INVITED PAPER

Compression of Sparse and Dense Dynamic Point Clouds—Methods and Standards

By Chao Cao, Marius Preda, Vladyslav Zakharchenko, Euee S. Jang, *Senior Member IEEE*, and Titus Zaharia

ABSTRACT | In this article, a survey of the point cloud compression (PCC) methods by organizing them with respect to the data structure, coding representation space, and prediction strategies is presented. Two paramount families of approaches reported in the literature—the projection- and octree-based methods—are proven to be efficient for encoding dense and sparse point clouds, respectively. These approaches are the pillars on which the Moving Picture Experts Group Committee developed two PCC standards published as final international standards in 2020 and early 2021, respectively, under the names: video-based PCC and geometry-based PCC. After surveying the current approaches for PCC, the technologies underlying the two standards are described in detail from an encoder perspective, providing guidance for potential standard implementors. In addition, experiment evaluations in terms of compression performances for both solutions are provided.

KEYWORDS | 3-D graphics; compression; multimedia systems; point clouds; realistic modeling; standards.

I. INTRODUCTION

Recent advances in computer graphics have made it possible to create realistic digital representations of 3-D objects and surroundings allowing real-time interactions with users. Such representations use various mathematical models to describe both geometries and attributes (colors, materials, and lighting behavior) of the objects. Among them, 3-D meshes have become popular and extensively used in the gaming and 3-D movie industries. Nevertheless, the creation of high-fidelity content requires enormous computational resources for storage, transmission, processing, and visualization purposes. This problem penalizes their deployment on light mobile devices.

Recently, due to the progress of advanced 3-D scanning technologies, a different approach, based on 3-D point cloud representations (PCRs), has emerged. A 3-D point cloud is a simple data structure defined as a set of unorganized points in the 3-D space with associated photometric attributes and can represent both static and dynamic 3-D objects.

PCRs are well-suited for various industrial applications, including 3-D scanning and modeling, environmental monitoring, agriculture and forestry, and autonomous driving, being able to fulfill different requirements in terms of precision, point density, and related attributes.

Therefore, point clouds with high density (up to millions of 3-D points) are required for hyperrealistic visualization applications, which raise enormous demands in terms of computational and memory requirements. Sparse point clouds are used for autonomous navigation applications, but high precision is required, which needs to be preserved. Moreover, for applications involving dynamic point clouds, the supplementary temporal dimension explodes the related bandwidth and needs to be appropriately considered.

Manuscript received June 8, 2020; revised March 21, 2021 and May 11, 2021; accepted May 28, 2021. *(Corresponding author: Chao Cao.)*

Chao Cao, Marius Preda, and **Titus Zaharia** are with the Télécom SudParis, Institut Polytechnique de Paris, 91011 Évry-Courcouronnes Cedex, France (e-mail: cao_chao@telecom-sudparis.eu).

Vladyslav Zakharchenko is with Ofinno LLC, Reston, VA 20190 USA (e-mail: vzakharchenko@ofinno.com).

Euee S. Jang is with the Department of Computer Science, Hanyang University, Seoul 04763, South Korea (e-mail: esjang@hanyang.ac.kr).

Digital Object Identifier 10.1109/JPROC.2021.3085957

With the increasing demand for different point clouds, developing efficient compression methods and technologies that can consider various application constraints has become a critical challenge. Recently, numerous studies have investigated 3-D point cloud compression (3DPCC). However, the related constraints are treated heterogeneously by various state-of-the-art approaches. The ISO [/IEC]'s Moving Picture Experts Group (MPEG) Standardization Committee identified this bottleneck and launched a 3DPCC-dedicated activity in 2014. After many years of intensive developments, involving both academic and industrial parties, MPEG issued two-point cloud compression (PCC) standards, as part of MPEG-I Coded representation of immersive media. The approach adopted by MPEG is based on the optimization and extension of the most pertinent state-of-the-art methodologies while ensuring the coverage of an as wide as a possible field of applications and related functionalities. Therefore, two distinct families of approaches have been retained. One called video-based PCC (V-PCC) adopts a projection-based methodology and handles highly dense 3-D point clouds. The other, geometry-based PCC (G-PCC) is more specifically devoted to the management of sparse 3-D point clouds.

The contributions of this study are twofold. First, we propose an extensive survey of the 3DPCC state-of-the-art methods, describing related principles and analyzing advantages and limitations. Particular attention is focused on the models adopted for MPEG standardization, and we notably highlight the features that guided their adoption. Second, we give a comprehensive overview of the two MPEG V-PCC and G-PCC approaches, with an extensive objective experimental evaluation. Some further optimizations are also presented.

The rest of this article is organized as follows. Section II provides a comprehensive analysis of various PCC approaches reported in the literature, followed by a brief introduction of the development history of the PCC standard by the MPEG committee, explaining why the proposed standard was published in two parts. The projection-based approach—V-PCC—and the octree-based approach—G-PCC—are detailed in Sections III and IV, respectively. The objective results are evaluated at the end of each section for V-PCC and G-PCC, respectively. Finally, Section V concludes this study and highlights some perspectives of future work.

II. STATE OF THE ART

Recently, various compression methods have been proposed for point clouds with various features and densities.

In this section, the compression methods are classified into three based on their targeted 3-D objects and methodology: 1) compression for static point cloud objects; 2) compression for dynamic point cloud objects; and 3) compression using deep learning (DL) methods. To provide a distinct evolutionary view for compression technologies, the approaches presented in this section include both the initial and recent studies, as well as both representative methods and optimization-focused approaches.

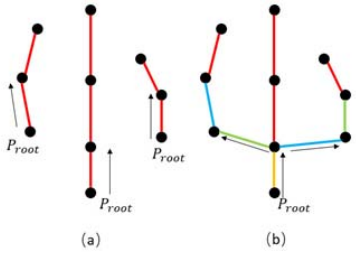


Fig. 1. *Examples of predictive trees where P_{root} represents the root node: (a) spanning tree predicting from a parent point as detailed in [1] and (b) spanning tree using four predictors [base (yellow), left (green), right (blue), and front (red)] as detailed in [2].*

A. Compression for Static Point Cloud Objects

A static point cloud, unlike a mesh, has no connectivity information between points. The unorganized nature of the point cloud makes it difficult to implement prediction techniques because of the lack of topological neighborhood information. Initially proposed methods attempt to derive some *ad hoc* topology by constructing connectivity structures, such as trees, the level of details (LODs), or clusters, to transform the 3-D point clouds into a more compression-friendly representation.

1) Methods Based on Topological Structures: In [1], a prediction tree was constructed by exploiting the geometric correlations between points [see Fig. 1(a)]. Then, four different predictors [see Fig. 1(b)] [2] were used to indicate the direction of the child nodes from a root (parent) node. For both methods, the constructed trees were encoded in breadth-first order. The resulting residuals were quantized and finally coded with an adaptive arithmetic coder.

Another approach is based on the concept of spanning trees. Within this framework, the most used structures are the kd-trees [3] and octrees [4], [5]. They share a similar mechanism by subdividing the given 3-D space into subspaces represented by nodes. Thus, a kd-tree recursively subdivides the space into a k-dimensional binary representation and is used for efficient neighbor search to provide information for connectivity coding as in [6].

The octree representation has become highly popular for yielding progressive 3-D structures for point clouds. Its principle consists of recursively splitting 3-D space from a parent node into eight children nodes with the same bounding box size.

Octrees were first employed for PCC in [7], with a limited number of LODs. Subsequently, the approach has been improved by the so-called superior full-range progressive coder, which optimizes the octree traversal order. A lossless octree-based compression method was also presented in [8]. A set of contexts was generated to describe the

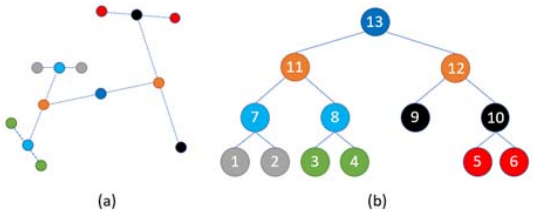


Fig. 2. *(a) Resulting point cloud clusters (with the same color) using the point pair clustering algorithm. (b) Example of a BT constructed with the bottom-up order.*

Fig. 3. *(a) PCR of a soldier (data set from [22]). (b) Voxelized soldier.*

binary pattern of the octree decomposition, thus enabling efficient arithmetic coding.

The spatial correlation, as well as the coherence of attributes between 3-D points, cannot be fully exploited with tree-based structures alone. Therefore, methods that gather points into clusters and construct a hierarchy of LODs have been proposed to achieve adaptive bitrate and quality for compression.

2) Clustering and Hierarchical LOD-Based Approaches: A clustering-based compression technique, which supports multiple LODs, was introduced in [9]. Both geometric and color features were considered for clustering the point cloud into a set of planar patches. Then, Gaussian mixture models (GMMs) were estimated with the help of an expectation–maximization algorithm and used to replace the clustered points within each planar patch with the corresponding GMM approximation. However, the inevitably lossy process led to inaccurate approximations over discontinuous parts of the point cloud.

Based on a method proposed in [10] for progressive geometry coding using octree, a generic point cloud coder that clusters the color information was introduced in [11]. Adaptive color quantization was achieved by applying the principal component analysis (PCA) and the generalized Lloyd algorithm (GLA)—a clustering algorithm widely used in the context of vector quantization [12] and pattern recognition [13]. In addition, an adapted GLA was proposed in [14] to construct a hierarchy of LODs.

A hierarchical LOD structure was also proposed in [15] based on a point pair clustering. The hierarchy was built by decomposing the point cloud at each step into disjunctive pairs (see Fig. 2).

The parent point was contracted with the averaged attributes of the paired points [see Fig. 2(a)]. A binary tree (BT) forest [see Fig. 2(b)], describing neighborhood relations, was finally constructed in a bottom-up manner. The resulting graph presented an LOD-like structure.

An LOD structure represented as a BT quadtree (BTQT) was introduced in [16]. A BT was built for occupancy detection to avoid encoding the full depth octree. The occupancy of "0" meant that the occupied points could be predicted from a local surface. Then, the corresponding points were projected onto the 2-D plane and encoded with a quadtree (QT). The other points with occupancy "1" from the BT were encoded with the full octree.

The voxel grid (VG) is another LOD structure that subdivides the 3-D space into voxels, which are centroid representations for a cluster of points within the given bounding box. The sparse voxel arrays were described as voxelized point clouds in [17] and [18] and exploited in [6], [8], and [19]–[21]. An example is illustrated in Fig. 3.

Most described methods consider the signals as discrete samples, which inherit the same nondifferentiable and nonprobabilistic characteristics as point clouds. Continuous approaches were introduced for compression by transforming a signal into a different representation space.

3) Transform-Based Methods: In [23], a wavelet-transform-based compression method using an atlas of height fields was proposed. The objective was to point clouds onto a 2-D image-like structure that could be efficiently encoded with image-based coding techniques. Inspired by Pauly and Gross [24] and Sander *et al.* [25], a heuristic split-and-merge partition of the point set was performed with the help of a mean least-squares approximation approach [see Fig. 4(a)]. The partitioned cluster was parameterized as a height field [see Fig. 4(b)], over the corresponding PCA-computed base plane. Then, each parametrized height field was projected onto the base plane to generate a 2-D depth [see Fig. 4(c)], which was compressed with a wavelet-based shape-adaptive image

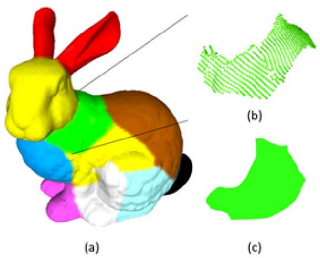


Fig. 4. *(a) Example of a clustered bunny in the 3-D PCR, where (b) clustered 3-D points are projected onto a respective base plane to generate (c) a 2-D image.*

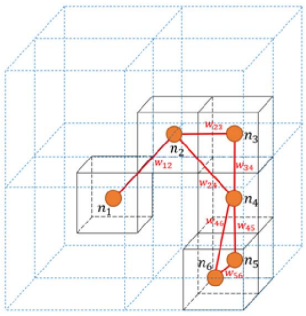


Fig. 5. ***Example of constructed graph based on octree-decomposed point cloud; n_i represents the ith node of the graph, and w_{ij} is the weight value connecting the ith and jth nodes.***

coding technique [26]. In addition, a rate-distortion optimization (RDO) step ensured progressivity.

A dedicated color compression scheme employing the 2-D discrete cosine transform (DCT) was proposed in [27]. The method involves a global color segmentation and a local geometric segmentation. A mean shift clustering algorithm [28], [29] was employed to iteratively cluster points with similar colors into a higher density region while avoiding distant points. Octrees were used to map smooth color streams onto a uniform 2-D grid via a z-scan. The projected stream was further encoded with 2-D-DCT. Notably, 3-D-DCT methods can also be considered within this context. Such approaches were explored in [30] and [31].

Apart from trees, graphs are also generic data representations that are highly useful to describe signals. A graph Fourier transform (GFT) [32]—an extension of the Karhunen–Loève transform (KLT) [33] to graphs—was first explored in [19] to decorrelate the attribute signal of a point cloud. The neighboring points were connected as a set of graphs within small neighborhoods of the point cloud, exploiting an octree-decomposed VG (see Fig. 5).

The color signal defined on the constructed graph nodes can be transformed with the help of a precision matrix [32] and further entropy coded. Although the graph transform in [19] outperformed traditional DCT methods, it created isolated subgraphs for sparse point clouds. To improve the efficiency, block-based prediction and k-nearest neighbors graph transforms were proposed in [34].

