Hybrid Octree-Plane Point Cloud Geometry Coding

Antoine Dricot
Instituto de Telecomunicações
Lisboa, Portugal
antoine.dricot@lx.it.pt

João Ascenso
Instituto Superior Técnico - Instituto de Telecomunicações
Lisboa, Portugal
joao.ascenso@lx.it.pt

***Abstract*—Point clouds provide an efficient way of representing 3D data for applications such as virtual or augmented reality, free-viewpoint video, and gaming. However, this rich, realistic and immersive representation demands a very high amount of data and thus efficient point cloud coding solutions are increasingly needed. A promising way to represent point clouds is through octree partitioning, which provides good compression performance with low complexity, as well as level-of-detail scalability. However, other data representations may offer additional benefits to the octree data structure. The goal of this paper is to propose a point cloud geometry coding solution that leverages the adaptive octree partitioning, and enhances it with a novel coding mode. This mode exploits a plane representation for leaf nodes at different layers of the octree. This corresponds to a hybrid solution where two point cloud representation models are combined. The proposed approach can outperform octree based solutions that are currently considered for standardization and it also provides significant compression gains against a static octree solution (average BD-rate gains of 35% are reported).**

Keywords—Point Cloud Coding, Octree, Plane

I. INTRODUCTION

Recently, the major trend in multimedia technologies has been to increase the immersion, realism and faithfulness of visual experiences, leveraging on advances in the sensors, processing algorithms, and displays. Nowadays, Point Clouds (PC) are recognized as very promising for the next generation of 3D content. A PC consists of a set of 3D points with some attributes that represent the surface of a 3D scene or object and its characteristics; an example is illustrated in Fig. 1. The 3D coordinates (x, y, z) of each point correspond to the geometry component of the PC, that may contain one or more additional components (attributes); the most common attributes are color, reflectance, and normal vectors. There are two types of point clouds: a static PC which represents a single time instant, and a dynamic PC which is a sequence of static PCs representing the temporal evolution of the visual scene.

The realism and immersiveness provided by the 3D point cloud representation comes at the cost of a very high amount of data, and thus, efficient coding solutions are needed. There have been recent efforts by the industry and academia towards the development of high performance point cloud coding solutions that allow an efficient transmission and storage and address other requirements of immersive media applications. Following these efforts, the MPEG group has initiated the standardization of Point Cloud Coding (PCC) solutions by issuing a Call for Proposals in January 2017 [1]. At this stage, two main solutions are under development: Video-PCC (V-PCC) [2], that uses 3D-to-2D video projection to leverage the performance of available 2D video codecs such as the High Efficiency Video Coding (HEVC) [3]; and Geometry-PCC (G-PCC) [4], that relies on the octree structure, often used in the computer graphics community.

Fig. 1. Rendering of longdress [1]: geometry; and geometry with color.

Octrees are widely recognized as a promising representation for point clouds geometry coding. The octree partitioning can be rigid (i.e. with a fixed target depth), or adaptive [5]; in the last case a recursive split decision process is necessary to adjust the depth of the partitioning depending on the input content characteristics. This flexible data structure relies on a concept akin to the quadtree partitioning that is common in 2D video coding, and has been a key factor to enhance the efficiency of standard video codecs (e.g. HEVC). Besides the partitioning, the addition of new modes (e.g. Intra) with different ways to perform prediction has also brought significant RD performance improvements. The objective of this paper is to propose a novel coding mode for static point cloud geometry coding that relies on plane estimation and reconstruction to fit the surface represented by the input points to be coded. This corresponds to a hybrid solution which combines octrees with planes, i.e. with geometric modelling of the point cloud surfaces.

This paper is organized as follows. Section 2 provides a brief review of the main types of PCC techniques from the literature, and Section 3 describes the adaptive octree-based partitioning with more detail. Section 4 describes the proposed coding solution. Finally, Section 5 provides the experimental performance assessment of the solution, while Section 6 concludes the paper.

II. POINT CLOUD CODING: BRIEF OVERVIEW

Several point cloud coding solutions have been proposed in the past, based on data structures such as graphs, patches or octrees targeting different kinds of applications.

