Efficient Projected Frame Padding for Video-based Point Cloud Compression

Li Li, *Member, IEEE*, Zhu Li, *Senior Member, IEEE*, Shan Liu, Houqiang Li, *Senior Member, IEEE*

***Abstract*—The state-of-the-art 2D-based point cloud compression algorithm is the video-based point cloud compression (V-PCC) developed by the Moving Pictures Experts Group (MPEG). It first projects the point cloud patch by patch from 3D to 2D and organizes the projected patches into a video. The video will then go through High Efficiency Video Coding (HEVC) in order to be efficiently compressed. However, there are many unoccupied pixels that may have significant influences on the coding efficiency. These unoccupied pixels are currently padded using either the average of 4-neighbors or the push-pull algorithm. While these algorithms are simple, the unoccupied pixels are not handled in the most efficient way. In this paper, we divide the unoccupied pixels into two groups: those that should be occupied and those that should not be occupied. We then design padding algorithms tailored to each group in order to improve the performance. The first group is the unoccupied pixels that should be occupied according to the block-based occupancy map. We try to pad those pixels using the real points in the original point cloud to improve the quality of the reconstructed point cloud. Additionally, we try to maintain the smoothness of each block so as not to influence the video compression efficiency. The second group is the unoccupied pixels that were correctly identified as unoccupied according to the block-based occupancy map. These pixels are useless for improving the reconstructed quality of the point cloud. Therefore, we propose padding the residue of these pixels using the average residue of the occupied pixels in order to reduce the residue bitrate as much as possible. The proposed algorithms are implemented in the V-PCC and the corresponding HEVC reference software. The experimental results show the proposed algorithms can bring significant bitrate savings compared with the V-PCC.**

***Index Terms*—Frame padding, High Efficiency Video Coding, Occupancy map, Point cloud compression, Video-based point cloud compression**

I. Introduction

The point cloud is a set of 3D points that can be used to represent a 3D surface. Each point contains some specific attributes, such as colors, material reflection, and so on. The point cloud can be used to accurately reconstruct 3D objects or scenes in virtual reality [1]. Because of this, the point cloud is the technique choice for scene reconstruction used by navigation systems [2]. The combination of high resolution 2D images and 3D point clouds is promising for the auto-driving applications. The point cloud is superior than the 360-degree video reconstruction because it can support 6 degrees of freedom (DoF) rather than 3 DoF, which results in an enhanced user experience [3]. For a more thorough review of point cloud applications, refer to [4].

A typical point cloud captured by 8i using the camera plus depth sensors has around one million points per frame [5]. Larger numbers of points generally support higher resolution 3D object for reconstruction, however, this also leads to serious transmission and storage constraints. With each point represented by 30 and 24 bits for geometry and attribute information, respectively, a single frame with 1 million points can be as large as $6M$ bytes without compression. For a dynamic point cloud with 30 frames per second, the bitstream size without compression can be as high as $180M$ bytes per second. Therefore, there is an urgent need to compress dynamic point cloud data more efficiently.

Dynamic point clouds are compressed using one of two main methods: the 3D-based method and the 2D-based method. As its name implies, the 3D-based point cloud compression (PCC) compresses the point cloud directly in the 3D domain. Under this method, the geometry of the first frame is compressed using either the octree [6] or plane [7] [8] techniques when some local planes are present. The attribute is then compressed using a transform such as Graph Fourier Transform (GFT) [9], Region-based Adaptive Hierarchical Transform (RAHT) [10] or layered-based prediction [11]. After the first frame is encoded, subsequent frames are compressed using 3D motion estimation (ME) and motion compensation (MC) with the previously coded frames as reference [12] [13]. However, neighboring frames may have a different number of points without a one-to-one correspondence for ME and MC. Therefore, the 3D ME and MC will not be able to characterize the inter correlations efficiently thus the compression performance is unsatisfactory.

The 2D-based PCC method attempts to utilize the more mature video compression standard for 2D video by projecting the point cloud onto a 2D video. [14] and [15] did this successfully by projecting the point cloud onto a cube and a cylinder, respectively. The cube and cylinder were then unfolded to generate the 2D video. Once the 2D video is generated, it can be compressed using highly efficient methods, such as High Efficiency Video Coding (HEVC) [16] since it is continuous in both spatial and temporal domains. However, these projections may lead to a number of missing points that may degrade the reconstructed quality of the point cloud significantly. To address this, a patch-based projection [11]

L. Li and Z. Li are with the Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City, MO 64110, USA. L. Li is also with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230027, China. Professor Zhu Li is the corresponding author (e-mail: lil1@umkc.edu; lizhu@umkc.edu).

S. Liu is with the Tencent America, 661 Bryant St, Palo Alto, CA 94301 (e-mail: shanl@tencent.com).

H. Li is with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230027, China (e-mail: lihq@ustc.edu.cn).

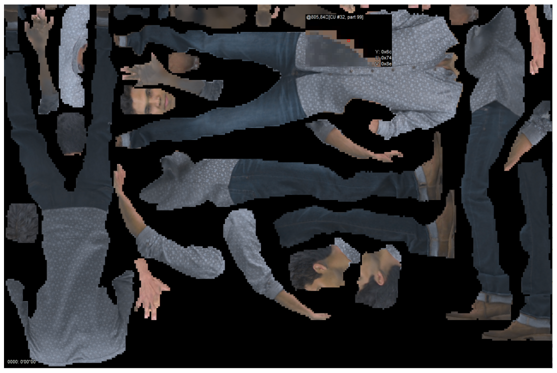


Fig. 1. Typical example of the projected attribute frame with unoccupied pixels padded as black of the DPC “Loot”. Its picture order count is 0.

that divides the point cloud into multiple patches according to the normals of points is proposed. This method is very successful and has won the Moving Pictures Experts Group (MPEG) call for proposals for dynamic PCC [17] in 2017. Consequently, it has been named video-based PCC (V-PCC) by the MPEG group. While this V-PCC method was being standardized, many works focused on improving the initial version were released [18]. However, the unoccupied pixels are still appear to be an issue. The reconstructed point cloud quality and the video compression bitrate need to be considered simultaneously to solve this.

Fig. 1 shows a typical example of the projected attribute frame from the point cloud. We can clearly see that black represents one type of unoccupied pixel. These unoccupied pixels will be signaled as such in the occupancy map and therefore will have no influence on the quality of the reconstructed point cloud at all. However, they may still cost a number of bits if not properly padded. These pixels will be called unoccupied-unoccupied pixels in the following sections.

The other type of unoccupied pixels, shown in the magnified area of Fig. 1, will be signaled as occupied. The occupancy map is currently signaled as 4×4 block. However, we cannot always guarantee the projected points will form an exact 4×4 block, and therefore, the unoccupied pixels typically appear in the patch boundary. If not properly padded, these unoccupied pixels will generate noisy points on the decoder side. They are called unoccupied-occupied pixels in the following sections.

The current methods in V-PCC handle these two cases using average of the 4-neighbor occupied pixels or the push-pull padding algorithm [19]. These methods are simple yet ineffective. For the unoccupied-unoccupied pixels, these methods will spend many bits on the residue, yet they are useless for reconstruction of the point cloud. For the unoccupied-occupied pixels, these methods will lead to noisy points on the decoder side, which will significantly reduce the quality of the reconstructed point cloud.

There are some works focusing on the padding for the general video [20] or the 360-degree video [21]. Li *et al.* [20] proposed a padding method for the frame boundary for arbitrary resolution video coding. When the picture does not contain an integer number of macroblocks, some pixels are padded in a manner that minimizes the bitrate. The pixels are padded to force the residue of unoccupied pixels to 0. However, this method only deals with padding regular structures near the picture boundary. Under the V-PCC framework, irregular structures within the picture need to be padded. Li *et al.* [21] proposed a padding method to maintain a continuous reference frame for the projected 360-degree video. This method is unsuitable for the V-PCC framework since it is designed for reference frame without considering the bitrate of the padded pixels.

In this paper, we propose a modified V-PCC framework with two advanced padding methods to deal with the unoccupied-occupied pixels and unoccupied-unoccupied pixels, respectively. The proposed algorithms mainly have the following key contributions.

- To deal with the unoccupied-occupied pixels, we propose padding them with the real points from the original point cloud as much as possible to avoid the noisy points. However, those points must be similar enough to the occupied pixels to avoid a bitrate significant increase. This scheme can reduce the noisy points while increasing some new points. This leads to improved performance.
- To deal with the unoccupied-unoccupied pixels, we propose a content-adaptive method to adaptively pad them according to the residue of the occupied pixels in order to reduce the bits spent as much as possible. To be more specific, we pad the residues of unoccupied-unoccupied pixels using the average value of the residues of the occupied pixels in the block. This residue block with less variation can be coded with a small number of bits without diminishing the reconstructed quality of the occupied pixels.
- We implement the proposed algorithms in the V-PCC and the corresponding HEVC reference software. The proposed two algorithms can individually lead to significant coding gains. They can also be combined into a unified padding framework to achieve an even better coding performance.

The rest of this paper is organized as follows. We will review the related works on the PCC in Section II. The proposed algorithms including the unoccupied-occupied padding and unoccupied-unoccupied padding will be described in Section III. In Section IV, the detailed experimental results will be shown. Section V will conclude the paper.

II. Related Work

The current works on PCC can be organized into two groups according to different types of point cloud: static PCC and dynamic PCC. Currently, it appears there is more focus on static PCC rather than dynamic PCC. In this paper, we will focus more on the dynamic PCC. We introduce dynamic PCC methods in more detail compared with static PCC.

