IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 27, 2021, accepted October 4, 2021, date of publication October 8, 2021, date of current version October 18, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3118806

Efficient Quadtree Search for HEVC Coding Units for V-PCC

TING-LAN LIN[1], (Member, IEEE), HONG-BIN BU[1], YAN-CHENG CHEN[1], JUN-RUI YANG[1], CHI-FU LIANG[1], KUN-HU JIANG[1], CHING-HSUAN LIN[2], AND XIAO-FENG YUE[1]

[1]Department of Electronic Engineering, National Taipei University of Technology, Taipei 10608, Taiwan
[2]Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan 32023, Taiwan

Corresponding author: Ting-Lan Lin (tinglan@ntut.edu.tw)

This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 106-2221-E-033-010, Grant 107-2221-E-027-125, Grant 108-2221-E-027-024, Grant 109-2221-E-027-085, and Grant 110-2221-E-027-044-MY3.

ABSTRACT Dynamic point clouds (DPC) are new media storage formats that allow end-users to watch objects/scenes in a three-dimensional (3D) sense. It can be displayed from different angles throughout time. However, the raw size of a point cloud is huge because there can be millions of points (each containing color triplet and location triplet information) in a point cloud, and there can be multiple point clouds in a DPC. Video-based point cloud compression (V-PCC) is developed to project a 3D point cloud to 2D images: attribute, geometry, and occupancy images. After padding, the 2D images are compressed using the well-established high-efficiency video coding (HEVC). In this study, we first employ an occupancy image to propose a blocky occupancy flag (BOF), to denote the occupancy information on ''a block basis''. For coding attribute and geometry images, we use a BOF to develop a fast coding unit (CU) algorithm for early termination of the CU search recursion. We also utilize the geometry images to calculate the 2D and 3D information of each pixel, for 2D/3D spatial homogeneity of the pixels to design fast CU decision. In addition, we proposed a modified rate-distortion optimization for different color components considering the picture order count (POC) structure in HEVC/V-PCC. Finally, we propose an HEVC input pixel modification method based on a BOF to reduce the unnecessary information to be coded for attribute images. Compared with the state-of-the-art fast V-PCC encoding method, the proposed work outperforms by up to 2.31% in Bjøntegaard delta bit rates (BDBR) (with very slight loss by only up to 0.38%), and improves the time saving performances by up to 7.84% for two different testing datasets.

INDEX TERMS Video-based point cloud compression (V-PCC), dynamic point cloud (DPC), high efficiency video coding (HEVC), fast coding unit (CU) decision algorithm, occupancy map.

I. INTRODUCTION

The three-dimensional (3D) point cloud is one of the most important formats for recording objects and scenes. It is commonly used in autonomous driving applications and cultural heritage reservations. The size of a point cloud is huge because each point has location information $\{x_i, y_i, z_i\}$ and color information $\{R_i, G_i, B_i\}$. A point cloud can comprise a million points; the number is multiplied when considering dynamic point cloud (DPC) with multiple time instances. Therefore, an efficient compression technique for point clouds or DPC is required. There are surveys [1]–[3] conducted regarding the development of the compression on point clouds, comprising two categories: Geometry-based point cloud compression (G-PCC) and Video-based point cloud compression (V-PCC). In this paper, we focus on V-PCC.

The compression concept of V-PCC [4] is to use six different projection angles to project a 3D point cloud to produce several 2D patches (patch generation). Attribute image, geometry image and occupancy image are formed after patch packing. Then, they are padded and encoded via well-established video coder high-efficiency video coding (HEVC) [5]. The attribute and geometry images are lossy-compressed, and the occupancy image is losslessly compressed. The procedures are reversed to reconstruct the 3D point cloud for decoding [6].

In recent studies, rate control between geometry and attribute images are discussed in [7]. The prediction units corresponding to 0-occupancy pixels are assigned zero bits.

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

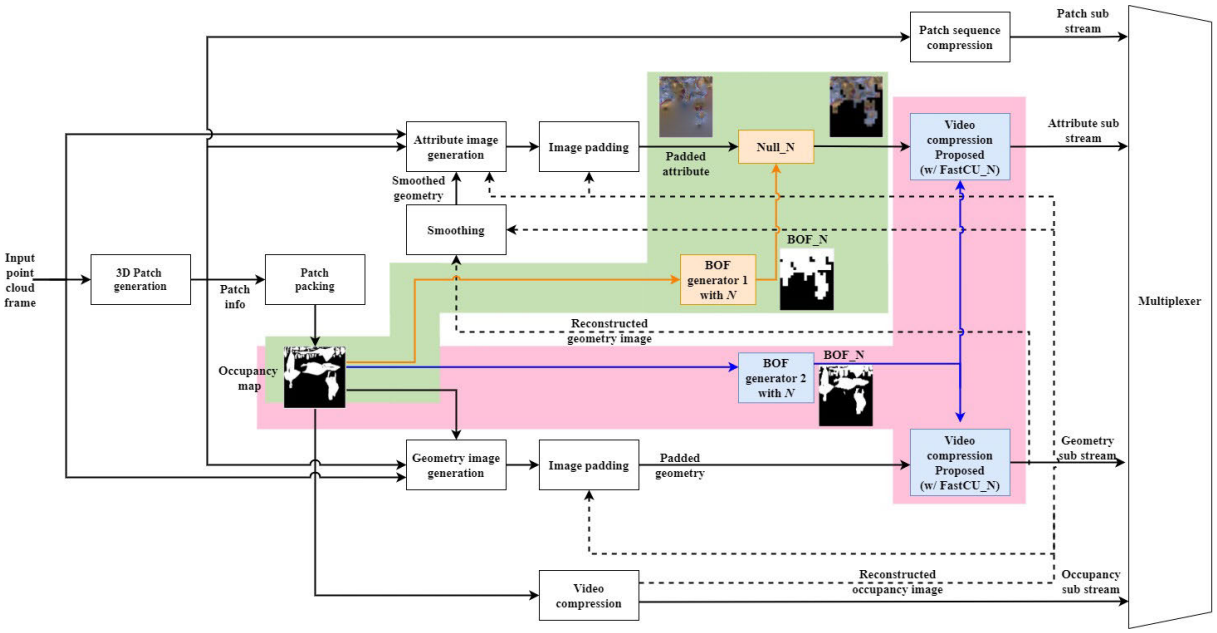


FIGURE 1. The V-PCC system [4] modified by the proposed algorithms Null_N (with green background) and proposed fast coding methods including FastCU_N (with pink background), with blocky occupancy flags with possibly different *N* (BOF_N).

Authors [8] proposed a 3D to 2D motion model and a geometry-based motion prediction for 2D attribute video. The found motion vector is used as an additional motion candidate, or as the center of motion estimation. Another study [9] used an occupancy image to change the weighting in the rate-distortion (RD) optimization; the distortions of the unoccupied pixels are not emphasized. The coding mechanisms are proposed for occupied and unoccupied partitions. A padding algorithm has been proposed using the occupancy image information [10]. The groups that should be occupied are padded with real point cloud pixels considering the smoothness of the block. However, the groups that should not be occupied are padded with the residue to minimize the bit cost. The above studies used occupancy information to improve the RD performance.

As discussed, the core coding module in V-PCC is HEVC, which is very efficient in terms of rate-distortion performance, but it is very time-consuming. Many existing works had discussed the fast coding methods for HEVC. For example, works on HEVC fast coding [11], [12] use convolutional neural network for INTRA prediction. **In the proposed work, we focus to exploit the *information from/ related to the V-PCC overall system* that can help reduce the HEVC coding complexity while still maintaining or even slightly increasing the rate-distortion performance; *pure standalone HEVC fast methods* such as [11], [12] are not considered and not compared in our context, to isolate the research directions and results.** The most relevant existing work for fast HEVC method in V-PCC is the state-of-the-art work [13], which utilizes different conditions of occupancy map to design a fast CU decision method and a fast mode decision method; this work [13] will be compared with our work in the experimental section.

