ISO/IEC JTC 1/SC 29/WG 1

(ITU-T SG16)

Coding of Still Pictures

JBIG

Joint Bi-level Image Experts Group

JPEG

Joint Photographic Experts Group

TITLE: **IT/IST/IPLeiria Response to the Call for Evidence on JPEG Pleno Point Cloud Coding**

AUTHORS: André F. R. Guarda[1], Nuno M. M. Rodrigues[2], Fernando Pereira[1]

EMAILS: andre.guarda@lx.it.pt, nuno.rodrigues@co.it.pt, fp@lx.it.pt

AFFILIATION: [1] Instituto Superior Técnico - Universidade de Lisboa, and Instituto de Telecomunicações, Lisbon, Portugal; [2] ESTG, Politécnico de Leiria and Instituto de Telecomunicações, Leiria, Portugal

Table of Contents

Summary

This document proposes two scalable point cloud (PC) geometry codecs, submitted to the JPEG Call for Evidence on Point Cloud Coding (PCC) [1], notably targeting two different types of scalability:

1) **Resolution Scalable Deep Learning-based Point Cloud Geometry Coding (RS-DLPCC)** [2]: This codec provides scalability on the number of points, called here resolution scalability; since the scalable layers are independently coded, this codec also offers multiple description coding, i.e. all layers by themselves offer useful PC reconstructions, which is a very interesting feature for error resilience, e.g. limiting the effect of packet losses in specific layers, etc.

2) **Quality Scalable Deep Learning-based Point Cloud Geometry Coding (QS-DLPCC)** [3]: This codec provides quality scalability using layer dependent coding, meaning that decoding a layer requires decoding the previous layers as well, thus not offering multiple description coding.

The proposed scalable codecs are based on recent developments in deep learning-based PC geometry coding (ADL-PCC) [4], and offer the key functionalities targeted by the JPEG Call for Evidence, notably number of points or resolution scalability, quality scalability, and spatial random access.

The proposed RS-DLPCC and QS-DLPCC coding solutions offer a compression efficiency that is rather competitive with the MPEG G-PCC standard [5], whereas the non-scalable version (ADL-PCC) of the proposed codecs is able to achieve significant RD performance gains over the G-PCC standard. Nevertheless, since these are some of the first (if not the first) deep learning-based scalable geometry coding solutions in the literature, the proposed scalable codecs shall be regarded more as a proof of concept as it is clear that substantial performance improvements may be expected in the future.

The proposed RS-DLPCC solution has been recently published at the IEEE International Workshop on Multimedia Signal Processing (MMSP'2020) and should be referenced as:

A. F. R. Guarda, N. M. M. Rodrigues, F. Pereira, "Deep Learning-based Point Cloud Geometry Coding with Resolution Scalability", IEEE International Workshop on Multimedia Signal Processing (MMSP'2020), Tampere, Finland, September 2020.

As for the proposed QS-DLPCC solution, it has been recently published at the IEEE International Conference on Image Processing (ICIP'2020) and should be referenced as:

A. F. R. Guarda, N. M. M. Rodrigues, F. Pereira, "Point Cloud Geometry Scalable Coding With a Single End-to-End Deep Learning Model", IEEE International Conference on Image Processing (ICIP'2020), Abu Dhabi, United Arab Emirates, October 2020.

The non-scalable ADL-PCC solution has been recently submitted to the IEEE Journal of Selected Topics in Signal Processing (J-STSP) as:

A. F. R. Guarda, N. M. M. Rodrigues, F. Pereira, "Adaptive Deep Learning-based Point Cloud Geometry Coding", submitted to IEEE Journal of Selected Topics in Signal Processing (J-STSP).

This proposal is focused on geometry coding only. However, for the purpose of subjective evaluation as

defined in the JPEG Call for Evidence, G-PCC coded color has been added to the RS-DLPCC decoded geometry after appropriate recoloring.

PART I – Non-Scalable PC Geometry Codecs

To better understand the proposed scalable codecs, this first part of the document provides a description of the non-scalable deep learning PC geometry coding solution which serves as the basis for RS-DLPCC and QS-DLPCC. A brief description of the current state-of-the-art, non-scalable coded (Adaptive DL-PCC, or ADL-PCC) will also be presented and used later as a performance benchmark.

1. DL-based PC Geometry Coding

This first section describes the basic non-scalable deep learning-based point cloud geometry coding (DL-PCC) solution which will be the core of the following DL-based codecs. The proposed scalable codecs (RS-DLPCC and QS-DLPCC), described in detail in later sections, are extensions of this solution, in particular by using different variations of the DL coding model in order to provide scalability.

1.1 High-level Description

The overall architecture of the DL-PCC codec is presented in Fig. 1, with the various modules briefly described as follows:

- **Encoder**:
    - **PC Block Partitioning**: The PC is divided into disjoint 3D blocks of the target size, which are coded separately for random access;
    - **Deep Learning (DL)-based Block Encoding**: Each block is encoded with an end-to-end DL coding model, which uses the architecture presented in Fig. 2. It can be compared to a typical transform coding approach, using in this case a convolutional autoencoder to learn a non-linear transform. The transform generates a set of coefficients, referred to as the *latent representation*, which are then quantized in the form of a simple rounding, and finally entropy coded. A learned adaptive entropy model is used, estimated via a variational autoencoder [6];
- **Decoder**:
    - **DL-based Block Decoding**: Blocks are decoded using the decoder counterpart of the DL-based block encoder mentioned before (see Fig. 2);
    - **PC Block Merging**: The decoded blocks are merged to reconstruct the full PC.

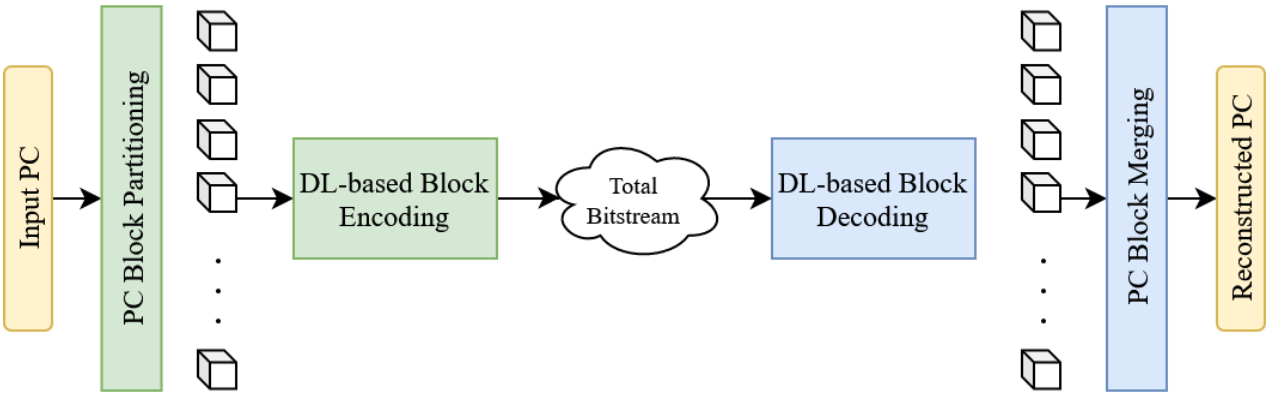


Fig. 1. Overall architecture of the non-scalable DL-PCC codec.

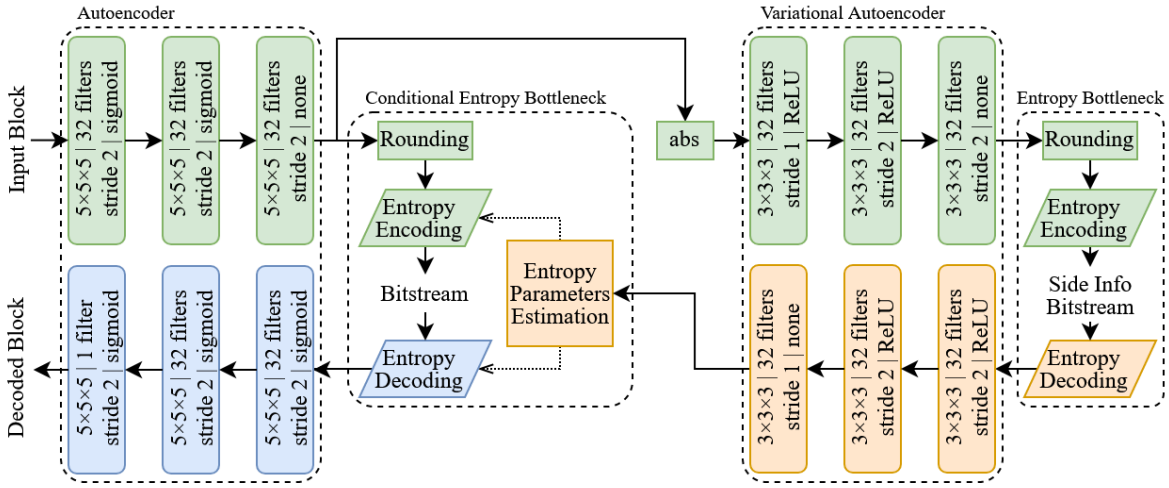


Fig. 2. End-to-end deep DL-based coding model architecture. Green blocks are encoder-only, blue blocks are decoder-only, and orange blocks are both encoder and decoder [2].

1.2 Detailed Description of each Architecture Module

Each of the modules presented in Fig. 1 is described here in more detail.

1.2.1 PC Block Partitioning/Merging

Before encoding, the PC geometry (3D coordinates) is converted into a binary, voxel-based 3D block representation, where voxels may be occupied or not; in practice, a '1' signals a filled voxel while a '0' signals an empty voxel. This voxel-based representation defines a regular structure that allows the use of convolutional neural networks (CNNs), similarly to image and video data; an example is shown in Fig. 3.

x	y	z
3	0	0
3	0	1
3	0	2
3	0	3
3	1	2
3	2	1
3	3	0
3	3	1
3	3	2
3	3	3

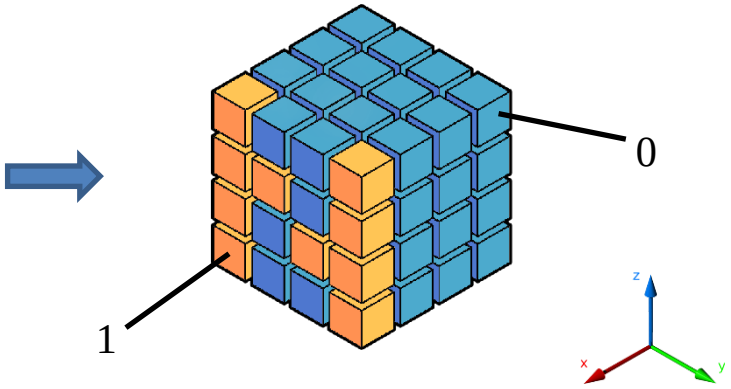


Fig. 3. Example of conversion from 3D PC coordinates to a 3D block of binary voxels.

Considering this new representation, a straightforward way to organize a PC is to divide it into disjoint blocks of a specific size, in this case 64×64×64, which can then be coded separately with a DL coding model. The position of each single 3D block is transmitted to the decoder.

At the decoder side, given the decoded blocks and their position, the full PC is reconstructed by merging the blocks accordingly.

