Towards Practical Lossless Neural Compression for LiDAR Point Clouds

Pengpeng Yu[1,2⋆], Haoran Li[1,2⋆], Runqing Jiang[1], Dingquan Li[2], Jing Wang[2†], Liang Lin[1,2], and Yulan Guo[1,2†]

Sun Yat-sen University, China
Pengcheng Laboratory, China
wangj@pcl.ac.cn
guoyulan@sysu.edu.cn

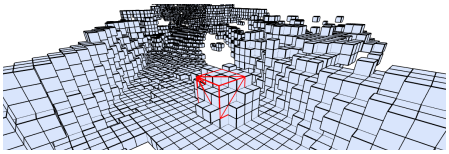
Abstract. LiDAR point clouds are fundamental to various applications, yet the extreme sparsity of high-precision geometric details hinders efficient context modeling, thereby limiting the compression speed and performance of existing methods. To address this challenge, we propose a compact representation for efficient predictive lossless coding. Our framework comprises two lightweight modules. First, the Geometry Re-Densification Module iteratively densifies encoded sparse geometry, extracts features at a dense scale, and then sparsifies the features for predictive coding. This module avoids costly computation on highly sparse details while maintaining a lightweight prediction head. Second, the Cross-scale Feature Propagation Module leverages occupancy cues from multiple resolution levels to guide hierarchical feature propagation, enabling information sharing across scales and reducing redundant feature extraction. Additionally, we introduce an integer-only inference pipeline to enable bit-exact cross-platform consistency, which avoids the entropy-coding collapse observed in existing neural compression methods and further accelerates coding. Experiments demonstrate competitive compression performance at real-time speed. Code is available at https://github.com/pengpeng-yu/FastPCC.

Keywords: Point cloud compression · Real-time codec · Cross-platform consistency

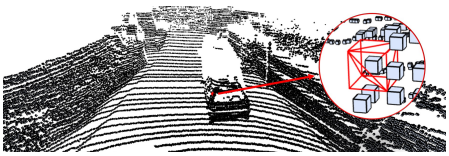
1 Introduction

With the rapid advancement of 3D sensing technologies, massive amounts of point cloud data have been accumulated in various fields such as autonomous driving and mapping [58]. This surge in data volume has led to an increasing demand for precise point cloud compression (PCC). Currently, most PCC

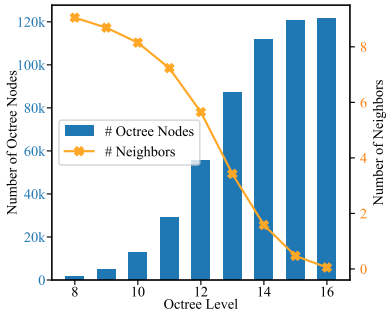
⋆ Equal contribution.
† Corresponding author.



(a) Level-8 octree of a KITTI point cloud



(b) Level-12 octree of a KITTI point cloud



(c) Number of nodes and neighbors vs. level.

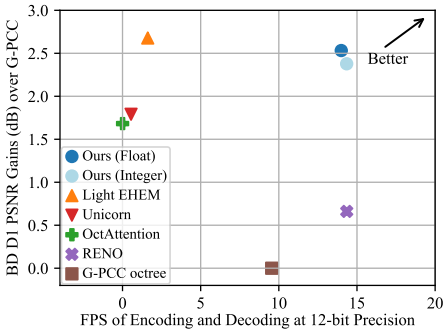
Fig. 1: Illustration of the High-Resolution Contextual Sparsity (HRCS) phenomenon: (a) and (b) depict the voxelized octree representations at levels 8 and 12 for a point cloud from the KITTI dataset, respectively. The red bounding box highlights a $3\times3\times3$ neighborhood. (c) quantifies HRCS on the KITTI dataset, where the average number of neighbors per node decreases sharply with increasing octree level.

methods represent raw coordinate data using quantized structures such as range images [24, 43, 50, 51, 64], voxels [17, 19, 26, 34, 38, 48, 59, 65], or octrees [2, 5, 11, 12, 18, 39, 42, 56], and then apply techniques like prediction or transformation to achieve compression.

Although existing PCC methods have made significant progress in rate-distortion (RD) performance, their foundational representations, namely voxels or octrees, exhibit inherent limitations in high-precision compression scenarios. Both representations quantize a 3D space into discrete volumes, marking each as occupied only if it contains at least one point. However, as shown in Fig. 1a and Fig. 1b, as the quantization resolution increases, the local neighborhood around a given voxel becomes increasingly sparse, drastically reducing the availability of contextual information. We term this phenomenon **High-Resolution Contextual Sparsity (HRCS)**. In such cases, occupancy prediction becomes increasingly difficult due to the lack of local context.

To quantify HRCS, we conducted a statistical analysis on all frames of the KITTI dataset. For the octree of each sample, we collected two key statistics: (i) the total number of nodes at each level, and (ii) the average number of occupied neighbors within a $3 \times 3 \times 3$ neighborhood. As illustrated in Fig. 1c, with increasing resolution (*i.e.*, at deeper octree levels), the average number of neighbors per node drops sharply. At certain levels, the average number of neighbors even falls below one. Notably, this decline exhibits a marked inflection point at a specific octree level, indicating a nonlinear loss of contextual richness.

To address HRCS without sacrificing coding efficiency, we propose a Geometry Re-Densification (GRED) strategy. Instead of directly predicting sparse high-resolution nodes, GRED traces back to a shallower level, where geometry remains relatively dense, and performs dense-to-sparse feature transformation through lightweight convolutions and upsampling. The resulting features are



(a) BD-PSNR gains vs. the speed of encoding–decoding process on KITTI at 12-bit precision.

(b) Non-deterministic floating-point inference leads to cross-platform decoding failure.

(c) Integer-only inference enables bit-exact cross-platform decoding.

Fig. 2: (a) RD performance versus coding speed on KITTI at 12-bit precision, evaluated on an AMD EPYC 7R32 CPU and an NVIDIA RTX 4090 GPU. (b) Decoding failure caused by non-deterministic floating-point computation, where encoding is performed on an NVIDIA RTX 4090 GPU and decoding on an RTX 5880 GPU. The reconstructed point cloud collapses into an approximately uniform distribution in 3D space. (c) Bit-exact decoding achieved with the proposed integer-only inference pipeline.

spatially aligned with the target level and used to facilitate occupancy prediction. This process forms an iterative *dense* → *sparse* → *predictive coding* cycle across octree levels, executed from coarse to fine resolutions. Built upon GRED, we further introduce a Cross-Scale Feature Propagation (XFP) module to fuse dense features from shallow levels with sparse geometric cues from deeper levels, enabling effective cross-scale interaction while maintaining computational efficiency.

