**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 7**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 7 m70062**

**Antalya, TR – November 2024**

**Title: AI-PCC CfP Response Proposal from Nanjing University and OPPO (Track2)**

**Authors: Tong Chen, Ruixiang Xue, Junteng Zhang, Jianqian Wang, Zhan Ma (Nanjing University), Shuting Xia, Ruixiang Xue, Zexing Sun, Chuang Ma, Yue Yu, Haoping Yu, Dong Wang (OPPO)**

**Abstract**

This document presents Nanjing University and OPPO’s joint proposal in response to the Call for Proposals (CfP) for AI-based Point Cloud Coding. The method proposed in this document is a learning-based solution that is capable of compressing both geometry and attribute of the input point cloud. It supports static as well as dynamic point cloud input of diverse characteristics in lossy or lossless modes. This response proposal corresponds to the geometryattribute coding to address track 2 of the CfP.

# Information form

Title of the proposal:

AI-PCC CfP Response Proposal from Nanjing University and OPPO (Track2).

Organization: Nanjing University, OPPO

Tong Chen chentong@nju.edu.cn

Test conditions covered by the proposal:

Table Check the appropriate box that applies to the submission.

|  |  |  |
| --- | --- | --- |
| Competition track | Use Case | |
| Dense | Sparse |
| Track 1: Geometry-only | X | X |
| Track 2: Geometry + Attribute | P | X |

Table Fulfillment of requirements for AI-based graphics coding of dynamic point clouds

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Category 1  (Dense Static PCs for Immersive Applications) | | Category 2  (Dense Dynamic PCs for Immersive Applications) | | Category 3. A  (Sparse Dynamic PCs for LiDAR) | |
| **Req.** | **Fulfilled?** | **Req.** | **Fulfilled?** | **Req.** | **Fulfilled?** |
| a) Lossy compression | P | P | P | - | P | - |
| b) Lossless geometry compression | P | P | P | - | P | - |
| c) Lossless attribute compression | P | P | P | - | P | - |
| d) Near-lossless geometry compression | *o* | P | *o* | - | P | - |
| e) Near-lossless attribute compression | *o* | P | *o* | - | *o* | - |
| f) Temporal variations | - | - | P | - | P | - |
| g) Low latency | P | P | P | - | P | - |
| h) Low complexity | P | P | P | - | P | - |
| i) Temporal scalability | - | - | P | - | P | - |
| j) Spatial scalability | *o* | P | *o* | - | *o* | - |
| k) Region-based scalability | *o* | *-* | *o* | *-* | *o* | - |
| l) Quality scalability | *o* | P | *o* | - | *o* | - |
| m) Spatial random access | *o* | *-* | *o* | - | *o* | - |
| n) Temporal random access | - | - | P | - | P | - |
| o) Error resilience | *o* | P | P | - | *o* | - |
| p) Parallel encoding and decoding | *o* | P | *o* | - | *o* | - |
| q) Separable attribute and geometry coding | *o* | P | *o* | - | *o* | - |
| q-1) Geometry only coding | P | P | P | - | P | - |
| q-2) Multiple attribute coding | P | P | P | - | P | - |
| r) Geometry precision | At least Up to 20 | P | At least Up to 12 | - | At least Up to 18 | - |
| s) Model architecture | P | P | P | - | P | - |
| t-1) On the fly Model Update | P | P | P | - | P | - |
| t-2) On-demand Model Update & download | P | P | P | - | P | - |
| u) Inference Reproducibility | P | P | P | - | P | - |

(‘P’ = Required ‘*o*’ = Optional ‘-’ = Not applicable)

Explanations of the requirement items above can be found in the requirement document [1].

# Architecture description

# General

Nanjing University and OPPO’s joint response proposal is a learning-based method that is based on the universal conditional coding framework named *Unicorn* to compress both the geometry and attribute of a point cloud.

# Terminology

To help readers understand *Unicorn*, the frequently used abbreviations are provided in Table 3 for the geometry codec and attribute codec of *Unicorn* respectively.

Table Frequently used abbreviations

|  |  |  |
| --- | --- | --- |
| **Codec** | **Abbreviation** | **Description** |
|  | OPU | Occupancy Processing Unit |
|  | CPA | Conditional Probability Approximation |
|  | CPR | Content-aware Predictive Reconstruction |
| Geometry | MP-POV | Most Probable Positively Occupied Voxel |
|  | PR | Predicting Residual |
|  | PM | Probability Modeling |
|  | OC | Occupancy Classification |
|  | AT | Analysis Transform |
|  | DDS | Dyadic Down-scaling |
|  | APU | Attribute Processing Unit |
|  | CPA | Conditional Probability Approximation |
|  | FeCPA | Feature-space CPA |
|  | AveP | Average Pooling |
| Attribute | AQL | Adjustable Quantization Layer |
|  | AT | Analysis Transform |
|  | ST | Synthesis Transform |
|  | GaU | Geometry-Aware Updating |
|  | GaS | Geometry-Aware Skipping |
|  | UnP | Unpooling |

# *Unicorn* general architecture

*Unicorn* first progressively downsamples the input point cloud to generate the multiscale sparse tensor representation. Upon this representation, the compression of geometry or attribute starts from its spatially lowest-scale tensor and finally arrives at the highest-scale tensor. The cross-scale occupancy processing unit (OPU) and the attribute processing unit (APU) are used for geometry and attribute compression, respectively.

Figure 1 and Figure 2 show high-level block diagrams of the proposed encoding and decoding processes, respectively.

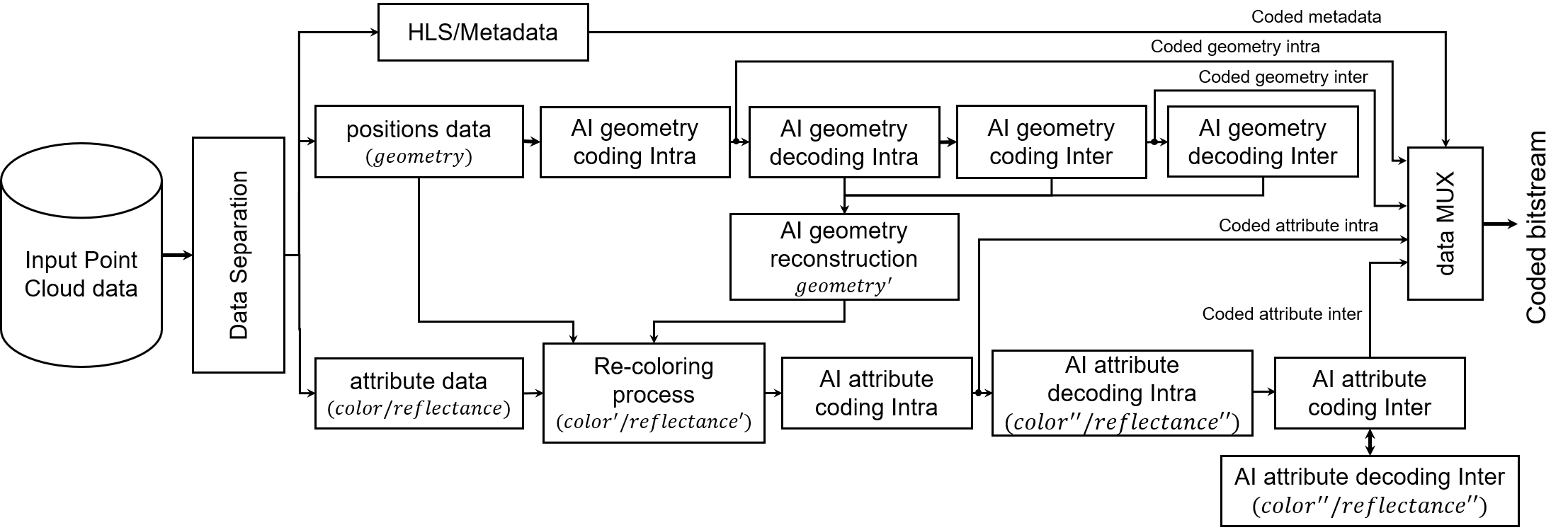


Figure Unicorn encoder architecture

The encoding process is based on a sequential coding of geometry followed by attribute coding. An attribute transfer module (the re-coloring process in Figure 1) is used for adjusting attribute values in case of lossy geometry coding.

