Encoding Auxiliary Information to Restore Compressed Point Cloud Geometry

Gexin Liu[1] , **Jiahao Zhu**[1] , **Dandan Ding**[1] and **Zhan Ma**[2]
[1]School of Information Science and Technology, Hangzhou Normal University, China
[2]School of Electronic Science and Engineering, Nanjing University, China
{liugexin, zhujiahao23}@stu.hznu.edu.cn, dandanding@hznu.edu.cn, mazhan@nju.edu.cn

Abstract

The standardized Geometry-based Point Cloud Compression (G-PCC) suffers from limited coding performance and low-quality reconstruction. To address this, we propose *AuxGR*, a performance-complexity tradeoff solution for point cloud geometry restoration: leveraging auxiliary bitstream to enhance the quality of G-PCC compressed point cloud geometry. This auxiliary bitstream efficiently encapsulates spatio-temporal information. For static coding, we perform paired information embedding (PIE) on the G-PCC decoded frame by employing target convolutions from its original counterpart, producing an auxiliary bitstream containing abundant original information. For dynamic coding, in addition to PIE, we propose temporal information embedding (TIE) to capture motion information between the previously restored and the current G-PCC decoded frames. TIE applies target kNN attention between them, which ensures the temporal neighborhood construction for each point and implicitly represents motions. Due to the similarity across temporal frames, only the residuals between TIE and PIE outputs are compressed as auxiliary bitstream. Experimental results demonstrate that *AuxGR* notably outperforms existing methods in both static and dynamic coding scenarios. Moreover, our framework enables the flexible incorporation of auxiliary information under computation constraints, which is attractive to real applications.

1 Introduction

Point clouds have gained increasing attention in immersive applications as a prevalent format for representing 3D objects and scenes. Given its gigantic volume, a point cloud frame or sequence must be compressed for efficient storage and transmission. Over the past decades, numerous explorations have been devoted to fulfilling this purpose [Cao *et al.*, 2019]. Recently, two different point cloud compression standards were concluded after years of development led by the ISO/IEC MPEG (Moving Picture Experts Group) committee, namely video-based point cloud compression (V-PCC) and geometry-based point cloud compression (G-PCC). The V-PCC standard first performs the 3D-to-2D projection to transform the 3D point cloud to 2D image sequences and then reuses existing video codecs such as H.265/HEVC [Sullivan *et al.*, 2012] and H.266/VVC [Bross *et al.*, 2021] for compression. On the other hand, the G-PCC standard applies octree structured geometry for exploiting correlations directly in 3D space.

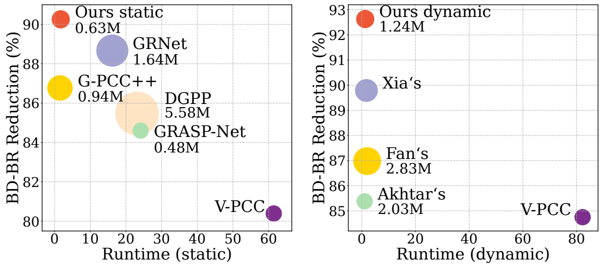


Figure 1: *AuxGR* reveals the optimal performance-complexity tradeoff in both static (*left*) and dynamic (*right*) coding modes. x-axis: runtime (seconds/frame), y-axis: BD-BR reduction against the G-PCC anchor, averaged on thirteen point clouds in static coding mode and four dynamic point clouds in dynamic coding mode. We copy the model size of each learned approach from its paper if applicable.

Lossy compression inevitably incurs distortion for both V-PCC and G-PCC solutions, significantly deteriorating the reconstruction quality. In the past, compressed video quality enhancement/restoration has been extensively studied [Yang *et al.*, 2021], and these methods can be easily extended to enhance V-PCC [Jia *et al.*, 2021]. Thus, this work focuses on restoring the G-PCC compressed point cloud geometry.

To alleviate G-PCC octree quantization-induced artifacts, an earlier attempt – DGPP [Fan *et al.*, 2022b] stacked 3D convolutions to construct a prediction model for restoration. Later, GRNet [Liu *et al.*, 2023] applied sparse convolutions to build coordinate expansion and coordinate refinement modules for a similar purpose. DGPP and GRNet were integrated at the decoder side as the post-processing modules. In the meantime, an alternative layered approach was explored, in which G-PCC was utilized as the base layer to produce a coarse representation, and then learned models were used at the enhancement layer to refine the base layer reconstruction for quality improvement. GRASP-Net [Pang *et al.*, 2022] is a representative example in this category, which explicitly encodes the residual between the original point cloud geometry and the base layer G-PCC reconstruction as the auxiliary information to enhance the quality. However, the random distribution nature of residuals challenges the neural model in learning a stable transforming relation.

All the above solutions focus on static point cloud coding. As for dynamic coding, temporal correlations can be leveraged to reduce data redundancy due to the similarity across point cloud frames. The key challenge is how to exploit such temporal correlations. Typical solutions include implicit motion caption through target convolution [Akhtar *et al.*, 2024] and explicit motion representation through motion estimation [Fan *et al.*, 2022a; Xia *et al.*, 2023]. However, these works generally perform inferior in point clouds having large motions as the limited size of convolution kernels fails to collect sufficient and relevant points for correlation exploration. On the other hand, these dynamic solutions are all learning-based, achieving better performance than G-PCC at the expense of higher computational complexity.

To this end, we propose *AuxGR*, leveraging auxiliary information to restore G-PCC compressed geometry. As such, G-PCC serves as the base coder, and the learned model is augmented for further quality enhancement. Accordingly, the auxiliary bitstream is produced by the learned model to aggregate information from the original frame and the prior decoded frame for transmission. *AuxGR* not only achieves impressive performance (shown in Figure 1) but also allows the encoder to flexibly enable and disable auxiliary information in real applications with constrained computational resources.