1.2.2 DL-based Block Encoding and Decoding

This section presents the adopted DL-based PC geometry coding solution acting at block-level. Based on successful CNN architectures for image coding [6], the adopted end-to-end DL coding model is presented in Fig. 2. The full architecture can be divided into four main coding stages as follows:

1. **Autoencoder** – The convolutional autoencoder (AE) transforms the input block into a latent representation with lower dimensionality, in a way comparable to the transform coding stage in traditional image coding. This latent representation can be considered the transform coefficients, and consists of multiple feature maps, which number depends on the chosen number of filters for the convolutional layers to learn. The AE consists of six 3D convolutional layers: the first three layers, corresponding to the encoder side, apply the so-called *direct transform*; likewise, the last three layers, corresponding to the decoder side, apply the so-called *inverse transform*. Each convolutional layer consists of 32 filters with 5×5×5 support, resulting in 520000 weights plus 129 biases, totaling 520129 trainable parameters.

2. **Conditional Entropy Bottleneck –** A conditional entropy bottleneck layer from the Tensorflow compression library [6] is used to quantize (using a simple rounding operation) and then entropy code the block latent representation. This bottleneck uses a Gaussian scale mixture conditioned on a hyperprior as the entropy coding model. During training, this layer estimates the entropy of the latent representation according to the entropy coding model, which is used for the rate-distortion (RD) optimization process. At coding time, a range encoder is used to create the block bitstream.

3. **Variational Autoencoder –** A variational autoencoder (VAE) is used to capture possible structure information still present in the block latent representation, which is then used as a hyperprior for the conditional entropy bottleneck. This way, the entropy coding model parameters can be more accurately estimated and adapted for each coded block. In this process, the VAE generates its own latent representation, which also has to be coded and transmitted in the bitstream as additional side information to the decoder, so that the entropy coding model parameters can be replicated at the decoder. The VAE has a similar design to the AE, with each convolutional layer consisting of 32 filters with 3×3×3 support, resulting in 165888 weights plus 160 biases, totaling 166048 trainable parameters.

4. **Entropy Bottleneck –** Similar to the conditional entropy bottleneck, this entropy bottleneck quantizes and entropy codes the VAE latent representation. However, it uses a fixed entropy coding model for all blocks instead of an adaptive one, which is learned during training. To learn this entropy coding model, 1472 trainable parameters are used. As all the components of the end-to-end DL coding model are jointly trained, the additional side information rate is compensated by reducing the rate associated with the latents, thus optimizing the overall RD performance.

The total number of trainable parameters in the full DL coding model is 687649.

At the decoder side, each block is decoded with the DL coding model shown in Fig. 2. The "Side Info Bitstream" is decoded to generate the entropy coding model parameters used for the current block, so that its "Bitstream" can finally be decoded.

1.3 DL Coding Model Training

In order to achieve efficient compression performance, the DL coding model from Fig. 2 was trained by minimizing a loss function that considers both the distortion of each decoded block as well as its estimated coding rate. For this purpose, the loss function follows a traditional formulation involving a Lagrangian multiplier, λ, given by:

$$Loss\ Function = Distortion + \lambda \times Coding\ rate. \quad (1)$$

DL-based codecs typically require training a different DL coding model for each target RD point, which is accomplished by varying the λ parameter in Equation (1).

As described in Section 1.2.1, a voxel-based representation was adopted to process the PCs. Thus, for the DL coding model, the input data is a block of binary voxels, and the decoded data represents a probability score between '0' and '1' for each voxel, i.e. the probability of each voxel being filled. With this in mind, the block distortion is measured at voxel level as a binary classification error using the so-called *Focal Loss* (FL) [7], defined as follows:

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

where u is the original voxel binary value and v is the corresponding decoded voxel probability score. A weight parameter, α, is used to control the class imbalance effect since the number of '0' valued voxels in a block is vastly superior to the number of '1' valued voxels. The parameter γ allows increasing the importance

of correcting misclassified voxels in relation to improving the classification score of already correct voxels; γ=2 was found to be appropriate.

The DL coding model is trained using a selection of static PCs from the MPEG Point Cloud Compression (PCC) dataset [8] (naturally, always different from the test PCs). The selected PCs were down-sampled to a precision of 9 or 10 bit, according to the MPEG PCC Common Test Conditions (CTC) [8], and then partitioned into blocks of size 64×64×64, as described in Section 1.2.13.2.1. The blocks with less than 500 'filled' voxels have been removed in order to avoid the blocks with such low point count which could negatively affect the training, due to the increased class imbalance. Overall, 6000 blocks were used in the training process.

Implementation and training are done in Tensorflow version 1.14, using the Tensorflow Compression library [6] version 1.2. For training, the Adam algorithm [9] is used with a learning rate of 10^{-4} and minibatches of 8 blocks during 10^6 steps.

2. Adaptive DL-based PC Geometry Coding

The basic DL-PCC solution has been further improved by introducing a DL coding model selection mechanism allowing the codec to adapt to different PC characteristics by using multiple trained DL coding models. This solution, named adaptive DL-PCC (ADL-PCC), will serve as benchmark to assess the current cost of scalability for the proposed scalable codecs, thus demonstrating the potential of DL-based coding.

2.1 High-level Description

In this non-scalable ADL-PCC solution, multiple DL coding models have been trained using the same architecture of Fig. 2, but with training parameters suited for different PC characteristics, e.g. density. At coding time, each block is encoded and decoded with each DL coding model, so that the best model can be chosen for each block. The overall ADL-PCC architecture is presented in Fig. 4, with the various modules briefly described as follows:

- **Encoder**:
    - **PC Block Partitioning**: The PC is divided into disjoint 3D blocks of the target size, which are coded separately for random access;
    - **DL-based Block Encoding and Reconstruction**: Each block is encoded with several available DL coding models;
    - **DL Coding Model Selection**: After encoding and decoding the block for each DL coding model, the reconstructions are evaluated to select the model which produces the best compression performance, i.e. adapts better to the specific block characteristics; the selected model is signaled in the bitstream to the decoder;
- **Decoder**:

- **DL-based Block Decoding/Reconstruction**: Blocks are decoded using the corresponding/signaled DL coding models, with the decoder counterpart of the DL-based block encoder mentioned before;

- **PC Block Merging**: The decoded blocks are merged to reconstruct the PC.

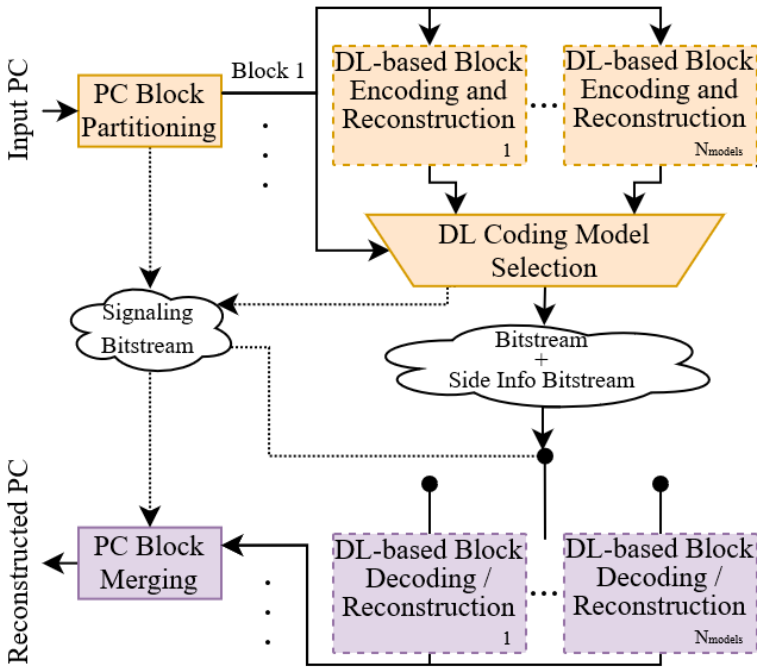


Fig. 4. Overall architecture of the non-scalable ADL-PCC codec [4].

The main ADL-PCC novelty when compared to DL-PCC is the module DL Coding Model Selection, which is described in the following subsection.

2.2 DL Coding Model Selection

Considering a given number of trained DL coding models N_{models}, the characteristics of each PC block are addressed in an adaptive way, by selecting the best from the available DL coding models, with the following procedure:

- **Block Reconstruction**: The block under consideration is coded with all the available N_{models} DL coding models; the various block reconstructions are then obtained and converted to PC coordinates for quality assessment;

- **Rate and Distortion Assessment**: The distortion between the original and the decoded blocks is assessed with a PC objective distortion metric, e.g. the point-to-point distance (D1); the number of bits per input point required by each DL coding model (rate) is determined;

- **RD-based DL Coding Model Selection**: The DL coding model providing the block reconstruction with the lowest RD cost is selected;

- **Selected DL Coding Model Signaling**: The selected DL coding model for each block is signaled to the decoder by using a dedicated symbol in the Signaling Bitstream. This symbol stream is coded with an adaptive arithmetic codec, using adaptive probability tables, which are updated as each block is processed.

2.3 DL Coding Model Training

All the DL coding models used by ADL-PCC were trained in the same conditions as described in Section 1.3 for DL-PCC, with the exception of a few parameters. Five RD points were obtained by training models with λ=500, 900, 1500, 5000 and 20000. In addition, for each RD point, multiple DL coding models were trained for different PC characteristics. With N_{models} =5, five models were trained for each RD point for α=0.5, 0.6, 0.7, 0.8 and 0.9. Larger α values are more suited to sparse PCs, while smaller α values perform better for denser PCs.

PART II – Scalable PC Geometry Codecs

The second part of this document describes the core of this proposal, notably the scalable PC geometry codecs. Both proposed scalable codecs are based on the basic, non-scalable DL-PCC solution presented above.

3. Resolution Scalable DL-based PC Geometry Coding

This section describes the first proposed scalable PC geometry codec (for additional details, please refer to [2]). Based on the non-scalable DL-PCC codec, the RS-DLPCC solution offers scalability on the number of points by using interlaced sampling to generate interlaced blocks that can be coded with the DL coding model from Fig. 2. The various scalable layers are independently coded, thus effectively offering multiple description coding which goes beyond scalable coding and may be useful for specific application domains.

3.1 High-level Description of the Proposal

The overall architecture of the proposed RS-DLPCC codec is presented in Fig. 5; moreover, the various modules are briefly described as follows:

- **Encoder**:
    - **Interlaced Blocks Creation**: The PC is first divided into large disjoint (super-)blocks; each super-block is further divided by applying interlaced sampling, thus generating up to 8 interlaced blocks for each super-block, which are coded separately;
    - **DL-based Block (En)Coding**: Each interlaced block is coded with the DL coding model described in Section 1.2.2;
    - **Block Coding Order Optimization**: After encoding the interlaced blocks within a super-block, the order by which they are attributed to each consecutive scalable layer is determined by minimizing the accumulated RD cost at each layer;
- **Decoder**:
    - **DL-based Block (De)Coding**: The interlaced blocks within each super-block, forming the different scalability layers, are progressively decoded with the DL coding model (from Fig. 2), yielding a fully (or partially) reconstructed PC.

