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 capturing and transmission are the fundamental enablers for immersive visual communication. The coding process of dynamic point cloud, especially the Video-based Point Cloud Compression (V-PCC) developed by the MPEG standardization group is now delivering the state of the art performance in compression efficiency. V-PCC is based on 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 the quality of reconstruction without additional bandwidth cost. We propose a novel framework that consists of a point cloud sampling scheme with a cube-based neighborhood patch extraction, these patches are then passed through an artifact removal network followed by an aggregation scheme to merge the overlapping output patches to obtain an artifact-removed point cloud. We employ a 3D deep convolutional feature learning for geometry artifact removal that jointly recovers both the quantization direction as well as the level of the quantization noise by exploiting projection and quantization prior. 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 lead to a surge in the usage of 3D point clouds in Virtual/Augmented reality (VR/AR) content creation and communications [1], as well as 3D sensing for robotics, smart city, telepresence [2], and automated driving applications [3]. A 3D point cloud is able to represent volumetric visual data such as 3D scenes and objects efficiently using a collection of discrete points with 3D geometry positions and other attributes (e.g., color, reflectance, etc). Point cloud data has advantages over polygonal meshes as it is more flexible, and has real-time processing potential as there is no need to process, store, and transfer surface topological information. With an increase in the applications of point cloud and an improvement in the capturing technologies, we now have very high resolution point clouds with millions of points per frame.

Based on their applications, 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 on top of 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/MR, volumetric video, and telepresence and can be generated using 3D models, i.e. CGI, or captured from real-world scenes using a variety of methods such as multi-cameras with depth sensors surrounding the object and capturing the movement over time.

A volumetric video such as dynamic point cloud gives us an immersive media experience. Dynamic point cloud describes a 3D object using its geometry, respective attributes, as well as any temporal changes. Temporal information in 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 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 a million points. About 30 bits are used to represent the geometry (x,y,z) and 24 bits for color (r,g,b). The size of a single instance can be approximated as 6Mbytes, which for a 30 frames per second dynamic point cloud translates to a bitrate of 180Mbytes per second without compression. The high data rate is one of the main problems faced by dynamic point cloud and efficient compression technologies to allow for the distribution of such content are still sought after.

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 in standardization by MPEG for dynamic point cloud. Under the V-PCC standard, a point cloud is first projected onto its bounding box patch by patch. Then, the patches are packed into a video for compression. During the video compression, the reconstructed geometry may suffer serious quality degradation due to the quantization errors. Blocking artifacts or compression artifacts are often introduced in compressed...
media due to distortion introduced in the lossy compression techniques [8]. The V-PCC coded point cloud yields excellent reproduction without noticeable artifacts at high or moderate bit rates. However, at low bit rates, the reconstructed point cloud suffers from visually annoying artifacts as a result of coarse quantization. Fig. 1 shows two versions of a point cloud encoded at different bit-rates. As can be seen, there is no visible blocking artifact in the point cloud coded at a high bit rate, while severe blocking artifacts exist in the one coded at a lower bit rate. Since blocking artifacts significantly degrade the visual quality of the reconstructed point cloud, it is desirable to be able to identify these artifacts and remove them from the reconstructed point cloud.

In this paper, we propose the first deep-learning-based geometry artifacts removal algorithm for the V-PCC standard for dynamic point cloud. Ours is a pioneer, first of its kind, work on V-PCC artifact removal. The proposed framework has 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 V-PCC reconstructed point cloud both in the subjective as well as objective evaluations.

- 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 as well as 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 metric.

- 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, where MPEG has been evaluating and improving the performances of the proposed technologies since then. 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) for dynamic content [11]. G-PCC employs octree in its coding scheme, whereas, V-PCC projects point cloud 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, especially 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, as well as deblurring, has been extensively studied in image and video coding. Since V-PCC also employs state-of-the-art HEVC video coding, there is a potential to learn from the video compression artifact removal techniques and use them for V-PCC artifact removal. Currently, the state-of-the-art compression artifact removal techniques are all deep learning based. There has been work using residual networks [12], GANs [13], as well as work employing memory-based deep learning architecture [14]. All these works are limited to a single image at a time and do not use information from previous frames. Recent works, however, have exploited temporal information in restoration tasks to improve video compression artifact removal [15], [16].

