An overview of MPEG Geometry-based Point Cloud Compression Standard

Wei Zhang, Wen Gao, Xiang Zhang, Yiting Shao, Sebástien Lasserre, Ohji Nakagami, David Flynn and Marius Preda

***Abstract*—Point cloud is a major data representation of 3D objects and scenes. Along with its superiority in describing the real-world in an immersive and interactive fashion, its huge amount of data volume also challenges the capacity of current multimedia ecosystems. With an increasing number of point cloud applications emerging in the market in recent years, the Moving Picture Expert Group (MPEG), a prolific international standardization body in multimedia coding, formally launched the point cloud compression project in 2017. The project progressed to two branches with different underlying philosophies in the past three years, namely the Video-based Point Cloud Compression (V-PCC) and Geometry-based Point Cloud Compression (G-PCC). This paper aims at delivering the main technical aspects of standardization activities in G-PCC which will be soon finalized and published in 2021.**

***Index Terms*—Point cloud compression, geometry coding, attribute coding, MPEG, G-PCC.**

I. Introduction

RECENT years have witnessed dramatic advances in 3D capturing and reconstruction technologies. Point cloud, a major 3D representation format, soon become popular in various applications including autonomous driving, robotics, geographic information system, biomedical, 3D immersive interaction and digital cultural heritage [1]. To describe a 3D object or a real-world scene, a point cloud usually comes with a large set of points in 3D space with geometric information and attribute information. The geometric information represents 3D coordinates of each point in the point cloud; the attribute information describes the characteristics (e.g., color and reflectance) of each point. Unfortunately, such representation format requires a large amount of data, bringing huge challenges to data storage and transmission. Point cloud compression therefore become a heated research area in both academia and industry.

To come up with a practical solution of point cloud compression for various use cases, MPEG announced the Call for Proposal on point cloud compression in April 2017 [2]. This project targets on the compression of various types of point clouds including static point clouds (noted as Category 1 data), dynamic point clouds (noted as Category 2 data) and dynamically acquired point clouds (noted as Category 3 data). The Category 3 data can be further classified into the frame-by-frame type (Cat3-frame) and the multiframe-fused type (Cat3-fused). According to the technical contributions and collaborations from industry and academia, the standardization activities progressed to two distinct technical solutions: V-PCC (ISO/IEC 23090 Part 5) for Category 2; and G-PCC (ISO/IEC 23090 Part 9) for Category 1 and Category 3. Figure 1 shows several samples from the MPEG PCC dataset including human models, buildings, objects, landscapes and LiDAR acquired data.

There are notable differences between the two solutions standardized by MPEG. Inspired by the displacement mapping [3] and texture mapping [4] technologies in computer graphics, V-PCC follows a projection-based coding principle. The point coordinate is encoded as the distance with respect to a particular plane. The attribute associated to a 3D point is projected to a 2D map. 2D video codecs can be used to code both the geometry and attribute information.

Different from V-PCC, G-PCC is a 3D oriented compression scheme in which the point cloud in 3D space is directly encoded. Data structures like octree and kd-tree are used to represent the point cloud. G-PCC encodes the geometry information and attribute information of point clouds separately. Geometry is coded first since the attribute coding depends on the reconstructed geometry. The codec architecture is illustrated in Fig. 2. This diagram may not represent the whole set of tools integrated in the codec but covers the key modules currently in the standard specification. In this paper, core functionalities in both geometry coding and attribute coding are introduced.

The G-PCC standard is now in the Final Draft International Standard (FDIS) status and is expected to be published by ISO/IEC in 2021. Accompanying this international standard, a reference software [5] named Test Model Category13 (i.e., TMC13, ISO/IEC 23090-21) including both the encoder and decoder functionalities is also provided.

This paper is organized as follows. Section II introduces the geometry coding design in the latest G-PCC specification draft. The attribute coding methods are described in Section III. Section IV highlights major functionalities supported by the G-PCC scheme. Section V discusses the profile design under consideration and potential future plans for G-PCC.

II. Geometry Coding Design

The geometry of a point cloud is often expressed in application-specific spaces where point positions may be represented by floating point numbers. Therefore, a pre-processing step to convert the application-specific space to

Wei Zhang is with the School of Telecommunications Engineering, Xidian University. Xi'an, China email: wzhang@xidian.edu.cn
Wen Gao and Xiang Zhang are with Tencent American.
Yiting Shao is with PKUsz.
Sebástien Lasserre is with Xiaomi.
Ohji Nakagami is with SNOY Corporation, Tokyo, Japan.
David Flynn is with Apple Inc.
Marius Preda is with Institut MINES-Télécom.
Manuscript submitted January 4, 2021.

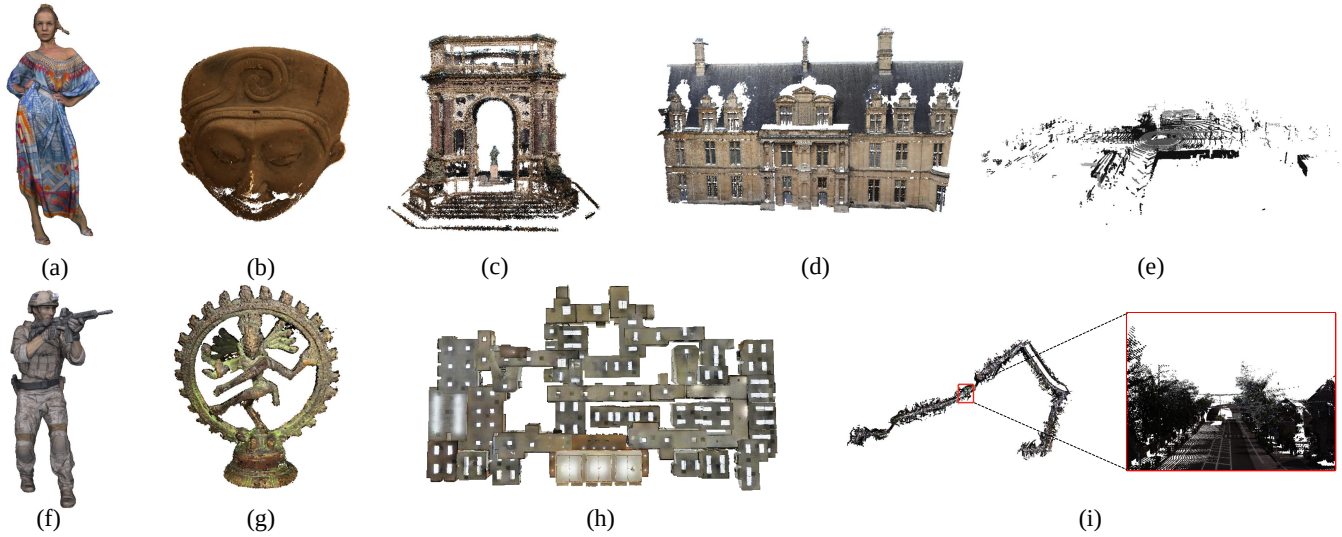

Fig. 1. Samples from the MPEG PCC dataset. (a) Longdress, (b) Egyptian mask, (c) Arco_Valentino_Dense, (d) Facade, (e) Ford, (f) Soldier, (g) Shiva, (h) Standford and (i) Citytunnel.

