Video-based Point Cloud Compression Artifact Removal

Anique Akhtar, *Student Member, IEEE,* Wen Gao, Li Li, *Member, IEEE,* Zhu Li, *Senior Member, IEEE*
Wei Jia, *Student Member, IEEE,* Shan Liu, *Senior Member, IEEE*

***Abstract*—Photo-realistic point cloud capture and transmission are the fundamental enablers for immersive visual communication. The coding process of dynamic point clouds, especially video-based point cloud compression (V-PCC) developed by the MPEG standardization group, is now delivering state-of-the-art performance in compression efficiency. V-PCC is based on the projection of the point cloud patches to 2D planes and encoding the sequence as 2D texture and geometry patch sequences. However, the resulting quantization errors from coding can introduce compression artifacts, which can be very unpleasant for the quality of experience (QoE). In this work, we developed a novel out-of-the-loop point cloud geometry artifact removal solution that can significantly improve reconstruction quality without additional bandwidth cost. Our novel framework consists of a point cloud sampling scheme, an artifact removal network, and an aggregation scheme. The point cloud sampling scheme employs a cube-based neighborhood patch extraction to divide the point cloud into patches. The geometry artifact removal network then processes these patches to obtain artifact-removed patches. The artifact-removed patches are then merged together using an aggregation scheme to obtain the final artifact-removed point cloud. We employ 3D deep convolutional feature learning for geometry artifact removal that jointly recovers both the quantization direction and the quantization noise level by exploiting projection and quantization prior. The simulation results demonstrate that the proposed method is highly effective and can considerably improve the quality of the reconstructed point cloud.**

***Index Terms*—Point Cloud, Artifact Removal, Compression, 3D Deep Learning, Quantization, V-PCC.**

I. Introduction

Recent significant advances in 3D sensors and capturing techniques have led to a surge in the usage of 3D point clouds in virtual reality/augmented reality (VR/AR) content creation and communications [1], as well as 3D sensing for robotics, smart cities, telepresence [2], and automated driving applications [3]. A 3D point cloud can efficiently represent volumetric visual data such as 3D scenes and objects using a collection of discrete points with 3D geometry positions and other attributes (e.g., color, reflectance). Point cloud data offers advantages over polygonal meshes because it is more flexible and has real-time processing potential, since there is no need to process, store, or transfer surface topological information. With an increase in point cloud applications and improved capturing technologies, we now have high-resolution point clouds with millions of points per frame.

Based on their usage, point clouds can be categorized into point cloud scenes and point cloud objects. Point cloud scenes are dynamically acquired and are typically captured by LIDAR sensors. One example of a dynamic point cloud would be LIDAR sensors mounted atop a vehicle for mobile mapping and autonomous navigation purposes [4]. Point cloud objects can be further subdivided into static objects and dynamic objects. A static point cloud is a single object, whereas a dynamic point cloud is time-varying, where each instance of a dynamic point cloud is a static point cloud. Dynamic time-varying point clouds are used in AR/VR, volumetric video, and telepresence and can be generated using 3D models, i.e. CGI, or captured from real-world scenes using various methods such as multiple cameras with depth sensors surrounding the object and capturing movement over time.

A volumetric video such as a dynamic point cloud provides an immersive media experience. A dynamic point cloud describes a 3D object using its geometry, respective attributes, as well as any temporal changes. Temporal information in the dynamic point cloud is included in the form of individual capture instances, much like 2D video frames. A dynamic point cloud can be viewed from any angle or viewpoint because it includes a complete 3D scene. This six degrees-of-freedom (6DoF) [5] viewing capability makes the dynamic point cloud essential for any AR or VR application. A single instance of a dynamic point cloud captured by 8i [6] could contain as many as one million points. Approximately 30 bits are used to represent the geometry (x,y,z), and 24 bits represent the color (r,g,b). The size of a single instance can be approximated as 6 Mbytes, which translates to a bitrate of 180 Mbytes per second without compression for a 30-fps dynamic point cloud. The high data rate is one of the main problems faced by dynamic point clouds, and efficient compression technologies to allow for the distribution of such content are still widely sought.

The current state-of-the-art dynamic point cloud compression algorithm is the video-based point cloud compression (V-PCC) method [7] which has been selected and developed for standardization by MPEG for dynamic point clouds. Under the V-PCC standard, a point cloud is first projected onto its bounding box patch by patch. Then, the patches are packed

A. Akhtar, Z. Li, and W. Jia are with the Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City, Kansas City, MO 64110 USA (e-mail: aniqueakhtar@mail.umkc.edu, zhu.li@ieee.org, wj3wr@umsystem.edu).

W. Gao and S. Liu are with Tencent America, 661 Bryant St., Palo Alto, CA 94301 (e-mail: wengao@tencent.com, shanl@tencent.com).

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

(a) Higher bitrate

(b) Lower bitrate

Fig. 1. Blocking effects in a point cloud coded at different bitrates using V-PCC encoding.

into a video for compression. During the video compression, the reconstructed geometry may suffer severe quality degradation due to the quantization errors. Blocking artifacts or compression artifacts are often introduced in compressed media due to distortion which is introduced by lossy compression techniques [8]. The V-PCC coded point cloud yields excellent reproduction without noticeable artifacts at high or moderate bitrates. However, at low bitrates, the reconstructed point cloud suffers from visually annoying artifacts due to coarse quantization. Fig. 1 shows two versions of a point cloud encoded at different bitrates. As can be seen, there is no visible blocking artifact in the point cloud coded at a higher bitrate, while severe blocking artifacts exist in the one coded at a lower bitrate. Since blocking artifacts significantly degrade the visual quality of the reconstructed point cloud, it is desirable to identify these artifacts and remove them from the reconstructed point cloud.

This paper proposes the first deep-learning-based geometry artifact removal algorithm for the V-PCC standard for dynamic point clouds. Ours is a pioneering work in V-PCC artifact removal. The proposed framework offers the following contributions:

- We present a projection-aware 3D sparse convolutional neural network-based framework for point cloud artifact removal. Our sparse convolutional network learns an embedding and then regresses over this embedding to learn the quantization noise. Experimental results show that our method significantly improves the quality of the V-PCC reconstructed point cloud in terms of both objective evaluations and visual comparison.
- We observe that the geometry distortion of the V-PCC reconstructed point cloud exists only in the direction of the V-PCC projection. We exploit this prior knowledge to learn both the direction and level of quantization noise by limiting the degree of freedom of the learned noise. We employ Chamfer distance as our loss function and use MSE-PSNR as our quality evaluation metrics.
- We identify a *patch correspondence mismatch problem* that arises due to a difference in the number of points in the original geometry and the V-PCC reconstructed geometry. To solve this, we propose a sampling and aggregation scheme using a cube-centered neighbor search algorithm to find a better correspondence between the reconstructed geometry (after V-PCC encoding) and the original geometry (before V-PCC encoding). The sampling and aggregation scheme makes our method scalable to larger point clouds since the framework is not dependent on the number of points in a point cloud.