The main contributions of this work include:

- We propose the *AuxGR* framework using auxiliary information to restore compressed point cloud geometry. On top of the G-PCC coder, the learning-based model is appended to generate the auxiliary bitstream for further quality improvement.
- The auxiliary bitstream contains spatio-temporal information by applying spatial information aggregation within a frame, paired information embedding between the decoded frame and its original counterpart, and temporal information embedding between the decoded frame and its prior frame.
- Extensive experiments demonstrate the superior performance of our method: it remarkably outperforms existing methods, e.g., >50% BD-BR gains over GRASP-Net [Pang *et al.*, 2022] in static coding and about 13% gains over Fan's [Fan *et al.*, 2022a] in dynamic coding.

2 Related Work

2.1 Point Cloud Geometry Compression (PCGC)

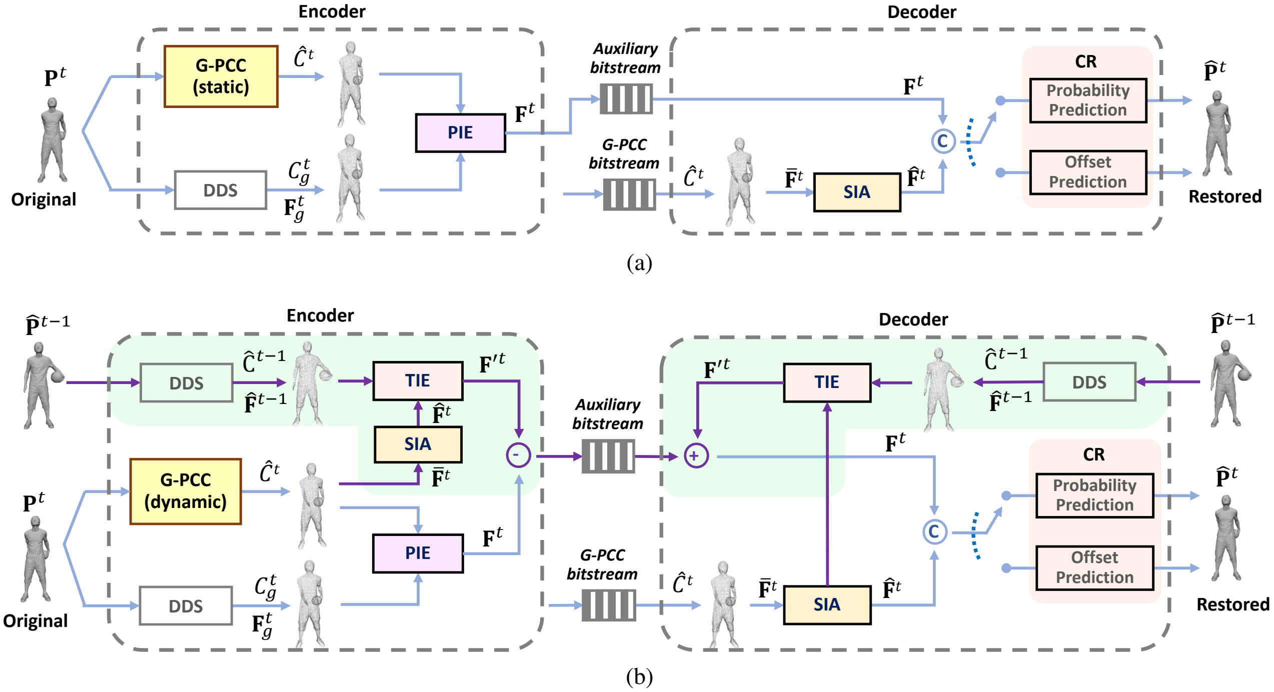
Recently, learning-based PCGC approaches have shown superior performance over rules-based methods in both static and dynamic PCGC tasks, which will be reviewed below.

Learning-based Static PCGC. At early times, learning-based static PCGC [Yan *et al.*, 2019; Huang and Liu, 2019; Wen *et al.*, 2020; Gao *et al.*, 2021; Zhang *et al.*, 2022] operates directly on unordered points of a point cloud, relying on the multilayer perceptron (MLP) to capture spatial correlations across points for compact representation. Such point-based methods usually can only handle small point clouds having thousands of points. For large-scale point clouds, e.g., the object point clouds having millions of points used in VR/MR, they need to chunk them into blocks for processing, which requires long processing time and intensive computation. To solve this, point clouds are later structured by octree model [Nguyen *et al.*, 2021a; Nguyen *et al.*, 2021b; Nguyen and Kaup, 2022; Fu *et al.*, 2022; Que *et al.*, 2021; Song *et al.*, 2023] or voxelization model [Wang *et al.*, 2021b; Wang *et al.*, 2021a; Wang *et al.*, 2022; Guarda *et al.*, 2019; Quach *et al.*, 2019; Nguyen and Kaup, 2023], facilitating the point correlation exploration through the use of regular 3D convolution or sparse tensor. As such, large-scale point clouds can be processed efficiently.

Learning-based Dynamic PCGC. On top of the static PCGC framework, dynamic PCGC methods are developed by introducing temporal information to remove redundancy. As point cloud frames are not aligned in temporal space (their numbers of points even vary), it is challenging to capture accurate motion between two frames. To address this challenge, a pioneer work [Akhtar *et al.*, 2024] was proposed by leveraging target convolutions to convolve the geometry feature of the reference point cloud frame onto the current frame, implicitly capturing motions between two frames. Instead, Fan *et al.* [Fan *et al.*, 2022a] explicitly extracted the motion information between two frames through a motion estimation (ME) network and coded this information as bitstream. Later, Xia *et al.* [Xia *et al.*, 2023] improved this ME to a hierarchical one with k Nearest Neighbour-attention block matching network for more accurate motion prediction. Overall, these methods are effective for dynamic point clouds with small motions. For point clouds having large motions, sufficiently large kernel size of target convolutions is demanded, resulting in huge memory cost.

