Gameleon: Gaussianizing Point Clouds for On-Demand 4D Performance

ZEHONG LI, Hangzhou Normal University, China
JINYANG LI, Nanjing University, China
YUEYU HU, New York University, USA
DANDAN DING*, Hangzhou Normal University, China
ZHAN MA, Nanjing University, China

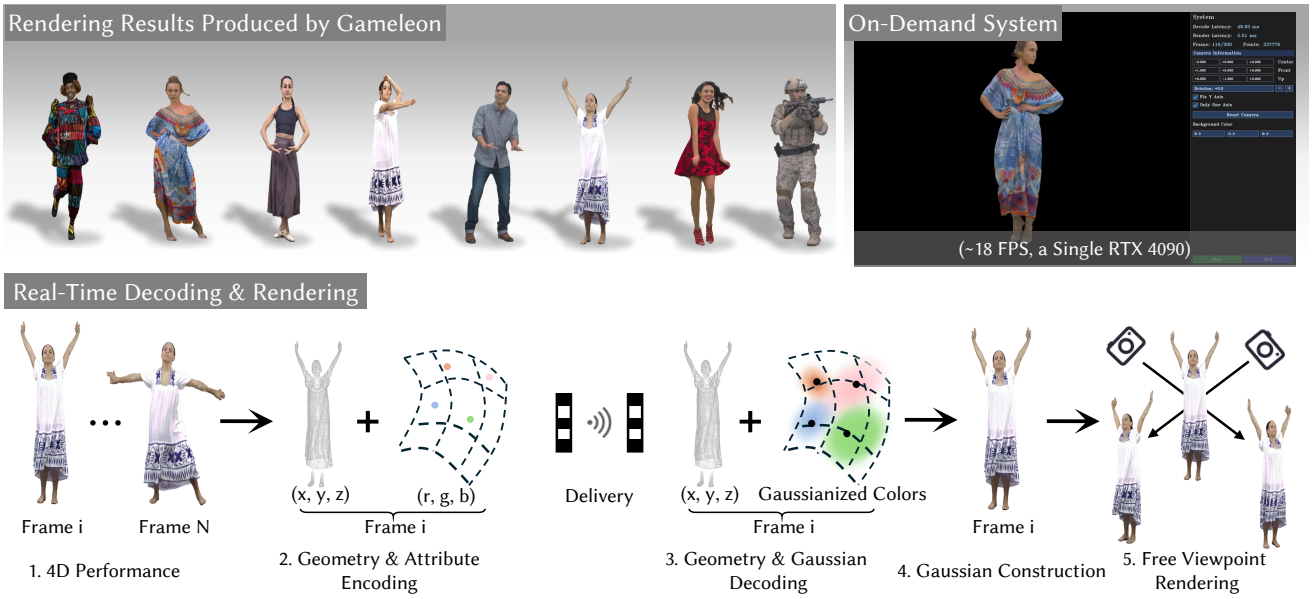

Fig. 1. **Gameleon System Overview.** Gameleon directly decodes compressed 3D data into renderable Gaussian primitives for on-demand playback. Top: representative rendering results and the live decoding and rendering interface. Bottom: the end-to-end pipeline from encoding to decoding and rendering.

The rapid growth of 3D content poses substantial challenges for on-demand streaming systems, where both real-time decoding and rendering are required. Existing works typically compress 3D data represented as point clouds, reconstruct them at the decoder to preserve point-wise fidelity, and then render them using external tools. This separated pipeline introduces considerable latency and hinders real-time performance. To address this issue, we propose an efficient end-to-end system, termed Gameleon, which reorganizes 3D data into geometry and color attributes and compresses them into rendering features. These features are directly decoded into Gaussian primitives for rendering, eliminating explicit point reconstruction and substantially reducing system-level overhead. Specifically, we develop low-complexity coders: lossless geometry coder (LGC) and Gaussianized attribute coder (GAC), built on an efficient multiscale coding framework with rendering-driven optimization. The system therefore achieves a balanced tradeoff among speed, rendering quality, and rate. Extensive experiments on diverse datasets demonstrate that Gameleon outperforms existing solutions in both compression efficiency and processing speed. Our prototype achieves 18.5 FPS for the full decoding-to-rendering pipeline on a single NVIDIA RTX 4090 GPU, while a lightweight variant reaches 25.6 FPS, satisfying the requirements of on-demand 4D streaming. Demo and code are available on the project website https://gameleon2026.github.io/anon-demo/.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Point Cloud, 3D Gaussian Splatting, Real-Time Playback, Volumetric Video, Compression

ACM Reference Format:
Zehong Li, Jinyang Li, Yueyu Hu, Dandan Ding, and Zhan Ma. 2026. Gameleon: Gaussianizing Point Clouds for On-Demand 4D Performance. In *SIGGRAPH Asia 2026 Conference Papers (SA Conference Papers '26), December 01–04, 2026, Kuala Lumpur, Malaysia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3829340.3842343

*Corresponding author

Authors' Contact Information: Zehong Li, Hangzhou Normal University, Hangzhou, China, 2024112011033@stu.hznu.edu.cn; Jinyang Li, Nanjing University, School of Electronic Science and Engineering, Nanjing, China, jinyangli@smail.nju.edu.cn; Yueyu Hu, New York University, New York, New York, USA, yyhu@nyu.edu; Dandan Ding, Hangzhou Normal University, Hangzhou, China, DandanDing@hznu.edu.cn; Zhan Ma, Nanjing University, Nanjing, China, mazhan@nju.edu.cn.

SA Conference Papers '26, Kuala Lumpur, Malaysia

ACM ISBN 979-8-4007-2842-6/2026/12
https://doi.org/10.1145/3829340.3842343

1 Introduction

3D data representations such as point clouds, meshes, and Gaussians have become essential digital content [Cao et al. 2021; Schwarz

et al. 2019], with broad applications in digital humans, volumetric video, immersive communication, and related areas. While these representations effectively preserve scene geometry and support free viewpoint interaction, the large volumes of geometric and appearance data place substantial pressure on storage, transmission, and real-time rendering [Cao et al. 2019; Dai et al. 2025; Quach et al. 2022; Wang et al. 2024b]. Therefore, practical 3D content delivery requires both efficient 3D data compression and high-quality rendering. In this paper, we focus on on-demand streaming scenarios that require real-time decoding and rendering of 3D data.

1.1 Motivation

Given a 3D sequence of T frames, each frame is represented as $P_t = (X_t, A_t)$. Here, $X_t = \{\mathbf{x}_{t,i}\}_{i=1}^{N_t}$ denotes the 3D coordinates, and $A_t = \{\mathbf{a}_{t,i}\}_{i=1}^{N_t}$ denotes the corresponding attributes of N_t points in frame t. The encoder produces a compact bitstream B_t, and the decoder reconstructs the frame for rendering as follows:

$$P_t = (X_t, A_t) \rightarrow B_t = (B_t^{\text{geo}}, B_t^{\text{attr}}) \rightarrow \hat{P}_t = (\hat{X}_t, \hat{A}_t) \rightarrow \hat{I}_t. \quad (1)$$

Based on this pipeline, high rendering quality for $\hat{I}_t$ is pursued by minimizing the point-wise geometry error and attribute error between the reconstructed $\hat{P}_t$ and the original P_t, as commonly adopted in point cloud compression [Wang et al. 2022, 2025a]. Such optimization improves the reconstruction fidelity of point cloud itself rather than the final rendered images $\hat{I}_t$. However, the ultimate output of 3D content transmission is the rendered images displayed on the screen. Due to the discrete and irregular structure of point clouds, low geometry or RGB reconstruction error does not necessarily lead into high rendering quality. Even with small reconstruction errors, the decoded point cloud may still produce holes, broken boundaries, and unstable appearance during rendering.

Therefore, a more principled way is to optimize rendering quality directly. Mesh provides an intermediate representation with explicit topology for rendering [Choi et al. 2022], but encoding topology increases coding complexity, while estimating it at the decoder slows rendering. The recent 3D Gaussian splatting (3DGS) [Kerbl et al. 2023, 2024; Mallick et al. 2024] offers an alternative by representing 3D content with Gaussian primitives. These primitives are projected to screen space through splatting, driving rendering loss to directly supervise image quality rather than relying on point-wise reconstruction fidelity. However, the large number of primitives introduces substantial compression overhead. Existing works based on either per-scene optimization or pre-trained models remain insufficient for real-time processing [Chen et al. 2024a, 2025b].

The observation above reveals a complementary property between point cloud and Gaussian. Point cloud is compact but difficult to render stably due to their discrete structure, whereas Gaussian primitives provide a continuous representation well suited for splatting-based rendering. This suggests a different perspective: instead of reconstructing colored point clouds as the final output, compressed point cloud data can be transformed into Gaussian primitives for rendering. Recent studies begin to explore this idea. Learned splatting converts discrete points into 3D elliptical Gaussians for high-quality, low-latency rendering [Hu et al. 2024]. However, existing methods remain far from practical deployment due to inefficient geometry and attribute coding, limited rate-distortion (R-D) performance, and unsatisfactory rendering quality. For example, B2P [Hu et al. 2025] supports only static coding and runs at about 3.75 FPS on an RTX 4090 with limited visual quality.