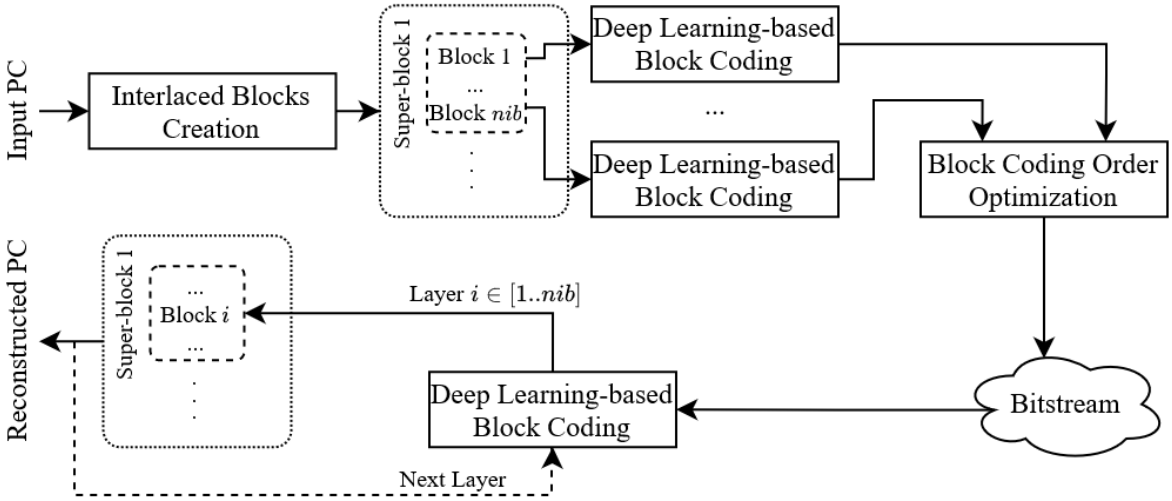


Fig. 5. Overall architecture of the proposed RS-DLPCC solution [2].

3.2 Detailed Description of each Architecture Module

Each of the modules presented in Fig. 5 is described here in more detail.

3.2.1 Interlaced Blocks Creation

As described in Section 1.2.1, a voxel-based 3D block representation is used. However, given the resolution scalability goal, it is proposed to divide the PC into interlaced blocks. This interlaced approach allows to successively increase the number of decoded points (i.e., the PC density) with each new decoded scalable layer, which is very effective from a subjective quality point of view. The interlaced blocks are obtained as follows:

1. **Division into Disjoint Super-blocks** – The PC is first divided into disjoint blocks, referred as super-blocks, here with size 128×128×128; other sizes may be used depending on the random access needs.
2. **Interlaced Sub-sampling of Super-blocks** – Using interlaced sub-sampling with a sampling factor of 2 in each 3D direction, each (128×128×128) super-block is divided into smaller blocks, here with the target size of 64×64×64. An example of interlaced sampling is shown in Fig. 6, where the coding blocks are half the size of the super-blocks in each spatial dimension.

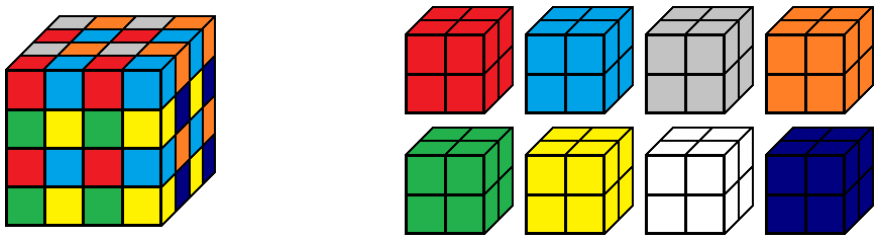


Fig. 6. Example of interlaced sampling with sampling factor of 2 in each 3D direction. A super-block of 4×4×4 samples results into 8 blocks, each with 2×2×2 samples [2].

This PC representation approach arranges the full PC into several super-blocks, each with up to 8 interlaced blocks, thus enabling resolution scalability with up to 8 scalable layers, as demonstrated in Fig. 7. Each interlaced block is independently coded with the DL-based coding solution described in Section 1.2.2. Additionally, the position of each super-block is transmitted to the decoder.

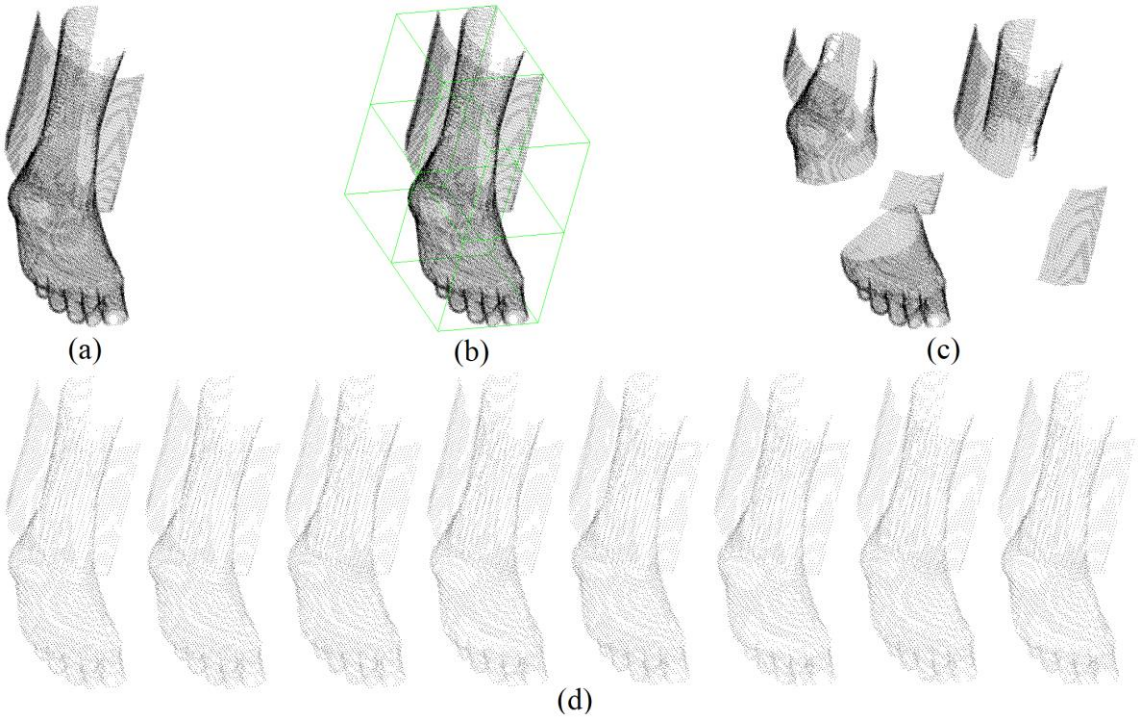


Fig. 7. Example of disjoint and interlaced blocks for the same super-block: (a) 128×128×128 super-block; (b) disjoint division into eight 64×64×64 blocks, four of which are empty; (c) the four disjoint occupied blocks obtained from (b); (d) the eight 64×64×64 interlaced blocks obtained from (a) [2].

3.2.2 DL-based Block Coding

The adopted end-to-end DL coding model for the RS-DLPCC solution is the same as the one presented in Fig. 2 for the DL-PCC solution. However, in this case, a single trained DL coding model is used to code all scalable layers of the proposed scalable coding solution.

3.2.3 Block Coding Order Optimization

In a resolution scalable context, at each layer, only one block in each super-block is decoded and added to the reconstructed PC, independently of the other blocks. Although the coding order of the blocks within a super-block does not impact the reconstruction quality at the last scalable layer, experiments have shown that the block coding order has an impact on the quality of the intermediate decoded layers. When decoding layers following a raster order, points are added along one spatial dimension at a time, meaning that there will be gaps in the other dimensions in intermediate layers, which will result in larger distances/errors.
With this in mind, after encoding all blocks with the end-to-end DL coding model, the best coding order for the 8 interlaced blocks constituting each super-block is determined by sequentially adding the block that most

reduces the accumulated distortion at each layer. This block coding order optimization only considers the accumulated distortion at each layer since it has been observed that the rate for all interlaced blocks within a super-block tends to be very similar; this approach allows reducing the overall coding complexity since no rate has to be computed for this optimization. An example of the benefit of an optimized coding order is shown in Fig. 8.

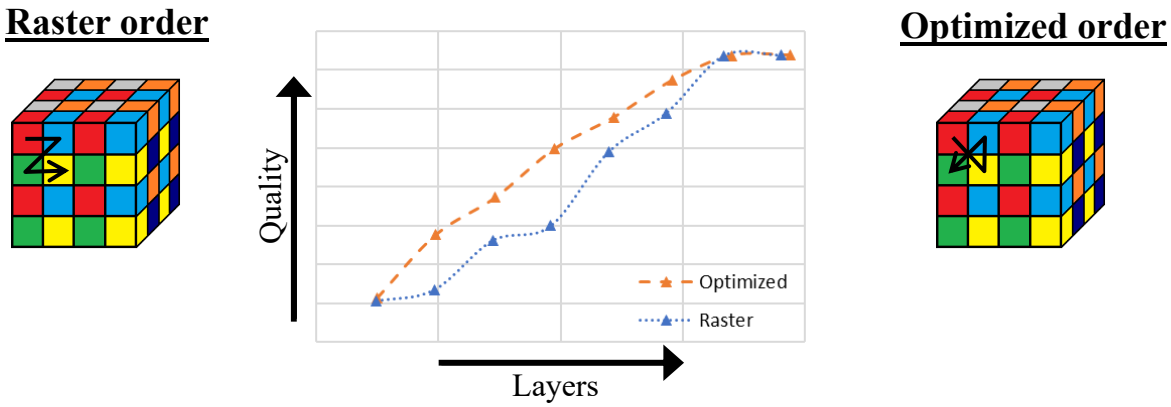


Fig. 8. Example of raster vs optimized layer decoding order.

3.2.4 DL-based Block (De)Coding

At the decoder side, each block is decoded with the DL coding model shown in Fig. 2. The PC is reconstructed by progressively decoding each layer of blocks, thus increasing the total number of decoded points and reconstructing an increasingly denser and richer PC. In fact, since blocks, and therefore layers, are coded independently and each offers a meaningful PC (what does not necessarily happen in all scalable coding solutions), RS-DLPCC can also be regarded as a multiple description coding solution.

3.3 DL Coding Model Training

For this RS-DLPCC proposal, the same loss function and a similar training process as for DL-PCC were used. In this case, only one model was trained using α=0.7, and λ=500 since this was found to provide a good RD trade-off, although not equally good for all PC densities. For the training dataset, interlaced blocks were used instead of disjoint blocks.

3.4 Scalability and Random Access Requirements

The proposed RS-DLPCC codec meets the JPEG Call for Evidence requirements due to the following features:

- **Random Access**: The PC is divided into super-blocks, which are coded independently. Since the positions of the coded super-blocks are transmitted to the decoder, this allows the user to choose decoding only selected regions of the PC, thus providing spatial random access.
- **Number of Points Scalability**: Within each super-block, its interlaced blocks are also coded separately, making it possible to decode only a subset of the interlaced blocks, while still allowing to reconstruct a

meaningful PC, although with fewer points. Then, the remaining interlaced blocks can be progressively decoded, adding points to the reconstructed PC, thus providing scalability on the number of points, as shown in Fig. 9.

