IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 9, 2021, accepted May 22, 2021, date of publication May 27, 2021, date of current version June 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3084180

Fast Grid-Based Refining Segmentation Method in Video-Based Point Cloud Compression

JIEON KIM AND YONG-HWAN KIM
Korea Electronics Technology Institute, Seongnam-si 13488, South Korea

Corresponding author: Jieon Kim (jieon.kim@keti.re.kr)

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2020-0-00920, Development of Ultra High Resolution Unstructured Plenoptic Video Storage/Compression/Streaming Technology for Medium to Large Space).

ABSTRACT The video-based point cloud compression (V-PCC, ISO/IEC 23090-5) is the state-of-the-art international standard for compressing dynamic point clouds developed by the moving picture experts group (MPEG). It has been achieved good rate-distortion (RD) performance by employing the 2D-based dynamic point cloud compression. As a brief look, V-PCC first converts the 3D input point cloud into a set of 2D patches followed by a packing process. The packing process then maps the patches into a 2D grid. Such a way allows compressing the patches utilizing the existing video coding standards. Besides the RD performance, complexity is another vital factor to consider in performance evaluations. In the V-PCC encoder, the self-time accounts for on average 15.9% and a maximum of 48.2% of the total-time, which can be a hindrance to realizing real-time V-PCC applications. One of the most computationally intensive modules of V-PCC is the grid-based refining segmentation (G-RS). Thus this paper proposes a fast G-RS method that can adaptively select voxels that need the refining segmentation. More concretely, the proposed method classifies the voxels based on the projection plane indices of 3D points and only applies the refining process to the selected voxels. Experimental results demonstrate that the proposed method reduces the complexity of the refining steps in G-RS, on average, by 60.7% and 62.5% without coding efficiency loss compared to the test model for category 2 (TMC2) version 12.0 reference software under the random access (RA) and all-intra (AI) configurations, respectively.

INDEX TERMS Video-based point cloud compression, dynamic point cloud, fast algorithms, low-complex video encoders, voxel-based refining segmentation, encoder optimization.

I. INTRODUCTION

Advances in three-dimensional (3D) capture technologies have opened a new chapter in 3D sensing beyond virtual/augmented reality (VR/AR) content creation to smart factories, robots, and automated driving applications. Furthermore, since the digitalization of 3D space has allowed users to explore 3D content from any point of view, the attention of the market in VR/AR has been dramatically increased [1]. Volumetric 3D data represents 3D scenes and objects with their geometries, attributes, and temporal variations and is usually generated by 3D models on a computer or captured in a real-world environment using multiple cameras. Three main categories of the volumetric 3D data are static objects (category 1), dynamic objects (category 2), and dynamically acquired objects (category 3) [2]. This paper focuses on the dynamic point cloud.

A dynamic point cloud consists of consecutive static 3D point cloud frames. In viewing the V-PCC common test conditions (CTCs) [3], each 3D point cloud frame consists of 800,000-2,900,000 3D points, and a 3D point is stored with 10 bits to represent the geometry information and 8 bits for color (RGB) information. A maximum of 4.3 Gbps bandwidth is required to transmit such a point cloud sequence with a frame rate of 30 fps, and it is challenging in the current network environment [4]. Therefore, efficient point cloud compression (PCC) technologies are indispensable.

To respond to such a need, the moving picture experts group (MPEG) called for a proposal (CfP) for PCC in 2017. The test model video-based point cloud compression (V-PCC) is a project that was initiated after CfP for the dynamic point cloud, and MPEG released a final draft

The associate editor coordinating the review of this manuscript and approving it for publication was You Yang.

international standard (FDIS) of V-PCC as ISO/IEC 23090-5 in July 2020 [5].

V-PCC is to leverage the existing 2D video codecs [1], [6]. Such a way requires a process that converts the 3D point cloud to 2D videos, and the patch projection proposal by Mammou *et al.* [7] was finally adopted in V-PCC. Based on the similarity of 3D point normals, several 2D patches are generated for the geometry [8] and attribute [9] components, respectively. These patches are mapped to a regular 2D grid without overlapping. Further, occupancy maps are generated to indicate whether the sample belongs to the patches or not. The occupancy maps, geometry videos, and attribute videos are encoded using the conventional 2D video codecs. The V-PCC solution is video codec agnostic [6], but the high-efficiency video coding (HEVC) [10] is set as a default for video-based coding.

The patch projection-based V-PCC achieves superior RD performance for the dynamic point cloud [11]. The research efforts to improve the point cloud compression performance have been continued [12]–[17]. Queiroz *et al.* [12] proposed a method that splits the point cloud into voxel blocks and encodes them in an octree. Mekuria *et al.* [13] have exploited different sizes of blocks in octree voxel space for real-time tele-immersive video processing. Queiroz *et al.* [14], [15] utilized Laplace and Gaussian transforms in hierarchical sub-band transforms coding. Cohen *et al.* [16] extended the k-nearest neighbors algorithm (KNN) for sparsely-populated blocks [16]. However, these outstanding achievements come at a high computational complexity of V-PCC encoders.

The complexity in V-PCC can be classified into two categories: the 2D domain and the 3D domain complexities. The former complexity is mainly from the 2D video coding process, whereas the latter complexity is from the patch-related process in 3D domain. While a plethora of research work has been reported to reduce the complexity of the former case [18]–[23], this paper draws attention to the lack of research for the latter case in V-PCC [24]–[28]. Becerra *et al.* [26] reported the V-PCC encoder complexity utilizing TMC2 software. According to the report, the refining process is one of the most computationally intensive modules in the V-PCC encoder concerning execution time and memory allocation. Thus, the refining procedure has attracted considerable attention in studies that reduce the complexity of V-PCC encoders. Faramarzi *et al.* [24] proposed a software optimization method, which is a bit-exact match with the anchor software implementation. A grid-based refining segmentation (G-RS) method [25] was proposed to reduce the complexity of a point-based refining segmentation method, and its performance evaluated additionally in a dedicated core experiment (CE) [29]. Despite these efforts, the G-RS procedure still accounted for 91% to 77% of the self-time-consuming of the V-PCC encoders [26]. Later, Higa *et al.* [27] proposed a software optimization method to avoid unnecessary hash map access, and it was integrated into the TMC2 software. Seidel *et al.* [28] proposed a cache-friendly implementation of G-RS, which has already been integrated into the TMC2 software. Complexity is a requisite factor to consider for a rapid generalization of the pervasive and ubiquitous applications of state-of-the-art technologies. Therefore, we propose an efficient and fast algorithm in the 3D domain to reduce the complexity of V-PCC encoders.

One of the complex processes in the 3D domain is the refining steps in the patch generation process. Each point is associated with one of the projection planes according to its normal vector. Then, the initial projection plane index (PPI) is refined using its neighbors. The conventional refining steps in V-PCC are applied to all 3D points in the point cloud iteratively. However, it is unlikely all 3D points are required for the refining steps because most 3D points are likely to have accurate initial PPI.

This paper proposes a simple but highly efficient algorithm to reduce the computational complexity of V-PCC encoders in the 3D domain. Our goal is to use the simple uniformity index of PPI distribution in G-RS to select only a few voxels that need the refining steps, without overly complex operations. For this purpose, the proposed G-RS scheme is carried out by adaptively categorizing voxels, and a new voxel classification strategy is introduced in the presented method. The experimental results show the proposed method reduces an average of 60.7% and 62.5% of the self-time of refining steps without coding efficiency loss by the proposed method under the random access (RA) and all-intra (AI) configurations, respectively.