Beyond compression performance and coding latency, practical neural codecs must guarantee cross-platform bitstream decodability. However, floating-point inference is inherently non-deterministic across heterogeneous devices and software stacks, and even minor numerical discrepancies in probability prediction may catastrophically break entropy coding. To address this practical challenge, we further develop an integer-only inference pipeline, ensuring bit-exact cross-platform consistency while enabling efficient execution.

In the following sections, we first review related work in PCC, followed by a detailed description of GRED, XFP, and our integer-only inference pipeline. Extensive experiments on KITTI [15] and Ford [32] demonstrate the effectiveness of the proposed method in terms of compression performance, coding efficiency, and cross-platform consistency.

2 Related work

This section reviews representative PCC methods and briefly discusses recent advances in cross-platform consistency for neural codecs. According to the un-

derlying geometric representation, existing PCC approaches can be broadly categorized into two groups: (1) voxel-based PCC and (2) octree-based PCC.

Voxel-based PCC. Voxel-based approaches split the point cloud into sufficiently small voxels, utilizing sparse convolution [44] to optimize memory usage. Based on voxels, many PCC techniques have emerged [29–31,33,47,55,61,62]. For example, Wang *et al*. [49] proposed a voxel-based geometry compression method that partitions point clouds into non-overlapping 3D cubes and leverages a variational autoencoder-driven convolutional neural network to extract latent features and hyperpriors for entropy coding. Recently, Wang *et al*. [48] proposed a universal multiscale conditional coding framework, Unicorn, which leverages sparse tensors from voxelized point clouds and cross-scale temporal priors to enhance geometry compression. Zhang *et al*. [60] proposed a dynamic point cloud compression framework based on voxelized data, featuring a slimmable architecture with multiple coding routes for rate-distortion optimization, and a coarse-to-fine motion module to improve inter-frame prediction.

Octree-based PCC. Octree-based approaches typically construct an L-level octree by recursively subdividing the point cloud within a pre-defined bounding volume, and achieve compression by predicting the occupancy status of each octree node. Based on the octree structure, many PCC techniques have emerged [14,16,22,27,52,54]. For example, Huang *et al*. [18] proposed an octree-based compression method that leverages a tree-structured conditional entropy model to exploit sparsity and structural redundancy in LiDAR point clouds. Similarly, Fu *et al*. [11] proposed an octree-based deep learning framework that encodes point clouds by leveraging rich sibling and ancestor contexts with an attention mechanism. Cui *et al*. [9] proposed OctFormer, which constructs node sequences with non-overlapping context windows and shares attention results to reduce computation. Song *et al*. [42] proposed an octree-based entropy model with a hierarchical attention mechanism and grouped context structure, reducing the complexity and decoding latency of large-scale autoregressive models.

Cross-platform consistency. Prior studies in neural image/video compression reveal that even tiny floating-point computation errors can cause encoder-decoder mismatch and lead to decoding failures [1,8,20,23,46,63]. This issue typically stems from entropy coding, which requires the encoder and decoder to share exactly the same probability distribution. A principled solution is model integerization, which enforces bit-exact deterministic computations [1,8,20,23]. Unlike conventional model quantization [21,25,53], which primarily targets computationally intensive neural modules for inference acceleration, model integerization focuses on fully integerizing the neural modules along the entropy-coding path to ensure cross-platform consistent coding. Alternative approaches transmit auxiliary calibration information to correct cross-platform mismatch, at the cost of extra side information [46]. Compared with image/video codecs, neural PCC often involves sparse operators and data-dependent control flow, which complicates deterministic deployment across heterogeneous platforms. While cross-platform consistency is increasingly studied for image/video neural codecs, it remains largely underexplored in neural PCC.

In summary, both voxel-based and octree-based PCC approaches have achieved remarkable progress, with learning-based methods significantly improving RD performance over traditional codecs [4,13,28,37,40,41,50]. However, under high-resolution settings, both representations tend to suffer from HRCS. See the supplementary materials for further discussion. A representative attempt to alleviate sparsity is GRASP-Net [34], which introduces PointNet-based [35,36] analysis for sparse geometry. Nevertheless, such heterogeneous designs introduce considerable computational complexity. Moreover, existing PCC studies primarily focus on RD performance, while the practical requirement of cross-platform deterministic coding remains largely overlooked. This motivates our work to jointly address HRCS and cross-platform consistency within a unified neural PCC framework.

3 Method

To address the HRCS problem while satisfying the practical requirements of real-time LiDAR PCC, we propose a fast octree-based encoding framework. The overall architecture is illustrated in Fig. 3. The proposed framework consists of four key components: octree construction, prior construction, cross-scale feature propagation, and entropy coding. In the following, we first introduce the Cross-Scale Feature Propagation module with its core Geometry Re-Densification design, and then describe the integer-only inference pipeline that guarantees cross-platform consistent coding.

3.1 Geometry Re-Densification Module

The irregular and unordered nature of LiDAR point clouds poses significant challenges for efficient processing on modern hardware architectures. To better exploit existing hardware, most compression methods convert raw point clouds into octree structures. By recursively dividing space into eight subcells at each level, octrees provide a compact and hierarchical representation of geometry. The maximum level L of the octree controls reconstruction fidelity. Using octrees, existing codecs can perform progressive, dense-to-sparse predictive lossless coding of occupancy codes, modeling the distribution by exploiting contexts from encoded sibling nodes and ancestral nodes to minimize storage.

Despite the compact and regular structure of octree representations, they face the challenge of HRCS in encoding high-resolution LiDAR point clouds. To address this problem, we propose a **Geometry Re-Densification (GRED)** module and integrate it into the dense-to-sparse progressive coding pipeline. At each HRCS-affected level, GRED downsamples sparse occupancy codes to obtain denser contextual features, and then reverts to the original sparse domain for predictive coding. For each HRCS-affected level, the module performs:

1. *Re-Densification.* Downsample the occupancy codes of the last encoded level into a denser octree level, producing an aligned dense feature map with zero-padding for empty nodes.

