IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 January 2024, accepted 27 January 2024, date of publication 5 February 2024, date of current version 16 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3362340

RESEARCH ARTICLE

Simple Grid-Based Refinement Segmentation Algorithm for MPEG Video-Based Point Cloud Compression

QIONG JIA[1], KYUTAE KIM[1], MIN KU LEE[2], AND EUEE S. JANG[1], (Senior Member, IEEE)
[1]Department of Computer Science, Hanyang University, Seoul 04763, Republic of Korea
[2]Korea Electronics Technology Institute, Seongnam-si, Gyeonggi-do 13509, Republic of Korea

Corresponding author: Euee S. Jang (prof.euee.s.jang@gmail.com)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government through Ministry of Science and ICT (MSIT) (Development of 3D space digital media standard technology) under Grant RS-2023-00227431.

ABSTRACT In this paper, we proposed two simple refinement segmentation algorithms that can provide options to improve the computational complexity of the Video-based Point Cloud Compression (V-PCC) encoder. The patch image generation process in the encoding process is the most time-consuming and computationally intensive, accounting for about 70% of the encoder's self-running time in TMC2 v13.0. Since the real-time encoding of V-PCC is within the requirement of industry, it is highly necessary to research methods that can achieve good compression performance with low computational complexity. The grid-based refinement segmentation is one of the most computationally intensive processes in V-PCC. We found that the computational complexity can be reduced by further reducing the refinement segmentation process. Therefore, we propose to change the grid-based refinement segmentation loop process, thereby reducing the computational complexity by reducing some computational processes when the projection plane index of the neighboring grid point does not change. In the experiment, the compression performance of some sequences is improved by 0.1% to 0.9%, and the refinement segmentation time used is 79.21% and 79.53% of the anchor.

INDEX TERMS Video-based point cloud compression, MPEG, fast encoding, low complexity.

I. INTRODUCTION

Point cloud data is in the spotlight for emerging 3D immersive services, such as Virtual Reality (VR), Augmented Reality (AR), and Metaverse [1]. Most dynamic point cloud objects require a huge data storage and transmission time like uncompressed video data. Typically, the dynamic objects used in the MPEG core experiments have 700,000 to 2,900,000 3D points per frame, and each point stores 10- or 11-bit geometry precision information, and 8-bit color (RGB) information [2]. A maximum of 4.5 Gbps of bandwidth and a minimum of 378 Mbps are required to transmit such point cloud data at a frame rate of 30 fps without compression. Therefore, it is necessary to compress the point cloud data for better services in immersive applications.

As an answer to such requirements in the market, the V-PCC standard from MPEG was standardized as an international standard for point cloud compression in 2021. V-PCC adopted a video-codec based compression on point cloud data, which requires a transformation of 3D point cloud data structure into 2D maps of occupancy, geometry, and attributes (e.g., color) [3]. Therefore, V-PCC consists of two main encoding processes as shown in Fig.1: patch video generation (PVG) and video coding [3]. In the PVG process, there are patch generation and patch packing steps. Because the PVG process precedes the video coding process and cannot be performed in parallel with video coding, PVG is a critical point in the V-PCC encoding process in computational complexity.

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

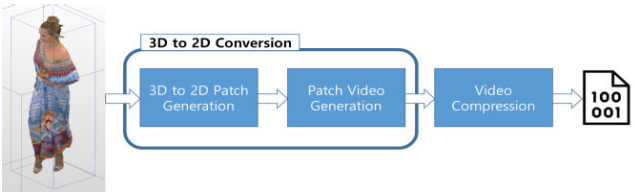

FIGURE 1. V-PCC encoding processes.

In PVG, the patch generation process is the most time-consuming one because it needs to group 3D geometry points into patches that share the similar surface normals. The surface normal computation is usually done by measuring the common surface normals of the current point and neighboring pixels. Once the normal data of points are collected, the next step is to group the points with similar normals into a single patch. The main objective of patch generation is to produce an appropriate number of patches for video compression. The number of patches directly affects the encoding time and compression efficiency. If the number of patches is small, the patch packing process will be fast, but the resulting compression efficiency may not be good, since each patch may contain non-neighboring, distant points containing more empty regions in a 2D patch. If the number of patches is large, on the other hand, the patch packing process will consume more time proportional to the number of patches. The resulting patch image may be small because the unoccupied area in each 2D patch becomes smaller [3]. In the report of the V-PCC encoder complexity utilizing TMC2 software, it took an average of six days to complete the analysis execution for a single test condition [5]. In this respect, the complexity of the V-PCC encoder is a key area for substantial improvement in the future.

In the V-PCC encoder, the patch segmentation process is the biggest cause of high complexity which accounts for about 70% of the encoder self-runtime in the test model for category 2 (TMC2) software [6]. The patch segmentation refers to converting a 3D point cloud frame on a given projection plane into 2D patches. This process is mainly composed of three procedures of initial segmentation, grid-based refine segmentation, and patch generation. In TMC2, what the patch segmentation process pursues is to generate the minimum number of patches with smooth boundaries, and when reconstructing the 3D point cloud frame with these patches, there is as little reconstruction error as possible compared to the original 3D point cloud frame [7].

In the process of segmentation refinement, the process of dividing and indexing smooth points on the point cloud surface has high complexity. In 2019, a grid-based partitioning (GBR) method was proposed to smooth the partition index of point cloud surface points [8], which reduces the complexity of the V-PCC encoder by dividing the coordinate space into refined segments. The final experimental results show that using this method can reduce the self-running time of the current version of the encoder by about 80% in V-PCC test model 5.0 (TMC2 v5) [9]. Moreover, this method has been implemented on the V-PCC test model v6.0 (TMC2 v6) [9].

Recently, a new approach has been proposed to solve the problem of high complexity of the patch segmentation process. Kim et al propose a fast grid-based refining segmentation (FGM) method which can reduce the complexity of V-PCC encoders by adaptively selecting voxels [10]. Instead of complex operations, the FGM selects a few voxels that require a refinement step based on a simple uniformity index of the initial projection plane index (PPI) distribution. This method of adaptively classifying voxels efficiently reduces the computational complexity of the V-PCC encoder in the 3D domain. Under random access (RA) and full intra (AI) configurations, the FGM can reduce the self-time of the refinement steps by an average of 60.7% and 62.5% without loss of coding efficiency [10]. At present, this method has outstanding performance in reducing the complexity of the V-PCC encoder, and it is also a relatively advanced method. However, the manner in which the voxels are selected using the uniformity index of the voxels results in that FGM may affect voxels with non-uniform PPI distribution.

In this paper, we proposed two simple grid-based refine segmentation algorithms to optimize the V-PCC encoder. The first proposed algorithm is skipping the step of calculating the PPI of a point in the loop when the PPI of the neighboring grid point does not change. This algorithm reduces the self-time of the fast grid-based refine segmentation while providing compression performance similar to the V-PCC test model 15.0 (TMC2 v15) [11]. The second proposed algorithm by stopping the grid-based refinement segmentation loop when the number of grid points whose PPI changed and the total number of points constituting a point cloud frame reaches a certain ratio. Although the self-time of the grid-based refine segmentation is slightly increased compared to the result of the fast G-RS, it improves compression performance in almost all experimental conditions.

