NPPC: Neural Point-Process Codec for Exchangeable and Parallel Point Cloud Geometry Compression

Yihang Ji[1], Shiqiang Long[2], Weimin Zhang[2], Ge Li[1*]
[1]*Guangdong Provincial Key Laboratory of Ultra High Definition Immersive Media Technology, School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University*
[2]*Guangdong Bohua UHD Innovation Center Co., Ltd.*
jiyihang@stu.pku.edu.cn, longsq@bohuauhd.com, zhangwm@bohuauhd.com, geli@ece.pku.edu.cn

***Abstract*—Point clouds are fundamental 3D representations for robotics, AR/VR, and autonomous driving, yet most learning-based compression methods impose artificial orderings to enable sequential entropy models. This creates a critical bottleneck: decoding must proceed sequentially, limiting throughput on modern parallel hardware and hindering real-time applications. We propose Neural Point-Process Codec (NPPC), which models point clouds as spatial point processes with an exchangeable factorization that respects their inherent unorderedness. The key insight is that by decomposing geometry into conditionally independent cell-wise components, we eliminate the need for any point ordering while enabling natural parallel decoding without complex synchronization. This exchangeable design provides both a principled probabilistic interpretation and practical parallelism. Experiments on ModelNet40 and ScanNet demonstrate that NPPC achieves competitive rate-distortion performance while enabling substantial decoding speedup, making it practical for latency-critical applications.**

***Index Terms*—point cloud compression, point processes, exchangeable models, parallel decoding**

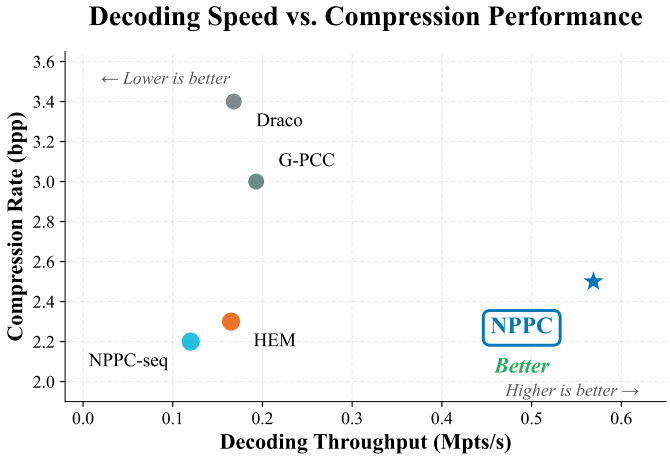

Fig. 1. NPPC achieves superior decoding throughput while maintaining competitive compression performance. By modeling point clouds as exchangeable spatial point processes, NPPC enables cell-wise parallel decoding with $\times 2.89$ speedup (single-thread to 8-thread) and only 7–14% rate overhead compared to its sequential variant NPPC-seq. Evaluated on ModelNet40 [2] at medium-rate operating points.

I. Introduction

Point clouds are a fundamental 3D representation for robotics, AR/VR, and autonomous driving [1], [2]. Unlike images or videos, a point cloud is an *unordered set* of samples in $\mathbb{R}^3$—the same set of 3D points can be permuted arbitrarily without changing its semantic or geometric meaning. However, most learning-based point cloud compression (PCC) pipelines still rely on imposing an artificial order (e.g., octree traversal, Morton/Hilbert scanning) to enable sequential entropy models [3], [4]. This design mismatch creates a recurring tension: stronger context models often increase serial dependencies and decoding latency, while parallel-friendly models may leave coding redundancies unexploited.

The standardization of point cloud compression has gained momentum with MPEG G-PCC [5], [6] and Google Draco [7], which provide robust octree-based geometry coding. Meanwhile, learning-based approaches have demonstrated superior rate–distortion (RD) performance by leveraging data-driven priors [8], [9]. These neural codecs typically adopt hierarchical latent representations paired with autoregressive (AR) entropy models inspired by image compression [10], [11]. However, the AR dependency chain creates a fundamental bottleneck: decoding must proceed sequentially, limiting throughput on modern parallel hardware. For real-time applications such as autonomous driving (where LiDAR point clouds arrive at 10-30 Hz) and telepresence (requiring low-latency streaming), this sequential bottleneck is a critical limitation.