The rest of this paper is organized as follows. Section II introduces an overview of the V-PCC encoder, related work, and analysis of the V-PCC encoder. In Section III, the proposed algorithm is presented in detail. Section IV shows the extensive experimental results of the proposed method, and Section V concludes this paper.

II. BACKGROUND

This section firstly presents an overview of the V-PCC encoder. Further, a summary of the time-complexity analysis of V-PCC encoders is presented.

A. OVERVIEW OF V-PCC ENCODING PROCESS

There are two main steps in the V-PCC encoding process [1]: 1) conversion from 3D points to 2D videos and metadata; 2) video compression. The first step converts a 3D point cloud into 2D video sequences (*i.e.*, attribute and geometry videos [8], [9]) and additional metadata such as an occupancy map and auxiliary patch information, which are essential to interpret video sequences. This approach has been studied for many years and enables to leverage of the existing video coding standards for compressing 2D videos from a point cloud [7], [11], [30], [31]. Then, the second step produces compressed video bitstreams and metadata by the HEVC test model (HM) software [32] multiplexed together to generate the final V-PCC bitstream. In V-PCC, any video coding standard can be employed but HEVC is set as a default in TMC2. In this paper, we are mainly interested in G-RS. For further

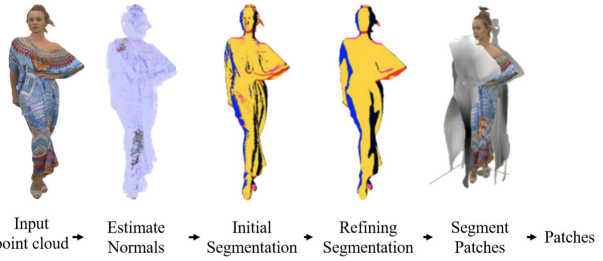

FIGURE 1. Patch generation process in V-PCC [1].

study, details related to G-RS are presented in the next. Other more detailed descriptions regarding V-PCC can be found in the literature [6], [33].

1) PATCH GENERATION

Fig.1 shows the patch generation process in V-PCC that is composed of four main procedures: estimate normals, initial segmentation, refining segmentation, segment patches. First, normal vectors at each 3D point are estimated as described in [34]. Based on the normal vectors, each 3D point is associated with the PPI in the initial segmentation. Here, the number of projection planes can be among 6, 10, and 18, and the default number is 6. The associated PPI may be unreliable due to inexact normal vectors. As it is fatal to the RD performance, G-RS refines the PPI using the neighbors. Then, the last step consists of extracting patches with smooth boundaries.

2) GRID-BASED REFINING SEGMENTATION

G-RS is a fast refining algorithm implemented in the TMC2 software [25]. While the point-based refining segmentation utilizes the individual PPI of nearest-neighboring 3D points in the refining process, the grid-based method utilizes the nearest-neighboring voxels.

a: PRE-PROCESSING STEPS

Firstly, the input point cloud in 3D space is divided into subset voxels by grids along x-, y-, and z- coordinates in a Cartesian coordinate system using the *voxDim* parameter Ψ. A 3D point is represented by a 3D vector and scaled by Ψ as follows:

$$\hat{\mathbf{P}} = \left[\hat{x}\ \hat{y}\ \hat{z}\right]^{\mathrm{T}} = \frac{1}{\Psi}\left[x\ y\ z\right]^{\mathrm{T}}, \tag{1}$$

where T denotes vector transpose operator.

If one or more 3D points exist in a voxel, it is labeled as a filled voxel $\hat{V}$, and the other case is labeled as an empty voxel. As for the each filled voxel $\hat{V}_i$, the nearest-neighboring voxels $\hat{H}_i(\cdot)$ are searched within the search range r_α using the K-D tree search and are represented as follows:

$$\begin{aligned} \hat{H}_i(r_\alpha) &= \{(\hat{x}_i, \hat{y}_i, \hat{z}_i) | d_x, d_y, d_z \in r_\alpha\}, \\ \text{where } d_x &= |\hat{x}_i - \hat{x}_j|, d_y = |\hat{y}_i - \hat{y}_j|, d_z = |\hat{z}_i - \hat{z}_j|. \end{aligned} \tag{2}$$

$\hat{x}_i, \hat{y}_i, \hat{z}_i$ represents a position of $\hat{V}_i$ along x-, y-, and z- coordinates in a Cartesian coordinate system. $\hat{x}_j, \hat{y}_j, \hat{z}_j$ represents a position of $\hat{V}_j$ which is a neighboring voxel of $\hat{V}_i$ along x-, y-, and z- coordinates. The default value of r_α is six for the Class A sequences and four for the other Class sequences.

b: REFINING STEPS

The refining steps in G-RS are performed every iteration. The PPI scores $S_i(\cdot)$ in each filled voxel $\hat{V}_i$ are calculated for each PPI p from 0 to $K-1$ as follows:

$$\begin{aligned} S_i(p) &= \sum_{m=0}^{M-1} \Phi_i^m(p), \\ \text{where } \Phi_i^m(p) &= \begin{cases} 1, & \Theta_i^m = p \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

Note that m is an index a 3D point in V_i. Θ_i^m represents p of m-th 3D point in $\hat{V}_i$. K represents the number of projection planes.

After obtaining $S_i(\cdot)$ of each voxel, the smooth score $\bar{S}_i(\cdot)$ is calculated using the PPI scores of its neighbors concerning p from 0 to $K-1$ as follows:

$$\bar{S}_i(p) = \sum_{j=0}^{N-1} S_j(p), \tag{4}$$

where N represents the total number of the nearest-neighbors of $\hat{V}_i$, which is equal to $|\hat{H}_i|$. Then, the score of normal vectors $N_i(\cdot)$ for each point in $\hat{V}_i$ is calculated as follows:

$$N_i^m(p) = \mathbf{n}_i^m \cdot \mathbf{c}(p), \tag{5}$$

where $\mathbf{n}_i^m$ is a normal vector of m-th 3D point in $\hat{V}_i$, $\mathbf{c}(p)$ represents the constant unit vector corresponding to the projection plane p, and $|N_i| \leq 1.0$.

Lastly, the final score of the m-th 3D point in $\hat{V}_i$ concerning the PPI p is calculated as follows:

$$\begin{aligned} F_i^m(p) &= N_i^m(p) + \Lambda \times \bar{S}_i(p), \\ \text{where } \Lambda &= \frac{\lambda}{M},\ \lambda = 3.0(\text{default}) \end{aligned} \tag{6}$$

Note M is the total number of 3D points of its neighbors. Accordingly, the updated PPI of m-th 3D point in $\hat{V}_i$ is determined according to

$$\acute{\Theta}_i^m = \arg\max_{p \in K} \{F_i^m(p)\}, \tag{7}$$

Consequently, G-RS is based on normal vectors of each 3D point and sum of PPI of its neighbor voxels.

B. TIME-COMPLEXITY PROFILING IN TMC2

The considerable complexity of the HEVC encoders [35] is beyond the scope of this paper. Our interest is the self-complexity of V-PCC encoders, which can be measured by subtracting HEVC encoding time and color conversion time from the total encoding time. Table 1 shows the average measured time under the AI and RA configurations. On average, it accounts for 38.4% and 15.9% of total encoding time under the AI and RA configurations, respectively. Although

TABLE 1. The average percentage of self-time of TMC2 12.0 over the total encoding time under the RA and AI configurations.