In the following parts of the paper, we will introduce some related work in Section II. Then we showed the detailed description of our proposed method in Section III. In Section IV we will present the results and discussions of both proposed algorithms. At last, we will conclude the paper in Section V.

II. RELATED WORK

A. V-PCC PATCH GENERATION

The main goal of patch generation is to obtain the minimum number of segments while minimizing errors that will occur during reconstruction. After loading a frame of the point cloud object, the process of forming a patch is shown in Fig. 2. The figure illustrates a process of reconstructing a 3D object into a patch segment.

In this process, three procedures are the critical ones that determine the performance and computational complexity: initial segmentation, grid-based refine segmentation, and patch generation [6]. In the initial segmentation procedure,

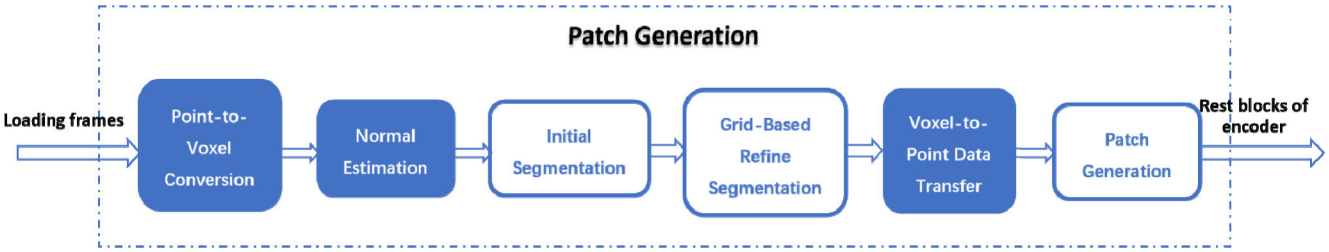

FIGURE 2. The processes of Patch Generation.

the points are clustered based on the directional proximity of their normal vectors to the normal vectors of the projection planes. Afterwards, in the grid-based refine segmentation procedure, the partitioning indices of points are smoothed out over the surface of the point cloud. Finally, in the patch generation procedure, the points are segmented into patches and all patch parameters such as width, height, occupancy map, and depth values are calculated [12]. Through this process, the 3D object is normalized and converted into an image having occupancy information, geometric information, and attribute information [3].

The patch segmentation process is a method of generating 2D patches by projecting points of each frame onto a projection plane. During the process, the 3D information of the projected points is converted into 2D information. The basic goal of this process is to generate a minimal number of patches with closely located points. In this way, the number of patches to be processed and compressed is reduced.

Fig. 3 shows the process of creating a piece by projecting points of a 3D object onto a 2D plane in the form of a patch. Point cloud objects composed of 3D coordinates are reclassified into points representing one surface to form a segment. A segment classifies points within a certain distance into one piece through normal information on a plane consisting of arbitrary points and neighboring points.

A patch segment is transformed by orthographic projection, which consists of objects composed of 3D information divided into several 2D segments. Each patch piece can usually be projected into 6 planes (i.e, XY, YZ, ZX, YX, ZY, and XZ). The projected points cluster with nearby neighboring points by measuring the distance from other points in the 3D space where the object is located. The 3D information of each point that forms a cluster exists at a point (x, y, z) in space, and when this point is projected at a right angle on an arbitrary plane, the point is projected at a point (x, y, 0) on the plane [3].

The bounding box is a boundary value calculated for each frame of the point cloud data so that the point cloud, which is 3D information, can subdivide the depth value normalized in the process of projecting into the 2D plane [3].

In this process, a number of repetitive tasks such as calculating the distance and normal line between points are performed, increasing the computational complexity. Stopping the classification of segments early can avoid computational complexity, but it will increase the number of segments. Especially the number of points in the piece increases, which is not good for compression efficiency and computational complexity [16].

B. GRID-BASED PARTITIONING

During the initial segmentation, points are clustered based on the directional proximity of their normal vectors to the normal vector of the projection plane. Then, during the refinement segmentation process, the partition index of the points is smoothed over the surface of the point cloud. Therefore, some methods have been proposed to reduce the complexity of the encoder to speed up patch segmentation in TMC2 encoder [17].

Among them, Grid-based partitioning (GBR) is a method of smoothing [8] the partition index of points on the surface of a point cloud. The GBR method significantly reduces the running time of the refinement segmentation process by dividing the coordinate space into refinement segments, thereby reducing the total runtime of the TMC2 encoder. At the same time, it also reduces memory usage and provides some gains in BDBR rates.

In the GBP method, the coordinate space is divided into a voxel grid, and filled voxels are found in the grid. The fraction of each filled voxel associated with each projection plane is then calculated by counting the number of points in the voxel that converge to that projection plane through the initial segmentation process. Next, K-D tree [18] partitioning is used to find the nearest neighboring filled voxels of each filled voxel. The values of adjacent filled voxels are then added to calculate the final value for each filled voxel. A normal score is calculated for each point associated with each projection plane. This normal score and the final score for each point associated with each projection plane are calculated as a weighted linear combination. Finally, each point is clustered to the projection plane with the highest final score. Then they repeated the above steps for several iterations.

In past techniques [19], the complexity of the nearest neighbor search routine using K-D tree partitioning increased significantly as the maximum number of neighboring points, N, increased. However, for the same value of N, N neighboring points in the GBP method may reside in M voxels, and M is always many times smaller than N. So the GBP method has lower complexity.

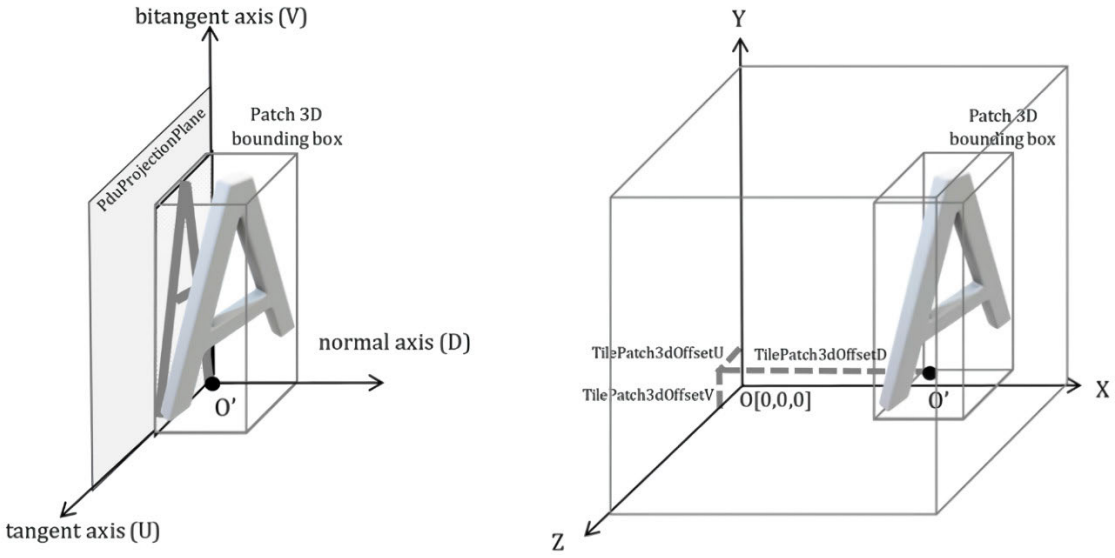

FIGURE 3. Figure 3D to 2D Patch Generation process [3].

