

Tool Space Exploration of the V-PCC Patch Generation for Practical Point Cloud Encoding

Louis Fréneau, Alexandre Mercat, Guillaume Gautier, Joose Sainio, and Jarmo Vanne
 Ultra Video Group, Tampere University, Tampere, Finland
 {louis.freneau, alexandre.mercat, guillaume.gautier, joose.sainio, jarmo.vanne}@tuni.fi

Abstract—Video-based Point Cloud Compression (V-PCC) is the latest MPEG standard for visual volumetric data coding. V-PCC leverages the coding efficiency of well-established 2D video codecs by creating patches that link sub-regions of a point cloud to their 2D projections. This paper provides an in-depth analysis of the V-PCC patch generation process, which is the most compute-intensive part of the V-PCC standard. To the best of our knowledge, this is the first work to explore the patch generation tools individually and evaluate their impact on coding efficiency and complexity with different coding parameters. The main purpose of this characterization is to offer key insights into the V-PCC encoder design process and thereby foster the development of practical V-PCC encoders.

Keywords—Extended reality (XR), Visual Volumetric Video-based Coding (V3C), Video-based Point Cloud Compression (V-PCC), patch generation, tool space exploration

I. INTRODUCTION

Recent advances in visual volumetric media technologies have provided the practical means to create detailed 3D digital representations of real-life objects and scenes. This progress has unleashed the potential to offer users immersive *extended reality (XR)* experiences with *six degrees of freedom (6DoF)* in a broad range of applications, such as in communication, remote supervision, gaming, broadcasting, and motion pictures. However, storage and transmission of volumetric data is not possible without efficient compression that has become the key enabling factor for these new XR applications. *Moving Picture Experts Group (MPEG)* has addressed this need by releasing the *Visual Volumetric Video-based Coding (V3C)* standard suite, specified as ISO/IEC 23090-5 [1]. A key component of V3C is the *Video-based Point Cloud Compression (V-PCC)* [1], which compresses dynamic point clouds by leveraging existing 2D video coding solutions.

Fig. 1 outlines the primary steps of the V-PCC encoding flow that converts point clouds into corresponding 2D video representations. The patch generation links areas of the input point clouds to their 2D projections. The patches are then packed into three 2D video frames containing geometry, texture, and occupancy data. Finally, these three 2D videos are encoded, e.g., into the *High Efficiency Video Coding (HEVC/H.265)* format [2]. The final V-PCC bitstream contains these three video streams along with the atlas information, which is needed to reconstruct point clouds from their 2D projections [3].

Currently, the only available implementation of V-PCC compression is a reference encoder called *test model category 2 (TMC2)* [4]. It uses an HEVC reference encoder called *HEVC test model (HM)* [5] as a default 2D encoder and offers an inclusive range of features for both lossless and lossy compression. Though, it is not optimized for practical use. For instance, encoding a single frame with TMC2 in *random*

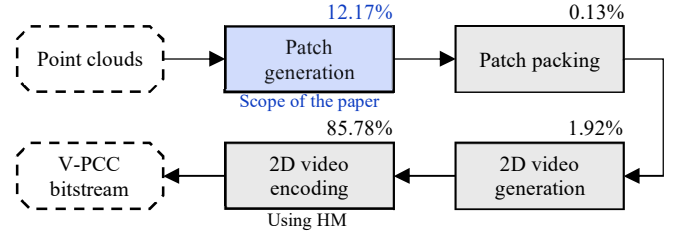


Fig. 1. Main steps of the V-PCC encoding flow with average complexity distribution of TMC2 in RA configuration for reference.

access (RA) configuration takes an average of 20.7s within our experimental setup. As shown in Fig. 1, the encoding time is primarily attributed to HM. However, even with a real-time 2D encoder, the complexity of the patch generation alone prevents the achievement of practical V-PCC encoding.