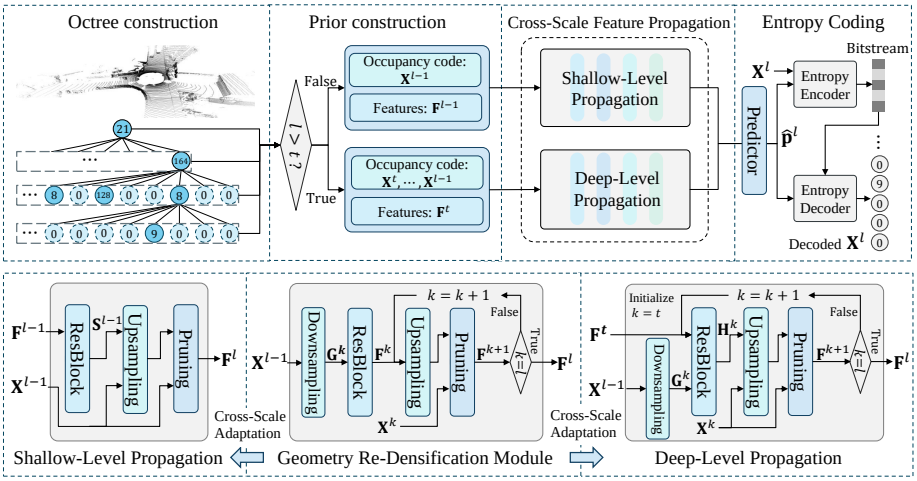


Fig. 3: Pipeline of compressing a single octree level in the proposed lossless LiDAR PCC framework. The framework consists of four main stages: octree construction, prior construction, cross-scale feature propagation, and entropy coding. The cross-scale feature propagation module comprises two key components: the shallow-level propagation block and the deep-level propagation block, both adapted from the geometry re-densification module to exploit cross-scale features.

2. *Feature Extraction.* Apply lightweight convolutions to the dense feature map to extract rich local spatial representations.
3. *Re-Sparsification.* Recursively upsample and prune the dense features using the encoded occupancy codes, producing a sparse feature map aligned with the nodes at the current level.
4. *Prediction & Coding.* Use a multilayer perceptron-based predictor on the sparse feature map to estimate the occupancy distribution over 255 classes (8-bit occupancy patterns), and entropy-encode the occupancy codes.

Without loss of generality, we denote an octree as $\mathbf{X} = \{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^L\}$, where L is the maximum octree level, $\mathbf{X}^l \in \{1, \ldots, 255\}^{N^l}$ represents the occupancy sequence of all nodes at level l, and N^l denotes the number of nodes at that level. Lossless compression approximates the true occupancy distribution $P(\mathbf{X})$ with an estimated distribution $Q(\mathbf{X})$ by minimizing the cross-entropy:

$$H(P, Q) = \mathbb{E}_{P(\mathbf{X})} \left[-\log Q(\mathbf{X}) \right]. \tag{1}$$

Standard octree-based codecs typically estimate the distribution of occupancy codes in a layer-wise autoregressive manner, *i.e.*, previously encoded levels serve as priors for predicting the current one:

$$Q(\mathbf{X}) = \prod_{l=1}^{L} Q\left(\mathbf{X}^l \mid \mathbf{X}^{1:l-1}\right), \tag{2}$$

where each conditional distribution is predicted by an occupancy predictor:

$$Q\left(\mathbf{X}^l \mid \mathbf{X}^{1:l-1}\right) = \text{Predictor}\left(\mathbf{X}^{1:l-1}\right). \tag{3}$$

Suppose predictions are being made at level l. Given the encoded occupancy codes $\{\mathbf{X}^1, \cdots, \mathbf{X}^{l-1}\}$, to obtain denser context features, GRED first downsamples $\mathbf{X}^{l-1}$ into a pre-defined dense octree level k:

$$\mathbf{G}^k = \text{Downsampling}\left(\mathbf{X}^{l-1}\right), \tag{4}$$

where the Downsampling$(\cdot)$ operation embeds the occupancy codes of $l-1$ into feature maps at level k using sparse convolutions. In $\mathbf{G}^k$, each channel corresponds to a specific occupancy state of a node at level $l-1$, thereby enriching contextual information with higher density.

To extract contextual features, $\mathbf{G}^k$ is fed to a ResBlock for *Feature Extraction*, thereby obtaining the feature $\mathbf{F}^k$:

$$\mathbf{F}^k = \text{ResBlock}\left(\mathbf{G}^k\right). \tag{5}$$

Although it is possible to directly predict occupancy in this dense space, it would incur prohibitive computational costs due to the vast number of potential sub-nodes. Instead, GRED progressively reverts $\mathbf{F}^k$ to the original sparse space $\mathbf{F}^l$ through multi-step upsampling, thereby achieving the *Re-Sparsification* of the features:

$$\mathbf{F}^{k+1} = \text{Pruning}\left(\text{Upsampling}\left(\mathbf{F}^k\right), \mathbf{X}^k\right), \tag{6}$$

where Upsampling$(\cdot)$ is a linear transformation followed by a PReLU activation, performing an $8\times$ channel expansion, and Pruning$(\cdot)$ discards features of unoccupied child nodes. This process is recursively applied until the feature map $\mathbf{F}^l$ is obtained. The feature is then fed into an MLP-based predictor to estimate the occupancy distribution:

$$\hat{\mathbf{p}}^l = \text{Predictor}\left(\mathbf{F}^l\right). \tag{7}$$

Then the true occupancy codes $\mathbf{X}^l$ are entropy-encoded using $\hat{\mathbf{p}}^l$, finishing *Prediction & Coding*. Overall, GRED enriches contextual information with low computational overhead while preserving the progressive decoding workflow.

Although many 3D tasks employ densification operations [6,10], such as quantization and downsampling, before processing and analysis, LiDAR point cloud compression presents a unique constraint: the decoder cannot access the full geometry at the beginning of decoding. Therefore, globally pre-densifying all octree levels is infeasible. This insight, combined with the observed nonlinear drop in occupancy density across octree levels, supports the necessity of on-the-fly re-densification within a progressive octree coding pipeline.

3.2 Cross-Scale Feature Propagation Module

While the proposed GRED module effectively mitigates HRCS, we delve into the rich inter-scale contextual dependencies across the octree. Existing octree-based

codecs typically extract features and predict occupancy codes independently at each octree level or within local node windows. However, this per-level processing overlooks the strong contextual dependencies across octree levels, leading to redundant feature extraction.

To fully leverage inter-scale context, we propose a unified **Cross-Scale Feature Propagation (XFP) Module** that (i) directly propagates features across octree levels and (ii) generalizes the core idea of the GRED Module into a broader, multi-scale framework. In fact, the GRED Module can be viewed as a special case of XFP, applied only at the deepest levels. XFP shares features from coarser (shallower) levels with finer (deeper) levels, avoiding redundant feature extraction and enhancing contextual awareness.