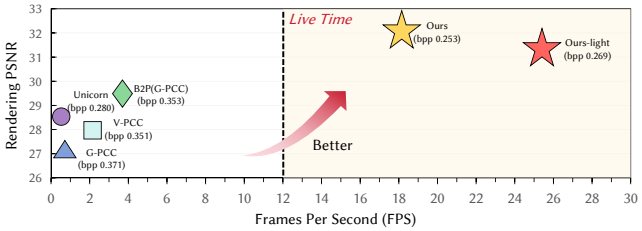

Fig. 2. **Runtime and Rendering Quality on Owlii**: screen-space PSNR vs. end-to-end decode-and-render FPS across existing solutions. The dashed line marks a 12 FPS live-interactive reference [Dong et al. 2024].

1.2 Proposed Approach

Building on this analysis, we propose Gameleon, which Gaussianizes point cloud for on-demand 4D performance (Fig. 1). Fundamentally, this formulation generalizes to diverse 3D data formats, and this paper exemplifies it on point clouds. Gameleon directly decodes compressed bitstreams into display-ready Gaussian primitives. This formulation unifies compression, representation, and rendering into a single pipeline, allowing the decoded data to be consumed by the renderer without intermediate processing.

Gameleon adapts a single model to diverse bandwidth conditions through a two-branch design. The first branch is a lossless geometry coder (LGC) that converts input coordinates into a multiscale sparse occupancy representation and reconstructs the spatial support for Gaussian formation. The second branch is a rendering-oriented Gaussianized attribute coder (GAC) based on restored geometry support. It encodes attributes into multiscale rendering features and decodes these features to Gaussian primitives for fast rendering. In deployment, LGC and GAC are pipelined in Gameleon, achieving a decode-and-render speed of 18.5 FPS. The lightweight variant, Ours-light, further improves the throughput to 25.6 FPS. Figure 2 compares the rendering PSNR and processing speed of Gameleon with existing methods.

Our contributions are summarized as follows:

- We present Gameleon, a 3D data-to-Gaussian compression system for real-time on-demand streaming. Built upon a multiscale coding framework and efficient acceleration pipeline, Gameleon directly decodes compressed 3D data into render-ready Gaussian primitives without reconstructing intermediate colored point clouds.
- We design two efficient coding modules: a lossless geometry coder (LGC) and a Gaussianized attribute coder (GAC), built on a multiscale coding framework. LGC encodes the geometry, upon which GAC generates Gaussian primitives from compressed attributes. A two-group two-stage schedule together with rendering-driven R-D optimization balances coding efficiency and decoding speed.
- Extensive experiments on static and dynamic datasets demonstrate the superiority of Gameleon in compression efficiency,

rendering quality, and runtime performance. On dynamic Owlii, Gameleon achieves 28.4% and 31.8% BPP reductions, together with 2.51 dB and 4.66 dB PSNR gains, over B2P (G-PCC) and G-PCC (Gaussian), respectively. Our prototype achieves 18.5 FPS decode-and-render throughput on an NVIDIA RTX 4090 GPU, while the lightweight variant reaches 25.6 FPS with only a 97.35 MB model.

2 Related Work

2.1 Point Cloud & Mesh Compression

Point cloud compression (PCC) generally consists of geometry part and an attribute part. MPEG has released two representative standards, V-PCC and G-PCC [VPC 2023; GPC 2023; Graziosi et al. 2020]. V-PCC projects 3D point clouds into 2D patches or image sequences and codes them by reusing mature video codecs such as H.265/HEVC and H.266/VVC [Bross et al. 2021; Sullivan et al. 2012]. G-PCC directly processes point sets in 3D space, using tools such as octree coding and trisoup surface approximation in the geometry coder, and RAHT and lifting transforms in the attribute coder.

Recent learning-based methods further improve PCC coding efficiency. Geometry coders usually organize point clouds using octree [Fu et al. 2022; Song et al. 2023; Wang et al. 2025d] or sparse tensor structures [Liu et al. 2026; Wang et al. 2022; Wang and Gao 2025; Wang et al. 2025c; You et al. 2025] for multiscale coding. At each scale, the coder estimates occupancy probabilities using spatiotemporal priors from decoded lower-scale symbols and/or neighboring symbols. Mapped to the decoded geometry, attribute components like RGB colors or reflectance are coded by directly predicting or modeling attribute values [Sheng et al. 2021; Wang and Ma 2022] or encoding residuals between true values and predictions [Umair et al. 2024; Wang et al. 2025b]. While most works attack either geometry or attribute, Yak [Zhang et al. 2025] and Unicorn [Li et al. 2025; Wang et al. 2025a] offer system-level solutions by incorporating both geometry and attribute coders for different PCC scenarios.

Despite these advances, existing PCC systems, particularly for dense point clouds used in digital humans, volumetric video, and immersive communication, mainly focus on improving coding efficiency by balancing bitrate and point-level errors, while rarely considering on-demand access, leaving real-time, high-quality interactive streaming and rendering largely unsupported.

On the other hand, minimizing point-level error does not necessarily reduce rendering error (see Fig. 5). Meshes extend point clouds by introducing explicit topology that is rendering friendly, [Chen et al. 2025a; Zou et al. 2025] but encoding the topology increases transmission cost. Some methods therefore estimate topology at the decoder, which increases decoding-and-rendering time. [Erler et al. 2024; Huang et al. 2023] Therefore, high-quality free-viewpoint display requires a coding paradigm that considers the entire capture-to-render pipeline rather than optimizing fidelity or rendering alone.

2.2 Gaussian Compression and Rendering

Gaussian representation offers a natural way to bridge this display gap. Each Gaussian primitive encodes position, opacity, scale, rotation, and appearance (59 parameters) and can be directly rendered via splatting. Recent works demonstrate high-quality real-time rendering [Dong et al. 2024; Kerbl et al. 2023], suggesting better alignment with screen-space rendering.

However, the large number of Gaussian primitives poses great challenges for transmission. For instance, a single primitive requires 1888 bits (59×32) if each element is represented with 32 bits, demanding efficient compression methods. One line of work compacts Gaussians through pruning, masking, and quantization [Fan et al. 2024; Lee et al. 2024; Liu et al. 2025] but achieves limited compression efficiency, e.g., LightGaussian [Fan et al. 2024] reduces the storage of a 3DGS scene from 727 MB to 42 MB, which is still substantial. Another compresses Gaussian geometry and attributes using entropy coding, spatial contexts, autoregressive models, or Scaffold-GS-based predictors [Wang et al. 2024a; Zhan et al. 2025]. These approaches often follow two paradigms: (i) per-scene optimization [Chen et al. 2024b, 2025b; Niedermayr et al. 2024] that overfits each scene and transmits the scene-specific model to the decoder, achieving high compression ratios but incurring long model renewal delays; and (ii) feed-forward methods [Chen et al. 2024a; Zhang et al. 2024] that rely on heavy pre-trained models, resulting in long decoding time and reduced coding efficiency (e.g., the recent FCGS requires 11-15 s to decode a frame).

As seen, despite the real-time rendering capability of Gaussian representations, their coding pipelines remain computationally intensive, making on-demand streaming and rendering difficult.

2.3 Summary

As discussed above, point clouds provide basic 3D data format through coordinates and attributes, while Gaussian representations offer rendering-friendly primitives for high-quality real-time display. This motivates rethinking the representation across compression and rendering. Recent studies show that 3D elliptical Gaussians can be predicted directly from points [Hu et al. 2024, 2025]. Building on this insight, we restructure 3D data coding toward Gaussian formation, directly decoding compressed points into Gaussian primitives and bypassing the costly reconstruction in conventional pipelines.

3 Our Method

3.1 Overview

As aforementioned, we propose a new coding-rendering framework through Gaussianizing point clouds. Specifically, given a point-cloud frame $P_t = (X_t, A_t)$, Gameleon decodes compressed geometry and attribute bitstreams directly into renderable Gaussian primitives $\mathcal{G}_t$, which are then rasterized into the output image $\hat{I}_t$. As illustrated in Fig. 3, Gameleon decomposes the input point cloud into multiscales for scale-wise coding, with each scale producing a bitstream from the support geometry coder (LGC) and a bitstream from the attribute to Gaussian coder (GAC). At each scale, spatiotemporal priors from its lower scale are leveraged for conditional coding. Specifically,

i. LGC losslessly encodes point occupancy by predicting occupancy probabilities using spatiotemporal priors, recovering the support geometry required for Gaussian formation in GAC. A two-group, two-stage schedule further improves coding efficiency.

ii. GAC converts RGB attributes to rendering features for transmission and progressively refines them along the coarse-to-fine