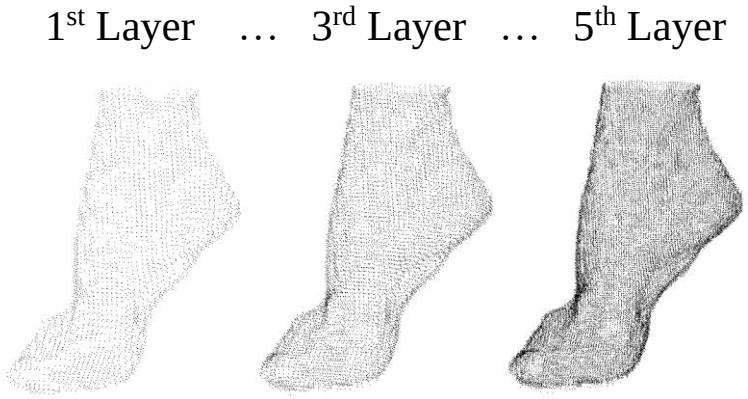


Fig. 9. Example of number of points scalability. The number of points increases with each decoded layer, thus also improving quality.

The full bitstream encapsulates:

- Super-block positions;
- Interlaced block coding order for the blocks within each super-block;
- Separate and identifiable sub-streams for all blocks within each super-block, consisting of the "Bitstream" and "Side Info Bitstream" obtained from the DL coding model shown in Fig. 2.

4. Quality Scalable DL-based PC Geometry Coding

This section describes the second proposed scalable codec, QS-DLPCC, which offers quality scalability. For additional details, please refer to [3].

4.1 High-level Description of the Proposal

The overall architecture of the proposed QS-DLPCC codec is presented in Fig. 10. This codec is also based on the previously described DL-PCC, with the addition of progressive coding of the latent values and a quality scalability control mechanism. QS-DLPCC shares the same DL coding model design described in Fig. 2, adapted for the proposed quality scalability control.

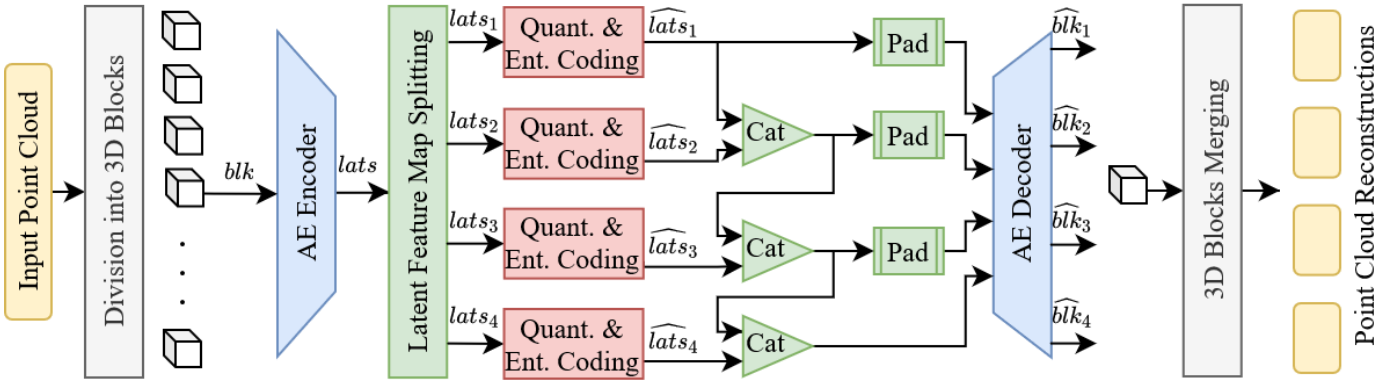


Fig. 10. Overall architecture of the proposed QS-DLPCC solution [3].

The various modules presented in Fig. 10 can be briefly described as follows:

- **Encoder**:
    - **Division into 3D Blocks**: The PC is divided into disjoint blocks of the target size, which are coded separately;
    - **AE Encoder**: Each block is encoded with an end-to-end DL coding model, which uses an architecture similar to the one in Fig. 2. The core step is once again an autoencoder, which transforms the input block into a latent representation;
    - **Latent Feature Map Splitting**: The latent representation, consisting of multiple feature maps, is divided and grouped into different layers, to be coded separately. This approach introduces the capability of performing quality scalability;
    - **Quantization and Entropy Coding**: The feature maps in each layer are then quantized and entropy coded, using the same DL-PCC approach (see Section 1.2.2);
- **Decoder**:
    - **Entropy Decoding and Inverse Quantization**: The bitstream is decoded to generate the feature maps in each layer;
    - **Latent Feature Map Grouping**: At each layer, its feature maps are grouped with the feature maps from the previously decoded layers;
    - **AE Decoder**: At each layer, the available feature maps are used to decode the block, with the inverse transform;
    - **3D Blocks Merging**: At each scalability layer, the decoded blocks are merged to reconstruct the PC.

4.2 Detailed Description of each Architecture Module

Each of the modules presented in Fig. 10 is described here in more detail.

4.2.1 Division into 3D Blocks & 3D Blocks Merging

Just like for the non-scalable codec (DL-PCC), the PC geometry (3D coordinates) is converted into a binary, voxel-based 3D block representation, as described in Section 1.2.1. The PC is then divided into disjoint blocks of size 64×64×64. Each block is independently coded with a DL-based coding solution, similar to the one described in Section 1.2.2, albeit with some differences to allow scalability, as described in the following sections. The position of each block is transmitted to the decoder so that, at the decoder, the decoded blocks may be merged to form the reconstructed PC.

4.2.2 AE Encoder & AE Decoder

The first step of the adopted DL-based PC geometry coding solution is the convolutional autoencoder (AE). Its design is the same as for the DL-PCC codec: 6 convolutional layers, each consisting of 32 filters with 5×5×5 support, resulting in 520000 weights plus 129 biases, totaling 520129 trainable parameters.

Given an input block of size 64×64×64, the AE encoder transforms it into a latent representation consisting of 32 feature maps of size 8×8×8, as detailed in Fig. 11.

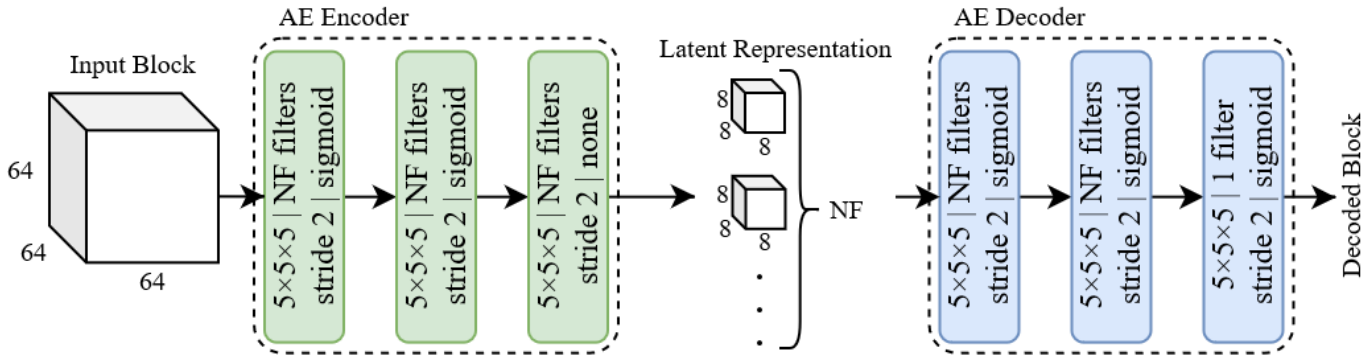


Fig. 11. Autoencoder design of the proposed QS-DLPCC solution. The number of filters (NF) was set to 32 [3].

4.2.3 Latent Feature Map Splitting & Latent Feature Map Grouping

After obtaining the full latent representation, this module implements the key feature of appropriately managing the latents to obtain a scalable PC geometry representation, thus allowing scalable/progressive decoding. At the encoder, the latent representation is split into several layers of feature maps to be separately encoded, thus multiple layer sub-streams are obtained after entropy coding.

In this proposal, 4 layers were defined, with a feature map distribution of 3, 5, 8 and 16 from the first to the last layer, respectively. While results for this proposal are presented considering these layer configurations, it is worth noting that the number of layers and the distribution of feature maps can be customized depending on the desired application requirements.

At the decoder side, these layers can be consecutively decoded, allowing to obtain four quality levels without the need to decode the entire representation/bitstream. The progressive decoding process is exemplified in

Fig. 12, and it can be described by the following stages:

- The sub-stream of the 1st layer, consisting of 3 feature maps, is entropy decoded; since the AE decoder requires 32 feature maps, the missing 29 are padded with zeros; the padded latent representation is given to the AE decoder, which reconstructs the block with a low quality.

- The sub-stream of the 2nd layer, consisting of 5 feature maps, is entropy decoded; these feature maps are grouped with the ones decoded in the 1st layer; the missing 24 feature maps are padded with zeros; the padded latent representation is given to the AE decoder, which reconstructs the block with slightly better quality.

- The sub-stream of the 3rd layer, consisting of 8 feature maps, is entropy decoded; these feature maps are grouped with the ones decoded in the previous two layers; the missing 16 feature maps are padded with zeros; the padded latent representation is given to the AE decoder, which reconstructs the block with improved quality.

- The sub-stream of the 4th layer, consisting of the last 16 feature maps, is entropy decoded; these are grouped with the ones decoded in the previous three layers; the complete latent representation is given to the AE decoder, which reconstructs the block with the best quality.

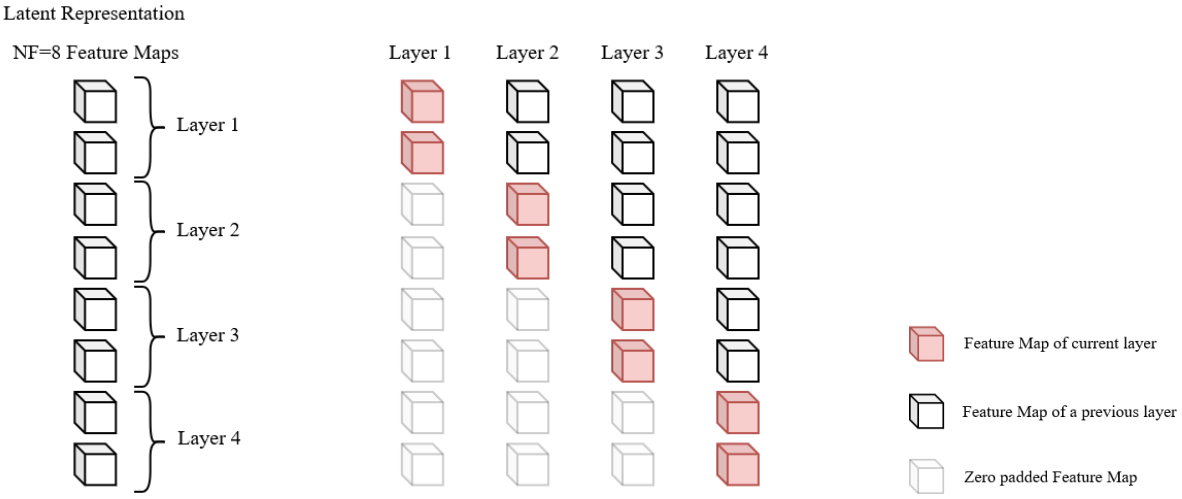


Fig. 12. Example of splitting and grouping of the feature maps in encoder and decoder, respectively.