Suppose we are predicting the occupancy codes at level l, meaning that the preceding feature maps $\{\mathbf{F}^1, \cdots, \mathbf{F}^{l-1}\}$ and occupancy codes $\{\mathbf{X}^1, \cdots, \mathbf{X}^{l-1}\}$ are available. The first step in XFP is to determine an appropriate feature propagation strategy. In this work, we define two propagation regimes based on a pre-defined threshold level t:

1. **Shallow levels** ($l \leq t$): feature propagation is conducted without re-densification, as the geometry remains relatively dense.
2. **Deep levels** ($l > t$): feature propagation incorporates contextual information from the dense level k through occupancy-based re-densification.

Shallow-Level Propagation. For levels $l \leq t$, the octree is relatively shallow and the contextual information is sufficiently dense, making the HRCS problem less prominent. At these levels, we adopt a simplified version of the GRED module, omitting the *Re-Densification* step.

In the *Feature Extraction* step, since the re-densified feature $\mathbf{G}$ is omitted, the input is directly the encoded feature from level $l-1$, denoted as $\mathbf{F}^{l-1}$. A ResBlock is then applied to extract features and obtain the representation $\mathbf{S}^{l-1}$:

$$\mathbf{S}^{l-1} = \text{ResBlock}(\mathbf{F}^{l-1}). \tag{8}$$

Next, through one step of *Re-Sparsification*, $\mathbf{S}^{l-1}$ is upsampled to level l to produce the re-sparsified feature $\mathbf{F}^l$:

$$\mathbf{F}^l = \text{Pruning}\big(\text{Upsampling}\big(\text{Concat}\big(\mathbf{S}^{l-1}, \mathbf{X}^{l-1}\big)\big), \mathbf{X}^{l-1}\big), \tag{9}$$

where Concat denotes channel-wise concatenation of matrices. The obtained feature $\mathbf{F}^l$ then undergoes the same *Prediction & Coding* as GRED.

Deep-Level Propagation with Re-Densification. For $l > t$, the spatial sparsity makes direct propagation less effective. Therefore, we apply the full GRED module at these levels. During this process, we incorporate additional inter-scale contextual information to enrich the extracted features.

GRED utilizes the re-densified feature $\mathbf{G}^k$ for *Feature Extraction*. To fully leverage the information from the previous scale, we concatenate $\mathbf{G}^k$ with the original feature map $\mathbf{F}^k$ and use ResBlock to obtain the fused representation:

$$\mathbf{H}^k = \text{ResBlock}\left(\text{Concat}(\mathbf{F}^k,\ \mathbf{G}^k)\right). \tag{10}$$

The fused representation $\mathbf{H}^k$ replaces the original input feature $\mathbf{F}^k$ in the *Re-Sparsification*, enabling multi-scale features to be fused into the current level:

$$\mathbf{F}^{k+1} = \text{Pruning}\big(\text{Upsampling}\big(\text{Concat}(\mathbf{H}^k, \mathbf{X}^k)\big), \mathbf{X}^k\big), \quad k = t, \ldots, l-1. \tag{11}$$

By recursively applying the above process, we obtain the fused feature $\mathbf{F}^l$. Finally, *Prediction & Coding* is performed at level l based on the feature $\mathbf{F}^l$.

This cross-scale propagation scheme reuses context-rich features from earlier levels and adapts them to finer resolutions through sparse, occupancy-aware operations. By combining shallow and deep propagation pathways, XFP unifies dense and sparse processing, enabling efficient feature extraction throughout the octree hierarchy.

3.3 Integer-Only Inference for Cross-Platform Consistency

Neural compression models require strict numerical consistency between the encoder and the decoder, since entropy coding is highly sensitive to numerical errors. Even minor discrepancies in predicted probabilities may lead to decoding failures [1, 8, 20, 46]. However, floating-point computations are inherently non-deterministic across hardware and software. In practical deployment, enforcing identical execution environments is unrealistic. To enable reliable cross-platform coding, we therefore develop an integer-only inference pipeline that guarantees deterministic computation. To the best of our knowledge, this is the first cross-platform integer-only inference pipeline for neural PCC.

Our objective is to eliminate all floating-point operations during inference while preserving both runtime efficiency and compression performance. To this end, we adopt a mixed-precision integer design that balances computational cost and numerical stability: (i) computation-intensive operators are executed using low-bit integer arithmetic for efficiency; (ii) lightweight yet numerically sensitive operators are implemented with higher-precision fixed-point arithmetic to control quantization error. This design enables fully integerized end-to-end deterministic execution without sacrificing much performance.

Quantization. We employ 8-bit integer quantization for compute-dominant operators, including linear layers and sparse convolutions, while using 32-bit fixed-point arithmetic for re-quantization and activation functions. Given a floating-point tensor $\mathbf{x}$, its quantized integer representation is defined as

$$\mathbf{q}_x = \mathcal{Q}(\mathbf{x}; s, z) = \text{clip}\Big(\Big\lfloor \frac{\mathbf{x}}{s} \Big\rceil + z,\ q_{\min}, q_{\max}\Big), \tag{12}$$

where s and z denote the scale and zero-point, respectively, and $\lfloor \cdot \rceil$ represents a rounding operation. Quantization parameters for activations in linear and sparse convolution operators are obtained via lightweight calibration on a small subset of training data. All remaining operators are implemented in fixed-point format and therefore do not require distribution statistics. During inference, all computations are performed directly in integer space, ensuring deterministic behavior.

Integer linear and sparse convolution. For linear layers, activations and weights are represented as 8-bit integers, while accumulation operations are performed using 32-bit integers:

$$\mathbf{y}_{\text{int32}} = \sum_{c} (\mathbf{q}_x - z_x)\,(\mathbf{q}_w - z_w)^{\mathrm{T}} + \mathbf{b}_{\text{int32}}, \tag{13}$$

where $\mathbf{q}_x$ and $\mathbf{q}_w$ denote quantized activations and weights, z_x and z_w are the corresponding zero-points, and $\mathbf{b}_{\text{int32}}$ denotes the quantized bias. In our implementation, weights are symmetrically quantized (*i.e.*, $z_w = 0$). The accumulated result is then re-quantized to 8-bit representation via fixed-point scaling:

$$\mathbf{q}_y = \text{clip}(\lfloor \mathbf{y}_{\text{int32}} \cdot m/2^r \rceil + z_y,\ q_{\min}, q_{\max}), \tag{14}$$

where m is a precomputed integer multiplier, r is a fixed right-shift factor, and z_y denotes the output zero-point.

For sparse convolution, each operator is decomposed into a deterministic coordinate mapping and multiple indexed linear transforms [7]. This decomposition preserves sparsity while enabling efficient execution using integer linear primitives.

