C^{3}SPLAT: EXPLOITING CROSS-VIEW REDUNDANCY FOR COMPACT FEED-FORWARD 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent feed-forward 3D Gaussian Splatting (3DGS) compression methods achieve substantial rate-distortion gains over directly compressing large sets of unordered Gaussian primitives by coding multi-view intermediate features. However, these view-wise latent streams are still entropy-coded independently, overlooking cross-view redundancy and its strong dependence on inter-view geometry. We introduce C^{3}Splat, a **C**ross-view **C**ausal **C**ompression framework for Gaussian Splatting, which formulates feature compression in feed-forward 3DGS as cross-view causal entropy modeling. Each view is encoded using only previously decoded views and decoder-available geometry. C^{3}Splat aligns decoded reference features to the current view using predicted depth and camera geometry, adaptively aggregates references according to spatial reliability, and combines the resulting cross-view condition with the existing single-view hierarchical prior to refine the probability distribution used for arithmetic coding. On the 8-view DL3DV-Benchmark, C^{3}Splat achieves a PSNR BD-rate of -21.82% over the state-of-the-art method. We further find that bitrate reduction is strongly correlated with geometric view overlap, confirming that cross-view redundancy is closely tied to shared visible scene content and remains exploitable even after multi-view interaction in the reconstruction backbone. These results establish cross-view causal entropy modeling as an effective mechanism for exploiting cross-view redundancy toward more compact feed-forward 3DGS representations.

1 INTRODUCTION

Feed-forward 3D Gaussian Splatting (3DGS) reconstructs a renderable scene directly from posed images, avoiding the per-scene optimization required by conventional 3DGS (Kerbl et al., 2023; Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025; Zhang et al., 2024; Xu et al., 2024). Compact storage and transmission, however, remain essential for deploying these models at scale. Most 3DGS compression methods operate only after reconstruction, reducing an already formed set of irregular Gaussian primitives through pruning, quantization, or learned context models (Lee et al., 2024; Fan et al., 2024; Chen et al., 2024a; Wang et al., 2024; Chen et al., 2025b;a). CodecSplat (Yu et al., 2026) instead moves compression inside the reconstruction pipeline and encodes structured, view-wise intermediate features with a hierarchical VAE (Duan et al., 2023). Yet its entropy model captures only within-view dependencies, coding each view independently.

The resulting latents may still retain cross-view redundancy because they describe the same 3D scene, even after multi-view interaction in the reconstruction backbone. Once a view has been decoded, its reconstructed feature and predicted depth, together with known camera parameters, can provide side information for later views. The central question is therefore whether such residual cross-view dependencies can be exploited to improve entropy modeling of the view-wise latents used for Gaussian prediction. This setting differs from stereo and multi-view image compression (Liu et al., 2019; Deng et al., 2021; Zhai et al., 2022; Wödlinger et al., 2024; Jabbireddy et al., 2024): the coded objects are internal latents that jointly produce a single renderable 3D representation, and distortion is measured only after Gaussian prediction and novel-view rendering rather than on reconstructed input images.

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.

Exploiting this redundancy requires more than transplanting a multi-view image-codec correspondence module. Before a target latent is decoded, the decoder has neither its feature nor its depth, so it cannot backward-warp from the target view. The cross-view condition must instead be reconstructed causally from previously decoded features, reference-view depth predictions, and camera metadata already available at both codec endpoints. Moreover, occlusion, out-of-bound projections, and depth errors make reference reliability vary spatially. Finally, effectively integrating cross-view evidence with the hierarchical within-view prior remains non-trivial.

To address these challenges, we introduce C^{3}Splat, a **C**ross-view **C**ausal **C**ompression framework for Gaussian Splatting that formulates structured feature compression in feed-forward 3DGS as a cross-view causal entropy modeling problem. C^{3}Splat codes views in a deterministic causal order and conditions each current view only on previously decoded views and decoder-available metadata. Geometry-aligned learned multi-reference aggregation constructs target-aligned references and combines them according to spatially varying reliability. Prior-aware cross-view refinement then uses the aggregated evidence to refine the mean–scale distribution predicted by the single-view hierarchical prior. A coder-aligned parameterization further ensures that the encoder and decoder reproduce identical entropy-model probabilities for deterministic arithmetic coding. Experiments on DL3DV and RE10K demonstrate consistent rate–distortion improvements over CodecSplat, while entropy-only adaptation and controlled overlap analysis further verify that the gains arise from exploiting cross-view redundancy in the latent streams.

Our contributions are:

- We formulate feed-forward 3DGS feature compression as cross-view causal entropy modeling that uses only decoder-available information and requires no transmitted reference graph or auxiliary geometry.
- We develop geometry-aligned learned multi-reference aggregation and prior-aware cross-view refinement to select spatially reliable evidence from previously decoded views and use it to refine the single-view hierarchical prior, together with a coder-aligned parameterization for deterministic arithmetic coding.
- Entropy-only adaptation isolates gains from cross-view probability modeling, while joint finetuning achieves PSNR BD-rates of -21.82% on DL3DV against CodecSplat. A controlled overlap analysis further verifies the mechanism behind these gains.

2 Related Work

2.1 Feed-Forward 3D Gaussian Splatting

Feed-forward 3DGS replaces per-scene optimization with direct Gaussian prediction: Splatter Image (Szymanowicz et al., 2024) maps pixels to Gaussians, and pixelSplat (Charatan et al., 2024) generalizes this paradigm to sparse-view reconstruction. MVSplat (Chen et al., 2024b), MVSGaussian (Liu et al., 2024a), and DepthSplat (Xu et al., 2025) improve localization through cost volumes, multi-view matching, and monocular depth priors, while GS-LRM (Zhang et al., 2024) and GRM (Xu et al., 2024) model cross-view relations with Transformers. Long-LRM (Chen et al., 2025c) and iLRM (Kang et al., 2026) scale input count, resolution, and scene extent, and NoPoSplat (Ye et al., 2025a) removes the requirement for known camera extrinsics. These methods emphasize reconstruction quality and scalability; compact storage of their representations remains less explored.

2.2 3D Gaussian Splatting Compression

Compact 3D Gaussian Representation (Lee et al., 2024) and LightGaussian (Fan et al., 2024) combine pruning and attribute quantization, while HAC (Chen et al., 2024a), ContextGS (Wang et al., 2024), and HAC++ (Chen et al., 2025b) model spatial, hierarchical, and inter-/intra-anchor dependencies. FCGS (Chen et al., 2025a) provides generalizable compression without per-scene codec optimization, and TinySplat (Song et al., 2026) removes redundancy from feed-forward Gaussians without training. These approaches compress final Gaussian primitives; CodecSplat (Yu et al., 2026) instead hierarchically codes structured 2D Gaussian-generation features inside the reconstruction pipeline, while GenSplatCodec (Hu et al., 2026) uses geometry-guided generative decoding at very

low rates. Cross-view statistical redundancy induced by shared geometry nevertheless remains underexploited.

