Emerging MPEG Standards for Point Cloud Compression

Sebastian Schwarz, *Senior Member, IEEE*, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, *Senior Member, IEEE*, Pablo Cesar, *Member, IEEE*, Philip A. Chou, *Fellow, IEEE*, Robert A. Cohen, *Senior Member, IEEE*, Maja Krivokuća, Sébastien Lasserre, Zhu Li, *Senior Member, IEEE*, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, *Member, IEEE*, Ernestasia Siahaan, *Member, IEEE*, Ali Tabatabai, *Life Fellow, IEEE*, Alexis M. Tourapis, and Vladyslav Zakharchenko

***Abstract*— Due to the increased popularity of augmented and virtual reality experiences, the interest in capturing the real world in multiple dimensions and in presenting it to users in an immersible fashion has never been higher. Distributing such representations enables users to freely navigate in multi-sensory 3D media experiences. Unfortunately, such representations require a large amount of data, not feasible for transmission on today's networks. Efficient compression technologies well adopted in the content chain are in high demand and are key components to democratize augmented and virtual reality applications. Moving Picture Experts Group, as one of the main standardization groups dealing with multimedia, identified the trend and started recently the process of building an open standard for compactly representing 3D point clouds, which are the 3D equivalent of the very well-known 2D pixels. This paper introduces the main developments and technical aspects of this ongoing standardization effort.**

***Index Terms*— Point cloud coding, 3D data coding, immersive video coding.**

Manuscript received July 30, 2018; revised September 26, 2018; accepted November 13, 2018. Date of publication December 10, 2018; date of current version March 11, 2019. This paper was recommended by Guest Editor W.-H. Peng. *(Corresponding author: Sebastian Schwarz.)*

S. Schwarz is with Nokia Technologies, 33100 Tampere, Finland (e-mail: sebastian.schwarz@nokia.com).

M. Preda is with the Institut Mines-Télécom, 75014 Paris, France.

V. Baroncini is with EVATech, 00149 Rome, Italy.

M. Budagavi is with Samsung Electronics, Richardson, TX 75025 USA.

P. Cesar and E. Siahaan are with the Centrum voor Wiskunde en Infomartica, 1098 XG Amsterdam, The Netherlands.

P. A. Chou is with Google, Mountain View, CA 94043 USA.

R. A. Cohen is with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada.

M. Krivokuća is with 8i, Wellington 6011, New Zealand.

S. Lasserre is with BlackBerry Ltd., Waterloo, ON N2K 0A7, Canada.

Z. Li is with the Department of Computer Science and Electrical Engineering, University of Missouri, Kansas City, MO 64110 USA.

J. Llach is with Technicolor, 35576 Cesson-Sévigné, France.

K. Mammou and A. M. Tourapis are with Apple Inc., Cupertino, CA 85014 USA.

R. Mekuria is with Unified Streaming and Point Cloud Compression, 1054 Amsterdam, The Netherlands.

O. Nakagami and A. Tabatabai are with Sony Corporation, Tokyo 108-0075, Japan.

V. Zakharchenko is with Futurewei Technologies Inc., Santa Clara, CA 95050 USA.

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

Digital Object Identifier 10.1109/JETCAS.2018.2885981

I. Introduction

ADVANCES in 3D sensing and capturing technology have unleashed a new wave of innovation in Virtual/Augmented/Mixed reality (VR/AR/MR) content creation and communication, as well as 3D sensing for smart city, robotics and automated driving applications. There is now a huge interest from the virtual reality market in being able to represent digitally the real world in three dimensions, thus enabling the end-user to freely navigate in this digital representation. Volumetric visual data describes a 3D scene and objects with its geometry (shape, size, position in 3D-space) and respective attributes (e.g., color, opacity, reflectance, albedo), plus any temporal changes. Such data is typically computer-generated from 3D models, or is captured from real-world scenes using a variety of solutions such as multiple cameras or a combination of video and dedicated geometry sensors. Common representation formats for such volumetric data are polygon meshes or point clouds. Temporal information is included in the form of individual capture instances, similar to frames in a 2D video, or by other means, e.g., position of an object as a function of time. Because volumetric video describes a complete 3D scene or object, such data can be visualized from any viewpoint. Therefore, volumetric video is a key enabling technology for any AR, VR, or MR applications, especially for providing Six Degrees of Freedom (6DoF) viewing capabilities.

While MPEG in prior standards has already addressed the coding of 3D worlds [1], specifically computer-generated worlds, recently it launched an ambitious road map of technologies for coding representations of real 3D scenes [2]. One of these technologies is called Point Cloud Compression (PCC) and is expected to be delivered as an ISO standard in the beginning of 2020. In 2017, MPEG issued a call for proposals on PCC, and since then it has been evaluating and improving the performances of the proposed technologies [3]. Such point cloud data presents new challenges to the signal processing and compression research community. Previous compression solutions for volumetric visual representations either focused on computer-generated content [1], [4] or suffered from low spatial and temporal compression performance [5], [6] when dealing with captured natural content. For natural captured

3D sensor signals, scene geometry needs an efficient representation that is scalable in level of detail and efficient in compression, while its photometric attributes are a new class of signal that is not sampled on an uniform Euclidean grid and therefore needs new sampling, filtering, and transform tools to represent and compress. Recent advances in Graph Signal Processing (GSP) [7] have provided a rich set of tools for that.

The remainder of this paper is structured as follows: First an overview on point cloud data and its characteristics is given in Section II, followed by an overview of previous works on point cloud compression in Section III. Section IV describes the development and evaluation of the MPEG CfP on PCC, with Sections V, VI, and VII describing the three selected approaches in more detail. A brief summary of the coding performance of each proposal is given in Section VIII, before this paper is concluded in Section IX.

II. Point Cloud Data

Many emerging applications including immersive VR/MR video, automotive/robotic navigation, and medical imaging require the capture and processing of 3D scene/object geometry data. This data, in its most primitive form, consists of a collection of points called a point cloud. This section will introduce some of the aspects of point cloud data.

A. Characteristics

A point cloud consists of a set of individual 3D points. Each point, in addition to having a 3D (x, y, z) position, i.e., spatial attribute, may also contain a number of other attributes such as color, reflectance, surface normal, etc. There are no spatial connections or ordering relations specified among the individual points.

For computer graphics and gaming applications in particular, 3D scene object geometry is typically represented by polygonal meshes comprising a list of vertices together with their connectivity information in terms of edges and faces. Such polygonal meshes are well suited for compact representation of dense surfaces, but they have problems representing non-manifold structures. Key advantages of a point cloud representation over polygonal meshes are its flexibility to represent non-manifold geometry and its real-time processing potential as there is no need to store, maintain, or process surface topological information.

For efficient processing of point cloud data, each point is quantized into a cubic grid composed of $2^{-d} \times 2^{-d} \times 2^{-d}$ size voxels which are formed from volumetric subdivision, up to d levels of detail (LoD), of a $1 \times 1 \times 1$ cubic root voxel. Resulting voxels may be mapped into an octree data structure to create a voxelized octree, which facilitates, in turn, the traversal, search, and access of the neighboring voxels [8], [9].

B. Use Cases & Applications

3D point cloud data finds applications in many fields, including cultural heritage/museums, 3D free viewpoint video, real-time immersive telepresence, content VR viewing with interactive parallax, mobile mapping, and autonomous navigation [10], [11]. Regarding cultural heritage applications, point cloud data scans are used to archive and visualize objects in museums including historical statues and buildings [12], [13]. Typical point clouds in this use case may contain from millions to billions of points with finer than 1 cm of geometric precision and an 8-12 bits per color component accuracy [10].

The goal of immersive video is to go beyond higher image quality (4K/8K TV) and to provide a higher sense of 3D user experience and interactivity. Real-time 3D telepresence is one of the key applications of immersive video and 3D point clouds, for which a collection of random and unrelated points is a preferred data representation format because of its simplicity for visualization, filtering and editing. Some industrial examples of 3D telepresence include Microsoft's Holoportation [14] and 8i's volumetric video technology [15]. Variations of immersive video include HMD (head-mounted display) based VR and 3D free viewpoint sports replay and broadcasting [16], which may not require real time processing and may in addition contain mesh based graphical data content. Such media-related use cases may usually contain between 100,000 and 10,000,000 point locations and color attributes with 8-10 bits per color component [17], along with as some sort of temporal information, similar to frames in a video sequence.

For navigation purposes, it is possible to generate a 3D map by combining depth measurements from a high-density laser scanner, e.g. LIDAR, camera captured images and localization data measured with GPS and an inertial measurement unit (IMU) [18]. Such maps can further be combined with road markings such as lane information and road signs to create maps to enable autonomous navigation of vehicles around a city. This use case requires the capture of millions to billions of 3D points with up to 1 cm precision, together with additional attributes, namely color with 8-12 bits per color component, surface normals and reflectance properties attributes.

To address this wide range of applications, the MPEG PCC standardization activity created three general categories of point cloud test data: static, dynamic, and dynamically acquired [10].

C. Capture & Acquisition

There already exist many standards to compress images, video, and LIDAR sensor data, so the objective of this emerging PCC standard is not to compress the raw sensor data, but to compress the point cloud representations of the objects or scenes captured by the sensors. The coding techniques developed here are generally designed to be agnostic of the specific sensors used to create the point cloud data, so it is assumed that prior to compression, 3D data from different sensors was fused to generate the point cloud representation to be compressed.

An example of a sensor system used to dynamically acquire data for mobile mapping and autonomous navigation purposes is shown in Fig. 1 [18]. The LIDAR sensors mounted on top of a vehicle continuously acquire point locations relative to the vehicle, based upon the azimuth and elevation of the emitted laser beam, along with the range and intensity of any returned reflections of the laser. GPS and inertial sensors on the vehicle are used to determine the location of the vehicle. By combining

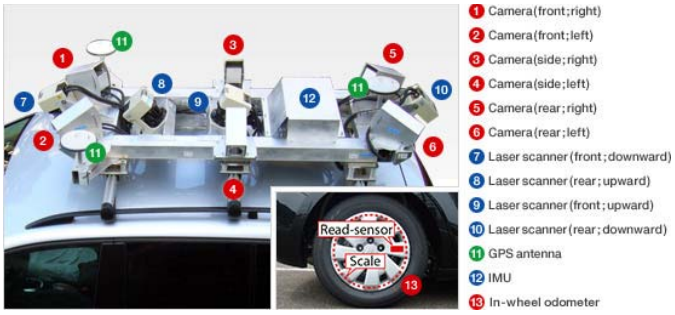


Fig. 1. Sensor system for generating mobile mapping point clouds (from [18]).

Fig. 2. Example studio for capturing dynamic point clouds.

the relative LIDAR-captured point locations along with the location of the vehicle, the point locations can be converted to absolute (x, y, z) coordinates relative to a fixed origin of a geographic coordinate system. Fixed RGB cameras mounted on the vehicle capture image sequences or video. These data are fused in a post-capture processing operation so that each point, in addition to having a LIDAR-captured reflectance attribute, can have a single RGB color attribute associated with it. The fusing process can also clean the data, e.g. by removing redundant or outlying points. The end result of this process is a point cloud comprising a list of (x, y, z) point coordinates along with reflectance and RGB attributes associated with each point. Additional attributes such as latitude, longitude, and GPS timestamps can also be included as attributes, however, compression of these additional attributes is currently outside the scope of this standard under development.

