2024 IEEE International Conference on Image Processing (ICIP) | 979-8-3503-4939-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICIP51287.2024.10648101

TSF-NET3D: TSF-NET FOR 3D POINT CLOUD ATTRIBUTE COMPRESSION ARTIFACTS REMOVAL

Birendra Kathariya[*]*, Zhu Li*[*]*, Geert Van der Auwera*[†]

[*]University of Missouri-Kansas City, MO 64110 USA
[†]Qualcomm Technologies Inc., San Diego, CA 92121 USA

ABSTRACT

Transformer-based Spatial and Frequency-Decomposed Feature Fusion Network (TSF-Net) exhibited great potential as a learned in-loop filter in Versatile Video Coding (VVC). Utilizing a channel-wise transformer for pixel and frequency-decomposed feature fusion in a multi-scale deep-learning setup, TSF-Net achieved remarkable success in removing video compression artifacts. In this article, considering the potential of TSF-Net, we extend this work to the 3D domain of point clouds and propose a new framework called TSF-Net3D. More specifically, we incorporate sparse convolution (SparseConv) to process point clouds and implement TSF-Net3D as a post-processing block in Geometry-based Point Cloud Compression (G-PCC) to enhance the quality of color attribute in the reconstructed frame. Implementation-wise, TSF-Net3D differs from TSF-Net in two fronts: (1) TSF-Net3D does not utilize frequency-decomposed information but rather pixel information only; (2) TSF-Net3D extends point cloud processing in three scales with two-level feature fusion, unlike TSF-Net, which processes features at only two scales with single-level feature fusion. We evaluate TSF-Net3D on the 8iVFBv2 dataset, and our experimental results demonstrate that our proposed method achieves a significant YUV Bjøntegaard Delta (BD) - bitrate saving of up to -13.12% over the G-PCC(TMC13v21) RAHT baseline while also outperforming other state-of-the-art methods.

Index Terms— point cloud, compression artifact, post-processing filter, sparse-convolution, multi-scale feature

1. INTRODUCTION

Video coding standards, such as High Efficiency Video Coding (HEVC) [1] and Versatile Video Coding (VVC) [2], have played a crucial role in enabling the various video applications we experience today. Similar efforts have also been made by various researchers and technology groups to develop efficient point cloud compression technology. One such effort, led by the MPEG-PCC group, proposed two solutions: Geometry-based Point Cloud Compression (G-PCC) [3] and Video-based Point Cloud Compression (V-PCC) [4], as PCC standards [5]. G-PCC was initially developed to compress static and LiDAR point clouds. Nonetheless, it has now been extended to compress dynamic point clouds as well. V-PCC is developed to compress dynamic point clouds only. The fundamental difference between G-PCC and V-PCC is that G-PCC processes 3D geometry and its attributes in the 3D domain only and compresses them into a bitstream, whereas V-PCC transforms 3D geometry and its attributes into 2D videos and utilizes highly optimized video compression technology like HEVC/VVC to compress them into a bitstream. At the decoder, V-PCC transforms the 2D videos back to 3D geometry and attributes.

G-PCC encodes geometry independently from attributes, where three separate methods: octree, predictive geometry, and trisoup, are introduced for this purpose. In contrast, attribute coding utilizes geometry-based tree structures to hierarchically predict and compute attribute residue. Currently, G-PCC implements two attribute compression methods: Region-Adaptive Hierarchical Transform (RAHT) [6] and Hierarchical Prediction as Lifting Transform (PredLift) [7]. In lossy attribute compression, due to transform and residue quantization, G-PCC often introduces compression artifacts in the attributes of the reconstructed point cloud. In video codecs such as HEVC and VVC, in-loop filters are employed to correct compression artifacts present in the reconstructed frame. Similalry, numerous learning-based in-loop filters and post-processing methods have been developed for HEVC/VVC. However, implementing hand-crafted in-loop filters is non-trivial in PCC codecs due to the point irregularity in the point cloud. However, researchers are currently exploring technologies such as PointNet/PointNet++[8], Graph-Convolution-Network(GCN) [9], Sparse-Convolution-Network (SCN) [10] to develop in-loop and post-processing filters to suppress compression artifacts in both G-PCC and V-PCC.