2.3 Multi-View Image Compression

Multi-view image codecs exploit shared content through disparity in DSIC (Liu et al., 2019), homography alignment in HESIC (Deng et al., 2021), latent shifts and stereo attention in SASIC (Wödlinger et al., 2022), and disparity-aligned priors in DispSIC (Zhai et al., 2022). Decoder-side and distributed formulations further exploit correlated views through side information and cross-view context (Ayzik & Avidan, 2020; Zhang et al., 2023; Huang et al., 2023). ECSIC (Wödlinger et al., 2024) uses epipolar attention, VEMIC (Jabbireddy et al., 2024) conditions entropy models on decoded views, and BiSIC (Liu et al., 2024b) and CAMSIC (Zhang et al., 2025) jointly model spatial, channel, and inter-view correlations. Related conditional entropy models also exploit previously decoded observations in learned video compression (Liu et al., 2020). These methods generally use image- or latent-domain correspondence rather than explicit 3D geometry; 3D-LMVIC (Huang et al., 2025) introduces a geometric prior but must construct and transmit an auxiliary 3DGS representation. C^{3}Splat instead reuses decoded features, predicted depth, and camera parameters already available in feed-forward 3DGS to construct decoder-reproducible, target-aligned conditions.

3 Preliminaries

3.1 Feature Compression in CodecSplat

Given N posed context views $\mathcal{C}$, the DepthSplat-style FF3DGS backbone (Xu et al., 2025) combines image and cross-view matching features, encoder-side depth, and depth matching-probability features into dense, view-aligned Gaussian-generation features. We omit the view index throughout this subsection and denote the feature of a representative view by F_g. CodecSplat's pre-codec refiner (Yu et al., 2026) applies depth-guided multi-view interaction to obtain F_c. Its per-view codec path is $F_g \to F_c \to \widetilde{Z} \to Z \to \hat{F}_c$. The analysis transform maps F_c to the pre-quantization latent $\widetilde{Z}$, quantization yields symbols $Z = Q(\widetilde{Z})$ for entropy coding, and the synthesis transform reconstructs $\hat{F}_c$. Applying this path to all context views produces the reconstructed features used for codec-side depth prediction, post-codec refinement, Gaussian parameter prediction, and 2D-to-3D mapping to obtain $\hat{\mathcal{G}}$. The scene bitstream packs compressed camera metadata and one feature stream of Z per context view.

3.2 Hierarchical VAE

Hierarchical and contextual priors are widely used in learned compression to capture dependencies among quantized latents and improve entropy modeling (Ballé et al., 2018; Minnen et al., 2018). We next summarize the hierarchical VAE used by CodecSplat. Each quantized latent Z is organized into L latent levels, $Z = \{Z^\ell\}_{\ell=0}^{L-1}$, ordered from coarse to fine. Following quantized hierarchical VAEs (Duan et al., 2023), decoding proceeds sequentially: before decoding level ℓ, the previously decoded coarser latents $Z^{<\ell}$ are summarized by a synthesis state h^ℓ. The hierarchical prior therefore factorizes as

$$p_0\left(Z \mid e_\lambda\right) = \prod_{\ell=0}^{L-1} p_0\left(Z^\ell \mid h^\ell, e_\lambda\right), \tag{1}$$

where e_λ denotes the rate-control embedding. At level ℓ, the prior network P_0^ℓ maps (h^ℓ, e_λ) to the mean and raw-scale parameters $(\mu_0^\ell, \rho_0^\ell)$ of the entropy model for Z^ℓ. After Z^ℓ is decoded, it updates the synthesis state for the next finer level, and the final synthesis state reconstructs the feature. This coarse-to-fine factorization captures dependencies across latent scales while retaining sequential entropy decoding.

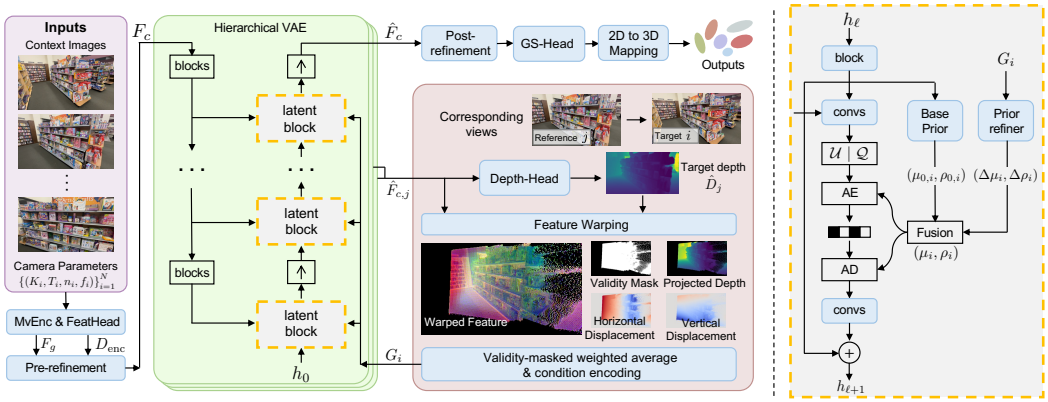


Figure 1: Overview of C^{3}Splat. Previously decoded views are geometrically aligned and reliability-weighted to construct multi-scale cross-view conditions G_i, which refine the hierarchical entropy prior at each latent level. The right inset illustrates how G_i is fused with the base prior to obtain the final mean–scale distribution for entropy coding.

4 Method

4.1 Overview

C^{3}Splat builds on the feed-forward 3DGS backbone and hierarchical feature codec described in Section 3. For each input view i, the Gaussian-generation feature $F_{c,i}$ is represented by a hierarchy of quantized latents $Z_i = \{Z_i^\ell\}_{\ell=0}^{L-1}$, ordered from coarse to fine. Before decoding level ℓ, the already decoded coarser latents $Z_i^{<\ell}$ are summarized by the hierarchical synthesis state h_i^ℓ; together with the rate-control embedding e_λ, they form the single-view context used by the base entropy model. The base codec applies this hierarchical prior independently to each view, leaving redundancies across views unmodeled. C^{3}Splat additionally introduces a deterministic causal view order $\pi = (\pi_1, \ldots, \pi_N)$. When coding the current view $i = \pi_t$, the previously decoded views $\mathcal{H}_i = \{\pi_1, \ldots, \pi_{t-1}\}$ are used as cross-view references. From their decoded features, predicted depths, and camera parameters, C^{3}Splat constructs level-specific cross-view conditions $G_i = \{G_i^\ell\}_{\ell=0}^{L-1}$ that are available at the latent block where Z_i^ℓ is decoded. The resulting joint distribution factorizes as

$$p_{\mathrm{C}^3}(Z_{1:N} \mid e_\lambda) = \prod_{\ell=0}^{L-1} p_0\big(Z_{\pi_1}^\ell \mid h_{\pi_1}^\ell, e_\lambda\big) \times \prod_{t=2}^{N} \prod_{\ell=0}^{L-1} p_{\mathrm{C}^3}\big(Z_{\pi_t}^\ell \mid h_{\pi_t}^\ell, G_{\pi_t}^\ell, e_\lambda\big)\,. \tag{2}$$

The first view therefore follows the original single-view hierarchical prior, whereas every subsequent view additionally exploits information reconstructed from the decoded history. Because both the causal order and the cross-view conditions are derived entirely from decoder-available quantities, this additional conditioning requires no extra side-information bits.

