

LINR-PCGC: Lossless Implicit Neural Representations for Point Cloud Geometry Compression

Wenjie Huang¹ Qi Yang² Shuting Xia¹ He Huang¹ Yiling Xu^{1†} Zhu Li²

¹ Shanghai Jiao Tong University ² University of Missouri-Kansas City

¹{huangwenjie2023, xiashuting, huanghe0429, yl.xu}@sjtu.edu.cn , ²{qiyang, lizhu}@umkc.edu

Abstract

Existing AI-based point cloud compression methods struggle with dependence on specific training data distributions, which limits their real-world deployment. Implicit Neural Representation (INR) methods solve the above problem by encoding overfitted network parameters to the bitstream, resulting in more distribution-agnostic results. However, due to the limitation of encoding time and decoder size, current INR based methods only consider lossy geometry compression. In this paper, we propose the first INR based lossless point cloud geometry compression method called Lossless Implicit Neural Representations for Point Cloud Geometry Compression (**LINR-PCGC**). To accelerate encoding speed, we design a group of point clouds level coding framework with an effective network initialization strategy, which can reduce around 60% encoding time. A lightweight coding network based on multiscale SparseConv, consisting of scale context extraction, child node prediction, and model compression modules, is proposed to realize fast inference and compact decoder size. Experimental results show that our method consistently outperforms traditional and AI-based methods: for example, with the convergence time in the MVUB dataset, our method reduces the bitstream by approximately 21.21% compared to G-PCC TMC13v23 and 21.95% compared to SparsePCGC. Our project can be seen on <https://huangwenjie2023.github.io/LINR-PCGC/>.

1. Introduction

Point clouds have become a pivotal data format for representing and interacting with 3D environments. Their ability to capture complex spatial structures with high fidelity makes them indispensable for many real-world applications, such as applications in the metaverse and autonomous driving [6, 27]. However, the large data volume of point clouds, specifically for the point cloud sequence that can capture dynamic content, poses significant challenges for storage,

† Corresponding Author.

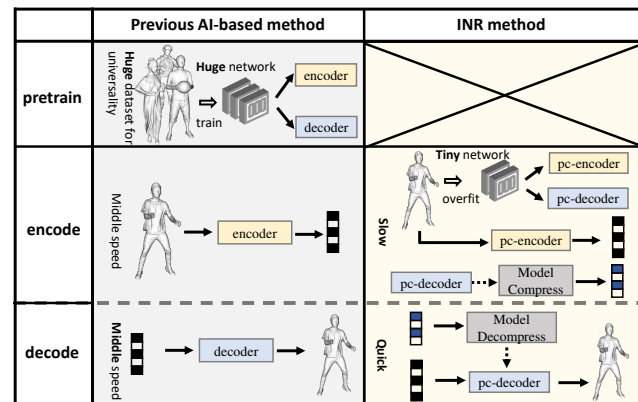


Figure 1. Previous AI-based method typically employs large-scale datasets to train a high-capacity neural network for universality, which operates in inference mode with fixed parameters during both encoding and decoding. INR method overfits the network with target point cloud to be compressed. Both the point cloud and the network parameters will be encoded into bitstream.

transmission, and real-time processing, necessitating the development of efficient compression techniques. In this paper, we focus on the Point Cloud sequence Geometry Lossless Compression (PCGLC) study.

Current PCGLC methods can be mainly categorized into traditional approaches and AI-driven approaches. Traditional methods, such as Geometry-based Point Cloud Compression (G-PCC) [2, 7] and Video-based Point Cloud Compression (V-PCC) [1, 13, 29], have been standardized by the Coding of 3D Graphics and Haptics (WG7) of the Moving Picture Experts Group (MPEG). These methods have demonstrated strong adaptability, interpretable operations, and impressive compression ratios. Despite these advantages, these methods rely on manually designed tools and parameters, which limit their ability to fully exploit geometry spatial, leading to suboptimal performance [6]. AI-driven methods, based on 3D regular voxels [10, 23, 33] or tree structures [5, 11, 17, 30], address these limitations using neural networks to model spatial correlation from

high-dimensional latent spaces. These approaches achieve state-of-the-art (SOTA) performance on specific datasets. However, they heavily rely on the training datasets, sample distribution shifts would result in significant performance drops, consequently limiting their practical application.

To achieve more stable compression performance for AI-driven methods, Implicit Neural Representation (INR) based compression methods [16, 18, 26, 28, 34] are proposed to generate overfitted encoder and decoder for target samples. Compared to previous AI-based methods, the INR method offers significant advantages in adaptability. In addition, they do not require a large number of training samples with various attributes, as shown in Fig. 1. However, the INR method has two main challenges: 1) the network parameters, especially for the decoder, need to be encoded into bitstreams. Therefore, current INR methods prefer to use simple networks, which generally have a relatively weak fitting capability and are therefore limited to lossy compression [18, 26], resulting in the emptiness of INR based PCGLC, and 2) the overfitting time for the network, which is considered part of the encoding time of INR methods, is too long to limit the potential for real applications [16, 28, 34].

To solve the above problems and fill the gap in PCGLC in terms of INR, we propose a novel framework called Lossless Implicit Neural Representation for Point Cloud Geometry Compression (**LINR-PCGC**). For the first problem, we draw inspiration from the Group of Pictures (GoP) concept in video encoding: the adjacent frames from a point cloud sequence should have close characteristics, and these frames can share a common lightweight network for encoding and decoding¹. By doing so, we can reduce the average bandwidth cost of the network parameters. The GoP-wise operation also leads to an initialization strategy for solving the second problem: the network trained in the previous GoP can be used as the initialized network for overfitting of the next GoP.

