decomposed into a tri-linear B-spline wavelet basis.

Multiple works have been proposed to encode color attributes of point clouds using graphs. The idea was first proposed by Zhang et al.,¹⁰ which decomposed the point cloud into blocks and used the Graph Fourier Transform (GFT) to locally decorrelate the attributes. Later, Shao et al.¹¹ adopted a partitioning method based on the kD-tree, which avoided the creation of sparse blocks, and used off-line data for the optimization of parameters used to obtain the graph weights. Xu et al.¹² used yet another partitioning method based on K-means clustering, and proposed the NWGFT to compute the weighted adjacency matrix using the angular

similarity between estimated normal vectors. The author extended this method on a later study¹³ where the generalized Laplacian was leveraged to decorrelate the input signal in spatio-temporal graphs, which served to build an inter-prediction scheme proposed for attribute compression of dynamic point clouds.

On the other hand, the work from Queiroz et al.¹⁴ opted for a different technique with lower computational complexity and proposed a region adaptive hierarchical transform (RAHT) for texture coding. This solution was included in the MPEG G-PCC standard,¹⁵ which relies on an octree-based algorithm for geometry compression and an optional module that represents leaf nodes as triangular primitives. The same standardisation group became also interested in the projection-based algorithm presented by Mammou et al.,¹⁶ where projected patches are assembled into a video sequence. This work essentially established the basis for the MPEG V-PCC standard,¹⁷ which employs the HEVC codec to encode geometry and texture of the generated projections.

Recently, learning-based techniques have been leveraged for compression of point cloud geometry and color. An early study was introduced by Guarda et al.¹⁸ where a convolutional autoencoder architecture was employed for compressing point cloud geometry. The voxelized input point cloud was partitioned into blocks with fixed size and translated into occupancy maps before feeding a neural network model. The architecture was further enhanced by the addition of a variational autoencoder capturing statistical variances of the latent features¹⁹ and two distinct methods allowing scalable coding by progressively obtaining higher resolution²⁰ or better quality.²¹ The author also proposed a loss function to capture spatial context during the training process²² and an adaptive framework that optimizes rate-distortion performance.²³

Adopting a similar approach, Quach et al.²⁴ built a simple autoencoder architecture which was improved with a variational autoencoder, deeper transforms and optimal thresholding of the output probability map in a later work.²⁵ The model from his first study²⁴ was also expanded by Alexiou et al.,²⁶ which studied the implementation of the framework for coding geometry and color both sequentially and simultaneously. Wang et al.²⁷ incorporated Voxception ResNet Blocks (VRN)²⁸ to the network, leveraging the power of residual blocks. The same authors further refined and developed this work by employing sparse convolutional layers in a multiscale framework,²⁹ reporting increased performance.

Adopting an alternative architecture, both Yan et. al.³⁰ and Huang et. al.³¹ implemented architectures operating directly in the point domain without the need for voxelization. Their employed architecture was however only evaluated using small scale point clouds. Wen et al.³² proposed an adaptive octree-based partition algorithm allowing for a point-based architecture to compress high resolution models, while Wiesmann et al.³³ adopted kernel point convolutions to compress point clouds obtained by LiDAR scans. Biswas et al.⁷ followed a different method and learned a conditional entropy model to compress geometry and intensity values obtained by LiDAR sweeps. Finally, the VoxelDNN was proposed by Nguyen et al.³⁴ to learn a probability distribution used for entropy coding, achieving lossless coding. The same author later proposed a multiscale algorithm³⁵ that models voxel occupancy at a coarse-to-fine order, achieving faster encoding times when compared to the previous voxel-by-voxel approach.

Among the aforementioned learning-based compression methods, the two studies from Guarda et al.^{20,21} are the only to focus on scalability, which is achieved by gradually increasing the resolution of the input point cloud at the encoder side or the amount of latent channels decoded. Adopting a different approach, Li et al.³⁶ proposed an end-to-end learned scalable solution for 2D images including a residual coding module. Lee et al.³⁷ employed a learning-based residual module as well, but integrated it with the VVC codec as the base layer in a hybrid framework. In this work, we take inspiration from the architecture proposed by³⁷ to build a detachable learning-based residual coding module for point clouds. Since our residual module can be included into any layered framework, we provide an implementation of our algorithm integrated with both geometry coding modules from the G-PCC standard, namely TriSoup and Octree.

3. PROPOSED METHOD

3.1 Layered architecture

The solution implemented in our study is composed by two coding layers: the base layer, which generates an initial compact representation of a point cloud, and a residual module, that enhances the quality of the decoded model at the expense of added bitrate. The latter receives the point cloud model distorted by the previous

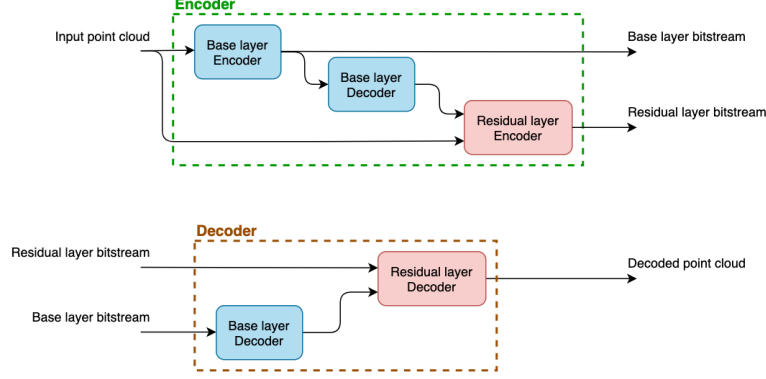


Figure 1. High level block diagram of the implemented layered architecture.

layer as well as the original input point cloud at encoder stage to generate the residual representation, which is concatenated with the base layer bitstream. At decoder stage, these residuals are combined with the point cloud decoded by the base layer in order to produce a reconstructed version of the point cloud with less degradation. A diagram of the implemented solution is presented at Figure 1.