This formulation also admits a simple information-theoretic interpretation. For simplicity, view and latent-level indices are omitted, and Z, C, and G denote the current latent, its single-view hierarchical context, and the corresponding cross-view condition, respectively. Under ideal entropy modeling, rate reduction enabled by cross-view conditioning is

$$\Delta R_{\mathrm{ideal}} = H(Z \mid C) - H(Z \mid C, G) = I(Z; G \mid C) \geq 0. \tag{3}$$

Thus, C^{3}Splat exploits the residual cross-view information about the current latent that is not already captured by the single-view hierarchical prior. This perspective is related to classical source coding with correlated side information (Slepian & Wolf, 1973; Wyner & Ziv, 1976), although C^{3}Splat operates in a causal setting where the cross-view condition is reconstructed from previously decoded views and is available at both codec endpoints. In practice, the learned entropy model only approximates these conditional distributions, so Eq. 3 characterizes the ideal coding opportunity rather than guaranteeing a reduction in realized bitrate. As illustrated in Figure 1, C^{3}Splat realizes this principle through geometry-aligned learned multi-reference aggregation, prior-aware refinement

of the base entropy model, and deterministic causal ordering. The following subsections describe these components in detail.

4.2 Geometry-Aligned Learned Multi-Reference Aggregation

Features from different camera views are not pixel-wise aligned, so directly conditioning the target entropy model on historical features or latents may associate the same image location with different 3D content. We therefore explicitly align each decoded reference to the current target view. For target $i = \pi_t$ with decoded history $\mathcal{H}_i = \{\pi_1, \ldots, \pi_{t-1}\}$, let $\hat{F}_{c,j}$ and $\hat{D}_j$ denote the reconstructed feature and predicted depth of reference $j \in \mathcal{H}_i$. Using camera intrinsics K_j, K_i and camera extrinsics T_j, T_i, we compute the relative projection geometry $d_{j\to i} = \mathcal{G}(K_j, T_j, K_i, T_i)$ and forward-splat the decoded feature into the target frame, $W_{j\to i} = \mathcal{W}(\hat{F}_{c,j}, \hat{D}_j, d_{j\to i})$. Forward splatting is essential here because target depth is unavailable before the target view is decoded, whereas the reconstructed reference feature, reference depth, and camera parameters are already available at both codec endpoints. Following CodecSplat (Yu et al., 2026), projected source features are distributed to neighboring target pixels with bilinear weights. When multiple source points project to the same target location, their contributions are further weighted by $\exp(-d/\tau)$ according to projected target depth d, favoring nearer surfaces and suppressing occluded or conflicting projections.

Geometric alignment alone does not make all references equally informative. Their reliability varies spatially due to occlusion, out-of-bound projections, depth errors, and view-dependent appearance changes. For each aligned reference, we therefore construct a local geometry descriptor $R_{j\to i}$ containing projection validity, normalized image-plane displacement, and projected target depth, with $M_{j\to i}$ denoting the corresponding validity mask. A lightweight scoring network shared across references predicts a per-pixel reliability score $s_{j,i}(u) = S(W_{j\to i}(u), R_{j\to i}(u))$ from both warped appearance and projection geometry. These scores are normalized across the available history using a validity-masked softmax:

$$\alpha_{j,i}(u) = \begin{cases} \dfrac{M_{j\to i}(u)\exp(s_{j,i}(u))}{\sum_{k\in\mathcal{H}_i} M_{k\to i}(u)\exp(s_{k,i}(u))}, & \text{if } \sum_{k\in\mathcal{H}_i} M_{k\to i}(u) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Thus, invalid references receive zero weight, while valid references compete independently at each spatial location according to their predicted reliability. A single valid reference naturally reduces the formulation to single-reference conditioning.

The aligned references are then combined as

$$W_i(u) = \sum_{j\in\mathcal{H}_i} \alpha_{j,i}(u) W_{j\to i}(u), \tag{5}$$

yielding a target-aligned representation that integrates complementary evidence from the decoded history. A condition encoder is applied once to the aggregated feature to obtain the multi-scale cross-view conditions $G_i = \{G_i^\ell\}_{\ell=0}^{L-1} = E_c(W_i)$, which are supplied to the corresponding hierarchical latent levels. Aggregating references before condition encoding preserves spatially adaptive reference selection while avoiding a separate encoder pass for every historical view. The scoring network is initialized to produce uniform weights over valid references, so training starts from valid-mask averaging and learns deviations only when they reduce the coding objective.

4.3 Prior-Aware Cross-View Refinement

After multi-reference aggregation and condition encoding, G_i^ℓ provides geometry-aligned cross-view evidence for latent level ℓ. Since the usefulness of this evidence depends on what has already been captured by the single-view hierarchical prior, we refine the base distribution rather than predicting a new distribution from G_i^ℓ alone. The following formulation applies independently to each latent level; we omit ℓ for clarity. Given hierarchical state h_i, the base prior predicts $(\mu_{0,i}, \rho_{0,i}) = P_0(h_i, e_\lambda)$. A level-specific refinement network then predicts residual corrections $(\Delta\mu_i, \Delta\rho_i) = R(G_i, \mu_{0,i}, \rho_{0,i})$, yielding $(\mu_i, \rho_i) = (\mu_{0,i}, \rho_{0,i}) + (\Delta\mu_i, \Delta\rho_i)$. Thus, the cross-view condition complements the existing hierarchical prior by refining its predicted distribution center and uncertainty.

The cross-view path modifies only the entropy model; the target posterior, synthesis transform, and reconstructed latent representation remain unchanged. The refined parameters are converted to the

mean and scale indices required by the arithmetic coder using the same deterministic quantization and CDF construction at both encoder and decoder. Detailed parameter transforms, CDF construction, and network configurations are provided in Appendices A and B.

4.4 Causal Ordering and Bitstream

Deterministic causal order C^{3}Splat must recover the same view order at both codec endpoints without transmitting a reference graph. Letting t_i denote camera center i, the camera medoid $\pi_1 = \arg\min_i \sum_j \|t_i - t_j\|_2$ is chosen as the first view, coded independently, and used as the first reference. For every remaining view, its minimum camera-center distance to the decoded set is computed, and the next target is selected by a deterministic greedy nearest-set rule. Both the medoid and subsequent selections break ties by the smaller original view index, allowing the encoder and decoder to reconstruct the same π from camera parameters already present in the bitstream. Selecting views near the decoded set progressively expands the visible region available to future targets, while learned aggregation suppresses unreliable warps and combines complementary history.

Bitstream organization The complete scene bitstream packs camera metadata followed by causally ordered view streams: $B = \mathrm{Pack}(B_{\mathrm{cam}}, B_{\pi_1}, \ldots, B_{\pi_N})$, where B_{cam} contains the image shape, view count, camera intrinsics, camera extrinsics, and near/far bounds. The decoder reconstructs the order and decodes each stream in turn; each decoded feature $\hat{F}_c$ and predicted depth $\hat{D}$ then enter the history for subsequent targets. Appendix C gives the complete causal encoding procedure and scene bitstream layout.