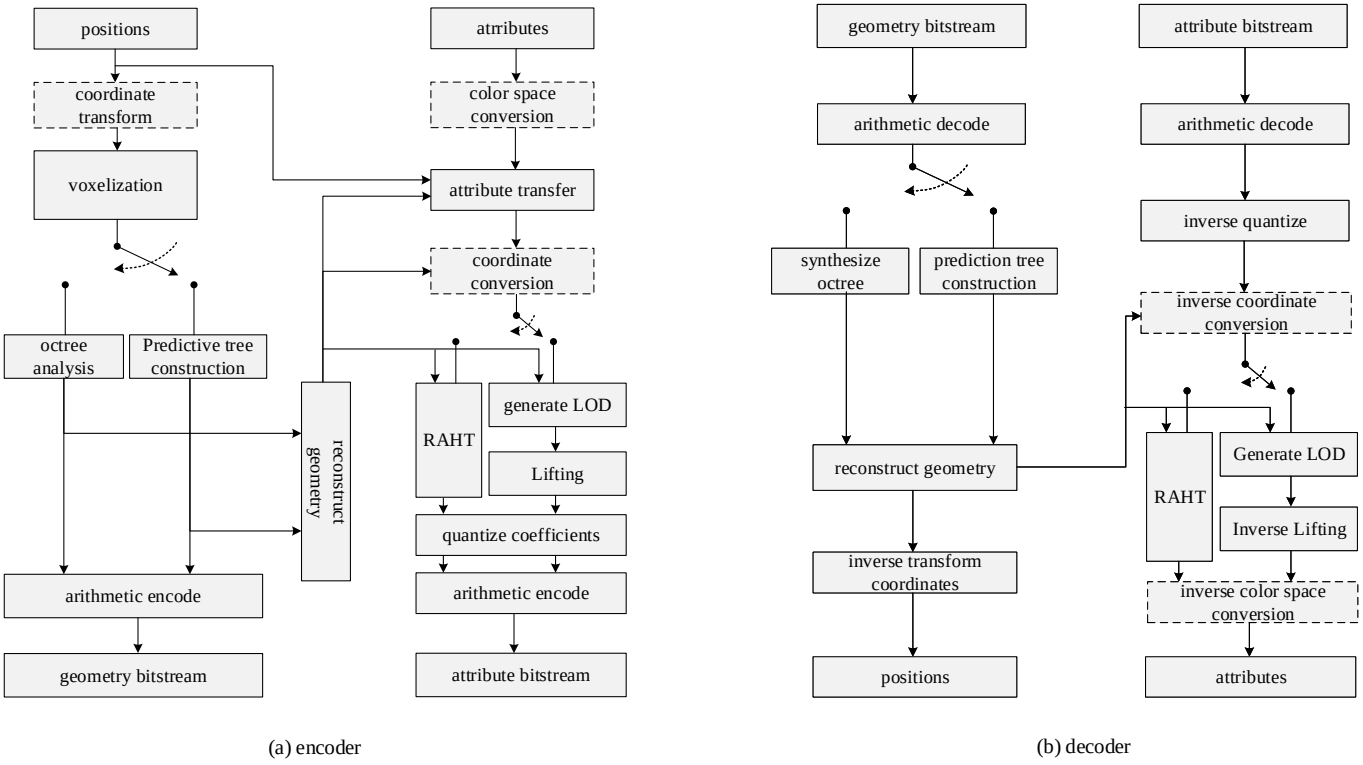

Fig. 2. G-PCC codec architecture. Modules in the dotted boxes are optional.

the finite-resolution internal space may be required. Point positions are represented internally as non-negative d-bit integers before being compressed. To obtain these integers, point positions in the internal coordinate system are rounded. After this process, there may be multiple points with the same position, called duplicate points. The duplicate points removal process is optional. If enabled, multiple points with the same quantized position and different attributes will be merged in a single point. The attributes associated with the single point will be computed by the attribute transfer module described in attribute coding section. The process of position quantization, duplicate point removal, and assignment of attributes to the remaining points is called voxelization. In other words, voxelization is the process of grouping points together into voxels.

A. Octree-based geometry coding

Octree geometry partition has been acknowledged as an efficient representation of 3D geometry space [6]. It was used

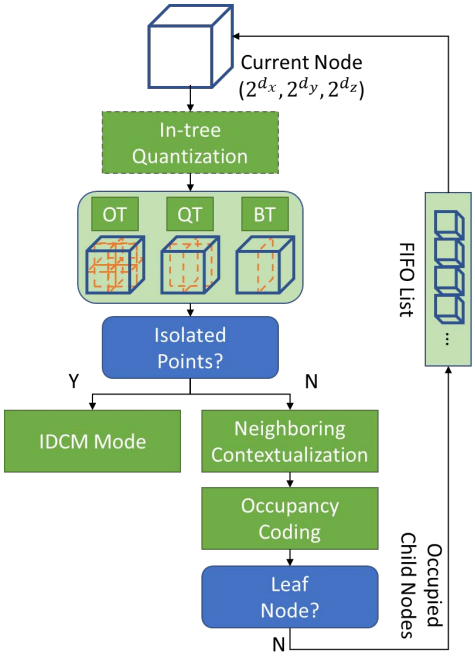

Fig. 3. Octree-based geometry coding framework.

for the purpose of point cloud compression back in 2000 [7]. In a nutshell, the octree representation iteratively partitions the 3D space into eight sub-spaces until reaching the leaf nodes where the voxel size is $1 \times 1 \times 1$.

In G-PCC standard, the framework of the octree-based geometry coding can be illustrated in Fig. 3. The basic pipeline follows the key idea in octree partitioning. First of all, all the points in 3D space can be wrapped in a cuboid bounding box with the size of $(2^{d_x}, 2^{d_y}, 2^{d_z})$, which corresponds to the root node of the tree structure. Then the root node starts splitting into N child sub-nodes, where $N = 8, 4, 2$ for octree (OT) quad-tree (QT) and binary-tree (BT) partitioning, respectively. The same partitioning process applied from the root node in depth 0 to the leaf nodes in depth d in the breadth-first search (BFS) order. The nodes in the same depth are partitioned via the same tree structures (either OT, QT or BT), and only the occupied child nodes are further partitioned into the next depth. Note that the tree node information is maintained in a first-in-first-out (FIFO) list, where the first element is the current node to be coded and the child nodes are pushed back to the end of the list, by which the tree nodes are traversed in the BFS order.

To code the tree structure in a bitstream, an N-bin occupancy code associated with a tree node, in which each bin represents whether a child-node is occupied or not, is coded for every tree node in the FIFO. The occupancy code is coded by the context-based arithmetic entropy coding where the neighboring coded nodes are used as context information to improve the coding efficiency of current coded node. Various techniques are applied to reflect efficient context models from different aspects such as neighbor-Dependent Entropy Context (NEIGHB) [8], Intra prediction [9], Planar coding mode [10], etc. On the other hand, Inferred Direct

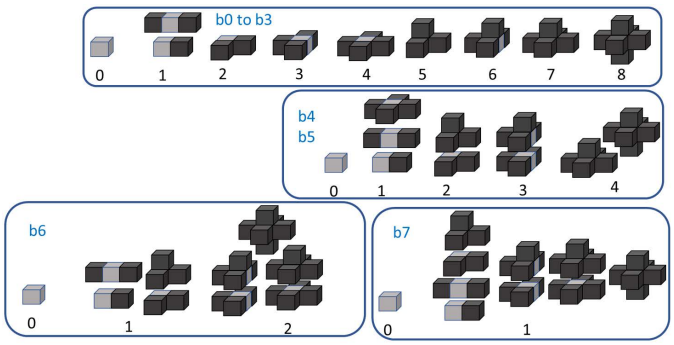

Fig. 4. Neighbor configurations (NC) in terms of each coded bit.

Coding Mode (IDCM) [11] is used to code the isolated points, in which their relative coordinates are simply by-pass coded. Isolated points are those away from all other points to correlate with. By IDCM, isolated points can be coded more efficiently because no further tree partitioning is performed on them. Details of some major coding tools in the octree-based geometry coding framework are elaborated as follows.

1) Adaptive geometry partition: BT and QT partitions can be viewed as generalized octree structures where only one or two axes out of three are divided. They are introduced as alternative partition modes of OT, allowing flexible geometry representations and better coding performance. However, searching the optimal partition mode among all possible OT/QT/BT combinations is impractical since the search range is enormous. A good trade-off between the flexibility and complexity is achieved by introducing the constraint that the nodes in the same partition depth should take the same partition mode. A 3-bit syntax is used to indicate the partition mode at each partition depth, where each bin represents whether x, y and z axis is divided, respectively. Furthermore, an implicit geometry partition scheme [12] can be further applied, where the partition modes can be simply determined by two hyper-parameters.

2) Neighbor-Dependent Entropy Context (NEIGHB): Neighbor-Dependent Entropy Context (NEIGHB) defines the configuration in context-based arithmetic coding in terms of the six coded neighbors of parent node and coded bins of current node. Let $b_j(j = 0, 1, ..., 7)$ denote the 8 bins of the occupancy code. The maximum number of neighbor configurations (NC) of the parent node is $2^6 = 64$, considering all combinations of six neighbors of the parent node. When coding b_i, the context is computed based on the NC and the previous coded bits, i.e., $b_j(j = 0, 1, ..., i-1)$, leading to a big number of total contexts. Techniques target on reducing the number of NC is thus considered in G-PCC. One method is to reduce the number of contexts according to the bit to be coded [13]. As shown in Fig. 4, the NC is classified into different numbers of groups for each b_i. For instance, 9 NCs are used for bits b_0 to b_3, 5 NCs are used for bits b_4 and b_5, 3 NCs are used for b_6 and only 2 NCs are used for b_7. This leads to a significant decrease of the number of contexts. Other context reduction methods are included such as by defining the falsely occupied neighbors [14] and improvements on advanced neighbors [15].