An optimized graph transform scheme with a kd-tree partition was introduced in [6]. The Laplacian sparsity for graph transform was optimized in [6] with an off-line training stage, followed by an RDO-based quantization procedure. The scheme was further optimized with a layered structure and block-based intraprediction in [35]. In addition, a normal weighted GFT [36] and a color-adaptive intra prediction method with context-based arithmetic coding [37] were proposed to optimize the GFT for attribute compression.

However, although the GFT-based methods exhibit better energy concentration than DCT, no complete and generic solution to the resulting subgraph problems has been proposed.

To overcome the complex matrix decomposition in the GFT, a so-called region-adaptive hierarchical transform (RAHT) approach was proposed in [20]. An octree-based RAHT process was considered to be reduced to the Haar transform when the region was regular and the weights were uniform. Notably, RAHT was used within the test model (TM) [38] for PCC standardization by MPEG—an aspect that is detailed in Section IV.

The Gaussian process transform (GPT), as introduced in [21], describes a modeling process of the statistics of point colors by assuming that they are samples of a stationary Gaussian process defined over an infinite 3-D space. The covariance function of the Gaussian process was used to derive the covariance matrix of the colors, which, in turn, was used to obtain the KLT of the color signal. It was demonstrated in [21] that the GPT-based methods outperformed the former GFT-based methods with the proposed entropy coder in terms of energy compaction, transform coding gain, and distortion-rate performance.

Remarkably, some of the abovementioned approaches, such as those in [7], [17], and [18], are also considered for dynamic 3DPCC. However, we consider dedicated approaches that significantly reduce the temporal redundancy between point clouds. State-of-the-art 3DPCC approaches are presented in Section II-B.

B. Dynamic 3DPCC

Several coding schemes for real-time compression have been proposed for point clouds obtained from multicamera systems. A generic point cloud codec was presented in [39] for a low-cost system using multiple Microsoft Kinects as input devices. Based on octree subdivision and occupancy coding, color information could be encoded with an efficient JPEG codec after projecting the point cloud onto an image grid. Similarly, image coding was used in [23] with a heightmap and in [40] with network conditions and application-related parameters as constraints.

In [39] and [41], the iterative closest point algorithm [42] was used with a skeleton-based motion estimation for Kinect-captured point clouds. Each frame was clustered into point sets associated with off-line-trained human skeletons [43]. The color and depth information, as well as the motion compensation residuals, was stored within an atlas. Then, the points were projected onto a regular 2-D grid embedded in a cylindrical coordinate system defined around the skeletons. Finally, the MPEG AVC/H.264 video codec was used to compress the resulting 2-D videos.

Besides 3-D-to-2-D projection-based methods, some pure 3-D approaches have been proposed. A differential octree was adapted for streaming applications in [44]. After the octree decomposition, the difference between the

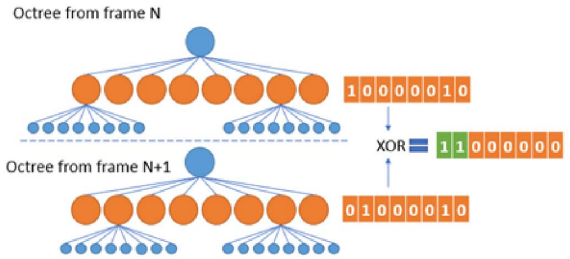


Fig. 6. ***Illustration of the differential coding technique where an XOR operation is executed to compute the difference between the octree data structures of two consecutive frames.***

octree data structures of successive frames was computed and encoded. As illustrated in Fig. 6, the octree structures of two consecutive frames were built and serialized into binary sequences separately representing the occupancy code.

Then, they were compared using the XOR operator to detect the changing part. Finally, the resulting difference vector was entropy encoded. The main advantage of this method is the significant reduction of the computational and memory requirements.

The lossless interframe compression method, introduced in [45], reorders the octree structure in the current point cloud frame based on the one in the previous frame. The predicted and current octrees are constructed to indicate the presence of the 3-D points in a given level in the form of binary symbols. Then, the binary symbols were sorted in ascending order to be compressed more efficiently.

The GFT was proposed in [17] and further developed in [46], with a complete framework for dynamic colored PCC. Based on extracted spectral features of constructed graphs, motion estimation was cast as a matching problem between successive graphs. Therefore, motion compensation and differential coding were further employed.

A 3-D cube-based motion compensation method for dynamic voxelized PCC was presented in [18]. The voxels were matched between consecutive frames by computing the level of relevance using their geometry and color information. The geometry and color residuals were considered motion vectors and color prediction errors, respectively. Then, the decision of coding the block with intracompensated or motion-compensated mode was determined using an RDO approach.

Another dynamic point cloud is the one obtained from light detection and ranging (LiDAR) scanners. In this case, the captured objects and scenes are of high precision (with an accuracy down to several millimeters) but sparse. The dynamic LiDAR point clouds can be captured using a sensor system; an example is illustrated in Fig. 7 for a mobile mapping system (MMS) and autonomous navigation.

Several laser scanners were mounted on the vehicle's top to emit laser beams and receive the reflections. RGB cameras were also mounted to capture image sequences. Based on a global positioning system (GPS) and inertial sensors, the generated point clouds can be converted into an absolute coordinate system. After a fusion process, dynamic sparse point clouds with coordinates along with the reflectance and RGB attributes can be achieved. In addition, other attributes, such as timestamps, longitude, and latitude, can also be included in the representation.

A wavelet-based LiDAR PCC framework, the so-called wavelet LiDAR zipper, was proposed in [48]. Each coordinate component (X, Y, or Z) was independently wavelet-transformed as a 1-D vector. Thus, highly correlated points, such as consecutively recorded points, generated small and similar detail coefficients, which could be efficiently compressed with a standard Huffman coding. Similarly, the so-called LiDAR compressor technology [49] performed a simplified Haar wavelet transform as in [50] on each array of point attributes individually. For lossless compression, some other approaches, such as [51] and [48], were proposed for LiDAR points in the LAS format sharing a similar compression scheme.

It was shown in [52] that efficient prediction can be achieved on line-by-line captured MMS point clouds with the computation of second-order difference. The principle was further exploited in [53], where 3-D point clouds were mapped to a 2-D pixelized domain, depending on the GPS time and parameters of the laser scanner. Then, the projected point clouds were segmented into growing regions by connecting the points with a directed spanning tree. The projected point clouds could be compressed without deteriorating the quality, exploiting a portable networking graphic compression. Besides the line-by-line scanning order, circular scanning provides a 360° representation of the environment with a point cloud per scan (frame), as in [54]. The raw point cloud data are converted into range images, and various image/video compression algorithms can be exploited to reduce the corresponding volume.

However, the 3-D characteristics of point clouds are not fully exploited by projecting point clouds onto images. To achieve accurate 3-D prediction, the simultaneous localization and mapping-based compression method was

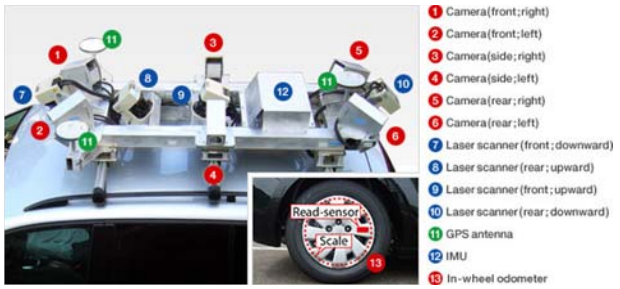


Fig. 7. ***Example of the sensor system for generating LiDAR point clouds [47].***

presented in [55]. The method jointly exploited the location and orientation information. The predicted beams were achieved by simulating scanner beams with the yaw angle (rotation angle of the sensor) and pitch angle (laser ID) of each beam.

Another dynamic 3DPCC algorithm was presented in [56]. The method employed the depth modeling mode (DMM) technique in the 3-D extension of high-efficiency video coding (HEVC) for PCC. The proposed algorithm first converts point clouds into range images. An object segmentation process [57], [58] was employed to generate the prediction map according to the clustering results. Then, the generated contour map was compressed with an algorithm introduced in [59]. In addition, the values of each segmented region were organized in laser scanning order and encoded with an arithmetic coding scheme. Since the range map was similar to the depth image used in 3-D videos, the DMM was used for block partitioning and prediction. Several compression methods, including lossless LZ4, Lizard, BZip2, lossy JPEG, and JPEG2000, compress the residual data. In [56], the proposed method outperformed those in [54] and [55]. Furthermore, the lossless method could provide up to a 20:1 compression ratio without distortion.

C. Compression Using DL

Recently, several DL-based methods, such as those in [60]–[63], have been proposed to achieve efficient compression for point clouds. These DL approaches could be applied to both geometry and attribute compression.

1) Geometry-Targeted Methods: One of the first attempts of applying DL to PCC was presented in [62]; a novel data-driven geometry compression method for static point clouds based on learned convolutional transforms and uniform quantization was proposed. The model is a 3-D convolutional autoencoder comprising an analysis transform followed by a uniform quantizer and a synthesis transform. Joint optimization of both rate and distortion using a tradeoff parameter was performed during the training phase. In [64], the impact of several parameters of the model proposed in [62] was evaluated through a series of experiments. Based on that, a hyperprior model and deeper transforms, as well as fine-tuning of the loss function and adaptive thresholding, were proposed, and they improved the compression performance.

Similarly, Guarda *et al.* [65] proposed compressing geometry using an autoencoder consisting of a small number of convolution and deconvolution layers for analysis and synthesis. As in 2-D images, the solution takes advantage of the ability of convolutional neural networks to extract adaptive features from the point clouds to create a latent representation that can be efficiently coded. Then, they extended an approach in [66]; an alternative lower-complexity quantization approach that combined implicit–explicit quantization was proposed to effectively reduce the number of DL models that needed to be trained and stored.

A learning-based point cloud geometry compression (PCGC) method, so-called learned-PCGC, was presented in [67], which comprised a set of stacked 3-D convolutions, dedicated to the extraction of latent features and hyperpriors; a variational autoencoder model was proposed for accurate entropy modeling of latent features. In addition, a weighted binary cross-entropy loss was used for training, and an adaptive thresholding scheme in inference was used for correct voxel classification. Several preprocessing steps were employed, including voxelization, scaling, and partitioning, before feeding a model block-by-block to the network similar to the method in [62].

Different from the previous methods, in [68], no voxelization was applied. Instead, raw point clouds were fed to the proposed architecture. The PointNet network [69] was used to extract features from unorganized 3-D point clouds. The synthesis transform was represented by a generative fully connected network.

2) Attribute-Targeted Methods: A different system for compressing point cloud attributes using DL was proposed in [70]. A 2-D parameterization of point clouds was acquired by mapping the attributes from a point cloud to a grid, making it possible to employ 2-D image processing algorithms and compression tools. Inspired by Yang [71], where a model was trained on a data set to learn how to fold a 2-D grid onto a 3-D point cloud, the proposed method in [73] employed the folding network as a parametric function that mapped an input 2-D grid to points in the 3-D space. However, the lossy 3-D-to-2-D mapping introduced distortion for reconstruction and caused the low accuracy of folding in the geometrically complex parts of point clouds.

3) Joint Compression: Two alternative architectures, involving separate and joint compression of geometric and texture information, were presented and compared in [72]. For the separate compression scheme, two cascading networks were trained separately to encode geometry and color individually. Moreover, a unified model was proposed to holistically learn point clouds, allowing fine-tuning for quality preservation per attribute.

D. Discussion

The analysis of the state-of-the-art showed that numerous 3DPCC approaches exist today. The reason behind it is the nature of PCRs, ranging from sparse to dense, static to dynamic, and surface approximation to volumetric data. However, this analysis allowed us to identify two main approaches: a purely geometric one, based on an octree decomposition, and a hybrid (geometry and video) one, based on 3-D-to-2-D projections, followed by 2-D image/video coding. Besides, there is no solid proof that the DL-based methods achieve a sufficient degree of maturity that can cover the various use cases addressed by traditional methods.

In terms of compression efficiency, the projection-based method is performant because it relies on image/video coding advanced technologies, but it works only when the initial point cloud is dense enough so that the 2-D planes obtained by projection can be efficiently compressed by a 2-D video codec. Contrariwise, the geometric method can handle both dense and sparse point clouds but introduces an additional cost in the case of dynamic content; more details are given in Section IV.

The variety of point clouds was considered by the MPEG committee, which initiated an activity of producing standards for 3DPCC.

After three years of experimenting with different methods, MPEG selected two approaches (projection-based and octree-based) and modified them.

E. MPEG 3DPCC

From the early 1990s, MPEG has been standardizing compression standards for 3-D graphic objects, such as 3-D meshes, animated objects, and point clouds. For the compression of static 3-D mesh objects, the MPEG-4 3-D mesh coding was standardized in 1998. For generic and animated objects, the MPEG-4 animation framework extension was standardized in 2003 and has been continuously modified until recently.

The need for PCC in MPEG was first raised in 2013. MPEG began its exploration in 2014, which later generated much interest from the industry for immersive 3-D services based on point clouds. During the exploration, a reference compression model [39], originally built from point cloud library (PCL) [73], was developed and tested for three test cases: 1) static point cloud objects; 2) time-varying (or dynamic) dense point cloud objects; and 3) dynamically acquired point cloud objects (e.g., LiDAR). Based on PCC's exploration and rising interest from industry, MPEG issued a call for proposals on PCC in 2017. As a response, 13 technical proposals were collected and evaluated to select three reference TMs that addressed the compression efficiency of the abovementioned three test cases.

