NEURAL IMPLICIT PROBABILITY FIELDS FOR LOSSLESS POINTMAP COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pointmaps have recently emerged as a new geometric output of feed-forward 3D reconstruction models, creating a new target for 3D data compression. Their high spatial resolution and coordinate precision, however, incur substantial lossless storage costs. We propose Neural Implicit Probability Fields, a lossless pointmap codec that uses an implicit neural representation (INR) as an entropy model rather than a signal reconstructor. From a compression perspective, backpropagation can be viewed as compressing pointmap-specific statistical structure into the parameters of a compact neural probability field. To make this idea practical, we combine reversible geometric prediction with neural probability modeling, allowing the INR to focus on the remaining uncertainty rather than reconstructing the signal itself. The resulting codec preserves the native pixel-to-point correspondence and supports exact 16-bit reconstruction, while explicitly accounting for the transmitted neural model. Across diverse domains relevant to world modeling and embodied intelligence, our method reduces BPP by 1.4–30.3% over the strongest grid-preserving baselines, demonstrating that neural optimization can provide an effective mechanism for lossless compression of high-precision 3D geometry.

1 INTRODUCTION

Pointmaps are an emerging representation of 3D scene geometry, where each valid image pixel is associated with a 3D coordinate. They can be obtained by back-projecting depth maps into 3D space with calibrated cameras, or generated directly by recent feed-forward 3D reconstruction models through dense pointmap regression Wang et al. (2024); Leroy et al. (2024); Wang et al. (2025a;b); Yang et al. (2025a); Wang et al. (2025c; 2026). By organizing 3D coordinates on a regular image grid, pointmaps preserve both the spatial structure of images and the pixel-to-3D correspondence, providing a convenient interface between 2D observations and 3D geometry.

Beyond serving as 3D reconstruction outputs Wang et al. (2025b;a; 2026), pointmaps can also provide persistent geometric representations for world models and embodied agents. Recent work increasingly uses persistent 3D memory to maintain geometric consistency beyond short temporal context and to support long-horizon spatial reasoning, exploration, and planning Zhou et al. (2025a); Yang et al. (2025b). As high-precision geometry is continuously accumulated over time, the storage and transmission costs of pointmaps grow rapidly, motivating efficient lossless compression.

A straightforward solution is to reuse existing image Weinberger et al. (2000); Taubman et al. (2002); Sneyers et al. (2025) or point cloud codecs Graziosi et al. (2020); Wang et al. (2022); Liu et al. (2026). Image codecs preserve the 2D raster structure, but are primarily designed for densely valid images and do not explicitly account for the irregular valid regions introduced by pointmap masks. Point cloud codecs, in contrast, model geometry directly in 3D space, but discard the original pixel-to-point organization. As a result, neither formulation simultaneously preserves the native pointmap structure and exploits correlations among neighboring valid coordinates.

Our analysis suggests that effective pointmap compression should preserve its pixel-aligned grid structure while modeling only the valid geometric locations. This observation naturally leads to a coordinate-based probability model. A pointmap itself can be viewed as a field that maps each 2D grid location to a 3D coordinate; correspondingly, we model the probability distribution of its

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.

discrete symbols as an implicit field defined over the same spatial domain:

$$\text{Pointmap} \quad \underbrace{(i,j)}_{\text{2D grid}} \mapsto \underbrace{(x,y,z)}_{\text{3D geometry}} \quad \Longleftrightarrow \quad \underbrace{(i,j)}_{\text{2D grid}} \xrightarrow{\ell,\mathbf{h}} \underbrace{(p_x,p_y,p_z)}_{\text{bit probabilities}} \quad \text{Probability Field}$$

Here, ℓ denotes the current bit plane and $\mathbf{h}$ denotes the decoded context. By sharing the same 2D spatial domain as the pointmap, the probability field can be queried directly at valid locations without explicitly modeling invalid regions. We therefore introduce a neural implicit probability field, parameterized by a compact implicit neural representation (INR), as the entropy model for lossless pointmap coding.

Specifically, we first apply a mask-aware reversible predictive transform to remove local redundancy among valid pointmap coordinates. The resulting residual codes are decomposed into bit planes and modeled autoregressively from MSB to LSB, while decoded spatial context is further exploited within each bit plane. Conditioned on the spatial location, bit-plane index, and decoded context, a compact INR predicts the bit probabilities of the XYZ channels for arithmetic coding. The INR is fitted separately to each pointmap, and its parameters are quantized and entropy-coded as part of the bitstream. To ensure exact decoding, both encoder and decoder use the reconstructed quantized INR to generate identical probability estimates.

We evaluate the proposed codec on pointmaps from four diverse domains, including simulator, robot, human, and Internet. Our method consistently achieves lower full-precision lossless bitrate than both strong grid-preserving image codecs and geometry-only point-cloud codecs across all four domains, while preserving the original pixel-to-point correspondence and accounting for the transmitted INR model in the total bitrate. We further analyze the optimization efficiency of per-pointmap fitting and the composition of the bitstream and coding runtime.

Our contributions are summarized as follows:

- We introduce a neural implicit probability field for lossless pointmap compression, which parameterizes the entropy model over the same pixel-aligned spatial domain as the pointmap and supports probability queries only at valid locations.
- We develop a fully reversible coding framework that combines mask-aware predictive residualization with MSB-to-LSB bit-plane autoregression and spatial context modeling to exploit the structured dependencies of high-precision pointmaps.
- We build a complete per-pointmap neural codec in which the fitted INR model is quantized, entropy-coded, and transmitted as part of the bitstream, achieving improved lossless compression over strong image and point-cloud codecs across four domains.

2 Related Work

2.1 Lossless Image Compression

Traditional lossless image codecs reduce statistical redundancy through reversible prediction or transforms followed by entropy coding. Common lossless codecs include PNG, JPEG-LS with MED-based predictive coding Weinberger et al. (2000), JPEG2000 with reversible wavelet and bit-plane coding Taubman et al. (2002), and JPEG XL with more advanced prediction and context modeling Sneyers et al. (2025). Learning-based approaches further replace hand-crafted probability models with neural entropy models; ArIB-BPS explicitly decomposes image values into bit planes for autoregressive lossless coding Zhang et al. (2024), while FNLIC adapts a compact probability model to each individual image through fitting Zhang et al. (2025). Our work instead targets masked, high-precision pointmaps and models their entropy distribution with a coordinate-based neural implicit probability field.

2.2 Point Cloud Compression

Traditional point cloud compression is mainly represented by MPEG G-PCC and V-PCC Graziosi et al. (2020), where G-PCC uses hierarchical octrees MPEG (2021) and V-PCC projects 3D points onto 2D patches for video coding MPEG (2022). Learning-based methods model geometry using sparse convolutions Choy et al. (2019); Wang et al. (2021); Wang & Gao (2025) or octree

structures Huang et al. (2020); Fu et al. (2022); Song et al. (2023). Representative approaches include SparsePCGC with multiscale context modeling Wang et al. (2022), NBP with next-bit prediction Liu et al. (2026), and recent INR-based codecs that fit and transmit compact neural representations Huang et al. (2025; 2026). These methods operate on geometry-only 3D representations and therefore do not preserve the original pixel-to-point correspondence of pointmaps.