There exist a few works that applies traditional methods to denoise point cloud color attribute. For e.g. [11] utilizes graph total variation (GTV) prior to formulate point cloud color denoising as maximum a posteriori (MAP) estimation problem. Then the cost function is minimized using alternating direction method of multipliers (ADMM) and proxi-

This work is accomplished in collaboration with *Qualcomm* and partially supported by the NSF under grant CNS-2148382.

mal gradient descent to show a satisfactory denoising performance. Similarly, [12] also employs graph signal processing to denoise the point cloud color. However, it utilizes 3D patch-based similarity to construct the graph where similarity is calculated with small 3D patches around the connected points. Learning-based methods such as [13] also utilizes graph-based method, however, to remove the compression artifacts in the G-PCC coded color. This work proposed Multi-Scale Graph Attention Network (MS-GAT) which uses Chebyshev graph convolution to extract features at multiple scales and weighted graph attention layer to pay attention on points with more compression artifacts. Likeiwse, [14] proposed Compression Artifact Reduction Network (CARNet) to reduce compression artifacts on G-PCC coded point cloud color. CARNet first generates multiple Most-Probable Sample Offsets (MPSOs) as potential compression distortion approximations, and then linearly weights them for artifact mitigation.

In this work, we propose TSF-Net3D, a multi-scale sparse feature learning and fusion method, to reduce compression artifacts in G-PCC coded attributes, specifically point cloud color. TSF-Net3D is an extension of *Transformer-based Spatial and Frequency-Decomposed Feature Fusion Network* (TSF-Net) [15] into the 3D point cloud domain. TSF-Net was introduced as a learned in-loop filter for VVC and demonstrated its effectiveness in reducing compression artifacts in images. The method relies on multi-scale feature learning on pixel and frequency-decomposed information. It implements a naive convolution-based feature extractor but an advanced channel-wise transformer-based feature aggregator to fuse both feature types at two different scales. Given the remarkable performance of TSF-Net, we adapt the implementation into the point cloud domain with two distinct differences: (1) With TSF-Net3D, we extend feature learning to three scales with two-level feature fusion. (2) To simplify feature learning, we design TSF-Net3D to learn only in the pixel space.

We summarize the contributions of this article in the following points.

- We propose TSF-Net3D, a multi-scale sparse feature learning approach with state-of-the-art performance in suppressing compression artifact present in G-PCC coded color attribute.
- We allow feature learning in TSF-Net3D at three scales, thereby exploiting a larger receptive field. This enables us to design a wider but shallower network, offering inherent advantage in inference speed.
- We train TSF-Net3D with the THUman2.0 dataset [16] and test it on five 8iVFBv2 point clouds included in the MPEG-PCC category 1. We utilize G-PCC(TMC13v21) reference software and perform a comprehensive evaluation on RAHT baselines.

2. PROPOSED METHOD

Our proposed TSF-Net3D is implemented as a post-processing filter in G-PCC with the goal of reducing compression artifacts in the color attribute and thereby enhancing the overall quality of the reconstructed point cloud. TSF-Net3D is based on SparseConv, which operates on a sparse-tensor $T = \{C, F\}$ where C and F represent coordinates and features, respectively. Therefore, the reconstructed point cloud $P = \{x_i, y_i\}, i = \{1, 2, ..., N\}$ with N points, is represented as a sparse-tensor where coordinates $C = x_i$ are the XYZ positions and features $F = y_i$ are the YUV color components.