3.2 Residual module

The residual coding module consists in a neural network with autoencoder architecture composed by stacked three-dimensional convolutional layers. Due to memory constraints, the module is not able to operate directly on the entire point cloud. Instead, both point clouds, i.e. the original and the point cloud produced by the base layer, are divided into non-overlapping cubic blocks with fixed size. Each block also goes through a process of voxelization before being fed into the convolutional layers, i.e. converting the coordinates of each point quantized with a constant step value. A regular volumetric grid is then fit to the block and the value 1 is assigned to the indexes of the grid occupied by a point, while the value 0 is assigned to empty positions. This generated data structures is referred to as an occupancy map, while the small cubic subdivisions associated to each index of the grid are denominated voxels. The architecture of the proposed residual module is presented at Figure 2, with an overall design inspired on the work from³⁷ for two-dimensional images.

The encoder stage of the proposed residual module receives an input block from the original point cloud x_{ref} and the corresponding block from the distorted point cloud x_{dist} . Both blocks are concatenated before feeding a sequence of three convolutional layers with stride 2 that reduce their dimension by a factor of 8. The statistical variance of each element of the generated feature block is then estimated by an added autoencoder architecture that generates a hyperprior,³⁸ which is encoded as side information. This hyperprior is used for both the entropy encoding and decoding of the quantized main feature block, and it is itself also entropy coded based on a factorized probability distribution estimated during training. Note that the hyper synthesis block, which is formally included at the decoder in our diagram, is needed to compute the scale indexes used for the entropy coding of the main feature block, and must therefore be present at both encoding and decoding stages.

The two bitstreams, generated after the entropy coding of both the hyperprior and the main feature block, are concatenated with an overhead of two bytes each indicating their individual lengths. They are also encapsulated with a sequence of three indexes associated to the block position within the point cloud. Finally, after compressing all the blocks of a point cloud model, the bitstreams of all blocks are concatenated together to compose the compressed representation of that model.

At decoder stage, the compressed hyperprior is entropy decoded and used to obtain the main feature block. This block then goes through three transposed convolutional layers that generate a latent residual representation with 32 channels and the same dimensions as the inputs x_{ref} and x_{dist} . This residual block is concatenated with the distorted block x_{dist} at the refiner, where it feeds a sequence of seven convolutional layers using ReLU activation functions that finally output the refined block $x_{refined}$. The last layer of the refiner is exceptionally composed by only one filter and employs the sigmoid activation function, in order to generate a block with the

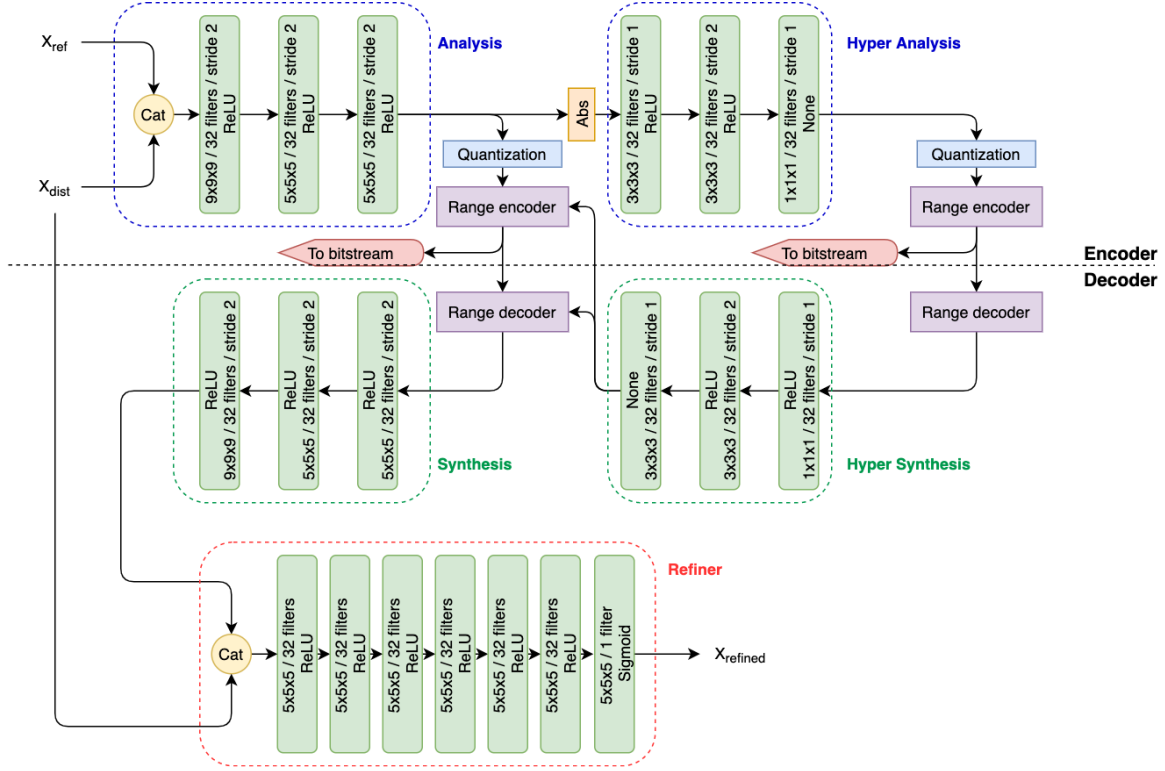


Figure 2. Block diagram of the proposed residual coding module.