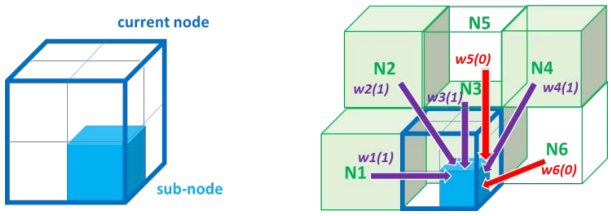

Fig. 5. Weights between neighbors and sub-nodes of current node.

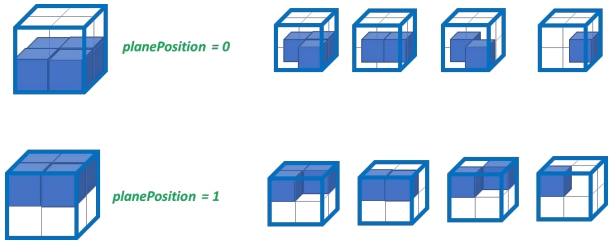

Fig. 6. Illustration of the Planar coding mode.

3) Intra Prediction: Intra prediction [9] uses 26 neighbors that share a face, an edge or a vertex with the current node to predict the occupancy code of current node. An occupancy score, i.e., $score_m$, is computed for each of the eight sub-nodes of the current node by using a weighted sum over the 26 neighbors as follows,

$$score_m = \frac{1}{26}\sum_{1}^{26} w_{k,m}(\delta_k)$$

where m is the sub-node index, k is a neighbor index, $w_{k,m}$ is the contribution (weight) from neighbor k to sub-node m, and δ_k is the occupancy status (0 for non-occupied, 1 for occupied) of the neighbor k, as depicted on Fig. 5. The $score_m$ is then transformed into a ternary information $pred_m$ belonging to the set "predicted non-occupied", "predicted occupied", "not predicted" associated with three prediction contexts by using two thresholds. If the $score_m$ is lower than the first threshold, then $pred_m$ is set to "predicted non-occupied"; if $score_m$ is higher than the second threshold, then $pred_m$ is set to "predicted occupied"; otherwise the score is between the two thresholds and $pred_m$ is set to "not predicted".

4) Planar, Angular and Azimuthal coding mode: Planar coding mode [10] is efficient in representing planar structures in point clouds. As shown in Fig. 6, the occupied sub-nodes in current node are distributed in the same horizontal plane along the vertical Z-axis, from which the planar coding mode can benefit. Specifically, a flag is first introduced to indicate whether the occupied child nodes belong to a same horizontal plane. If the flag is true, then another bit is signaled to indicate whether the plane is the lower plane or the upper plane. The similar technique applies to the plane structures along the X-axis and Y-axis as well. To further improve the planar mode for the laser scanned point clouds, angular [] and azimuthal [16] coding modes [17] are proposed, where the laser angles are utilized as prior knowledge to predict the plane position of the planar mode, by which the bits can be further saved.

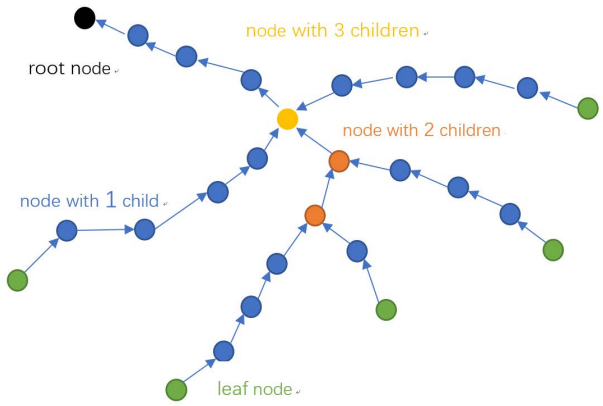

Fig. 7. An example of a prediction tree.

5) Inferred Direct Coding Mode (IDCM): IDCM [11] is proposed to deal with isolated points, which are defined as points with less correlations than others. The relative coordinates of all isolated points in the current node are coded directly without going through the context based entropy coding process. The further geometry partitions can be omitted as well. In addition, IDCM provides multiple intensity levels for various trade-offs between complexity and coding efficiency. It is able to provide much faster encoder and decoder with negligible coding performance loss.

6) Adaptive geometry quantization: Different quantization steps (QS) can be applied to different slices and the entire point cloud as configured in high-level syntax [18], [19]. To achieve finer granularity of QS control, the octree-based adaptive geometry quantization [20] is introduced in octree coding process, where the octree nodes at a specified partition depth can apply different quantization steps. It provides functionalities such as the rate control and region of interests, where different nodes at different regions can apply various QS according to the quality requirement.

B. Predictive geometry coding

As discussed in previous section, the octree-based geometry coding scheme provides good compression performance for dense point clouds but exhibits limited performance for sparse point clouds acquired by LiDAR devices. In October 2019, a different geometry coding scheme, i.e., a prediction tree-based geometry coding scheme, was proposed [21], targeting low latency applications or applications requiring low decoding complexity. Note that a basic version of prediction tree-based compression scheme was introduced in 2005 [22].

1) Principles of predictive geometry coding: The principles of the scheme is summarized as following: A tree is constructed in which a point in a point cloud is associated with a node in the tree; a node in the tree can only be predicted from its ancestors. Hence such kind of tree is called a prediction tree, which is shown in Fig. 7, in which the root node is denoted as a black solid circle where other colors indicates the number of children nodes for a given node. For example, a yellow node has three children, a blue node has only one child, etc. In addition, the arrow starting from a given node in Fig. 7 is used to point to its parent node.

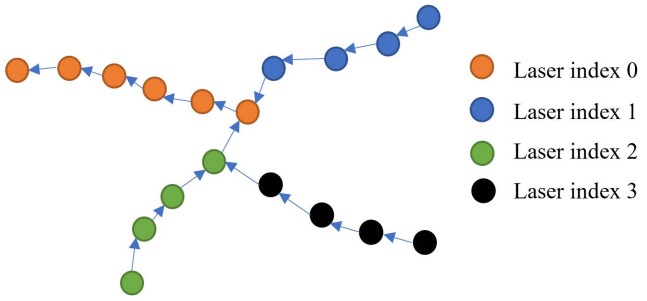

Fig. 8. An example of a prediction tree based on laser index.

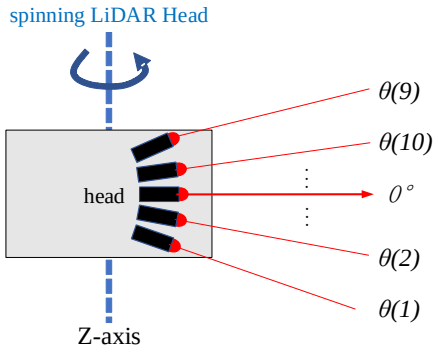

Fig. 9. An illustration of spinning LiDAR acquisition model

Once the prediction tree structure is constructed, various prediction schemes can be applied. In MPEG G-PCC standard specifications, the following four prediction modes are included:

- PCM: prediction from a default position
- DPCM: delta prediction, i.e., $p0$
- Linear-2: linear prediction, i.e., $2p0 - p1$
- Linear-3: parallelogram prediction, i.e., $p0 + p1 - p2$

where $p0$, $p1$, $p2$ are the positions of the parent, grandparent and grand-grandparent of the current node. The prediction tree structure is encoded by traversing the tree in a depth-first search (DFS) order and encoding child count, prediction mode and prediction residuals for each node in the tree. Note the prediction residuals of a node are computed by subtracting the prediction position from the node geometry position. If duplicate points, i.e., points with the same geometry positions, are allowed, additional syntax elements indicating the number of duplicated points are carried in the bitstream for each node of the tree. In addition, different QS can be applied to different nodes during the DFS order-based encoding process, thus achieving finer granularity of QS control.

2) Encoder algorithms: Note that constructing an optimal prediction tree itself is an NP-hard problem. Variety of simplified algorithms can be developed to trade-off different aspects of the prediction tree-based geometry coding method, such as compression performance, computational complexity, memory consumption, latency requirement, etc. During the development of predictive geometry coding, following encoder algorithms have been studied [21]:

- High latency use case slow mode
- High latency use case fast mode
- Low latency use case

In the high latency use case, all points in a point cloud are re-ordered based on their Morton codes: (1) In the slow mode, a kd-tree is used to keep track of all the possible prediction positions. To find the parent node of a given point (node), several nearest prediction positions to the point are found by searching the K-D tree; One position is chosen based on certain criterion and its associated node in the prediction tree is used as the parent node of the given node; (2) In the fast mode, the K-D tree is not used due to its complexity. Instead, k-D tree search is replaced by an approximate nearest neighbor search with a pre-defined search range according to the Morton order. In the low latency use case, no re-ordering process is applied and the encoder processes points in their receiving order. In addition, a buffer of limited size are employed to control the latency and the nearest neighbor search is done in the buffer.