2.1. Description of TSF-Net3D

The architecture of TSF-Net3D is shown in Fig. 1(a). TSF-Net3D is designed to process the input point cloud at three scales: the original scale (1×), two-times downscaled (2×↓), and four-times downscaled (4×↓) denoted as $T_{1\times}$, $T_{2\times\downarrow}$, and $T_{4\times\downarrow}$ respectively. The input point cloud $T_{1\times}$ is first processed with a **head**, which expands the 3-channel YUV-color to a $d = 16$ channel-wide feature. The output is then processed with two successive **head↓**s, each time expanding the feature channel by a factor of two. The **head** consists of **"conv→BN→ReLU→conv"**, as shown in Fig. 2(a), while **head↓** consists of **"conv↓→BN→ReLU→conv"**, as shown in Fig. 2(b). The presence of a stride $s = 2$ sparse-convolution layer (**"conv↓"**) in the **head↓** downscales the input sparse-tensor by a factor of two. Therefore, the output from the **head** is a sparse tensor $T_{1\times} = \{C_{1\times}, F_{1\times} \mid F_{1\times} \in \mathbb{R}^{N\times 16}\}$ at the original point cloud scale. Similarly, the output from the first and second **head↓**s are two-times $T_{2\times\downarrow} = \{C_{2\times\downarrow}, F_{2\times\downarrow} \mid F_{2\times\downarrow} \in \mathbb{R}^{N_1\times 32}\}$ and four-times $T_{4\times\downarrow} = \{C_{4\times\downarrow}, F_{4\times\downarrow} \mid F_{4\times\downarrow} \in \mathbb{R}^{N_2\times 64}\}$ down-scaled sparse-tensors, respectively, where N_1 and N_2 are the numbers of points at two- and four-times downscale.

Next, the sparse-tensors $T_{1\times}$, $T_{2\times\downarrow}$, and $T_{4\times\downarrow}$ are further processed by B instances of the *3-Level Residual Fusion Block* (RFB-3L) successively. The RFB-3L takes these sparse-tensors at three-scales as inputs, extracts deeper features separately, and gradually aggregates the features from lower scale to higher scale. The outputs are again the three sparse-tensors at the same three-scales but with deeper features. Once the sparse-tensors are processed by B instances of RFB-3L blocks, the $T_{1\times}$ output from the last RFB-3L block is added to the output from the **head** as a feature-level residual connection. This summed output is again processed with a **tail**, which has the same layer configuration as the **head**, except for the last **"conv"** layer, which reduces the channel size from 16 down to 3. This output is then added again with the input point cloud as a global-level residual connection to generate the final clean output.

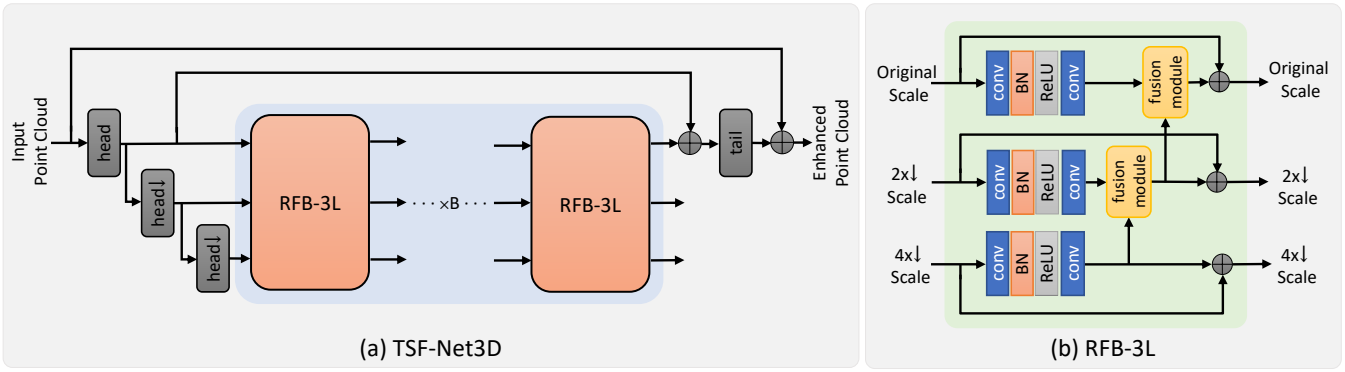

Fig. 1. The overall architecture of our proposed TSF-Net3D.

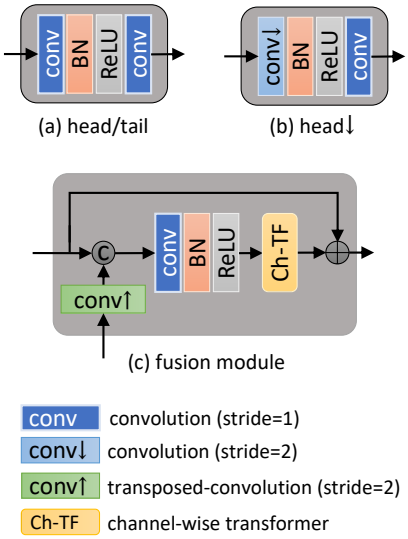

Fig. 2. The components of TSF-Net3D.