A. Static point cloud compression

Botsch *et al.* [22] first proposed compressing the geometry using octree. The point cloud is first bounded by a cube and then the cube is iteratively divided into 8 sub-cells. The occupancy of the sub-cells is signaled using 1 byte to indicate whether the sub-cells have points or not. All the bytes are then compressed using arithmetic coding. This method is simple yet effective and is adopted as the base for the MPEG sparse point cloud compression framework [11]. In addition, Peng and Kuo [23] proposed encoding the octree using the number of nonempty cells and the nonempty-child-cell tuples. Schnabel and Klein [6] further proposed using an approximated plane to predict these two values. Huang *et al.* [24] introduced an occupancy code reordering method to increase the occupancy compression efficiency. Instead of encoding the occupancy directly, Kämpe *et al.* [25] proposed using the directed acyclic graph (DAG) to reduce the bitrate. Most recently, Rente *et al.* [26] proposed a scalable compression scheme with the octree as the base layer and the graph-based transform approach as the enhancement layer.

In addition to the octree-based method, Park and Lee [7] introduced multi-scale planes to approximate the sub-cells. Smith *et al.* [27] proposed plane pruning and merging methods to achieve satisfiable performance under different target bitrates. Fan *et al.* [28] introduced Generalized Lloyd Algorithm (GLA) to generate the level of details to encode the geometry. Schnabel *et al.* [29] proposed using plane and height-maps to compress the point cloud. Golla and Klein *et al.* [30] further introduced an occupancy map to enhance the compression of plane and height-maps. Ahn *et al.* [31] proposed using the range image to compress the large-scale 3D point cloud. During the PCC standardization process, Chou *et al.* [32] proposed using the plane to approximate the octree to encode the dense point cloud. Kathariya *et al.* [8] introduced quadtree combined with kd-tree to divide the space. In addition to lossy compression, Zhu *et al.* [33] introduced Traveling Salesmen Problem (TSP) to scan the points and compress the geometry losslessly.

The first group of works focusing on attribute compression is the transform-based method. Zhang *et al.* [9] proposed using GFT to exploit the correlations in the geometry to compress the attribute. However, deriving the transform kernel requires solving a very complex eigen problem. Therefore, Queiroz and Chou [10] introduced RAHT to compress the attribute to get a better balance between the performance and the complexity. In essence, RAHT is a weighted wavelet transform related to cell occupancy. They [34] further provided a Gaussian-Process-Model-based transform to encode the attribute more efficiently.

The second group of works on attribute compression is the geometry-structure-based method. In [24], the attribute of the parent node is used to predict that of the child node and the residue is encoded using an adaptive quantization scheme. Dado *et al.* [35] proposed using palette coding to compress the attribute derived from the DAG. Morell *et al.* [36] introduced k-means to perform color segmentation for each plane generated from geometry and then the centroid is entropy coded for compression. During the PCC standardization process, Mammou *et al.* [11] introduced a layer-based prediction scheme to encode the attribute from coarse to fine granularities. A lifting scheme is further provided in [37] to improve the performance. There are also some works focusing on deriving a layer-based structure using the kd-tree [38] or the neighboring information [39].

B. Dynamic point cloud compression

Efficiently performing ME and MC is key to exploiting the correlations between neighboring frames in a dynamic point cloud. The dynamic PCC methods can be divided into two groups: 3D-based method and 2D-based method. As its name implies, the 3D-based method tries to perform the ME and MC in the 3D domain. Kammerl *et al.* [40] proposed encoding the differences in geometry for exploiting the inter correlations from frame to frame. However, this method is limited since there is not always a one-to-one point correspondence between the neighboring frames. Thanou *et al.* [12] proposed that the correlations between neighboring frames can be better characterized in the spectral domain rather than in the pixel domain and performed 3D ME by feature matching. However, some points may be unable to find the corresponding points using this global method. To solve this problem, Queiroz and Chou [41] proposed dividing the point cloud into multiple cubes and performed ME and MC cube by cube. However, this 3D translation motion model cannot characterize motion efficiently. Mekuria *et al.* [13] proposed using an iterative closest point (ICP) method to better characterize the motions. However, this method is also insufficient at characterizing motion. While a number of methods have been proposed to improve the compression efficiency of dynamic PCC, they can only partially alleviate the problems of 3D ME and MC. Due to inflexible partitions and inaccurate motion vector predictors, efficiently performing ME and MC is still an issue with dynamic 3D point clouds.

2D video compression efficiency is known to be very high. It effectively makes full use of the inter correlations using the flexible partitions and accurate motion vector prediction. Therefore, there are some works trying to project the point cloud to the 2D domain to utilize the video compression standard. Lasserre *et al.* [42] and Budagavi *et al.* [43] proposed sorting the points directly into a video using an octree or point position in a lossless manner. However, the videos generated are unsuitable for the current video compression framework since the spatial and temporal correlations are limited. To solve this problem, Schwarz *et al.* [14] and He *et al.* [15] proposed projecting the point cloud to video using cubes or cylinders and unfolding it into a 2D video. The generated videos are easy to encode, however, many points are lost due to occlusion. Mammou *et al.* [11] proposed a patch-based method to project the point cloud to the cube patch by patch and organized all the patches into a video. As we have mentioned in Section I, this is a successful method and won the MPEG call for proposals for the dynamic PCC. Nonetheless, the projected video introduces many unoccupied pixels among the patches and therefore, is still unsuitable for current video compression standards. In this paper, we propose padding the unoccupied pixels using methods that result in a significant performance improvement.

III. Proposed algorithms

In this section, we will introduce the proposed padding methods. The padding of unoccupied-occupied pixels will be introduced in subsection III-A. The padding of the unoccupied-unoccupied pixels will be introduced in subsection III-B.

TABLE I
PERFORMANCE OF THE PIXEL-BASED OCCUPANCY COMPARED WITH THE 4×4-BLOCK-BASED OCCUPANCY.

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	95.1%	78.9%	–1.4%	–2.4%	–4.4%
RedAndBlack	98.5%	74.0%	–1.9%	–4.2%	–2.5%
Soldier	115.0%	97.7%	–4.9%	–11.6%	–8.8%
Queen	129.6%	103.7%	–9.5%	–12.6%	–9.8%
LongDress	90.0%	72.1%	–2.0%	–2.7%	–1.5%
Avg.	**105.6%**	**85.3%**	**–3.9%**	**–6.7%**	**–5.4%**

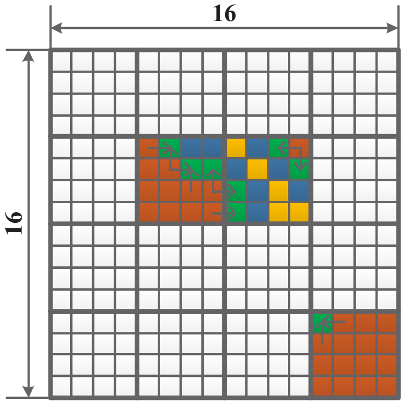


Fig. 2. Original padding process. The colored three 4×4 blocks are signaled as occupied. The orange small squares represent the occupied pixels.

A. Padding of unoccupied-occupied pixels

As mentioned in Section I, the block-based occupancy map is the reason why the unoccupied-occupied pixels exist. Therefore, before introducing the proposed padding of unoccupied-occupied pixels, we first compare the performance of the pixel-based occupancy with the 4×4-block-based occupancy as shown in Table I. From Table I, we can see that the pixel-based occupancy can bring an average of 3.9%, 6.7%, and 5.4% performance improvements compared with the 4×4-block-based occupancy for the Luma, Cb, and Cr components. However, for geometry, it leads to 105.6% and 85.3% bitrate increase for D1 and D2, respectively, due to the increase in size of the occupancy. According to our observation, the bits of the pixel-level occupancy show a 4 times increase above the 4×4-block-level occupancy. For this reason, unoccupied-occupied pixels exist under the V-PCC framework.

Fig. 2 shows the padding process of the unoccupied-occupied pixels of a typical 16×16 block in the original V-PCC framework. Each small square represents a pixel. The colored three 4×4 blocks are signaled as occupied. The orange small squares are the occupied pixels. The unoccupied-occupied pixels are iteratively padded using the 4-neighbor occupied pixels. The green pixels are first padded using the average of the 4-neighbor orange pixels as indicated by the arrows. The blue pixels are then padded using the average of the 4-neighbor green pixels. Finally, the yellow pixels are padded using the blue pixels. This method creates a smooth block that can effectively reduce the bitrate. However, the padded pixels create noisy points in the reconstructed point cloud, which show as an increase in distortions, particularly on point-to-plane errors. Graziosi and Tabatabai [44] proposed using the uncompressed full resolution occupancy map to guide the padding process. Ke *et al.* [45] introduced patch expansion in the boundary to avoid data loss. However, these methods are temporally costly and lead to images that increase the bitrate significantly.

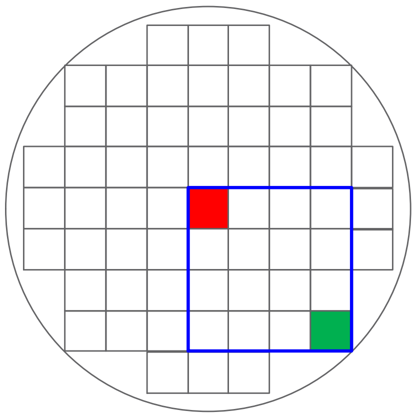


Fig. 3. Search range illustration of the proposed unoccupied-occupied padding.

