PRIORGUARD-GS: TAMING OVERFITTING IN SPARSE-VIEW RECONSTRUCTION WITH GUIDANCE FROM PRE-TRAINED PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reconstructing scenes from sparse views requires fitting observed images while generalizing to novel views. We present PriorGuard-GS (PGGS), which uses pretrained feed-forward priors to tame overfitting in sparse-view reconstruction. Prior-Guided Latent Optimization (PLO) retains the pretrained prior during adaptation, implicitly regularizing scene updates through learned relationships among Gaussian attributes. Conditional Gaussian Decoding (CGD) generates a variable set of rendering Gaussians, allowing the representation to evolve during optimization. Dynamic Event Awareness (DEA) preserves state across densification, pruning, and opacity resets during optimization. This formulation provides a common interface for multiple feed-forward architectures. We evaluate PGGS across four datasets and five backbones. Across backbones, PGGS improves overall novel-view PSNR by 1.48–4.10 dB over direct Gaussian optimization from the same initialization. With DepthSplat, it outperforms all six optimization-based baselines in all three aggregate metrics. With its fastest backbone, PGGS requires only 30.2 s in the optimization loop on average, compared with 89.1–786.8 s for optimization-based baselines. Qualitative comparisons show reduced ghosting and fewer fragmented structures, supporting the role of continued prior guidance in limiting sparse-view overfitting.

1 INTRODUCTION

Reconstructing a scene from only a few images requires recovering unobserved surfaces, occlusions, and appearance; reducing the input-view error is not enough. 3D Gaussian splatting provides a differentiable representation in which scene-specific optimization can recover detailed geometry and appearance (Kerbl et al., 2023). However, with sparse inputs, lower input-view error does not necessarily imply better novel-view quality, because multiple 3D explanations can fit the same observations.

Feed-forward reconstruction provides a complementary source of information. Models trained across many scenes learn priors that help infer geometry and appearance from limited observations (Xu et al., 2025; Ren et al., 2026; Ye et al., 2026). Even a strong prediction can benefit from scene-specific optimization. Directly optimizing the predicted Gaussians, however, uses the prior only for initialization; subsequent RGB updates bypass the relationships learned across scenes. We instead keep the pretrained prior in the optimization chain so that adaptation remains guided while fitting the current scene.

Figure 1 makes this mismatch explicit. Direct Gaussian optimization reduces input-view error from both MVS and DepthSplat initializations, but increases novel-view error. Starting from the same DepthSplat prediction, our method improves both objectives.

We address this problem with **PriorGuard-GS (PGGS)**. Prior-guided Latent Optimization (PLO) updates an intermediate residual and replays the frozen decoder, keeping anchor attributes coupled through the learned mapping. Conditional Gaussian decoding (CGD) generates rendering Gaussians from anchors, local codes, and a scene-specific decoder. Dynamic Event Awareness (DEA) carries densification, pruning, and opacity-reset state across forward passes. The anchor layout remains tied to the pretrained model while the active rendering set can be updated, and the final attributes can be

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.

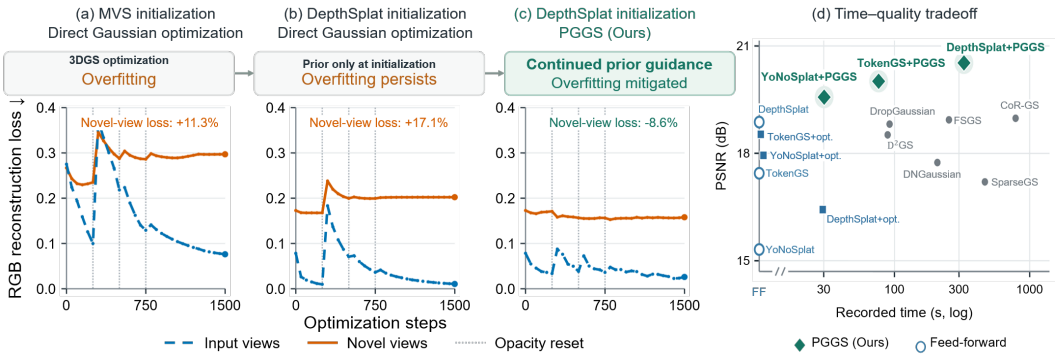


Figure 1: **Input fitting and the time–quality tradeoff.** (a–c) RGB loss on *bonsai* with four inputs and three novel views; PGGS reduces novel-view loss by 8.6%, while direct optimization increases it. Dotted lines mark opacity resets. (d) Mean optimization-loop time versus overall posed PSNR over 24 scenes and three input counts. FF markers indicate reconstruction quality, not inference time.

exported explicitly. Direct optimization allows Gaussian attributes to adapt independently to sparse observations. PLO couples their updates through relationships learned across scenes, providing implicit regularization even in a high-dimensional latent space. CGD adds anchor-conditioned capacity to accommodate scene details.

Our contributions are threefold:

- We introduce Prior-Guided Latent Optimization (PLO), which keeps the pretrained prior throughout optimization. Supervision can therefore adapt the scene through relationships encoded by the prior rather than replacing them with unconstrained Gaussian updates.
- We introduce a conditional Gaussian representation that integrates CGD and DEA to map a fixed-size latent tensor to a variable-size Gaussian set, enabling adaptive capacity while preserving prior-conditioned structure.
- We evaluate PGGS across three posed and two unposed backbones, including a two-view stress test and component ablations that measure both its gains over direct optimization and the limits of adapting a strong feed-forward prediction.

2 Related Work

Sparse-view Gaussian reconstruction. Optimization-based methods fit a Gaussian scene representation to the available views, but sparse observations leave geometry and visibility underconstrained. To alleviate this ambiguity, FSGS (Zhu et al., 2025) and DNGaussian (Li et al., 2024) incorporate monocular depth priors, while SparseGS (Xiong et al., 2025) combines depth guidance with generative regularization at unseen viewpoints. CoR-GS (Zhang et al., 2025) regularizes independently optimized reconstructions through co-pruning and pseudo-view consistency. DropGaussian (Park et al., 2025) and D^{2}GS (Song et al., 2026) mitigate overfitting through random and depth-and-density-guided Gaussian dropout, respectively. Despite these advances, per-scene optimization can still overfit sparse observations, limiting novel-view generalization.

Feed-forward Gaussian reconstruction and refinement. Feed-forward methods learn cross-scene priors to predict Gaussian representations directly from input images. DepthSplat (Xu et al., 2025), YoNoSplat (Ye et al., 2026), NoDrift3R (Sun et al., 2026), and SPFSplatV2 (Huang & Mikolajczyk, 2025) predict pixel-aligned Gaussians, while AnySplat (Jiang et al., 2025) aggregates per-view predictions through voxel fusion and OffTheGrid (Moreau et al., 2026) places primitives at subpixel locations. TokenGS (Ren et al., 2026) predicts Gaussians using learnable tokens and supports scene-latent refinement at test time with frozen model weights, an instance of the broader test-time training paradigm (Sun et al., 2020). In a complementary line of conditional scene representations, Scaffold-GS (Lu et al., 2024) uses anchor-conditioned neural decoding for adaptive Gaussian generation. PGGS couples pretrained latent refinement with a variable-size rendering set, using conditional Gaussian generation and persistent event state to preserve density-control effects across successive decoding passes.

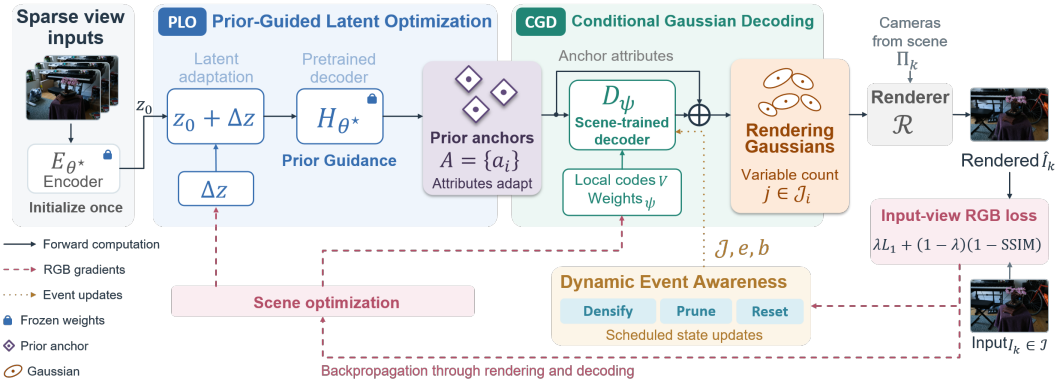


Figure 2: **PriorGuard-GS overview.** PLO updates a latent residual through the frozen decoder; CGD generates Gaussians from the anchors, local codes, and event state; DEA persists active-set and opacity changes. Unposed runs also optimize input extrinsics.

3 METHOD

3.1 PRELIMINARIES AND OVERVIEW

Given input images $\mathcal{I} = \{I_k\}_{k=1}^K$ and cameras Π_k, we represent a scene as Gaussians with center $\boldsymbol{\mu} \in \mathbb{R}^3$, positive scales $\mathbf{s} \in \mathbb{R}^3_+$, unit quaternion $\mathbf{q}$, opacity α, and spherical-harmonic color coefficients $\mathbf{c}$:

$$g = (\boldsymbol{\mu}, \mathbf{s}, \mathbf{q}, \alpha, \mathbf{c}), \qquad \hat{I}_k = \mathcal{R}(\mathcal{G}, \Pi_k). \tag{1}$$

Here $\mathcal{G}$ denotes the rendering Gaussian set, $\mathcal{R}$ is the differentiable renderer, and $\hat{I}_k$ is the rendered image. PGGS starts from a pretrained feed-forward reconstructor that predicts Gaussians from sparse images and available camera information. We split it into frozen encoding and decoding stages, exposing an intermediate representation for scene-specific adaptation:

$$z_0 = E_{\theta^\star}(\mathcal{I}; \mathcal{P}), \qquad A_0 = H_{\theta^\star}(z_0). \tag{2}$$

Here, $\mathcal{P}$ denotes available camera information, and A_0 provides the initial *prior anchors*. Depending on the backbone, z_0 consists of image or patch features or scene tokens. $H_{\theta^\star}$ includes the remaining pretrained decoder and conversion to the common Gaussian attributes. Each adapter retains fixed upstream conditions, such as predicted depth or voxel membership, and the model's native attribute constraints. Appendix A.1 specifies the adaptation endpoints and conversions.

As illustrated in Figure 2, PGGS learns a latent residual Δz, local codes V, and decoder parameters ψ. DEA stores active indices $\mathcal{J}$, event states e, and reset biases b. The pretrained weights and anchor count remain fixed, while anchor attributes and the active rendering set may change. Known-camera runs keep input cameras fixed; unposed runs initialize and optimize input extrinsics while retaining the predicted intrinsics.