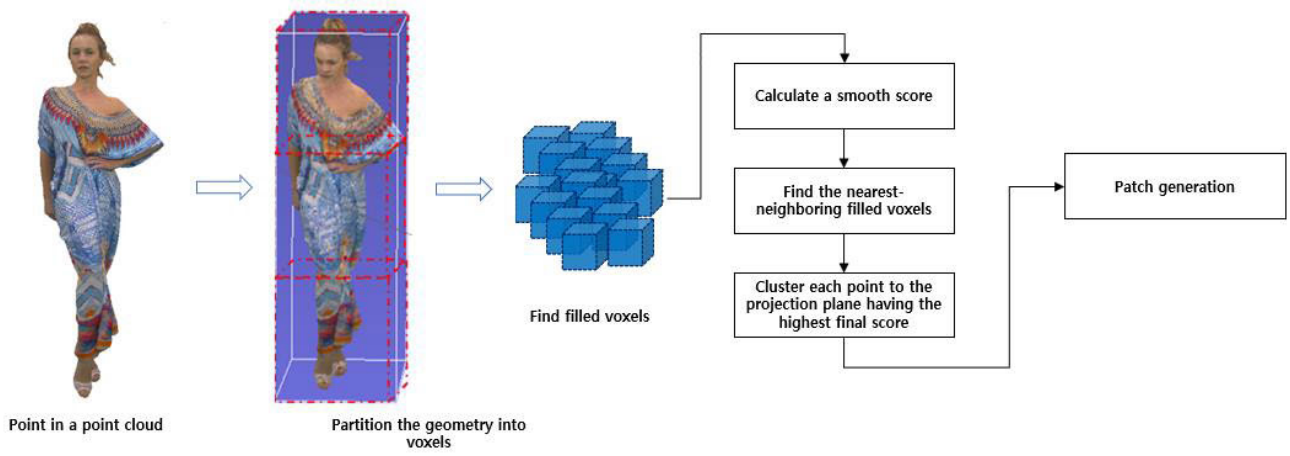

FIGURE 4. The processes of patch segmentation of GRS.

C. GRID-BASED REFINE SEGMENTATION

In order to make good patches out of the given point cloud set, all points had to be calculated to determine the nearest neighboring points for every single point. This can be a major factor in increasing the memory usage and encoding time of the encoder. One way to reduce complexity is to reduce the number of points involving the neighboring point computation. Grid-based refine segmentation [10] has been introduced to reduce the computational complexity by segmenting an object based on a filled voxel of each grid by dividing a coordinate space into a grid. A point cloud object is divided into grids having a predetermined size based on the coordinate space. In each divided grid, the 3D coordinates expressed in points are converted into voxel to determine whether they are filled or empty in each voxel space. The filled voxel is used for clustering between adjacent points by scoring ''smooth''. Among the voxels projected on the plane, those with sufficiently high scores are clustered to generate patches. Fig. 4 briefly illustrates the processing of GRS.

The coordinate space divided into voxels is searched using the K-D tree partitioning method [20]. It searches and clusters any search radius or proximity voxels of the maximum neighboring voxel in the filled voxel. The smooth score for each filled voxel related to each projection plane is calculated through the clustered voxels. The final score of the points related to the projection plane is calculated by applying a weighted linear combination to the smooth score. Finally, the G-RS is clustered on the projection plane with the highest final score calculated at each point based on the sum of the normal vectors of each 3D point and the PPI of neighboring voxels [5].

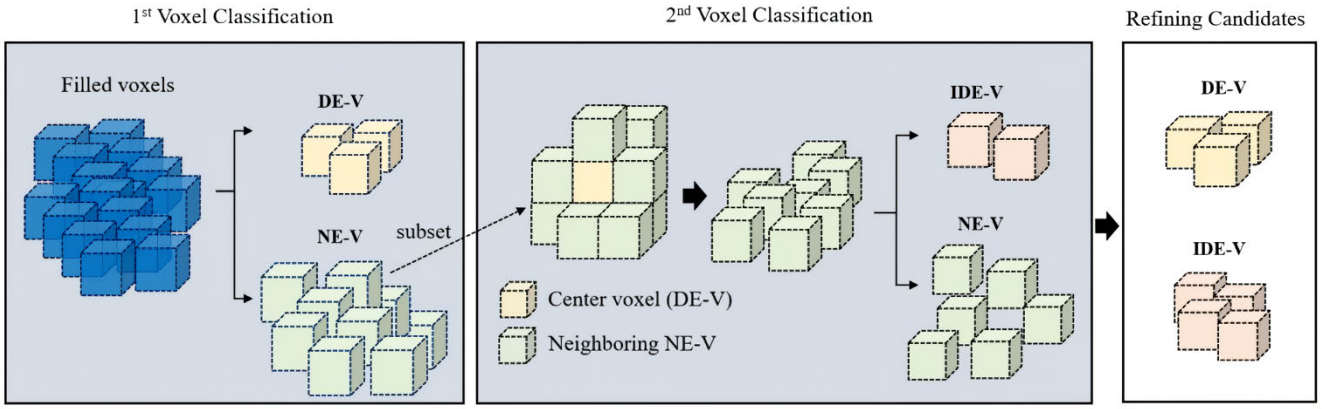

FIGURE 5. voxel classification scheme [5].

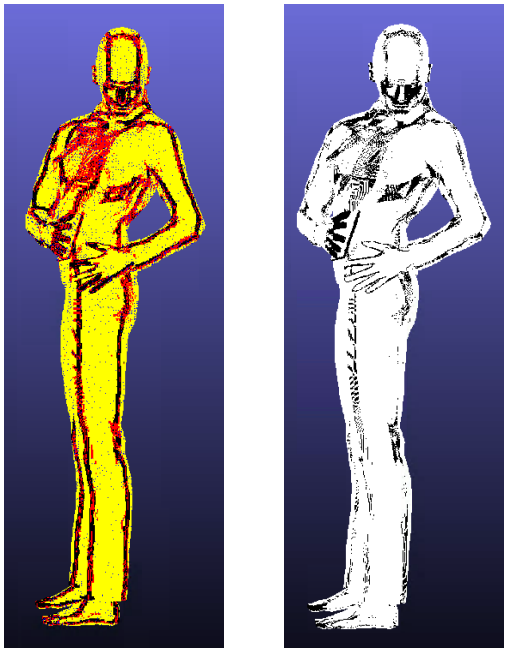

FIGURE 6. Visualization of point cloud data: (a) Classification after fast grid-based segmentation and (b) Difference before and after fast grid-based segmentation.