3) Angular mode: One benefit of the predictive geometry coding is its low decoding complexity and the ability to achieve low latency. On the other hand, it is also observed that the encoder runtime of predictive geometry coding is much higher than that of the octree-based coding scheme. To address the above issue, an improved scheme based on a spinning LiDAR acquisition model was proposed in [23] to further improve the compression efficiency and reduce computational complexity. This mode is called angular mode which is only used to Cat3-frame type of point clouds.

First, the spinning Lidar model is described as N lasers, for example, $N = 16, 32, 64$, spinning around the Z-axis at a relatively constant angular speed; each laser has different elevation angles $\theta(i), i = 1, ..., N$ and height $\varsigma(i), i = 1, ..., N$. Suppose that the laser i hits a point M, with Cartesian integer coordinates (x, y, z). An example of a spinning LiDAR is shown in Fig. 9.

The Cartesian coordinates (x, y, z) is then converted to three parameters (r, ϕ, i):

$$r = \sqrt{x^2 + y^2}$$

$$\phi = \text{atan2}(y, x)$$

$$i = \arg \min_{j=1,...N} \{z + \varsigma(j) - r \times \tan(\theta(j))\}$$

where i denotes the laser index. In G-PCC, the quantized version of (r, ϕ, i) denoted as $(\tilde{r}, \tilde{\phi}, i)$ are used.

After converting points in a point cloud to the $(\tilde{r}, \tilde{\phi}, i)$ representation, the same approach as described for general predictive geometry coding can be applied. In addition, a new predictor leveraging the characteristics of LiDAR scanning could be introduced. For instance, the rotation speed of the LiDAR scanner around the Z-axis is usually constant. Thus, the current $\tilde{\phi}(j)$ can be predicated as:

$$\tilde{\phi}(j) = \tilde{\phi}(j-1) + n(j) \times \delta_{\phi}(k)$$

where $\delta_{\phi}(k), k = 1...K$ is a set of potential speeds which the encoder could choose from and $n(j)$ is the number of skipped points.

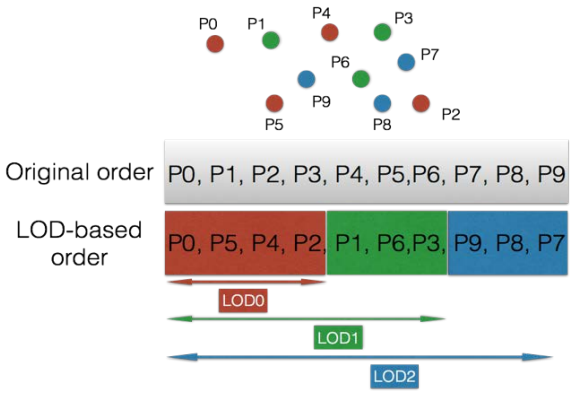

Fig. 10. An illustration of the distance-based LoD construction procedure.

4) Low complexity encoder: After the coordinate conversion, a low complexity prediction tree generation method can be employed. Specifically, each node chooses the latest node associated with the same laser index i as its parent node. If such a parent node is unavailable, the current node chooses a node with a different laser index j as its parent node such that the prediction residuals are minimum. An example of a prediction tree constructed using the above method is provided in Fig.8. With the angular mode and the simple approach to construct the prediction tree using laser index, the predictive geometry coding is now able to achieve comparable compression performance for Cat3-frame point clouds compared to octree-based coding under common test conditions (CTC) but with significantly less encoding and decoding time.

III. Attribute Coding Design

There are three major types of attribute coding methods in G-PCC: interpolation-based hierarchical nearest-neighbor prediction (Predicting transform) [24], interpolation-based hierarchical nearest-neighbor prediction with an update/lifting step (Lifting transform) [25] and Region Adaptive Hierarchical Transform (RAHT) [26], [27]. While RAHT and Lifting transform are typically used for Category 1 data, predicting transform is typically used for Category 3 data.

A. Attribute transfer

Attribute transfer [28] is an encoder-only technique when geometry compression procedure (e.g., quantization) changes the point coordinates from the original. In such case, the reconstructed positions do not have any attribute attached. Given the original point cloud positions and attributes as well as the reconstructed positions, the objective of attribute transfer procedure is to recolor the reconstructed points that minimize the attribute distortions. In TMC13, the attribute transfer procedure is implemented as the calculation of distance-weighted averages from multiple nearest neighbors in the original point cloud. Prior to the attribute transfer process, TMC13 supports a conversion from RGB color space to other color space (e.g., YCbCr and YCoCg) and back again if desired.

B. Coordinate conversion

Coordinate conversion [29] is an optional process performed prior to the attribute coding especially for the LiDAR acquired point clouds (i.e., Category 3 frame sequences). The Euclidean distance of two points sampled from adjacent lasers is increasing dramatically with with the increase of the radial distance, leading to a non-uniform sampling manner and thus the inefficiency in attribute coding. To correct the non-uniform sample volume, point positions in Cartesian coordinates are transformed to spherical coordinates. It should be noted that when angular coding mode is activated, this conversion should be disable.

C. Predicting transform

The predicting transform implements a nearest neighbor prediction method relying on a multi-layer structure called Level-of-details (LoDs). To construct efficient LoD representations for individual point cloud, the distance-based LoD construction method [30] is applied for Category 1 and Category 3-fused data while the decimation-based LoD construction method [31] is considered for Category 3-frame data.

For attributes signals of relatively uniformly distributed points, the distance-based LoD generation process is performed. A Morton order sorting is performed before the LoD construction to keep the correlation between neighboring points. To achieve the low delay attribute coding, a flag is introduced to skip of the sorting process when the number of LoD equals to 1. Figure 10 gives an example of the distance-based LoD generation process. This process reorganizes the points into a set of refinement layers according to a deterministic distance criterion. More specifically, $LoD0$ is generated by screening points at a wide distance threshold and therefore $P0$, $P4$, $P5$ and $P2$ having a sufficient distance in between are selected. Then, $LoD1$ is formed at a slightly smaller distance interval with $P1$, $P6$ and $P3$ included as the refinement layer. Finally, $LoD2$ is formed with the narrowest interval. Once the LoD representation is built, the attribute values of each point are predicted as the linear interpolation of the reconstructed values of the nearest neighbors in the LoDs of the same level or one level above. The maximum number of the reference neighbors is a parameter calculated by the encoder which is arithmetically encoded in the bitstream. More precisely, the attributes of $P2$ can be predicted by the reconstructed value of $P4$, $P5$ and $P0$, or by a weighted summation of them. The weight of each reference point is determined according to the distance from the reference to the current point. If more than one point can be referenced, a rate distortion optimization-based selection is performed to determine whether using a single reference or the weighted average of selected neighbors. Finally, the attribute prediction residual is quantized and coded by the arithmetic coder. To get different tradeoffs between coding efficiency and parallel processing, an intra LoD prediction flag is introduced to control the reference structure for predicting transform. If the intra LoD prediction is enabled, points in the same LoD could be referred. It is noted that intra LoD prediction is always used when the number of LoDs is equal to one.

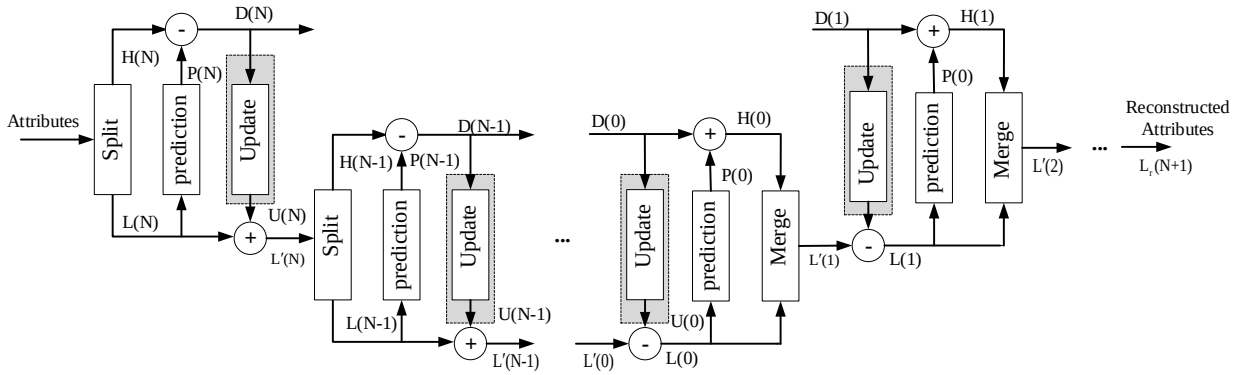

Fig. 11. The LoD-based predicting/lifting transform scheme. The update operators highlighted in gray are only for lifting transform.