3.2 PRIOR-GUIDED LATENT OPTIMIZATION

Prior-guided latent optimization (PLO) keeps the pretrained prior in the loop:

$$z = z_0 + \Delta z, \qquad A(\Delta z) = H_{\theta^\star}(z_0 + \Delta z), \qquad \Delta z^{(0)} = 0. \tag{3}$$

Freezing $\theta^\star$ fixes the latent-to-attribute mapping, not the attributes themselves. The loss reaches the residual through

$$\nabla_{\Delta z} \mathcal{L} = J_H(z_0 + \Delta z)^\top \nabla_A \mathcal{L}. \tag{4}$$

Here $J_H(z) = \partial H_{\theta^\star}(z)/\partial z$ is the Jacobian of the frozen pretrained mapping. Thus every anchor update remains shaped by the pretrained mapping. Let $z_t = z_0 + \Delta z_t$ and $A_t = H_{\theta^\star}(z_t)$. For a gradient-descent step, the induced anchor update is approximately

$$A_{t+1} - A_t \approx -\eta J_t J_t^\top \nabla_A \mathcal{L}_t, \qquad J_t = J_H(z_t). \tag{5}$$

Here η is the gradient-descent step size. This relation illustrates decoder-induced coupling; our implementation uses Adam (Kingma & Ba, 2015) with gradient clipping.

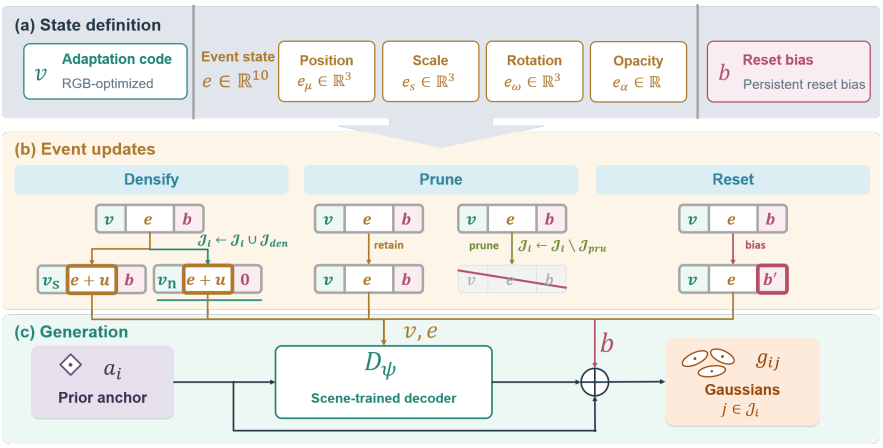


Figure 3: **Persistent state for dynamic Gaussian generation.** Local codes, event states, and reset biases persist through densification, pruning, and opacity resets.

3.3 Conditional Gaussian decoding

A pretrained decoder has a fixed output layout, while scene adaptation may require a different number of rendering Gaussians. Conditional Gaussian decoding (CGD) keeps the predicted Gaussians as conditioning anchors and defines the active set as

$$\mathcal{G} = \{g_{ij} \mid i = 1, \ldots, N_a,\ j \in \mathcal{J}_i\}. \tag{6}$$

Here a_i is the ith of N_a anchors, and $\mathcal{J}_i$ indexes its active rendering members. Anchors do not participate in rendering. Each active Gaussian has a local code v_{ij} and event state e_{ij}. The decoder and local codes are optimized using the current input images. Conditioning on an anchor couples its local corrections, while $H_{\theta^\star}$ retains the cross-scene prior without additional training. Residuals are composed in native attribute coordinates:

$$\delta_{ij} = D_\psi(a_i, v_{ij}, e_{ij}), \qquad g_{ij} = a_i \oplus (\delta_{ij}, b_{ij}), \tag{7}$$

where $\oplus$ composes the residual in the native attribute coordinates, including the corresponding scale, rotation, and opacity parameterizations, and b_{ij} is DEA's reset bias. Each anchor starts with one member. Residuals, local codes, event states, biases, and CGD output are initialized with zero to preserve the feed-forward rendering at initialization.

3.4 Dynamic event awareness

Explicit density control changes both attributes and membership. DEA records these events in the active set and in state consumed by subsequent generations; Figure 3 summarizes the mechanism.

Event state. We allocate ten event-state channels for position, log scale, rotation, and opacity:

$$e = [e_\mu^\top, e_s^\top, e_\omega^\top, e_\alpha]^\top \in \mathbb{R}^{10}, \qquad e_\mu, e_s, e_\omega \in \mathbb{R}^3, \quad e_\alpha \in \mathbb{R}. \tag{8}$$

Densification updates local codes and event states, pruning changes the active set, and opacity resets update a separate persistent bias $b \in \mathbb{R}$.

Densification. We accumulate a decayed score from the world-coordinate position-gradient norms of visible rendering Gaussians, taking the maximum within each anchor. At an event, anchors above the gradient threshold compete for bounded growth and per-anchor capacity. A local-code score selects an inheritance source w; the source and new member receive the same direction and scale hint u_i^{grow}. Their local codes receive independent perturbations to allow the two members to specialize:

$$\begin{aligned} v_{iw}^{\text{new}} &\leftarrow v_{iw}^{\text{old}} + \epsilon_{iw}, \quad v_{i,\text{new}} \leftarrow v_{iw}^{\text{old}} + \epsilon_{i,\text{new}}, \\ \epsilon_{iw}, \epsilon_{i,\text{new}} &\overset{\text{ind}}{\sim} \mathcal{N}(0, \sigma_i^2 \mathbf{I}), \quad \sigma_i = 0.1\ \text{RMS}(v_{iw}^{\text{old}}), \\ e_{iw}^{\text{new}} &\leftarrow e_{iw}^{\text{old}} + u_i^{\text{grow}}, \quad e_{i,\text{new}} \leftarrow e_{iw}^{\text{old}} + u_i^{\text{grow}}. \end{aligned} \tag{9}$$

Here RMS is the root-mean-square magnitude of the inherited code. Existing members retain their optimizer state, whereas new members start with zero Adam moments and opacity bias.

Table 1: **Datasets and backbones.** Angle and Dist. denote mean adjacent optical-axis angle θ and normalized camera-center spacing d/D. Resolutions refer to feed-forward inputs.

(a) Camera geometry

	4 views		6 views		8 views	
Dataset	Angle	Dist.	Angle	Dist.	Angle	Dist.
Mip-NeRF 360	47.31°	0.374	32.76°	0.254	26.52°	0.207
TNT	27.87°	0.231	17.94°	0.139	14.27°	0.109
ACID	1.48°	0.334	0.92°	0.200	0.66°	0.143
DL3DV	43.49°	0.453	26.49°	0.289	27.66°	0.267

(b) Backbone settings

Backbone	Camera	Resolution
DepthSplat	Posed	768×448
TokenGS	Posed	448×256
YoNoSplat	Posed	224×224
AnySplat	Unposed	448×448
OffTheGrid	Unposed	$518 \times$ var.

Pruning and opacity reset. Pruning removes members whose generated opacity remains below threshold. An anchor whose members are all pruned is soft-removed: it has no active rendering path and receives no direct rendering gradient.

At an opacity reset, the generated logit ℓ_{ij} updates the bias without gradients:

$$\begin{aligned} \ell_{ij} &= \operatorname{logit}(\alpha_i) + \delta\ell^{\alpha}_{ij} + b_{ij}, \\ b_{ij} &\leftarrow b_{ij} + \min(0, \operatorname{logit}(\alpha_{\text{cap}}) - \ell_{ij})\,. \end{aligned} \tag{10}$$

Here $\delta\ell^{\alpha}_{ij}$ is the predicted opacity-logit residual, and α_{cap} is the opacity cap applied at reset. This lowers opacities above α_{cap} and persists through regeneration, while continuous optimization may subsequently raise them.

3.5 Supervision

With continued prior guidance, PGGS requires only a simple input-view photometric objective:

$$\mathcal{L}_{\text{RGB}}(\Delta z, V, \psi) = \frac{1}{K}\sum_{k=1}^{K}\Big[\lambda\ \text{mean}\,|\hat{I}_k - I_k| + (1-\lambda)\big(1 - \text{SSIM}(\hat{I}_k, I_k)\big)\Big]\,. \tag{11}$$

The weight $\lambda \in [0, 1]$ balances pixel and structural errors. Each step renders one input view, updates the continuous parameters, and applies scheduled densification, pruning, and reset; unposed runs also update camera extrinsics. After the final step, we materialize the active attributes so that the exported scene renders without either decoder.

4 Experiments

4.1 Experimental setup

Datasets. We evaluate 24 scenes: nine from Mip-NeRF 360 (Barron et al., 2022), eight from Tanks and Temples (Knapitsch et al., 2017), four from ACID (Liu et al., 2021), and three from DL3DV (Ling et al., 2024). Each scene uses nested 4-, 6-, and 8-view input sets and three shared novel views from the same capture trajectory; writing these sets as $\mathcal{C}_4 \subset \mathcal{C}_6 \subset \mathcal{C}_8$ and $\mathcal{T}$, we have $\mathcal{T} \cap \mathcal{C}_8 = \varnothing$. These are custom sparse-view splits. For each dataset, all compared methods use identical input and novel-view selections. Table 1 summarizes camera geometry and backbone settings; scene indices and image preprocessing are detailed in Appendix B. The four-view splits already contain substantial viewpoint changes: the mean adjacent optical-axis angles are $47.31°$ on Mip-NeRF 360 and $43.49°$ on DL3DV, while ACID provides a much smaller-angle sequence (Table 1). Posed optimization and evaluation share image resolutions and camera conventions, while feed-forward inputs follow model-specific requirements.

Baselines. For DepthSplat (Xu et al., 2025), TokenGS (Ren et al., 2026), and YoNoSplat (Ye et al., 2026), we compare feed-forward prediction with direct Gaussian optimization, a dropout-regularized variant adapted from DropGaussian (Park et al., 2025), and PGGS. For each backbone, the three refinement variants share initialization, input images, known cameras, gsplat rendering (Ye et al.,

Table 2: **Posed reconstruction.** Matched refinement variants share initialization and 1,500 steps; external systems use input-only MVS and native schedules. Red/yellow indicate best/second-best results at displayed precision, including ties.

Method	Type	4 views			6 views			8 views		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FSGS	Optimization-based	17.38	0.515	0.425	18.98	0.586	0.363	20.45	0.653	0.302
DNGaussian		16.36	0.481	0.505	17.72	0.549	0.435	19.15	0.605	0.367
CoR-GS		17.56	0.536	0.479	18.88	0.588	0.418	20.50	0.650	0.320
DropGaussian		17.39	0.514	0.435	18.72	0.581	0.373	20.34	0.650	0.306
D^2GS		17.19	0.509	0.437	18.47	0.573	0.383	19.89	0.640	0.316
SparseGS		15.73	0.445	0.442	17.32	0.526	0.381	18.56	0.592	0.317
DepthSplat	Feed-forward	18.58	0.588	0.309	18.90	0.608	0.293	19.14	0.614	0.290
TokenGS		16.89	0.522	0.409	17.55	0.542	0.395	17.89	0.555	0.380
TokenGS + TTT*		16.23	0.500	0.442	16.80	0.522	0.427	17.10	0.532	0.413
YoNoSplat		15.00	0.447	0.509	15.32	0.454	0.493	15.62	0.462	0.481
DepthSplat	Direct opt.	15.63	0.456	0.481	16.41	0.510	0.449	17.24	0.527	0.435
TokenGS		17.55	0.536	0.436	18.63	0.584	0.409	19.42	0.610	0.393
YoNoSplat		16.97	0.508	0.447	18.00	0.545	0.418	18.86	0.580	0.393
DepthSplat	Direct + Dropout	14.65	0.449	0.487	15.38	0.504	0.454	16.21	0.521	0.438
TokenGS		16.81	0.531	0.446	17.74	0.578	0.421	18.41	0.603	0.405
YoNoSplat		16.52	0.504	0.453	17.54	0.541	0.425	18.37	0.577	0.400
DepthSplat	PGGS	**19.56**	**0.602**	**0.288**	**20.62**	**0.646**	**0.263**	**21.40**	**0.668**	**0.246**
TokenGS		18.90	0.577	0.381	20.24	0.623	0.361	20.90	0.646	0.347
YoNoSplat		18.71	0.535	0.405	19.68	0.566	0.376	20.32	0.592	0.354

* Official TokenGS test-time tuning setting.