To capture high-resolution real-time point clouds of moving objects such as people for applications such as AR/VR/MR, volumetric video, and telepresence, an arrangement of sensors in a studio environment can be used to surround and capture representations of anything within a 3D space, such as that shown in Fig. 2. Multiple video or imaging cameras can be used to capture the color attributes in the scene, and the location of objects in 3D space can be captured through means such as infrared depth cameras, photogrammetry and stereo disparity, and illumination of the scene with structured light or lasers. Real-time and post-production computer processing of the data results in a sparse voxelized point cloud representing the captured objects. An example of a method for capturing voxelized point clouds using only cameras is described in [19].

The same types of sensors described here can also be used to acquire data for generating point clouds of static objects such as buildings and their interiors, objects and assemblies for industrial and cultural heritage applications, and terrain features. For example, RGB and depth cameras were used to generate the large-scale dataset of indoor scenes described in [20]. By fusing data from aerial images and LIDAR scans along with ground based LIDAR and imaging data, point cloud models of cities can be generated, as demonstrated in [21]. Capturing point clouds of cultural and historical objects or archeological sites can also be done with these kinds of sensors. A high-level overview of various methods for acquiring point cloud representations of cultural objects can be found in [22].

III. Previous Work

There has been plenty of work on point cloud compression in the past, but most works aim only at the compression of static point clouds, instead of time-varying point clouds as needed for AR/VR/MR applications. For example, a point cloud codec was introduced in [23] based on octree composition. Techniques were based on bit reordering in the subdivision bytes to reduce the entropy. This method also included color coding based on frequency of occurrence (colorization) and normal coding based on spherical quantization. A similar work in [24] used surface approximations to predict occupancy codes and an octree structure to encode color information. The work in [5] introduced a real-time octree-based codec that could also exploit temporal redundancies by XOR operations on the octree byte stream. This method could operate in real time, as the XOR prediction is simple and fast. A disadvantage of this approach is that the effectiveness is significant only for scenes with limited movement, which is not always the case. In [6], an extension to this framework was introduced, combining the octree-based codec with a common image codec for color attribute coding. Thanou *et al.* [25] introduced a time-varying point cloud codec that can predict graph-encoded octree structures between adjacent frames. The method uses spectral wavelet-based features to achieve this and an encoding of differences to achieve a lossless encoding. This method also includes the color coding method from [26], which defines small subgraphs based on the octree of the point cloud. These subgraphs are then used to efficiently code the colors by decomposing them on the eigenvectors of the graph Laplacian.

In comparison to point clouds, 3D objects are often coded as 3D meshes, for which a significant number of compression methods were developed. Early work on mesh compression includes [27]–[29]. Mesh codecs can be categorized as progressive, i.e., allowing a lower resolution rendering from partial bit streams, and single rate, for which only decoding at full resolution is available [30]. For networked transmission, progressive methods have generally been preferred, but for 3D immersive video, single rate methods can also be useful, as they introduce less encoder computation and bitrate overhead [31]. Several works [32]–[34] have aimed at compressing object-based immersive 3D video using single-rate coding. While these methods are promising, it seems that methods based on 3D point clouds can result in coding with even less overhead and more flexible progressive rendering capabilities, as the point cloud format is simpler to acquire and process. There are international standards for mesh compression [1], [35] defined, which are greatly beneficial for interoperability

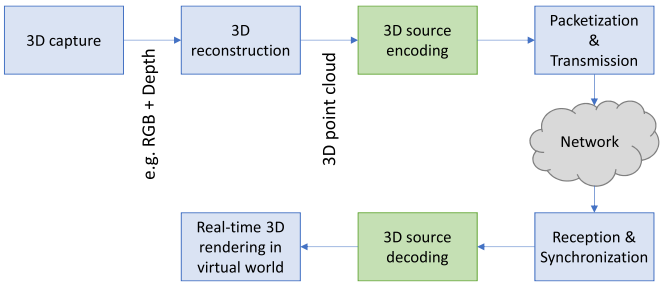


Fig. 3. Data path for 3D geometry based tele-immersion use case.

between devices and services. These methods have been mostly designed for remote rendering and have low decoder complexity and a slightly higher encoder complexity. For 3D immersive and augmented 3D video coding, it is essential to have both low encoder and decoder complexity, analogous to video coding in video conferencing systems as compared with video on demand.

Somewhat related, multiview video plus depth (MVD) representation was considered for storing video and depth maps from multiple cameras in extensions of the international HEVC standard [36]. For such representations arbitrary view points can be rendered by interpolation between different camera views using techniques from depth image based rendering (DIBR), enabling free viewpoint functionality. While these formats can be used to represent the visual 3D scene, they do not explicitly store 3D object geometries, which is useful for composite rendering in immersive communications and augmented reality. Therefore, these formats are not directly applicable to immersive and augmented 3D object-based video combining real and virtual content.

IV. MPEG CfP Process

In 2014, the MPEG 3D graphics coding (3DG) group started an exploration to study the feasibility of adapting its tools to advanced immersive applications such as virtual teleportation. These applications typically deal with photo-realistic meshes and point clouds of millions of points acquired from 3D scanners and/or computer vision algorithms. Initial scans of 3D meshes and point cloud content were contributed, and a practical streaming prototype was developed as part of the Reverie FP7 project [37]. The block diagram of the data flow in immersive communications is shown in Fig. 3. In such systems real-time communication processing is important as is the resilience to noisy data and handling of dense point clouds.

The advantage of this approach is that composite rendering in scenes facilitates AR, VR, and free view point functionalities. However, for this use case, existing MPEG standards for 3D graphics were found to be less suitable due to the fact that several requirements like noise resilience and low encoder latency were not fully addressed. These standards were mostly developed with computer animated content in mind, which typically dealt with sparse geometric content with limited amounts of noise. This realization led to the start of an exploration activity.

(a) (b) (c)

Fig. 4. Point cloud examples for the three different categories. (a) Static. (b) Dynamic (detail). (c) Dynamically acquired (detail).

A. CfP Development

Following several experiments and evaluations of techniques available for immersive media applications, it was found that point clouds were particularly suitable for these kinds of applications. Point cloud data scored well in experiments comparing visual quality, bitrate, and compression performance as well as computational complexity [38]. In addition, other use cases were introduced such as free viewpoint broadcasting and 3D scans produced by mobile mapping systems.

A call for proposals was developed in close cooperation with stakeholders including major mobile device manufacturers and leading startups that also provided some of the highly realistic content needed for the effort. The call for proposals on point cloud compression (PCC) was published in January 2017 targeting an international standard for PCC [3] addressing three categories: static point clouds (category 1); dynamic, time varying point clouds used for immersive video and AR video (category 2); and dynamically acquired point clouds, e.g., used in mobile mapping (category 3). Representative examples for data in these three categories are shown in Fig. 4.

B. Evaluation Methodology

During the CfP development period, evaluation metrics were developed in a series of experiments, requirements, and use case assessments. To have a baseline for determining target bitrates and distortions, a recent hybrid octree-image point cloud codec for tele-immersive video [6] was chosen as anchor. Quality metrics described in [39] and [40] were selected for the objective quality assessment. These metrics are referred as point-to-point ($D1$) and point to plane ($D2$) geometry distortion metrics. In the first geometry metric ($D1$),

the comparison is such that the Mean Square Error (MSE) between the reconstructed point and the closest corresponding points in the reference point cloud is calculated. In the second geometry metric ($D2$), the MSE is calculated between the reconstructed point and the surface plane in the given reference test data. Surface normals are provided with the reference test data to facilitate the computation of the surface planes. The $D1$ metric is also used for assessing attribute (color or reflectance) distortion in YUV color space. Peak signal-to-noise-ratios (PSNR) are obtained based on the 3D volume resolution for geometry, and respectively for color depths for each color channel. Bjontegaard-delta (BD) metrics are derived comparing the distortions against the anchor implementation at predefined target bitrates [41]. Additional effort was put in choosing meaningful target bitrates, e.g. covering a wide range of applications and qualities, and establishing a method for plotting rate-distortion (RD) curves based on the objective metrics and rate-points. For example, for dynamic (category 2) point clouds, the target bitrates were in the range of 3 to 55 MBit/s, representing 0.2% to 5% of the original uncompressed data.

In addition to using objective metrics, a subjective evaluation methodology was defined that consisted of rendering the point clouds using a virtual camera path and then performing the quality assessment via techniques similar to those used to evaluate video quality. For the subjective assessment only three static objects and three dynamic scenes were considered among the total of 30 test objects considered in the overall CfP. The entire set of 19 static objects, five dynamic objects and six dynamic acquisition scenes were considered for the objective evaluation. This reduced subjective test set was due to the need to minimize the effort required to complete the tests and because dynamic acquisition scenes are typically processed by a computer and are not directly viewed as a final product. The subjective visual quality assessment of the static and dynamic scenes was made possible by using a point cloud renderer designed by Technicolor [42]. This software allows specifying a camera view path, displaying a static representation of a point cloud, rotating it on three axes, and zooming. When rotating and zooming a static object, it is possible to record a track of all the movements. The recorded tracks are used to create video clips. The same process is applied on dynamic sequences, where the video clips were produced by rotating the object while playing it out. The tracks used to create video from the static and dynamic 3D files were not known to the proponents; this was done to avoid any bias in the coding process to a particular point of view.

The subjective tests for 3D point cloud compression were done using the absolute category rating (ACR) test method as specified by ITU-T Recommendation P.910 [43]. The decision to use ACR was made considering the high number of submissions received (9 + anchor), the high number of test points to evaluate (171), and the short time available to run the subjective experiments before the CfP evaluation. The range used for the experiment was a five grades of quality scale from 1 (bad) to 5 (excellent). The testing environment was carefully designed and implemented. Two to three people at the same time were seated in front of a 4K top quality 55” consumer TV set. All viewers were seated at two times the active screen height (2H) distance from the TV screen. There were in total 22 participants: 9 male, and 13 female, with ages between 20 and 30 years, all screened for vision acuity and color vision. Additional details on the objective and subjective evaluation methods are available in the CfP document [3] and its corrigenda [44].

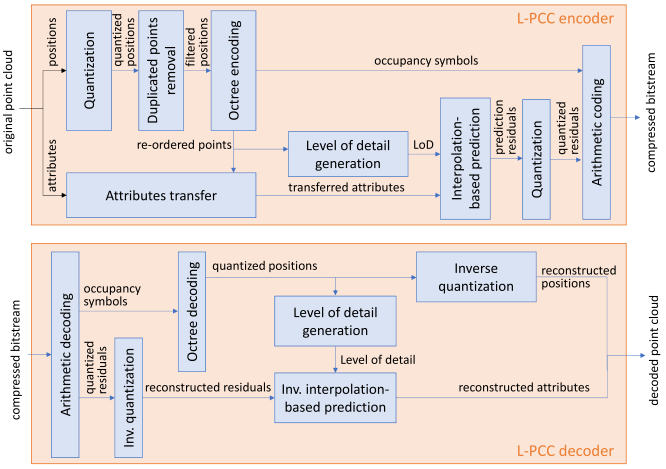


Fig. 5. Overview of the L-PCC compression and decompression process.