same number of channels as the input and values between 0 and 1 that correspond to the estimated probability of that voxel being occupied. These probability values are then rounded to 0 or 1 using 0.5 as a threshold value, and the obtained occupancy map is finally translated back to a coordinate list corresponding to the occupied voxels. After the decoder of the residual module is run for all blocks, the point cloud is merged in a process that places each block at its respective index indicated in the bitstream.

3.3 Loss function

As an inherent condition in all lossy compression methods, this residual module is trained to solve a rate-distortion optimization problem. This problem is embedded in the loss function, which consists in the weighted sum of two terms R and D that estimate, respectively, the rate and the distortion, as in

$$loss = R + \lambda \cdot D, \quad (1)$$

with the Lagrange multiplier λ controlling the rate-distortion trade-off. The rate is estimated for each block as in the original work from Ballé et al.³⁸ for both the main bitstream and the compressed hyperprior, which are added to result in the term R . In order to allow for optimization using gradient descent methods, the quantization operator is replaced by additive uniform noise during training, following an earlier work from Ballé et al.³⁹ For the computation of distortion term, we consider the occupancy probability estimation as a classification problem that is separately solved for each voxel. We then employ the focal loss⁴⁰ to estimate the classification error per voxel, as given by

$$FL(u, v) = \begin{cases} -\alpha \cdot (1 - v)^\gamma \cdot \log v & \text{if } u = 1 \\ -(1 - \alpha) \cdot v^\gamma \cdot \log(1 - v) & \text{if } u = 0, \end{cases} \quad (2)$$

with v representing the calculated occupancy probability and u denoting the ground truth value for the voxel. Since empty voxels are most often more abundant than occupied voxels, the hyperparameter α compensates this class imbalance by penalizing the misclassification of an occupied voxel more than that of an empty voxel. The parameter γ is on its turn responsible for favoring the correction of a wrongly classified voxel over the minimization of the error for accurate estimations. In order to compute the final D term, the focal loss for all individual voxels are pooled through the sum operator and divided by the number of occupied voxels for that block.

3.4 Dataset

The HighResGC dataset assembled by Alexiou et al.²⁶ was employed for training and testing purposes. This collection is composed by 44 high resolution point cloud models for training and 6 for testing, all voxelized with bit depths of 9 or 10. Since our residual coding module operates only in the geometry, the color values of the point clouds were ignored. A sample of the training set and the entire test set can be visualized at Figure 3 and 4, respectively.



Figure 3. Point cloud models sampled from the training dataset.

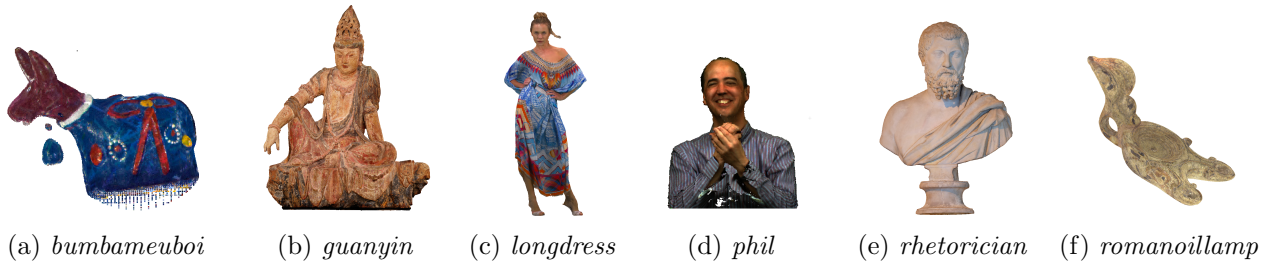


Figure 4. Point cloud models composing the test dataset.

3.5 Configuration

Since the proposed module is designed to adapt to the compression algorithm used in the base layer, we evaluated it combined with two distinct compression algorithms using different configurations. In particular, we select three different compression levels from TriSoup and two from Octree, in order to assess the performance of our module

