ADAPTIVE PLANE PROJECTION FOR VIDEO-BASED POINT CLOUD CODING

Eurico Manuel Rodrigues Lopes

Electrical and Computer Engineering
Instituto Superior Técnico
Lisbon, Portugal
eurico.lopes@tecnico.ulisboa.pt

ABSTRACT

One of the most promising emerging 3D representation paradigms is the point cloud model, notably due to the new set of applications enabled, from immersive telepresence to 3D geographic information systems. Recognizing the potential of this representation model, MPEG has launched a standardization project to specify efficient point cloud coding solutions [1]. This project has given rise to the so-called Video-based Point Cloud Coding (V-PCC) standard, which projects a dynamic point cloud geometry and texture into a sequence of images to be coded with HEVC. In V-PCC, this projection is always performed using the same projection planes, independently of the point cloud. This paper proposes a more flexible coding solution, which adopts the V-PCC architecture but selects a different set of projection planes more adapted to the characteristics of the point clouds, to obtain a better compression performance. In this paper, this tool is applied to static point clouds but it may be extended to dynamic point clouds. The experimental results show an improvement of the geometry compression performance regarding V-PCC, especially for medium to larger rates.

Index Terms— *Point Clouds*, Video-based Point Cloud Coding, Projection Planes, Static Point Clouds.

1. INTRODUCTION

With the increased popularity of 3D-based applications, notably in gaming, the interest in point cloud representation has significantly increased. A point cloud is a set of points defined by their 3D coordinates (x,y,z), corresponding to the so-called *geometry*, which may be complemented with attributes, such as color, normal vectors, reflectance, among others. Point cloud coding involves coding at least its geometry, and optionally some attributes, and may target still frames, i.e. static point clouds, or a sequence of point cloud frames, i.e. dynamic point clouds. This type of 3D model is able to efficiently represent the visual data while providing the functionalities required for many applications such as geographic information systems, cultural heritage, and immersive telepresence. Typically, 3D data acquisition solutions can produce 3D+t spatio-temporal models with millions of 3D points per second and, thus, it is critical to have available efficient point cloud coding solutions to enable the targeted applications.

Recognizing the industry need to have efficient coding solutions, which should enable the storage and transmission of 3D data, notably point clouds, JPEG and MPEG have launched standardization projects to specify new, appropriate coding formats. In January 2017, MPEG has issued a Call for Proposals on Point Cloud Compression (PCC) [1], which targets the efficient representation of static objects and scenes, as well as dynamic and real-time environments, represented using point clouds (PC). Following this call, two PC coding solutions have been developed, notably Video-based Point Cloud Coding (V-PCC) [2] for dynamic content and Geometry-based Point Cloud Coding (G-PCC) [3] for static and dynamically acquired content. JPEG has launched the JPEG Pleno project which also targets PC coding but it is still at an earlier stage.

V-PCC has adopted a coding paradigm where the PC is segmented into smaller parts, close to surface component, which are projected onto a set of planes to derive the so-called geometry and texture maps. These maps may be coded as independent images (intra mode) or a correlated sequence of images (inter mode) using an available video coding standard, e.g. HEVC. The V-PCC 3D to 2D projection is performed using always the same set of projection planes without adapting them to the PC characteristics. This may cause some occlusions during the projection, as well as a larger geometry map variance.

In this context, this paper proposes a variation of the V-PCC coding solution for static PC where the projection planes are flexible in number and orientation and are thus able to adapt to the PC characteristics, targeting improving the final rate-distortion (RD) performance. The obtained results show promising gains for geometry coding.

To achieve its objectives, this paper is organized as follows: Section 2 reviews the most relevant PC coding solutions; Section 3 describes V-PCC overall architecture; Section 4 details the proposed solution; Section 5 presents performance assessment; and Section 6 closes this paper with some final remarks.