The lightweight network used for encoding and decoding is based on multiscale SparseConv [8]: continuously down-sampling until there are only a few dozen or a few hundred points (high scale to low scale), followed by an effective upsampling network called Child Node Prediction (CNP) to estimate the occupancy probability of the higher scale point. Because all scales of a point cloud share the same set of network parameters, we propose the Scale Context Extraction (SCE) module to distinguish different scales and improve compression efficiency. To generate a compact network bitstream, we propose an Adaptive Quantization (AQ) module and a Model Compression (MC) module to quantize and encode the network parameters. A regularization term is introduced for the optimization process to make the training

¹Adjacent frames are only used to share the network parameters. We have not used inter prediction, the order of frames has no impact on our method.

process more stable and the network parameters easier to compress. Our main contributions are as follows.

- We propose the first INR based method for PCGLC called LINR-PCGC. A lightweight multiscale SparseConv network is designed to realize effective point cloud lossless compression.
- Our method allows a group of frames to share a common decoder, reducing the bandwidth cost of the network. And we design an initialization strategy based on GoP which has been shown to save about 65.3% encoding time.
- The results of the experiments show that the proposed method reports superior performance to traditional and AI-driven SOTA methods.

2. Related Works

2.1. Traditional methods

Two main traditional point cloud compression methods are Geometry-based Point Cloud Compression (G-PCC) and Video-based Point Cloud Compression (V-PCC). G-PCC directly encodes point clouds in 3D space using hierarchical structures like octrees, which recursively subdivide the 3D volume to represent point locations. This method is particularly effective for sparse point clouds, such as LiDAR data [12]. However, G-PCC can be computationally demanding due to the need for complex 3D data processing and the lack of efficient temporal prediction for dynamic sequences. On the other hand, V-PCC transforms 3D point clouds into 2D images by segmenting the point cloud into patches, projecting them onto 2D planes, and packing these patches into images for compression using existing video codecs like HEVC. This approach leverages the efficiency and real-time decoding of 2D video compression and is well-suited for dense, dynamic point clouds [6, 19], but struggles with sparse point clouds.

2.2. AI-based methods

PCGCv2 [31] employs a multiscale end-to-end learning framework that hierarchically reconstructs point cloud geometry through progressive re-sampling. It uses sparse convolution neural network (SparseCNN) based autoencoders [3, 4] to compress binary occupancy attributes into downscaled point clouds with geometry and feature attributes. The lowest scale geometry is losslessly compressed using an octree codec, while feature attributes are lossy compressed using a learned probabilistic context model. Subsequently, SparsePCGC [32] introduces a unified framework based on multiscale sparse tensor representation. It processes only the most probable positively occupied voxels (MP-POV) using sparse convolutions, reducing computational complexity. SparsePCGC incorporates a SparseCNN-based Occupancy Probability Approximation (SOPA) model to estimate occupancy probabili-

ties by exploiting cross-scale and same-scale correlations. Additionally, it uses SparseCNN-based Local Neighborhood Embedding (SLNE) to aggregate local variations as spatial priors, further enhancing compression efficiency. SparsePCGC achieves excellent performance in both lossless and lossy compression across several datasets ruled by MPEG, including dense objects and sparse LiDAR point clouds. These methods demonstrate the potential of AI-driven techniques in addressing the challenges of point cloud compression. And the state-of-the-art AI-based geometry point cloud compression method Unicorn-Part I [33] is based on SparsePCGC. There are two reasons why we use SparsePCGC instead of Unicorn-Part I as the baseline: 1) Unicorn-Part I contains many parameters and a more complex network structure, making model simplification very difficult, which is not conducive to initial exploration due to its complexity. 2) Unicorn-Part I is based on SparsePCGC, so subsequent improvements are compatible with our framework construction based on SparsePCGC.

3. Method

3.1. Pipeline

As shown in Fig. 2 (a), we design a multiscale network for point cloud encoding and decoding. 3 steps are required to encode a GoP:

- Initialize. Initialize network by network has been overfitted by the previous GoP. The first GoP is initialized randomly.
- Encode. Overfit the initialized network with target point clouds that need to be compressed. Next, split the model into the point cloud encoder (pc-encoder) and point cloud decoder (pc-decoder). Finally, encode the point cloud with the pc-encoder and encode the pc-decoder parameters with the AQ and MC module.
- Decode. Decode the model parameters of the pc-decoder from the Model Decompress (MD) module. Then, decode point clouds using the pc-decoder.

3.2. Initialization Strategy

The encoding process is carried out on a GoP-wise basis, treating each GoP as a unit of compression. Then, there can be a useful initialization strategy: initialize the network of the current GoP with the network overfitted by the previous GoP. This approach significantly accelerates the overfitting process. For the first GoP, we can either initialize it randomly or using other similar content, which will be discussed in Sec. 4.3.1.

3.3. Network

Let $S = \{x_1, \dots, x_t, \dots, x_M\}$ represent a point cloud sequence with M frames, where $x_t = \{C_t, F_t\}$ represents a single point cloud frame with a time index t , C_t represents

the coordinates of occupied points in x_t , and F_t represents its associated attributes². The point cloud sequence S can be uniformly grouped into multiple GoPs: $G_1 = \{x_1, \dots, x_T\}$, $G_2 = \{x_{T+1}, \dots, x_{2T}\}, \dots, G_r = \{x_{(r-1)T+1}, \dots, x_M\}$.

