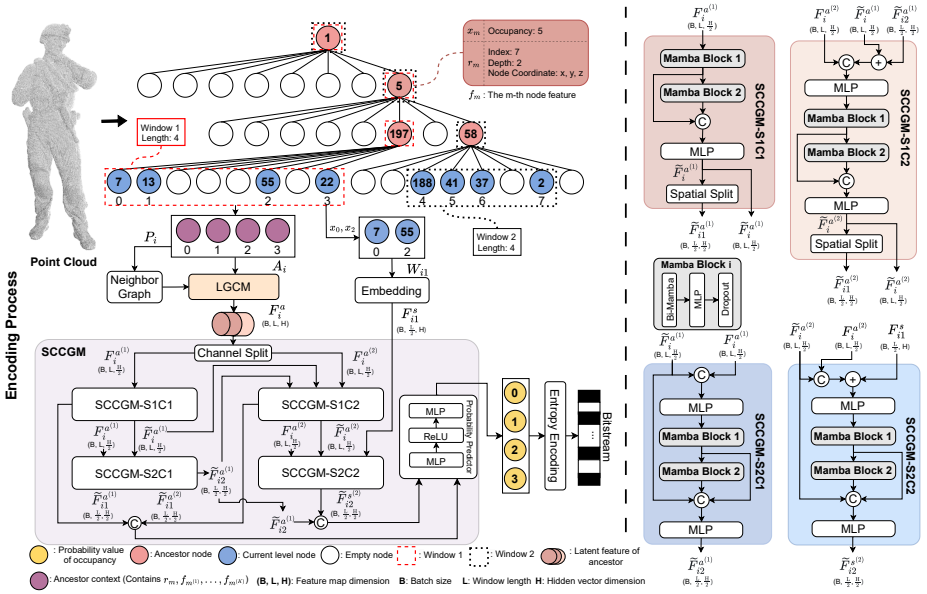
Graphical Abstract

OctMamba: Mamba-Based Octree Context Entropy Model for Point Cloud Geometry Compression

Zhaoyi Jiang, Yi Xu, Frederick W.B. Li, Gary K.L Tam, Chao Song, Bailin Yang



Highlights

OctMamba: Mamba-Based Octree Context Entropy Model for Point Cloud Geometry Compression

Zhaoyi Jiang, Yi Xu, Frederick W.B. Li, Gary K.L Tam, Chao Song, Bailin Yang

- Jointly models spatial, channel, and topological redundancies, moving beyond conventional spatial-only designs.
- Embedding Mamba layers locally within specialized subcomponents instead of as a global backbone, enabling structured context modeling.
- Achieves efficient long-range modeling with linear complexity, yielding a smaller model and faster decoding while outperforming baselines.

OctMamba: Mamba-Based Octree Context Entropy Model for Point Cloud Geometry Compression

Zhaoyi Jiang[a], Yi Xu[a], Frederick W.B. Li[b], Gary K.L Tam[c], Chao Song[a], Bailin Yang[a,*]

[a]*School of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou, 310018, Zhejiang, China*
[b]*Department of Computer Science, Durham University, United Kingdom*
[c]*Department of Computer Science, Swansea University, United Kingdom*

Abstract

Existing learned point cloud compression frameworks face two major limitations: (1) they focus almost exclusively on spatial redundancy and (2) rely on architectures built around local-global transformers or global Mamba blocks. Transformers incur quadratic complexity, while global Mamba lacks the granularity to capture structured correlations across multiple dimensions. We propose OctMamba, the first unified framework to jointly exploit spatial, channel, and topological redundancies, dimensions previously overlooked in point cloud geometry compression. Our approach introduces a new architectural principle: embedding Mamba modules within specialized subcomponents rather than applying them globally, challenging existing design paradigms. OctMamba combines two modules: Spatial-Channel Coupled Grouping Mamba (SCCGM) for spatial-channel fusion and Local Graph CNN-Mamba (LGCM) for topological encoding. This design enables efficient long-range modeling with linear complexity, delivering a smaller model and faster decoding while outperforming transformer-based and global Mamba baselines. On SemanticKITTI, OctMamba reduces bitrate by 60.2% over GPCC (D1 PSNR) and achieves state-of-the-art performance across LiDAR and dynamic human point cloud benchmarks with practical speed and scalability. By introducing multi-dimensional redundancy modeling, OctMamba has the potential to influ-

*Corresponding author
Email addresses: zyjiang@zjgsu.edu.cn (Zhaoyi Jiang), 23020100053@zjgsu.edu.cn (Yi Xu), frederick.li@durham.ac.uk (Frederick W.B. Li), k.l.tam@swansea.ac.uk (Gary K.L Tam), csong@zjgsu.edu.cn (Chao Song), ybl@zjgsu.edu.cn (Bailin Yang)

ence future research on efficient point cloud compression.

Keywords: Point cloud compression, State space models, Multi-dimensional redundancy, Occupancy-probability modeling

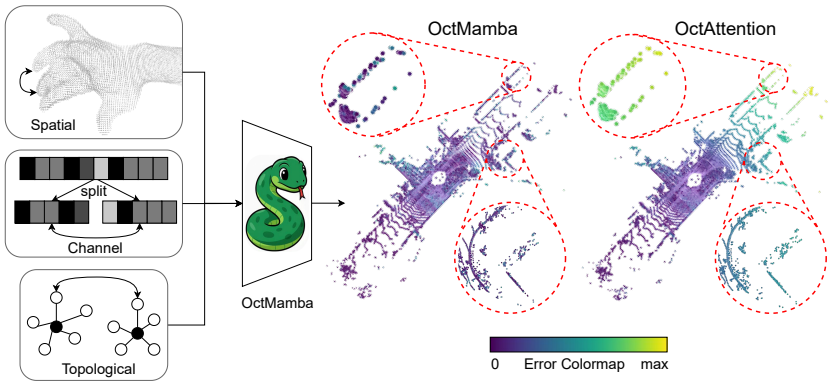


Figure 1: Overview of OctMamba. By coupling spatial, channel, and topological information, OctMamba aggregates multi-dimensional redundancies, reduces edge/sparsity ambiguity, and improves occupancy estimation, yielding smoother reconstructions and a more uniform error distribution. Visual comparisons with OctAttention further demonstrate more stable reconstructions in edge/sparse regions and flatter error maps.

1. Introduction

Point clouds play a fundamental role in 3D scene representation across applications such as autonomous driving, virtual reality (VR), and augmented reality (AR) [1, 2]. While they capture rich spatial detail, their large data volumes pose significant challenges for storage, transmission, and real-time processing. Efficient geometry compression is therefore essential for scalable and high-quality 3D scene understanding. Recent deep learning methods have advanced point cloud compression using data-driven entropy models and hierarchical neural architectures [3, 4]. Octree-based models, such as OctAttention [5], exploit autoregressive context for probability estimation, while block- or grouped-processing methods like EHEM [6] improve efficiency through parallelized node processing. Beyond octrees, implicit neural representation (INR) frameworks [7] model occupancy and attributes via coordinate-based networks, and graph- or attention-based models, such as GAEM [8], capture local structural dependencies.

Conventional 3D CNNs [9, 10] model local context, and transformers [5, 6] handle global dependencies.

Despite these advances, current approaches exhibit fundamental limitations that hinder both efficiency and fidelity [11]. First, most frameworks primarily focus on spatial redundancy, with limited use of channel and topological correlations. Point clouds exhibit rich dependencies beyond spatial structure (Fig. 1): multi-scale neighbor relationships arise from varying densities and geometric patterns [12, 13], latent feature channels often display strong correlations [14, 15], and topological relationships govern connectivity and hierarchy. Explicitly modeling these dimensions can reduce redundancy and improve prediction quality. Second, existing architectures typically rely on local–global transformers or globally applied Mamba blocks to capture long-range dependencies. While transformers can model global context, they incur quadratic complexity with sequence length, making them costly for large point clouds. Global Mamba improves efficiency but does not explicitly model structured spatial, channel and topological correlations within the octree. Introducing multi-dimensional dependencies further complicates the design space: modeling spatial, channel, and topological features jointly requires architectures that balance fidelity, complexity, and parallelism (Fig. 2).

To address these limitations, we propose OctMamba, the first unified octree-based framework that jointly exploits spatial, channel, and topological redundancies for point cloud geometry compression. Our approach introduces a new architectural principle: embedding Mamba modules within specialized subcomponents rather than applying them globally. OctMamba comprises two modules: Spatial–Channel Coupled Grouping Mamba (SCCGM), which performs staged spatial–channel fusion inside each context window, and Local Graph CNN–Mamba (LGCM), which encodes topological dependencies through local neighbor graphs fused with ancestor context. This localized embedding principle preserves granularity, removes the autoregressive bottleneck, achieves linear complexity, and delivers a significantly smaller model compared to transformer-based and global Mamba designs.

Our design is rigorously validated through standardized protocols and comprehensive analysis. We adopt standard PSNR-based rate–distortion evaluation and dataset

splits for fair comparison and report complexity metrics—FLOPs, memory, and parameter counts—alongside encoding and decoding times across octree depths. Experimental results demonstrate OctMamba's efficiency and scalability: on SemanticKITTI [16], it reduces BD-rate by 60.2% over GPCC and achieves consistent gains on Ford [17], MVUB [18], and MPEG 8i [19]. Ablation studies confirm the necessity of channel grouping and LGCM, the benefits of larger context windows and neighborhood sizes, and the superiority of localized Mamba embedding over transformer-based alternatives. Computational analysis further shows that OctMamba uses fewer parameters and lower FLOPs per window, while decoding remains faster than transformer-based and global Mamba designs at deeper octrees.

