FECOSPLAT: FEEDBACK-GUIDED COMPRESSION FOR FEED-FORWARD 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Feed-forward 3D Gaussian Splatting (3DGS) enables efficient novel-view synthesis from sparse multi-view images, yet its representations remain costly to store and transmit. Existing approaches compress either the input images, incurring heavy receiver-side reconstruction, or the reconstructed Gaussian primitives, which are difficult to compress due to their heterogeneous and irregular attributes. We instead compress compact intermediate features, providing a better balance between compression efficiency and receiver-side complexity. Based on this paradigm, we propose **FeCoSplat**, a **Fe**edback-guided **Co**mpression framework for feed-forward 3DGS. FeCoSplat first compresses multi-view features to obtain an intermediate 3DGS, whose rendered views are used as feedback to guide a second-stage compression for further refinement. The resulting bitstreams are decoded into a compact implicit state, from which the final Gaussian primitives are reconstructed with a lightweight predictor. Experiments demonstrate that FeCoSplat achieves favorable rate–distortion performance, particularly at low bitrates, while requiring only 3.45M parameters for receiver-side Gaussian reconstruction.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) represents a scene with a set of anisotropic Gaussian primitives and enables efficient differentiable rendering. Conventional 3DGS obtains these primitives through time-consuming per-scene optimization. To address this limitation, feed-forward 3DGS approaches have been proposed to directly predict Gaussian representations from sparse multi-view images in a single forward pass, substantially reducing reconstruction time while generalizing across scenes (Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025; 2026; Wang et al., 2026c; Ye et al., 2025; 2026; Veicht et al., 2026).

Despite these advances, feed-forward 3DGS approaches still produce a large number of Gaussian primitives, often requiring tens of megabytes (MB) for storage and transmission, making efficient compression essential. In a typical feed-forward 3DGS pipeline, multi-view images are first processed by a computationally intensive multi-view feature extractor to aggregate cross-view information, such as the cost-volume construction in Chen et al. (2024b), followed by several feature transformation stages. The resulting features are then fed into lightweight Gaussian prediction heads to predict Gaussian attributes, while the camera intrinsics and extrinsics are used to determine their corresponding 3D positions. Therefore, an important preliminary question for feed-forward 3DGS compression is: *what should be compressed?* An effective choice should achieve favorable rate–distortion (RD) performance while keeping the receiver lightweight. This is particularly important because receivers are often resource-constrained devices and may execute decoding repeatedly, whereas sender-side computation can be performed on more capable infrastructure and amortized across multiple transmissions or receivers. Several compression paradigms can be considered for this problem, as illustrated in Figure 1:

- **Image Compression** compresses the input multi-view images in the RGB domain using existing image codecs (He et al., 2022), and subsequently reconstructs 3DGS from the decoded images at the receiver. While this paradigm benefits from mature image compression techniques, it has two major drawbacks. First, compression distortions in the decoded images, particularly at low bitrates, can propagate through the downstream 3DGS reconstruction pipeline and degrade the final rendering quality. Second, in addition to image

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.

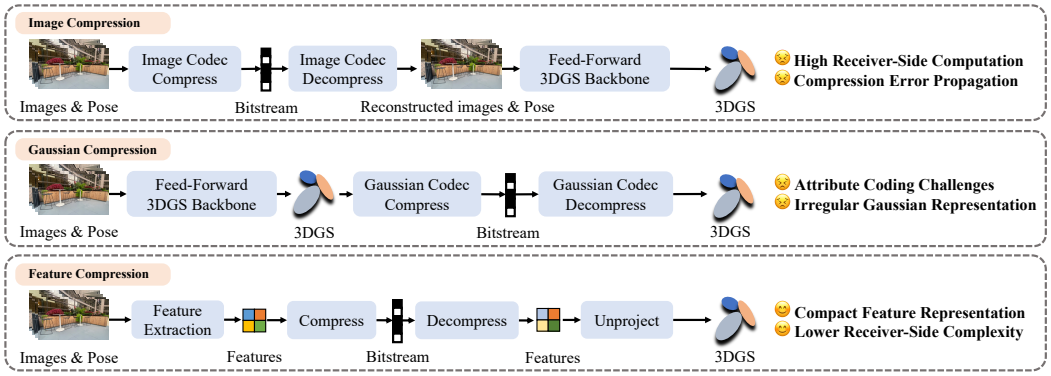


Figure 1: Different compression pipelines for feed-forward 3DGS. Image compression incurs high receiver-side computation and error propagation, while Gaussian compression is challenged by heterogeneous attributes and irregular primitives. Feature compression operates on compact, structured intermediate feature representations, balancing efficiency and receiver-side complexity.

decoding, the receiver must execute the entire feed-forward 3DGS reconstruction pipeline, resulting in substantial receiver-side computational overhead.

- **Gaussian Compression** takes the opposite approach by first reconstructing Gaussian primitives from the input images at the sender and then compressing their attributes for transmission. Although this avoids running the feed-forward 3DGS reconstruction pipeline at the receiver, Gaussian representations typically consist of numerous primitives with heterogeneous attributes, whose distinct distributions and dependencies make them challenging to compress effectively. This challenge becomes even more pronounced for recent feed-forward 3DGS methods (Wang et al., 2026c; Ye et al., 2025; 2026; Veicht et al., 2026) that produce non-pixel-aligned Gaussians with irregular spatial organization. Moreover, since these attributes directly determine the geometry and appearance of the rendered scene, compression distortions can directly degrade the final rendering quality (Chen et al., 2025a).
- **Feature Compression** provides an intermediate solution by compressing features extracted within the feed-forward 3DGS pipeline. After feature decoding, Gaussian primitives can be directly reconstructed using lightweight predictors at the receiver. Compared with explicit Gaussian representations, features are more compact, spatially structured, and more tolerant to compression distortions. Meanwhile, placing the coding interface after computationally intensive multi-view feature extraction allows the major reconstruction cost to remain at the sender, substantially reducing the computational burden at the receiver.

We further investigate how intermediate representations can be more effectively compressed. Despite the promising potential of this paradigm, its compression efficiency remains underexplored. Specifically, existing feature compression methods such as CodecSplat (Yu et al., 2026) compress intermediate representations within a single coding stage, without explicitly considering the resulting 3D reconstruction. However, the decoded features in feed-forward 3DGS are further transformed into an explicit 3D scene representation for rendering. This reconstruction can be rendered back to the RGB observation domain, providing explicit feedback to guide subsequent coding.

To this end, we propose FeCoSplat, a ***feedback-guided compression framework for feed-forward 3DGS***. Our key idea is to make the coding process aware of the current 3D reconstruction, rather than encoding all information in a single stage without observing the reconstruction outcome. Concretely, multi-view features are first compressed to establish an intermediate Gaussian representation, which is rendered back to the input views and compared with the original observations. The resulting reconstruction feedback reveals information that is not sufficiently captured by the first-stage representation and guides a second-stage compression for further refinement. To support efficient information reuse across the two stages, we maintain an implicit Gaussian state in the compact feature domain. The state accumulates reconstruction-relevant information and provides decoded context for subsequent coding, allowing refinement to remain in a structured and compression-friendly representation rather than directly operating on explicit Gaussian attributes. Importantly, the interme-

diate Gaussian reconstruction and rendering are required only at the sender for feedback extraction. At the receiver, the transmitted bitstreams progressively update the implicit state, and only the final state is converted into explicit Gaussian primitives. This design combines reconstruction-aware compression with lightweight receiver-side reconstruction. Our main contributions are summarized as follows:

- We propose FeCoSplat, a compression framework for feed-forward 3DGS that establishes the coding interface at compact and structured intermediate representations. This design avoids both the heavy receiver-side reconstruction required by image compression and the difficulty of efficiently compressing numerous heterogeneous Gaussian attributes.
- We introduce a feedback-guided two-stage coding strategy. The first stage reconstructs an intermediate Gaussian representation from multi-view features, while the second stage compares its rendered results with the original observations and encodes reconstruction-conditioned feedback for refinement. A unified feature codec maintains an implicit Gaussian state represented across the two stages, which aggregates reconstruction-relevant information and serves as a compact representation for Gaussian reconstruction.
- Extensive experiments demonstrate that FeCoSplat achieves improved RD performance over existing feed-forward 3DGS compression approaches while maintaining high-quality novel-view synthesis and enabling lightweight receiver-side reconstruction with only 3.45M parameters.