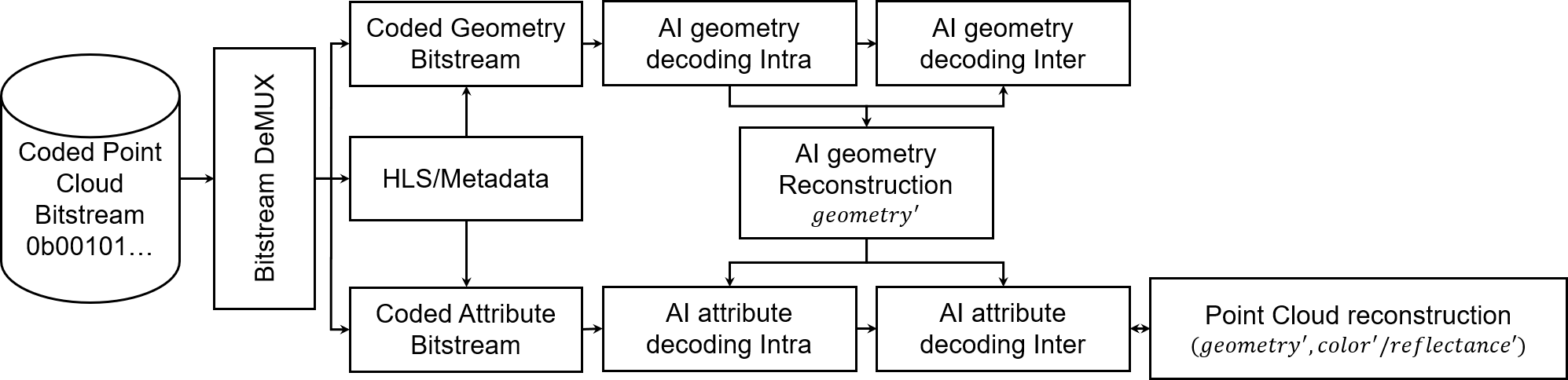


Figure Unicorn decoder architecture

# Geometry codec module

# General

This section provides a description of the geometry coding process of *Unicorn*. A dynamic point cloud , where is a collection of static point cloud frames over time, as illustrated in the left of Figure 3. The geometry and attribute components of are separately processed as shown in the right of Figure 3, and the geometry component is first compressed.

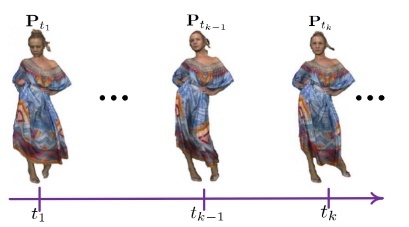
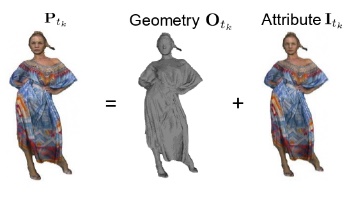
 

Figure Data processing in Unicorn

*Unicorn* first progressively downsamples the input to generate multiscale sparse tensors , where .The dyadic down-scaling (DDS) is applied to for squeezing every eight inter-connected voxels into a single merged one when decreasing the ：

As shown in Figure 4, as long as there is at least one occupied voxel in each group of eight inter-connected voxels, the merged voxel is an occupied one.

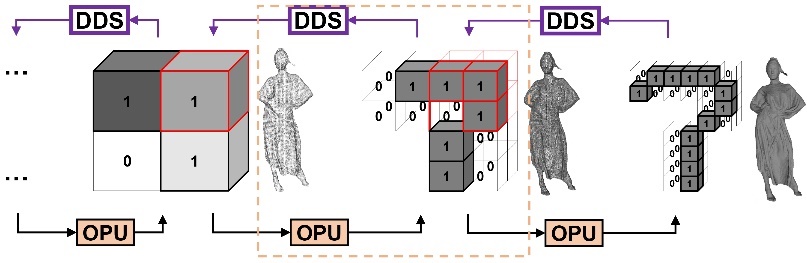


Figure Unicorn's geometry multiscale sparse representation, 1 - Occupied voxel, 0 - Unoccupied voxel. OPU is the occupancy processing unit.

Upon multiscale sparse tensors being generated, the compression of geometry component starts from its spatially lowest-scale tensor and finally arrives at the highest scale to process . The cross-scale occupancy processing unit is termed as OPU.

# Framework of geometry codec

The geometry coding of *Unicorn* is based on a two-stage coding and consists of lossless and lossy components. The architecture of *Unicorn* is based on multiscale sparse tensor representation, and at each scale it receives previous scale priors and performs a set of corresponding computations (lossless OPU or lossy OPU). The lossless geometry coding is based on estimating the occupancy probability of the most probable positively occupied voxel (MP-POV) in each decomposition scale, and it is referred as lossless OPU. The lossy geometry coding is based on the content-aware predictive reconstruction, and it is referred as lossy OPU.

Figure 5 and Figure 6 illustrate the lossless and lossy modes of *Unicorn*'s static geometry coding, and Figure 8 reveals the meaning of each component.

* In the lossless mode, a lossless OPU process runs at each scale from to to estimate the occupancy probability of every relevant voxel, i.e., MP-POV.
* The lossy mode comprises lossless and lossy coding phases consecutively. First, a lossless OPU process is employed from to . Then, a lossy OPU process is applied through the remaining scales from to . Adapting can provide multiple discrete rate points with a single model.

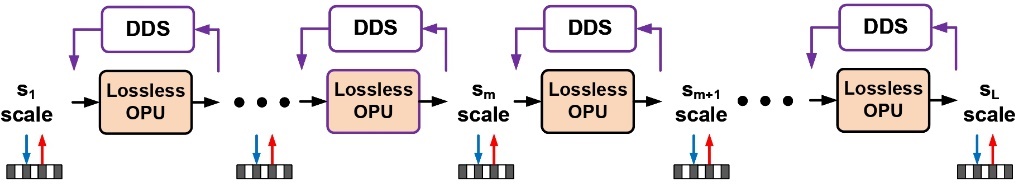


Figure Lossless geometry coding of Unicorn

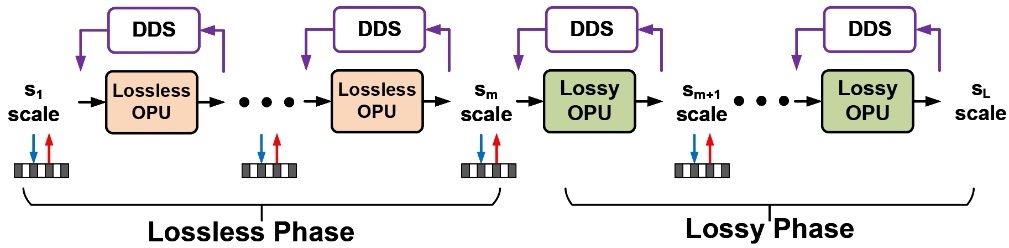


Figure Lossy geometry coding of Unicorn

Figure 7 further depicts the dynamic geometry coding, which can be simply facilitated in *Unicorn* by aggregating and warping priors from the temporal reference frame at to enhance OPU when encoding the frame at . Figure 8 reveals the meaning of each component.

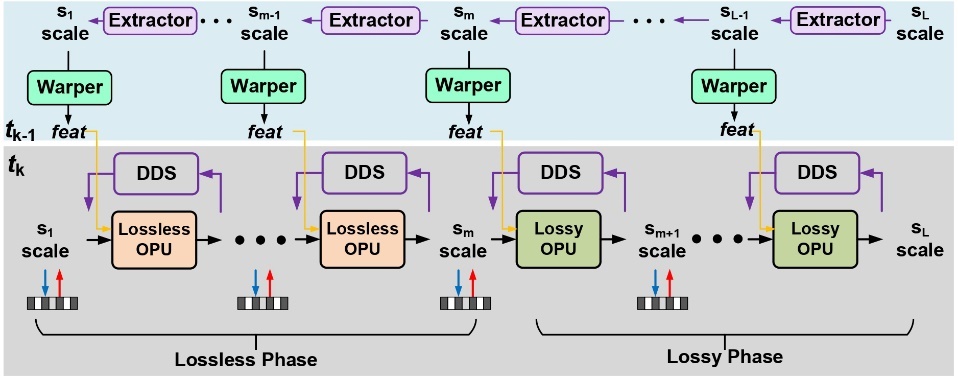


Figure Lossy dynamic geometry coding of Unicorn



Figure Legends of each component

# Model description for static geometry coding

# General

This section contains the description of two geometry coding methods with occupancy processing units (OPU), i.e., lossless OPU and lossy OPU. First, the details of lossless OPU are provided, which is followed by the operation description of the lossy OPU.

# Lossless OPU

For a given geometry tensor , a lossless OPU process takes inputs from the lower-scale prior and executes the conditional probability approximation (CPA) of each relevant element in the superset of , i.e., MP-POV for arithmetic coding, e.g.,

Each POV in is dyadically upscaled to a 2x2x2 3D patch whereas each input is MP-POVs. These patches are geometrically aligned to a corresponding 2x2x2 voxel cuboid that consists of POVs and inter-connected non-POVs in . Each MP-POV represents a probability that is subsequently used to support lossless compression of the corresponding voxel element (POV or non-POV).

Following the extensive studies in SparsePCGC [2], multistage CPA is applied in lossless OPU for its justified performance-complexity trade-off. Such a multistage procedure partitions voxels into groups to appreciate cross-group correlations. For instance, elements are partitioned into eight groups according to their geometric positions in each cube, i.e., , . Raster scanning is used to order the geometric position for grouping. The geometric position of the voxel in the -th group is highlighted in one local patch as illustrated in the upper part of Figure 9. All these voxels belong to the same -th group form the . Parallel processing is inherently supported for all voxel elements in the same group.

The POVs in the previously reconstructed groups are used to improve the probability estimation of voxels in the latter groups.

is used to collect all POVs reconstructed before -th group, denotes concatenation process. in is upscaled to the resolution of after passing through the first CPA for subsequent computation.