C. Results and Next Steps

A total of 13 different proposals were submitted to MPEG, and they were evaluated in October 2017. The results of the subjective assessment [45] were almost in line with the objective evaluations [46]. The average confidence interval for the subjective evaluation of static content was 0.48 MOS values, while that of dynamic content was 0.34 MOS values. Thus, it was concluded that the experiment was correctly conducted and the results obtained could be used by the group for making decisions with respect to the appropriate technologies to be selected.

As an outcome, three different technologies were chosen as test models (TMs) for the three different categories targeted: LIDAR point cloud compression (L-PCC) for dynamically acquired data, surface point cloud compression (S-PCC) for static point cloud data, and video-based point cloud compression (V-PCC) for dynamic content. The three different approaches are described in more detail in the following sections.

V. LIDAR Point Cloud Compression

The L-PCC codec was designed to efficiently compress LIDAR point clouds, which usually exhibit highly irregular sampling. Because of such characteristics, L-PCC compresses first the point cloud geometry information by exploiting an octree-based encoding strategy. The reconstructed geometry is then used to build a Level-Of-Detail (LoD) structure, which makes it possible to efficiently predict attributes and encode/transmit them in a scalable manner.

Fig. 5 provides an overview of the L-PCC compression and decompression processes.

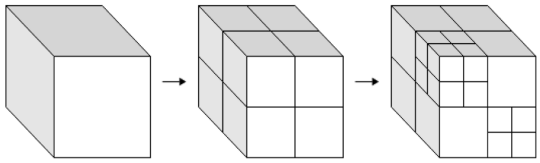


Fig. 6. Generating octree structure by recursive subdivision.

The remainder of this section is organized as follows: Section V-A describes the octree-based coding process for the geometry information. Section V-B then describes the attributes transfer module, which is a process that helps determine the appropriate attribute values that should be associated with the reconstructed geometry information. The LoD generation, an essential process that enables efficient hierarchical prediction of the attributes, is then described in Section V-C. Finally, an interpolation-based prediction module that is used to further improve the coding efficiency of the attribute values by exploiting spatial correlations as well as the quantization and dequantization steps that are applied on the residuals are described in Section V-D.

A. Octree-Based Geometry Coding

Let $(X_i = (x_i, y_i, z_i))_{i=1...N}$ be the set of 3D positions associated with the points of the input point cloud. The L-PCC encoder computes the quantized positions $(\hat{X}_i)_{i=1...N}$ as follows:

$$\hat{X}_i = \lfloor (X_i - X_{shift}) \times s \rfloor, \tag{1}$$

where X_{shift} and s are user-defined parameters that are signaled in the bitstream.

Equivalently, at the decoder, the reconstructed positions $(\tilde{X}_i)_{i=1...N}$ are generated by applying the following inverse quantization process:

$$\tilde{X}_i = \frac{\hat{X}_i}{s} + X_{shift}. \tag{2}$$

After quantization, an optional process that removes duplicate points may be applied. It consists of merging points sharing the same quantized positions into a single point. The attribute values associated with the merged point are computed using the attributes transfer module described in subsection V-B.

The octree-based encoding process compresses the quantized positions as follows: First, a cubical axis-aligned bounding box B is defined by the two extreme points (0,0,0) and $(2^n,2^n,2^n)$, where n is the smallest integer that verifies the following inequality:

$$2^n > max\left(\max_{1\le j\le n}(\hat{x}_j), \max_{1\le j\le n}(\hat{y}_j), \max_{1\le j\le n}(\hat{z}_j).\right) \tag{3}$$

An octree structure is then built by recursively subdividing B, as depicted in Fig. 6. At each stage, the current cube is subdivided into 8 sub-cubes. An 8-bit code, named the *subdivision code*, is then generated by associating an 1-bit value with each sub-cube in order to indicate whether it contains points (i.e., is occupied and has a value of 1) or not (i.e., is empty and has a value of 0). Only occupied sub-cubes with a size higher than 1 are further subdivided. Since points may be duplicated, multiple points may be mapped to the same sub-cube of size 1. In order to handle such a situation, the number of points c for each sub-cube of dimension 1 is also arithmetically encoded.

On the decoder side, the decoding process starts by reading from the bitstream the bounding box B. The same octree structure is then built by subdividing B according to the subdivision codes read from the bitstream. Each time a sub-cube of dimension 1 is reached, the number of points c for that sub-cube is arithmetically decoded and c points located at the origin of the sub-cube are generated.

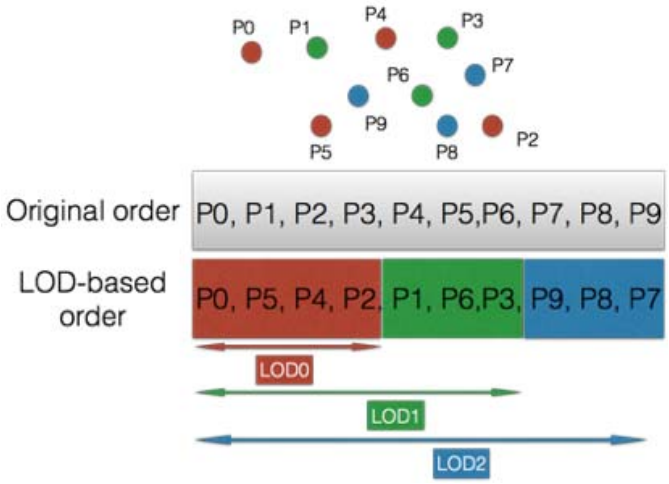


Fig. 7. Overview of Level of detail generation process.

B. Attribute Transfer

Given the input point cloud positions $(X_i)_{i=1...N}$, the input point cloud attributes $(A_{1i}, A_{2i}, \ldots, A_{Di})$, where D is the number of attributes, and the reconstructed positions $(\tilde{X}_i)_{i=1...N_{rec}}$, the objective of the attributes transfer module is to determine the attribute values $(\tilde{A}_{Di})_{i=1...N_{rec}}$ associated with the reconstructed positions $(\tilde{X}_i)_{i=1...N_{rec}}$ that minimize the attribute distortions described in Section IV-B.

For each point i in the reconstructed point cloud, let $\tilde{X}_i$ be its position, X_i^* be the position of its nearest neighbor in the original point cloud, A_{Di}^* be the attribute value associated with X_i^*, and $Q_i^+ = (X_i^+(h))_{h=1...H(i)}$ be the set of points in the original point cloud that share $\tilde{X}_i$ as their nearest neighbor in the reconstructed point cloud. $A_{Di}^+(h))_{h=1...H(i)}$ are also the corresponding attribute values of the points in Q_i^+.

If $Q^+(i)$ is empty, then $\tilde{A}_{Di}$ is assigned the attribute value A_{Di}^*. Otherwise, $\tilde{A}_{Di}$ is computed as attribute value averaged over all the points in the original point cloud that share the reconstructed position as their nearest neighbor, as follows:

$$\tilde{A}_{Di} = \frac{1}{H(i)} \sum_{h=1...H(i)} A_D h^+ \tag{4}$$

C. Level of Detail Generation

The level of detail (LOD) generation process illustrated in Fig. 7 re-organizes the input point cloud into a set of refinement levels $(R_l)_{l=1...L}$. This is done according to a set of Euclidean distances $(d_l)_{l=1...L}$ that are specified by the user

and by verifying the following two conditions: (1) $d_L = 0$ and (2) $d_l < d_{l-1}$. The objective of this process is that the first LOD, i.e. refinement level R_1, contains points that are in effect a coarse representation of the point cloud, because all points in that LOD are separated by a distance of at least d_1. Subsequent LODs or refinement levels include points that are closer together, because the distances d_l decrease as l increases.

The re-ordering process is deterministic and operates on the quantized positions ordered according to the octree decoding process. It is applied at both the encoder and the decoder side. This process first marks all the points as non-visited, and the set of visited points, denoted as V, is set as empty. L-PCC proceeds iteratively. At each iteration l, the refinement level R_l is generated as follows: L-PCC iterates over all the points. If the current point has been visited, then it is ignored. Otherwise, the minimum distance D of the current point to the set V is computed. If D is strictly lower than d_l, then the current point is ignored. Otherwise, the current point is marked as visited and added to both R_l and V. This process is repeated until all the points are traversed. The level of detail at iteration l, LOD_l, is obtained by taking the union of the refinement levels $R_1, R_2, \ldots, R_l$.

D. Interpolation-Based Attributes Prediction

The attributes associated with the point cloud are encoded/decoded in the order defined by the LOD generation process. At each step, only the already encoded/decoded points are considered for prediction. More precisely, the attribute value $\tilde{A}_i$ is predicted by using a linear interpolation process based on the distances of the nearest neighbors of point $\tilde{X}_i$. More precisely, let ∇_i be the set of the k-nearest neighbors of the current point $\tilde{X}_i$, let $(\hat{A}_j)_{j\in\nabla_i}$ be their decoded/reconstructed attribute values, and $(\delta_j)_{j\in\nabla_i}$ their distances to $\tilde{X}_i$. The predicted attribute value $\tilde{\Gamma}_i$ is then given by:

$$\tilde{\Gamma}_i = \frac{1}{k\sum_{j\in\nabla_i}\frac{1}{\delta_j^2}}\sum_{j\in\nabla_i}\frac{1}{\delta_j^2}\hat{A}_j \tag{5}$$

The prediction residuals $\tilde{\rho}_i$ are then computed as follows:

$$\tilde{\rho}_i = \tilde{A}_i - \tilde{\Gamma}_i \tag{6}$$

The residuals $\tilde{\rho}_i$ are quantized and arithmetically encoded. The reconstructed attributes values are subsequently obtained as follows:

$$\hat{A}_i = \hat{\rho}_i + \tilde{\Gamma}_i, \tag{7}$$

where $\hat{\rho}_i$ are the reconstructed prediction residuals.

The experimental evaluation of the proposed hierarchical prediction scheme shows that the optimal choice of the number of nearest neighbors k and the distances $(d_l)_{l=1\ldots L}$ is content dependent and may be computationally expensive. However, a practical encoder implementation could be designed by considering different computational complexity and RD performance trade-offs. For instance, the optimal number of nearest neighbors k could be chosen based on a Rate-Distortion Optimization (RDO) process. This process consists of evaluating the Lagrangian costs associated with different values of $k \in \{1, 2, 3, \ldots, K\}$ and selecting the one with the lowest cost. Determining the optimal set of distances $(d_l)_{l=1\ldots L}$ could also be formulated as an RDO optimization problem. However, evaluating the Lagrangian cost for all possible combinations would be computationally prohibitive. One way to reduce the size of the search space is to impose the following simple recursive relationship between the sampling distances:

$$d_{l-1} = \frac{d_l}{2}, \quad \forall l \in \{2, \ldots, L-1\}. \tag{8}$$

By introducing such a constraint, the sequence of distances $(d_l)_{l=1\ldots L}$ becomes entirely defined by d_{L-1}. A search strategy, such as the binary search method, could be then applied to determine the distance d^*_{L-1} that minimizes the Lagrangian cost function.