2.2. 3-Level Residual Fusion Block (RFB-3L)

The RFB-3L is the primary component of TSF-Net3D, as illustrated in Fig. 1(b). It extracts deeper features at all the three scales of sparse-tensors $T_{1\times}$, $T_{2\times\downarrow}$, and $T_{4\times\downarrow}$ through separate **"conv→BN→ReLU→conv"** blocks. The output feature $F_{4\times\downarrow} \in T_{4\times\downarrow}$ is first fused into $F_{2\times\downarrow} \in T_{2\times\downarrow}$ through a **fusion module**. The fused output feature $F_{2\times\downarrow} \in T_{2\times\downarrow}$ is then fused again into $F_{1\times} \in T_{1\times}$ through another **fusion module**. The output at all three scales is now then added to their respective inputs as a residual connection to form the final outputs of RFB-3L. These residual connections allow TSF-Net3D to form a deeper network by stacking multiple RFB-3L blocks, much like the *ResNet* architecture.

2.2.1. *Fusion Module*

The fusion module takes sparse-tensors at two different scales and fuses the feature at lower scale onto the higher scale as depicted in the Fig. 2(c). Let $T_h = \{C_h, F_h \mid F_h \in \mathbb{R}^{M\times d}\}$ and $T_l = \{C_l, F_l \mid F_l \in \mathbb{R}^{M_1\times 2d}\}$ are the sparse-tensors at these two scales, where M and M_1 represent the numbers of points, and d is the channel-size. T_l is first processed with a transposed-convolution with stride $s = 2$ (**"conv↑"**) which up-scales T_l by a factor of 2 and reduces the channel-size from $2d$ to d. Since, the T_h and T_l are now at the same geometric scale, the features are concatenated to form a sparse-tensor $T_c = \{C_h, F_c \mid F_c \in \mathbb{R}^{M\times 2d}\}$, where $F_c = (F_h © F_l)$ and © implies concatenation. Then T_c is processed with a **"conv→BN→ReLU"** blocks and output feature $F_c \in \mathbb{R}^{M\times 2d}$ is provided to the channel-wise transformer module (**Ch-TF**) for feature aggregation. **Ch-TF** fuses the feature by applying the self-attention along the channel. The T_c is now assigned the fused feature F_f as $T_c = \{C_h, F_f \mid F_f \in \mathbb{R}^{M\times d}\}$. Lastly, the input T_h is added back to T_c as a residual connection to form the final output of the **fusion module**.

2.2.2. *Channel-wise Feature Fusion*

The original TSF-Net [15] work utilizes *Spectral-wise Multi-Head Self-Attention* (SMSA) layer from [17] and redesigns it as a channel-wise feature fusion module. TSF-Net3D follows the same design from the TSF-Net, except the average-pooling operation is avoided before computing *key* and *query* and *convolution* layers are replaced with *linear* layers while computing position-embedding. This channel-wise transformer (**Ch-TF**) as feature-fusion layer is illustrated in Fig. 3.

In **Ch-TF**, the input feature $F_c \in \mathbb{R}^{M\times D_f}$, where $D_f = 2d$, is first linearly projected into *key* $K \in \mathbb{R}^{M\times D_f}$, *query* $Q \in \mathbb{R}^{M\times D_f}$ and *value* $V \in \mathbb{R}^{M\times D_f}$ using 3 linear layers with weights W^q, W^k and $W^v \in \mathbb{R}^{D_f\times D_f}$ respectively. Now the Q, K and V is sub-divided equally into h heads as Q_i, K_i