Recent work has highlighted the importance of reducing decoding latency and improving parallelism in neural PCC. Hierarchical entropy models with grouped contexts [12] and channel-wise AR structures [13] trade some compression efficiency for faster decoding. Recent advances explicitly target extremely low decoding latency for large-scale scenes [14] and real-time LiDAR compression at 10Hz sampling rates [15], demonstrating that *parallel decodability* and throughput are becoming as important as classical RD metrics. Yet, the dom-

*Ge Li is the corresponding author. This work is supported by the Natural Science Foundation of China (62531022), Guangdong Provincial Key Laboratory of Ultra High Definition Immersive Media Technology (Grant No. 2024B1212010006), and Outstanding Talents Training Fund in Shenzhen.

inant paradigm still treats point clouds as ordered sequences of symbols, rather than unordered spatial point configurations.

A key insight motivates our approach: point clouds are fundamentally *exchangeable* data structures. The theory of exchangeable sequences and spatial point processes [16] provides a principled framework for modeling unordered collections. Recent work on entropy coding of multisets [17], [18] has shown that standard sequence-based coding is suboptimal for unordered data—explicitly encoding permutation information wastes bits. By designing a codec that respects exchangeability from the ground up, we can simultaneously achieve order invariance and unlock cell-wise parallel decoding without complex synchronization mechanisms.

We propose **Neural Point-Process Codec (NPPC)**, a geometry codec that models point clouds as *spatial point processes* over a hierarchical partition of space. The key innovation is an *exchangeable factorization* where each spatial cell can be decoded independently given coarse features, eliminating the need for any point ordering while enabling natural parallel decoding. This design provides: (1) order invariance, (2) cell-wise parallel entropy decoding, and (3) a principled probabilistic interpretation through learned point-process distributions.

We evaluate NPPC on ModelNet40 [2] and ScanNet [1], demonstrating competitive rate-distortion performance with state-of-the-art neural codecs while achieving substantial decoding speedup through multi-threaded parallel processing.

II. Related Work

A. Conventional Point Cloud Compression

MPEG G-PCC [3], [5] is the standardized geometry codec, employing octree-based spatial partitioning with context-adaptive binary arithmetic coding (CABAC) for occupancy symbols. The TMC13 reference implementation [6] uses predictive coding and geometry lifting to exploit local correlations. Google Draco [7] adopts KD-tree reordering with delta encoding and range arithmetic coding, targeting web-based 3D graphics. While these methods provide robust baselines with deterministic bitstreams, they rely on hand-crafted predictors and cannot adapt to complex data distributions. Their octree traversal imposes a fixed Morton or Hilbert ordering, which is arbitrary for inherently unordered point sets.

B. Learning-Based Point Cloud Compression

Neural point cloud codecs leverage learned representations to achieve superior rate-distortion performance. Early work [8] applied convolutional transforms to voxelized occupancy grids, followed by learned entropy models. Subsequent methods [9] adopted sparse convolutional networks to handle irregular point distributions more efficiently. These approaches typically pair an analysis/synthesis transform (encoder/decoder) with hierarchical entropy models inspired by image compression [10], [11], including hyperpriors for side information and autoregressive (AR) context models for symbol prediction. However, AR models introduce sequential dependencies: each symbol must be decoded conditioned on all previously decoded symbols, creating a fundamental throughput bottleneck.

Recent advances have begun prioritizing decoding latency as a first-class metric. Pointsoup [14] targets extremely low decoding latency for large-scale sparse scenes through efficient parallel inference of quantized feature distributions. RENO [15] addresses real-time LiDAR compression by eliminating octree construction overhead and adopting one-shot occupancy code inference with multiscale sparse tensors, achieving 10Hz real-time decoding. TopNet [19] enhances octree-based occupancy prediction with Transformer-efficient architectures and window attention mechanisms, improving compression performance while maintaining parameter efficiency. CRCIR [20] reduces coding latency through KNN-based local context modeling combined with INR-based refinement, reporting two orders of magnitude reduction in model complexity. These works demonstrate the growing emphasis on latency-throughput tradeoffs beyond pure rate-distortion optimization.