2.3 Implicit Neural Representations

Implicit neural representations (INRs) parameterize signals as coordinate-based neural functions, typically using multilayer perceptrons to map spatial coordinates to signal values. SIREN Sitzmann et al. (2020) employs periodic activation functions to improve the representation of fine-scale and high-frequency details. INRs have also been explored for compression, where compact coordinate-based networks or instance-specific representations are optimized and transmitted Dupont et al. (2021; 2022); Schwarz et al. (2023); Ladune et al. (2023). More recently, LosslessINR decomposes discrete signals into bit planes to enable exact reconstruction of high-precision values Han et al. (2025). Unlike these methods that use INRs to represent signal values, we use an INR to parameterize a probability field over pointmap bit-plane symbols for entropy coding.

3 Method

3.1 Overall Framework

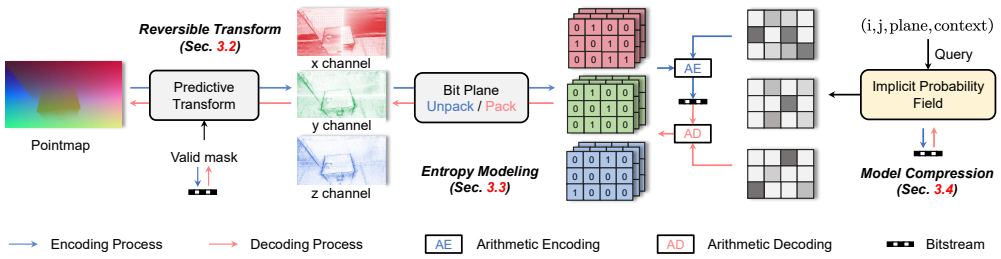


Figure 1: Overall framework of the proposed codec. A reversible predictive transform converts the pointmap into residuals, which are unpacked into bit-plane symbols and entropy-coded using probabilities predicted by an MLP-based INR. The INR parameters are also compressed and transmitted for decoding. Blue and red arrows denote encoding and decoding paths, respectively.

We represent a pointmap as a 16-bit coordinate field $\mathbf{P} \in \{0, \dots, 2^{16}-1\}^{H\times W\times 3}$ together with a valid mask $\mathbf{M} \in \{0,1\}^{H\times W}$. Each valid pixel stores a 3D coordinate (x, y, z) on the regular image grid, preserving the pixel-to-point correspondence of the scene geometry. Fig. 1 illustrates the proposed lossless codec, which consists of a reversible predictive transform, bit-plane entropy modeling with a per-pointmap neural implicit probability field, and model compression.

At the encoder, the valid mask is first encoded losslessly using PNG. A mask-aware reversible predictive transform then converts the valid pointmap coordinates into integer residual codes, which are decomposed into 16 bit planes and processed from MSB to LSB. For each pointmap, we fit an MLP-based INR to predict the conditional bit probabilities from spatial coordinates and previously decoded context. The fitted INR is quantized and entropy-coded for transmission, and the reconstructed quantized model is subsequently used to generate the probabilities for arithmetic coding of the residual bit planes. The resulting bitstream consists of the arithmetic-coded residual stream, the compressed INR model, and the valid-mask stream. The total bitrate is therefore

$$R_{\text{total}} = r_{\text{bit}} + r_{\text{model}} + r_{\text{mask}}, \tag{1}$$

where r_{bit}, r_{model}, and r_{mask} denote the residual, model, and mask streams, respectively.

At the decoder, the valid mask and quantized INR are first recovered from r_{mask} and r_{model}, respectively. The reconstructed INR provides the same conditional probabilities for arithmetic decoding, which recovers the residual bit planes from r_{bit} in MSB-to-LSB order. The decoded bit planes are then packed back into integer residuals, and the inverse mask-aware predictive transform exactly reconstructs the original 16-bit pointmap.

3.2 Mask-Aware Reversible Predictive Transform

Pointmap coordinates are locally smooth within valid geometric regions, making causal neighbors effective predictors of the current coordinate. Following JPEG-LS Weinberger et al. (2000), we adopt the median edge detector (MED) predictor. For a valid pixel (i, j) and coordinate channel $c \in \{x, y, z\}$, let L, T, and D denote the left, top, and top-left neighboring coordinates, respectively. When all three neighbors are available, the prediction is

$$\tilde{P}_{i,j,c} = \begin{cases} \min(L_c, T_c), & D_c \geq \max(L_c, T_c), \\ \max(L_c, T_c), & D_c \leq \min(L_c, T_c), \\ L_c + T_c - D_c, & \text{otherwise.} \end{cases} \tag{2}$$

By removing the locally predictable component, this prediction produces more compact residuals for subsequent bit-plane coding.

Unlike dense images, pointmaps may contain invalid pixels. We therefore make the predictor mask-aware by using only valid causal neighbors. When the full MED neighborhood is unavailable, we average the valid left and top neighbors using integer floor division, copy the available one if only one is valid, and use zero otherwise. This deterministic rule prevents invalid pixels from entering prediction and ensures identical predictions at the encoder and decoder.

Given the prediction $\tilde{P}_{i,j,c}$, we compute the residual in the 16-bit modular domain:

$$d_{i,j,c} = \left(P_{i,j,c} - \tilde{P}_{i,j,c}\right) \bmod 2^{16}. \tag{3}$$

The modular residual $d_{i,j,c}$ is then interpreted as a signed 16-bit residual $r_{i,j,c}$ and mapped to a non-negative zigzag code $u_{i,j,c}$ for bit-plane decomposition. Since the predictor is causal and deterministic and the residual mapping is bijective, the transform is exactly reversible. Detailed conversion and inverse rules are provided in the appendix.