4.2.4 Quantization and Entropy Coding & Entropy Decoding and Inverse Quantization

The quantization and entropy coding approach is the same as the one used for the non-scalable DL-PCC codec, described in Section 1.2.2. However, in this case, instead of using a single Variational autoencoder (VAE) for all 32 feature maps of the complete latent representation, four separate VAEs are used, one for each scalable layer individually:

- 1st layer: the VAE consists of 6 convolutional layers, each learning 3 filters with 3×3×3 support, resulting in 1458 weights plus 15 biases; to learn the entropy model (Entropy bottleneck in Section 1.2.2), 138 trainable parameters are used;

- 2nd layer: the VAE consists of 6 convolutional layers, each learning 5 filters with 3×3×3 support, resulting in 4050 weights plus 25 biases; to learn the entropy model (Entropy bottleneck in Section 1.2.2), 230 trainable parameters are used;

- 3rd layer: the VAE consists of 6 convolutional layers, each learning 8 filters with 3×3×3 support, resulting in 10368 weights plus 40 biases; to learn the entropy model (Entropy bottleneck in Section 1.2.2), 368 trainable parameters are used;

- 4th layer: the VAE consists of 6 convolutional layers, each learning 16 filters with 3×3×3 support, resulting in 41472 weights plus 80 biases; to learn the entropy model (Entropy bottleneck in Section 1.2.2), 736 trainable parameters are used.

The number of trainable parameters of a convolutional layer (not to be confused with the scalable layer) is dependent on the number of input data channels, the number of filters (output channels), and the filter support size (weights = input channels × filter support × filters; bias = filters). As for the number of parameters to learn the entropy model, it is directly dependent on the number of filters. For this reason, since each scalable layer has a different number of feature maps/filters, the number of trainable parameters is also different.

A single trained DL coding model is used to code all scalable layers of the proposed progressive coding solution. The total number of trainable parameters in the full DL coding model is 579109.

4.3 DL Coding Model Training

Similar to the previous codec, the DL coding model used in the QS-DLPCC codec was trained by minimizing a RD loss function. However, in this solution all four scalable layers were trained jointly, meaning that a RD term is minimized for each layer. The loss function is thus given by:

$$Loss\ Function = \sum_{i=1}^{4} Distortion_i + \lambda_i \times Coding\ rate_i\,. \tag{3}$$

It is important to select appropriate λ_i values, since they can significantly impact the RD performance. Similar λ_i values for two layers may cause the RD performance to be the same, thus effectively eliminating one scalable layer, which is undesirable. Since the base layer must offer a low bitrate, λ_1 should be largest, to obtain an initial rough PC reconstruction. As the layers progress, λ_i should be reduced to allow more rate to encode latter feature maps, so that the AE can learn additional features with latter filters, thus achieving successively higher quality with every layer. For this proposal, using λ_i=20000, 4000, 1000 and 100 were found to be appropriate values. As for the distortion metric, the Focal Loss [7] was also used with α=0.7 however, for sparse PCs, larger values are preferable, e.g. α=0.9. Similar to DL-PCC and ADL-PCC, training data was divided into disjoint blocks.

4.4 Scalability and Random Access Requirements

The proposed QS-DLPCC codec meets the JPEG Call for Evidence requirements due to the following features:

- **Random Access**: Just as for the DL-PCC codec, the PC is divided into disjoint blocks, which are coded independently. Since the positions of the coded blocks are transmitted to the decoder, this allows the user to choose decoding only selected regions of the PC, thus providing spatial random access.

- **Quality Scalability**: For each block, the latent representation (i.e. the transform coefficients) is divided into different layers that are coded separately, making it possible to decode only part of the latent representation, allowing to reconstruct a meaningful PC, albeit with lower quality. The latent representation can be progressively decoded, generating more and more refined reconstructions with each layer, as shown in Fig. 13.

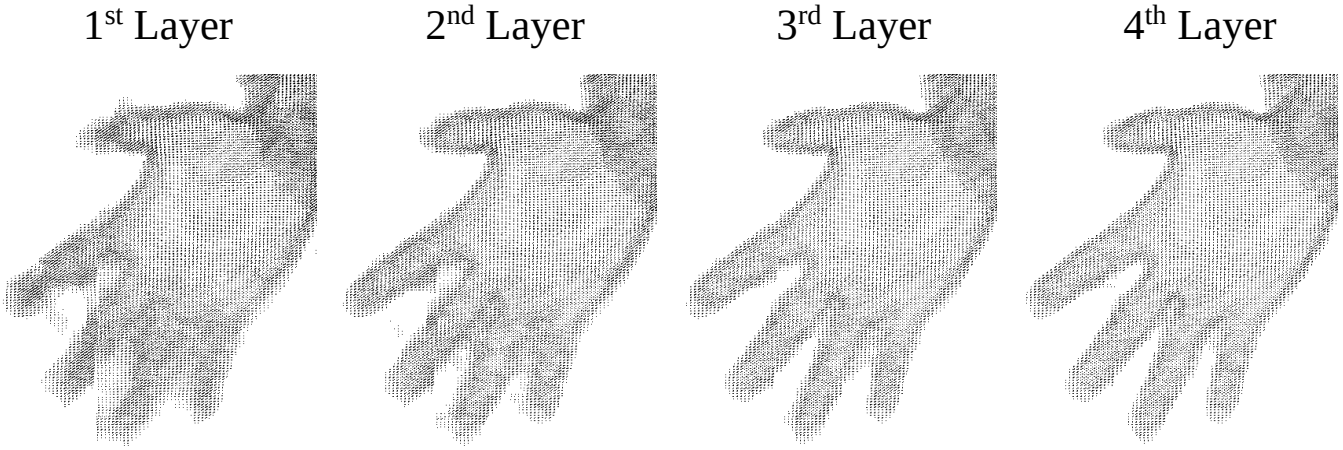


Fig. 13. Example of quality scalability. The reconstructed PC becomes more refined with each decoded layer, increasing quality.

The full bitstream encapsulates:

- Block positions;

- Separate and identifiable sub-streams for all blocks, consisting of the "Bitstream" and "Side Info Bitstream" obtained from the DL coding model.

5. Performance Assessment

To assess the performance of the proposed scalable coding solutions, the JPEG test PCs were coded following the CTC defined for the JPEG Call for Evidence on Point Cloud Coding [10].

5.1 Test Material

The test material used for the assessment of the proposed scalable codecs consists of 8 PCs, including people (full bodies and upper bodies) as well as inanimate objects. This dataset was made available in the context of

the JPEG Call for Evidence on Point Cloud Coding [10], and is presented in Fig. 14.

(a) Bumbameuboi *(b) Guanyin* *(c) Rhetorician* *(d) Romanoillamp*

(e) Phil *(f) Ricardo* *(g) Longdress* *(h) Soldier*

Fig. 14. Example rendering for the test PCs.

5.2 RD Performance Results: Tables

For each scalable/progressive layer, the JPEG recommended geometry quality metrics – point-to-point (PSNR D1), point-to-plane (PSNR D2) and plane-to-plane angular similarity (MSE AS) – were computed.

The key results, i.e. number of decoded points (Out Points), number of bytes to code the geometry (Geo Bytes), bits per geometry input point (Geo bpp), and objective quality metrics for geometry, for each RD point and for each test PC, are detailed in Table I and Table II for RS-DLPCC and QS-DLPCC, respectively.

Table I. RS-DLPCC coding results, at different RD points, for each test PC.

Point Cloud	In Points	RD Point	Out Points	Geo Bytes	Geo bpp	PSNR D1	PSNR D2	MSE AS
Bumbameuboi	113160	R01	16344	45281	3.201	47.957	66.551	0.567
		R02	31203	88121	6.230	51.065	68.048	0.578
		R03	46013	131139	9.271	52.957	68.848	0.591
		R04	60101	173461	12.263	54.305	69.607	0.606
		R05	73740	215077	15.205	55.384	69.969	0.622
		R06	86673	256654	18.144	56.290	70.013	0.637
		R07	99265	297454	21.029	57.096	70.024	0.655
		R08	110997	337436	23.855	57.757	70.030	0.669
Guanyin	2297852	R01	369006	85901	0.299	59.230	69.971	0.842
		R02	714638	168782	0.588	65.305	72.062	0.896
		R03	1081832	254093	0.885	66.394	73.035	0.908
		R04	1436507	337845	1.176	67.562	73.092	0.916
		R05	1801637	422968	1.473	68.684	73.109	0.918
		R06	2155293	506551	1.764	70.114	73.118	0.920
		R07	2501924	589250	2.051	70.387	73.106	0.921
		R08	2851054	672564	2.342	70.361	73.077	0.923
Longdress	857966	R01	134776	32266	0.301	62.214	70.168	0.863
		R02	265807	63829	0.595	65.502	71.699	0.907
		R03	400028	95756	0.893	66.511	72.712	0.919
		R04	531394	127279	1.187	67.642	73.750	0.928
		R05	665280	159117	1.484	68.758	73.760	0.932
		R06	798191	190753	1.779	70.228	73.755	0.934
		R07	929315	222269	2.073	70.831	73.766	0.936
		R08	1061100	253838	2.367	70.817	73.752	0.938
Phil	356258	R01	56054	12001	0.269	56.748	62.991	0.837
		R02	110175	23349	0.524	59.546	64.849	0.871
		R03	165793	34944	0.785	60.471	65.820	0.879
		R04	220232	46371	1.041	61.437	66.888	0.887
		R05	275694	57923	1.301	62.500	67.332	0.889
		R06	330543	69350	1.557	63.857	67.339	0.891
		R07	384529	80724	1.813	64.120	67.350	0.893
		R08	439073	92144	2.069	64.106	67.328	0.895
Rhetorician	1764588	R01	285865	66934	0.303	61.681	70.157	0.853
		R02	562068	132464	0.601	65.323	71.784	0.902

		R03	845633	198742	0.901	66.400	72.797	0.914
		R04	1122369	264231	1.198	67.553	73.094	0.922
		R05	1405170	330446	1.498	68.650	73.108	0.924
		R06	1684737	396256	1.796	70.073	73.121	0.926
		R07	1961188	461836	2.094	70.203	73.126	0.928
		R08	2239320	527297	2.391	70.186	73.110	0.929
Ricardo	1414040	R01	221608	43435	0.246	62.862	68.879	0.851
		R02	436162	85407	0.483	65.665	70.800	0.883
		R03	656572	128153	0.725	66.577	71.960	0.891
		R04	873021	170455	0.964	67.538	73.380	0.899
		R05	1092521	212974	1.205	68.609	73.731	0.901
		R06	1309399	255314	1.444	69.993	73.739	0.903
		R07	1523615	297200	1.681	70.371	73.748	0.905
		R08	1739816	339176	1.919	70.354	73.728	0.906
Romanoillamp	638071	R01	117260	37504	0.470	59.351	71.133	0.805
		R02	229806	74161	0.930	63.823	72.678	0.905
		R03	345049	111118	1.393	65.420	73.047	0.924
		R04	456687	147693	1.852	66.860	73.036	0.935
		R05	571815	184583	2.314	67.940	73.062	0.940
		R06	683247	221230	2.774	68.479	73.080	0.943
		R07	795735	257870	3.233	68.475	73.071	0.945
		R08	908199	294310	3.690	68.459	73.046	0.947
Soldier	1089091	R01	172101	41611	0.306	62.254	70.239	0.864
		R02	339563	82292	0.604	65.545	71.675	0.906
		R03	510993	123432	0.907	66.547	72.580	0.917
		R04	677998	164045	1.205	67.663	73.753	0.925
		R05	848530	205121	1.507	68.780	73.765	0.930
		R06	1017803	245951	1.807	70.251	73.770	0.933
		R07	1184791	286658	2.106	70.797	73.777	0.935
		R08	1352547	327281	2.404	70.784	73.762	0.937

Table II. QS-DLPCC coding results, at different RD points, for each test PC.