Deep learning on point cloud has been attracting more and more attention, especially in the last five years [17]. PointNet [18] was among the earliest deep learning architectures for point cloud learning which employed pointwise fully connected layer followed by max pooling. This architecture was further improved into PointNet++ [19] by adding hierarchical learning that could learn local features with better contextual scale. Wang et al. [20] proposed an Octree-based CNN for 3D
shape classification that used convolutional on the nodes of an octree. PointCNN [21] achieved state-of-the-art results using convolutional neural network on 3D point cloud without any preprocessing. It employed a fully connected layer to learn a transformation matrix to make the input permutation invariant which made convolution possible on them. SparseConvNet [22] by Facebook was among the first sparse convolutional neural networks that achieved state-of-the-art results on point cloud. SparseConvNet introduced submanifold sparse convolutions that exploited the sparse nature of point cloud and made sure the convolutions wouldn’t “dilate” the data. Since sparse convolutions are memory efficient and the data remains sparse throughout, this enables deep neural network architectures to be used for point cloud. 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 since the early ‘90s. 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 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 on synthetic noise (e.g., Gaussian noise) and some are even tested on real-world noise that is introduced during point cloud capture. However, these methods are not optimized to work on the compression artifact removal because of the nature of the quantization noise introduced during V-PCC.

Similarly, there has been some work focusing on point cloud inpainting [37], [38], [39] where portions of point cloud lost during point cloud capturing are completed. However, these methods do not work for compression artifact removal in V-PCC. In the last year, there has been some work 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 being widely used. There has also been some work done on the improvement of V-PCC standard
to a 2D grid using orthogonal projection. This is shown in Fig. 3a. V-PCC iteratively divides the point cloud into smaller patches to avoid auto-occlusions and 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., maximizes the dot product of the point normal and the plane normal). This initial clustering is then refined by iteratively updating the cluster index by taking into account the point's normal as well as the neighboring point's cluster index. 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 axis \((x, y, \text{or } z)\). These patches are then projected onto the 2D grid using a process called packing. The final video sequence frame for texture is shown in Fig. 3b.

Afterward, these projected video frames are encoded by leveraging video compression techniques. Since this compression technique is lossy, compression artifacts are often introduced due to quantization noise. However, the artifact noise introduced in V-PCC is only in one direction as shown in Fig. 4. This is because each point is projected to only one plane, 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 axis \((x, y, \text{or } z)\). We leverage this property to learn both the quantization noise level and quantization noise direction introduced by the V-PCC codec. Since the quantization noise is along one of the axis, we make sure that our learned quantization noise for each point is also along a single axis. We exploit the 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 a lot, from point clouds with few thousand points to point clouds with millions of points. To make our framework scalable to all sizes of point clouds, we propose a patch-based sampling and aggregation scheme. To feed a large point cloud to the network, it needs to be sampled to smaller neighborhood patches. This also ensures the complexity efficiency for practical application, by offering affordable memory consumption on a cube basis as well as parallel processing.

For each individual 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 reconstructed point cloud tends to have a fewer number of points compared to 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 and the original point cloud is different, the k-NN search for extracting a neighborhood patch would not work since it would give us different patch surface area for the reconstructed point cloud and original point cloud. E.g. for \(k = 61\); using k-NN search to extract 61 neighboring points from the same point location to form a patch would occupy a much larger area in a sparser point cloud compared to a dense point cloud. We call this patch correspondence mismatch problem and show an example of it in Fig. 5.

To solve the patch correspondence mismatch problem, we instead propose a cube-centered neighborhood search algo-
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 UNet 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.

rithm where we extract all the points inside a fixed cube volume. We employ farthest point sampling to sample points on the noisy point cloud and then extract cube patches around the sampled points. We get the noisy point cloud patch as well as 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 smaller neighborhood patches using the formula:

$$N = \frac{n \times C}{k}$$

Where $n$ is 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 that is used to control the average number of overlapping patches per point. If we want more number of points sampled, we would have to increase $C$ 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 cube are normalized between zero and the length of the cube side. These smaller input patches are fed to our 3D UNet 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, for each point, we obtain multiple processed points. We employ an aggregation scheme to merge the output patches to obtain the final output point cloud. An example of this can be seen in Fig. 6.