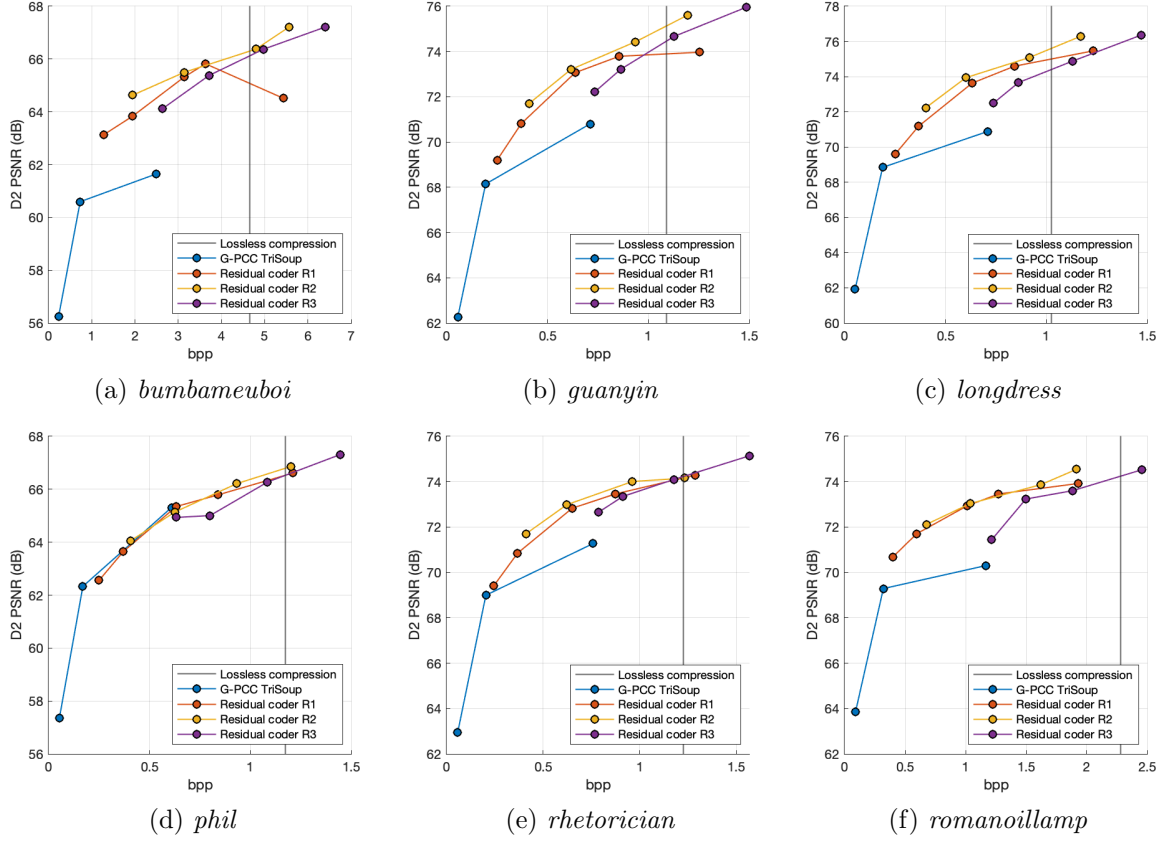


Figure 5. Rate-distortion plots with D2 PSNR metric for all employed λ values using TriSoup as the base layer.

to enhance the quality of models degraded with artifacts of different natures. We also trained the module using different values for λ , enabling different rate-distortion trade-offs for the compressed residual representation.

Following the JPEG Pleno Point Cloud Common Test Conditions document,⁴¹ the parameter *positionQuantizationScale* was set to 1 while *trisoup_node_size_log2* assumed the values of 2, 3 and 4 in order to obtain three different compression levels for TriSoup, which are here denominated R1, R2 and R3. On the other hand, *trisoup_node_size_log2* was kept constant at 0 for Octree, while the values of 0.5 and 0.75 were selected for *positionQuantizationScale* to obtain the levels R1 and R2 used for training and testing. We additionally compressed the test set with the latter parameter set to 0.875 for comparison purposes.

The entire training set was distorted using all five configurations. The residual module was then trained separately for each compression level, and multiple values of λ were employed. For TriSoup, the training was conducted using the values of 10, 25, 50, and 100, with the additional value of 5 selected for R1. With Octree, only the values of 10, 25 and 50 were recruited. All point clouds from the dataset were partitioned into blocks, and the block size assumed the values of 32 and 64 for training and for testing, respectively.

After training the residual model, all point clouds from the test set were distorted with the aforementioned compression levels from the base layer codecs Octree and TriSoup. The residual layer was then employed for quality enhancement, and the bitrate of the layered architecture was computed by adding the bitrates for both the base and the residual layers. In order to evaluate the distortion induced by these compression methods, we employed the point-to-plane metric⁴² with the PSNR version (D2 PSNR). The values for the normal vectors needed for this computation were estimated with plane fitting with 10 nearest neighbours, using the MeshLab v2020.06.⁴³

The residual module was trained using a selection of 10'000 blocks from the training set randomly sampled