Recent works have provided an overview of the norm [6], proposed new projections [7], [8], and introduced specific optimization techniques [9], [10] for V-PCC. However, a comprehensive impact analysis of individual V-PCC encoding tools is absent, especially considering the dynamic evolution of the V-PCC standardization. To the best of our knowledge, this paper is the first to fill this gap by undertaking a comprehensive study of the coding tools and parameters in the V-PCC patch generation. This study gives key insights into the steps of the patch generation workflow, elucidating their inherent behaviour, and highlighting their trade-offs on coding efficiency and complexity. Finding the sweet spots is essential for the development and fine-tuning of practical V-PCC encoders.

This paper is structured as follows. Section II details our experimental setup, Section III delves into the proposed tool space exploration, and Section IV concludes the paper.

II. EXPERIMENTAL SETUP

Table I details the test set composed of the first 32 frames of Class A and B sequences from the *common test conditions (CTC)* for V-PCC [11]. *Queen* sequence was excluded due to its unique characteristics and Class C was omitted to maintain manageable coding complexity. Encoding runs were performed with both *all intra (AI)* and RA configurations across the five *rates (R1–R5)*, as defined in the CTC. Note that while MPEG tends to study the impact of new patch generation tools using 2D lossless encoding, our study relies on lossy conditions to model practical coding scenarios.

In our experiments, *Bjontegaard delta bitrate (BD-BR)* [12] was used to measure coding efficiency of the studied tools. BD-BR was computed with PSNR Y and PSNR D2 objective metrics, defined in the CTC [11].

TABLE I. TEST SEQUENCES AND ASSOCIATED PARAMETERS

Class	Sequence	# Pts	Geometry precision	Iteration count	Voxel dimension
A	<i>Loot</i>	~780 000	10-bit	10	2
	<i>Red and Black</i>	~700 000			
	<i>Soldier</i>	~1 500 000			
B	<i>Long dress</i>	~800 000		50	4

This work was supported in part by the XR Simulation and Presence at the Cloud Edge (XR-SPACE) project led by Nokia and funded by Business Finland, and the Academy of Finland (decision no. 349216). In addition, we wish to acknowledge CSC – IT Center for Science, Finland, for computational and storage resources.

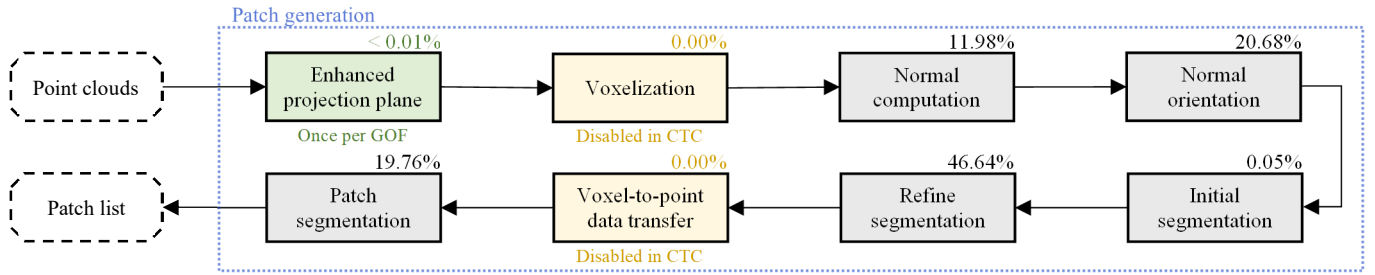


Fig. 2. Main steps of the V-PCC patch generation with average complexity distribution in RA configuration for reference.

PSNR Y and PSNR D2 assess texture and geometry quality of dynamic point clouds, respectively. AI and RA CTC configurations were used as anchors in our BD-BR calculations, so positive values imply coding efficiency loss. Note that the reported BD-BR values only reflect the relative impact of parameters, so they are not meant for coding efficiency comparison of AI and RA configurations. In addition, the atlas size represents the metadata bitstream size derived from the compressed atlas information. It becomes the relative atlas size when compared with the CTC anchor. The atlas size is directly linked to the number of patches and is independent of their individual sizes. Therefore, a larger atlas size indicates higher patch counts which typically leads to smaller patch sizes and a less efficient 2D video encoding [3].