Three TMs were named TM category X (or TMCX, $X = 1$, 2, or 3), respectively. TMC1 and TMC3 provided compression based on octree decomposition of 3-D coordinates of a given point cloud, whereas TMC2 transformed 3-D coordinates into 2-D maps containing depth and color information from the given 2-D projection plane, followed by a 2-D video compression using MPEG video codecs, such as HEVC. In 2018, TMC1 and TMC3 were merged to become TMC13 since TMC1 and TMC3 shared many common coding tools, such as octree decomposition, encoding and decoding structure, and recoloring process. The names of the two remaining TMs were updated accordingly. TMC2 was named V-PCC because the compression was performed by a video codec after projection, whereas TMC13 was named G-PCC because it exploited a pure geometry-based representation. The MPEG issued the V-PCC and G-PCC as standards, as parts of MPEG-I *Coded representation of immersive media* in 2020 and 2021, respectively.

The standardization of V-PCC and G-PCC is likely to be ongoing for some years because the first version of the standards mainly focuses on the provision of high compression efficiency. Extended functionalities, such as better scalability, support of light-field data, increased bit depths, and point density, are among the features that future versions of V-PCC and G-PCC may address.

III. MPEG V-PCC

The main goal for V-PCC is to enable rapid deployment of PCRs on current devices by reusing existing hardware.

A. MPEG V-PCC Methodology

The V-PCC approach transforms 3-D geometry and the attribute data of point clouds into a set of 2-D patches. Then, these patches are mapped to a predefined set of 2-D planes through orthogonal projections, without self-occlusions, and with limited distortion. Next, a mapping between the point clouds and a regular 2-D grid is obtained by packing the projected patches within the 2-D domain. Afterward, the mapping is used to generate 2-D images representing the point cloud geometry and its attributes. Since the mapping between the point clouds and 2-D grid is not bijective, an extra binary image, referred to as the occupancy map, is needed to distinguish between the filled (i.e., associated with a point) and empty (i.e., unassociated with any point) cells of the 2-D grid. Similar to the geometry and attribute video sequences, the occupancy map video sequence is compressed using 2-D-based image or video codecs. To be able to reconstruct the 3-D point cloud from the 2-D geometry, attribute, and occupancy videos, additional information specifying per patch information (e.g., 3-D/2-D patch location and projection plane index) is required. Therefore, V-PCC introduces a new codec specifically optimized to handle the patch information substream. This substream occupies a relatively small amount (less than 5%) of the overall bitstream. Additional information needed to synchronize and link the video and patch substreams is signaled in the bitstream. All these components can be multiplexed into a single bitstream or encoded as separate tracks of an ISO base media file format container [74].

Similar to video encoding standards, the MPEG V-PCC only specifies the decoding process. The encoding process is out of scope, and it is left to an encoder manufacturer to create the most appropriate encoder that generates bitstreams conforming to the V-PCC specification. However, to simplify the explanation of how V-PCC represents compressed point clouds, we use the encoder diagram illustrated in Fig. 8.

We observe the intensive use of video codecs in the PCC coding architecture. Leveraging traditional video codecs to encode point clouds requires mapping the input point cloud to a regular 2-D grid. The objective is to find a temporally coherent, low-distortion, and injective mapping,

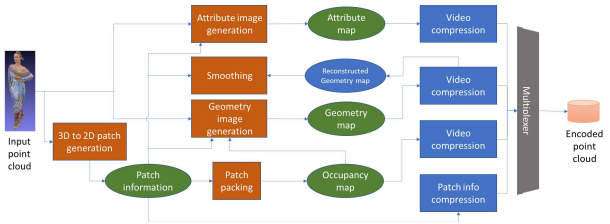


Fig. 8. ***V-PCC encoder diagram.***

which would assign each point of the 3-D point cloud to a cell of the 2-D grid. Maximizing temporal coherency while minimizing distance/angle distortions enables the video encoder to exploit the spatiotemporal correlation between the point cloud geometry and attribute signals. Injective mapping guarantees that all input points are captured by the geometry and attribute images and can be reconstructed without loss. By simply projecting that the point cloud on the faces of a cube or a sphere does not guarantee lossless reconstruction due to auto-occlusions (i.e., auto-occluded points are not captured), and it can generate significant distortions in practice, thus, a clustering process is performed before projection.

We now give further details of the various stages involved.

1) Patch Generation: V-PCC decomposes an input point cloud into a set of patches, which can be independently mapped, through a simple orthogonal projection, to a 2-D grid without resulting in auto-occlusions and without requiring any resampling of the point cloud geometry. Furthermore, the patch generation process aims at generating patches with smooth boundaries while minimizing their number and mapping distortions. Patch generation is outside the scope of the standard; however, an implementation of a heuristic segmentation approach (see Fig. 9) is proposed as an example.

First, the normal at every point is estimated as described in [75]. Then, the initial clustering of point clouds is obtained by associating each point with one of the six unit cube-oriented planes. Precisely, each point is associated with the plane that has the closest normal (i.e., maximizes the dot product of the point normal and plane normal). Next, the initial clustering is refined by iteratively updating the clustered index associated with each point based on its normal and the cluster indexes of its nearest neighbors. The final step consists of extracting patches by applying a connected component extraction procedure.

The number of generated patches depends on the complexity of the point cloud. Usually, less than a hundred patches are needed to capture more than 95% of the points. The remaining points are usually isolated points or isolated groups of points that would require a high number of patches to capture. Depending on the application constraints, the encoder may decide to totally discard the remaining points at the risk of introducing holes and cracks in the point cloud. To avoid such artifacts with a limited rate-distortion (RD) performance impact, V-PCC makes it possible to control the minimum number of points per patch, allowing the user to filter isolated points or small groups of isolated points that may have a limited impact on visual quality. The encoder may also choose to store all or a subset of the remaining points in special patches, named unprojected or independent point patches. Then, the Cartesian coordinates of the points corresponding to an independent point patch are directly embedded in the bitstream without using any prediction method. The independent patch geometry and attribute values could be stored in the same or separate video sequences as the regular patches.

2) Patch Packing: How to pack the different patches obtained by projection in a 2-D frame is also not part of the standard, but several methods were experimented with while developing the standard. The most efficient one iteratively tries to place patches in an image of W (width) $\times$ H (height) size. First, the patches are ordered by their sizes. Then, the location of each patch is determined by an exhaustive search in the raster scan order of the first location that guarantees overlap-free insertion. Precisely, the bounding box of the current patch expressed in terms of $T \times T$ blocks, where T is the packing block size, should not contain any $T \times T$ block belonging to a previously placed patch. Such constraint makes it possible to determine the path that each $T \times T$ block belongs by analyzing the 2-D bounding boxes of all patches. To improve fitting chances, different patch orientations are also allowed. In the case where there is no empty space available for the next patch, the height H of the image is doubled, and the insertion of the current patch is evaluated again. After inserting all patches, the final height of all frames in the sequence is trimmed to the minimum needed value.

To improve the compression efficiency, one can exploit the temporal correlation between patches. For instance, patches with similar content could be placed in similar positions across time, generating a more temporally coherent video. Furthermore, the 3-D reconstruction information associated with corresponding patches may be similar, which could be exploited for more efficient data transmission. The corresponding patches can be identified by evaluating the amount of overlap between the patches' bounding boxes.

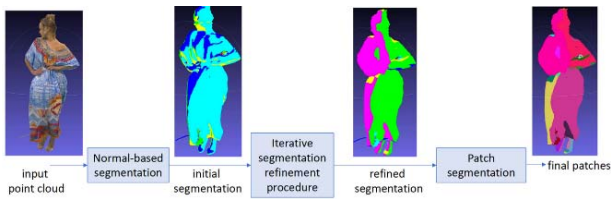


Fig. 9. ***Patch generation process.***

Fig. 10. ***Example of geometry (left), occupancy map (center), and attribute (e.g., texture) (right) images.***

3) Geometry, Occupancy Map, and Attribute Image Generation: Once the patch-packing procedure determines the projection of the 3-D patches onto the 2-D image canvas, the geometry image can be generated by inserting the depth values in the luminance channel of the image. However, a patch can have multiple points being projected onto the same 2-D pixel location. To handle such cases, the V-PCC standard allows the encoder to use up to 16 layers to store overlapping points. In particular, let us assume that we have a set of points $H(u, v)$ being projected to the same (u, v) location. V-PCC currently uses two layers. One layer, referred to as the near layer or layer 0, stores the point from $H(u, v)$ with the lowest depth value $D0$, and the other layer, referred to as layer 1, stores the point with the highest depth value within a user-defined interval $[D0, D0 + D]$. The user-defined interval size D represents the surface thickness and can be used to improve geometry coding and reconstruction.

V-PCC has the option to code layer 1 as either a differential layer from layer 0 or use its absolute value. Moreover, the layers can be coded in two separate video streams or temporally interleaved into a single stream. The interval size D can be used to improve depth reconstruction since the reconstructed values must lie within the predefined interval. Layer 1 can be dropped, in which case the decoder can generate the far layer from layer 0 and the interval size D. Furthermore, both frames can be subsampled and spatially interleaved to improve reconstruction quality and coding efficiency.

The mapping of geometry from 3-D to 2-D does not use all pixel positions in the geometry image, thereby leaving some empty spaces. A binary image, also known as occupancy map, indicates with value 1 the used positions, and value 0 the empty ones. The occupancy map can be compressed at a lower resolution, which is determined by a user-defined parameter, leading to an enhanced compression efficiency but may affect the geometry reconstruction. Fig. 10 shows an example of geometry, occupancy map, and attribute (e.g., texture) images.

B. MPEG V-PCC Evaluation

To evaluate the performances of MPEG V-PCC, we conducted several experiments using the TMs provided by the MPEG, with both the encoder and the decoder. Notably, better implementations of the encoder may be available in the future because the TMs are mainly designed to produce conformant bitstream and exclude many nonnormative tools, which can improve coding efficiency. However, the results presented in this section provide a good illustration of the MPEG V-PCC capabilities.

Fig. 11. ***Evaluation data set from [76] and [77].***

The evaluation data set comprises five sequences of dense point clouds (see Fig. 11). The sequences have numbers of points ranging from 800 thousand (k) to one million (M), with a frame rate of 30–50 frames/s. Each coordinate of the point positions is stored in 10-bit precision. The texture information is stored with an 8-bit precision for each color channel.

For the geometry metrics, MPEG retained two peak signal-to-noise ratio (PSNR)-like measures, denoted by metrics D1 and D2. Both metrics measure the geometric distortion of a decoded point cloud B compared with a reference point cloud A by averaging all distances between each point in point cloud A and its nearest neighbor in point cloud B. The main difference between D1 and D2 is that D1 uses a point-to-point distance, whereas D2 uses a point-to-plane distance. The D1 and D2 distances are put under the form of a PSNR score for deriving RD curves. To compute the quality of the attributes, the same nearest-neighbor mapping is used between point clouds A and B. The PSNR score is computed by considering the difference in attribute values for corresponding points. To globally compare RD performance of two codecs C1 and C2, MPEG uses BD rates [78] (see Fig. 12).

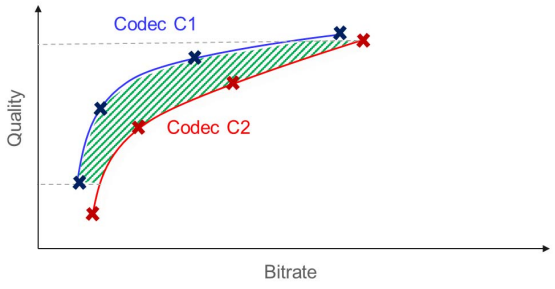


Fig. 12. ***Example of the RD metric denoted as BD rate.***

Fig. 13. ***Composition of the bitstream components for the longdress sequence (the other sequences have similar distribution).***

First, the RD curves for both codecs were computed by encoding the same sequence with different parameters. Each encoding configuration corresponds to a point in the RD curve. The horizontal and vertical axes correspond to bitrate and geometric/attribute quality (e.g., PSNR), respectively. The RD curve was created by interpolating between those points. The BD rate provides a global relative performance measure, defined as the average (or integral) over the area between the two curves (green shade) of the bitrate difference between the two codecs for the same quality.

The V-PCC codec uses several video codecs to encode different bitstreams. Besides, it encodes some metadata for representing the relations between the 3-D points and 2-D patches. The composition of a V-PCC bitstream is measured in different coding modes in terms of percentages of the total bitstream size; an example is illustrated in Fig. 13 for the *longdress* sequence and two coding modes denoted, respectively, as *random access* (RA) and *all intra* (AI). Notably, in the AI configuration, each frame is independently encoded as a static point cloud with no reference to other frames in the sequence. In the RA configuration, a hierarchical predictive structure is employed based on the used video codec to provide predictions between pictures, which roughly reduces the bitrate by a half.

Fig. 13 shows the contributions in the percentage of the total bitstream size of the V-PCC substreams corresponding to geometry, attribute, occupancy (all represented as 2-D videos), and atlas parameters (represented as metadata). The results showed that 95%–99% of the total bitstream was dedicated to video substreams, which were efficiently handled by the existing video codec infrastructure.