In our work, we develop a fast coding system with HEVC for V-PCC, as shown in Fig. 1. **Compared with [13], the novelties of the proposed method are as follows:**

1) **The occupancy image in V-PCC is used to generate a blocky occupancy flag (BOF_N), indicating the occupied pixels "on a block basis," to be aligned with the subsequent block-based encoder HEVC.**
2) **The BOF_N is used to develop a fast CU algorithm FastCU_N, which terminates the recursive CU partition early when the collocated block in the BOF_N are all 0-valued pixels (unoccupied CU).**
3) **The 3D spatial information and the 2D spatial information of each block are used to further evaluate the CU content homogeneity for early termination in a fully-occupied CU or a partially-occupied CU, respectively.**
4) **A modified rate-distortion optimization is proposed with optimal re-weighting for different color domains, and the consideration of picture order count (POC) hierarchy in V-PCC.**
5) **A pixel modification method Null_N for the HEVC input is also proposed to zero out the block corresponding to 0-valued pixels in a BOF_N, to prevent the wastage of the resources (coding time and bitrates) in the HEVC.**
6) **Compared against the most relevant state-of-the-art work [13], the proposed system can reduce the coding time by up to 7.84%, with the improvement in BDBR by up to 2.31% (and very slight loss by only up to 0.38%) under two datasets.**

The remainder of the paper is organized as follows: in Section 2, the BOF based on the occupancy image is designed. A fast CU algorithm in HEVC for the V-PCC

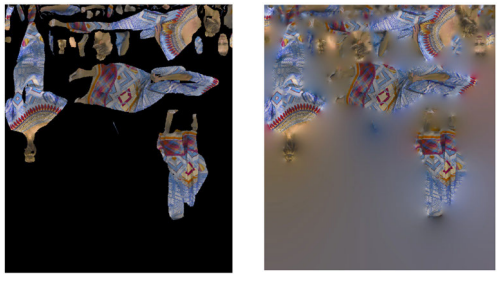
standard is proposed based on the BOF in Section 3. Section 4 introduces how 3D and 2D spatial information are used for fast CU method. A modified rate-distortion optimization is designed in Section 5. In Section 6, the BOF is used to design a pixel modification algorithm for the HEVC input. Section 7 presents the experimental results. Finally, Section 8 concludes the paper.

II. MOTIVATION AND GENERATION OF BLOCKY OCCUPANCY FLAG

In the coding procedure of V-PCC, a point cloud is first projected to six planes in a cube. The projections are *patches* [4] to be allocated in a 2D image, called *attribute image* as in Fig. 2 (b). The corresponding depth information is a *geometric image* [6] as in Fig. 2 (a), and the corresponding *occupancy image* is shown in Fig. 2 (c).

FIGURE 2. (a) Geometry image, (b) attribute image, and (c) occupancy image from left to right.

Then, the unoccupied locations in the attribute and geometry images are ***padded*** to reduce the high-frequency part of the image for later video coding. Figure 3 shows an example of the attribute image. There are push–pull [4] and Harmonic background fillings [4] for padding in TMC2 [5]. The padded image is then input into the HEVC encoder.



(a) (b)

FIGURE 3. (a) Left: unpadded and(b) right: padded attribute images.

Based on the point cloud reconstruction procedure, we understand that *only* 1-occupied pixels are used for reconstruction. Because the video coder (such as HEVC) is block-based, only the block *with* 1-occupied pixels are meaningful; the block full of unoccupied (padded) pixels does not contribute to the point cloud reconstruction. *Therefore, one motivation of the proposed algorithm is to identify suitable unoccupied blocks and design different mechanisms in the V-PCC system for coding performance improvement.*

We first perform square $N \times N$ partitions on the original occupancy image, as shown in the top-left portion in Fig. 4; $N = 64$ for example. For each $N \times N$ block, if there is at least one 1-occupied pixel in the block, pixels at the same block location in another *flag* image are all set to be 1, and 0 otherwise. This *flag* image is defined as **BOF_N**, indicating unoccupied blocks. Figure 5 shows the BOF with different Ns. Notably, the 0-pixel-blocks in BOF_i are also 0s in BOF_j when $j < i$. **The BOF_N of $N \in \{64, 32, 16, 8\}$ show the coarse-to-fine indication of occupancy on a block basis.**

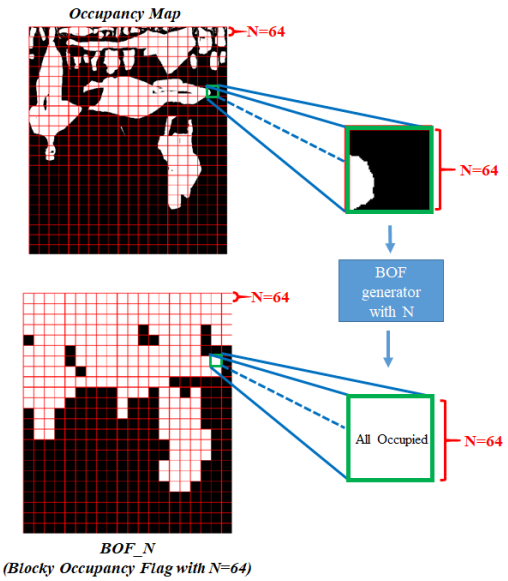


FIGURE 4. Generation of blocky occupancy flag with $N = 64$.

The BOF information is used to improve the coding performance in the following Section 3 and Section 6. **Notably, $N \in \{64, 32, 16, 8\}$ for different coarse-to-fine blocky effects is to *match and align* with the possible CU blocks in the subsequent 2D video encoder. Moreover, the BOF is only for the reference of our algorithm in the encoder; the BOFs are neither to replace/modify the original occupancy image nor to be saved in the bitstream.**

III. FAST CU ENCODING ALGORITHM IN HEVC USING BOF FOR UNOCCUPIED CU

In V-PCC, the padded images are coded using HEVC. The operation of HEVC determining the optimal CU partition is computationally complex because of the quadtree search. Based on the discussion, the unoccupied CU blocks (indicated by the BOF) do not contribute to point cloud reconstruction. Thus, they do not need to be coded with good quality; this is where we can save coding complexity.

The CU exhaustive search in HEVC starts by performing a prediction (INTER/INTRA) for the current 64 × 64 block (LCU, largest CU) at depth 0. Smaller partitions are recursively tested for prediction to the size as small as 8 × 8 at depth 3 (Fig. 6). The best combination of partitions is that with optimal RD cost.

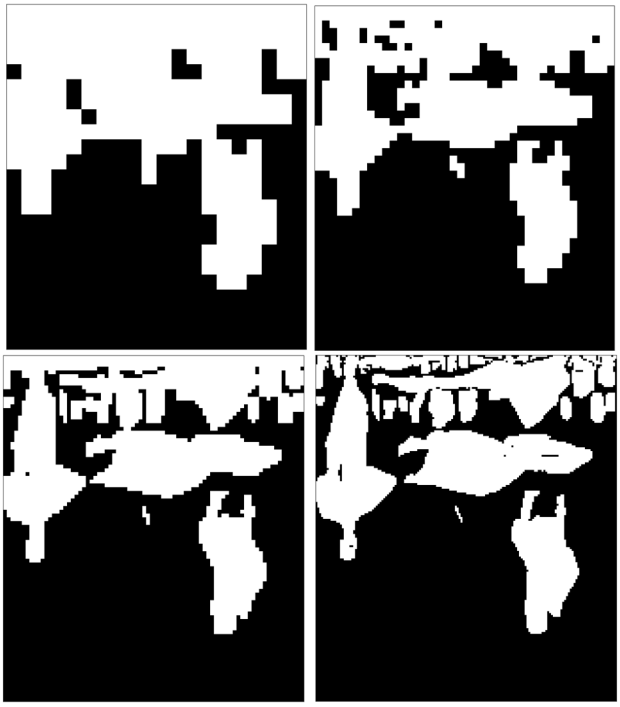


FIGURE 5. (a) BOF_64, (b) BOF_32, (c) BOF_16 and (d) BOF_8, in raster order.

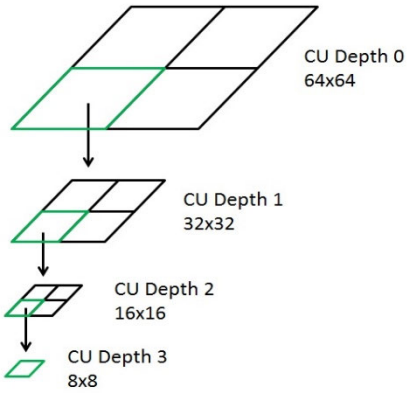


FIGURE 6. Exhaustive CU search at different depths in HEVC.

A deeper CU search aims for better and detailed motion matches to improve the encoded image quality with extra search time. **Since the unoccupied CU blocks (indicated by the BOF) are not used in the point cloud reconstruction, they are allowed to have lower coded-pixel quality;** the complexity to go deeper for those blocks is a wastage.

For example, **Fig. 7(a)** shows the partition result for exhaustive CU search. For three unoccupied 64 × 64 blocks, the original exhaustive CU search not only performs full searches for them, causing unnecessary complexity cost, but also results in depth 1 (32 × 32), introducing unnecessary bitrate costs. Based on just looking at the occupancy map, those 64 × 64 blocks could stop very early at depth 0.

Therefore, we use the proposed **BOF_N** as a reference for the proposed CU partition algorithm to work. *The idea is that*

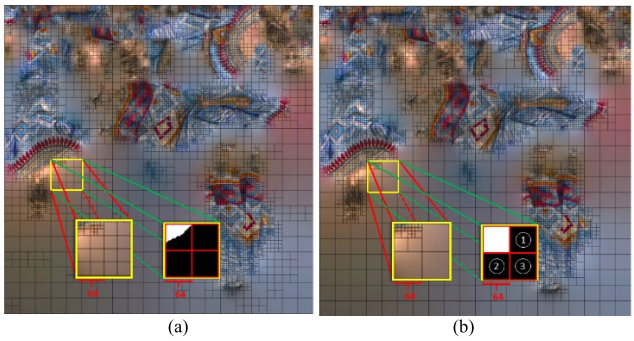


FIGURE 7. CU partition results of (a) original exhaustive CU search and (b) FastCU_64. The windows with black-and-white pixels in (a) and (b) are the original occupancy image and the BOF_64 of the same spatial location, respectively. Yellow windows (target area) are of the same spatial locations. The blocks with circled numbers in (b) are desired effects of the proposed work.