C. Parallel Decoding in Neural Compression

Recent work has explored parallelism in neural entropy models. Minnen et al. [13] proposed channel-wise autoregressive models that decode spatial locations in parallel but retain sequential dependencies across channels. Song et al. [12] introduced hierarchical entropy models with grouped contexts for point cloud compression, enabling limited parallelism by decoding groups of points simultaneously. However, these methods still impose artificial orderings (e.g., raster scan, octree traversal) to define context neighborhoods, and their parallelism is constrained by group boundaries. NPPC differs fundamentally: by modeling point clouds as spatial point processes with exchangeable factorization, we eliminate the need for any point ordering and enable cell-wise parallel decoding without group synchronization overhead.

D. Point Processes and Unordered Data

Outside compression, point processes [16] provide a principled framework for modeling unordered sets. Recent theoretical work on entropy coding of multisets [17] has shown that standard sequence-based arithmetic coding is suboptimal for unordered data: encoding a sorted sequence implicitly wastes bits on permutation information. Practical Shuffle Coding [21] advances this line by providing efficient one-shot implementations that remove order redundancy for lossless compression of unordered objects. While these methods achieve theoretical optimality for lossless coding, their engineering complexity and focus on discrete objects differ from our lossy geometry compression setting. NPPC adopts a simpler approach: by factorizing the codec into exchangeable components (occupancy, count, location) and using canonical sorting within cells, we avoid explicit permutation codes while maintaining practical efficiency and enabling parallel decoding.

III. Method

NPPC compresses geometry with an exchangeable factorization that makes finest-level cells conditionally independent given coarse features. We describe the spatial representation,