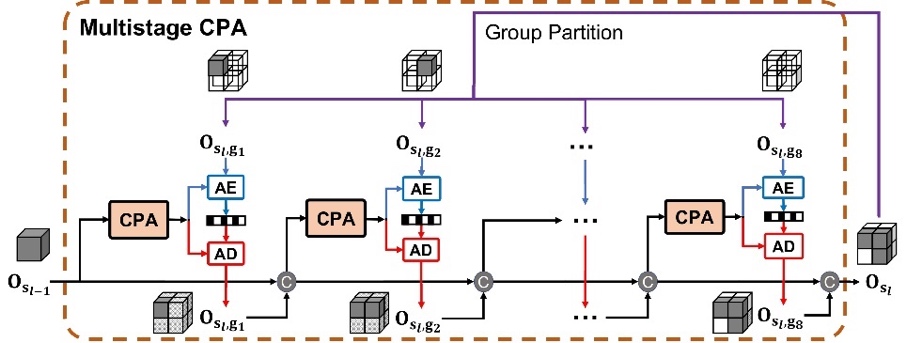


Figure Lossless OPU

# Lossy OPU

Different from lossless OPU, each MP-POV in lossy OPU is used to conduct the probabilistic occupancy classification of POV or non-POV. The lossless OPU minimizes the bitrate consumption by improving the efficiency of CPA to obtain more accurate contextual probability for entropy coding. In contrast, lossy OPU jointly optimize the bits consumption and distortion measures to improve overall R-D performance [3].

The fixed reconstruction process usually involves point vanishing and displacement, producing a visually unpleasant appearance. Point vanishing or displacement is mainly attributed to the resolution downsampling involved in the proposed multiscale representation and other octree-based approaches, and they are highly correlated with content sparsity. For instance, point vanishing dominates when downsampling a solid object point cloud, while point displacement is more visible when downscaling a scant scene sample.

Thus, the scale-dependent fractal dimension is used to quantify a point cloud’s sparsity in [4], i.e., . Here and  are the total number of occupied voxels at scale and , which are carried in the bitstream as the metadata.

Figure 10 illustrates fractal dimension indices for representative point clouds at different bit precisions. As seen, such a scale-wise fractal dimension index reflects the geometric sparsity discussed above.

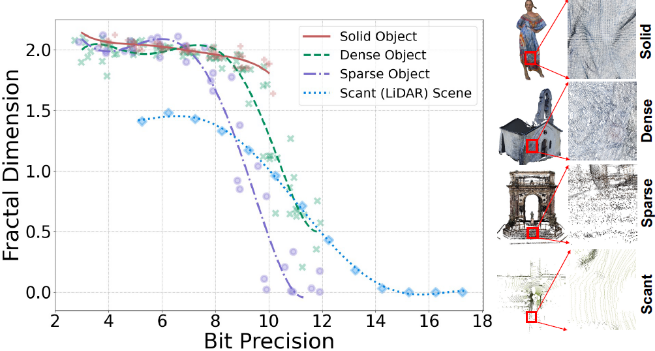


Figure Fractal dimension vs. bit precision for solid/dense/sparse objects and scant scene samples

The content-aware predictive reconstruction (CPR) is proposed to mitigate impairments and reduce distortion. Such a CPR process consists of CPR-V (see Section 2.4.3.3.2) and CPR-D (see Section 2.4.3.3.1) components. Having scale-wise metadata  encapsulated in the compressed bitstream, the can be derived on the fly to adapt CPR-V and CPR-D intelligently. Next, the two basic components of CPR, i.e., CPR-D and CPR-V are explained in detail, which are also illustrated in Figure 11.

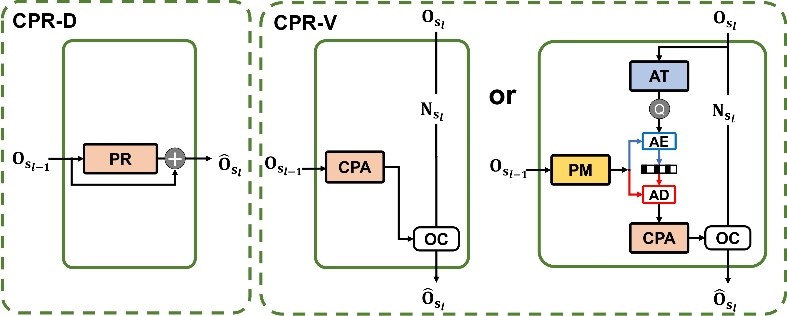


Figure Lossy OPU

# Coordinate refinement of displaced points via CPR-D

The left subplot in Figure 11 depicts the proposed CPR-D. It takes inputs (decoded) from lower-scale prior for predicting residual (PR) so that coordinates can be properly refined as follows:

It is noted that if the occupancy tensor is losslessly compressed at scale ; and if lossy compression is used with quantization noises. Resolution scaling is enforced in this function implicitly.

# Restoring vanished points via CPR-V

The right subplot of Figure 11 visualizes the CPR-V. Similarly, an occupied voxel in is upscaled to eight possibly-occupied voxels at first. As a result, restoring vanished points is equivalent to the occupancy classification (OC) of the possibly-occupied voxel. Assuming a possibly-occupied voxel with a higher probability will have a higher possibility of being an occupied voxel, the core issue again becomes its probability approximation, which is the same as that in lossless OPU.

**Feature augmented CPA.** It turns out that multistage CPA used in lossless OPU is unsuitable in lossy OPU as lossy compression-induced incorrect classification of voxel's occupancy in earlier stages would propagate to the latter ones, and such an error accumulation is very difficult to be characterized and resolved. The naive CPA shown in Equation (1), is a solution, but its probability estimation efficiency is usually limited due to simple one-stage processing.

Recalling that multistage CPA exploits fine-grained correlation from same-scale reconstructed neighbors i.e., through sub-scale grouping for more accurate probability approximation, an alternative solution is to aggregate and embed neighborhood correlation as an auxiliary feature payload to enhance the capacity of CPA. It is called feature augmented CPA for short.

In feature augmented CPA, neural network layers form an analysis transform (AT) to generate feature-space neighborhood embedding which is then quantized and compressed with a neural probability modeling (PM) module that generates the conditional context using (decoded) lower-scale prior. Decoded neighborhood embedding is accordingly attached to the lower-scale prior's coordinates to form for probability estimation, i.e.,

As seen, it can be deduced to if neighborhood embedding is not used, e.g., .

**OC.** Given possibly-occupied voxels and their associated probabilities, a two-step OC decision is then applied. Possibly-occupied voxels at scale can be clustered accordingly so that each patch consisting of eight inter-connected elements, a.k.a a cube, is from the same occupied voxel in . As a result, at least one sub-voxel in each cube patch is occupied. At the first step, a possibly-occupied voxel with the highest probability in each patch is marked as the occupied voxel.

The remaining possibly-occupied voxels are sorted in probability descending order and transform the first elements into occupied voxels accordingly. Here . It is noted that occupied voxels are already determined in the first step. Finally, the same number of occupied voxels are restored as the ground truth at each scale.

# Model description for dynamic geometry coding\*

(In mechanism, *Unicorn* supports dynamic geometry coding. However, currently, the results of the dynamic sparse and dynamic dense categories are unavailable in the attached spreadsheet. We mark this section with \*)

This section contains a description of the proposed dynamic geometry coding. The proposed framework enables dynamic coding by warping temporal priors into the lossless and lossy OPUs to exploit spatiotemporal correlations jointly.

As shown in Figure 7, multiscale temporal priors are generated from a (decoded) temporal point cloud reference . Formulating is similar to the generation of multiscale sparse tenors of the current frame discussed in Section 2.4.1. Instead of simply reusing the DDS, the Extractor is proposed to progressively aggregate and embed neighborhood correlations when performing the geometry downsampling dyadically. Thus, temporal prior at scale comprises the geometry coordinates and associated features of all occupied voxels, i.e., .

As temporal motion is inevitable across consecutive frames, the scale-wise temporal priors must be properly warped to the current frame for conditional coding. Here, the Warper is proposed in which it applies a target convolution to transfer the reference's features to the current frame so that spatiotemporal prior is aggregated for lossless or lossy compression in Equation (2) and Equation (4), i.e.,

Figure 12 details the modular unit to exploit spatiotemporal correlations for dynamic coding. Extractor and Warper are implemented by stacking neural network layers (see Figure 13). A target convolutional layer with a fixed kernel size of is utilized in the Warper model to aggregate spatiotemporal prior for dynamic coding.

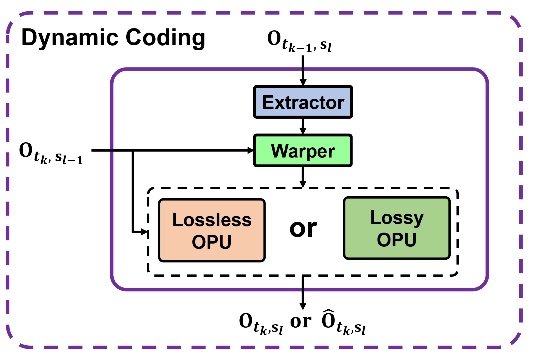


Figure Models in dynamic coding

# 2.4.5. Functional model and neural network

# 2.4.5.1. Functional model

The main functional models of *Unicorn*, i.e., the multistage CPA, the feature augmented CPA and the one-stage CPA, as well as the PR model, are powered with neural networks.