The experiments were conducted on Intel Xeon Gold 6230 @2.1 GHz processors using TMC2 version 21 [4] and single-threaded encoding processes. Although TMC2 encoder is not seen optimal for studying practical coding scenarios, it is currently the only available encoder and it can still bring out the behaviour of individual coding tools. Hence, this study prioritizes relative data analysis, focusing on discerning trends and threshold effects in lieu of absolute complexity values.

Coding complexities of coding tools of interest were measured as wall time. Given that the patch generation process is unaffected by the V-PCC encoding rates (R1–R5) and configurations (AI/RA), complexity values were averaged across these parameters and all frames for better accuracy. In addition to that, standard deviation bars are often included in graphs to illustrate the complexity distribution, which primarily arises from the fluctuating number of points among dynamic point-cloud frames.

III. EXPLORATION OF V-PCC PATCH GENERATION TOOLS

Patches are the core structures of the V-PCC standard, acting as intermediaries between 3D and 2D realms. The patch generation is designed to minimize information loss in 3D to 2D projection while enabling efficient 2D video encoding. Fig. 2 depicts the main steps of this process, excluding auxiliary steps like k -dimensional tree construction.

- **Normal computation** calculates normals for each point in the input point cloud.
- **Normal orientation** ensures these normals share a globally consistent orientation.
- **Initial segmentation** associates each point with a *projection plane index (PPI)*, which refers to a face of the rectangular point cloud bounding box. This step selects the plane that maximizes the dot product between the normal of the point and that of the projection plane.
- **Refine segmentation** is an iterative process that locally enhance the consistency of the PPI selection.
- **Patch segmentation** clusters neighbouring points with the same PPI to create patches.

To speed up the patch generation, the number of input points can be reduced by **voxelization**. Furthermore, specific weights can be calculated during the **enhanced projection plane** step to prioritize front and back projection planes in the initial segmentation. These weights are updated once for each *group of frames (GOF)*.

To ensure clarity and readability, simplified parameter names used in this work are listed in Table II. Their corresponding names in V-PCC and TMC2 are also provided.

A. Normal computation & orientation

Normal computation at each 3D point relies on a matrix calculation, wherein the number of neighbouring points is defined by the *nearest neighbour (NN) count* parameter [13]. However, the obtained normals lack correct orientation, i.e., they do not specify the inside and outside of the global shape.

In Fig. 3, the upper graphs (#1 and #2) present the impact of *NN count* on coding efficiency and complexity of normal computation in RA configuration. The same trends are observed in AI configuration. When *NN count* < 5, the considered neighbourhood does not surround the point, leading to inaccurate normal calculations and inconsistent BD-BRs. When *NN count* \geq 5, there is a linear relationship between coding complexity and efficiency usable to finetune practical coding scenarios. Increasing *NN count* tends to smooth out the normals, diminishing noise but reducing their accuracy at sharp edges. A consistent normal orientation among all points is ensured with a minimum spanning tree,

TABLE II. TOOL AND PARAMETER LEXICON

	V-PCC naming	Paper naming	TMC2 parameter
III.A Fig. 2 Fig. 3	Normal estimation	Normal computation	nnNormalEstimation* normalComputationNN*
		Normal orientation	normalSpanningTreeNN*
III.B Tb. III	Additional projection planes		additionalProjectionPlaneMode
III.C Fig. 2 Fig. 4 Fig. 5 Fig. 6	Fast grid-based refine segmentation	Refine segmentation	gridBasedRefineSegmentation
	Iteration count		iterationCountRefineSegmentation
	Lambda		lambdaRefineSegmentation
	Max point count		maxNNCountRefineSegmentation
	Voxel dimension		voxelDimensionRefineSegmentation
		Search radius	searchRadiusRefineSegmentation
III.D Fig. 8		Max NN count	maxNNCountPatchSegmentation
		Min point per CC	minPointCountPerCCPatchSegmentation
		Surface thickness	surfaceThickness
		Distance detection	maxAllowedDist2RawPointsDetection
III.E Fig. 2 Fig. 8	Grid-based segmentation	Voxelization	gridBasedSegmentation
	Point-to-Voxel Conversion		
	Voxel dimension	Voxel size	voxelDimensionGridBasedSegmentation
III.F Tb. IV	Enhanced projection plane		enhancedPP
	Minimum weight		minWeightEPP

*nnNormalEstimation controls *NN counts* for both computation and orientation. We added two command flags to expose and control separately those parameters.

