PROTEUS: A TRUNCATION-ROBUST ENTROPY MODEL FOR PROGRESSIVE LIDAR COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

LiDAR point clouds provide explicit, deterministic physical boundaries critical for collaborative safety-critical perception. However, modern wireless channels are subject to dynamic bandwidth fluctuations and transient link outages. While most existing LiDAR codecs assume ideal channels and collapse under transmission disruptions, robust alternatives (such as deep JSCC or MDC) rely on statistical estimation, turning exact physical measurements into unverified algorithmic estimates. To address this dilemma, we propose **Proteus**, a learned LiDAR codec operating on 2D range images. Built upon a progressive bit-plane slicing representation, Proteus inherently ensures that stream truncation mathematically maps to deterministic spatial precision degradation without geometric hallucinations. To reconcile high-efficiency bits-back coding with stream-level truncation robustness, Proteus decouples the frame representation into two independent coding pathways: (i) the **sig**nificant range bit-planes (SIG), which employs a self-contained bits-back coder equipped with an Autoregressive Initial Bits (ArIB) mechanism to eliminate single-frame overhead while securing an essential perceptual lower bound; and (ii) the **ins**ignificant range bit-planes and attributes (INS), which serializes the remaining low range planes before subordinate attributes via FIFO range coding, gracefully shedding attribute precision first during bandwidth drops. Extensive experiments on the Waymo Open Dataset and SemanticKITTI demonstrate that Proteus tolerates up to approximately 70% bitstream truncation, while outperforming established standards (e.g., G-PCC) and the representative learned compressor Unicorn under ideal channel conditions. The code will be released upon the acceptance of the paper.

1 INTRODUCTION

LiDAR point clouds are a key data structure for scene understanding in autonomous driving, robotics, and remote sensing, providing rich geometric, shape, and scale information (Guo et al., 2021). Beyond geometry, modern LiDARs also record per-point attributes such as reflectance intensity that carry complementary cues for downstream perception (Zhu et al., 2025). Crucially, active LiDAR sensors directly measure explicit, deterministic physical boundaries (Royo & Ballesta-Garcia, 2019). In safety-critical tasks such as collision detection, freespace boundary delineation, and emergency hazard avoidance, these precise physical measurements serve as an indispensable and mathematically verifiable safety net (Royo & Ballesta-Garcia, 2019; Mao et al., 2023). While single-agent perception is inherently limited by physical line-of-sight constraints and severe occlusions (Wang et al., 2020), collaborative applications resolve these bottlenecks by continuously sharing these deterministic physical boundaries. Yet, in collaborative applications such as vehicle-to-everything (V2X) communications or UAV swarms, transmitting these dense, high-volume point clouds (Gao et al., 2025) over shared and highly dynamic wireless channels (Hu et al., 2022) places both high-efficiency compression and channel-robust transmission on the same critical path.

Specifically, under modern communication protocol stacks (3GPP, 2024a), the wireless channel between agents is primarily subject to bandwidth fluctuation (Hu et al., 2022) and transient link outages (Li et al., 2023; Ren et al., 2024), as detailed in Appendix D. Bandwidth fluctuation causes in-transit stream truncation (Jeon et al., 2023). However, conventional codecs collapse under such truncation, requiring latency-heavy feedback loops to renegotiate the bitrate and stalling the real-time pipeline (Li et al., 2024). This demands a prefix-decodable bitstream that yields a valid reconstruction with-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

out prior coordination. Meanwhile, transient link outages cause complete, frame-level loss, requiring each frame to be self-contained, as inter-frame prediction breaks down when reference frames are lost in transit (Wiegand et al., 2003; Zhou et al., 2022). Most existing LiDAR codecs, primarily designed for ideal channels, provide neither property (Gao et al., 2025; Wang et al., 2025b; You et al., 2025).

To mitigate these transmission bottlenecks, recent learned robust communication frameworks, such as deep joint source-channel coding (Deep JSCC) (Bourtsoulatze et al., 2019; Zhang et al., 2025) or multiple description coding (MDC) (El Gamal & Cover, 1982; Chen et al., 2024), have successfully achieved continuous quality degradation under channel impairments, maintaining competitive PSNR or downstream mAP. However, because these approaches primarily rely on statistical or parametric estimation to reconstruct the missing or corrupted geometric regions, the decoded coordinates inevitably become an algorithmic estimate rather than an exact, faithful measurement (Zhang et al., 2025; Cheng et al., 2024; Hung et al., 2022). For safety-critical decisions, preserving the exact, physically bounded nature of the sensor's measurement, even at a coarser resolution, remains the indispensable bedrock of trustworthy collaborative intelligence. This prompts the question: ***how can a learned codec achieve robust progressive compression while strictly preserving mathematically bounded physical boundaries?***

To address these challenges, we propose Proteus, a learned LiDAR codec operating on 2D range images. By restricting coding dependencies strictly intra-frame, Proteus achieves frame-level packet-loss resistance. Proteus is built upon a progressive bit-plane slicing representation, ensuring that stream truncation mathematically maps to a deterministic spatial precision degradation of physical distances, rather than relying on unconstrained statistical estimation. To reconcile the high efficiency of bits-back coding with stream-level progressive truncation, we decouple the single-frame representation into two independent pathways: a self-contained bits-back coder for the **sig**nificant range bit-planes (SIG), and a FIFO range coder for the **ins**ignificant range bit-planes and attributes (INS). The non-truncatable SIG block utilizes an Autoregressive Initial Bits (ArIB) mechanism (Ryder et al., 2022; Zhang et al., 2024) to eliminate single-frame state overhead, securing a necessary base perceptual lower bound. Meanwhile, INS strategically serializes geometric bit-planes before subordinate attributes, allowing the system to naturally sacrifice attribute precision first during bandwidth dips while preserving structural geometry. Since SIG accounts for roughly 30% of the bitstream, Proteus remains decodable under up to 70% truncation.

Our contributions are as follows.

- (i) We address the fundamental incompatibility between truncation robustness and high-efficiency bits-back coding equipped with the Autoregressive Initial Bits (ArIB) mechanism to eliminate single-frame overhead, incurring only a minor penalty in lossless compression efficiency.
- (ii) Built upon a bit-plane slicing (BPS) representation, we design a joint geometry-attribute coding framework that strategically orders subordinate attribute data after geometry in the truncatable stream. This enables progressive, graceful degradation under truncation, allowing the system to naturally sacrifice attribute precision first during bandwidth dips while maintaining respectable downstream perception performance.
- (iii) Our proposed framework outperforms established standards and industrial codecs (such as G-PCC, Draco, and JPEG XL) as well as the representative learned baseline Unicorn under ideal channels, while uniquely tolerating up to 70% bitstream truncation under severe channel degradation.

2 Related Work

2.1 Point Cloud Compression

A point cloud consists of point positions and per-point attributes such as reflectance (Gao et al., 2025; Fang et al., 2022; Zhu et al., 2025), and most compression pipelines encode these two streams in separate stages. Geometry coders organize the unordered point set into a structured representation amenable to entropy coding (Liu et al., 2026). Octree-based methods represent geometry as occupancy trees with hand-designed (Graziosi et al., 2020) or learned context models (Huang et al.,

2020; Fu et al., 2022; Song et al., 2023). Voxel approaches discretize 3D space into regular grids via 3D convolutions (He et al., 2022; Que et al., 2021), while sparse tensor methods (Wang et al., 2023; 2025b; You et al., 2025) lower processing overheads on sparse scans. Alternatively, range images offer a memory-efficient 2D data structure to organize and represent point clouds (Meyer et al., 2019; Wang et al., 2022; Liu et al., 2023).

To exploit temporal redundancy, several coders condition each frame on previously decoded scans (Biswas et al., 2020; Wang et al., 2025b; Zhou et al., 2022; Song et al., 2026; Feng et al., 2020; Fan et al., 2022). While these inter-frame methods improve rate-distortion under ideal channels, a lost reference frame inevitably propagates reconstruction errors to subsequent frames (Wiegand et al., 2003; Zhou et al., 2022), favoring robust, self-contained intra-frame coding.

For attribute coding, G-PCC (Graziosi et al., 2020) is built on fixed transforms, while learned coders adopt neural transforms and context models (Wang et al., 2025a; Fang et al., 2022; Zhu et al., 2025). However, the common practice of running separate networks for geometry and attributes (Wang et al., 2025a;b) substantially increases end-to-end latency (Song et al., 2026).

2.2 Robust Compression and Transmission

Robust transmission under bandwidth fluctuation or packet loss has received less attention than rate-distortion optimization, particularly for point clouds (Gao et al., 2025; Zhang et al., 2025; Chen et al., 2024). We group existing approaches into three families.

The first family abandons source-channel separation, directly mapping the source signal to channel symbols via a learned encoder-decoder. Stemming from pseudo-analog transmission (Jakubczak & Katabi, 2010) and deep JSCC (Bourtsoulatze et al., 2019; Erdemir et al., 2023), this paradigm has been extended to point clouds (Zhang et al., 2025) to allow continuous quality degradation without cliff effects, at the cost of rendering decoded coordinates as statistical approximations.

The second family retains digital source coding and addresses packet loss via error resilience or multiple description coding (MDC) (El Gamal & Cover, 1982; Wiegand et al., 2003; Chen et al., 2024; Hung et al., 2022). Recent learning-based robust codecs similarly reconstruct missing regions from partial bitstreams (Cheng et al., 2024). In these approaches, lost regions are restored through codec-side statistical estimation rather than preserved faithfully from the source.

The third family encodes the bitstream so that any prefix decodes to a valid reconstruction (Liu et al., 2026; Jeon et al., 2023). Classical embedded-bitstream algorithms (Shapiro, 1993; Skodras et al., 2001) ensure deterministic, non-parametric precision degradation. Conversely, neural latent-variable progressive schemes like PLONQ (Lu et al., 2021) and DPICT (Lee et al., 2022) rely on parametric synthesis to reconstruct truncated latents. Integrating the principle of deterministic progressive degradation with neural latent-variable frameworks remains an open challenge.

3 Preliminaries

3.1 Range Image and Bit-Plane Slicing

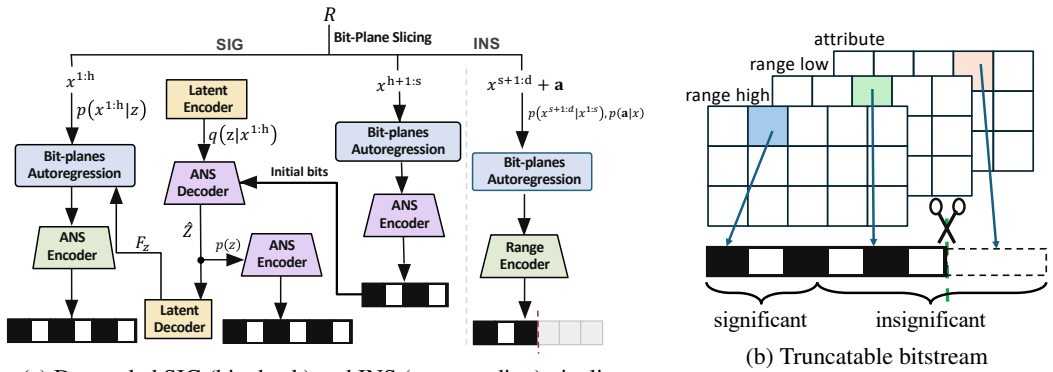
A LiDAR sensor records the scene by emitting laser beams along H elevation angles $\theta = \{\theta_1, \ldots, \theta_H\}$ and W azimuth angles $\phi = \{\phi_1, \ldots, \phi_W\}$. Each beam (i, j) measures the radial distance $r_{i,j}$ to the target surface. The resulting $H \times W$ matrix forms a 2D range image, from which the 3D Cartesian coordinates are losslessly reconstructed via

$$(x, y, z) = (r_{i,j} \cos\theta_i \cos\phi_j,\ r_{i,j} \cos\theta_i \sin\phi_j,\ r_{i,j} \sin\theta_i), \tag{1}$$

given that the scanning angles θ, ϕ are predetermined and fixed (Sun et al., 2020; Zhou et al., 2022). For a d-bit range image $\mathbf{x}$, bit-plane slicing decomposes each scalar value into d binary planes ordered from the most significant plane (MSP, $l = 1$) to the least significant plane (LSP, $l = d$):

$$\mathbf{x}^l = \lfloor \mathbf{x}/2^{d-l} \rfloor \bmod 2, \quad l \in \{1, \ldots, d\}. \tag{2}$$

Under this representation, dropping bit-planes progressively from LSP to MSP naturally corresponds to a deterministic degradation of spatial precision.



(a) Decoupled SIG (bits-back) and INS (range coding) pipelines

(b) Truncatable bitstream

Figure 1: Overview of the Proteus framework designed for truncation-robust LiDAR compression.