Other neural network modules including AT, PM, Extractor, Warper, etc., are integrated with these functional models to enable desired functionalities.

For example, PM and AT are used in feature augmented CPA, while Extractor and Warper enable dynamic coding.

Figure 13 sketches the neural network blocks used to implement these functional models:

* A DNN block can be fulfilled by simply stacking three Inception ResNet (IRN) blocks in [2] or three NPFormer blocks, as shown in Figure 17. IRN comprises sparse convolutions, while NPFormer relies on neighborhood point attention (NPA).
* Down is a sparse convolution layer (), and is integrated with DNN blocks to build up the AT or Extractor units for neighborhood correlation characterization and embedding.
* Up is a transposed sparse convolution layer () applied to expand an occupied voxel at scale to a corresponding possibly-occupied voxel patch at scale as in the CPA model.

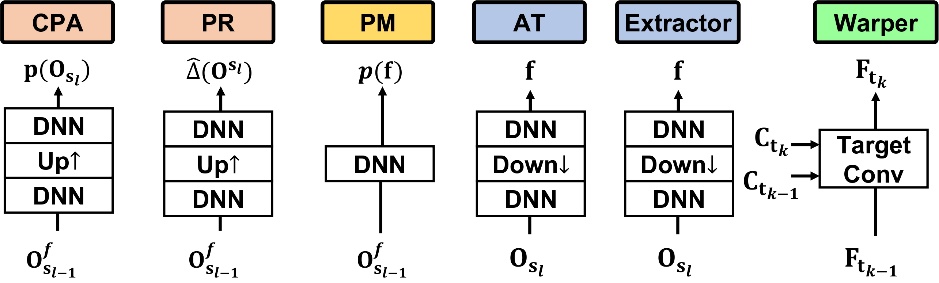


Figure Functional models implemented using neural networks

It is noted that the structure of the Warper module can be different from the current implementation, and it is referred as the new Warper (see Figure 14).

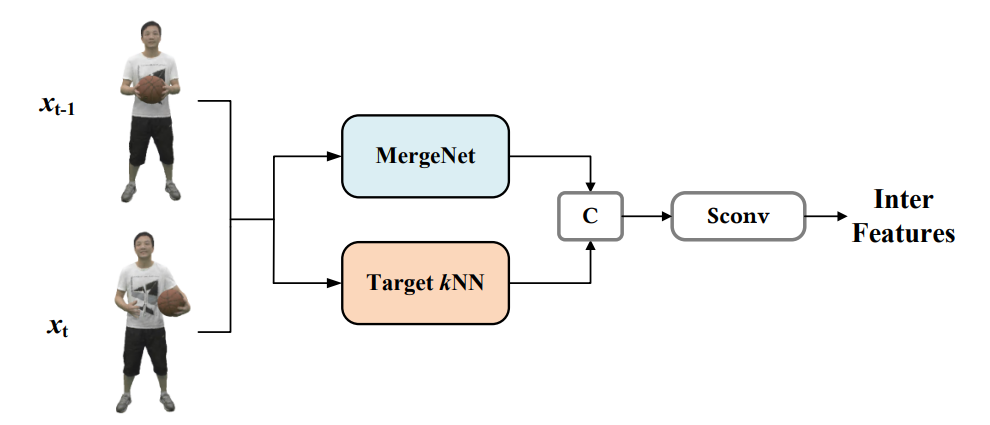


Figure Structure of new Warper

This new Warper mainly consists of MergeNet and Target KNN. The MergeNet is used to substitute the target convolution layer, whose core idea, i.e., Merge operation is to jointly convolve the two reference frames and the current frame according to the following formula. The specific formula and module structure are shown in Figure 15 below:



Figure Structure of MergeNet

The structure of TargetKNN is shown in Figure 16.

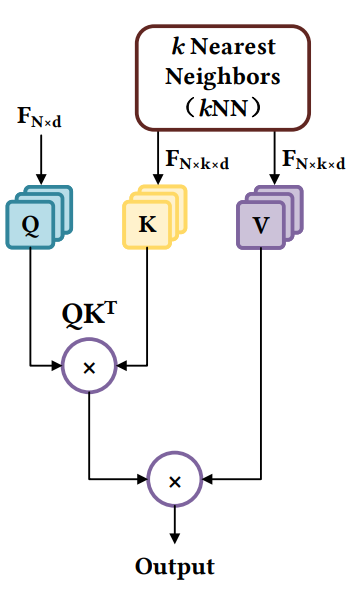
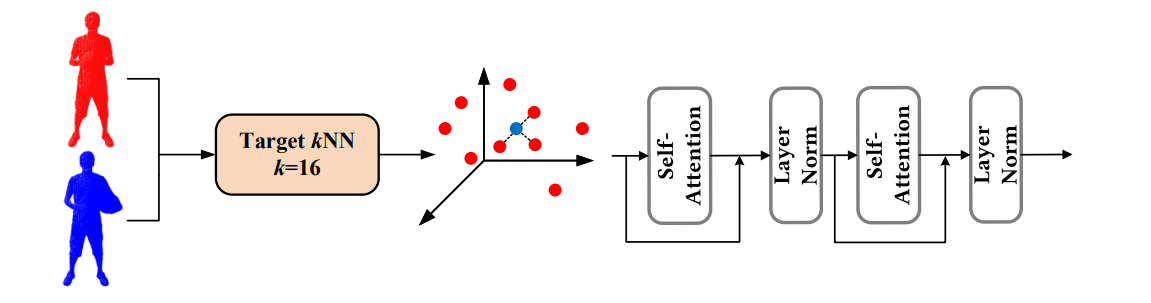


Figure Structure of TargetKNN

This new Warper decreases the storage size of modules (from 9.1MB to 7.0MB) and greatly improves the compression performance on the test sequences of the dynamic dense category. But its compression gain on the test sequences of the dynamic sparse category is not obvious, and it is only used on the test sequences of the dynamic dense category. It also proves the great scalability of our proposed framework.

# 2.4.5.2. Sparse convolution

A sparse tensor, e.g., an occupancy tensor , comprises a set of coordinates and associated features , i.e., . Thus, the sparse convolution is formulated as:

where and are the input and output coordinates, respectively. if the resolution is remains unchanged.

and are the input and output feature vectors at coordinate = (, , ).

defines a 3D local patch with a size of or and the center at in for information aggregation.

**Target convolution** can be supported by setting and in Equation (5) in different frames (at different time stamps), such as the temporal reference and the current frame used in dynamic coding. Therefore, the convolution aggregates related latent features in the reference and transfers them to in the current frame.

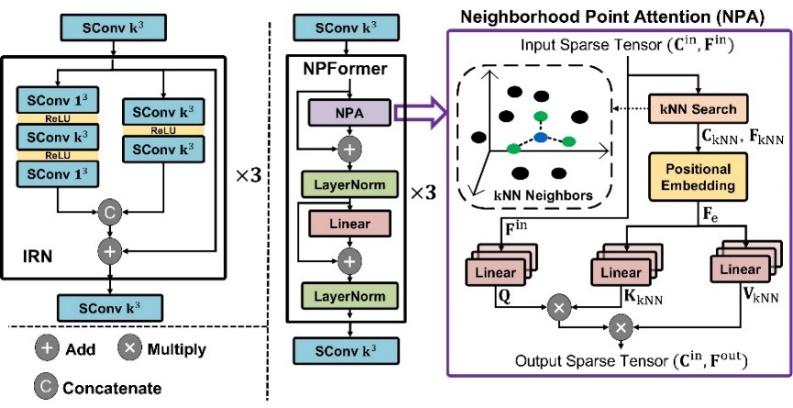


Figure IRN (Left) and NPFormer (Right) used to form the DNN block

# 2.4.5.3. Neighborhood point attention

The architecture of NPA is shown on the right of Figure 17. Assuming the input of a NPA layer is the sparse tensor consisting of coordinates and features , the NN search is conducted for each element in , forming tensors , which is then extended using relative positions, i.e.,

This process is referred to as the positional embedding.

Let , and be the , , and vectors respectively. is linearly transformed from ; and are computed by another two separate linear transformations of . The weights of these three linear transformations are , , and , respectively. Then, the NPA is

The output of NPA is . assumes the same resolution involved in NPA.

The NN neighborhood and the attention mechanism in NPA facilitate the network to adaptively exploit local correlations regardless of the varying density of the underlying content. In contrast, the fixed receptive field setting in sparse convolutions may not be able to include sufficient (and valid) neighbors, especially for sparse contents.

# Training

Functional models, e.g., multistage CPA, feature augmented CPA, one-stage CPA, and PR model, are trained independently, assuming the static coding first. The training is limited to using the data from two consecutive (spatial) scales, and the trained models are applied to any scale without limitations. They are later fine-tuned with the inclusion of Extractor and Warper to support dynamic coding, where two consecutive point cloud frames (from a dynamic sequence) are used.

For a given application scenario, e.g., lossy or lossless compression, static or dynamic coding, solid or scant content, etc., these functional models are properly composited to fulfill the purpose.