For non-smooth attribute signals of irregularly distributed point clouds (e.g., LiDAR acquired data), decimation-based LoD generation method is applied. Instead of using distance thresholds, it uses a set of sampling rates denoted as $(k_l)_{l=0...L-1}$, where k_l is an integer describing the sampling rate for the $LOD(l)$ (e.g., $k = 4$). The sampling rate k_l could be further updated within an LOD to adapt to the local point cloud distribution, thus leading to more efficient predictions. It is noted that different subsampling rates may be defined per attributes (e.g., color, reflectance) and per channel (e.g., Y and U/V).

D. Lifting transform

From the implementation point of view, the predicting transform detailed in previous subsection can be described as a series of splitting and merge process at the encoder side and the decoder side, respectively. As shown in Fig. 11, based on the LoD construction method, the split operator takes the set of attributes associated with the point cloud as input and generates two outputs: (1) $L(N)$ which is the corresponding attributes associated with $LoD(N)$, and (2) $H(N)$ which is the set of attributes associated with the refinement layer $R(N)$. The prediction operator predicts the attribute value of points in $R(N)$ by using the attribute values of its k-nearest neighbors in $L(N)$. The reversed process is performed by the merge operator at the decoder side.

The lifting transform was built on top of the predicting transform with an update process and an adaptive quantization strategy. Since the LoD-based prediction strategy makes points in lower LoDs more influential, an influence weight for each point is defined and recursively updated based on the prediction residual $D(N)$, the distances between the predicted point, and its neighbors. The influence weights computed during the transform process are further used to adaptively quantize the prediction residual. More specifically, the prediction residual of a point in $H(N)$ is scaled by the squared root of its influence weight before quantization. An inverse scaling process by the same factor is applied after inverse quantization at the decoder side. It should be noted that the decoder can determine the scaling factors by the reconstructed geometry without explicit signaling in the bitstream.

E. RAHT

The transform of RAHT is a hierarchical sub-band transform that resembles an adaptive variation of a Haar wavelet. Based on a hierarchical tree structure, attributes of occupied voxels in the same parent node are recursively transformed along each dimension in a bottom-up style, where all low-pass (DC) coefficients at one level of hierarchically organized data would be passed to the transform process of next level, while all high-pass (AC) coefficients are coded by arithmetic codec.

Regarding to the hierarchical transform process, in general, RAHT combines the attributes of occupied voxels in a given level l of the hierarchy, passing the combined DC coefficients to predict the attributes of the next level $l-1$ of the hierarchy. In the combination process, RAHT also produces AC coefficients that can be used to reconstruct the attributes of occupied points in the given level l. At the next level $l-1$ of the hierarchy, RAHT repeats the process of combining attributes of occupied points, passing combined attributes to level $l-2$, and producing AC coefficients for the attribute reconstruction of occupied points at the level $l-1$. In this way, attributes of all occupied voxels are transformed from level l to level 1,merging the smallest voxels into successively larger voxels until reaching the root for the entire 3D space.

The region-adaptive characteristic of RAHT is reflected in that transform weights of each iteration are adaptively determined by the count of occupied points in a region. If one of the voxels in the transformed pair is unoccupied, the other occupied voxel is promoted to the next transformation step without any processing.

An example of the RAHT and inverse RAHT for level $l = 3$ is presented in Fig. 12, where $w_{i,j}$ is the transform weight for the node $n_(i,j)$ with the level index i and the node index j and $g_(i,j)$ is the attribute signal of the node located at the leaves of the tree. The DC coefficient $g'_{i,j}$ and the AC coefficient $h_(i,j)$ are obtained and located at the intermediate nodes of the tree when merging two occupied nodes. If one of the nodes to be transformed is unoccupied, original attribute of current node would be directly passed to next level and no AC coefficient would be generated. When it reaches to the tree depth, DC and AC coefficients of the root node and AC coefficients of intermediate nodes would be quantized and encoded into the

TABLE I
MAIN TOOLS IN G-PCC AND THEIR TARGET POINT CLOUD CATEGORIES.

Coding tool	Solid	Dense	Sparse	Scant	Lidar-Fused	Lidar-Frame
Adaptive geometry partition			1	1	1	1
Neighbor Dependent Entropy Context	1	1				
Intra Prediction	1	1				
IDCM			1	1	1	1
Planar			1	1		
Angular						1
Predictive tree					1	1
LoD scheme	1	1	1	1		
RAHT			1	1	1	1

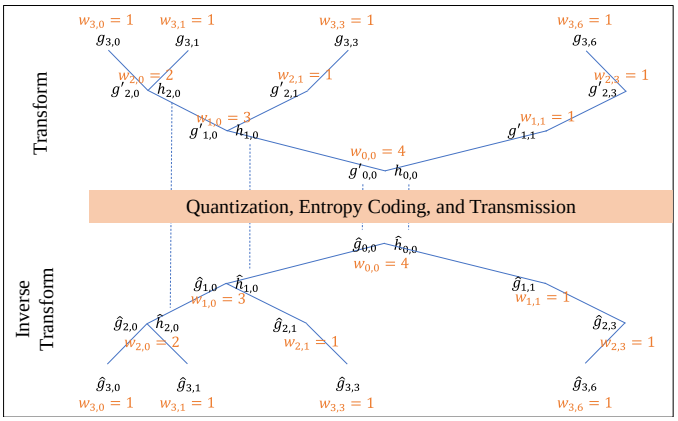

Fig. 12. An example of the RAHT and inverse RAHT for a 3-level hierarchy.

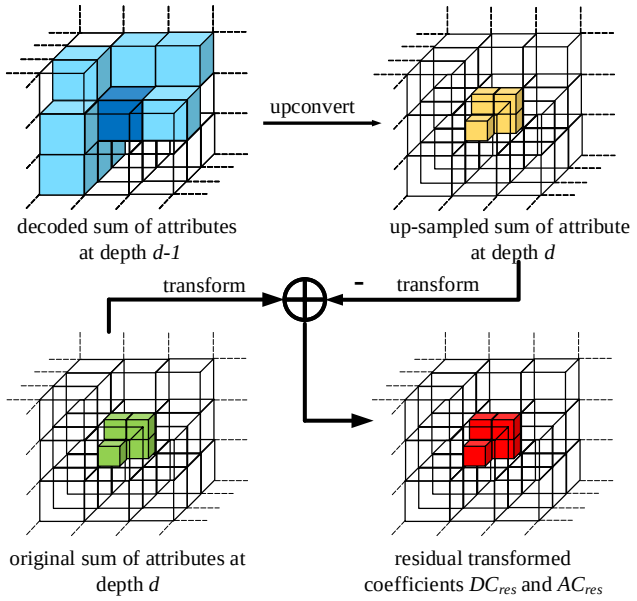

Fig. 13. Illustration of the upsampled RAHT.

bitstream.

To further improve the coding efficiency on RAHT, a transform domain prediction is introduced [32]. To do so, the RAHT tree traversal approach is changed from bottom-up to top-down. A tree of attribute and weighted sums is constructed and then RAHT is performed from the root of the tree to the leaf nodes. The transform is also performed in units of $2\times2\times2$ blocks. Within the block, the encoder transform order is from leaves to the root. The inverse transformed coefficients of one $2 \times 2 \times 2$ block become the inherited DC coefficients of the next level. For each $2 \times 2 \times 2$ block, a predicted block is produced by upconverting the previous transform level. The prediction is transformed and subtracted from the transformed attributes at the encoder, as shown in Fig. 13.

F. Attribute residual encoding

For point cloud attributes such as color, an inter-channel residual prediction [33] can be applied to perform prediction of residuals between channels and code the residual resulting from this prediction. In this regard, the correlation among color channels is further considered to improve the attribute coding efficiency. The quantized, transformed coefficients or residuals are entropy coded using run length coding and an arithmetic coder [34], [35]. It is noted that the above-mentioned attribute coder can only handle integer attributes with one or three components with limited bit depths. In case that attributes are desired to be lossless coded in a floating-point format or exceed the codec's bit-depth requirements, a raw attribute coder [36] is introduced to code attributes as a sequence of fixed or variable-width values.

In general, the G-PCC standard is designed to handle various content categories for different use cases using the tools introduced above. Table. I summarizes the major tools presented in this paper and the main content categories they apply to. In this table, point clouds are categorized according to their density and sampling.

IV. G-PCC FUNCTIONALITIES

The G-PCC standard is designed to meet multiply requirements and to support various functionalities. This section briefs the key functionalities supported by the coding architecture.