Point Cloud	In Points	RD Point	Out Points	Geo Bytes	Geo bpp	PSNR D1	PSNR D2	MSE AS
Bumbameuboi	113160	R01	305286	33072	2.338	45.128	62.480	0.550
		R02	374799	61120	4.321	47.140	64.617	0.575
		R03	321028	182749	12.920	55.422	70.660	0.669

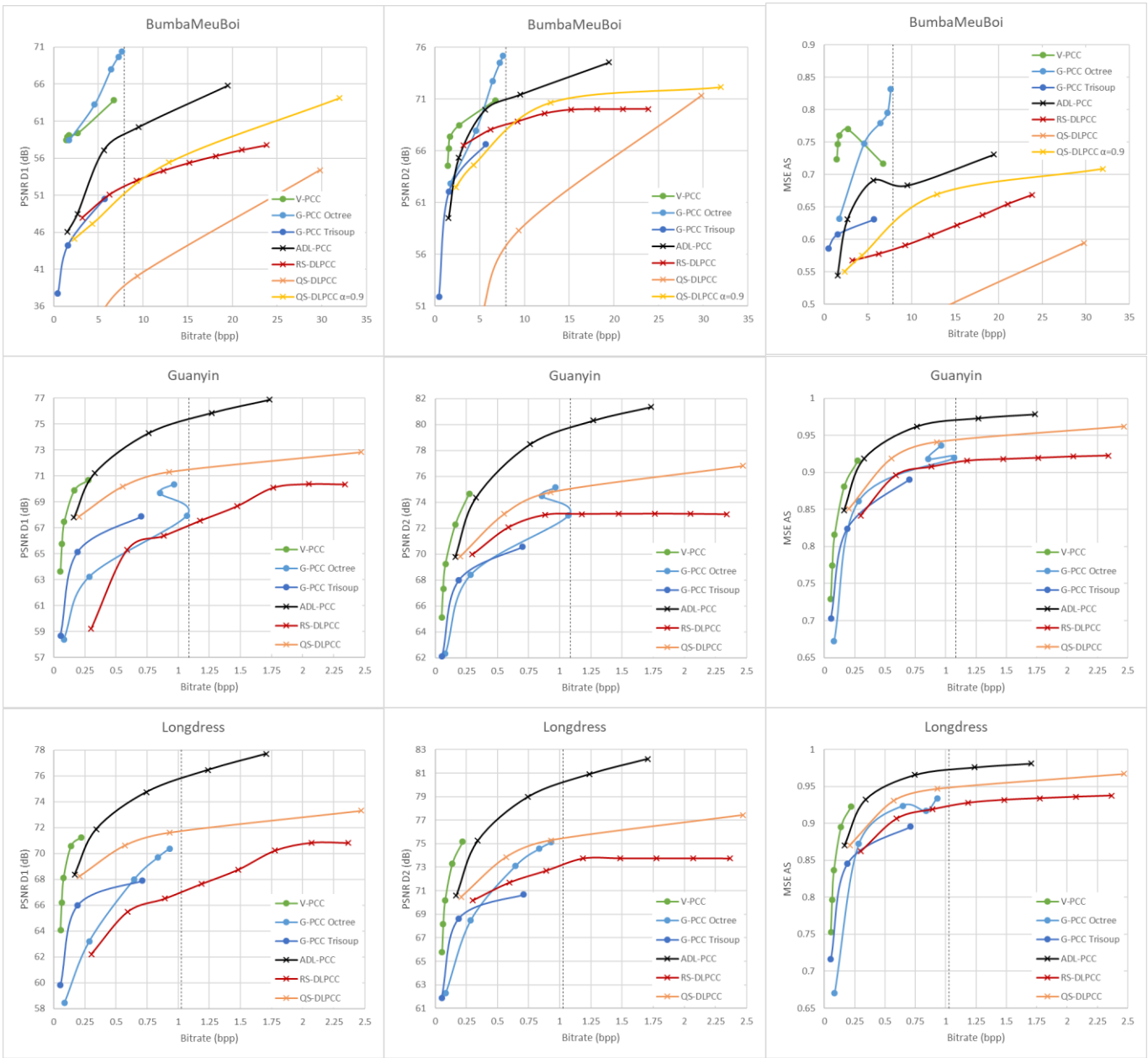
		R04	389317	452218	31.970	64.124	72.151	0.709
Guanyin	2297852	R01	3126253	58177	0.203	67.849	69.843	0.851
		R02	2960283	159006	0.554	70.176	73.106	0.919
		R03	2811976	265875	0.926	71.308	74.792	0.941
		R04	2662705	709478	2.470	72.843	76.820	0.962
Longdress	857966	R01	1165557	22487	0.210	68.248	70.462	0.870
		R02	1087195	61112	0.570	70.624	73.845	0.931
		R03	1037178	99865	0.931	71.624	75.282	0.947
		R04	977774	265100	2.472	73.320	77.423	0.967
Phil	356258	R01	459195	8815	0.198	61.989	64.706	0.816
		R02	435346	22533	0.506	64.186	67.732	0.892
		R03	422084	36694	0.824	65.158	69.068	0.917
		R04	411515	94234	2.116	66.341	70.598	0.942
Rhetorician	1764588	R01	2426829	46805	0.212	67.743	69.950	0.859
		R02	2327674	127218	0.577	69.772	72.804	0.923
		R03	2213386	211187	0.957	70.705	74.109	0.941
		R04	2085923	562896	2.552	72.333	76.410	0.963
Ricardo	1414040	R01	1828147	32094	0.182	68.348	71.192	0.835
		R02	1735765	82851	0.469	70.243	73.832	0.899
		R03	1695908	135176	0.765	71.010	74.892	0.919
		R04	1658598	352250	1.993	72.008	76.192	0.944
Romanoillamp	638071	R01	871880	24686	0.310	62.373	70.059	0.854
		R02	1071201	66936	0.839	67.467	72.413	0.931
		R03	1016506	113266	1.420	67.956	73.307	0.943
		R04	945745	313758	3.934	69.049	75.827	0.964
Soldier	1089091	R01	1488172	28926	0.212	68.353	70.675	0.871
		R02	1388370	78785	0.579	70.594	73.874	0.931
		R03	1321438	128894	0.947	71.618	75.325	0.948
		R04	1244676	339062	2.491	73.292	77.422	0.968

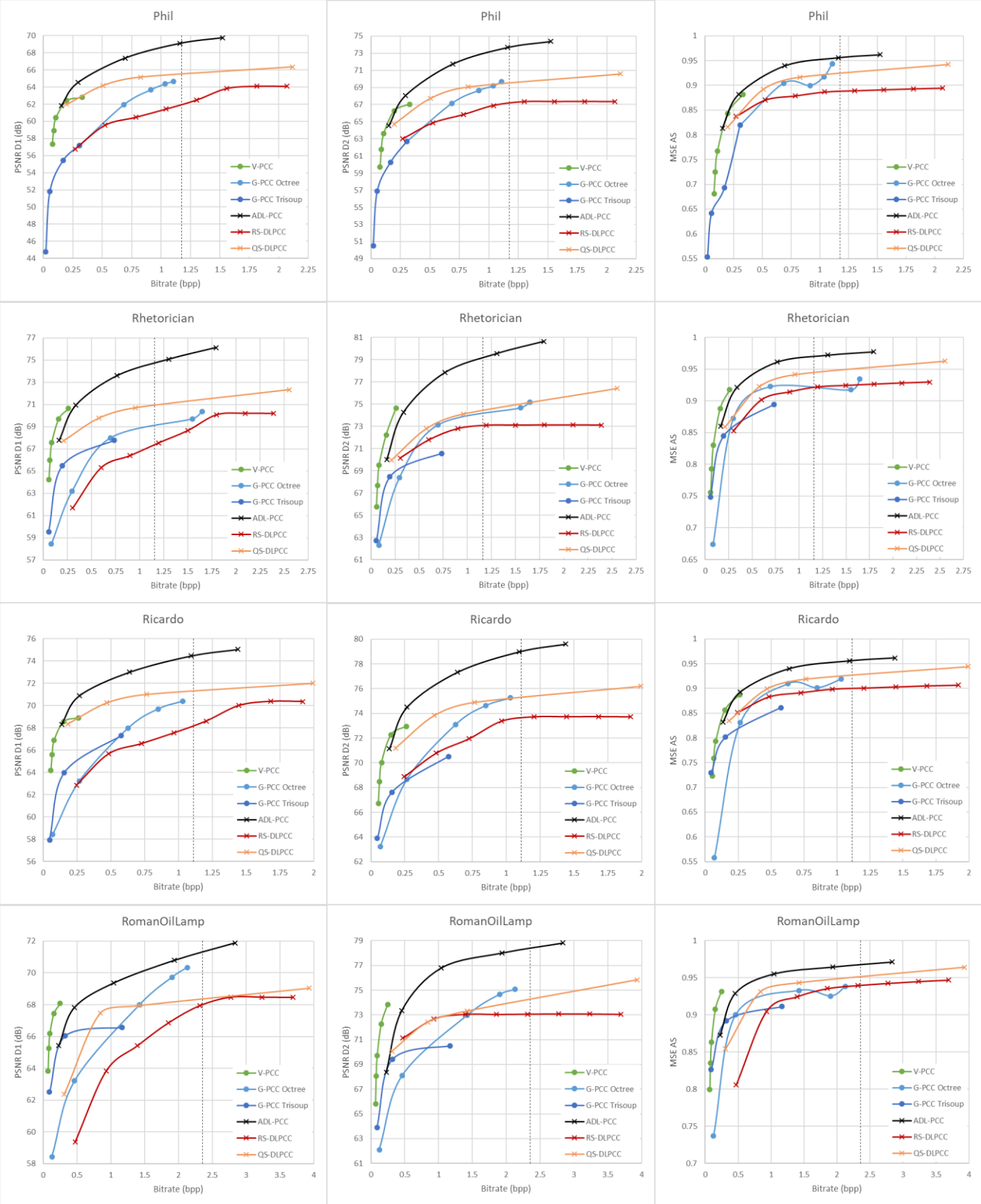
5.3 RD Performance Results: RD Charts and BD-PSNR

The RD performance results for RS-DLPCC and QS-DLPCC are plotted as RD charts and compared with the G-PCC and V-PCC (Intra) anchors in Fig. 15, using the three previously mentioned geometry objective

quality metrics. The results for the ADL-PCC non-scalable solution are also presented in Fig. 15, to allow for a RD performance comparison and the assessment of the RD penalty cost introduced by the use of scalability (keeping in mind that these scalable coding solutions are the first scalable designs made and better performing designs are expected in the future).