The basic network architecture is shown in Fig. 2 (b). First, progressively downsample the point cloud until there are only a few dozen or a few hundred points in the lowest scale point cloud. Then, the spatial position of the low-scale point cloud is used to predict the occupancy information of the high-scale point cloud. Each scale has a bitstream to encode the occupancy information using the predicted occupancy probability. Next, the lowest point cloud is converted to bytes directly because there are very few points left, and the decoder network is compressed by the MC module. Finally, the lowest scale point cloud information, the decoder network parameter information, and the occupancy information of each scale make up the final bitstream.

3.3.1. Point Cloud Downsampling

Downsampling is used to express the approximate structure of a point cloud using a lower resolution point cloud.

$$x_t^{i+1} = DS(x_t^i) \quad (1)$$

Here, we use $DS(\cdot) = \text{Maxpooling}(\cdot)$ to do the downsampling.

3.3.2. Scale Context Extraction

To distinguish the different spatial scales of point clouds, we design a module called SCE, depicted in Fig. 2 (d). SCE considers scale embedding (SEMB) as global information and neighbor occupancy as local information. For each scale, an MLP is used after concatenating global information and local information to generate scale context features, which can be formulated as:

$$Nb^{i+1} = \mathcal{F}(\hat{x}_t^{i+1}) \quad (2)$$

$$l_t^{i+1} = \text{MLP}_i(\text{Concate}(Nb^{i+1}, \text{SEMB}(i))) \quad (3)$$

where \mathcal{F} denotes finding the occupancy of the “front, behind, left, right, up, down, self” position for each point. $\text{SEMB}(i)$ denotes a scale embedding that uses an 8-channel implicit feature to extend the scale index i . $\text{Concate}(Nb^{i+1}, \text{SEMB})$ denotes channel connection for Nb^{i+1} and SEMB . MLP_i denotes multilayer perceptron for merging global and local information of the $(i + 1)$ th scale to derive l_t^{i+1} , which denotes the scale information of the current scale and will be used in the following introduced CNP module.

3.3.3. Child Node Prediction

Child Node Prediction (CNP) is designed to upsample point clouds from a lower to a higher spatial scale. The previous method [32] uses transpose convolution to upsample

²Since this work focuses on the compression of point cloud coordinates, F_t of the initialized point cloud is a vector of ones.

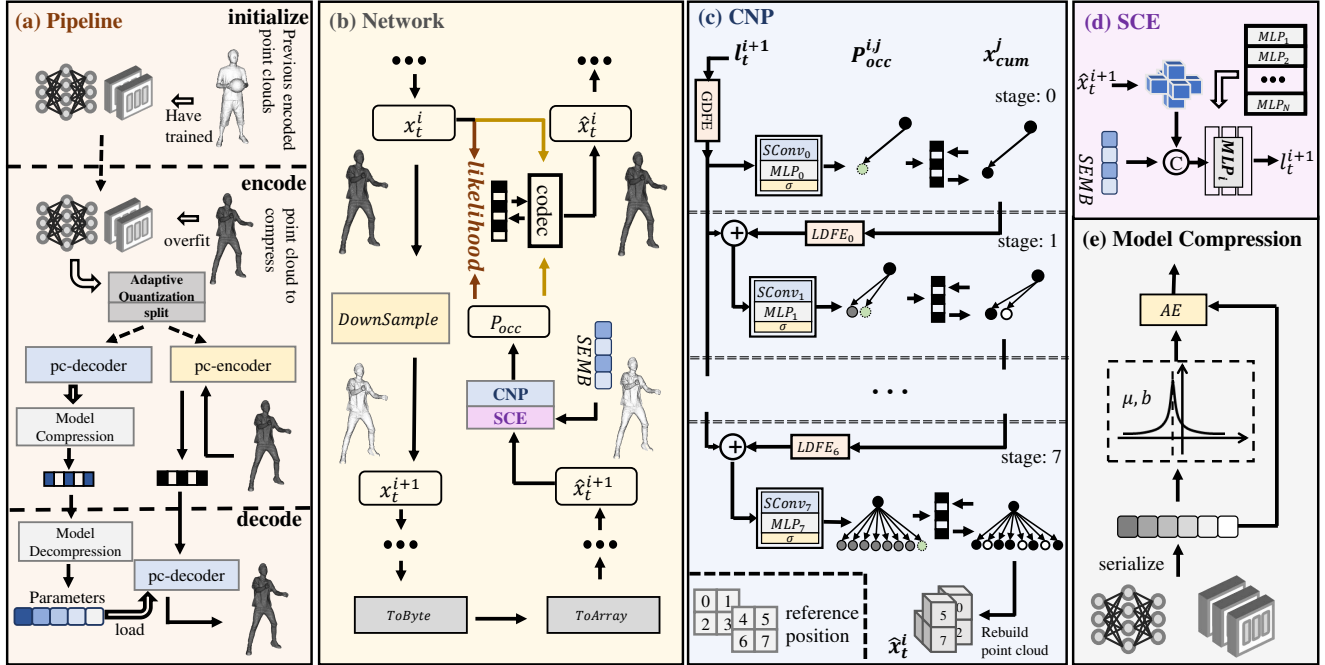


Figure 2. LINR-PCGC Framework. (a) Pipeline. (b) Network. (c) Child Node Prediction. (d) Scale Context Extraction. (e) Model Compression.