These findings highlight two core contributions addressing the observed limitations. First, OctMamba explicitly models spatial, channel, and topological redundancies, moving beyond a spatial-only focus to capture richer correlations for improved prediction and structural fidelity. Second, by embedding Mamba locally within submodules (SCCGM and LGCM), it eliminates the autoregressive bottleneck, achieves linear complexity, and delivers a smaller model without sacrificing accuracy. Together, these innovations combine high compression performance with practical scalability, providing a solid foundation for future research on efficient and high-fidelity point cloud geometry compression.

Our main contributions are summarized as follows:

- We introduce OctMamba, the first efficient octree-based framework that jointly exploits spatial, channel, and topological redundancies for point cloud geometry compression, moving beyond spatial-only approaches.
- We propose two novel modules, SCCGM for spatial–channel fusion and LGCM for topological graph modeling, combined with the principle of embedding Mamba locally for fine-grained correlation and linear complexity.
- OctMamba achieves state-of-the-art performance across LiDAR and dynamic human point cloud benchmarks while maintaining practical encoding speed and scalability.

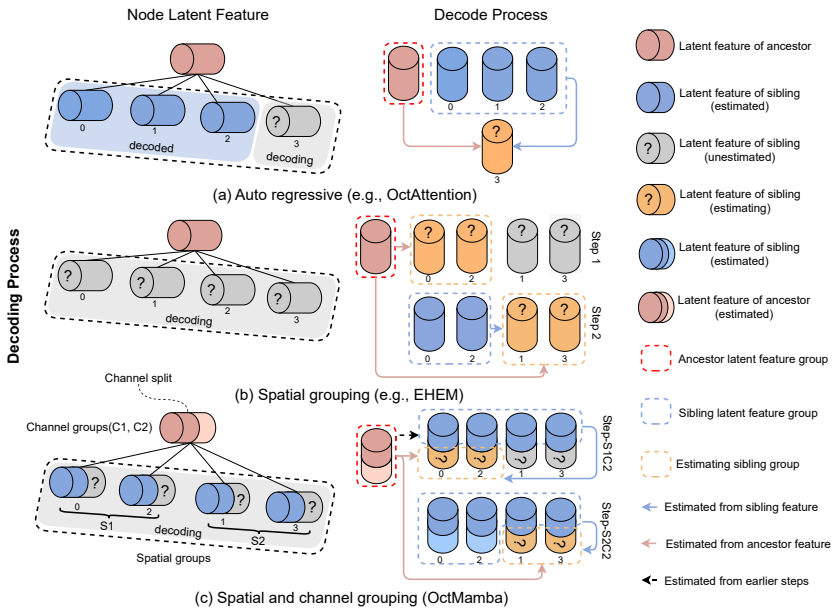


Figure 2: Comparison of octree-based point cloud compression approaches. (a) **Autoregressive (e.g., OctAttention [5]):** Sequential decoding restricts each undecoded node ("?") to access only previously decoded nodes (blue), resulting in high latency and no context sharing among nodes at the same level. (b) **Spatial grouping (e.g., EHEM [6]):** Nodes are processed in isolated spatial groups, so the first-row undecoded groups ("?") lack access to decoded siblings at the same level, resulting in fragmented context across group boundaries. (c) **Spatial and channel grouping (OctMamba):** All undecoded nodes ("?") at the current level can access information from all decoded siblings (blue), enabling full spatial-channel context integration within the level and efficient multi-dimensional redundancy exploitation with linear complexity.

2. Related Work

2.1. Learned Point Cloud Geometry Compression

Recent deep learning-based methods have achieved impressive results in point cloud compression by leveraging diverse data structures and models [20, 21, 22, 23]. Many adopt PointNet-style architectures [12, 24] or sparse voxel grids, encoding latent representations through hyper-prior entropy models to enable lossy compression at low bitrates. However, the transformation to latent space often distorts high-frequency details due to limited capacity in low-dimensional representations. Mamba-PCGC [22] recently introduced a state-space model to extend contextual dependencies with minimal complexity, but its voxel-based, globally focused framework struggles with point

cloud sparsity.

In lossless geometry compression, octree-based learned entropy models have demonstrated strong performance [5, 6], minimizing spatial partitioning loss via hierarchical structures. Some methods combine voxel data with octrees for improved local feature extraction [9], though often at high computational cost. For example, OctAttention [5] captures rich octree contexts but suffers from slow autoregressive decoding. Recent efforts such as EHEM [6] and ECM-OPCC [25] enhance global attention efficiency, yet still face the quadratic complexity inherent to attention mechanisms.

A key limitation across existing methods is their predominant focus on spatial redundancy, often overlooking additional multi-dimensional feature correlations. Feature channels, including encoding geometry, spatial context, and semantics, exhibit strong interdependencies, while topological structures represent point connectivity and higher-order geometry, both of which are critical for preserving structural fidelity. However, the high computational cost of modeling these dependencies has hindered their integration into current frameworks.

To address this, we propose OctMamba: a unified framework that explicitly incorporates spatial, channel, and topological correlations for point cloud geometry compression. By combining octree representations with the efficient Mamba sequence model, OctMamba captures multi-dimensional dependencies while maintaining low complexity and strong compression performance.

2.2. Multi-dimensional Redundancy Beyond Point Clouds

Research on hyperspectral image (HSI) denoising offers complementary evidence that jointly modeling correlations across multiple axes is beneficial. The SLRP–DSP framework demonstrates that explicitly combining spectral low-rank structures with deep spatial priors effectively suppresses noise while preserving spatial detail [26]. A recent survey on HSI denoising systematically compares model-driven, data-driven, and model–data-driven paradigms, emphasizing the importance of coordinated spectral–spatial priors [27]. Additionally, a TDSAT method couples band-wise and spatial features through attention mechanisms [28].

These approaches operate on grid-structured images and primarily address spectral–spatial axes, with spectra loosely analogous to latent channels in our context, but do not incorporate explicit octree hierarchy or graph/topological relations. Furthermore, their attention-based designs typically incur quadratic complexity in sequence length, while our approach utilizes linear-time state-space modeling to integrate spatial, channel, and topological dependencies within octree sequences.

2.3. State Space Models

Linear state-space models (SSMs) [29, 30] have recently gained traction for sequence modeling, combining latent state transitions with deep learning. The S4 model [31] improved efficiency via convolution while retaining RNN-like strengths. Mamba [32] further advanced SSMs by introducing a selective mechanism that achieves linear complexity, in contrast to the quadratic cost of Transformers [33], enabling its adoption in vision [34], graphs [35], and point cloud processing [36, 37, 38, 39].

In point cloud analysis, PointMamba [39] adapts Mamba using space-filling curves and geometric reordering, while Mamba3D [38] introduces bidirectional channel modeling to enhance local feature extraction. PointCloudMamba [37] proposes Consistent Traverse Serialization with order prompts to preserve spatial adjacency and inform sequence structure. These approaches demonstrate Mamba's utility for tasks like classification and segmentation, focusing mainly on spatial and sequential modeling.

In compression, Mamba-PCGC [22] applies Mamba globally for point cloud geometry compression but does not exploit multi-dimensional feature correlations. Our work addresses this gap by introducing OctMamba, which is a unified framework that integrates spatial, channel, and topological features using Mamba's efficient sequence modeling. Unlike prior methods that apply Mamba globally [36, 38] or stack Mamba blocks for long-range spatial modeling [37, 22], we embed Mamba modules locally within LGCM and SCCGM subcomponents. This localized strategy enables fine-grained modeling and fusion across dimensions, resulting in structurally aware, topologically informed, and channel-sensitive representations that improve compression fidelity and efficiency.

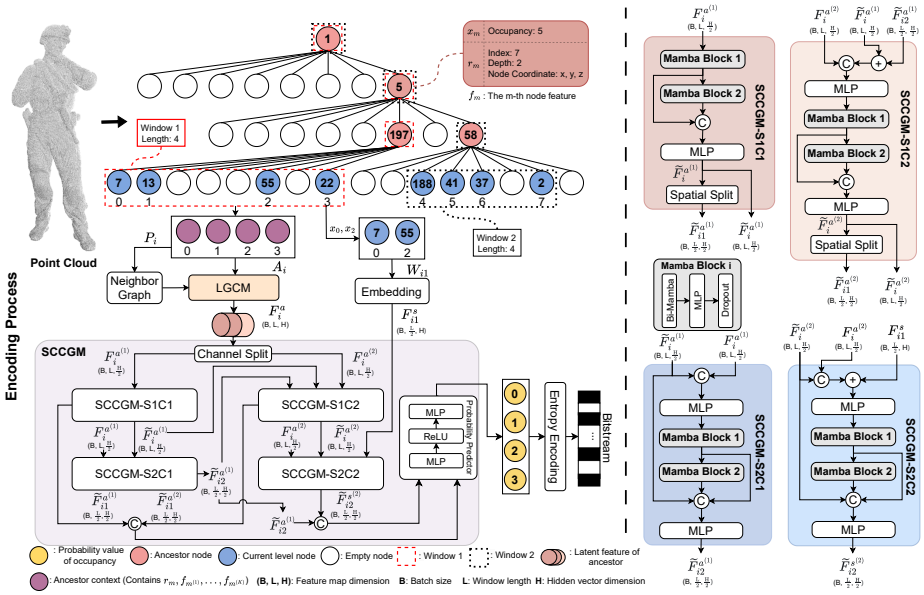


Figure 3: **Architecture Overview. Left:** The point cloud is organized as an octree with node occupancy (colored circles) and Z-order indices (numbers), divided into non-overlapping context windows (e.g., Window 1 and Window 2, N=4) shown by red dashed and black dotted boxes. With D=3 ancestor levels, ancestor (red) and sibling (blue) features in each window are aggregated and fused by LGCM, capturing local topology (via neighborhood graphs) and hierarchical context. The fused features are split and fed to SCCGM. **Right:** SCCGM has four spatial–channel groups (S1C1, S1C2, S2C1, S2C2), each with Mamba stages, MLPs, and split operations. Inter-group dependencies (arrows) propagate ancestor and sibling context from earlier (e.g., S1C1) to later groups (e.g., S2C2), enabling multi-dimensional redundancy modeling for occupancy prediction and entropy coding. Feature tensor dimensions are annotated; notation definitions are in Tab. 1.