A. slice and tile

In G-PCC, a single point cloud frame can be divided into multiple data units as shown in Fig. 14 to allow parallel

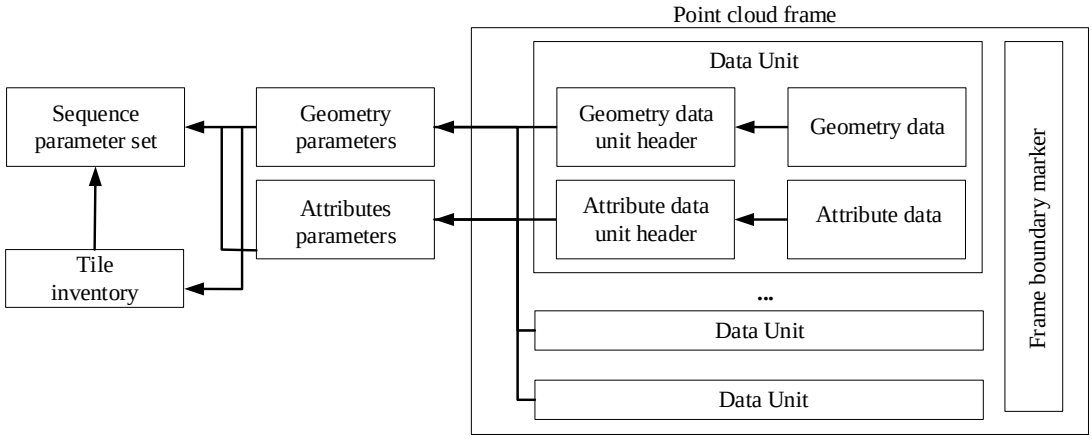

Fig. 14. G-PCC bitstream structure.

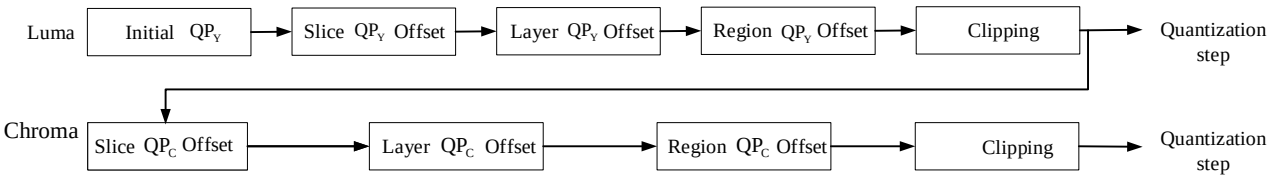

Fig. 15. The attribute quantization step derivation for luma and chroma.

processing within a single frame. A slice is a set of points comprising one geometry data unit and zero or more attribute data units [37]. A group of slices may be identified by a common tile identifier [38]. A tile inventory contains metadata that defines zero or more spatial regions. Each spatial region is identified by either an implicit or explicit tile id. Each slice that belongs to the same tile may be allowed to have dependencies while the slices with different indices of the tile should be independent.

To perform the slice and tile partition, the original point cloud would be firstly quantized (preprocessed) in terms of use cases. Then the quantized cloud is split into tiles, which are several cubes with certain side lengths. The bounding box of tile should be signaled in high-level syntax. After that, each tile is split into slices only when the number of points of a tile is larger than a threshold that indicates the maximum number of points in a slice. An encoder is free to choose how to perform slice partition. In any case, by encoding each region with data units and appending tile metadata, the decoder can access at specific locations by the corresponding bitstream. Such configuration flexibility is available depending on the use case.

From one slice to another, there should be a process to re-initialize the state of the entropy coding engine to default value at the end of each slice for parallelism. The initialization of entropy context table will cause a decrease in encoding performance. However, if slice is implemented for data segmentation purposes in low latency application and not for parallelism, an entropy context continuation flag [39] is introduced to save the context probability table at the end of slice and set it back at the beginning of the next slice for the sustainable use of the same context probability. This method will be able to improve the coding performance for slices.

B. Quality control

In G-PCC, the geometry quality and attribute quality can be adjusted independently.

For geometry, the quality adjustment can be performed by the non-normative pre-processing as voxelization. The voxelized input can be further quantized with a quality parameter control. For octree case, the tool is called in-tree quantization [40] that adjusts the voxel resolution for each spatial region. It cuts leaf nodes from specific depth with controlled granularity. For predicting geometry case, the position quality is adjusted by controlling the accuracy of the residual signal. These adjustments can be made independently for each data unit.

As for attribute, except for the slice-based quality control enabled by the slice and tile functionality, there are two alternatives to control the quality of the reconstructed signals: the layer-based quantization control [41] and the region-wise quantization control [42]. The attribute quantization parameter of each refinement layer in the LoD scheme and the octree layer in the RAHT scheme can be adjusted by using the layer-based quantization control. More precisely, a layer QP offset parameter is implemented on each attribute slice header. The effective QP value for each layer in a particular slice is obtained by adding the layer QP offset of that layer with the QP of that slice. In addition, the quantization parameter can be further adjusted in a region-wise manner based on the geometry position in slice, as the visual importance may vary across the 3D space. For points within the box region where

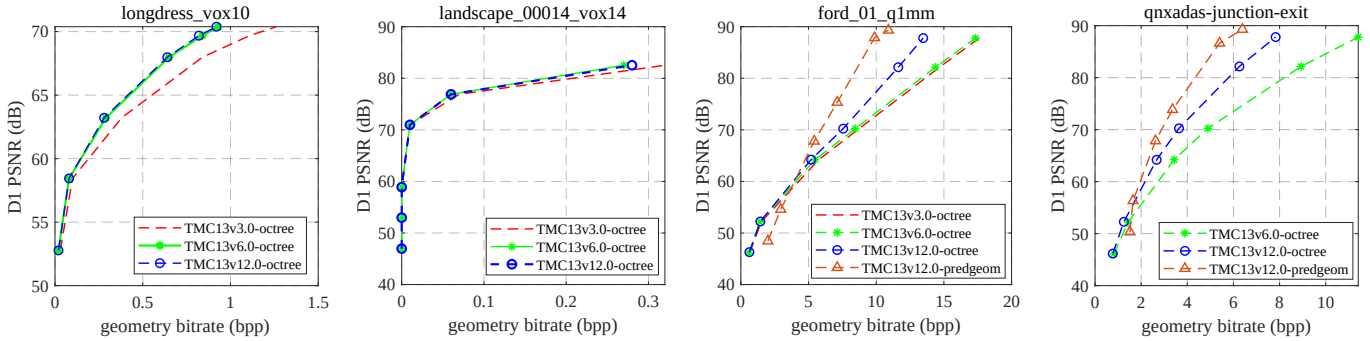

Fig. 16. Examples of G-PCC geometry coding performance comparison between TMC13v3, TMC13v6 and TMC13v12.

QP value is desired to be changed, a region QP offset value will be added on top of the the slice QP. The derivation of the quantization step size is shown in Fig. 15. In conclusion, the region-based adjustment provides the quality control in spatial domain while the layer-based adjustment provides quality control in frequency domain.

C. Scalability

G-PCC can achieve regional scalability and spatial scalability [43]. Regional scalability is the partial access to a region. This can be done with the slice and tile features. In contrast, spatial scalability is the access to data with small spatial resolutions. If octree coding is used for geometry, the point cloud can be decoded with partial resolution as thumbnail with less complexity and bandwidth. It is also desirable to decode the corresponding attribute bitstream harmonized with partial geometry. To do so, an octree harmonized LoD construction [44] is used to grouping points into refinement layers. The harmonization allows immediate decoding of the smaller resolution data than the original with both geometry and attribute information present.

D. Combined frame coding

In frame-based point cloud contents, each frame may be relatively smaller in file size which is less efficient for the I/O interface. The overhead of initializing decoder becomes more significant in the edge device. In this regard, combined frame coding technique [45] is proposed to address both issues by introducing the encoding of frame index in the combined Group of Point cloud (GOP). As consecutive point cloud frames are highly correlated, this technique also improves the coding efficiency largely so that it could be also beneficial for storage usage of frame-based point cloud content. The frame indices, which are used in the decoder to reconstruct the input frames, are encoded in the bitstream.

E. Adaptive axis coding order

This functionality introduces a mechanism to globally map the output axes labels to the internal axes [46]. More precisely, a syntax is added to identify one of six possible permutations of x, y, z to be used to label the three components of the position coordinates. This mapping provides flexibility to systems that have different native orders. The chosen permutation applies to both geometry and attribute coding for all frames in a sequence. A non-normative encoder-side adaptive axis order determination method [47] based on the point cloud density is also included in the codec to select the best axis coding order.