All CPA models, i.e., multistage CPA, feature augmented CPA and one-stage CPA try to predict possibly-occupied voxel's occupancy probability. Thus, they first employ binary-cross-entropy (BCE) loss in training:

where is the ground-truth occupancy status: 1 for occupied or 0 for unoccupied and is predicted probability. is the number of original input points used for normalization.

Since feature augmented CPA includes an auxiliary embedding process to characterize neighborhood correlation, an additional cross entropy (CE) loss is augmented:

where the context probability of coding latent feature is conditioned on generated by the probability model using lower-scale prior.

Thus, the total loss used to train feature augmented CPA is , where adjusts the rate-distortion trade-off in lossy coding and is typically set to 1 by default. The adjustment of can control the rate-distortion trade-off to provide more rate points.

The PR model for refining coordinates in CPR-D directly predicts residuals to approach the ground truth. It is optimized using mean square error (MSE) loss:

where , , are the ground-truth residuals, and , , are the predicted residuals.

Further training details, e.g., epoch, learning rate, and device can be found in the attached sheet file.

# Geometry quality scalability

In the geometry codec of *Unicorn*, the compression quality can be adjusted by two separate approaches, which makes the compressed geometry quality scalable possible:

* As described in Section 2.4.2, the geometry coding model consists of two parts, i.e., lossless and lossy phase, which are achieved by lossless OPU and lossy OPU respectively. Adjusting the number of scales allocated to these two phases through adapting the can provide multiple discrete rate points with a single model.
* Apart from controlling the scale factor , *Unicorn* can further realize finer-grained rate control by adjusting the weight of rate-distortion optimization in the feature augmented CPA module during training, i.e., in its loss function . The details can be found in Section 2.4.6. The feature augmented CPA can also be substituted as a one-stage CPA directly.

# Attribute transfer module

This section describes the attribute transfer module in *Unicorn*. It is worth pointing out that it is the separate processing of geometry and attribute information in *Unicorn* that allows us to engineer a unified framework to fulfill the purpose above conveniently. To this end, the original attribute tensor must be re-mapped to the reconstructed geometry  before running the attribute coder to code it. Here, the output of geometry coder  is generally assumed with compression noise.

The improved color transfer algorithm [5] in G-PCC is used in the *Unicorn* platform. For each point of the target :

1. Find the (1 < ) nearest neighbors in source to and create a set of points denoted by .
2. Find the set of source points that belongs to their set of nearest neighbors. Denote this set of points by .
3. Compute the distance-weighted average of points in and by:

where denotes the Euclidian distance between the points and , and denotes the color of point .

1. Compute the average (or the weighted average with the number of points of each set as the weights) of and and transfer it to .

# Attribute codec module

# General

This section provides a description of the attribute coding process of *Unicorn*. A dynamic point cloud , where is a collection of static point cloud frames over time, as illustrated in the left of Figure 3. The geometry and attribute components of is separately processed as shown in the right of Figure 3. The geometry occupancy is first compressed, and an attribute transfer algorithm (see Section 2.5) is carried out to calculate the recolored point cloud . Finally, the from the recoloring process is compressed. For simplicity, the is used to represent the attribute component.

Average pooling (AveP) is applied to to derive the attribute intensity of the corresponding merged voxel:

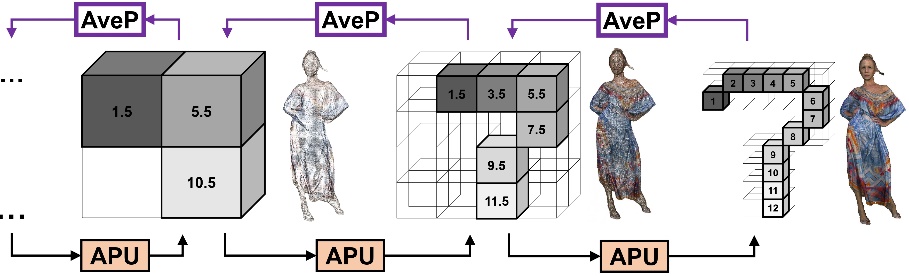


Figure Unicorn's attribute multiscale sparse representation, color attribute exemplified using luma or Y intensity. APU is the attribute processing unit.

As shown in Figure 18, upon multiscale sparse tensors being generated, the compression of attribute intensity starts from its spatially lowest-scale tensor and finally arrives at the highest scale to process . The cross-scale attribute processing units are termed as APU.

# Model description for static attribute coding

The lossless attribute compression is first discussed in Section 2.6.2.1 and the lossy mode is followed in Section 2.6.2.2. Here, the discussion is focused on static coding using spatial priors only.

# Lossless attribute compression

To compress at scale , its lower-scale reconstruction is available as the conditional prior . As for , its -th occupied voxel , is first geometrically expanded to a local cube with eight child nodes (a.k.a., octree expansion). Since the geometry occupancy is available in advance when processing attribute components of a point cloud, known occupied voxels in this cube is filled with the same , e.g.,

Such a step is referred as the unpooling (UnP) process, which is iterated for all the elements in to produce a tensor as the initial prediction of , i.e.,

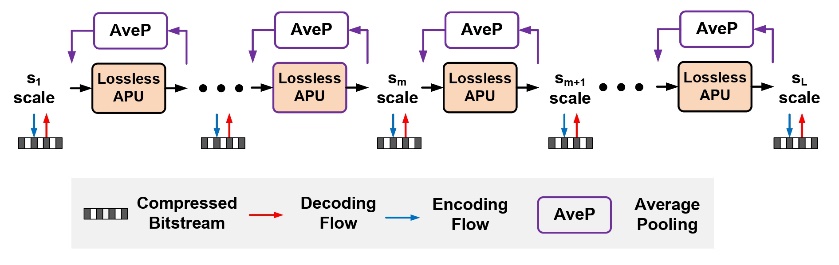


Figure Lossless attribute coder, lossless APUs are uniformly devised for conditional probability approximation (CPA) in compression across all scales

**CPA**. Subsequently, the conditional probability approximation (CPA) model takes inputs to derive the probability for each element in , e.g.,

The processing of Equation (14) comprises two steps:

* Generating means and variances for all the elements assuming the Laplacian distributed probability density function ;
* Deducing the probability by integrating via

to encode into the bitstream; or decoding a level from the bitstream to reverse Equation (15) for reconstructing . is a uniform distribution ranging from to .

The proposed CPA in Equation (14) can be facilitated using a DNN model.

As the UnP process in Equation (13) offers an initial approximation of the current-scale attribute intensity, in practice, the difference between and is predicted, which is implemented by adding a residual connection in CPA's neural model.

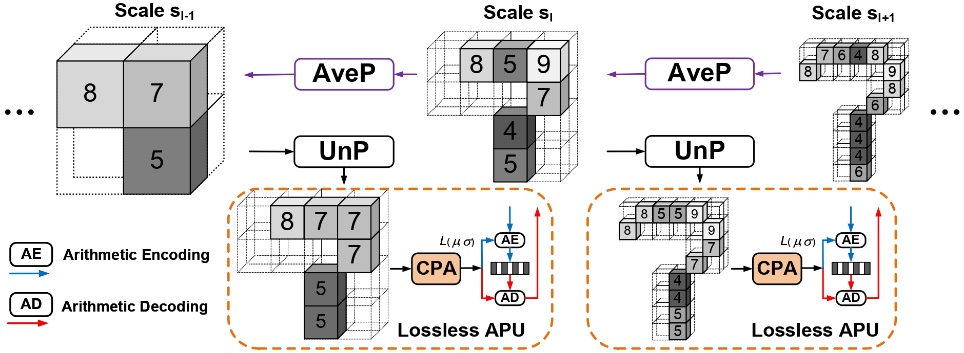


Figure One-stage CPA, the Unpooling (UnP) block upsamples the lower-scale reconstruction as the initial prediction of current-scale attribute for CPA

**Multistage CPA**. CPA in Equation (14) performs the estimation for all the elements in . It can be extended to process elements from one group to another in a multistage manner, where the nearby neighboring elements from the same scale in previously processed groups can be further utilized to exploit correlations better. It is dubbed multistage CPA.

Here, an example of using multistage CPA to process with eight groups is given in Figure 21.

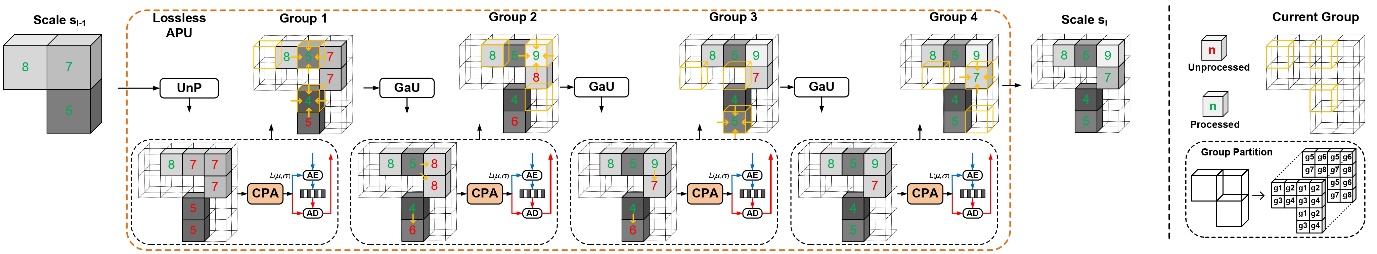


Figure Multistage CPA, it is fulfilled by stacking multiple CPA to process grouped elements sequentially, in which the geometry-aware updating (GaU) is applied to update the initial prediction using geometry prior and previously-processed same-scale neighbors for more accurate CPA. Here, single-channel attributes are illustrated with the intensity levels annotated.