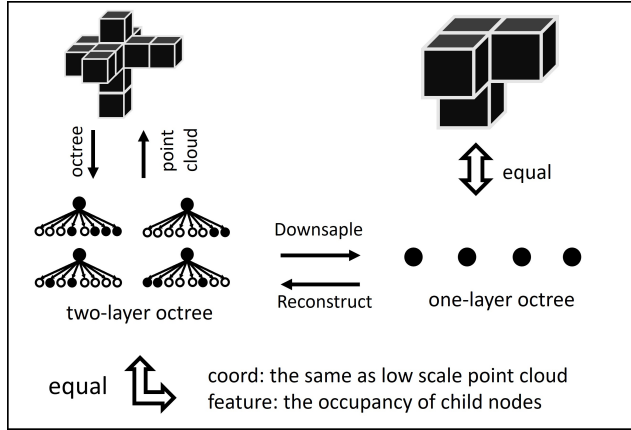


Figure 3. Example of two-layer octree in CNP.

the point cloud, which incurs high memory usage and time complexity. So we propose a new method for upsampling: seeking the child node of the octree as depicted in Fig. 3.

A high-scale point cloud can be used to establish a two-layer octree. The two-layer octree equals a point cloud in which the coordinates are the same as the low-scale point cloud, and the features are the occupancy of the child nodes. In Fig. 3, downsampling can be seen as removing the feature of the two-layer octree. The one-layer octree and the low-scale point cloud are the same. Reconstruction of the high-scale point cloud equals reconstructing the **occupancy of**

the child nodes. In this way, the point involved in the whole process is the low-scale point cloud, which can significantly simplify computational complexity.

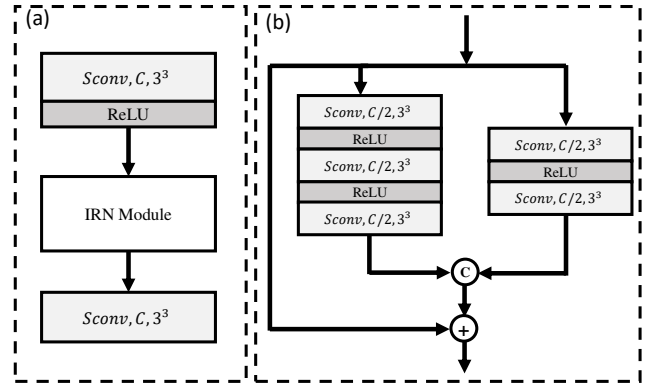


Figure 4. (a) The structure of GDHE/LDFE. (b) The structure of IRN Module. $Sconv, C, k^3$ denotes a Sparse convolution layer with C output channels and kernel size k , respectively.

To improve the accuracy of the prediction and reduce the bitstream, we predict the occupancy of the child nodes in a channel-wise rule. The decoded child nodes serve as context information for the child nodes to be decoded. We use an 8-stage method to reconstruct a high-scale point cloud as shown in Fig. 2 (c). Let j ($0, 1, 2, \dots, 7$) represent the stage index. x_{cum}^j represents the reconstructed point cloud in j th stage. $P_{occ}^{i,j}$ represents the predicted probability of each child

node in i th scale j th stage. l_t^{i+1} denotes the $(i+1)$ th scale point cloud with scale information derived from the SCE module. Two feature extraction modules, GDFE (Global Deep Feature Extraction) and LDFE (Local Deep Feature Extraction), are proposed as shown in Fig. 4.

We first use GDFE and LDFE modules to extract two latent features, i.e., G_{feat} and L_{feat}^{j-1} . Then we merge two features as input to a classification network, consisting of a sparse convolution $Sconv_j$, a multilayer perceptron MLP_j and a nonlinear activation unit σ (*Sigmoid*), to obtain $P_{occ}^{i,j}$ that describes the probability of child node occupancy. Specifically, $Sconv_j$ is to extract local neighborhood information, MLP_j is to efficiently integrate point-wise features, and *Sigmoid* is used to normalize the features to ensure that the occupancy probability falls within the range $[0,1]$. Finally, the estimated occupancy probability $P_{occ}^{i,j}$ is used for arithmetic coding of the ground truth occupancy values $x_t^{i,j}$. Meanwhile, the cross entropy between $x_t^{i,j}$ and $P_{occ}^{i,j}$ is calculated to estimate the bitstream size for encoding $x_t^{i,j}$ with $P_{occ}^{i,j}$, where $x_t^{i,j}$ is the ground truth in i -th scale j -th stage. The detailed process of CNP during training can be found in Algorithm 1.

3.4. Adaptive Quantization

To make it easier to encode the network parameters, we need to quantize them using the AQ module. Let p_{dec} represent the parameters related to the pc-decoder.

$$normalize(p_{dec}) = \frac{p_{dec} - \min(p_{dec})}{\max(p_{dec}) - \min(p_{dec})} \quad (4)$$

$$p_{quant} = round(normal(p_{dec}) \times (2^B - 1)) \quad (5)$$

Where *normal* is a normalization for p_{dec} to ensure its values within the range $[0, 1]$. Then, it multiplies by $2^B - 1$, followed by a round operation, to quantize it to B bits. The dequantization is:

$$p_{dequant} = \frac{p_{quant}}{2^B - 1} \times (\max(p_{dec}) - \min(p_{dec})) + \min(p_{dec}) \quad (6)$$

3.5. Model Compression

To better understand the parameter distribution of the model, we plot and analyze its frequency histogram in Fig. 5. It is evident that the model parameters trained with regularization terms closely follow a Laplace distribution. Consequently, in quantized model parameters' entropy coding, we employ a Laplace distribution as an approximation of their actual distribution, denoted as f in Eq. (7). The mean and scale parameters are estimated using Eq. (8) and Eq. (9), which

Algorithm 1: Process of CNP during training

Input: l_t^{i+1} extracted from SCE.
Output: Estimated bitstream size E of point cloud.