3.2 Bits-Back Coding and Generalized ArIB

Consider a latent-variable model $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ with variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$. Bits-back coding (Townsend et al., 2019) uses Asymmetric Numeral Systems (ANS) (Duda, 2013) to encode $\mathbf{x}$ by first decoding $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ from existing stream bits, then encoding $\mathbf{x}$ under $p_\theta(\mathbf{x}|\mathbf{z})$ and $\mathbf{z}$ under $p(\mathbf{z})$. While this achieves the optimal negative ELBO rate over infinite sequences, a standalone single frame lacks prior bits and must inject dummy initial bits of length $-\log q_\phi(\mathbf{z}|\mathbf{x})$, causing the single-frame transmission rate to degrade to

$$R_{\text{raw}} = -\log p_\theta(\mathbf{x} \mid \mathbf{z}) - \log p(\mathbf{z}), \tag{3}$$

which completely forfeits the bits-back savings $\log q_\phi(\mathbf{z}|\mathbf{x})$.

Autoregressive Initial Bits (ArIB) (Ryder et al., 2022; Zhang et al., 2024) resolves this bottleneck by partitioning the data into two subsets $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_B)$ (e.g., spatial partitions or bit-plane slices). The latent $\mathbf{z}$ is assigned exclusively to $\mathbf{x}_A$, and the bits generated by directly encoding $\mathbf{x}_B$ serve as the internal initial bits for bits-back coding on $\mathbf{x}_A$. Consequently, the dummy overhead is eliminated, yielding the net rate

$$R_{\text{ArIB}} = \underbrace{-\log p_\theta(\mathbf{x}_A \mid \mathbf{z}) - \log p(\mathbf{z}) + \log q_\phi(\mathbf{z} \mid \mathbf{x}_A)}_{-\text{ELBO}(\mathbf{x}_A)} - \log p(\mathbf{x}_B \mid \mathbf{x}_A). \tag{4}$$

Compared to R_{raw}, ArIB fully reclaims the rate reduction $\log q_\phi(\mathbf{z} \mid \mathbf{x}_A)$ for single-frame compression.

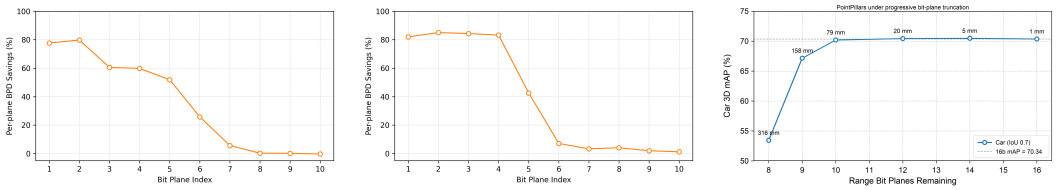
4 Method

4.1 Overview

We build on ArIB-BPS (Zhang et al., 2024) and redesign its bitstream for truncation robustness. In ArIB-BPS the insignificant planes and the bits-back coding of the significant planes share a single coupled bitstream, and truncating its tail leaves the significant planes undecodable. Our redesign decouples them into two independent coders: SIG, a self-contained bits-back coder for the significant planes, and INS, a range coder for the insignificant planes whose FIFO order tolerates truncation.

The redesign is grounded in three properties of LiDAR range images that distinguish them from generic RGB images:

- **A1 (bit-plane physical meaning).** The range value is a quantized physical distance. Truncating the N least significant bits of every range sample halves the precision N times, so bit-plane truncation is a deterministic precision degradation rather than an arbitrary distortion.
- **A2 (attributes subordinate to range).** Attribute channels are less important than geometry in downstream perception. Industrial pipelines accordingly represent them at lower bit depth than the range channel, and we order attributes after the range in the INS bitstream.



(a) SemanticKITTI BPD savings (b) WOD BPD savings (c) Downstream s-sweep

Figure 2: Parameter selection analysis for head depth h and split index s. (a, b) BPD savings achieved using latent variables used to determine h on SemanticKITTI and WOD. (c) Downstream 3D object detection performance under different s selections, where the dashed line represents the untruncated baseline.

- **A3 (range as skeleton prior).** The high bit planes of the range predict both the low bit planes of the range and the attribute channels (Zhang et al., 2024; Zhu et al., 2025). A hyperprior derived from the significant planes guides INS's encoding.

Figure 1 illustrates the resulting codec. Let $\mathbf{R} = (\mathbf{x}, \mathbf{a})$ denote the range image, where $\mathbf{x}$ is the range channel of bit depth d and $\mathbf{a}$ denotes the per-pixel attribute channels (e.g., reflectance intensity). The range channel is sliced into d bit planes $\mathbf{x}^{1:d}$ (Zhang et al., 2024), partitioned into three contiguous parts: the head sub-block $\mathbf{x}^{1:h}$, the mid sub-block $\mathbf{x}^{h+1:s}$, and the tail planes $\mathbf{x}^{s+1:d}$. The significant planes ($\mathbf{x}^{1:s}$, comprising the head and mid) form SIG to perform self-contained bits-back coding, generating initial bits internally to eliminate the single-frame overhead of Section 3.2. The tail planes $\mathbf{x}^{s+1:d}$ and attributes $\mathbf{a}$ form INS, encoded via a FIFO range coder to allow progressive decoding. Parameter configurations of h, s, d, and attribute depths are detailed in Section 5.1.

4.2 SIG Coding

The SIG coder encodes the significant planes $\mathbf{x}^{1:s}$ with a bits-back coding closed entirely inside SIG. SIG is split into a head sub-block $\mathbf{x}^{1:h}$ and a mid sub-block $\mathbf{x}^{h+1:s}$, where h is the head depth.

Encoding proceeds in three stages (Townsend et al., 2019; Ryder et al., 2022). (i) The mid sub-block $\mathbf{x}^{h+1:s}$ is encoded under $p(\mathbf{x}^{h+1:s})$ without latent conditioning, and the resulting bitstream provides the initial bits. (ii) The latent $\mathbf{z}$ is decoded from these bits under the posterior $q(\mathbf{z}|\mathbf{x}^{1:h})$. (iii) The head sub-block $\mathbf{x}^{1:h}$ is encoded under $p(\mathbf{x}^{1:h}|\mathbf{z})$, and $\mathbf{z}$ is encoded under $p(\mathbf{z})$. Decoding reverses this order. The net rate of SIG is

$$R_{\text{SIG}} = -\log p(\mathbf{x}^{h+1:s}) - \log p(\mathbf{x}^{1:h} \mid \mathbf{z}) - \log p(\mathbf{z}) + \log q(\mathbf{z} \mid \mathbf{x}^{1:h}), \tag{5}$$

where the first term is the rate of the mid sub-block and the remaining three terms are the bits-back rate of the head sub-block, equal to $-\text{ELBO}(\mathbf{x}^{1:h})$. The mid sub-block supplies the initial bits internally, so SIG requires no padding and does not depend on INS. The compression overhead of this decoupling design is evaluated in Section 5.3.

4.3 INS Coding and Truncation Properties

The INS coder encodes the tail planes of the range (the low $d - s$ bit planes) and the d_a-bit attribute channels $\mathbf{a}$. Following **A2**, we serialize INS as a sequence of bit-planes, ordering the range tail planes $\mathbf{x}^{s+1:d}$ before the attribute planes $\mathbf{a}^{1:d_a}$.

Let $k_r \in [s, d]$ and $k_a \in [0, d_a]$ denote the retained bit depths after truncation. A range coder provides the first-in-first-out (FIFO) encoding-decoding order that allows progressive decoding from any stream prefix, subject to the constraint that $k_a > 0$ only if $k_r = d$. To leverage the high range planes as a skeleton prior **A3**, the tail planes and attribute channels are sequentially conditioned on the reconstructed geometry. At any valid truncation point (k_r, k_a), the actual rate of the INS stream is:

$$R_{\text{INS}}(k_r, k_a) = -\log p(\mathbf{x}^{s+1:k_r} \mid \mathbf{x}^{1:s}) - \log p(\mathbf{a}^{1:k_a} \mid \mathbf{x}). \tag{6}$$

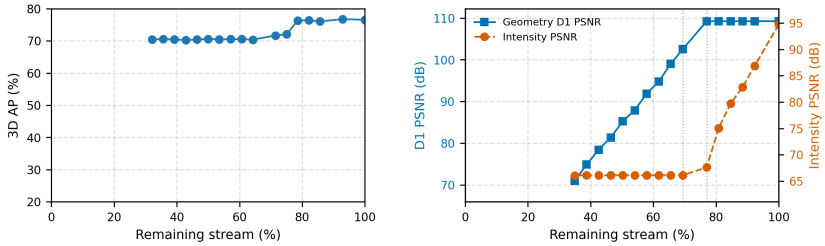


Figure 3: Performance degradation under progressive bitstream truncation. Left: Downstream 3D object detection AP. Right: Geometry and attribute reconstruction quality.

Because SIG and INS are written into independent bitstreams, truncating the INS stream leaves the base geometry intact. The decoding and reconstruction of the remaining INS components proceed as follows.

Range Reconstruction (Precision Degradation). Truncating the $N = d - k_r$ least significant planes is equivalent to zero-padding, yielding the reconstructed range image $\hat{\mathbf{x}} = \mathbf{x}^{1:k_r} \cdot 2^N$. The truncation length maps directly to geometry precision through **A1**. Let Δ_0 be the range precision of the full d-bit representation. Truncating N planes halves the precision N times:

$$\Delta_N = 2^N \cdot \Delta_0. \tag{7}$$

For KITTI range images, $\Delta_0 \approx 1$mm under the 16-bit quantization convention (You et al., 2025), so truncating $N = 1, 2, 3$ planes yields approximately 2mm, 4mm, 8mm precision.

Attribute Reconstruction (Hybrid Lossless-Predictive). Unlike geometry, attributes lack a strict physical-to-scale mapping and can tolerate predictive approximation. For any truncated attribute depth $k_a < d_a$, we reconstruct the remaining bits within the active quantization cell using the lossless prefix $\mathbf{a}^{1:k_a}$ and the range prior $\hat{\mathbf{x}}$:

$$\hat{\mathbf{a}} = \mathbf{a}^{1:k_a} + \delta_{k_a} \cdot \sigma\big(h_\theta(\hat{\mathbf{x}}, \mathbf{a}^{1:k_a}, k_a)\big), \tag{8}$$

where $\delta_{k_a} = 2^{-k_a}$ is the cell width, and $\sigma(\cdot) = 0.5 + 0.5\tanh(\cdot)$ scales the prediction network h_θ's output to $[0, 1]$. The reconstruction overhead and the design choices of h_θ are evaluated in Section 5.3.

Four properties follow. (i) **No prior negotiation**: the sender and receiver need not agree on a truncation point; any INS suffix can be dropped in transit. (ii) **No geometric hallucination**: truncated range planes are not predicted, ensuring the range is reconstructed exactly at the precision the bitstream carries. (iii) **Deterministic degradation**: the geometry precision is $\Delta_N = 2^N \cdot \Delta_0$ regardless of the range content. (iv) **Progressive decoding**: INS's FIFO order allows decoding to halt at the truncation point and still recover every plane preceding it.

4.4 Training Objective

The training proceeds in two stages:

Stage 1: Compression Network Training. We optimize the entropy model parameters using the joint rate-distortion loss under maximum capacity ($k_r = d, k_a = d_a$):

$$\mathcal{L}_{\text{compress}} = -\log p(\mathbf{x}^{h+1:s}) - \text{ELBO}(\mathbf{x}^{1:h}) - R_{\text{INS}}(d, d_a), \tag{9}$$

Stage 2: Prediction Head Training. With the compression network frozen, we train the prediction head h_θ. For each training instance, we sample $k_a \sim \text{Uniform}\{0, \ldots, d_a\}$ to represent the progressive truncation states. The network is optimized via a cell-normalized masked Mean Squared Error (MSE) over valid range pixels ($x > 0$):

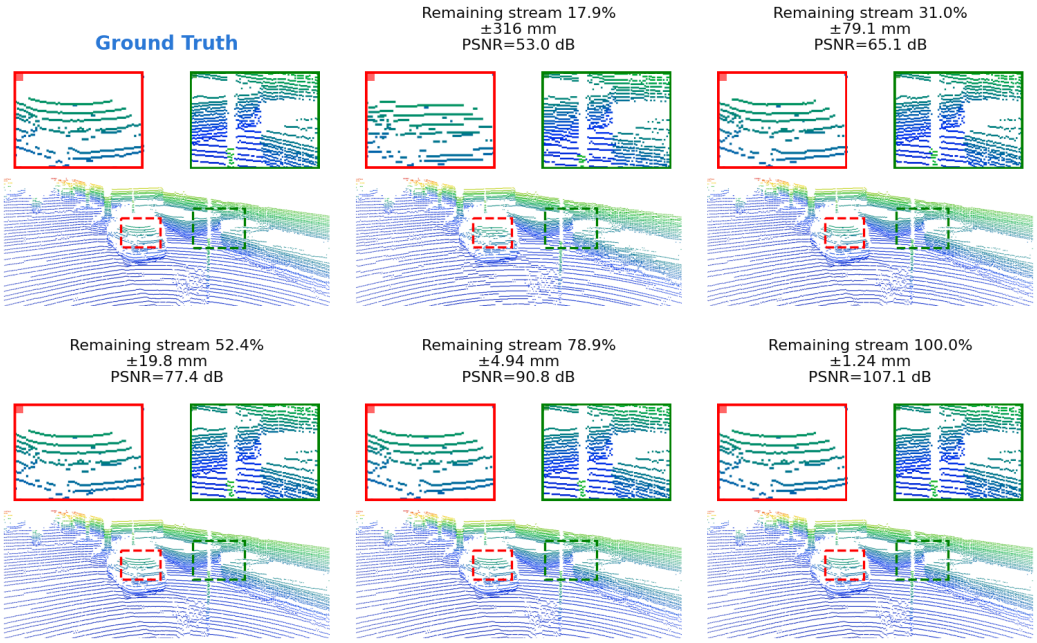


Figure 4: Visualization of reconstructed LiDAR point clouds under progressive bitstream truncation.

$$\mathcal{L}_{\text{rec}} = \frac{1}{N_{\text{valid}}} \sum_{\text{valid}} \left(\frac{\hat{\mathbf{a}} - \mathbf{a}}{\delta_{k_a}} \right)^2, \tag{10}$$

where $\delta_{k_a} = 2^{-k_a}$. Normalizing the residual by the quantization cell width δ_{k_a} scales the errors at different truncation levels to the same relative magnitude, balancing the gradient contributions across all progressive bit-planes.