3. Method

3.1. Overview

The core challenge in point cloud geometry compression lies in effectively modeling not only spatial redundancy but also the rich channel and topological correlations present in the data. Most existing methods primarily exploit spatial context, leaving other dimensions underutilized. OctMamba addresses this gap through a unified framework that captures and integrates spatial, channel, and topological features, enabling more efficient compression with high geometric fidelity (Figure 3). To aid understanding, key mathematical symbols used in the following sections are summarized in Tab. 1,

Table 1: Notations

Notation	Description
M	Total number of nodes in the octree
m	Index of the node in the octree's Z-order traversal.
X	Occupancy sequence of all nodes via Z-order traversal in the octree
i	Index of the occupancy window
W_i	The i-th occupancy window
N	Length of W_i
D	Number of ancestor levels used in context
x_m	Occupancy of the m-th node in the octree
r_m	Structural metadata of the m-th node (octant index, coordinates, and depth)
f_m	Node feature, defined as $f_m = \{x_m, r_m\}$
a_m	Ancestor context of the m-th node, defined as $a_m = \{r_m, f_{m^{(1)}}, \ldots, f_{m^{(L)}}\}$
$m^{(d)}$	The index of the d-th ancestor of node m
A_i	Ancestor context for W_i
E_i	Local topological features for W_i
C_i	Context window for W_i
P_i	Coordinates of nodes in the i-th window
F_i^a	Ancestor latent feature for window i; superscript ^{a} stands for ancestor-based latent feature
$F_i^{a^{(1)}}$, $F_i^{a^{(2)}}$	Two channel-wise groups of F_i^a; superscript $^{(1)}$ or $^{(2)}$ denotes channel grouping
W_{i1}, W_{i2}	Spatial subgroups of window W_i; subscript $_1$ or $_2$ indicates spatial grouping
F_{i1}^s	Embedded feature of spatial group W_{i1}; superscript ^{s} indicates sibling-based latent feature
$\widetilde{F}_{i2}^{s^{(2)}}$	Sibling-context-inferred feature for spatial group W_{i2} from the second $^{(2)}$ channel group

with brief descriptions covering indices, structural features, context, and intermediate tensors. Specifically, our approach consists of two main components:

- **Local Graph CNN-Mamba (LGCM)** builds local neighborhood graphs among sibling octree nodes to extract topological features reflecting connectivity and geometry, fused with ancestor context to enhance structural representation.
- **Spatial-Channel Coupled Grouping Mamba (SCCGM)** jointly models spatial-channel features in grouped stages, where earlier groups inform later ones, enabling accurate occupancy probability estimation via richer spatial-channel interactions.

OctMamba captures multi-dimensional dependencies and improves occupancy prediction for each octree node by cascading LGCM and SCCGM, with Mamba embedded in LGCM and each SCCGM submodule for fine-grained feature modeling and fusion. This design reduces bitrate on evaluated datasets while maintaining competitive runtime. Its unified modeling of spatial, channel, and topological redundancies uses a linear-complexity architecture, making OctMamba both novel and efficient.

3.2. Multi-Dimensional Feature Processing Pipeline

To implement the multi-dimensional modeling, OctMamba processes the point cloud through a structured pipeline comprising sequence construction, hierarchical feature extraction, and probabilistic occupancy modeling. Each step is designed to incrementally encode the spatial, topological, and channel dependencies essential for high-fidelity compression.

1. Octree-Based Sequence Construction: A point cloud is encoded into an octree [40]. Each voxel has 8-bit occupancy and is organized via Z-order traversal into a sequence $X = \{x_1, x_2, \ldots, x_M\}$, where x_m is the occupancy of the m-th node and M is the total number of nodes. X is divided into fixed-length, non-overlapping windows W_i of size N, with $X = W_1 \cup ... \cup W_{M/N}$, enabling localized processing and parallelization. At each node, x_m is paired with structural metadata r_m (octant index, coordinates, depth) forming the node feature $f_m = \{x_m, r_m\}$. The ancestor context of a node comprises features from its D most recent ancestors: $a_m = \{r_m, f_{m^{(1)}}, \ldots, f_{m^{(D)}}\}$. For window W_i, its ancestor context is $A_i = \{a_{i\times N-N+1}, \ldots, a_{i\times N}\}$, and the contextual information including neighbors' occupancies and A_i is the context window C_i. Ancestors are zero-padded when fewer than D exist.

2. Hierarchical Feature Extraction: Once the context windows are formed, feature extraction proceeds through two coordinated modules that specialize in distinct dependency types:

- The **LGCM** module processes each window by constructing a local voxel-level topological graph among the neighors. This graph is passed through an EdgeConv operator and fused with ancestor context via a Mamba block, yielding enhanced representations F_i^a that encode both local structure and global hierarchy.

- The **SCCGM** module refines F_i^a via a stepwise decoding strategy that progressively integrates earlier channel groups. Features are split along spatial and channel dimensions, allowing contextual reuse. This staged interaction between decoded and undecoded features enables SCCGM to iteratively produce more accurate occupancy distributions for each node in W_i.

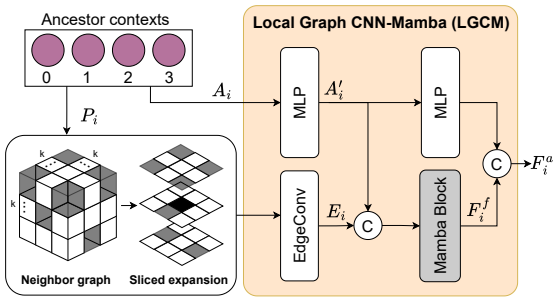


Figure 4: Local Graph CNN-Mamba architecture: neighbor graph with local neighborhood (left) and sliced expansion (right). Gray partially shaded squares are existing nodes, white unshaded squares are empty nodes, the black solid square is the current node, and C denotes concatenation along the feature dimension.

3. Entropy Modeling and Bitstream Generation: The final stage estimates the conditional occupancy distribution for the entire sequence:

$$\widetilde{q}(X) = \prod_{i=1}^{M/N} \widetilde{q}_i(W_i \mid C_i), \tag{1}$$

where $\widetilde{q}_i$ is the predicted conditional probability of the i-th window. Assuming conditional independence between windows, this formulation enables scalable entropy estimation. Lossless compression is achieved via arithmetic encoding based on the predicted distribution $\widetilde{q}(X)$, producing a compact bitstream suitable for lossless reconstruction. A Z-order curve [41] guides the octree traversal, preserving spatial coherence and ensuring compatibility with OctMamba's linear-complexity decoding path.

3.3. Local Graph CNN-Mamba (LGCM)

To enhance coding performance, the LGCM module explicitly models the fine-grained local topology, which is often overlooked by methods that exploit solely on hierarchical or global spatial cues. As illustrated in Fig. 4, LGCM constructs a neighbor graph within a $k \times k \times k$ spatial region to enrich each node's representation by capturing relational structure at the same resolution level.

The module takes ancestor-derived features A_i and local neighbor node coordinates P_i as input, combining hierarchical structure and topological information to produce the enhanced ancestor latent feature F_i^a. This process consists of the following steps:

Local Topology Encoding: A local neighbor graph $\mathrm{NG}(P_i)$, with P_i as node coordinates, is built by connecting each center voxel to its non-empty neighbors within a

$k \times k \times k$ voxel block. Topological features E_i are then extracted via EdgeConv [42]: $E_i = \text{EdgeConv}(\text{NG}(P_i))$. Although EdgeConv was originally designed for point clouds, here it operates on octree nodes. At each level, the centers of non-empty nodes form a coarse point cloud that preserves spatial structure, so neighbor graphs on these multi-scale nodes yield topological features aligned with the original point cloud and compatible with the octree-based compression pipeline.

Feature Projection and Fusion: The ancestor-derived features A_i are first projected into an embedded space via a multilayer perceptron (MLP), yielding $A'_i = \text{MLP}(A_i)$. The local topological features E_i and the projected ancestor features A'_i are then concatenated along the channel dimension and processed by a Mamba block to fuse local structures with the global context features:

$$F_i^f = \text{Mamba}(E_i \copyright A'_i), \tag{2}$$

Residual Enhancement: The final ancestor latent feature F_i^a is obtained by combining the Mamba output F_i^f with a further transformed version of A_i, acting as a residual connection to preserve information:

$$F_i^a = F_i^f \copyright \text{MLP}(A'_i). \tag{3}$$

By integrating local graph reasoning with global sequence modeling, LGCM aggregates local topology via EdgeConv and captures long-range dependencies through a linear state-space model [32]. This linear-complexity approach significantly improves occupancy prediction accuracy and structural consistency.