- 1 Calculating the global feature of all stages,
 $G_{feat} = GDFE(l_t^{i+1});$
- 2 let bitstream size $E = 0;$
- 3 **for** each j in $0, \dots, 7$ **do**
- 4 **if** $j == 0$ **then**
- 5 Use global feature to represent the merge
feature directly, $F_{merge}^j = G_{feat};$
- 6 **end**
- 7 **else**
- 8 Derive local feature from decoded child node
occupancy by LDEF,
 $L_{feat}^{j-1} = LDEF_{j-1}(x_{cum}^{j-1});$
- 9 Add global feature and local feature together
to get the merge feature,
 $F_{merge}^j = G_{feat} + L_{feat}^{j-1};$
- 10 **end**
- 11 Calculate the child node occupancy probability
 $P_{occ}^{i,j} = \sigma(MLP_j(SConv_j(F_{merge}^j)));$
- 12 Use entropy to estimate the bitstream size of
current stage, $E_j = Entropy(x_t^{i,j}, P_{occ}^{i,j});$
- 13 Cumulative the bitstream size, $E = E + E_j;$
- 14 **if** $j == 0$ **then**
- 15 Record the decoded child node occupancy
 $x_{cum}^j = x_t^{i,j};$
- 16 **end**
- 17 **else**
- 18 Accumulated the decoded child node
occupancy $x_{cum}^j = Concat(x_{cum}^{j-1}, x_t^{i,j}),$
Concat represents channel connection;
- 19 **end**
- 20 **end**

are encoded into bitstream as side information.

$$f(p_{quant}; \mu, b) = \frac{1}{2b} \exp\left(-\frac{|p_{quant} - \mu|}{b}\right) \quad (7)$$

$$\mu = \frac{1}{n} \sum (p_{quant}) \quad (8)$$

$$b = \frac{1}{n} \sum |p_{quant} - \mu| \quad (9)$$

3.6. Loss Function

The loss of the training process is derived from the CNP module, which calculates the cross entropy between $x_t^{i,j}$ and $P_{occ}^{i,j}$ of each stage on each spatial scale as an estimation of bitstream size. As the ground truth occupancy $x_t^{i,j}$ is

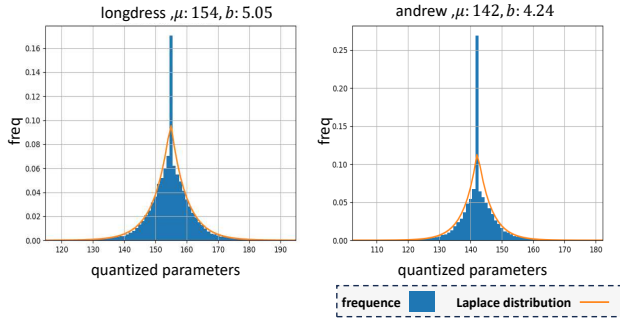


Figure 5. Distribution of quantized network parameters.

binary, the Binary Cross-Entropy (BCE) is used,

$$L_{BCE}^{i,j} = Entropy(x_t^{i,j}, P_{occ}^{i,j}) \quad (10)$$

$$\mathcal{L} = \sum_{i=0}^N \sum_{j=0}^7 L_{BCE}^{i,j} + \lambda \|\theta\|_2^2 \quad (11)$$

where $Entropy(p, q) = -(p \log_2(q) + (1-p) \log_2(1-q))$, θ represents network parameters, λ is an L2 regularization coefficient. Since we only use multi-frames to share network parameters without utilizing inter-frame features, each frame in a GoP can be optimized separately, and there is no need to sum in the t dimension.

4. Experiment

4.1. Experiment Configuration

Training. Our model is implemented in Pytorch [25] and MinkowskiEngine [8]. The number of spatial scales is determined by the downsampling times of the first frame until its point number is less than or equal to 64. We use the Adam optimizer [21] with a learning rate decayed from 0.01 to 0.0004. We set the number of frames per sequence to 96, the GoP size to 32 and the bit depth B in AQ to 8. Further details on hyperparameters are provided in the Appendix. We train the first GoP for 6 epochs and subsequent GoPs for 1 to 6 epochs on a single NVIDIA RTX 3090 GPU with AMD EPYC 7502 CPUs.

Dataset. As seen in Fig. 6, we utilize three different dynamic human datasets with geometric bit-depth 10: 1) 8i Voxelized Full Bodies (8iVFB) dataset [9], which contains 4 sequences at a frame rate of 30 fps over 10 seconds; 2) OwlII Dynamic Human DPC (OwlII) [20], which has 4 sequences with 30 fps over 20 seconds; 3) Microsoft voxelized upper bodies (MVUB) [22], which provides 5 sequences of upper body movements. We select the first 96 frames from each sequence in the above datasets.

Baseline. For traditional compression methods, we select G-PCC v23 [14] and V-PCC v23 [15] as baselines. For AI-based methods, we choose SparsePCGC. For fairness, we

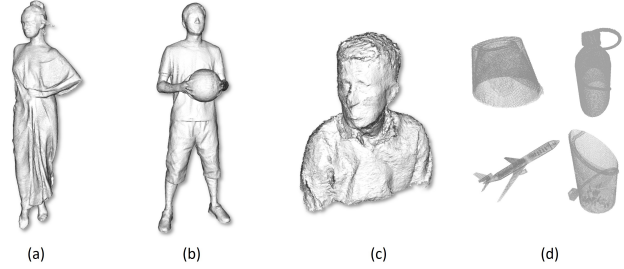


Figure 6. (a) Longdress from 8iVFB. (b) Basketball Player from OwlII. (c) Andrew from MVUB. (d) Samples in Shapenet.

use pre-trained models trained on ShapeNet provided by the authors [24].