5 Experiments

5.1 Experimental Setup

Datasets.

We evaluate on the Waymo Open Dataset (WOD) (Sun et al., 2020) and SemanticKITTI (Behley et al., 2019). WOD natively provides raw range images and precise beam emission angles, enabling mathematically lossless transformation between 2D range grids and 3D Cartesian coordinates. Although SemanticKITTI is likewise captured by rotating LiDAR sensors, it only releases Cartesian point clouds without raw beam calibration; following established protocols (Zhou et al., 2022), we evaluate on pseudo range images projected using estimated beam emission angles.

Baselines.

Since no existing learned point cloud compression (PCC) frameworks support self-contained, prefix-decodable progressive coding under truncation, directly comparable baselines in this regime are unavailable. Moreover, although 2D progressive image codecs (e.g., JPEG XL (Sneyers et al., 2025) or JPEG 2000 (Skodras et al., 2001)) natively support truncation, their frequency- or wavelet-domain algorithms do not translate to deterministic, bit-plane-aligned 3D coordinate degradation. We therefore select G-PCC v23 (MPEG, 2023), Google Draco (Google, 2023), JPEG XL (Sneyers et al., 2025), and Unicorn (Wang et al., 2025a;b) to evaluate and verify our compression performance under ideal channel conditions (0% truncation).

Table 1: BD-Rate gains to G-PCC (%) and runtimes (in seconds). For intensity compression, we report the total runtime for coding both geometry and intensity. The best results are marked in bold.

(a) KITTI

	Geometry			Intensity		
Method	BD-Rate	Enc.	Dec.	BD-Rate	Enc.	Dec.
G-PCC	0.00	1.53	0.88	0.00	2.18	1.44
Draco	+20.64	**0.02**	**0.01**	+100.05	**0.03**	**0.02**
JPEG XL	−11.81	0.10	**0.01**	+136.28	0.18	**0.02**
Unicorn	−24.41	6.65	6.01	−5.82	9.79	8.00
Proteus	**−35.41**	0.10	0.08	**−15.59**	0.10	0.11

(b) WOD

	Geometry			Intensity		
Method	BD-Rate	Enc.	Dec.	BD-Rate	Enc.	Dec.
G-PCC	0.00	0.49	0.33	0.00	0.66	0.48
Draco	+35.02	**0.03**	**0.01**	−13.41	**0.05**	**0.02**
JPEG XL	+2.83	0.11	**0.01**	−2.72	0.19	**0.02**
Unicorn			—			
Proteus	**−42.72**	0.14	0.10	**−80.47**	0.14	0.14

Table 2: Ablation studies on (a) entropy coding variants and (b) attribute reconstruction components.

(a) Entropy coding

Variant	Relative Difference	Robustness
w/o bits-back	+20.19%	No
w/o ArIB	+31.17%	No
ArIB (coupled)	−6.56%	No
Proposed	**Ref. (0.00%)**	**Yes**

(b) Attribute reconstruction

Metric	w/o Prediction Head	**Proposed**
BD-Rate	+67.8%	**Ref. (0.00%)**
Attr. Dec. Overhead	+0 ms	**+4.6 ms**
Extra Storage	0	**1.3 MB**

Implementation Details.

We train separate models for WOD and SemanticKITTI. All experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU using the Adam optimizer (Kingma & Ba, 2015) with an initial learning rate of 1×10^{-4} and a step decay schedule. For our model configuration, we set the head depth h to 7 (with mid $= 3$) for SemanticKITTI and 6 (with mid $= 4$) for WOD based on the inflection points of bits per dimension (BPD) savings achieved using latent variables in Figure 2(a,b). To provide a necessary self-contained perceptual lower bound under progressive truncation, the split index s is consistently set to 10 for both datasets based on the PointPillars evaluation (Lang et al., 2019) in Figure 2(c).

5.2 Performance Evaluation

Truncation Robustness under Dynamic Channels.

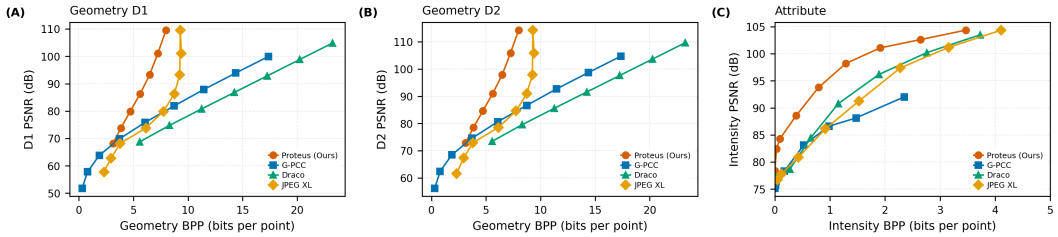
We evaluate the progressive degradation behavior of our codec under dynamic channel conditions by sweeping the truncation ratio of the INS stream from 0% to 100%. Since SIG only accounts for approximately 30% of the overall bitstream, Proteus natively supports a truncation ratio of up to approximately 70%. Under truncation, our framework sacrifices attribute precision first to prioritize geometric preservation, as reflected by the geometry and intensity PSNR trends in Figure 3 (right), while downstream 3D object detection AP degrades progressively down to the self-contained lower bound (Figure 3, left).

Specifically, the geometric error remains mathematically bounded by Equation 7, and Figure 4 presents the visual results of the reconstructed point clouds under varying truncation levels.

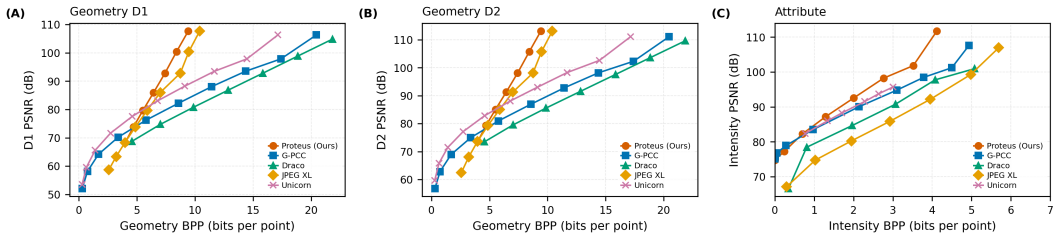
Rate-Distortion Performance under Ideal Channels.

We evaluate the rate-distortion (R-D) performance of Proteus under ideal channel conditions (0% truncation) using BD-Rate (Barman et al., 2024); geometry quality follows the point-to-point metric (Tian et al., 2017).

Geometry Compression. Proteus outperforms other baselines, achieving BD-rates of -35.41% on SemanticKITTI and -42.72% on WOD compared to G-PCC. This demonstrates the efficiency of our entropy coding framework. Additionally, to achieve a coordinate precision of n bits, 3D voxel- or octree-based methods must subdivide the 3D space, scaling their spatial cell representation exponentially as $O(k^n)$ (where k is the spatial dimension factor). This is highly efficient at low



(a) WOD



(b) SemanticKITTI

Figure 5: Rate-distortion performance on WOD (top) and SemanticKITTI (bottom).

precision where sparse, empty space is skipped, but the symbol overhead scales rapidly at higher precision levels. In contrast, range-image-based methods operate on a fixed 2D grid, requiring only additional bit planes that scale the complexity linearly as $O(n)$. Consequently, while 3D methods excel at coarse reconstructions at low bitrates, range-image approaches scale more favorably and outperform them in high-precision regimes.

Attribute Compression. For intensity compression, Proteus outperforms other baselines, achieving BD-rates of -15.59% on SemanticKITTI and -80.47% on WOD compared to G-PCC. The performance difference between the two datasets is primarily driven by variations in dataset distribution, as detailed in Appendix C.

Latency Analysis.

Under full-precision 16-bit geometry and 8-bit attribute coding, Proteus consistently outperforms recent low latency baseline RangeCM (Song et al., 2026) and achieves near-real-time throughput on SemanticKITTI. Comprehensive comparisons and profiling are provided in Appendix E.

5.3 Ablation Studies

We evaluate the entropy coding variants on SemanticKITTI in Table 2(a). Disabling bits-back (w/o bits-back) and ArIB (w/o ArIB) incurs rate increases of +20.19% and +31.17%, respectively; w/o ArIB performs worse in practice due to larger protective padding required by ANS. Crucially, compared to the coupled ArIB scheme, our decoupled framework pays only a minor 6.56% efficiency tax to unlock progressive truncation robustness, which coupled ArIB entirely lacks.

We also evaluate the attribute prediction head (h_θ) in Table 2(b). Removing this module (w/o Prediction Head) incurs a substantial +67.8% BD-rate penalty, whereas our proposed head secures this gain with only 1.3 MB of extra storage and a minor 4.6 ms decoding overhead.

6 Conclusion

This paper presents Proteus, a learned LiDAR codec leveraging ArIB-assisted bits-back coding. By uniting a progressive bit-plane slicing representation with a decoupled bitstream architecture, Proteus guarantees stream-level truncation robustness and deterministic precision degradation without geometric hallucinations. Consequently, Proteus withstands up to 70% bitstream truncation while outperforming established baselines under ideal channels. Finally, we discuss the scope of range representations and their generalization to other sensor modalities in Appendix F.

AI USE STATEMENT

In this work, generative AI tools were used to assist with the implementation of methods designed by the authors. We did not use generative AI tools to develop the theoretical or conceptual framework, formulate mathematical claims, propose or refine research hypotheses, design or provide feedback on the research methodology or experiments, clean or reformat datasets, or interpret experimental results. Tasks related to synthetic data generation, mathematical proofs, translation, and qualitative or thematic data analysis were not applicable to this work.

Additionally, generative AI tools were used for code editing and debugging, improving the readability of the manuscript, and formatting references. All AI-assisted work was reviewed by the authors. AI-assisted code was checked for consistency with the methods and data-processing procedures specified by the authors and tested for correctness. We take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

REFERENCES

3GPP. 3GPP TS 23.287 Version 18.4.0 Release 18: Architecture Enhancements for 5G System (5GS) to Support Vehicle-to-Everything (V2X) Services, 2024a.

3GPP. NR; Physical layer procedures for data. Technical Specification 38.214 V18.4.0, Release 18, 3rd Generation Partnership Project (3GPP), 2024b.

3GPP. NR; NR and NG-RAN Overall description; Stage-2. Technical Specification 38.300 V18.2.0, Release 18, 3rd Generation Partnership Project (3GPP), 2024c.

5G-SMART. Evaluation of radio network deployment options. Deliverable D1.5, Version 1.0, 5G-SMART, 2021. URL `https://5gsmart.eu/wp-content/uploads/5G-SMART-D1.5-v1.0.pdf`.

F. Baker and G. Fairhurst. IETF Recommendations Regarding Active Queue Management. RFC 7567, July 2015.

Nabajeet Barman, Maria G. Martini, and Yuriy Reznik. Bjøntegaard Delta (BD): A Tutorial Overview of the Metric, Evolution, Challenges, and Recommendations. arXiv preprint (arXiv:2401.04039), 2024.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9297–9307, 2019.

Sourav Biswas, Jerry Liu, Kelvin Wong, Shenlong Wang, and Raquel Urtasun. MuSCLE: Multi Sweep Compression of LiDAR using Deep Entropy Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22170–22181, 2020.

Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. Deep Joint Source-Channel Coding for Wireless Image Transmission. *IEEE Transactions on Cognitive Communications and Networking*, 5(3):567–579, 2019.

Anthony Chen, Shiwen Mao, Zhu Li, Minrui Xu, Hongliang Zhang, Dusit Niyato, and Zhu Han. Multiple Description Coding for Point Cloud. In *IEEE International Conference on Communications*, pp. 3408–3413, 2024.

Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y. Yan, Amrita Mazumdar, Nick Feamster, and Junchen Jiang. GRACE: Loss-Resilient Real-Time Video through Neural Codecs. In *21st USENIX Symposium on Networked Systems Design and Implementation*, pp. 509–531. USENIX Association, 2024.

Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. arXiv preprint (arXiv:1311.2540), 2013.