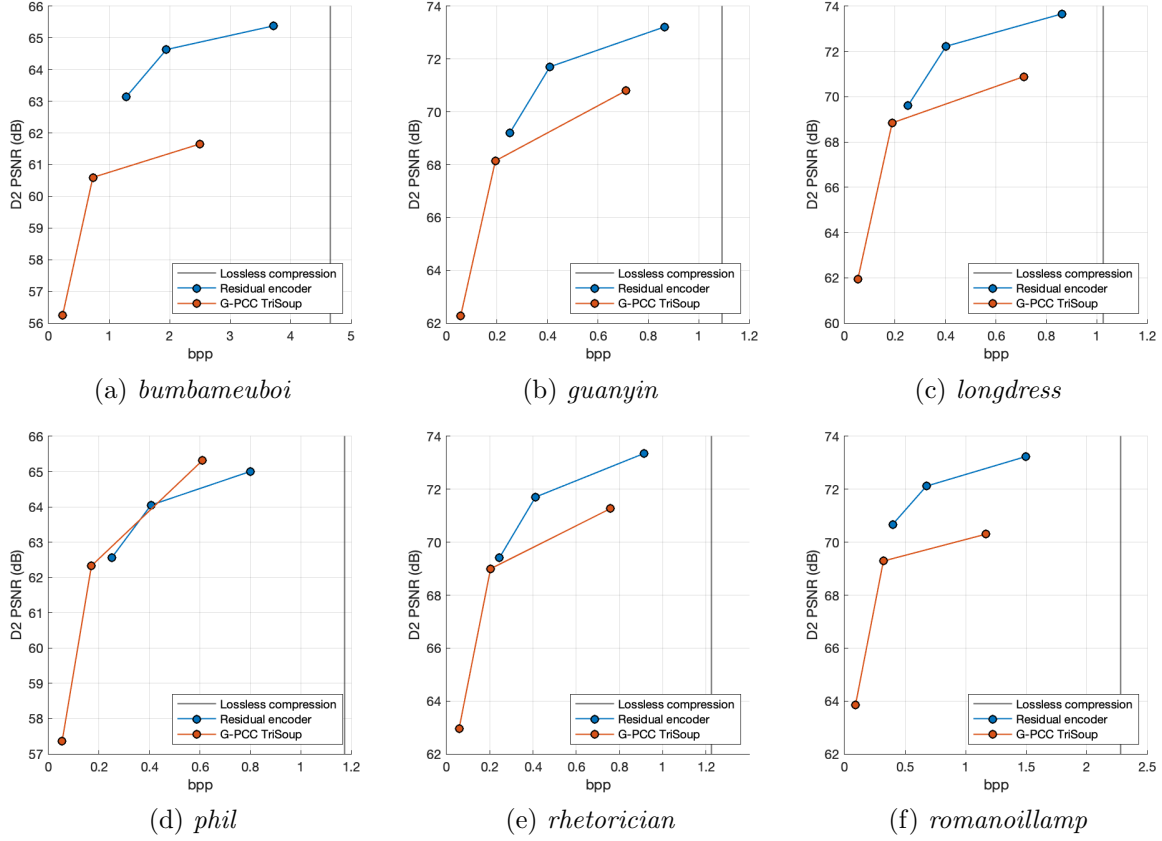


Figure 6. Rate-distortion plots with D2 PSNR metric for selected λ values using TriSoup as the base layer.

after discarding blocks with less than 500 points. As an exception, the training set contained only 3'000 blocks when Octree at the level R1 was used as a base layer, since the majority of the blocks distorted with this configuration had less points than the imposed threshold. The values of $\alpha = 0.9$ and $\gamma = 2$ were employed to compute the focal loss. The Adam Optimizer⁴⁴ was adopted with a learning rate of $1e-4$, and hyperparameter values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

4. EXPERIMENTAL RESULTS

4.1 Objective evaluation

The evaluation procedure described in the previous section yielded several pairs of (rate, distortion) values that were plotted separately for each point cloud from the test set. We start our analysis by presenting in Figure 5 the rate-distortion plots for both TriSoup and the proposed layered architecture using TriSoup as the base layer. We group the points corresponding to the layered architecture according to the compression level used for the base layer and link them in one curve for each level, resulting in three separate curves where our residual module is employed. We also report the bitrate corresponding to the lossless configuration of the G-PCC standard as a vertical line for each point cloud.

As expected, we observe that, at most of the evaluated points, the proposed residual module allows for quality increase at the expense of added bitrate, the exception being *phil* when using the compression level R3 with the two first values of λ . It is however clear that, for all other models of the test set, adding our residual results in less distortion at equivalent and even lower bitrate levels. For instance, it is worthy of mention that our module trained with $\lambda = 10$ with the compression level R2 produces higher D2 PSNR level than the base layer R3 level, at a lower bitrate, for all the test set except *phil*. We however observe that some modules trained with $\lambda = 50$