4.5 Training Objectives

Stage 1: Entropy-only adaptation. Starting from the corresponding pretrained CodecSplat checkpoint, this stage optimizes only the newly introduced cross-view conditioning and entropy-modeling modules for the estimated rate of quantized latents. The pretrained reconstruction path is frozen, so the quantized latent representation and reconstruction lattice remain unchanged. The objective is

$$\mathcal{L} = \mathcal{R} = \sum_{i,\ell,u} -\log_2 p\left(\hat{Z}_i^\ell(u) \mid h_i^\ell(u), G_i^\ell, e_\lambda\right), \tag{6}$$

where u indexes spatial and channel positions in latent tensor ℓ, $h_i^\ell(u)$ is the hierarchical state used to produce the single-view base prior, G_i^ℓ is the aligned cross-view condition, and e_λ is the rate-control embedding. This objective directly improves mean–scale prediction for the symbols being coded.

Stage 2: Rate-distortion joint finetuning. Starting from the entropy-only model, the codec, reconstruction-related modules, and cross-view entropy model are jointly finetuned under a rate–distortion objective:

$$\mathcal{L}_{\mathrm{stage2}}(\lambda) = \frac{\lambda}{\lambda_{\max}}\left[\mathcal{D}_{\mathrm{render}}\left(\hat{\mathbf{Y}}_t, \mathbf{Y}_t\right) + \gamma \mathcal{D}_{\text{codec-depth}}\right] + \beta\mathcal{R}, \tag{7}$$

where $\hat{\mathbf{Y}}_t = \mathrm{Render}(\hat{\mathcal{G}}, K_t, T_t)$ is the target view rendered from decoded Gaussians, $\mathcal{D}_{\mathrm{render}}$ combines RGB MSE and weighted LPIPS, $\mathcal{D}_{\text{codec-depth}}$ is normalized inverse-depth ℓ_1 loss against an encoder-side depth teacher, and $\mathcal{R}$ is the estimated latent rate from equation 6. The objective jointly adapts probability modeling and reconstruction to quantization error.

4.6 Complexity Analysis

Let N be the number of input views and let $\mathcal{C}_{\mathrm{align}}$, $\mathcal{C}_{\mathrm{cond}}$, and $\mathcal{C}_{\mathrm{refine}}^\ell$ denote the costs of processing one reference, encoding one aggregated condition, and refining latent level ℓ. Since target π_t uses $t-1$ references, the additional scene-level computation is

$$\Delta\mathcal{C} = \frac{N(N-1)}{2}\mathcal{C}_{\mathrm{align}} + (N-1)\left(\mathcal{C}_{\mathrm{cond}} + \sum_{\ell=0}^{L-1}\mathcal{C}_{\mathrm{refine}}^\ell\right). \tag{8}$$

For an $H \times W$ feature map with C channels, alignment over the full causal history requires $\mathcal{O}(N^2HWC)$ work. Aggregating before condition encoding evaluates E_c once per target, reducing

its cost from $\mathcal{O}(N^2\mathcal{C}_{\text{cond}})$ to $\mathcal{O}(N\mathcal{C}_{\text{cond}})$. The shared parameters are independent of N, require $\mathcal{O}(NHWC)$ peak reference workspace, and introduce no additional bitstream side information.

5 Experiments

5.1 Experimental Setup

Datasets and metrics. We evaluate C^{3}Splat on DL3DV (Ling et al., 2024) and RealEstate10K (RE10K) (Zhou et al., 2018), following the CodecSplat (Yu et al., 2026) protocol. For DL3DV, we follow the ReSplat (Xu et al., 2026) 8-input-view benchmark protocol at 256×448 resolution. For RE10K, we use the common 2-input-view protocol at 256×256 resolution (Xu et al., 2025; 2026). All methods share the same scene splits, context views, and target views. We report PSNR, SSIM, and LPIPS on rendered target views and measure storage by the arithmetic bitstream length.

Training setup. Models initialize from the corresponding public CodecSplat checkpoint and retain its feed-forward Gaussian backbone, hierarchical VAE, and variable-rate configuration. Residual or zero initialization makes the new cross-view branches initially match CodecSplat's behavior. In Stage 1 (entropy-only adaptation), only cross-view condition construction and probability prediction are updated, while the reconstruction path and quantized symbols remain fixed. Stage 2 (rate-distortion joint finetuning) starts from the entropy-only model and jointly updates the codec, decoded-geometry and Gaussian prediction paths, and cross-view entropy modules with a maximum learning rate of 10^{-6}; the multi-view backbone remains frozen. Each scene shares one sampled $\lambda \in [16, 1024]$ across its input views, with $\lambda_{\max} = 1024$. On DL3DV, $\gamma = 0.1$, $\beta = 10^{-4}$, and the LPIPS weight in $\mathcal{D}_{\text{render}}$ is 0.05.

Baselines and rate accounting. We compare against CodecSplat and the Gaussian-output baselines reported with it (Yu et al., 2026), which pair DepthSplat or ReSplat with FCGS (Chen et al., 2025a), gsplat PNG Compression (Ye et al., 2025b), or SOGS (Morgenstern et al., 2025). Rate is the average per-scene bitstream length, including decoder-required camera metadata and all view-wise latent streams for C^{3}Splat and CodecSplat, and the complete compressed Gaussian representation for the Gaussian-output baselines.

5.2 Main Rate-Distortion Results



Figure 2: Qualitative and quantitative comparison on DL3DV. Each row shows a context view, target view, and renderings from ReSplat→FCGS, CodecSplat, and C^{3}Splat. PSNR and total bitstream size are shown in each result.

Table 1 compares C^{3}Splat against CodecSplat and Gaussian-output compression baselines at representative operating points, while Appendix D reports all evaluated variable-rate points. C^{3}Splat provides substantially higher storage efficiency than pipelines that combine DepthSplat or ReSplat with a separate 3DGS compressor. Figure 2 shows that these quantitative gains are achieved while preserving or improving visual reconstruction quality. We evaluate C^{3}Splat at both stages of our two-stage training procedure: entropy-only adaptation and subsequent rate–distortion joint finetuning. The entropy-only stage isolates the rate reduction attributable to cross-view probability modeling, whereas joint

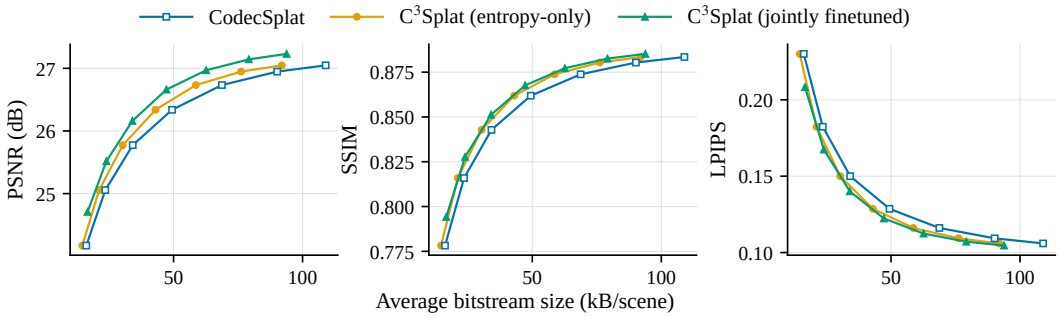


Figure 3: Rate–distortion curves on DL3DV. Entropy-only adaptation lowers rate without changing reconstruction metrics, while subsequent rate–distortion joint finetuning further improves the overall trade-off.

