A Comprehensive Study and Comparison of Core Technologies for MPEG 3-D Point Cloud Compression

Hao Liu, Hui Yuan, *Senior Member, IEEE*, Qi Liu, Junhui Hou, *Member, IEEE*, and Ju Liu, *Senior Member, IEEE*

***Abstract*—Point cloud based 3D visual representation is becoming popular due to its ability to exhibit the real world in a more comprehensive and immersive way. However, under a limited network bandwidth, it is very challenging to communicate this kind of media due to its huge data volume. Therefore, the MPEG have launched the standardization for point cloud compression (PCC), and proposed three model categories, i.e., TMC1, TMC2, and TMC3. Because the 3D geometry compression methods of TMC1 and TMC3 are similar, TMC1 and TMC3 are further merged into a new platform namely TMC13. In this paper, we first introduce some basic technologies that are usually used in 3D point cloud compression, then review the encoder architectures of these test models in detail, and finally analyze their rate distortion performance as well as complexity quantitatively for different cases (i.e., lossless geometry and lossless color, lossless geometry and lossy color, lossy geometry and lossy color) by using 16 benchmark 3D point clouds that are recommended by MPEG. Experimental results demonstrate that the coding efficiency of TMC2 is the best on average (especially for lossy geometry and lossy color compression) for dense point clouds while TMC13 achieves the optimal coding performance for sparse and noisy point clouds with lower time complexity.**

***Index Terms*—3D point clouds, MPEG compression standard, compression performance, virtual/augmented reality.**

Manuscript received June 8, 2019; revised September 6, 2019; accepted October 29, 2019. Date of publication December 30, 2019; date of current version September 3, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC0831003, in part by the National Natural Science Foundation of China under Grant 61571274 and Grant 61871342, in part by the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University under Grant VRLAB2019B03, in part by the Shandong Natural Science Funds for Distinguished Young Scholar under Grant JQ201614, in part by the Shandong Provincial Key Research and Development Plan under Grant 2017CXGC1504, in part by the Young Scholars Program of Shandong University under Grant 2015WLJH39. *(Corresponding author: Hui Yuan.)*

H. Liu, Q. Liu, and J. Liu are with the School of Information Science and Engineering, Shandong University, Qingdao 266200, China (e-mail: liuhaoxb@gmail.com; sdqi.liu@gmail.com; juliu@sdu.edu.cn).

H. Yuan is with the School of Control Science and Engineering, Shandong University, Jinan 250000, China (e-mail: huiyuan@sdu.edu.cn).

J. Hou is with the Department of Computer Science, City University of Hong Kong, Hong Kong 999077 (e-mail: jh.hou@cityu.edu.hk).

This article includes an additional appendix pdf file which provides rate distortions performance of TMC13 and TMC2 under all experiment settings in details.

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2019.2957652

I. Introduction

NOWADAYS, 3D models are becoming more and more popular and important in many application fields such as 3D gaming, animation, virtual/augmented reality (VR/AR), and scientific visualization. 3D polygonal meshes can represent a 3D model by reconstructing the 3D surface through polygonal meshes consisting of a set of vertices (geometry) and the corresponding connectivity information (topology). However, there are also lots of natural scenes or objects (such as hairs, forest) with non-manifold geometry [1] which cannot be easily represented by 3D polygonal meshes. Besides, the data volume of a 3D polygonal mesh is very large, while the rendering complexity is also very high. Previously, there are lots of compression methods for 3D polygonal meshes [2]–[8], but they mainly focus on the computer generated (or man-made) 3D models.

In contrast, 3D point clouds are usually composed of a set of 3D coordinates indicating the locations of points, along with some attributes, e.g., color and normal vector. 3D point clouds can represent the points in 3D scenes and objects directly, and thus, the polygonal overhead is saved. Besides, 3D point clouds are flexible enough to represent non-manifold structure, and they are more convenient for computing, transmission, and storage for real time acquired 3D scenes without the connectivity constraint. In the early days, because of the limited 3D sense technology, the 3D acquisition devices can only get the sparse 3D point clouds of small objects, which limited their extensive applications. With the advancement of 3D acquisition technology in recent years, the acquisition of dense 3D point clouds can be accomplished, and thus, the application of 3D point clouds in daily life is becoming more and more applicable, e.g., robotics [9], vehicle-based mobile mapping system [10], intelligent transportation systems [11], VR/AR [12], cultural heritage preservation, automated driving.

Although the 3D point cloud is very practical, the huge data volume of a 3D point cloud limits its extensive applications. Taking a 3D point cloud with one million points of 3D coordinates (12bits per dimension of a position) and RGB (8bits per component of a point) color attributes as an example, the date volume is $1000000 \times (12 \times 3 + 8 \times 3) = 6 \times 10^7 \approx 57$M bits. Note that it is only a static point cloud. For a dynamic point cloud with 25 frames per second, the data volume to be transmitted in a second will be $25 \times 57 \approx 1425$M bits, which is unaffordable for the current network bandwidth. Therefore,

efficient compression for 3D point cloud is becoming an urgent task to be addressed.

Currently, some LiDAR sensor data compression standards [13], [14] have been published and applied in related industries, but they directly compress the geometry raw data (e.g., *.las, *.e57) associated with other attributes such as GPS timestamps, longitude and latitude. Nevertheless, a lot of application fields need fused data that can not be captured by a single sensor. For example, VR/AR needs spatial geometry information from LiDAR and color information captured by RGB cameras to accurately represent colorful 3D scenes. In other words, the geometry and color information should be fused together before application.

Based on the fused data that is usually represented by the Polygon File Format (*.ply), MPEG called for proposals (CfP) from 2017 and is developing the corresponding compression standards for 3D point clouds. In October 2017, the first several platforms, i.e., test model category 1 (TMC1) [15] for static 3D point cloud compression, test model category 2 (TMC2) [16] for dynamic 3D point cloud compression, and test model category 3 (TMC3) [17] for dynamically acquired 3D point clouds, were proposed. Although Schwarz *et al.* [1] have introduced the progress of 3D point cloud compression standardization of MPEG, only qualitative evaluations of the three test models are provided. Since MPEG PCC standards is still an ongoing project, more and more new technologies are emerging. Recently, because the 3D geometric compression methods of TMC1 and TMC3 are similar, TMC1 and TMC3 have been combined into a new platform, namely TMC13 [18] (also called as G-PCC, i.e., geometry-based PCC), while the TMC2 is called as V-PCC (i.e., video-based PCC).

In this paper, we first introduce some basic technologies for 3D point cloud compression briefly, and then introduce the core technologies and the development trends of the G-PCC and V-PCC platforms in detail. After that, we compare the subjective and objective quality of the G-PCC and V-PCC in different experimental settings (i.e., both geometry and color are compressed losslessly; geometry is compressed losslessly, while the color is compressed lossily; both geometry and color are lossily compressed), and summarize some important conclusions based on the experimental results, aiming at encouraging more researchers to participate into the research of 3D point cloud compression.

The remaining of the paper is organized as follows. Section II summarizes the related works of point cloud compression in recent years. Important basic technologies for point cloud compression are briefly presented in Section III. In Section IV, the core technologies of G-PCC and V-PCC are reviewed in detail. Experimental results and analysis are given in Section V, while the conclusions are given in Section VI.

II. State of the Art

In the past few years, great progress on 3D point cloud compression has been made [19]–[38]. There are two main problems in point cloud compression, i.e., geometry and attributes (i.e., generally, it refers to color) compression.

For geometry compression, Schnabel and Klein proposed an octree based representation [19] which has been proven to be an efficient way to compress the geometry information. Elseberg *et al.* [20] proposed an efficient octree to store and compress 3D data without loss of precision. Huang *et al.* [21] proposed a progressive point cloud compression based on octree. Besides octree representation, there are also some other lossy compression methods [22]–[24] for geometry information. Ochotta and Saupe [22] partitioned the point cloud in a number of point clusters. A surface patch associated to each cluster is parameterized as a height field, which is efficiently encoded with a shape adaptive wavelet coder. Ahn *et al.* [23] proposed an adaptive range image coding algorithm for the geometry compression of large-scale 3D point clouds. Recently, de Oliveira Rente *et al.* [24] proposed a hybrid geometry compression algorithm based on octree and graph transform.

Besides geometry information, color information should also be compressed efficiently. Different from geometry information, lossy compression is usually used for color information. Similar to the traditional image compression, the compression of color in 3D point clouds is also composed of prediction, transform, quantization, and entropy coding. With the help of the excellent compression efficiency of JPEG codec, Mekuria *et al.* [25] proposed to compress the color information by mapping it to a 2D grid with a depth-first octree traversal. Houshiar and Nüchter [26] mapped the point cloud to the panoramic image, and then compressed the mapped image using the existing image compression methods. This kind of method cannot exploit the inherent correlation among points efficiently. Accordingly, a lot of researchers focused on how to design efficient decorrelation method for the irregular data structure of 3D point clouds. The recently emerged graph transform (GT) [27], [28] has proven to be very suitable for 3D point cloud compression. In the graph transform, a graph is created by calculating the spatial distance of different nodes, and was used to remove color information correlation. Shao *et al.* [29] used a k-d tree partitioning method to code color information based on GT. On the basis of [29], they also proposed a slice-partition schemes and established a tree prediction mode based on k-d tree structure in [30]. To explore the relevance of point cloud attributes, a region adaptive hierarchical transform (RAHT) was proposed in [31]. Besides, sparse representations have also proved to be efficient for 3D point cloud compression. Hou *et al.* [32] proposed a sparse representation method to encode point cloud attributes based on GT basis. Subsequently, they improved the performance of [32] through an inter-block prediction scheme and effective entropy coding strategy in [33]. The above methods are all designed for static point clouds. Since dynamic point clouds are becoming more and more necessary in practical applications, efficient compression methods for dynamic point clouds are also researched. Anis *et al.* [34] used consistent sub-divisional triangular meshes to represent point clouds and design efficient wavelet transforms. Thanou *et al.* [35] extended the GT-based work to dynamic point clouds. De Queiroz and Chou [36] proposed a motion-compensated approach to encoding dynamic voxelized point

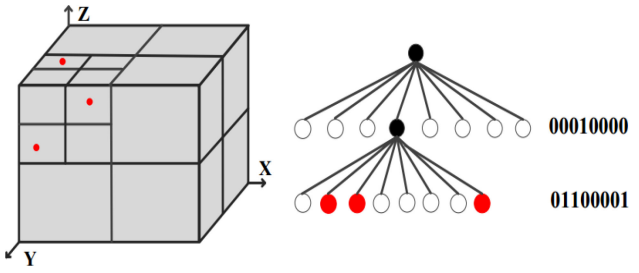


Fig. 1. Octree decomposition. Current octree depth is 2, three red occupied points are divided into upper left corner in bounding box.

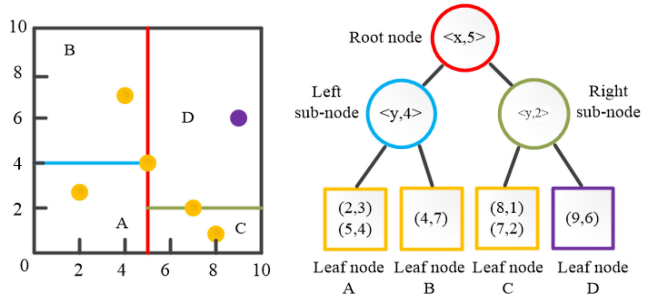


Fig. 2. 2D k-d tree splitting and nearest neighbor search. A 2D bounding box with size 10×10 is generated, the longest edge of within the bounding box is used to the splitting direction, and the median of the points within the bounding box is applied to be separation point. Assuming that the coordinates of the point being queried is (9,7), the coordinates of the points at which the k-d tree is generated are (2,3),(4,7),(5,4),(9,6),(8,1),(7,2). m indicates splitting direction and n represents the coordinate of separation point in <m,n> . The 2D points are divided by k-d tree in yellow and purple rectangle, and (9, 6) is the point closest to the query point in purple rectangles.

clouds. Li *et al.* [37] proposed a novel geometry-based motion prediction method for 3D point clouds. Besides coding tools, efficient rate distortion optimization can also affect the compression efficiency of 3D point clouds. In [38], Liu *et al.* developed a model-based joint color-geometry bit allocation method which uses an octree depth and the JPEG quality value as the encoding parameters.

III. Important Basic Technologies

In 3D point cloud compression, there are some extensively adopted important basic technologies, i.e., octree decomposition, k-d tree decomposition, and level of details (LoD) description. In this section, we will briefly introduce these technologies. Note that octree, k-d tree and LoD are just the preprocess before compression and not really compress the data. They add overhead for accessing the data. Thus, data size after k-d tree or LoD description could be larger.