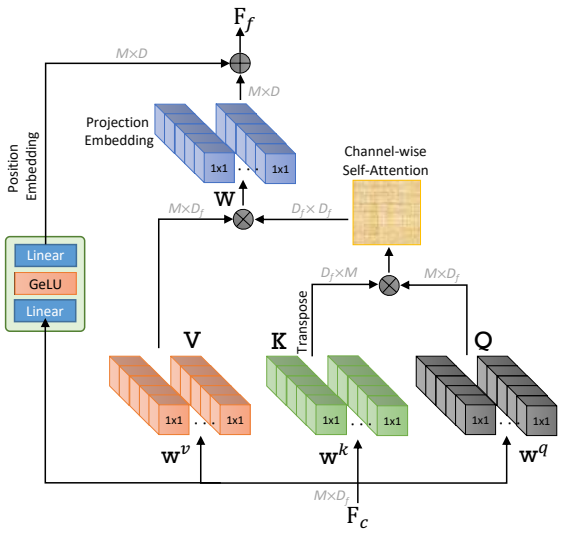

Fig. 3. Ch-TF: Channel-wise Transformer.

and V_i, where $i = 1, 2, ..., h$. For each head, self-attention is computed according to equation (1a), and the fused feature H_i is computed following equation (1b).

$$A_i = softmax(\sigma_i K_i^T Q_i), \quad A_i \in \mathbb{R}^{(D_f/h)\times(D_f/h)} \tag{1a}$$

$$H_i = V_i A_i, \quad H_i \in \mathbb{R}^{M\times(D_f/h)} \tag{1b}$$

where $\sigma_i \in \mathbb{R}^1$ is a learnable parameter employed to adapt self-attention A_i. The fused feature H_i from all h heads are concatenated and linearly projected into projection-embedding $F_{pr} = (concat(H_i))W, F_{pr} \in \mathbb{R}^{M\times D}$ using a linear layer (W), where $D = d$. Similarly, input feature F_c is processed with a **"Linear→GeLU→Linear"** block to generate position-embedding $F_{po} \in \mathbb{R}^{M\times D}$. The final output from **Ch-TF** is calculated as $F_f = F_{pr} + F_{po}, F_f \in \mathbb{R}^{M\times D}$.

3. IMPLEMENTATION DETAILS

3.1. Training/Validation Dataset

We utilize THUman2.0 [16] to create the training and validation datasets. This dataset contains 526 high-quality full-body human mesh data with an 8k texture map. Initially, we convert the mesh to a point cloud by sampling 5 million points from the mesh surface. The point cloud is then voxelized to 10 bits, resulting in roughly one million points per point cloud. The color values of each point in the newly created 10-bit voxelized point cloud are assigned based on the nearest point in the original point cloud.

Next, we use the G-PCC (TMC13v21) reference software to encode all 526 point clouds with the *octree-raht* coding-mode and *lossless-geometry-lossy-attribute* coding-condition, strictly following the MPEG-PCC common test conditions (CTC). This results in a point cloud color processed at 6 quantization-parameter (QP) rate points, including 22, 28, 34, 40, 46, and 51. We then utilize a binary tree [18] of depth 4 to divide a point cloud into 16 leaf-node patches, each with approximately 62,500 points. This results in a total of $526 \times 16 = 8416$ point cloud patches for each QP rate point. The same depth binary tree is also applied to the original 10-bit point cloud to create the ground-truth patches. We utilize patches from the first 511 point clouds ($511 \times 16 = 8176$) to train the network, while the remaining ($15 \times 16 = 240$) patches are used for network validation.

3.2. Test Dataset

We select five point clouds from the 8iVFBv2 dataset (*longdress*, *loot*, *queen*, *redandblack*, and *soldier*), which are included in the mpeg-pcc category-1 [19], as the test dataset. These point clouds are all 10-bit voxelized. We follow the same coding conditions mentioned in section 3.1 to encode the test dataset at 6 QP rate points using the G-PCC (TMC13v21) reference software. Subsequently, we utilize the encoded test point clouds to evaluate the TSF-Net3D.

3.3. Network Training

In this work, we implement TSF-Net3D in the PyTorch-based point cloud processing framework called TorchSparse [20]. The network is optimized to jointly improve the YUV quality. We construct TSF-Net3D by stacking $B = 8$ instances of RFB-3L blocks, with each RFB-3L configured to have 16, 32, and 64 channel sizes at 1×, 2×↓, and 4×↓, scales, respectively. All convolution operations are carried out with a kernel size of $3 \times 3 \times 3$. We use $h = 2$ heads in the channel-wise transformer layer. Similarly, we adopt the Adam optimizer with $\beta = (0.9, 0.999)$ and $L1$ loss to optimize the network parameters. However, the $L1$ loss is computed for Y, U, and V components individually as $\mathcal{L}_Y$, $\mathcal{L}_U$, and $\mathcal{L}_V$, respectively. Then, the final loss $\mathcal{L}_{joint}$ is obtained using equation 2.