TABLE 1. The first frame of the loot sequence for comparison.

Iteration	$\text{Number}_{\text{Before}}$	$\text{Number}_{\text{After}}$	Reduction Ratio[%]
0	73159	73159	0.00
1	61526	58889	4.29
2	58553	55739	4.81
3	59424	54839	7.72
4	57737	53407	7.50
5	58662	52521	10.47
6	57205	52097	8.93
7	58065	50111	13.70
8	56941	46948	17.55
9	57913	45007	22.29
10	56861	42509	25.24
11	57847	38391	33.63
12	56792	35739	37.07
13	57792	32681	43.45
14	56738	31276	44.88
15	57751	29111	49.59
16	56752	26566	53.19
17	57719	24717	57.18
18	56738	23075	59.33
19	57673	20822	63.90
20	56695	18717	66.99
Total	1228543	866321	29.48

D. FAST GRID-BASED REFINE SEGMENTATION

In the patch generation process, G-RS, a grid-based voxel classification scheme, was able to significantly reduce its complexity compared to conventional methods of exploring all points. G-RS classifies non-uniform voxels by segmenting PPIs to perform a more accurate patch segmentation process. Fig. 5 shows the proposed voxel classification scheme. The voxels located at the edges are classified as direct edge voxels (DE-V), and others are classified as no edge voxels (NE-V).

In voxels not located at the edge, voxels with unequal PPI distribution are clustered as discontinuous PPIs from their neighbors and are denoted by IDE-V. Voxels with uniform PPI distribution within the nearest neighborhood to voxels not located at the edge are considered uniform regions and are denoted NE-V. The three classified types of voxels use an optional G-RS procedure that is treated differently. Among them, voxels with non-uniform PPI distributions are likely to be affected by G-RS, and voxels with uniform PPI distributions will not be affected by G-RS. Therefore, in the case of a

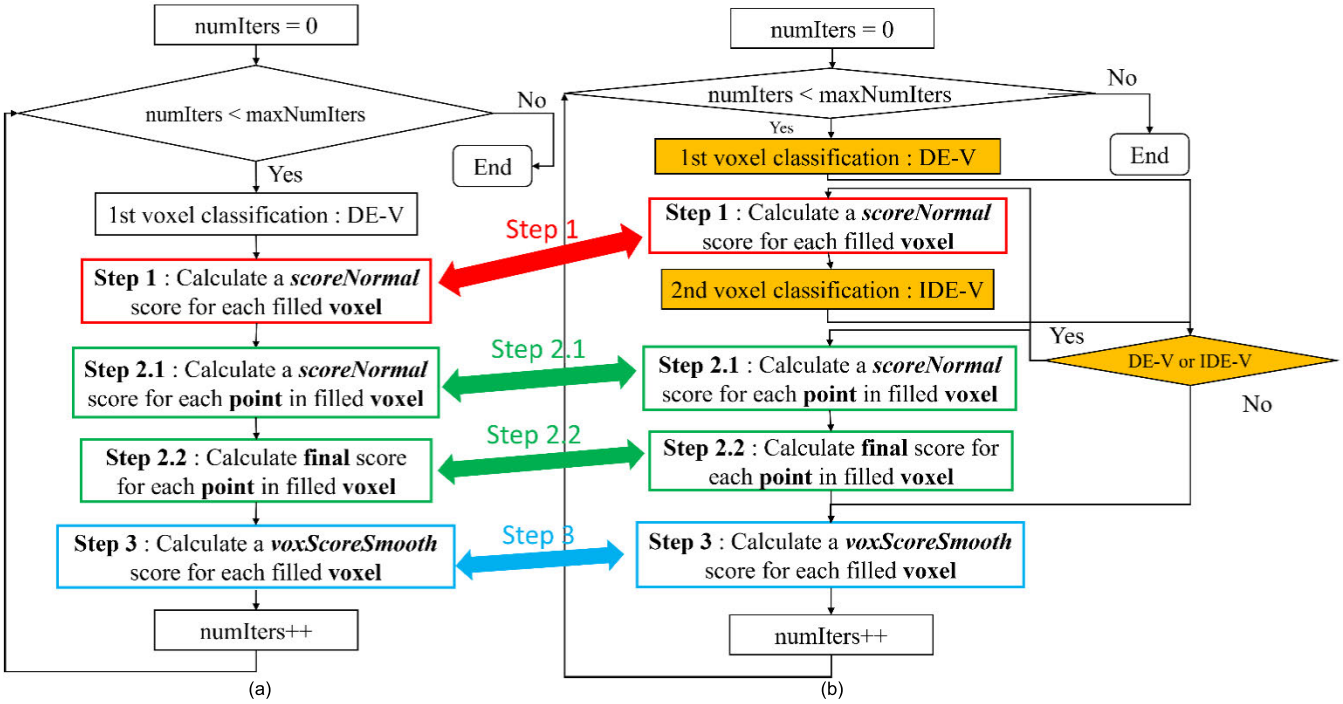

FIGURE 7. Flowchart of the existing refine segmentation procedure: (a) Grid-based refine segmentation and (b) Fast grid-based refine segmentation.

voxel having uniform PPI distribution, the FGM method may be omitted.

G-RS has been able to reduce the total runtime that time of the encoder by about 60% by adding segmentation procedures from existing methods [10]. Since the fast G-RS method (FGM) is not affected by G-RS in the case of voxels with uniform PPI distribution, the processing step for voxels with uniform PPI distribution is omitted using the uniformity index of voxels to reduce complexity in the encoding process. In the VPCC encoder, the complexity of the Lossy-AI and Lossy-RA conditions is 41% of the average saving time considering the self-time of G-RS compared to the FGM method, and 28% of the average saving time considering the self-time of V-PCC compared to FGM [15].

III. PROPOSED METHOD

A. BASIC CONCEPT

Building on the existing method, we propose our method such that the computational complexity can be reduced by further skipping the refinement segmentation process for some points. We visualized each point according to the classification of voxels after the fast grid-based refine segmentation procedure was completed as shown in Fig. 6(a). The points visualized in yellow are of the NE-V classification. The parts visualized in red and blue represent points belonging to the DE-V classification, while the parts visualized in black represent points belonging to the IDE-V classification. Furthermore, we visualized each point according to the difference in PPI before and after the fast grid-based refine segmentation, as shown in Figure 6(b). The parts visualized in white mean points where the PPI remains unchanged. While the parts visualized in black mean points where the PPI changes. We found that the part visualized in black in Fig. 6(b) is larger than the part visualized in yellow in Fig. 6(a).