3.3 Bit-Plane Entropy Modeling with a Neural Implicit Probability Field

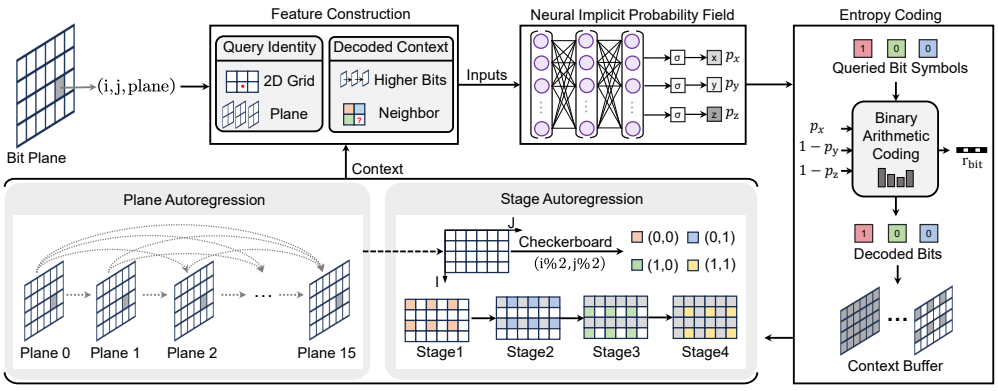


Figure 2: Bit-plane entropy modeling with a neural implicit probability field. The 16-bit zigzag residual codes are encoded plane by plane from MSB to LSB, with higher bit planes and earlier checkerboard stages providing decoded context for each query. An MLP-based INR predicts the bit probabilities of the XYZ channels for arithmetic coding, while the processed bits are written back to the context buffer for subsequent predictions.

After the reversible predictive transform, each valid coordinate is represented by a 16-bit zigzag residual code. We decompose the residual codes into 16 binary bit planes $\mathbf{B} = \{\mathbf{B}^0, \ldots, \mathbf{B}^{15}\}$, where $\mathbf{B}^0$ and $\mathbf{B}^{15}$ denote the MSB and LSB, respectively. Since the decomposition is bijective, lossless coding of the residual codes is equivalent to lossless coding of these bit planes.

Plane and stage autoregression. We encode the bit planes from MSB to LSB so that decoded higher-order bits can serve as context for subsequent prediction:

$$p_\theta(\mathbf{B}) = \prod_{\ell=0}^{15} p_\theta\left(\mathbf{B}^\ell \mid \mathbf{B}^{<\ell}\right), \tag{4}$$

where $\mathbf{B}^{<\ell} = \{\mathbf{B}^0, \ldots, \mathbf{B}^{\ell-1}\}$ denotes the previously decoded bit planes.

Within each bit plane, we further adopt a four-stage checkerboard order based on the spatial parity $(i \bmod 2, j \bmod 2)$. Let $\mathbf{B}_s^\ell$ denote the symbols in stage s of plane ℓ. The plane-wise distribution is further factorized as

$$p_\theta\left(\mathbf{B}^\ell \mid \mathbf{B}^{<\ell}\right) = \prod_{s=1}^{4} p_\theta\left(\mathbf{B}_s^\ell \mid \mathbf{B}^{<\ell}, \mathbf{B}_{<s}^\ell\right), \tag{5}$$

where $\mathbf{B}_{<s}^\ell$ contains the already decoded stages in the current plane. This ordering ensures that all conditioning information is available at both the encoder and decoder.

Neural implicit probability field. For each valid pixel (i, j) at bit plane ℓ and stage s, we construct a query feature

$$\mathbf{q}_{i,j,\ell,s} = \phi\left(i, j, \ell, \mathbf{h}_{i,j,\ell,s}\right), \tag{6}$$

where $\mathbf{h}_{i,j,\ell,s}$ contains decoded higher-bit history and previously decoded spatial context. The feature further includes the normalized 2D coordinate, bit-plane index, stage index, and valid/known flags.

A compact MLP-based INR maps each query to three bit probabilities:

$$\boldsymbol{\pi}_{i,j,\ell,s} = \sigma\left(f_\theta(\mathbf{q}_{i,j,\ell,s})\right) = [\pi_x, \pi_y, \pi_z], \tag{7}$$

where π_x, π_y, and π_z denote the probabilities that the current bits of the three coordinate channels equal one. The three probabilities are predicted jointly from the same context, while invalid pixels are skipped.

Entropy coding and context update. The predicted probabilities are used by a binary arithmetic coder to encode the corresponding bit symbols. After each coding stage, the processed bits are written back to the context buffer and become available for subsequent stages and bit planes. Following the same coding order therefore keeps the encoder and decoder synchronized, producing the residual bitstream r_{bit}.

3.4 Model Training and Model Compression

Our codec independently fits an INR entropy model to each pointmap, where per-pointmap optimization acts as an encoder that distills pointmap-specific statistical structure into compact neural parameters. After the reversible transform and bit-plane decomposition, the INR is optimized on valid locations under the plane and stage autoregressive order. We use a compact SIREN-style MLP Sitzmann et al. (2020) with four hidden layers and a hidden width of 128. For each query $\mathbf{q}_{i,j,\ell,s}$, the network predicts the three bit probabilities defined in Eq. 7. The model is optimized using the binary cross-entropy loss

$$\mathcal{L}_{\text{bit}}(\theta) = -\mathbb{E}_{(i,j,\ell,s)}\left[\sum_{c\in\{x,y,z\}} \left(b_{i,j,\ell,c} \log \pi_{i,j,\ell,s,c} + (1 - b_{i,j,\ell,c}) \log(1 - \pi_{i,j,\ell,s,c})\right)\right]. \tag{8}$$

During optimization, samples are drawn only from valid locations, with half allocated to seven LSB planes and the remainder to the other planes. By default, each pointmap is optimized for 800 steps using Muon for the main weight matrices and AdamW for the remaining parameters.

Since the INR is fitted separately to each pointmap, its parameters must also be compressed and transmitted. We apply per-tensor adaptive 8-bit quantization to each parameter tensor $\mathbf{W}$:

$$\mathbf{Q} = \text{round}\left(\frac{\mathbf{W} - w_{\min}}{w_{\max} - w_{\min}} \cdot 255\right), \tag{9}$$

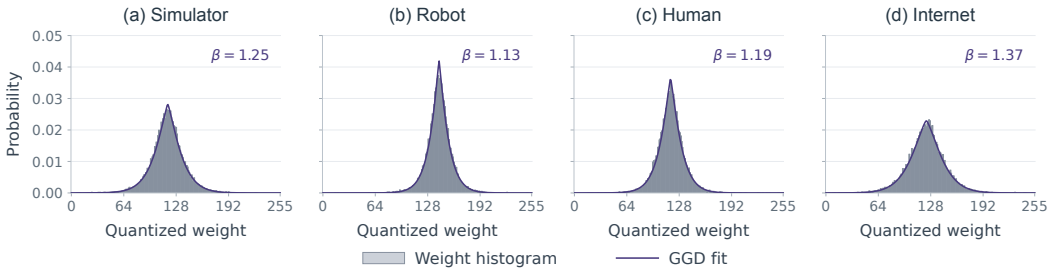


Figure 3: Quantized weight distributions across four domains. We visualize the same hidden layer per domain. Gray bars show the observed frequencies of 8-bit quantized weights, while purple curves show the fitted GGD probabilities. β denotes the GGD shape parameter.

where $w_{\min}$ and $w_{\max}$ denote the tensor-wise minimum and maximum. For constant tensors, we set $\mathbf{Q} = \mathbf{0}$. The quantized symbols are modeled with a generalized Gaussian distribution (GGD):

$$p(q \mid \mu, \alpha, \beta) = \frac{\exp\left[-\left|\frac{q-\mu}{\alpha}\right|^{\beta}\right]}{\sum_{k=0}^{255} \exp\left[-\left|\frac{k-\mu}{\alpha}\right|^{\beta}\right]}, \qquad q \in \{0, \ldots, 255\}. \tag{10}$$

The parameters μ, α, and β are estimated independently for each tensor by moment matching. As illustrated in Fig. 3, the fitted GGD closely follows the empirical distributions of the quantized weights. Each tensor is then arithmetic-coded using its fitted distribution, and the resulting streams, together with the required quantization and distribution parameters, form r_{model}.