II. Background

In 2017, MPEG issued a call for proposals on Point Cloud Compression (PCC) to target an international standard for PCC [9]. As a result of this call, multiple proposals were submitted to MPEG. Since then, MPEG has been evaluating and improving the performances of the proposed technologies. MPEG has selected two technologies for PCC: Geometry-based PCC (G-PCC) [10] for static point cloud data as well as for dynamically acquired LIDAR point cloud data, and video-based point cloud compression (V-PCC) [11] for dynamic content. G-PCC employs octree in its coding scheme, whereas V-PCC projects point clouds onto 2D surfaces and then uses state-of-the-art HEVC video encoding to encode dynamic point clouds. However, V-PCC does introduce compression artifacts, primarily when encoded with a low bitrate.

To the best of our knowledge, compression artifact removal in V-PCC has not been studied so far. However, compression artifact removal techniques and deblocking have been extensively studied in image and video coding. Since V-PCC also employs state-of-the-art HEVC video coding, there is potential to learn from the video compression artifact removal techniques and use them for V-PCC artifact removal. The current state-of-the-art compression artifact removal techniques are based on deep learning. There have been several previous studies using residual networks [12] and GANs [13], as well as efforts employing memory-based deep learning architecture [14]. All of these works are limited to a single image and do not utilize information from previous frames. Recent works, however, have exploited temporal information in restoration tasks to improve video compression artifact removal [15], [16].

Point cloud deep learning has been attracting increasing attention, especially in the last five years [17]. PointNet [18] was among the earliest deep learning architectures for point cloud learning and employed pointwise fully connected

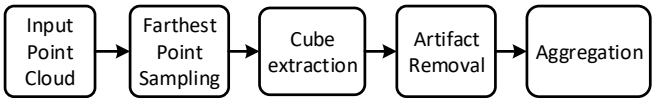
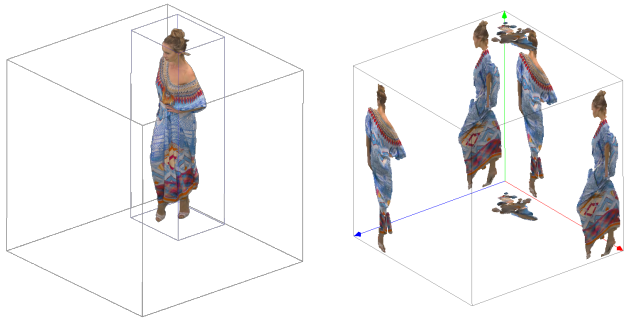


Fig. 2. System Model.

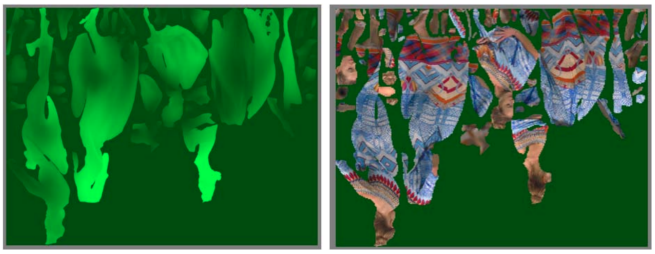
layers followed by max pooling. This architecture was further improved into PointNet++ [19] by adding hierarchical learning that could learn local features with a better contextual scale. Wang et al. [20] proposed an octree-based CNN for 3D shape classification that performs 3D CNN operations on the octants of the octree data structure. PointCNN [21] achieved state-of-the-art results using a convolutional neural network on raw 3D point clouds. To perform convolution on raw point clouds, PointCNN makes the input permutation invariant by learning a transformation matrix using fully connected layers. SparseConvNet [22] by Facebook was among the first sparse convolutional neural networks that achieved state-of-the-art results on point clouds. SparseConvNet introduced submanifold sparse convolutions that exploited the sparse nature of point clouds and ensured that the convolutions would not "dilate" the data. Since sparse convolutions are memory-efficient and the data remain sparse throughout the network, deeper architectures can be used for point clouds. MinkowskiNet [23] is another such implementation that employs sparse convolutions for 3D point cloud learning. Recent works have also explored newer architectures for point cloud learning [24], [25].

The problem of point cloud denoising is an active research field which originated in the early 1990s. The works can be broadly divided into two categories: **optimization-based methods** and **deep-learning-based methods**. The optimization-based methods include techniques such as moving least squares (MLS)-based methods [26], [27], locally optimal projection (LOP)-based methods [28], sparsity-based methods [29], [8], non-local similarity-based methods [30], and graph-based methods [31], [32]. However, the current state-of-the-art solutions are all deep learning-based methods. PU-Net [33] employed deep learning to learn multi-level features for point cloud denoising. This work was further improved to EC-Net [34], which added edge-awareness to the network to further improve the results, especially around the edges. PU-GAN [35] utilized generative adversarial networks (GANs) with patch-based learning for point cloud denoising. PUGeo-Net [36] incorporated discrete differential geometry into deep learning to learn underlying geometric structures from given sparse point clouds. These methods work well for synthetic noise (e.g. Gaussian noise), and some have even been tested on real-world noise introduced during point cloud capture. However, these methods are not optimized to work on compression artifact removal because of the nature of the quantization noise introduced during V-PCC.

Similarly, some work focuses on point cloud inpainting [37], [38], [39], wherein portions of point clouds lost during point cloud capture are completed. However, these methods do not work for compression artifact removal in V-PCC. In the last year, there has been notable work performed on deep learning solutions for point cloud compression [40], [41], [42], [43]. However, these solutions are still immature, and the standardized V-PCC is still widely used. There has also been some work conducted with respect to the improvement of the V-PCC standard [44].



(a) V-PCC projection onto "bounded box" planes. Image from [45]



(b) V-PCC patch packing onto a 2D grid. Example of geometry (left) and texture (right) images. Image from [46].

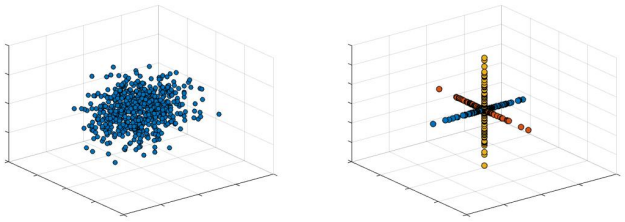
Fig. 3. V-PCC: Example of 3D-to-2D projection. Although the figures show both geometry and texture information, in our work we are only concerned with geometry artifact removal.