We used the first frame of the loot sequence for comparison and recorded the data in Table 1. In iteration 0, the number of voxels that performed the calculations is identical before (NumberBefore) and after (NumberAfter) our proposed further classification. In the first iteration, the number of voxels that performed the calculations was reduced by 4.29%; in the 10th iteration, it was reduced by 25.24%; in the 20th iteration, it was reduced by 63.27%. So, the computational complexity can be reduced by further skipping the refinement segmentation process. After 20 iterations, the cumulative reduction ratio is 29.48%.

The G-RS [10] and the FGM algorithm [15] commonly go through steps 1 to 3 in a single loop, as shown in Fig. 7(a). G-RS differs from the fast grid-based refine segmentation in that it skips the refine segmentation procedures for the voxels of which PPI distribution is uniform NE-V.

B. ALGORITHM 1

Fig. 8 shows the modified part of the fast algorithm and the proposed simple algorithm 1 in the loop of grid-based refine segmentation. The basic concept of the fast algorithm and the proposed simple algorithm 1 is to skip the process of steps 1 to 3 under certain conditions in a single loop. Simple algorithm 1 goes through the following steps.

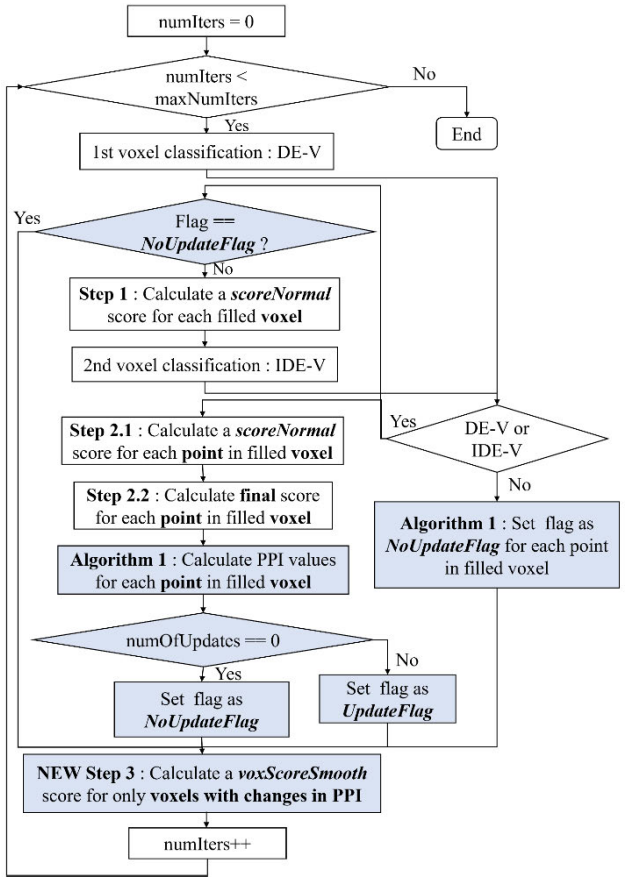

FIGURE 8. Flowchart of the Algorithm 1.

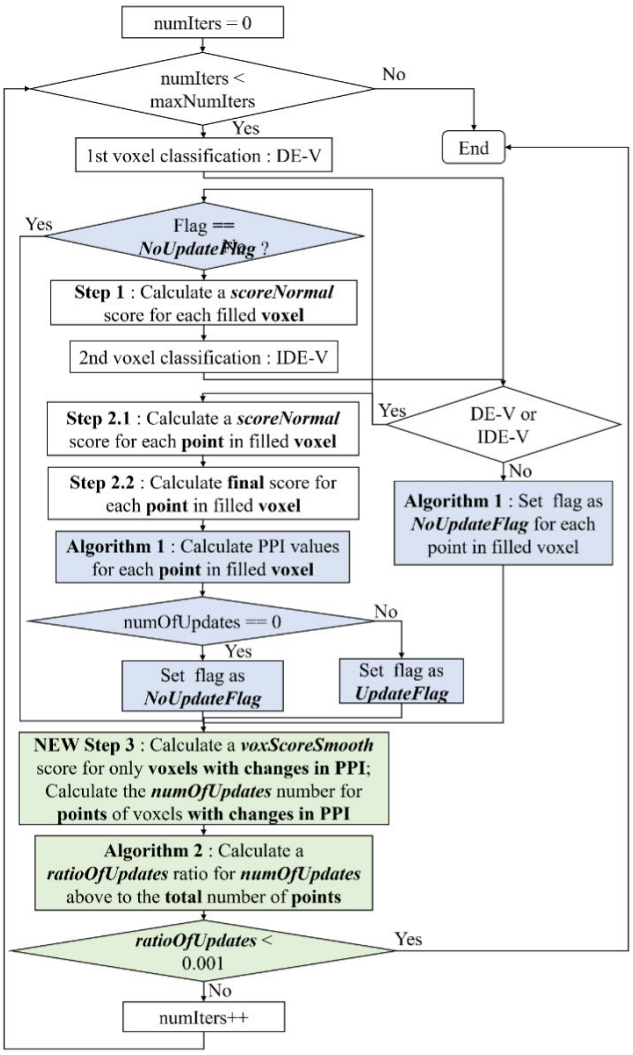

FIGURE 9. Flowchart of the Algorithm 2.

After step 2, the projection plane index values determined in the previous and current loops are compared for each point existing in a voxel. If the number of points that are different from each other is 0, the PPI of all points in the voxel is set to no change. If the project plane index of the neighboring voxels of the current voxel does not change before step 1, the process from step 1 to 3 is skipped in the loop. In step 3, voxScoreSmooth scores are not calculated for all voxels, but only voxels with changes in PPI.

C. ALGORITHM 2

As shown in Fig. 9, stopping the loop of existing grid-based refine segmentation is determined by the iterationCount value read from the configuration file. Fig. 9 shows the modified part of the proposed simple algorithm 2 in the loop of grid-based refine segmentation. Simple algorithm 2 goes through the following steps.

In step 3, the number of points of voxels with changes in PPI is obtained. It stops the loop when the ratio of the number obtained in 1 above to the total number of points constituting a point cloud frame is 0.001.

IV. RESULTS AND DISCUSSIONS

A. RESULT COMPARISON OF ONE FRAME

As shown in Fig. 10, we compared the computational complexity of grid-based segmentation of the existing and proposed methods. The x-axis of the chart represents the iteration number of the refinement segmentation, while the y-axis denotes the number of voxels that underwent calculations during steps 1 to 3 in each cycle of grid-based refinement segmentation. To compare computational complexity, we used the first frame of the loot sequence. In addition, we investigated the number of voxels that performed the calculations from steps 1 to 3 in each loop of grid-base refine segmentation. If the number of voxels that do not skip steps 1 to 3 for each loop is large, the computational complexity of grid-based segmentation is high. First, TMC2 v13 is V-PCC reference software to which fast algorithms are not applied. Second, TMC2 v15 is V-PCC reference software to which fast algorithms are applied. Third, p1 is V- PCC reference software to which fast algorithm and simple algorithm p1 are applied. Fourth, p2 is V-PCC reference software to which fast algorithm, simple algorithm p1, and simple algorithm p2 are applied. Finally, v15' is V-PCC reference software with a fast