A. Octree Decomposition

In octree-based decomposition, the corresponding position of a cubical bounding box $\mathbb{B}$ is given by two extreme axes (0,0,0) and $(2^n, 2^n, 2^n)$:

$$2^n \geq \max\left(\max(x_i), \max(y_i), \max(z_i)\right)_{i=0,\ldots,T-1}, \tag{1}$$

where n is the minimum integer satisfying the above inequality, $(x_i, y_i, z_i)_{i=0,\ldots,T-1}$ is the set of 3D positions associated with the points of the original point cloud, and T is the total number of points in point clouds.

As shown in Fig. 1, an octree data structure is generated by splitting $\mathbb{B}$ recursively. At each octree level, a cube (corresponding to a node in the octree) is divided into 8 sub-cubes. A subdivision code with 8 bits is then generated by associating a 1-bit value with each sub-cube in order to indicate whether it contains points (labeled as 1) or not (labeled as 0). Only the sub-cubes that consist of points more than 1 are further divided, until all the sub-cubes only contain 1 point or the predefined octree depth is reached.

B. K-d Tree Splitting and Nearest Neighbor Search

The k-d tree is a binary tree in which each leaf node consists of a set of points with k-dimensions. k-d tree is widely used in various fields, e.g., nearest neighbor search [39] and collision detection [40]. To take a 3D point cloud as an example, the k-d tree is usually generated based on the three dimensional coordinates of points. At first, the whole point cloud is considered as a root node. Then, a plane is used to divide the root node into two parts. Points belonging to the left (resp. right) of this plane are considered as the left (resp. right) sub-node of the root node. The above splitting method is then iterated based on the left and the right sub-nodes respectively, until a stopping condition is achieved.

In practice, a bounding box containing all the points is generated first. Then, a plane is chosen to divide the bounding box into two sub-bounding box (sub-nodes) along with a certain direction. There are two main methods to choose the splitting direction, one is to divide each dimension of bounding box recursively in a certain order; another is to split the longest edge of the bounding box. After that, the median or the average value of the points' coordinates in the splitting direction is used as the separation position. The splitting strategy is then looped until a termination condition (the number of points in a bounding box is smaller than a predefined threshold or a predefined tree depth, etc.) is achieved. Note that the partitioning method of the k-d tree is likely to be flexible for various application scenarios.

Based on the generated k-d tree, given a target point position, one can easily find its nearest neighbor in a leaf node by a binary tree search operation. Fig. 2 shows a simple example of k-d tree splitting and nearest neighbor search for a set of points with only 2 dimensions.

C. LoD Description

In the LoD generation, points will be divided into S reorganized level sets R_s ($s \in \{1, \ldots, S\}$) based on a user specified set of point-to-point spatial Euclidean distance thresholds $dist_s$. Note that the distance thresholds must satisfy the following constraints:

$$\begin{cases} dist_s < dist_{s-1}, \\ dist_S = 0. \end{cases} \tag{2}$$

As shown in Fig. 3, in the beginning, all the points except for the selected initial point are marked as unvisited, i.e., the

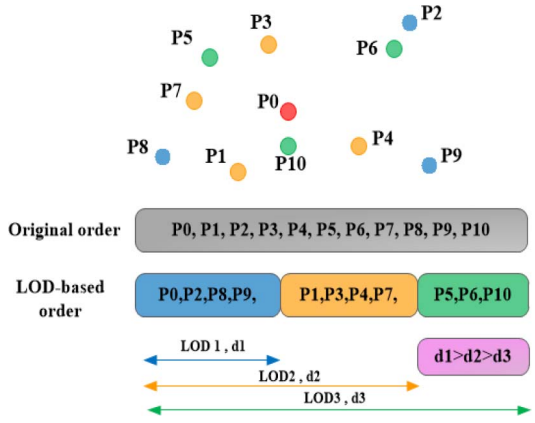


Fig. 3. LOD generation description. Red point P0 is the initial point.

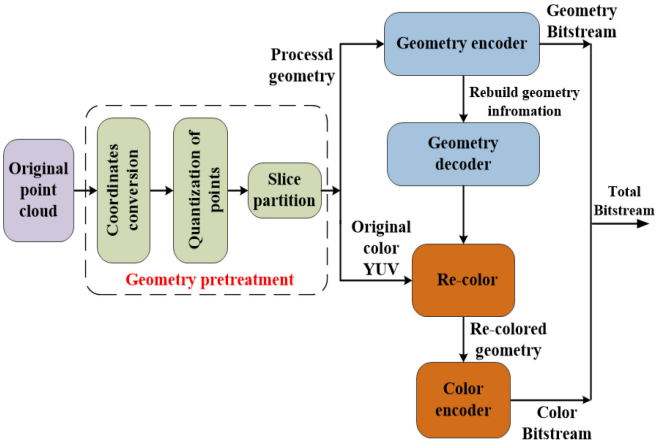


Fig. 4. TMC1 encoder architecture.

set of visited points, denoted by $\mathcal{V}$, only contains the initial point. Then, the LoD is generated by the following steps iteratively.

1) Set s=1, $dist_1$=MAX, traverse all the points to construct the set of visited points $\mathcal{V}$ and the reorganized level set R_s: The current point would be neglected, if it has been visited; or else the minimum distance $D_{\min}$ of the current point to the points in the set $\mathcal{V}$ is calculated. If $D_{\min}$ is strictly smaller than $dist_s$, the current point is neglected, otherwise, the current point would be marked as visited and put into $\mathcal{V}$ and R_s.
2) The s-th LoD, denoted as LoD_s, is acquired by taking the union of the reorganized level sets $R_1, \ldots, R_s$.
3) $s \leftarrow s+1$, and go to 1), until all the LoDs are built or all the points have been marked as visited.

IV. Test Model Categories for 3D Point Cloud Compression

Since TMC1 and TMC3 have been merged into the new platform TMC13, in the following, we will first introduce the core technologies of TMC13, and then describe the core technologies of TMC2.

In TMC13, there are two kinds of encoder choice, i.e., the RAHT-based encoder (corresponds to the previous TMC1 [15]), and the LoD-based encoder (corresponds to the previous TMC3 [17]).

A. RAHT-Based Encoder of TMC13 (TMC1, G-PCC [41])

Chou *et al.* [15] first proposed RAHT-based point cloud compression platform, i.e., TMC1 in October 2017. Fig. 4 shows the encoder architecture of TMC1. As we can see that geometry compression and color compression are separately processed. The coordinates conversion module converts original geometry (namely world coordinates) to normalized geometry (namely frame coordinates). The quantization module will down-sample the points with a predefined quantization scale. The geometry encoder and decoder modules use octree and triangulation to encode/decode geometry information. The decoded geometry points are then recolored by the re-color module. The re-colored geometry is then encoded by a color encoder module in which the RAHT is used to efficiently compress the color information.

1) Coordinates Conversion: The geometry information of a 3D point cloud is first normalized from the 3D world coordinate to the frame coordinates:

$$(x_i, y_i, z_i) = \left(\left(x_i^{world}, y_i^{world}, z_i^{world}\right) - \left(t_x, t_y, t_z\right)\right)/\alpha, \quad (3)$$

where (t_x, t_y, t_z) and α are translation and scale parameters, $(x_i^{world}, y_i^{world}, z_i^{world})$ and (x_i, y_i, z_i) are the 3D world coordinate and the corresponding frame coordinate of the point i, $i \in 1, \ldots, N$.

2) Quantization of Points: After coordinateaŕs conversion, the coordinates of a 3D point cloud can be quantized depending on user requirement. Let $\boldsymbol{X}_i = (x_i, y_i, z_i)$ denote the 3D coordinate of a point. The quantized coordinates $\breve{\boldsymbol{X}}_i = (\breve{x}_i, \breve{y}_i, \breve{z}_i)$ can be calculated by

$$\breve{\boldsymbol{X}}_i = Round((\boldsymbol{X}_i - \boldsymbol{X}_{\min}) \times q), \quad (4)$$

where q is a position quantization scale factor, $\boldsymbol{X}_{\min} = (x_{\min}, y_{\min}, z_{\min})$, and $x_{\min}$, $y_{\min}$, and $z_{\min}$ are the minimum coordinates along the three axes of all the points. Note that there may exist points with the same coordinates after the coordinate quantization. The TMC13 platform also provides an option to remove the duplicated points.

3) Slice Partition: There are two slice partition methods in TMC13. One is the longest edge-based uniform partition. That is to say, the minimum and the maximum edge ($edge_{min}$ and $edge_{max}$) among the three directions x, y, and z, are first calculated. Then, $edge_{min}$ is used as the slice partition interval, to cut the point cloud along with the direction whose edge is the longest. The remainder of points are also considered as slices. Another slice partition method is the octree-based uniform partition. In this method, each side of the bounding box of the point cloud is rounded up to a power of 2, and a pre-defined parameter $depth_{partition}$ is used as the termination condition of the octree decomposition, as described in Part A of Section III.

4) Geometry Encoder: In the geometry encoder module, octree-based decomposition is further conducted on all the partitioned slices. A bounding box is first constructed for each

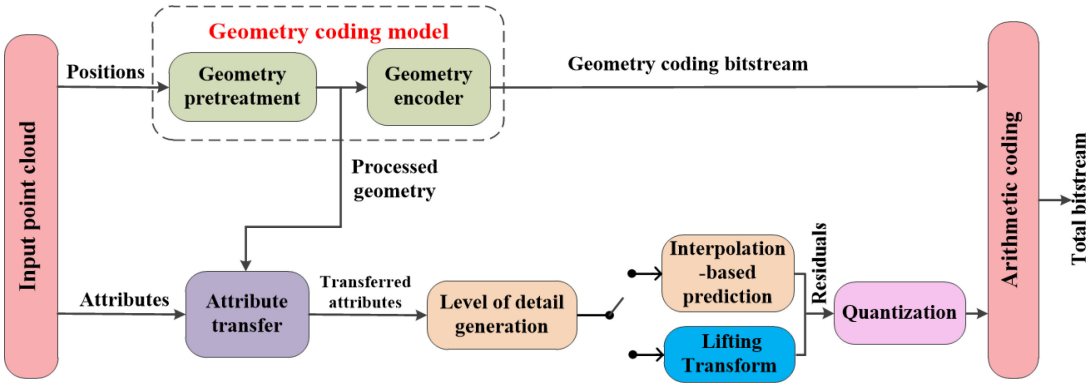


Fig. 5. TMC3 encoder architecture.

slice. The partitioned slices can then be divided into voxels with size of $W \times W \times W$, where $W = 2^{d-l}$, d denotes the octree depth, and l denotes the octree level that corresponds to the available octree depth which is less than or equal to d. If d=l, there only exists one point in a voxel. In this case, the geometry information will be compressed losslessly; otherwise, it will be compressed lossily.

5) Lossless Geometry Compression: In this case, the point cloud will be octree decomposed finely until there exists only one point in a voxel, i.e., $d = l$, $W = 1$. If the points are distributed uniformly, the geometry information can be efficiently compressed by recoding the octree decomposition procedure. However, when there are isolated points, the isolated points should also be decomposed repeatedly to record their fine positions in the bounding box. Consequently, a lot of unnecessary bits must be consumed. In order to avoid this circumstance, a technique namely direct coding mode (DCM) [42] is proposed. In this method, the coordinates of the isolated points are extracted and encoded independently. Furthermore, a neighbor-dependent entropy context [43] is used for entropy coding. In this method, to calculate the probability distribution of semantic symbols for the successive arithmetic encoder, 10 context patterns are proposed and updated separately based on the neighboring structure of the current node. In order to further improve the coding efficiency, an occupancy prediction method is also proposed in [44] for the context update.

6) Lossy Geometry Compression: In this case, l is smaller than d, i.e., the geometry loss is inevitable. The geometry information within a voxel with the size of $W \times W \times W$ is represented as a surface that intersects each edge of the voxel. Since there are 12 edges bounding on a voxel, there are at most 12 intersections between the surface and the voxel. The intersections are also called as vertices. A vertex along an edge can be discovered when there is no less than one occupied voxel adjacent to the edge among the whole blocks that share the edge. The location of a discovered vertex along corresponding edge is the average location along the edge of all such voxels adjacent to the edge among whole blocks that share the edge.

The vertices are then encoded as follows. Firstly, a set of flags are used to label whether an edge of a cube contains a vertex or not. Secondly, for the edge that includes a vertex, the relative location of the vertex on the edge is uniformly quantized. Finally, the set of flags and the quantized relative locations are entropy coded. By using these vertices, non-planar polygons (triangles) could be constructed in the geometry decoder, and the lost voxels could be approximately estimated by extracting the refined vertices from the constructed triangles.

Besides the triangle surface approximation-based lossy compression method, geometry information can also be quantized directly (2 of Part A of Section IV) for lossy compression namely direct geometry quantization method.