*for the unoccupied blocks in **BOF_N**, their collocated blocks in the attribute (or geometry) image are coded at CU depth as small (shallow) as possible, achieving an early termination.* Note that we use the proposed **BOF_N** as opposed to the original occupancy image for the CU decision algorithm because the **BOF_N** is block-based to be better aligned with the block-based video coder HEVC.

For the effects of different *N* s for BOF_N, if we use **BOF_64** (Fig. 5(a)) as a reference, the unoccupied 64 × 64 blocks (in attribute and geometry images) can be early-terminated at CU depth 0 without going deeper. If **BOF_32** (Fig. 5(b)) is used, which produces additional 32 × 32 unoccupied blocks, those blocks can be early-terminated at CU depth 1. If **BOF_16** (Fig. 5(c)) is used, additional unoccupied 16 × 16 blocks exist and are early-terminated at CU depth 2. 8 × 8 is the smallest block size for CU and no need for early termination. The optimization problem of the proposed fast CU algorithm is to achieve the desired result **: For the (aligned) unoccupied blocks, the depths of early-terminated (attribute/geometry) blocks should be as small (shallow) as possible to save the maximal amount of time.**

The desired result is shown in Table 1. The algorithm using a particular **BOF_N** should have 64 × 64 unoccupied blocks early-terminated, followed by 32 × 32 unoccupied blocks being early-terminated; the procedure continues along the priorities in Table 1 to N × N unoccupied blocks. Therefore, if **BOF_N** with smaller *N* is used, the resulting algorithm can determine extra smaller unoccupied blocks to early terminate for more time saving.

To achieve the desired result in Table 1, we propose the following fast CU search algorithm procedure. For a given reference BOF_N, when the CU search algorithm is done performing INTRA/INTER prediction at a specific depth of Q × Q:

1) **If the Q × Q pixel values in the same spatial locations in the BOF_N are *all 0s*, deeper depths are not required. Thus, we do not need to use extra efforts**

TABLE 1. Priority of the early termination category of the unoccupied blocks.

Priority	Early termination categories	BOF_64	BOF_32	BOF_16
1	Aligned unoccupied *64×64* blocks are terminated at depth *0*	✓	✓	✓
2	Aligned unoccupied *32×32* blocks are terminated at depth *1*		✓	✓
3	Aligned unoccupied *16×16* blocks are terminated at depth *2*			✓

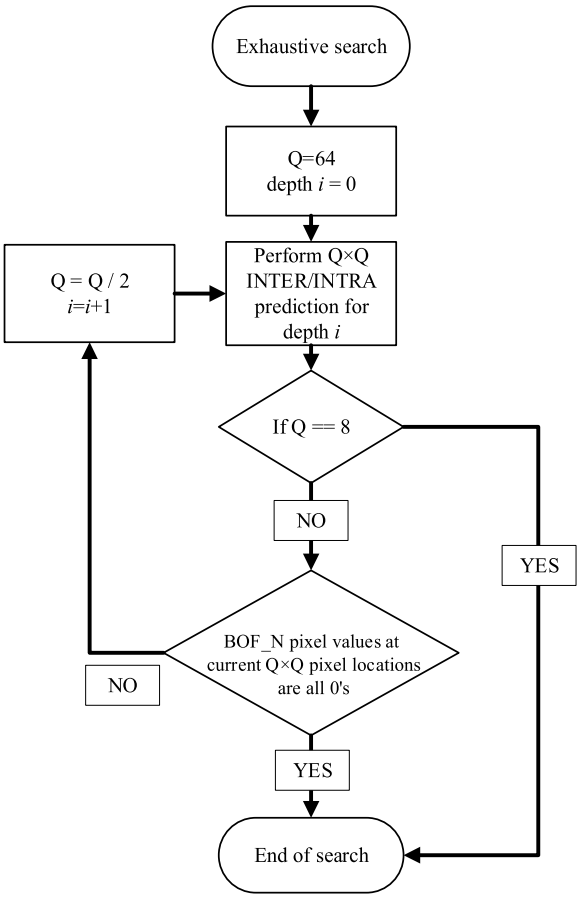


FIGURE 8. The flow diagram of proposed FastCU_N.

(time complexity) for all-0-pixel blocks in BOF to encode its detail because this block is not used by the point cloud reconstruction.

2) **If the Q × Q pixel values in the same spatial locations in the BOF_N are NOT *all 0s*, the search is required in the next deeper depth.**

We denote the proposed fast CU algorithm that uses **BOF_N** as **FastCU_N.** Figure 8 shows the flow diagram of the **FastCU_N**.

Figure 7 (b) illustrates an example resulting partition of the proposed **FastCU_64**. Since the **BOF_64** has unoccupied 64 × 64 blocks in circled 1, 2 and 3, the attribute blocks of the same locations are staying at depth 0 and are prohibited

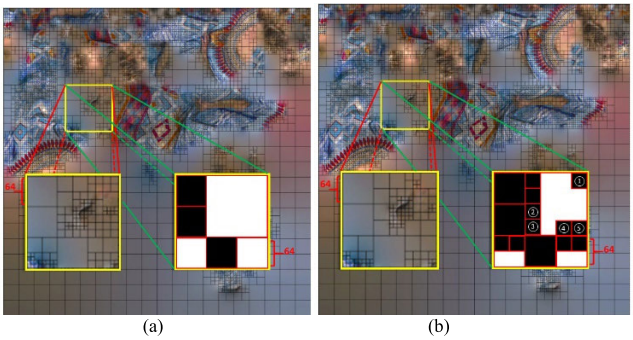


FIGURE 9. CU partition results of (a) FastCU_64 and (b) FastCU_32. The windows with black-and-white pixels in (a) and (b) are the BOF_64 and the BOF_32 of the same spatial location, respectively. Yellow windows (target area) are of the same spatial locations. The blocks with circled number in (b) are desired effects of the proposed work.

to go deeper, to save more time, compared with Fig. 7 (a) of the same locations.

Another example is illustrated in Fig. 9. Compared with **FastCU_64**, due to more smaller unoccupied blocks in **BOF_32**, **FastCU_32** can early-terminate for more smaller CUs such as blocks circled 1∼5, that go unnecessarily deeper in **FastCU_64**. Therefore, **FastCU_32** can save more time than **FastCU_64** can.

IV. FAST CU ENCODING ALGORITHM IN HEVC USING 2D&3D INFORMATION FOR OCCUPIED CU

In previous section, a fast CU method **FastCU_N** for unoccupied blocks using **BOF_N** is proposed. In this section, fast CU methods for fully-occupied CU and partially-occupied CU are designed, using 3D and 2D information with the geometry map, respectively.

A. 3D INFORMATION FOR FULLY-OCCUPIED CU

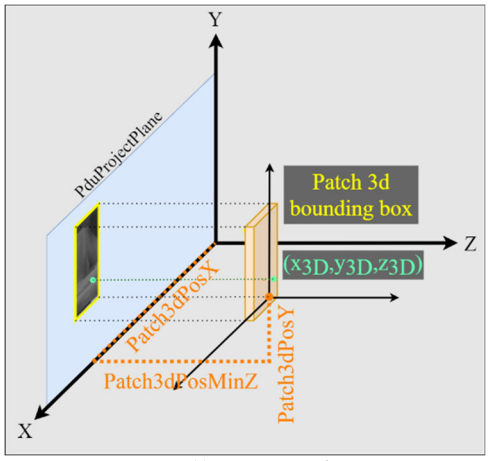
In V-PCC, a crucial part is the 2D-3D transformation of coordination of each patch in the encoder and the decoder, to transform the pixel locations between 3D point clouds and the 2D images. Based on the standard documents [4] and the source code [5], we can compute the 3D location of each 2D pixel using its 2D location (x_{2D}, y_{2D}), the geometry pixel value $G(x_{2D}, y_{2D})$, and various parameters, which are shown in Fig. 10 (created based on the standard documents [4]). The computation formula is as follows:

$$\begin{cases} x_{3D} = (x_{2D} - \text{Patch2dPosX} \times \text{bpr}) + \text{Patch3dPosX} \\ y_{3D} = (y_{2D} - \text{Patch2dPosY} \times \text{bpr}) + \text{Patch3dPosY} \\ z_{3D} = \text{Patch3dPosMinZ} + G(x_{2D}, y_{2D}) \end{cases} \quad (1)$$

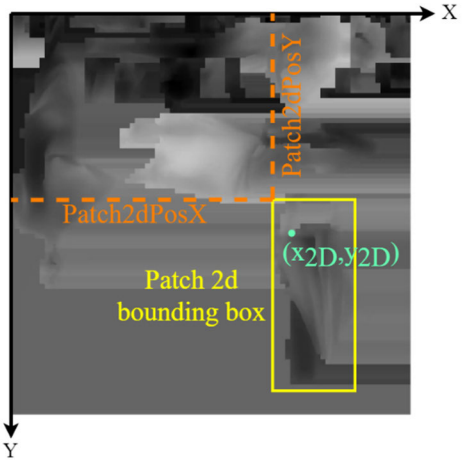
where *bpr* (block resolution) is 16, and other parameters are illustrated in Fig. 10.