When V-PCC activity was initiated in 2017, the state-of-the-art point cloud codec was the one implemented in the PCL [73] based on octrees. For comparison, this codec [39] was used as a reference anchor method for benchmarking the performances of the approaches [79]. After the Call for Proposals (CfPs), eight different technologies were proposed by MPEG. The best among them—a projection-based method—was selected and transformed into TM v1. It can be considered as a prototype of the current V-PCC TM. During the development of the standard, 13 consecutive versions of TM were developed based on the previous version to achieve better performance. The optimization mainly concerns the refinement of 3-D-to-2-D projection, the improvement of patch packing and padding strategies, and data signaling. In this article, we evaluated the progress of the encoding efficiency for the sequences in the evaluation data set using V-PCC TM v1 as a benchmark and v11 as the one providing state-of-the-art performance. The results are illustrated in Fig. 14 for the AI mode.

Fig. 14 shows the RD curves of three codecs: the green curves correspond to V-PCC TM v11, named TMC2, and the orange curves correspond to V-PCC TM v1 initially submitted as a response to MPEG CfP, whereas the blue curves correspond to the anchor codec. As expected, TM v1 and TM v11 outperformed the anchor codec, and TM v11 outperformed TM v1. In Fig. 15, we illustrate a similar comparison but for the RA mode.

While the progress with respect to the anchor codec is easy to observe in Fig. 15, noticing the differences between TM v1 and TM v11 is nontrivial. Therefore, we illustrate the overall bitstream size reduction, integrated over the five encoding rates (BD rates), for each sequence and the two modes (AI and RA) in Table 1. This table should be interpreted as follows: considering a sequence, e.g., *redandblack*, the bitrate difference between the two TMs integrated over the five rating points was 28% and 58%, respectively, for the geometries in the AI and RA modes in

Table 1 Percentage of BD Rate Reduction for Geometry and Texture in AI and RA Modes

AI/RA	Geometry	Texture		
Sequence	**D1**	**Y**	**Cb**	**Cr**
Loot	45%/74%	18%/25%	8%/18%	8%/14%
Redandblack	28%/58%	14%/6%	11%/4%	46%/40%
Soldier	1%/59%	23%/32%	12%/21%	10%/10%
Queen	52%/81%	29%/45%	59%/69%	70%/74%
Longdress	15%/52%	22%/25%	32%/39%	44%/47%
Average	28%/65%	21%/27%	24%/30%	36%/39%

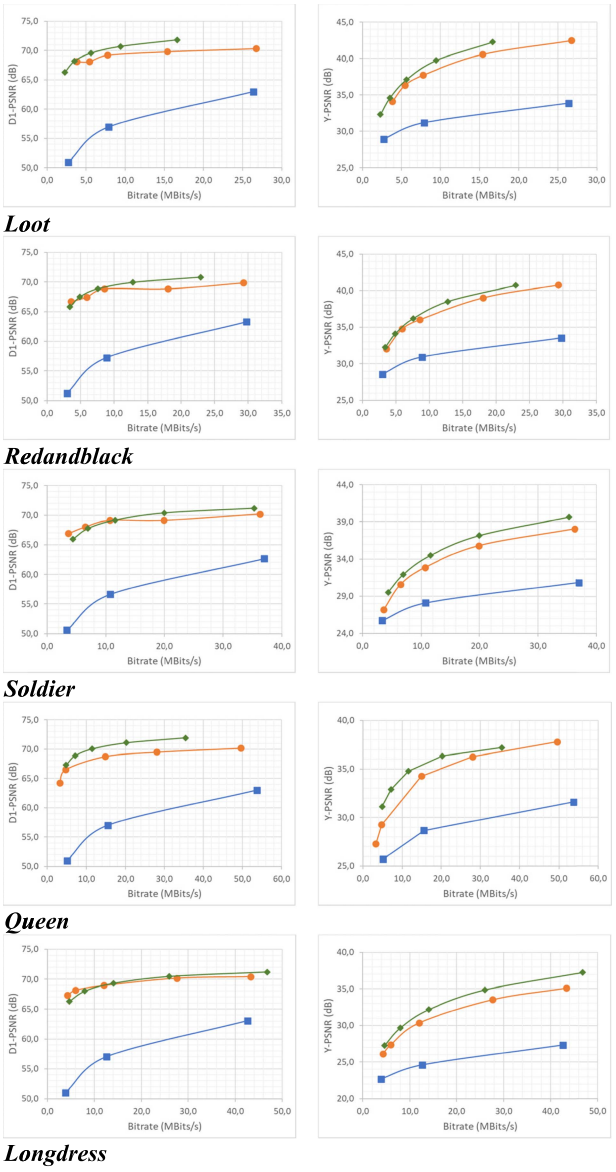


Fig. 14. ***Geometry (left) and attribute (right) PSNRs comparison between anchors (blue), TM v1 (orange), and TM v11 (green) for the AI mode.***

favor of TM v11. The same interpretation was valid for the texture coding: 14% (6%), 11% (4%), and 46% (40%) for each color channel in the AI mode (RA mode).

Notably, for all metrics (geometry and texture), global improvements were obtained from TM v1 to TM v11 for all sequences. For the geometry, the improvements were more significant for the RA condition than for AI. For the texture metrics, the behavior is more heterogeneous.

V-PCC relies on video codecs; however, the 2-D frames obtained by projecting point clouds are not similar to the ones that video codecs usually handle. Since the clustering operation (and, therefore, the computation of the 2-D patches) is performed for each frame individually (notably, this is the TM behavior, and the clustering is nonnormative), patches may not preserve the same index. Therefore, the 2-D patches may not preserve the same location on the 2-D atlas. Fig. 16 illustrates two consecutive 3-D point cloud frames for the *longdress* sequence (frames 12 and 13) and the corresponding texture maps obtained by projecting the point clouds.

The first observation concerns the noisy structure of the 2-D frames due to the patch-packing operation. We also observe some temporal inconsistency between 3-D and 2-D spaces. In particular, small 3-D motions can lead to significant variations in the 2-D positioning of patches. We evaluated the performances of the TM v11 in terms of how motion was addressed by comparing the results of the

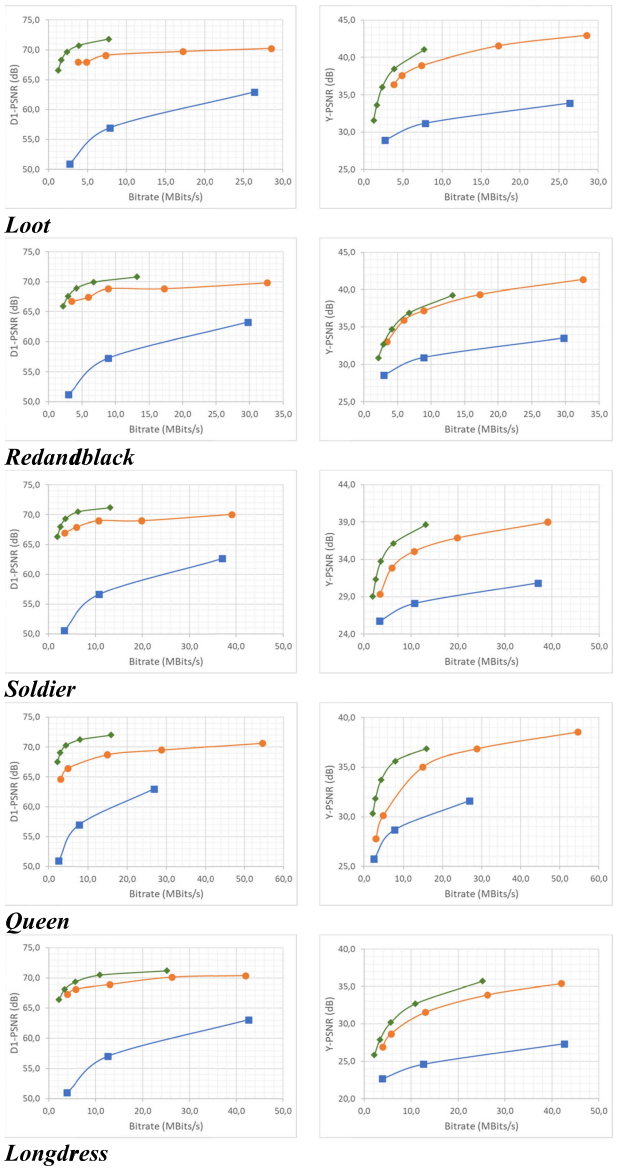


Fig. 15. ***Geometry (left) and attribute (right) PSNRs comparison between anchors (blue), TM v1 (orange), and TM v11 (green) for the RA condition.***

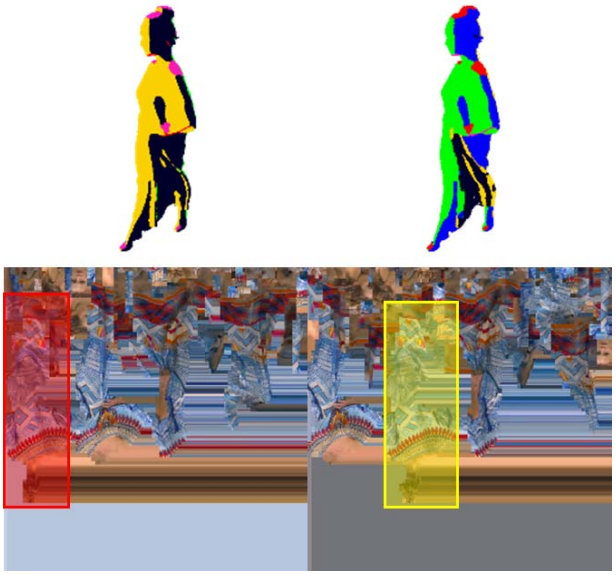


Fig. 16. ***Two consecutive frames in 3-D and the corresponding texture maps in 2-D.***

RA versus AI modes. The obtained results are presented in Fig. 17.

As expected, the RA mode globally performed better. However, the difference was not as significant as in the case of natural video encoders. The corresponding BD rates are summarized in Table 2.

The reduction in average was between 40% and 50%, which was less than the value of 90% that video codecs can usually obtain using the RA mode on the traditional video content. These results showed that there is still room for improving the V-PCC intercoding RD performance.

Apart from normative tools required in the standard specifications, it is possible to employ additional nonnormative tools to improve the compression performances. Two examples are shown here illustrating the potential gain: 1) 3-D motion estimation [80] and 2) occupancy-based RD [81] in Tables 3 and 4, respectively.

The motion estimation was performed in 3-D space by searching the nearest position of a point. Then, a motion vector was generated in 2-D to guide the video codec in performing motion prediction for the attributes.

Table 2 BD Rate Reduction When RA Mode Is Used Instead of AI

RA vs AI	Geometry	Texture		
Sequence	**D1**	**Y**	**Cb**	**Cr**
Loot	56%	45%	56%	54%
Redandblack	43%	22%	33%	18%
Soldier	68%	61%	69%	68%
Queen	42%	48%	58%	55%
Longdress	58%	32%	41%	38%
Average	53%	42%	51%	47%

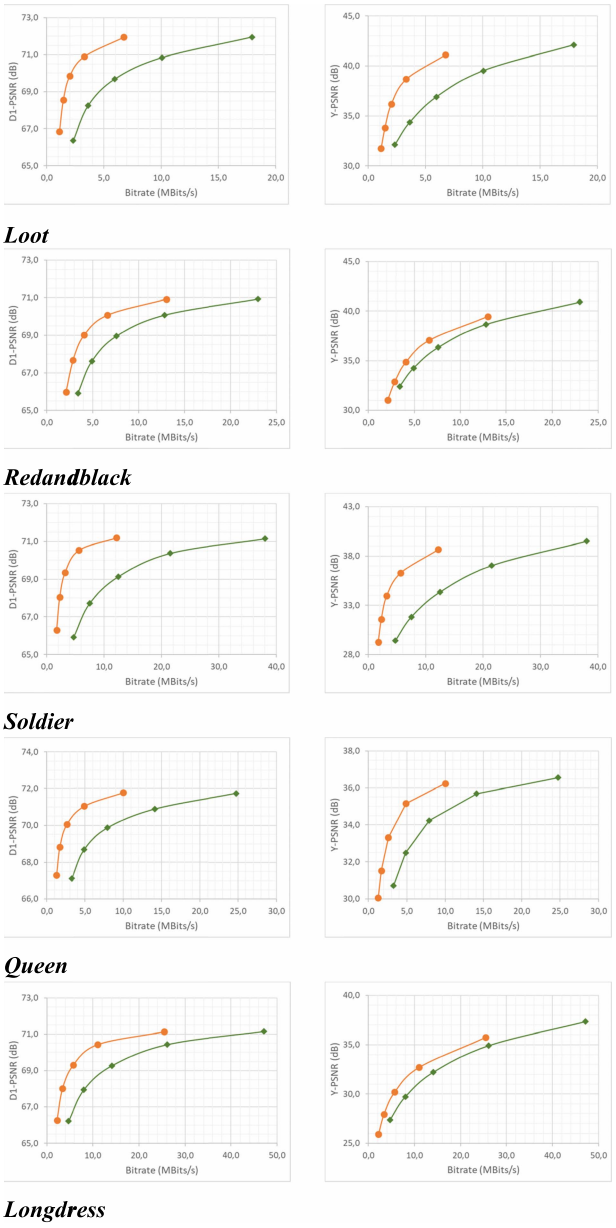


Fig. 17. ***Geometry (left) and attribute (right) PSNRs comparison between AI (green) and RA (orange) for TM v11.***

For encoding dynamic content, a significant performance improvement can be expected with this tool.