In this paper, we propose padding the unoccupied-occupied pixels with the real points in the original point cloud to reduce the distortion. The real points can come from the missing points in the point cloud or the duplicate points from the other patches. However, this algorithm may lead to serious bitrate increases if they are not properly handled. Therefore, we aim to find a better balance between the increased bitrate and the reconstructed quality of the point cloud.

Before padding of the unoccupied-occupied pixels, we should first locate the to-be-padded 4×4 blocks. These 4×4 blocks should include both occupied and unoccupied pixels. While locating those 4×4 blocks, we simultaneously record the 3D coordinate of the first occupied pixel $(x3_q, y3_q, z3_q)$ in the block. This pixel will be used as the center point when searching for padding candidates in 3D space. It can also be used to anchor the smoothness of the current block, and to prevent the encoder from spending extra bits on the current 4×4 block.

After the to-be-padded 4×4 block is located, we then try to find the candidate pixels to be padded. We use the first occupied pixel with coordinate $(x3_q, y3_q, z3_q)$ as the starting point. We then find the nearest candidate points using kd-tree [46]. In the kd-tree, every leaf is a k-dimensional point. Every non-leaf node can be thought as a hyperplane that splits the space into two parts.

Consider the extreme case where the number of nearest points K to be searched as shown in Fig. 3. Each small square represents a pixel. The large blue outline square represents the 4×4 block to be padded. The red square is the template point. The green square is the farthest to-be-padded point compared with the template point. According to the circle in Fig. 3, we need to search 61 points in 2D space. Since a point cloud is essentially surface of an object, each 2D point will not correspond to more than 2 points in 3D space in most cases.

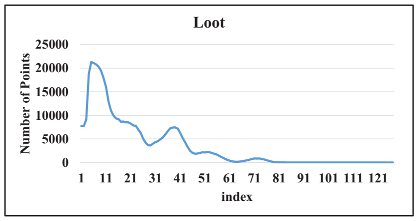


Fig. 4. Histogram of choosing different nearest points of the dynamic point cloud “Loot”.

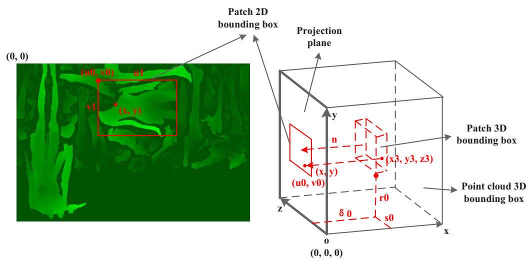


Fig. 5. Illustration of the projection from 3D to 2D coordinates.

So, we need to search 122 points in 3D space. To avoid some extreme cases that each 2D point may correspond to more than 2 points, we choose to search the nearest 128 points to maximize the performance. To better illustrate the reason of choosing the nearest 128 points, we show the histogram of the index of the various nearest points selected of the point cloud “Loot” as shown in Fig. 4. We can see that indices larger than 80 will rarely be selected, thus making 128 as a good threshold.

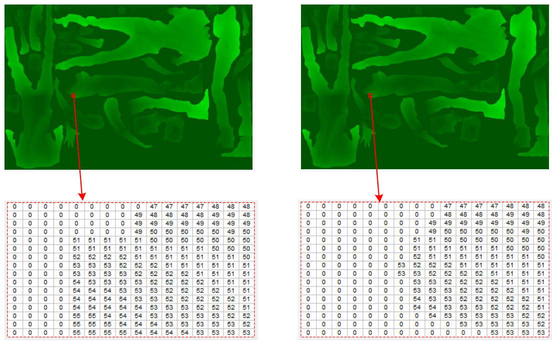
We must satisfy two objectives. First, the padded points must be real points from the original point cloud. Second, the bitrate of the current block cannot significantly increase. To do this, we must satisfy the following two constraints. First, the tangential shift and the bi-tangential shift of the candidate points should be the same as the to-be-padded position $(y3, z3)$. This guarantees that the padded points are real points as shown in Fig. 5. We assume that the current patch is projected to the yoz plane. Given a to-be-padded position with coordinate (x, y), the start position of the current patch $(u0, v0)$, and the start tangential and bi-tangential shifts of the current patch $(s0, r0)$, the tangential and bi-tangential shifts of the candidate points $(y3, z3)$ can be calculated as

$$\begin{cases} y3 = s0 + (x - u0) \\ z3 = r0 + (y - v0) \end{cases} \tag{1}$$

After deriving the tangential shift and the bi-tangential shift of the to-be-padded position, we cycle through the 128 candidates to find those satisfying the constraint.

Second, in order to maintain the smoothness of the current block, the difference between the to-be-padded candidates and the occupied pixels should be small. Therefore, we only pad the candidate pixel if the absolute difference between the depth of the to-be-padded pixel $x3$ and the query point $x3_q$ is smaller than a given threshold θ,

$$|x3 - x3_q| < \theta. \tag{2}$$



0	0	0	0	0	0	0	0	0	47	47	47	47	48	48	48
0	0	0	0	0	0	0	0	49	48	48	48	48	49	49	48
0	0	0	0	0	0	0	0	49	49	49	49	49	49	49	49
0	0	0	0	0	0	0	0	49	50	50	50	50	50	49	50
0	0	0	0	51	51	51	51	51	50	50	50	50	50	50	50
0	0	0	0	51	51	51	51	51	51	51	51	51	50	50	50
0	0	0	0	52	52	52	52	51	51	51	51	51	51	51	50
0	0	0	0	53	53	53	52	52	52	51	51	51	51	51	51
0	0	0	0	53	53	53	53	52	52	52	52	51	51	51	51
0	0	0	0	54	53	53	53	53	52	52	52	52	51	51	51
0	0	0	0	54	54	54	53	53	53	52	52	52	52	51	51
0	0	0	0	54	54	54	54	53	53	52	52	52	52	52	51
0	0	0	0	54	54	54	54	54	53	53	53	52	52	52	51
0	0	0	0	55	55	54	54	54	53	53	53	53	53	52	52
0	0	0	0	55	55	55	54	54	54	53	53	53	53	53	52
0	0	0	0	55	55	55	55	54	54	54	54	53	53	53	53

0	0	0	0	0	0	0	0	0	47	47	47	47	48	48	48
0	0	0	0	0	0	0	0	0	48	48	48	48	49	49	48
0	0	0	0	0	0	0	0	49	49	49	49	49	49	49	49
0	0	0	0	0	0	0	0	49	50	50	50	50	50	49	50
0	0	0	0	0	0	0	51	51	50	50	50	50	50	50	50
0	0	0	0	0	0	0	51	51	51	51	51	51	50	50	50
0	0	0	0	0	0	0	52	51	51	51	51	51	51	51	50
0	0	0	0	0	0	53	52	52	52	51	51	51	51	51	51
0	0	0	0	0	0	53	53	52	52	52	52	51	51	51	51
0	0	0	0	0	0	0	53	53	52	52	52	52	51	51	51
0	0	0	0	0	0	0	53	53	53	52	52	52	52	51	51
0	0	0	0	0	0	0	54	53	53	52	52	52	52	52	51
0	0	0	0	0	0	0	54	54	53	53	53	52	52	52	51
0	0	0	0	0	0	0	0	54	53	53	53	53	53	52	52
0	0	0	0	0	0	0	0	0	0	53	53	53	53	53	52
0	0	0	0	0	0	0	0	0	0	0	0	53	53	53	53

Fig. 6. Typical comparison between the padded frame (left) and the to-be-padded frame (right).

Larger θ values will yield more points to be padded as unoccupied-occupied pixels. This leads to higher quality point cloud reconstruction. However, the smoothness of the current block will become less and the bitrate will increase. We set θ to 2.0 as the baseline value, then analyze the performance of the proposed algorithm for different values of θs in the experimental results.

There may be multiple points satisfying the above two constraints. We choose one based on the projection mode of the current patch. In the current patch-based method, each point cloud will be projected to 2 frames, d0 and d1. The projection mode determines whether the current patch projects the minimum value or the maximum value to d0. If the minimum value is projected onto d0, we cycle through the candidate points to find which has the minimum $x3$. If the maximum value is projected onto d0, we first find an available candidate point satisfying the two constraints and then cycle through the candidates to find which has the maximum $x3$. After finding the most appropriate point $(x3, y3, z3)$, we then project it onto (x, y) using the following formula,

$$\begin{cases} x = z3 - s0 + u0 \\ y = y3 - r0 + v0 \\ h(x, y) = x3 - \delta 0 \end{cases} \tag{3}$$

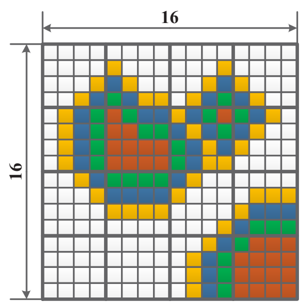
where $h(x, y)$ is the geometry value padded.

The above process pads part of the unoccupied-occupied pixels. However, some unpadded pixels still remain. We then pad those using the average value of the 4-neighbor pixels. Fig. 6 gives a comparison between the padded frame (left) and the to-be-padded frame (right). As can be seen from the figure, despite being padded with real points, the block is quite smooth. Furthermore, the padding may come from either missing points or padded points from other patches. This will be verified in the experimental results. Either way, the padded pixels improve the reconstructed quality of the point cloud. After using the above steps to generate a video from the original point cloud, the resulting video will be encoded to finish the compression process.