2 Related Work

2.1 3D Gaussian Splatting and its Feed-Forward Approaches

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) represents a scene as a set of anisotropic Gaussian primitives and optimizes their attributes for high-quality novel-view synthesis. Formally, each Gaussian is parameterized by its 3D position, opacity, scale, rotation, and spherical harmonics coefficients for view-dependent appearance. Despite its rendering efficiency, 3DGS typically requires costly per-scene optimization. Feed-forward 3DGS addresses this limitation by directly predicting Gaussian primitives from input images, enabling generalizable novel-view synthesis. Splatter Image (Szymanowicz et al., 2024) and Flash3D (Szymanowicz et al., 2025) reconstruct 3D Gaussian representations from a single image, while PixelSplat (Charatan et al., 2024) and MVSplat (Chen et al., 2024b) exploit cross-view correspondences for sparse multi-view reconstruction. DepthSplat (Xu et al., 2025) further incorporates monocular depth priors and ReSplat (Xu et al., 2026) iteratively refines 3D Gaussians based on rendering errors. GS-LRM (Zhang et al., 2024) scales feed-forward Gaussian prediction with a transformer-based reconstruction model that directly predicts per-pixel Gaussians from sparse posed images. Recent methods further improve the flexibility, quality, and efficiency of feed-forward reconstruction. NoPoSplat (Ye et al., 2025), YoNoSplat (Ye et al., 2026), and ZipSplat (Veicht et al., 2026) extend reconstruction to more flexible camera settings, while VolSplat (Wang et al., 2026c) and AnySplat (Jiang et al., 2025) improve geometric consistency through voxel-aligned prediction. Other methods explore compact representations and scalable reconstruction through efficient Gaussian allocation or latent scene representations (Wang et al., 2026b; Zhang et al., 2026b; Itkin et al., 2026; Wang et al., 2026a). However, transmission-oriented compression of feed-forward 3DGS remains relatively underexplored.

2.2 Compression of Gaussian Representations

Existing 3DGS compression methods mainly operate directly on Gaussian primitives. Scene-specific approaches reduce the storage of optimized 3DGS through pruning (Lee et al., 2024; Fan et al., 2024; Girish et al., 2024; Wang et al., 2024a; Ali et al., 2024), attribute quantization (Navaneet et al., 2024), and entropy coding (Chen et al., 2024a; 2025b; Wang et al., 2024b). More recent studies explore generalizable codecs that avoid per-scene codec optimization. FCGS (Chen et al., 2025a) introduces a feed-forward codec for generalizable compression of existing Gaussian representations, while D-FCGS (Zhang et al., 2026a) extends feed-forward compression to dynamic Gaussian sequences by exploiting temporal correlations through motion compression and compensation. With the recent development of feed-forward 3DGS reconstruction, several methods further

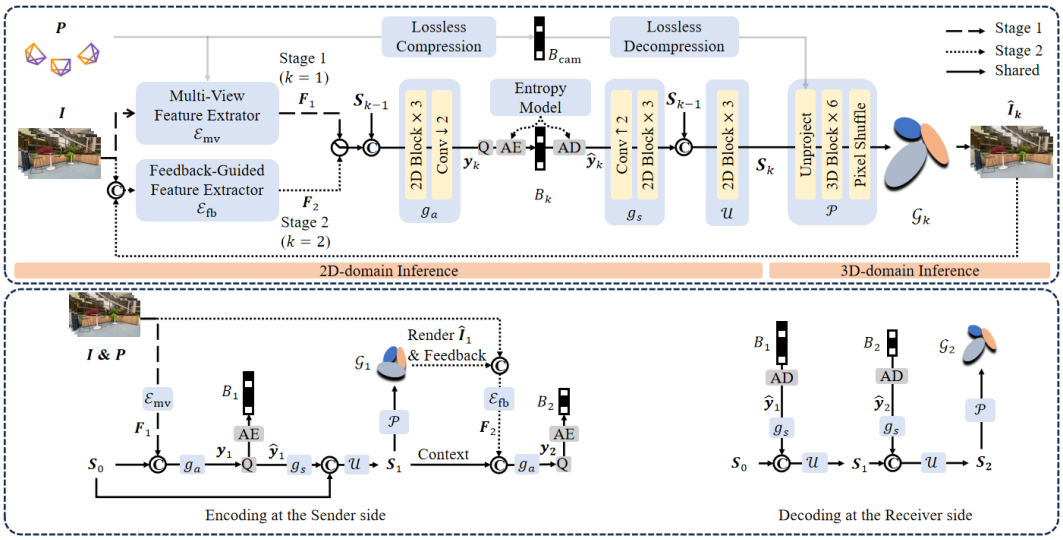


Figure 2: Overview of FeCoSplat. **Top:** FeCoSplat performs feature compression in two stages, encoding multi-view features in Stage 1 ($k = 1$) and feedback-guided features in Stage 2 ($k = 2$). A unified feature codec integrates the decoded information into an implicit Gaussian state across stages, followed by geometry-aware 3D interaction for Gaussian reconstruction. **Bottom Left:** Encoding process, where reconstruction feedback is introduced to capture complementary information, and $\boldsymbol{S}_1$ serves as decoded context to reduce redundancy in the second-stage coding. **Bottom Right:** Decoding process, which requires only lightweight modules and avoids explicitly reconstructing the intermediate 3DGS. The encoding and decoding of camera parameters are omitted for clarity.

target the compression of feed-forward generated Gaussians. TinySplat (Song et al., 2026) transforms Gaussian attributes into 2D representations for conventional video coding, while GenSplat-Codec (Hu et al., 2026) introduces geometry-guided generative decoding to improve reconstruction fidelity at low bitrates. Nevertheless, these methods perform compression after explicit Gaussian reconstruction, where numerous heterogeneous attributes and irregular spatial organization make effective compression challenging.

More recently, CodecSplat (Yu et al., 2026) integrates compression into feed-forward 3DGS reconstruction by encoding intermediate features and directly reconstructing Gaussian primitives from the decoded representation. Building upon this feature compression paradigm, our method further explores reconstruction feedback as an additional cue for organizing the information derived from multi-view observations. Concretely, we decouple feature compression into two stages: the first stage of multi-view scene information for establishing an initial reconstruction and the second stage of reconstruction-conditioned feedback information for supplementing the current representation, enabling the coding process to explicitly consider the information already recovered.