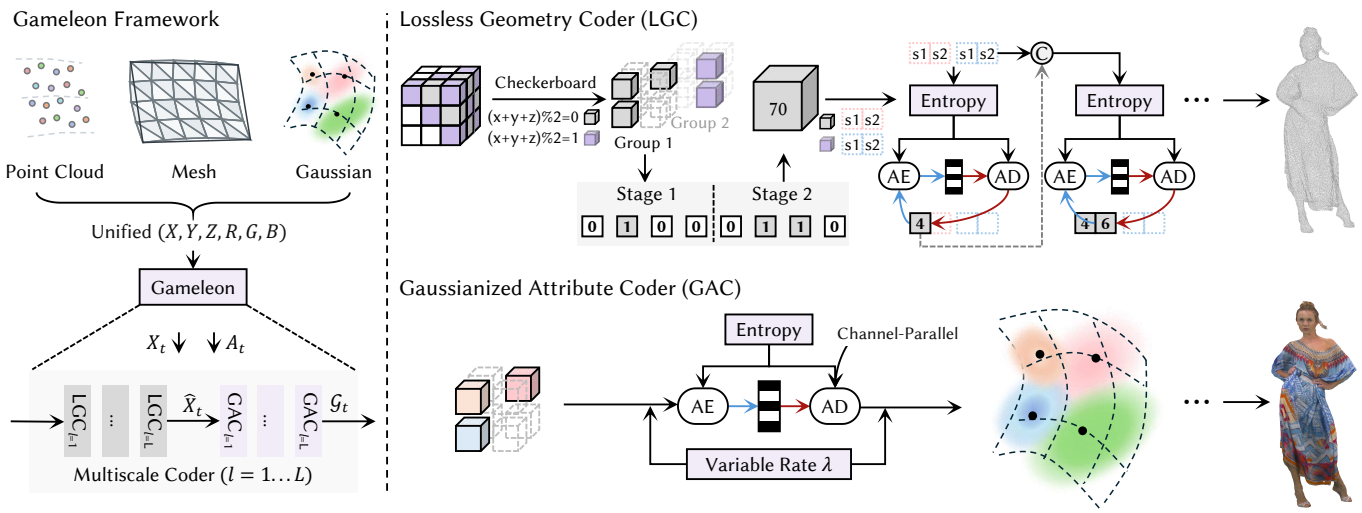

Fig. 3. **Gameleon Framework.** It contains a lossless geometry coder called LGC and a rendering-oriented attribute-to-Gaussian coder called GAC. LGC converts coordinates into multiscale occupancy symbols and recovers support geometry through entropy decoding. Upon the recovered support, GAC directly predicts Gaussian parameters for rendering. AE: arithmetic encoding; AD: arithmetic decoding. The detailed neural network structure and two-group two-stage schedule involved in this framework can be found in our Appendix.

decoded support geometry. It uses a λ-conditioned quantizer to control the quality of rendering features.

3.2 Lossless Geometry Coder (LGC)

We first voxelize X_t into a multiscale sparse occupancy hierarchy $\mathcal{O}_t = \{O_t^l\}_{l=1}^L$. At each scale, every occupied voxel is divided into eight sub-voxels at the next finer scale. The occupancy states of these eight sub-voxels form an 8-bit pattern, represented as an occupancy symbol $s_{t,i}^l \in \{0, \ldots, 255\}, i \in [1, N^l]$, where N^l denotes the number of occupied voxels in scale l. Thus, LGC is formulated as lossless entropy coding of multiscale occupancy symbols. After decoding, the occupied voxels at each scale are converted back to support coordinates $\hat{X}_t^l$.

We encode the occupancy symbols from coarse scale to fine scale. Let p_θ^g denote the learned entropy model. At scale l, p_θ^g predicts the probability distribution of current occupancy symbols conditioned on the decoded geometry context C_t^l:

$$p_\theta^g(\mathcal{O}_t) = \prod_{l=1}^{L} p_\theta^g\left(O_t^l \mid C_t^l\right), \; C_t^l = \begin{cases} \{\hat{O}_t^{<l}\}, & \text{static,} \\ \{\hat{O}_t^{<l}, O_{t-1}^l\}, & \text{dynamic.} \end{cases} \tag{2}$$

with $\hat{O}_t^{<l}$ denoting the decoded lower-scale occupancy of the current frame. O_{t-1}^l denotes the decoded occupancy context of the reference frame at the same scale in dynamic coding. After downsampling the reference geometry into the same multiscale hierarchy, we apply a convolutional extractor to O_{t-1}^l to embed reference-frame geometry features and warp them to the current geometry support through a target convolution. The same extract-and-warp strategy is employed in GAC, with decoded rendering features replacing geometry features. The probabilities predicted by p_θ^g are then used by the arithmetic engine for lossless encoding and decoding.

Within each scale, we use a two-group two-stage schedule to improve coding efficiency. First, occupied voxels at scale l are split into two groups based on the parity of their coordinates (x, y, z):

$$G_k^l = \left\{ s_{t,i}^l \in O_t^l \,\middle|\, (x_{t,i} + y_{t,i} + z_{t,i}) \bmod 2 = k - 1 \right\}, \; k \in \{1, 2\}. \tag{3}$$

The first group G_1^l is decoded conditioned on the scale context C_t^l to predict occupancy probabilities $p_\theta^g(G_1^l | C_t^l)$, as depicted in Fig. 3. The second group G_2^l further exploits the decoded G_1^l as an intra-scale causal context, yielding $p_\theta^g(G_2^l | C_t^l, G_1^l)$.

Within each group, each 8-bit occupancy symbol is decomposed into lower and upper four-bit parts for two-stage coding, with the lower bits encoded before the upper bits. As a result, the decoding order is $G_1^{\text{low}} \to G_1^{\text{high}} \to G_2^{\text{low}} \to G_2^{\text{high}}$, forming a conditional coding chain while retaining parallelism within each group.

3.3 Gaussianized Attribute Coder (GAC)

Upon decoded support geometry $\{\hat{X}_t^l\}_{l=1}^L$, GAC converts RGB attributes $\{A_t^l\}_{l=1}^L$ into rendering features $\{F_t^l\}_{l=1}^L$, quantizes them to $\{\hat{F}_t^l\}_{l=1}^L$, and encapsulates them as a bitstream. The decoder then recovers $\{\hat{F}_t^l\}_{l=1}^L$ to construct Gaussian primitives $\mathcal{G}_t$.

Let p_ϕ^a denote the learned attribute entropy model. At scale l, p_ϕ^a predicts the probability distribution of current $\hat{F}_t^l$ conditioned on the decoded attribute context $\mathcal{E}_t^l$:

$$p_\phi^a(\hat{\mathcal{F}}_t) = \prod_{l=1}^{L} p_\phi^a\left(\hat{F}_t^l \mid \mathcal{E}_t^l\right), \; \mathcal{E}_t^l = \begin{cases} \{\hat{F}_t^{<l}\}, & \text{static,} \\ \{\hat{F}_t^{<l}, F_{t-1}^l\}, & \text{dynamic,} \end{cases} \tag{4}$$

with decoded lower-scale features $\hat{F}_t^{<l}$ and reference-frame features F_{t-1}^l. F_{t-1}^l is obtained using the extract-and-warp strategy above. The probabilities predicted by p_ϕ^a are then used to encode $\hat{F}_t^l$.

To support a wide rate range with a single model, we use a λ-conditioned quantizer $\hat{F}_t^l = Q_\lambda(F_t^l)$, where λ controls the quantization strength. The decoder reconstructs and refines $\hat{F}_t^l$ scale by scale,

deriving the final rendering features $\hat{F}_t$ for Gaussian construction:

$$\mathcal{G}_t = \Phi_{\mathrm{G}}(\hat{X}_t, \hat{F}_t) = \{(\hat{\mathbf{x}}_i + \Delta\mathbf{x}_i, \alpha_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{c}_i)\}_{i=1}^{\hat{N}_t}, \tag{5}$$

where $\Delta\mathbf{x}_i$, α_i, $\mathbf{s}_i$, $\mathbf{r}_i$, and $\mathbf{c}_i$ denote Gaussian position offset, opacity, scale, rotation, and color representation, respectively. Φ_{G} maps each decoded support point in $\hat{X}_t$ and its rendering feature in $\hat{F}_t$ to the corresponding Gaussian parameters $\mathcal{G}_t$. In this way, the attribute bitstream is decoded to renderable Gaussian primitives, successfully avoiding the conventional two-step pipeline of reconstructing colored points and then applying a separate renderer.

3.4 Systematic Acceleration

We further apply several acceleration strategies to Gameleon for on-demand streaming.

Channel-parallel Entropy Decoding. In GAC, entropy context for all latent channels are available before arithmetic decoding. Let $\hat{F}$ denote the quantized attribute latent and Ψ denote the predicted entropy context. We have $p_\phi^a(\hat{F} \mid \mathcal{E}) = \prod_{c=1}^{C} \prod_i p_\phi^a(\hat{F}_{c,i} \mid \mathcal{E}_{c,i})$, where c indexes the latent channel and i indexes spatial positions. Based on this factorization, in each scale, we organize the attribute bitstream as independent channel streams and decode them in parallel when conducting arithmetic decoding. This only changes the execution layout of arithmetic decoding. The entropy model, quantized symbols, and reconstructed features remain unchanged.