The main idea of this optimization is to ignore padded pixels while computing distortion in the video RDO block.

Other than the potential gain that we may obtain from the additional nonnormative tools, V-PCC is future-proof and could automatically leverage improvements introduced by future video compression standards.

The MPEG V-PCC was designed for efficient coding of dense dynamic point clouds. To cover the numerous use cases for point clouds, MPEG developed, in parallel, a more generic technique, exploiting a geometry-based

Table 3 BD Rate Reduction When Optimization via 3-D Motion Estimation Is Applied

	Geometry	Texture		
Sequence	**D1**	**Y**	**Cb**	**Cr**
Loot	6%	9%	17%	17%
Redandblack	6%	9%	13%	9%
Soldier	15%	21%	30%	31%
Queen	6%	7%	5%	6%
Longdress	6%	7%	9%	9%
Average	8%	11%	15%	14%

approach. The so-called G-PCC standard is presented in Section IV.

IV. MPEG G-PCC

G-PCC is dedicated to generic point clouds, covering both sparse and dense, as well as static and dynamic objects and scenes. In this section, we provide a brief overview of G-PCC, considering the key coding tools.

A. MPEG G-PCC Methodology

A general block diagram of the G-PCC encoder is illustrated in Fig. 18. The G-PCC encoding is a sequential operation, which first processes geometry and then the attributes. This behavior, useful for lossy coding, is directed by the need to first map the original attributes on the reconstructed (encoded and decoded) geometry and the encoding of these mapped attributes instead of the original ones.

1) Coordinate Transformation and Voxelization: The input point coordinates are represented in their coordinate system provided by the acquisition system and are usually expressed as floating numbers. In such cases, before G-PCC encoding, a conversion to integer values is performed. To normalize the original point cloud, the input world coordinate system is converted to the so-called frame coordinate system (FCS), represented by N bits for each of the three dimensions. Therefore, the bounding box of the original point cloud is geometrically aligned with the FCS while keeping the aspect ratio of the bounding box.

Table 4 BD Rate Reduction When Occupancy-Based RDO Is Applied

	Geometry	Texture		
Sequence	**D1**	**Y**	**Cb**	**Cr**
Loot	23%	24%	21%	21%
Redandblack	11%	13%	11%	13%
Soldier	23%	27%	21%	22%
Queen	21%	24%	19%	20%
Longdress	9%	11%	10%	10%
Average	17%	20%	17%	17%

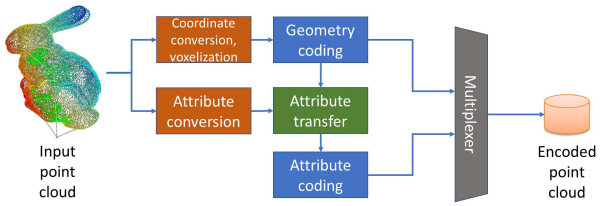


Fig. 18. *G-PCC encoder diagram.*

Thus, a voxelization is needed for the G-PCC encoder processes integer values. The quantization process is applied to the points represented in FCS, converting the input bit depth from N bit (per dimension) to M bit, where $M < N$. The value of N depends on the precision of the acquisition system, and the value of M depends on the desired precision of the reconstruction. Then, each point position can be considered belonging to a cube, with coordinate values from $(0, 0, 0)$ to $(2^M, 2^M, 2^M)$. Once the quantization process is applied, some new points are likely duplicated (they have the same quantized coordinate values). Such points are called duplicate points, and they may have distinct attribute values (because the original unquantized points have distinct attribute values). Different strategies can be used to compensate for the attribute accuracy loss due to geometry quantization, and a recoloring process is included in the encoding scheme.

2) Geometry Coding: G-PCC includes two geometry encoding modes. The first one uses an octree decomposition of the 3-D space and can potentially be complemented by a tool called trisoup used to terminate the octree decomposition prematurely. The second is a simpler predictive mode when the points are sequentially traversed, and the value of the current point is predicted from the previous (up to three) ones. This second mode is mainly appropriate for objects acquired by lasers when the acquisition order can be used to predict the values of point coordinates. The overall schema of these geometry encoding modes is shown in Fig. 19.

In the following, we provide details for each of these blocs.

a) Octree coding: After voxelization, the voxels in a cube may be empty or occupied by the points in a

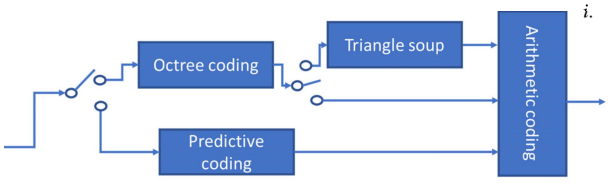


Fig. 19. *G-PCC geometry encoder diagram.*

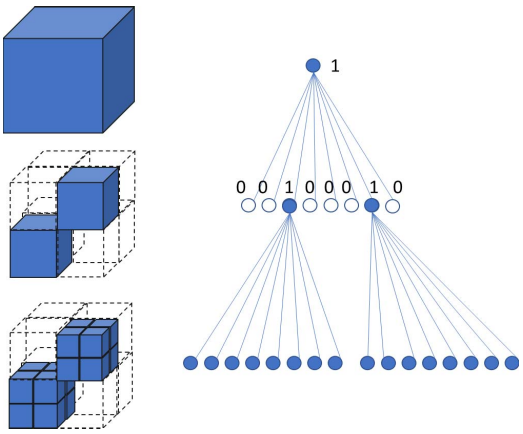


Fig. 20. ***Octree decomposition of a cuboid.***

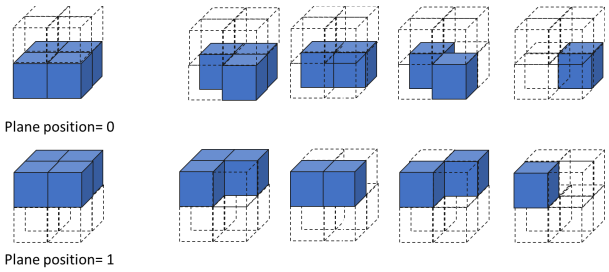


Fig. 21. ***Example of planar mode where occupied nodes are in the same plane.***

point cloud. Then, an octree decomposition is performed. The occupied voxels correspond to the quantized point locations.

As illustrated in Fig. 20, a cube can be decomposed into eight subcubes of equivalent size, which can be done iteratively until the size of each subcube is the same as a voxel. A subcube that does not contain any occupied voxel will not be decomposed any further. Every node of an octree can be represented by either 0 (empty) or 1 (occupied). At each node decomposition in an octree, each of the eight child nodes is represented by a binary value denoting its occupancy, which leads to an 8-bit pattern. Then, the octree structure is described in a breadth-first search manner; the nodes at the lower level are encoded earlier than those at the higher level. Next, the 8-bit patterns of child nodes are entropy coded using arithmetic coding.

In addition to the traditional octree decomposition, G-PCC includes specific encoding tools to optimize the octree analysis: direct coding mode (DCM), planar mode, QT/BT, and angular/azimuthal mode.

b) DCM: The octree decomposition works effectively when there are many points in the given cube. When an object presents isolated points (which occurs in the case of sparse objects), the 8-bit pattern used to describe the child level is more expensive than simply encoding the voxel containing the isolated point directly—a mechanism called DCM.

Therefore, G-PCC exploits the correlation of parent-level node information to determine the eligibility of DCM. Specifically, there are two conditions to determine the eligibility of DCM: 1) the parent node is the only occupied child, and the grandparent is either the only occupied child or one of the two occupied children and 2) the parent node is the only occupied child, and the six face-neighboring cubes to the parent node are unoccupied.

c) Planar mode and QT/BT: In the case of sparse point clouds, it is suboptimal to predict the coordinates from the neighbors. The planar mode exploits the fact that points are coplanar (e.g., ground and building walls), even when they are far from each other. When such a case is identified, the number of possible cases for prediction reduces, which contributes to further compression (see Fig. 21).

In the planar mode, the occupancy pattern of an octree is compacted by the correlation from the neighborhood patterns. However, it is possible to express the distribution not by octrees but by QT or BT, making it possible to reduce the occupancy code to 4 or 2 bits. This signaling mode can be employed for the partial areas of the point cloud or from the specific depth level of an octree. Using QT or BT only for the simple pattern leaf nodes, the occupancy code can be more compact without sacrificing accuracy (from 8 to 4 and 2 bits, respectively; see Fig. 22).

d) Triangle soup (Trisoup) coding: Trisoup-based geometry compression (TGC) is an additional geometry compression tool that can be used complementary to the octree decomposition. In an octree, an occupied leaf node represents the final resolution of point coordinates, whereas, in TGC, an occupied leaf node represents a 3-D cube that may contain one or more points inside. Then, each leaf-level occupied 3-D cube is represented as surfaces of triangle strips, where each triangle can be formed with vertices on the edges of the 3-D cube. This triangle information is encoded instead of point coordinates belonging to the 3-D cube. At the decoder side, an appropriate number of point coordinates is generated from the triangle surfaces. Depending on the formation of input point coordinates, there can be 3–12 identified vertices. With these vertices, there can be one to ten triangles in a 3-D cube.

e) Predictive coding: Unlike the octree decomposition, which recursively divides the bounding box, predictive

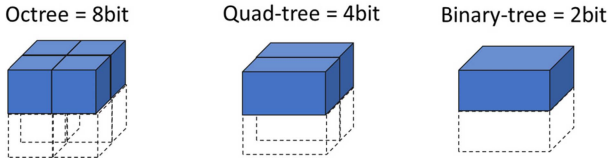


Fig. 22. ***Encoding with QT/BT reduces the occupancy code.***

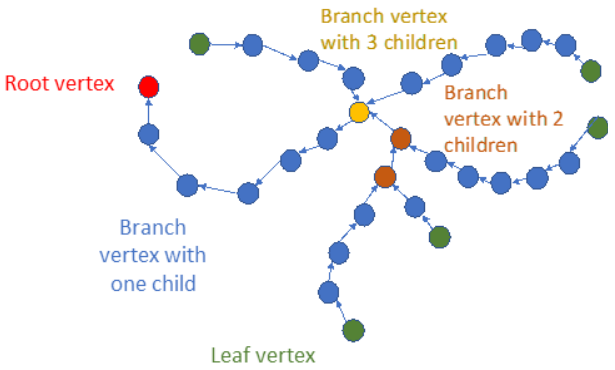


Fig. 23. ***Example of a geometry prediction mode. A prediction tree can branch into up to three subtrees at a point.***

geometry coding is a point-by-point coding method. This method is intended to cover the low-delay use cases, for example, to handle, in real time, the input from 3-D sensors. In this mode, the encoder orders a set of points as a prediction tree (see Fig. 23).

In the example illustrated in Fig. 23, a tree starts from the red point and then branches at the yellow point into three. The encoder performances depend on the construction of the trees, and *a priori* information, such as the scanning order, can be used.

f) Arithmetic coding: The final step of the geometry encoding is arithmetic encoding, contextualized with respect to the geometry encoding mode (octree, trisoup, or predictive). For the octree mode, several improvements are made in G-PCC. To maximize the gain of arithmetic coding, context-based probability modeling is used. There are two different strategies to entropy code the 8-bit pattern: bitwise and bytewise context-based arithmetic coding. This context-based arithmetic coding is bypassed for some specific cases (such as DCM) when data are statistically unsuitable for entropy coding.

There are quite a few context-based models to efficiently encode 8-bit patterns in G-PCC. As illustrated in Fig. 24, up to six neighbors of the parent node can be occupied. Therefore, 64 (i.e., 26) different neighbor configurations (NCs) are possible.

When NC is zero, it means that the parent node does not have any occupied neighboring node. If the parent node has only one occupied child, it is subject to be encoded with DCM. Otherwise, the 8-bit pattern is encoded with the context of NC $= 0$.

In addition to NC, a further context refinement is performed by adding the previously encoded bits in the 8-bit pattern of the same node, which increases the number of contexts at 128 (2^8) multiplied by 64. G-PCC reduces the 128 possible combinations to ten invariant cases using symmetries and rotations.

An additional improvement of compression efficiency is obtained by exploiting encoded child nodes that are neighboring the current node (see Fig. 25). By examining the already-encoded child nodes, it is possible to accurately determine whether an occupied neighboring block is truly occupied (and connected to the current node) (see Fig. 26).

3) Geometry Reconstruction: The geometry reconstruction process in Fig. 18(a) decodes and reconstructs point coordinates; the reconstruction does not mean a full decoding process since it does not contain any arithmetic decoding. The main reason for geometry reconstruction is for recalculating attribute data according to the decoded geometry information. If the compression of position information is lossless, the geometry reconstruction and attribute transfer processes are unnecessary, and parallel (geometry and attribute) encoding can be conducted.