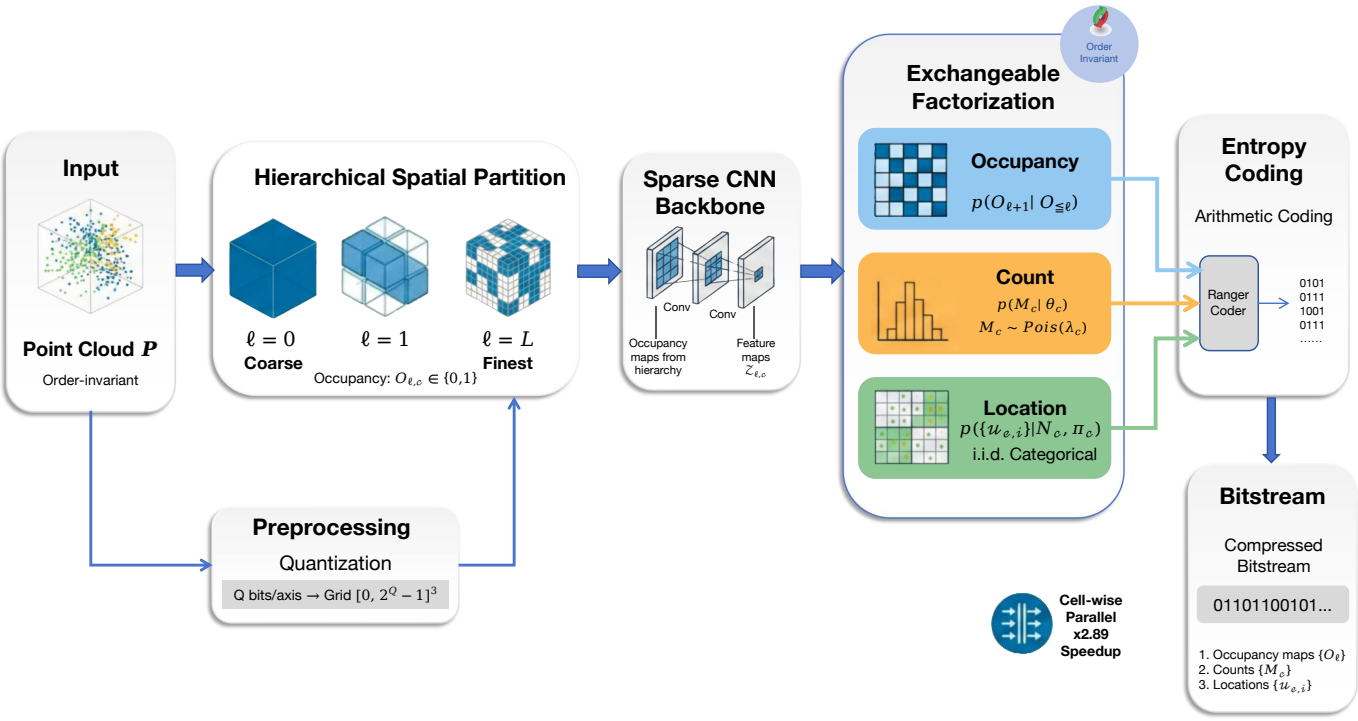

Fig. 2. Overview of NPPC. Input points are quantized and hierarchically partitioned, then processed by a sparse 3D CNN to predict occupancy, count, and intra-cell location distributions. Given coarse features, finest-level cells are conditionally independent, enabling cell-wise parallel arithmetic decoding without point ordering.

probabilistic model, network parameterization, and parallel decoding.

A. Spatial Representation and Problem Setup

Let a point cloud be an unordered set

$$\mathcal{P} = \{\mathbf{x}_i\}_{i=1}^{N}, \quad \mathbf{x}_i \in \mathbb{R}^3. \tag{1}$$

We normalize coordinates to a bounded volume (e.g., unit cube) and quantize them to an integer grid with Q bits per axis:

$$\hat{\mathbf{x}}_i \in \{0, 1, \ldots, 2^Q - 1\}^3. \tag{2}$$

This quantization bounds the coordinate space for tractable probability modeling and controls the rate-distortion tradeoff: higher Q preserves finer details at increased bitrate. NPPC encodes geometry only; attributes can be appended with a separate codec.

a) Hierarchical spatial partition: We define a grid hierarchy over the quantized space. Level ℓ has cell size s_ℓ and cell set $\mathcal{C}_\ell$, with each parent subdivided into 2^3 children from coarse to fine. Coarse levels capture global structure, while fine levels refine local detail.

Define a binary occupancy variable for each cell:

$$o_{\ell,c} \in \{0, 1\}, \tag{3}$$

where $o_{\ell,c} = 1$ iff cell (ℓ, c) contains at least one point. Let $\mathcal{O}_\ell = \{o_{\ell,c}\}_{c \in \mathcal{C}_\ell}$ denote the occupancy map. Encoding proceeds coarse to fine by recursively refining occupied regions.

At level L, each occupied cell is further discretized into a K^3 micro-grid, and each point is represented by a micro-bin index $u \in \{1, \ldots, K^3\}$. This coarse-cell plus micro-bin representation balances precision and efficiency.

B. Exchangeable Point-Process Factorization

NPPC models geometry as a product of *cell-wise point processes*. This removes point-order coding and enables parallel decoding. The joint probability of geometry $\mathcal{P}$ is factorized as:

$$\begin{aligned} p(\mathcal{P}) = &\prod_{\ell} p(\mathcal{O}_{\ell+1} \mid \mathcal{O}_{\leq \ell}) \\ &\cdot \prod_{c} p(M_c \mid \theta_c) \cdot \prod_{c} p(\{u_{c,i}\} \mid N_c, \phi_c), \end{aligned} \tag{4}$$

where the three terms correspond to occupancy, count, and location. Here $M_c = N_c - 1$ is the shifted count, θ_c parameterizes the count distribution, and ϕ_c parameterizes the micro-bin categorical π_c. The product over cells reflects conditional independence given coarse occupancy.