2.2 Compressed Geometry Restoration

While these learning-based methods exhibit superior performance, they usually require more intensive computational resources than rules-based methods. This motivates researchers to adopt a hybrid approach, combining the rules-based and learning-based techniques: utilizing the rules-based method as the baseline and incorporating an additional learning-based model to enhance its performance. Such a strategy enables cost-effective application of the rules-based method, reserving the incorporation of the learned model for scenarios where computational resources permit. Typical works include GRASP-Net [Pang *et al.*, 2022], G-PCC++ [Zhang *et al.*, 2023], and GRNet [Liu *et al.*, 2023], which employ G-PCC as the baseline coder and leverage the learning-based model to compensate for the compression distortion. Specifically, GRASP-Net compresses the residuals between G-PCC decoded and original point clouds as auxiliary bitstream; G-PCC++ and GRNet are more like post-processing methods for quality restoration of compressed geometry. Up to now, they can only perform static PCGC.



(a)

(b)

Figure 2: Framework of *AuxGR*. (a) In static coding, information of the original point cloud is embedded into the G-PCC decoded point cloud through PIE (denoted as $\mathbf{F}^t$), which is combined with SIA at the decoder to predict coordinate occupancy probabilities or coordinate offsets. (b) In dynamic coding, the temporal information from the prior restored point cloud frame is embedded into the G-PCC decoded one by SIA and then TIE (denoted as $\mathbf{F}'^t$). The residuals between TIE and PIE outputs i.e., $\mathbf{F}^t$ and $\mathbf{F}'^t$, are transmitted. DDS: dyadic downsampling; PIE: paired information embedding; TIE: temporal information embedding; SIA: spatial information aggregation; CR: coordinate reconstruction. A sparse convolution layer is applied to produce latent features $\bar{\mathbf{F}}^t$ before SIA, which is omitted in the figure for clarity.

2.3 Remarks

As seen, combining rules-based G-PCC and learning-based models offers a versatile solution to real applications within constrained computational resources. Therefore, the critical problem is how to leverage the learned model for G-PCC performance enhancement, especially in dynamic coding scenarios where capturing point cloud motions is challenging.

3 Method

3.1 Framework

Figure 2 sketches the framework of the proposed *AuxGR* used to enhance standard-compliant G-PCC. Both static and dynamic point cloud compression modes are supported.

Static Coding. Define the original point cloud as time t as $\mathbf{P}^t = \{C_i^t\}_{i=1}^M$. The corresponding G-PCC decoded point cloud is $\hat{\mathbf{P}}^t = \{\hat{C}_i^t\}_{i=1}^N$, with the geometry quantization parameter *Position Quantization Scale* set as $1/2^r$. For the static mode of *AuxGR*, as illustrated in Figure 2a, the original point cloud $\mathbf{P}^t$ is also dyadically downscaled (DDS) r times to generate C_g^t, a thumbnail point cloud mostly retaining the native information. Then, we apply the paired information aggregation unit (PIE) for target convolution from C_g^t to $\hat{C}^t$, embedding features of the original point cloud into the G-PCC decoded one. The obtained features, referred to as $\mathbf{F}^t$, are coded through entropy coding and converted as auxiliary bitstream to enhance the quality of G-PCC decoded point clouds. Note that we use two sparse convolution (Sconv) layers and InceptionResNet (Inception Residual Network) [Szegedy *et al.*, 2016; Wang *et al.*, 2021a] to implement DDS where ReLU activation is applied after each Sconv layer.

At the decoder, the G-PCC decoded point cloud goes through a spatial information aggregation (SIA) unit via k Nearest Neighbors (kNN)-based attention to construct the neighborhood for each point for spatial correlation exploration. The obtained features $\hat{\mathbf{F}}^t$ are then fused with the auxiliary features $\mathbf{F}^t$ from PIE for coordinate reconstruction.

Dynamic Coding. In dynamic coding, the restored prior point cloud frame can also be utilized. On the one hand, we generate $\mathbf{F}^t$ using the same PIE as the static mode. On the other hand, we embed features of the prior frame $\hat{\mathbf{P}}^{t-1} = \{\hat{C}_j^{t-1}\}_{j=1}^{N'}$ into the current G-PCC decoded point cloud $\hat{C}^t$. Specifically, the current G-PCC decoded point cloud goes through SIA, producing latent features $\hat{\mathbf{F}}^t$; accordingly, the prior restored point cloud goes through DDS, producing the same scale latent features $\hat{\mathbf{F}}^{t-1}$. As these exist deep temporal correlations between $\hat{\mathbf{F}}^t$ and $\hat{\mathbf{F}}^{t-1}$, we fuse them using the temporal information embedding (TIE) unit, where the target kNN attention is utilized for temporal correlation exploration. As such, motion information from the prior frame is embedded into the G-PCC decoded frame, denoted as $\mathbf{F}'^t$.

In this way, we derive $\mathbf{F}^t$ using PIE and $\mathbf{F}'^t$ using TIE. There is redundancy between these two groups of features