Importantly, the encoder reconstructs the quantized INR after model compression and uses it, rather than the original full-precision model, to predict bit-plane probabilities. The decoder reconstructs the same INR from r_{model}, ensuring identical probability predictions at both sides.

4 Experiments

4.1 Pointmap Dataset Construction

As shown in Table 1, we construct pointmaps from four OmniWorld domains Zhou et al. (2025b): Simulator, Robot, Human, and Internet, using OmniWorld-Game, DROID Khazatsky et al. (2024), HOI4D Liu et al. (2022), and OmniWorld-CityWalk, respectively. These data cover synthetic scenes, robot manipulation, human-object interaction, and unconstrained Internet videos. For Game, DROID, and HOI4D, pointmaps are obtained by back-projecting aligned depth maps using camera intrinsics, while CityWalk pointmaps are reconstructed using Pi3X, an enhanced version of π^3 Wang et al. (2026). Each frame contains a three-channel 16-bit pointmap and a valid mask, and frames are organized into 64-frame clips with shared coordinate quantization parameters.

We use a clip-level split with 1,939 training clips and 250 test clips, avoiding temporal overlap between the two sets. Our codec requires no dataset-level training, and all reported results are evaluated on the test split. We select frames 0 and 32 from each test clip, resulting in 500 evaluation pointmaps. Fig. 4 shows examples from the four domains.

Table 1: Statistics of the data used in our experiments. Each clip contains 64 frames, and the last column reports the selected test frames for codec evaluation.

Domain	Source	Construction	Resolution	Clips (train/test)	Frames (train/test)	Eval. frames
Simulator	OmniWorld-Game	Depth back-projection	720×1280	350/49	22,400/3,136	98
Robot	DROID	Depth back-projection	720×1280	413/54	26,432/3,456	108
Human	HOI4D	Depth back-projection	1080×1920	256/32	16,384/2,048	64
Internet	OmniWorld-CityWalk	Pi3X reconstruction	378×672	920/115	58,880/7,360	230
Total	–	–	Mixed	**1,939/250**	**124,096/16,000**	**500**

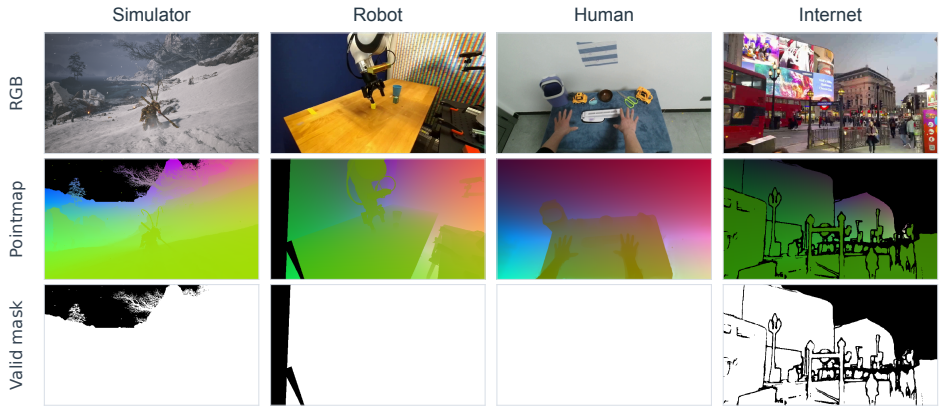


Figure 4: Pointmap examples across four domains. Rows show RGB images, pointmaps with normalized XYZ coordinates mapped to RGB, and valid masks (white: valid; black: invalid). Each valid pointmap pixel stores a 3D coordinate while preserving the image-grid layout.

4.2 Experimental Setup

We compare our method with both grid-preserving and geometry-only codecs. PNG, JPEG2000, JPEG XL, and ArIB-BPS Zhang et al. (2024) operate directly on the pointmap raster and preserve its spatial layout, while G-PCC MPEG (2021) and NBP Liu et al. (2026) encode the valid 3D points without preserving their original pixel locations. All grid-preserving methods are evaluated on the original full-resolution 16-bit pointmaps. The learning-based baselines, ArIB-BPS and NBP, are trained on the training split constructed in Sec. 4.1.

Our method requires no dataset-level pre-training; by default, a per-pointmap INR is optimized for 800 Muon steps following Sec. 3.4. Grid-preserving methods report *bpp-pixel*, whereas geometry-only methods use *bpp-point*. For our method, bitrate includes the residual, model, and mask streams, while encoding time includes optimization. All experiments run on the same server, with learning-based methods using a single NVIDIA A40 GPU; runtimes are averaged per frame.

4.3 Experiment Results

Lossless Compression Performance. Table 2 reports the lossless compression results averaged over 500 test pointmaps. Among grid-preserving codecs, our method achieves the lowest BPP in all four domains. Compared with the best competing codec in each domain, it reduces BPP by 13.6%, 2.4%, 30.3%, and 1.4% on Simulator, Robot, Human, and Internet, respectively. Notably, the reported bitrate includes both the transmitted INR model and valid-mask streams. The improved compression comes at the cost of per-pointmap optimization during encoding. At the default 800-step setting, encoding takes 26.61–28.45 seconds per frame, while decoding requires only 0.44–2.31 seconds. Compared with ArIB-BPS, our method achieves approximately $19\times$–$26\times$ faster decoding, although traditional codecs remain faster.

For geometry-only codecs, G-PCC and NBP are evaluated using bpp-point after removing duplicated 3D coordinates. Since duplicated coordinates account for only a small fraction of valid pixels, bpp-point remains close to bpp-pixel in our data. G-PCC and NBP progressively refine geometry across spatial scales using local occupancy or neighborhood dependencies. At the fine resolutions required by 16-bit coordinates, however, the corresponding 3D representations become increasingly sparse, weakening the available spatial context. Pointmaps instead preserve dense pixel-aligned neighborhoods in the 2D domain, allowing our model to exploit local geometric correlations more effectively.