a) Counts as a point-process component: In occupied cells, we model the shifted count $M_c = N_c - 1 \geq 0$ as:

$$M_c \sim \text{Pois}(\lambda_c) \text{ or } \text{NB}(r_c, p_c), \tag{5}$$

where parameters (λ_c) or (r_c, p_c) are predicted from coarse features $\mathbf{z}_{L,c}$. The shift avoids zero-truncated distributions and simplifies coding.

b) Intra-cell locations as exchangeable samples: Conditioned on N_c and cell parameters, we model micro-bin indices as exchangeable draws:

$$u_{c,i} \overset{\text{i.i.d.}}{\sim} \text{Categorical}(\pi_c), \quad \pi_c \in \Delta^{K^3-1}. \tag{6}$$

This defines a discretized inhomogeneous point process within each cell. We encode the multiset in a canonical order to avoid transmitting permutations.

c) Handling multiple points in the same micro-bin: The i.i.d. model (6) permits multiple points per micro-bin. We mitigate this by training on deduplicated data and, if needed, adding a micro-occupancy bitmap. Empirically, collision rates are low (¡0.3% on ModelNet40, ¡0.8% on ScanNet; Sec. IV-F).

C. Neural Architecture and Training

a) Sparse CNN backbone and probability prediction: A sparse 3D CNN extracts features $\{\mathbf{z}_{\ell,c}\}$ for occupied cells across levels. We use a U-Net-style backbone with residual blocks and 32–128 channels.

For each parent cell at level ℓ, the network predicts Bernoulli occupancy probabilities for its child cells at level $\ell+1$:

$$p(o_{\ell+1,c'} = 1 \mid \mathcal{O}_{\le \ell}) = \sigma(g_{\text{occ}}(\mathbf{z}_{\ell,\text{parent}(c')})), \tag{7}$$

where g_{occ} is a small MLP and σ is the sigmoid function.

For occupied finest-level cells, we predict count parameters $\theta_c = g_{\text{cnt}}(\mathbf{z}_{L,c})$ and micro-bin logits $\pi_c = \text{softmax}(g_{\text{loc}}(\mathbf{z}_{L,c}))$ over K^3 bins. These outputs are deterministic functions of sparse features, enabling fast parallel inference.

b) Entropy coding and bitstream structure: The bitstream contains (1) hierarchical occupancy maps, (2) per-cell shifted counts, and (3) canonical micro-bin indices. Arithmetic coding uses the predicted probabilities; optional spatial tiling forms independent substreams for parallel decoding.

c) Rate–distortion training objective: We train to minimize bitrate via maximum-likelihood:

$$\mathcal{L} = \mathbb{E}\left[R_{\text{approx}}(\mathcal{P}; \boldsymbol{\theta})\right], \tag{8}$$

where R is approximated by cross-entropy:

$$R \approx \sum_{\ell,c} H(o_{\ell,c}) + \sum_{c} \left(H(M_c) + \sum_{i} H(u_{c,i}) \right). \tag{9}$$

We use teacher forcing for occupancy, Adam (10^{-4}, batch 16), and train for 200 epochs on ModelNet40 and 100 on ScanNet. RD points are produced by varying $L \in \{3,4,5\}$ or $Q \in \{8,10,12\}$.

D. Parallel Decoding

After coarse occupancy is decoded, each finest-level cell is independent in (4), so counts and locations can be decoded in parallel.

a) Implementation and speedup analysis: We realize this with multi-threaded arithmetic decoding over byte-aligned tile substreams. The encoder partitions the finest grid into T tiles and encodes each tile's counts and locations after the shared occupancy maps; the decoder assigns one worker per tile. This incurs a 7–14% rate penalty because tile contexts do not adapt across boundaries, but yields $\times 2.89$ speedup on 8 threads.

IV. Experiments

A. Experimental Setup

a) Datasets: We evaluate on ModelNet40 [2] (2,468 test models, 1024 points each) and ScanNet [1] (8 indoor scenes, 100K–500K points). All geometry is quantized to 10-bit precision.

b) Baselines: We compare against MPEG G-PCC [3], [6], Google Draco [7], HEM [12], and NPPC-seq, our sequential autoregressive variant.

c) Metrics: Rate is measured in bits per point (bpp), distortion by Chamfer Distance (CD, $\times 10^{-4}$) and MPEG D1/D2 PSNR, and runtime by encoding/decoding latency and throughput. Timing uses an Intel Xeon E5-2690 v4 with single-core encoding and 1/8-thread decoding.