3 Method

3.1 Framework Overview

As a compression-oriented framework, our approach aims at transmitting sparse multi-view observations through compact feature representations and reconstructing 3D Gaussian primitives at the receiver for novel-view rendering. As shown at the top of Figure 2, given N input images $\boldsymbol{I} = \{\boldsymbol{I}^i\}_{i=1}^N$ and their poses $\boldsymbol{P} = \{\boldsymbol{P}^i\}_{i=1}^N$, FeCoSplat encodes them into $B = \{B_1, B_2, B_{\text{cam}}\}$, where B_1 and B_2 are feature bitstreams of two stages compressed with ANS coding (Duda, 2009; Bégaint et al., 2020) and B_{cam} is the camera parameters losslessly compressed using Deflate. To generate B_1 and B_2, FeCoSplat decouples feature compression into two stages. The first stage extracts multi-view features $\boldsymbol{F}_1$ from the input posed images using $\mathcal{E}_{\text{mv}}$ and encodes them into B_1, which conveys the scene information and updates the initial Gaussian state $\boldsymbol{S}_0$ to an intermediate one $\boldsymbol{S}_1$, from which the intermediate Gaussian $\mathcal{G}_1$ is predicted. The rendering of $\mathcal{G}_1$ is then used as reconstruction feed-

back to extract feedback features $\boldsymbol{F}_2$ using $\mathcal{E}_{\text{fb}}$ by comparison with the input images. These features are encoded into the second-stage bitstream B_2 to complement the information in $\boldsymbol{S}_1$, producing the updated state $\boldsymbol{S}_2$, from which the refined Gaussian representation $\mathcal{G}_2$ is predicted. The coding processes at both the sender and receiver are shown at the bottom of Figure 2. The following sections detail the framework.

3.2 Stage I: Multi-View Feature Coding

In the first stage, geometry-aware multi-view features are extracted from the posed input images as:

$$\boldsymbol{F}_1 = \mathcal{E}_{\text{mv}}\left(\boldsymbol{I}, \boldsymbol{P}\right), \tag{1}$$

where $\mathcal{E}_{\text{mv}}$ adopts the feature extraction backbone of DepthSplat (Xu et al., 2025), consisting of a monocular transformer (Yang et al., 2024), a multi-view transformer (Xu et al., 2023), and a cross-view U-Net (Ronneberger et al., 2015; Rombach et al., 2022). The monocular transformer provides strong per-view depth priors, while the multi-view transformer establishes correspondences across input views and provides matching features for cost-volume construction (Chen et al., 2024b). These features, together with the monocular geometric features and cost-volume information, are further fused by the cross-view U-Net to produce geometry-aware multi-view features $\boldsymbol{F}_1 \in \mathbb{R}^{N\times C\times \frac{H}{4}\times \frac{W}{4}}$. $\boldsymbol{F}_1$ is jointly transformed with a learnable initial implicit Gaussian state $\boldsymbol{S}_0 \in \mathbb{R}^{1\times C\times 1\times 1}$ (which is broadcast to match the spatial and view dimensions of $\boldsymbol{F}_1$) into a compact latent representation $\boldsymbol{y}_1$ using g_a: $\boldsymbol{y}_1 = g_a(\text{concat}(\boldsymbol{F}_1, \boldsymbol{S}_0))$, where concat is channel-wise concat. Specifically, to enable quantization-aware training, $\boldsymbol{y}_1$ is perturbed with additive uniform noise to approximate quantization (Ballé et al., 2018), yielding $\hat{\boldsymbol{y}}_1$. To estimate the coding rate of $\hat{\boldsymbol{y}}_1$, we employ a hyper-prior (Ballé et al., 2018) together with a quadtree entropy model (Li et al., 2023). The hyper-latent $\hat{\boldsymbol{z}}_1$ is modeled with a factorized prior to provide side information, while the quadtree entropy model predicts the parameters of a Gaussian distribution over $\hat{\boldsymbol{y}}_1$ conditioned on $\hat{\boldsymbol{z}}_1$. The resulting likelihoods are calculated as:

$$R_1 = R(\hat{\boldsymbol{y}}_1) + R(\hat{\boldsymbol{z}}_1), \tag{2}$$

which is used for end-to-end RD optimization. We refer readers to Ballé et al. (2018) for further details on the entropy modeling framework. Note that *during inference*, the additive uniform noise used to approximate quantization is replaced by Round. Then the quantized latents are entropy coded into the first-stage bitstream B_1. At the receiver, the quantized latent $\hat{\boldsymbol{y}}_1$ is directly available during training, whereas at inference time, it is recovered by entropy decoding the bitstream B_1. The resulting $\hat{\boldsymbol{y}}_1$ is subsequently transformed by the synthesis transform g_s and incorporated into the initial state $\boldsymbol{S}_0$ through the state update module $\mathcal{U}$:

$$\boldsymbol{S}_1 = \mathcal{U}\left(\text{concat}(\boldsymbol{S}_0, g_s(\hat{\boldsymbol{y}}_1))\right). \tag{3}$$

The updated state $\boldsymbol{S}_1$ is then mapped to the intermediate 3D Gaussian representation $\mathcal{G}_1$ through the Gaussian predictor $\mathcal{P}$:

$$\mathcal{G}_1 = \mathcal{P}\left(\boldsymbol{S}_1\right). \tag{4}$$

To maintain low reconstruction complexity, $\mathcal{P}$ operates primarily on a low-resolution 2D feature grid. It first predicts per-view depth maps from $\boldsymbol{S}_1$ and back-projects the features into 3D space. Complementing the preceding 2D feature processing, KNN attention (Zhao et al., 2021) performs geometry-aware aggregation directly in 3D by collecting spatially neighboring features across views. A lightweight prediction head then estimates the Gaussian attributes, which are rearranged by PixelShuffle to produce pixel-aligned 3D Gaussians $\mathcal{G}_1$ at half the input resolution, corresponding to one Gaussian per 2×2 image region. This design combines efficient 2D processing with explicit 3D geometric interaction while avoiding costly high-resolution feature computation.

3.3 Stage II: Feedback-Guided Feature Coding

The first stage produces an intermediate Gaussian representation $\mathcal{G}_1$, which provides an initial reconstruction of the scene. To enable reconstruction-aware refinement, we render $\mathcal{G}_1$ back to the input viewpoints with the corresponding camera parameters $\boldsymbol{P}$:

$$\hat{\boldsymbol{I}}_1 = \text{Render}\left(\mathcal{G}_1, \boldsymbol{P}\right). \tag{5}$$

where $\hat{\boldsymbol{I}}_1$ denotes the rendered observations of $\mathcal{G}_1$ at the input viewpoints, providing explicit feedback on the current reconstruction state of the intermediate 3DGS. Afterwards, we jointly process each original image and its corresponding rendering using a feedback feature extractor $\mathcal{E}_{\text{fb}}$:

$$\boldsymbol{F}_2 = \left\{ \mathcal{E}_{\text{fb}} \left(\text{concat} \left(\boldsymbol{I}^i, \hat{\boldsymbol{I}}_1^i \right) \right) \right\}_{i=1}^{N}. \tag{6}$$

where $\mathcal{E}_{\text{fb}}$ applies PixelUnshuffle to the input and extracts low-resolution feedback features $\boldsymbol{F}_2 \in \mathbb{R}^{N \times C \times \frac{H}{4} \times \frac{W}{4}}$. Conditioned on the input images $\boldsymbol{I}$, $\boldsymbol{F}_2$ captures information that is not sufficiently represented by $\mathcal{G}_1$. The extracted feature representation $\boldsymbol{F}_2$ is jointly transformed with the current implicit Gaussian state $\boldsymbol{S}_1$, which serves as decoded context, into a compact latent representation: $\boldsymbol{y}_2 = g_a(\text{concat}(\boldsymbol{F}_2, \boldsymbol{S}_1))$. Following the same coding procedure as in the first stage, $\boldsymbol{y}_2$ and its corresponding hyper latent $\boldsymbol{z}_2$ follow the same quantization and entropy-modeling scheme to obtain the second-stage rate R_2, and are entropy coded into the second-stage bitstream B_2 during inference. Then the quantized latent $\hat{\boldsymbol{y}}_2$ is decoded and integrated into the current state $\boldsymbol{S}_1$ through the shared state update module $\mathcal{U}$:

$$\boldsymbol{S}_2 = \mathcal{U}\left(\text{concat}(\boldsymbol{S}_1, g_s(\hat{\boldsymbol{y}}_2))\right). \tag{7}$$