C. Aggregation

Once we have the artifact-removed output patches, we aggregate them back together to form the final point cloud. The location of each patch in the point cloud is saved before they are normalized. Post-processing is performed on the output patches where the normalization is removed and they are moved back to their original location as shown in Fig. 7. Depending on the value of $C$, we get a lot of overlapping patches; therefore, each point gets multiple geometry values from different patches. Since for each point, we get multiple clean points from overlapping patches we have to find a way to aggregate them. We employ mean aggregation, i.e., take the mean of overlapping patch output results to obtain the final artifact-removed point cloud.

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

D. Network Architecture

As explained in the section before, we employ a cube-based patch sampling algorithm so the number of points in the input patch is variable. 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 works on variable input patch size. We employ sparse convolutional networks to create a fully convolutional network that gives us the advantage of using a different number of points per patch; therefore 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 per point quantization noise. Our 3D deep learning architecture outputs per point scalar weights, and a per point projection vector that we use to calculate the quantization noise.

1) 3D U-Net: Due to the inherent sparsity of 3D point clouds, we employ submanifold sparse convolutions [46] that uses 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 doesn’t “dilate” the sparse data and keeps the same level of sparsity throughout the network. This helps us build and train deeper architecture like U-Net [47].

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 projection vector, and 1 dimension as a scalar weight. We use $3 \times 3 \times 3$ ($3^3$) submanifold sparse convolutions in each layer. We employ $2 \times 2 \times 2$ ($2^3$) convolutions with a stride of 2 for each downsampling whereas $2 \times 2 \times 2$ ($2^3$) deconvolutional 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 get both the contextual information as well as localized information. U-Net only contains convolutional layers and does not employ any dense layers which make the network fully convolutional which has the added advantage of working with variable input size patches.

2) Quantization Noise Calculation: The projection vector gives us the direction of the quantization noise whereas the scalar weight gives us 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:

$$Z_i(j) = \begin{cases} 
1 & \text{if } j = \text{argmax}_j V_i(j) \\
0 & \text{else}
\end{cases} \quad (2)$$

Where $i$ is the point number, $j$ is the dimension i.e. $j \in \{x, y, z\}$-axis, $Z_i(j)$ 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 and not after the aggregation. We use chamfer distance as the loss function
in our architecture:

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

(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 an even coverage of the output point cloud and penalizes any gaps.

IV. SIMULATION RESULTS

We perform extensive simulations and show both the objective as well as subjective results of our framework. Since this is the first work on V-PCC artifact removal, we have not been able to compare it with other works. However, we do show considerable improvement in the quality of the point cloud and perform multiple ablation studies to learn 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 8i voxelized full bodies dataset by 8i labs [6] that have up to a million point per point cloud, and is widely used by MPEG. 8i dataset has multiple sequences of point clouds. Each sequence has multiple point cloud that represents a 10 seconds long video captured at 30 fps with 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 which are shown in Table I. We label these bitrates as \( br_1 \), \( br_2 \), and \( br_3 \) with the highest bitrate to the lowest bitrate respectively.

<table>
<thead>
<tr>
<th>Bitrate label</th>
<th>Actual bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>( br_1 )</td>
<td>0.01866 bpp</td>
</tr>
<tr>
<td>( br_2 )</td>
<td>0.01632 bpp</td>
</tr>
<tr>
<td>( br_3 )</td>
<td>0.01502 bpp</td>
</tr>
</tbody>
</table>

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 metric calculated using MPEG’s PC error tool [48]. We refer to MSE PSNR as simply PSNR in the rest of the section. We also measure the BD-rate improvement to see 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 seen for all three bitrates, our artifact removal technique considerably improves the PSNR of the reconstructed point cloud as well as improve the Hausdorff PSNR.

We follow the MPEG common test condition to calculate the BD-rate using PSNR metric. We compute the point-to-point for each frame of a sequence, and then 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 individual 8i point cloud sequence separately for each bitrate in Table III. We can observe PSNR improvement for the bitrates for each individual point cloud sequence too. 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 remove efficiently.