7) Geometry Decoder: Before compressing the color information, the geometry information must be decoded first. For losslessly compressed geometry, direct entropy decoder is enough, while for lossy compressed geometry, surface reconstruction will be performed after entropy decoder of the vertices. Then, non-planar triangles will be constructed based on the decoded vertices. Finally, refined vertices $(\tilde{x}_j, \tilde{y}_j, \tilde{z}_j)$, $j \in 1, \ldots, M$, could be obtained by up-sampling these triangles so as to approximately estimate the lost geometry information [41].

8) Re-Color: The re-color module assigns original voxel colors to the refined vertices. It is implemented by assigning the color $(\tilde{Y}_j, \tilde{U}_j, \tilde{V}_j)$ that is equal to (Y_i, U_i, V_i) to a refined vertex $(\tilde{x}_j, \tilde{y}_j, \tilde{z}_j)$, where i is the index of voxel position (x_i, y_i, z_i) that is closest to $(\tilde{x}_j, \tilde{y}_j, \tilde{z}_j)$.

9) Color Encoder: The reconstructed voxel colors $(\tilde{Y}_j, \tilde{U}_j, \tilde{V}_j)$ are then transformed by RAHT, uniformly quantized, and entropy coded. To see the detailed procedure of RAHT, please refer to [31].

B. LOD-Based Encoder of TMC13 (TMC3, G-PCC)

LOD-based TMC13 (i.e., TMC3) is proposed by *Apple Inc.* in October 2017. As shown in Fig. 5, geometry information of a 3D point cloud is also compressed by octree decomposition, which is the same with the RAHT-based TMC13 encoder (i.e., TMC1). The color information is then differentially encoded by a LoD generation procedure.

1) Geometry Pretreatment & Geometry Encoder: The geometry preprocessing and geometry encoder are the same

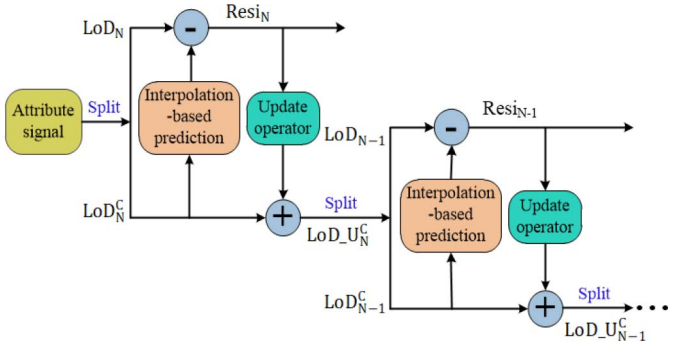


Fig. 6. Lifting transform scheme.

with the RAHT-based TMC13, as shown in Part A of current section.

2) Attributes Transfer: The color information of the original 3D point cloud is then transferred to the quantized coordinates by k-d tree to find the minimum Euclidian difference between the original coordinates and the reconstructed coordinates generated from inverse quantization.

3) LoD Generation: Afterwards, the geometry information of the original point cloud is re-organized by LoD, as shown in the part C of Section III. Besides, there are some other similar LOD generation methods, i.e., scalable complexity-based method [45] and binary tree-based method [41], that are also adopted in TMC13.

4) Interpolation-Based Prediction (Predict Transform): The color information is then predicted and differentially encoded in the order defined by the LoD generation procedure. Note that only the already encoded/decoded points could be taken as reference points for prediction. For the i-th point, its attribute value c_i can be predicted by the linear combination of its k nearest neighbors:

$$\hat{c}_i = Round\left(\sum_{j\in\Im_i}\left(\frac{\frac{1}{\eta_j^2}}{\sum_{j\in\Im_i}\frac{1}{\eta_j^2}}\right)\tilde{c}_j\right), \tag{5}$$

where $\Im_i$ is the set of the k-nearest neighbors that are already decoded of the i-th point, $\tilde{c}_j$ is the corresponding reconstructed attribute values, η_j is the distance between the i-th point and the j-th neighbor. Finally, residual r_i between the attribute value of the i-th point c_i and the predicted attribute value $\hat{c}_i$ is quantized and arithmetically encoded.

5) Lifting Transform (LT): To further improve the coding efficiency, a LT is proposed in [41]. The basis of this method is to make the points in the lower LODs [45] more influential (i.e., because they are frequently used to make predictions). In this method, the points (denoted as $\mathbf{\Psi}$) in lower LoD that are used to predict the points (denoted as $\mathbf{\Phi}$) in the N-th LoD is updated by the residual of the points in the N-th LoD, as shown in Fig. 6. In this figure, the attribute values of points in the N-th LoD (denoted as LoD_N) are predicted by the attribute values in the other LoDs (denoted as LoD_N^C) whose LoD order is smaller than N. Let $Resi_N$ denotes the predicted attribute residuals of the points in the N-th LoD, $\mathrm{LoD\_U}_N^C$ represents the updated attributes in lower LoDs, where N is largest LOD order. The update procedure is described as follows.

Take the points in the N-th LoD as an example. Let ψ be a point in $\mathbf{\Psi}$. Then, find the point set (i.e., $\mathbf{\Phi}$ in the N-th LoD) that are partially predicted (influenced) by $\mathbf{\Psi}$. The attribute values of updated point $Att(\psi')$ in $\mathbf{\Psi}$ can be calculated by

$$Att(\psi') = Att(\psi) + \frac{\sum_i^K \zeta_i \times w(\phi_i) \times Resi(i)}{\sum_i^K \zeta_i \times w(\phi_i)}, \tag{6}$$

where $Att(\psi)$ is the attribute value of point ψ, ϕ_i is a point in $\mathbf{\Phi}$, $w(\phi_i)$ is the corresponding influence weight (with initial value of 1), K denotes the number of points that are influenced by ψ, ζ_i depends on the Euclidean distance between by ψ and ϕ_i. Afterwards, the influence weight $w(\psi)$ of the point ψ in $\mathbf{\Psi}$ can be calculated by

$$w(\psi) = w(\psi) + \sum_i^K \zeta_i \times w(\phi_i). \tag{7}$$

Similarly, the updated points in the $(N-1)$-th LoD can be processed based on the above procedure recursively, until the last LoD is reached.

The residuals are finally quantized and entropy encoded to generate a bit stream. During the quantization procedure, an additional technique namely adaptive quantization [41] was also adopted in the LoD-based TMC13. In the adaptive quantization, the influence weights of each point are used to guide the quantization, i.e., the coding attribute residual of a point is further multiplied by the root of the influence weight of the point before the quantization. Note that it is no need to encode the influence weights of each points because they can be calculated during the lifting transform iteratively.

C. Test Model Category 2 (TMC2, V-PCC [46])

TMC2 was also proposed by *Apple Inc*. in October 2017, the main idea of TMC2 is to convert the point cloud data into a set of video sequences and use state-of-the-art video codec (e.g., HEVC) to compress the geometry and texture information, as shown in Fig. 7.

1) Patch Generation: Given a 3D point cloud, the normal of each point is first computed by principal component analysis (PCA) based on the set of neighboring points. By utilizing normal information, an initial clustering of the point cloud is obtained through mapping each point to six predefined planes. The mapping operation could be performed by maximizing the dot product of the point normal and the pre-defined plane normal. Then, the spatially neighboring pixels in a cluster are extracted as patches.

2) Packing & Occupancy Map Generation: Afterwards, the extracted patches are mapped onto a 2D grid (with size of $W \times H$) by a packing procedure during which the occupancy information of each of the 2D grid is also recorded. The packing procedure aims at mapping the extracted patches onto a 2D grid, while trying to minimize the unused space of the 2D grid and guarantee that the pixels in each block (e.g., with size of 16×16) of the 2D grid corresponds to a unique patch. The occupancy map (OM) is a binary map that indicates each block of the 2D grid whether it belongs to the point cloud or empty. Specifically, each block consists of multiple smaller $h \times h$ (e.g., $h = 4$) sub-blocks. If a sub-block is marked as 1

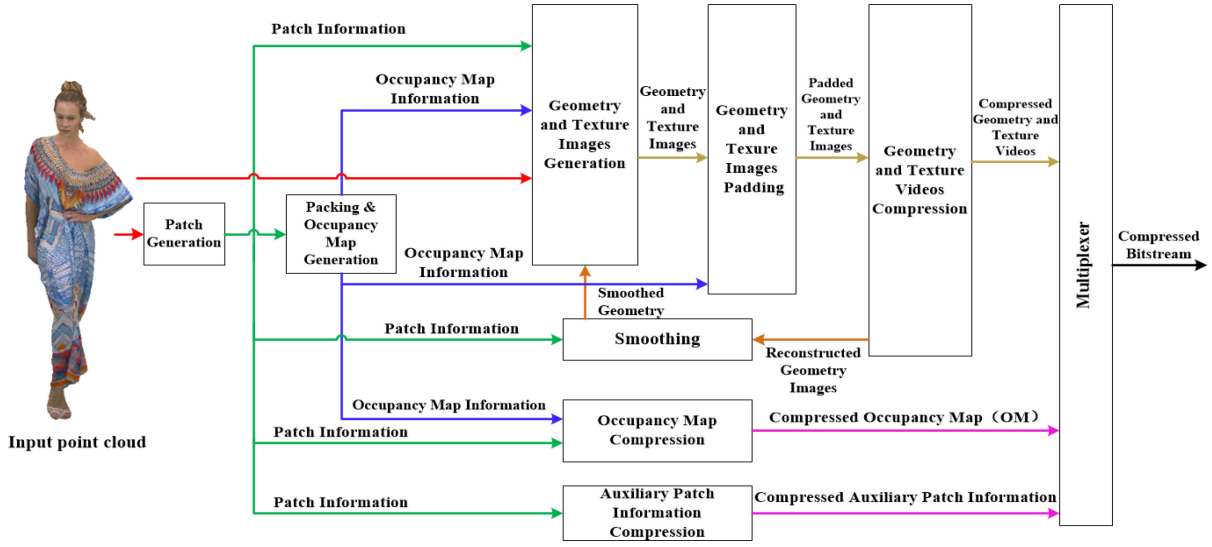


Fig. 7. TMC2 encoder architecture.

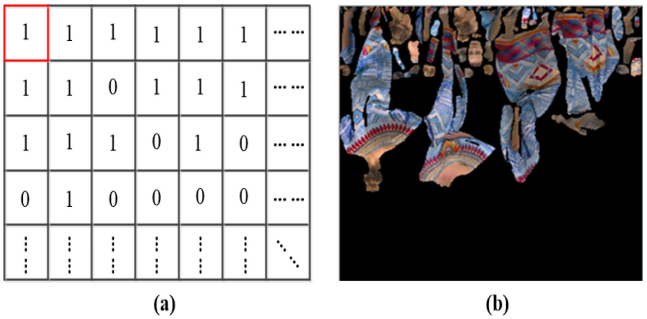


Fig. 8. (a) A block-based occupancy map. Occupancy map with size of $W \times H$, a block (red block in upper left corner) with size 16×16 was indicated 1, if all sub-blocks of it are marked as full, and 0 otherwise. (b) A mapped 2D point cloud texture image [46] is generated by image generation (the empty pixels are filled in black).

(namely full), then it must include at least a real (non-padded) pixel. Otherwise it is marked as 0 (namely non-full). When all sub-blocks in a block are marked 1, then corresponding block is labeled 1, and 0 otherwise. Note that above all marked binary flags would be encoded as additional information [46]. Fig. 8 (a) shows a block-based occupancy map.

3) Image Generation: 3D to 2D mapping is exploited to convert the geometry and texture of the point cloud into images, as shown in Fig. 8 (b). To overcome the problem that several points may be projected onto the same pixel, each patch is projected onto two images called near layer and far layer. More precisely, let $I(u, v)$ denote the set of points in the current patch that would be projected to the same pixel position (u, v). The near layer will store the point of $I(u, v)$ with the minimum depth D_0, while the far layer, records the point with the maximum depth $D_0 + \Delta$, where Δ refers to surface thickness parameter.

4) Image Padding & Coding: To generate piecewise smooth images suited for video compression, the empty space between patches in images is filled with the neighbors by a padding procedure. This smoothing operation is designed to alleviate potential compression artifacts caused by the discontinuities at the patch boundaries. Note that the padding points are not counted as occupied point. Finally, the generated occupancy map, geometry and texture images are encoded by HEVC as video frames. Different quantization steps in the HEVC encoding of the geometry and color video frames give different distortions and bit rates.

It is worthy pointing out that the related technologies are still ongoing. Recently, several new technologies, i.e., color and geometry smoothing [47], patch flexible orientation [48], push-pull background filling [49], interleaved absolute depth and color coding [50], patch-based color sub-sampling [51], and spatially adaptive geometry interpolation [52], were proposed and implemented in the newest version of TMC2 by MPEG PCC group, to further improve its performance.