3.4. Spatial-Channel Coupled Grouping Mamba (SCCGM)

Traditional octree-based entropy models struggle to fully exploit contextual information during decoding. Autoregressive methods rely on overlapping windows that extend backward from the target node until enough context is gathered. While this captures fine-grained dependencies, it suffers from high latency due to its inherently sequential nature. In contrast, spatial grouping methods utilize non-overlapping windows to allow parallel processing, but this limits intra-level context because roughly half of the nodes depend only on ancestor features, not on sibling relationships.

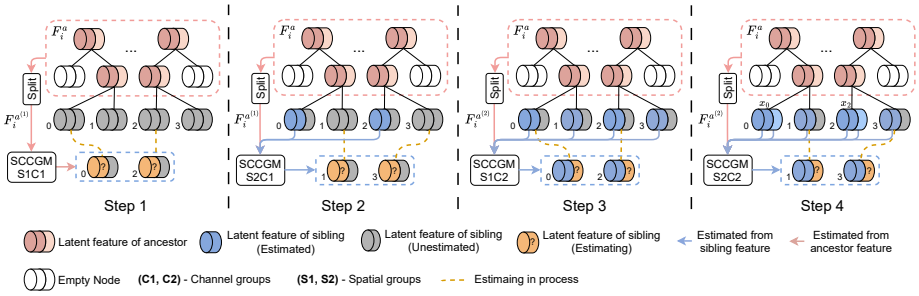


Figure 5: SCCGM's four-stage hierarchical context estimation during entropy decoding (window size $N = 4$). **Top:** Ancestor latent features F_i^a split into channel groups $F_i^{a^{(1)}}$ and $F_i^{a^{(2)}}$. Node states: red boxes (ancestor features), grey (unprocessed channels), blue (completed context), orange dashed "?" (current targets). **Bottom:** Sequential activation (S1C1 → S2C1 → S1C2 → S2C2). Progressive context integration: S1C1 processes $F_a^{(1)}$; S2C1 uses S1C1 output; S1C2 integrates priors; S2C2 leverages all prior stages, enriching context for enhanced occupancy probability estimation via decoded sibling features (x_0, x_2).

The SCCGM module uses a structured decoding scheme integrating spatial and channel-wise context. Unlike autoregressive or spatial grouping methods, which ignore sibling interactions or suffer from sequential bottlenecks, SCCGM carefully and progressively unifies ancestor- and sibling-derived features via grouped refinement strategy.

Within each context window, We first consider the ancestor latent feature $F_i^a \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\mathrm{H}}$, where B denotes the batch size, L is the number of nodes within a window (spatial dimension), and H is the hidden vector dimension (channel dimension); this feature is then partitioned along the channel dimension into groups denoted as $F_i^{a^{(c)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{N^{(c)}}}$, where $c \in \{1, \ldots, N^{(c)}\}$ indexes the channel group and $N^{(c)}$ is the total number of channel groups. Each channel group is further divided into two spatial subgroups according to the even and odd indices in the window sequence. This forms a hierarchical structure in which decoding proceeds in a stepwise manner: channel subgroups that are computed earlier provide contextual information for subsequent subgroups, while spatial subgroups that are decoded earlier provide sibling occupancy information for the later ones.

Formally, the full ancestor latent feature $F_i^a \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\mathrm{H}}$ is split into two channel partitions, $F_i^{a^{(1)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$ and $F_i^{a^{(2)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$, such that $F_i^a = F_i^{a^{(1)}} \copyright F_i^{a^{(2)}}$, where © de-

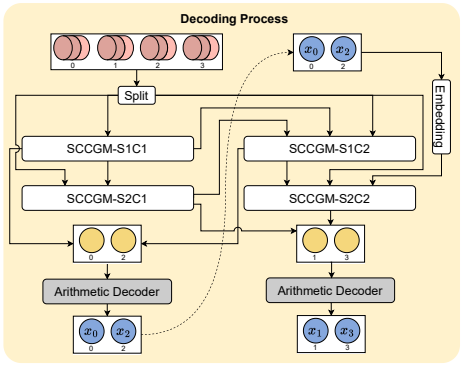


Figure 6: Decoding process illustrated with two channel groups. Red components indicates ancestral context, blue components denotes sibling context, and yellow components shows the predicted occupancy probability.

notes concatenation along the channel dimension. Similarly, the window $W_i \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times 1}$ is divided into two spatial subgroups: $W_{i1} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times 1}$, consisting of even-indexed occupancies, and $W_{i2} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times 1}$, consisting of odd-indexed ones. This two-level grouping, first by channel, then by spatial index, supports a stepwise refinement strategy in which earlier-computed subgroups condition the computing of subsequent ones. Specifically, we define the encoding and decoding process below.

Encoding Process: Fig. 3 illustrates the overall encoding architecture, with specific steps detailed in Fig. 5:

$$
\begin{aligned}
\text{Step 1:}\quad & \widetilde{F}_i^{a^{(1)}} = \text{SCCGM-S1C1}(F_i^{a^{(1)}}),\\
\text{Step 2:}\quad & \widetilde{F}_{i2}^{a^{(1)}} = \text{SCCGM-S2C1}(F_i^{a^{(1)}}, \widetilde{F}_i^{a^{(1)}}),\\
\text{Step 3:}\quad & \widetilde{F}_i^{a^{(2)}} = \text{SCCGM-S1C2}(F_i^{a^{(2)}} \widetilde{F}_i^{a^{(1)}}, \widetilde{F}_{i2}^{a^{(1)}}),\\
\text{Step 4:}\quad & \widetilde{F}_{i2}^{s^{(2)}} = \text{SCCGM-S2C2}(F_i^{a^{(2)}}, \widetilde{F}_i^{a^{(2)}}, F_{i1}^{s}),
\end{aligned}
$$

In ***Step 1***, $F_i^{a^{(1)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$ is fed into SCCGM-S1C1 to obtain the first channel-group feature $\widetilde{F}_i^{a^{(1)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$, which contains subgroup $\widetilde{F}_{i1}^{a^{(1)}} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times\frac{\mathrm{H}}{2}}$. In ***Step 2***, $F_i^{a^{(1)}}$ together with $\widetilde{F}_i^{a^{(1)}}$ is passed to SCCGM-S2C1, producing $\widetilde{F}_{i2}^{a^{(1)}} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times\frac{\mathrm{H}}{2}}$ as context applied in the probability predictor. In ***Step 3***, $\widetilde{F}_{i2}^{a^{(1)}}$, $F_i^{a^{(2)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$ and the complete first-channel-group feature are supplied to SCCGM-S1C2, yielding the second channel-group feature $\widetilde{F}_i^{a^{(2)}} \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$. Then, its first spatial subgroup $\widetilde{F}_{i1}^{a^{(1)}}$ together with $\widetilde{F}_{i1}^{a^{(2)}} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times\frac{\mathrm{H}}{2}}$ is exploited to predict the occupancy probabilities of the first spatial subgroup and thus applied to encode or decode its occupancy values W_{i1}, which

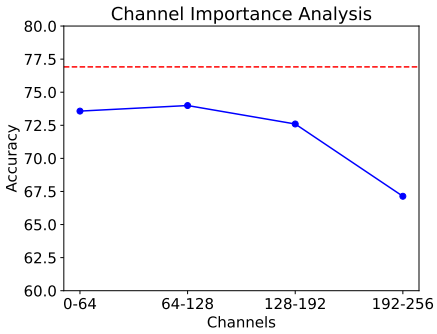


Figure 7: Effect of Different Channel Masking on Compression Performance. Masking later channel leads to accuracy drop. It shows the importance of later channel features.

is embedded as $F^s_{i1} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times\mathrm{H}}$. **Step 4:** F^s_{i1} is concatenated with both $\widetilde{F}^{a^{(2)}}_i \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$ and $F^{a^{(2)}}_i \in \mathbb{R}^{\mathrm{B}\times\mathrm{L}\times\frac{\mathrm{H}}{2}}$, and the combined features are fed into SCCGM-S2C2, which generates the remaining features $\widetilde{F}^{s^{(2)}}_{i2} \in \mathbb{R}^{\mathrm{B}\times\frac{\mathrm{L}}{2}\times\frac{\mathrm{H}}{2}}$. These are then combined with $\widetilde{F}^{a^{(1)}}_{i2}$ to predict the occupancy probabilities of the second spatial subgroup, which are used to encode its occupancy values W_{i2}. Specifically, as shown in Fig. 5, SCCGM-S2C2 integrates ancestral context, the first channel group, and the known occupancy values of nodes 0 and 2 to decode the second channel group for nodes 1 and 3. The predicted occupancy probabilities are subsequently passed to an arithmetic encoder. This hierarchical and cumulative scheme allows each subgroup to effectively incorporate relevant ancestor and sibling information, improving both the prediction accuracy and compression efficiency of OctMamba.

Decoding process SCCGM decoding (Fig. 6) mirrors encoding: nodes 0 and 2 decode first, providing context for nodes 1 and 3. By coupling spatial and channel groups, our method enables all nodes within a window to exploit intra-level context, greatly improving decoding efficiency and accuracy over purely spatial grouping methods. Furthermore, encoding and decoding across different windows within the same level are independent and can be executed in parallel, whereas cross-level computation must proceed serially due to dependencies on ancestor nodes. Since both employ the same two-stage procedure across windows and exhibit identical intra-level parallelism, their runtimes are of the same order.