2025), RGB loss, and 1,500 optimization steps (Appendix A). We also report native TokenGS TTT: 50 scene-latent steps with its original $L_2 + 0.5\,\mathrm{LPIPS}$ objective, separate from the matched 1,500-step RGB comparison. We also compare six optimization-based systems: FSGS (Zhu et al., 2025), DNGaussian (Li et al., 2024), CoR-GS (Zhang et al., 2025), DropGaussian (Park et al., 2025), D^2GS (Song et al., 2026), and SparseGS (Xiong et al., 2025). These use MVS point clouds (Schönberger et al., 2016) reconstructed from input views only and the same known cameras, but retain their native optimization schedules.

Evaluation. We report full-image PSNR, SSIM (Wang et al., 2004), and AlexNet-based LPIPS (Zhang et al., 2018) from floating-point renders. Scores are averaged over target views and scenes; overall scores additionally average the three input counts equally within each scene. All experiments run on a single NVIDIA GeForce RTX 5090 GPU.

4.2 Posed Reconstruction

Reconstruction quality. Relative to feed-forward prediction, PGGS improves overall novel-view PSNR by 1.65, 2.57, and 4.26 dB for DepthSplat, TokenGS, and YoNoSplat, respectively, with gains in SSIM and LPIPS. Averaged over 24 scenes, PGGS outperforms both direct optimization variants in all three metrics at each input count for all three posed backbones (Table 2), with overall PSNR gains of 1.48–4.10 dB over direct optimization and 2.09–5.11 dB over dropout. With DepthSplat, PGGS also outperforms all six optimization-based systems in all three metrics averaged over input counts (Table 2).

Optimization time. As shown in Figure 1, we supplement the quality evaluation with timing experiments that measure the optimization loop. With YoNoSplat and TokenGS, PGGS averages 30.2 and 76.4 s, respectively, compared with 89.1–786.8 s for the six optimization-based baselines. The timing protocol is detailed in Appendix A.4.

Resistance to overfitting. Relative to the DepthSplat initialization, direct optimization raises input-view PSNR by 4.90 dB but lowers novel-view PSNR by 2.45 dB; PGGS improves them by 8.02 and 1.65 dB, respectively. For DepthSplat, TokenGS, and YoNoSplat, respectively, novel-view PSNR falls below feed-forward prediction in 68/72, 38/72, and 25/72 settings after direct optimization,

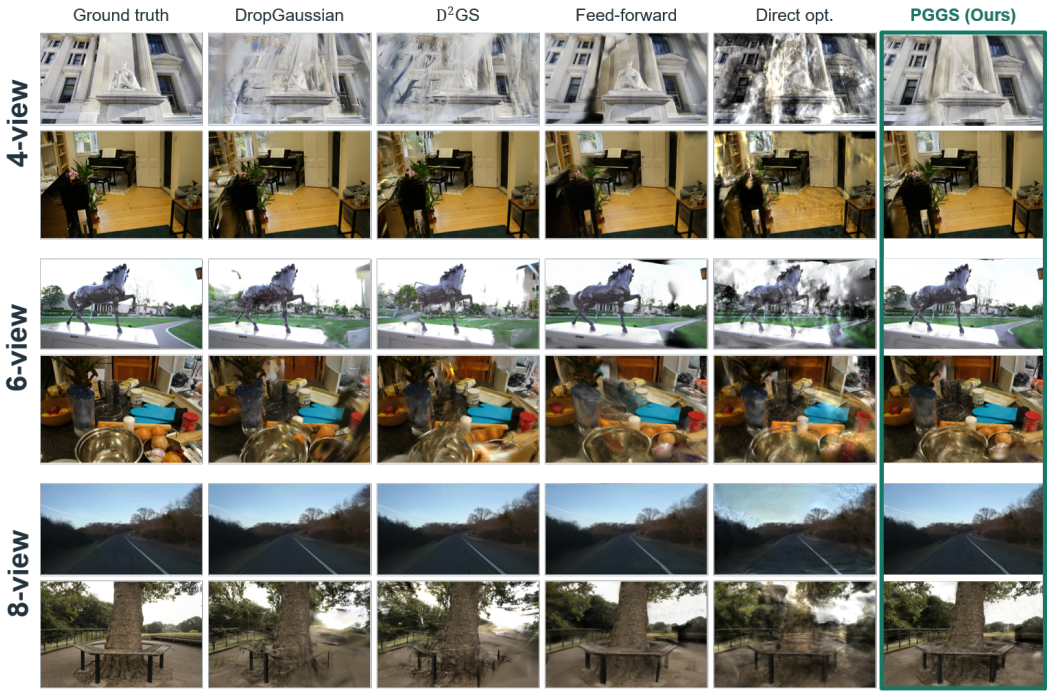


Figure 4: **Qualitative posed reconstruction.** Novel views with 4, 6, and 8 inputs. The last three columns use the same DepthSplat prediction.

Table 3: **Unposed reconstruction.** Dataset means over 4, 6, and 8 inputs with 100-step EPA. EPA optimizes target extrinsics with the scene frozen. Red/yellow indicate best/second; ties are shown. Ranks are within each backbone and dataset.

		Mip-NeRF 360			DL3DV		
Backbone	Variant	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
AnySplat	Feed-forward	15.82	0.324	**0.405**	16.18	0.556	0.314
	Direct	14.14	0.251	0.611	14.10	0.360	0.538
	Direct + Dropout	13.94	0.242	0.619	13.87	0.351	0.537
	PGGS	**17.20**	**0.366**	0.448	**18.19**	**0.587**	**0.303**
OffTheGrid	Feed-forward	16.68	0.345	0.505	17.33	0.535	0.302
	Direct	14.90	0.275	0.628	15.20	0.458	0.426
	Direct + Dropout	14.70	0.272	0.635	14.88	0.462	0.431
	PGGS	**17.08**	**0.368**	**0.480**	**17.93**	**0.584**	**0.275**

versus 3/72, 1/72, and 10/72 after PGGS (Appendix C.6). Figure 4 likewise shows fewer fragmented structures and distortions after PGGS than after direct optimization.

4.3 Unposed Reconstruction

We evaluate AnySplat (Jiang et al., 2025) and OffTheGrid (Moreau et al., 2026) on the nine Mip-NeRF 360 and three DL3DV scenes with predicted cameras at model-native resolutions. The refinement variants jointly optimize scene parameters and input-camera extrinsics for 1,500 steps. For evaluation, we use Evaluation-time Pose Alignment (EPA) to align predicted target cameras to each variant's coordinate frame and refine them for 100 Adam steps against target RGB. All variants use target-assisted camera inference and EPA with the scene frozen; EPA optimizes only target extrinsics (Appendix B.4).

Across all four backbone–dataset pairs, PGGS outperforms both direct optimization variants in every metric (Table 3). It also improves PSNR and SSIM over feed-forward prediction. Figure 5 shows that direct optimization, with or without dropout, introduces ghosting and duplicated contours around the statue, columns, and kitchen objects. In contrast, PGGS preserves coherent scene structure and

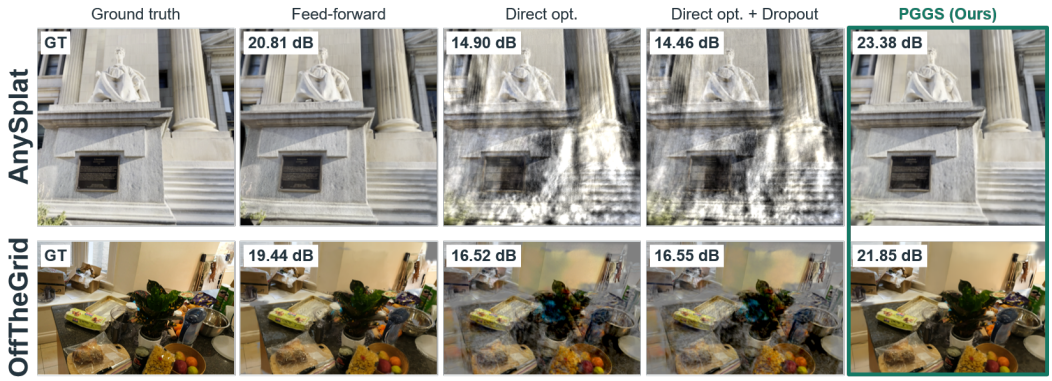


Figure 5: **Qualitative comparison for unposed reconstruction.** Novel-view renderings from eight input views using AnySplat on DL3DV (top) and OffTheGrid on Mip-NeRF 360 (bottom). All results use the same 100-step EPA protocol, which refines only target-camera extrinsics; PSNR is shown in dB.

Table 4: **Two-view stress test.** Optimization-based baselines receive four-view MVS initialization. Red/yellow indicate best/second; ties are shown.

		Optimization-based						TokenGS			AnySplat*		
Dataset	Metric	FSGS	DNG	CoR	DropG	D^2GS	SpGS	FF	Opt.	PGGS	FF	Opt.	PGGS
Mip-NERF	PSNR ↑	14.61	14.07	14.03	14.98	15.15	13.15	15.15	14.39	**16.21**	14.26	12.30	15.64
	SSIM ↑	0.364	0.358	0.392	0.384	0.387	0.299	0.407	0.416	**0.450**	0.325	0.183	0.345
	LPIPS ↓	0.523	0.590	0.730	0.522	0.519	0.559	0.480	0.492	0.448	**0.391**	0.632	0.466
ACID	PSNR ↑	22.40	21.64	23.79	22.48	22.29	15.40	21.49	22.04	**24.97**	23.62	17.66	23.84
	SSIM ↑	0.753	0.747	**0.792**	0.766	0.767	0.600	0.752	0.732	0.790	0.752	0.494	0.744
	LPIPS ↓	0.249	0.281	0.279	0.242	0.242	0.553	0.221	0.298	**0.185**	0.201	0.528	0.209

* AnySplat uses 448×448 crops and target-assisted camera inference with EPA.

produces sharper boundaries, improving PSNR over feed-forward prediction by 2.57 dB for AnySplat and 2.41 dB for OffTheGrid.

4.4 Two-view stress test

We reduce the inputs to two views to test whether prior guidance can limit overfitting with minimal multiview redundancy, using nine Mip-NeRF 360 and four ACID scenes with one novel target per scene. TokenGS uses known cameras, whereas AnySplat predicts them; both initialize their scene representations from the two inputs. The six optimization-based baselines instead receive four-view MVS initialization, excluding the target, before fitting the two input images. AnySplat uses 448×448 crops and target-assisted camera inference with input-camera alignment and photometric EPA.

PGGS improves all three metrics over direct optimization for both backbones on both datasets (Table 4). With TokenGS, it also improves all three metrics over feed-forward prediction, reaching 16.21 and 24.97 dB PSNR on Mip-NeRF 360 and ACID, respectively. With AnySplat, PGGS also exceeds the non-PGGS baselines in PSNR under their respective evaluation protocols (Table 4). Figure 6 illustrates the distinction between fitting and generalization: on *counter*, PGGS achieves higher novel-view PSNR than direct optimization (16.24 vs. 13.14 dB) despite lower PSNR on the displayed input view (31.57 vs. 33.77 dB).