The updated state $\boldsymbol{S}_2$ integrates the information decoded from both coding stages. Notably, the second-stage refinement is performed entirely in the compact feature-state domain rather than directly on explicit Gaussian primitives, preserving the structured and compression-friendly representation throughout the coding process. The final state is then mapped to the Gaussian representation through the shared Gaussian predictor $\mathcal{P}$:

$$\mathcal{G}_2 = \mathcal{P}\left(\boldsymbol{S}_2\right). \tag{8}$$

Feedback-Free Decoding. Importantly, reconstruction feedback is required only during encoding, but not during decoding, as shown at the bottom of Figure 2. At the sender, $\boldsymbol{S}_1$ is mapped to the intermediate Gaussian representation $\mathcal{G}_1$, which is rendered to extract the feedback feature $\boldsymbol{F}_2$ for second-stage encoding. At the receiver, however, neither $\mathcal{G}_1$ nor its rendering needs to be reconstructed. Once $\boldsymbol{S}_1$ and the second-stage latent $\hat{\boldsymbol{y}}_2$ are decoded, they can be directly integrated to obtain the final state $\boldsymbol{S}_2$, from which the final Gaussian representation $\mathcal{G}_2$ is reconstructed. Therefore, the feedback loop is used only at the sender, while decoding proceeds directly in the compact feature-state domain. This makes the receiver-side decoding process causal and efficient, as each stage depends only on previously decoded states and the current bitstream, without requiring any backward feedback or intermediate rendering.

3.4 Training Strategy

Loss function. We train the two-stage framework end-to-end using a joint RD loss:

$$\mathcal{L} = \sum_{k=1}^{2} \left(\lambda R_k + \mathcal{L}_{\text{render},k} \right), \tag{9}$$

where R is the rate term, $\mathcal{L}_{\text{render}}$ is the rendering loss, and λ controls the RD trade-off. The rendering loss $\mathcal{L}_{\text{render}}$ follows the setting of Xu et al. (2025); Yu et al. (2026):

$$\mathcal{L}_{\text{render},k} = \sum_{i=1}^{M} (\mathcal{L}_{\text{MSE}}(\boldsymbol{I}^{\text{tgt},i}, \hat{\boldsymbol{I}}_k^{\text{tgt},i}) + \beta\, \mathcal{L}_{\text{LPIPS}}(\boldsymbol{I}^{\text{tgt},i}, \hat{\boldsymbol{I}}_k^{\text{tgt},i})), \tag{10}$$

where $\boldsymbol{I}^{\text{tgt}}$ denotes the target image, $\hat{\boldsymbol{I}}^{\text{tgt}}$ is the rendering from $\mathcal{G}$, and M is the number of target views. We set $\beta = 0.05$. To enable variable-rate coding, we condition each 2D block on λ through Adaptive Layer Normalization (AdaLN) (Duan et al., 2024). Specifically, each 2D block employs a lightweight modulation network ϕ to predict the modulation parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ from λ:

$$(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \phi(\log \lambda), \tag{11}$$

Given an intermediate feature $\boldsymbol{X}$, the normalized feature is modulated as

$$\text{AdaLN}(\boldsymbol{X}, \lambda) = \boldsymbol{\gamma} \odot \text{LN}(\boldsymbol{X}) + \boldsymbol{\beta},$$

where LN denotes Layer Normalization and $\odot$ denotes element-wise multiplication. In this way, by varying λ during training, the feature transformation is adaptively modulated according to the desired RD trade-off, enabling a single codec to operate across multiple rate points.

Table 1: Comparison of sender/receiver parameters, coding latency, and peak GPU memory on RealEstate10K. FeCoSplat achieves a lightweight receiver with only 3.45M parameters and 0.117s latency, while also maintaining low sender-side complexity and the lowest peak GPU memory among the compared learning-based approaches.

Methods	Params (M) ↓		Latency (s) ↓		Peak GPU Mem (GiB) ↓
	Sender	Receiver	Sender	Receiver	
Image Compression					
ELIC→DepthSplat	26.45	62.40	0.077	0.181	1.505
ELIC→ReSplat	26.45	100.69	0.074	0.305	2.119
Gaussian Compression					
DepthSplat→TinySplat	38.34	N/A*	1.285	0.375	1.365
DepthSplat→FCGS	40.87	2.05	2.035	0.821	4.706
ReSplat→TinySplat	76.63	N/A*	1.448	0.497	1.925
ReSplat→FCGS	79.16	2.05	2.319	0.784	5.157
Feature Compression					
CodecSplat	181.73	45.00	0.189	0.138	2.065
FeCoSplat	37.28	3.45	0.168	0.117	1.316

* TinySplat employs the traditional image/video codec HEVC (Sullivan et al., 2012) for encoding and decoding, and thus has no trainable codec parameters.

4 Experiments

4.1 Implementation Details

Datasets. Following prior works on feed-forward 3D Gaussian Splatting, we conduct experiments on RealEstate10K (Zhou et al., 2018) and DL3DV (Ling et al., 2024). For fair comparison, we adopt the same experimental settings as CodecSplat (Yu et al., 2026). Specifically, we use two input views at a resolution of 256×256 for RealEstate10K and eight input views at 256×448 for DL3DV. To further assess the model's cross-dataset generalization ability, we directly evaluate our RealEstate10K-trained model on ACID (Liu et al., 2021), without any additional fine-tuning.

Training Details. We optimize FeCoSplat using AdamW (Loshchilov & Hutter, 2019) with an initial learning rate of 2×10^{-4} and a cosine learning-rate scheduler. The multi-view feature extractor is initialized from the pretrained DepthSplat checkpoint, while the remaining modules are trained from scratch. To support variable-rate compression, the rate-distortion trade-off parameter λ is sampled from a log-uniform distribution during training, with ranges of $[10^{-5}, 10^{-2}]$ and $[10^{-5}, 10^{-3}]$ for RE10K and DL3DV, respectively. All models are trained for 300K steps on four NVIDIA RTX L40S GPUs with a total batch size of 4 using BF16 mixed precision.

Baselines. For compression performance, we compare with three paradigms introduced in section 1: ***Image Compression***, ***Gaussian Compression***, and ***Feature Compression***. For Image Compression, we use ELIC (He et al., 2022) to compress the input views, followed by DepthSplat (Xu et al., 2025) or ReSplat (Xu et al., 2026) for Gaussian reconstruction. For Gaussian Compression, Gaussian primitives reconstructed by DepthSplat or ReSplat are compressed using FCGS (Chen et al., 2025a) or TinySplat (Song et al., 2026). For Feature Compression, we compare with CodecSplat (Yu et al., 2026), which directly compresses intermediate features for Gaussian reconstruction. For a fair comparison, we select model variants for each baseline with parameter counts as close as possible to ours, thereby maintaining comparable model capacity across different compression paradigms.