This generation of 3D location of each 2D pixel can be used when we perform 2D HEVC compression: it will help us understand the 3D location information of the patches/objects in a block in the 2D image frame we are going to compress.

We use this information to determine whether the patches/objects in a CU have large variation in 3D spatial locations. If they do, the CU might need finer CU partition



(a) 3D representation



(b) 2D representation

FIGURE 10. Transformation between 2D and 3D [4].

to fully record the information. Otherwise, if they are close in 3D locations, the finer CU partition is not necessary, achieving fast coding.

To be able to execute the idea, the current CU must be fully-occupied (to have geometry value $G(x_{2D}, y_{2D})$ for each pixel). For an $N \times N$ 2D CU block, the 3D locations of all its pixels can be computed based on eq. (1), to obtain $(x^i_{3D}, y^i_{3D}, z^i_{3D})$, $i = 1 \sim N^2$. The 3D location variances for each direction of x, y and z can be derived as:

$$\begin{cases} var_x = \dfrac{1}{N^2}\sum_{i=1}^{N^2}(x^i_{3D} - \mu_{x_{3D}})^2 \\ var_y = \dfrac{1}{N^2}\sum_{i=1}^{N^2}(y^i_{3D} - \mu_{y_{3D}})^2 \\ var_z = \dfrac{1}{N^2}\sum_{i=1}^{N^2}(z^i_{3D} - \mu_{z_{3D}})^2 \end{cases} \tag{2}$$

where μ indicates the average operation. We measure the average of them to have:

$$var_{3D} = \frac{var_x + var_y + var_z}{3} \tag{3}$$

Based on the design logic, when this 3D variance var_{3D} is small enough ($var_{3D} < TH_{3D}$), the HEVC does not need to perform finer CU partition (to save encoding time) due to the homogeneity of the 3D locations of the CU pixels. The optimal threshold TH_{3D} will be derived in the experimental section using 3D point cloud measurements.

B. 2D INFORMATION FOR PARTIALLY-OCCUPIED CU

The method in the subsection A can only be applied on the CU that is fully occupied since the geometry pixel values $G(x_{2D}, y_{2D})$ of all the pixels are required. For the CU that is partially occupied, we introduce the following method.

For the CU that is partially-occupied, the un-occupied part will be padded (as discussed previously) with estimation, which are not suitable to be used for the process in subsection A due to estimation error propagation. Therefore, the use of this estimation stays within 2D domain.

For a 2D partially-occupied $N \times N$ CU block, after padding, the geometry values including the estimations can be defined as $\tilde{G}\left(x^i_{2D}, y^j_{2D}\right)$ for $i = 1 \sim N, j = 1 \sim N$. The 2D geometry variance of this block can be computed as:

$$var_{2D} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\tilde{G}\left(x^i_{2D}, y^j_{2D}\right) - \mu_{\tilde{G}}\right)^2 \tag{4}$$

Similarly, when var_{2D} of the current CU is small enough ($var_{2D} < TH_{2D}$), it means the depths of the objects in this CU are similar. Therefore, it is not required to perform finer CU partition. Again, the optimal threshold TH_{2D} will be found in the experimental section using 3D point cloud measurements.

V. MODIFIED RATE-DISTORTION OPTIMIZATION

Rate-distortion optimization is used in HEVC to help make decision and find out the best prediction candidate, considering a weighted sum J of the distortion D and the possibly consumed bits R of a particular testing decision/candidate:

$$\mathrm{J} = \mathrm{D} + \lambda \mathrm{R} \tag{5}$$

where λ is the Lagrange multiplier, which varies from QPs and other factors. The decision/candidate with the minimum J is considered to be the optimal decision/candidate.

When compressing luma Y layer, the weighted sum J is represented as:

$$\mathrm{J_Y} = \mathrm{D_Y} + \lambda_\mathrm{Y}\mathrm{R_Y} \tag{6}$$

And when compressing chroma U and V layers:

$$\mathrm{J_{UV}} = \mathrm{D_{UV}} + \lambda_\mathrm{UV}\mathrm{R_{UV}} \tag{7}$$

In the proposed work, we aim to adjust the weighting by the proposed parameters $\mathrm{P_Y}$ and $\mathrm{P_{UV}}$ for $\mathrm{J_Y}$ and $\mathrm{J_{UV}}$, respectively:

$$\begin{cases} \mathrm{J_Y} = \mathrm{D_Y} + \mathrm{P_Y}\cdot\lambda_\mathrm{Y}\mathrm{R_Y} \\ \mathrm{J_{UV}} = \mathrm{D_{UV}} + \mathrm{P_{UV}}\cdot\lambda_\mathrm{UV}\mathrm{R_{UV}} \end{cases} \tag{8}$$

For each 3D point cloud frame, V-PCC will generate two attribute frames: far layer frame and near layer frame [4],

as shown in Fig. 11. Near layer frame, recorded at even POC, is first produced to store points with lowest depth. And far layer frame, recorded at odd POC, is then produced for points with highest depths [4].

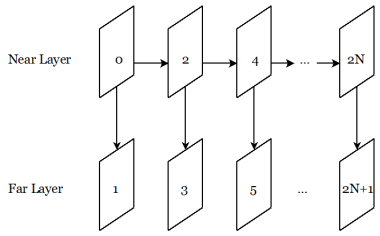


FIGURE 11. Example of layer projection structure and POC [4].

Since both layer frames are similar, and the near layer frame acts as the reference frame for the far layer frame, our modification of rate-distortion optimization only applies on the near layer frame since it is more important. That is, eq. (8) is only applied for even POC in the HEVC system. The optimal parameters P_Y and P_{UV} in eq. (8) will be found in the experimental section using 3D point cloud measurements.

VI. NULLING ALGORITHM FOR INPUT IMAGE USING BOF

In Section 3, BOF is used for developing fast CU decision algorithm to reduce the coding time. In this section, BOF is to be used to modify the input image to the HEVC algorithm, to reduce the input information and possibly achieve lower bitrate and complexity. Note that this idea is implemented after the padding, for lower complexity and more compatibility to the standard pipeline.

As discussed, BOF identifies the blocks that contain no original occupancy pixel. For the input image (Fig. 3(b)) to HEVC, the padded pixels in those blocks are not helping the reconstruction of the point cloud, and are just wasting coding complexity and coded bits. Therefore, we aim to null out (make black) those pixels, on a block basis, giving HEVC some idea to not encode them with much resources. To this end, as in Fig. 12, the padded image in (Fig. 12(a)) is point-by-point multiplied with **BOF_N** (Fig. 10(b)), resulting in a blocky image (Fig. 12 (c)). This blocky image is then input to the HEVC for encoding. We denote this method as **Null_N**.

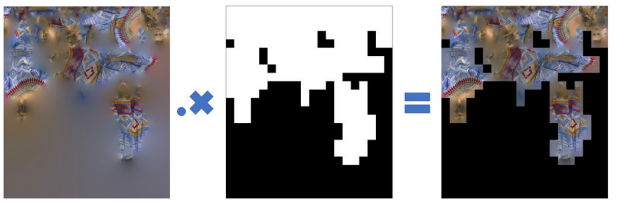


FIGURE 12. Illustration of Null_N method where N = 64 in this example.

By this procedure, we force the 1-occupied pixels to be in some hypothetical blocks, and **the padded pixels in those blocks are kept and useful to reduce the high frequency of those blocks for better coding efficiency**. But for the rest of the blocks that do not contain any 1-occupied pixels, they are nulled out (made black) **to save the corresponding coding complexity in the video coder and the coded bits.**

Figure 13 shows decoded images when the encoder activates **Null_N**. As can be seen, as the N becomes smaller, it approaches to unpadded image (as Fig. 3(a)). Therefore, the bit saving performance can start to be worse at a certain point, as the N becomes smaller. We will discuss the experiment results for **Null_N** with different Ns.

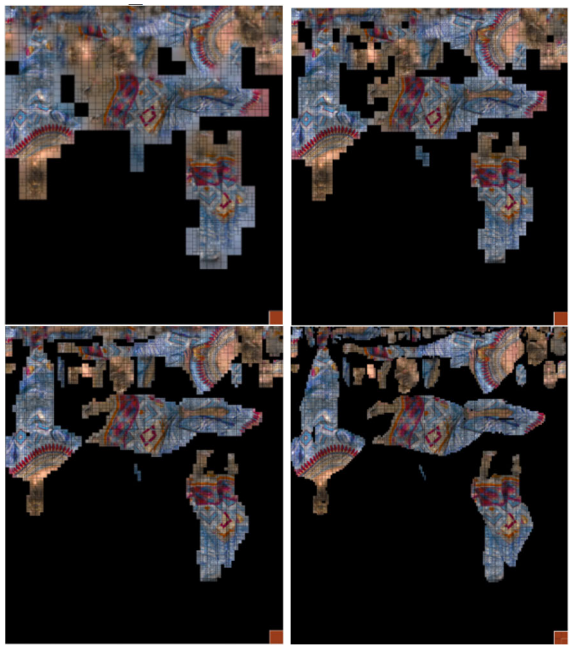


FIGURE 13. Decoded images (displayed with coded partition information) with Null_N activated in encoder with N = 64, 32, 16 and 8, displayed in raster order.