In many solutions, the point clouds surface can be divided into clusters of points, each represented by a patch [2][6][7][8]. Patches are then projected (or mapped) onto 2D frames which are encoded as traditional images or videos. In [9], the points are clustered based on the RANSAC algorithm, fitted to 3D planes, and the resulting grid is coded with a 2D-DCT transform. This approach can be applied to all the different components of static and dynamic point clouds.

Another form of point cloud representation is the graph which characterizes the relationship between a point and its neighbors. Graphs were exploited for attribute coding, both for static [10][11] and dynamic [12] point clouds. In these solutions, geometry and attributes are considered signals on the graph and the efficient graph transform (which is

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

equivalent to the Karhunen-Loève transform on graphs) can efficiently exploit redundancy between neighboring points by decorrelating the data to be coded.

Another approach for PCC is to rely on a tree data structure. Several alternatives have been proposed, such as binary trees [13] or spanning trees [14], however, many of the solutions available in literature are based on octrees [4][5][15]. Spatial [16] and temporal [17][18] prediction strategies have been proposed to further increase compression efficiency. In [19], an enhancement layer is proposed on top of the octree, where the local surfaces are projected onto planes and the residual information is entropy coded.

III. Octree based Point Cloud Coding

In the context of PCC, octree structures can be used to divide the 3D space into increasingly smaller volumes. An example of octree and its associated 3D structure is shown in Fig. 2 (top). The bounding box of the main volume corresponds to the node in the root layer (or depth) of the tree and includes all the points. First, the root node is split into 8 octants associated to the 8 children nodes in the second layer. Then, every occupied octant (i.e. that contains at least one point) at every depth is further split, until the target depth is reached. All the points are therefore organized in the leaf nodes (i.e. nodes of the target layer with no children).

The octree structure can be transmitted to the decoder as a stream of occupancy bytes; a byte for each occupied octant indicates the occupancy of its children, with one bit per node, set to 1 if occupied or to 0 if empty. The decoded PC corresponds to the set of points reconstructed at the center of the occupied octants associated to the leaf nodes. The occupancy bytes are arithmetically coded to further increase compression efficiency.

The intrinsic resolution of a PC is defined as 2^p, where p is the number of bits (or precision) per spatial coordinate (x, y, z). Therefore, octree encoding is by definition lossless if the target depth of the tree is equal to p. If the target depth is smaller than p, the use of an octree structure can be seen as a form of geometry quantization. This case, illustrated in Fig. 2 (middle), is usually referred to as a pruned octree. Additional information can be coded to increase the precision of the pruned octree, for example in the form of residual distances between the center of the octant and the original points [1][17], or alternatively as vertices, representing the intersection of some polygonal surface (estimated from the original points) and the edges of the octant [4].

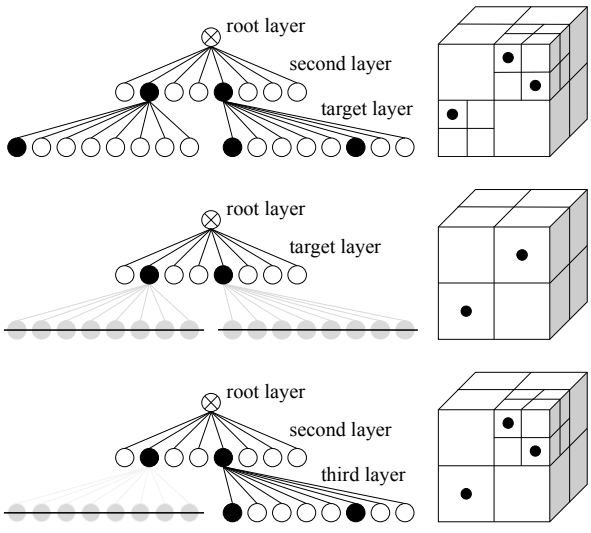

Fig. 2. *top*: Octree partitioning; *middle*: Pruned; *bottom*: Adaptive.

The octree partitioning may also be adapted to the input point cloud characteristics [5] (e.g. density) by allowing leaf nodes at different depths, smaller than or equal to the target depth, as illustrated in Fig. 2 (bottom). For each octant, a decision is taken between two alternative coding options: leaf or split. In practice, a suitable split decision process must be adopted to manage the rate-distortion trade-off, as usual in 2D video coding and now being extended to point cloud coding.