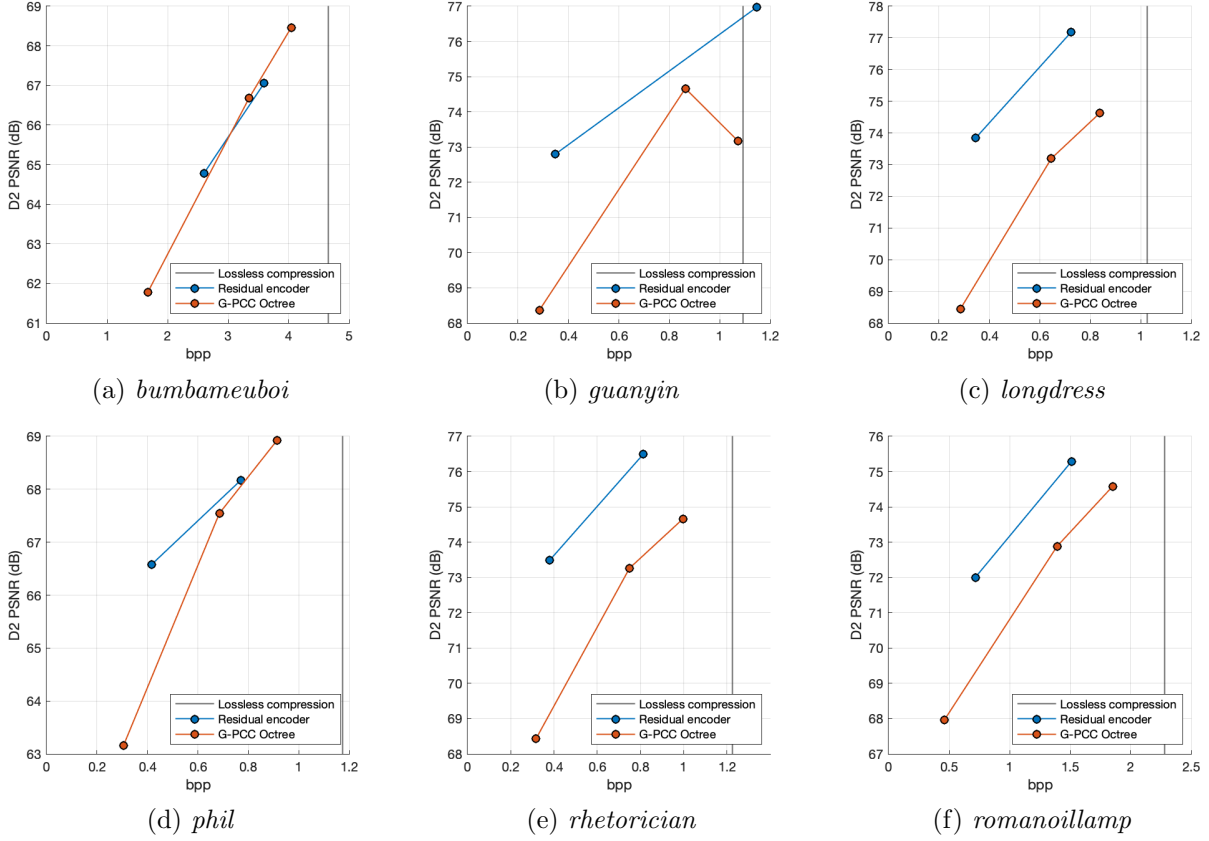


Figure 7. Rate-distortion plots with D2 PSNR metric for selected λ values using Octree as the base layer.

and $\lambda = 100$ result in bitrates going beyond the lossless threshold, invalidating all advantages from the proposed approach. In order to ignore these less interesting points and to make a more clear comparison between TriSoup and the proposed architecture, we select a single value of λ for each compression level. In particular, we recruit the values of 5, 10 and 25 for the levels R1, R2 and R3, respectively. The corresponding rate-distortion plots are presented in Figure 6.

We can clearly observe from the generated plots that our implemented layered architecture outperforms the base layer in terms of rate-distortion performance at the assessed levels. Even for *phil*, the worst model from the test set for our module, the first two points are aligned with the underlying curve of the base layer, while retaining the advantage of scalability. The remaining plots demonstrate that encoding the residuals between the distorted version and the original model is more beneficial than moving to the next compression level of the base layer when higher quality is desired.

Following our analysis, the same values were computed using the Octree module as the base layer. After careful analysis, the λ values of 10 and 50 were chosen for the compression levels R1 and R2, respectively. The corresponding rate-distortion plots can be observed in Figure 7. The obtained results allow us to draw similar conclusions for the Octree module. For most of the point clouds, we obtain higher quality representations at equivalent bitrate levels when employing the proposed residual module. The exceptions are *bumbameuboi* and *phil*, for which we obtain similar rate-distortion performance as the base layer while adding scalability as an extra feature. We also acknowledge the anomalous point obtained for *guanyin* for the level R2 of the base layer, where we observe a high bitrate than R3 while keeping lower quality. However, the residual module still behaves as expected in this situation, slightly increasing the bitrate while substantially enhancing the quality.






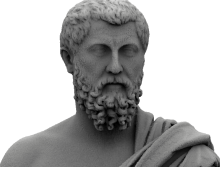
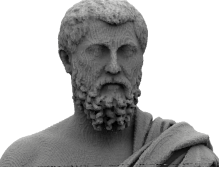
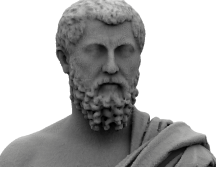
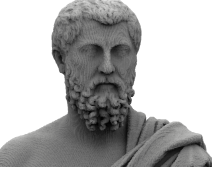
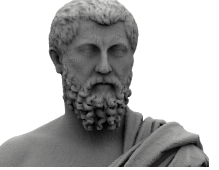
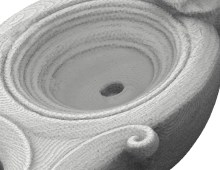
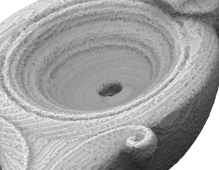
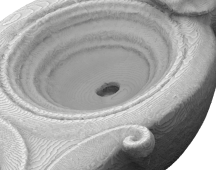
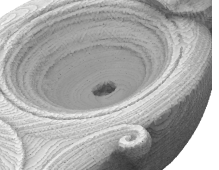

Original	TriSoup R3	Trisoup R2 & residual coding	Octree R2	Octree R1 & residual coding
 <i>longdress</i>	 0.711 bpp 70.89 dB	 0.403 bpp 72.23 dB	 0.646 bpp 73.19 dB	 0.346 bpp 73.84 dB
 <i>rhetorician</i>	 0.759 bpp 71.27 dB	 0.411 bpp 71.70 dB	 0.749 bpp 73.26 dB	 0.379 bpp 73.48 dB
 <i>romanoillamp</i>	 1.168 bpp 70.30 dB	 0.676 bpp 72.12 dB	 1.392 bpp 72.88 dB	 0.717 bpp 72.00 dB

Figure 8. Visual comparison of the compression performance of the evaluated compression methods.