Evaluation Metrics. We evaluate reconstruction quality using LPIPS (Zhang et al., 2018), PSNR, and SSIM (Wang et al., 2004), and measure the compression rate by the total number of bits required to represent the transmitted data. We further compare the computational complexity of different compression pipelines on RealEstate10K under the two-view 256×256 setting. We report the number of parameters, encoding and decoding latency, and peak GPU memory. All runtime measurements are conducted on the same hardware under identical settings.

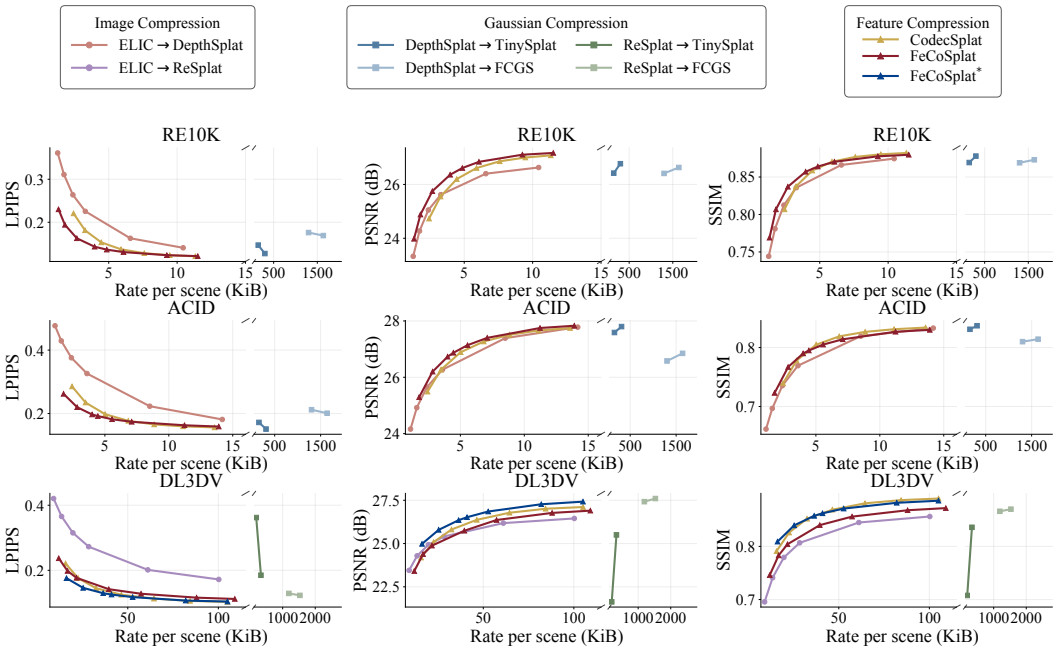


Figure 3: Performance comparison.

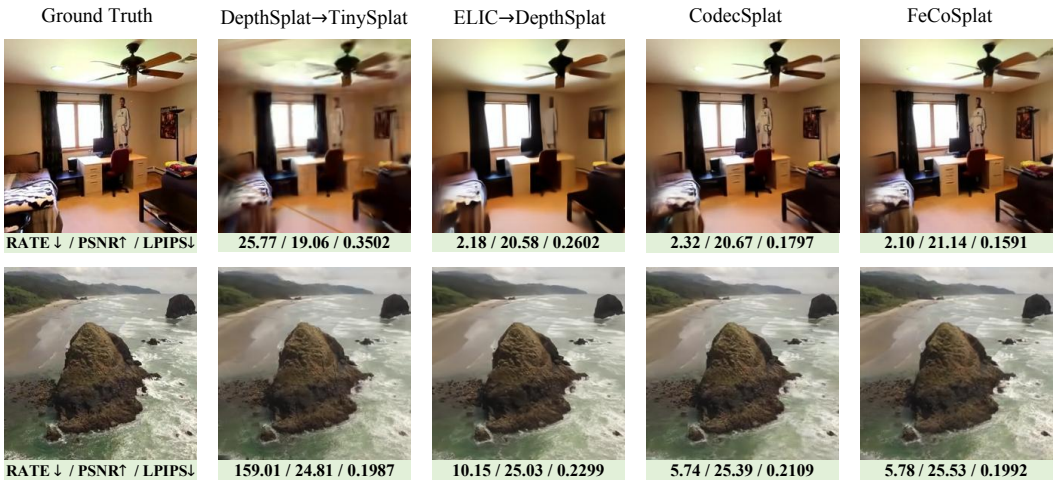


Figure 4: Qualitative comparison on RE10K and ACID. The rate per scene (KiB), PSNR (dB), and LPIPS are reported for each result. FeCoSplat achieves favorable reconstruction quality even at extremely low bitrates compared with competing methods. More results are in Appendix D.

4.2 Parameters and Coding Latency

Table 1 compares the parameter counts and coding latency of different methods. The sender-side parameter count of FeCoSplat is comparable to that of the Image Compression baselines (He et al., 2022), while the receiver contains only 3.45M parameters, substantially fewer than the tens to hundreds of millions of parameters required by competing approaches. Meanwhile, FeCoSplat achieves efficient sender- and receiver-side processing, with latencies of 0.168s and 0.117s, respectively, both lower than those of CodecSplat. The lightweight receiver and efficient coding pipeline make FeCoSplat particularly suitable for edge devices with limited computational resources.

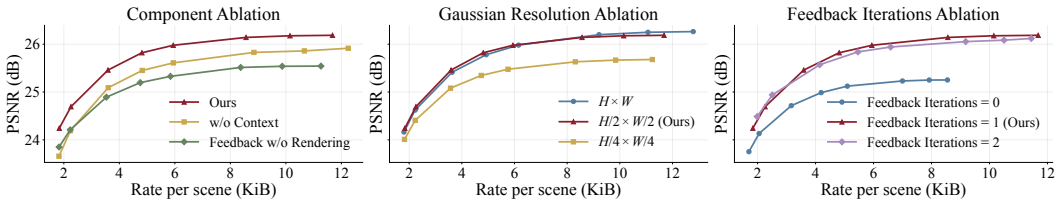


Figure 5: Ablation study on component design, Gaussian resolution and feedback iterations.

4.3 Rate-Distortion Performance

Figure 3 shows the RD performance of FeCoSplat against competing approaches. On RE10K and ACID, FeCoSplat consistently achieves the state-of-the-art RD performance, with particularly pronounced gains in LPIPS, highlighting its advantage in preserving perceptual quality under constrained bit budgets. Image-compression-based approaches degrade rapidly at low bitrates, as the limited coding budget is primarily used to preserve RGB appearance rather than the geometric information required for accurate 3D reconstruction. Gaussian-compression-based approaches exhibit considerably higher compression size, as the numerous heterogeneous Gaussian attributes and their irregular spatial organization are inherently more difficult to compress effectively. Importantly, FeCoSplat uses only about 20% of CodecSplat's sender-side parameters (37M vs. 181M). To evaluate the scalability of our framework, we additionally construct a variant FeCoSplat * that adopts a larger multi-view feature extractor from CodecSplat (Yu et al., 2026), which increases the sender-side parameters to 126.11M, while still keeping the receiver unchanged, as shown in the DL3DV results. This variant surpasses CodecSplat in RD performance, showing that FeCoSplat can benefit from stronger feature extractors without increasing receiver-side complexity. We also compare our standard lightweight model against larger variants of image-compression-based methods. Additional results are in Appendix B.