Metric. For the lossless compression of point cloud geometric information, our primary metric is the average bitstream size per point, denoted as bits per point (bpp). In addition, we evaluate the encoding and decoding times to assess the computational efficiency of the compression process. Furthermore, we present how the bitstream size varies with encoding time, providing insights into the trade-off between compression efficiency and computational resources. And we randomly initialize the first GoP of each sequence.

4.2. Experiment Result

Fig. 7 shows the encoding times vs. bpp curves for the 8iVFB, OwlII, and MVUB datasets. As the first GoP is randomly initialized, we tend to train more epochs, i.e., F , for the first GoP. We choose different F values in Fig. 7, and the different points with the same F denote 1 to 6 training epochs for subsequent GoPs. We can observe that a longer encoding time can achieve a higher compression ratio. In order to observe the compression effect under a comparable encoding time state and the compression effect under a relatively sufficient encoding time state, we collect the first and last points in the time dimension of each figure as quantitative results and construct Tabs. 1 to 3. The columns in tables called “ours” and “ours 2” are the first and last sampling points (e.g., training 1 and 6 epochs for subsequent GoPs), respectively.

According to the quantitative results, we can observe that: 1) our method can achieve the best compression ratio in a comparable encoding time for all datasets; 2) our method can maintain a fast decoding time, which can be about half that of G-PCC or SparsePCGC (marked as S.PCGC); 3) our method can maintain stable compression performance on datasets with various coordinate distributions. Especially, in MVUB our method achieves a 13.33% gain compared to SparsePCGC. Fig. 6 shows coordinate distributions of different datasets, where MVUB has fewer smooth surfaces and more virtual edges than other datasets. Existing AI-based methods like SparsePCGC exhibit poor adaptabil-

	G-PCC	S.PCGC	V-PCC	ours	ours 2
longdress	0.74	0.619	1.384	0.618	0.571
loot	0.69	0.586	1.27	0.57	0.521
red&black	0.81	0.676	1.539	0.689	0.626
soldier	0.734	0.619	1.469	0.588	0.538
bpp (avg)	0.743	0.625	1.415	0.616	0.564
r.t. bpp	100	84.044	190.411	82.925	75.894
w/o over.	-	-	-	0.477	0.434
enc. time	2.72	2.202	194.261	2.464	16.423
dec. time	0.923	1.048	2.304	0.501	0.459

Table 1. Quantitative results on 8iVFB dataset, where bpp (avg) denotes the average bpp of all sequences in the dataset, r.t. bpp denotes the relative bpp (%) of other methods over G-PCC, and w/o over. denotes the encoding time without overfitting time of our method. All the times are in seconds. The best and the second best results are denoted by red and blue.

	G-PCC	S.PCGC	V-PCC	ours	ours 2
basketball	0.578	0.466	1.097	0.452	0.411
dancer	0.606	0.485	1.192	0.473	0.431
exercise	0.585	0.472	1.104	0.460	0.417
model	0.592	0.485	1.155	0.475	0.431
bpp (avg)	0.59	0.477	1.137	0.465	0.423
r.t. bpp	100	80.815	192.683	78.759	71.599
w/o over.	-	-	-	0.402	0.395
enc. time	2.24	1.932	146.295	2.493	14.174
dec. time	0.725	0.926	1.985	0.422	0.415

Table 2. Quantitative results on OwlII dataset. The best and the second best results are denoted by red and blue.

ity in handling large data distribution shifts. 4) If there is enough encoding time (about ten to twenty seconds to encode a frame on average), a larger compression ratio can be obtained, which reduces the bitstream by approximately 9.70% in 8iVFB, 11.40% in OwlII, 21.95% in MVUB than SparsePCGC; about 24.11%, 28.40%, 21.21% than G-PCC.

Bitstream allocation and time composition. We illustrate bitstream allocation and the time composition in Fig. 8, and present the corresponding proportions in Tab. 4. Bitstream allocation indicates that higher spatial scales result in larger bitstream consumptions, as point clouds at higher spatial scales contain more geometric details and thus require more information to predict occupancy. Network parameters occupy an ignorable proportion of the bitstream as they are shared across frames in a GoP. The time composition shows that most of the time is still spent in the hierarchical point cloud reconstruction process, rather than the encoding and decoding of network parameters and the lowest scale point cloud pc_{low} .

	G-PCC	S.PCGC	V-PCC	ours	ours 2
andrew10	0.941	0.947	1.611	0.833	0.748
david10	0.898	0.909	1.462	0.778	0.704
phil10	0.969	0.966	1.636	0.841	0.768
ricardo10	0.904	0.925	1.519	0.802	0.705
sarah10	0.892	0.901	1.486	0.777	0.702
bpp	0.921	0.93	1.543	0.806	0.725
r.t. bpp	100	100.947	167.561	87.548	78.788
w/o over.	-	-	-	0.524	0.513
enc. time	3.951	3.06	213.192	2.712	18.564
dec. time	1.284	1.456	3.071	0.554	0.544

Table 3. Quantitative results on MVUB dataset. The best and the second best results are denoted by red and blue.

Metrics	Decoder	pc_{low}	scale 2-6	scale 1	scale 0
bpp	0.73	0.17	5.83	18.10	75.17
enc. time	0.47	8.58	30.47	14.92	45.56
dec. time	0.52	0.00	31.60	16.25	51.63

Table 4. Statistics of bitstream proportion and encoding/decoding time proportion (%) in MVUB.