III. System Model

As described earlier, the quantization noise from V-PCC coding can result in compression artifacts that can cause considerable degradation to the quality of the reconstructed point cloud. Our goal is to perform deep learning-based geometry artifact removal. We make our framework scalable to larger point clouds. We propose a sampling and aggregation scheme in which the point cloud is divided into smaller patches, and then these patches are passed through a geometry compression artifact removal network. Afterward, we combine the artifact-removed patches to form an artifact-removed point cloud. The system model is shown in Fig. 2. Taking advantage of the sparse nature of point clouds, we employ the sparse convolutional network [22] which uses submanifold sparse convolutions [47] for our 3D learning.

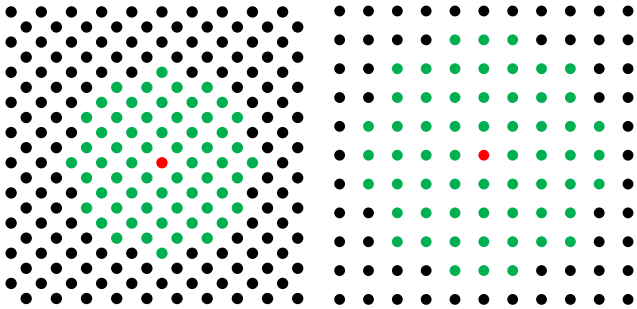
A. Problem Formulation

V-PCC attempts to leverage existing video compression codecs for point cloud geometry and texture compression. V-PCC converts the point cloud into a set of different video sequences, one for geometry and one for texture information. In our work, we are only concerned with the noise introduced in the geometry compression. The video-generated bitstreams and the metadata needed to interpret these videos are then



(a) Gaussian Noise (b) V-PCC Quantization Noise

Fig. 4. Comparison of two different types of noises in a point cloud.



(a) Dense surface (b) Sparse surface

Fig. 5. Patch correspondence mismatch problem for $k = 61$. The k-NN search covers a smaller surface on a denser point cloud as compared to a sparser point cloud.

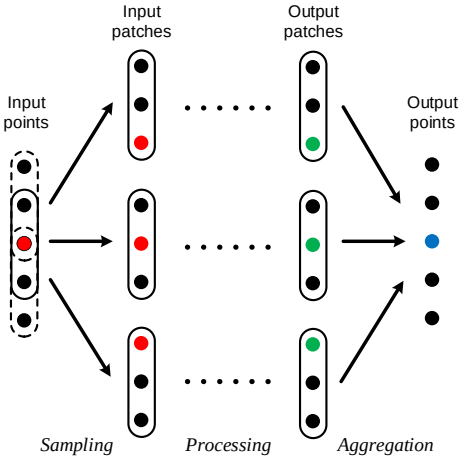


Fig. 6. An example sampling and aggregation scheme. The red point is the input point that is sampled into three patches. These three patches are processed to obtain three processed green points. The three green points are then aggregated to form the blue output point.

multiplexed together to generate the final point cloud V-PCC bitstream. V-PCC maps the input point cloud to a regular 2D grid by first decomposing the point cloud into a set of patches and then mapping these patches independently to a 2D grid using orthogonal projection. This process is shown in Fig. 3a. V-PCC iteratively divides the point cloud into smaller patches to avoid auto-occlusions and to generate patches with smooth boundaries. To generate these patches, the normal for each point is first estimated. An initial clustering is obtained by associating each point to one of the six cube-oriented planes. More precisely, each point is associated with the plane that has the closest normal (i.e., the dot product of the point normal and the plane normal is maximum). This initial clustering is then refined by iteratively updating the cluster index by taking into account the point's normal and the neighboring point's cluster index. In this way, all the points in a refined patch are associated with a single plane. The majority of the points in the point cloud are projected to the cube plane that is closest to the normal of that point. This projection is only along one of the axes $(x, y,$ or $z)$. These patches are then projected onto the 2D grid using a process called packing to obtain a frame with texture and another with geometry. The final video sequence frames for geometry and texture are shown in Fig. 3b.

Afterward, these projected video frames are encoded by leveraging video compression techniques. Since these compression techniques are lossy, compression artifacts are often introduced due to quantization noise affecting the point cloud geometry. However, the artifact noise introduced in V-PCC geometry is only in one direction, as shown in Fig. 4. This is because each point is projected to only one plane, and therefore the artifact noise in that point is only in the direction of that plane. ***This means that quantization noise introduced in each point is only along one of the axes (x, y, or z)***. We leverage this property to learn the quantization noise level and quantization noise direction introduced by the V-PCC codec in the point cloud geometry. Since the quantization noise is along one of the axes, we make sure that our learned quantization noise for each point is also along a single axis. We exploit this prior knowledge of quantization noise direction to limit the degree of freedom of the learned quantization noise. We use the learned quantization noise to remove geometry artifacts from the reconstructed point cloud and improve its PSNR.

B. Sampling

The size of a point cloud can vary greatly, from point clouds with only a few thousand points to point clouds with millions of points. We propose a patch-based sampling and aggregation scheme to make our framework scalable to all sizes of point clouds. We sample a large point cloud into smaller neighborhood patches to ensure efficiency for practical application by offering affordable memory consumption on a cube basis, along with parallel processing.

For each extracted patch, we need to find the patch in the original/ground-truth point cloud and the corresponding patch in the V-PCC reconstructed/noisy point cloud. A lossy low bitrate V-PCC reconstructed point cloud usually has fewer points than the original point cloud, consequently making the reconstructed point cloud sparser. Traditional patch-based point cloud deep learning models employ k-nearest neighbor (k-NN) search algorithms to obtain patches. However, when the number of points in the reconstructed point cloud differs from the number of points in the original point cloud, the k-NN search for extracting a neighborhood patch would not work because it would result in a different patch surface area for the two point clouds. Consider the example of $k = 61$:

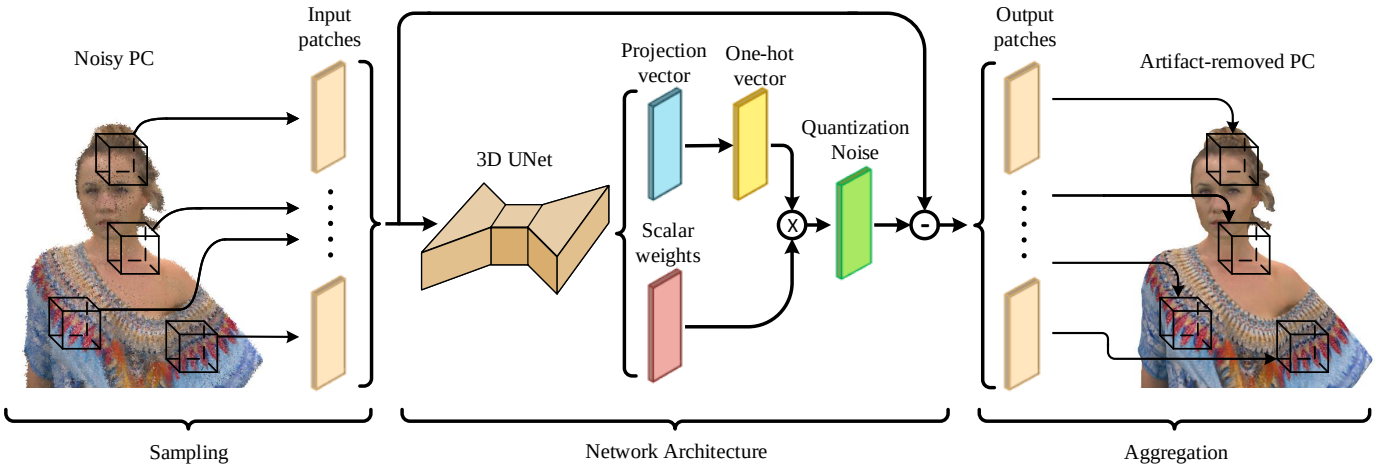


Fig. 7. Overview of the proposed point cloud artifact removal scheme. The input point cloud is divided into smaller patches that are fed into a sparse U-Net, which produces the projection vector and the scalar weights for each patch. The projection vector and the scalar weights are used to calculate the quantization noise that is then removed from the reconstructed point cloud patch. The output patches are then aggregated to obtain the artifact-removed point cloud.

using k-NN to extract a patch of 61 points would occupy a much larger area in a sparser point cloud as compared with a dense point cloud. We call this the *patch correspondence mismatch problem* and show an example of it in Fig. 5.

To solve the *patch correspondence mismatch problem*, we propose a cube-centered neighborhood search algorithm wherein we extract all the points inside a fixed cube volume. We employ farthest point sampling (FPS) [48] to sample points on the noisy point cloud and then extract cube patches around the sampled points. We obtain the noisy point cloud patch and the associated ground truth patch of the same cube volume extracted from the same location from both point clouds.

Farthest point sampling is employed to sample N points over the point cloud, and a cube neighborhood around these points is used to extract neighborhood patches. N patches are extracted using the formula:

$$N = \frac{n * C}{k} \tag{1}$$

where n represents the total number of points in the point cloud, k is the approximate average number of points in the neighborhood patch, and C is a variable used to control the average number of overlapping patches per point. More points can be sampled by increasing the value of C, which would result in a larger average number of overlapping patches per point. Each of the sampled points is used as a center point for a cube and all the points inside the cube are extracted to form an input neighborhood patch. The geometry of the points inside the patch is zero-centered and then normalized between zero and the length of the cube side. These smaller input patches are fed to our 3D U-Net architecture as shown in Fig. 7. Depending on the value of C, each point is sampled into multiple input patches and processed to obtain output patches. Therefore, we obtain multiple processed points for each point. We employ an aggregation scheme to merge the output patches to obtain the final output point cloud. An example of this can be found in Fig. 6.

C. Aggregation

Once we have the artifact-removed output patches, we aggregate them back together to form the final point cloud. Post-processing is performed on the output patches for which the normalization is removed, and they are moved back to their original locations as shown in Fig. 7. Depending on the value of C, we obtain many overlapping patches: therefore, each point receives multiple geometry values from different patches. Each input point is sampled into multiple patches, resulting in multiple clean point outputs for each input point. The geometries of these clean output points are mean aggregated to obtain the final clean output point.

One example sampling and aggregation scheme is shown in Fig. 6. The number of input points is $n = 5$, the number of patches sampled is $N = 3$, and each patch is of size $k = 3$. For reference, we can examine the red input point. This input point is sampled into three patches and processed to obtain three processed green points. We then take the mean of the three green processed points to obtain the blue aggregated output point.

D. Network Architecture

As explained in the previous section, we employ a cube-based patch sampling algorithm, so the number of points in the input patch varies. Traditional deep-learning-based point cloud processing networks work on a fixed number of input points to the network. We propose a fully convolutional 3D network that functions for variable input patch size. We employ sparse convolutional networks to create a fully convolutional network that offers the advantage of using a different number of points per patch, making the cube neighborhood patch extraction viable. Recall that our goal is to take in a 3D input patch of a V-PCC reconstructed point cloud and learn the per point quantization noise. The output of our 3D deep learning architecture is per point scalar weights and a per point projection vector. We use these to calculate the quantization noise.

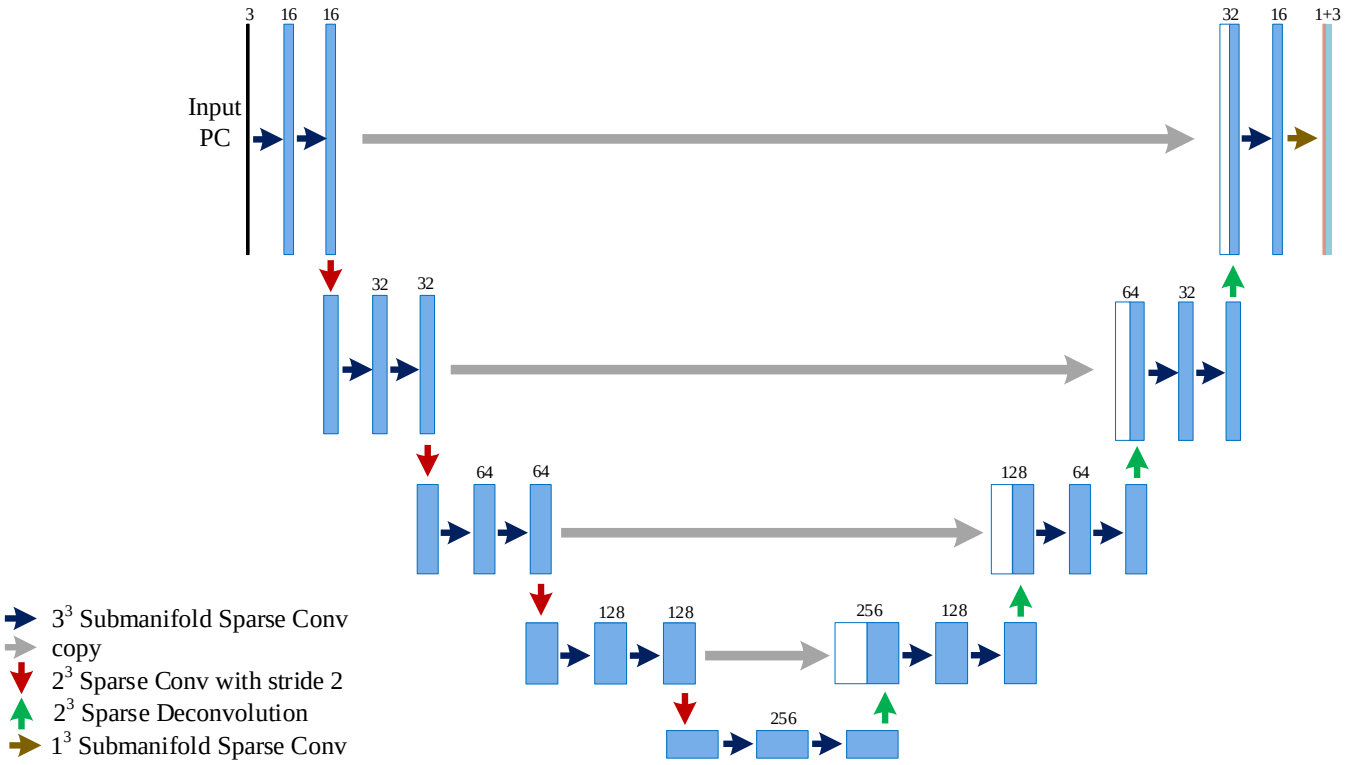


Fig. 8. 3D U-Net Network Architecture.