W. Eddy. Transmission Control Protocol (TCP). RFC 9293, August 2022.

Abbas A. El Gamal and Thomas M. Cover. Achievable rates for multiple descriptions. *IEEE Transactions on Information Theory*, 28(6):851–857, 1982.

Ecenaz Erdemir, Tze-Yang Tung, Pier Luigi Dragotti, and Deniz Gündüz. Generative Joint Source-Channel Coding for Semantic Image Transmission. *IEEE Journal on Selected Areas in Communications*, 41(8):2645–2657, 2023.

Tingyu Fan, Linyao Gao, Yiling Xu, Zhu Li, and Dong Wang. D-DPCC: Deep Dynamic Point Cloud Compression via 3D Motion Prediction. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 898–904, 2022.

Guangchi Fang, Qingyong Hu, Hanyun Wang, Yiling Xu, and Yulan Guo. 3DAC: Learning Attribute Compression for Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14799–14808, 2022.

Yu Feng, Shaoshan Liu, and Yuhao Zhu. Real-Time Spatio-Temporal LiDAR Point Cloud Compression. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 10766–10773, 2020.

Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. OctAttention: Octree-Based Large-Scale Contexts Model for Point Cloud Compression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(1):625–633, 2022.

Wei Gao, Liang Xie, Songlin Fan, Ge Li, Shan Liu, and Wen Gao. Deep Learning-Based Point Cloud Compression: An In-Depth Survey and Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(11):10731–10752, 2025.

Google. Draco 1.5.6. Official software release, 2023. URL `https://github.com/google/draco/releases/tag/1.5.6`.

D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing*, 9(1):e13, 2020.

Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021.

Katsuyuki Haneda, Jianhua Zhang, Lei Tian, Guangyi Liu, Yi Zheng, Henrik Asplund, Jian Li, Yi Wang, David Steer, Clara Li, Tommaso Balercia, Sunguk Lee, YoungSeok Kim, Amitava Ghosh, Timothy Thomas, Takehiro Nakamura, Yuichi Kakishima, Tetsuro Imai, Haralabos Papadopoulos, Theodore S. Rappaport, George R. MacCartney, Jr., Mathew K. Samimi, Shu Sun, Ozge Koymen, Sooyoung Hur, Jeongho Park, Charlie Zhang, Evangelos Mellios, Andreas F. Molisch, Saeed S. Ghassemzadeh, and Arun Ghosh. 5G 3GPP-like Channel Models for Outdoor Urban Microcellular and Macrocellular Environments. In *2016 IEEE 83rd Vehicular Technology Conference – Spring*, pp. 1–7, 2016.

Yun He, Xinlin Ren, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Density-Preserving Deep Point Cloud Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2323–2332, 2022.

C. Hornig. A Standard for the Transmission of IP Datagrams over Ethernet Networks. RFC 894, April 1984.

Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. Where2comm: Communication-Efficient Collaborative Perception via Spatial Confidence Maps. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4874–4886, 2022.

Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1310–1320, 2020.

Tzu-Kuan Hung, I-Chun Huang, Samuel Rhys Cox, Wei Tsang Ooi, and Cheng-Hsin Hsu. Error Concealment of Dynamic 3D Point Cloud Streaming. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 3134–3142, 2022.

J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021.

Szymon Jakubczak and Dina Katabi. SoftCast: One-Size-Fits-All Wireless Video. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pp. 449–450, 2010.

Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. Context-Based Trit-Plane Coding for Progressive Image Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14348–14357, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12689–12697, 2019.

Jae-Han Lee, Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. DPICT: Deep Progressive Image Compression Using Trit-Planes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16113–16122, 2022.

Jinlong Li, Runsheng Xu, Xinyu Liu, Jin Ma, Zicheng Chi, Jiaqi Ma, and Hongkai Yu. Learning for Vehicle-to-Vehicle Cooperative Perception under Lossy Communication. *IEEE Transactions on Intelligent Vehicles*, 8(4):2650–2660, 2023.

Yueheng Li, Hao Chen, Bowei Xu, Zicheng Zhang, and Zhan Ma. Improving Adaptive Real-Time Video Communication via Cross-Layer Optimization. *IEEE Transactions on Multimedia*, 26: 5369–5382, 2024.

Bojun Liu, Yangzhi Ma, Li Li, Dong Liu, Zhu Li, and Houqiang Li. Next Bit Prediction: A Unified Lossless and Lossy Point Cloud Geometry Compression Framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(5):5371–5387, 2026.

Chia-Sheng Liu, Jia-Fong Yeh, Hao Hsu, Hung-Ting Su, Ming-Sui Lee, and Winston H. Hsu. BIRD-PCC: Bi-Directional Range Image-Based Deep Lidar Point Cloud Compression. In *2023 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1–5, 2023.

Yadong Lu, Yinhao Zhu, Yang Yang, Amir Said, and Taco S. Cohen. Progressive Neural Image Compression with Nested Quantization and Latent Ordering. In *2021 IEEE International Conference on Image Processing*, pp. 539–543, 2021.

Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3D Object Detection for Autonomous Driving: A Comprehensive Survey. *International Journal of Computer Vision*, 131(8): 1909–1963, 2023.

M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018, October 1996.

Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12677–12686, 2019.

MPEG. G-PCC TMC13 v23.0-rc2. Official G-PCC reference software tag, 2023. URL `https://github.com/MPEGGroup/mpeg-pcc-tmc13/releases/tag/release-v23.0-rc2`.

Zizheng Que, Guo Lu, and Dong Xu. VoxelContext-Net: An Octree based Framework for Point Cloud Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6042–6051, 2021.

Shunli Ren, Zixing Lei, Zi Wang, Mehrdad Dianati, Yafei Wang, Siheng Chen, and Wenjun Zhang. Interruption-Aware Cooperative Perception for V2X Communication-Aided Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 9(4):4698–4714, 2024.

Santiago Royo and Maria Ballesta-Garcia. An Overview of Lidar Imaging Systems for Autonomous Vehicles. *Applied Sciences*, 9(19):4093, 2019.

Tom Ryder, Chen Zhang, Ning Kang, and Shifeng Zhang. Split Hierarchical Variational Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 386–395, 2022.

Jerome M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 Still Image Compression Standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.

Jon Sneyers, Jyrki Alakuijala, Luca Versari, Zoltán Szabadka, Sami Boukortt, Amnon Cohen-Tidhar, Moritz Firsching, Evgenii Kliuchnikov, Tal Lev-Ami, Eric Portis, Thomas Richter, and Osamu Watanabe. The JPEG XL Image Coding System: History, Features, Coding Tools, Design Rationale, and Future. arXiv preprint (arXiv:2506.05987), 2025.

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

Rui Song, Yan Wang, Tongda Xu, Zhening Liu, Zehong Lin, and Jun Zhang. Low-Latency Neural LiDAR Compression with 2D Context Models. In *International Conference on Learning Representations*, 2026.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2443–2451, 2020.

Dong Tian, Hideaki Ochimizu, Chen Feng, Robert A. Cohen, and Anthony Vetro. Geometric Distortion Metrics for Point Cloud Compression. In *2017 IEEE International Conference on Image Processing*, pp. 3460–3464, 2017.

James Townsend, Thomas Bird, and David Barber. Practical Lossless Compression with Latent Variables using Bits Back Coding. In *International Conference on Learning Representations*, 2019.

Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse Tensor-Based Multiscale Representation for Point Cloud Geometry Compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):9055–9071, 2023.

Jianqiang Wang, Ruixiang Xue, Jiaxin Li, Dandan Ding, Yi Lin, and Zhan Ma. A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part II: Attribute. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1):252–268, 2025a.

Jianqiang Wang, Ruixiang Xue, Jiaxin Li, Dandan Ding, Yi Lin, and Zhan Ma. A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part I: Geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1):269–287, 2025b.

Sukai Wang, Jianhao Jiao, Peide Cai, and Lujia Wang. R-PCC: A Baseline for Range Image-based Point Cloud Compression. In *2022 International Conference on Robotics and Automation*, pp. 10055–10061, 2022.

Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction. In *European Conference on Computer Vision*, volume 12347, pp. 605–621. Springer, 2020.

M. Watson, A. Begen, and V. Roca. Forward Error Correction (FEC) Framework. RFC 6363, October 2011.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

Kang You, Tong Chen, Dandan Ding, M. Salman Asif, and Zhan Ma. RENO: Real-Time Neural Compression for 3D LiDAR Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22172–22181, 2025.

Cixiao Zhang, Mufan Liu, Wenjie Huang, Yin Xu, Yiling Xu, and Dazhi He. Deep Joint Source-Channel Coding for Wireless Point Cloud Transmission. In *2025 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1–5, 2025.

Zhe Zhang, Huairui Wang, Zhenzhong Chen, and Shan Liu. Learned Lossless Image Compression Based on Bit Plane Slicing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27569–27578, 2024.

Xuanyu Zhou, Charles R. Qi, Yin Zhou, and Dragomir Anguelov. RIDDLE: Lidar Data Compression with Range Image Deep Delta Encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17191–17200, 2022.

Jiahao Zhu, Kang You, Dandan Ding, and Zhan Ma. Efficient LiDAR Reflectance Compression via Scanning Serialization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 80330–80346. PMLR, 2025.

A Experimental Protocol and Evaluation Details

A.1 Sensor Data Preprocessing and Quantization

To establish a unified and reproducible evaluation benchmark, raw LiDAR measurements are preprocessed and quantized prior to compression as follows:

16-Bit Geometry Quantization. Following standard range-image projection pipelines, radial distance measurements r are normalized by the maximum sensor operating range $R_{\max}$ and mapped onto the discrete 16-bit unsigned integer range $[0, 65535]$:

$$x = \text{clip}\left(\text{round}\left(\frac{r}{R_{\max}} \times 65535\right), 0, 65535\right). \tag{11}$$

For SemanticKITTI, we set $R_{\max} = 81.0\,\text{m}$, yielding a base quantization step size $\Delta_0 = 81.0/65535 \approx 1.24\,\text{mm}$ (where empirical maximum ranges in test sequences remain within $80.06\,\text{m}$, avoiding boundary clipping). For the Waymo Open Dataset (WOD), we set $R_{\max} = 75.0\,\text{m}$, which yields a base resolution $\Delta_0 = 75.0/65535 \approx 1.14\,\text{mm}$.

8-Bit Reflectance Mapping. For SemanticKITTI, raw reflectance values are naturally bounded in $[0, 1]$ and directly scaled to 8-bit integers:

$$a = \text{round}\left(a_{\text{raw}} \times 255\right). \tag{12}$$

In contrast, raw LiDAR intensity in WOD exhibits high dynamic range characteristics, where retroreflective surfaces (e.g., traffic signs, lane markers, and vehicle plates) frequently return calibrated intensities exceeding nominal diffuse bounds ($a_{\text{raw}} > 1.0$). Prior range-image codecs such as RangeCM (Song et al., 2026) enforce a $[0.0, 1.0]$ clip, which causes all retroreflective returns in $[1.0, 2.0]$ to artificially saturate at 255, discarding physical contrast among high-reflectance targets. To preserve these safety-critical cues without saturation clipping, we set an upper clipping threshold of 2.0:

$$a = \text{round}\left(\frac{\text{clip}(a_{\text{raw}}, 0.0, 2.0)}{2.0} \times 255\right). \tag{13}$$

A.2 2D Codec Baseline Protocol: JPEG XL Adaptation

To establish a rigorous 2D baseline, we configure JPEG XL by adhering to its standardized architecture across range representations. In range images, missing-beam zero regions ($r = 0$) and object silhouettes represent discrete step discontinuities rather than continuous-tone gradients. According to the ISO/IEC 18181-1 JPEG XL specification, continuous-tone transform coding (`VarDCT`) is designed for natural photographic imagery, whereas non-continuous integer fields (such as auxiliary depth channels and discrete masks) are designated to operate under the **Modular Mode**. In practice, applying `VarDCT` to range images triggers severe frequency-domain Gibbs ringing across sharp boundaries, which project into 3D space as high-frequency geometric outliers ("flying noise") that catastrophically degrade point-to-point PSNR.

Because Modular Mode functions as an integer predictive tree coder without continuous frequency-domain quality parameters, rate-distortion (R-D) points cannot be steered via continuous quality sliders. Following standard scalar quantization principles, we sweep target bitrates by applying deterministic bit-plane shift truncation prior to lossless modular compression:

1. **Deterministic Shift Quantization:** To sweep target rate-distortion points monotonically, the input 16-bit range image is uniformly quantized by retaining only the highest k bit-planes:
$$x_{\text{quant}} = (x \gg (16 - k)) \ll (16 - k), \quad k \in \{1, \ldots, 16\}. \tag{14}$$
2. **Lossless Modular Encoding:** The quantized array is subsequently compressed using JPEG XL's native integer-tree Modular engine (`lossless=True, effort=7`).

This configuration aligns with the official standard for range and depth fields, completely suppresses transform ringing artifacts, and provides an authoritative rate-distortion upper bound for 2D codecs under ideal channel conditions.

A.3 Evaluation Metrics and Quality Assessment

Following established conventions in point cloud compression (Tian et al., 2017; Biswas et al., 2020; Song et al., 2026; Wang et al., 2025a;b), all objective reconstruction metrics are evaluated using the official MPEG point cloud error tool `pc_error_d` under LiDAR evaluation mode (`-lidar=1`).