2. RELATED WORK

The most popular PC coding solutions, notably at the start of the MPEG PCC project, was clearly the Point Cloud Library (PCL) solution [4]. In this solution, the PC is organized as an octree, which is a tree data structure, where each branch node has up to eight children – one for each sub-octant (also called voxel), starting from the initial node associated to the full PC bounding box. The color of each voxel is the average color of its points. However, other PC coding solutions are available in the literature. In [5], *Zhang et al.* proposed a graph-based coding solution (later extended by *Cohen et al.* [6]), which still uses an octree structure, but improves the rather simple PCL-attributes coding. In the proposed solution blocks of $k \times k \times k$ voxels are assembled to create a graph structures. For each graph structure, a graph-based transform is derived to be applied to the corresponding set of colors within the respective block. In [7], *Mekuria et al.* extended

the PCL-based coding solution by mapping the vertex color into an image grid and encoding the corresponding blocks with the most popular image coding standard, JPEG.

Another coding solution, closely related to V-PCC, has been proposed by *Golla et al.* [8], where the PC is segmented into chunks of data and each chunk is further sub-divided into an octree structure. Each voxel, including multiple points, is represented as a patch, which is characterized by its reference position, size and orientation. To represent the patch geometry, height and occupancy maps are generated, accounting for the represented points original geometry offsets from the patch reference position, and for the holes in the geometry, i.e. pixels in the height maps that do not correspond to points, respectively. Likewise, additional maps may be generated for texture and other attributes. The height and attribute maps are encoded using a standard image coding standard, e.g. JPEG, while the occupancy maps are encoded with lossless JBIG2 [9].

3. VIDEO-BASED POINT CLOUD CODING (V-PCC)

Since the solution proposed in this paper is a V-PCC improvement, it is appropriate to briefly review this emerging PC coding standard.

The original PC is decomposed into the minimum necessary number of patches with smooth boundaries. This is achieved by first computing the normals at every point and clustering these normals into six clusters according to the maximization of the dot product to one of six pre-defined projection planes, each with a different orientation along the 3D axis (X, -X, Y, -Y, Z and -Z respectively). The patches are defined by applying a connected component extraction procedure [2], and after projected to generate the 2D patch depth maps, which represent the PC geometry. The set of patches (geometry and texture) are after packed into a single 2D map using a packing strategy which tries to iteratively insert patches in a W×H grid, while minimizing the unused space, and ensuring that every T×T block (e.g. 16×16) in the grid is associated with one single patch, where *T*, *W* and *H* are user-defined parameters. The result is a so-called occupancy map, indicating for each packing grid block (B0×B0 pixels) if it belongs or not to a PC patch.

Next, the image generation process takes place, where the strategic insertion of patches to a 2D grid performed during the packing process is exploited to construct the geometry image (from the patches depth maps) and texture image. As the decoded point clouds may be lossy (regarding the number of points and their positions), the texture generation process must have 'access to' the reconstructed geometry to compute the colors associated with the re-sampled points. Naturally if dynamic PC coding is targeted, one geometry image and one texture image are generated for each PC frame and the correlations among the 2D frame sequence are exploited by HEVC. It is worth mentioning that the reconstructed geometry goes through a smoothing process to avoid discontinuities that may exist at the patch.

Between the patches in the depth and texture map, there are empty spaces that are filled by a so-called *padding process*, which will generate a piecewise smooth image for video compression. The occupancy maps dictate which pixels correspond to points. The successive generated geometry and texture maps/images are coded as video frames using the HEVC video coding standard. While the texture video is YUV420 with 8-bit depth, the geometry video is monochromatic and also with 8-bit depth. Finally, all generated coded streams are multiplexed into a single compressed bit stream.

4. PROPOSING V-PCC WITH ADAPTIVE PLANE PROJECTIONS

This section proposes a so-called Adaptive Plane Projection for V-PCC (AP2V-PCC) tool. Since V-PCC uses a set of six pre-defined planes for the initial points clustering and later a set of three pre-defined projection planes (intimately related to the respective clustering planes) to project the geometry and texture into the 2D depth and texture maps, there is no consideration in this process on the specific PC characteristics. If the PC set of normals is not well aligned with the global coordinate system (x,y,z) and, as a consequence, with the clustering and projection planes, the angular differences between the V-PCC projection planes and the respective clusters set of normals may be large, meaning that the projection planes are not well adapted to the input data. This has two main consequences: i) the projected geometry map values will have a larger variance, which potentially increases the geometry rate; and ii) some points may be occluded during the projection, which has a negative impact both in objective and subjective qualities.