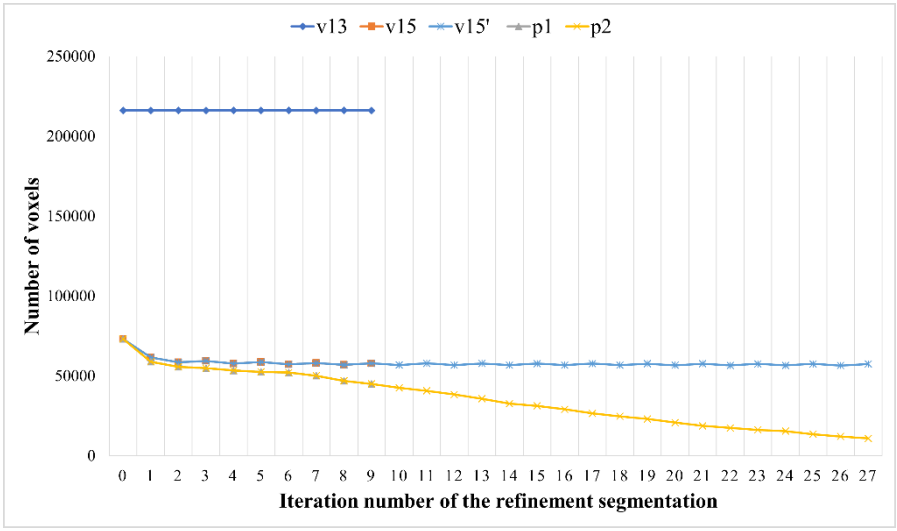

FIGURE 10. Comparison of computational complexity of grid-based refinement segmentation according to each method.

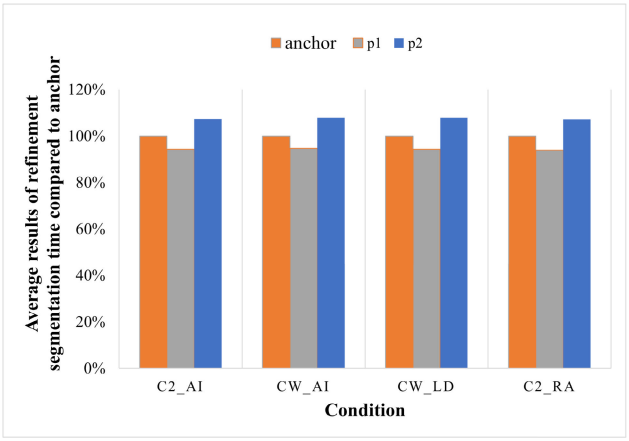

FIGURE 11. Average results of refinement segmentation time compared to anchor of 32 frames in four conditions.

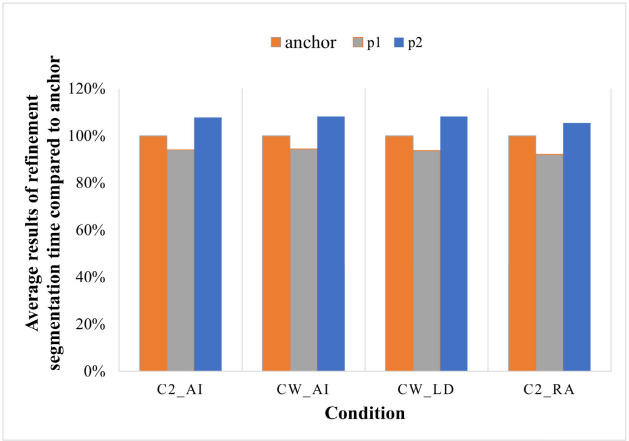

FIGURE 12. Average results of refinement segmentation time compared to anchor of all frames in four conditions.

algorithm applied like v15, but the number of loops is the same as p2.

TABLE 2. Test point cloud.

Class	Sequence	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	~2,600,000	11-bit	30	64

As shown in Fig. 10, p1 with simple algorithm 1 has lower computational complexity than v15 with fast algorithm. In addition, p2 with simple algorithm 2 has lower computational complexity than v15' without simple algorithm p2. This trend of computational complexity was also seen in the Category 2 test sequences such as redandblack, soldier, queen, longdress, basketball player, and dancer. Therefore, it can be addressed that simple algorithms p1 and p2 lower the computational complexity than the existing methods.

B. RESULT COMPARISON

We tested HEVC-based V-PCC using the proposed method and compared the coding efficiency under four conditions as follows: C2-all-intra (AI), CW-all-intra (AI), CW-low-delay (LD) and C2-inter random-access (RA) [21]. In the experiments, we used class A, B, and C sequences as test data. The sequence points, geometry precision, fps and frame number information used in the experiment are shown in Table 2 [5].

We used TMC2 version 15.0 in our Windows 10 development environment. The test PC has an i7-10700K CPU and 32 GB memory. In the development of our algorithms, we utilized Visual Studio (version 2019) as our integrated

TABLE 3. The refinement segmentation time comparison result between anchor and algorithms 1&2 of 32 frames in C2 ai.

Class	Sequence	p1	p2
A	loot	99.19%	115.54%
	redandblack	99.67%	122.67%
	soldier	100.20%	125.41%
	queen	97.65%	109.89%
B	longdress	80.04%	80.42%
C	basketball player	92.29%	98.65%
	dancer	91.44%	98.96%
Average		94.35%	107.36%

TABLE 4. The refinement segmentation time comparison result between anchor and algorithms 1&2 of 32 frames in condition CW ai.

Class	Sequence	p1	p2
A	loot	100.05%	115.01%
	redandblack	99.02%	123.30%
	soldier	99.58%	126.37%
	queen	97.99%	111.76%
B	longdress	79.69%	81.56%
C	basketball player	92.56%	98.15%
	dancer	93.94%	99.34%
Average		94.69%	107.93%

TABLE 5. The refinement segmentation time comparison result between anchor and algorithms 1&2 of 32 frames in condition CW ld.

Class	Sequence	p1	p2
A	loot	99.31%	117.06%
	redandblack	100.19%	122.16%
	soldier	99.35%	125.39%
	queen	97.68%	113.64%
B	longdress	80.80%	80.53%
C	basketball player	90.78%	98.08%
	dancer	92.35%	98.29%
Average		94.35%	107.88%

TABLE 6. The Refinement Segmentation Time Comparison Result Between Anchor and Algorithms 1&2 OF 32 FRAMES IN condition C2 ra.

Class	Sequence	p1	p2
A	loot	99.03%	116.10%
	redandblack	99.37%	122.62%
	soldier	99.96%	125.02%
	queen	97.29%	109.74%
B	longdress	80.07%	80.51%
C	basketball player	91.45%	98.71%
	dancer	90.78%	98.14%
Average		93.99%	107.26%

TABLE 7. The Refinement Segmentation Time Comparison Result Between Anchor and Algorithms 1&2 OF ALL FRAMES IN condition C2 ai.