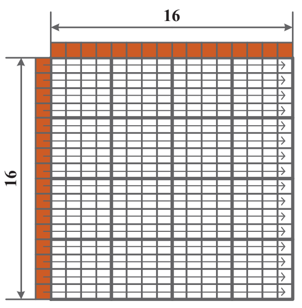
We have also attempted padding d1 with a different value from d0 using a similar process. However, due to the bitrate increase, the experimental results suggest a degradation in the overall PCC performance. Therefore, we pad d1 with the same value as d0 for the unoccupied-occupied pixels. To clearly

Algorithm 1 Unoccupied-occupied padding algorithm

```
1:  Input: Center point $(x3_q, y3_q, z3_q)$, To-be-padded position $(x, y)$, Projection mode $PM$
2:  Output: Padded value $h(x, y)$
3:  Find the nearest $K$ points of $(x3_q, y3_q, z3_q)$
4:  Calculate the tangential and bi-tangential shifts of the candidate points $(y3, z3)$ according to (1)
5:  $x3 = infinitDepth$ ▷ $x3$ initialized as the infinite depth
6:  for $i \leftarrow 1$ to $K$ do
7:      if $PM = 0$ and $|x3_i - x3_q| < \theta$ and $x3_i < x3$ then ▷ $x3_i$ is the $ith$ candidate
8:          $x3 = x3_i$
9:      else if $PM = 1$ then
10:         if $x3 = infinitDepth$ and $|x3_i - x3_q| < \theta$ then
11:             $x3 = x3_i$
12:         else if $|x3_i - x3_q| < \theta$ and $x3_i > x3$ then
13:             $x3 = x3_i$
14:         end if
15:     end if
16: end for
17: Calculate $h(x, y)$ according to (3)
```



(a) Part unoccupied



(b) Full unoccupied

Fig. 7. Original padding of the unoccupied-unoccupied pixels for the geometry.

delineate the proposed unoccupied-occupied algorithm above, we list the process steps in Algorithm 1.

B. Padding for unoccupied-unoccupied pixels

The padding for unoccupied-unoccupied pixels in the V-PCC framework can be divided into two groups: geometry padding and attribute padding. The geometry padding is based on 16×16 block as shown in Fig. 7. If part of the 16×16 block is occupied as shown in Fig. 7 (a), the 16×16 block will be padded iteratively using the average of the 4-neighbor occupied pixels. If the whole 16×16 block is unoccupied as shown in Fig. 7 (b), the 16×16 block will be padded from the neighboring pixels using horizontal copy or vertical copy if the horizontal neighboring pixels are not available. The attribute padding is based on the push-pull algorithm as shown in Fig. 8 [47]. The push-pull algorithm [19] is an efficient pyramid algorithm for the interpolation of scattered data. In addition, a sparse linear model based method [48], a harmonic background filling method [49], and a smoothed push-pull algorithm [50] are proposed to improve the padding

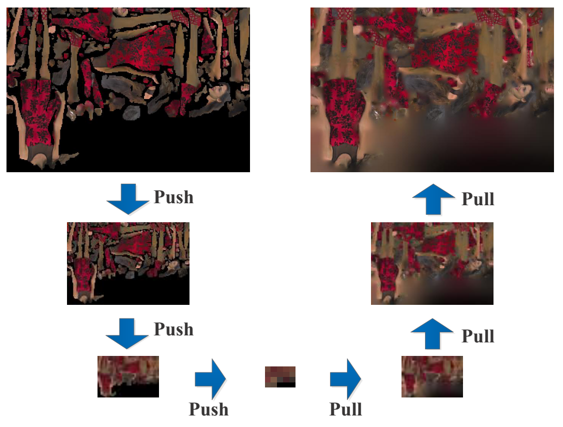


Fig. 8. Original padding of the unoccupied-unoccupied pixels for the attribute.

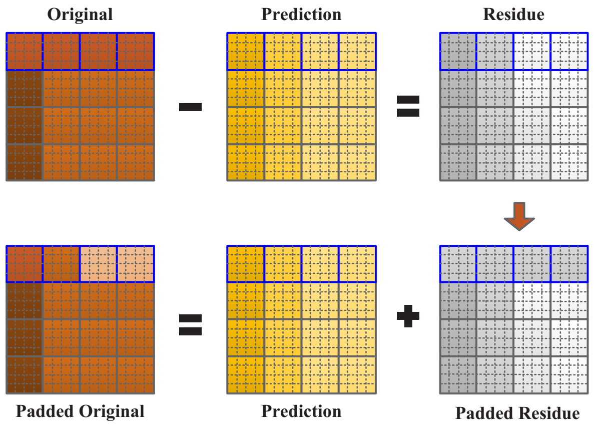


Fig. 9. Basic idea of the unoccupied-unoccupied padding.

of the attribute. However, all these methods can only improve the padding of the attribute by about 3% compared with the original push-pull algorithm. After the geometry and attribute padding, the unoccupied-unoccupied pixels of d0 and d1 are padded using the average [51] in order to code d1 with fewer bits.

The original padding methods for geometry and attribute are quite simple and have guaranteed the smoothness of the block if it is independently used as a prediction unit (PU). However, if the unoccupied and occupied pixels are combined into one PU, the unoccupied pixels may prevent the occupied pixels from finding the corresponding block. Even if the occupied pixels can find the corresponding block, the unoccupied pixels may not be adequately predicted, leading to a significant bitrate increase. In this paper, we try to adaptively pad the unoccupied pixels resulting in the smallest bitrate.

Fig. 9 shows the basic process of the proposed adaptive padding method for the unoccupied-unoccupied pixels using 16×16 block. Each small block represents a 4×4 block. The small blocks with blue edges in the original PU represent the unoccupied 4×4 blocks. First, we can obtain a prediction block from intra or inter prediction. In intra prediction, the prediction block is obtained based on the intra prediction direction. In inter prediction, the prediction block is obtained using motion compensation based on the motion information.

Then we can obtain the residue block through subtracting the prediction block from the original block. After that, the unoccupied pixels of the residue block are padded to minimize the bitrate of the current PU. Finally, the unoccupied pixels of the original PU are updated according to the prediction and the padded residue. Using this process, the padding result will vary as the prediction varies. Therefore, the proposed unoccupied-unoccupied padding method is considered a content-adaptive method.

The key step of the proposed unoccupied-unoccupied padding process is correctly padding the residue of the unoccupied pixels in order to minimize the bits cost of the current PU,

$$\min_{Res_{uo}} R(Res_o, Res_{uo}) \tag{4}$$

where Res_o and Res_{uo} are the sets of the occupied and unoccupied pixels of the residue block, respectively. $R(Res_o, Res_{uo})$ is the bitrate of the residue block related to both the occupied and unoccupied pixels. Since multiple pixels in the current block need to be padded, iterating through all possible combination of padding pixels to find the optimized solution may increase the encoder complexity significantly. In this work, we propose intuitive methods to reduce the bitrate.

The first method is to pad the residues of unoccupied pixels with zeros. This method is similar to the picture boundary padding in [20]. However, the zeros padding combined with the residue of the occupied pixels may increase the variance of the transformed residue block. This makes the transformed residue block difficult to compress. Additionally, the experiments show this method leads to significant performance losses for the Chroma components.

The second method we propose is to pad the residues of the unoccupied pixels with the average residue of the occupied pixels. This method has shown to maintain or increase the smoothness of all of the residue blocks, and is therefore more suitable for transform. Furthermore, this method can improve the performance significantly for both the Luma and Chroma components. In the special case where all the pixels of the current PU are unoccupied, the residues of all the pixels will be padded as 0.

The proposed method is implemented in the mode decision processes of the HEVC reference software for both the intra and inter predictions. The proposed method is implemented in a different way to deal with various kinds of distortions. When the distortion is measured by the sum of the absolute difference (SAD) between the original block and the prediction block, it is determined by the distortion of the occupied pixels, and the padding method becomes irrelevant. If we pad the unoccupied pixels of the residue block with 0, the SAD will only be determined by the occupied pixels. If we pad the unoccupied pixels of the residue block with the average value of the occupied residues, the SAD of the whole block is equal to the SAD of the occupied pixels divides the ratio of occupied pixels to total pixels. Since this ratio is a fixed value, the SAD of the block essentially is still determined by the occupied pixels. To prevent frequently changing the residue and original values of the current block, we add a mask to the current block to calculate the SAD of the occupied pixels only,

$$SAD = \sum_{i=1}^{N} |Org_i - Pred_i| \times M_i \tag{5}$$

where N is the number of pixels in the current block. Org_i and $Pred_i$ are the original and prediction values of the pixel i in the current block. M_i is the mask determined by the occupancy map. M_i is 1 when pixel i is occupied and is 0 when pixel i is unoccupied.

When the distortion of the current block is measured using either the sum of the absolute transformed difference (SATD) between the original block and the prediction block or the sum of the squared difference (SSD) between the original block and the reconstruction block, we follow the residue and original padding processes as shown in Fig. 9. Since these methods rely on either a Hadamard transform or discrete cosine transform (DCT), estimating the impact padding has on the distortion after transform becomes difficult. However, since we only have limited search points for calculating the SATD or SSD, the frequently changing of the residue and original values of the current block does not increase the complexity obviously.