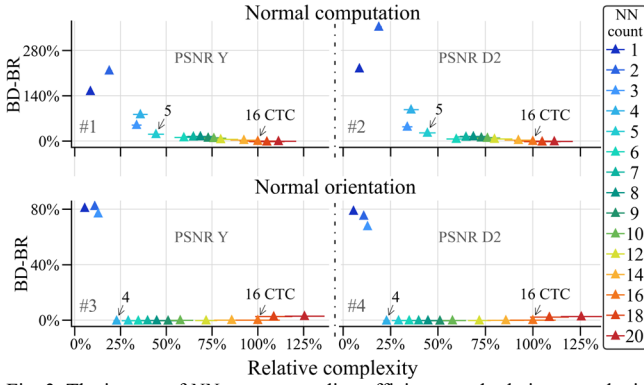


Fig. 3. The impact of $NN\ count$ on coding efficiency and relative complexity in RA configuration for normal computation and orientation.

TABLE III. CODING EFFICIENCY ACROSS ADDITIONAL PROJECTION PLANES

Additional projection planes		6 (CTC)		10x		10y		10z		18xyz	
BD-BR	PSNR	AI	RA	AI	RA	AI	RA	AI	RA	AI	RA
	PSNR Y (%)	0.0	0.0	6.1	16.8	11.6	30.4	5.3	17.8	19.6	50.0
	PSNR D2 (%)	0.0	0.0	7.9	15.7	14.8	29.9	6.6	16.0	26.9	50.9
	Relative atlas size (%)	100.0	100.0	113.4	122.5	121.7	132.1	110.6	127.0	129.0	153.9

which allows a propagation algorithm to rectify normals that deviate from the global orientation [13]. The related $NN\ count$ parameter defines the number of neighbouring points considered to orient each normal.

The lower graphs (#3 and #4) in Fig. 3 illustrate that, as in the normal computation step, low $NN\ count$ leads to inaccurate orientations. However, in contrast to normal computation, increasing $NN\ count$ does not enhance coding efficiency or lead to smoother normal orientation. Indeed, the process can flip the orientation of the normals to ensure that every normal is directed toward the outside of the global shape. This orientation stays consistent for nearby normals, whether we consider a big or small neighbourhood. Consequently, for practical applications, an $NN\ count$ value such as 4 allows significant complexity savings without any coding efficiency overhead.

B. Initial segmentation

TMC2 allows the use of *additional projection planes*. In the CTC, the ‘6’ planes of the input bounding boxes are used. Another configuration is ‘10x’ which adds four extra planes parallel to the x-axis and rotated by 45°. ‘10y’ and ‘10z’ follow the same pattern for their respective axes. Finally, the ‘18xyz’ configuration includes the six default planes and all twelve additional planes, four for each axis direction.

Table III presents the effect of *additional projection planes* on coding efficiency. These results show that adding projection planes tends to decrease the coding efficiency while increasing the number of patches. Indeed, a greater number of planes leads to fewer points sharing the same PPI, resulting in smaller patches and less efficient 2D video encoding. Notably, the ‘10y’ configuration influences the coding efficiency more than the ‘10x’ and ‘10z’ configurations. Indeed, CTC sequences capture full-body standing people, leading to a more frequent selection of y-axis plane.