4.3. Ablation Study

We use the average performance on 8iVFB, OwlII and MVUB to evaluate the effectiveness of different settings of LINR-PCGC. The following figures and tables are shown on 8iVFB and MVUB. Tables and figures on OwlII dataset of this part are in the appendix.

4.3.1. Ablation of initialization strategies

To demonstrate the acceleration effect of the initialization strategy on training, we set three different initialization methods: 1) random initialization for each GoP (rand.); 2) randomly initialize the first GoP and use the first GoP to initialize subsequent GoPs (ini.); and 3) use other similar sequences to initialize the first GoP, e.g., *basketball* initializes the first GoP of *dancer*, and use the first GoP to initialize subsequent GoPs (fur. ini.). Fig. 9 shows training time vs. bpp curves of the three initialization methods. And by integrating the overlapping parts of the three curves along the bpp axis, we can estimate the average training time for the three methods and calculate the time ratios shown in Tab. 5. Leveraging the correlation between GOPs and similarity among sequences can significantly improve training efficiency, i.e., 65.3% and 76.0% of average time saving for method 2) and 3) compared to method 1).

4.3.2. Ablation of modules

To demonstrate the effectiveness of each module, we start by retaining only the CNP module and then sequentially adding other modules until the complete LINR-PCGC without the

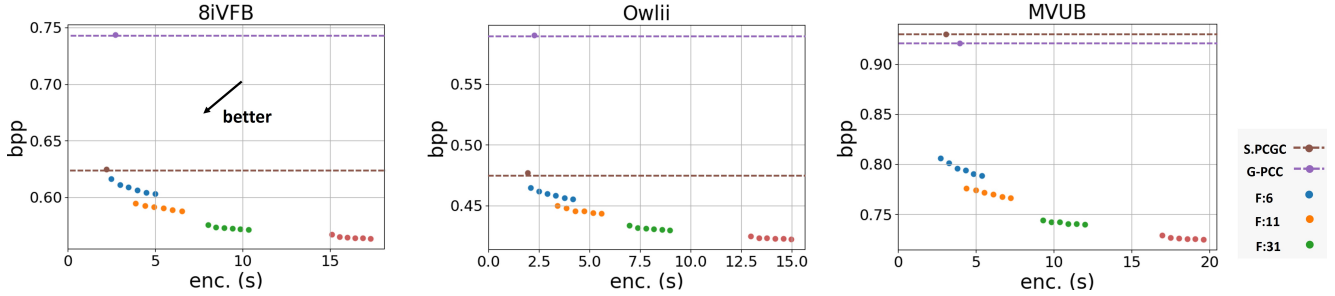


Figure 7. Encoding times vs. bpp curves with different training epochs for the first GoP and subsequent GoPs.

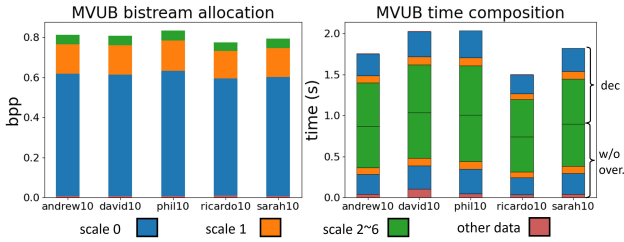


Figure 8. Bitstream allocation and time composition in MVUB. “Other data” in the figure includes the size of network parameters together with the size of the lowest scale point cloud pc_{low} . And w/o over. denotes the encoding time without overfitting time.

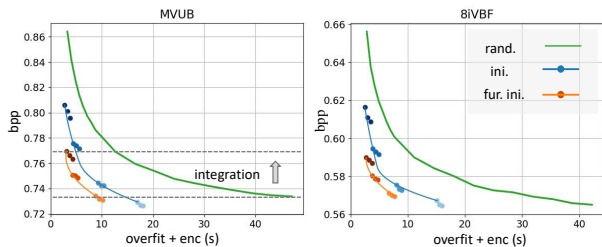


Figure 9. Training times vs. bpp curves with randomly initializing each GoP (rand.), randomly initialize the first GoP (ini.), and using similar sequences to initialize the first GoP (fur. ini.).

	8iVFB	OwlII	MVUB	avg.
ini.	36.0	34.4	33.7	34.7
fur. ini.	22.9	29.2	20.0	24.0

Table 5. Relative time (%) that ini. and fur. ini. take compared to rand.

initialization strategy is implemented. The results are shown in Fig. 10. Then we integrate the overlapping parts of all the curves over time to obtain the average bpp ratios relative to the green curve, as shown in Tab. 6. AQ and MC modules can reduce 8.1% bpp, while SCE can further reduce 3.1% bpp, indicating the effectiveness of the proposed modules.

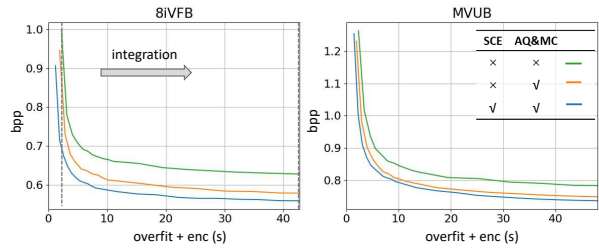


Figure 10. Impact of each module in LINR-PCGC.

SCE	AQ&MC	r.t. bpp (%)↓
x	x	100.0
x	✓	91.9
✓	✓	88.8

Table 6. Impact of each module in LINR-PCGC, where r.t. bpp denotes relative bpp (%) over the method w/o. SCE, AQ&MC.