B. Rate–Distortion Results

a) Operating points: All results use $Q = 10$ bits per axis. Low/medium/high-rate settings use $L = 3/4/5$, $s_L = 128/64/32$, and $K = 2/4/4$.

TABLE I
RATE-DISTORTION ON MODELNET40 [2] (1024 PTS, $Q=10$). RATE IN BPP; CD = CHAMFER ($\times 10^{-4}$).

Method	Low		Med		High	
	Rate	CD	Rate	CD	Rate	CD
G-PCC	1.2	18.5	3.0	4.2	8.3	0.8
Draco	1.5	22.1	3.4	5.8	9.0	1.1
HEM	0.9	12.3	2.3	2.1	6.6	0.35
NPPC (ours)	0.95	11.8	2.5	2.3	6.9	0.42
NPPC-seq	**0.88**	**10.1**	**2.2**	**1.9**	**6.3**	**0.31**

1) ModelNet40:

TABLE II
RATE-DISTORTION ON SCANNET [1] (8 SCENES, 100K–500K PTS). RATE WEIGHTED BY POINT COUNT; D1 IN DB.

Method	Moderate		High	
	Rate	D1	Rate	D1
G-PCC	5.8	28.2	12.5	35.1
HEM	4.1	32.5	9.8	38.2
NPPC (ours)	4.3	31.8	10.2	37.5
NPPC-seq	**3.9**	**33.2**	**9.3**	**39.1**

2) ScanNet (real-world RGB-D indoor scenes):

3) Analysis: NPPC incurs 7–14% rate overhead vs. NPPC-seq (e.g., 2.5 vs. 2.2 bpp at medium rate), while its 8-thread decoder achieves $\times 2.89$ speedup over single-thread NPPC. On ScanNet, NPPC remains competitive with HEM while retaining parallel decoding.

C. Timing Analysis

Timing measured on 1K/100K/1M point clouds with 8-thread parallelism.

a) Key observations: NPPC achieves $\times 2.89$ speedup (5.2→1.8 ms on 1K pts), $\times 3.4$ faster than HEM and $\times 2.9$ faster than G-PCC. Throughput stabilizes at 5.6 Mpts/s for clouds $\geq$100K pts, supporting real-time LiDAR decoding (170K pts/frame at 30 fps).

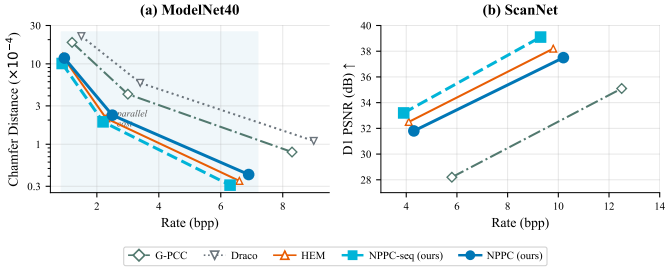

Fig. 3. Rate–distortion comparison on ModelNet40 and ScanNet. **(a)** NPPC trades 7–14% rate for parallel decoding relative to NPPC-seq. **(b)** On ScanNet, NPPC remains competitive while retaining parallelism unavailable to G-PCC and Draco.

TABLE III
TIMING ON 1024-POINT CLOUDS (MODELNET40). THROUGHPUT IN KPTS/S.

Method	Enc	Dec-Seq	Dec-Par	Speedup	Thr-Par
G-PCC	2.1	5.3	—	—	—
Draco	2.8	6.1	—	—	—
HEM	**3.2**	**8.7**	6.2	×1.4	165.1
NPPC-seq	3.1	8.5	—	—	—
NPPC (ours)	3.0	5.2	**1.8**	×**2.9**	**568.9**

Time in ms. G-PCC/Draco/NPPC-seq: no parallel decoding.