$$\mathcal{L}_{joint} = \frac{6}{8}\mathcal{L}_Y + \frac{1}{8}\mathcal{L}_U + \frac{1}{8}\mathcal{L}_V \tag{2}$$

We train a total of 6 models, one for each QP. Initially, we train a model for the highest QP, i.e., $QP = 51$. The model is trained for 50 epochs, with the learning rate initialized at 10^{-4} and decreased to 10^{-6} during training using a cosine-annealing learning rate scheduler. Subsequently, the models for $QP = \{46, 40, 34\}$ are initialized from the pre-trained weights of the model at $QP = 51$ and fine-tuned for 30 epochs. The initial learning rates for these three models are set to 0.5×10^{-4} and decreased to 10^{-6} during training. Finally, the models for $QP = \{28, 22\}$ are trained by initializing their weights from the pre-trained model at $QP = 34$. These two models are fine-tuned for 30 epochs, with an initial learning rate of 10^{-5} while decreasing to 10^{-6} during train-

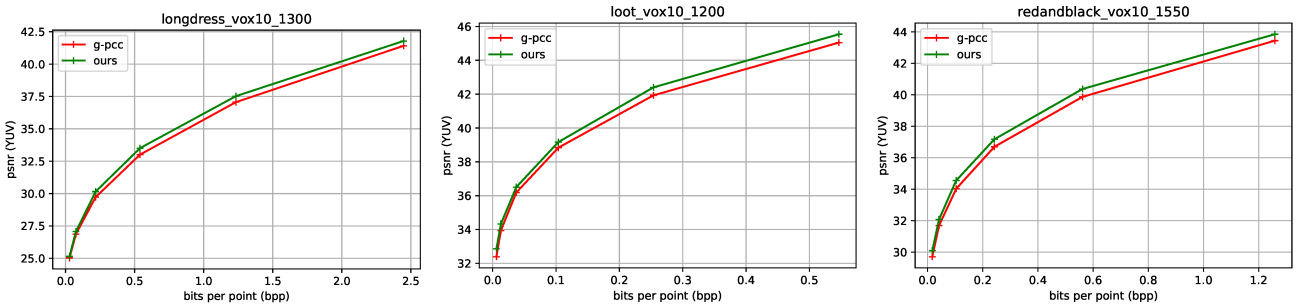

Fig. 4. Rate-Distortion (RD) plots depicting the improvement due to TSF-Net3D over G-PCC (TMC13v21). The YUV-PSNR (dB) are evaluated at 6 QPs={51, 46, 40, 34, 28, 22} over RAHT baseline following the MPEG-PCC Common-Test-Conditions.

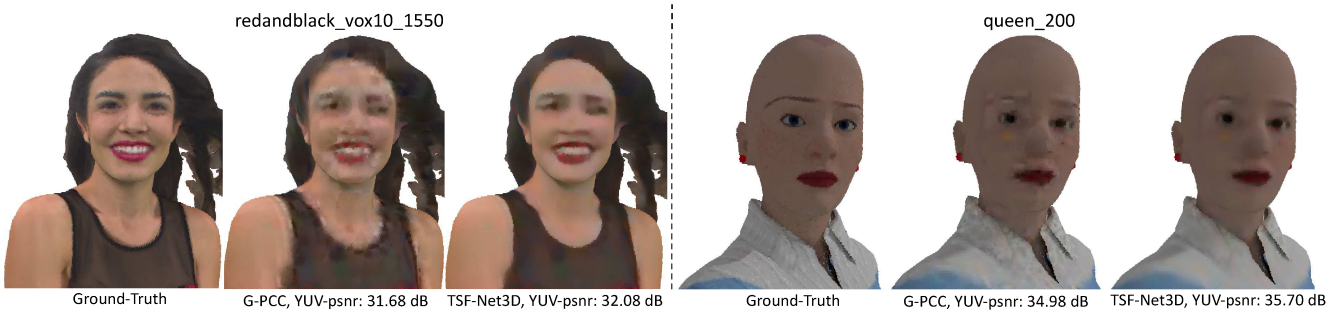

Fig. 5. Visual quality comparison on *redandblack* and *queen* point clouds before and after applying TSF-Net3D. The color of *redandblack* and *queen* are compressed with RAHT(TMC13v21) at QP=51 and QP=46 respectively.

ing. Throughout all training phases, we set the batch size to 2.