V. Experimental Results and Analysis

To verify the performance of the test model categories, extensive experiments were conducted over 16 3D point cloud datasets, as shown in Fig. 9, in which the sub-figures (a)-(j) are portraits with dense points, and (k)-(p) are engravings with sparse points. Sub-figures (a)-(e) are chosen from dynamic voxelized point cloud data sequences of Microsoft Voxelized Upper Bodies [53], and the corresponding extracted frame indexes are 0, 39, 0, 1, 18, respectively. Sub-figures (f)-(j) were obtained from 8i Voxelized Full Bodies [54]. Remaining large-scale sparse point clouds in sub-figures (k)-(p) were extracted from static objects and scenes datasets of test class A in common test conditions (CTC) [55] that can be downloaded from [56]. In the experiments, the RGB formatted attributes were first converted to the YUV format for compression. The reference software [57], TMC13 version 5 (TMC13 V5) and TMC2 version 5 (TMC2 V5) were used. To evaluate the quality of reconstructed 3D point clouds, point to point distortion metric [58] was used for both geometry and color information. Based on the point to point distortion, the corresponding PSNRs of geometry and color information were

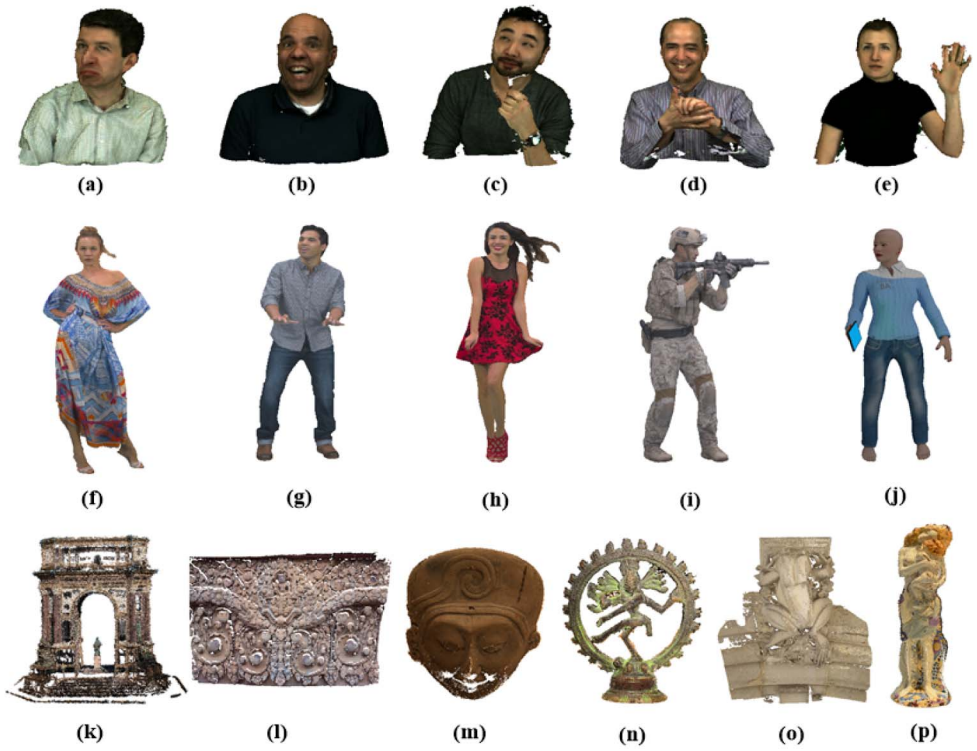


Fig. 9. 3D point cloud sequences used in experiments. (a) Andrew, (b) Ricardo, (c) David, (d) Phil, (e) Sarah, (f) Longdress_vox10_1300, (g) Loot_vox10_1200, (h) Redandblack_vox10_1550, (i) Soldier_vox10_0690, (j) Queen_0200, (k) Arco_Valentino_Dense_vox12, (l) Facade_00009_vox12, (m) Egyptian_mask_vox12, (n) Shiva_00035_vox12, (o) Frog_00067_vox12, (p) Staue_Klimt_vox12.

then calculated by

$$PSNR_g = 10log_{10}\Big(\Omega^2/d_{s\_rms}\big(\boldsymbol{P}_{ori,g}, \boldsymbol{P}_{rec,g}\big)^2\Big), \tag{8}$$

$$PSNR_c = 10log_{10}\Big(255^2/d_{s\_rms}\big(\boldsymbol{P}_{ori,c}, \boldsymbol{P}_{rec,c}\big)^2\Big), \tag{9}$$

where

$$d_{s\_rms}\big(\boldsymbol{P}_{ori,\varphi}, \boldsymbol{P}_{rec,\varphi}\big) = max\big(d_{rms}\big(\boldsymbol{P}_{ori,\varphi}, \boldsymbol{P}_{rec,\varphi}\big), d_{rms}\big(\boldsymbol{P}_{rec,\varphi}, \boldsymbol{P}_{ori,\varphi}\big)\big), \tag{10}$$

$$d_{rms}(\boldsymbol{P}_A, \boldsymbol{P}_B) = \sqrt{\frac{1}{k}\sum_{\boldsymbol{p}\in\boldsymbol{P}_A} \|\boldsymbol{p}_\varphi - \boldsymbol{p}_{B\_nn,\varphi}\|_2^2}, \tag{11}$$

$$\Omega = max((x_{max} - x_{min}), (y_{max} - y_{min}), (z_{max} - z_{min})), \tag{12}$$

$\boldsymbol{P}_{ori}$ and $\boldsymbol{P}_{rec}$ are original and reconstructed point clouds, the subscript $\varphi \in \{g, c\}$ denotes the geometry or color information, $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ are two arbitrary point clouds that corresponds to $\boldsymbol{P}_{ori,\varphi}$ and $\boldsymbol{P}_{rec,\varphi}$, k is the number of points in $\boldsymbol{P}_A$, $\boldsymbol{p}$ denotes a point in $\boldsymbol{P}_A$, $\boldsymbol{P}_{B\_nn}$ is the nearest neighbor of $\boldsymbol{p}$ in $\boldsymbol{P}_B$, x_{max}, x_{min}, y_{max}, y_{min}, z_{max}, z_{min} are the maximum and the minimum coordinates along with the x, y, and z axis. Similar to traditional image/video compression, the rate-distortion curves and the BD-PSNR (resp. BD-BR) [59] were used to do qualitative and quantitative comparisons respectively. The performance of TMC13 and TMC2 are compared under 3 cases:

Case 1, both geometry and color are compressed losslessly;

Case 2, geometry is compressed losslessly, while the color is compressed lossily;

Case 3, both geometry and color are compressed lossily.

To conduct the experiments, a computer equipped with Intel Core i7 8700K CPU (3.7GHz frequency), 64GB memory, and windows 10 operating system was used. The main software configurations of the three cases are shown in Table I. Note that LODC is the level of detail count, PQS indicates position quantization scale, DBODL defines the difference between octree depth and level, which can be used for triangle surface approximate. RQS represents RAHT color quantization steps (both luma and chroma), LTQS refers to quantization steps (both luma and chroma) of LT, and PTQS means color quantization steps (both luma and chroma) of predict transform (refers to the LoD-based encoder without LT). GQP expresses geometry quantization steps, while TQP denotes the texture (color) quantization step of TMC2. Other parameters were set according to the parameter encoding files provided in the TMC13 and TMC2 software packages.

A. Results of Case 1

In this case, the LoD-based encoder without LT (denoted as **LoD encoder w/o LT**) of TMC13 was used. Because the TMC2 V5 has some bugs in lossless compression for large-scale sparse point cloud (i.e., Fig. 9 (k)-(p)), TMC2 V6 was used to perform lossless compression for these point clouds. The corresponding coding bits per point (*bpp*) of geometry and color are given in Table II. In this case, the **LoD encoder w/o LT** in TMC13 was used for the lossless compression of color information. We can see that, for the geometry information, the

TABLE I
MAIN CODING PARAMETERS FOR TMC13 AND TMC2

Coding parameters	TMC13						TMC2	
	LODC	PQS	DBODL	RQS	LTQS	PTQS	GQP	CQP
Case1	8	1	—	—	—	0	—	—
Case2	8	1	—	2 - 26	2 - 26	2 - 32	—	—
Case3	8	0.9, 0.7, 0.4, 0.2	1, 2, 3, 4	2 - 32	2 - 32	2 - 32	6 - 40	2 - 40

TABLE II
RATE AND CODING TIME COMPARISONS OF TMC13 AND TMC2 FOR CASE 1

Datasets	TMC13				TMC2				
	Geometry rate (bpp)	Color rate (bpp)	Total rate (bpp)	Coding Time (s)	Occupancy map rate (bpp)	Geometry rate (bpp)	Color rate (bpp)	Total rate (bpp)	Coding Time (s)
Andrew	1.17	15.19	16.36	2.30	0.19	1.67	13.99	15.74	50.98
Ricardo	1.08	8.13	9.22	1.66	0.14	1.43	8.04	9.55	38.24
David	1.14	10.13	11.27	2.64	0.15	1.60	10.11	11.73	58.09
Phil	1.21	14.12	15.33	3.03	0.23	1.79	13.73	15.61	67.40
Sarah	1.14	6.88	8.02	2.40	0.19	1.62	6.92	8.64	51.16
Longdress_vox10_1300	1.05	14.08	15.12	7.10	0.09	1.69	13.75	15.20	177.87
Loot_vox10_1200	1.00	8.89	9.89	6.44	0.10	1.59	7.90	9.30	159.74
Redandblack_vox10_1550	1.13	10.27	11.40	6.13	0.16	1.81	11.54	13.25	161.09
Soldier_vox10_0690	1.06	11.09	12.15	8.73	0.11	1.82	8.91	10.45	219.87
Queen_0200	0.80	9.72	10.51	8.00	0.14	1.32	8.72	9.86	186.66
Arco_Valentino_Dense_vox12	9.86	21.05	30.92	14.91	6.17	31.34	29.62	67.13	33309.60
Facade_00009_vox12	7.25	14.97	22.23	15.83	3.69	13.31	27.09	44.09	13938.20
Egyptian_mask_vox12	12.58	11.31	23.89	2.80	9.93	43.14	34.14	87.21	4598.19
Shiva_00035_vox12	9.68	19.07	28.75	10.37	6.77	28.99	31.16	66.92	33465.60
Frog_00067_vox12	6.44	13.08	19.52	35.07	3.74	10.84	26.53	41.11	21472.60
Staue_Klimt_vox12	9.94	15.71	25.65	5.07	7.14	31.09	31.11	69.34	11143.30
Average	**4.16**	**12.73**	**16.89**	**8.28**	**2.43**	**10.94**	**17.70**	**30.95**	**7443.66**

bpp of TMC13 is lower than TMC2, i.e., the average geometry *bpp* of TMC13 and TMC2 are 4.16 and 10.81 respectively. Similar results appear in the performance of color compression and coding complexity. The color *bpp* and coding time of TMC13 is also lower than TMC2.

Accordingly, TMC13 is obviously more suitable for lossless compression.

B. Results of Case 2

Because the TMC2 cannot work in this case, we only compared the RAHT-based encoder (denoted as **RAHT encoder**), the **LoD encoder w/o LT** and the LoD based encoder with LT (denoted as **LoD encoder with LT**) of TMC13. Fig. 10 shows the rate-PSNR curves of color information for the three encoders. We can see that **LoD encoder with LT** is the best when the rate is low, while the **LoD encoder w/o LT** is usually the best when the rate is high. The performance of the **RAHT encoder** is in between with or lower than **LoD encoder with LT** and **LoD encoder w/o LT**, for different point clouds.

Due to the limitation of the paper space, we put the detailed rate distortion data in the additional **Appendix**.[1] The detailed comparisons of the three methods can be found in **Appendix A**, whereas Table III shows the BD-PSNR and BD-BR when using the **RAHT encoder** as the benchmark. From Table III, we can see that the performance of the Y component of **LoD encoder w/o LT** is better than other two methods, i.e., an average 0.84 dB BD-PSNR can be achieved by **LoD encoder w/o LT** when comparing it to **RAHT encoder**, while the corresponding BD-PSNR of the **LoD encoder with LT** is 0.77 dB higher than that of the **RAHT encoder**.

Besides, we also compared the performance of U and V components as shown in Table III. Because the local correlation of the U and V components are much higher than that of the Y component, the performance of the **LoD encoder with LT** is the best. We can also find that the performance of the **LoD encoder w/o LT** is the worst. Therefore, we can conclude that both the RAHT and the LT can efficiently utilize the correlation of color information, but the performance of LT is the better in this case.

C. Results of Case 3

In this case, for the TMC13, the geometry can be lossily compressed by two approaches, i.e., the triangle surface approximation-based method, and the direct geometry quantization-based method, as described in 2 of Part A of Section IV.