4.5 Component ablations

Table 5 compares six DepthSplat variants on all 24 scenes under the posed protocol (Appendix C.5). With fixed Gaussian counts and no discrete events, FF + PLO outperforms FF + Direct in all three metrics, supporting the benefit of prior guidance without increasing the Gaussian count. FF + Direct exceeds the dynamic direct baseline in Table 2 by 3.15, 3.53, and 3.45 dB PSNR at 4, 6, and 8 views; full PGGS (FF + PLO + CGD + DEA) adds 0.68–0.78 dB over this fixed-count baseline.

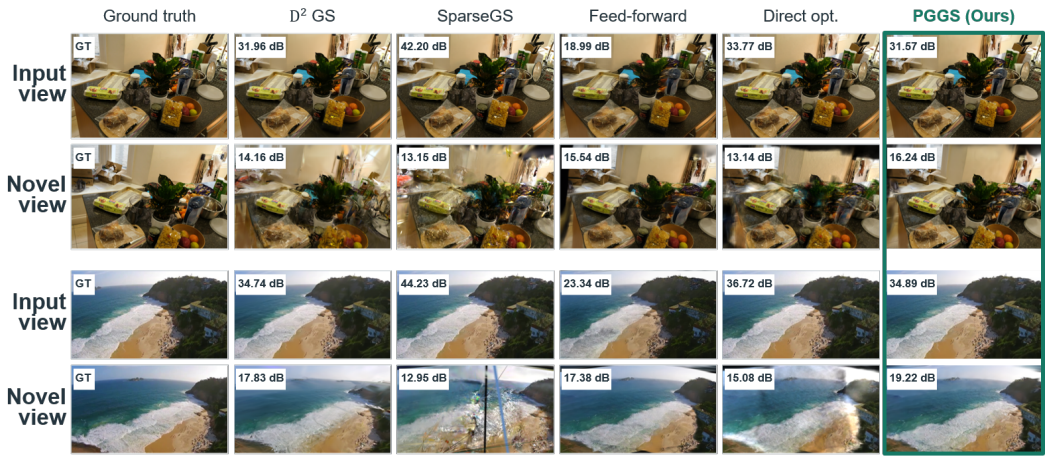


Figure 6: **Two-view fitting and generalization.** Input and novel-view renders; the last three columns use TokenGS. Labels report PSNR in dB.

Table 5: **Component ablations.** DepthSplat backbone, 24 scenes, and all optimized variants use 1,500 optimization steps. Red/yellow indicate best/second; ties are shown.

	4 views			6 views			8 views		
Variant	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FF	18.58	0.588	0.309	18.90	0.608	0.293	19.14	0.614	0.290
FF + Direct†	18.78	0.573	0.294	19.94	0.618	0.269	20.69	0.635	0.255
FF + PLO†	18.83	0.580	**0.269**	20.18	0.638	**0.233**	20.90	0.659	**0.218**
FF + PLO + Explicit	17.23	0.534	0.386	18.40	0.592	0.347	19.50	0.622	0.321
FF + PLO + CGD‡	**19.57**	0.600	0.296	20.58	0.642	0.270	21.37	0.660	0.258
FF + PLO + CGD + DEA	19.56	**0.602**	0.288	**20.62**	**0.646**	0.263	**21.40**	**0.668**	0.246

FF: feed-forward; Explicit: independent attribute residuals. † Fixed count. ‡ Zero event inputs.

This suggests that the extra freedom introduced by density control can hinder generalization without continued prior guidance; the comparison changes growth, pruning, and opacity reset together.

FF + PLO + CGD, with zero event inputs, outperforms FF + PLO + Explicit in all three metrics, with 1.87–2.34 dB PSNR gains. FF + PLO achieves the lowest LPIPS, whereas full PGGS has higher PSNR and SSIM. Compared with the zero-event-input CGD variant, full PGGS improves SSIM and LPIPS, while PSNR changes by at most 0.04 dB (Appendix C.5). Overall, full PGGS provides the most balanced performance across the three metrics, ranking first or second in all nine metric–view-count comparisons at the displayed precision.

5 Conclusion

We presented PriorGuard-GS (PGGS), which combines continued pretrained prior guidance with a scene-adaptive Gaussian representation. PLO refines scenes through frozen pretrained decoders, while CGD and DEA support variable-size Gaussian sets with persistent state across dynamic events. Experiments across four datasets and five backbones demonstrate improved novel-view quality over matched direct Gaussian optimization. Across the three posed backbones, PGGS improves overall PSNR by 1.48–4.10 dB over direct optimization and 1.65–4.26 dB over feed-forward prediction. With YoNoSplat, it averages 30.2 s of optimization-loop time, compared with 89.1–786.8 s for the six optimization-based baselines. Future directions include extending PGGS to a broader range of feed-forward architectures and exploring different latent adaptation endpoints to achieve flexible tradeoffs between reconstruction quality and optimization cost. More broadly, our results indicate that scene-specific optimization need not discard a feed-forward prior after initialization: retaining the pretrained mapping during adaptation provides a practical way to add scene-specific detail while reducing the tendency to overfit sparse observations.

AI USE STATEMENT

Generative AI tools were used to assist with validation of the proposed method. The core research ideas, method design, and experimental design were developed by the authors; AI assistance in the research process was limited to the subsequent validation stage. Additionally, AI tools supported literature discovery and information retrieval, as well as the drafting and refinement of manuscript passages. The authors led the drafting and revision process, using AI suggestions to improve wording and organization and to identify errors and inconsistencies. The authors verified retrieved information and references against the original sources, reviewed all AI-assisted material, and revised it where necessary. The authors assume full responsibility for the final manuscript, including the accuracy and originality of its content and the validity of its scientific claims.

REPRODUCIBILITY STATEMENT

Section 3 describes the representation, optimization objective, and dynamic event mechanisms. Appendix A provides backbone configurations, implementation details, optimization settings, event schedules, and the timing protocol. Appendix B describes scene selection, image preprocessing, evaluation metrics, camera procedures, and baseline settings. Appendix C presents additional results and ablation configurations. The experiments use publicly available datasets and existing pretrained reconstruction models. We will make the source code publicly available upon acceptance.

ETHICS STATEMENT

This work investigates sparse-view 3D reconstruction using publicly available scene datasets and existing pretrained models. The study does not involve recruiting human participants or collecting new personal data. Applications of the method should respect privacy and applicable data licenses.

REFERENCES

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5470–5479, 2022.

Ranran Huang and Krystian Mikolajczyk. SPFSplatV2: Efficient self-supervised pose-free 3D gaussian splatting from sparse views. *arXiv preprint arXiv:2509.17246*, 2025. URL https://arxiv.org/abs/2509.17246v2.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. AnySplat: Feed-forward 3D gaussian splatting from unconstrained views. *ACM Transactions on Graphics*, 44(6):1–16, 2025. doi: 10.1145/3763326.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. doi: 10.1145/3592433.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017. doi: 10.1145/3072959.3073599.

Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. DNGaussian: Optimizing sparse-view 3D gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20775–20785, 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.

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.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20654–20664, 2024.

Arthur Moreau, Richard Shaw, Michal Nazarczuk, Jisu Shin, Thomas Tanay, Zhensong Zhang, Songcen Xu, and Eduardo Pérez-Pellitero. Off The Grid: Detection of primitives for feed-forward 3D gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11756–11766, 2026.

Hyunwoo Park, Gun Ryu, and Wonjun Kim. DropGaussian: Structural regularization for sparse-view gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21600–21609, 2025.

Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. TokenGS: Decoupling 3D gaussian prediction from pixels with learnable tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15365–15375, 2026.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, 2016. URL https://demuc.de/papers/schoenberger2016mvs.pdf.

Meixi Song, Xin Lin, Dizhe Zhang, Haodong Li, Xiangtai Li, Bo Du, and Lu Qi. D^2GS: Depth-and-density guided gaussian splatting for stable and accurate sparse-view reconstruction. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7yvz93kBw9.

Xiangyu Sun, Liu Liu, Seungkwon Yang, Jingbing Han, Seungtae Nam, Zhizhong Su, and Eunbyung Park. NoDrift3R: Raymap-guided coupling for drift-robust unposed feed-forward 3D reconstruction. In *European Conference on Computer Vision*, 2026. URL https://arxiv.org/abs/2607.07168.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 9229–9248, 2020.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero 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.

Haolin Xiong, Sairisheek Muttukuru, Hanyuan Xiao, Rishi Upadhyay, Pradyumna Chari, Yajie Zhao, and Achuta Kadambi. SparseGS: Sparse view synthesis using 3D gaussian splatting. In *International Conference on 3D Vision*, pp. 1032–1041, 2025. doi: 10.1109/3DV66043.2025.00100.

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.

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*, 2026. URL https://openreview.net/forum?id=ImRhA9xmay.

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, 2025.

Jiawei Zhang, Jiahe Li, Xiaohan Yu, Lei Huang, Lin Gu, Jin Zheng, and Xiao Bai. CoR-GS: Sparse-view 3D gaussian splatting via co-regularization. In *Computer Vision – ECCV 2024*, pp. 335–352. Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-73232-4_19.

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.

Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. FSGS: Real-time few-shot view synthesis using gaussian splatting. In *Computer Vision – ECCV 2024*, pp. 145–163. Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-72933-1_9.

A Implementation and Optimization Details

A.1 Backbone configurations and adaptation interfaces

All pretrained weights are frozen. Model-specific preprocessing and conversions remain in the corresponding adapter; the optimization mechanism operates on the common scene attributes.

Table 6 summarizes the adaptation endpoints used by the five backbones. DepthSplat uses the released base model trained first on RealEstate10K and then on RealEstate10K and DL3DV, with two to six training views. TokenGS uses the June 2026 DL3DV latent checkpoints trained with LPIPS: the four-view weights for two and four inputs, and the six-view weights for six and eight. Thus its two-view test uses weights trained for four inputs. YoNoSplat uses its released DL3DV model trained with two to 32 inputs at 224×224 resolution. AnySplat uses its released pretrained weights. OffTheGrid uses pretrained foundation-model and Gaussian-decoder weights. The backbone comparisons vary in pretraining, architecture, image processing, and camera conditions.

Table 6: **Backbone adaptation endpoints.** K is the input count; H, W are spatial dimensions.

Backbone	Adaptation endpoint	Tensor shape
DepthSplat	Dense image features	$[K, 96, 448, 768]$
TokenGS	Scene tokens	$[4096, 1024]$
YoNoSplat	Dual-stream head inputs	$[K, 1024, 2048]$
AnySplat	Gaussian head input	$[K, 128, 448, 448]$
OffTheGrid	Decoder U-Net output	$[K, 32, H, W]$

For a fixed scene and input set, each backbone has a fixed latent endpoint and pretrained anchor layout. The scene-specific variables comprise the latent residual, local codes, conditional decoder, active members, and unposed input-camera residuals. Each adapter preserves the native attribute constraints when mapping its output to the common representation.

Backbone-specific attributes. DepthSplat discards Gaussians originating in image padding using a fixed validity mask. The selected dense-feature endpoint keeps the upstream predicted depth fixed, but latent changes still affect decoded position offsets, scales, rotations, opacity, and color. No additional cross-view voxel fusion is applied.