Table 1: Quantitative comparison on DL3DV and RE10K scenes at two representative learned-codec operating points, with Gaussian-output baselines included for context. Rates are average total bitstream sizes per scene.

Method	Codec Setting	DL3DV				RE10K			
		Rate (kB)↓	PSNR↑	SSIM↑	LPIPS↓	Rate (kB)↓	PSNR↑	SSIM↑	LPIPS↓
DepthSplat→PNG	$K = 8192$	12226.4	24.94	0.848	0.128	1719.7	27.28	0.885	0.117
DepthSplat→SOGS	Default	12037.7	24.94	0.848	0.128	1709.2	27.28	0.885	0.116
DepthSplat→FCGS	$\lambda = 0.0016$	12666.8	24.93	0.847	0.129	1918.9	26.01	0.872	0.146
ReSplat→PNG	$K = 8192$	880.8	22.75	0.843	0.172	1887.4	25.90	0.887	0.122
ReSplat→SOGS	Default	901.8	28.36	0.894	0.120	1887.4	29.36	0.908	0.102
ReSplat→FCGS	$\lambda = 0.0016$	1457.5	28.72	0.897	0.113	2590.0	29.16	0.905	0.109
CodecSplat	$\lambda = 64$	34.18	25.77	0.843	0.150	5.11	26.08	0.854	0.150
	$\lambda = 1024$	108.95	27.05	0.883	0.106	11.98	26.82	0.875	0.118
C^{3}Splat (entropy-only)	$\lambda = 64$	30.34	25.77	0.843	0.150	5.00	26.08	0.854	0.150
	$\lambda = 1024$	91.94	27.05	0.883	0.106	11.55	26.82	0.875	0.118
C^{3}Splat (jointly finetuned)	$\lambda = 64$	34.04	26.16	0.851	0.140	5.65	26.30	0.859	0.140
	$\lambda = 1024$	93.81	27.23	0.885	0.105	12.45	26.85	0.876	0.117

finetuning adapts both the entropy and reconstruction paths. Figure 3 compares the rate–distortion curves on the DL3DV benchmark using average arithmetic bitstream length. Entropy-only adaptation consistently reduces the total bitstream size while preserving exactly the same reconstruction metrics as CodecSplat. Subsequent joint finetuning further improves the overall rate–distortion trade-off, yielding PSNR BD-rates of -21.82% on DL3DV and -2.97% on RE10K relative to CodecSplat. It improves both rate and reconstruction quality on DL3DV, and improves reconstruction quality at a modest rate increase on RE10K.

5.3 Ablation Studies

Table 2 summarizes three ablations that isolate the effects of learned multi-reference aggregation, prior-aware refinement, and causal multi-reference history.

Valid-Mask Mean. Replacing learned aggregation with validity-mask averaging increases the rate in both training stages. This comparison preserves geometric alignment, the reference set, and validity masking while removing learned reliability scores, showing that the aggregation module distinguishes spatially varying reference quality beyond projection validity alone.

Condition-Only Refinement. Predicting the probability correction solely from G_i, without exposing the base prior $(\mu_{0,i}, \rho_{0,i})$ to the refinement network, increases the rate in both training stages. This

result isolates the role of prior awareness: cross-view evidence is more effective when interpreted relative to the uncertainty already captured by the hierarchical prior.

First Reference Only. Restricting every target to the first decoded view π_1, rather than all previously decoded views, increases the rate in both training stages. This comparison isolates the contribution of causal multi-reference history and shows that later views provide complementary information unavailable from a fixed anchor alone.

Table 2: Ablation studies on DL3DV at $\lambda = 1024$; rates are reported in kB. The entropy-only column reports the bitstream rate after entropy-only adaptation, while the remaining columns report performance after rate-distortion joint finetuning. Each ablated variant changes only one component from the jointly finetuned model.

	entropy-only	jointly finetuned			
Variant	Rate (kB)↓	Rate (kB)↓	PSNR↑	SSIM↑	LPIPS↓
C^{3}Splat (full)	91.94	93.81	27.23	0.885	0.105
Valid-Mask Mean	94.61	96.67	27.23	0.885	0.104
Condition-Only Refinement	96.35	97.86	27.22	0.885	0.105
First Reference Only	99.42	103.45	27.23	0.885	0.104

5.4 Redundancy Analysis

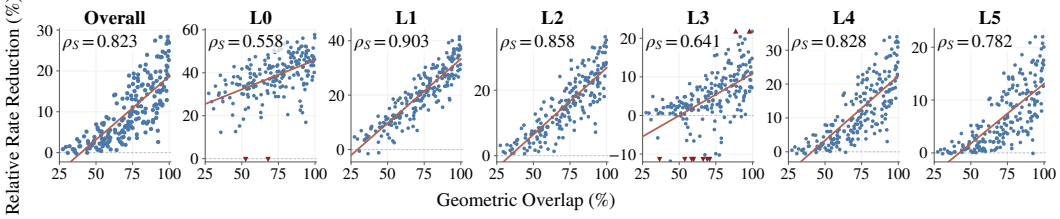


Figure 4: Cross-view redundancy analysis on 203 target views from the DL3DV test split at $\lambda = 1024$. The panels show the overall relative reduction in target-only estimated latent bitrate and the corresponding reductions at each of the six hierarchical latent levels versus geometric overlap with the first reference. Each line is a linear regression fit, and ρ_S denotes Spearman correlation. Outward-pointing triangles denote samples outside the displayed Tukey-fenced ranges.

This analysis tests whether rate reduction grows with the visible content shared with a decoded reference while the quantized symbols and reconstruction path remain fixed. We use the entropy-only model restricted to the first decoded reference π_1, giving each target a single interpretable reference. At $\lambda = 1024$, overlap is the valid-pixel fraction of the reference-to-target forward-splatting mask, and rate reduction is evaluated under the single-view and cross-view priors on the same quantized symbols. Figure 4 shows that larger overlap yields larger target-only rate reductions, with positive trends at every latent level. This controlled dependence supports the main claim that geometry-aligned decoded references predict later-view latents rather than merely adding entropy-model capacity.

6 Conclusion

C^{3}Splat exploits cross-view redundancy among feed-forward 3DGS latents through causal entropy modeling. It aligns decoded features using camera geometry, aggregates reliable evidence from previously decoded views, and refines the single-view hierarchical prior with decoder-reproducible conditions. Entropy-only adaptation isolates the probability-modeling gain, while joint rate–distortion finetuning achieves PSNR BD-rates of -21.82% on DL3DV and -2.97% on RE10K against Codec-Splat. Bitrate reduction also grows with geometric overlap, confirming that cross-view redundancy remains exploitable after multi-view interaction in the reconstruction backbone.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, manuscript organization, formatting, and software development, as well as to provide feedback on experimental design and result presentation. All research decisions, implementations, experiments, and scientific conclusions were reviewed and verified by the authors. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide the model architecture and entropy-coding formulation in Section 4, including the causal view ordering, geometry-aligned reference construction, and coder-aligned probability parameterization. Section 5.1 specifies the datasets, evaluation protocols, training procedure, and rate accounting used in our experiments. Appendices A–C provide the entropy-table construction, network details, and bitstream procedure, while Appendix D reports all evaluated rate–distortion operating points. All reported rates are measured from actual arithmetic-coded bitstreams under the same benchmark splits and view settings. An anonymized implementation is available at `https://anonymous.4open.science/r/C3Splat-7E1C`.