4.2 Subjective evaluation

Even if the obtained results indicate that the implemented layered framework outperforms the base layer in terms of objective quality metrics, an equivalent enhancement on the subjective perception cannot be automatically derived. Indeed, recent studies have shown that frequently objective quality metrics do not correlate well with human opinion in the presence of artifacts caused by codecs of different natures.⁴⁵ We therefore provide rendered samples from the evaluated test set for visual inspection. Since the evaluated algorithms handle geometry-only point clouds, color values were attributed by simulating a light source and attributing a greyscale value computed as a function of the angle of incidence. This operation was performed by the PCV plugin from the CloudCompare⁴⁶ applied with default settings. The models were then rendered using fixed point size, which was adjusted in order to produce the appearance of a water-tight surface. Naturally, point clouds with more spaced out points were rendered with larger point size, which was determined individually for each model after expert viewing.

In Figure 8, we present the models *longdress*, *rhetorician* and *romanoillamp*. These models were compressed using Octree and Trisoup at compression levels R2 and R3, respectively, as well as with the layered framework at compression level R1 for Octree and R2 for Trisoup and previously selected values of λ . The Figure also includes the original uncompressed models at the left column. During the rendering process, the zoom and position of the camera were adjusted in order to allow for inspection of details while still portraying high level shapes of the point clouds.

A careful observation of the rendered models asserts conclusions drawn from the objective quality metrics, namely that the proposed layered architecture produces lower bitrates at equivalent or even higher quality levels. While the computation of triangular primitives for the point cloud on TriSoup can generate holes that are particularly visible at *longdress*, the exclusion of points with Octree is perceived as a coarser grid and hinders the reproduction of fine detail. At the assessed compression levels, these effects are not present when using the

proposed method, which reveals increased subjective performance at lower bitrates.

5. CONCLUSION

In this paper, we propose a detachable learning-based module for residual coding of point clouds. Among the different solutions proposed for compression of point cloud residuals in the literature, our work is the first to leverage deep learning techniques for this purpose. We implement our module in a hybrid architecture composed of two coding layers and employ two distinct conventional codecs as a base layer, namely the Octree and TriSoup module from the G-PCC standard. By exploring advantages from both conventional and data driven techniques while adopting a scalable solution, we obtain results that indicate that the proposed layered architecture including the residual coding module outperforms the base layer codecs, both in terms of objective metrics and subjective quality. In order to further investigate the possible contributions of this residual module, future work can implement a fully learned framework by adopting a learning-based solution as the base layer as well. Such an approach would constitute an end-to-end optimized alternative architecture for scalable coding. We can additionally propose the integration of our module with an intra or inter prediction algorithm,⁴⁷ used to encode the residuals between a prediction and a reference point cloud. Finally, future research could also concentrate on adapting this proposed architecture for compression of point cloud attributes.

ACKNOWLEDGMENTS

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

REFERENCES

- [1] Stuart Perry, “JPEG Pleno Point Cloud – Use Cases and Requirements v1.3.” ISO/IEC JTC1/SC29/WG1 Doc. N86012 (Jan. 2020).
- [2] Schnabel, R. and Klein, R., “Octree-based point-cloud compression,” in [*Symposium on Point-Based Graphics 2006*], (Jul. 2006).
- [3] 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).
- [4] Kammerl, J., Blodow, N., Rusu, R. B., Gedikli, S., Beetz, M., and Steinbach, E., “Real-time compression of point cloud streams,” in [*2012 IEEE International Conference on Robotics and Automation*], 778–785 (2012).
- [5] Mekuria, R., Blom, K., and Cesar, P., “Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video,” *IEEE Transactions on Circuits and Systems for Video Technology* **27**, 828–842 (Apr. 2017).
- [6] Rusu, R. B. and Cousins, S., “3d is here: Point cloud library (pcl),” in [*2011 IEEE International Conference on Robotics and Automation*], 1–4 (2011).
- [7] Dricot, A. and Ascenso, J., “Hybrid octree-plane point cloud geometry coding,” in [*2019 27th European Signal Processing Conference (EUSIPCO)*], 1–5 (2019).
- [8] 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).
- [9] Krivokuća, M., Chou, P. A., and Koroteev, M., “A volumetric approach to point cloud compression–part ii: Geometry compression,” *IEEE Transactions on Image Processing* **29**, 2217–2229 (2020).
- [10] Zhang, C., Florêncio, D., and Loop, C., “Point cloud attribute compression with graph transform,” in [*2014 IEEE International Conference on Image Processing (ICIP)*], 2066–2070 (2014).
- [11] Shao, Y., Zhang, Z., Li, Z., Fan, K., and Li, G., “Attribute compression of 3d point clouds using laplacian sparsity optimized graph transform,” in [*2017 IEEE Visual Communications and Image Processing (VCIP)*], (10 2017).
- [12] Xu, Y., Hu, W., Wang, S., Zhang, X., Wang, S., Ma, S., and Gao, W., “Cluster-based point cloud coding with normal weighted graph fourier transform,” in [*2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*], 1753–1757 (2018).