Class	Test point cloud	AI [%]	RA [%]
A	loot	41.6	18.7
	redandblack	37.9	14.7
	soldier	48.2	23.1
	queen	43.2	21.8
B	longdress	28.9	10.8
C	basketball_player	35.5	10.9
	dancer_player	33.4	11.0
	Average	38.4	15.9

TABLE 2. Time consumption ratio of each procedure in the patch generation process.

Class	Test point cloud	Estimate Normals	Initial Segmentation	Grid-based Refinement	Segment Patches
A	loot	24.95%	0.05%	63.08%	11.92%
	redandblack	23.44%	0.05%	63.34%	13.17%
	soldier	24.06%	0.05%	62.13%	13.76%
	queen	22.63%	0.05%	63.37%	13.95%
B	longdress	43.12%	0.09%	32.33%	24.46%
C	basketball_player	48.43%	0.09%	17.56%	33.92%
	dancer_player	50.77%	0.09%	18.58%	30.56%

it seems to be a small percentage in the entire encoding time, the encoding performance of the *basketball_player* point cloud is of 0.016 fps. Such can be an obstacle to the rapid spread of high-quality real-time V-PCC applications.

Becerra *et al.* [26] carried out a detailed complexity profiling to identify the complexity level of each coding module. According to the report, the patch generation procedure is the most complex in the V-PCC encoder.

The average time consumption ratio of each procedure in the patch generation is investigated and shown in Table 2. G-RS accounts for the most execution time in the patch generation process: an average of 62.9% for the Class A test sequences, 32.33% on average for the Class B sequence, and 18.0% or more on average for the Class C sequences.

Further, Fig 2 shows the average execution time of each step in G-RS. Note that *step*3 refers to finding the nearest-neighboring voxels in G-RS, and the *rest* represents steps that the refining PPI iteratively. The *step*3 and *rest* steps can be optimized further, which takes up on average 44% and 53% of the entire G-RS time, respectively. However, the complexity from *step*3 is within the library used for the K-D tree search. Thus, our goal is to reduce complexity of the refining steps but to avoid overly complicated schemes. Details are presented in the next section.

III. PROPOSED METHOD

Fig. 3 shows the overall proposed voxel classification scheme in G-RS. The proposed method first classifies a voxel as either direct edge-voxel (DE-V) or no edge-voxel (NE-V) according to *uniformity index*. The neighboring

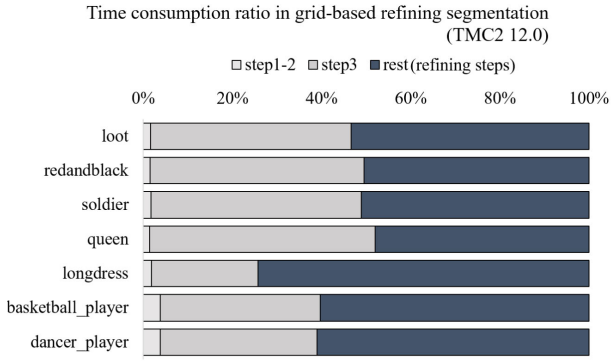

FIGURE 2. Time consumption ratio in the G-RS process with TMC2 12.0.

voxels labeled with NE-V of the current DE-V are classified as either NE-V or indirect edge-voxel (IDE-V) according to *extended uniformity index*. Based on the voxel classification, the refining steps are only limited to DE-V and IDE-V. This section presents details on the proposed method. Then, the overall algorithm is summarized.

A. VOXEL CLASSIFICATION

G-RS smooths out inaccurately associated PPI to obtain uniform regions for more accurate patch segmentation. It is intuitively clear that voxels with non-uniform PPI distribution are the candidates to be examined by G-RS. We distinguish three types of voxels:

a. Voxels with PPI variations within their voxel. They are denoted by DE-V and can be either a single-point (S-DE, Fig.4 (a)) or multi-points (M-DE, Fig.4 (b)) in a voxel. The former is usually isolated 3D points, and the latter indicates the presence of a point cloud surface.
b. Voxels with uniform PPI distribution within their voxel but non-uniform compared to the nearest-neighbors. They are clustered with discontinuity PPI from their neighbors and denoted by IDE-V.
c. Voxels with uniform PPI distribution within their voxel and the nearest-neighbors. These voxels are considered as uniform regions and they are denoted by NE-V (Fig.4 (c)).

In our approach, we employ a selective G-RS procedure, which treats three types of voxels differently. DE-V and NE-V are determined using *uniformity index*. Later, IDE-V is determined among NE-V using *extended uniformity index*. The following subsections present details on simple yet profoundly effective voxel-type decision schemes that can iteratively detect the refining candidates.

B. DIRECT EDGE-VOXEL AND NO EDGE-VOXEL DECISION

We introduce the notion of *uniformity index* that indicates whether the PPI distribution is uniform in a voxel. For each filled voxel, the PPI score S_i represents a histogram that indicates the self-statistics about the homogeneity of PPI in a voxel. From these statistics, we can estimate the uniformity level within a voxel in terms of segmentation. The *uniformity*

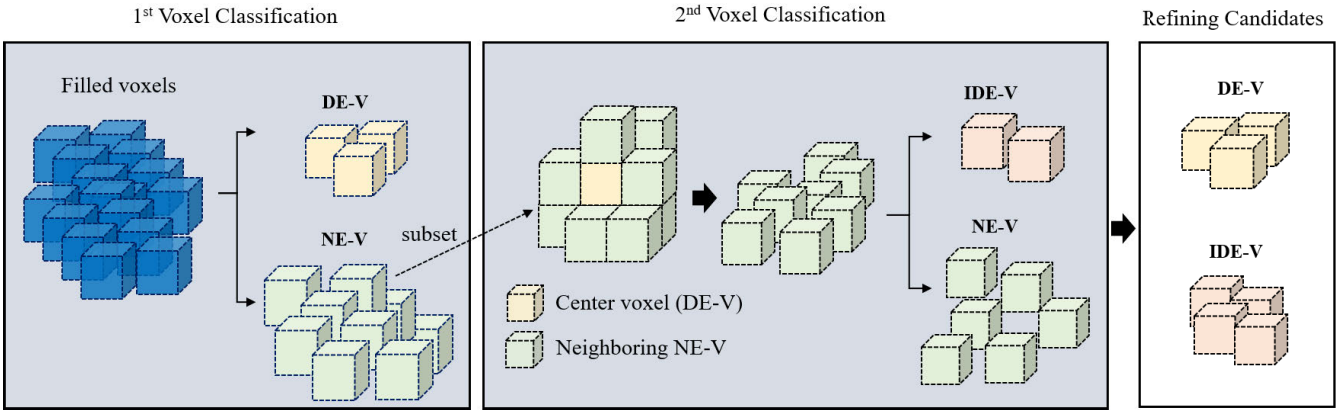

FIGURE 3. Proposed voxel classification scheme. DE-V stands for direct edge-voxel, IDE-V stands for indirect edge-voxel, and NE-V stands for no edge-voxel.