In our previous work [52], we proposed an occupancy-map-based rate distortion optimization (RDO) to ignore the distortion of unoccupied pixels. In this work, we propose an adaptive padding scheme to reduce the bitrate of blocks containing the unoccupied pixels. Both schemes provide significant bitrate savings from the V-PCC anchor. Additionally, the unoccupied-unoccupied padding algorithm can have similar effects on the distortions of the unoccupied-unoccupied pixels as the occupancy-map-based RDO. However, combining these two algorithms together leads to further improvement of the rate distortion (RD) performance.

TABLE II
CHARACTERISTICS OF THE TEST DYNAMIC POINT CLOUD

Test point cloud	Frame rate	Number of points	Geometry precision	Attributes
Loot	30	$\sim$ 780000	10 bit	RGB
RedAndBlack	30	$\sim$ 700000	10 bit	RGB
Soldier	30	$\sim$ 1500000	10 bit	RGB
Queen	50	$\sim$ 1000000	10 bit	RGB
LongDress	30	$\sim$ 800000	10 bit	RGB

IV. EXPERIMENTAL RESULTS

A. Simulation setup

The proposed algorithms are implemented in the state-of-the-art video-based point cloud compression (V-PCC) reference software TMC2-4.0 [53] and the corresponding HEVC reference software HM16.18-SCM8.7 [54] to compare with the V-PCC anchor to demonstrate its effectiveness. We test both the lossy geometry, lossy attributes, random access (RA) and all intra (AI) cases to demonstrate the effectiveness of the proposed algorithms. We perform the experiments on the five dynamic point clouds defined in the V-PCC common test condition (CTC) [55]. We test all frames of the dynamic point clouds for the combination of the proposed padding algorithms. When measuring the performance of each algorithm

TABLE III
PERFORMANCE OF THE COMBINATION OF THE PROPOSED PADDING ALGORITHMS IN RA CASE

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–18.4%	–17.5%	–20.6%	–17.7%	–21.0%
RedAndBlack	–10.6%	–9.7%	–11.9%	–14.9%	–11.6%
Soldier	–16.2%	–15.9%	–11.1%	–5.7%	–5.3%
Queen	–21.3%	–21.5%	–15.7%	–17.8%	–15.7%
LongDress	–11.6%	–10.9%	–4.4%	–4.7%	–4.1%
Avg.	**–15.6%**	**–15.1%**	**–12.7%**	**–12.2%**	**–11.5%**
Enc. self	105%				
Dec. self	99%				
Enc. child	95%				
Dec. child	95%				

TABLE V
PERFORMANCE OF THE COMBINATION OF THE PROPOSED PADDING ALGORITHMS IN RA CASE TESTED WITH 32 FRAMES

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–24.2%	–23.5%	–25.7%	–19.0%	–27.1%
RedAndBlack	–10.9%	–9.6%	–12.0%	–14.8%	–11.9%
Soldier	–19.6%	–19.4%	–15.2%	–10.1%	–7.7%
Queen	–18.5%	–18.7%	–15.6%	–15.3%	–11.8%
LongDress	–11.4%	–10.7%	–5.5%	–6.3%	–5.7%
Avg.	**–16.9%**	**–16.4%**	**–14.8%**	**–13.1%**	**–12.8%**
Enc. self	106%				
Dec. self	99%				
Enc. child	95%				
Dec. child	95%				

TABLE IV
PERFORMANCE OF THE COMBINATION OF THE PROPOSED PADDING ALGORITHMS IN AI CASE

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–9.3%	–8.0%	–9.1%	–1.6%	–5.9%
RedAndBlack	–7.9%	–7.2%	–8.0%	–6.8%	–6.7%
Soldier	–5.1%	–4.5%	–1.8%	6.1%	6.3%
Queen	–8.9%	–9.1%	–8.7%	–9.4%	–8.5%
LongDress	–7.7%	–7.2%	–3.2%	–2.2%	–1.9%
Avg.	**–7.8%**	**–7.2%**	**–6.2%**	**–2.8%**	**–3.4%**
Enc. self	108%				
Dec. self	99%				
Enc. child	100%				
Dec. child	100%				

TABLE VI
PERFORMANCE OF THE COMBINATION OF THE PROPOSED PADDING ALGORITHMS IN AI CASE TESTED WITH 32 FRAMES

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–9.9%	–8.3%	–8.7%	0.0%	–5.3%
RedAndBlack	–8.3%	–7.2%	–7.5%	–6.5%	–6.7%
Soldier	–6.8%	–6.0%	–2.8%	6.1%	5.9%
Queen	–7.9%	–8.3%	–8.8%	–9.4%	–7.8%
LongDress	–7.7%	–7.3%	–3.4%	–2.2%	–1.9%
Avg.	**–8.1%**	**–7.4%**	**–6.2%**	**–2.4%**	**–3.1%**
Enc. self	108%				
Dec. self	99%				
Enc. child	101%				
Dec. child	100%				

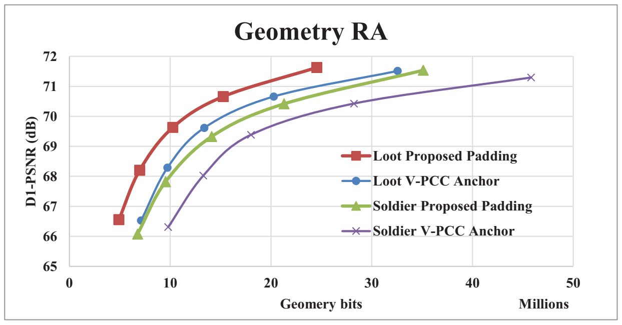
individually, we test on a 32 frame subset of the point cloud. The detailed characteristics of the dynamic point clouds are shown in Table II. We test all five rate points from low bitrate (r1) to high bitrate (r5) [55]. Since the bitrates generated by the anchor and the proposed algorithms are not the same, the Bjontegaard-Delta-rate (BD-rate) [56] is used to compare the respective RD performances.

For the geometry, we report the BD-rates for both point-to-point PSNR (D1) and point-to-plane PSNR (D2) [55]. For the attribute, the BD-rates for the Luma, Cb, and Cr components are reported. For the complexity, we made changes to both the PCC and HEVC reference softwares, and therefore report the encoding and decoding time change for both the PCC (self) and HEVC (child), separately. In the following subsections, we first introduce the overall performance of the combination of the proposed padding algorithms. Then we report the performance and analysis of the proposed algorithms individually.

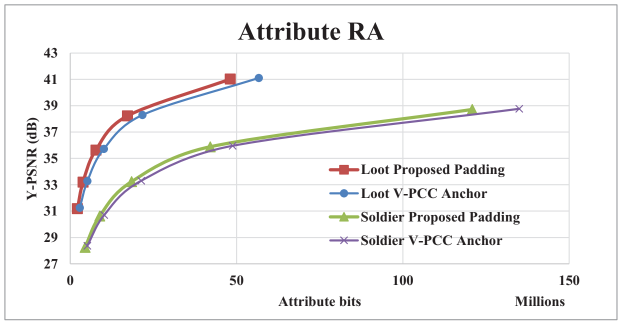
B. Overall performance

Table III and Table IV show the performances of the combination of the proposed algorithms in RA and AI cases, respectively. The value of θ is set as 2.0 to obtain a good trade-off between the reconstructed point cloud quality and the bitrate. From Table III, we can see that the combination of the proposed algorithms can achieve an average of 15.6% and 15.1% bitrate savings compared with the V-PCC anchor for the D1 and D2 in RA case, respectively. The combination of the proposed algorithms can bring performance improvements of 12.7%, 12.2%, and 11.5% for Luma, Cb, and Cr components, respectively. From Table IV, we can see that the combination of the proposed algorithms can lead to an average of 7.8% and 7.2% bitrate savings compared with the V-PCC anchor for the D1 and D2 in AI case, respectively. The attributes are improved on average by 6.2%, 2.8%, and 3.4% for Luma, Cb, and Cr components. We also show some examples of the RD curves as in Fig. 10 to better illustrate the performance in both RA and AI cases. Obvious performance improvements are also observed in the RD curves for both the geometry and attribute. From these experimental results, the proposed algorithms bring a clear performance improvement. We also show that performance improvements when testing 32 frames in Table V and Table VI are consistent with the performance of the whole point cloud.