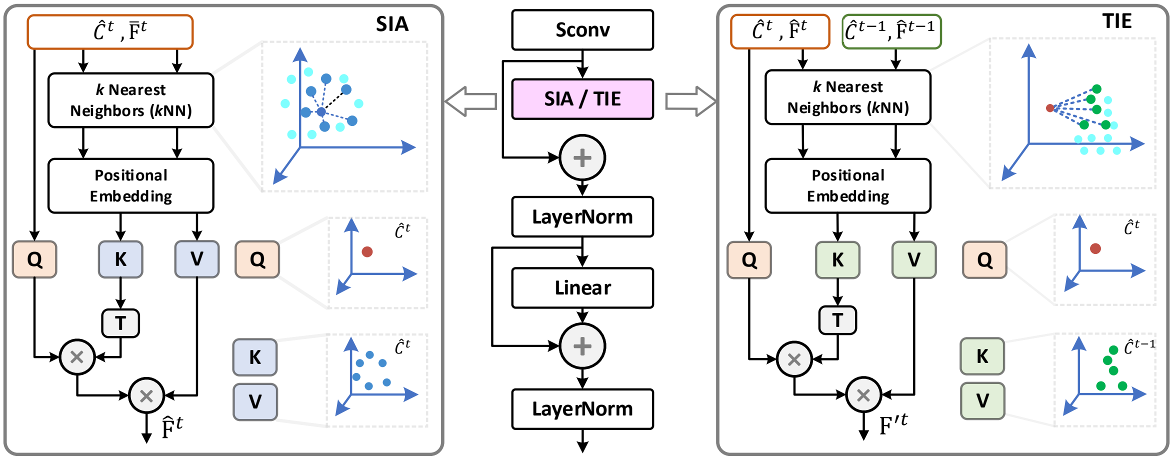


Figure 3: SIA & TIE. SIA is applied on the current decode point cloud: for each point, it uses k Nearest Neighbors (kNN) to construct its spatial neighborhood for information aggregation. TIE is applied between two point clouds: for each point in the G-PCC decoded point cloud $\hat{C}^t$, its k neighbors in the prior point cloud $\hat{C}^{t-1}$ are aggregated for temporal neighborhood construction and feature embedding.

due to the temporal similarity between $\mathbf{P}^{t-1}$ and $\mathbf{P}^t$. Since both features are embedded based on the G-PCC decoded point cloud, we can directly compute their residuals for entropy coding, eventually forming the auxiliary bitstream.

As the prior restored point cloud is also available to the *AuxGR* decoder, the decoder can mirror the encoding process to generate $\mathbf{F}'^t$ using SIA and TIE, which is then added to the received residual features to derive $\mathbf{F}^t$. Similarly, $\mathbf{F}^t$ and $\hat{\mathbf{F}}^t$ are utilized together to enhance the point cloud quality.

3.2 Modular Component

Next, we detail the key modular components in *AuxGR*.

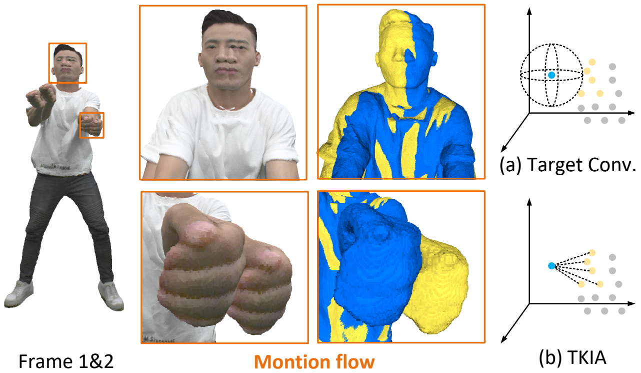


Figure 4: TIE vs. Conventional Target Convolution. We merge two successive frames in the point cloud *Dancer* to observe the motion flow. The conventional target convolution may involve no point in the receptive field due to the large motion, while TIE always ensures a local neighborhood for each point by searching its k nearest neighbors temporally.

SIA. SIA is applied within the current G-PCC decoded point cloud. For each point, we first collect the spatially k nearest neighboring points to construct its neighborhood. Then, self-attention is applied, as shown in the left of Figure 3, to adaptively aggregate features close to this point, denoted as $\hat{\mathbf{F}}^t$.

PIE. PIE is a target convolutional layer used to perceive spatial variations between the paired decoded and original point clouds. As there is no motion but only distortion between them, each point in the G-PCC decoded point cloud is highly correlated to its neighbors in the original counterpart. To this end, we directly apply PIE to embed the original features into the G-PCC decoded point cloud coordinates.

TIE. In addition to the spatial correlation, there exists temporal similarity across point cloud frames. Therefore, we can use the previously decoded frame to improve the compression efficiency of the current frame. The remained problem is how to fully exploit such temporal similarity. Due to the point cloud's irregular and disordered nature, there is no accurate point-to-point motion between the two frames. As a result, directly capturing motion for each point is exhaustive and inaccurate. To address this issue, this paper proposes TIE to embed temporal motion and correlation information into the current G-PCC decoded point cloud frame.

As illustrated in the right of Figure 3, the prior frame is downscaled by DDS, outputting $\hat{\mathbf{F}}^{t-1}$ of the same scale as $\hat{\mathbf{F}}^t$. For each point in $\hat{\mathbf{F}}^t$, we apply the TIE unit to construct its neighborhood by searching k nearest neighbors in $\hat{\mathbf{F}}^{t-1}$. Afterward, self-attention is performed in each local neighborhood to exploit their temporal correlation and represent their motion implicitly. Notice that we use TIE instead of conventional target convolution because the object motion easily leads to a mismatch between frames, especially in large motion, making the conventional target convolution extract no features or irrelevant features. For example, Figure 4 shows the motion between two successive frames of *Dancer*, in which using target convolutions may involve no points in the receptive field due to the large motion. By contrast, TIE can always ensure a local neighborhood for each point, and most points in this neighborhood are highly correlated.

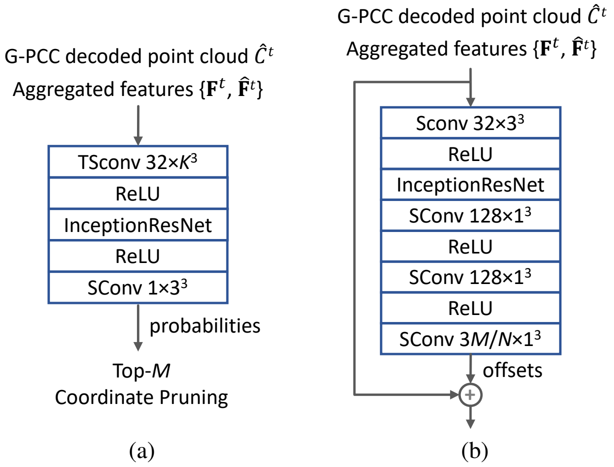


Figure 5: Modular components of Coordinate Reconstruction: (a) Probability Generation and (b) Offset Generation.