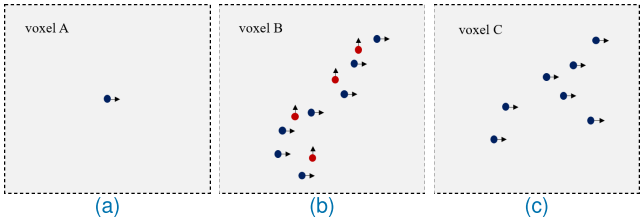

FIGURE 4. Examples of direct edge-, and no edge-voxels in a 2D domain. (a) Single 3D point in a voxel (S-DE) (b) Multiple 3D points in a voxel (M-DE). (c) No edge voxel. The color indicates the PPI of its 3D point.

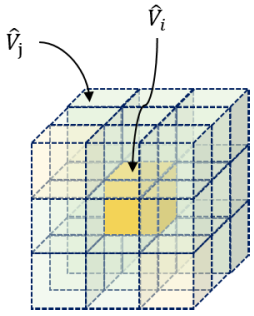

FIGURE 5. An example of $\hat{V}_i$ and its neighboring voxels $\hat{V}_j \in \hat{H}_i(1)$.

index U_i is defined using S_i as follows:

$$U_i = \frac{S_i(p_i^*)}{|S_i|}, \quad \text{where } p_i^* = \arg\max_{p \in K} S_i(p). \tag{8}$$

Note $|S_i|$ denotes the total number of 3D points in $\hat{V}_i$. Each filled voxel $\hat{V}_i$ are firstly classified based on U_i as below:

$$\hat{V}_i \in \begin{cases} \text{DE-V}, & U_i \neq 1 \\ \text{NE-V}, & U_i = 1. \end{cases} \tag{9}$$

C. INDIRECT EDGE-VOXEL DECISION

We extend the PPI reference range for the *uniformity index* from within a voxel to the nearest-neighbors and introduce a notion of *extended uniformity index* that indicates whether NE-V is still uniform compared to its neighbors. A histogram for the *extended uniformity index* is $\bar{S}_i$. For each filled voxel, the smooth score $\bar{S}_i$ represents a histogram that indicates the local-statistics about the homogeneity of PPI in a voxel within the defined reference range. Similar to *uniformity index*, we can estimate *extended uniformity index* $\hat{U}_i$ within the selected reference range as follows:

$$\hat{U}_i = \frac{S_i(\bar{p}_i^*)}{|S_i|}, \quad \text{where } \bar{p}_i^* = \arg\max_{p \in K} \bar{S}_i(p). \tag{10}$$

As shown in Table 3, NE-V typically constitutes most of a point cloud, and it is expensive to examine $\hat{U}_i$ for every NE-V. For this reason, IDE-V candidates $\hat{V}_j$ are constrained to the neighbors of $\hat{V}_i$ to reduce the amount of necessary computation and memory access for obtaining $\bar{S}_j$. However, to achieve the best RD performance and speed the G-RS process even further, we have limited the distance between $\hat{V}_i$ and $\hat{V}_j$ for determining IDE-V. The distance is denoted by r_β, and $\hat{U}_j$ in the refining process of $\hat{V}_i$ is represented as follows:

$$\hat{U}_j = \frac{S_j(\bar{p}_i^*)}{|S_j|}, \quad \text{where } \hat{V}_j \in H_i(r_\beta), \quad \bar{p}_i^* = \arg\max_{p \in K} \bar{S}_i(p). \tag{11}$$

Because we only consider NE-V in the second classification, $\hat{U}_j$ can be further simplified by comparing p_j^* and $\bar{p}_i^*$ ($\hat{U}_j = \bar{p}_i^*/p_j^*$). NE-V is finally classified based on $\hat{U}_j$ as below:

$$\hat{V}_j \in \begin{cases} \text{IDE-V}, & U_j = 1 \text{ and } \hat{U}_j \neq 1 \\ \text{NE-V}, & U_j = 1 \text{ and } \hat{U}_j = 1. \end{cases} \tag{12}$$

D. OVERALL ALGORITHM

Fig.6 shows a flowchart of the proposed algorithm, and the additional steps by the proposed algorithm are represented with a dotted line and summarized as follows:

TABLE 3. Distribution of the three types of voxels in V-PCC.

Class	Test point cloud	M-DE [%]	S-DE [%]	NE-V [%]
A	loot	10.13	11.90	77.97
	redandblack	11.59	13.46	74.95
	soldier	11.45	13.91	74.64
	queen	11.20	11.43	77.37
B	longdress	23.22	6.13	70.65
C	basketball_player	18.04	5.71	76.25
	dancer_player	19.20	5.99	74.81
	Average	14.98	9.79	75.23

Pre-processing steps:

Step 1. Partitioning the coordinate space into voxels, Eq. (1)

Step 2. Tagging the filled voxels in the grid.

Step 3. Searching $\hat{H}$ of the filled voxels, Eq. (2)

Refining steps:

Step 4-1) Calculating S for each filled voxel (Eq. (3))

Step 4-2) 1st voxel classification according to Eq. (9)

Step 5-1) Checking whether $\hat{V}_i$ is in a list of the total filled voxels.

Step 5-2) If $\hat{V}_i \in$ {DE-V, IDE-V}, go to Step 5-3. Otherwise, go to Step 5-1.

Step 5-3) Calculating $\bar{S}_i$, Eq. (4).

Step 5-4) 2nd voxel classification according to Eq. (12).

Step 6 - 7) Following the conventional way.

IV. EXPERIMENTAL RESULTS

In this section, the performance of the proposed method is evaluated. First, the experimental conditions are introduced. Further, the overall performance of the proposed algorithm is compared with TMC2 12.0. Lastly, the performance of the proposed method is analyzed with different values of encoding parameters.

A. EXPERIMENTAL CONDITIONS

The proposed method was implemented in the V-PCC reference software TMC2 12.0 [36]. As shown in Table 4, the performance was evaluated for lossy geometry and lossy attributes under the AI and RA configurations, respectively. The seven dynamic point cloud sequences, defined in V-PCC common test conditions (CTCs), were used for experiments. The detailed descriptions of the point clouds are in Table 5, and each test point cloud is shown in Fig. 7. The performance evaluation of the proposed algorithm complied with the number of frames presented in Table 5. We used the five quantization parameters (QP) are in Table 6, from low bitrate (R01) to high bitrate (R05) [37].

Furthermore, since the bitrates generated by the anchor and the proposed algorithm are not the same, the Bjontegaard-delta-rate (BD-rate) [37] was used to compare the respective rate-distortion (RD) performances. The BD-rates are reported for the geometry and attribute independently. For the geometry, the BD-rates were reported in terms of both

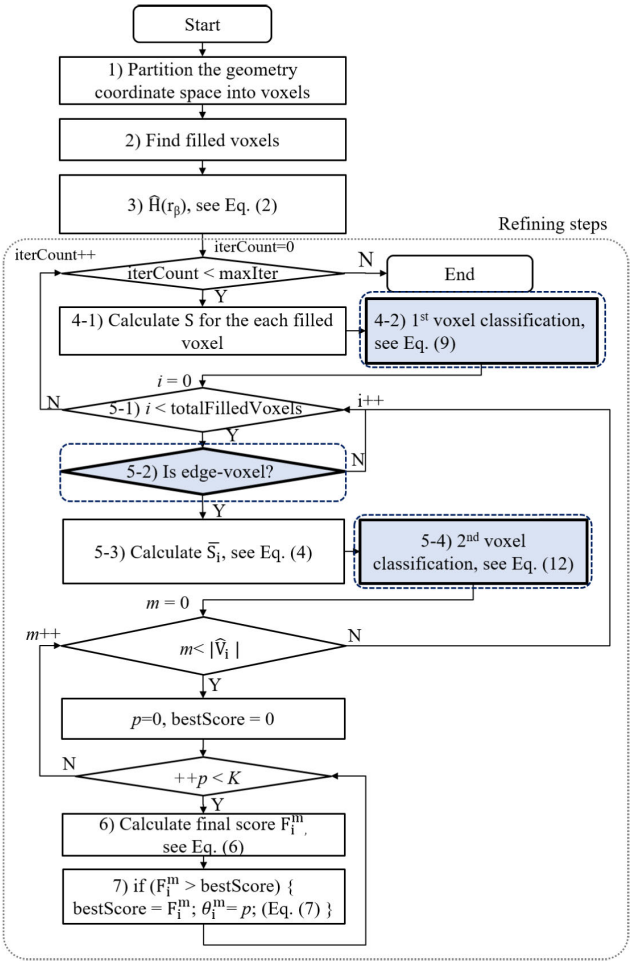

FIGURE 6. Flowchart of the proposed grid-based refining segmentation procedure in TMC2.