Specifically, is organized as a collection of cubes and, each of them contains eight inter-connected voxels (e.g., some of them are occupied voxels and the remaining ones are unoccupied voxels). For each cube, the voxel with its group identification according to the geometric position in a raster scanning order (see ``g1'' to ``g8'' in the legend plot) is marked. As a result, a total of eight groups are specified, having .

Instead of making a one-shot probability estimation for all the elements in as in Equation (14), CPA is applied in a group-wise fashion, where both lower-scale reconstruction (after the UnP process) and previously processed elements in preceding groups, e.g., are employed as the spatial priors for conditional prediction. Here, the is used to include all elements before -th group.

As the occupied voxels of each cube in are known in advance, the is processed and updated from one group to another before feeding it to CPA, which is so-called as geometry-aware updating (GaU), i.e.

is the number of occupied voxels in each cube and .

When only one occupied voxel exists in a cube, its ground-truth value can be inferred directly without further steps, which is referred to as geometry-aware skipping (GaS). For example, the intensity level 8 of upper-left occupied voxel at scale directly infers the attribute of the corresponding solely occupied voxel in upscaled cube at scale , as shown in Figure 21

**Remark**. Previous examinations assume the processing of a single-channel attribute component, which can be directly used for single-channel of a LiDAR point cloud.

For a colorized object point cloud with three-channel colors, its color space is often converted from to for lossless coding. Using can remove

cross-color correlation among , , and components to some extent, which, thus, benefits the compression. Furthermore, and components are simultaneously processed conditioning on the prior.

It is noted that transforming to can also reduce the cross-color redundancy, but it is not mathematically lossless. For lossy attribute compression, color space [5] is used. Ideally, the attribute can be first compressed, then with the help of and with the help of both and .

# Lossy attribute compression

Figure 22 illustrates the lossy PCAC solution, where it applies the lossless APU from scale to and constitutes lossy APU thereafter till .

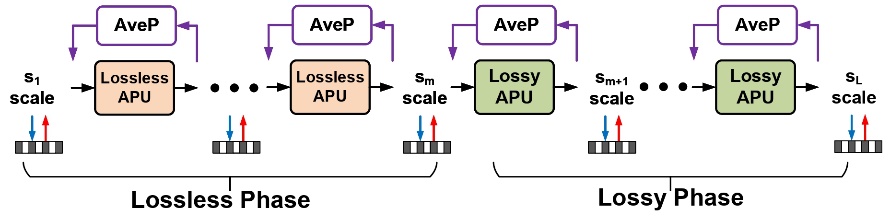


Figure Lossy attribute coder

**Attribute residual generation**. For a given scale , the attribute tensor is and the corresponding attribute residual is derived via

where is the cross-scale prediction generated by the lower-scale reconstruction . Here, we assume it is the quantization noise augmentation that differentiates the reconstruction tensor from its uncompressed counterpart in lossy mode. It is also worth pointing out that the UnP process is discussed in Section 2.6.2.1 above.

**Transform Quantization.** Then, attribute residual is transformed and quantized, e.g.,

The sparse convolutional layers are stacked to form the analysis transform (AT) and synthesis transform (ST) using an autoencoder structure. The adjustable quantization layer shown in Figure 23 is introduced for variable-rate coding using a single neural model, which significantly reduces the complexity. Such an AQL mechanism usually scales latent elements in in encoding and decoding to fulfill the purpose, and the scaling can be implemented using either numerical factors or network models [6] [7].

**FeCPA**. To perform the entropy coding of , proper contexts models that can help the notably reduction of the bitrate consumption [8] must be constructed. Here, lower-scale priors are utilized as the conditional knowledge for probability modeling, e.g.,

where and stand for lower-scale reconstructions of attribute residual and attribute intensity, respectively. They are concatenated and fed into the FeCPA unit. Although it is possible only using the closest lower-scale to model the current-scale residual in latent space, having can provide more conditional knowledge, because recurrently aggregates the residual information from all preceding scales through

Note that the processing steps in Equation (19) are similar to Equation (14). The only difference lies in element type: in the lossless mode, the probability of attribute intensity itself is estimated, while in the lossy mode, the estimation is conducted for transform-domain attribute residual in the latent feature space. Thus, it is called feature-space CPA (FeCPA) in Equation (19).

**Remark**. The lossy attribute coder progressively refines the reconstruction scale by scale, offering a coarse-to-fine representation through the transform-domain conditional coding of attribute residual. Theoretically, it might be possible to conditionally compress the attribute intensity rather than the attribute residual in this work. However, according to our extensive studies, the model used for conditional compressing the attribute residual is more robust and converges much faster. This is because the attribute residual exhibits a more clustered distribution that is much easier to characterize.

In the lossless mode, GaU and GaS are applied in multistage CPA to infer and update attribute intensity without requiring any extra bitstream signaling (see Figure 21). Here, the GaS in the lossy mode is applied to leverage the geometry prior. Assuming only one voxel is occupied in a cube at scale , it is directly inferred from its octree parent at .

In lossy mode, RGB colors are transformed to YUV space for compression. Unlike the lossless APU in which individual color component is processed separately, lossy APUs compress three-channel YUV together.

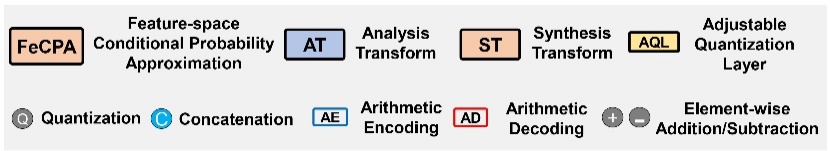
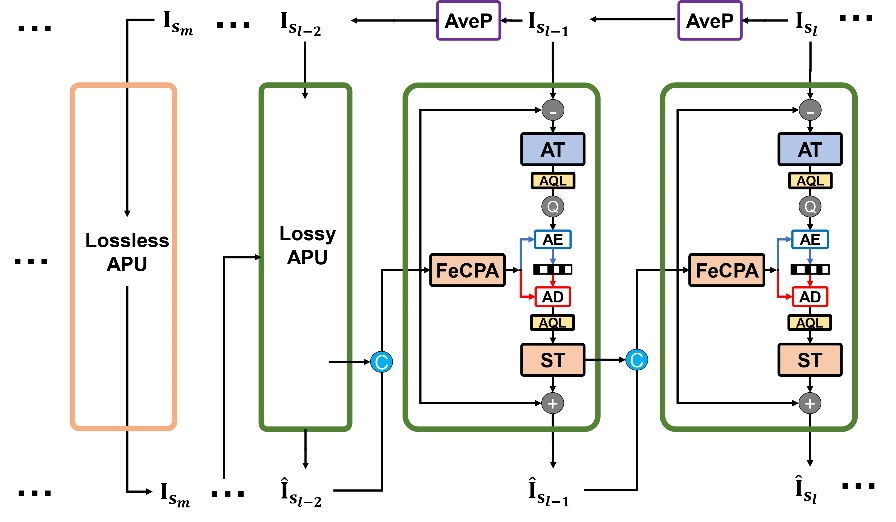


Figure Lossy APU, it adopts Transform-domain conditional residual coding to progressively refine the attribute reconstructions scale by scale

# Model description for dynamic attribute coding\*

(In mechanism, *Unicorn* supports dynamic attribute coding. However, currently, the results of the dynamic sparse and dynamic dense categories are unavailable in the attached spreadsheet. We mark this section with \*)

Previous discussions assume the static attribute coding, for which each point cloud frame is compressed independently, and the spatial prior is only leveraged in the current frame.

On the contrary, enabling dynamic attribute coding allows us to exploit correlations from both the current and temporal reference frames for achieving a better compression.

In Figure 24, the Extractor is proposed to progressively aggregate and embed spatial features from to to formulate multiscale temporal priors from a temporal point cloud reference . These priors are attached with the corresponding coordinates since the geometry information is known in advance.

To alleviate the impact of temporal motion across consecutive frames, the scale-wise temporal priors to the current frame are warped for conditional coding. To this end, Warper is devised to apply a target convolution to transfer the reference's attribute features to the current frame so that the spatial and temporal priors are concatenated and fed into lossy or lossless APUs for probability approximation.

The proposed Extractor and Warper modules are implemented by stacking neural network layers, as shown in Figure 25. A target convolutional layer with a fixed kernel size of is utilized in Warper to aggregate spatiotemporal priors for dynamic coding.