REFERENCES

Sharon Ayzik and Shai Avidan. Deep image compression using decoder side information. In *European Conference on Computer Vision (ECCV)*, pp. 699–714, 2020.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19457–19467, 2024.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision (ECCV)*, pp. 422–438, 2024a.

Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3d gaussian splatting compression. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac++: Towards 100x compression of 3d gaussian splatting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(11):10210–10226, 2025b.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision (ECCV)*, volume 15079 of *Lecture Notes in Computer Science*, pp. 370–386, 2024b.

Ziwen Chen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Fuxin Li, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4349–4359, 2025c.

Xin Deng, Wenzhe Yang, Ren Yang, Mai Xu, Enpeng Liu, Qianhan Feng, and Radu Timofte. Deep homography for efficient stereo image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1492–1501, 2021.

Zhihao Duan, Ming Lu, Zhan Ma, and Fengqing Zhu. Lossy image compression with quantized hierarchical vaes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 198–207, 2023.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. In *Advances in Neural Information Processing Systems*, volume 37, pp. 140138–140158, 2024.

Qiang Hu, Zhenlong Wu, Lei Huang, Zihan Zheng, Xiaoyun Zhang, and Wenjun Zhang. Gensplatcodec: Feed-forward gaussian splatting compression via one-step diffusion. *arXiv preprint arXiv:2607.24403*, 2026.

Yujun Huang, Bin Chen, Shiyu Qin, Jiawei Li, Yaowei Wang, Tao Dai, and Shu-Tao Xia. Learned distributed image compression with multi-scale patch matching in feature domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4322–4329, 2023.

Yujun Huang, Bin Chen, Niu Lian, Xin Wang, Baoyi An, Tao Dai, and Shu-Tao Xia. 3D-LMVIC: Learning-based multi-view image compression with 3d gaussian geometric priors. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 25141–25158. PMLR, 2025.

Susmija Jabbireddy, Davit Soselia, Max Ehrlich, Christopher Metzler, and Amitabh Varshney. VEMIC: View-aware entropy model for multi-view image compression. In *35th British Machine Vision Conference 2024, BMVC 2024*. BMVA, 2024.

Gyeongjin Kang, Seungtae Nam, Seungkwon Yang, Xiangyu Sun, Sameh Khamis, Abdelrahman Mohamed, and Eunbyung Park. ilrm: An iterative large 3d reconstruction model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 37332–37342, 2026.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

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

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22160–22169, 2024.

Jerry Liu, Shenlong Wang, and Raquel Urtasun. DSIC: Deep stereo image compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3136–3145, 2019.

Jerry Liu, Shenlong Wang, Wei-Chiu Ma, Meet Shah, Rui Hu, Pranaab Dhawan, and Raquel Urtasun. Conditional entropy coding for efficient video compression. In *European Conference on Computer Vision (ECCV)*, pp. 453–468, 2020.

Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In *European Conference on Computer Vision (ECCV)*, volume 15076 of *Lecture Notes in Computer Science*, pp. 37–53. Springer, 2024a.

Zhening Liu, Xinjie Zhang, Jiawei Shao, Zehong Lin, and Jun Zhang. Bidirectional stereo image compression with cross-dimensional entropy model. In *European Conference on Computer Vision (ECCV)*, pp. 480–496, 2024b.

David Minnen, Johannes Ballé, and George D. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. In *European Conference on Computer Vision (ECCV)*, pp. 18–34. Springer Nature Switzerland, 2025.

David Slepian and Jack K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, 1973.

Zetian Song, Jiaye Fu, Jiaqi Zhang, Xiaohan Lu, Chuanmin Jia, Siwei Ma, and Wen Gao. Tinysplat: Feedforward approach for generating compact 3d scene representation. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(4):5567–5580, 2026.

Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10208–10217, 2024.

Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C. Kot, and Bihan Wen. Contextgs: Compact 3d gaussian splatting with anchor level context model. In *Advances in Neural Information Processing Systems*, volume 37, pp. 51532–51551, 2024.

Matthias Wödlinger, Jan Kotera, Jan Xu, and Robert Sablatnig. Sasic: Stereo image compression with latent shifts and stereo attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 661–670, 2022.

Matthias Wödlinger, Jan Kotera, Manuel Keglevic, Jan Xu, and Robert Sablatnig. ECSIC: Epipolar cross attention for stereo image compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3424–3433, 2024.

Aaron D. Wyner and Jacob Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, 1976.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16453–16463, 2025.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. ReSplat: Learning recurrent gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2026.

Yinghao Xu, Zifan Shi, Yifan Wang, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In *European Conference on Computer Vision (ECCV)*, pp. 1–20, 2024.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In *International Conference on Learning Representations*, 2025a.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025b.

Pengpeng Yu, Runqing Jiang, Qi Zhang, Dingquan Li, Jing Wang, and Yulan Guo. Codecsplat: Ultra-compact latent coding for feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2605.25563*, 2026.

Yongqi Zhai, Luyang Tang, Yi Ma, Rui Peng, and Ronggang Wang. Disparity-based stereo image compression with aligned cross-view priors. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 2351–2360, 2022.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, volume 15080 of *Lecture Notes in Computer Science*, pp. 1–19. Springer, 2024.

Xinjie Zhang, Jiawei Shao, and Jun Zhang. Ldmic: Learning-based distributed multi-view image coding. In *International Conference on Learning Representations*, 2023.

Xinjie Zhang, Shenyuan Gao, Zhening Liu, Jiawei Shao, Xingtong Ge, Dailan He, Tongda Xu, Yan Wang, and Jun Zhang. CAMSIC: Content-aware masked image modeling transformer for stereo image compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 10239–10247, 2025.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4), 2018.

A CODER-ALIGNED MEAN–SCALE ENTROPY CODING

The learned probability model must be converted into deterministic arithmetic-coding tables. The neural network outputs continuous probability parameters, but both endpoints must construct exactly the same CDF for every discrete symbol. In the continuous mean–scale mode, the refinement's mean correction $c = \mathrm{clip}(\Delta\mu, -16, 16)$ is quantized rather than the final mean that includes μ_0. Let $Q_\phi = 16$. The value $q = \mathrm{round}(Q_\phi c)$ is first computed and decomposed as

$$k = \left\lfloor \frac{q + Q_\phi/2}{Q_\phi} \right\rfloor, \qquad \delta = \frac{q - kQ_\phi}{Q_\phi}, \qquad r_\phi = q - kQ_\phi + Q_\phi/2. \tag{9}$$

Here, k is an integer center shift applied to the canonical integer symbol, $\delta \in \{-8/16, \ldots, 7/16\}$ is the fractional mean phase used by the CDF, and $r_\phi \in \{0, \ldots, 15\}$ is the phase index. The canonical symbol is $n = \mathrm{round}(z - \mu_0)$; the arithmetic coder receives $n - k$, and the decoder adds k back after decoding. This branch therefore preserves the reconstruction lattice anchored at μ_0. The raw scale parameter is transformed into a positive σ by the softplus–exponential mapping and then mapped to a shared scale-table index r_σ. The final CDF row is selected in scale-major order by $r_\sigma Q_\phi + r_\phi$.