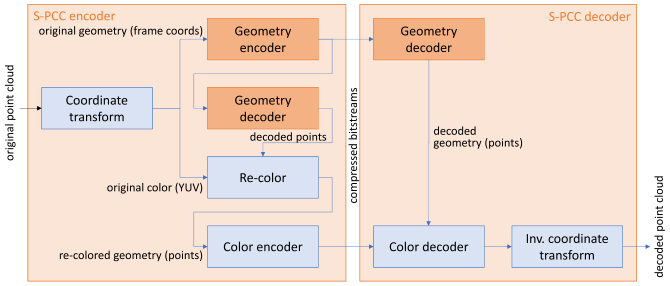


Fig. 8. Block diagram of the S-PCC encoder and decoder.

VI. Surface Point Cloud Compression

The S-PCC codec was designed to efficiently compress high-detail static point clouds, which usually exhibit a high sampling density, approximating a 3D surface. The remainder of this section is organized as follows: Section VI-A describes the overall S-PCC architecture, Section VI-B describes the encoder, and Section VI-C describes the decoder.

A. S-PCC Architecture

The S-PCC architecture comprises an encoder and decoder, which in turn comprise various modules, as shown in Fig. 8. Communication between modules is done by passing lists of point locations and/or point attributes. Parameters to the S-PCC encoder include the following, many of which are passed to the decoder in the bitstream header:

- *depth*: depth of encoding octree
- *level*: octree level for geometry encoding
- *geomstepsize*: geometry quantization stepsize
- *colorstepsize*: color quantization stepsize
- *mbpstarget*: total bitrate target in Mbps
- *fps*: frames per second
- *scale*: frame-to-world scale parameter for decoded PLY
- *translation*: frame-to-world translation for decoded PLY

B. S-PCC Encoder

The input to the S-PCC encoder is the original, uncompressed point cloud. The point cloud comprises a list of real-valued point locations, $(x_i^{world}, y_i^{world}, z_i^{world})$, $i = 1, \ldots, N$, where N is the number of points in the input point cloud,

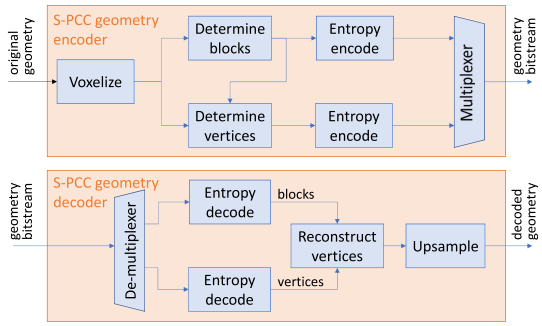


Fig. 9. Block diagram of the S-PCC geometry encoder and decoder.

and a corresponding list of real-valued point attributes, $(A_{1i}, A_{2i}, \ldots, A_{D_i})$, $i = 1, \ldots, N$, where D_i is the number of attributes for point i. All attributes are processed in independent channels. In this section, we will focus on the case where the attributes are the three RGB color channels: (R_i, G_i, B_i), $i = 1, \ldots, N$, and therefore $D_i = 3\,\forall i$.

The original point locations are expressed in a coordinate system that has meaning to the user. In the S-PCC, this coordinate system is called the *world* coordinate system. Furthermore, the original point colors are typically expressed in an RGB color space. However, it is more convenient to process the point locations in a different coordinate system, and to process the point colors in a different color space. Thus, just after the original point cloud enters the S-PCC encoder, it is processed by a coordinate transformation module, in which the original point locations are transformed from their original (*world*) coordinates into internal (*frame*) coordinates, and the original point colors are transformed from RGB to YUV.

The transformation from world to frame coordinates may be specified by using the parameters $translation = (t_x, t_y, t_z)$ and $scale = s$, as

$$(x_i, y_i, z_i) = \left((x_i^{world}, y_i^{world}, z_i^{world}) - (t_x, t_y, t_z)\right)/s. \quad (9)$$

If *translation* and *scale* are specified, the transformed location parameters (x_i, y_i, z_i) must lie in the cube $[0, 2^{depth})^3$. If they are not specified, they are derived in the module by computing a minimum bounding cube of the input point locations, and scaling and translating such that in the frame coordinate system, all point locations (x_i, y_i, z_i) lie in the cube $[0, 2^{depth})^3$, and along at least one dimension, the minimum and maximum are respectively 0 and $2^{depth} - 1$. For the color attributes, the transformation from (R_i, G_i, B_i) to (Y_i, U_i, V_i), $i = 1, \ldots, N$, in the module follows ITU-R Rec. BT.709 as required in [3].

Details of the S-PCC geometry encoder module are shown in Fig. 9, and described as follows:

- **Voxelization**. Voxelization is the process of grouping points together into voxels, which are the set of unit cubes $[i - 0.5, i + 0.5) \times [j - 0.5, j + 0.5) \times [k - 0.5, k + 0.5)$ for integer values of i, j, and k between 0 and $2^{depth} - 1$. Specifically, the locations of all points within a voxel are quantized to the voxel center, and the attributes of all points within the voxel are averaged and assigned to the voxel. A voxel is said to be *occupied* if it contains any point of the point cloud.
- **Determining blocks**. In the S-PCC, the cube of voxels is partitioned into *blocks* of $W \times W \times W$ voxels, analogous to the partitioning of video pictures into blocks of $W \times W$ pixels. W is known as the *blockwidth*. The blockwidth is constant, $W = 2^{depth-\ell}$, where $\ell = level$ is a parameter to the encoder and is passed to the decoder in the bitstream header. A block is said to be *occupied* if it contains any occupied voxels. The use of blocks to represent geometry is important for spatial random access, view-dependent coding and rendering, parallel processing, out-of-core processing for large datasets, and the formation of "slices" and other units for network packetization and error resilience.
- **Entropy encoding of blocks**. The set of occupied blocks is encoded with an octree, in which the leaves of the octree represent the occupied blocks. If the octree has height $\ell = level$, then the blocks at the leaves have blockwidth $W = 2^{(depth-\ell)}$ voxels on a side. The parameter *level* is placed in the bitstream header. An octree can be represented by one byte for each internal (non-leaf) node of the tree, where the bits indicate the occupied children of the node. These are known as *occupancy bytes*. Currently, the occupancy bytes are entropy-coded. If $\ell = level$ is equal to *depth*, then $W = 1$, the blocks are $1 \times 1 \times 1$, and the octree represents the collection of voxels losslessly. If the depth of the tree is large enough, then there is at most one point in each voxel, and thus the geometry of the original point cloud can be represented losslessly, up to *depth* bits of precision (maximum of 21 bits is currently allowed) for each spatial component.
  If $\ell = level$ is less than *depth*, then the blocks are $2 \times 2 \times 2$ or larger and it is necessary to represent the collection of voxels within the block, possibly with loss. S-PCC represents the geometry within each block as a surface that intersects each edge of the block at most once. Since there 12 edges of a block, there can be at most 12 such intersections. Each such intersection is called a *vertex*. The collection of vertices is a list $(\hat{x}_k, \hat{y}_k, \hat{z}_k)$, $k = 1, \ldots, N_{vert}$. These vertices are sufficient to reconstruct a surface within the block by reconstructing a non-planar polygon through the vertices as a collection of triangles. Although there are other surfaces that can be parameterized by such a set of vertices, for example, as an implicit surface of a Bézier Volume, S-PCC uses triangles as they are particularly friendly to standard graphics processing.
- **Entropy encoding of vertices**. Vertices, nominally being intersections of a surface with edges of a block, are shared across neighboring blocks, not only guaranteeing continuity across blocks of the reconstructed surface, but also reducing the number of bits required to code the collection of vertices. The set of vertices is coded in two steps. Firstly, the set of all unique edges of occupied blocks is computed, and a bit vector determines which edges contain a vertex and which do not. Secondly, for each edge

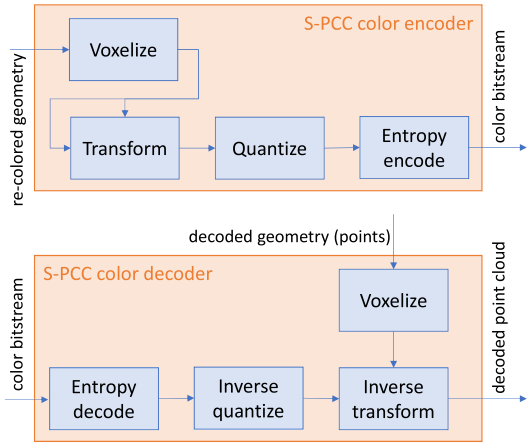


Fig. 10. Block diagram of S-PCC color encoder and decoder.

that contains a vertex, the position of the vertex along the edge is uniformly scalar-quantized to a small number of levels, typically equal to the block width if the geometric spatial resolution is desired to approximate the voxel resolution, but it could be any number of levels. The number of levels is equal to the block width divided by *geomStepsize*. The bit vector and the vertex positions are further compressed by an entropy coder, along with the octree occupancy bytes, and become the geometry bitstream.

The S-PCC encoder contains an instantiation of the geometry decoder. Details of the geometry decoder module are described in Section VI-C. The output of the geometry decoder is a list of refined vertices $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$, $r = 1, \ldots, N_{ref}$.

The re-coloring module assigns colors to the refined vertices, by taking colors from the original (uncompressed) point cloud. There are many ways to perform re-coloring, with corresponding effects on both computation and compression performance. In the S-PCC, re-coloring is implemented by coloring each refined vertex $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$ with the color $(\tilde{Y}_r, \tilde{U}_r, \tilde{V}_r) = (Y_{i_r}, U_{i_r}, V_{i_r})$ of the input point $(x_{i_r}, y_{i_r}, z_{i_r})$ closest to $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$ in Euclidean distance, i.e.,

$$i_r = \arg\min_i \left((\hat{x}_r - x_i)^2 + (\hat{y}_r - y_i)^2 + (\hat{z}_r - z_i)^2 \right). \quad (10)$$

Thus the output of the re-coloring module is the list of colors $(\tilde{Y}_r, \tilde{U}_r, \tilde{V}_r)$, $r = 1, \ldots, N_{ref}$, corresponding to the refined vertices $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$, $r = 1, \ldots, N_{ref}$.