VII. EXPERIMENTAL RESULTS

In this section, we discuss the test conditions, optimal parameter selections for different methods, and experimental results of the proposed works against anchor and the state-of-the-art work [13] with fast coding method for V-PCC.

A. TEST CONDITIONS

The V-PCC reference anchor we used is V-PCC TMC2 [5] with HEVC HM-16.20+SCM-8.8 [14]. The coding configuration is *Random Access*. We follow the test condition described in [15]. The dynamic point clouds (DPC) for the experiment are Class A (*loot*, *redandblack*, *soldier*, *queen*), Class B (*longdress*), Class C (*basketball_player*, *dancer*) [16], [17]. We also use Microsoft Voxelized Upper Bodies dataset with *andrew10, david10, phil10, ricardo10, sarah10* in [18], [19]. The rates are from low (r1) to high (r5), as defined in [15]. The number of coded DPC frames is 32 for both datasets. The padding algorithm is *smooth pushpull*.

The Bjøntegaard delta bit rates (BDBR) of various aspects of the reconstructed DPC, by a modified method against the anchor, are computed by [15]. The BDBR (%) is the average increase (positive number) or decrease (negative number) bits in percentage to reconstruct point clouds of the same quality; this number is the lower the better. GeomRate is computed by Geometry PSNR and Geometry bitrate. AttrRate is computed by Attribute PSNR and Attribute bitrate. TotalRate considers the PSNR of both Attribute and Geometry against total bitrate. D1 is for point-to-point computation, whereas D2 is for plane-to-plane computation. For the complexity measurement, we emphasize the *encoder child time* (%, the percentage of running time by the modified method over that by the anchor) for video codec (HEVC), computed by [15]. Above measurements are against the V-PCC reference anchor.

In the following subsections B∼E, DPC dataset is used to analyze and find the optimal parameters for different proposed methods. And in subsection F where the overall proposed system is compared with the anchor and the state-of-the-art work [13], in addition to DPC dataset, we also test on Microsoft Voxelized Upper Bodies dataset, to validate the effectiveness of the proposed works and the parameters in different datasets.

We also note that in the following subsections B∼E to select optimal parameters for a specific sub-proposed method, the other sub-proposed methods are not activated, to isolate the effects.

B. OPTIMAL N FOR FASTCU_N

Table 2 shows the experiment results of average BDBR and encoder child time of the proposed **FastCU_N** (proposed in Section 3) against the V-PCC reference anchor for different N, $N = 64$, 32 and 16. Each entry is the average performance over all tested DPCs. As shown in the table, the encoder child time *decreases* as N decreases. This means that as we allow smaller N to consider more unoccupied blocks to early terminate, less encoding time can be achieved (can be from 66.73% to as low as 57.69%). And for the BD performance, there are some minor degradations (0.1%) in some cases, but also some improvements (−0.1%∼−1.1%) in other cases. **FastCU_16** achieves significantly lower encoding time 57.69%, and BD rates do not change significantly. Therefore, we choose $N = 16$ for **FastCU_N**.

TABLE 2. Average BD and encoder child time for FastCU_N in different *N*s. Shaded number is negative, bold number is positive.

N	BD (%)										encoder child time (%)
	Geom. BD-TotGeomRate		End-to-End BD-AttrRate			Geom. BD-TotalRate		End-to-End BD-TotalRate			
	D1	D2	Luma	Cb	Cr	D1	D2	Luma	Cb	Cr	
64	0.0	**0.1**	-0.1	0.0	-0.3	0.0	**0.1**	-0.1	0.0	-0.2	66.73
32	0.0	**0.1**	-0.1	-0.8	-1.2	**0.1**	**0.1**	-0.1	-0.5	-0.8	60.49
16	**0.1**	**0.1**	-0.1	-1.1	**0.1**	**0.1**	**0.1**	0.0	-0.6	**0.1**	57.69

C. OPTIMAL VARIANCE THRESHOLD $\{TH_{3D}, TH_{2D}\}$

In Section 4, we propose to use 3D and 2D spatial information of the pixels in a CU to design a fast encoding method for fully-occupied and partially occupied CU blocks with threshold TH_{3D} and TH_{2D}, respectively. As stated, this 3D/2D information originated from the geometry map information, therefore here we focus on the geometry map. In this sub section, we aim to find a pair of thresholds $\{TH_{3D}, TH_{2D}\}$ that makes the CU encoding time reduction large (this method is about fast coding), and make the resulting BDBR small. Therefore, we use a measurement TS/BDBR from [20], [21], where

$$\text{TS} = \frac{Time_{anchor} - Time_{proposed}}{Time_{anchor}} \tag{9}$$

and BDBR = GeomRate.D1 is used. That is, an optimal pair of thresholds $\{TH_{3D}, TH_{2D}\}$ is to be found to maximize TS/BDBR. To obtain it, we use different TH_{3D} and TH_{2D} from 5 to 25 in the increment of 5, for the method in Section 4 for different runs of encoding. The results of different runs are used to compute TS/BDBR. The TS/BDBR against different $\{TH_{3D}, TH_{2D}\}$ are plotted in Fig. 14. As can be seen, the maximum values of TS/BDBR occurs at about $\{TH_{3D}, TH_{2D}\} = \{20, 10\}$; these values are to be used as our optimal thresholds for our method in Section 4.

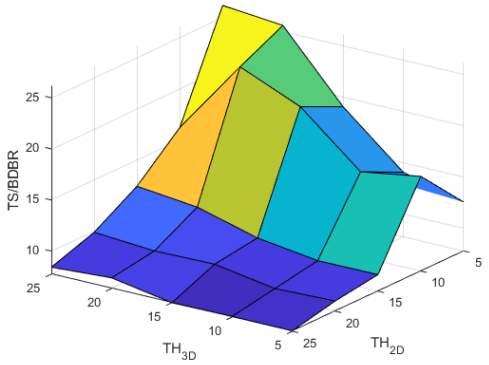


FIGURE 14. TS/BDBR against $\{TH_{3D}, TH_{2D}\}$.

D. OPTIMAL RE-WEIGHTING PARAMETER $\{P_Y, P_{UV}\}$

In Section 5, a modified rate-distortion optimization is proposed using reweighting factors P_Y and P_{UV}, for luma and chroma components, respectively. It is our purpose to find a pair of P_Y and P_{UV} that can minimize the resulting BDBR considering all the luma Y, chroma Cb (U) and Cr (V) domains. Therefore, we use a combined metric $BDBR_{YUV}$ from [22] to properly weight the BDBR from three different layers:

$$BDBR_{YUV} = \frac{1}{8}(6 \times \text{AttrRate.Luma} + \text{AttrRate.Cb} + \text{AttrRate.Cr}) \tag{10}$$

Thus, optimal parameters P_Y and P_{UV} in our experiment is the pair that minimize $BDBR_{YUV}$. We range P_Y from 0.7 to 1.3 with increment 0.1, and P_{UV} from 0.8 to 1.1 with increment 0.1. The results of the coding runs with different

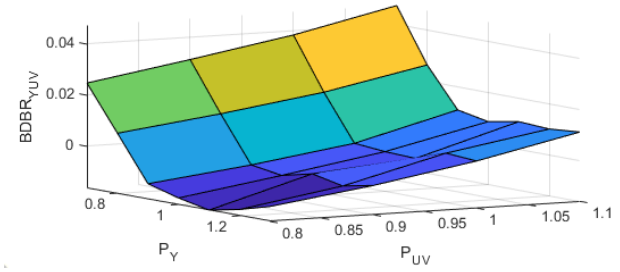


FIGURE 15. $BDBR_{YUV}$ against P_Y and P_{UV}.

TABLE 3. Average BD and encoder child time for NULL_N in different *Ns*. Shaded number is negative, bold number is positive.

N	BD (%)										encoder child time (%)
	GeomRate		AttrRate			TotalRate		TotalRate			
	D1	D2	Luma	Cb	Cr	D1	D2	Luma	Cb	Cr	
64	0.0	0.0	-1.8	-2.4	-3.4	-0.8	-0.9	-1.0	-1.4	-1.9	94.72
32	0.0	0.0	**0.9**	**0.9**	**0.1**	0.0	-0.2	-0.2	0.0	-0.6	94.14
16	0.0	0.0	**5.8**	**8.9**	**5.7**	**1.5**	**1.1**	**1.4**	**3.5**	**1.4**	92.64
8	0.0	0.0	**18.6**	**33.5**	**27.9**	**6.0**	**5.1**	**5.9**	**15.6**	**11.6**	93.45

combinations of P_Y and P_{UV} are used to compute $BDBR_{YUV}$, and the results are illustrated in Fig. 15. As can be seen, the minimum value of $BDBR_{YUV}$ can be found approximately at $\{P_Y, P_{UV}\} = \{1.0, 0.9\}$, which are to be used in our method in Section 5.