Cross-frame Pipeline. For sequence playback, we adopt a two-stage pipeline between geometry decoding, attribute decoding, and rendering to reduce system idle time. Once the support geometry of frame t is decoded, its attribute decoding and rendering can proceed. Meanwhile, the geometry decoding of frame $t+1$ starts. Therefore, the per-frame latency of the serial schedule $T_{\mathrm{serial}} = T_{\mathrm{geo}} + T_{\mathrm{attr}} + T_{\mathrm{ren}}$ is reduced to $T_{\mathrm{pipe}} \approx \max\left(T_{\mathrm{geo}}, T_{\mathrm{attr}} + T_{\mathrm{ren}}\right)$, where T_{geo}, T_{attr}, and T_{ren} denote the runtimes of geometry decoding, attribute decoding, and rendering, respectively.

Compact Gaussian Decoder. To further reduce the decode-to-Gaussian latency, we develop a compact variant of Gameleon. Layer and channel sensitivity in Gameleon model is estimated on a validation set using rendering loss, followed by structured removal of low-contribution channels. This lightweight model preserves the same support-geometry input and Gaussian output interface while reducing inference cost and improving playback speed with minor impact on rendering quality (see our results in Fig. 7).

4 Experiments and Analysis

4.1 Experimental Settings

Datasets. To ensure fair comparison and broad evaluation, we evaluate Gameleon on diverse static and dynamic 3D contents, following common settings from prior works. The evaluation covers 8iVFB [d'Eon et al. 2017], Owlii [Yi et al. 2017], THuman [Yu et al. 2021], and DanceNet3D datasets. The use of 8iVFB and Owlii aligns with MPEG common test conditions (CTC) [MPEG 3D Graphics Coding and Haptics Coding 2025; WG 07 MPEG 3D Graphics Coding and Haptics Coding 2024]. The use of THuman matches the static attribute-to-Gaussian setting of B2P [Hu et al. 2025]. We use DanceNet3D following its default settings.

Training. For MPEG point cloud data, geometry model training follows the dataset split and sample preparation protocol defined in MPEG AI-PCC CTC. The static LGC is trained on RWTT [Maggiordomo et al. 2020], and the dynamic LGC is trained on 8iVFB. For the non-MPEG THuman, the static GAC is trained on THuman 2.0 to align with B2P's setting. For the non-MPEG DanceNet3D, LGC and GAC use the same data split. We use *BigKicks* and *BiancaGolden_ReleaseToFloor* as training sequences. During training, we randomly sample 12 views from 30 fixed candidate cameras. We render the reference data and decoded Gaussians from the same views to compute the rendering loss. The detailed camera configuration is provided in the Appendix.

Testing. The static test set includes 8iVFB, Owlii, THuman, and DanceNet3D. The dynamic test set includes Owlii and DanceNet3D. In addition, to examine the generalizability of Gameleon, we test it on five 12-bit non-human samples recommended by MPEG CTC, including two dense samples (*facade*, *house*) and three sparse samples (*arco*, *shiva*, *statue*), as well as a real urban scene from UrbanScene3D [Lin et al. 2022].

Settings. Models are trained on an NVIDIA H20 GPU, and all reported runtimes are measured on a single NVIDIA RTX 4090 GPU.

LGC is trained to predict the conditional probabilities of occupancy symbols, using the loss:

$$\mathcal{L}_{\mathrm{geo}} = -\sum_{l=1}^{L} \frac{1}{|O^l|} \sum_{s_i^l \in O^l} \log_2 p_\theta^g\left(s_i^l \mid C_i^l\right), \tag{6}$$

where s_i^l denotes an 8-bit occupancy symbol at scale l. Its lower four bits and upper four bits are entropy-coded in two stages. The geometry context C_i^l is defined in Eq. (2).

GAC is trained using a screen-space rate-distortion (R-D) loss:

$$\mathcal{L}_{\mathrm{attr}}(\lambda) = \mathcal{D}_{\mathrm{render}}(\hat{I}_t, I_t) + \beta(\lambda) \cdot \mathcal{R}_{\mathrm{attr}}(\lambda). \tag{7}$$

with $\mathcal{D}_{\mathrm{render}}$ computed between the rendered image set $\hat{I}_t$ and the reference image set I_t having 12 viewpoints. $\hat{I}_t$ is derived by rendering Gaussian primitives $\mathcal{G}_t$ in Eq. (5). For 8iVFB, Owlii, and THuman, I_t is rendered from the corresponding meshes. For DanceNet3D, I_t is rendered from the original Gaussian primitives. $\mathcal{R}_{\mathrm{attr}}$ is estimated from the entropy likelihood of quantized attribute latents. $\beta(\lambda)$ controls the R-D trade-off.

Compared Methods. We compare three groups of baselines. G-PCC TMC13v23 [GPC 2023], V-PCC TMC2v25 (utilizing HM-16.20 by default) [VPC 2023] and Unicorn [Wang et al. 2025a,b] represent standard and learning-based point cloud compression methods. Their decoded colored point clouds are rendered with OpenGL [Zhou et al. 2018] or a Gaussian renderer. B2P (G-PCC) [Hu et al. 2025] is the state-of-the-art decode-to-Gaussian route, where G-PCC provides the geometry information and the attribute stream is decoded into renderable Gaussians. Our Gameleon performs the lossless geometry decoding, variable-rate Gaussianized attribute decoding, and final rendering within an end-to-end playback system.

Evaluation Metrics. We adopt a display-oriented evaluation protocol with all methods compared under the same conditions. Each subject is rendered from 12 fixed orbit cameras uniformly distributed at 30° azimuth intervals, and the same views are used for metric computation. For G-PCC and Unicorn, we first decode

Table 1. Compression and rendering performance. Static results evaluate intra-frame coding with independent frames, while dynamic results evaluate inter-frame coding using decoded references as temporal priors. CR denotes the ratio between raw data size and compressed bitstream size.

Method	8iVFB (static) *4 seq. / 1200 frames*					Owlii (static) *4 seq. / 400 frames*					THuman (static) *1 seq. / 5 frames*					Owlii (dynamic) *4 seq. / 400 frames*				
	BPP	CR ↑	PSNR ↑	LPIPS ↓	MS-SSIM ↑	BPP	CR ↑	PSNR ↑	LPIPS ↓	MS-SSIM ↑	BPP	CR ↑	PSNR ↑	LPIPS ↓	MS-SSIM ↑	BPP	CR ↑	PSNR ↑	LPIPS ↓	MS-SSIM ↑
G-PCC (OpenGL)	0.5195	340.28	26.6142	0.0333	0.9790	0.3680	501.48	26.9085	0.0266	0.9845	0.5179	352.54	26.4200	0.0266	0.9810	0.3706	531.73	24.2630	0.0299	0.97308
G-PCC (Gaussian)	0.5195	340.28	24.3725	0.0740	0.9433	0.3680	501.48	28.6886	0.0316	0.9883	0.5179	352.54	27.7733	0.03014	0.9775	0.3706	531.73	27.4140	0.0369	0.9814
V-PCC (OpenGL)	0.4843	365.01	24.6367	0.02531	0.97437	0.3576	516.07	25.2321	0.02597	0.97903	0.4710	387.65	26.5723	0.02726	0.97386	0.3506	562.07	24.3919	0.03652	0.96947
V-PCC (Gaussian)	0.4843	365.01	29.5786	0.01932	0.99242	0.3576	516.07	29.5571	0.02402	0.99256	0.4710	387.65	27.9783	0.02613	0.97980	0.3506	562.07	27.9836	0.03350	0.98388
Unicorn (OpenGL)	0.4818	366.91	27.0392	0.0188	0.9863	0.2857	645.94	28.0814	0.0225	0.9875	0.5166	353.43	28.6936	0.0427	0.9732	0.2796	704.80	26.0173	0.0310	0.9782
Unicorn (Gaussian)	0.4818	366.91	29.6311	0.0232	0.9920	0.2857	645.94	29.7463	0.0253	0.9898	0.5166	353.43	29.8150	0.0326	0.9755	0.2796	704.80	28.5123	0.0314	0.9819
B2P (G-PCC)	0.4752	372.00	33.1318	0.0254	0.9950	0.3399	542.94	31.4695	0.0290	0.9943	0.6163	296.25	34.4874	0.0201	0.9953	0.3530	558.25	29.5628	0.0383	0.9858
Ours	**0.3855**	**458.56**	**35.6303**	**0.0122**	**0.9972**	**0.2451**	**752.94**	**34.3529**	**0.0175**	**0.9964**	**0.4262**	**428.39**	**36.3617**	**0.0118**	**0.9937**	**0.2528**	**779.51**	**32.0710**	**0.0257**	**0.9899**

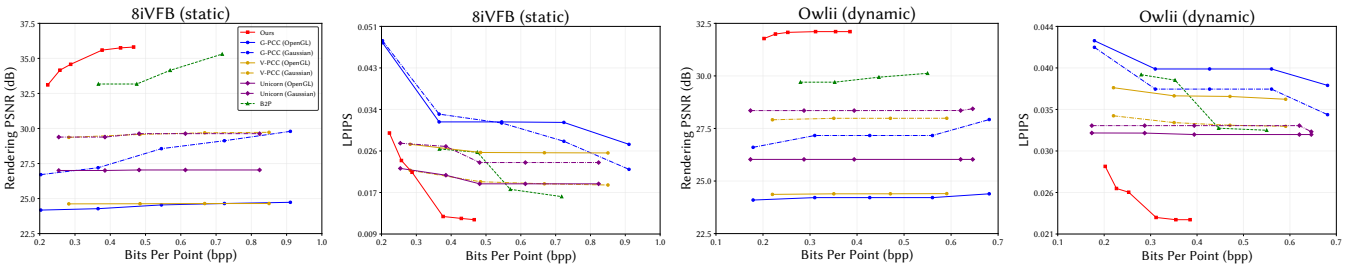

Fig. 4. **Rate-Distortion Curves.** We evaluate rendering quality (PSNR and LPIPS) under various bits per input point (bpp) on diverse datasets.

colored point clouds and then render target-view images using either OpenGL [Zhou et al. 2018] or a Gaussian renderer. For B2P and our Gameleon, the decoded features directly form Gaussian primitives for rendering.