The color encoder module compresses the colors $(\tilde{Y}_r, \tilde{U}_r, \tilde{V}_r)$, $r = 1, \ldots, N_{ref}$, of the re-colored points, using information from the already-available (or already-decoded) locations $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$, $r = 1, \ldots, N_{ref}$, of the re-colored points as side information. Details of the color encoder module are shown in Fig. 10, and described as follows:

- **Voxelization**. All the refined vertices within a voxel are quantized to the voxel center, and the attributes of the refined vertices within the voxel are averaged and assigned to the voxel. This produces a list of voxel colors $(\tilde{Y}_n, \tilde{U}_n, \tilde{V}_n)$, $n = 1, \ldots, N_{vox}$, along with a list of the associated voxel locations $(\hat{x}_n, \hat{y}_n, \hat{z}_n)$, $n = 1, \ldots, N_{vox}$, as side information.
- **Spatial Transform**. The voxel colors $(\tilde{Y}_n, \tilde{U}_n, \tilde{V}_n)$, $n = 1, \ldots, N_{vox}$, are transform-coded, analogously to a color image, by a spatial transform, quantizer, and entropy coder. The colors are spatially transformed using the Region Adaptive Hierarchical Transform (RAHT) [47], [48], to obtain transformed colors (TY_n, TU_n, TV_n), $n = 1, \ldots, N_{vox}$.
- **Quantization**. The transformed coordinates are quantized by a uniform scalar quantizer with stepsize *colorStepsize*, to obtain the quantized transform coordinates $(\widehat{TY}_n, \widehat{TU}_n, \widehat{TV}_n)$, $n = 1, \ldots, N_{vox}$,. The same stepsize is used for all color components. The *colorStepsize* is communicated to the color decoder through the bitstream header.
- **Entropy encoding**. The quantized, transformed coefficients are entropy-encoded using RLGR [49].

The output from the S-PCC encoder and the input to the decoder is a bitstream that comprises a geometry bitstream, a color bitstream, and a bitstream header. The bitstream header contains parameters needed to decode the geometry and color bitstreams, namely *depth*, *level*, *geomStepsize*, *colorStepsize*, *translation*, and *scale*.

C. S-PCC Decoder

The geometry decoder module decompresses the geometry bitstream into decoded geometry. This module is also instantiated in the S-PCC encoder. Details of the geometry decoder are shown in Fig. 9, and described as follows:

- **Entropy decoding of blocks**. The occupancy bytes of the octree are entropy-decoded and the octree is reconstructed. If the *level* of the octree is equal to the *depth* as indicated in the bitstream header, then this is a lossless representation of the geometry at that level of precision.
- **Entropy decoding of vertices**. If the *level* of the octree is smaller than the *depth*, then this is a lossy representation of the geometry, and vertices are entropy-decoded, in two steps. Firstly, the set of all unique edges of occupied blocks is computed, and a bit vector is entropy-decoded to indicate which edges contain a vertex and which do not. Secondly, for each edge that contains a vertex, the position of the vertex along the edge is entropy-decoded and dequantized, resulting in a list of vertices $(\hat{x}_k, \hat{y}_k, \hat{z}_k)$, $k = 1, \ldots, N_{vert}$.
- **Vertex reconstruction**. For each block, the vertices on the block edges determine a surface through the block. The surface is a non-planar polygon. The polygon is triangulated into planar triangles. The method of triangulation is defined so that the triangulation is unique given the vertices on the block edges.
- **Upsampling**. Each triangle is refined (or subdivided or upsampled) by an $upsamplingFactor$ (times the blockwidth) to obtain regularly-spaced points on the surface of the triangle, called $refined\ vertices$, $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$, $r = 1, \ldots, N_{ref}$. The purpose of the refined vertices is to create geometry at a spatial resolution greater than or equal to the spatial resolution of the color

information. The list of these refined vertices is the output of the geometry decoder.

The color decoder module decompresses the color bitstream into decoded color components $(\hat{Y}_n, \hat{U}_n, \hat{V}_n)$, $n = 1, \ldots, N_{vox}$, and their associated locations $(\hat{x}_n, \hat{y}_n, \hat{z}_n)$, $n = 1, \ldots, N_{vox}$, given the refined vertices $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$, $r = 1, \ldots, N_{ref}$, as side information. Details of the color decoder module are shown in Fig. 10. First, the refined vertices $(\hat{x}_r, \hat{y}_r, \hat{z}_r)$, $r = 1, \ldots, N_{ref}$, are voxelized to obtain the decoded voxel locations $(\hat{x}_n, \hat{y}_n, \hat{z}_n)$, $n = 1, \ldots, N_{vox}$. Then, the color transform coefficients are entropy-decoded, inverse-quantized, and inverse-transformed to produce a list of decoded colors, $(\hat{Y}_n, \hat{U}_n, \hat{V}_n)$, $n = 1, \ldots, N_{vox}$. The inverse transform uses, as side information, the list of decoded voxel locations $(\hat{x}_n, \hat{y}_n, \hat{z}_n)$, $n = 1, \ldots, N_{vox}$.

Before the reconstructed point cloud exits the S-PCC decoder, the point locations are transformed from "frame" coordinates into "world" coordinates in the inverse coordinate transform module, according to the parameters $translation = (t_x, t_y, t_z)$ and $scale = s$ (which are obtained from the bitstream header), as

$$(x_i^{world}, y_i^{world}, z_i^{world}) = s \cdot (x_i, y_i, z_i) + (t_x, t_y, t_z), \quad (11)$$

and the reconstructed point colors are transformed from $(\hat{Y}_n, \hat{U}_n, \hat{V}_n)$ to $(\hat{R}_n, \hat{G}_n, \hat{B}_n)$ according to ITU Rec. BT.709, $n = 1, \ldots, N_{out}$, where $N_{out} = N_{vox}$.

The output from the decoder is the point cloud reconstructed from the bit stream. The reconstructed point cloud comprises a list of real-valued point locations, $(x_i^{world}, y_i^{world}, z_i^{world})$, $n = 1, \ldots, N_{out}$, where N_{out} is the number of points in the output point cloud, and a corresponding list of color components, $(\hat{R}_n, \hat{G}_n, \hat{B}_n)$,$n = 1, \ldots, N_{out}$. The number of output points N_{out} is generally different from the number of input points N.

VII. Video-Based Point Cloud Compression

The main philosophy behind V-PCC is to leverage existing video codecs for compressing the geometry and texture information of a dynamic point cloud. This is essentially achieved by converting the point cloud into a set of different video sequences. In particular, two video sequences, one that captures the geometry information and another that captures the texture information of the point cloud data, are generated and compressed using existing video codecs, such as MPEG-4 AVC, HEVC, AV1 etc. Additional metadata, which are needed for interpreting the two video sequences, i.e., an occupancy map and auxiliary patch information, are also generated and compressed separately. The video generated bitstreams and the metadata are then multiplexed together so as to generate the final point cloud V-PCC bitstream. It should be noted that the metadata information represents a relatively small amount (i.e., 5-20%) of the overall bitstream. The bulk of the information is handled by the video codec. Fig. 11 and Fig. 12 provide an overview of the V-PCC compression and decompression processes, respectively.

The remainder of this section is organized as follows: Subsection VII-A describes the patch generation and packing processes, which aim at determining how to best decompose

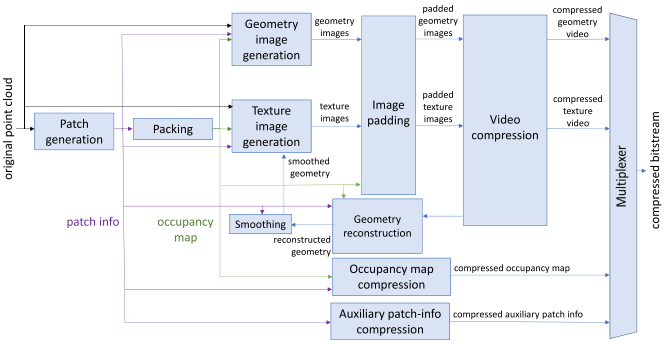


Fig. 11. Overview of the V-PCC encoding process.

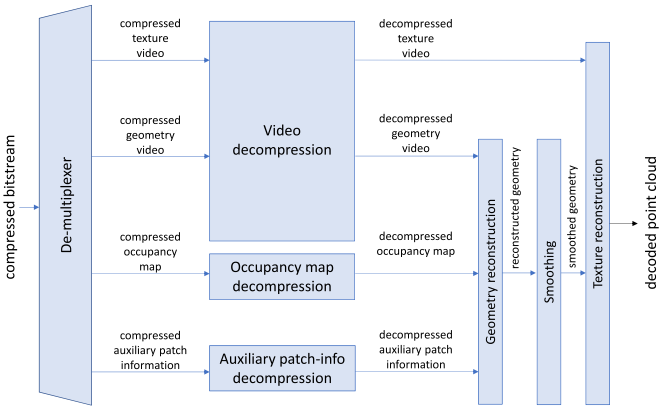


Fig. 12. Overview of the V-PCC decoding process.

the input point cloud into patches and how to most efficiently fit those patches into a rectangular 2D grid. Subsection VII-B details the image generation and padding processes, which transform the point cloud geometry and texture information into temporally correlated, piecewise smooth, 2D images suited for coding using traditional video codecs. The processes of generating the auxiliary patch information and occupancy map are described in subsections VII-C and VII-D, respectively. Subsection VII-E describes the smoothing module and the geometry and texture reconstruction processes.

A. Patch Generation & Packing

Leveraging traditional video codecs to encode point clouds requires mapping the input point cloud to a regular 2D grid. The objective is to find a temporally-coherent low-distortion injective mapping that would assign each point of the 3D point cloud to a cell of the 2D grid.

Maximizing the temporal coherency and minimizing the distance/angle distortions enables the video encoder to take full advantage of the temporal and spatial correlations of the point cloud geometry and attributes signals. An injective mapping guarantees that all the input points are captured by the geometry and attributes images and could be reconstructed without loss. Simply projecting the point cloud on the faces of a cube or on the sphere does not guarantee lossless reconstruction due to auto-occlusions (i.e., auto-occluded points are not captured), and generates in practice significant distortions.

In order to avoid such limitations, V-PCC decomposes the input point cloud into a set of patches, which could be independently mapped, through a simple orthogonal projection,

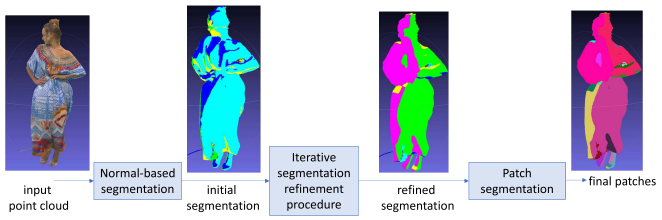


Fig. 13. Overview of the V-PCC patch generation process.

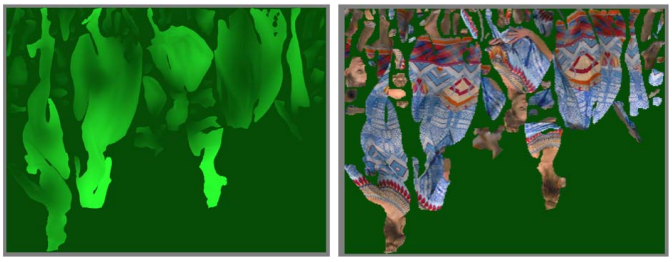


Fig. 14. Example of geometry (left) and texture (right) images.

to a 2D grid without suffering from auto-occlusions nor requiring re-sampling of the point cloud geometry. Furthermore, the patch generation process aims at generating patches with smooth boundaries, while minimizing their number and the mapping distortions. In order to resolve this NP-hard optimization problem, V-PCC applies a heuristic segmentation approach that is described in Fig. 13.