4.4. Conclusion

We propose LINR-PCGC to compress a sequence of point clouds with the basic architecture of INR methods. So, our method inherits the biggest advantage of INR: it does not rely on specific data distributions to work. Additionally, we employ an initialization strategy for acceleration and thus achieve an encoding time comparable to non-INR methods. In addition, the lightweight network design ensures a shorter decoding time. Further, we will include inter-frame prediction for temporal redundancy removal and extend to lossy compression as our method reduces restrictions of network size in INR methods.

5. Acknowledgments

This paper is supported in part by National Natural Science Foundation of China (62371290), National Key R&D Program of China (2024YFB2907204), the Fundamental Research Funds for the Central Universities of China, and STCSM under Grant (22DZ2229005).

References

- [1] V-pcc codec description. ISO/IEC JTC1/SC29/WG7 MDS20352/N00100, 2021. 1
- [2] G-pcc 2nd edition codec description. ISO/IEC JTC1/SC29/WG7 MDS24176/N00942, 2024. 1
- [3] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 2
- [4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 2
- [5] Sourav Biswas, Jerry Liu, Kelvin Wong, Shenlong Wang, and Raquel Urtasun. Muscle: Multi sweep compression of lidar using deep entropy models. In *Advances in Neural Information Processing Systems*, pages 22170–22181. Curran Associates, Inc., 2020. 1
- [6] Chao Cao, Marius Preda, and Titus Zaharia. 3d point cloud compression: A survey. In *Proceedings of the 24th International Conference on 3D Web Technology*, page 1–9, New York, NY, USA, 2019. Association for Computing Machinery. 1, 2
- [7] Chao Cao, Marius Preda, Vladyslav Zakharchenko, Euee S Jang, and Titus Zaharia. Compression of sparse and dense dynamic point clouds—methods and standards. *Proceedings of the IEEE*, 109(9):1537–1558, 2021. 1
- [8] 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. 2, 6
- [9] 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) m40059/M74006, 2017. 6
- [10] Tingyu Fan, Linyao Gao, Yiling Xu, Dong Wang, and Zhu Li. Multiscale latent-guided entropy model for lidar point cloud compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(12):7857–7869, 2023. 1
- [11] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octtattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, pages 625–633, 2022. 1
- [12] Danillo Graziosi, Ohji Nakagami, Shinroku Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali 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*, 9:e13, 2020. 2
- [13] Danillo Graziosi, Ohji Nakagami, Shinroku Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali 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*, 9(e13), 2024. 1
- [14] MPEG Group. Mpeg-pcc-tmc13. <https://github.com/MPEGGroup/mpeg-pcc-tmc13/releases/tag/release-v23.0-rc2>, . Accessed: 2024-09-10. 6
- [15] MPEG Group. Mpeg-pcc-tmc2. <https://github.com/MPEGGroup/mpeg-pcc-tmc2/releases/tag/release-v23.0>, . Accessed: 2024-09-10. 6
- [16] Yueyu Hu and Yao Wang. Learning neural volumetric field for point cloud geometry compression. In *2022 Picture Coding Symposium*, pages 127–131, 2022. 2
- [17] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1313–1323, 2020. 1
- [18] Berivan Isik, Philip A Chou, Sung Jin Hwang, Nick Johnston, and George Toderici. Lvac: Learned volumetric attribute compression for point clouds using coordinate based networks. *Frontiers in Signal Processing*, 2:1008812, 2022. 2
- [19] 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*, 36(3):118–123, 2019. 2
- [20] Cao Keming, Xu Yi, Lu Yao, and Wen Ziyu. OwlII dynamic human mesh sequence dataset. Document ISO/IEC JTC1/SC29/WG11 m42816, 2018. 6
- [21] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–15, San Diego, CA, USA, 2015. ICLR. 6
- [22] Charles Loop, Qin Cai, Sergio Orts Escolano, and Philip A. Chou. Microsoft voxelized upper bodies - a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) m38673/M72012, 2016. 6
- [23] 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*, pages 1–6. IEEE, 2021. 1
- [24] NJUVISION. Sparsepcgc. <https://github.com/NJUVISION/SparsePCGC>. Accessed: 2025-03-01. 6
- [25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6
- [26] Francesca Pistilli, Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Signal compression via neural implicit representations. In *2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3733–3737, 2022. 2
- [27] Maurice Quach, Jiahao Pang, Dong Tian, Giuseppe Valenzise, and Frédéric Dufaux. Survey on deep learning-based point cloud compression. *Frontiers in Signal Processing*, 2: 846972, 2022. 1
- [28] Hongning Ruan, Yulin Shao, Qianqian Yang, Liang Zhao, and Dusit Niyato. Point cloud compression with implicit

- neural representations: A unified framework. In *2024 IEEE/CIC International Conference on Communications in China*, pages 1709–1714, 2024. [2](#)
- [29] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018. [1](#)
- [30] 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. [1](#)
- [31] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference*, pages 73–82, 2021. [2](#)
- [32] 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*, 45(7):9055–9071, 2023. [2](#), [3](#)
- [33] Jianqiang Wang, Ruixiang Xue, Jiaxin Li, Dandan Ding, Yi Lin, and Zhan Ma. A versatile point cloud compressor using universal multiscale conditional coding – part i: Geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1):269–287, 2025. [1](#), [3](#)
- [34] Wei Zhang, Dingquan Li, Ge Li, and Wen Gao. Lightweight super resolution network for point cloud geometry compression. In *2024 Data Compression Conference*, pages 602–602, 2024. [2](#)