IV. Adaptive planes for Point Cloud geometry coding

The objective of the solution proposed in this paper is to increase the efficiency of static point cloud geometry coding. For point clouds with high precision, octree based methods require a lot of subdivisions (i.e. larger depths) and thus have a low compression efficiency. The idea in this paper is to model PC surfaces with a simple plane as the 3D geometric primitive that can be fitted locally and thus providing a better approximation for more complex surfaces.

A. Point cloud coding architecture

Fig. 3 shows the architecture of the proposed hybrid coding solution which exploits the octree and plane representation paradigms. Points are organized into an adaptive octree data structure, and local surfaces are modelled with a plane as a geometric primitive.

First, the octree is built with a target depth, as described in Section 3. Next, each octant is compressed with three coding modes: split mode, center mode, and the novel plane mode. The octants are compressed in recursive order (from bottom to top) so that the decisions made for the children are considered when testing the split mode. The decision of which octants are split and which are not (thus becoming early leaf nodes) allows to adapt the partitioning to the input content [5]. For every mode, the decoded points within the compressed octant are reconstructed and the distortion D is estimated using the symmetric point-to-point mean square error (MSE). The point-to-point MSE $d(O_{org}, O_{rec})$ corresponds to the average square distance of the N points p from the set of original points O_{org}, against their respective nearest neighbor nn from the set of reconstructed points O_{rec}, as shown in (1). When the number of points in O_{org} and O_{rec} is different, D is the maximum between $d(O_{org}, O_{rec})$ and $d(O_{rec}, O_{org})$ as shown in (2).

$$d(O_{org}, O_{rec}) = \frac{1}{N} \sum_{\substack{p \in O_{org}, \\ nn \in O_{rec}}} \begin{pmatrix} (x_p - x_{nn})^2 \\ +(y_p - y_{nn})^2 \\ +(z_p - z_{nn})^2 \end{pmatrix} \quad (1)$$

$$D = \max\left(d(O_{org}, O_{rec}), d(O_{rec}, O_{org})\right) \quad (2)$$

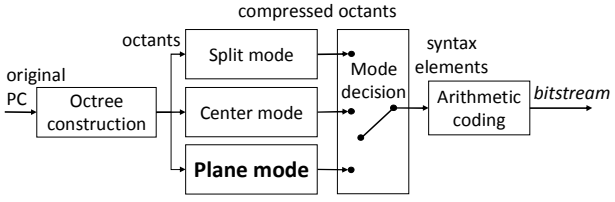

Fig. 3. Architecture of the proposed adaptive plane based point cloud geometry coding solution (AP-PCC).

For each octant, the coding mode that minimizes the distortion is selected in the mode decision module and the associated syntax elements are arithmetically coded. The main contribution of this paper is the novel plane mode which provides an alternative to the original center mode (i.e. one point only reconstructed at the center of the octant), efficiently representing the points in early leaf nodes with a plane.

B. Plane mode

The architecture of the plane mode at the encoder is shown in Fig. 4. First, a plane is estimated to fit the surface represented by the points in the octant. Then, the edge points corresponding to the intersections between the estimated plane and the edges of the octant are coded. Finally, the plane is re-estimated from these edge points and reconstructed with a suitable number of points. This allows to estimate distortion accurately to select the best mode.

For each octant of the full-depth octree, the following is performed for the plane mode:

- **Plane estimation**: The goal of this module is to estimate a plane that fits the surface represented by the original 3D points of the octant (Fig. 5 top-left). The equation of the plane is solved by linear regression along the axis corresponding to the main orientation of the surface (i.e. the component x, y, or z of the surface normal that have the largest absolute value). The output of this module is a set of three edge points with integer coordinates (i.e. floating point values along the main axis are rounded), which are the intersections of the plane with three edges of the octant (Fig. 5 top-right). The edge points are then coded and used for the plane reconstruction.

- **Edge occupancy coding**: In this module, the edges associated to the three edge points are coded and transmitted to the decoder. Thus, a 12-bit symbol is arithmetically encoded, with each bit signaling the occupancy of one of the twelve edges of the octant, i.e. set to 1 if occupied by an edge point and 0 otherwise. This symbol can have at most three bits set to 1, corresponding to the three edge points. The probability tables for the arithmetic coding of this symbol are initialized as equiprobable, updated with each symbol coded, and reset at the beginning of each layer of the octree (following the layered arithmetic coding method from [5]).