TokenGS exposes a scene bottleneck of shape $[4096, 1024]$: the first axis indexes the fixed scene-token slots and the second is the channel dimension. PGGS optimizes a residual at this endpoint and replays the frozen decoder and Gaussian-token queries; the official TTT reference optimizes the same scene-latent tensor while keeping the model weights and Gaussian tokens fixed. Four-input experiments use the released four-view checkpoint; six- and eight-input experiments use the six-view checkpoint. The pretrained latent-to-Gaussian path is replayed while the image-conditioned computation preceding the selected endpoint remains fixed. Official test-time tuning is disabled for the shared optimization branches; the separate TokenGS+TTT reference in Table 2 uses the upstream 50-step scene-latent procedure. YoNoSplat runs in known-camera mode, retains its native color order, and receives no additional spatial fusion. Its endpoint concatenates the two native streams, giving $[K, 1024, 2048]$ (views, point slots, concatenated channels).

AnySplat retains its native voxel fusion. Its latent endpoint is the 128-channel feature map entering the final Gaussian-parameter convolution. The source points and voxel membership are fixed, but changes in predicted fusion weights can change the fused anchor positions.

OffTheGrid's selected latent is the 32-channel output of the Gaussian decoder's U-Net. Fixed adaptive patch selection preserves the anchor output layout, while continuous image-plane detections and sampled attributes change through the frozen heads. The upstream depth field is cached, and native color is sampled from the input images at the detected locations. Thus, even with a fixed source depth field and RGB image, moving a detection changes the sampled depth and color.

The native spherical-harmonic orders are 2 for DepthSplat, 0 for TokenGS and YoNoSplat, 4 for AnySplat, and 0 for OffTheGrid. Both unposed backbones use scalar opacity. OffTheGrid retains its native renderer. These model-specific attribute semantics remain part of the adaptation interface.

A.2 Parameterization and optimizer

Residual composition. For an anchor with canonical attributes $(\boldsymbol{\mu}_i, \log \mathbf{s}_i, \mathbf{q}_i, \ell_i^\alpha, \mathbf{c}_i)$, the decoder output is applied as

$$\boldsymbol{\mu}_{ij} = \boldsymbol{\mu}_i + \delta\boldsymbol{\mu}_{ij}, \quad \log \mathbf{s}_{ij} = \log \mathbf{s}_i + \delta\boldsymbol{\ell}_{ij}^s, \quad \mathbf{q}_{ij} = \mathrm{normalize}\big(\mathbf{q}_i \otimes \mathrm{Exp}_q(\delta\boldsymbol{\omega}_{ij})\big),$$
$$\ell_{ij}^\alpha = \ell_i^\alpha + \delta\ell_{ij}^\alpha + b_{ij}, \qquad \mathbf{c}_{ij} = \mathbf{c}_i + \delta\mathbf{c}_{ij}.$$

Here $\ell_i^\alpha = \mathrm{logit}(\alpha_i)$, and canonical quaternions use the $wxyz$ order and Hamilton multiplication. The map $\mathrm{Exp}_q(\boldsymbol{v})$ gives the unit quaternion $(\cos(\|\boldsymbol{v}\|/2), \frac{1}{2}\,\mathrm{sinc}(\|\boldsymbol{v}\|/(2\pi))\boldsymbol{v})$, where $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$ for $x \neq 0$ and $\mathrm{sinc}(0) = 1$. The color term retains the adapter's native SH degree, and b_{ij} is the persistent DEA opacity bias. For a fixed scene and input set, the pretrained anchor count and output slots remain fixed, while the number of active rendering Gaussians may change during optimization.

The conditional decoder concatenates the current anchor attributes, a 16-dimensional local code, and the 10-dimensional event state. After layer normalization, two fully connected hidden layers of width 64 use GELU activations. The final linear layer predicts position, log-scale, rotation-vector, opacity-logit, and native color residuals. Its weights and bias are initialized to zero. The other hidden-layer weights use their standard initialization. Together with zero latent residuals, codes, events, and reset biases, this gives one initial rendering Gaussian per anchor with unchanged initial attributes.

We use Adam (Kingma & Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and no weight decay. The initial learning rates are 10^{-3} for the latent residual, 10^{-3} for the conditional decoder, and 10^{-2} for local codes. All three groups use the multiplier $0.1^{s/1500}$ at optimization step s. The joint gradient norm of the scene parameters is clipped to 1. Each optimization step uses one input image. Inputs are visited in a random order without replacement within each epoch. The posed comparison and component ablations use run seed 0 and a separate input-order seed of 2; growth perturbations and dropout also use seed 0. Each reported setting uses one run, with averages taken over scenes and input counts. The objective has $\lambda = 0.8$ and uses an 11×11 Gaussian SSIM window with $\sigma = 1.5$. The structural term is $1 - \mathrm{SSIM}$. Latent, code, and scale regularizers are disabled in the reported full configuration.

The dropout comparator independently masks each Gaussian's training-time opacity and scales surviving contributions by the inverse survival probability. Its dropout probability is $p(s) = 0.2s/1500$. The mask affects the rendered contribution without changing the stored opacity, and dropout is disabled for evaluation. This adapts the stochastic masking mechanism of DropGaussian to the shared renderer and 1,500-step schedule.

A.3 Event schedules and state updates

Densification occurs every 50 optimization steps from step 100 through 1200. Pruning occurs every 50 optimization steps from step 100 through 1350. Opacity resets occur at steps 250, 500, and 750, after any growth and pruning scheduled at the same step. The direct comparators use the same event steps with their explicit-Gaussian strategy; PGGS uses the conditional representation and rules below.

For the direct strategy, the cumulative projected-gradient threshold is 3.5×10^{-4}. A qualifying Gaussian is cloned when its maximum world-space scale is at most 0.01 times the scene scale and split otherwise; all qualifying Gaussians are considered, without PGGS's per-anchor member cap or event budget. Direct pruning uses opacity below 0.005; size pruning is disabled. Fixed-count direct disables growth, pruning, and opacity reset together.

For PGGS, visible position-gradient norms are reduced by the maximum over members of an anchor and added to a score whose previous value is multiplied by 0.95. An anchor is eligible for growth when its score is at least 3.5×10^{-4} and it has fewer than six active members. Each event adds at most one member per selected anchor, with a budget of at most 15% of the anchor count. Anchors are selected by decreasing score. The inheritance source within an anchor is chosen using the corresponding decayed local-code gradient norm. Growth scores are cleared after the event.

The current growth hint places a value of -0.5 in the position-event component corresponding to the largest anchor scale and adds $-\log(1.6)$ to the three scale-event components. The same hint is

applied to the source and new member. These values are supplied to CGD as event conditions. The current rules leave the rotation and opacity event channels unchanged; opacity reset uses the separate bias. After copying the local code, independent Gaussian perturbations are added to the source and new code, with per-component standard deviation equal to 10% of the pre-perturbation code's root-mean-square magnitude. An exactly zero code receives zero perturbation. The new member starts with zero opacity bias and zero Adam moments; the tensor-level Adam step counter is retained. The source retains its existing optimizer state. If no inheritance source is available, the new code starts at zero with the current event hint.

Pruning uses generated opacity below 0.005, a minimum member age of 100 optimization steps, and one qualifying pruning check. The same selection removes all associated parameters, event state, biases, and Adam moments. Opacity reset uses $\alpha_{\text{cap}} = 0.01$ and the bias update in Equation 10. The bias persists until another event changes it or the member is removed.

A.4 Supplementary timing experiments

We supplement the reconstruction-quality evaluation with timing experiments on the same 72 scene–input-count settings: 24 scenes with four, six, and eight inputs. All reported times use a common optimization-loop scope, from the first optimization step through the final step, including scheduled events. We report the arithmetic mean of one recorded run per setting, giving equal weight to scenes and input counts.

Table 7 lists the recorded times and PSNR coordinates used in Figure 1(d). The plotted PSNR is the equal-weight mean of the displayed four-, six-, and eight-input PSNR values in Table 2. FF markers denote the initial feed-forward predictions at a schematic axis reference.

Table 7: **Time–quality coordinates in Figure 1(d).** Times average 72 settings.

Method	Time (s)	PSNR ↑	Method	Time (s)	PSNR ↑
FSGS	252.851	18.937	DepthSplat + opt.	29.468	16.427
DNGaussian	207.624	17.743	DepthSplat + PGGS	326.477	20.527
CoR-GS	786.810	18.980	TokenGS + opt.	10.268	18.533
DropGaussian	92.000	18.817	TokenGS + PGGS	76.376	20.013
D^2GS	89.053	18.517	YoNoSplat + opt.	10.755	17.943
SparseGS	466.177	17.203	YoNoSplat + PGGS	30.212	19.570

B Datasets and Evaluation Protocols

B.1 Scene subsets and input selection

The Mip-NeRF 360 scenes are *bicycle, bonsai, counter, flowers, garden, kitchen, room, stump*, and *treehill*. The Tanks and Temples scenes are *Family, Francis, Horse, Lighthouse, M60, Panther, Playground*, and *Train*. The four ACID sequences have identifiers `011aadb7a9e434a1`, `0618197a0b64b5b9`, `0c24d66ce8159b79`, and `1e4a0aa7a5bc2f21`. The DL3DV scenes are identified by prefixes `032dee9f`, `0569e83f`, and `06da7966`.

For these custom sparse-view splits, each posed scene has input sets satisfying

$$\mathcal{C}_4 \subset \mathcal{C}_6 \subset \mathcal{C}_8, \qquad \mathcal{T} \cap \mathcal{C}_8 = \varnothing,$$

where $\mathcal{T}$ contains the same three target views for all three input counts. The two-view stress test uses a separate 13-scene split with one target view per scene.

Input selection. We use registered camera poses and shared SfM tracks to select sparse inputs that span different viewpoints while retaining overlap. Video captures follow frame order, orbital captures use local azimuth, and other captures use camera-center paths. Starting from four inputs, we add views from the same trajectory to form the nested six- and eight-view sets.

Novel-view selection and split freezing. We construct a custom development split for the posed experiments. Novel-view candidates are drawn from trajectory gaps using registered camera poses

and shared SfM tracks. We apply scene-specific camera-center separation checks against all eight input views when constructing the candidate sets. After fixing the input views, we use DepthSplat to predict geometry for back-projection and render the resulting geometric proxy at candidate novel viewpoints. We select the eligible candidates with the highest rendered coverage. The selected three targets are excluded from all eight context views and are shared by the nested input counts. The resulting 72 input/target configurations are fixed and shared by all compared methods. To ensure a fair comparison, all methods use the same input and test views for each scene and input-view setting. Each scene is optimized using its input images and evaluated on the three selected novel views.

Figure 7 shows the selected cameras and SfM geometry of three Mip-NeRF 360 scenes in orthographic views. Frusta indicate camera positions and orientations; gray dots mark the remaining registered cameras. Point clouds are subsampled and cropped for display.