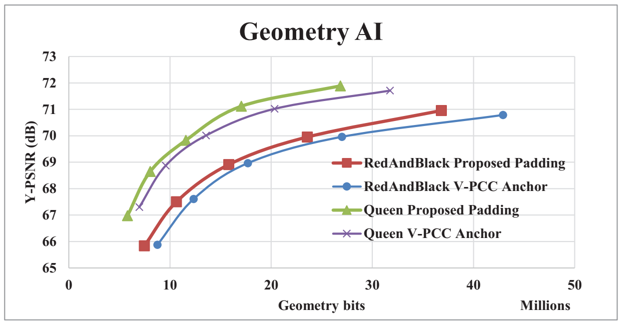
The encoding and decoding complexity of the proposed algorithms are shown in Table III and Table IV. From these two tables, we can see that the proposed algorithms lead to a few complexity increases for the V-PCC encoder. This is because of the nearest points search in the unoccupied-occupied padding algorithm in both the RA and AI cases. For the V-PCC decoder, the decoder is unchanged and therefore no obvious complexity change is observed. For the HEVC encoder, the proposed algorithm will decrease in complexity since the unoccupied-unoccupied padding algorithm use fewer RD operations, especially for the unoccupied blocks in RA case. For the HEVC decoder, the complexity is decreased because it chooses larger blocks for MC. For the AI case,



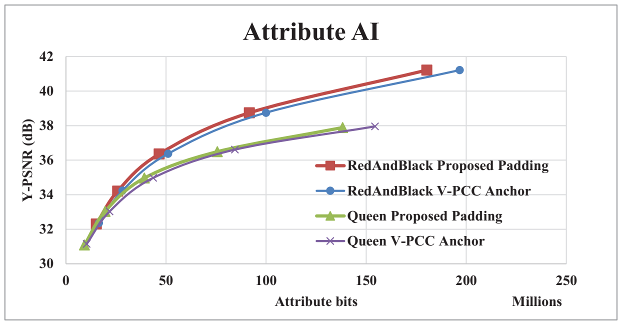
(a) Geometry RA



(b) Attribute RA



(c) Geometry AI



(d) Attribute AI

Fig. 10. Some examples of the RD curves.

the proposed algorithm does not lead to obvious complexity changes for either the HEVC encoder or decoder.

C. Performance of the unoccupied-occupied padding

The performances of the occupied-unoccupied padding method with θ equal to 2.0 in RA and AI cases are shown in Table VII and Table VIII, respectively. We can see that the padding of the unoccupied-occupied pixels can lead to an average of 2.2% and 4.3% performance improvements for D1 and D2 in RA case, respectively. The proposed algorithm achieves similar bitrate savings for the geometry in AI case. However, the proposed algorithm leads to a few performance losses especially for the Chroma components since the attribute video becomes less smooth after padding. For HEVC, the proposed algorithm will not lead to an observable complexity increase for both the encoder and decoder. For the V-PCC reference software, the proposed algorithm may lead to a slight encoder complexity increase caused by finding of the nearest neighbors of the current point. We also show one typical example of the subjective quality comparison as shown in Fig. 11. Here, we can obviously see that the proposed unoccupied-occupied padding can partially reduce the noisy points and lead to better subjective quality for the point cloud.

The reason behind the the improvements of the proposed unoccupied-occupied padding can be seen by comparing the number of missed points after projection and the number of duplicate points after reconstruction between the anchor and the proposed unoccupied-occupied padding as shown in IX. Note that the number of duplicate points after reconstruction is the average of all the rate points from r1 to r5. From Table IX, we can see that the number of missed points

TABLE VII
PERFORMANCE OF THE PADDING OF THE UNOCCUPIED-OCCUPIED PIXELS IN RA CASE TESTED WITH 32 FRAMES

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–1.4%	–3.9%	0.2%	0.4%	1.2%
RedAndBlack	–1.5%	–4.7%	–0.4%	–0.2%	–0.1%
Soldier	–1.3%	–3.6%	0.6%	3.2%	4.8%
Queen	–4.5%	–4.7%	0.0%	1.5%	1.0%
LongDress	–2.5%	–4.7%	–1.6%	–2.1%	–1.6%
Avg.	**–2.2%**	**–4.3%**	**–0.2%**	**0.6%**	**1.1%**
Enc. self	107%				
Dec. self	100%				
Enc. child	101%				
Dec. child	100%				

TABLE VIII
PERFORMANCE OF THE PADDING OF THE UNOCCUPIED-OCCUPIED PIXELS IN AI CASE TESTED WITH 32 FRAMES

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–1.3%	–3.9%	0.0%	–0.1%	0.5%
RedAndBlack	–1.7%	–4.7%	0.2%	0.3%	0.4%
Soldier	–1.2%	–4.0%	0.3%	1.2%	1.2%
Queen	–4.4%	–4.8%	1.2%	0.8%	1.3%
LongDress	–1.2%	–3.5%	–0.3%	0.3%	0.4%
Avg.	**–2.0%**	**–4.2%**	**0.3%**	**0.5%**	**0.7%**
Enc. self	107%				
Dec. self	99%				
Enc. child	100%				
Dec. child	100%				

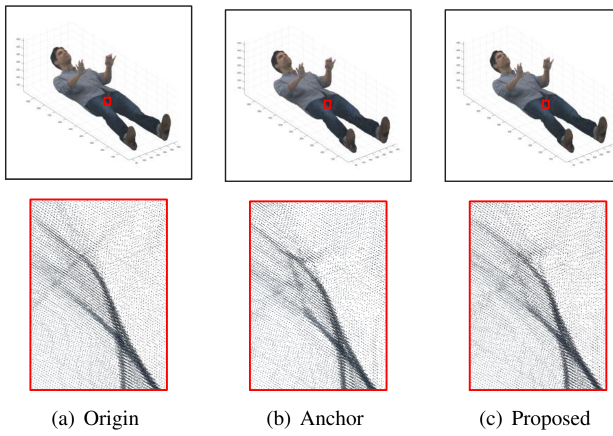


Fig. 11. Typical example of the subjective quality comparison. The test point cloud "Loot" with picture order count 1000 under r4 in RA case.

TABLE IX
COMPARISONS OF THE NUMBER OF MISSED POINTS AND DUPLICATE POINTS UNDER THE ANCHOR AND THE PROPOSED UNOCCUPIED-OCCUPIED PADDING IN RA CASE

Test point cloud	Missed points		Duplicate points	
	anchor	proposed	anchor	proposed
Loot	109	106	440461	446985
RedAndBlack	983	931	387439	394802
Soldier	348	330	559069	571041
Queen	4598	4520	306302	315097
LongDress	1323	1258	421661	427674

decreases. This means there is a greater number of real points in the reconstructed point cloud, which leads to its improved reconstructed quality. Additionally, the number of duplicate points increases, which indicates that the proposed unoccupied-occupied padding is reducing the number of noisy points. These two observations clearly explain the improvements of the proposed algorithm.

We compare the average performances of the proposed algorithm using different θs in RA case as shown in Table X. We can see that the proposed algorithm with θ equal to 1.0 will lead to geometry compression improvements compared with the V-PCC anchor. However, due to the limited number of padded points, the performance improvement is limited. We can also see from Table X that the proposed algorithm with θ equal to 3.0 can lead to slightly higher performance improvements compared to θ equal to 2.0 for the geometry under the D2 quality measurement. However, this value of θ will lessen the smoothness and lead to a bitrate increase for the attribute. Therefore, we choose θ as 2.0 in order to achieve a better balance between the compression performances of the geometry and attribute.

We also compare the performance of changing the number of nearest points K as shown in Table XI. As we can see, the RD performance keeps improving as the value of K increases. Simultaneously, the improvement speed keeps decreasing. Therefore, 128 is a suitable threshold for the proposed padding algorithm.

TABLE X
COMPARISONS OF THE PROPOSED UNOCCUPIED-OCCUPIED PADDING IN RA CASE USING DIFFERENT θS

θ	Geom.BD-GeomRate		Attr.BD-AttrRate		
	D1	D2	Luma	Cb	Cr
1.0	–0.6%	–1.2%	0.1%	0.5%	0.4%
2.0	–2.2%	–4.3%	–0.2%	0.6%	1.1%
3.0	–1.9%	–5.3%	1.5%	3.1%	2.7%

TABLE XI
COMPARISONS OF THE PROPOSED UNOCCUPIED-OCCUPIED PADDING IN RA CASE USING DIFFERENT NUMBER OF NEAREST POINTS K

K	Geom.BD-GeomRate		Attr.BD-AttrRate		
	D1	D2	Luma	Cb	Cr
16	–1.6%	–3.2%	0.0%	0.7%	0.2%
32	–2.0%	–4.0%	–0.1%	1.2%	1.4%
64	–2.2%	–4.3%	–0.1%	0.8%	0.3%
128	–2.2%	–4.3%	–0.2%	0.6%	1.1%

D. Performance of the unoccupied-unoccupied padding

Table XII and Table XIII show the performance of the proposed unoccupied-unoccupied padding in RA and AI cases, respectively. In RA case, we can see that the proposed algorithm is able to achieve an average of 13.0% and 12.1% bitrate savings for the D1 and D2 compared to the V-PCC anchor. For the attribute, the proposed algorithm can bring an average of 15.7%, 13.6% and 13.2% performance improvements for the Luma, Cb, and Cr components, respectively. In AI case, the proposed algorithm can bring an average of 4.1% and 2.8% RD performance improvements for the D1 and D2 compared to the V-PCC anchor. For the attribute, average performance improvements of 6.4%, 2.5% and 3.2% are observed for the Luma, Cb, and Cr components. The experimental results obviously demonstrate that the proposed unoccupied-unoccupied padding algorithm brings significant bitrate savings compared with the V-PCC anchor. Additionally, we can see that the proposed unoccupied-unoccupied padding brings much better performance improvement in RA case compared with that in AI case. In essence, the unoccupied pixels are a result of the extrapolation of the occupied pixels. Therefore, the influences of the unoccupied-unoccupied pixels on the inter prediction are larger than that of the intra prediction.

To better explain the performance of the proposed unoccupied-unoccupied padding, we compare the reconstructed attribute of the proposed unoccupied-unoccupied padding with the V-PCC anchor as shown in Fig. 12. We can see from the red squares that the unoccupied-unoccupied pixels are padded to match the MV of the occupied pixels. Consequently, the bits of the current PU can be significantly reduced without influencing the quality of the occupied pixels. Also, when all the pixels of the current PU are unoccupied, the unoccupied pixels will pad with zero residues to minimize the bitrate. This figure obviously demonstrates that the proposed unoccupied-unoccupied padding can minimize the bitrate of the unoccupied pixels, and thus can improve the performance significantly.