However, the objective metrics reported in Table III do not capture subjective quality improvements. Adding projection planes reduces the angular difference between the normals of the points and their corresponding projection plane. This reduces the number of occluded points, which decreases the number of missing points in the encoded point cloud and mitigates issues like cracks and holes. Note that to add

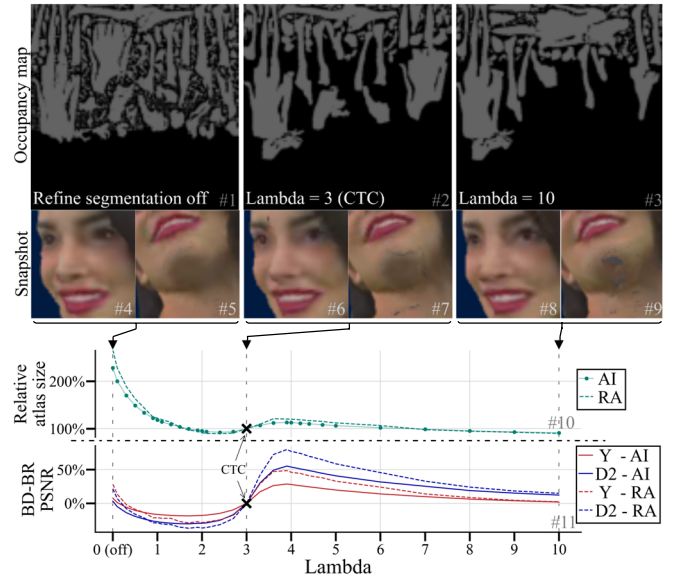


Fig. 4. λ impact on coding efficiency, illustrated with snapshots and occupancy maps from *Red and Black* (AI-R3) first frame. The coding efficiency metrics are averaged across all sequences. Markers on the plain green curve (#10 AI) indicate the λ values used for all measurements.

projection planes do not significantly impact the patch generation complexity.

C. Refine segmentation

The refine segmentation process is an iterative step that smooths out the PPI selection, ensuring that neighbouring points are projected on the same plane. It allows the creation of patches that suits 2D video encoding: large, without gaps, and with smooth boundaries. Moreover, a pre-processing step allows for a significant reduction of the points processed during the iterative smoothing operation [14]. Despite this reduction, the refine segmentation step still remains the most complex step of the patch generation.

The refine segmentation process seeks to find a trade-off between the 2D video coding efficiency and the presence of cracks and holes in the compressed point cloud. This can be illustrated with the λ parameter which determines the degree of influence of neighboring PPIs during a smoothing iteration. Its impact on coding efficiency is shown in Fig. 4. Snapshots and occupancy maps are also provided for specific λ values. With λ set to 0, the refine segmentation becomes disabled and the original PPI selection is used during patch segmentation. Each point is then projected onto its optimal projection plane. This approach minimizes occlusions and prevents cracks and holes. However, it results in numerous small patches, leading to inconsistent patch placement across frames. This temporal inconsistency clearly reduces RA coding efficiency, as shown in the bottom graph (#11) of Fig. 4, whereas the AI case is only slightly impacted by the artefacts emerging from the high number of borders. Conversely, a high λ value, like 10, leads to fewer but larger patches. As shown in the snapshot #9 of Fig. 4, this can cause points, such as those on the chin, initially assigned to the bottom plane during the initial segmentation, to switch to less optimal planes. This shift is due to the strong influence of nearby points and increases the risk of occlusions, ultimately leading to the formation of cracks and holes. However, the larger patch size benefits 2D inter coding, bringing the relative coding efficiencies of AI and RA configurations closer together.

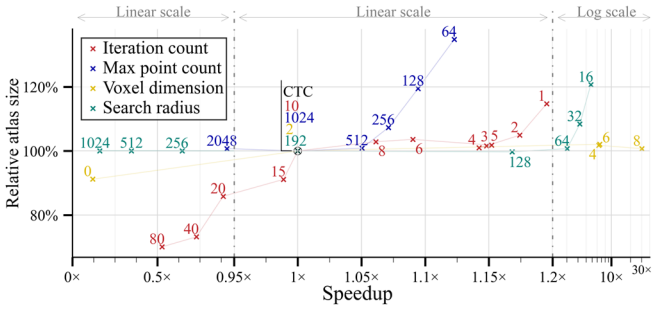


Fig. 5. Atlas size and refine segmentation speed when varying independently selected parameters. Average measures over class A only.

The smoothing operation of the refine segmentation process is performed iteratively according to *iteration count*. Fig. 5 illustrates the effect of this parameter on atlas size, which is similar to the impact of *lambda*. Both parameters determine the degree of smoothing, but unlike *lambda*, *iteration count* is inherently tied to the complexity of the iterative process. Thus, in practical applications, optimizing the refine segmentation process could involve minimizing *iteration count* while compensating with higher *lambda*.