Empirical analysis (Fig. 7 and ablation in Tab. 9) show that later channel groups

carry greater semantic importance. We therefore divide channels unevenly, assigning larger capacity to early blocks and finer granularity to later ones, ensuring influential features benefit from richer prior context. Within the Mamba framework, SCCGM maintains linear complexity, enabling efficient long-range dependency modeling without autoregressive delays. Overall, SCCGM unifies intra-window dependency modeling and improves occupancy probability estimation in our experiments.

3.5. Probabilistic Modeling and Optimization

To support accurate and efficient entropy coding, OctMamba formulates occupancy prediction as a conditional probability estimation task. Given contextual features C_i extracted per window, the model outputs the probability distribution $\widetilde{q}_i(W_i \mid C_i)$ over node occupancies W_i. This prediction is optimized via a cross-entropy loss, which penalizes deviations from the true distribution q_i:

$$\mathcal{L} = -\sum_i q_i \log \widetilde{q}_i(W_i \mid C_i) \tag{4}$$

$$\widetilde{q}_i(W_i \mid C_i) = \text{SCCGM}(\ \text{LGCM}(C_i)\ ). \tag{5}$$

The context C_i encodes rich structural priors, including occupancy history, spatial hierarchy, and octree metadata (e.g., octant index, coordinates, and depth), enabling precise modeling of local dependencies. Unlike prior entropy models that treat context in a limited or sequential fashion, OctMamba's formulation fully exploits the fused multi-dimensional features produced by LGCM and SCCGM. This integration allows for more faithful occupancy probability estimation and directly contributes to bitrate reduction while maintaining decoding accuracy.

4. Experiments

4.1. Experimental Settings.

Datasets Experiments were conducted on SemanticKITTI [16] and Ford [17] LiDAR datasets, the human-object dataset MVUB [18], and MPEG 8i [19].

LiDAR Dataset. SemanticKITTI consists of 43,552 LiDAR scans from autonomous driving scenarios, organized into 22 sequences. For effective model training and evaluation, we adhere to the default dataset split: sequences 00 to 10 are designated as

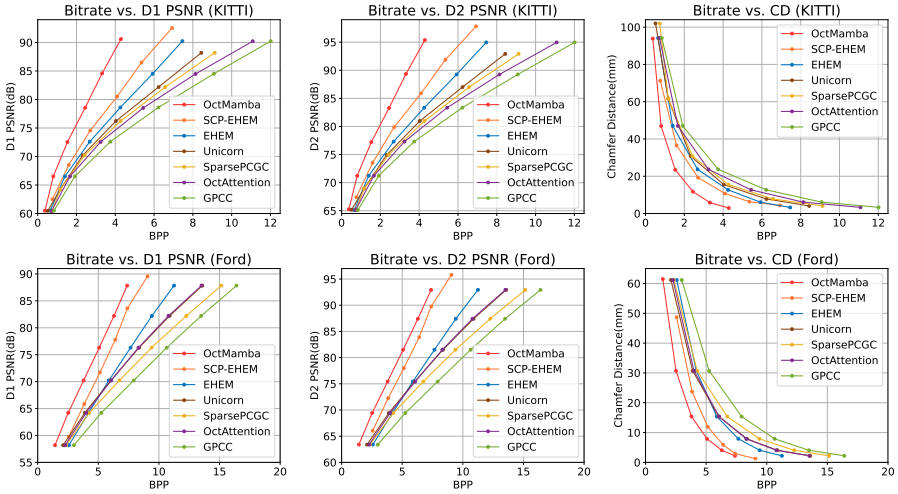


Figure 8: Rate-distortion performance of different methods on SemanticKITTI and Ford.

Table 2: BD-rate gains (%↓) of different methods over the G-PCC baseline on the Ford and SemanticKITTI datasets in terms of D1 PSNR, D2 PSNR, and Chamfer Distance (CD).

Method	Ford			SemanticKITTI		
	D1 PSNR	D2 PSNR	CD	D1 PSNR	D2 PSNR	CD
SparsePCGC [21]	-14.0%	-14.0%	-17.6%	-15.7%	-10.8%	-6.5%
OctAttention [5]	-22.2%	-22.2%	-22.8%	-22.4%	-26.6%	-24.6%
EHEM [6]	-25.6%	-25.6%	-21.0%	-29.8%	-33.4%	-26.4%
SCP-EHEM [43]	-40.8%	-43.9%	-37.7%	-40.6%	-40.3%	-37.7%
GAEM [8]	-42.8%	-45.8%	-39.5%	-53.9%	-54.1%	-49.4%
Ours	**-52.7%**	**-52.6%**	**-52.6%**	**-60.2%**	**-60.3%**	**-59.0%**

the training set, while sequences 11 to 21 serve as the test set. The Ford dataset is a LiDAR point cloud dataset utilized in the MPEG point cloud geometry compression standardization. It comprises three sequences, each containing 1,500 scans. According to the MPEG standard [44], sequence 01 is designated for training, while sequences 02 and 03 are reserved for evaluation.

Object Point Cloud Dataset. MVUB is a voxelized point cloud dataset featuring dynamic human bodies, encompassing five sequences with 9-bit and 10-bit precision.

Table 3: This table reports the bitrate (bpp ↓) and PSNR results for various methods on the MVUB and MPEG 8i datasets. The table also highlights the bitrate gain of OctMamba over G-PCC, demonstrating its superior compression efficiency.

Dataset	Sequence	G-PCC	VoxelDNN	OctAttention	EHEM	ECM-OPCC	OctMamba (Ours)	Gain over G-PCC
MVUB	Phil10	1.07	0.83	0.79	-	0.76	**0.34**	-68.2%
	Phil9	1.23	0.92	0.83	-	0.79	**0.40**	-67.4%
	Ricardo10	1.07	0.75	0.72	-	0.69	**0.38**	-64.5%
	Ricardo9	1.04	0.72	0.72	-	0.76	**0.34**	-67.3%
	Average	1.10	0.81	0.76	-	0.75	**0.37**	-66.3%
8iVFB	Loot10	0.95	0.64	0.62	0.58	0.55	**0.27**	-71.6%
	Redandblack10	1.09	0.73	0.73	0.69	0.66	**0.33**	-69.8%
	Average	1.02	0.69	0.67	0.64	0.61	**0.30**	-70.6%
Average		1.06	0.75	0.72	0.64	0.68	**0.34**	-68.5%

Table 4: Bitrate comparison (bpp ↓) between the lossy Mamba-PCGC and the proposed lossless OctMamba on selected MVUB and MPEG 8i sequences. For Mamba-PCGC, the D1 PSNR values are reported in parentheses.

Dataset	Sequence	Mamba-PCGC (bpp, D1)	OctMamba (Ours)
MVUB	andrew10	0.37 (D1 67.1)	**0.37**
8iVFB	Soldier10	0.42 (D1 76.5)	**0.30**
8iVFB	Longdress10	0.41 (D1 76.3)	**0.29**
Average		0.40 (D1)	**0.34**

We utilize the Andrew10, David10, and Sarah10 sequences for training, reserving the remaining sequences for testing. The MPEG 8i dataset includes various human shape point clouds with 10-bit and 12-bit precision. Following the dataset split in OctAttention [5], Soldier10 and Longress10 are used for training, while Loot10 and redandblack10 are employed for testing. To further validate the generalization capability of our model, we perform cross-experimentation by interchanging the training and testing sets within both MPEG 8i and MVUB.

Baselines. We comprehensively compared OctMamba with several methods. First, we compared it with sparse convolution-based techniques, including Unicorn [23],

Mamba-PCGC [22], and SparsePCGC [21], while Unicorn supports both lossy and lossless compression, the others primarily perform lossy compression on voxelized representations of point clouds. Next, we compared it with 3D CNN-based point cloud geometry compression entropy models, such as VoxelDNN [45]. Additionally, we included large-scale context-based point cloud compression models utilizing attention mechanisms, namely GAEM [8], SCP [43], EHEM [6], ECM-OPCC [25] and OctAttention [5], all of which are dedicated to lossless point cloud compression. Finally, we also benchmarked OctMamba against the representative handcrafted compression method MPEG G-PCC [20].

Implementation Settings. To validate the effectiveness of our OctMamba, we begin by constructing an octree from the point cloud data. For the SemanticKITTI dataset, we adhere to the quantization settings proposed in OctSqueeze [3], utilizing a quantization step size defined as $\frac{400}{2^D-1}$ and set the quantization step to 2^{18-D} on Ford, where D indicates the octree depth. We employ a maximum depth of 16 for SemanticKITTI and Ford, while for the MVUB and MPEG 8i datasets, the octree is segmented into 10 levels during both training and testing phases. The Spatial-Channel Coupled Grouping Mamba (SCCGM) module's Mamba stage comprises 3 layers, with feature channel dimensions configured to 256. The ancestor depth K is set to 3 , context window size N is 8192, and channel groups are specified as 128, 64, and 64. For the LGCM, the local voxel block size set to 3. For training, the learning rate is fixed at 10^{-4} for SemanticKITTI and Ford, while for MVUB and MPEG 8i, it is set to 10^{-3}. All models undergo training for 10 epochs, with evaluations performed on an NVIDIA RTX 4090.