Table 1. BD-Rate(%) results of TSF-Net3D compared against G-PCC(TMC13v21).

Point Cloud	BD-Rate (%)			
	Y	U	V	YUV
longdress_vox10_1300	-11.89	0.024	0.06	-9.40
loot_vox10_1200	-15.67	0.32	0.24	-12.86
queen_0200	-20.43	-0.27	-1.84	-16.51
redandblack_vox10_1550	-17.32	0.073	-0.02	-13.44
soldier_vox10_0690	-16.11	2.77	3.67	-13.37
average	-16.29	0.58	0.42	-13.12

4. EXPERIMENTAL RESULTS

4.1. Comparison with G-PCC

In this section, we compare the compression artifact removal performance of TSF-Net3D with the G-PCC (TMC13v21) RAHT baseline. The performance comparison in terms of BD-Rate gain is presented in Table 1. As evident from the table, TSF-Net improves the RAHT baseline by -16.29%, 0.58%, 0.42%, and -13.12% BD-Rate gain in Y, U, V, and YUV color spaces, respectively. Although we observe a slight loss in the U and V components due to joint YUV learning, there is a significant performance gain in the combined YUV space. While *longdress_vox10_1300* shows the least improvement with -9.40% YUV BD-Rate gain, *queen_200* exhibits the most improvement with -16.51% YUV BD-Rate gain. The BD-Rate improvement is inversely correlated with the color complexity of a point cloud.

Table 2. BD-Rate(%) results of CARNet and TSF-Net3D compared against G-PCC(TMC13v14).

Point Cloud	YUV BD-Rate (%)	
	CARNet [14]	TSF-Net3D
longdress	-9.05	-13.61
loot	-5.72	-17.61
queen	-13.78	-15.72
redandblack	-8.35	-19.89
soldier	-4.51	-19.30
average	-8.28	-17.23

Similarly, the improvement in color quality due to TSF-

Net3D over RAHT can be observed in Fig. 4 as a rate-distortion (RD) plot ('YUV-psnr' vs 'bits-per-point'). We notice that TSF-Net3D consistently outperforms RAHT in all 6 QP rates, with a slightly larger PSNR gap at higher QPs. Likewise, the improvement in visual quality attained by TSF-Net over RAHT is shown in Fig. 5. The figure indicates the obvious presence of compression artifacts in the RAHT reconstructed point cloud. The higher the QP value, the more severe the compression artifacts. Nonetheless, TSF-Net3D suppresses the compression artifacts to a large degree, as seen in *redandblack* and *queen*. TSF-Net typically smooths the artifacts to create a pleasing picture, as evident in the case of *queen*. Nevertheless, it also preserves sharp edges, as observed in the contour of the cloth in *redandblack*.

4.2. Comparison with Other Methods

We compare TSF-Net3D with CARNet [14], which is currently the state-of-the-art (SOTA) method for removing compression artifacts in G-PCC coded point cloud attributes. However, in the original work, the authors evaluate CARNet with RAHT baselines of G-PCC (TMC13v14) at only the largest 4 QPs. Therefore, for the purpose of comparison with CARNet, we train TSF-Net3D with the THUman2.0 dataset coded with G-PCC (TMC13v14) RAHT at the same 4 QPs. Additionally, the authors of CARNet select the first frame of 8iVFBv2 sequences included in the MPEG-PCC category 2 dataset as the test dataset. Thus, we also follow the same test setup. Similarly, the authors of CARNet train the model on Y, U, and V individually, as well as on YUV jointly. Since we optimize TSF-Net3D jointly on YUV space, we compare our method with the jointly optimized CARNet model.

The YUV BD-Rate result of CARNet and TSF-Net3D on the five 8iVFBv2 point clouds is presented in Table 2. From the average BD-Rate results, it is evident that TSF-Net significantly outperforms CARNet by almost -9% BD-Rate gain. Interestingly, CARNet exhibits subpar performance even on point clouds with less complex color structures, such as *loot* and *soldier*. In contrast, TSF-Net demonstrates a very robust BD-Rate performance across all types of point clouds.