TABLE 4. Test Conditions.

OS	Windows 10
CPU	Intel Xeon E5-2687 v4 @ 3.00GHz
Memory	128GB
Reference Software	TMC2 v12.0
Video Compression	HM16.20-SCM8.8
GOP Structure	Random Access (RA) and All-Intra (AI)

point-to-point PSNR (D1) and point-to-plane PSNR (D2) [3]. Besides, the BD-rates were represented for the Luma, Cb, and Cr components for the attribute. To compare the complexity reduction, we made changes to the V-PCC reference software and, therefore, to report the self-time of the refining steps, G-RS, and the V-PCC encoder. As an index of the complexity reduction, the self-time was compared respectively, as follows:

$$\Delta T[\%] = \frac{1}{R_N} \sum_{i=0}^{R_N-1} \frac{T_A(i) - T_P(i)}{T_A(i)} \times 100, \qquad (13)$$

FIGURE 7. 3D point cloud sequences. (a) loot (b) redandblack (c) soldier (d) queen (e) longdress (f) basketball_player (g) dancer_player.

TABLE 5. Test point cloud.

Class	Test point cloud	Pts	Geometry Precision	fps	Frames
A	loot	∼780,000	10-bit	30	300
	redandblack	∼700,000	10-bit	30	300
	soldier	∼1,500,000	10-bit	30	300
	queen	∼1,000,000	10-bit	50	250
B	longdress	∼800,000	10-bit	30	300
C	basketball_player	∼2,900,000	11-bit	30	64
	dancer_player	∼2,600,000	11-bit	30	64

TABLE 6. Quantisation parameters of the common test conditions.

Variant	Geometry Video	Texture Video	Occupancy Precision
R01	32	42	4
R02	28	37	4
R03	24	32	4
R04	20	27	4
R05	16	22	2

where T_A and T_P are the self-time of original TMC2 and TMC2 with the proposed method respectively. R_N represents the total number of variants. A higher value of ΔT represents more time saving by the proposed method, and a negative BD-rate indicates that the proposed method achieves a better coding efficiency than the V-PCC anchor.

In the following subsections, firstly, the overall performance of the proposed fast algorithm is introduced. Then the individual performance of the proposed algorithm with different encoding options is presented.

B. OVERALL PERFORMANCE

Tables 7 and 8 show the performance of the proposed method under the AI and RA configurations, respectively. Note that r_β for IDE-V is set to 2 for the Class A test point cloud and 1 for the rest of test point cloud sequences in the overall experimental result. The performance comparisons depending on r_β are discussed in the next subsection. Besides, ΔT_T, ΔT_G, and ΔT_R are represented the time saving of the self-time of the V-PCC encoder, G-RS, and the refining steps, respectively. From Table 7, the proposed method can reduce an average of 18.0%, 26.9%, and 62.5% of self-time of V-PCC encoder, G-RS and the refining step compared with the V-PCC anchor respectively in the AI case. As a trade-off between complexity and RD coding performance, the proposed method provides a fast refining procedure based on edge-voxels in G-RS procedure without coding loss. The performance loss of Luma, Cb, and Cr components are only on average 0.04%, 0.06%, and 0.08%, respectively. Besides, the proposed method can achieve an average of -0.02% and -0.03% BD-rate gain compared with the V-PCC anchor for the D1 and D2 in the AI case, respectively.

Table 8 shows the proposed method saves on average of 17.1%, 28.2%, and 60.7% of self-time of V-PCC encoder, G-RS, and the refining step compared with the V-PCC anchor in the RA case without coding loss, which is on average, 0.06%, -0.03%, and -0.27% for the Luma, Cb, and Cr components, respectively. Also, the proposed algorithm can achieve an average of 0.12% and 0.14% BD-rate loss compared with the V-PCC anchor for the D1 and D2 in the RA configuration, respectively. From these experimental results, we had a conclusion that the proposed algorithm significantly reduces the computational complexity of the refining step in G-RS procedure of the V-PCC anchor without coding loss.

Fig. 8 shows a typical example of the subjective quality comparison between the V-PCC anchor and the proposed method in the soldier test point cloud, whose RD performance loss is the largest among the test sequences. As can be seen, overall, the edges of the point cloud were well preserved by the proposed method. Moreover, some parts in the reconstructed point cloud by the proposed algorithm show better subjective quality than the anchor, for example, the shoulder part as shown in Fig. 8. However, the differences are pretty trivial. Therefore, the proposed method reduced the

TABLE 7. Performance of the proposed algorithm compared with the V-PCC anchor under the AI configuration.

Class	Test point cloud	Geom.BD-GeomRate		Attr.BD-AttrRate			Geom.BD-TotalRate		Attr.BD-TotalRate			ΔT_T	ΔT_G	ΔT_R
		D1	D2	Luma	Cb	Cr	D1	D2	Luma	Cb	Cr			
A	loot	0.01%	0.00%	0.02%	-0.52%	0.05%	0.05%	0.01%	0.01%	-0.35%	0.04%	17.6%	21.8%	67.3%
	redandblack	-0.01%	-0.06%	0.02%	0.02%	0.06%	-0.01%	-0.08%	0.02%	0.02%	0.05%	15.6%	19.7%	62.4%
	soldier	0.02%	0.05%	0.01%	0.39%	0.43%	-0.04%	0.04%	0.02%	0.29%	0.33%	15.7%	19.5%	61.7%
	queen	0.02%	0.00%	0.23%	0.10%	0.29%	0.01%	0.00%	0.17%	0.07%	0.20%	22.8%	24.3%	68.4%
B	longdress	-0.02%	0.00%	-0.01%	-0.07%	0.04%	-0.13%	-0.09%	0.00%	-0.05%	0.04%	24.1%	41.1%	53.9%
C	basketball_player	0.03%	0.05%	0.03%	0.51%	-0.01%	-0.02%	0.02%	0.05%	0.38%	-0.01%	15.1%	30.8%	63.0%
	dancer_player	0.00%	-0.04%	0.00%	0.07%	-0.13%	-0.04%	-0.10%	0.01%	0.07%	-0.08%	15.0%	31.3%	61.1%
Overall average		0.01%	0.00%	0.04%	0.07%	0.10%	-0.02%	-0.03%	0.04%	0.06%	0.08%	18.0%	26.9%	62.5%

TABLE 8. Performance of the proposed algorithm compared with the V-PCC anchor under the RA configuration.