1) Triangle Surface Approximation-Based Lossy Geometry Compression: In this case, **Appendix B** and **Appendix C** show the rate and the PSNRs of the TMC13 and TMC2 in detail. The rate-PSNR curves of the four methods are compared in Fig. 11, in which the *x*-axis defines the total bits of geometry and color, while the *y*-axis indicates the Y-PSNRs of the reconstructed point clouds. The average performance, i.e., BD-PSNR is compared in Table IV. We can see that the rate-PSNR curves of TMC2 is usually higher than those of the other methods. It should also be noted that the rate-PSNR

[1]The Appendix can be found in the additional appendix.

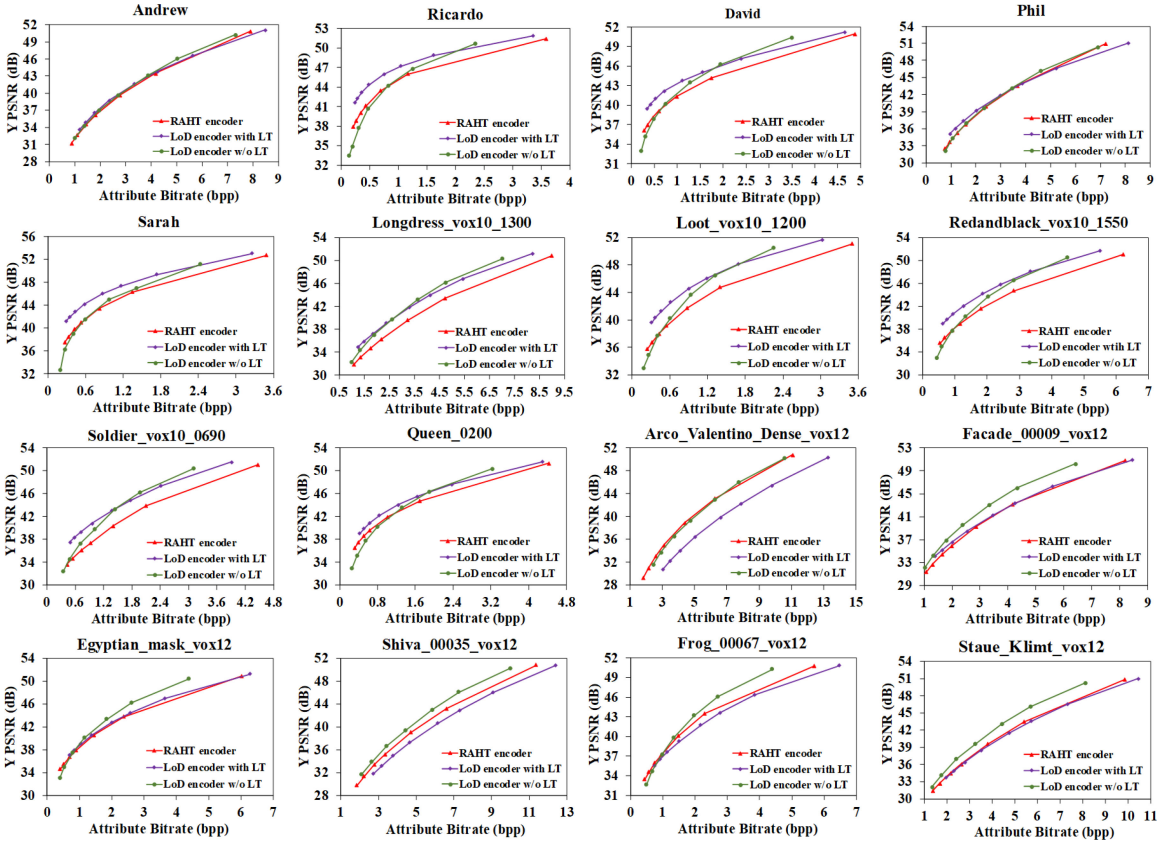


Fig. 10. Comparisons of rate-PSNR curves of case 2.

TABLE III
BD-PSNR AND BD-BR COMPARISONS OF LOD ENCODER W/O LT AND LOD ENCODER WITH LT FOR CASE 2, RAHT ENCODER IS THE BENCHMARK

Datasets	LoD encoder w/o LT						LoD encoder with LT					
	Y-BD-PSNR (dB)	BD-BR (%)	U-BD-PSNR (dB)	BD-BR (%)	V-BD-PSNR (dB)	BD-BR (%)	Y-BD-PSNR (dB)	BD-BR (%)	U-BD-PSNR (dB)	BD-BR (%)	V-BD-PSNR (dB)	BD-BR (%)
Andrew	**0.49**	**-4.62**	-1.18	17.85	-1.24	18.40	-0.05	0.65	-0.50	7.86	-0.55	7.92
Ricardo	-0.47	7.26	-4.64	228.77*	-5.63	137.18*	**2.36**	**-40.44**	1.14	-45.57	1.17	-34.96
David	0.46	-7.24	-4.60	127.18*	-3.73	94.06	**2.05**	**-37.29**	1.09	-38.14	1.03	-34.72
Phil	0.06	-0.74	-3.63	58.78	-3.03	45.93	**0.47**	**-5.41**	-0.24	6.03	-0.68	12.94
Sarah	0.18	-3.09	-2.86	59.13	-2.67	55.67	**2.43**	**-36.06**	1.88	-44.42	1.65	-36.64
Longdress_vox10_1300	2.22	-20.68	0.18	-1.26	0.22	-1.56	2.03	-19.34	**2.71**	**-33.00**	2.63	-31.56
Loot_vox10_1200	1.08	-14.27	-6.71	314.65*	-7.33	206.02*	**2.74**	**-37.81**	2.18	-59.91	2.27	-61.47
Redandblack_vox10_1550	0.86	-10.52	-2.23	26.13	-0.32	3.96	2.48	-32.77	2.23	-44.28	**2.63**	**-32.54**
Soldier_vox10_0690	2.22	-23.82	-5.96	178.44*	-6.41	175.79*	**3.01**	**-33.13**	2.50	-57.49	2.50	-57.73
Queen_0200	-0.21	3.77	-5.32	144.89*	-4.70	127.82*	**1.15**	**-20.10**	1.02	-27.84	1.12	-29.66
Arco_Valentino_Dense_vox12	**-0.07**	**0.55**	-0.37	6.23	-0.35	6.16	-3.93	37.34	-2.96	38.07	-3.06	37.02
Facade_00009_vox12	**2.09**	**-18.89**	-2.18	25.01	-1.05	11.46	0.22	-2.04	1.61	-28.39	1.01	-15.98
Egyptian_mask_vox12	**0.95**	**-13.17**	-3.01	49.81	-3.22	59.30	0.24	-3.71	0.61	-16.28	0.70	-19.71
Shiva_00035_vox12	**1.17**	**-9.73**	-0.44	2.70	-0.91	6.97	-1.57	14.91	-0.49	6.45	-0.30	4.76
Frog_00067_vox12	**0.63**	**-7.06**	-4.21	84.68	-6.17	340.76*	-0.99	15.65	0.55	-21.86	0.26	-17.79
Staue_Klimt_vox12	**1.72**	**-15.11**	0.36	-2.50	-0.22	3.05	-0.39	3.96	0.21	-2.17	0.41	-4.89
Average	**0.84**	**-8.58**	**-2.93**	**82.53**	**-2.92**	**80.69**	**0.77**	**-12.22**	**0.85**	**-22.56**	**0.80**	**-19.69**

Note that BD-BR* results in some unreliable results, and BD-PSNR may better reflect the gaps between rate distortion data.

curves of TMC2 for the sparse point clouds are not stable (even weird for *Shiva_00035_vox12* and *Staue_Klimt_vox12*). This reason maybe the sparsity and noise of the *Shiva_00035_vox12* and *Staue_Klimt_vox12*. In other words, TMC2 is not robust enough for large scale sparse and noise point clouds, although its average performance is better. In the meantime, we can also observe that the performance of the Y component of the **LoD encoder with LT** is better than that of the **LoD encoder w/o LT** and **RAHT encoder**. Strictly speaking, the triangle surface approximation used in the lossily geometry compression of TMC13 can be thought as a local up-sampling process. The color information will be inevitably distorted before compression. This maybe the reason why the objective performance of TMC13 is much lower than TMC2.

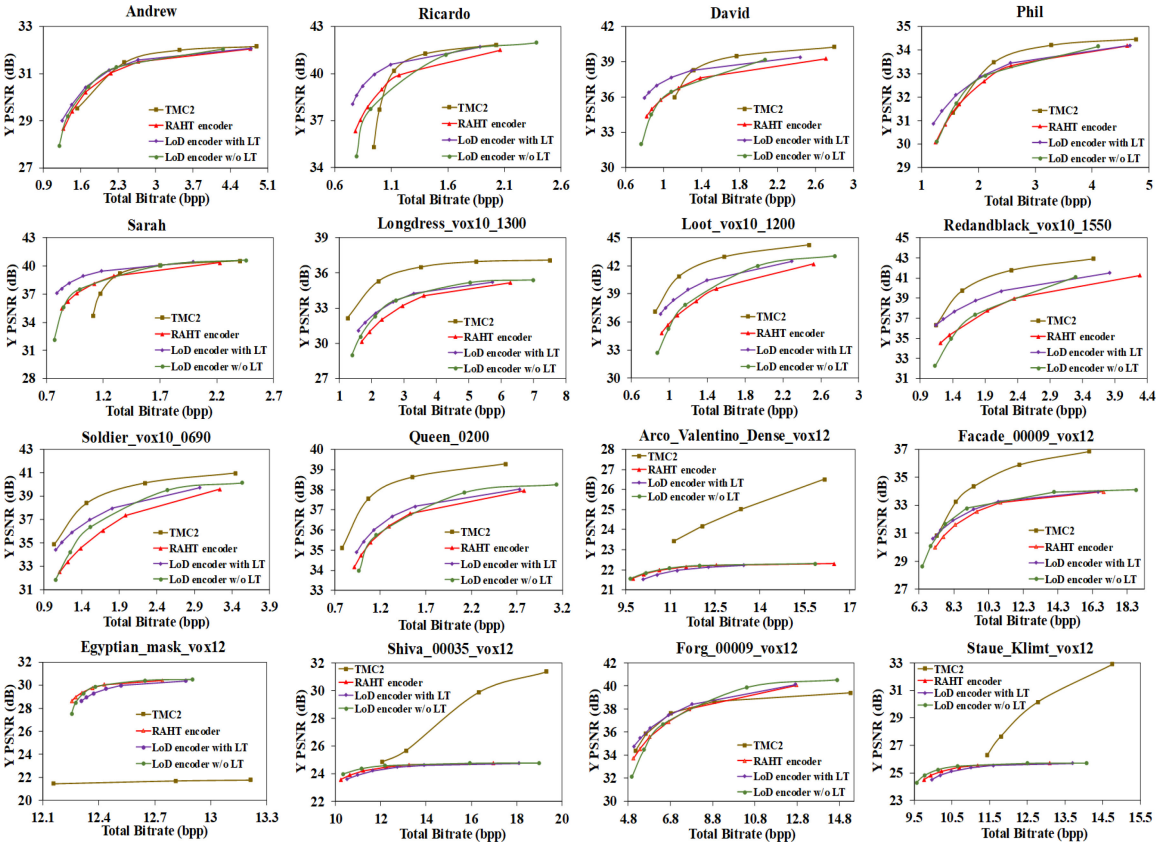


Fig. 11. Comparisons of rate-PSNR curves of case 3, in which the triangle surface approximation-based lossy geometry compression is used.