We report rate, screen-space rendering quality, and runtime. Rate is measured in bits per input point (bpp), computed as the total bitstream size divided by the total number of input points. Rendering quality is measured using PSNR, LPIPS [Zhang et al. 2018], and MS-SSIM [Wang et al. 2004, 2003]. Coding efficiency over rate-quality curves is further evaluated using BD-BR [Bjøntegaard 2001]. A negative BD-BR indicates bitrate savings at the same quality.

Runtime is decomposed into geometry decoding, Gaussianized attribute decoding, and rendering. Gaussianized attribute decoding includes entropy decoding, rendering feature reconstruction, and Gaussian formation. Rendering time measures the cost of generating target-view images from the decoded representation.

4.2 Performance

In the following, we report results at certain bitrates only due to page limitation. Additional bitrate points and detailed BD-BR results are provided in Fig. 8 and Appendix.

Static Coding. Table 1 summarizes results on static 3D content. Figures 4 and 9 shows the corresponding R-D curves. Gameleon achieves the best rendering-oriented rate-quality trade-off on all datasets. For datasets with reliable overlapping R-D curves, Gameleon achieves BD-BR savings of 50.62% and 54.78% over B2P (G-PCC) on 8iVFB and THuman, respectively. Table 1 presents that Gameleon reduces BPP by 18.9%, 27.9%, and 30.8% over B2P (G-PCC) on 8iVFB, Owlii, and THuman, while improving PSNR by 2.50 dB, 2.88 dB, and 1.87 dB, respectively. Compared with the strongest G-PCC-based and Unicorn-based rendering pipeline, Gameleon consistently attains remarkable PSNR, MS-SSIM, and LPIPS gains. Visualization results are provided in Fig. 5 and Fig. 10.

Table 2. Comparison with G-PCC on 3D Gaussian representation.

Dataset	Method	Size (MB/s) ↓	PSNR ↑	LPIPS ↓	MS-SSIM ↑
DanceNet3D	G-PCC	0.287	30.255	0.0399	0.9644
	Ours	**0.213**	**30.871**	**0.0319**	**0.9747**

Dynamic Coding. Table 1 also reports dynamic coding results on Owlii. Gameleon outperforms both reconstruction-oriented playback pipelines and the decode-to-Gaussian baseline. For example, compared with B2P (G-PCC), it reduces BPP from 0.3530 to 0.2528, improves PSNR from 29.56 dB to 32.07 dB, LPIPS from 0.0383 to 0.0257, and MS-SSIM from 0.9858 to 0.9899. R-D curves in Fig. 4 further present our superiority over a wide bit range.

Table 2 further reports dynamic coding results on DanceNet3D, a 3DGS-based dynamic human performance dataset containing the native Gaussian primitives. For Gaussian attributes, we retain only RGB colors for coding, while G-PCC compresses all attributes. The decoded Gaussians are rendered to images and compared with images rendered from the original Gaussians to compute PSNR. For a fair comparison, we set the quantization parameter in G-PCC to achieve a comparable bitrate as ours. Results show that Gameleon significantly outperforms G-PCC in all metrics, even though G-PCC transmits much more attributes for rendering. Although the native Gaussian attributes are reduced to color only, the rendered images remain highly consistent with the original ones, achieving a PSNR of 30.871 and an MS-SSIM of 0.9747. This is because the screen-space loss drives Gameleon to learn Gaussian properties during construction at the decoder side.

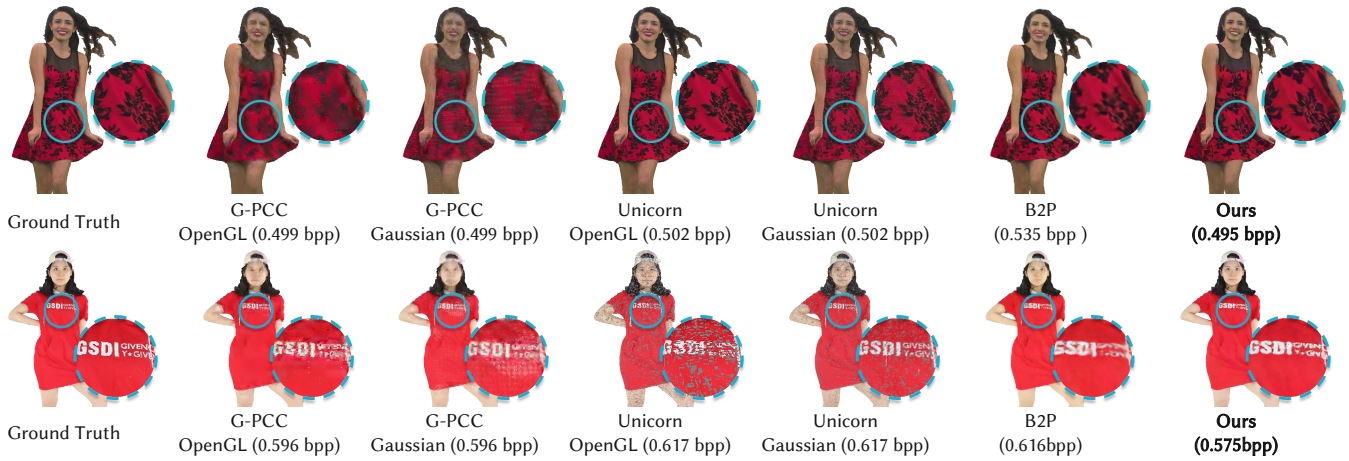

Fig. 5. **Rendering Visualization Comparison**. Top row: the sample *red_and_black* from 8iVFB. Bottom row: the sample #0519 from THuman. Columns from left to right: Ground Truth, G-PCC (OpenGL/Gaussian), Unicorn (OpenGL/Gaussian), B2P (G-PCC), and Gameleon (Ours).

Table 3. Runtime comparison on dynamic datasets. All decoding and rendering times are reported in seconds per frame.

Dataset	Method	Model Size (MB)	Decoding (s)	Render (s)	FPS
Owlii	G-PCC	–	2.376	0.280 / 0.348	0.37 / 0.36
	V-PCC	–	0.158	0.313 / 0.375	2.12 / 1.87
	Unicorn	281.66	3.215	0.337 / 0.412	0.28 / 0.28
	B2P (G-PCC)	105.73	0.263	0.003	3.75
	Ours	**97.35**	**0.051**	**0.003**	**18.51**
DanceNet3D	G-PCC	–	12.26	0.41	0.07
	Ours	**347.23**	**0.065**	**0.002**	**14.93**

Computational Complexity. We evaluate dynamic runtime of various methods with an end-to-end playback protocol and report model size, per-frame decoding time, rendering time, and FPS in Table 3. For G-PCC, V-PCC, and Unicorn, the runtime covers geometry decoding, color attribute decoding, and rendering using the OpenGL/Gaussian renderer. For B2P and our Gameleon, the runtime includes bitstream decoding, rendering-feature decoding, Gaussian construction and rendering. It is observed that Gameleon reaches 18.51 FPS on Owlii and 14.93 FPS on DanceNet3D on a single NVIDIA RTX 4090 GPU, whereas other methods are much slower, e.g., V-PCC runs at about 2 FPS and B2P (G-PCC) at 3.75 FPS. The speed on DanceNet3D decreases because we enlarge the kernel size from 3 to 7, as its sparser geometry requires a wider spatial context for stable feature aggregation.

Table 4. Ablation comparison with B2P. Ours-GAC (G-PCC) uses the same G-PCC geometry coder as B2P, but replaces B2P decoder with our GAC.

Dataset	Variant	Geo.	Attr.	BPP	PSNR	LPIPS
8iVFB	B2P (G-PCC)	G-PCC	B2P	0.4752	33.1318	0.0254
	Ours-GAC (G-PCC)	G-PCC	GAC	0.4047	35.6303	0.0122
	Ours	**LGC**	**GAC**	**0.3855**	**35.6303**	**0.0122**
THuman	B2P (G-PCC)	G-PCC	B2P	0.6163	34.4874	0.0201
	Ours-GAC (G-PCC)	G-PCC	GAC	0.4503	36.3617	0.0118
	Ours	**LGC**	**GAC**	**0.4262**	**36.3617**	**0.0118**

4.3 Ablation Study

Ablation studies are conducted to evaluate components in Gameleon.

Dynamic Prior Modeling. We ablate the temporal prior on the dynamic Owlii dataset, and the results are illustrated in Fig. 6. The static version encodes each frame using spatial lower-scale context only, while the dynamic one uses spatiotemporal context. All other settings remain unchanged. Results show that the dynamic model achieves 10.81% BD-BR reduction over the static model and yields lower LPIPS across the tested bitrate range. This confirms the effectiveness of properly using decoded reference priors in Gameleon.