<table>
<thead>
<tr>
<th>Bitrate</th>
<th>PSNR (dB)</th>
<th>Hausdorff PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noisy PC</td>
<td>Cleaned PC</td>
<td>Noisy PC</td>
</tr>
<tr>
<td>( br_3 )</td>
<td>59.62</td>
<td>60.47</td>
</tr>
<tr>
<td>( br_2 )</td>
<td>61.84</td>
<td>62.36</td>
</tr>
<tr>
<td>( br_1 )</td>
<td>64.20</td>
<td>64.53</td>
</tr>
</tbody>
</table>

BD-rate savings: 11.3 %

C. Visual Results

Visual results for point clouds 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. This way, we are able to observe the point cloud geometry which is more intuitive than vertex-color rendered image. 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.

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 contains obvious artifacts when the bitrate is low. Our method greatly removes these artifacts and improves the visual quality of the point cloud considerably. An interesting observation is that our artifact-removed point cloud fills 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 V-PCC encoded bitrate of $br_3$ for this experiment. Our current structure outputs 1-dimensional scalar weights, and 3-dimensional projection vector. We convert the projection vector to a one-hot encoded vector and multiply it with 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 compare this method with two alternative methods. **Method 1:** The network outputs a 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 3-dimensional projection vector. Projection vectors are directly multiplied to 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 simply remove the one-hot encoding part from our original architecture. The results of these methods are compared and shown in Table IV.

<table>
<thead>
<tr>
<th>Test PC</th>
<th>Noisy PC</th>
<th>Our Method</th>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queen</td>
<td>60.23</td>
<td><strong>60.79</strong></td>
<td>60.35</td>
<td>60.41</td>
</tr>
<tr>
<td>RedAndBlack</td>
<td>59.44</td>
<td><strong>60.38</strong></td>
<td>59.92</td>
<td>60.02</td>
</tr>
<tr>
<td>Soldier</td>
<td>59.20</td>
<td><strong>60.25</strong></td>
<td>59.76</td>
<td>59.93</td>
</tr>
<tr>
<td>Average</td>
<td>59.62</td>
<td><strong>60.47</strong></td>
<td>60.01</td>
<td>60.12</td>
</tr>
</tbody>
</table>

The results of **Method 1** show that learning quantization noise directly from the network yields poor results. Similarly,
if we compare Our Method with Method 2, we can see that converting the projection vector to a one-hot vector before calculating the quantization noise considerably improves our results. This also shows that exploiting the prior knowledge that quantization noise is only in the direction of the projection helps in the calculating 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 this technique to a non-overlapping octree based [49] cube division of the original point cloud, followed by artifact removal of each node in the octree, and then putting the node patches back together to form the artifact-removed point cloud. The differences in octree based method and our 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 outperforms the octree based sampling method substantially.

<table>
<thead>
<tr>
<th>Test PC</th>
<th>Noisy PC</th>
<th>Our Method</th>
<th>Octree based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queen</td>
<td>60.23</td>
<td>60.79</td>
<td>60.38</td>
</tr>
<tr>
<td>RedAndBlack</td>
<td>59.44</td>
<td>60.38</td>
<td>59.97</td>
</tr>
<tr>
<td>Soldier</td>
<td>59.20</td>
<td>60.25</td>
<td>59.80</td>
</tr>
<tr>
<td>Average</td>
<td>59.62</td>
<td>60.47</td>
<td>60.05</td>
</tr>
</tbody>
</table>

F. Choosing 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 patches randomly sampled. 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 (Queen, RedAndBlack, and Soldier) 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 seen, the PSNR increases with the value of $C$. PSNR is maximum 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 NVIDIA GeForce GTX 1080 Ti GPU. The computation time includes the sampling, forward propagation through the network, as well as the aggregation scheme. As the figure shows, the computation time increases linearly with the value of $C$.

V. CONCLUSION

V-PCC encoding is the current state-of-the-art 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 serious quality degradation as it introduces compression artifacts. In this paper, we present a first of its kind deep learning-based point cloud geometry compression artifact removal framework for V-PCC encoded dynamic point clouds. We employ a 3D sparse convolutional neural network to learn the direction as well as the level of quantization noise. During V-PCC, the quantization noise is introduced only in the direction of the projection of point cloud. We leverage this prior knowledge to learn the direction of the 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 together to form the final artifact-removed point cloud. Experimental results show that our method considerably improves the geometry quality of the V-PCC reconstructed point cloud both in the subjective as well as objective evaluations.

REFERENCES


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

IEEE TRANSACTIONS ON MULTIMEDIA 10