4.4 Ablation Study

We perform ablation studies on the proposed framework. The results are shown in Figure 5. All variants are trained on RE10K for 100K steps under the same training settings. **Left:** Without the decoded context $\boldsymbol{S}_1$, the model requires a substantially larger rate budget to achieve comparable reconstruction quality, demonstrating the importance of exploiting decoded context across coding stages. Removing the rendered-image feedback primarily lowers the achievable reconstruction fidelity, with the degradation becoming more remarkable at high bitrates. **Mid:** Using an overly low Gaussian resolution significantly limits the reconstruction fidelity ceiling. Full-resolution Gaussians provide only slight quality gains at the highest bitrates, while incurring substantially higher computational cost. This indicates that a moderate Gaussian resolution provides a better trade-off. **Right:** Without iterative refinement, the model lacks reconstruction feedback and therefore reaches a lower fidelity ceiling. Increasing the number of iterations slightly improves reconstruction quality, but the gains quickly diminish, while each additional iteration requires transmitting a new bitstream for the newly extracted information. This suggests that a small number of feedback-guided refinement stages provides a better trade-off between reconstruction fidelity and coding overhead. More discussions and visualizations regarding iteration steps are in Appendix C and Appendix D, respectively.

5 Conclusion

In this work, we present FeCoSplat, a two-stage feedback-guided feature compression framework for feed-forward 3D Gaussian Splatting. FeCoSplat progressively compresses geometry-aware multi-view features and reconstruction-conditioned feedback features, where intermediate Gaussian renderings guide the extraction of complementary information in the second stage. A shared variable-rate feature codec and an implicit Gaussian state enable efficient information reuse and progressive reconstruction without explicitly generating intermediate Gaussians at the decoder. Experimental results demonstrate favorable RD performance, particularly at low bitrates, while maintaining low decoding complexity and high-quality novel-view synthesis.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, including grammar correction, stylistic refinement, and improving the clarity and readability of the manuscript. We did not rely on generative AI tools to generate experimental results, fabricate citations, or make scientific claims without verification. All AI-assisted text was carefully reviewed and revised by the authors to ensure technical accuracy, consistency with the actual methodology and experiments, and appropriate attribution of prior work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: efficient compression of 3d gaussian splats through pruning. In *Proceedings of the British Machine Vision Conference*, 2024.

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.

Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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 feed-forward 3D Gaussian splatting compression. In *International Conference on Learning Representations (ICLR)*, 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*, 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)*, pp. 370–386, 2024b.

Zhihao Duan, Ming Lu, Jack Ma, Yuning Huang, Zhan Ma, and Fengqing Zhu. Qarv: Quantization-aware resnet vae for lossy image compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):436–450, 2024. doi: 10.1109/TPAMI.2023.3322904.

Jarek Duda. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*, 2009.

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

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In *European Conference on Computer Vision*, pp. 54–71. Springer, 2024.

Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5708–5717. IEEE, 2022.

Qiang Hu, Zhenlong Wu, Lei Huang, Zihan Zheng, Xiaoyun Zhang, and Wenjun Zhang. Gensplat-codec: Feed-forward gaussian splatting compression via one-step diffusion. *IEEE Transactions on Multimedia*, 2026.

Roni Itkin, Noam Issachar, Yehonatan Keypur, Xingyu Chen, Anpei Chen, and Sagie Benaim. Globalsplat: Efficient feed-forward 3d gaussian splatting via global scene tokens. In *European Conference on Computer Vision*, 2026.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

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

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

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22616–22626. IEEE, 2023.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22160–22169, 2024.

Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the ieee/cvf international conference on computer vision*, pp. 14458–14467, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, pp. 330–349. Springer, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

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*, 2026.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Stanislaw Szymanowicz, Chrisitian 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.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, Joao F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. In *2025 International Conference on 3D Vision (3DV)*, pp. 670–681. IEEE, 2025.

Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. Hisplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction. In *International Conference on Learning Representations*, volume 2025, pp. 48436–48449, 2025.

Alexander Veicht, Sunghwan Hong, Dániel Baráth, and Marc Pollefeys. Zipsplat: Fewer gaussians, better splats. *arXiv preprint arXiv:2606.05102*, 2026.

Can Wang, Lei Liu, Wei Jiang, and Dong Xu. Z-order transformer for feed-forward gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7362–7371, June 2026a.

Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3d gaussian representation. In *European Conference on Computer Vision*, pp. 76–92. Springer, 2024a.

Weijie Wang, Donny Y Chen, Zeyu Zhang, Duochao Shi, Akide Liu, and Bohan Zhuang. Zpressor: Bottleneck-aware compression for scalable feed-forward 3dgs. *Advances in Neural Information Processing Systems*, 38:113407–113436, 2026b.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Jia-Wang Bian, Zheng Zhu, et al. Volsplat: Rethinking feed-forward 3d gaussian splatting with voxel-aligned prediction. In *European Conference on Computer Vision*, pp. 1–19. Springer, 2026c.

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

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

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 Computer Vision and Pattern Recognition Conference*, pp. 16453–16463, 2025.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Resplat: Learning recurrent gaussian splatting. In *European Conference on Computer Vision*, pp. 364–382. Springer, 2026.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 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*, volume 2025, pp. 54009–54033, 2025.

Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. In *International Conference on Learning Representations*, volume 2026, pp. 39852–39871, 2026.

Pengpeng Yu, Runqing Jiang, Qi Zhang, Dingquan Li, Jing Wang, and Yulan Guo. Codecsplat: Ultra-compact latent coding for feed-forward 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 2026.

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*, pp. 1–19. Springer, 2024.

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

Wenkang Zhang, Yan Zhao, Qiang Wang, Zhixin Xu, Li Song, and Zhengxue Cheng. D-fcgs: Feed-forward compression of dynamic gaussian splatting for free-viewpoint videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 16361–16369, 2026a.

Zicheng Zhang, Xiangting Meng, Ke Wu, and Wenchao Ding. Sparsesplat: Towards applicable feed-forward 3d gaussian splatting with pixel-unaligned prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5049–5058, 2026b.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16259–16268, 2021.

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.

Appendix

A More Implementation Details

A.1 Architectural Details of Blocks and Feature Extractors

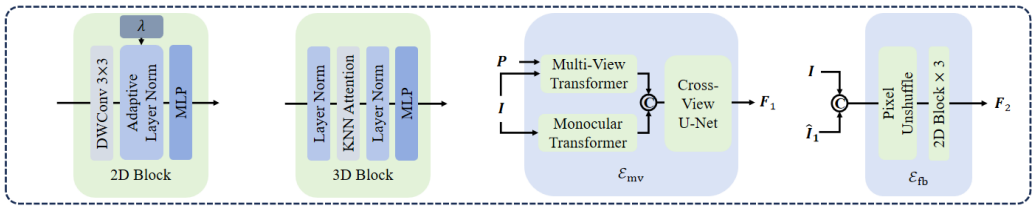


Figure A1: Detailed architectures of the building blocks and feature extractors used in FeCoSplat. The figure illustrates the designs of the 2D and 3D blocks, together with the multi-view feature extractor $\mathcal{E}_{\text{mv}}$ and the feedback-guided feature extractor $\mathcal{E}_{\text{fb}}$.