Integer softmax. The occupancy predictor outputs a 255-way probability distribution via a softmax operation that is directly consumed by entropy coding. To avoid platform-dependent floating-point exponentiation and division, we implement softmax entirely in fixed-point arithmetic. Given the logits $\boldsymbol{\ell}$, the softmax probability is computed in a numerically stable form as

$$p_i = \frac{\exp(\ell_i - \max_k \ell_k)}{\sum_j \exp(\ell_j - \max_k \ell_k)}, \quad i, j, k \in \{1, \ldots, 255\}. \tag{15}$$

The exponential function is approximated using a precomputed lookup table that covers only the non-positive domain (i.e., $\ell_i - \max_k \ell_k \leq 0$), which significantly reduces table size while preserving numerical stability. Specifically, we use a 16-bit fixed-point LUT over $[-12, 0]$ with a step size of $1/512$. Accumulation and normalization are performed using 32-bit integer arithmetic. The resulting probabilities are represented in high-precision fixed-point format, guaranteeing bit-exact consistency across different platforms.

4 Experiments

In this section, we present a comprehensive experimental evaluation of our method, including experimental settings, comparative results with state-of-the-art approaches, and ablation studies.

4.1 Settings

In this section, we detail the experimental setup, including the benchmark datasets, evaluation metrics, and comparative baselines. The implementation details are provided in the supplementary materials.

Benchmark Datasets. Experiments are conducted on two different LiDAR datasets: KITTI [15] and Ford [32]. The KITTI dataset consists of 22 stereo sequences collected by a Velodyne LiDAR scanner across diverse continuous scenes, totaling 43,552 frames. Following Fu *et al*. [11], we use sequences #00 to #10 for training and #11 to #21 for testing. The Ford dataset comprises three sequences, each containing 1,500 frames. Consistent with Song *et al*. [42], we use sequence #01 for training, and sequences #02 and #03 for testing.

Evaluation Metrics. We adopt point-to-point PSNR (D1 PSNR) and point-to-plane PSNR (D2 PSNR) [45] as distortion measures. These are standard metrics recommended by MPEG [40]. We employ the Bjøntegaard Delta (BD) metrics [3] for evaluating rate-distortion performance, namely Bjøntegaard Delta Peak Signal-to-Noise Ratio (BD-PSNR) and Bjøntegaard Delta Rate (BD-Rate). It is important to note that both BD-Rate and BD-PSNR measure the **relative gains** of a tested model compared to a baseline. A negative BD-Rate or a positive BD-PSNR indicates that the tested model outperforms the baseline.

Compared Methods. We compare the proposed framework with 5 representative PCC methods. G-PCC [40], standardized by MPEG, serves as the classical geometry-based benchmark. OctAttention [11] and Light EHEM [42] are transformer-based octree compression methods, while Unicorn [48] represents a recent voxel-based PCC framework. In addition, RENO [57] introduces an efficient sampling strategy to balance compression performance and computational efficiency. Unless otherwise specified, all methods are re-evaluated on a computer equipped with an AMD EPYC 7R32 CPU and an NVIDIA RTX 4090 GPU. Due to the unavailability of the official implementation of Light EHEM, we report the performance metrics as provided in the original paper.

4.2 Performance Analysis

This section evaluates the proposed method in terms of rate-distortion performance and computational efficiency, providing a comprehensive assessment of its effectiveness.

Rate-Distortion Performance. This section presents the RD performance of the proposed method compared to several existing methods, using two standard evaluation curves: D1 PSNR vs. Bits Per Input Point (BPP) and D2 PSNR vs. BPP. A curve closer to the upper-left corner indicates higher geometry precision at lower bitrates, reflecting better compression performance.

The experimental results are illustrated in Fig. 4. On the KITTI dataset, our method achieves performance comparable to the transformer-based Light EHEM and outperforms the sparse convolution-based Unicorn, demonstrating clear advantages in RD performance while providing substantially improved computational efficiency. Notably, the integer-only model maintains RD performance

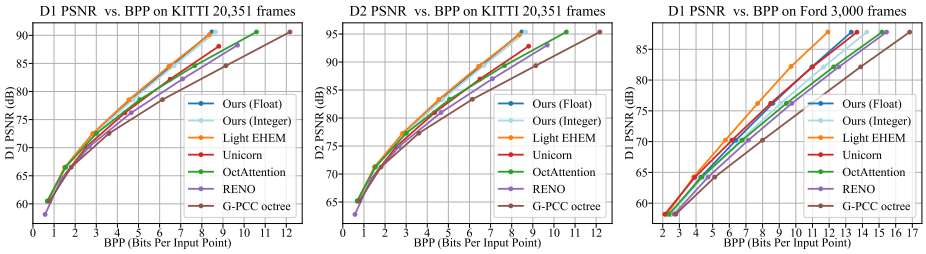


Fig. 4: RD performance comparison with existing methods on KITTI and Ford datasets from 11 bits to 16 bits.

Table 1: BD-Rate (%) and BD-PSNR (dB) gains of our floating-point model.

Ours (Float) vs. Existing Methods	KITTI BD-Rate D1	KITTI BD-Rate D2	KITTI BD-PSNR D1	KITTI BD-PSNR D2	Ford BD-Rate D1	Ford BD-Rate D2	Ford BD-PSNR D1	Ford BD-PSNR D2
OctAttention [11] (AAAI'22)	-7.294	-7.332	0.754	0.760	-6.205	-6.193	0.969	0.969
Light EHEM [42] (CVPR'23)	1.429	1.422	-0.152	-0.152	11.535	11.987	-1.899	-1.960
Unicorn [48] (TPAMI'25)	-10.862	-10.889	1.367	1.373	3.922	3.989	-0.622	-0.637
RENO [57] (CVPR'25)	-15.582	-15.579	1.887	1.890	-11.049	-11.040	1.938	1.938
G-PCC octree [40]	-21.931	-21.954	2.532	2.541	-19.150	-19.143	3.411	3.412

comparable to its floating-point counterpart. On the Ford dataset, the overall RD performance is relatively less competitive, yet our method still surpasses approaches with similar coding latency, such as RENO and G-PCC octree. This performance gap is likely due to the limited number of training samples (1,500 frames), which constrains generalization capability. Table 1 presents the quantitative BD-rate and BD-PSNR metrics of the proposed method compared with existing approaches, and Table 2 reports the corresponding results of the integer-only model against real-time baselines. On the KITTI dataset, compared with the efficiency-oriented method RENO, our approach achieves gains of 1.887 dB and 1.890 dB in D1 and D2 PSNR, respectively, while the integer-only model obtains gains of 1.728 dB and 1.730 dB under the same setting. These results demonstrate that the proposed framework effectively exploits structural redundancy within the octree representation, achieving competitive compression per-

Table 2: BD-Rate (%) and BD-PSNR (dB) gains of our integer-only model.