Bearing these issues in mind, the proposed AP2V-PCC approach adopts an adaptive plane projection paradigm, where the three V-PCC fixed planes are replaced by more 'friendly' planes in the sense that: i) reduce the depth maps variance since the PC main surfaces should be more aligned/parallel to the adaptive projection planes and, therefore, potentially reduce the geometry rate; ii) reduce the number of points overlapping in the same geometry map pixel, since the angular difference between the points and their projection plane is reduced.

4.1. AP2V-PCC Architecture and Walkthrough

The AP2V-PCC approach implies significant changes in the patch generation process, which are illustrated in Figure 1. The novel tools regarding the initial V-PCC patch generation process, i.e. K-Means Clustering, Adaptive Planes Selection and Coordinates Transformation, and Adaptive Resolution Control, are highlighted in red and detailed in the next sub-section. The AP2V-PCC patch generation process is now described

The normals are calculated for each point by applying Principal Component Analysis (PCA) over the 16 closest neighbors and assigning the vector with lower variance as the respective point normal. The calculated normals are then segmented into K (user-specified) clusters according to the maximization of the dot product to one of the K clusters normal centroids. Then, the just created clusters are segmented into sub-clusters of points that are spatially connected (and belong to the same cluster). The sub-clusters with less than 16 points are filtered, since the encoded metadata of these sub-clusters would be very expensive in terms of bits per point when compared to sub-clusters with more points.

Next, the projection plane is chosen for each sub-cluster, and the respective local coordinate system is calculated. In practice, the projection plane is defined by setting an appropriate (Δ) depth axis, which in this case is computed as the average sub-cluster set of normals. The tangent (S) axis is computed as $S = \Delta \times (0;-1;0)$, while the bi-tangent is computed as $R = \Delta \times S$. The coordinates transformation has some drawbacks, such as the fact that the coordinate values are initially expressed as floating-point values but must be rounded to fir the regular grid. The rounding leads to

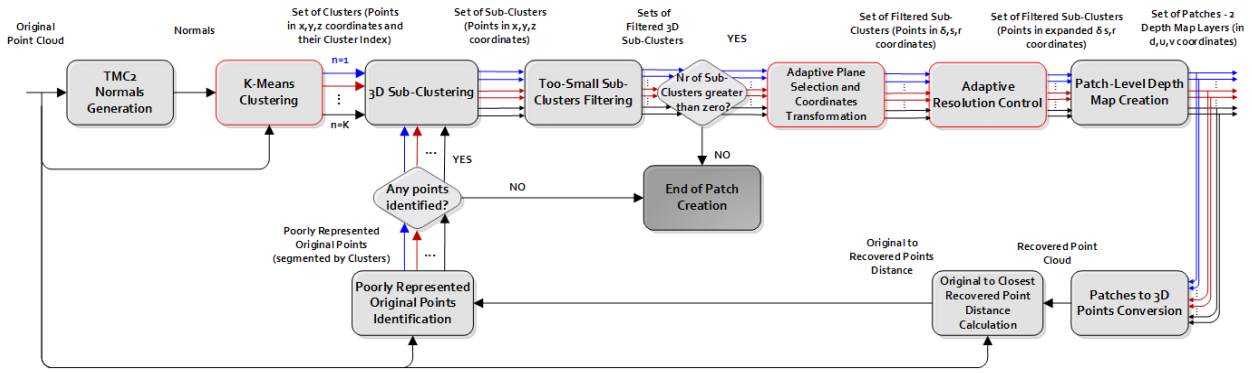

Figure 1 - Adaptive Plane Projection Patch Generation architecture, with the new modules highlighted in red.

some geometry distortion, which is mitigated by multiplying all sub-clusters 3D local coordinate values by an expansion factor.