To summarize the results, Table III shows the Bjontegaard-Delta PSNR (BD-PSNR) gains for each solution, using G-PCC Trisoup as the reference codec, since it is the one showing the poorest RD performance. For many cases, mostly for ADL-PCC, the intersection of quality ranges between the RD curves is rather short, or even non-existing; in these situations, the computation of the BD-Rate savings is unreliable or not possible at all, and thus BD-Rate results are not shown together with BD-PSNR.




Phil
PSNR D1 (dB)
Bitrate (bpp)
Phil
PSNR D2 (dB)
Bitrate (bpp)
Phil
MSE AS
Bitrate (bpp)
Rhetorician
PSNR D1 (dB)
Bitrate (bpp)
Rhetorician
PSNR D2 (dB)
Bitrate (bpp)
Rhetorician
MSE AS
Bitrate (bpp)
Ricardo
PSNR D1 (dB)
Bitrate (bpp)
Ricardo
PSNR D2 (dB)
Bitrate (bpp)
Ricardo
MSE AS
Bitrate (bpp)
RomanOilLamp
PSNR D1 (dB)
Bitrate (bpp)
RomanOilLamp
PSNR D2 (dB)
Bitrate (bpp)
RomanOilLamp
MSE AS
Bitrate (bpp)
V-PCC
G-PCC Octree
G-PCC Trisoup
ADL-PCC
RS-DLPCC
QS-DLPCC

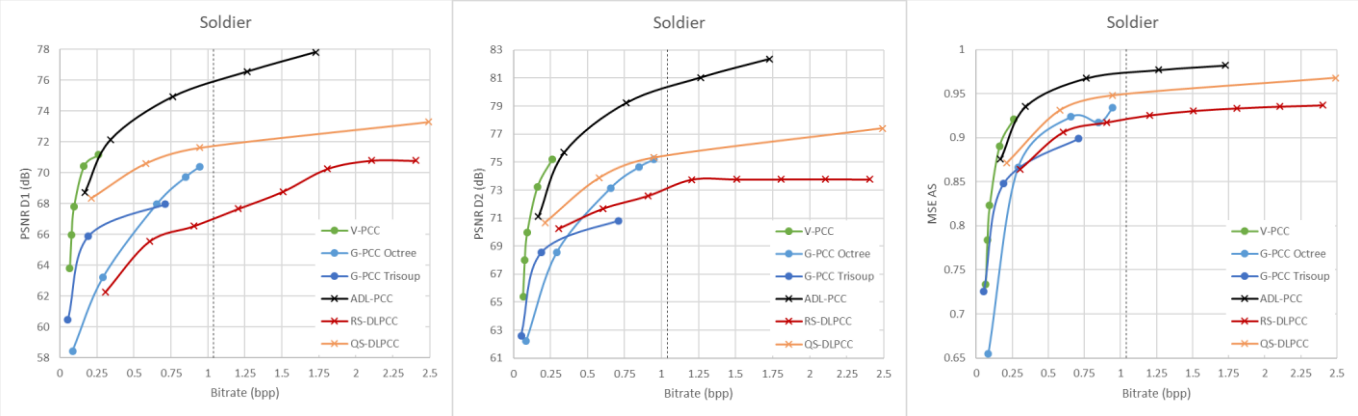


Fig. 15. RD performance for the proposed RS-DLPCC and QS-DLPCC solutions, in comparison with non-scalable ADL-PCC, and the V-PCC and G-PCC anchors. Each row corresponds to a different test PC. From left to right: PSNR D1, PSNR D2 and MSE AS geometry quality metrics. The dashed vertical line marks the lossless bitrate achieved with G-PCC Octree.

Table III. Bjontegaard-Delta PSNR gains between the DL-based solutions using G-PCC Trisoup as reference. The best results are presented in bold, and the second best in italic.

	BD-PSNR D1 (dB)			**BD-PSNR D2 (dB)**		
	ADL-PCC	**RS-DLPCC**	**QS-DLPCC**	**ADL-PCC**	**RS-DLPCC**	**QS-DLPCC**
Bumbameuboi	**2.979**	*0.135*	-1.824	*0.976*	**1.159**	-1.510
Guanyin	**4.630**	-3.928	*2.449*	**4.973**	1.305	*2.250*
Longdress	**4.685**	-3.186	*2.435*	**5.308**	0.804	*2.547*
Phil	**7.099**	0.063	*5.952*	**5.224**	0.769	*3.688*
Rhetorician	**4.215**	-3.175	*2.045*	**4.645**	1.054	*1.899*
Ricardo	**5.146**	-1.792	*3.429*	**5.464**	0.396	*3.489*
Romanoillamp	**1.602**	-4.067	*-0.377*	**3.701**	*1.930*	1.571
Soldier	**4.977**	-3.162	*2.461*	**5.683**	0.742	*2.579*
Average	**4.417**	-2.389	*2.071*	**4.497**	1.020	*2.064*

The key observation is that both ADL-PCC and QS-DLPCC show substantial quality gains regarding G-PCC Trisoup while this is not the case for RS-DLPCC which shows quality losses, very much related to the independent coding of the scalable layers.

The dashed vertical line indicates the rate for which G-PCC Octree achieves lossless coding, thus clearly indicating that lossy compression performance on the right side of that line may not be very meaningful.

5.4 Number of Decoded Points

Fig. 16 plots the number of decoded points for each coding solution. These charts are very illustrative of the meaning of number of points scalability.

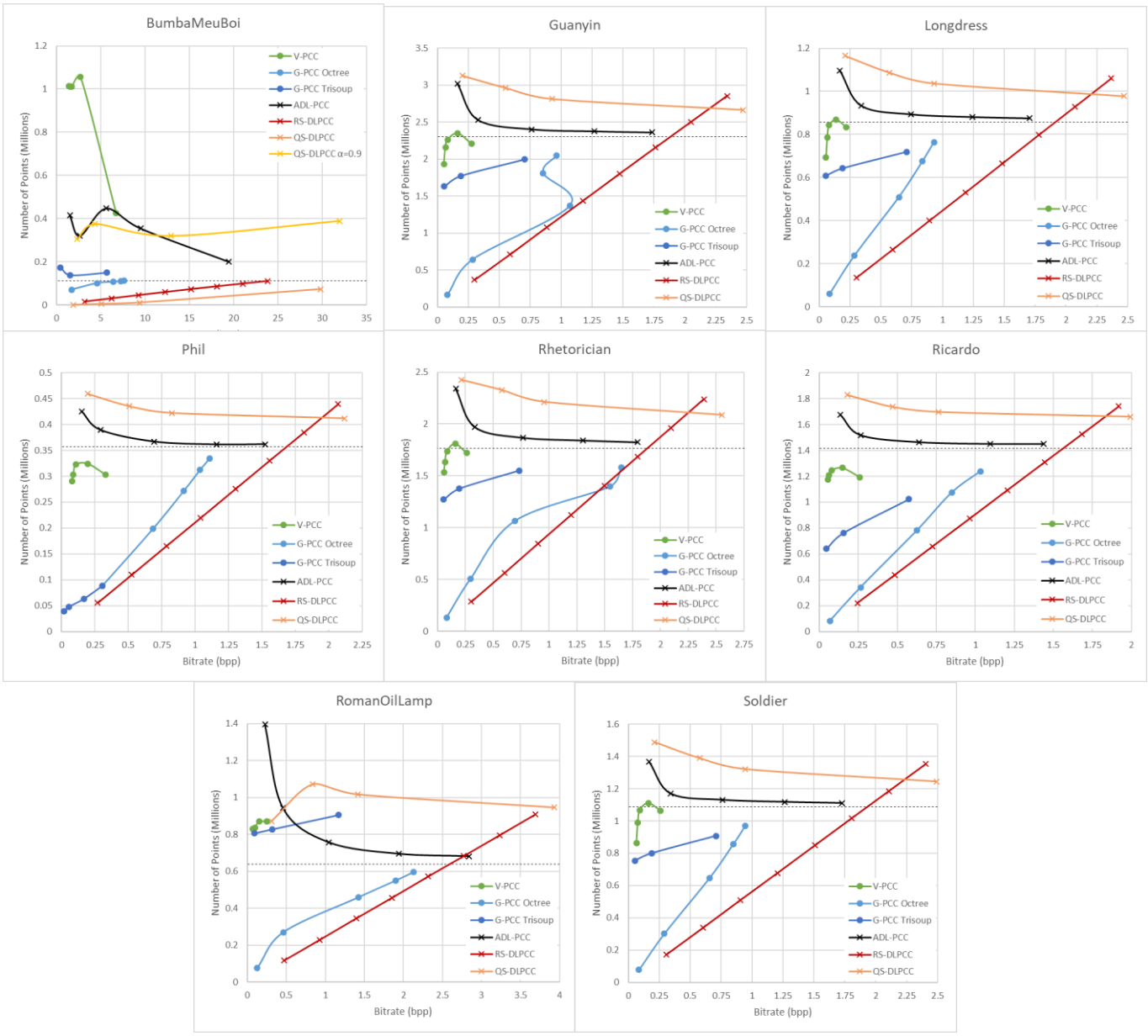


Fig. 16. Evolution of the number of points decoded, for every rate point, for the proposed RS-DLPCC and QS-DLPCC solutions, the non-scalable ADL-PCC, and the V-PCC and G-PCC anchors. The dashed horizontal line marks the original number of input points.

The key observation is the good resolution scalability offered by G-PCC Octree and RS-DLPCC solutions while ADL-PCC and QS-PCC start with a large number of decoded points to approach later the original number of decoded points. Interestingly, V-PCC does not have a large variation on the number of decoded points for the various RD points.

5.5 Analysis of the Results

From the full set of performance results included in the previous subsections, it is possible to observe and

conclude:

ADL-PCC versus MPEG Benchmarks

- ADL-PCC achieves significant compression gains when compared with both G-PCC codecs, at medium and high bitrates.

- Moreover, ADL-PCC has often RD performance comparable or just below V-PCC Intra, thus demonstrating the great potential of DL-based PC coding solutions.

RS-DLPCC versus MPEG Benchmarks

- The RS-DLPCC RD performance is comparable to G-PCC performance of both Trisoup and Octree, with fluctuations depending on the G-PCC method and the input PC features, e.g. density; however, this also largely depends on the used geometry objective quality metric.

- Compared with V-PCC, however, RS-DLPCC performs considerably worse.

- The RS-DLPCC performance is significantly poorer for the PSNR D1 quality metric than for the other quality metrics, mainly for the first scalable layers. This is due to the very low number of decoded points at early layers, as shown in Fig. 16. This mismatch with the number of points in the reference PC results in a large error distance when directly measuring the distance between neighboring points. As such, this metric alone may not be appropriate to measure the performance of coding solutions with number of points scalability, notably for the first layers.

- The RS-DLPCC performance improves for the PSNR D2 and MSE AS metrics, notably for the first layers, outperforming G-PCC Trisoup in many situations. These quality metrics seem to be more robust to differences on the number of points between the reference and decoded PCs, since they measure the error considering the surface formed by the points.

- The number of points decoded at each RS-DLPCC layer increases linearly, since each layer contains approximately the same number of interlaced blocks and points.