E. OPTIMAL N FOR NULL_N

Table 3 shows the experiment results of average BDBR and encoder child time of the proposed **Null_N** (proposed in Section 6) on attribute image against the V-PCC reference anchor in different N, $N = 64$, 32, 16 and 8. Each entry is the average performance over all tested DPCs. **The bitrate decreases in all entries in a row only occur for $N = 64$.** For other smaller *Ns*, there are several increases. This may due to the fact that for smaller N, some small occupied blocks can be coded along with black pixels in a CU, introducing high frequency and thus bitrate increase. Also, for smaller N, some small occupied blocks are coded on its own as a CU; this increases the number of CUs and thus coding overheads. The reason **Null_64** works is for $N = 64$, pixels are more likely to be coded without black pixels, and the coded blocks tend to be larger. Therefore, we choose $N = 64$ for the **Null_N** method. Note that this nulling method is only for attribute image since the improvement of nulling on geometry image is insignificant.

F. PERFORMANCE COMPARISON WITH [13]

In this subsection, we compare the overall proposed work against the state-of-the-art work [13] of fast encoding in V-PCC (implemented in the same versions of HEVC and V-PCC as ours), which includes a fast CU decision method and a fast mode decision method. For fair comparison, the fast mode decision method is also used in the proposed work. The comparisons are made for two different datasets: DPC and Microsoft Voxelized Upper Bodies dataset. In the following tables, *self time* (%) [15] is reported, but it is for TMC2 time, and not for codec (in which our modification takes places) time. Therefore, its changes are very minor and not of our emphasis. For the main concern of the complexity of codec, we emphasize on *child time* (%). The $BDBR_{YUV}$ in eq. (10) is also used in the following tables for the discussion purpose.

For DPC dataset, the numbers in Table 4 are computed based on the proposed work against the V-PCC reference anchor, whereas those in Table 5 are based on the state-of-the art work [13] against the V-PCC reference anchor. Note again that the BDBR and the child encode time is the lower the better. For GeomRate, on average we slightly lose [13] by about 0.26% and 0.38% for D1 and D2. However, for the rest of the BDBR performances on average, we outperform [13] largely by 2.31% in $BDBR_{YUV}$ for AttrRate(Luma, Cb, Cr), by 0.38% and 0.39% for TotalRate D1 and D2, by 1.25% in $BDBR_{YUV}$ for TotalRate(Luma, Cb, Cr). For time saving comparison (*child*), on average, we outperform [13] by 7.84%.

For the comparison in another dataset Microsoft Voxelized Upper Bodies, Table 6 shows the results of the proposed work, and Table 7 shows that of the work in [13]. Similarly, we lose slightly by 0.31% and 0.18% in GeomRate for

TABLE 4. BDBR and running time for proposed method with DPC [16], [17] Dataset.

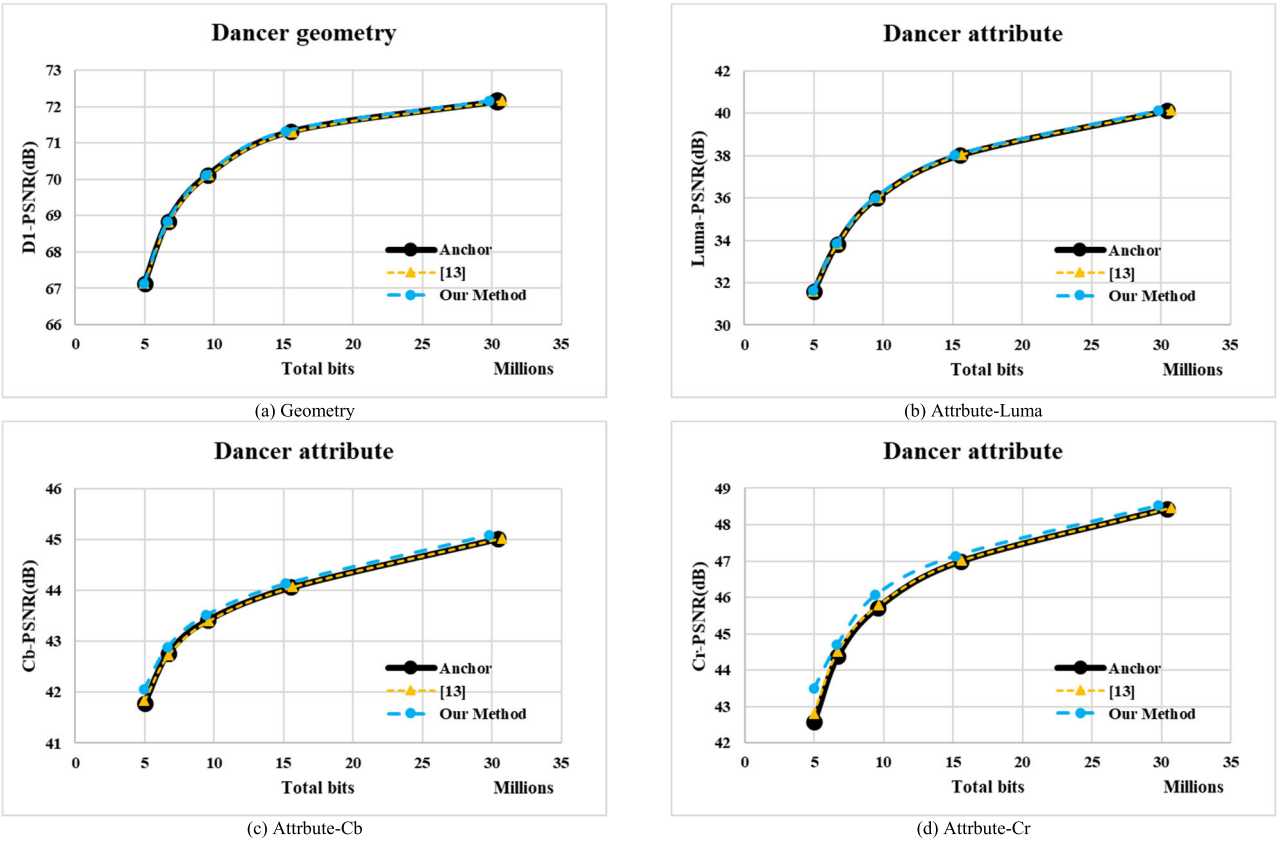
Test point cloud	GeomRate		AttrRate			TotalRate		TotalRate			Enc. Time	
	D1	D2	Luma	Cb	Cr	D1	D2	Luma	Cb	Cr	self	child
Loot	0.12%	0.05%	3.43%	-3.17%	-1.31%	0.76%	0.49%	0.81%	-3.04%	-1.42%	100.98%	49.97%
RedAndBlack	0.80%	0.89%	1.19%	-2.86%	-3.17%	0.97%	0.80%	0.96%	-1.93%	-2.18%	101.47%	45.14%
Soldier	0.27%	0.58%	0.06%	-6.94%	-7.72%	0.32%	0.66%	0.05%	-3.81%	-4.75%	100.39%	63.18%
Queen	0.22%	0.64%	-1.64%	-5.02%	-5.24%	-0.28%	0.07%	-0.64%	-3.06%	-2.91%	100.45%	49.64%
Longdress	0.87%	1.24%	-0.73%	-6.57%	-6.34%	0.43%	0.74%	-0.24%	-4.70%	-4.48%	102.01%	45.07%
Basketball_player	1.27%	1.15%	-5.06%	-10.76%	-10.73%	-0.74%	-1.07%	-1.48%	-5.32%	-5.20%	99.83%	33.59%
Dancer	1.16%	1.17%	-5.31%	-12.61%	-16.43%	-1.32%	-1.59%	-1.64%	-6.74%	-9.25%	100.18%	32.64%
Avg.	**0.67%**	**0.82%**	-1.15%	-6.85%	-7.28%	**0.02%**	**0.02%**	-0.31%	-4.08%	-4.31%	**100.75%**	**44.57%**
$BDBR_{YUV}$				**-2.49%**					**-1.28%**			

TABLE 5. BDBR and running time for [13] with DPC [16], [17] dataset.