Rate–optimization trade-off. As shown in Fig. 5, increasing the optimization budget consistently reduces BPP, but with diminishing returns. Increasing the number of steps from 400 to 800 lowers BPP by 6.0%, 3.6%, 14.7%, and 6.0% on Simulator, Robot, Human, and Internet, respectively,

Table 2: Lossless compression results across four domains. BPP denotes bpp-pixel for grid-preserving methods and bpp-point for geometry-only methods. Enc./Dec. reports encoding/decoding time in seconds per frame. Our encoding time includes per-instance optimization.

		Simulator		Robot		Human		Internet	
Method	**Type**	**BPP**	**Enc. / Dec.**	**BPP**	**Enc. / Dec.**	**BPP**	**Enc. / Dec.**	**BPP**	**Enc. / Dec.**
Grid-preserving methods (BPP = bpp-pixel)									
PNG	Traditional	14.80	2.59 / 0.05	14.53	2.80 / 0.05	9.12	6.18 / 0.10	20.32	0.74 / 0.01
JPEG2000	Traditional	13.94	0.37 / 0.28	10.03	0.29 / 0.23	11.88	0.66 / 0.56	16.34	0.13 / 0.08
JPEG XL	Traditional	10.26	2.54 / 0.58	7.74	1.78 / 0.65	5.56	3.17 / 0.95	11.71	0.66 / 0.24
ArIB-BPS	Shared learned	11.37	28.46 / 30.02	8.00	22.21 / 26.25	6.62	44.96 / 59.17	11.10	7.08 / 8.33
Ours (400 steps)	Per-instance	9.44	14.43 / 1.15	7.84	14.46 / 1.14	4.55	15.50 / 2.25	11.65	13.60 / 0.46
Ours (800 steps)	Per-instance	8.87	27.75 / 1.18	7.56	28.15 / 1.15	3.88	28.45 / 2.31	10.95	26.61 / 0.44
Ours (1600 steps)	Per-instance	**8.72**	53.95 / 1.14	**7.42**	53.68 / 1.14	**3.73**	54.14 / 2.47	**10.73**	52.19 / 0.44
Geometry-only methods (BPP = bpp-point)									
G-PCC	Traditional	10.63	13.13 / 8.54	11.45	16.39 / 10.83	7.22	42.50 / 26.80	14.84	3.81 / 2.54
NBP	Shared learned	10.48	5.84 / 6.22	12.66	7.63 / 8.16	13.97	17.25 / 18.63	13.91	2.28 / 2.41

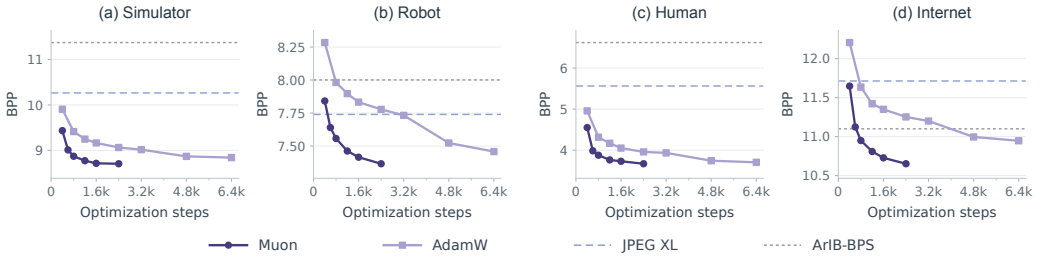


Figure 5: Lossless BPP versus optimization steps across four domains. We compare Muon and AdamW, with JPEG XL and ArIB-BPS shown as horizontal references.

while 1600 steps provide only a further 1.7–3.9% reduction at nearly twice the encoding time. We therefore use 800 steps as the default setting.

Muon also converges faster than AdamW under the same optimization budget. By orthogonalizing momentum updates, Muon better exploits the matrix structure of the INR weights and improves optimization conditioning. This matrix-aware update is well suited to per-pointmap fitting, allowing Muon to achieve consistently lower BPP with fewer steps across all four domains.

Bitstream and runtime composition. Table 3 shows that the entropy-coded bit planes dominate the bitstream, accounting for 93.74–95.59% in the Simulator, Robot, and Human domains. For Internet, the model share increases to 27.31%, as the per-pointmap model overhead is amortized over a smaller pointmap. The mask contributes less than 4% in all domains, while the reversible transform introduces no additional bits. Excluding fitting, bit-plane coding also dominates the runtime, accounting for 62.78–83.79% of encoding and 81.57–89.11% of decoding time.

These results suggest that future efficiency improvements should focus on faster per-pointmap optimization and bit-plane probability coding, while more compact INR models could further reduce the overhead for smaller pointmaps. Fitting is excluded only from this runtime breakdown and remains included in the encoding times reported in Table 2.

Table 3: Bitstream and runtime composition (%). Bitstream proportions are computed over all evaluation frames, while runtime proportions are measured on four frames per domain after one warm-up frame.

	Simulator				Robot				Human				Internet			
Metric	**Trans.**	**Bits**	**Model**	**Mask**	**Trans.**	**Bits**	**Model**	**Mask**	**Trans.**	**Bits**	**Model**	**Mask**	**Trans.**	**Bits**	**Model**	**Mask**
Bitstream	0.00	94.19	4.93	0.88	0.00	93.74	5.70	0.56	0.00	95.59	4.21	0.20	0.00	69.04	27.31	3.65
Enc. time (w/o fit)	6.30	78.45	13.06	2.19	5.65	77.58	14.93	1.84	7.98	83.79	8.18	0.05	2.60	62.78	26.86	7.76
Dec. time	6.52	86.93	6.16	0.39	5.18	87.13	7.30	0.39	6.99	89.11	3.88	0.02	3.17	81.57	14.85	0.41

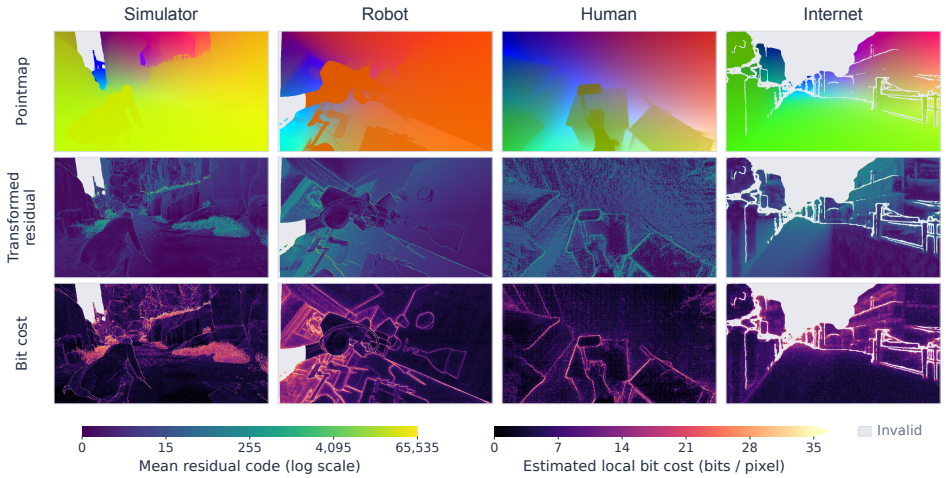


Figure 6: Spatial bit allocation. Rows show pointmaps, transformed residual codes, and estimated bit costs. Pointmaps are normalized using the 1st–99th percentiles of valid values; residual maps show the mean XYZ code on a logarithmic scale. Bit costs sum $-\log_2 p$ over channels and bit planes, excluding model overhead. Gray denotes invalid pixels, and costs above 35 bits/pixel are saturated for visualization.