The other parameters of the refine segmentation process affect the pre-processing step and influence the range of the smoothing operation. *Voxel dimension* controls a voxelization process, different from the one shown in Fig. 2. The obtained voxels are classified based on the similarity of the PPI among the points they contain. This optimizes the smoothing process by focusing only on points that are likely to be situated at patch borders. Fig. 5 and Fig. 6 show how *voxel dimension* impacts the coding efficiency and complexity of the refine segmentation process. Notice that classes A and B do not share the same CTC configuration (see Table I); similar trends are obtained but the orders of magnitude are different. The results show that higher *voxel dimension* can substantially reduce the coding complexity of the refine segmentation process while moderately degrading the coding efficiency. This degradation is primarily due to the voxelization not only allowing for skipping non-relevant operations, but also averaging the PPI from adjacent areas during the smoothing process. High *voxel dimension* can then reduce the precision of the refine segmentation process.

Search radius and *maximum point count (MPC)* control which neighbouring points are considered for each smoothing operation. During pre-processing, a *k*-dimensional tree is used to create a list of neighbouring voxels for each voxel, all within a defined *search radius*. This adjacency list is then refined until the total number of points across all neighbouring voxels reach the *MPC* threshold. When this threshold is met, remaining neighbouring voxels are removed from the list. With high *search radius*, many voxels may be considered as neighbouring voxels, but *MPC* limits the number used for smoothing. This can be seen in Fig. 5, where beyond a value of 128, *search radius* does not impact the patch generation process but increases complexity. Thus, for practical applications, *search radius* should be regarded as a mean of controlling the smoothing process complexity and *MPC* as a fine-tuning parameter controlling its efficiency.

D. Patch segmentation

During the patch segmentation step, neighbouring points in the input point cloud sharing the same PPI are clustered into *connected components (CCs)*. To extract a patch from a CC, all points that violate a set of constraints, such as the maximum patch thickness, are removed from the CC. The

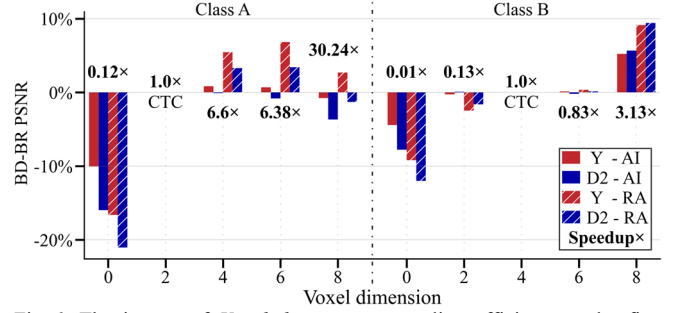


Fig. 6. The impact of *Voxel dimension* on coding efficiency and refine segmentation speed.

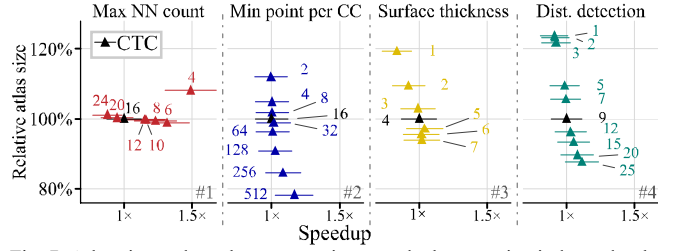


Fig. 7. Atlas size and patch segmentation speed when varying independently selected parameters.

process iterates until all points have been either associated with a patch or dismissed. This study focuses on a set of parameters that significantly influence the patch segmentation process. Their impact on coding efficiency is shown in Fig. 7.

Maximum NN count defines the number of neighbouring points considered for each point during the CC clustering. Graph #1 of Fig. 7 shows that setting *maximum NN count* to 6 results in a significant speedup, while also slightly improving coding efficiency. Conversely, a lower value, like 4, reduces coding efficiency. In that case, the NN count may be insufficient, causing the clustering algorithm to miss points located in a continuous neighbourhood.