Coordinate Reconstruction (CR) provides two options: *Probability Generation* for solid point clouds and *Offset Generation* for dense and sparse point clouds, shown in Figure 5. As the encoder can easily determine the point cloud type through the density information [Alexandre *et al.*, 2022], we thus only need to signal a 1-bit flag indicating the point cloud type in the bitstream. Besides, the number of points in the original point cloud (i.e., M) is also signaled in the bitstream to facilitate the CR module in generating the same number of points as the original point cloud.

It is observed that G-PCC decoded point clouds exhibit two distinct characteristics [Liu *et al.*, 2023]: the solid point cloud predominantly experiences the point vanishing issue because its points are so close to each other that many points (e.g., eight points) are merged as one after the octree-based quantization; instead, the dense and sparse point clouds suffer from both point vanishing and point displacement problems and more serious in the latter one due to its low point density. Therefore, to restore the decoded solid point cloud, we need to generate more points to resist point vanishing. Thus, we apply the Probability Generation unit to uniformly generate a set of points around each point and select points having the highest probabilities as the restored ones. As depicted in Figure 5a, given the decoded point cloud $\hat{C}^t$ having N points and features $\{\mathbf{F}^t, \hat{\mathbf{F}}^t\}$, the Probability Generation unit uses a transposed sparse convolution (TSconv) with kernel size K ($K = 9$ in this work) for coordinate generation, followed by an InceptionResNet [Szegedy *et al.*, 2016; Wang *et al.*, 2021a] layer and a Sconv layer to produce a set of coordinates associated with their probabilities. Given that the original point cloud has M points, we select points with the top M probabilities as the restored point clouds.

For dense and sparse point clouds, most points are retained but significantly shifted after compression, which inspires us to rectify these displacements via coordinate offsets. Meanwhile, we also need to retrieve those vanished points. To achieve this, we devise the Offset Generation unit (see Figure 5b). To restore the same number of points as the original point cloud, we generate $W = \lfloor \frac{M}{N} \rfloor$ offsets via stacked Sconv and InceptionResNet layers for each point in the decoded point cloud. As such, we receive W new coordinates by adding these offsets to the coordinate of this point. Given N decoded points, we finally obtain $\lfloor \frac{M}{N} \rfloor \times N$ new points, a similar number of points to the original point cloud.

4 Experimental Results and Analysis

4.1 Preparation

We employ GeS-TM v1[1], a branch split out of the G-PCC test model with the best practices for static and dynamic solid point cloud compression, as the baseline coder of *AuxGR*. For the static point cloud compression, GeS-TM v1 remains the same as G-PCC TMC13v23[2]. In the following, we use GeS-TM (static) and GeS-TM (dynamic) for short.

Training Dataset. For static coding, we built different training datasets for solid, dense, and sparse point clouds using the ShapeNet dataset [Chang *et al.*, 2015]. Specifically, we randomly selected 20,000 3D mesh from ShapeNet and sampled each model to a maximum of 400,000 8-bit points, 25,000 8-bit points, and 20,000 9-bit points to mimic the characteristics of solid, dense, and sparse point clouds, respectively, forming their corresponding static training datasets.

For dynamic coding, we followed the Common Test Conditions (CTC) [MPEG, 2022] recommended by the international standardization committee MPEG AI-PCC group to create our training dataset. Specifically, the 8i Voxelized Full Bodies (8iVFB) [d'Eon *et al.*, 2017] dataset with 10-bit geometry precision is used. This dataset includes four sequences and each has a total of 300 frames. In the training, each frame was chunked into small blocks of 100,000 to 200,000 points, finally obtaining 7,432 blocks.

Testing Dataset. The datasets recommended by MPEG [Alexandre *et al.*, 2022], including thirteen solid point clouds, nine dense point clouds, and four sparse point clouds, are used as our static testing dataset. These point clouds range from 9 bits to 12 bits.

As for the dynamic coding, the Owlii dataset [Xu *et al.*, 2017] containing four dynamic point clouds is used for testing. Following the MPEG CTC, we quantized point clouds in Owlii to 10-bit precision and employed the first 100 frames of each sequence for testing. Notice that the first frame is processed using the static solution, and the following 99 frames are processed using the dynamic solution.

Settings. Our project is implemented using the PyTorch and MinkowskiEninge [Choy *et al.*, 2019] on a computer with an Intel® i7-8700K CPU, 32 GB memory, and NVIDIA RTX 4090. Adam is used for network model optimization and parameters β_1 and β_2 are set to 0.9 and 0.999, respectively. The learning rate decays from 8e-4 to 1e-4 every 4000 steps. Our static and dynamic models are trained for 64,000 and 24,000 steps, respectively. It takes around 24 hours for each model to converge on our platform.

[1] http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-ges-tm

[2] https://github.com/MPEGGroup/mpeg-pcc-tmc13

Mode	**Point Cloud**	**vs. GeS-TM (static)**		**vs. V-PCC**		**vs. DGPP**		**vs. GRASP-Net**		**vs. G-PCC++**		**vs. GRNet**	
		D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)
Static	Solid Average	-90.76	-83.62	-47.28	-42.59	-30.33	-16.19	-53.35	-58.07	-20.26	-24.62	-7.00	-7.72
	Dense Average	-84.69	-73.16	-	-	-	-	-26.49	-28.15	-	-	-52.83	-23.41
	Sparse Average	-35.90	-36.01	-	-	-	-	-14.80	-22.37	-	-	-28.15	-7.53
Mode	**Point Cloud**	**vs. GeS-TM (dynamic)**		**vs. V-PCC**		**vs. Akhtar's**		**vs. Fan's**		**vs. Xia's**		**vs. Proposed Static**	
		D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)
Dynamic	Basketball vox10	-92.83	-89.90	-79.59	-67.87	-18.53	-	-14.03	-16.13	-6.46	-31.21	-25.31	-27.77
	Dancer vox10	-92.04	-88.53	-81.85	-69.35	-26.99	-	-14.34	-17.31	-9.90	-33.17	-22.44	-25.53
	Exercise vox10	-92.72	-89.86	-81.26	-69.82	-22.19	-	-10.30	-11.00	-2.42	-30.00	-29.59	-31.91
	Model vox10	-91.50	-86.91	-85.46	-68.73	-28.03	-	-13.13	-14.94	-3.35	-26.29	-29.43	-33.51
	Average	-92.27	-88.80	-82.04	-68.94	-23.94	-	-12.95	-14.85	-5.53	-30.17	-26.69	-29.68