Spatial Bit Allocation. Fig. 6 visualizes the transformed residuals and spatial coding costs across the four domains. Higher bit costs generally concentrate around geometric boundaries and complex structures, whereas smooth regions require fewer bits. Although the residual and bit-cost maps show similar spatial patterns, they are not identical: the coding cost is determined by the conditional probability of each observed bit rather than its residual magnitude alone. This indicates that the learned probability field captures spatially varying predictability beyond the reversible transform.

Component Ablation. Table 4 evaluates the contributions of the reversible transform, bit history, and spatial context. Without the transform, combining the two contexts already reduces BPP substantially, showing that the INR can capture useful dependencies from the original coordinates. The transform alone provides a much larger reduction, decreasing BPP from 30.56–35.55 to 9.38–13.10 across the four domains, confirming the importance of removing local geometric redundancy before entropy modeling.

Table 4: Component ablation at 800 Muon steps. T: transform; H: bit history; S: spatial context.

T	H	S	Simulator	Robot	Human	Internet
–	–	–	30.56	32.26	35.55	33.45
–	✓	✓	18.12	18.85	15.76	21.47
✓	–	–	12.16	10.39	9.38	13.10
✓	–	✓	11.41	9.76	8.45	11.87
✓	✓	–	9.54	8.20	4.39	12.10
✓	✓	✓	**8.87**	**7.56**	**3.88**	**10.95**

On top of the transform, bit history provides particularly strong gains, especially on Human, where BPP drops from 9.38 to 4.39. Spatial context brings further consistent improvements by exploiting already decoded neighbors within each bit plane. Combining both contexts achieves the lowest BPP in every domain, demonstrating that cross-plane and within-plane dependencies are complementary.

5 Conclusion

We presented a lossless pointmap codec based on neural implicit probability fields. A compact INR models bit-level probabilities for entropy coding, together with a mask-aware reversible transform and MSB-to-LSB bit-plane context modeling. The fitted INR is quantized and transmitted to enable exact decoding. Experiments across four domains demonstrate consistent bitrate gains over strong grid-preserving baselines, while revealing the trade-off between per-pointmap optimization and coding efficiency. These results support per-pointmap implicit probability modeling as a promising direction for high-precision 3D geometry compression and efficient 3D storage and transmission.

AI USE STATEMENT

In the preparation of this manuscript, we utilized a Large Language Model (LLM) as a writing assistant. The LLM was used solely to improve clarity, correct grammatical errors, and refine the language to meet professional academic standards. All scientific contributions, including the core ideas, methodology, experimental design, and interpretation of results, are the original work of the authors.

REFERENCES

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3075–3084, 2019.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. Coin++: Neural compression across modalities. *arXiv preprint arXiv:2201.12904*, 2022.

Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 625–633, 2022.

Danillo Graziosi, Ohji Nakagami, Satoru Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali 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:e13, 2020.

Woo Kyoung Han, Byeonghun Lee, Hyunmin Cho, Sunghoon Im, and Kyong Hwan Jin. Towards lossless implicit neural representation via bit plane decomposition. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2269–2278. IEEE, 2025.

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. 1313–1323, 2020.

Wenjie Huang, Qi Yang, Shuting Xia, He Huang, Yiling Xu, and Zhu Li. LINR-PCGC: Lossless implicit neural representations for point cloud geometry compression. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 28577–28586. IEEE, 2025.

Wenjie Huang, Qi Yang, Shuting Xia, He Huang, Zhu Li, and Yiling Xu. HybridINR-PCGC: Hybrid lossless point cloud geometry compression bridging pretrained model and implicit neural representation. *arXiv preprint arXiv:2602.21662*, 2026.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. Cool-chic: Coordinate-based low complexity hierarchical image codec. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13469–13476. IEEE, 2023.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 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*, 2026.

Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. HOI4D: A 4d egocentric dataset for category-level human-object interaction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20981–20990. IEEE, 2022.

MPEG. MPEG G-PCC TMC13. https://github.com/mpeggroup/mpeg-pcc-tmc13. *ISO/IEC JTC 1/SC 29/WC 7*, 2021.

MPEG. MPEG V-PCC TMC2. https://github.com/mpeggroup/mpeg-pcc-tmc2. *ISO/IEC JTC 1/SC 29/WG 7 N00100*, 2022.

Jonathan Richard Schwarz, Jihoon Tack, Yee Whye Teh, Jaeho Lee, and Jinwoo Shin. Modality-agnostic variational compression of implicit neural representations. *arXiv preprint arXiv:2301.09479*, 2023.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Jon Sneyers, Jyrki Alakuijala, Luca Versari, Zoltán Szabadka, Sami Boukortt, Amnon Cohen-Tidhar, Moritz Firsching, Evgenii Kliuchnikov, Tal Lev-Ami, Eric Portis, et al. 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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14368–14377. IEEE, 2023.

David S Taubman, Michael W Marcellin, and Majid Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286–287, 2002.

Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pp. 73–82. IEEE, 2021.

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, 2022.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025a.

Kangli Wang and Wei Gao. Unipcgc: towards practical point cloud geometry compression via an efficient unified approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12721–12729, 2025.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10510–10522. IEEE, 2025b.

Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5261–5271. IEEE, 2025c.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE, 2024.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations*, volume 2026, pp. 10481–10497, 2026.

Marcelo J Weinberger, Gadiel Seroussi, and Guillermo Sapiro. The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls. *IEEE Transactions on Image processing*, 9(8):1309–1324, 2000.

Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21924–21935. IEEE, 2025a.

Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3D-mem: 3D scene memory for embodied exploration and reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17294–17303, 2025b.

Zhe Zhang, Huairui Wang, Zhenzhong Chen, and Shan Liu. Learned lossless image compression based on bit plane slicing. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27569–27578. IEEE, 2024.

Zhe Zhang, Zhenzhong Chen, and Shan Liu. Fitted neural lossless image compression. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23249–23258. IEEE, 2025.

Siyuan Zhou, Yilun Du, Yuncong Yang, Lei Han, Peihao Chen, Dit-Yan Yeung, and Chuang Gan. Learning 3D persistent embodied world models. In *Advances in Neural Information Processing Systems*, 2025a.