D. Tile-Parallel Decoding: Implementation and Overhead

NPPC partitions the finest-level grid into spatial tiles and encodes each tile as an independent substream with byte offsets for random access. Overhead from context fragmentation is small for large clouds (¡1%) but significant for small clouds (8% at 4^3 tiling), so adaptive tiling is preferred.

Small clouds (1K pts): 4^3 tiling achieves ∼8% overhead with ×2.9 speedup. Large clouds (200K pts): overhead ¡1% for all tilings; 2^3 tiling achieves ×4.0 speedup with 0.6% overhead. Adaptive tiling recommended.

E. Ablation Study

All variants trained at ∼2.5 bpp (medium rate) on ModelNet40.

TABLE IV
SCALING TEST: DECODE LATENCY FOR POINT CLOUDS OF VARYING SIZES. NPPC WITH 8-THREAD PARALLEL DECODING.

Point Count	1K	100K	500K	1M
NPPC Decode-Par (ms)	1.8	18.2	89.5	176.3
Throughput (Mpts/s)	0.569	5.49	5.59	**5.67**

TABLE V
TILE-PARALLEL OVERHEAD ON SMALL (1K PTS) VS. LARGE (200K PTS) CLOUDS. RATE IS PAYLOAD-ONLY BPP; OVERHEAD FROM MODEL FRAGMENTATION.

		Small (1K)		Large (200K)	
Tiling	#Tiles	Ovhd%	Speedup	Ovhd%	Speedup
Single	1	**0.3%**	×1.0	**0.0%**	×1.0
8^3	8	+1.5%	×1.7	+0.02%	×1.7
4^3	64	+8.2%	×2.9	+0.08%	×2.9
2^3	512	+64%	×**4.0**	+0.6%	×**4.0**

Coarse tiling (4^3) recommended for small; fine (2^3) for large clouds.

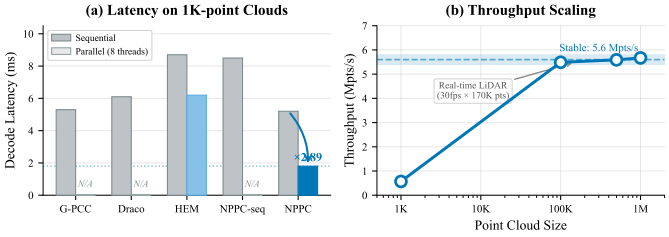

Fig. 4. Parallel decoding analysis. **(a)** On 1K-point clouds, NPPC reduces decoding time from 5.2 to 1.8 ms with 8 threads (×2.89). **(b)** Throughput stabilizes at about 5.6 Mpts/s beyond 100K points, sufficient for real-time LiDAR.

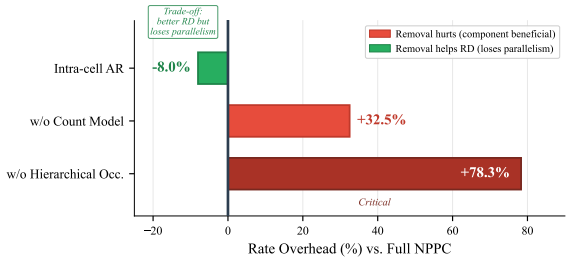

Fig. 5. Ablation on ModelNet40. Removing hierarchical occupancy or count modeling increases rate, while intra-cell AR reduces rate by 8% but sacrifices intra-cell parallelism.

Hierarchical occupancy is critical (+78% rate if removed), and the count model gives a moderate gain (+33%). Intra-cell AR improves RD slightly (CD 2.3→1.9) but slows decoding by ×1.2. NPPC-seq gives the best RD but is ×4.7 slower.

F. Micro-Bin Collision Analysis

Collision rates are low: 0.27% on ModelNet40 and 0.76% on ScanNet. This supports the i.i.d. categorical model with canonical sorting.