Geometric Reconstruction Quality. Geometry reconstruction is evaluated via symmetric Point-to-Point (D1) and Point-to-Plane (D2) Peak Signal-to-Noise Ratios:

$$\text{PSNR}_{\text{geo}} = 10 \log_{10} \left(\frac{3r^2}{\text{MSE}} \right), \tag{15}$$

where MSE denotes the corresponding symmetric D1 or D2 Euclidean mean squared error, and r is the dataset-specific geometric peak value defined as the maximum nearest-neighbor distance across the entire evaluation sequence (Tian et al., 2017):

$$r = \max_{P \in \mathcal{D}} \left(\max_{\mathbf{p}_i \in P} \min_{\mathbf{p}_j \in P, j \neq i} \|\mathbf{p}_i - \mathbf{p}_j\|_2 \right). \tag{16}$$

Following previous work (Song et al., 2026; Biswas et al., 2020), we set $r = 59.70$ for SemanticKITTI and $r = 57.41$ for the Waymo Open Dataset.

Reflectance and Evaluation Fairness. Reflectance PSNR is reported directly by `pc_error_d`. All codecs share identical evaluation scripts and PLY serialization; thus, downstream BD-Rate calculations remain strictly invariant to constant metric offsets, ensuring fair comparison.

B Proteus Architecture and Decoupled Bits-Back State Machine

Proteus implements a 2-layer hierarchical Variational Autoencoder to execute the Decoupled Bits-Back coding mechanism within the significant sub-block (SIG). While Proteus retains the 2D convolutional layouts of RangeCM's feature extraction modules (g_a, h_a, h_s, g_s) to maintain structural efficiency (Song et al., 2026), their functional roles are repurposed: rather than performing direct spatial feature transforms, these modules parameterize the location parameters of discrete hierarchical distributions over binary bit-planes.

B.1 Parameter Prediction Networks (Param Nets)

To map intermediate continuous representations into discrete probability spaces for bit-plane coding, we introduce three lightweight parameter prediction networks (heads) built with 2D convolutional layers:

1. **q_head_z:** A posterior parameter network that maps the hyper-latent feature space z_{features} to the top-level location parameter μ_{q_z}:
$$\mu_{q_z} = \text{q_head_z}(z_{\text{features}}) \in \mathbb{R}^{B \times z_{\text{channels}} \times H/8 \times W/128}. \tag{17}$$
2. **q_head_y:** A joint posterior parameter network that concatenates the bottom-up feature extraction y_{features} and the top-down reconstructed feature space $h_{s_{\text{out}}}$ to generate the bottom-level location parameter μ_{q_y}:
$$\mu_{q_y} = \text{q_head_y}\big([h_{s_{\text{out}}}, y_{\text{features}}]\big) \in \mathbb{R}^{B \times y_{\text{dim}} \times H/4 \times W/64}. \tag{18}$$
3. **p_head_y:** A prior parameter network that maps the top-down features $h_{s_{\text{out}}}$ alone to the prior location parameter μ_{p_y} for bottom-level decoding:
$$\mu_{p_y} = \text{p_head_y}(h_{s_{\text{out}}}) \in \mathbb{R}^{B \times y_{\text{dim}} \times H/4 \times W/64}. \tag{19}$$

B.2 Hierarchical Distribution Parameterization

The hierarchical model comprises a top-level hyper-latent z and a bottom-level main latent y. Both latents are modeled using discrete Logistic distributions with a unit scale parameter.

LAYER z (TOP LEVEL)

- **Posterior** $q(z|\mathbf{x}^{1:h})$**:** where μ_{q_z} is predicted using q_head_z, and $z_{\text{features}} = h_a(g_a(\mathbf{x}^{1:h}))$.

$$q(z|\mathbf{x}^{1:h}) = \text{Logistic}(\mu_{q_z}, 1). \tag{20}$$

- **Prior** $p(z)$**:** where the location parameter is a static zero tensor ($\mu_{p_z} = \mathbf{0}$). This design standardizes the hyperprior space, eliminating the need for hyperprior projection layers.

$$p(z) = \text{Logistic}(0, 1). \tag{21}$$

LAYER y (BOTTOM LEVEL)

- **Posterior** $q(y|\mathbf{x}^{1:h}, z)$**:** where μ_{q_y} is generated using q_head_y with the concatenated features, and $h_{s_{\text{out}}} = h_s(z)$.

$$q(y|\mathbf{x}^{1:h}, z) = \text{Logistic}(\mu_{q_y}, 1). \tag{22}$$

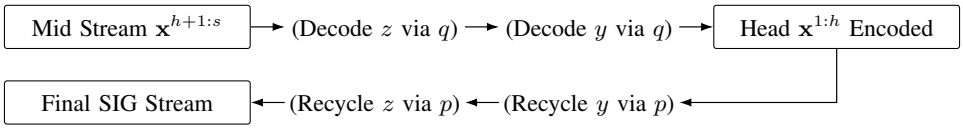
- **Prior** $p(y|z)$**:** where μ_{p_y} is computed from the top-down propagation using p_head_y.

$$p(y|z) = \text{Logistic}(\mu_{p_y}, 1). \tag{23}$$

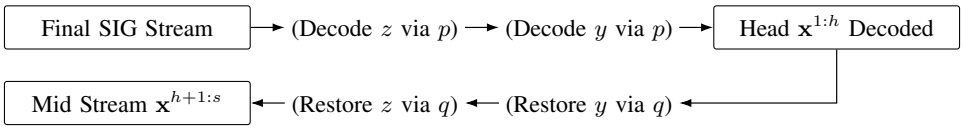
B.3 BITS-BACK STATE MACHINE FLOW

Due to the Last-In-First-Out (LIFO) stack property of Asymmetric Numerical Systems (ANS), the compression and decompression routines must execute "decode" and "encode" operations in strictly reversed order (Duda, 2013; Townsend et al., 2019). In contrast to prior coupled frameworks, Proteus executes a strictly self-contained bits-back state machine within SIG: the mid sub-block $\mathbf{x}^{h+1:s}$ is encoded first under an unconditional prior $p(\mathbf{x}^{h+1:s})$ to internally populate the ANS stack with the required initial bits, fully decoupling SIG from INS.

Encoder (SIG Compression)



Decoder (SIG Decompression)



COMPRESSION PIPELINE

The encoding sequence within the SIG block is mathematically defined as follows:

1. **Internal State Initialization:** Encode the mid sub-block $\mathbf{x}^{h+1:s}$ using the unconditional distribution model $p(\mathbf{x}^{h+1:s})$. The resulting bitstream initializes the ANS coder (MixEncoder) with internal non-empty initial bits.
2. **Bottom-Up Feature Extraction:** Extract latent feature representations from the head sub-block: $y_{\text{features}} = g_a(\mathbf{x}^{1:h})$ and $z_{\text{features}} = h_a(y_{\text{features}})$.
3. **Decapsulate** z **(Bits-Back Decode):** Predict the posterior location parameter $\mu_{q_z} = \text{q_head_z}(z_{\text{features}})$. **Decode** z_{ids} from the active ANS stack using the CDF of $q(z|\mathbf{x}^{1:h})$. Reconstruct the latent representation as $z = \mu_{p_z} + \delta_z$, where δ_z is mapped from z_{ids} and $\mu_{p_z} = \mathbf{0}$:

$$z_{\text{ids}} \sim q(z|\mathbf{x}^{1:h}). \tag{24}$$

4. **Top-Down Propagation:** Generate top-down conditioning features $h_{s_{\text{out}}} = h_s(z)$.

5. **Decapsulate** y **(Bits-Back Decode):** Predict $\mu_{q_y} = \text{q_head_y}([h_{s_{\text{out}}}, y_{\text{features}}])$ and $\mu_{p_y} = \text{p_head_y}(h_{s_{\text{out}}})$. **Decode** y_{ids} from the ANS stack using the CDF of $q(y|\mathbf{x}^{1:h}, z)$. Reconstruct the main latent as $y = \mu_{p_y} + \delta_y$, where δ_y is mapped from y_{ids}:
$$y_{\text{ids}} \sim q(y|\mathbf{x}^{1:h}, z). \tag{25}$$
6. **Head Synthesis and Encoding:** Synthesize $\hat{y} = y + h_{s_{\text{out}}}$ and generate the conditional probability parameters for the head sub-block via $g_s(\hat{y})$. Encode the head planes $\mathbf{x}^{1:h}$ into the stream.
7. **Recycle Latents (Prior Encode):** To satisfy LIFO, **encode** the latent variables back into the ANS stack in reverse order ($y \to z$) using their respective priors:
   - First, encode the main latent y_{ids} using the prior $p(y|z)$, which corresponds to encoding the residual $y - \mu_{p_y}$ under $\text{Logistic}(0, 1)$.
   - Second, encode the hyper-latent z_{ids} using the prior $p(z)$, which reduces to encoding z under $\text{Logistic}(0, 1)$.

DECOMPRESSION PIPELINE

At the receiver, SIG decoding reverses the execution order:

1. **Latent Decompress (_decompress_pz):** Since z was encoded last, it resides at the top of the received bitstream stack.
   - **Decode** z_{ids} from the stream using the prior $p(z) \sim \text{Logistic}(0, 1)$. Reconstruct $z = \mathbf{0} + \delta_z$.
   - Propagate z deterministically to compute $h_{s_{\text{out}}} = h_s(z)$ and calculate the prior location parameter $\mu_{p_y} = \text{p_head_y}(h_{s_{\text{out}}})$.
   - **Decode** y_{ids} using the prior $p(y|z)$ (reconstructed from the residual under $\text{Logistic}(0, 1)$ offset by μ_{p_y}). Reconstruct $y = \mu_{p_y} + \delta_y$.
   - Formulate $\hat{y} = y + h_{s_{\text{out}}}$ and generate the conditional parameters for the head sub-block via $g_s(\hat{y})$.
2. **Decode Head Sub-Block:** Decode the head planes $\mathbf{x}^{1:h}$ using the generated conditional parameters.
3. **Restore Stack State (_compress_qz):** To restore the ANS stack to its state prior to Step 1, re-encode the latent variables in LIFO order ($y \to z$):
   - Pass the decoded head planes $\mathbf{x}^{1:h}$ through g_a and h_a to re-extract y_{features} and z_{features}.
   - Compute the posterior parameter μ_{q_y}. **Encode** y_{ids} into the ANS stack using the posterior CDF of $q(y|\mathbf{x}^{1:h}, z)$.
   - Compute the posterior parameter μ_{q_z}. **Encode** z_{ids} into the ANS stack using the posterior CDF of $q(z|\mathbf{x}^{1:h})$.
4. **Decode Mid Sub-Block:** The ANS stack has now been returned exactly to the state after mid-subblock encoding. Decode the mid planes $\mathbf{x}^{h+1:s}$ using the unconditional model $p(\mathbf{x}^{h+1:s})$.

C ANALYSIS OF DATASET INTENSITY DISTRIBUTION AND BIT-PLANE SLICING EFFICIENCY

The rate-distortion (R-D) efficiency of bit-plane slicing is determined by the probability density function (PDF) of the raw reflectance intensity $S \in [0, 255]$. We formalize this relationship using information-theoretic principles to demonstrate why bit-plane representation is highly suitable for WOD but less optimal for SemanticKITTI.

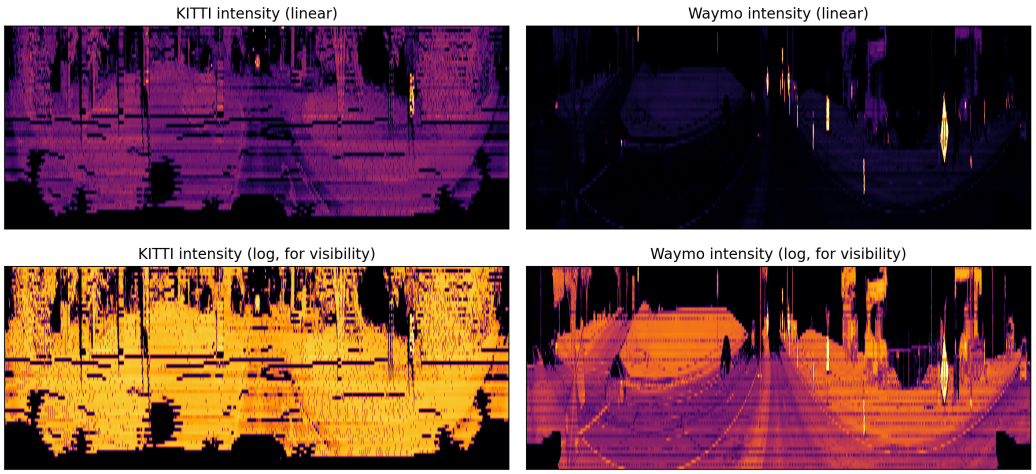


Figure C.1: Reflectance intensity range images under linear (top) and logarithmic (bottom) scales. Left: SemanticKITTI. Right: WOD.