TABLE IV
BD-PSNR COMPARISONS OF CASE 3 (TRIANGLE SURFACE APPROXIMATION-BASED LOSSY GEOMETRY COMPRESSION), THE RESULTS OF THE RAHT ENCODER ARE USED AS THE BENCHMARK

Datasets	LOD encoder w/o LT			LOD encoder with LT			TMC2		
	Y-BD-PSNR (dB)	U-BD-PSNR (dB)	V-BD-PSNR (dB)	Y-BD-PSNR (dB)	U-BD-PSNR (dB)	V-BD-PSNR (dB)	Y-BD-PSNR (dB)	U-BD-PSNR (dB)	V-BD-PSNR (dB)
Andrew	0.08	-1.50	-1.96	0.07	**0.29**	0.24	0.12	-1.89	-1.96
Ricardo	0.11	-2.46	-3.10	0.73	**0.88**	0.76	-0.23	-1.34	-1.88
David	-0.50	-11.89	-7.90	0.68	**0.87**	0.84	0.72	-1.05	-1.11
Phil	0.09	-2.77	-1.68	0.14	0.29	0.09	**0.35**	-1.92	-2.19
Sarah	0.21	-1.79	-2.01	0.67	**1.39**	1.03	-0.14	-1.52	-2.48
Longdress_vox10_1300	0.74	-0.68	-0.56	0.85	2.20	1.92	**3.12**	1.25	0.82
Loot_vox10_1200	0.65	-4.92	-7.30	1.15	1.91	1.86	**2.80**	0.57	0.49
Redandblack_vox10_1550	0.37	-1.60	-0.19	1.12	1.66	0.99	**2.81**	0.28	0.02
Soldier_vox10_0690	0.84	-4.98	-4.18	1.00	2.12	2.12	**2.16**	0.05	0.03
Queen_0200	0.25	-2.49	-1.95	0.18	0.31	0.47	**1.67**	-5.61	-7.58
Arco_Valentino_Dense_vox12	0.02	-0.69	-0.40	-0.05	-0.56	-0.54	**2.78**	-0.99	-1.05
Facade_00009_vox12	0.31	-1.82	-0.85	0.10	1.28	0.92	**2.31**	-0.01	-0.67
Egyptian_mask_vox12	**0.11**	-1.06	-1.76	-0.23	-0.56	-0.52	-8.48	-8.80	-14.54
Shiva_00035_vox12	0.06	0.00	-0.10	-0.06	-0.27	-0.21	**2.72**	-1.68	-1.74
Frog_00067_vox12	0.22	-3.66	-4.42	0.18	1.15	**0.82**	-0.22	-0.47	-0.47
Staue_Klimt_vox12	0.08	0.06	-0.13	-0.08	-0.34	-0.38	**3.17**	-0.29	-0.68
Average	**0.23**	**-2.64**	**-2.40**	**0.40**	**0.79**	**0.65**	**0.98**	**-1.46**	**-2.19**

However, as shown in Fig. 12 and Fig. 13, the gap between TMC13 and TMC2 is not significant when comparing the subjective quality under the similar rate. Specially, we can observe that *David*'s face rebuilt by TMC13 is better than TMC2, but TMC2 works well on the upper body of *David* in Fig. 13. The right shoulder of *Phil* gets serious damage by the encoding of TMC2. Interestingly, for *Loot_vox10_1200*, and *Redandblack_vox10_1550*, we can hardly see the difference in subjective quality among TMC13 and TMC2.

Moreover, detailed quantitative comparisons of the U and V components are shown in Table IV. It can be observed that the average performance of the **LoD encoder with LT** is significantly better than the other methods, i.e., the average U-BD-PSNR and V-BD-PSNR of the **LoD encoder with**

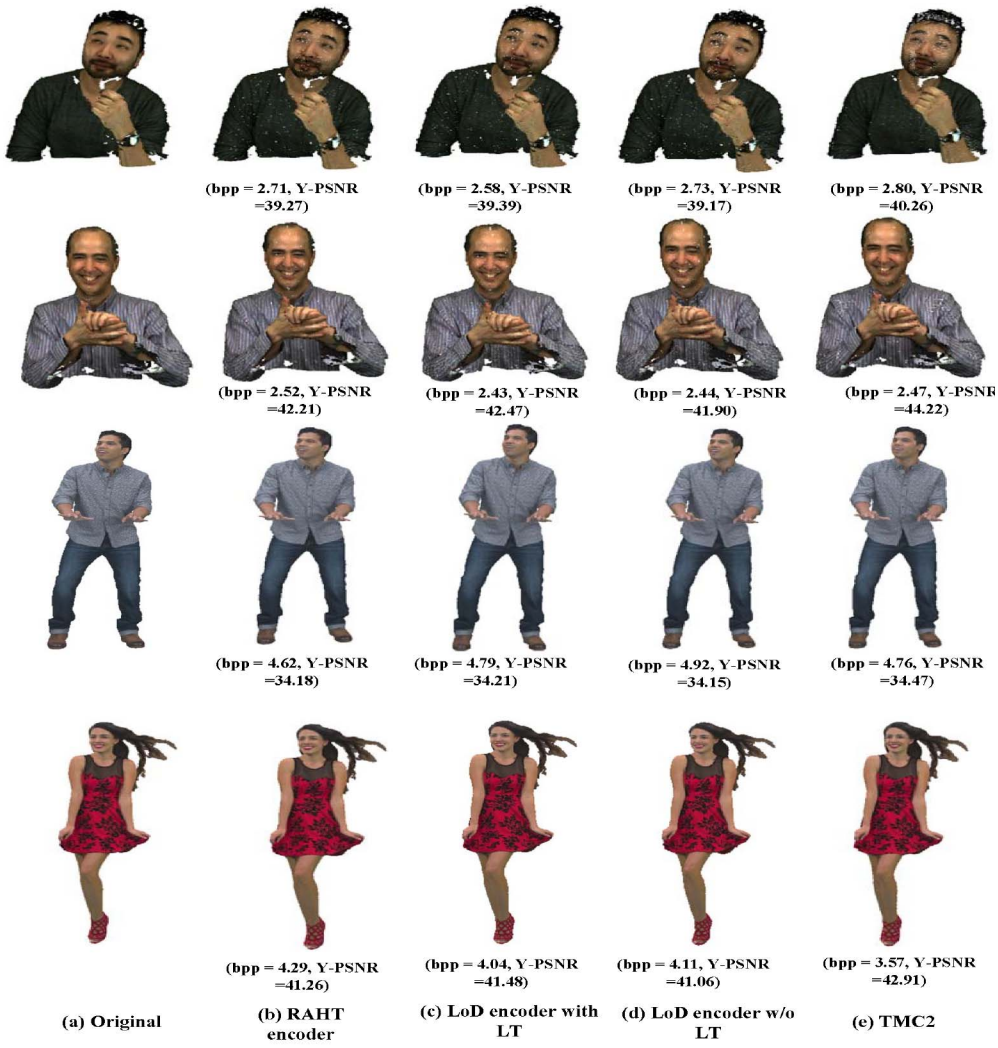


Fig. 12. Subjective quality comparisons of case 3, in which the triangle surface approximation-based lossy geometry compression is used.

LT are 0.79 and 0.65 dB higher than that of the **RAHT encoder**, which is similar to the results of case 2. Since the lossy geometry compression in TMC13 deteriorates the color information among points, (especially for sparse point clouds), the **LoD encoder with LT** cannot work well for the sparse point clouds, as we can see that it exhibits a bad BD-PSNR for sparse point clouds in Table IV, i.e., *Arco_Valentino_Dense_vox12*, *Egyptian_mask_vox12*, *Shiva_00035_vox12*, *Staue_Klimt_vox12*.

2) Direct Geometry Quantization-Based Lossy Geometry Compression: In this case, **Appendix D** shows the rate and the PSNR data for three methods of TMC13 in detail. For the Y component, the rate-PSNR performance of different methods are compared in Fig. 14 where the *x*-axis refers to the total rate of geometry and color, while, the *y*-axis denotes the Y-PSNRs of reconstructed point cloud. The detailed BD-PSNR is compared in Table V. As we can see in Table V that **LoD encoder w/o LT** has the best coding efficiency, i.e., an average 0.70 dB BD-PSNR can be achieved compared to the **RAHT encoder**. However, the coding performance of TMC2 is terrible. The main reason maybe that TMC2 requires to generate a large 2D plane for large scale sparse point clouds (i.e., Fig. 9 (k)–(p)), and thus the correlation of the projected points is broken. Another possible reason is the noise in these point

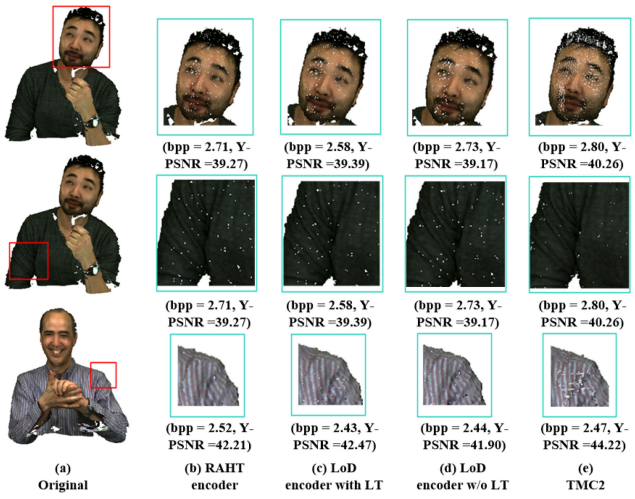


Fig. 13. Detailed subjective quality comparisons of *David* and *Phil*.

TABLE V
BD-PSNR COMPARISONS OF CASE 3 (DIRECT GEOMETRY QUANTIZATION-BASED LOSSY GEOMETRY COMPRESSION), THE RESULTS OF THE RAHT ENCODER ARE USED AS THE BENCHMARK

Datasets	LOD encoder w/o LT			LOD encoder with LT			TMC2		
	Y-BD-PSNR (dB)	U-BD-PSNR (dB)	V-BD-PSNR (dB)	Y-BD-PSNR (dB)	U-BD-PSNR (dB)	V-BD-PSNR (dB)	Y-BD-PSNR (dB)	U-BD-PSNR (dB)	V-BD-PSNR (dB)
Andrew	0.19	-1.20	-1.24	0.21	**0.64**	0.58	-1.10	-1.70	-1.84
Ricardo	-0.05	-5.51	-2.63	**2.09**	1.77	1.89	0.96	0.66	0.18
David	0.45	-3.11	-2.88	1.13	**1.23**	1.18	0.44	-0.45	-0.79
Phil	0.28	-1.65	-0.84	0.37	**0.88**	0.66	-0.59	-1.38	-1.88
Sarah	0.52	-1.31	-1.15	0.76	**1.47**	1.14	-1.28	-0.84	-2.47
Longdress_vox10_1300	0.83	-0.17	-0.13	0.83	2.20	1.86	**2.40**	0.80	0.35
Loot_vox10_1200	0.75	-6.80	-7.82	2.12	2.14	2.26	**3.49**	0.92	0.93
Redandblack_vox10_1550	0.82	-0.39	0.18	1.13	1.76	0.87	**1.93**	0.14	-1.07
Soldier_vox10_0690	1.42	-5.09	-5.54	1.50	2.42	2.35	**1.81**	0.41	0.35
Queen_0200	0.16	-4.01	-3.40	0.64	0.90	0.87	-0.38	-6.10	-8.26
Arco_Valentino_Dense_vox12	**-0.20**	-1.52	-1.30	-2.94	-2.06	-2.11	-11.69	-3.59	-3.59
Facade_00009_vox12	**1.89**	-2.96	-1.67	-0.50	1.41	0.65	-3.49	-0.32	-0.94
Egyptian_mask_vox12	**0.79**	-3.67	-3.84	0.28	0.64	0.73	-17.80	-11.34	-16.53
Shiva_00035_vox12	**1.10**	-0.03	-0.44	-1.64	-0.58	-0.37	-11.46	-5.33	-4.41
Frog_00067_vox12	**0.61**	-4.09	-5.99	-0.63	0.61	0.29	-1.62	0.13	-0.14
Staue_Klimt_vox12	**1.66**	0.02	-0.58	-0.30	0.43	0.63	-7.80	-3.92	-3.31
Average	**0.70**	**-2.59**	**-2.45**	**0.32**	**0.99**	**0.84**	**-2.89**	**-1.99**	**-2.71**

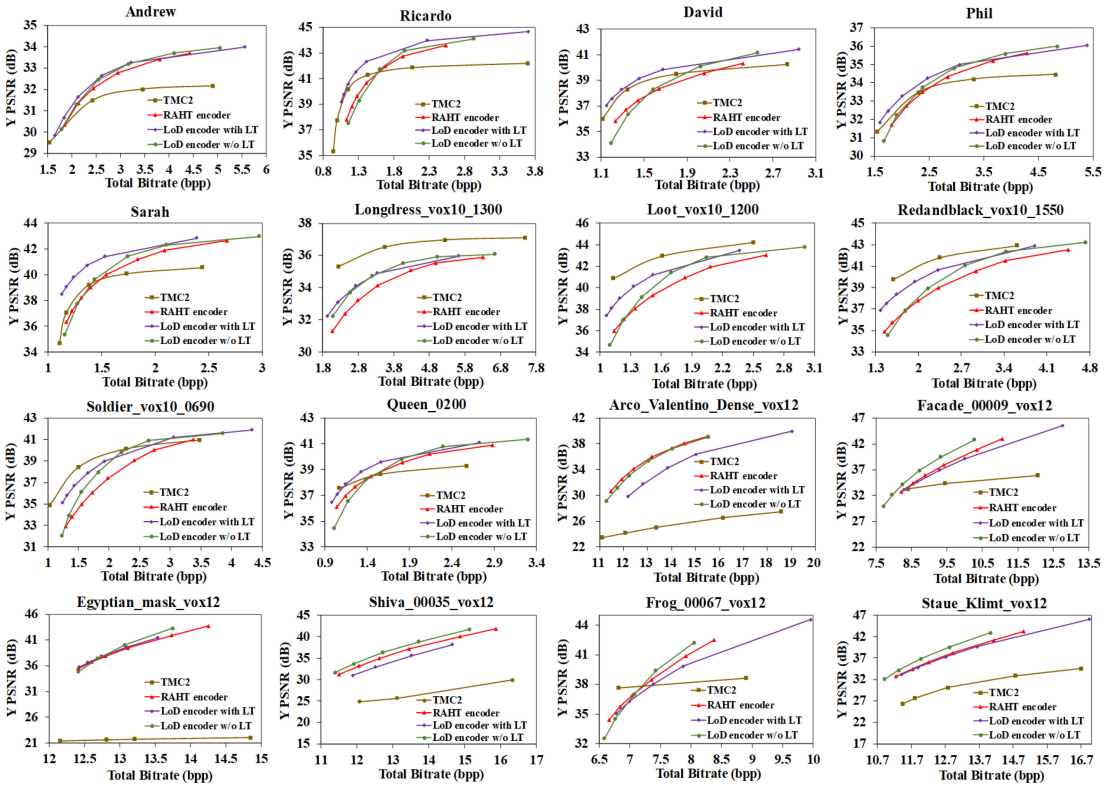


Fig. 14. Rate-PSNR curves comparison of case 3, in which the direct geometry quantization-based lossy geometry compression is used.

clouds. For the dense point clouds (i.e., Fig. 9 (a)–(j)), the **LoD encoder with LT** is the better when comparing to the **RAHT encoder** and the **LoD encoder w/o LT**, whereas, for the sparse point clouds, the performance of the **LoD encoder w/o LT** is better.