Finally, the sub-clusters are converted into patches that represent their geometry. This is achieved by projecting the 3D local coordinates to depth maps composed of two layers (D0 and D1) to accommodate possible occluded points, where the difference between the same pixel on the two layers can only be up to 4. If multiple points fall into the same pixel, the one with lowest depth falls on the first layer, while the point with second lowest depth falls on the second layer, if it respects this difference; the other points are simply not projected. After all the segmented sub-clusters are converted to patches, all patches are converted back to 3D points, thus generating a reconstructed PC.

The distance between each original point and the closest one in this reconstructed PC is calculated. The original points for which this distance is greater than 1 are considered missed points, meaning that they are poorly represented by the selected patches. If after reconstructing the 3D points from the patches, no missed points exist, then the patch generation is over, and the list of patches goes to the packing module. Otherwise, the missed points are given to the 3D sub-clustering module so that new patches can represent them. These iterations continue until there are no more missed points or all the sub-clusters have been filtered.

4.2. Main Tools

The main tools are now detailed:

- ***K-Means Clustering*** - To avoid selecting clusters which are biased by the V-PCC six pre-defined clustering planes, the V-PCC clustering module is replaced by a K-Means based clustering process, which segments the points into a user-specified number of clusters, *K*, considering the normals previously estimated (as for V-PCC). Each resulting cluster is characterized by its number of points, and its normal centroid, which corresponds to the average of the normals associated to the clusters points. The K-means clustering algorithm was adopted from Rosetta Code [10], which minimizes the distance of each point to its cluster centroid. Considering our target, the clustering follows a different criterion, notably maximizes the internal product between each point normal and the respective cluster normal centroid to obtain normal adapted clusters.
- ***Adaptive Plane Selection and Coordinates Transformation*** – A projection plane is defined by the normal vector to it, which in this case is the depth axis (Δ). Thus, defining Δ axis is the same as defining the projection plane, and in AP2V-PCC, it is computed as the average of the sub-cluster set of normals. While the corresponding depth coordinate value (δ) will be the coded geometry value, s/S (tangent) and r/R (bi-tangent) coordinates/axes will be horizontal and vertical axes of the projected image, respectively. The calculation of the tangent axis (S) is inspired from the solution in [8], so that the sub-clusters have a similar orientation when they are projected into the projection plane compared to their orientation in the usual point cloud rendering. This similar orientation leads to an efficient packing of patches but also better compression with standard 2D video codecs (as the DCT transform basis functions are aligned with the patch direction). To achieve this, it is desirable that *S* and *R* are horizontally and vertically aligned with the PC typical orientation. Thus, the *S* axis is determined as $S = \Delta \times (0;-1;0)$, since the point clouds are typically oriented vertically in the Y-axis; naturally, *R* is calculated as $R = \Delta \times S$.
- ***Adaptive Resolution Control*** - The global to local coordinate system transformation has some consequences, since the obtained coordinate values are no longer grid aligned; notably, the local coordinate system values have to be initially expressed with floating-point precision and later rounded to fit a regular grid to be coded by a standard video codec. In turn, the rounding process in the local coordinate system will increase the geometry distortion by itself, as well as cause some possible point overlapping into the same 3D bin of the local coordinate system, which during the projection are only counted as a single point, thus increasing the geometry distortion even more. To address this problem, this module increases the 3D local coordinates resolution, reducing the geometry distortion caused by rounding as the 3D bins will be smaller. The resolution increase is a user-defined parameter, the so-called *expansion factor (EF)*, equal for all sub-clusters. In this context, each point local coordinate value is multiplied by this parameter after the respective local coordinate transformation.

4.3. Additional Tools

With the increase of resolution and the fact that the occupancy map precision is set to 4 (it decreases the bitrate drastically when compared to a precision of 1 or 2), the number of duplicate points increases significantly, which will penalize the performance of the recoloring module, as well as generate multiple decoded points in the same position when only one can be displayed. Thus, two additional tools were introduced outside the scope of the Patch Generation architecture, notably: Recoloring *pre- and post-processing*, extending the V-PCC recoloring module; and *duplicates removal*, which is included as a final step of the decoder module. Both tools are now described:

- ***Recoloring Pre and Post-Processing*** - At the encoder after reconstructing the point cloud geometry (R), which has floating-point precision, from the respective reconstructed geometry image and occupancy map, a filtered version (F) is created, where the duplicates are merged: F will have 1 point per filled 3D position. Next, F is recolored from the original point cloud and all points in the same position (duplicates) in R get the same color of that position in F. After, all the encoding of V-PCC is followed. The result is that at the decoder, all duplicates will have the same color.
- **Duplicates Removal –** At the decoder, the reconstruction of the point cloud is performed in the same way, but the geometry must be rounded, since the original content is represented with integers. This leads to the same effect where different points in the same position have different colors. Thus, for each decoded position the average color is calculated, and all points within the same position are merged into a single decoded point, being assigned the just-calculated average color.