C.1 Mathematical Formulation of the Bit-Plane Representation

Let $f(S)$ denote the continuous probability density function of the intensity. Bit-plane slicing decomposes the scalar value S into a sequence of binary values $x^k \in \{0, 1\}$ for $k \in [1, d]$, where $d = 8$. The binarization mapping for the k-th bit plane is defined as:

$$x^k = \left\lfloor \frac{S}{2^{d-k}} \right\rfloor \pmod 2. \tag{26}$$

For the most significant bit-plane (MSB-0, where $k = 1$), this mapping simplifies to an indicator function:

$$x^1 = \mathbb{I}(S \geq 128) = \begin{cases} 1 & \text{if } S \in [128, 255], \\ 0 & \text{if } S \in [0, 127]. \end{cases} \tag{27}$$

The fill rate p_k of the k-th bit-plane is the probability $\mathbb{P}(x^k = 1)$. For $k = 1$, the fill rate is determined by the cumulative density over the upper half of the intensity range:

$$p_1 = \int_{128}^{255} f(S)\, dS. \tag{28}$$

The zero-order marginal entropy of this bit-plane is:

$$H(p_k) = -p_k \log_2 p_k - (1 - p_k) \log_2 (1 - p_k). \tag{29}$$

C.2 Distortion and Sparsity Properties in WOD

As illustrated in the intensity maps and histograms (Figures C.1 and C.2), WOD's intensity distribution is highly skewed and sparse, featuring a sharp peak at zero and a heavy, power-law tail. The PDF can be approximated as a mixture model:

$$f_{\text{WOD}}(S) = (1 - \epsilon)\delta(S) + \epsilon g(S), \tag{30}$$

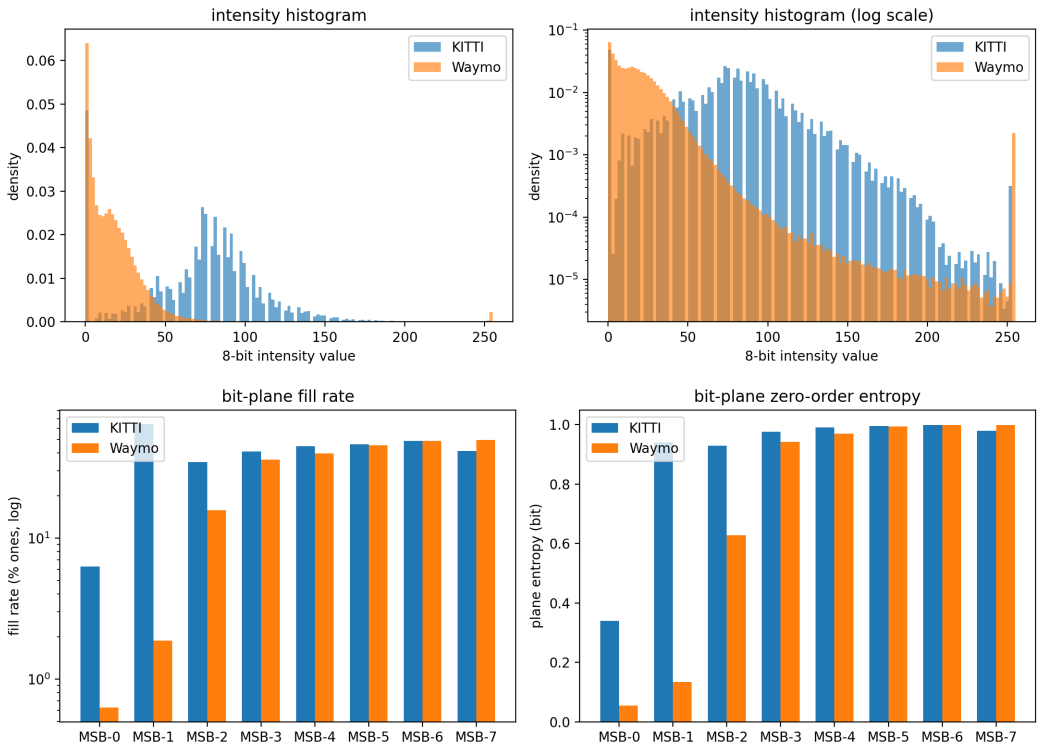


Figure C.2: Reflectance intensity and bit-plane statistics on SemanticKITTI and WOD. Top: Intensity histograms on linear (left) and logarithmic (right) scales. Bottom: Bit-plane fill rates (left) and zero-order entropy (right) from MSB-0 to MSB-7.

where $\delta(S)$ is the Dirac delta function representing low-intensity background surfaces, $\epsilon \approx 0.02$ represents the ratio of retroreflective objects, and $g(S)$ is a high-magnitude probability distribution spanning up to $S = 255$.

On WOD, evaluating the integral yields a very low fill rate for MSB-0 ($p_1 \approx 0.63\%$). According to the entropy formulation, the marginal coding cost is:

$$H_{\text{WOD}}(p_1) \approx 0.055 \text{ bits/pixel.} \tag{31}$$

However, this plane dictates the presence of the sparse, high-magnitude spikes that dominate the overall Mean Squared Error (MSE). Under an additive distortion model, the expected distortion D is dominated by these uncompressed outliers if MSB-0 is truncated:

$$D = \mathbb{E}[(S - \hat{S})^2] = \int_0^{255} (S - \hat{S})^2 f(S)\, dS. \tag{32}$$

Because S_i^2 is large for retroreflective pixels, omitting x^1 forces $\hat{S} \leq 127$, causing the squared residual $(S - \hat{S})^2$ to be exceptionally large. By coding MSB-0, these high-magnitude outliers are localized to their correct coarse cell ($S \geq 128$), reducing the expected distortion D drastically. Consequently, on WOD, **transmitting MSB-0 yields an extremely high R-D efficiency (approximately** 90 **dB/bpp)** because the information-theoretic cost $H(p_1)$ is near zero.

C.3 Continuous Density Properties in SemanticKITTI

In contrast, SemanticKITTI features a dense, continuous, mid-range distribution (Figures C.1 and C.2). Its PDF $f_{\text{KITTI}}(S)$ is characterized by a broad gaussian-like hump centered near $S \approx 85$. Evaluating the integral yields $p_1 \approx 6.0\%$, which results in a significantly higher marginal entropy:

$$H_{\text{KITTI}}(p_1) \approx 0.34 \text{ bits/pixel.} \tag{33}$$

For the second bit-plane (MSB-1), the fill rate on SemanticKITTI approaches $p_2 \approx 50\%$, which maximizes the entropy $H(p_2) \approx 1.0$ bit/pixel. Because the intensity field is spatially continuous and lacks extreme isolated spikes, the marginal distortion reduction of coding these high-significance planes is relatively small (1.0 dB for 0.24 bpp on the first plane).

These mathematical properties explain the attribute compression behaviors observed in our experiments:

1. **WOD** natively benefits from bit-plane slicing, as the sparse, high-magnitude intensity spikes are encoded at a negligible bitrate cost.
2. **SemanticKITTI**'s continuous distribution is less suited for direct bit-plane coding, and its compression efficiency relies on the decoder-side conditional-mean reconstruction to model spatial correlations.

D Physical and Protocol-Level Justification of Channel Impairment Modeling

In mobile wireless communications (*e.g.*, 3GPP 5G-NR V2X, IEEE 802.11bd), raw radio channels exhibit complex physical impairments, including multipath fading, Doppler shifts, and variable shadowing (Haneda et al., 2016). However, under modern layered communication protocol stacks, these physical phenomena do not propagate to the transport interface as arbitrary stochastic bit corruptions. Instead, the combination of physical-layer coding, data-link recovery mechanisms, and transport-layer delivery contracts inherently collapses channel impairments into two canonical operational regimes: **transient link outages** and **bandwidth fluctuation (manifesting as contiguous prefix truncation)**. Below, we mathematically and systematically formalize why this abstraction strictly holds.

D.1 Transient Link Outages (Macro-Scale Disconnection)

Digital radio receivers rely on a minimum Signal-to-Interference-plus-Noise Ratio (SINR) threshold $\gamma_{\min}$ determined by the lowest available Modulation and Coding Scheme (MCS) (3GPP, 2024b) (*e.g.*, BPSK or QPSK rate-1/3 in 3GPP specifications). When the instantaneous received SINR $\gamma(t)$ drops below $\gamma_{\min}$, the physical baseband fails to perform carrier synchronization, channel estimation, and pilot decoding. Consequently, the instantaneous channel transmission rate step-wise collapses to zero:

$$R(t) = 0, \quad \forall t \text{ where } \gamma(t) < \gamma_{\min}. \tag{34}$$

In mobile environments, this condition is driven by two primary physical mechanisms:

1. **Dynamic Line-of-Sight (LOS) Blockage and Severe Shadowing:** Large surrounding obstacles (*e.g.*, high-profile transit vehicles, overpasses, or urban geometries) can abruptly occlude direct propagation paths. This blockage introduces sudden shadow fading drops exceeding 20 dB to 30 dB (Haneda et al., 2016), far outpacing the tracking capability of transmit power control (TPC) algorithms.
2. **Beam Misalignment and Handover Discontinuity:** In directional or high-frequency communications (*e.g.*, FR2/mmWave), rapid angular displacement causes directional beam tracking failure. Re-establishing link alignment via beam sweeping, as well as Radio Resource Control (RRC) handover between base stations, incurs an unavoidable connection gap typically spanning tens of milliseconds (5G-SMART, 2021).

During such periods, the channel enters an absolute erasure state. Transport blocks transmitted across this blackout window cannot be decoded, causing link-layer retransmission budgets to rapidly exhaust. At the socket delivery interface, this macro-scale physical blockage is observed as a clean, complete transmission void—a **transient link outage** where entire data bursts are lost.

D.2 Bandwidth Fluctuation and In-Transit Prefix Truncation

When the channel remains connected ($\gamma(t) \geq \gamma_{\min}$), transmission operates in the active communication regime. In this state, rate adaptation and network queuing naturally map channel capacity drops into tail truncation, while link- and transport-layer contracts prevent mid-stream fragmentation.

Time-Varying Integral Capacity and Drop-Tail Queuing

Sensors generate structured data bursts of size S at quasi-periodic service intervals τ. Within this transmission interval, small-scale multipath interference causes rapid fluctuations in $\gamma(t)$. To maintain a low target Block Error Rate (BLER $\leq 10^{-1}$), the physical layer continuously adjusts the modulation order and code rate via Adaptive Modulation and Coding (AMC) (3GPP, 2024b). The total payload volume that the physical channel can successfully deliver during τ is strictly governed by the time integral of the instantaneous service rate $R(t)$:

$$C = \int_0^\tau R(t)\, dt. \tag{35}$$

If channel conditions degrade such that $C < S$, the data ingress rate surpasses the channel egress service rate, creating persistent queue backlog at intermediate routers or interface transmission buffers. To guarantee queue stability and prevent unbounded queuing latency (bufferbloat), standard network buffer policies (*e.g.*, bounded ring buffers or Active Queue Management (AQM) such as CoDel) enforce **tail-drop** on unserved packets (Baker & Fairhurst, 2015).

Consequently, capacity shortages under time-varying channels physically manifest as the drop of trailing unserved data segments, rather than arbitrary in-transit bit corruptions.

Mid-Stream Erasure Protection and the In-Order Delivery Contract

A critical question is whether channel packet loss could introduce intermediate gaps ("holes") in the delivered bitstream, wasting prior bandwidth. Modern protocol stacks resolve this through a multi-tier defense that guarantees continuous prefix delivery:

1. **Active Multi-Tier Loss Recovery:** Modern protocols strongly prioritize preserving in-flight data efficiency. At Layer 2 (MAC/RLC), Selective Repeat ARQ and Hybrid ARQ (HARQ) request targeted retransmissions strictly for corrupted protocol data units (PDUs) within milliseconds, while holding correctly received out-of-order PDUs in baseband reassembly buffers (3GPP, 2024c). At Layer 4, Selective Acknowledgment (SACK in TCP/QUIC) (Mathis et al., 1996; Iyengar & Thomson, 2021) and packet-level erasure forward error correction (FEC) (Watson et al., 2011) allow the sender to pinpoint and repair isolated missing packets without resending subsequent bytes.
2. **The In-Order Delivery Contract:** In cases where channel degradation is so severe that local retransmission budgets are exhausted before an intermediate gap can be healed, the transport layer's **in-order byte-stream contract** governs data delivery (Eddy, 2022; Iyengar & Thomson, 2021). Transport sockets (*e.g.*, standard stream-oriented TCP, QUIC, or ordered RTP profiles) strictly forbid delivering disjoint, non-contiguous memory chunks to the user space. When an unrecoverable packet loss occurs and the stream terminates, any out-of-order bytes received after the missing segment are flushed by the transport kernel buffer.

As a result, the maximum payload accessible by the receiver through the standard stream read interface terminates precisely at the byte preceding the first unrecovered packet. Thus, intermediate packet losses, if not seamlessly healed by lower-layer ARQ, converge identically to a **prefix truncation**.

D.3 Packet Granularity (MTU) vs. Continuous Bitstream Truncation

In physical network implementations, data is transmitted across the channel in discrete packets bounded by the Maximum Transmission Unit (MTU), typically 1500 bytes for standard Ethernet (Hornig, 1984) and cellular encapsulation. Consequently, physical stream truncation occurs in discrete steps of one MTU rather than infinitesimal bit increments.