First, the normal at every point is estimated as described in [50]. An initial clustering of the point cloud is then obtained by associating each point with one of the six unit cube oriented planes. More precisely, each point is associated with the plane that has the closest normal (i.e., maximizes the dot product of the point normal and the plane normal). The initial clustering is then refined by iteratively updating the cluster 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 packing process aims at mapping the extracted patches onto a 2D grid, while trying to minimize the unused space and to guarantee that every $T \times T$ block (e.g., 16×16 block) of the grid is associated with a unique patch.

V-PCC uses a simple packing strategy that iteratively tries to insert patches into a $W \times H$ grid. W and H are user defined parameters, which correspond to the resolution of the geometry/texture images that will be encoded. The patch location is determined through an exhaustive search that is performed in raster scan order. The first location that can guarantee an overlapping-free insertion of the patch is selected and the grid cells covered by the patch are marked as used. If no empty space in the current resolution image can fit a patch then the height H of the grid is temporarily doubled and the search is performed again. At the end of the process, H is reduced so as to account only for the used grid cells.

B. Image Generation & Padding

The image generation process exploits the 3D to 2D mapping computed during the packing process to store the geometry and texture of the point cloud as images. Fig. 14 shows an example of generated geometry and texture images.

In order to better handle the case of multiple points being projected to the same pixel, each patch is projected onto two images, referred to as layers. More precisely, let $H(u, v)$ be the set of points of the current patch that get projected to the same pixel (u, v). The first layer, also called the near layer, stores the point of $H(u, v)$ with the lowest depth D_0. The second layer, referred to as the far layer, captures the point of $H(u, v)$ with the highest depth within the interval $[D_0, D_0 + \tau]$, where τ is a user-defined parameter that describes the surface thickness.

The padding process aims at filling the empty space between patches in an attempt to generate a piecewise smooth image that may be better suited for video coding. V-PCC uses a simple padding strategy, which processes each block of $T \times T$ pixels independently. If the block is empty (i.e., all its pixels belong to the empty space), then the pixels of the block are filled by copying either the last row or column of the previous $T \times T$ block in raster order. If the block is full (i.e., does not contain any empty pixels), nothing is done. If the block has both empty and filled pixels, then the empty pixels are iteratively filled with the average value of their non-empty neighbors.

C. Auxiliary Patch and Block Information Coding

In order for the decoder to be able to reconstruct the 3D point cloud from the geometry and texture images, the following patch/block metadata information is encoded in the bitstream:

- For each patch, the index of its projection plane, its 3D location, and its 2D bounding box.
- For each $T \times T$ block, the index of the patch to which it belongs.

The patch metadata is predicted and arithmetically encoded. The block to patch mapping information, is encoded as follows: Let L be the ordered list of the indexes of the patches such that their 2D bounding box contains that block. The order in the list is the same as the order used to encode the 2D bounding boxes. L is called the list of candidate patches. The empty space between patches is considered as a patch and is assigned the special index 0. This patch is also added to the candidate patches list of all the blocks. Let I be the index of the patch to which the current $T \times T$ block belongs to and let J be the position of I in L. Instead of explicitly encoding the index I, its position J is arithmetically encoded. This can lead to better coding efficiency.

D. Occupancy Map Coding

The occupancy map consists of a binary map that indicates for each cell of the grid whether it belongs to the empty space or to the point cloud. The occupancy map compression leverages the auxiliary information described in the previous subsection, in order to detect the empty $T \times T$ blocks (i.e., blocks with patch index 0). The remaining blocks are encoded using the following process.

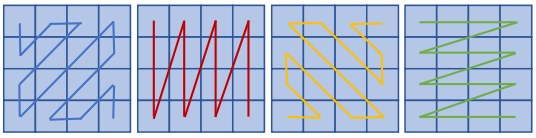


Fig. 15. Sub-blocks traversal orders.

The occupancy map could be encoded with a precision of $B_0 \times B_0$ blocks. B_0 is a user-defined parameter. In order to achieve lossless encoding, B_0 should be set to 1. In practice $B_0 = 2$ or $B_0 = 4$ result in visually acceptable results, while significantly reducing the number of bits required to encode the occupancy map.

The occupancy map compression module first associates binary values with all $B_0 \times B_0$ sub-blocks belonging to the same $T \times T$ block. A value of 1 is associated with a sub-block if it contains at least a non-padded pixel and a value of 0 otherwise. If a sub-block has a value of 1 it is said to be full; otherwise it is an empty sub-block. If all the sub-blocks of a $T \times T$ block are full then also the block is said to be full. Otherwise, the block is said to be non-full. Then, for each $T \times T$ block, a flag is arithmetically encoded that indicates whether this block is full or not. If the block is non-full, additional information indicating the location of the full/empty sub-blocks is encoded by using the following strategy. First, the encoder chooses one of the four sub-block traversal orders depicted in Fig. 15 and explicitly signals its index in the bitstream. Then, the binary values associated with the sub-blocks are ordered according to the chosen traversal order and compressed using a run-length encoding strategy.

E. Smoothing & Geometry/Texture Reconstruction

The smoothing procedure aims at alleviating potential discontinuities that may arise at the patch boundaries due to compression artifacts. The implemented approach moves boundary points to the centroid of their nearest neighbors. The point cloud geometry reconstruction process exploits the occupancy map information in order to detect the non-empty pixels in the geometry/texture images/layers. The 3D positions of the points associated with those pixels are computed by leveraging the auxiliary block/patch information and the geometry images. More precisely, let P be the point associated with the pixel (u, v), let (d_0, s_0, r_0) be the 3D location of the patch to which it belongs, and let (u_0, v_0, u_1, v_1) be its 2D bounding box. P could be expressed in terms of depth $d(u, v)$, tangential shift $s(u, v)$, and bi-tangential shift $r(u, v)$ as follows:

$$d(u, v) = d_0 + g(u, v) \tag{12}$$

$$s(u, v) = s_0 - u_0 + u \tag{13}$$

$$r(u, v) = r_0 - v_0 + v \tag{14}$$

where $g(u, v)$ is the luma component of the geometry image.

VIII. Performance Evaluation

The initial submissions to the PCC CfP were evaluated as described in Section IV-B and a selection of results is presented in Tab. I: Overall Bjontegaard-delta bitrates (BDBR) are reported for lossy coding using L-PCC, S-PCC and V-PCC. For the dynamic content in V-PCC, the results are separated in all-intra (AI) and random-access (RA) coding. It shall be noted that objective distortions introduced by V-PCC were actually so low, that there was little to no overlap with the anchor distortion values. Thus no reliable objective BDBR calculations could be performed in many cases. Nonetheless, this is an indicator that the anchor was considerably worse than the proposed technologies, and even at its highest specified rate, for a lot of the content, it could not match the performance of V-PCC at the worst specified rate. The full sets of objective and subjective results, including RD-curves, are available in [45] and [46], where S-PCC is denoted as "P02" and V-PCC as "P07".

TABLE I

Selection of Objective Evaluation Results (BDBR)

	D1	D2	Y	U	V
L-PCC lossy	-36.8%	-22.0%	N/A	N/A	N/A
S-PCC lossy	-26.9%	-8.8%	-14.7%	-78.0%	-78.3%
V-PCC AI	-100%[1]	-97.8%[2]	-84.2%[2]	-90.3%[2]	-91.9%[2]
V-PCC RA	-100%[1]	-97.8%[2]	-89.6%[2]	-96.9%[2]	-97.64%[2]

Note: Non-overlapping (1) and low overlapping (2) RD-curves lead to less reliable BDBR calculations, as described in [51].

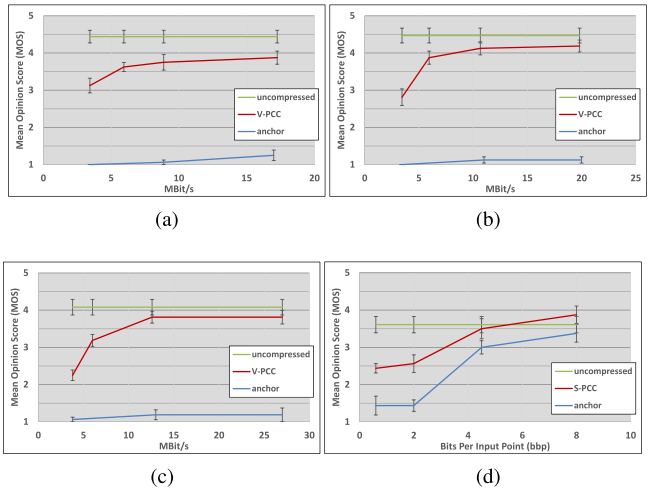


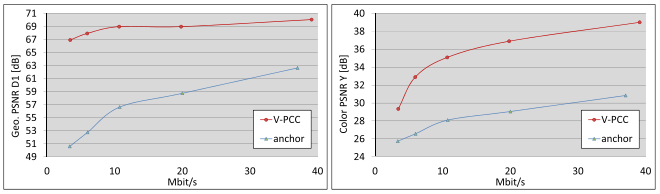
Fig. 16. Subjective evaluation V-PCC results for *RedandBlack* (a), *Soldier* (b), and *Longdress* (c), as well as S-PCC for a static frame of *Longdress* (d).

A. L-PCC Coding Performance

Only one solution was submitted as proposal for LIDAR point cloud compression. The proposal, as described in Section V, showed significant improvements over the anchor data and was consequently selected as the basis for the test model for this category. For lossless geometry compression without attributes, an encoded size of around 18 Bits per point (bpp) was achieved, which translates to a compression ratio of almost 20%. For lossy geometry without attributes, almost 40% BDBR savings for higher quality rate points was achieved. No subjective evaluation was carried out for L-PCC.

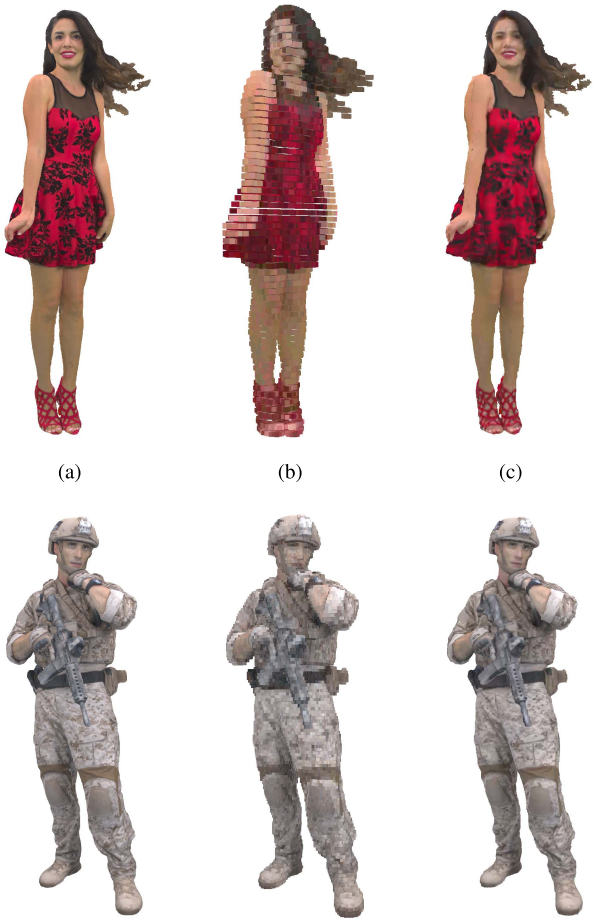
B. S-PCC Coding Performance

Three solutions were submitted as S-PCC solutions. Out of these, the proposal described in Section VI scored the highest



(a) (b)

Fig. 17. Objective metric RD-curves for sequence *Soldier*.