1) 3D U-Net: We employ submanifold sparse convolutions [47] that use a 3D sparse convolutional network [22]. Sparse convolutions exploit the sparse nature of a point cloud and are much more memory efficient. Furthermore, sparse submanifold convolutions make sure the network does not "dilate" the sparse data and maintain the same sparsity level throughout the network. This helps us build and train deeper architectures like U-Net [49].

U-Net architecture has been widely used in biomedical image segmentation tasks and usually employs 2D convolutions. We implemented a sparse convolution-based 3D U-Net architecture, the details of which are shown in Fig. 8. The architecture takes in a 3-dimensional geometry input patch, and the output is 4-dimensional: 3 dimensions for the projection vector and 1 dimension as a scalar weight. We use 3x3x3 (3^3) submanifold sparse convolutions in each layer. We employ 2x2x2 (2^3) convolutions with a stride of 2 for each *downsampling*, whereas 2x2x2 (2^3) deconvolution is used for *upsampling*. The U-Net architecture typically consists of two paths: the encoder path and the decoder path. The encoder path captures the context of the point cloud, producing feature maps using strided convolutions to downsample. In the encoder path, with each downsampling, the number of points decreases but the feature dimension is doubled. The decoder path employs 3D deconvolution to upsample the point cloud. U-Net combines the information from the encoder path with that of the decoder path to obtain both the contextual information and localized information. U-Net only contains convolutional layers and does not employ any dense layers, making the network fully convolutional. This offers the added advantage of working with patches of variable input size.

2) Quantization Noise Calculation: The projection vector yields the direction of the quantization noise, whereas the scalar weight provides the level of the quantization noise. In V-PCC, a point is typically projected in one direction (x, y, or z); therefore, the quantization noise for each point is in a specific direction. Hence, it makes more sense to use a one-hot encoded projection vector for each point. We convert the projection vector into a one-hot vector by performing one-hot encoding using the maximum of the projection vector:

$$Z_i(j) = \begin{cases} 1 & \text{if } j = \underset{j}{\operatorname{argmax}}\, V_i(j) \\ 0 & \text{else} \end{cases} \tag{2}$$

Where i is the point number, j is the dimension, i.e. $j \in \{x, y, z\}$-axis, and $Z_i(j)$ is the one-hot vector, whereas $V_i(j)$ is the projection vector.

Once we have the one-hot vector, we multiply it with the scalar weight to learn the quantization noise. The per point quantization noise is then removed from the input patch to obtain an artifact-removed output patch. This is also illustrated in Fig. 7.

3) Loss Function: Our loss function is calculated by comparing the artifact-removed output patch to the ground truth patch. The loss function is applied to each patch before

aggregation. We use Chamfer distance as the loss function in our architecture:

$$L_{CD}(P_O, P_G) = \sum_{x \in P_O} \min_{y \in P_G} ||x - y||_2^2 + \sum_{y \in P_G} \min_{x \in P_O} ||x - y||_2^2 \quad (3)$$

Where L_{CD} is the Chamfer distance loss function, P_G is the ground truth patch and P_O is the output artifact-removed patch calculated using input patch, projection vector, and scalar weights. Intuitively, the first term measures an approximate distance between each output point to the target surface, whereas the second term rewards even coverage of the output point cloud and penalizes any gaps.

IV. Simulation Results

We perform extensive simulations and show both the objective results and the visual comparison of our framework. Since this is the first work on V-PCC artifact removal, we have nothing with which to compare our work. However, we do show considerable improvement in the quality of the point cloud and perform multiple ablation studies to gain insight into the problem and explore alternative methods. For our simulation, we use the values of $k = 10000$ and $C = 20$.

A. Dataset

We use the 8i voxelized full bodies dataset from 8i labs [6], which contains up to one million points per point cloud and is widely used by MPEG. The 8i dataset includes multiple sequences of point clouds. Each sequence has multiple point clouds representing 10 seconds of video captured at 30 fps for a total of 300 frames. We use two sequences for training (*longdress, loot*) and three sequences for testing (*redandblack, soldier, queen*). We use three different bitrates to encode these point clouds using V-PCC, shown in Table I. We label these bitrates as $br1$, $br2$, and $br3$ from the highest bitrate to the lowest bitrate, respectively.

TABLE I
Different bitrates used in simulation

Bitrate label	Actual bitrate
br1	0.01866 bpp
br2	0.01632 bpp
br3	0.01502 bpp

B. Objective Evaluation

We use mean-squared-error (MSE) point-to-point PSNR (dB) as well as point-to-point Hausdorff PSNR (dB) as our objective metrics, which are calculated using MPEG's PC error tool [50]. We refer to MSE PSNR as simply PSNR in the rest of the section. We also measure the BD-Rate (Bjöntegaard Delta Rate) [51] improvement to determine how much savings our method achieves. The PSNR of the reconstructed point clouds obtained from the V-PCC encoder is measured before and after our artifact removal technique. The results are shown in Table II. As can be determined for all three bitrates, our artifact removal technique considerably improves the PSNR as well as Hausdorff PSNR of the reconstructed point cloud.

We follow the MPEG common test condition to calculate the BD-rate using the PSNR metric. We compute the point-to-point distance for each frame of a sequence and obtain a total score by averaging across all frames. The final objective score was obtained by averaging across all three test sequences. We obtain a BD-rate savings of 11.3%. We also show the PSNR results for each 8i point cloud sequence separately for each bitrate in Table III. We can also observe PSNR improvement for the bitrates for each point cloud sequence. From the results, we observe that higher PSNR improvement is achieved at a lower bitrate. This is because lower bitrates suffer from higher quantization noise, which our system can efficiently remove.