Figure A1 provides the architectural details of the main building blocks and feature extractors used in FeCoSplat. Specifically, the 2D blocks operate on per-view feature maps for intra-view feature transformation, while the 3D blocks aggregate information across views based on the corresponding 3D representations. We further detail the multi-view feature extractor $\mathcal{E}_{\text{mv}}$, which extracts geometry-aware features from the input observations, and the feedback-guided feature extractor $\mathcal{E}_{\text{fb}}$, which jointly processes the original observations and reconstruction feedback to capture complementary information that is not sufficiently represented by the current reconstruction.

A.2 Details of Multi-View Feature Extraction

Multi-View Branch. Following DepthSplat (Xu et al., 2025), the multi-view branch first extracts per-view features using a lightweight ResNet-style backbone. The resulting features are subsequently processed by a multi-view Transformer composed of alternating self- and cross-view attention layers, allowing each reference view to aggregate correspondence information from the other input views. For more than two input views, cross-view interaction is restricted to nearby source views following the original DepthSplat design to maintain computational efficiency. We resize the resulting multi-view features to a common spatial resolution of $H/4 \times W/4$, yielding

$$\boldsymbol{F}_{\text{mv}} \in \mathbb{R}^{N \times C \times \frac{H}{4} \times \frac{W}{4}}, \tag{12}$$

where N denotes the number of input views and C is the feature dimension.

Cost-Volume Construction. To explicitly encode geometric correspondence across views, we construct per-view cost volumes using plane-sweep stereo (Chen et al., 2024b). For each reference view, a set of depth candidates is sampled within the corresponding near–far range. Features from the source views are warped to the reference camera at each candidate depth using the known camera parameters, and their correlations with the reference features are computed to form a cost volume. When multiple source views are available, the correlations from the selected neighboring views are aggregated to obtain the matching evidence for each reference view. Different from the original implementation, all feature representations in our framework are ultimately aligned to the $1/4$ spatial resolution before subsequent feature fusion and compression. For RealEstate10K and ACID, the cost volume is directly constructed from the $1/4$-resolution multi-view features. For DL3DV, whose larger spatial resolution and increased number of input views make cost-volume construction substantially more expensive, we first bilinearly downsample the multi-view features to a lower spatial resolution for cost-volume computation and then bilinearly upsample the resulting cost-volume features back to $H/4 \times W/4$. This dataset-specific treatment reduces the computational and memory overhead of multi-view matching while keeping the feature resolution presented to the subsequent modules consistent across datasets.

Monocular Branch. In parallel with the multi-view branch, the monocular depth branch is initialized from DepthAnything V2 (Yang et al., 2024) and used to extract per-view geometric priors. Unlike the multi-view features, these monocular features are obtained independently for each view and provide complementary geometric cues in regions where multi-view matching is ambiguous. The monocular features are resized to the same $H/4 \times W/4$ spatial resolution before feature fusion.

Cross-View U-Net Fusion. Finally, the monocular geometric features, multi-view matching features, and cost-volume information are fused by the cross-view U-Net. In contrast to the original reconstruction pipeline, which progressively upsamples the fused features for depth and Gaussian prediction, we retain the low-resolution U-Net representation at $1/4$ of the input resolution as the output of the multi-view feature extractor. The resulting feature

$$\boldsymbol{F}_1 \in \mathbb{R}^{N \times C \times \frac{H}{4} \times \frac{W}{4}} \tag{13}$$

serves directly as the input to our first-stage feature codec.

A.3 Codec and Baseline Settings

Codec Configuration. For both coding stages, the analysis transform g_a and synthesis transform g_s use a feature dimension of 128 channels throughout the main codec blocks. Given an input feature at $1/4$ of the input spatial resolution, the analysis transform g_a produces a 32-channel latent representation:

$$\boldsymbol{y}_k \in \mathbb{R}^{N \times 32 \times \frac{H}{8} \times \frac{W}{8}}. \tag{14}$$

The entropy model operates on the same 32-channel latent dimensionality and consists of a hyperprior (Ballé et al., 2018) and a quadtree context model (Li et al., 2023). The corresponding hyper-latent $\boldsymbol{z}_k$ is further downsampled to $1/32$ of the input spatial resolution.

Backbone and Baseline Settings. For our method, we adopt DepthSplat-Small as the multi-view feature extraction backbone. For the competing feed-forward 3DGS pipelines, we use DepthSplat-Small and ReSplat-Small, with the model variants selected to keep their parameter counts as close as possible to that of our method for a more comparable evaluation. CodecSplat is configured following its official implementation and is built upon DepthSplat-Base.

B Additional Experimental Results

B.1 Compression Using CodecSplat's Multi-View Feature Extractor

To enable a more controlled comparison with CodecSplat (Yu et al., 2026), we conduct an additional experiment on DL3DV by replacing the multi-view feature extractor of FeCoSplat with the feature extraction backbone in CodecSplat. Specifically, we take the intermediate features produced by CodecSplat and first apply PixelUnshuffle to reduce their spatial resolution from full resolution to $1/4$ resolution. The downsampled features are then processed by three 2D blocks to adapt them to our feature representation, yielding the multi-view features $\boldsymbol{F}_1$ used as the input to the subsequent coding stage. All remaining components, including the feedback-guided feature extractor $\mathcal{E}_{\text{fb}}$, analysis and synthesis transforms g_a and g_s, state update module $\mathcal{U}$, and Gaussian predictor $\mathcal{P}$, retain the original design of FeCoSplat without architectural modification.

For training, we freeze the pretrained feature extraction backbone of CodecSplat and only optimize the remaining components of FeCoSplat. These components are initialized from our standard lightweight model and further fine-tuned for 150K iterations with the CodecSplat features. As shown in Figure 3, this scaled variant surpasses CodecSplat in RD performance. These results further demonstrate the scalability of our framework, as a stronger feature extractor can improve compression performance while the lightweight receiver architecture remains unchanged. Moreover, the ability to directly accommodate features from a substantially different and larger feature extractor indicates that our compression framework is not tightly coupled to a specific multi-view backbone or feature representation, demonstrating its compatibility with different feed-forward 3DGS architectures.

B.2 Comparison with Base Models

In the main experiments, model variants with comparable capacities are used whenever available. However, DepthSplat does not provide a corresponding Small model on DL3DV, while ReSplat does not provide Small models on RealEstate10K and ACID. To provide a more comprehensive comparison with the available official models, we additionally compare FeCoSplat with DepthSplat-Base and ReSplat-Base (over 140 M parameters at the receiver). We report the corresponding RD curves and model parameter counts. The results are in Figure A2 and Table A1. Despite its substantially smaller model capacity, FeCoSplat achieves better performance on several metrics at low bitrates and superior LPIPS performance across the evaluated rate range, demonstrating its effectiveness in preserving perceptual quality under constrained bit budgets. Due to time and computation constraints, we are unable to train FeCoSplat at exactly the same model scale as these Base variants. Nevertheless, the scaling experiments in the main paper show that increasing the capacity of FeCoSplat consistently improves reconstruction fidelity, suggesting that the performance gap can be further narrowed with larger models and highlighting the promising scalability of our framework.