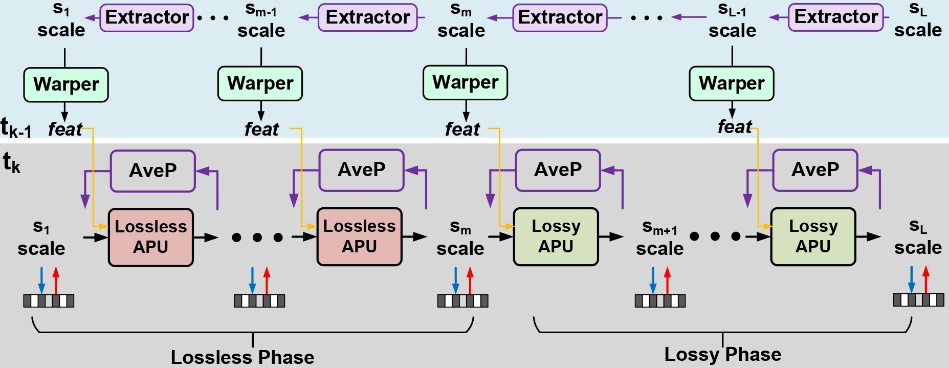


Figure Lossy dynamic attribute coder, multiscale temporal priors are generated using Extractor, transferred using Warper, and concatenated with the spatial prior for the processing in lossless and lossy APUs

# Functional model and neural network

Functional models, including CPA and FeCPA, power the attribute codec of *Unicorn*. Other modules, like AT, ST, Extractor, and Warper, are integrated with functional models for compression.

Figure 25 shows the neural network blocks in our implementation:

1. A DNN block stacks three ResNet blocks, mainly comprising sparse convolutions (SConv) using 128 channels and kernel size, e.g., C128, K3.

* Such a DNN block is used in CPA to estimate the probability of element intensity in attribute space and is applied in FeCPA to estimate the probability of transform-domain attribute residual in latent space.
* It is also used in AT, ST, and Extractor units to aggregate and embed neighborhood characteristics as latent features.

1. Down is a sparse convolution layer (), and it is typically integrated with DNN blocks to build up AT or Extractor unit.
2. Up is a transposed sparse convolution layer (), and it is applied in the ST unit with DNN blocks to upscale latent features.
3. AQL uses a simple MLP (MultiLayer Perceptron) structure to properly scale latent features for variable-rate coding.

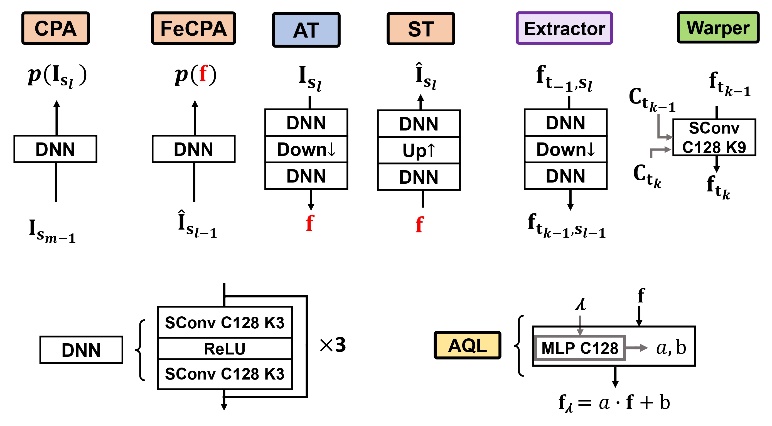


Figure Neural network units for functional models

As there are no obvious performance gains when using the neighborhood self-attention (see Section 2.4.5.3) to build up the neural network units, the sparse convolution is used for simplicity.

As discussed in Section 2.4.5.1, the original Warper module is substituted as the new Warper module on the test sequence of dynamic dense category in the attribute codec part.

# Training

In training, the cross-entropy is used to measure the attribute bitrate in lossless coding, as well as the bitrate of latent features in lossy coding. In the meantime, the mean square error (MSE) loss between the original and reconstructed attributes is used for distortion measurement in lossy coding.

Since the multiscale representation framework is used in *Unicorn*, the total loss is the sum of the losses at all scales (and sub-scales), and the corresponding model weights are determined in an end-to-end manner.

As for the lossless coding mode, its loss function is:

where stands for the total number of scales. denotes three-channel YCoCg color attribute, while denotes single-channel reflectance. is set to .

is the number of points for color component in group at scale ; and is estimated using Equation (15).

On the other hand, the loss function for lossy coding is

where is the scale index, is the number of points at the scale , and is estimated using Equation (19). The determines the R-D trade-off. During the training, the is randomly set between and for object point clouds, and between and for LiDAR samples to derive a single model supporting variable bitrates.

Further training details, e.g., epoch, learning rate, device, optimizer can be found in the attached spreadsheet.

# Attribute quality scalability

In the attribute codec of *Unicorn*, compression quality can be adjusted by three different methods, which helps make the compression quality scalable:

* Clearly, the quality of geometry has a great effect on the quality of attribute. Through controlling the geometry quality as described in Section 2.4.7, the attribute quality can be adjusted.
* Similar to the *Unicorn’s* geometry codec, the attribute codec also consists of two parts, i.e., lossless and lossy coding, which are achieved by lossless APUs and lossy APUs respectively. Adjusting the number of scales allocated to these two phases, i.e., adapting the can provide multiple discrete rate points with a single model.
* Apart from controlling the scale factor , *Unicorn* can further realize finer-grained rate control by adjusting the in the lossy APU.

# Coded bitstream description

# General

This section describes the coded bitstream structure for compressed point clouds.

# Bitstream structure description

The bitstream format of the proposed solution contains three components:

* A V3C container header;
* A geometry component;
* An attribute component.

The V3C header defines the specifics of the point cloud such as module selection, number of frames in the sequence, and number and type of associated attributes, etc.

The format of the point cloud frame consists of geometry data followed by an attribute data (see Figure 26 and Figure 27).

A statue of a person and person

Description automatically generated

Figure Static point cloud bitstream presentation format

A person in a dress

Description automatically generated

Figure Dynamic point cloud sequence bitstream presentation format

# Geometry component bitstream

The geometry bitstream is represented as a series of OPU outputs, which is comprised of a combination of lossless and lossy OPU coded representations. The decoding process requires sequential reconstruction.

For the lossless phase, the bitstream representing a specific scale can be subdivided into 8 units (stages) corresponding to an octree structure, each corresponding to the occupancy information of the position in the point cloud. It is illustrated in Figure 28.

A diagram of a lossless phase

Description automatically generated

Figure Geometry bitstream (lossless phase)

As for the lossy phase, the highest scale of the lossless phase is first transformed into the latent feature, then quantized, and finally encoded into bitstream via arithmetic coding. The bitstream is comprised of a header that carries the information about a number of points in each scale and arithmetic coded latent feature payload. It is illustrated in Figure 29.

A diagram of a lossless opu

Description automatically generated

Figure Geometry bitstream (lossy phase)

# Attribute component bitstream

Similar to the geometry component, the attribute component is represented as a combination of a set of lossless and lossy coded scales sequentially recoded in the attribute bitstream.  
As shown in Figure 30, the lossless component of the attribute residual of each scale is encoded into bitstream using arithmetic coding. Specifically, the bitstream representing particular scale is subdivided into 8 units (octree subdivision), and each of the subdivisions corresponds to the attribute residual associated with a particular location of a point in a point cloud.

A diagram of a low-cost program

Description automatically generated

Figure Attribute bitstream (lossless phase)

The lossy attribute components corresponding to each scale are transformed into latent feature representation, quantized and encoded into bitstream as shown in Figure 31.

A diagram of a machine

Description automatically generated

Figure Attribute bitstream (lossy phase)

# Bitstream parsing process

Below (see Figure 32) exemplify the bitstream structure of geometry component, attribute component, single frame, and whole sequence.

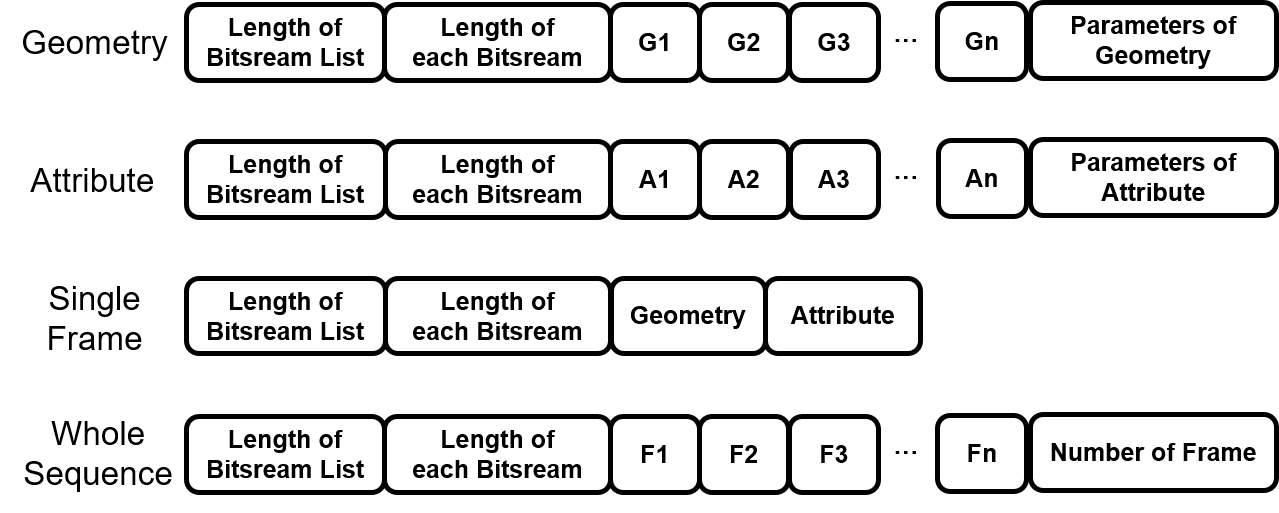


Figure Bitstream structure

The bitstream packaging and reading method is demonstrated in the form of pseudo code below.