(a) (b) (c)

(d) (e) (f)

Fig. 18. Original (uncompressed) and reconstructed point clouds for sequences *RedAndBlack* at 3.5 MBit/s (top), and *Soldier* at 11 MBit/s (bottom). (a) Original. (b) Anchor. (c) V-PCC. (d) Original. (e) Anchor. (f) V-PCC.

in the objective and subjective evaluations. For lossy geometry and lossy attribute coding, around 30% $D1$ BDBR, 10% $D2$ BDBR, and 15% Luma BDBR bit savings were achieved, compared to the anchor. The subjective evaluation showed conclusive results, as seen in the example shown in Fig. 16d. Therefore this solution was selected to be the basis for the test model for this category.

C. V-PCC Coding Performance

A total of nine solutions were submitted as dynamic point cloud compression proposals. These submissions included several video-based solutions. Out of all submission, the proposal described in Section VII scored highest in the objective and subjective evaluations. Fig. 16a-c show the results of the subjective evaluation for three dynamic sequences compressed with V-PCC, against the anchor at different bitrate points. During the subjective evaluation, uncompressed point clouds were shown as hidden reference, thus the bitrates shown for "uncompressed" in Fig. 16 do not represent actual bitrates, but the respective target bitrate point of the test point. The benefits of V-PCC over the anchor in terms of visual quality are clearly visible and in line with the objective evaluation results, e.g. as shown in Fig. 17: Even at the lowest target point, reasonable quality was achieved, and already at the third target point the achieved quality was close to the uncompressed data. Depending on the sequence, this means compression factors between 1:100 to 1:500 are feasible. Thus this approach was selected as the basis for the test model for this category.

IX. Conclusion & Outlook

At the time of writing this paper, the standardization process is still ongoing. However, the main development orientation is set. The final standard, to be published early 2020, will consist of two classes of solutions in order to address the compression of point clouds. The first class, called video-based and equivalent to V-PCC, will leverage the usage of well-known 2D video technologies by projecting the points into 2D frames [52]. This approach is appropriate for point sets with a relatively uniform distribution of points in 3D space and clearly outperforms any state-of-the-art. For more sparse distributions, a second class is more appropriate. It is called G-PCC, for geometry-based, and is equivalent to the combination of L-PCC and S-PCC [53]. G-PCC consists of decomposing the 3D space into a hierarchical structure of cubes and encoding each point as an index of the cube it belongs to. The first class – the V-PCC – has the advantage of rapid deployment in the market and reuse of decades of technology advancements in video encoding, while the G-PCC has the advantage of a native 3D representation and the potential of improvements yet to be exploited. While relying on video coding may be a comfortable solution because it is expected that the importance of video content will be conducive to the development of even better technologies in the future, it is also clear that V-PCC has limitations in exploiting temporal correlations. The traditional motion estimation based on 2D macro-blocks is not well suited to compensate color patches having forms and locations that vary with high frequencies. Creating and arranging the patches is outside the scope of the standard; however it will not be always easy for encoders to obtain stationary patches. Future technology may consider a hybrid approach combining the two classes.

Software [54], [55] and latest performance evaluations are published through MPEG and updated regularly [56]. Interested parties are kindly invited to participate and provide their own contributions to the standardization process.

References

[1] K. Mamou, T. Zaharia, and F. Prêteux, "FAMC: The MPEG-4 standard for animated mesh compression," in *Proc. Int. Conf. Image Process.*, Oct. 2008, pp. 2676–2679.

[2] *MPEG121 Version of MPEG Standardisation RoadMap*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17332, MPEG, Jan. 2018.

[3] *Call for Proposals for Point Cloud Compression (V2)*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 16763, 3D Graphics, Apr. 2017.

[4] Google. *Draco 3D Data Compression*. Accessed: Jun. 22, 2018. [Online]. Available: https://github.com/google/draco

[5] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proc. Int. Conf. Robot. Autom.*, May 2012, pp. 778–785.

[6] 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.

[7] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–89, May 2013.

[8] 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.

[9] C. Jackins and S. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 249–270, 1980.

[10] C. Tulvan, R. Mekuria, Z. Li, and S. Laserre, *Use Cases for Point Cloud Compression*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 16331, Jun. 2016.

[11] K. Sugimoto, R. A. Cohen, D. Tian, and A. Vetro, "Trends in efficient representation of 3D point clouds," in *Proc. APSIPA ASC*, Dec. 2017, pp. 364–369.

[12] SCANLAB. *London Shipping Galleries*. Accessed: Jun. 22, 2018. [Online]. Available: http://scanlabprojects.co.uk/projects/sciencemuseum

[13] *Culture 3D Cloud*. Accessed: Jun. 22, 2018. [Online]. Available: http://c3dc.fr/

[14] *Microsoft Holoportation*. Accessed: Jun. 22, 2018. [Online]. Available: http://research.microsoft.com/en-us/projects/holoportation/

[15] *8i—Real Human Holograms for Augmented, Virtual and Mixed Reality*. Accessed: Jun. 22, 2018. [Online]. Available: http://8i.com/

[16] *Intel 360 Replay Technology*. Accessed: Jun. 22, 2018. [Online]. Available: https://www.intel.com/content/www/us/en/sports/360-replay.html

[17] R. Mekuria, C. Tulvan, and Z. Li, *Requirements for Point Cloud Compression*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 16330, Jun. 2016.

[18] *Mitsubishi Electric's Mobile Mapping System (MMS)*. Accessed: Jun. 22, 2018. [Online]. Available: http://www.mitsubishielectric.com/bu/mms/

[19] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proc. 5th High-Perform. Graph. Conf.*, 2013, pp. 73–79.

[20] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. (Feb. 2017). "Joint 2D-3D-semantic data for indoor scene understanding." [Online]. Available: https://arxiv.org/abs/1702.01105

[21] C. Früh and A. Zakhor, "Constructing 3D city models by merging aerial and ground views," *IEEE Comput. Graph. Appl.*, vol. 23, no. 6, pp. 52–61, Nov. 2003.

[22] J. Bedford, *Photogrammetric Applications for Cultural Heritage: Guidance for Good Practice*. Swindon, U.K.: Historic England, 2017.

[23] 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.

[24] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. 3rd Eurograph./IEEE VGTC Conf. Point-Based Graph. (SPBG)*, Jul. 2006, pp. 111–121.

[25] 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.

[26] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 2066–2070.

[27] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Trans. Graph.*, vol. 17, no. 2, pp. 84–115, 1998.

[28] C. Touma and C. Gotsman, "Triangle mesh compression," in *Proc. Graph. Interface Conf.*, Vancouver, BC, Canada, Jun. 1998, pp. 26–34. [Online]. Available: http://graphicsinterface.org/proceedings/gi1998/gi1998-4/

[29] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 5, no. 1, pp. 47–61, Jan. 1999.

[30] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Commun. Image Represent.*, vol. 16, no. 6, pp. 688–733, Dec. 2005.

[31] R. Mekuria, P. Cesar, and D. Bulterman, "Low complexity connectivity driven dynamic geometry compression for 3D tele-immersion," in *Proc. Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 6162–6166.

[32] R. N. Mekuria and P. S. C. Cesar, "A basic geometry driven mesh coding scheme with surface simplification for 3DTI," *IEEE COMSOC MMTC E-Lett.*, vol. 9, no. 3, pp. 6–8, May 2014.

[33] A. Doumanoglou, D. Alexiadis, S. Asteriadis, D. Zarpalas, and P. Daras, "On human time-varying mesh compression exploiting activity-related characteristics," in *Proc. Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 6147–6151.

[34] J. Hou, L. P. Chau, Y. He, and N. Magnenat-Thalmann, "A novel compression framework for 3D time-varying meshes," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2161–2164.

[35] K. Mamou, T. Zaharia, and F. Prêteux, "TFAN: A low complexity 3D mesh compression algorithm," *Comput. Animation Virtual Worlds*, vol. 20, nos. 2–3, pp. 343–354, Jun. 2009.

[36] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, and Y.-K. Wang, "Overview of the multiview and 3D extensions of high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 35–49, Jan. 2016.

[37] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. Cesar, "A 3D tele-immersion system based on live captured mesh geometry," in *Proc. 4th ACM Multimedia Syst. Conf. (MMSys)*, 2013, pp. 24–35.

[38] R. N. Mekuria, P. Cesar, and D. C. Bulterman, "Source coding for transmission of reconstructed dynamic geometry: A rate-distortion-complexity analysis of different approaches," *Proc. SPIE*, vol. 9217, Sep. 2014. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/9217/92170S/Source-coding-for-transmission-of-reconstructed-dynamic-geometry–a/10.1117/12.2063657.short?SSO=1

[39] D. Tian, H. Ochimizu, C. Feng, R. A. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3460–3464.

[40] R. Mekuria, S. Laserre, and C. Tulvan, "Performance assessment of point cloud compression," in *Proc. Vis. Commun. Image Process. (VCIP)*, Dec. 2017, pp. 1–4.

[41] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T SG16/Q6, VCEG-M33, 2001.

[42] C. Guede, J. Ricard, S. Lasserre, and J. Llach, *Technicolor Point Cloud Renderer*, document ISO/IEC JTC1/SC29/WG11 MPEG, M 40229, Apr. 2017.

[43] *Subjective Video Quality Assessment Methods for Multimedia Applications*, document ITU-T Rec. P.910, 2008.

[44] *Clarification and Corrigenda for CfP on Point Cloud Coding V2*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17091, 3D Graphics, Jul. 2017.

[45] V. Baroncini, P. Cesar, E. Siahaan, I. Reimat, and S. Subramanyam, *Report of the Formal Subjective Assessment Test of the Submission Received in Response to the Call for Proposals for Point Cloud Compression, Contribution to MPEG*, document ISO/IEC JTC1/SC29/WG11 MPEG, M 41786, Oct. 2017.

[46] M. Preda, *Merged Results of PCC CfP*, document ISO/IEC JTC1/SC29/WG11 MPEG, M 41501, Oct. 2017.

[47] 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.

[48] P. A. Chou and R. L. de Queiroz, *Transform Coder for Point Cloud Attributes*, document ISO/IEC JTC1/SC29/WG11 MPEG, M 38674, May 2016.

[49] H. S. Malvar, "Adaptive run-length/Golomb-Rice encoding of quantized generalized Gaussian sources with unknown statistics," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2006, pp. 23–32.

[50] H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proc. SIGGRAPH*, 1992, pp. 71–78.

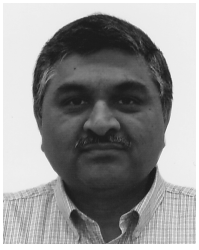
[51] A. M. Tourapis, D. Singer, Y. Su, and K. Mammou, *BD-Rate/BD-PSNR Excel Extensions*, document ISO/IEC JTC1/SC29/WG11 MPEG, M 41483, Oct. 2017.