Class	Test point cloud	Geom.BD-GeomRate		Attr.BD-AttrRate			Geom.BD-TotalRate		Attr.BD-TotalRate			ΔT_T	ΔT_G	ΔT_R
		D1	D2	Luma	Cb	Cr	D1	D2	Luma	Cb	Cr			
A	loot	0.08%	0.03%	0.06%	-0.47%	0.50%	0.08%	0.01%	0.06%	-0.28%	0.13%	14.4%	20.3%	58.5%
	redandblack	0.03%	0.11%	0.24%	0.55%	0.27%	0.10%	0.20%	0.13%	0.33%	0.12%	18.1%	21.4%	60.6%
	soldier	0.29%	0.33%	0.35%	0.43%	-1.07%	0.30%	0.36%	0.31%	0.29%	-0.47%	14.1%	18.8%	54.4%
	queen	0.13%	0.13%	-0.48%	1.11%	-0.03%	0.14%	0.16%	-0.25%	0.61%	0.02%	22.0%	24.3%	63.7%
B	longdress	0.05%	0.07%	0.08%	-0.16%	0.07%	0.11%	0.12%	0.06%	-0.12%	0.05%	24.9%	47.7%	67.7%
C	basketball_player	0.08%	0.09%	-0.07%	-2.16%	-2.48%	0.06%	0.06%	0.00%	-1.21%	-1.48%	13.1%	33.1%	68.1%
	dancer_player	-0.05%	-0.05%	0.32%	0.59%	-0.36%	0.04%	0.05%	0.12%	0.17%	-0.28%	13.1%	32.0%	66.2%
Overall average		0.09%	0.10%	0.07%	-0.01%	-0.44%	0.12%	0.14%	0.06%	-0.03%	-0.27%	17.1%	28.2%	60.7%

complexity while maintaining the objective and subjective quality of the anchor.

C. ACCURACY OF DIRECT EDGE-VOXEL

The determination rate (DR) and hit rate (HR) were calculated using the TMC2 software as follows:

$$R_{\mathrm{D}}(P|A) = \frac{N(P|A)}{N(A)} \times 100$$
$$R_{\mathrm{H}}(A|P) = \frac{N(A|P)}{N(P)} \times 100, \quad (14)$$

where $R_{\mathrm{D}}(P|A)$ and $R_{\mathrm{H}}(A|P)$ denote DR and HR respectively. N(·) indicates the total number of voxels of the equivalent event. A denotes voxels included 3D points associated with the refined PPI by G-RS in TMC2 12.0 (anchor), and P represents DE-V.

As shown in Table 9, the $R_{\mathrm{H}}(A|P)$ of DE-V is from 93.38% to 97.91% and 95.97% on average. It indicates that most of the 3D points in DE-V are associated with the refined PPI by anchor G-RS. That is, the accuracy of DE-V is high to predict voxels that contain the 3D points that are associated with the refined PPI. Besides, the $R_{\mathrm{D}}(P|A)$ of DE-V is from 92.02% to 59.16%. 75.09% of voxels, on average, are DE-V among voxels include 3D points associated with the refined PPI by G-RS in TMC2 12.0.

The updated 3D points by G-RS and 3D points belongs to DE-V are compared visually. Fig. 9 shows the Class C test sequences in which the white dots represent the updated 3D points by G-RS. As seen, most updated 3D points are at the boundaries (surfaces) of the point cloud. It is due to the fact it is challenging to obtain accurate normal vectors on the surface of the point cloud [38]. Similarly, Fig.10 shows the case in which the white dots represent 3D points that belong to DE-V. 3D points in DE-V align within the surface of the point cloud. It is consistent with the updated 3D points in G-RS, as shown in Fig. 9.

TABLE 9. Hit rate (HR, R_H) and determination rate (DR, R_D) of the direct edge-voxels.

Class	Test point cloud	$R_{\mathrm{H}}(A\|P)[\%]$	$R_{\mathrm{D}}(P\|A)[\%]$
A	loot	95.69	64.23
	redandblack	93.38	62.07
	soldier	94.88	59.16
	queen	95.28	70.69
B	longdress	96.79	85.48
C	basketball_player	97.87	92.02
	dancer_player	97.91	92.01
	Average	95.97	75.09

D. PERFORMANCE COMPARISONS UNDER DIFFERENT VALUES OF SEARCH RANGE FOR IDE-V

The number of excluded voxels in the refining process is presented. The higher the number of skipped voxels, the lower the complexity of the encoder. Fig.11 depicts the specific percentage of skipped voxels at each iteration under r_β is equal to r_α. Since each test Class employs different values for

FIGURE 8. An example of the subjective quality comparison between the proposed method and V-PCC anchor. The example is the "soldier" with the frame number of 536 under r1, the RA configuration. Note that the frame number of 536 is the first frame of the "soldier" defined in the V-PCC CTC.

the voxel-dimension and the number of iterations, we summarized statistical data accordingly. From it, the number of skipped voxels increases as the iteration increases. The reason behind this is that the number of DE-V decreases through the refining process. For instance, in the first iteration, approximately 10% of the total number of voxels on average are excluded as for the Class A and Class B test point-cloud sequences, and almost 40% of voxels are skipped in the case of the Class C point cloud sequences. These numbers converged to over 90% in the last iteration for the Class B and Class C test sequences. Furthermore, the proposed edge-voxel-based fast algorithm shows better performance with a point cloud encoded with a larger voxel-size *e.g.*, Class B and Class C test sequences) than a smaller one (*e.g.*, Class A test sequences), which is consistent outcomes with the original G-RS when compared with the point-based method.

Furthermore, Fig.12 shows counterpart skipped voxels under the value of r_β is equal to one ($r_\beta = 1$). Fewer voxels were selected for the refining step when the search range for IDE-V is limited to the value one compared with the full-search range ($r_\beta = r_\alpha$). The Class A test sequences show distinct differences, in particular, at the beginning of the iterations. 10% to 20% of voxels on average were skipped with $r_\beta = r_\alpha$, whereas around 60% to

FIGURE 9. Updated 3D points in V-PCC grid-based refining segmentation. White color represents points that are associated with refined PPI. (a) Basketball player (b) Dancer player.

FIGURE 10. 3D points in direct-edge voxels in V-PCC grid-based refining segmentation. White color dots represent 3D points in direct edge-voxels. (a) Basketball player (b) Dancer player.

70% of voxels were excluded with $r_\beta = 1$ at the first iteration. Further, while about 20% of voxels went through the refining step at the last iteration under $r_\beta = r_\alpha$, only

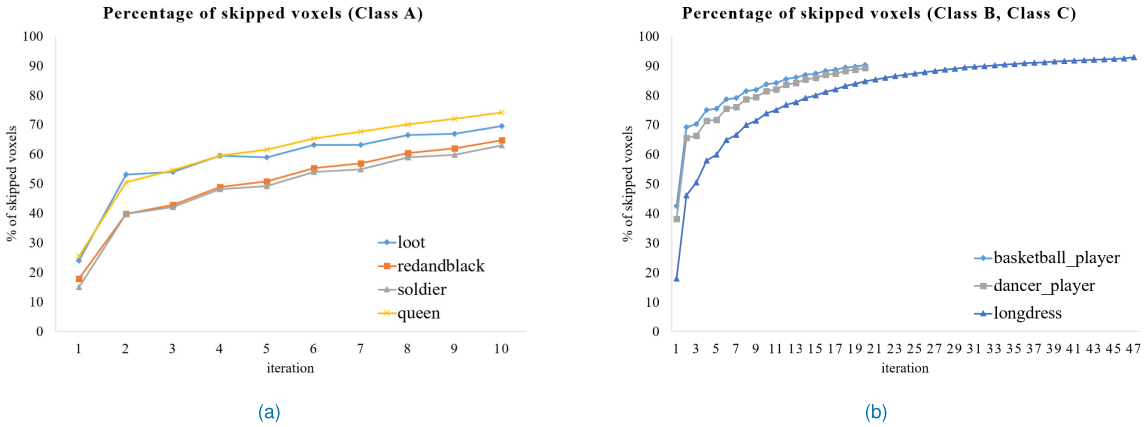

FIGURE 11. Percentage of excluded voxels by the proposed method under $r_\beta = r_\alpha$. (a) Class A sequences. (b) Class B and C sequences.