TABLE II
Average PSNR results tested on three 8i sequences

	PSNR (dB)		Hausdorff PSNR (dB)	
Bitrate	**Noisy PC**	**Cleaned PC**	**Noisy PC**	**Cleaned PC**
br3	59.62	60.47	37.02	37.21
br2	61.84	62.36	39.58	39.64
br1	64.20	64.53	41.15	41.20
BD-rate savings:			11.3 %	

TABLE III
Simulation Results For Each Individual Sequence

		PSNR (dB)		
Test PC	**Bitrate**	**Noisy PC**	**Cleaned PC**	**Improvement**
Queen	br3	60.23	60.79	0.56
	br2	62.35	62.90	0.55
	br1	64.94	65.21	0.27
RedAndBlack	br3	59.44	60.38	0.94
	br2	61.62	62.18	0.56
	br1	63.90	64.27	0.37
Soldier	br3	59.20	60.25	1.05
	br2	61.53	62.01	0.48
	br1	63.76	64.12	0.36

C. Visual Results

Visual results for point cloud artifact removal are shown in Fig. 9 and Fig. 10 for two different sequences: *queen* and *soldier*. We show the original (ground truth) point cloud, V-PCC reconstructed point cloud, and the artifact-removed point cloud for three different bitrates of V-PCC encoding. To visualize the point cloud, we first compute the normals for each point using 100 neighboring points; then, we set the shading to vertical and view the point cloud as a mesh. In this way, we can observe the point cloud geometry, which is more intuitive than vertex-color rendered images. However, since we use normals to visualize the point cloud, some points might appear black due to flipped normals, as shown from the right foot in Fig. 9. We also plot the error map based on the point-to-point (P2point) D1 distance between decoded point clouds and ground truth to visualize the error distribution.

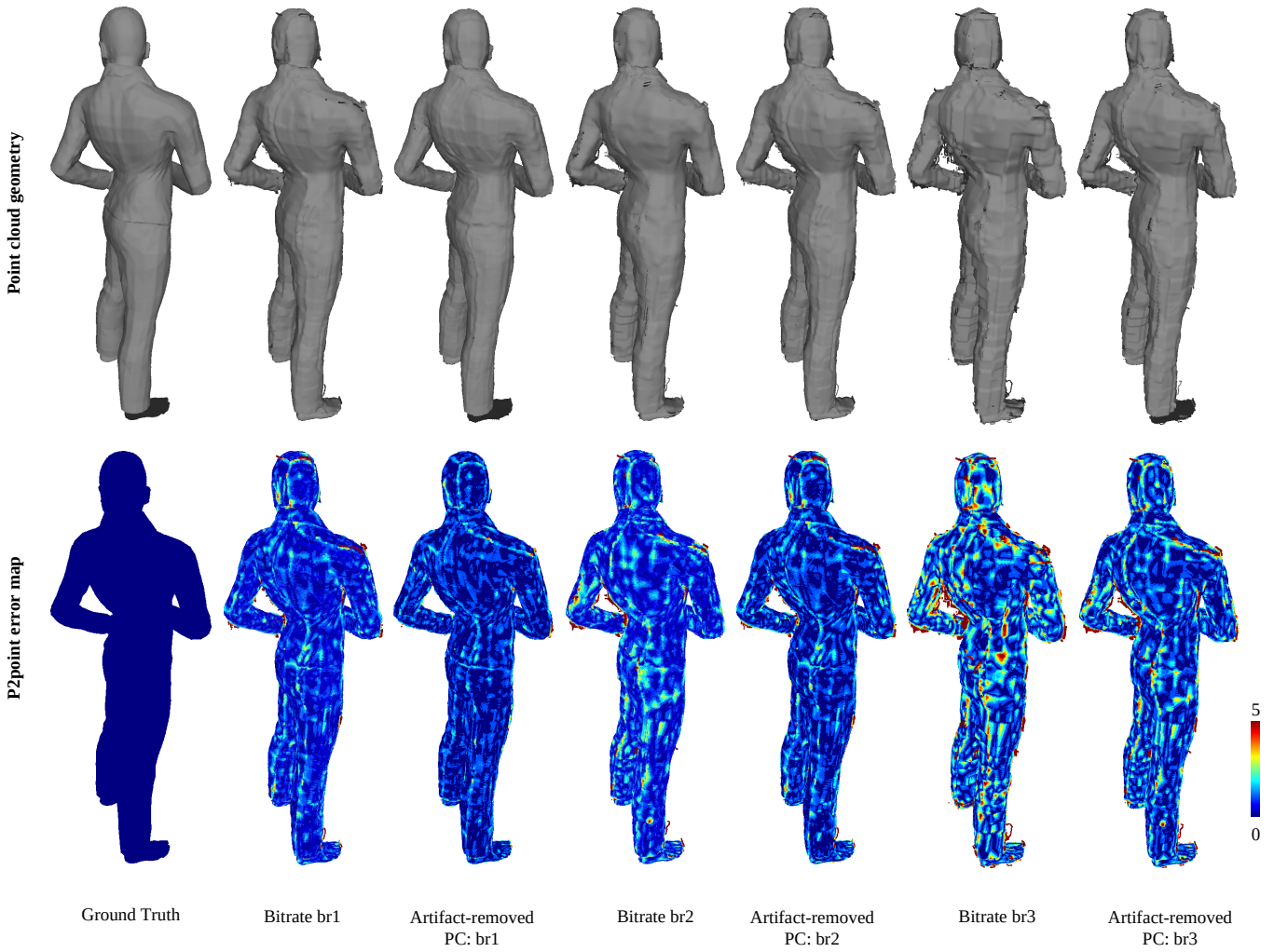


Fig. 9. Visual comparison of 'queen' showing ground truth, three different V-PCC reconstructed point clouds, and their corresponding artifact-removed point cloud. We show the geometry as well as point-to-point error map.

We can see that our method improves the quality of the point cloud, especially on the edges and surface of the point cloud. Although V-PCC performs well in quantitative objective comparison, its reconstructed point clouds contain noticeable artifacts when the bitrate is low. Our method removes these artifacts and considerably improves the visual quality of the point cloud. An interesting observation is that our artifact-removed point cloud compensates for some broken parts in the V-PCC reconstructed point cloud.

D. Quantization Noise Calculation

In this section, we compare our quantization noise calculation method with alternative methods. We use the V-PCC encoded bitrate of $br3$ for this experiment. Our current structure outputs 1-dimensional scalar weights and a 3-dimensional projection vector. We convert the projection vector to a one-hot encoded vector and multiply it by the scalar weights to calculate the quantization noise. After removing the quantization noise from the input patch, Chamfer loss is used to train the network. We label this **Our Method** and compare it with two alternative methods. **Method 1:** The network outputs 3-dimensional quantization noise that is directly removed from the input patch without any post-processing, and then Chamfer loss is used to train the network. **Method 2:** The network outputs 1-dimensional scalar weights and a 3-dimensional projection vector. Projection vectors are directly multiplied by the scalar weights to find quantization noise without converting them to a one-hot vector first. After removing the quantization noise from the input patch, Chamfer loss is used to train the network.

To summarize, in *Method 1* the network directly outputs the quantization noise, whereas in *Method 2* we remove the one-hot encoding part from our original architecture. The results of *Our Method*, *Method 1*, and *Method 2* are compared in Table IV.