[52] *PCC WD V-PCC (Video-Based PCC)*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17770, 3D Graphics, Jul. 2018.
[53] *PCC WD G-PCC (Geometry-Based PCC)*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17771, 3D Graphics, Jul. 2018.
[54] *PCC Test Model Cat2*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17767, 3D Graphics, Jul. 2018.
[55] *PCC Test Model Cat13*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17762, 3D Graphics, Jul. 2018.
[56] *Common Test Conditions for PCC*, document ISO/IEC JTC1/SC29/WG11 MPEG, N 17766, 3D Graphics, Jul. 2018.

Sebastian Schwarz (SM'18) received the Dipl.Ing. degree in media technology from the Technical University of Ilmenau, Germany, in 2009, and the Dr.Tech. degree from Mid Sweden University, Sweden, in 2014. Before joining Nokia, he held a Marie Skłodowska-Curie Fellowship as an Experienced Researcher with BBC R&D, London. He is currently the Research Leader of the Volumetric Video Coding Team, Nokia Technologies, Tampere, Finland. He has authored over 20 conference and journal papers on immersive media and video coding topics. He is currently an Editor of the ISO/IEC committee draft for video-based point cloud compression (ISO/IEC 23090-5) and the Co-Chair of the MPEG Ad Hoc Group on System Technologies for V-PCC.

Marius Preda received the degree in engineering from the Politehnica Bucharest, the M.B.A. degree from the IMT Business School, Paris, and the Ph.D. degree in mathematics and informatics from University Paris V. He is currently an Associate Professor with the Institut MINES-Télécom and the Chairman of the 3D Graphics Group, ISO Moving Picture Expert Group. He contributes to various ISO standards with technologies in the fields of 3D graphics, virtual worlds, and augmented reality. He leads a research team with a focus on augmented reality, cloud computing, games, and interactive media and regularly presents results in journals and at speaking engagements worldwide. He received several ISO certifications of appreciation.

Vittorio Baroncini received the Bacca Laurea degree in physics from Rome University in 1974. From 1986 to 2016, he was a Researcher with Fondazione Ugo Bordoni. From 2000 to 2008, he was the Chairman of ITU-R WP 6C (Quality Assessment). Since 2004, he has been the Chairman of the MPEG Test Group. He is an expert in objective and subjective video quality assessment. He is currently acting as a consultant in the area of visual quality assessment for EVAtech.



Madhukar Budagavi (SM'05) received the Ph.D. degree in electrical engineering from Texas A&M University. He is currently the Senior Director R&D and the Multimedia Standards Team Leader of the Standards and Mobility Innovation Lab, Samsung Research America. He represents Samsung in multimedia standards activities in MPEG, UHD Alliance, and SMPTE. He was a co-editor of *High Efficiency Video Coding (HEVC): Algorithms and Architectures* (Springer, 2014) and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, special issue on HEVC extensions and efficient implementation. He is the Co-Chair of the MPEG Ad Hoc Group on Point Cloud Coding.

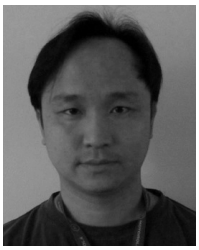
Pablo Cesar (M'17) currently leads the Distributed and Interactive Systems Group, Centrum voor Wiskunde en Infomartica (CWI). He is also an Associate Professor with TU Delft, The Netherlands. He is also a principal investigator from CWI on two H2020 projects about object-based broadcasting (2-IMMERSE) and 3D tele-immersion (VRTogether). He is a member of the Editorial Board of the IEEE MULTIMEDIA, *ACM Transactions on Multimedia*, and IEEE TRANSACTIONS OF MULTIMEDIA, among others. He has acted as an Invited Expert at the European Commission's Future Media Internet Architecture Think Tank.

Philip A. Chou (F'04) received the B.S.E. degree from Princeton University, the M.S. degree from the University of California at Berkeley, and the Ph.D. degree from Stanford University. He has been a Research Staff with AT&T Bell Laboratories, the Xerox Palo Alto Research Center, Microsoft, and Google. He has worked for start-ups Telesensory Systems, Speech Plus, VXtreme, and 8i. He has been an Affiliate Faculty Member with Stanford University, the University of Washington, and The Chinese University of Hong Kong. He is an Editor of the ISO/IEC working draft for geometry-based point cloud compression (ISO/IEC 23090-9).

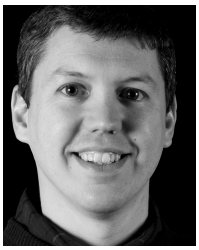
Robert A. Cohen (S'85–M'90–SM'12) received the B.S. and M.Eng. degrees in computer and systems engineering and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, MI, USA. He was a Senior Member of the Research Staff at Philips Research from 1990 to 2001 and a Principal Member of the Research Staff at Mitsubishi Electric Research Laboratories from 2007 to 2018. He is currently a Senior Research Scientist at Simon Fraser University, Canada. He is an Editor of the ISO/IEC working draft for geometry-based point cloud compression (ISO/IEC 23090-9).

Maja Krivokuća received the B.Eng. degree in computer systems engineering and the Ph.D. degree from The University of Auckland, New Zealand, in 2010 and 2015, respectively. Since then, she was with Mitsubishi Electric Research Laboratories and 8i, where she is researching new compression algorithms for 3D point clouds. He has been actively contributing to the emerging MPEG Point Cloud Compression standard. He is an Editor of the ISO/IEC working draft for geometry-based point cloud compression (ISO/IEC 23090-9).

Sébastien Lasserre received the B.Sc. degree in mathematics from the University of Strasbourg in 2000, the M.Sc. degree in mathematics from the University of Versailles St Quentin and Ecole Normale Supérieure de Lyon, France, in 2002, and the Ph.D. degree in applied mathematics from the University Pierre et Marie Curie, France, in 2005. From 2004 to 2006, he was a Monbusho Research Fellow with the Institute for Laser Engineering, Osaka, Japan. From 2007 to 2013, he was a Research Scientist at Canon Research Centre France. From 2013 to 2017, he was a Principal Scientist at Technicolor HDR, where he was involved in video and point cloud compression technologies. Since 2017, he has been the Senior Standards Manager at BlackBerry Ltd., where he continues his work on point cloud compression.



Zhu Li (SM'07) received the Ph.D. degree in electrical and computer engineering from Northwestern University in 2004. From 2012 to 2015, he was the Senior Staff Researcher/Senior Manager with Samsung Research America's Multimedia Standards Research. He is currently an Associate Professor with the Department of Computer Science and Electrical Engnieering, University of Missouri, and the Director of the NSF I/UCRC Center for Big Learning, UMKC. His research interests include point cloud and light field compression, graph signal processing, and deep learning.



Joan Llach received the M.S. and Ph.D. degrees from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 1997 and 2003, respectively. He has been a member of the Technicolor Research Center, Princeton, NJ, USA, and the Philips Research Lab, France. He is currently the Technology Area Leader of Technicolor R&I France, where he leads a research team working on compression, transmission, and interactive applications of video. He has actively participated in several standardization efforts. He is also an Editor of the ISO/IEC committee draft for video-based point cloud compression (ISO/IEC 23090-5).

Khaled Mammou received the Ph.D. degree in applied mathematics and computer science from the University of Paris V in 2008. He is currently a Senior Software Engineer with Apple Inc., where he is involved in designing and optimizing multimedia codec solutions. He has been a member of the ISO/IEC Moving Picture Experts Group (MPEG) Committee since 2005, especially focusing on 3D graphics compression. He chaired the MPEG Ad-Hoc Group on Multi-Resolution 3D Mesh Coding (MR3DMC) and significantly contributed to the standardization of the MR3DMC, Scalable Complexity 3D Mesh Compression, and Frame-based Animated Mesh Compression MPEG standards for static and animated 3D mesh compression. He is also the Co-Chair of the MPEG Point Cloud Compression Ah-Hoc Group and an Editor of the ISO/IEC committee draft for video-based point cloud compression (ISO/IEC 23090-5) and the working draft for geometry-based point cloud compression (ISO/IEC 23090-9).

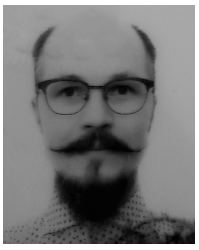
Rufael Mekuria received the M.Sc. degree from TU Delft in 2011 and the Ph.D. degree from VU University in 2017. From 2011 to 2016, he was a Researcher with the Centrum Wiskunde Informatica. In 2016, he joined Unified Streaming/CodeShop, where he is leading standardization and research efforts. He is active in MPEG, where he established the MPEG Point Cloud Coding Ad Hoc Group in 2014, and continues working on point cloud compression at Point Cloud Compression B.V., an established CodeShop spinoff.

Ohji Nakagami (M'14) received the B.Eng. and M.S. degrees in electronics and communication engineering from Waseda University, Tokyo, Japan, in 2002 and 2004, respectively. He has been with Sony Corporation, Tokyo, Japan, since 2004. Since 2011, he has been with the ITU-T Video Coding Experts Group and the ISO/IEC Moving Pictures Experts Group, where he has been contributing on video coding standardization. He is 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).

Ernestasia Siahaan (M'17) received the Ph.D. degree from the Multimedia Computing Group, Delft University of Technology, The Netherlands, in 2018. She was a Post-Doctoral Researcher with Centrum voor Wiskunde en Infomartica, Amsterdam, The Netherlands. She is currently a Consultant for the International Trade Center, Switzerland.

Ali Tabatabai (LF'17) received the bachelor's degree in electrical engineering from Tohoku University, Sendai, Japan, and the Ph.D. degree in electrical engineering from Purdue University. He was the Vice President of the Sony US Research Center, where he was responsible for research activities related to VR/AR capture and next-generation video compression. He is currently a Consultant and a Technical Advisor to the Sony US Research Center and the Sony Tokyo R&D Center. He is an Editor of the ISO/IEC committee draft for video-based point cloud compression (ISO/IEC 23090-5).

Alexis M. Tourapis received the Ph.D. degree in electrical engineering from The Hong Kong University of Science and Technology in 2001. He has held positions at Microsoft, Thomson Research, DoCoMo USA Labs, and Dolby Laboratories, among others. He has made several key contributions to standards, such as MPEG-4 AVC and HEVC. He is currently a Software Standards Engineer with Apple Inc. and represents Apple in several standardization activities, including MPEG, ITU, and SMPTE. His research interests span video/image processing and compression, high-dynamic-range imaging, machine learning, and point cloud compression, among others.



Vladyslav Zakharchenko received the Ph.D. degree in optics and photonics from the National Technical University of Ukraine, Kyiv, in 2012. He was with Samsung Electronics and AMD, where he was involved in multimedia technologies. He is currently a Principal Engineer and the Point Cloud Coding Group Leader of Futurewei Technologies, Inc., Santa Clara, CA, USA. His work is devoted to innovations for volumetric media and close collaboration with international standard committees. He is 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).