- **Edge position coding**: For each of the edges signaled as occupied, the associated edge point position is arithmetically encoded as a d bit symbol, where d is equal to the absolute difference between the current depth of the octant and the precision of the input PC. This allows to have a resolution for the edge position that depends on the size of the octant. The edge position symbols are also encoded using the layered arithmetic coding method. Both the edge occupancy and position coding are lossless. Using this approach, it is not necessary to transmit the (x, y, z) position of each edge point,

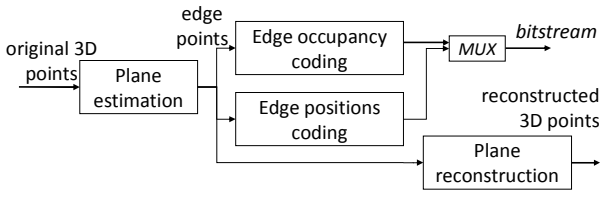

Fig. 4. Architecture of the proposed plane mode at the encoder.

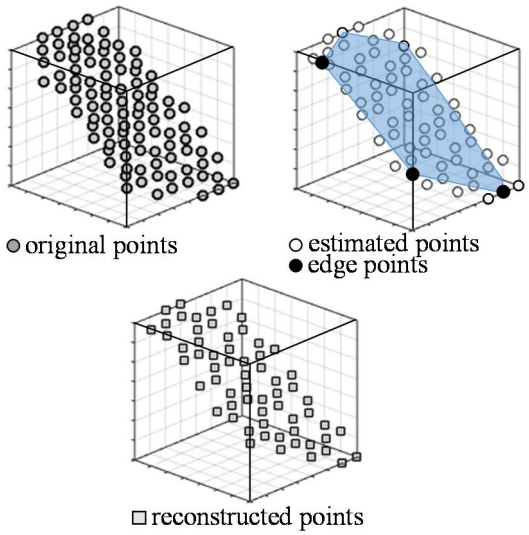

Fig. 5. Example of an octant compressed with the plane mode.

but just its relative position on the edge, thus saving some bitrate. This approach is also rather efficient compared to a plane parametric representation which requires the transmission of floating-point values for each parameter or the usage of some (lossy) quantization process.

- **Plane reconstruction**: This module is available both at the encoder (for distortion estimation purpose) and decoder (to obtain the decoded point cloud). First, the plane is re-estimated with the same process as in the plane estimation module, this time only with the three edge points as input instead of the original points (which are not available at the decoder). Then, in its simple form, plane reconstruction just corresponds to the creation of points that belong to this plane (Fig. 5 bottom). Different point clouds may have varying surface thickness and density (i.e. distance between neighboring points). Therefore, to avoid a dramatic increase of the number of decoded points, the (global) density of the input PC is measured by computing the median of the distance of every point to its closest neighbor; this density value is transmitted to the decoder. The plane is filled (like a grid) with reconstructed points at integer positions, spaced by a regular interval corresponding to the density value. For content with thick surfaces, a single plane reconstructed with these points is not enough to represent the underlying surface. In this case, instead of only rounding the floating point coordinates along the main axis, two points are generated, one by using the math floor operation (greatest integer smaller than or equal to the input real value), and another one with the ceil math operation (smallest integer greater than or equal to the input real value). This allows to reduce the geometric error at the expense of having more decoded points, which is especially important when the surfaces have some thickness. The two proposed variations are respectively referred to as AP-PCC and AP-PCC-Thick in the following.

V. PERFORMANCE ASSESSMENT

This section presents the test conditions that were used to evaluate the performance of the proposed coding solution, and the results regarding the relevant benchmark.

A. Test conditions

The experiments are conducted on 2 point clouds with a resolution p=10 bits per component (vox10), and 5 point clouds with p=12 (vox12), from the MPEG PCC repository [1]. The objective performance assessment is based on Rate-Distortion (RD) performance. The rate is reported as bits per point (bpp), computed by dividing the total rate by the number of input points. The distortion is reported as the peak signal to noise ratio (PSNR) of a point-to-point error metric adopted by MPEG [1][20]. The anchor used as a benchmark is the pure octree-based coding solution of [5]. The main parameter to obtain different levels of quality in octree based solutions is the maximum depth, that varies in the range {7, 8, 9, 10, 11}. In AP-PCC and AP-PCC-Thick, octants at depths lower than 6 are always split, while mode decisions are enabled for larger depths. The proposed solutions are compared to two other benchmarks: MPEG Test Model 1 (TMC1), that uses a pruned octree with additional coding tools representing the points as vertices of polygonal surfaces; and the more recent MPEG Test Model 13 (TMC13), using only octree encoding for geometry (with results obtained from [21] and [22] respectively). The RD performance against the anchor of [5] is measured with the Bjøntegaard Delta (BD) rate metric [23].