V. G-PCC Performance

The performance of G-PCC for different data categories is reported in this section under the CTCs [48] set by the G-PCC group. The test sets used are G-PCC datasets including Category 1-A, Category 1-B, Category 3-fused and Category 3-frame data.

A. Evaluation methodology

For lossy compression, the geometry distortion is measured by the point-to-point (D1) and point-to-plane (D2) metrics [49]. The attribute distortion is measured by PSNR. Different bitrates with various quantization step sizes are set in terms of applications. Bjontegaard-delta (BD) rate [50] calculated as the measurement of coding gains of the test method against the anchor method. For lossless compression, the compression ratio of the compressed data size over the original data size is used for comparison.

To reflect the progress of G-PCC standardization activities, the latest test model (i.e., TMC13v12) is compared to its early version (i.e., TMC13v3 released in 2018) and midterm version (i.e., TMC13v6 released in 2019). More specifically, TMC13v6 and TMC13v12 are benchmarked taking TMC13v3 as the anchor. It is noted that TMC13v1 is not chosen as the baseline since at that time a lot of tools in the model were immature to make a formative comparison.

B. Coding performance

Figure 16 illustrates the performance of different geometry coding tools in G-PCC. The four subfigures are drawn with point clouds of different types (e.g., human body, 3D scene, and LiDAR content from left to right). Since predictive geometry coding scheme is mainly designed for automotive use case, its performance is only shown for LiDAR acquired point cloud. It is obvious that the performance of G-PCC significantly improved from TMC13v3 to TMC13v12 on all

TABLE II
PERFORMANCE COMPARISON BETWEEN THE PREDICTIVE GEOMETRY SCHEME (PREDGEOM) AND THE OCTREE-BASED GEOMETRY CODING SCHEME UNDER LOSSY CONDITION.

	octree+RAHT vs.predgeom+RAHT					
	Geometry		Color			Reflectance
	D1	D2	Y	U	V	
Cat1-A average	408.70%	19.60%	2.0%	3.2%	3.6%	N/A
Cat1-B average	546.00%	103.30%	1.7%	2.3%	1.0%	N/A
Cat3-fused average	245.00%	156.40%	13.3%	3.5%	3.3%	2.8%
Cat3-frame average	-8.80%	-30.10%	N/A			-3.80%

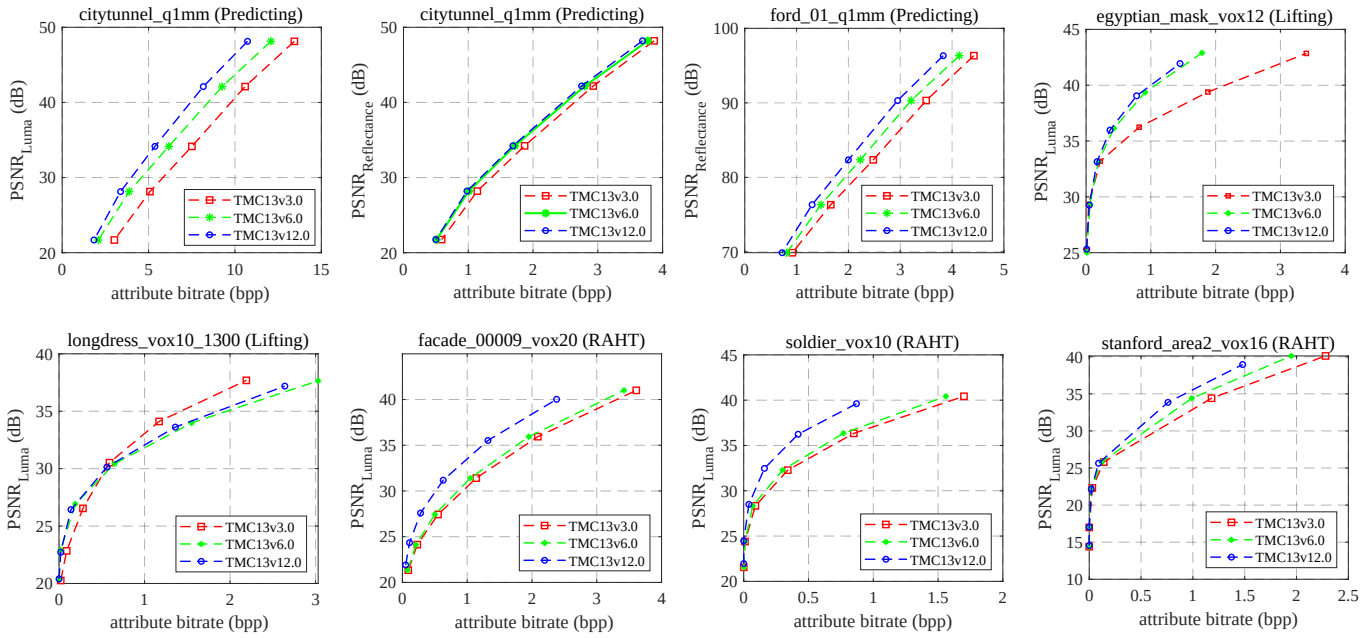

Fig. 17. Examples of G-PCC attribute coding performance comparison between TMC13v3, TMC13v6 and TMC13v12.

TABLE III
PERFORMANCE COMPARISON BETWEEN THE PREDICTIVE GEOMETRY SCHEME (PREDGEOM) AND THE OCTREE-BASED GEOMETRY CODING SCHEME UNDER LOSSLESS CONDITION.

	octree+predicting vs.predgeom+predicting)		
	Geometry	Color	Reflectance
Cat1-A	137.30%	100.00%	N/A
Cat1-B	103.70%	100.00%	N/A
Cat3-fused	85.50%	100.00%	100.00%
Cat3-frame	86.90%	N/A	99.80%

data categories. It is also noteworthy that the predictive geometry coding scheme outperforms the octree-based scheme on Ford_01_q1mm and qnxadas-junction-exit which are Category 3-frame data. A more comprehensive comparison between the predictive geometry scheme and octree-based scheme for each data category can be found in Table. II and Table. III for lossless and lossy conditions, respectively. For comparison, octree-based scheme is selected as the anchor. As shown in Table. II, the octree-based scheme is noticeably more efficiency for Category 1 and Category 3-fused data on average. On the other hand, predictive geometry scheme is able to achieve 8.8% and 30.1% coding gain on Category 3-frame sequences on average. For lossless condition, predictive geometry scheme has a coding gain of approximately 15% over octree-based scheme, as shown in Table. III.

As for attributes, Fig. 17 illustrates the performance of different attribute coding tools in G-PCC. Depending on the use case and test conditions (lossy, lossless and near-lossless), one tool may be selected for a particular dataset. In Fig. 17, predicting transform is selected for automotive use case where near-lossless or even lossless attribute coding are desired; lifting transform and RAHT are performed on relatively dense point clouds. Experimental results show that the all attribute coding tools significantly improved over the past twos years for coding color and reflectance information. According to the evaluation results in MPEG G-PCC group, if compared between the LoD scheme and RAHT scheme, LoD schemes gives better compression performance for uniformly sampled point clouds such as solid human body and object, while RAHT provides higher coding efficiency for noisy points clouds or irregularly sampled point cloud such as the LiDAR

TABLE IV
PERFORMANCE OF OCTREE-BASED GEOMETRY CODING COMBINED WITH PREDICTING ATTRIBUTE CODING FOR LOSSLESS CONDITION

	TMC13v3 vs. TMC13v6			TMC13v3 vs. TMC13v12		
	Geometry	Color	Reflectance	Geometry	Color	Reflectance
Cat1-A average	96.90%	95.90%	N/A	94.20%	75.70	N/A
Cat3-fused average	99.10%	93.40%	98.70%	97.40%	85.10%	95.30%
Cat3-frame average	98.80%	N/A	97.70%	74.20%	N/A	90.20%

TABLE V
PERFORMANCE OF OCTREE-BASED GEOMETRY CODING COMBINED WITH LIFTING ATTRIBUTE CODING FOR LOSSLESS GEOMETRY LOSSY ATTRIBUTE CONDITION

	TMC13v3 vs. TMC13v6					TMC13v3 vs. TMC13v12				
	Geometry	Color			Reflectance	Geometry	Color			Reflectance
		Y	U	V			Y	U	V	
Cat1-A average	96.90%	-15.70%	-15.00%	-15.20%	N/A	94.20%	-20.70%	-20.30%	-28.20%	N/A
Cat3-fused average	99.10%	-23.20%	-21.40%	-22.00%	-19.20%	97.40%	-25.60%	-24.50%	-25.30%	-22.20%
Cat3-frame average	98.80%	N/A			-32.50%	74.20%	N/A			-49.10%