Minimum point per CC is directly linked to the number of missing points in the encoded point cloud. A high value decreases the number of small patches, which leads to cracks and holes. A low value increases the number of small patches, which makes the 2D video encoding less efficient.

Surface thickness refers to the maximum depth difference that can separate the closest and farthest points of a patch in relation to its projection plane. In V-PCC, not all points within patches are encoded. Points can be missed when multiple points share the same location on the projection plane. Higher *surface thickness* increases the likelihood of encountering such situations. However, lower *surface thickness* increases the number of small patches.

Points that are part of a CC but do not become part of a patch are addressed in the next iteration of the patch segmentation process. At the end of each iteration, a temporary point cloud is generated, including all points being part of a patch. To determine if a remaining point can act as the starting point for a new CC in the next iteration, the distance between the point and its nearest neighbour within the temporary point cloud needs to be higher than the *distance detection* value. This mechanism serves as a filter, preventing the propagation algorithm from creating CCs that would likely be excessively small. Lowering the *distance detection* value results in an increased number of very small patches.

Fig. 7 shows that the *minimum point per CC*, *surface thickness* and *distance detection* parameters have a non-

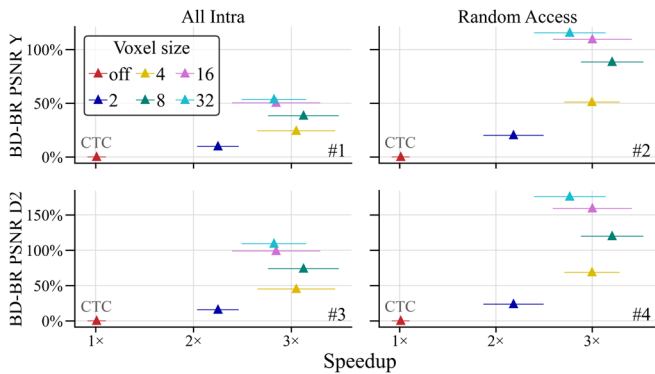


Fig. 8. *Voxel size* impact on coding efficiency and patch generation speed.

significant impact on the patch segmentation coding complexity but are highly correlated with the final number of patches. Those parameters must therefore be considered to fine-tune 2D video encoding efficiency.

E. Voxelization

The extensive smoothing applied during the refine segmentation process makes calculations at the individual patch level irrelevant. Thus, the voxelization step allows to partition the input point cloud using a 3D grid based on the *voxel size* parameter. This associates all points within a grid cell to its centre and makes subsequent calculations (see Fig. 2) focus solely on these central points. Before patch segmentation, the PPI associated with the centre of a grid cell is directly applied to the points inside it.

Fig. 8 presents the impact of *voxel size* on coding efficiency and complexity. A *voxel size* of 8 or lower speeds up the whole patch generation significantly, with only a moderate loss in coding efficiency. Those results emphasize the critical role of the voxelization step in practical coding. However, *voxel sizes* higher than 8 not only reduce speedup but also degrade coding efficiency. Indeed, in those cases, the refine segmentation process becomes less effective, leading to a higher patch count. This shifts complexity towards patch segmentation, negating the initial optimization goal.

F. Enhanced projection plane

The enhanced projection plane tool improves the 2D video encoding efficiency by increasing the patch size. It enlarges patches projected onto front and back planes while reducing the size of top and bottom patches. Projection plane weights, set at the first frame of each GOF, are calculated based on the ratio of the area covered by projecting the input point cloud onto the XY, YZ, and ZX planes. The *minimum weight* parameter controls how biased the projection plan selection is. For instance, when set to 0, no points are projected onto the top and bottom planes. Its impact on coding efficiency is detailed in Table IV. This tool modifies the initial PPI of points on surfaces parallel to the top and bottom planes, resulting in a greater angular difference between the normal of the point and their selected projected plane. This can increase occlusions and lead to the formation of cracks and holes,