Ours (Integer) vs. Existing Real-time Methods	KITTI BD-Rate D1	KITTI BD-Rate D2	KITTI BD-PSNR D1	KITTI BD-PSNR D2	Ford BD-Rate D1	Ford BD-Rate D2	Ford BD-PSNR D1	Ford BD-PSNR D2
RENO [57] (CVPR'25)	-14.289	-14.285	1.728	1.730	-6.875	-6.864	1.185	1.185
G-PCC octree [40]	-20.705	-20.728	2.378	2.387	-15.355	-15.348	2.694	2.694

Table 3: Comparison of coding time across 11–16 bits with existing real-time methods.

Methods	Device	KITTI		Ford	
		Enc Time	Dec Time	Enc Time	Dec Time
G-PCC octree	AMD EPYC 7R32	0.149s	0.103s	0.150s	0.107s
RENO	NVIDIA RTX 4090	0.059s	0.056s	0.072s	0.057s
Ours (Float)	NVIDIA RTX 4090	0.075s	0.081s	0.093s	0.103s
Ours (Integer)	NVIDIA RTX 4090	0.060s	0.070s	0.067s	0.082s
Ours (Integer)	NVIDIA RTX 5880	0.047s	0.056s	0.054s	0.067s
Ours (Integer)	NVIDIA Tesla V100	0.166s	0.175s	0.187s	0.199s

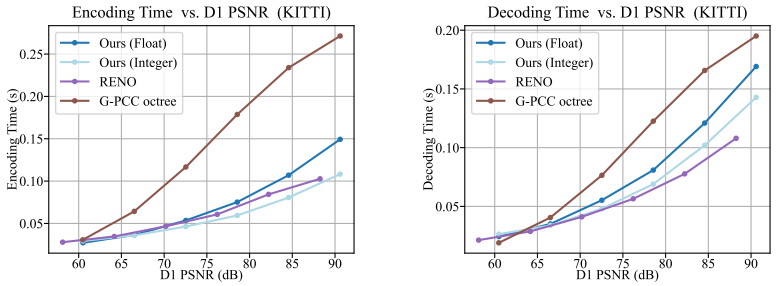


Fig. 5: Comparison of coding time across 11–16 bits with existing real-time methods.

formance together with high runtime efficiency and strict cross-platform consistency.

Computational Efficiency. To evaluate the real-time capability of the proposed framework, we measure the encoding and decoding time of our method and several representative methods, as summarized in Table 3. The main cross-method comparison uses the RTX 4090 GPU. G-PCC has no GPU implementation, so we report its CPU runtime following common PCC practice. FLOPs statistics are additionally provided in the supplementary materials. The proposed method demonstrates faster runtime than most competing methods. For a clearer comparison, Fig. 2(a) illustrates the BD-PSNR gains versus frames per second (FPS) on the 12-bit quantized KITTI dataset. Our method achieves 14 FPS for the complete encoding–decoding process, surpassing other methods with similar compression performance. To further analyze runtime behavior under varying geometry precisions, we compare the averaged encoding and decoding time across 11–16-bit quantization settings against existing real-time methods. As shown in Fig. 5, the integer-only model consistently outperforms G-PCC, while maintaining runtime comparable to the efficiency-oriented RENO. These results confirm that the proposed framework achieves a favorable balance between compression performance and computational efficiency.

In addition, Table 3 reports the averaged runtime of our integer-only model across multiple GPUs beyond the default RTX 4090, including RTX 5880 and Tesla V100. The integer-only pipeline achieves robust throughput on different

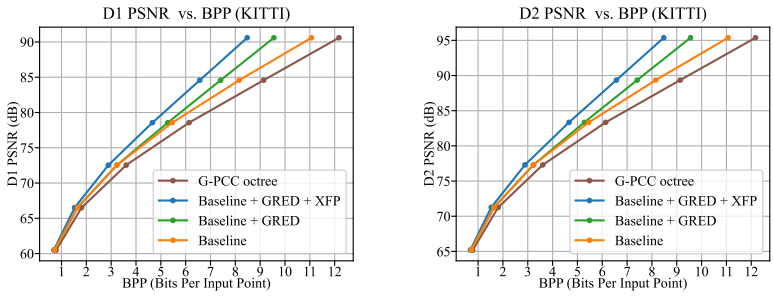


Fig. 6: RD performance comparison of ablated models from 11 bits to 16 bits.

Table 4: BD-Rate (%) and BD-PSNR (dB) gains of the proposed modules.

Methods	Compared with Baseline				Compared with G-PCC octree			
	BD-Rate		BD-PSNR		BD-Rate		BD-PSNR	
	D1	D2	D1	D2	D1	D2	D1	D2
Baseline	0.00	0.00	0.00	0.00	-10.04	-10.08	1.13	1.14
+ GRED	-3.78	-3.78	0.40	0.40	-13.44	-13.47	1.49	1.50
+ GRED + XFP	-13.22	-13.21	1.46	1.46	-21.93	-21.95	2.53	2.54

devices, while its runtime varies with hardware capability as expected (*e.g.*, faster on RTX 5880 and slower on V100). These results further validate the practicality of our cross-platform design for deployment on heterogeneous platforms.

4.3 Ablation Studies

To evaluate the individual contributions of the proposed components, we conduct ablation studies on the GRED module and the XFP module. Each component is systematically removed to assess its impact on overall RD performance. See the supplementary materials for more ablation studies.

Ablation of XFP. We ablate the XFP module by removing cross-scale feature propagation. As shown by the "Baseline + GRED" curves in Fig. 6, this leads to a clear RD drop at high quantization precisions, but has negligible impact at low precisions, consistent with our theoretical analysis. Quantitative results in Table 4 show that this removal results in a degradation of approximately 1.06 dB, indicating that the integration of cross-scale information through XFP is critical for improving RD performance.

Ablation of GRED. To evaluate the effectiveness of the GRED module, we removed GRED on top of the XFP ablation. In this setting, the dense features extracted from the shallow level are no longer utilized for predicting the occupancy of deeper levels. As a result, the model is directly exposed to the HRCS problem under high-resolution encoding. The corresponding RD performance is shown as the "Baseline" curves in Fig. 6. The results reveal that, as the resolution increases, the RD performance of the model without GRED de-

clines significantly compared to the variant equipped with GRED. Quantitative results in Table 4 show that removing the GRED module leads to a further performance drop of approximately 0.40 dB. This performance gap highlights the positive impact of the GRED module in mitigating the effects of HRCS and enhancing performance.