Table 1: Average BD-BR gains of the proposed method against existing solutions in static and dynamic point cloud compression

Mode	**Point Cloud**	**GeS-TM (static)**	**V-PCC**	**DGPP**	**GRASP-Net**	**G-PCC++**	**GRNet**	**Proposed Static**
Static	Solid Average	1.01	61.44	23.11	24.12	1.60	16.18	1.72
	Dense Average	4.42	-	-	193.48	-	191.62	6.16
	Sparse Average	1.22	-	-	15.38	-	14.43	3.32
Mode	**Point Cloud**	**GeS-TM (dynamic)**	**V-PCC**	**Akhtar's**	**Fan's**	**Xia's**	**Proposed Static**	**Proposed Dynamic**
Dynamic	Owlii Average	0.88	82.26	1.07	2.00	1.74	1.01	1.29

Table 2: Average runtime (seconds/frame) of compared methods

	w/o PIE		**w/o SIA**	
	D1	D2	D1	D2
Average	-9.29%	-8.91%	-17.54%	-12.91%

Table 3: Ablation of each modular component in *AuxGR*

In our experiments, quantization parameters *Position Quantization Scale* and λ are used to control the GeS-TM bitrate and the auxiliary bitrate, respectively. They can be arbitrarily combined to achieve different quality levels and bitrates. In experiments, we try our best to apply the same settings of compared methods. Specifically,

- For static coding, in addition to rules-based GeS-TM (static) and V-PCC, we compare with state-of-the-art point cloud geometry restoration methods DGPP [Fan *et al.*, 2022b][3], GRASP-Net [Pang *et al.*, 2022][4], G-PCC++ [Zhang *et al.*, 2023], and GRNet [Liu *et al.*, 2023]. Among all, GRASP-Net uses G-PCC as the basic coder and the auxiliary bitstream for residual coding. Thus, we follow the G-PCC settings of GRASP-Net and tune λ in our learned model to produce similar bitrate points to GRASP-Net in testing (see Figure 6c). For fair comparisons, we replace the G-PCC coder with GeS-TM v1 in GRASP-Net for evaluation.
- For dynamic coding, we compare with GeS-TM (dynamic) and V-PCC, as well as learning-based point cloud compression solutions [Akhtar *et al.*, 2024], [Fan *et al.*, 2022a], and [Xia *et al.*, 2023]. Since these learning-based works all follow the MPEG CTC for training in their MPEG proposals or papers, we thus directly use corresponding results for comparison. As Akhtar's work uses lossless G-PCC to compress downscaled geometry, we replace their G-PCC with GeS-TM v1 for fair comparisons. Fan's and Xia's methods downscale the point cloud two times and use a high-efficiency learned model for lossless compression. As their corresponding lossless bitrate is around 0.05 bpp, this work sets *Position Quantization Scale* of GeS-TM as 0.25 to produce a comparable bitrate (see Figure 6c).

Evaluation metrics include bitrate bpp (bit per point), D1 (point-to-point) PSNR, D2 (point-to-plane) PSNR, and BD-BR (Bjøntegaard Delta bitrate) [Bjøntegaard, 2001] recommended by MPEG for compression performance evaluation.

4.2 Static Coding Performance

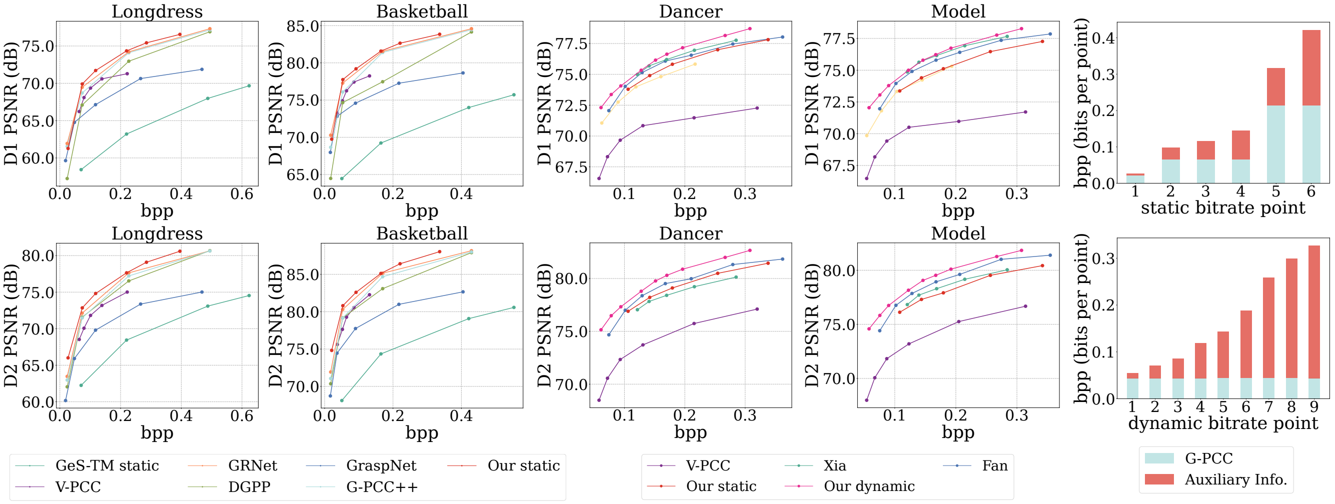
We use the pre-trained models of GRASP-Net and DGPP for testing. For G-PCC++ and GRNet, we reproduce them using the same training dataset as ours for fair comparisons. Notice that GRASP-Net [Pang *et al.*, 2022] uses ModelNet, which contains similar 3D models to ShapeNet, as its training dataset. Thus, there is almost no difference between the coding performance of ShapeNet and ModelNet-trained GRASP-Net models. Concerning DGPP [Fan *et al.*, 2022b], it randomly selects 60 frames from both *Longdress* and *Loot* sequences in 8iVFB for training. It is worth noting that this may lead to the overfitting of DGPP models on specific sequences. Nevertheless, we choose to utilize its pre-trained models to preserve its peak performance.