For a given integer symbol n, the discrete Gaussian probability mass is

$$p(n) = \Phi\left(\frac{n + \frac{1}{2} - \delta}{\sigma}\right) - \Phi\left(\frac{n - \frac{1}{2} - \delta}{\sigma}\right), \tag{10}$$

where $\Phi(\cdot)$ denotes the standard Gaussian CDF. The zero-phase CDF row directly reuses the original zero-mean coder table. Arithmetic coding requires matching discrete phase and scale indices, not bitwise-identical intermediate floating-point values; nevertheless, any numerical perturbation that crosses a quantization boundary could change the index, so both endpoints use the same condition-construction path.

The shared scale table contains 64 values logarithmically spaced from 0.11 to 20. Combining these values with the 16 mean phases gives 1024 CDF rows per latent block. Each row is constructed with a tail-mass target of 10^{-9}. The zero-phase row retains the original symmetric support, while nonzero phases receive one guard bin at each end before the tail mass is folded into the finite table. The predicted scale is mapped to this shared table before probability evaluation and arithmetic coding, so rate estimation and the deployed coder use the same discrete scale.

For each latent level, the encoder first evaluates the base hierarchical prior and the cross-view refinement. It then quantizes the mean correction into (k, r_ϕ), maps the scale to r_σ, and arithmetic-codes the shifted canonical symbol $n - k$ with CDF row $r_\sigma Q_\phi + r_\phi$. The decoder reconstructs the same indices from already decoded quantities, decodes $n - k$, restores n, and reconstructs the latent on the original base-prior lattice. Consequently, entropy-only adaptation can change the code length without changing the decoded latent symbols or feature reconstruction.

B ARCHITECTURE AND IMPLEMENTATION DETAILS

Base feature codec and rate embedding. The Gaussian-generation feature entering the codec has 64 channels. Before analysis, its spatial dimensions are padded by boundary replication to multiples of 64; the original dimensions are stored in each view header and used to crop the synthesized feature after decoding. The rate-control value λ is embedded by first forming a 256-dimensional sinusoidal code from $\log \lambda / \log 8192$ and then applying a two-layer MLP with GELU nonlinearities, producing a 256-dimensional embedding. The same embedding modulates the analysis, synthesis, and condition-encoder ConvNeXt-AdaLN blocks.

The analysis transform applies non-overlapping convolutional downsampling with factors $(4, 2, 2, 2, 2)$ and widths $(192, 384, 512, 512, 512)$. Table 3 lists the exact stage layout. Stored keys are captured before the optional post-key ConvNeXt block and are later consumed by the corresponding latent posterior and conditional prior paths.

Table 3: Analysis and condition-encoder stage layout. Kernel entries give the spatial kernels of ConvNeXt-AdaLN blocks. The condition encoder uses the same stage layout with independent parameters.

Output stride	Width	Downsample	Pre-key kernels	Key / post-key kernels
4	192	4×	7, 7, 7, 7	– / –
8	384	2×	7, 7, 7	enc_s8 / 7
16	512	2×	5, 5, 5	enc_s16 / 7
32	512	2×	3, 3	enc_s32 / 7
64	512	2×	1, 1	enc_s64 / –

The synthesis transform starts from a learned 512-channel bias at stride 64 and progressively upsamples with PixelShuffle. Its stage widths are $(512, 512, 384, 256, 128)$ before the final $4\times$ upsampling to the 64-channel feature reconstruction. Six latent blocks inject information in coarse-to-fine order at strides $(64, 32, 16, 16, 8, 8)$, with latent widths $(32, 32, 64, 64, 16, 16)$. Two distinct latent blocks are used at strides 16 and 8; blocks at the same stride receive the same condition-encoder key but maintain independent hierarchical states, base priors, posterior transforms, and refinement parameters.

Latent block internals. Each latent block first applies a ConvNeXt-AdaLN block to the current synthesis state and predicts the base mean and raw-scale tensors with a 1×1 prior convolution. Its posterior path processes the matched analysis feature and the current synthesis state separately, concatenates them, applies a 1×1 merge convolution and another ConvNeXt-AdaLN block, and finally predicts the latent tensor with a 3×3 convolution. After quantization and entropy coding, the decoded latent is projected by a 1×1 convolution and added back to the synthesis state, followed by a second ConvNeXt-AdaLN block. During bitstream compression, the codec stops after the sixth latent stream is emitted; during decompression, the remaining deterministic synthesis blocks are executed and the output is cropped to the stored feature size.

Cross-view condition encoder. The final C^{3}Splat configuration uses direct aggregation before condition encoding. For each non-anchor target, all available references are first warped and aggregated into one target-aligned 64-channel feature $W_i \in \mathbb{R}^{64\times H\times W}$; this feature is then encoded exactly once by the condition encoder. The condition encoder duplicates the analysis topology in Table 3 but has independent parameters. The deployed model uses content-only condition encoding: the four-channel geometric map and the 35-dimensional pairwise camera descriptor are not allocated as condition-encoder input channels. Geometry instead determines the forward splat and the reference scores before condition encoding. The condition encoder produces keys at strides 8, 16, 32, and 64 with widths 384, 512, 512, and 512. The stride-16 key is routed to latent levels 2 and 3, while the stride-8 key is routed to levels 4 and 5. The same rate-control embedding used by the base codec is supplied to every condition-encoder block.

Table 4: Hierarchical latent and prior-refinement dimensions. Stride is measured relative to the codec input feature. The refiner input concatenates the condition with the base mean and raw scale.

Level ℓ	Stride	z_ℓ	G_i^ℓ	Refiner input	Refiner output
0	64	32	512	576	64
1	32	32	512	576	64
2	16	64	512	640	128
3	16	64	512	640	128
4	8	16	384	416	32
5	8	16	384	416	32

Reference scoring and aggregation. For each decoded reference, a codec-side depth head predicts the reference depth from the reconstructed feature and the reference near/far bounds. The reference feature is then projected to the target camera by depth-softmax splatting with temperature $\tau = 1$. The returned auxiliary map has four channels: a binary validity mask, normalized horizontal displacement, normalized vertical displacement, and normalized projected depth. The reliability

network concatenates this four-channel map with the 64-channel warped feature, giving 68 input channels. A 3×3 convolution maps the input to 64 hidden channels, GELU supplies the nonlinearity, and a 1×1 convolution produces one spatial score. This scalar score is shared across feature channels but varies over references and target locations.

The final scoring convolution is initialized to zero, so the initial validity-masked softmax assigns equal weight to all valid references. Invalid references are masked before normalization. If no reference is valid at a target location, the normalized weights and aggregated feature are both exactly zero. The scoring network is shared across all reference–target pairs and supports a variable number of causal references without changing its parameter count. The residual reference adapter present in earlier variants is not used in the final direct-aggregation path; coverage and disagreement statistics are still computed for logging and ablations, but they are not concatenated to the condition encoder input.