- [13] Xu, Y., Hu, W., Wang, S., Zhang, X., Wang, S., Ma, S., Guo, Z., and Gao, W., “Predictive generalized graph fourier transform for attribute compression of dynamic point clouds,” *IEEE Transactions on Circuits and Systems for Video Technology* **PP**, 1–1 (08 2020).
- [14] 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).
- [15] 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).
- [16] 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).
- [17] 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).
- [18] 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).
- [19] 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).
- [20] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., “Deep learning-based point cloud geometry coding with resolution scalability,” in *[2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)]*, 1–6 (2020).
- [21] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., “Point cloud geometry scalable coding with a single end-to-end deep learning model,” in *[2020 IEEE International Conference on Image Processing (ICIP)]*, 3354–3358 (2020).
- [22] Guarda, A., Rodrigues, N., and Pereira, F., “Neighborhood adaptive loss function for deep learning-based point cloud coding with implicit and explicit quantization,” *IEEE MultiMedia* , 1–1 (2020).
- [23] Guarda, A. F. R., Rodrigues, N. M. M., and Pereira, F., “Adaptive deep learning-based point cloud geometry coding,” *IEEE Journal of Selected Topics in Signal Processing* **15**(2), 415–430 (2021).
- [24] 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).
- [25] Quach, M., Valenzise, G., and Dufaux, F., “Improved Deep Point Cloud Geometry Compression,” in *[IEEE International Workshop on Multimedia Signal Processing (MMSP’2020)]*, (Sep. 2020).
- [26] Alexiou, E., Tung, K., and Ebrahimi, T., “Towards neural network approaches for point cloud compression,” in *[Applications of Digital Image Processing XLIII]*, 4 (08 2020).
- [27] Wang, J., Zhu, H., Liu, H., and Ma, Z., “Lossy point cloud geometry compression via end-to-end learning,” *IEEE Transactions on Circuits and Systems for Video Technology* , 1–1 (2021).
- [28] Brock, A., Lim, T., Ritchie, J., and Weston, N., “Generative and discriminative voxel modeling with convolutional neural networks,” *IEEE NeurIPS: 3D Deep Learning* **abs/1608.04236** (2016).
- [29] Wang, J., Ding, D., Li, Z., and Ma, Z., “Multiscale point cloud geometry compression,” (2020).
- [30] Yan, W., Shao, Y., Liu, S., Li, T. H., Li, Z., and Li, G., “Deep autoencoder-based lossy geometry compression for point clouds,” *CoRR* **abs/1905.03691** (2019).
- [31] Huang, T. and Liu, Y., “3d point cloud geometry compression on deep learning,” in *[Proceedings of the 27th ACM International Conference on Multimedia]*, MM ’19, 890–898, Association for Computing Machinery, New York, NY, USA (2019).
- [32] Wen, X., Wang, X., Hou, J., Ma, L., Zhou, Y., and Jiang, J., “Lossy geometry compression of 3d point cloud data via an adaptive octree-guided network,” in *[2020 IEEE International Conference on Multimedia and Expo (ICME)]*, 1–6 (2020).
- [33] Wiesmann, L., Milioto, A., Chen, X., Stachniss, C., and Behley, J., “Deep compression for dense point cloud maps,” *IEEE Robotics and Automation Letters* **6**(2), 2060–2067 (2021).
- [34] Nguyen, D., Quach, M., Valenzise, G., and Duhamel, P., “Learning-based lossless compression of 3d point cloud geometry,” in *[(ICASSP 2021) 2021 IEEE International Conference on Acoustics, Speech, and Signal Processing]*, (2021).

- [35] Nguyen, D. T., Quach, M., Valenzise, G., and Duhamel, P., “Multiscale deep context modeling for lossless point cloud geometry compression,” (2021).
- [36] Li, W., Sun, W., Zhao, Y., Yuan, Z., and Liu, Y., “Deep image compression with residual learning,” *Applied Sciences* **10**, 4023 (06 2020).
- [37] Lee, W.-C., Chang, C.-P., Peng, W.-H., and Hang, H.-M., “A hybrid layered image compressor with deep-learning technique,” in [2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)], 1–6 (2020).
- [38] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N., “Variational image compression with a scale hyperprior,” in [6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings], OpenReview.net (2018).
- [39] Ballé, J., Laparra, V., and Simoncelli, E. P., “End-to-end optimized image compression,” in [5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings], OpenReview.net (2017).
- [40] Lin, T., Goyal, P., Girshick, R., He, K., and Dollar, P., “Focal loss for dense object detection,” in [2017 IEEE International Conference on Computer Vision (ICCV)], (2017).
- [41] Stuart Perry, “JPEG Pleno Point Cloud Coding Common Test Conditions v3.5.” ISO/IEC JTC1/SC29/WG1 Doc. N90053 (Jan. 2021).
- [42] Tian, D., Ochimizu, H., Feng, C., Cohen, R., and Vetro, A., “Geometric distortion metrics for point cloud compression,” in [2017 IEEE International Conference on Image Processing (ICIP)], 3460–3464 (2017).
- [43] 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], (2008).
- [44] Kingma, D. P. and Ba, J., “Adam: A method for stochastic optimization,” (2017).
- [45] Lazzarotto, D., Alexiou, E., and Ebrahimi, T., “Benchmarking of objective quality metrics for point cloud compression,” in [2021 IEEE International Workshop on Multimedia Signal Processing (MMSP)], (Accepted in 2021).
- [46] “Cloudcompare (version 2.11) [gpl software],” (2021).
- [47] Lazzarotto, D., Alexiou, E., and Ebrahimi, T., “On block prediction for learning-based point cloud compression,” in [2021 IEEE International Conference on Image Processing (ICIP)], (Accepted in 2021).