Yang Zhou, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Haoyu Guo, Zizun Li, Kaijing Ma, Xinyue Li, Yating Wang, Haoyi Zhu, et al. OmniWorld: A multi-domain and multi-modal dataset for 4d world modeling. *arXiv preprint arXiv:2509.12201*, 2025b.

A Appendix

A.1 Detailed Reversible Predictive Transform

This appendix provides the complete mask-aware prediction and inverse rules. Let $\mathbf{P} \in \{0, \dots, 2^{16}-1\}^{H\times W\times 3}$ be the quantized pointmap and $\mathbf{M} \in \{0,1\}^{H\times W}$ its valid mask. Pixels are processed in raster order, and the three coordinate channels are handled independently. Invalid pixels are skipped, while the losslessly decoded mask determines which causal neighbors are available.

Mask-aware prediction. For a valid pixel (i, j) and channel c, let L_c, T_c, and D_c denote the left, top, and top-left reconstructed coordinates. When all three neighbors are valid, we use the MED predictor

$$\mathrm{MED}(L_c, T_c, D_c) = \begin{cases} \min(L_c, T_c), & D_c \geq \max(L_c, T_c), \\ \max(L_c, T_c), & D_c \leq \min(L_c, T_c), \\ L_c + T_c - D_c, & \text{otherwise.} \end{cases} \tag{11}$$

When the full MED neighborhood is unavailable, the prediction becomes

$$\tilde{P}_{i,j,c} = \begin{cases} \lfloor (L_c + T_c)/2 \rfloor, & L_c, T_c \text{ valid}, \\ L_c, & L_c \text{ valid only}, \\ T_c, & T_c \text{ valid only}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Thus, the top-left neighbor is used only when the complete MED neighborhood is available.

Residual and zigzag mapping. For notation simplicity, we omit pixel and channel indices. The prediction residual is first computed in the 16-bit modular domain, then interpreted as a signed value

and mapped by zigzag coding:

$$d = (P - \tilde{P}) \bmod 2^{16}, \tag{13}$$

$$r = \begin{cases} d, & d < 2^{15}, \\ d - 2^{16}, & d \geq 2^{15}, \end{cases} \tag{14}$$

$$u = \begin{cases} 2r, & r \geq 0, \\ -2r - 1, & r < 0. \end{cases} \tag{15}$$

Here $r \in [-2^{15}, 2^{15} - 1]$ and $u \in \{0, \ldots, 2^{16} - 1\}$, so each zigzag code can be represented by exactly 16 bit planes.

Inverse transform. The decoder processes valid pixels in the same raster order and recomputes $\tilde{P}$ from previously reconstructed neighbors using Eq. (12). The signed residual is recovered by

$$r = \begin{cases} u/2, & u \text{ even}, \\ -(u+1)/2, & u \text{ odd}, \end{cases} \tag{16}$$

and the original coordinate follows from

$$P = (\tilde{P} + r) \bmod 2^{16}. \tag{17}$$

Because the mask, prediction rule, coding order, and zigzag mapping are identical at the encoder and decoder, each prediction is reproduced exactly. Since the residual mapping is bijective, the original 16-bit coordinates are recovered losslessly.

A.2 INR Architecture and Optimization Details

We fit a separately initialized INR entropy model to each full-resolution pointmap. Table 5 summarizes the default 800-step optimization configuration. Here, n denotes the fan-in, i.e., the input dimension of the corresponding linear layer.

Query representation. A query is defined by a valid pixel (i, j) and bit plane ℓ. Its checkerboard stage is determined by

$$s = 2(i \bmod 2) + (j \bmod 2) + 1, \qquad s \in \{1, 2, 3, 4\}. \tag{18}$$

The query concatenates the normalized grid coordinates, bit-plane index, higher-bit history, spatial context, and stage index. For 16-bit residual codes, its dimension is

$$D_{\text{in}} = \underbrace{2}_{\text{grid}} + \underbrace{1}_{\text{bit}} + \underbrace{3(16-1)}_{\text{bit history}} + \underbrace{4(3+2)}_{\text{spatial context}} + \underbrace{1}_{\text{stage}} = 69. \tag{19}$$

Each of the four spatial neighbors (top, left, right, and bottom) contributes three current-plane bit values together with a known flag and a valid flag.

For zero-based row and column indices, spatial coordinates are normalized as

$$\bar{i} = \frac{2i}{\max(H-1, 1)} - 1, \qquad \bar{j} = \frac{2j}{\max(W-1, 1)} - 1. \tag{20}$$

Following the main text, $\ell = 0$ denotes the most-significant bit plane. Let $k = 15 - \ell$ denote its bit position from the LSB. The bit-plane and stage indices are normalized as

$$\bar{k} = \frac{2k}{15} - 1, \qquad \bar{s} = \frac{2(s-1)}{3} - 1. \tag{21}$$

Available context bits and binary flags are represented by -1 and $+1$, while unavailable bit values are set to zero. Only information available under the decoding order is exposed during fitting.

Sampling. Each sampled pixel–bit query jointly predicts the three coordinate bits, yielding $3 \times 262{,}144 = 786{,}432$ binary targets per optimization step. Half of the queries are sampled from the seven least-significant bit planes, while the remaining half are sampled from the other nine planes.

Table 5: Implementation details of the INR entropy model. Learning rates denote their initial values.

Setting	Configuration
Network architecture	
Backbone	SIREN with four sine hidden layers of width 128
Layer dimensions	$69 \to 128 \to 128 \to 128 \to 128 \to 3$
Activation	$\sin(\omega_0 z)$ with $\omega_0 = 30$
Input dimension	$D_{\text{in}} = 69$
Output	Three logits for the XYZ-channel bit probabilities
Parameter count	58,883, including biases
Initialization	
First-layer weights	$\mathcal{U}(-1/n,\, 1/n)$
Remaining weights	$\mathcal{U}(-\sqrt{6/n}/30,\, \sqrt{6/n}/30)$
Biases	$\mathcal{U}(-1/\sqrt{n},\, 1/\sqrt{n})$
Per-pointmap fitting	
Default budget	800 optimization steps
Queries per step	$262{,}144 = 2^{18}$ pixel–bit queries
Pixel sampling	Uniform over valid pixels with replacement
Bit-plane sampling	0.5 for the seven least-significant planes and 0.5 for the remaining nine planes, uniform within each group
Objective	Binary cross-entropy over the three coordinate channels
Muon	
Parameters	Input and hidden-layer weight matrices
Learning rate	1×10^{-3}
Momentum	0.95 with Nesterov momentum
Newton–Schulz	5 FP32 iterations with coefficients $(3.4445, -4.7750, 2.0315)$
Update normalization	Frobenius normalization with $\epsilon = 10^{-7}$
Update scaling	$0.2\sqrt{\max(m, n)}$ for an $m \times n$ matrix
AdamW	
Parameters	Output-layer weight matrix and all biases
Learning rate	5×10^{-4} for Robot; 3×10^{-4} for the other domains
Coefficients	$(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$
Shared settings	
Weight decay	0
Learning-rate schedule	Multiply both learning rates by 0.333 after steps 500 and 700

A.3 Dataset Construction and Preprocessing Details

This section provides additional details on the construction and preprocessing of the pointmaps used in our experiments. In addition to the train/test split reported in Table 1, we reserve 238 clips for validation, including 42 Simulator, 49 Robot, 32 Human, and 115 Internet clips. All splits are performed at the clip level to avoid temporal overlap.