Metrics. We evaluate distortion using four metrics: point-to-point PSNR (D1 PSNR), point-to-surface PSNR (D2 PSNR) [46], Chamfer Distance (CD), and BD-Rate. To ensure consistency with prior works, such as MuSCLE and OctSqueeze [47, 3], we adopt a PSNR peak value of 59.70 when evaluating the SemanticKITTI dataset [16]. This peak value represents the maximum possible range of reconstructed point cloud values used in the calculation of PSNR (Peak Signal-to-Noise Ratio) [6]. For the Ford dataset, a peak value of 30,000 is used, reflecting its unique characteristics in terms of range and scale [48]. By aligning our testing environment with these previous methods, we enable fair and direct comparisons of rate-distortion performance.

It is important to note that the use of these peak values does not influence the compression performance of our method but ensures that the reported PSNR results are comparable across studies. This standardization reinforces the validity and relevance of our experimental findings within the context of established benchmarks.

BD-Rate (Bjøntegaard delta rate) [49] is a metric used to quantify the average bitrate difference between two video encoders at the same distortion level. To compute BD-Rate, the rate-distortion (RD) data for both encoders are first fitted using cubic polynomials, resulting in the functions $R_A(D)$ and $R_B(D)$. Next, the integrals of these functions are calculated over the common distortion range $[D_{\min}, D_{\max}]$, yielding $\int_{D_{\min}}^{D_{\max}} R_A(D)\, dD$ and $\int_{D_{\min}}^{D_{\max}} R_B(D)\, dD$. The core formula for BD-Rate is given by:

$$\Delta R = \frac{\int_{D_{\min}}^{D_{\max}} R_B(D)\, dD - \int_{D_{\min}}^{D_{\max}} R_A(D)\, dD}{\int_{D_{\min}}^{D_{\max}} R_A(D)\, dD} \tag{6}$$

A negative ΔR value indicates a bitrate reduction (performance improvement), while a positive value signifies an increase in bitrate (performance degradation). Using the BD-Rate metric on the SemanticKITTI dataset, our method achieved a 38.27 % bitrate reduction compared to the state-of-the-art EHEM, demonstrating its superior compression efficiency.

4.2. Performance Evaluation

Our quantitative results (Fig. 8) show that OctMamba surpasses all baseline compressors, such as the sparse-convolution Unicorn and the advanced entropy model EHEM, owing to its efficient fusion of spatial, channel, and octree-topological cues. In particular, OctMamba achieves a 42.8% BD-Rate reduction relative to EHEM on SemanticKITTI and a 36.4% saving on the Ford LiDAR dataset, based on the average D1-PSNR difference. Additionally, Tab. 2 reports the BD-rate gains of SCP-EHEM [43], GAEM [8], and our method over the G-PCC baseline on the Ford and SemanticKITTI datasets, where our method also demonstrates a considerable advantage. These results highlight OctMamba's superior compression efficiency in maintaining quality while substantially reducing data requirements.

In Tab. 3, we compare lossless compression performance across several representative codecs on voxelized point cloud datasets. On MVUB, OctMamba reduces the

average bitrate from 1.10 bpp (G-PCC) to 0.37 bpp, corresponding to a 66.3% bitrate saving while also outperforming stronger learned baselines such as VoxelDNN, OctAttention, and ECM-OPCC on every sequence. A similar trend is observed on MPEG 8i: OctMamba lowers the average bitrate from 1.02 bpp (G-PCC) to 0.30 bpp, achieving 70.6% savings and consistently yielding the lowest bitrate among all lossless methods. Overall, the averaged gain of 68.5% over G-PCC demonstrates that OctMamba delivers substantially more efficient lossless geometry compression across both MVUB and MPEG 8i.

Our method is designed for lossless geometry compression and is therefore mainly compared against lossless baselines. For completeness, we additionally report in Tab. 4 a reference comparison with the lossy Mamba-PCGC, which does not provide exact reconstruction. In this case, we take its bit-per-point (bpp) at the peak D1-PSNR as the operating point. Moreover, because the training/test split used by Mamba-PCGC on the 8i dataset is opposite to that adopted by most lossless methods (including OctMamba), we follow its experimental protocol by swapping the training and testing sets to ensure a fair comparison. As summarized in Tab. 4, OctMamba achieves better lossless compression performance even when compared against this high-quality lossy codec.

We also evaluated the compression efficiency of various methods. Tab. 5 reports encoding and decoding times for different quantization steps, highlighting OctMamba's effectiveness. Notably, decoding requires reconstructing the octree at each level from already decoded results, so its latency is slightly higher than encoding but remains of the same order.

We compress the occupancy sequence after octree construction; reported encoding and decoding times exclude octree building, following the same protocol as OctAttention for fair comparison. Tab. 6 compares FLOPs, GPU memory (GPU Mem.), and model parameters (Params) for a single window of $N = 8192$. OctMamba shows lower complexity than previous methods, highlighting its compression efficiency. Its channel grouping requires inference steps equal to the number of channel groups, while EHEM and OctAttention require only two and one, respectively. Despite this, Mamba's linear complexity keeps encoding speed competitive for large point clouds, confirming OctMamba's practical efficiency.

Table 5: Comparison of encoding / decoding times (in seconds) for different methods at various octree levels (D) in the KITTI dataset.

Method	D=12	D=14	D=16
G-PCC	0.25 / 0.12	0.66 / 0.32	1.07 / 0.54
SparsePCGC	1.14 / 0.86	1.76 / 1.43	2.43 / 2.04
OctAttention	0.08 / 83.00	0.31 / 321.00	0.66 / 708.00
EHEM	0.40 / 0.43	1.21 / 1.39	2.53 / 3.01
SCP-EHEM	0.77 / 0.84	1.91 / 2.13	3.40 / 3.93
GAEM	0.35 / 0.43	0.97 / 1.16	1.96 / 2.38
OctMamba	0.21 / 0.33	0.52 / 0.64	0.97 / 1.19

Table 6: Comparison of model complexity per window for different methods on the KITTI dataset.

Method	FLOPs	GPU Mem.	Params
OctAttention	124.3G	**1.3G**	4.23M
EHEM	184.4G	2.9G	13.01M
OctMamba	**113.2G**	2.0G	**2.78M**

4.3. Ablation Studies and Analysis

Effectiveness of Channel Grouping and LGCM. To evaluate the impact of channel grouping and the LGCM module, we compared OctMamba with variants that exclude these components. As shown in Fig. 9, removing channel grouping results in suboptimal utilization of inter-channel correlations, leading to decreased accuracy in the entropy model. Similarly, substituting the LGCM module with a simple MLP restricts the extraction of aggregated local and global features, diminishing the effectiveness of contextual information. These findings demonstrate that both components are crucial for enhancing feature dependencies and information aggregation, ultimately improving OctMamba's performance in entropy modeling and compression.

Context Scale. OctMamba scales linearly with context size N, enabling model expansion. As Fig. 10 shows, increasing N from 2048 to 8192 expands the capture region, linking both legs of the model. This improves rate-distortion performance (Tab. 7), enhances entropy accuracy, and reduces the number of context windows, lowering com-

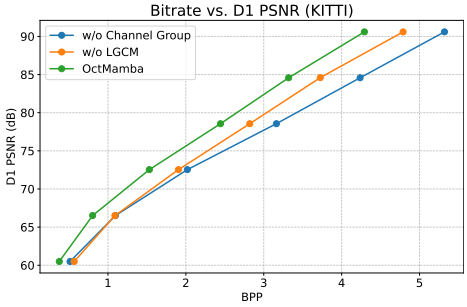


Figure 9: Ablation studies were conducted for the absence of LGCM and without channel grouping.

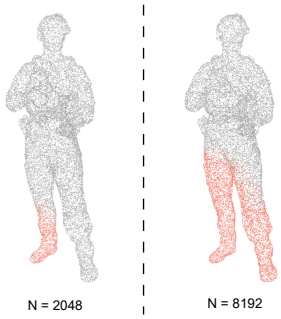


Figure 10: For MPEG-8i downsampled to 20,000 points, we visualize different context capacities, with the points in the context highlighted in red.

putation and speeding up encoding/decoding. These results demonstrate how context expansion boosts OctMamba's efficiency in capturing spatial relationships.

Neighborhood Size. The neighborhood size significantly affects the Local Graph CNN-Mamba (LGCM) module. We evaluated different k values with a fixed window size $N = 8192$ (Tab. 8). $k = 1$ uses no neighborhood, $k = 8$ a $2\times2\times2$ region, and $k = 27$ a $3\times3\times3$ region. Larger k captures richer local features but increases computation and latency. Results show $k = 27$ gives the best performance, with the lowest average bpp for Loot10 and Redandblack10 (0.271 and 0.328). Compared to KNN-based neighborhoods, our block-based approach is faster while maintaining LGCM performance. We chose $k = 27$ to fully leverage local voxel dependencies, improving compression efficiency with competitive computation.

In OctMamba, the Spatial-Channel Coupled Grouping Mamba (SCCGM) module efficiently allocates channels based on information density. To validate this, we compared different channel grouping schemes on the MPEG 8i dataset. Results (Tab. 9)

Table 7: Comparison of the impact of different context sizes N on the model for the MPEG 8i dataset.