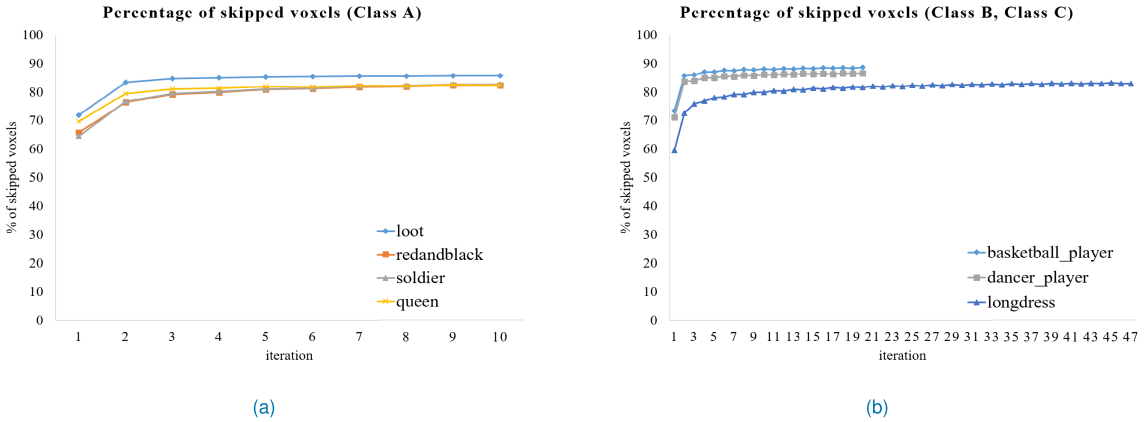

FIGURE 12. Percentage of skipped voxels by the proposed method under $r_\beta = 1$. (a) Class A sequences. (b) Class B and C sequences.

TABLE 10. Performance of the proposed algorithm compared with the V-PCC anchor under the RA configuration with r_β set to 1.

Class	Test point cloud	Geom.BD-TotalRate		Attr.BD-TotalRate			ΔT_G
		D1	D2	Luma	Cb	Cr	
A	loot	0.19%	0.19%	0.05%	-0.37%	0.41%	24.0%
	redandblack	0.14%	0.25%	0.21%	0.37%	0.13%	23.6%
	soldier	0.28%	0.31%	0.38%	0.07%	0.54%	22.4%
	queen	0.05%	0.07%	-0.16%	0.14%	0.11%	27.2%
B	longdress	0.11%	0.12%	0.06%	-0.12%	0.05%	47.7%
C	basketball_player	0.06%	0.06%	0.00%	-1.21%	-1.48%	33.1%
	dancer_player	0.04%	0.05%	0.12%	0.17%	-0.28%	32.0%
Overall average		0.13%	0.15%	0.09%	-0.13%	-0.08%	30.0%

TABLE 11. Performance of the proposed algorithm compared with the V-PCC anchor under the RA configuration with r_β set to 2.

Class	Test point cloud	Geom.BD-TotalRate		Attr.BD-TotalRate			ΔT_G
		D1	D2	Luma	Cb	Cr	
A	loot	0.08%	0.01%	0.06%	-0.28%	0.13%	20.3%
	redandblack	0.10%	0.20%	0.13%	0.33%	0.12%	21.4%
	soldier	0.30%	0.36%	0.31%	0.29%	-0.47%	18.7%
	queen	0.14%	0.16%	-0.25%	0.61%	0.02%	24.3%
B	longdress	0.07%	0.11%	0.06%	0.01%	0.12%	33.0%
C	basketball_player	0.09%	0.01%	0.10%	-0.74%	0.13%	27.2%
	dancer_player	-0.14%	-0.13%	-0.16%	-0.61%	0.37%	25.3%
Overall average		0.09%	0.10%	0.03%	-0.06%	0.06%	24.3%

10% of the total number of voxels were examined under $r_\beta = 1$.

Fig.13 visually compared the refined voxels in the soldier test point cloud at the iteration-number ten by the anchor and proposed method, respectively, in which the white dots represent points in updated voxels. As can be seen, the white dots in the original and the proposed method are notably aligned. It indicates that the proposed method predicts well the voxels that need the refining process with high accuracy.

Lastly, we analyzed the RD performance of the proposed method with different search ranges for IDE-V. In terms of complexity reduction, the self-time of G-RS was compared with the anchor under the RA configuration. Table 10, Table 11, and Table 12 show the RD comparisons of the proposed method with the anchor under r_β set to one, two, and r_α. As can be observed, the larger r_β is, the better

FIGURE 13. Updated points of the *soldier* sequence in V-PCC grid-based refine segmentation. White color represents points in refined voxels at iteration no 10. (a) anchor (b) $r_\beta = 2$.

TABLE 12. Performance of the proposed algorithm compared with the V-PCC anchor under the RA configuration with r_β set to r_α.

Class	Test point cloud	Geom.BD-TotalRate		Attr.BD-TotalRate			ΔT_G
		D1	D2	Luma	Cb	Cr	
A	loot	-0.10%	-0.11%	-0.01%	-0.71%	0.34%	11.0%
	redandblack	0.13%	0.23%	0.18%	0.37%	0.18%	13.7%
	soldier	-0.01%	0.09%	0.03%	-0.59%	-0.47%	9.8%
	queen	0.12%	0.12%	-0.03%	0.15%	-0.07%	27.3%
B	longdress	-0.04%	-0.10%	0.02%	-0.02%	0.12%	44.8%
C	basketball_player	0.15%	0.11%	0.15%	-1.26%	-0.84%	30.9%
	dancer_player	-0.04%	0.02%	-0.07%	-0.17%	-0.63%	29.7%
Overall average		0.03%	0.05%	0.04%	-0.32%	-0.20%	23.9%

RD performance the proposed method can achieve for the geometry and the attribute. Also, r_β is related to the number of skipped voxels in the refining step. It is corresponding experimental results with our analysis mentioned previously. Based on these experimental results, in this paper, we selected r_β as 1 for the Class B and Class C sequences and 2 for the Class A sequences to obtain a better RD performance balance in the encoding time reduction.

V. CONCLUSION

This paper presents a fast G-RS method in V-PCC. Voxels with the non-uniform PPI distribution are likely to be affected by G-RS. For this reason, we employ *uniformity index* of a voxel in the proposed method. The uniformity is examined within a voxel or predefined search range. The former indicates the self-uniformity of a voxel itself, and the latter means the local-uniformity that is compared with neighbors. The proposed method skips the refining steps for voxels of which PPI distribution is uniform. Experimental results show that the proposed fast algorithm can reduce on average 60.7% and 62.5% of the self-time of the refining steps of G-RS in the TMC2 12.0 reference software without RD coding loss under the RA and AI conditions, respectively.