Level-specific prior refinement. Each latent block has its own prior-aware refiner. At level ℓ, the condition G_i^ℓ is concatenated with the base mean and raw-scale tensors $(\mu_{0,i}^\ell, \rho_{0,i}^\ell)$, giving the input widths listed in Table 4. A 1×1 convolution maps this tensor to 64 hidden channels, GELU supplies the only nonlinearity, and a second 1×1 convolution outputs $2z_\ell$ channels. These output channels are split evenly into a mean correction and a raw-scale correction before the coder-aligned quantization in Appendix A. The final model sets the refinement-only option: no separate condition-only prior branch is added. The refiner's last convolution is initialized to zero, making every initial refined distribution identical to the pretrained base prior while allowing each level to learn a different correction.

Numerical and training configuration. Training uses mixed precision during rate–distortion joint finetuning, while autocasting is disabled for the condition encoder, prior refiners, and probability-scale construction. The reference-scoring network follows the surrounding training precision. On the deployed bitstream path, geometric splatting is evaluated on the CPU in FP32, and probability parameters are reproduced deterministically before table lookup. The refinement mean correction is clipped to $[-16, 16]$, and the 16-phase mean table described in Appendix A is used at every level. During training, λ is sampled once per scene from the configured range $[16, 1024]$ by sampling uniformly in $\lambda^{1/3}$ space; during bitstream evaluation, the selected λ is stored in each view stream.

During entropy-only adaptation, the trainable modules are the condition encoder, reference-scoring network, and six prior refiners; the base feature codec and reconstruction path remain frozen. The training forward pass still constructs conditions from reconstructed features. To keep this efficient, non-anchor reconstructed features are first obtained on the fixed reconstruction lattice and then reused to build all causal contexts. During joint rate–distortion finetuning, the feature codec, codec-side depth head, pre- and post-codec refiners, Gaussian regressor, and Gaussian decoder are also trainable, while the upstream multi-view matching backbone remains frozen. None of these modules changes the transmitted syntax: no reference graph, attention map, condition feature, or predicted depth is added to the bitstream.

C Causal Encoding and Bitstream Specification

The encoder and decoder derive the view order solely from the transmitted camera extrinsics. The first view is the camera medoid. At every subsequent step, the remaining view with the smallest distance to any selected camera center is chosen; ties are resolved by the original view index. This rule fixes both the target order and the set of available references without signaling either one.

Encoding proceeds as follows. First, the medoid feature is entropy-coded with the original single-view hierarchical prior and immediately decoded locally. For each later target, every previously decoded feature is assigned a codec-side depth, forward-splatted into the target camera, and scored by the shared reliability network. The aggregated feature is encoded once by the condition encoder, and its six level-specific outputs refine the corresponding hierarchical priors. After arithmetic coding, the target stream is also decoded locally and inserted into the reference history. Local decoding is required because future conditions must be constructed from reconstructed rather than original features.

Decoding mirrors these operations. After the camera payload is read, the same causal order is recovered. The first view is decoded unconditionally; later streams are decoded sequentially using conditions reconstructed from the current decoded history. The geometry projection, depth-softmax

splatting, and validity-mask construction used by the real bitstream path run on the CPU in FP32. This avoids nondeterministic GPU scatter accumulation and ensures that both endpoints select identical discrete mean-phase and scale-table indices.

Table 5: Scene bitstream layout. Length delimiters use the existing CodecSplat variable-length packing routine.

Field	Contents
Scene header	Three-byte big-endian length of the camera payload.
Camera payload	Deflate-compressed NumPy arrays for image shape, number of context views, camera intrinsics, camera extrinsics, and near/far bounds.
Feature payload	A length-delimited list of per-view streams in causal order $(\pi_1, \ldots, \pi_N)$.
Per-view header	The 32-bit rate-control value λ and two 16-bit feature dimensions.
Latent payload	A length-delimited list of six coarse-to-fine arithmetic-coded latent streams.

The reported rate includes the scene header, complete camera payload, all packing metadata, and every per-view latent stream. Original RGB images, decoded depth maps, causal order, and reference weights are not included because none is required in the bitstream. Scene export currently uses batch size one, matching the per-scene evaluation protocol.

D Complete Rate–Distortion Results

Tables 6 and 7 report all seven evaluated variable-rate settings. Each entry is the average over the same 29 test scenes, and $1\,\mathrm{kB} = 1000$ bytes. Rates are measured from complete arithmetic bitstreams rather than entropy estimates. The entropy-only models preserve CodecSplat's decoded symbols and therefore have identical reconstruction metrics. For the reported PSNR BD-rates on both datasets, cubic polynomials are fitted to log-rate as a function of PSNR and integrated only over the common PSNR interval.

Table 6: Complete rate–distortion results on DL3DV.

Method	λ	Rate (kB)↓	PSNR↑	SSIM↑	LPIPS↓
CodecSplat	16	16.19	24.17	0.778	0.230
	32	23.56	25.06	0.816	0.182
	64	34.18	25.77	0.843	0.150
	128	49.42	26.34	0.862	0.129
	256	68.74	26.73	0.874	0.116
	512	90.17	26.95	0.880	0.109
	1024	108.95	27.05	0.883	0.106
C^{3}Splat (entropy-only)	16	14.64	24.17	0.778	0.230
	32	21.16	25.06	0.816	0.182
	64	30.34	25.77	0.843	0.150
	128	43.00	26.34	0.862	0.129
	256	58.64	26.73	0.874	0.116
	512	76.17	26.95	0.880	0.109
	1024	91.94	27.05	0.883	0.106
C^{3}Splat (jointly finetuned)	16	16.66	24.71	0.794	0.208
	32	23.96	25.52	0.828	0.167
	64	34.04	26.16	0.851	0.140
	128	47.24	26.66	0.868	0.122
	256	62.63	26.97	0.877	0.112
	512	79.16	27.14	0.883	0.107
	1024	93.81	27.23	0.885	0.105

Table 7: Complete rate–distortion results on RE10K. The jointly finetuned model uses the same step-300 checkpoint reported in Table 1.

Method	λ	Rate (kB)↓	PSNR↑	SSIM↑	LPIPS↓
CodecSplat	16	3.11	24.86	0.808	0.218
	32	3.92	25.52	0.835	0.179
	64	5.11	26.08	0.854	0.150
	128	6.55	26.40	0.864	0.133
	256	8.26	26.66	0.870	0.124
	512	10.14	26.76	0.873	0.120
	1024	11.98	26.82	0.875	0.118
C^3Splat (entropy-only)	16	3.04	24.86	0.808	0.218
	32	3.83	25.52	0.835	0.179
	64	5.00	26.08	0.854	0.150
	128	6.39	26.40	0.864	0.133
	256	8.05	26.66	0.870	0.124
	512	9.84	26.76	0.873	0.120
	1024	11.55	26.82	0.875	0.118
C^3Splat (jointly finetuned)	16	3.45	25.24	0.822	0.196
	32	4.36	25.82	0.844	0.163
	64	5.65	26.30	0.859	0.140
	128	7.17	26.57	0.868	0.128
	256	8.92	26.72	0.872	0.122
	512	10.75	26.82	0.875	0.118
	1024	12.45	26.85	0.876	0.117