Size N	Loot10 (bpp↓)	Redandblack10 (bpp↓)	Avg Time (s) Encoder	Decoder
1024	0.282	0.341	3.26	3.74
8192	0.271	0.328	0.80	1.08
9216	**0.267**	**0.321**	**0.78**	**1.07**

Table 8: Impact of different k values for neighborhood construction on the model (MPEG 8i dataset).

k	Loot10 (bpp↓)	Redandblack10 (bpp↓)	Avg Time (s) Encoder	Decoder
k=20(KNN)	0.273	0.334	1.35	1.54
k=1	0.278	0.337	**0.74**	**1.02**
k=8	0.278	0.334	0.75	1.03
k=27	**0.271**	**0.328**	0.80	1.08

show that uneven grouping with finer-grained division for important channels (e.g., 128,64,64) achieves superior performance (bpp↓: 0.271 for Loot10, 0.328 for Redandblack10) while minimizing encoding/decoding times (0.80s, 1.08s). Assigning fewer channels to high-information groups captures critical features, while larger groups enhance parallelism and reduce computational overhead. Compared to coarser uneven (64,64,128) or uniform (85,85,86) schemes, our design optimizes both compression and runtime, highlighting SCCGM's ability to balance computational cost with compression quality in OctMamba.

Base Model. We investigated different backbone networks in OctMamba to validate the use of Mamba. Replacing all Mamba stages with Transformer Encoder Layers (window size $N = 8192$) and 4 attention heads to capture long- and short-term dependencies, results (Tab. 10) show the transformer backbone underperforms OctMamba in efficiency and long-range modeling for octree sequences, highlighting OctMamba's integration of local and global information with linear complexity versus the transformer's quadratic complexity. Notably, these findings challenge the sufficiency of simple local-global modeling for point cloud compression. Our design strategically integrates Mamba to 1) enhance feature extraction and fusion at each submodule, 2)

Table 9: Results under different channel grouping schemes on the model (MPEG 8i dataset).

Channel Grouping Scheme	Loot10 (bpp↓)	Redandblack10 (bpp↓)	Avg Time (s) Encoder	Decoder
64,64,128	0.276	0.333	0.81	**1.06**
85,85,86	0.272	0.330	0.85	1.11
128,64,64	**0.271**	**0.328**	**0.80**	1.08

Table 10: Comparison of the results under different base model configurations on the KITTI dataset.

Base Model	Loot10 (bpp↓)	Redandblack10 (bpp↓)	Avg Time (s) Encoder	Decoder
Transformer	0.783	0.889	1.65	1.89
OctMamba	**0.271**	**0.328**	**0.80**	**1.08**

capture local-global correlations, and 3) align with iterative encoding. OctMamba is optimized for point cloud compression, achieving superior performance.

Table 11: Generalization from SemanticKITTI (training) to Ford, MPEG 8i, and MVUB (testing).

Dataset	Seq	bpp↓
MPEG 8i	Loot10	0.61
	Redandblack10	0.71
Ford	02&03	6.19 (PSNR D1 82.21)
MVUB	Andrew10	0.63
	Ricardo10	0.62
	Phil10	0.65

Generalization Ability. To evaluate the generalization ability of the OctMamba model across different datasets, we trained the model on the SemanticKITTI dataset and tested it on the Ford dataset (also a LiDAR dataset), as well as on two human point cloud datasets, MPEG 8i and MVUB. As shown in Tab. 11, the results on the Ford dataset are comparable to those obtained when training and testing on Ford itself, indicating strong intra-domain generalization. Although the performance on the human point cloud datasets, where the data distribution differs significantly, degrades slightly compared to the in-domain results, OctMamba still outperforms OctAttention, further demonstrating the strong potential of our model's generalization capability.

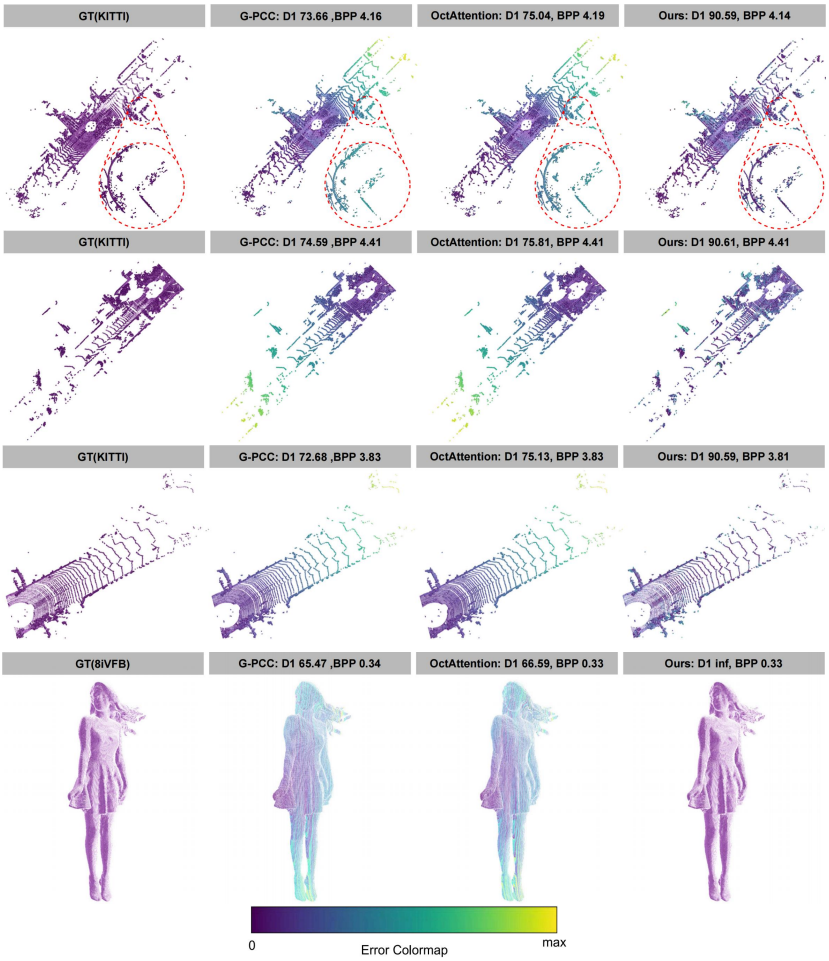


Figure 11: Visualized compression results on SemanticKITTI and MPEG 8i, with points color-coded by error levels (lighter indicating greater discrepancy from ground truth). highlight OctMamba's superior performance compared to G-PCC and OctAttention, introducing fewer errors.

Table 12: Results under different OctAttention configurations with and without LGCM.

Dataset	OctAttention(bpp↓)	OctAttention + LGCM(bpp↓)
Loot10	0.62	**0.56**
Redandblack10	0.73	**0.63**

LGCM Effectiveness. To validate the effectiveness of the LGCM module, we integrated it into the OctAttention framework for comparative experiments. The goal

Table 13: Ablation study on channel grouping on the 8iVFB sequences Loot and Redandblack. "Params" denotes the number of trainable parameters. "Loot10" and "Redandblack10" report the average bitrate (bpp).

Model	Params (M↓)	Loot10 (bpp↓)	Redandblack10 (bpp↓)
W/O Channel Group	4.93	0.375	0.425
OctMamba (Ours)	**2.78**	**0.273**	**0.334**

was to equip OctAttention with the ability to aggregate both local topological features and global contextual information. As shown in Tab. 12, OctAttention enhanced with the LGCM module demonstrates improved compression performance, benefiting from its strengthened capability to capture diverse information.

Channel Grouping Effectiveness. Channel grouping not only aggregates cross-channel information but also exploits inter-channel redundancy and enables a lightweight design. As Tab. 13 shows, disabling channel grouping and processing all latent channels jointly ("W/O Channel Group") increases parameters from 2.78M to 4.93M and degrades rate–distortion performance on the 8i dataset. On KITTI, Fig. 9 shows that channel grouping reduces BD-Rate by 16.4% at the same distortion level. These results confirm that channel grouping produces a smaller model and better compression efficiency, improving both effectiveness and implementation.

4.4. Qualitative Results

In Fig. 11 we present visual comparisons of compression artifacts produced by different methods at similar bitrates across various datasets. The colors of the point clouds are encoded based on error levels, with lighter colors indicating greater discrepancies from the ground truth. Our method, which leverages large-scale context and spatial-channel coupling, demonstrates superior modeling capabilities for sparser regions of point clouds. Notably, In the sparse regions of the first three rows, our model significantly outperforms other baseline methods. Additionally, on voxelized human point cloud datasets, our approach achieves lossless compression, as demonstrated by the distortion-free visual representations.

In contrast, other baseline methods, such as G-PCC and OctAttention, show varying levels of distortion, especially in regions with higher error levels, which are high-

lighted by lighter yellowish colors. For instance, G-PCC shows noticeable artifacts in the human model, while OctAttention retains more detail but still suffers from visible discrepancies (esp. foots). This highlights the effectiveness of our method in preserving the integrity of the data while operating at similar bits per pixel (bpp).

5. Conclusion

This study introduces OctMamba, to our knowledge, the first unified framework that exploits multi-dimensional redundancies including spatial (local and global), channel, and topological for high-fidelity point cloud geometry compression. OctMamba incorporates two novel modules: SCCGM for spatial–channel modeling and LGCM for topological encoding, guided by a new principle of localized Mamba embedding. Embedding Mamba within these subcomponents rather than globally enables fine-grained correlation modeling, achieves a smaller model and alleviates traditional autoregressive bottlenecks by enabling intra-level windowed decoding with linear complexity. This design addresses major limitations of prior work, including restricted context modeling and inefficient sequential decoding, while improving performance in sparse and boundary regions. By leveraging richer multi-dimensional context, OctMamba delivers state-of-the-art compression fidelity and efficiency across diverse benchmarks and opens promising directions for future research, such as hyperprior integration, geometric priors, and adaptive multi-dimensional strategies for next-generation 3D data compression

6. Acknowledgements

This research was partially supported by Zhejiang Province Natural Science Foundation No. LY21F020013, the National Natural Science Foundation of China No. 62172366. The work is also supported by the Royal Society grant IEC/NSFC/211159.