TABLE IV. CODING EFFICIENCY ACROSS MINIMUM WEIGHTS

Minimum weight	BD-BR PSNR Y		BD-BR PSNR D2		Relative atlas size	
	AI	RA	AI	RA	AI	RA
1.0 (Off)	1.6%	8.0%	1.7%	6.3%	105.7%	109.2%
0.8	-0.8%	1.9%	-1.2%	0.5%	100.4%	102.7%
0.6 (CTC)	0.0%	0.0%	0.0%	0.0%	100.0%	100.0%
0.4	3.3%	9.2%	4.8%	9.1%	101.1%	103.8%
0.2	5.5%	13.1%	7.9%	13.1%	101.8%	105.3%
0.0	5.5%	13.1%	7.9%	13.1%	101.8%	105.3%

visible from the top or bottom viewpoint. However, these viewpoints are uncommon when handling upright human full-body sequences, which is the main application targeted by V-PCC. As a result, the enhanced projection plane tool effectively improves 2D video coding efficiency with minimal impact on subjective quality. Furthermore, as depicted in Fig. 2, the complexity introduced by this tool is very low.

IV. CONCLUSION

This study provided implementation guidelines for the practical V-PCC encoders. In summary, our results indicated that the current CTC parameters for the V-PCC patch generation are not optimal in practical scenarios. Notably, significant complexity reduction can be achieved by voxelizing the input point cloud and by reducing the range of NN-based algorithms. Furthermore, adjusting parameters in the patch segmentation process and using enhanced or additional projection planes lead to favorable tradeoffs between coding efficiency and subjective quality without any substantial complexity overhead.

Redesigning patch generation and deploying it with real-time 2D video encoders are crucial steps towards practical V-PCC encoding. In addition to that, the patch packing and video generation stages need to be further optimized. Even though they have a minor impact on the V-PCC encoding complexity, they greatly influence the 2D video coding efficiency. Focused research in these areas would therefore be of utmost importance, especially for practical XR applications.

REFERENCES

- [1] ISO/IEC 23090-5:2023. "Information technology — coded representation of immersive media — part 5: visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)," Nov. 2023.
- [2] ITU-T Recommendation H.265, "High Efficiency Video Coding," *Int. Telecommunication Union*, Apr. 2013.
- [3] ISO/IEC JTC 1/SC 29/WG 7, "V-PCC codec description," *document N00100*, Online, Oct. 2020.
- [4] *Test Model Category 2 Version 21*. [Online]. Available: <http://mpegx.int-ervy.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc2> (accessed Mar. 20, 2023).
- [5] *HEVC Reference Software Version 16.20*. [Online]. Available: <https://vcgit.hhi.fraunhofer.de/jct-vc/HM/-/tags/HM-16.20> (accessed Mar. 20, 2023).
- [6] 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, Apr. 2020.
- [7] F. Tohidi and M. Paul, "Dynamic point cloud compression approach using hexahedron segmentation," in *Proc. IEEE Int. Conf. Image Process.*, Kuala Lumpur, Malaysia, Oct. 2023.
- [8] F. Tohidi and M. Paul, "Video-based Point Cloud Compression using density-based variable size hexahedrons," in *Proc. IEEE Int. Conf. Multimedia and Expo Workshops*, Brisbane, Australia, Jul. 2023.
- [9] 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*, Yeosu, South Korea, Oct. 2022.
- [10] Y. Kim and Y. -H. Kim, "A low-complexity patch segmentation in the V-PCC encoder," in *Proc. Int. Tech. Conf. Circuits/Syst., Comput., Commun.*, Jeju, South Korea, Jun. 2023.
- [11] ISO/IEC JTC1/SC29/WG11, "Common test conditions for V3C and V-PCC," *document N19518*, Online, Jul. 2020.
- [12] G. Bjøntegaard, "Improvements of the BD-PSNR model," *document VCEG-A111*, Berlin, Germany, Jul. 2008.
- [13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 71–78, Jul. 1992.
- [14] J. Kim, Y.-H. Kim, and E.-S. Ryu, "Fast grid-based refining segmentation method in V-PCC," ISO/IEC JTC 1/SC 29/WG 7, *document m56635*, Online, May. 2021.