The results of Method 1 show that learning quantization noise directly from the network yields poor results. Similarly, comparison of our method with Method 2 shows that converting the projection vector to a one-hot vector before calculating the quantization noise considerably improves our results. This also shows that utilizing the prior knowledge of quantization noise which is only in the direction of the

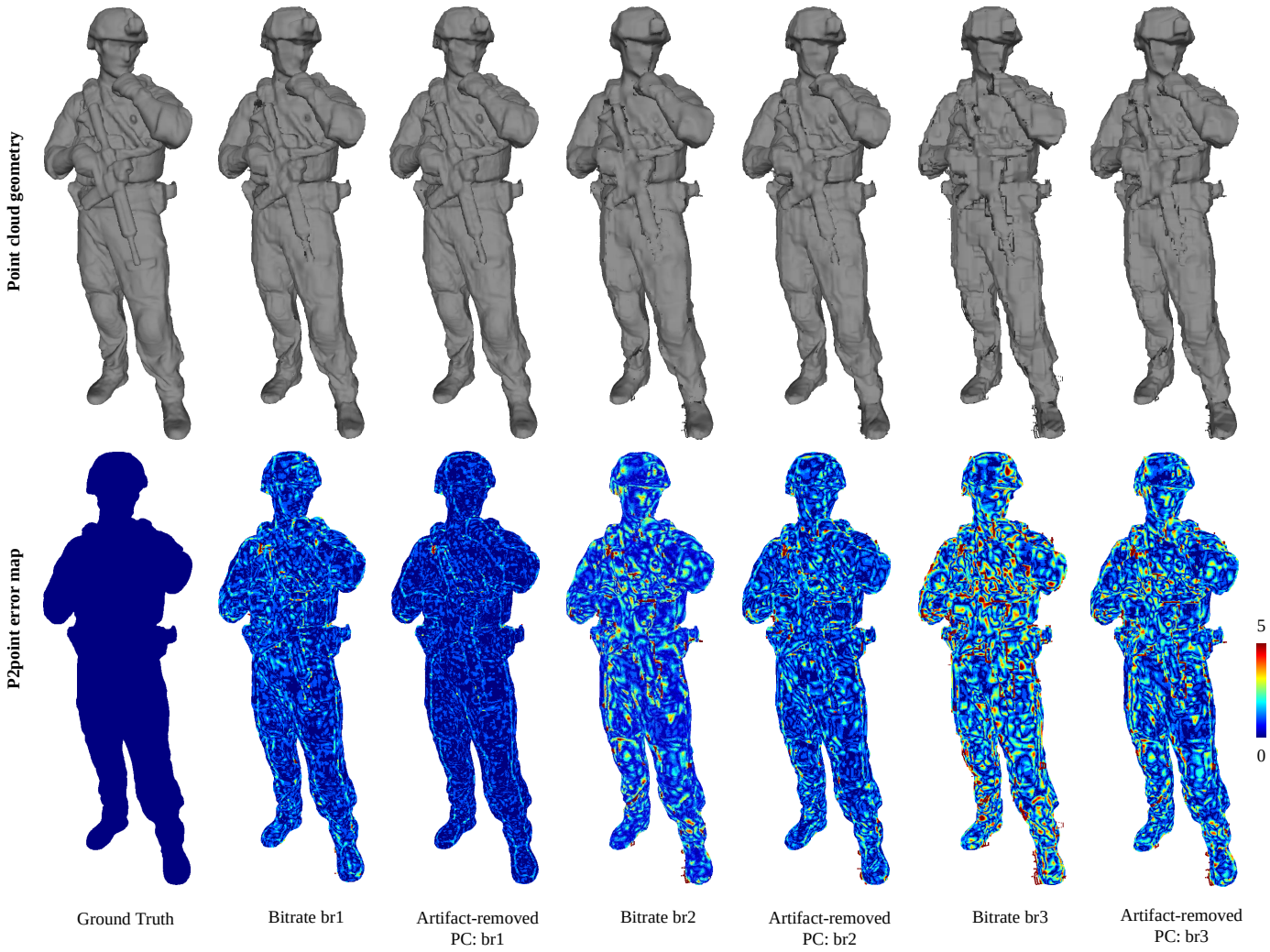


Fig. 10. Visual comparison of 'soldier' showing ground truth, three different V-PCC reconstructed point clouds, and their corresponding artifact-removed point clouds. We show the geometry as well as point-to-point error map.

TABLE IV
DIFFERENT QUANTIZATION NOISE CALCULATION METHODS

		PSNR (dB)		
Test PC	**Noisy PC**	**Our Method**	**Method 1**	**Method 2**
Queen	60.23	**60.79**	60.35	60.41
RedAndBlack	59.44	**60.38**	59.92	60.02
Soldier	59.20	**60.25**	59.76	59.93
Average	59.62	**60.47**	60.01	60.12

projection is beneficial to the learning of quantization noise.

E. Sampling and Aggregation Schemes

Currently, we employ the Farthest Point Sampling technique to sample points and extract a neighborhood around these points using cube extraction. Since there are overlapping neighborhood patches, we perform a mean aggregation scheme to obtain the final artifact-removed point cloud. We compare our method to a non-overlapping **octree-based** [52] cube division method. In the octree-based method, the point cloud is partitioned into cube nodes, and artifact removal is applied to each node. The differences between our method and the octree-based method are: 1) the patch sampling is performed using an octree; 2) there is no aggregation scheme, since the sampled patches are non-overlapping. The results of the comparison are shown in Table V. We can observe from the results that our overlapping cubes-based sampling method substantially outperforms the octree-based sampling method.

TABLE V
DIFFERENT SAMPLING AND AGGREGATION SCHEMES

		PSNR (dB)	
Test PC	**Noisy PC**	**Our Method**	**Octree-based**
Queen	60.23	**60.79**	60.38
RedAndBlack	59.44	**60.38**	59.97
Soldier	59.20	**60.25**	59.80
Average	59.62	**60.47**	60.05

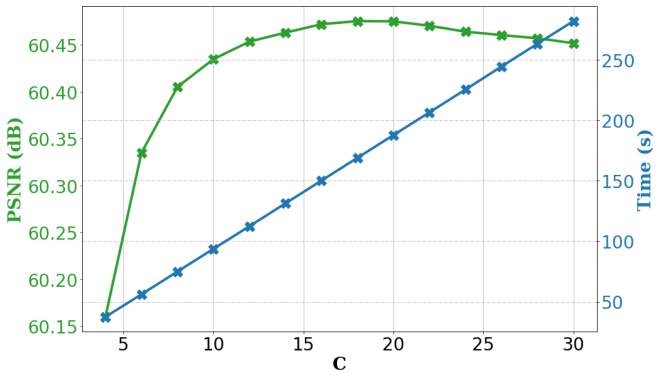


Fig. 11. PSNR (dB) and Time (s) complexity for different values of C.