B. Results and analysis

BD-rate results for the vox12 content are shown in Table 1. Average gains of 35% (up to 68%) are reported for the proposed AP-PCC, while TMC13 and TMC1 respectively provide 25% and 48% gains. The RD performance results are analyzed in the following.

The gains of TMC13 over the anchor are mostly due to advanced entropy coding tools for the occupancy bytes. TMC1 uses the same entropy coding tools as TMC13, and although older, has a better RD performance (for geometry) due to the usage of polygonal surfaces. AP-PCC uses a simple layered arithmetic coding which may justify some lack of efficiency in comparison to TMC1/TMC13. Therefore, it is expected that a combination of the proposed AP-PCC solution (including the plane mode and mode decision modules) with more advanced and mature entropy coding tools would further increase the compression efficiency.

AP-PCC is outperformed in average by TMC1, mainly because of its low performance on content with a lower density, for example with only 5% gains on *Egyptian_mask*, which is very sparse (with a median distance of 10 between points). The proposed solution performs significantly better for point cloud content with high density (such as Facade and Frog), for which the plane reconstruction is more accurate. The denser the surface is, the better it can be represented with a plane primitive.

TABLE I. BD-RATE (%) RESULTS ON TARGET DEPTHS {7, 8, 9, 10} FOR *VOX12* CONTENT.

Content (and median point to point distance)		TMC13 [22]	TMC1 [21]	AP-PCC
Arco_Valentino	(3)	-19	**-38**	-19
Egyptian_mask	(10)	-23	**-49**	-5
Facade	(2)	-28	-35	**-48**
Frog	(2)	-35	-61	**-68**
Statue_Klimt	(4)	-21	**-60**	-35
Average		**-25**	**-48**	**-35**

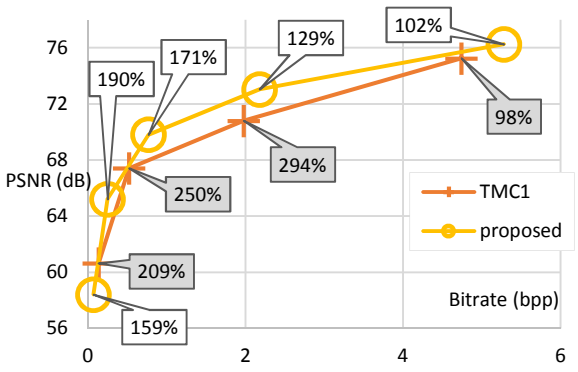

Fig. 6. RD performance on *Facade* (vox12) and ratios of numbers of decoded over original points.

The high performance of TMC1 for vox12 content comes at the cost of a large increase of the number of points (142% in average) with inconsistent variations, as illustrated in Fig. 6, which shows the RD performance on *Facade* along with the associated ratio of decoded/original points. The proposed solution with its adaptive density provides a smaller increase (125% in average) with less inconsistent variations.

On the vox10 dataset, AP-PCC (without adaptive thickness) is outperformed by TMC1, as shown in Fig. 7, especially for the *queen* point cloud. This dataset contains objects with a thicker surface, which negatively impacts the ratio of decoded over original points. In this case, this ratio is much lower for AP-PCC (e.g. 57% to 65% for *queen*) than for TMC1 (82% to 117%). The proposed method to model the thickness of the reconstructed planes (AP-PCC-Thick) increases this ratio up to an acceptable value (approximately 120%), with AP-PCC-thick having a significantly better RD performance than TMC1 for several cases.