4) Attribute Conversion, Recoloring, and Coding: In a point cloud, there may be many different attribute types (e.g., color, normal, reflectance, and laser ID). However, the current G-PCC supports only color and reflectance

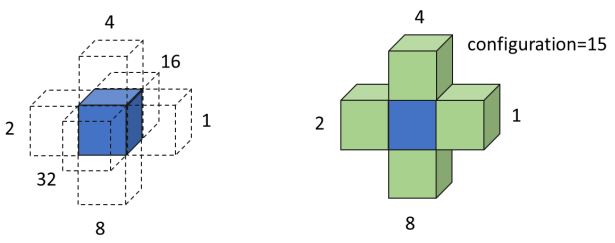


Fig. 24. ***NC (left) and example of NC $= 15$ (right).***

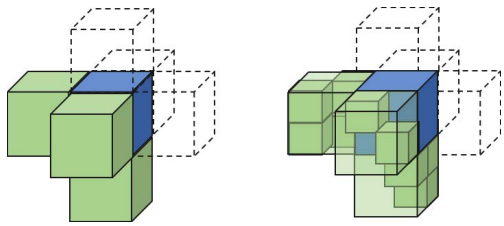


Fig. 25. ***Exploitation of already-encoded child nodes for further compression efficiency.***

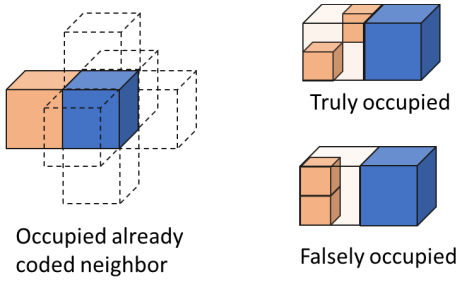


Fig. 26. ***Determination of truly or falsely occupied neighbor.***

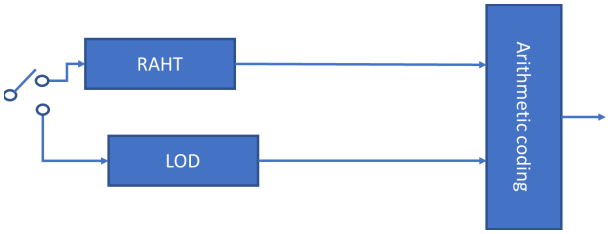


Fig. 27. *G-PCC attribute encoder diagram.*

data. The color attributes in G-PCC are represented in the YCbCr color space. A color space conversion is needed if the original acquisition color space is different as mentioned in standard conversion process defined in related standards, such as ITU-R BT.709 [82]. For encoding the reflectance data, the same encoding path of color data is used; the only difference is that the dimension of reflectance is one instead of three.

The attribute conversion process recomputes the attributes based on the reconstruction geometry when the compression of geometry is lossy. In this case, most point locations after encoding are different from those of the original input point cloud. Therefore, the values of attributes, such as color and reflectance, need to be recomputed. There are many possible ways to *recolor* points; a simple one is to use the weighted average using the nearest points from the input point cloud. Since this process is performed only at the encoder and is outside the standardization scope, there is room for further improvement. Moreover, good recoloring directly influences the performance of the encoder.

There are two methods to encode attributes in G-PCC—RAHT and LOD—each has some advantages and limitations with respect to the point cloud density. The overall schema of these encoding modes for the attribute is presented in Fig. 27.

In the following, we provide details for each of these blocs.

a) RAHT: RAHT [20] is a 3-D transform designed for data that may contain empty locations in a 3-D space. Point coordinates represented by an octree structure are transformed by RAHT. RAHT provides a hierarchical spatial transform similar to wavelet coding in image and video compression, which provides data compaction to the lowest frequency component among the geometry signals. RAHT becomes a 3-D version of the Haar transform if all points in a given 3-D cube are occupied. RAHT is performed hierarchically such that the current level transformation is performed at the lower level (or child level) transformed coefficients.

b) LOD generation: The LOD generation process aims at providing spatial scalability to G-PCC. The points are separated into layers (a Euclidian distance can be used; however, how LODs are generated is not in the scope of the standard). The attributes associated with the points are predicted from points encoded at the same layer or a lower layer. An example of LOD generation is illustrated in Fig. 28. The LOD generation process has to generate a different encoding order of attributes from that of geometry. For example, a distance-based clustering is performed in Fig. 28. The LOD generation process is an encoder issue, which means that there can be different (and even more efficient) strategies to group points into layers. Once the order and grouping of points in layers are determined, the attribute values at each point are predicted, and their differences are entropy coded using arithmetic coding.

The prediction process determines a prediction value of the current point from the nearest already-encoded points. Searching the nearest points increases the encoder computational complexity and compression efficiency.

If there are k attribute values $(a_i, i = 0, \ldots, k-1)$, they are predicted by a linear interpolation process based on the distances to the nearest neighbors of the current point i.

c) Quantization and arithmetic coding: The RAHT transformed coefficients and residual prediction values are uniformly quantized. Quantization is controlled by a global parameter denoted by QP, which is used just like in the previous MPEG AVC/HEVC/VVC standards. Finally, the quantized coefficients are entropy coded with run-length coding and binary arithmetic coding.

B. G-PCC Evaluation

The main difference between G-PCC and V-PCC is that G-PCC is designed to handle various point cloud categories, whereas V-PCC is optimized for dense surface-like point clouds. These point clouds are usually uniformly voxelized, where the coordinates are quantized into integer values and contain a low level of noise with limited bit-depth positions. The data set for G-PCC evaluation includes individual frames from some of the sequences used for V-PCC evaluation and other categories of point clouds, which are described in the following.

The first category is the one with static human models (single frames extracted from the sequences introduced in Section III-B). In this case, the point clouds are dense, smooth, and uniformly sampled. The bit depth for

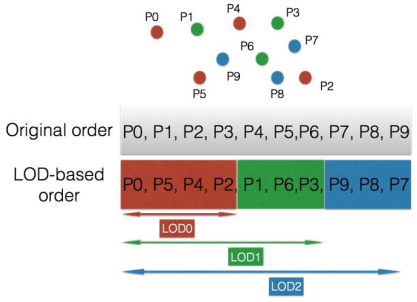


Fig. 28. *Example of the LOD generation process.*

Facade 9 [83] with around 1.6 M points

House without a roof [83] with around 5 M points

Palazzo Carignano [84] with around 4.2 M points

Arco Valentino [84] with around 1.5 M points

Fig. 29. ***Objects from the "façade & buildings" category.***

positions varies between 10 and 12. The number of points is between 800k and 3M points.

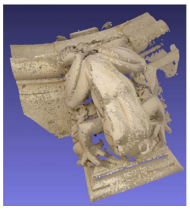
The second category is called "façade & buildings, as shown in Fig. 29." The density of the point clouds varies from sparse to dense with a medium to a high level of noise. The bit depth for such content can increase from 12 to 20 bits per coordinate. The number of points can reach 20M points, and the textures are 8 bits per color.

The third category, named "objects," contains various archeological, artistic, and ordinary objects. The point cloud density varies from sparse to dense, with a low to a high level of noise. Bit depth for positions varies from 11 to

Egyptian mask [83] with around 0.2 M points

Statue Klimt [83] with around 0.5 M points

Shiva [83] with around 1 M points



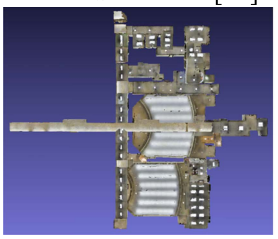
Frog [83] with around 3.6 M points

Head [83] with around 14 M points

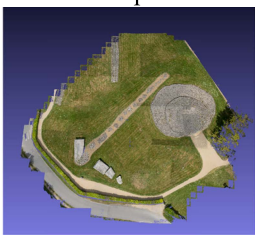
Unicorn [85] with around 63 M points

Fig. 30. ***Objects from the "objects" category.***

Stanford 2 [86] with around 47 M points



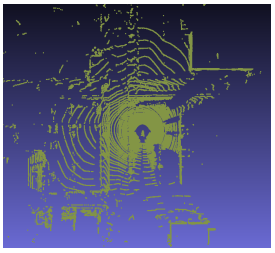
Stanford 4 [86] with around 43 M points



Landscape 14 [83] with around 72 M points

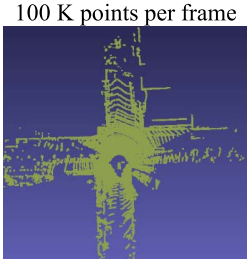
Fig. 31. ***Objects from the "landscapes" category.***

20 bits. The number of points can be as low as 300k and goes up to 64M points. Some of these objects are depicted in Fig. 30.

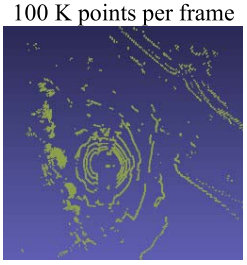


Ford 01 [87] with around 100 K points per frame

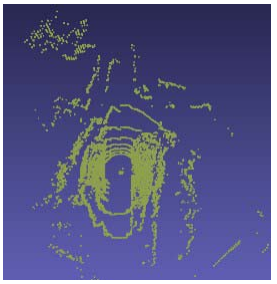
Ford 02 [87] with around 100 K points per frame



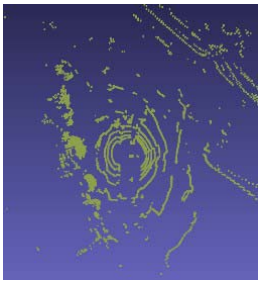
Ford 03 [87] with around 32 K points per frame



Junction approach [88] with around 32 K points per frame



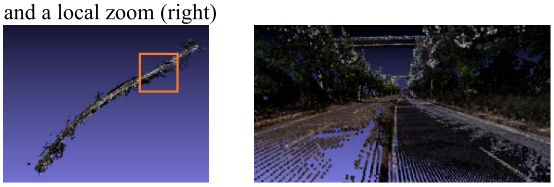
Motorway join [88] with around 32 K points per frame



Navigating bends [88] with around 32 K points per frame

Fig. 32. ***Objects from the "frame-based LiDAR" category.***

City tunnel [89]: entire scene (left) with around 19 M points and a local zoom (right)



Overpass [89]: entire scene (left) with around 5 M points and a local zoom (right)

Fig. 33. ***Objects from the "fused LiDAR" category.***

The fourth category is called "landscapes" (see Fig. 31). It contains large point clouds with 44–72M points. The bit depth for positions varies from 14 to 20 bits per coordinate.

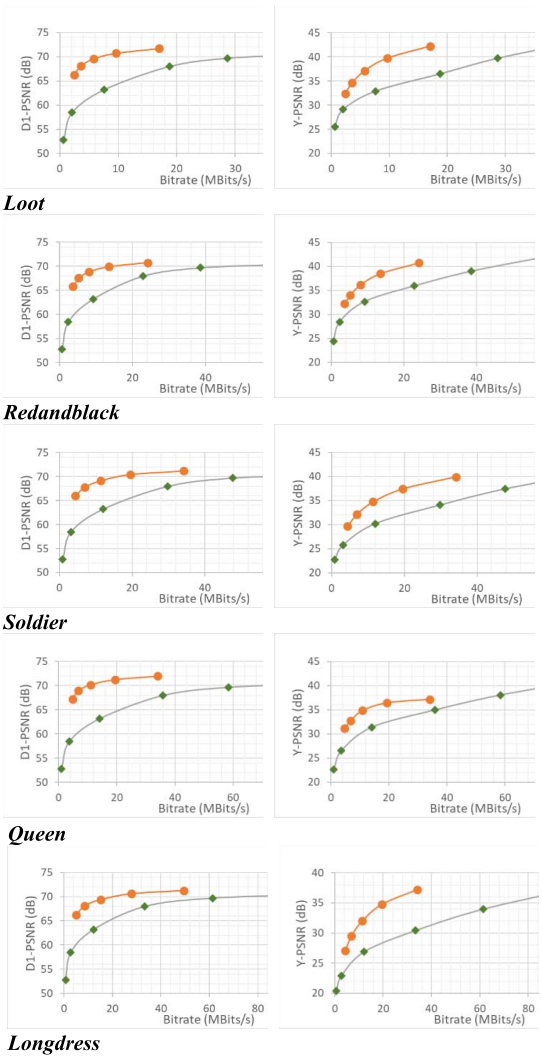
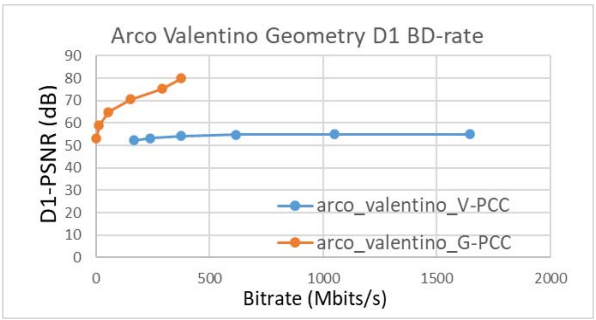
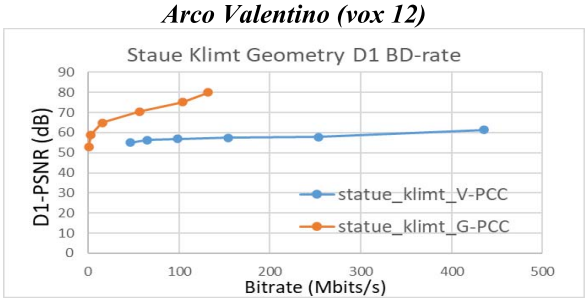


Fig. 34. ***Geometry (left) and attribute (right) PSNRs comparison between V-PCC (orange) and G-PCC (green) for individual frames in the human category.***



Arco Valentino (vox 12)



Klimt statue (vox 12)

Fig. 35. ***Geometry PSNR comparison between G-PCC (orange) and V-PCC (blue) for the two categories.***

The density varies from sparse to dense. The point clouds are usually tiled and encoded by smaller chunks to allow efficient navigation and RA.