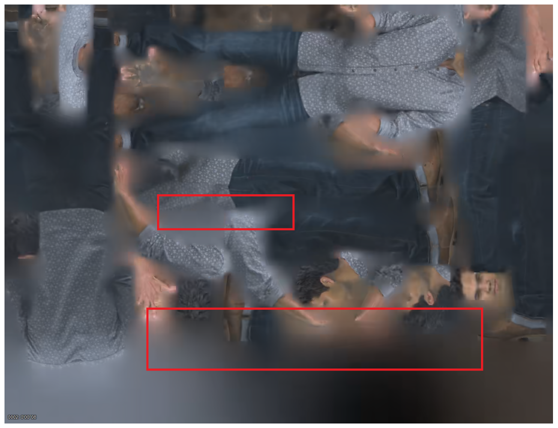
We also compare the proposed unoccupied-unoccupied padding with average padding to the proposed algorithm with

TABLE XII
PERFORMANCE OF THE AVERAGE PADDING OF THE UNOCCUPIED-UNOCCUPIED PIXELS IN RA CASE TESTED WITH 32 FRAMES

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–20.1%	–19.4%	–26.4%	–21.1%	–25.7%
RedAndBlack	–6.3%	–4.2%	–13.2%	–16.4%	–13.3%
Soldier	–16.0%	–15.6%	–16.8%	–9.1%	–9.1%
Queen	–14.9%	–15.1%	–16.7%	–15.4%	–12.7%
LongDress	–7.2%	–6.1%	–5.2%	–5.9%	–5.4%
Avg.	**–13.0%**	**–12.1%**	**–15.7%**	**–13.6%**	**–13.2%**
Enc. self	101%				
Dec. self	99%				
Enc. child	95%				
Dec. child	95%				

TABLE XIII
PERFORMANCE OF THE AVERAGE PADDING OF THE UNOCCUPIED-UNOCCUPIED PIXELS IN AI CASE TESTED WITH 32 FRAMES

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–6.4%	–4.1%	–8.9%	0.8%	–4.7%
RedAndBlack	–4.0%	–2.0%	–7.7%	–7.4%	–7.2%
Soldier	–2.9%	–1.6%	–2.9%	5.5%	6.0%
Queen	–3.0%	–3.3%	–9.2%	–9.1%	–7.9%
LongDress	–3.9%	–3.0%	–3.1%	–2.4%	–2.1%
Avg.	**–4.1%**	**–2.8%**	**–6.4%**	**–2.5%**	**–3.2%**
Enc. self	101%				
Dec. self	101%				
Enc. child	102%				
Dec. child	100%				


(a) V-PCC anchor padding

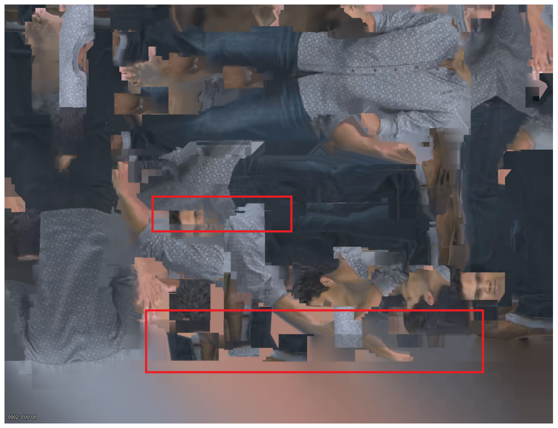

(b) Proposed unoccupied-unoccupied padding

Fig. 12. Typical examples of the reconstructed frames with POC 2 from the DPC "Loot" encoded under bitrate r4 defined in the CTC using V-PCC anchor and the proposed unoccupied-unoccupied padding, respectively.

zero padding in RA case as shown in Table XIV. Here, we can see that the average padding can bring an average of over 4% bitrate improvements for the geometry. For the attribute, the average padding can bring an average of 1.3%, 17.4%, and 13.3% bitrate savings for the Luma, Cb, and Cr components. The experimental results clearly demonstrate the effectiveness of the average padding compared with the zero padding. In addition, we can see that the average padding can achieve the best performance for the chroma components, the second best for the geometry, and the worst for the Luma component. This is in essence determined by the smoothness of the content. For chroma, as the smoothest of the three components, the zero padding may lead to serious unsmoothness in the residue, leading to a significant increase in the bitrate.

In addition, we present the RD performance improvement when the proposed unoccupied-unoccupied padding is combined with the occupancy-map-based RDO [52] in RA case in Table XV. Comparing Table XII and Table XV, we can see that combining the proposed algorithm with the occupancy-map-based RDO brings extra 1.9% bitrate savings for the geometry under the quality measurements for both D1 and D2. For the attribute, RD performance improvements of an extra 1.7%, 0.9%, and 0.6% for Luma, Cb, and Cr components are achieved, respectively. These experimental results obviously demonstrate that through explicitly ignoring the distortions of the unoccupied pixels, we can obtain some extra performance improvements.

TABLE XIV
PERFORMANCE OF THE AVERAGE PADDING COMPARED WITH THE ZERO PADDING IN RA CASE

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–2.9%	–2.8%	–0.1%	–20.2%	–19.8%
RedAndBlack	–3.7%	–3.4%	–1.9%	–19.8%	–2.1%
Soldier	–4.3%	–4.1%	–2.3%	–24.9%	–26.4%
Queen	–6.8%	–6.7%	–2.0%	–16.9%	–13.7%
LongDress	–3.6%	–3.5%	–0.5%	–5.5%	–4.3%
Avg.	**–4.3%**	**–4.1%**	**–1.3%**	**–17.4%**	**–13.3%**
Enc. self	100%				
Dec. self	99%				
Enc. child	102%				
Dec. child	100%				

E. Comparisons of the individual algorithms with the combination of them

From the experimental results shown above, we can see that the combination of the proposed algorithms outperforms the two, individually. For geometry, the combination is better than the sum of the two algorithms, individually. When we pad the unoccupied-occupied pixels with the real points from the original point cloud, the currently occupied block has a higher probability of finding a corresponding block. The higher probability results in higher performance improvement for the

TABLE XV
PERFORMANCE OF THE PADDING OF THE UNOCCUPIED-UNOCCUPIED PIXELS COMBINED WITH OCCUPANCY-MAP-BASED RDO IN RA CASE

Test	Geom.BD-GeomRate		Attr.BD-AttrRate		
point cloud	D1	D2	Luma	Cb	Cr
Loot	–22.6%	–21.6%	–28.5%	–22.5%	–25.7%
RedAndBlack	–8.3%	–6.2%	–14.7%	–14.5%	–15.1%
Soldier	–17.7%	–17.3%	–18.3%	–11.4%	–8.4%
Queen	–16.7%	–16.8%	–18.3%	–17.8%	–13.6%
LongDress	–9.4%	–8.3%	–7.1%	–6.3%	–6.4%
Avg.	**–14.9%**	**–14.0%**	**–17.4%**	**–14.5%**	**–13.8%**
Enc. self	100%				
Dec. self	99%				
Enc. child	97%				
Dec. child	95%				

geometry. For attribute, however, the increased variance of the residue block due to the unoccupied-occupied pixels may reduce the coding efficiency. Therefore, the performance of the combination is worse than the sum of the performance of the two algorithms.

V. CONCLUSION

In this paper, we first point out that the unoccupied pixels in the state-of-the-art video-based point cloud compression (V-PCC) may lead to the inefficiency of the video compression. We divide the unoccupied pixels into two groups based on whether they are signaled as occupied or unoccupied: the unoccupied-occupied pixels and the unoccupied-unoccupied pixels. For the unoccupied-occupied pixels, we propose padding them using the real points in the original point cloud to increase the number of projected points and reduce the number of duplicate points. For the unoccupied-unoccupied pixels, we propose padding them adaptively according to the corresponding prediction block and residue block to reduce the bitrate as much as possible. The proposed algorithms are implemented in the reference softwares of the state-of-the-art V-PCC and the corresponding video compression framework High Efficiency Video Coding. The experimental results show that the proposed algorithms can bring about 16% bitrate savings for the geometry and attribute compared with the V-PCC anchor. The experimental results obviously demonstrate the effectiveness of the proposed algorithms. In the future, we will further investigate more suitable video compression frameworks for the projected videos from the dynamic point cloud.

REFERENCES

[1] G. Bruder, F. Steinicke, and A. Nüchter, "Poster: Immersive Point Cloud Virtual Environments," in *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, 2014, pp. 161–162.

[2] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[3] M.-L. Champel, R. Doré, and N. Mollet, "Key Factors for a High-Quality VR Experience," in *2017 SPIE Optical Engineering and applications*, vol. 10396, 2017.

[4] C. Tulvan, R. Mekuria, Z. Li, and S. Laserre, "Use Cases for Point cloud compression (PCC)," Document ISO/IEC JTC1/SC29/WG11 MPEG2015/N16331, Geneva, CH, Jun. 2016.

[5] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "Input to Ad Hoc Groups on MPEG Point Cloud Compression and JPEG PLENO," Document ISO/IEC JTC1/SC29/WG11 m40059, Geneva, Switzerland, Jan. 2017.

[6] R. Schnabel and R. Klein, "Octree-based Point-Cloud Compression," in *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, 2006, pp. 111–121.