Table 1 presents the coding performance comparison of various methods under the same testing conditions. For the solid point clouds, our method largely outperforms all other methods, e.g., it rivals GRASP-Net which also employs G-PCC as the base coder by 53.35% (58.07%) BD-BR gains in D1 (D2) measurement. In comparison to G-PCC++, a learning-based geometry restoration method for G-PCC, *AuxGR* obtains 20.26% (24.62%) BD-BR reduction.

For dense and sparse point clouds, we only compare *AuxGR* with GeS-TM (static), GRASP-Net, and GRNet, as other methods lack support for compressing these types of point clouds. Taking the dense point clouds for example, compared with GeS-TM (static), our method attains as high as 84.69% (73.16%) BD-BR reduction in terms of D1 (D2); compared with GRASP-Net, *AuxGR* also gains significantly,

[3] https://github.com/fxqzb/Deep-Geometry-Post-Processing

[4] https://github.com/InterDigitalInc/GRASP-Net



(a) (b) (c)

Figure 6: R-D curves of (a) static and (b) dynamic point cloud compression. As GeS-TM (dynamic) is much lower than others, we omit it here for clarity. (c) The bitrate distribution of GeS-TM and our *AuxGR* in static and dynamic coding.

26.49% (28.15%) D1 (D2) BD-BR on average. Similarly, *AuxGR* performs remarkably better than other methods in sparse point clouds. R-D curves in Figure 6a consistently present the superior performance of *AuxGR*. We also show the bitrate of our auxiliary information in Figure 6c.

4.3 Dynamic Coding Performance

As presented in Table 1 and Figure 6b, compared with GeS-TM (dynamic) and V-PCC, *AuxGR* achieves 92.83% (89.90%) and 79.59% (67.87%) BD-BR reduction in D1 (D2) measurement. In comparison to Akhtar's and Fan's works, *AuxGR* gains 18.53% (-) and 14.03% (16.13%), respectively. To demonstrate the effectiveness of our dynamic solution, we compare it with our static version, showing 25.31% (27.77%) BD-BR gains. In comparison to Xia's work, our gain is also significant, especially in D2 measurement. The detailed bitrates of GeS-TM and our *AuxGR* are presented in Figure 6c.

Overall, *AuxGR* performs the best, particularly in point clouds with large motions. For example, the point cloud sequence *Dancer* has large motions while *Exercise* remains almost unchanged. Accordingly, Fan's and Xia's methods behave worse in *Dancer* because they mostly rely on convolutions to capture motions, limiting the receptive field and failing to aggregate enough information in large-motion scenarios. By contrast, our *AuxGR* performs much better in *Dancer*, due to the use of TIE for implicit motion perception and effective temporal information embedding.

4.4 Space and Runtime Complexity

We collect the model parameters of compared methods in Figure 1. For static compression, our model uses slightly more parameters than GRASP-Net (0.63M vs. 0.48M) and fewer parameters than others. For dynamic compression, our model parameter is increased to 1.24M, which is still much smaller than others, e.g., 44% of Fan's and 61% of Akhtar's.

Table 2 further presents the runtime of different methods. Notice that GeS-TM and V-PCC run on the CPU while learning-based methods run on the GPU and CPU. As observed, the runtime of *AuxGR* is in the same order of magnitude as that of GeS-TM in both static and dynamic coding, much faster than most other methods, e.g., 65% of Fan's and 74% of Xia's.

4.5 Ablation Study

Ablation studies are conducted to evaluate the contribution of SIA, PIE, and TIE. We first disable the PIE unit in static compression, i.e., no auxiliary bitstream will be generated. In this way, the static compression becomes geometry post-processing on the decoder side, resulting in 9.29% (8.91%) D1 (D2) BD-BR loss against the full *AuxGR*. Additionally, we replace the SIA unit on the decoder using two convolution layers for ablation. It is observed that without efficient neighborhood construction, 17.54% (12.91%) BD-BR loss is introduced. At last, disabling TIE simplifies our dynamic coding to static coding, leading to about 27% BD-BR loss.

5 Conclusion

This paper proposes *AuxGR*, leveraging auxiliary bitstream to restore G-PCC compressed point cloud geometry. The auxiliary bitstream is generated by a learning-based model using SIA to exploit point correlation within the current G-PCC decoded frame, PIE to embed the original point cloud information into the G-PCC decoded frame, and TIE to embed the temporal information of the prior restored frame into the G-PCC decoded frame. By configuring SIA, PIE, and TIE, *AuxGR* can support static and dynamic coding modes. Experimental results confirm that *AuxGR* remarkably surpasses state-of-the-art solutions under the same testing conditions in both static and dynamic coding scenarios.

Acknowledgments

This research was supported by the National Key Research and Development Project of China (2022YFF0902402) and the National Natural Science Foundation of China (62171174).

References

[Akhtar *et al.*, 2024] Anique Akhtar, Zhu Li, and Geert Van der Auwera. Inter-frame compression for dynamic point cloud geometry coding. *IEEE Transactions on Image Processing*, 33:584–594, 2024.