For the U and V components, compared to the **RAHT encoder**, an average 0.99, 0.84 dB and the maximum 2.42, 2.35 dB BD-PSNR can be achieved by the **LoD encoder with LT**, yet the **LoD encoder w/o LT** and the TMC2 is not better than the **RAHT encoder**.

3) Geometry Compression Results Comparison: In this case, we compared the rate-distortion performance of lossy geometry compression among three lossy geometry methods (i.e., the triangle surface approximation-based (**TSA**) method, the direct geometry quantization-based (**DGQ**) method and TMC2 lossy geometry compression method). The rate-PSNR curves are showed in Fig. 15, in which the *x*-axis denotes the bits of geometry, while, the *y*-axis represents the Geometry-PSNR of the reconstructed point clouds. Corresponding BD-PSNRs are compared in Table VI, when **TSA** method is used

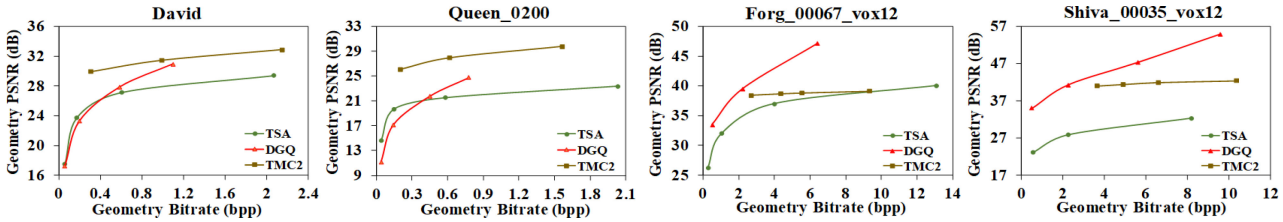


Fig. 15. Rate-PSNR curves comparison of lossy geometry compression.

TABLE VI
BD-PSNR COMPARISONS OF LOSSY GEOMETRY COMPRESSION, THE RESULTS OF THE TSA METHOD ARE USED AS THE BENCHMARK

Datasets	DGQ BD-PSNR (dB)	TMC2 BD-PSNR (dB)
Andrew	0.12	3.14
Ricardo	-0.89	3.50
David	-0.13	3.73
Phil	-0.18	2.86
Sarah	0.13	3.37
Longdress_vox10_1300	-1.81	5.96
Loot_vox10_1200	-1.61	6.13
Redandblack_vox10_1550	-1.48	4.81
Soldier_vox10_0690	-1.51	5.70
Queen_0200	-1.45	6.29
Arco_Valentino_Dense_vox12	17.44	7.65
Facade_00009_vox12	8.51	2.85
Egyptian_mask_vox12	26.34	10.61
Shiva_00035_vox12	14.58	11.19
Frog_00067_vox12	5.05	1.11
Staue_Klimt_vox12	15.40	12.61
Average	**4.91**	**5.72**

TABLE VII
THE COMPARISONS OF TIME COMPLEXITY FOR CASE 2

Datasets	RAHT encoder	Lod encoder w/o LT	Lod encoder with LT
	Coding times (s)		
Andrew	**2.17**	2.28	2.28
Ricardo	**1.57**	1.66	1.67
David	**2.51**	2.71	2.81
Phil	**2.85**	3.04	3.05
Sarah	**2.30**	2.52	2.44
Longdress_vox10_1300	**6.70**	7.28	7.13
Loot_vox10_1200	**6.29**	6.65	6.69
Redandblack_vox10_1550	**5.94**	6.25	6.27
Soldier_vox10_0690	**8.72**	9.01	9.13
Queen_vox10_0200	**7.74**	8.25	8.28
Arco_Valentino_Dense_vox12	**15.51**	16.84	16.81
Facade_00009_vox12	**15.53**	16.62	16.75
Egyptian_mask_vox12	**2.61**	2.89	2.94
Shiva_00035_vox12	**10.03**	11.13	11.05
Frog_00067_vox12	**34.88**	37.19	37.46
Staue_Klimt_vox12	**4.84**	5.28	5.34
Average	**8.14**	**8.72**	**8.76**

as benchmark. We can observe that TMC2 is better than the other two methods, i.e., an average 5.72 dB BD-PSNR can be achieved by TMC2 compared to the **TSA** method. The reason is that the **TSA** method cannot accurately establish the topological surfaces, and thus cannot approximate the original 3D point cloud. Although overall performance of the **DGQ** method is not the optimal, it is worth pointing out that the **DGQ** method exhibits an excellent performance among three lossy geometry compression methods for sparse point clouds.

Therefore, we can conclude that TMC2 and the **LoD encoder with LT** can work well for dense point clouds, but not suitable for sparse point clouds. Although the average performance of the **RAHT encoder** is not the best, it is robust for all the tested point clouds. Compared with lossy geometry compression of **DGQ** and **TSA**, TMC2 exhibits a better coding performance. Note that the performance of **DGQ** method is much better than **TSA** method, this is why the overall objective coding performance of case 3.1 is not as good as case 3.2 for TMC13.

D. Complexity Analysis

Regarding to the complexity, we can see that the complexity of TMC2 is much larger than the other methods, as shown in Table II, VII, and VIII. From Table II, the average coding time of TMC13 and TMC2 are 8.28 and 7443.66 seconds respectively, when the geometry and color are losslessly compressed. The high complexity of TMC2 is mainly due to the complexity of the HEVC encoder. From Table VII, we can observe that the **RAHT encoder** has the smallest coding time compared to the **LoD encoder with LT** and the **LoD encoder w/o LT** in case 2. For case 3 in Table VIII, it can also be observed that the encoding time of TMC2 is significantly larger than that of all the other methods. Compared with the **TSA** method, the **DGQ** method has lower complexity in case 3, The main reason is that the triangle surface reconstruction procedure used in the lossy compression of geometry of TMC13 is time consuming.

E. Summary

To sum up, we have the following conclusions: (a) the rate distortion performance of TMC2 is the best on average for dense point clouds, especially for lossy compression, but it is not suitable for large scale sparse and noisy point clouds; (b) among the three encoders in TMC13, i.e., the **RAHT encoder**, the **LoD encoder w/o LT**, and the **LoD encoder with LT**, the rate distortion performance of the **LoD encoder with LT** is usually the best especially when the *bpp* is low, nevertheless, it is not suitable for compressing sparse point clouds; (c) among all the encoders, the time complexity of the **RAHT encoder** is the lowest, whereas, that of TMC2 is the highest; (d) the point-to-point quality metric is not efficient to evaluate the subjective quality of the reconstructed point clouds.

TABLE VIII
THE COMPARISONS OF TIME COMPLEXITY FOR CASE 3

Datasets	Triangle surface approximate			Direct geometry quantization			TMC2
	RAHT encoder	LoD encoder w/o LT	LoD encoder with LT	RAHT encoder	LoD encoder w/o LT	LoD encoder with LT	Coding Time (s)
	Coding Times (s)			Coding Times (s)			
Andrew	2.16	2.25	2.25	**1.83**	1.93	1.94	60.03
Ricardo	31.03	31.87	31.79	**15.41**	16.64	16.91	64.27
David	2.50	2.63	2.64	**2.13**	2.29	2.28	61.11
Phil	3.39	3.41	3.49	**2.59**	2.90	2.92	67.51
Sarah	46.48	48.88	48.61	**15.30**	16.54	16.58	63.82
Longdress_vox10_1300	120.06	125.30	126.85	**34.38**	37.11	37.20	159.19
Loot_vox10_1200	7.59	7.94	7.92	**5.80**	6.12	6.15	162.63
Redandblack_vox10_1550	6.97	7.24	7.28	**5.43**	5.74	5.71	165.00
Soldier_vox10_0690	2.87	3.00	3.01	**2.45**	2.56	2.60	202.92
Queen_vox10_0200	7.80	8.14	8.13	**6.57**	6.98	7.02	189.36
Arco_Valentino_Dense_vox12	6.66	6.96	6.97	**5.08**	5.47	5.44	2254.24
Facade_00009_vox12	1.55	1.61	1.64	**1.38**	1.40	1.42	1366.22
Egyptian_mask_vox12	2.28	2.37	2.42	**1.95**	2.04	2.06	3255.60
Shiva_00035_vox12	20.27	21.33	21.35	**9.97**	10.98	11.04	1963.06
Frog_00067_vox12	9.49	9.91	10.09	**7.33**	7.78	7.80	1593.47
Staue_Klimt_vox12	8.48	8.77	8.78	**4.85**	5.29	5.37	835.25
Average	**17.47**	**18.22**	**18.33**	**7.65**	**8.23**	**8.28**	**778.98**

VI. CONCLUSION AND FUTURE WORKS

In this paper, we reviewed some basic technologies for point cloud compression as well as the TMC13 (G-PCC) and TMC2 (V-PCC) encoder proposed by MPEG PCC group in detail. At the same time, the performances of the two encoders were compared comprehensively in terms of objective quality (i.e., rate distortion performance), subjective quality and complexity. By observing the comparison results, we concluded that the **RAHT encoder** is robust for all types of point clouds, while the **LoD encoder w/o LT** is more efficient in high bit rates, especially for large scale sparse point clouds. Compared to the **RAHT encoder** and the **LoD encoder w/o LT**, the **LoD encoder with LT** has a better performance for dense point clouds, especially in lower bit rates. The TMC2 encoder can achieve the best rate distortion performance on average among all compression settings (especially for lossy geometry and lossy color compression) for dense point clouds. However, it is not suitable for large scale sparse point clouds. The reason maybe that TMC2 requires to generate a large 2D plane for large scale sparse point clouds, and thus, the correlation of the projected points is broken. Another possible reason is the noise in these point clouds. Meanwhile, its time complexity should be optimized further for real-time applications. By introducing and comparing the key technologies of the 3D point cloud encoders proposed by MPEG, we want more researchers to understand the necessity and the current progress of 3D point cloud compression, and encourage them to participate in the research of point cloud compression.

Although great achievements of point cloud compression have been made in recent years, there are still a large space to further improve the compression efficiency for point clouds. For example, the attribute prediction tools of TMC13 is very limited, and the time complexity of TMC2 is very high. Besides, effective motion prediction methods, efficient entropy encoding methods, rate distortion optimization methods, optimal bit allocation, and rate control technologies for point cloud compression should also be investigated in the near future.

REFERENCES

[1] S. Schwarz *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[2] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Compressing 3-D human motions via keyframe-based geometry videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 1, pp. 51–62, Jan. 2015.

[3] J. Hou, L.-P. Chau, M. Zhang, N. Magnenat-Thalmann, and Y. He, "A highly efficient compression framework for time-varying 3-D facial expressions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 9, pp. 1541–1553, Sep. 2014.

[4] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proc. High Perform. Graph. Conf.*, Anaheim, CA, USA, 2013, pp. 73–79.

[5] J. Hou, L.-P. Chau, Y. He, M. Zhang, and N. Magnenat-Thalmann, "Rate-distortion model based bit allocation for 3-D facial compression using geometry video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 9, pp. 1537–1541, Sep. 2013.

[6] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Sparse low-rank matrix approximation for data compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 5, pp. 1043–1054, May 2017.

[7] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Human motion capture data tailored transform coding," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 7, pp. 848–859, Jun. 2015.

[8] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Low-latency compression of mocap data using learned spatial decorrelation transform," *Comput.-Aided Geometric Design*, vol. 43, pp. 211–225, Mar. 2016.