G. Comparison with Sequential Variant

NPPC trades 7–14% rate for ×4.7 decoding speedup vs. NPPC-seq (Fig. 3a, Table III). NPPC is preferable for real-time use, while NPPC-seq may suit archival settings.

TABLE VI
ABLATION ON MODELNET40. BASELINE: FULL NPPC AT ∼2.5 BPP.

Variant	CD	Impact
NPPC (full)	2.3	baseline
w/o hierarchical	4.1	+78% rate
w/o count model	3.7	+33% rate
Intra-cell AR	**1.9**	-8% rate, ×1.2 dec.
NPPC-seq (global AR)	**1.9**	-14% rate, ×4.7 dec.

CD = Chamfer ($\times 10^{-4}$). AR variants trade parallelism for rate.

TABLE VII
MICRO-BIN COLLISION STATISTICS.

Dataset	Collision Rate	Avg/Cell	Max/Cell
ModelNet40	**0.27%**	**0.003**	**2**
ScanNet	0.76%	0.008	4

Collision rate = #collided pts / #total pts. Avg/Max over non-empty cells.

V. Conclusion

We presented NPPC, a point-process geometry codec with exchangeable occupancy, count, and location factors. By making finest-level cells conditionally independent given coarse features, NPPC achieves order-invariant compression and parallel entropy decoding. On 1K-point clouds, 8-thread NPPC reduces its own decoding latency from 5.2 to 1.8 ms ($\times 2.89$) at 7–14% rate overhead versus NPPC-seq. Future work includes stronger count models, attribute compression [22], hardware acceleration, and dynamic sequences [23].

References

[1] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5828–5839.

[2] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[3] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[4] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "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.

[5] ISO/IEC, "ISO/IEC 23090-9: Geometry-based point cloud compression," 2023, international Standard (MPEG G-PCC).

[6] MPEG Group, "MPEG PCC TMC13 reference software," 2026, gitHub repository.

[7] Google, "Draco: 3d data compression library," 2026, gitHub repository.

[8] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4320–4324.

[9] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, 2021.

[10] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.

[11] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[12] R. Song, C. Fu, S. Liu, and G. Li, "Efficient hierarchical entropy model for learned point cloud compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 368–14 377.

[13] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3339–3343.

[14] K. You, K. Liu, L. Yu, P. Gao, and D. Ding, "Pointsoup: High-performance and extremely low-decoding-latency learned geometry codec for large-scale point cloud scenes," *arXiv preprint arXiv:2404.13550*, 2024.

[15] K. You, T. Chen, D. Ding, M. S. Asif, and Z. Ma, "RENO: Real-time neural compression for 3d LiDAR point clouds," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 22 172–22 181.

[16] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[17] D. Severo, J. Townsend, A. Khisti, A. Makhzani, and K. Ullrich, "Compressing multisets with large alphabets," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 4, pp. 605–615, 2023.

[18] J. Kunze, D. Severo, G. Zani, J.-W. van de Meent, and J. Townsend, "Entropy coding of unordered data structures," *arXiv preprint arXiv:2408.08837*, 2024.

[19] X. Wang, Y. Zhang, T. Liu, X. Liu, K. Xu, J. Wan, Y. Guo, and H. Wang, "TopNet: Transformer-efficient occupancy prediction network for octree-structured point cloud geometry compression," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 27 305–27 314.

[20] H. Xu, X. Zhang, and X. Wu, "Fast point cloud geometry compression with context-based residual coding and INR-based refinement," in *European Conference on Computer Vision*. Springer, 2024, pp. 270–288.

[21] J. Kunze, D. Severo, J.-W. van de Meent, and J. Townsend, "Practical shuffle coding," *Advances in Neural Information Processing Systems*, vol. 37, pp. 84 081–84 113, 2024.

[22] C. Peng, "Generalized gaussian entropy model for point cloud attribute compression with dynamic likelihood intervals," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 11 779–11 788.

[23] Z. Pan, M. Xiao, X. Han, D. Yu, G. Zhang, and Y. Liu, "patchDPCC: A patchwise deep compression framework for dynamic point clouds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 5, 2024, pp. 4406–4414.