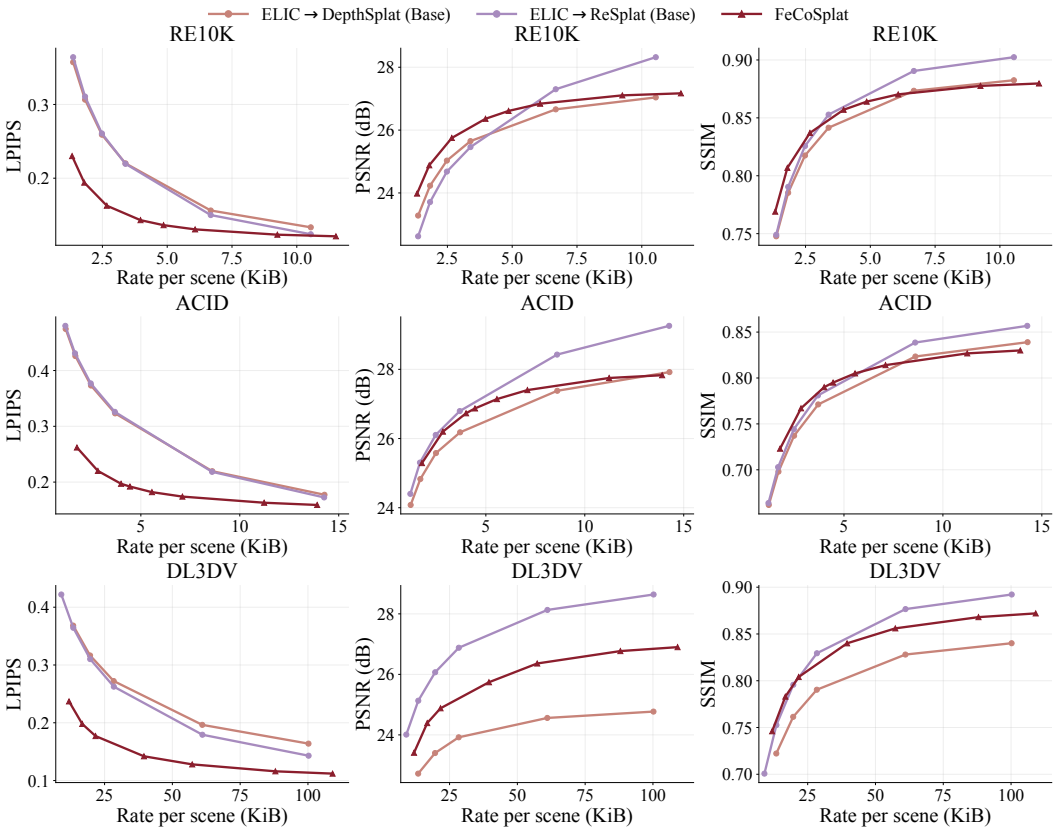


Figure A2: Performance comparison with two base-size methods.

Table A1: Comparison of model parameters at the sender and receiver.

Methods	Sender Params (M) ↓	Receiver Params (M) ↓
ELIC→DepthSplat (Base)	26.45	144.81
ELIC→ReSplat (Base)	26.45	246.20
FeCoSplat	37.28	3.45

C Discussion on Iterative Refinement in Reconstruction and Compression

Iterative refinement plays a different role in compression from that in reconstruction-oriented methods such as ReSplat (Xu et al., 2026) and HiSplat (Tang et al., 2025). In ReSplat, residual updates are directly incorporated into the Gaussian representation without transmission or compression, so additional iterations mainly increase computation and can continuously improve reconstruction quality.

In our framework, each refinement stage is also a coding stage. The complementary information extracted from reconstruction feedback must be quantized, entropy coded, transmitted, and decoded before being integrated into the reconstruction. Therefore, each iteration introduces both lossy compression and additional rate cost, making the gains from repeated refinement smaller from a rate–distortion perspective.

Moreover, our refinement is performed in the compact feature-state domain rather than directly on explicit Gaussian primitives. This preserves the advantages of feature compression while allowing reconstruction feedback to guide subsequent coding. Hence, a small number of refinement stages, such as our two-stage design, provides a better balance between reconstruction fidelity and coding efficiency.

D Additional Visualizations

We provide additional qualitative results to further analyze the reconstruction behavior of FeCoSplat. First, we visualize the intermediate and final reconstructions produced by the two coding stages, together with the bitrate contributed by each stage in Figure A3, to illustrate how the second-stage feedback-guided features complement the intermediate reconstruction. Although the first-stage Gaussian representation is compact, its rendering quality remains relatively limited, resulting in an unfavorable RD trade-off when used alone. The second stage therefore exploits reconstruction feedback to identify and encode complementary information that is insufficiently captured in the first stage, substantially improving reconstruction fidelity with a moderate additional rate. Further, we provide qualitative comparisons with competing methods on DL3DV, demonstrating the reconstruction quality of FeCoSplat at low bitrates in Figure A4.

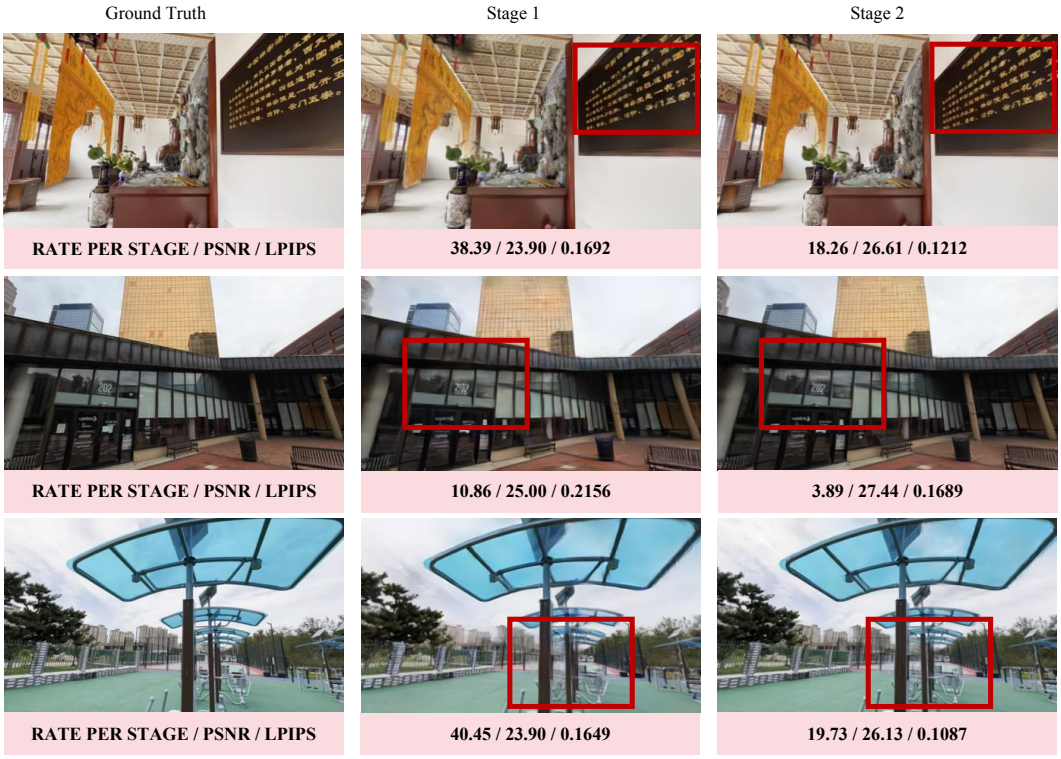


Figure A3: Visualization of the two-stage reconstruction process of FeCoSplat.



Figure A4: Qualitative comparison on DL3DV.