For typical single-frame high-throughput data payloads spanning 500 KB to 600 KB, an individual burst is partitioned across approximately 330 to 400 independent network packets. The effective resolution of a single packet-level drop-tail event therefore corresponds to a step size of roughly 0.25% to 0.3% of the total stream.

Analogous to standard discrete floating-point representations capturing continuous real-valued functions, this sub-0.3% granularity ensures that discrete packet-level truncation behaves as a statistically and functionally smooth continuum across the $[0\%, 100\%]$ operating range. Modeling bitstream truncation as a continuous prefix cut is therefore a mathematically rigorous and engineering-faithful characterization of physical network behavior.

E Real-Time Benchmarking and Runtime Analysis

E.1 Benchmarking Low-Latency Codecs under Full Precision and Modality

Real-time processing in autonomous driving is typically bounded by the sensor frame rate (typically 10 Hz, corresponding to a 100 ms per-frame budget). However, evaluating "real-time" or "low-latency" performance in LiDAR compression requires standardized consideration across two foundational dimensions:

1. **Deterministic Coordinate Precision:** Coarser coordinate quantization (e.g., 12-bit to 14-bit, corresponding to precision thresholds of ± 3.2 cm to ± 8 mm) simplifies entropy and voxel depth, greatly reducing coding complexity. In safety-critical perception, however, active safety systems increasingly demand sub-centimeter or millimeter-level physical precision (16-bit quantization, $\Delta_0 \approx 1$ mm) to delineate freespace boundaries and minute road hazards faithfully.
2. **Multi-Modal Completeness:** A complete point cloud scan encompasses both 3D geometry and auxiliary attributes (e.g., 8-bit reflectance intensity). Evaluating geometry alone provides an incomplete picture of operational latency, as it bypasses the non-trivial computational and entropy demands of attribute compression. Benchmarking geometry and reflectance jointly is essential to reflect realistic deployment constraints.

To establish a transparent and equitable benchmark, we compare Proteus with two representative state-of-the-art codecs designed for low latency: RangeCM (Song et al., 2026) and RENO (You et al., 2025). Since runtime measurements are inherently susceptible to hardware platforms, runtime environments, and driver stacks, to guarantee fairness and eliminate discrepancies arising from secondary hardware differences, **we directly report the official runtimes claimed in their original publications**.

Specifically, Proteus, RENO (You et al., 2025), and Unicorn (Wang et al., 2025b;a) are evaluated on an NVIDIA GeForce RTX 3090 GPU, while RangeCM (Song et al., 2026) is evaluated on an NVIDIA RTX A6000 GPU. As discussed by Song et al. (2026), benchmarking on such modern GPUs serves as a realistic and valid proxy for automated driving deployment, since emerging in-vehicle computing platforms (such as the NVIDIA Drive AGX Thor, offering over 1000 TOPS with dedicated hardware sensor I/O) match or exceed the computing power of the RTX 3090.

As summarized in Table 3, evaluating codecs across standardized configurations reveals significant operational distinctions:

Comparison with RangeCM. RangeCM introduces an effective hybrid 2D context model evaluated at 0.01 m precision. When performing joint geometry and reflectance compression, its sequential context modeling incurs an end-to-end decoding latency of 0.170 s on SemanticKITTI and 0.270 s on WOD. In comparison, operating under full 16-bit precision, Proteus improves execution

Table 3: Comprehensive latency comparison against representative low-latency LiDAR codecs. Baselines are evaluated under their respective optimal setups, with runtimes directly cited from their original publications. ("—" denotes unsupported modality or unreported benchmark).

			SemanticKITTI (s)		WOD (s)	
Method	Precision	Modality	Encoding	Decoding	Encoding	Decoding
Geometry Only						
RENO (You et al., 2025)	14-bit (±8 mm)	Geometry	0.095	0.090	—	—
RangeCM-G (Song et al., 2026)	12–13-bit (±10 mm)	Geometry	**0.090**	0.140	0.200	0.200
Proteus (Ours)	16-bit (1 mm)	Geometry	0.098	**0.077**	**0.136**	**0.102**
Geometry + Attribute						
RangeCM-GI (Song et al., 2026)	12–13-bit (±10 mm)	Geometry + Intensity	0.100	0.170	0.220	0.270
Proteus (Ours)	16-bit (1 mm)	Geometry + Intensity	**0.098**	**0.106**	**0.136**	**0.141**

efficiency across both tasks: on SemanticKITTI, Proteus reduces decoding latency from 0.170 s to 0.106 s (a 37.6% speedup); on the large-scale WOD, Proteus reduces encoding from 0.220 s to 0.136 s (38.2% faster) and decoding from 0.270 s to 0.141 s (47.8% faster). In the geometry-only setting, Proteus decodes SemanticKITTI in 0.077 s compared to RangeCM-G's 0.140 s (45.0% faster).

Comparison with RENO. RENO achieves impressive sub-100 ms latency on SemanticKITTI by leveraging sparse convolutional backbones for 14-bit geometry. When evaluated under the corresponding geometry-only setting, Proteus supports full 16-bit precision while matching RENO's encoding throughput (0.098 s vs. 0.095 s) and delivering a 14.4% faster decoding speed (0.077 s vs. 0.090 s).

E.2 Algorithmic and Implementation Optimizations

To attain practical real-time throughput without altering model parameters or compromising lossless reproducibility, we systematically redesign the execution pipeline and entropy micro-kernels. These enhancements are grouped into three primary architectural tiers:

1. Entropy Coding Micro-Kernel Vectorization and Branch Elimination. Because autoregressive bit-plane modeling generates millions of binary symbols per frame, the entropy coder represents a prominent computational hotspot. We re-engineer the inner loops of both Asymmetric Numeral Systems (ANS) and Range Coding at the register and instruction level:

- **Branch-Free Single-Multiply Range Decoding:** Standard FIFO range decoders incur severe branch misprediction penalties and integer division overheads in their renormalization loops. Recognizing that the normalized binary probability distribution satisfies $p_1 + (256 - p_1) = 256$, we reformulate the state update using a branch-free selection primitive and single-multiply arithmetic: $(r \ll 8) -$ bound. This eliminates integer division entirely, reducing range decoding latency from 0.264 ms to 0.188 ms per 65k-symbol block (a 28.8% reduction, operating at ≈ 6.9 cycles/symbol).
- **Register-Mapped rANS with Pad-Mask Pre-fetching:** For the bidirectional rANS coder used in bits-back coding (MixDecoder), each decoded symbol previously incurred structural lookup overhead. We align decoding elements to 4-byte boundaries, utilize 32-bit register storage, and implement an active row-cache for likelihood parameter IDs (`lpID`). Furthermore, by pre-filling renormalization buffers with $(1 \ll nb) - 1$ masks, we eliminate conditional refill branches in hot loops, lowering rANS decoding latency to ≈ 3.85 ns/symbol.
- **Elimination of Structural Copy Overhead in ANS Encoding:** We refactor the ANS state machine (`MixEncoder::encodeBin`) by replacing per-symbol 264-byte structural stack copies with zero-copy index references, yielding an immediate $\approx 3\times$ throughput improvement in bits-back initialization.

2. Execution Pipeline Scheduling and Kernel Fusion. In conventional neural codecs, interleaving deep network inference with entropy coding causes severe host-device synchronization bubbles:

- **Asymmetric One-Shot Probability Staging (Encoding):** Unlike the decoder, which is inherently constrained by causal autoregressive dependencies, the encoder has simultaneous access to all ground-truth bit-planes. We exploit this asymmetry by computing the conditional probability tensor p_1 for all significant bit-planes in a single batched GPU forward pass (one-shot staging). The pre-computed distributions are subsequently consumed sequentially by the CPU entropy coder, entirely eliminating dozens of fine-grained GPU-CPU synchronization rounds and reducing encoding latency by 2–9 ms per frame.
- **Custom CUDA Kernel Fusion for Autoregressive Steps:** Within the autoregressive loop, preparing bit-plane priors requires element-wise bit extraction, zero-masking, and floating-point probability normalization. We fuse these disparate operations (e.g., `lsp_is0` and parameter mapping) into monolithic CUDA kernels. This coalesces memory accesses and eliminates over 280 kernel launch calls per frame, mitigating host dispatch latency.
- **Static Spatial Context Caching:** For the initial autoregressive step where spatial context planes are all-zero, we pre-compute and cache the corresponding 2D convolutional filter responses in device memory, bypassing redundant tensor allocations during execution.

3. Zero-Copy Dataflow and Memory Hierarchy Optimization. Frequent tensor transfers between host RAM and GPU VRAM compound pipeline stalls:

- **End-to-End Native `uint8` Representation:** All intermediate likelihood indexes (`lpID`) and discrete probability maps (p_1) are maintained strictly in 8-bit unsigned integer precision throughout both C++ and CUDA pipelines. This halves PCIe transfer volume compared to standard FP16/FP32 implementations and bypasses intermediate vector staging allocations.
- **Pinned-Buffer Blocking Memory Transfers:** We replace asynchronous non-blocking transfers—which introduce thread-safety hazards and driver synchronization locks under high-frequency dispatch—with statically allocated pinned-memory staging buffers. This maximizes PCIe burst efficiency and reduces device-to-host transfer overhead by $\approx 30\%$.
- **Device-Resident Frequency Tables:** Static cumulative frequency tables and prior distribution constants are persisted directly in GPU constant/device memory, entirely avoiding repetitive pageable host-to-device transfers across consecutive LiDAR sweeps.

E.3 Pipeline Stage Breakdown and Execution Asymmetry

To provide a transparent, hardware-grounded profile of where execution latency is spent, we decompose the end-to-end processing of Proteus into unified pipeline stages rather than obscured low-level operator runtimes. All evaluations are conducted on a single NVIDIA GeForce RTX 3090 under the full-precision setting (16-bit geometry and 8-bit reflectance intensity). As demonstrated below, while the underlying mathematical components are symmetrical, the execution timelines of encoding and decoding exhibit fundamental architectural asymmetry due to causality constraints.

Table 4: Pipeline stage runtime breakdown (in milliseconds) on SemanticKITTI (64×2048) and WOD (64×2688). Component definitions are strictly unified across both pipelines.

(a) Decoding: Strict Serial Execution

Pipeline Stage / Component	KITTI	WOD
Bits-Back Latent ($p(z)$ / $q(z)$)	13	17
SIG Head-planes (Planes 1–7)	23	24
SIG Mid-planes (Planes 8–10)	19	32
INS Tail Geometry (Planes 11–16)	22	29
Geometry Subtotal (Range-only)	**77**	**102**
INS Tail Attribute (Intensity)	29	39
Total Full Decoding Latency	**106**	**141**

(b) Encoding: Pipelined Concurrent Execution

Pipeline Stage / Branch	KITTI	WOD
Shared Sequential Prefix:		
SIG Mid-planes	17	31
Bits-Back Latent (Hyperprior)	13	16
Prefix Subtotal	*30*	*47*
Concurrent Branches (Parallel Window):		
Branch 1: SIG Head-planes	38	45
Branch 2: INS Tail Geometry	59	73
Branch 3: INS Tail Attribute	63	85
Synchronization, Defer-Flush & Join	5	4
Total Full Encoding Latency	**98**	**136**

Decoding: Causality-Locked Serial Accumulation. As detailed in Table 4(a), decoding must strictly obey causal dependencies across bit-planes and modalities. Decoding begins with the top-level latent prior $p(z)$ to seed the hyperprior (≈ 13 ms on KITTI), proceeds through the autoregressive decoding of *SIG Head-planes* (≈ 23 ms), re-encodes the variational posterior $q(z)$ into the ANS stack to restore state, decodes the *SIG Mid-planes* (≈ 19 ms), and finally decodes the FIFO range stream comprising *INS Tail Geometry* (≈ 22 ms) followed by *INS Tail Attribute* (≈ 29 ms).

Because no future bit-plane can be predicted prior to the exact reconstruction of preceding planes, decoding latency strictly accumulates as the direct scalar sum of all constituent pipeline stages ($\sum$ Stages $\approx$ 106 ms on SemanticKITTI and $\approx$ 141 ms on WOD). Crucially, *INS Tail Attribute* represents the single largest serial stage ($\approx 27\%$ of the decoding budget). In downstream perception tasks where sensory attributes are non-essential, omitting the attribute stream allows Proteus to decode full 16-bit geometry alone in **77 ms on SemanticKITTI** and **102 ms on WOD**, achieving near-real-time throughput that operates in the close vicinity of the 10 Hz sensor baseline.

Encoding: Asymmetric Pipelining and Multi-Threaded Concurrency. In contrast to decoding, the encoder has full simultaneous access to the ground-truth range image and reflectance channels. As characterized in Table 4(b), this asymmetry allows Proteus to reorganize the encoding timeline:

$$T_{\text{enc}} = T_{\text{prefix}} + \max\left(T_{\text{head}}, T_{\text{tail_geo}}, T_{\text{tail_attr}}\right) + T_{\text{sync}}, \tag{36}$$

where the shared sequential prefix T_{prefix} comprises the unconditional encoding of *SIG Mid-planes* and subsequent *Bits-Back Latent* hyperprior generation (≈ 30 ms on KITTI). Once the hyperprior is synthesized, three concurrent worker threads are dispatched across the GPU stream and host queues: (i) *SIG Head-planes* entropy coding on the primary thread, (ii) *INS Tail Geometry* range coding, and (iii) *INS Tail Attribute* range coding. A deferred-flush mechanism ensures that tail range bits precede attribute bits in the final byte stream without stalling concurrent arithmetic coding.