5. PERFORMANCE ASSESSMENT

5.1. Test Conditions

The test material has been selected from the MPEG dataset [11], notably *Loot, Redandblack, Soldier and Longdress*.

The AP2V-PCC RD performance is compared to V-PCC and G-PCC, under the so-called MPEG Common Test Conditions (CTC) [12] with some additional quantization points to better cover the rate range.. The selected QPs are presented on Table 1, where R1 to R5 are already contemplated in the MPEG CTC [12].

Table 1 – QPs for RD performance assessment [12].

V-PCC			*AP2V-PCC*		
Test Point	Geom. QP	Col. QP	Test Point	Geom. QP	Col. QP
R1	*32*	*42*	R01	40	50
R2	*28*	*37*	R02	36	47
R3	*24*	*32*	*R1*	*32*	*42*
R4	*20*	*27*	*R2*	*28*	*37*
R5	*16*	*22*	*R3*	*24*	*32*
R6	13	18	*R4*	*20*	*27*
R7	10	14	*R5*	*16*	*22*

The quality metrics used for RD performance comparison were geometry D1 PSNR (P2Point) and geometry D2 PSNR (P2Plane), and color PSNR as defined in the MPEG CTC [12]. Following some optimization studies, the AP2V-PCC parameters have been set to K=12 and EF=1.6 to obtain the best geometry RD performance. G-PCC results were taken from [13].

5.2. RD Performance Evaluation and Informal Subjective Assessment

The geometry D1 PSNR and color Y PSNR RD performance for frame 1550 of *Redandblack* are depicted in Figure 2. The results for the other test PC show similar trends.

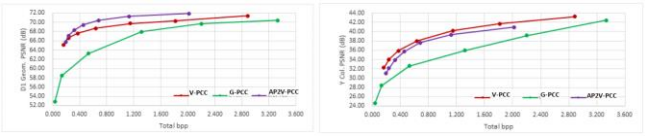

Figure 2 – Geometry D1 PSNR (left); Y Component PSNR (right) RD performance for frame 1550 of *Redandblack* sequence.

From Figure 2 and remaining results, it can be concluded that both AP2V-PCC and V-PCC outperform G-PCC, in terms of D1 geometry, and Y color. Thus, the BD-rate and BD-PSNR are only computed for AP2V-PCC, taking V-PCC as a reference on Table 2.

Table 2 – AP2V-PCC BD-rate and BD-PSNR regarding V-PCC for geometry and colour.

Sequence	Geometry				Colour	
	D1		D2		Y	
	BD-rate (%)	BD-PSNR (dB)	BD-rate (%)	BD-PSNR (dB)	BD-rate (%)	BD-PSNR (dB)
Redandblack	-42%	1.286	-17%	0.604	30%	-1.091
Soldier	-48%	1.549	-24%	0.870	21%	-0.846
Longdress	-55%	1.583	-27%	0.837	28%	-0.942
Loot	-32%	0.977	-2%	0.173	38%	-1.408
Average	-44%	1.349	-17%	0.621	29%	-1.072

From the Y RD charts and Bjontegaard tables, it is clear that the color quality does not benefit as much from the adaptive plane projections as the geometry. The geometry RD performance gains are achieved at the price of increasing the geometry images and occupancy map resolutions, and this effect is also propagated to the texture maps (since they all have the same resolution). Since rate has been invested on increasing the resolution, to achieve a comparable/similar total rate, the AP2V-PCC color QP must be larger than for V-PCC thus penalizing the color quality. Although this compensates for geometry, it does not compensate for color, in an RD sense, which naturally leads to a trade-off between the geometry and color qualities.