5 Conclusion

This paper proposes a cross-platform, real-time neural compression framework for LiDAR point clouds, jointly addressing deterministic coding requirements and the High-Resolution Contextual Sparsity challenge. We introduce an octree-based design that integrates the Geometry Re-Densification module and the Cross-Scale Feature Propagation module, enabling effective intra-scale and cross-scale context modeling under extreme sparsity. In addition, an integer-only inference pipeline is developed to ensure bit-exact coding across heterogeneous platforms. Extensive experiments demonstrate that the proposed framework achieves competitive rate-distortion performance while maintaining real-time encoding and decoding speed.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 62372491), the Guangdong S&T Programme (No. 2025B0101130003), the Guangdong Basic and Applied Basic Research Foundation (2023B1515120087), and the Science and Technology Planning Project of Key Laboratory of Advanced IntelliSense Technology, Guangdong Science and Technology Department (2023B1212060024), and the Major Key Project of Pengcheng Laboratory (No. PCL2024A04).

References

1. Ballé, J., Johnston, N., Minnen, D.: Integer networks for data compression with latent-variable models. In: International Conference on Learning Representations (ICLR) (2019)
2. Biswas, S., Liu, J., Wong, K., Wang, S., Urtasun, R.: MuSCLE: Multi sweep compression of LiDAR using deep entropy models. In: Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS) (2020)
3. Bjøntegaard, G.: Calculation of average PSNR differences between RD-curves. Input document VCEG-M33, Video Coding Experts Group, 13th VCEG Meeting, Austin, Texas, USA (2001)
4. Cao, Y., Wang, Y., Chen, H.: Real-time LiDAR point cloud compression and transmission for resource-constrained robots. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 12789–12795 (2025)
5. Chen, Z., Qian, Z., Wang, S., Chen, Q.: Point cloud compression with sibling context and surface priors. In: European Conference on Computer Vision (ECCV). pp. 744–759 (2022)

6. Choe, J., Joung, B., Rameau, F., Park, J., Kweon, I.S.: Deep point cloud reconstruction. In: International Conference on Learning Representations (ICLR) (2022)
7. Choy, C., Gwak, J., Savarese, S.: 4D spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3075–3084 (2019)
8. Conceição, R., Porto, M., Peng, W.H., Agostini, L.: Cross-platform neural video coding: A case study. In: IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5 (2025)
9. Cui, M., Long, J., Feng, M., Li, B., Kai, H.: OctFormer: Efficient octree-based transformer for point cloud compression with local enhancement. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 470–478 (2023)
10. Deng, H., Jing, K., Cheng, S., Liu, C., Ru, J., Bo, J., Wang, L.: LinNet: Linear network for efficient point cloud representation learning. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 37, pp. 43189–43209 (2024)
11. Fu, C., Li, G., Song, R., Gao, W., Liu, S.: OctAttention: Octree-based large-scale contexts model for point cloud compression. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 625–633 (2022)
12. Fu, C., Qin, T., Wang, S., Li, Z.: DeepRAHT: Learning predictive RAHT for point cloud attribute compression. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). vol. 40, pp. 14738–14746 (2026)
13. Garcia, D.C., Fonseca, T.A., Ferreira, R.U., de Queiroz, R.L.: Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts. IEEE Transactions on Image Processing **29**, 313–322 (2020)
14. Garcia, D.C., de Queiroz, R.L.: Context-based octree coding for point-cloud video. In: IEEE International Conference on Image Processing (ICIP). pp. 1412–1416 (2017)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3354–3361 (2012)
16. Golla, T., Klein, R.: Real-time point cloud compression. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5087–5092 (2015)
17. He, Y., Ren, X., Tang, D., Zhang, Y., Xue, X., Fu, Y.: Density-preserving deep point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2333–2342 (2022)
18. Huang, L., Wang, S., Wong, K., Liu, J., Urtasun, R.: OctSqueeze: Octree-structured entropy model for LiDAR compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1313–1323 (2020)
19. Huang, R., Yu, P., Liao, S., Liang, F.: Efficient point cloud attribute compression using rich parallelizable context model. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8436–8440 (2024)
20. Jia, Z., Li, B., Li, J., Xie, W., Qi, L., Li, H., Lu, Y.: Towards practical real-time neural video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12543–12552 (2025)
21. Jiang, R., Zhang, Y., Wang, L., Yu, P., Guo, Y.: AIQViT: Architecture-informed post-training quantization for vision transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 17635–17643 (2025)
22. Kammerl, J., Blodow, N., Rusu, R.B., Gedikli, S., Beetz, M., Steinbach, E.: Real-time compression of point cloud streams. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 778–785 (2012)

23. Koyuncu, E., Solovyev, T., Alshina, E., Kaup, A.: Device interoperability for learned image compression with weights and activations quantization. In: Picture Coding Symposium (PCS). pp. 151–155 (2022)
24. Li, H., Xu, L., Xie, L., Gao, W., Ren, Z., Li, G., Guo, Y.: Slide deformable transformer for high-precision LiDAR point cloud compression. IEEE Transactions on Image Processing pp. 1–1 (2026)
25. Liang, G., Liu, X., Wu, J.: GPLQ: A general, practical, and lightning QAT method for vision transformers. In: Advances in Neural Information Processing Systems. vol. 38, pp. 7033–7057 (2025)
26. Liu, X., Liang, F.: Fast dynamic point cloud geometry compression via resolution-adaptive context model. In: International Conference on Visual Communications and Image Processing (VCIP). pp. 1–5 (2025)
27. Luo, A., Song, L., Nonaka, K., Unno, K., Sun, H., Goto, M., Katto, J.: SCP: Spherical-coordinate-based learned point cloud compression. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). vol. 38, pp. 3954–3962 (2024)
28. Mekuria, R., Blom, K., Cesar, P.: Design, implementation, and evaluation of a point cloud codec for tele-immersive video. IEEE Transactions on Circuits and Systems for Video Technology **27**(4), 828–842 (2017)
29. Meng, L., Ou, R., Zhang, Q., Liu, S., Li, G.: PCGCD: Joint point cloud geometry compression and denoising. In: Data Compression Conference (DCC). pp. 23–32 (2025)
30. Nguyen, D.T., Kaup, A.: Learning-based lossless point cloud geometry coding using sparse tensors. In: IEEE International Conference on Image Processing (ICIP). pp. 2341–2345 (2022)
31. Nguyen, D.T., Quach, M., Valenzise, G., Duhamel, P.: Lossless coding of point cloud geometry using a deep generative model. IEEE Transactions on Circuits and Systems for Video Technology **31**(12), 4617–4629 (2021)
32. Pandey, G., McBride, J.R., Eustice, R.M.: Ford campus vision and LiDAR data set. The International Journal of Robotics Research **30**(13), 1543–1552 (2011)
33. Pang, J., Bui, K., Tian, D.: PIVOT-Net: Heterogeneous point-voxel-tree-based framework for point cloud compression. In: International Conference on 3D Vision (3DV). pp. 1270–1279 (2024)
34. Pang, J., Lodhi, M.A., Tian, D.: GRASP-Net: Geometric residual analysis and synthesis for point cloud compression. In: Proceedings of the International Workshop on Advances in Point Cloud Compression, Processing and Analysis (APCCPA). pp. 11–19 (2022)
35. Qi, C.R., Su, H., Kaichun, M., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 77–85 (2017)
36. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS). pp. 5105–5114 (2017)
37. Qin, T., Li, G., Gao, W., Liu, S.: Multi-grained point cloud geometry compression via dual-model prediction with extended octree. ACM Transactions on Multimedia Computing, Communications, and Applications **20**(9) (2024)
38. Quach, M., Valenzise, G., Dufaux, F.: Learning convolutional transforms for lossy point cloud geometry compression. In: Proceedings of the IEEE International Conference on Image Processing (ICIP). pp. 4320–4324 (2019)
39. Que, Z., Lu, G., Xu, D.: VoxelContext-Net: An octree based framework for point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6042–6051 (2021)

40. Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P.A., Cohen, R.A., Krivokuća, M., Lasserre, S., Li, Z., Llach, J., Mammou, K., Mekuria, R., Nakagami, O., Siahaan, E., Tabatabai, A., Tourapis, A.M., Zakharchenko, V.: Emerging MPEG standards for point cloud compression. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **9**(1), 133–148 (2019)
41. Song, F., Shao, Y., Gao, W., Wang, H., Li, T.: Layer-wise geometry aggregation framework for lossless LiDAR point cloud compression. IEEE Transactions on Circuits and Systems for Video Technology **31**(12), 4603–4616 (2021)
42. Song, R., Fu, C., Liu, S., Li, G.: Efficient hierarchical entropy model for learned point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14368–14377 (2023)
43. Stathoulopoulos, N., Saucedo, M.A., Koval, A., Nikolakopoulos, G.: RecNet: An invertible point cloud encoding through range image embeddings for multi-robot map sharing and reconstruction. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 4883–4889 (2024)
44. Tang, H., Yang, S., Liu, Z., Hong, K., Yu, Z., Li, X., Dai, G., Wang, Y., Han, S.: TorchSparse++: Efficient training and inference framework for sparse convolution on GPUs. In: Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO). pp. 225–239 (2023)
45. Tian, D., Ochimizu, H., Feng, C., Cohen, R., Vetro, A.: Geometric distortion metrics for point cloud compression. In: IEEE International Conference on Image Processing (ICIP). pp. 3460–3464 (2017)
46. Tian, K., Guan, Y., Xiang, J., Zhang, J., Han, X., Yang, W.: Towards real-time neural video codec for cross-platform application using calibration information. In: ACM International Conference on Multimedia (ACM MM). pp. 7961–7970 (2023)
47. Tzamarias, D.E.O., Chow, K., Blanes, I., Serra-Sagristà, J.: Fast run-length compression of point cloud geometry. IEEE Transactions on Image Processing **31**, 4490–4501 (2022)
48. Wang, J., Xue, R., Li, J., Ding, D., Lin, Y., Ma, Z.: 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)
49. Wang, J., Zhu, H., Liu, H., Ma, Z.: Lossy point cloud geometry compression via end-to-end learning. IEEE Transactions on Circuits and Systems for Video Technology **31**(12), 4909–4923 (2021)
50. Wang, S., Jiao, J., Cai, P., Wang, L.: R-PCC: A baseline for range image-based point cloud compression. In: International Conference on Robotics and Automation (ICRA). pp. 10055–10061 (2022)
51. Wang, S., Liu, M.: Point cloud compression with range image-based entropy model for autonomous driving. In: European Conference on Computer Vision (ECCV). pp. 323–340 (2022)
52. Wang, X., Zhang, Y., Liu, T., Liu, X., Xu, K., Wan, J., Guo, Y., Wang, H.: TopNet: Transformer-efficient occupancy prediction network for octree-structured point cloud geometry compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 27305–27314 (June 2025)
53. Wei, X., Gong, R., Li, Y., Liu, X., Yu, F.: QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In: International Conference on Learning Representations (2022)
54. Wen, X., Wang, X., Hou, J., Ma, L., Zhou, Y., Jiang, J.: Lossy geometry compression of 3D point cloud data via an adaptive octree-guided network. In: IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6 (2020)

55. Wiesmann, L., Milioto, A., Chen, X., Stachniss, C., Behley, J.: Deep compression for dense point cloud maps. IEEE Robotics and Automation Letters **6**(2), 2060–2067 (2021)
56. Xie, L., Gao, W., Zheng, H., Li, G.: SPCGC: Scalable point cloud geometry compression for machine vision. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 17272–17278 (2024)
57. You, K., Chen, T., Ding, D., Asif, M.S., Ma, Z.: RENO: Real-time neural compression for 3D LiDAR point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–10 (2025)
58. You, Y., Wang, Y., Chao, W.L., Garg, D., Pleiss, G., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-LiDAR++: Accurate depth for 3D object detection in autonomous driving. In: International Conference on Learning Representations (ICLR) (2020)
59. Yu, P., Zhang, Y., Liang, F., Li, H., Guo, Y.: Hierarchical distortion learning for fast lossy compression of point clouds. IEEE Transactions on Multimedia **27**, 6031–6046 (2025)
60. Zhang, C., Gao, W.: AdaDPCC: Adaptive rate control and rate-distortion-complexity optimization for dynamic point cloud compression. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). pp. 13188–13196 (2025)
61. Zhang, J., Wang, J., Ding, D., Ma, Z.: Scalable point cloud attribute compression. IEEE Transactions on Multimedia **27**, 889–899 (2025)
62. Zhang, Q., Shao, Y., Meng, L., Jiao, H., Liu, S., Li, G.: Rate-distortion optimized motion estimation for dynamic point cloud geometry compression. In: Data Compression Conference (DCC). pp. 417–417 (2025)
63. Zhou, L., Ruan, C., Ling, N., Chen, Z., Wang, W., Jiang, W.: TVC: Tokenized video compression with ultra-low bit rate. Visual Intelligence **3**(1), 25 (2025)
64. Zhou, X., Qi, C.R., Zhou, Y., Anguelov, D.: RIDDLE: Lidar data compression with range image deep delta encoding. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17191–17200 (2022)
65. Zuo, D., Yu, P., Huang, R., Huang, Y., Sun, W., Liang, F.: Diverse context model for large-scale dynamic point cloud compression. In: IEEE International Conference on Visual Communications and Image Processing (VCIP). pp. 1–5 (2023)