Test point cloud	GeomRate D1	GeomRate D2	AttrRate Luma	AttrRate Cb	AttrRate Cr	TotalRate D1	TotalRate D2	TotalRate Luma	TotalRate Cb	TotalRate Cr	Enc. Time self	Enc. Time child
Loot	0.07%	-0.18%	-0.20%	-2.38%	2.16%	0.13%	-0.23%	-0.15%	-1.33%	0.78%	100.28%	58.53%
RedAndBlack	0.42%	0.34%	-0.30%	0.16%	-0.42%	0.27%	0.09%	0.04%	0.44%	-0.07%	100.56%	53.73%
Soldier	0.24%	0.50%	-0.17%	2.04%	1.29%	0.42%	0.72%	-0.10%	1.36%	0.75%	99.73%	74.28%
Queen	0.03%	0.02%	0.12%	1.44%	-0.27%	0.38%	0.39%	-0.19%	0.59%	-0.32%	100.36%	59.61%
Longdress	0.63%	1.00%	-0.20%	0.16%	-0.26%	0.60%	1.11%	0.02%	0.30%	-0.02%	107.03%	55.63%
Basketball_player	0.62%	0.53%	-0.49%	-0.71%	-2.98%	0.33%	0.14%	0.14%	0.07%	-1.21%	100.54%	37.48%
Dancer	0.85%	0.90%	-0.01%	0.56%	-3.19%	0.67%	0.66%	0.42%	0.75%	-1.58%	100.49%	37.38%
Avg.	**0.41%**	**0.44%**	-0.18%	0.18%	-0.52%	**0.40%**	**0.41%**	0.03%	0.31%	-0.24%	**101.26%**	**52.41%**
$BDBR_{YUV}$				**-0.18%**					**-0.03%**			

TABLE 6. BDBR and running time for proposed method with microsoft voxelized upper bodies [19] dataset.

Test point cloud	GeomRate D1	GeomRate D2	AttrRate Luma	AttrRate Cb	AttrRate Cr	TotalRate D1	TotalRate D2	TotalRate Luma	TotalRate Cb	TotalRate Cr	Enc. Time self	Enc. Time child
andrew10	0.19%	0.41%	-0.73%	-7.22%	-6.66%	-0.03%	0.27%	-0.29%	-5.15%	-4.88%	100.30%	51.79%
david10	0.49%	0.53%	-1.39%	-5.23%	-4.94%	0.24%	0.15%	-0.27%	-2.97%	-2.69%	100.65%	54.59%
phil10	0.75%	0.88%	0.40%	-5.16%	-6.02%	0.86%	0.92%	0.42%	-3.65%	-4.21%	100.87%	54.12%
ricardo10	0.49%	0.86%	-2.30%	-7.69%	-0.20%	0.07%	0.31%	-0.09%	-4.07%	1.60%	100.38%	40.64%
sarah10	0.48%	0.59%	0.82%	-3.95%	-1.70%	0.63%	0.65%	0.43%	-2.65%	-1.53%	100.60%	55.82%
Avg.	**0.48%**	**0.65%**	-0.64%	-5.85%	-3.91%	**0.35%**	**0.46%**	0.04%	-3.70%	-2.34%	**100.56%**	**51.06%**
$BDBR_{YUV}$				**-1.70%**					**-0.73%**			



(a) Geometry (b) Attrbute-Luma

(c) Attrbute-Cb (d) Attrbute-Cr

FIGURE 16. RD curve comparisons of *Dancer*, among anchor, Our Method, and [13]. For (a) geometry, (b) attribute-Luma, (c) attribute-Cb, and (d) attribute-Cr.

D1 and D2. We also lose slightly by 0.17% in TotalRate D1. But for other BDBR results, we outperform [13] largely by 1.59% in $BDBR_{YUV}$ for AttrRate(Luma, Cb, Cr), by 0.69% in $BDBR_{YUV}$ for TotalRate(Luma, Cb, Cr), by 0.08% for TotalRate D2. In the aspect of time saving, we perform better than [13] by 6.85%. **Above results show that we have better**

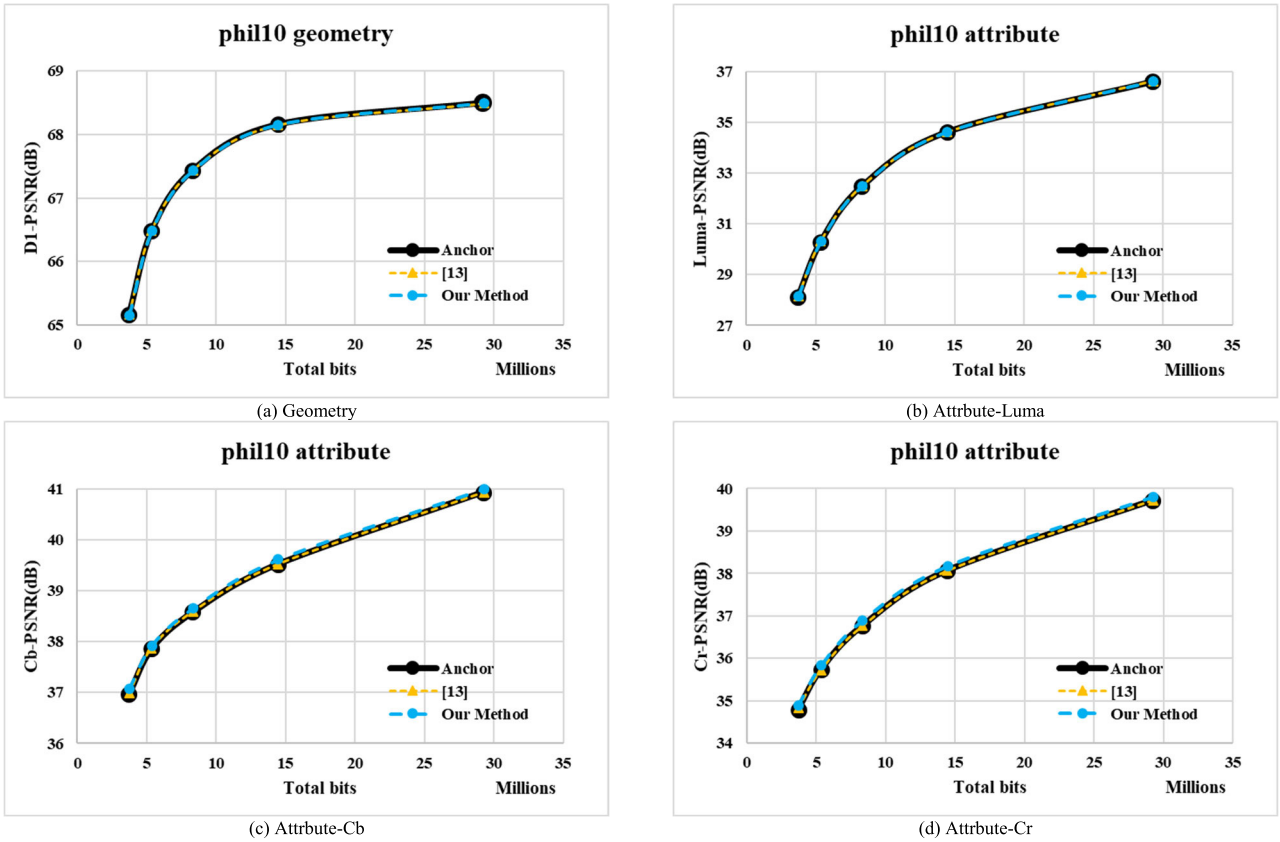


FIGURE 17. RD curve comparisons of *phil10*, among anchor, Our Method, and [13]. For (a) geometry, (b) attribute-Luma, (c) attribute-Cb, and (d) attribute-Cr.

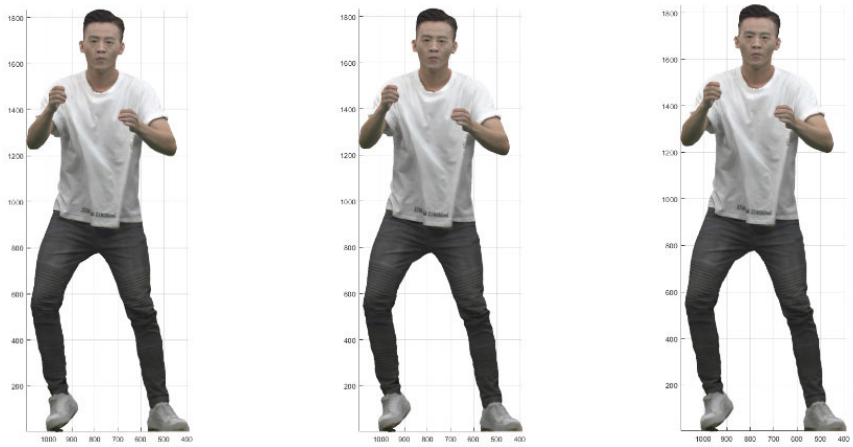


FIGURE 18. Visual comparisons of the reconstruction results (Sequence *Dancer*, R5, frame 32) by (a) anchor, (b) Proposed method, and (c) [13].