Table Bitstream packaging and reading

|  |  |
| --- | --- |
| **Algorithm:** write\_bitstream | **Algorithm:** read\_bitstream |
| **Input:** bitstream\_list, bin\_dir, dtype=’uint32’  **Output:** bin\_size | **Input:** bin\_dir, dtype=’uint32’  **Output:** bitstream\_list |
| **1.** bitstream\_all = np.array(len(bitstream\_list), dtype=dtype).tobytes()  **2.** bitstream\_all += np.array([len(bitstream) for bitstream in bitstream\_list], dtype=dtype).tobytes()  **3.** for bitstream in bitstream\_list:  **4.** assert len(bitstream)<2\*\*32-1  **5.**  bitstream\_all += bitstream  **6.** bitstream\_all += bitstream  **7.** with open(bin\_dir, 'wb') as f:  **8.** f.write(bitstream\_all)  **9.** return os.path.getsize(bin\_dir)\*8 | **1.** with open(bin\_dir, 'rb') as fin:  **2.** bitstream\_all = fin.read()  **3.** s = 0  **4.** num = np.frombuffer(bitstream\_all[s:s+1\*4], dtype=dtype)[0]  **5.** s += 1\*4  **7.** lengths = np.frombuffer(bitstream\_all[s:s+num\*4], dtype=dtype)  **8.** s += num\*4  **9.** bitstream\_list = []  **10.** for l in lengths:  **11.** bitstream = bitstream\_all[s:s+l]  **12.**  bitstream\_list.append(bitstream)  **13.** s += l  **14.** return bitstream\_list |

# Requirements

# Requirements fulfillment

The justification of the requirements fulfillment:

1. Lossy compression  
   The proposed solution supports lossy compression (see Section 2.4.2 and Section 2.6.2.2) through a combination of lossless and lossy coding phases for both geometry and attribute components.
2. Lossless geometry compression  
   Lossless geometry coding (see Section 2.4.2) is supported by the lossless coding phase, which mainly comprises AI-based probability approximation for geometric occupancy by the lossless occupancy processing units (Lossless OPU, see Section 2.4.3.2), and arithmetic coding units.
3. Lossless attribute compression  
   Lossless attribute compression is supported by lossless coding phase (see Section 2.6.2.1), which mainly comprises AI-based probability approximation for attribute residual by the lossless attribute processing units (Lossless APU) and arithmetic coding units. It is noted that the lossless attribute compression is only applicable in case of lossless geometry coding.
4. Near-lossless geometry compression  
   Adjusting encoder parameters (see Section 2.4.7) can achieve error deviation no more than required by the near-lossless threshold.
5. Near-lossless attribute compression  
   Adjusting encoder parameters (see Section 2.6.6) can achieve error deviation no more than required by the near-lossless threshold
6. Temporal variations\*

(In mechanism, *Unicorn* supports dynamic geometry coding and dynamic attribute coding. However, currently, the results of the dynamic sparse and dynamic dense categories are unavailable in the attached spreadsheet. We mark this section with \*)  
The geometry codec module of *Unicorn* supports temporal variations coding (see Section 2.4.4);   
The attribute codec module of *Unicorn* supports temporal variations coding (see Section 2.6.3).

1. Low latency\*

(In mechanism, *Unicorn* supports dynamic geometry coding and dynamic attribute coding. However, currently, the results of the dynamic sparse and dynamic dense categories are unavailable in the attached spreadsheet. We mark this section with \*)  
By adjusting the encoder parameter (see Section 2.4.7 and Section 2.6.6), the proposed solution supports latency with a granularity of a single frame.

1. Low complexity  
   The AI-based model's complexity is 589.3% compared to the anchor in static dense category. The maximum inference memory requirement is 10834 MB.
2. Temporal scalability\*

(In mechanism, *Unicorn* supports dynamic geometry coding and dynamic attribute coding. However, currently, the results of the dynamic sparse and dynamic dense categories are unavailable in the attached spreadsheet. We mark this section with \*)  
The temporal scalability is achieved by dropping specific P frames, which can be easily accomplished considering the bitstream structure described in Section 3.

1. Spatial scalability  
   The spatial scalability is achieved by early termination and adaptive transmission of the bottleneck layer. For example, spatial scalability can be achieved by partial reconstruction and early decoding termination in the lower scale rather than the highest scale, whereas the scale corresponds to the spatial resolution of the point cloud.
2. Region-based scalability  
   It is not supported in the current implementation; It is supported in principle by specifically setting encoder parameter on regions of interests. However, with the current implementation some blocking artifacts may appear on the ROI boundary.
3. Quality scalability  
   Supported in the current implementation by indicating a certain spatial resolution with subsequent processing by geometry codec that applies feature augmented up-sample block with corresponding lambda parameter or by dropping features, etc. (see Section 2.4.7); The attribute codec applies AQL block to adjust the corresponding quality of attribute using different lambda parameter, etc. (see Section 2.6.6).
4. Spatial random access  
   It is not supported in the current implementation; However, it is supported in principle by spatially partitioning the input point cloud and coding them respectively.
5. Temporal random access\*

(In mechanism, *Unicorn* supports dynamic geometry coding and dynamic attribute coding. However, currently, the results of the dynamic sparse and dynamic dense categories are unavailable in the attached spreadsheet. We mark this section with \*)  
Supported on frame level granularity for dynamic sparse and dynamic dense content.

1. Error resilience  
   Error concealment is supported on a frame level by dropping specific bitstreams (P frames), which can be easily accomplished considering the bitstream structure described in Section 3.2.
2. Parallel encoding and decoding  
   The various scales can be compressed independently by the encoder, however the decoder is a sequential process.
3. Separable attribute and geometry coding  
   Multiple elementary bitstreams represent geometry and attributes and can be containerized in v3c elementary bitstream.
   1. Geometry only coding  
      Supported
   2. Multiple attribute coding  
      Supported by associating multiple attributes with a single geometry value
4. Geometry precision  
   The supported geometry coding precision range is up to 22 bits. The range may be extended by applying spatial subdivision to avoid memory overflow caused by the excessive number of points. There is no bit depth limit technology wise.
5. Model architecture  
   The model description for geometry coding can be found in Section 2.4;  
   The model description for attribute transfer can be found in Section 2.5;  
   The model description for attribute coding can be found in Section 2.6;  
   The architecture of geometry and attribute models are stable.
6. Model update
   1. On the fly Model Update  
      Not supported. The model parameters are unified across data types and rate points. The model update can improve the compression performance according to the application seniors, but it is not necessary for such a model update, not to mention a sequence-specific update.
   2. On-demand Model Update & download  
      Model parameters update can be updated on demand by distributing updated model parameters. The decoder can download the model demanded prior to operation.
7. Inference Reproducibility  
   A proponent confirmed the inference reproducibility on several platforms and that it is available for x-check upon request.

# Coding results subjective quality

This section provides examples of subjective quality for the point clouds decoded by *Unicorn*. The point clouds are rendered by CloudCompare software, and R1-R4 corresponds to the quality from low to high.







Figure Subjective quality (example)

# Coding results objective quality

The objective results can be found in the attached spreadsheet file.

# Reference

1. **WG2, MPEG Technical requirements.** Requirements for AI-based Point Cloud Coding. *ISO/IEC JTC 1/SC 29/WG2.* 2024.

2. **Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, Zhan Ma.** Sparse Tensor-based Multiscale Representation for Point Cloud Geometry Compression. *IEEE Trans. Pattern Analysis and Machine Intelligence.* July 2023, pp. 9055-9071.

3. **Sullivan, G.J. and Wiegand, T.** Rate-distortion optimization for video compression. *IEEE signal processing magazine.* 1998, pp. 74--90.

4. **Hausdorff, Felix.** Dimension und äußerem Maß. *Mathematische Annalen.* 1918, Vol. 79.

5. **WG07.** G-PCC codec description v12. *ISO/IEC JTC1/SC29/WG7 N00271.* January 2022.

6. **Chen, Tong and Ma, Zhan.** Variable Bitrate Image Compression with Quality Scaling Factors. *IEEE ICASSP.* 2020, pp. 2163-2167.

7. **Zhihao Duan, Ming Lu, Jack Ma, Yuning Huang, Zhan Ma, Fengqing Zhu.** QARV: Quantization-Aware ResNet VAE for Lossy Image Compression. *IEEE transactions on pattern analysis and machine intelligence.* 2023.

8. **Thomas M. Cover, Joy A. Thomas.** *Elements of Information Theory.* 2006.