[7] S. Park and S. Lee, "Multiscale Representation and Compression of 3-D Point Data," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 177–183, Jan. 2009.

[8] B. Kathariya, L. Li, Z. Li, J. Alvarez, and J. Chen, "Scalable Point Cloud Geometry Coding with Binary Tree Embedded Quadtree," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[9] C. Zhang, D. Florncio, and C. Loop, "Point Cloud Attribute Compression with Graph Transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 2066–2070.

[10] R. L. de Queiroz and P. A. Chou, "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[11] K. Mammou, A. M. Tourapis, D. Singer, and Y. Su, "Video-based and Hierarchical Approaches Point Cloud Compression," Document ISO/IEC JTC1/SC29/WG11 m41649, Macau, China, Oct. 2017.

[12] D. Thanou, P. A. Chou, and P. Frossard, "Graph-Based Compression of Dynamic 3D Point Cloud Sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, Apr. 2016.

[13] 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.

[14] S. Schwarz, M. M. Hannuksela, V. Fakour-Sevom, N. Sheiki-Pour, V. Malamalvadakital, and A. Aminlou, "Nokias Response to CfP for Point Cloud Compression (Category 2)," Document ISO/IEC JTC1/SC29/WG11 m41779, Macau, China, Oct. 2017.

[15] L. He, W. Zhu, and Y. Xu, "Best-Effort Projection based Attribute Compression for 3D Point Cloud," in *2017 23rd Asia-Pacific Conference on Communications (APCC)*, Dec. 2017, pp. 1–6.

[16] G. J. 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. 22, no. 12, pp. 1649–1668,

Dec. 2012.

[17] M. Preda, "Report on PCC CfP Answers," Document ISO/IEC JTC1/SC29/WG11 w17251, Macau, China, Oct. 2017.

[18] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokua, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[19] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The Lumigraph," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, 1996, pp. 43–54.

[20] M. Li, Y. Chang, F. Yang, and S. Wan, "Rate-Distortion Criterion Based Picture Padding for Arbitrary Resolution Video Coding Using H.264/MPEG-4 AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 9, pp. 1233–1241, Sept. 2010.

[21] L. Li, Z. Li, X. Ma, H. Yang, and H. Li, "Advanced Spherical Motion Model and Local Padding for 360 Video Compression," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2342–2356, May 2019.

[22] M. Botsch, A. Wiratanaya, and L. Kobbelt, "Efficient High Quality Rendering of Point Sampled Geometry," in *Proceedings of the 13th Eurographics workshop on Rendering*, 2002, pp. 53–64.

[23] J. Peng and C.-C. J. Kuo, "Geometry-Guided Progressive Lossless 3D Mesh Coding with Octree (OT) Decomposition," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 609–616, 2005.

[24] Y. Huang, J. Peng, C. . J. Kuo, and M. Gopi, "A Generic Scheme for Progressive Point Cloud Coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440–453, Mar. 2008.

[25] V. Kämpe, E. Sintorn, and U. Assarsson, "High resolution sparse voxel DAGs," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 101, 2013.

[26] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-Based Static 3D Point Clouds Geometry Coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 284–299, Feb. 2019.

[27] J. Smith, G. Petrova, and S. Schaefer, "Progressive Encoding and Compression of Surfaces Generated from Point Cloud Data," *Computers & Graphics*, vol. 36, no. 5, pp. 341–348, 2012.

[28] Y. Fan, Y. Huang, and J. Peng, "Point Cloud Compression Based on Hierarchical Point Clustering," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 2013, pp. 1–7.

[29] R. Schnabel, S. Möser, and R. Klein, "Fast Vector Quantization for Efficient Rendering of Compressed Point-Clouds," *Computers & Graphics*, vol. 32, no. 2, pp. 246–259, 2008.

[30] T. Golla and R. Klein, "Real-Time Point Cloud Compression," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2015, pp. 5087–5092.

[31] J. Ahn, K. Lee, J. Sim, and C. Kim, "Large-Scale 3D Point Cloud Compression Using Adaptive Radial Distance Prediction in Hybrid Coordinate Domains," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 3, pp. 422–434, Apr. 2015.

[32] P. A. Chou, M. Krivokuca, G. Cernigliaro, and E. dEon, "Point Cloud Compression using a Blockable Geometry Representation and Region Adaptive Hierarchical Transform," ISO/IEC JTC1/SC29/WG11 m41645, Macau, China, Oct. 2017.

[33] W. Zhu, Y. Xu, L. Li, and Z. Li, "Lossless Point Cloud Geometry Compression via Binary Tree Partition and Intra Prediction," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2017, pp. 1–6.

[34] R. L. de Queiroz and P. A. Chou, "Transform Coding for Point Clouds Using a Gaussian Process Model," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3507–3517, 2017.

[35] B. Dado, T. R. Kol, P. Bauszat, J.-M. Thiery, and E. Eisemann, "Geometry and Attribute Compression for Voxel Scenes," in *Computer Graphics Forum*, vol. 35, no. 2, 2016, pp. 397–407.

[36] V. Morell, S. Orts, M. Cazorla, and J. Garcia-Rodriguez, "Geometric 3D Point Cloud Compression," *Pattern Recognition Letters*, vol. 50, pp. 55–62, 2014.

[37] K. Mammou, A. Tourapis, J. Kim, F. Robinet, V. Valentin, and Y. Su, "Lifting Scheme for Lossy Attribute Encoding in TMC1," Document ISO/IEC JTC1/SC29/WG11 m42640, San Diego, CA, US, Apr. 2018.

[38] V. Zakharchenko, B. Kathariya, and J. Chen, "[G-PCC] [CE13.15] Response on Level of Detail Generation using Binary Tree for Lifting Transform," Document ISO/IEC JTC1/SC29/WG11 m45966, Marrakech, MA, Jan. 2019.

[39] K. Mammou, A. Tourapis, and J. Kim, "[G-PCC][New Proposal] Efficient Low-Complexity LOD Generation," ISO/IEC JTC1/SC29/WG11 m46188, Marrakech, MA, Jan. 2019.

[40] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-Time Compression of Point Cloud Streams," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 778–785.

[41] R. L. de Queiroz and P. A. Chou, "Motion-Compensated Compression of Dynamic Voxelized Point Clouds," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017.

[42] S. Lasserre, J. Llach, C. Guede, and J. Ricard, "Technicolors Response to the CfP for Point Cloud Compression," Document ISO/IEC JTC1/SC29/WG11 m41822, Macau, China, Oct. 2017.

[43] M. Budagavi, E. Faramarzi, T. Ho, H. Najaf-Zadeh, and I. Sinharoy, "Samsungs Response to CfP for Point Cloud Compression (Category 2)," Document ISO/IEC JTC1/SC29/WG11 m41808, Macau, China, Oct. 2017.

[44] D. Graziosi and A. Tabatabai, "[V-PCC] New Contribution on Geometry Padding," Document ISO/IEC JTC1/SC29/WG11 m47496, Geneva, CH, Mar. 2019.

[45] E.-C. Ke, S.-P. Wang, Y.-T. Tsai, C.-C. Lin, C.-L. Lin, Y.-H. Lee, J.-L. Lin, Y.-C. Chang, and C.-C. Ju, "[V-PCC] [New proposal] Patch Expansion for Improving Visual Quality," Document ISO/IEC JTC1/SC29/WG11 m47772, Geneva, CH, Mar. 2019.

[46] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[47] D. Graziosi, "[V-PCC] TMC2 Optimal Texture Packing," Document ISO/IEC JTC1/SC29/WG11 m43681, Ljubljana, SI, Jul. 2018.

[48] K. Mammou, J. Kim, V. Valentin, F. Robinet, A. Tourapis, and Y. Su, "CE2.12 Related: Sparse Linear Model Based Padding Method for the Texture Images," Document ISO/IEC JTC1/SC29/WG11 m44837, Macau, CH, Oct. 2018.

[49] D. Graziosi, "V-PCC New Proposal (related to CE2.12): Harmonic Background Filling," Document ISO/IEC JTC1/SC29/WG11 m46212, Marrakesh, MA, Jan. 2019.

[50] E. Faramarzi and M. Budagavi, "[V-PCC] [New Proposal] Improved Texture Padding," Document ISO/IEC JTC1/SC29/WG11 m46202, Marrakesh, MA, Jan. 2019.

[51] S. Rhyu, Y. Oh, and J. Woo, "PCC CE2.13 Report on Texture and Depth Padding Improvement," Document ISO/IEC JTC1/SC29/WG11 m43667, Ljubljana, SI, Jul. 2018.

[52] L. Li, Z. Li, S. Liu, and H. Li, "Occupancy-Map-Based Rate Distortion Optimization for Video-Based Point Cloud Compression," arXiv:1902.04169, 2019.

[53] "Point Cloud Compression Category 2 Reference Software, TMC2-4.0," http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc2.git, accessed: 2019.

[54] "High Efficiency Video Coding Test Model, HM-16.18+SCM8.7," https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/, accessed: 2019.

[55] S. Schwarz, G. Martin-Cocher, D. Flynn, and M. Budagavi, "Common Test Conditions for Point Cloud Compression," Document ISO/IEC JTC1/SC29/WG11 w17766, Ljubljana, Slovenia, Jul. 2018.

[56] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-Curves," Document VCEG-M33, Austin, Texas, USA, Apr. 2001.