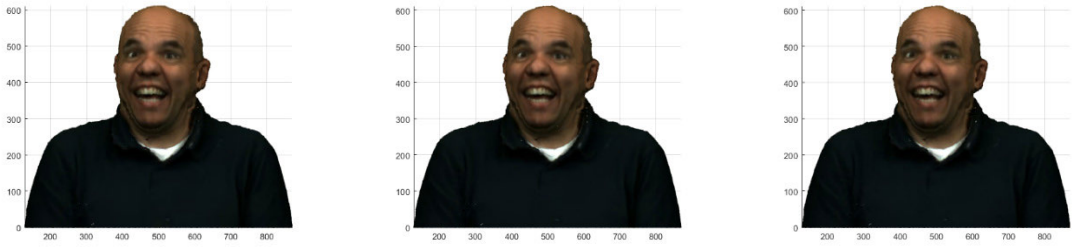
performances than [13] in BDBR and in time saving for different datasets.

On RD curve comparisons, we plot for (a) Geometry (D1-PSNR), (b) Luma-PSNR, (c) Cb-PSNR, and (d) Cr-PSNR using anchor, our method, and [13], for *Dancer* and *phil10* in Fig. 16 and 17, respectively. As can be seen for both figures, the RD curves for Geometry differ very little. And for the performance of Luma, Cb and Cr, we are almost always better than the anchor and [13], on average, as shown in Table 4∼7, and observed in Fig. 16 and 17. **Again, we have to note that the proposed method mainly focuses on complexity reduction, as the state-of-the-art [13] does, as opposed to greatly improving the RD performances.**

On visual comparison, the reconstructed point clouds by anchor, our method, and [13] are shown in Fig. 18 and Fig. 19 for *Dancer* and *ricardo10*, respectively. As demonstrated,

TABLE 7. BDBR and running time for [13] with microsoft voxelized upper bodies [19] dataset.

Test point cloud	GeomRate D1	GeomRate D2	AttrRate Luma	AttrRate Cb	AttrRate Cr	TotalRate D1	TotalRate D2	TotalRate Luma	TotalRate Cb	TotalRate Cr	Enc. Time self	Enc. Time child
andrew10	-0.11%	0.37%	0.01%	-0.51%	0.18%	-0.08%	0.63%	-0.01%	-0.46%	0.06%	99.93%	57.26%
david10	0.38%	0.41%	-0.06%	0.74%	-1.97%	0.42%	0.46%	0.01%	0.69%	-1.13%	101.13%	62.66%
phil10	0.41%	0.39%	0.03%	-0.38%	-0.03%	0.39%	0.37%	0.17%	-0.13%	0.11%	100.09%	61.83%
ricardo10	0.06%	0.69%	0.11%	-2.96%	2.90%	0.09%	0.75%	0.00%	-2.11%	1.89%	97.90%	44.69%
sarah10	0.11%	0.47%	-0.11%	-1.96%	-0.44%	0.09%	0.50%	-0.01%	-0.97%	-0.35%	103.67%	65.67%
Avg.	**0.17%**	**0.47%**	0.00%	-1.02%	0.13%	**0.18%**	**0.54%**	0.03%	-0.60%	0.12%	**100.53%**	**57.91%**
$BDBR_{YUV}$				**-0.11%**					**-0.04%**			



(a) (b) (c)

FIGURE 19. Visual comparisons of the reconstruction results (Sequence *ricardo10*, R5, frame 31) by (a) anchor, (b) Proposed method, (c) and [13].

the proposed methods can generate the reconstructed point clouds with almost the same quality as the anchor does.

VIII. CONCLUSION

In this paper, we proposed a fast algorithm for HEVC used for V-PCC coder. First, the blocky occupancy flag of size *N* (**BOF_N**) is generated using the occupancy image produced by the V-PCC. Based on the **BOF_N**, a fast CU decision method **FastCU_N** that modifies the HEVC is developed for attribute and geometry images. Second, 3D and 2D information are used for fast CU decision using geometry images. Third, a modified rate-distortion optimization is designed for different color components. Forth, a pixel modification method **Null_N** on attribute images for the HEVC input is proposed based on **BOF_N**. Compared with the state-of-the-art method [13], we can improve the BDBR performances by up to 2.31% (with slight loss by only up to 0.38%), and improve the time saving performances by up to 7.84%, considering two testing datasets. Our work on the combination of VVC (Versatile Video Coding) and V-PCC can be investigated as future work.

REFERENCES

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, and P. Cesar, "Emerging MPEG standards for point cloud compression," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[2] L. Cui, R. Mekuria, M. Preda, and E. S. Jang, "Point-cloud compression: Moving picture experts group's new standard in 2020," *IEEE Consum. Electron. Mag.*, vol. 8, no. 4, pp. 17–21, Jul. 2019.

[3] 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, pp. 1–4, Oct. 2020.

[4] *V-PCC Codec Description*, document ISO/IEC JTC 1/SC 29/WG 11 MPEG w19526, Sep. 2020.

[5] *VPCC—MPEG-PCC-TMC2—Realese 8.0*. Accessed: Jun. 2021. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc2/tree/release-v8.0

[6] 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.

[7] L. Li, Z. Li, S. Liu, and H. Li, "Rate control for video-based point cloud compression," *IEEE Trans. Image Process.*, vol. 29, pp. 6237–6250, 2020.

[8] 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, Aug. 2019.

[9] L. Li, Z. Li, S. Liu, and H. Li, "Occupancy-map-based rate distortion optimization and partition for video-based point cloud compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 1, pp. 326–338, Jan. 2021.

[10] L. Li, Z. Li, S. Liu, and H. Li, "Efficient projected frame padding for video-based point cloud compression," *IEEE Trans. Multimedia*, vol. 23, pp. 2806–2819, 2021.

[11] S. Kuanar, "Deep learning based fast mode decision in HEVC intra prediction using region wise feature classification," Ph.D. dissertation, Dept. Elect. Eng., Univ. Texas Arlington, Arlington, TX, USA, 2018.

[12] S. Kuanar, K. R. Rao, M. Bilas, and J. Bredow, "Adaptive CU mode selection in HEVC intra prediction: A deep learning approach," in *Circuits, Systems, and Signal Processing*, vol. 38, no. 11. New York, NY, USA: Springer, 2019, pp. 5081–5102.

[13] 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.

[14] *HEVC HM-16.20+SCM-8.8*. Accessed: Jun. 2021. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20+SCM-8.8/

[15] *Common Test Conditions for PCC*, document ISO/IEC JTC1/SC29/WG11 N18883, Dec. 2019.

[16] E. Eon, B. Harrison, T. Myers, and P. A. Chou, *8i Voxelized Full Bodies, Version 2—A Voxelized Point Cloud Dataset*, document ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) m40059/M74006, Jan. 2017.

[17] Y. Xu, Y. Lu, and Z. Wen, *Owlii Dynamic Human Textured Mesh Sequence Dataset*, document ISO/IEC JTC1/SC29/WG11 m41658, Oct. 2017.
[18] 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.
[19] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, *Microsoft Voxelized Upper Bodies—A Voxelized Point Cloud Dataset*, document ISO/IEC JTC1/SC29/WG11 m38673, 2016.
[20] G. Tang, M. Jing, X. Zeng, and Y. Fan, "Adaptive CU split decision with pooling-variable CNN for VVC intra encoding," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2019, pp. 1–4.
[21] Y. Fan, J. Chen, H. Sun, J. Katto, and M. Jing, "A fast QTMT partition decision strategy for VVC intra prediction," *IEEE Access*, vol. 8, pp. 107900–107911, 2020.
[22] S.-H. Park and J. Kang, "Fast multi-type tree partitioning for versatile video coding using a lightweight neural network," *IEEE Trans. Multimedia*, early access, Dec. 2, 2020, doi: 10.1109/TMM.2020.3042062.

TING-LAN LIN (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Chung Yuan Christian University, Zhongli, Taoyuan, Taiwan, in 2001 and 2003, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego, La Jolla, CA, USA, in 2010. In 2008, he interned with the Display System Group, Qualcomm, San Diego. Since 2011, he has been an Assistant Professor with the Department of Electronic Engineering, Chung Yuan Christian University, Taiwan, where he has been an Associate Professor, since August 2015. Since August 2018, he has been an Associate Professor with the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, where he has been a Professor, since February 2021. His current research interests include (multiview) video compression, pixel estimation, and 3D point cloud video compression, with the techniques of mathematical optimization.

HONG-BIN BU is currently pursuing the M.S. degree with the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan. His research interests include image processing and 3D point cloud video compression.

YAN-CHENG CHEN is currently pursuing the M.S. degree with the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan. His research interests include image processing and 3D point cloud video compression.

JUN-RUI YANG received the M.S. degree from the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, in 2020. His research interests include image processing and 3D point cloud video compression.

CHI-FU LIANG received the M.S. degree from the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, in 2021. His research interests include image processing and 3D point cloud video compression.

KUN-HU JIANG received the M.S. degree from the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, in 2021. His research interests include image processing and 3D point cloud video compression.

CHING-HSUAN LIN is currently pursuing the M.S. degree in electronic engineering from Chung Yuan Christian University, Zhongli, Taoyuan, Taiwan. His research interests include image processing and 3D point cloud video compression.

XIAO-FENG YUE received the M.S. degree from the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, in 2020. His research interests include image processing and 3D point cloud video compression.