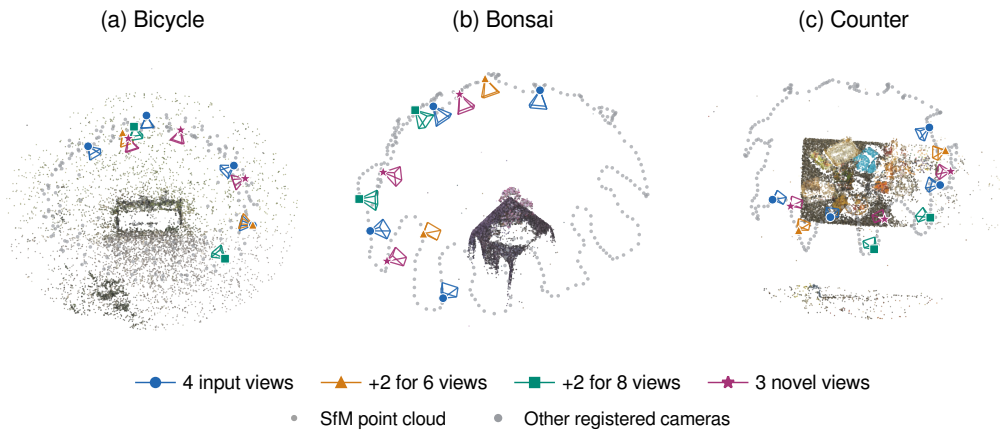


Figure 7: **Sparse-view camera layouts on Mip-NeRF 360.** SfM geometry, nested input sets, and three shared novel views per scene.

For the spacing statistics in Table 1, adjacency follows video order for video captures, local azimuth order for orbital captures, and the shortest open camera-center path for other captures. For a registered world-to-camera rotation R and translation t, the camera center is $-R^\top t$ and the world optical axis is the third row of R. The angle for an adjacent pair is the arccosine of the dot product of their unit optical axes, reported in degrees; the baseline is the Euclidean center distance. The denominator D is the maximum pairwise distance between all registered source-camera centers in a scene and is shared across its three input counts. First averaging adjacent pairs within a scene and then scenes equally prevents a longer trajectory from receiving more weight merely because it contains more camera pairs. In each of the 13 stress-test scenes, the two inputs belong to the four-view MVS support set and the target does not.

B.2 Image preprocessing

For posed supervision and evaluation, Mip-NeRF 360 and Tanks and Temples use a maximum image edge of 800 pixels without upsampling; ACID uses 640×360 and DL3DV uses 480×270. These canvases are separate from the model-specific feed-forward input canvases in Table 1, and intrinsics are scaled with the corresponding image transformations.

The posed backbones resize images with preserved aspect ratios and zero padding to a 768×448 canvas for DepthSplat, 448×256 for TokenGS, and 224×224 for YoNoSplat (width $\times$ height). Camera intrinsics follow the resize and padding transformations. These network inputs are distinct from the higher-resolution image supervision above.

AnySplat uses its official 448×448 center crop. OffTheGrid resizes image width to 518, rounds height to a multiple of 14, center-crops when needed, and pads differing image shapes within a batch. For both unposed backbones, optimization and scoring use the corresponding native canvas and intrinsics.

Table 8: **Views used in the posed evaluation. Each row lists the four-view seed C_4, the two views added for C_6, the two views added for C_8, and the three shared test views T; the input sets are cumulative.**

Scene	C_4	$C_6 \setminus C_4$	$C_8 \setminus C_6$	T test views
M360/bicycle	_DSC8689.JPG, _DSC8695.JPG, _DSC8818.JPG, _DSC8723.JPG	_DSC8747.JPG, _DSC8778.JPG	_DSC8751.JPG, _DSC8727.JPG	_DSC8873.JPG, _DSC8861.JPG, _DSC8742.JPG
M360/bonsai	DSCF5615.JPG, DSCF5582.JPG, DSCF5793.JPG, DSCF5761.JPG	DSCF5601.JPG, DSCF5778.JPG	DSCF5566.JPG, DSCF5850.JPG	DSCF5786.JPG, DSCF5842.JPG, DSCF5591.JPG
M360/counter	DSCF5981.JPG, DSCF6005.JPG, DSCF6065.JPG, DSCF5864.JPG	DSCF5989.JPG, DSCF6080.JPG	DSCF6023.JPG, DSCF6051.JPG	DSCF5984.JPG, DSCF6029.JPG, DSCF6073.JPG
M360/flowers	_DSC9175.JPG, _DSC9179.JPG, _DSC9143.JPG, _DSC9105.JPG	_DSC9104.JPG, _DSC9139.JPG	_DSC9100.JPG, _DSC9147.JPG	_DSC9177.JPG, _DSC9205.JPG, _DSC9145.JPG
M360/garden	DSC08109.JPG, DSC08113.JPG, DSC07963.JPG, DSC08123.JPG	DSC07986.JPG, DSC07965.JPG	DSC07957.JPG, DSC07961.JPG	DSC08111.JPG, DSC08085.JPG, DSC08091.JPG
M360/kitchen	DSCF0820.JPG, DSCF0866.JPG, DSCF0757.JPG, DSCF0748.JPG	DSCF0840.JPG, DSCF0741.JPG	DSCF0878.JPG, DSCF0914.JPG	DSCF0766.JPG, DSCF0882.JPG, DSCF0897.JPG
M360/room	DSCF4920.JPG, DSCF4776.JPG, DSCF4714.JPG, DSCF4706.JPG	DSCF4774.JPG, DSCF4711.JPG	DSCF4762.JPG, DSCF4720.JPG	DSCF4725.JPG, DSCF4741.JPG, DSCF4952.JPG
M360/stump	_DSC9305.JPG, _DSC9232.JPG, _DSC9293.JPG, _DSC9240.JPG	_DSC9324.JPG, _DSC9313.JPG	_DSC9233.JPG, _DSC9217.JPG	_DSC9220.JPG, _DSC9329.JPG, _DSC9337.JPG
M360/treehill	_DSC8946.JPG, _DSC8896.JPG, _DSC8930.JPG, _DSC8929.JPG	_DSC8973.JPG, _DSC8933.JPG	_DSC8899.JPG, _DSC8984.JPG	_DSC8949.JPG, _DSC8926.JPG, _DSC8980.JPG
TNT/Family	00073.jpg 00140.jpg 00144.jpg 00148.jpg	00074.jpg 00149.jpg	00080.jpg 00086.jpg	00007.jpg 00142.jpg 00146.jpg
TNT/Francis	00259.jpg 00275.jpg 00147.jpg 00155.jpg	00154.jpg 00260.jpg	00148.jpg 00274.jpg	00267.jpg 00140.jpg 00152.jpg
TNT/Horse	00025.jpg 00030.jpg 00034.jpg 00137.jpg	00026.jpg 00136.jpg	00035.jpg 00088.jpg	00028.jpg 00032.jpg 00037.jpg
TNT/ Lighthouse	00247.jpg 00091.jpg 00105.jpg 00121.jpg	00120.jpg 00248.jpg	00092.jpg 00106.jpg	00085.jpg 00100.jpg 00266.jpg
TNT/M60	00116.jpg 00294.jpg 00133.jpg 00147.jpg	00117.jpg 00146.jpg	00134.jpg 00295.jpg	00120.jpg 00298.jpg 00006.jpg
TNT/Panther	00046.jpg 00187.jpg 00172.jpg 00160.jpg	00047.jpg 00161.jpg	00173.jpg 00186.jpg	00195.jpg 00012.jpg 00165.jpg
TNT/ Playground	00065.jpg 00234.jpg 00093.jpg 00256.jpg	00066.jpg 00255.jpg	00092.jpg 00235.jpg	00071.jpg 00240.jpg 00248.jpg
TNT/Train	00176.jpg 00181.jpg 00029.jpg 00044.jpg	00043.jpg 00177.jpg	00020.jpg 00030.jpg	00018.jpg 00022.jpg 00038.jpg

Table 9: **Views used in the posed evaluation (continued).**

Scene	C_4	$C_6 \setminus C_4$	$C_8 \setminus C_6$	T test views
ACID/011aadb7	frame_000000.jpg, frame_000006.jpg, frame_000011.jpg, frame_000016.jpg	frame_000009.jpg, frame_000013.jpg	frame_000002.jpg, frame_000004.jpg	frame_000005.jpg, frame_000010.jpg, frame_000012.jpg
ACID/0618197a	frame_000000.jpg, frame_000020.jpg, frame_000039.jpg, frame_000057.jpg	frame_000007.jpg, frame_000027.jpg	frame_000014.jpg, frame_000051.jpg	frame_000017.jpg, frame_000034.jpg, frame_000046.jpg
ACID/0c24d66c	frame_000000.jpg, frame_000042.jpg, frame_000081.jpg, frame_000118.jpg	frame_000056.jpg, frame_000105.jpg	frame_000028.jpg, frame_000093.jpg	frame_000018.jpg, frame_000067.jpg, frame_000086.jpg
ACID/1e4a0aa7	frame_000000.jpg, frame_000067.jpg, frame_000108.jpg, frame_000185.jpg	frame_000051.jpg, frame_000080.jpg	frame_000029.jpg, frame_000157.jpg	frame_000062.jpg, frame_000074.jpg, frame_000114.jpg
DL3/032dee9f	frame_000000.png, frame_000128.png, frame_000241.png, frame_000318.png	frame_000084.png, frame_000275.png	frame_000175.png, frame_000215.png	frame_000110.png, frame_000233.png, frame_000304.png
DL3/0569e83f	frame_000000.png, frame_000095.png, frame_000202.png, frame_000306.png	frame_000042.png, frame_000241.png	frame_000129.png, frame_000162.png	frame_000073.png, frame_000105.png, frame_000264.png
DL3/06da7966	frame_000111.png, frame_000141.png, frame_000183.png, frame_000205.png	frame_000122.png, frame_000191.png	frame_000157.png, frame_000171.png	frame_000134.png, frame_000162.png, frame_000201.png

B.3 Metric implementation and aggregation

Prediction and target RGB values are clamped to $[0, 1]$ for evaluation. PSNR is computed from full-image RGB MSE with data range one; SSIM uses the single-scale Gaussian 11×11 window with $\sigma = 1.5$, zero padding, and population moments. LPIPS uses AlexNet with its standard $[-1, 1]$ input mapping. All three metrics are evaluated on floating-point renders before image quantization; the full-image scores include padding and do not apply an alpha mask. For each setting we first average the target views, then average scenes equally within each dataset. The overall posed means average the three input counts within each scene before averaging the 24 scenes, whereas the displayed per-input-count overall rows average all 24 scenes equally.

B.4 Unposed camera protocol

Table 10 summarizes the five comparison groups. The groups retain their respective input cameras, target-camera predictors, and scoring canvases.

Table 10: **Camera and evaluation protocols.**

Experiment	Input cameras	Target-camera procedure	EPA	Scoring canvas
Posed main	Known, fixed	Known novel-view cameras	None	Dataset evaluation canvas
Unposed main	Predicted and optimized	Target-assisted predicted cameras aligned to each branch	100 steps	Backbone-native canvas
Two-view TokenGS	Known, fixed	Known target camera	None	Known-camera canvas
Two-view AnySplat	Predicted and optimized	Target-assisted prediction followed by input-camera similarity alignment	100 steps	Official 448×448 crop
Two-view baseline methods	Known, fixed	Known target camera	None	Known-camera canvas

The first feed-forward pass uses input images only and defines the initial scene and input-camera coordinate system. In both the main unposed experiments and the two-view AnySplat experiment, each optimized branch jointly adapts all input extrinsics while holding intrinsics and the decoder's initial camera conditions fixed. Camera adaptation uses zero-initialized translation and continuous six-dimensional rotation residuals, with a separate Adam optimizer. Its learning rate is $10^{-5}\, 0.1^{s/1500}$, its weight decay is 10^{-6}, and the updated camera-to-world transform is composed as $T_0 \Delta T$.