Class	Sequence	p1	p2
A	loot	99.19%	116.10%
	redandblack	99.53%	123.62%
	soldier	99.80%	124.59%
	queen	97.62%	111.48%
B	longdress	79.21%	79.53%
C	basketball player	91.65%	99.52%
	dancer	91.67%	99.48%
Average		94.10%	107.76%

TABLE 8. The refinement segmentation time comparison result between anchor and algorithms 1&2 of all frames in condition CW ai.

Class	Sequence	p1	p2
A	loot	99.00%	116.58%
	redandblack	100.11%	124.86%
	soldier	99.98%	124.90%
	queen	97.39%	111.10%
B	longdress	79.58%	80.06%
C	basketball player	93.19%	99.29%
	dancer	91.68%	100.51%
Average		94.42%	108.19%

TABLE 9. The refinement segmentation time comparison result between anchor and algorithms 1&2 of all frames in condition CW ld.

Class	Sequence	p1	p2
A	loot	99.04%	117.52%
	redandblack	98.73%	123.52%
	soldier	100.02%	124.96%
	queen	97.02%	111.01%
B	longdress	79.36%	80.12%
C	basketball player	90.68%	98.60%
	dancer	91.39%	101.70%
Average		93.75%	108.20%

TABLE 10. The refinement segmentation time comparison result between anchor and algorithms 1&2 of all frames in condition C2 ra.

Class	Sequence	p1	p2
A	loot	84.95%	99.46%
	redandblack	99.62%	124.00%
	soldier	100.08%	124.88%
	queen	97.52%	111.34%
B	longdress	79.38%	79.58%
C	basketball player	91.49%	99.52%
	dancer	91.58%	99.47%
Average		92.09%	105.47%

development environment and C++ as our primary programming language. We chose TMC2 version 15.0, which already uses the FGM method, as the baseline. We then created two new versions of TMC2, p1 and p2, by applying Algorithm 1 and Algorithm 2, respectively. The experimental environment and other conditions were kept consistent for all three versions. Finally, we compared the refining segmentation time of the three methods for different sequences.

Tables 3 through 6 show the computational complexity evaluation with encoder refinement segmentation time between the anchor and proposed algorithms in C2 AI, CW AI, CW LD, and C2 RA conditions of 32 frames, respectively. As we anticipated, algorithm p1 shows its advantage in further complexity reduction over the anchor. Nevertheless, algorithm p2 can be utilized for further compression efficiency while increasing the computational complexity. The average results of 32 frames for all sequences of algorithm

TABLE 11. The compression performance result between anchor and algorithms 1 in condition C2 ai.

C2_ai	lossy geometry, lossy attributes [all intra]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Class A average	0.0%	0.1%	0.0%	0.1%	-0.1%	-0.1%	0.0%	0.0%	0.1%	0.0%
Class B average	0.2%	0.1%	0.0%	-0.1%	0.1%	0.2%	-0.1%	0.0%	0.0%	0.1%
Class C average	-0.3%	-0.4%	-0.1%	0.0%	-0.2%	-0.1%	-0.1%	-0.2%	-0.2%	-0.4%
Overall average	-0.1%	-0.1%	0.0%	0.0%	-0.1%	0.0%	0.0%	-0.1%	0.0%	-0.1%

TABLE 12. The compression performance result between anchor and algorithms 1 IN condition CW ai.

CW_ai	lossless geometry, lossless attributes [all intra]		
	bpip ratio [%]		
	Tot.Geom	Colour	Total
Class A average	100.0%	99.8%	99.8%
Class B average	100.0%	100.0%	100.0%
Class C average	100.0%	100.0%	100.0%
Overall average	100.0%	99.9%	99.9%

TABLE 13. The compression performance result between anchor and algorithms 1 IN condition CW ld.

CW_ld	lossless geometry, lossless attributes [inter, low delay]		
	bpip ratio [%]		
	Tot.Geom	Colour	Total
Class A average	100.0%	100.0%	100.0%
Class B average	100.0%	100.0%	100.0%
Class C average	100.0%	100.0%	100.0%
Overall average	100.0%	100.0%	100.0%

TABLE 14. The compression performance result between anchor and algorithms 1 IN condition C2 ra.

C2_ra	lossy geometry, lossy attributes [inter, random access]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Class A average	0.2%	0.2%	0.4%	-1.1%	1.1%	0.1%	0.1%	0.4%	-0.4%	0.9%
Class B average	0.5%	0.3%	-0.2%	0.0%	0.1%	0.8%	0.5%	-0.1%	0.0%	0.1%
Class C average	-0.3%	-0.4%	0.1%	-1.1%	1.5%	0.0%	-0.1%	-0.2%	-1.1%	0.7%
Overall average	0.1%	0.0%	0.3%	-0.9%	1.1%	0.2%	0.1%	0.2%	-0.5%	0.7%

TABLE 15. The compression performance result between anchor and algorithms 2 IN condition C2 ai.

C2_ai	lossy geometry, lossy attributes [all intra]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Class A average	-0.3%	-0.3%	0.3%	0.4%	0.5%	0.2%	0.3%	0.0%	0.0%	0.1%
Class B average	0.0%	-0.3%	-0.2%	-0.2%	-0.1%	0.0%	-0.4%	-0.2%	-0.2%	-0.1%
Class C average	-0.8%	-0.9%	-0.2%	-0.4%	0.2%	-0.4%	-0.4%	-0.5%	-0.7%	-0.3%
Overall average	-0.4%	-0.5%	0.1%	0.1%	0.3%	0.0%	0.0%	-0.2%	-0.2%	0.0%

p1 and algorithm p2 compared with the anchor under four conditions are shown in Fig. 11.

Tables 7-10 show the computational complexity of the encoder refinement segmentation time between the anchor and proposed algorithms in C2 AI, CW AI, CW LD, and C2 RA conditions for all frames. As expected, algorithm p1 shows its advantage in further complexity reduction over the anchor. However, algorithm p2 can be used to further improve compression efficiency, albeit at the cost of increased computational complexity. The average results of all frames for all sequences of algorithm p1 and algorithm p2 compared with the anchor under four conditions are shown in Fig. 12.

TABLE 16. The compression performance result between anchor and algorithms 2 IN condition CW ai.

CW_ai	lossless geometry, lossless attributes [all intra] bpip ratio [%]		
	Tot.Geom	Colour	Total
Class A average	99.3%	99.6%	99.5%
Class B average	100.0%	100.0%	100.0%
Class C average	99.5%	99.9%	99.9%
Overall average	99.5%	99.8%	99.8%

TABLE 17. The compression performance result between anchor and algorithms 2 IN condition CW ld.

CW_ld	lossless geometry, lossless attributes [inter, low delay] bpip ratio [%]		
	Tot.Geom	Colour	Total
Class A average	99.6%	100.1%	100.1%
Class B average	100.0%	100.0%	100.0%
Class C average	99.5%	99.9%	99.9%
Overall average	99.6%	100.0%	100.0%

TABLE 18. The compression performance result between anchor and algorithms 2 IN condition C2 ra.