TABLE VI
PERFORMANCE OF OCTREE-BASED GEOMETRY CODING COMBINED WITH RAHT ATTRIBUTE CODING FOR LOSSY GEOMETRY LOSSY ATTRIBUTE CONDITION

	TMC13v3 vs. TMC13v6						TMC13v3 vs. TMC13v12					
	Geometry		Color			Reflectance	Geometry		Color			Reflectance
	D1	D2	Y	U	V		D1	D2	Y	U	V	
Cat1-A average	-14.00%	-13.90%	-24.30%	-25.10%	-24.80%	N/A	-14.70%	-14.60%	-52.40%	-53.90%	-53.00%	N/A
Cat1-B average	-6.00%	-6.10%	-15.10%	-15.80%	-15.30%	N/A	-8.10%	-8.10%	-39.00%	-41.20%	-39.00%	N/A
Cat3-fused average	-5.60%	-5.60%	-13.20%	-13.10%	-13.00%	-9.50%	-6.80%	-6.80%	-32.60%	-27.80%	-27.30%	-29.90%
Cat3-frame average	-4.20%	-4.20%	N/A			-16.80%	-12.40%	-12.00%	N/A			-37.20%

fused data. Interested readers can refer to [51] for more information.

In addition to the coding performance of one individual tool on a specific point cloud shown above, the average performance comparisons between TMC13v3, TMC13v6 and TMC13v12 on each data category under different test conditions were also performed. In Table. IV, the performance of octree-based geometry coding combined with predicting attribute coding under lossless condition is reported. Table. V shows the performance of octree-based geometry coding combined with lifting attribute coding for lossless geometry lossy attribute condition. Lossy geometry and lossy attribute coding results are reported in Table. VI for the combination of octree-based geometry coding and RAHT attribute coding. Results in all three tables demonstrate rapid progress of MPEG G-PCC group over the last two years. For geometry, an average gain of 25.8% is witnessed for Cat3-frame data in terms of lossless compression. For lossy attribute compression, the coding performance improved 52.4% for Cat1-A dataset and 49.1% for Cat3-frame data on average, respectively.

VI. DISCUSSIONS AND CONCLUSIONS

As the G-PCC standardization activity is approaching to the International Standard (IS) status, one of the most significant topics currently under discussion in G-PCC group is the definition of the profiles and levels. There are currently four preliminary profiles under consideration [52]: Main Profile, Simple Profile, Predictive Profile and Dense Profile. It should be noted that the name of four profiles may be modified in the future by MPEG G-PCC group. Among these four profiles, the Main profile targets on the general use case while the Simple profile targets on low-complexity scenarios. Predictive profile is specifically designed for angular coding for LiDAR applications. Dense profile is of higher complexity for better coding performance on dense content.

With the rapid progress of immersive media technology, new use cases of point cloud and new requirements of point cloud compression may be identified by the market in the near future. MPEG is considering to start the Phase II standardization activities to extend G-PCC with new features and functionalities. Technologies such as point cloud inter prediction will be considered in the next phase.

In conclusion, this paper summarizes the geometry-based point cloud compression scheme designed by MPEG 3DG group at the FDIS stage. Major coding tools handling geometry and attribute information are described. Different aspects of the G-PCC functionalities are also introduced. The coding performance of G-PCC for different data categories and various conditions are also reproted. Further information related to the technical contributions are available in the MPEG document management system. Readers can also visit the MPEG-PCC project website (https://mpeg-pcc.org/) for more details.

ACKNOWLEDGMENT

The authors would like to thank the experts of ISO/IEC MPEG 3DG for their contributions.

REFERENCES

[1] "Use cases for point cloud compression," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. N16631, 2016.
[2] "Call for proposals for point cloud coding v2," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. N16763, 2017.
[3] L. Szirmay-Kalos and T. Umenhoffer, "Displacement mapping on the gpu - state of the art," in *Computer Graphics Forum*, vol. 27, no. 6. Wiley Online Library, 2008, pp. 1567–1592.
[4] P. S. Heckbert, "Survey of texture mapping," *IEEE computer graphics and applications*, vol. 6, no. 11, pp. 56–67, 1986.
[5] "Text of iso/iec cd 23090-21 reference software for g-pcc," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. W19643, 2020.
[6] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 249–270, 1980.
[7] R. Schnabel and R. Klein, "Octree-based point-cloud compression." *Spbg*, vol. 6, pp. 111–120, 2006.
[8] "Neighbour-dependent entropy coding of occupancy patterns in TMC3," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M42238, 2018.
[9] "Intra mode for geometry coding in TMC3," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M43600, 2018.
[10] "Planar mode in octree-based geometry coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M48906, 2019.
[11] "Inference of a mode using point location direct coding in TMC3," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M42239, 2018.
[12] "Implicit geometry partition for point cloud coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M49231, 2019.
[13] "On further reduction of neighbour configurations," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M46148, 2019.
[14] "A new neighbour node occupancy decision," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M44752, 2018.
[15] "Entropy coding an octree node occupancy depending on neighbour's child nodes," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M44753, 2018.
[16] "The new azimuthal coding mode," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M51596, 2020.
[17] "[GPCC] [CE13.22-related] an improvement of the planar coding mode," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M50642, 2019.
[18] "[G-PCC][EE13.6-related]slice-based geometry quantization," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M50927, 2019.
[19] "[G-PCC] normative bounding box and global scaling for point clouds," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54584, 2019.
[20] "[G-PCC] EE13.6 report on geometry quantization," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M50924, 2019.
[21] "Predictive geometry coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M51012, 2019.
[22] S. Gumhold, Z. Kami, M. Isenburg, and H.-P. Seidel, "Predictive point-cloud compression," in *ACM SIGGRAPH 2005 Sketches*, 2005, pp. 137–es.
[23] "CE13.38 report on angular coding for the predictive geometry coder," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54671, 2020.
[24] "G-pcc codec description," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. N19620, 2020.
[25] "Efficient implementation of the lifting scheme in TMC13," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M43781, 2018.
[26] R. L. De Queiroz and P. A. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, 2016.
[27] G. P. Sandri, P. A. Chou, M. Krivokuća, and R. L. de Queiroz, "Integer alternative for the region-adaptive hierarchical transform," *IEEE Signal Processing Letters*, vol. 26, no. 9, pp. 1369–1372, 2019.
[28] "PCC CE1.3 recolor method," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M42538, 2018.
[29] "[G-PCC] [EE13.43] report on coordinate conversion," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54622, 2020.
[30] "Ce13.6 report on attribute lod construction and neighbour search," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54688, 2020.
[31] "[G-PCC [new proposal] efficient low-complexity lod generation," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M46188, 2019.
[32] "On an improvement of raht to exploit attribute correlation," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M47378, 2019.
[33] "[G-PCC] [EE13.7] improved encoding for inter-channel residual prediction," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M52719, 2020.
[34] "Attribute residual coding in TMC13," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M46108, 2019.
[35] "[G-PCC] [CE13.33] report on modifying entropy coding of attributes coefficients with dictionary removal and run-length coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54265, 2020.
[36] "[G-PCC: A raw attribute coder," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M53679, 2020.
[37] "G-PCC CE13.2 report on tile and slice based coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M48892, 2019.
[38] "[G-PCC] [new] software refinement to achieve to the tile partitioning," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54651, 2020.
[39] "[G-PCC] [new] implementation of entropy continuation for attribute," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54659, 2020.
[40] "G-PCC CE13.29 report on in-tree geometry quantisation," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M53389, 2020.
[41] "[G-PCC] CE13.19 report on attribute layer quantization control," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M47834, 2019.
[42] "[G-PCC] EE13.5 report on region-wise quantization control," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M51063, 2019.
[43] "Spatial scalability support for G-PCC," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M47352, 2019.
[44] "[G-PCC] [new] integration of decimation-based and scalable lod generation," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M55344, 2020.
[45] "Tmc13 new proposal on combine frame coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M44813, 2018.
[46] "G-pcc ce13.26 report on geometry swizzling," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M51516, 2020.
[47] "[G-PCC] CE13.26 report on raht transform order," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M53332, 2020.
[48] "Common test conditions for G-PCC," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. N19642, 2020.
[49] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3460–3464.
[50] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.
[51] "[G-PCC] EE13.46 review of v11 attribute coding," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54166, 2020.
[52] "[G-PCC] profile and level," *document ISO/IEC JTC1/SC29/WG11 MPEG*, no. M54166, 2020.