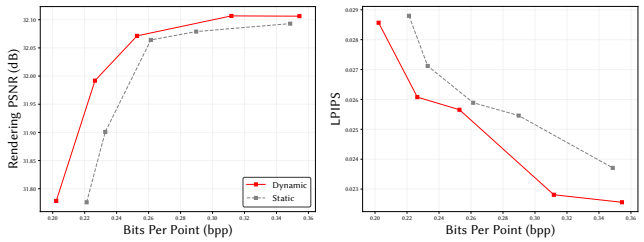

Fig. 6. **Dynamic Prior Ablation on Owlii.** We compare the dynamic model with a static one that removes the reference-frame temporal prior. Using the temporal prior improves the rate-quality trade-off, achieving a BD-BR (PSNR) reduction of −10.81% relative to the static model.

Runtime Acceleration. Figure 7 evaluates three runtime acceleration strategies on the dynamic Owlii test set: channel-parallel entropy decoding (CP), cross-frame pipeline (CF), and compact Gaussian constructor (CG). The left plot shows the R-D curves, while the right plot reports throughput of different configurations. CF only changes the execution schedule and therefore does not affect R-D performance. Starting from the serial baseline of 8.78 FPS, enabling CP, CF, and CG individually increases throughput to 15.32 FPS, 12.58 FPS, and 11.06 FPS, respectively. CP slightly degrades R-D performance but substantially improves decoding throughput. Combining CP+CF forms our main mode (Ours), reaching 18.51 FPS.

Table 5. Object generalizability of Gameleon on five 12-bit non-human samples recommended by MPEG CTC.

Method	Dense Average *facade, house*					Sparse Average *arco, shiva, statue*				
	BPP	CR ↑	PSNR ↑	LPIPS ↓	MS-SSIM ↑	BPP	CR ↑	PSNR ↑	LPIPS ↓	MS-SSIM ↑
G-PCC (OpenGL)	3.0587	156.32	20.4832	0.1038	0.9083	3.6895	251.42	22.3501	0.0889	0.9327
G-PCC (Gaussian)	3.0587	156.32	21.8531	0.0977	0.9235	3.6895	251.42	24.8794	0.0823	0.9557
V-PCC (OpenGL)	2.8422	145.26	20.8727	0.1053	0.9107	3.4835	286.24	22.9847	0.0855	0.9345
V-PCC (Gaussian)	2.8422	145.26	21.7427	0.0979	0.9231	3.4835	286.24	25.1147	0.0803	0.9560
Unicorn (OpenGL)	2.9584	163.83	20.7935	0.0958	0.9131	3.1372	283.15	23.3704	0.0861	0.9328
Unicorn (Gaussian)	2.9584	163.83	22.6835	0.0928	0.9357	3.1372	283.15	25.4498	0.0785	0.9588
B2P (G-PCC)	2.8932	189.17	23.1523	0.0894	0.9375	3.3859	300.16	25.8955	0.0768	0.9619
Ours	**2.1673**	**228.05**	**24.9437**	**0.0835**	**0.9518**	**2.4138**	**338.45**	**27.8845**	**0.0716**	**0.9723**

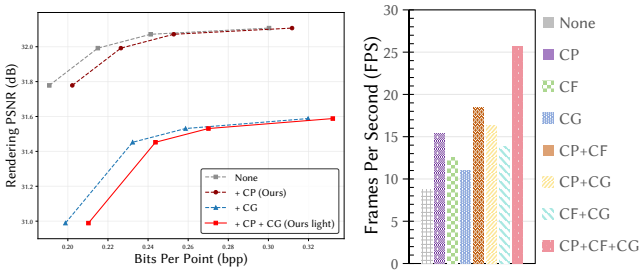

Fig. 7. **Runtime Acceleration Ablation on Owlii.** End-to-end FPS is reported under combinations of channel-parallel entropy decoding (CP), cross-frame pipeline (CF), and compact Gaussian constructor (CG). "None" denotes the serial baseline without any acceleration. CP+CF forms **Ours** with 18.51 FPS; CP+CF+CG forms **Ours-light** with 25.64 FPS.

Table 6. Scene generalizability of Gameleon on UrbanScene3D.

Method	BPP ↓	Decoding (s/1M points)	PSNR ↑	LPIPS ↓	MS-SSIM ↑
G-PCC (Gaussian)	1.620	4.87	23.733	0.1359	0.9027
Ours	**1.365**	**0.33**	**25.683**	**0.1038**	**0.9213**

The combinations CP+CG and CF+CG achieve 16.32 FPS and 13.88 FPS, respectively. Applying all three designs yields our lightweight mode, Ours-light, reaching 25.64 FPS and fully meeting on-demand performance requirements.

Comparison with B2P. We further compare Gameleon with B2P to separate the effects of geometry and attribute coding. In Table 4, Ours-GAC (G-PCC) keeps the same lossless G-PCC geometry as B2P but replaces the B2P decoder with our GAC, substantially improving rendering quality and bitrate efficiency. For example, on 8iVFB, PSNR increases from 33.13 dB to 35.63 dB while BPP decreases from 0.4752 to 0.4047. Replacing G-PCC with our LGC further reduces BPP to 0.3855. These results further confirm the effectiveness of LGC and GAC in Gameleon.

4.4 Generalizability

Further, we evaluate the generalizability of Gameleon beyond the primary experiments from three aspects: object generalization, scene generalization, and edge deployment. Detailed settings for scene generalization and edge deployment are provided in the Appendix.

Table 7. On-device deployment performance on Owlii sequences.

Device	Backend	Version	TFLOPS	Decoding (s/frame)	FPS ↑
NVIDIA RTX 4090	CUDA	Full	82.58	0.054	18.51
		Light		0.039	25.64
Mac (M5 Pro)	Metal	Full	10.27	0.129	7.73
		Light		0.105	9.51

Object Generalizability. To evaluate the object generalizability of Gameleon, we test it on five 12-bit non-human samples from MPEG CTC, including two dense samples and three sparse samples. Table 5 reports the average results over these samples. Gameleon achieves consistent gains over existing methods, reducing BPP by 32.64%, 24.48%, and 27.40% compared with G-PCC (Gaussian), Unicorn (Gaussian), and B2P (G-PCC), respectively. The corresponding PSNR improvements are 3.04 dB, 2.36 dB, and 1.91 dB.

Scene Generalizability. We further evaluate Gameleon on a real urban scene from UrbanScene3D [Lin et al. 2022]. As shown in Table 6, Gameleon outperforms G-PCC by achieving 15.74% BPP reduction, 1.95 dB PSNR improvement, and a decoding time reduction from 4.87 s to 0.33 s per million points.

On-device Deployment. We further deploy Gameleon on an M5 Pro Mac to evaluate its on-device performance by replacing CUDA operators with custom Metal kernels. Despite its substantially lower computational capability than RTX 4090 (10.27 vs. 82.58 TFLOPS), the full and light versions of Gameleon achieve 7.73 and 9.51 FPS, respectively, by leveraging the M5 Neural Accelerators (Table 7), compared with 18.51 and 25.64 FPS on RTX 4090, while maintaining nearly identical compression performance.

5 Conclusion

Gameleon is a real-time system for on-demand 4D performance that replaces the conventional separate coding and rendering pipeline with end-to-end coding-to-rendering. Its multiscale geometry and attribute coders transform input 3D geometry and attributes into Gaussian rendering features, which are directly decoded as renderable Gaussian primitives. This design balances rendering quality and coding speed.

In this paper, we demonstrate the framework of Gameleon on point clouds and Gaussian primitives with RGB attributes. Future work will support richer attributes and representations, along with real-time encoding.

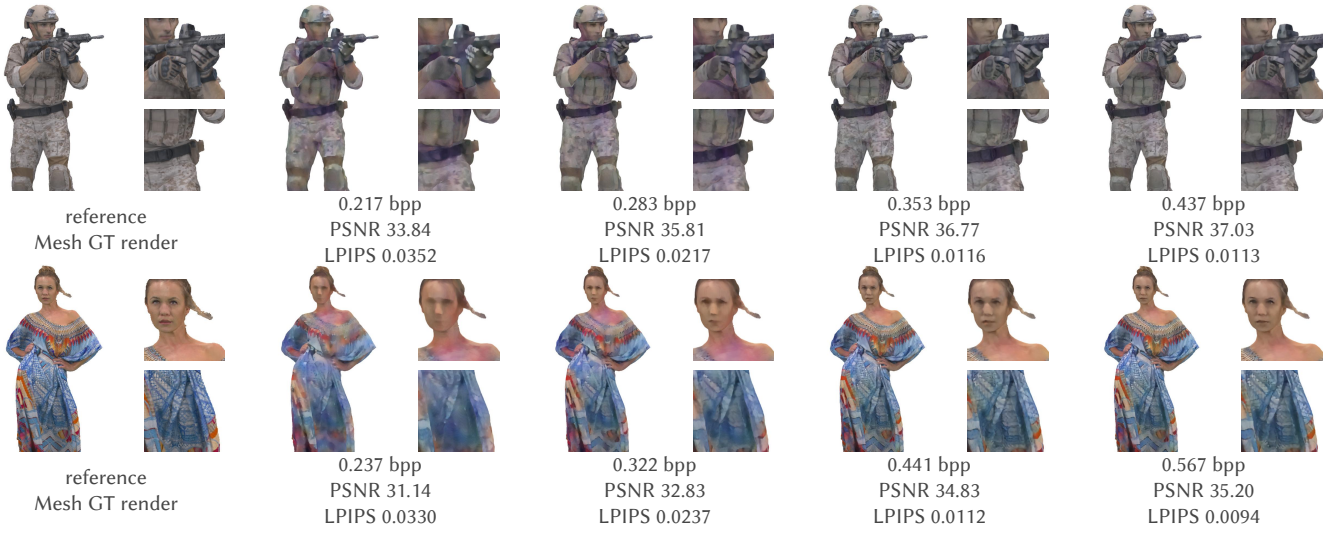

Fig. 8. **More Rendering Results.** Rendering results of Gameleon at different bitrate points are visualized.