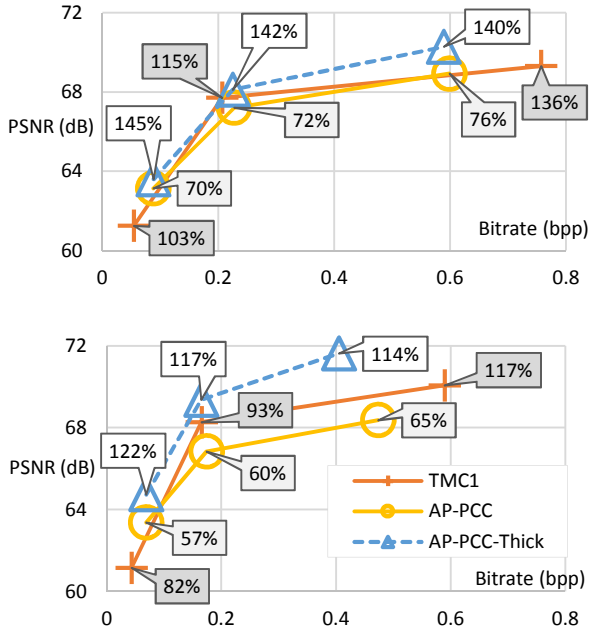

Fig. 7. RD performance on *longdress* (*top*) and *queen* (*bottom*) (vox10) and ratios of numbers of decoded over original points.

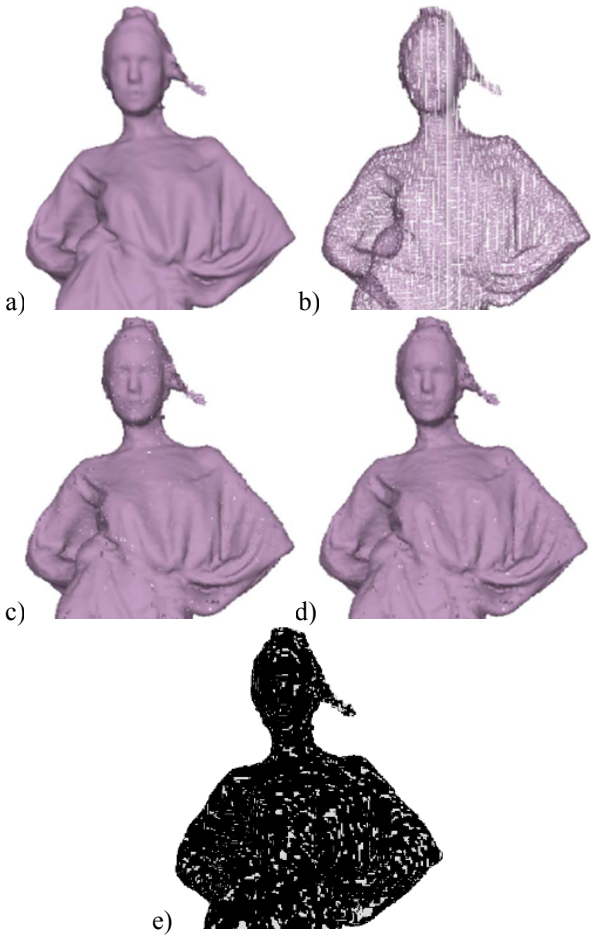

Fig. 8. Rendering of *longdress* a) original; and compressed at 0.6 bpp with: b) anchor (63.2 dB); c) AP-PCC (68.9 dB); d) AP-PCC-Thick (70.3 dB); e) mode decision map (dark: plane, light: center).

Fig. 8 shows the visual improvement provided by the proposed solutions over the anchor and the associated objective quality for a given bitrate (approximately 0.6 bpp). The level of detail is clearly increased, with a noticeable lower number of missing points. Moreover, the mode decision map illustrates how AD-PCC adaptively selects the plane mode when appropriate, while going back to the original split mode and center mode when the plane reconstruction cannot fit the surface with a sufficient quality level, i.e. being adaptive to the characteristics of the input content.

VI. Conclusions

This paper proposes to model the point cloud surface with local planes and exploits this representation in the context of an octree based static point cloud geometry codec. This solution leverages the adaptive octree partitioning by enabling a novel plane coding mode for leaf nodes at different depths of the octree. A mode decision module is proposed to select the best coding mode for each octant based on a distortion minimization. This approach offers a significant increase in coding efficiency against the pure octree anchor and can outperform some octree based solutions that are nowadays considered for standardization. Average BD-rate gains of 35% (up to 68%) are reported. Future work may include the improvement of the mode decision process based on a rate-distortion trade-off, as well as a combination with more advanced entropy coding tools.