To informally assess the subjective quality, the *Loot* PC is used, since it shows the main trade-offs of AP2V-PCC versus V-PCC. Rendered PCs are presented in Figure 3 for AP2V-PCC and V-PCC when using a low and a high rate

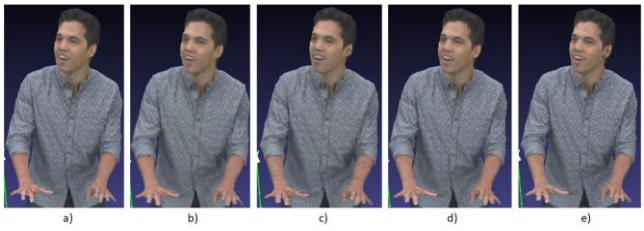

Figure 3 – Frame 1200 of Loot sequence [11]: a) Original; b) AP2V-PCC for 0.25 total bpp; c) V-PCC for 0.24 total bpp; d) AP2V-PCC for 1.47 total bpp; e) V-PCC for 2.00 total bpp.

For the lower rates, the color of AP2V-PCC coded PC is rather blurred, when compared to V-PCC, since the quality of the color is penalized when the same total rate is used. However, V-PCC has many more noticeable holes, since this solution has a lot more missed points. Moreover, for the larger rates, there is a V-PCC artefact that stands out, namely at the ear, which has a lot of holes caused by the high angular difference between the ear estimated normals to the fixed projection plane. Since AP2V-PCC uses adaptive planes, the angular difference is lower and the ear does not have noticeable holes.

6. FINAL REMARKS

This paper proposes a PC coding solution which extends the MPEG V-PCC solution by adapting the projection planes for the geometry and texture. This should allow obtaining better RD performance trade-offs between geometry and color. The trade-offs compensate for larger rates as proven by the informal subjective quality analysis. Future work should consider selecting AP2V-PCC parameters jointly optimized for geometry and texture, expansion factors optimized to each sub-cluster as well as extending the proposed solution to dynamic PCs.

7. REFERENCES

[1] MPEG 3DG and Requirements Subgroups, "Call for proposals for point cloud compression V2," Doc. ISO/IEC JTC1/SC29/WG11/N16763, Hobart, AU, Apr. 2017.

[2] V. Zakharchenko, "Algorithm description of mpeg-pcc-tmc2," ISO/IEC JTC1/SC29/WG11 MPEG2018/N17767, Ljubljana, Slovenia, Jul. 2018.

[3] K. Mammou, P. A. Chou, D. Flynn, and M. Krivokuća, "PCC Test Model Category 13 v3," ISO/IEC JTC1/SC29/WG11 N17762, Ljubljana, Slovenia, Jul. 2018.

[4] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," ICRA'2012, Minnesota, USA, May 2012.

[5] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," ICIP'2014, Paris, France, Oct. 2014.

[6] R. A. Cohen, D. Tian, and A. Vetro, "Attribute compression for sparse point clouds using graph transforms," ICIP'2016, Arizona, USA, Sep. 2016.

[7] 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. 27, no. 4, pp. 828–842, Apr. 2017.

[8] T. Golla and R. Klein, "Real-time Point Cloud Compression," IROS'2015, Hamburg, Germany, Oct. 2015.

[9] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The emerging JBIG2 standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 838–848, Nov. 1998.

[10] "Rosetta Code - K-means++ Clustering." [Online]. Available: http://rosettacode.org/wiki/K-means%2B%2B_clustering. [Accessed: 25-Sep-2018].

[11] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized full bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, Switzerland, Jan. 2017.

[12] S. Schwarz, G. Martin-Cocher, D. Flynn, and M. Budagavi, "Common test conditions for point cloud compression," ISO/IEC JTC1/SC29/WG11 N17766, Ljubljana, Slovenia, Jul. 2018.

[13] D. Flynn, "PCC TMC13v3 performance evaluation and anchor results," ISO/IEC JCTC1/SC29/WG11 W17768, Ljubljana, Slovenia, Jul. 2018.