After scene optimization, AnySplat processes each target with all input images through its camera-prediction path. Only the corresponding input cameras determine the positive-scale similarity transform to the evaluated branch. The feed-forward branch uses its initial input-camera system, whereas each optimized branch uses its final one. AnySplat retains the predicted target intrinsics. OffTheGrid jointly predicts from all inputs and the three targets. It retains its scalar translation-alignment convention and initial mean input intrinsics, followed by alignment to the final input cameras of the evaluated branch. For every unposed model, target RGB first enters after scene optimization, when the model-specific camera path predicts or aligns evaluation cameras. These predictions supply only the evaluation cameras; each branch retains its optimized Gaussian state.

The EPA stage freezes the Gaussians, input cameras, and intrinsics. Each target camera receives 100 optimization steps, with constant learning rates of 5×10^{-3} for its six-dimensional rotation and translation residuals and zero weight decay. The updated world-to-camera transform is $W_0 \Delta W$. The loss is image MSE plus 0.1 times LPIPS-Alex after the shared resize-and-center-crop processing to 256×256. MSE uses the raw prediction, while the LPIPS input is clamped to $[0, 1]$. Reported metrics are recomputed on the corresponding native canvas. The final pose is evaluated even if an earlier iterate achieved a lower fitting objective. We use the same 100-step EPA protocol across all unposed experiments, including the two-view test.

The two-view AnySplat experiment uses the official 448×448 crop and one target at a time. Its inferred target camera is mapped to each branch using a positive-scale similarity fitted from the two corresponding input cameras. Input-camera orientations determine the rotation, and the two noncoincident centers determine scale and translation. After this mapping, each frozen endpoint follows the same 100-step EPA procedure described above, and metrics are computed using the final target pose.

B.5 Baseline methods

The six baseline methods use MVS point clouds built from the selected input images with the shared known cameras, using COLMAP's multi-view stereo machinery (Schönberger et al., 2016). FSGS, DNGaussian, CoR-GS, DropGaussian, D^{2}GS, and SparseGS use their native schedules of 10,000, 6,000, 30,000, 10,000, 10,000, and 30,000 optimization steps, respectively. A native step can contain multiple parameter updates. For the two-view stress test, these baseline methods receive four-view MVS initialization and then train with two-view RGB supervision.

We use COLMAP 4.0.3 with fixed input cameras, PatchMatch stereo, and stereo fusion. MVS uses the selected context images and their registered cameras, and all six baseline methods share the resulting point clouds.

C Additional Results and Interpretation

C.1 Complete posed results

Tables 11, 12, 13, and 14 report all dataset-by-input-count results underlying Table 2. Table 15 aggregates the 24 scenes separately for each input count. The TTT row uses the official 50-step scene-latent objective. The same novel views are reused across methods and input counts. The reported means are computed from full floating-point values and rounded only for display. The table colors and rank statements use the displayed values: all cells sharing the best displayed value are first, and all cells sharing the next distinct displayed value are second.

C.2 Complete unposed results

Tables 16 and 17 report the unposed results by dataset and input count, using the same 100-step EPA protocol as Table 3.

C.3 Evaluation coverage

The posed experiment covers 24 scenes at three input counts for each of 19 configurations, totaling 1,368 reconstruction endpoints and 4,104 per-view metric evaluations. The unposed comparison

Table 11: **Complete posed results on Mip-NeRF 360.** Means over nine scenes and three novel views per setting. Red/yellow indicate best/second; ties are shown.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FSGS	15.15	0.374	0.552	17.61	0.483	0.460	18.57	0.546	0.409
DNGaussian	13.43	0.319	0.660	15.80	0.426	0.547	16.90	0.485	0.484
CoR-GS	15.39	0.397	0.645	17.36	0.475	0.565	18.31	0.528	0.469
DropGaussian	15.12	0.376	0.562	17.33	0.473	0.473	18.32	0.540	0.414
D^2GS	14.82	0.366	0.569	16.73	0.460	0.488	17.75	0.521	0.429
SparseGS	14.28	0.314	0.530	16.78	0.442	0.430	18.04	0.511	0.347
DepthSplat									
Feed-forward	17.11	0.429	0.419	18.19	0.486	0.368	18.44	0.505	0.358
Direct	15.46	0.384	0.512	16.40	0.459	0.469	17.43	0.497	0.435
Direct + Dropout	15.18	0.378	0.524	16.11	0.455	0.476	17.12	0.495	0.438
PGGS	17.46	**0.436**	**0.402**	**19.25**	**0.522**	**0.348**	**19.79**	**0.549**	**0.322**
TokenGS									
Feed-forward	16.13	0.388	0.554	16.85	0.413	0.534	17.00	0.423	0.524
Direct	15.88	0.403	0.558	17.13	0.472	0.518	17.91	0.498	0.506
Direct + Dropout	15.77	0.404	0.562	16.94	0.470	0.522	17.61	0.496	0.509
PGGS	17.03	0.425	0.536	18.85	0.494	0.505	19.15	0.514	0.495
TokenGS + TTT*	15.24	0.366	0.589	15.91	0.394	0.565	16.16	0.400	0.560
YoNoSplat									
Feed-forward	**17.96**	0.412	0.537	18.55	0.422	0.513	18.94	0.435	0.488
Direct	16.65	0.410	0.505	17.94	0.451	0.473	19.03	0.505	0.438
Direct + Dropout	16.52	0.413	0.506	17.66	0.453	0.474	18.69	0.504	0.441
PGGS	**17.96**	0.417	0.497	18.95	0.451	0.460	19.59	0.485	0.432

* Official TokenGS test-time tuning setting.

Table 12: **Complete posed results on TNT.** Means over eight scenes and three novel views per setting. Red/yellow indicate best/second; ties are shown.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FSGS	14.95	0.542	0.431	15.42	0.578	0.395	17.96	0.685	0.283
DNGaussian	14.60	0.526	0.478	15.16	0.564	0.444	16.95	0.632	0.350
CoR-GS	15.01	0.565	0.439	15.49	0.585	0.431	18.13	0.686	0.282
DropGaussian	15.02	0.541	0.437	15.17	0.575	0.404	17.82	0.683	0.288
D^2GS	14.93	0.533	0.443	15.13	0.571	0.413	17.42	0.675	0.297
SparseGS	13.79	0.466	0.456	14.29	0.506	0.427	15.91	0.603	0.339
DepthSplat									
Feed-forward	16.69	0.644	0.276	15.99	0.620	0.296	16.24	0.612	0.297
Direct	12.57	0.455	0.499	12.52	0.467	0.499	12.70	0.440	0.529
Direct + Dropout	12.10	0.444	0.505	12.07	0.456	0.505	12.20	0.428	0.538
PGGS	**17.72**	**0.662**	**0.262**	17.80	**0.666**	**0.269**	**18.83**	**0.691**	**0.257**
TokenGS									
Feed-forward	15.30	0.559	0.385	16.00	0.571	0.376	16.70	0.595	0.350
Direct	14.47	0.552	0.445	15.27	0.588	0.426	16.27	0.626	0.392
Direct + Dropout	14.35	0.548	0.454	14.99	0.582	0.436	15.91	0.619	0.404
PGGS	16.56	0.629	0.358	17.52	0.654	0.349	18.64	0.688	0.325
TokenGS + TTT*	14.66	0.532	0.412	15.17	0.551	0.402	15.61	0.571	0.373
YoNoSplat									
Feed-forward	14.48	0.504	0.456	14.75	0.512	0.444	14.96	0.514	0.442
Direct	15.73	0.565	0.423	16.70	0.594	0.395	17.55	0.621	0.367
Direct + Dropout	15.51	0.562	0.433	16.36	0.591	0.404	17.22	0.618	0.377
PGGS	17.05	0.585	0.389	**17.90**	0.608	0.368	18.53	0.634	0.336

* Official TokenGS test-time tuning setting.

Table 13: **Complete posed results on ACID.** Means over four scenes and three novel views per setting. Red/yellow indicate best/second; ties are shown.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FSGS	29.65	0.878	0.129	30.87	0.902	0.113	31.79	0.914	0.103
DNGaussian	28.63	0.862	0.173	29.64	0.883	0.162	30.27	0.895	0.147
CoR-GS	**30.27**	0.891	0.122	**31.31**	0.910	0.098	**32.08**	**0.921**	0.089
DropGaussian	29.73	0.880	0.125	31.03	0.905	0.113	31.96	0.915	0.105
D^2GS	29.39	0.878	0.125	31.03	0.905	0.112	31.63	0.913	0.109
SparseGS	24.56	0.774	0.221	25.73	0.808	0.202	26.28	0.831	0.208
DepthSplat									
Feed-forward	28.16	0.880	0.116	28.01	0.887	0.118	27.71	0.883	0.133
Direct	23.82	0.675	0.396	25.39	0.734	0.354	26.13	0.756	0.335
Direct + Dropout	19.82	0.676	0.388	21.06	0.732	0.353	21.97	0.757	0.333
PGGS	30.23	**0.900**	**0.084**	31.29	**0.913**	**0.078**	31.65	0.917	**0.077**
TokenGS									
Feed-forward	23.01	0.811	0.163	23.17	0.815	0.159	23.13	0.814	0.160
Direct	29.91	0.877	0.154	30.88	0.893	0.151	31.19	0.897	0.151
Direct + Dropout	26.01	0.861	0.190	26.64	0.877	0.188	26.80	0.880	0.190
PGGS	30.25	0.892	0.098	31.15	0.908	0.099	31.51	0.910	0.098
TokenGS + TTT*	23.12	0.810	0.162	23.40	0.818	0.151	23.29	0.817	0.152
YoNoSplat									
Feed-forward	11.62	0.513	0.510	11.55	0.514	0.516	11.59	0.522	0.523
Direct	21.49	0.695	0.357	21.99	0.734	0.338	22.43	0.752	0.333
Direct + Dropout	19.99	0.677	0.372	20.69	0.715	0.354	21.08	0.736	0.349
PGGS	25.99	0.804	0.185	27.37	0.839	0.172	28.12	0.855	0.173

* Official TokenGS test-time tuning setting.

Table 14: **Complete posed results on DL3DV.** Means over three scenes and three novel views per setting. Red/yellow indicate best/second; ties are shown.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FSGS	14.22	0.382	0.420	16.74	0.499	0.320	17.59	0.541	0.293
DNGaussian	13.49	0.337	0.552	14.41	0.435	0.439	16.92	0.505	0.352
CoR-GS	13.98	0.402	0.561	15.92	0.506	0.366	17.93	0.561	0.286
DropGaussian	14.07	0.366	0.463	15.96	0.488	0.336	17.64	0.541	0.295
D^2GS	14.10	0.378	0.439	15.82	0.479	0.349	17.21	0.538	0.303
SparseGS	13.49	0.345	0.436	15.79	0.453	0.350	16.93	0.487	0.316
DepthSplat									
Feed-forward	15.21	0.524	0.325	16.66	0.571	0.291	17.56	0.588	0.276
Direct	13.34	0.385	0.451	14.82	0.477	0.379	16.94	0.538	0.320
Direct + Dropout	13.00	0.373	0.457	14.38	0.471	0.382	16.51	0.535	0.316
PGGS	**16.55**	**0.543**	**0.284**	**18.00**	**0.603**	**0.242**	**19.41**	**0.629**	**0.209**
TokenGS									
Feed-forward	15.26	0.435	0.369	16.27	0.486	0.338	16.73	0.498	0.324
Direct	14.29	0.435	0.420	15.77	0.500	0.385	16.62	0.521	0.378
Direct + Dropout	14.25	0.428	0.420	15.61	0.497	0.385	16.32	0.515	0.380
PGGS	15.65	0.478	0.352	17.14	0.546	0.312	18.06	0.574	0.292
TokenGS + TTT*	14.16	0.399	0.456	15.01	0.432	0.447	15.60	0.445	0.427
YoNoSplat									
Feed-forward	11.96	0.315	0.562	12.19	0.317	0.534	12.76	0.323	0.508
Direct	15.21	0.397	0.457	16.34	0.445	0.424	17.09	0.470	0.406
Direct + Dropout	14.55	0.395	0.458	16.13	0.443	0.426	16.88	0.473	0.408
PGGS	15.68	0.394	0.461	16.32	0.438	0.415	16.86	0.448	0.408

* Official TokenGS test-time tuning setting.