The fifth category, named "frame-based LiDAR," contains point clouds captured by LiDAR scanners mounted on moving vehicles. The point clouds in this case are sparse, with a low to medium level of noise (see Fig. 32). The sampling is highly irregular and depends on the acquisition process implemented by the LiDAR scanner. The positions are quantized into 18 bits per coordinate. The objects include reflectance information associated with the point cloud. The number of points per frame varies between 27k and 84k. The frame rate is between 5 and 10 frames/s.

The last category is called "fused LiDAR," containing large point clouds with 5–20M points, which cover an extended geographic area (see Fig. 33). The point clouds are usually generated by fusing multiple frame-based LiDAR acquisitions into a single point cloud. The point clouds are sparse with a low to medium level of noise. The sampling is highly irregular. The positions are encoded with 20-bit precision. The point clouds have both texture and reflectance information associated with their points.

The diversity of encoding tools in G-PCC allows addressing different use cases with different requirements.

The first evaluation concerns dense and static objects, such as the human point clouds of the first category. The results obtained are presented in Fig. 34. They confirmed that, for this category, the V-PCC approach (as implemented in TM v11) outperformed G-PCC, for all encoding rates.

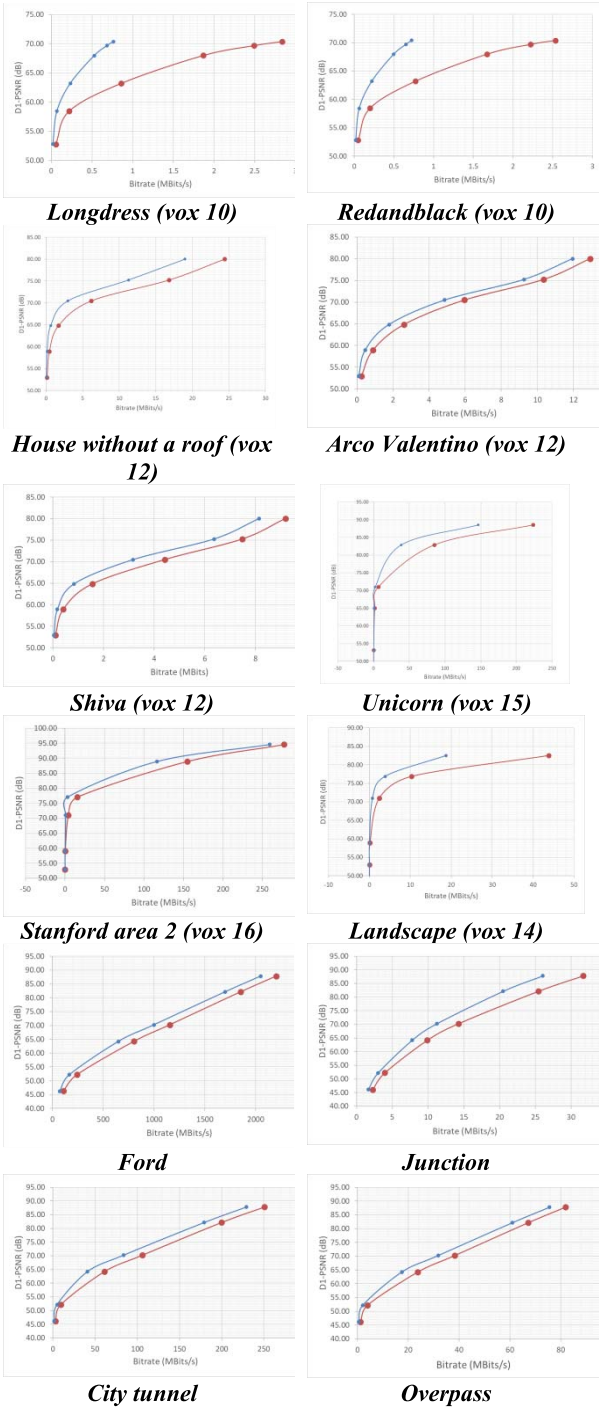


Fig. 36. ***Geometry PSNR comparison between G-PCC (blue) and Draco (red) for the five categories.***

The second evaluation concerns the sparse objects from the second to fifth categories. Notably, for all cases, the number of missed points (3-D points projected in the same 2-D pixel) was much larger than for the first category objects, V-PCC creating more patches, and a bigger raw patch. For example, *House without a roof* had 4.8M points, where 2.9M (60%) points were classified as missed points,

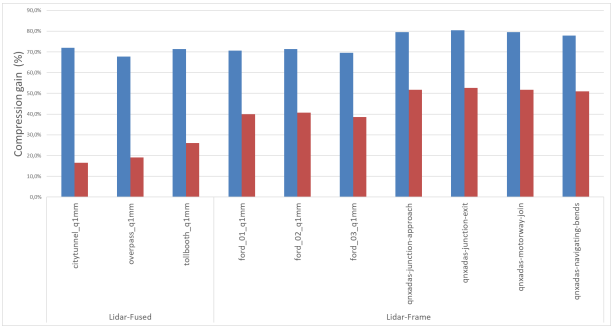


Fig. 37. ***G-PCC TMC13-v11 versus raw (in blue) and versus Draco (in red) for lossless geometry compression.***

and 430 2-D patches were generated. An even smaller object, such as *Shiva* (1 M points), had 0.5M (50%) points classified as missed points and 738 patches. This should be compared with only 0.4% of miss points for *longdress* and 61 patches.

Despite this observation that V-PCC (and, therefore, 3-D-to-2-D projection methods) was inappropriate for the objects from the second to fifth categories, we generated the RDO curves for some objects (see Fig. 35).

Since G-PCC had a good coding performance for sparse objects from the second to fifth categories, we compared these performances with the ones of Draco, an open-source graphics codec provided by Google [90]. In these experiments, G-PCC is set in the octree mode for the geometry and *predlift* for attributes coding. The results are illustrated in Fig. 36.

Notably, G-PCC outperformed Draco for all objects and in all coding rates.

In automotive and mapping applications, it is sometimes important to preserve and use the data acquired from scanners in a lossless mode. We compared the G-PCC lossless performance with respect to the raw (uncompressed) data representation and the Draco implementation. The results are presented in Figs. 37 and 38. Notably, an average of 61% reduction with respect to raw data for geometry was obtained (38% reduction with respect to Draco), and

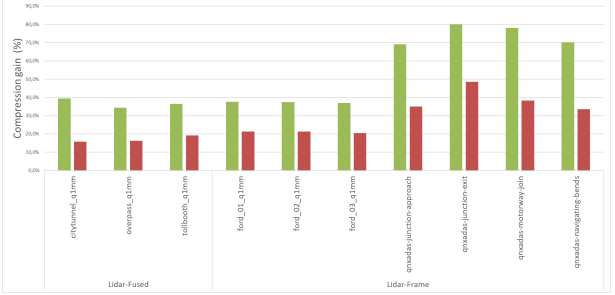


Fig. 38. ***G-PCC TMC13-v11 versus raw (in green) and versus Draco (in red) for lossless reflectance compression.***

an average of 73% reduction with respect to raw data for reflectance was obtained (27% reduction with respect to Draco).

V. CONCLUSION

In this article, we addressed a research hotspot—the compression of point clouds. With recent hardware and software advances of capturing systems, sparse and dense dynamic point clouds can be acquired with various features to satisfy the requirements of different use cases. For example, sparse point clouds with high precision can provide accurate navigation for autonomous driving; the dense point clouds can offer an immersive experience with high-fidelity content. However, with the increasing demand for point clouds with higher precision and density, compression becomes increasingly paramount inside the content chain. The research community started addressing the topic of PCC in 2001, and we reviewed, in this article, the main approaches published in the literature. Identifying the potential of PCC in innovative applications, MPEG produced two standards for representing compressed point clouds based on two prominent approaches: 1) a 3-D-to-2-D projection-based approach, followed by video coding (V-PCC) and 2) a purely geometric octree-based approach (G-PCC). We also described, in detail, the two MPEG approaches with experimental evaluations of their compression performance. Remarkably, V-PCC is optimized for dense surface-like dynamic point clouds, with uniform sampling and a low level of noise. Moreover, G-PCC is useful for more diverse use cases, designed to handle various point clouds, ranging from human models to world-scale city maps. We showed that the main advantage of V-PCC is the employment of efficient video technologies. It is also shown that there is still room to further improve the coding efficiency of both G-PCC and V-PCC by developing nonnormative tools. ■

Acknowledgment

The authors would like to express their gratitude to all moving picture experts group (MPEG) point cloud compression (PCC) experts for their participation and hard work in developing and documenting video-based PCC (V-PCC) and geometry-based PCC (G-PCC) standards. Some materials (text and images) created during the standard development were used in this article; they would like to thank all the original authors of this material: Sebastian Schwarz, Madhukar Budagavi, Philip A. Chou, Sébastien Lasserre, Khaled Mammou, Ohji Nakagami, Ali Tabatabai, and Alexis M. Tourapis.

REFERENCES

[1] S. Gumhold, Z. Kami, M. Isenburg, and H.-P. Seidel, "Predictive point-cloud compression," in *Proc. ACM SIGGRAPH Sketches (SIGGRAPH)*, 2005, p. 137, doi: 10.1145/1187112.1187277.

[2] B. Merry *et al.*, "Compression of dense and regular point clouds," in *Proc. 4th Int. Conf. Comput. Graph., Virtual Reality, Vis. Interact. Afr.*, 2006, pp. 15–20, doi: 10.1145/1108590.1108593.

[3] P.-M. Gandoin and O. Devillers, "Progressive lossless compression of arbitrary simplicial complexes," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 372–379, Jul. 2002, doi: 10.1145/566654.566591.

[4] M. Botsch, "Efficient high quality rendering of point sampled geometry," in *Proc. 13th Eurograph. Workshop Rendering*, 2002, pp. 53–64.

[5] J. Peng and C.-C. J. Kuo, "Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 609–616, 2005, doi: 10.1145/1073204.1073237.

[6] Y. Shao, "Attribute compression of 3D point clouds using Laplacian sparsity optimized graph transform," in *Proc. VCIP*, 2018, pp. 1–4, doi: 10.1109/VCIP.2017.8305131.

[7] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. SPBG*, 2006, pp. 111–120, doi: 10.2312/SPBG/SPBG06/111-120.

[8] D. C. Garcia and R. L. de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 1807–1811, doi: 10.1109/ICIP.2018.8451802.

[9] D. Viejo and M. Cazorla, "A robust and fast method for 6DoF motion estimation from generalized 3D data," *Auto. Robots*, vol. 36, no. 4, pp. 295–308, Apr. 2014, doi: 10.1007/s10514-013-9354-z.

[10] Y. Huang *et al.*, "Octree-based progressive geometry coding of point clouds," Eurographics Symp. Point-Based Graph, 2006, doi: 10.2312/SPBG/SPBG06/103-110.

[11] Y. Huang, J. Peng, C.-C.-J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 2, pp. 440–453, Mar. 2008, doi: 10.1109/TVCG.2007.70441.

[12] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980, doi: 10.1109/TCOM.1980.1094577.

[13] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd ed. New York, NY, USA: Academic, 2006.

[14] Y. Fan, "Point cloud compression based on hierarchical point clustering," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, 2013, pp. 1–7, doi: 10.1109/APSIPA.2013.6694334.

[15] M. Waschbüsch *et al.*, "Progressive compression of point-sampled models," in *Proc. Eurograph. Symp. Point-Based Graph*, 2004, pp. 95–102.

[16] B. Kathariya, L. Li, Z. Li, J. Alvarez, and J. Chen, "Scalable point cloud geometry coding with binary tree embedded quadtree," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6, doi: 10.1109/ICME.2018.8486481.

[17] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3235–3239, doi: 10.1109/ICIP.2015.7351401.

[18] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017, doi: 10.1109/TIP.2017.2707807.

[19] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 2066–2070, doi: 10.1109/ICIP.2014.7025414.

[20] R. L. de Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016, doi: 10.1109/TIP.2016.2575005.

[21] R. L. de Queiroz and P. A. Chou, "Transform coding for point clouds using a Gaussian process model," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3507–3517, Jul. 2017, doi: 10.1109/TIP.2017.2699922.

[22] *Labs, 8i Voxelized Full Bodies, Version 2—A Voxelized Point Cloud Dataset*, Standard ISO/IEC JTC1/SC29/WG11 m40059 ISO/IEC JTC1/SC29/WG1 M74006, 2017.

[23] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," *Proc. Symp. Point-Based Graph*, vol. 2004, pp. 103–112, doi: 10.2312/SPBG/SPBG04/103-112.

[24] M. Pauly and M. Gross, "Spectral processing of point-sampled geometry," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2001, pp. 379–386, doi: 10.1145/383259.383301.

[25] P. V. Sander *et al.*, "Multi-chart geometry images," in *Proc. SGP Eurograph./ACM SIGGRAPH Symp. Geom. Process.*, vol. 255, 2003, pp. 146–155.

[26] J. E. Fowler, "Shape-adaptive tarp coding," in *Proc. Int. Conf. Image Process.*, 2003, p. 621, doi: 10.1109/icip.2003.1247038.

[27] K. Zhang, W. Zhu, and Y. Xu, "Hierarchical segmentation based point cloud attribute compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 3131–3135, doi: 10.1109/ICASSP.2018.8461644.

[28] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002, doi: 10.1109/34.1000236.

[29] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, vol. 1, Oct. 2003, pp. 456–463, doi: 10.1109/iccv.2003.1238382.