[Alexandre *et al.*, 2022] Zaghetto Alexandre, Danillo Graziosi, and Tabatabai Ali. Dataset Split for AI-PCC. *ISO/IEC JTC 1/SC 29/WG 7 m58754*, 2022.

[Bjøntegaard, 2001] G. Bjøntegaard. Calculation of average PSNR differences between RD-curves. In *ITU-T SG 16/Q6, 13th VCEG Meeting*. document VCEG-M33, April 2001.

[Bross *et al.*, 2021] Benjamin Bross, Jianle Chen, Jens-Rainer Ohm, Gary J. Sullivan, and Ye-Kui Wang. Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC). *Proceedings of the IEEE*, 109(9):1463–1493, 2021.

[Cao *et al.*, 2019] Chao Cao, Marius Preda, and Titus Zaharia. 3D point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*, pages 1–9, 2019.

[Chang *et al.*, 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[Choy *et al.*, 2019] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[d'Eon *et al.*, 2017] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 2017.

[Fan *et al.*, 2022a] Tingyu Fan, Linyao Gao, Yiling Xu, Zhu Li, and Dong Wang. D-DPCC: deep dynamic point cloud compression via 3D motion prediction. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 898–904, 2022.

[Fan *et al.*, 2022b] Xiaoqing Fan, Ge Li, Dingquan Li, Yurui Ren, Wei Gao, and Thomas H. Li. Deep geometry post-processing for decompressed point clouds. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2022.

[Fu *et al.*, 2022] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 625–633, 2022.

[Gao *et al.*, 2021] Linyao Gao, Tingyu Fan, Jianqiang Wan, Yiling Xu, Jun Sun, and Zhan Ma. Point cloud geometry compression via neural graph sampling. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3373–3377. IEEE, 2021.

[Guarda *et al.*, 2019] André F. R. Guarda, Nuno M. M. Rodrigues, and Fernando Pereira. Point cloud coding: Adopting a deep learning-based approach. In *2019 Picture Coding Symposium (PCS)*, pages 1–5, 2019.

[Huang and Liu, 2019] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*, pages 890–898, 2019.

[Jia *et al.*, 2021] Wei Jia, Li Li, Zhu Li, and Shan Liu. Deep learning geometry compression artifacts removal for video-based point cloud compression. *International Journal of Computer Vision*, 129(11):2947–2964, 2021.

[Liu *et al.*, 2023] Gexin Liu, Ruixiang Xue, Jiaxin Li, Dandan Ding, and Zhan Ma. GRNet: Geometry restoration for g-pcc compressed point clouds using auxiliary density signaling. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[MPEG, 2022] MPEG. Description of exploration experiment 5.3 on AI-based dynamic PC coding. *ISO/IEC JTC 1/SC 29/WG 7 N00344*, 2022.

[Nguyen and Kaup, 2022] Dat Thanh Nguyen and André Kaup. Learning-based lossless point cloud geometry coding using sparse tensors. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2341–2345. IEEE, 2022.

[Nguyen and Kaup, 2023] Dat Thanh Nguyen and André Kaup. Lossless point cloud geometry and attribute compression using a learned conditional probability model. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[Nguyen *et al.*, 2021a] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Learning-based lossless compression of 3D point cloud geometry. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4220–4224, 2021.

[Nguyen *et al.*, 2021b] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Multiscale deep context modeling for lossless point cloud geometry compression. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6, 2021.

[Pang *et al.*, 2022] Jiahao Pang, Muhammad Asad Lodhi, and Dong Tian. Grasp-net: Geometric residual analysis and synthesis for point cloud compression. In *Proceedings of the 1st International Workshop on Advances in Point*

Cloud Compression, Processing and Analysis, pages 11–19, 2022.

[Quach *et al.*, 2019] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4320–4324, 2019.

[Que *et al.*, 2021] Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6042–6051, 2021.

[Song *et al.*, 2023] Rui Song, Chunyang Fu, Shan Liu, and Ge Li. Efficient hierarchical entropy model for learned point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14368–14377, 2023.

[Sullivan *et al.*, 2012] 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*, 22(12):1649–1668, 2012.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[Wang *et al.*, 2021a] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82, 2021.

[Wang *et al.*, 2021b] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4909–4923, 2021.

[Wang *et al.*, 2022] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multiscale representation for point cloud geometry compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nov. 2022.

[Wen *et al.*, 2020] Xuanzheng Wen, Xu Wang, Junhui Hou, Lin Ma, Yu Zhou, and Jianmin Jiang. Lossy geometry compression of 3D point cloud data via an adaptive octree-guided network. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2020.

[Xia *et al.*, 2023] Shuting Xia, Tingyu Fan, Yiling Xu, Jenq-Neng Hwang, and Zhu Li. Learning dynamic point cloud compression via hierarchical inter-frame block matching. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7993–8003, 2023.

[Xu *et al.*, 2017] Yi Xu, Yao Lu, and Ziyu Wen. Owlii dynamic human mesh sequence dataset. *ISO/IEC JTC1/SC29/WG11 m41658, 120th MPEG Meeting*, 2017.

[Yan *et al.*, 2019] Wei Yan, Yiting shao, Shan Liu, Thomas H Li, Zhu Li, and Ge Li. Deep autoencoder-based lossy geometry compression for point clouds, 2019.

[Yang *et al.*, 2021] Ren Yang, Radu Timofte, et al. NTIRE 2021 challenge on quality enhancement of compressed video: Methods and results. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2021.

[Zhang *et al.*, 2022] Junteng Zhang, Gexin Liu, Dandan Ding, and Zhan Ma. Transformer and upsampling-based point cloud compression. In *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, pages 33–39, 2022.

[Zhang *et al.*, 2023] Junzhe Zhang, Tong Chen, Dandan Ding, and Zhan Ma. G-PCC++: Enhanced geometry-based point cloud compression. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1352–1363, 2023.