References

[1] S. Schwarz, M. Preda, V. Baroncini, et al., Emerging mpeg standards for point cloud compression, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9 (2018) 133–148.

[2] C. Sun, H. Yuan, S. Jiang, D. Ai, W. Zhang, R. Hamzaoui, Lpcm: Learning-based predictive coding for lidar point cloud compression, 2025. URL: https://arxiv.org/abs/2505.20059.

[3] L. Huang, S. Wang, K. Wong, et al., Octsqueeze: Octree-structured entropy model for lidar compression, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 1313–1323.

[4] Z. Chen, Z. Qian, S. Wang, et al., Point cloud compression with sibling context and surface priors, in: European Conference on Computer Vision, Springer Nature Switzerland, 2022, pp. 744–759.

[5] C. Fu, G. Li, R. Song, et al., Octattention: Octree-based large-scale contexts model for point cloud compression, in: Proceedings of the AAAI conference on artificial intelligence, volume 36, 2022, pp. 625–633.

[6] R. Song, C. Fu, S. Liu, et al., Efficient hierarchical entropy model for learned point cloud compression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 14368–14377.

[7] W. Huang, Q. Yang, S. Xia, H. Huang, Z. Li, Y. Xu, Linr-pcgc: Lossless implicit neural representations for point cloud geometry compression, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2025, pp. 28577–28586.

[8] M. Cui, Y. Zhong, M. Feng, J. Long, Y. Ling, J. Xu, K. Huang, Gaem: Graph-driven attention-based entropy model for lidar point cloud compression, IEEE Transactions on Circuits and Systems for Video Technology 35 (2025) 9105–9118.

[9] Z. Que, G. Lu, D. Xu, Voxelcontext-net: An octree based framework for point cloud compression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 6042–6051.

[10] D. T. Nguyen, M. Quach, G. Valenzise, P. Duhamel, Multiscale deep context modeling for lossless point cloud geometry compression, in: 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2021, pp. 1–6.

[11] C. Sun, H. Yuan, S. Li, X. Lu, R. Hamzaoui, Enhancing context models for point cloud geometry compression with context feature residuals and multi-loss, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 14 (2024) 224–234.

[12] C. R. Qi, L. Yi, H. Su, et al., Pointnet++: Deep hierarchical feature learning on point sets in a metric space, Advances in Neural Information Processing Systems 30 (2017).

[13] A. Floris, L. Frittoli, D. Carrera, G. Boracchi, A flexible operator for deep learning on 3d point clouds, Pattern Recognition 153 (2024) 110557.

[14] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, Y. Wang, Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 5718–5727.

[15] A. B. Koyuncu, P. Jia, A. Boev, E. Alshina, E. Steinbach, Efficient contextformer: Spatio-channel window attention for fast context modeling in learned image compression, IEEE Transactions on Circuits and Systems for Video Technology 34 (2024) 7498–7511.

[16] J. Behley, M. Garbade, A. Milioto, et al., Semantickitti: A dataset for semantic scene understanding of lidar sequences, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 9297–9307.

[17] G. Pandey, J. R. McBride, R. M. Eustice, Ford campus vision and lidar data set, The International Journal of Robotics Research 30 (2011) 1543–1552.

[18] C. Loop, Q. Cai, S. O. Escolano, et al., Microsoft voxelized upper bodies-a voxelized point cloud dataset, ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673 M 72012 (2016).

[19] E. d'Eon, B. Harrison, T. Myers, et al., 8i voxelized full bodies-a voxelized point cloud dataset, ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006 7 (2017) 11.

[20] M. Group, Mpeg g-pcc tmc13, 2021. https://github.com/MPEGGroup/mpeg-pcc-tmc13.

[21] J. Wang, D. Ding, Z. Li, et al., Sparse tensor-based multiscale representation for point cloud geometry compression, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (2022) 9055–9071.

[22] M. Yim, J. C. Chiang, Mamba-pcgc: Mamba-based point cloud geometry compression, in: 2024 IEEE International Conference on Image Processing (ICIP), IEEE, 2024, pp. 3554–3560.

[23] J. Wang, R. Xue, J. Li, et al., A versatile point cloud compressor using universal multiscale conditional coding–part i: Geometry, IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

[24] C. R. Qi, H. Su, K. Mo, et al., Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[25] Y. Jin, Z. Zhu, T. Xu, Y. Lin, Y. Wang, Ecm-opcc: Efficient context model for octree-based point cloud compression, in: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 7985–7989.

[26] Q. Zhang, Q. Yuan, M. Song, H. Yu, L. Zhang, Cooperated spectral low-rankness prior and deep spatial prior for hsi unsupervised denoising, IEEE Transactions on Image Processing 31 (2022) 6356–6368.

[27] Q. Zhang, Y. Zheng, Q. Yuan, M. Song, H. Yu, Y. Xiao, Hyperspectral image denoising: From model-driven, data-driven, to model-data-driven, IEEE Transactions on Neural Networks and Learning Systems 35 (2023) 13143–13163.

[28] Q. Zhang, Y. Dong, Y. Zheng, H. Yu, M. Song, L. Zhang, Q. Yuan, Three-dimension spatial–spectral attention transformer for hyperspectral image denoising, IEEE Transactions on Geoscience and Remote Sensing 62 (2024) 1–13.

[29] A. Gu, T. Dao, S. Ermon, et al., Hippo: Recurrent memory with optimal polynomial projections, Advances in Neural Information Processing Systems 33 (2020) 1474–1487.

[30] R. E. Kalman, A new approach to linear filtering and prediction problems, Journal of Basic Engineering 82 (1960) 35–45.

[31] A. Gu, K. Goel, C. Ré, Efficiently modeling long sequences with structured state spaces (2021). URL: https://arxiv.org/abs/2111.00396.

[32] A. Gu, T. Dao, Mamba: Linear-time sequence modeling with selective state spaces (2024). URL: https://openreview.net/forum?id=tEYskw1VY2.

[33] A. Vaswani, Attention is all you need, Advances in Neural Information Processing Systems (2017).

[34] X. Liu, C. Zhang, F. Huang, S. Xia, G. Wang, L. Zhang, Vision mamba: A comprehensive survey and taxonomy, IEEE Transactions on Neural Networks and Learning Systems (2025) 1–21.

[35] A. Behrouz, F. Hashemi, Graph mamba: Towards learning on graphs with state space models, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 119–130.

[36] H. Li, Y. Hou, X. Xing, Y. Ma, X. Sun, Y. Zhang, Occmamba: Semantic occupancy prediction with state space models, in: Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 11949–11959.

[37] T. Zhang, H. Yuan, L. Qi, J. Zhang, Q. Zhou, S. Ji, S. Yan, X. Li, Point cloud mamba: Point cloud learning via state space model, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, 2025, pp. 10121–10130.

[38] X. Han, Y. Tang, Z. Wang, X. Li, Mamba3d: Enhancing local features for 3d point cloud analysis via state space model (2024) 4995–5004.

[39] D. Liang, X. Zhou, W. Xu, X. Zhu, Z. Zou, X. Ye, X. Tan, X. Bai, Pointmamba: A simple state space model for point cloud analysis, Advances in neural information processing systems 37 (2024) 32653–32677.

[40] D. Meagher, Geometric modeling using octree encoding, Computer graphics and image processing 19 (1982) 129–147.

[41] P. S. Wang, Octformer: Octree-based transformers for 3d point clouds, ACM Transactions on Graphics (TOG) 42 (2023) 1–11.

[42] Y. Wang, Y. Sun, Z. Liu, et al., Dynamic graph cnn for learning on point clouds, ACM Transactions on Graphics (TOG) 38 (2019) 1–12.

[43] A. Luo, L. Song, K. Nonaka, K. Unno, H. Sun, M. Goto, J. Katto, Scp: Spherical-coordinate-based learned point cloud compression, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 3954–3962.

[44] I. JTC1/SC29/WG7, MPEG 3D Graphics Coding. Preliminary dataset for AI-based point cloud experiments, Technical Report W21570, ISO/IEC, 2022.

[45] D. T. Nguyen, M. Quach, G. Valenzise, P. Duhamel, Learning-based lossless compression of 3d point cloud geometry, in: 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 4220–4224.

[46] M. D. G. Coding, Common test conditions for g-pcc, 2021. ISO/IEC JTC1/SC29/WG7 N00106.

[47] S. Biswas, J. Liu, K. Wong, et al., Muscle: Multi sweep compression of lidar using deep entropy models, Advances in Neural Information Processing Systems 33 (2020) 22170–22181.

[48] I. JTC1/SC29/WG7, MPEG 3D Graphics Coding. Common test conditions for G-PCC, Technical Report N00106, ISO/IEC, 2021.

[49] G. Bjøntegaard, Calculation of Average PSNR Differences Between RD-Curves, VCEG Document VCEG-M33, ITU-T Video Coding Experts Group (VCEG), 2001.