REFERENCES

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, 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 J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[2] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi, "Video-based point-cloud-compression standard in MPEG: From evidence collection to committee draft [standards in a nutshell]," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 118–123, May 2019.

[3] S. Schwarz and D. Flynn, *Common Test Conditions for PCC*, Standard ISO/IEC JTC1/SC29/WG11 MPEG2020/N19324, Apr. 2020.

[4] S. Rhyu, J. Kim, J. Im, and K. Kim, "Contextual homogeneity-based patch decomposition method for higher point cloud compression," *IEEE Access*, vol. 8, pp. 207805–207812, 2020.

[5] *ISO/IEC FDIS 23090-5 Visual Volumetric Video-Based Coding and Video-Based Point Cloud Compression*, Standard ISO/IEC JTC1/SC29/WG11 N19579, 3D Graphics, Jul. 2020.

[6] *V-PCC Codec Description*, Standard ISO/IEC JTC1/SC29/WG7 N19332, 3D Graphics, Jun. 2020.

[7] K. Mammou, A. M. Tourapis, D. Singer, and Y. Su, *Video-Based and Hierarchical Approaches Point Cloud Compression*, Standard ISO/IEC JTC1/SC29/WG11 M41649, Oct. 2017.

[8] M. Krivokuca, P. A. Chou, and M. Koroteev, "A volumetric approach to point cloud compression—Part II: Geometry compression," *IEEE Trans. Image Process.*, vol. 29, pp. 2217–2229, 2020.

[9] P. A. Chou, M. Koroteev, and M. Krivokuća, "A volumetric approach to point cloud compression—Part I: Attribute compression," *IEEE Trans. Image Process.*, vol. 29, pp. 2203–2216, 2020.

[10] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[11] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A comprehensive study and comparison of core technologies for MPEG 3-D point cloud compression," *IEEE Trans. Broadcast.*, vol. 66, no. 3, pp. 701–717, Sep. 2020.

[12] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017.

[13] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 828–842, Apr. 2017.

[14] R. L. de Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[15] R. L. de Queiroz and P. A. Chou, "Transform coding for point clouds using a Gaussian process model," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3507–3517, Jul. 2017.

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

[17] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li, "Advanced 3D motion prediction for video-based dynamic point cloud compression," *IEEE Trans. Image Process.*, vol. 29, pp. 289–302, 2020.

[18] Y.-T. Kuo, P.-Y. Chen, and H.-C. Lin, "A spatiotemporal content-based CU size decision algorithm for HEVC," *IEEE Trans. Broadcast.*, vol. 66, no. 1, pp. 100–112, Mar. 2020.

[19] K.-H. Tai, M.-J. Chen, J.-R. Lin, R.-Y. Huang, C.-H. Yeh, C.-Y. Chen, S. D. Lin, R.-M. Weng, and C.-Y. Chang, "Acceleration for HEVC encoder by bimodal segmentation of rate-distortion cost and accurate determination of early termination and early split," *IEEE Access*, vol. 7, pp. 45259–45273, 2019.

[20] X. Liu, Y. Li, D. Liu, P. Wang, and L. T. Yang, "An adaptive CU size decision algorithm for HEVC intra prediction based on complexity classification using machine learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 1, pp. 144–155, Jan. 2019.

[21] M. Grellert, B. Zatt, S. Bampi, and L. A. da Silva Cruz, "Fast coding unit partition decision for HEVC using support vector machines," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 6, pp. 1741–1753, Jun. 2019.

[22] Y. Lu, X. Huang, H. Liu, Y. Zhou, H. Yin, and L. Shen, "Hierarchical classification for complexity reduction in HEVC inter coding," *IEEE Access*, vol. 8, pp. 41690–41704, 2020.
[23] J. Xiong, H. Gao, M. Wang, H. Li, and W. Lin, "Occupancy map guided fast video-based dynamic point cloud coding," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Mar. 2, 2021, doi: 10.1109/TCSVT.2021.3063501.
[24] E. Faramarzi and M. Budagavi, *On Complexity Reduction of TMC2 Encoder*, Standard ISO/IEC JTC1/SC29/WG11 M46201, Jan. 2019.
[25] E. Faramarzi, M. Budagavi, and R. Joshi, *Gridbased Partitioning*, Standard ISO/IEC JTC1/SC29/WG11 M47600, Mar. 2019.
[26] H. Becerra, R. Higa, P. Garcia, and V. Testoni, *V-PCC Encoder Performance Analysis*, Standard ISO/IEC JTC1/SC29/WG11 M50659, Oct. 2019.
[27] R. Higa, P. Garcia, and V. Testoni, *V-PCC Encoder Performance Optimization and Speed Up*, Standard ISO/IEC JTC1/SC29/WG11 M52200, Jan. 2020.
[28] I. Seidel, C. D. D. Freitas, R. U. Ferreira, D. C. Garcia, R. Higa, R. L. de Queiroz, and V. Testoni, *Cache-Friendly Refine Segmentation for TMC2 Speed-Up*, Standard ISO/IEC JTC1/SC29/WG7 M55143, Oct. 2020.
[29] E. Faramarzi, R. Joshi, M. Budagavi, S. Rhyu, and J. Song, *CE2.27 Report on Encoder's Speedup*, Standard ISO/IEC JTC1/SC29/WG11 M49587, Jul. 2019.
[30] H. Houshiar and A. Nuchter, "3D point cloud compression using conventional image compression for efficient data transmission," in *Proc. 25th Int. Conf. Inf., Commun. Autom. Technol. (ICAT)*, Oct. 2015, pp. 1–8.
[31] 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)*, Standard ISO/IEC JTC1/SC29/WG11 M41779, Oct. 2017.
[32] HM-16.20+SCM-8.8. (2019). *High Efficiency Video Coding Test Model*. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20+SCM-8.8/
[33] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Trans. Signal Inf. Process.*, vol. 9, p. e13, Apr. 2020.
[34] H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proc. SIGGRAPH*, 1992, pp. 71–78.
[35] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.
[36] *Point Cloud Compression Category 2 Reference Software*. [Online]. Available: http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc2.git
[37] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T SG16 Q.6 VCEG-M33, Apr. 2001.
[38] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3460–3464.

JIEON KIM received the B.Sc. and M.Sc. degrees in electronic engineering from Kwangwoon University, Seoul, South Korea, in 2009 and 2011, respectively, and the Ph.D. degree in electronic engineering from the Queen Mary University of London, U.K., in 2020, with a thesis on computational complexity optimization methods for video encoding. She is currently a Postdoctoral Researcher with the Korea Electronics Technology Institute (KETI). Her research interests include image and video processing, coding, and transmission for multi-view video content.

YONG-HWAN KIM was born in Jeju, South Korea, in 1972. He received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in image engineering from Chung-Ang University, Seoul, South Korea, in 1996, 1998, and 2008, respectively. From 1999 to 2001, he was worked with SungJin C&C, Seoul, where he optimized MPEG-1/2 Video CODEC for DVR. Since 2001, he has been with KETI, Seongnam, South Korea, where he is currently a Chief Researcher with the Intelligent Image Processing Research Center. His current research interests include V-PCC, VVC, AV1 video coding, plenoptic video coding, and its optimized implementation.