Because the three branches execute concurrently, the parallel window is bounded by the longest branch—namely *INS Tail Attribute* ($\approx$ 63 ms on KITTI and $\approx$ 85 ms on WOD). Consequently, incorporating full 8-bit attribute compression does not prolong the overall wall-clock time beyond the concurrent window ($T_{\text{enc_geo}} \approx T_{\text{enc_full}} \approx 98$ ms on KITTI and ≈ 136 ms on WOD).

Discrepancy in Intra-Component Runtimes between Encoding and Decoding. We note that individual component wall-clock times in Table 4(b) (e.g., INS Tail Geometry at 59 ms) cannot be directly compared to their counterparts in Table 4(a) (22 ms). In decoding, each entry reflects the isolated, uninterrupted runtime of a single serial thread. In encoding, the three concurrent branches share a single GPU context and host dispatcher; thus, the measured branch wall-clock times include thread scheduling overhead and kernel queue contention behind prior dispatches, whereas the actual per-component compute complexity remains mathematically symmetrical.

E.4 Architectural Constraints and Exploration of Infeasible Paths

During our performance optimization, we explored several prominent compiler-, hardware-, and algorithmic-level acceleration paradigms commonly adopted in high-performance neural computing. However, each of these avenues fundamentally conflicts with the core theoretical guarantees of Proteus—specifically, **deterministic bit-exactness**, **byte-level truncation robustness**, or **SIMT hardware execution efficiency**. We document these empirical findings below to delineate the rigorous boundaries of progressive LiDAR compression.

1. Empirical Disproval of GPU-Side Sequential Entropy Decoding. A seemingly intuitive approach to eliminate the $\approx$ 24 ms host-device synchronization gap is migrating the entire entropy decoding loop directly onto the GPU. We implemented a dedicated, single-lane sequential CUDA kernel replicating the bit-exact state transitions of our FIFO range and rANS decoders. Over an evaluation suite spanning 27.5 million symbols, this kernel maintained absolute bit-exactness (errsum $= 0$).

However, hardware profiling revealed that **GPU-side sequential decoding is approximately** $7.5\times$ **slower than the optimized CPU implementation**: decoding a standard 65k-symbol sub-block required 2.4 ms on the GPU versus 0.32 ms on the CPU. The root cause lies in the microarchitectural mismatch of SIMT hardware for serial state machines. The inner loop of entropy decoding

features tight, carry-dependent scalar arithmetic chains spanning ≈ 50 cycles per symbol, which stall massively on in-order GPU streaming multiprocessors lacking hardware branch predictors and instruction-level speculative execution. Conversely, modern CPU cores utilize sophisticated branch target buffers (BTB) and deeply pipelined superscalar out-of-order (OoO) engines, amortizing the state update to ≈ 16 effective cycles per symbol. Consequently, decoupling entropy decoding to the host CPU while executing neural inference on the GPU represents the global Pareto optimum; forcing serial entropy decoding onto the GPU severely degrades end-to-end throughput.

2. Incompatibility with Multi-Stream Packetization (GPUAC-like Paradigms). Parallel entropy decoders (e.g., GPUAC) achieve massive parallelism by partitioning the continuous symbol sequence into independent chunks (e.g., 8192 symbols per packet) separated by byte-aligned checkpoint headers, allowing multiple GPU warps to decode sub-streams independently.

While attractive for massive unconstrained data streams, this paradigm directly violates the foundational mathematical property of Proteus: **arbitrary byte-level prefix truncation**. Under physical wireless channel drops (as formalized in Appendix D), network packet drop-tail truncates the bitstream at arbitrary byte offsets. Introducing discrete packet boundaries and checkpoint tables would either (i) coarse-grain truncation tolerance to packet boundaries, causing entire sub-blocks to be discarded if an incomplete packet is received, or (ii) render trailing bits completely undecodable if intermediate headers are severed. Moreover, because each autoregressive sub-step in Proteus operates on a fine-grained grid ($\approx$ 65k symbols), the synchronization and memory-scatter overhead of multi-streaming outweighs its parallel efficiency. Preserving a continuous, single-stream FIFO bitstream is mathematically essential for continuous prefix decodability.

3. Numerical Hazards of cuDNN Algorithm Selection and JIT Compilation. We investigated standard deep learning compiler optimizations, including PyTorch 2.x Inductor JIT compilation (`torch.compile`), `cudnn.benchmark`, and native `channels_last` (NHWC) tensor layouts:

- **cuDNN Algorithm Switching vs. Discretization Grids:** Enabling `cudnn.benchmark` allows the runtime to dynamically select the fastest convolution algorithm (e.g., switching between implicit-GEMM and Winograd/tensor-core kernels). While mathematically identical in infinite precision, different convolution algorithms reorder floating-point summation trees. In FP16 half-precision, this algorithmic drift introduces maximum logit perturbations of $|\Delta| \approx 10^{-2}$ to 10^{-1}. In continuous perception tasks, this deviation is negligible; however, in lossless entropy modeling, conditional probabilities are quantized onto a discrete 8-bit parameter grid ($p_1 = \text{round}(\text{sigmoid}(\cdot) \cdot 256)$) where the threshold margin near decision boundaries is merely $\approx 1/256$. A microscopic floating-point deviation flips the predicted binary outcome, which immediately triggers an **autoregressive avalanche collapse**—corrupting all subsequent bit-planes and resulting in over 200,000 decoding errors per frame.
- **Compilation Inconsistencies:** Compiling sub-modules with Triton/Inductor similarly breaks cross-platform bit-exactness between the forward encoding graph and the inverse decoding pass.

To guarantee mathematical safety and crash-free decoding under all driving scenarios, Proteus strictly locks deterministic implicit-GEMM convolution primitives, rejecting heuristics that trade numerical reproducibility for modest GPU gains.

4. The Irreducible Nature of Causal Decoding. Finally, we highlight the fundamental theoretical difference between encoding and decoding throughput. While the encoder leverages ground-truth priors to execute one-shot parallel stage prediction ($T_{\text{enc}} \approx 98$ ms), the decoder is physically constrained by the causal information barrier: plane $l+1$ cannot be synthesized until plane l is materialized and fed back through the convolutional receptive field. The measured 106 ms decoding floor on SemanticKITTI represents the physical synergy of GPU arithmetic compute ($\approx 65\%$), irreducible host-device causal synchronization ($\approx 23\%$), and optimal CPU entropy throughput ($\approx 9\%$). Bypassing this floor without sacrificing either metric precision or deterministic safety remains a fundamental open challenge for all autoregressive neural compression frameworks.

F SCOPE OF RANGE REPRESENTATIONS AND GENERALIZATION TO BROADER MODALITIES

In this section, we provide a principled discussion on the representation scope of range images, examine operational characteristics across diverse LiDAR hardware architectures, and elaborate on the broader theoretical applicability of our deterministic truncation framework.

F.1 RANGE IMAGES AS SENSOR-NATIVE RAW REPRESENTATIONS

A common concern regarding range-image-based point cloud compression is whether organizing 3D points into 2D grids represents an ad-hoc compromise that introduces sampling artifacts, occlusions, or coordinate collisions ("holes"). From the perspective of physical sensor acquisition, the range image is not an artificial proxy, but the *sensor-native raw representation* of rotating and rasterized active range sensors:

- **Sensor-Native Topological Prior:** Mechanical LiDAR sensors emit discrete laser pulses along fixed elevation angles $\Theta = \{\theta_1, \dots, \theta_H\}$ and scan azimuthally across angles $\Phi = \{\phi_1, \dots, \phi_W\}$. Each beam sequentially records the radial Time-of-Flight (ToF) distance r. Arranging these measurements into an $H \times W$ range image inherently preserves the spatio-temporal scanning sequence and the physical neighborhood of the sensor array. Conversely, converting measurements into unorganized 3D Cartesian coordinates (x, y, z) discards this rasterization prior, forcing subsequent 3D codecs to expend substantial computational and entropy budgets reconstructing geometric adjacency via octrees or voxel grids.
- **Exact Bijective Equivalence vs. Calibration Artifacts:** When sensor beam calibration tables are provided—as in the Waymo Open Dataset (WOD) (Sun et al., 2020)—the projection between 2D range images and 3D Cartesian coordinates is an exact, mathematically lossless bijection (Equation 1). The coordinate collisions and "holes" historically observed in earlier benchmarks (e.g., SemanticKITTI (Behley et al., 2019)) arise entirely because those datasets released raw Cartesian point clouds without public beam calibration tables, requiring empirical spherical approximations. Such artifacts reflect historical calibration-omission residues rather than intrinsic flaws of the range representation.
- **Industrial and Perceptual Adoption:** Preserving sensor-native range images has long been established in both industrial platforms and autonomous perception pipelines. WOD natively stores, benchmarks, and distributes LiDAR data in range image format. Concurrently, native range-view perception backbones (e.g., LaserNet (Meyer et al., 2019)) demonstrate that operating directly on 2D range grids achieves high detection accuracy while dramatically lowering latency and memory footprints relative to dense 3D voxelization.

F.2 APPLICABILITY ACROSS DIVERSE LIDAR SENSOR ARCHITECTURES

While our empirical evaluations focus on standard rotational LiDAR benchmarks, the architectural principles of Proteus extend naturally across alternative LiDAR hardware architectures:

- **Solid-State / Flash LiDAR:** Solid-state Flash LiDARs replace rotating mechanical assemblies with an instantaneous 2D receiver array (e.g., SPAD/SiPM matrices). Each laser pulse captures a regular 2D grid of radial distances, directly mirroring the format of a range image. Consequently, Flash LiDAR data natively interfaces with the bit-plane slicing and decoupled entropy coding pipeline of Proteus with zero structural modification.
- **Non-Repetitive Scanning Semi-Solid-State LiDAR (e.g., Livox):** Sensors utilizing dual Risley prisms or scanning mirrors produce non-repetitive scanning patterns (e.g., petal-shaped trajectories). At the driver level, official protocols (e.g., Livox SDK Ethernet packets `LivoxEthPacket`) stream structured bursts of spherical coordinates (r, θ, ϕ) and reflectance values over UDP. Proteus can interface with such non-rasterized streams via two practical routes: (i) *Spatial-Temporal Angular Binning:* Because Livox hardware directly outputs spherical angles (θ, ϕ), measurements accumulated over a short temporal window (e.g., 50–100 ms) can be directly assigned to angular bins without trigonometric

inversion, synthesizing a 2D range grid; or (ii) *1D Streaming Decoupled Coding:* More fundamentally, the radial distance r in sequential packets represents an ordered sequence of physical distance measurements. The core engine of Proteus—partitioning quantized physical scalars into significant bits (SIG) encoded via self-contained bits-back coding and insignificant bits (INS) encoded via FIFO range coding—can be deployed directly over 1D sequential depth chunks, preserving strict truncation robustness and deterministic precision bounds.

F.3 Underlying Paradigm and Prospective Extensions to Broader Modalities

At its core, the methodology of Proteus decouples theoretical compression efficiency (reclaimed via Bits-Back coding on structural base layers) from in-transit channel adaptation (achieved via progressive range coding on physically grounded bit-planes). Because this mathematical decoupling is inherently modality-agnostic, the core architecture can be generalized to any structured physical field where measurements represent grounded physical quantities and progressive degradation must remain strictly deterministic:

- **4D Imaging Radar:** Modern high-resolution automotive radars generate dense 4D tensors spanning Range, Doppler, Azimuth, and Elevation. The spatial range and Doppler velocity channels represent physically calibrated scalar fields. Applying bit-plane decoupling to imaging radar tensors could enable graceful, deterministic resolution shedding during cooperative V2X bandwidth shortages without unverified spatial distortion.
- **Robotic and Consumer Depth Sensing (RGB-D / ToF):** In mobile robotics and edge spatial computing, structured-light and Time-of-Flight depth cameras generate dense depth maps where pixel values denote metric physical distances. Proteus's hybrid architecture—prioritizing coarse geometric boundaries before fine details and subordinate RGB attributes—naturally aligns with the hierarchical needs of real-time spatial mapping and obstacle avoidance under constrained uplinks.
- **Scientific and Telemedicine Imaging (Volumetric CT / MRI):** In high-stakes fields such as remote emergency triage and telemedicine, data representations (e.g., Hounsfield Units in Computed Tomography or relaxation times in Magnetic Resonance Imaging) require verifiable fidelity guarantees. Over low-bandwidth or fluctuating field connections, a progressive bit-plane degradation provides medical practitioners with an exact, mathematically bounded coarse reconstruction rather than an unverified statistical estimate.

In summary, while this work is evaluated on LiDAR point clouds, the underlying principles of Proteus offer the community a general blueprint for reconciling state-of-the-art entropy coding efficiency with progressive, channel-robust transmission across diverse physical sensing modalities.