QS-DLPCC versus MPEG Benchmarks

- The QS-DLPCC performance is competitive with that of both G-PCC anchors, achieving significant compression gains in most cases.

- QS-DLPCC is still fairly below V-PCC, but in some cases it achieves a close performance.

- While for most PCs the trained DL coding model with $\alpha=0.7$ (from Equation 2) provides very good results, for the sparse *Bumbameuboi* PC the performance is poor, leading to a rather low number of decoded points. This is because the α parameter strongly influences the number of decoded points

and its ideal value is very dependent on the PC density/sparsity. By training a DL coding model with α=0.9 instead, the performance for *Bumbameuboi* is significantly improved.

- The QS-DLPCC solution is highly customizable, in the sense that the DL coding model can be trained for different combinations of the parameters that control the scalability: the number of layers, and the distribution of feature maps per layer. By fine tuning these parameters, this solution can further improve RD performance.

QS-DLPCC versus RS-DLPCC

- Overall, QS-DLPCC considerably outperforms RS-DLPCC, with the largest gains being observed for the PSNR D1 quality metric. This codec does not present the issue of having a very small number of points at early layers, which would penalize results for PSNR D1.

- QS-DLPCC tends to decode a large number of points, especially at early layers, and approaches the original number of points at the last layer. This suggests that the α value from the focal loss (Equation (2)) used in training may not be the ideal one. Using an adaptive DL coding selection method with multiple trained DL coding models, similar to the one used in ADL-PCC, may be able to reduce the number of (unnecessary) points, ultimately improving the RD performance.

- The key reason for the better QS-DLPCC performance seems to be related to the independent coding of the RS-DLPCC scalable layers; this allows concluding that there is a RD price to pay to obtain multiple description coding which may be a desired functionality for specific applications.

Scalable (RS-DLPCC and QS-DLPCC) versus non-scalable (ADL-PCC)

- There is still a significant RD performance penalty associated to the use of the scalable codecs as may be observed by comparing with the non-scalable ADL-PCC benchmark. It is expected that this RD penalty may be reduced in the future since RS- and QS-DLPCC are the first, certainly non-optimum, DL-based scalable coding solutions which serve as a proof of concept.

- The ADL-PCC results indicate that there is still a large margin of improvement for the scalable solutions and thus many improvements may still be obtained. For instance, there is still redundancy between the scalable layers that can be better exploited, and the DL coding model parameters can be better optimized according to the input PC characteristics.

6. Color Coding

This proposal is focused on PC geometry coding only, notably proposing the RS-DLPCC and QS-DLPCC solutions. It is worth noting that the proposed DL-based geometry scalable solutions can be extended to jointly code both geometry and color, for instance by adding the color information in separate channels to the binary 3D blocks that represent the geometry.

However, the JPEG PCC CTC [10] only foresee subjective quality assessment for the proposals coding both

the geometry and color. Thus, to allow for some subjective quality assessment, the proponents decided to code the color with the G-PCC standard for the decoded geometry/points generated with the RS-DLPCC solution, after recoloring. This choice was determined by the limited time to code the texture and the interest in performing subjective assessment when the number of decoded points changes substantially as it happens for RS-DLPCC.

In this context, the G-PCC standard was used for color coding, as follows:

- Geometry is encoded and decoded with the proposed RS-DLPCC solution;
- For each geometry scalable layer, the decoded points are recolored by transferring the color from the original input PC, considering the nearest neighboring point;
- For each geometry scalable layer, the color for the recolored points is encoded with G-PCC, using the lossless geometry coding mode with Octree (so that the (decoded) geometry is not changed during color coding), and the *Predlift* color encoder, which has been reported to be superior to RAHT [10].

For G-PCC coding, the "MPEG PCC tmc13 version release-v7.0-0-g47cf663" reference software has been used, with the parameters described in the JPEG Pleno PCC CTC v3.3 [10] associated to the JPEG PCC Call for Evidence.

The CTC specifies the following five target bitrates (with a ±10% tolerance) for the joint coding of geometry and color, to be evaluated in subjective tests:

- R01: 0.10 bits per point (bpp);
- R02: 0.35 bpp;
- R03: 1.00 bpp;
- R04: 2.00 bpp;
- R05: 4.00 bpp.

For color coding, the *qp* parameter was set for each PC and each rate point in order to achieve the target joint (geometry + color) bitrates specified in the CTC; the used *qp* values for the color are shown in Table IV.

Table IV. G-PCC qp parameter values for color coding.

Point Cloud	R01	R02	R03	R04	R05
Bumbameuboi	-	46	46	34	28
Guanyin	-	46	46	34	28
Longdress	-	46	46	34	28
Phil	-	40	40	34	22
Rhetorician	-	40	40	28	19

Ricardo	-	28	22	22	10
Romanoillamp	-	46	40	40	28
Soldier	-	40	40	28	22

No results could be obtained for the first target rate point R01, corresponding to lower rate (0.1 bpp), for any test PC.

For the remaining rate points, the obtained results are presented in Table V. Only the PSNR D2 geometry metric is shown since it is more suited to express the subjective quality of resolution scalable solutions; nonetheless, the results for all metrics are available in the supplemental results Excel file.

It is important to note that, for the *Bumbameuboi* PC, the obtained bitrates surpassed the target bitrates by a significant margin, which may justify its exclusion from the subjective assessment experiment. This is largely due to the fact that this PC is very sparse and the current DL coding model parameters may not be optimal for this type of PC. This also reinforces the importance of having an adaptive coding solution as ADL-PCC.

Likewise, for the *Romanoillamp* PC, the second rate point R02 could not be achieved, presenting a slightly larger bitrate. These cases are marked with color in Table V.

Table V. RS-DLPCC geometry + G-PCC Predlift color coding results for the target rate points, for each test PC.

Point Cloud	RD Point	Out Points	Geo Bytes	Color Bytes	Total Bytes	bpp	PSNR D2	PSNR Color
Bumbameuboi	R02	16344	45281	1273	46554	3.29	66.55	21.69
	R03	46013	131139	3200	134339	9.50	68.85	23.36
	R04	73740	215077	11126	226203	15.99	69.97	24.78
	R05	110997	337436	28566	366002	25.88	70.03	26.33
Guanyin	R02	369006	85901	11096	96997	0.34	69.97	25.57
	R03	1081832	254093	32625	286718	1.00	73.04	26.34
	R04	1801637	422968	174502	597470	2.08	73.11	27.82
	R05	2851054	672564	537709	1210273	4.21	73.08	29.54
Longdress	R02	134776	32266	5196	37462	0.35	70.17	25.52
	R03	400028	95756	14416	110172	1.03	72.71	26.48
	R04	665280	159117	65169	224286	2.09	73.76	28.22
	R05	1061100	253838	193271	447109	4.17	73.75	29.86
Phil	R02	56054	12001	3500	15501	0.35	62.99	28.98
	R03	165793	34944	9913	44857	1.01	65.82	29.94
	R04	275694	57923	24764	82687	1.86	67.33	31.20
	R05	439073	92144	101153	193297	4.34	67.33	32.87

Rhetorician	R02	285865	66934	9623	76557	0.35	70.16	32.94
	R03	845633	198742	25520	224262	1.02	72.80	34.07
	R04	1405170	330446	99871	430317	1.95	73.11	35.69
	R05	2239320	527297	352948	880245	3.99	73.11	37.29
Ricardo	R02	221608	43435	13912	57347	0.32	68.88	40.94
	R03	656572	128153	62662	190815	1.08	71.96	42.25
	R04	1309399	255314	124759	380073	2.15	73.74	44.37
	R05	1739816	339176	336120	675296	3.82	73.73	45.09
Romanoillamp	R02	117260	37504	2249	39753	0.50	71.13	27.19
	R03	229806	74161	8968	83129	1.04	72.68	28.01
	R04	456687	147693	22292	169985	2.13	73.04	28.85
	R05	683247	221230	102091	323321	4.05	73.08	29.99
Soldier	R02	172101	41611	6617	48228	0.35	70.24	32.80
	R03	510993	123432	17926	141358	1.04	72.58	33.85
	R04	848530	205121	70680	275801	2.03	73.76	35.53
	R05	1352547	327281	194662	521943	3.83	73.76	37.06

7. Submitted Materials

This section lists the materials that constitute this proposal, notably:

- JPEG document with textual description:
    - IT-IST-IPLeiria Response to Call for Evidence on JPEG.pdf
- Excel file with complete results:
    - SupplementaryResults.xlsx
- Source code/scripts of the decoder for the proposed scalable solutions (RS-DLPCC, and QS-DLPCC for 2 different parameter α values):
    - 01_SourceCode/
- Bitstream files of the proposed scalable solutions (RS-DLPCC, and QS-DLPCC for 2 different parameter α values):
    - 02_Bitstream_Files/
- Decoded PCs with geometry only, obtained with the proposed scalable solutions (RS-DLPCC, and QS-

DLPCC for 2 different parameter α values):

- 03_Decoded_PCs_Geo/

- Decoded PCs with color for subjective assessment, with geometry obtained with the RS-DLPCC solution, and color with G-PCC Predlift:
  - 04_Decoded_PCs_Geo+Color/

References

1. *ISO/IEC JTC1/SC29/WG1 N88014, "Final Call for Evidence on JPEG Pleno Point Cloud Coding," Online Meeting, July 2020.*

2. *A. Guarda, Nuno M. M. Rodrigues and F. Pereira, "Deep Learning-based Point Cloud Geometry Coding with Resolution Scalability", IEEE International Workshop on Multimedia Signal Processing (MMSP'2020), Tampere, Finland, September 2020.*

3. *A. Guarda, Nuno M. M. Rodrigues and F. Pereira, "Point Cloud Geometry Scalable Coding With a Single End-to-End Deep Learning Model", IEEE International Conference on Image Processing (ICIP'2020), Abu Dhabi, United Arab Emirates, October 2020.*

4. *A. Guarda, Nuno M. M. Rodrigues and F. Pereira, "Adaptive Deep Learning-based Point Cloud Geometry Coding", submitted to IEEE Journal of Selected Topics in Signal Processing (J-STSP).*

5. *L. Cui, R. Mekuria, M. Preda and E. S. Jang, "Point-Cloud Compression: Moving Picture Experts Group's New Standard in 2020," IEEE Consumer Electronics Magazine, vol. 8, no. 4, pp. 17-21, July 2019.*

6. *J. Ballé, D. Minnen, S. Singh, S. J. Hwang and N. Johnston, "Variational Image Compression with a Scale Hyperprior," International Conference on Learning Representations (ICLR'2018), Vancouver, Canada, Apr. 2018.*

7. *T. Lin, P. Goyal, R. Girshick, K. He and Piotr Dollár, "Focal Loss for Dense Object Detection," IEEE International Conference on Computer Vision (ICCV'2017), Venice, Italy, Oct. 2017.*

8. *ISO/IEC JTC1/SC29/WG11 N19084, "Common Test Conditions for Point Cloud Compression," Brussels, Belgium, Jan. 2020.*

9. *D. P. Kingma and J. Ba, "Adam: a Method for Stochastic Optimization," International Conference on Learning Representations (ICLR'2015), San Diego, CA, USA, May 2015.*

10. *ISO/IEC JTC1/SC29/WG1 N88044, "JPEG Pleno Point Cloud Coding Common Test Conditions v3.3," Online Meeting, July 2020.*