F. Choosing the value of C

As described earlier, C is the variable used to control the average number of overlapping patches per point. A higher value of C would result in a larger number of randomly sampled patches. To further study our sampling scheme, we perform a hyperparameter optimization experiment for the parameter C. We perform the simulation on the three test sequences of our 8i dataset (*redandblack, soldier, queen*) and then plot the combined results. We vary the value of C and measure the PSNR results as well as the computation time. Results of this experiment are shown in Fig. 11. As can be observed, the PSNR increases with the value of C: PSNR is maximal at $C = 18$ and starts to slightly decrease after that. The computation time is calculated as the average time to process a single 8i point cloud on an NVIDIA GeForce GTX 1080 Ti GPU. The computation time includes sampling, forward propagation through the network, and aggregation. As Fig. 11 shows, the computation time increases linearly with the value of C.

V. Conclusion

V-PCC encoding is the current state-of-the-art method for dynamic point cloud compression that has been selected by MPEG to be developed into a standard. However, quantization noise during V-PCC encoding results in severe quality degradation because it introduces compression artifacts. This paper presents a first-of-its-kind deep learning-based point cloud geometry compression artifact removal framework for V-PCC encoded dynamic point clouds. We leverage the prior knowledge that during V-PCC, the quantization noise is introduced only in the direction of the point cloud projection. We employ a 3D sparse convolutional neural network to learn both the direction and the magnitude of geometry quantization noise. To make our work scalable, we propose a cube-centered neighborhood extraction scheme with a sampling and aggregation method to extract small neighborhood patches from the original point cloud. These patches are passed through the network to remove compression artifacts and then aggregated to form the final artifact-removed point cloud. Experimental results show that our method considerably improves the V-PCC reconstructed point cloud's geometry quality in both objective evaluations and visual comparisons.

References

[1] C. Tulvan, R. Mekuria, Z. Li, and S. Laserre, "Use cases for point cloud compression (PCC)," 2016.
[2] H. Fuchs, A. State, and J.-C. Bazin, "Immersive 3D telepresence," *Computer*, vol. 47, no. 7, pp. 46–52, 2014.
[3] A. Akhtar, J. Ma, R. Shafin, J. Bai, L. Li, Z. Li, and L. Liu, "Low latency scalable point cloud communication in vanets using v2i communication," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.
[4] Mobile Mapping System. (2019). [Online]. Available: http://www.mitsubishielectric.com/bu/mms/index.html
[5] 6DoF vs 3DoF. (2014). [Online]. Available: https://packet39.com/blog/2018/02/25/3dof-6dof-roomscale-vr-360-video-and-everything-in-between/
[6] M. Krivokuća, P. Chou, and P. Savill, "8i voxelized surface light field (8iVSLF) dataset," in *ISO/IEC JTC1/SC29/WG11 MPEG, input document m42914*, 2018.
[7] 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 Transactions on Signal and Information Processing*, vol. 9, 2020.
[8] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.
[9] "Call for proposals for point cloud compression (v2)," 2017, document ISO/IEC JTC1/SC29/WG11 MPEG, N 16763, 3D Graphics, Apr. 2017.
[10] "G-PCC test model v7 user manual," iSO/IEC JTC1/SC29/WG11 MPEG output document N18664.
[11] "V-PCC codec description," 2019, iSO/IEC JTC1/SC29/WG11 N18673, Aug, 2019.
[12] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.
[13] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo, "Deep generative adversarial compression artifact removal," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4826–4835.
[14] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4539–4547.
[15] G. Lu, W. Ouyang, D. Xu, X. Zhang, Z. Gao, and M.-T. Sun, "Deep kalman filtering network for video compression artifact reduction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 568–584.
[16] Z. Guan, Q. Xing, M. Xu, R. Yang, T. Liu, and Z. Wang, "Mfqe 2.0: A new approach for multi-frame quality enhancement on compressed video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
[17] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," *arXiv preprint arXiv:1912.12033*, 2019.
[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.
[20] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.
[21] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in neural information processing systems*, 2018, pp. 820–830.
[22] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.
[23] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.
[24] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "Pointhop: An explainable machine learning method for point cloud classification," *IEEE Transactions on Multimedia*, 2020.

[25] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized representations of point clouds with graph-convolutional generative adversarial networks," *IEEE Transactions on Multimedia*, 2020.

[26] G. Guennebaud and M. Gross, "Algebraic point set surfaces," in *ACM SIGGRAPH 2007 papers*, 2007, pp. 23–es.

[27] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 493–501.

[28] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, pp. 22–es, 2007.

[29] E. Mattei and A. Castrodad, "Point cloud denoising via moving rpca," in *Computer Graphics Forum*, vol. 36, no. 8. Wiley Online Library, 2017, pp. 123–137.

[30] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on image processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[31] C. Dinesh, G. Cheung, and I. V. Bajic, "3d point cloud denoising via bipartite graph approximation and reweighted graph laplacian," *arXiv preprint arXiv:1812.07711*, 2018.

[32] X. Gao, W. Hu, and Z. Guo, "Graph-based point cloud denoising," in *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*. IEEE, 2018, pp. 1–6.

[33] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799.

[34] ——, "Ec-net: an edge-aware point set consolidation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 386–402.

[35] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7203–7212.

[36] Y. Qian, J. Hou, S. Kwong, and Y. He, "Pugeo-net: A geometry-centric network for 3d point cloud upsampling," *arXiv*, pp. arXiv–2002, 2020.

[37] Z. Fu, W. Hu, and Z. Guo, "Point cloud inpainting on graphs from non-local self-similarity," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2137–2141.

[38] Y. Yu, Z. Huang, F. Li, H. Zhang, and X. Le, "Point encoder gan: A deep learning model for 3d point cloud inpainting," *Neurocomputing*, vol. 384, pp. 192–199, 2020.

[39] W. Hu, Z. Fu, and Z. Guo, "Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 4087–4100, 2019.

[40] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned point cloud geometry compression," *arXiv preprint arXiv:1909.12037*, 2019.

[41] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Point cloud compression for 3D LiDAR sensor using recurrent neural network with residual blocks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3274–3280.

[42] T. Huang and Y. Liu, "3d point cloud geometry compression on deep learning," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 890–898.

[43] K. Matsuzaki and K. Tasaka, "Binary Representation for 3D Point Cloud Compression based on Deep Auto-Encoder," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2019, pp. 489–490.

[44] L. Li, Z. Li, S. Liu, and H. Li, "Efficient projected frame padding for video-based point cloud compression," *IEEE Transactions on Multimedia*, 2020.

[45] "Coding of moving pictures and audio," document ISO/IEC JTC1/SC29/WG11 w18892.

[46] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[47] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[48] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[49] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

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

[51] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

[52] A. Akhtar, B. Kathariya, and Z. Li, "Low latency scalable point cloud communication," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 2369–2373.