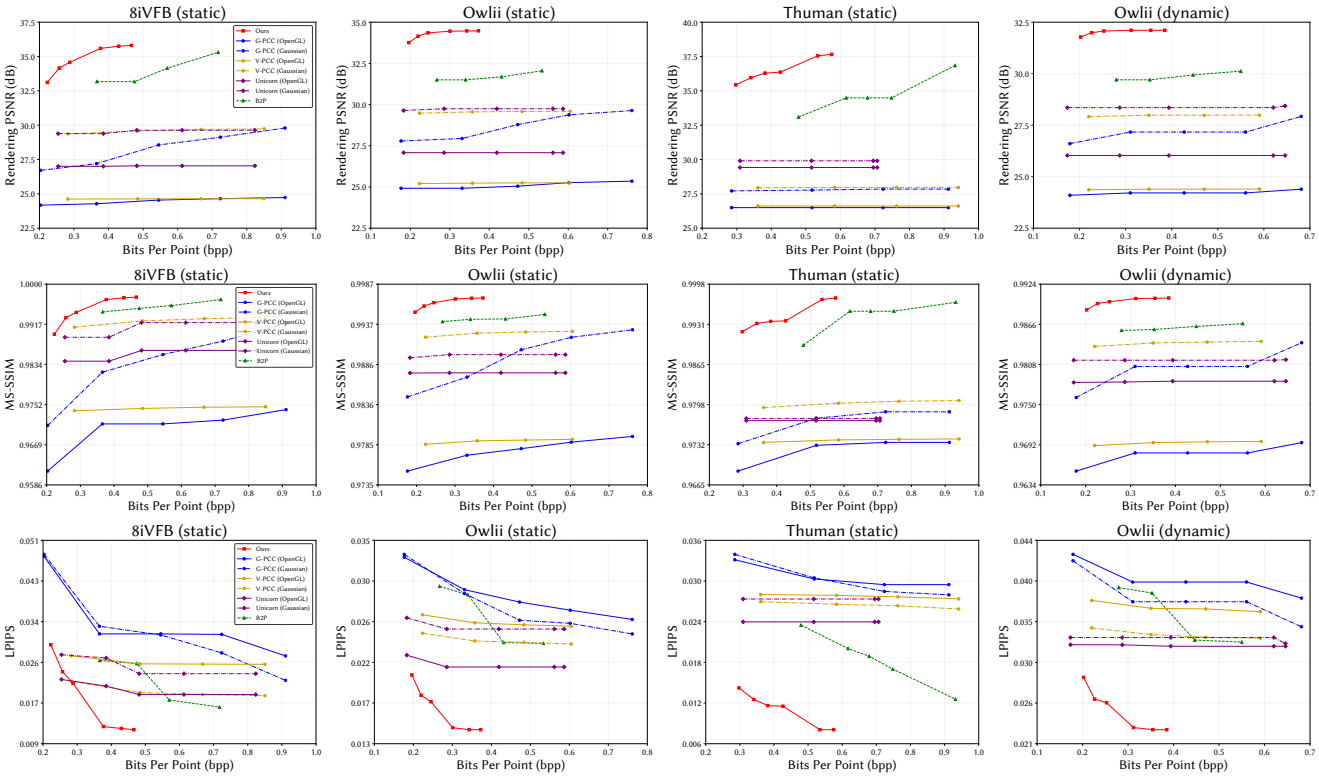

Fig. 9. **Rate-Distortion Curves.** We evaluate rendering quality (PSNR, MS-SSIM, and LPIPS) under various bits per input point (bpp) on more datasets.

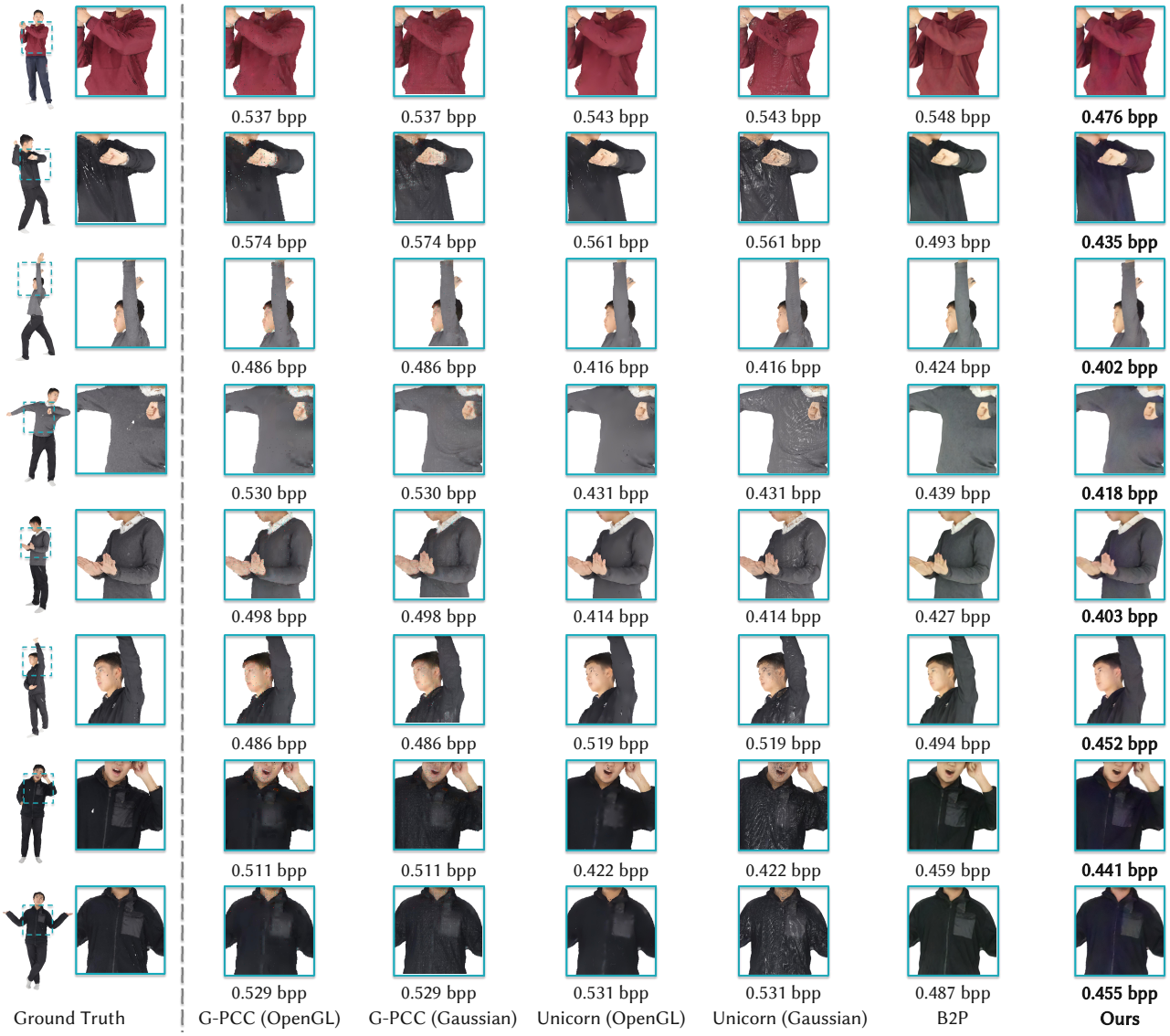

Fig. 10. **Visual Quality Comparison on THuman**. From left to right: GT, G-PCC with OpenGL/Gaussian rendering, Unicorn with OpenGL/Gaussian rendering, B2P (G-PCC), and our Gameleon.

Fig. 11. **Gameleon on-demand System**. Given any 3D data having x, y, z coordinates and RGB colors, our system directly decodes the compressed bitstream into Gaussian primitives for rendering.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant No. 62571174) and the Advanced Computation Center of Hangzhou Normal University. We also thank Zhenyan Lin (2024210402004@stu.hznu.edu.cn) at Hangzhou Normal University for his valuable contribution to the implementation of operators using custom Metal kernels.