Source data. For the Simulator domain, we retain 441 clips from 116 OmniWorld-Game scenes. Pointmaps are constructed from the RGB–depth sequences and corresponding camera intrinsics at the native 720×1280 resolution.

For the Robot domain, we use 516 camera clips from 175 DROID episodes. We use the released `foundation_stereo` depth maps together with the corresponding RGB frames and camera calibration to construct pointmaps at 720×1280 resolution.

For the Human domain, we retain 320 clips from HOI4D. Pointmaps are constructed from the released `prior_depth` maps and the corresponding camera intrinsics at the native 1080×1920 resolution.

For the Internet domain, we extract 1,150 64-frame clips from OmniWorld-CityWalk. Since aligned depth is unavailable, we generate dense pointmaps using Pi3X, an enhanced version of π^3 Wang et al. (2026), at 378×672 resolution.

Depth back-projection. For Simulator, Robot, and Human, pointmaps are obtained by back-projecting the aligned depth maps using the corresponding camera intrinsics. Let $D(u, v)$ denote the depth at image coordinate (u, v) and (f_x, f_y, c_x, c_y) denote the camera intrinsics. For a valid depth value, the corresponding 3D coordinate is

$$\begin{aligned} X(u,v) &= \frac{u - c_x}{f_x} D(u,v), \\ Y(u,v) &= \frac{v - c_y}{f_y} D(u,v), \\ Z(u,v) &= D(u,v). \end{aligned} \tag{22}$$

The resulting coordinates are stored at their original image locations, thereby preserving the pixel-to-point correspondence.

Valid mask. The valid mask indicates pixels for which a reliable 3D coordinate is available. For depth-based pointmaps, pixels without valid depth are excluded. For Pi3X-based pointmaps, invalid or unavailable geometric predictions are excluded according to the reconstruction output. Only valid coordinates participate in the predictive transform, INR fitting, and entropy coding.

Coordinate quantization. All pointmaps are represented using 16-bit coordinates. Frames are organized into 64-frame clips, and the same coordinate quantization parameters are shared by all frames within each clip. This preserves a consistent coordinate representation over time while allowing every encoded pointmap to be reconstructed exactly at the quantized 16-bit precision.

A.4 Validity Masks and Rate Accounting

All grid-preserving methods preserve the original pointmap raster. PNG, JPEG2000, and JPEG XL encode the complete XYZ raster, including values at invalid locations. For ArIB-BPS, invalid pixels are set to zero before tiled coding; fully invalid tiles contain no image payload, while their required side information is still counted. For all grid-preserving baselines, the validity mask is encoded losslessly as a separate PNG stream. An all-valid mask is represented implicitly and incurs no mask bits.

For each pointmap, we report

$$\mathrm{BPP}_{\mathrm{pixel}} = \frac{B_{\mathrm{data}} + B_{\mathrm{mask}} + B_{\mathrm{side}} + B_{\mathrm{model}}}{N_{\mathrm{valid}}}, \qquad N_{\mathrm{valid}} = \sum_{u,v} M_{u,v}, \tag{23}$$

where one valid pixel corresponds to one XYZ coordinate triplet. Here, B_{data} denotes the coded pointmap payload, B_{mask} the lossless validity-mask stream, and B_{side} any required codec headers or side information. B_{model} denotes the transmitted INR model stream for our method and is zero for the conventional and shared learned baselines.

For PNG, JPEG2000, and JPEG XL, the reported rate is computed from the actual compressed raster and mask file sizes together with the required side information. For tiled ArIB-BPS, the rate includes the coded tile streams, PNG mask, and a header cost of $80 + 64T$ bits per pointmap, where T is the total number of tiles, including fully invalid ones. For our method, all quantized INR parameters and the metadata required to reconstruct them are included in B_{model}.

Invalid pixels and coding padding are excluded from N_{valid}, while any bits spent coding these locations remain included in the numerator.

A.5 Unique-to-Valid Point Ratio

We quantify coordinate duplication over the 500 evaluation pointmaps. For each pointmap, N_{valid} denotes the number of valid pixels, while N_{unique} denotes the number of distinct 16-bit quantized

Table 6: Mean unique-to-valid point ratio over the evaluation pointmaps.

Domain	**Frames**	$\mathbf{N}_{\text{unique}}/\mathbf{N}_{\text{valid}}$
Simulator	98	98.54%
Robot	108	100.00%
Human	64	100.00%
Internet	230	99.82%

XYZ coordinates among them. Deduplication is performed independently for each pointmap. Table 6 reports the mean $N_{\text{unique}}/N_{\text{valid}}$ ratio for each domain.

The ratios are close to 100% across all four domains, indicating that coordinate duplication is limited after 16-bit quantization. Therefore, bpp-point and bpp-pixel use nearly identical normalization in our evaluation. Nevertheless, geometry-only coding does not preserve the original raster layout or pixel multiplicity, and thus does not guarantee lossless reconstruction of the pointmap representation.

A.6 Entropy Coding and Model Reconstruction Details

For each INR output logit z, we first convert the predicted probability into a 16-bit integer CDF:

$$p = \text{clip}\big(\sigma(z), 10^{-5}, 1 - 10^{-5}\big)\,, \tag{24}$$

$$k = \text{clip}(\text{round}(65535p), 1, 65534)\,, \tag{25}$$

$$c = 1 + \text{round}\left[65534\left(1 - \frac{k}{65535}\right)\right]. \tag{26}$$

The binary arithmetic coder therefore uses the integer CDF $(0, c, 65536)$ for symbols $(0, 1)$. All probability conversion is performed in FP32 using round-to-nearest with ties to even, and the encoder and decoder apply the same conversion rule.

For an 8-bit quantized weight symbol $a \in \{0, \ldots, 255\}$, the transmitted INR parameter is reconstructed as

$$\widehat{w} = \frac{a}{255}(w_{\max} - w_{\min}) + w_{\min}. \tag{27}$$

Dequantization is performed in FP32. The tensor bounds $w_{\min}$ and $w_{\max}$ and the GGD parameters μ, α, and β are serialized as 64-bit floating-point values in the compressed model header without additional decimal truncation.