References

[1] MPEG-3DG, “Call for proposals for point cloud compression”, ISO/IEC JTC1/SC29/WG11/MPEG N16732, January 2017.

[2] MPEG-3DG, “V-PCC Test Model v4”, ISO/IEC JTC1/SC29/WG11/MPEG W17996, October 2018.

[3] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 22, pp. 1649-1668, September 2012.

[4] MPEG-3DG, “G-PCC Test Model v4”, ISO/IEC JTC1/SC29/WG11/MPEG W17994, October 2018.

[5] A. Dricot, F. Pereira, J. Ascenso, “Rate-distortion driven adaptive partitioning for octree-based point cloud geometry coding”, IEEE International Conference on Image Processing, Athens, Greece, October 2018.

[6] J. Digne, R. Chaine, and S. Valette, “Self-similarity for accurate compression of point sampled surfaces”, Computer Graphics Forum, Vol. 33, No. 2, pp. 155-164, May 2014.

[7] T. Golla, and R. Klein, "Real-time point cloud compression." Intelligent Robots and Systems, Hamburg, Germany, September 2015.

[8] T. Ochotta, and D. Saupe, “Image-based surface compression”, Computer Graphics Forum, Vol. 27, No. 6, pp. 1647-1663, September 2008.

[9] X. Zhang, W. Wan, and X. An, “Clustering and DCT Based Color Point Cloud Compression”, Journal of Signal Processing Systems, vol. 86, no. 1, pp. 41-49, January 2017.

[10] R. Cohen, D. Tian, and A. Vetro, “Attribute compression for sparse point clouds using graph transforms”, IEEE International Conference on Image Processing, Phoenix, AZ, USA, September 2016.

[11] C. Zhang, D. Florencio, and C. Loop, “Point cloud attribute compression with graph transform”, IEEE International Conference on Image Processing, Paris, France, October 2014.

[12] D. Thanou, P. Chou, and P. Frossard, “Graph-based compression of dynamic 3D point cloud sequences”, IEEE Transactions on Image Processing, Vol. 4, No. 25, pp. 1765-1778, February 2016.

[13] Y. Shao, Z. Zhang, Z. Li, K. Fan, and G. Li, “Attribute compression of 3D point clouds using Laplacian sparsity optimized graph transform”, IEEE Visual Communications and Image Processing, St Petersburg, FL, USA, December 2017.

[14] S. Gumhold, Z. Kami, M. Isenburg, and H. Seidel, “Predictive point-cloud compression”, ACM SIGGRAPH Sketches, Los Angeles, CA, USA, July 2005.

[15] R. Schnabel, and R. Klein, “Octree-based point-cloud compression”, Eurographics Symposium on Point-Based Graphics, Boston, MA, USA, September 2006.

[16] R. Cohen, D. Tian, and A. Vetro, “Point cloud attribute compression using 3-D intra prediction and shape-adaptive transforms”, IEEE Data Compression Conference, Snowbird, UT, USA, March 2016.

[17] R. Mekuria, K. Blom, and P. Cesar, “Design, implementation, and evaluation of a point cloud codec for tele-immersive video”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No. 27, pp. 828-842, March 2017.

[18] J. Kammerl, N. Blodow, R. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, “Real-time compression of point cloud streams”, IEEE International Conference on Robotics and Automation, St Paul, MN, USA, May 2012.

[19] K. Ainala, R. Mekuria, B. Khathariya, Z. Li, Y. Wang, and R. Joshi, “An improved enhancement layer for octree based point cloud compression with plane projection approximation”, Applications of Digital Image Processing XXXIX, vol. 9971, p. 99710R, September 2016.

[20] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, “Geometric distortion metrics for point cloud compression”, IEEE International Conference on Image Processing, Beijing, China, October 2017.

[21] MPEG-3DG, “Anchors for PCC TMC1”, ISO/IEC JTC1/SC29/WG11/MPEG N17358, January 2018.

[22] MPEG-3DG, “PCC TMC13 performance evaluation and anchor results”, ISO/IEC JTC1/SC29/WG11/MPEG N17768, July 2018.

[23] G. Bjontegaard, “Calculation of average PSNR differences between RD-curves”, ITU-T SG1 VCEG Meeting, Austin, TX, USA, January 2001.