References

2023. Information technology - Coded representation of immersive media - Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC). *ISO/IEC* (11 2023), 23090–5.

2023. Information technology - Coded representation of immersive media Part 9: Geometry-based point cloud compression. *ISO/IEC* (3 2023), 23090–9.

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

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. 2021. Overview of the Versatile Video Coding (VVC) Standard and its Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3736–3764. doi:10.1109/TCSVT.2021.3101953

Chao Cao, Marius Preda, and Titus Zaharia. 2019. 3D Point Cloud Compression: A Survey. In *Proceedings of the 24th International Conference on 3D Web Technology*. 1–9.

Chao Cao, Marius Preda, Vladyslav Zakharchenko, Euee S Jang, and Titus Zaharia. 2021. Compression of sparse and dense dynamic point clouds—methods and standards. *Proc. IEEE* 109, 9 (2021), 1537–1558.

Guodong Chen, Filip Hácha, Libor Váša, and Mallesham Dasari. 2025a. TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes. In *Proceedings of the 16th ACM Multimedia Systems Conference*. 79–89.

Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2024a. Fast feedforward 3D gaussian splatting compression. *arXiv preprint arXiv:2410.08017* (2024).

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2024b. HAC: Hash-grid assisted context for 3D gaussian splatting compression. In *European Conference on Computer Vision*. Springer, 422–438.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2025b. HAC++: Towards 100x compression of 3D gaussian splatting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

YiHyun Choi, Jong-Beom Jeong, Soonbin Lee, and Eun-Seok Ryu. 2022. Overview of the video-based dynamic mesh coding (V-DMC) standard work. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 578–581.

Pinxuan Dai, Peiquan Zhang, Zheng Dong, Ke Xu, Yifan Peng, Dandan Ding, Yujun Shen, Yin Yang, Xinguo Liu, Rynson WH Lau, et al. 2025. 4d gaussian videos with motion layering. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–14.

E. d'Eon, B. Harrison, T. Myers, and P. A. Chou. 2017. *8i voxelized full bodies - a voxelized point cloud dataset*. Input document WG11M40059/WG1M74006. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG).

Zheng Dong, Ke Xu, Yaoan Gao, Hujun Bao, Weiwei Xu, and Rynson WH Lau. 2024. Gaussian surfel splatting for live human performance capture. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–17.

Philipp Erler, Lizeth Fuentes-Perez, Pedro Hermosilla, Paul Guerrero, Renato Pajarola, and Michael Wimmer. 2024. Ppsurf: Combining patches and point convolutions for detailed surface reconstruction. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15000.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. Lightgaussian: Unbounded 3D gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems* 37 (2024), 140138–140158.

Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. 2022. OctAttention: Octree-Based Large-Scale Contexts Model for Point Cloud Compression. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 625–633.

Danillo Graziosi, Ohji Nakagami, Shinroku Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020), e13.

Yueyu Hu, Ran Gong, Qi Sun, and Yao Wang. 2024. Low latency point cloud rendering with learned splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5752–5761.

Yueyu Hu, Ran Gong, and Yao Wang. 2025. Bits-to-photon: End-to-end learned scalable point cloud compression for direct rendering. In *2025 IEEE International Conference on Image Processing (ICIP)*. IEEE, 953–958.

Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. 2023. Neural kernel surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4369–4379.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 2023. 3D gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. 2024. A hierarchical 3D gaussian representation for real-time rendering of very large datasets. *ACM Transactions On Graphics (TOG)* 43, 4 (2024), 1–15.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3D gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21719–21728.

Zehong Li, Jiahao Zhu, Dandan Ding, and Zhan Ma. 2025. Improving Occupancy Prediction for Multiscale Point Cloud Geometry Compression. *IEEE Transactions on Circuits and Systems for Video Technology* (2025), 1–1.

Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. 2022. Capturing, reconstructing, and simulating: the urbanscene3d dataset. In *European Conference on Computer Vision*. Springer, 93–109.

Bojun Liu, Yangzhi Ma, Li Li, Dong Liu, Zhu Li, and Houqiang Li. 2026. Next bit prediction: A unified lossless and lossy point cloud geometry compression framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2026), 1–18. doi:10.1109/TPAMI.2025.3650590

Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. 2025. Maskgaussian: Adaptive 3D gaussian representation from probabilistic masks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 681–690.

Andrea Maggiordomo, Federico Ponchio, Paolo Cignoni, and Marco Tarini. 2020. Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools. *Computer Aided Geometric Design* 83 (2020), 101943.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. 2024. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.

MPEG 3D Graphics Coding and Haptics Coding. 2025. CTC on AI-based Point Cloud Coding. ISO/IEC JTC1/SC29/WG7, Document N01122.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. 2024. Compressed 3D gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10349–10358.

Maurice Quach, Jiahao Pang, Dong Tian, Giuseppe Valenzise, and Frédéric Dufaux. 2022. Survey on Deep Learning-Based Point Cloud Compression. *Frontiers in Signal Processing* 2 (2022), 846972.

S. Schwarz, M. Preda, V. Baroncini, et al. 2019. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9 (2019), 133–148.

Xihua Sheng, Li Li, Dong Liu, Zhiwei Xiong, Zhu Li, and Feng Wu. 2021. Deep-PCAC: An end-to-end deep lossy compression framework for point cloud attributes. *IEEE Transactions on Multimedia* 24 (2021), 2617–2632.

Rui Song, Chunyang Fu, Shan Liu, and Ge Li. 2023. Efficient Hierarchical Entropy Model for Learned Point Cloud Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14368–14377.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1649–1668.

Sajid Umair, Birendra Kathariya, Zhu Li, Anique Akhtar, and Geert Van der Auwera. 2024. ResNeRF-PCAC: Super Resolving Residual Learning NeRF for High Efficiency Point Cloud Attributes Coding. In *2024 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3540–3546.

Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. 2022. Sparse Tensor-Based Multiscale Representation for Point Cloud Geometry Compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 7 (2022), 9055–9071.

Jianqiang Wang and Zhan Ma. 2022. Sparse tensor-based point cloud attribute compression. In *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*. 59–64. doi:10.1109/MIPR54900.2022.00018

Jianqiang Wang, Ruixiang Xue, Jiaxin Li, Dandan Ding, Yi Lin, and Zhan Ma. 2025a. A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part I: Geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

Jianqiang Wang, Ruixiang Xue, Jiaxin Li, Dandan Ding, Yi Lin, and Zhan Ma. 2025b. A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding — Part II: Attribute. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, 01 (Jan. 2025), 252–268.

Kangli Wang and Wei Gao. 2025. UniPCGC: Towards Practical Point Cloud Geometry Compression via An Efficient Unified Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kangli Wang, Qianxi Yi, Yuqi Ye, Shihao Li, and Wei Gao. 2025c. AnyPcc: Compressing Any Point Cloud with a Single Universal Model. *arXiv preprint arXiv:2510.20331* (2025).

Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. 2024b. Vˆ 3: Viewing Volumetric Videos on Mobiles via Streamable 2D Dynamic Gaussians. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–13.

Xinjie Wang, Yifan Zhang, Ting Liu, Xinpu Liu, Ke Xu, Jianwei Wan, Yulan Guo, and Hanyun Wang. 2025d. TopNet: Transformer-efficient occupancy prediction network for octree-structured point cloud geometry compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 27305–27314.

Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C Kot, and Bihan Wen. 2024a. Contextgs: Compact 3D gaussian splatting with anchor level context model. *Advances in neural information processing systems* 37 (2024), 51532–51551.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The thrity-seventh asilomar conference on signals, systems & computers, 2003*, Vol. 2. Ieee, 1398–1402.

WG 07 MPEG 3D Graphics Coding and Haptics Coding. 2024. Common test conditions for G-PCC. ISO/IEC JTC1/SC29/WG7, Document N00944.

X. Yi, L. Yao, and W. Ziyu. 2017. *Owlii dynamic human mesh sequence dataset.* Input document m41658. ISO/IEC Joint MPEG/JPEG.

Kang You, Tong Chen, Dandan Ding, M Salman Asif, and Zhan Ma. 2025. RENO: Real-Time Neural Compression for 3D LiDAR Point Clouds. *arXiv preprint arXiv:2503.12382* (2025).

Tao Yu, Zerong Zheng, Kaiwen Guo, Pengpeng Liu, Qionghai Dai, and Yebin Liu. 2021. Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5746–5756.

Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. 2025. Cat-3dgs: A context-adaptive triplane approach to rate-distortion-optimized 3dgs compression. *arXiv preprint arXiv:2503.00357* (2025).

Junteng Zhang, Tong Chen, Dandan Ding, and Zhan Ma. 2025. Neural Compression System for Point Cloud Video Streaming. *IEEE Transactions on Image Processing: a Publication of the IEEE Signal Processing Society* 34 (2025), 8032–8045.

Junteng Zhang, Tong Chen, Hao Zhu, Dong Wang, Dandan Ding, and Zhan Ma. 2024. Compressing 3D Gaussian Splatting via a Generalizable Neural Coder. In *2024 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 1–5.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018).

Wenjie Zou, Shizhuo Zhang, Fuzheng Yang, and Marius Preda. 2025. Standardization status of MPEG video-based dynamic mesh coding (V-DMC). In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.