[9] M. Kaess, R. C. Arkin, and J. Rossignac, "Compact encoding of robot-generated 3D maps for efficient wireless transmission," in *Proc. IEEE Int. Conf. Adv. Robot. (ICAR)*, Coimbra, Portugal, Jun. 2003, pp. 324–331.

[10] K. Kohira and H. Masuda, "Point-cloud compression for vehicle-based mobile mapping systems using portable network graphics," *ISPRS Ann. Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. 6, pp. 99–106, Sep. 2017.

[11] P. Caillet and Y. Dupuis, "Efficient LiDAR data compression for embedded V2I or V2V data handling," *arXiv preprint arXiv:1904.05649v1*, Apr. 2019.

[12] *8i–Real Human Holograms for AR, VR and MR*. Accessed: Dec. 20, 2019. [Online]. Available: http://8i.com/

[13] M. Isenburg, "LASzip: Lossless compression of LiDAR data," *Photogrammetric Eng. Remote Sens.*, vol. 79, no. 2, pp. 209–217, Feb. 2013.

[14] *LASzip-Free and Lossless LiDAR Compression*. Accessed: Dec. 20, 2019. [Online]. Available: https://laszip.org/

[15] P. A. Chou, O. Nakagami, and E. S. Jang, *Point Cloud Compression Test Model for Category 1 V0*, document N17223, ISO/IEC JTC1/SC29/WG11 MPEG, Macau, China, Oct. 2017.

[16] K. Mammou, *PCC Test Model Category 2 V0*, document N17248, ISO/IEC JTC1/SC29/WG11 MPEG, Macau, China, Oct. 2017.

[17] K. Mammou, *PCC Test Model Category 3 V0*, document N17249, ISO/IEC JTC1/SC29/WG11 MPEG, Macau, China, Oct. 2017.

[18] K. Mammou and P. A. Chou, *PCC Test Model Category 13 V2*, document N17519, ISO/IEC JTC1/SC29/WG11 MPEG, San Diego, CA, USA, Apr. 2018.

[19] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. Symp. Point Based Graph.*, Boston, MA, USA, Jul. 2006, pp. 111–121.

[20] J. Elseberg, D. Borrmann, and A. Nüchter, "One billion points in the cloud—An octree for efficient processing of 3D laser scans," *ISPRS J. Photogrammetry Remote Sens.*, vol. 76, pp. 76–88, Feb. 2015.

[21] 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./Apr. 2008.

[22] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," in *Proc. Symp. Point Based Graph.*, Zürich, Switzerland, Jun. 2004, pp. 103–112.

[23] J.-K. Ahn, K.-Y. Lee, J.-Y. Sim, and C.-S. Kim, "Large-scale 3D point cloud compression using adaptive radial distance prediction in hybrid coordinate domains," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 3, pp. 422–434, Apr. 2015.

[24] P. de Oliveira Rente, C. Brites, J. M. Ascenso, and F. Pereira, "Graph-based static 3D point clouds geometry coding," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 284–299, Feb. 2019.

[25] 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.

[26] H. Houshiar and A. Nüchter, "3D point cloud compression using conventional image compression for efficient data transmission," in *Proc. XXV Int. Conf. Inf. Commun. Autom. Technol. (ICAT)*, Sarajevo, Bosnia and Herzegovina, Oct. 2015, pp. 1–8.

[27] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 2066–2070.

[28] R. A. Cohen, D. Tian, and A. Vetro, "Attribute compression for sparse point clouds using graph transforms," in *Proc. IEEE Int. Conf. Image Process.*, Phoenix, AZ, USA, Sep. 2016, pp. 1374–1378.

[29] Y. Shao, Z. Zhang, Z. Li, K. Fan, and G. Li, "Attribute compression of 3D point clouds using Laplacian sparsity optimized graph transform," in *Proc. IEEE Vis. Commun. Image Process.*, St. Petersburg, FL, USA, Dec. 2017, pp. 1–4.

[30] Y. Shao, Q. Zhang, G. Li, and Z. Li, "Hybrid point cloud attribute compression using slice-based layered structure and block-based intra prediction," in *Proc. ACM Multimedia*, 2018, pp. 1199–1207. [Online]. Available: https://arxiv.org/abs/1804.10783

[31] 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.

[32] J. Hou, L.-P. Chau, Y. He, and P. A. Chou, "Sparse representation for colors of 3D point cloud via virtual adaptive sampling," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, New Orleans, LA, USA, Mar. 2017, pp. 2926–2930.

[33] S. Gu, J. Hou, H. Zeng, H. Yuan, and K.-K. Ma, "3D point cloud attribute compression using geometry-guided sparse representation," *IEEE Trans. Image Process.*, vol. 29, pp. 796–808, Aug. 2019.

[34] A. Anis, P. A. Chou, and A. Ortega, "Compression of dynamic 3D point clouds using subdivisional meshes and graph wavelet transforms," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Shanghai, China, Mar. 2016, pp. 6360–6364.

[35] 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.

[36] 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.

[37] 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.

[38] Q. Liu, H. Yuan, J. Hou, H. Liu, and R. Hamzaoui, "Model-based encoding parameter optimization for 3D point cloud compression," in *Proc. Asia Pac. Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Honolulu, HI, USA, Nov. 2018, pp. 1981–1986.

[39] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter, "Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration," *J. Softw. Eng. Robot.*, vol. 3, pp. 2–12, Mar. 2012.

[40] J. Schauer and A. Nüchter, "Collision detection between point clouds using an efficient *k*-d tree implementation," *Adv. Eng. Informat.*, vol. 29, no. 3, pp. 440–458, Aug. 2015.

[41] K. Mammou, P. A. Chou, D. Flynn, M. Krivokuča, O. Nakagami, and T. Sugio, *G-PCC Codec Description V2*, document N18189, ISO/IEC JTC1/SC29/WG11 MPEG, Marrakesh, Morocco, Jan. 2019.

[42] S. Lasserre and D. Flyn, *Inference of a Mode Using Point Location Direct Coding in TMC3*, document m42239, ISO/IEC JTC1/SC29/WG11 MPEG, Gwangju, South Korea, Jan. 2018.

[43] S. Lasserre and D. Flyn, *Neighbour-Dependent Entropy Coding of Occupancy Patterns in TMC3*, document m42238, ISO/IEC JTC1/SC29/WG11 MPEG, Gwangju, South Korea, Jan. 2018.

[44] S. Lasserre and D. Flyn, *Intra Mode for Geometry Coding in TMC3*, document m43600, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[45] K. Mammou, J. Kim, V. Valentin, F. Robinet, A. Tourapis, and Y. Su, *Efficient Implementation of the Lifting Scheme in TMC13*, document m43781, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[46] V. Zakharchenko, *V-PCC Codec Description*, document N18190, ISO/IEC JTC1/SC29/WG11 MPEG, Marrakesh, Morocco, Jan. 2019.

[47] H. Najaf-Zadeh and M. Budagavi, *Improved Point Cloud Compression Efficiency in TMC2 via Color Smoothing of Point Cloud Prior to Texture Video Generation*, document m43721, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[48] D. Graziosi, *TMC2 Patch Flexible Orientation*, document m43680, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[49] D. Graziosi, *TMC2 Optimal Texture Packing*, document m43681, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[50] N. Dawar, H. Najaf-Zadeh, R. Joshi, and M. Budagavi, *PCC TMC2 Interleaving in Geometry and Texture Layers*, document m43723, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[51] A. Vosoughi, D. Graziosi, and A. Tabatabai, *Point Cloud Color Processing*, document m42733, ISO/IEC JTC1/SC29/WG11 MPEG, San Diego, CA, USA, Apr. 2018.

[52] C. Guede, J. Ricard, and J. Llach, *Spatially Adaptive Geometry and Texture Interpolation*, document m43658, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana, Slovenia, Jul. 2018.

[53] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, *Microsoft Voxelized Upper Bodies—A Voxelized Point Cloud Dataset*, document m38673/M72012, ISO/IEC JTC1/SC29 Joint WG11/WG1(MPEG/JPEG), Geneva, Switzerland, May 2016.

[54] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, *8i Voxelized Full Bodies—A Voxelized Point Cloud Dataset*, document WG11M40059/WG1M74006, ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Geneva, Switzerland, Jan. 2017.

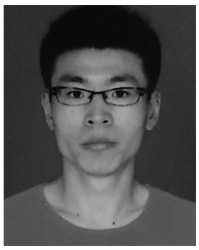
[55] S. Schwarz and D. Flynn, *Common Test Conditions for Point Cloud Compression*, document N18175, ISO/IEC JTC1/SC29/WG11 MPEG, Marrakesh, Morocco, Jan. 2019.

[56] *3D Point Cloud Dataset for MPEG-PCC*. Accessed: Dec. 20, 2019. [Online]. Available: http://mpegfs.int-evry.fr/MPEG/PCC/DataSets/pointCloud/CfP/datasets/

[57] *Reference software of MPEG-PCC*. [Online]. Available: http://mpegx.int-evry.fr/software/MPEG/PCC/TM/

[58] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, *Evaluation Metrics for Point Cloud Compression*, document m39966, ISO/IEC JTC1/SC29/WG11 MPEG, Geneva, Switzerland, Jan. 2017.

[59] G. Bjontegaard, *Improvements of the BD-PSNR Model*, document VCEG-AI11, ITU-T SG16 Q.6, Berlin, Germany, Jul. 2008.



Hao Liu received the B.S. degree from the Department of Telecommunication Engineering, Shandong Agricultural University, Shandong, China, in 2017. He is currently pursuing the Ph.D. degree in information science and engineering with Shandong University. His research interest is point cloud compression and processing.

Hui Yuan (S'08–M'12–SM'17) received the B.E. and Ph.D. degrees in telecommunication engineering from Xidian University, Xi'an, China, in 2006 and 2011, respectively. Since April 2011, he has been a Lecturer, an Associate Professor, and a Full Professor with Shandong University, Jinan, China. From January 2013 to December 2014, he also worked as a Post-Doctoral Fellow (Hong Kong Scholar) with the Department of Computer Science, City University of Hong Kong. His current research interests include video/image/immersive media processing, compression, adaptive streaming, and computer vision.

Qi Liu received the B.S. degree from Shandong Technology and Business University, Shandong, China, in 2011, and the M.S. degree from the School of Telecommunication Engineering, Xidian University, Xi'an, China, in 2014. She is currently pursuing the Ph.D. degree in information science and engineering with Shandong University. Her research interests include point clouds coding and processing.

Junhui Hou (S'13–M'16) received the B.Eng. degree in information engineering (Talented Students Program) from the South China University of Technology, Guangzhou, China, in 2009, the M.Eng. degree in signal and information processing from Northwestern Polytechnical University, Xi'an, China, in 2012, and the Ph.D. degree in electrical and electronic engineering from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2016.

He has been an Assistant Professor with the Department of Computer Science, City University of Hong Kong since 2017. His research interests fall into the general areas of visual signal processing, such as image/video/3D geometry data representation, processing and analysis, semi-supervised/unsupervised data modeling for clustering/classification, and data compression and adaptive transmission. He was a recipient of several prestigious awards, including the Chinese Government Award for Outstanding Students Study Abroad from China Scholarship Council in 2015 and the Early Career Award from the Hong Kong Research Grants Council in 2018. He is serving or has served as an Associate Editor for *The Visual Computer*, an Area Editor for *Signal Processing: Image Communication*, the Guest Editor for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, the *Journal of Visual Communication and Image Representation*, and *Signal Processing: Image Communication*, and an Area Chair of ACM International Conference on Multimedia 2019 and IEEE International Conference on Multimedia & Expo 2020.

Ju Liu (M'02–SM'09) received the B.S. and M.S. degrees in electronics engineering from Shandong University (SDU), Jinan, China, in 1986 and 1989, respectively, and the Ph.D. degree in signal processing from Southeast University, Nanjing, China, in 2000. Since July 1989, he has been with the School of Information Science and Engineering, SDU. From July 2002 to December 2003, he was a Visiting Professor with the Department of Signal Theory and Communication, Polytechnic University of Catalonia and the Telecommunications Technological Center of Catalonia, Barcelona, Spain. From November 2005 to January 2006, he was a Visiting Researcher with the Department of Communications Engineering, University of Bremen, Bremen, Germany, and the Department of Communication Systems, University of Duisburg–Essen, Germany. From June to December 2009, he was a Senior Research Fellow with the Department of Electrical Engineering, University of Washington, Seattle. He has authored or coauthored of more than 200 journal papers and conference proceedings. His research interests include space–time processing in wireless communication, blind signal separation, and multimedia communications. He received the Program for New Century Excellent Talents in University of China Award, three conference best paper awards, and three national and local academic awards in blind signal processing. He is a member of the editorial committee of the *Journal of Swarm Intelligence Research*, the *Journal of Communications*, and the *Journal of Circuits and Systems* in China.