Table 15: **Overall posed results at each input count.** Means over 24 scenes. Red/yellow indicate best/second; ties are shown.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FSGS	17.38	0.515	0.425	18.98	0.586	0.363	20.45	0.653	0.302
DNGaussian	16.36	0.481	0.505	17.72	0.549	0.435	19.15	0.605	0.367
CoR-GS	17.56	0.536	0.479	18.88	0.588	0.418	20.50	0.650	0.320
DropGaussian	17.39	0.514	0.435	18.72	0.581	0.373	20.34	0.650	0.306
D^2GS	17.19	0.509	0.437	18.47	0.573	0.383	19.89	0.640	0.316
SparseGS	15.73	0.445	0.442	17.32	0.526	0.381	18.56	0.592	0.317
DepthSplat									
Feed-forward	18.58	0.588	0.309	18.90	0.608	0.293	19.14	0.614	0.290
Direct	15.63	0.456	0.481	16.41	0.510	0.449	17.24	0.527	0.435
Direct + Dropout	14.65	0.449	0.487	15.38	0.504	0.454	16.21	0.521	0.438
PGGS	**19.56**	**0.602**	**0.288**	**20.62**	**0.646**	**0.263**	**21.40**	**0.668**	**0.246**
TokenGS									
Feed-forward	16.89	0.522	0.409	17.55	0.542	0.395	17.89	0.555	0.380
Direct	17.55	0.536	0.436	18.63	0.584	0.409	19.42	0.610	0.393
Direct + Dropout	16.81	0.531	0.446	17.74	0.578	0.421	18.41	0.603	0.405
PGGS	18.90	0.577	0.381	20.24	0.623	0.361	20.90	0.646	0.347
TokenGS + TTT*	16.23	0.500	0.442	16.80	0.522	0.427	17.10	0.532	0.413
YoNoSplat									
Feed-forward	15.00	0.447	0.509	15.32	0.454	0.493	15.62	0.462	0.481
Direct	16.97	0.508	0.447	18.00	0.545	0.418	18.86	0.580	0.393
Direct + Dropout	16.52	0.504	0.453	17.54	0.541	0.425	18.37	0.577	0.400
PGGS	18.71	0.535	0.405	19.68	0.566	0.376	20.32	0.592	0.354

* Official TokenGS test-time tuning setting.

Table 16: **Complete unposed results for AnySplat.** Dataset means after 100-step EPA. Red/yellow indicate best/second; ties are shown. Ranks are within each dataset and input count.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF 360									
Feed-forward	14.98	0.295	**0.436**	16.01	0.330	**0.400**	16.48	0.348	**0.380**
Direct	13.63	0.229	0.630	14.08	0.251	0.613	14.71	0.273	0.590
Direct + Dropout	13.46	0.218	0.643	13.93	0.245	0.615	14.44	0.264	0.601
PGGS	**16.59**	**0.350**	0.484	**17.34**	**0.370**	0.438	**17.66**	**0.377**	0.421
DL3DV									
Feed-forward	15.43	0.505	0.359	16.12	0.565	0.305	16.99	0.598	0.279
Direct	13.53	0.330	0.556	13.75	0.346	0.558	15.01	0.403	0.502
Direct + Dropout	13.27	0.321	0.567	13.61	0.346	0.548	14.71	0.387	0.497
PGGS	**17.00**	**0.543**	**0.343**	**18.21**	**0.593**	**0.299**	**19.35**	**0.625**	**0.266**

Table 17: **Complete unposed results for OffTheGrid.** Dataset means after 100-step EPA. Red/yellow indicate best/second; ties are shown. Ranks are within each dataset and input count.

	4 views			6 views			8 views		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Mip-NeRF 360									
Feed-forward	16.12	0.331	0.531	16.40	0.336	0.524	17.52	0.368	0.459
Direct	14.60	0.253	0.643	14.73	0.267	0.636	15.39	0.305	0.606
Direct + Dropout	14.24	0.251	0.657	14.67	0.274	0.638	15.19	0.292	0.609
PGGS	**16.21**	**0.332**	**0.514**	**16.91**	**0.363**	**0.498**	**18.13**	**0.407**	**0.430**
DL3DV									
Feed-forward	16.58	0.502	0.338	17.46	0.545	0.295	17.97	0.559	0.274
Direct	14.74	0.421	0.444	15.29	0.479	0.425	15.58	0.473	0.409
Direct + Dropout	14.33	0.422	0.452	14.89	0.464	0.432	15.41	0.499	0.410
PGGS	**16.84**	**0.533**	**0.308**	**18.22**	**0.601**	**0.271**	**18.73**	**0.619**	**0.247**

covers two backbones, 12 scenes, three input counts, and four branches, yielding 288 endpoints. The two-view stress test evaluates six baseline methods and three branches of each of TokenGS and AnySplat on 13 scenes, yielding 156 endpoints. The six ablation variants cover 72 settings and 432 reported endpoints, including 144 feed-forward and full-model endpoints reused from the posed experiment. Together, these experiments evaluate 2,100 distinct reconstruction endpoints.

C.4 Adaptation under uncertain cameras and sparse geometry

The unposed experiments improve PSNR and SSIM over feed-forward prediction for all four backbone–dataset pairs. For AnySplat on Mip-NeRF 360, however, LPIPS increases from 0.405 to 0.448. With jointly estimated cameras and geometry, an input-image residual can be explained by either a camera correction or a scene change. Sparse supervision may therefore favor changes that improve pixel agreement without preserving the structure or texture statistics of the initial prediction. EPA adjusts target extrinsics after scene fitting; it cannot repair a scene deformation or a texture change.

Two inputs provide still less geometric redundancy. With TokenGS, PGGS improves all three dataset-mean metrics on both Mip-NeRF 360 and ACID, while AnySplat increases PSNR but retains higher LPIPS. For AnySplat on ACID, SSIM changes from 0.752 to 0.744. On Mip-NeRF 360, its mean LPIPS increases by 19.2%, with increases in seven of the nine scenes. A strong feed-forward prediction may already resolve ambiguities that two input images cannot reliably constrain during adaptation. Continued prior guidance improves on direct optimization under this stress test, while the feed-forward prior can remain preferable for perceptual quality.

C.5 Ablation configurations

The ablation covers the same 24 scenes and three input counts as the posed experiment: 72 settings and 432 reported endpoints across six variants, including 144 feed-forward and full-model endpoints reused from the main experiment. Each input-count mean weights the nine Mip-NeRF 360, eight TNT, four ACID, and three DL3DV scenes equally. The switches below specify the optimized variables and event mechanisms.

Table 18: **Ablation switches.** The rows correspond to the six variants in Table 5; all optimized variants use the same input-view RGB objective.

Variant	Δz	CGD/codes	Explicit residuals	Dynamic events	Event input
Feed-forward	–	–	–	–	–
Direct, fixed count	–	–	attributes	–	–
PLO, fixed count	yes	–	–	–	–
PLO + explicit	yes	–	yes	yes	–
PLO + CGD	yes	yes	–	yes	zero
Full PGGS	yes	yes	–	yes	active

The abbreviated names below correspond, in the same order, to the six configurations in Table 5. The fixed-count variants carry †; PLO + CGD denotes the zero-event-input variant (‡), and full PGGS denotes FF + PLO + CGD + DEA. Their parameterizations are:

1. **Feed-forward:** the initial scene without adaptation.
2. **Direct, fixed count:** explicit Gaussian attributes are optimized with no growth, pruning, or opacity reset.
3. **PLO, fixed count:** only the latent residual is optimized, and frozen decoder outputs are rendered directly, with no discrete events.
4. **PLO + explicit:** latent adaptation is combined with independent per-member attribute residuals and dynamic events; there is no conditional MLP or local code.
5. **PLO + CGD:** the full conditional parameterization and all events remain active, but the ten event-condition channels supplied to the MLP are zero.
6. **Full PGGS:** the complete configuration described in Section 3.

The explicit-residual variant uses additive position, log-scale, rotation-vector, opacity-logit, and color corrections. It receives no code-space perturbation, since that perturbation has no equivalent scale

across physical attribute coordinates. Its inheritance-source score uses position gradients, while CGD uses local-code gradients. The variants consequently differ in noise, parameter dimensions, gradient statistics, and optimizer coordinates. On growth, this variant copies the inherited explicit residual and opacity bias, whereas a new CGD member starts with zero bias.

Fixed-count direct uses the same six Adam parameter groups, learning rates, image sequence, and loss as the dynamic direct baseline, while disabling growth, pruning, and opacity reset together. Its PSNR is higher by 3.15, 3.53, and 3.45 dB for four, six, and eight inputs, with improvements in the other two metrics as well. The improvement suggests that the extra freedom introduced by this event schedule can favor input fitting over novel-view generalization.

Feed-forward and full PGGS endpoints are reused from the main experiment. The remaining four variants were run separately from the same feed-forward initialization and use the same input-image sequence. They reset the random state before optimization, whereas the reused full-model runs inherit their random state from the preceding initialization. The zero-event-input CGD variant and full PGGS consequently use different conditional-decoder initializations, each following its respective random-state history. The zero-event-input CGD variant retains all dynamic events and opacity reset through the separate bias. At the displayed precision, full PGGS ranks first or second in all nine metric–input-count comparisons.

C.6 Adaptation gains and degradation

For each posed backbone, we compute the change in novel-view PSNR relative to its feed-forward initialization for all 24 scenes and three input counts. PSNR is first averaged over the three novel views. Table 19 reports the mean and median changes and the number of negative changes, computed before rounding. These 72 settings comprise three input counts for each of 24 scenes, with shared novel views within each scene. Direct and Direct + Dropout use the dynamic-event configurations in Table 2; fixed-count Direct is the separate ablation in Table 5.

Table 19: Adaptation gains and degradation relative to feed-forward prediction. Each row summarizes 72 scene–input-count settings (24 scenes with 4, 6, and 8 inputs) at the final 1,500-step endpoint. Changes are computed after averaging three novel views per setting. Negative changes count settings with strictly lower PSNR than the corresponding feed-forward prediction, using unrounded values.

Backbone	Method	Mean ΔPSNR (dB) ↑	Median ΔPSNR (dB) ↑	Negative changes / 72 ↓
DepthSplat	Direct	-2.45	-2.00	68/72
	Direct + Dropout	-3.46	-2.53	70/72
	PGGS	+1.65	+1.30	3/72
TokenGS	Direct	+1.09	-0.04	38/72
	Direct + Dropout	+0.21	-0.25	43/72
	PGGS	+2.57	+1.62	1/72
YoNoSplat	Direct	+2.63	+0.70	25/72
	Direct + Dropout	+2.17	+0.61	30/72
	PGGS	+4.26	+1.66	10/72