5. CONCLUSION

Numerous handcrafted and learning-based in-loop and post-processing filters have been successfully applied in video standards such as VVC/HEVC. However, G-PCC lacks handcrafted in-loop filters capable of processing unorganized points in a point cloud. This article introduces TSF-Net3D, a learning-based post-processing filter for G-PCC designed to mitigate compression artifacts in point cloud color. This work extends TSF-Net, a successful application of a learned in-loop for VVC, by utilizing sparse convolution (SparseConv) to process sparsely structured points and extending its capabilities to point clouds. Employing three-scale feature-learning and channel-wise transformer-based cross-scale feature-fusion, TSF-Net3D achieves a remarkable -13.12% YUV BD-Rate gain against the G-PCC RAHT baselines. Similarly, it surpasses previous state-of-the-art methods like CARNet by a substantial margin.

6. REFERENCES

[1] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[2] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm, "Overview of the versatile video coding (vvc) standard and its applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[3] MPEG, "G-pcc codec description v12," *ISO/IEC JTC 1/SC 29/WG 7 N00151*, 2021.

[4] Euee S. Jang, Marius Preda, Khaled Mammou, Alexis M. Tourapis, Jungsun Kim, Danillo B. Graziosi, Sungryeul Rhyu, and Madhukar Budagavi, "Video-based point-cloud-compression standard in mpeg: From evidence collection to committee draft [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 118–123, 2019.

[5] 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, pp. e13, 2020.

[6] Ricardo L. de Queiroz and Philip A. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, 2016.

[7] Birendra Kathariya, Vladyslav Zakharchenko, Zhu Li, and Jianle Chen, "Level-of-detail generation using binary-tree for lifting scheme in lidar point cloud attributes coding," in *2019 Data Compression Conference (DCC)*, 2019, pp. 580–580.

[8] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.

[9] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang, "Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[10] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy, "Sparse convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 806–814.

[11] Chinthaka Dinesh, Gene Cheung, and Ivan V. Bajić, "3d point cloud color denoising using convex graph-signal smoothness priors," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, 2019, pp. 1–6.

[12] Ryosuke Watanabe, Keisuke Nonaka, Eduardo Pavez, Tatsuya Kobayashi, and Antonio Ortega, "Graph-based point cloud color denoising with 3-dimensional patch-based similarity," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[13] Xihua Sheng, Li Li, Dong Liu, and Zhiwei Xiong, "Attribute artifacts removal for geometry-based point cloud compression," *IEEE Transactions on Image Processing*, vol. 31, pp. 3399–3413, 2022.

[14] Dandan Ding, Junzhe Zhang, Jianqiang Wang, and Zhan Ma, "Carnet:compression artifact reduction for point cloud attribute," 2022.

[15] Birendra Kathariya, Zhu Li, and Geert Van der Auwera, "Joint pixel and frequency feature learning and fusion via channel-wise transformer for high-efficiency learned in-loop filter in vvc," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2023.

[16] Tao Yu, Zerong Zheng, Kaiwen Guo, Pengpeng Liu, Qionghai Dai, and Yebin Liu, "Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2021)*, June 2021.

[17] Yuanhao Cai, Jing Lin, Zudi Lin, Haoqian Wang, Yulun Zhang, Hanspeter Pfister, Radu Timofte, and Luc Van Gool, "Mst++: Multi-stage spectral-wise transformer for efficient spectral reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 745–755.

[18] Birendra Kathariya, Li Li, Zhu Li, Jose Alvarez, and Jianle Chen, "Scalable point cloud geometry coding with binary tree embedded quadtree," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[19] Evangelos Alexiou, Irene Viola, Tomás M Borges, Tiago A Fonseca, Ricardo L De Queiroz, and Touradj Ebrahimi, "A comprehensive study of the rate-distortion performance in mpeg point cloud compression," *APSIPA Transactions on Signal and Information Processing*, vol. 8, pp. e27, 2019.

[20] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han, "Torchsparse++: Efficient point cloud engine," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 202–209.