[30] L. Wang, "Point-cloud compression using data independent method—A 3D discrete cosine transform approach," in *Proc. ICIA*, 2017, pp. 1–6, doi: 10.1109/ICInfA.2017.8078873.

[31] R. A. Cohen, "Point cloud attribute compression using 3-D intra prediction and shape-adaptive transforms," in *Proc. (DCC)*, 2016, pp. 141–150, doi: 10.1109/DCC.2016.67.

[32] C. Zhang and D. Florencio, "Analyzing the optimality of predictive transform coding using graph-based models," *IEEE Signal Process. Lett.*,

vol. 20, no. 1, pp. 106–109, Jan. 2013, doi: 10.1109/LSP.2012.2230165.

[33] A. K. Jain, "Fundamentals of digital image processing," *Comput. Vis. Graph. Image Process.*, vol. 46, no. 3, p. 400, Jun. 1989, doi: 10.1016/0734-189X(89)90041-8.

[34] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 98075, no. 2, Oct. 2014, pp. 2066–2070, doi: 10.1109/ICIP.2014.7025414.

[35] R. A. Cohen, "Attribute compression for sparse point clouds using graph transforms," in *Proc. ICIP*, 2016, pp. 1374–1378, doi: 10.1109/ICIP.2016.7532583.

[36] Y. Shao, "Hybrid point cloud attribute compression using slice-based layered structure and block-based intra prediction," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1199–1207, doi: 10.1145/3240508.3240696.

[37] Y. Xu, "Cluster-based point cloud coding with normal weighted graph Fourier transform," in *Proc. ICASSP*, 2018, pp. 1753–1757, doi: 10.1109/ICASSP.2018.8462684.

[38] *3DG, Common Test Conditions for PCC*, Standard ISO/IEC JTC1/SC29/WG11, 2020.

[39] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 828–842, Apr. 2017, doi: 10.1109/TCSVT.2016.2543039.

[40] T. Golla and R. Klein, "Real-time point cloud compression," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 5087–5092, doi: 10.1109/IROS.2015.7354093.

[41] J.-M. Lien, G. Kurillo, and R. Bajcsy, "Multi-camera tele-immersion system with real-time model driven data compression," *Vis. Comput.*, vol. 26, no. 1, pp. 3–15, Jan. 2010, doi: 10.1007/s00371-009-0367-8.

[42] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Sensor Fusion IV: Control Paradigms Data Struct.*, vol. 1611, pp. 586–606, Apr. 1992, doi: 10.1117/12.57955.

[43] G. K. M. Cheung *et al.*, "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2003, pp. I-77–I-84, doi: 10.1109/cvpr.2003.1211340.

[44] J. Kammerl *et al.*, "Real-time compression of point cloud streams," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 778–785, doi: 10.1109/ICRA.2012.6224647.

[45] D. C. Garcia and R. L. de Queiroz, "Context-based octree coding for point-cloud video," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 1412–1416, doi: 10.1109/ICIP.2017.8296514.

[46] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3D point cloud sequences," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1765–1778, Apr. 2016, doi: 10.1109/TIP.2016.2529506.

[47] Mitsubishi Electric's Mobile Mapping System (MMS). *Electric*. Accessed: Apr. 2017. [Online]. Available: http://www.mitsubishielectric.com/bu/mms/features/index.html

[48] H. Wiman and Q. Yuchu, "Fast compression and access of LiDAR point clouds using wavelets," in *Proc. Joint Urban Remote Sens. Event*, 2009, pp. 1–6, doi: 10.1109/URS.2009.5137589.

[49] *LiDAR Compressor*. Accessed: Jan. 2018. [Online]. Available: https://www.extensis.com/

[50] J. J. Im, "A real-time data compression for ground-based 3D LIDAR data using wavelets and compressive sensing," in *Proc. SCIS ISIS*, 2010, pp. 772–777.

[51] M. Isenburg, "LASzip," *Photogramm. Eng. Remote Sens.*, vol. 79, no. 2, pp. 209–217, Feb. 2013, doi: 10.14358/PERS.79.2.209.

[52] H. Masuda and J. He, "TIN generation and point-cloud compression for vehicle-based mobile mapping systems," *Adv. Eng. Informat.*, vol. 29, no. 4, pp. 841–850, Oct. 2015, doi: 10.1016/j.aei.2015.05.007.

[53] K. Kohira and H. Masuda, "Point-cloud compression for vehicle-based mobile mapping systems using portable network graphics," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 4, no. 2, pp. 99–106, 2017, doi: 10.5194/isprs-annals-IV-2-W4-99-2017.

[54] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing continuous point cloud data using image compression methods," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1712–1719, doi: 10.1109/ITSC.2016.7795789.

[55] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Continuous point cloud data compression using SLAM based prediction," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1744–1751, doi: 10.1109/IVS.2017.7995959.

[56] X. Sun, H. Ma, Y. Sun, and M. Liu, "A novel point cloud compression algorithm based on clustering," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2132–2139, Apr. 2019, doi: 10.1109/LRA.2019.2900747.

[57] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3D laser scans for online operation," in *Proc. IROS*, 2016, pp. 163–169, doi: 10.1109/IROS.2016.7759050.

[58] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3D laser scans," *PFG—J. Photogramm., Remote Sens. Geoinf. Sci.*, vol. 85, no. 1, pp. 41–52, Feb. 2017, doi: 10.1007/s41064-016-0003-y.

[59] B. Matejek, D. Haehn, F. Lekschas, M. Mitzenmacher, and H. Pfister, *Compresso: Efficient Compression of Segmentation Data for Connectomics* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10433. Springer-Verlag, 2017, pp. 781–788, doi: 10.1007/978-3-319-66182-7_89.

[60] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Point cloud compression for 3D LiDAR sensor using recurrent neural network with residual blocks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3274–3280, doi: 10.1109/ICRA.2019.8794264.

[61] T. Huang and Y. Liu, "3D point cloud geometry compression on deep learning," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 890–898, doi: 10.1145/3343031.3351061.

[62] M. Quach, "Learning convolutional transforms for lossy point cloud geometry compression," in *Proc. ICIP*, 2019, pp. 4320–4324, doi: 10.1109/ICIP.2019.8803413.

[63] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Real-time streaming point cloud compression for 3D LiDAR sensor using U-net," *IEEE Access*, vol. 7, pp. 113616–113625, 2019, doi: 10.1109/ACCESS.2019.2935253.

[64] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," 2020, *arXiv:2006.09043*. [Online]. Available: http://arxiv.org/abs/2006.09043

[65] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Point cloud coding: Adopting a deep learning-based approach," in *Proc. Picture Coding Symp. (PCS)*, Nov. 2019, pp. 19–23, doi: 10.1109/PCS48520.2019.8954537.

[66] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Deep learning-based point cloud coding: A behavior and performance study," in *Proc. 8th Eur. Workshop Vis. Inf. Process. (EUVIP)*, Oct. 2019, pp. 34–39, doi: 10.1109/EUVIP47703.2019.8946211.

[67] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned point cloud geometry compression," pp. 1–13, 2019, *arXiv:1909.12037*. [Online]. Available: https://arxiv.org/abs/1909.12037

[68] W. Yan, Y. Shao, S. Liu, T. H. Li, Z. Li, and G. Li, "Deep autoencoder-based lossy geometry compression for point clouds," 2019, *arXiv:1905.03691*. [Online]. Available: https://arxiv.org/abs/1905.03691

[69] C. R. Qi, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 652–660.

[70] M. Quach, G. Valenzise, and F. Dufaux, "Folding-based compression of point cloud attributes," 2020, *arXiv:2002.04439*. [Online]. Available: http://arxiv.org/abs/2002.04439

[71] Y. Yang, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215, doi: 10.1109/CVPR.2018.00029.

[72] E. Alexiou, "Towards neural network approaches for point cloud compression," *Proc. SPIE*, vol. 11510, Aug. 2020, Art. no. 1151008, doi: 10.1117/12.2569115.

[73] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4, doi: 10.1109/ICRA.2011.5980567.

[74] *Information Technology—Coding of Audio-Visual Objects—Part 12: ISO Base Media File Format*, Standard ISO/IEC 14496-12:2015, ISO/IEC, 2015.

[75] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 71–78, Jul. 1992, doi: 10.1145/142920.134011.

[76] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies—A voxelized point cloud dataset," 8i Labs, Culver City, CA, USA, Tech. Rep. m40059, Jan. 2017.

[77] S. L. J. Ricard *et al.*, "CGI-based dynamic point cloud test content," Technicolor, Rennes, France, Tech. Rep. m40050, Jan. 2017.

[78] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," Telecommun. Standardization Sector Video Coding Experts Group, Austin, TX, USA, Tech. Rep. ITU-T VCEG-M33, 2001.

[79] *3DG Current Status on Point Cloud Compression*, Standard ISO/IEC JTC1/SC29/WG11, 2015.

[80] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li, "Advanced 3D motion prediction for video-based dynamic point cloud compression," *IEEE Trans. Image Process.*, vol. 29, pp. 289–302, 2019, doi: 10.1109/TIP.2019.2931621.

[81] L. Li, Z. Li, S. Liu, and H. Li, "Occupancy-map-based rate distortion optimization for video-based point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 3167–3171, doi: 10.1109/ICIP.2019.8803233.

[82] *Parameter Values for the HDTV Standards for Production and International Programme Exchange BT Series Broadcasting Service*, document Recommended ITU-R BT.709-5, International Telecommunication Union, 2002, vol. 5, pp. 1–32. [Online]. Available: http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-5-200204-I!!PDF-E.pdf

[83] M. P. C. Tulvan and A. Gabrielli, *Datasets Update on Point-Cloud Compression for Cultural Objects*, document ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland. Accessed: May 2016. [Online]. Available: http://c3dc.fr/

[84] GTI-UPM. *3D Models/Point Clouds/Meshes*. Accessed: Jun. 2021. [Online]. Available: https://mpegfs.int-evry.fr/mpegcontent/

[85] G. L. H.-L. Guillaume *et al.*, *ULB Unicorn Photogrammetric Point Cloud Data*, Standard ISO/IEC JTC1/SC29/WG11, Macao, China, Oct. 2017.

[86] Stanford. *Vol. 2 Dataset*. [Online]. Available: http://buildingparser.stanford.edu/dataset.html

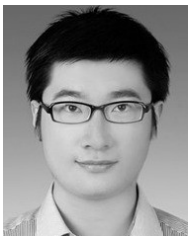
[87] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *Int. J. Robot. Res.*, vol. 30, no. 13, pp. 1543–1552, 2011, doi: 10.1177/0278364911400640.

[88] *BlackBerry Limited and QNX Software Systems Limited, Content Provided to MPEG*, document ISO/IEC JTC1/SC29/WG11, Ljubljana, Slovenia, 2018.

[89] A. V. R. Cohen *et al.*, *Mobile Mapping System Point Cloud Data From Mitisubishi Electric*, document ISO/IEC JTC1/SC29/WG11, Hobart, NSW, Australia, 2017.

[90] *Draco*. Accessed: Mar. 2021. [Online]. Available: https://github.com/google/draco

ABOUT THE AUTHORS



Chao Cao received the master's degree in multimedia networking from the Université Paris-Saclay, Paris, France, in 2018. He is currently working toward the Ph.D. degree in point cloud compression at Télécom Sud-Paris, Institut Polytechnique de Paris, É vry, France.

He has been participating and contributing to the moving picture experts group (MPEG) since 2018.

Euee S. Jang (Senior Member, IEEE) received the M.S.E.E. and Ph.D. degrees from the State University of New York, Buffalo, NY, USA, in 1994 and 1996, respectively.

He is currently a Professor with Hanyang University, Seoul, South Korea. He has been participating in the moving picture experts group (MPEG) in various leading roles since 1996.

Marius Preda received the degree in engineering from the University POLITEHNICA of Bucharest Bucharest, Romania, in 1998, the Ph.D. degree in mathematics and informatics from University Paris V, Paris, France, in 2002, and the eMBA degree from the IMT Business School, Paris, in 2015.

He is currently an Associate Professor with the Institut MINES-Télécom, Paris, where he is also the Chairman of the 3D Graphics Group of ISO MPEG. He contributed to various ISO standards with technologies in the fields of 3-D graphics, virtual worlds, and augmented reality.

Vladyslav Zakharchenko received the Ph.D. degree in optics and photonics from the National Technical University of Ukraine, Kyiv, Ukraine, in 2012.

He worked with Samsung Electronics, Seoul, South Korea, and AMD, Kyiv, on multimedia technologies. He is currently a Principal Engineer and a Point Cloud Coding Group Leader with Ofinno LLC, Reston, VA, USA. His current work is devoted to innovations for volumetric media and close collaboration with international standard committees.

Dr. Zakharchenko is also an Editor of the ISO/IEC Committee Draft for Video-Based Point Cloud Compression (ISO/IEC 23090-5) and the ISO/IEC Working Draft for Geometry-Based Point Cloud Compression (ISO/IEC 23090-9).

Titus Zaharia received the Engineering degree in electronics and telecommunications and the M.Sc. degree from University POLITEHNICA of Bucharest Bucharest, Romania, in 1995 and 1996, respectively, and the Ph.D. degree in mathematics and computer science from University Paris V, Paris, France.

He is currently a Full Professor with Télécom SudParis, Institut Polytechnique de Paris, Évry, France. His research interests include visual content representation methods, with 2-D/3-D compression, reconstruction, recognition, and indexing applications.