C2_ra	lossy geometry, lossy attributes [inter, random access]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Class A average	2.9%	2.7%	4.2%	1.5%	4.8%	3.4%	3.1%	3.2%	1.6%	3.5%
Class B average	0.3%	0.1%	0.0%	0.2%	0.2%	0.5%	0.2%	0.0%	0.1%	0.2%
Class C average	-0.4%	-0.5%	1.1%	-1.5%	1.7%	0.2%	0.1%	0.0%	-1.5%	0.5%
Overall average	1.6%	1.4%	2.7%	0.5%	3.3%	2.0%	1.8%	1.9%	0.5%	2.2%

Tables 11 - 14 show the compression performance between the anchor and proposed algorithm p1 in C2 AI, CW AI, CW LD, and C2 RA conditions, respectively. And Tables 15 through 18 show the compression performance between the anchor and proposed algorithm p2 in C2 AI, CW AI, CW LD, and C2 RA conditions, respectively.

V. CONCLUSION

Overall, our experimental results show that our introduced algorithms, p1 and p2, have exhibited significant reductions in computational complexity and improvements in compression performance within the V-PCC encoder. In our experiments, these algorithms achieved reductions in refinement segmentation time by 79.21% and 79.53%, respectively, relative to the existing grid-based fast refinement segmentation method.

Furthermore, we observed enhanced compression performance ranging from 0.1% to 0.9%, particularly under the C2 AI condition. These findings underscore the substantial potential of our research in simplifying the V-PCC encoder while simultaneously enhancing its performance. However, to ensure result accuracy and consistency, further research and validation are needed to fully harness the potential of these algorithms and advance the efficiency of V-PCC.

REFERENCES

[1] *Common Test Conditions for Point Cloud Compression*, ISO/IEC JTC1/SC29/WG11 Standard N1866, MPEG 3DG, Gothenburg, Jul. 2019.

[2] 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, no. 1, p. e13, 2020.

[3] 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.

[4] *Common Test Conditions for Point Cloud Compression*, ISO/IEC JTC1/SC29/WG11 Standard N18665, MPEG 3DG, Jul. 2019.

[5] H. Becerra, R. Higa, P. Garcia, and V. Testoni, *V-PCC Encoder Performance Analysis*, ISO/IEC JTC1/SC29/WG7 Standard m50659, Geneva, Oct. 2019.

[6] E. Faramarzi, M. Budagavi, and R. Joshi, *[V-PCC] [New Proposal] TMC2 Encoder Speedup Using Grid-based Segmentation*, ISO/IEC JTC1/SC29/WG7 Standard m56857, Apr. 2021.

[7] E. Faramarzi, R. Joshi, and M. Budagavi, "Mesh coding extensions to MPEG-I V-PCC," in *Proc. IEEE 22nd Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2020, pp. 1–5.

[8] *[V-PCC] CE2.27 Report on Encoder's Speedup*, ISO/IEC JTC1/SC29/WG7 Standard m49587, Gothenburg, Sweden, 2019.

[9] (2019). *Point Cloud Compression Category 2 Reference Software TMC2*. [Online]. Available: https://git.mpeg.expert/MPEG/3dgh/v-pcc/software/mpeg-pcc-tmc2

[10] Y. Kim and Y.-H. Kim, "Low complexity fast grid-based refining segmentation in the V-PCC encoder," in *Proc. IEEE Int. Conf. Consum. Electronics-Asia (ICCE-Asia)*, Oct. 2022, pp. 1–4.

[11] *PCC Test Model V15*, ISO/IEC JTC1/SC29/WG7 Standard N00147, Jul. 2021.

[12] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of sparse and dense dynamic point clouds—methods and standards," *Proc. IEEE*, vol. 109, no. 9, pp. 1537–1558, Sep. 2021.
[13] (2021). *Point Cloud Compression Category 2 Reference Software TMC2-15.0*. [Online]. Available: https://git.mpeg.expert/MPEG/3dgh/v-pcc/software/m2021peg-pcc-tmc2
[14] *V-PCC Codec Description*, ISO/IEC JTC1/SC29/WG7 Standard N00012, 2020.
[15] J. Kim and Y. H. Kim, "Fast grid-based refining segmentation method in video-based point cloud compression," *IEEE Access*, vol. 9, pp. 80088–80099, 2021.
[16] 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.
[17] *V-PCC Core Experiment 2.27 on Encoder Speedup*, ISO/IEC JTC1/SC29/WG7 Standard N18509, 2019.
[18] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[19] S. Gu, J. Hou, H. Zeng, H. Yuan, and K.-K. Ma, "3D point cloud attribute compression using geometry-guided sparse representation," *IEEE Trans. Image Process.*, vol. 29, pp. 796–808, 2020.
[20] *Grid-based Partitioning*, ISO/IEC JTC1/SC29/WG11 Standard m47600, Mar. 2019.
[21] *Common Test Conditions for Essential Video Coding*, Standard ISO/IEC JTC 1/SC 29/WG, Brussels, Belgium, Jan. 2020.

QIONG JIA received the B.S. degree in computer science and technology from Dalian Maritime University, Dalian, Liaoning, China, in 2018, and the M.S. degree in computer science from Hanyang University, Seoul, South Korea, in 2021, where she is currently pursuing the Ph.D. degree with the Department of Computer Science. Her research interests include 2D video compression and 3D video compression.

KYUTAE KIM was born in Anseong-si, Gyeonggi-do, South Korea, in 1996. He received the B.S. degree in computer engineering from Gangnam University, South Korea, in 2020. He is currently pursuing the M.S. degree with the Department of Computer Science, Hanyang University. His research interests include point cloud compression and multimedia-related compression technology and application.

MIN KU LEE received the B.S. and M.S. degrees from the Department of Information Telecommunication Engineering, Soongsil University, Seoul, South Korea, in 2000 and 2002, respectively, and the Ph.D. degree from the Department of Computer Science, Hanyang University, Seoul, in 2022. From 2002 to 2017, he worked on multimedia compression (H.263, H.264, WMA, AC3, AMR-NB, and AMR-WB), multimedia transmission (H.222, UDP, RTP, and MTP), multimedia security (WMDRM), and multimedia systems (DMB, android smartphones, IPTV, and MRF). He is currently a Postdoctoral Researcher with the Korea Electronics Technology Institute (KETI). His research interests include video security, 2D video coding, 3D graphics coding, and point cloud compression.

EUEE S. JANG (Senior Member, IEEE) received the B.S. degree from Jeonbuk National University, South Korea, and the Ph.D. degree from SUNY University at Buffalo, Buffalo, NY, USA.

He is currently a Professor with the Department of Computer Science and Engineering, Hanyang University, Seoul, South Korea. He has authored more than 150 articles on MPEG standardization, more than 30 journal articles and conference papers, 35 pending or accepted patents, and two book chapters. His research interests include image/video coding, reconfigurable video coding, and computer graphics objects.

Dr. Jang received three ISO/IEC Certificates of Appreciation for contributions to MPEG-4 development. He received the Presidential Award from the Korean Government for his contribution to MPEG standardization.